审定成绩：_____

重庆邮电大学
毕业设计（论文）

中文题目	基于卷积神经网络的车牌识别
英文题目	**License Plate Recognition based on Convolutional Neural Network**
学院名称	先进制造工程学院
学生姓名	黄旭峰
专　　业	机械电子工程
班　　级	**14021601**
学　　号	**2016214540**
指导教师	苏祖强 副教授
答辩组负责人	张毅 教授

2016 年 6 月

重庆邮电大学教务处制

先进制造工程学院本科毕业设计(论文)诚信承诺书

本人郑重承诺:

我向学院呈交的论文《基于卷积神经网络的车牌识别》，是本人在指导教师的指导下，独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本论文不含任何其他个人或集体已经发表或撰写过的作品成果。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明并致谢。本人完全意识到本声明的法律结果由本人承担。

年级

专业

班级

承诺人签名

年　　月　　日

学位论文版权使用授权书

本人完全了解重庆邮电大学有权保留、使用学位论文纸质版和电子版的规定，即学校有权向国家有关部门或机构送交论文，允许论文被查阅和借阅等。本人授权重庆邮电大学可以公布本学位论文的全部或部分内容，可编入有关数据库或信息系统进行检索、分析或评价，可以采用影印、缩印、扫描或拷贝等复制手段保存、汇编本学位论文。

（注：保密的学位论文在解密后适用本授权书。）

学生签名:　　指导老师签名:

日期:　　年　　月　　日　　日期:　　年　　月　　日

摘 要

随着我国交通网络的不断完善，人们开始寻求更加智能高效的交通管理模式，为此，车牌识别技术应运而生，本论文便是针对基于神经网络的车牌识别系统展开研究。

车牌识别技术主要分为车牌定位和字符识别两个主要部分。车牌定位主要是依靠 OPENCV 框架提供的 mser(Maximally Stable Extremal Regions)算法和颜色识别算法完成的，在进行车牌定位时，需要先将 RGB 色彩空间的图片转换至 HSV 色彩空间，此时便可以使用 mser 算法和颜色提取算法对图片提取车牌候选区域，最后，通过训练好的 SVM(Support Vector Machine)分类器模型对候选区域进行筛选并最终得到车牌区域。

字符分割是字符识别的前置工作，由于字符识别模型的输入是单字符图像，所以在得到车牌区域后需要利用边缘检测算法和滑动窗口算法将车牌分割成单字符图片以供模型识别。

字符识别是车牌识别的最后一步，本论文采用的是基于卷积神经网络的车牌识别方法，在经过车牌分割得到单字符图片后，单字符图片会被调整大小后传入神经网络模型进行识别，本文采用的卷积神经网络模型均为分类模型，模型在识别完成后，会返回该图片属于每一种类的概率作为识别结果，程序将概率最大的那一类作为该图片的识别结果，最后将所有图片的识别结果组合后输出，至此，车牌识别过程全部完成。

关键词：车牌识别，车牌定位，字符分割，字符识别，神经网络模型

Abstract

With the continuous improvement of China's transportation network, people began to seek more intelligent and efficient traffic management mode, to this end, License plate recognition technology came into being, this thesis is a research on License plate recognition based on neural network.

License plate recognition technology is mainly divided into two main parts, plate detection and character recognition. The plate detection is mainly rely on the mser algorithm provided by the OPENCV framework and color recognition algorithm. In the license plate detection, first need to convert the image of the RGB color space to the HSV color space. And then can use the mser algorithm and color extraction algorithm to extract the license plate candidate area of the image. finally, through the trained SVM classifier model to filter the candidate area and finally get the truth license plate area.

Character segmentation is a precursor to character recognition, and the input to the character recognition model is a single-character image, the license plate area needs to be segmented into single-character images for model recognition using edge detection algorithm and sliding window algorithm after getting the license plate area.

Character recognition is the last step of the license plate recognition. This paper uses a license plate recognition method that base on convolutional neural network. After the license plate split into single-character picture, single-character picture will be resized and input neural network model for recognition. The convolutional neural network models used in this paper are all classification models. Model after the recognition is completed, will return all probabilities that the picture belong to each kind as the recognition result. The program will select a class that with the largest probability. Finally the program will combine all the picture recognition result and output the result. To this point, the license plate recognition process is complete.

Keywords: license plate recognition, plate detection, character segmentation, character recognition, neural network

目录

第一章 绪论

1.1 课题的研究背景

近年来，随着我国经济的不断发展，基础建设不断完善，大量资金被投入公路建设当中，以求建设更加完善的交通网络体系，进一步促进我国的经济发展。同时我国的汽车保有量也在逐年增加，截止至2019年底，我国小型载客汽车保有量超过2.2亿台。人们发现越来越多的车流量和越来越长的里程数已经慢慢超过了人工处理的上限，落后的公路管理技术成为了限制现代化公路的主要因素之一。在此背景下，智能的车牌识别系统应运而生，车牌照识别系统是诸多现代化公路管理技术的基础，最常见的高速公路过站管理系统就是在此基础上发展而来。同时，自动交通执法系统也需要车牌识别技术的支持，快速而准确的识别出违章车连的车牌是自动交通执法系统的基础之一。此外，车牌识别系统也与人们的生活息息相关，如智能车库的入库管理，自动停车收费等，在这些需求的催化下，车牌识别系统从上个世纪九十年代开始飞速发展。

车辆牌照识别(License Plate Recognition，LPR) 技术是计算机视觉与模式识别技术在交通领域应用的重要研究课题之一，该技术主要依靠于从图片中自动定位出车牌图像，再从分割出的车牌图像中提取字符，进而对车牌进行识别，目前国内外在车牌识别方面已经有了十足进步，但在不同光线，天气下的识别精度优化一直是各国研究的主要目标。

1.2 国内外研究现状

各国对于车辆牌照识别技术的研究开始于上个世纪九十年代，至今已取得了十足的进步。车辆识别系统在识别车牌时，需要先对图片进行预处理，字符分割，字符分类和后处理等步骤，目前世界各国的研究重点主要集中在图像预处理，车牌定位，字符分割和字符识别这四点上。

图像预处理是车牌识别的第一步，车牌识别系统在实际使用时，其输入图像均是真实环境下的车辆图像，这些图像不可避免的存在过曝，模糊，光线不足等问题，图像的质量直接影响了车牌识别的准确率，将质量较差的图片经过算法处

理到符合系统的输入要求，这个过程称之为图像的预处理。在图像修复方面，二值化是一种常见的处理方式，二值化算法利用阈值将图片分为前景和背景，将识别目标从背景中分离出来，极大的提高了识别精度。在图像二值化处理中，阈值的选取是核心问题，Otsu算法是Nobuyuki Otsu[1]在1979年提出的一种阈值选取算法，至今仍然是使用最多的阈值选取算法之一，otsu算法会将图像中的像素以阈值为界按照灰度值分为两类，然后用最大类间方差解出阈值，这种阈值选取方法对于灰度区分明显的图像有非常好的效果。此外，争对特定场景的图像预处理方案也在被不断起初，张涛等[2]提出了一种在雾霾天气下去除图片雾霾的方案，该方案首先使用主成分分析和Fisher线性判别分析检测图片中的雾霾，再用暗原色先验的方式去除图片中的雾霾。在设计车牌识别系统时，将特定场景的预处理算法与通用与处理算法结合，通常可以取得良好的效果。

车牌定位是车牌照识别的第二步，车牌的检出率和定位精度直接影响到了最终识别结果的准确性，如何提高在复杂环境下的车牌定位精度一直是车牌识别的重点之一。根据车牌颜色固定的特性，沈勇武等[3]提出了一种基于颜色的车牌定位方法，通过分析局部区域内指定颜色的分布特征来定位车牌，缩短了处理时间，同时提高了定位精度。但由于环境复杂，环境中其他物品的颜色可能会干扰车牌的定位，不同的天气情况也会使颜色失真，所以更多优化的方案也在不断被人提出。基于纹理的车牌定位技术是另一种被广泛应用的车牌定位方式，Bai Hongliang等[4]将边缘检测技术和统计学分析结合，在多阈值下对车牌进行定位，达到了99.6%的准确率。Yuan Yule等[5]也提出了一种将线密度滤波器与sobel算子结合的车牌定位算法，在结合SVM分类器筛选候选区域后准确率同样达到了96.2%。除了颜色与纹理学特征，最大极值稳定区域也是车牌定位中常用方法，Chao Gou等[6]采用了先利用纹理特征对车牌进行粗定位，然后利用最大极值稳定区域进行精定位的方案，取得了98.9%的定位准确率。近年来，由于目标检测网络的发展，卷积神经网络也被引入了车牌定位程序，朱倩倩[7]等提出了将Fast R-CNN用于车牌定位的解决方案，经过测试，其车牌定位准确率达到了99.6%，可见卷积神经网络在车牌定位方面比起传统定位算法存在一定优势。

字符分割是要处理的第三个重要课题，字符分割是指将完整的车牌图像分割为数个单字符图片，以供字符识别程序识别，字符分割中得到的字符图片质量很

大程度上决定了字符识别的准确性。贺智龙等[8]采用了一种基于垂直像素统计的字符分割方案，该方案首先将车牌转换为二值图像，然后统计图像中每列的像素种类，在两个字符的间隔出白色的像素点必然是最少的，便可以以此为依据分割字符。邓运生等[9]采用了另一种基于二值车牌的分割方法，首先统计车牌中前景区域的连通性，然后利用连通区域的外接矩形分割字符，同样取得了良好的效果。除了传统基于像素的车牌字符分割方式，全新的也在不断被提出，焦娜[10]提出了一种基于主曲线和软K段主曲线算法的字符分割算法而郑泽鸿[11]则将AP聚类算法引入了车牌字符分割，这两种算法都拥取得了很高的准确率。

字符识别是要处理的最后一个课题，也是车牌识别中的核心内容，在过去人工神经网络是实现车牌字符识别最常用的方法，如李飞[12]采用的先对图片进行多点特征提取，再以特征点作为节点，搭建BP神经网络用以识别字符的处理方式，获得了不错的效果，但BP神经网络面临的问题是，在需要更高的识别精度时需要提取更多的特征点，但过多的特征点会使BP神经网络的训练时间大幅增加，对此，Yujie Liu[13]提出了将卷积神经网络用于改善中文车牌识别效果的方案，与传统的神经网络相对比，对比传统的神经网络算法，卷积神经网络拥有权值数量低的优点，降低了网络模型的复杂度，提高模型的运算速度的同时提高了模型的预测精度，因此，该方案在使用多种复杂环境下的车辆图片测试后，识别准确率达到了93%。此外，许多基于卷积神经网络的车牌识别衍生方案也在不断被提出，訾晶等[14]将CNN网络与现场可编程门列阵（Field Programmable Gate Array）相结合，利用FPGA的动态部分重构技术，将神经网络的训练速度提高了3倍以上，同时得到了99.45%的识别准确率，证明了FPGA与CNN技术结合的可能性，为技术的发展提供了新的方向。

1.3 论文的主要内容

本论文主要提出了一种专门针对中文车牌的识别方式和其具体的程序实现。该方法通过多种方法的协同工作实现了车牌的定位，分割和校正工作，通过车牌的形态学特征实现了车牌的字符分割，然后将分割好的车牌传入训练好的神经网络模型实现了最终的车牌字符识别。

本论文的主要部分分为四章，分别是图像预处理，车牌定位，字符分割和字符识别。

本论文的第二章介绍了图片预处理的全部过程和所用算法的理论解释。图像预处理是本论文的基础工作之一，本论文提出的程序不仅需要对过曝、噪点过多的图片进行校正，同时还需要将原本在 RGB 色彩空间的图片转换到 HSV 色彩空间，并且对图片进行二值化，滤波，形态学处理等操作后才能被车牌定位程序所使用。包括之后用于神经网络使用的归一化操作，也是图像预处理的一部分。本章节主要阐述了不同图像预处理函数的原理，并且展示了每一次处理后图像的预览图。

本论文的第三章主要讲解了车牌定位和车牌字符分割的具体内容。车牌定位是本论文的核心工作之一，包括车牌定位，车牌区域分割和车牌处理三个部分。考虑到真实环境复杂，本论文采用了颜色定位和 mers 定位两种方式来对车牌区域进行定位，总体准确率在 96%以上。车牌区域分割则是将定位到的车牌从原图像中单独分割出来以供使用。车牌处理是考虑到真实情况，需要对得到的车牌区域进行校正，去铆钉等操作，以供后续的字符分割程序使用。字符分割指将车牌定位中得到的车牌区域，根据车牌的形态特征，将其分割成七个单字符的图片，这些单字符图片将被用于传输到后续生成的神经网络模型中进行具体的字符识别。

本论文的第四章讲解了车牌字符识别的主要内容，在第四章中本论文先是对卷积神经网络的基础理论进行了详细解释，包括基础网络模型，损失函数，激活函数，卷积层，池化层，全连接层等卷积神经网络基本组件，以及回归分析，梯度下降，反向传播等卷积神经网络所用到的基本算法。在本章接下来的内容中对本论文提出的程序所用的卷积神经网络模型进行了阐述，包括数据集划分，网络参数等内容，并且在本章最后对模型测试结果进行了展示。

本论文的第五章对本论文提出的程序做出了详细的解释，讲解了程序所需要的编译器，依赖库，神经网络训练框架等内容，同时阐述了程序的部分核心实现代码，在本章的最后对程序的实际运行结果进行了展示。

图 1.1 展示了本论文提出的车牌识别方案的全过程。

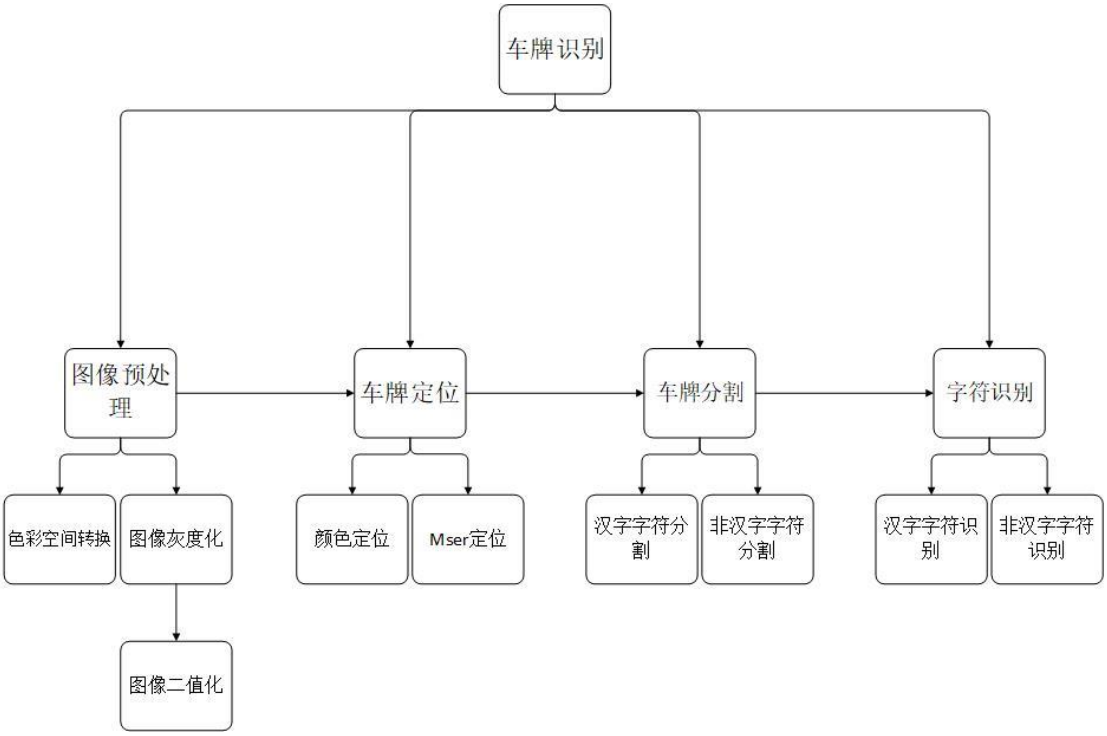
车牌识别
图像预处理
车牌定位
车牌分割
字符识别
色彩空间转换
图像灰度化
图像二值化
颜色定位
Mser定位
汉字字符分割
非汉字字符分割
汉字字符识别
非汉字字符识别

图 1.1 字符识别流程图

第二章 车辆图像预处理

图像预处理在本论文提出的程序中主要负责两个方面，第一个方面是对原始输入图片进行修正，在本论文提出的程序中，程序的直接输入是真实环境下的车辆图片，这些照片往往伴随过曝，光线不足，背光等不良因素，这些图片数据是无法直接被使用的，所以必须要经过二值化等操作，使其符合车牌定位，字符识别等后续程序的输入要求。第二个方面就是对已经经过调整后的图片进行更复杂的图像操作，如色彩空间转换，形态学操作，使其满足车牌定位或字符识别函数的特殊需求。在车牌识别中，图像预处理是一切工作的前置工作。

2.1 RGB 色彩空间与 HSV 色彩空间的转换

在本论文提出的程序中，颜色定位是车牌定位主要依靠的定位方法之一，由于我国车牌颜色组合非常特殊，通常只要在原始图片中提取蓝色区域后再将大小明显不符合的区域去除掉便可排除绝大部分的非车牌区域。

在本论文收集的车辆原始图片中，绝大部分采用的都是 RGB 色彩标准，即图像一共分为红（R），绿（G），蓝（B）三个颜色通道，每种颜色在各自的颜色通道内被分为 256 个亮度阶级（0 为最弱，255 为最强），原图中每一个像素点的颜色都是由不同亮度阶的三种颜色叠加而成，例如当三个通道亮度阶级都为 0 时，该点的颜色就是黑色，当三个通道亮度阶级都为 255 是，该点的颜色就为白色。利用 RGB 色彩空间描述色彩非常直观，但是在做数字图像处理时 RGB 色彩空间有着致命的缺点。

RGB 色彩空间模型如图 2.1 所示：

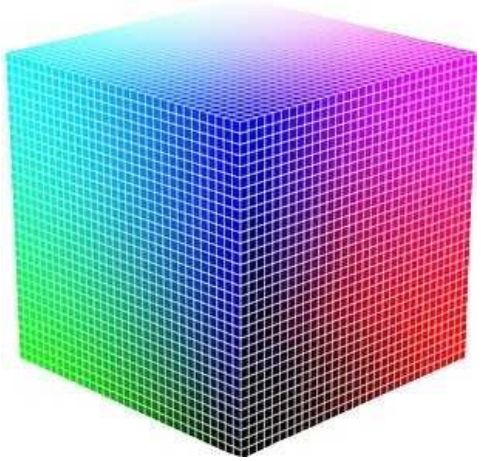

图 2.1 RGB 色彩空间

上图中 X,Y,Z 轴分别表示 R,G,B 三个通道的亮度阶级。可以看到蓝色的分布大致为一个三角锥形，这意味着很难在单个轴上确定一种颜色的范围。例如在 Z 轴上，蓝色的范围覆盖了 0 到 255 的所有区域，必须要借助 X 和 Y 轴的数据才能确定颜色。这对于程序的实现是十分不利的。对此，我们需要引入一种新的颜色空间，即 HSV 颜色空间[15][16]。

HSV 颜色空间中颜色也是靠三个变量确定的，即色相（H），饱和度（S），明度（V），其颜色空间模型如图 2.2 所示。

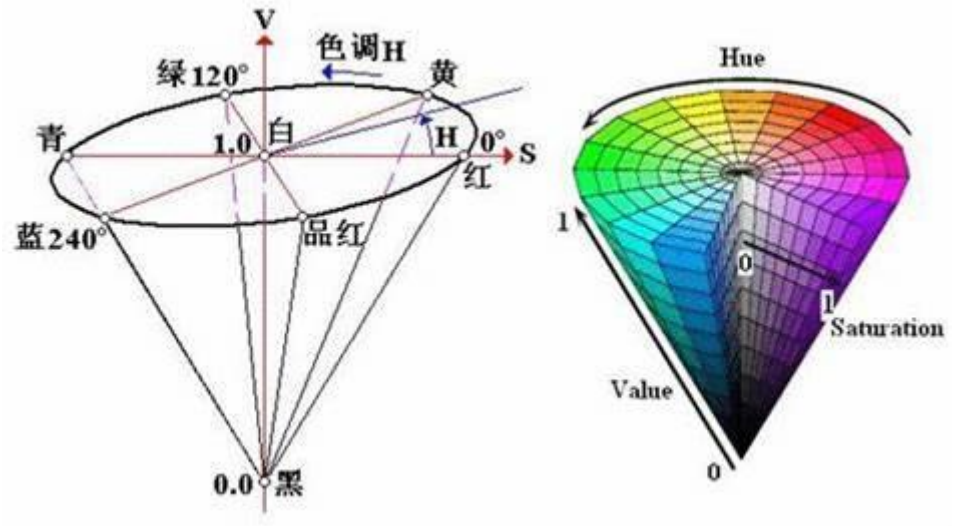

图 2.2 HSV 色彩空间

可以在模型中看到，色相（H）表示颜色的信息，该参数是一个角度值，红绿蓝三色分别间隔 120 度。饱和度（S）是一个比例值，范围是 0 到 1。代表所选颜色的纯度和该颜色最大纯度之间的比率，当 S=0 时，只有灰度。明度（V）表示明亮的程度，范围是 0 到 1。

从图上可以看到，对于参数 H 来说，蓝色全部分布在 180 度到 300 度之间。考虑车牌的实际颜色和光照对颜色的影响，可以得出蓝色车牌的取值为 200<H<280, 0.4<S<1, 0.4<V<1。

2.2 图像二值化

2.3.1 生成灰度图

灰度图是一种只用不同强度的黑色与白色作为参数的图像表示方式，与 RGB 图像不同，灰度图只有一个通道，这意味着在作模型训练和预测时，只需要传入一组数据即可，这对于模型训练是非常有利的。

RGB 图像与灰度图的转化方法一般基于一个著名的心理学公式：

$$Gray = R \times 0.299 + G \times 0.587 + B \times 0.114 \tag{2.1}$$

但是由于计算机处理浮点数运算效率较低，在图片像素过多时可能会导致转换时间过长，所以在程序中一般会将其参数转换成整数进行运算，即：

$$Gray = \left(R \times 299 + G \times 587 + B \times 114\right) / 1000 \tag{2.2}$$

图 2.3 是灰度图和 RGB 图像的对比图。

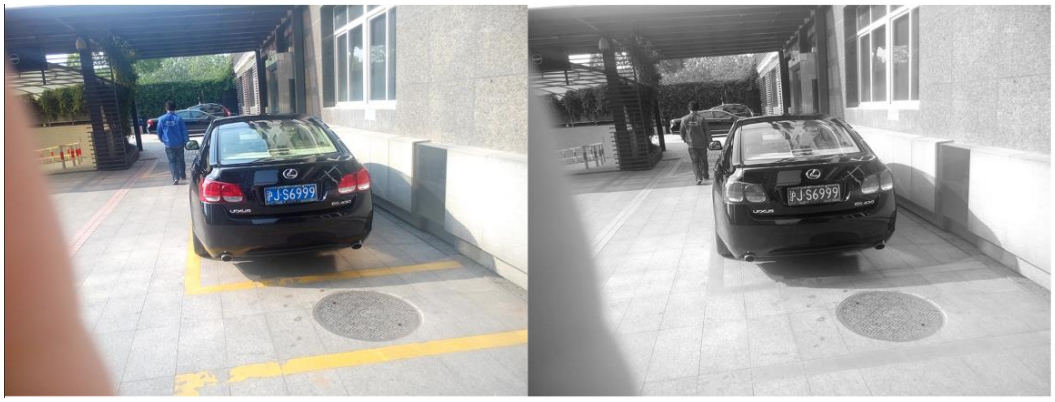

图 2.3 RGB 图片与 Gray 图片对比

2.3.2 灰度图的二值化

图像二值化是指将灰度图片转换成黑白图片的过程，程序会根据算法生成一个阈值，在图片中灰度大于阈值的点灰度会被设置为 255，灰度小于阈值的点会被设置为 0。这个产生阈值的过程被称为阈值操作，它是图像预处理中最为核心的部分。

阈值操作一般分为全局阈值和自适应阈值两种，全局阈值是指算法会根据图片全部的像素灰度生成一个阈值，该阈值会被用于处理图中的每一个像素。自适应阈值则是指算法会将图片分成多个区域，然后对这些区域分别计算阈值，生成的阈值也只会用于处理对应区域的像素。自适应阈值一般用于原图片区分度过低的时候。

本论文提出的算法基本采用基于 OTSU 算法的阈值操作，OTSU 算法会根据直方图将灰度图中的像素分为C_1和C_2两类，此时我们引入类间方差的概念，类间方差是指用C_1，C_2两类的像素灰度平均值去与整张图的像素灰度平均值求方差后再相减。即：

$$n^2 = P_1\left(m_{g1} - m_g\right)^2 + P_2\left(m_{g2} - m_g\right)^2 \tag{2.3}$$

式中 P_1 ——C_1类像素占总像素的比例；

P_2 ——C_2类像素占总像素的比例；

m_{g1} ——C_1类中像素的平均灰度值；

m_{g2} ——C_2类中像素的平均灰度值；

m_g ——整张图片的平均灰度值；

n^2 ——类间方差；

OTSU 算法便是期望找到一个阈值，以该阈值为界将图片的像素划为两类后，使得两类的类间方差最大。由算法可知，如果将灰度图中像素的灰度图统计并生成直方图，当直方图呈现如图 2.4 这种典型的双峰图的话，OTSU 算法都能取得很好的效果。

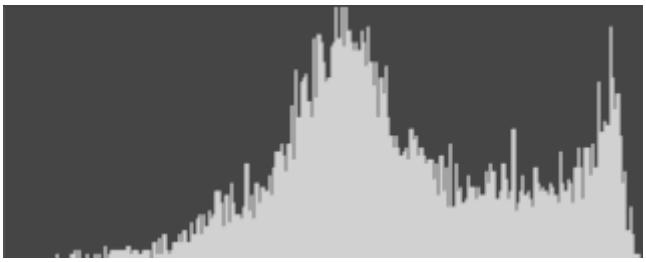

图 2.4 双峰图

由于我国绝大部分车牌都采用双色表示，在车牌灰度化后都会呈现典型的双峰图，所以 OTSU 算法在本程序中都能起到良好的效果。

图 2.5 展示了车牌灰度图和经过二值化后的图像。

图 2.5 车牌二值化效果展示

2.3 形态学处理

形态学处理是指一系列处理图像形态特征的记数，原理是利用一种特殊形状的结构元测量或提取图片中特定的形状或特征。

形态学处理所需要的输入图像要求是经过二值化的图像，输入图像中灰度为255的点称之为前景，灰度为0的点称之为背景。

用于操作图像的结构元实际上是一个特定大小的矩阵，矩阵中特定位置的点为1，其余位置的点为0。结构元矩阵中还会设定一个特殊的点，称之为锚点，锚点一般设在结构元矩阵的中心。在进行形态学处理时，需要将结构元矩阵扫过输入图像，根据不同的操作将结构元矩阵与输入图像进行不同的矩阵操作。

1）膨胀

膨胀操作是指将结构元矩阵扫过输入图像，将结构元矩阵锚点所对应的输入图像中的灰度值设置为结构元矩阵中所有值为 1 的点所对应的输入图像中的点的最大灰度值。图2.6就是膨胀操作的示意图。

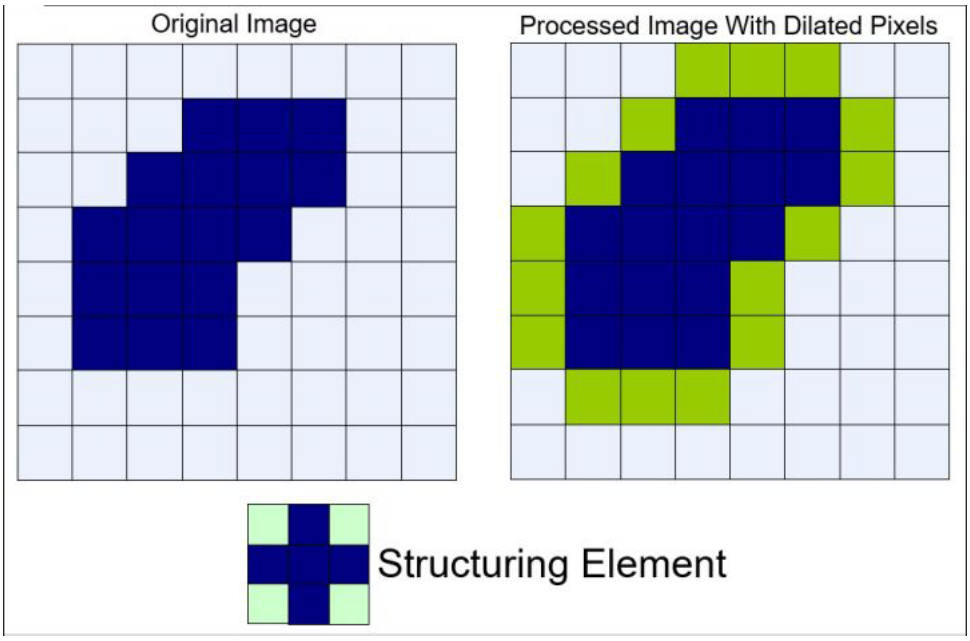

图2.6 形态学操作（膨胀）

上图中左边为输入图像，蓝色的格子为前景，灰度值为255，白色格子为背景，灰度值为0。现在将一个十字形的结构元矩阵划过输入图像，锚点在每一个格子上进行一次矩阵运算。右图为进行膨胀后的图像，绿色格子是在膨胀操作中被设为255的点。可以看到前景区域变大了，这就是所谓的“膨胀”。

2）腐蚀

腐蚀操作是指将结构元矩阵扫过输入图像，将结构元矩阵锚点所对应的输入图像中的灰度值设置为结构元矩阵中所有值为 1 的点所对应的输入图像中的点的最小灰度值。图 2.7 就是腐蚀操作的示意图。

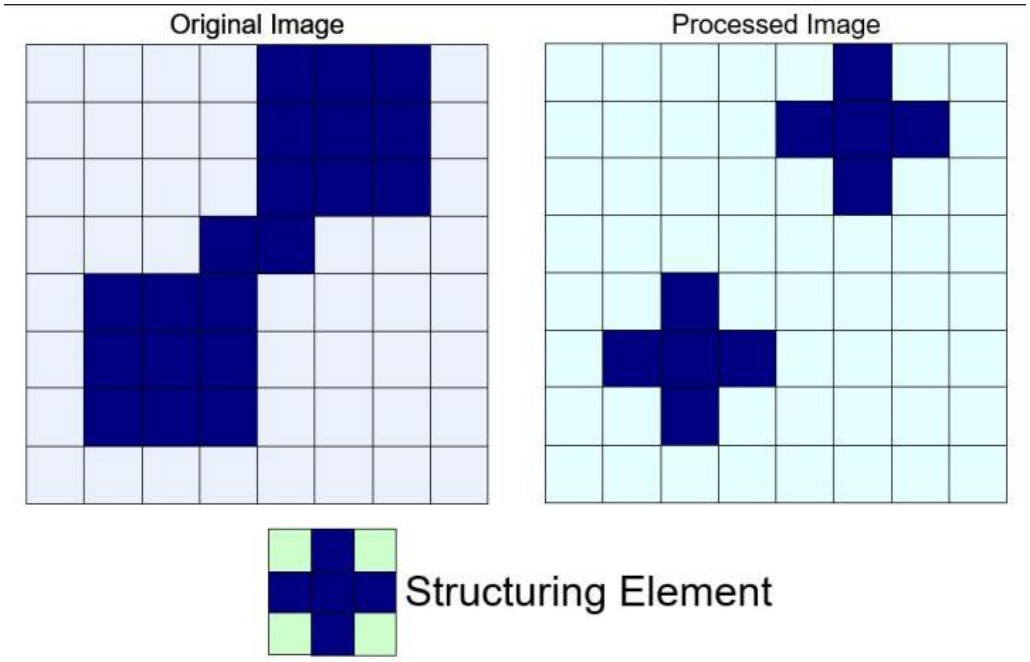

图 2.7 形态学操作（腐蚀）

右图为进行腐蚀后的图像，绿色格子是在膨胀操作中被设为 0 的点。可以看到前景区域变小了，这就是所谓的“腐蚀”。

3）开运算

开运算是膨胀和腐蚀的复合操作，指先对输入图像进行腐蚀，再对腐蚀后的图像进行膨胀，图 2.8 是开运算的示意图。

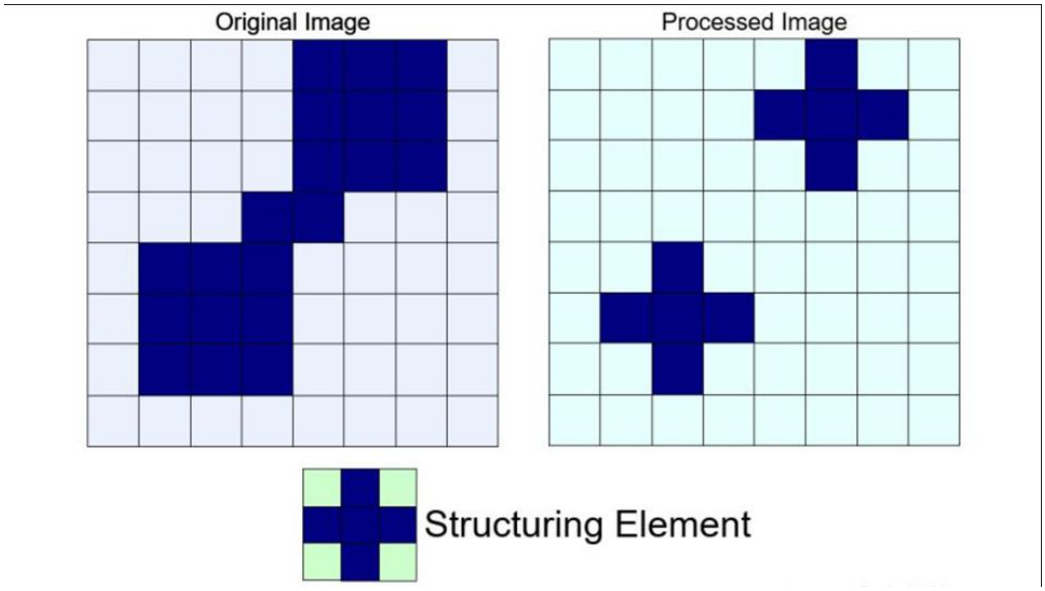

图 2.8 形态学操作（开运算）

可以看到经过开运算处理的图像中，原本连接的前景区域断开了，这便是开运算的作用：使图像中细小的连接部分断开。

4）闭运算

闭运算是指指先对输入图像进行膨胀，再对膨胀后的图像进行腐蚀，图 2.9 是闭运算的示意图。

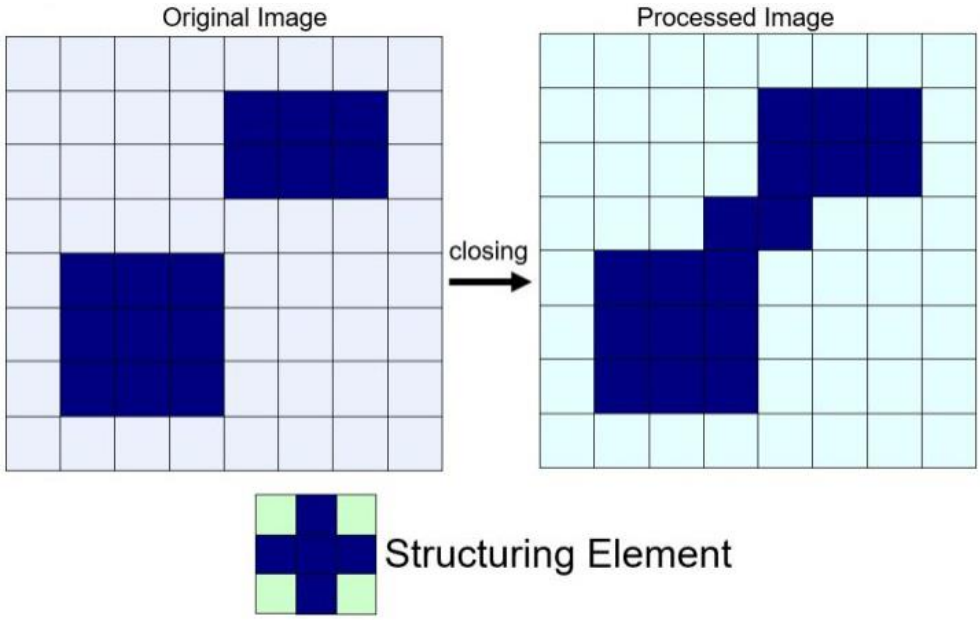

图 2.9 形态学操作（闭运算）

可以看到经过闭运算处理的图像中，原本断开的前景区域链接在了一起，这便是闭运算的作用：使图像中相近的区域连接在一起。

2.4 本章总结

本章介绍了本课题所要用到的图片处理技术和他们的理论依据，介绍了RGB 色彩空间和 HSV 色彩空间的基础理论，分析了图像二值化的基础流程和OTSU 算法的原理，同时解释了形态学操作中膨胀和腐蚀的工作原理，并且分析了复合操作开运算和闭运算的操作原理和具体作用。

本章不仅介绍了本课题所用图片处理技术的原理，还对它们在本课题提出的程序中如何工作进行了讲解，本章详细分析了两种颜色空间的优缺点并讲解了它们在程序中使用的情况，解释图像二值化原理的同时分析了为什么要选用 OTSU 算法作为二值化阈值选取的依据。在本章的最后介绍了形态学操作的具体原理，并且对形态学操作在本程序中具体工作效果进行了展示。

第三章 车牌定位及车牌字符分割

车牌定位是字符识别的前置工作之一，在车牌识别的实际应用中，程序的输入图片是真实的车辆图像，此时需要先在原图中找到车牌的位置，将其分割出来形成单独的车牌图像后，才可以被后续程序使用。本文提出的程序采用两种车牌定位算法，即基于颜色的车牌定位算法和基于 mser 的目标识别算法。本论文详细分析了两种算法的理论、优缺点和它们在实际应用中的表现，最终选择了两种方法协同工作的方式作为解决方案。

字符分割是字符识别和车牌定位的中间步骤，在本论文中，字符识别阶段要求的输入图片为单字符图片，而车牌定位后得到的输出图片为单个的车牌图片，此时便需要字符分割程序来将定位后的车牌图片分割成七个单字符图片以供程序作字符识别。

不仅如此，在本论文提供的程序中，汉字识别程序和英文及数字字符识别程序使用的是不同的卷积神经网络模型，所以，此时字符分割程序还需要负责将汉字从分割好的字符图片中分类出来，可以说，字符分割程序的好坏直接影响了字符识别程序的效率和准确率。

3.1 定位方式的分析与选择

3.1.1 基于颜色的边缘提取方式

基于颜色的边缘提取是本论文主要采用的车牌定位方式之一，它是一种简单高效的车牌定位方式，其理论依据是利用车牌独特的颜色组合来确定车牌的位置。本论文识别的是我国小型汽车的车牌，也就是蓝底白字白线框的车牌。在进行车牌定位时，我们只需要提取图中蓝色的区域，再去掉其中大小明显不合适的预选区域，就可以去除绝大部分非车牌区域。

基于颜色的边缘提取方式的输入是 RGB 彩色图像，算法会首先将 RGB 色彩空间的图像转换到 HSV 色彩空间，进过实际调研，可以发现在正常光照情况下，车牌所用蓝色在 HSV 色彩空间中分布在 $200<H<280,0.4<S<1,0.4<V<1$ 的范围内。所以程序会扫描整张图片的像素，将其中在蓝色范围内的像素设置为白色，

将非蓝色范围内的像素设置为黑色。此时相当于对车牌进行了一次特殊的二值化操作，图 3.1 为经过处理的图片。

图 3.1 经过颜色处理的车辆图片

在得到经过二值化的图像后，程序会对图片进行形态学操作中的开运算，以消除图片中的噪点，然后程序会对图像进行边缘提取，并且将提取区域的最小外接矩形作为候选区域。

由上述可知，基于颜色的车牌定位方式的理论简单，可靠性强，最终得到的候选区域一般在 0 到 3 个，数量合适，传输到 SVM 分类器中并不会造成过大的运算负担，总体来说是一种可靠的算法。然而该方法受光照条件限制过大，在极端光照条件下车牌检出率会受到较大的影响，

图 3.2 是原始车辆图片和经过颜色车牌定位算法处理后提取到的车牌图片，可以看到，算法误提取到的车牌区域很少，在图 3.5 这种情况下，不需要进行复杂的程序处理就可以筛选掉非车牌区域。

图 3.2 颜色定位算法效果展示

3.1.2 基于 mser 的边缘检测算法

Mser(Maximally Stable Extrernal Regions)算法又称最大稳定区域算法，是一种依靠图像灰度来定位区域的算法。Mser 算法的输入是灰度图，在做 Mser 算法定位时，程序会设定一个灰度阈值，该灰度阈值会依次取[0,255]中的所有值，每取一个值，就会以该阈值作为分界值对输入图像做一次二值化。二值化的图像中，黑色区域会形成数个连通区域，随着灰度阈值取值的不断变大，原图中灰度较小的像素点会被设置为白色，黑色连通区域的面积就会缩小。设Δ为微小的灰度阈值变化，则有公式：

$$v(i)=\frac{|Q_{i+\Delta}-Q_{i-\Delta}|}{|Q_i|} \tag{3.1}$$

式中： $v(i)$ —— 表示第 i 个区域面积的变化率。

Q_i —— 表示第 i 个连通区域的面积。

$Q_{i+\Delta}$ —— 表示梯度阈值增加Δ后第 i 个连通区域的面积。

$Q_{i-\Delta}$ —— 表示梯度阈值减少Δ后第 i 个连通区域的面积。

在程序处理 mser 算法时会设定一个变化率阈值v_0，当$v(i)<v_0$时，则将该连通区域记为最大稳定区域。换言之，mser 算法会在图像中提取出灰度接近的连通区域，由于车牌颜色一致，所以 mser 算法在车牌定位中有良好的表现，同时，

由于 mser 算法的处理对象是灰度图，所以 mser 算法受到光照带来的色差影响较小，弥补了颜色定位的不足。

本程序在处理 mser 算法时会设置一个面积阈值，算法只会提取在阈值范围内的最大稳定区域。图 3.3 展示了经过 mser 算法提取车牌后的图片效果，图中红线画出的区域就是车牌提取到的区域，黑线画出的区域为错误提取到的非车牌区域，可见 mser 算法不仅提取到了车牌区域，而且误提取到的非车牌区域较少，提取效果良好。

图 3.3 mser 车牌定位算法效果展示

3.2 SVM 分类器详解

SVM 分类器又称作支持向量机（support vector machine），它是一种典型的二分类算法，SVM 分类器期望在一个数据空间中找出一个超平面，该超平面能最正确的将两类数据划分开[17]。SVM 分类器与深度神经网络的不同在于 SVM 分类器拥有两个独特的结构，Hinge Loss 和 kernel method[17]。

3.2.1 支持向量与 Hinge Loss

SVM 分类器中，超平面的表达式是$w^T x + b = 0$,图 3.4 展示了一个在二维平面上的数据集，SVM 分类器便是要在该平面上找出一条直线，它可以将两类数据最优的区分开。根据点到直线的距离公式，数据点到超平面的距离可表示为$\frac{|w^T x+b|}{||w||}$,则最优的超平面就是当这个值最大时的超平面。

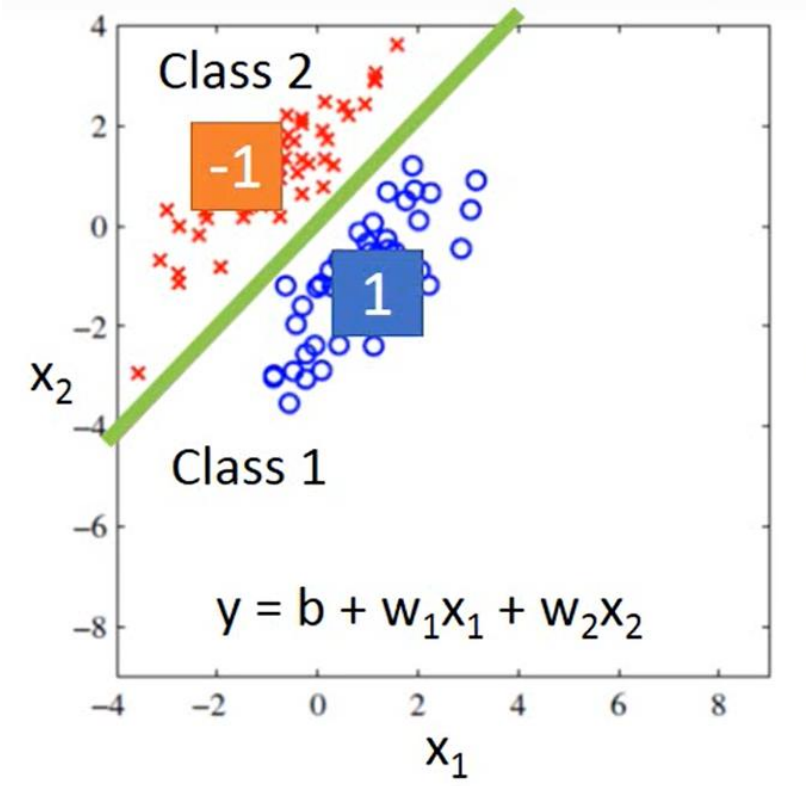

图 3.4 二位平面数据集

对于 SVM 分类器而言，模型并不是要找出所有数据点的平均距离，而是使离超平面最近的两个异类点（或多个共线数据点）点距超频面的距离之和最大，这些点就称之为支持向量（support vector），如图 3.5 所示，图中虚线上的三个点就是该 SVM 分类器的支持向量。

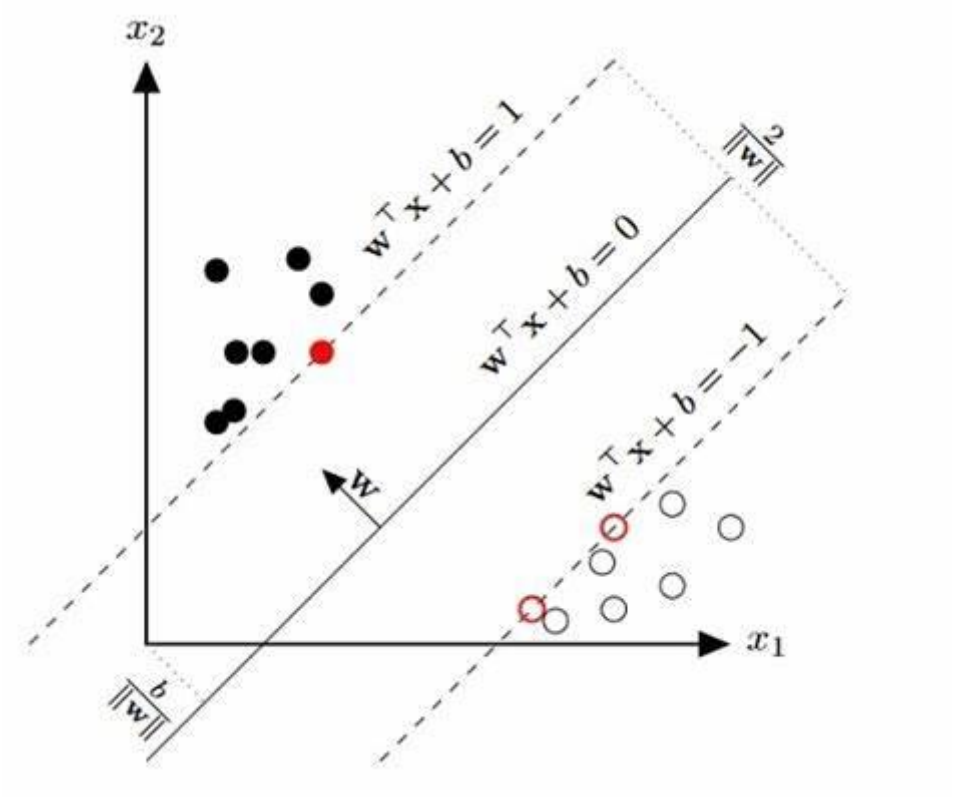

图 3.5 支持向量

现假设一个 SVM 分类器能将某平面上的数据点正确分类，设$g(x)=w^Tx+b$, $\hat{y}^n$表示第 n 个数据对应的真值标签（1 或-1）则有：

$$\begin{cases} g(x)\geq +1 & \hat{y}^n=+1 \\ g(x)\leq -1 & \hat{y}^n=-1 \end{cases} \tag{3.2}$$

根据公式（3.2）可得出:

$$y^n g(x)\geq 1 \tag{3.3}$$

式（3.3）便是 SVM 分类器的函数表达式。

Hinge Loss 是一个专用于二分类模型的损失函数， 在式（3.2）中$g(x)>+1$为一类，函数输出+1，$g(x)<-1$为另一类，函数输出-1。如果用 Hinge Loss 作为该模型的损失函数，设，则 Hinge Loss 为：

$$\max\left(0,1-y^n g\left(x\right)\right) \tag{3.4}$$

如果将$y^n g(x)$作为横轴，损失函数的值作为纵轴作图，Hinge Loss 的图像如图 3.6 所示。

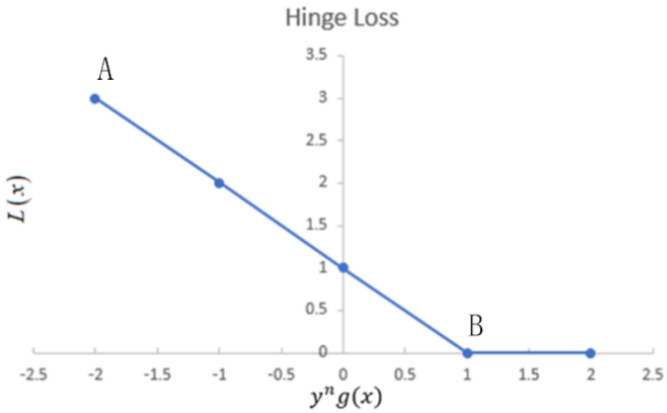

图 3.6 Hinge Loss

如图所示，Hinge Loss 对$g(x)$的导数为：

$$\frac{\partial\max\left(0,1-y^n g\left(x\right)\right)}{\partial g\left(x\right)}=\begin{cases}-\hat{y}^n & 1-y^n g\left(x\right)>0\\ 0 & otherwise\end{cases} \tag{3.5}$$

可见，只有在图 3.6 中$y^n g(x)$在 A 到 B 段之间的数据点才会更新 SVM 分类器的参数，这些可以更新模型参数的点，就是本节开头提到的“支持向量”。

对于 SVM 分类器而言如果要得到最优模型就要使式(3.4)的值最小，则$L\left(x\right)$等价于：

$$\begin{cases}L\left(x\right)\geq 0\\ L\left(x\right)\geq 1-y^n g\left(x\right)\end{cases}=>y^n g\left(x\right)\geq 1-L\left(x\right) \tag{3.6}$$

式（3.6）中的$L\left(x\right)$又叫做宽松变量（slack variable）,是用于在 SVM 模型无法满足$y^n g\left(x\right)\geq 1$时，用于稍微放宽临界条件的变量。从（3.6）中去除$L\left(x\right)$就从 Hinge Loss 推导出了 SVM 分类器的基本形式: $y^n g\left(x\right)\geq 1$。

3.3.2 核方法（kernel method）

在实际工程使用中，同一平面上的两类数据在它们所在的平面可能是线性不可分的，即数据在低维空间线性不可分，但如果将数据映射到高维平面，则有可能使其线性可分，核方法便是提供了在原始数据对应的高维空间使用线性方法来分析和解决问题的能力。核方法的核心式核函数（kernel function）,由 3.3.1 节可知，SVM 分类器中超平面的表达式为 $g(x)=w^Tx+b$,已知 $w^T=\sum_n a_n x^n$,则有 $g(x)=\sum_n a_n(x^n\cdot x)$,设：

$$K(x^n,x)=(x^n\cdot x) \tag{3.7}$$

式（3.7）就是核函数。

核函数不仅拥有在高维空间解决分类问题的能力，同时也能大幅简化高维空间向量的运算时间。假设有向量 x=$\begin{bmatrix}x_1\\x_2\end{bmatrix}$,z=$\begin{bmatrix}z_1\\z_2\end{bmatrix}$,假设这两个向量在二维平面上不可分但在其三维平面上的多项式映射可分，则有:

$$\varnothing(x)=\begin{bmatrix}x_1\\\sqrt{2}x_1x_2\\x_2\end{bmatrix}\varnothing(z)=\begin{bmatrix}z_1\\\sqrt{2}z_1z_2\\z_2\end{bmatrix} \tag{3.8}$$

将式（3.8）带入式（3.7）求内积有:

$$K(\varnothing(x),\varnothing(z))=(x_1z_1+x_2z_2)^2 \tag{3.9}$$

可以发现式（3.9）等于：

$$K(\varnothing(x),\varnothing(z))=(x\cdot z)^2=\left(\begin{bmatrix}x_1\\x_2\end{bmatrix}\cdot\begin{bmatrix}z_1\\z_2\end{bmatrix}\right)^2 \tag{3.10}$$

式（3.10）证明了特征向量在高维空间多项式映射的内积等于其在低维空间内积的平方。这个计算方式在特征向量原始维度高时可以显著降低计算时间，因为高维向量的更高维投影其参数会几何倍数增长，此时求高维映射向量的内积计算量极大，使用核函数可以显著降低计算时间。

值得一提的是，根据高维映射的方式不同，核函数也不同，表 3.1 列举了一些常见的核函数。

表 3.1 常见的核函数

名称	表达式
线性核	$k(x_i, x_j) = x_i^T x_j$
多项式核	$k(x_i, x_j) = (x_i^T x_j)^d$
高斯核	$k(x_i, x_j) = \exp\left(-\frac{\|\|x_i - x_j\|\|^2}{2\sigma^2}\right)$
拉普拉斯核	$k(x_i, x_j) = \exp\left(-\frac{\|\|x_i - x_j\|\|}{\sigma}\right)$

3.3 车牌分类模型

本论文提出的程序在由两种车牌定位算法提取车牌后，通常会得到 2-10 个非车牌区域，然后程序会将车牌候选区域输入训练好的 SVM 分类器判别该区域是否为车牌，然后将车牌区域传入字符识分割程序使用。

1）数据集划分

本论文使用的 SVM 分类器训练数据集由车牌和非车牌图像组成，在训练集中共有 1400 张车牌图片，2170 张非车牌图片。测试集中共有 514 张车牌图片，1815 张非车牌图片。图 3.10 和 3.11 展示了车牌数据集和非车牌数据集的切片。

图 3.10 车牌数据集切片

图 3.11 非车牌数据集切片

2）SVM 模型训练

本论文所用的 SVM 模型是 licsvm 库中的 C-SVC 模型，采用的核函数是径向基函数，即表 3.1 中的高斯核函数，函数中的参数σ设定为 0.1。

测试完成后，利用 64 张随机抽取的图片进行模型测试，其中正样本 36 个负样本 28 个，其测试结果如表 3.2 所示。

表 3.2 SVM 模型测试结果

名称	预测结果	数量
正样本	1	34
正样本	0	2
负样本	1	4
负样本	0	24

由表 3.2 可知，SVM 分类器在测试集上取得了 90.63%的准确率。

3.4 车牌分割

3.4.1 汉字提取和滑动窗口详解

我国小型汽车车牌是由七个汉字及英文和数字字符组成的，其构成特点之一就是第一个字符必然是我国省份的汉字简称且汉字不会出现在其他位置，在这个前提下，将汉字和其他字符放在一起识别显得并不合理。所以，将汉字的部分切

割出来单独训练一个识别模型用于识别，从准确率角度上来看是更加正确的处理方案。

在做字符分割时，字符间的间隔是算法使用的关键，程序会将分割好的车牌进行二值化处理和形态学处理后对其进行提取像素灰度跃变点，再对这些像素做最小外接矩形。换言之就是利用每一个字符间的间隔来分割字符。这个方法对非汉字字符有很好的表现，但是汉字字符与非汉字字符有着一个根本性的不同是汉字存在左右结构。实际上我国部分省份的简称也存在上下结构，但上下结构的汉字上下两部分分离程度并不是很大，在经过形态学处理后上下结构基本可以连成整体，但左右结构的结构体间分离过大，程序很难判断汉字是一个字符还是多个字符。例如，“川”这个汉字被传入程序后，常规方法很难分辨究竟是“川”还是三个“1”。此时便需要用其他算法来辅助分割汉字字符。本论文提出的方式是滑动窗口来处理这个问题。

1）滑动窗口算法

滑动窗口算法顾名思义就是建立一个蒙版，然后将这个蒙版滑过图像，并且最终在这些蒙版覆盖区域内选择最适合的区域当作目标区域。当然，在本论文提出的程序中，考虑到车牌的字符结构特征。并不需要让蒙版滑过整个车牌图片。在本论文提出的程序中，只需要让蒙版在车牌左边起始处至第二个字符也就是地级行政区代号的起始处之间滑动即可。在蒙版滑动后，会产生数个候选区域用于进行进一步筛选。

在本程序的滑动窗口实现代码中，考虑到车牌的大小并不固定，所以滑动窗口的大小也不是定值，滑动窗口的取值基本遵循以下代码：滑动窗口的宽度是第二个字符区域也就是地级行政区代号区域的宽度乘以 1.4.滑动窗口高度是二个字符区域高度乘以 1.05。

在确定滑动窗口大小后，窗口会以一个像素为步长在车牌区域滑动，在滑动完成后会产生数个候选区域，这些区域会被放进一个数组中，在字符识别时，这些区域会被依次传入汉字识别模型进行识别，之后程序会选择出识别结果中最多的那一项作为最终的汉字识别结果。

3.4.2 非汉字字符分割及车牌字符分割结果展示

在上节中我们提到，非汉字字符分割主要依靠的是字符间的间隔，程序会将分割好的车牌进行二值化处理和形态学处理后对其进行提取像素灰度跃变点，再对这些像素做最小外接矩形，每一个矩形就是一个单字符区域。在程序实际实现时，程序会得到 7 到 12 个矩形区域，除了 7 个字符区域外，还会有因为噪声产生的数个非字符区域，算法需要利用矩形的长宽比来筛掉这些明显不合理的矩形区域。然后，程序会依照矩形区域左上角的坐标来定位出第二个字符区域也就是地级行政区代号区域所对应的矩形框并记录该矩形框为“特殊字符”，“特殊字符”所对应矩形框的数据会影响到汉字的提取。同时，程序还会舍去所有“特殊字符”左边的矩形框，因为那里是汉字区域，需要交给滑动窗口算法来处理。

图 4.1 展示了原始车牌图片，分割后的车牌区域与分割后的车牌字符三者的对比。

图 4.1 车牌分割效果展示

3.5 章节总结

本章的主要内容一共分为四部分，详细介绍了三种车牌定位算法，SVM 分类器的理论基础，车牌分类模型，及两种车牌字符的分割算法。

1）本论文在本章第一节中详细介绍了两种车牌定位方法的理论基础，分析了两种车牌定位方式的优势和不足。最后，对两种车牌定位方法在程序中实际表现进行了展示。

2）本章详细介绍了 SVM 分类器的理论基础，分析了 SVM 分类器独特的损失函数 Hinge Loss,的特性和优点，介绍了支持向量（support vector）和核函数（kernel Function）的理论基础，列举了核函数的优点和常见形式。并且详细介绍了本论文用到的 SVM 分类器的基本参数，以及 SVM 分类器所用数据库的组成和划分。最后展示了 SVM 分类器的训练效果。

3）本章解释了车牌字符分割时汉字字符与非汉字字符的不同，介绍了滑动窗口算法和利用 CNN 分类模型筛选最佳汉字区域的算法实现。然后提出了基于像素跃变点的非汉字字符分割算法。

第四章 车牌字符识别

字符识别是本论文最重要的部分，其主要负责将字符分割中得到的图片传入神经网络模型识别，然后将识别结果添加到输出数组。在字符分割中，我们已经得到了一个大小为 7 的图像数组，分别代表着车牌的 7 个字符，在字符识别的程序实现中，字符图片先会被重新调整成32×32像素大小的二值图像后再传入卷积神经网络模型中识别，最终程序会将识别后的结果存入输出数组传给输出程序处理。在本章中会设计诸多机器学习和神经网络的理论，但本论文只会对这些内容作概述性的介绍。

4.1 神经网络详解

4.1.1 神经网络简介

神经网络是机器学习的一种方法，本论文所希望通过机器学习达到的目的是给程序一张图片，程序就能自己分析并识别出图片中的内容。这是机器学习中重要的一个领域，其称为影像辨识。机器学习在处理影像辨识问题是，会期望于得到一个函数，当程序从输入图像中提取特征并传入函数后，函数的输出就是识别的结果。得到函数的过程称之为训练，本章节将会介绍训练所用方法的理论基础，以及这些方法在本论文提出的程序中是如何运作的。

4.1.2 回归分析

机器学习的理论基础便是依据已有数据拟合出一组可以根据输入特征来预测结果的函数，这个过程称之为回归分析。举个简单的例子，假设在二位平面上有 10 个数据，现在我们希望通过这十个数据拟合出一条曲线$f(x)$，这条曲线在遇到一个初次输入的 X 时，可以得出一个尽可能准确地预测 Y，这时的 X 称为输入特征而 Y 称为预测值。

1）损失函数

假设$f(x)=wx+b$，在这个函数中，函数预测的好坏由参数 w 和 b 决定，为了衡量一个$f(x)$的好坏，引入损失函数（LOSS FUNCTION），损失函数是一个关于$f(x)$的函数，损失函数可以是任何合理的函数，在这里将损失函数设为:

$$L(f)=\sum_{n=1}^{10}\left(\hat{y}-f\left(x_{cp}^{n}\right)\right)^{2} \tag{4.1}$$

式中： x_{cp}^{n} ——十个数据中第 n 个对应的 x 的值。

$.\hat{y}.$ ——是个数据中第 n 个对应的 y 值

这个函数的意义是，用每笔数据对应的 Y（真实值）减去$f(x)$根据每笔数据对应的 X 预测出的值（预测值），取平方再相加，就可以得到损失函数的值。那么由该损失函数的意义可知，如果要使 f(x)预测的尽量准确，就要找出一组w和b,使损失函数尽量的小。下一节将介绍缩小损失函数的具体方法。

2）梯度下降

梯度下降是回归模型拟合的具体方法之一，在此进一步简化 4.1 节中提到的$f(x)$为$f(x)=wx$。此时损失函数变为：

$$L(wx)=\sum_{n=1}^{10}\left(\hat{y}-wx_{cp}^{n}\right)^{2} \tag{4.2}$$

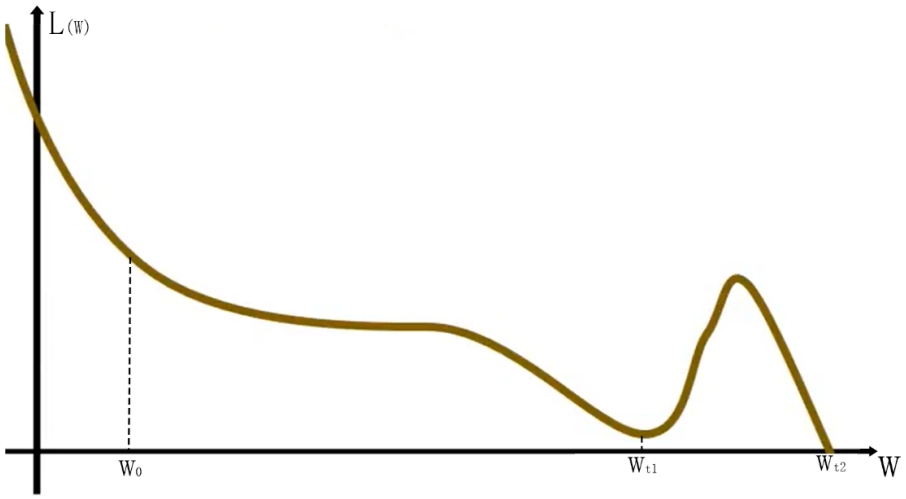

图 4.1 损失函数

在式(4.2)中，用wx替换$f(x)$。可以知道，损失函数现在是一个以 w 为变量的的线性方程，$L(wx)$和 w 的关系如图 4.1 所示。

由上图可知，为了拟合出最合理的$f(x)$，程序要使上图中的损失函数尽量小。在程序处理该函数时，会随机取出一个w的值，记作w_0，然后程序会求出当$w = w_0$时损失函数对w的偏导。如上图所示，损失函数在w_0处的偏导为负值，所以为了使损失函数减小，我们用如下公式更新w的值：

$$w_1 = w_0 - \eta \frac{dL}{dw}|_{w=w^0} \tag{4.3}$$

公式（4.3）中的标识符（η）表示学习速率（learning rate），它代表了模型学习的快慢。函数在更新w得到w_1后，会再一次求损失函数在w_1处关于w的偏导，进一步更新模型参数。由图 4.1 可知，该过程在将参数w更新到$w = w_{t1}$时边不在更新，此时w_{t1}称为极小值，但显而易见，最好的损失函数应该是当$w = w_{t2}$时的损失函数，此时w_{t2}称为最小值。我们可以通过“SGD”等方法来防止梯度下降过程陷入极小值。但任何方法都只能减少损失函数陷入极小值的概率，并不能完全避免这种情况。

4.1.3 深度神经网络

深度神经网络是本论文研究的主要内容，它可以看作一种函数的表达方式，图 4.2 是一个最简单的深度神经网络，左边黄色的正方形是输入层，中间的圆形称为隐藏层，每一个圆形称为节点，最右边的黄色正方形称之为输出层。所以该深度神经网络是一个拥有三个隐藏层，每个隐藏层拥有两个节点的网络。在图中可以看到，直线上的数字称之为权重（w）而绿色方块里的数字称之为偏置（b），所以每条直线实际上都表示了一个 $wx+b$ 的函数，深度神经网络将大量的 $wx+b$ 函数线性组合起来，网络便具有了拟合复杂函数的能力。

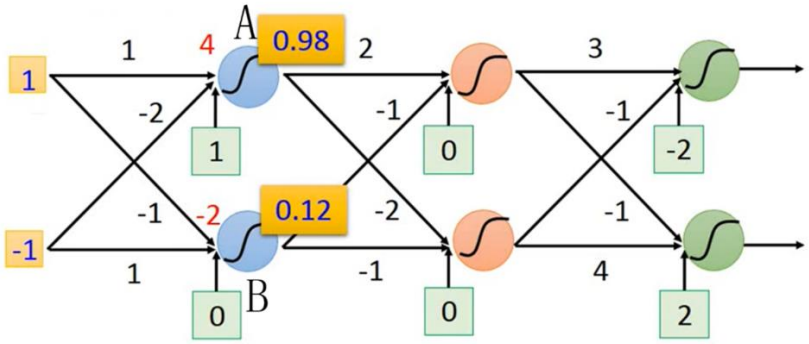

图 4.2 神经网络模型

显而易见，图中节点 A 的值等于 4，而实际工程中一般会为每一个节点附加一个激活函数，以求网络拥有拟合曲线的能力，图中用的是 sigmoid 激活函数，所以最终节点 A 的值是 0.98。关于激活函数的内容将在后文中讲解。

1）激活函数与梯度消失

由 4.1 节可知，神经网络中每一层中每一个节点的值都是由上一层中节点值乘以权重矩阵再加上偏置得出的，在实际使用中，卷积神经网络并不会直接将前一层和权重矩阵的运算结果作为后一层的节点值，而是会将运算结果作为参数传入一个函数中，然后将该函数的值作为后一层的节点值，这个函数就是所谓的激活函数。

激活函数是优化网络的重要手段，由 4.1 节可知，多层神经网络中每一层都是上一层的线性组合，这样无论叠加多少层网络也只具有线性映射的能力，此时引入激活函数才可以使网络具有非线性映射的能力。例如在分类问题中最常用的 sigmoid 函数，sigmoid 函数公式如（4.4）所示。

$$f(x)=\frac{1}{1+e^{-x}} \tag{4.4}$$

在图 4.3 所示的网络中，节点 C 的值根据计算可知是 4，此时节点 C 依旧是上一层中节点 A，B 线性组合的结果，但如果将 4 作为参数传入 sigmoid 函数后节点 C 的值变为了 0.98，此时网络便具有了非线性表达的能力。

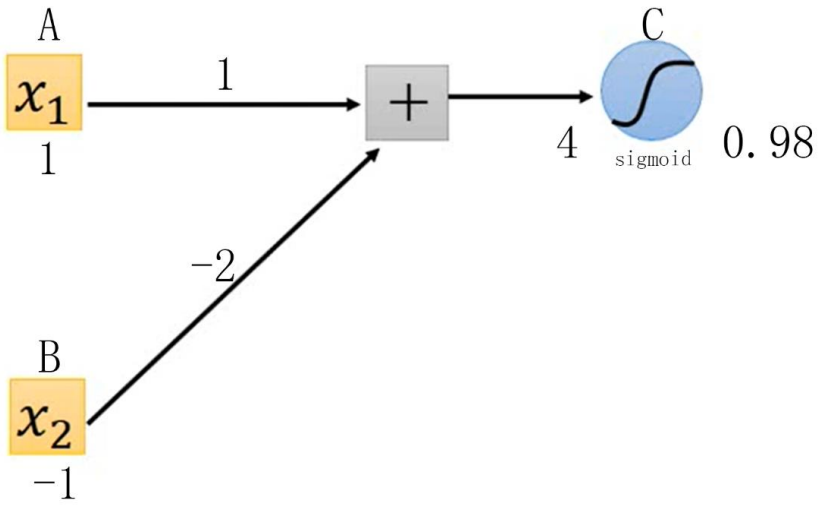

图 4.3 激活函数

在实际工程中 sigmoid 函数的使用有诸多限制，sigmoid 函数的导数图像如图 4.4 所示，可见 sgmoid 函数的梯度在 0 到 0.25 之间，在 5.2.3 节中介绍了回归分析更新参数的主要方法是梯度下降。如果在每个节点上都加入 sgmoid 激活函数，那么 sgmoid 函数的导数也会加入反向传播中进行计算，而 sgmoid 函数的导数是一个小于 1 的值，当多个小于 1 的导数相乘后，最终就会导致偏导变得过于小以至于深度神经网络只有靠近输出层的隐藏层梯度相对正常而靠近输入层的隐藏层梯度更新陷入停滞。这种现象叫做“梯度消失”。在实际使用中，一般使用 sgmoid 激活函数在五层左右就会发生梯度消失的情况，所以 sgmoid 函数在实际工程中使用的很有限。

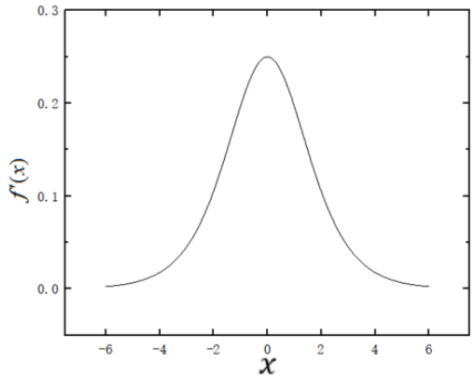

图 4.4 sigmoid 函数的导数

2）反向传播算法

反向传播算法是深度神经网络更新网络权重的主要方式，深度神经网络将输入传入网络得到有误差的输出的过程称为向前传播，将误差作为输入反向传入网络用以更新权重参数的过程称为反向传播。

微积分中的链式求导法则是反向传播的核心理论之一，假设存在两个函数 $z=h(y), y=g(x)$，则链式求导法则如公式（4.5）所示：

$$\frac{dz}{dx}=\frac{dz}{dy}\frac{dy}{dx} \tag{4.5}$$

在回归分析中，梯度下降的第一步是找出损失函数的偏导，在神经网络中则需要求出每一层的偏导，然后更新每一层之间的权值矩阵。图 4.5 所示的是一个简单的深度神经网络图。

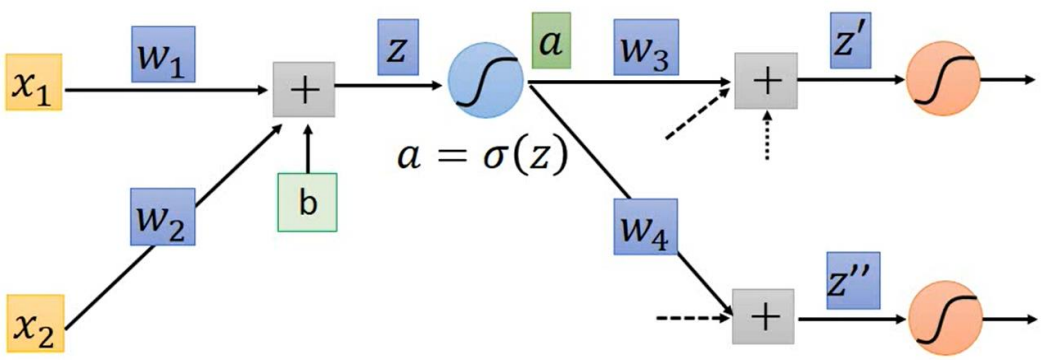

图 4.5 神经网络模型（部分）

假设有 n 个数据，则第 n 个数据对模型的损失函数记作 l^n，总体的损失函数是每个数据损失函数的和，即：$L=\sum_{n=1}^{N} l^n$，所以如果要求损失函数对某一个参数 w 的偏导，只需要将所有 l^n 对 w 的偏导加和即可，即 $\frac{\partial L}{\partial w}=\sum_{n=1}^{N}\frac{\partial l^n}{\partial w}$。

对图 4.5 所示神经网络先向前计算各节点间导数，根据链式法则有：

$$\frac{\partial l}{\partial z}=\frac{\partial l}{\partial a}\frac{\partial a}{\partial z} \tag{4.6}$$

$$\frac{\partial l}{\partial a}=\frac{\partial l}{\partial z'}\frac{\partial z'}{\partial a}+\frac{\partial l}{\partial z''}\frac{\partial z''}{\partial a} \tag{4.7}$$

上式中函数 a 是网络的激活函数，在本例中为 sigmoid 函数，其导数是已知的记作$\sigma'(z)$.

由 4.1 节可知，每一个节点间的连线是一个简单的 $wx+b$ 函数，所以有 $\frac{\partial l}{\partial z'}=w_3$，$\frac{\partial l}{\partial z''}=w_4$。所以公式（4.6）可变换为：

$$\frac{\partial l}{\partial z}=\sigma'(z)\left[w_3\frac{\partial l}{\partial z'}+w_4\frac{\partial l}{\partial z''}\right] \tag{4.8}$$

公式（4.8）实际上是一个如图 4.6 所示的网络，由于节点 C,D 直接连接输出层，所以 $\frac{\partial l}{\partial z'}$ 和 $\frac{\partial l}{\partial z''}$ 是已知的，将其作为输入，可计算出 $\frac{\partial l}{\partial z}$ 值，重复此过程，就可求出模型中每个节点对损失函数的偏导，可以看到，图 4.6 所示的过程是向前传播的逆过程，把损失函数的偏导作为输入，来逐层求出每个节点对于损失函数的偏导，这个过程就是反向传播。

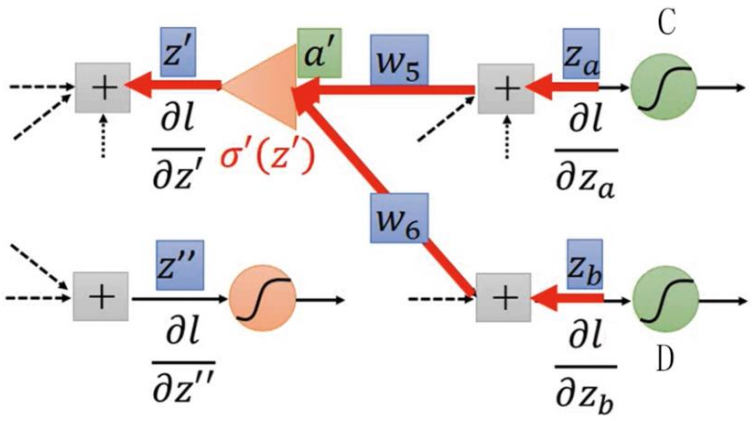

图 4.6 反向传播

4.1.4 卷积神经网络

卷积神经网络是一种由深度神经网络发展而来的神经网络结构，其在图像识别上有着远高于一般深度神经网络的识别率，卷积神经网络与深度神经网络最大的区别在于卷积神经网络引入了卷积层与池化层的概念，这两个结构使得

网络变为了不完全连接，大大降低了网络的参数两，提高计算效率的同时提高了网络的识别率。

1）卷积层与池化层

在卷积神经网络处理卷积层时，会先决定一个卷积核，卷积核是一个任意大小任意形状的数字矩阵，如图 4.7 所示，同时，卷积核还有一锚点，一般情况下，卷积核的锚点位于卷积核的中心。在做卷积时，我们将卷积核划过原始图片，并且使卷积核与图片做卷积，然后将卷积后的数值填入原始图片中卷积核锚点所对应的像素。卷积的完整过程如图 4.8 所示。

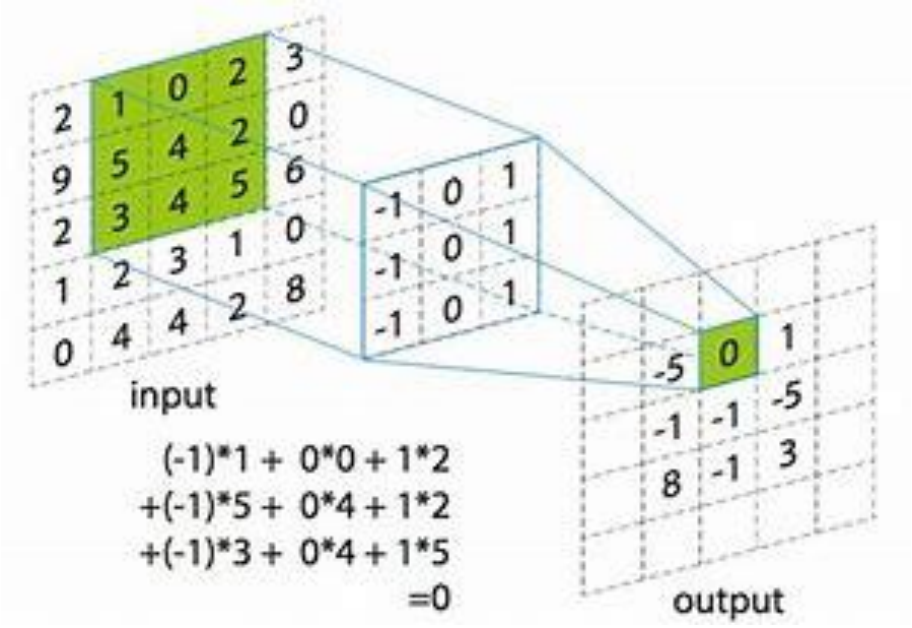

图 4.7 卷积与卷积核

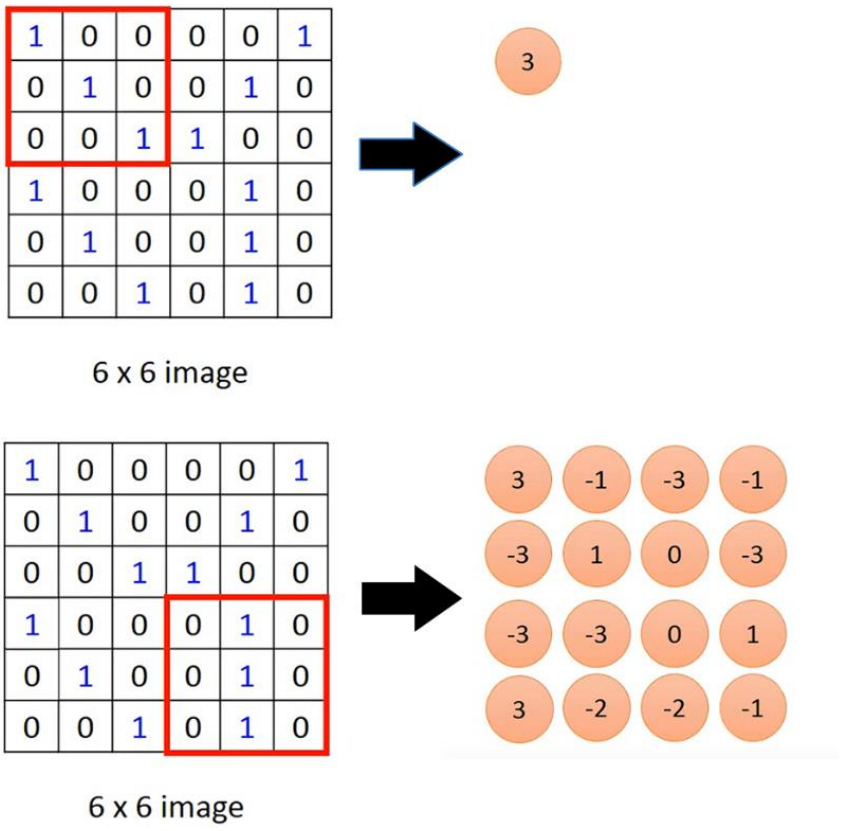

图 4.8 卷积层全过程

可以看到，当卷积核与整幅图片做完卷积后，会生成一张新的图片，这个新的图片要比原来的图片小一点。整个卷积过程可以执行多次。每执行一次，在卷积神经网络中便增加一个卷积层。在图片经过一个到数个卷积层的处理后，图片就会交由池化层处理。

池化层的输入是卷积后的图片，在程序处理池化层时，同样也会生成一个池化核，池化核与卷积核不同，它并不是一个数字矩阵，它只代表一个范围，表示池化层一次池化操作处理的像素的范围。如图 4.9 所示。

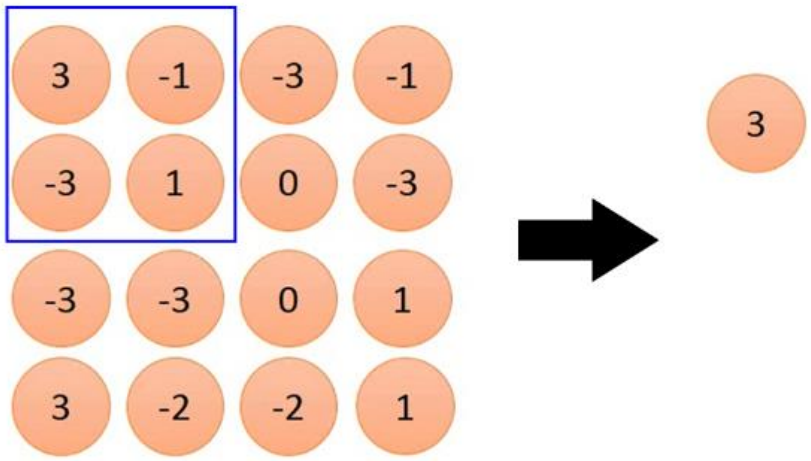

图 4.9 池化核与池化

该图中池化层的大小是一个2×2的正方形面积，在程序进行池化操作时，会如同卷积一样让池化核划过整幅图片，每次池化操作都只对池化核覆盖的像素生效。池化的方法一般有两种分别是 Max Pooling 和 Average Pooling。

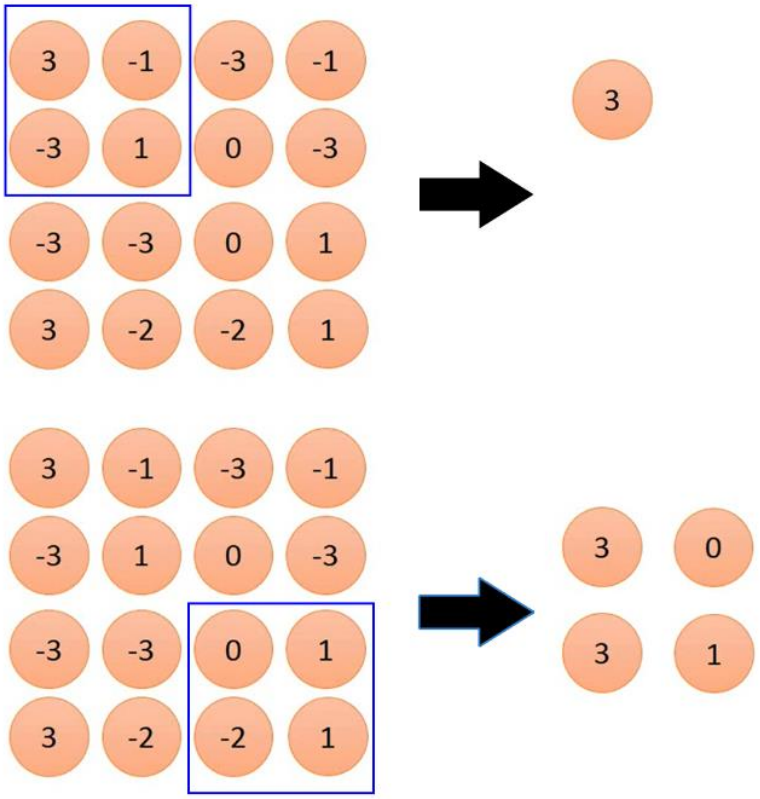

图 4.10 池化层全过程

Max Pooling 指在进行池化操作时，程序从池化核覆盖的像素中取出灰度值最大的一个，然后将每一次池化操作取出的值从新组合成一幅新的图像。图 4.10 展示了 Max Pooling 的全过程。Average Pooling 指每次池化操作时，程序会将池化核覆盖的像素灰度加和再取平均，然后将每次池化操作得到的值取出组成一幅新的图片。与卷积层不同，图片不能连续经过两个池化层。但是，将卷积层和池化层看作一组。卷积神经网络可以有多组这样的卷积池化组合。在本论文提出的程序中，汉字识别网络便是以两个卷积层和一个池化层为一组，一共有三组，也就是六个卷积层和三个池化层。同时本论文涉及的所有模型均采用 Max Pooling 的池化方式。

2）平铺化与全连接层

在卷积神经网络的最后实际上与人工神经网络十分相似，在输入图片经过多个卷积与池化操作后，通常大小已经变得相当小，此时程序将图片的像素排成一列，作为特征输入给后续网络的输入层。这个过程称之为平铺化（flatten）。后续网络的构成实际上与人工神经网络十分相似，在参数经过数个隐藏层后，便会在输入层得到模型的输出。在本论文提出的程序中，卷积后的图片一共会经过两个全连接层，然后便会达到有 31 个节点的输出层。在汉字识别模型中，输出层实际上是 31 个省份及特别行政区汉字简称所对应的概率。

4.2 汉字识别网络与非汉字识别网络详解

4.2.1 字符识别网络概述

本论文提出的程序所涉及的所有字符识别模型均为卷积神经网络模型，其训练框架基于 tensorflow2.1.0 所使用 gpu 加速模块为 cuda9.0。

本论文所使用的字符识别模型一共有两个，分别是汉字字符识别模型与非汉字字符识别模型，两模型存在一定的共通性，本节将会综述性的介绍两种模型。

本论文使用的汉字识别模型一共包括六个卷积层和三个池化层，其中两个卷积层和一个池化层为一组。对于卷积层，它的输入要求是长宽均为 42 像素的单通道图片，所以在将图片传入识别时，需要先调整大小。卷积层所使用的卷积核是3×3大小的数字矩阵，激活函数是“relu”。 关于池化层，池化层的池化核为一

个大小为2 × 2的矩阵，池化方法为“MAXPOOLING”。最后，经过 8 个卷积层和 4 个池化层处理的图片会被做平铺化处理，即“Flatten”,平铺后的图像会接入一个拥有 512 个节点的全连接层，该层的激活函数为“relu”。然后该全连接层与一个拥有 31 个节点的输出层相连，这 31 个节点代表了我国 31 个省级行政单位的简称。该输出层的激活函数为“softmax”.

本论文使用的非汉字字符识别模型共包含 6 个卷积层和 4 个池化层，其中两个卷积层和一个池化层为一组，一共四组。对于卷积层，它的输入要求是长宽均为 32 像素的单通道非汉字字符图片。卷积层所用的卷积核是 3X3 的数字矩阵。激活函数是“relu”,池化层层的池化核大小和池化方式与汉字字符识别模型一致。经过卷积和池化层的图片会被传入有 1024 个节点的全连接层，全连接层的激活函数是“relu”,然后传入有 34 个节点的输出层，这 34 个节点对应我国车牌的 34 个非汉字字符。

最后，本模型在训练时使用的优化器为“adam”,使用的损失函数为“cross entrop”即交叉熵函数。

4.2.2 数据集的划分

在开始模型训练之前，需要先将用于训练的数据集划分为训练集和测试集，训练集的作用是训练模型调整权重矩阵，测试集的作用是评估调整参数后模型对于初次遇到的数据的实际表现。

根据规定，我国车牌的非汉字部分由 0-9 十个阿拉伯数字和 A-Z24 个英文字符组成（为了防止混淆，我国车牌不使用 O 和 I），所以，本论文提出的程序为每个非汉字字符生成了一个独热编码。当字符图片传入网络模型后，网络模型将返回每个独热编码的置信度作为预测结果。

在本程序中，非汉字字符识别模型的训练数据是一个分割好的数字或英文的单字符图片，经统计，该数据集包含 22529 张图片，其中 13080 张作为训练集，9538 张图片作为测试集，图 4.11 展示了数据集的切片。

图 4.11 非汉字字符数据集切片

我国车牌的汉字部分是由中国 31 省份及直辖市的汉字简称组成的。同样，本论文提出的程序为每一个汉字设置了一个唯一的独热编码。在图片传入网络模型后，模型将返回每个独热编码的置信度作为预测结果。

本论文中汉字字符识别模型的数据集由已经二值化后的汉字单字符图片组成，其主要由对原始车牌图片分割后得到，因此其大小各异，需要由函数将其转化成（42,42）像素的统一大小后才能被网络使用。本论文所使用的中文汉字数据集总共有 16242 张图片，其中 11031 张图片被用作训练集，5290 张图片被用作测试集。

图 4.12 展示了数据集的切片。

图 4.12 汉字字符数据集切片

4.2.3 训练字符识别模型

本论文提出的程序采用的数据读取方式是：先将图片按照分类存入不同的文件夹中，然后将文件夹中的文件作为训练数据，文件夹名称作为真值，将两者组合为 map 后读入程序，等待所有训练数据读入后将所有数据打乱后输入模型进行训练。本方法的优点是不需要人工标注图片标签，本论文使用的模型所使用的数据集数量巨大，人工标注图片标签无法做到，本方法节省了大量处理前置工作的时间。

本论文提出的程序在训练汉字识别模型时将训练数据分为 64 个数据一组，每次训练共使用 250 组数据，一共迭代 10 次，在 10 次迭代训练后，模型在训练集上的损失函数值为 0.0181，正确率为 99.50%。在测试集上的损失函数为 0.正确率为 95.31%。表 4.3 展示了汉字识别模型的结构，表 4.4 展示了模型的训练结果。

表 4.3 汉字字符识别模型结构

	输出大小	卷积核个数	单层参数数目
卷积层 1	（42,42）	16	160
卷积层 2	（42,42）	16	2320
池化层	（21,21）	—	0
卷积层 3	（21,21）	32	4640
卷积层 4	（21,21）	32	9248
池化层	（10,10）	—	0
卷积层 5	（10,10）	64	18496
卷积层 6	（10,10）	64	36928
池化层	（5,5）	—	0
卷积层 7	（5,5）	128	73856
卷积层 8	（5,5）	128	147584
池化层	（2,2）	—	0
平铺层	（1,512）	—	—
全连接层	—	—	65664
输出层	（1,31）	—	3999

表 4.4 汉字字符识别模型训练结果

迭代次数	测试集准确率	测试集损失函数	验证集准确率	验证集损失函数
1	73.98%	0.9778	89.06%	0.6237
2	97.76%	0.0926	91.64%	0.2544
3	98.72%	0.0484	95.31%	0.0799
4	98.58%	0.0468	96.88%	0.1338
5	99.19%	0.0302	93.75%	0.2932
6	98.85%	0.0434	96.88%	0.2061
7	99.11%	0.0351	96.88%	0.0470
8	99.16%	0.0294	98.44%	0.0312
9	99.08%	0.0365	95.31%	0.0576
10	99.61%	0.0150	98.44%	0.0316

本论文提出的程序在训练非汉字识别模型时将训练数据分为 64 个数据一组，每次训练使用 300 组数据，一共迭代 10 次，在五次迭代后模型在训练集上的损失函数的值为 0.0272，准确率为 99.40%，在测试集上的损失函数的值为 0.0189，准确率为 98.44%。表 4.5 展示了非汉字识别模型的结构，表 4.6 展示了模型的训练结果。

表 4.5 非汉字字符识别模型结构

	输出大小	卷积核个数	单层参数数目
卷积层 1	（32,32）	16	160
卷积层 2	（32,32）	16	2320
池化层	（16,16）	—	0
卷积层 3	（16,16）	32	4640
卷积层 4	（16,16）	32	9248
池化层	（8,8）	—	0
卷积层 5	（8,8）	64	18496
卷积层 6	（8,8）	64	36928
池化层	（4,4）	—	0
平铺层	（1,1024）	—	—
全连接层	（1,1024）	—	1049600
输出层	（1,34）	—	34850

表 4.6 非汉字字符识别模型训练结果

迭代次数	测试集准确率	测试集损失函数	验证集准确率	验证集损失函数
1	91.57%	0.4076	98.40%	0.9157
2	99.45%	0.0224	98.00%	0.1209
3	99.53%	0.0177	99.60%	0.0177
4	99.75%	0.0088	98.88%	0.0664
5	99.97%	0.0010	99.60%	0.0526

4.3 本章总结

本章一共分为两个主要部分，主要讲解了神经网络的基本原理并且展示了本程序所用模型的详细内容。

1）本章详细介绍了神经网络的基本原理，并且引出了神经网络中重要的分析算法“回归分析”，在 4.1.2 中，本文对回归分析原理以及回归分析中最重要的算法“梯度下降算法”的计算过程进行了详细讲解。

2）本文在神经网络的基础上介绍了深度神经网络的原理，介绍了输入层，隐藏层，输出层，权值矩阵，偏置等深度神经网络的重要组成部分，并且解释了反向传播算法的理论依据。

3）本章在深度神经网络的基础上介绍了本论文的核心理论“卷积神经网络”，在 4.1.4 中首先对卷积神经网络与传统神经网络的区别进行了简要分析，然后介绍了卷积层和池化层的运算过程，并且结合本论文所用的模型对卷积神经网络整体结构进行了讲解。

4）本章的第二部分是对本论文用到的两种卷积神经网络的详细介绍，展示了两种模型所用的训练数据集，然后对两中模型的结构进行了介绍，在 4.2 节最后，本文对两种模型的训练结果进行了展示。

第五章 系统实现与实验结果

本论文结合图像预处理，车牌定位，车牌字符分割，车牌字符识别四个部分，综合性的设计了一个完整的车牌识别程序，该程序不仅能完成车牌识别的所有需求，同时拥有完整的交互界面，可以根据用户选择使用不同的识别模式，并且在界面上展示识别结果。本章将会详细介绍本程序所用的各项依赖库，并且会对程序的部分核心实现代码进行展示，在本章的最后对程序不同识别模式的结果进行了展示。

5.1 程序的搭建环境

本论文提出的程序采用了 C++与 Python 语言的混合编程，主要使用的集成开发工具是 Visual Studio 2019, Python 版本为 3.7，集成环境管理器为 Anaconda3.1，程序所使用的计算机图形库为 Opencv4.2,神经网络训练框架是 tensorflow2.1，界面开发框架为 Qt5.13。

1）Visual Studio 2019 是由微软公司开发的集成开发环境，其功能强大支持 C++，C#，Java，Python 等语言。同时 VS 提供详细的错误列表反馈程序 BUG，配备堆栈，内存等资源监视窗口，方便程序编写者实时监控系统资源状况，此外 vs 还提供单步调试和实时的变量监控，帮助程序编写者监控程序中每一条代码执行后变量的变化情况。

2）Anaconda 是一个集成的 Python 环境管理器，它不仅提供了最新版本的 Python 环境，还集成了包括 numpy 在内的 100 余个数据科学库，同时也可以在 Anaconda 上安装 Jupyter Notebook，spyder 等开发环境。

3）Opencv4.2 是一个开源计算机视觉库，它支持包括 C++在内的多种语言。Opencv 提供了较为完善的计算机图像处理函数接口，本论文提出的程序中 Mser 定位算法，色彩空间转换算法，OTSU 阈值算法等函数均调用自 Opencv 视觉库。同时，高版本的 OPENCV 提供了简单的神经网络算法，本程序所使用的 SVM 分类器的训练算法便是引用自 Opencv4.2。

4）tensorflow 是一个由谷歌开发的开源神经网络算法库，其包含完善的神经网络训练算法，可以训练包括 ANN,CNN 在内的多种神经网络模型，同时它也集成了简单的图像处理算法，例如灰度化，二值化等，方便程序处理输入数据。在最新版本的 tensorflow 中已经集成的另一个神经网络算法库 keras,本论文所使用的所有卷积神经网络模型均是由 Tensorflow 中的 KerasAPI 训练而成。

5）Qt 是一个由 Qt Company 开发的 C++图形界面开发框架。作为一款图形界面开发框架，Qt 功能齐全，编写简单。同时 Qt 与本程序使用的集成开发工具 VS2019 兼容性良好，Qt 可以直接作为扩展程序插入 VS2019 的界面中。而且 VS2019 下建立的 Qt 工程可以直接与 C++程序进行混合编译，节省了时间成本。

5.2 核心程序展示

1）本文程序中核心类为 Plate 类，该类存储了车牌的所有信息，包括车牌的图像，姿态，识别结果等信息。图 5.1 展示了 Plate 类的代码构成。

```
class Plate {
public:
    Plate() {
        Confidence = -1;
        PlateStr = "";
    }

    Plate(const Plate& other) { ... }

    Plate& operator=(const Plate& other) { ... }

    inline void setPlateMat(Mat TP) { PlateMat = TP; }
    inline Mat getPlateMat() const { return PlateMat; }

    inline void setPlatePosition(RotatedRect TP) { Position = TP; }
    inline RotatedRect getPlatePosition() const { return Position; }

    inline void setPlateStr(String TP) { PlateStr = TP; }
    inline String getPlateStr() const { return PlateStr; }

    inline void setPlateScore(double TP) { Confidence = TP; }
    inline double getPlateScore() const { return Confidence; }

    bool operator < (const Plate& TP) const { return (Confidence < TP.Confidence); }
    bool operator > (const Plate& TP) const { return (Confidence > TP.Confidence); }
    bool operator = (const Plate& TP) const { return (Confidence == TP.Confidence); }
private:
    Mat PlateMat;

    RotatedRect Position;

    String PlateStr;

    double Confidence;
};
```

图 5.1 Plate 类

在 Plate 类中，PlateMat 参数存储了车牌的图像信息，他是一个经由车牌定位函数得到的三通道 RGB 图片，该参数储存了完整的车牌图像。Position 参数储存了一个矩形框点阵数据，它记录了车牌的位置信息，该矩形框点阵的表示方式采用了依据原始车辆图片得出的相对坐标，矩形框在原始车牌图像中所在的位置就是车牌所在的位置。PlateStr 参数是一个字符串数据，它记录了车牌经过识别后的识别结果。Confidence 参数是一个浮点型数据，它记录了该车牌类中的 PlateMat 图像在 SVM 分类器中得到的置信度，该参数主要用于筛选最佳车牌区域。

为了方便管理函数，本程序的所有车牌定位函数均被封装在一个名为 PlateLocat 的类中，PlateLocat 类的声明如图 5.2 所示。

```
class PlateLocate {
public:
    PlateLocate() {};

    static PlateLocate* instance();

    int ColorLocate(Mat src, std::vector<Plate>& Plate);
    int ColorSearch(const Mat& src, Mat& out, std::vector<RotatedRect>& Rect);

    int MserLocate(Mat src, std::vector<Plate>& Plate);
    int MserSearch(const Mat& src, vector<Mat>& out, std::vector<RotatedRect>& Rect);

    bool rotation(Mat& in, Mat& out, const Size rect_size, const Point2f center, const double angle);

    static const int WIDTH = 135;
    static const int HEIGHT = 36;
    static const int TYPE = CV_8UC3;

protected:
    static PlateLocate* instance_;
};
```

图 5.2 PlateLocate 类

PlateLocate 类不存在私有变量，仅拥有两种车牌定位函数以及车牌校正函数的函数接口，在 PlateLocate 函数中，ColorSearch 是颜色定位算法的核心函数，实现了颜色定位算法的主要功能 ColorLocate 函数是颜色定位算法的主函数，整合了颜色定位算法的核心函数以及候选区域筛选等数据处理函数。MserSearch 函数是 Mser 定位算法的核心函数，实现的 Mser 定位算法的主要功能，MserLocate 函数同 ColorLocate 函数类似，整合了 Mser 定位算法的核心函数和一些数据处理函数。Rotation 函数是车牌校正函数，主要负责将有倾角的

车牌图片调整回水平。同时 PlateLocate 类还提供了一个静态指针 instance_，可以方便程序中非成员函数调用 PlateLocate 类的成员函数。

3）本程序的字符识别模型训练函数和字符识别函数均是使用 Python 语言编写的，在字符测试函数中，主程序会调用并将需识别的字符传给 Python 函数，Python 函数会载入模型并识别字符，然后将识别结果返回主程序。在主程序的库文件中并不包含字符识别函数和字符识别模型。图 5.3 展示了字符识别函数的调用过程，图 5.4 展示了使用 Python 编写的非汉字字符识别函数，图 5.5 展示了汉字字符识别函数。

```
std::pair<std::string, std::string> CharClassify::Classify2(cv::Mat input)
{
    std::string strImagePath = "temporary_image2.jpg";
    imwrite(strImagePath.c_str(), input);
    std::string path = "E:/personal_file/test/test6/QtGuiApplication1/QtGuiApplication1";
    path = path + "/" + strImagePath;
    Py_SetPythonHome(L"E:/boot_file/anaconda/envs/tensorflow");
    Py_Initialize();
    if (!Py_IsInitialized())
    {
        std::cout << "fail to initial!" << std::endl;
        Py_Finalize();
    }
    PyRun_SimpleString("import sys");
    PyRun_SimpleString("sys.path.append('E:/personal_file/test/test4/PythonApplication1/PythonApplication1')");
    PyObject* pModule = NULL;
    PyObject* pFunc = NULL;
    PyObject* pParams = NULL;
    PyObject* pResult = NULL;
    pModule = PyImport_ImportModule("PythonApplication1");
    if (pModule == NULL)
    {
        std::cout << "don't find the python file!" << std::endl;
    }
    pFunc = PyObject_GetAttrString(pModule, "char_reco");
    pParams = Py_BuildValue("(s)", path.c_str());
    pResult = PyEval_CallObject(pFunc, pParams);
    std::ifstream first_file("E:/personal_file/test/test4/PythonApplication1/PythonApplication1/temporary_file.txt");;
    std::string first;
    getline(first_file, first);
    first_file.close();
    return std::make_pair(first, first);
}
```

图 5.3 Python 函数调用过程

```
import matplotlib as mpl
import matplotlib.pyplot as plt
import numpy as np
import sklearn
import pandas as pd
import os
import sys
import time
import tensorflow as tf
from tensorflow import keras
def char_reco(path):
    character=['0','1','2','3','4','5','6','7','8','9','A'
             ,'B','C','D','E','F','G','H','J','K','L','M'
             ,'N','P','Q','R','S','T','U','V','W','X','Y','Z']
    model=tf.keras.models.load_model("E:/personal_file/test/test4/PythonApplication1/PythonApplication1/my_model.h5")
    img_string=tf.io.read_file(path)
    img_decode=tf.io.decode_jpeg(img_string)
    img_resize=tf.image.resize(img_decode,[32,32])
    img = np.array(img_resize)
    img = img.reshape((-1,32,32,1)).astype('float32')
    y_pre = model.predict(img)
    maxindex  = np.argmax(y_pre)
    maxindex_int=int(maxindex)
    filename = 'E:/personal_file/test/test4/PythonApplication1/PythonApplication1/temporary_file.txt'
    with open(filename,'w') as f:
        f.write(character[maxindex_int])
```

图 5.4 非汉字字符识别函数

```
import matplotlib as mpl
import matplotlib.pyplot as plt
import numpy as np
import sklearn
import pandas as pd
import os
import sys
import time
import tensorflow as tf
from tensorflow import keras
def char_reco(path):
    province=['川','甘','黑','津','辽','闽','琼','晋','新','粤','浙'
             ,'鄂','贵','沪','京','鲁','宁','陕','皖','豫','云','赣'
             ,'桂','冀','吉','蒙','青','苏','湘','瑜','藏']
    model=tf.keras.models.load_model("E:/personal_file/test/test2/PythonApplication2/PythonApplication2/my_model.h5")
    img_string=tf.io.read_file(path)
    img_decode=tf.io.decode_jpeg(img_string)
    img_resize=tf.image.resize(img_decode,[32,32])
    img = np.array(img_resize)
    img = img.reshape((-1,32,32,1)).astype('float32')
    y_pre = model.predict(img)
    maxindex  = np.argmax(y_pre)
    maxindex_int=int(maxindex)
    filename = 'E:/personal_file/test/test2/PythonApplication2/PythonApplication2/temporary_file.txt'
    with open(filename,'w') as f:
        f.write(str(maxindex_int))
    filename2 = 'E:/personal_file/test/test2/PythonApplication2/PythonApplication2/temporary_file2.txt'
    with open(filename2,'w') as f:
        f.write(province[maxindex_int])
```

图 5.5 汉字字符识别函数

5.3 程序测试及结果展示

本程序为用户提供了详细的数据展示界面，在程序界面中不仅会展示识别的结果，同时也会展示程序识别过程中单步的效果图片，包括图片灰度化，车牌定位，字符分割和字符识别。图 5.6 展示了本程序的运行结果。

图 5.6 识别结果展示

本程序对 20 组车牌数据进行了批量测试，结果如表 5.1 所示，总体识别率达 90%。

表 5.1 程序批量测试结果

原车牌	识别结果
沪 KT5583	**沪 KT5583**
鲁 Y44748	**鲁 Y44748**
粤 B3RS91	**粤 B3RS91**
苏 A36E80	甘 A36580
苏 A75976	**苏 A75976**
皖 A22T43	**皖 A22T43**
湘 G60009	**湘 G60009**
粤 AF9C00	**粤 AF9C00**
苏 A9YP07	**苏 A9YP07**

表 5.1 程序批量测试结果（续）

原车牌	识别结果
鲁 LD9016	鲁 LD9016
浙 F397C0	浙 F397C0
皖 AX688A	皖 AX688A
豫 K91239	豫 K91239
粤 A84266	粤 A84266
粤 B5PQ23	粤 B5PQ23
浙 E6686V	浙 E6686V
浙 C01701	浙 CD1701
苏 A20Q03	苏 A20Q03
粤 BR75Y3	粤 BR75Y3
皖 A1T235	皖 A1T235

5.4 本章总结

本章的主要内容分为三个部分，分别介绍了本论文提出程序的搭建环境，核心代码实现以及程序的测试结果。

1）本章介绍了本程序所有所需的依赖库和工具，以及这些依赖库及工具在本论文中的作用，并且分析了这些依赖库和工具的优点，阐述了选择它们的理由。

2）本章从系统结构和系统实现两个方面介绍了本系统的部分程序实现，并且展示了部分程序源代码。

3）本章在最后展示了系统的界面，展示了单张图片的测试结果，并且利用 20 张车牌图像测试了系统的识别率。

第六章 总结与展望

6.1 本课题的主要工作

本课题是关于基于卷积神经网络的车牌识别系统的研究，本课题对车牌识别中从原始车辆图片获取和图像预处理到车牌字符识别并输出的全部过程进行了理论研究并且进行了算法实现，完成了一个完整的车牌识别系统。在本论文中，着重对图片预处理，车牌定位，字符分割和字符识别四部分进行了理论分析和程序实现。

在图片预处理中，本论文提出的程序完成了对原始车辆图片的去噪灰度化处理，完成了基于 OTSU 的二值化处理以及形态学操作的程序实现。

在车牌定位中，本论文采用了颜色定位算法和 mser 算法协同使用的方案实现了对车牌的定位与分割工作，总体车牌检出率达到 90%以上。

在车牌分割中，本论文旨在期望将完整的车牌分割成七个单字符图片以供字符识别程序使用，在具体实现中，本论文提出的程序将汉字字符与非汉字字符分开处理，利用滑动窗口算法实现了对汉字字符的切割，利用基于像素跃变点的边缘检测算法实现了对于非汉字字符的分割。

在字符识别中，本论文提出的算法利用汉字字符识别模型和非汉字字符识别模型这两个不同的卷积神经网络模型实现了对于车牌字符的识别工作，在测试结果中，车牌识别准确率达到了 90%以上。

6.2 未来的工作与展望

本论文提出的程序在常规情况下已经有了可观的正确率，然而本论文依旧存在一些不足和未完成的工作：

1）.更优秀的图像预处理算法。本论文提出的程序测试的均是在正常路况下的车辆照片，一旦遇到严重背光，大雾等极端天气和光照条件，识别的准确率就会大幅下降，所以本论文期望能找到更优秀的图像预处理算法，将恶劣环境下的图像处理成能被程序使用的图像。

2）.利用卷积神经网络定位车牌。本论文提出的程序在实现车牌定位时依旧采用的是传统图像处理算法，本论文期望能将卷积神经网络加入车牌定位算法中以进一步提高车牌的检出率。

3）.更合理的的网络结构。卷积神经网络如果只是将卷积层和池化层不断堆叠起来，必然会导致梯度消失，进而引起网络性能退化[10][14]，如果在网络结构上对网络进行优化，如“ResNet”[16][18]等网络结构，则有希望进一步提高车牌识别的准确率。

4）识别更多种车牌，本论文提出的是针对我国小型车牌的识别程序，并不具备其他类型车牌的识别能力，期望在未来能扩充更多车牌的识别种类。

参考文献

[1] Nobuyuki Otsu. A Threshold Selection Method from Gray-Level Histograms[C]. //IEEE Transactions on Systems, Man, and Cybernetics. Piscataway, NJ:IEEE Press, 1979:62-66.

[2] 张涛, 王剑魁, 张国山, 等. 多雾霾天气车牌识别系统图像预处理算法[J]. 山东理工大学学报(自然科学版), 2015, 29(01):15-19.

[3] 沈勇武, 章专. 基于特征颜色边缘检测的车牌定位方法[J], 仪器仪表学报, 2008, 29(12):2673-2677.

[4] Bai Hongliang, Liu Changping, A hybrid license plate extraction method based on edge statistics and morphology[C]//Proceedings of the 17th International Conference on Pattern Recognition 2004. Piscataway, NJ:IEEE Press, 2004:831-834.

[5] Yuan Yule, Zou Wenbin, Zhao Yong, et al. A Robust and Efficient Approach to License Plate Detection[C]//IEEE Transactions on Image Processing. Piscataway, NJ:IEEE Press, 2017:1102-1114.

[6] Chao Gou, Kunfeng Wang, Yanjie Yao, et al. Vehicle license plate recognition based on extremal regions and restricted Boltzmann machines[C]//IEEE Transactions on Intelligent Transportation Systems . Piscataway, NJ:IEEE Press, 2016:1096 - 1107

[7] 朱倩倩, 刘森, 郭维明. 基于 Faster R-CNN 的车牌检测研究[J]. 汽车工业研究, 2019(01):57-60.

[8] 贺智龙, 肖中俊, 严志国. 基于 HSV 与边缘信息的车牌定位与字符分割方法[J]. 齐鲁工业大学学报, 2019, 33(03):44-48.

[9] 邓运生, 郑晨霞, 尹安. 车牌定位和字符分割方法对比研究及实现[J]. 兰州文理学院学报(自然科学版), 2019, 33(06):78-83.

[10] 焦娜. 基于软 K 段主曲线的 LPR 字符特征的提取方法[J]. 计算机科学, 2017, 44(09):49-52.

[11] 郑泽鸿. AP 聚类算法在车牌字符分割中的应用与研究:[D]. 贵州:贵州民族大学, 2018.

[12] 李飞. 基于人工神经网络的车牌识别系统的研究:[D], 山西:中北大学, 2007.

[13] Yujie Liu, He Huang,Jinde Cao, Tingwen Huang. Convolutional neural networks-based intelligent recognition of Chinese license plates[J]. Springer Berlin Heidelberg, 2018, 22(7).

[14] 訾晶, 张旭欣, 金婕. 基于 FPGA 动态重构的快速车牌识别系统[J]. 传感器与微系统, 2019, 38(12):69-72.

[15] 杨涛, 张森林. 一种基于 HSV 颜色空间和 SIFT 特征的车牌提取算法[J]. 计算机应用研究, 2011, 28(10):3937-3939.

[16] 高攀. 基于卷积神经网络的车牌识别关键技术的研究与应用[D]. 北京:北京邮电大学, 2019.

[17] 周志华. 机器学习[M]. 北京:清华大学出版社. 2016:121-22.

[18] 王元东. 基于 ResNet 模型的图像分类方法及应用研究[D]. 北京:北京交通大学, 2019

致谢

本课题《基于卷积神经网络的车牌识别》所涉及的领域是本人先前从未涉及的，能够完成程序的设计并最终完成论文的撰写令我十分欣喜，回顾在毕业设计期间遇到的困难，真是令我感慨万千。在完成论文期间遇到的诸多困难，没有苏老师的指导，仅依靠我个人的努力想必是绝对无法克服的。

虽然进行毕业设计的期间仅有短短不足一年，但是在毕业设计期间学到的知识与克服困难的精神令我难以忘怀。相信这些记忆会在我以后的生活中不断激励我前进。

再次感谢苏祖强老师从知识和思想上的教导，您教会我的东西，我会终身铭记，师恩永存。

附录 英文翻译

Lcense plate detection and recognition using hierarchical feature layers from CNN.

Qiang Lu1 & Yu Liu2 & Jing Huang2 & Xiaohui Yuan3 & Qingxin Hu2

Abstract

In recent years, a variety of systems using deep convolutional neural network (CNN) approaches have achieved good performance on license plate detection and character recognition. However, most of these systems are not stable when the scenes changed, specification of each hierarchical layer to get the final detection result, which can detect multi-scale license plates from an input image. Meanwhile, at the stage of character recognition, data annotation is heavy and time-consuming, giving rise to a large burden on training a better model. We devise an algorithm to generate annotated training data automatically and approximate the data from the real scenes. Our system used for detecting license plate achieves 99.99% mean average precision (mAP) on OpenITS datasets. Character recognition also sees high accuracy, thus verifying the superiority of our method.

Keywords： License plate detection . Character recognition . Hierarchical feature layers . Generated training data

Introduction

Over the past few years, thanks to the popularity and progress of accident monitoring technology, license plate detection and character recognition have received increasing attention around the world. Convolution neural networks (CNNs) have resulted in a much-improvedperformance in many tasks, such as face detection and recognition [22], and take a new perspective to the problem of license plate detection and recognition. However, this task remains challenging. At the stage of license plate detection, the current state-of-the-art systems using CNNs cannot solve the problem well with different scales of license plates while the camera is fixed at different angles and heights , especially since detecting small license plates is difficult [17, 20, 26]. For the sake of overcoming this deficiency, we adopt a hierarchical mechanism for license plate detection. The bottom layers have small receptive fields, which used for small license plates detection, and the topper layers have larger receptive fields, which used for larger license plates detection. For each hierarchical layer, a set of specific size default boxes with different aspect ratios and scales are associated with each map cell. During prediction, the network generates confidence scores and offsets of each default box to identify and bound license plates [23]. According to multiple feature layers with different resolutions, plates with various sizes are detected automatically. Besides, the network designed in previous work requires the size of the video frame as the input of a CNN model is fixed, which leads to inevitable image distortion [13]. To solve the problem, the license plate detection network we adopt is a kind of fully convolutional network (FCN) [9], which can take arbitrary-sized images as inputs [18]. Experimental results show that our method can detect license plates of arbitrarily sizes with very high accuracy. For character recognition, data annotation by artificial means for a large amount of training data is heavy and time-consuming, giving rise to a large burden on training a better model. For the sake of efficiency at this stage, an algorithm is designed to automatically generate annotated license plates, with varying degrees of noise and rotation to approach the training data from the real scenes. It is worth noting that our method can

generate any license plate, such as police car license plate, tractor license plate. Our method has confirmed its effectiveness in practice, along with high accuracy.

Our contributions in this work are summarized as follows:

1)We propose CNN-based hierarchical feature layers to detect multi-scale license plates from the input image.

2)The network is fit for an arbitrary-sized image as inputs to avoid image distortion.

3)We propose an algorithm to generate annotated license plates automatically for character recognition, reducing the cost of model training.

2.Related work

License plate detection Methods for license plate detection are based on four categories: the edge information, the texture information, the color information, and the machine learning. The method based on the edge information, using the edge feature of a license plate to segment connected areas, obtains the candidate region of the license plate, and then the false license plate is removed by the gradient of the candidate region and entropy [15, 20, 26]. Such methods heavily rely on the edge information of the image and face difficulty in unconstrained environments. Regarding the method based on texture information [4, 16, 17], Wu et al. [27] propose a method to detect license plate candidates by using morphological operations of bottom-hat and morphology gradients and then choose the right candidate. However, this method requires input images with high quality, and involves significant computation; thus, it is difficult to realize real-time calculations. The method based on color information segments the input image by using color information which removes most of the background and uses texture analysis to achieve the final license plate location [1, 10]. However, because the color information is unstable, this method may be affected by illumination. For the method based on machine learning, Dlagnekov et al. [11] extract the Haar-like features, then detect the license plate by using the Adaptive boosting algorithm to train the cascade classifier. This method is used for license detection with very high accuracy, but the classifier only applies to fixed-size images, there is no scale invariance. Support vector machine can solve the problem of small sample size, nonlinear and high dimensional pattern recognition [6, 12]. Zhang et al. [29] use the edge information of the input image, determine the initial candidate regions, and then extract the covariance of the candidate region (Covariance Feature), finally training the linear SVM classifier to eliminate the false license plate. In addition, the method based on neural networks is also widely used in this study [8, 21]. However, artificial features often have a low differentiating degree, which limits the application of this approach in practice.

FCN FCN has been successfully used for semantic segmentation since it is

proposed by Long et al. [19]. They adopt modern classification networks into an FCN, and deconvolution layers are used for up-sampling the feature maps into image size. It is beneficial for extending these classification nets to segmentation. The pixel-wise prediction is made for semantic segmentation in up-sampling layers. In addition, earlier layers are combined with the latter layers to detect more details for more precision. We are inspired by this idea [7] and use the earlier layers to detect the smaller plates and bottom layers to detect the larger plates. Furthermore, FCNs support arbitrary-sized image inputs to avoid image distortion.

Region proposal Many studies have been done on region proposals, such as MCG [3],Selective Search [25], and sliding windows [2, 28, 30]. Girshick et al. [13] combine Selective Search with CNNs for generating region proposals and improve mean average precision (mAP) dramatically compares to the best results of previous work on VOC2012. However, Selective Search is an independent part of object detection, it will waste a lot of time. Ren et al. [23] propose Region Proposal Networks (RPNs), which computing proposals with a DCNN and can be trained end-to-end to generate region proposals. Liu et al. [18] implement an end-to-end training mechanism, which is much faster than Faster R-CNN, and more mAP than previous work. We follow this idea for license plate detection.

3 Module design

The method we proposed consists of two components: license plate detection and character recognition. A CNN is trained to detect license plates in the input video frames and a second network is used for character recognition. Figure 1 illustrates the architecture of our system.

3.1 License plate detection

Hierarchical Feature Layers The pipeline of our license plate detection network is shown in Fig. 2. The early network layers designed before conv3_3 are based on VGG net [24], and these layers are used to encode semantic information. We then add an auxiliary structure, the conv3_3, conv4_3, and conv5_3, which are used for license plate detection by predicting confidence scores and box offsets. Then, we integrate the information from these three layers using the Non-Maximum Suppression (NMS) algorithm to obtain the final candidate license plate and its position. Conv3_3 has much finer spatial details and is used for small license plate detection. Meanwhile, conv5_3 has larger receptive fields, which benefits detection of larger license plates. Using our structure, we easily detect license plates with different scales, especially small license plates. Next, we add supporting structure into the training network, such as batch normalization, which improves the mAP and regularizes the model. In our experiments, we obtain 1% improvements by adding batch normalization. It also helps speed up convergence.

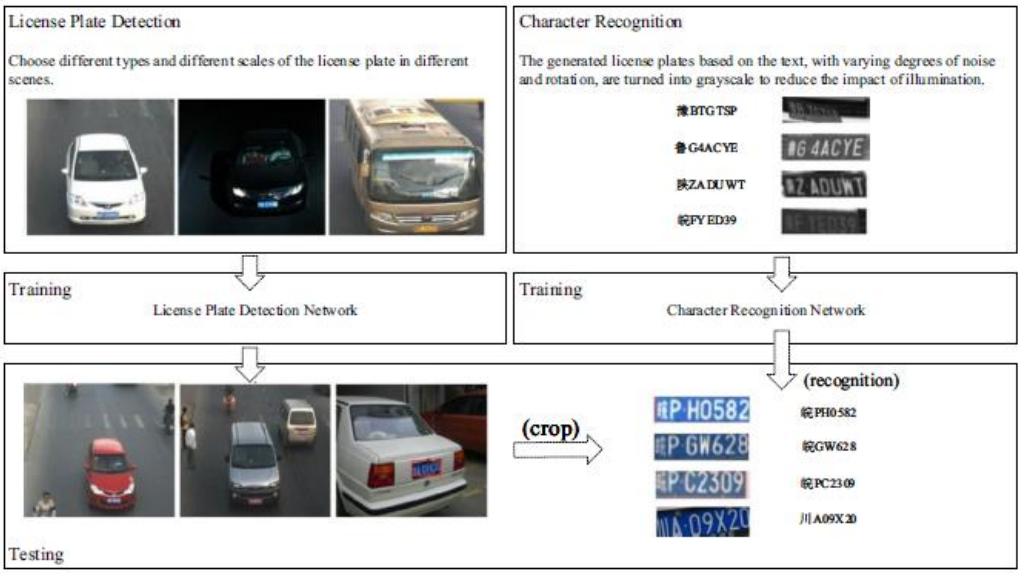

Fig 1 The architecture of our system.

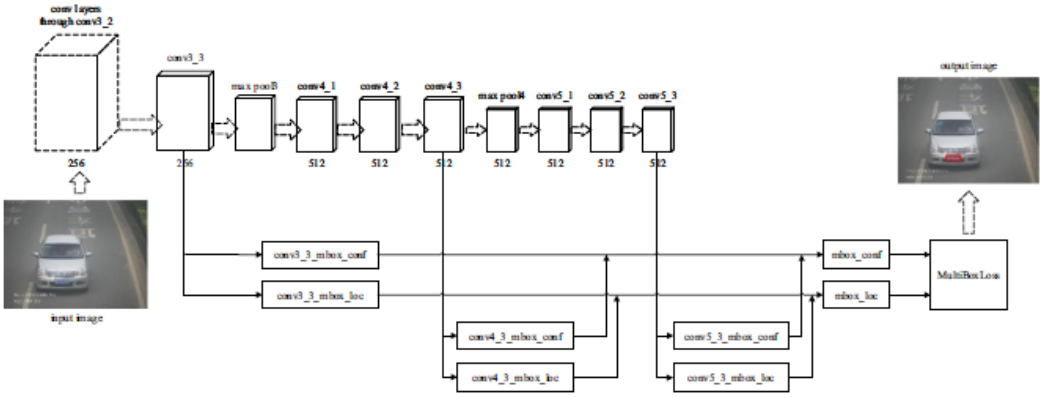

Fig 2 The pipeline of license plate detection

Table1 Different types of license plates in mainland China

No.	Type	Dimension	Description
1	Small vehicle license plate	440 × 140	blue background, white font
2	Large vehicle license plate	440 × 140	yellow background, black font
3	Tractor license plate	440 × 140	yellow background, black font
4	Police car license plate	440 × 140	white background, black font
5	Embassy license plate	440 × 140	black background, white font

Default boxes Various methods have been proposed to generate region proposal because this is the foundation of object detection. Traditional methods include Selective Search and SPP-net are proposed by He et al. [14]. RPNs are used in Faster R-CNN

and are much faster than previous methods. Our system adopts the solution like RPNs and we associate a set of default bounding boxes with each feature map cell in each hierarchical feature layer. Each cell at hierarchical feature layer is relative to 6 default boxes (2 scales and 3 aspect ratios). Different from Faster R-CNN, we use default boxes from different feature layers, which fit for different scales of license plate detection.

Loss function We output two sibling layers for each hierarchical layer. The first sibling layers output each bounding-box regression offsets, while the second sibling layers output each default box confidences. The localization loss is:

$$Loss_l = \frac{1}{N_{reg}} \sum_i L_{reg}(t_i, t_i^*) \tag{1}$$

where Nreg is the number of selected default boxes, Lregis the regression loss of the prediction box, which is calculated by Smooth-L1, ti is the offset of the predicted bounding-box, and ti∗ is the ground-truth box coordinates.

We assign a positive sample while a default boxes has an Intersection-over-Union (IoU) overlap with a ground-truth more than a threshold (e. g, 0.7). We use these samples and adopt the Smooth L1 loss for bounding-box regression, as Fast R-CNN does.

The classification loss is:

$$Loss_c = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*) \tag{2}$$

where Ncls is the number of training samples including negatives and positives, Lcls is the classification loss of a single prediction box, which is calculated by softmax

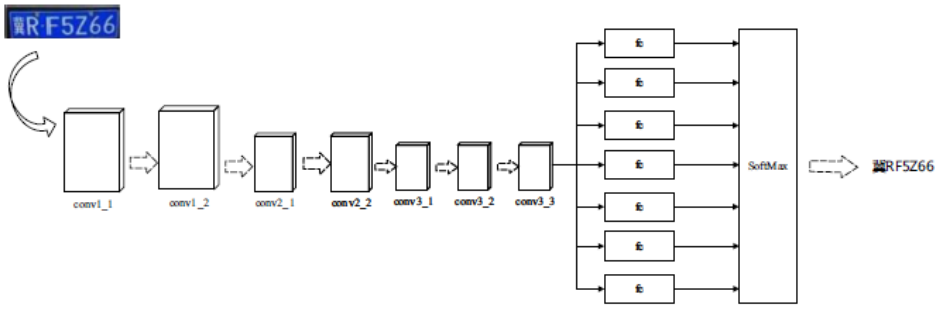

Fig3 The pipeline of character recognition network

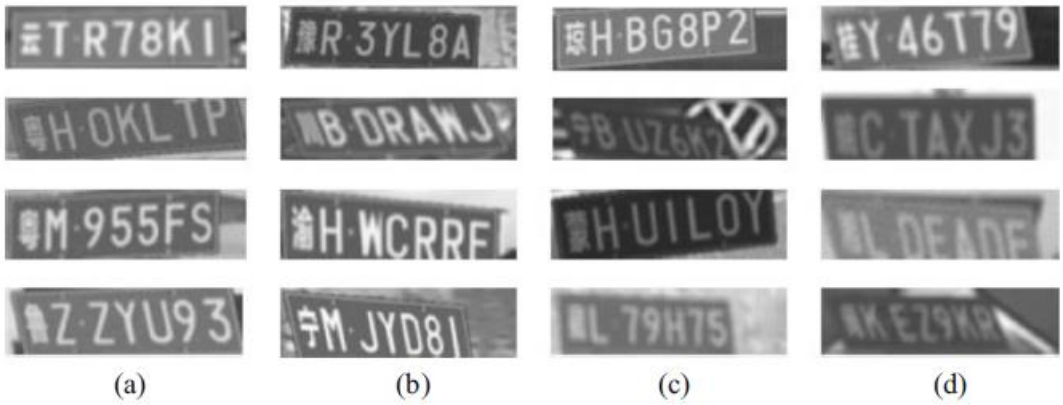

Fig4 Examples of license plates

pi represents the ith default box predicted probability, p*i is the ground-truth label, and p*i if the default box is positive and p*I ¼ 0 if the default box is negative.

Therefore, the combined loss function is:

$$Loss = Loss_l + \alpha Loss_c \tag{3}$$

the two terms are weighted by a balancing parameter α. In our implementation, αas set to 1.

3.2 Character recognition

Table 1 lists a few types of license plates. Many existing methods for license plate recognition are limited by the character segmentation algorithm. Our method follows the idea of object detection using CNNs, which localizes and classifies each character on the license plate. We design the network inspired by the VGG net, but make some changes at the topper layer. Figure 3 shows the architecture of our network. License plate generation algorithm For training the character recognition model, we need to annotate the location and class of each character on a license plate; and several license plates. may be present in a single image. Obviously, this work is heavy and time-consuming. Therefore, we design an algorithm for automatically generating license plates as follows. Two font libraries are established: the first is used to create Chinese characters, while the second is used for alphabets and numbers. When producing a license plate, we extract from the first font library randomly to generate Chinese

Table 2 The scales and aspect ratios of default boxes used in our network

Layer	conv3_3		Conv4_3		Conv5_3	
Scale	8^2	16^2	32^2	64^2	128^2	256^2
Aspect ratio (1:1)	8 x 8	16 x 16	32 x 32	64 x 64	128 x 128	256 x 256
Aspect ratio (1:1)	11 x 6	23 x 11	45 x 23	91 x 45	181 x 90	362 x 181
Aspect ratio (1:1)	15 x 4	9 x 27	19 x 54	37 x 111	74 x 221	148 x 443

characters and extract the remaining six characters from the second character library; then, paste the seven characters on the template we prepared. Different types of license plates can be produced while changing the template, such as the templates of a small car or police car license plates. At last, some processing methods like rotation, affine transformation, and noise adding are manipulated to simulate real scenes. Our algorithm is shown in the following Algorithm 1. Rot () rotates the image along the x axis, rotRandom () flips the image along the y axis, randomEnvironment () adds a background to the generated car, tfactor () adds noise.

Figure 4 shows the generated license plates by using our algorithm.

Algorithm 1 Framework of license plate generation algorithm

Prepared: Two font libraries;
The license plate background template.
Generation: Implementation details

```
1: def  rot (img, angel, shape, MaxAngel); % rotation function
2: def  RandomEnvirment (img, factor, size); % noise function
3: def  tfactor (img); % noise function
…
4: def  GenCh (f, val); % extract characters function
5: def  generatePlate (self, intval, text): % combine functions above randomly
        if (intval > 0&intval ≤50):
            com = rot (com, r1(45) – 30, com. shape, 15);
            com = rotRandrom (com, 10, (com. shape [1], com. shape [0]));
            com = tfactor (com);
            com = randomEnvirment (com, self. Noplatespath);
            com = AddGauss (com, 1 + r (4));
            com = addNoise(com);
            com = cv2. cvtColor (com, cv2. BGR2GRAY);
        if (intval > 50&intval ≤ 60):
            com = tfactor(com);
```

```
                com = randomEnvirment (com, self. Noplatespath);
                com = AddGauss (com, 1 + r (6));
                com = addNoise (com);
                com = cv2. cvtColor (com, cv2. BGR2GRAY);
                …
6: def main (self, batchSize, pos, charRange, outputPath, size):
        img = generatePlate (np. random. randint (0, 99), plateStr);
7: return img;
```

4 Implementation details

In this section, we will give the implementation details for license plate detection. For training the model, we use the VOC2007 and VOC2012 for pre-training and fine-tune the resulting model by adopting SGD (Stochastic Gradient Descent) with an initial momentum of 0.9,weight decay of 5×10−4, and batch size of 64. We first use the learning rate of 10−3 for 7000 iterations and then decay it to 2×10−3 for a subsequent 3000 iterations.

Default boxes scales and aspect ratios To detect license plates in different scales, our system selects default boxes with two scales in each hierarchical feature layer. At each hierarchical layer, particularly scales of license plates are detected. For each scale, we also choose three aspect ratios: 1:1, 1:2, and 1:3. Table 2 shows the different scales and aspect ratios used in each hierarchical feature layer in our work..

Hard negative mining Many methods using CNNs for objective detection only define the negative samples as the default boxes IoU with ground truth boxes lower than 0.5,and the positive samples as the default boxes IoU with ground truth boxes higher than 0.7. However, in practice, almost all default boxes are negatives, and thus, we must find a balance between positive and negative samples. Therefore, we sort the highest loss confidence default boxes and choose the top ones. We keep the ratio between negatives and positives below 3:1.

Data augmentation To improve the detection accuracy of small license plates and make the model robust, we do some work for augmenting training data:

1) Resize the original image into the interval of [0.3, 1.2] randomly;
2) Randomly use Gaussian blur and various noises;
3) Add low levels of rotations.

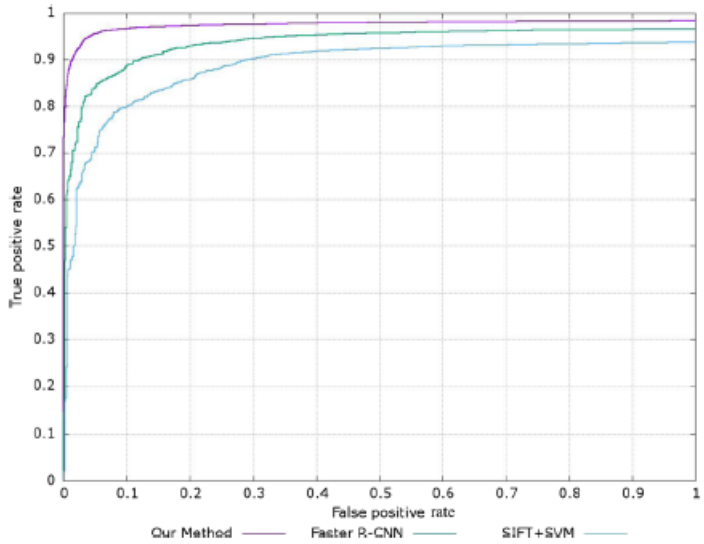

Fig6 Precision-recall curves on ours test data set

4.2 Details of character recognition

In this section, we present the implementation details for character recognition. The structure of recognition network is shown in Fig. 3. We train the model using SGD with a learning rate of 10−3, a momentum of 0.9, weight decay of 10−3, iteration of 600,000, and batch size of 20. According to our algorithm for automatically generating training data, 40,000 license plates are used to training a model, while 2000 training data from real scenes are added to avoid over-fitting. For anchors, we use one scale of 32×64 and one aspect ratio of 1:2. This setting is sufficient for training a model with great performance.

5 Experimental results

5.1 License plate detection experiment

5.1.1 Experiment on openITS

We evaluate the effectiveness of our approach on OpenITS, which is a dataset built by ZhongShan University. OpenITS includes 1403 images with 1403 annotated license plates,

Table 3 Different Settings of Default Boxes

Setting	Aspect ratios	Default boxes scales	mAP
1 ratio,1 scale	1:1	128^2	91.17%
1 ratio,1 scale	1:1	256^2	91.41%
3 ratio,1 scale	{1: 1, 2: 1; 3: 1}	128^2	92.09%
3 ratio,1 scale	{1: 1, 2: 1; 3: 1}	256^2	92.09%
3 ratio,3 scale	{1: 1, 2: 1; 3: 1}	{64^2, 128^2, 256^2}	95.21%

Table 4 Effects of hierarchical feature layers and batch normalization

Prediction source layers form:			mAP	
conv3_3	Conv4_3	Conv5_3	no batch normalization	batch normalization
√	√	√	94.03%	95.21%
√	√	—	92.91%	93.89%
√	—	—	90.27%	91.32%

covering all provinces and municipalities in mainland China. Figure 5 shows the effectiveness

of our method compared to the previous work on OpenITS. The performance is evaluated by

mean average precision (mAP). As we can see in Fig. 5, the mAP nearly approaches to 100% on OpenITS by using our model, which is better than systems by using Faster R-CNN and HOG + SVM.

5.1.2 Experiment on our own test datasets

To evaluate the performance of our method with more complex scenarios with various camera angles, image resolutions, and illumination conditions, we build our test dataset, which includes 962 images. The full test dataset built by us is available at http://t.cn/RaLJTKr. Figure 6 shows that on our complex test dataset, the performance of our model is also better than that of Faster R-CNN and HOG + SVM.

5.1.3 Different settings of default boxes based on the same network structure

We conduct experiments using different scales and aspect ratios on our own test dataset. Our network structure is based on the network from Fig. 2. Table 3 shows the effectiveness of different settings of default boxes on the mAP.

5.1.4 More hierarchical layers based on a similar number of boxes

We reduce the number of hierarchical layers to measure the advantage of our network. We keep the number of default boxes close to 2304 while adjusting the network structure. As shown in Table 4, with fewer hierarchical layers, the mAP drops dramatically from 95.21% to 91.31%, which illustrates that hierarchical feature layers are beneficial for license plates detection in different scales. To measure the effectiveness of the batch normalization, we add this layer after all of the convolutional layers and before the activation function. It improves the convergence of the model

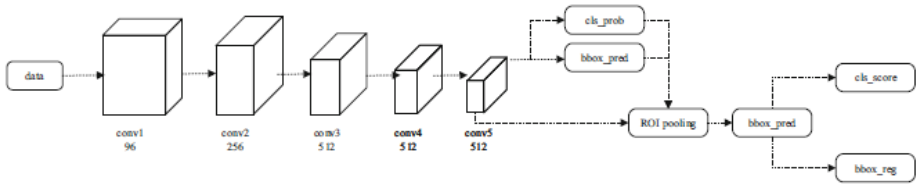

Fig7 The network structure used for the comparative experiment

Table 5 The scale and aspect ratio of the anchor boxes based on the Faster R-CNN

Layer	conv3_3			
Scale	16^2	32^2	64^2	128^2
Aspect ratio (1:1)	16 x 16	32 x 32	64 x 64	128 x 128
Aspect ratio (1:1)	23 x 11	45 x 23	91 x 45	181 x 90
Aspect ratio (1:1)	9 x 27	19 x 54	37 x 111	74 x 221

greatly and helps to regularize the model. Table 4 also shows the effect of adding batch normalization on mAP.

5.2 Comparative experiment with faster r-cnn

We design a comparative experiment to verify whether our method is more advantageous than the method based on Faster R-CNN. The network architecture used in the experiment is modified on the basis of the ZF network, and the training dataset contained 31,696 images.

Figure 7 shows the network structure used for the comparative experiment.

Firstly, we scale the input image to a size of 1000*600 and send it to the convolutional neural network. After passing through convolutional layer, 256 feature maps with size 85*51 are obtained. For the size characteristics of the license plate, we choose Anchor generation strategy different from the method describe in the paper before. For each unit block on the feature map, four different sizes and three different aspect ratio prediction boxes are generated. The size and ratio of the generated prediction frame are shown in Table 5.

5.3 Character recognition experiment

We conduct experiments on our test data set for character recognition. Because 2000 training data from real scenes annotated by artificial means is insufficient for training the CNN model, we generate nearly 40,000 license plates by using the algorithm of automatically generating training data. To increase the complexity of the training data set and prevent over-fitting, we add 2000 training data from the real scenes

to train the CNN model. A traditional method (HOG + SVM) is also trained with the same training data for contrast. Table 6 shows the performance comparison of character recognition.

Some samples of final results under different scales and backgrounds are shown in Fig. 8. The first Chinese character cannot be recognized well at the stage of character recognition due to noise and low resolution.

Table 6 The performance comparison of character recognition

Methods	Accuracy	
—	data all from generation	add data from real scenes
—	(40000)	(40,000 + 2000)
Our CNN model	0.92415	0.96782
HOG + SVM	0.87973	0.89355

(a) (b) (c)

Fig 8 Exemplar results

6 Conclusion

This work introduces a method based on CNN for license plate detection and character recognition. An important characteristic of our work is the use of hierarchical feature layers, which are helpful for detecting license plates in different scales. Besides, our network is fit for arbitrary-sized image as inputs to avoid image distortion. Meanwhile, we design an algorithm to automatically generating license plates, reducing the cost of model training at the stage of character recognition. Our model achieved perfect performance on license plate detection benchmarks (OpenITS), and character recognition also saw high accuracy. However, our system cannot be applied to the case of incomplete license plate recognition, which is an open problem for future work.

References

1. Abolghasemi V, Ahmadyfard A (2009) An edge-based color-aided method for license plate detection. Image Vis Comput 27(8):1134

2. Alexe B, Deselaers T, Ferrari V (2012) Measuring the Objectness of Image Windows. IEEE Trans Pattern Anal Mach Intell 34(11):2189

3. Arbelaez P, Ponttuset J, Barron J, Marques F, Malik J (2014) Multiscale Combinatorial Grouping, in Computer Vision and Pattern Recognition, 328–335

4. Cao G, Chen J, Jiang J (2003) An adaptive approach to vehicle license plate localization, in Industrial Electronics Society, 2003. IECON '03. the Conference of the IEEE, 1786–1791 Vol.2

5. Chen YN, Han CC, Wang CT, Jeng BS (2006) The Application of a Convolution Neural Network on Face and License Plate Detection, in International Conference on Pattern Recognition, 552–555

6. Chen Y, Nguyen TV, Kankanhalli M, Yuan J (2014) Audio Matters in Visual Attention, Circuits and Systems for Video Technology IEEE Transactions on 24(11), 1992

7. Chen Y, Zhang L, Liu X, Chen C (2016) Flickr Circles: Aesthetic Tendency Discovery by Multi-View Regularized Topic Modeling, Information Sciences 372(C), 148

8. Chen Y, Hu Y, Zhang L, Li P, Zhang C (2017) Engineering Deep Representations for Modeling Aesthetic Perception, IEEE Transactions on Cybernetics PP (99), 1

9. Dai J, Li Y, He K, Sun J (2016) R-fcn: Object detection via region-based fully convolutional networks, Advances in neural information processing systems, 379–387

10. Deb K, Jo K (2009) A Vehicle License Plate Detection Method for Intelligent Transportation System Applications 40(8), 689

11. Dlagnekov (2004) License plate detection using adaboost, Computer Science and Engineering Department pp. 42–53

12. Feng BY, Ren M, Zhang XY, Liu CL (2014) Effective license plate detection using fast candidate region selection and covariance feature based filtering, in IEEE

International Conference on Advanced Video and Signal Based Surveillance, 1–60

13. Girshick R, Donahue J, Darrell T, Malik J (2013) Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation, in IEEE Conference on Computer Vision and Pattern Recognition, 580–587

14. He K, Zhang X, Ren S, Sun J (2015) Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition, IEEE Transactions on Pattern Analysis and Machine Intelligence 37(9), 1904

15. Hongliang B, Changping L (2004) A hybrid license plate extraction method based on edge statistics and morphology, in International Conference on Pattern Recognition, 831–834 Vol. 2

16. Huang H, Gu M, Chao H (2008) An EfficientMethod of License Plate Location in Natural-Scene Image, in International Conference on Fuzzy Systems and Knowledge Discovery, Fskd 2008, 18–20 October 2008, Jinan, Shandong, China, Proceedings, Volume, 15–19

17. Hung KM, Hsieh CT (2010) A Real-Time Mobile Vehicle License Plate Detection and Recognition. Tamkang J Sci Eng 13(4):197

18. Liu W, Anguelov D, Erhan D, Szegedy C, Reed S, Fu CY, Berg AC (2016) Ssde: Single shot multibox detector, European conference on computer vision, pp. 21–37

19. Long J, Shelhamer E, Darrell T (2014) Fully convolutional networks for semantic segmentation, IEEE Transactions on Pattern Analysis and Machine Intelligence PP (99), 640

20. Luo L, Sun H, Zhou W, Luo L (2009) An Efficient Method of License Plate Location, in International Conference on Information Science and Engineering, 770–773

21. Maglad KW (2012) A vehicle license plate detection and recognition system. J Comput Sci 8(3):310

22. Parkhi OM, Vedaldi A, Zisserman A (2015) Deep Face Recognition, in British Machine Vision Conference, 41.1–41.12

23. Ren S, He K, Girshick R, Sun J (2015) Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks, IEEE Transactions on Pattern Analysis and

Machine Intelligence PP (99), 1

24. Simonyan K, Zisserman A (2014) Very Deep Convolutional Networks for Large-Scale Image Recognition, Computer Science

25. Uijlings JR, Sande KE, Gevers T, Smeulders AW (2013) Selective Search for Object Recognition. Int J Comput Vis 104(2):154

26. Wazalwar D, Oruklu E, Saniie J (2012) A Design Flow for Robust License Plate Localization and Recognition in Complex Scenes. J Trans Technol 02(1):13

27. Wu HHP, Chen HH, Wu RJ, Shen DF (2006) License Plate Extraction in Low Resolution Video, in International Conference on Pattern Recognition, 824–827

28. Xia Y, Liu Z, Yan Y, Chen Y, Zhang L, Zimmermann R (2017) Media Quality Assessment by Perceptual Gaze-Shift Patterns Mining, IEEE Transactions on Multimedia PP (99), 1

29. H. Zhang, W. Jia, X. He, Q. Wu (2006) Learning-Based License Plate Detection Using Global and Local Features, in International Conference on Pattern Recognition, 1102–1105

30. Zitnick CL, Dollár P (2014) Edge Boxes: Locating Object Proposals from Edges 8693, 391

基于分层卷积神经网络的车牌识别系统

Qiang Lu & Yu Liu & Jing Huang & Xiaohui Yuan & Qingxin Hu

摘要

近几年，有许多基于卷积神经网络的系统在车牌识别领域取得了良好的表现，然而几乎所以基于多层网络的多尺度车牌识别系统在面对多变的场景时表现得都不稳定。同时，在字符识别阶段，数据标注繁重而且费时，给训练更好的模型带来了沉重的负担。为此，我们设计了一种算法，它可以自动生成标注好的训练数据，并且可以从真实场景中自动生成数据。我们的系统在识别车牌时的测试集准确率达到了 99.99%。（OpenITS 数据库）。同时，在字符识别方面也有很高的正确率。这些测试都验证了我们模型的优势。

关键词；车牌定位，字符识别，分层特征层，生成训练数据

1.介绍

在过去几年，得益于事故监控系统的普及和发展，车牌识别技术在世界范围内得到了很高的重视。卷积神经网络的出现改善了许多技术的表现，利于人脸识别技术。同时也为车牌照识别技术提供了新的视野。当然，高效准确的车牌照识别技术依旧是一个挑战。在车牌识别方面，即使是最新的使用卷积神经网络搭建的系统在处理因为相机拍摄角度而导致的车牌扭曲问题是表现得依旧不好。尤其是当车牌在照片中显得很小的时候。为了解决这个问题，我们建立了一个分层机构来处理。它的最底层拥有最小的感受野，用于识别较小的车牌。并且高层就拥有更大的感受野。用于识别更大的车牌。对于每个结构层，都有一组拥有不同长宽比和缩放比例的特定大小的默认框与每一个映射单元对应。通过预测，由网络为每个默认框生成置信度和偏置，用来识别和绑定车牌。根据不同特征层的不同处理策略，系统可以自动识别不同尺寸的车牌。此外，过去设计的 CNN 模型要求输入 CNN 模型的图像尺寸是固定的，这导致了不可避免地图像失真。为了解决这个问题，我们采用了全卷积网络（FCN），它可以接受任意尺寸的输入，实验证明我们的系统对于任意尺寸的车牌输入都有很高的正确率。

关于字符识别，人工为图片添加注释意味着大量的数据量和过长的时间消耗，这会给模型训练时带来很大的困难。为了解决这个问题，我们设计了一种算法用于自动生成注释后的车牌，通过不同等级的噪声和旋转来接近真实场景的训练集。重要的是，我们的算法可以生成任何车牌，例如警车车牌，拖拉机车牌等。而且我们的方法被证明在训练速度和精准度上都有良好的表现。

我们在本研究中中的贡献如下：

1.我们设计了多层特征层的 CNN 模型用来识别多尺寸的车牌。

2.该网络适用于任何大小的输入图片。

3.我们为字符识别设计了一种自动生成标注车牌的算法，减少了模型的训练时间。

2.相关工作

车牌识别技术的方法基于四个类别：边界信息，材质信息，颜色信息和机器学习。基于边界的方法利用了车牌的边界信息来将车牌分离出来，在图像中选出车牌可能在的区域，然后利用梯度下降法和熵去除其中不正确的区域。这种方法非常依赖图中的边界信息，并且在不受限制的环境中表现得不好（通用性差）。基于材质的识别技术是利用“底帽”和形态学梯度等形态学方法在图像中识别出车牌。但是这种方法需要输入高质量的图片而且计算量非常大，所以不能用于实时的计算。基于颜色的方法是利用颜色来尽可能的移除图片中的背景然后利用纹理分析技术来找出车牌的区域。但是，由于颜色信息非常的不稳定，这种方法受光照影响极大。基于机器学习的方法是通过先提取出“Haar-like”特征，然后利用“adaptive boosting”算法训练联级分类器来分离车牌。这个方法用在车牌识别时非常的高效。但是训练后的联级分类器只能用于固定尺寸的照片，并不适用于所有尺寸。支持向量机可以用非线性高维识别模式解决图像尺寸过小的问题。Zhang et al.提出了一种方法，他先是利用图片中的边界信息，确定出最开始的候选区域，然后提取出候选区域的协方差特性，最后训练 SVM 分类器去去除候选区域中错误的区域。此外，这个研究也广泛用到了神经网络的技术。但是，人工特征往往区分度较低，这限制该方法的实用性。

FCN：自 Long et al.改进了 FCN 后，FCN 在语义分割领域获得了巨大成功，他们将先进的分类网络引入了 FCN，并且将反卷积技术用于将特征图上采样还原到原图大小。这有利于将分类网络应用到分割技术上。像素预测技术就是为了上采样层的语义分割设计的。此外，前面的层将与后面的层结合，以发现更多的细节，从而获得更高的精度。我们受到这个技术的启发，用较前的层检测小车牌，用较后的层检测大车牌。此外，FCN 支持 任意大小的车牌输入以应对图像变形。

区域候选区：目前有许多关于区域候选区的研究，例如 MCG，选择性搜索，滑动窗口等。Girshick 将选择性搜索和卷积神经网络结合起来，用来生成区域候选区，并且取得远超过去 VOC2012 方法下得到的平均准确率。然而，选择性搜索目标识别中的独立部分，他会浪费大量的时间。Ren er al.设计了一

种 RPN(区域候选网络)，它使用 DCNN（深度卷积网络）来计算候选区。并且可以端对端的训练生产区域候选区。Liu et al 应用了一种端对端的训练机制，它取得了比 RCNN 更快的速度和更高的平均正确率，于是我们决定采取这种方法来进行车牌识别。

3.模型设计

我们设计的方法结合了两个部分：车牌识别和字符识别。一个卷积神经网络别训练用作从视频中识别出车牌，然后另一个网络用来识别字符。图 1 解释了我们系统的结构。

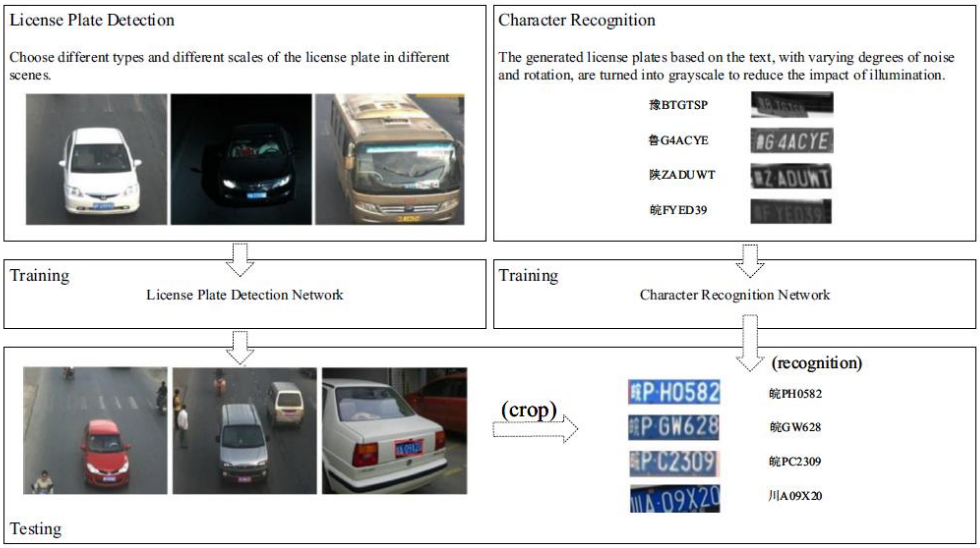

图 1 系统结构

3.1 车牌识别

特征层：我们车牌识别网络的传输图如图二所示，在 conv3_3 前的层是基于 VGG 设计的，这些层用于编码语义信息。我们接下来加入了辅助层 conv3_3,conv4_3,conv5_3，他们负责评估预选车牌区域的可信度。然后我们利用 NMS 整合这三层得到的结果，得到最终的车牌候选区域。Conv3_3 拥有更好的空间细节用以识别小的车牌。同时，conv5_3 拥有更大的感受野，这对识别更大的车牌有很大的帮助，使用这个设计，我们可以更轻松地识别车牌，尤其是小的车牌。接下来，我们加入了很多辅助结构，比如批量归一化。它负责调整模型和改善平均正确率。根据我们的经验，加入归一化可以提高 1%的正确率。同时也能加快收敛速度。

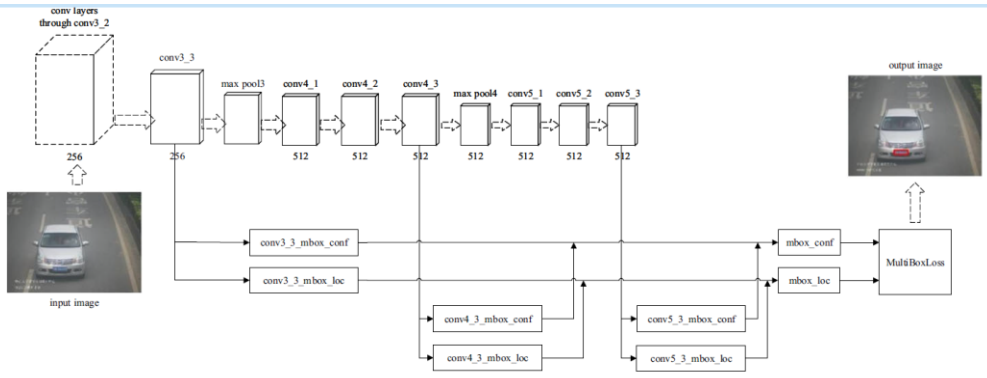

Fig. 2 The pipeline of license plate detection. Here we omit some of the structures, such as the batch normalization behind the convolutional layers

图 2 车牌识别过程

候选框：因为生成区域候选区是目标识别的基础，所以有很多方法被用于实现生成区域候选区。传统的方法包括选择性搜索和 SPP-net。而之后出现地技术 RPN 则要比之前的技术快很多。我们的系统采用的方式类似于 RPN.，并且我们将一组候选框和特征图上的单元格结合起来。与 Faster R-CNN 不同，我们使用的默认框来自不同的特征层，这有助于我们识别不同大小的车牌。

损失函数：我们为每一层输出一对兄弟层，兄弟层的一次层高是一个候选框的回归偏移值，兄弟层的第二层是一个默认框的可信度。损失函数如下：

$$Loss_l = \frac{1}{N_{reg}} \sum_I L_{reg}(t_i, t_i^*) \quad (1)$$

上式中N_{reg}表示选中的默认框的数量，L_{reg}表示预测框的回归损失函数。t_i是预测框的偏移，,t_i^*是真值框的坐标。

我们设计了一个阳性例子，这个例子的检测评价函数和真值框的重叠部分超过了阈值。我们对这个例子采用了 Smooth L1 损失函数进行回归分析。

这个例子的损失函数如下：

$$Loss_c = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*) \quad (2)$$

其中N_{cls}是样本的数量，其中包含了阳性和隐形样本，L_{cls}是单个预测框的分类损失函数，它由分类器计算。p_i表示第 I 个默认框的预测可信度。,p_i^*表示基础真值如果默认框是阴性的则,p_i^*=0.

因此，总的损失函数如下：

$$Loss = Loss_l + \alpha Loss_c$$

该式中的第二个损失函数被乘上了一个权重值 α 在我们的例子中，α=1.。

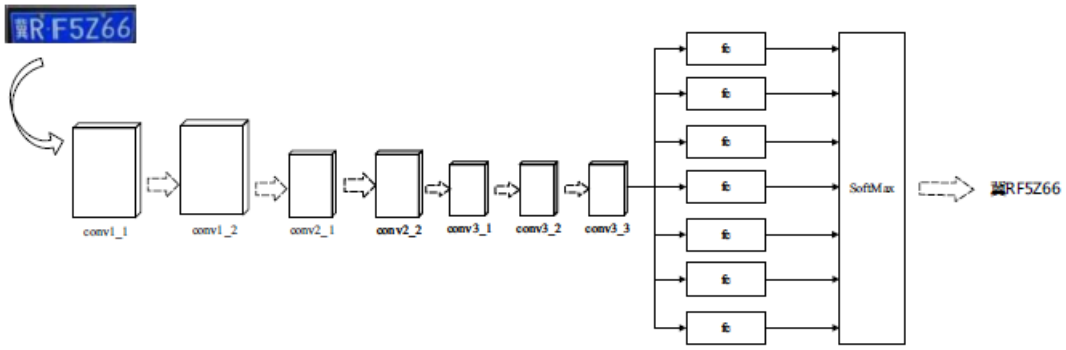

图 3 字符识别网络结构

3.2 字符识别

表 1 列出了常见的几种车牌，现有的车牌识别方法多受制于字符分割算法。我们的方法基于 CNN 进行目标识别,通过算法定位和分类车牌上的每一个字符。我的设计的网络起源于 VGG，但对它的顶层做出了很多调整。

表 3 展示了我们网络的架构。

车牌生成算法：为了训练字符识别模型，我们需要标注车牌上每个字符的位置和类别。并且多个车牌可能出现在同一张图片中。显然，这项工作相当繁重和费时。因此我们生成了一种算法，自动生成这些车牌。

我们一共建立了连个字体库：第一个字体库用于存储汉字字符。第二个字体库用用于储存字母和数字。当我们要生成车牌时，我们从第一个库中随机取出一个

表 1 不同类型的车牌

No.	Type	Dimension	Description
1	小车牌	440 × 140	蓝底白字
2	大车牌	440 × 140	黄底黑字
3	拖拉机车牌	440 × 140	黄底黑字
4	警车车牌	440 × 140	白底黑字
5	使馆车牌	440 × 140	黑底白字

汉字，然后从第二个库中随机取出六个字符，然后将这七个字符贴到我们提前准备好的模板上。如果想得到不同的车牌可以更换模板。比如更换成小车或者警车的模板。最后运用如旋转，仿射变换，加入噪声等方式来模拟真实的情况。我们的算法如算法一所示，Rot()将照片绕 X 轴旋转一定角度，rotRandom（）函数将图片绕 y 轴旋转一定角度，randomenvironment()函数为图片添加背景，tfactor()函数为图片加入噪音。

算法 1：车牌生成算法

```
Prepared: Two font libraries;
          The license plate background template.
Generation: Implementation details
1: def  rot (img, angel, shape, MaxAngel); % rotation function
2: def  RandomEnvirment (img, factor, size); % noise function
3: def  tfactor (img); % noise function
…
4: def  GenCh (f, val); % extract characters function
5: def  generatePlate (self, intval, text): % combine functions above
randomly
       if (intval > 0&intval ≤50):
           com = rot (com, r1(45) - 30, com. shape, 15);
           com = rotRandrom (com, 10, (com. shape [1], com. shape
[0]));
           com = tfactor (com);
           com = randomEnvirment (com, self. Noplatespath);
           com = AddGauss (com, 1 + r (4));
           com = addNoise(com);
           com = cv2. cvtColor (com, cv2. BGR2GRAY);
       if (intval > 50&intval ≤ 60):
           com = tfactor(com);
           com = randomEnvirment (com, self. Noplatespath);
           com = AddGauss (com, 1 + r (6));
           com = addNoise (com);
           com = cv2. cvtColor (com, cv2. BGR2GRAY);
          …
6: def main (self, batchSize, pos, charRange, outputPath, size):
     img = generatePlate (np. random. randint (0, 99), plateStr);
7: return img;
```

图 4 展示了该算法生成的一些车牌

(a) (b) (c) (d)

图 4 由算法生成的车牌

4.实现细节

4.1.车牌定位详解

这个部分将会详细讲解车牌识别的实现细节为了训练模型，我们使用了 VOC2007 和 VOC2012 做预训练，然后用 SGD 验证结果。我们将初动量设置为 0.9，权值衰减设置为5×10^{-4},批次大小为 64.我们第一次设置学习步长为10^{-3}，并且迭代 7000 次，然后以2×10^{-3}的步长迭代 3000 次。

默认框的缩放和长宽比：为了检测不同大小的车牌，我们的系统选择了不同特征层上的两个缩放。在每一层上，都用来检测特定尺寸的车牌。对于每一个缩放，我们也选择了三个不同的长宽比 1：1，1：2，1：3.表 2 展示了每一层上不同的大小和长宽比。

表 2 网络使用候选框的大小和长宽比

名称	conv3_3		Conv4_3		Conv5_3	
大小	8^2	16^2	32^2	64^2	128^2	256^2
长宽比（1:1）	8 x 8	16 x 16	32 x 32	64 x 64	128 x 128	256 x 256
长宽比（1:1）	11 x 6	23 x 11	45 x 23	91 x 45	181 x 90	362 x 181
长宽比（1:1）	15 x 4	9 x 27	19 x 54	37 x 111	74 x 221	148 x 443

难分样本挖掘：许多运用 CNN 技术的目标识别技术只将阴性样本定义成它的默认框的 IOU 与真值框的 IOU 详细度小于 0.5，将阳性样本定义为其默认框的 IOU 与基础真值框的 IOU 相似度大于 0.7，但在实际运用中，这么做的话几乎所有样本都是阴性样本，因此，必须找到阴性样本和阳性样本间的正确平衡点。所以我们将所有样本按照可行度排序，然后将阴性样本和阳性样本的比例控制在低于 3：1.。

数据增强：为了增加模型对小车牌的识别准确率，并且增加模型的健壮性。我们对训练数据做了一些增强。

4.2 字符识别的细节

在这个部分，我们将展示字符识别的实现细节，图 3 展示的是字符识别网络的架构。我们由 SGD 训练模型，其中步长设置为10^{-3}，动量设置为 0.9，权值损失设置为10^{-3}迭代次数设置为 600000 次，批次大小为 20.训练数据是由我们设计的算法生成的 40000 个车牌，同时，为了防止过拟合发生，我们在训练数据中加入了 2000 个真实的数据。这些数据足够悬链出一个良好的模型。

5.实验结果

5. 1. 车牌定位实验结果

5.1.1 在 OPENITS 库上测试车牌定位

OpenITS 是一个由中山大学建立的数据库，我们在它上面测试了我们算法的有效性。OpenITS 库含有 1403 张含有车牌的图片，包含了中国大陆的所有省市和自治区，图五展示了我们算法和过去算法的平均准确率对比。可以在图中看到我们的模型正确率接近 100%，结果上要优秀与使用 Faster R-CNN 和 HOG+SVM。

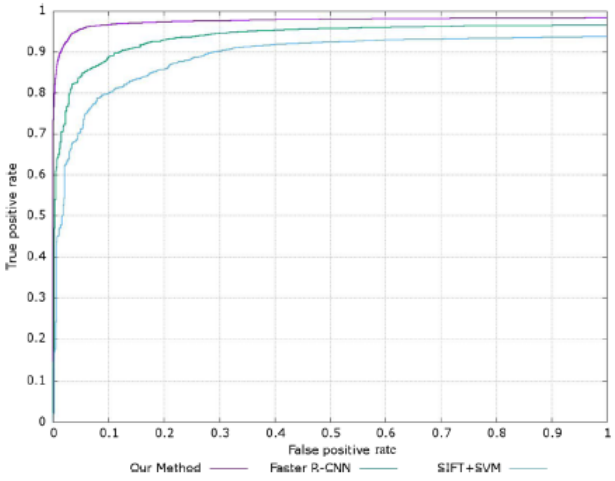

图 6 测试集上的精准度曲线

5.1.2 在我们自己的数据库上的测试结果

为了再更多样的情境下测试我们模型的表现，我们建议了一个自己的数据库，它包含 962 张图片，图六显示了再我们这个更复杂的数据库上，我们模型的表现要好于使用 Faster R-CNN 和 HOG+SVM。

5.1.3 再同样的网络构架下不同的默认框设置

我们在不同的缩放和长宽比下测试了我们的模型，图 2 是我们的网络结构，表三展示了在不同的默认框设置下我们模型的平均准确率。

表 3 不同设置的候选框

设置	长宽比	候选框大小	平均准确度
1 个长宽比，1 个大小	1:1	128^2	91.17%
1 个长宽比，1 个大小	1:1	256^2	91.41%
3 个长宽比，1 个大小	{1: 1, 2: 1; 3: 1}	128^2	92.09%
3 个长宽比，1 个大小	{1: 1, 2: 1; 3: 1}	256^2	92.09%
3 个长宽比，3 个大小	{1: 1, 2: 1; 3: 1}	$\{64^2, 128^2, 256^2\}$	95.21%

5.1.4 在默认框设置不变下，层数对模型的影响

我们减少了模型的层数以测量我们模型的优势，我们保持默认框的数量接近 2304 的同时调整模型网络。结果如表四所示，在减少层数后，平均准确率从 95.21% 减少到了 91.31%，这证明了特征层的个数对模型的影响大于不同缩放比对模型的影响。为了验证批归一化的有效性，我们在最后一个卷积层和激励函数之间插入这层，结果证明它有助于模型的收敛和调整。表四展示了批归一化对平均准确率的影响。

表 4 分层特征与批处理的影响

卷积层			平均准确率	
conv3_3	Conv4_3	Conv5_3	无批处理	有批处理
√	√	√	94.03%	95.21%
√	√	—	92.91%	93.89%
√	—	—	90.27%	91.32%

5.2 和 Fast R-cnn 的对比试验

我们设计了一个对比实验，以验证我们的方法是否比基于 Faster R-CNN 的

方法更具优势。 实验中使用的网络架构是在 ZF 网络的基础上进行修改的，训练数据集中包含 31,696 张图像。

图 7 显示了用于比较实验的网络结构。

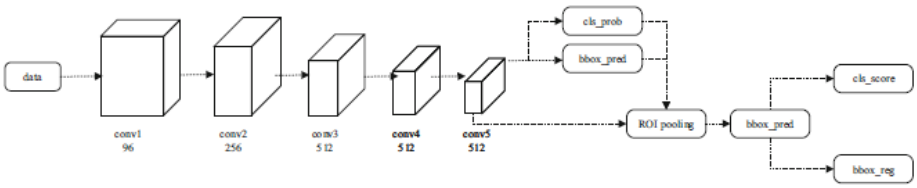

图 5 对比试验网络结构

首先，我们将输入图像缩放到 1000 * 600 的大小，并将其发送到卷积神经网络。通过卷积层后，获得 256 个尺寸为 85 * 51 的特征图。 对于车牌的尺寸特征，我们选择不同于先前论文中描述的方法的生成策略。对于特征图上的每个单元块，将生成四个不同的大小和三个不同的纵横比预测框。 表 5 中显示了 Fast R-cnn 网络候选框的长宽比和大小。

表 5 Fast R-cnn 网络候选框的大小和长宽比

名称	conv3_3			
大小	16^2	32^2	64^2	128^2
长宽比（1:1）	16 x 16	32 x 32	64 x 64	128 x 128
长宽比（1:1）	23 x 11	45 x 23	91 x 45	181 x 90
长宽比（1:1）	9 x 27	19 x 54	37 x 111	74 x 221

5.3 字符识别实验

我们在测试数据集上进行字符识别实验。由于 2000 个真实场景的训练数据不足以训练 CNN 模型，因此我们使用自动生成训练数据的算法生成了近 40,000 个车牌。 为了增加训练数据集的复杂性并防止过度拟合，我们从真实场景中添加了 2000 个训练数据来训练 CNN 模型。 还使用相同的训练数据对传统方法（HOG + SVM）进行训练以进行对比。 表 6 显示了字符识别的性能比较。

图 8 显示了在不同比例和背景下的一些最终结果样本。由于噪声和分辨率低，第一个汉字在字符识别阶段不能被很好地识别。

表 6 不同训练方法的性能对比

训练方法	准确率	
—	算法生成的数据	加入真实图片的数据
—	(40000)	(40,000 + 2000)
Our CNN model	0.92415	0.96782
HOG + SVM	0.87973	0.89355

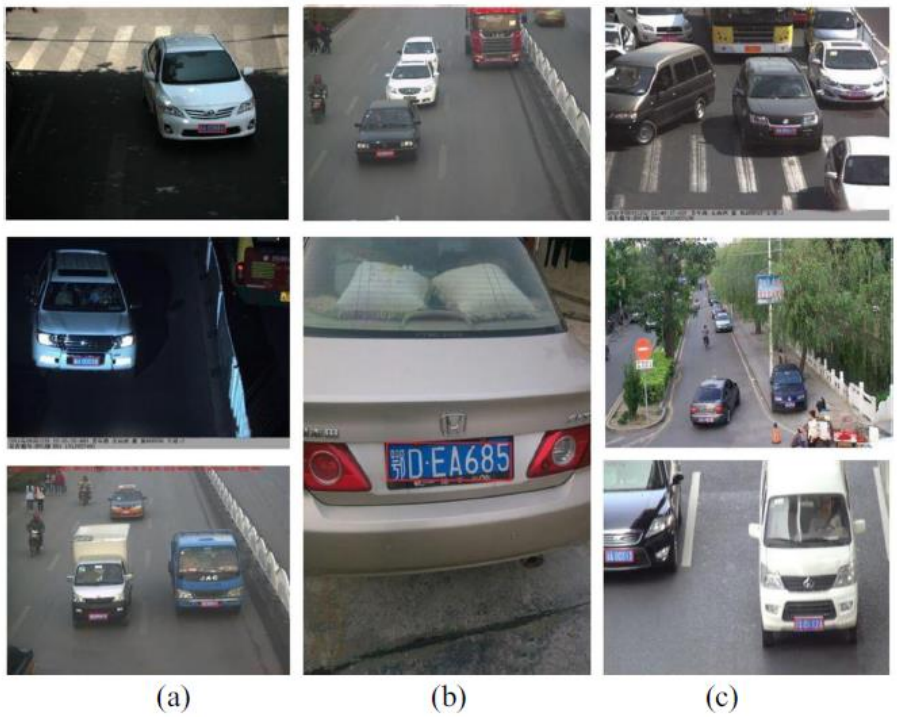

图 8 结果展示

6 总结

本课题介绍了一种基于 CNN 的车牌检测和字符识别方法。 我们课题的一个重要特点是使用分层特征图层，这有助于检测不同比例的车牌。 此外，我们的网络适合于任意大小的图像作为输入，以避免图像失真。 同时，我们设计了一种算法来自动生成车牌，从而降低了字符识别阶段模型训练的成本。 我们的模型在车牌检测基准（OpenITS）上实现了完美的性能，并且字符识别也具有很高的准确性。 但是，我们的系统无法应用于车牌识别不完全的情况，这是以后工作的一个未解决问题。

参考文献

1. Abolghasemi V, Ahmadyfard A (2009) An edge-based color-aided method for license plate detection. Image Vis Comput 27(8):1134

2. Alexe B, Deselaers T, Ferrari V (2012) Measuring the Objectness of Image Windows. IEEE Trans Pattern Anal Mach Intell 34(11):2189

3. Arbelaez P, Ponttuset J, Barron J, Marques F, Malik J (2014) Multiscale Combinatorial Grouping, in Computer Vision and Pattern Recognition, 328–335

4. Cao G, Chen J, Jiang J (2003) An adaptive approach to vehicle license plate localization, in Industrial Electronics Society, 2003. IECON '03. the Conference of the IEEE, 1786–1791 Vol.2

5. Chen YN, Han CC, Wang CT, Jeng BS (2006) The Application of a Convolution Neural Network on Face and License Plate Detection, in International Conference on Pattern Recognition, 552–555

6. Chen Y, Nguyen TV, Kankanhalli M, Yuan J (2014) Audio Matters in Visual Attention, Circuits and Systems for Video Technology IEEE Transactions on 24(11), 1992

7. Chen Y, Zhang L, Liu X, Chen C (2016) Flickr Circles: Aesthetic Tendency Discovery by Multi-View Regularized Topic Modeling, Information Sciences 372(C), 148

8. Chen Y, Hu Y, Zhang L, Li P, Zhang C (2017) Engineering Deep Representations for Modeling Aesthetic Perception, IEEE Transactions on Cybernetics PP (99), 1

9. Dai J, Li Y, He K, Sun J (2016) R-fcn: Object detection via region-based fully convolutional networks, Advances in neural information processing systems, 379–387

10. Deb K, Jo K (2009) A Vehicle License Plate Detection Method for Intelligent Transportation System Applications 40(8), 689

11. Dlagnekov (2004) License plate detection using adaboost, Computer Science and Engineering Department pp. 42–53

12. Feng BY, Ren M, Zhang XY, Liu CL (2014) Effective license plate detection using fast candidate region selection and covariance feature based filtering, in IEEE

International Conference on Advanced Video and Signal Based Surveillance, 1–60

13. Girshick R, Donahue J, Darrell T, Malik J (2013) Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation, in IEEE Conference on Computer Vision and Pattern Recognition, 580–587

14. He K, Zhang X, Ren S, Sun J (2015) Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition, IEEE Transactions on Pattern Analysis and Machine Intelligence 37(9), 1904

15. Hongliang B, Changping L (2004) A hybrid license plate extraction method based on edge statistics and morphology, in International Conference on Pattern Recognition, 831–834 Vol. 2

16. Huang H, Gu M, Chao H (2008) An EfficientMethod of License Plate Location in Natural-Scene Image, in International Conference on Fuzzy Systems and Knowledge Discovery, Fskd 2008, 18–20 October 2008, Jinan, Shandong, China, Proceedings, Volume, 15–19

17. Hung KM, Hsieh CT (2010) A Real-Time Mobile Vehicle License Plate Detection and Recognition. Tamkang J Sci Eng 13(4):197

18. Liu W, Anguelov D, Erhan D, Szegedy C, Reed S, Fu CY, Berg AC (2016) Ssde: Single shot multibox detector, European conference on computer vision, pp. 21–37

19. Long J, Shelhamer E, Darrell T (2014) Fully convolutional networks for semantic segmentation, IEEE Transactions on Pattern Analysis and Machine Intelligence PP (99), 640

20. Luo L, Sun H, Zhou W, Luo L (2009) An Efficient Method of License Plate Location, in International Conference on Information Science and Engineering, 770–773

21. Maglad KW (2012) A vehicle license plate detection and recognition system. J Comput Sci 8(3):310

22. Parkhi OM, Vedaldi A, Zisserman A (2015) Deep Face Recognition, in British Machine Vision Conference, 41.1–41.12

23. Ren S, He K, Girshick R, Sun J (2015) Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks, IEEE Transactions on Pattern Analysis and

Machine Intelligence PP (99), 1

24. Simonyan K, Zisserman A (2014) Very Deep Convolutional Networks for Large-Scale Image Recognition, Computer Science

25. Uijlings JR, Sande KE, Gevers T, Smeulders AW (2013) Selective Search for Object Recognition. Int J Comput Vis 104(2):154

26. Wazalwar D, Oruklu E, Saniie J (2012) A Design Flow for Robust License Plate Localization and Recognition in Complex Scenes. J Trans Technol 02(1):13

27. Wu HHP, Chen HH, Wu RJ, Shen DF (2006) License Plate Extraction in Low Resolution Video, in International Conference on Pattern Recognition, 824–827

28. Xia Y, Liu Z, Yan Y, Chen Y, Zhang L, Zimmermann R (2017) Media Quality Assessment by Perceptual Gaze-Shift Patterns Mining, IEEE Transactions on Multimedia PP (99), 1

29. H. Zhang, W. Jia, X. He, Q. Wu (2006) Learning-Based License Plate Detection Using Global and Local Features, in International Conference on Pattern Recognition, 1102–1105

30. Zitnick CL, Dollár P (2014) Edge Boxes: Locating Object Proposals from Edges 8693, 391