说明：

本系统实现的主要功能为银行卡号端到端的检测与识别。解决了在自然场景下图像基于CTPN的文本检测的问题和基于CRNN的文本识别的问题。相对于传统的检测识别算法，无需对图片中的文字进行单独的切割，由此整个系统更加高效完整。

通过依赖于TensorFlow的CTPN算法与依赖于Pytorch的CRNN算法

在对有限的原始数据图片进行数据增强之后，对在Linux上配置部署的CRNN的网络通过GPU进行加速训练获得良好的神经网络，再将其迁移Windows系统中，最终实现在Windows10上的成功运行。

目录

[1. 项目实现技术栈 1](#_Toc12984049)

[2. 项目实现思路介绍 1](#_Toc12984050)

[3. 卡号识别的实现步骤和介绍： 2](#_Toc12984051)

[3.1 处理不定长字符定位与识别时的思路： 2](#_Toc12984052)

[3.2 数据增强方式： 2](#_Toc12984053)

[3.3 数据预处理与后处理: 2](#_Toc12984054)

[4. 项目实现流程介绍 3](#_Toc12984055)

[4.1 项目流程图： 3](#_Toc12984056)

[4.2 环境配置搭建： 4](#_Toc12984057)

[4.3 CTPN配置： 5](#_Toc12984058)

[4.4 CTPN检测效果演示： 9](#_Toc12984059)

[4.5 CRNN配置： 12](#_Toc12984060)

[4.6 数据增强： 14](#_Toc12984061)

[4.7 数据集制作： 16](#_Toc12984062)

[4.8 代码编写与优化： 17](#_Toc12984063)

[5. UI实现： 23](#_Toc12984064)

[6. 运行结果： 26](#_Toc12984065)

[7. 项目中卡号定位与卡号识别结果说明 28](#_Toc12984066)

**银行卡号识别项目文档**

## 项目实现技术栈

文本检测算法为CTPN

文本识别算法为CRNN+CTC

UI界面实现为Tkinter

使用软件主要有Ananconda搭建Python运行环境，CUDA作为GUP训练运算平台，IDE为Visual Studio Code。

使用的深度学习框架CTPN为TensorFlow，CRNN为Pytorch。

编程语言主要为Python3.6。部分代码语言为C。

训练环境搭建在Ubuntu 16.04在，检测及识别环境通过编译文件，实现在Windows 10上的正确运行。

## 项目实现思路介绍

本套系统使用CTPN+CRNN进行实现。大体思路为通过CTPN进行银行卡图片中文本信息的位置检测，通过算法获取并切割出银行卡号图片，再将该图片输入到已经训练好的CRNN神经网络中进行识别，最后输出结果给前端UI。

## 卡号识别的实现步骤和介绍：

### 3.1 处理不定长字符定位与识别时的思路：

由于银行卡背景的多样性和复杂性，使用传统图形学算法难以进行降噪处理以及提取出数字的特征信息，因此最终选择使用CRNN进行端到端的数字识别，无需再对单个数字进行切割和识别。

### 3.2 数据增强方式：

包括对图片的仿射变换，增加噪声，图片模糊处理，亮度的调节等

### 3.3 数据预处理与后处理:

首先，将原始的训练图片划分为训练集和验证集。由于原始训练数据图片的正确标签为图片文件名，因此在训练之前，需要使用正则表达式将图片正确的标签进行提取存储。

之后，将分类好的图片数据转化为LMDB文件。数据后处理包括选择出CTPN检测出的银行卡号区域，算法返回图片的边缘处理，PIL格式和OpenCV格式的转变等。

## 项目实现流程介绍

### 4.1 项目流程图：

### 4.2 环境配置搭建：

Anaconda：

为了更好的解决python包管理和Python不同版本的问题，我选择使用Anaconda进行安装搭建总体环境 （可以便捷获取包且对包能够进行管理，同时对环境可以统一管理的发行版本。包含了conda、Python在内的超过180个科学包及其依赖项。）

创建新的Python环境并安装环境所需的各种库，包括opencv, tensorflow, numpy等。

Anaconda Navigator 
File Help 
D ANACONDA NAVIGATOR 
Sign in to Anaconda Cloud 
Home 
Environments 
Learning 
Community 
Documentation 
Developer a log 
Search Environments 
base (root) 
Mlearning 
Tensor 
V-Tensor 
Installed 
Name 
_pytorch_select 
flow select 
Osl- py 
blas 
ca-certiäcates 
cffi 
80 packages available 
Channels 
Description 
Update indexm 
o 
Abseil python common libraries, see 
https://github.com/abseil/abseil-py. 
Read, rewrite, and write python 
nicely 
Certificates for use with other packages. 
Python package for providing mozilla's 
ca bundle. 
Foreign function interface for python 
Search Pack.. 
Version 
c.7.c 
0.7.1 
2019.5.15 
2019.3.9 
Crea te 
Clone 
Import 
Remove 

此处为Windows环境搭建，由于后期CRNN训练需要在Linux环境下进行，因此还需 要再次为Linux搭建环境。

CUDA：

为了在识别以及后期进行训练时更加高效，我们需要使用到计算机的GPU，在对crnn进行训练时，需要单独安装CUDA。（CUDA（Compute Unified Device Architecture），是显卡厂商NVIDIA推出的运算平台。 CUDA™是一种由NVIDIA推出的通用并行计算架构，该架构使GPU能够解决复杂的计算问题。 它包含了CUDA指令集架构（ISA）以及GPU内部的并行计算引擎。 ）

包括CUDA，cudnn，cudatoolkit三部分。

Pytorch：

Pytorch是torch的python版本，是由Facebook开源的神经网络框架。与Tensorflow的静态计算图不同，pytorch的计算图是动态的，可以根据计算需要实时改变计算图。

Installed 
Name 
pytorch_select 
pytorch 
Channels 
Description 
Update indexm 
pytorch 
Version 
x 
Pytorch is an optimized tensor library for 
1.1.o 
deep learning using gpus and cpus_ 

其他库还还包括tensorflow，cython等，在readme文档中已经说明，不依次列举。

### 4.3 CTPN配置：

#### 4.3.1 CTPN介绍：

关于CTPN原理及其实现：

对于复杂场景的文字识别，首先要定位文字的位置，即文字检测。

CTPN是在ECCV 2016提出的一种文字检测算法。CTPN结合CNN与LSTM深度网络，能有效的检测出复杂场景的横向分布的文字。

3x3xt-»ş6D 
v13G16 
(b) 

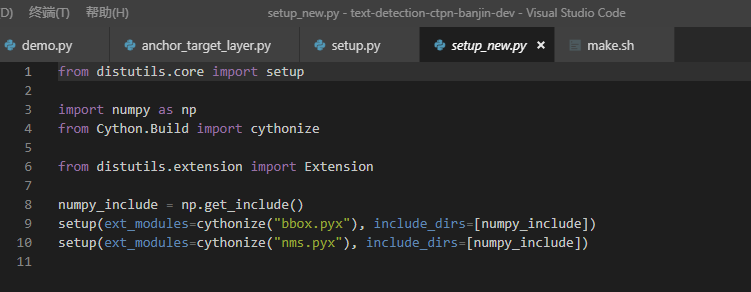
CTPN网络结构

#### 4.3.2 配置流程：

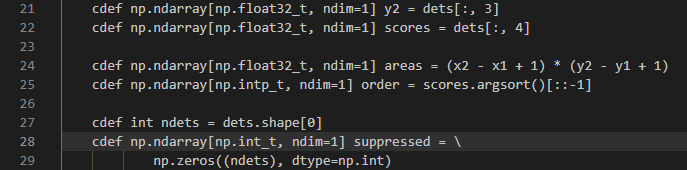
由于参考代码是适用于Linux平台，在这里我对该CTPN网络进行了部分修改以迁移至Windows平台进行运行。

核心是对lib\utils\bbox文件夹中的部分pyx文件使用cyphon编译成pyd文件

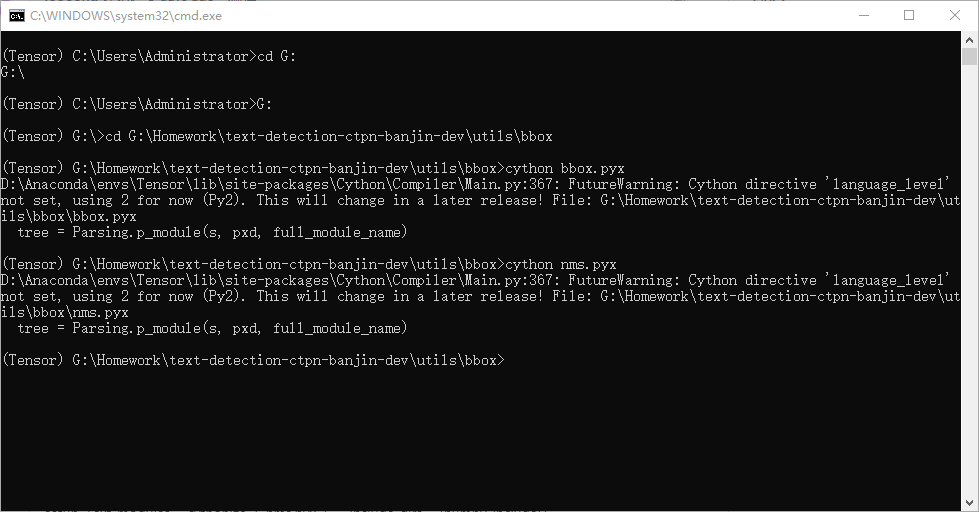
修改编译脚本

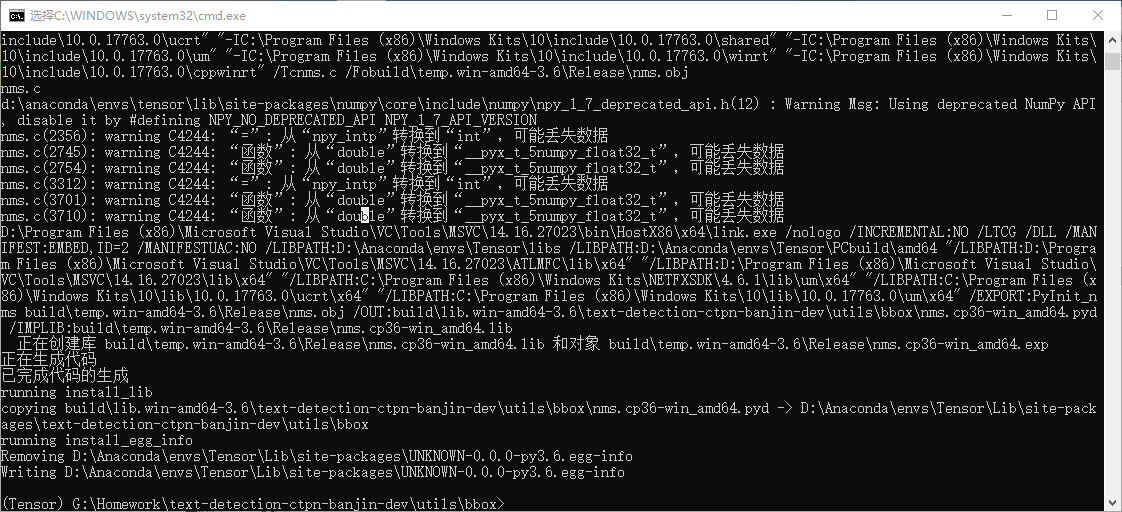


在文件lib \ utils \ bbox \ nms.pyx的第25行进行一些更改“np.int\_t”更改为“np.intp\_t”   
否则出现“ValueError：Buffer dtype mismatch，expected'int\_t' but get 'long在步骤6中“长”。



cd到编译路径



进行编译

Homework 
_pycache 
Banca rd Identifier 
text detection_ctpn utils 
b box 
fuse hiddenOOOOc4ff00000003 
_i nit_.py 
b box.c 
b box.cp36 -wi me 64. pyd 
b box. pyx 
b box_tra nsfor m. py 
ma ke .sh 
n ms.cp36 -wi me 64. pyd 
n ms. pyx 
setu p_new. py 
2019/6/20 23:08 
2019/2/15 0:23 
2019/2/15 0:23 
2019/5/18 22:44 
2019/5/18 22:45 
2019/2/15 0:23 
2019/2/15 0:23 
2019/2/15 0:23 
2019/5/18 22:44 
2019/5/18 22:45 
2019/5/18 22:03 
2019/2/15 0:23 
2019/5/18 21:24 
FUSE HIDDENOO... 
Python File 
Python Extension. 
PYX 
Python File 
Shell Script 
Python Extension. 
PYX 
Python File 
Python File 最终编译出的完整文件

（[Cython](https://link.jianshu.com/?t=http%3A%2F%2Fcython.org%2F)是让Python脚本支持C语言扩展的编译器，Cython能够将Python+C混合编码的.pyx脚本转换为C代码，主要用于优化Python脚本性能或Python调用C函数库。由于Python固有的性能差的问题，用C扩展Python成为提高Python性能常用方法

来自 <<https://www.jianshu.com/p/fc5025094912>> ）

### 4.4 CTPN检测效果演示：

原图

文本检测结果图

检测的文本定位数据

2.txt - 
,0.99963427 

八行分别是检测出的八个文本区域。每两个数据x和y构成区域的一个坐标点，四个坐标定位一个区域，最后一项为获得的分数。

（注：由于CTPN的训练需要大量的文本区域标注数据，此处由于无法获得数据，因此使用的是用他人提供的已经训练好的神经网络）

银行卡号区域的获取：

此处我选择了一个简单的检测方式，即使用检测出的文本长度作为判别依据。由于银行卡号几乎都是卡面上最长的文本区域，因此这种方法基本上是可行的。

检测代码：

def getcardpos(cardposdir): 
maxlen = e 
f 
open(cardposdir, 'r ' ) 
for i in 
strlist = i.split( , 
#print(strlist) 
if len(strlist) 
if (int(str1ist[2]) 
maxlen 
Int(str1ist[2]) 
print(maxlen) 
cardpos 
strlist[:] 
#print(cardpos) 
#print(cardpos) 
strlist.clear() 
print (cardpos) 
return cardpos 
maxlen : 切割代码：def : 
text box detect 
#r_img, f = resize_im(img, scale-scale, max_scale=max_scale) 
scores, boxes, img, cardposdir = get_ctpn_info(img_dir) 
print (cardposdir) 
cardpos = getcardpos(cardposdir) 
#cardpos = C int(x) for x in cardpos 
im = Image. 
image = Image. cv2.COLOR BGR2RGB)) 
supply 
(int(cardpos[2]) 
int(cardpos [e] ) ) / 2e 
image. crop( (int(cardpos [e] ) 
supply, 
int(cardpos[l]), 
region 
region . save( ". / data/ cut/l. jpeg" ) 
print(" save") 
scores, boxes, image, region 
return 
Int(cardpos[2]) + supply, int(cardpos[S]))) 

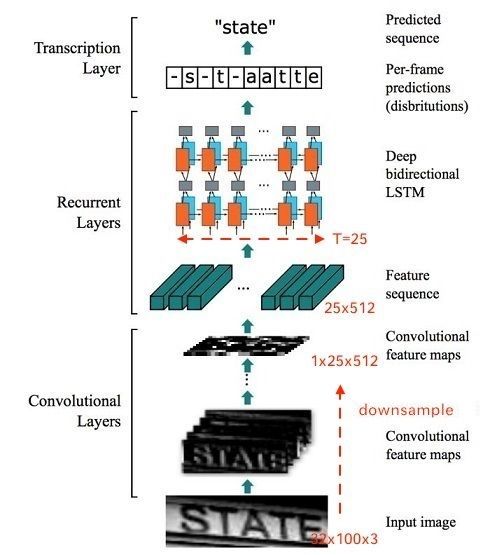
（代码对图片区域进行了误差弥补，以保证获取所有文字区域）

C:\Users\Administrator\AppData\Local\Packages\Microsoft.Office.OneNote_8wekyb3d8bbwe\TempState\msohtmlclip\clip_image011.png输出检测效果：

### 4.5 CRNN配置：

#### 4.5.1 CRNN介绍：

采用的是CNN+RNN+CTC(CRNN+CTC)框架实现文本识别。



CRNN网络结构 <<https://zhuanlan.zhihu.com/p/43534801>>

实现环节

* 首先使用CNN提取图像卷积特征
* 然后LSTM进一步提取图像卷积特征中的序列特征
* 最后引入CTC解决训练时字符无法对齐的问题

要最终实现使用CRNN进行文字识别需要两个步骤，分别是识别与训练。

**训练部分：**

CRNN的训练部分实在Linux上实现的。

步骤大致分为：

环境搭建

数据增强

数据集制作

Crnn网络参数调整

Crnn网络训练

**环境搭建：**

该网络是借助于Pytorch框架实现，使用GPU来加速网络训练。

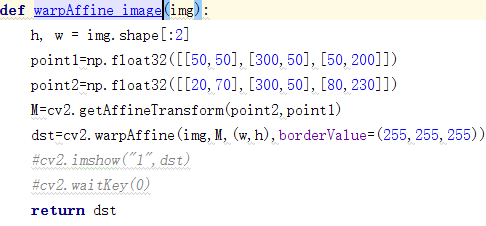
为了便于管理Python版本和包，在Linux上依旧采用了Anaconda。

vercus@vercus-ubuntu: — 
(base) conda activate DL 
conda tist 
(DL) vercus@vercus-ubuntu : -$ 
# packages in environment at 
/ Anaconda/ Anaconda3/envs/DL: 
Version 
# Name 
astroid 
1.5.3 
a.tsinghua.edu . cn/anaconda/pkgs/f ree 
blas 
1.0 
a.tsinghua.edu . cn/anaconda/pkgs/f ree 
certifi 
2016.2.28 
a.tsinghua.edu . cn/anaconda/pkgs/f ree 
1.10.0 
a.tsinghua.edu . cn/anaconda/pkgs/f ree 
cudatoolkit 
8.0 
a.tsinghua.edu . cn/anaconda/pkgs/f ree 
cudnn 
6.9.21 
a.tsinghua.edu . cn/anaconda/pkgs/f ree 
cycler 
9.10.0 
cython 
0.29.10 
easydict 
1.9 
f reetype 
2.5.5 
a.tsinghua.edu . cn/anaconda/pkgs/f ree 
futures 
h5py 
3.1.1 
2.9.0 
Build 
py36_o 
mld 
py36_o 
py36_o 
3 
cuda8 . O 
o 
PYpi_O 
PYpi_O 
PYpi_O 
2 
PYpi_O 
PYpi_O 
Channel 
https : //mirrors . tun 
https : //mirrors . tun 
https : //mirrors . tun 
https : //mirrors. tun 
https : //mirrors . tun 
https : //mirrors . tun 
pypi 
pypi 
pypi 
https : //mirrors . tun 
pypi 
pypi 

图为所创建的DL环境

CUDA，Torch7, fblualib 和 LMDB等安装不做复述

### 4.6 数据增强：

1 图片仿射变换，

使用opencv的getAffineTransform：由三对点计算仿射变换，

cv2.getAffineTransform(point2,point1)

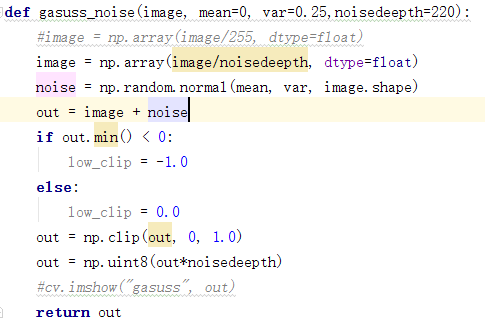
point2：输入图像的三角形顶点坐标。

point1：输出图像的相应的三角形顶点坐标。

C:\Users\Wolf\PycharmProjects\Tensorflow-Resnet-Image-Classification-master\data\train2\_020a_0_warp_1f+5.png效果如下，

C:\Users\Wolf\PycharmProjects\Tensorflow-Resnet-Image-Classification-master\data\train\_020a_0.png

2 图片增加噪声，



将图片以array的形式存储

随机增加噪声

转回原形式

C:\Users\Wolf\PycharmProjects\Tensorflow-Resnet-Image-Classification-master\data\train2\_020a_0noise160.png效果如下，

C:\Users\Wolf\PycharmProjects\Tensorflow-Resnet-Image-Classification-master\data\train\_020a_0.png

3 图片模糊

C:\Users\Wolf\PycharmProjects\Tensorflow-Resnet-Image-Classification-master\data\train\_020a_0.png使用Pillow库的图片模糊，PIL.BUR

C:\Users\Wolf\PycharmProjects\Tensorflow-Resnet-Image-Classification-master\data\train3\_020a_0.pngmohu.png效果如下，

4 改变亮度，

使用Pillow库的改变亮度，

图片变亮，PIL.point(lambda p: p \* 1.4),

效果如下，

C:\Users\Wolf\PycharmProjects\Tensorflow-Resnet-Image-Classification-master\data\train2\_020a_0.pngliang0.6.pngC:\Users\Wolf\PycharmProjects\Tensorflow-Resnet-Image-Classification-master\data\train\_020a_0.png

C:\Users\Wolf\PycharmProjects\Tensorflow-Resnet-Image-Classification-master\data\train\_020a_0.png图片变暗，PIL.point(lambda p: p \* 0.8)

C:\Users\Wolf\PycharmProjects\Tensorflow-Resnet-Image-Classification-master\data\train2\_020a_0.pngmohu.png效果如下，

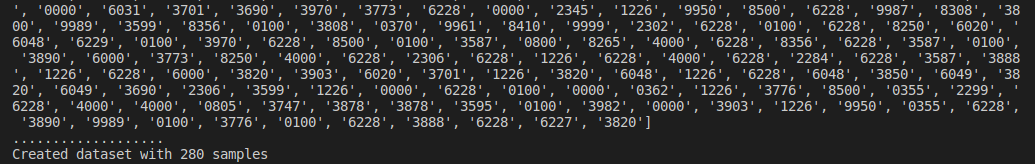
### 4.7 数据集制作：

数据增强之后，需要将图片制作成LMDB数据集，便于后续数据的读取处理。

由于使用的数据图片其正确标签是其文件名，因此需要将图片的标签再单独提取。

def getrightword(path)• 
— path.split( 
text — 
re. ) 
#pattern 
— re.match(r' (\dl . , text) .group() 
textl — 
— , textl) .group() 
text2 — 
print (text2)l 
return text2 使用正则表达式来提取图片正确的数字标签

加载图片路径，使用其中的图片分别制作训练数据集和验证数据集

获得测试结果

crnn.pytorch 
data.mdb 
Imdb 
train Imdb 
lock.mdb 生成的lmdb数据集

### if [ (p, read text(p. replace(• jpg' , • . txt' for p in imagePathList] name maln ##lmdb #outputPath •Imdb/train Imdb' outputPath = Imdb/val Imdb' #path "train/data/train/* . png " path = "train/data/valtrain/*. png" #pathl "train/data/train/*.jpg ' pathl = "train/data/valtrain/* . j pg " imagePathList = glob.glob(path) imagePathList glob . glob(pathl) print( imgLabelLists — ' , len(imagePathList) , for p in imagePathList: try print(p) #imgLabelLists.append((p, read text(p. replace(• .png' , imgLabelLists. append( (p, getrightword(p))) except : continue # imgLabelList ##sort by lebelList . txt')))) 4.8 代码编写与优化：

#### 4.8.1 Crnn网络参数调整

parser . add argument( ' 
Ir' , type—float, default=€ . *01, 学习率调整。在初次训练时，由于学习率过大，导致网络无法收敛。调节为0.0001后可正常收敛。

lument(' -workers' , 
type—int, help; ' number of da 
-batchsize' , 
lument( ' 
type—int, default=128, hel BatchSize调整。BatchSize需要选择为2的次方数，过小可能导致局部网络梯度方向与整体相反。过大则会降低网络的训练效率。

. add 
argument(' -trainRoot' , 
parser 
. add 
argument(' -valRoot' , 
parser 
help; ' path to dataset ' , default; ' (Homework/crnn. 
help; 'path to dataset ' /HomeWork/crnn.px, 训练集和验证集路径指定

argument( -ne*ch', type—int, help;' number of 
ieru): epoch -> iter 训练次数，决定整个数据集需下训练多少次

TrainBatch函数对Batch进行训练，并返回损失。

def ltrainBatch (net, criterion, optimizer, 
train iter): 
data = train iter. next() 
cpu images, cpu texts = data 
#bytes(cpu texts, encoding 
#print(cpu texts) 
"utf8") 
re.match( ' ,cpu texts) .group() 
#textl 
, textl) .group() 
#cpu texts 
#print(cpu texts) 
#print( "cpu texts") 
#print(type(cpu texts[o])) 
batch 
size = cpu images.size(o) 
utils. loadData(image, cpu images) 
t, I — converter . encode(cpu texts) 
utils. loadData(text, t) 
utils. loadData(length, l) 
preds = crnn(image) 
preds 
size = Variable(torch.IntTensor([preds.size(O)] * batch size)) 
— criterion(preds, text, preds size, length) / batch size 
cost — 
crnn.zero grad() 
cost . backward() 
optimizer. step( ) 
return cost 

parser 
parser 
parser 
parser 
. add 
. add 
. add 
. add 
argument( ' 
- -displaylrlterval 
type=int, default=4, help= 
argument( test disp' , type—int, default=10, help;' Nu 
argument(' -vallnterval' , type—int, default=4, help;' Int 
argument(' -savelnterval' , 
type=int, default=10, help='l 

显示参数为displayInterval

验证参数为valInterval

保存参数为saveInterval

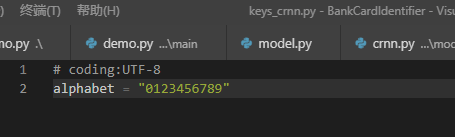
定期对训练模型进行保存，并决定训练效果在终端的显示间隔以及对训练模型进行验证的间隔。

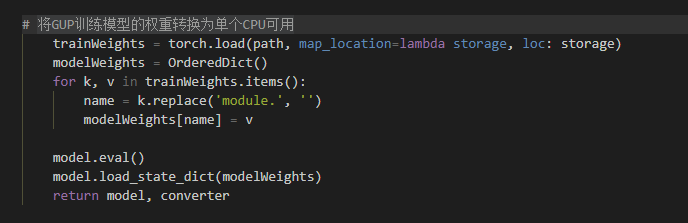
for epoch in range(opt .nepoch) • 
train iter = iter(train loader) 
print( "epoch") 
while i < ten (train loader): 
for p in crnn. parameters(): 
p. requires grad = True 
crnn.train() 
cost = trainBatch(crnn, criterion, 
#print(cost) 
loss avg. add(cost) 
if i % opt. displaylnterval O 
optimizer , 
train iter) 
print( ' Loss: % 
(epoch, opt. nepoch, i, len(train loader), 
loss avg. reset() 
if i % opt.vallnterval € 
val(crnn, test dataset, criterion) 
# do checkpointing 
#if epoch % opt.savelnterval 
if epoch % opt. nepoch/4 O 
print( " save" + str(epoch)) 
torch . save( 
crnn.state dict(), 
' {€}/netCRNN {1} {2} .pth 
loss avg.val())) 
. format(opt.expr dir, 
epoch , 
i)) （多少次迭代在执行一次）

22- -- -66- 
-99-- 
-99- 
_ _ 99 _ 
88- 11- 
-33- 
-88- -ΘΘ- 
-222--- -33- 
-8- 
-66- 
loss: Θ. 
Test 
_ 99 _ _ 
-1- 
- -ΘΘ- 
- -ΘΘ- 
ΘΘ2222 , 
--5- 
-ΘΘΘ- 
8265 
9999 
9919 
81Θ 
38Θ8 
23Θ5 
Θ8Θ5 
6ΘΘΘ 
. 645833 
8265 
9999 
9949 
81Θ 
38Θ8 
23Θ6 
Θ8Θ5 
6ΘΘΘ 
accuray : 对训练中验证结果截图，如当前准确率为0.65

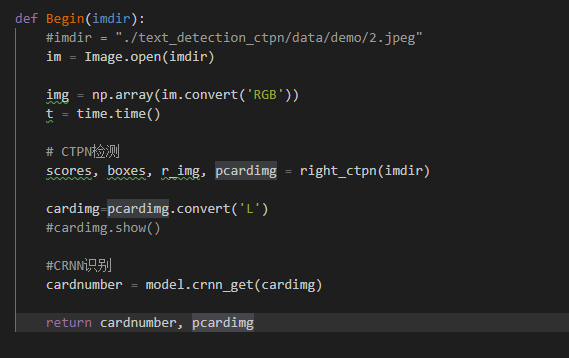
crnn.pytorch 
expr 
netCRNN o_21.pth 
netCRNN 40 21. 
pth¯ 
netCRNN 10 21. 
p th 
netCRNN 20 21. 
p th 
netCRNN 30 21. 
p th 生成的模型

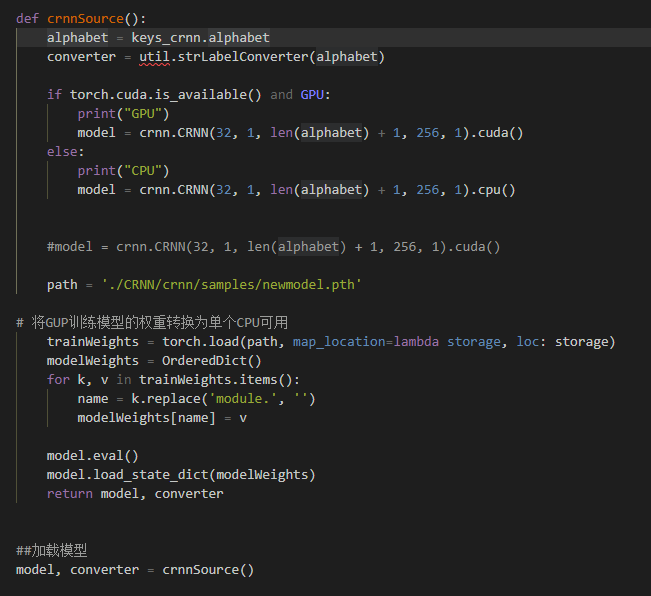
#### 4.8.2 识别部分：

定义分类标签表

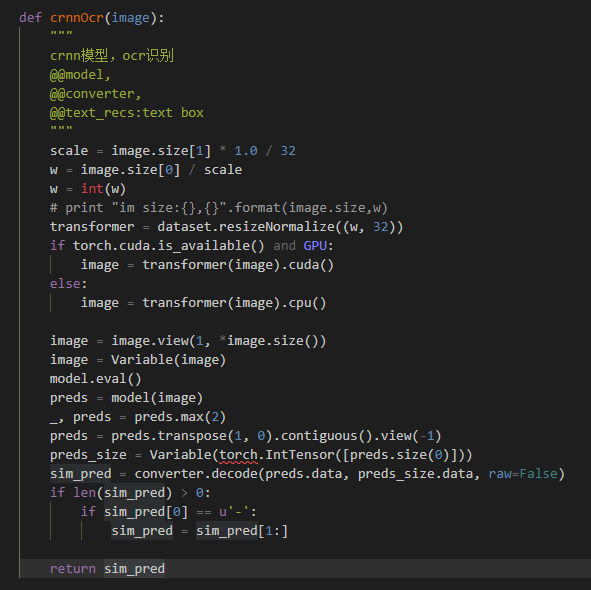
将GUP训练模型的权重提取转变为单个CPU可用（因为模型是使用GPU训练的，但是我是使用CPU运行的）

在Begin()中调用CTPN检测接口right\_ctpn()和CRNN识别接口crnn\_get()

right\_ctpn() 对银行卡号区域进行检测并返回输出图片。crnn\_get() 对输入的卡号图片进行识别

crnn模型加载方法：

crnn识别方法：



#### 4.8.3 识别结果：

返回的CTPN检测结果

C:\Users\Administrator\AppData\Local\Packages\Microsoft.Office.OneNote_8wekyb3d8bbwe\TempState\msohtmlclip\clip_image017.png输出CRNN识别结果

6222601310012850053

## UI实现：

UI使用了Python GUI 库**Tkinter。**Python 使用 Tkinter 可以快速的创建 GUI 应用程序。

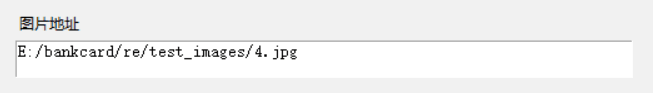
Tkinter 是 Python 的标准 GUI 库。Python 使用 Tkinter 可以快速的创建 GUI 应用程序。

由于 Tkinter 是内置到 python 的安装包中、只要安装好 Python 之后就能 import Tkinter 库、而且 IDLE 也是用 Tkinter 编写而成、对于简单的图形界面 Tkinter能够应付自如。

Opencv

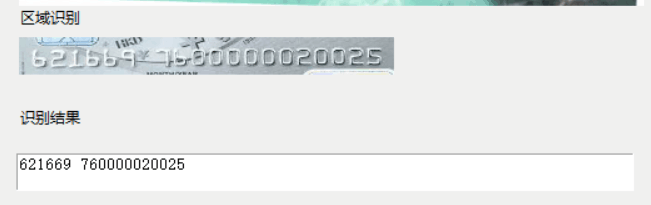
OpenCV是一个基于BSD许可（开源）发行的跨平台计算机视觉库，可以运行在Linux、Windows、Android和Mac OS操作系统上。它轻量级而且高效——由一系列 C 函数和少量 C++ 类构成，同时提供了Python、Ruby、MATLAB等语言的接口，实现了图像处理和计算机视觉方面的很多通用算法。

前端代码实现

button\_load（上传文件）函数选择要进行识别的文件，选择完图片后将文件的地址传入区域识别函数，进行图片字符区域的定位。

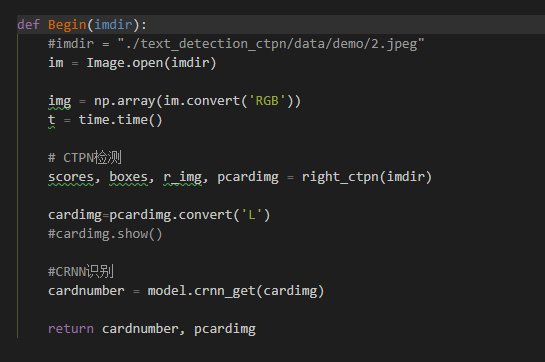
同时将上传的图片展示在窗口，由于图片尺寸不统一，使用opencv里面的resize函数将上传的图片进行尺寸调整，长度超过500的将长度调整到500，在检测宽度将宽度超过300的调整为300之后显示在窗口的指定位置

Button\_orc 函数将图片地址传到识别区域模块，并将识别后的图片获取到图片的地址之后加载到区域识别栏，并将获取到的识别结果输出到label\_out区域

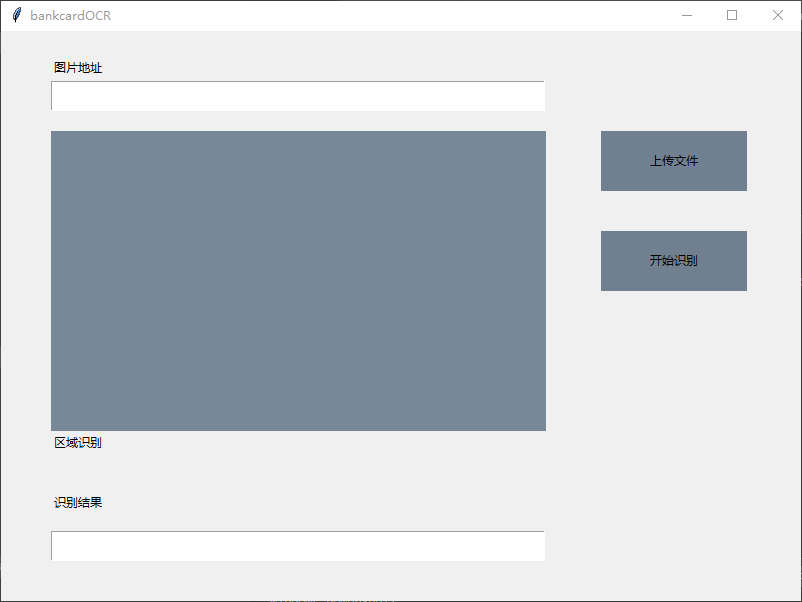


在编写UI前，需要将CTPN，CRNN以及UI界面进行整体的整合。

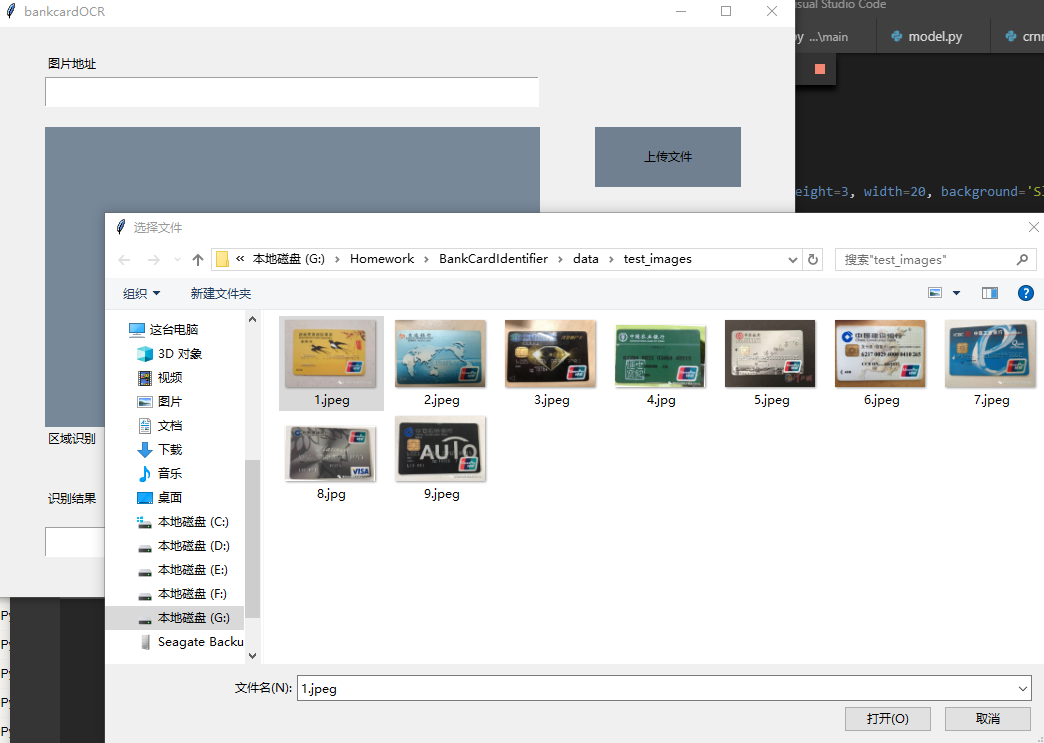
首先将CRNN和CTPN整合到Begin()方法中：

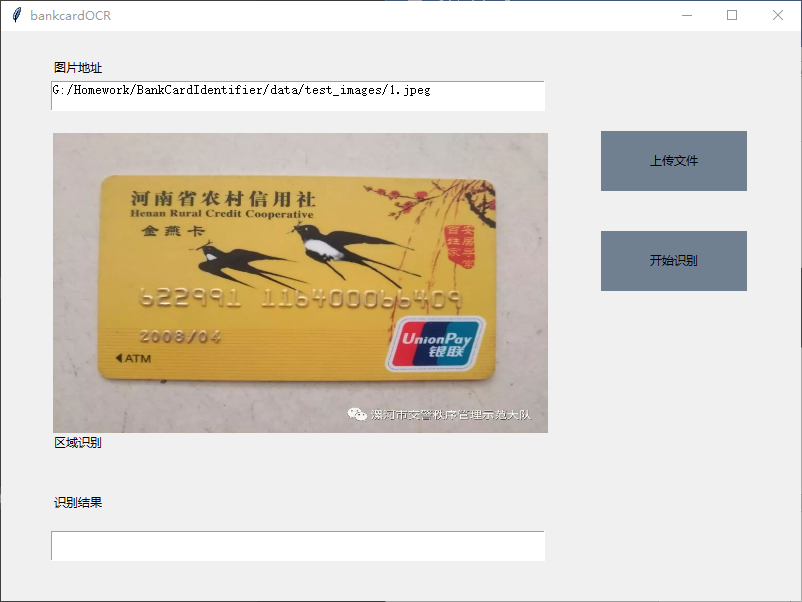


在前端UI点击方法中，调用识别方法，将图片识别结果进行返回。

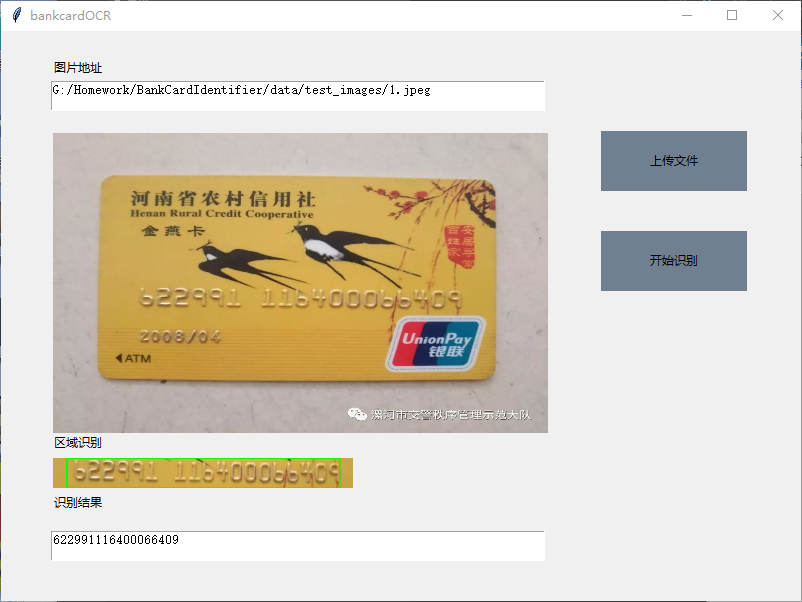
编写运行UI界面如图：

## 运行结果：

上传测试图片文件：获取识别图片文件路径以及显示图片

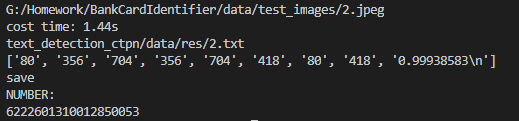


点击开始识别，输出文本检测区域，以及文本识别结果



## 项目中卡号定位与卡号识别结果说明

在控制台打印输出信息：



检测图片路径：

G:/Homework/BankCardIdentifier/data/test\_images/2.jpeg

花费时间：

cost time: 1.44s

CTPN检测结果输出存储路径：

text\_detection\_ctpn/data/res/2.txt

银行卡号图片区域坐标及得分：

['80', '356', '704', '356', '704', '418', '80', '418', '0.99938583\n']

CRNN识别结果：

6222601310012850053