**北京理工大学**

本科生毕业设计（论文）

**无锚点框的全卷积单阶段目标检测方法研究**

**Research of Anchor-Free Full Convolutional One-Stage Object Detection**

**学 院：** 计算机学院

**专 业：** 软件工程

**班 级：** 08111606

**姓 名：** 陶润洲

**指导教师：** 赵三元

2020年4月12日

**无锚点框的全卷积单阶段目标检测方法研究**

摘 要

目标检测作为计算机视觉和数字图像处理的一个热门方向，广泛应用于机器人导航、智能视频监控、工业检测、航空航天等诸多领域，通过计算机视觉减少对人力资本的消耗，具有重要的现实意义。

本课题主要研究无锚点框的全卷积单阶段目标检测方法，通过消除预定义的锚点框设置，该方法完全避免了与锚点框相关的复杂计算，大大减少了训练内存占用。该方法还避免了所有与锚点框相关并对最终检测性能非常敏感的超参数。由于该方法最后使用非最大抑制(NMS)对回归框进行处理，该检测器比以前的基于锚点框的单阶段检测器具有更加简单的优点。该方法还提出新分支中心度(center-ness)，用于打压距离目标中心较远位置的一定数量的低质量边界框，降低低质量检测框的权重，提高检测性能。

本文基于Python的科学计算软件包和Pytorch深度学习框架来搭建全卷积单阶段目标检测网络，并使用PASCAL VOC和COCO数据集进行模型的训练和评估，最后使用训练好的模型对实物图片进行目标检测。

**关键词：目标检测；深度学习；无锚点框**

**Research of Anchor-Free Full Convolutional One-Stage Object Detection**

Abstract

As a popular direction of computer vision and digital image processing, object detection is widely used in robot navigation, intelligent video surveillance, industrial inspection, aerospace and many other fields. It has great practical significance to reduce the consumption of the human capital through computer vision.

This paper mainly studies the full convolutional one-stage object detection without anchor boxes. By eliminating the predefined anchor box settings, this method completely avoids the complex calculations related to anchor boxes and greatly reduces the training memory footprint. This method also avoids all hyperparameters that are related to the anchor box and very sensitive to the final detection performance. Since this method finally uses non-maximum suppression(NMS) to process the regression box, the detector has a simpler advantage than the previous single-stage based on the anchor box. This method also proposes a new branch center-ness, which is used to suppress a certain number of low-quality bounding boxes far away from the target center, reduce the weight of low-quality detection boxes, and improve detection performance.

Based on Python’s scientific computing software package and Pytorch deep learning framework, this paper builds a full convolutional one-stage object detection network, and uses the PASCAL VOC and COCO datasets for model training and evaluation. Finally, the trained model is used to object detection for real pictures.

**Key words：object detection; deep learning; anchor-free**

**目录**

[摘 要 I](#_Toc38315169)

[Abstract II](#_Toc38315170)

[第1章 绪论 1](#_Toc38315171)

[1.1课题研究背景与意义 1](#_Toc38315172)

[1.2 国内外研究现状分析 2](#_Toc38315173)

[1.2.1 基于锚点框目标检测算法 2](#_Toc38315174)

[1.2.2 无锚点框目标检测算法 3](#_Toc38315175)

[1.3 研究内容和主要工作 4](#_Toc38315176)

[1.4 论文组织结构 4](#_Toc38315177)

[第2章 实验平台搭建与数据集处理 6](#_Toc38315178)

[2.1 相关的开发平台与主要技术介绍 6](#_Toc38315179)

[2.1.1 开发平台Anaconda与VScode 6](#_Toc38315180)

[2.1.2 深度学习框架Pytorch 6](#_Toc38315181)

[2.1.3 GPU加速运算库CuPy 7](#_Toc38315182)

[2.2 开发环境搭建 8](#_Toc38315183)

[2.2.1 实验环境介绍 8](#_Toc38315184)

[2.2.2 实验环境搭建 8](#_Toc38315185)

[2.3 数据集介绍与处理 8](#_Toc38315186)

[2.3.1 数据集介绍 8](#_Toc38315187)

[2.3.2 数据集处理 8](#_Toc38315188)

[2.4 本章小节 8](#_Toc38315189)

[第3章 全卷积单阶段目标检测网络的实现 8](#_Toc38315190)

[3.1 数据增强及预处理 8](#_Toc38315191)

[3.2 骨干网络搭建 8](#_Toc38315192)

[3.3 特征图金字塔网络与全卷积单阶段网络检测头搭建 8](#_Toc38315193)

[3.4 训练模型、评估模型、结果分析与检测实物功能实现 8](#_Toc38315194)

[3.5 本章小节 8](#_Toc38315195)

[第4章 基于全卷积单阶段目标检测网络的实验 8](#_Toc38315196)

[4.1 CuPy库加速运算 8](#_Toc38315197)

[4.2 VOC0712数据集 8](#_Toc38315198)

[4.3 COCO2017数据集 8](#_Toc38315199)

[4.4 实物检测 8](#_Toc38315200)

[4.5 本章小节 8](#_Toc38315201)

[结论 8](#_Toc38315202)

[**参考文献** 9](#_Toc38315203)

[致谢 9](#_Toc38315204)

1. 绪论

1.1课题研究背景与意义

当人类最开始发明计算机的时候，就在思考如何才能将计算机变得更加智能，来帮助人类完成一些工作。现如今，人工智能已经成为了一个非常热门的领域，不仅仅应用在众多活跃研究课题中，并且在人们生活的方方面面中也有很多实际的应用。目前，这个领域正在以几何倍的速度发展着，并且在未来也将继续健康发展。正因如此，人们也越来越希望可以通过人工智能来处理一些主观的、非规范性的事物，如识别图像等。

对于一些人类觉得处理起来很困难的抽象和形式化的任务，计算机却很擅长处理。在上个世界，计算机就在国际象棋方面崭露头角，战胜了人类棋手。但是直到最近几年，计算机在图像和语音的识别任务中才能达到人类的一般水平。一个人的思维发育是需要海量的外界相关知识，并且其中有很大一部分的知识是主观的，难以用形式化的符号和结构表示出来；如同人类一样，计算机也需要获取相等数量级的知识才能表现的足够智能。在如何让计算机学会这些主观的、非形式化的知识这一问题上，科学家首次提出了层次化的概念，借助人脑的工作方式，使得计算机通过构建简单的模型来学习一些较为复杂的特征。这种学习方式因其最终是构造出了一张“深层次”的图，且层与层之间通过简单的规则进行连接，所以被称为深度学习。

自2006年开始，大量的深度神经网络论文被发表，特别是在2012年，Hinton课题组首次参加ImageNet图像识别大赛，通过构建的CNN网络AlexNet一举多得冠军，从那之后，神经网络就开始收到广泛的关注。现如今，这项技术已经成功地应用在包括计算机视觉领域在内的多种模式分类问题上。而目标检测则是计算机视觉领域需要解决的基础任务之一，也是视频监控技术的基础任务。同时目标检测作为泛身份识别领域的一个基础算法，对之后的人脸识别、步态识别、人群计数、实例分割等任务有着至关重要的作用。所以研究出高效、准确的目标检测算法具有非常重要的意义。

现如今，几乎所有的最先进的目标检测网络，如RetinaNet、SSD、YOLOv3和Faster R-CNN都依赖于预定义的锚点框，而本课题研究的无锚点框全卷积单阶段目标检测网络，通过消除预定义的锚点框，避免了大量与锚点框相关的运算和相关的超参数。相比于目前主流的基于锚点框的单阶段网络，全卷积单阶段网络做到了涉及复杂度更低、性能较好，速度较快。鉴于其有效性和高效性，全卷积单阶段网络可以作为目前主流基于锚点框检测器的一个简单而又强大的替代品，甚至可以扩展到许多其他实例级的识别任务中去，对目标检测领域的发展有着较为重大的意义。

1.2 国内外研究现状分析

目标检测是“在哪里有什么”的任务，对于目标检测任务，目标的类别不确定、数量不确定、位置不确定、尺度不确定，根据传统非深度学习方法如VJ和DPM和早期深度学习方法如OverFeat，都使用金字塔多尺度和遍历滑窗的方式，逐尺度位置判断该尺度该位置处有没有可以识别的目标，非常笨重且耗时长。

锚点框这个概念首次提出是在Faster R-CNN论文中，通过预置的一组固定参考框，避免复杂的遍历和计算，直接判断该固定参考框和目标框的关系来进行检测，相较于之前的方法又快又好。但是同时还有以关键点等特征取代锚点框进行目标检测的方法，也能够达到类似的效果。所以根据有无锚点框，可以大致的将目标检测方法分为：基于锚点框目标检测算法和无锚点框目标检测算法。

1.2.1 基于锚点框目标检测算法

目前来说，比较顶尖的目标检测方法几乎都是用了锚点框技术。首先预定义一组不同尺寸不同位置的固定参考框，覆盖几乎所有的位置和尺寸，每个参考框均负责检测与其交并比大于阈值的目标，通过直接判断该参考框中有没有需要检测的目标以及目标框偏离参考框多远，来避免多尺度遍历滑窗，使得检测器检测效果好，并且检测时间短。

在首次提出锚点框概念的Faster R-CNN方法中，首先通过VGG16骨干网络输出256维的特征图，再通过区域提案网络(RPN)对之前输出的特征图进行预测。在RPN网络中，特征图先经过一个3×3的卷积层，最后进入两个分支。根据预置的由3个不同尺度{128, 256, 512}以及3个不同比例{1:1, 1:2, 2:1}所构成的9个固定的锚点框，两个分支分别会输出2×9=18个分类的分数以及4×9=36个回归的坐标。再将以上得到的分类器结果和回归结果输入到RPN网络的最后一部分Proposal Layer，对预测的回归边界框进行修正和结果的输出。根据Faster R-CNN论文锚点框训练后学习到的平均区域大小(见表1-1)显示，这9个固定的锚点框大约能覆盖到边长70~768的目标。

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| anchor | 128², 2:1 | 128², 1:1 | 128², 1:2 | 256², 2:1 | 256², 1:1 | 256², 1:2 | 512², 2:1 | 512², 1:1 | 512², 1:2 |
| proposal | 188×111 | 113×114 | 70×92 | 416×229 | 261×284 | 174×332 | 768×437 | 499×501 | 355×715 |

最终在使用锚点框之后，Faster R-CNN在VOC07和VOC12数据集上的mAP分别达到了73.2，70.4，在COCO数据集上mAP也达到了21.9的高度。这样的识别精度在当时众多算法框架中是非常高的，从这之后锚点框技术也广泛应用在众多的目标检测框架中，也都取得了很优异的效果。

在2016年ECCV的SSD框架中也采用了锚点框技术，同样也达到了很好的效果。SDD框架的作者认为仅仅只靠同一层特征图上的多个锚点框进行回归的话，精度还远远不够，因为有很大可能这一层上所有预置的锚点框和目标框的交并比都很小，这样一来训练误差就会很大。通过对多个层级上的锚点框计算其相关的交并比，去找到与目标框的尺寸、位置最接近的一批锚点框，这样在训练时也能达到最好的准确度。

在SSD框架中骨干网络采用与Faster R-CNN一样的VGG16网络，但是把最后的两层全连接层换成了普通的卷积层，之后再通过多个卷积层，最后得到6个不同尺度的特征图。对于这6层特征图每层分配4或6个不同形状的锚点框，对于不同尺度的特征图，采用不同尺度、不同比例的锚点框进行匹配 。根据原文的计算，6个特征图攻击8732个锚点框，总数量会比Faster R-CNN的RPN少很多，并且锚点框设置更加合理，小尺度锚点框多且密，大尺度锚点框少且疏。最终SSD框架在VOC07和VOC12数据集上得到的mAP分别达到了76.8，74.9，在COCO数据集上mAP也达到了十分优异的31.2。

1.2.2 无锚点框目标检测算法

自从2018年在ECCV上发表的CornerNet开始，无锚点框的目标检测模型层出不穷，最更是达到了井喷的状态，宣告无锚点框目标检测算法开始在目标检测领域占有一席之地。

早在2016年CVPR上发表提出的曾经大火的YOLO框架，就是最早的无锚点框目标检测模型。YOLO将物体检测的几个部分统一成一个简单的神经网络。YOLO模型关注整张图片以及在这张图片中所有物体的信息，所以模型支持使用整张图片的特征去预测每一个边界框，同时也支持对一张图片预测出所有的边界框。YOLO模型直接对全图进行训练，首先将输入图片分成S×S个网格，如果检测目标的中心落在了网格内，那么这个网格负责检测这个目标。每一个网格单元将预测B个边界框以及对应边界框的置信分数，以此来进行训练和回归。通过这样无锚点框的全局回归的方式，YOLO模型在目标检测时，处理速度很快，并且在VOC07和VOC12数据集上的mAP也分别达到了66.4和57.9。虽然效果相比Faster R-CNN较差，但是综合检测效果和处理速度而言，在同时期的检测器中还是很优异的。

而2018年发表在ECCV上的CornerNet则是通过关键点预测的方式来进行目标检测。CornerNet将检测目标的边界框这个任务转换成了检测边界框的左上角角点和右下角角点，然后将其组合起来完成检测任务，完全抛弃了锚点框这一概念。模型首先通过骨干网络沙漏网络(Hourglass Network)提取特征，之后分出两个单独的模块用来得到左上角角点和右下角角点的类别分类(Heatmaps)，并找到每个目标的一对关键点(Embeddings)，以及减少基于坐标回算目标位置时的偏置(Offsets)。通过所得到的这些信息来进行训练，最终CornerNet在COCO数据集上达到了42.1的好成绩，也同时为无锚点框目标检测算法框架打开了一扇大门。

1.3 研究内容和主要工作

本课题的主要任务是实现无锚点框全卷积单阶段目标检测模型。本课题的主要研究内容是：掌握深度学习中常用的数据预处理和数据增强方法，提高检测器训练效果；熟悉主流的目标检测模型网络结构，为搭建全卷积单阶段目标检测网络模型做准备；了解mAP等模型评估指标计算方式，以便对搭建好的网络模型进行评估。

本文完成工作如下：

1. 对数据集图像进行图像处理，实现数据预处理和数据增强。数据预处理部分主要对彩色图像进行色彩归一化处理；数据增强部分主要对彩色图像进行缩放、平移、左右翻转、上下翻转。
2. 搭建全卷积单阶段目标检测网络模型，结合原有的中心度分支增加中心部分采样处理，以提高检测器性能。
3. 对搭建好的网络模型进行训练、评估，并判断中心部分采样处理是否能提高检测器性能；使用训练好的模型检测实物图片，查看检测效果。

1.4 论文组织结构

本文的结构是严格按照此次课题的研究顺序进行安排的。本文详细结构如下：

第1章：绪论部分。介绍了深度学习与目标检测的技术发展，以及目标检测的发展对人工智能的重要意义。介绍了几款具有代表性的基于锚点框的目标检测框架与无锚点框目标检测框架，分析了锚点框的作用与意义以及无锚点框模型的处理流程。

第2章：主要介绍了与搭建网络模型相关的一些准备工作。首先对搭建网络模型相关的开发平台、所使用的深度学习框架和相关的重要Python库进行介绍；然后，介绍了实验的环境以及实验环境的搭建；最后介绍了本次实验所采用的数据集以及制作训练集和测试集的过程。

第3章：主要介绍了搭建网络模型的具体流程和相关工作。首先对数据进行增强与预处理；然后依次搭建骨干网络、特征金字塔网络和全卷积单阶段网络头部部分；最后实现训练、模型评估、结果分析与检测实物等功能。

第4章：主要介绍了搭建网络模型之后的实验过程。主要有以下几个部分：CuPy库对模型训练的加速效果；搭建好的模型分别在处理好的VOC0712与COCO2017数据集上的检测效果；训练完成的模型对于实物的检测效果。

1. 实验平台搭建与数据集处理

2.1 相关的开发平台与主要技术介绍

2.1.1 开发平台Anaconda与VScode

Anaconda是针对Python进行数据科学研究而专门建立的一组软件包，涵盖了大部分数据科学领域常用的Python库，并且为了解决软件环境以来问题，Anaconda自带有Conda包管理系统。在Anaconda的帮助下，数据工作者可以更加简单地处理在不同项目下对软件库甚至是对Python版本的不同需求。Anaconda包含有其可视化图形界面Anaconda Navigator，专门用来管理软件包和环境，可以在不使用命令行的情况下管理软件包和环境等。目前Anaconda也因其良好的用户体验，如今被广泛的应用在人工智能、科学计算、大数据及云计算、金融等领域。

VScode是微软推出的一款轻量级的代码编辑器，免费、开源而且功能强大，支持Windows，OS X和Linux系统。VScode内置有JavaScript、TypeScript和Node.js支持，而且自带有丰富强大的插件生态系统，可以通过安装插件来支持C++、C#、Python、PHP等不同的主流语言。对比于其他代码编辑器，VScode在打开特别大的文件时不会存在明显卡顿的情况，速度甚至可以和NotePad++相媲美。并且VScode在处理数据文件时，如json文件，有相对应的快捷键来格式化文件的排版，使得在处理数据文件时，能够更加舒适的查看数据格式。

VScode在开发基于Python语言的项目时，能够匹配Anaconda所创建的虚拟环境，以供用户选择。虽然VScode是代码编辑器，但是因为其自带底部终端的特质，在编写Python代码时，可以通过其快捷按钮来调用内部的终端，并使用选择好的Python环境来运行代码，十分方便快捷，易于开发。

2.1.2 深度学习框架Pytorch

Pytorch是由Facebook开源的神经网络框架，基于Torch，专门针对GPU加速的深度神经网络编程。与TensorFlow的静态计算图不同，Pytorch的计算图是动态的，可以更加实时的更改计算图，使得调试网络更加简单。Pytorch不仅支持动态神经网络，而且实现了强大的GPU加速计算，同时还提供了自动求导功能。Pytorch在源码的设计方面，避免重复造轮子，不同于TensorFlow中大量的全新概念，Pytorch追求最少的封装；并且在设计中张量、变量和神经网络这三个由低到高的抽象层次之间紧密结合，可以同时进行修改和操作。由于Pytorch这种简洁明了的设计方式，其源码的可读性十分高，并且其运行速度也比TensorFlow和Keras等框架快很多。

Pytorch在使用时，其内部的API也十分易于使用。构建神经网络时可以通过神经网络模块torch.nn中的Sequential方法来构建一层一层序列化的网络结构模型；然后通过神经网络优化器模块torch.Optim中的不同的方法来构建不同的优化器，以此来进行训练；再通过神经网络模块torch.nn中的损失函数来计算误差值；最后通过调用变量的.backward()方法计算梯度、反向传播，计算参数更新值，再调用优化器的.step()方法将计算得到的新参数更新到网络模型的中。这样就通过Pytorch的API实现了一个完整的简单训练模型结构，非常适合新手刚开始使用。

2.1.3 GPU加速运算库CuPy

在一般的数据运算时，通过Python软件包Numpy，可以将数据向量化然后通过CPU并行运算，但是一般来说CPU内核数为8个甚至更少，所以这也就限制了Numpy库的加速效果。而对于CuPy而言，可以将其理解为一个借助GPU并行运算的Numpy库，因为GPU自身的特性是具有多个CUDA核心，这样一来CuPy就可以实现更好的并行运算加速效果。

同时CuPy与Numpy、Pytorch之间的数据格式的转换也简单。CuPy与Numpy之间的转换可以直接通过CuPy库自带的方法来进行实现，而CuPy与Pytorch之间则需要通过中间库dlpack来进行实现转换，具体代码如下：

CuPy的ndarray类型与Numpy的ndarray类型的相互转换:

|  |
| --- |
| import cupy as cp  # cupy -> numpy  numpy\_data = cp.asnumpy(cupy\_data)  # numpy -> cupy  cupy\_data = cp.asarray(numpy\_data) |

CuPy的ndarray类型与Pytorch的tensor类型的相互转换：

|  |
| --- |
| from cupy.core.dlpack import toDlpack, fromDlpack  from torch.utils.dlpack import to\_dlpack, from\_dlpack  # cupy -> tensor  tensor\_data = from\_dlpack(toDlpack(cupy\_data))  # tensor -> cupy  cupy\_data = fromDlpack(to\_dlpack(tensor\_data)) |

2.2 开发环境搭建

2.2.1 实验环境介绍

1. 软件环境：
   1. 操作系统：Ubuntu 18.04.4 LTS
   2. 开发平台：Anaconda 4.7.10、VScode
   3. 开发语言：Python 3.7.7
   4. 运算平台：CUDA 10.1
   5. 深度学习加速库：CuDNN 7.6.5
   6. 深度学习网络框架：Pytorch 1.4.0
2. 硬件环境：
   1. CPU：Intel(R) Xeon(R) CPU E5-2650 v4 @ 2.20GHz
   2. GPU：GeForce RTX 2080 Ti
   3. 内存：125GB

2.2.2 实验环境搭建

2.3 数据集介绍与处理

2.3.1 数据集介绍

2.3.2 数据集处理

2.4 本章小节

1. 全卷积单阶段目标检测网络的实现

3.1 数据增强及预处理

3.2 骨干网络搭建

3.3 特征图金字塔网络与全卷积单阶段网络检测头搭建

3.4 训练模型、评估模型、结果分析与检测实物功能实现

3.5 本章小节

1. 基于全卷积单阶段目标检测网络的实验

4.1 CuPy库加速运算

4.2 VOC0712数据集

4.3 COCO2017数据集

4.4 实物检测

4.5 本章小节

结论

**参考文献**

致谢