|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 分类号 |  | |  | | | 密 级 | |  |
| U D C |  | |  | | | 编 号 | |  |
|  | | | | | | | | |
| 武汉大学logo  **硕 士 专 业 学 位 论 文** | | | | | | | | |
|  | | | | | | | | |
| 标题 | | | | | | | | |
|  | | 研究生姓名 | | ： | 张逸文 | |  | |
| 学号 | | ： | 2017282110364 | |
| 指导教师姓名、职称 | | ： | 何发智 教授 | |
| 专业类别（领域） | | ： | 计算机技术 | |
|  | |  |  | |
|  | |  |  | |
|  | | | | | | | | |
| 二〇一九年四月 | | | | | | | | |

论文原创性声明

本人郑重声明：所呈交的学位论文，是本人在导师指导下，独立进行研究工作所取得的研究成果。除文中已经标明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的研究成果。对本文的研究做出贡献的个人和集体，均已在文中以明确方式标明。本声明的法律结果由本人承担。

学位论文作者（签名）：

年 月 日

摘　　要

随着社会的高速发展，一线技术人员的供给需求日益增长，因此在高等职业教

**关键词：卷积神经网络，目标检测，数据增强**

**ABSTRACT**

With the rapid development of society, the supply need of first-line technical staff is

**Key words: convolutional neural network, Target Detection, Data Augmentation**

目录

[第1章　绪论 1](#_Toc5821404)

[1.1　研究背景及意义 1](#_Toc5821405)

[1.2　目标检测算法 3](#_Toc5821406)

[1.2.1 传统的目标检测算法 4](#_Toc5821407)

[1.2.2 基于深度卷积神经网络的算法 5](#_Toc5821408)

[1.2.3 评价指标 6](#_Toc5821409)

[1.3　经验风险最小化准则和领域风险最小化准则 6](#_Toc5821410)

[1.4　本文的研究内容 7](#_Toc5821411)

[1.5　本文的组织结构 8](#_Toc5821412)

[第2章　目标检测算法的对比和分析 9](#_Toc5821413)

[2.1　传统的目标检测算法 9](#_Toc5821414)

[2.1.1 HOG(方向梯度直方图) 9](#_Toc5821415)

[2.1.2 SIFT(尺度不变特征转换) 9](#_Toc5821416)

[2.1.3 SURF(加速稳健特征) 10](#_Toc5821417)

[2.2　基于卷积神经网络的算法 11](#_Toc5821418)

[2.2.1 基于区域建议的目标检测算法 11](#_Toc5821419)

[2.2.1.1 R-CNN 11](#_Toc5821420)

[2.2.1.2 Fast-RCNN 12](#_Toc5821421)

[2.2.2 基于回归的目标检测算法 13](#_Toc5821422)

[2.2.2.1 YOLO 13](#_Toc5821423)

[2.2.2.2 SSD 13](#_Toc5821424)

[2.2.3 基于搜索的目标检测算法 14](#_Toc5821425)

[2.2.3.1 AttentionNet 14](#_Toc5821426)

[2.3　两类算法的分析 15](#_Toc5821427)

[2.4　本章小节 16](#_Toc5821428)

[第3章 数据增强技术 17](#_Toc5821429)

[3.1　监督学习理论 17](#_Toc5821430)

[3.2　经验风险最小化和领域风险最小化 17](#_Toc5821431)

[3.2.1 经验风险最小化和结构风险最小化 17](#_Toc5821432)

[3.2.2 领域风险最小化 18](#_Toc5821433)

[3.3　Mixup(混合)技术与数据增强 18](#_Toc5821434)

[3.3.1 大型深度神经网络的缺陷 18](#_Toc5821435)

[3.3.2 ERM准则的替代方案 19](#_Toc5821436)

[3.3.3 使用Mixup技术 19](#_Toc5821437)

[3.4　本章小结 20](#_Toc5821438)

[第4章　将Mixup技术运用到Faster-RCNN算法 21](#_Toc5821439)

[4.1　Faster-RCNN 21](#_Toc5821440)

[4.1.1 Conv layers层 21](#_Toc5821441)

[4.1.2 RPN网络 21](#_Toc5821442)

[4.1.3 ROI-Pooling层 23](#_Toc5821443)

[4.1.4 Classification层 23](#_Toc5821444)

[4.1.5 基础网络 23](#_Toc5821445)

[4.1.5.1 resnet101网络 23](#_Toc5821446)

[4.1.5.1 VGG16网络 24](#_Toc5821447)

[4.2　运用Mixup技术到Faster-RCNN 25](#_Toc5821448)

[4.2.1 混合输入图片 26](#_Toc5821449)

[4.2.2在网络中混合 26](#_Toc5821450)

[4.2.2.1 RPN网络 26](#_Toc5821451)

[4.2.2.2池化层前的混合 27](#_Toc5821452)

[4.2.2.3 res101网络中的混合 27](#_Toc5821453)

[4.2.2.4 VGG16网络中的混合 28](#_Toc5821454)

[4.2.3 Mixup Loss 28](#_Toc5821455)

[4.3　本章小结 29](#_Toc5821456)

[第5章　实验 30](#_Toc5821457)

[5.1　PASCAL VOC数据集 30](#_Toc5821458)

[5.1.1 PASCAL VOC挑战赛 30](#_Toc5821459)

[5.1.2 PASCAL VOC数据集的介绍 31](#_Toc5821460)

[5.1.3 PASCAL VOC数据集的制作 32](#_Toc5821461)

[5.1.4 PASCAL VOC数据集的使用 33](#_Toc5821462)

[5.2　参数设置 33](#_Toc5821463)

[5.2.1 的生成 33](#_Toc5821464)

[5.2.2 原始的Faster-RCNN模型的主要参数 34](#_Toc5821465)

[5.2.3 改进模型的主要参数 35](#_Toc5821466)

[5.3　模型训练 35](#_Toc5821467)

[5.3.1 Faster-RCNN原始模型 35](#_Toc5821468)

[5.3.2 RCNN\_ONLY策略 35](#_Toc5821469)

[5.3.3 RPN\_ONLY策略 36](#_Toc5821470)

[5.3.4 RCNN+RPN策略 36](#_Toc5821471)

[5.4　实验结果 36](#_Toc5821472)

[5.4.1 实验平台 36](#_Toc5821473)

[5.4.2 原始模型的结果 37](#_Toc5821474)

[5.4.3 RCNN\_ONLY策略 37](#_Toc5821475)

[5.4.4 RPN\_ONLY策略 38](#_Toc5821476)

[5.4.5 RCNN+RPN策略 38](#_Toc5821477)

[5.5　实验结果的分析 39](#_Toc5821478)

[5.6　本章小节 39](#_Toc5821479)

[第6章　改进模型的实际应用 40](#_Toc5821480)

[6.1　CDnet数据集 40](#_Toc5821481)

[6.2　在CDnet数据集上的比较与分析 41](#_Toc5821482)

[6.1.1 Baseline 41](#_Toc5821483)

[6.1.2 Dynamic Background 41](#_Toc5821484)

[6.1.3 Camera Jitter 41](#_Toc5821485)

[6.1.4 Shadow 42](#_Toc5821486)

[6.1.5 Intermittent Objection Motion 42](#_Toc5821487)

[6.1.6 Thermal 42](#_Toc5821488)

[6.1.7 PTZ 42](#_Toc5821489)

[6.1.8 Bad Weather 42](#_Toc5821490)

[6.1.9 Low Frame-Rate 42](#_Toc5821491)

[6.1.10 Night Videos 42](#_Toc5821492)

[6.1.11 Air Turbulence 42](#_Toc5821493)

[6.3　本章小节 42](#_Toc5821494)

[第7章　总结与展望 43](#_Toc5821495)

[7.1　总结 43](#_Toc5821496)

[7.1.1 目标检测算法的发展 43](#_Toc5821497)

[7.1.2 深度卷积神经网络的不足与改进 43](#_Toc5821498)

[7.1.3 改进模型的实际运用 43](#_Toc5821499)

[7.2　展望 43](#_Toc5821500)

[7.2.1 的控制 43](#_Toc5821501)

[7.2.2 加入新的正则化形式 43](#_Toc5821502)

[7.2.3 改进边界框回归的方式 44](#_Toc5821503)

第1章　绪论

1.1　研究背景及意义

人们日常使用的数据多种多样，可以分为文本数据、语音数据、图像数据等。在这之中图像数据表达直观、容易理解、形式多样且内容丰富。随着视频图像采集设备日渐普及，通信网络的不断升级，人们主要使用的数据将是以图像的形式展现。因此，对于图像的分析显得尤为重要。人类可以通过双目视觉准确的分辨场景中的对象并精确确定其位置，可以快速而准确的理解视频或图像中的信息。现在，是否计算机能够做到这一点？目标检测就是解决这个问题的重要一环。

目标检测检测是计算机视觉的核心问题之一，既是研究热点又是研究难点。其主要任务[[[1]](#endnote-2)][[[2]](#endnote-3)]就是将感兴趣的目标(ROI)从视频或者是图片中检测出来，并能够使用一定的方法如绑定框，着色突出等方法将ROI从中分割提取出来。运用目标检测技术，可以得到关于ROI的形状，位置，运动趋势等重要形象。检测出的结果是的肉眼对于视频及图像的观察更为容易，根据得到的关于ROI运动信息也会对后续的分析提供帮助。

国内外研究人员已经多目标检测算法做了大量的研究，但任然有许多挑战有待解决，主要有：

1. ROI在不同状态下有不同的形状，姿态和外观。ROI有时会和场景中的其他目标有相似的特征。例如：在雪地中身穿白色衣服的行人，由于衣服的颜色和雪的颜色相近，所以行人的特征将会变得和雪地的特征十分类似。优秀的目标检测算法要能够识别出不同特征间的细小差异，从而避免遗漏。
2. 光照变化还有阴影问题。在不同的光照条件下，ROI会有不同的颜色，有时也会被场景中其他物体的阴影所覆盖。这会造成ROI被误检测为背景，而阴影却被认为是前景物体。
3. 场景的遮挡，摄像机的抖动还有噪声等因素。由于最终的应用场景是识别现实世界的对象，就存在ROI和背景都可能是处在运动状态的情况，两者会相互影响，干扰识别。
4. 缺少对应的数据集。检测特定的ROI就需要用特定的数据集进行训练，目前来看数据集的标签(Labels)，原标记框(Ground Truth)等都需要人为标注，费时费力。
5. 周期性运动问题。收到图像帧率的影响，物体运动的周期性会带来两张可能。如果目标运动的足够快，如旋转的风扇，那么就很有可能被判别为一个圆。

这些问题都不同程度的影响的目标检测检测算法的精度、速度，还有同一时间检测ROI的数量。

目标检测算法的精度对于人们利用其信息进行分析预测有很大对影响，更为重要的一点，通过检测ROI便可抛弃我们不关心的像素点，这为后续处理减少了大量的计算量。但是优秀的目标检测算法都要面对在精度和速度上的妥协问题，好的算法都是既能满足检精度要求，又能快速计算，满足可以对视频进行检测的要求。目前，大多数目标检测检测算法都着眼于提高精度和速度。

近年来，由于硬件设备的不断跟新，尤其是GPU的计算能力不断提升，使得PC的性能不断增强。从1999年英伟达公司推出了第一款GPU——Nvidia Vanta，到2018年RTX2080TI问世，GPU的运算能力达到了惊人的每秒10.1T次浮点数计算。计算机算力的提升使得使用深度卷积神经网络[[[3]](#endnote-4)]的算法不断涌现。2012年在ImageNet[[[4]](#endnote-5)]图像识别大赛中杰弗里·辛顿的模型[[[5]](#endnote-6)]夺得冠军，其采用的ReLU[[[6]](#endnote-7)]函数解决了梯度消失的问题。同年，在吴恩达的带领下其团队又一次使用深度卷积神经网络夺得了ImageNet大赛的冠军，并成功将错误率由原先的26%降低到15%。之后的ImageNet大赛中深度卷积神经网络独领风骚。深度卷积神经网络的蓬勃发展得到了各界的广泛关注，尤其是在计算机视觉领域。其中就有一部分人在探索基于深度卷积神经网络的目标检测算法。人工智能的加入大大提高了算法的识别准确度。在国家大力发展人工智能的趋势下，这类算法将成为今后的主流方向。

在早期的深度学习或者是机器学习中，数据集一直是一个困扰大家的问题。图像数据的建立是一个非常复杂和繁琐的过程。首先要取得合适的样本图像，要满足算法的需求，又不能是被刻意制造出来的。然后确定图像的尺寸，其次还要确定图片中包含的对象的类别及其边界框，最后将这些数据整理成对应的注释文件。数据增强(data augmentation)[[[7]](#endnote-8)]是一种非常简单的增加训练数据的方式。通常的数据增强方式有：旋转、剪裁、缩放和模糊。通过这些简单的处理，就可以成倍的增加训练集的容量，相对于较小的数据集，此方法通常都是首选。但是这种方式显然不是最好的解决方式。而且，经过了多年的研究，直到最近人们对于数据增强的原理又有了全新的认识。

随着深度学习的不断发展，人们对于大型深度卷积神经网络的理解不断深入。许多在机器学习中的经典理论不断被否定，新的理论和解释不断出现，使得人们对于神经网络的训练不断优化，最终得到的模型性能更加优秀。已经有人意识到[[[8]](#endnote-9)][[[9]](#endnote-10)]数据增强是一种控制模型复杂度的手段，而不单单只是增加训练数据。经典的监督学习理论[[[10]](#endnote-11)]都假设训练集和测试集独立同分布抽样(independent identically distributed)。数据增强增强了对于某些变换的不变性，即减少模型的方差。所以在实施数据增强之后，模型的复杂度降低了。这样，提高基于深度神经网络的目标检测算法精度的方法就多了一种——数据增强。在不改变现有模型的情况下，提高精度的最有效办法就是数据增强，而且简单的数据增加所引入的计算负担微乎其微，故这方面今后也将成为研究的热点。

目标检测技术最初的动力来自军事及安全领域，运用此技术可以提高军事系统的检测智能性以及可靠性，辅助士兵作战。随着新的目标检测算法的不断提出，还有智能手机，数码相机等视频采集设备等大众化，目标检测技术等应用领域也变得越来越多。在道路交通方面，行人检测可用来统计客流量。车辆检测、自动驾驶都需要用到目标检测算法来识别道路，周遭环境。在工业领域，可以进行缺陷检测，比如墙面裂纹，电路板表面缺陷。在农业方面，能够识别害虫和作物的病变，及早预警，降低损失。自动识别医学图像中的肿瘤等肉眼难以识别等微小病灶，争取更多治疗时间。

近年来，随着目标检测的实际运用领域不断扩大，其面临大挑战也越来越多。于此同时，人们的要求也越来越高。随着人工智能技术的不断普及，智能化的目标检测算法将在无人驾驶、社会安全、物流等领域大放异彩。有利于建设智慧城市，有利于推进我国的现代化建设，对国民经济与社会的平稳发展具有重大意义和深远影响。

1.2　目标检测算法

目前有两大类目标检测算法，其一是传统目标检测算法，其二是基于深度卷积神经网络的目标检测算法。近年来深度卷积神经网络目标检测算法的表现都优于传统目标检测算法，尤其是在各类目标检测大赛上，传统方法几乎没有了身影。

1.2.1 传统的目标检测算法

虽然卷积神经网络在很早之前就已经被提出。但是在早期由于深度学习理论的不完善，以及训练数据的稀少等原因使得其通过深度学习解决现实问题能力远不如传统方法。而且，还有一个关键的问题就是当时使用神经网络训练一个模型将花费数周甚至是一月的时间，时间成本太高。所以，在上个世纪时，大多数经验学者都专注于传统目标检测算法的研究。其典型算法有：SIFT [[[11]](#endnote-12)]、HOG [[[12]](#endnote-13)]、SURF [[[13]](#endnote-14)]。

传统方法一般可以分为两大类：

1. 基于滑动窗口的算法。首先用不同大小及比例的滑动窗口提取图像中的特征信息。然后，给提取到的特征进行距离计算，确定其属于哪个标签，并打分。最后进行非最大化抑制[[[14]](#endnote-15)]，找出最优的滑动窗口，即为检测结果。
2. 人工设计目标特征算子(descriptor)的算法。这些算法主要流程是：第一，选取兴趣点，主要用到海塞矩阵或者是高斯算子进行特征提取。第二，将提取的特征转化为自己的特征算子。不同算法设计不同的特征算子，主要是为了应对不同的光照条件和形状变化、旋转等。第三，根据特征算子进行分类。主要是用SVM[[[15]](#endnote-16)]、聚类算法等进行分类。

滑动窗口算法这种类似穷举的策略没有任何针对性，许多窗口的重叠率很高，在计算时会有大量的重复计算。而且受旋转，光照变化的影响较大。而特征算子是经过精心设计的，鲁棒性相比于滑动窗口算法有所提升。大量特征最后都会被归到有限数量的特征算子中，减少了大量的重复计算。传统方法只是针对单帧图像进行提取，设计特征算子时主要是考虑像素间的数学关系，难免会忽视了其他联系，因而被提取特征能力更全面的深度学习算法逐渐取代。对于如何设计优秀的特征算子，并没有系统性的指导方法。所以，只有在此领域研究多年的专家才能胜任此项任务。在早期时，由于大部分学者都专注于传统目标检测算法，故其可以快速的发展。现在，随着反向传播、激活函数、正则化等理论的不断推出，深度学习算法的理论不断完善，从另一方面进一步挤压传统目标检测算法的发展。

1.2.2 基于深度卷积神经网络的算法

[[[16]](#endnote-17)]自1991起由于反向传播的梯度消失问题无法解决，使得深度卷积神经网络的头部网络层无法有效的进行参数学习，深度学习的热潮降到又一次退去。2011年，ReLU激活函数被提出有效的抑制了梯度消失问题。再上AlexNet以碾压SVM方法(第二名)的优势夺得ImageNet图像识别大赛的冠军，深度学习迎来了又一个新的爆发时期。采用深度学习进行图像检测也成为流行的趋势。现有的基于深度学习的目标检测算法大致可以分为三类：

1. 基于区域建议的目标检测算法，如R-CNN [[[17]](#endnote-18)],Fast-RCNN[[[18]](#endnote-19)],Faster-RCNN[[[19]](#endnote-20)],Mask-RCNN[[[20]](#endnote-21)],R-FCN[[[21]](#endnote-22)]等。
2. 基于回归的目标检测算法，如YOLO[[[22]](#endnote-23)],SSD[[[23]](#endnote-24)]等。
3. 基于搜索的目标检测，如AttentionNet[[[24]](#endnote-25)]等。

2014年，由Ross Girshick提出了R-CNN算法，作为这个系列的开山之作。该系列算法主要流程是先对输入图像进行预处理，产生一定数目的区域建议(Region proposal)，然后将这些区域建议送入VGG16或者res101等优秀的分类网络中提取图像特征，最后使用SVM进行分类识别。之后的文章如Fast-RCNN对区域建议的分类做了优化，共享了计算量大的部分，提高了效率。Faster-RCNN使用RPN网络代替Selective Search[[[25]](#endnote-26)]算法，提升了区域建议的质量，进一步共享了的特征提取的计算量，大幅提高了算法的时间性能。这一系列算法在准确度方面较高，但是牺牲了训练和识别的速度，还未能达到实时检测的要求。

2015年提出的SSD算法是另一种基于回归的算法。其核心是在特征图[[[26]](#endnote-27)]上采用多种大小的卷积核来预测一系列默认边界框的类别分数和偏移量。一个特征图含有多个默认框(default boxes)，每个默认框计算c个类别的分数和4个坐标参数的偏移量，让后进行卷积计算得到每个类别的置信度和位置。然后进行匹配的到正负样本，用于指导学习走向。

YOLO算法也是2015年提出的算法。该算法只使用单个神经网络，将图片分割成S×S的网格每个单元格检测中心点落在其中的目标，得到对应的边界框和置信度。这两种算法有速度优势，但是准确度不及R-CNN系列。

AttentionNet是通过一种基于搜索的算法。每次迭代都会将包含目标的区域进行剪裁，舍弃与目标对象无关的部分，这样使得其边界框和置信度随着每次迭代都更加准确，这种方法将多个弱检测聚合为一个强检测。整个网络中都没有没有区域建议和边界框回归。其思想与之前提到的算法都不相同。

随着深度学习的不断发展，基于深度卷积神经网络的目标检测算法也变得更加优秀。由于使用卷积神经网络提取的特征与人眼的视觉机理十分类似，都是从局部到整体对特征进行感知，提取特征的能力全面而强大。而且，随着高性能GPU的不断发展，计算机的计算能力不断加强，深度更深结构更庞大的的卷积神经网络将更好的为目标检测服务。

近两年，目标检测和识别技术火热发展，新的算法不断被提出。最好的目标检测算法在公开的数据集上有着跨越式的进步，算法性能在不断接近人类的能力。

1.2.3 评价指标

这些通用的评价指标都是来自[**[[27]](#endnote-28)**].

**TP** (true positive)，实际为正样本，检测正确的样本。

**FP**(false positive)，实际为负样本，误检测为正确的样本。

**TN**(true negative)，实际为负样本，检测为错误的样本。

**FN**(false negative)，实际为正样本，误检测为错误的样本。

**召回率(Recall)** = TP / （TP + FN）

**准确率(Precision)** = TP / （TP + FP）

由**Reacll**和**Precision**构成PR曲线，求积分即是**AP（Average Precision）**

(1)

(2)

对**AP**求均值就是**mAP**。

**IoU(Intersection over Union)**，交并比。边界框和原标记框的交叠率。

1.3　经验风险最小化准则和领域风险最小化准则

Mixup(混合)技术是一个新颖的数据增强技术，主要是基于以下两个准则：

1. 经验风险最小化(ERM[[[28]](#endnote-29)][[[29]](#endnote-30)])。这是统计学习理论中的一个准则，其定义了一系列的学习算法，并用来估计其性能的理论极限。我们通常使用的深度学习算法的一般步骤都是将作为输入得到结果，然后将与Y进行比较，不断修改参数最小化其期望风险。而我们使用的训练数据又叫经验数据，用其计算得到的误差就是经验风险。用经验风险模拟期望风险并不断逼近其理论边界的过程就是ERM。
2. 领域风险最小化(VRM[[[30]](#endnote-31)])。不断缩小经验误差，可能会导致在训练样本上有很好的表现，但在实际的运用中表现不佳，这就是过拟合了。这是就需要VRM来解决问题。主要是通过使用领域函数构造出最邻近 (认为是和X同类，服从同一分布)来进行训练。

多年来，人们对于深度卷积神经网络是如何工作的，一直无法给出合理的解释。鉴于深度学习是机器学习的一个分支，人们使用ERM理论解释深度卷积神经网络。但是，似乎ERM理论也不能完成这项工作。深度卷积神经网络目前依然是一个黑盒结构，由此派生出了VRM理论弥补了ERM理论的不足。

在以上两个准则的指导下，使用数据增强函数构造新的训练数据。用构造的数据训练可以抵抗大型卷积神经网络的记忆性，减少过拟合，提高泛化性。

1.4　本文的研究内容

本论文将以基于深度卷积神经网络的目标检测算法和数据增强力量为基础。复现了两个时下流行的算法——Faster-RCNN算法和Mixup算法，寻找两者的结合点。通过混合对数据集进行增强，克服Faster-RCNN算法中的卷积神经网络对于数据集的记忆性问题以及对于错误标签的敏感性问题等，增强Faster-RCNN算法的鲁棒性，提高其分类和检测的效果。主要内容包括包括：

1. 研究多种目标检测算法的基本原理。指出它们的不足之处，在不大幅改变网络结构的情况下寻找可行的解决方案。
2. 研究Mixup算法的基本原理，找到其能够改进这些目标检测算法通病的理论解释，并找到Mixup算法的代码实现方式。
3. 复现一个流行的目标检测算法，以Faster-RCNN算法为例。通过微调网络结构，将Mixup技术运用到目标检测算法之中。训练和生成原始模型和改进后的模型。
4. 通过对比实验分析加入Mixup后是否能提高Faster-RCNN算法的精度。以及通过不同的尝试找到最优的Mixup的方式。实验结果表明，在参数合适的情况下，在某些地方加入Mixup之后Faster-RCNN算法会有稳定的提升，于此同时引入的额外开销几乎没有。
5. 在生成好的多个改进模型中选择一个在测试集上表现最优的，将其用于实际的目标检测中，与原始的Faster-RCNN模型进行对比，观察其是否可以运用到实际生活中，分析其适合于不适合的现实场景。
6. 总结Mixup能成功增强Faster-RCNN算法的原因，提出需要完善之处，以及对今后工作的展望。

1.5　本文的组织结构

第一章 绪论。本章主要引入本课题的研究背景与研究意义，简要介绍一些相关的目标检测算法和其使用的评价指标，最后说明了本文的研究内容与组织结构。

第二章 目标检测算法的对比和分析。本章详细介绍了各种目标检测算法的原理和网络结构，包括传统的和非传统的基于深度神经网络的目标检测算法。最后，对这两类算法进行了总结和比较。

第三章 数据增强技术。本章首先介绍了监督学习问题的一些基础理论，由此引入了ERM准则和SRM准则，目前深度神经网络主要遵循的两个准则，分析了它们为何被广泛采纳以及其有哪些不足之处。之后，提出替代方案——VRM准则，并给出了VRM准则的实际实现——Mixup数据增强技术。

第四章 将Mixup技术运用到Faster-RCNN算法。本章首先介绍了本文复现的Faster-RCNN的具体结构。之后指出了本文是如何改进Faster-RCNN网络，将Mixup技术融合进Faster-RCNN中去的。

第五章 实验。本章介绍了PASCAL VOC数据集，其优秀之处在哪里，为何选用其进行使用。之后，详细说明了本文是如何训练不同的模型。本章提出了三种不同的训练策略，并根据这些策略训练出了改进后的新模型。最后，做了多组对比分析实验，最终得到实验结论。

第六章 实际应用。本章选择了一个在测试集上表现最优的模型，进行实际检测，与复现的原始Faster-RCNN算法的模型进行比较。选择的数据集是CDnet因为其更贴近实际。

第七章 总结与展望。本章首先对本文所做的工作进行了总结，提出了创新点，随后，对未来的工作进行了展望，分析了未来应该改进的不足。

第2章　目标检测算法的对比和分析

2.1　传统的目标检测算法

2.1.1 HOG(方向梯度直方图)

Historgrams of Oriented Gradients(HOG)由Navneet Dalal和Bill Triggs于2005年提出，其在人体检测方面表现很优秀。HOG的主要思路是：即便是我们不能精确的获悉相关梯度或者是边缘的位置，局部对象的外表以及形状可以被局部强度梯度的分布或者是边缘的方向相当好的表征出来。为此提出了一种基于梯度和边缘的特征算子，并使用SVM作为贯穿整个过程的基线分类器。HOG算法可以分成六部分：

1. 伽马/颜色修正。这一步操作对整体性能的影响不大，因为之后的特征算子计算时也会进行一次规范化，主要是为了降低阴影以及光照变化的影响。
2. 梯度计算。主要是用了高斯平滑，使用大的掩膜可以提高检测结果，但会降低性能。在检测彩色图像时，每个通道的图像都会单独计算，最后取模最长的向量为该点的梯度向量。
3. 空间/方向直方图计算。每一个像素根据以自身为中心的梯度的方向为边缘方向直方图投票(加权投票)，这些投票会被累积到一个局部区域(称为cells)的方向箱子中。在相邻的箱子中间的选票会先进行双线性插值以减少混淆。
4. 归一化以及特征算子生成。归一化是为了克服光照变化造成的对比度大范围变化。将cells组成更大的区块，每个区块将单独归一化。最终生成的特征算子就是由检测窗口中所有归一化之后的区块中的cells的向量组成。
5. 窗口检测。检测窗口大小为64×128。
6. SVM分类。使用SVM对检测窗口中的向量组进行分类。

2.1.2 SIFT(尺度不变特征转换)

Scale Invariant Feature Transform(SIFT)。该算法由David Lowe于1999年提出，累积被引用25000多次。SIFT算法中的提出了一种具有在尺度发生变化时保持不变的局部特征特征算子。采用层叠过滤的方法，加快了提取特征的速度。

SIFT算法的流程如下：

1. 探测尺度空间的极值。在尺度空间中使用高斯函数作为核函数，这就会导致有许多极值存在的情况。层叠过滤方法先初步找出候选位置，在进行进一步确认。首先将采样点和其相邻的26个点进行比较，确定其是否为这27个点中点极值。然后在不同的尺度空间采样，找出那些不变的点作为候选点。
2. 定位兴趣点(关键点)。通过精细拟合候选点的位置、尺度、主曲率，淘汰对噪音敏感的点和边角的点。因为对比度低的点会受到噪声的干扰而影响检测准确性，所以需要被去除。但是这还不够，边角处的点在高斯差分函数里也会有很大的响应，故需要排除边角的响应。
3. 兴趣点定向。通过计算兴趣点的属性为其指定一个方向，特征算子将会与这些方向相关联，从而具有在图像旋转之后特征值依然不变性的能力。将兴趣点及其附近点的梯度方向构成一张梯度直方图，图的峰值就是梯度的主方向。在最高峰的80%范围内的其他峰值也分配方向，但最终只有少数点会被指定有多个方向。
4. 由兴趣点生成局部图像的特征算子。生成的特征算子对于光照和观察点的变化，会尽可能的保持不变。首先计算兴趣点局部区域的梯度方向和大小，再由高斯函数进行加权投票，得到一个局部矢量的结果。为了具有方向不变性，特征算子的坐标和梯度方向都会基于关键点进行旋转。最终特征算子就是由局部区域内所有兴趣点的方向直方图中的矢量组成。
5. 特征匹配。首先会生成一个SIFT特征数据库，这些特征在第一次从训练图像中提取出后存入数据库。检测时将所有特征与数据库内特征进行比较，计算其欧式距离。当有三个及以上的特征于某一个类匹配时就可以进行进一步确认，通过精细匹配得出最终结果。

2.1.3 SURF(加速稳健特征)

Speed-Up Robust Features(SURF)是对SIFT算法的改进，主要是对特征的提取和特征算子进行优化。特征提取更加高效，特征算子对于尺度变化和旋转变化更具鲁棒性。实现流程如下：

1. 海塞矩阵(Hessian Matrix)。本算法利用海塞矩阵提取所有兴趣点的特征，并且使用积分图像减少计算量。使用这种盒子滤波器来近似的代替二阶微分，利用积分图像就可以快速计算卷积的结果，跟卷积核的尺寸无关。
2. 构建尺度空间，是关于不同尺度的特征金字塔。由于使用了盒子滤波器以及积分图像进行计算，故可以使用对应尺寸的滤波器并行计算不同层的图像。
3. 兴趣点定位。在3×3×3的领域空间内运用非最大化抑制找到27个点中的极值。
4. 兴趣点主方向匹配。在6s(s为一个计算单位)的计算半径内统计一个60度扇形区域内，x方向(水平)以及y方向(垂直)上的哈尔 (Haar)小波响应总和。最后，进行高斯加权得到两个最终分量构成候选方向矢量，则主方向就是模最长的那个矢量。
5. 生成兴趣点特征算子。在一个以兴趣点为中心的方形领域内，将此区域分割成16个子区域，统计所有像素的哈尔小波响应，此操作同上一步，只不过主方向是上一步计算的方向，像素的响应的方向也需随之发生变化。只统计4个值、、、，大大减少了计算量。为了进一步提高计算速度，旋转操作是在计算出响应结果后进行的。
6. 兴趣点匹配。使用兴趣点的拉普拉斯(Laplacian)符号(即海塞矩阵的迹)匹配，只匹配符号相同的特征点，这样可以加快匹配速度。

2.2　基于卷积神经网络的算法

2.2.1 基于区域建议的目标检测算法

2.2.1.1 R-CNN

Regions with CNN features(R-CNN)算法由是由Ross Girshick于2014年提出，是第一个将深度学习应用到目标检测领域的算法。其基本思路和传统方法相同，都是提取局部特征，进行分类，最大化抑制，最终得到结果。不同点是在提取特征时使用的是深度卷积神经网络，而不是自行设计的特征算子提取方法。

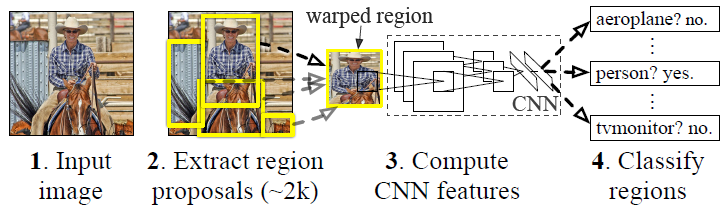
该模型主要有三个模块组成，其流程如图2 – 1所示：

图2 - 1 RCNN算法的流程图

1. 区域推荐模块(Region proposal)。使用的是Selective search算法的快速模式。首先，将图像分割成多个局部区域。之后，将两个可能性最高(颜色、纹理相近的)的区域合并，直到只有一个区域。在这个过程中所有曾经存在的区域都是一个区域建议(proposal)。在每张图片上产生约2000个类别无关的候选区域。
2. 特征提取模块(Feature extraction)。将得到的区域建议进行放缩后统一输入深度神经网络中，最终的输出就是特征就相当于传统算法的特征算子。R-CNN算法使用的基础网络是Hinton在2012年Image Net上的分类网络。
3. SVM分类模块。将特征输入到分类器中得到评分。对于同一个原标记框可能有多个重叠的区域获得高分，使用NMS(非最大化抑制)将和最高分者重复率高的结果去除。在SVM分类器之后是边界框回归(Bounding Box-Regression)部分，其最终的输出是预测窗口的4维向量(x，y，w，h)，使得预测对象可以在图像中被方框标出。其中x、y是中心点的坐标，w、h是边界框的高宽。

R-CNN算法的网络如图2 - 2所示。在pool5之前都是卷积层，用来提取特征。最后是两个池化层，用来恢复输出，组装所以局部特征。使用ReLU函数作为激活函数，可以解决梯度消失的问题。norm层用来简化计算，加速训练。

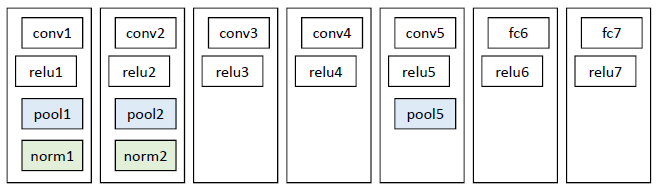


图2 - 2 R-CNN算法的网络结构图

2.2.1.2 Fast-RCNN

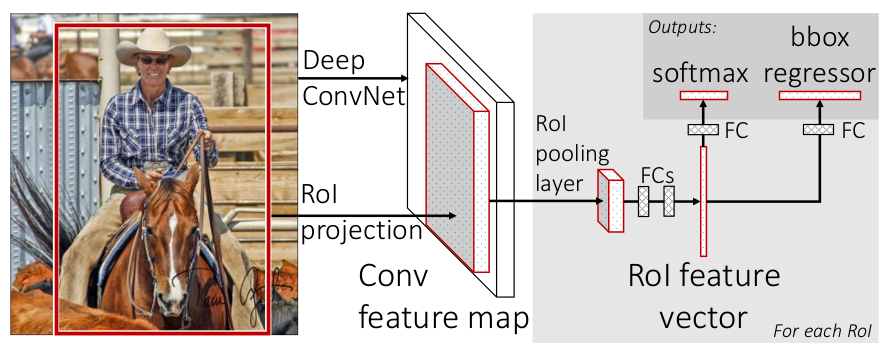
Fast-RCNN算法是对R-CNN算法对改进，也是出自Ross Girshick之手，是其在受SPPnet[[[31]](#endnote-32)]启发后于2015年提出的。主要改进如下，如图2 - 3所示：

图2 - 3 Fast-RCNN算法的流程图

每一个区域建议都要进入卷积神经网络中，2000个区域建议就要计算2000次，这里面有些部分是重复计算。对此的改进就是只对全部图像做一次特征提取操作得到特征图(feature map)，之后通过Roi pooling层找到每个区域建议在这张总特征图上的映射结果，以此作为之后层的输入。这样可以节省大量的计算时间。这样改进的优点是：神经网络的输入大小是固定的，在R-CNN算法中需要重新调整大小，这样就不可避免会造成失真。Roi pooling层是简化的金字塔池化层，通过这个金字塔池化层任意大小的输入最后都可以变成一个7×7特征向量。

整个过程不是端到端的而是分成了三个阶段。但是使用Softmax函数做分类，同时把边界框回归也放到同一个阶段。fc7层的输出同时作为分类和回归的输出。这两个分支的损失(loss)统一汇入到一个多任务损失函数(Multi-task loss)，使得可以共享卷积特征相互促进，提升结果。公式(3)表示Multi-task loss：

新的网络的输入依然是由Selective Search得到的2000个区域建议。pool5层替换为金字塔池化层，其输入是由区域建议得到的ROI特征和整个图片的特征，输入到全连接层进一步组合特征。这样修改之后Fast-RCNN变成了single-stage，不需要存储额外的特征，同时计算速度大幅提升。

2.2.2 基于回归的目标检测算法

2.2.2.1 YOLO

You Only Look Once(YOLO)算法的作者认为像R-CNN这里基于区域建议的算法有多个流程，每个流程都很复杂，都需要进行单独的训练，所以速度很慢，需要进行针对性的优化。所以可以把目标检测框架看作是一种回归问题，直接由像素得到边界框和类概率。只需要在图像上看一次，就能得出预测的结果。

单个卷积网络同时预测多个检测盒子。一个盒子可以对应多个边界框和类概率。将图像分成S×S的网格，凡是中心落日某个网格中的目标都由该网格复杂预测其边界框和类置信度。YOLO的网络结构类似GoogLeNet，24个卷积层后加上2个全连接层，分别做特征提取和结果预测。最后预测出的边界框的坐标是相对于网格中心的偏移量，高度和宽度都被规范为相对于网格高宽的一组比例值，故(x,y,w,h)四个结果都在0到1之间。图2 - 4为YOLO算法的网络结构：

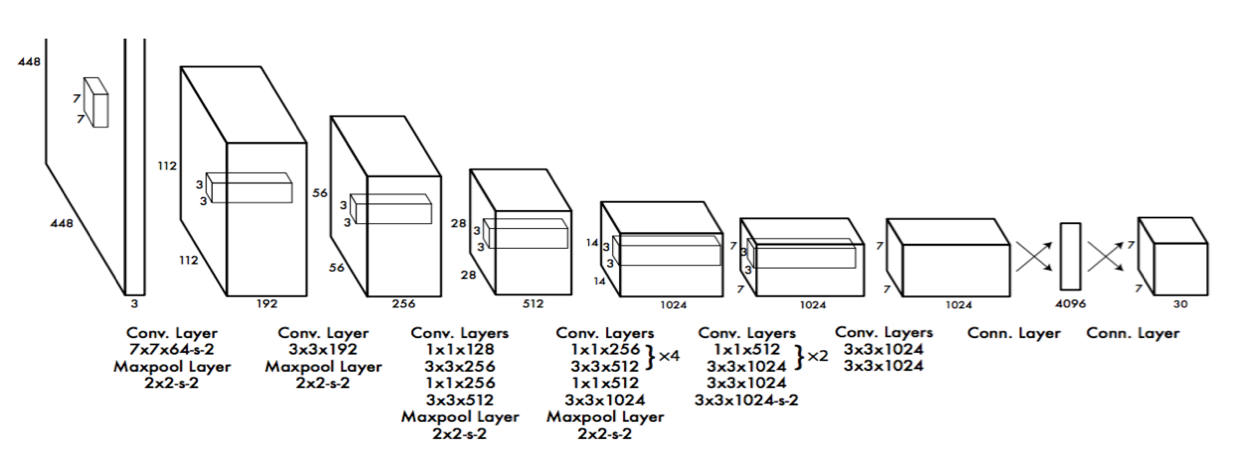


图2 - 4 YOLO算法的网络结构

YOLO算法缺点在于对于边界框的预测施加了较强的空间约束，每个网格只计算2个检测盒，只识别一个类。这样就限制了预测目标的数量。由于学习的是如何从数据中得到边界框，所以泛化性不好，对于不常见的对象很难准确预测。YOLO相比于其他检测算法的优势在于速度，可以达到45帧/秒，完全可以做到实时检测，代价就是牺牲了精度。目前，一些直播网站上已经YOLO算法的实际应用了。

2.2.2.2 SSD

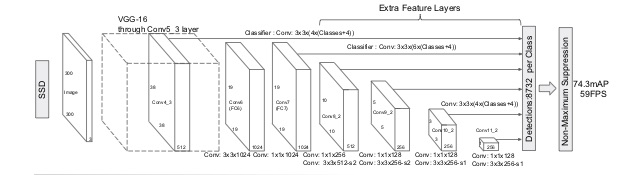
Single Shot MultiBox Detector(SSD)算法是一种快速的单次多类目标检测算法，它没有生成区域准则和特征重采样的过程，这也是SSD算法速度快于Fast-RCNN算法的根本原因。

图2 - 5 SSD算法的网络结构

网络的结构如图2 - 5所示，Fc7之前都是普通VGG16网络，用于提取图片特征。Fc7的输出将会依次进入不同的卷积预测器(小的卷积滤波器，Extra Feature Layers)，这些预测器将负责检测特定比例的特征，会依据自身在特征图上的位置做不同比例的缩放，可以涵盖各种输入目标的大小和形状。实现了将边界框的输出空间离散到一组默认的边界框中，所有Jaccard[[32]](#footnote-1)1系数高于0.5的默认边界框都会进行打分。每个框都会给出每一个类的结果，以及相对于默认框的原始形状的4个偏移量。最后大概会输出8732个结果，进行最大化抑制之后就是最终结果。

使用卷积预测器，使得即使输入了低清图像也可以获得高精度结果，并保持较快的运算速度。这些卷积预测器检测特定的特征，可以涵盖各种输入目标大小和形状，而且不断减小特征的尺寸提高计算了速度。而且为了使得算法更具鲁棒性，会做一些数据增强。将每张图片进行随机采样，如果原标记框的中心在采样图像内，就保留两者重叠的部分，并调整到300×300且以0.5的可能性进行水平翻转。

2.2.3 基于搜索的目标检测算法

2.2.3.1 AttentionNet

将对象检测问题看作是一种迭代分类问题。该算法着重于物体检测，而不是超更丰富的图像理解发展，使用的是深度神经网络的分类模型而不是回归模型。AttentionNet将对象的置信度和绑定框融合到了一个单一的网络中，整个算法也没有提出区域建议到过程。

该算法首先将图片调整到227×227，经过7个卷积层之后，到Conv8(见图2 - 6)层中会有两个分支，会得到5种结果：不包含任何对象、检测成功、以及三种剪裁方向。当结果是三种方向时就会按照得出的方向剪裁一个固定的长度，最后重新调整大小之后进入AttentionNet进行迭代。当最终检测成功，最后的输出图像就是检测出的目标，其执行的已知，只要将图片重新映射回原图就可以得到边界框。当一张图片中有多个对象实例是，可以分割图片得到多个候选区域，每个区域只包含一张图片，之后分别输入到AttentionNet进行预测。

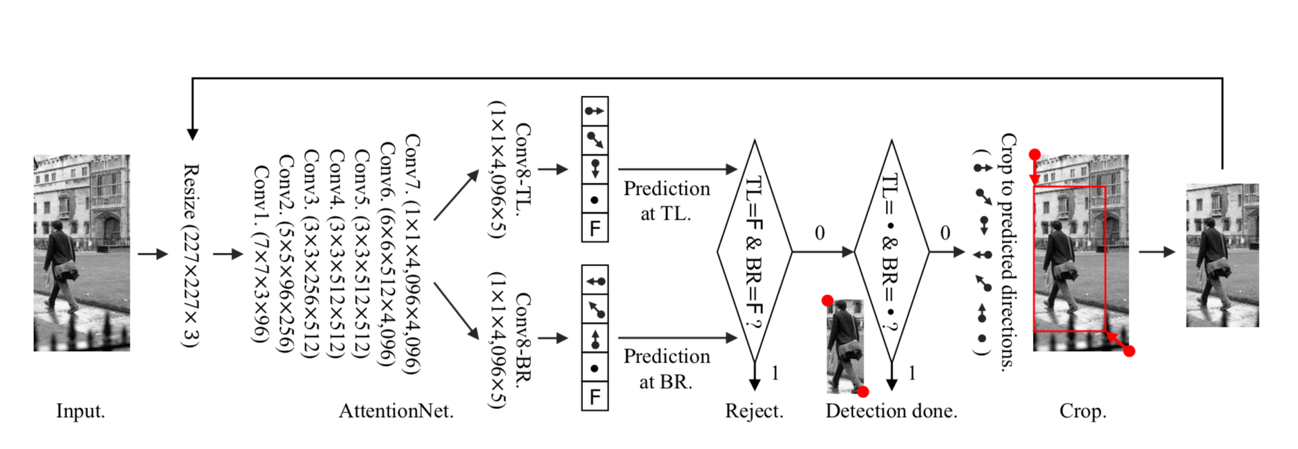


图2 - 6 AttentionNet算法的网络结构

AttentionNet通过聚合多个弱的分类来得到精确的边界框，图片剪裁的越小预测变得越强。整个网络成为一体，没有区域建议、对象分类和边界框回归三个模块。其他算法通常是使用SVM来计算得分，有一个缺点是不能保证分数最大的结果契合整个目标对象，因为使用区域建议大部分情况下会截断目标对象，因而需要增加模块来细致处理截断问题。

2.3　两类算法的分析

本章2.1节介绍的传统目标检测算法。大致流程基本相同，首先都是使用不同尺度的滑动窗口选定候选区域，之后从滑动窗口中提取中间特征并转化为作者设计的特征算子形式，最后使用AdaBoost[[[33]](#endnote-33)]或者是SVM等方法基于特征算子进行分类，得出目标的置信度绘制出检测轮廓。这里就有一个问题，即产生的滑动窗口数量巨大，该类算法的性能瓶颈就在于如何使用少的滑动窗口而不产生丢失的情况。而且特征算子是人工设计的，主要是为了针对光照变化和旋转等问题，对于其他特征的提取难免会有欠考虑，所以这些算法的提升都不高。而且，由于需要人工设计特征算子，所以特征算子的好坏完全取决于设计者的经验。只有在此领域研究多年的专家才能胜任此项任务，这也限制了传统目标检测算法的进一步发展。

本章2.2节介绍了多个基于深度卷积神经网络的算法，其中R-CNN系列的最出的结构也是遵循了传统方法的结构，但是由于深度卷积神经网络在提取特征的能力上更加强大而且全面，所以R-CNN算法一问世，就将远远优于传统算法。之后，本类算法努力的方向是将仅用卷积神经网络提取特征发展到把特征提取和分类、回归都结合到到一起，形成一个端到端的网络。R-CNN系列算法中区域建议也是限制其速度的主要部分，故也诞生了一部分算法抛弃了区域建议方式，转而直接从图像中生成候选的绑定框，所以该类算法速度相当快。

目前来看，基于深度卷积神经网络的算法在精度和检测能力上的优势相当明显。尤其是各种图像分类和检测的大赛上，排名靠前的算法中传统的目标检测算法几乎不见了踪影。在未来的多年中目标检测算法发展的主流将会是基于深度卷积神经网络的算法。

2.4　本章小节

本章主要是简要介绍了多个传统的且如雷贯耳的目标检测算法，如HOG算法  
还有SIFT算法，还介绍了近几年新崛起的基于深度神经网络的目标检测算法，如Faster-RCNN和AttentionNet。分析了这些个模型的基本组织结构，详细的阐述了每个算法的原理。在最后，总结了这两大类算法的基本流程，将这两大类算法各自的优缺点进行了分析。

第3章 数据增强技术

3.1　监督学习理论

在监督学习中，模型就是要学习的条件概率分布和决策函数，这些都包含在了模型的假设空间之中，如公式(4)所示：

(4)

其中，X与Y是定义在输入和输出空间上的变量。

利用深度神经网络可以模拟这个假设空间。在深度神经网络中，使用不同的参数组合就会得到中不同的映射，搜索一个能够真实反应这种与分布的映射就是训练神经网络的过程。

在监督学习中我们会定义损失函数：。通常我们得到的损失值越小，就会认为训练的这个网络的这次预测就越成功，此时对应的这组参数就越可能是我们最终要寻找的参数。由于和是在输入输出空间上随机变量，因此损失函数的期望就可以用来评价我们的训练成果。如果能得到最小的期望，由此得到的网络模型越能贴近真实数据的分布。

3.2　经验风险最小化和领域风险最小化

3.2.1 经验风险最小化和结构风险最小化

我们将网络训练时得到的平均损失称为经验风险(Empirical Risk)，如公式(5)所示：

(5)

根据大数定律，当训练集的大小接近无穷时，所以可以使用经验风险来逼近期望风险。Empirical Risk Minimization(ERM)准则认为经验风险最小的网络就是最优的模型。故按照此准则，最小化经验风险就是寻找最优网络的过程，如公式(6)所示：

当样本足够大时，使用经验风险可以得到很好的学习效果。因为在神经网络参数数量、损失函数的确定情况下，随着新的训练数据的不断加入训练最终会收敛到这个最优模型。

Structural Risk Minimization(SRM[[[34]](#endnote-34)])准则是为了防止神经网络的过拟合而提出的策略。当训练样本不足时，按照ERM未必可以产生优秀的泛化模型，反而会出现过拟合现象：仅在训练集上表现优秀，而在面对现实数据时效果不佳，一种解释是模型最终收敛到了局部最优而不是全局最优。SRM等价于Regularzation(正则化)。结构风险是在经验风险的基础上加上正则化项，如公式(7)所示：

这里面是惩罚项。是神经网络的复杂程度，由神经网络的参数的数量决定，这种复杂度表示了对于复杂模型的惩罚。为系数，用来平衡经验风险和模型复杂程度。

当正则化项和经验风险都最小时才能得到最优的模型，如公式(8)所示：

当加入正则化之后，过拟合现象将显著减少。现在，监督学习问题转化为求结构风险函数的最优化问题，即求一组最优参数的问题。

3.2.2 领域风险最小化

在实际中通常认为和服从分布，所以可以通过一个函数来改进输入数据。通过合成出的领域点作为训练的实际输入，即使用分布来模拟分布，使得训练数据可以更好的服从分布。这就是Vicinal Risk Minimization(VRM)准则。

影响ERM的关键因素是假设空间的容量。过大的假设空间会导致过拟合，而小的假设空间会导致欠拟合，使用ERM得到的结果很大程度上取决于在假设空间上对于的选择。但是VRM仍能在这两种情况下获得优秀的表现。如果对于的逼近的质量较差，只要空间具有适当小的容量，就不会出现欠拟合。这一点可由ERM保证。如果对于的近似是良好的，但是空间的容量没有选择恰当，甚至可以是无限的，VRM仍然可以有良好的表现。影响VRM的关键因素有两个，首先是模拟的质量，其次是假设空间的大小。使用VRM可以克服ERM选择假设空间不当而产生的缺点。究其原因是ERM默认原始输入数据能很好的服从分布。

VRM可以被归为是数据增强的一种手段。可以解决ERM无法处理的问题：

1. 缺少训练数据。训练数据中只有极少部分带有标签，大量的数据不带有标签。
2. 训练数据并不是遵循唯一分布，不同类的数据有不同分布，甚至其中有对抗性的样本。

3.3　Mixup(混合)技术与数据增强

3.3.1 大型深度神经网络的缺陷

深度神经网络的优势在于全面的提取特征。进年来深度神经网络在强大越发明显。但也伴随着一些不足之处，如记忆性和对于对抗性数据的敏感性，这就导致了大的泛化误差。

只要深度神经网络的参数个数超过训练点的个数就可以获得好的效果，深度神经网络就具有了足够的容量记忆整个数据集。[[[35]](#endnote-35)]使用随机标签替换真实标签训练，发现用于训练的几个标准网络也能很好的拟合跟换了标签的数据，而且仅在数次迭代之后便开始拟合。

深度神经网络对于输入数据在隐藏空间中的限制很小，在训练时会认为训练数据的分布与测试数据的分布是相同的，也就是两种在隐藏空间中都属于相同的区域。[[[36]](#endnote-36)]只要测试数据与训练数据服从同一分布，深度神经网络测试时表现出好的泛化性。反之，出现对抗性样本时，容易给出错误的结果，且置信度很高，因为两种数据在隐藏空间中属于不同的局部区域。

3.3.2 ERM准则的替代方案

在SRM准则中正则化是有效的排除不必要的方法，但在深度学习中却起了不同的作用。显式正则化可以改善泛化性，但既不必要也不能有效控制泛化误差。没有正则化也能的很好泛化。对显式和隐式正则化器的观察结果一致表明正则化器在适当调整后可以帮助改善泛化性能。然而，正规化器不太可能是泛化的根本原因，因为在所有正规化器移除后，网络继续表现良好。

ERM准则无法提供对于对抗性测试样本的泛化能力。即便是测试数据的分布与训练数据的分布只有轻微的不同之处。ERM准则也使得神经网络即使是在有正则化的情况下记忆训练数据，只要其参数数量超个训练数据数量，就具有了记忆所有训练数据的能力。

VRM准则可以是一种替代方案。VRM准则用函数将点替换为领域点，使用改进的密度估计进行训练。这就给原本在隐藏空间中位于两块不同局部区域的数据建立起了桥梁，将它们转换到同一区域。而且，训练的成功不仅取决于对于的近似，也可以由空间，即网络的参数数量决定。这种双重保证可以改善在实际训练中，输入腐蚀标签的或者是对抗性的数据所导致的泛化误差，提升网络的泛化性。

3.3.3 使用Mixup技术

VRM要求用一个优秀合成的分布来近似实际的分布。这时候就需要用到人类的知识来利用现有的训练数据合成出新的领域数据，以扩大对于训练分布的支持，通常被称为数据增强。一般的数据增强包括对图像进行翻转、缩放、旋转等。数据增强可以改善深度神经网络的泛化性，但是这些操作都默认增强的数据和原数据服从同一分布，在隐藏空间中属于相同的局部空间。

一般的数据增强操作不会使两个不同类间数据建立联系。为了更好的泛化性，势必要对隐藏空间中不同分布的数据进行约束，使得大多数数据最后都服从同一个分布，并且都是随机数据。Mixup技术基于这些考虑来构造领域样本，如公式(9)和公式(10)所示：

(9)

(10)

其中(是随机从训练集中选出的两个数据的特征向量，是由分布决定的一个混合系数。

这种线性插值的方法可以将两个分属不同类的数据联系起来，这样在预测训练样本之外的数据时的会减少不良震荡，取得更好的泛化性。生成系数以及最后做插值引入的额外计算负担很小。线性插值也符合奥卡姆剃刀定律：“若有多个假设与观察一致，则选最简单的那个”的要求。

在深度神经网络中使用Mixup非常简单，只需将某一层的输出插值之后作为下一层的输入即可。

3.4　本章小结

本章介绍的ERM准则、SRM准则、VRM准则和Mixup技术。介绍了为何在监督学习中使用ERM准则和SRM准则，以及它们的缺点。本章还介绍了VRM准则为何能克服ERM准则和SRM准则的不足，提高泛化性降低过拟合的原因。最后具体介绍了VRM准则的实际应用——Mixup技术。

第4章　将Mixup技术运用到Faster-RCNN算法

Faster-RCNN算法是本文所使用的目标检测算法。是R-CNN系列算法中第一个真正实现了端到端检测的目标检测算法，并且大幅提高了计算速度和精度。

4.1　Faster-RCNN

Faster-RCNN的结构[[[37]](#endnote-37)]如图4 - 1所示。较之Fast-RCNN主要改进在于使用RPN网络取代了Selective Search结构。Selective Search用于生产区域建议，但是为了能够和后续的检测网络共享特征计算，不得不放弃使用GPU加速。所以，此处成为提升Fast-RCNN计算速度的瓶颈，于是RPN网络诞生了。RPN网络可以和检测网络共享卷积特征，同时生成不同尺寸和不同比例的区域建议。

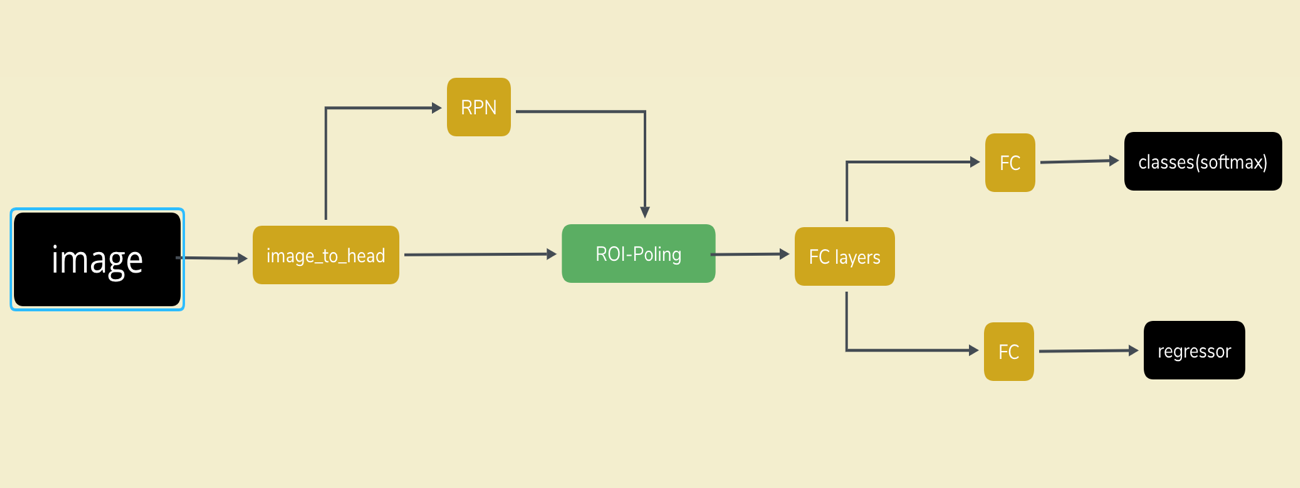
任意大小的图像作为输入，首先进入头部的Conv layers提取特征，之后特征被送入RPN网络生成区域建议。区域建议结合卷积特征得到每个区域的局部特征。之后，经过多个全联接层[[[38]](#endnote-38)]得到参数化的特征向量，用于预测分类和边界框。

图4 - 1 Faster-RCNN算法的流程图

4.1.1 Conv layers层

Conv layers层根据使用不同的模型有不同的结构，在VGG16和res101网络中都是7个网络层，都是多个卷积层、激活层、dropout层的组合。最终会生成整张输入图片的卷积特征图，作为RPN网络和POI-Pooling层的输入。

4.1.2 RPN网络

RPN网络是一个小的窗口网络，是一个深度全联接神经网络。RPN网络不停在特征图上滑动产生锚点框(anchor box)，每一个锚点框会作为一个参考，以产生一个特定比例和大小的区域建议，其工作原理如图4 - 2所示。RPN网络以特征图上的每一个点作为锚点，以锚点为中心3×3领域的局部窗口为输入，输出锚点框的坐标4元组(r,c,h,w)和两个得分，其中r和c是区域的左上角坐标，h和w是区域的高和宽，得分分别是前景和背景类的得分。每一个窗口会产生3种比例、3种尺寸共计9种锚点框，如图4 - 3所示。

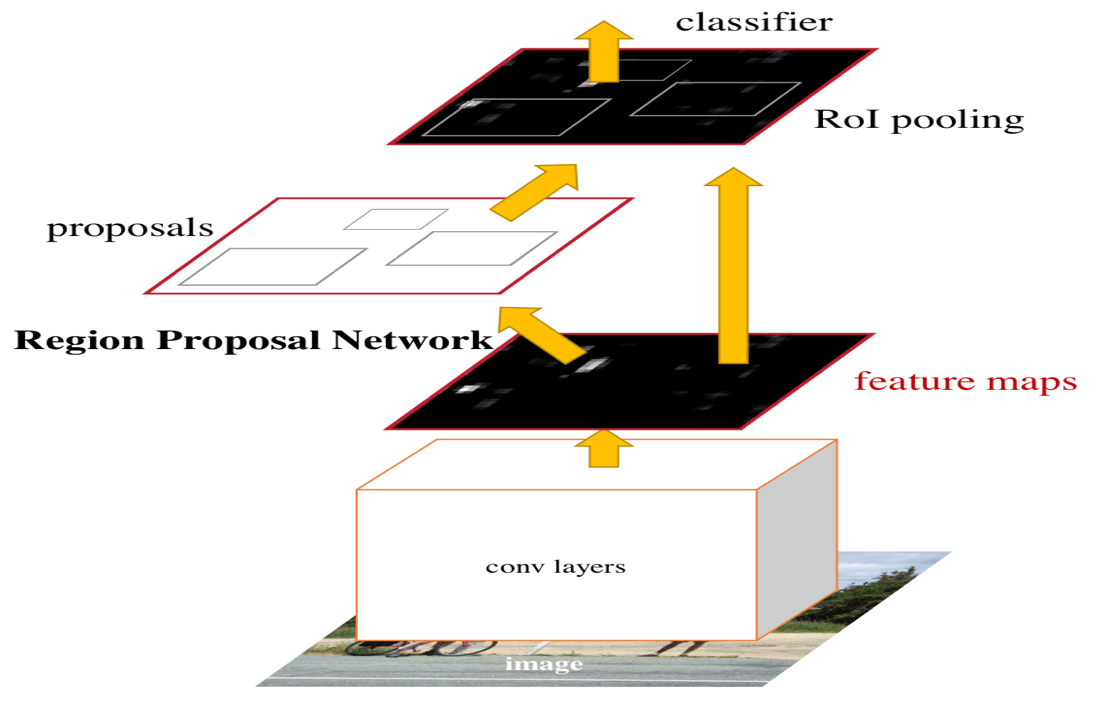


图4 - 2 RPN网络的工作图

基于锚点框的多尺度预测相比于传统的金字塔特征预测有更高的效率。传统金字塔特征会重新调整图像的大小，计算每一种比例的特征。然后使用不同大小 的滑动窗口在某一比例的特征图上提取特征。显然这种方式计算量很大，而且有许多不必要的重复计算。使用锚点框的方法只需要一张特征图和一个滑动窗口就可以实现得到不同比例和尺寸的区域建议的目的。特征图已由Conv layers层计算出，因此此方法增加的计算负担不是很大，只是引入了区域映射的计算量。

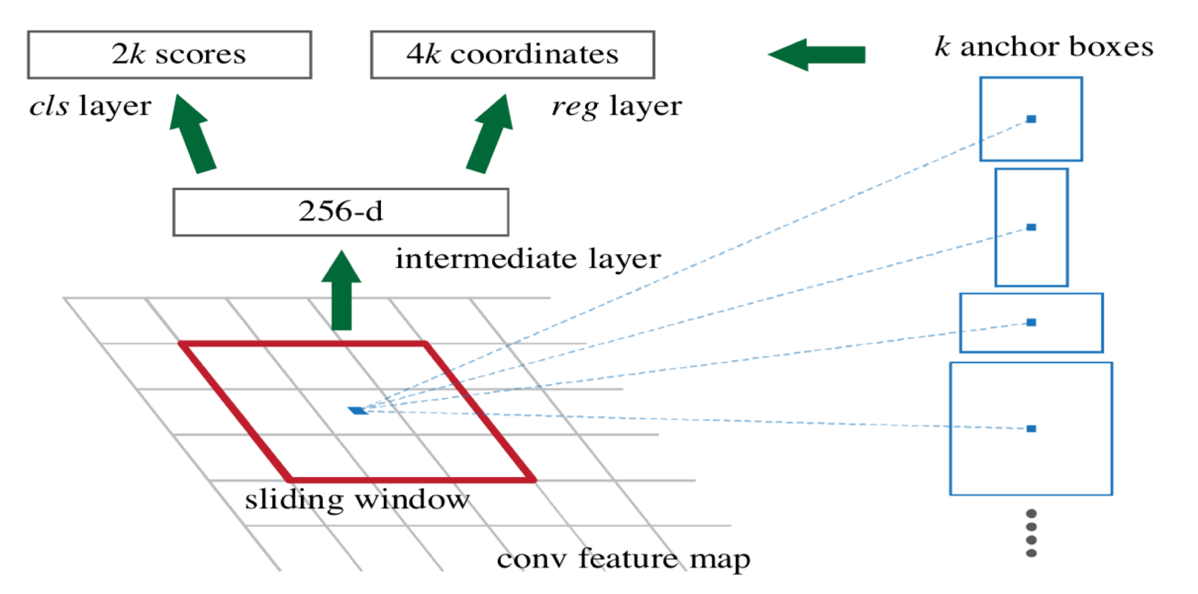


图4 - 3 anchor box的使用

这些区域建议可能会有很高的重叠率，会使用最大化抑制(NMS)保留和原边框IOU最大且得分最高的区域建议，最终大概会有12000个左右的区域建议被保留。之后，对超过图像边界的区域建议进行剪裁和修正，并放弃掉一些长度或宽度太大的区域建议使得剩下只有2000个左右的区域建议。在这2000个区域建议中IOU大于0.7的作为正样本，小于0.3的为负样本。随机选取256个区域建议，正负各128个作为最终提出的区域建议。

4.1.3 ROI-Pooling层

ROI-Pooling层的作用是根据得到的区域建议，将其对应的局部特征从已经计算好的卷积图中提取出来，并调整这些特征的形状到7×7，最终输入到之后的全联接层。如果有多个通道，每个通道独立进行池化。

ROI-Pooling层是简化版的金字塔池化网络，只有一层。使用max-pooling将特征降采样为7×7大小的固定输出。首先，将w×h大小的区域建议划分成7×7个小块。最后，提取最大值作为这一块的特征。最终的输出便是7×7局部最大特征。

4.1.4 Classification层

接收到最终的256个随机选择的区域建议的特征之后，首先通过几个全联接层将像素特征转换成特征向量的形式，这就生成了256个回归器。每个回归器虽然输入的是相同的特征的，但是在ROI-pooling层会得到各自独有的特征输出。它们各自的回归量不共享权重，因而可以预测不同尺寸和比例的窗口。而Fast-RCNN算法在对任意size的ROI进行池化提取操作时回归量是共享的。

池化层的输出会分别输入到两个不同的线性回归层中。其一是做Softmax分类预测，得到每一个ROI的分数。其二是得到每一个ROI的边界框修正参数，和RPN网络类似用于修正边界框。

4.1.5 基础网络

4.1.5.1 resnet101网络

表4 - 1 res网络的具体参数

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| layer name | output size | 18-layer | 34-layer | 50-layer | 101-layer | 152-layer |
| conv\_1 | 112×112 | 7×7,64，stride 2 | | | | |
| conv2\_x | 56×56 | 3×3 max pool，stride 2 | | | | |
|  |  |  |  |  |
| conv3\_x | 28×28 |  |  |  |  |  |
| conv4\_x | 14×14 |  |  |  |  |  |
| conv5\_x | 7×7 |  |  |  |  |  |
|  | 1×1 | Average pool ，1000-d fc，softmax | | | | |
| 参数 |  |  |  |  |  |  |

Faster-RCNN使用的残差网络是res101网络，ResNet [[[39]](#endnote-39)]的结构见表4 - 1。conv1是一个简单的初始化结构，由Conv2d层、BatchNorm2d层、ReLU层和MaxPool2d层各一个组成。

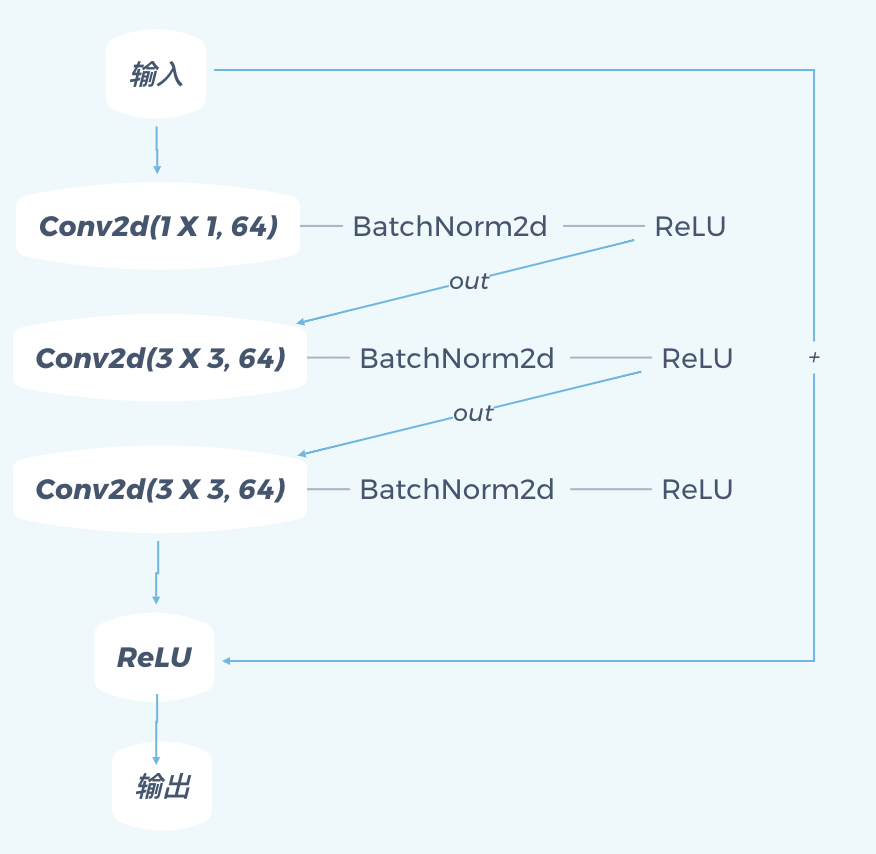


图4 - 1 一个bottleneck结构的示意图

在res101网络中除了conv\_1以外其余的卷积块都是由一种称为bottleneck的结构块组成，其结构如图4 - 1所示。每一个bottleneck块都是由三个Conv2d层、三个BatchNorm2d层和一个ReLU层构成主体部分，最后将卷积结果和输入相加后可以得到最终输出。conv2\_x中有3个bottleneck，conv3\_x有8个bottleneck，conv4\_x有23个bottleneck。res101网络共有101个卷积层。

conv4\_x之前的网络作用是得到整张图片的特征图。conv4\_x就是上文提到的全联接层，用以训练256个回归器。conv5\_x及之后都没有被使用。

4.1.5.1 VGG16网络

表4 - 1 VGG网络的具体参数

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| VGGnet | | | | | | |
| layer name | A | A-LRN | B | C | D | E |
|  | 11 layers | 11 layers | 13 layers | 16 layers | 16 layers | 19 layers |
|  | 输入 （的RGB图像） | | | | | |
| feature layer | Conv3-64 | Conv3-64  LRN | Conv3-64  Conv3-64 | Conv3-64  Conv3-64 | Conv3-64  Conv3-64 | Conv3-64  Conv3-64 |
|  | maxpool | | | | | |
| feature layer | Conv3-128 | Conv3-128 | Conv3-128  Conv3-128 | Conv3-128  Conv3-128 | Conv3-128  Conv3-128 | Conv3-128  Conv3-128 |
|  | maxpool | | | | | |
| feature layer | Conv3-256  Conv3-256 | Conv3-256  Conv3-256 | Conv3-256  Conv3-256 | Conv3-256  Conv3-256  Conv1-256 | Conv3-256  Conv3-256  Conv3-256 | Conv3-256  Conv3-256  Conv3-256  Conv3-256 |
|  | Maxpool | | | | | |
| feature layer | Conv3-512  Conv3-512 | Conv3-512  Conv3-512 | Conv3-512  Conv3-512 | Conv3-512  Conv3-512  Conv1-512 | Conv3-512  Conv3-512  Conv3-512 | Conv3-512  Conv3-512  Conv3-512  Conv3-512 |
|  | Maxpool | | | | | |
| feature layer | Conv3-512  Conv3-512 | Conv3-512  Conv3-512 | Conv3-512  Conv3-512 | Conv3-512  Conv3-512  Conv1-512 | Conv3-512  Conv3-512  Conv3-512 | Conv3-512  Conv3-512  Conv3-512  Conv3-512 |
|  | Maxpool | | | | | |
| Classifier[0] | FC-4096 | | | | | |
| Classifier[3] | FC-4096 | | | | | |
| Classifier[6] | FC-1000 | | | | | |
|  | Soft-max | | | | | |

VGGNet[[[40]](#endnote-40)]几乎没有改动，其结构如表4 - 2所示。所有feature layer都被用作计算图片的特征图。

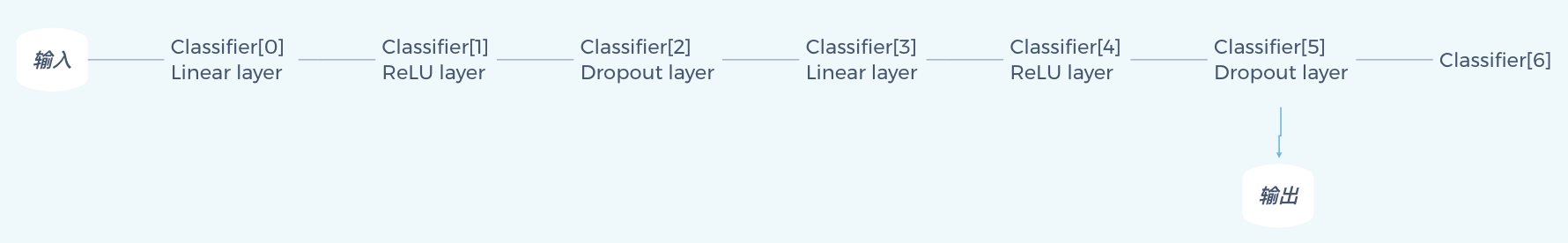
在特征计算网络之后3个Linear层被加入到一个名为Classifier的网络结构作为上文提到的全联接层。VGG16网络的Classifier结构由3块结构共7层网络构成，分别对应3个全联接层、2个Dropout层和2个ReLU层，具体结构见图4 - 4。Classifier[6]没有被使用。

图4 - 4 Classifier结构的示意图

4.2　运用Mixup技术到Faster-RCNN

在Faster-RCNN中有多处地方可以使用混合[[41]](#footnote-2)1，主要是读取图片时、计算损失时以及在网络内部实现混合。根据使用不同的基础网络，内部混合略有不同。在网络中有多处地方可以混合，为了节省计算时间，只选择一处进行混合。所以总共有3处混合。

4.2.1 混合输入图片

首先是在输入时混合。一次读取两张图片，为了使两张图片可以完美混合，需要将第二张图片调整大小使其和第一张图片一样大。同时还要按照比例修改第二张图片的原标记框等其他信息，后续的混合需要用到。混合后的图片其类别任然属于第一张图片，标签使用第一张图片的标签。将图片及其属性封装后输出。以后称第一张图片为主要图片。

这一阶段的主要操作如表4 - 4：

表4 - 4图片的混合

|  |
| --- |
| 算法1 ：输入图片的混合 |
| 1: 输入：im1，im2，  2: shape1，shape2 = im1.size，im2.size  3: ratio = shape1 / shape2  4: 根据ratio重新调整im2的大小  5:  6: 使用blob封装im，im2  7: 设置im，im2对应的属性  8: 输出：blob |

4.2.2在网络中混合

4.2.2.1 RPN网络

首先是使用主要图片根据特征图产生所以肯能的锚点框，根据得分和IOU使用最大化抑制之后，得到2000个左右分数最高重叠率较小的区域。由于RPN网络是单独训练的，故要预测锚点框的得分和边界框，要分别计算两张图片的loss，进行混合之后用于RPN网络的反向传播。

这里对于每张图片都有4个参数：rpn\_labels、rpn\_bbox\_targets、rpn\_bbox\_inside\_weights、rpn\_bbox\_outside\_weights，分别对应各自图片的前景背景得分、回归修正、输入权重、输出权重。

而挑选出来的那256个区域也会产生对应的4个参数：labels、bbox\_targets、bbox\_inside\_weights、bbox\_outside\_weight，这4个参数会用于整个检测网络的反向传播。

这些loss都要进行混合。

4.2.2.2池化层前的混合

在池化层前将256个区域ROI进行一次混合。此处是256个ROI自己内部混合。首先将这256个ROI随机洗牌，记下洗牌后的序号，见表4 - 4。然后将原先按顺序的ROI和打乱顺序的ROI进行混合，见表4 - 5。

表4 - 4 生成rcnn\_mix\_index

|  |
| --- |
| 算法2 ：生成混合序号 |
| 1: 输入：batch\_size  2: rcnn\_mix\_index = np.arrange(batch\_size)  3: np.random.shuffle(rcnn\_mix\_index)  4: 输出：rcnn\_mix\_index |

表4 - 5 pool5处的混合示例

|  |
| --- |
| 算法3 ：池化层前的混合 |
| 1: 输入：pool5，rcnn\_mix\_index，  2:  3: 输出：pool5 |

得到的rcnn\_mix\_index是一个全局量，用于之后所有的混合。Batch\_szie则是训练参数，由于受到硬件限制本文中统一是256。

4.2.2.3 res101网络中的混合

当以res101为基础网络时，在混合时有多处地方可以进行，如表4 - 6：

表4 - 6 res101网络中的混合示例

|  |
| --- |
| 算法4 ：res101网络中的混合 |
| 1: 输入：，rcnn\_mix\_index，  2:  3: 输出： |

首先，可以在最后一层全联接层conv4\_x之后进行混合。其次，可以在conv4\_x中进行混合，由于每一个bottleneck都可以分成3层结构，所以在每一块之后都可以进行一次混合。同一时间，一个bottleneck只混合一次。因为conv4\_x有23个bottleneck结构，所以有23处地方同时进行混合。

4.2.2.4 VGG16网络中的混合

当以VGG16网络为基础网络时，在混合时也有多处地方可以进行，主要集中在classifier结构内部，具体见表4 - 7：

表4 - 7 VGG16网络混合示例

|  |
| --- |
| 算法5 ：VGG16网络中的混合 |
| 1: 输入：classifier[n]，rcnn\_mix\_index，  2:  3: 输出：classifier[n] |

首先，可以在第一块之后进行混合，即classifier[2]之后。其次，可以在classifier[5]之后进行混合，即将整个网络的输出进行混合。

4.2.3 Mixup Loss

在Faster-RCNN中边界框的损失都是使用的smooth\_l1 loss。公式(11)中当X小于1时，损失与输入的平方成比例，当大于等于1时都是输入的绝对值加上-0.5的偏置。分类的损失都是softmax loss，见公式(12)。总的损失如公式(13)所示是两者之和，是权重系数以平衡两个损失。

(13)

这里有多处损失需要使用，分别计算两张图片对应的损失，然后进行混合：

1. rpn\_cross\_entropy。此损失是计算RPN网络得出的分数和图片中对应绑定框里检测对象的标签的交叉熵。专门用于训练RPN网络。
2. rpn\_bbox\_box。这个损失是计算RPN网络预测的绑定框和在特征图上映射出的实际的绑定框的smoothL1 loss。专门用于训练RPN网络。
3. loss\_box。这个损失是计算最后检测网络输出的绑定框的预测值和实际的原标记框间的smoothL1 loss。
4. cross\_entropy。此损失是计算最后检测网络输出的置信度和图片中绑定框所检测对象的标签的交叉熵。

这4处的损失都有统一的形式，如公式(13) 所示：

在原始的Faster-RCNN中最后的总损失total\_loss = rpn\_cross\_entropy + rpn\_bbox\_box + loss\_box + cross\_entropy，实验表明加入mixup之后选择total\_loss = loss\_box + cross\_entropy，即保持RPN网络与预训练模型一致，只训练加入混合后的检测网络。其结果最佳。为保持RPN网络的权重不变，将RPN对应的两个损失值设为固定值-1。这样由于使用的是ReLU激活函数，保证了次处的损失不会反向传播。

4.3　本章小结

本章首先从4大模块入手详细解释了Faster-RCNN算法的原理和网络结构，介绍了其对本系列算法的改进之处。主要优化部分在于使用RPN网络代替了Select Search，这种设计不仅让负责提取ROI共享了计算，而且使得整个网络成为一个端到端的结构。其次，说明了本文对于VGG网络和res网络做了哪些修改，将这两个网络的前部分用于提取特征，最后一层用作Faster-RCNN的检测器。最后，讲解了如何在Faster-RCNN算法中实际运用混合以提高其泛化性。

第5章　实验

5.1　PASCAL VOC数据集

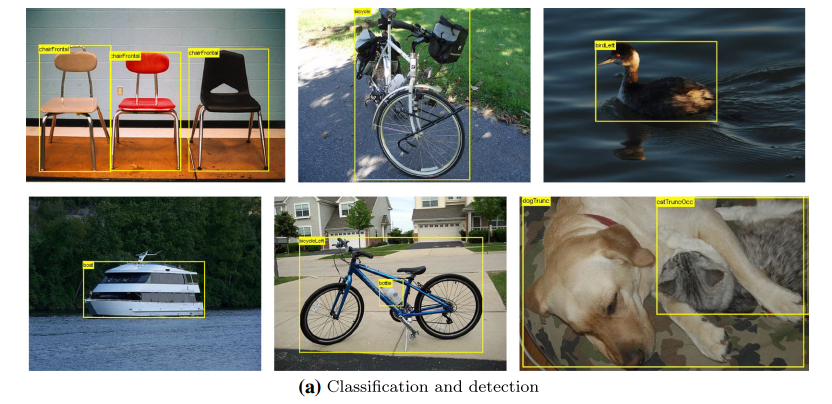
5.1.1 PASCAL VOC挑战赛

图5 - 1 PASCAL VOC数据集的检测和分类示例

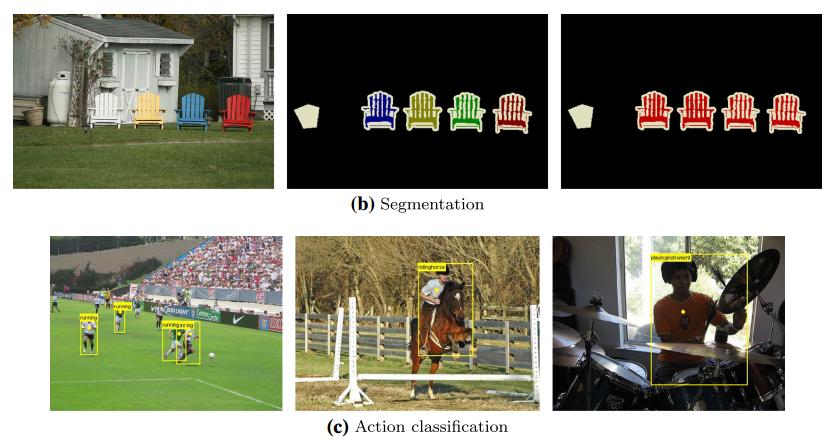
PASCAL VOC[[[42]](#endnote-41)](Pattern Analysis, Statical Modeling and Computational Learning Visual Object Classes)是一个由欧盟资助的网络组织举办的世界级计算机视觉挑战赛。该组织为视觉和机器学习社区提供标准的图像和注释数据集，以及标准评估程序。

图5 - 2 PASCAL VOC数据集的分割和动作分类示例

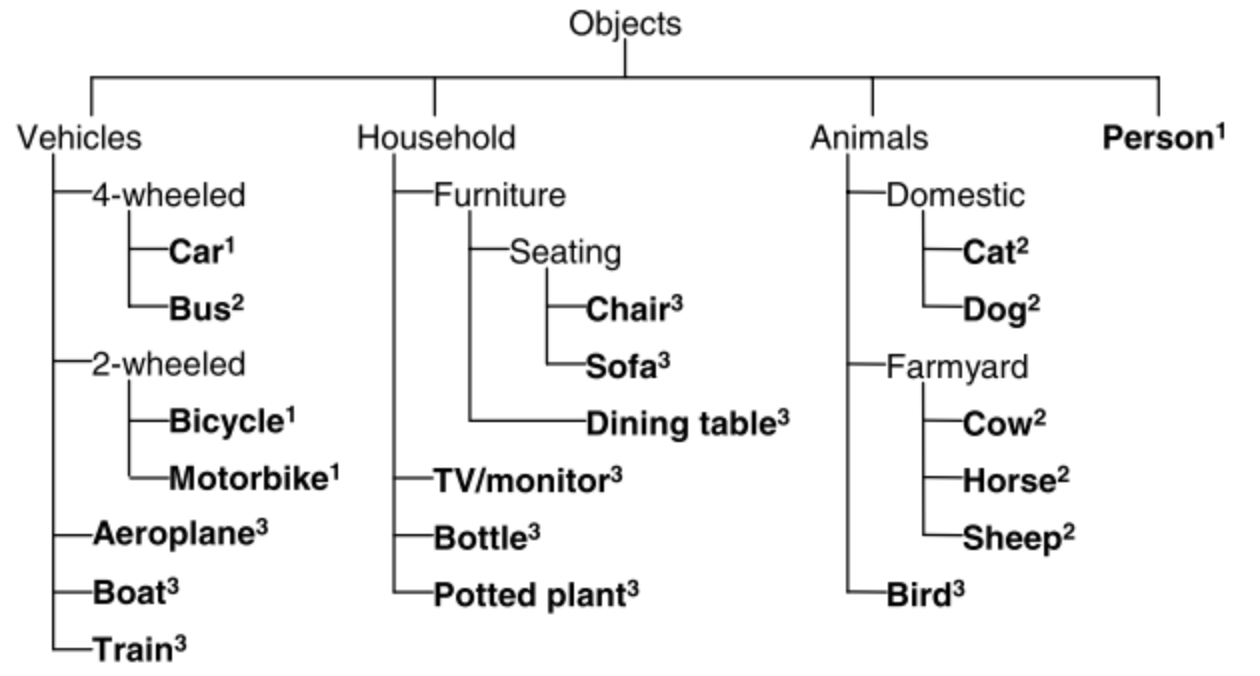
该项挑战赛从2005年开始到2012结束，一共举办了8届。期间涌现出来一大批优秀的算法，其中大多数是利用了深度卷积神经网络，极大的促进了深度学习的发展。

大赛的主要内容是分类和检测。如图5 - 1分类就是判断图像中的实例是否属于数据集中的某一个类别。检测就是确定实例在图像中的具体位置。其他任务如分割和动作识别也有涉及。分割即细致的判断每一个像素所属的类别，如图5-2。动作识别就是判断人处于某种行为如图5 - 2。图5 - 4是人体部位识别的示例。

尽管比赛已经停办，但仍然可以在PASCAL VOC的网站[[43]](#footnote-3)上提交结果与其他模型进行比较。目前在PASCAL VOC的主页上传统的视觉算法，无论是检测还是分类都几乎销声匿迹了，只剩下基于深度卷积神经网络的模型了。

5.1.2 PASCAL VOC数据集的介绍

PASCAL VOC数据集[[[44]](#endnote-42)]总共有4个大类，车辆、家具、动物以及人。目前这4个大类又分成了20个小类。每一张图片中的标签都对应于这20个小类中的一种。如图5 - 3，加粗的部分即为实际的标签。2005年时，该数据集还不太完整，只有自行车、汽车、摩托车和人。而且此时的数据集容量也非常小，测试集、训练集加上验证集之后的图片总数也只有1578张，总共包含2209个对象。到2007年时，PASCAL VOC数据集初步完善，有了如图5 - 3的结构。2009年，数据集开始不断扩充，在原来的基础上又加入了新的图片。但是由于数据集本身的制作方式，每年的数据集实际内容是不同的，没有重叠的图片。2012年之后数据集由于比赛不在举办数据集也停止了扩充。

图5 - 3 PASCAL VOC数据集的结构

虽然PASCAL VOC挑战赛已经停止举办，但由于该数据集2005年就已经出现，是一个较早的数据集，所有仍然有许多分割或者检测的文章会使用PASCAL VOC数据集。

本文使用的数据集事PASCAL VOC2007和PASCAL VOC2012数据集，两者是互斥的，PASCAL VOC2012数据集包含了从2008年到2012间的所有数据。目前PASCAL VOC2012对测试集仍没有公开。所以不管使用了哪个数据集进行训练，如果是本地测试最终都使用PASCAL VOC2007都测试集进行测试。

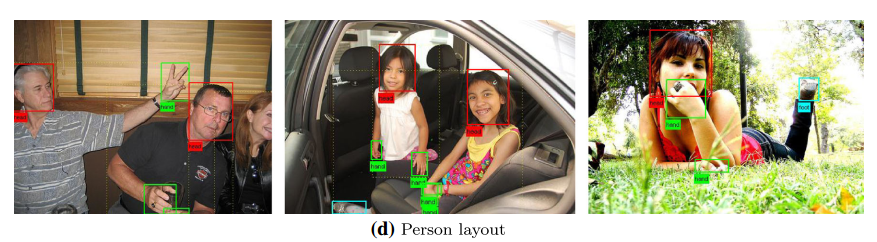


图5 - 4 PASCAL VOC数据集的人体部位识别示例

5.1.3 PASCAL VOC数据集的制作

PASCAL VOC数据及有一套独特的生成方法使得PASCAL VOC2007和PASCAL VOC2012互不相容。所有图片都是收集自一个名为flickr[[45]](#footnote-4)1的图片分享网站。

PASCAL VOC数据集的一个优秀特性就是无偏差性。为了确保无偏差性所选择的图片都是未被学者选择和研究的个人图片，而且在拍摄时没有特定的目的。例如，某张图片中会含有电视机的实例，但是电视机不一定会是图像的焦点，场景中仅是含有电视机。PASCAL VOC数据集也有很大的宽泛性。图像中的实例对象会有多种姿态和照明条件。在其他的数据集中人就基本等同于行人，在PASCAL VOC数据集中人会有更多的姿态，可以从事各种活动，而不单单只是行走。

在获取图片时，没制作一个类别都会从flickr中检索500,000张图片。为了保证无偏差性，检索时只根据类别检索而无其他额外条件。在这些图片集合中随机选择足够多的图片作为数据集的实际内容。在随机选择图片的过程中，会有一套算法来保证图像不是重复的或者是近似的，减少数据集的冗余。

PASCAL VOC的标注信息非常丰富。包括了source(图片的来源等信息)、size(图片的大小)、name(类别名称20个类中的某一个)、truncated(物体是否截断)、segmented(是否分割)、bounding box(边界框) 、viewpoint(拍摄角度)。截断的对象只有部分出现在图片里，但也会被用于训练。边界框包括左上角和右下角的4个坐标。在同一个场景中，可能会从不同的视角进行观察，得到不同的图像，这些图片会被认为是不同的。如果某一个实例对象的大部分被遮挡了，则这个对象会被标注为遮挡。

5.1.4 PASCAL VOC数据集的使用

虽然近年来MS COCO数据集大有取代PASCAL VOC数据集之势。但本文仍然采用PASCAL VOC数据集进行训练和测试，主要原因在于其无偏差性和被广泛使用性。

目前PASCAL VOC数据集只公开了PASCAL VOC2007的测试集、验证集和训练集，而PASCAL VOC2012的测试集没有公开。所有通常对使用方法有如下几种：

1. 使用PASCAL VOC2007的训练集和验证集进行训练，使用PASCAL VOC2007的测试集进行测试。
2. 使用PASCAL VOC2012的训练集和验证集进行训练，使用PASCAL VOC2012的测试集进行测试。此结果需要上传到PASCAL VOC的官方服务器[[46]](#footnote-5)1上进行评价。
3. 使用PASCAL VOC2012的训练集和验证集进行训练，使用PASCAL VOC2007的测试集进行测试。这种方式，我们可以自行测试结果。
4. 为了获得最大的训练数据，还可以把PASCAL VOC2007的测试集、训练集和验证集，以及PASCAL VOC2012的训练集和验证集都用于训练，最后使用PASCAL VOC2012的测试集进行测试。

本文使用了两种训练方法，即使用方法1和方法3。

5.2　参数设置

5.2.1 的生成

混合系数是根据分布决定的，其超参数控制的生成，当时表示ERM准则。其生成很简单，在python的科学计算库numpy中有其API，见公式(14)：

为了方便分析实验结果，在对比实验中使用了控制变量方法，所有改进模型的都是统一的。在所有模型训练之前已经产生。

5.2.2 原始的Faster-RCNN模型的主要参数

表5 - 1 原始Faster-RCNN算法的部分训练参数

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名称 | 参数值 | 参数名称 | 参数值 |
| IMS\_PER\_BATCH | 1 | WEIGHT\_DECAY | 0.001 |
| BATCH\_SIZE | 256 | FG\_THRESH | 0.5 |
| LEARNING\_RATE | 0.001 | BG\_THRESH\_HI | 0.5 |
| GAMMA | 0.1 | BG\_THRESH\_LO | 0.1 |
| RPN\_POSITIVE\_OVERLAP | 0.7 | POOLING\_SIZE | 7 |
| RPN\_NEGATIVE\_OVERLAP | 0.3 | ANCHOR\_RATIOS | [0.5,1,2] |
| RPN\_NMS\_THRESH | 0.7 | ANCHOR\_SCALES | [8,16,32] |

Faster-RCNN的基本参数和原作者使用的参数基本保持一致，具体见表5 - 1。IMS\_PER\_BATCH代表了每一个批次的读取图片的数量，即输入数据的数量，这里一次只读取一张图片。BATCH\_SIZE表示了每一次迭代会有多少个从RPN网络中产生的区域建议被用来标注正负标签，之后会被用于后续的检测网络的训练，可以是512也可以更大，本文选取其大小为256是受到显存大小的限制。LEARNING\_RATE和GAMMA表示学习率和学习率的减少因子，当迭代超过300,00次后，LEARNING\_RATE = LEARNING\_RATE \* GAMMA，原作者没有给出减小等步数，本文选取了300，00次。RPN\_POSITIVE\_OVERLAP以及RPN\_NEGATIVE\_OVERLAP是用于区分正负样本的阈值，分别表示在RPN网络里当IOU大于RPN\_POSITIVE\_OVERLAP时这个区域建议就是正样本，小于RPN\_NEGATIVE\_OVERLAP时则是负样本。RPN\_NMS\_THRESH是对于RPN网络提出的区域建议，使用最大化抑制时的阈值。WEIGHT\_DECAY是正则化系数。FG\_THRESH是用于判断实例是否为前景对象的系数，当一个区域建议与原边界框的的IOU大于FG\_THRESH时，此实例就会被认为是前景物体。BG\_THRESH\_HI以及 BG\_THRESH\_LO是用于判断实例是否为背景的系数，这里有两个系数，表示IOU在[LO, HI]取决内的区域建议才是背景，低于LO的不做研究。POOLING\_SIZE表示ROI-pooling层的输出大小，即最终用于检测网络的特征矩阵的大小。ANCHOR\_RATIOS还有ANCHOR\_SCALES是用于产生区域建议的参数，根据这些参数将固定的输入窗口变为9个各种尺寸和比例的锚点框。

5.2.3 改进模型的主要参数

表5 - 2 加入Mixup后的部分训练参数

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名称 | 参数值 | 参数名称 | 参数值 |
| IMS\_PER\_BATCH | 2 | WEIGHT\_DECAY | 0.001 |
| BATCH\_SIZE | 256 | FG\_THRESH | 0.5 |
| LEARNING\_RATE | 0.001 | BG\_THRESH\_HI | 0.5 |
| GAMMA | 0.1 | BG\_THRESH\_LO | 0.1 |
| RPN\_POSITIVE\_OVERLAP | 0.7 | POOLING\_SIZE | 7 |
| RPN\_NEGATIVE\_OVERLAP | 0.3 | ANCHOR\_RATIOS | [0.5,1,2] |
| RPN\_NMS\_THRESH | 0.7 | ANCHOR\_SCALES | [8,16,32] |
| MIX\_LOCATION | TBD | RCNN\_MIX\_INDEX | TBD |
| LOSS\_STRATGY | TBD |  |  |

基于控制变量的原则，改进模型大部分参数与原始模型保持一致。产生三个新的参数，都是用于辅助控制混合的进行，见表5 - 2。

MIX\_LOCATION用于控制在全联接层中不同的位置进行混合，全联接层里有多处地方可以进行混合，本文所做的对比实验之一就是比较在不同的位置进行混合的效果。

RCNN\_MIX\_INDEX是在检测网络中进行混合的序号，每一个训练批次都是统一的，随机生成，不同批次之间没有关联。

LOSS\_STRATGY是控制反向传播的参数，有三种训练策略：只训练RPN网络、只训练检测网络、同时训练RPN网络和检测网络。

5.3　模型训练

5.3.1 Faster-RCNN原始模型

本文的基本模型是基于[[[47]](#endnote-43)]的代码，采用其提供的初始化模型(vgg[[48]](#footnote-6)1,res[[49]](#footnote-7)2)之后训练出自己的原始模型。本文在之后训练加入混合技术的模型时，会把这些原始模型作为训练的初始化模型。

加入混合后的新模型会有三种不同的训练策略。主要原因是分析它们的训练时间。在比较三种不同训练策略后，寻找出一个既能保证检测精度，又能在训练和检测速度上接近原始Faste-RCNN模型的新模型。

5.3.2 RCNN\_ONLY策略

此训练策略只训练检测网络。RPN网络的参数将保持和原始模型相同。为保持RPN网络的权重不变，将RPN网络对应的两个损失值设为固定值- 1。这样由于使用的是ReLU激活函数，其特性保证了此处的损失不会被反向传播。在训练时会将之前已经训练好的原始Faster-RCNN模型作为初始化模型，其一是为了保持新模型的RPN网络能够和原始的Faster-RCNN模型中的RPN网络一致，其二是出于控制变量的原则，尽量减少与原始Faster-RCNN模型的不一致。具体见公式(15):

5.3.3 RPN\_ONLY策略

此训练策略只训练RPN网络。检测网络的参数将保持和原始模型相同。为保持检测网络的权重不变，将检测网络的两个损失值设为固定值- 1。这样由于使用的是ReLU激活函数，其特性保证了此处的损失不会被反向传播。在训练时会将之前已经训练好的原始Faster-RCNN模型作为初始化模型，其一是为了保持新模型的检测网络能够和原始的Faster-RCNN模型中的检测网络一致，其二是出于控制变量的原则，尽量减少与原始Faster-RCNN模型的不一致。如公式(16):

5.3.4 RCNN+RPN策略

此训练策略将同时训练检测网络和RPN网络。在训练时也会将之前已经训练好的原始Faster-RCNN模型作为初始化模型。是以原始Faster-RCNN模型为基础的再训练。这样做可以使得模型可以更快的收敛，最终找到最优模型。

5.4　实验结果

5.4.1 实验平台

表5 - 3 实验平台

|  |  |
| --- | --- |
| 名称 | 型号 |
| CPU | Intel(R) Core(TM) i3-3220 CPU @ 3.30GHz×4 |
| GPU | NVIDIA GeForce GTX 1060GB/PCIe/SSE2,计算能力6.1 |
| 内存 | Kingston DDR3 1333Mhz,大小8192MB |
| 操作系统 | Ubuntu 16.04LTS |
| 代码环境 | CUDA Version V9.0.176  Anaconda Version 4.5.12  Python Version 3.6.7  Pytorch Version 0.4.0 |

Pytorch是由基于Lua语言的深度学习框架Torch移植而来的。由于python语言被广泛用于机器学习和深度学习的开发，Torch的开发团队将其移植到了python平台上，即如今的Pytorch。Pytorch对于Torch的很多模块进行了重构，加入了自动反向传播模块。Pytorch相比于TensorFlow有很多优势[[[50]](#endnote-44)]：

1. 简洁。Pytorch相比于另一个被广泛使用的框架TensorFlow更加简单。没有session、tensor等复杂的概念，只有张量、变量和网络3个层次，可以同时进行修改和操作。其源码也十分易读。
2. 快速。Pytorch在许多测试中表现优于TensorFlow。
3. 易用。TensorFlow的接口经常会随着版本的更新而改变，这就导致就的代码无法和新的版本兼容。Pytorch的接口设计是出于易用性考虑的，符合人们的思维习惯。
4. 开发社区活跃。Pytorch的使用文档完整。开发者经常活跃于社区中和用户进行交流，而且有FaceBook作为幕后支持，确保了Pytorch能够稳定持续的被跟新和维护。

现如今，Pytorch支持两个python版本：python2.7和python3.6。

复现的代码是基于pthon3.6和pytorch0.4.0。具体参数见表5 - 3。所有的实验均是在同一台机器及其以及同样的配置环境下运行，以确保实验的公平性。采取控制变量法，大多数参数均是相同的。

5.4.2 原始模型的结果

表5 - 4 Baseline

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 模型+数据集 | 平均训练时间(s) | 平均检测时间(s) | 平均准确度(%) | 最好准确度(%) | 最坏准确度(%) |
| VGG16+07 | 0.431 | 0.138 | 71.02 |  |  |
| VGG16+0712 | 0.386 | 0.127 | 75.54 |  |  |
| res101+07 | 0.763 | 0.239 | 75.90 |  |  |
| res101+0712 | 0.706 | 0.228 | 80.11 |  |  |

见表5 – 4。本文采用了训练

5.4.3 RCNN\_ONLY策略

表5 - 5 只训练检测网络的结果

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 模型+数据集+LOCATION | 平均训练时间(s) | 平均检测时间(s) | 平均准确度(%) | 最好准确度(%) | 的准确度(%) |
| VGG16+07, 1 |  |  |  |  |  |
| VGG16+07, 2 |  |  |  |  |  |
| VGG16+0712, 1 |  |  |  |  |  |
| VGG16+0712, 2 |  |  |  |  |  |
| res101+07, 1 | 0.751 | 0.240 | 73.76 |  |  |
| res101+07, 2 |  |  |  |  |  |
| res101+07, 3 |  |  |  |  |  |
| res101+0712, 1 |  |  |  |  |  |
| res101+0712, 2 |  |  |  |  |  |
| res101+0712, 3 |  |  |  |  |  |

5.4.4 RPN\_ONLY策略

表5 - 6 只RPN网络的结果

|  |  |  |  |
| --- | --- | --- | --- |
| 模型+数据集+LOCATION | 平均训练时间(s) | 平均检测时间(s) | 平均准确度(%) |
| VGG16+07, 1 | 0.290 | 0.127 | 64.40 |
| VGG16+07, 2 | 0.296 | 0.125 | 64.70 |
| VGG16+0712, 1 | 0.303 | 0.124 | 69.52 |
| VGG16+0712, 2 |  |  |  |
| res101+07, 1 | 0.275 | 0.206 | 70.64 |
| res101+07, 2 | 0.286 | 0.190 | 69.81 |
| res101+07, 3 |  |  |  |
| res101+0712, 1 | 0.289 | 0.187 | 74.53 |
| res101+0712, 2 |  |  |  |
| res101+0712, 3 |  |  |  |

5.4.5 RCNN+RPN策略

表5 - 7 两种网络都训练的结果

|  |  |  |  |
| --- | --- | --- | --- |
| 模型+数据集+LOCATION | 平均训练时间(s) | 平均检测时间(s) | 平均准确度(%) |
| VGG16+07, 1 | 0.439 | 0.126 | 70.22 |
| VGG16+07, 2 | 0.439 | 0.126 | 70.19 |
| VGG16+0712, 1 | 0.453 | 0.123 | 76.94 |
| VGG16+0712, 2 |  |  |  |
| res101+07, 1 | 0.765 | 0.188 | 72.98 |
| res101+07, 2 | 0.771 | 0.184 | 72.69 |
| res101+07, 3 |  |  |  |
| res101+0712, 1 | 0.819 | 0.194 | 79.47 |
| res101+0712, 2 |  |  |  |
| res101+0712, 3 |  |  |  |

5.5　实验结果的分析

5.6　本章小节

本章在开头介绍了PASCAL VOC挑战赛及其所公开的用于目标检测、分类的数据集，详细说明了PASCAL VOC数据集的优秀之处。之后，给出了本文在实验时所使用的参数，大部分参数未曾修改，与原始Faster-RCNN模型所使用的参数一致。详细说明了主要参数的作用和使用方式。

本章提出了三种不同的训练策略，结合不同的训练位置，会有多种新模型被训练出来。使用三种策略是为了寻找一个训练时间和训练精度的平衡点。

在本章的最后，分别根据三种不同的训练策略的实验结果进行分析。得出最终的实验结论。

第6章　改进模型的实际应用

从目前目标检测的实际运用情况来看，大多数模型的数据来源都是一些低轻清摄像头。而这些摄像头普遍分布在公路边、入口处以及一些需要安全监控的区域。摄像头实际的工作情况多种多样，有些一年四季都需要不间断工作。所以目标检测检测模型实际应对的数据将远比其训练时的数据差。为此，本文需要一个贴近实际的数据集来模拟各种显示情况。而且此数据集能够是一个公开而被广泛认同的数据集。

CDnet[[[51]](#endnote-45)]是一个运动检测数据集，其内容比较贴近实际的各种生活场景的拍摄结果，而且数据都是来自真实相机且包括了很多恶劣的照明情况和天气情况。故本文将此数据集中的内容作为实际数据，进行实际检测。

6.1　CDnet数据集

目前几个公认的被广泛使用的数据集如MS COCO[[[52]](#endnote-46)]、CIFAR-100[[[53]](#endnote-47)]等获取的图像都是在十分良好的条件下得到的，都是在光照条件，天气条件的情况下拍摄的。虽然也有算法会针对这些不良条件进行研究，但是往往这些算法都只考虑某一个细致的问题，他们所使用的数据也只是适合其研究的问题。

CDnet数据集的建立就是为了在一般的甚至是有挑战性的场景中测试算法的实际能力。CDnet数据集可以作为一个公开的统一的标准来满足此类需求。

CDnet数据集的内容由一连串视频图像组成，其每一个类别都和一个实际的运动监测问题相关联。提供了目标检测中各种场景的数据，是一个都是由真实相机捕获到的平衡数据集。其结构如表6 – 1所示：

表6 - 1 CDnet2014数据集的结构

|  |  |
| --- | --- |
| 类别 | 主要场景 |
| 基线 | 高速公路，办公室，行人，PETS2006[[[54]](#endnote-48)] |
| 动态背景 | 小船，独木舟，马路，喷泉，人行立交桥 |
| 不稳定相机 | 网球场，林荫大道，人行道，公路 |
| 阴影 | 街道，马路，公交车站，复印店，走廊，人的影子 |
| 间歇性运动对象 | 十字路口，停车场，沙发，高速公路，电车站，冬季车道 |
| 远红外线摄像头 | 走廊，饭厅，湖边，图书馆，公园 |
| 恶劣天气 | 暴风雪，溜冰，下雪，雨夹雪 |
| 低帧率 | 码头\_17fps，电车十字路口\_1fps，隧道出口\_35fps，高速公路\_5fps |
| 夜间 | 环形路口，十字路口，高速公路交汇处，公路路口，电车站，积雪公路 |
| PTZ摄像机 | 连续场景，断续场景，位置切换场景，缩放 |
| 空气湍流 | 4个不同的不稳定气流场景。 |

基线(Baseline)的4个场景主要作为参考，场景简单易于检测。动态背景(Dynamic Background)中物体具有强烈的运动。这些对象会互相干扰，而且各自也会有很大形变。不稳定相机(Camera Jitter)在获取此类别中的数据时摄像机不停抖动，可以考验模型抵抗摄像机运动或者是图像模糊的能力。阴影(Shadow)，此类中的图像都会产生阴影，防止阴影被误认为ROI是产生此类别的目的。间歇性运动对象(Intermittent Objection Motion)中的图像包含鬼影，即静止物体突然运动导致有重影和叠影。远红外线摄像头(Thermal)图像都是由远红外摄像头捕捉的，图像清晰度低，且为灰度图。PTZ类的图像由PTZ摄像机捕捉，可以考验模型在切换检查场景时是否可以快速的重新恢复检测。恶劣天气主要是指下雪天，此时能见度较低，图像模糊。低帧率(Low Frame-Rate)的图像普遍帧数较低，图像不连贯，时域信息不足。夜间图像照明条件不好，整体像素偏暗，物体难以分辨。空气湍流(Air Turbulence)是在极端热浪天气情况下得到的，此种情况下受热空气的影响物体会发生形变和扭曲。

CDnet的Ground Truth已经被细致的标注到像素级别。一共有5种级别：静态对象的像素为0，物体的阴影被标记为50，非的ROI的对象其像素为85，未知对象的像素被标为170，移动物体的像素则是255。

6.2　在CDnet数据集上的比较与分析

6.1.1 Baseline

6.1.2 Dynamic Background

6.1.3 Camera Jitter

6.1.4 Shadow

6.1.5 Intermittent Objection Motion

6.1.6 Thermal

6.1.7 PTZ

6.1.8 Bad Weather

6.1.9 Low Frame-Rate

6.1.10 Night Videos

6.1.11 Air Turbulence

6.3　本章小节

本章解释了使用CDnet数据集的原因。介绍了CDnet数据集的结构，和各类主要场景的内容，介绍了各场景对于检测模型的考验。比较了Faster-RCNN原始模型和加入混合之后的改进模型在CDnet上的表现，分析其优缺点。

第7章　总结与展望

7.1　总结

自2012年深度学习爆发以来，深度卷积神经网络在多个领域发展迅猛。但是在实际的实践中深度卷积神经网络的不足也不断暴露出来，而现有的理论不足以支持深度学习的发展。故本文以基于深度卷积神经网络的目标检测算法Faster-RCNN为研究对象，探索改进深度卷积神经网络的方法。

7.1.1 目标检测算法的发展

本文分析了多个目标检测算法的原理，包括传统的以及基于深度卷积神经网络的目标检测算法。总结了这些算法的流程和各个模块。得出了以下结论：

1. 特征算子。传统方法特征算子是人为设计的。深度卷积神经网络作为算法的特征算子的优秀之处在于其提取特征能力的全面。这是传统算法被逐步淘汰的根本原因。
2. 算法流程。两类算法流程是相同的，都是采集特征、根据特征分类、边界框回归。

7.1.2 深度卷积神经网络的不足与改进

本文结合最近提出的新的理论VRM理论和Mixup理论等，研究了深度卷积神经网络的不足，提出了将Mixup技术运用于深度卷积神经网络的想法。本文将

多种混合加入到实际的Faster-RCNN代码之中，并训练出了新的改进的模型，将新模型和原始模型的比较后得出结论：算法。

7.1.3 改进模型的实际运用

本文将改进的模型结合实际的数据，不仅验证了改进的有效性，而且。

7.2　展望

本文对于深度卷积神经网络的改进有许多不足之处，只是选用了Faster-RCNN算法，加入了简单的混合。并且由于数据集以及硬件设备等因素等局限性，对于深度卷积神经网络的改进结合他人新的研究，在今后的研究中可以从以下多个方面入手。

7.2.1 的控制

的值只采用的一个超参数进行控制，超参数是固定的，的值也是固定的，本文还没有针对不同的进行对比实验。所以，可以像在训练时变化学习率一样可，自适应的寻找新的值。

7.2.2 加入新的正则化形式

VRM理论否定了通用的正则化在深度学习中的作用。但是最近一种新的正则化DropBlock[[[55]](#endnote-49)]被提了出来，其实验证明DropBlock对于深度卷积神经网络模型的泛化能力的提升是有显著帮助的。

7.2.3 改进边界框回归的方式

现有的边界框回归只是将4个参数输入线性回归函数进行学习，只用到了两个点，使用的参数太少了。我们有理由相信，如果使用多个点进行回归，如结合Cornernet[[[56]](#endnote-50)]、Grid R-CNN[[[57]](#endnote-51)]，边界框预测的准确度将会提高。

。

1. 参考文献

   [] 韩光. 运动目标检测理论与方法[M]. 北京:电子工业出版社, 2018. [↑](#endnote-ref-2)
2. [] 陈哲. 图像目标检测技术集应用[M]. 北京:人民邮电出版社, 2016. [↑](#endnote-ref-3)
3. [] Haykin S. Neural networks: a comprehensive foundation[M]. Prentice Hall PTR, 1994. [↑](#endnote-ref-4)
4. [] Deng J, Dong W, Socher R, et al. Imagenet: A large-scale hierarchical image database[C]//2009 IEEE conference on computer vision and pattern recognition. Ieee, 2009: 248-255. [↑](#endnote-ref-5)
5. [] Krizhevsky A, Sutskever I, Hinton G E. Imagenet classification with deep convolutional neural networks[C]//Advances in neural information processing systems. 2012: 1097-1105. [↑](#endnote-ref-6)
6. [] Nair V, Hinton G E. Rectified linear units improve restricted boltzmann machines[C]//Proceedings of the 27th international conference on machine learning (ICML-10). 2010: 807-814. [↑](#endnote-ref-7)
7. [] Perez L, Wang J. The effectiveness of data augmentation in image classification using deep learning[J]. arXiv preprint arXiv:1712.04621, 2017. [↑](#endnote-ref-8)
8. [] Zhang H, Cisse M, Dauphin Y N, et al. mixup: Beyond empirical risk minimization[J]. arXiv preprint arXiv:1710.09412, 2017. [↑](#endnote-ref-9)
9. [] Wong S C, Gatt A, Stamatescu V, et al. Understanding data augmentation for classification: when to warp?[C]//2016 international conference on digital image computing: techniques and applications (DICTA). IEEE, 2016: 1-6. [↑](#endnote-ref-10)
10. [] Goodfellow I, Bengio Y, Courville A. Deep learning[M]. MIT press, 2016. [↑](#endnote-ref-11)
11. [] Lowe D G. Distinctive image features from scale-invariant keypoints[J]. International journal of computer vision, 2004, 60(2): 91-110. [↑](#endnote-ref-12)
12. [] Dalal N, Triggs B. Histograms of oriented gradients for human detection[C]//international Conference on computer vision & Pattern Recognition (CVPR'05). IEEE Computer Society, 2005, 1: 886--893. [↑](#endnote-ref-13)
13. [] Bay H, Tuytelaars T, Van Gool L. Surf: Speeded up robust features[C]//European conference on computer vision. Springer, Berlin, Heidelberg, 2006: 404-417. [↑](#endnote-ref-14)
14. [] Neubeck A, Van Gool L. Efficient non-maximum suppression[C]//18th International Conference on Pattern Recognition (ICPR'06). IEEE, 2006, 3: 850-855. [↑](#endnote-ref-15)
15. [] Suykens J A K, Vandewalle J. Least squares support vector machine classifiers[J]. Neural processing letters, 1999, 9(3): 293-300. [↑](#endnote-ref-16)
16. []唐进民.深度学习之PyTorch实战计算机视觉[M].北京:电子工业出版社,2018 [↑](#endnote-ref-17)
17. [] 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. [↑](#endnote-ref-18)
18. [] Girshick R. Fast r-cnn[C]//Proceedings of the IEEE international conference on computer vision. 2015: 1440-1448. [↑](#endnote-ref-19)
19. [] 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. [↑](#endnote-ref-20)
20. [] He K, Gkioxari G, Dollár P, et al. Mask r-cnn[C]//Proceedings of the IEEE international conference on computer vision. 2017: 2961-2969. [↑](#endnote-ref-21)
21. []Dai J, Li Y, He K, et al. R-fcn: Object detection via region-based fully convolutional networks[C]//Advances in neural information processing systems. 2016: 379-387. [↑](#endnote-ref-22)
22. [] Redmon J, Divvala S, Girshick R, et al. You only look once: Unified, real-time object detection[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 779-788. [↑](#endnote-ref-23)
23. [] Liu W, Anguelov D, Erhan D, et al. Ssd: Single shot multibox detector[C]//European conference on computer vision. Springer, Cham, 2016: 21-37. [↑](#endnote-ref-24)
24. [] Yoo D, Park S, Lee J Y, et al. Attentionnet: Aggregating weak directions for accurate object detection[C]//Proceedings of the IEEE International Conference on Computer Vision. 2015: 2659-2667. [↑](#endnote-ref-25)
25. [] Uijlings J R R, Van De Sande K E A, Gevers T, et al. Selective search for object recognition[J]. International journal of computer vision, 2013, 104(2): 154-171. [↑](#endnote-ref-26)
26. [] Zeiler M D, Fergus R. Visualizing and understanding convolutional networks[C]//European conference on computer vision. springer, Cham, 2014: 818-833. [↑](#endnote-ref-27)
27. [] 周志华. 机器学习[M]. 清华大学出版社, 北京, 2016. [↑](#endnote-ref-28)
28. [] 李航. 统计学习方法[J]. 清华大学出版社, 北京, 2012. [↑](#endnote-ref-29)
29. [] Vapnik V. Principles of risk minimization for learning theory[C]//Advances in neural information processing systems. 1992: 831-838. [↑](#endnote-ref-30)
30. [] Chapelle O, Weston J, Bottou L, et al. Vicinal risk minimization[C]//Advances in neural information processing systems. 2001: 416-422. [↑](#endnote-ref-31)
31. [] He K, Zhang X, Ren S, et al. Spatial pyramid pooling in deep convolutional networks for visual recognition[J]. IEEE transactions on pattern analysis and machine intelligence, 2015, 37(9): 1904-1916. [↑](#endnote-ref-32)
32. 1 Jaccard系数用于比较有限样本集之间的相似性与差异性 [↑](#footnote-ref-1)
33. [] Rätsch G, Onoda T, Müller K R. Soft margins for AdaBoost[J]. Machine learning, 2001, 42(3): 287-320. [↑](#endnote-ref-33)
34. [] Shawe-Taylor J, Bartlett P L, Williamson R C, et al. Structural risk minimization over data-dependent hierarchies[J]. IEEE transactions on Information Theory, 1998, 44(5): 1926-1940. [↑](#endnote-ref-34)
35. [] Szegedy C, Vanhoucke V, Ioffe S, et al. Rethinking the inception architecture for computer vision[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 2818-2826. [↑](#endnote-ref-35)
36. []Verma V, Lamb A, Beckham C, et al. Manifold Mixup: Learning Better Representations by Interpolating Hidden States[J]. 2018. [↑](#endnote-ref-36)
37. [] SnalTyan[EB/OL]. <https://github.com/SnailTyan/deep-learning-papers-translation> [↑](#endnote-ref-37)
38. [] Long J, Shelhamer E, Darrell T. Fully convolutional networks for semantic segmentation[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2015: 3431-3440. [↑](#endnote-ref-38)
39. [] Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition[J]. arXiv preprint arXiv:1409.1556, 2014. [↑](#endnote-ref-39)
40. [] He K, Zhang X, Ren S, et al. Identity mappings in deep residual networks[C]//European conference on computer vision. Springer, Cham, 2016: 630-645. [↑](#endnote-ref-40)
41. 1 此处以及下文都称Mixup技术为混合 [↑](#footnote-ref-2)
42. [] arleyzhang[EB/OL].<https://arleyzhang.github.io/articles/1dc20586/> [↑](#endnote-ref-41)
43. <http://host.robots.ox.ac.uk/pascal/VOC/> [↑](#footnote-ref-3)
44. [] Everingham M, Van Gool L, Williams C K I, et al. The pascal visual object classes (voc) challenge[J]. International journal of computer vision, 2010, 88(2): 303-338. [↑](#endnote-ref-42)
45. 1 <http://secure.flickr.com/> [↑](#footnote-ref-4)
46. 1 <http://host.robots.ox.ac.uk:8080/> [↑](#footnote-ref-5)
47. [] ruotianluo[EB/OL].<https://github.com/ruotianluo/pytorch-faster-rcnn> [↑](#endnote-ref-43)
48. 1 <https://github.com/jcjohnson/pytorch-vgg> [↑](#footnote-ref-6)
49. 2 <https://github.com/ruotianluo/pytorch-resnet> [↑](#footnote-ref-7)
50. [] Sin\_Geek[EB/OL].<https://blog.csdn.net/sin_geek/article/details/82587435> [↑](#endnote-ref-44)
51. [] Wang Y, Jodoin P M, Porikli F, et al. CDnet 2014: an expanded change detection benchmark dataset[C]//Proceedings of the IEEE conference on computer vision and pattern recognition workshops. 2014: 387-394. [↑](#endnote-ref-45)
52. [] Lin T Y, Maire M, Belongie S, et al. Microsoft coco: Common objects in context[C]//European conference on computer vision. Springer, Cham, 2014: 740-755. [↑](#endnote-ref-46)
53. [] Krizhevsky A, Hinton G. Learning multiple layers of features from tiny images[R]. Technical report, University of Toronto, 2009. [↑](#endnote-ref-47)
54. [] Young D P, Ferryman J M. Pets metrics: On-line performance evaluation service[C]//2005 IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance. IEEE, 2005: 317-324. [↑](#endnote-ref-48)
55. [] Ghiasi G, Lin T Y, Le Q V. DropBlock: A regularization method for convolutional networks[C]//Advances in Neural Information Processing Systems. 2018: 10727-10737. [↑](#endnote-ref-49)
56. [] Law H, Deng J. Cornernet: Detecting objects as paired keypoints[C]//Proceedings of the European Conference on Computer Vision (ECCV). 2018: 734-750. [↑](#endnote-ref-50)
57. [] Lu X, Li B, Yue Y, et al. Grid R-CNN[J]. 2018.

    致　　谢

    历经近三个月的毕业设计即将落下帷幕，敬请各位老师对我的设计过程和设计成果作最后检查。

    毕业设计是研究生生涯的完结，更是对研究生所学知识的一种全面的总结与展现。从陌生到了解，从了解到熟悉，这是每个人在学习路上所必经的一个过程，我对压缩感知视频跟踪的认识也是如此。经过三个月的努力，我相信这次毕业设计一定能为研究生阶段划上一个完美的句号，也将为我未来的视野奠定坚实的基础。在这一整个过程中，我得到了周围老师和同学们的很多帮助，我表示由衷的感谢。

    首先，要感谢我的毕业设计指导老师何发智教授，何教授为人热心随和，给了提供了研究课题和详细的研究资料，多亏了何老师的帮助我才能从零到一、从无到有，完成这次的毕业设计。

    然后，要感谢毕业设计完成过程中一直给予我指导和帮助的全权同学、余波同学、司灵通同学。在此表示衷心的感谢。

    其次，感谢这篇论文中所引用到文献的各位作者，多亏了他们的研究成果所给予的启发和方向性，我才能成功的完成这篇论文。

    最后感谢两年来所有指导我的老师以及关心帮助我的同学们。谢谢！

    由于我能力有限，所撰写的论文定有不少不足之处，敬请各位老师批评和指正。

    附录：攻读硕士学位期间参与的科研工作 [↑](#endnote-ref-51)