



山东大学 (威海)
SHANDONG UNIVERSITY, WEIHAI

毕业论文（设计）

设计(论文)题目 基于深度学习的目标识别方法研究

姓 名: 于超辉
学 号: 201300800660
学 院: 机电与信息工程学院
专 业: 通信工程
年 级 2013 级
指导教师: 郭尊华

2017 年 5 月 10 日

目 录

摘 要.....	1
一、绪论.....	4
(一) 研究背景与意义.....	4
(二) 国内外研究现状.....	4
(三) 本论文的主要内容.....	5
二、卷积神经网络模型基础理论.....	6
(一) 卷积神经网络.....	6
1. 卷积神经网络概念.....	6
2. 卷积神经网络中层的策略.....	6
(二) ZF 和 VGG 模型简介.....	8
1. ZF 网络模型理论简介.....	8
2. VGG 网络模型理论简介.....	9
(三) Faster R-CNN 模型原理.....	10
1. 基于区域的卷积神经网络 R-CNN.....	10
2. 快速基于区域的卷积神经网络 Fast R-CNN.....	10
3. Faster R-CNN.....	11
三、简易分布式爬虫系统设计与数据集制作.....	15
(一) 爬虫系统框架.....	15

1. 爬虫框架 Scrapy 简介.....	15
2. 分布式爬虫系统设计架构.....	15
3. 爬虫运行结果.....	16
(二) 数据集制作.....	17
1. 数据集标注.....	17
2. 训练测试数据集制作.....	18
四、网络模型的搭建、训练与测试.....	19
(一) 网络模型搭建.....	19
1. 模型训练与测试平台.....	19
2. ZF 网络模型搭建.....	20
3. VGG 网络模型搭建.....	20
(二) 模型训练.....	21
1. 模型训练环境.....	21
2. ZF 网络模型训练.....	21
3. VGG 网络模型训练.....	23
4. 两种网络模型训练结果分析.....	24
(三) 模型测试.....	25
1. ZF 模型测试.....	25
2. VGG 模型测试.....	26

3. 两种网络模型测试结果分析.....	28
五、总结与展望.....	29
参考文献.....	30
谢 辞.....	32

摘要

随着计算机软硬件的发展、智能电子设备的普及、社交媒体的广为传播，每天都会产生海量的图像数据，如何从图像中提取感兴趣的信息，具有实用意义。目标识别的任务是检测和定位图像中的目标，得益于卷积神经网络（Convolutional Neural Network, CNN），基于深度学习的目标识别技术得到了飞速发展，经过一系列研究与应用，研究学者提出了很多基于深度学习的目标识别方法，基于区域的卷积神经网络（Region-based Convolutional Neural Network, R-CNN）和区域提取网络(Regina proposal network, RPN)的 Faster R-CNN 在 PASCAL VOC2007 数据集的目标识别中实现了最先进的性能。本文在大量前人研究成果的基础上，通过爬取大约 60 万张百度街景图片作为数据集，这些图像涵盖了光照和天气条件的变化，并利用第三方开源工具（Convolutional Architecture For Feature Extraction, Caffe）和 Python 语言构建了两种 Faster R-CNN 模型，并完成了对模型的训练、测试和对比分析，实验结果表明，最好的模型的识别平均精度（mean Average Precision, mAP）能达到 74.03%。

关键词

深度学习，目标识别，卷积神经网络，Faster R-CNN，Caffe

Abstract

With the development of computer hardware and software technology, popularity of intelligent electronic devices, the spread of

social media, every day massive image data will be produced, how to extract the information of interest from images, which has practical significance. The task of target recognition is to detect and locate the target in images. Thanks to the Convolutional Neural Network (CNN), the target recognition technology based on deep learning has been developed rapidly. After a series of research and application, researchers have proposed a number of target recognition methods based on deep learning, the Region-based Convolutional Neural Network (R-CNN) and Region Proposal Network (RPN) based model, Faster R-CNN, achieved state-of-the-art performance at target recognition on the PASCAL VOC2007 datasets. This paper is based on a large number of previous research results, by crawling about 600,000 Baidu Street View as datasets, which cover large variations in illumination and weather conditions, and using an open source deep learning toolbox called Convolutional Architecture For Feature Extraction (Caffe) and Python language, two Faster R-CNN models were built. And completed the training, testing and comparative analysis of the models. The experimental results show that the best model's mean Average Precision (mAP) can reach 74.03%.

Keywords

Deep Learning, Target Recognition, Convolutional Neural Network,

Faster R-CNN, Caffe

一、绪论

(一) 研究背景与意义

随着计算机视觉技术研究的进一步加深,如何从图像中更精确地识别出目标信息成了迫切需求。不仅图像的简单分类领域受到了人们的关注,而且人们越来越希望准确获得图像中目标的语义类别和位置,因此图像中目标识别技术受到广泛关注。随着社交媒体的广为传播,每天都会产生大量的图像等数据,如何更准确更高效地从图像中提取感兴趣的信息,成为了重要问题,得益于大数据、计算能力的提升和卷积神经网络,基于深度学习的目标识别技术得到了飞速发展和应用。

传统目标识别方法一般包括区域选择、特征提取和分类器分类三大部分,其缺点一个是基于滑动窗口的区域选择策略没有针对性,时间复杂度高,窗口冗余;二是手工设计的特征对于多样性的变化并没有很好的鲁棒性,而深度学习相对于之前的方法,只要给它足够的符合标准的数据,它就能够自主的学习到数据内的特征,为后续的任务提供特征来源。本课题研究基于深度学习的道路交通信息中的目标识别,在自动驾驶、机器人感知、智慧交通等领域具有重要的理论和实践意义。

(二) 国内外研究现状

目标识别是指一个特殊目标或一种类型的目标从其它目标或其它类型的目标中被区分出来的过程^[1]。基于图像的目标识别技术是指能从背景复杂的图像中检测和定位出感兴趣的目标。该领域的研究始于二十世纪六十年代,传统图像目标识别方法一般包括区域选择、特征提取和分类器分类,其中区域选择阶段常采用滑动窗口策略;特征提取是传统方法中至关重要的一部分,该阶段要从图像中提取感兴趣的重要信息,以便下一步训练分类器。常用的特征有 SIFT 特征点^[2]、Kadir 和 Brady 显著点^[3]、多尺 Harris 角点等^[4];训练分类器阶段常用的方法有 Boosting 分类法^[5]和支持向量机 (Support Vector Machine, SVM)^[6]。在传统方法中,挑选何种特征是一大难题,虽然人们提出的特征很多,但是至今还没有哪一种特征可以万能地处理一切图像目标识别问题,人们在挑选特征时往往靠经验和运气。近年来,随着 GPU 计算能力的大幅提升和海量数据的涌现,基于卷积神经网络 (Convolutional Neural Network, CNN)^[7]的深度学习技术在图像目标识别中取得了优异的成绩。

深度学习理论由加拿大多伦多大学的 Geoffrey Hinton 教授带领的团队在 2006 年的《Science》杂志上发表的论文中提出^[8],引起了学术界的关注。Yann Le Cun 教授最早提出了 CNN 模型, CNN 属于深度学习领域中的一种经典模型,在计算机视觉技术中有着极高的效率。Hinton 教授带领团队构建的深度卷积神经网络模型,在 Image Net 2012 的比赛中将 Top5 错误率由 25% 降低至 17%^[9]。在图像目标识别领域, Ross B. Girshick (RBG) 使用区域提取结合 CNN 代替传统目标检测设计了 R-CNN 框架^[10],使得目标识别取得巨大突破,并带动了基于深度学习的目标识别热潮。RBG 还提出了 Fast R-CNN,大大提高了检测速度,在

VOC2007 上的平均正确率百分比（mAP）也由 58% 提高到了 68%。

2014 年，香港中文大学的汤晓鸥教授领导的小组开发了模型 DeepID-Net^[11]，该模型的人脸识别准确率超过 99%。2015 年，何凯明和 RBG 等人提出了 Faster R-CNN^[12]，速度更快了，且在 VOC2007 上 mAP 能到 73%。此外，电子科技大学、中科大、国防科大等在基于深度学习的目标识别研究中也做出了突出贡献。程欣创新的提出了 N-VGG 网络模型，提升了 RPN 网络的性能^[13]；阮怀玉提出了基于多尺度稀疏表示和深层去噪自编码网络的 SAR 图像目标识别算法^[14]；傅瑞罡改进了 HMAX 模型中存在的采样层特征块冗余情况明显的问题，并提出了一种基于梯度场的可去冗余的 SIFT 特征点提取方法^[15]。

（三）本论文的主要内容

基于深度学习的目标识别技术是计算机视觉领域的热点，本课题在大量前人研究成果的基础上，主要利用 Faster R-CNN 完成对深度学习中两种典型的网络模型（ZF^[16] 和 VGG^[17]）的研究和 Python 实现，并对模型进行了训练、测试和对比分析。

本文的主要章节内容如下：

第一章 绪论，分析基于深度学习的目标识别方法的研究背景，研究意义和国内外研究现状，对本文的研究内容进行概述。

第二章 卷积神经网络模型基础理论，介绍卷积神经网络和基于卷积神经网络的 ZF 网络和 VGG 网络的基础理论知识；介绍 Fater R-CNN 目标识别技术的发展、基础理论和识别方法。

第三章 简易分布式爬虫系统设计与数据集制作，介绍爬虫系统的设计框架和从百度街景中爬取的数据，制作数据集用于训练和测试。

第四章 网络模型的搭建、训练与测试，基于 ZF 网络和 VGG 网络，用 Python 进行模型搭建，在 GPU 平台上进行模型训练和测试，对结果进行分析比较。

第五章 总结与展望，总结本文所作的主要工作，并展望今后的重点研究方向。

二、卷积神经网络模型基础理论

本章介绍卷积神经网络和基于卷积神经网络的 ZF 网络和 VGG 网络的基础理论知识；介绍 Fater R-CNN 目标识别技术的发展、基础理论和识别方法。

(一) 卷积神经网络

1. 卷积神经网络概念

在深度学习中，CNN 是一种前馈人工神经网络，其神经元之间的连接模式受动物视觉皮层的启发^[19]。单个神经元对被称为接受场的空间受限区域的刺激作出反应并传递给下一个神经元。不同神经元的接受场部分重叠，使得它们能平铺整个视野，单个神经元对其接收场中的刺激的响应可以通过卷积运算在数学上近似。CNN 有许多层，层之间的权重在训练过程中学习，在反向传播过程中得到更新，通过堆叠多个层，可以在较深层中学习更高阶的特征，这导致了卷积神经网络在计算机视觉中的主导地位。

2. 卷积神经网络中层的策略

CNN 是由许多不同的层堆叠形成的，通常使用几种不同类型的层：

(1) 卷积层

卷积层是 CNN 的核心构件，该层的参数由一组可学习的过滤器组成，它们具有小的接收场，但是延伸到输入量的整个深度，在前向传播期间，每个滤波器覆盖输入量的宽度和高度进行卷积，计算滤波器和输入量之间的点积，并产生该滤波器的二维激活图。因此，当网络在输入中的某个空间位置检测到特定类型的特征时，网络学习的滤波器将会被激活。如下图 2-1 所示：

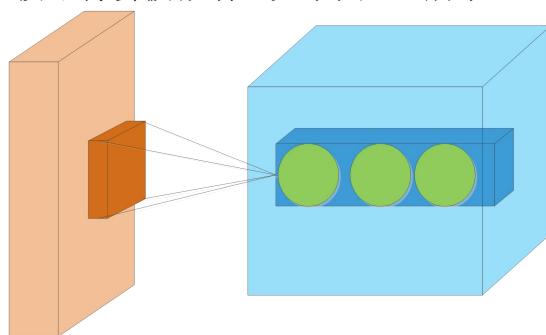


图 2-1 卷积层（绿色为神经元，橙色为接受场）

(2) 池化层

CNN 的另一个重要概念是池化，这是一种非线性下采样的形式。其中最大池化是非线性函数实现的池化层中最常见的。它将输入图像分割成一组非重叠的矩形，并且对于每个子区域，输出最大值。池化层用于逐渐减小输入表示的空间大小，以减少网络中的参数数量和计算量，从而也可以控制过拟合。通常在 CNN 架构中的连续卷积层之间定期插入一个池层。最大池化层如下图 2-2 所示：

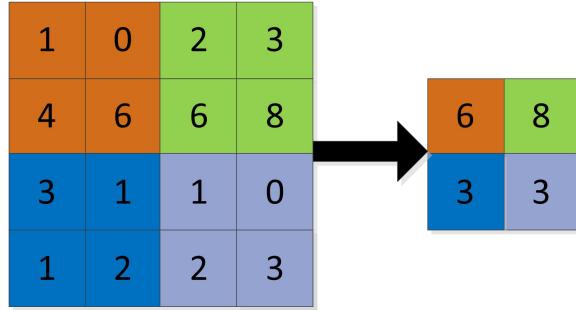


图 2-2 最大池化层 (Max-Pooling)

(3) 激活函数层

激活函数是用来引入非线性因素的，如果 CNN 网络中仅有线性模型，表达能力不够，所以，我们要在网络中加入非线性的激活函数层。它增加了整个网络的非线性特性，而不影响卷积层的接收场，常见的激活函数有 Sigmoid，(式 2-1)、Tanh，(式 2-2)、ReLU，(式 2-3)。ReLU 的使用更为可取，因为它使神经网络训练速度提高了几倍^[18]，而不会对泛化精度产生显着差异。

$$s(x) = \frac{1}{1+e^{-x}} \quad (2-1)$$

$$\tanh x = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad (2-2)$$

$$f(x) = \max(x, 0) \quad (2-3)$$

(4) 全连接层

CNN 中，经过若干卷积层、池化层、激活函数层之后，神经网络中的高级推理是通过全连接层完成的。全连接层常出现在最后几层，用于对前面设计的特征做加权和，在整个 CNN 中起到“分类器”的作用。如图 2-3 所示：

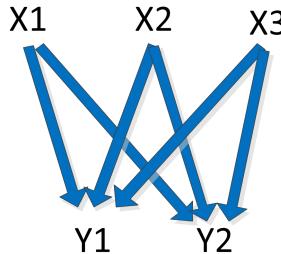


图 2-3 全连接层示意图

$$(x_1 \ x_2 \ x_3)(w_{21} \quad w_{22}) + (b_1 \ b_2) = (y_1 \ y_2) \quad (2-4)$$

$$\begin{matrix} w_{11} & w_{12} \\ w_{31} & w_{32} \end{matrix}$$

式 (2-4) 中， x 是图像矩阵， w 是连接权重， b 是偏差， y 是输出结果。

(5) 损失函数层

损失层指定网络训练如何惩罚预测值和真实标签之间的偏差，通常是网络中的最后一层。可以使用适合于不同任务的各种损失函数，Softmax 损失函数用于预测 K 个互斥类中的单个类别；Sigmoid 交叉熵损失用于预测 [0, 1] 中的 K 个独

立概率值；欧几里德（Euclidean）损失用于回归实值标签。目前，Softmax 分类器是 CNN 模型中最常用的分类器，其函数定义如式（2-5）。

$$P(y=c|x) = \frac{1}{1 + \sum_{k=1}^{c-1} e^{\theta_k^T x}} \quad (2-5)$$

其中 c 表示分类的类别， x 表示属性变量值， θ 为待估计参数，通过似然求解可以得到 θ 值。一般情况下 c 取整数。

（6）典型 CNN 架构图

如图 2-4 所示：

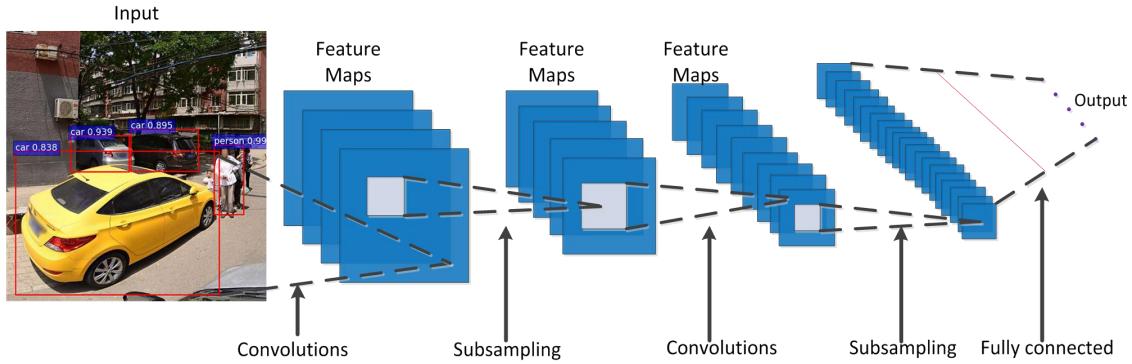


图 2-4 典型 CNN 架构图

（二）ZF 和 VGG 模型简介

1. ZF 网络模型理论简介

随着 AlexNet 在 2012 年的震撼出现^[20]，在 2013 年 ImageNet Large Scale Visual Recognition Competition (ILSVRC) 大赛中被提交的 CNN 模型的数量大幅增加。该竞赛当年获奖者是纽约大学的 Matthew Zeiler 和 Rob Fergus，他们建立的网络名为 ZF 网络，该模型达到了 11.2% 的错误率。ZF 网络对以前的 AlexNet 结构进行了微调，开发了一些非常关键的用于提高网络性能的措施。作者同时花了大量的时间解释了 CNN 背后的含义，并展示了如何正确地展现网络模型的过滤器和权重。作者还讨论了这样一个观点，即 CNN 这种网络模型之所以受到广泛关注是由于大量训练集的可及性，以及因 GPU 的使用而增加的计算能力。ZF 网络模型的原理如图 2-5 所示：

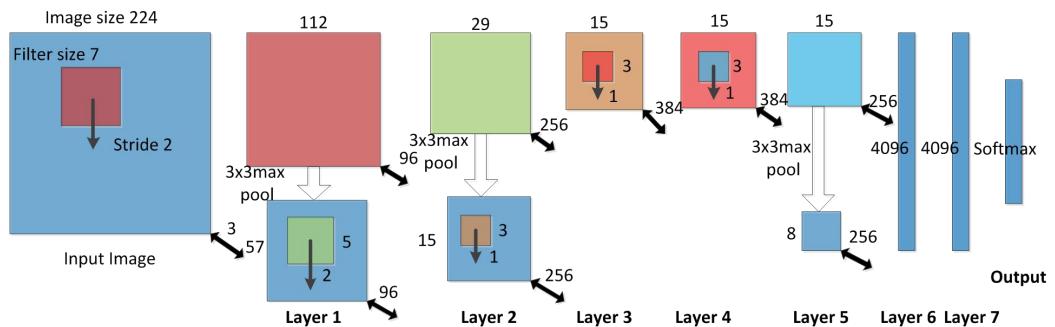


图 2-5 ZF 网络模型图

ZF 网络模型是一个八层的卷积网络模型，把一个 224×224 的 3 通道图像作为输入，然后与 96 个不同权重的第一层滤波器进行卷积操作，每个滤波器大小为 7×7 ，步长为 2。然后经过一个 3×3 最大池化层的欠采样和归一化函数后，得出 96 个不同的 57×57 大小的特征图。类似的操作在第 2,3,4,5 层中重复执行，最后两层是全连接层，它们从上一层的卷积层中提取出 256 个 8×8 的特征图，用于分类。

综上，ZF 网络虽然与 AlexNet 网络相似，但其将第一层的滤波器从 AlexNet 网络的 11×11 改为 7×7 ，这样有助于在第一层卷积后保留大量原始像素信息，ZF 网络也体现了网络的深度对模型非常重要，同时随着网络的增长，使用的过滤器数量也不断增加。

2. VGG 网络模型理论简介

简洁与深度，是 2014 年卷积神经网络领域中的主要模式，牛津大学的 Karen Simonyan 和 Andrew Zisserman 创建了一个 19 层 CNN，命名为 VGG 网络，该模型主要探讨了深度对于网络的重要性，并且获得了很好的结果，在 ILSVRC 2014 大赛上取得了图像目标定位第一，分类第二的好成绩。VGG 网络模型和 ZF 相似，也发展自 AlexNet 网络模型，在 ZF 网络上做了修改，并且深度更深，VGG 的网络模型如图 2-6 所示：

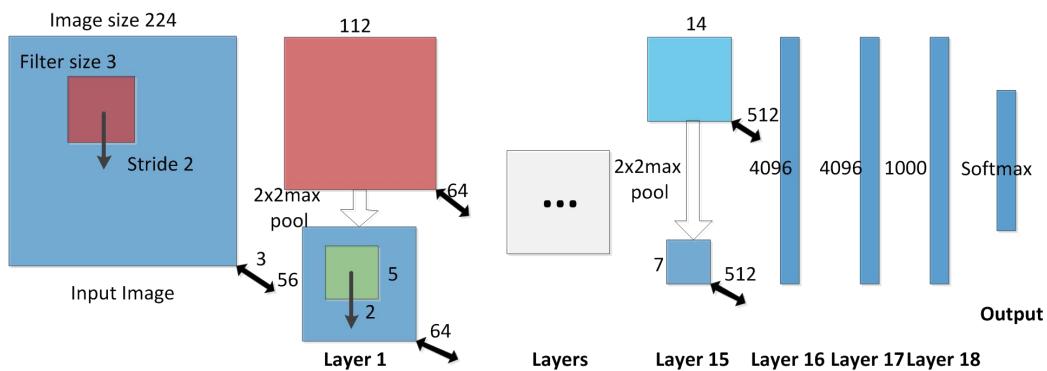


图 2-6 VGG 网络模型图

VGG 和最初的 AlexNet 网络相比主要在两个方面做了修改：第一，在第一个卷积层使用更小的滤波器尺寸和间隔；第二，在整个图片和多尺度上训练和测试图片。VGG 网络仅使用 3×3 尺寸的过滤器，与 AlexNet 的第一层 11×11 过滤器和 ZF Net 的 7×7 过滤器完全不同。作者的推论是，两个 3×3 卷积层的组合具有 5×5 的有效接收场，优点是参数数量的减少。另外，使用两个卷积层，我们可以使用两个 ReLU 层。随着每个层的输入量的空间大小减少，过滤器的数量增加，卷的深度就会增加。并且，在每个最大池化层之后，过滤器数量将翻倍，这实现了缩小空间维度但扩大深度的想法。

VGG 网络是 CNN 网络模型中最有影响力的网络之一，它增强了 CNN 必须具有深层网络层次的概念，保持深层的同时把事情简单化。

(三) Faster R-CNN 模型原理

1. 基于区域的卷积神经网络 R-CNN

基于区域的卷积神经网络（Region-based Convolution Neural Networks）由 Ross B. Girshick 于 2014 年设计和发表，R-CNN 的原理是使用区域提取（region proposal）结合 CNN 代替传统目标检测，先利用 Selective Search^[21]算法在图像中提取 2000 个左右的候选区域；然后将每个候选区域缩放成 227×227 的大小并输入到 CNN，因为 CNN 全连接层的输入需要保证维度固定，将 CNN 的全连接层的输出作为特征；最后将每个候选区域提取到的特征送入 SVM^[22]进行分类，输出类别。图 2-7 为 R-CNN 目标检测流程图：

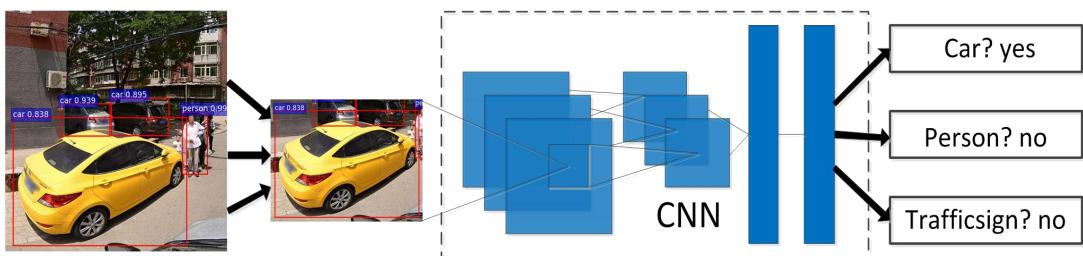


图 2-7 R-CNN 目标检测流程图

R-CNN 在 PASCAL VOC2007 上的检测结果从 34.3% 直接提升到了 66%(mAP)。如此大的提升使我们看到了候选区域提取结合 CNN 的巨大优势。

2. 快速基于区域的卷积神经网络 Fast R-CNN

Fast R-CNN 是对 R-CNN 的改进，检测速度更快，准确率更高。首先，它使用边界框提案方法（如 Selective Search 或 Edge Boxes^[23]）为图像生成边框；接下来，整个图像被传递到 CNN，直到最后一个卷积层。对于每个边界框，将感兴趣区域（Region of Interest, RoI）池化后用于最终层转换特征的向量，然后将该 RoI 向量传递给分类网络和边界框回归网络。分类层是一个全连接层，边界框回归网络是一个 2 层网络，可以对边界框进行微调，使定位更加准确。最后，采用非最大值抑制方法独立地应用于每个类的所有边界框。Fast R-CNN 目标检测流程如图 2-8 所示：

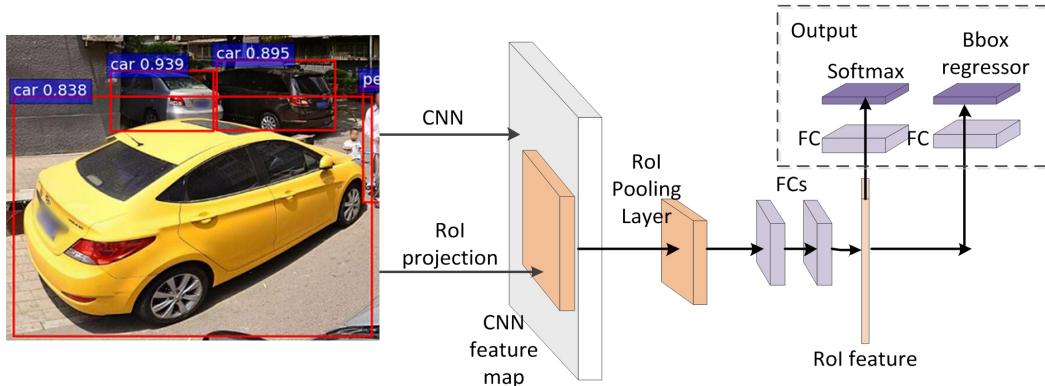


图 2-8 Fast R-CNN 目标检测流程图

对比图 2-8 和图 2-7 可以看出：与 R-CNN 相比，Fast R-CNN 在最后一个卷积层后加了一个 ROI 池化层，并且使用了多任务损失函数 Softmax 作为损失函数。Fast R-CNN 融合了 R-CNN 的精髓，使整个网络的训练和测试变得十分方便。但目标检测时间大多消耗在候选区域上，无法满足实时应用。

3. Faster R-CNN

吸取 R-CNN 和 Fast RCNN 的经验，Ross B. Girshick 于 2016 年提出了新的 Faster R-CNN 网络模型，在结构上，Faster R-CNN 将特征抽取，区域提取，边界框回归和分类都融合在了同一个神经网络中，因此，它的性能有很大的提高，检测速度明显加快。Faster R-CNN 在 PASCAL VOC2007 测试集上 mAP 达到了 73.2%，目标识别的速度可以达到每秒 5 帧。

(1) 卷积层

卷积层包含了卷积，池化，激活函数层三种层，在 Faster R-CNN 卷积层对所有的卷积都做了扩边处理（pad=1，即填充一圈 0），假设原图大小是 $M \times N$ ，导致原图变为 $(M+2) \times (N+2)$ ，再做 3×3 卷积后输出 $M \times N$ 。正是这种设置，导致卷积层并没有改变输入和输出图像矩阵的大小。该过程如图 2-9 所示：

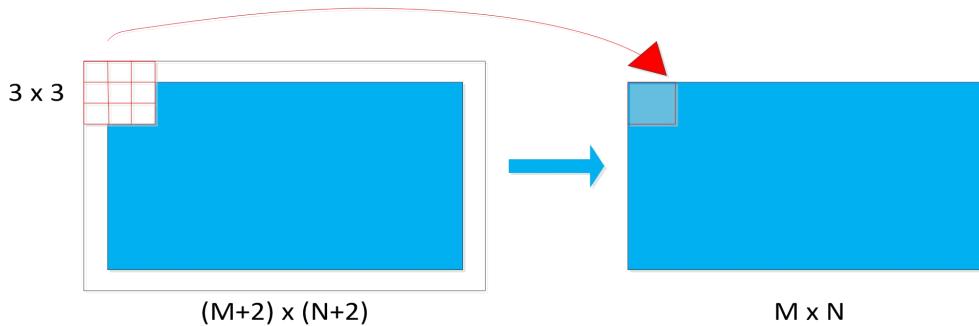


图 2-9 卷积示意图

在整个卷积层中，卷积层和激活层并不改变输入、输出的大小，而池化层因为做了步长为 2 次采样，导致输出矩阵尺寸缩小一半。这种情况下，当输入矩阵为 $M \times N$ ，经过卷积层后变为 $(M/16) \times (N/16)$ ，这样卷积层生成的特征图中就可以和原图对应起来。

(2) RPN 网络

在比较经典的边界框检测中如使用滑动窗口策略非常耗时；或如 R-CNN 使用 Selective Search (SS) 方法生成检测框。Faster R-CNN 直接使用（区域提取网络）RPN 生成边界框，这样就能使得边界框的生成速度得到很大的提高。anchors 是 RPN 生成的 9 个矩形，长宽比为大约为 [1:1, 1:2, 2:1] 三种，anchors 机制体现了目标检测中的多尺度方法。

图 2-10 展示了 RPN 网络的具体结构：

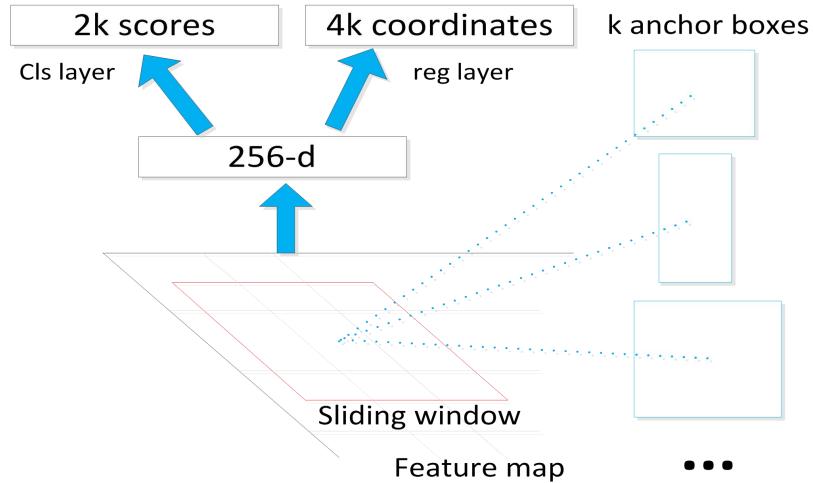


图 2-10 RPN 网络结构

上图使用的是 ZF 模型，其中最后的卷积层输出是 256，对应生成 256 张特征图，所以相当于 Feature map 每个点都是 256-d。在卷积层之后，做了 RPN 阶段的 3×3 卷积且输出为 256，相当于每个点使用了周围 3×3 的空间信息，同时 256-d 不变。假设每个点一共有 k 个 anchor，每个 anchor 要分前景和背景，所以类别为 2k 个；因为每个 anchor 都有 $[x, y, w, h]$ 4 个偏移量，所以边界框回归为 4k 个。对 2k 个类别 anchors 用 softmax 判定背景与前景，相当于初步提取了检测目标候选区域框，接下来需要对边界框进行更精确的定位，如图 2-11 所示：

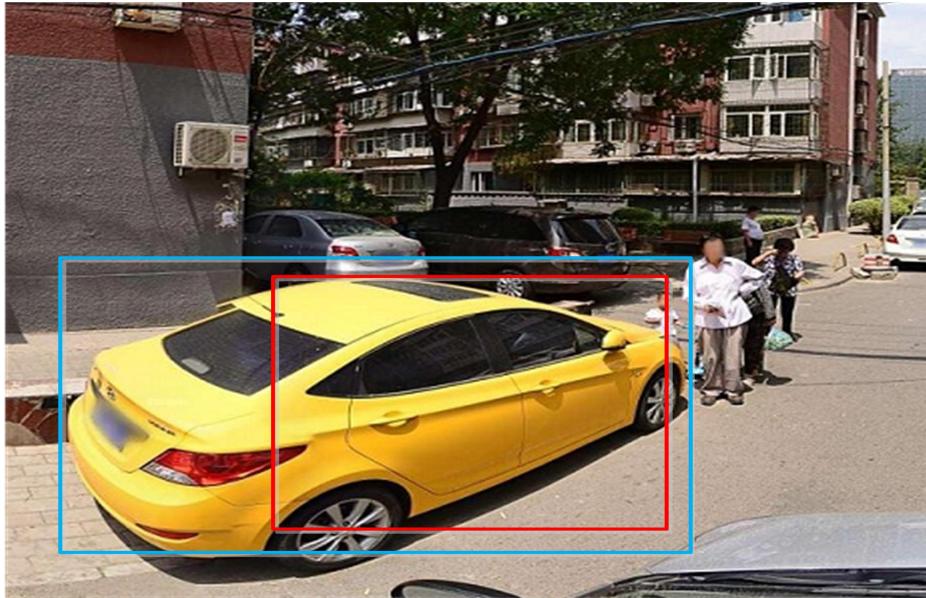


图 2-11 预测框（红色）与真实框（蓝色）

如上图，由于红色的框定位不准，这张图相当于没有正确的检测出汽车，Faster R-CNN 采用如下策略进行边框精确定位：

窗口一般使用四维向量 (x, y, w, h) 表示，对于图 2-12，红色的框 A 代表原始的 RPN 预测边界框，绿色边界框 G 代表图像中识别出目标的真实边界框，所以，可以看出，我们希望找到映射关系，输入原始的 A 经过映射后得到一个跟真实边界框 G 更接近的边界框 G_1 ，即给定 $A = (A_x, A_y, A_w, A_h)$ ，寻找一种映射 f ，使得

$f(A) = (G1_x, G1_y, G1_w, G1_h)$, 而 $(G1_x, G1_y, G1_w, G1_h) \approx (G_x, G_y, G_w, G_h)$ 。

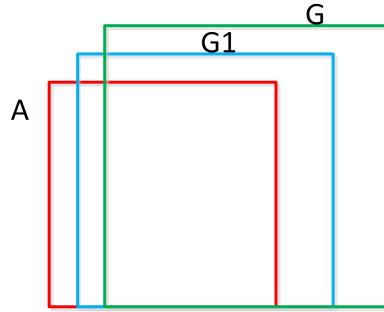


图 2-12

先做平移:

$$G1_x = A_w \cdot d_x(A) + A_x \quad (2-6)$$

$$G1_y = A_h \cdot d_y(A) + A_y \quad (2-7)$$

再做缩放:

$$G1_w = A_w \cdot \exp(d_w(A)) \quad (2-8)$$

$$G1_h = A_h \cdot \exp(d_h(A)) \quad (2-9)$$

所以需要学习 $d_x(A), d_y(A), d_w(A), d_h(A)$ 这四个变换，并且认为是线性变换，用线性回归建模对边界框进行微调，平移量 (t_x, t_y) 和尺度因子 (t_w, t_h) 如下：

$$t_x = \frac{(G_x - A_x)}{A_w} \quad (2-10)$$

$$t_y = \frac{(G_y - A_y)}{A_h} \quad (2-11)$$

$$t_w = \frac{\ln(G_y - A_y)}{A_w} \quad (2-12)$$

$$t_h = \frac{\ln(G_y - A_y)}{A_h} \quad (2-13)$$

接下来用线性回归获得 $d_x(A), d_y(A), d_w(A), d_h(A)$ ，目标函数可以表示为：

$$d_*(A) = w^T \cdot \phi(A) \quad (2-14)$$

其中 $\phi(A)$ 是对应预测框的特征图组成的特征向量， w 是需要学习的参数， $d(A)$ 是预测值 (* 表示 x, y, w, h)，为了最小化预测值和真实值之差，得到损失函数：

$$Loss = \sum_i^N (t_*^i - \bar{w} \cdot \phi(A^i))^2 \quad (2-15)$$

通过这种方式可以从前景回归出候选区域。

(3) RoI 池化

RoI 池化是解决输入输出矩阵尺度问题的，对于经典的 CNN(如 ZF, VGG)，一个训练好的网络，必须输入一个固定大小的图像矩阵，而且，这个网络模型的输出也同样是固定大小的矩阵。所以，若是输入图像矩阵的大小不定的话，问题就会变复杂。有两种解决方法：一种方法是从图像中裁剪出一部分传入网络模型，另一种方法是将图像尺度化成该网络需求的尺寸。但是，这两种方法都不友好，方法一裁剪后图像的完整性被损坏，方法二破坏了图像原始形状大小。在 RoI 池化层，每个候选区域在水平和竖直方向上都被分为 N 份，接着对每一份区域都进行最大池化操作，这样即使大小不同的区域，输出结果都是 $N \times N$ 大小，就实现了固定大小输出。如图 2-13 所示为 N 为 7 时的示例：

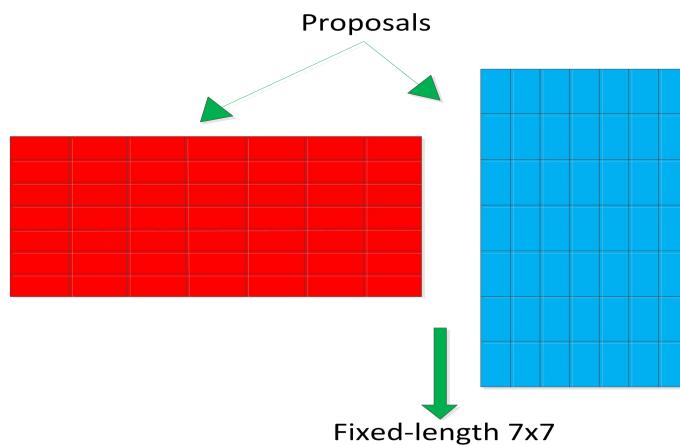


图 2-13 区域提案示意图 (proposals)

(4) 分类

分类部分用卷积层，激活层，池化层等处理后获得的特征图，利用全连接层与 softmax 损失函数计算每个候选区域属于哪个类别（如人，车，交通标志等），输出每一个类别的概率向量；然后利用边界框回归函数获得每个候选区域的位置偏移量，使得目标定位更加精确。

三、简易分布式爬虫系统设计与数据集制作

本章介绍简易分布式爬虫系统的设计框架和从百度街景中爬取的数据，并制作数据集用于训练和测试。

(一) 爬虫系统框架

1. 爬虫框架 Scrapy 简介

Scrapy 是爬虫框架里十分出众的一个，主要因为它不但提供了许多可直接使用的基础模块，还表现在它可以让开发者尽情地发挥自定义功能。Scrapy 这个爬虫框架主要是为了爬取网站数据，并从中提取用户感兴趣的结构化数据，Scrapy 的应用比较广泛，可以应用在获取网站 API 接口返回的结构化数据或针对某一个网站进行定制化网络爬虫设计。之所以选 Scrapy 来做爬虫系统，是因为它自定义程度高，非常适合针对某一网站进行定制化设计，比 PySpider 更底层一些，适合学习研究，拿来研究分布式和多线程等等是最合适不过的。

2. 分布式爬虫系统设计架构

深度学习技术之所以火起来主要有三大原因：GPU 所带来的的计算能力的提升、更深层的网络模型、大数据。其中数据是深度学习案例能成功所必备的一环。所以为了最后训练的深度学习模型有更高的泛化能力和准确度，优质并且大量的训练和测试数据是不可获取的，用网络爬虫来获取百度街景图片是最廉价，也是最符合本课题做道路交通目标识别的方式。为此设计了一个简易分布式爬虫系统，系统框图如图 3-1 所示：

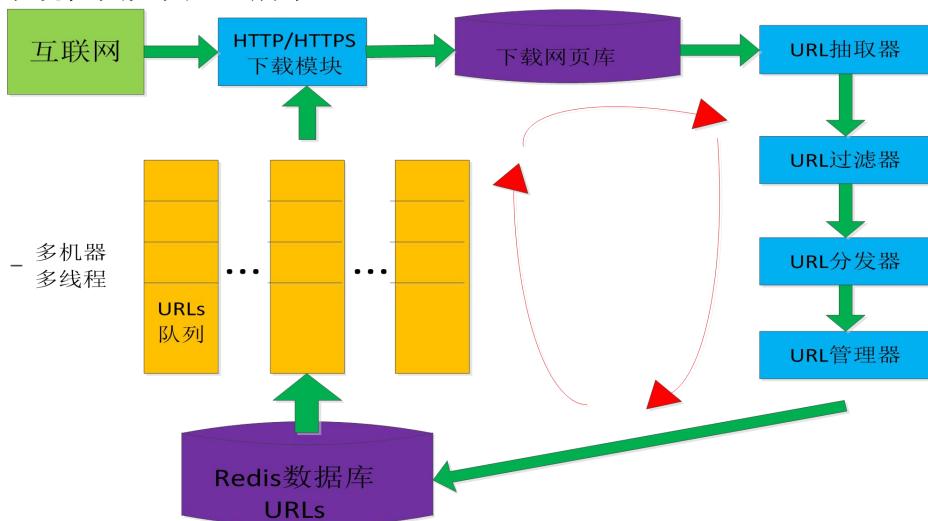
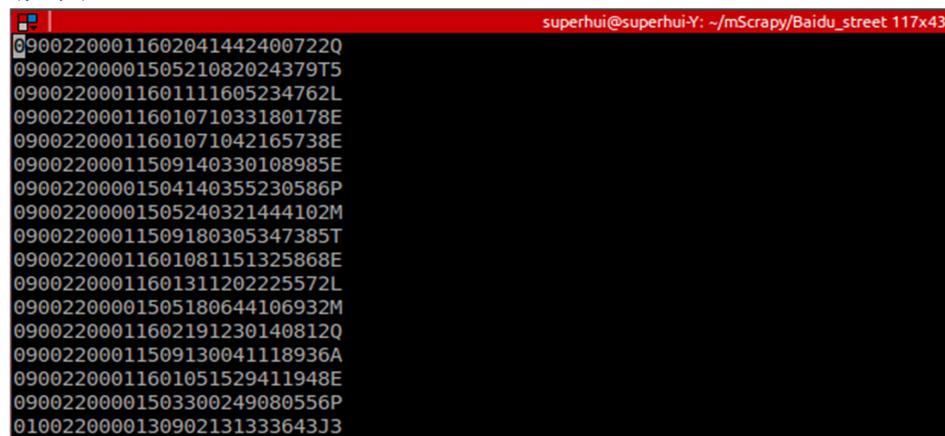


图 3-1 分布式爬虫系统架构

3. 爬虫运行结果

一个好的网络爬虫系统应该是一个可循环的过程，从图 3-1 可以看出，本文设计的爬虫系统做到了这一点，这是在初步分析了百度街景网站的协议后所设计的定制化的爬虫。首先爬虫系统访问待爬网站，即百度街景，因为百度街景每个地方的图片对应一个 ID，所以第一步是爬取带有 ID 的百度街景网页，保存到本地数据库，然后用 URL 处理模块对下载的网页进行进一步的分析处理，剥离出代表图片地址的 URL，以入栈的形式存到 Redis 数据库。Redis 数据库配置在服务器机器上，在 3 台机器上部署爬虫客户端，采用多线程编程从服务器的 Redis 数据库中弹栈出 URLs，通过下载模块进行图片下载，同时利用每张图片的 ID 去获取更多的含有 ID 的网页，不断循环下去。爬虫运行截图如图 3-2，运行结果截图如图 3-3：



```
superhui@superhui-Y: ~/mScrapy/Baidu_street 117x43
09002200011602041442400722Q
090022000150521082024379T5
09002200011601111605234762L
09002200011601071033180178E
09002200011601071042165738E
09002200011509140330108985E
0900220001504140355230586P
0900220001505240321444102M
09002200011509180305347385T
09002200011601081151325868E
09002200011601311202225572L
0900220001505180644106932M
09002200011602191230140812Q
09002200011509130041118936A
09002200011601051529411948E
0900220001503300249080556P
010022000130902131333643J3
```

图 3-2 爬虫运行图

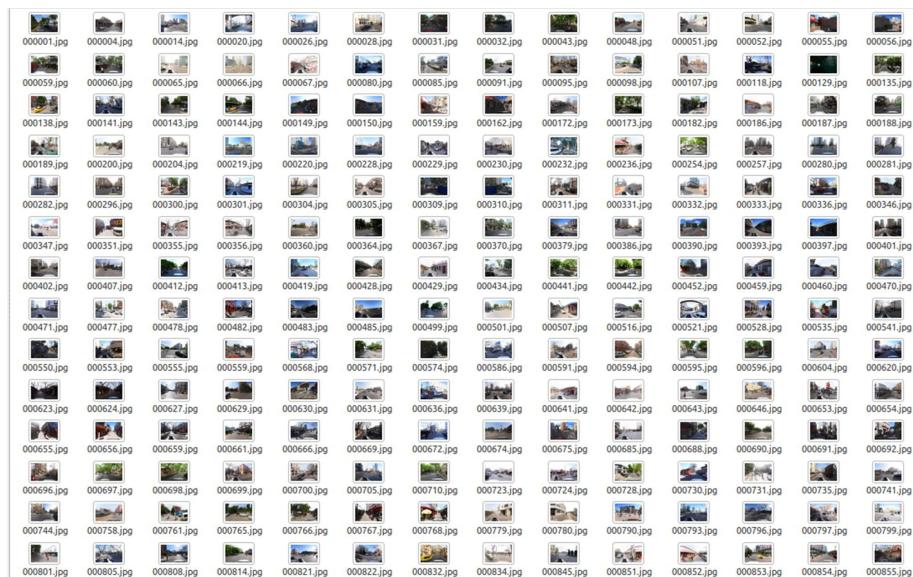


图 3-3 爬虫运行结果图

最后共爬取大约 60 万张北京市的百度街景图片，其中用 6 张合成一张大图，得到大约 10 张图片。

(二) 数据集制作

1. 数据集标注

收集完百度街景图片后，需要对图片进行标注，目标信息分为 3 大类：车、人、交通标志，其中交通标志类包括红绿灯、交通路牌等，属于一个大类。对图片标注时，主要标注图片中目标的类别信息和目标的边界框信息，用于训练对图片中目标进行识别和定位的模型。国外的交通路况信息和我国可能有不同，所以尝试用本国的数据来训练和测试。

用开源工具 LabelImg 对图片进行标注，部分标注结果如图 3-4 所示：

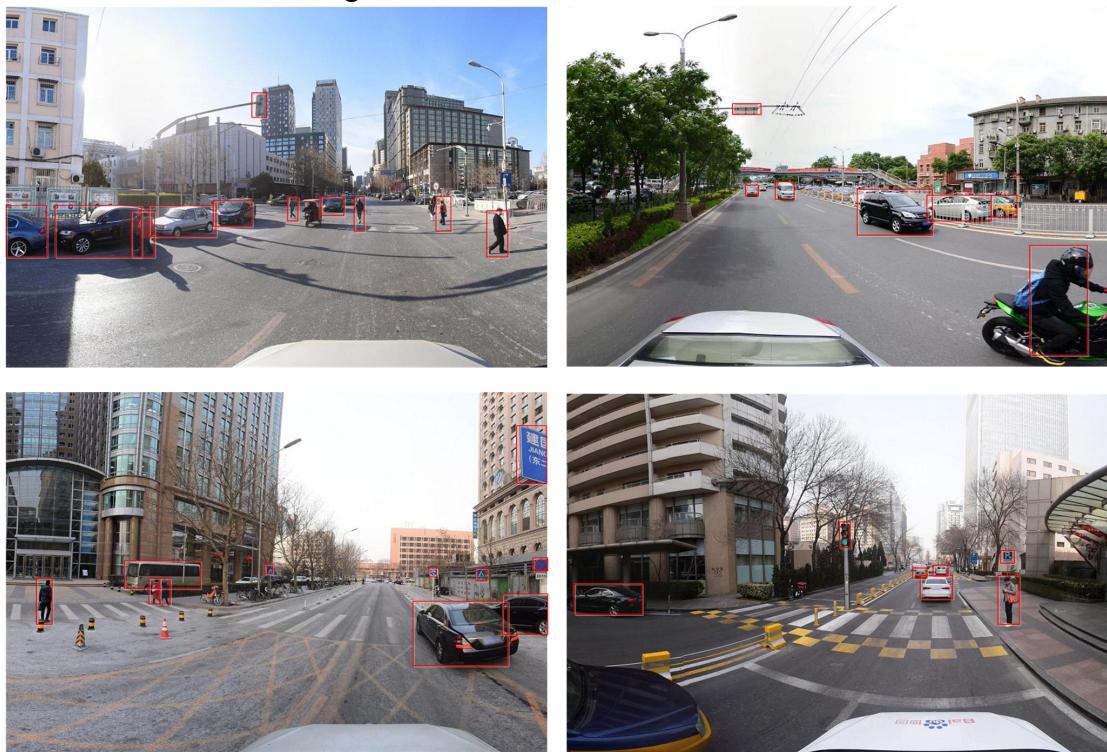


图 3-4 部分标注结果

用 LabelImg 将标注结果生成 XML 文件，每张图片对应一个 XML 文件，这些 XML 文件作为图片中目标的类别和边界框监督信息用于训练和测试过程，深度学习框架 Caffe 提供了读取 XML 标注信息的接口，如图 3-5 数据集标注信息示例：

```
superhui@superhui-Y: ~/caffe/Faster_rcnn/new/py-faster-rcnn/lib/datasets/VOCdevkit/VOC2007/Annotations/000001 86x43
<annotation>
    <filename>000001/000578.jpg</filename>
    <source>
        <database>My Database</database>
        <image>flickr</image>
        <flickrid>NULL</flickrid>
    </source>
    <owner>
        <name>yuchaozhi</name>
    </owner>
    <size>
        <width>1280</width>
        <height>720</height>
        <depth>3</depth>
    </size>
    <object>
        <name>trafficsign</name>
        <truncated>0</truncated>
        <difficult>0</difficult>
        <bndbox>
            <xmin>423</xmin>
            <ymin>221</ymin>
            <xmax>513</xmax>
            <ymax>278</ymax>
        </bndbox>
    </object>
</annotation>
~
```

图 3-5 XML 标注信息

2. 训练测试数据集制作

数据标注是一个很费时间的工作，本文近 10 万张图片的标注耗时大约 7 天。对图片数据进行标注之后，需要制作用于模型训练的训练集和用于模型测试的测试集。根据前人实验经验，从所有标注好的数据集中随机抽取大约 75000 张用于训练集，剩余的大约 25000 张用于测试集，训练集和测试集的比例为 3:1。

四、网络模型的搭建、训练与测试

本章基于 ZF 网络和 VGG 网络，用 Python 进行模型搭建，在 GPU 平台上进行模型训练和测试，并对结果进行分析比较。

(一) 网络模型搭建

1. 模型训练与测试平台

本课题是基于 Caffe 这一深度学习框架进行研究，Caffe 是一个高效清晰的深度学习框架，由毕业于加州大学伯克利分校的贾杨清博士发布，Caffe 相比于其他深度学习框架（TensorFlow，Mxnet，Theano 等）有许多优势：

- (1) Caffe 的模型和优化设置都是以配置文本的形式，方便上手。
- (2) 速度快，当它结合 Nvidia 的 GPU 和 GPU 加速组件 cuDNN，能够训练更深的模型和海量数据。
- (3) 模块化方便自定义，Caffe 提供了各层类型，方便用户自定义。
- (4) 多语言支持，支持 Python, C++, Matlab。

Caffe 主要包括四大组件：Blob、Net、Layer、Solver，其中 Blob 是 Caffe 的数据传递形式；Layer 是 CNN 中层的定义；Net 是由若干 Layer 组成的网络模型框架；Solver 提供了 CNN 模型训练策略，为 Net 提供一些训练参数配置等。图 4-1 是 Caffe 中 CNN 模型的训练流程图：

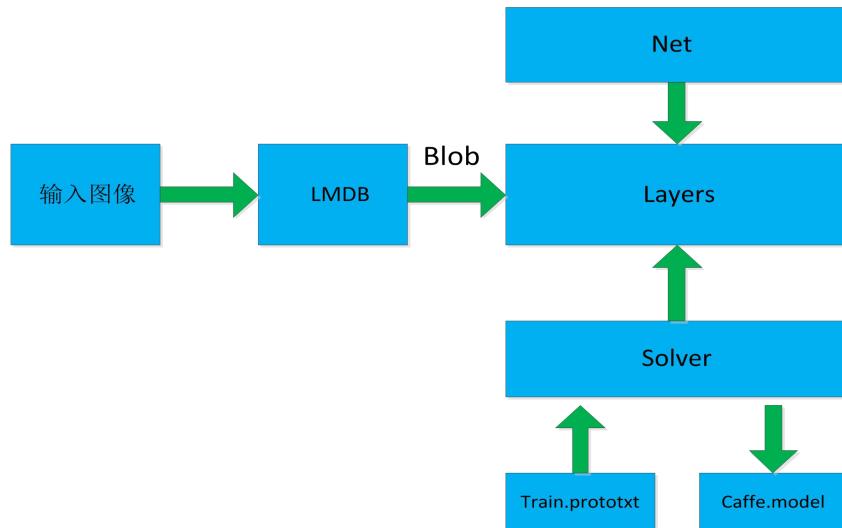


图 4-1 CNN 模型训练流程

输入图像经过预处理后生成 Caffe 训练需要的数据格式 LMDB，然后 Caffe 以 Blob 形式读取数据送入网络训练，Net 完成网络初始化等工作，Solver 控制训练的学习率和速度等指标，在训练结束后生成一个训练模型 Caffe.model。

2. ZF 网络模型搭建

用 Python 语言基于 Caffe 框架搭建基于 ZF 网络的 Faster R-CNN 模型的训练模型 `zf_faster_rcnn_train.prototxt`, `zf_faster_rcnn_test.prototxt`, 和 `zf_faster_rcnn_solver.prototxt`。用开源工具 Netscop 对简化后的训练模型进行了作图, 图 4-2 为基于 ZF 网络的训练模型, 图 4-3 为模型定义文件:

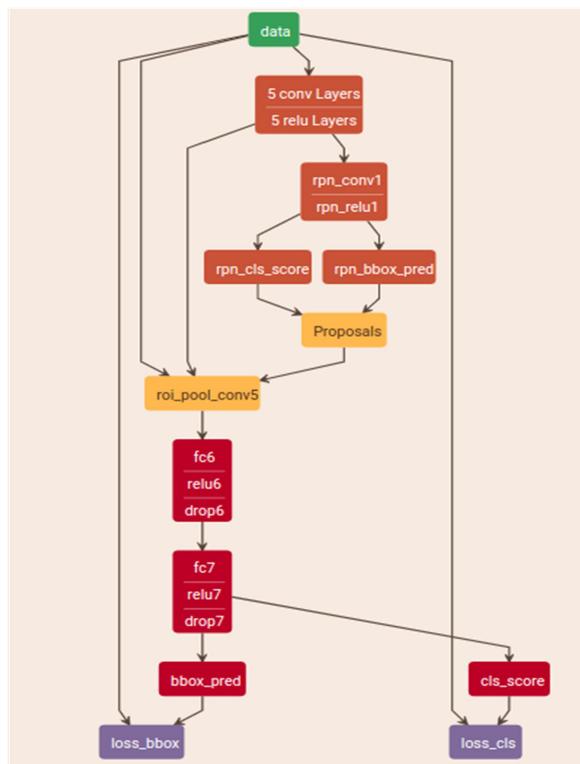


图 4-2 基于 ZF 网络的 Faster R-CNN 训练模型

```
superhui@superhui-Y: ~/caffe/Faster_rcnn/new/py-faster-rcnn
Name: "ZF"
layer {
    name: 'data'
    type: 'Python'
    top: 'data'
    top: 'rois'
    top: 'labels'
    top: 'bbox_targets'
    top: 'bbox_inside_weights'
    top: 'bbox_outside_weights'
    python_param {
        module: 'roi_data_layer.layer'
        layer: 'RoIDataLayer'
        param_str: "'num_classes': 4"
    }
}
=====
layer {
    name: "conv1"
    type: "Convolution"
    bottom: "data"
    top: "conv1"
    param { lr_mult: 1.0 }
    param { lr_mult: 2.0 }
    convolution_param {
        num_output: 96
        kernel_size: 7
        pad: 3
        stride: 2
    }
}
layer {
    name: "relu1"
    type: "ReLU"
    bottom: "conv1"
```

图 4-3 模型定义文件示例

3. VGG 网络模型搭建

与 ZF 网络模型的搭建过程类似, 首先用 Python 语言基于 Caffe 框架搭建基于 VGG 网络的 Faster R-CNN 模型的训练模型 `vgg_faster_rcnn_train.prototxt`, 测试模型 `vgg_faster_rcnn_test.prototxt`, 和 `vgg_faster_rcnn_solver.prototxt`。用开源工具 Netscop 对简化后的训练模型进行了作图, 图 4-4 为基于 VGG 网络的训练模型, 图 4-5 为模型定义文件:

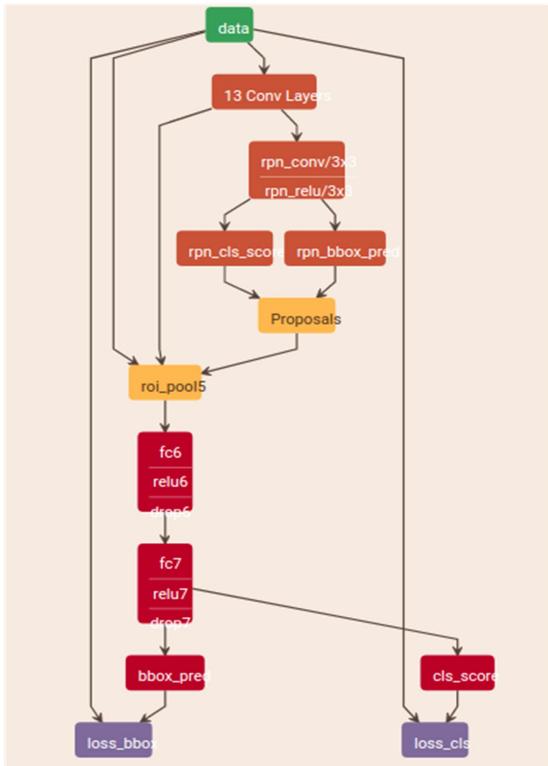


图 4-4 基于 VGG 网络的 Faster R-CNN 训练模型

```

superhui@superhui: ~/caffe/Faster_rcnn/new/py-faster-rcnn/models/p
name: "VGG"
layer {
    name: 'data'
    type: 'Python'
    top: 'data'
    top: 'rois'
    top: 'labels'
    top: 'bbox_targets'
    top: 'bbox_inside_weights'
    top: 'bbox_outside_weights'
    python_param {
        module: 'roi_data_layer.layer'
        layer: 'RoIDataLayer'
        param_str: "'num_classes': 4"
    }
}
layer {
    name: "conv1_1"
    type: "Convolution"
    bottom: "data"
    top: "conv1_1"
    param { lr_mult: 0 decay_mult: 0 }
    param { lr_mult: 0 decay_mult: 0 }
    convolution_param {
        num_output: 64
        pad: 1
        kernel_size: 3
    }
}
layer {
    name: "relu1_1"
    type: "ReLU"
    bottom: "conv1_1"
    top: "conv1_1"
}

```

图 4-5 模型定义文件示例

(二) 模型训练

1. 模型训练环境

- (1) 操作系统: Linux (Ubuntu 16.04)
- (2) CPU: 8 核
- (3) GPU: Nvidia GTX 1060 (3G 显存)
- (4) 内存: 16G
- (5) Nvidia 驱动: CUDA8.0+cuDNN5.1

2. ZF 网络模型训练

在准备好训练数据和 Caffe 训练用的模型配置文件后，就可以在 GPU 平台上进行训练任务，图 4-6 所示为训练时输出的日志文件：

```

superhui@superhui-Y:~/caffe/Faster_rcnn/1060/ZF_ych/30000ZF 120x41
I0427 18:50:11.628878 2713 net.cpp:816] Ignoring source layer drop
I0427 18:50:11.628880 2713 net.cpp:816] Ignoring source layer fc8
I0427 18:50:11.628881 2713 net.cpp:816] Ignoring source layer prob
Solving...
I0427 18:50:11.815829 2713 solver.cpp:229] Iteration 0, loss = 1.83439
I0427 18:50:11.815871 2713 solver.cpp:245] Train net output #0: rpn_cls_loss = 0.787563 (* 1 = 0.787563 loss)
I0427 18:50:11.815876 2713 solver.cpp:245] Train net output #1: rpn_loss_bbox = 1.04683 (* 1 = 1.04683 loss)
I0427 18:50:11.815881 2713 sgd solver.cpp:106] Iteration 0, lr = 0.001
I0427 18:50:14.374547 2713 solver.cpp:229] Iteration 20, loss = 0.254013
I0427 18:50:14.374578 2713 solver.cpp:245] Train net output #0: rpn_cls_loss = 0.207482 (* 1 = 0.207482 loss)
I0427 18:50:14.374583 2713 solver.cpp:245] Train net output #1: rpn_loss_bbox = 0.0465313 (* 1 = 0.0465313 loss)
I0427 18:50:14.374588 2713 sgd solver.cpp:106] Iteration 20, lr = 0.001
I0427 18:50:16.891893 2713 solver.cpp:229] Iteration 40, loss = 0.50134
I0427 18:50:16.891924 2713 solver.cpp:245] Train net output #0: rpn_cls_loss = 0.296383 (* 1 = 0.296383 loss)
I0427 18:50:16.891930 2713 solver.cpp:245] Train net output #1: rpn_loss_bbox = 0.204957 (* 1 = 0.204957 loss)
I0427 18:50:16.891934 2713 sgd solver.cpp:106] Iteration 40, lr = 0.001
I0427 18:50:19.426743 2713 solver.cpp:229] Iteration 60, loss = 0.402798
I0427 18:50:19.426774 2713 solver.cpp:245] Train net output #0: rpn_cls_loss = 0.238741 (* 1 = 0.238741 loss)
I0427 18:50:19.426779 2713 solver.cpp:245] Train net output #1: rpn_loss_bbox = 0.164057 (* 1 = 0.164057 loss)
I0427 18:50:21.955282 2713 sgd solver.cpp:106] Iteration 60, lr = 0.001
I0427 18:50:21.955286 2713 solver.cpp:229] Iteration 80, loss = 0.360691
I0427 18:50:21.955278 2713 solver.cpp:245] Train net output #0: rpn_cls_loss = 0.218132 (* 1 = 0.218132 loss)
I0427 18:50:21.955284 2713 solver.cpp:245] Train net output #1: rpn_loss_bbox = 0.142559 (* 1 = 0.142559 loss)
I0427 18:50:21.955288 2713 sgd solver.cpp:106] Iteration 80, lr = 0.001
I0427 18:50:24.483348 2713 solver.cpp:229] Iteration 100, loss = 0.609349
I0427 18:50:24.483379 2713 solver.cpp:245] Train net output #0: rpn_cls_loss = 0.326145 (* 1 = 0.326145 loss)
I0427 18:50:24.483384 2713 solver.cpp:245] Train net output #1: rpn_loss_bbox = 0.283204 (* 1 = 0.283204 loss)
I0427 18:50:24.483389 2713 sgd solver.cpp:106] Iteration 100, lr = 0.001
I0427 18:50:27.015445 2713 solver.cpp:229] Iteration 120, loss = 0.250849
I0427 18:50:27.015476 2713 solver.cpp:245] Train net output #0: rpn_cls_loss = 0.112665 (* 1 = 0.112665 loss)
I0427 18:50:27.015482 2713 solver.cpp:245] Train net output #1: rpn_loss_bbox = 0.138184 (* 1 = 0.138184 loss)
I0427 18:50:27.015486 2713 sgd solver.cpp:106] Iteration 120, lr = 0.001
I0427 18:50:29.534883 2713 solver.cpp:229] Iteration 140, loss = 0.520874
I0427 18:50:29.534915 2713 solver.cpp:245] Train net output #0: rpn_cls_loss = 0.477579 (* 1 = 0.477579 loss)
I0427 18:50:29.534921 2713 solver.cpp:245] Train net output #1: rpn_loss_bbox = 0.0432949 (* 1 = 0.0432949 loss)
I0427 18:50:29.534925 2713 sgd solver.cpp:106] Iteration 140, lr = 0.001
I0427 18:50:32.037425 2713 solver.cpp:229] Iteration 160, loss = 0.340058
I0427 18:50:32.037473 2713 solver.cpp:245] Train net output #0: rpn_cls_loss = 0.233287 (* 1 = 0.233287 loss)
I0427 18:50:32.037479 2713 solver.cpp:245] Train net output #1: rpn_loss_bbox = 0.10677 (* 1 = 0.10677 loss)
I0427 18:50:32.037484 2713 sgd solver.cpp:106] Iteration 160, lr = 0.001

```

图 4-6 ZF 模型训练日志

在迭代到 30000 次时 loss 不再下降，停止训练，共历时大约 24 个小时完成模型训练，训练完成后根据训练日志作图，图 4-7 为分类 loss 图：

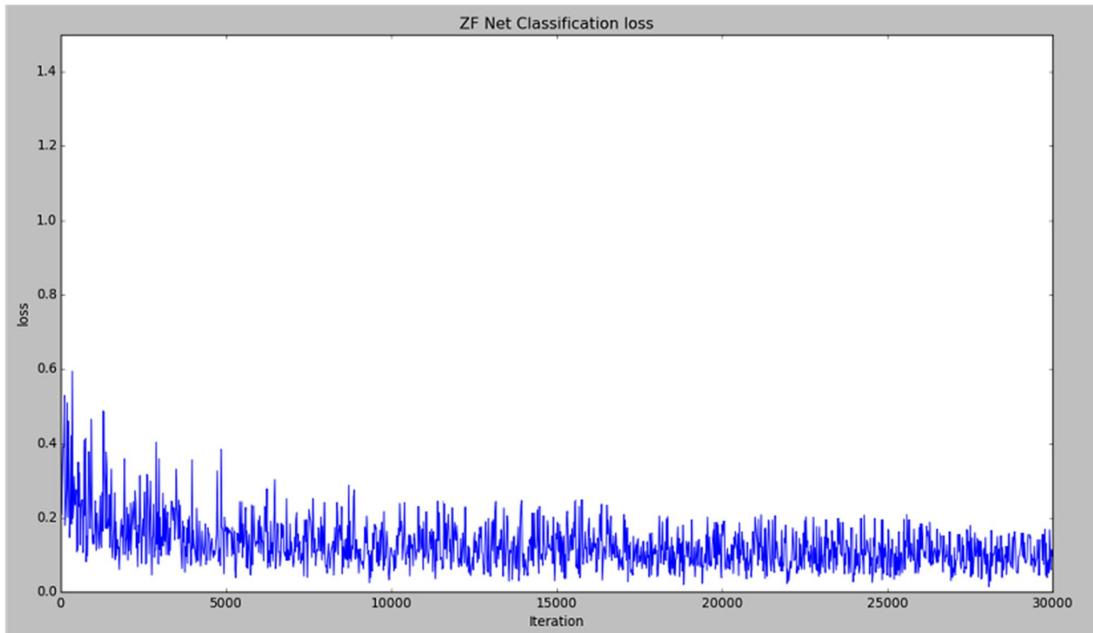


图 4-7 ZF 模型分类 loss 图

图 4-8 为 ZF 模型预测用于定位目标的边界框 loss 图：

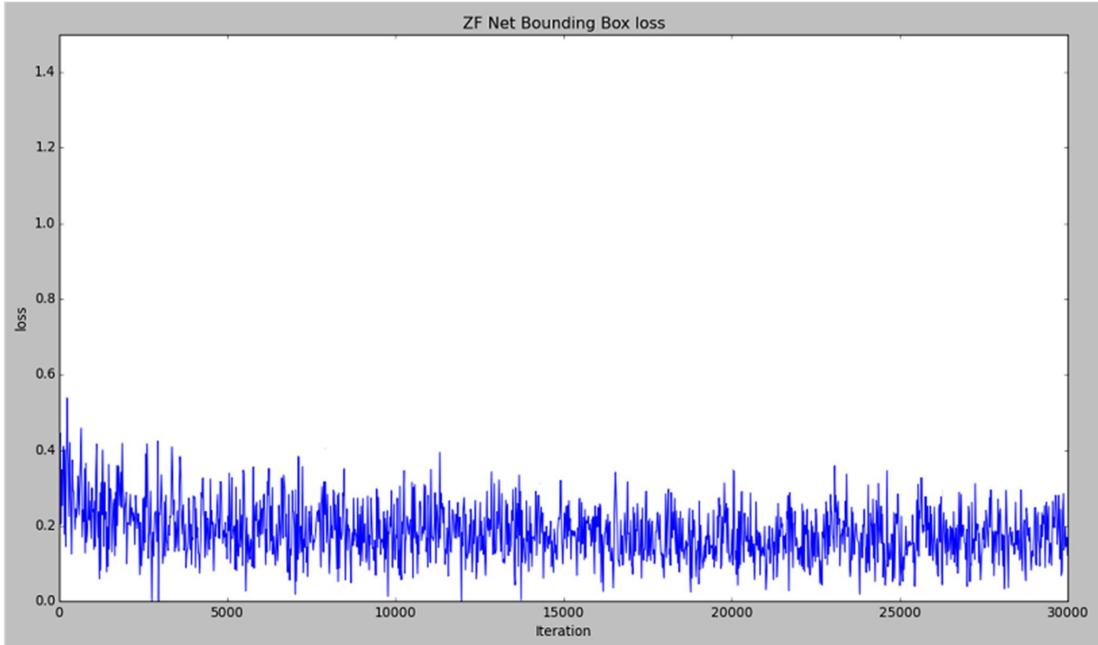


图 4-8 ZF 模型边界框 loss 图

3. VGG 网络模型训练

与训练 ZF 模型类似，在准备好训练数据和模型配置文件后，在 GPU 平台上开始进行训练任务，图 4-9 为训练时输出的日志文件：

```
superhui@superhui-Y:~/caffe/Faster_rcnn/1060/VGG16_fc/fc12 120x41
I0226 10:27:01.570170 21209 net.cpp:816] Ignoring source layer prob
Solving...
I0226 10:27:02.096491 21209 solver.cpp:229] Iteration 0, loss = 0.85383
I0226 10:27:02.096549 21209 solver.cpp:245] Train net output #0: rpn_cls_loss = 0.803254 (* 1 = 0.803254 loss)
I0226 10:27:02.096559 21209 solver.cpp:245] Train net output #1: rpn_loss_bbox = 0.0505763 (* 1 = 0.0505763 loss)
I0226 10:27:02.096566 21209 sgd_solver.cpp:106] Iteration 0, lr = 0.001
I0226 10:27:10.941480 21209 solver.cpp:229] Iteration 20, loss = 0.739145
I0226 10:27:10.941545 21209 solver.cpp:245] Train net output #0: rpn_cls_loss = 0.274291 (* 1 = 0.274291 loss)
I0226 10:27:10.941553 21209 solver.cpp:245] Train net output #1: rpn_loss_bbox = 0.464854 (* 1 = 0.464854 loss)
I0226 10:27:10.941561 21209 sgd_solver.cpp:106] Iteration 20, lr = 0.001
I0226 10:27:20.179414 21209 solver.cpp:229] Iteration 40, loss = 0.445814
I0226 10:27:20.179468 21209 solver.cpp:245] Train net output #0: rpn_cls_loss = 0.326512 (* 1 = 0.326512 loss)
I0226 10:27:20.179476 21209 solver.cpp:245] Train net output #1: rpn_loss_bbox = 0.119303 (* 1 = 0.119303 loss)
I0226 10:27:20.179482 21209 sgd_solver.cpp:106] Iteration 40, lr = 0.001
I0226 10:27:27.775333 21209 solver.cpp:229] Iteration 60, loss = 0.484145
I0226 10:27:27.775390 21209 solver.cpp:245] Train net output #0: rpn_cls_loss = 0.378658 (* 1 = 0.378658 loss)
I0226 10:27:27.775398 21209 solver.cpp:245] Train net output #1: rpn_loss_bbox = 0.105486 (* 1 = 0.105486 loss)
I0226 10:27:27.775404 21209 sgd_solver.cpp:106] Iteration 60, lr = 0.001
I0226 10:27:37.549383 21209 solver.cpp:229] Iteration 80, loss = 0.242671
I0226 10:27:37.549439 21209 solver.cpp:245] Train net output #0: rpn_cls_loss = 0.190022 (* 1 = 0.190022 loss)
I0226 10:27:37.549448 21209 solver.cpp:245] Train net output #1: rpn_loss_bbox = 0.0526493 (* 1 = 0.0526493 loss)
I0226 10:27:37.549473 21209 sgd_solver.cpp:106] Iteration 80, lr = 0.001
I0226 10:27:47.299036 21209 solver.cpp:229] Iteration 100, loss = 0.634716
I0226 10:27:47.299091 21209 solver.cpp:245] Train net output #0: rpn_cls_loss = 0.277841 (* 1 = 0.277841 loss)
I0226 10:27:47.299099 21209 solver.cpp:245] Train net output #1: rpn_loss_bbox = 0.356875 (* 1 = 0.356875 loss)
I0226 10:27:47.299105 21209 sgd_solver.cpp:106] Iteration 100, lr = 0.001
I0226 10:27:56.993199 21209 solver.cpp:229] Iteration 120, loss = 1.2828
I0226 10:27:56.993257 21209 solver.cpp:245] Train net output #0: rpn_cls_loss = 0.321526 (* 1 = 0.321526 loss)
I0226 10:27:56.993265 21209 solver.cpp:245] Train net output #1: rpn_loss_bbox = 0.961271 (* 1 = 0.961271 loss)
I0226 10:27:56.993271 21209 sgd_solver.cpp:106] Iteration 120, lr = 0.001
I0226 10:28:05.019271 21209 solver.cpp:229] Iteration 140, loss = 0.745484
I0226 10:28:05.019326 21209 solver.cpp:245] Train net output #0: rpn_cls_loss = 0.417335 (* 1 = 0.417335 loss)
I0226 10:28:05.019333 21209 solver.cpp:245] Train net output #1: rpn_loss_bbox = 0.328149 (* 1 = 0.328149 loss)
I0226 10:28:05.019340 21209 sgd_solver.cpp:106] Iteration 140, lr = 0.001
I0226 10:28:14.495766 21209 solver.cpp:229] Iteration 160, loss = 0.389684
I0226 10:28:14.495827 21209 solver.cpp:245] Train net output #0: rpn_cls_loss = 0.249423 (* 1 = 0.249423 loss)
I0226 10:28:14.495836 21209 solver.cpp:245] Train net output #1: rpn_loss_bbox = 0.14026 (* 1 = 0.14026 loss)
I0226 10:28:14.495843 21209 sgd_solver.cpp:106] Iteration 160, lr = 0.001
I0226 10:28:23.971082 21209 solver.cpp:229] Iteration 180, loss = 0.510766
I0226 10:28:23.971143 21209 solver.cpp:245] Train net output #0: rpn_cls_loss = 0.237885 (* 1 = 0.237885 loss)
```

图 4-9 VGG 模型训练日志

在迭代到 40000 次时 loss 不再下降，停止训练，共历时大约 35 个小时完成模型训练，训练完成后根据训练日志作图，图 4-10 为分类 loss 图：

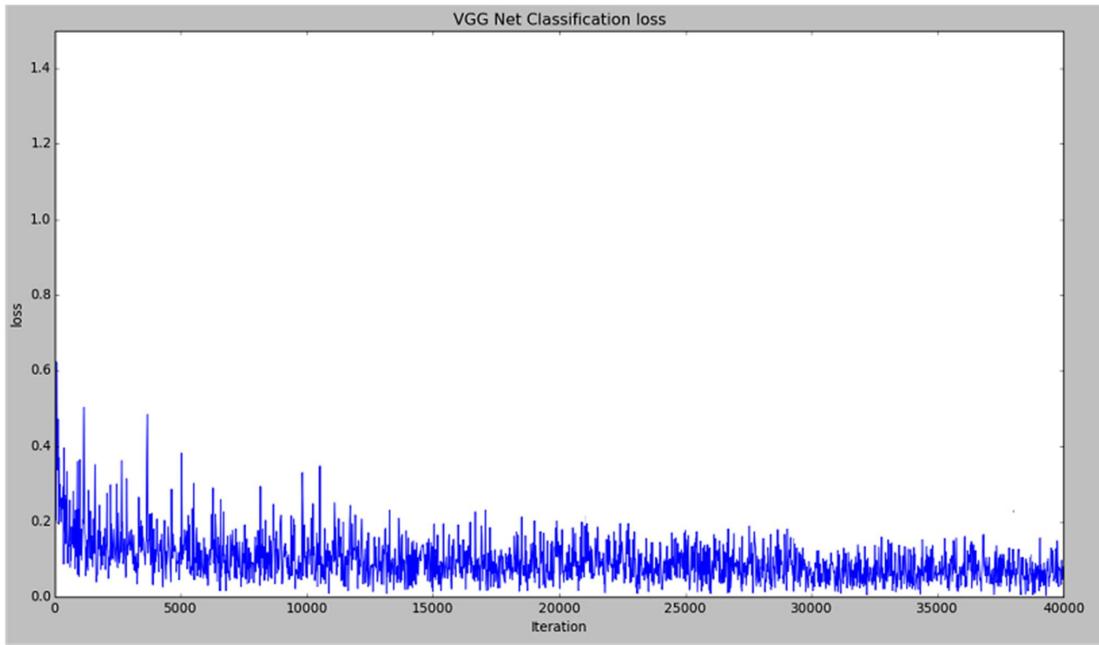


图 4-10 VGG 模型分类 loss 图

图 4-11 为 VGG 模型预测用于定位目标的边界框 loss 图：

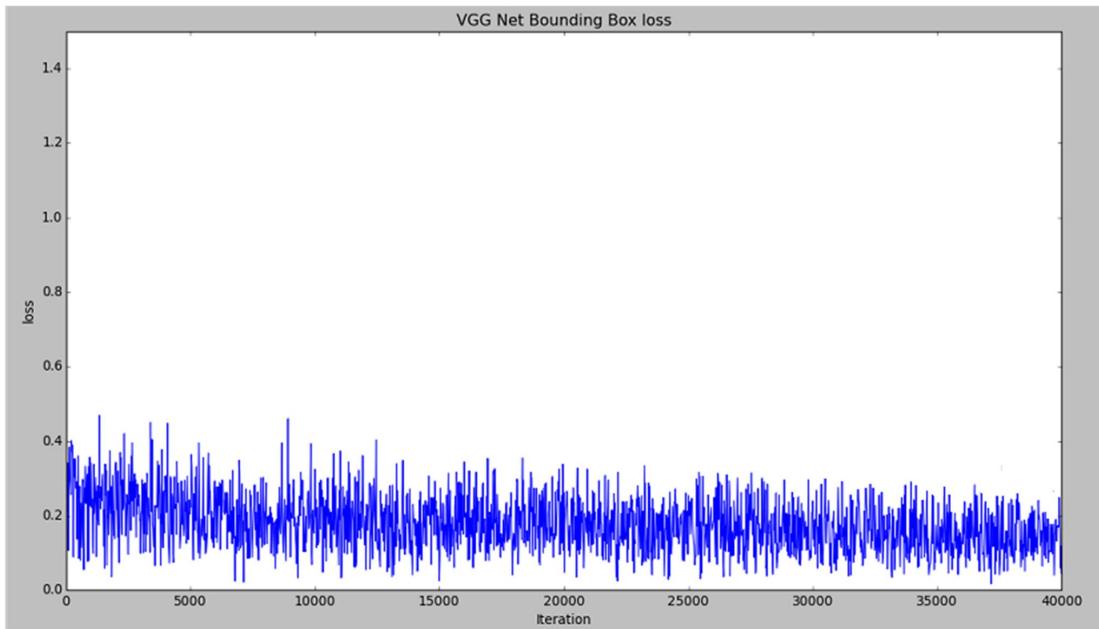


图 4-11 VGG 模型边界框 loss 图

4. 两种网络模型训练结果分析

通过对两种模型的分类 loss 图和边界框 loss 图，可以发现有相同点也有不同点。相同点是：随着训练迭代次数的增加，不论是用于表示对目标分类的 loss，还是用于表示对目标定位的 loss，最后都会处于一定范围内波动，模型训练效果基本达到。不同点是：不论从分类 loss 还是从边界框 loss 来说，VGG 模型都比 ZF 模型达到更好的收敛效果，可见对于相同数据集，网络模型越深，层数越多，训练效果可能越好。

(三) 模型测试

1. ZF 模型测试

测试之前，要搭建基于 ZF 网络的 Faster R-CNN 模型的测试网络，就是 ZF 网络和一个 RPN 网络的结合，与训练模型是相对应的，图 4-12 所示为 zf_faster_rcnn_test.prototxt 表示的框图：

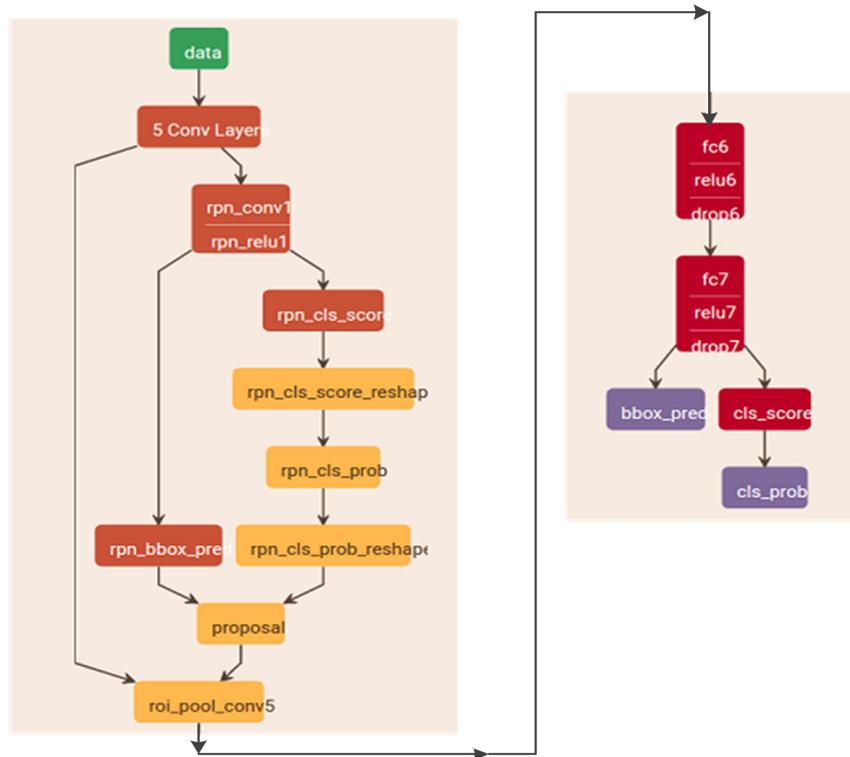


图 4-12 基于 ZF 网络的 Faster R-CNN 测试模型

完成模型的训练之后，生成了相应可用于测试的模型文件，接下来对模型进行测试，图 4-13 为 ZF 模型的测试日志图，图 4-14 为 ZF 模型的测试效果图：

```

superhui@superhui-Y: ~/caffe/Faster_rcnn/1060/ZF_ych/30000ZF 120x41
Reading annotation for 22401/24438
Reading annotation for 22501/24438
Reading annotation for 22601/24438
Reading annotation for 22701/24438
Reading annotation for 22801/24438
Reading annotation for 22901/24438
Reading annotation for 23001/24438
Reading annotation for 23101/24438
Reading annotation for 23201/24438
Reading annotation for 23301/24438
Reading annotation for 23401/24438
Reading annotation for 23501/24438
Reading annotation for 23601/24438
Reading annotation for 23701/24438
Reading annotation for 23801/24438
Reading annotation for 23901/24438
Reading annotation for 24001/24438
Reading annotation for 24101/24438
Reading annotation for 24201/24438
Reading annotation for 24301/24438
Reading annotation for 24401/24438
Saving cached annotations to /home/shawn/py-faster-rcnn/data/VOCdevkit2007/annotations_cache/annots.pkl
AP for trafficsign = 0.4872
AP for car = 0.7892
AP for person = 0.8082
Mean AP = 0.6934
~~~~~
Results:
0.4872
0.7892
0.8082
0.6934
~~~~~

```

图 4-13 ZF 模型测试日志图



图 4-14 ZF 模型测试效果图

2. VGG 模型测试

测试之前，搭建基于 VGG 网络的 Faster R-CNN 模型的测试网络，这也是与训练模型相对应的，图 4-15 所示为 vgg_faster_rcnn_test.prototxt 表示的框图，可以看出它与基于 ZF 网络的 Faster R-CNN 模型相比，最主要的不同是开始部分的 5 层卷积层变成了 13 层卷积层，层数更深。

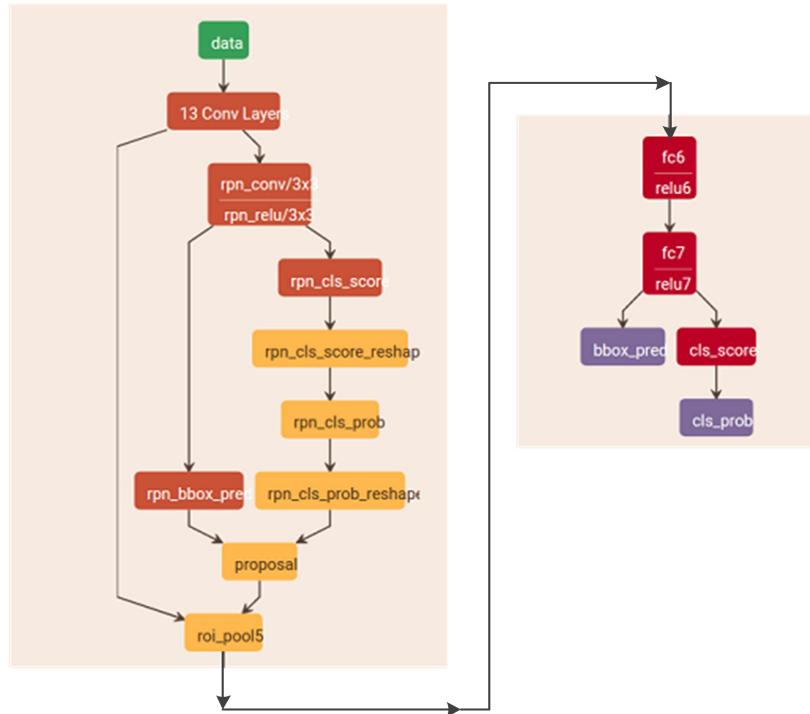


图 4-15 基于 VGG 网络的 Faster R-CNN 测试模型

图 4-16 为 VGG 模型的测试日志图，图 4-17 为 VGG 模型测试效果图：

```

superhui@superhui-Y: ~/caffe/Faster_rcnn/1060/VGG16_Fc/fcl2 164x43
Reading annotation for 11801/13799
Reading annotation for 22401/24438
Reading annotation for 22501/24438
Reading annotation for 22601/24438
Reading annotation for 22701/24438
Reading annotation for 22801/24438
Reading annotation for 22901/24438
Reading annotation for 23001/24438
Reading annotation for 23101/24438
Reading annotation for 23201/24438
Reading annotation for 23301/24438
Reading annotation for 23401/24438
Reading annotation for 23501/24438
Reading annotation for 23601/24438
Reading annotation for 23701/24438
Reading annotation for 23801/24438
Reading annotation for 23901/24438
Reading annotation for 24001/24438
Reading annotation for 24101/24438
Reading annotation for 24201/24438
Reading annotation for 24301/24438
Reading annotation for 24401/24438
Saving cached annotations to /home/madaiqian/py-faster-rcnn/data/VOCdevkit2007/annotations_cache/annots.pkl
AP for trafficsign = 0.5662
AP for car = 0.8018
AP for person = 0.8528
Mean AP = 0.7403
~~~~~
Results:
0.5662
0.7718
0.8528
0.7403
~~~~~
```

图 4-16 VGG 模型测试日志图

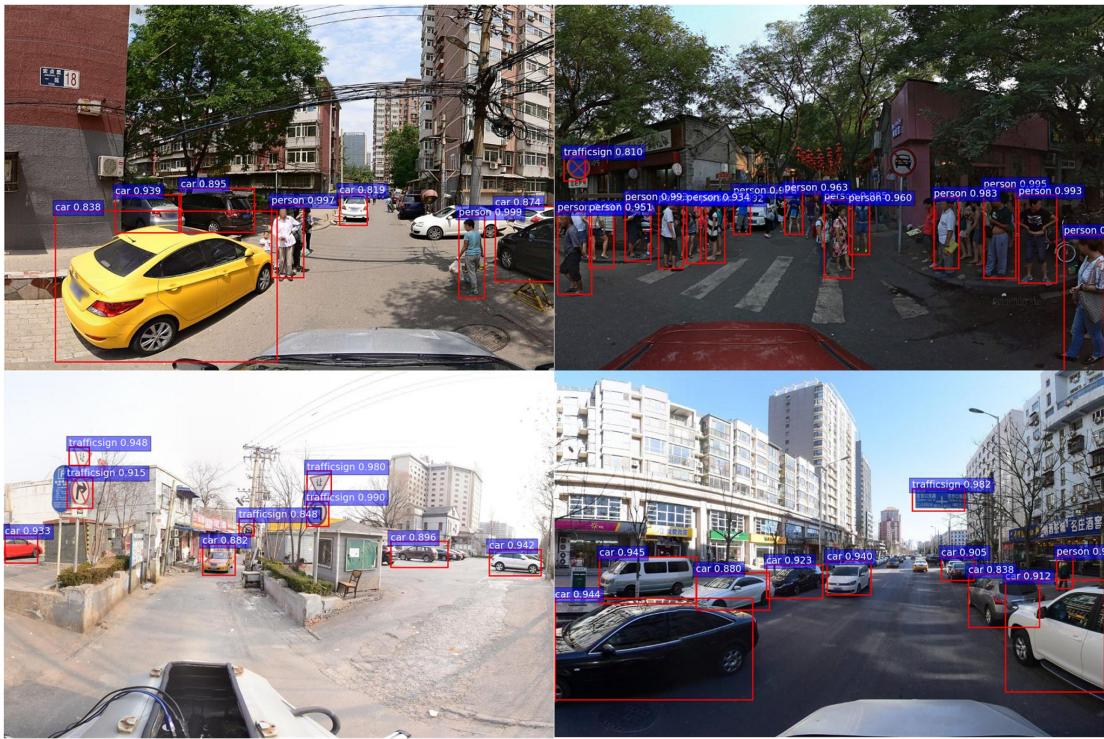


图 4-17 VGG 模型测试效果图

3. 两种网络模型测试结果分析

首先，对比 ZF 模型和 VGG 模型的测试日志看出，ZF 模型的三种类型的 mAP 为 69.34%（类型 Trafficsign: 48.72%，类型 Car: 78.92%，类型 Person: 80.82%），而 VGG 模型的三种类型的 mAP 为 74.03%（类型 Trafficsign: 56.62%，类型 Car: 80.18%，类型 Person: 85.28%），如表 4-1 所示：

表 4-1 ZF 模型与 VGG 模型测试 mAP

类 模型	交通标志类 (Trafficsign) AP	汽车类 (Car) AP	行人类 (Person) AP	mAP
ZF模型	48.72%	78.92%	80.82%	69.34%
VGG模型	56.62%	80.18%	85.28%	74.03%

由表 4-1，通过对比可以发现，基于 VGG 网络的 Fster R-CNN 模型在三种类别的测试平均精确度都要比 ZF 网络高，而且总的 mAP 比 ZF 网络高了 5% 左右；通过对图 4-14 和图 4-17 可以发现，基于 VGG 网络的 Fster R-CNN 模型在测试中在复杂背景图片中识别并定位出三种类别目标的能力和效果都比 ZF 要强，这也符合由图 4-18 所展现的 mAP 差异。

五、总结与展望

基于深度学习的目标识别技术是计算机视觉领域的热点，本文首先介绍了基于深度学习的目标识别方法的研究背景和国内外研究现状，然后从深度学习中神经网络模型基础理论入手，介绍了 CNN 网络、ZF 网络和 VGG 网络，Faster R-CNN 模型的理论基础和原理，通过设计简易分布式爬虫系统爬取了百度街景大约 60 万张街景图片作为模型的训练集和测试集，用 Python 语言基于深度学习框架 Caffe 接口搭建了两种 Faster R-CNN 模型（ZF 模型和 VGG 模型），对这两种模型的训练和测试结果进行了对比分析。

实验结果表明：在计算能力平台、训练数据集和测试数据集等相同的情况下，更深层的卷积神经网络能使模型训练更快的达到收敛，并且训练生成的模型在相同的测试集上不论是 mAP 还是测试效果都更好，有更好的泛化能力。

测试也发现不足之处，这两种模型在目标识别过程中有误识别或漏识别，对于大图片中更小的目标无法识别等现象，通过分析原因，可以在以下几个方面进行改进：在数据集方面，应该继续扩大数据集，包括训练集和测试集；在图片标注方面，应该想办法提高图片中目标的标注精度，使得标注涵盖的目标类别和定位框更精准；在网络模型方面，可以试着选用更深层的网络模型等。基于深度学习的目标识别技术的的网络模型和应用范围在未来研究中还需进一步扩展。

参考文献

- [1] 魏敦生. 星空背景下小目标的跟踪与识别[D]. 上海交通大学, 2012.
- [2] Lowe D. Object recognition from local scale-invariant feature. International Conference on Computer Vision, 1999, 1150-1157.
- [3] Kadir T, Brady M. Scale, saliency and image description[J]. Computer Vision, 2001, 45(2):83-105.
- [4] Mikolajczyk K, Schmid C. Indexing based on scale invariant interest points[C]. In: Proc. of 8th IEEE International Conference on Computer Vision(ICCV 2001), Vancouver, BC, Canada, 2001, Vol.1:525-531.
- [5] Freund Y. Boosting a Weak Learning Algorithm by Majority[J]. Information and Computation, 1995, 121(2):256-285.
- [6] 边肇棋,张学工. 模式识别[M]. 北京:清华大学出版社, 2003.
- [7] Y. LeCun, L. Bottou, Y. Bengio and P. Haffner: Gradient-Based Learning Applied to Document Recognition, Proceedings of the IEEE, 86(11):2278 -2324, November 1998.
- [8] Hinton G E, Salakhutdinov R R. Reducing the dimensionality of data with neural networks[J]. Science, 2006, 313(5786):504-507.
- [9] Deng J, Dong W, Socher R, et al. Imagenet: A large-scale hierarchical image database[C]:Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on. IEEE, 2009:248-255.
- [10] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In CVPR, 2014.
- [11] Ouyang W, Luo P, Zeng X, et al. Deep ID-Net:multi-stage and deformable deep convolutional neural networks for object detection[J]. arXiv preprint arXiv:1409.3505, 2014.
- [12] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In Neural Information Processing Systems (NIPS), 2015.
- [13] 程欣. 基于深度学习的图像目标定位识别研究[D]. 电子科技大学, 2016.
- [14] 阮怀玉. 基于稀疏表示和深度学习的 SAR 图像目标识别研究[D]. 中国科学技术大学, 2016.
- [15] 傅瑞罡. 基于深度学习的图像目标识别研究[D]. 国防科学技术学, 2014.
- [16] Zeiler, M.D. and Fergus, R., 2014, September. Visualizing and understanding convolutional networks. In European conference on computer vision (pp. 818-833). Springer International Publishing.
- [17] Simonyan, K. and Zisserman, A., 2014. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv: 1409.1556.
- [18] Krizhevsky, A., Sutskever, I. and Hinton, G.E., 2012. Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems (pp. 1097-1105).
- [19] Matsugu, M., Mori, K., Mitari, Y. and Kaneda, Y., 2003. Subject independent facial expression recognition with robust face detection using a convolutional neural network. Neural Networks, 16(5), pp.555-559.
- [20] Krizhevsky, A., Sutskever, I. and Hinton, G.E., 2012. Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems (pp. 1097-1105).

- [21] Uijlings, J.R., Van De Sande, K.E., Gevers, T. and Smeulders, A.W., 2013. Selective search for object recognition. *International journal of computer vision*, 104(2), pp.154-171.
- [22] Suykens, J.A. and Vandewalle, J., 1999. Least squares support vector machine classifiers. *Neural processing letters*, 9(3), pp.293-300.
- [23] C. Lawrence Zitnick and Piotr Dollar. Edge boxes: Locating object proposals from edges. In *ECCV*. European Conference on Computer Vision, September 2014.

谢 辞

首先感谢我的毕业设计指导老师郭尊华老师，在毕业设计中，郭老师给我很大鼓舞和指导。论文形成，老师对格式内容等进行了修改、完善。在此，我要向郭尊华老师表示我最诚挚的谢意。同时要感谢中科院计算所的彭晓晖老师，在完成毕业设计的过程中给予我的鼓励和方向性指导。

感谢教授过我的老师们和辅导员姜文老师，是他们帮助我完成我大学本科阶段的学习。感谢四年来朝夕相处的室友和同学，他们在学习和生活上给予了我多方面的照顾与帮助，我非常珍视和他们一起度过的美好时光。最后，感谢我的父母，他们鼓励我支持我，为我的成长倾注了大量的心血，让我成长为今天的模样。

在山东大学（威海）的学习生活即将结束，感谢这四年的大学时光让我成长，我会在新的起点继续努力，但行好事，莫问前程。