分类号 T391.41 密级 公开
UDC 004.93 学位论文编号 D-10617-308-(2020)-3054

重庆邮电大学硕士学位论文

中文题目	基于 SS-YOLO 算法的管廊巡检机器人视觉检测技术研究与实现
英文题目	Research and Implementation of Visual Detection Technology for Pipeline Inspection Robot based on SS-YOLO Algorithm
学　　号	S170301054
姓　　名	吕灿
学位类别	工学硕士
学科专业	控制科学与工程
指导教师	李勇　副教授
完成日期	2020 年 6 月 30 日

独 创 性 声 明

本人声明所呈交的学位论文是本人在导师指导下进行的研究工作及取得的研究成果。尽我所知，除了文中特别加以标注和致谢的地方外，论文中不包含他人已经发表或撰写过的研究成果，也不包含为获得 重庆邮电大学 或其他单位的学位或证书而使用过的材料。与我一同工作的人员对本文研究做出的贡献均已在论文中作了明确的说明并致以谢意。

作者签名：吕凯山　　日期：2020年6月30日

学位论文版权使用授权书

本人完全了解 重庆邮电大学 有权保留、使用学位论文纸质版和电子版的规定，即学校有权向国家有关部门或机构送交论文，允许论文被查阅和借阅等。本人授权 重庆邮电大学 可以公布本学位论文的全部或部分内容，可编入有关数据库或信息系统进行检索、分析或评价，可以采用影印、缩印、扫描或拷贝等复制手段保存、汇编本学位论文。

（注：保密的学位论文在解密后适用本授权书。）

作者签名：吕凯山　　导师签名：李勇

日期：2020年6月30日　　日期：2020年6月30日

摘要

随着城市化的快速发展，城市综合管廊的建设开始陆续推进。城市综合管廊的安全与人民群众的生命安全和财产安全息息相关，因此城市综合管廊的缺陷检测至关重要。传统的检测需要人工观察机器人传回的实时视频，检测的成本较高同时无法保证检测的准确率。目标检测算法可以实时地检测视频中管廊常出现的缺陷，不需要人工观察，同时检测的精度较高。因此，研究目标检测算法的改进以及将目标检测算法用在管廊的缺陷检测具有重要意义。

管廊巡检机器人的计算能力相对于台式机图形处理器较低。针对目标检测算法在检测速度达到要求时无法保证检测精度的问题，本文对 YOLOv3-Tiny 进行改进，提出一种新的目标检测算法用于城市综合管廊的检测，主要研究内容如下：

针对 YOLOv3-Tiny 检测精度较低的问题，本文改进了其特征提取网络。在 ShuffleNet V2 网络结构的基础上，根据特征图中每个通道的重要性得到每个通道的权重，权重与特征图中每一个通道相乘，实现在通道维度上对特征图进行重标定。通过上述改进得到 Shuffle_Senet 模块，基于此模块创建了特征提取网络。本文使用 kmeans++算法获取先验框，在初始化聚类中心时会考虑距离的因素，避免了可能会陷入局部最优的问题。通过上述对 YOLOv3-Tiny 改进，提出了一种新的算法 SS-YOLO，用来检测城市综合管廊中的缺陷。

针对数据集较少无法满足目标检测算法训练需要的问题，本文利用深度卷积生成对抗网络模型生成缺陷样本增加数据集。本文从数据集中提取部分数据训练深度卷积生成对抗网络模型，训练后生成与原始数据集的特征相同的数据集，从而丰富目标检测算法的训练集，提高检测的精度。

由实验结果可知，本文改进后得到的 SS-YOLO 每秒传输帧数(Frames Per Second,FPS)比 YOLOv3-Tiny 高出 4 帧，平均精度均值 (mean average precision,MAP)高出 6.5%，检测的速度和精度都超过改进之前的算法。在城市综合管廊的检测中，本文采用的 SS-YOLO 算法检测的平均精度均值达到了 0.895，相比 YOLOv3-Tiny 的 0.735，有了明显的提高。由实验结果表明，本文提出的 SS-YOLO 可以满足城市综合管廊缺陷检测的需要。

关键词：目标检测，YOLOv3-Tiny，Shuffle_Senet 模块，SS-YOLO，深度卷积生成对抗网络

Abstract

With the fast development of urbanization, urban underground pipe gallery has been well developed. The defects of urban underground pipe gallery are closely related to the safety of people's lives and property, so the defect detection of urban underground pipe gallery is very important. Traditional detection requires manual observation of the real-time video returned by the robot, the cost of detection is high and the accuracy of detection cannot be guaranteed. The object detection algorithm can detect the defects in the pipe gallery in real time without manual observation, and the detection accuracy is high. This thesis, by improving the feature extraction network and the method of obtaining a prior box, a new object detection algorithm is proposed for the detection of urban underground pipe gallery.

The computing power of the pipeline inspection robot is lower than the computer's graphics processing unit. In order to solve the problem that the object detection algorithm cannot guarantee the detection accuracy when the detection speed meets the requirements, this thesis proposes a new object detection algorithm for the detection of urban underground pipe gallery after the improvement of YOLOv3-Tiny. The main research contents are as follows:

In order to solve the problem that the low detection accuracy of YOLOv3-Tiny, this thesis improves its feature extraction network. Based on the ShuffleNet V2 network structure, the weight of each channel is obtained according to the importance of each channel in the feature map, the feature map is recalibrated in the channel dimension by multiplying the weight with each channel in the feature map. Through the above improvements, the Shuffle_Senet module is obtained. Based on this module, the feature extraction network was created. In this thesis, the kmeans++ algorithm is used to obtain the prior box, and the distance will be considered when initializing the clustering center to avoid the problem of local optimization. Through the above improvements to YOLOv3-Tiny, a new algorithm SS-YOLO is proposed to detect defects in the urban underground pipe gallery.

In order to solve the problem that few data set can't meet the needs of training the object detection algorithm, this thesis uses deep convolutional generative adversarial networks model to generate the defect samples to increase the data set. This thesis extracts

part of data from the data set to train the deep convolutional generative adversarial networks model, and after training generates data set with the same characteristics as the original data set, enriching the training set of the object detection algorithm to improve the accuracy of detection.

According to the experimental results, the improved SS-YOLO's frames per second (FPS) is 4 frames higher than YOLOv3-Tiny, and the mean average precision (MAP) is 6.5% higher, the speed and accuracy of the detection are both higher than the previous algorithm. In the detection of urban underground pipe gallery, the mean average precision of SS-YOLO algorithm detection in this thesis has reached 0.895, which is improved compared to 0.735 of YOLOv3-Tiny. It can be seen from the experimental results that the SS-YOLO proposed in this thesis can meet the needs of defect detection of urban underground pipe gallery.

Keywords: object detection, YOLOv3-Tiny, Shuffle_Senet module, SS-YOLO, deep convolutional generative adversarial networks

目录

图录

表录

注释表

BIM	Building Information Model，建筑信息模型
HOG	Histogram of Oriented Gradient，方向梯度直方图
SVM	Support Vector Machine，支持向量机
SIFT	Scale-Invariant Feature Transform，尺度不变特征变换
RPN	Region Proposal Network，区域生成网络
IOU	Intersection over Union，交并比
Relu	Rectified Linear Unit，修正线性单元
Leaky Relu	Leaky Rectified linear unit，带泄露修正线性单元
AP	Average Precision，平均精度
MAP	Mean Average Precision，平均精度均值
DCGAN	Deep Convolutional Generative Adversarial Networks，深度卷积生成对抗网络

第 1 章 绪论

1.1 选题背景与研究意义

城市综合管廊，全称为地下城市管道综合走廊，即在城市的地下建造管道，将城市的通信、电力、燃气、排水、供热等各种管线集于一体[1]。在管道内部设有巡检人员的出入口，现代化智能化监控管理系统，从而实现各种管线的统一规划、建设和管理。

随着我国经济的快速发展和城镇化的快速发展，城市的建筑密度越来越大，各种管线由于缺少统一的管理从而导致诸多的问题[2]。

1. 管线由于缺少统一的管理，地上地下的各种管线的分布错综复杂，也影响城市市容。

2. 现有的部分管线直接埋在地下，从而导致需要检修的时候经常会出现重复的开挖，不利于管线的故障检修。

3. 管线由于埋在地下会受到土壤等各种物质的腐蚀，同时土壤和上面物体的压力也对管线有一定的伤害从而减少管线的寿命。

城市综合管廊在地下管道当中统一规划、建设和管理各种管线，从而没有上述各种问题的困扰，同时有诸多优势[3]。深埋地下的城市综合管廊让管线免遭诸如台风等自然灾害的伤害，从而让城市可以在恶劣的自然条件之下正常地运行。城市综合管廊深埋地下，在统一规划和管理之下可以节省更多空间。城市综合管廊在空间利用、管理、检修、安全等方面拥有无可比拟的优势，我国许多城市开始大力推进城市综合管廊的建设。

城市综合管廊虽然有众多优势，但是其也可能会受到各种外界的影响从而对管线甚至路面等形成伤害。我国东北、西北、华中和华东部分地区广泛存在的湿陷性黄土，其浸水后在上覆土层自重应力作用下，或者在自重应力和附加应力共同作用下，土的结构破坏会发生显著附加变形，从而管廊容易发生开裂破损等伤害[4]。地震、超量开采地下水引起地面沉降、矿山采空区落顶或岩溶塌陷等原因引起的土层中产生裂隙或断层，这些裂隙或断层导致房屋开裂、道路破坏、管廊破裂等问题。

众多管廊完成修建之后，上述各种问题对管线和人们的生命安全威胁与日俱增，所以管廊的维护和管理变得更加重要。综合管廊中存在着各种管线，如果管廊出现裂缝等导致管线损坏，很可能会影响到其他管线的正常使用。传统的管线巡检要依靠人工巡检，这种效率很低，同时巡检工人的安全无法得到保障。近年兴起的巡检机器人能有效地保护巡检工人的安全，工人通过远程视频对管线进行检查，但是这种依靠人工的检测效率很低，同时检测的精度也无法保证[5]。

研究一种视觉检测算法对管廊的视频实时检测，机器人不间断运行可以大幅度地提高巡检的检查效率、保证了准确率，保障了巡检工人的生命安全和降低了巡检的工作强度。综上所述研究出用于城市综合管廊的视觉检测算法有很强的实用意义和价值。

1.2 国内外研究现状

1.2.1 管廊检测技术研究现状

城市综合管廊位于地下，无法正常接收地面上的信号，如果使用传统的巡检方式，数据传输比较麻烦需要铺设大量的线缆，所以不适用于城市综合管廊。城市综合管廊如果发生安全问题，因此产生的后果是不可估量的，国内外为了解决管廊的运维管理中出现的问题，提出了各种解决方案。以物联网技术为基础的管廊综合监控系统，由环境与设备监控系统、安全防范系统、预警与报警系统、地理信息系统等部分组成，从而实现对地下管线的实时监测，其中包括对城市综合管廊中积水的监测、有毒有害气体泄漏的监测、管线的实时运行情况监测等[6]。

常用的还有利用建筑信息模型(Building Information Model,BIM)提高城市综合管廊运营和维护等的效率[7]。BIM 技术最开始在 20 世纪 70 年代被提出，但是由于当时计算设备的计算能力比较弱，所以无法达到较好的发展。进入 21 世纪之后，随着计算设备的增强和各种理论研究的发布，BIM 技术得到了较好的发展，基于 BIM 技术的建筑信息集成管理系统广泛在美国的建筑行业中使用[8]。国内的建筑行业在 2000 年左右开始引进 BIM 技术，后面随着各个部门的推动，BIM 技术在国内有了较好的发展。2015 年，国内的 BIM 技术的未来的发展目标由住建部提出。随着各个政策和需求的推动，BIM 技术广泛应用于国内的实际工程当中。

随着物联网、传感器等相关技术的发展，新提出了很多基于物联网等技术的监测方案，如基于磁感应的无线传感器网络，可实现低成本的实时泄漏检测和定位[9]。随着城市化进程的快速稳定发展，城市综合管廊大量建设的同时，管廊的检测和管理系统也在飞速发展，国内的管廊管理系统与国外完善的综合管廊管理系统仍有一定的差距。目前的国内外管廊检测的重点基本都是在出现泄漏之后，通过检测泄漏的成分来进行检测，无法实现预防等功能。

1.2.2 视觉检测研究现状

传统的目标检测算法基于机器学习实现，首先获取候选区域之后，使用特征描述子提取特征。常用的特征描述子有方向梯度直方图(Histogram of Oriented Gradient, HOG)，尺度不变特征变换(Scale-invariant feature transform,SIFT)等[10,11]。提取到特征之后使用分类器识别，如 1964 年提出的支持向量机(Support Vector Machine, SVM)[12]。

海量数据集的累积和计算设备性能的快速发展，让深度学习有了充足的数据可以提高性能，计算设备让训练时间大幅度地减少，从而让深度学习得到了广泛地关注。2012 年 Hinton 团队提出基于卷积神经网络的 AlexNet，在 ImageNet 比赛中以远超第二名的好成绩获得了第一名[13]。Hinton 团队的成功显示出了深度学习相对于传统机器学习的巨大的优势，此后深度学习广泛应用于目标检测、语义分割等各个领域。

基于深度卷积神经网络的目标检测算法主要有两类，一类是首先获取候选区域然后进行目标检测的两阶段的目标检测算法，另一类是直接进行目标检测的单阶段的目标检测算法。2014 年提出的基于深度卷积神经网络的两阶段的目标检测算法 R-CNN[14]。R-CNN 中的输入尺寸是固定的，所以影响了模型的性能，2014 年何凯明等人提出了 SPPNet(Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition)，使用空间金字塔池化层解决了输入尺寸是固定的问题[15]。2015 年对 R-CNN 进行了改进，提出了 Fast R-CNN[16]。2015 年任少卿等人对网络进行了改进提出了 Faster R-CNN，使用区域生成网络生成候选区域，大幅度地提高了模型的速度和精度[17]。

2015 年提出基于深度卷积神经网络的单阶段的目标检测算法 YOLO(You Only Look Once)，YOLO 的检测速度远远高于两阶段的目标检测算法，但其对于小目标

地检测精度较低[18]。之后提出的 SSD(Single Shot MultiBox Detector)模型相对于 YOLO 更好地平衡了检测精度和速度[19]。2018 年在原有网络的基础上增加了多尺度预测，改进了特征提取网络等，从而提出了 YOLOv3 算法，在检测的精度和速度上都有了很好的提升[20]。

1.3 本文主要研究内容及创新点

城市综合管廊检测的时候，由于现场环境和安全等问题不适宜人工检测，移动机器人可以替代人去危险和狭窄的空间执行任务。当前使用的移动机器人无法对检测到的数据进行处理，需要将视频等数据传输到控制室，然后使用人工观察的方式进行检测，这样不仅增加了视频传输和人工等各种成本，同时无法保证检测的准确率。目标检测算法可以代替人工进行检测，同时节约了成本，研究目标是视觉检测算法可以在巡检机器人上运行，实现对城市综合管廊的实时检测，发现管廊中水泥裂缝以及墙皮破损等问题。

本文的研究内容主要如下。

1. 改进特征提取网络

目标检测算法中特征提取对网络检测的速度和精度影响较大，巡检机器人检测的时候需要在保证精度的前提下尽可能地提升速度。本文在参考 ShuffleNet V2 网络结构的基础上，借鉴了 SENet(Squeeze and Excitation Networks)网络的思想，通过学习获得不同通道的重要程度，然后根据得到的结果去提升有用的通道抑制不太重要的通道，基于上述思想提出了 Shuffle_Senet 模块，通过多个 Shuffle_Senet 模块的组合即可得到特征提取网络。

2. 目标检测算法的改进与实现

本文借鉴了 YOLOv3-Tiny 的多尺度检测的思想，对模型的特征提取算法以及获取先验框的方法进行了改进，得到了基于 YOLOv3-Tiny 改进的目标检测算法 SS-YOLO，在保证速度的同时提高了检测的精度。采集的样本和使用深度卷积生成对抗网络生成的样本一同构成目标检测算法的数据集，利用数据集训练模型之后，利用测试集测试算法的性能。

3. 基于深度卷积生成对抗网络的样本生成

现实当中通常缺陷的数据集都比较少，如墙皮破损等缺陷采集到的数据集数量都比较稀少，但是基于深度学习的目标检测需要大量的数据集才可以得到较好的性能。本文在采集样本的基础上，使用深度卷积生成对抗网络生成与采集样本特征一致的样本，从而丰富训练集，增强模型的泛化能力。

本文的创新点，有以下几点。

(1) 基于 ShuffleNet V2 网络结构和 SENet 的特征通道重标定的思想提出了新的网络结构 Shuffle_Senet 模块，在保证精度的同时提升了速度。

(2) 改进了目标检测算法的特征提取网络以及获取先验框的方式，提出了一种新的目标检测算法 SS-YOLO。SS-YOLO 在保证检测精度的同时，网络比较简单从而可以在巡检机器人等移动设备上使用。

(3) 使用深度卷积生成对抗网络生成训练样本，从而保证小样本情况下的目标检测的精度。

1.4 论文主要结构

本文共有 6 章，每章的主要内容如下。

第一章是绪论，首先介绍了城市综合管廊检测机器人视觉检测算法的研究背景，然后对国内外管廊检测和视觉检测技术的研究现状进行分析。最后介绍了本文的研究目标和研究内容。

第二章介绍了管廊巡检机器人视觉检测技术的理论基础，主要介绍了传统目标检测算法、基于深度学习的两阶段目标检测算法、基于深度学习的单阶段目标检测算法的原理和主要操作流程。

第三章介绍了目标检测算法的细节，改进了特征提取网络以及获取先验框的方法提出一种新的目标检测算法 SS-YOLO。使用相同的开源数据集训练改进前后的目标检测算法，从而对比改进前后算法的性能的变化。

第四章介绍了基于深度卷积生成对抗网络的样本生成，实际训练过程中缺陷的数据集正常都较少，为了满足训练的需要，使用深度卷积生成对抗网络生成与原始数据集特征相似的样本，增强目标检测算法的性能。

第五章介绍了目标检测算法的验证与测试，使用巡检机器人的计算设备，对测试集中的数据进行测试，从而评价目标检测算法的性能。

第六章是总结与展望，对本文进行总结，并且对当前的不足提出下一步的研究和改进方向。

第 2 章 管廊巡检机器人视觉检测技术的理论基础

传统的视觉检测的准确率比较依靠人工设置的特征提取算法，常用的特征匹配算法有 SIFT（Scale-invariant feature transform，尺度不变特征变换）、SURF（Speeded up robust features，加速稳健特征）、BRISK（Binary robust invariant scalable keypoints，二进制鲁棒不变尺度特征关键点）等[21,22]。基于深度学习的检测算法的精度和泛化能力要远高于传统的视觉检测算法，常用的有基于深度学习的两阶段目标检测算法、基于深度学习的单阶段目标检测算法。

2.1 基于 SIFT 的特征匹配算法

使用 SIFT 进行特征匹配时流程大致如下，首先对管廊中缺陷的图片进行去噪等预处理，其次对图片进行关键点提取，然后对特征点进行特征描述，最后通过对比两个的特征来判断是否相同[23]。

2.1.1 关键点提取和方向确定

首先创建一个高斯金字塔，将原始图片通过方差分别为σ、$k\sigma$、$k^2\sigma$、$k^3\sigma$、$k^4\sigma$、$k^5\sigma$的高斯滤波，得到 6 张图片。其中σ取值为 1.6，k 的取值约为 1.256，图片从左到右越来越模糊。利用 6 张相邻的图片分别相减，即可得到 5 张含有特征的 DOG（difference of gaussian，高斯差异）图像[24]。假设原始图片的尺寸为$2W\times 2H$，则将尺寸为$W\times H$、$\frac{W}{2}\times\frac{H}{2}$的图片分别通过方差值为之前的 2 倍和 4 倍的高斯滤波，分别用同一个尺寸内相邻的图片相减，从而得到另外两种尺寸的 DOG 图像。

在一个尺度的 DOG 图像内，遍历 DOG 图像中的每一个像素点，每一个目标点与当前 DOG 图像包围它的 8 个像素点，以及相邻 DOG 图像的 9 个像素点进行比较。因为每一个 DOG 图像有两个相邻的 DOG 图像，目标点共与 26 个像素点进行比较，如果这个点在 26 个点中为极值点，则称为极值点。然后在另外两个尺度的 DOG 图像中重复上述操作，从而得到另外两个尺度 DOG 图像的极值点。由于上面得到的极值点是离散空间的极值点，使用子像素插值的方法对极值点进行曲线拟合，

从而提高极值点的精确度。由于图像的边缘区域，灰度值突变会形成极值点，但是这些极值点是无效的，利用 Hessian 矩阵可以去除这些极值点，从而解决边缘的影响，得到更加精确的极值点[25]。

首先计算关键点的梯度和领域内像素的梯度和方向，之后利用直方图统计邻域内像素的梯度和方向。梯度直方图按照每 10 度为一个范围将 0~360 度的方向范围分为 36 份，统计其领域像素的梯度和方向之后，直方图的峰值方向代表了关键点的主方向。

2.1.2 特征点描述

首先确定关键点描述所需要的图像区域，将关键点附近的领域划分成4×4个子领域，每一个子领域作为一个种子点各有 8 个方向。将坐标轴根据关键点的方向进行旋转，从而保证旋转不变性。将领域内的采样点分配4×4个子区域中的对应位置，然后将子区域内的梯度值分配到 8 个方向上[26]。利用插值计算每个种子点八个方向的梯度，上述得到的4×4×8个梯度即为关键点的特征向量。为了减少光照等对梯度值的影响，上面得到的梯度值进行归一化处理，然后按照特征点的尺度对特征向量进行排序，排序得到的即为特征点的描述[27]。

2.1.3 特征匹配

两张图片利用上述的特征点描述方法，获得元素为浮点数的特征向量。通常使用欧氏距离计算特征向量之间的距离，如果两个特征向量之间的距离小于某一个阈值，则认为匹配成功，否则认为匹配失败。

2.2 基于深度学习的两阶段目标检测算法

卷积神经网络在物体分类上的应用取得不错效果之后,Girshick R.将卷积神经网络用于目标检测提出来一种新的目标检测算法 RCNN。RCNN 是一种基于深度学习的两阶段目标检测算法，其主要流程有以下几步。首先使用选择性搜索方法将一张图片生成两千多个候选区域，然后使用卷积神经网络提取候选区域的特征。将上面提取的特征输入到训练好的分类器中进行分类，如用 SVM 分类器进行判别，最后

对候选的目标框的位置大小等进行微调从而提高准确性。RCNN 中的选择性搜索生成的候选区域有很多重叠，所以后续的特征提取等操作有大量的冗余，同时分类和位置的回归单独进行耗费了较多的存储空间，针对上述各种问题，改进得到了 Fast-RCNN[28]。用区域生成网络(Region Proposal Network,RPN)取代之前的效率较低的候选框生成方式，Faster-RCNN 的检测速度有了大幅度的提升[29]。Faster-RCNN 整体结构如图 2.1 所示。

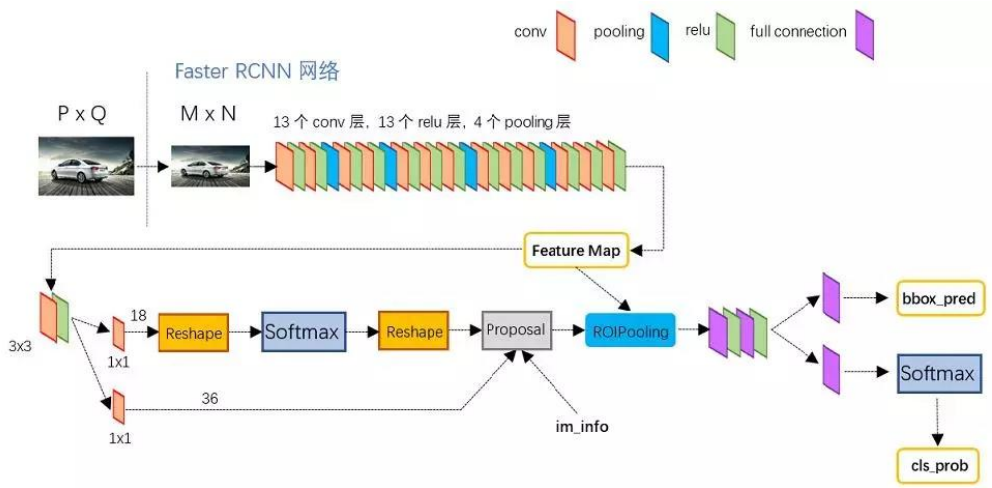


图 2.1 Faster-RCNN 整体结构图

Faster-RCNN 主要由四个部分组成，第一用于提取图片特征的卷积神经网络，第二用于生成候选区域的区域生成网络，第三候选区域池化层，第四全连接层

2.2.1 用于特征提取的卷积神经网络

近年来随着卷积神经网络的发展，在图像分类和目标检测中广泛存在各种卷积神经网络模型。相对于传统的特征提取方法，卷积神经网络拥有更好的泛化能力等好处。常用的网络模型有 AlexNet、VGG16、GoogLeNet 等[30,31,32]。Faster-RCNN 中用于提取特征的卷积神经网络模型由 13 个卷积层，13 个 relu 层和 4 个池化层组成。

2.2.2 区域生成网络

Faster-RCNN 中使用区域生成网络生成候选区域。其原理如图 2.2 所示。

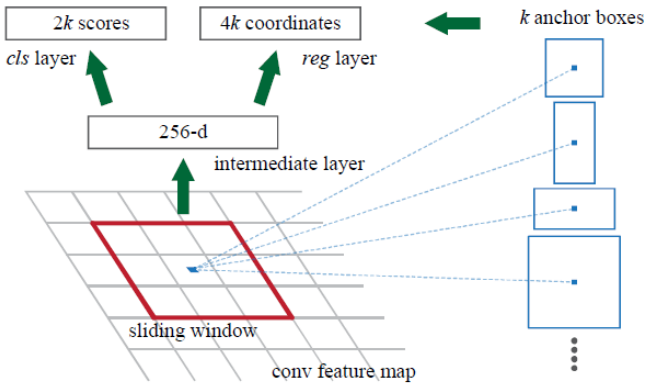


图 2.2 Faster-RCNN 的区域生成网络原理图

Faster-RCNN 中的区域生成网络主要是以 anchor（锚框）为核心生成候选框，Faster-RCNN 使用的是 9 个 anchor。假设原始图片通过卷积神经网络提取到的特征图维度为(H,W,N)，通过一个3×3的滑动窗口遍历整个特征图，每一个窗口会生成 9 个对应原始图片的候选区域，在区域生成网络中使用对特征图进行一次卷积核为3×3的卷积操作完成上述步骤，通过卷积得到一个(H,W,256)的特征图。生成候选区域之后通过分类层和回归层分别对特征图进行操作[33]。

分类层中(H,W,256)的特征图经过卷积核为1×1的卷积，得到的维度为(H,W,18)（18=9×2，Faster-RCNN 中共有 9 个 anchor 即锚框），每一个滑动窗口中有 9 个候选区域，18 为每个候选区域是检测物体和不是检测物体的概率。为了方便 softmax 分类将原始维度通过 reshape（重塑）操作改变为$(2,9\times H,W)$,然后用 softmax 函数对候选区域进行二分类，然后通过重塑操作恢复维度为(18,H,W)。

回归层中(H,W,256)的特征图经过卷积核为1×1的卷积，得到的维度为(H,W,36)（36=9×4，Faster-RCNN 中每个候选框有四个坐标）。每个候选区域有 4 个坐标值，每一个滑动窗口中有 9 个候选区域共有 36 个坐标值。设定好的 anchor 尺寸不可能与目标的尺寸完全一致，所以用得到的(x,y,w,h)对候选区域进行调整，使其更接近原始尺寸，其中 x、y 表示中心点坐标，w、h 表示宽和高。

由上述操作得到每一个候选区域的分类结果和回归坐标之后。首先利用回归层的坐标对候选区域的边框进行调整。根据正样本中的 softmax 分类的分数从大到小进行排序，选择分数靠前的 N 个候选区域。根据设定的阈值，利用非极大值抑制方法删除交并比(Intersection over Union,IOU)超出阈值的候选区域[34]。

2.2.3 感兴趣区域池化层

ROI Pooling 全称为 Region of interest pooling 即感兴趣区域池化，利用池化将不同尺寸的输入转化为固定尺寸[35]。传统的卷积神经网络的输入和输出的尺寸通常需要为固定的统一的尺寸，但是 Faster-RCNN 中的区域生成网络生成的候选区域的尺寸都是不一样的，同时通过卷积神经网络提取的特征图和区域生成网络生成的候选区域的尺寸也是不同的，所以利用感兴趣区域池化层将输入和输出的尺寸统一。

区域生成网络生成的候选区域中的坐标为原始图片中的坐标，所以需要首先将候选区域的坐标映射到特征图大小的尺度。假设原始图片和特征图的比例为 K，将候选区域的坐标全部除以 K 即可完成坐标映射，通过映射后的坐标即可得到候选区域在特征图中映射的位置。假设需要最后输出的尺寸是长为 w、宽为 h，则首先将特征图中映射的位置划分成一个长为 w、宽为 h 的网格，然后在每一个网格中进行 max pooling（最大池化）操作。候选区域经过上述的处理之后，即可以得到尺寸一致的候选区域。

2.2.4 全连接层

通过上述操作特征图中的候选部分，所以需要进一步地计算从而得到候选区域的类别和精确的尺度。Faster-RCNN 中利用全连接和 softmax 来计算候选区域属于的类别。Softmax 计算过程如下所示。

假设需要计算的数组为 T，数组内共有 n 个元素 $\{T_1,T_2,\ldots,T_n\}$，则数组中第 i 个元素 T_i 的 softmax 值计算如公式(2.1)所示[36]。

$$S_i=\frac{e^{T_i}}{\sum_{j=1}^{n}e^{T_j}} \tag{2.1}$$

通过 softmax 计算之后，数组中的元素值会映射成零到一之间的小数，所有元素累加的和为 1，可以近似理解成每个元素的概率。在输出的时候选择 softmax 计算之后最大的元素，即是预测的目标种类。

Faster-RCNN 中使用 bounding box regression（边界框回归）获得每个候选区域的精确边界框[37]。边界框回归中输入的数据为 $\{(P^1,G^1),(P^2,G^2),\ldots,(P^n,G^n)\}$，其中

$G^i=\left(G_x^i,G_y^i,G_w^i,G_h^i\right)$表示第 i 个目标的真实边界框坐标，$P^i=\left(P_x^i,P_y^i,P_w^i,P_h^i\right)$表示第 i 个目标候选的边界框坐标即通过区域生成网络和感兴趣区域池化层得到的候选区域。P_x^i表示候选边界框中心点的 x 坐标，P_y^i表示候选边界框中心点的 y 坐标，P_w^i表示候选边界框的宽度，P_h^i表示候选边界框的高度，真实边界框坐标的含义与上述相同。边界框回归通过找到一种映射，让候选的边界框 P 通过映射得到一个更接近真实边界框 G 的预测边界框$\hat{G}$,三者关系如图 2.3 所示，图中红色的框为候选边界框，蓝色的框为候选边界框经过映射之后得到的预测边界框，绿色的框为真实目标的边界框，红色、蓝色、绿色圆点分别为候选边界框、预测边界框、真实边界框的中心点。

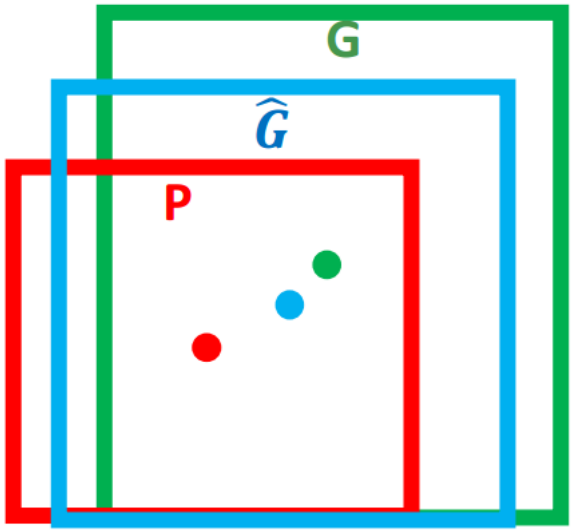


图 2.3 候选边界框、预测边界框、真实边界框示意图

边界框回归通过找到一个映射 f，让映射后的边界框更加接近真实的边界框，即如公式(2.2)所示。Faster-RCNN 中使用的是先平移然后进行尺度缩放的方式进行映射[38]。

$$\mathrm{f}\left(P_x^i,P_y^i,P_w^i,P_h^i\right)=\left(\hat{G}_x^i,\hat{G}_y^i,\hat{G}_w^i,\hat{G}_h^i\right)\approx\left(G_x^i,G_y^i,G_w^i,G_h^i\right) \tag{2.2}$$

首先对原始坐标进行平移，计算如公式(2.3)、公式(2.4)所示。

$$\hat{G}_x=P_w d_x\left(P\right)+P_x \tag{2.3}$$

$$\hat{G}_y=P_h d_y\left(P\right)+P_y \tag{2.4}$$

然后对原始坐标进行尺度缩放，计算如公式(2.5)、公式(2.6)所示。

$$\hat{G}_w=P_w e^{d_w(P)} \tag{2.5}$$

$$\hat{G}_h = P_h e^{d_h(P)} \tag{2.6}$$

由于对原始坐标进行边界框回归之后得到的是预测的边界框不是真实的边界框，真实需要的坐标平移(t_x, t_y)和缩放尺度(t_w, t_h)计算如公式(2.7)、公式(2.8)、公式(2.9)、公式(2.10)所示。

$$t_x = \frac{G_x - P_x}{P_w} \tag{2.7}$$

$$t_y = \frac{G_y - P_y}{P_h} \tag{2.8}$$

$$t_w = \log \frac{G_w}{P_w} \tag{2.9}$$

$$t_h = \log \frac{G_h}{P_h} \tag{2.10}$$

边界框回归中需要的值$d_*(P)$(*代表 x,y,w,h)，可以通过对特征向量的学习得到。计算如公式(2.11)所示，其中$\phi_5(P)$是候选区域的特征向量，w_*^T 是要学习的参数(*代表 x,y,w,h)。

$$d_*(P) = w_*^T \phi_5(P) \tag{2.11}$$

通过参数的学习，让预测的边界框和真实的边界框的差距最小，即求解最优的w_*，计算如公式(2.12)所示。

$$w_* = \arg\min_{\hat{w}_*} \sum_{i=1}^{n} \left(t_*^i - \hat{w}_*^T \phi_5(P^i) \right)^2 + \lambda \left\| \hat{w}_* \right\|^2 \tag{2.12}$$

2.2.5 损失函数

区域生成网络中锚框被分成正样本和负样本，如果满足下列条件中的任意一个即为正样本。第一，与真实边界框的交并比(Intersection over Union,IOU)最高的锚框为正样本。第二，锚框与真实的边界框的交并比超过 0.7。锚框与真实的边界框的交并比小于 0.3 的即为负样本。对于不是正样本也不是负样本的锚框，因为对训练没有帮助所以不计算损失。Faster-RCNN 的损失函数如公式(2.13)所示[39]。

$$L(\{p_i\},\{t_i\}) = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*) + \lambda \frac{1}{N_{reg}} \sum_i p_i^* L_{reg}(t_i, t_i^*) \tag{2.13}$$

上式中 i 表示这一批次训练中的第 i 个锚框，p_i为第 i 个锚框中有目标的概率，t_i是预测边界框的 4 个坐标相关的向量。L_{cls}为锚框正样本和负样本分类的损失，L_{reg}表示坐标回归的损失，坐标回归损失乘以 p_i^*表示只有正样本的时候才会计算坐标损失，N_{cls}和N_{reg}分别用来对分类损失和回归损失进行归一化操作。N_{cls}取值为训练时候一个批次的样本的数量，N_{reg}取值为锚框坐标相关参数的数量，由于N_{cls}和N_{reg}的值差值较大，使用λ平衡两者的权重，从而让两部分损失的权重大致相等。p_i^*表示真实边界框是否为正样本，正样本p_i^*等于 1，否则等于 0。t_i^*是正样本边界框的坐标。

Faster-RCNN 中分类损失使用的是对数损失函数，$L_{cls}\left(p_i, p_i^*\right)$计算如公式(2.14)所示。

$$L_{cls}\left(p_i, p_i^*\right) = -\log\left[p_i^* p_i + \left(1-p_i^*\right)\left(1-p_i\right)\right] \tag{2.14}$$

Faster-RCNN 中回归损失用的是 Smooth L1 损失函数，Smooth L1 损失函数如公式(2.15)所示。

$$Smooth_{L1}\left(x\right) = \begin{cases} 0.5x^2 & |x|<1 \\ |x|-0.5 & |x|\geq 1 \end{cases} \tag{2.15}$$

坐标回归的损失L_{reg}计算如公式(2.16)所示,其中 R 表示 Smooth L1 损失函数。

$$L_{reg}\left(t_i, t_i^*\right) = R\left(t_i - t_i^*\right) \tag{2.16}$$

上式中的t_i和t_i^*分别表示预测边界框和真实边界框的坐标相关数据，其中t_i包括t_x、t_y、t_w、t_h分别为预测边界框中心坐标以及宽度和高度，t_i^*包括t_x^*、t_y^*、t_w^*、t_h^*分别为真实边界框中心坐标以及宽度和高度。

2.3 基于深度学习的单阶段目标检测算法

基于深度学习的单阶段目标检测算法不需要利用生成好的候选区域来计算目标的坐标和分类结果，可以直接得到目标的坐标和分类结果，极大的提高了检测的速度让实时目标检测成为了可能。常用的基于深度学习的单阶段目标检测算法有 SSD、YOLO 等目标检测算法，下面以 YOLO 为代表介绍单阶段目标检测算法。

2.3.1 YOLOv1 目标检测整体思路

YOLOv1 目标检测算法全称为 You Only Look Once: Unified, Real-Time Object Detection 由 Joseph Redmon 等人提出，是一种不需要产生候选区域可以直接生成目标的坐标和分类结果的目标检测算法。YOLOv1 在检测一张图片的时候，首先将一张图片分成 $N \times N$ 的网格，假设物体 O 的中心点落在其中一个单元格 L 中，则这个单元格 L 就负责检测物体 O[40]。对于划分之后的每一个单元格，YOLOv1 会预测 B 个边界框(bounding box)，同时预测出每个边界框的位置 (x, y, w, h) 和置信度(confidence)。边界框坐标中 (x, y) 表示边界框中心点相对于单元格左上角的坐标，w 为边界框的宽度，h 为边界框的高度。边界框的置信度为边界框内是否含有物体的置信度，用来判断边界框内是否有物体，以及对物体的位置的预测是否准确，置信度计算如公式(2.17)所示。

$$Confidence = \Pr(object) * IOU \frac{Truth}{Pred} \tag{2.17}$$

上式中 Confidence 表示边界框的置信度。$\Pr(object)$ 表示边界框内是否有物体，如果边界框内有物体则 $\Pr(object)$ 等于 1，如果边界框内没有物体则 $\Pr(object)$ 等于 0。$IOU \frac{Truth}{Pred}$ 表示真实边界框和预测边界框的交并比，真实边界框和预测边界框的面积的交集和两个的面积的并集的比值。

YOLOv1 检测网格中是否有物体的同时也计算该物体属于每个类别的置信度 $\Pr(class_i \mid object)$，一个单元格共有 B 个边界框但是只计算一组概率值。每个单元格中计算的每个类别的置信度，用来判断单元格内的目标属于每一个类别的可能性，计算如公式(2.18)所示。

$$\Pr(Class_i \mid Object) * \Pr(Object) * IOU \frac{Truth}{Pred} = \Pr(Class_i) * IOU \frac{Truth}{Pred} \tag{2.18}$$

综上所述，即通过将一个图片分成 $N \times N$ 个单元格，然后在每个单元格中预测两组边界框的位置和置信度，同时每个单元格中计算目标属于每个类别的置信度。假设 YOLOv1 的每个单元格中有 B 组边界框，每个边界框会有位置 (x, y, w, h) 和置

信度(confidence)五个数据，若检测目标共有 N 个类别，则最后通过图片共预测到 $N\times N\times\left[5\times B+N\right]$ 组数据。

2.3.2 YOLOv1 原理

YOLOv1 在接收输入的图片之后，会首先将尺寸不一的图片进行大小的调整，如将所有图片的尺寸都调整为 448×448。YOLOv1 中使用卷积神经网络进行特征提取，使用全连接层预测概率和坐标，网络结构如图 2.4 所示。

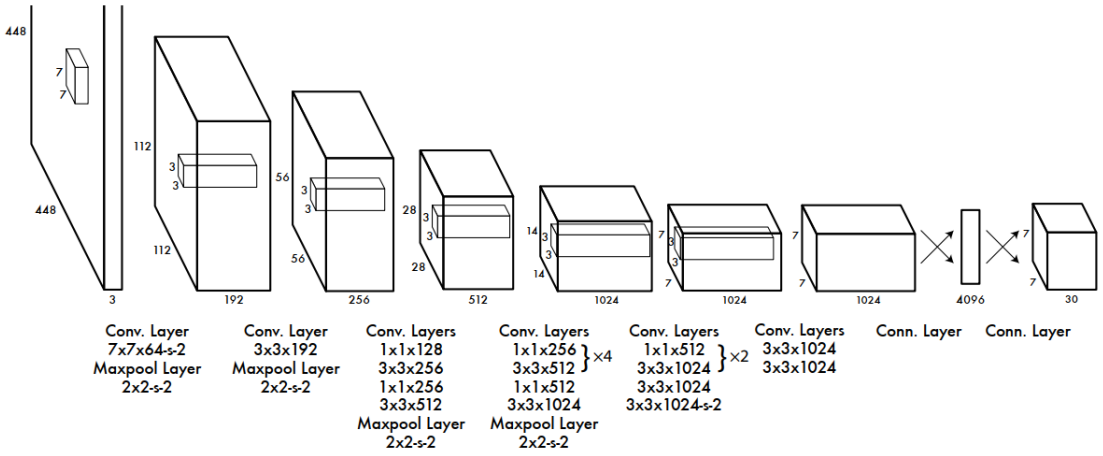


图 2.4 YOLOv1 网络结构图

YOLOv1 的网络模型是参考 GoogLeNet 设计的，前面有 24 个卷积层提取特征，后面是两个全连接层最后一层全连接层用于预测概率和坐标。卷积层中主要使用的是卷积核大小为1×1和3×3的卷积层，使用卷积核大小为1×1的卷积层对特征的尺寸进行降维，然后使用卷积核大小为3×3的卷积层提取特征。YOLOv1 中全连接层和卷积层使用的 Leaky Relu 作为激活函数，最后一个全连接层使用的是线性激活函数。YOLOv1 可以检测 20 个类别，然后每个单元格有 2 个边界框，所以每个单元格的输出个数为5×2+20共 30 组数据，整张图片共有7×7×30组数据。

YOLOv1 的损失函数由边界框坐标的损失函数、边界框宽高的损失函数、置信度的损失函数、类别的损失函数几部分组成。模型中主要有以下几个参数，原始图片共分成 $N\times N=N^2$ 份，每个单元格内有 B 个边界框。在损失函数中添加 λ_{coord} 和 λ_{noobj} 等参数用来增加边界框坐标的损失值和减小置信度的损失值，从而解决某些没有物

体的单元格置信度接近 0 以及梯度过高导致的模型不稳定等问题。边界框坐标的损失函数如公式(2.19)所示。

$$\lambda_{coord}\sum_{i=0}^{N^2}\sum_{j=0}^{B}\ell_{ij}^{obj}\left[\left(x_i-\hat{x}_i\right)^2+\left(y_i-\hat{y}_i\right)^2\right] \tag{2.19}$$

上式中的 x_i 表示第 i 个单元格的预测的边界框的相对于单元格左上角的 x 坐标，$\hat{x}_i$ 表示第 i 个单元格的真实的边界框的相对于单元格左上角的 x 坐标，y_i 和 $\hat{y}_i$ 分别表示 i 个单元格的预测的和真实的边界框的 y 坐标。ℓ_{ij}^{obj} 表示第 i 个单元格中第 j 个边界框负责预测这个目标，因为第 i 个单元格中第 j 个边界框与真实的边界框的 IOU(Intersection over Union,交并比)较大。

实际检测中的物体尺寸可能差距较大，较大物体有一个相对于自身尺寸较小的偏差 A 时，此时的损失函数应该要小于较小物体有一个相对于自身尺寸较大的偏差 A，所以宽和高的损失函数中使用的是宽和高的平方根。损失函数如公式(2.20)所示，公式中的 w_i 为预测的边界框的宽度，h_i 为预测的边界框的高度，$\hat{w}_i$ 为真实的边界框的宽度，$\hat{h}_i$ 为真实的边界框的高度。

$$\lambda_{coord}\sum_{i=0}^{N^2}\sum_{j=0}^{B}\ell_{ij}^{obj}\left[\left(\sqrt{w_i}-\sqrt{\hat{w}_i}\right)^2+\left(\sqrt{h_i}-\sqrt{\hat{h}_i}\right)^2\right] \tag{2.20}$$

YOLOv1 的置信度的损失函数如公式(2.21)所示，置信度的损失函数是由含有目标的边界框的置信度的损失函数和没有含有目标的边界框的置信度的损失函数两部分组成。

$$\sum_{i=0}^{N^2}\sum_{j=0}^{B}\ell_{ij}^{obj}\left(C_i-\hat{C}_i\right)^2+\lambda_{nobj}\sum_{i=0}^{N^2}\sum_{j=0}^{B}\ell_{ij}^{nobj}\left(C_i-\hat{C}_i\right)^2 \tag{2.21}$$

上式中 ℓ_{ij}^{obj} 表示第 i 个单元格中的第 j 个边界框中有目标，ℓ_{ij}^{nobj} 表示第 i 个单元格中的第 j 个边界框中没有目标。C_i 表示第 i 个单元格的预测的置信度，$\hat{C}_i$ 表示第 i 个单元格的真实的置信度。

YOLOv1 的分类的损失函数如公式(2.22)所示，$p_i(c)$ 表示预测的类别 c 的概率，$\hat{p}_i(c)$ 表示真实的类别 c 的概率。

$$\sum_{i=0}^{N^2}\ell_i^{obj}\sum_{c\in classes}\left(p_i(c)-\hat{p}_i(c)\right)^2 \tag{2.22}$$

综上所述，YOLOv1 的损失函数即为上述几种损失函数的和，损失函数如公式(2.23)所示。

$$\begin{aligned} &LOSS \\ &=\lambda_{coord}\sum_{i=0}^{N^2}\sum_{j=0}^{B}\ell_{ij}^{obj}\left[\left(x_i-\hat{x}_i\right)^2+\left(y_i-\hat{y}_i\right)^2\right] \\ &+\lambda_{coord}\sum_{i=0}^{N^2}\sum_{j=0}^{B}\ell_{ij}^{obj}\left[\left(\sqrt{w_i}-\sqrt{\hat{w}_i}\right)^2+\left(\sqrt{h_i}-\sqrt{\hat{h}_i}\right)^2\right] \\ &+\sum_{i=0}^{N^2}\sum_{j=0}^{B}\ell_{ij}^{obj}\left(C_i-\hat{C}_i\right)^2+\lambda_{nobj}\sum_{i=0}^{N^2}\sum_{j=0}^{B}\ell_{ij}^{nobj}\left(C_i-\hat{C}_i\right)^2 \\ &+\sum_{i=0}^{N^2}\ell_i^{obj}\sum_{c\in classes}\left(p_i(c)-\hat{p}_i(c)\right)^2 \end{aligned} \tag{2.23}$$

2.3.3 YOLOv2 原理

YOLOv1 由于模型自身的缺陷所以存在一些缺点。YOLOv1 中每个单元格只有两个边界框，而且只能属于同一个类别，所以 YOLOv1 检测小物体和两个位置比较近的物体的能力比较弱。YOLOv1 对于一种物体出现新的宽高比时的泛化能力不足。YOLOv2 在 YOLOv1 的基础上进行改进，提高了检测的速度，增强了预测的准确性，以及可以识别更多的种类[42]。

YOLOv2 在 YOLOv1 的基础上有了以下改进。

1. 在卷积层之后添加了批量归一化层

YOLOv2 在所有的卷积层后面添加了 Batch Normalization（批量归一化）层[43]。假设一个批次输入的样本为$\{x_1,x_2,x_3,\ldots,x_n\}$，则可求得其均值$\mu_B$如公式(2.24)所示。

$$\mu_B=\frac{1}{n}\sum_{i=1}^{n}x_i \tag{2.24}$$

根据方差的计算公式由上面求得的均值可以得到这个批次样本的方差σ_B^2，如公式(2.25)所示。

$$\sigma_B^2=\frac{1}{n}\sum_{i=1}^{n}\left(x_i-\mu_B\right)^2 \tag{2.25}$$

通过上面求得的均值和方差，可以对这个批次的每一个样本进行归一化处理，从而让样本符合标准正态分布，即均值为 0，方差为 1，如公式(2.26)所示，ε是一个大于 0 同时值很小的常数用来确保除数大于 0。

$$\hat{x}_i = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \varepsilon}} \tag{2.26}$$

最后对归一化处理后得到的样本进行缩放和偏移等操作，即可得到批量归一化的输出结果 y_i,如公式(2.27)所示，其中 γ 为缩放的系数， β 为偏移量。

$$y_i = \gamma \hat{x}_i + \beta \tag{2.27}$$

YOLOv2 通过添加批量归一化层，解决了模型的过拟合问题，也有利于解决网络层数太多时反向传播时的梯度消失和梯度爆炸等问题，同时提升了模型收敛的速度和效果。YOLOv2 的 MAP（mean average precision, 平均精度均值）由于添加了批量归一化层有了 2%的提升。

2. 使用锚框(Anchor Boxes)预测边界框

YOLOv1 中使用全连接层预测边界框，同时一个单元格中只有 2 个边界框，而且每个单元格只能预测一组每个种类的概率，所以每个单元格只能检测一种物体，同时对于物体在不同宽高比的情况下的泛化能力不够。YOLOv2 中每个锚框都单独的预测一组每个种类的概率，所以如果每个单元格内共有 K 个锚框，则每个单元格可以预测 K 个物体。YOLOv2 中的锚框是借鉴了 Faster-RCNN 中的锚框的思想，然后使用 K-Means 算法利用训练集中的边界框生成的，从而有更好的泛化能力[44]。

K-Means 算法生成锚框的流程主要为以下几步。假设共需要 K 个锚框，第一步从训练集的边界框中随机选 K 个作为初始的簇的中心点。第二步计算每一个样本与每一个中心点的距离，然后将每一个样本划分到距离最近的那个中心点的簇当中。第三步计算每一个簇的均值，然后每个簇的均值为新的簇的中心。然后不断重复第二步和第三步操作，直到簇的中心不再改变，或者达到要求。因为预测目标准确度主要是与真实边界框和预测边界框的 IOU（Intersection over Union,交并比）有关，与边界框的大小无关，所以 K-Means 算法中的距离使用两个边界框的 IOU 计算距离，如公式(2.28)所示，公式中的 $d\left(box, centroid\right)$ 为边界框和簇中心的距离，$IOU\left(box, centroid\right)$ 为边界框和簇中心的 IOU。

$$d\left(box, centroid\right) = 1 - IOU\left(box, centroid\right) \tag{2.28}$$

3. 边界框坐标预测

YOLOv2 借鉴 Faster-RCNN 的区域生成网络中的锚框，直接使用 Faster-RCNN 中的锚框会出现在早期训练的时候不稳定的情况。目标检测算法会预测出坐标

(t_x, t_y)，x_a、y_a分别为聚类得到的锚框的中心点坐标 x、y，w_a、h_a分别为锚框的宽和高，故预测边界框中心坐标(x,y)的计算如公式(2.29)、公式(2.30)所示。

$$x = (t_x * w_a) - x_a \tag{2.29}$$

$$y = (t_y * h_a) - y_a \tag{2.30}$$

预测到的数据会直接影响最后的边界框的位置，比如t_x=1会让边界框往左移一个位置，t_x=-1会让边界框往右移一个位置。由于边界框中心坐标的计算公式没有任何的限制条件，所以会出现各种取值情况，从而导致边界框会出现在图片上的任意位置，而不是在指定的单元格之内。所以 YOLOv2 在借鉴 Faster-RCNN 的区域生成网络中的锚框的基础上，使用 YOLOv1 中的直接预测的方法，直接预测边界框相对单元格的相对位置[45]。

YOLOv2 在每个单元格内预测 5 个边界框的相关参数，每个边界框预测以下五个参数t_x、t_y、t_w、t_h以及t_o。假设单元格的左上角坐标为(c_x, c_y)，通过聚类得到的先验框的宽和高分别为 p_w和 p_h，则可以得到边界框相对于单元格的坐标如公式(2.31)、公式(2.32) 、公式(2.33) 、公式(2.34) 、公式(2.35)所示。

$$b_x = \sigma(t_x) + c_x \tag{2.31}$$

$$b_y = \sigma(t_y) + c_y \tag{2.32}$$

$$b_w = p_w e^{t_w} \tag{2.33}$$

$$b_h = p_h e^{t_h} \tag{2.34}$$

$$\Pr(object) * IOU(b, object) = \sigma(t_o) \tag{2.35}$$

上式中b_x、b_y分别为预测边界框中心相对于单元格左上角的坐标 x，y，b_w、b_h分别为预测边界框的宽和高。$\Pr(object) * IOU(b, object)$为预测边界框的置信度。

4. 特征融合

YOLOv2 最终是将特征图分割成13×13的网格，在检测尺寸大的目标时候可以完全胜任，但是使用特征融合可以提升检测尺寸较小的目标的准确性。Faster-RCNN 是在多种尺度的特征图上获取候选区域，从而适应不同尺寸的目标。YOLOv2 中使用 passthrough 层将通过卷积神经网络提取特征后的倒数第二层的特征图处理之后，与最后一层的特征图进行叠加从而获得更多细节的信息[46]。

YOLOv2 中图片输入的尺寸是$(416,416,3)$，通过卷积神经网络下采样，最后得到的特征图尺寸是$(13,13,1024)$。在经过最后一个池化层之前特征图的尺寸为$(26,26,512)$，YOLOv2 中的 passthrough 层通过卷积和池化等操作，将其转换为尺寸为$(13,13,2048)$的特征图。将处理后的尺寸为$(13,13,2048)$的特征图与通过卷积神经网络下采样得到的尺寸为$(13,13,1024)$的特征图连接，即可得到一个尺寸为$(13,13,3072)$的特征图，同时包含了更多小尺寸目标的信息。

5. 多尺度训练

YOLOv1 中网络的输入图片尺寸只可以是$(448,448,3)$，对各种不同尺寸图片的泛化能力不够，预测的精准度不够。YOLOv2 由于使用的是卷积层和池化层，从而可以在多种尺度中运行。YOLOv2 使用卷积网络提取特征，从输入图片到最后输出的特征图的下采样倍数是 32 倍，所以输入的图片的尺寸为 32 的倍数即可。

为了提升不同尺寸图片检测时的精度，YOLOv2 在训练的时候使用{320,352,…，608}等十种是 32 的倍数的尺寸的图片用于训练。上述中最小的尺寸是(320,320)，最大的尺寸是(608,608)，通过卷积神经网络下采样，最后得到的最小的特征图尺寸为(10,10),最大的特征图为尺寸为(19,19)。在训练的时候每隔几次迭代之后，将从上面的十种尺寸中随机地更换一种尺寸继续训练，由于训练的时候输入尺寸的多样性，所以检测的时候面对各种尺寸的泛化能力和精度有了提升[47]。

2.3.4 YOLOv3 和 YOLOv3-Tiny 原理

YOLOv3 是一种单阶段的目标检测算法，整体的思想是：输入的图片划分成 13×13 的网格，如果需要检测的目标的中心落在其中某个单元格上，则该单元格负责预测这个目标。YOLOv3 中每个单元格预测 3 个边界框，每个边界框会单独预测相关数据。边界框预测的数据有每个边界框的位置相关数据、边界框内有物体的预测分数、每一个类别的得分。

1. 整体网络结构

YOLOv3 在 YOLOv2 的基础上进行了改进，在多种尺度的特征图上进行目标检测，从而提高了小目标检测的准确率。YOLOv3 延续了 YOLO 系列单阶段的目标检测的思想，只需要关注输入端和输出端，YOLOv3 的整体网络结构如图 2.5 所示。

YOLOv3 输入图片之后，首先经过由卷积模块(Convolutional Block)和残差模块(Residual Block)构成的特征提取网络提取特征，在提取特征的时候使用步长为 2 的卷积进行下采样。卷积模块和残差模块的结构如图 2.6 所示，卷积模块主要由卷积、批量归一化和激活函数 Leaky Relu 组成。残差模块借鉴了 Resnet 的残差网络结构，由卷积模块组成，可以避免网络层数较多时的梯度消失和梯度爆炸的问题。

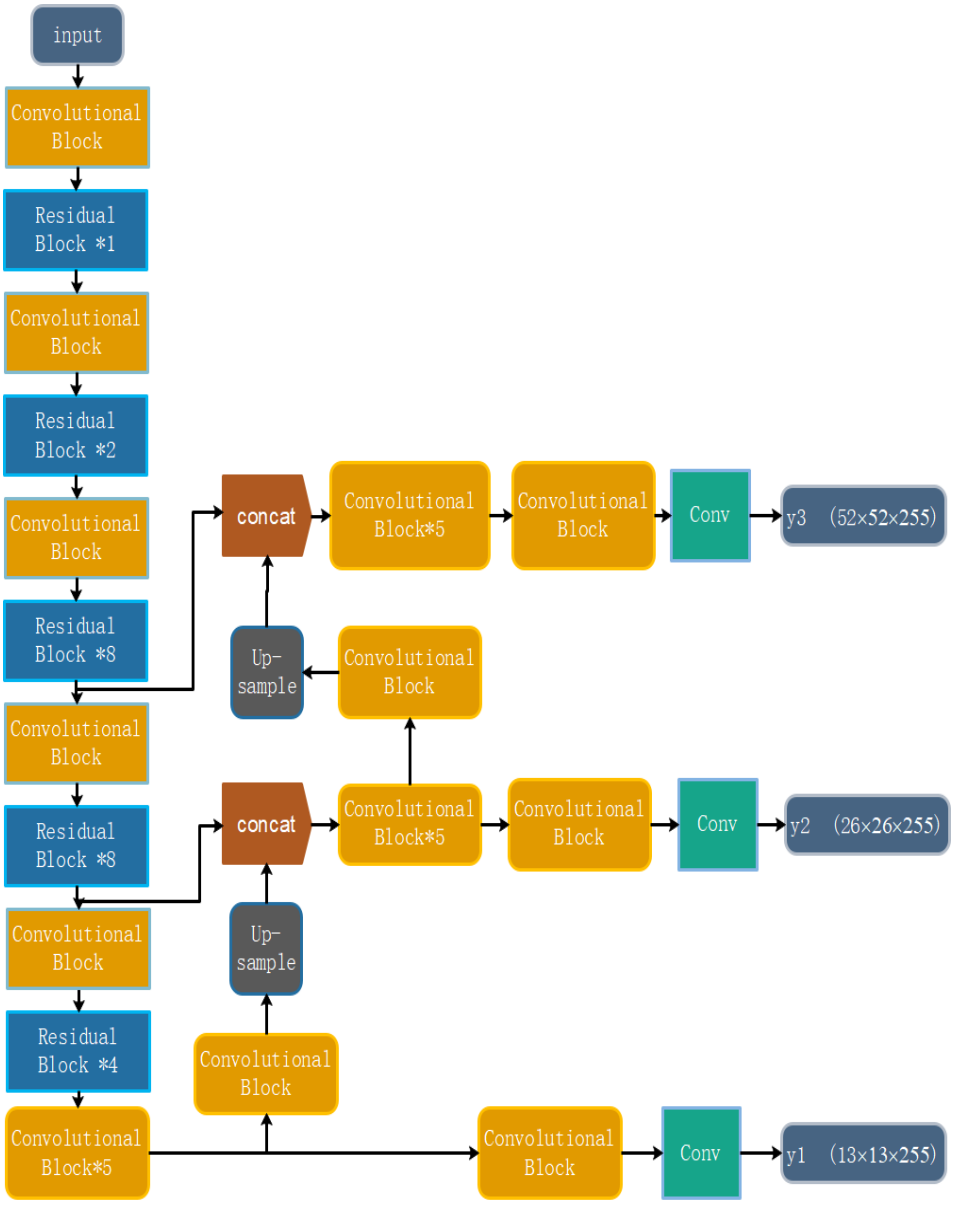
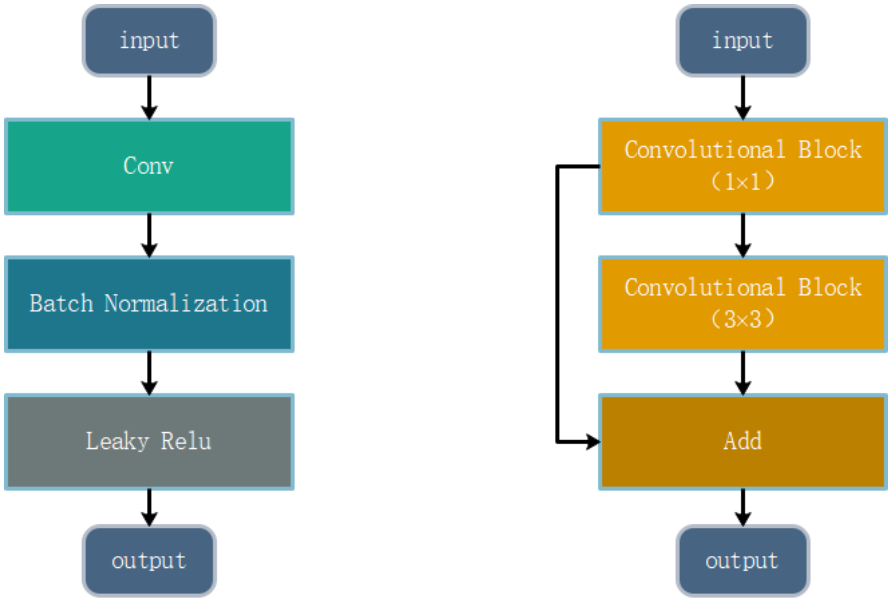


图 2.5 YOLOv3 的整体网络结构



(a) 卷积模块结构图　　(b) 残差模块结构图

图 2.6 卷积模块和残差模块结构图

通过卷积模块和残差模块提取到的特征图，利用卷积模块和卷积进一步提取特征，从而得到一个13×13×255 的输出 y1。YOLOv3 中每一个单元格中共有 3 个边界框，每一个边界框需要单独的预测位置、置信度和每一个类别的概率，其中位置包括边界框的中心坐标 x、y 以及边界框的宽和高。检测的目标共有 80 种，所以每一个单元格预测的值个数为$3\times(4+1+80)$，即每个单元格预测 255 个数据。通过特征提取网络得到的最后一层的特征利用卷积模块和上采样操作得到的新的特征，将新的特征与特征提取网络上一层提取得到的特征进行拼接(concat)，拼接了两层的特征从而使得到的特征图包含更多的信息。拼接后得到的特征图，通过连续的 5 层卷积模块提取特征，得到新的特征图用于本层和上一层的预测，本层的特征通过进一步的卷积模块提取特征之后得到一个 26×26×255 的输出 y2。经过上采样的特征图继续与上一层拼接，得到含有大量信息的新的特征图，新的特征图通过卷积模块提取特征，然后得到一个 52×52×255 的输出 y3。

最后将三种不同尺度的 y1，y2，y3 同时输出，由于有各种尺度的特征图，从而在大目标和小目标检测时都会有较好的精度。当输入图像的尺寸是 416×416×3 的时候，最多可以检测的目标个数为13×13×3+26×26×3+52×52×3，共 10647 个目标。

但是由于 YOLOv3 的特征提取网络层数较多，计算量较大，无法在计算能力较差的移动设备上使用。为了满足在计算能力较差的设备上使用，YOLOv3 有一个网

络结构比较简单，计算量较小可以在移动设备上运行的版本 YOLOv3-Tiny，其网络结构如图 2.7 所示。

YOLOv3-Tiny 网络由卷积层和池化层组成，相对于 YOLOv3 的网络结构简单很多，计算量较少，从而使得检测的速度有了大幅度的提升。YOLOv3-Tiny 中使用最大池化(Maxpool)进行下采样，前五次的最大池化的步长为 2，所以共有 32 倍的下采样，假设输入的图片尺寸为416×416，则得到的特征图尺寸为13×13。假设检测的目标种类总共有 N 种,则得到的输出 y1 尺寸为$13\times13\times\left(3\times\left(N+5\right)\right)$。通过特征提取网络得到的$13\times13$的特征图，通过卷积和上采样操作之后与尺寸为$26\times26$特征图拼接，从而使特征图中保存更多信息。拼接后的特征图通过卷积操作即可得到输出 y2，尺寸为$26\times26\times\left(3\times\left(N+5\right)\right)$。y1 和 y2 可以满足多种尺度的目标的检测。

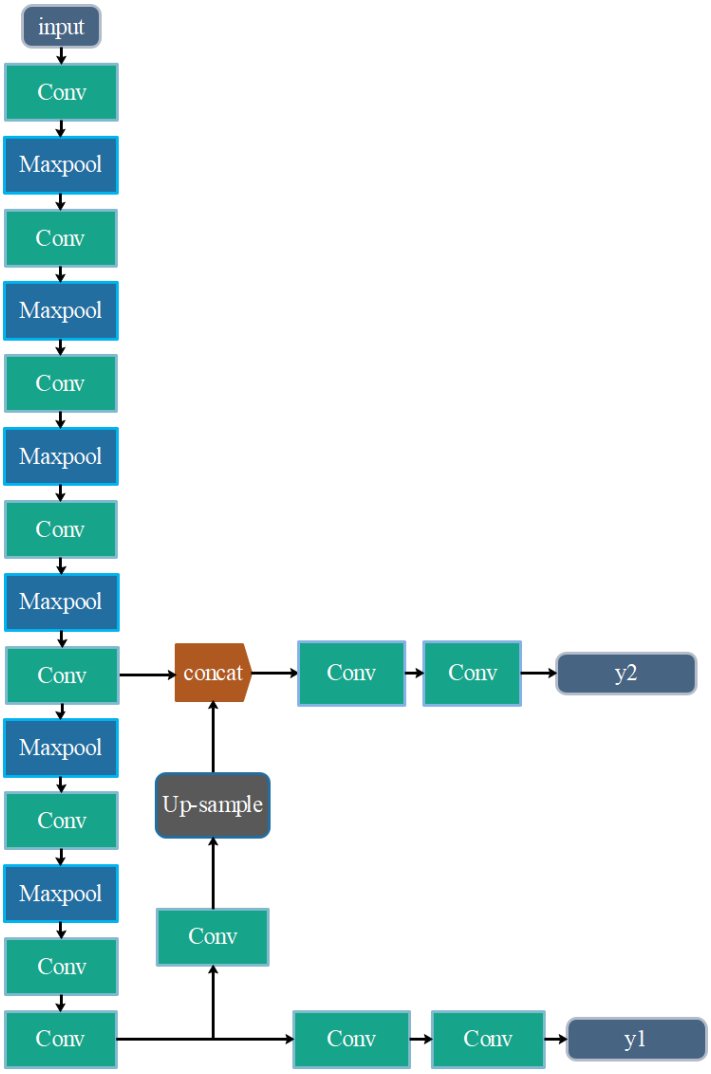


图 2.7 YOLOv3-Tiny 网络结构图

2. 特征提取网络

YOLOv3-Tiny 中特征提取网络由卷积层和最大池化层组成，网络结构如图 2.8 所示。YOLOv3-Tiny 由 8 层卷积层和 6 层最大池化层组成，由于网络结构比较简单，所以检测的精度较低。

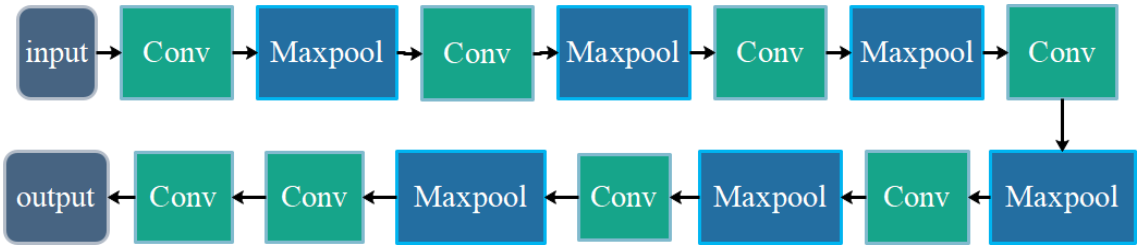


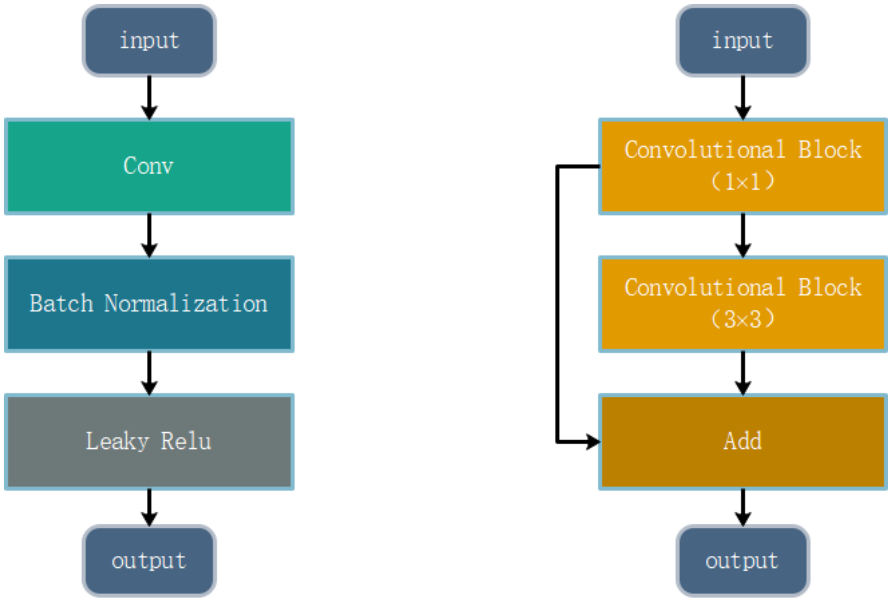
图 2.8 YOLOv3-Tiny 特征提取网络

YOLOv3 中特征提取网络用的是 Darknet-53 的网络结构，Darknet-53 中共有 53 个卷积层故取名 Darknet-53。其网络结构如图 2.9 所示。

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

图 2.9 YOLOv3 特征提取网络

Darknet-53 主要由卷积模块组成，借鉴了残差网络的思想，卷积模块的结构如图 2.10 中(a)所示。Darknet-53 中共由五个残差模块组成，每一个基本的残差模块由卷积核为1×1和卷积核为3×3的卷积模块组成，基本的残差模块的结构如图 2.10 中(b)所示。



(a) 卷积模块结构图　　(b) 残差模块结构图

图 2.10 卷积模块和残差模块结构图

Darknet-53 中没有使用池化层，使用步长为 2 的卷积模块进行下采样，总共有 32 倍的下采样，假设输入图片的尺寸是 256×256 则通过 Darknet-53 下采样之后得到的特征图的尺寸为8×8 。在 YOLOv3 中结合了不同尺度的特征图，即原始图片经过 32 倍、16 倍、8 倍下采样后得到的特征图用来预测目标。

3. 边界框的预测

YOLOv3 和 YOLOv3-Tiny 的边界框预测的方法是一致的，下文只介绍其中一个。YOLOv3 中每一个单元格会预测 3 个边界框，每一个边界框需要预测下列数据。每个边界框需要预测其中心坐标 x、y 以及边界框的宽和高。边界框的置信度，用来判断这个边界框内是否有目标。假设共需要检测 N 种目标，则每个边界框还需要预测每一种目标的概率。

YOLOv3 中借鉴 YOLOv2 中的锚框，使用 k 均值聚类算法(k-means clustering algorithm)求得 9 个先验框。具体步骤如下，首先求出训练集中的所有边界框的长度和宽度，然后用两个边界框的 IOU（Intersection over Union，交并比）作为距离，使用 K 均值聚类算法求得 9 个先验框。比如在 COCO 数据集中求得的 9 个先验框如下所示：(10×13)，(16×30)，(33×23)，(30×61)，(62×45)，(59×119)，(116×90)，(156×198)，(373×326)。由于不同尺度的特征图对不同尺寸目标的感知能力不同，特征图较大时感受野较小，对小目标的检测比较好。由上可知，YOLOv3 中 52×52×255 的输出 y3 对小目标的检测效果比较好，所以在 y3 中使用的先验框是下列三个(10×13)，(16×30)，

(33×23)。YOLOv3 中13×13×255的输出 y1，特征图较小，所以对大目标的检测效果较好，所以用下列三个先验框(116×90)，(156×198)，(373×326)。输出 y2 使用的先验框为(30×61)，(62×45)，(59×119)。

YOLOv3 为每个边界框预测四个值t_x、t_y、t_w、t_h，通过直接预测先验框相对于单元格左上角的相对位置来预测边界框。假设单元格左上角的坐标为(c_x, c_y)，通过 k 均值聚类算法得到的先验框的宽和高分别为p_w、p_h，则可得到预测的边界框的宽和高的计算如公式(2.36)、公式(2.37)所示。

$$b_w = p_w e^{t_w} \tag{2.36}$$

$$b_h = p_h e^{t_h} \tag{2.37}$$

上式中的b_w、b_h即为预测的边界框的宽和高，利用t_w、t_h对先验框的宽和高进行缩放从而得到预测边界框的宽和高。边界框的中心坐标预测如公式(2.38)、公式(2.39)所示。

$$b_x = \sigma(t_x) + c_x \tag{2.38}$$

$$b_y = \sigma(t_y) + c_y \tag{2.39}$$

上式中的b_x、b_y是预测的边界框的中心坐标的 x、y，YOLOv3 中预测的是相对于单元格左上角的偏移量t_x、t_y。$\sigma(t_x)$、$\sigma(t_y)$即将预测的偏移量t_x、t_y利用 sigmoid 函数处理使其值域处于0到1之间，从而让模型在训练前期的时候收敛的速度更快。预测的偏移量经过 sigmoid 函数处理之后，偏移量值不会大于 1 可以防止边界框的中心坐标偏移过多，从而保证预测的边界框中心在单元格之内。

4. 置信度和类别的预测

YOLOv3 和 YOLOv3-tiny 的置信度和类别预测的计算方法都是一样的，如下文所述。YOLOv3 中每一个边界框都会预测一个置信度的值，边界框的置信度主要由两部分组成。首先边界框的置信度用来判断边界框内是否有目标的概率，从而判断边界框内是只有背景还是有目标。其次边界框的置信度包含了预测的边界框和真实的边界框的 IOU（Intersection over Union，交并比）。边界框的置信度计算如公式(2.40)所示。

$$C_i^j = P_r(Object) * IOU_{pred}^{truth} \tag{2.40}$$

上式中的C_i^j是第 i 个单元格的第 j 个边界框的置信度，$P_r\left(Object\right)$是当前边界框是否有目标的概率，IOU_{pred}^{truth}是预测的边界框和真实的边界框的交并比。

YOLOv3 中每一个边界框会单独预测一组各种类别的概率，假设共检测 N 种目标，则 YOLOv3 的每一个边界框会预测 N 种目标的概率。在对目标进行分类的时候，常用的函数是 softmax 函数，但是边界框使用 softmax 函数分类时一个边界框只能检测一种目标，然而有些时候一个边界框内不止有一种目标。YOLOv3 中使用逻辑回归预测每一种类别的概率，所以检测的时候一个边界框可以检测多个目标。

2.4 目标检测算法对比与选择

传统的检测算法需要人工提取特征，所以只可以在背景比较简单，特征比较明显的场景中使用。但是在城市综合管廊的检测中，环境复杂多变，同时特征不是特别统一，所以传统的检测算法无法满足本文的需要。

基于深度学习的两阶段目标检测算法，利用深度卷积神经网络提取特征，拥有较好的泛化能力。但是两阶段目标检测算法需要首先获得候选区域，然后再检测目标，导致计算速度达不到要求，无法在移动设备上实时运行。

基于深度学习的单阶段目标检测算法，输入图片通过特征提取网络之后可以直接得到预测目标的坐标和种类等信息。通过改进特征提取网络的结构和进行多种尺度的检测，在保证检测速度的同时提高了检测的精度。

综上所述，基于深度学习的单阶段目标检测算法对计算设备的要求相对较低，可以在移动设备上运行，同时检测的精度较高可以满足检测的需要。所以本文通过改进单阶段目标检测算法，实现对城市综合管廊的实时检测。

2.5 本章小结

本章详细地介绍了各种视觉检测算法的原理，以及检测的主要流程。分别介绍了传统检测算法、基于深度学习的两阶段目标检测算法、基于深度学习的单阶段目标检测算法的原理和检测流程，并介绍了各种算法的优缺点。

第 3 章 基于改进 YOLOv3-Tiny 的 SS-YOLO 目标检测算法

YOLOv3-Tiny 为了保证检测的速度，导致检测的精度较低。本章通过对 YOLOv3-Tiny 特征提取网络进行改进提升检测的精度和速度，同时对获取先验框的方法进行改进，提出了新的算法 SS-YOLO。利用开源数据集分别训练改进前后的算法，判断改进的效果。

3.1 算法优化

管廊巡检机器人的所用的控制器计算能力只有 1.5T Flops，所以目标检测算法在保证检测精度的基本上应该尽可能得提高检测的速度。YOLOv3 中使用的特征提取网络是 Darknet-53，由于 Darknet-53 网络较为复杂，计算量较大，无法在移动设备上使用。YOLOv3-Tiny 中特征提取网络比较简单，计算量较小，从而实现了较快的检测速度，但是检测精度较低，无法满足检测的需要。本文借鉴 ShuffleNet V2 中多个提高网络速度的技巧从而提高特征提取网络的运算速度，同时参考了 SENet 中通过对特征图在特征通道维度进行重标定从而提高检测精度的思想，对特征提取网络进行了改进。YOLOv3 系列算法中使用 k 均值聚类算法获取先验框，由于其选择初始类别中心时随机挑选，所以可能因为初始类别中心的距离过近导致训练陷入局部的最优点。本文对特征提取网络和获取先验框的方式进行了改进，具体方法如下所示。

3.1.1 特征提取网络改进

由于算法需要在移动设备上使用，所以要尽可能得在降低计算量的同时保证检测的精度。目前用来评价模型复杂度常用的一个指标是 FLOPS（floating-point operations per second, 每秒所执行的浮点运算次数），这个指标主要受卷积运算中的乘法和加法的数量影响。但是两个 FLOPS 相同的模型，运行的速度却不一致，影响模型运行速度的因素除了 FLOPS 还有内存访问成本等因素。

本文的特征提取网络借鉴了 ShuffleNet V2 的思想，主要通过以下四点提高网络的运算速度。

1. 输入通道数和输出通道数相等的时候内存访问成本最小

假设在一个1×1的卷积中，输入通道数和输出通道数分别为c_{in}和c_{out}，特征图的长和宽分别为 h、w，则卷积的 FLOPS 的计算如公式(3.1)所示。

$$FLOPS = c_{in} * c_{out} * h * w \tag{3.1}$$

此时需要的内存访问成本(memory access cost,MAC)计算如公式(3.2)所示。

$$MAC = h * w * \left(c_{in} + c_{out}\right) + c_{in} * c_{out} \tag{3.2}$$

当每秒所执行的浮点运算次数是固定的时候，可以得到$c_{out} = \frac{FLOPS}{c_{in} * h * w}$。将$c_{out}$代入内存访问成本计算公式，然后根据均值不等式可以得到结果如公式(3.3)所示。

$$\begin{aligned} MAC &= h * w * \left(c_{in} + c_{out}\right) + c_{in} * \frac{FLOPS}{c_{in} * h * w} \\ &\geq h * w * 2 * \sqrt{c_{in} * c_{out}} + \frac{FLOPS}{h * w} \\ &= 2 * \sqrt{c_{in} * c_{out} * h * w * h * w} + \frac{FLOPS}{h * w} \\ &= 2 * \sqrt{FLOPS * h * w} + \frac{FLOPS}{h * w} \end{aligned} \tag{3.3}$$

上式中由均值不等式原理可知，当且仅当c_{in}=c_{out}时候，$\left(c_{in} + c_{out}\right)$取得最小值为$2 * \sqrt{c_{in} * c_{out}}$。故由上可知，当且仅当输入通道数$c_{in}$等于输出通道数$c_{out}$的时候，内存访问成本(MAC)取得最小值，从而运算速度也最快。

2. 组卷积中分组的数量越少内存的访问成本越小

卷积神经网络中，很多网络模型会在卷积神经网络中使用组卷积(group convolution)来提升模型的速度，因为组卷积会大幅度地降低模型的每秒所执行的浮点运算次数，同时分组之后的每组卷积提取的特征不会大量冗余，从而在提高了模型的速度的同时提高了检测的精度。分成 g 组之后，在一个1×1的卷积中，输入通道数和输出通道数分别为c_{in}和c_{out}，特征图的长和宽分别为 h、w，则卷积的 FLOPS 如公式(3.4)所示。

$$FLOPS = \frac{c_{in} * c_{out} * h * w}{g} \tag{3.4}$$

此时的需要的内存访问成本，计算如公式(3.5)所示。

$$MAC = h*w*\left(c_{in}+c_{out}\right)+\frac{c_{in}*c_{out}}{g} \tag{3.5}$$

假设卷积的每秒所执行的浮点运算次数(FLOPS)是固定的，则输出通道的个数可以转换成$c_{out}=\frac{FLOPS*g}{c_{in}*h*w}$，内存访问成本的计算公式可以转换成公式(3.6)。

$$\begin{aligned} & MAC \\ & = h*w*\left(c_{in}+c_{out}\right)+c_{in}*\frac{FLOPS}{c_{in}*h*w} \\ & \geq h*w*2*\sqrt{c_{in}*c_{out}}+\frac{FLOPS}{h*w} \\ & = 2*\sqrt{c_{in}*c_{out}*h*w*h*w}+\frac{FLOPS}{h*w} \\ & = 2*\sqrt{FLOPS*h*w}+\frac{FLOPS}{h*w} \end{aligned} \tag{3.6}$$

由上式可知当每秒所执行的浮点运算次数不变的时候，分组数 g 越大的时候，内存访问成本越大，所以在模型中不使用分组数较大的组卷积有利于提升模型的速度。

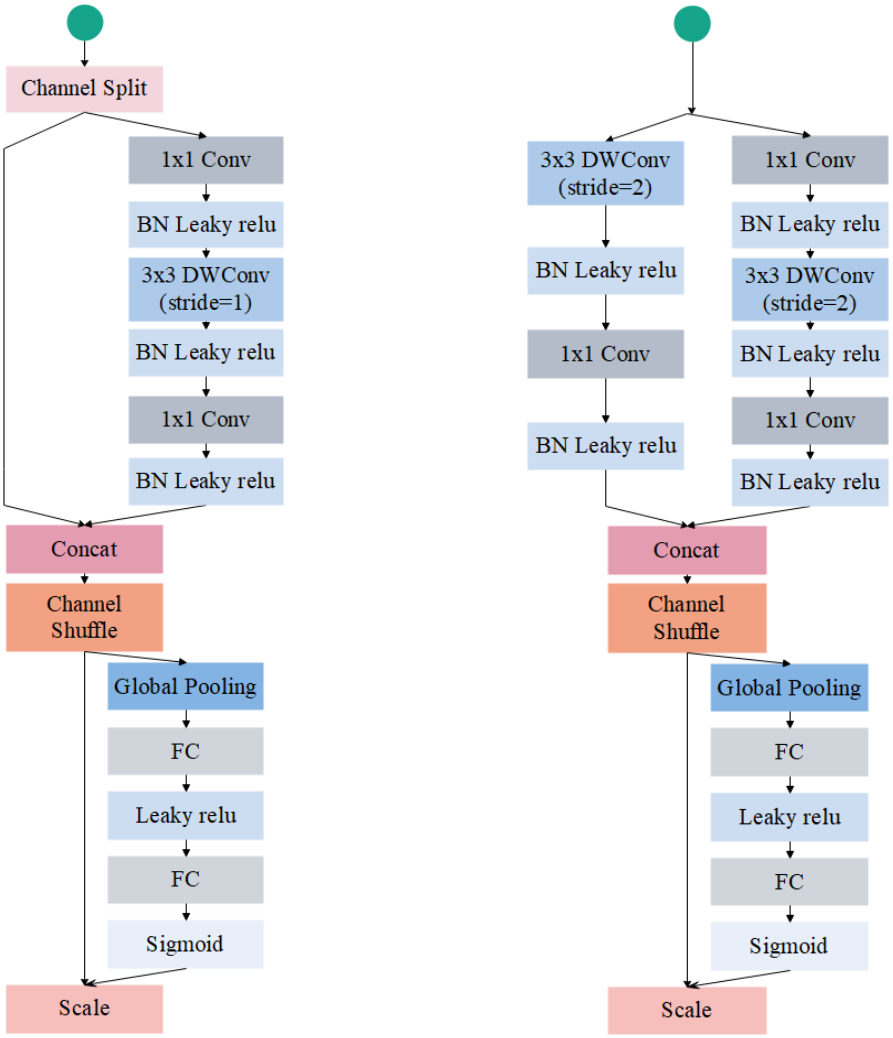
3. 使用较少的支路可以提高网络的运算速度

网络的碎片化也会降低模型的并行度，即当模型中有较多的支路并行的时候，类似于 Inception 的网络结构，由于支路较多导致运算的速度较低。所以在模型中应尽可能得减少支路的数量，从而提高模型的速度。

4. 减少元素级操作可以提高网络的运算速度

模型中的元素级操作(element-wise operators)，如卷积神经网络中常用的 Relu 和 Add 等操作，虽然这些操作的每秒所执行的浮点运算次数较少，但是却需要的内存访问成本较大，所以会导致模型的速度降低。

综上所述，本文借鉴了上述 ShuffleNet V2 的思路，通过以下几点对网络进行优化。第一，让输入输出的通道数相等。第二，较少的使用组卷积，即使必须使用组卷积也要尽可能得减小分组数。第三，避免模型的碎片化，减少模型中的支路。第四，减少模型中的类似于 Relu 和 Add 等的元素级操作。综合上述特点，本文设计了特征提取网络中的 Shuffle_Senet 模块，结构如图 3.1 所示。



(a) 步长为 1 时 Shuffle_Senet 模块网络结构　(b) 步长为 2 时 Shuffle_Senet 模块网络结构

图 3.1 Shuffle_Senet 模块网络结构

Shuffle_Senet 模块有两种网络结构，其中步长为 2 的 Shuffle_Senet 模块用来做特征图的下采样。在步长为 1 的 Shuffle_Senet 模块中，模块的输入首先通过通道分割(Channel Split)操作,将输入的特征图换分成两个部分，假设输入的通道数为 c，如果输入的通道数 c 是 2 的倍数，则两个部分的通道数都为 $\frac{c}{2}$。如果不能整除，则一部分的通道数为 c 除以 2 得到的商称为 c_1，另一部分的通道数为 c_2 值为 $c_2 = c - c_1$。为了减少分支和基本的元素，其中一部分不采取任何操作，另外一部分使用卷积和深度可分离卷积(Depthwise Separable Convolution)进行特征提取。卷积层和深度可分离卷积层之后添加批量归一化(Batch Normalization)层,从而提高训练的稳定性，提升模型训练的速度，在一定程度上解决了梯度消失的问题。同时批量归一化有一定的正则化的效果，可以防止模型的过拟合。ShuffleNet V2 中使用的

激活函数 Relu，但是当输入值为负的时候，激活函数的输出将一直是零，同时一阶导数也一直是零，从而导致神经元不再更新参数。模块中使用的激活函数是 Leaky Relu（Leaky Rectified linear unit, 带泄露修正线性单元），可以有效的解决 ShuffleNet V2 中使用的 Relu 激活函数出现的相关问题。然后将两个分支拼接(concat)起来，从而保证输入和输出通道相同。

拼接之后得到的特征图，借鉴了 SENet(Squeeze and Excitation Networks)网络的思想，通过学习获得不同通道的重要程度，然后根据得到的结果去提升有用的通道抑制不太重要的通道，从而提高网络的性能。具体操作如下所述，首先压缩(squeeze)操作,使用全局平均池化(global average pooling)对特征图进行压缩，假设输入的特征图尺寸为 $h*w*c$，则通过全局平均池化得到的尺寸为 $1*1*c$。通过全局平均池化处理之后，使用两个全连接层学习不同通道的重要性，第一个全连接层使用的激活函数是 Leaky Relu，第二个全连接层使用的激活函数是 sigmoid。通过 sigmoid 函数得到一个处于零到一之间的标量，用来作为通道的权重。得到的权重与原始的特征图的每一个通道相乘，从而根据通道的重要性对原始特征图在通道维度上进行重标定。

当 Shuffle_Senet 模块中步长为 2 的时候，利用步长为 2 的卷积进行下采样。输入的特征图复制一份，然后两个支路输入为相同的特征图，两边分别使用步长为 2 的卷积进行下采样操作。通过卷积和深度可分离卷积提取特征之后，将两边的特征图进行拼接得到新的特征图，由于两边都进行了下采样，所以最后得到的特征图的空间减小，但是通道数是原来的 2 倍。上述操作得到新的特征图之后，将新的特征图使用全局平均池化(global average pooling)对特征图进行压缩，之后使用两层全连接层学习不同通道的重要性。最后一个全连接层的激活函数是 sigmoid 函数，从而得到一个处于零到一之间的权重，用权重将原始特征图在通道维度上进行重标定。本文使用的特征提取网络主要由上述的 Shuffle_Senet 模块组成，特征提取网络整体结构如图 3.2 所示。

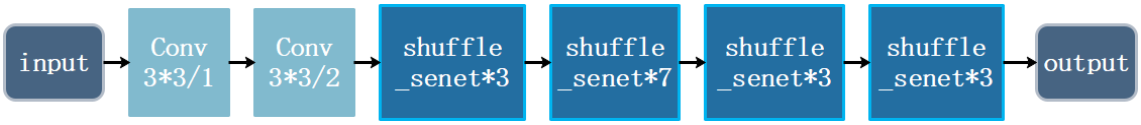


图 3.2 特征提取网络结构

当图片输入到特征提取网络之后，首先会通过两个普通的卷积操作提取特征和下采样，之后分别将 3 个、7 个、3 个、3 个 Shuffle_Senet 模块组合得到新的模块。组合之后得到的新的模块，除了第一个 Shuffle_Senet 模块用来下采样操作，其他的 Shuffle_Senet 模块均用来提取特征，所以每一个新的模块之后均可以输出不同尺度的特征图，从而满足多尺度检测的需要。特征提取网络中共有 5 次下采样操作，第一次为步长为 2 的卷积层实现的下采样，后面四次为多个 Shuffle_Senet 模块组合在一起的时候，每次的第一个 Shuffle_Senet 模块使用的是步长为 2 的 Shuffle_Senet 模块进行下采样，其他的使用步长为 1 的 Shuffle_Senet 模块。假设输入的图片的尺寸是 416×416，在目标检测算法中则通过特征提取网络得到的多种尺度的特征图的尺寸分别为13×13、26×26、52×52。

3.1.2 获取先验框方法改进

YOLOv3 系列算法中使用 k 均值聚类算法(k-means clustering algorithm)获取先验框，但是 k 均值聚类算法的初始点是随机选取的，导致每一种不同的初始点的时候会有不同的聚类结果，所以在训练的时候有可能会陷入局部最优。

本文使用 k 均值聚类算法的优化版本 kmeans++来获取先验框，数据集为目标检测训练集中的所有边界框。标准的 k 均值聚类算法中用的是欧几里德距离，假设两个点分布为(x_1,y_1)、(x_2,y_2)，则这两个点的欧式距离计算如公式(3.7)所示。

$$\text{distance}=\sqrt{(x_2-x_1)^2+(y_2-y_1)^2} \tag{3.7}$$

但是如果边界框聚类的时候使用欧几里德距离，则聚类的结果主要是受边界框大小的影响，但是在边界框使用的时候性能主要由交并比决定，与边界框的大小无关，所以使用交并比作为距离计算规则，可以得到两个边界框box_1、box_2的距离计算如公式(3.8)所示。

$$\text{distance}=1-IOU(box_1,box_2) \tag{3.8}$$

上式中的 IOU 是两个边界框的交并比，即两个边界框的交集的面积除以两个边界框的并集的面积。假设box_1的宽和高都小于box_2的宽和高则 IOU 计算如公式(3.9)所示。

$$IOU(\text{box}_1, box_2) = \frac{W_1 * H_1}{W_2 * H_2} \tag{3.9}$$

假设box_1的宽小于box_2的宽，box_1的高大于box_2的高则 IOU 计算如公式(3.10)所示。

$$IOU(\text{box}_1, box_2) = \frac{W_1 * H_2}{W_1 * H_1 + (W_2 - W_1) * H_2} \tag{3.10}$$

假设box_1的宽大于box_2的宽，box_1的高小于box_2的高则 IOU 计算如公式(3.11)所示。

$$IOU(\text{box}_1, box_2) = \frac{W_2 * H_1}{W_1 * H_1 + (H_2 - H_1) * W_2} \tag{3.11}$$

假设box_1的宽大于box_2的宽，box_1的高大于box_2的高则 IOU 计算如公式(3.12)所示。

$$IOU(\text{box}_1, box_2) = \frac{W_2 * H_2}{W_1 * H_1} \tag{3.12}$$

使用 kmeans++算法获取先验框的具体步骤如下所述。首先从训练集中提取出所有边界框的坐标信息，一个边界框的坐标构成一个样本。从所有的样本之中随机选择一个样本作为一个类别的中心，然后将数据集中所有样本根据上面的距离公式计算每一个样本与第一个类别中心的距离。第 i 个样本成为下一个类别中心的概率，通过第 i 个样本与第一个类别中心的距离除以所有样本与第一个类别中心的距离之和得到。第 i 个样本成为下一个类别中心的概率 p_i 计算如公式(3.13)所示。

$$p_i = \frac{\text{distance}_i}{\sum_{j=1}^{n} \text{distance}_j} \tag{3.13}$$

假设候选样本有三个，得到的概率分别为 0.2、0.3、0.5，则可以根据概率划分出三个区间$[0.0, 0.2)$、$[0.2, 0.5)$、$[0.5, 1.0)$。随机生成一个零到一之间的随机数，随机数落在哪个区间之内则哪一个样本为下一个类别的中心。假设生成的随机数为 0.3，由上面可知落在第二个区间内，则第二个样本为下一个类别的中心。根据第二个样本的生成规则，继续生成类别的中心，直到生成 K 个初始的类别的中心。

生成 K 个初始的类别的中心之后，依次计算每一个样本与 K 个初始的类别的中心的距离，然后将每一个样本放到距离类别的中心最近的那个类别之中。计算每

一个类别的均值，然后利用得到的均值更新每一个类别的中心。最后不断重复将样本分类和更新类别的中心这两步操作，直到类别的中心不再改变为止，即可得到需要的先验框的数据。本文根据上述算法求得的先验框如表 3.1 所示。

表 3.1 先验框宽度与高度

序号	1	2	3	4	5	6	7	8	9
宽度（像素）	19	49	58	50	93	107	122	193	228
高度（像素）	74	34	65	160	92	43	157	73	198

3.2 基于 YOLOv3-Tiny 改进的目标检测算法 SS-YOLO

由于本文的目标检测算法最终需要在移动设备上使用，所以计算能力无法满足 YOLOv3 的需求，所以本文参考速度更快、计算能力需求更低的 YOLOv3-Tiny 模型。本文借鉴了 YOLOv3-Tiny 的多尺度检测的思想，对模型的特征提取算法以及获取先验框的方法进行了改进，得到了基于 YOLOv3-Tiny 改进的目标检测算法 SS-YOLO，在保证速度的同时提高了检测的精度。

3.2.1 SS-YOLO 的网络结构

SS-YOLO 的特征提取模块主要由 Shuffle_Senet 模块组成，借鉴了 YOLOv3-Tiny 的多尺度检测的思想。本文中的输入图片的尺寸是 416×416，有三种维度的输出分别为13×13、26×26、52×52，其整体网络结构如图 3.3 所示。

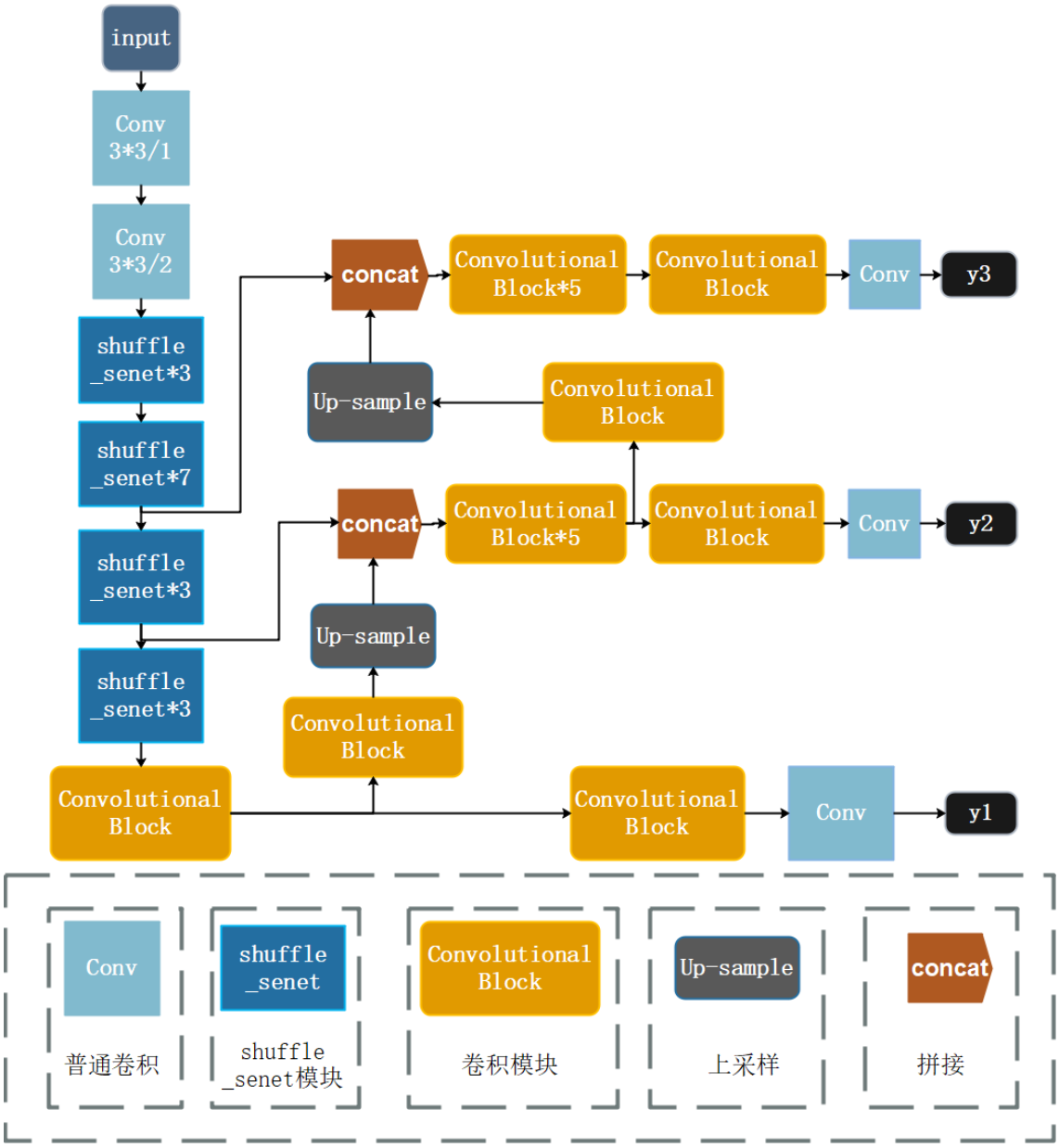
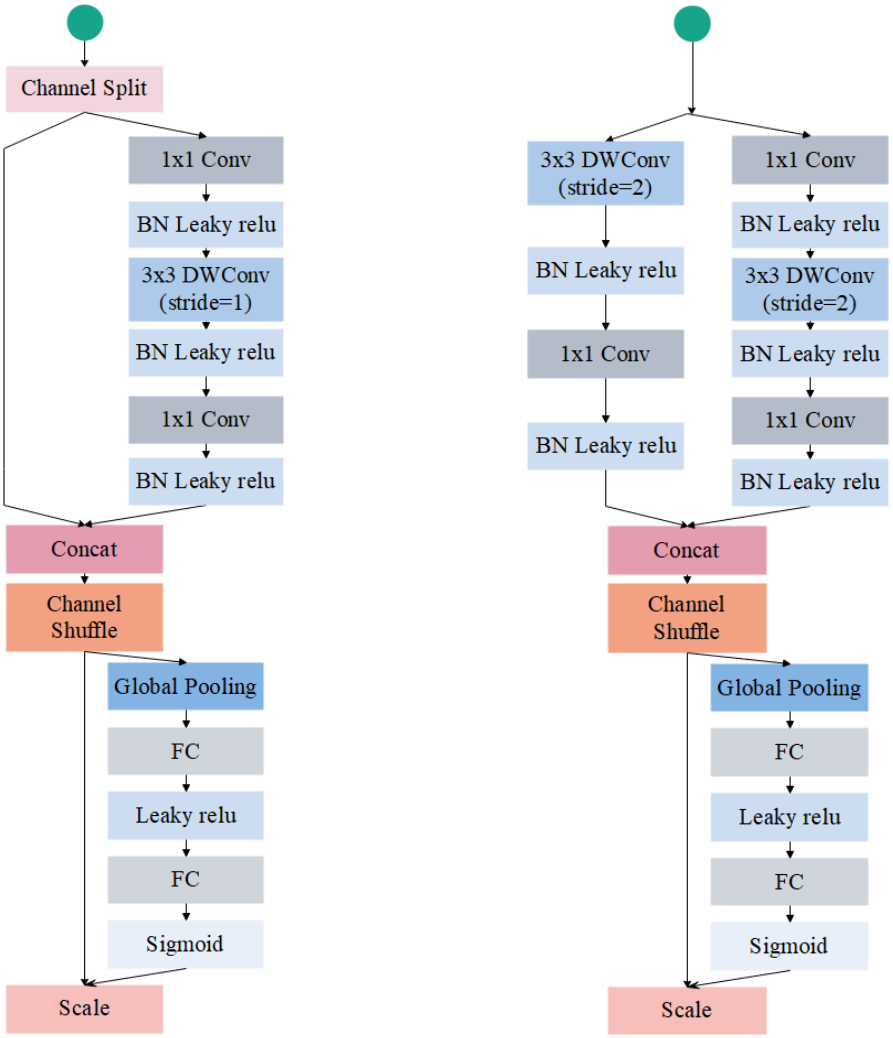


图 3.3 SS-YOLO 整体网络结构

图片输入到 SS-YOLO 之后，通过一个步长为 1 的普通卷积提取特征，然后经过一个步长为 2 的普通卷积实现下采样，之后通过多个 Shuffle_Senet 模块提取特征，Shuffle_Senet 模块具体结构如图 3.4 所示。当有多个 Shuffle_Senet 模块在一起的时候，如网络结构图中的 Shuffle_Senet 模块乘以 3，即表示共有 3 个 Shuffle_Senet 模块，其中第一个 Shuffle_Senet 模块中卷积的步长为 2，用来实现下采样，其余几个 Shuffle_Senet 模块的步长为 1 用来提取特征。SS-YOLO 中使用普通卷积实现了一次下采样，四次在 Shuffle_Senet 模块中实现下采样操作，故总共有 32 倍下采样。



(a) 步长为 1 时 Shuffle_Senet 模块网络结构　(b) 步长为 2 时 Shuffle_Senet 模块网络结构

图 3.4 Shuffle_Senet 模块网络结构

通过特征提取网络提取到的13×13的特征图，使用卷积模块进一步提取特征之后，即可得到尺寸最小的输出 y1。卷积模块主要由普通卷积、批量归一化和激活函数 Leaky Relu 三个部分组成。假设检测的类别总共有 N 类，则 y1 的尺寸为 $13\times13\times\left(3\times\left(5+N\right)\right)$。SS-YOLO 中每一个单元格内有 3 个先验框，每一个先验框需要预测的数据有中心的坐标偏移数据 t_x、t_y，先验框的宽度和高度的缩放数据 t_w、t_h，先验框的置信度、以及每一个类别的概率，所以一个单元格需要预测的数据个数为 $3\times\left(5+N\right)$。通过特征提取网络提取到的13×13的特征图，经过卷积和上采样等操作，即可得到一个新的26×26的特征图。新的特征图与特征提取网络中上一层得到的26×26的特征图拼接，通过卷积模块进一步提取特征之后，既可以得到第二个输入 y2，y2 的尺寸为 $26\times26\times\left(3\times\left(5+N\right)\right)$。上一步拼接后得到的特征图通过卷积和

上采样操作之后得到一个52×52的特征图，将此特征图与特征提取网络中得到的52×52的特征图拼接，然后通过卷积模块提取特征之后，得到输出 y3，y3 的尺寸为$52\times52\times\left(3\times\left(5+N\right)\right)$。

综上所述，一个尺寸为416×416的图片输入到 SS-YOLO 网络之后，会得到$13\times13\times\left(3\times\left(5+N\right)\right)$、$26\times26\times\left(3\times\left(5+N\right)\right)$、$52\times52\times\left(3\times\left(5+N\right)\right)$三种尺度的输出，分别用来检测大目标、中目标、小目标。每一个尺度的输出中，图片会被划分成多个单元格，每一个单元格内有 3 个先验框可以单独预测目标。每一个先验框预测的数据有先验框中心点的坐标偏移的数据t_x、t_y和边界框宽和高的缩放数据t_w、t_h，利用上述数据通过计算可以得到预测的边界框，具体操作如下节所述。每个先验框还会预测一个置信度的值，在检测的时候会设置一个阈值，当置信度超过阈值的时候输出这个目标，否则不输出。假设共检测 N 类目标，则每个先验框会预测每一个类别的概率，当置信度超过阈值的时候，用预测的概率判断先验框中目标的种类。

3.2.2 边界框的预测

SS-YOLO 中使用 kmeans++算法求得 9 个先验框。具体步骤如下，首先求出训练集中所有边界框的长度和宽度，然后用两个边界框的 IOU(Intersection over Union, 交并比)作为距离，使用 kmeans++算法求得 9 个先验框。在管廊检测数据集中求得的 9 个先验框如下所示：(19×74), (49×34), (58×65), (50×160), (93×92), (107×43), (122×157), (193×73), (228×198)。由于不同尺度的特征图对不同尺寸目标的感知能力不同，特征图较大时感受野较小，对小目标的检测比较好。由上可知，SS-YOLO 中$52\times52\times\left(3\times\left(5+N\right)\right)$的输出 y3 对小目标的检测效果比较好，所以在 y3 中使用的先验框是下列三个(19×74), (49×34), (58×65)。SS-YOLO 中$13\times13\times\left(3\times\left(5+N\right)\right)$的输出 y1，特征图较小，所以对大目标的检测效果较好，所以用下列三个先验框(122×157), (193×73), (228×198)。输出 y2 使用的先验框为(50×160), (93×92), (107×43)。

SS-YOLO 预测边界框的方法与 YOLOv3 基本一致，使用 kmeans++算法得到的先验框的宽和高分别为p_w、p_h。通过 SS-YOLO 得到先验框的宽和高的缩放系数t_w、t_h，则可得到预测的边界框的宽和高的计算如公式(3.14)、公式(3.15)所示。

$$b_w = p_w e^{t_w} \tag{3.14}$$

$$b_h = p_h e^{t_h} \tag{3.15}$$

上式中的b_w、b_h即为预测的边界框的宽和高，利用t_w、t_h对先验框的宽和高进行缩放从而得到预测边界框的宽和高。边界框中心坐标的预测通过直接预测边界框相对于单元格左上角的相对位置来计算，SS-YOLO 中预测出相对于单元格左上角的偏移量t_x、t_y。假设单元格左上角的坐标为(c_x, c_y)边界框的中心坐标预测如公式(3.16)、公式(3.17)所示。

$$b_x = \sigma(t_x) + c_x \tag{3.16}$$

$$b_y = \sigma(t_y) + c_y \tag{3.17}$$

上式中的b_x、b_y是预测的边界框的中心坐标的 x、y，$\sigma(t_x)$、$\sigma(t_y)$为预测的偏移量t_x、t_y利用 sigmoid 函数处理使其值域处于 0 到 1 之间，从而让模型在训练前期的时候收敛的速度更快。预测的偏移量经过 sigmoid 函数处理之后，偏移量值不会大于 1 可以防止边界框的中心坐标偏移过多，从而保证预测的边界框中心在单元格之内。

综上所述，可以得到预测的边界框的中心坐标b_x、b_y和预测边界框的宽和高b_w、b_h，从而获得目标的边界框。

3.2.3 损失函数

SS-YOLO 的损失函数由边界框中心坐标损失、边界框的宽和高损失、置信度损失、分类损失四个部分组成。SS-YOLO 的中心坐标损失函数如公式(3.18)所示。

$$loss_{x,y} = \sum_{i=0}^{N^2} \sum_{j=0}^{B} \ell_{ij}^{obj} \left[\left(x_i^j - \hat{x}_i^j \right)^2 + \left(y_i^j - \hat{y}_i^j \right)^2 \right] \tag{3.18}$$

公式(3.18)中的x_i^j表示第 i 个单元格的预测的边界框的相对于单元格左上角的 x 坐标，$\hat{x}_i^j$表示第 i 个单元格的真实的边界框的相对于单元格左上角的 x 坐标，y_i^j和$\hat{y}_i^j$分别表示 i 个单元格的预测的和真实的边界框的 y 坐标。ℓ_{ij}^{obj}表示第 i 个单元格中第 j 个边界框负责预测这个目标，因为第 i 个单元格中第 j 个边界框与真实的边

界框的 IOU(Intersection over Union,交并比)较大。公式中的 N^2 是因为共有 N^2 个单元格，B 为每一个单元格内的边界框的个数。

SS-YOLO 边界框的宽和高损失函数如公式(3.19)所示。

$$loss_{w,h}=\sum_{i=0}^{N^2}\sum_{j=0}^{B}\ell_{ij}^{obj}\left[\left(\sqrt{w_i^j}-\sqrt{\hat{w}_i^j}\right)^2+\left(\sqrt{h_i^j}-\sqrt{\hat{h}_i^j}\right)^2\right] \tag{3.19}$$

公式中的 w_i^j 为预测的边界框的宽度，h_i^j 为预测的边界框的高度，$\hat{w}_i^j$ 为真实的边界框的宽度，$\hat{h}_i^j$ 为真实的边界框的高度。实际检测中的物体尺寸可能差距较大，较大物体有一个相对于自身尺寸较小的偏差 A 时，此时的损失函数应该要小于较小物体有一个相对于自身尺寸较大的偏差 A，所以宽和高的损失函数中使用的是宽和高的平方根。

SS-YOLO 中置信度损失用的是交叉熵损失函数，计算如公式(3.20)所示。

$$\begin{aligned}loss_{confidence}=&\sum_{i=0}^{N^2}\sum_{j=0}^{B}\ell_{ij}^{obj}-\left(C_i^j\log(\hat{C}_i^j)+(1-C_i^j)\log(1-\hat{C}_i^j)\right)\\&-\sum_{i=0}^{N^2}\sum_{j=0}^{B}\ell_{ij}^{nobj}\left(C_i^j\log(\hat{C}_i^j)+(1-C_i^j)\log(1-\hat{C}_i^j)\right)\end{aligned} \tag{3.20}$$

公式(3.20)中 ℓ_{ij}^{obj} 表示第 i 个单元格中的第 j 个边界框中有目标，ℓ_{ij}^{nobj} 表示第 i 个单元格中的第 j 个边界框中没有目标。$\hat{C}_i^j$ 表示第 i 个单元格中第 j 个边界框的预测的置信度，C_i^j 表示第 i 个单元格中第 j 个边界框的真实的置信度。

SS-YOLO 中分类损失用的是交叉熵损失函数，计算公式如公式(3.21)所示。

$$loss_{classes}=\sum_{i=0}^{N^2}\sum_{j=0}^{B}\ell_i^{obj}-\sum_{c\in classes}\left[p_i^j(c)\log(\hat{p}_i^j(c))+(1-p_i^j(c))\log(1-\hat{p}_i^j(c))\right] \tag{3.21}$$

公式(3.21)中的 $\hat{p}_i^j(c)$ 表示预测的类别 c 的概率，$p_i^j(c)$ 表示真实的类别 c 的概率。在损失函数中添加 λ_{coord} 和 λ_{noobj} 等参数用来增加边界框坐标的损失值和减小置信度的损失值，从而解决某些没有物体的单元格置信度接近 0 以及梯度过高导致的模型不稳定等问题。综上所述，可以得到 SS-YOLO 的损失函数如公式(3.22)所示。

$$
\begin{aligned}
&LOSS \\
&= \lambda_{coord} \sum_{i=0}^{N^2} \sum_{j=0}^{B} \ell_{ij}^{obj} \left[\left(x_i^j - \hat{x}_i^j \right)^2 + \left(y_i^j - \hat{y}_i^j \right)^2 \right] \\
&+ \lambda_{coord} \sum_{i=0}^{N^2} \sum_{j=0}^{B} \ell_{ij}^{obj} \left[\left(\sqrt{w_i^j} - \sqrt{\hat{w}_i^j} \right)^2 + \left(\sqrt{h_i^j} - \sqrt{\hat{h}_i^j} \right)^2 \right] \\
&- \sum_{i=0}^{N^2} \sum_{j=0}^{B} \ell_{ij}^{obj} \left(C_i^j \log(\hat{C}_i^j) + (1 - C_i^j) \log(1 - \hat{C}_i^j) \right) \\
&- \sum_{i=0}^{N^2} \sum_{j=0}^{B} \ell_{ij}^{nobj} \left(C_i^j \log(\hat{C}_i^j) + (1 - C_i^j) \log(1 - \hat{C}_i^j) \right) \\
&- \sum_{i=0}^{N^2} \sum_{j=0}^{B} \ell_{i}^{obj} \sum_{c \in classes} \left[p_i^j(c) \log(\hat{p}_i^j(c)) + (1 - p_i^j(c)) \log(1 - \hat{p}_i^j(c)) \right]
\end{aligned}
\tag{3.22}
$$

3.3 算法改进前后性能的对比

3.3.1 实验系统环境

本文所实现的目标检测算法是基于 Keras 和 TensorFlow 框架开发的，实验在 win10 系统下实现。电脑的 CPU 型号为 Intel(R)Core(TM)i7-8700K，GPU 型号为 NVIDIA GeForce GTX1080，显存为 8G。

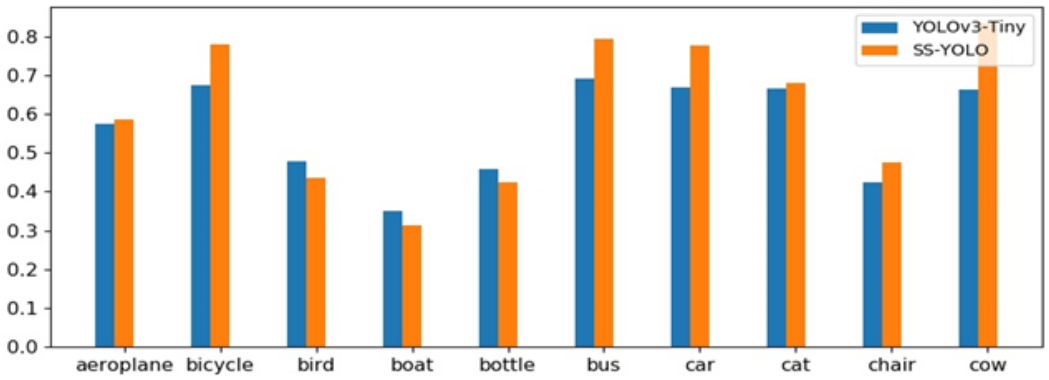
3.3.2 实验数据集

为了对比算法改进前后性能的差异，本文在公开的数据集上分别训练两种算法。本文选择的数据集是目标检测算法中常用的数据集 VOC2007，是 PASCAL VOC(Pattern Analysis， Statistical Modelling and Computational Learning Visual Object Classes)比赛中使用的数据集。VOC2007 数据集中训练集有 5011 张图片，共有 12608 个目标，测试集中有 4952 张图片，共有 12032 个目标。VOC2007 中有 20 类目标，分为 4 大类，第一大类，人：人(person) ；第二大类，动物：鸟(bird)，猫(cat)，牛(cow)，狗(dog)，马(horse)，羊(sheep)；第三大类，交通工具：飞机(aeroplane)，自行车(bicycle)，船(boat)，公共汽车(bus)，汽车(car)，摩托车(motorbike)，火车(train)；第四大类，室内物品：瓶子(bottle)，椅子(chair)，餐桌(dining table)，盆栽(potted plant)，沙发(sofa)，电视或显示器(tv/monitor)。

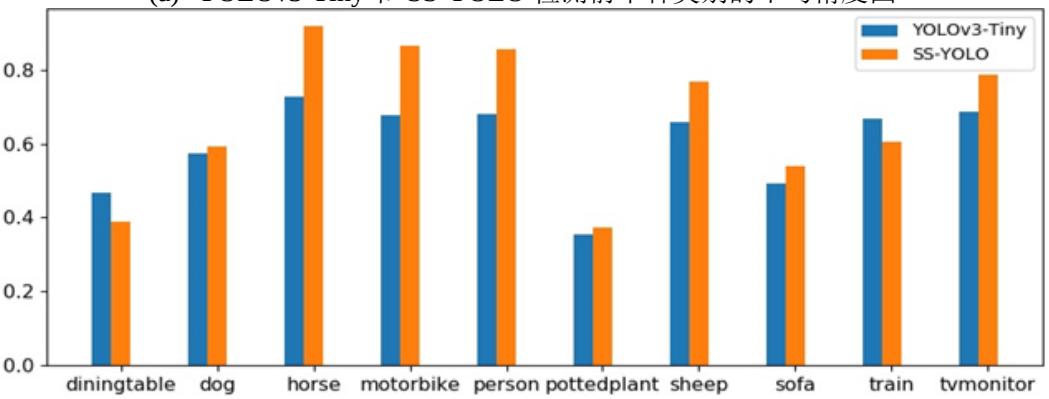
3.3.3 实验结果分析

本文使用 VOC2007 数据集的训练集分别训练 YOLOv3-Tiny 和 SS-YOLO 之后，使用 VOC2007 数据集的测试集对两个训练好的模型进行测试。测试的指标有两个，一个是评价目标检测速度的指标 FPS(Frames Per Second, 每秒传输帧数)，另一个是评价目标检测精度的指标 MAP(Mean Average Precision,平均精度均值)。

通过计算 20 种类别的平均精度(Average Precision)的平均值从而得到平均精度均值，通过计算得到 YOLOv3-Tiny 和 SS-YOLO 两种模型检测每一个类别的平均精度，如图 3.5 所示。



(a) YOLOv3-Tiny 和 SS-YOLO 检测前十种类别的平均精度图



(b) YOLOv3-Tiny 和 SS-YOLO 检测后十种类别的平均精度图

图 3.5 YOLOv3-Tiny 和 SS-YOLO 检测的平均精度图

上图中的蓝色柱状图为 YOLOv3-Tiny 检测的每一类的平均精度，橙色柱状图为 SS-YOLO 检测的每一类的平均精度。从图中可以看出，SS-YOLO 检测的大部分类别的平均精度都要高于 YOLOv3-Tiny 的平均精度，只有几个类别的平均精度略低于 YOLOv3-Tiny。通过计算所有类别的平均精度的均值从而得到 SS-YOLO 和 YOLOv3-Tiny 的平均精度均值如表 3.2 所示，同时 SS-YOLO 和 YOLOv3-Tiny 的每秒传输帧数如表 3.2 所示。

表 3.2 SS-YOLO 和 YOLOv3-Tiny 的检测相关数据

Object detection algorithm	FPS（帧）	MAP
YOLOv3_Tiny	25	0.582
SS-YOLO	29	0.647

由于 SS-YOLO 使用基于 Shuffle_Senet 模块的特征提取网络，所以算法的速度和精度有了一定的提高，同时获取到更加合适的先验框也提高了网络的检测精度。由上表可以看出 YOLOv3-Tiny 的检测精度和检测的速度都较低，SS-YOLO 在每秒传输帧数比 YOLOv3-Tiny 多出 4 帧，SS-YOLO 的平均精度均值要比 YOLOv3-Tiny 高 6.5%。由上述结果可知，SS-YOLO 的性能比 YOLOv3-Tiny 提升了很多，更适合在管廊巡检机器人上应用。

3.4 本章小结

本章首先介绍了 YOLOv3 和 YOLOv3-Tiny 的原理，通过对特征提取网络和获取先验框的方式进行改进，提出一种新的目标检测算法 SS-YOLO。利用开源数据集 VOC2007 训练和测试两种算法，SS-YOLO 的平均精度均值为 0.647 比 YOLOv3-Tiny 的高 6.5%，同时每秒传输帧数比 YOLOv3-Tiny 高 4 帧。从实验结果可以看出改进后在速度和精度上都比之前的网络有所提升。

第 4 章 基于深度卷积生成对抗网络的数据集扩充

现实环境中可以采集到的缺陷数据较少，利用深度卷积生成对抗网络可以扩充数据集。从采集的数据集中随机抽取部分样本作为深度卷积生成对抗网络的训练集，训练之后可以生成与原始样本特征一致的样本，从而扩充数据集。

4.1 生成对抗网络原理

Ian J. Goodfellow 等人在 2014 年提出 Generative Adversarial Networks(生成对抗网络)，网络模型中主要包含两个部分：生成器(Generator)和判别器(Discriminator)。网络模型接收随机噪声向量之后，通过生成器生成需要的结果，如生成图片、文字等结果[48]。生成器得到的图片和文字等结果之后，利用生成器来判断是否是真实的图片或者文字等。在训练的过程中，生成器会尽可能的生成接近真实图片的图片，从而让判别器无法判断是否是真实的图片。判别器会通过训练，增强判断一个图片是否是真实图片的能力。

假设已知图片的概率分布为$P_{\text{data}}\left(x\right)$，其中 x 表示一个图片。为了尽可能的接近真实图片的概率分布，为生成模型定义一个概率分布为$P_G\left(x,\theta\right)$，θ是这个分布函数的参数，如果$P_G\left(x,\theta\right)$是高斯混合模型，则θ就是高斯分布的平均值和方差。通过调整θ使$P_G\left(x,\theta\right)$和$P_{\text{data}}\left(x\right)$的分布尽可能一致。假设$\left\{\mathrm{x}^1,x^2,\ldots,x^n\right\}$是满足$P_{\text{data}}\left(x\right)$分布的一组数据，则可以得到似然函数如公式(4.1)所示。

$$L=\prod_{i=1}^{n}P_G\left(x^i;\theta\right) \tag{4.1}$$

想要使生成真实照片的概率最大，就等价于让似然函数最大，就是最大似然估计中的问题，找到一个值θ^*让似然函数的值最大。对一个函数取对数不会影响其单调性所以最大化似然函数和最大化 log 似然函数是等价的。根据对数函数的计算规则可以将其装换为每个对数函数的相加。如公式(4.2)所示。

$$\begin{aligned}\theta^{*} &= \arg\max_{\theta}\prod_{i=1}^{n}P_{G}\left(x^{i};\theta\right)\\ &= \arg\max_{\theta}\log\prod_{i=1}^{n}P_{G}\left(x^{i};\theta\right)\\ &= \arg\max_{\theta}\sum_{i=1}^{n}\log P_{G}\left(x^{i};\theta\right)\end{aligned} \tag{4.2}$$

公式(4.2)中求和可以近似为$\log P_{G}\left(x^{i};\theta\right)$的期望，然后可以推导出积分的形式。因为$\int_{x}P_{data}\left(x\right)\log P_{data}\left(x\right)dx$与$\theta$无关，所以在公式上减去此项对计算的结果没有影响，对其进行合并可以得到如公式(4.3)所示。

$$\begin{aligned}\theta^{*} &= \arg\max_{\theta}\sum_{i=1}^{n}\log P_{G}\left(x^{i};\theta\right)\\ &\approx \arg\max_{\theta}E_{x\sim P_{data}}\left[\log P_{G}\left(x;\theta\right)\right]\\ &= \arg\max_{\theta}\int_{x}P_{data}\left(x\right)\log P_{G}\left(x;\theta\right)dx-\int_{x}P_{data}\left(x\right)\log P_{data}\left(x\right)dx\\ &= \arg\max_{\theta}\int_{x}P_{data}\left(x\right)\left(\log P_{G}\left(x;\theta\right)-\log P_{data}\left(x\right)\right)dx\\ &= \arg\max_{\theta}\int_{x}P_{data}\left(x\right)\log\frac{P_{G}\left(x;\theta\right)}{P_{data}\left(x\right)}dx\end{aligned} \tag{4.3}$$

KL(Kullback-Leibler divergence)散度又称相对熵是一种衡量两种概率分布的距离，当两种分布相同时，KL 散度为 0，当两种分布的差异增大时，KL 散度也随之增大。假设$P\left(x\right),Q\left(x\right)$是随机变量 x 的两个概率密度，则 KL 散度的定义如公式(4.4)所示[49]。

$$KL\left(P\|Q\right)=\int P\left(x\right)\log\frac{P\left(x\right)}{Q\left(x\right)}dx \tag{4.4}$$

故上式转换成 KL 散度可以转换为如公式(4.5)所示。

$$\begin{aligned}\theta^{*} &= \arg\max_{\theta}\int_{x}P_{data}\left(x\right)\log\frac{P_{G}\left(x;\theta\right)}{P_{data}\left(x\right)}dx\\ &= \arg\min_{\theta}\int_{x}P_{data}\left(x\right)\log\frac{P_{data}\left(x\right)}{P_{G}\left(x;\theta\right)}dx\\ &= \arg\min_{\theta}KL(P_{data}\|P_{G})\end{aligned} \tag{4.5}$$

由公式(4.5)可知让生成器生成尽可能真实照片的调价就是找一个θ让P_{G}的概率分布尽可能的与P_{data}的概率分布相同。由于P_{G}和P_{data}没有具体的公式，所以可以分布从

P_G 和 P_{data} 中取样，使用判别器来判别两者的差距。定义函数 $V(G,D)$ 如公式(4.6)所示，其中 G 是生成器，D 是判别器用来判断 P_G 和 P_{data} 的差距。

$$\begin{aligned} V(G,D) &= E_{x\sim P_{data}}\left[\log D(x)\right] + E_{x\sim P_z}\left[\log\left(1-D\left(G(z)\right)\right)\right] \\ &= \int_x P_{data}(x)\log\left(D(x)\right)dx + \int_z P_z(z)\log\left(1-D\left(G(z)\right)\right)dz \\ &= \int_x \left[P_{data}(x)\log D(x) + P_G(x)\log\left(1-D(x)\right)\right]dx \end{aligned} \tag{4.6}$$

在给定 G 的前提下，D 取一个合适的值使得 $V(G,D)$ 取到最大值。通过上面的积分，就是在给定 x 的情况下希望可以有一个最优的 D 可以最大化公式(4.7)。

$$P_{data}(x)\log D(x) + P_G(x)\log\left(1-D(x)\right) \tag{4.7}$$

为了表示方便，$P_{data}(x)$ 用 a 表示，$D(x)$ 用 D 表示，$P_G(x)$ 用 b 表示。则公式(4.7)转换为公式(4.8)。

$$\mathrm{f}(D) = a\log(D) + b\log(1-D) \tag{4.8}$$

通过将公式(4.8)求导，令求导的结果为 0 即可求得极值。如下公式(4.9)、公式(4.10)所示。

$$\frac{\mathrm{df}(D)}{dD} = a\times\frac{1}{D} + b\times\frac{1}{1-D}\times(-1) = 0 \tag{4.9}$$

$$a\times\frac{1}{D^*} = b\times\frac{1}{1-D^*} \tag{4.10}$$

从公式(4.10)可以求出 D 的最优值 D^* 如公式(4.11)所示。

$$D^* = \frac{a}{a+b} = \frac{P_{\mathrm{data}}(x)}{P_{\mathrm{data}}(x) + P_G(x)} \tag{4.11}$$

上面得到了在给定 G 的前提下，让 $V(G,D)$ 取到最大值的 D，将得到的 D 代入 $V(G,D)$ 中可以得到公式(4.12)。

$$\begin{aligned}&\max_D V(G,D)\\&=V(G,D^*)\\&=E_{x\sim P_{data}}\left[\log\frac{P_{data}(x)}{P_{data}(x)+P_G(x)}\right]+E_{x\sim P_G}\left[\log\frac{P_G(x)}{P_{data}(x)+P_G(x)}\right]\\&=\int_x P_{data}(x)\log\frac{P_{data}(x)}{\frac{1}{2}\left(P_{data}(x)+P_G(x)\right)}dx\\&+\int_x P_G(x)\log\frac{P_G(x)}{\frac{1}{2}\left(P_{data}(x)+P_G(x)\right)}dx-2\log 2\end{aligned}\tag{4.12}$$

由 KL 散度的定义公式可以将公式(4.12)转换为如下公式(4.13)所示。

$$\begin{aligned}&\max_D V(G,D)\\&=\int_x P_{data}(x)\log\frac{P_{data}(x)}{\frac{1}{2}\left(P_{data}(x)+P_G(x)\right)}dx\\&+\int_x P_G(x)\log\frac{P_G(x)}{\frac{1}{2}\left(P_{data}(x)+P_G(x)\right)}dx-2\log 2\\&=KL\left(P_{data}\,\|\,\frac{P_{data}+P_G}{2}\right)+KL\left(P_G\,\|\,\frac{P_{data}+P_G}{2}\right)-2\log 2\end{aligned}\tag{4.13}$$

JS 散度(Jensen-Shannon divergence)是一种基于 KL 散度的变体，用于衡量两个概率分布的相似度，解决了 KL 散度的非对称问题。定义如公式(4.14)所示[50]。

$$JS(P_1\,\|\,P_2)=\frac{1}{2}KL\left(P_1\,\|\,\frac{P_1+P_2}{2}\right)+\frac{1}{2}KL\left(P_2\,\|\,\frac{P_1+P_2}{2}\right)\tag{4.14}$$

根据 JS 散度定义公式，可以推得 $\max_D V(G,D)$ 如公式(4.15)所示。

$$\begin{aligned}&\max_D V(G,D)\\&=KL\left(P_{data}\,\|\,\frac{P_{data}+P_G}{2}\right)+KL\left(P_G\,\|\,\frac{P_{data}+P_G}{2}\right)-2\log 2\\&=2JSD(P_{data}\|P_G)\end{aligned}\tag{4.15}$$

从上式可以得出，求解最大化 $V(G,D)$ 就是求 P_G 和 P_{data} 的 JS 散度。所以训练判别器的时候，就是通过 P_G 和 P_{data} 中的样本去求这两种样本中的差异。因为最终的目标是找到一个最优的 G 可以让 P_G 和 P_{data} 的差异最小。即如公式(4.16)所示。

$$G^*=\arg\min_G Div(P_G,P_{data})\tag{4.16}$$

由于 P_{data} 和 P_G 之间的散度无法直接计算，但是可以通过判别器来计算两者之间的差异。由公式(4.15)可知 $\max_D V(G,D)$ 近似于 P_{data} 和 P_G 的 JS 散度，所以公式(4.16)可以变为如公式(4.17)所示。

$$G^* = \arg\min_G \max_D V(G,D) \tag{4.17}$$

通过上述的推导，可以得到最后的目标就是找到一个最优的 G 让 $\max_D V(G,D)$ 可以取到最小值。所以训练过程如下所述。给定初始的 G_0 之后，通过求解 $\max_D V(G,D)$，可以得到满足条件的 D_0^*。通过 $V(G,D_0^*)$ 对 θ_G 求微分然后对 G_0 进行更新可以得到 G_1，根据之前的操作，可以求得 D_1^*。之后不断重复上述操作可以不断更新判别器和生成器直到达到要求为止。

4.2 深度卷积生成对抗网络基本原理

传统的生成对抗网络由于网络模型的原因，训练的时候经常不太稳定，生成器经常会产生没有意义的输出。深度卷积生成对抗网络是将生成对抗网络和卷积神经网络结合的成果，极大的提高了模型训练的稳定性和输出的质量。

深度卷积生成对抗网络(Deep Convolutional Generative Adversarial Networks，DCGAN)相比传统生成对抗网络主要有以下几点改进[51]。(1)深度卷积生成对抗网络中取消了所有的池化层，通过改变卷积的步长来实现下采样从而代替池化操作；(2)深度卷积生成对抗网络中使用 global average pooling(全局平均池化)层代替原有的全连接层；(3)深度卷积生成对抗网络在模型中添加 Batch Normalization(批量归一化)，增强了模型的训练速度和稳定性，以及模型的泛化能力[52]。

深度卷积生成对抗网络的生成器模型如图 4.1 所示[53]。

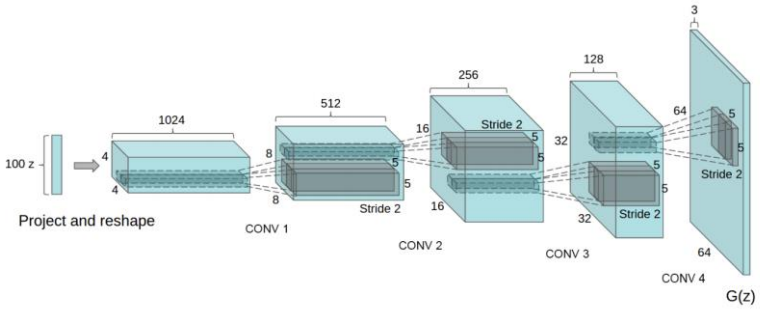


图 4.1 深度卷积生成对抗网络的生成器模型

深度卷积生成对抗网络生成器接收一个随机噪声，通过重塑等操作将随机噪声转换成一个 4×4×1024 的特征图，然后多次利用转置卷积进行上采样操作得到 64×64×3 的图片 G(z)。转置卷积原理如下所述，当 strides(步长)为 1 时，通过 padding(填充)处理扩充原始特征图，然后将卷积核进行水平和垂直翻转得到新的卷积核，利用新的卷积核和扩充之后的特征图进行卷积操作即可得到新的特征图。操作如图 4.2 所示，图中下方的特征图中蓝色的为原始特征图的原色，白色为填充的元素 0，绿色的特征图为通过转置卷积得到的新的特征图[54]。

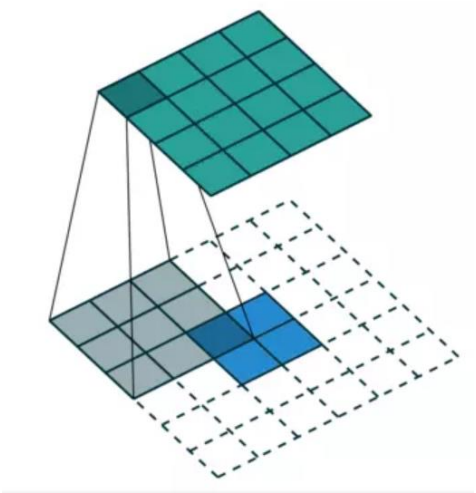


图 4.2 步长为 1 时的转置卷积示意图

当 strides(步长)大于 1 时,首先通过对原始特征图进行插入元素 0 来扩充特征图尺寸，操作如图 4.3 所示，即在原始特征图中每两个元素之间插入元素 0，图中下方的特征图中蓝色的为原始特征图的元素，白色即为填充的元素 0。

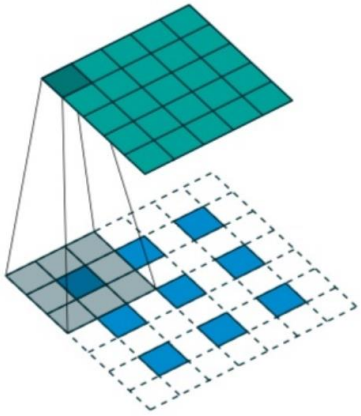


图 4.3 步长为 2 时的转置卷积示意图

通过上述填充操作之后，使用正常的卷积计算即可得到新的特征图。如图 4.3 中原始特征图大小为3×3，通过填充操作之后，使用大小为3×3的卷积核，padding(填充)为 1，strides(步长)为 1，对特征图进行卷积操作即可得到一个5×5的新的特征图。

深度卷积生成对抗网络的判别器模型由多个卷积层构成，判别器的输入为真实图片和生成器生成的图片。假设输入的图片的尺寸为$(64,64,3)$，判别器模型的网络结构如表 4.1 所示[55]。

表 4.1 判别器模型结构

Layer	Stride	Output size	Filters
Input	None	(64,64)	3
Conv	2	(32,32)	64
Batch Normalization+ Leaky Relu	None	(32,32)	64
Conv	2	(16,16)	128
Batch Normalization+ Leaky Relu	None	(16,16)	128
Conv	2	(8,8)	256
Batch Normalization+ Leaky Relu	None	(8,8)	256
Conv	2	(4,4)	512
Batch Normalization+ Leaky Relu	None	(4,4)	512
Reshape+Sigmoid	None	1	1

判别器模型接收到输入的图片之后通过步长为 2 的卷积对图片进行卷积操作，卷积层之后添加 Batch Normalization(批量归一化)层，有效的提高了训练速度，已经训练过程的稳定性[56]。激活函数使用的是带泄露修正线性单元(Leaky Rectified linear unit, Leaky Relu),解决了修正线性单元(Rectified linear unit，ReLU)在输入小于零的时候输出和梯度都为零导致神经元不能更新的问题。通过 4 次步长为 2 的卷积操作之后，特征图的尺寸为(4,4,512),然后通过重塑(reshape)操作将特征图转换为一维向量。上面得到的向量通过 sigmoid 函数得到一个处于 0 到 1 之间的值，用以判断输入图片是正样本还是负样本。

4.3 利用深度卷积生成对抗网络生成墙皮破损缺陷样本

在现实场景当中，由于各种缺陷发生的概率较低，从而导致各种缺陷的数据集也较少，但是基于各种深度学习的模型训练的时候需要大量的数据集。在拥有少量的训练样本时，训练之后无法达到有效的检测精度，本文使用部分训练样本利用深度卷积生成对抗网络生成与训练样本特征基本一致的数据集，从而利用生成的数据集提高目标检测网络的检测精度。

4.3.1 网络模型

城市综合管廊由于长埋于地下，管廊内部比较潮湿，所以管廊墙壁会出现脱落破损等问题。由于缺陷的数据比较少，本文使用深度卷积生成对抗网络利用训练数据集生成特征一致的数据集，从而增强模型的泛化能力。本文所用的深度卷积生成对抗网络模型如图 4.4 所示。

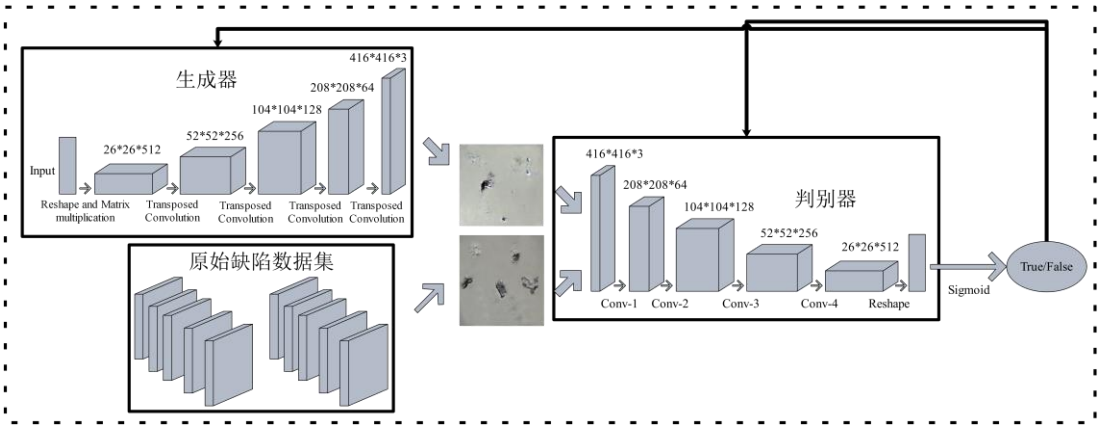


图 4.4 深度卷积生成对抗网络模型

生成器的初始输入是一个长度为 100 的一维向量随机噪声，将其转换成矩阵之后通过一个随机数据的 100×346112 矩阵与其相乘(346112=26×26×512)，将其转换成一个1×346112的矩阵。通过对得到的矩阵进行重塑操作，即可得到26×26×512的矩阵。使用转置卷积将上面得到的特征矩阵进行上采样，既可得到52×52×256的矩阵。模型中用 Batch Normalization(批量归一化)对得到的矩阵进行归一化处理，从而提高模型的训练速度以及稳定性。模型在前几层转置卷积中使用的激活函数是修正线性单元(Rectified Linear Unit, ReLU)。之后利用四次转置卷积等操作，将矩阵转换成图片需要的尺寸416×416×3，每一次转置卷积之后都会添加批量归一化层，前

3 次使用的激活函数是修正线性单元，最后一次转置卷积之后的激活函数是双曲正切（tanh）。生成器模型通过上述的转置卷积等操作之后即可得到所需要生成的图片。

判别器模型的输入有两种，一种是训练集中真实的图片，另一种是生成器生成的图片。判别器通过步长为 2 的卷积层，对图片进行下采样，图片通过卷积操作得到$208\times208\times64$的特征矩阵。卷积层之后添加 Batch Normalization(批量归一化)层，提高了训练速度，以及训练过程的稳定性。激活函数使用的是带泄露修正线性单元(Leaky Rectified linear unit, Leaky Relu)。输入图片通过 4 次卷积操作之后，得到一个$26\times26\times512$的特征矩阵。利用重塑(Reshape)操作，将上面得到的特征矩阵转换为 1 行的矩阵。然后使用 sigmoid 函数得到一个处于 0 与 1 之间的值，用来判断输入的图片是真实的图片还是生成器生成的图片。

4.3.2 损失函数

深度卷积生成对抗网络的目标函数与生成对抗网络的目标函数一致，如公式(4.18)所示。

$$\min_G \max_D V(G,D) = E_{x\sim P_{data}}\left[\log D(x)\right] + E_{x\sim P_z}\left[\log\left(1-D\left(G(z)\right)\right)\right] \tag{4.18}$$

上式中的$D(x)$中的 x 为训练集中真实的图片样本，$D(x)$为真实的图片样本通过判别器得到的值，训练之后$D(x)$的值越接近与 1 则判别器的性能越好。z 为随机给的噪声，$G(z)$为噪声通过生成器生成的图片，当$D\left(G(z)\right)$的值越小则判别器的性能越好。训练时候通过随机给定的生成器，求出让$V(G,D)$可以取得最大值的判别器，然后将求得的判别器代入$V(G,D)$，即可计算出最优的生成器。

深度卷积生成对抗网络的损失函数主要包括生成器的损失函数和判别器的损失函数两个部分。深度卷积生成对抗网络的判别器的损失函数用的是交叉熵损失函数，判别器的损失函数如公式(4.19)所示。

$$loss_D = \frac{1}{n}\sum_{i=1}^{n} -\left(y_i \log\left(D(x_i)\right) + (1-y_i)\log\left(1-D(x_i)\right)\right) \tag{4.19}$$

上式中的x_i为输入到判别器的第 i 个样本，可能为真实样本和生成器生成的样本中的任意一个。$D(x_i)$为第 i 个样本通过判别器得到的结果，y_i为第 i 个样本的真实的标签即此样本为正样本或者正样本。

深度卷积生成对抗网络的生成器的损失函数如公式(4.20)所示。

$$loss_G = \frac{1}{n}\sum_{i=1}^{n} -(1-y_i)\log\left(1-D\left(G\left(z_i\right)\right)\right) \tag{4.20}$$

上式中的z_i为为生成器中输入的第 i 个随机噪声，$G(z_i)$为第 i 个随机噪声通过生成器得到的图片。y_i为第 i 个样本的真实标签，即此样本是正样本或者负样本。

4.3.3 训练优化

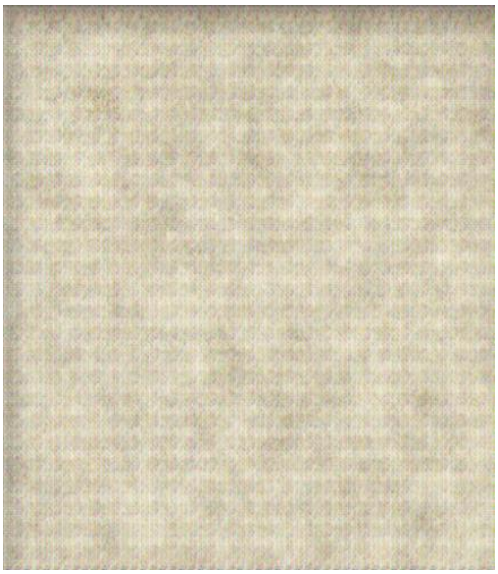
生成器最后一层使用的激活函数是双曲正切（tanh），所以生成器生成的图片的像素值范围是$[-1,1]$，由于生成器生成的图片和训练集的图片都是判别器的输入，所以将训练集的图片的像素值转换到$[-1,1]$与生成器的输出一致。模型中权重使用满足均值为 0 方差为 0.02 的高斯分布的随机数，激活函数 Leaky ReLU 的斜率选择 0.2。

深度卷积生成对抗网络训练的时候优化器(optimizer)使用的是 Adam,学习率(learning rate)为 0.0002，动量(momentum)参数值为 0.5 从而防止震荡和不稳定，从而使训练更加稳定。训练的时候使用小批量(mini batch)来更新参数，本次训练中使用的一个批次是 128 个样本，保证了训练过程中梯度下降的稳定性，同时一个批次的样本相比整个数据集也小了很多，所以参数更新一次时计算量也相对较小。

4.4 实验结果

墙皮破损缺陷数据集中采集的原始缺陷数据有 270 张，从原始数据集中随机选 50 张原始图片作为深度卷积生成对抗网络的训练集，利用深度卷积生成对抗网络生成 1000 张图片用来增加训练集的数量，从而增加目标检测模型的泛化能力。本文基于上述改进后的模型共训练了 10470 次，部分训练集图片和训练后生成的样本图片如图 4.5、图 4.6 所示。

图 4.5 训练集图片之一



(a) 训练 120 次后生成的样本图片

(b) 训练 10470 次后生成的样本图片

图 4.6 部分训练生成样本图片

通过对比训练集图片以及训练后生成的图片，由图中可以看出，深度卷积生成对抗网络生成的样本中含有与训练集中相似的特征信息。综上所述，因为深度卷积生成对抗网络生成的样本与初始训练集样本的特征相似，所以可以用来作为目标检测的训练集从而增强模型的泛化能力。

4.5 本章小结

本章首先介绍了生成对抗网络和深度卷积生成对抗网络的原理，采集的数据集中随机提取一部分作为深度卷积生成对抗网络的训练集，利用深度卷积生成对抗网络生成与原样本特征相似的样本，从而丰富训练数据集，增强模型的性能。

第 5 章 基于 SS-YOLO 算法的管廊缺陷检测实验

本章将改进的算法运用到管廊缺陷的检测当中，首先对采集的图片进行预处理，然后用处理后的图片训练 SS-YOLO。最后使用训练之后的 SS-YOLO 检测测试集中的样本。

5.1 系统环境

本文检测的缺陷主要是针对城市综合管廊中容易出现且影响较大的缺陷，如综合管廊中的水泥表面裂缝和墙皮破损两种缺陷。城市综合管廊内部结构如图 5.1 所示。

图 5.1 城市综合管廊内部结构图

本文所实现的目标检测算法是基于 Keras 和 TensorFlow 框架开发的，实验在 Ubuntu16.04 操作系统下实现。本文所实现的目标检测算法最终需要在城市综合管廊检测机器人上面运行，所以用的 GPU 计算平台是可以在移动机器人上面运行的小觅智能魔方 NV2-TX2，外形如图 5.2 所示。小觅智能魔方 NV2-TX2 是基于 NVIDIA Jeston TX2 主机开发的，随机存取存储器(Random Access Memory，RAM)和显存共享 8GB 内存，具有 1.5T Flops(floating point operations per second, 每秒浮点运算次数)的运算能力。本文使用的摄像头是 USB 高清摄像头，分辨率为1080×720，如图 5.3 所示。在管廊中检测时使用自主移动机器人进行实时数据的采集，本文使用的机

器人如图 5.4 所示，机器人控制器使用的是工控机 X30-j1900，利用上述的摄像头进行视频的采集。

图 5.2 小觅智能魔方 NV2-TX2 外形图

图 5.3 USB 高清摄像头外形图

图 5.4 移动机器人外形图

5.2 数据集以及评价指标

5.2.1 数据集以及图片预处理

本文中水泥表面裂缝缺陷数据集使用的是 github 上面的开源数据集，共有 2022 张图片，其中 1800 张为训练集，剩下的 222 张为测试集。墙皮破损缺陷数据集使用的是采集的原始缺陷数据 270 张，从原始数据集中随机选 50 张原始图片作为深度卷积生成对抗网络的训练集，深度卷积生成对抗网络生成的 1000 张图片加上原始的 50 张图片共 1050 张图片为训练集，220 张原始图片为测试集。水泥表面裂缝缺陷数据集中图片的尺寸为1024×1024，通道数为 3，颜色空间为 RGB。墙皮破损缺陷数据集中图片的尺寸为416×416，通道数为 3，颜色空间为 RGB。

首先使用 LabelMe 对数据集中所有图片完成标注操作，标注出每一张图片中的所有缺陷，然后得到一个 json 文件[57]。标注图片得到的 json 文件中包含了每一个缺陷的左上角、右下角坐标以及缺陷的类别，通过对 json 处理即可得到训练需要的所有数据。

为了提高算法的检测精度，在训练之前，使用数据增强操作对所有输入的图片进行预处理。首先对图片的尺寸进行缩放，在 0.25 到 1.75 的范围内随机生成一个缩放比例，在 0.7 到 1.3 的范围内随机生成一个宽高比，根据随机生成的缩放比例和宽高比对图片进行处理，从而得到一个新的图片。随机生成一个位置坐标，将新得到的图片根据生成的位置坐标粘贴在与原始图片尺寸一样的纯灰色图片之上。粘贴后得到的图片根据随机生成的数据进行水平的位移，以及随机的旋转。最后对通过上述操作之后的图片的颜色空间进行随机的变换，既可得到用于训练的图片。由于图片进行了缩放、平移、翻转等操作，所以标注的位置也发生了改变，根据上面图片的数据增强处理的数据对图片标注的数据进行变换，从而得到训练时图片的标注数据。整体操作流程如图 5.5 所示。

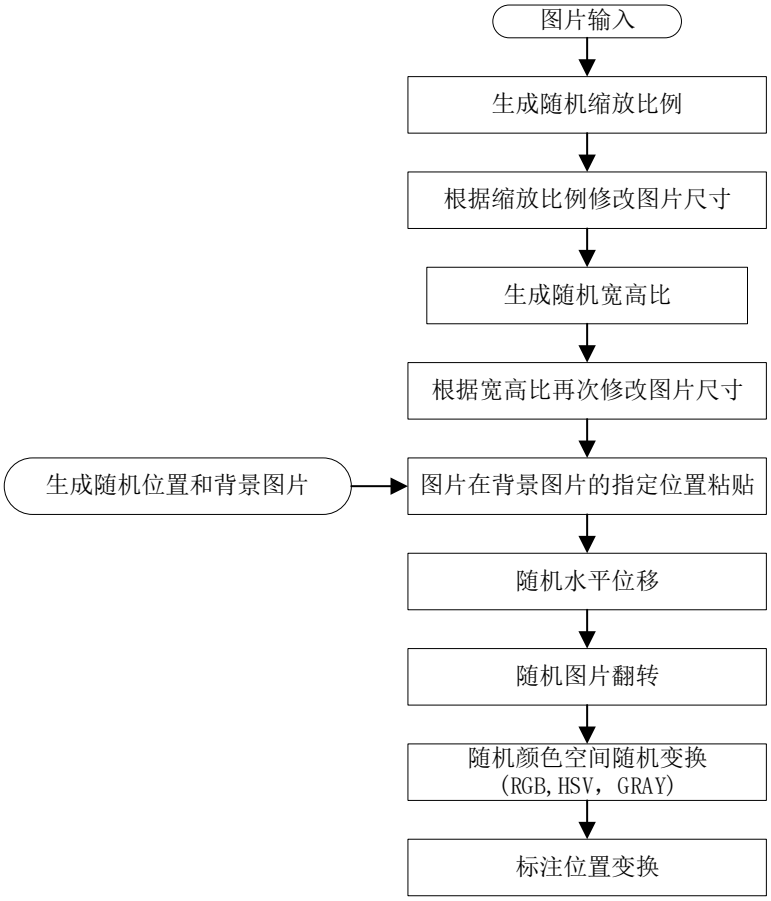
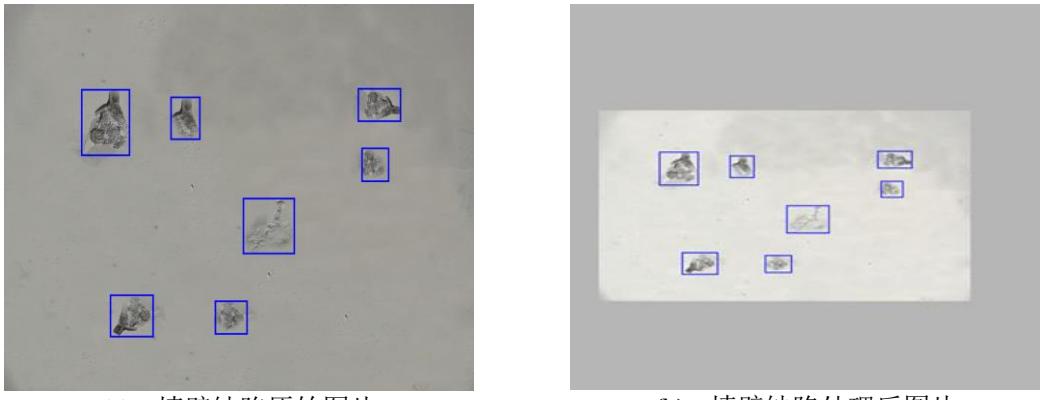


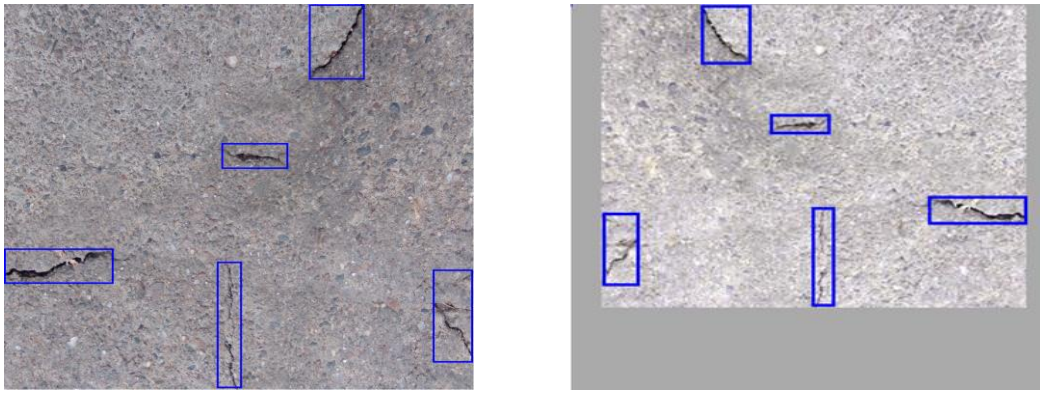
图 5.5 数据增强流程图

随机一个图片通过数据增强处理之后得到的图片如图 5.6、图 5.7 所示。



(a) 墙壁缺陷原始图片 (b) 墙壁缺陷处理后图片

图 5.6 墙壁缺陷原始图片和数据增强处理后图片



(a) 水泥裂缝缺陷原始图片　　(b) 水泥裂缝缺陷处理后图片

图 5.7 水泥裂缝缺陷原始图片和数据增强处理后图片

5.2.2 目标检测性能评价相关指标

目标检测中常用的评价指标有精确率(Precision)、召回率(Recall)、平均精度(Average-Precision,AP)和平均精度均值(Mean Average Precision,MAP)[58]。本文检测的目标有水泥裂缝缺陷和墙皮破损缺陷两种。算法会预测样本的位置，计算每一个样本与对应的真实样本的交并比，同时也会设置一个阈值，当交并比超过了这个阈值的时候这个样本称为正样本，否则这个样本称为负样本。假设 SS-YOLO 算法检测出某一张图片中有水泥裂缝缺陷，如果此时的检测是正确的称这种被正确识别了正样本为 True Positives 简称 TP，如果是墙皮破损缺陷被识别成了水泥裂缝缺陷即负样本被当成了正样本称为 False Positives 简称 FP，如果图片中有水泥裂缝缺陷但是 SS-YOLO 算法将此缺陷识别成了墙皮破损缺陷称为 False Negatives 简称 FN。

精确率(Precision)即所有样本中，被正确检测的正样本占所有预测为正样本的比例，计算如公式(5.1)所示[59]。

$$precision = \frac{TP}{TP + FP} \tag{5.1}$$

召回率(Recall)即所有正样本之中，能正确被识别为正样本的比例，计算如公式(5.2)所示[60]。

$$\text{recall} = \frac{TP}{TP + FN} \tag{5.2}$$

使用训练后的 SS-YOLO 算法检测所有的测试集图片，即可得到所有检测出来的目标以及目标的置信度。将所有检测到的目标按照置信度的大小排序，每一个目标对应的精确率和召回率由当前检测到的目标和排序在其之上的目标的数据计算求

得。通过上述操作即可求得每一个目标对应的精确率和召回率，然后计算当召回率大于 0、0.1、0.2……1 这十一个数时分别对应的精确率的最大值。通过计算十一个精确率的最大值的平均值即可得到平均精度(Average-Precision,AP)，然后对两个类别的平均精度计算均值即可得到平均精度均值(Mean Average Precision,MAP)。

5.3 管廊巡检机器人检测流程

城市综合管廊内遍布各种管线，人工进行检查无法保证人员的生命安全，同时管廊内部空间狭小，不利于人工进行检查。管廊巡检机器人体积较小，可以在各种狭小的空间运动。操作人员在管廊之外远程控制也保证了操作人员的生命安全。

首先在城市综合管廊内采集各种需要检测的缺陷的图片，这些图片作为算法训练的数据集。研究出一种适合检测管廊内缺陷的算法，利用上述数据集作为算法的训练集对算法进行训练。训练后精度达到要求的算法即可转移到巡检机器人上进行管廊内缺陷的检测。

管廊巡检机器人常见的种类有轮式巡检机器人、履带式巡检机器人等。管廊巡检机器人通常配有多种传感器，可以在管廊内自主导航。将上面训练好的检测算法转移到管廊巡检机器人上，然后将管廊巡检机器人置于城市综合管廊之内。管廊巡检机器人在城市综合管廊之内运动的时候，巡检机器人使用摄像头采集到实时的视频，并在 ROS 平台中发布视频话题。NVIDIA Jeston TX2 开发板接收话题，从而获得实时视频数据，并使用 Cv_Bridge 工具将 ROS 图像转换为 opencv 格式的图像，最后用训练好的 SS-YOLO 算法检测视频中的缺陷，如果检测正常则巡检机器人继续运动，如果检测到缺陷则在 ROS 中输出当前缺陷的位置和视频信息。管廊巡检机器人整体检测流程如图 5.8 所示。

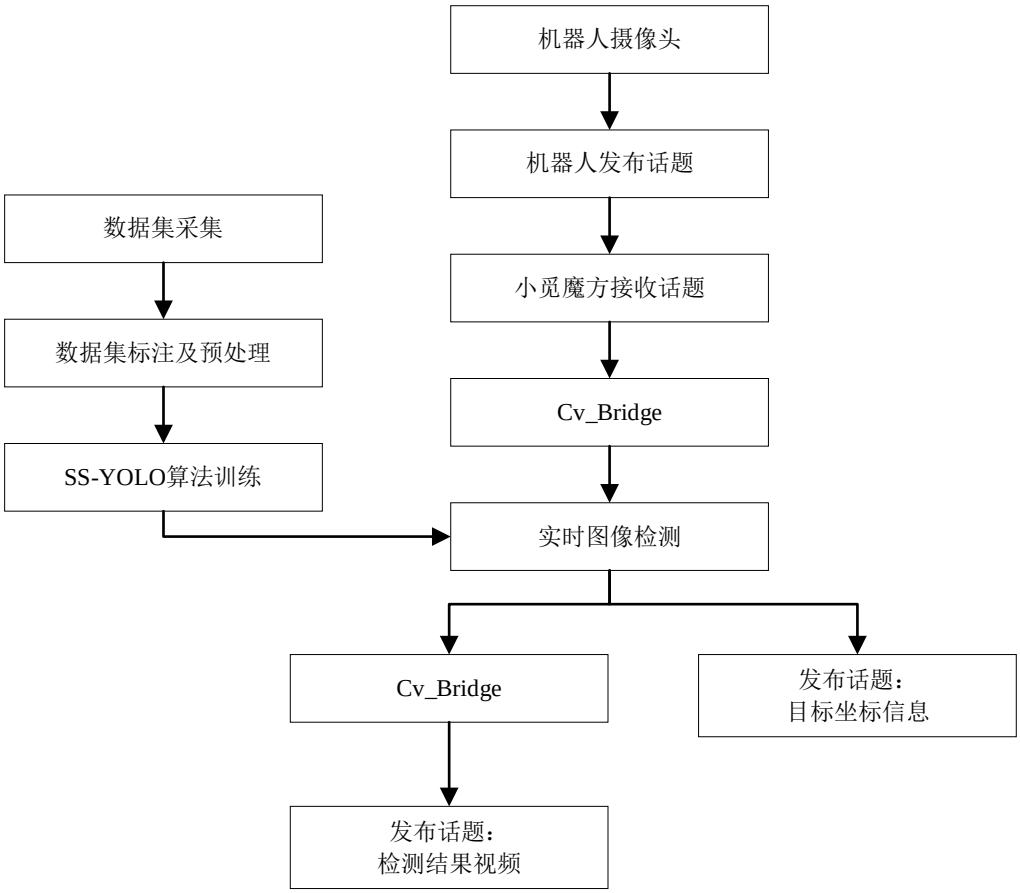


图 5.8 管廊巡检机器人整体检测流程图

5.4 缺陷检测整体流程

本文的整体流程主要包括基于 DCGAN 模型的墙壁缺陷数据生成、数据集标注及预处理、训练及缺陷检测三个部分。整体流程如图 5.9 所示。

首先采集水泥裂缝缺陷和墙皮破损缺陷的样本，由于采集到的墙皮破损缺陷的数据集较少，无法满足训练的需要，所以本文使用深度卷积生成对抗网络生成部分样本。从采集的 270 张样本中，随机抽取 50 个样本用来训练深度卷积生成对抗网络，利用训练好的深度卷积生成对抗网络生成 1000 个样本，用来丰富训练的数据集。采集到的样本和利用深度卷积生成网络生成的样本，使用 LabelMe 进行标注缺陷从而得到训练和测试用的数据集。每一个样本输入到 SS-YOLO 训练的时候，首先通过平移、缩放、旋转等操作进行数据增强处理，然后输入到 SS-YOLO 中。训练后的 SS-YOLO 即可用到管廊巡检机器人上进行缺陷的检测。

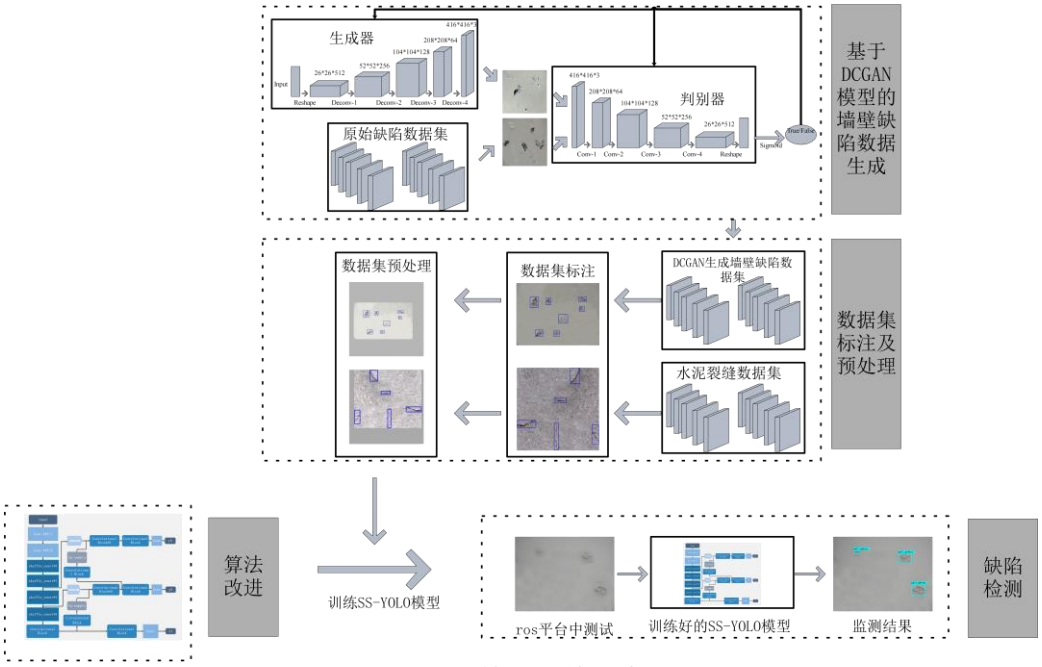
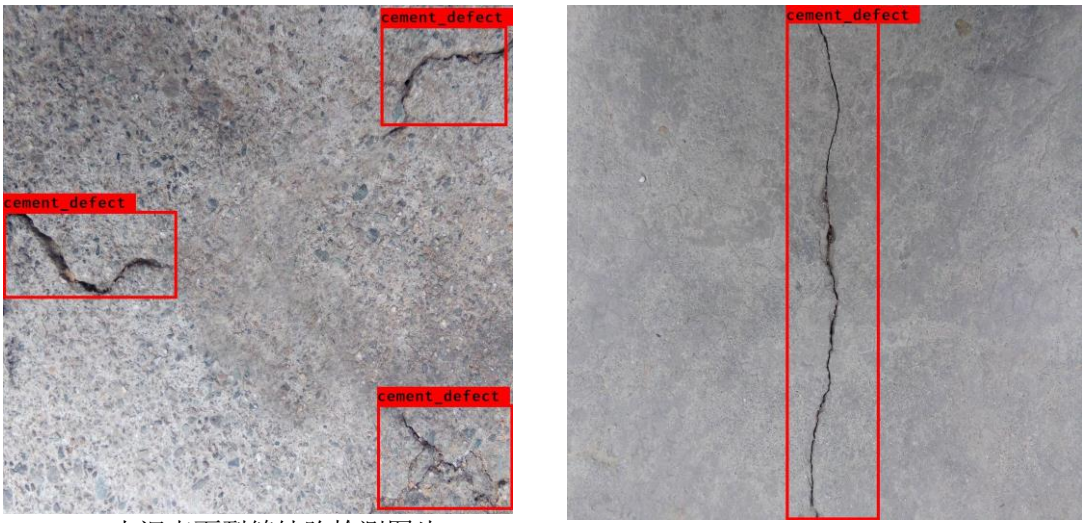


图 5.9 缺陷检测的整体流程图

5.5 实验结果分析

5.5.1 水泥裂缝缺陷检测实验

本文使用了开源数据集中的 1800 张图片为水泥表面裂缝缺陷的训练集，222 张图片为水泥表面裂缝缺陷测试集。SS-YOLO 模型利用上述的训练集进行了充分的训练，利用验证集检测模型的性能以及训练的效果，部分图片检测结果如图 5.10 所示。



(a) 水泥表面裂缝缺陷检测图片 1　　(b) 水泥表面裂缝缺陷检测图片 2

图 5.10 水泥表面裂缝缺陷检测图片

通过检测结果可以看出检测时大部分缺陷都可以检测出来，但是在检测水泥表面裂缝缺陷中较小的缺陷的精度不足。

本文所用的测试集中 222 张图片为水泥表面裂缝缺陷测试集。每张图片内包含的缺陷数量不同，所以经过统计得到每种缺陷的数量如表 5.1 所示。利用被正确检测出的缺陷数量和测试集中缺陷的数量进行计算，即可得到每种缺陷的漏检率如表 5.1 所示。

表 5.1 测试集中缺陷的检测结果

缺陷种类	图片数量	缺陷数量	正确检测出的缺陷数量	漏检率
水泥表面裂缝缺陷	222	541	491	9.2%

本文设置的置信度阈值为 0.5，计算可以求得预测的目标为正样本或者负样本，从而可以计算出每一个类别的精确率和召回率。通过计算求得水泥表面裂缝缺陷中每一个样本对应的精确率，从而绘制出用 SS-YOLO 和 YOLOv3-Tiny 检测的精确率和召回率的曲线如图 5.11、图 5.12 所示。

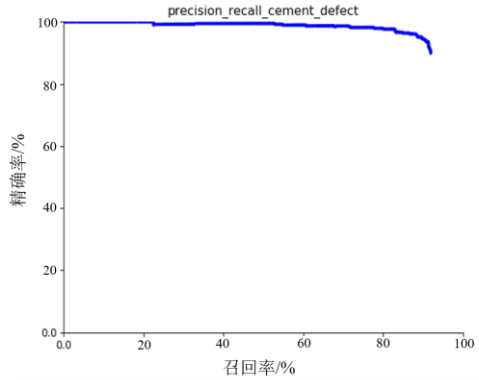


图 5.11 SS-YOLO 检测的水泥表面裂缝缺陷精确率与召回率曲线

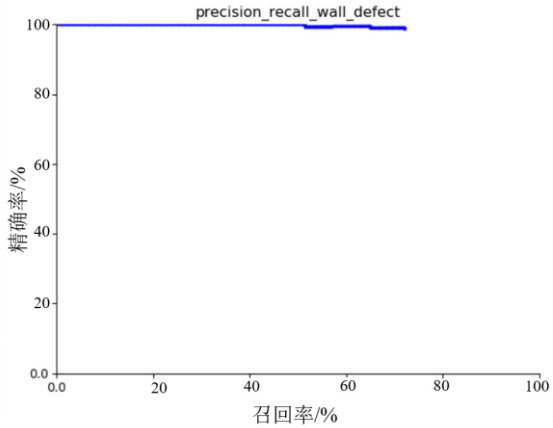


图 5.12 YOLOv3-Tiny 检测的水泥表面裂缝缺陷精确率与召回率曲线

通过上述计算的每一个样本的精确率和召回率既可以得到每一个召回率范围内的精确率的最大值，如表 5.2 所示。

表 5.2 水泥表面裂缝缺陷各个召回率范围对应的精确率最大值

召回率范围	SS-YOLO 精确率最大值	YOLOv3-Tiny 精确率最大值
≥0.0%	100%	100%
≥10%	100%	100%
≥20%	100%	100%
≥30%	99.7%	100%
≥40%	99.7%	100%
≥50%	99.7%	100%
≥60%	99.1%	99.5%
≥70%	98.7%	99.1%
≥80%	97.8%	0.00%
≥90%	94.5%	0.00%
≥100%	0.00%	0.00%

通过上表中准确率的值根据平均精度的公式，可以求得 SS-YOLO 和 YOLOv3-Tiny 水泥表面裂缝缺陷检测的平均精度如公式(5.3)、公式(5.4)所示。

$$\begin{aligned} &AP_{SS-YOLO} \\ &= \frac{1.000+1.000+1.000+0.997+0.997+0.997+0.991+0.987+0.978+0.945+0}{11} \\ &= \frac{9.892}{11} = 0.89 \end{aligned} \tag{5.3}$$

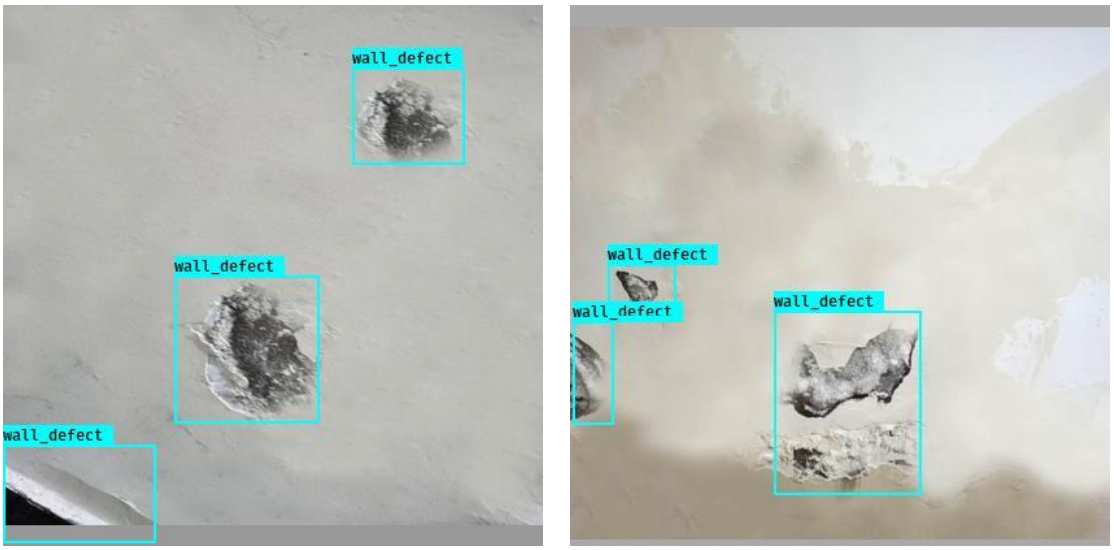
$$\begin{aligned} &AP_{YOLOv3-Tiny} \\ &= \frac{1.000+1.000+1.000+1.000+1.000+1.000+0.995+0.991+0+0+0}{11} \\ &= \frac{7.986}{11} = 0.72 \end{aligned} \tag{5.4}$$

从而可以得到 SS-YOLO 检测水泥表面裂缝缺陷检测的平均精度为 0.89，YOLOv3-Tiny 检测水泥表面裂缝缺陷检测的平均精度为 0.726，由于 YOLOv3-Tiny 中只有两种尺度的检测，所以在检测小目标时的精度较差。

5.5.2 墙皮破损缺陷检测实验

本文在使用机器人上的 USB 高清摄像头采集 270 个样本之后，随机选取 50 个样本作为深度卷积生成对抗网络的训练集，训练后的深度卷积生成对抗网络生成 1000 张图片用来扩充数据集。最后使用 1050 张图片为墙皮破损缺陷的训练集，220 张图片为墙皮破损缺陷的测试集。

SS-YOLO 模型利用上述的训练集进行了充分的训练，利用验证集检测模型的性能以及训练的效果，部分图片检测结果如图 5.13 所示。



(a) 墙皮破损缺陷检测图片 1 (b) 墙皮破损缺陷检测图片 2

图 5.13 墙皮破损缺陷检测图片

通过检测结果可以看出检测时大部分缺陷都可以检测出来，但是检测墙皮破损缺陷时会有部分误检测的情况出现。

在现实环境中测试结果如图 5.14 所示，算法可以正常运行。摄像头发布图像话题之后，NVIDIA Jeston TX2 开发板通过订阅话题可以接收到视频相关数据，使用 SS-YOLO 算法可以检测出视频中的缺陷，并将检测视频和位置信息单独通过话题发布在 ROS 平台中，只需订阅相关话题即可获得对应的数据。

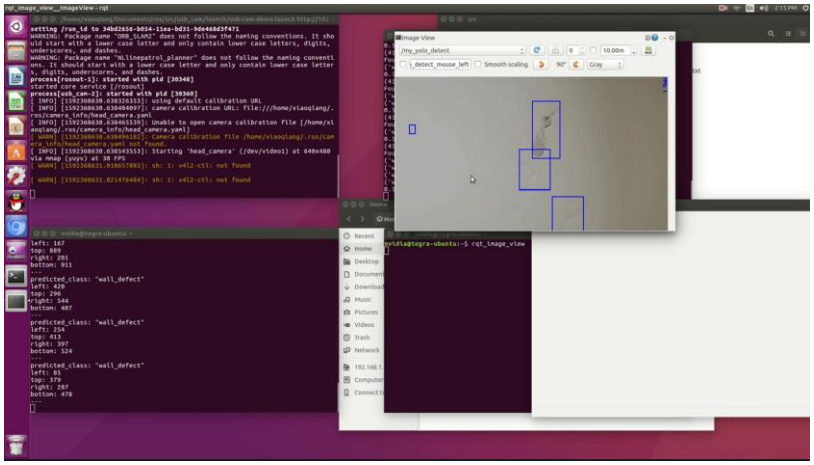


图 5.14 算法在 ROS 平台中测试结果

本文所用的测试集中 220 张图片为墙皮破损缺陷的测试集。每张图片内包含的缺陷数量不同，经过统计得到缺陷的数量如表 5.3 所示。利用被正确检测出的缺陷数量和测试集中缺陷的数量进行计算，即可得到缺陷的漏检率如表 5.3 所示。

表 5.3 测试集中缺陷的检测结果

缺陷种类	图片数量	缺陷数量	正确检测出的缺陷数量	漏检率
墙皮破损缺陷	220	531	458	13.7%

通过计算求得墙皮破损缺陷中每一个样本对应的精确率，从而绘制出精确率和召回率的曲线如图 5.15、图 5.16 所示。

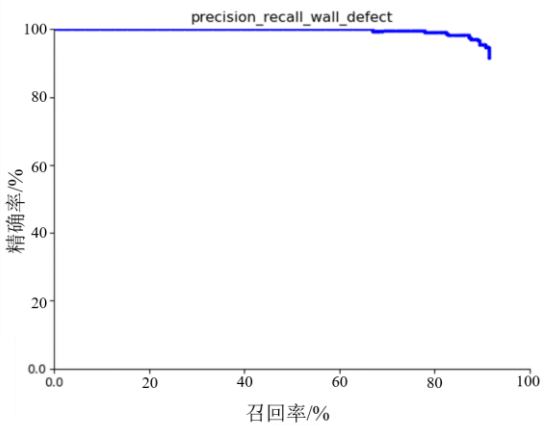


图 5.15 SS-YOLO 墙皮破损缺陷精确率与召回率曲线

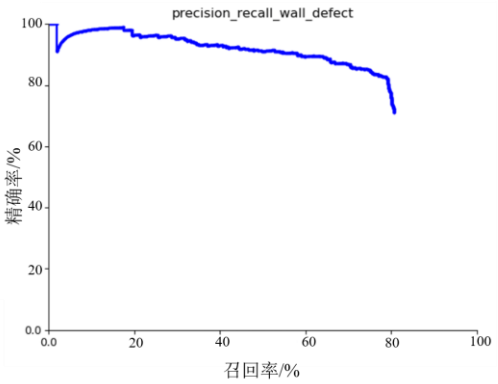


图 5.16 YOLOv3-Tiny 墙皮破损缺陷精确率与召回率曲线

通过上面计算得到的所有样本的精确率和召回率数据，可以得到墙皮破损缺陷检测中各个召回率范围内对应的精确率最大值，如表 5.4 所示。

表 5.4 墙皮破损缺陷各个召回率范围对应的精确率最大值

召回率范围	SS-YOLO 精确率最大值	YOLOv3-Tiny 精确率最大值
≥0.0%	100%	100%
≥10%	100%	98.9%
≥20%	100%	96.5%
≥30%	100%	95.4%
≥40%	100%	93.1%
≥50%	100%	91.5%
≥60%	100%	89.6%
≥70%	99.5%	86.9%
≥80%	99.1%	75.8%
≥90%	95.5%	0.00%
≥100%	0.00%	0.00%

通过上表得到的召回率在各个范围内精确率的最大值，可以求出墙皮破损缺陷检测的平均精度，计算如公式(5.5)、公式(5.6)所示。

$$\begin{aligned} AP_{SS-YOLO} &= \frac{1.000+1.000+1.000+1.000+1.000+1.000+1.000+0.995+0.991+0.955+0}{11} \\ &= \frac{9.941}{11} = 0.90 \end{aligned} \tag{5.5}$$

$$\begin{aligned} &AP_{YOLOv3-Tiny} \\ &=\frac{1.000+0.989+0.965+0.954+0.931+0.915+0.896+0.869+0.758+0+0}{11} \\ &=\frac{8.277}{11}=0.75 \end{aligned} \tag{5.6}$$

通过计算得到的水泥裂缝缺陷检测和墙皮破损缺陷检测的平均精度根据公式可以得到检测的平均精度均值如公式(5.7)、公式(5.8)所示。

$$MAP_{SS-YOLO}=\frac{0.89+0.90}{2}=0.895 \tag{5.7}$$

$$MAP_{YOLOv3\text{-}Tiny}=\frac{0.72+0.75}{2}=0.735 \tag{5.8}$$

综上所示可以得到 SS-YOLO 检测时的精度和速度如表 5.5 所示。

表 5.5 SS-YOLO 检测的速度与精度数据

算法种类	平均精度均值(MAP)	每秒传输帧数 (FPS)
SS-YOLO	0.895	10
YOLOv3-Tiny	0.735	8

由上表的数据可以看出 YOLOv3-Tiny 由于只有两种尺度的检测，所以对小目标的检测精度不够，整体的检测精度较低无法满足检测的需要。SS-YOLO 比 YOLOv3-Tiny 多一个用来检测小目标的52×52尺度的输出，所以检测的精度高出很多，可以满足检测的需要。同时 SS-YOLO 由于特征提取网络的改进，检测的速度也比 YOLOv3-Tiny 快。综上所述，SS-YOLO 检测的整体性能比 YOLOv3-Tiny 优异很多，可以满足管廊巡检机器人检测的需要。

5.6 本章小结

本章将 SS-YOLO 算法在实际的数据集中进行验证和测试，首先介绍了系统环境、训练用的数据集和相关的性能评价指标。由实验结果可知，SS-YOLO 的平均精度均值为 0.895 比 YOLOv3-Tiny 的平均精度均值 0.735 高，满足检测的需要，可以在管廊检测中使用。

第 6 章 总结与展望

6.1 总结

现如今大部分城市拥有城市综合管廊，城市综合管廊深埋地下，良好的设计可以节省更多空间。城市综合管廊在空间利用、管理、检修、安全等方面拥有无可比拟的优势，但同时大量的城市综合管廊的安全问题也影响重大。传统的巡检人工成本较高，检测的准确率不容易保证。随着目标检测算法的发展，使用目标检测算法检测管廊的安全问题可以降低人工的成本、保证检测的准确性同时可以 24 小时不间断的检测。本文使用深度卷积生成对抗网络生成样本从而丰富训练数据集，参考了 ShuffleNet v2 中的各种提高速度的思想以及 SENet 中的利用在特征通道维度上进行重标定从而提高精度的思路，提出了一个新的网络结构 Shuffle_Senet 模块，基于此模块搭建了特征提取网络。在借鉴了 YOLOv3-Tiny 的多尺度预测的思想，改进了其获取先验框的方式，结合上述特征提取网络得到了新的目标检测算法 SS-YOLO。本文主要的研究工作内容如下所示。

1. 对城市综合管廊检测的背景和意义进行讨论，并介绍了管廊检测和目标检测的研究现状。

2. 总结了目标检测的相关的理论基础，分别介绍了传统目标检测算法、基于深度学习的两阶段的目标检测算法、基于深度学习的单阶段的目标检测算法的原理以及检测的主要流程。

3. 借鉴了 ShuffleNet v2 中各种提高模型速度的技巧和 SENet 中利用在特征通道维度对特征图进行重标定从而提高检测的精度的思路，提出了新的网络结构 Shuffle_Senet 模块。在借鉴了 YOLOv3-Tiny 的多尺度预测的思想的基础上，利用上述的 Shuffle_Senet 模块搭建特征提取网络，改进获取先验框的方式，提出了一种新的目标检测算法 SS-YOLO。SS-YOLO 在 VOC 数据集中检测时的平均精度均值比 YOLOv3-Tiny 高 6.5%，每秒传输帧数高 4 帧。

4. 现实环境当中，缺陷数据集的数量通常比较少，所以本文在采集缺陷样本的基础上，使用部分采集的样本作为深度卷积生成对抗网络的训练集，利用深度卷积

生成对抗网络生成与原始样本特征相似的样本，用来丰富训练数据集，从而增加目标检测算法的性能。

5. 使用标注好的管廊相关的数据集对 SS-YOLO 算法进行训练，训练后检测的平均精度均值为 0.895 高于 YOLOv3-Tiny 的 0.735。训练后的 SS-YOLO 在巡检机器人上可以正常运行，成功的检测出目标。

综上所述，本文提出了一种新的目标检测算法 SS-YOLO，相对于原始的算法在速度和精度上都有所提升，同时可以在城市综合管廊检测中使用。

6.2 展望

基于现有的研究成果，仍有需要进一步研究的地方，主要由以下几点。

1.数据集的丰富。本文中使用的两种缺陷数据集为管廊中常见的缺陷，但是有很多比较罕见的缺陷也对管廊的安全有威胁，因此应该尽可能多的搜集到管廊中可能出现的各种缺陷以丰富训练集，从而使训练后的目标检测算法可以检测到管廊中可能出现的各种安全问题。

2.深度卷积生成对抗网络训练的时候需要的样本数量不是特别少，如果在缺陷样本只有几张图片的时候无法训练出需要的样本，因此下一步应该对深度卷积生成对抗网络的结构进行改进，从而得到一种基于一张或者几张图片就可以生成样本的生成对抗网络。

参考文献

[1] 杨颖. 城市地下综合管廊监控系统研究及应用[J]. 昆明冶金高等专科学校学报, 2017, 33(03): 64-70.

[2] 郭佳奇, 钱源, 王珍珍, et al. 城市地下综合管廊常见运维灾害及对策研究[J]. 灾害学, 2019, 34(01): 27-33.

[3] 谭忠盛, 陈雪莹, 王秀英, et al. 城市地下综合管廊建设管理模式及关键技术[J]. 隧道建设, 2016, 36(10): 1177-1189.

[4] 樊锦仁, 夏远洋. 湿陷性黄土地基综合管廊的设计探讨[J]. 特种结构, 2019, 36(01): 40-43+48.

[5] Bright G., Ferreira D., Mayor R. Automated pipe inspection robot[J]. Industrial Robot An International Journal, 1997, 24(4): 285-289.

[6] 周亮. 城市地下综合管廊安全监测系统建设关键技术研究[J]. 现代测绘, 2016, 39(06): 39-41.

[7] Gabriele P., Thilo w. BIM im Verkehrswasserbau[J]. Bautechnik, 2017, 94(8): 509-513.

[8] Al-Zwainy F., Mohammed I., Al-Shaikhli K. Diagnostic and Assessment Benefits and Barriers of BIM in Construction Project Management[J]. Civil Engineering Journal, 2017, 3(1): 63-77.

[9] Zhi S., Mehmet C., Mznah A., et al. MISE-PIPE: Magnetic induction-based wireless sensor networks for underground pipeline monitoring[J]. Ad Hoc Networks, 2011, 9(3): 218-227.

[10] Mueid R., Ahmed C., Ahad M. Pedestrian activity classification using patterns of motion and histogram of oriented gradient[J]. Journal On Multimodal User Interfaces, 2016, 10(4): 299-305.

[11] By Z., Huiyu Z., Yuan Y., et al. Object tracking using SIFT features and mean shift[J]. Computer Vision and Image Understanding, 2009, 113(3): 345-352.

[12] Munir A., Shabib A., Muhammad S. SVM Optimization for Sentiment Analysis[J]. International Journal of Advanced Computer Science and Applications, 2018, 9(4): 6.

[13] Liu X., Kan M., Wanglong W., et al. VIPLFaceNet: an open source deep face recognition SDK[J]. Frontiers Of Computer Science, 2017, 11(2): 208-218.

[14] Girshick R., Donahue J., Darrell T., et al. Rich feature hierarchies for accurate object detection and semantic segmentation[C]// 2014 IEEE Conference on Computer Vision and Pattern Recognition. Columbus, OH, USA: IEEE Press, 2014: 580-587.

[15] Kaiming H., Xiangyu Z., Shaoqing R., 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.

[16] Zhitao X., Lei P., Lei G. Surface Parameter Measurement of Braided Composite Preform Based on Faster R-CNN[J]. Fibers and Polymers, 2020, 21(3): 590-603.

[17] Shaoqing R., Kaiming H., Ross G., et al. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks[J]. IEEE Transactions On Pattern Analysis And Machine Intelligence, 2017, 39(6): 1137-1149.

[18] Redmon J., Divvala S., Girshick R., et al. You Only Look Once: Unified, Real-Time Object Detection[C]// 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Las Vegas, NV, USA: IEEE Press, 2016: 779 - 788.

[19] Liu W., Anguelov D., Erhan D., et al. SSD: Single Shot Multibox Detector[C]// 2016 European conference on computer vision. Amsterdam: Springer Press, 2016: 21-37.

[20] Zhang X., Dong X., Wei Q., et al. Real-time object detection algorithm based on improved YOLOv3[J]. Journal Of Electronic Imaging, 2019, 28(5): 1-8.

[21] Yujie L., Yu D., Xiaoming C., et al. TOP-SIFT: the selected SIFT descriptor based on dictionary learning[J]. Visual Computer, 2019, 35(5): 667-677.

[22] 张恒, 刘大勇, 刘艳丽, et al. 融合深度信息的 BRISK 改进算法[J]. 计算机应用, 2015, 35(08): 2285-2290.

[23] Sheng-hua Z., Yan L., Qing-cai C. Visual orientation inhomogeneity based scale-invariant feature transform[J]. Expert Systems With Applications, 2015, 42(13): 5658-5667.

[24] 穆柯楠, 赵祥模, 惠飞. 基于非采样高斯差分金字塔的多尺度融合边缘检测[J]. 四川大学学报(工程科学版), 2015, 47(05): 130-138.

[25] Pedro H., Rien Q. Quasi-Newton Jacobian and Hessian Updates for Pseudospectral based NMPC[J]. IFAC PapersOnLine, 2018, 51(20): 22-27.

[26] Liu C., Yuen J., Torralba A. SIFT flow: dense correspondence across scenes and its applications[J]. IEEE Transactions On Pattern Analysis And Machine Intelligence, 2011, 33(5): 978-994.

[27] Wan-Lei Z., Chong N. Flip-invariant SIFT for copy and object detection[J]. IEEE Transactions On Image Processing, 2013, 22(3): 980-991.

[28] Jiannan L., Xiaodan L., Shengmei S., et al. Scale-Aware Fast R-CNN for Pedestrian Detection[J]. IEEE Transactions on Multimedia, 2018, 20(4): 985-996.

[29] Xudong S., Pengcheng W., Hoi. Face Detection using Deep Learning:An Improved Faster RCNN Approach[J]. Neuro Computing, 2018, 299(1): 42-50.

[30] Shanthi T., Sabeenian. Modified Alexnet architecture for classification of diabetic retinopathy images[J]. Computers And Electrical Engineering, 2019, 76(1): 56-64.

[31] Aravind K. R., Raja P. Disease Classification in Eggplant Using Pre-trained VGG16 and MSVM[J]. Scientific Reports, 2020, 10(1): 1419-1460.

[32] Khan R. U., Zhang X., Kumar R. Analysis of ResNet and GoogleNet models for malware detection[J]. Journal of Computer Virology and Hacking Techniques, 2019, 15(1): 29-37.

[33] Xiaobo L., Huaiyuan W., Hongdi J., et al. Research on Intelligent Identification of Rock Types Based on Faster R-CNN Method[J]. IEEE Access, 2020, 8(1): 21804-21812.

[34] Ran L., Xiangrui Z., Stephanie E., et al. Automatic localization and identification of mitochondria in cellular electron cryo-tomography using faster-RCNN[J]. Bmc Bioinformatics, 2019, 20(3): 75-85.

[35] 高鑫, 李慧, 张义, et al. 基于可变形卷积神经网络的遥感影像密集区域车辆检测方法[J]. 电子与信息学报, 2018, 040(12): 2812-2819.

[36] 汪海波, 陈雁翔, 李艳秋. 基于主成分分析和 Softmax 回归模型的人脸识别方法[J]. 合肥工业大学学报(自然科学版), 2015, 38(06): 759-763.

[37] 胡炎, 单子力, 高峰. 基于 Faster-RCNN 和多分辨率 SAR 的海上舰船目标检测[J]. 无线电工程, 2018, 48(02): 96-100.

[38] 张汇, 杜煜, 宁淑荣, et al. 基于 Faster RCNN 的行人检测方法[J]. 传感器与微系统, 2019, 38(02): 147-149.

[39] Jiaxing L., Dexiang Z., Jingjing Z., et al. Facial Expression Recognition with Faster R-CNN[J]. Procedia Computer Science, 2017, 107(1): 135-140.

[40] 沈军宇, 李林燕, 夏振平. 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.

[41] 李鹏飞, 刘瑶, 李珣, et al. YOLO9000 模型的车辆多目标视频检测系统研究[J]. 计算机测量与控制, 2019, 27(08): 21-24+29.

[42] 王超, 付子昂. 基于 YOLO v2 模型的交通标识检测算法[J]. 计算机应用, 2018, 38(A2): 276-278.

[43] Wang L., Yang S., Yang S., et al. Automatic thyroid nodule recognition and diagnosis in ultrasound imaging with the YOLOv2 neural network[J]. World Journal Of Surgical Oncology, 2019, 17(1): 1-9.

[44] 宋建国, 吴岳. 基于 YOLOv2 模型的道路目标检测改进算法[J]. 软件导刊, 2019, 18(12): 126-129.

[45] 王琳, 卫晨, 李伟山, et al. 结合金字塔池化模块的 YOLOv2 的井下行人检测[J]. 计算机工程与应用, 2019, 55(3): 133-139.

[46] 寇大磊, 权冀川, 张仲伟. 基于深度学习的目标检测框架进展研究[J]. 计算机工程与应用, 2019, 55(11): 25-34.

[47] 魏湧明, 全吉成, 侯宇青阳. 基于 YOLOv2 的无人机航拍图像定位研究[J]. 激光与光电子学进展, 2017, 54(11): 101-110.

[48] Liu J., Wang C., Su H., et al. Multistage GAN for Fabric Defect Detection[J]. IEEE Transactions on Image Processing, 2020, 29(1): 3388-3400.

[49] Popescu P. G., Dragomir S., Sluşanschi E. Bounds for kullback-leibler divergence[J]. Electronic Journal of Differential Equations, 2016, 2016(129-324): 1-6.

[50] Joshi R., Kumar S. A dissimilarity measure based on Jensen Shannon divergence measure[J]. International Journal of General Systems, 2019, 48(3): 280-301.

[51] Sagawa Y., Hagiwara M. Face Image Generation System using Attributes Information with DCGANs[J]. Transactions of Japan Society of Kansei Engineering, 2018, 17(3): 347-355.

[52] Fang W., Ding Y., Zhang F., et al. Gesture recognition based on CNN and DCGAN for calculation and text output[J]. IEEE Access, 2019, 7(1): 28230-28237.

[53] Iliyasu A., Deng H. Semi-Supervised Encrypted Traffic Classification With Deep Convolutional Generative Adversarial Networks[J]. IEEE Access, 2020, 8(1): 118-126.

[54] Yang X., Mei H., Zhang J., et al. DRFN: Deep Recurrent Fusion Network for Single-Image Super-Resolution With Large Factors[J]. IEEE Transactions on Multimedia, 2019, 21(2): 328-337.

[55] 蔡晓龙. 基于 DCGAN 算法的图像生成技术研究[D]. 青岛: 青岛理工大学, 2018.

[56] 张澎，崔梦天，谢琪, et al. 基于深度卷积生成对抗网络的植物图像识别方法的研究[J]. 西南民族大学学报(自然科学版), 2019, 45(02): 185-191.

[57] Russell B., Torralba A., Murphy K., et al. LabelMe: A Database and Web-Based Tool for Image Annotation[J]. International Journal of Computer Vision, 2008, 77(1-3): 157-173.

[58] 任培铭. 基于 YOLO 的实时目标检测方法研究[D]. 无锡: 江南大学, 2019.

[59] Qayyum A., Anwar S., Awais M., et al. Medical image retrieval using deep convolutional neural network[J]. Neurocomputing, 2017, 266(1): 8-20.

[60] 郭祥云，胡敏，王文胜, et al. 基于深度学习的非结构环境下海参实时识别算法[J]. 北京信息科技大学学报(自然科学版), 2019, 34(03): 27-31.

致谢

年轮流转，时光飞逝，转眼间已经度过了三年，研究生的三年让我得到了前所未有的成长。我不仅学习到了受用终生的知识，也提高了自己的学习方法和思维方式。在这三年里，很多人在学习和生活中给予了我各种帮助，在这里向他们表示真诚的感谢。

感谢我的导师李老师，从研一开始帮助了解各种课题，然后帮忙进行各种课题的可行性分析，到后来研究课题时帮助我开拓研究思路。进入学校之后的每一个阶段，李老师都悉心指导，让我在三年的学习生涯中受益良多，有了较大的进步。李老师渊博的知识、创新的精神以及严谨的科研态度让我不仅学习到了知识也学习到了学习方法和态度。在学习中遇到问题的时候，李老师总是耐心的指导，在此对李老师的培养表示最衷心的感谢。

感谢团队的负责人王平院长，为我们提供了一个良好的学习环境和平台，同时通过各种讲座和课堂向我们传授先进的技术。让我们对科研有了进一步的认知，在此向他表示最崇高的敬意。

感谢实验室的同学们，在学习和生活当中遇到困难的时候，实验室的同学们总会站出来帮忙解决困难，在解决困难的过程中一起学习一起成长。有了这些同学的帮助，我在研究生期间一次次成长，受益匪浅。你们的陪伴和帮助是我今生的财富。

感谢我的家人在生活上给予的帮助和鼓励，因为他们的鼓励有了奋斗的动力。

最后，向从百忙之中抽出宝贵时间的各位评审老师致以崇高的敬意和衷心的感谢。

攻读硕士学位期间从事的科研工作及取得的成果

参与科研项目：

[1] 自主移动检测机器人系统研发及其在城市地下综合管廊中的现场应用(2018-0504)，重庆市建设科技计划项目，2018.7 – 2020.7 .

发表及完成论文：

[1] Yong Li,**Can LV**. SS-YOLO: An Object Detection Algorithm based on YOLOv3 and ShuffleNet [C]//2020 IEEE 4th Information Technology,Networking, Electronic and Automation Control Conference (ITNEC). Chongqing:IEEE Press.2020: 769-772.