

班 级 031213  
学 号 03121300

西安电子科技大学

# 本科毕业设计论文



题 目 机动车车牌的实时监测

与识别系统

学 院 计算机学院

专 业 计算机科学与技术

学生姓名 王昌旭

导师姓名 李辉



# 毕业设计（论文）诚信声明书

本人声明：本人所提交的毕业论文《机动车车牌的实时检测与识别系统》是本人在指导教师指导下独立研究、写作成果，论文中所引用他人的无论以何种方式发布的文字、研究成果，均在论文中加以说明；有关教师、同学和其他人员对文本的写作、修订提出过并为我在论文中加以采纳的意见、建议，均已在我的致谢辞中加以说明并深致谢意。

本文和资料若有不实之处，本人承担一切相关责任。

论文作者：\_\_\_\_\_ (签字) 时间：2016 年 6 月 3 日  
指导教师已阅：\_\_\_\_\_ (签字) 时间：2016 年 6 月 3 日



## 摘要

随着近几年中国汽车保有量的持续增加，智能交通系统以及自动驾驶技术俨然已成为十分热门的研究领域。而机动车车牌识别技术作为这两项技术的基础和关键，有着十分重要的研究意义和应用价值。本文主要对车牌识别系统所涉及的车牌检测、车牌定位、车牌字符分割和车牌识别四个子系统展开研究，并运用最新的深度学习技术实现一个简单的车牌识别系统。全文工作将分为以下几点：

第一，本文首先阐述本课题的研究背景及意义，并简要介绍深度学习技术相较于传统技术的异同；

第二，本文尝试使用 Faster R-CNN 技术来进行车牌检测，来克服传统方法严重依赖手工特征和手工规则、鲁棒性差等缺点；

第三，本文尝试使用 CNN 回归车牌顶点坐标的方法以实现车牌定位；

第四，车牌字符分割问题可以看做对车牌图片中的文字进行提取的问题。本文尝试使用 Class-specific Extremal Region 的方法进行车牌字符分割；

第五，在图像识别领域，目前最理想的方法当属卷积神经网络（简称 CNN），它有着更好的识别准确率和更强的鲁棒性。因此本系统采用 CNN 的方法进行车牌识别。

最后，本文总结了深度学习技术及其在车牌识别这一计算机视觉任务上的应用前景，并提出对未来的展望。

**关键词：**车牌检测 车牌定位 车牌字符分割 车牌识别 深度学习

---

## 摘要

---

## Abstract

With the continued increase in vehicle ownership in China in recent years, intelligent transportation system and autopilot technology has became a very popular research field. As the foundation and key of these two technologies, the vehicle license plate recognition technology is of great importance in both research and application. This paper focuses on four essential subsystem in the vehicle license plate recognition system including detection, localization, segmentation and recognition. We implement a simple vehicle license plate recognition systems based on the up to date deep learning technologies. We summarize the whole work as follows:

First, we illustrates the certain necessity of the research and implementation of this vehicle license plate recognition system by describing the background and significance of the research.

Second, This paper will present an approach of vehicle license plate detection based on Faster R-CNN technology, in order to overcome the shortcomings of traditional methods such as the dependency of hand-made features as well as the poor robustness.

Third, in this paper, we try to localize vehicle license plate by regressing the vertex positions using CNN.

Fourth, The vehicle license plate segmentation problem can be viewed as the problem of detecting characters from license plate pictures. In this paper, we try to segment license into character by using Class-specific Extremal Region technology.

Fifth, in the field of image recognition, there is no doubt that CNN is the state-of-art approach currently, it has a higher accuracy and a better robustness. So we adopt the CNN for license plate recognition in our final system.

Sixth, through the study of the technologies above, this paper implements a simple license plate recognition system. In our recognition system, we use RCNN for detection, CNN regression for location, CSER for character segmentation and

## Abstract

---

CNN for the final recognition, respectively.

**Key words:** vehicle license plate detection    vehicle license plate location    vehicle license character segmentation    vehicle license plate recognition    deep learning

# 目录

<b>第一章 引言 .....</b>	<b>1</b>
1.1 研究背景和研究意义 .....	1
1.1.1 研究背景 .....	1
1.1.2 研究意义 .....	1
1.2 深度学习与计算机视觉 .....	3
1.2.1 传统方法 .....	3
1.2.2 深度学习 .....	4
1.3 本文的内容安排 .....	5
1.4 本章小结 .....	7
<b>第二章 车牌精确定位 .....</b>	<b>9</b>
2.1 卷积神经网络概述 .....	9
2.1.1 卷积层 .....	9
2.1.2 池化层 .....	10
2.1.3 回归问题与 $SmoothL_1$ 损失函数 .....	11
2.1.4 反向传播算法 .....	11
2.2 Residual Network .....	12
2.3 使用卷积神经网络进行车牌精确定位 .....	13
2.3.1 数据处理 .....	13
2.3.2 模型设计 .....	13
2.3.3 算法实现 .....	14
2.4 本章小结 .....	14
<b>第三章 车牌检测 .....</b>	<b>17</b>
3.1 R-CNN 概述 .....	17
3.1.1 R-CNN.....	17
3.1.2 Fast R-CNN .....	19

3.1.3 Faster R-CNN.....	21
3.2 使用 Faster R-CNN 进行车牌检测.....	24
3.2.1 数据集准备 .....	24
3.2.2 算法实现 .....	24
3.3 本章小结 .....	25
<b>第四章 车牌字符分割 .....</b>	<b>29</b>
4.1 Extremal Region 综述 .....	29
4.1.1 Extremal Region .....	29
4.1.2 Class-specific Extremal Region.....	30
4.2 使用 Class-specific Extremal Region 进行车牌字符分割 .....	32
4.2.1 算法设计 .....	32
4.2.2 算法实现 .....	33
4.3 本章小结 .....	34
<b>第五章 车牌字符识别 .....</b>	<b>37</b>
5.1 使用 CNN 进行多类别分类 .....	37
5.1.1 Softmax 激活函数 .....	37
5.1.2 交叉熵损失函数 .....	38
5.2 使用 CNN 进行车牌字符识别 .....	38
5.2.1 模型设计 .....	38
5.2.2 算法实现 .....	39
5.3 本章小结 .....	40
<b>第六章 实验结果 .....</b>	<b>41</b>
<b>第七章 总结与展望 .....</b>	<b>43</b>
7.1 总结 .....	43
7.2 展望 .....	44
<b>致谢 .....</b>	<b>45</b>
<b>参考文献 .....</b>	<b>47</b>
<b>附录 A 实验环境 .....</b>	<b>49</b>

# 第一章 引言

## 1.1 研究背景和研究意义

### 1.1.1 研究背景

自从十九世纪八十年代人类世界的第一辆汽车诞生以来，人类的生活就再也离不开汽车这种方便的交通工具。尤其是进入 21 世纪以来，中国大陆地区汽车保有量快速上升，几乎到了家家都有车的地步，一些富裕一些的家庭甚至一人一辆车。据不完全统计，在 2002 年左右我国的汽车保有量在 2000 万辆左右，而到了 2015 年则增加到 1.72 亿辆之多，并且以每年百分之十的速度持续增长。如此大量的汽车，在给我们带来诸多便利的同时，也增加了城市道路的拥堵和交通事故的发生率。如今，道路拥堵、违规停车、交通事故已成为每个城市管理者的心头病。因此，各国政府也在一直不停的研究如何对车辆有效监管，提高道路利用率，避免交通事故。在此前提下，诞生了两大热门研究方向：智能交通系统 (Intelligent Transportation System, ITS) 和自动驾驶系统。这两项技术都旨在通过计算机技术来解决交通管理等问题。而这两项技术中必不可少的一环便是进行车牌识别，即通过计算机视觉技术，对图像（或视频）中的车牌进行准确的定位、分割和识别。

### 1.1.2 研究意义

从研究背景中我们可以知道，VLPR 技术在实际生活中有着十分重要且广泛的应用。下面我们举出几个实际生活中自动车牌识别系统的典型应用：

#### 停车场的智能管理系统

一般而言，在住宅小区、大型商场、车站等地方都会建设配套的停车场。而如何对进出停车场的车辆进行管理，则是一个十分重要的问题。目前常见的有三种方式：

- 纯人力管理

- 发放出入门禁卡
- 智能的电子化管理

其中前两种方式都有着管理成本高、需要人力维护等缺点，而通过使用智能化电子管理系统，不仅有效降低了管理、维护成本，而且可以实现无人值守式停车场。因此可以看到目前使用智能化电子管理的停车场数量在不断增加。一般的智能化电子管理系统是在停车场的出入口架设摄像头，自动对进出的车辆进行拍照，并对车辆的车牌号进行识别，从而对车辆进行登记管理。



图 1.1 校门口的停车场门禁系统

## 停车场自动找车系统

随着停车场越建越大，相信在若大的停车场里找寻自己的车辆是每个车主都头疼过得问题。目前市面上出现了一项新的自动找车系统，即通过在停车场内部署的大量摄像头，对停在不同车位的车辆进行车牌识别。车主在找车的时候，只需在终端机上输入自己车辆的车牌号，系统就会自动找到车辆位置，并为车主提供导航。这项技术也大大方便了人们的生活、提高了停车场的用户体验。

## 电子警察

目前的电子警察系统，通过大量安装道路路口、路段上的摄像头，对车道内的机动车驾驶行为进行不间断自动检测和记录，并检测车辆的违法驾驶行为。而这项系统也必然少不了车牌识别系统的帮助。可以看到，目前通过对城市道路电子警察的大量部署，可以有效对城市道路上的车辆进行 24 小时的监控，很大程度上降低了交通事故的发生率。

## 行车记录系统

随车道路上车辆的不断增加，交通事故总是不可避免。为了方便在发生事故时进行责任认定，许多车主都在汽车上加装了行车记录仪以记录行车情况。而如果一个行车记录仪能够自动进行车牌识别并记录行车过程中周围的车辆信息，则无疑会大大方便车主和交警进行事故责任认定等。

## 自动驾驶系统

基于人工智能技术的自动驾驶系统被认为是未来交通系统的希望之星。众所周知，人类驾驶员在驾驶过程中难免存在疲劳、注意力分散、视角盲区等问题，从而造成了大量的交通事故。而自动驾驶系统则无这些缺陷，因此可以在很多情况下降低交通事故的发生概率。因此许多科技公司和研究机构都投入了大量的人力物力以进行自动驾驶技术的研究。事实上德国已经开始尝试使用自动驾驶的卡车进行高速公路货运，以减少人力成本和交通事故。因此，车牌识别系统作为自动驾驶系统里必不可少的一个子环节，有着十分重要的研究意义。

综上所述，基于计算机视觉技术的车牌识别方法在诸多领域都有着极为广泛的应用前景和市场价值，能够对社会和人类带来极为深远的影响。

## 1.2 深度学习与计算机视觉

### 1.2.1 传统方法

本文中提到的传统方法，是指在机器学习技术引入计算机视觉之前，人们为了解决常见的视觉问题发明的大量基于手工特征和手工规则的方法。比较经典的，比如用于直线、曲线检测的广义 Hough 变换，用于边缘检测的 Sobel 算子以及



图 1.2 Google 公司研制的无人驾驶汽车

SIFT、SURF、HoG 等图像特征。这些方法在早期的计算机视觉任务中被大量的使用，并取得了一定的效果。比如，许多车牌检测系统使用 Hough 变换对车牌的边缘进行检测从而实现车牌边缘的定位；使用二值化方法结合直方图投影进行车牌字符分割。

由于传统的计算机视觉方法严重依赖于图像特征的选择，因此这些方法不仅需要花费大量的精力进行特征工程，而且难以胜任一些复杂的计算机视觉任务，比如自然场景中的物体检测、字符识别等任务。同时，即使是在一些已取得成功应用的领域，这些方法的效果也与人类视觉系统的能力相距甚远。于是很自然的，人们就想如果手工设定的规则和特征不能很好的适用于各种复杂计算机视觉任务，那为什么不让计算机自己从数据中学习特征和规则呢？于是机器学习技术开始被应用于计算机视觉任务中，其中在近几年的研究中又以深度学习技术效果最好、应用最广。

### 1.2.2 深度学习

所谓深度学习技术，实际上就是神经网络技术，只不过相较于传统的多层次感知机等浅层神经网络模型，深度学习中使用的网络模型层数更多（即更深）、规模更大。目前主流的深度神经网络模型可以分为卷积神经网络 CNN 和递归神经网路 RNN 两类。其中 CNN 被广泛应用于各类图像相关的计算机视觉任务中，而 RNN 更多的应用在自然语言处理、时间序列预测等时序相关的任务中。本文所采

用的方法都是基于 CNN 的，因此在此着重介绍 CNN 的发展和应用。1998 年前后，Yann LeCun 教授第一次成功将 CNN 技术应用于手写数字识别任务<sup>[1]</sup>，并取得了与人类水平相当的准确率。这一成果被直接应用于许多支票识别系统系统中。但随后由于人工智能的第二次低潮，CNN 技术的研究一直进展缓慢。不过随着摩尔定律导致的计算能力提升和互联网带来的海量数据，在 2012 年前后，Hinton 等人使用深度神经网络技术在 ImageNet 比赛中以准确率高出第二名百分之二十的惊艳成绩夺得第一名<sup>[2]</sup>。值得注意的是，2012 年 ImageNet 比赛中第二名所使用的方法正是基于复杂特征工程和统计机器学习的“传统方法”，因此这次事件揭示出在面对计算机视觉任务时相较于传统方法的巨大潜力与优势。值得一提的是，自 2012 年以后，ImageNet 每年的第一名都是深度学习方法，比如 2015 年的第一名是由 Kaiming He 等人发明的 Residual Network<sup>[3]</sup>。

相较于传统方法，CNN 最大的特点在于，它可以自动从数据集中学习图像特征，而不需要依赖复杂的特征工程。实践证明，通过 CNN 学习出来的特征，在许多计算机复杂视觉任务中要优于通过特征工程设计的手工特征。同时，许多研究也表明，人类的视觉系统有着与 CNN 相似的结构，这似乎也解释了为何在复杂计算机视觉任务中 CNN 可以取得和人类视觉系统相当的水平。目前，除了字符识别、图像识别等任务外，深度学习技术也被成功应用于物体检测、图像语义分割、图像标注等任务中。

但另一方面，由于关于神经网络人们还未能建立一个精确地数学模型来描述其特性和行为，神经网络作为一个难以解释的“黑箱”一直饱受统计学派机器学习研究者的诟病。不过相信随着研究的深入，这些问题在未来都可以得到解决。

### 1.3 本文的内容安排

一般的车牌识别系统结构如图1.3:

显然，车牌检测、车牌精确定位、车牌字符分割和车牌字符识别则是整个系统的核心和难点，本文也着重对这四个子问题进行分析研究。

本文的内容安排如下：

第一章，为引言内容。介绍了课题研究的背景和意义。

第二章，为车牌精确定位方法研究。这一章首先介绍了 CNN 的基本概念和原理，然后阐述如何使用 CNN 进行关键点回归以进行车牌的精确定位

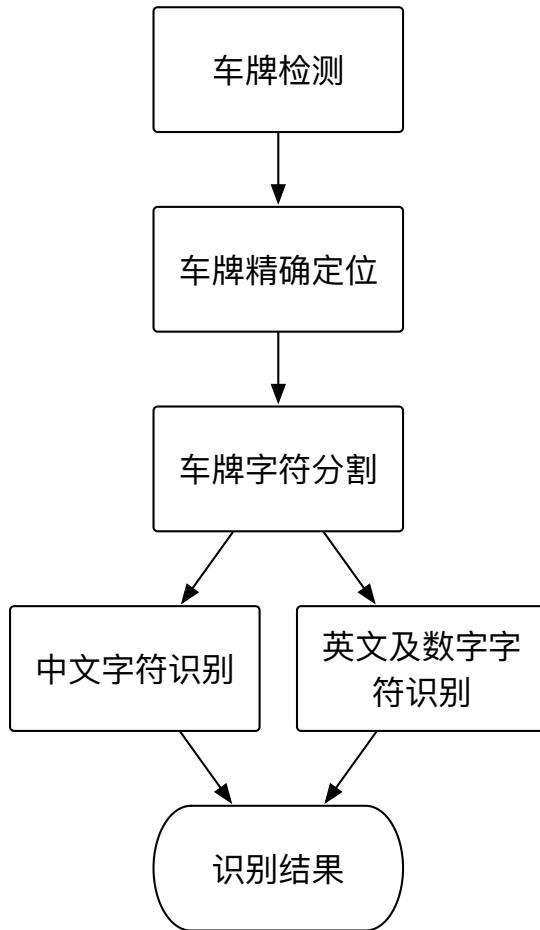


图 1.3 车牌识别系统结构图

第三章，为车牌的检测方法研究。这一章首先介绍了 Region CNN (R-CNN) 系列技术，包括 R-CNN、Fast R-CNN 和 Faster R-CNN。然后通过 Faster R-CNN 技术实现了自然场景中车牌的检测。

第四章，为车牌字符分割方法方法研究。本章首先介绍了几种传统的车牌分割方法，并分析了他们的优缺点。然后本章提出一种新的基于 Extremal Region 的方法进行车牌字符分割。

第五章，为车牌识别方法的研究。本章阐述如何通过卷积神经网络 (CNN) 技术进行车牌字符识别，并进行相关实验。

第六章，为实验效果展示。

第七章，为总结和展望。总结了本文的主要工作，并对未来的研究做了简单的介绍。

## 1.4 本章小结

本章从车牌识别的研究背景和研究意义展开讨论，说明了本研究的意义和价值；然后简要介绍了两种研究思路——即传统方法和深度学习技术；最后简要说明了本文的内容安排。



## 第二章 车牌精确定位

在许多未使用深度学习技术的车牌识别系统中，倾向于使用边缘检测、Hough 变换等寻找车牌的边缘与顶点，直接实现车牌的检测与精确定位，但这些传统方法有着对图像质量、背景颜色敏感，抗噪声能力差，难以处理特殊情况等诸多缺点，在很大程度上限制了其应用。因此在本文中我们将车牌的定位分为车牌检测与车牌精确定位两步进行。其中车牌检测用于找到图像中包含整个车牌的小区域，然后车牌精确定位则从这个小区域中找寻车牌的四个顶点以进行精确定位。关于车牌检测技术将在本文第三章进行介绍，本章将介绍如何使用 CNN 技术进行车牌精确定位。在本章内容中，我们将车牌精确定位问题视为一个回归问题，然后讨论如何使用 CNN 对关键点进行回归的以实现精确定位，并进行实验验证。

### 2.1 卷积神经网络概述

卷积神经网络是一种前馈神经网络，有生物学家们在早期对猫视觉皮层的研究发展而来。一个卷积神经网络一般由许多个卷积层和顶端的全连接层（对应经典的多层感知机）组成，同时也包括相应的权重和池化层（Pooling Layer）。这一结构是的卷积神经网络能够有效利用输入数据的二维结构，同时还极大程度减少了网络参数，降低了训练难度和过拟合的风险。因此卷积神经网络在图像检测、识别方面取得了非常有优异的结构。如 2015 年微软亚洲研究院 Kaiming He 等人发明的 Residual Network 成功的将 ImageNet Top-5 错误率降低到了 5% 左右<sup>[3][4]</sup>。

#### 2.1.1 卷积层

所谓卷积层，是指使用一个相对较小的卷积核（Kernel）在整幅图像上滑动并进行卷积操作并通过非线性激活函数从而得到输出的一种特殊的神经网络结构。常见的卷积核大小有  $1 \times 1, 3 \times 3, 5 \times 5, 7 \times 7$  等。可以看到，卷积层通过权值共享和稀疏连接两大特性，使之参数要远远少于具有相同输入输出大小的全连接层，参数的减少不仅极大程度降低了网络的训练难度，而且有效避免了过拟合的出现。

一般一个卷积层会有多个卷积核，每一个卷积核可以提取图像的不同特征，从而得到不同的特征映射。图 2.1 是 LeNet<sup>[1]</sup> 的网络结构，该网络由深度学习四巨头之一的 Yann LeCun 教授提出，并成功应用于手写数字识别问题<sup>[1]</sup>。

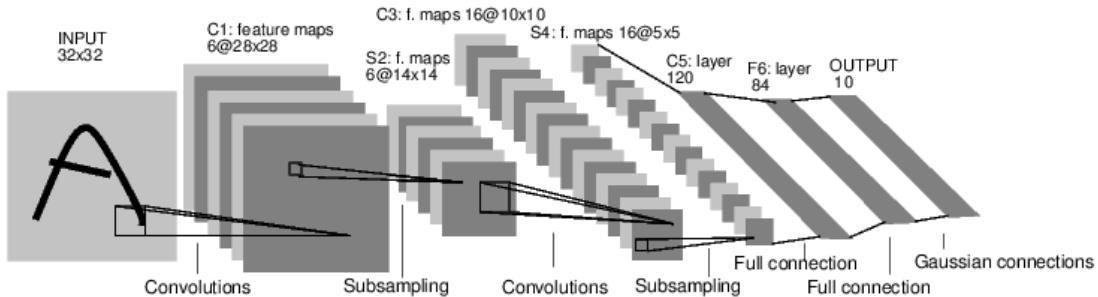


图 2.1 用于手写数字识别的 LeNet5<sup>[1]</sup>

### 2.1.2 池化层

另一个关于卷积神经网络的重要概念是最大池化层，它是一种非线性降采样方法。

在实际应用中，经过卷积层所提取的特征映射仍可能维数较大，例如：对于一个  $32 \times 32$  像素的图像，假设我们的卷积层包含 512 个卷积核，同时使用边缘填充技术（Padding）使卷积输出与输入尺寸一样，则我们将一共得到  $512 \times 32 \times 32 = 524288$  个特征输出，这无论对于网络的储存、计算还是对训练数据集的需求都是十分巨大的，而且增加了过拟合的风险。

因此我们在通过卷积层获取特征映射后，经常使用池化层对卷及特征进行降维。具体来讲，我们卷积后得到的特征映射划分为  $n \times n$  个子区域（子区域间可能重叠也可能不重叠，取决于对池化层参数的定义），然后我们对每个子区域取其最大值（或平均值）所输出，最后得到一个尺寸为  $n \times n$  的特征映射，从而完成了对卷积特征映射的降维操作。

池化技术在计算机视觉中的应用价值有两个方面：

- 它减少了网络特征个数，从而降低了网络复杂度，使训练更容易进行而且不容易出现过拟合
- 这些池化单元使得 CNN 具有了一定的平移不变性，即图像有小的位移时仍能保证提取的特征保持不变

### 2.1.3 回归问题与 $SmoothL_1$ 损失函数

一般回归问题经常使用最小均方误差作为损失函数，但在许多回归问题上，人们倾向使用  $SmoothL_1$  损失函数：

$$smooth_{L_1}(x) = \begin{cases} 0.5x^2 & if |x| < 1 \\ |x| - 0.5 & otherwise \end{cases} \quad (2-1)$$

### 2.1.4 反向传播算法

和其他机器学习算法一样，我们需要找到一种算法能够对神经网络进行训练。事实上，神经网络的训练可以看作是一个最优化问题：

$$\min_{\mathbf{w}} Loss \quad (2-2)$$

其中， $\mathbf{w}$  神经网络模型参数，即每层神经元的权重， $Loss$  代表损失值，用于度量神经网络输出与真实值之间的误差大小。显然，这个优化问题是一个高维空间中的非凸优化问题，不存在显式的解析解。一种常见做法是基于梯度的优化方法，即求取损失函数对参数的梯度，并根据梯度对网络参数进行优化：

$$\mathbf{w} = \mathbf{w} - \alpha \nabla_{\mathbf{w}} Loss \quad (2-3)$$

其中  $\alpha$  控制参数更新的程度，被称为学习率。于是现在我们的问题转换成了如何求取损失关于网络参数的梯度  $\nabla_{\mathbf{w}} Loss$ 。事实上，在多层感知机模型出现之初，当时的神经网络研究者被这个问题困扰了 10 年之久，直到 1986 年前后 Hinton 等人重新发明反向传播算法，从而解决了这个问题。下面我们将简要介绍反向传播算法的原理。

为了叙述方便，我们定义如下记号： $\mathbf{a}^l$  代表第  $l$  层网络的输出， $\mathbf{w}^l$  代表第  $l$  层网络的权重参数， $\sigma$  代表激活函数， $\mathbf{z}^l$  满足  $\mathbf{a}^l = \sigma(\mathbf{z}^l)$ ， $\delta^l = \nabla_{\mathbf{z}^l} Loss$  损失函数关于  $\mathbf{z}^l$  的梯度。注意，上述粗体记号均为向量。根据上述定义和链式求导法则，我

们可以得到以下方程组：

$$\begin{aligned}\delta^L &= \nabla_{\mathbf{a}^L} Loss \odot \sigma'(\mathbf{z}^L) \\ \mathbf{delta}^l &= ((\mathbf{w}^{l+1} \delta^{l+1})) \odot \sigma'(\mathbf{z}^l) \\ \nabla_{\mathbf{w}^l} Loss &= \delta^l \times \mathbf{a}^{l-1}\end{aligned}\tag{2-4}$$

其中， $\odot$  表示向量汉密尔顿积， $\times$  表示向量叉积（内积）。注意到在此方程组中，损失函数关于  $\mathbf{z}^l$  的导数  $\delta$  沿着网络反向传播，逐层向前传递，即网络输出的误差在网络中逐层向前传递并据此修正网络参数  $\mathbf{w}$ ，因此本算法被称作反向传播算法。

## 2.2 Residual Network

按照我们一般的经验，在不出现梯度耗散/梯度爆炸以及过拟合的情况下，CNN 应该是越深效果越好。可是在实验中我们发现，当网络加深后，整个神经网络的错误率反而上升了，我们称这种情况为 Degradation。目前梯度耗散/梯度爆炸问题已经通过引入 ReLU 激活函数而被很好的解决<sup>[5]</sup>，而随着大数据的出现过拟合问题也变得不是那么明显。这时候限制神经网络走向更深的问题就只剩 Degradation 了。而微软亚洲研究院的 Kaiming He 等人于 2015 年提出的 Residual Network 模型则很好的解决了 Degradation 问题，将网络层数加深到了 152 层之多（最新的 Residual Network 已经达到 1200 层）。

Residual Network 的思想是，既然我们使用梯度下降法使一个多层次神经网络去逼近一个复杂的非线性函数  $H(x)$  会遇到 Degradation 问题而难以训练，那我们何不让这个多层次网络去逼近该映射测残差函数  $F(x) = H(x) - x$ ，我们猜想拟合残差函数会比拟合原函数容易一些。虽然我们无法从数学上证明该结论，但是实验表明这种方法确实可行、有效。

图2.2 所示的结构为构成被称之为 Residual Block，其中的 weight layer 一般是多个卷积层及配套的池化层。一个 Residual Network 由多个 Residual Block 堆叠而成。

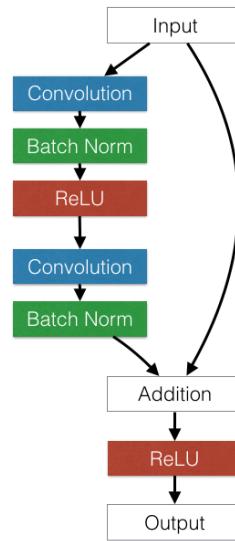


图 2.2 用于 ImageNet 识别的 Residual Network 所使用的 Residual Block

## 2.3 使用卷积神经网络进行车牌精确定位

### 2.3.1 数据处理

在本文的实现中，我们首先将待处理的图片统一缩放至  $448 \times 224$  的分辨率，并进行零化和标准化操作，图像色彩空间选择为 RGB 色彩空间。此外，由于卷积神经网路不具有仿射不变性，因此我们需要对训练集进行数据扩张（Data Augmentation），具体来讲就是对图片进行随机的旋转和放缩，从而增加 CNN 抵抗仿射变换的能力。

### 2.3.2 模型设计

车牌的精确定位问题可以看作一个对车牌四个顶点坐标进行回归的回归问题，因此我们可以很容易的通过卷积神经网络搭建这样一个回归模型，其输入是经过缩放的车牌区域图片，输出则是车牌四个顶点的坐标。本文并未从头训练一个全新的模型完成此任务，而是使用一个训练好的模型，保留其卷积部分并将以 Softmax 为激活函数的分类层更换为无激活函数的回归层。具体来讲本文中使用的是训练好的 34 层 Residual Network<sup>[3]</sup> 模型，其中前 34 个 Residual Block 为卷积特征提取，最后的全连接层实现顶点坐标的回归。

### 2.3.3 算法实现

本章的神经网络模型使用 Torch 框架进行实现，实验机器配置见附录。

训练数据集大小为 900 张，测试数据集为 100 张，训练使用参数见表 2.1。

表 2.1 车牌精确定位网络参数

batch 尺寸	25
学习率	0.1
GPU 数量	2
迭代次数	1000
损失函数	<i>SmoothL<sub>1</sub></i> 损失函数
优化算法	Adam <sup>[6]</sup>

训练及测试过程损失曲线如图2.3，效果如图2.4。

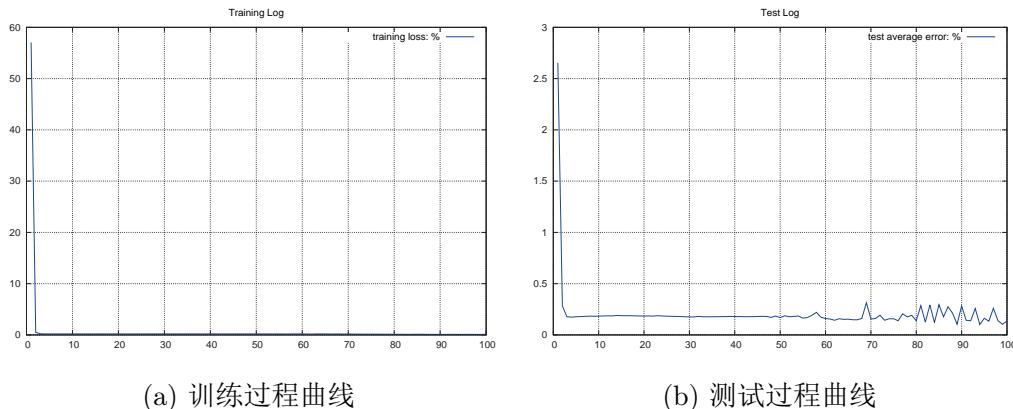


图 2.3 定位网络训练及测试损失

可以看出，使用 CNN 进行车牌精确定位可以较为有效的进行车牌定位，而且不需要手工选择特征，鲁棒性强。但另一方面，由于神经网络的训练由于不需要先验知识，导致其对数据质量和数量要求很高，而能否获取大量带标注的车牌数据进行训练，将直接影响这一步的结果。相信如果能获得更大的带标注训练数据集，本算法的效果还有进一步提升的空间。

## 2.4 本章小结

本章主要简要介绍了 CNN 的基本概念，然后基于 34 层的 Residual Network 设计了一个用于实现车牌精确定位的卷积神经网络回归模型，并通过实验验证了其效果，证明了本方法的可行性。最后提出通过增加训练数据集的大小，可以进一步提高模型的精度和效果。

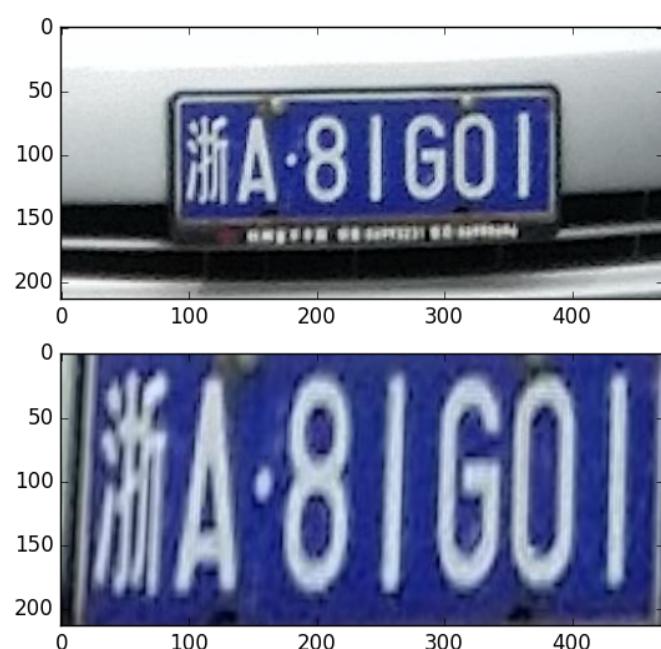


图 2.4 车牌定位效果图



## 第三章 车牌检测

所谓车牌检测，即是从一幅图像中找寻包含车牌的子图像，显然这属于目标检测（Object Detection）问题。目标检测技术作为一个十分热门的研究领域，经历了从十年前 HoG 方法大行其道到如今 R-CNN 技术一统天下的发展过程。R-CNN 技术于 2014 年由 Ross Girshick 等人提出，随后又出现了 Fast R-CNN 和 Faster R-CNN 两代的改进。如今，Faster R-CNN 技术不仅在 Pascal VOC 和 COCO 两个数据集上的目标检测任务中取得了最佳的效果，而且在借助 GPU 加速的情况下做到实时检测（0.2s/帧）。因此本章将首先简要介绍 R-CNN 系列技术，然后在我们的数据集上对其效果进行试验验证。

### 3.1 R-CNN 概述

#### 3.1.1 R-CNN

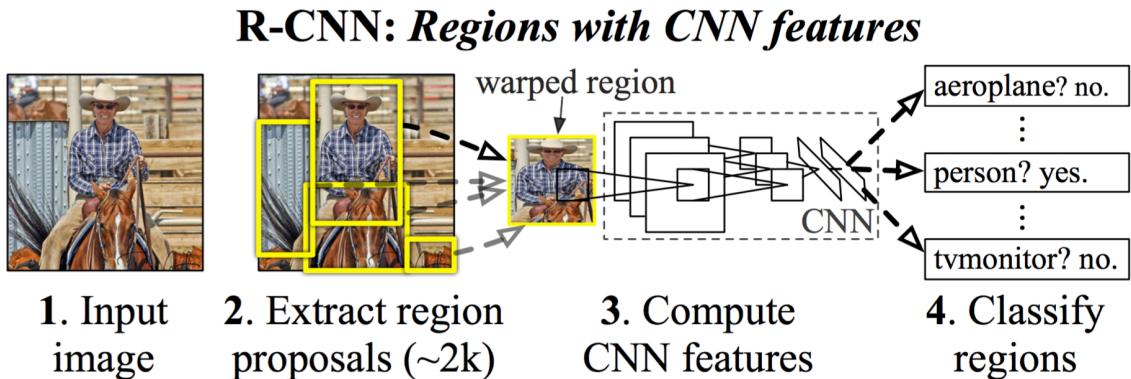
传统的目标检测算法中，要对各种目标实现检测、定位和识别，一般分为三步：

- 扫描图像产生可能的候选区域
- 对每个候选区域提取其特征
- 使用提取的特征对物体进行识别（背景也算为一类）

在 R-CNN 技术（图 3.1）中同样分为这三步，不同的是对每个 proposal 使用训练好的 CNN 进行特征提取以取代 HoG 等传统手工特征。下面我们将分别对这三个步骤咋 R-CNN 模型中的实现进行阐述。

#### 区域选择

常用的候选区域选择算法有滑动窗口法、objectness、类别无关目标选择、Selective Search 等方法。出于对性能和效果的考虑，R-CNN 模型中选择使用 Selective

图 3.1 R-CNN<sup>[7]</sup>

Search 进行目标区域的选择。

## 特征提取

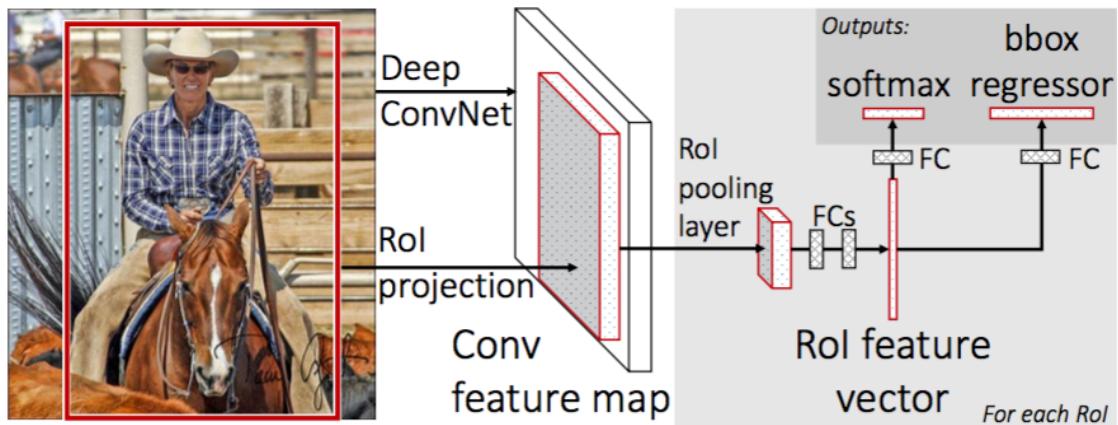
在 R-CNN 模型中，我们使用一个在 ImageNet 数据集上训练好的 VGG 模型用作特征提取。我们摘掉 VGG 模型最顶层的 Softmax 分类层，将倒数第二层的 4096 维向量作为图像特征用于下一步的目标识别。为了使每个候选区域符合 VGG 网络的输入尺寸（ $227 \times 227$  像素 RGB 图像），我们将不同尺寸的目标候选区域统一缩放至  $227 \times 227$  像素进行处理。

## 物体识别

在模型的最后一步，我们使用 SVM 对上一步得到的特征向量进行分类，以得到物体的类别。注意到我们的类别包含  $N$  个物体类和 1 个背景类一共  $N + 1$  类。

R-CNN 模型使用 CNN 提取特征的特性，使得它在 Pascal VOC 数据集上的测试效果要好于所有传统方法，成功将平均精度提高到了 53.7%<sup>[7]</sup>，但同时我们也注意到，模型中有两个重大缺陷：

- 将不同尺寸的候选区域强制缩放至  $227 \times 227$  会损失物体长宽比等许多信息
- 对每个候选区域分别使用 VGG 网络提取特征使得整个识别过程速度十分缓慢，平均每幅图像需要 40s 之久<sup>[7]</sup>。

图 3.2 Fast R-CNN<sup>[8]</sup>

### 3.1.2 Fast R-CNN

上一节末我们提到了原始的 R-CNN 模型有两大缺陷严重限制了其应用场景与范围。为了解决这两大不足，Girshick 在 R-CNN 的基础上进行了改进，提出了 Fast R-CNN 模型<sup>[8]</sup>，如图 3.2。相对于 R-CNN，Fast R-CNN 的改进有两点：

- 引入 RoI 池化层
- 使用神经网络代替 SVM 进行目标的检测与定位

在原始的 R-CNN 模型中，我们需要对每个候选区域进行尺寸缩放，并使用一个训练好的 CNN 进行特征提取，这一步使得 R-CNN 模型变得十分缓慢。因此我们自然的会考虑，我们是否可以只对整幅图进行一次卷积，然后在提取到的卷积特征上直接提取每个候选区域的特征，避免冗余的卷积计算和对候选区域尺寸的缩放。当然，答案是肯定的。

在 Fast R-CNN 模型中，我们首先对整幅图像经过一系列卷积层和池化层产生一个卷积特征映射。然后我们使用 RoI 池化层从得到的特征映射图上提取每个候选区域的特征。

由于不同候选区域的尺寸是不同的，但是为了目标检测和识别，我们需要从特征映射上提取出一个固定长度的向量作为区域特征，因此我们必须实现一种可以根据候选区域尺寸改变窗口大小的特殊池化层，这种池化层被称为 RoI 池化层。和普通池化层固定池化窗口的行为不同，RoI 池化层的池化窗口大小随候选区域的大小而改变，而池化后的特征映射尺寸则固定。

具体来讲，我们定义一个 RoI 为一个四元组  $(r, c, h, w)$ ，其中  $(r, c)$  为 RoI 区域左上角坐标， $(h, w)$  为其长和宽。RoI 池化层将一个尺寸为  $h \times w$  的 RoI 区域分割成  $H \times W$  个子窗口网格，每个窗口尺寸大约为  $\frac{h}{H} \times \frac{w}{W}$ ，然后分别对每个网格窗口中的特征进行池化操作，最后得到的尺寸为  $H \times W$  的特征映射为 RoI 池化层的输出。同时注意到 RoI 池化层在反向传播时，梯度公式与普通池化层是不同的，RoI 池化层的反向传播公式如下：

$$\frac{\partial L}{\partial x_i} = \sum_i \sum_j [i = i^*(r, j)] \frac{\partial L}{\partial y_{rj}} \quad (3-1)$$

其中  $y_{rj} = x_{i^*(r, j)}$ ,  $i^*(r, j) = \arg \max_{i'(r, j)} x_{i'}$ ,  $R(r, j)$  为产生输出单元  $y_{rj}$  的池化窗口中包含的元素坐标集合。

在通过 RoI 层为每个候选区域提取了  $H \times W$  维特征后，这些特征被送入一个多层次感知机中，与 R-CNN 中使用 SVM 进行物体识别不同，这个多层次感知机有两个输出：

- 目标的类别
- 目标的边界矩形 bbox(bounding box)

由于使用了神经神经网路代替了 SVM 进行分类和 bbox 回归，使得整个 Fast R-CNN 可以作为一个完整的神经网络在 GPU 中执行，从而避免了显存和主存间的数据复制，极大程度提高了计算性能。

由于整个网络有两个输出：目标的类别和 bbox，因此整个网络的损失函数不能再是简单的交叉熵损失或者最小二乘损失。在此，作者针对 Fast R-CNN 提出了一种特殊的多任务损失函数。我们定义第一个输出为目标在  $K + 1$  个类别（背景也算一类）上的概率分布  $p = (p_0, \dots, p_K)$ ，第二层的输出为目标的 bbox 坐标尺寸—— $t_k = (t_x^k, t_y^k, t_w^k, t_h^k)$ ，其中  $k$  代表该 bbox 为第  $k$  个物体类别。此外，对每个用于训练的 RoI，我们定义真实类别为  $u$ ，真实 bbox 坐标为  $v$ 。我们定义每个 RoI 的多任务损失函数  $L$  如下：

$$L(p, u, t^u, v) = L_{cls}(p, u) + \lambda[u \geq 1]L_{loc}(t^u, v) \quad (3-2)$$

其中， $[u \geq 1]$  为指示函数——当  $u$  大于等于 1 时为 1，否则为 0。目标分类

的损失函数  $L_{cls}(p, u) = -\log p_u$  为对真实类别  $u$  的对数损失函数。bbox 回归任务的损失函数  $L_{loc}(t^u, v)$  定义如下：

$$L_{loc}(t^u, v) = \sum_{i \in \{x, y, w, h\}} smooth_{L_1}(t_i^u - v_i) \quad (3-3)$$

在有些时候我们也会用最小二乘损失代替  $Smooth_{L_1}$  损失函数。

有了如上的结构定义，我们就可以通过梯度下降法对整个 Fast R-CNN 进行训练。文献<sup>[8]</sup> 中的实验结果表明，Fast R-CNN 在 Pascal VOC 数据集上 mAP 可以达到 70.0%，一举超越 R-CNN 和 SPPnet 而夺得头筹。但是另一方面，虽然 Fast R-CNN 的速度较 R-CNN 有了很大的提升，但仍旧未能达到实时级别，还需进一步改进。

### 3.1.3 Faster R-CNN

虽然 Fast R-CNN 算法在目标识别问题上已能达到准实时的速度，然而这阻挡不了科学家追求更快、更好的脚步。因此微软亚洲研究院的 Shaoqing Ren 等人在 Fast R-CNN 的基础上提出了速度更快的 Faster R-CNN 算法<sup>[9]</sup>（值得一提的是，也正是该组于 2015 年提出了 Residual Network，笔者对他们的科研实力只能佩服不已）。分析发现，使用 Selective Search 的方法进行目标候选区域选择是制约 Fast R-CNN 性能的瓶颈，因此一个很自然的想法便是能否使用基于神经网络的方法进行候选区域的选择以替代速度缓慢的 Selective Search 方法。Ren 等人正是基于这一思想，创造性的提出 Region Proposal Network (RPN)，将 R-CNN 的整个流程归纳进神经网络框架内，进一步提高了检测速度，在 GPU 加速的帮助下达到了实时级别的性能（图 3.3）。

RPN 网络以 Fast R-CNN 网络中经由卷积层提取的卷积特征映射作为输入，并以此产生一系列矩形候选区域，并对每个区域为一个目标的可能性打分。为了生成候选区域，RPN 使用一个小型卷积神经网路在特征映射图上滑动，这个小型网络输入为特征映射图上一个尺寸为  $n \times n$  窗口，然后该窗口内的特征将通过 RoI 池化的方式被映射为一个 256 维的向量（有时也用 512 维，取决于计算特征映射所用卷积网络的复杂度）。最后，这 256 维特征将被送入两个独立的全连接层中——一个全连接层用于目标分类（记为分类层），另一个用于目标位置回归（记为

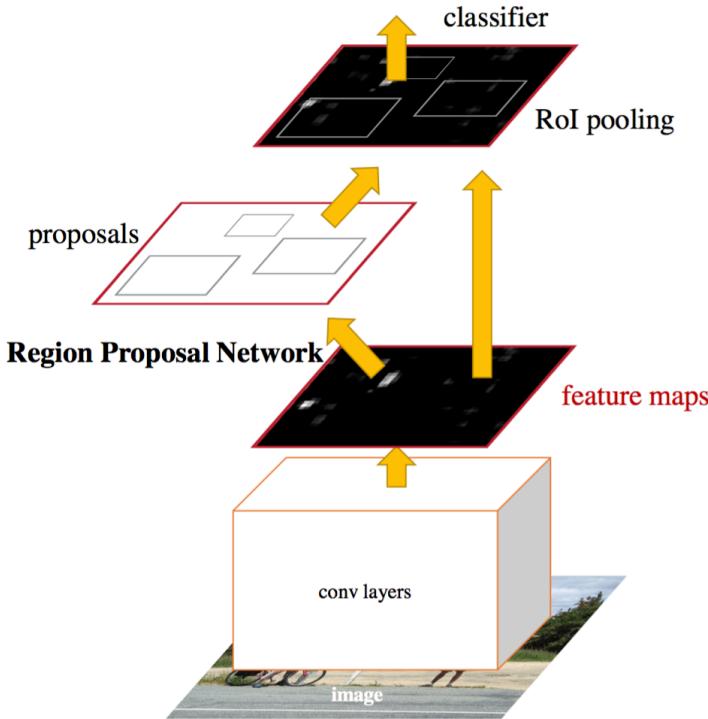


图 3.3 Faster R-CNN 结构图<sup>[9]</sup>

回归层)。整个 RPN 结构如图 3.4 所示。

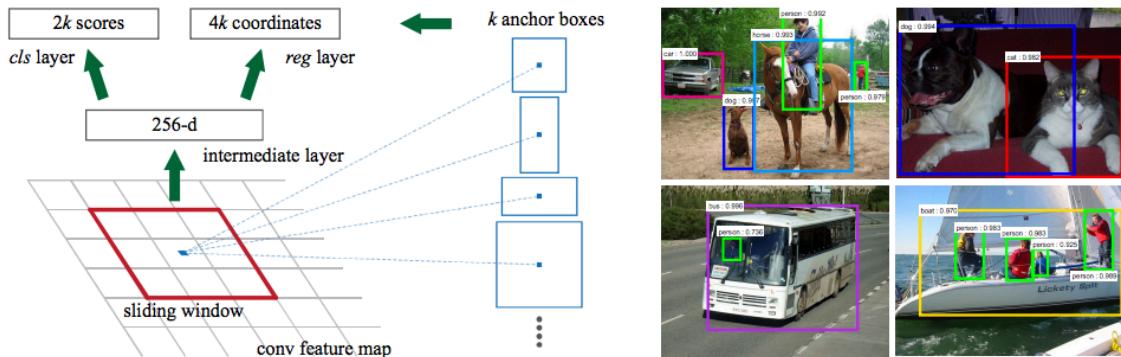


图 3.4 Region Proposal Network<sup>[9]</sup>

但是光有上述固定的网络无法处理不同尺寸、不同长宽比的目标物体，为了使 RPN 能够处理不同尺寸、长宽比的物体，作者提出了 Anchor 的概念。我们在滑动窗口每次滑到的位置，我们一共提出  $k$  个尺寸、长宽比不同的候选区域，每一个这样的区域成为一个 Anchor。所以我们网络的分类层有  $2k$  个输出，为每个 Anchor 是目标物体/背景的概率，网络的回归层有  $4k$  个输出，为每个 Anchor 的坐标。一般在一个位置，我们使用 3 个不同尺寸放缩比和 3 个不同长宽比共生成 9 个 Anchor。对一个尺寸为  $W \times H$  的特征映射图，我们共产生  $WHk$  个 Anchor。

为了训练 RPN，我们需要为每一个 Anchor 提供一个二元标签（代表是物体还是背景）。我们将两类 Anchor 标记为正例（是物体）：(1) 该 Anchor 和真实的边框有最高的 Intersection-over-Union(IoU) 重叠，或者 (2) 该 Anchor 和任一个真实的边框的 IoU 重叠都高于 0.7。值得注意的是，根据这种标记方法，一个真实的边框可能会产生多个标记为正例的 Anchor。有了如上对标记的定义，我们的损失函数定义如下：

$$L(\{p_i\}, \{t_i\}) = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*) + \lambda \frac{1}{N_{reg}} \sum_i p_i^* L_{reg}(t_i, t_i^*) \quad (3-4)$$

其中， $i$  为 Anchor 的下标， $p_i$  代表第  $i$  个 Anchor 是一个目标物体的概率， $p_i^*$  为真实值，当 Anchor 为正例时  $p_i^* = 1$ ，否则是 0； $t_i$  是一个四元组，包含表示 Anchor 坐标坐标的四个值， $t_i^*$  则是相应的真实值； $L_{cls}$  是常用于二分类问题的对数损失函数， $L_{reg}$  则同 Fast R-CNN 一样使用  $SmoothL_1$  损失函数； $N_{cls}$  和  $N_{reg}$  为两个归一化参数，在实现中常取  $N_{cls}$  为 mini-batch 的大小，而  $N_{reg}$  取不同位置 Anchor 的数量。此外，由于 RPN 在很多时候生成的候选区域彼此有很大的重叠，因此我们基于 RPN 网络对区域为目标物体可能性的打分以及区域间的 IoU，使用非极大抑制 (NMS) 对候选区域进行筛选，从而减少计算量。

综上所述，可以看出来 Faster R-CNN 实际上是由 RPN 和 Fast R-CNN 组合而成，因此我们在训练 Faster RCNN 时需要分别训练这两个网络。但是这两个网络并非完全独立，而是有着共享的卷积部分，因此如何才能正确的训练整个网络也是一个需要解决的问题。在 Faster R-CNN 的文献<sup>[9]</sup> 中，作者提出了一种 4 步训练法：第一步，我们使用预训练好的模型初始化 RPN 的卷积部分，并单独训练 RPN 网络，包括共用的卷积部分。第二步，我们使用预训练好的模型初始化 Fast R-CNN 的卷积部分，并使用上一步训练好的 RPN 网络产生候选区域来训练 Fast R-CNN 的识别网络。到这一步为止，RPN 和 Fast R-CNN 并不共享彼此的卷积部分。第三步，我们使用 Fast R-CNN 的卷积部分初始化 RPN 的卷积部分，这时候两个网络共享同样的卷积层，此时我们固定卷积层，对 RPN 独有的分类层和回归层进行训练。最后一步，我们仍旧固定卷积部分，对 Fast R-CNN 的分类、回归层进行训练。最后我们得到的两个网络共享同样的卷积层，但是却是不同的网络。

Faster R-CNN 技术不仅在 Pascal VOC 数据集和 COCO 数据集上取得了和

Fast R-CNN 比肩的出色效果，而且在 GPU 加速的帮助下可以达到 17fps 的处理速度，基本满足对实时性的需求。Faster R-CNN 无疑已成为时下最主流的物体识别技术之一。

### 3.2 使用 Faster R-CNN 进行车牌检测

由于 Faster R-CNN 在物体识别领域有着速度快、准确率高、鲁棒性好、人工干预少等优点，我们自然可以联想到将此技术用于车牌检测工作。下面我们将介绍如何使用 Faster R-CNN 实现车牌检测。

#### 3.2.1 数据集准备

##### 车牌图片

本文选择了 1000 张在道路上拍摄的含有车牌的照片，注意这些照片为步行或行车过程中使用手机拍摄而来，存在大量倾斜、变形的情况。

##### 数据标注

本文中首先对这些车牌的四个顶点进行手工标注，然后取包含车牌区域的最小外接矩形作为 bbox 用于 Faster R-CNN 的训练。

#### 3.2.2 算法实现

本文的 Faster R-CNN 实现基于开源项目 py-faster-rcnn<sup>①</sup>修改而来，增加了对上一步构造出的车牌数据的支持。

训练网络使用的工作站配置见附录中表 A.1。

训练数据集图片为 900 张，测试数据集图片为 100 张，模型参数见表 3.1。

表 3.1 Faster R-CNN 参数

训练第一步迭代次数	80000
训练第二步迭代次数	40000
训练第三步迭代次数	80000
训练第四步迭代次数	40000

---

<sup>①</sup><https://github.com/rbgirshick/py-faster-rcnn.git>

经过训练，Faster R-CNN 在 100 张测试数据集上精度达到 84.54%，相信如果换用更大的训练数据集后效果还有极大的提升空间，训练损失曲线如图3.5和3.6。

最后，使用 Faster R-CNN 对车牌进行检测效果如图3.7。

### 3.3 本章小结

本章首先介绍了使用 R-CNN 系列算法进行物体检测的基本原理，然后通过修改开源项目 py-faster-rcnn，在我们的车牌数据集上进行训练，最后验证其平均准确度可以达到 85% 左右，验证了使用 Faster R-CNN 进行车牌检测的有效性。

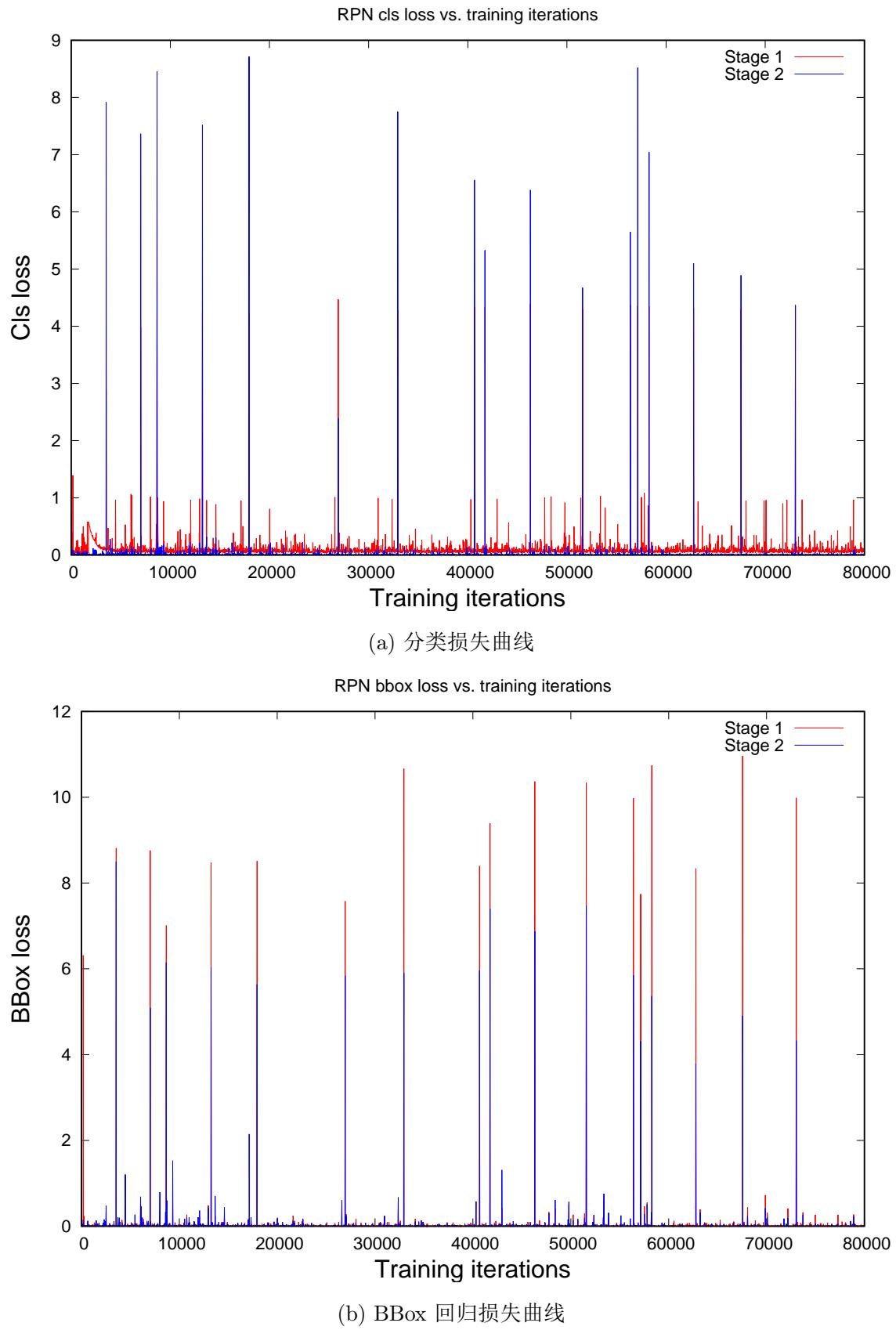


图 3.5 RPN 训练损失曲线

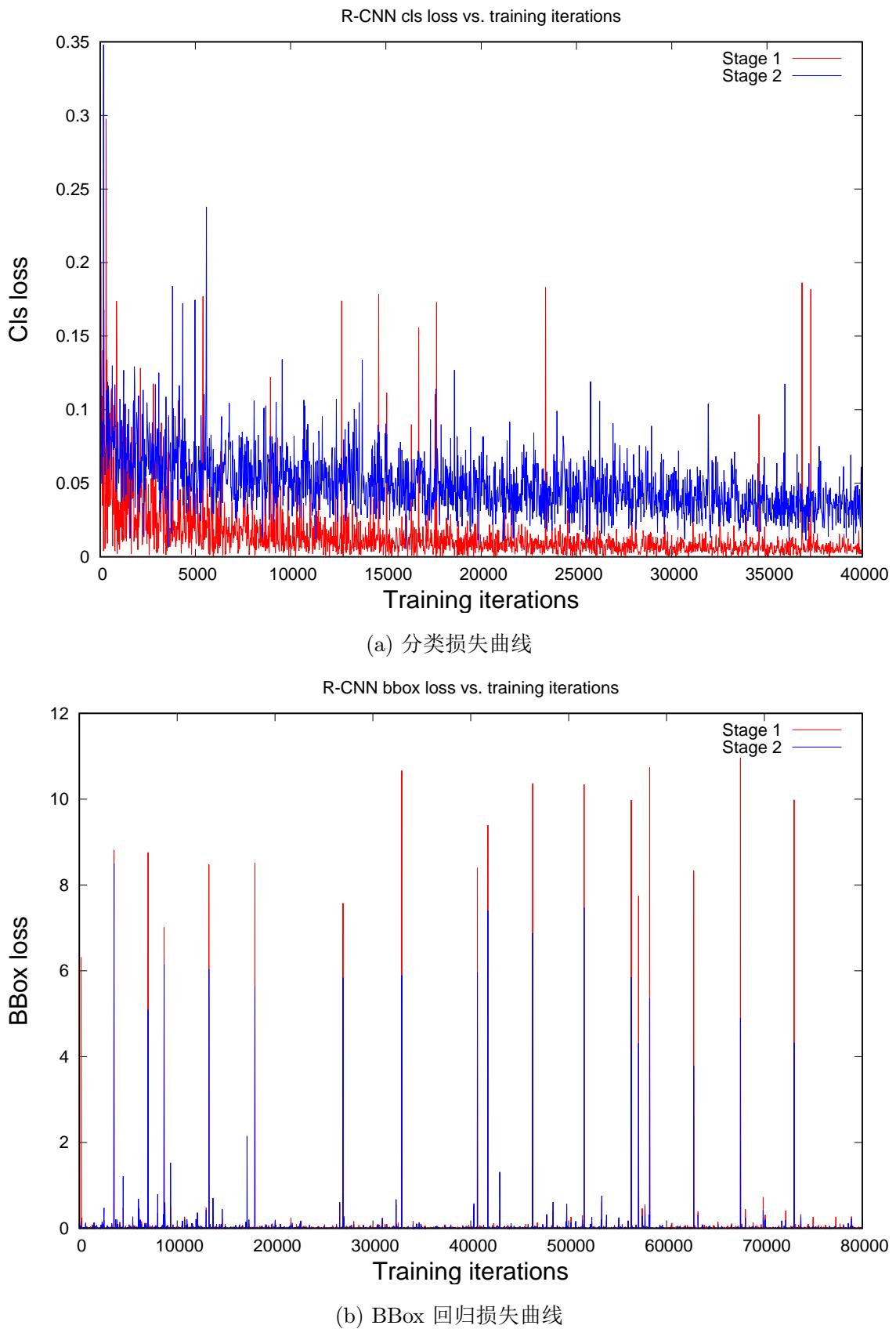


图 3.6 Fast R-CNN 训练损失曲线



(a) 示例一

(b) 示例二

图 3.7 使用 Faster R-CNN 进行车牌检测

## 第四章 车牌字符分割

传统的车牌字符分割有两种方法——基于连通域的分割方法和基于直方图垂直投影的分割方法。但这两种方法非常显著的缺点：一方面，基于连通域的方法取决于对车牌进行二值化操作的效果，而由于车牌受光照、拍摄角度、天气、污损等因素的影响，很难找到一种完美的二值化操作方法对图像进行二值化，这大大影响了连通域方法的效果；另一方面，基于直方图垂直投影的方法只能处理绝对水平的车牌，对存在倾斜、错切变形的车牌则无能为力，甚至相机抖动所产生的图像模糊也会影响该方法的效果，使之无法准确分割。因此本文放弃传统上述两种思路，尝试从字符检测与提取的角度来处理车牌字符分割问题。

### 4.1 Extremal Region 综述

#### 4.1.1 Extremal Region

我们首先定义一幅图像  $\mathbf{I}$  为一个映射  $\mathbf{I} : \mathcal{D} \subset \mathbb{N}^2 \rightarrow \mathcal{V}$ ，其中  $\mathcal{V}$  一般为  $\{0, \dots, 255\}^3$ （即一个色彩空间）。然后，我们定义图像的一个通道为  $\mathbf{C} : \mathcal{D} \rightarrow \mathcal{S}$ ，其中  $\mathcal{S}$  为一个全序集并且存在一个映射  $f_c : \mathcal{V} \rightarrow \mathcal{S}$  将像素值映射到该全序集。我们定义  $A$  为邻接关系  $A \subset \mathcal{D} \times \mathcal{D}$ ，常见的邻接关系有 4-邻接和 8-邻接，在本章的实现中我们使用 4-邻接关系。

我们定义图像  $I$ （或通道  $C$ ）的一个 Region 为  $\mathcal{D}$  的一个连续子集（所为连续，是指  $\forall p_i, p_j \in \mathcal{R}, \exists p_i, q_1, q_2, \dots, q_n, p_j : p_i A q_1, q_1 A q_2, \dots, q_n A p_j$ ）。我们定义 Region 边界  $\partial \mathcal{R}$  为那些与 Region  $\mathcal{R}$  邻接却不属于  $\mathcal{R}$  的像素的集合，即  $\partial \mathcal{R} = \{p \in \mathcal{D} : \exists q \in \mathcal{R} : p A q\}$ 。现在，我们定义 Extremal Region(ER) 为那些边界像素值比内部像素高许多的 Region，写成数学语言即  $\forall p \in \mathcal{R}, q \in \partial \mathcal{R} : \mathbf{C}(q) > \theta > \mathbf{C}(p)$ ，其中  $\theta$  为 ER 的阈值。由上式的定义，我们注意到一个阈值为  $\theta$  的 ER  $r$  可以由多个或零个阈值为  $\theta - 1$  的 ER 和值为  $\theta$  的像素和并集： $r = (\bigcup_{u \in R_{\theta-1}} u) \cup (\bigcup_{p \text{ in } \mathcal{D} : \mathbf{C}(p) = \theta} p)$  构成，其中  $R_{\theta-1}$  表示阈值为  $\theta_1$  的 ER。这一性质将用于下一节计算 CSER 的过

程中。

#### 4.1.2 Class-specific Extremal Region

由上一节可知，对于一幅图像，字符区域一定是 ER，但反之不然。因此，对于字符提取任务，我们只需要将一幅图像的 ER 中非字符的那些去掉，只保留字符区域的 ER 即可，这样，我们的问题就转换为如何识别一个 ER 是否包含有效的字符这一判定问题。而 Class-specific Extremal Region(CSER)<sup>[10]</sup> 的核心思想正是通过机器学习的方法，使用一个分类器对 ER 进行筛选，从而快速准确地提取图像中的字符区域。因此整个 CSER 算法可以归结为两部分：ER 区域的产生和 ER 区域的判别。

#### ER 的搜索

由于阈值为  $\theta$  的 ER 可以由阈值为  $\theta - 1$  的 ER 区域所产生，因此我们可以从 0 开始逐步增加阈值  $\theta$  从而逐步产生 ER，这些 ER 根据其产生关系可以组成一个树状结构（如图4.1，我们对 ER 的搜索过程便是自顶向下遍历这棵树的过程。但是，这棵树往往包含数量巨大的 ER 区域，若不对搜索过程进行剪枝，则无论是在时间上还是内存上都将难以承受。因此，我们在搜索过程中对每个 ER 使用识别器进行判别，若其为字符区域的概率小于给定阈值，我们则放弃在该节点上的进一步搜索（即剪枝），从而减少大量不必要的计算。下一节，我们将描述如何设计这种分类器，以实现对 ER 的判别。

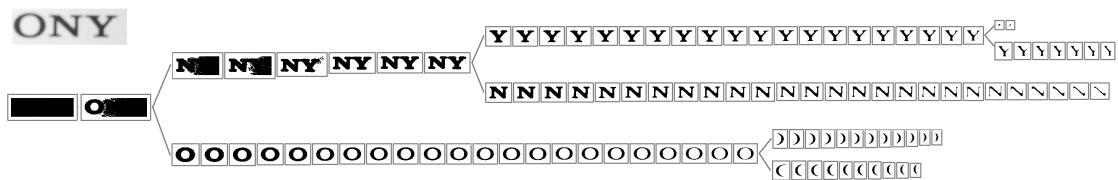


图 4.1 ER 搜索过程中产生的搜索树

#### ER 的判别

若要实现对 ER 区域的判别，最简单也是最理想的情况是使用一个复杂分类器（如 SVM、CNN）直接对 ER 区域的图像进行判别。但是此种分类器往往相对较慢，无法达到对实时性的要求。因此 Neumann 等人在发明 CSER 时提出使用一

些能够快速计算且效果良好的简单特征作为分类器的输入进行分类。注意到，我们使用 CSER 进行字符 ER 提取的时候一共使用了两次分类器——第一次是使用分类器对 ER 搜索树进行剪枝，第二次则为最终输出时对 ER 区域进行判别和筛选。显然，这两次对分类器的使用有着不同的要求：用于搜索的分类器对精度要求低但要求计算速度快；而用于最终输出的分类器对性能要求不高但要求精度更高。因此，我们在此分别设计这两个分类器，对于搜索过程中使用的分类器（下文中将称之为分类器 1）我们使用少但容易计算的特征进行分类，而对于最终输出时使用的分类器（下文中将称之为分类器 2）我们使用计算量更大但更有效的特征进行分类。

为了性能需要，我们希望对于阈值为  $\theta$  的 ER 的特征可以从  $\theta-1$  的 ER 是的特征快速计算得出，即假设我们已知特征  $\phi(u), \forall u \in R_{\theta-1}$ ，我们希望对每个阈值为  $\theta$  的 ER  $r = (\bigcup u \in R_{\theta-1}) \cup (\bigcup p \in \mathcal{D} : \mathbf{C}(p) = \theta)$ ，其特征  $\phi(r) = (\oplus \phi(u)) \oplus (\oplus \psi(p))$ ，其中  $\psi(p)$  被称为初始化函数， $\oplus$  代表将特征进行合并的算子。作者称这类特征可增量计算描述子。在文献<sup>[10]</sup> 中，作者首先定义了如下可增量计算描述子：

- 面积  $a$ : ER 区域的面积（即像素数量）。
- 边框  $(x_{min}, y_{min}, x_{max}, y_{max})$ : ER 区域的右上角和左下角坐标。
- 周长  $p$ : ER 区域边界的长度（即像素数量）。
- 欧拉数  $\eta$ : 欧拉数是二值图像的一种拓扑特征，为连通域数目和孔洞数目之差。
- 水平交叉点数  $c_i$ : 用一个长度为图像高度的向量来保存对应行像素在属于 ER 与不属于 ER 之间转变的次数。

对于以上描述子，我们可以很容易的找到其对应的初始化函数  $\psi(p)$  和合并算子  $\oplus$ 。

对于分类器 1，我们基于基于上面的描述子使用如下特征：

- 长宽比  $(w/h)$
- compactness( $\sqrt{a}/p$ )

- 孔洞数  $(1 - \eta)$
- 水平交叉点特征  $(\hat{c} = median(\mathbf{c}_{\frac{1}{6}w}, \mathbf{c}_{\frac{3}{6}w}, \mathbf{c}_{\frac{5}{6}w}))$

对于分类器 2，由于我们需要其精度更高，因此我们额外使用如下特征：

- 孔洞面积比  $a_h/a$ : 其中  $a_h$  代表 ER 区域内孔洞的面积（像素数）。
- 凸包面积比  $a_c/a$ : 其中  $a_c$  为 ER 区域凸包的面积。
- 外轮廓拐点数  $\kappa$ : 代表 ER 区域边界凹角与凸角的变化数目。

此外，为了平衡性能与训练难度，两个分类器我们均采用基于决策树的 AdaBoost 分类器。

## 4.2 使用 Class-specific Extremal Region 进行车牌字符分割

### 4.2.1 算法设计

#### 候选区域的提取

我们首先将一幅图像划分为  $R, G, B, -R, -G, -B, \nabla$  七个通道，其中  $R, G, B$  为红、绿、蓝三色通道， $-R, -G, -B$  为对相应颜色通道进行反转所得到的通道， $\nabla$  为梯度。然后我们分别对这些通道提取 CSER 作为候选区域。

#### 候选区域的进一步筛选

得到所有 CSER 后，仍不免有很多误判和包含不完整字符的区域（如图4.2），在此我们就需要使用一些车牌字符的性质对这些候选区域进行进一步筛选。

在我们的算法中我们从如下方面对候选字符区域进行筛选：

- 长宽比根据中华人民共和国公共安全行业标准 GA36—2007 中关于车牌的相关规定，我们限定候选字符的长宽比在区间 0.3 到 10 之间
- 尺寸我们根据候选区域相对于图像的尺寸，去除那些过大或过小的区域
- 区域中心位置根据区域的中心位置，合并那些 IoU 较大的候选区域，去除不可行的区域

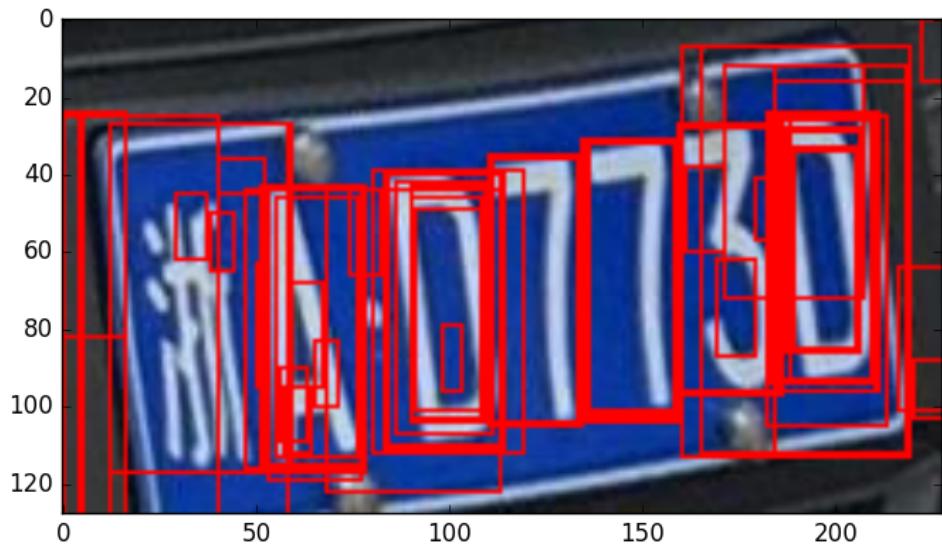


图 4.2 使用 CSER 提取出的所有候选区

此外，有时候可能会出现缺失部分字符的情况，在这种时候我们将根据已取得的字符位置和尺寸，对缺失的字符区域进行推算。

通过上述步骤，最终我们得到车牌中后六个字符的位置（即英文和数字部分），接下来，我们将通过这六个字符的位置推算中文字符位置

### 中文字符的处理

由于 CSER 是基于连通域进行字符提取的，因此它无法直接处理中文这种由多个连通域组成的字符（如“穿”、“京”）。为了解决这一问题，在本文中我们使用的方法是，仅使用 CSER 提取车牌中的英文和数字，然后根据车牌的结构特点和字符间的相对位置关系对中文字符的位置进行推算。

#### 4.2.2 算法实现

CSER 的相关算法已经在最新版的 OpenCV Contrib<sup>①</sup> 中得以实现；CSER 所使用的两个分类器的训练则由作者基于 OpenCV 使用 C++ 开发完成；最后整个字符分割模块使用 Python 语言和最新版的 OpenCV（需手工编译最新版）开发完成。字符分割效果如图 4.3。

可以看出基于 CSER 的车牌分割方法即使在车牌有旋转、变形、污损的情况下

<sup>①</sup>[https://github.com/Itseez/opencv\\_contrib](https://github.com/Itseez/opencv_contrib)

下，仍可以准确的从车牌图像中找寻到字符外接轮廓，具有极强的鲁棒性，且检测速度快，非常适用于车牌识别系统中并代替传统的字符分割方法。

### 4.3 本章小结

本章受自然场景中字符识别任务的启发，提出使用 CSER 技术对车牌图像中的字符进行分割、提取，克服了传统方法的诸多缺点，具有准确率高、鲁棒性强、性能好等特点。但是另一方面，本方法仍称不上完美，还有许多可以改进的空间：其一，CSER 对字符区域的提取基于连通域，而中文字由于其往往包含多个连通域（如“川”字），因此不能直接使用 CSER 技术进行提取，一种可行的改进是使用<sup>[11]</sup> 中提出的搜索方法对 ER 区域进行合并操作，以求取中文字；另一方面，可以在 CSER 算法中加入非极大抑制来对 ER 进行预筛选。此外，由于 CSER 算法具有并行的特点，因此可以考虑进行 GPU 加速来提升执行性能。

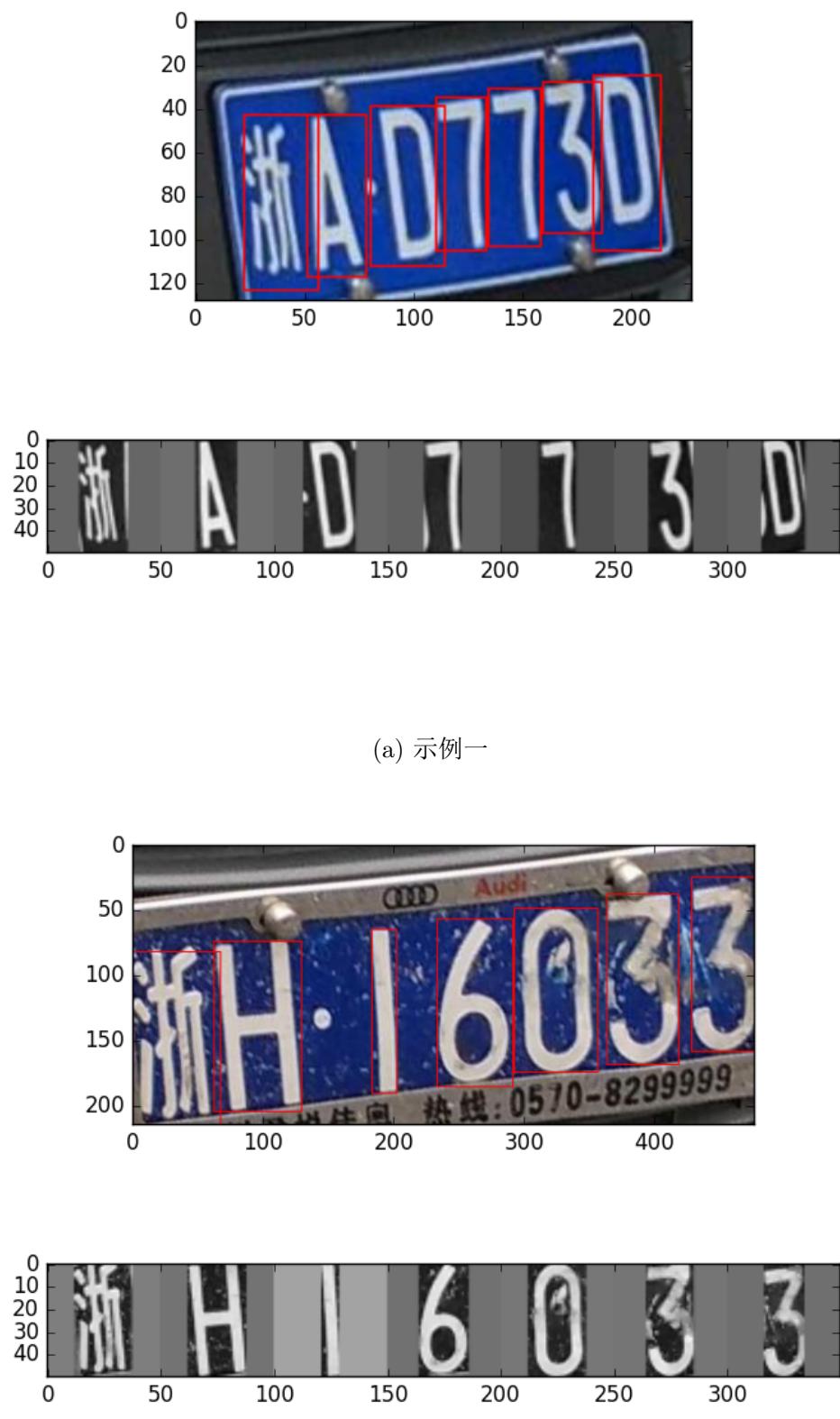


图 4.3 车牌字符分割



## 第五章 车牌字符识别

车牌识别是车牌识别系统最后也是最关键的一步，幸运的是，字符识别问题已经被人们充分的研究了几十年，并且提出了一系列十分有效的方法。早期的字符识别系统中经常使用 SVM 作为分类器对字符进行识别，但是随着深度学习的发展，人们发现 CNN 能在字符识别任务上取得非常优秀的效果，因此越来越的字符识别系统选择使用 CNN 作为其字符识别算法。本文也采用了 CNN 对车牌字符进行识别，下面对此进行描述。

### 5.1 使用 CNN 进行多类别分类

在前几章中我们已多次使用 CNN 来完成回归任务，并使用  $SmoothL_1$  损失函数进行梯度下降优化。虽然，分类任务可以看做是一种特殊的回归任务，但显然是  $SmoothL_1$  损失函数并不适合于分类问题。下面我们将介绍用于回归问题的 Softmax 激活函数和交叉熵损失函数。

#### 5.1.1 Softmax 激活函数

对于分类任务，首先我们考虑如何表示类别信息，目前最常用的表示方法当属 One-Hot 表示法——即对于一个  $N$  类别分类问题，我们使用一个  $N$  元组  $(p_1, \dots, p_N)$  表示其类别信息，其中  $p_i$  为目标属于第  $i$  类的概率。例如，对于真实类别为  $j$  的目标，可令  $p_j$  等于 1 而其他项等于 0。

于是，对于一个用于  $N$  类别分类问题的神经网络，我们很自然的选择 One-Hot 形式的表示作为网络的最后一层输出，即最后一层的第  $i$  个神经元输出目标属于第  $i$  类的概率，由于输出的是概率值，因此我们要选择特殊的激活函数以将输出幅度限制在 0 到 1 之间。一种选择是使用 Sigmoid 函数，不过后来人们根据广义线性模型提出了更加适合该问题的 Softmax 函数：

$$p(y = j | \mathbf{x}; \mathbf{w}) = \frac{e^{\mathbf{x}^T \mathbf{w}_j}}{\sum_i^K e^{\mathbf{x}^T \mathbf{w}_k}} \quad (5-1)$$

其中  $w$  为网络权重参数。

### 5.1.2 交叉熵损失函数

选择了 Softmax 激活函数，自然也要选择与之配套的损失函数。最小二乘损失是一个很容易想到的损失函数，但是却未必是最适合多分类问题的损失函数。业已证明，最适合与 Softmax 激活函数配合使用以进行多目标分类的损失函数是交叉熵损失函数：

$$Loss(\mathbf{x}) = \sum_{j=1}^K [y = j] \log p(y = j | \mathbf{x}; \mathbf{w}) \quad (5-2)$$

其中  $[y = j]$  为指示函数，当  $y = j$  时为 1，否则为 0。

现在我们求交叉熵损失函数对 Softmax 层参数  $w$  的导数，为此我们引入如下记号： $x$  代表 Softmax 层输入； $L$  代表损失函数， $o_j = p(y = j | \mathbf{x}; \mathbf{w})$ ，即 Softmax 层第  $j$  个神经元的输出； $z_j = \mathbf{x}^T \mathbf{w}_j$ 。于是根据上述定义和链式求导法则，我们有：

$$\frac{\partial L}{\partial w_{ji}} = \frac{\partial L}{\partial o_i} \frac{\partial o_i}{\partial x_i} \frac{\partial x_i}{\partial w_{ji}} = (o_i - [y = i]) o_j \quad (5-3)$$

我们记  $t_i$  为样本的实际类别信息，即  $t_i = 1$  当且仅当  $y = i$ ，否则为 0。因此我们可以将上述导数公式写为梯度形式：

$$\nabla_{\mathbf{w}} L = (\mathbf{o} - \mathbf{t}) \times \mathbf{o} \quad (5-4)$$

其中  $\mathbf{o}, \mathbf{t}$  为  $o_i, t_i$  的向量形式， $\times$  表示外积（叉积）。有了上述梯度公式，我们就可以通过反向传播算法对 CNN 进行训练。

## 5.2 使用 CNN 进行车牌字符识别

### 5.2.1 模型设计

根据之前的叙说，我们设计如图5.1的两个 CNN 分别进行中文和数字、英文的字符识别，网络的最后一层使用 Softmax 激活函数，其它层则使用 ReLU。

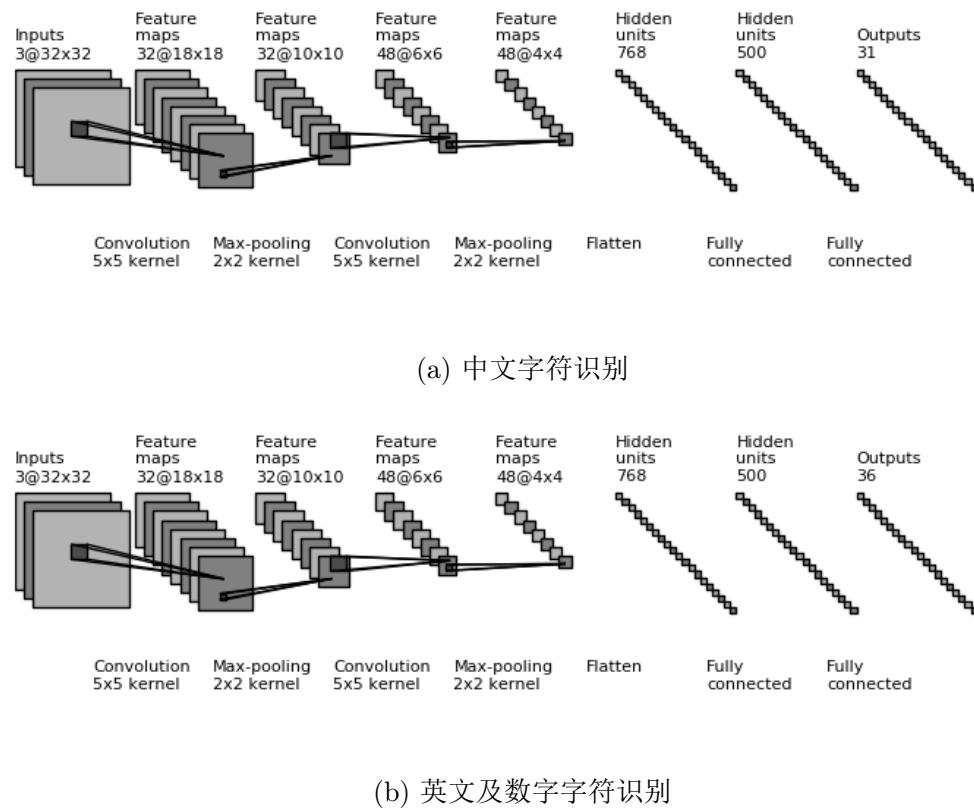


图 5.1 字符识别 CNN 模型

### 5.2.2 算法实现

本章所用 CNN 基于 Google 出品的著名开源深度学习引擎 TensorFlow<sup>①</sup>进行实现，开发语言为 Python，此外还需要开源库 Keras 进行配合。本章两个神经网络训练训练参数见表5.1：

表 5.1 CNN 训练参数

batch 大小	128
迭代次数	20
验证集站训练集比例	20%
优化算法	AdaDelta <sup>[12]</sup>

训练结果表明，网络在中文识别任务上可以达到 92% 的准确率，在英文和数字集上的达到 99% 的准确率，充分说明其在字符识别问题上的有效性。

同时，分析和实验发现，当训练数据集比较小的时候，很容易发生过拟合现象而导致测试识别率反而下降，因此为了进一步提升精度，我们需要使用更大的训练集进行训练。

<sup>①</sup><https://www.tensorflow.org>

### 5.3 本章小结

本章提出了使用 CNN 进行字符识别的模型，并通过实验验证了其在车牌识别问题上的有效性，从而完成了车牌识别系统中最后也是最关键的一步。

## 第六章 实验结果

最终我们将上述各个子系统串联起来形成一个完整的车牌检测、识别系统。系统运行效果如图6.1。



图 6.1 系统运行效果

性能方面，在分辨率为  $800 \times 600$  的图像上，车牌检测使用 GPU 加速需要 90 毫秒，字符分割需要约 400 毫秒，字符识别使用 CPU 需要约 90 毫秒。可以看出，字符分割是制约性能的瓶颈，不过本文中字符分割使用 Python 实现，未使用多线程优化，相信在此还有进一步提升性能的可能。

## 第七章 总结与展望

### 7.1 总结

自从 2006 年 Hinton 带领神经网络学派以深度学习的新名字复苏以来，深度学习由于其令人惊艳的性能和易于使用等特点，无论是在学术界还是在工业界都有着十分广泛的应用。今年三月 Google DeepMind 团队开发的 AlphaGo 以 4:0 的优秀成绩大败人类围棋冠军李世乭轰动世界，而深度学习、人工智能等名词常见于各种媒体平台，称为当之无愧的流行用语。此外，无论是金融圈还是 IT 业，都有无数的公司投入大量的资金和人力进行深度学习、人工智能的研究，毋庸置疑，机器学习技术，尤其是深度学习技术已成为时下最热门的研究领域。不仅如此，深度学习技术也在潜移默化间改变着我们的生活。小到越来越准确的语音识别技术、越来越贴合用户需求的搜索引擎、购物平台、音乐推荐系统等，大到可以下赢世界冠军的 AlphaGo 围棋程序，都离不开深度学习技术的助力，甚至连 NVIDIA 公司也借卖显卡做深度学习而发了财。可以预见，在不远的将来深度学习将像两次科技革命一样颠覆我们的生活，造福人类社会。本文试图将最新、最好的深度学习技术引入车牌识别这一任务中，从而实现更鲁棒、更智能的车牌识别系统。具体来讲，有以下几点：首先，本文试图使用 Faster R-CNN 这一目标检测神器来实现车牌的检测。基于 Faster R-CNN 的方法能有效克服传统方法不够鲁棒、依赖先验知识、特殊情况多等缺点，实现更加智能、更加鲁棒的车牌定位系统。再次，本文提出使用 CNN 回归的方式对车牌坐标进行精确定位，从而抛弃传统的基于边缘检测的方法，以提高性能的适应性。然后，本文抛弃了传统的基于连通域和垂直投影的车牌分割方法，尝试将车牌字符分割变为一个文字检测问题，并通过 Class-specific Extremal Region 的方法进行解决，有效克服了两种传统方法的缺点。最后，本文使用 CNN 技术进行车牌字符识别，获得了极高的准确率。将中文字符和英文数字分开训练的做法更是增加了识别的准确率和可靠性。

## 7.2 展望

本文虽然使用深度学习技术进行车牌识别，克服了传统方法的诸多缺点，但也可以看到深度学习方法也不是完美的，甚至在很多应用场景下性价比不如传统方法高。展望未来工作，我希望以下几个方面对整个系统进行改进：

第一，Faster R-CNN 技术虽然在目标检测问题上表现优异，但其速度仍旧偏慢，不能很好地适用于高速公路等对检测速度要求高的场合。而近年流行的另一种基于 CNN 的目标检测技术——You Only Look Once——简称 YOLO，虽然准确率不如 Faster R-CNN，却有着非常快的速度，因此可以考虑使用 YOLO 代替 Faster R-CNN 技术进行车牌目标检测。第二，在车牌字符分割过程中，我们需要去除大量假阳性的 Extremal Region，目前我们采用的方法是基于车牌字符的相对位置和尺寸进行筛选，但这种方法严重依赖先验知识和对筛选规则的制定，造成了大量无法处理的特殊情况。因此我们希望通过机器学习技术完成 ER 的合并工作，一种可行的思路是使用文献<sup>[13]</sup> 中的方法进行区域聚合，来同时实现车牌文字区域的精确定位和分割。

## 致谢

在论文完成之时，我要感谢两位指导老师：首先是我毕业设计的指导老师，西安电子科技大学的李辉老师，他不仅悉心指导我查找文献、撰写论文，还为我提供了实验室环境和工作站；然后我要感谢我的研究生导师，浙江大学 CAD 实验室的蔡登教授，是他为我指明了大方向，并帮助我修正我的思路。没有他们的帮助，我的论文也不能进展的如此顺利。

在此，祝二位指导老师身体健康、事业有成，桃李满天下。

然后，我还要感谢诸多学长对我的悉心指导和帮助，是他们帮我解决了许多技术细节上的问题和困惑，帮助我开发完善相关代码。

在此，祝学长们学业有成，万事如意。

最后，应当感谢崔元顺同学和齐飞老师提供本 L<sup>A</sup>T<sub>E</sub>X 模板，谢谢他们的无私贡献。



## 参考文献

- [1] B B Le Cun, J S Denker, and D Henderson. Handwritten digit recognition with a back-propagation network. *Advances in neural ...*, 1990.
- [2] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet Classification with Deep Convolutional Neural Networks. *NIPS*, 2012, pages 1106–1114.
- [3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. *arXiv.org*, 2015, December.
- [4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity Mappings in Deep Residual Networks. *arXiv.org*, 2016, March.
- [5] Vinod Nair and Geoffrey E Hinton. Rectified Linear Units Improve Restricted Boltzmann Machines. *ICML*, 2010, pages 807–814.
- [6] Diederik Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *arXiv.org*, 2014, December.
- [7] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 580–587. IEEE, 2014.
- [8] Ross B Girshick. Fast R-CNN. *ICCV*, 2015, pages 1440–1448.
- [9] Shaoqing Ren, Kaiming He, Ross B Girshick, and Jian Sun 0001. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *NIPS*, 2015, pages 91–99.
- [10] Lukas Neumann and Jiri Matas. Real-time scene text localization and recognition. *2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012, pages 3538–3545.
- [11] Lukas Neumann and Jiri Matas. Text Localization in Real-World Images Using Efficiently Pruned Exhaustive Search. *Document Analysis and Recognition ( ... , 2011*, pages 687–691.
- [12] Matthew D Zeiler. ADADELTA: An Adaptive Learning Rate Method. *arXiv.org*, 2012, December.
- [13] Lluis Gomez and Dimosthenis Karatzas. A Fast Hierarchical Method for Multi-script and Arbitrary Oriented Scene Text Extraction. 2014, July.



## 附录 A 实验环境

表 A.1 实验用工作站配置

操作系统	Ubuntu Server 14.04
CPU	Intel E5-2603 v3
内存	32GB DDR3
显卡	NVIDIA Quardo M5000 x2
Python 版本	Python 3.5(Anaconda3 4)
CUDA 版本	CUDA 7.5 with CuDNN