

专业代码 080202

湖南师范大学

本科毕业设计

题 目 (中文): 基于深度学习的机器人实时目标检测算法研究

(英文): Study on Deep Learning Based Real-time Object
Detection for Robotics

姓 名: 周如意

学 号: 2013180429

学 院: 工程与设计学院

专业 年级: 机械设计制造及其自动化2013 级

指 导 教 师: 康辉梅

二〇一七年五月

湖南师范大学本科毕业设计诚信声明

本人郑重声明：所呈交的本科毕业设计，是本人在指导老师的指导下，独立进行研究工作所取得的成果，成果不存在知识产权争议，除设计中已经注明引用的内容外，本设计不含任何其他个人或集体已经发表或撰写过的作品成果。对本设计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。本人完全意识到本声明的法律结果由本人承担。

本科毕业设计作者签名：

二〇一七年五月四日

湖南师范大学本科毕业设计任务书

毕业设计题目	基于深度学习的机器人实时目标检测算法研究				
作 者 姓 名	周如意	所属院、专业、年级	工程与设计学院 机械设计制造及其自动化专业 2013 级		
指导教师姓名、职称	康辉梅 副教授	预计字数	30000	开题日期	2017.01.07

选题的目的和意义:

自然界的一切图像都是连续变化的模拟图像，日常生活中，图像里的某些特定目标才是我们比较关心的。例如，在亚马逊的物流分拣竞赛上，工业机械臂需要运用目标检测技术进行商品分拣。快速、准确的目标检测算法在智能化监控系统、自动驾驶系统和智能机器人系统等领域均有广泛的应用，目标检测正成为近年来理论和应用的研究热点。

同时，在深度学习技术的支持下，基于卷积神经网络的学习技术让自动物体识别和目标检测等任务有了显著的效果提升，人脸识别更是有了超过人眼的准确率。目前，基于深度学习的目标检测算法在检测精度和实时性能方面还有提升的空间。

因此，本文深度调研目标检测算法发展，实现一种最新的基于深度学习的实时目标检测算法 Single Shot MultiBox Detector(SSD)，并训练测试其在不同超参数下的算法性能，提出算法改进思路，对进一步提高该算法性能具有重要意义。

主要研究内容:

1. 最新目标检测算法的深入研究。针对目前最新的基于深度学习的目标检测算法 Single Shot MultiBox Detector(SSD)做深入研究，理解学术创新点和其背后的逻辑，分析与其他计算方法相比的主要优势和劣势。
2. 目标检测算法实现。学习和掌握在深度学习开源平台 MXNet 下的基本开发流程，配置算法实现框架，自行实现 Single Shot MultiBox Detector(SSD)目标检测算法。
3. 算法性能测试。深度学习技术需要大量的超参数设计网络结构，对已实现的 Single Shot MultiBox Detector(SSD)目标检测算法，在不同超参数下训练算法的收敛性能和目标检测结果的有效性。

应达到的技术指标或要求:

1. 在 Pascal VOC2007 标准训练集上训练网络，在 VOC2007 上测试算法性能，测试结果的平均检测精度(mAP)达到 66%。
2. 使用 GPU 运行算法，采用 VGG16 模型作为基本构架，算法检测速度应达到实时要求。

主要设计方法或技术路线:

1. 文献调研与理论学习。查阅目标检测算法相关文献，了解国内外研究现状以及最新研究进展。同时学习机器学习与深度学习的基础理论知识，为深入理解与实现算法奠定理论基础。
2. Single Shot MultiBox Detector(SSD)算法深入研究。理解该算法的学术创新点及其背后的逻辑，分析该算法与其他算法相比的优势和劣势。
3. 在深度学习开源平台 MXNet 下配置算法实现框架。学习和掌握 MXNet 框架的基本开发流程，学习 python 语言并在 MXNet 平台下实现 Single Shot MultiBox Detector(SSD)算法。
4. 训练深度神经网络。在标准数据集上利用样本训练构建的卷积神经网络，得到合适的特征参数。
5. 算法性能测试。在多个训练集上测试已训练的网络，得出该算法的单项/平均检测精度以及算法检测速度。
6. 网络超参数调整。调整网络超参数，在不同超参数网络结构下进行训练并测试，得出算法的收敛性能以及目标结果的有效性，选择较优参数。

完成本课题应具备的环境（软件、硬件）:

软件环境：MXNet、Python

硬件环境：计算机、GPU

各阶段任务安排:

第一阶段（2016.12-2017.01）：查阅相关文献资料，了解目标检测算法的发展，扩展对机器学习和深度学习相关基本理论知识的学习，完成开题报告；

第二阶段（2017.01-2017.02）：针对 Single Shot MultiBox Detector (SSD) 目标检测算法做深度研究，深入理解算法思想与学术创新点。完成深度学习开源平台 MXNet 的框架配置，熟悉编程流程和相应的编程语言；

第三阶段（2017.02-2017.03）：完成最新的基于深度学习的目标检测算法 Single Shot MultiBox Detector(SSD) 在 MXNet 平台上的实现工作；

第四阶段（2017.03-2017.04）：分析已实现的 Single Shot MultiBox Detector(SSD) 目标检测算法对不同的超参数的收敛性与检测有效性，撰写毕业设计初稿以及准备预答辩；

第五阶段（2017.04-2017.05）：修改完善毕业设计以及形成论文定稿。

主要参考资料:

- [1] Redmon J, Divvala S, Girshick R, et al. You only look once: Unified, real-time object detection[J]. arXiv preprint arXiv:1506.02640, 2015.
- [2] Girshick R. Fast r-cnn[C]//Proceedings of the IEEE International Conference on Computer Vision. 2015: 1440-1448.
- [3] Ren S, He K, Girshick R, et al. Faster R-CNN: Towards real-time object detection with region proposal networks[C]//Advances in neural information processing systems. 2015: 91-99.
- [4] Liu W, Anguelov D, Erhan D, et al. SSD: Single Shot MultiBox Detector[J]. arXiv preprint arXiv:1512.02325, 2015
- [5] Erhan D, Szegedy C, Toshev A, et al. Scalable object detection using deep neural networks[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2014: 2147-2154.
- [6] 丰晓霞. 基于深度学习的图像识别算法研究[D].太原理工大学,2015.
- [7] 黄凯奇,任伟强,谭铁牛. 图像物体分类与检测算法综述[J]. 计算机学报,2014,06:1225-1240.
- [8] 李义. 基于相关学习神经网络的图像识别方法研究[D].哈尔滨工业大学,2015.
- [9] 卢宏涛,张秦川. 深度卷积神经网络在计算机视觉中的应用研究综述 [J]. 数据采集与处理,2016,01:1-17.
- [10] 王振,高茂庭. 基于卷积神经网络的图像识别算法设计与实现 [J]. 现代计算机(专业版),2015,20:61-66.

指导教师意见:

指导教师签名:

年 月 日

开题报告会纪要					
时间			地点		
指导 小组 成员	姓 名	职务（职称）	姓 名	职务（职称）	姓 名

会议记录摘要：

会议主持人签名： 记录人签名： 年 月 日

指导 小组 意见			
	负责人签名：	年	月
学院 意见			
	负责人签名：	年	月

二、湖南师范大学本科毕业设计评审表

毕业设计 项目	基于深度学习的机器人实时目标检测算法研究				
作者姓名	周如意	所属院、专业、年级	工程与设计学院 机械设计制造及其自动化专业 2013 级		
指导教师 姓名、职称	康辉梅 副教授	字 数	30000	定稿日期	2017.05.04
中文摘要	<p>无人仓库中，具有智能抓取能力的移动作业机器人承担着举足轻重的角色。抓取过程中，机器人视觉系统对目标物体的正确识别和精确定位是抓取成功的先决条件。本文主要研究一种基于深度学习的实时目标检测算法 Single Shot MultiBox Detector(SSD)，来实现机器人实时目标检测，旨在将视觉领域的最新研究应用到机器人技术中，为抓取提供实时性与精确率俱佳的感知信息。本文的工作主要包括以下几个部分：</p> <p>第一部分以亚马逊物流分拣挑战赛为智能抓取任务场景，设计移动作业机器人的软硬件系统。通过分析多种目标检测算法的检测过程，比较各算法性能特点，选取 SSD 算法作为抓取任务在目标检测环节中的算法原型。</p> <p>第二部分，分别从网络结构、参数调节、训练技巧等方面解析 SSD 算法的实现细节。通过在 Pascal VOC 数据集上的网络训练测试实验，探讨提高算法性能的策略，训练时的参数设置，以及训练技巧对算法性能的影响等问题。</p> <p>第三部分面向亚马逊物流分拣挑战赛中的任务对象，通过图像采集、数据标注、标签制作等步骤创建 VOC 格式的图像数据集。然后在自建数据集上进行 SSD 算法网络微调实验，训练网络学习数据集的特征表示。最后，实验测试训练效果，通过调整参数和增加数据增广策略，研究进一步提高训练效果的有效方法。</p> <p>本文所研究的实时目标检测算法是机器人完成智能抓取时所涉及到的主要技术问题之一，论文的研究成果可以在一定程度上增强机器人识别和定位目标的智能能力，是目标检测算法 SSD 应用于智能抓取机器人的一次有利探索。</p>				
关键词 (3-5 个)	智能抓取机器人；深度学习；目标检测				

英文摘要

In unmanned warehouses, mobile operating robots with independent picking ability take critical roles. During the picking process, correct object classification and precise locating are prerequisites. This thesis describes a method of deep learning based real-time object detection for robotics, Single Shot MultiBox Detector (SSD), applying the latest object detection method on intelligent picking robots. This method strengthens robustness on robot vision, with which robots get real-time and accurate information for picking. The main contents of this thesis are summarized as follows.

First of all, taking Amazon Picking Challenge (APC) as a real picking scene, the hardware and software systems for intelligent picking robots are designed. We overview several popular object detection methods and compare their performance. Trading off strength and weakness of these methods, SSD is selected as a prototype for intelligent picking robots on the object detection part.

Secondly, details about SSD realization on MXNet is described with the aspects of network structure, parameter adjustment and training tricks. With SSD network experiment of training and testing on Pascal VOC dataset, we discuss performance improving strategies; parameters setting as well as the influence of training tricks.

Thirdly, with steps of image collecting, annotating and format converting, we build an APC object dataset. SSD convolution neural network is trained on self-built dataset through finetune. The network learns dataset features and updates its parameters to fit forecast results. Finally, examinations about prefixed parameters as well as data augmentation of rebrightness are showed contributed or not.

All in all, the real-time object detection method this thesis studied is one of the main difficulties in robotic intelligent picking. The results promote classification as well as locating performance for picking robots, which is a positive try to apply SSD on intelligent picking robots.

关键词
(3-5个)

Intelligent Picking Robot; Deep Learning; Object Detection

毕业设计指导教师评定成绩								
评审基元	评审要素	评审内涵	满分	实评分				
选题质量 28%	目的明确符合要求	选题符合专业培养目标，体现学科、专业特点和综合训练的基本要求	9					
	选题恰当	题目规模适当	5					
		题目难度适中	5					
能力水平 30%	联系实际	题目与生产、科研、实验室建设等实际相结合，具有一定的实际价值	9					
	综合运用知识能力	能将所学专业知识和机能用与毕业设计中；设计内容有适当的深度、广度和难度	5					
	应用文献资料能力	能独立查阅相关文献资料，能对本设计所涉及的有关研究状况及成果归纳、总结和恰当运用	5					
	实验（设计）能力	能运用本学科常用的研究方法，选择合理可行的方案，能对实际问题进行分析，进行实验（设计），具有较强的动手能力	5					
	计算能力	原始数据搜集得当；能进行本专业要求的计算，理论依据正确，数据处理方法和处理结果正确	5					
	计算机应用能力	能根据设计题目要求编程上机或使用专业应用软件完成设计任务	5					
设计质量 32%	分析能力	能对设计项目进行技术经济分析或对实验结果进行综合分析	5					
	插图或图纸质量	能用计算机绘图，且绘制图纸表格符合标准	5					
	说明书撰写水平	设计说明书齐全；概念清楚，内容正确，条理分明，语言流畅，结构严谨；篇幅达到学校要求	12					
	规范化程度	设计的格式、图纸、数据、用语、量和单位、各种资料引用和运用规范化，符合标准；设计栏目齐全合理	5					
	成果的实用性与科学性	较好地完成设计选题的目的要求，成果富有一定的理论深度和实际运用价值	5					
外文资料翻译 10%	外文应用能力	具有创新意识，设计具有一定的创新性	5					
		能搜集、阅读、翻译、归纳、综述一定量的本专业外文资料与外文摘要，并能加以运用，体现一定的外语水平，译文汉字数 1500-2000 没有要求外文资料翻译的，参照英文摘要及英文文献等情况计分	10					
总成绩：			评定等级：					
指导教师评审意见： (结合设计内容，评价设计是否符合要求；选题是否恰当；是否联系实际；综合运用知识和应用文献资料的能力；实验（设计）能力、说明书撰写水平；成果的实用性与科学性；外文应用能力等)								
指导教师签名：			年 月 日					

说明：评定成绩分为优秀、良好、中等、及格、不及格五个等级，总成绩 90—100 分记为优秀，80—89 分记为良好，70—79 分记为中等，60—69 分记为及格，60 分以下记为不及格。若译文成绩为零，则不计总成绩，评定等级记为不及格。

三、湖南师范大学本科毕业设计答辩记录表

毕业设计 题 目	基于深度学习的机器人实时目标检测算法研究		
作者姓名	周如意	所属院、专业、年级	工程与设计学院 机械设计制造及其自动化专业 2013 年级
指导教师 姓名、职称	康辉梅 副教授		

答 辩 会 纪 要

时间				地点		
答辩 小组 成员	姓 名	职务（职称）	姓 名	职务（职称）	姓 名	职务（职称）

答辩中提出的主要问题及回答的简要情况记录:

会议主持人签名:

记录人签名:

年 月 日

答辩小组意见	评定成绩（百分制）及等级（优秀、良好、中等、及格、不及格）： 负责人签名： 年 月 日		
学院意见	评定成绩（百分制）及等级（优秀、良好、中等、及格、不及格）： 负责人签名： 学院（公章） 年 月 日		
学校意见	评定成绩（百分制）及等级（优秀、良好、中等、及格、不及格）： 负责人签名： 年 月 日		

目 录

摘 要	1
ABSTRACT	2
1 绪 论	3
1.1 课题背景	3
1.2 研究目的和意义	4
1.3 国内外研究现状	5
1.4 本文的主要研究内容及结构安排	8
2 总体方案与背景知识	9
2.1 任务场景介绍	9
2.2 智能抓取机器人总体方案	9
2.2.1 机器人硬件系统	9
2.2.2 机器人软件系统	12
2.3 目标检测算法方案选择	13
2.3.1 目标检测任务	13
2.3.2 目标检测算法性能要求	13
2.3.3 实时目标检测算法性能评估指标	13
2.3.4 目标检测算法常用数据集	15
2.3.5 多种目标检测算法过程分析与性能比较	16
2.4 深度学习框架选择及相关基本概念	20
2.4.1 常用深度学习框架	20
2.4.2 深度学习基本概念	21
2.5 本章小结	24
3 基于实验的 SSD 算法实现及参数匹配	25
3.1 实时目标检测算法 SSD 解析	25
3.1.1 SSD 卷积神经网络拓扑结构	25
3.1.2 SSD 网络训练及相关技巧	29
3.2 目标检测算法训练参数设置	32
3.2.1 批量大小	32
3.2.2 学习速率	32
3.2.3 动量	33
3.2.4 权重衰减	33

3.3 SSD 算法的参数网络训练实验及结果分析	33
3.3.1 SSD 算法在 Pascal VOC2007 上的训练测试结果	33
3.3.2 SSD 训练的特殊技巧对检测精度的影响分析.....	36
3.3.3 检测速度测试结果	40
3.4 本章小结	40
4 基于 SSD 算法的物体抓取测试	41
4.1 面向亚马逊物流分拣挑战赛的目标图像采集	41
4.1.1 亚马逊物流分拣挑战赛的目标对象	41
4.1.2 RGB-D 传感器的成像模型及图像采集.....	41
4.2 目标检测数据集的创建.....	43
4.2.1 生成标注信息	43
4.2.2 xml 标签文件生成	45
4.2.3 训练和测试数据集生成	45
4.3 自建数据集上的网络微调实验及结果分析	46
4.3.1 自建数据集训练结果	46
4.3.2 针对自建数据集的 SSD 网络超参数调节	48
4.3.3 数据增广策略对自建数据集训练效果的影响实验	50
4.3.4 部分检测结果展示	52
4.4 本章小结	52
结 论	54
参考文献	55
附 录	58
附录 A 主流深度学习框架性能表	58
附录 B MXNET 框架下的 SSD 网络构建程序	59
致 谢	67

基于深度学习的机器人实时目标检测算法研究

机械设计制造及其自动化 2013 级 周如意

摘要

无人仓库中，具有智能抓取能力的移动作业机器人承担着举足轻重的角色。抓取过程中，机器人视觉系统对目标物体的正确识别和精确定位是抓取成功的先决条件。本文主要研究一种基于深度学习的实时目标检测算法 Single Shot MultiBox Detector(SSD)，来实现机器人实时目标检测，旨在将视觉领域的最新研究应用到机器人技术中，为抓取提供实时性与精确率俱佳的感知信息。本文的工作主要包括以下几个部分：

第一部分以亚马逊物流分拣挑战赛为智能抓取任务场景，设计移动作业机器人的软硬件系统。通过分析多种目标检测算法的检测过程，比较各算法性能特点，选取 SSD 算法作为抓取任务在目标检测环节中的算法原型。

第二部分，分别从网络结构、参数调节、训练技巧等方面解析 SSD 算法的实现细节。通过在 Pascal VOC 数据集上的网络训练测试实验，探讨提高算法性能的策略，训练时的参数设置，以及训练技巧对算法性能的影响等问题。

第三部分面向亚马逊物流分拣挑战赛中的任务对象，通过图像采集、数据标注、标签制作等步骤创建 VOC 格式的图像数据集。然后在自建数据集上进行 SSD 算法网络微调实验，训练网络学习数据集的特征表示。最后，实验测试训练效果，通过调整参数和增加数据增广策略，研究进一步提高训练效果的有效方法。

本文所研究的实时目标检测算法是机器人完成智能抓取时所涉及到的主要技术问题之一，论文的研究成果可以在一定程度上增强机器人识别和定位目标的智能能力，是目标检测算法 SSD 应用于智能抓取机器人的一次有利探索。

关键词：智能抓取机器人；深度学习；目标检测

Study on Deep Learning Based Real-time Object Detection for Robotics

ABSTRACT

In unmanned warehouses, mobile operating robots with independent picking ability take critical roles. During the picking process, correct object classification and precise locating are prerequisites. This thesis describes a method of deep learning based real-time object detection for robotics, Single Shot MultiBox Detector (SSD), applying the latest object detection method on intelligent picking robots. This method strengthens robustness on robot vision, with which robots get real-time and accurate information for picking. The main contents of this thesis are summarized as follows.

First of all, taking Amazon Picking Challenge (APC) as a real picking scene, the hardware and software systems for intelligent picking robots are designed. We overview several popular object detection methods and compare their performance. Trading off strengthness and weakness of these methods, SSD is selected as a prototype for intelligent picking robots on the object detection part.

Secondly, details about SSD realization on MXNet is described with the aspects of network structure, parameter adjustment and training tricks. With SSD network experiment of training and testing on Pascal VOC dataset, we discuss performance improving strategies; parameters setting as well as the influence of training tricks.

Thirdly, with steps of image collecting, annotating and format converting, we build an APC object dataset. SSD convolution neural network is trained on self-built dataset through finetune. The network learns dataset features and updates its parameters to fit forecast results. Finally, examinations about prefixed paramaters as well as data augmentation of rebrightness are showed contributed or not.

All in all, the real-time object detection method this thesis studied is one of the main difficulties in robotic intelligent picking. The results promote classification as well as locating performance for picking robots, which is a positive try to apply SSD on intelligent picking robots.

Keywords: Intelligent Picking Robot; Deep Learning; Object Detection

1 绪 论

1.1 课题背景

随着移动互联网的发展，电子商务正在深刻地改变着零售行业的格局，全球范围内网络购物渗透率快速上升，电商的兴起为物流仓储自动化市场带来了历史性机遇。纵观物流自动化现状，与大量高速流出的线上订单相比，线下的物流技术面临明显压力。电商平台的实际需求无论从频次、品类还是总量上，都对现有的物流框架提出了更高的要求。因此，以亚马逊为首的各大电商平台纷纷致力于创建自己的智能化物流仓储系统。大型无人仓库在各地建立，送货无人机、仓储自动化机器人等设备也相继投入使用。依据目前的发展态势，物流技术正大步迈向智慧物流阶段。智慧物流技术以“无人仓库”为载体，其核心特色体现为数据感知、机器人融合和算法指导，全面改变目前仓储的运行模式，极大提升效率和降低人力消耗。在这样的仓储环境中，具有智能抓取能力的移动作业机器人是实现高效无人运作的重要载体，也是智能物流装备中不可或缺的重要角色。开展具有智能抓取能力的移动作业机器人研究是推进智慧物流系统的迫切需求，也是移动作业机器人研究的热点问题。

在智慧物流仓储系统中，针对各部分的运作机制，各大电商有各自不同的系统框架，也进行了多种尝试。2012 年，亚马逊发布其无人物流仓储系统工作视频，特制的 Kiva 系统作业效率比传统的物流作业高出 2-4 倍。其运作原理为：工作人员向 Kiva 下达订单，机器接受指令后扫描地上条码运行到指定位置，将目标货物所在的货架从仓库区搬送至员工处理区。

2016 年 11 月，京东对外演示其物流仓储场景，整个仓储系统依靠搬运机器人、货架穿梭车、分拣机器人、堆垛机器人、六轴机器人、无人叉车等一系列物流机器人的相互协作。其中，扮演重要角色的 DELTA 型分拣机器人采用 3D 视觉系统，能够实现动态拣选、自动更换拾取器以及 155ppm 的作业节拍，具有三轴并联机械结构以及适应货物转角偏差辅助轴的特点。

2016 年底，深圳蓝胖子第一代移动运输机器人发布，其运作过程为：机器人适应并学习周围的环境，接入 WMS 和 TMS 等系统接受指令，通过自主导航到达大概位置，通过三维视觉判断物体及摆放姿态，自主“思考”，进行轨迹规划和抓取规划，并把物体放在合作的小车上，小车返回到进出口完成整个过程。这种不同于 Kiva 的思路，

在尽量不改造仓库的前提下，直接落地机器人，更符合未来物流全程自动化的发展趋势。

2015年5月，亚马逊在西雅图举办的国际机器人和自动化大会上(ICRA 2015)发起“亚马逊分拣机器人挑战”(Amazon Picking challenge, APC)，希望将学术研究和工业机器人的成果付诸产业化应用，完全取代人类进行包裹分拣、包装。

本课题来源于哈工大机器人(合肥)国际创新研究院工业智能装备所的“基于视觉感知的智能抓取机器人”项目。该项目旨在将目标检测领域的最新研究进展与机器人技术相结合，研究一种具有智能抓取能力的移动作业机器人。

本文拟重点研究智能抓取过程中的目标检测环节，实现一种实时目标检测算法，使其满足智能抓取机器人实时、精确的感知需求。以完成亚马逊物流分拣挑战赛任务为目标，针对相应检测对象，调整算法参数和训练技巧，将训练参数网络应用于智能移动作业机器人上。

1.2 研究目的和意义

机器人应用领域越来越广，机器人自动化水平也不断提高，在大规模仓储环境中，人们迫切希望用具有智能抓取能力的机器人装备代替人工，实现快速有效的无人化智能物流仓储管理。同时，机器人的智能抓取也是机器人研究领域的热点问题。

一个完整的机器人智能抓取过程由目标检测、位姿估计、抓取姿态生成、移动作业规划与执行控制等多个环节构成。其中，目标物体的正确识别与精确定位是抓取成功的前提条件。尽管近年来无线射频识别(Radio Frequency Identification, RFID)在环境感知领域取得了广泛的应用，基于视觉方式的识别仍然是机器人感知环境的重要渠道。尤其在机器人执行抓取任务中，单纯的环境感知难以提供足够的抓取信息，而必须与正确的物体识别和精确的物体定位紧密结合^[1]。同时，目标检测受到光线强弱、视角变换、位姿多样、部分遮挡等多重因素干扰，传统的视觉方法已无法满足这种环境下的目标识别与定位。因此，必须借助深度学习的方法，使算法具备更加智能的视觉判断能力，满足检测结果鲁棒性、准确性和实时性的要求。本研究的主要目的是实现机器人在智能抓取过程中的目标检测，为后续的位姿估计、抓取姿态生成、移动作业规划等步骤提供实时精确的环境感知信息，从而完成复杂的智能抓取任务。通过对目标检测算法发展脉络梳理和性能分析，本研究决定选用 SSD(Single Shot MultiBox Detector)算法框架作为完成实时目标检测的原型。此种算法具有较高检测精度，同时保

持实时性的检测速度，能及时为智能服务机器人的运动控制提供精确的感知信号。

1.3 国内外研究现状

Fetch Robotics 公司开发的 Fetch 和 Freight 机器人系统^[2] (如图 1.1 所示)是执行智能抓取任务的著名机器人。Fetch 具备自动导航功能，可以在货架间移动，识别产品并将其取下架运送到 Freight 的自动驾驶机器人里，Freight 的作用则与 Amazon 的 Kiva 相当。机器人可以自主规划路线和充电，从而保证整个仓储系统的无缝运行。其中，Fetch 由全方位移动基与 7 自由度机械臂构成，其末端执行器为平行的两指夹钳，另配有一个 3D 深度传感器，还可通过 SSH 进行信号传输。同时，Fetch 和 Freight 与 ROS 系统完全集成，可以运行 MoveIt，ROS 导航和自动校准等高级应用程序。

西班牙 Robotnik 公司推广的 RB-1 机器人^[3] (如图 1.2 所示)，其独特之处在于构架主要使用 Dynamixel Pro 伺服机构。该机构供电给 7 自由度的手臂、夹具、以及可左右旋转上下倾斜的头部。其躯干能升起近 40 公分，手臂可及范围达 70 公分(长度足够捡起掉在地上的东西)，可以提起重 2 公斤的物体，并拥有 3.5 小时的电池寿命。在传感器方面，RB-1 头部配置有华硕 Xtion RGB-D 传感器，底座配备有 Hokuyo 射频雷达，运行在下 Linux 的 ROS 系统上。

Robotnik 公司的另一款机器人 JR2(如图 1.3 所示)是一个 6 自由度的工业协作机器人。其手臂可安装几乎任何标准的末端执行器，包括 Schunk 或 Weiss Robotics 的两指伺服夹持器。基座平台可全向移动，安装有两个测距仪用于导航和安全检测，同时前部和后部集成了两个 RGB-D 传感器部件以检测不同高度处的障碍物。JR2 使用 ROS 开放式构架，许多软件可直接应用，包括导航系统、用于任务规划、诊断和远程控制的 HMI.JR2 模型以及完全配置的 MoveIt 包。



图 1.1 Fetch 和 Freight



图 1.2 RB-1



图 1.3 JR2

I am Robotics 公司的 Swift 移动挑选机器人(如图 1.4 所示), 针对药店仓库的自动挑选需求设计。该机器人可自主导航, 在多个仓库通道移动挑选产品, 能够检测并通知工作人员错误或储存不正确的产品。Swift 可自行浏览、检测和避免障碍物, 同时能够和人类一起工作, 以敏捷的方式实现仓库管理。Swift 采用独特的 RapidVision 技术使机器人可以实时查看和定位 3D 对象, 使用深度传感器识别和提取收集的 3D 几何信息, 并与闪光扫描仪收集的数据匹配。Swift 有 360 度的旋转能力, 可以检索任何高达 18 英寸深的货架。同时, 其有能力使用多个末端执行器来实现抓取, 最常见的末端执行器是真空吸盘, 可以抓取广泛的消费产品。

Unbounded Robotics 公司的 UBR-1 是一个具有 13 个自由度的移动机器人, 运行 ROS 系统, 其包括一个七个自由度的单手, 一个 3D 相机, 立体声麦克风以及扬声器。它能够以每小时 2 英里的速度在地板上移动, 并通过命令提示符轻松编程。可移动其手臂和头部来“示教”UBR-1, 也可以训练它在其视野中抓取物体。URB-1 在基座上使用激光扫描仪进行导航, 在其头部使用 PrimeScene 3D 传感器进行感知, 可升降躯干允许机器人将地板上的物体捡起, 并将其放在桌子或台面上。

西班牙的机器人制造商 PAL Robotics 推出的 Tiago(Take It And Go)^[4]行动机械手臂机器人, 如图 1.6 所示。Tiago 有三种不同级别的配置, 其中最高级的配置为 7 自由度的手臂, 外加一个包含力矩传感器的五指手爪, 同时配备 10 米的激光雷达进行导航。

Willow Garage 公司开发但未公开发布的一款机器人 PlatformBot^[5], 如图 1.7 所示。其基本设计包括一个移动基座, 一个带传感器的头部, 一个手臂和一个垃圾箱。PlatformBot 设法将机器人技术(如移动基座、手臂、夹具和传感器)的不同方面整合到低成本的平台中。



图 1.4 Swift



图 1.5 UBR-1



图 1.6 Tiago



图 1.7 PlatformBot

近年来，我国多所大学、研究所等也在机器人的智能抓取方面展开了一系列研究，其中比较有代表性的成果是深圳蓝胖子创业公司发布的移动运输机器人(如图 1.8 所示)。该机器人具有 3D 视觉，可分辨目标和障碍物，规划出无碰撞的运动轨迹，并移动到目标区域。其除了全方位全速移动之外，还具有地图构建与自动导航功能，可启动触觉抓取帮助灵巧手和吸盘完成控制任务；灵巧手采取模块化设计，触觉传感器和关节角度反馈使其能够实现仿真驱动抓握，轻松重构，并能抓取重达 80 公斤的不规则物体。上海交通大学团队开发了一款全方位移动双臂机器人(如图 1.9 所示)，该机器人头部有一个偏转自由度，双臂各有 6 个自由度，能够通过躯干的单目视觉检测物体，实现对竖直放置在固定高度水平桌面上的饮料瓶的定位与抓取^[1]。上海大学研制了一款轮式移动双臂服务机器人，该机器人双臂各有 6 个自由度，末端执行器有 1 个自由度，执行抓取任务时，机器人利用 RFID 进行定位，依靠头部的双目立体视觉实现基于颜色的物体检测，并计算其三维坐标，最后通过“look-then-move”的控制策略进行抓取^[6]。山东大学针对家庭环境下的物体识别与抓取设计了一款机器人，构建服务机器人智能空间，在每个待抓取物体上贴上特殊设计的 QR Code 标签。执行抓取任务时，机器人先利用 RFID 实现物体的粗定位，再通过 QR Code 标签实现物体的准确定位，最终利用开环视觉控制实现物体的抓取^[7]。但其使用的机械臂仅有 4 个自由度，难以实现对任意姿态的物体的抓取。此外，还有多家研究机构对机器人的智能抓取进行了研究，取得了丰硕的成果。

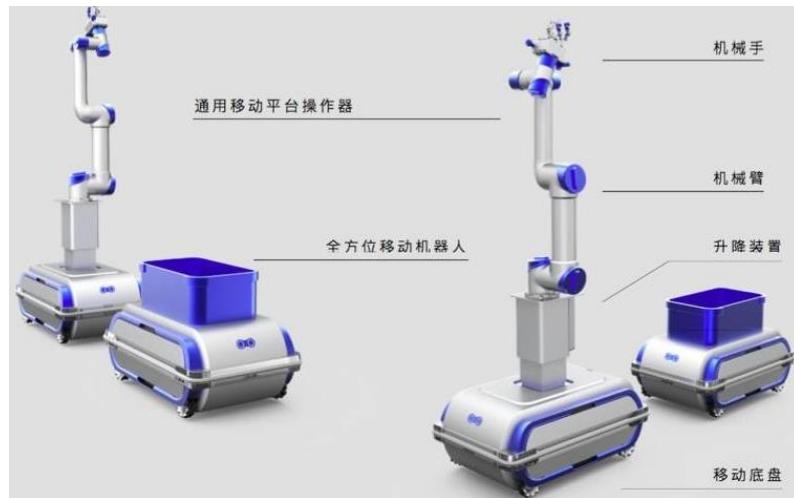


图 1.8 蓝胖子移动运输机器人



图 1.9 全方位移动双臂机器人^[1]

从总体上看，以上所介绍的执行抓取任务的机器人都只是在硬件上具备了基本的“视觉”和“抓取”模块，其末端大都难以适应各种材质、形状的物品抓取动作。同时，其软件上的“智能”水平也有待提高，主要表现为视觉感知算法不足以处理环境因素干扰严重时的目标识别

和定位。

1.4 本文的主要研究内容及结构安排

本文主要研究用于移动作业机器人在智能抓取过程中的目标检测算法。通过对机器视觉领域最新理论成果的研究，选取、实现并实验分析一种基于深度学习的机器人实时目标检测算法，即 **SSD** 算法，并将这种算法应用到移动作业机器人上，使其具有更精确更快速更智能的视觉能力，辅助完成复杂的亚马逊物流分拣挑战赛任务。本文主要内容包括：

(1)简述亚马逊物流分拣挑战赛任务场景，根据需求确定智能移动作业机器人整体方案。分析目标检测算法发展脉络，选择适当的目标检测算法作为智能抓取机器人的视觉算法原型，同时选取深度学习开源库开发平台。

(2)解析目标检测算法 **SSD** 在 **MXNet** 下的具体实现，详细论述网络训练时的参数设定和特殊技巧。在标准数据集 **Pascal VOC2007** 上测试该算法训练效果，通过分析假阳性样本判断检测器的敏感程度。通过实验得到网络训练较优参数，并探究各特殊技巧对网络训练结果的影响。

(3)面向亚马逊物流分拣挑战赛的特定对象采集图像数据，制作训练数据集。开发制作 **Pascal VOC** 形式的数据集标签工具两套，一套批量处理单目标图像生成训练集，一套手工标注多目标图像生成测试集。在自建数据集上，通过调整超参数及改变训练技巧完成网络微调训练，实验测试和分析训练效果。

2 总体方案与背景知识

2.1 任务场景介绍

为了模拟机器人在大型无人仓中的运作过程，测试其自主完成智能抓取任务情况，我们选取亚马逊物流分拣挑战赛的比赛场景做为既定任务场景。

该比赛场景是从世界各地的真实仓库场景中抽取出的简化版本：抓取机器人完全自主操作，完成从货架上取出目标商品放入相应容器中的系列动作。货架为 **Kiva System** 的原型架，目标商品是预先选定的 24 种在亚马逊上销售的日用产品。每个架子包含 12 个格子，24 件产品分布在各个格子之间。为了适应抓取各种商品的需要，目标商品在选择时考虑了商品大小、形状、轻重、易碎等特殊属性，并进行多样化选取，所选商品包含圆柱形的水杯、较重的小哑铃、易碎的饼干、灯泡等^[8]。

亚马逊物流分拣挑战赛为机器人研究提出了一个极具挑战性的问题，这个比赛任务的完成涉及物体感知、运动规划、抓取控制以及现实工业环境中的任务调配等一系列技术的整合，是仓储自动化的长期目标。

2.2 智能抓取机器人总体方案

针对亚马逊物流分拣挑战赛的任务场景，设计智能抓取机器人的总体方案，主要包括其硬件系统和软件系统。

2.2.1 机器人硬件系统

机器人硬件系统包括机器人本体系统、末端执行装置以及视觉系统，其整体框架如图 2.1 所示。

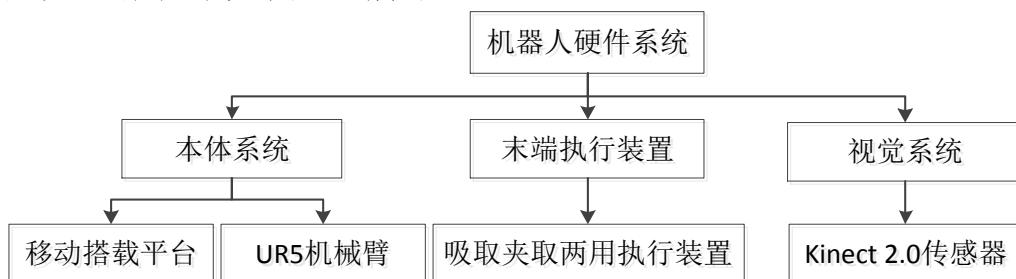


图 2.1 机器人硬件系统整体框架图

2.2.1.1 本体系统

机器人本体系统由一个全方位移动搭载平台和一个六自由度的

UR5 机械臂构成,其中 UR5 机械臂通过固件固定在移动搭载平台上。

机械臂系统采用丹麦 Universal Robot 公司的 UR5 六自由度机械臂,其外观和结构如图 2.2 所示。UR5 机械臂各关节采用模块化设计,电机为科尔摩根的订制中空电机,定子直接铸造为关节,因此关节既充当机器人连接部件,又充当电机定子。自锁部分使用六爪形的机械刹车机构,配合绝对式编码器(雷尼绍的磁式 19bit 单圈绝对编码器)实现。减速器采用德国 HD, 中空方案设计, 转动惯量大。各关节采用串联形式连接,滑环作为连接前后两段的电气接口。UR5 性能参数和各关节的结构参数如下表 2.1 和表 2.2 所示。移动搭载平台可全方位自由移动,并有一定空间用于存放 UR5 机械臂控制柜。



图 2.2 UR5 机械臂

表 2.1 UR5 性能参数表

名称	单位	数值
重复精度	mm	±0.1
温度范围	°C	0 – 50
消耗功率	W	90 – 325, 通常为 150
有效载荷	kg	5
最大工作范围	mm	850
自由度	--	6

表 2.2 UR5 各关节的结构参数

机器人关节位置	活动范围(°)	最快速度(°/s)
底座	±360	±180
肩膀	±360	±180
手肘	±360	±180
手腕 1	±360	±180
手腕 2	±360	±180
手腕 3	±360	±180

对于由移动基座与机械臂构成的移动抓取系统,可以将移动基座

视为一个 3 自由度的 PPR(Prismatic-Prismatic-Rotational)型关节，加上机械臂绕各轴旋转的 6 个自由度，因此该整体具有 9 个自由度，是一个高冗余度的机器人。

2.2.1.2 末端执行装置

针对物品多样性在抓取时的不同需求，末端执行装置专门设计成吸盘吸取和夹爪夹取两种工作方式一体化的结构。其模型结构如图 2.3 所示。吸盘采用气泵驱动，夹取动作的实现为一个连杆机构。整个末端执行装置通过连接件固定在 UR5 机械臂第六轴末端，通过 UR5 的末端 IO 接口传输信号控制其执行相应动作，完成目标商品的抓取。

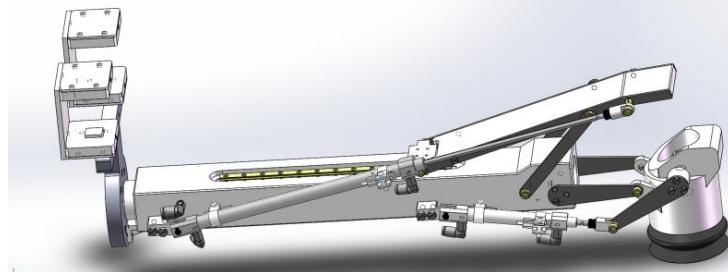


图 2.3 末端执行装置

2.2.1.3 视觉系统

机器人的环境感知与物体识别依赖其视觉系统，主要由固定在机器人身体上方的 Kinect2.0 传感器(如图 2.4)组成。Kinect2.0 有三个镜头，左边的镜头是 RGB 彩色摄像机，中间的是红外线发射器，右侧的是红外线 CMOS 摄影机所构成的广角飞行时间(Time Of Flight, TOF)深度传感器。其各项技术参数如表 2.3 所示。



图 2.4 kinect2.0 RGB-D 传感器

表 2.3 Kinect2.0 技术配置详情

名称	单位	数值
视场	°	70 °水平, 60 °垂直
工作范围	m	0.8-4.0
彩色视频流	--	1920 × 1080 × 16bpp 16:9 YUY2@30fps
深度视频流	--	512 × 424 × 16bpp, 13-bit
主动红外视频流	--	512 × 424 11-bit
配准	--	彩色-深度-主动红外
音频捕捉	--	4 阵列麦克风 48K Hz audio
接口	--	USB3.0
延迟	--	带处理 60ms

2.2.2 机器人软件系统

机器人的软件系统整体构架如图 2.5 所示，软件系统分为四层，分别是环境感知层、运动规划层、执行控制层和任务调度层，各层由不同的子模块构成。

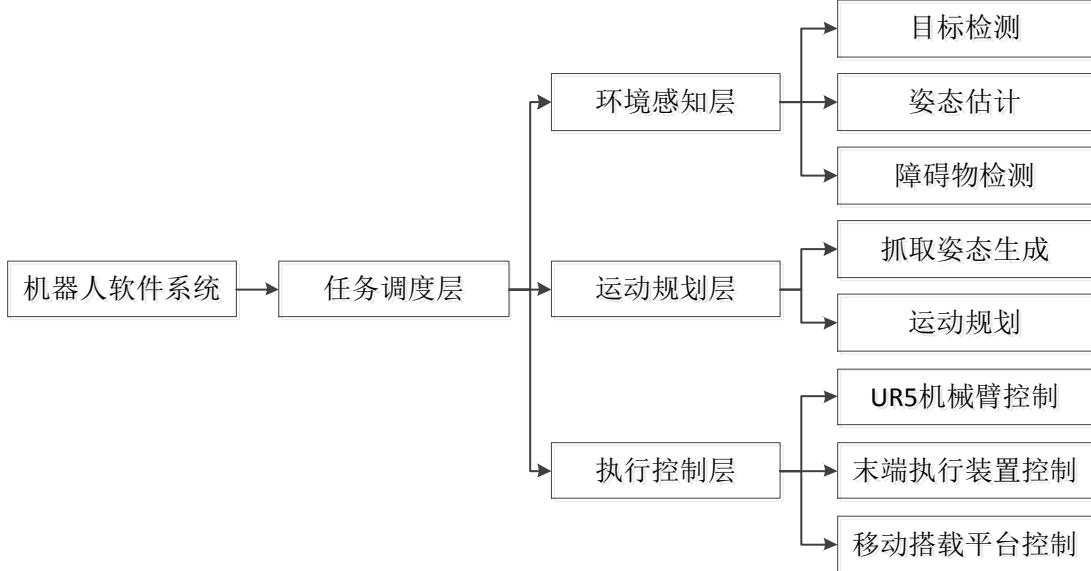


图 2.5 机器人软件系统整体构架

环境感知层由目标检测模块、目标姿态估计模块与障碍物检测模块构成。通过 Kinect2.0 传感器获取的彩色图像进行目标检测，得到视野范围内的目标物体及其相应位置信息，其获取的深度图像计算生成点云。截取目标包围框内的点云，与离线建立的物体点云模型库进行匹配，估计目标物体的当前位姿。

运动规划层包括抓取姿态生成和运动规划两个模块。抓取姿态生成模块根据目标检测与位姿计算结果生成相应抓取姿态，通过逆运动学计算将其转换为构型空间内的目标构型。运动规划模块结合感知层的障碍物检测信息，在起始构型与目标构型之间，规划出一条无碰撞的运动路径。

执行控制层根据规划结果通过 IO 接口驱动 UR5 以一定路径移动到目标位置，驱动末端执行装置执行相应的吸取或抓取操作。然后，驱动臂移出柜子，移动搭载平台移动到储物框处，放置目标物体于储物框中，至此完成一次分拣规程。

任务调度层将任务分解为抓取和放置操作的计划表，以最快完成任务为调度规则，通过一个 ROS SMACH 状态机监控每次拾取和放置操作的执行情况，管理整个任务的运作状态。

由于软件系统包括多种不同的功能模块，为了方便软件系统的实现和维护，利用机器人操作系统(Robot Operating System, ROS)将各功

能模块封装成不同的软件模块。各软件模块间通过数据端口或服务器端口进行通讯，从而使得各模块的具体实现不受平台与编程语言的限制，并可分布在多电脑上运行。

2.3 目标检测算法方案选择

2.3.1 目标检测任务

整个抓取动作的完成，首先要做的就是感知环境，识别环境中的目标分布情况，即完成目标检测任务。本文探讨的目标检测任务为分类并定位预定义类别涵盖的所有可能个体。总的来说，目标检测任务可分为两个关键的子任务：目标分类和目标定位。目标分类任务负责判断输入图像中是否有目标物体出现，输出一系列带分数的标签表明目标物体出现在输入图像的可能性。目标定位任务负责确定输入图像中目标物体的位置和范围，输出物体的包围盒或物体中心点或物体的闭合边界等，通常方形包围盒是最常用的选择。

2.3.2 目标检测算法性能要求

智能抓取任务完成的成功与否，与目标检测算法的三方面性能有密切关系：鲁棒性、准确性和实时性。

鲁棒性：鲁棒性是指目标检测算法能够适应不同的复杂环境，可稳定地得到当前视点下目标的位置和类别信息。在实际工作场景中，光线情况、物体摆放姿态、局部遮挡等多重因素均为检测算法的鲁棒性带来挑战。

准确性：准确性包括两个方面要求，一是目标定位精度高，智能抓取机器人的抓取路径及位姿都与定位的准确性相关联，因此定位精度在很大程度上影响抓取的成败；二是，目标识别准确性好，在物体被随意摆放时，不同视点下观察到的同一物体可能差异巨大，正确的识别物体类别才能达到任务要求。

实时性：工程应用如工业机器人，为了满足工况需求，要求视觉方面的检测速度必须比机械运动速度高一个量级，才能达到准确控制机器人运动的目的。同时，准确性和实时性是一对相互制衡的性能指标，大多实时性能较好的算法都是以牺牲检测精度为代价的。在不降低检测精度的前提下，拥有较高的实时性能是选取目标检测算法的关键^[9]。

2.3.3 实时目标检测算法性能评估指标

实时目标检测算法的效果好坏可以从精度和速度两个方面衡量，

分别对应了性能要求中的准确性和实时性。

2.3.3.1 精度评估指标

在目标检测的实际预测过程中，预测结果与真实结果之间存在四种情况，如表 2.4 所示。

表 2.4 预测结果与真实结果各情况表示

	Positive	Negative
True	True Positive(TP)	True Negative(TN)
False	False Positive(FP)	False Negative(FN)

注：根据预测结果与真实情况是否相符标记结果属性为真或者假(True/False)，根据样本本身是否为目标对象标记样本属性为阴或者阳(Positive/Negative)。由以上标记规则，预测结果统计表示为以下四个指标：(1) 真正 (True Positive, TP): 将正样本预测为正的数量。(2) 真负 (True Negative, TN): 将负样本预测为负的数量。(3) 假正(False Positive, FP): 将负样本预测为正的数量，表现为一种错误类型，即误报。(4) 假负(False Negative, FN): 将正样本预测为负的数量，表现为另一种错误类型，即漏报。

根据表 2.4 各项符号标记所代表的意义，预测结果的准确率(Accuracy)定义为：

$$Acc = \frac{TP + TN}{TP + TN + FP + FN} \quad (2.1)$$

精确率(Precision)定义为：

$$P = \frac{TP}{TP + FP} \quad (2.2)$$

精确率是针对预测结果而言，表示预测为正的样本预测正确的概率。

召回率(Recall)定义为：

$$R = \frac{TP}{TP + FN} \quad (2.3)$$

召回率是针对原来的样本而言，表示样本中正例被预测为正确的概率。

此外还有精确率和召回率的调和均值 F_1 值：

$$\frac{2}{F_1} = \frac{1}{P} + \frac{1}{R} \quad (2.4)$$

$$F_1 = \frac{2TP}{2TP + FP + FN} \quad (2.5)$$

精确率和准确率都很高的情况下， F_1 值也会很高。

对于一个分类器而言，精确率和召回率往往是此消彼长的，仅用任何一个指标判断都对结果的表示不够全面。因此，通常采用平均精度 AP (Average Precision)的统计指标，兼顾两者的结果。AP 的值相当于 Precision/Recall 曲线下方的面积，其计算方法为：将 Precision 看成是 Recall 的分段常数函数，Recall 轴上从 0 到 1 以 0.1 为间隔取 11 个点，叠加其对应的 Precision 值，而后取其平均^[10]。

多标签图像分类(Multi-label Image Classification)任务中，图片的标签不止一个，常常根据所有类别 AP 的平均值(mean Average Precision, mAP)来衡量识别精度的高低。

$$mAP = \frac{\sum_i AP_i}{N_{classes}} \quad (2.6)$$

本文中对检测器检测精度的评估主要采用平均检测精度 mAP 作为判断指标，同时借用准确率、精确率和召回率等指标，辅助判断检测器在训练过程的收敛情况。

2.3.3.2 速度评估指标

检测速度通常采用帧率 (frame per second, FPS) 表示，即每秒处理的图像帧数。若要达到实时检测的目的，则要求对应的检测速度至少为 24FPS 及以上。

2.3.4 目标检测算法常用数据集

目标检测训练的数据来源对训练的结果影响重大，数据集中样本的多样性情况一定程度上决定了训练得到的特征的泛化能力。为了用统一的标准评估不同目标检测算法的性能情况，各国学者建立了许多具有代表性的标准数据集，并针对相应数据集定期举办竞赛，推动物体识别、目标检测、图像分割等计算机视觉关键问题的研究进展。

2.3.4.1 Pascal VOC 数据集

Pascal VOC^[11]是用于图像识别和分类的一套标准化的优秀数据集，广泛用于评估目标类别检测的性能，部分示例图像如图 2.6 所示。Pascal VOC2007 数据库包含 20 个目标类别，每类包含 96-2008 张图像，共 9963 张图片，超过 27000 个目标包围框，图片来源包括 Flickr 等互联网站点以及其他数据库，均为一般尺寸的自然图像。Pascal VOC 标准数据集因为样本图片具有代表性，标注严格受到多数研究人员的青睐。数据集包含 20 个对象类别的边界框注释，使用详尽的遮挡，可见性和五个视点类别标签来扩展注释。Pascal VOC 的图像中每张图像可能包含多个不同类别物体实例，且物体尺度变化很大，因而分类和检测难度都非常大。该数据库的提出，对物体分类与检测的算法提出了极大的挑战，也催生了大批优秀的理论与算法，将物体识别研究推向了一个新的高度。为了便于与其他算法进行性能比较，本文后续的算法性能测试实验将在 Pascal VOC 上进行。

2.3.4.2 ImageNet 数据集

ImageNet^[12]是一个根据 WordNet 层组织起来的数据库，其包含 22000 个类别，每类 500-1000 张图像。其下的 detection 数据集包含 200 个目标类别，400000 张图像，350000 个包围框。ImageNet 广泛

用于训练基础网络，因为 ImageNet 中的数据数量庞大，其训练出来的网络虽然过拟合但是对一般物体具有很强的特征表示能力。同时，由于数据量大，样本种类繁多，遍历网络中所有图片的训练周期长，也给大数据吞吐量下的网络训练带来了极大的挑战。

2.3.4.3 COCO 数据集

COCO^[13]数据集是微软团队获取的一个可以用来图像辨识、分割和配字幕的数据集，包含 91 个目标类别，其最新版本的数据集包含 165482 张训练图像，81208 张验证图像以及 81434 张测试图像。由图 2.7 中 COCO 部分示例图像可以看出，相较于 Pascal VOC，COCO 图像中的对象通常具有小目标的特点，且同一张图中目标数量更多。COCO 数据集的这些特点提升了检测难度，同时也使其成为新一代目标检测任务要攻克的热点对象。

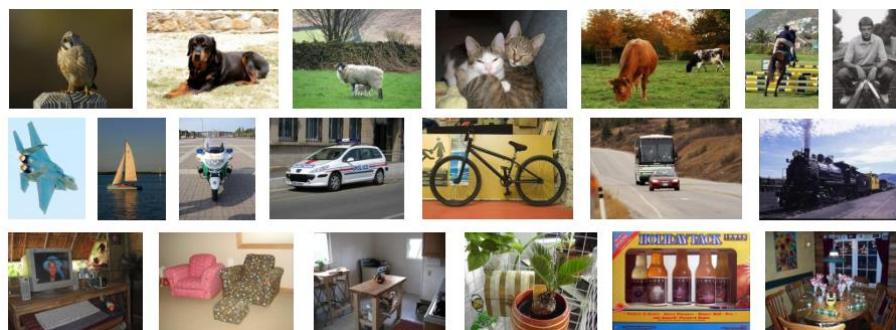


图 2.6 Pascal VOC 示例图片



图 2.7 COCO 示例图片

2.3.5 多种目标检测算法过程分析与性能比较

通过分析多种目标检测算法的检测过程，从鲁棒性、准确性和实时性等方面考量算法性能，比较并选择一种适宜的目标检测算法用于智能抓取移动作业机器人的目标检测环节，实现正确的目标分类和精确的目标定位功能。

2.3.5.1 基于传统机器学习的目标检测算法

传统目标检测方法的检测过程一般分为三个步骤，首先在给定的图像上选择一些候选的区域，然后对这些区域提取特征，最后使用训

练的分类器进行分类。

(1)区域选择：为了定位目标位置，采用设置不同的尺度和长宽比的滑动窗口对整幅图像进行遍历。这种穷举策略虽然包含了目标可能出现的位置，但时间复杂度高，产生冗余窗口多，严重影响后续特征提取和分类的速度。同时，滑动窗口的长宽比一般是预设的几个固定的值，对于长宽比浮动较大的多类别目标检测，即便是遍历也不能得到很好的区域。

(2)特征提取：由于目标的形态多样，光照变化多样，背景多样等因素使得设计一个鲁棒的特征并不是那么容易。然而，提取特征的好坏直接影响到分类的准确性，这个阶段常用的特征有 SIFT^[14]、HOG^[15]等。

(3)分类器：主要有支持向量机(Support Vector Machine, SVM)、自适应增强(Adaboost)等。

总的来说，传统目标检测存在两个主要问题：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。

2.3.5.2 基于深度学习的目标检测算法

随着深度学习的快速发展，其在目标检测上的探索也越来越深入。借助人工智能的方法，目标检测无论在检测精度还是检测速度上都迈上了一个新的台阶。基于深度学习的目标检测算法可分为基于候选区域和基于回归方法两大类，前者对检测精度的提升非常大，后者则致力于简化网络结构，提高实时性能。

R-CNN^[16]算法是基于候选区域的代表算法，其网络框架如图 2.8 所示。主要通过选择性搜索从原始图片中提取 2000 个左右区域候选框，然后利用卷积神经网络提取特征，最后添加两个全连接层，采用支持向量机进行分类，线性回归微调边框位置与大小。其中，每个类别单独训练一个边框回归器。

R-CNN 有很多缺点：(1)虽然摆脱了穷举，但依然有两千个左右的候选框进行 CNN 操作，且其中有不少是重复计算；(2)分类部分采用支持向量机模型(线性模型)，在标注数据不缺的情况下不是最好的选择；(3)训练测试分为多步，候选区域、特征提取、分类、回归都是断开的训练过程，中间数据需要单独保存；(4)训练的空间和时间代价高、速度慢。

卷积神经网络的全连接层要求输入图片的大小一致，R-CNN 采用修剪的方式使其大小归一化，但因此带来了变形偏差问题。而 SPP-net^[17]针对此问题引入空间金字塔池化层^[18](Spatial Pyramid Pooling, SPP)，将维度不一的卷积特征转换为维度一致的全连接输入。

其解决方法是先对整图提取固定维度的特征，再把图片均分成 4 份，每份提取相同维度的特征，再把图片均分为 16 份，以此类推，其过程如图 2.9 所示。因此，无论图片大小如何，提取出来的维度数据都是一致的，然后统一送至全连接层了。SPP-net 总体上策略与 R-CNN 相同，都是候选区域、支持向量机与边框微调结果，因此，R-CNN 存在的问题 SPP-net 也存在。

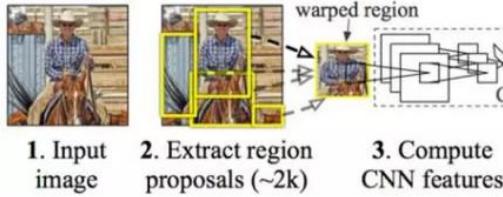


图 2.8 R-CNN 框架图^[16]

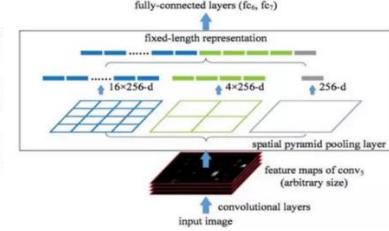


图 2.9 SPP-net 网络结构^[17]

Fast R-CNN^[19]的诞生是为了解决 R-CNN 和 SPP-net 中候选框重复计算的问题。Fast R-CNN 框架结构如图 2.10 所示，其对网络的改进为：(1)使用 ROI(Region of Interest)Pooling 层简化 SPP 层操作。(2)分类和回归用多任务的方式连接在一起，训练和测试不再分多步，不需要存储中间层的特征，梯度通过 ROI Pooling 层直接传播；(3)使用 SVD 分解全连接层的参数矩阵，压缩为两个规模小很多的全连接层。

Fast R-CNN 使用选择性搜索得到候选区域，速度依然不够快。Faster R-CNN^[20]则直接利用候选区域网络 RPN(Region Proposal Networks)来计算候选框。RPN 以一张任意大小的图片为输入，输出一批矩形候选区域，每个区域对应一个目标分数和相应位置信息。Faster R-CNN 中的 RPN 结构如图 2.11 所示。

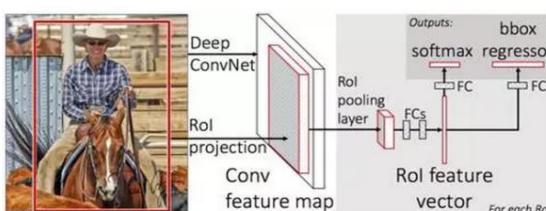


图 2.10 Fast R-CNN 框架^[19]

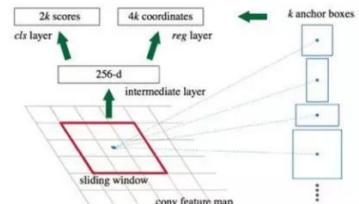


图 2.11 Region Proposal Network(RPN)^[20]

Faster R-CNN 的主要步骤为：首先，将整张图片做为输入，利用 CNN 得到图片的特征层；然后，在最终的卷积特征层上利用 k 个不同的矩形框(Anchor Box)进行提名， k 一般取 9；接着，对每个矩形框对应的区域进行目标/非目标二分类，并用 k 个回归模型(各自对应不同的 Anchor Box)微调候选框的位置与大小；最后进行目标分类。Faster R-CNN 抛弃了选择性搜索，引入了 RPN 网络，使得候选区域、分类、回归共用卷积特征，从而得到了进一步的加速。但是，Faster R-CNN

将分类过程分成了两步，先对两万个 Anchor Box 进行目标判定，即判断是否是目标，再进行目标识别。这种两步法的方式过程不够简洁，一定程度上减缓了目标检测的速度。

YOLO^[21]是基于回归方法的目标检测算法代表，其采用完全不同的思路，把目标判定和目标识别合二为一，在实时性能上有了很大提升。YOLO 的检测速度可达到每秒 45 帧，其快速版 YOLO(Fast YOLO，卷积层更少)甚至可以达到每秒 155 帧。YOLO 追求网络结构的至简，其处理步骤(如图 2.12 所示)为先把输入图片缩放到 480×480 大小，将 480×480 大小的图切成 $S \times S$ 的网格，目标中心点所在的格子则负责该目标的相关检测，每个网格预测 B 个边框及其置信度，以及 C 种类别的概率。

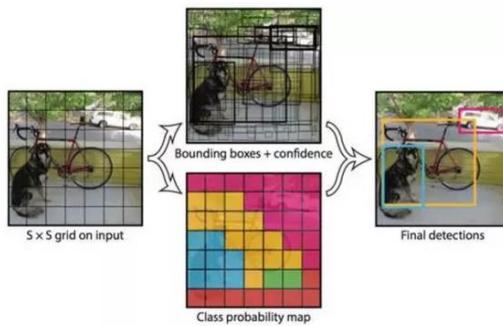


图 2.12 YOLO 模型^[21]

YOLO 简化了整个目标检测流程，速度提升也很大，但是 YOLO 还有不少可以改进的地方，比如 $S \times S$ 的网格就是一个比较启发式的策略，如果两个小目标同时落入一个格子中，模型也只能预测一个；另一个问题是损失函数对不同大小的包围盒未做区分。这些都使得 YOLO 在检测精度上尚有欠缺。

另一种面向实时性能提升的算法为 Single Shot MultiBox Detector(SSD)算法。SSD 算法针对 YOLO 的不足，在 YOLO 的基础上加以改进，构建相对更复杂的网络结构，增加了诸多训练技巧。SSD 算法在保持检测速度为 95FPS 的情况下，采用 300×300 大小的图像在 Pascal VOC2007 上的检测结果仍能达到 74.3% 的平均检测精度，处于较高水平。

2.3.5.3 主流目标检测算法在 Pascal VOC 上的表现

通过查阅资料，统计各种主流目标检测算法在 Pascal VOC 上的表现如表 2.5 所示。

从表 2.5 的统计结果可以得出，SSD 是唯一一个检测平均精确率达到 70% 以上的实时目标检测器。YOLO 虽然检测速度占绝对优势，但平均检测精度仍未达到理想指标。Faster R-CNN 虽有更高的检测精度表现，但其网络的复杂程度决定了它在检测速度上难以达到实时性

能。权衡检测精度与检测速度，选择 SSD 算法作为应用于智能抓取机器人的目标检测算法原型，并将在后续章节详解 SSD 算法的具体实现及训练细节。

表 2.5 主流目标检测算法在 Pascal VOC 上的测试结果

Real-Time Detectors	Train	mAP	FPS
YOLO	07+12	63.4	45
Fast YOLO	07+12	52.7	155
SSD300	07+12	74.3	46
SSD512	07+12	76.8	19
<hr/>			
Less Than Real-Time			
R-CNN BB	07	63.1	--
SPP-net BB	07	66.0	--
Fast R-CNN	07+12	70.0	0.5
Faster R-CNN (VGG-16)	07+12	73.2	7
Faster R-CNN	07+12+COCO	78.8	--
YOLO (VGG-16)	07+12	66.4	21

注：表 2.5 表示各主流目标检测算法在 Pascal VOC 上训练的检测器检测结果统计情况。“07”表示采用 VOC2007 作为训练集；“07+12”表示采用 VOC2007 和 VOC2012trainval 的整合作为训练集。

2.4 深度学习框架选择及相关基本概念

SSD 算法是一种基于深度学习的实时目标检测算法，选择适当的深度学习框架有利于算法实现和提升训练效率。随后，介绍相关深度学习基本概念。

2.4.1 常用深度学习框架

许多深度学习平台的开源，为深度学习的快速普及提供了良好的基础。其中，具有代表性的开源平台有：伯克利大学发起的 DCNN 开源项目 Caffe，Facebook 发起的 Torch，DMLC 发起的 MXNet，微软发起的 CNTK，DBN 等深度学习算法的 python 深度学习项目 Theano，以及谷歌的深度学习开源库 Tensorflow 等。各主流深度学习框架基本情况见附录表 A，从编程语言、灵活性、运行平台等多方面比较以上主流深度学习开源平台，最终选择 MXNet 作为本算法的实现框架。

MXNet 的优点如下：采用 symbolic 接口，使得用户可以快速构建一个神经网络；支持更多语言，目前支持比较好的是 python；支持多卡和多机运行；总体性能上更优。

MXNet 是一个兼具效率和灵活性的深度学习框架。它独创性地将命令式编程和声明式语言两种编程模式融合到一种框架中，命令式

编程上提供张量运算，声明式语言支持符号表达式，用来构建神经网络。其核心是动态依赖调度程序，该程序可以动态自动进行并行化符号和命令的操作。其中部署的图形优化层使得符号操作更快和内存利用率更高，通过自由地混合两种编程模式快速实现设计的网络，并达到效率最大化。该库轻量且便携，甚至能在智能手机上运行诸如图像识别等任务。同时，MXNet 提供一个分布式的键-值存储来支持多设备间的数据交换，可扩展到多个 GPU 和多台主机上。另外，MXNet 对“云计算”友好，直接兼容 S3、HDFS 和 Azure。

2.4.2 深度学习基本概念

2.4.2.1 深度学习与神经网络

深度学习是机器学习的一个分支，是一种基于对数据进行表征学习的方法。本文介绍的 SSD 算法采用监督学习的方式从数据中进行表征学习，即通过已有的一部分输入数据与输出数据之间的对应关系，生成一个函数，将输入映射到合适的输出。

为了构建更好的模型来拟合输入与输出之间的映射关系，采用神经网络模型的层级特征表达来模拟人类视觉系统对信息的分级处理。一个三层神经网络模型如图 2.13 所示，其包含多重非线性变换构成的多个处理层对数据进行高层抽象。图中最左边的原始输入信息称之为输入层，最右边的神经元称之为输出层，中间的叫隐藏层。同时，每一层都可能由单个或多个神经元组成，单个神经元模型如图 2.14 所示，每一层的输出将作为下一层的输入。隐藏层可以为多层，层与层之间是全连接的结构，每一层的各神经元之间没有连接。理论证明，只要拥有足够多的掩藏层，前馈神经网络能以任意精度逼近任意复杂的连续函数。

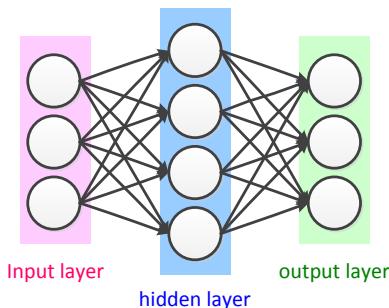


图 2.13 三层神经网络结构模型

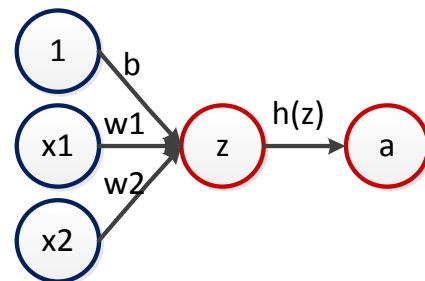


图 2.14 单个神经元模型

2.4.2.2 卷积神经网络

三层以上的神经网络称为深度神经网络，其表达能力相较单层网路更好。卷积神经网络^[22](Convolution Neural Network, CNN)是深度神经网络的一种，它采用局部连接与权值共享的方法区别于传统神经网

络的全连接方式。全连接的方式中，下一层神经元与上一层所有神经元连接，导致网络参数量巨大，网络训练耗时甚至难以训练。与此相比，权值共享网络结构更类似于生物神经网络，减少了权值的数量。同时，对平移、比例缩放、倾斜或者其他形式的变形具有高度不变性，增强了网络的鲁棒性。

在网络连接时，隐藏层的每个神经元仅与特征图的局部图像相连接。所连接的局部图像称为局部感受野，与该局部图像做卷积运算的数据窗口称为卷积核，不同的卷积核对应不同的局部特征，这样需要训练的特征参数数量就大大减少。同时，在图像局部学习到的特征同样适用于图像其余部分，因此，该权值参数可共享给同层的其余神经元，也就是说隐藏层中各神经元的权值参数相同，即所谓的权值共享。

VGG-16 是典型的 CNN 网络，也是图像分类中的一种标准构架，其网络结构如图 2.15 所示。

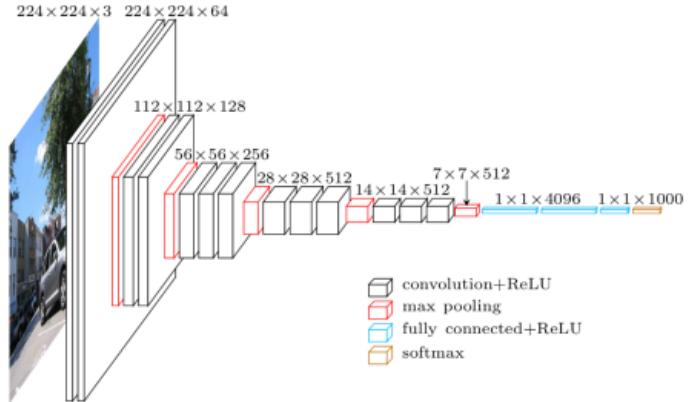


图 2.15 VGG-16 网络结构图

由图 2.15 可以看出，卷积神经网络中主要有两种类型的网络层，分别是卷积层(Convolutions)和池化层(Pooling)。卷积层的作用是提取图像的各种特征；池化层的作用是对原始特征信号进行抽象，从而大幅度减少训练参数，另外还可以减轻模型过拟合的程度^[23]。

在前馈网络中，卷积层是卷积核在上一级输入层上通过逐一滑动窗口计算而得，如图 2.16 所示。输入图像表示成一个数据窗口，若为 RGB 图像则有三个颜色通道数据窗口，每个窗口对应像素点的值表示该颜色通道在该点的亮度值，数值范围为 0~255。卷积核又称为滤波矩阵，其中每一个参数都相当于传统神经网络中的权值参数，与对应的局部像素相连接，将卷积核的各个参数与对应的局部像素值相乘之和，再加上偏置参数，得到卷积层的结果。

池化层，为了进一步降低网络训练参数及模型的过拟合程度，对卷积层进行池化处理(如图 2.17 所示)。池化的方法通常有一下两种：(1)最大值池化，选择池化窗口中的最大值作为采样值；(2)平均值池

化，将池化窗口中的所有值相加取平均，以平均值作为采样值。

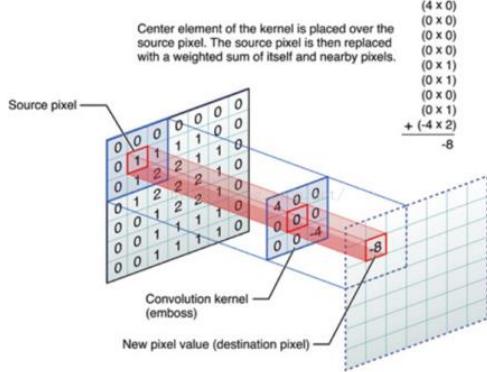


图 2.16 卷积操作

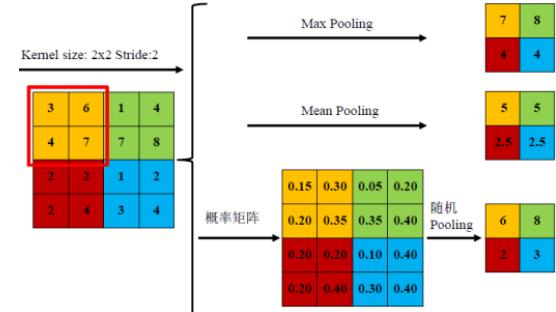


图 2.17 池化操作

2.4.2.3 反向传播算法

对前馈网络输入标准数据集中的图片，其前向传播的结果为：

$$h_{W,b}(x) = h_n(\dots(h_2(h_1(XW^{(1)})W^{(2)})W^{(2)})\dots) \quad (2.7)$$

为了使输入的预测结果更好拟合目标输出，需要不断整定权重参数与偏置参数，即卷积核中的特征参数。拟合结果的好坏程度由损失函数表示，对于单个样例 (x, y) ，损失函数 $J(W, b; x, y)$ 为输出值 $h(x)$ 与目标值 y 之间的均方误差：

$$J(W, b; x, y) = \frac{1}{2} \|h_{W,b}(x) - y\|^2 \quad (2.8)$$

对于一个含 m 个样本的数据集，整体的代价函数 $J(W, b)$ 可以定义为：

$$J(W, b) = \frac{1}{m} \sum_{i=1}^m J(W, b; x^{(i)}, y^{(i)}) + \frac{\lambda}{2} \sum_{l=1}^{n_l-1} \sum_{i=1}^{s_i} \sum_{j=1}^{s_j+1} (W_{ji}^{(l)})^2 \quad (2.9)$$

公式(2.9)中第一项是样例在数据集上的平均误差，第二项是一个权重衰减项，其目的是减少权值的幅度，防止过拟合。

为了获得损失函数的局部最小值，采用反向传播(Back Propagation, BP)算法。反向传播算法的思想是将损失函数 $J(W, b; x, y)$ 看做以权重向量每个元素为变量的高维函数，通过将误差分散到每一层的每个权重上，不断更新权重寻找损失函数的最低点，并按梯度下降的方向更新权值。

在梯度下降法中，每一次迭代都按照公式(2.10)和(2.11)更新 W 和 b ：

$$W_{ij}^{(l)} = W_{ij}(l) - \alpha \frac{\partial}{\partial W_{ij}(l)} J(W, b) \quad (2.10)$$

$$b_i^{(l)} = b_i^{(l)} - \alpha \frac{\partial}{\partial b_i^{(l)}} J(W, b) \quad (2.11)$$

其中的 α 为学习率，上标 l 表示参数所在的层是第 l 层。对损失函数求各个权重参数的偏导数，其算法推导如下：

(1)利用公式(2.7)进行前向传播计算,逐层计算每一层的神经元的激活值,用 $a_i^{n_l}$ 表示,其中 n_l 表示神经元在第 l 层, i 表示神经元在这一层中的序号。

(2)对于输出层(用 n_l 表示)的每个输出单元 i ,根据公式(2.12)计算残差:

$$\delta_i^{n_l} = \frac{\partial}{\partial z_i^{(n_l)}} \frac{1}{2} \|h_{W,b}(x) - y\|^2 = -(y_i - a_i^{(n_l)}) \times f'(z_i^{(n_l)}) \quad (2.12)$$

(3)对于 $l = n_l - 1, n_l - 2, n_l - 3, \dots, 2$ 的各个层,第 l 层的第 i 个节点的残差如下:

$$\delta_i^l = (\sum_{j=1}^{s_{l+1}} W_{ij}^{(l)} \delta_j^{(l+1)}) f'(z_i^{(n_l)}) \quad (2.13)$$

(4)计算需要的偏导数:

$$\frac{\partial}{\partial W_{ij}^{(i)}} J(W, b; x, y) = a_j^{(l)} \delta_i^{l+1} \quad (2.14)$$

通过迭代地使用前向传播和反向传导的步骤,我们能够训练得到需要的网络模型。

2.5 本章小结

本章首先以亚马逊物流分拣挑战赛介绍智能抓取机器人任务场景,接着概述智能抓取机器人的总体方案,包括其硬件系统与软件系统的大体情况。通过归纳总结目标检测算法优劣性能,针对智能抓取任务中的性能需求,权衡精度和速度指标,选择基于深度学习的实时目标检测算法 SSD 为视觉实现的算法原型。最后通过对几种开源深度学习框架的比较,选择在 MXNet 开发平台上做算法实现,并介绍相关深度学习基本概念。

3 基于实验的 SSD 算法实现及参数匹配

本章将通过解析 SSD 算法的网络结构，阐述其在 MXNet 深度学习框架下的具体实现。从精度和速度两方面，测试实现的算法在 Pascal VOC 上的训练效果，并通过实验探究 SSD 算法网络结构以及训练技巧对训练效果的影响。

3.1 实时目标检测算法 SSD 解析

实时目标检测算法 SSD 是一种前向传播的卷积神经网络，其产生一系列固定大小的包围框，以及每一个包围框中包含物体实例的可能性和该包围框的位置微调整，通过非极大值抑制处理后得到最终的预测结果。

3.1.1 SSD 卷积神经网络拓扑结构

SSD 算法所使用的卷积神经网络拓扑结构以 VGG-16 为基础网络，在此基础上又添加了额外的辅助网络。

3.1.1.1 基础网络

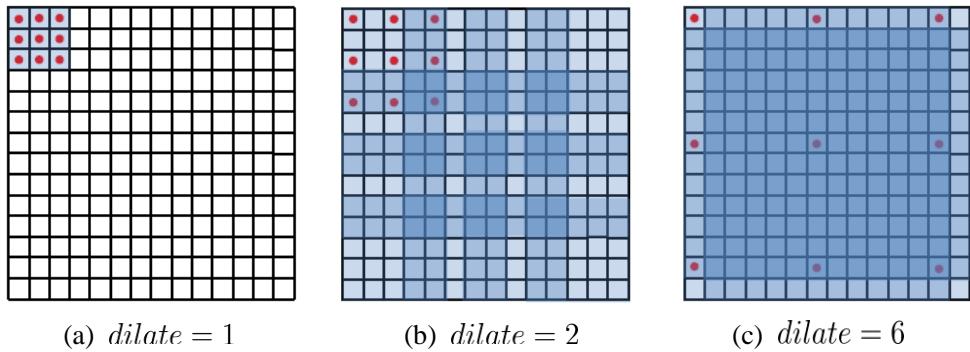
SSD 算法中采用的基础网络为 VGG-16，在保持其整体结构基本不变的情况下，在局部稍作修改，图 3.1 为实际采用的基础网络各层结构及参数情况。

operation	input: 300 × 300 × 3	data size
conv1_1,relu1_1	3 × 3 – s1 – p1,64	300 × 300 × 64
conv1_2,relu1_2	3 × 3 – s1 – p1,64	300 × 300 × 64
pool1	max, 2 × 2 – s2 – p0	150 × 150 × 64
conv2_1,relu2_1	3 × 3 – s1 – p1,128	150 × 150 × 128
conv2_2,relu2_2	3 × 3 – s1 – p1,128	150 × 150 × 128
pool2	max, 2 × 2 – s2 – p0	75 × 75 × 128
conv3_1,relu3_1	3 × 3 – s1 – p1,256	75 × 75 × 256
conv3_2,relu3_2	3 × 3 – s1 – p1,256	75 × 75 × 256
conv3_3,relu3_3	3 × 3 – s1 – p1,256	75 × 75 × 256
pool3	max, 2 × 2 – s2 – p0	38 × 38 × 256
conv4_1,relu4_1	3 × 3 – s1 – p1,512	38 × 38 × 512
conv4_2,relu4_2	3 × 3 – s1 – p1,512	38 × 38 × 512
conv4_3,relu4_3	3 × 3 – s1 – p1,512	38 × 38 × 512
pool4	max, 2 × 2 – s2 – p0	19 × 19 × 512
conv5_1,relu5_1	3 × 3 – s1 – p1,512	19 × 19 × 512
conv5_2,relu5_2	3 × 3 – s1 – p1,512	19 × 19 × 512
conv5_3,relu5_3	3 × 3 – s1 – p1,512	19 × 19 × 512
pool5	max, 3 × 3 – s1 – p1	19 × 19 × 512
conv6	3 × 3 – d6 – p6,1024	19 × 19 × 1024
conv7	1 × 1 – s1 – p0,1024	19 × 19 × 1024
conv8_1,relu8_1	1 × 1 – s1 – p0,256	19 × 19 × 256
conv8_2,relu8_2	3 × 3 – s2 – p1,512	10 × 10 × 512
conv9_1,relu9_1	1 × 1 – s1 – p0,128	10 × 10 × 128
conv9_2,relu9_2	3 × 3 – s2 – p1,256	5 × 5 × 256
conv10_1,relu10_1	1 × 1 – s1 – p0,128	5 × 5 × 128
conv10_2,relu10_2	3 × 3 – s2 – p1,256	3 × 3 × 256
conv11_1,relu11_1	1 × 1 – s1 – p0,128	3 × 3 × 128
conv11_2,relu11_2	3 × 3 – s2 – p0,256	1 × 1 × 256
	output: 1 × 1 × 256	

图 3.1 SSD 基础网络参数图

其中，*pool5*层以1进行边缘填充后采用 3×3 的核进行池化，并以1步幅移动，这样即使经过了一次池化操作，特征图的尺度也没有改变。其后的*conv6*和*conv7*层将全连接操作改成了卷积操作，保持特征图尺度不变，减少训练参数的同时为后面的附加网络保留更多细节特征。

*conv6*层采用了一种特殊的卷积操作，膨胀卷积(Dilate Convolution)^[24]。膨胀卷积可以使用较少的计算来覆盖较大的感受野，而且数据膨胀还可以抑制过拟合的产生。其引入一个新的超参数 *dilate*，表示每隔 *dilate* - 1个像素取一个像素做卷积操作，通过这样的处理使感受野范围指数增加。膨胀卷积的操作过程如图 3.2 所示：



注：通常意义上的卷积操作默认 *dilate* = 1，即如图(a)所示，其感受野大小为 3×3 ；当 *kernel* = 3×3 , *dilate* = 2, *stride* = 3时，其卷积操作如图(b)所示，其感受野大小为 5×5 ；本算法中采用了 *kernel* = 3×3 , *dilate* = 6, *stride* = 1的方式在 12×12 的 feature map 中其表现如图(c)所示，其感受野大小为 13×13 。

图 3.2 膨胀卷积操作示意图

3.1.1.2 辅助网络

SSD 算法中辅助网络的增加使得模型层内部结构更加丰富，其辅助网络结构如图 3.3 所示，该网络设计具有下述两个特点。

其一是采用了多尺度的特征图进行检测。一张图像通常具有两类特征，总体特征和细节特征。小卷积核滤波器用于捕捉细节特征，随着小卷积不断滤波下去，图像尺度减小，相同卷积核对应的感应区域变大，总体特征也通过反向传播被学习到，表现在滤波的卷积核参数上。大部分的模型只采用一个特定尺度的卷积核进行处理，这样在网络前段只有细节特征，后段才慢慢显现出总体特征。

为了将两方面的特征汇聚起来，同时在预测中发挥作用，SSD 模型没有采用加深模型深度的方式，而是把模型加厚。每一次特征图尺度变化前后做更多分析，把不同尺度上的信息都运用到最终的预测中来。因此，辅助网络设计从基础网络上提取不同尺度的特征图，通过 MultiBox 层进行检测。

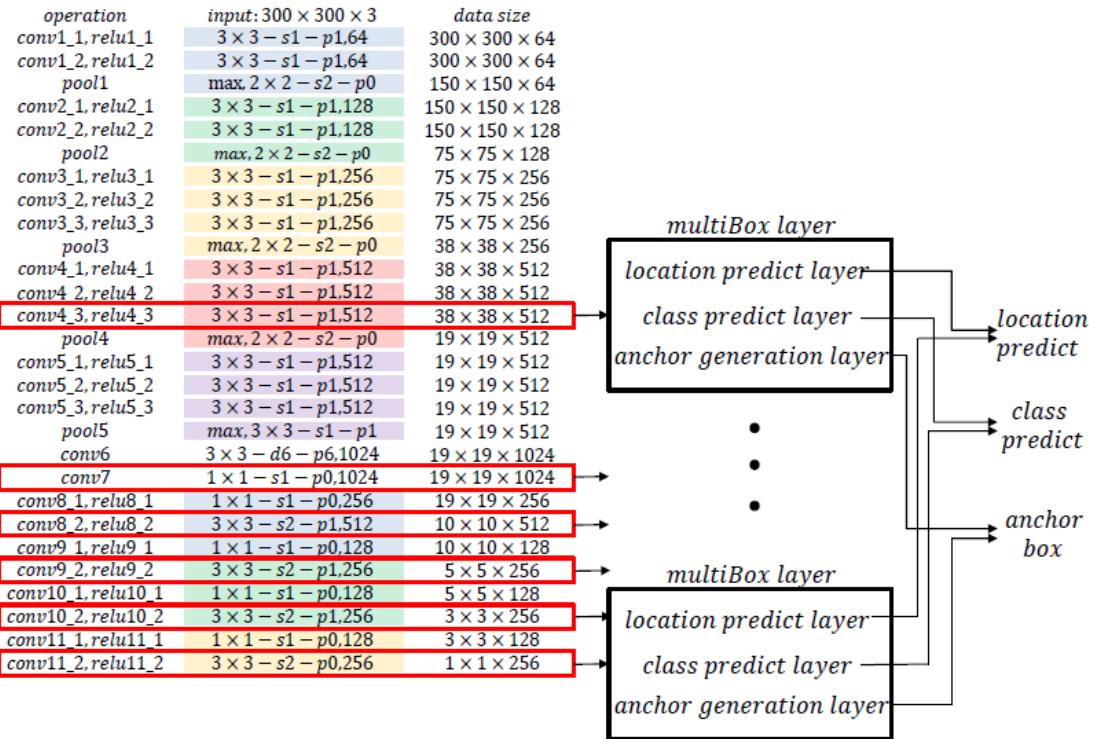


图 3.3 SSD 辅助网络结构图

其二是采用了 MultiBox 层，使用回归的思想，直接在输入图像的多个位置回归出这个位置的目标边框以及目标类别。在设定好提取特征层，每层默认包围框尺寸和纵横比的情况下，MultiBox 层通过三个通道分别在提取的特征层上产生位置预测、所属类别预测以及所有默认包围框的位置信息。假设抽取的特征层为 $conv9_2$ ，该特征层各维度大小为 $5 \times 5 \times 256$ ，设定该层的包围框尺寸为 0.56，纵横比为 $\{1, 2, 3, \frac{1}{2}, \frac{1}{3}\}$ ，则其在中心点(0.5, 0.5)的位置产生的各包围框如图 3.4 所示，其每个包围框都会产生对应的位置预测和所属类别预测结果。

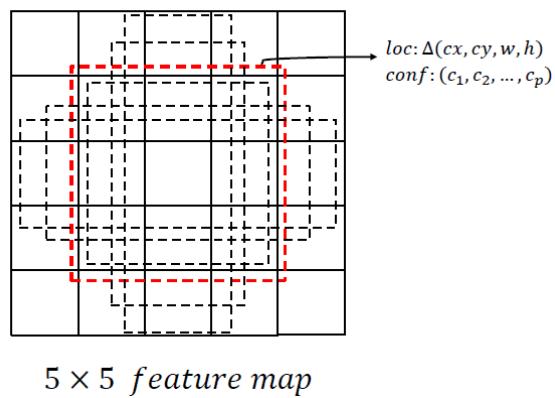


图 3.4 SSD MultiBox 层产生预测示意图

(1)默认包围框位置层，该层根据设定提取的特征层、各层产生包围框的尺寸和纵横比，产生各个特征层上所有包围框的位置信息。

包围框的位置信息以左上和右下角点坐标的形式表示为 $(x_{min}, y_{min}, x_{max}, y_{max})$ 。各层包围框的尺寸依据等差形式分布，当从基础网络中提取 m 层时，第 k 层的包围框尺寸 s_k 计算如下：

$$s_k = s_{min} + \frac{s_{max} - s_{min}}{m-1}(k-1), k \in [1, m] \quad (3.1)$$

设各层对应的纵横比 $r_a \in \{r_1, r_2, \dots, r_k\}$ 如 $r_a \in \{1, 2, 3, \frac{1}{2}, \frac{1}{3}\}$ ，则可计算出包围框的长宽分别如下：

$$w_k^a = s_k \sqrt{r_a} \quad (3.2)$$

$$h_k^a = \frac{s_k}{\sqrt{r_a}} \quad (3.3)$$

对于纵横比为1的情况，增加了一个包围框，其尺寸 s'_k 计算如下：

$$s'_k = \sqrt{s_k s_{k+1}} \quad (3.4)$$

记抽取特征层第 k 层的尺寸为 f_k ，则归一化后各包围框中心点的坐标 (cx_i, cy_j) 计算如下：

$$cx_i = \frac{i + 0.5}{f_k}, i \in [0, f_k) \quad (3.5)$$

$$cy_j = \frac{j + 0.5}{f_k}, j \in [0, f_k) \quad (3.6)$$

故

$$x_{min} = cx_i - \frac{w_k^a}{2} \quad (3.7)$$

$$y_{min} = cy_j - \frac{h_k^a}{2} \quad (3.8)$$

$$x_{max} = cx_i + \frac{w_k^a}{2} \quad (3.9)$$

$$y_{max} = cy_j + \frac{h_k^a}{2} \quad (3.10)$$

(2)位置预测层，对于任一抽取的大小为 $m \times m$ ，有 p 个通道的特征层，使用 a 个 3×3 的卷积核进行卷积滤波，产生预测包围框相对于该位置上的默认包围框的位置偏移量，表示为 $\Delta(cx, cy, w, h)$ ，以微调预测出的包围框坐标信息。因每个包围框需预测上下左右4个方向的偏移量，故卷积核的数量 a 为该特征层各位置上默认包围框数量的4倍。

$$a = N_{default\ box} \times 4 \quad (3.11)$$

(3)所属类别预测层，与位置预测相似，为了预测该位置上每个默认包围框所属类别的可能性，使用 b 个 3×3 的卷积核进行卷积滤波，同时将背景也当作种类之一进行预测，其结果表示为 (c_1, c_2, \dots, c_p) 。

$$b = N_{default\ box} \times (N_{classes} + 1) \quad (3.12)$$

3.1.2 SSD 网络训练及相关技巧

3.1.2.1 匹配策略

监督学习训练的关键是将人工标注的标签与预测的结果相比较，用其差值反向传播得到更新参数。因此，需要将真实包围框(即人工标注的包围框)映射到特征图生成的若干默认包围框中去。

SSD 网络中的匹配策略遵照下述方法。对于每一个真实包围框，首先从特征图生成的若干默认包围框中挑选与其匹配度最高的一个。匹配的优劣程度以 Iou 值表示。 Iou 的计算方法如图 3.5 所示。

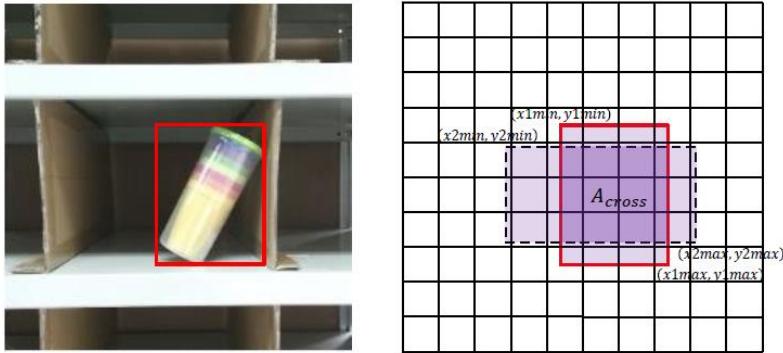


图 3.5 Iou 计算原理示意图

$$A_{cross} = (\min(x1_{max}, x2_{max}) - \max(x1_{min}, x2_{min})) \\ \times (\min(y1_{max}, y2_{max}) - \max(y1_{min}, y2_{min})) \quad (3.13)$$

$$A_{union} = (x1_{max} - x1_{min}) \times (y1_{max} - y1_{min}) + (x2_{max} - x2_{min}) \times (y2_{max} - y2_{min}) - A_{cross} \quad (3.14)$$

$$IoU = \frac{A_{cross}}{A_{union}} \quad (3.15)$$

然后，将每个未匹配的默认包围框与真实包围框配对，标记所有 Iou 大于阈值(通常为 0.5)的默认包围框为正样本。这样同一真实包围框可能有多个默认包围框与其匹配，即使这些包围框不是匹配度最高的，预测出任意一个都为找到目标，增加了包围框被预测出来的概率。

3.1.2.2 正负样本平衡

在匹配后，大部分的默认包围框将被标记为负样本，尤其是当默认包围框数量很大时，正负样本的数量不平衡表现得更加明显。这种不平衡将会增加优化的难度并影响模型的稳定性。为了平衡这种数量差，根据 Iou 从高到低排序，在 Iou 高的负样本中选取与正样本成一定数量比例的样本做标记，该比例为负样本采样比例，以此在易错区域区分样本正负。

匹配标志 $x_{ij}^p = \{1, 0\}$ 表示第 i 个默认包围框与第 j 个真实包围框的匹配情况和预测类别情况的联合结果，当两者符合匹配策略并被预

测为类别 q 时，即记为正样本，则 $x_{ij}^p = 1$ ，否则 $x_{ij}^p = 0$ 。

3.1.2.3 数据增广策略

由于深度网络的训练需要从大量图像中学习泛化特征，如果原始图像数据集包含有限的训练图像，通过数据增广策略可以加强模型的鲁棒性，适应不同尺寸大小的输入情况。因此，数据增广已经成为训练深度网络必须做的事情。

常用的数据增广策略包括水平翻转、图像旋转、随机裁剪、颜色亮度调整等，也可以将几种不同的处理方式组合起来，比如同时旋转和随机缩放。下图 3.6 所示为原始图像与经数据增广后的图像的比较：

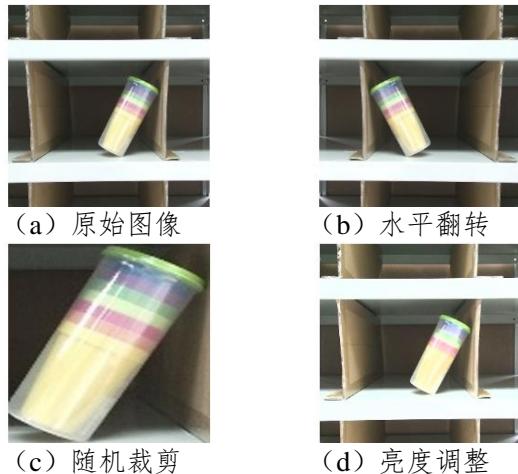


图 3.6 原始图形与数据增广后的图像比较

SSD 中对数据增广采取以下策略：

- (1) 使用整张图片；
- (2) 使用一些从原图中以大小在 $[0.1, 1]$ 之间，相应的宽高比在 $[\frac{1}{2}, 2]$ 之间裁剪一些图像块，这些图像块与目标物体之间的 Iou 为 $0.1, 0.3, 0.5, 0.7, 0.9$ ；
- (3) 随机裁剪一个图像块；
- (4) 当真实包围框的中心在采样的图像块中时，保留重叠部分。
采样步骤之后，每一个采样的图像块被统一缩放到固定大小，并以 0.5 的概率随机水平翻转。

3.1.2.4 参数初始化

深度学习中的权重初始化对模型收敛速度和训练出的模型质量有重要的影响。当以随机的小权重初始化网络时，逐层观察梯度变化可以发现梯度值逐层衰减，反向传播后返回到第一层的梯度值太小以至于不能得到有意义的权值更新，因此无法有效地学习和更新得到一个好的深度模型。

我们采用 Xavier 算法对新增的层进行权重初始化，Xavier 初始化的基本思想是保持输入和输出的方差一致，避免所有输出值都趋向

于 0。保持信号通过多层网络后强度仍在一个合理的范围内，使信号深入网络。记 W 为神经元初始化时的参数集合，采用 Xavier 算法初始化权值，使其分布为高斯分布或均匀分布，且呈零均值，方差符合

$$Var(W) = \frac{2}{n_{in} + n_{out}} \quad (3.16)$$

假设输入为 $X = [x_1, x_2, \dots, x_n]$ ，随机权重 $W = [w_1, w_2, \dots, w_n]$ ，则输出 Y 可表示为

$$Y = WX = w_1x_1 + w_2x_2 + \dots + w_nx_n \quad (3.17)$$

可计算出 Y 的方差符合

$$Var(Y) = E[x_i]^2 Var(w_i) + E[w_i]^2 Var(x_i) + Var(w_i)Var(x_i) \quad (3.18)$$

若输入和权重都符合零均值，则公式(3.18)可简化为

$$Var(Y) = nVar(w_i)Var(x_i) \quad (3.19)$$

由此可见，输出的方差是输入的方差的 $nVar(w_i)$ 倍缩放。若希望输入方差与输出方差相等，则要求

$$nVar(W_i) = 1 \quad (3.20)$$

即

$$Var(W_i) = \frac{1}{n} = \frac{1}{n_{in}} \quad (3.21)$$

同理，以相同的步骤针对反向传播信号进行推导，可得到

$$Var(W_i) = \frac{1}{n_{out}} \quad (3.22)$$

为了保持输入梯度和输出梯度的方差相同，则需满足同时满足(3-21)(3-22)两个条件，故要求

$$n_{in} = n_{out} \quad (3.23)$$

输入神经元数与输出神经元数不常相同，作为妥协，故取两者的平均值作为权重分布的方差，即

$$Var(w_i) = \frac{2}{n_{in} + n_{out}} \quad (3.24)$$

3.1.2.5 损失函数

SSD 训练的损失函数源自于 MultiBox 的损失函数，但在此基础上加以拓展，使其可以处理多个目标类别。目标损失函数 $L(x, c, l, g)$ 是位置损失 $L_{loc}(x, l, g)$ 和置信损失 $L_{conf}(x, c)$ 的加权综合。

$$L(x, c, l, g) = \frac{1}{N}(L_{loc}(x, l, g) + \alpha L_{conf}(x, c)) \quad (3.25)$$

其中， α 为权重系数，一般设置为 1， N 为匹配的默认包围框数，若 $N = 0$ ，则记 $L(x, c, l, g) = 0$ 。

$L_{loc}(x, l, g)$ 是预测包围框(l)和真实包围框(g)各参数之间的SmoothL1损失，其具体表达形式如下：

$$L_{loc}(x, l, g) = \sum_{i \in Pos}^N \sum_{m \in \{cx, cy, w, h\}} x_{ij}^p smooth_{L1}(l_i^m - \hat{a}_j^m) \quad (3.26)$$

$$\begin{aligned} \hat{g}_j^{cx} &= \frac{g_j^{cx} - d_i^{cx}}{d_i^w} & \hat{g}_j^{cy} &= \frac{g_j^{cy} - d_i^{cy}}{d_i^h} \\ \hat{g}_j^w &= \log\left(\frac{g_j^w}{d_i^w}\right) & \hat{g}_j^h &= \log\left(\frac{g_j^h}{d_i^h}\right) \end{aligned} \quad (3.27)$$

其中， \hat{g}_j^m 中的各变量按照上述方法以归一化后的形式带入计算。
置信损失 $L_{conf}(x, c)$ 的计算方法如下：

$$L_{conf}(x, c) = - \sum_{i \in Pos}^m x_{ij}^p \log(\hat{c}_i^p) - \sum_{i \in Neg} \log(\hat{c}_i^0) \quad where \quad \hat{c}_i^p = \frac{\exp(c_i^p)}{\sum_p \exp(c_i^p)} \quad (3.28)$$

3.2 目标检测算法训练参数设置

3.2.1 批量大小

在深度学习中，一般采用小批量随机梯度下降进行优化，即每次迭代在训练集中取批量大小(batch size)数量的样本进行优化。因此，批量大小与遍历整个数据集的迭代次数成反比。

从算法角度来说，在随机梯度下降中，使用更大的批量可以减少随机梯度更新的方差(通过采用小批量的梯度下降平均值)，这也意味着要采用更大的步长，即优化算法将以更快的速度进入局部最优。一般来说，在一定范围内，批量越大，该批迭代时确定的下降方向越准，引起训练的震荡越小。

基于随机梯度下降的训练，过大的批量会使网络很容易收敛到一些不好的局部最优点。同样，太小的批量大小也存在训练速度慢，训练不容易收敛等问题。因此，批量大小不同会引起训练速度和优化结果的差异，通常批量大小取2的幂次方，如16、32、64、128等，选择一个适当的批量需要不断实验。在接下来的实验中，若不多加设定，则采用批量大小为32进行。

3.2.2 学习速率

运用梯度下降算法进行优化时，权重更新规则中，在梯度项前乘以一个系数，该系数为学习速率 η (learning rate)。由反向传播算法的推导过程不难理解，学习速率过小，会导致收敛速度很慢；学习速率

过大，则会阻碍收敛，即在极值点附近出现振荡。因此，选择一个合理的学习速率很难，往往需要在训练过程中随梯度下降的优化而适应性变化学习速率以提高性能并减少训练时间。

在训练过程中，最简单常用的学习速率适应策略是随迭代次数减小学习速率，即学习速率调度。学习速率在间隔若干个迭代周期后衰减一个较小的阈值。这有利于在训练程序开始时使用较大的学习速率值，使权重有较大的改变，并在训练靠后阶段(已接近局部最小值)，降低学习速率，使得权重随迭代有较小的更新，慢慢靠近局部最小值。接下来的实验中，我们选择学习速率为 0.001，并以每 50 个迭代周期更新一次学习速率，每次更新衰减学习速率为上次的 0.8 倍。

3.2.3 动量

动量(momentum)方法旨在加速学习过程，特别是处理高曲率、小但一致的梯度，或是带噪声的梯度时。动量算法积累了之前梯度指数级衰减的移动平均，并且继续沿该方向移动。在更新模型参数时，对于当前梯度方向与上一次梯度方向相同的参数进行加强，即在这些方向上梯度下降更快；对于当前梯度与上一次梯度方向不同的参数进行削弱，因此可以获得更快的收敛速度与较少的振荡。本文实验中设置动量参数为 0.9。

3.2.4 权重衰减

权重衰减(weight decay)即进行 L_2 正则化，进行正则化的目的是减小权重的幅度，防止产生过度拟合。权重衰减参数用于控制损失函数中均方差项与正则化项的相对重要性，本文所述实验采用 0.0001 为权重衰减参数。

3.3 SSD 算法的参数网络训练实验及结果分析

3.3.1 SSD 算法在 Pascal VOC2007 上的训练测试结果

为了将自己的训练结果与其他算法的训练结果相比较，选择在 Pascal VOC 标准数据集上进行参数网络训练和结果测试。

通过加载预训练模型 VGG-16，以学习率 0.001，动量 0.9，权重衰减 0.0001，每 50 次迭代更新一次权重，每次权重的更新因子为 0.8，在 Pascal VOC2007 和 2012 训练验证集上进行训练，通过 150 个迭代周期后测试其训练效果。其训练过程如图 3.7 所示。

图 3.7 为训练过程中每个迭代周期的精确率、召回率以及 SmoothL1 的变化情况。从图中可以看出，精确率和召回率随迭代周

期先急剧增加然后逐渐稳定，局部振荡，*SmoothL1*损失值随迭代周期先极具减小然后逐渐平稳。由此可得，整个学习过程是收敛的，但已经出现过拟合现象，即训练得到的模型习得过多训练数据的细节从而降低了其泛化能力，主要表现在训练集精确率仍在增加但验证集的各项指标局部震荡、基本不变。

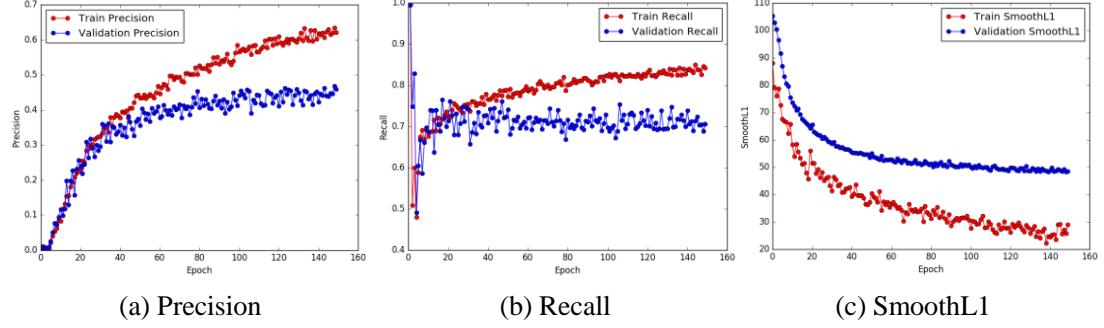


图 3.7 SSD 在 Pascal VOC07+12 上训练时的精确率、召回率、准确率和损失函数变化情况

迭代 150 个周期后，在 Pascal VOC2007 测试集上进行预测实验，自主训练的检测器检测结果如下表 3.1 所示。

表 3.1 Pascal VOC2007 测试检测结果

Method	Data	mAP	aero	bike	Bird	boat	bottle	bus	car	cat	Chair	cow
Fast ^[19]	07	66.9	74.5	78.3	69.2	53.2	36.6	77.3	78.2	82.0	40.7	72.7
Fast ^[19]	07+12	70.0	77.0	78.1	69.3	59.4	38.3	81.6	78.6	86.7	42.8	78.8
Faster ^[20]	07	69.9	70.0	80.6	70.1	57.3	49.9	78.2	80.4	82.0	52.2	75.3
Faster ^[20]	07+12	73.2	76.5	79.0	70.9	65.5	52.1	83.1	84.7	86.4	52.0	81.9
SSD300	07	68.0	73.4	77.5	64.1	59.0	38.9	75.2	80.8	78.5	46.0	67.8
SSD300	07+12	71.6	72.7	81.6	71.3	62.4	37.8	84.6	81.4	83.2	48.5	71.3
SSD512	07	71.6	75.1	81.4	69.8	60.8	46.3	82.6	84.7	84.1	48.5	75.0
SSD512	07+12	76.8	82.4	84.7	78.4	73.8	53.2	86.2	87.5	86.0	57.8	83.1
			Table	dog	horse	mbike	person	plant	Sheep	sofa	Train	tv
Fast ^[19]	07	67.9	79.6	79.2	73.0	69.0	30.1	65.4	70.2	75.8	65.8	
Fast ^[19]	07+12	68.9	84.7	82.0	76.6	69.9	31.8	70.1	74.8	80.4	70.4	
Faster ^[20]	07	67.2	80.3	79.8	75.0	76.3	39.1	68.3	67.3	81.1	67.6	
Faster ^[20]	07+12	65.7	84.8	84.6	77.5	76.7	38.8	73.6	73.9	83.0	72.6	
SSD300	07	69.2	76.6	82.1	77.0	72.5	41.2	64.2	69.1	78.0	68.5	
SSD300	07+12	71.1	83.1	82.2	79.2	74.7	47.6	72.0	73.9	82.6	71.1	
SSD512	07	67.4	82.3	83.9	79.4	76.6	44.9	69.9	69.1	78.1	71.8	
SSD512	07+12	70.2	84.9	85.2	83.9	79.7	50.3	77.9	73.9	82.5	75.3	

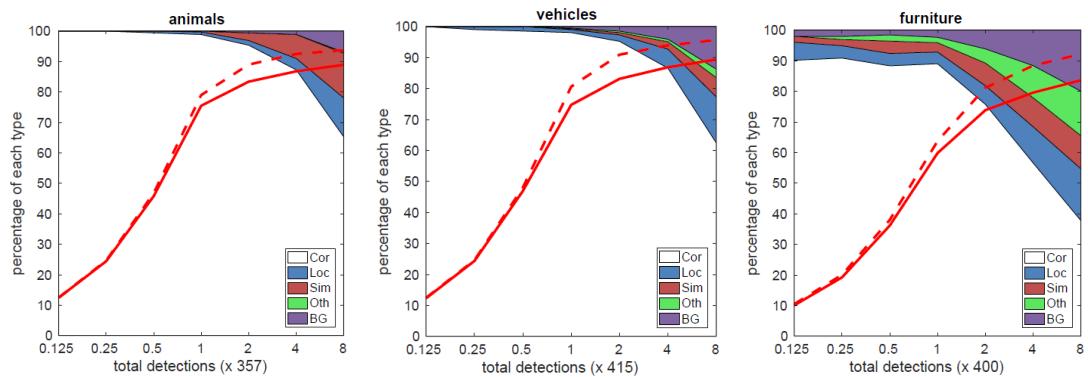
注：表 3.1 表示各已训练检测器在 Pascal VOC2007 测试集上的测试结果统计。其中，Fast 与 Faster R-CNN 使用的输入图像最小尺度为 600；SSD300 代表训练时的输入图像尺寸为 300×300 ，SSD512 代表训练时的输入图像尺寸为 512×512 。数据列中，“07”代表采用 VOC2007trainval 作为训练集，“07+12”表示采用 VOC2007 和 VOC2012trainval 的整合作为训练集。

迫于训练检测器的实验周期长，毕业设计期间时间紧张，故仅自主训练了 SSD300(07)和 SSD300(07+12)两个检测器，并测试其训练结果。由表中数据可以看出，对 SSD 检测器而言，输入图像尺寸更大训练出的效果更好，大尺寸图像可获取更多信息用于特征提取，同时，训练样本数量的增加也有助于效果的提升。

为了更深层次地分析 SSD 卷积神经网络模型的检测结果，使用检测工具 Detection-Analyze 进行辅助分析^[25]。SSD 算法在 Pascal VOC2007 测试集上的测试结果累积统计情况如图 3.8 所示，这里展示

其分别在动物、车辆以及家具类别上的表现。

测试结果的累积统计结果表明，总体上，SSD 算法对各种类别的检测质量都较高(大面积的白色区域)，召回率在 85%-90%之间，采用宽松的召回标准时检测结果的正确率甚至更高。与 R-CNN 相比，SSD 的定位失误更少，主要原因采用了直接回归目标边界框的形状和分类物体类别的策略。但是，SSD 受相似对象的混淆(Sim)影响严重，可推测原因为对多类别目标共享了同一位置。因此，SSD 检测器对相似类别的类间差异敏感程度有待提高。各类别之间，SSD 对家具类别的检测结果正确率相对差些，且其受背景或未标记对象的混淆是产生检测失误的最突出原因。推测可能是由于家具类别的类间差异较大，提取泛化特征较困难所导致的结果。



注：(Cor)代表检测为正确的结果，其余为不同原因引起检测结果假阳性的累积统计，(Loc)代表由于定位不准确引起，(Sim)代表受相似类别的混淆，(Oth)代表受其他 VOC 对象的混淆，(BG)代表与背景或未标记对象混淆。红色的实线代表采用严格的召回标准(与真实边界框之间的重叠阈值为 0.5)时，召回率随检测次数增加的变化情况。红色的虚线是采用较宽松标准的召回标准(与真实边界框之间的重叠阈值为 0.1)时，召回率随检测次数增加的变化情况。

图 3.8 SSD300 在动物、车辆和家具类别上的测试结果可视化情况

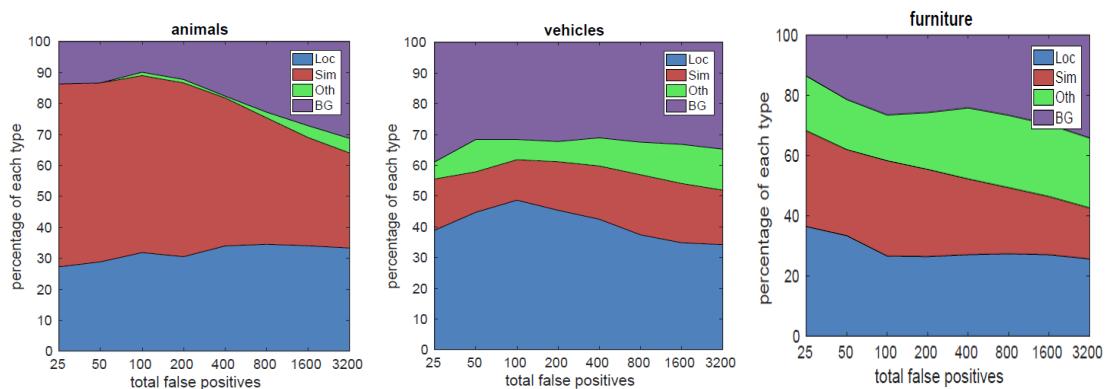
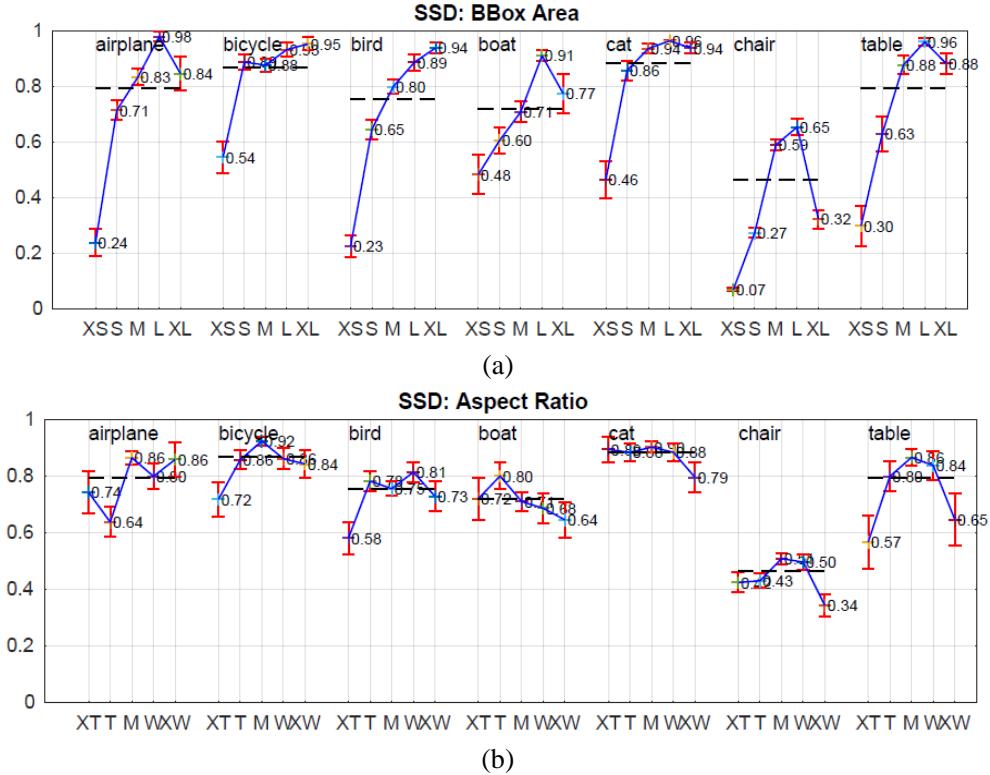


图 3.9 SSD300 测试结果的假阳性样本分布

图 3.9 为 SSD300(07+12)在 Pascal VOC2007 测试集上的检测结果假阳性样本分析。由表中的假阳性样本分布可以看出，动物类别的检测受相似类别影响最大，诸如将猫检测为狗等误判情况。汽车类别则

主要受背景或未标记类别的混淆以及由于定位不准确引起检测失误。针对假阳性样本的分析结果，针对其检测错误情况，可以通过先验知识改进检测器。



注：检测器对不同目标对象特性的敏感程度以及受其影响大小。(a)图表示每个类别包围框面积的影响，(b)图表示各类别纵横比的影响。包围框面积中，XS=extra-small，S=small，M=medium，L=large，XL=extra-large。不同纵横比中：XT=extra-tall/narrow，T=tall，M=medium，W=wide，XW=extra-wide。

图 3.10 SSD300 检测器对不同对象特性的敏感程度

从图 3.10(a)中可以看出，训练得到的 SSD300 检测器对包围框面积小的对象的检测效果很不好，主要由于小目标在卷积的最后几层中几乎没有多少信息，因此加强对小目标的检测是提高检测性能的一个方向。另外，从图 3.10(b)可以看出，SSD300 检测器对不同纵横比的对象具有很强的鲁棒性，不难推断，这主要归功于在网络结构中采用多个默认层设置多种不同纵横比的包围框来定位目标位置的策略。

3.3.2 SSD 训练的特殊技巧对检测精度的影响分析

3.3.2.1 多尺度特征图对检测精度的影响

SSD 检测精度相较于 YOLO 的提升，一个主要的原因为采用了多尺度特征图，综合运用各层次的结果进行最终预测。每个特征图产生对应尺度和一定数量的包围框，特征图产生的包围框数量计算如上图 3.11 所示。

为了衡量采用多尺度特征图对检测结果的影响程度，我们设计如下实验，每次减少若干参与最终预测的特征层，训练网络并在 Pascal VOC2007 测试集上测试平均检测精度。实验结果统计如表 3.2。

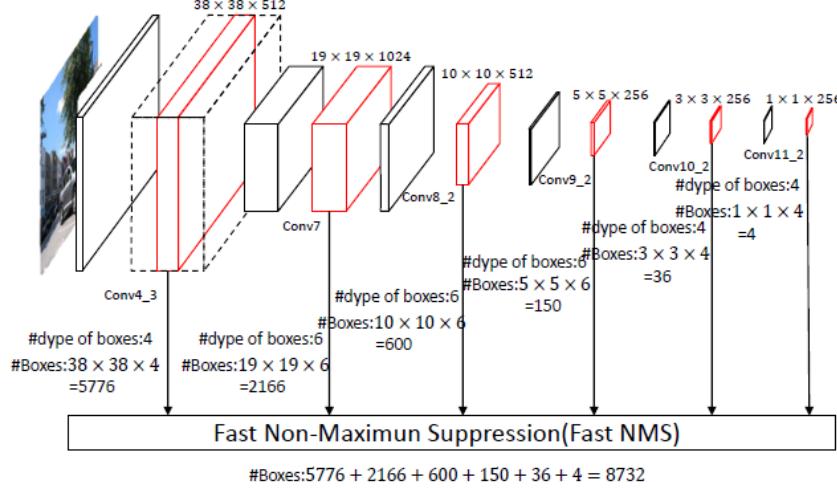


图 3.11 多尺度特征图中包围框数量的计算图示

表 3.2 从整体上看，采用更多的特征图参与最终预测能够提高预测的平均精度。

表 3.2 采用不同特征图进行训练的检测精度统计

Prediction source layers from:						mAP	#Boxes
Conv4_3	Conv7	Conv8_2	Conv9_2	Conv10_2	Conv11_2		
✓	✓	✓	✓	✓	✓	71.61	8732
✓	✓	✓	✓			69.45	8692
✓	✓					61.82	7942
		✓	✓	✓	✓	59.05	790

实验组别两两对比，第二组与第一组实验条件比较，仅去掉 *conv10_2* 和 *conv11_2* 层，平均检测精度下降了 2.16%，该两层特征图尺度小，产生的包围框数量少，因此对最终检测精度影响较小。第三组与第一组实验比较，仅采用 *conv3_4* 和 *conv7* 层进行最终预测，在包围框总数保持 8000 个左右的时候，检测精度降低 9.79%，主要原因可归结为，该两层产生的包围框面积较小，对于较大目标，由于裁剪包围框缺失信息而降低了预测精确率。第三组与第四组实验比较，仅用后面四层 (*conv8_2*/*conv9_2*/*conv10_2*/*conv11_2*) 进行预测时，产生的包围框数量迅速减少，仅为 790 个，但其平均检测精度相较第三组数据相差不大，说明采用较大的边界框能够包含的目标内容更多，虽然使用的特征图更粗糙，但却显现更多的高级特征，便于预测。第四组与第一组结果对比，由于第四组缺少了高分辨率的特征图导致检测效果大打折扣，说明了高分辨率的更精细的特征图对检测结果的影响很大。因此，最好的方式就是结合各层次的特征，将精细特征与高级特征都考虑到最终的预测中去，才能得到更高的精度水平。

3.3.2.2 SSD 模型训练特殊技巧对检测精度的影响实验

为了衡量 SSD 模型中运用的其他特殊技巧对检测精度的影响，通过控制变量的方式，设计实验如表 3.3 所示。

表 3.3 训练时不同技巧对检测精度的影响

	SSD300			
More data augmentation	✓	✓	✓	✓
Dilate convolution	✓	✓	✓	✓
Include $\{\frac{1}{2}, 2\}$ box or not	✓	✓	✓	✓
Include $\{\frac{1}{3}, 3\}$ box or not	✓	✓		✓
VOC2007 test mAP	62.18	71.44	69.69	67.59
				71.61

从测试结果中可以看出，采用数据增广策略对网络训练来说至关重要，缺少数据增广的实验组检测平均精度直接下降 9.43%。不同于 Fast R-CNN 和 Faster R-CNN 在数据增广时仅使用水平翻转，SSD 的数据增广策略中使用了随机采样的方式来增加数据量。但此种数据增广策略是否对其他模型也存在卓著的效果，结果未可知，因为诸如 Fast R-CNN 与 Faster R-CNN 在分类过程中的池化操作已具备相类似的作用。

膨胀卷积对检测精度的提升微乎其微，但是有文献中曾提到，膨胀卷积的处理结果能够使得检测速度上有近 20% 的提升^[26]，具体情况可后续通过实验进行论证。

默认包围框的形状多样对检测精度提高有一定效果，第三、四组实验分别去掉了纵横比为 $\{\frac{1}{2}, 2\}$ 和 $\{\frac{1}{3}, 3\}$ 两种形状的包围框，其相应的检测精度对比第五组结果都有不同程度的降低，分别下降了 1.92% 和 4.02%。由此看来，采用各种不同纵横比的包围框使得精确预测更容易。

3.3.2.1 初始化策略对检测精度的影响

训练开始前，加载预训练模型的权重参数，初始化新增层进行网络训练。对于新增层的初始化关系到整体的网络训练速度以及训练的结果情况。为了衡量初始化策略对训练结果的影响，我们设计对照实验，如图 3.13 所示。一组采用 Xavier 算法初始化新增层，另一组根据均值为 0，方差为 0.06 的均匀分布初始化新增层的权重参数。经过 150 次迭代，跟踪训练过程，可以看出采用随机初始化的网络在训练时是发散的，未出现收敛现象。其余多种初始化策略，如 MSRAPrelu 初始化、LSTMbias 初始化等，均由于时间有限尚未进行实验测试效果，可在进一步研究中进行后续探索。

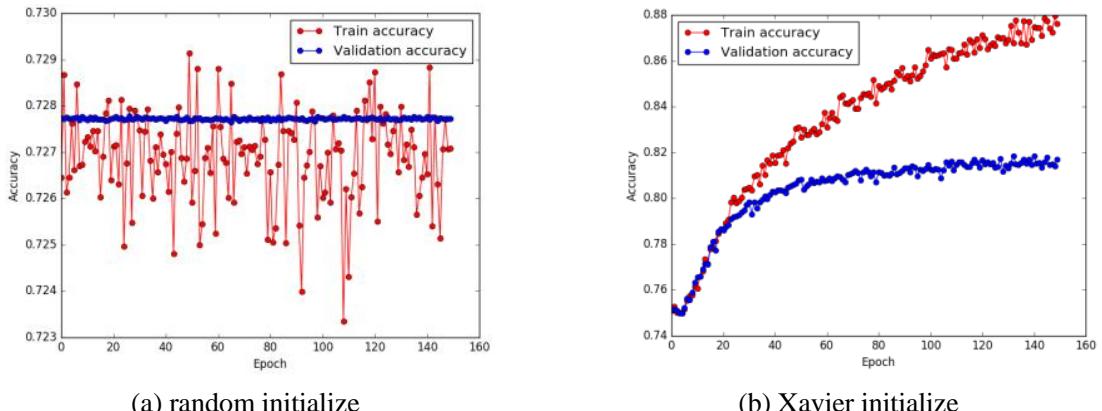


图 3.13 不同初始化方式下的训练情况

3.3.2.4 固定参数对检测精度的影响

深度神经网络权重参数数量庞大，而加载的预训练模型在 ImageNet 上已学习到一些可用的泛化特征。因此，在网络训练过程中，固定某些层的参数不参与权重更新，可适当抑制模型的过拟合程度。一般来说，固定的参数层越多，则模型权重更新的辐射范围就越小，则其与训练集的拟合能力也就越弱。因此，为了寻找设置怎样的固定参数对训练结果最好，设计如下表 3.4 所示实验。在实验中，设定勾选的层作为参数固定层，不参与权重更新，以迭代 150 个周期为标准，进行网络训练和测试实验，测试结果记录如表 3.4。

表 3.4 不同固定参数对检测精度的影响

Prefix parameters:												
Conv1_Bais1	Conv2_Bais2	Conv3_Bais3	Conv4_Bais4	Conv5_Bais5	Conv6_Bais6	Conv7_Bais7	Conv8_Bais8	Conv9_Bais9	Conv10_Bais10	Conv11_Bais11	epoch	mAP
✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	150	20.45
✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	150	52.35
✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	150	71.61
											150	70.82

从表 3.4 的实验结果可以看出，当固定 conv1_ 和 conv2_ 层参数时，平均检测精度最高。第四组没有固定参数，参与更新的权重更多，但训练效果不及第三组，可推测为出现过拟合。图 3.14 表示该组训练过程中的精确率和损失函数值变化情况，从图中也可看出训练已出现过拟合，验证推测。前面两组实验固定较多的参数层，则参与更新的权重更少，其对训练集的拟合能力也相应减弱。由以上现象可推测，加载的预训练模型 VGG-16 在 ImageNet 上训练后已具有多类的泛化特征，而其在低层的权重往往表示一些基础特征，具有特定的含义，比如边缘特征、条纹特征以及颜色信息等，这些特征对于分类具有普适性，因此固定这些层可适当抑制训练时的过拟合。

固定参数层的选择也会受到训练数据量大小的影响，一般来说，数据量小的可以固定更多的参数层，以抑制过拟合。从另一个层面来说，数据量越大，学到的特征越泛化，得到的权重更新对预测越有意义。而缺少数据就需要借用预训练的参数来表示基础特征。

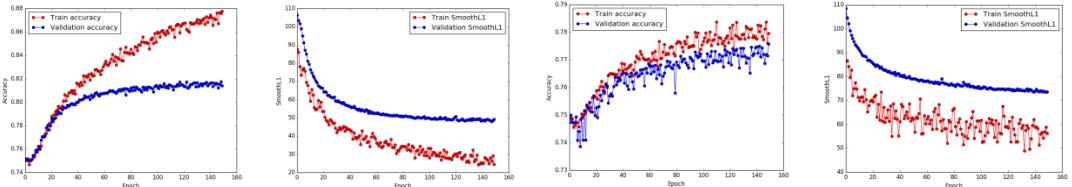


图 3.14 没有固定参数的训练情况(过拟合) 图 3.15 固定 1-7 层参数的训练情况(欠拟合)

3.3.3 检测速度测试结果

为了测试网络的实时性能，我们分别在 CPU 和 GPU 上测试其检测速度情况，实验结果统计如表 3.5 所示。

表 3.5 检测速度测试统计表

Model	CPU/GPU	CUDNN	Batch-size	FPS*
VGG16_reduced 300 × 300	TiTan X	V5.1	16	80
VGG16_reduced 300 × 300	TiTan X	V5.1	8	78
VGG16_reduced 300 × 300	TiTan X	V5.1	1	64
VGG16_reduced 300 × 300	CPU	--	16	0.42
VGG16_reduced 300 × 300	CPU	--	8	0.40
VGG16_reduced 300 × 300	CPU	--	1	0.33

由实验结果可知，该网络已达到实时检测的性能要求。且 GPU 上的最快速度为 80FPS，是 CPU 上的检测速度的近 200 倍，在 CPU 上处理一张照片需要 2.4 秒。同时，SSD 网络也已在检测速度上超过精度相当的 Faster R-CNN 网络，Faster R-CNN 简单网络目标检测速度为 17FPS，在 Pascal VOC 上的准确率为 59.9%；复杂网络速度为 5FPS，准确率为 78.8%。

3.4 本章小结

本章阐述了 SSD 算法在 MXNet 上的具体实现，解析 SSD 基础网络和辅助网络细节信息，详述网络训练的特殊技巧和训练相关参数设置情况。通过在训练集上的进行多次训练得出 SSD 算法在 Pascal VOC20 个类别上的检测精度以及平均检测精度。完成了 SSD 算法采用的特殊策略对训练效果影响程度的实验，得出采用多尺度特征图、随机裁剪样本的数据增广策略、默认包围框的尺度多样选择对检测精度都有不同程度的增益的结论，其他策略如膨胀卷积能够加快检测速度，初始化 Xavier 方法有助于模型在训练中收敛，训练时固定前两个卷积层参数有助于抑制模型过拟合以及得到较好的训练效果。最后，分别在 CPU 和 GPU 上进行了检测速度测试，结果表明网络已达到实时检测的性能要求。

4 基于 SSD 算法的物体抓取测试

针对亚马逊物流分拣挑战赛目标对象，选取适合抓取实验的实验物品，针对这些物品采集图像的彩色图像、红外图像、深度图像以及点云信息。针对彩色图像的目标标注需求，设计批量处理单目标图像的方式生成标注训练集，以及通过编制 GUI 进行多目标图像人工标注的方式生成测试集。然后，对 SSD 算法在自建的数据集上进行网络微调实验，尝试通过改变实验参数以及增加训练技巧的方式提高训练效果。

4.1 面向亚马逊物流分拣挑战赛的目标图像采集

4.1.1 亚马逊物流分拣挑战赛的目标对象

亚马逊物流分拣挑战赛(Amazon Picking Challenge,APC)中参与抓取的商品涵盖 20 多个类别，30 多种日用产品。根据前两届比赛情况，实验选取 30 件国内可买到的同类商品，用于目标检测数据集的创建以及抓取实验。选取的商品如图 4.1 所示。



图 4.1 实验用品

4.1.2 RGB-D 传感器的成像模型及图像采集

在不考虑镜头畸变的情况下，传统的透视摄像机符合小孔成像模型，如图 4.2 所示，三维空间中的物体点 $P = [x, y, z]^T$ 与其在图像平面上所生成的像 $P = [u, v]^T$ 间满足如下关系，其中 f 为摄像机的焦距。

$$u = \frac{(x \cdot f)}{z} \quad (4.1)$$

$$v = \frac{(y \cdot f)}{z} \quad (4.2)$$

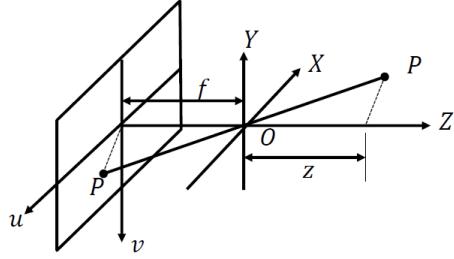


图 4.2 小孔成像模型^[1]

由于透视成像过程丢失了深度信息，一般无法简单地从图像点推知物体点的三维坐标。RGB-D^[27]传感器中的深度感知单元弥补了该方面的不足，通过飞行时间技术或光编码技术(light coding)形成场景的深度图像，深度图像的每一像素 $I_d(u, v)$ 保存了场景中的一个点距离传感器的深度信息(以米为单位)^[1]。通过相机标定，可以将彩色图像与深度图像的各像素逐一对应起来，从而获得彩色图像中的像素点所对应的空间三维坐标：

$$z = I_d(u, v) \quad (4.3)$$

$$x = \frac{(u - c_x)z}{f} \quad (4.4)$$

$$y = \frac{(v - c_y)z}{f} \quad (4.5)$$

其中， c_x, c_y 表示彩色摄像机的光心。



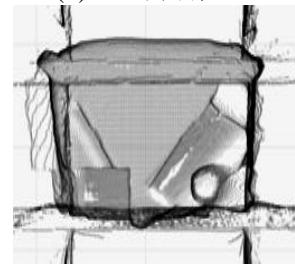
(a) 彩色图像



(b) 红外图像



(c) 深度图像



(d) 可视化点云

图 4.3 RGB-D 传感器采集到的各类图像样本

利用 RGB-D 传感器，采集实验所需的彩色图像、深度图像以及点云信息。为了统一开发环境，采用 kinect2.0 在 linux 下的开源驱动

libfreenect2 配合其 python 接口 `pylibfreenect2` 进行，编写程序控制图像采集过程。采集为保持原始图像信息，得到的各图如图 4.3 所示。

4.2 目标检测数据集的创建

为了使自建数据集具有通用性，便于后续的测试和使用，将采集得到的彩色图像按照 VOC 数据集的标准形式进行处理。

4.2.1 生成标注信息

众所周知，机器学习的训练效果与样本的丰富程度之间有重要关联，有时甚至是其主要决定性因素。因此，为了得到更好的训练效果，实验需要大量经过处理的样本数据。庞大的数据样本单纯采用人工处理将花费巨大的时间和精力，为了加快处理速度，编程实现通过简化图像信息和数字图像处理的方式，对训练数据集中的单目标图像做相应处理，使其批量生成各图像的标签信息，并存储在对应的文本文件中。文本文件中的信息包含该图像中的目标对象数量，目标对象名称及其包围框的左上和右下角点坐标位置。同时，为保证测试结果评估的准确性，验证和测试数据集仍采用人工标注的方式进行处理。

4.2.1.1 图像处理生成标注信息

批量生成标签文件的方法是采集一张单目标物体的彩色图像，称为前景，同时采集一张不含该目标物体的彩色图像，称为背景，通过对前景与背景的比较和处理，生成该图像的标签文件。由于图像中仅含一个目标物体，其对应的图像处理过程相对简单，同时，这种仅含一种标签的数据样本，对训练的准确性并不造成影响。

数字图像处理采用 OpenCV 进行，其处理流程为：

- (1) 将前景与背景两图相减取绝对值；
- (2) 将彩色图像转换为灰度图像；
- (3) 模糊化灰度图像；
- (4) 通过设置阈值二值化灰度图像，并通过连通区域分析返回连通区域面积最大的包围框的中心点坐标以及宽度和高度。

以上步骤对图像的处理过程如图 4.4 所示：



(a) 图像相减

(b) 彩色转灰度

(c) 模糊化处理

(d) 阈值二值化

图 4.4 各步骤处理效果图

采用这种方式处理后统计准确率可达 90%以上，检测效果展示如图 4.5。但仍然存在如图 4.6 所示的个别情况，使其得到的包围框不够准确。图 4.6 表示的是目标对象较小且灰暗时，包围框框选的目标对象将阴影包括在内使得包围框扩大，这种情况预测结果的影响不大故暂且保留；图 4.7 表示的是前景与背景由于短时间内光线情况变化较大导致包围框明显错位，实际中操作中，将这种情况的样本予以剔除。

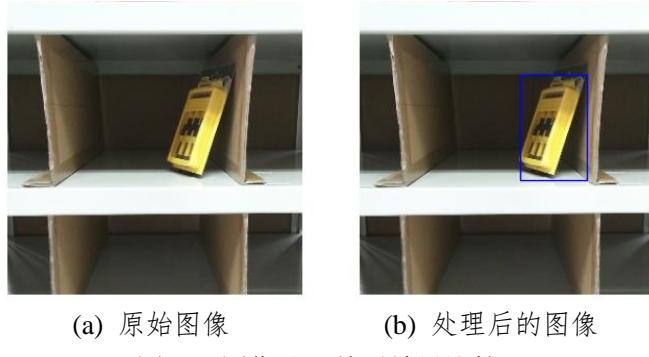


图 4.5 图像处理前后效果比较

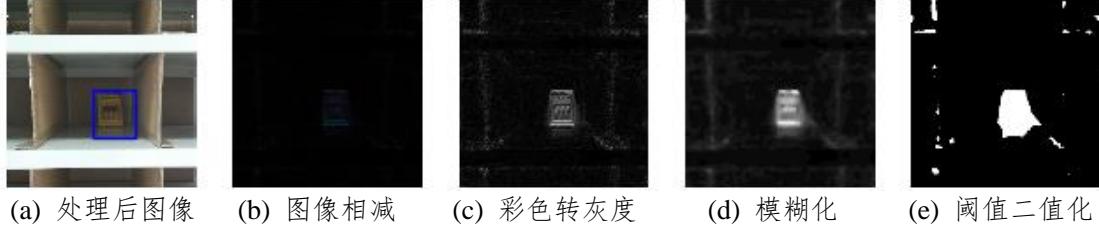


图 4.6 引起标注不准确的情况一

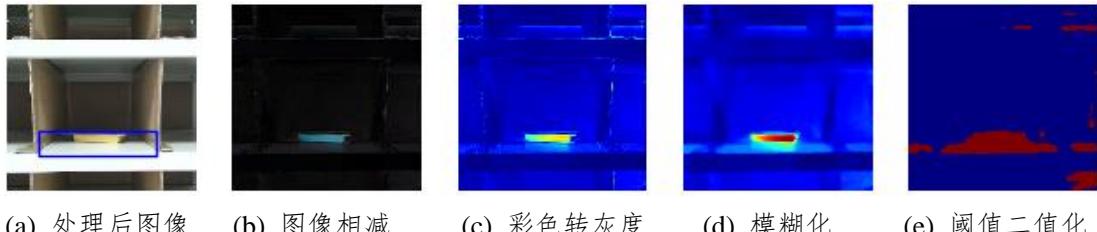


图 4.7 引起标注不准确的情况二

4.2.1.2 人工标注生成标注信息

为了便于人工标注，使用 python Tkinter 编制人工标注小工具 Label Tool(如图 4.8 所示)，进行界面标注操作。下面将详细介绍 Label Tool 的各部分功能及具体操作。

首先，在“Image Dir”栏中输入指定目录下要加载的文件夹编号，如“001”、“002”等，点击“Load”按钮，将目标文件夹中的所有未标注图片逐一载入并显示在界面中间。然后，在“class”类别群中点击当前要标注的目标类别按钮，对应目标类别名称显示在“class”指示的文本框中。移动鼠标，在显示的图片上框选出目标对象的位置，目标包围框的左上和右下角点坐标值将会成对显示在右侧文本框中。点击

“Delete”按钮可删除当前标注的包围框；点击“Clear All”按钮可清除当前图中所有标注的包围框；“<<Prev”和“Next >>”控制图片载入和标注信息的保存；“Go”按钮控制跳转到指定编号的图片进行标注操作；鼠标的当前位置信息显示在界面右下角处。

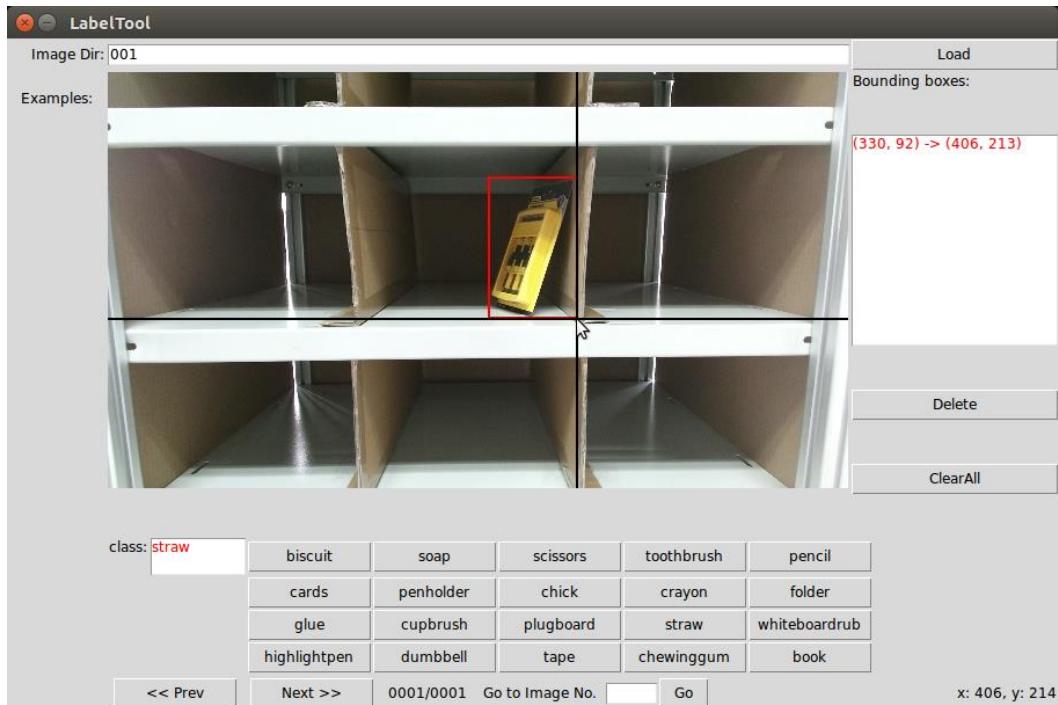


图 4.8 人工标注界面

4.2.2 xml 标签文件生成

VOC2007 中，图像标签都是以.xml 文件的形式进行存储的。借助 python 中的 ElementTree 处理工具简单、易用的特性，建立和读写 xml 文档，生成每张图片的标签文件。xml 文件中记录的图像信息相对于单纯的标注目标位置信息更加详尽、丰富，还包括图像取景的视点、目标对象是否被裁剪等情况表示。

4.2.3 训练和测试数据集生成

将获取的图像按照一定比例随机分成训练图片集(train)、验证图片集(val)和测试图片集(test)，生成对应文件存储参与相应部分的图像序列。有时，训练数据也会用训练和验证的图片合集(trainval)，验证数据也可用测试图像集。训练、验证和测试的图像数量比例一般为 2:1:1，也可以自定比例，从所有图像中随机抽取生成训练、验证和测试序列。本文生成的数据集采用训练、验证、测试图像数量遵循 2:1:1 的比例随机抽取生成。

4.3 自建数据集上的网络微调实验及结果分析

一般来说，在特定领域的识别分类任务中，很难得到大量的训练数据。在这种情况下，从头训练一个新的网络往往比较复杂，而且参数调整较困难，训练数据量也不够丰富，因此在预训练参数网络基础上进行网络微调是一个比较理想的选择。

网络微调就是借助在 ImageNet 上训练好的模型参数文件，根据用户特定的分类识别任务需求进行网络训练，微调网络参数。比如 SSD 网络中的基础网络 VGG-16，就是在 ImageNet 上 1200000 数据上训练得到的，其后的训练也是在 VGG-16 网络参数上的微调。由于从这些预训练的网络抽出来的深度特征有很好的泛化能力，可以应用到其他不同的视觉问题中，而且比传统的人工设计的特征效果好很多，所以得到广泛的应用。一般来说，采用的 CNN 网络在 ImageNet 上的分类效果越好，其深度特征的泛化能力越强。采用网络微调的原因：一是从头训练一个网络需要 ImageNet 等百万量级的数据，一般的视觉任务没有这么多的数据；二是预训练的模型本身的特征已经足够泛化，比如低层的卷积核实现的就是边缘检测滤波器等功能，可以立刻应用到另外一个视觉任务中。

因此，我们使用 SSD 算法以 VGG-16 为基础网络在自建数据集上进行网络微调实验。

网络微调实验中，整个流程可分为以下几个步骤：

- (1) 准备自建的训练数据和测试数据；
- (2) 修改网络结构使其与目标类别相符，并且加快最后一层的参数学习速率；
- (3) 调整训练的配置参数，通常为学习速率和步长，迭代次数适当减少；
- (4) 启动训练，加载预训练的模型参数。

4.3.1 自建数据集训练结果

由于时间紧张，数据采集及处理过程繁琐，截止至目前，共采集与处理完成七个类别(如图 4.9 所示)的图像数据，分别为 pencil/highlighter/penholder/tape/soap/scissors/dumbells，每类单目标对象图片 200 张，包含多目标类别的图片 200 张，总量 1600 张。暂且在这七类数据上进行实验，后续会继续采集更多类别的数据，进行更大规模的数据集创建以及在此基础上进行训练。



图 4.9 已采集完成的 7 个类别目标对象

通过加载预训练模型 VGG-16，以学习率 0.001，动量 0.9，权重衰减 0.0001，每 50 次迭代更新一次权重，每次权重的更新因子为 0.8，在自建数据集上进行训练，通过 300 个迭代周期后测试其训练效果，训练过程如图 4.10 所示，训练结果统计于表 4.1。

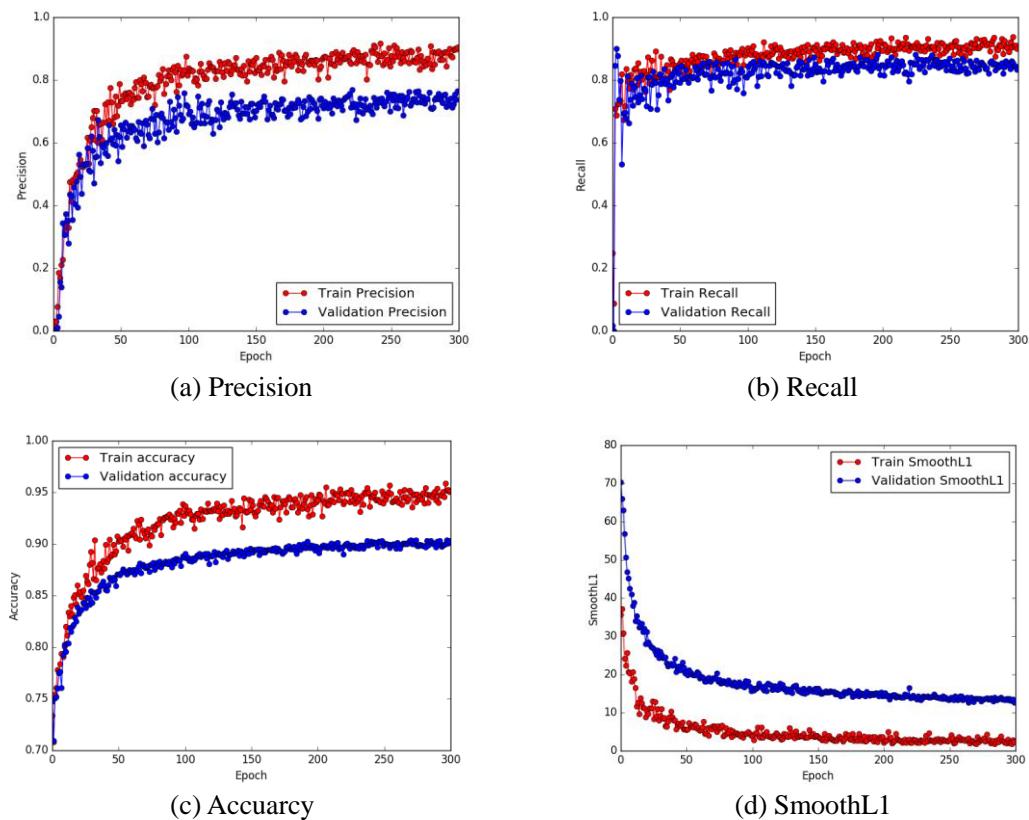


图 4.10 自建数据集网络微调实验训练过程

由图 4.10，网络微调实验训练过程中各项性能指标的变化情况可以看出，整个训练过程收敛的，验证集各项指标与训练集同步变化，且前 300 个迭代周期中尚未出现过拟合或欠拟合的趋势。同时，从图中也不难看出，从第 100 个迭代周期开始，精确率、召回率和准确率已基本不再增加，SmoothL1 损失基本保持不变，局部震荡，可推测此时训练已经接近局部最小值，以小步幅逼近局部最优解。为验证推测，测试第 100 个迭代周期后网络参数的预测统计结果。

表 4.1 自建数据集网络微调训练结果

Method	Epoch	mAP	Pencil	Highlighter	penholder	tape	soap	Scissors	dumbbells
SSD300	300	92.34	89.99	89.07	90.91	97.88	98.16	90.07	90.30
SSD300	100	90.83	89.26	90.55	90.91	87.62	97.53	90.22	89.71

由表 4.1 中数据可看出，训练的平均检测精度在 90% 以上。七个类别之间，**soap** 目标的检测结果精度最高，**highlighter** 和 **tape** 的检测精度相对其他类别低。

可视化检测结果，通过观察结果中的假阳性样本可以发现，**highlighter** 类受未标注对象影响较大。其中，水杯与其颜色配比相似，将水杯误判为彩笔是引起预测失误的主要因素(如图 4.11 所示)，也是导致该类预测精度相对较低的主要原因。同时，也可以就此现象推测，训练得到的卷积核参数具备某种颜色判断的滤波功能，检测器将颜色配比作为判断依据之一。

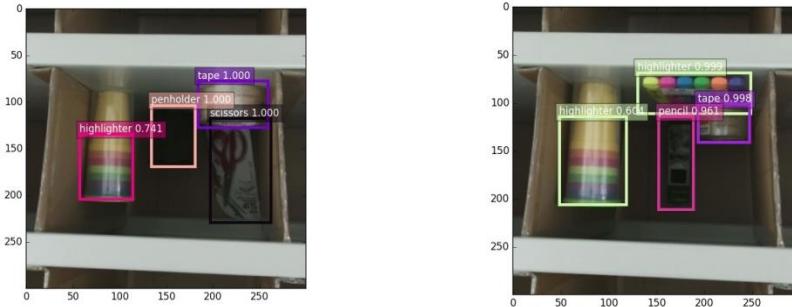


图 4.11 highlighter 类别预测受未标注对象影响的假阳性样本

Tape 则主要因为未被预测出来而造成该类别预测精度较低，部分 **tape** 类别假阳性样本如图 4.12 所示。由此可推测，训练得到的检测器对 **tape** 类提取的特征还不够泛化，缺少鲁棒性。

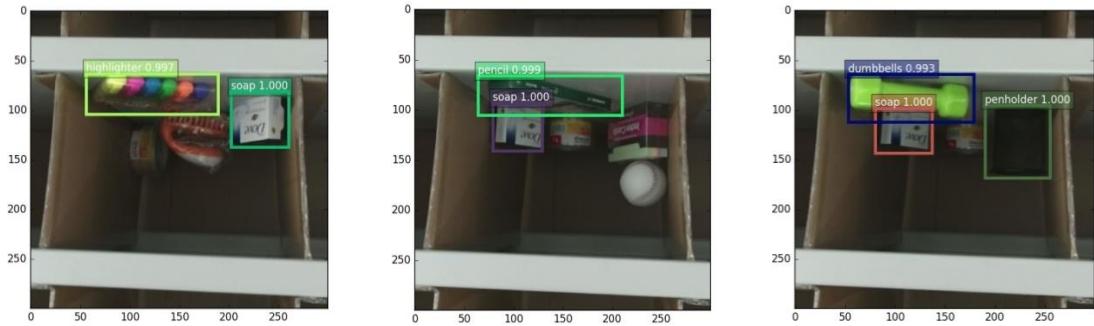


图 4.12 部分 tape 类别假阳性样本

4.3.2 针对自建数据集的 SSD 网络超参数调节

自建数据集的样本数量与标准数据集相差较大，为了寻找小训练样本下适合训练的网络固定参数设定，抑制训练结果过拟合，设计如下表 4.2 所示的网络超参数调节实验。通过固定不同的参数层，探究该训练集下参数固定情况对训练结果的影响。

其中，*conv5*_之后的层在初始化时未加载预训练的模型参数，仅

通过 Xavier 算法初始化权值，故此实验在参数固定时仅设置 4 种不同的固定组合。

表 4.2 自建数据集的固定参数实验结果

Prefix paramters					epoch	mAP
Conv1_Bais1_	Conv2_Bais2_	Conv3_Bais3_	Conv4_Bais4_	Conv5_Bais5_		
✓	✓				300	92.07
✓	✓		✓		300	92.34
✓	✓	✓	✓		300	89.56
✓	✓	✓	✓	✓	300	84.42

通过分析表 4.2 记录的实验结果，可以发现固定卷积层前两层即 *conv1* 和 *conv2* 的参数不变时，自建数据集的训练效果最好，这与之前在 Pascal VOC 上得出的结论一致。其余参数设定情况均有一定程度的训练过拟合或者参数拟合不够情况。因此，在一般网络微调实验中，普遍采取固定前两层不变的方式进行训练。

表中所示各组实验训练过程如图 4.13、图 4.14 和图 4.15 所示。

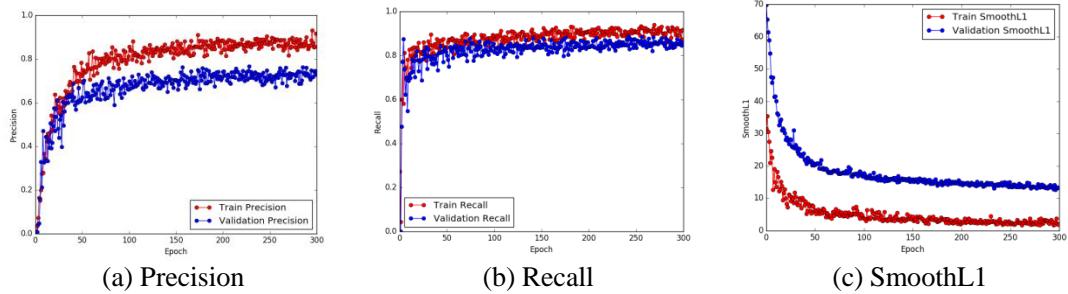


图 4.13 没有固定参数的网络训练过程

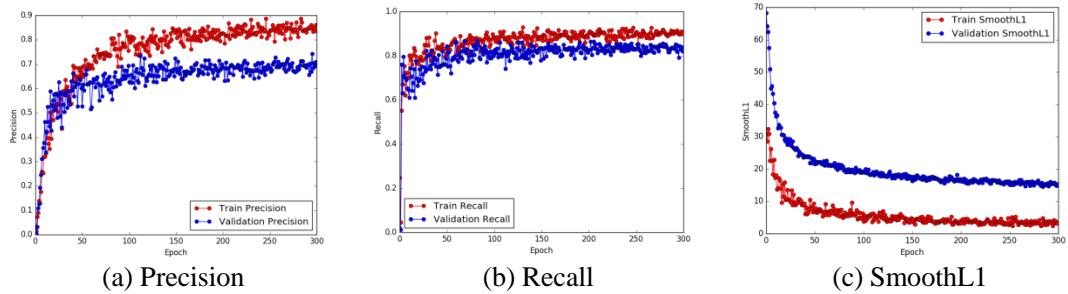


图 4.14 固定前 4 层参数的网络训练过程

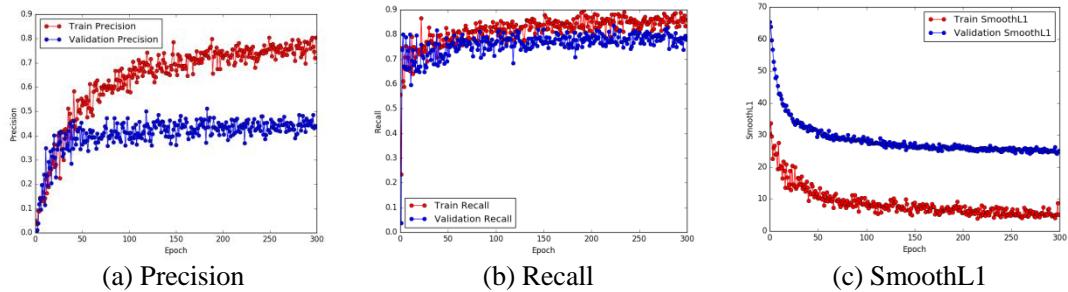


图 4.15 固定前 5 层参数的网络训练过程

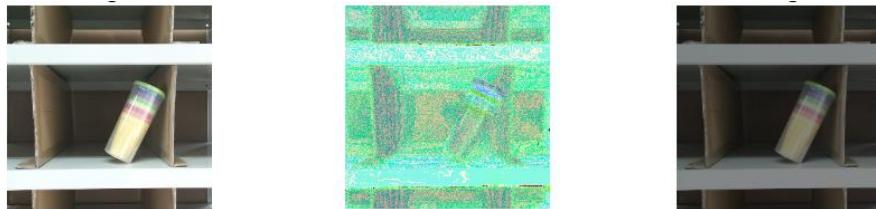
由图 4.13、图 4.14、图 4.15 中各项指标的变化过程对比可以看得出，固定的参数层越多，训练过程的欠拟合程度越严重。主要表现在两方面：(1)各组训练集的 Precision 和 Recall 值在收敛后的稳定值越来越小，SmoothL1 值在收敛后的稳定值越来越大；(2)各组训练集与测试集的 Precision、Recall 以及 SmoothL1 在收敛后的差值越来越大。

4.3.3 数据增广策略对自建数据集训练效果的影响实验

通过对实验样本的观察，发现所采集的测试样本随光照强度有较大变化，尤其是部分样本出现严重曝光的情况。针对此现象，我们通过数据增广时增加光强变化策略，使学习到的特征对不同光强图像具有鲁棒性。所采用的光强变化策略为：

- (1) 将图像的 RGB 颜色空间变换为 HSV 颜色空间；
- (2) 随机更改 HSV 颜色空间中的亮度(Value)值；
- (3) 将 HSV 颜色空间变换回 RGB 颜色空间。

该处理过程如图 4.16 所示。



(a) RGB 到 HSV 颜色空间转换 (b) 随机亮度更改 (c) HSV 到 RGB 颜色空间转换
图 4.16 改变图像亮度中间处理过程

通过光强变换数据增广后的各图像可视化结果如图 4.17 所示。



图 4.17 不同亮度图像可视化结果

为了验证光强变化的数据增广策略对训练测试结果的是否有所提高以及其他数据增广策略在该数据集中对训练效果的贡献程度，通过控制变量进行网络训练，设计如表 4.3 所示实验。此处实验采用七个类别对象的数据集，以迭代 200 个训练周期为标准进行。

表 4.3 不同数据增广策略对训练结果的增益情况

Data augmentation	SSD300			
crop	✓		✓	✓
mirror	✓	✓		✓
rebrightness	✓	✓	✓	
mAP	91.22	77.91	90.81	91.12

由表 4.3 的检测结果可得，随机裁剪对检测效果的贡献最大，当取消随机裁剪策略时，检测精度迅速下降到 77.91%，极大影响了检测效果。同时，其他增广策略对训练结果都有不同程度的增益，其中，镜像带来 0.41% 的性能提升，随机改变亮度也贡献了 0.1% 的性能。

其中各组实验的训练过程如图 4.18、图 4.19、图 4.20 以及图 4.21 所示。

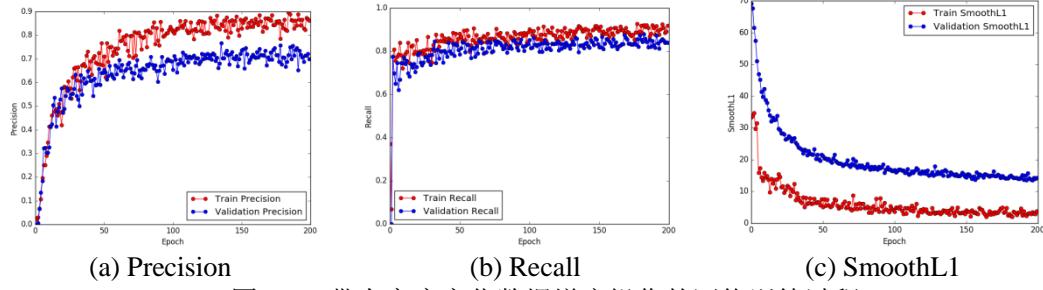


图 4.18 带有亮度变化数据增广操作的网络训练过程

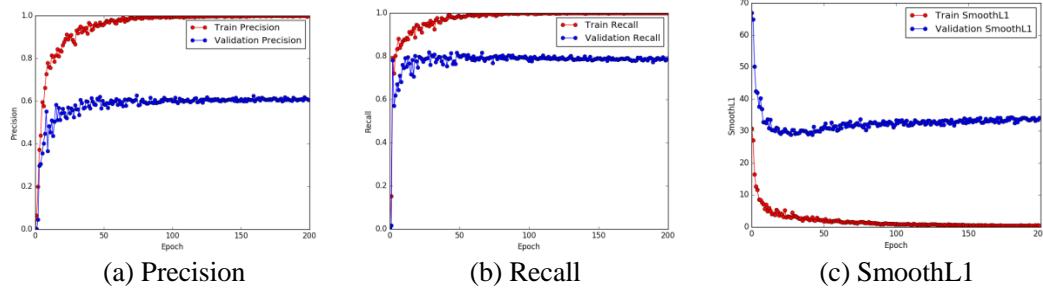


图 4.19 去掉随机裁剪数据增广操作的网络训练过程

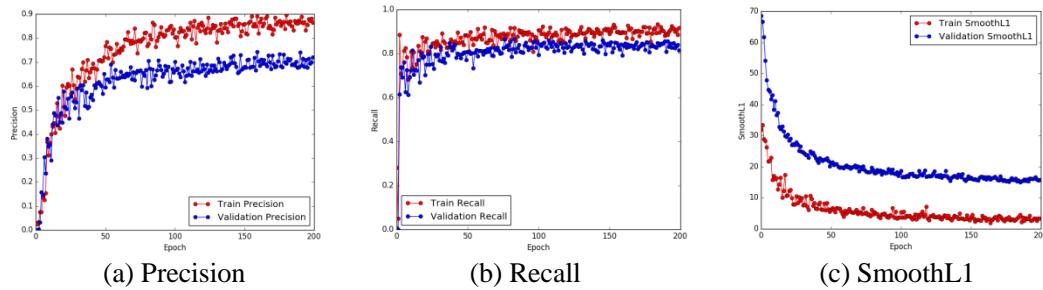


图 4.20 去掉图片镜像数据增广操作参数的网络训练过程

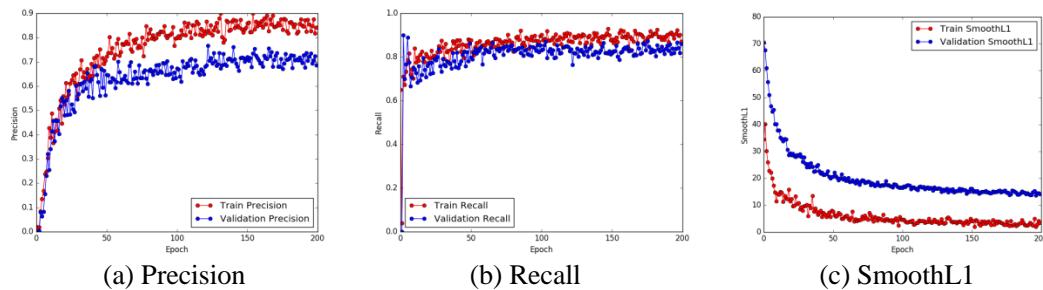


图 4.21 去掉图片亮度变化数据增广操作的网络训练过程

从图 4.18、图 4.19、图 4.20 和图 4.21 可以看出，各组训练过程都是收敛的。其中去掉随机裁剪数据增广策略的网络训练过程有明显

的过拟合现象，图片镜像和亮度变换策略对训练过拟合的影响相对较小。各种不同的数据增广操作一定程度上提高了训练数据的多样程度，对于由数据驱动学习的 **SSD** 算法来说，多样化的数据样本是得到强泛化能力特征的必要条件。

4.3.4 部分检测结果展示

自建数据集上的部分检测结果如图 4.22 所示，由这些检测结果可视化可以得出，**SSD** 的实际效果比较理想。基本上能达到正确识别和精确定位的需求。图中第六幅图像有曝光，针对这样质量的图像，训练得到的网络也能准确地预测目标位置及类别，这是传统的图像处理技术难以达到的，同时，**SSD** 对目标物体在不同位姿、不同视角下都能够检测出来，说明了网络具有很强的鲁棒性，一定程度上表现了其区别于一般图像处理技术的智能能力。

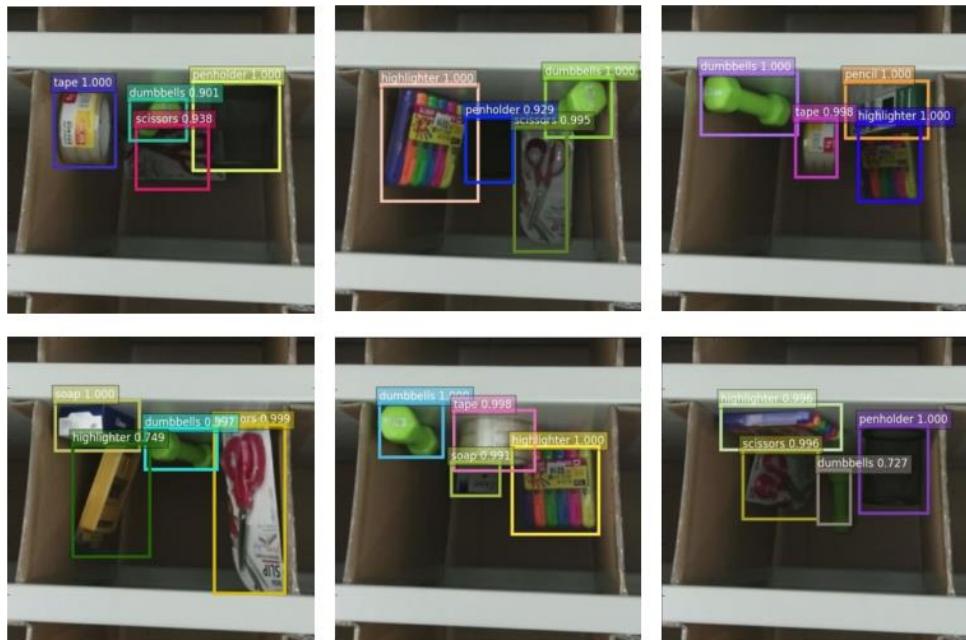


图 4.22 部分检测结果展示

4.4 本章小结

本章首先介绍了亚马逊物流分拣挑战赛的目标对象，针对这些对象，选取相应的实验物品，用于图像采集和抓取实验。接着，通过 **kinect2.0** 相机采集各类物品不同姿态不同视角下的图像数据，包括彩色图像、红外图像、深度图像以及点云信息。为了对采集完成的数据图像进行人为加工，制作成训练和测试所需的 **VOC** 格式标签文件，借助 **opencv** 强大的数字图像处理功能，编程实现了一种自动批量处理简单图像的方式来标注训练样本生成标签，并编制一个手工标注

的工具 Label Tool 来人工处理测试样本生成标注标签。然后，在创建好的数据集上进行利用 SSD 算法网络结构加载 VGG-16 预训练模型进行网络微调实验，并分析假阳性样本，提出提高训练效果的改进策略。针对小样本训练数据集，实验寻找合适的固定参数设定；对数据增广策略做相应补充，实验验证各数据增广策略对自建数据集训练效果的贡献程度。

结 论

基于深度学习的机器人实时目标检测算法研究，针对智能移动作业机器人实现其视觉感知功能，主要通过对一种最新的实时目标检测算法 **SSD** 进行研究、实现以及训练实验，为完成亚马逊物流分拣挑战赛的抓取机器人提供决策基础。

本文取得的研究成果如下：

(1)以亚马逊物流分拣挑战赛为任务场景，从软硬件两个层面设计了智能抓取机器人整体方案，通过对主流目标检测算法的检测过程分析和性能比较，选取了 **SSD** 算法作为目标检测环节的算法原型，**MXNet** 作为开发平台，进行算法实现。

(2)针对已实现的 **SSD** 算法，通过在 **Pascal VOC** 上进行网络训练，与 **FastR-CNN** 和 **Faster R-CNN** 进行检测结果比较，得出 **SSD** 算法无论在检测精度还是检测速度上都具有优异性能。同时，对尺度较小的输入图像仍有理想的检测结果。通过分析假阳性样本，可得到 **SSD** 算法对不同大小和纵横比的目标具有很强的鲁棒性。通过实验分析了 **SSD** 算法采用多尺度特征图、特殊训练技巧、不同初始化策略以及固定参数设置对训练效果的影响。得出了采用特征图数量越多，训练效果越好，但同时网络参数也越多。权衡参数数量及训练效果，采用提取 *conv4_3/conv7/conv8_2/conv9_2/conv10_2/conv11_2* 特征层进行预测以训练网络。数据增广、膨胀卷积以及多种包围框纵横比等都对训练结果有不同的增益，其中数据增广对网络训练来说至关重要。采用 **Xavier** 初始化新增层参数，配合固定 *conv1-/conv2-* 层加载参数训练效果最佳。

(3)针对亚马逊物流分拣挑战赛的目标对象建立了相应的数据集，并制作成与 **Pascal VOC** 相同的标准格式，可适用于多种算法的研究和效果验证。针对自建数据集进行网络参数微调实验，得到较好训练效果的参数网络，可用于智能移动作业机器人的抓取实验。

鉴于时间有限，本文中的很多方面还有待于进一步的详细研究。

(1)建立更加健全的面向亚马逊物流分拣挑战赛的实验数据集，用于后续实验探究。

(2)可进一步探究在自建数据集训练时能够提高训练效果的策略，借助先验知识对网络结构进行修改，尝试不同网络结构下的训练及结果比较，争取获得更好的训练效果。

参考文献

- [1] 杨扬. 基于机器视觉的服务机器人智能抓取研究[D]. 上海交通大学, 2014.
- [2] Ackerman E. Fetch robotics introduces fetch and freight: your warehouse is now automated[J]. IEEE spectrum, 2015, 2.
- [3] Ackerman E. Robotnik enters mobile manipulator market with rb-1[J]. Retrieved 2015, March 30 from <http://spectrum.ieee.org/automaton/robotics/industrialrobots/robotnik-enters-mobile-manipulator-market-with-the-rb1>, 2015.
- [4] Roig Moreno A. High level task planning with inference for the TIAGo robot[D]. Universitat Politècnica de Catalunya, 2016.
- [5] Cousins S. Willow garage retrospective [ros topics][J]. IEEE Robotics & Automation Magazine, 2014, 21(1): 16-20.
- [6] 丁美昆, 徐昱琳, 蒋财军, et al. 基于 Kinect 的机器人臂手系统的目标抓取[J]. 上海大学学报: 自然科学版, 2016, 22(4): 421-431.
- [7] 李国栋, 田国会, 薛英花. 基于 QRCode 技术的家庭服务机器人视觉伺服抓取操作研究[J]. 东南大学学报: 自然科学版, 2010, 40: 30-36.
- [8] Correll N, Bekris K E, Berenson D, et al. Lessons from the amazon picking challenge[J]. arXiv preprint arXiv:1601.05484, 2016.
- [9] 张娟, 毛晓波, 陈铁军. 运动目标跟踪算法研究综述[J]. 计算机应用研究, 2009, 26(12): 4407-4410.
- [10] Zhu W, Zeng N, Wang N. Sensitivity, specificity, accuracy, associated confidence interval and ROC analysis with practical SAS implementations[J]. NESUG proceedings: health care and life sciences, Baltimore, Maryland, 2010: 1-9.
- [11] Everingham M, Zisserman A, Williams C K, et al.: The 2005 pascal visual object classes challenge, Machine Learning Challenges. Evaluating Predictive Uncertainty, Visual Object Classification, and Recognising Tectual Entailment: Springer, 2006: 117-176.
- [12] Krizhevsky A, Sutskever I, Hinton G E. Imagenet classification with deep convolutional neural networks[C]. Advances in neural information processing systems, 2012: 1097-1105.
- [13] Lin T-Y, Maire M, Belongie S, et al. Microsoft coco: Common objects in context[C]. European Conference on Computer Vision, 2014: 740-755.

- [14] Lowe D G. Distinctive image features from scale-invariant keypoints[J]. International journal of computer vision, 2004, 60(2): 91-110.
- [15] Mizuno K, Terachi Y, Takagi K, et al. Architectural study of HOG feature extraction processor for real-time object detection[C]. Signal Processing Systems (SiPS), 2012 IEEE Workshop on, 2012: 197-202.
- [16] Girshick R, Donahue J, Darrell T, et al. Rich feature hierarchies for accurate object detection and semantic segmentation[C]. Proceedings of the IEEE conference on computer vision and pattern recognition, 2014: 580-587.
- [17] He K, Zhang X, Ren S, et al. Spatial pyramid pooling in deep convolutional networks for visual recognition[C]. European Conference on Computer Vision, 2014: 346-361.
- [18] Bo L, Ren X, Fox D. Unsupervised feature learning for RGB-D based object recognition[C]. Experimental Robotics, 2013: 387-402.
- [19] Girshick R. Fast r-cnn[C]. Proceedings of the IEEE International Conference on Computer Vision, 2015: 1440-1448.
- [20] Ren S, He K, Girshick R, et al. Faster R-CNN: Towards real-time object detection with region proposal networks[C]. Advances in neural information processing systems, 2015: 91-99.
- [21] Redmon J, Divvala S, Girshick R, et al. You only look once: Unified, real-time object detection[J]. arXiv preprint arXiv:1506.02640, 2015.
- [22] Socher R, Huval B, Bath B P, et al. Convolutional-Recursive Deep Learning for 3D Object Classification[C]. NIPS, 2012: 8.
- [23] Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition[J]. arXiv preprint arXiv:1409.1556, 2014.
- [24] Bakshi B R, Stephanopoulos G. Wave - net: A multiresolution, hierarchical neural network with localized learning[J]. AIChE Journal, 1993, 39(1): 57-81.
- [25] Hoiem D, Chodpathumwan Y, Dai Q. Diagnosing error in object detectors[C]. European conference on computer vision, 2012: 340-353.
- [26] Liu W, Anguelov D, Erhan D, et al. SSD: Single Shot MultiBox Detector[J]. arXiv preprint arXiv:1512.02325, 2015.
- [27] Sturm J, Engelhard N, Endres F, et al. A benchmark for the evaluation of RGB-D SLAM systems[C]. Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on,

2012: 573-580.

附录

附录 A 主流深度学习框架性能表

框架名称	接口语言	网络定义方式	网络和模型能力	模型部署	速度	灵活性
Caffe	命令行 支持 pycaffe	图层由 C++ 定义，网络由 Protobuf 定义	出色的卷积神经网络实现，对递归网络和语言建模的支持很差	多设备编译、跨平台	快	一般
Torch	Lua 语言 (非主流语言)	图层	卷积网络、递归网络	模型运行需要 LuaJIT 支持	快	好
MXNet	C++、Python、Julia、Matlab 等多种高级语言	利用 Symbol 类定义	随机存储器/动态贝叶斯网络、卷积网络、循环网络	跨平台	快	好
CNTK	命令行 (无高级语言接口)	向量运算符号图	支持大部分网络	跨平台、不支持 ARM 构架	快	好
Theano	Python	向量运算符号图	支持大部分先进网络	跨平台、支持 windows	中等	好
TensorFlow	C/C++、Python	向量运算符号图	理想的递归神经网络实现，不支持双向 RNN 和 3D 卷积	跨平台、不支持 Windows	中等	好

附录 B MXNET 框架下的 SSD 网络构建程序

基础网络构建：

```
##### symbol_vgg16_reduced.py #####
import mxnet as mx
from common import conv_act_layer
from common import multibox_layer

def get_symbol_train(num_classes=20):
    data = mx.symbol.Variable(name="data")
    label = mx.symbol.Variable(name="label")

    # group 1
    conv1_1 = mx.symbol.Convolution(data=data, kernel=(3, 3), \
        pad=(1, 1), num_filter=64, name="conv1_1")
    relu1_1 = mx.symbol.Activation(data=conv1_1, act_type="relu", \
        name="relu1_1")
    conv1_2 = mx.symbol.Convolution(data=relu1_1, kernel=(3, 3), \
        pad=(1, 1), num_filter=64, name="conv1_2")
    relu1_2 = mx.symbol.Activation(data=conv1_2, act_type="relu", \
        name="relu1_2")
    pool1 = mx.symbol.Pooling(data=relu1_2, pool_type="max", \
        kernel=(2, 2), stride=(2, 2), name="pool1")
    # group 2
    conv2_1 = mx.symbol.Convolution(data=pool1, kernel=(3, 3), \
        pad=(1, 1), num_filter=128, name="conv2_1")
    relu2_1 = mx.symbol.Activation(data=conv2_1, act_type="relu", \
        name="relu2_1")
    conv2_2 = mx.symbol.Convolution(data=relu2_1, kernel=(3, 3), \
        pad=(1, 1), num_filter=128, name="conv2_2")
    relu2_2 = mx.symbol.Activation(data=conv2_2, act_type="relu", \
        name="relu2_2")
    pool2 = mx.symbol.Pooling(data=relu2_2, pool_type="max", \
        kernel=(2, 2), stride=(2, 2), name="pool2")
    # group 3
    conv3_1 = mx.symbol.Convolution(data=pool2, kernel=(3, 3), \
        pad=(1, 1), num_filter=256, name="conv3_1")
    relu3_1 = mx.symbol.Activation(data=conv3_1, act_type="relu", \
        name="relu3_1")
    conv3_2 = mx.symbol.Convolution(data=relu3_1, kernel=(3, 3), \
```

```

        pad=(1, 1), num_filter=256, name="conv3_2")
relu3_2 = mx.symbol.Activation(data=conv3_2, act_type="relu", \
                                name="relu3_2")
conv3_3 = mx.symbol.Convolution(data=relu3_2, kernel=(3, 3), \
                                pad=(1, 1), num_filter=256, name="conv3_3")
relu3_3 = mx.symbol.Activation(data=conv3_3, act_type="relu", \
                                name="relu3_3")
pool3 = mx.symbol.Pooling(data=relu3_3, pool_type="max", \
                           kernel=(2, 2), stride=(2, 2), pooling_convention="full", \
                           name="pool3")
# group 4
conv4_1 = mx.symbol.Convolution(data=pool3, kernel=(3, 3), \
                                 pad=(1, 1), num_filter=512, name="conv4_1")
relu4_1 = mx.symbol.Activation(data=conv4_1, act_type="relu", \
                                name="relu4_1")
conv4_2 = mx.symbol.Convolution(data=relu4_1, kernel=(3, 3), \
                                pad=(1, 1), num_filter=512, name="conv4_2")
relu4_2 = mx.symbol.Activation(data=conv4_2, act_type="relu", \
                                name="relu4_2")
conv4_3 = mx.symbol.Convolution(data=relu4_2, kernel=(3, 3), \
                                pad=(1, 1), num_filter=512, name="conv4_3")
relu4_3 = mx.symbol.Activation(data=conv4_3, act_type="relu", \
                                name="relu4_3")
pool4 = mx.symbol.Pooling(data=relu4_3, pool_type="max", \
                           kernel=(2, 2), stride=(2, 2), name="pool4")
# group 5
conv5_1 = mx.symbol.Convolution(data=pool4, kernel=(3, 3), \
                                 pad=(1, 1), num_filter=512, name="conv5_1")
relu5_1 = mx.symbol.Activation(data=conv5_1, act_type="relu", \
                                name="relu5_1")
conv5_2 = mx.symbol.Convolution(data=relu5_1, kernel=(3, 3), \
                                pad=(1, 1), num_filter=512, name="conv5_2")
relu5_2 = mx.symbol.Activation(data=conv5_2, act_type="relu", \
                                name="relu5_2")
conv5_3 = mx.symbol.Convolution(data=relu5_2, kernel=(3, 3), \
                                pad=(1, 1), num_filter=512, name="conv5_3")
relu5_3 = mx.symbol.Activation(data=conv5_3, act_type="relu", \
                                name="relu5_3")
pool5 = mx.symbol.Pooling(data=relu5_3, pool_type="max", \
                           kernel=(3, 3), stride=(1, 1), pad=(1,1), name="pool5")
# group 6
conv6 = mx.symbol.Convolution(data=pool5, kernel=(3, 3), \

```

```

        pad=(6, 6), dilate=(6, 6), num_filter=1024, name="conv6")
relu6 = mx.symbol.Activation(data=conv6, act_type="relu", \
                             name="relu6")
# drop6 = mx.symbol.Dropout(data=relu6, p=0.5, name="drop6")
# group 7
conv7 = mx.symbol.Convolution(data=relu6, kernel=(1, 1), \
                             pad=(0, 0), num_filter=1024, name="conv7")
relu7 = mx.symbol.Activation(data=conv7, act_type="relu", \
                             name="relu7")
# drop7 = mx.symbol.Dropout(data=relu7, p=0.5, name="drop7")

### ssd extra layers ###
conv8_1, relu8_1 = conv_act_layer(relu7, "8_1", 256, kernel=(1,1),\
                                   pad=(0,0), stride=(1,1), act_type="relu", \
                                   use_batchnorm=False)
conv8_2, relu8_2 = conv_act_layer(relu8_1, "8_2", 512,\ \
                                   kernel=(3,3),pad=(1,1), stride=(2,2), \
                                   act_type="relu", use_batchnorm=False)
conv9_1, relu9_1 = conv_act_layer(relu8_2, "9_1", 128, \
                                   kernel=(1,1), pad=(0,0), stride=(1,1), \
                                   act_type="relu", use_batchnorm=False)
conv9_2, relu9_2 = conv_act_layer(relu9_1, "9_2", 256, \
                                   kernel=(3,3), pad=(1,1), stride=(2,2), \
                                   act_type="relu", use_batchnorm=False)
conv10_1, relu10_1 = conv_act_layer(relu9_2, "10_1", 128, \
                                    kernel=(1,1), pad=(0,0), stride=(1,1), \
                                    act_type="relu", use_batchnorm=False)
conv10_2, relu10_2 = conv_act_layer(relu10_1, "10_2", 256, \
                                    kernel=(3,3), pad=(1,1), stride=(2,2), \
                                    act_type="relu", use_batchnorm=False)
conv11_1, relu11_1 = conv_act_layer(relu10_2, "11_1", 128, \
                                    kernel=(1,1), pad=(0,0), stride=(1,1), \
                                    act_type="relu", use_batchnorm=False)
conv11_2, relu11_2 = conv_act_layer(relu11_1, "11_2", 256, \
                                    kernel=(3,3), pad=(0,0), stride=(2,2), \
                                    act_type="relu", use_batchnorm=False)

# specific parameters for VGG16 network
from_layers = [relu4_3, relu7, relu8_2, relu9_2, relu10_2, relu11_2]
sizes = [[.1, .141], [.2, .276], [.38, .461], [.56, .644], [.74, .825], \
         [.92, 1.01]]
ratios = [[1,2,.5], [1,2,.5,3,1./3], [1,2,.5,3,1./3], [1,2,.5,3,1./3], \

```

```

[1,2,.5,3,1./3], [1,2,.5,3,1./3]]
normalizations = [20, -1, -1, -1, -1, -1]
num_channels = [512]

loc_preds, cls_preds, anchor_boxes = multibox_layer(from_layers, \
    num_classes, sizes=sizes, \
    ratios=ratios, \
    normalization=normalizations, \
    num_channels=num_channels, \
    clip=True, interm_layer=0)

tmp = mx.symbol.MultiBoxTarget(*[anchor_boxes, label, \
    cls_preds], overlap_threshold=.5, ignore_label=-1, \
    negative_mining_ratio=3, minimum_negative_samples=0, \
    negative_mining_thresh=.5, variances=(0.1, 0.1, 0.2, 0.2), \
    name="multibox_target")
loc_target = tmp[0]
loc_target_mask = tmp[1]
cls_target = tmp[2]

cls_prob = mx.symbol.SoftmaxOutput(data=cls_preds, \
    label=cls_target, ignore_label=-1, use_ignore=True, \
    grad_scale=3., multi_output=True, normalization='valid', \
    name="cls_prob")
loc_loss_ = mx.symbol.smooth_l1(name="loc_loss_", \
    data=loc_target_mask * (loc_preds - loc_target), \
    scalar=1.0)
loc_loss = mx.symbol.MakeLoss(loc_loss_, grad_scale=1., \
    normalization='valid', name="loc_loss")

# monitoring training status
cls_label = mx.symbol.MakeLoss(data=cls_target, grad_scale=0, \
    name="cls_label")

# group output
out = mx.symbol.Group([cls_prob, loc_loss, cls_label])
return out

def get_symbol(num_classes=20, nms_thresh=0.5, force_suppress=True):
    net = get_symbol_train(num_classes)
    # print net.get_internals().list_outputs()
    cls_preds = net.get_internals()["multibox_cls_pred_output"]

```

```

loc_preds = net.get_internals()["multibox_loc_pred_output"]
anchor_boxes = net.get_internals()["multibox_anchors_output"]

cls_prob = mx.symbol.SoftmaxActivation(data=cls_preds, \
    mode='channel', name='cls_prob')
# group output
# out = mx.symbol.Group([loc_preds, cls_preds, anchor_boxes])
out = mx.symbol.MultiBoxDetection(*[cls_prob, loc_preds, \
    anchor_boxes], name="detection",\
    nms_threshold=nms_thresh, force_suppress=force_suppress, \
    variances=(0.1, 0.1, 0.2, 0.2))
return out

```

辅助网络构建：

```

##### common.py #####
import mxnet as mx
import numpy as np

def conv_act_layer(from_layer, name, num_filter, kernel=(1,1), \
    pad=(0,0), stride=(1,1), act_type="relu", \
    use_batchnorm=False):
    assert not use_batchnorm, "batchnorm not yet supported"
    conv = mx.symbol.Convolution(data=from_layer, kernel=kernel, \
        pad=pad, stride=stride, num_filter=num_filter, \
        name="conv{}".format(name))
    relu = mx.symbol.Activation(data=conv, act_type=act_type, \
        name="{}{}".format(act_type, name))
    if use_batchnorm:
        relu = mx.symbol.BatchNorm(data=relu, \
            name="bn{}".format(name))
    return conv, relu

def multibox_layer(from_layers, num_classes, sizes=[.2, .95],\
    ratios=[1], normalization=-1, num_channels=[], clip=True, \
    interm_layer=0):
    assert len(from_layers) > 0, "from_layers must not be empty list"
    assert num_classes > 0, \
        "num_classes {} must be larger than 0".format(num_classes)
    assert len(ratios) > 0, "aspect ratios must not be empty list"

```

```

if not isinstance(ratios[0], list):
    # provided only one ratio list, broadcast to all from_layers
    ratios = [ratios] * len(from_layers)
assert len(ratios) == len(from_layers), \
    "ratios and from_layers must have same length"

assert len(sizes) > 0, "sizes must not be empty list"
if len(sizes) == 2 and not isinstance(sizes[0], list):
    # provided size range, we need to compute the sizes for each
    # layer
    assert sizes[0] > 0 and sizes[0] < 1
    assert sizes[1] > 0 and sizes[1] < 1 and sizes[1] > sizes[0]
    tmp = np.linspace(sizes[0], sizes[1], \
        num=(len(from_layers)-1))
    min_sizes = [start_offset] + tmp.tolist()
    max_sizes = tmp.tolist() + [tmp[-1]+start_offset]
    sizes = zip(min_sizes, max_sizes)
assert len(sizes) == len(from_layers), \
    "sizes and from_layers must have same length"

if not isinstance(normalization, list):
    normalization = [normalization] * len(from_layers)
assert len(normalization) == len(from_layers)

assert sum(x > 0 for x in normalization) == len(num_channels), \
    "must provide number of channels for each normalized layer"

loc_pred_layers = []
cls_pred_layers = []
anchor_layers = []
num_classes += 1 # always use background as label 0

for k, from_layer in enumerate(from_layers):
    from_name = from_layer.name
    # normalize
    if normalization[k] > 0:
        from_layer = mx.symbol.L2Normalization\
            (data=from_layer, mode="channel",\
            name="{ }_norm".format(from_name))
        scale =mx.symbol.Variable(\n
            name="{ }_scale".format(from_name), \
            shape=(1, num_channels.pop(0), 1, 1)), \

```

```

        from_layer = normalization[k]*mx.symbol.\
            broadcast_mul(lhs=scale, rhs=from_layer)
    if interm_layer > 0:
        from_layer = mx.symbol.Convolution(data=from_layer,\n
            kernel=(3,3), stride=(1,1), pad=(1,1), \
            num_filter=interm_layer, \
            name="{ }_inter_conv".format(from_name))\
        from_layer = mx.symbol.Activation(\n
            data=from_layer, act_type="relu", \
            name="{ }_inter_relu".format(from_name))

    # estimate number of anchors per location
    size = sizes[k]
    assert len(size) > 0, "must provide at least one size"
    size_str = "(" + ",".join([str(x) for x in size]) + ")"
    ratio = ratios[k]
    assert len(ratio) > 0, "must provide at least one ratio"
    ratio_str = "(" + ",".join([str(x) for x in ratio]) + ")"
    num_anchors = len(size) - 1 + len(ratio)

    # create location prediction layer
    num_loc_pred = num_anchors * 4
    loc_pred = mx.symbol.Convolution(data=from_layer, \
        kernel=(3,3), stride=(1,1), pad=(1,1),\
        num_filter=num_loc_pred, \
        name="{ }_loc_pred_conv".format(from_name))
    loc_pred = mx.symbol.transpose(loc_pred, axes=(0,2,3,1))
    loc_pred = mx.symbol.Flatten(data=loc_pred)
    loc_pred_layers.append(loc_pred)

    # create class prediction layer
    num_cls_pred = num_anchors * num_classes
    cls_pred = mx.symbol.Convolution(data=from_layer, \
        kernel=(3,3), stride=(1,1), pad=(1,1), \
        num_filter=num_cls_pred, \
        name="{ }_cls_pred_conv".format(from_name))
    cls_pred = mx.symbol.transpose(cls_pred, axes=(0,2,3,1))
    cls_pred = mx.symbol.Flatten(data=cls_pred)
    cls_pred_layers.append(cls_pred)

    # create anchor generation layer
    anchors = mx.symbol.MultiBoxPrior(from_layer, sizes=\

```

```

size_str, ratios=ratio_str, clip=clip, \
name="{}_anchors".format(from_name))
anchors = mx.symbol.Flatten(data=anchors)
anchor_layers.append(anchors)

loc_preds = mx.symbol.Concat(*loc_pred_layers, \
num_args=len(loc_pred_layers), dim=1, name="multibox_loc_pred")
cls_preds = mx.symbol.Concat(*cls_pred_layers, \
    num_args=len(cls_pred_layers), dim=1)
cls_preds = mx.symbol.Reshape(\
data=cls_preds, shape=(0, -1, num_classes))
cls_preds = mx.symbol.transpose(cls_preds, axes=(0, 2, 1), \
    name="multibox_cls_pred")
anchor_boxes = mx.symbol.Concat(*anchor_layers, \
    num_args=len(anchor_layers), dim=1)
anchor_boxes = mx.symbol.Reshape(data=anchor_boxes, \
shape=(0, -1, 4), name="multibox_anchors")
return [loc_preds, cls_preds, anchor_boxes]

```

致 谢

衷心感谢康辉梅老师对本人的悉心教诲和耐心指导。她的言传身教使我终身受益。

感谢哈尔滨工业大学丁亮教授的悉心指导，特别感谢哈工大机器人(合肥)国际创新研究院的程栋梁博士，在整个毕业设计期间，包括选题、方向、实验等方面给予了耐心的指导以及于我而言不可或缺的帮助。

本论文在哈工大机器人(合肥)国际创新研究院工业智能装备研究所实习期间完成，感谢研究所提供的设备支持，包括 Amax-workstation 工作站以及 UR5 机械臂系统，使我论文中的实验得以顺利完成，特此致谢。

英文文献翻译

译文：

你只看一次：统一的实时对象检测

摘要

我们提出 YOLO，一种新的对象检测方法。对象检测的先前工作重新使用分类器来执行检测。相反，我们将对象检测框架视为一个关于空间分离的边界框和关联类概率的回归问题。单个神经网络直接从全图像在一个评价中预测边界框和类概率。由于整个检测管道是一个单一的网络，它可以在检测性能上直接进行端到端的优化。

我们的统一架构非常快。我们的 YOLO 基础模型以每秒 45 帧的速度实时处理图像。一个较小版本的网络，Fast YOLO，以惊人的速度每秒 155 帧处理图像同时平均检测精度(mAP)仍然达到其他实时检测器的两倍。与最先进的检测系统相比，YOLO 有更多的定位误差，但不太可能在背景预测上成假阳性。最后，YOLO 学习对象的一般表示。从自然图像到其他领域，如艺术品，概括来说，它的表现胜过其他检测方法，包括 DPM 和 R-CNN。

1.介绍

人类看一眼图像就立即知道什么对象在图像中，它们在哪里，以及它们如何相互作用。人类的视觉系统是快速和准确的，允许

我们执行复杂的任务，如用很少的自主意识驾驶。快速，精确的对象检测算法将允许计算机驾驶汽车而不需要用到专业传感器，使用辅助设备来传送实时场景信息给人类用户，并为通用的目的和灵敏的机器人系统解锁更多潜力。

当前的检测系统重新利用分类器来执行检测。为了检测一个对象，这些系统需要一个对象的分类器，在各个位置评估对象以及在测试图像中进行缩放。系统如可变形零件模型(DPM)使用滑动窗口方法，分类器在整个图像中均匀间隔的位置运行 [10]。

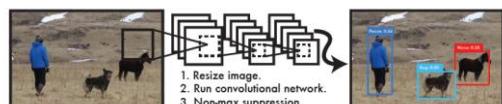


图 1：YOLO 检测系统。处理图像与 YOLO 是简单和直接。我们的系统（1）调整大小输入图像为 448×448 ，（2）运行单个卷积网络在图像上，以及（3）阈值所得到的检测结果模型的信心。

最近的方法，如 R-CNN 使用候选区域，首先在图像中生成潜在边界框的方法，然后在这些提议的框上运行分类器。分类后，运用后处理来细化边界框，消除重复检测，并基于场景中其他对象的框重新分级 [13]。这些复杂管道是缓慢的并且难以优化，因为每个个别组件必须单独进行

训练。

我们将对象检测重构为单个回归问题，直接从图像像素到边界框坐标和类概率。使用我们的系统，你只在图像中查找一次

(YOLO) 在一张图像中预测目前有哪些对象以及这些对象在哪里。

YOLO 很简单：见图 1。单卷积网络同时预测多个边界框和那些框的类概率。YOLO 训练完整的图像并且直接优化检测性能。这种统一的模型有多项优点胜过传统的物体检测方法。

首先，YOLO 检测速度非常快。因为我们把检测作为一个回归问题来重构，而不需要一个复杂的管道。我们只是在测试时在一个新的图像上运行我们的神经网络来预测检测。我们的基础网络在 Titan X GPU 上在没有批处理的情况下以每秒 45 帧的速度运行，一个快速版本可以以超过 150 fps 的速度运行。这个意味着我们可以以少于 25 毫秒延迟的速度实时处理视频流。此外，YOLO 达到了其他实时系统两倍以上的平均精度。这里是我们的系统在网络摄像头上运行的实时演示，可以查看我们的项目网页：<http://pjreddie.com/yolo/>。

第二，YOLO 面向图像的全局做预测。与滑动窗口和基于候选区域的技术不同，YOLO 在训练和测试时可以看到整幅图像，因此它隐含地编码类的环境信息以及它们的外观。Fast R-CNN，

一种顶部检测方法[14]，错误地将图像中的背景补丁判断成对象，因为它看不到更大的环境内容。与 Fast R-CNN 相比，YOLO 的背景误判概率小于一半。

第三，YOLO 学习对象的概括性表示。当训练自然图像和测试艺术品时，YOLO 优于顶级检测方法，如 DPM 和 R-CNN。因为 YOLO 是高度可推广的，当应用于新领域或意外输入时，它不太可能发生故障。

但是，YOLO 仍然在精度上落后于最先进的检测系统。虽然它可以快速识别图像中的对象，但是它在精确地定位一些对象，尤其是小对象时仍需要一些努力。我们将在进一步实验中研究权衡。

我们所有的训练和测试代码都是开源的。各种预训练模型也可供下载。

2. 统一检测

我们统一了进入单个神经网络进行对象检测的独立组件。我们的网络使用从整幅图像得到的特征来预测每个边界框。它也可以同时预测图像的所有类的所有边界框。这意味着我们的网络全局地推断整幅图像和图像中的所有对象。YOLO 设计支持端到端的训练和实时速度，同时保持较高平均精度。

我们的系统将输入图像分成 $S \times S$ 网格。如果对象的中心落入网格单元格，则该网格单元负责检测该对象。

每个网格单元预测 B 个边界框和那些边界框的置信度分数。这些置信度分数反映了盒子包含一个对象的模型的可信程度以及它认为盒子是它预测的准确程度。我们将置信度正式地定义为 $\text{Pr}(\text{Object}) \times \text{IOU}_{\text{pred}}^{\text{truth}}$ 。如果没有对象存在于该单元格中，置信分数应该是零。否则，我们希望置信分数等于预测的边界框和理论正确的边界框之间的交叉区域与联合区域(IOU)的比值。

每个边界框包括 5 个预测量： x, y, w, h 和置信度。 (x, y) 坐标表示边界框中心相对于网格单元边界的位置。宽度和高度相对于整幅图像进行预测。最后预测的置信度表示预测边界框和任何真实边界框之间的联合区域 IOU 。

每个网格单元还预测 C 个类的条件概率， $\text{Pr}(\text{Class}_i | \text{Object})$ 。这些概率条件地分布在包含对象的网格单元上。我们只预测每个网格单元中一组类的概率，而不管盒子数量 B 。

在测试时，我们乘以类的条件概率和每个盒子的预测置信度，

$$\begin{aligned} & \text{Pr}(\text{Class}_i | \text{Object}) \times \text{Pr}(\text{Object}) \\ & \quad \times \text{IOU}_{\text{pred}}^{\text{truth}} \\ & = \text{Pr}(\text{Class}_i) \\ & \quad \times \text{IOU}_{\text{pred}}^{\text{truth}} \end{aligned} \tag{1}$$

这给我们每个盒子对特定类的置信分数。这些分数编码该类出现在框中的概率以及预测框适

合对象的程度。

为了评价 YOLO 对 PASCAL VOC 的影响，我们使用 $S = 7$, $B = 2$ 。PASCAL VOC 有 20 个标记类，因此 $C = 20$ 。我们的最终预测是 $7 \times 7 \times 30$ 的张量。

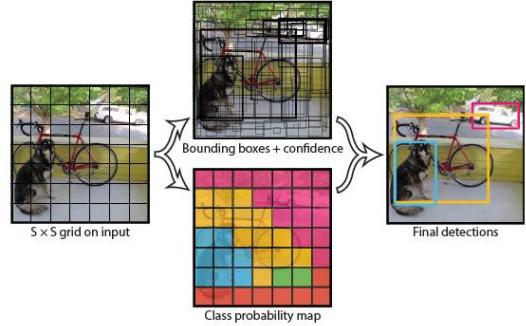


图 2：模型。我们的系统将检测作为回归问题建模。它将图像划分为 $S \times S$ 的网格，每个网格单元预测 B 个边界框，那些框的置信度，以及 C 类的概率。这些预测被编码为 $S \times S \times (B * 5 + C)$ 的张量。

2.1 网络设计

我们将这个模型作为卷积神经网络来实现并在 PASCAL VOC 检测数据集上进行评估[9]。网络的初始卷积层从图像中提取特征同时用完全连接的图层来预测输出概率和坐标。

我们的网络架构的灵感来自 GoogLeNet 图像分类模型[34]。我们的网络有 24 个卷积层，之后是 2 个完全连接的层。而不是 GoogLeNet 使用的 inception 模块，我们简单地使用 1×1 还原层，然后 3×3 卷积层，类似于 Lin 等人的工作[22]。完整的网络如图 3 所示。

我们还训练了一个快速版的 YOLO，为了推动快速对象检测的边界。Fast YOLO 使用一个具有较少卷积层的神经网络(用 9

代替 24)和更少的过滤器。除了网络的大小，所有的训练和测试参数在 YOLO 和 Fast YOLO 之间都是相同的。

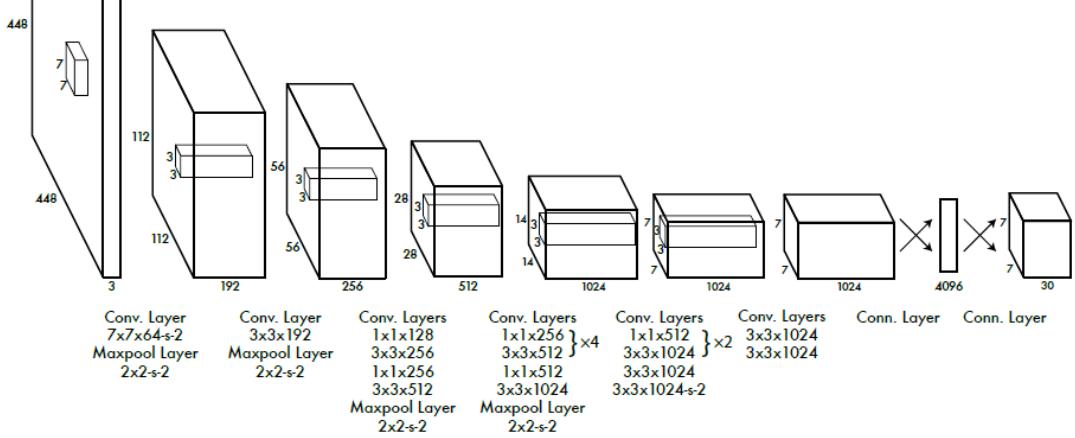


图 3：架构。我们的检测网络有 24 个卷积层，其次是 2 个完全连接的层。交替 1×1 卷积层减少来自前面层的特征空间。我们预处理 ImageNet 分类上的卷积层任务在分辨率的一半 (224×224 输入图像)，然后双倍的分辨率进行检测。

我们网络的最终输出是 $7 \times 7 \times 30$ 的预测张量。

2.2 训练

我们在 ImageNet1000 类竞赛数据集上预训练我们的卷积层 [30]。对于预训练，我们使用图 3 中的前 20 个卷积层，之后是一个平均池化层和一个完全连接层。我们大约用了一个星期来训练这个网络，在 ImageNet 2012 验证集中单个裁剪精确度达到排名前五为 88%，与在 Caffe 的 Model Zoo 里的 GoogLeNet 模型表现相当。我们使用 Darknet 框架训练和推理[26]。

然后我们转换模型以执行检测。Ren et al 显示添加卷积和连接层到预训练网络中可以提高性能 [29]。按照他们的例子，我们添加了四个卷积层和两个完全连接

层采用随机初始化权重。检测通常需要细粒度的可视信息所以我们将网络的输入分辨率从 224×224 增加到 448×448 。

我们的最后一层预测类概率和边界框坐标。我们将边界框宽度和高度通过图像的宽度和高度归一化，以便它们的数值落在 0 和 1 之间。我们参数化边界框 x 和 y 的坐标为特定网格单元位置的偏移量因此它们也在 0 和 1 之间。

我们在最后一层使用线性激活函数，所有其他层使用以下泄漏校正线性激活函数：

$$\phi(x) = \begin{cases} x, & \text{if } x > 0 \\ 0.1x, & \text{otherwise} \end{cases} \quad (2)$$

我们优化了模型输出中的平方误差。我们使用平方误差，因为它很容易优化，然而它不完全符合我们最大化平均精度的目标。

它加权定位误差和分类误差一样，同样可能不是理想的结果。此外，在每个图像中，许多网格单元不包含任何对象。这推动那些单元的“置信度”分数趋于零，经常压倒来自包含对象的单元的梯度。这可能导致模型不稳定，造成训练早早发散。

为了弥补这一点，我们增加了边界框坐标预测的损失权重，减少了不包含对象的框的置信度预测的损失权重。我们使用两个参数， λ_{coord} 和 λ_{noobj} 来完成这一点。

我们设置 $\lambda_{coord} = 5$ 和 $\lambda_{noobj} = .5$ 。

平方误差对大箱子和小箱子也平均地进行加权。我们的误差矩阵应该反映出大框问题比小框问题小一点的偏差。为了部分解决这个问题，我们预测边框宽度和高度的平方根而不是直接预测宽度和高度。

YOLO 预测每个网格单元的多个边界框。在训练时，我们只需要一个边界框预测器负责每个对象。我们分配一个预测变量以“负责”对象，基于这种方式，预测结果是拥有与真实结果最高 IOU。这导致边界框之间的专门化预测。每个预测变量更好地预测某些尺寸，长宽比或对象类别，总体上提高召回率。

在训练时，我们对一下部分进行优化，多部分的损失函数：

$$\lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{I}_{ij}^{obj} [(x_i - \cdot)^2]$$

其中 \mathbb{I}_i^{obj} 表示对象出现在单元格 i 和 \mathbb{I}_{ij}^{obj} 表示单元 i 中的第 j 个边界框预测变量是“负责”用于该预测。

请注意，损失函数仅惩罚错误分类，如果一个对象存在于该网格单元格中（此前讨论过的类的条件概率）。它也只惩罚边界框的坐标误差如果该预测变量负责真实边界框（即在该网格中有最高 IOU 的预测器）。

我们在 PASCAL VOC 2007 和 2012 的训练和验证数据集上通过 135 次迭代训练网络。在 2012 上测试时，我们还包括使用了 VOC 2007 训练的测试数据。在整个训练过程中，我们使用批大小为 64，动量为 0.9，衰减为 0.0005。

我们的学习率安排如下：第一次迭代，我们将学习率从 10^{-3} 缓慢提高至 10^{-2} 。如果我们从一个高的学习率开始，我们的模型通常会由于梯度下降不稳定而发散。我们以 10^{-2} 的学习率迭代 75 次继续训练，然后以 10^{-3} 的学习率迭代 30 次，最后以 10^{-4} 的学习率迭代 30 次。

为了避免过度拟合，我们使用 dropout 和大量的数据增广。在第一个连接层之后，有一个 $rate = .5$ 的 dropout 层防止层与层之间的共适应 [18]。对于数据增广，我们引入随机缩放和缩放范围最高为原始图像尺寸的 20%。我们也在 HSV 色彩空间中随机调整图像的曝光和饱和度，

调整因素最高为 1.5 倍。

2.3 推理

就像在训练时一样，测试图像的预测检测只需要一次网络评估。在 PASCAL VOC 网络中，每幅图像预测 98 个边界框，对每个边界框的预测类概率。YOLO 测试时速度极快，因为它只需要一个网络评估，不同于基于分类器的方法。

网格设计在边界框预测中实现了空间多样性。通常对象落入哪个网格单元很清楚并且该网络对每个对象只预测一个边界框。然而，一些大的对象或在多个单元的边界附近对象可以通过多个单元来更好地定位。非极大值抑制可以用于修复这些多单元的检测。虽然这种方式对性能并不重要，因为这种方法是针对 R-CNN 或 DPM 的，但非极大值抑制仍增加了 2-3% 的平均检测精度 (mAP)。

2.4 YOLO 的局限性

YOLO 对边界框预测强加了空间约束，因为每个网格单元只预测两个框并且只能有一个类。这个空间约束限制了我们的模型可以预测的附近对象的数量。我们的模型还需要在以群体形式出现的小对象这方面做更多研究，如鸟群。

因为我们的模型从数据中学不会预测边界框，它努力推广到新的或不寻常的对象纵横比或配置。我们的模型也使用相对粗糙的特征用于预测边界框，因为我们的

架构有多个来自输入图像的下采样层。

最后，当我们训练一个近似的损失函数来检测性能，我们的损失函数在小边界框与大边界框上的误差表现相同。在一个大盒子里的一个小错误通常是良性的，但是小盒子中的小误差对 IOU 有较大的影响。我们的主要错误来源是不正确的定位。

3.与其他检测系统的比较

对象检测是计算机视觉中的核心问题。检测管道通常从输入图像中提取一组鲁棒的特征开始 (Haar [25], SIFT [23], HOG [4], 卷积特征 [6])。然后，分类器 [36,21,13,10] 或定位器 [1,32] 用于在特征空间中识别对象。这些分类器或定位器以滑动窗口的方式在整个图像或图像中的一些区域子集上运行 [35,15,39]。我们将 YOLO 检测系统与几个顶部检测框架进行比较，突出关键的相似点和差异。

可变形零件模型。 可变形零件模型 (DPM) 使用滑动窗口的方法来进行对象检测 [10]。DPM 使用不相交的管道提取静态特征，分类区域，预测高评分的边界框区域等。我们的系统用单个卷积神经网络取代了所有这些分离部分。网络同时执行特征提取，边界框预测，非最大值抑制和上下文推断。不同于静态特征，网络联机训练特征并对其进行优化以用于检测任务。我们的统一架构使得模型比 DPM 更快，更准确。

R-CNN。 R-CNN 及其变体使用候选区域而不是滑动窗口在图像中寻找对象。Selective Search[35]生成潜在的边界框，从卷积网络提取特征，支持向量机 SVM 对盒子进行评分，线性模型调整边界框位置，以及非极大值抑制消除重复检测。这个复杂流水线模型的每个阶段必须独立精确调谐并且由此产生的系统非常缓慢，测试时每张图片的处理时间超过 40 秒[14]。

YOLO 与 R-CNN 有一些相似之处。每个网格单元格提出可能的边界框，并使用卷积特征对这些边界框进行评分。但是，我们的系统对网格单元提出空间约束有助于减轻同一对象的多个检测。我们的系统也提出了很少的边界框，只有 98 个每幅图像相比从选择性搜索中得到的约 2000 个边界框。最后，我们的系统把这些单独的组件组合成为一个单独的，联合优化的模型。

其他快速检测器 Fast 和 Faster R-CNN 专注于通过共享计算来加速 R-CNN 框架，并使用神经网络来代替选择性搜索提出候选区域[14] [28]。虽然他们相对于 R-CNN 在速度和精度上都有提高，但两者仍然在实时性能方面表现不足。

许多研究工作集中在加速 DPM 管道[31][38][5]。他们加速 HOG 计算，使用级联和推送计算到 GPU。然而，只有 30Hz 的 DPM [31]实际上是实时运行的。

不是在一个大的检测管道中尝试优化个别组件，YOLO 完全抛弃管道模型并在设计上获得更快的速度。

面部或人的单类检测器可以高度优化，因为他们必须处理的差异少得多 [37]。YOLO 是一个学习同时检测各种对象的通用检测器。

Deep MultiBox。 与 R-CNN 不同，Szegedy 等人训练卷积神经网络来预测感兴趣的区域[8]，而不是使用选择性搜索。MultiBox 也可以通过用单类预测替换置信度预测来执行单个对象检测。但是，MultiBox 不能执行一般的对象检测，仍然只是一个在更大的检测管道中的一个步骤，还需要进一步的图像补丁分类。YOLO 和 MultiBox 都使用卷积网络在图像中预测边界框但 YOLO 是一个完整的检测系统。

OverFeat。 Sermanet 等人训练卷积神经网络来执行定位并且应用该定位器执行检测[32]。OverFeat 有效地执行滑动窗口检测，但它仍然是一个不相交的系统。OverFeat 对定位而不是检测性能做了优化。像 DPM 一样，定位器只在做出预测的时候看到本地信息。OverFeat 不能推测全局环境因此需要大量的后处理来生产相干检测。

MultiGrasp。 我们的工作在设计上类似于 Redmon 等人[27]的抓握检测工作。我们的网格方法进行边界框预测是基于

MultiGrasp 系统回归到掌握。然而，抓握检测是一个比对象检测简单很多的任务。**MultiGrasp** 只需要在一张包含一个对象的图像中预测单个可抓取区域。它不必估计大小，位置，或对象的边界或预测它的种类，只需要找到一个适合于抓握的区域即可。**YOLO** 在一张包含多个对象多种类的图片中预测边界框和多个对象的类概率。

4.实验

首先我们在 PASCAL VOC 2007 上比较 **YOLO** 和其他实时检测系统。为了了解 **YOLO** 和 R-CNN 变体之间的差异，我们探讨 **YOLO** 和 Fast R-CNN，一个的最高性能版本的 R-CNN [14]，在 VOC 2007 上产生的错误。基于不同的误差曲线，我们发现 **YOLO** 可以用于重定向 Fast R-CNN 检测并减少背景假阳性的错误，带来了显著的性能提升。我们还提供 VOC 2012 结果以及将 mAP 与当前最先进的方法进行比较。最后，我们发现 **YOLO** 与其他探测器相比，能更好地推广到新域，如在两个艺术品数据集上。

4.1 与其他实时系统的比较

对象检测中的许多研究工作都集中在使标准检测管道加快速度。[5][38][31][14][17][28]然而，只有 Sadeghi 等人真正地做出了实时运行的检测系统(每秒 30 帧或更好)[31]。我们将 **YOLO** 和以 30Hz 或 100Hz 运行在 GPU 上实

现的 DPM 进行比较。而其他努力没有达到实时里程碑，我们还比较他们的相对 mAP 和速度来检查在对象检测系统中可用的折衷的精度性能。

Fast YOLO 是在 PASCAL 上最快的对象检测方法；据我们所知，它是现存的最快的对象探测器。使用 52.7% 的 mAP，它在实时检测上的准确度是先前工作的两倍多。**YOLO** 推动 mAP 到 63.4%，同时仍然保持实时性能。

我们还使用 VGG-16 训练 **YOLO**。这个模型更加准确，但也明显慢于 **YOLO**。与其他依赖 VGG-16 的检测系统进行比较是有用的，但由于它比实时要慢。其余文章都集中在我们的更快的模型上。

Fastest DPM 在没有牺牲很多平均检测精度 mAP 的情况下，有效地加速 DPM，但它仍然不满足实时性能，相差 2 个因子[38]。与神经网络方法相比，它也受 DPM 相对较低的检测精度的限制。

R-CNN 减去 R 代替使用静态的候选边界框产生方法选择性搜索 [20]。虽然它比 R-CNN 快得多，但仍然低于实时性要求并且在没有好的候选区域的情况下需要很好的准确性。

Fast R-CNN 加快了 R-CNN 的分类阶段，但它仍然依赖于每幅图像大约花费 2 秒生成候选边界框的选择性搜索。因此，它具有高的 mAP，但是以 0.5fps 的速

度，仍然离实时要求很远。

Real-Time Detectors	Train	mAP	FPS
100Hz DPM [31]	2007	16.0	100
30Hz DPM [31]	2007	26.1	30
Fast YOLO	2007+2012	52.7	155
YOLO	2007+2012	63.4	45
<hr/>			
Less Than Real-Time			
Fastest DPM [38]	2007	30.4	15
R-CNN Minus R [20]	2007	53.5	6
Fast R-CNN [14]	2007+2012	70.0	0.5
Faster R-CNN VGG-16[28]	2007+2012	73.2	7
Faster R-CNN ZF [28]	2007+2012	62.1	18
YOLO VGG-16	2007+2012	66.4	21

表 1: PASCAL VOC 2007 实时系统快速检测器的性能和速度。快速 YOLO 是在 PASCAL VOC 检测记录上最快的检测器，并且检测精确度仍然是任何其他实时检测器的两倍。YOLO 是 10 mAP 比快速版更准确，同时仍然高于实时速度。

最近 Faster R-CNN 用一个神经网络来产生候选边界框取代了选择性搜索，类似 Szegedy 等人。[8]在我们的测试中，他们最准确的模型实现 7 fps，而更小，准确度较低的模型可以以每秒 18 帧的速度运行。VGG-16 版本的 Faster R-CNN 是 10 mAP，检测精度更高，但也比 YOLO 慢 6 倍。ZeilerFergus 的 Faster R-CNN 只比 YOLO 慢 2.5 倍但也不太准确。

4.2 VOC 2007 错误分析

进一步检查 YOLO 和最先进的检测器之间的差异，我们看看在 VOC 2007 上的详细的分解结果。我们比较了 YOLO 和 Fast RCNN，因为 Fast R-CNN 在 PASCAL 上是性能最高的探测器之一，并且它的检测是公开可用的。

我们使用 Hoiem 等人的方法和工具。[19]对于测试时的每个类别，我们查看该类别的前 N 个预测结果。每个预测是正确的或

者它根据错误的类型进行分类：

- 正确：**正确的类且 $\text{IOU} > .5$

- 本地化：**正确的类， $.1 < \text{IOU}$

$<.5$

- 类似：**类相似， $\text{IOU} > .1$

- 其他：**类错了， $\text{IOU} > .1$

- 背景：**对于任何对象， $\text{IOU} < .1$

$<.1$

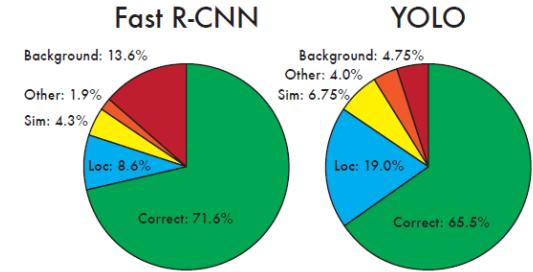


图 4：错误分析：Fast R-CNN 与 YOLO 这些图表显示本地化和背景错误的百分比在前 N 个检测中为各种类别 ($N = \#$ 个对象在对应类别中)。

图 4 表明了平均的每个误差类型的细分跨越所有 20 个类。

YOLO 努力正确地定位对象。定位错误比所有其他源组合的错误所占的比例还要多。Fast R-CNN 使定位错误更少但背景错误更多。13.6% 它的顶部检测是假阳性，这不包含任何对象。Fast R-CNN 预测背景检测的可能性几乎是 YOLO 的 3 倍。

4.3 结合 Fast R-CNN 和 YOLO

YOLO 的背景错误比 Fast R-CNN 少得多。通过使用 YOLO 消除背景检测，在 Fast R-CNN 上，我们得到显著的性能的提高。对于 R-CNN 预测的每个边界框，我们检查 YOLO 是否预测了一个类似的框。如果是，我们基于 YOLO 预测的概率以及两个框之间的重叠，给予该预测增强。

最好的 Fast R-CNN 模型在 VOC 2007 测试集上实现了 71.8% 的 mAP。当与 YOLO 组合时，其 mAP 从 3.2% 增加至 75.0%。我们也尝试结合 Fast R-CNN 顶部模型与其他几个版本的 Faster R-CNN。这些合奏产生了 mAP 在 .3 和 .6% 之间的小幅增长，详见表 2。

YOLO 的提升不仅仅是一个模型组合的副产品，因为不同版

VOC 2012 test	mAP	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	personplant	sheep	sofa	train	tv	
MR.CNN_MORE..DATA [11]	73.9	85.5	82.9	76.6	57.8	62.7	79.4	77.2	86.6	55.0	79.1	62.2	87.0	83.4	84.7	78.9	45.3	73.4	65.8	80.3	74.0
HyperNet_VGG	71.4	84.2	78.5	73.6	55.6	53.7	78.7	79.8	87.7	49.6	74.9	52.1	86.0	81.7	83.3	81.8	48.6	73.5	59.4	79.9	65.7
HyperNet_SP	71.3	84.1	78.3	73.3	55.5	53.6	78.6	79.6	87.5	49.5	74.9	52.1	85.6	81.6	83.2	81.6	48.4	73.2	59.3	79.7	65.6
Fast R-CNN + YOLO	70.7	83.4	78.5	73.5	55.8	43.4	79.1	73.1	89.4	49.4	75.5	57.0	87.5	80.9	81.0	74.7	41.8	71.5	68.5	82.1	67.2
MR.CNN_S.CNN [11]	70.7	85.0	79.6	71.5	55.3	57.7	76.0	73.9	84.6	50.5	74.3	61.7	85.5	79.9	81.7	76.4	41.0	69.0	61.2	77.7	72.1
Faster CNN [28]	70.4	84.9	78.9	74.3	53.9	49.8	77.5	75.9	88.5	45.6	77.1	55.3	86.9	81.7	80.9	79.6	40.1	72.6	60.9	81.2	61.5
DEEP_ENS_COCO	70.1	84.0	79.4	71.6	51.9	51.1	74.1	72.1	88.6	48.3	73.4	57.8	86.1	80.0	80.7	70.4	46.6	69.6	68.8	75.9	71.4
NoC [29]	68.8	82.8	79.0	71.6	52.3	53.7	74.1	69.0	84.9	46.9	74.3	53.1	85.0	81.3	79.5	72.2	38.9	72.4	59.5	76.7	68.1
Fast R-CNN [14]	68.4	82.3	78.4	70.8	52.3	38.7	77.8	71.6	89.3	44.2	73.0	55.0	87.5	80.5	80.8	72.0	35.1	68.3	65.7	80.4	64.2
UMICH_FGS_STRUCT	66.4	82.9	76.1	64.1	44.6	49.4	70.3	71.2	84.6	42.7	68.6	55.8	82.7	77.1	79.9	68.7	41.4	69.0	60.0	72.0	66.2
NUS_NIN_C2000 [7]	63.8	80.2	73.8	61.9	43.7	43.0	70.3	67.6	80.7	41.9	69.7	51.7	78.2	75.2	76.9	65.1	38.6	68.3	58.0	68.7	63.3
BabyLearning [7]	63.2	78.0	74.2	61.3	45.7	42.7	68.2	66.8	80.2	40.6	70.0	49.8	79.0	74.5	77.9	64.0	35.3	67.9	55.7	68.7	62.6
NUS_NIN	62.4	77.9	73.1	62.6	39.5	43.3	69.1	66.4	78.9	39.1	68.1	50.0	77.2	71.3	76.1	64.7	38.4	66.9	56.2	66.9	62.7
R-CNN VGG BB [13]	62.4	79.6	72.7	61.9	41.2	41.9	65.9	66.4	84.6	38.5	67.2	46.7	82.0	74.8	76.0	65.2	35.6	65.4	54.2	67.4	60.3
R-CNN VGG [13]	59.2	76.8	70.9	56.6	37.5	36.9	62.9	63.6	81.1	35.7	64.3	43.9	80.4	71.6	74.0	60.0	30.8	63.4	52.0	63.5	58.7
YOLO	57.9	77.0	67.2	57.7	38.3	22.7	68.3	55.9	81.4	36.2	60.8	48.5	77.2	72.3	71.3	63.5	28.9	52.2	54.8	73.9	50.8
Feature Edit [33]	56.3	74.6	69.1	54.4	39.1	33.1	65.2	62.7	69.7	30.8	56.0	44.6	70.0	64.4	71.1	60.2	33.3	61.3	46.4	61.7	57.8
R-CNN BB [13]	53.3	71.8	65.8	52.0	34.1	32.6	59.6	60.0	69.8	27.6	52.0	41.7	69.6	61.3	68.3	57.8	29.6	57.8	40.9	59.3	54.1
SDS [16]	50.7	69.7	58.4	48.5	28.3	28.8	61.3	57.5	70.8	24.1	50.7	35.9	64.9	59.1	65.8	57.1	26.0	58.8	38.6	58.9	50.7
R-CNN [13]	49.6	68.1	63.8	46.1	29.4	27.9	56.6	57.0	65.9	26.5	48.7	39.5	66.2	57.3	65.4	53.2	26.2	54.5	38.1	50.6	51.6

表 3: PASCAL VOC 2012 排行榜。 YOLO 与完整 comp4 (允许外部数据) 2015 年 11 月 6 日的公开排行榜比较。显示各种检测方法的平均精度和每类平均精度。YOLO 是唯一的实时检测器。Fast R-CNN + YOLO 是第四高的评分方法，比 Fast R-CNN 增加了 2.3%。

	mAP	Combined	Gain
Fast R-CNN	71.8	-	-
Fast R-CNN (2007 data)	66.9	72.4	.6
Fast R-CNN (VGG-M)	59.2	72.4	.6
Fast R-CNN (CaffeNet)	57.1	72.1	.3
YOLO	63.4	75.0	3.2

表 2: 在 VOC 2007 上的模型组合实验。 我们用最佳版本的 Fast R-CNN 检查与各种模型组合的效果。其他版本的 Fast R-CNN 仅提供一个小的提升，而 YOLO 提供了显著的性能提高。

4.4 VOC 2012 结果

在 VOC 2012 测试集上，YOLO 得分为 57.9% mAP。这低于现有技术的当前状态，更接近原始使用 VGG-16 的 R-CNN，参见表 3。我们的系统在小对象检测上与其竞争对手斗争。在类别如瓶、羊和电视/监视器上，YOLO

本的 Fast R-CNN 的组合受益很少。相反，它恰恰是因为 YOLO 在测试时会产生不同种类的错误，这在提高 Fast R-CNN 的性能上相当有效。

不幸的是，这种组合不会从 YOLO 的速度中受益，因为我们分别运行每个模型然后合并结果。然而，因为 YOLO 是如此之快，与 Fast R-CNN 相比，它不增加任何显着的计算时间。

评分比 R-CNN 或者 Feature Edit 低 8-10%。但是，在其他类别如猫和列车上，YOLO 取得更高的性能。

我们结合 Fast R-CNN + YOLO 的模型是最高性能的检测方法的其中之一。Fast R-CNN 从与 YOLO 的组合上得到了 2.3% 的性能改善，在公众排行榜上提升了 5 个点。

4.5 通用性：艺术品中的人体检测

用于对象检测的学术数据集绘制训练和测试相同分布的数据。在现实世界中应用程序中，很难预测所有可能的使用情况，测试

数据可能与系统之前所见的不同 [3]。我们在毕加索数据集[12] 和人物艺术数据集[3]上比较 YOLO

和其他检测系统，这两个数据集是用于测试艺术品上的人物检测的数据集。

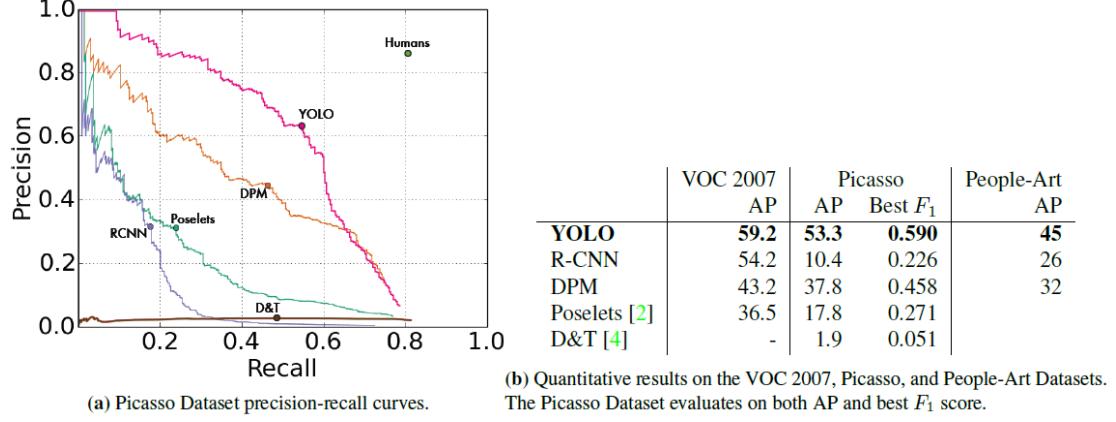


图 5: 毕加索和人 - 艺术数据集的泛化结果。

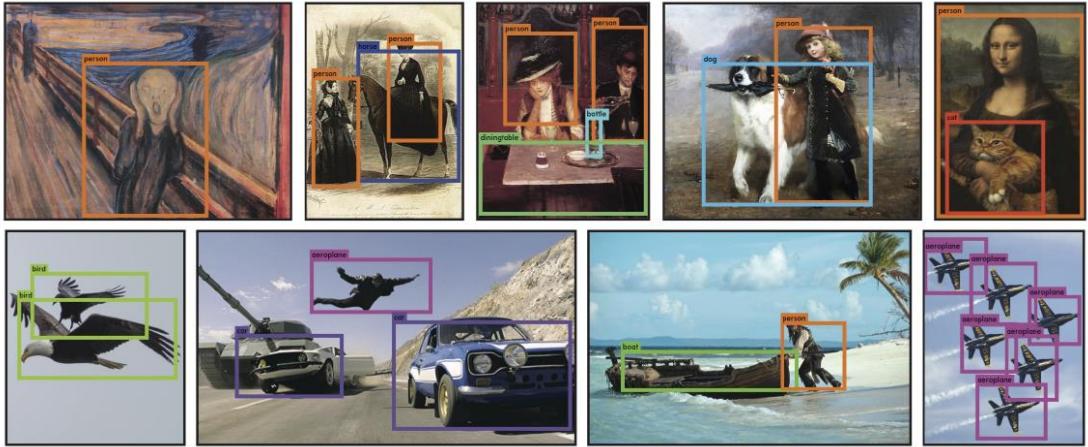


图 6: 定性结果。YOLO 在从互联网的示例图稿和自然图像上运行。它大多数是准确的，虽然它确实认为一个人是一架飞机。

图 5 显示了 YOLO 和其他检测方法之间的性能比较。供参考，我们给出了 VOC 2007 检测在人像上的 AP，所有模型都是仅对 VOC 2007 数据进行训练。毕加索模型在 VOC 2012 上训练，而人 - 艺术在 VOC 2010 上训练。

R-CNN 在 VOC 2007 上具有高的 AP。然而，当应用于艺术品时，R-CNN 的精确度大大降低。R-CNN 对候选边界框使用选择性搜索，候选边界框在自然图像上调整。在 R-CNN 中的分类器

步骤中，仅看到小地区，这里需要好的候选提议。

DPM 在应用于艺术品时保持其 AP 良好。先前的工作理论化地证明，DPM 表现良好是因为它有对象的形状的强大的空间模型和对象的布局。虽然 DPM 不会像 R-CNN 那样降解，但它从较低的 AP 开始。

YOLO 在 VOC 2007 上具有良好的性能，同时，当应用于艺术品时，其 AP 比其他方法降解更少。像 DPM，YOLO 对对象的

大小和形状，以及对象和对象之间经常出现的关系建模。艺术品和自然图像在像素级别上有非常大的不同，但是它们在对象的大小和形状类型上是相似的，因此 YOLO 仍然可以预测良好的边界框和实现检测。

5.野外实时检测

YOLO 是一个快速，精确的对象检测器，是理想的计算机视觉应用。我们将 YOLO 连接到网络摄像头并验证其是否保持实时性能，包括从相机和显示器获取图像的时间和显示检测结果。

结果系统是互动的和可参与的。而 YOLO 单独处理图像，当附加到网络摄像头它表现得像一个跟踪系统，检测对象因为它们移动和改变外观。一个演示系统和源代码可以在我们的项目网站上找到：

<http://pjreddie.com/yolo/>.

6.结论

我们介绍 YOLO，一个统一的对象检测模型。我们的模型构架简单并且可以直接在完整的图像上训练。与基于分类器的方法不同，YOLO 直接在对应的损失函数上训练，这使得检测性能和整个模型的共同训练相对应。

Fast YOLO 在文字检测上是最快的通用对象检测器，YOLO 推动了最先进的实时对象检测。YOLO 也很好地归纳新域使其成为依赖快速，强大的对象检测的应用程序的理想选择。

Acknowledgements：这项工

作部分地得到 ONR N00014-13-1-0720，NSF IIS-1338054 和 The Allen 杰出研究员奖的支持。

参考文献

- [1] M. B. Blaschko and C. H. Lampert. Learning to localize objects with structured output regression. In Computer Vision–ECCV 2008, pages 2–15. Springer, 2008. 4
- [2] L. Bourdev and J. Malik. Poselets: Body part detectors trained using 3d human pose annotations. In International Conference on Computer Vision (ICCV), 2009. 8
- [3] H. Cai, Q. Wu, T. Corradi, and P. Hall. The crossdepiction problem: Computer vision algorithms for recognizing objects in artwork and in photographs. arXiv preprint arXiv:1505.00110, 2015. 7
- [4] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on, volume 1, pages 886–893. IEEE, 2005. 4, 8
- [5] T. Dean, M. Ruzon, M. Segal, J. Shlens, S. Vijayanarasimhan, J. Yagnik, et al. Fast, accurate detection of 100,000 object classes on a single machine. In Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on, pages 1814–1821. IEEE, 2013. 5
- [6] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. arXiv preprint arXiv:1310.1531, 2013. 4
- [7] J. Dong, Q. Chen, S. Yan, and A. Yuille. Towards unified object detection and semantic segmentation. In Computer Vision–ECCV 2014, pages 299–314. Springer, 2014. 7
- [8] D. Erhan, C. Szegedy, A. Toshev, and D. Anguelov. Scalable object detection using deep neural networks. In Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on, pages 2155–2162. IEEE, 2014. 5, 6
- [9] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes challenge: A retrospective. International Journal of Computer Vision, 111(1):98–136, Jan. 2015. 2
- [10] P. F. Felzenszwalb, R. B. Girshick, D.

- McAllester, and D. Ramanan. Object detection with discriminatively trained part based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1627–1645, 2010. 1, 4
- [11] S. Gidaris and N. Komodakis. Object detection via a multiregion & semantic segmentation-aware CNN model. *CoRR*, abs/1505.01749, 2015. 7
- [12] S. Ginosar, D. Haas, T. Brown, and J. Malik. Detecting people in cubist art. In *Computer Vision-ECCV 2014Workshops*, pages 101–116. Springer, 2014. 7
- [13] R. Girshick, J. Donahue, T. Darrell, and J. 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. 1, 4, 7
- [14] R. B. Girshick. Fast R-CNN. *CoRR*, abs/1504.08083, 2015. 2, 5, 6, 7
- [15] S. Gould, T. Gao, and D. Koller. Region-based segmentation and object detection. In *Advances in neural information processing systems*, pages 655–663, 2009. 4
- [16] B. Hariharan, P. Arbel áez, R. Girshick, and J. Malik. Simultaneous detection and segmentation. In *Computer Vision– ECCV 2014*, pages 297–312. Springer, 2014. 7
- [17] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *arXiv preprint arXiv:1406.4729*, 2014. 5
- [18] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012. 4
- [19] D. Hoiem, Y. Chodpathumwan, and Q. Dai. Diagnosing error in object detectors. In *Computer Vision–ECCV 2012*, pages 340–353. Springer, 2012. 6
- [20] K. Lenc and A. Vedaldi. R-cnn minus r. *arXiv preprint arXiv:1506.06981*, 2015. 5, 6
- [21] R. Lienhart and J. Maydt. An extended set of haar-like features for rapid object detection. In *Image Processing. 2002 International Conference on*, volume 1, pages I–900. IEEE, 2002. 4
- [22] M. Lin, Q. Chen, and S. Yan. Network in network. *CoRR*, abs/1312.4400, 2013. 2
- [23] D. G. Lowe. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 1150–1157. Ieee, 1999. 4
- [24] D. Mishkin. Models accuracy on imagenet 2012 val. <https://github.com/BVLC/caffe/wiki/Models-accuracy-on-ImageNet-2012-val>. Accessed: 2015-10-2. 3
- [25] C. P. Papageorgiou, M. Oren, and T. Poggio. A general framework for object detection. In *Computer vision, 1998. sixth international conference on*, pages 555–562. IEEE, 1998. 4
- [26] J. Redmon. Darknet: Open source neural networks in c. <http://pjreddie.com/darknet/>, 2013–2016. 3
- [27] J. Redmon and A. Angelova. Real-time grasp detection using convolutional neural networks. *CoRR*, abs/1412.3128, 2014. 5
- [28] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *arXiv preprint arXiv:1506.01497*, 2015. 5, 6, 7
- [29] S. Ren, K. He, R. B. Girshick, X. Zhang, and J. Sun. Object detection networks on convolutional feature maps. *CoRR*, abs/1504.06066, 2015. 3, 7
- [30] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 2015. 3
- [31] M. A. Sadeghi and D. Forsyth. 30hz object detection with dpm v5. In *Computer Vision–ECCV 2014*, pages 65–79. Springer, 2014. 5, 6
- [32] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *CoRR*, abs/1312.6229, 2013. 4, 5
- [33] Z. Shen and X. Xue. Do more dropouts in pool5 feature maps for better object detection. *arXiv preprint arXiv:1409.6911*, 2014. 7
- [34] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014. 2
- [35] J. R. Uijlings, K. E. van de Sande, T. Gevers, and A. W. Smeulders. Selective search for object recognition. *International journal of computer vision*, 104(2):154–171, 2013. 4
- [36] P. Viola and M. Jones. Robust real-time object detection. *International Journal of Computer Vision*, 4:34–47, 2001. 4
- [37] P. Viola and M. J. Jones. Robust

- real-time face detection. International journal of computer vision, 57(2):137–154, 2004. 5
- [38] J. Yan, Z. Lei, L. Wen, and S. Z. Li. The fastest deformable part model for object detection. In Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on, pages 2497–2504. IEEE, 2014. 5, 6
- [39] C. L. Zitnick and P. Dollár. Edge boxes: Locating object proposals from edges. In Computer Vision–ECCV 2014, pages 391–405. Springer, 2014. 4