基于目标检测与语义分割的车牌识别

曾子涵 3190103963

摘要 本文将通过 Python 中的 tensorflow 库函数搭建神经网络来实现车牌的识别。本文将会通过记录我逐步实现车牌识别处理的过程，来梳理一些常用的神经网络与图像处理的方法。在本次实验中，分别使用了 Fast-RCNN 网络于 Unet 语义分割网络对车牌进行对位，完成倾斜校正后通过数据增强的 CNN 神经网络实现字符的识别。

关键词 车牌识别，Fast-rcnn，Unet，数据增强，CNN

License plate recognition based on target detection and semantic segmentation

曾子涵 3190103963

Abstract This paper will build a neural network through the tensorflow library function in Python to realize license plate recognition. This paper will sort out some common neural networks and image processing methods by recording the process of license plate recognition. In this experiment, fast RCNN network is used to align the license plate in UNET semantic segmentation network. After tilt correction, character recognition is realized by data enhanced CNN neural network.

Key words License plate recognition, fast RCNN, UNET, data enhancement, CNN

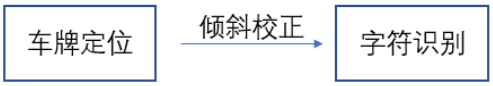
1 引言

对于目前的社会市场而言，车牌识别技术已经非常成熟了，应用也很广泛，从停车场到各种收费站点，车牌识别系统为我们的生活提供了很大的便利。然而我在网上查询了不少的资料发现，尽管车牌识别系统的技术相当成熟，但缺少数据集并且缺少与之相关教程，为广大学习机器学习的同学带来了很多的困扰。其中特别是缺少在图像中定位车牌的具体方法，或者其方法本身就不具有可实现性。因此，本文会根据我对车牌识别的理解和思路，利用开源数据库实现车牌的识别。

2 设计思路

2.1 任务拆分

这个人物的核心是通过一张车牌的照片来实现对其中车牌信息的提取。很自然的这个任务能够被分为两大部分，一是在照片中提取车牌识别的准确位置，二是对车牌进行字符识别的处理，而二者之间通过倾斜透视矫正相互串联。



核心思路

2.2 车牌定位

目前，在百度上搜索车牌识别，大部分教程在车牌定位中使用了传统的图像算法，包括但不限于通过颜色确定车牌位置[1]，通过二值化，边缘检测等办法检测出原图中的车牌号的位置[2]，或是将二者结合办法来实

[1]https://www.cnblogs.com/kekexxr/p/11574589.html

[2] https://zhuanlan.zhihu.com/p/98877416

现[3]。然而很显而易见的是，通过这种纯粹的图像处理方法是非常容易受到光线或是外界环境的干扰从而出现误判，只能在特定场合和条件下使用，作为一般的车牌识别算法，显然可行性是不高的。

那么我们能否使用神经网络来实现车牌位置的判断呢？答案是可行的。在一份使用了颜色定位的教程中，作者提到可以使用目标识别算法来实现车牌的第一次定位。因此我选了 Fast-Rcnn 作为第一次尝试的模型。

2. 3 倾斜校正

倾斜校正的算法比较复杂，所幸的是 python 中为我们提供了倾斜校正的库函数，而我们需要做的是确定车牌的四个顶点即可。在识别四个顶点的过程中，可能仍然还会用到传统的图像处理方法。

2. 4 字符识别

在字符识别的过程中，在百度上查阅到的大多数识别方法都是直接使用了字符切割+单字符识别的方式完成训练的。显然，由于车牌具有形状规则，字符位置明确的特点，因此只要我们能够精确的提取出车牌的位置，就可以很精准的进行字符切割从而识别。然而在图像倾斜校正中，我们几乎不可避免地需要用到传统图像算法，因此这种算法似乎并非最优解。在查询相关文献后，我找到了一种更适合车牌识别的模型：RNN 神经网络。只是一种可以以一张图片为输入，直接输出 7 位车牌信息的算法，显然效果会更加优秀。

3　使用 Fast-Rcnn 实现车牌定位

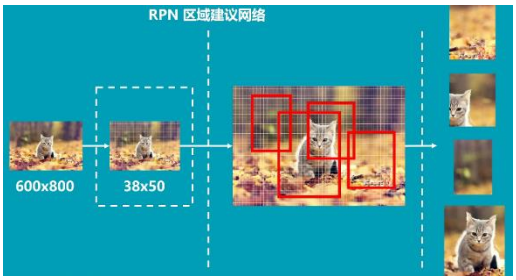
Fast-Rcnn 是一种目标检测算法，在本次项目中，我将会使用这种算法，对车牌的位置进行确定，辅助之后的车牌判断操作。本次实验中所有 Fast-Rcnn 的相关内容参考了于 B 站 up 主 Bubbliiiing 的目标检测教程。

3. 1 什么是 Fast-Rcnn

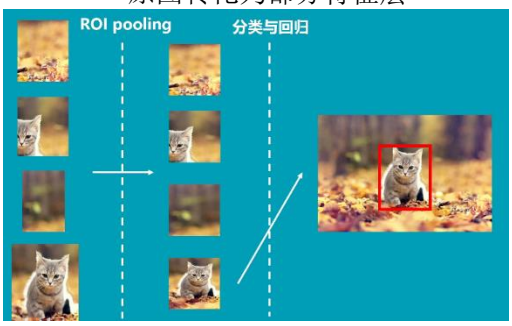
图片中是一只猫咪，人类可以一眼框出猫咪的具体位置，如下图所示。

那么计算机时如何判断出小猫的具体位置，画出这个框的位置呢？其中使用到的一种可行的方法，就是 Fast-RCNN 算法。Fast-Rcnn 是 RCNN 算法的一种变形，同时也是 RCNN 的一种优化形式。当使用 Fast-RCNN 何实现小猫位置判断的时候，照片首先会被重新调整到一个标准大小，下面以 600*800 举例。随后这张图片会被传入到主干特征提取网络中，获得一个 38*50 的特征层，相当于图片被切分成为了 38*50 的先验框。随后图片会传入到名为 RPN 的区域建议网络，这个网络会提供这些先验框的调整参数以及框种是否包含物体的信息。此时我们得到的就是建议框，利用建议框我们可以截取出不同的小图片，这些小图片随后传入到 ROI pooling 层并被重新调整到同样的大小，然后利用分类与回归网络判断建议框中是否包含有目标图像，并对建议框进行调整，而调整的最终结果就是调整的最终结果。

[3]https://blog.csdn.net/GK_2014/article/details/84779



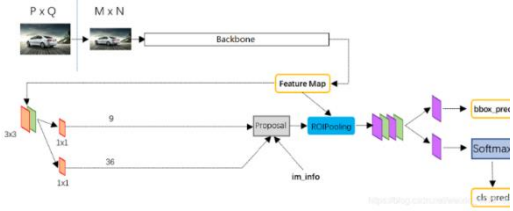
原图转化为部分特征层



部分特征层进入分类与回归网络

总结来说，如果要搭建一个 Fast-RCNN 网络，我们需要做的主要工作一共包括三步，

分别是搭建主干特征提取网络，搭建获得建议框的网络，搭建 ROIPooling 层。总的流程如下：

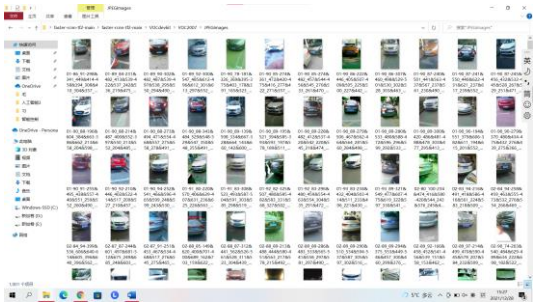


3. 2 数据与数据预处理

想要训练一份车牌定位的网络，首先需要做的是收集一份车牌标定的数据。经过查询了网上的很多资料后，我只找到了一份来自中科大的开源车牌数据，其项目开源地址是 https://github.com/detectRecog/CCPD。这份数据库是在合肥市由停车场工作人员手动采集并标注的，由将近 28 万张图片，其种不同图片的分类如下：

类型	图片数	说明
ccpd_base	199998	正常车牌
ccpd_challenge	10006	比较有挑战性的车牌
ccpd_db	20001	光线较暗或较亮
ccpd_fn	19999	距离摄像头较远或较近
ccpd_np	3036	没上牌的新车
ccpd_rotate	9998	水平倾斜20-50°，垂直倾斜-10-10°
ccpd_tilt	10000	水平倾斜15-45°，垂直倾斜15-45°
ccpd_weather	9999	雨天、雪天或者雾天的车牌
总共：283037张车牌图像		

这份数据集的数据量非常的大，高密度压缩以后都有 20G，因此我选择解压了 ccpd_base 文件并选择前 1000 张照片作为测试，其中部分图片如下：



训练集种的图片

如果我们需要对这些照片进行训练，我们还需要其中车牌照片的位置信息。中科大的这份数据集是没有标注文件的，所有的信息都被保存在了图片的名称上，例如:025-95_113-154&383_386&473-386&473_177&454_154&383_363&402-0_0_22_27_27_33_16-37-15.jpg 这张照片，车牌信息由分隔符'-'分为几个部分:

1) 025 为区域,

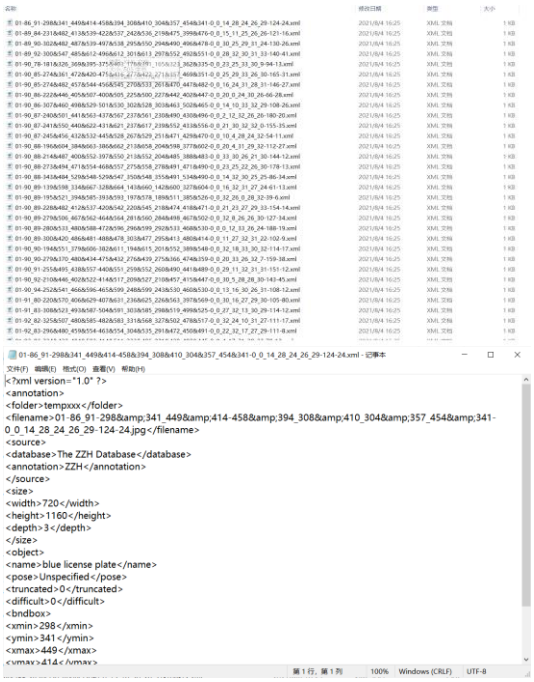
2) 95_113 对应两个角度，水平 95°，竖直 113°

3) 154&383_386&473 对应边界框坐标:左上(154, 383)，右下(386, 473)

4) 386&473_177&454_154&383_363&402 对应四个角点坐标

5) 0_0_22_27_27_33_16 为车牌号码 映射关系如下：第一个为省份 0 对应省份字典皖，后面的为字母和文字，查看 ads 字典.如 0 为 A 等等。

在 Fast-RCNN 的训练过程中，一般会选择 xml 文件作为标签文件，因此，我写了一个用来解析图像文件名称的 python 程序并生成对应的 xml 文件，最终的效果如下。



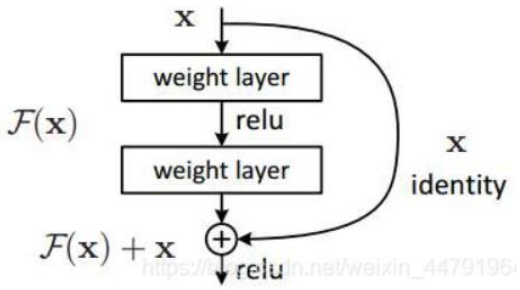
至此，我们数据预处理过程全部完成。

3.3 Fast-RCNN 网络搭建

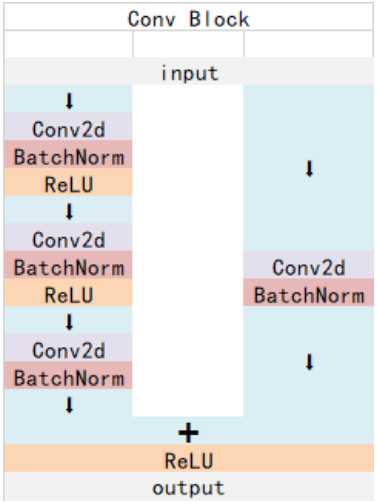
前面提到，Fast-RCNN 网络分为三个核心部分，下面将介绍三部分的搭建。本次代码的搭建过程大部分都参照了 up 主 Bubbliiiing 的这份 B 站教程。

3.3.1 主干特征提取网络

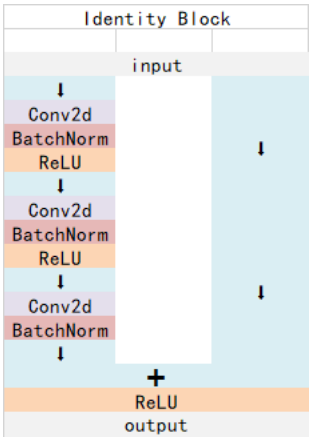
在主干特征提取网络种可以选择的有很多，常见的有 VGG，Resnet，Xception。在本次实验中我选择的是 Resnet 残差网络，残差网络指的是将靠前若干层的某一层数据输出直接跳过多层引入到后面数据层的输入部分，这意味着意味着后面的特征层的内容会有一部分是由其前面的某一层线性贡献的，其常见的结构如下。



而 ResNet50 残差网络由基本的两个功能块构成，分别是 Conv Block 和 Identity Block，其中 Conv Block 输入和输出的维度是不一样的，所以不能连续串联，它的作用是改变网络的维度；Identity Block 输入维度和输出维度相同，可以串联，用于加深网络的。

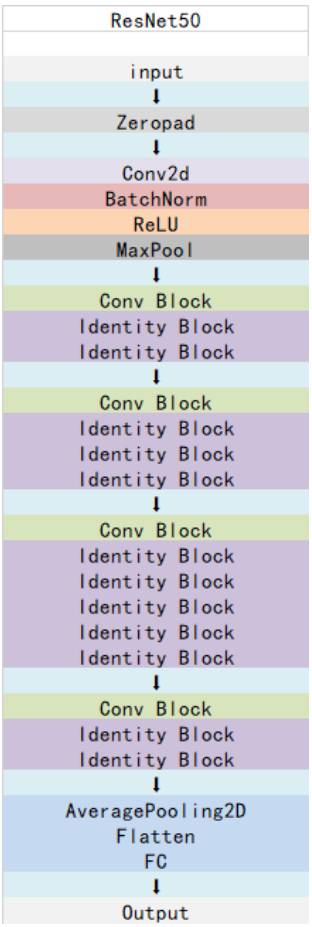


Conv Block 结构



Identity Block 结构

而一个完整的 ResNet50 网络形式如下：



在经过了这样一个 resnet50 网络后，我们输入的 600x600x3 的图片最终会变为 38x38x1024 的共享特征层。

3.3.2 RPN 建议网络

我们对于上面得到的 38x38x3 的特征层进行一个 3*3 的 2D 卷积进行特征整合，在分别进行一个 9 通道（9 x 1）1*1 的卷积和 36 通道（9 x4）的 1*1 的卷积，分别用来预测当前框内是否存在目标物体以及建议框的变化位置。

3.3.3 RoiPooling 层

在通过 RNP 建议网络得到建议框后，我们会利用建议框的结果，在 RoiPooling 层中对第一步得到的 38*38*1024 的共享特征层进行截取，resize 与池化操作，并再一次进行的对 num_classes 的全连接和 (num_classes-1)x4 全连接，这样就可以得到建议框的物体类别和进一步精细调整。

3.3.4 小结

总的来说，Fast-RCNN 对目标的判断框选的三个步骤的作用分别是对图片特征信息的提取，对包含物品范围的粗略调整，以及最终的精细调整三个步骤。

3. 4 训练与预测结果

因为本机电脑的显卡为 1060，在跑完一轮所需要的时间为十分钟，因此训练部分我拜托了一位有台式机且配有 RTX3060ti 的同学帮我完成了数据训练。最终训练得到了一个在训练集上 loss 为 0.0691，验证集上 loss 为 0.0675 的模型，测试结果如下：

3. 5 问题与分析

尽管这个模型在某些情况下可以运行的很好，然而我发现当车牌存在倾斜角度过大的问题时（如下图），下一步车牌的矫正会存在很大的困难。如果想要对其进行倾斜矫

正，我最初的设想是通过颜色过滤筛选的方式配合 cv2 的边缘检测算法提取四个焦点，进而进行矫正。但当设想付诸实践时我发现这种图像算法存在非常多的问题，首先就是由于光线条件的不同，使得色块提取的方法存在很大误差，而当倾斜角度过大时，图片中会包含很多无效信息，但又会被边缘检测检测到，使得对车牌四个角点的确定带来了重重困难。

正如上图所示，如果严格按照 rcnn 分割的结果很容易丢失车牌信息，如果稍作放大处理，就容易有其他干扰信息影响后续倾斜校正的过程

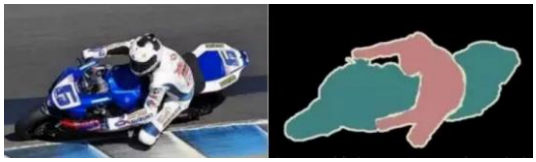
3.6 问题解决

当我对这个问题束手无策之时，我在这位 B 站 UP 主的其它视频里获得了灵感。在其它视频中，up 主介绍了另一种类似目标识别却又不同的算法：图像语义分割算法。这个算法对于车牌识别的定位有很大的帮助。

4 语义分割算法的实现

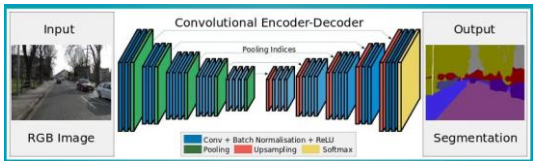
4.1 语义分割算法

语义分割算法是一种以像素点为基础的识别算法，会将像素点按照图像中所表达的不同语义来进行分组与切割。如果说目标检测算法可以告诉我们左边图片中有一车一人，而语义分割算法就可以精确的告诉我们照片中哪个像素点是车，哪个像素点是人。



如果可以将语义分割算法应用在车牌识别中，那么之前遇到的角点提取困难的问题就不复存在了，我们可以利用语义分割算法得到的结果来很轻松的获得车牌的四个顶点，进而获得倾斜校正等相关数据。

那么语义分割算法是如何实现的呢？一般来说其流程如下：



我们需要很多的卷积网络对照片进行特征提取，然后再对这些特征进行反卷积恢复到原来的图像。接下来我会尝试使用 Unet 网络实现车牌的语义分割算法

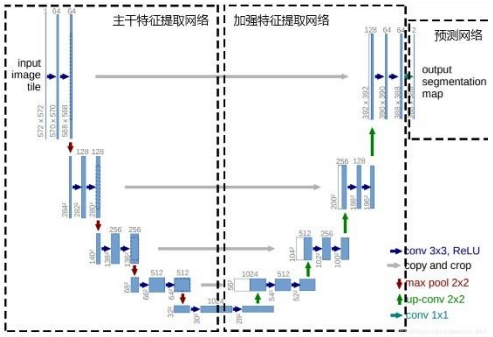
4.2 Unet 语义分割

Unet 算法最早发表在 2015 年的 MICCAI 上，短短几年时间就获得了巨大的应用推广。Unet 可以分为三个部分，如下图所示：

第一部分是主干特征提取部分，我们可以利用主干部分获得一个个的特征层。利用主干特征提取部分我们可以获得五个初步有效特征层。

第二部分是加强特征提取部分，我们可以利用主干部分获取到的五个初步有效特征层进行上采样，并且进行特征融合，获得一个最终的，融合了所有特征的有效特征层。

第三部分是预测部分，我们会利用最终获得的最后一个有效特征层对每一个特征点进行分类，相当于对每一个像素点进行分类。



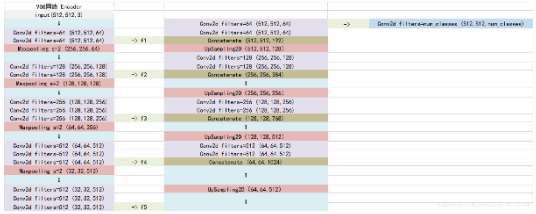
在主干特征提取网络部分，常用的是VGG16，结构如下：

VGG网络 Encoder	
input (512, 512, 3)	
⬇	
Conv2d filters=64 (512, 512, 64)	
Conv2d filters=64 (512, 512, 64)	-> f1
Maxpooling s=2 (256, 256, 64)	
⬇	
Conv2d filters=128 (256, 256, 128)	
Conv2d filters=128 (256, 256, 128)	-> f2
Maxpooling s=2 (128, 128, 128)	
⬇	
Conv2d filters=256 (128, 128, 256)	
Conv2d filters=256 (128, 128, 256)	
Conv2d filters=256 (128, 128, 256)	-> f3
Maxpooling s=2 (64, 64, 256)	
⬇	
Conv2d filters=512 (64, 64, 512)	
Conv2d filters=512 (64, 64, 512)	
Conv2d filters=512 (64, 64, 512)	-> f4
Maxpooling s=2 (32, 32, 512)	
⬇	
Conv2d filters=512 (32, 32, 512)	
Conv2d filters=512 (32, 32, 512)	
Conv2d filters=512 (32, 32, 512)	-> f5

在 VGG16 中我们可以得到五个特征层，我们需要对这五个特征层进行特征融合。特征融合的目的,是把从图像中提取的特征,合并成一个比输入特征更具有判别能力的特征。随着层数的加深，其分辨率更低，但语义性更强，因此结合好低层和高层网络才可以实现对图片更好的分割。

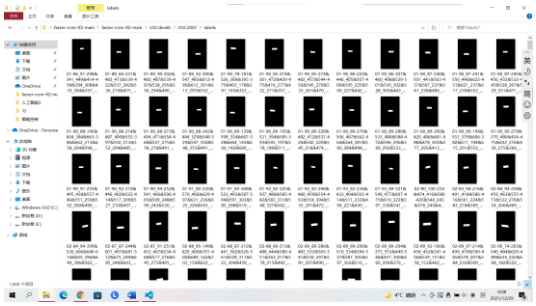
最后一部分是通过特征获取结果，只需要将通道数调整到分类数目即可，在车牌识别中调整为 1 即可。

完整的结构图如下：



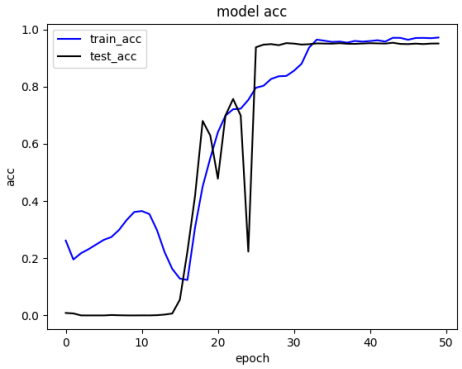
4. 3 数据预处理

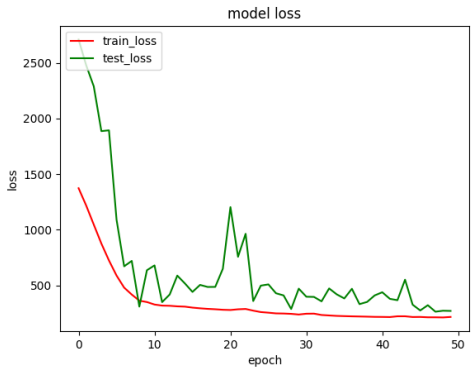
在 Unet 训练网络中，我们需要的是车牌位置的像素信息。为了获取训练文件，我修改了之前的解析代码，将车牌对应的像素值标为 255，其最终的效果如下：



4. 4 训练与测试结果

训练发现 Unet 网络训练效率很高，下面是我花了一个小时训练了 50 轮迭代的效果。

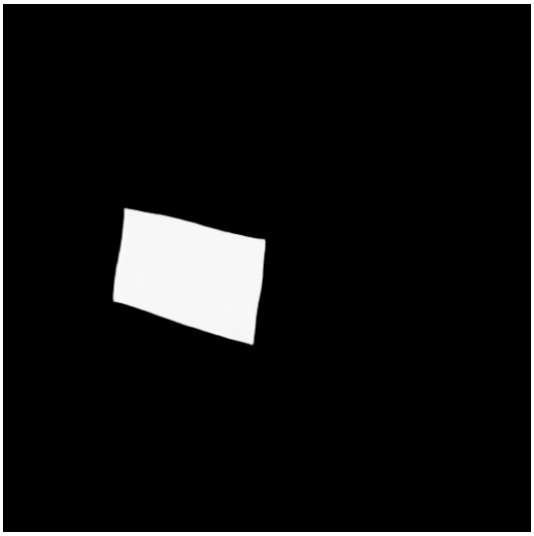




可以看到准确率在 25 轮过后就基本稳定。之后用实际照片展示效果。

输入图片：

输出图片：

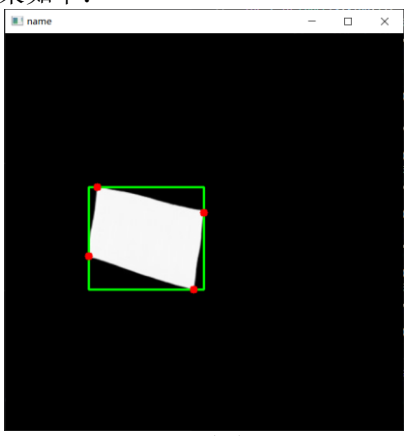


可以看到，输出的图片效果非常之好，利用这张图片可以很方便的对图像的四个角点做定位并提取其中的车牌信息。

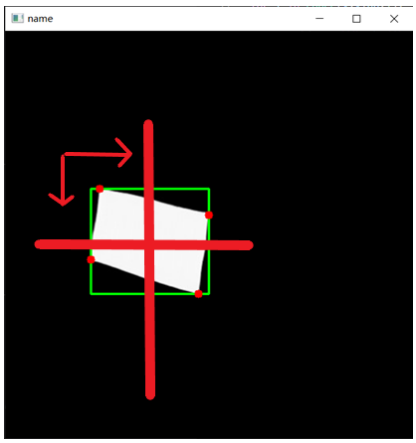
5　车牌切取与倾斜校正

5. 1 车牌角点的提取

在实验中想要获取角点信息，我的想法是首先通过 cv2.boundingRect 函数获取车牌的最小外接矩形。此时可以将将矩形想象为四部分构成，从每一个部分的顶点出发，往外侧寻找是否有交点，最终寻找到的效果如下：



找到的四个角点的位置



寻找角点的过程示意图

5. 2 倾斜校正

确定了车牌四个角点的位置以后，我们可以利用透视变换的原理将其矫正为 240*80 的矩阵。最终效果如下图所示：

6 车牌字符识别

车牌字符识别一直以来有两种方法，即依赖于字符分割的单字符算法以及以车牌为整体输入，然后输出七个字符的卷积神经网络。显而易见的是字符分割算法非常依赖于图像的预处理等性能，其抗干扰能力非常的差，本着将一切交给神经网络的想法，我决定搭建以车牌整体为输入，七个字符为输出的卷积神经网络。

6. 1 网络搭建

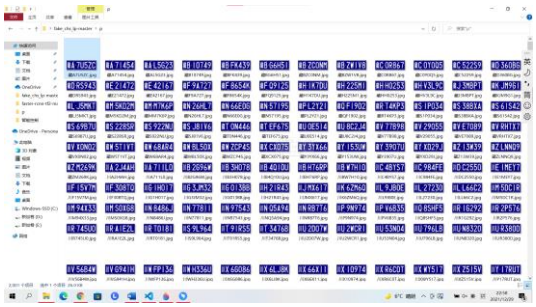
相比较于 Fast-RCNN 和 Unet 的复杂神经网络，我搭建的车牌识别神经网络结构就比较简单了，由四个基本单元构成，每个基本单元包含了两个二维卷积和一个池化层，并且为了过拟合都添加了一个遗忘层。

6. 2 数据预处理

车牌识别真正的难点在于数据的寻找。之前的车牌识别的过程中我都使用了中科大的开源数据集，这个数据集中虽然包含了车牌的信息，但是却存在一个很大的问题，即由于采集地是来自合肥的原因，几乎所有的车牌都是皖 A 的，导致这份数据集几乎不可用。而网上关于整块车牌的数据集的信息几乎没有。

幸运的是，我在网上找到了一个开源的伪造中国车牌的项目（https://github.com/ufownl/fake_chs_lp），可以输出任意的车牌。其生成的蓝牌如下：

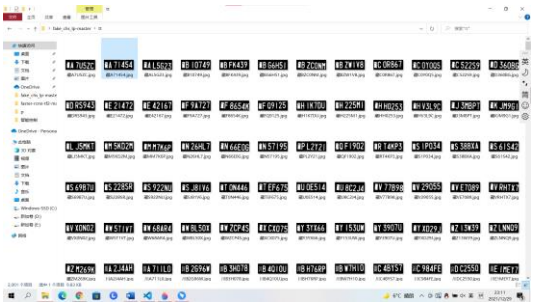
于是我生成了 2000 张随机的车牌信息用来训练，如下图，其中的车牌信息以文件名的形式保存。



我最初的设想是利用这 2000 张随机车牌训练车牌识别网络，但在迭代了 100 轮后很快发现了问题：训练出来的网络对于识别同样伪造生成的车牌有着极高的准确率，但是对于真实的车牌几乎完全无法识别。

我分析了原因后认为这个造成这个问题的原因是这 2000 张车牌太过于规则了：每一张车牌的信息都非常的清楚，而位置都非常地精准，造成了几乎无法识别的现象。

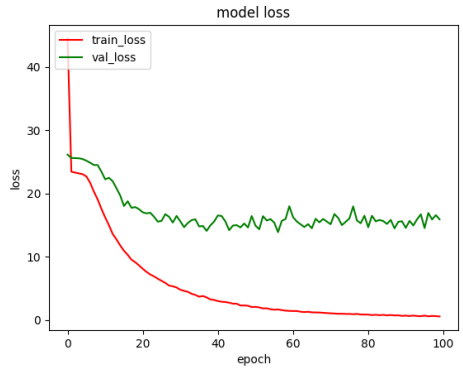
为了解决这个问题，我决定对这 2000 张照片做数据增强的处理。我对这 2000 张照片作了随机的旋转、平移操作，并往里面添加了一些噪声，并用随机的对其作模糊处理。最后为了消除色差对判断的影响，我还对这 2000 张图片做了二值化的处理，最后得到的效果图片如下：

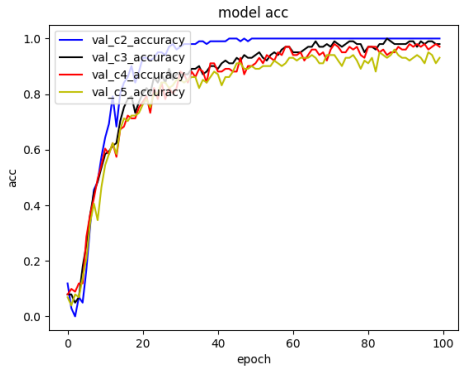


至此，数据处理全部完成。

6. 3 训练结果

最后在迭代了 100 轮后的结果如下：



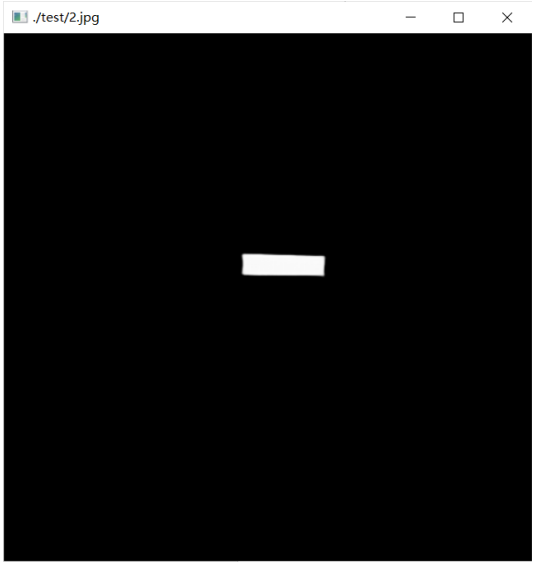


可以看到，由于这里损值取了七个字符之和，因此总的损失率比较高；但是结合准确率来看，验证集上每个字符的准确率都达到了 90%以上，取得了非常好的效果。

7　最终效果

我把所有三个步骤训练完成的代码拼装起来，并通过一张我在玉泉拍的照片为测试，得到的最终展示结果如下。

输入图片



Unet 输出结果

角点提取

提取到的车牌

识别结果（最终结果由虚假生成车牌软件提供）

8 个人心得

在真的完成这个项目之前其实我完全没有想过这个项目的识别效果会有这么好。在最开始决定这个项目的时候，想着是能完成 Fast-RCNN 的目标检测网络的搭建就算成功的。但通过在搭建网络过程中不断地查阅相关资料，更换神经网络，最终成功的实现了车牌网络的识别。

在搭建网络的过程中，遇到的最大的困难来源于数据的收集，因为网上的车牌数据非常稀少，特别是车牌数据，非常稀缺。最终这个问题通过我个人尝试的虚假车牌+数据增强的方式解决，并且效果很好，我也希望自己能够把这个方法分享给大家。

本次大作业中搭建的网络模型基本上参考了 B 站 up 主 Bubbliiiing 出的目标检测和语义分割的教程。在搭建的过程中，让我对各种复杂的神经网络结构有了更深层次的了解。

References

1. https://github.com/ufownl/fake_chs_lp
2. https://space.bilibili.com/472467171?from=search&seid=11485935467968362990&spm_id_from=333.337.0.0
3. https://blog.csdn.net/weixin_44791964/article/details/108513563