

南京理工大学泰州科技学院

毕业设计说明书(论文)

作者: 蒋恩 学号: 1803900137

学院(系): 计算机科学与工程学院

专业: 计算机科学与技术

题目: 图像目标检测算法的研究与实现

指导者: 姜枫 教授

评阅者: 贾波 教授

2022 年 5 月

毕业设计说明书（论文）中文摘要

图像目标检测是计算机视觉领域不可或缺的重要组成部分，它往往用于提取图像内物体位置和类别信息。近年来随着用于检测任务的深度学习网络的快速发展，目标检测器的性能得到了极大的提高。为了深入了解目标检测的现状，本文首先由传统的图像检测方法开始，从背景和历史发展角度进行叙述，一直论述到了现代基于深度学习的目标检测方法。接着逐一分析并介绍了几种主流的现代目标检测方法。之后着重介绍了经典的图像目标检测算法 Faster R-CNN (Faster Regions with CNN) 原理。通过实验复现 Faster R-CNN 算法，先对 Faster R-CNN 网络模型进行训练，多次训练后进行了预测，将预测结果进行了精度和召回率分析，以 mAP (平均 AP 值) 值等评估参数分析其精度。最后对基于深度学习的目标检测算法的发展进行总结反思。

关键词 目标检测 Faster R-CNN mAP 深度学习

毕业设计说明书（论文）外文摘要

Title Research And Implementation of The Image Object

Detection Algorithm

Abstract

Image object detection is an indispensable part of the field of computer vision, which is often used to extract object location and category information within an image. With the rapid development of deep learning networks for detection tasks, the performance of object detectors for recent. In order to deeply understand the current situation of object detection, this paper begins with the traditional image detection method, narrates from the perspective of background and historical development, and discusses the modern object detection method based on deep learning. Several mainstream modern target detection methods are then analyzed and introduced. Then, the classical algorithm Faster R-CNN (Faster Regions with CNN) is introduced. Through the experimental repetition of Faster R-CNN algorithm, the Faster R-CNN network model was trained and predicted after many training. The prediction results were analyzed, and their accuracy was analyzed by evaluation parameters such as mAP (average AP value) value. Finally, the development of the object detection algorithm based on deep learning is summarized and reflected.

Keywords Object-Detection Faster R-CNN mAP Deep-Learning

目 录

1 引言（或绪论）	1
1.1 课题背景及研究的目的意义	1
1.2 国内研究现状	2
1.3 本文主要工作和组织结构	2
2 基于深度学习的目标检测网络模型	4
2.1 RCNN 模型	4
2.2 SPP-Net 模型	4
2.3 YOLO 模型	5
2.4 SSD 模型	6
2.5 本章小结	7
3 Faster R-CNN 模型的介绍	8
3.1 共享卷积层	9
3.2 RPN 网络	9
3.3 RoI Pooling 与分类网络	11
3.4 损失函数	12
3.5 本章小结	13
4 实验及分析	14
4.1 数据集和实验平台选择	14
4.2 训练过程	15
4.3 训练评估参数	20
4.4 准确率与召回率分析	21
4.5 目标检测效果	22
4.6 本章小结	22
结束语	23
致谢	24
参考文献	25

1 绪论

近年来，目标检测被广泛应用于安全、军事、交通、医疗等领域，是计算机视觉、机器学习、图像处理不可或缺的组成部分。目标检测即目标提取，是一种将待测目标信息从图像中提取出来的手段。绝大多数目标检测器通过对图像提取特征、分类和定位获得待测目标的类别和位置信息。

1.1 课题背景及研究的目的意义

图像目标检测方法在过去的几十年可以分为两个时期，一是采取手工提取特征的传统图像目标检测方法时期，二是基于深度学习的现代目标检测方法时期。

传统的图像检测方法使用滑动窗口或图像分割生成大量的候选框，接着再对候选框进行特征提取，传统的特征提取一般通过采集物体的边缘、颜色、纹理等，例如 HOG^[1] (Histogram of Oriented Gradient) 边缘检测算子、SIFT^[2] (Scale-invariant Feature Transform) 局部特征检测算子，最后由分类器对输出的特征进行判断类别。尽管传统的目标检测方法已经取得了一定的检测效果，但是仍存在着一些不足：（1）手工提取到的特征鲁棒性差；（2）滑动窗口生成候选框的方法浪费了大量时间且实行成本高。

传统的目标检测流程如图 1.1 所示：

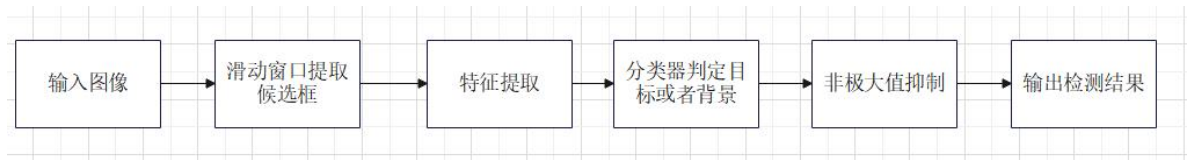


图 1.1 传统目标检测流程

为解决传统目标检测方法的不足之处，现代图像目标检测成功将深度学习与目标检测相融合，提取特征方式由手工提取变为了卷积神经网络。现代图像目标检测分为单阶段和双阶段方法，单阶段方法通常一步到位，检测速度快，YOLO^[3]和 SSD^[4]就是其中的典型代表。而 RCNN、Fast R-CNN、Faster R-CNN 此类需要多次运行检测和分类的模型就是双阶段方法。它们更好地优化了传统的检测方法，增加了精确率。单阶段模型与双阶段模型的不同之处在于，单阶段往往比较直接，仅利用回归就完成了对目标的检测；双阶段会先生成候选框，并利用网络大概得到目标框的位置和类别，最后进行修正得到目标检测结果。双阶段模型拥有更高的检测准确率，在公共数据集中的表现也更突出，但单阶段模型的检测速度更快，因此更适用于目标的实时检测。

从原先手工提取特征大跨度到如今智能化的卷积神经网络提取，目标检测器发展是一步一步提升的，因此想要在现代图像目标检测方法的广度和深度进行突破，对目标检测的背景历史进行了解并且熟悉几种主流的现代目标检测方法的原理是很有必要的。

1.2 国内外研究现状

近年来，由于图像目标检测领域深度和广度的不断提升，大批量优秀的图像目标检测相继出现。首先是将卷积网络与目标检测相融合的 R-CNN 算法在 2014 年由 Girshick 等人提出，由于其鲁棒性高、精度高，一度在目标检测领域中流行。YOLO 算法在 2015 年由 Redmon 等人紧接着^[5]提出，YOLO 算法特点就是速度快，模式相对比较直接，仅利用一个卷积神经网络就成功的预测了目标的位置和种类，接着又对 YOLO 算法做出完善并相继提出 YOLOv2^[6]和 YOLOv3^[7]算法，改进后的 YOLOv3 算法显著检测精度和速度。同年 Girshick 等人^[8]将 RCNN 算法改进，提出了 Fast R-CNN 算法。2017 年大名鼎鼎的 Faster R-CNN 问世，同样是由 Girshick 等人^[9]提出。Faster R-CNN 将 Fast R-CNN 与 RPN (Region Proposal Network) 相结合，精度大大提升。戴陈卡等^[10]将 Faster R-CNN 与多部件方法相结合提高了模型的检测速度。林莉等人^[11]提出基于候选框生成和多尺度自适应学习的小目标检测算法，成功提高对小目标的检测。董永峰等人^[12]以 Mask-RCNN (Mask Region-based Convolutional Neural Network) 为基础与残差网络相结合应用于遥感。孙佳等人^[13]基于 YOLO 算法成功改进了网络结果以及增强了视频目标检测，检测速度和检测精度显著提升。如今两类目标检测算法的代表 YOLO 系列和 RCNN 系列都有许多他们各自的优缺点，前者检测精度较高，但是速度较慢，对小目标检测能力不足，并且物体实时性检测效果也差强人意；后者直接对图像进行预测分析，因此检测速度较快，但是精度不够高，对于小目标的检测常常不尽人意。

而现如今的图像目标检测算法研究已经不是单一研究其算法本身，许多将图像检测与日常生活相结合的研究也不断出现。例如将其用于改进车辆实时检测、将其用于工业界的工件表面瑕疵检测等等。

1.3 本文主要工作和组织结构

本文主要首先阐述了图像识别技术的背景和现状，接着对现有的目标检测算法以及其核心技术点进行了逐一介绍和分析。接着本文着重介绍了目前较为先进前沿的图

像目标检测算法——基于 Faster R-CNN (Faster Regions with CNN) 的图像识别方法的原理。最后使用 python 语言以及 vscode 编译器搭建了基于深度学习的卷积神经网络模型，以 pascal voc 2007 为数据集，采用 Faster R-CNN 算法对图像进行处理，将待测图像进行卷积处理，并通过准确率、回归率和 mAP 值对 Faster R-CNN 算法进行了评估和分析比较。将论文主要结构分布展示如下：

第一章是绪论部分。笼统介绍了图像目标检测算法的历史，由传统迈入现代的进化过程。接着主要围绕国内外现代图像检测算法的时间线对算法展开了粗略介绍，最后将本文的主要内容和组织结构总体进行整合和叙述。

第二章分别介绍了 RCNN、YOLO 等几种主要的目标检测算法，详细描述了它们的原理并加以分析。

第三章是是对本文选取的复现模型，具体的介绍了 Faster-RCNN 算法的各个组成部分、运行细节以及分析了损失函数，主要是以 Faster-RCNN 的重要组成部分即共享卷积层、RPN、ROI Pooling 与分类网络分别展开叙述。

第四章为实验过程与结果分析，利用 Pascal voc 2007 数据集进行训练，并对 Faster-RCNN 网络进行了复现，通过大量图片进行预测并得到结果，对结果进行了准确率、召回率分析，计算得出 mAP 值总结得出 Faster R-CNN 算法的精度高这一优点，最后对本章进行小结。

2 基于深度学习的目标检测网络模型

本章细致的介绍了几种基于深度学习的目标检测算法，以原理入手对 R-CNN、YOLO、SPP-NET 和 SSD 网络展开了分析，最后加以总结优缺点。

2.1 R-CNN 模型

在 R-CNN（Region-CNN，基于区域的卷积网络方法）^[15]的问世前，传统的目标检测方法是机械的、非智能的。通过大范围穷举将有可能存在物体的区域框挑选出，接着对这些区域进行提取特征并加以分类，最后利用非极大值抑制方法（NMS）对区域框进行处理然后在输出结果。非极大值抑制方法的核心理念是抑制非最大值元素然后得到局部的最大值，而 R-CNN 的不同之处在于，在提取特征这一步，R-CNN 采用了 Selective Search^[16]（选择性搜索）的方法代替了传统的滑动窗口法提取候选区域，接着再在这些候选区域上提取特征。Selective Search 较为聪明的一点是将图像分成很多个小的区域，接着按照一定的合并原则对这些小区域进行合并，将其进行多次合并，图像最终被合并成一个区域，然后进行输出。合并规则是颜色纹理相近的、吻合度高的进行合并，并且为了防止大区域吞并小区域，合并后的区域总面积要小。因此 R-CNN 算法由生成候选区域（选择性搜索）、将候选区域提取特征（卷积神经网络）、对于特征进行判断类别（SVM 分类器（Support Vector Machine，支持向量机））和修正位置偏移量（利用回归器）这四个步骤组成。流程图如下图 1.1：

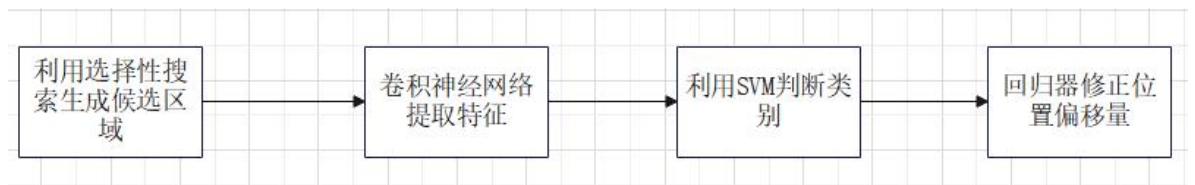


图 2.1 R-CNN 的基本流程

图片进入 RCNN 模型中往往需要被修改成固定的尺寸，这导致了 RCNN 算法的许多缺点，首先是速度瓶颈：选择性搜索在使用过程中会生成大量候选区域，并且要将这些生成的候选区域一一进行特征提取处理，大大降低了检测速率。同时 RCNN 性能上也存在着瓶颈。对于每个候选区域放缩到固定尺寸会导致出现几何畸变，并且由于速度瓶颈的存在，不可能使用大量的数据增强训练模型。

2.2 SPP-NET 模型

在 SPP-NET (Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition, 用于视觉识别的深度卷积网络中的空间金字塔池) 诞生之前, 图像进入卷积神经网络都需要转化为固定尺寸的图片, 在此过程中许多原本图像的信息经过裁剪、变化所损失, 为了对这些问题进行改进, SSP-NET 不再对原有图片进行变形等操作, 而是直接将卷积特征进行 SPP 池化处理, 得到固定大小的特征图 (feature maps), 这样处理极大的保留了原有图片的信息, 也显著提升了速度。

经过 SPP 池化后得到的固定大小的特征图将会被送入全连接层, 通过 SVM 二分类进行判断类别再通过非极大值抑制得到合适的框以及利用边框回归进行修正

SPP-NET 网络模型流程图如图 2. 1:

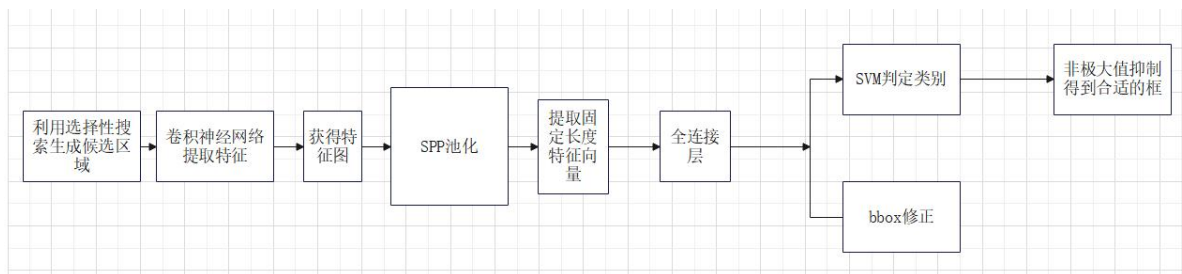


图 2. 2SPP-NET 网络模型的流程图

SPP-NET 通常使用不同大小的块将每个候选框进行划分, 分别是 4×4 , 2×2 , 1×1 的块, 并把每个块都进行特征提取处理, 得到特征图, 接着将每个块都最大池化法处理并进行采样, 由于 SPP-NET 只对图片使用了一次特征提取, 这也大大提升了检测速度。

SPP-NET 有减少了重复操作、保留了原有图片信息、成功加快了检测速度, 但是仍然有一些问题存在: 训练阶段卷积层比较固定, 遇到新任务时需要再次微调卷积层; 由于 SVM 的存在, 特征需要写入磁盘, 训练依旧耗时较长、效率低下。

2. 3 YOLO 模型

YOLO (You Only Look Once) 网络模型顾名思义得到物体的类别和位置信息仅需要一次目标检测, 它的特点是直接快速。YOLO 网络模型不需要提前找到可能存在目标的区域, 而是直接使用回归器对目标物体进行检测代替了以往的需要生成候选框的方法,

YOLO 模型采用整图的方式来对候选框进行筛选, 可在依次运算过程中得到多个候选框的位置和候选区域物体的类别, 能够更好的区别目标物体和背景。YOLO 模型首先

对图片进行分割处理，它将图片分割为 $S \times S$ 个网格，这些网格的大小是相等的，每个网络相当于一个任务，YOLO 模型相较于滑动窗口最大的不同在于，滑动窗口只能识别出一个物体且要求这个物体必须在窗口内，而 YOLO 则不同，它的关键点在于只需要这个物体的中心点落在这个窗口内，此时就可以进行检测。候选框一共有五个变量分别是物体的中心点坐标 (x, y) 和它的宽高 (w, h) 以及置信度评分。

YOLO 网络模型流程如图 2.2

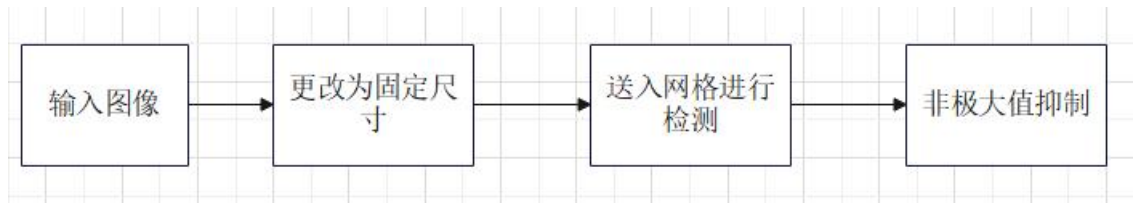


图 2.3 YOLO 网络模型流程

YOLO 网络模型去除了大量多余设计过程，也因此显著提升速度。图像进入 YOLO 网络模型大小首先会被改变，接着将改变好尺寸的图像送入卷积层和全连接层，利用卷积层和全连接层分别对图像进行特征提取操作和判断图像的位置和类别，最后通过 NMS 进行预测。

YOLO 网络模型虽然设计精巧、速度快，但是仍然有许多不足的地方，比如一个网格只能识别出一个物体，且小目标预测效果很差，以及 YOLO 的网络检测模型精度是不如 R-CNN 这样双阶段检测方法的。但这些缺点在 YOLO v2 和 YOLO v3 中将得到更巧妙的设计优化。

2.4 SSD 模型

SSD^[15] (Single Shot multibox Detector) 单次检测器模型主要使用当前位置周围的特征来预测某个位置。相较于其他目标检测算法，SSD 最大的不同之处在于采用了多尺度特征图和设置了先验框用于检测，特征映射图意思是将图像进行卷积层转换处理后所得到的多维数值。

SSD 网络由三个重要组成部分组成。第一个就是基础网络，基础网络结构主要由 VGG-16 构成，这部分作用是为了提取低尺度的特征映射图；第二部分是辅助卷积层，这部分的作用是接受上部分传输来的低尺度的特征映射图，并将其送入卷积神经网络卷积神经网络进行处理得到 4 个高尺度的特征映射图，该层通过卷积对不同的特征图来提取检测结果；而这第三部分是预测卷积层，用于预测特征映射图每个点的矩形框信息和所属类别的信息。并且通过卷积滤波器可以对每个添加卷积层预测进行预测，

最后通过非最大化抑制输出结果。

SSD 在 YOLO 模型的基础上主要改进了三点：利用多尺度变换对图片进行特征提取处理、利用卷积进行检测、设置先验框。同时是 SSD 在拥有单阶段模型的速度优点时，精确度也不逊于双阶段模型。缺点是 SSD 在小目标的处理方面往往不尽人意，并且无法满足实时检测。

2.5 本章小结

本章是对几种常见的目标检测网络模型进行细致的介绍，首先以 R-CNN 为开端对原理进行了具体讲解，R-CNN 是第一个成功的深度学习目标检测算法，在此基础上，SPP-NET 等模型相继出现。可以看到每一个目标检测算法的诞生都是基于先前目标检测算法的基础并加以改进。在下一章，经典双阶段算法 Faster R-CNN 网络模型将会被着重介绍。

3 Faster R-CNN 模型的介绍

Faster R-CNN（基于区域更快的卷积神经网络方法）是截止目前 RCNN 系列算法的最杰出产物，双阶段中最为经典的图像目标检测算法。预测第一阶段先找出图片中待检测物体的 anchor 矩形框（对背景、待检测物体进行二分类），第二阶段对 anchor 框内待检测物体进行分类。R-CNN 系列物体检测算法的思路都是，先产生一些待检测框，再对检测框进行分类。Faster R-CNN 使用神经网络生成待检测框，替代了其他 R-CNN 算法中通过规则等产生候选框的方法。

Faster R-CNN 是在 Fast R-CNN 的基础上，提出 RPN 的概念，将原有的选择性搜索方法进行替换，也可以说 Faster R-CNN 就是 Fast R-CNN 的结合体。将它的具体网络结构模型如图 3-1

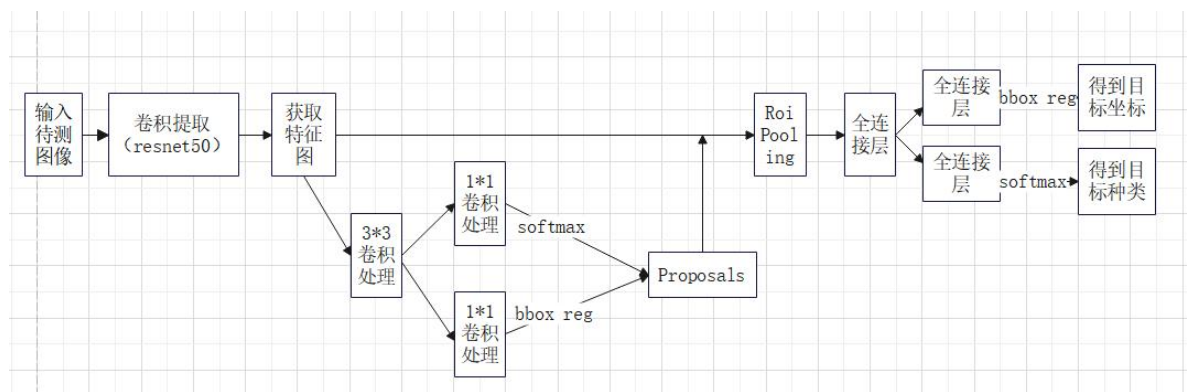


图 3-1 Faster R-CNN 网络结构模型

Faster R-CNN 算法主要流程如下：

待测图像首先进入 backbone(主干特征提取网络)，该层用于提取待测图像的特征，例如 Resnet50，对图像特征提取处理输出特征图，特征图分为两个流向，RPN 将其中的一部分用于生成候选区域，另一部分流向 Roi Pooling 层池化。进入 RPN 网络这一部分的特征图首先要分别进行一个 3*3 卷积处理和 2 个 1*1 卷积处理，输出两个特征图，其中一个特征图利用 softmax 二分类参照框获得有物体的锚框，另一个特征图通过 bbox reg (bounding box regression) 获得参照框的四维向量（建议区域的中心点坐标、宽、高）。Proposals 则负责综合所有有物体的参照框和它们的坐标，整合出建议区域送入 Roi Pooling 层。此时 Roi Pooling 层一共有最开始的一部分特征图和 RPN 输出的建议区域两个输入，Roi Pooling 层将其统一尺寸并提取出建议特

征图，然后将建议特征图送入全连接层。全连接层再次利用 softmax 交叉熵函数判断出建议区域物体的种类有哪些，并计算出物体可能是该种类的可能性；同时再次利用边框回归计算出建议框与实际的偏离值，将其用于修改目标检测框，以达到精确的检测效果。

3.1 共享卷积层

在 Faster R-CNN 特征提取中，许多主干特征提取网络可供选择，常见的有 Resnet50, Xception 等等，以 Resnet50 为例，Faster R-CNN 会把输入进来的图片首先变为固定的尺寸。ResNet50 由两个基本块 Conv Block 和 Identity Block 组成，其中 Conv Block 输入和输出由于维度不同不能进行串联，它的作用是将网络的维度更改；Identity Block 可以将输入维度和输出维度串联，它的作用是可用来加深网络。

Resnet50 网络总共包含了 49 个卷积层和 1 个全连接层。并且 Resnet50 网络结构被分成了七个部分，第一部分不包含残差块，主要对输入进行卷积、正则化、激活函数处理、最大池化的计算。第二、三、四、五部分结构都包含了残差块，图中的绿色图块不会改变残差块的尺寸，只用于改变残差块的维度。在 Resnet50 网络结构中，残差块都有三层卷积，那网络总共有 49 个卷积层，加上最后的全连接层总共是 50 层，这也是 Resnet50 名称的由来。网络的输入为 $224 \times 224 \times 3$ ，经过前五部分的卷积计算，输出为 $7 \times 7 \times 2048$ ，池化层会将其转化成成一个特征向量，最后分类器会对这个特征向量进行计算并输出类别概率

3.2 RPN 网络

RPN 利用 9 个不同尺寸的窗口对区域进行选择搜索，假设得到 1000 个区域，并得到了个 256 维向量，接着进行两次全连接得到 2000 个分数（1000 个 positive anchors 和 1000 个 negative anchors 的得分）和 4000 个坐标（每个区域都用中心点坐标和宽高这个四维向量来表示）。2000 个分数用来分别图像是否包含物体，4000 个位置坐标则用来得到有物体的图像的坐标。RPN 网络的结构图如下图 3.2

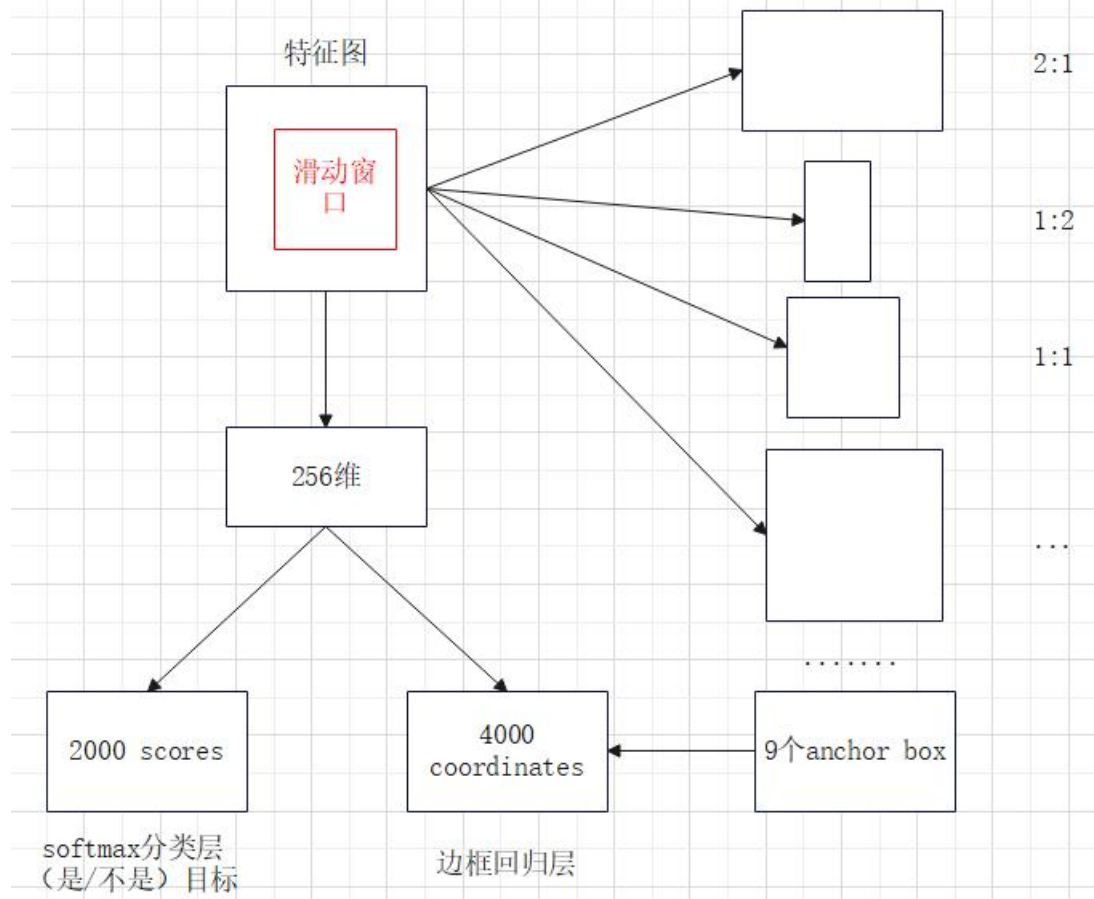


图 3.2 RPN 网络结构图

图像经过一系列卷积处理得到特征图。一部分流向 Roi pooling 层池化，一部分则进入 RPN 网络生成待检测框。因为 RPN 生成候选区域的方式是滑动窗口，所以整个图像的参数都是一样的。这里假设输入的图像为 $n \times 16 \times 16$ 。流入 RPN 网络的特征图要先做一个 3×3 卷积升维为 256 维（进行升维目的是使特征图鲁棒性更强），再分别做两次 1×1 卷积得到两个特征图（分别含有 $2 \times 9 \times 16 \times 16$ 和 $4 \times 9 \times 16 \times 16$ 的 anchor box）。其中的 9 指特征图的每个像素点含有 9 个 anchor box，2 指每个 anchor 有两个得分 scores，分别代表的是 positive anchor 的得分和 negative anchor 的得分，4 是指每个 anchor box 包含 4 个偏移量（中心点坐标和高宽）然后将其中一个特征图（ $2 \times 9 \times 16 \times 16$ ）经过 reshape（因为每个特征图的大小不一样）后送入 softmax 二分类。根据 positive anchor 的得分和 negative anchor 的得分判断得出有物体的目标框。另一个特征图（ $4 \times 9 \times 16 \times 16$ ）通过 bbox reg（bounding box regression）获得 anchor 框的二维向量（候选区域的中心点坐标、宽、高）。此时我们得到了含有物体的目标框和它相应的二维向量，Proposals 的作用就是在此，Proposals 将所有 positive anchors 和它们的坐标进行综合，计算出精准的建议框送入 Roi Pooling 层。

Softmax 二分类的实现由 softmax 交叉熵损失函数进行，逻辑回归的二分器是他的前身。Softmax 交叉熵函数如下：

$$y_i = \frac{e^{a_i}}{\sum_{k=1}^c e^{a_k}} \quad \forall i \in 1 \dots c \quad (3-1)$$

3.3 Roi pooling 与分类网络 Classification

Roi pooling layer 实际上是 SPP-NET 的一个精简版，SPP-NET 对每个 proposal 使用了不同大小的金字塔映射，Roi pooling layer 利用 RPN 生成的建议区域和 Resnet50 最后一层得到的特征图，得到固定大小的建议特征图，然后将其送入全连接层。具体操作如下：Roi pooling 首先根据输入图像，将 Roi 映射到特征图对应的位置，接着将映射后的区域划分为相同大小的块，最后对每个块最大池化处理。Roi pooling 层的实现如图 3.3 所示

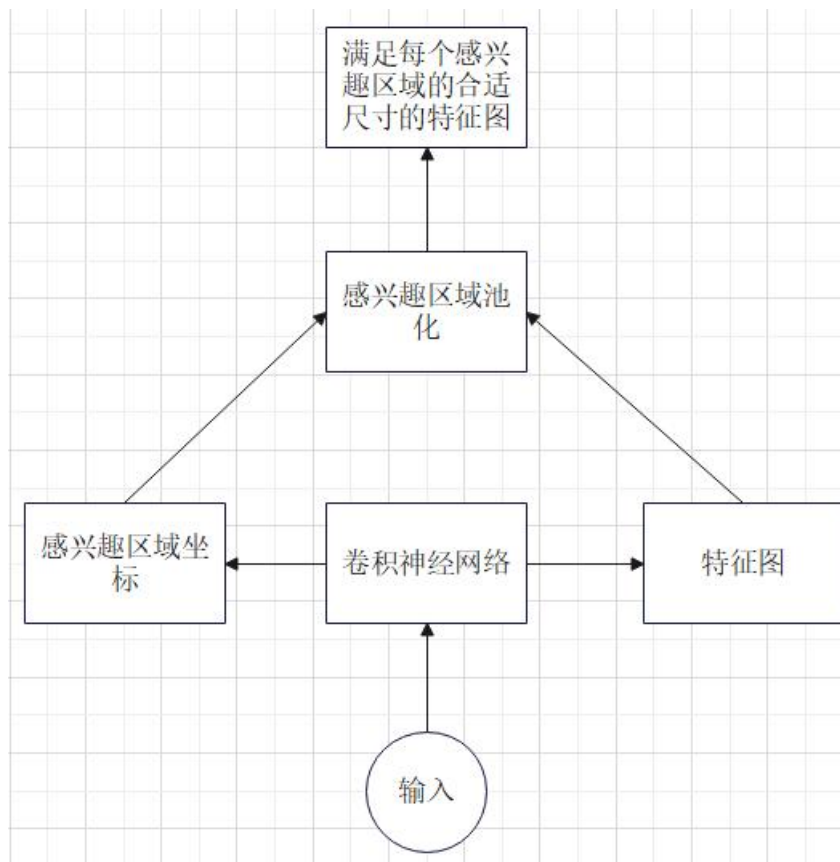


图 3.3 Roi pooling 层的实现图

这样我们就可以从不同大小的方框得到固定大小的相应的特征图。值得一提的是，输出的特征图的大小不取决于 Roi 和卷积特征图大小。ROI pooling 最大的好处

就在于极大地提高了处理速度。

Classification 部分包含了 Roi pooling 层和全连接层。池化后所得的建议特征图首先进入全连接层，全连接层利用 softmax 交叉熵函数和边框回归对图像进行分类和回归处理。

bbox (bounding box regression) 也就是边框回归，对于一个窗口往往是由中心点坐标、宽、高这样的四维变量来表示。全连接层所得到的预测框往往是与实际目标有一定偏移，而边框回归就是来解决这个问题的，边框回归对预测框的四维变量平移加放缩得到新的边框，进行处理后所得的边框与真实的边框类似。最终通过一系列处理，获得了较为精确的建议区域以及建议区域的位置坐标

3.4 损失函数

Faster R-CNN 的损失主要由分类和回归损失构成。损失函数计算公式如下：

$$L(\{p_i\}, \{t_i\}) = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*) + \lambda \frac{1}{N_{reg}} \sum_i p_i^* L_{reg}(t_i, t_i^*) \quad (3-2)$$

分类损失即公式的前半段 $\frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*)$ ，RPN 中的分类损失主要是来源于训练 RPN 的过程中会选择 256 个 anchor，256 就是公式中的 N_{cls} 。 p_i 是 anchor 预测为 positive anchor 的概率， $L_{cls}(p_i, p_i^*)$ 是两个类别 positive anchor 和 negative anchor 的对数损失。因此 $L_{cls}(p_i, p_i^*) = -\log[p_i^* p_i + (1-p_i^*)(1-p_i)]$ 。这是一个经典的 softmax 二分类交叉熵损失。对每个 anchor 计算对数损失，然后求和除以总的 anchor 数量 N_{cls} 就得到了 RPN 的分类损失函数。

回归损失即公式的后半段 $\lambda \frac{1}{N_{reg}} \sum_i p_i^* L_{reg}(t_i, t_i^*)$ ，其中的 $t_i = \{t_x, t_y, t_w, t_h\}$

是一个向量，是建议框在 RPN 训练阶段的预测偏移量， t_i^* 是实际的偏移量。

其中 $L_{reg}(t_i, t_i^*) = R(t_i - t_i^*)$ ， R 代表的是 smooth_{L1} 函数，其中的 $x = t_i - t_i^*$ 。

Smooth_{L1} 函数为：

$$\text{smooth}_{L1}(x) = \begin{cases} 0.5x^2 \times 1/\sigma^2 & \text{if } |x| < 1/\sigma^2 \\ |x| - 0.5 & \text{otherwise} \end{cases} \quad (3-3)$$

在损失计算中，背景的损失值不被计算在内，因此得出 $L_{reg}(t_i, t_i^*)$ 的值后之后还要乘以 P^* ， P^* 用于判断是否为背景。在实际过程中， λ 的作用是用来平衡 N_{cls} 和 N_{reg} ，避免他们的差距太大，这也使得总的损失计算中回归损失和分类损失相对均匀。

3.4 本章小结

本章详细介绍了 Faster R-CNN 网络模型的运行流程，并分别对卷积提取层、RPN 网络、分类回归等进行了细致的原理介绍。下一章将会对 Faster R-CNN 网络进行复现，对其网络模型进行训练，并且进行精度和召回率等分析。

4 实验及分析

4.1 数据集和实验平台选择

本次实验选用 Pascal voc 2007 作为数据集。Pascal voc 2007 数据集由 5011 幅训练集和 4952 幅测试集组成，总共包含了人、牛、马、飞机等 20 个不同种类。

部分 Pascal voc 2007 数据集展示图如图 4.1|4.2:

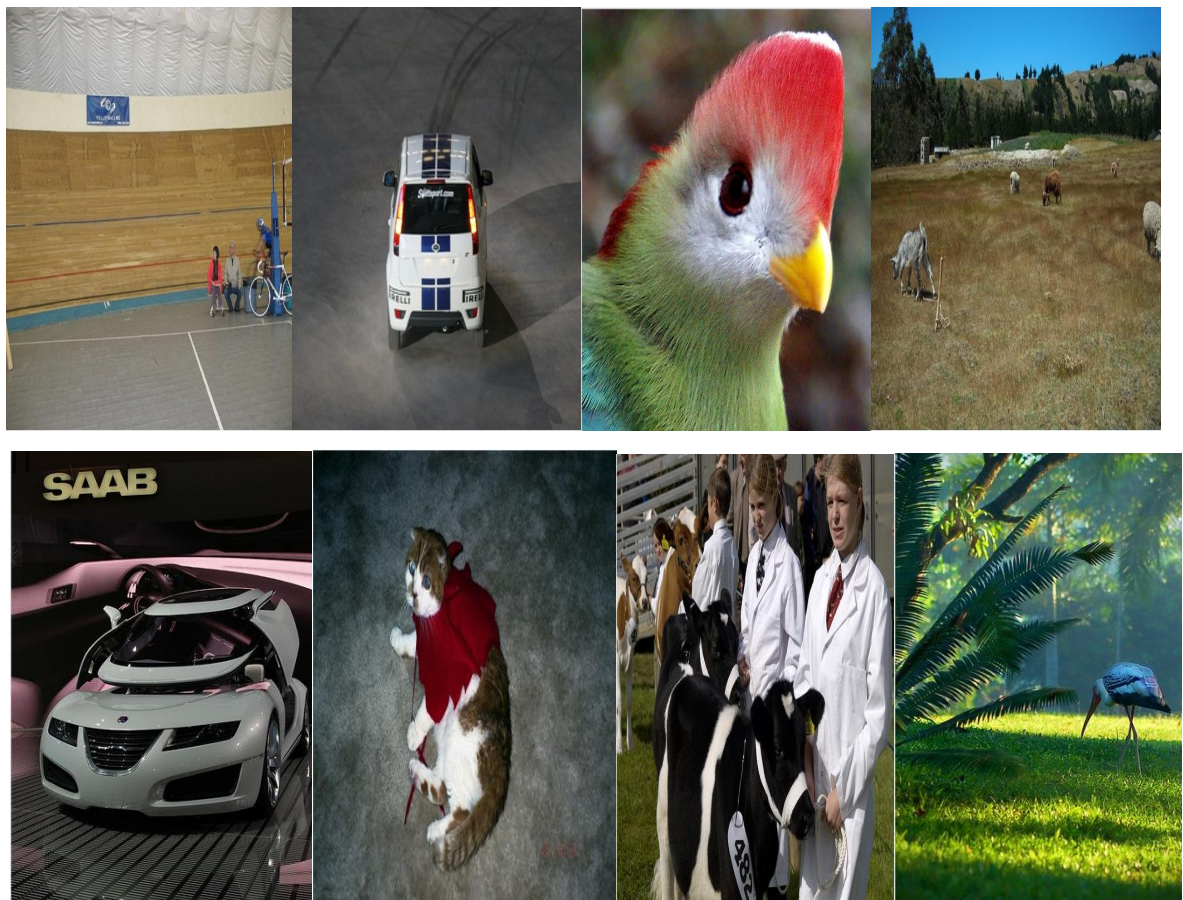


图 4.1 部分 Pascal voc 2007 展示图

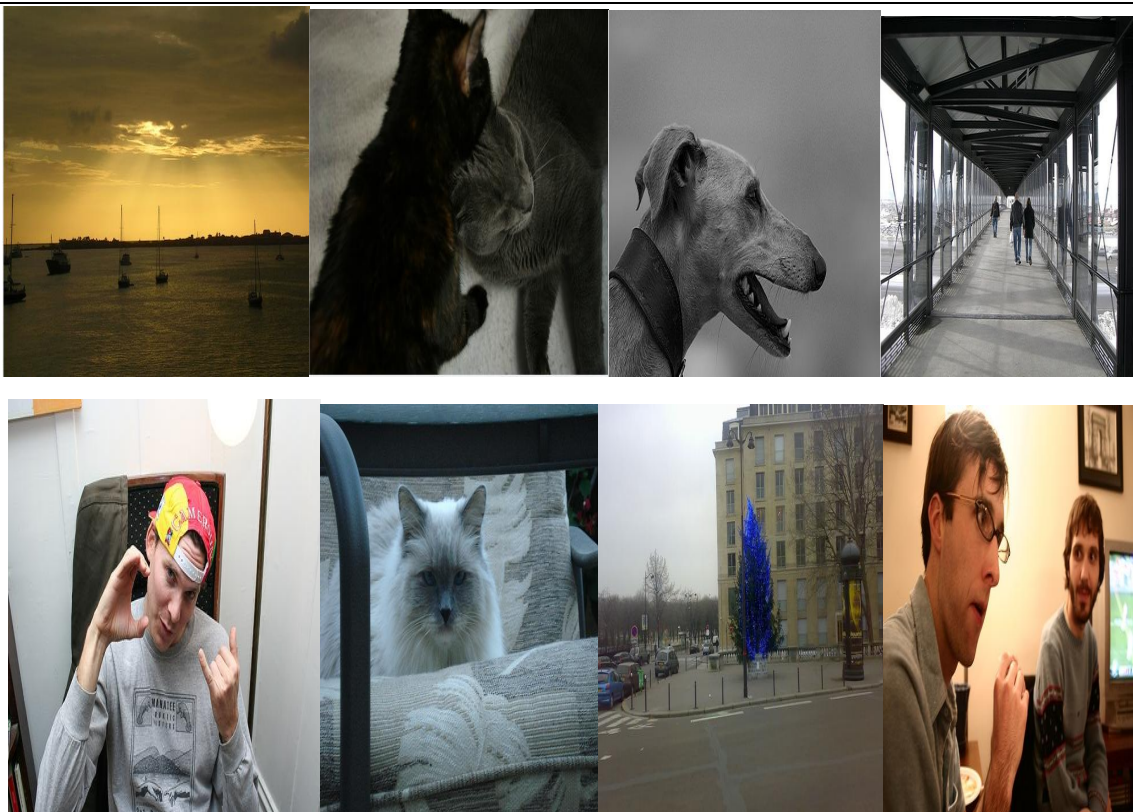


图 4.2 部分 Pascal voc 2007 展示图

此次实验主要在个人电脑进行，中央处理器采用 intel i7-7700，GPU 采用 GTX1050TI，内存 8G，操作系统为 Windows10，使用 python 语言搭载 Pytorch 框架进行编程，编译器使用 Vscode，语言版本为 Python3.6。

4.2 训练过程

对于 Faster R-CNN 的训练，由于 Faster R-CNN 是在 Fast R-CNN 基础上构建的，因此我们将 Faster R-CNN 分为 RPN 网络和 Fast R-CNN 两部分。首先需要训练 Resnet50，在此基础上进行训练。实际中训练过程第一个步骤是在已经训练好的 Resnet50 上，训练 RPN 网络；步骤二是接着利用上步骤中训练好的 RPN 网络，收集 proposals；步骤三是进行第一次训练 Fast RCNN 网络；第四个步骤进行第二次训练 RPN 网络；接着再次利用步骤四中训练好的 RPN 网络，收集 proposals；最后像第三个步骤一样进行第二次训练 Fast RCNN 网络。

训练流程图如下图 4.1 所示

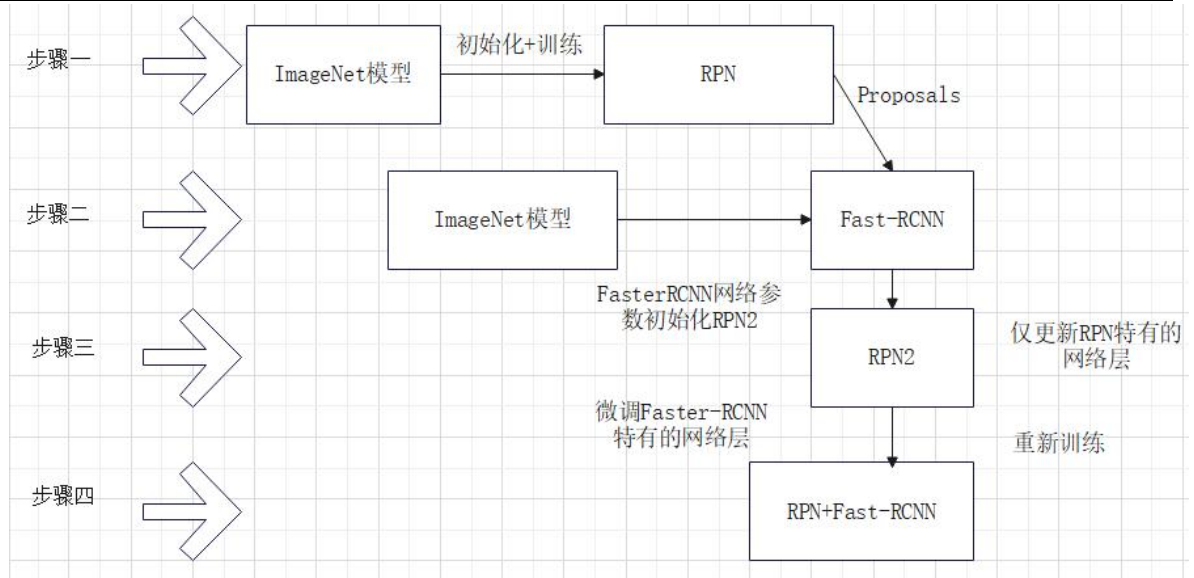


图 4.1Faster R-CNN 训练流程图

也就是说 Faster R-CNN 进行一次训练分为训练 RPN 网络和训练 Fast RCNN 两大主要模块。

RPN 网络训练的部分代码如下：

```

1  def bbox_iou(bbox_a, bbox_b):
2      if bbox_a.shape[1] != 4 or bbox_b.shape[1] != 4:
3          print(bbox_a, bbox_b)
4          raise IndexError
5      tl = np.maximum(bbox_a[:, :2], bbox_b[:, :2])
6      br = np.minimum(bbox_a[:, :2], bbox_b[:, :2])
7      area_i = np.prod(br - tl, axis=2) * (tl < br).all(axis=2)
8      area_a = np.prod(bbox_a[:, :2] - bbox_a[:, :2], axis=1)
9      area_b = np.prod(bbox_b[:, :2] - bbox_b[:, :2], axis=1)
10     return area_i / (area_a[~, None] + area_b - area_i)
11
12 def bbox2loc(src_bbox, dst_bbox):
13     width = src_bbox[:, 2] - src_bbox[:, 0]
14     height = src_bbox[:, 3] - src_bbox[:, 1]
15     ctr_x = src_bbox[:, 0] + 0.5 * width
16     ctr_y = src_bbox[:, 1] + 0.5 * height
17
18     base_width = dst_bbox[:, 2] - dst_bbox[:, 0]
19     base_height = dst_bbox[:, 3] - dst_bbox[:, 1]
20     base_ctr_x = dst_bbox[:, 0] + 0.5 * base_width
21     base_ctr_y = dst_bbox[:, 1] + 0.5 * base_height
22
23     eps = np.finfo(height.dtype).eps
24     width = np.maximum(width, eps)
25     height = np.maximum(height, eps)

```

```
27     dx = (base_ctr_x - ctr_x) / width
28     dy = (base_ctr_y - ctr_y) / height
29     dw = np.log(base_width / width)
30     dh = np.log(base_height / height)
31
32     loc = np.vstack((dx, dy, dw, dh)).transpose()
33     return loc
34
35 class AnchorTargetCreator(object):
36     def __init__(self, n_sample=256, pos_iou_thresh=0.7, neg_iou_thresh=0.3, pos_ratio=0.5):
37         self.n_sample = n_sample
38         self.pos_iou_thresh = pos_iou_thresh
39         self.neg_iou_thresh = neg_iou_thresh
40         self.pos_ratio = pos_ratio
41
42     def __call__(self, bbox, anchor):
43         argmax_ious, label = self._create_label(anchor, bbox)
44         if (label > 0).any():
45             loc = bbox2loc(anchor, bbox[argmax_ious])
46             return loc, label
47         else:
48             return np.zeros_like(anchor), label
49
50     def _calc_ious(self, anchor, bbox):
51         #-----#
```

```
51         #-----#
52         #   anchor和bbox的iou
53         #   获得的ious的shape为[num_anchors, num_gt]
54         #-----#
55         ious = bbox_iou(anchor, bbox)
56
57         if len(bbox)==0:
58             return np.zeros(len(anchor), np.int32), np.zeros(len(anchor)), np.zeros(len(bbox))
59         #-----#
60         #   获得每一个先验框最对应的真实框 [num_anchors, ]
61         #-----#
62         argmax_ious = ious.argmax(axis=1)
63         #-----#
64         #   找出每一个先验框最对应的真实框的iou [num_anchors, ]
65         #-----#
66         max_ious = np.max(ious, axis=1)
67         #-----#
68         #   获得每一个真实框最对应的先验框 [num_gt, ]
69         #-----#
70         gt_argmax_ious = ious.argmax(axis=0)
71         #-----#
72         #   保证每一个真实框都存在对应的先验框
73         #-----#
74         for i in range(len(gt_argmax_ious)):
75             argmax_ious[gt_argmax_ious[i]] = i
76
77         return argmax_ious, max_ious, gt_argmax_ious
```

```

77         return argmax_ious, max_ious, gt_argmax_ious
78
79     def _create_label(self, anchor, bbox):
80         # ----- #
81         # 1是正样本, 0是负样本, -1忽略
82         # 初始化的时候全部设置为-1
83         # ----- #
84         label = np.empty((len(anchor),), dtype=np.int32)
85         label.fill(-1)
86
87         # ----- #
88         # argmax_ious为每个先验框对应的最大的真实框的序号 [num_anchors, ]
89         # max_ious为每个真实框对应的最大的真实框的iou [num_anchors, ]
90         # gt_argmax_ious为每一个真实框对应的最大的先验框的序号 [num_gt, ]
91         # ----- #
92         argmax_ious, max_ious, gt_argmax_ious = self._calc_ious(anchor, bbox)
93
94         # ----- #
95         # 如果小于门限值则设置为负样本
96         # 如果大于门限值则设置为正样本
97         # 每个真实框至少对应一个先验框
98         # ----- #
99         label[max_ious < self.neg_iou_thresh] = 0
100        label[max_ious >= self.pos_iou_thresh] = 1
101        if len(gt_argmax_ious) > 0:
102            label[gt_argmax_ious] = 1
103
104        # ----- #
105        # 判断正样本数量是否大于128, 如果大于则限制在128
106
107        # ----- #
108        n_pos = int(self.pos_ratio * self.n_sample)
109        pos_index = np.where(label == 1)[0]
110        if len(pos_index) > n_pos:
111            disable_index = np.random.choice(pos_index, size=(len(pos_index) - n_pos), replace=False)
112            label[disable_index] = -1
113
114        # ----- #
115        # 平衡正负样本, 保持总数量为256
116        # ----- #
117        n_neg = self.n_sample - np.sum(label == 1)
118        neg_index = np.where(label == 0)[0]
119        if len(neg_index) > n_neg:
120            disable_index = np.random.choice(neg_index, size=(len(neg_index) - n_neg), replace=False)
121            label[disable_index] = -1
122
123        return argmax_ious, label

```

Fast RCNN 网络的训练部分代码如下：

```

1 class ProposalTargetCreator(object):
2     def __init__(self, n_sample=128, pos_ratio=0.5, pos_iou_thresh=0.5, neg_iou_thresh_high=0.5, neg_iou_thresh_low=
3         self.n_sample = n_sample
4         self.pos_ratio = pos_ratio
5         self.pos_roi_per_image = np.round(self.n_sample * self.pos_ratio)
6         self.pos_iou_thresh = pos_iou_thresh
7         self.neg_iou_thresh_high = neg_iou_thresh_high
8         self.neg_iou_thresh_low = neg_iou_thresh_low
9
10    def __call__(self, roi, bbox, label, loc_normalize_std=(0.1, 0.1, 0.2, 0.2)):
11        roi = np.concatenate((roi.detach().cpu().numpy(), bbox), axis=0)
12        # -----#
13        # 计算建议框和真实框的重合程度
14        # -----#
15        iou = bbox_iou(roi, bbox)
16
17        if len(bbox)==0:
18            gt_assignment = np.zeros(len(roi), np.int32)
19            max_iou = np.zeros(len(roi))
20            gt_roi_label = np.zeros(len(roi))
21        else:
22            #-----#
23            # 获得每一个建议框最对应的真实框 [num_roi, ]
24            #-----#
25            gt_assignment = iou.argmax(axis=1)
26            #-----#
27
28            #-----#
29            # 获得每一个建议框最对应的真实框的iou [num_roi, ]
30            #-----#
31            max_iou = iou.max(axis=1)
32            #-----#
33            # 真实框的标签要+1因为有背景的存在
34            #-----#
35            gt_roi_label = label[gt_assignment] + 1
36
37            #-----#
38            # 满足建议框和真实框重合程度大于neg_iou_thresh_high的作为负样本
39            # 将正样本的数量限制在self.pos_roi_per_image以内
40            #-----#
41            pos_index = np.where(max_iou >= self.pos_iou_thresh)[0]
42            pos_roi_per_this_image = int(min(self.pos_roi_per_image, pos_index.size))
43            if pos_index.size > 0:
44                pos_index = np.random.choice(pos_index, size=pos_roi_per_this_image, replace=False)
45
46            #-----#
47            # 满足建议框和真实框重合程度小于neg_iou_thresh_high大于neg_iou_thresh_low作为负样本
48            # 将正样本的数量和负样本的数量的总和固定成self.n_sample
49            #-----#
50            neg_index = np.where((max_iou < self.neg_iou_thresh_high) & (max_iou >= self.neg_iou_thresh_low))[0]
51            neg_roi_per_this_image = self.n_sample - pos_roi_per_this_image
52            neg_roi_per_this_image = int(min(neg_roi_per_this_image, neg_index.size))
53            if neg_index.size > 0:
54                neg_index = np.random.choice(neg_index, size=neg_roi_per_this_image, replace=False)

```

```
53
54     #-----#
55     #   sample_roi      [n_sample, ]
56     #   gt_roi_loc     [n_sample, 4]
57     #   gt_roi_label   [n_sample, ]
58     #-----#
59     keep_index = np.append(pos_index, neg_index)
60
61     sample_roi = roi[keep_index]
62     if len(bbox)==0:
63         return sample_roi, np.zeros_like(sample_roi), gt_roi_label[keep_index]
64
65     gt_roi_loc = bbox2loc(sample_roi, bbox[gt_assignment[keep_index]])
66     gt_roi_loc = (gt_roi_loc / np.array(loc_normalize_std, np.float32))
67
68     gt_roi_label = gt_roi_label[keep_index]
69     gt_roi_label[pos_roi_per_this_image:] = 0
70     return sample_roi, gt_roi_loc, gt_roi_label
```

4.3 实验评估参数

在实验评估时，假设将目标分成正例 positive 和负例 negative 两类，本文以人物 x 为例，正例 positive 是人物 x，负例 negative 是除人物 x 以外的其他人物，具体参数如下：

True positives(tp): 人物 x 的图片被正确的识别成人物 x；

True negatives(tn): 其他人物的图片未被识别为人物 x，系统正确地认为成对应的其他人的图片；

False positives(fp): 其他人物的图片被错误的识别成人物 x；

False negatives(fn): 人物 x 的图片未被识别为人物 x，系统错误地认为成其他人物的图片。

本次训练评估内容，主要包括准确率 P(precision)、召回率 R(recall)、交并比 Iou(intersection over union)。

(1) 正样本的准确率 Precision，就是在识别某物体时，正确识别为人物 x 的图片占被系统识别的人物的图片之比，即预测样本中实际包含的正例数与所有的正例数之比，也就是在识别人物 x 时，在被系统识别为人物 x 的图片中准确识别为人物 x 的比例，准确率公式如下： $Precision = tp / (tp + fp)$

(2) 正样本的召回率 Recall，就是预测样本中实际正样本数与预测的样本数之比，即在输入的人物 x 图片中有多少被正确的识别为人物 x，召回率越高，准确率越低，召回率公式如下： $Recall = tp / (tp + fn)$

（3）交并比 IoU 指预交并比 IoU 测框和原标记框之间的重叠度，也就是预测框和原标记框重叠部分与集合部分之比。

4.4 准确率和召回率分析

实验选取了 450 张标签对应图片和 50 张错误图片为例，识别人物 x，对实验的准确率 Precision 和召回率 Recall 进行分析。交并比 IoU 的阈值在机器学习中被用于区分正例 positive 和负例 negative，超过规定的阈值，则为正例 positive，而反之则为负例 negative。本次分析选取不同的阈值对准确率和召回率进行统计。

准确率随阈值变化如下图 4.2:



图 4.2 准确率随阈值变化图

召回率随阈值变化如下图 4.3:



图 4.3 召回率随阈值变化图

由图像可得，在阈值为 0.5 时，实验的准确率为 79.33%，召回率为 83.51%，图像清楚的说明了准确率和召回率随着阈值的变化是矛盾的，不会同时增高也不会同时降低。实验说明在阈值为 0.5 时，实验的准确率和召回率都较高。

4.5 目标检测效果

在人工智能领域，机器学习的效果需要用各种指标来评价，本实验使用 mAP 来评估目标检测效果，mAP 就是平均 AP 值，是将多个数据集的 AP 值求出平均值所得，是衡量目标检测效果常用的指标。AP(Average-p) 是指平均精确度，对 PR (Precision-Recall) 曲线上的准确率求平均值。AP 值越高分类器性能越好。

具体本次实验 mAP 值如下图 4.4:

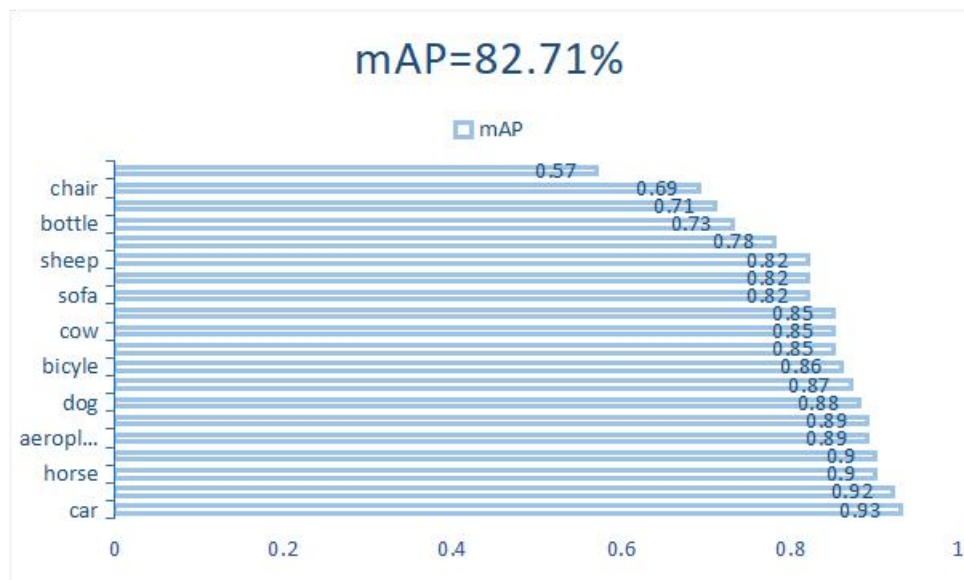


图 4.4 本次实验 mAP 值

由图所示本次实验的 mAP 值为 82.71%。这也又一次验证了 Faster R-CNN 算法精度较高这一特点。

4.6 本章小结

本章为复现 Faster R-CNN 的训练、实验部分，进行了准确率和召回率分析以及目标检测结果分析，传统的目标检测算法往往速度慢，精确度不够，但经过 rpn 网络改良的 Faster R-CNN 算法的精度大大增加，在大量的图片测试下，实验完成的效果较好，基本符合实验预期结果。

结束语

本文选取 Faster R-CNN 作为主要复现模型的原因是一一它是现存的图像目标检测算法中最让我所惊叹的。它的标志双阶段以一个常人无法想象的方法利用回归实现了精度的显著提升，RPN 网络是区分它和其他图像目标检测算法的关键之处。图像目标检测算法从传统的目标检测方法一步一步走来，它的经历也充满了曲折。但是如今的深度学习算法仍然存在着各种各样的缺点，例如本文的复现模型 Faster R-CNN，虽然双阶段显著的提升了图像目标检测精度，但是也正是因为双阶段，需要对预测框的不断调整，它的速度也随之变慢。图像检测算法是深度学习的智慧结晶；深度学习是人工智能的智慧结晶；而人工智能则是人类大脑的智慧结晶。我相信随着图像检测算法的不断优化，精度更高，速度更块的算法将会不断涌现。

致 谢

时至今日，即将毕业之时，过往的一幕幕仍在眼前，从刚迈入大学校园时的懵懂羞涩一直到现如今的成熟大方。在本论文完成之初，由于对深度学习领域知识的匮乏，我无从下手，是我的导师姜枫老师给予我最新文献的资料和许多外文网站。如果不是他，单单入门两个轻松之字我不知道要花费多久，当然还有论文中期阶段，姜枫老师不停的给我们开组会，给我们提出建议也为我们指点了迷津。他的严厉也让我知道了论文不可马虎潦草，态度很重要。因此也诚以谢意像姜枫老师表达。

大学的知识沉淀不仅让我拓宽了眼界，也对科学文化世界充满了敬畏之心。因此第二个感谢的对象是我的母校，感谢它接纳了青春懵懂的我，反哺以成熟的我，诚以最高的敬意向我的母校献出。

最后要感谢的对象是我的辅导员朱倩老师，她温柔的性格，细心的态度，让我感谢万分。这四年大学生活中，我有着大大小小的错误，也正是她宽容的原谅以及对我的不断关心给予我人生的第一课教育——不要怕犯错误，要勇于面对的不足，不断提高自己。因此，一声“谢谢”也应该送给她。

谨以这致谢向对本人论文、学习、生活等提供过帮助的人表示感谢！

参 考 文 献

- [1]Pang Y, Yuan Y, Li X, et al.Efficient HOG human detection[J].Signal Processing, 2011, 91(4):773-781.
- [2]Rublee E, Rabaud V, Konolige K, et al.ORB: An efficient alternative to SIFT or SURF[C].2011 International conference on computer vision.IEEE, 2011:2564-2571.
- [3]沈军宇, 李林燕, 夏振平等. 一种基于 YOLO 算法的鱼群检测方法[J]. 中国体视学与图像分析, 2018, 23(02):54-60.
- [4]朱敏. 基于 SSD 的行人检测方法[J]. 科学技术创新, 2017(36):75-76.
- [5]Li J, Liang X, Shen S M, et al.Scale-aware fast R-CNN for pedestrian detection[J]. IEEE transactions on Multimedia, 2017, 20(4):985-996.
- [6]Redmon J, Divvala S, Girshick R, et al. You only look once: unified, real-time object detection [A] . IEEE Conference on Computer Vision and Pattern Recognition [C] . USA: IEEE, 2016. 779-778.
- [7]Redmon J, Farhadi A. YOLO9000: Better, faster, stronger [A] . Computer Vision and Pattern Recognition, Hawaii [C] . USA: IEEE, 2017. 6517-6525.
- [8]Redmon J, Farhadi A. YOLOV3: An incremental improvement [A] . Computer Vision and Pattern Recognition [C] . USA: IEEE, 2018. 523-3541.
- [9]Girshick R. Fast R-CNN[C]//IEEE International Conference on Computer Vision.2015.
- [10]Ren, S Q, He, K M, Girshick R, et. al. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks[J]. IEEE Transactions on Pattern Analysis & Machine Intelligence, 2015, 39(6):1137-1149.
- [11]戴陈卡, 李毅, 基于 Faster RCNN 以及多部件结合的飞机场场面静态飞机检测[J], 计算机应用. 2017, 37(z2): 85-88.
- [12]林莉, 基于深度学习的小目标检测[D]. 电子科技大学, 2020.
- [13]亚妮, 汪西莉. 基于注意力和特征融合的遥感图像目标检测模型[J]. 激光与光电子学进展, 2021, 58(2): 3-28.
- [14]孙佳, 郭大波, 杨甜甜, 马识途. 基于改进的 YOLOv3 网络的实时目标检测[J]. 激光与光电子学进展, 2020, 57(22): 221-505.
- [15]王万国, 田兵, 刘越. 基于 RCNN 的无人机巡检图像 电力小部件识别研究[J]. 地球信息科学学报, 2017, 19(2): 256-263.

-
- [16]Uijlingd J R, De Sande K E, Gevers T, eta. Selec-tive search for object recognition[J]. International Journal of Computer Vision, 2013, 104(2) : 154—71.