**摘 要**

在当今全面放开的大背景下，佩戴口罩已经几乎成为了保护个人、家庭和社会最重要的举措之一。实验研究表明，在公共场合佩戴口罩可以大大降低飞沫传播途径，可以说，不论是在工厂生产中的卫生标准，还是在个人防护中的安全措施，都需要口罩发挥其积极作用。但是在实际生活中，仍存在许多没有认识到口罩重要作用或根本不在意的人员；且目前只有少数的机构注意到了口罩检测方面的相关研究，不同方案的识别效果在密集人流的检测上，表现参差不齐。

本文选择结合了R-CNN精度和Yolo速度优点的SSD算法作为主要研究模型并利用更具针对性的PyramidBox算法进行迁移学习，从而更好地实现人脸检测和口罩识别任务。本文基于文献研究，尝试利用Pytorch框架复现SSD模型，在此基础上完成人脸口罩数据集的收集、数据的预处理和图像增强、模型的训练及评估、改进模型训练方法等工作，以期得到准确性较高、鲁棒性较好，泛化能力较强的口罩检测模型；除此之外，本文利用PaddleHub框架在PyramidBox算法的基础上进行Fine-tune，更好的适应实际场景下的需求；最后，在得到效果较好的人脸检测与口罩识别检测器后，为了项目的完整落地，我们开发了一套完整的线上口罩识别系统。利用flask框架搭建服务器，使用户可以方便的进行本地推理。除此之外，本设计还将模型部署到了手机等端侧设备，通过安卓开发apk的方式更好的对模型进行实际场景的应用。

关键词：目标检测；单发多尺度；PyramidBox；小目标场景；上下文学习；特征融合

**目 录**

[1 绪论 1](#_Toc26016)

[1.1 研究背景与意义 1](#_Toc16223)

[1.1.1 项目研究背景 1](#_Toc17406)

[1.1.2 项目研究意义 1](#_Toc23427)

[1.2 国内外研究现状 2](#_Toc21501)

[1.2.1 传统目标检测方法 2](#_Toc24544)

[1.2.2 基于深度学习的目标检测 3](#_Toc1264)

[1.2.3 改进的目标检测器 4](#_Toc390)

[1.3 论文主要研究内容 5](#_Toc13985)

[1.4 论文组织结构 5](#_Toc3109)

[2 相关理论基础 7](#_Toc3278)

[2.1 卷积神经网络 7](#_Toc10280)

[2.2 常用的目标检测器 8](#_Toc23371)

[2.2.1 Faster R-CNN 8](#_Toc23235)

[2.2.2 YOLO 10](#_Toc22628)

[2.3 单发多框检测(SSD) 11](#_Toc22356)

[2.3.1 SSD网络架构 11](#_Toc28479)

[2.3.2 SSD训练过程 12](#_Toc13498)

[2.3.3 与其它模型的比较与改进 13](#_Toc6726)

[2.4 Pyramidbox 人脸检测 15](#_Toc28315)

[2.4.1 Pyramidbox网络结构 15](#_Toc11744)

[2.4.2 Pyramidbox性能分析 17](#_Toc6578)

[3 算法设计 18](#_Toc32265)

[3.1 网络搭建 18](#_Toc11153)

[3.1.1 高宽减半块 18](#_Toc23937)

[3.1.2 基本网络块 19](#_Toc31108)

[3.1.3 不同尺度的连接 20](#_Toc11186)

[3.1.4 类别预测层和回归预测层 20](#_Toc28532)

[3.1.5 完整网络结构 21](#_Toc3705)

[3.2 数据预处理 22](#_Toc18172)

[3.2.1 图像导入 22](#_Toc8047)

[3.2.2 图像尺寸 23](#_Toc2063)

[3.2.3 图像增强 24](#_Toc20597)

[3.3 模型训练 25](#_Toc10206)

[3.4 数据可视化 27](#_Toc10629)

[4 系统实现 29](#_Toc32734)

[4.1 前端实现 29](#_Toc21011)

[4.1.1 图像传输 30](#_Toc26150)

[4.1.2 视频传输 31](#_Toc10890)

[4.1.3 摄像头传输 31](#_Toc14707)

[4.2 后端实现 33](#_Toc7852)

[4.2.1 接收图片文件 33](#_Toc1684)

[4.2.2 接收视频文件 33](#_Toc24532)

[4.2.3 接收摄像头文件 34](#_Toc23064)

[5 系统测试 35](#_Toc28272)

[5.1 图片检测 35](#_Toc10344)

[5.2 视频检测 36](#_Toc21553)

[5.3 摄像头实时检测 37](#_Toc18358)

[6 安卓部署 38](#_Toc15545)

[6.1 部署流程 38](#_Toc4715)

[6.2 功能实现 38](#_Toc18648)

[6.2.1 界面实现 38](#_Toc15862)

[6.2.2 逻辑实现 39](#_Toc6297)

[6.2.3 打包为apk 39](#_Toc4398)

[6.3 结果展示 40](#_Toc10912)

[7 总结与展望 41](#_Toc16037)

# 绪论

## 研究背景与意义

### 项目研究背景

在硬件算力显著增长、数字信息愈发庞大，计算机技术日新月异的时代背景下，目标检测、OCR等人工智能技术越来越多地应用到了我们生活的方方面面，为我们提供便捷。在全社会众志成城的努力下，抗击疫情取得了巨大胜利，在这场“战役”中，人工智能和大数据技术发挥了巨大作用。随着疫情新阶段的到来，我们仍然可以运用人工智能相关知识继续为社会发展和居民生活带来便利。

“控制传染源、切断传播途径和排查易感人群”是打赢抗疫的三种手段。在当今全面放开的大背景下，佩戴口罩已经几乎成为了保护个人、家庭和社会最重要的举措之一。虽然抗疫已经取得了巨大成果，但是我们不应就此翻篇，而是应该充分吸取经验和教训，未雨绸缪，防患于未然，争取在以后出现类似事件时能够快速反应，将其造成的破坏和影响降到最低。实验研究表明，在公共场合佩戴口罩可以大大降低飞沫传播途径，可以说，不论是在工厂生产中的卫生标准，还是在个人防护中的安全措施，都需要口罩发挥其积极作用。但是在实际生活中，仍存在许多没有认识到口罩重要作用或根本不在意的人员，且目前只有少数的机构注意到了口罩检测方面的相关研究，并且不同方案的识别效果在密集人流的检测表现参差不齐。许多开发商在闸机、门禁等细节场景的分析时，更是因为数据集的匮乏或模型开发的欠缺经验而感到棘手。

在此背景下，我们尝试根据现有理论设计构建自己的深度学习网络并进行相应场景的模型训练从而达到口罩识别的功能。在此基础上，为了更好的和实际环境与场景需求相结合，我们还运用百度飞桨自主研发的预训练模型管理工具PaddleHub，并结合Fine-tune模型微调等完成对更先进算法的迁移学习以及落地部署，从而让模型能更好服务于用户特定场景的应用。

### 项目研究意义

在国家和全社会的共同努力下，抗击疫情取得了巨大的成果。随着疫情的毒

性减弱，在全面放开的大背景下，个人成为了健康的第一监护人。但是这并不意味着我们可以随意摘下口罩；相反，戴口罩不仅能够有效保护自己减少感染概率，也能对整个社会的病毒的传播起到一个非常好的作用。因此在许多公共场合，佩戴口罩是一项十分被建议的积极行为。通过基于PaddleHub架构研发的人脸检测与口罩识别系统，可以在大型公共场所入口等处有效识别人脸并检测人员是否佩戴口罩，除此之外，在工厂生产车间等对卫生要求严格或对佩戴口罩有必要需求的实际岗位场景中该模型也能很好地应用，从而为社会生产发展做出一定贡献，为社会安全和经济产生积极作用。

## 国内外研究现状

目标检测(Object Detection)是一项经典的视觉任务。如果说在最基本的图像分类场景，我们需要通过模型的学习来成功的判断输出图片的类别，那么目标检测任务就是在对目标区域进行分类之前，还需要先通过回归得到所要对目标进行分类的区域。而图像分割则是在图像检测的基础上，精确到对每个像素进行分类，可以说这一系列应用场景相辅相成，构成了深度学习中常见的计算机视觉任务。而目标检测就是起这样一个承前启后的作用，并且该任务本身也具有广泛的应用与需求。

### 传统目标检测方法

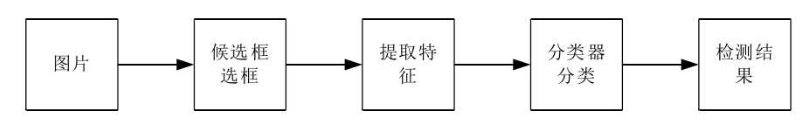
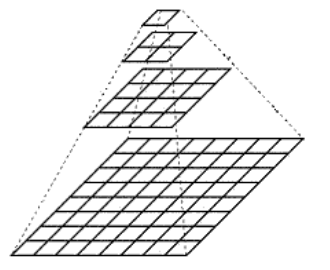
传统的目标检测算法主要包括3个步骤（如图1.1所示），首先是在图像中选择可能会存在目标的区域，模型会通过滑动窗口等方法将图片化分为成百上千的候选区域框；其次是对各区域进行提取特征，常用的方法有HOG，图像金字塔等；最后将其回归为传统的机器学习分类问题，通过SVM。随机森林等分类器对目标进行识别。

图1.1 传统目标检测流程图

HOG（Histograms of Oriented Gradient）是一种常用于目标检测任务的梯度直方图，它将区域内图像的梯度直方作为该区域内的体征[1]。HOG在计算直方图时是根据梯度而不是通道颜色来计算。这些根据梯度得到的特征可以帮助模型有效的进行目标检测等图像任务。HOG提取人脸的各种特征信息可以有效实现对人脸的检测功能。有两个因素会影响HOG在特征提取时的效果：即图片中目标的位置以及目标在图像上的尺度。相同的物体在尺度不同，位置各异时的检测效果参差不齐，为了使检测效果具有更好的鲁棒性，一般会结合图像金字塔、滑动窗口等操作对图像进一步处理。

顾名思义，图像金字塔可以将图像放缩成一个尺寸递增或递减的集合（如图1.2所示），对同一个图像进行不同尺度下的多次采样。把最高尺寸的图像放在底部，在其之上的图片依次递减，直至顶层只有一个像素的图像。它可以帮助人们从不同的尺度对一张图片进行分析，这一思想也被后来深度学习的SSD等算法借鉴。

图1.2 图像金字塔

### 基于深度学习的目标检测

近十余年，随着硬件设备的更新迭代和数据规模的指数级增长，深度学习迎来的飞速的发展，依托于GPU等强大的计算能力以及数据集的完善，相比于传统机器学习方法，深度学习在目标检测中的表现更加突出，是未来技术发展的主要方向。

在目标检测领域目前有 one-stage 和 two-stage 两种方法。two-stage 是指该算法有两大步骤，首先通过学习得到可能存在目标的区域，然后在这些学习得到的候选区域里进行分类，比如经典的R-CNN算法；与之相对的是 one-stage 检测，即算法无需提前学习来预测目标框的位置，而是同时对目标区域的位置和目标分类进行迭代学习，如Yolo、SSD等算法。相比于two-stage，one-stage在牺牲精度的同时提升了收敛速度，在实现时可根据需求和实际情况选择合适的方法。

传统的目标检测方法需要人为标注特征，无论是训练速度还是模型精度都差强人意。在2014年，[Ross Girshick](https://arxiv.org/search/cs?searchtype=author&query=Girshick,+R)[2]等人创新性的提出了基于深度学习的卷积神经网络R-CNN,该模型先对候选特征区域学习，再对待检测区域进行分类的两段式范式，即它的先监督式预训练，后在特定领域微调思想对后面其他的目标检测方法产生了广泛影响。R-CNN先利用滑动窗口算法得到可能的候选框，并在候选框中进行卷积来提取特征，最后利用支持向量机（SVM）对候选区域进行分类。随后作者在此基础上对算法进行了创新，如加入ROI池化，优化分类器等；在2016年，作者设计出了Faster R-CNN算法[3]，在原有的基础上采用RPN网络来生成候选框，大大提高了模型的训练速度和预测精准度。

YOLO[4]系列算法是常用的One-Stage算法，也是目前最为流行的目标检测算法。这主要得益于它快速的训练速度以及不断地迭代更新(目前已发展到V8版本)。它将目标检测重新定义为单个回归问题，可以从图像像素的学习中直接得到目标区域和目标分类。但是YOLO在检测相隔很近的不同目标时由于分类的局限性往往只能得到其中一个类别，并且在同一物体不同尺度的检测中表现不佳。

SSD(单发多框检测)是另一种一阶段检测算法，它解决了当时Faster R-CNN对小目标检测效果差和检测速度慢的问题。SSD借鉴了featurized image pyramid的思想，将同一图像放在不同尺度上进行学习，从而对不同大小的目标进行较好的检测。在CoCo，VOC等数据集的表现上，它的训练速度及网络灵活性要优于Faster R-CNN，同时在预测目标尤其是小目标的准确度上也要高于Yolo。[5]在本文的后半段，本文将尝试根据论文文献复现一个SSD目标检测器，用于通用场景下的人脸检测与口罩识别。

### 改进的目标检测器

随着深度学习的发展，各式各样的目标检测器大放异彩，但是在现实生活具体任务场景划分时，不同的检测器有不同的特点，因此需要我们根据实际需求选择合适的算法，并针对性的加以改进。Pyramidbox就是这样的一种算法[6]。人脸检测是经典的计算机视觉任务，对小脸、模糊、光照、角度等复杂问题的处理决定了模型在实际应用中的鲁棒性。PyramidBox在SSD的基础上加强了在未受控场景下对面部遮挡、模糊等困难图像的学习，它将目标的环境应用到检测当中，在目标检测尤其是小目标人脸检测方面取得了较好的效果。

## 论文主要研究内容

小目标、部分遮挡以及模糊等是目标检测尤其是人脸检测面临的重要挑战，因为生活场景中的闸机，门禁等嵌入式端设备算力有限，因此需要选择速度更快，模型更小的一阶段目标检测算法部署落地；而现实环境中人脸尤其是是否佩戴口罩检测在监控设备采集图像时往往只有很小的区域，即小目标和多尺度是大部分需要检测的目标，因此本文选择结合了R-CNN精度和Yolo速度优点的SSD算法作为主要研究模型，并利用更具针对性的PyramidBox算法进行迁移学习，从而更好地实现人脸检测和口罩识别任务。本文的主要工作及研究内容如下：

1. 在检测目标小，目标尺度随机的实际环境下，本文基于文献尝试利用Pytorch框架复现SSD模型，在此基础上完成人脸口罩数据集的收集、数据的预处理和图像增强、模型的训练及评估、改进模型训练方法等工作，以期得到准确性较高、鲁棒性较好，泛化能力较强的口罩检测模型。
2. PaddleHub是百度自主研发框架飞桨下的模型社区，相较于torchvision的model库，该框架提供了丰富的API功能，可以方便地获取PaddlePaddle生态中的预训练模型，结合Fine-tune API，可以利用大规模预训练模型实现快速的迁移学习，让预训练模型能更好地适应用户特定场景的需求。本文使用的PyramidBox人脸检测器就是基于Paddle框架开发的。我们可以很方便的进行迁移学习。
3. 在得到效果较好的人脸检测与口罩识别检测器后，为了项目的完整落地，我们开发了一套完整的线上口罩识别系统。在前端，我们利用HTML、JavaScript以及Ajax等技术构建用户页面；在后端，我们利用Flask构建了一个微服务器，将模型在线部署，使得人们可以很方便的在线上体验口罩检测。本系统支持图片检测、视频检测以及摄像头实时检测。为了进一步应用，我们还将模型部署到android等端侧设备，使其可以应用于任何带有摄像头的区域。

## 论文组织结构

本文受口罩检测的社会必要性及当前对此研究的匮乏性所启发，根据口罩检测目标小，尺度多等特点，选择SSD及PyramidBox这类适合进行小目标多尺度检测的单阶段检测器，力图实现模型从零到有的搭建，并采取Dropout、半精度检测，焦点损失等较先进的思想改进模型训练效果。在此基础上，利用PaddleHub库进行迁移学习，使得模型有较好的鲁棒性，兼顾检测精度和检测速度。最后基于训练好的模型，利用flask以及前端框架实现人脸检测与口罩识别的部署开发。本文的具体结构如下：

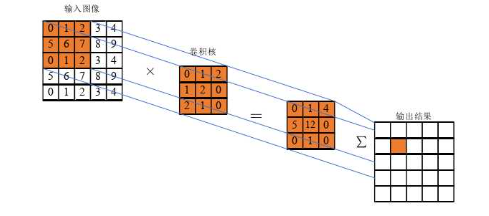
1. 绪论，主要进行了项目研究的背景分析及其实际意义，指出当前口罩检测系统落地开发的必要性及当前相关研究的不足。由此引出了目标检测这一主要任务领域及其相应的国内外研究现状。在比较分析了各类传统方法及深度学习方法的优缺点后，选用了适合目标任务的算法——SSD及PyramidBox。
2. 相关理论基础，基于计算机视觉这一大方向下介绍了卷积神经网络这一基本网络块的原理及计算方法。在此基础上介绍了目标检测的常用方法，并引出了本文所使用的算法模型。详细介绍了本文所用算法的网络构成及特色，为下一章的具体实现奠定基础。
3. 算法设计，通过简洁的介绍及大量的代码视图言简意赅地阐述模型的完整构建过程以及改进方案。
4. 系统实现，俗话说，数据决定了模型的上限，而算法只是在不断逼近这一上限。由此可见模型训练前的数据收集以及预处理的重要性。本文在具体训练前，通过爬虫以及kaggle等网站收集到合适的大量数据集，并进行了剪切、旋转、缩放等图像增强操作；在模型训练完毕后必须要进行相应的推理预测，而模型的输出只是一系列字符串，若想将预测结果展示为图像，还需要进行相应的可视化操作。在模型训练完毕后还需要必要的前后端工作使得基于该模型的系统能够简单便捷地供大众使用。
5. 系统测试部分展示了经前后端分离，接口封装后该系统的完整功能测试，以期该系统能够具有现实落地使用的功能实现。
6. 安卓部署，为了更适用于实际应用场景，本设计利用Android Studio将模型部署到了移动端，使得设计成果能在更广泛的场景进行应用。
7. 总结与展望，对本文所研究的人脸检测与口罩识别进行总结，归纳了本文主要的研究内容，提出了不足与改进之处，并结合未来趋势对该项目的发展进行了展望。

# 相关理论基础

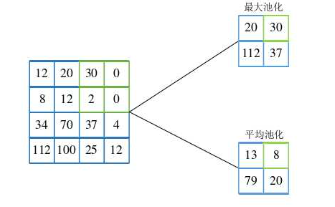
## 卷积神经网络

卷积神经网络通常包含卷积、池化、全连接层这几个主要模块，并且还运用了激活函数、损失函数等知识[7]。卷积神经网络通常用于计算机视觉处理方面的应用，这主要得益于它对三维数据即长宽高的高效处理同图像的高宽颜色具有相同维度，使得图像特征的局部相关性能够很好的保留下来，从而使网络学习到更加深层次的内在特征联系，增强对特征的学习效果[8]。

卷积层 卷积层是卷积神经网络的核心模块，主要通过对图像各像素进行卷积操作从而学习目标特征（如图2.1）。在该层中需要人为或随机的指定一个卷积核用来对图像进行卷积操作。卷积核的数量以及核元素的选取很大程度上决定了卷积学习的质量。卷积层通过划分多个区域分别学习局部特征，进而将各区域特征组合起来，不仅能提高学习速度，还能降低过拟合的风险；同时，权值共享使得网络通过较少的参数就能得到较好的效果。

图2.1 卷积过程

池化层 由于常见的图片长和宽都能达到数百，因此一张图片包含的像素点的数量是巨大的，同时目标区域可能只包含很小的区域。为了防止卷积层过拟合学习这些无用特征，同时对数据进行降维加快学习速度需要引入池化层。池化层减少了对背景等无用信息的学习，使网络对较大尺寸的图像也有一定的学习能力。池化分为平均池化和最大池化两种（如图2.2）。一般来说，最大池化使得更多的纹理信息得以保存，所以适合模型较小、网络层数较浅的情况，而平均池化使更多的背景信息得以保留，更加适用于模型大，网络层数较深的情况[9]。

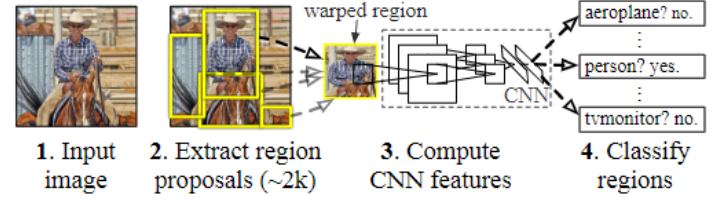
图2.2 两种池化方式对比

全连接层 该层一般在网络结构的尾部，用于将前层提取到的特征进行整合，从而得到模型的输出结果，该输出是各层特征以一定权重加权后对数据的学习结果，反映了各特征对数据学习的某种映射关系[10]。一般情况下会在全连接之后紧接一个激活函数隐藏层用于提高模型的非线性学习能力。虽然全连接层能很好的整合各层对特征的学习，提高预测精准度，但是全连接层数量过多可能会使得模型的计算量增大，学习速率变慢，且可能会使模型有过拟合的风险[11]。

## 常用的目标检测器

### Faster R-CNN

1. CNN系列无疑是最经典的双阶段检测模型之一，并且对整个目标检测领域都产生了深远影响。在R-CNN系列之前，目标检测大多都还处于传统的机器学习检测方式。R-CNN利用深度学习的方法使得检测的精度大幅提升，各方面指标都大幅度领先传统目标检测方法。首先，该模型会使用SS算法对每个输入图像都生成数千个候选区域；然后，对每个生成的区域，模型都会使用深度分类网络对区域进行特征的提取；随后提取到的特征会被送入SVM分类器进行分类得到所属类别；最后，模型会对目标区域框进行精修，提高回归的准确度（如图2.3）。相比传统的目标检测，该模型大大提高了精确度，但是因为要进行大量的卷积运算，会需要较长的训练时间与较大的空间占用。

图2.3 RCNN模型的训练流程

随后，模型设计者在原有的基础上加入了ROI(Region of Interest)池化层对每个候选区域的特征进行最大池化下采样。分类器则由SVM更换为SoftMax，在一定程度上改善了R-CNN计算量大，训练时间长的缺点（如图2.4）。

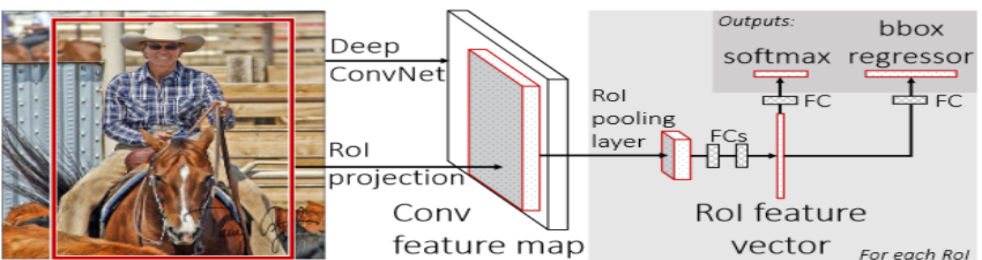
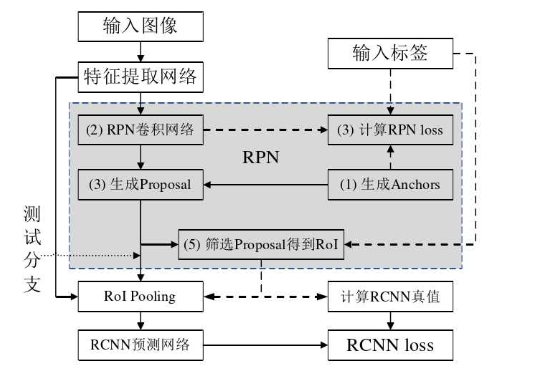


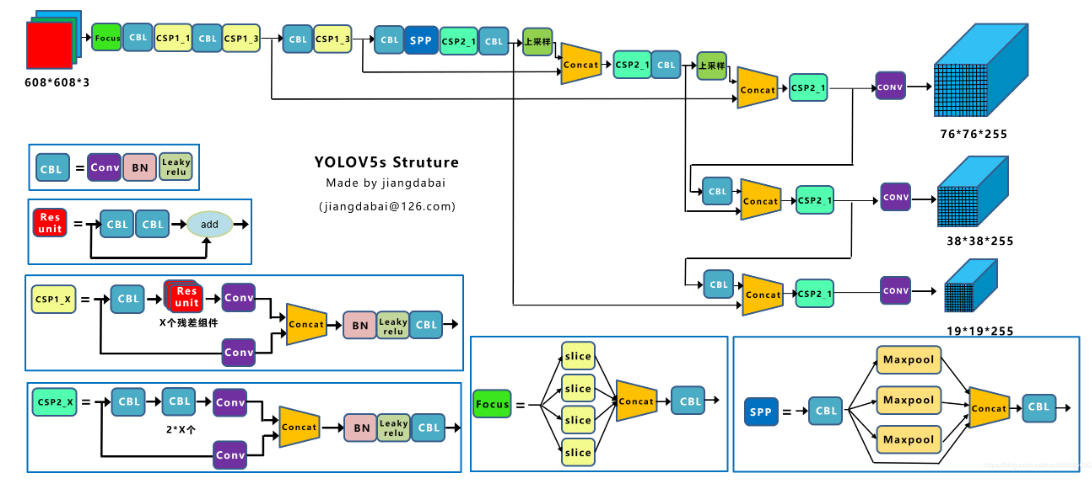
图2.4 改进的R-CNN模型

该模型作者在2016年在改进后的Fast R-CNN模型上创造性地加入了区域建议网络(RPN)，这就是现在广泛使用的Faster R-CNN。RPN网络是全卷积网络，其核心思想是用CNN直接生成Region Proposal[12]，该网络借用了滑动窗口的思想，使得模型能够快速准确地得到候选框，取代了耗时的选择性搜索，从而使得模型的检测速度大大提升（如图2.5）。

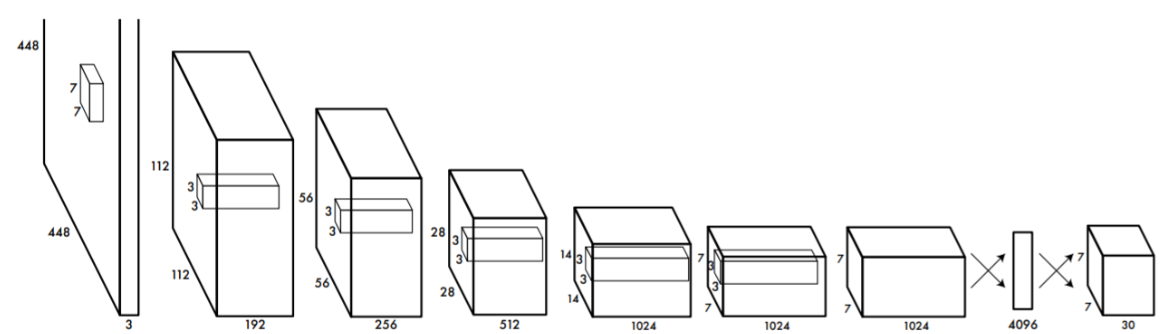
图2.5 Faster R-CNN模型

### YOLO

不同于R-CNN等双阶段检测模型，YOlO是单阶段目标检测的经典算法。YOLO没有双阶段算法那样先取得候选框再分类的过程，而是将问题统一为一个回归问题。它的核心思想（如图2.6）就是将整个图像作为网络的输入，直接将锚框的位置以及该锚框所属的类别回归到输出层。首先，模型会将输入的图片resize为正方形，并均等分为若干个网格，每个网格有固定数量的预测锚框。在训练时，每个锚框会回归自身的位置并给出自身区域中是否含有目标的得分。而包含它的网格则会对网格内的目标类别进行回归；然后，网格会选择得分最高的那个锚框，即每个网格只会输出一个预测框；最后模型会去掉那些得分低的预测框，并用非极大抑制去掉冗余的框。最初版本的YOLO存在以下问题：由于每个网格只会判断出一个类别，因此若两个较小目标紧邻在一起时，模型可能会漏检。另外，在YOLO的损失函数中，大物体和小物体的损失贡献度相近，这就使得模型在检测一些小目标时精度受到影响。此外，对于形状不规则或较抽象的物体，模型的检测效果也并不理想。

图2.6 YOLO检测流程

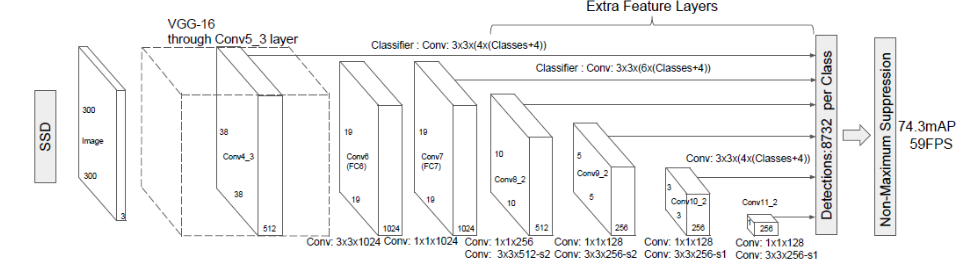
YOLO开源后，其一步到位的思想迅速引起了人们的关注与研究。YOLO的版本也不断的改进完善。目前最流行的YOLO v5版本分为输入端、主干网络（Backbone）、Neck、输出预测端四个部分。在输入端，新版本的模型采用了Mosaic数据增强，将四张图片采用随机增强的方式拼接在一起。并且采用了自适应锚框大小及图片缩放；在主干网络中，使用了Focus结构进行切片操作，引入了CSP结构下采样减少梯度信息重复的问题；在Neck中使用FPN+PAN的结构增强了对特征的学习能力；在输出端设计了DIOU\_Loss损失函数来替代常用的L1，交叉熵等损失函数。新版本的YOLO不论是在精度还是速度上都有了显著提升，使其以极强的灵活性和快速高效的训练而广受欢迎。

图2.7 YOLOv5网络结构

## 单发多框检测(SSD)

SSD是一个一阶段的检测算法。它结合了R-CNN和YOLO各自的优点：采用了类似R-CNN区域划分候选框的思想，自动生成一系列锚框。同时使用了类似YOLO的一步到位思想，在训练中直接回归目标的类别和偏移量。使得该模型不论推理速度还是预测精度都有较好的效果。

### SSD网络架构

图2.8 SSD网络结构

首先，模型在输入端会自动将图像放缩为300×300的尺寸，随后喂入模型的主干网络VGG16。VGG是常用的图像分类网络，SSD截取了其一部分(至第五卷积块第三层)作为模型的主干网络。此时的输出即为第一层特征矩阵(38×38×512)。

通常的算法(例如YOLO)会在卷积后接两个全连接层。第一层用于映射特征关系，第二层会输出分类信息，这样做虽然能成功的输出结果，但会带来大量的冗余计算，减缓模型速度。SSD模型摒弃了末端的全连接层，转而用卷积层来提取特征并分类回归。[13]在此基础上，SSD创新性地采用了6层不同尺寸的特征提取层。图像在经VGG输出后，会喂入一系列不同大小的卷积特征层。浅层的特征层会生成更多的锚框，同时这时图像的细节特征被更好的保存，因此适合较小目标的检测；接下来随着特征的前向传播，深层的特征块会将接收的特征图长和宽减半，减少当前尺度下的锚框数量，从而使得当前特征具有更为广阔的感受野,能够对较大的目标特征进行学习。当待检测目标较小时，模型会在前端的特征提取层中进行标注；反之，当待检测的目标较大时，拥有更广阔感受野的后端特征层会对目标有更好地学习能力（如图2.9）。

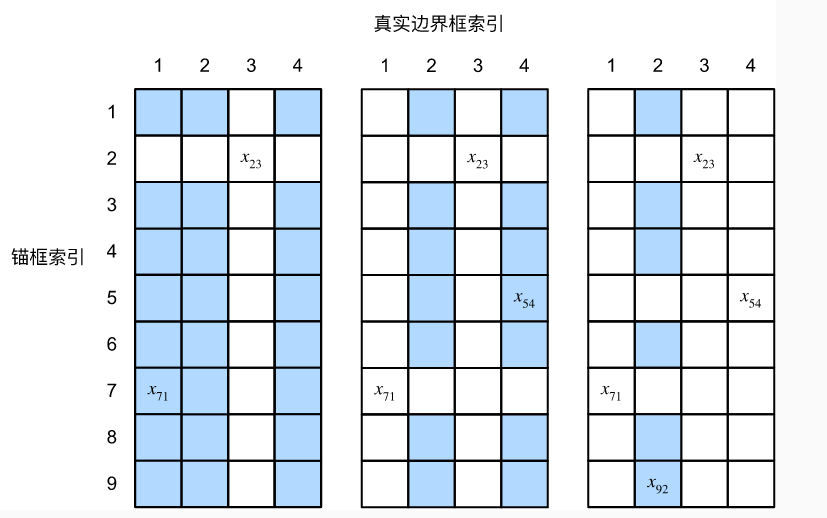
图2.9 不同尺度的感受野

如图2.9所示，当图像经过8×8检测层时，该尺度下的锚框成功匹配到了较小的猫，而当图像传入到4×4检测层时，较大的狗被本层的锚框检测到。简单来说，单发多框检测通过在不同的层级生成不同尺寸的锚框，通过对锚框的梯度训练得到偏移修正后的位置及目标类别来检测不同尺寸的物体，从而实现了多尺度目标检测。

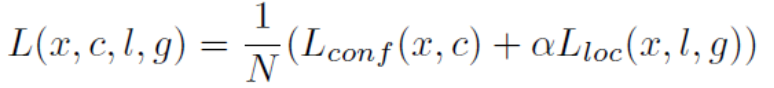
### SSD训练过程

SSD模型的训练过程通常分为四部分：锚框生成、锚框标注、损失函数计算以及非极大值抑制（NMS）。每个步骤的具体操作如下：

1. 锚框的生成 在训练前需要规定以每个网格为中心生成锚框的数量。若设置m个缩放比和n个高宽比，那么每个网格会生成(m+n-1)个锚框。若图像的宽和高分别为w、h，以每个像素为一格，那么整张图片会生成的锚框数量为w×h×(m+n-1)。
2. 锚框的标注 模型在训练时是以锚框为基本样本的。交并比(IoU)是评测锚框质量的重要指标。设共有m个真实框，n个预测框(m<n)，模型会生成一个m×n的张量(tensor)，根据交并比来给每个预测框评分。首先会迭代m次，每次都找到当前张量中的最大值，把对应的真实框分配给该预测框，然后将本元素所在行和列的元素都删除，再进行下一次迭代，直至所有真实框都被分配给了一个预测框。此时，张量中可能还存在一些未被删除但是也未被分配的预测框，模型会遍历这些剩余框，若IoU大于规定的阈值则被标为正样本，否则被标记为负样本。

图2.10 将真实锚框分配给预测框

1. 损失函数的计算 在为每个预测框进行标注后，模型会对每个被标注为正样本的锚框进行损失值计算。每个框对类别的置信度和与真实框的偏移量相加构成了总的损失函数，分类损失conf常用交叉熵及softmax，回归损失loc则常用Smooth L1损失，如公式（2-1）所示：



公式(2-1)

1. 非极大抑制 经过初步训练后可能有非常庞大数量的锚框检测到了目标，这些锚框常常相似并且有很大的重叠部分，这时，我们就需要利用非极大值抑制（NMS）来简化模型的输出，因为通常评分最高的那个锚框就已经很好的框出了目标。

### 与其它模型的比较与改进

深度学习中的大多数检测算法，不论是SSD、YOLO还是R-CNN,其本质都是对滑动窗口生成的区域进行分类，但因为其主干网络与具体实现方法不同，所以产生了各种差异，可以说没有最好的算法，只要能解决问题的就是好算法。

表 2.1 各目标检测算法性能

|  |  |  |  |
| --- | --- | --- | --- |
| 模型 | 平均精度(mAP) | 推理速度(FPS) | 复杂度 |
| SSD | 75.1% | 58-72 | 低 |
| YOLO | 58.8% | 45-150 | 中 |
| Faster R-CNN | 73.1% - 85.6% | 15 | 高 |

通过分析表2.1(数据来源:Dominic\_S)我们可以发现，在对目标预测平均精度的表现(实际正样本/预测正样本)上，SSD与R-CNN都有优异表现，但是SSD却比R-CNN的推理速度快了数倍；在同样具有较快推理速度的前提下，SSD比YOLO平均预测精度大幅领先。初此之外，SSD对于不同近邻目标可以预测出多个，而不是像YOLO一样只能输出一个目标。同时，SSD基于部分VGG加入少量特征层的网络结构也比其他两种模型更为灵活轻便，能够部署到嵌入式设备中进行实时预测。

与任何算法一样，SSD也存在一些不足之处。首先由于只有较浅特征层才能学习到小目标，后端其他特征层更专注于更大尺寸，使得小目标的检测效果相较还有提升空间；同时，由于其锚框生成方式，在密集场景检测时可能会出现重复检测问题。针对以上问题，为了改善模型的性能，可以通过一些措施可以改善效果。

表 2.2 [13]对SSD模型进行改进

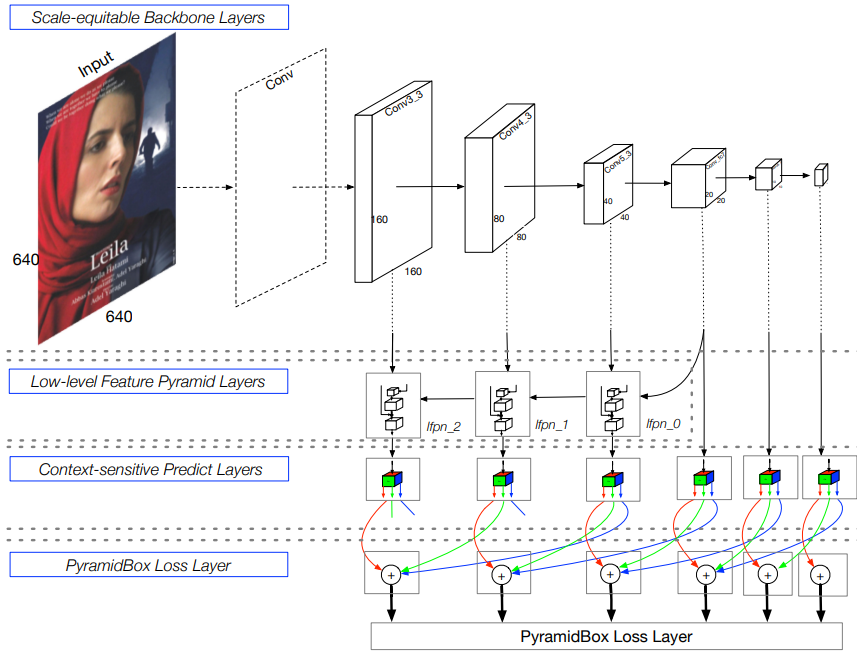
|  |  |
| --- | --- |
| 改进措施 | 平均精度(mAP%) |
| SSD300 | 68.0 |
| SSD300+07+12+coco | 79.6 |
| SSD512 | 71.6 |
| SSD512+07+12+coco | 81.6 |
| SSD300 + Augmentation | 74.3 |

表2.2表明，数据的质量和数量对提高模型效果至关重要。在用VOC 2007、VOC2012和CoCo三个常用数据集代替单一数据集对模型进行训练后，mAP提升了11.6个百分点，在将输入图片尺寸从300放大到512后，mAP提升了3.6个百分点，在对数据进行图像增强操作后，mAP提升了6.3个百分点，由此可见数据对模型训练的重要性。林思海[14]在原有的基础上将基础网络替换为更深层次的ResNet50，并提出了BAFF特征融合来增强各特征层间的交互能力，使模型效果得到了进一步提升。除此之外，为每个网格增加生成锚框个数，增加更多不同尺寸特征层，改进默认边界框平铺等操作可能也会对提升模型效果有帮助。

## Pyramidbox 人脸检测

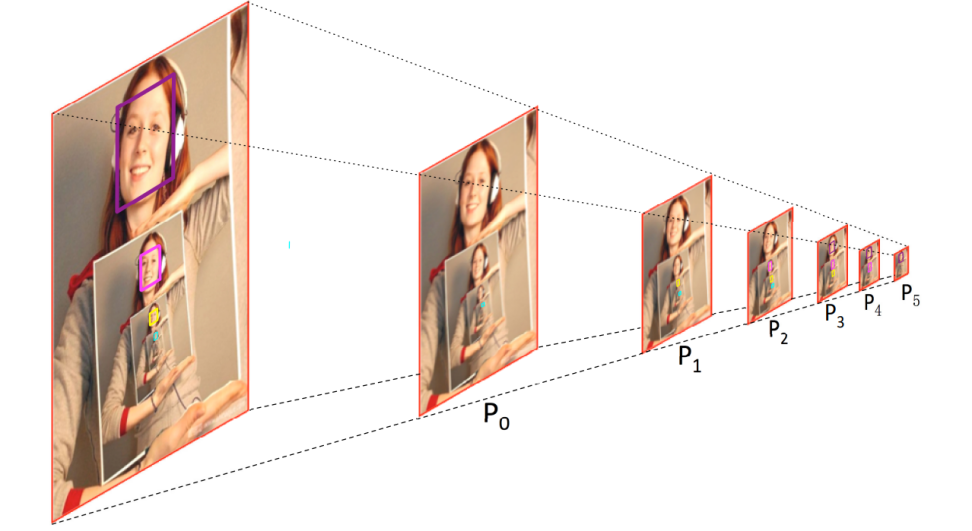
### Pyramidbox网络结构

PyramidBox是百度于2018年发布的一款专注于小尺寸，难识别的人脸检测算法。该算法是2018 年 WIDER FACE人脸检测冠军算法。它是一种基于SSD改进而来的算法，巧妙利用了人脸的环境信息来帮助困难场景下的人脸检测。

图2.11 PyramidBox网络结构

该模型共分为Backbone、LFPN、CPM和损失输出四层，并设计了Data-anchor-sampling方法来对图像进行增强，各模块的分析如下：

主干网络层 算法使用了与SSD算法相同的截断VGG16网络和锚框[15]，可以在不同尺寸的六层特征网络上生成一系列锚框。与SSD不同的是，为了实现目标环境上下文的学习，网络在训练时用半监督的方法来为每个脸部目标生成头部和身体的上下文特征，并将其命名为PyramidAnchors。通常情况下人脸不会在图像中单独存在，而是会与头部、身体等特征共同出现，算法作者认为，对人脸上下文的学习可以有效地提高小目标的检测效果。在多层次多尺度目标检测中，更深层的特征层有更大的感受野，可以对人脸周围的环境进行学习，并半监督的对周围环境进行标注，如下图2.12所示：在对图像中标注的尺寸为8的最小人脸进行学习时，P0通过原始人脸标注，P1则通过对应的尺寸为16的头部进行，P3则通过对应的尺寸为32的身体标注。

图2.12 对人脸上下文进行标注

浅层特征金字塔层(LFPN) 在深层特征层学习了目标环境上下文后，需要将其与浅层的细节特征层融合，所以引入了特征金字塔(FPN)[16]。FPN使用了一种自上而下的结构，可以将高层的环境特征融合入细节信息保存更好的浅层中，从而更好的利用深层次的环境特征。一般的FPN是从最高层开始融合的，然而不是所有的深层次特征都对小目标检测有意义，过大的尺寸使得大部分特征都对浅层帮助不大。基于此，论文作者提出了LFPN，该方法是从整个网络的中层而不是顶层对特征进行融合的，这就使得深层能包含更多浅层需要的环境信息，同时还减小了信息冗余量。在PyramidBox算法中，作者将前三层特征层喂入不同的特征金字塔，并从第四层开始进行特征融合。

环境敏感预测层(CPM) 在较为基础的目标检测器，比如 SSD[13]和YOLO[4] 中，模型直接在特定的特征层上进行预测。MS-CNN [17]指出，增大任务的子网络可以提高模型的准确度。SSH (Single stage headless face detector)[18]在SSD基础上通过在层上设置不同大小的预测块提升了感受野。DSSD(Deconvolutional single shot detector)[19]设置了残差块用于预测。PyramidBox结合了两者的优点，同时加入了max-in-out 层，设计出了CPM来帮助模型更好的利用融合特征进行回归和分类。在CPM输出的20个通道中，对人脸的分类用了4个通道，对身体和头部的分类用了4个通道，另有12个通道分别用于对三者的回归。

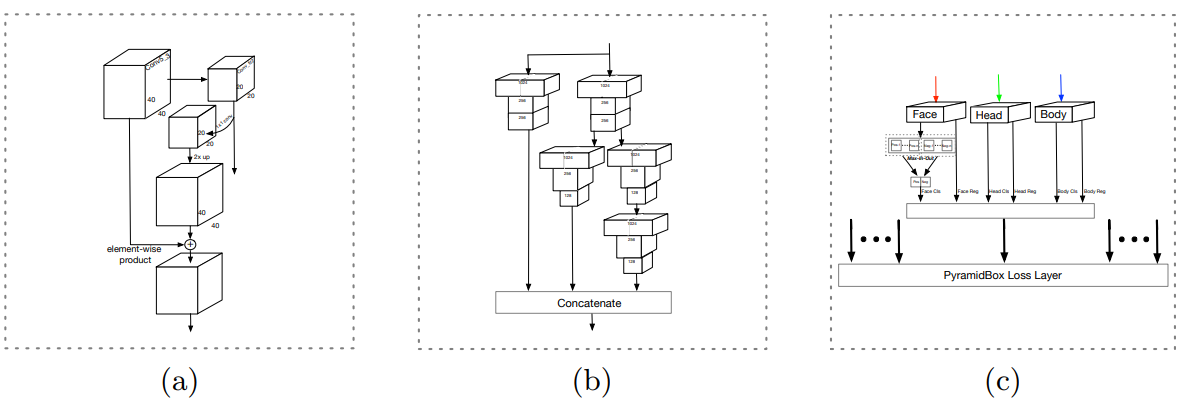
最后，为了改善对小目标的检测效果，作者提出了一种 data-anchor-sampling 策略用于改变训练样本的分布，使其更适用于小人脸检测。简单地说，就是把图像中的一个随机的人脸变成一个随机的更小的尺寸。通过这一方法，可以使训练样本中有更多的小尺寸目标。

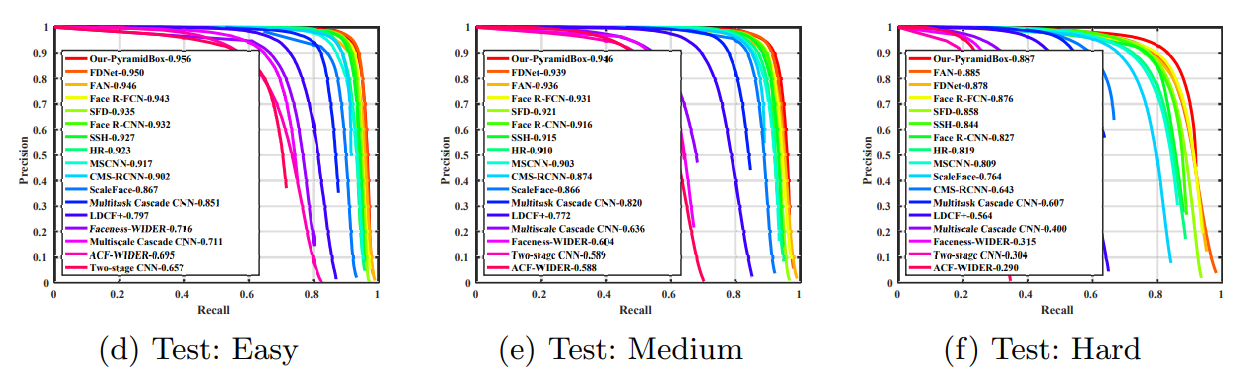
图2.13 （a）LFPN （b）CPM （c）LOSS

### Pyramidbox性能分析

表 2.3 [6]各模块性能对比

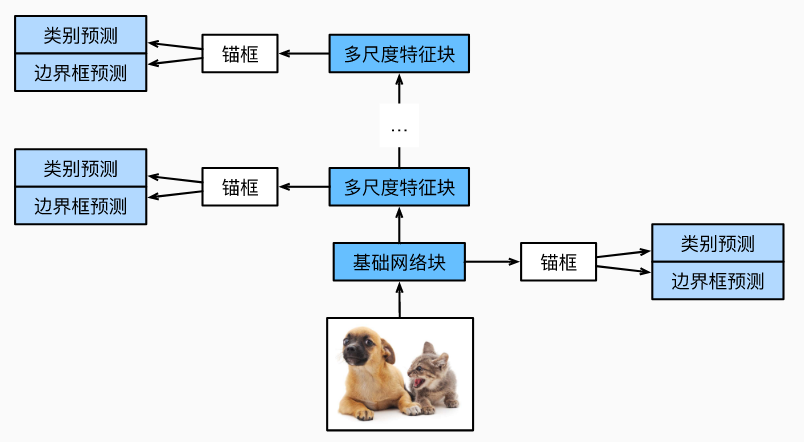
|  |  |
| --- | --- |
| 措施 | mAP  easy medium hard |
| Baseline | 94.0 92.7 84.2 |
| LFPN | 94.3 93.3 86.1 |
| Data Sampling | 94.7 93.7 86.7 |
| Pyramid Anchors | 95.5 94.3 88.3 |
| CPM | 95.9 94.7 88.8 |
| Pyramidbox | 96.1 95.0 88.9 |

通过控制变量分析可发现，在基础网络中引入特征金字塔使模型在困难数据的精确率提升了1.9个百分点，表示FPN的引入对模型对小目标的检测有显著提升；同时在增强数据集小目标样本后，模型的效果进一步提升；值得一提的是在运用了上下文信息生成额外标签锚框后，模型对小尺寸人脸的检测精度相较基础网络提升了4.1个百分点，可见利用上下文进行困难目标检测是一种较好的思路。

图2.14 各人脸检测模型性能对比

# 算法设计

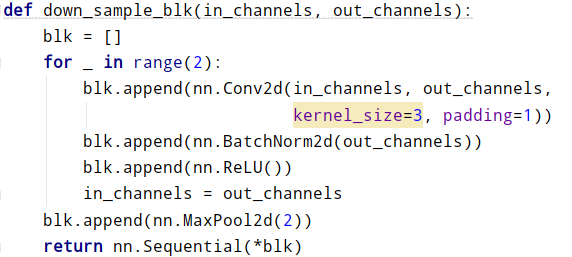
在对SSD算法（2.3）的学习研究后，本文尝试根据论文书籍等文献，利用Pytorch框架从零搭建一个SSD模型用于对任务目标人脸识别与口罩检测进行研究和分析。受限于硬件设备与研究需要，本文主要利用单发多框检测的思想，在五个不同的尺度上对目标进行检测，并取得了不错的效果，模型的具体流程和结构如下：

图3.1 模型的网络结构

## 网络搭建

### 高宽减半块

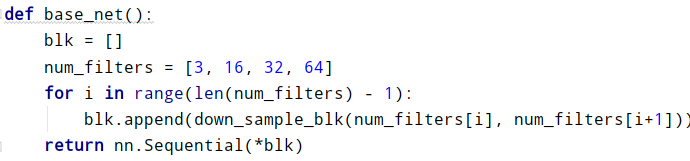
若想要实现SSD算法，构建不同尺度的特征预测层是关键。通过对VGG的借鉴，我们定义了高宽减半块用于实现不同尺度下的目标检测。该模块由两个卷积核大小为3，填充为1的卷积层和一个2\*2，步长为2的最大池化层构成。该形状下的卷积操作不会改变输出的尺寸，但后接的池化层会将输出的特征图长宽减半。根据1\*2+(3-1)\*2=6可知当图像经过该模块后会有6\*6的感受野，此时模型可以从更大的尺度上对图像特征进行学习。实现代码如下：

图3.2 高宽减半块

在此函数中，我们将两个相同大小的卷积层相连，其中每个卷积层在进行卷积操作后会连入NB隐藏层进行归一化操作，并用ReLu激活函数进行非线性映射，最后连入最大池化层。函数的两个参数为输入的通道数（初始为3）和期望输出的通道数。若我们输入一个([2, 3, 512, 512])的张量（注：分别代表批大小、通道数、宽和高），参数为3和10，则通过高宽减半块后，图片张量会变为([2, 10, 256, 256]),即实现了特征图高和宽的减半。

### 基本网络块

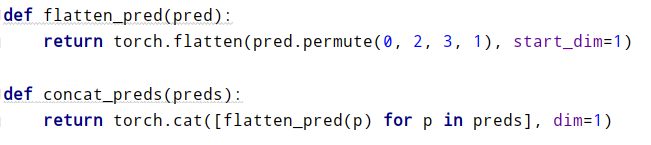
基本网络块是模型的第一个特征层（浅层），图像在输出后，会先进入基本网络块进行特征学习。本模型的基本块由三个高宽减半块串联，并且每次迭代会将图片的通道翻倍，代码如下：

图3.3 基本网络块

将我们尺寸为512\*512的训练样本喂入后，因为经过了三个减半层，每次特征尺寸会减半，根据512/2^3=64，模型会输出64\*64大小的特征图。

### 不同尺度的连接

在单发多框检测中，不同的尺度下锚框的数量和大小都不尽相同，因此各层之间输出的特征图大小以及预测的形状也形态各异。为了将不同层之间的输出结果连接起来以加快学习速度，我们设计了不同尺度的连接函数。

图3.4 不同尺度的连接

初始的张量格式为（批次,通道,高,宽），在当前模块中，我们先将通道移到最后一维，然后对除批次外的其他维进行扁平化操作，具体计算方法为（通道×高×宽）。随后将转化过后的二维张量进行按列拼接。这样，虽然个层间的尺寸不一，我们仍然可以连接不同尺度下的预测。

### 类别预测层和回归预测层

在本文的口罩检测模型中，模型需要区分Mask和Face两种标签，在加上背景后拢共需要预测3个类别。如果以每个像素为中心生成3个锚框，那么在512×512的尺寸下，每张图片会生成512\*512\*3=786432个锚框，如果我们利用全连接层进行输出，那么所需要的计算量和参数信息是十分巨大的。由上文（2.3.1）可知，SSD利用卷积层来代替了全连接层用于输出，从而降低了参数冗余，加快模型训练速度。

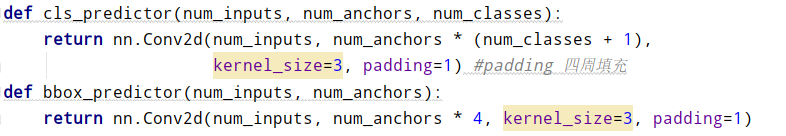
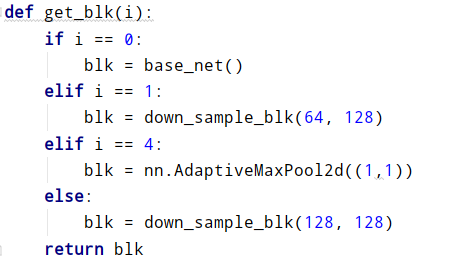
在类别预测层中，我们利用一个3\*3卷积核保持输入输出高宽不变，此时输出的通道数为每个像素锚框数×类别数，并且因为尺寸未变，所以每个坐标都一一对应。

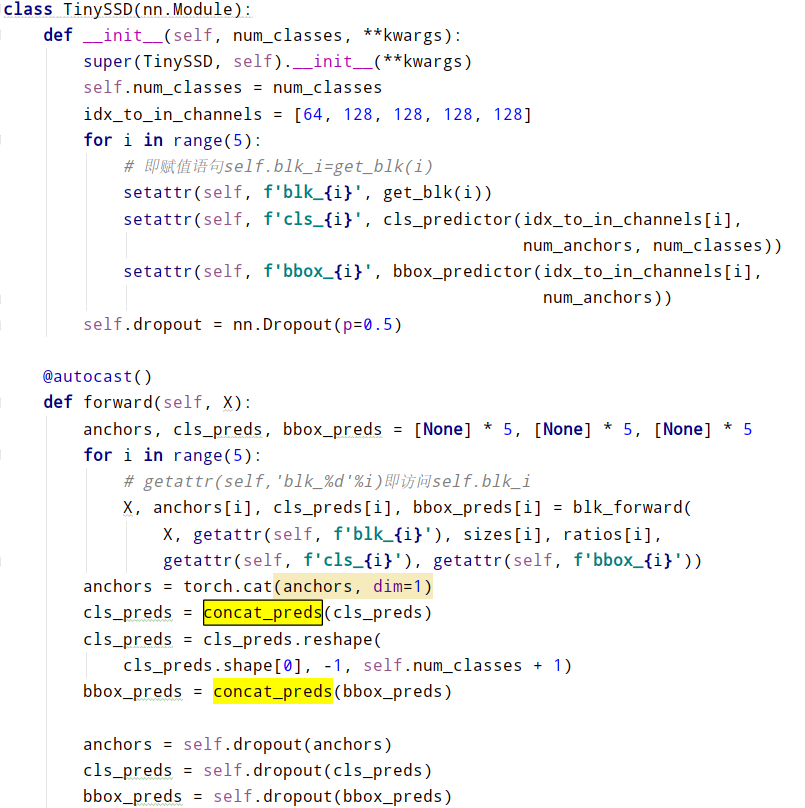
图3.5 类别及回归预测框

在本任务中，我们输入一个([2, 3, 512, 512])的张量，预测层会输出一个([2, 9, 512, 512]),其中通道数9由(锚框数3)×(类别数3)=9得到。在回归预测层中，与类别层唯一不同的是每个锚框会输出4个回归偏移量而不是3个类别。

### 完整网络结构

图3.6 网络结构

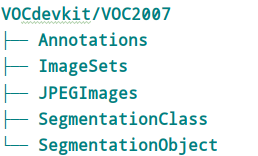
本模型共分为5层，其中第一层为基本网络块，初始尺寸为512×512的图像在经过基本块后会变为64×64；随后，图像会经过三个减半块使尺寸变为8×8，最后由一个最大池化层将尺寸降为1×1，从而实现不同尺度上的预测。

图3.7完整模型

## 数据预处理

本模型用到的训练数据来自AI Studio的开源口罩人脸数据集以及部分从互联网收集并转换为XML结构的数据。共含有2000张戴口罩和人脸数据，共8512个标签框。其中用1600张作为训练样本，又通过随机将图像进行五种不同处理方法对训练样本进行了图像增强，总计3200个训练样本。

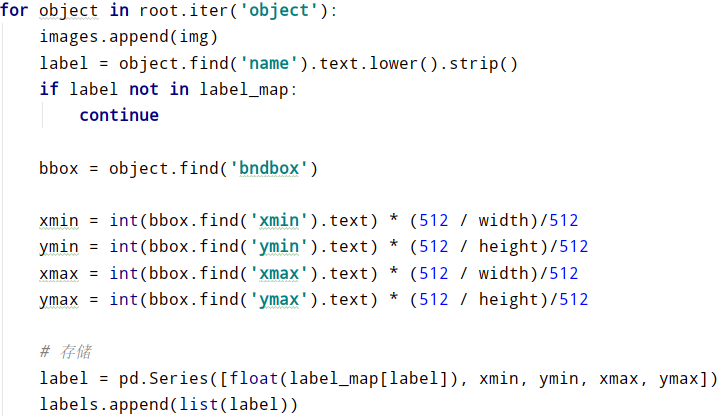
### 图像导入

图3.8 VOC数据格式

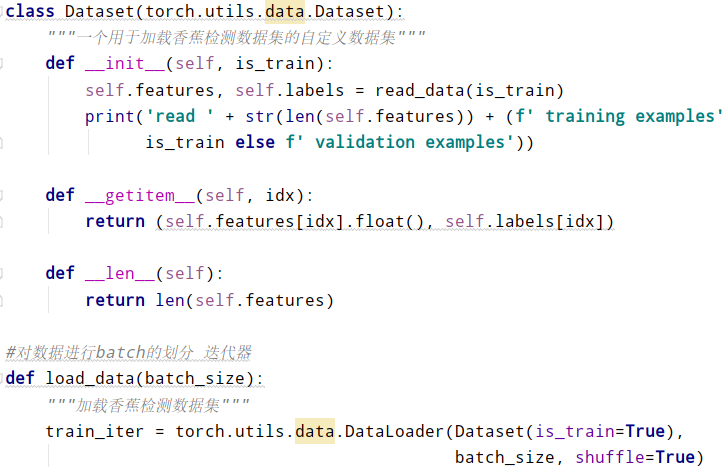
上图是VOC数据集的基本格式，其中Annotations中存放标注图片信息的XML文件，JPEGImages中存放训练图片样本，ImageSets中存放的是图片的路径和名称用于索引。

图3.9 读取图片

在这个函数中，我们利用ET库对读入的xml文件进行解析，根据解析到的图片路径对图片进行访问，并存储在列表中（如图3.9所示）。需要注意的是，我们在读入图片后，还需要对数据除以255来归一化，从而更好的训练和收敛。

图3.10 解析锚框位置

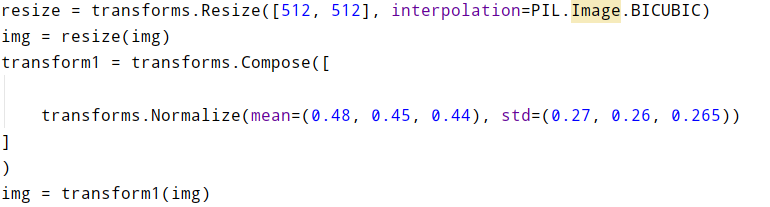
在读入图片后，我们需要提取出该图片的标签框的信息（如图3.10所示），它同样存储在xml文件中。同样，我们也需要对位置信息进行归一化，转换为在图片中的相对位置，并将位置信息传入到另一个列表中使其与图片一一对应。

图3.11 构建迭代器

最后，我们只需要重写Dataset函数，并构建迭代器，便完成了数据导入的工作。

### 图像尺寸

为了平衡各图像由于尺寸带来的权重问题，我们还需要对输入的图像进行resize，使各训练数据更加规整，方便训练。

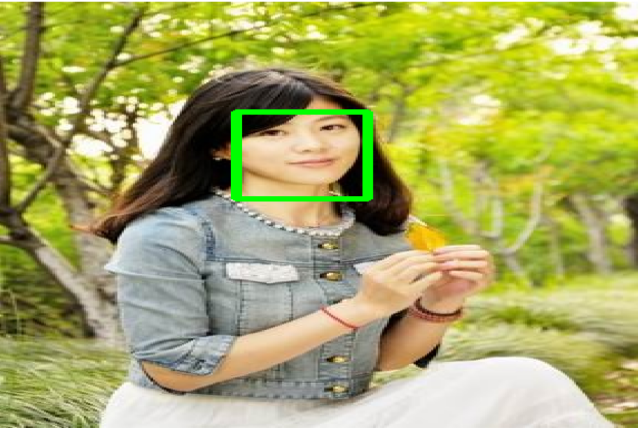
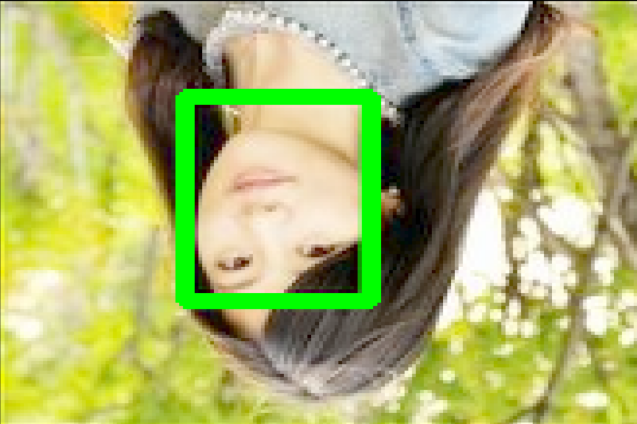
图3.12 修改图像尺寸

首先定义一个形状转换器，来将图片resize为512×512尺寸。随后，计算了数据集各通道的平均值和方差，从而对训练样本进行标准化，更符合正态分布。

### 图像增强

数据的质量和数量是影响模型训练效果的两个重要因素。由于受限于内存和硬件，对数据进行图像增强更为高效和实际。虽然pytorch的torchvision.transforms

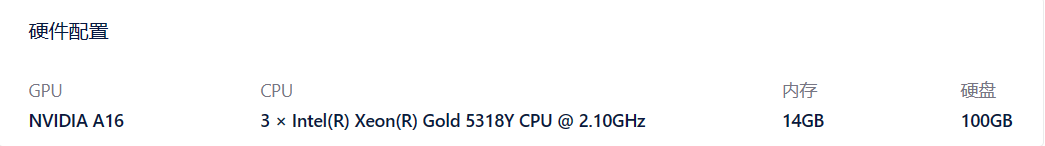
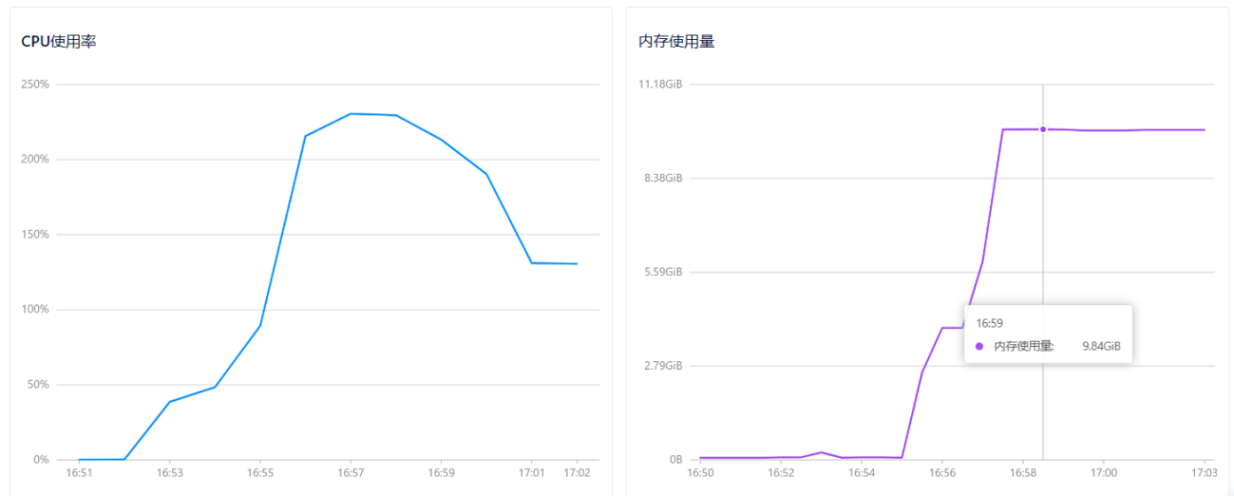
提供了一些常用的数据增强操作，但是却只能对图片进行有限的增强。为此，本文设计了一种新的图像增强方法。该方法会随机对图像进行裁剪、旋转、平移、亮度、噪声和翻转等多种操作，使得模型能接触到各钟各样的数据，从而提高模型鲁棒性。在示例图片中，图像增强算法对原图像进行了裁剪、旋转和亮度操作。

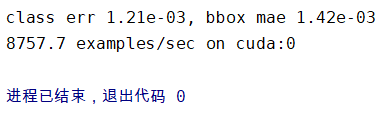
图3.13 图像增强效果对比

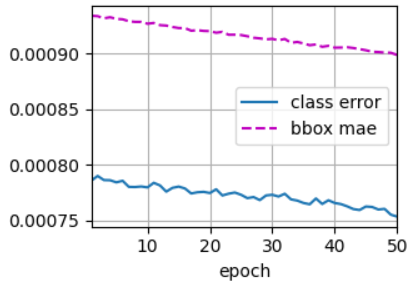
## 模型训练

图3.14 模型主要参数

整个训练过程共迭代了300个epoch；批大小初始为16，在200轮后为了更好的收敛降为8；利用矩池云的NVIDIA A16单核GPU进行训练。大约每十轮迭代用时1小时。模型的优化器为SGD优化器，初始学习率1e-3。对学习率采用十轮未下降除10的动态下降策略；损失函数方面，对分类采用交叉熵损失，预测框损失则采用Smooth L1损失。

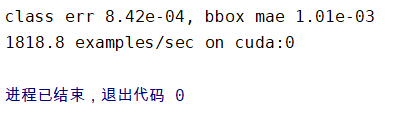
图3.15 训练硬件情况

在迭代到205轮左右时，模型趋近与收敛，平均损失如下：

图3.16 损失下降情况

基于训练结果，对训练过程做了以下改进：

1. 将学习率降低至1e-4；
2. 采用了焦点损失（FocalLoss）代替交叉熵损失；
3. 在网络末层增加了Dropout防止过拟合，更好的正则化；
4. 采用半精度训练；

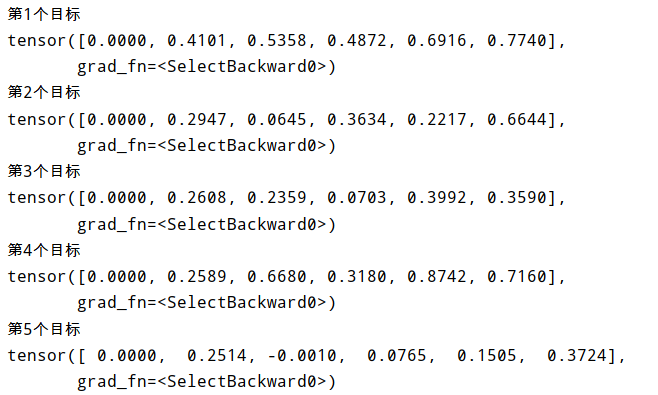
改进后模型从200轮开始迭代100轮，成功使loss进一步降低，提高了模型的预测能力，平均损失如下：

完整训练过程如下图所示：

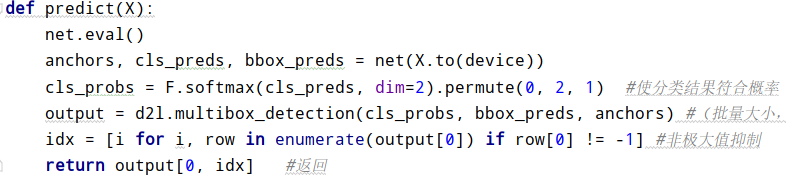
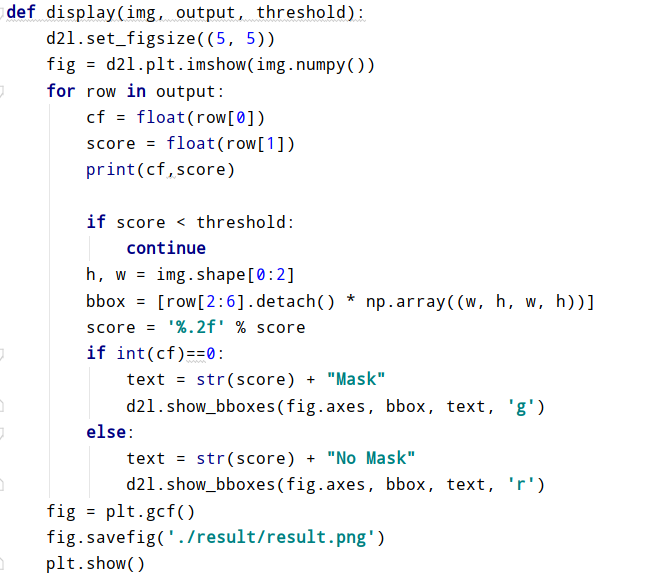
图3.17 完整训练过程

## 数据可视化

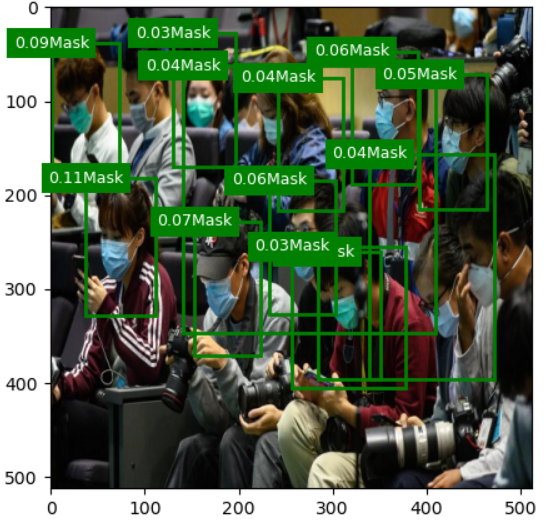
模型在预测时的输出结果是一系列预测的锚框（如下图所示）列表，如果想要直观的展示预测结果，还需要对输出进行可视化操作。

图3.18 模型输出结果

本文利用matplotlib库对结果进行可视化，具体代码如下：

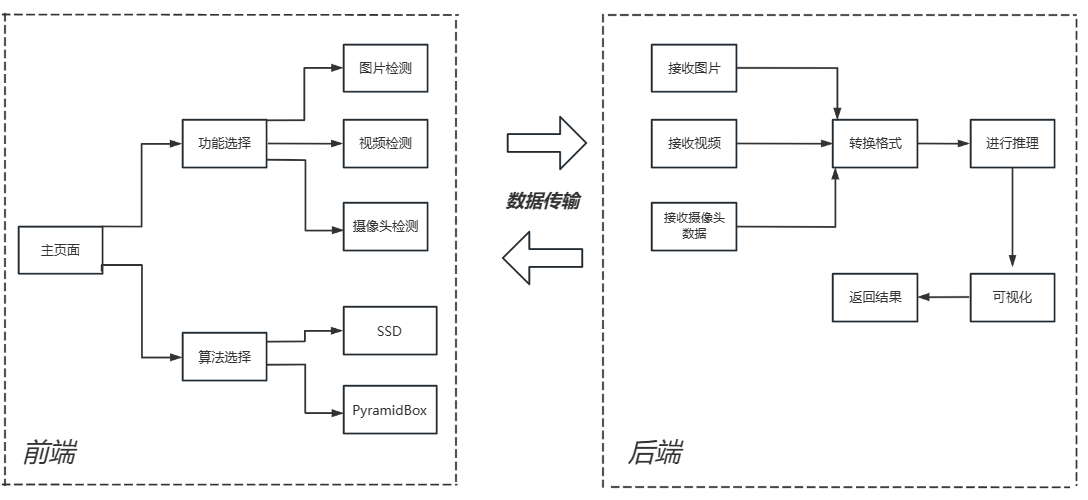
图3.19 可视化实现

在predict函数中，我们在得到预测结果后对结果执行非极大值抑制，在display中我们在利用matplotlib库绘制后将得分在threshold阈值以下的预测框过滤掉，从而保证结果的整洁。在下图的展示中，15个戴口罩人脸共检测到12个。达到了80%的检测率。

图3.20 可视化结果展示

# 系统实现

本项目使用了前后端分离的设计（如图4.1所示），在前端利用HTML编写页面，并运用Ajax、JavaScript技术进行前后端的数据传输。在后端，利用Flask搭建服务器，接收前端数据并进行推理，进而将结果返回前端页面。具体的流程如下：

图4.1系统流程

## 前端实现

我们设计了一个风格简约美观的前端页面，在启动flask开启本地5000端口后返回人脸检测与口罩识别系统的主页面，该页面可以选择三种检测功能：图片检测、视频检测与摄像头实时检测。除此之外，我们还可以选择检测的算法：SSD亦或是Pyramidbox。由于后者是基于预训练模型迁移学习而来，具有更完善的架构和更快的推理速度，建议在进一步开发时使用Pyramidbox算法，SSD可以用来演示demo。由于前端的功能主要有两个：向使用者提供可视化UI界面和向后端服务器传递数据，而不同的功能其传输的数据类别与具体流程也稍有差别，因此我们分三个小节分别展示不同功能下的代码实现。

图4.2 主页面

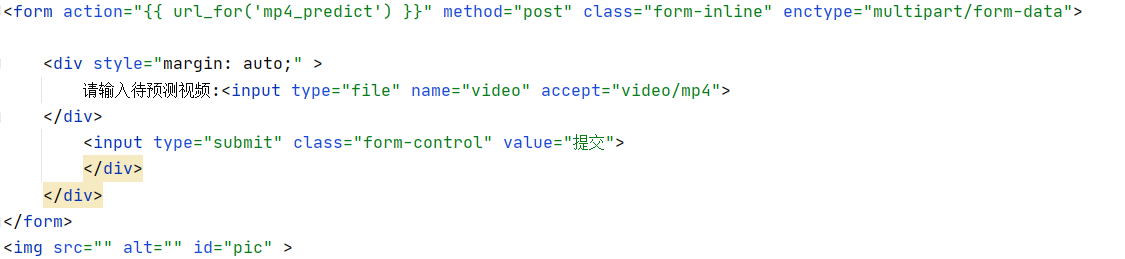
### 图像传输

我们利用<form action="{{ url\_for('main\_use') }}" method="post">标签将选择的算法及功能信息传输到后端的main\_use路由。若选择的是图片检测功能，服务器会自动为我们返回图片检测页面。

图4.3 图片检测页面

在此页面中，我们在选择图片点击提交按钮后，图片会以流文件的形式传输到后端服务器。服务器在处理完毕后会重新返回当前页面。此时在当前页面的下方会显示出待检测的图片和返回的检测结果。返回图片是利用flask的<img src={{ url\_for('static'}}>方法实现。

### 视频传输

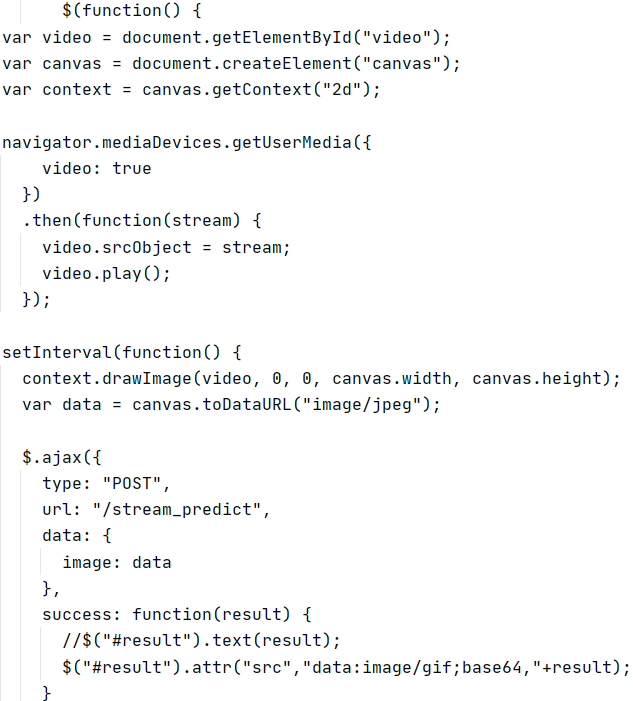
图4.4 视频检测页面

如果我们选择视频检测功能，那么前端会以mp4流文件的方式传送到后端。服务器会将视频转码后逐帧分解进行预测，并将预测结果实时返回至前端，这是利用Ajax的异步请求实现的。

图4.5 Ajax异步请求

### 摄像头传输

若我们选择摄像头实时检测，则前端会调用摄像头并将监控画面实时传输到后端进行分析。

图4.6 Ajax创建摄像头对象

该该段代码的实现逻辑如下：

1. 获取页面中的video元素，并创建一个canvas元素；
2. 通过调用canvas元素的getContext方法获取一个绘图环境context；
3. 调用navigator.mediaDevices.getUserMedia方法获取用户的摄像头视频流。getUserMedia方法返回一个Promise对象，当获取到视频流时会触发then回调函数；
4. 在then回调函数中，将视频流设置为video元素的srcObject属性，并调用video元素的play方法开始播放视频；
5. 使用setInterval方法定时执行一个函数。每次执行时，调用 context.drawImage方法将video元素中的视频帧绘制到canvas元素中，并通过canvas.toDataURL方法将绘制后的图像转换为base64编码的字符串数据。

## 后端实现

在该任务中，后端负责接收文件并转换为相应的格式，随后调用模型对数据进行预测，最后将预测的结果返回给前端。

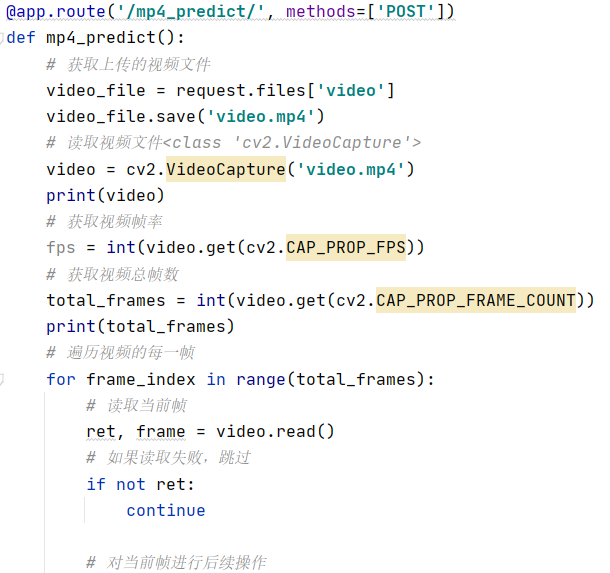
### 接收图片文件

图4.7 图片文件推理

在前端图片预测界面选择图片并上传后会调用pic\_predict方法，该方法读取前端传来的流文件并转换为numpy格式，随后利用cv的imdecode对图像进行解码，随后就可以放入模型中进行推理，返回结果至前端页面。

### 接收视频文件

后端调用mp4\_predict方法对视频文件进行推理。首先服务器在接受文件后会将其保存在服务器本地，随后调用cv2的VideoCapture方法遍历视频的每一帧并进行分析。最后用base64.b64encode(img\_stream).decode()方法传输每帧结果。

图4.8 视频文件推理

### 接收摄像头文件

摄像头实时推理与图片推理类似，不同的是需要传输每帧摄像头获取的画面，前端也需要配置Ajax进行实时传输。

图4.9 摄像头文件推理

# 系统测试

在后端服务器开启5000端口后，打开本地浏览器，会进入系统的主界面，在此页面我们可以选择要实现的功能以及用于检测的算法，如下图所示：

图5.1 主界面展示

## 图片检测

若我们选择图片检测功能，系统会返回图片检测界面（如图5.2），在选择文件进行上传后（如图5.3），服务器会返回待检测图片和检测结果在页面相应位置。

图5.2 图片界面展示

图5.3 结果展示

## 视频检测

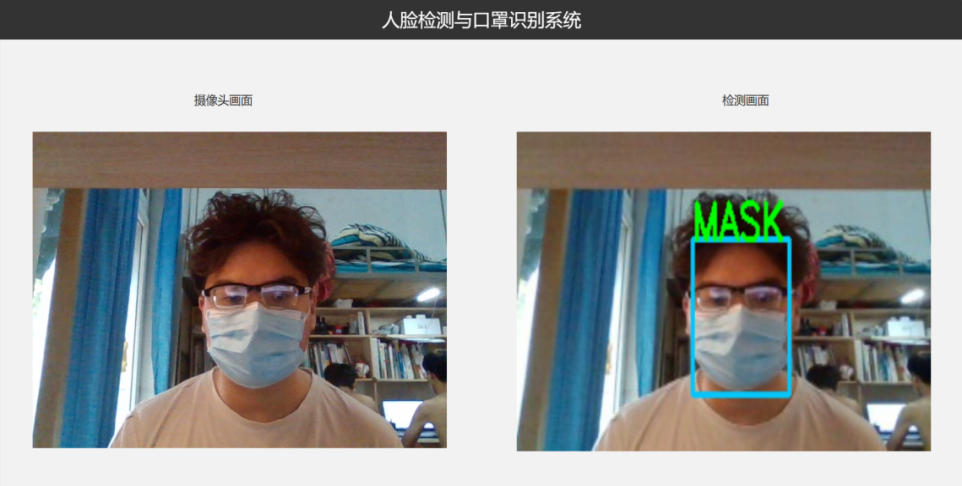
若我们选择视频检测功能，系统会返回视频检测界面（如图5.4），在选择视频文件进行上传后（如图5.5），服务器会返回对上传文件每帧的检测结果。

图5.4 视频界面展示

图5.5 结果展示

## 摄像头实时检测

若我们选择摄像头实时检测功能，系统会返回实时检测界面（如图5.6），此时JavaScript会自动调用设备摄像头并传输到服务器，服务器会返回对摄像头流每帧的检测结果。

图5.6 摄像头界面及结果展示

# 安卓部署

## 部署流程

图6.1 安卓部署流程

若想从模型训练到安卓开发具体落地，拢共需要三步：首先，我们需要利用TorchScript将.pt格式的模型权重转化为支持其他通用框架的中间格式（若使用的是Paddle框架，可以使用PaddleLite的opt工具对模型进行中间格式的转化）；其次，我们可以在Android Studio中使用转化后的模型进行界面编写、逻辑实现等编程；最后，我们可以将调试后的系统打包成apk文件导出到手机等端设备。

## 功能实现

### 界面实现

首先在生成原始项目后，我们需要在activity\_main.xml文件中编写程序的界面UI（如图6.2、6.3所示）。

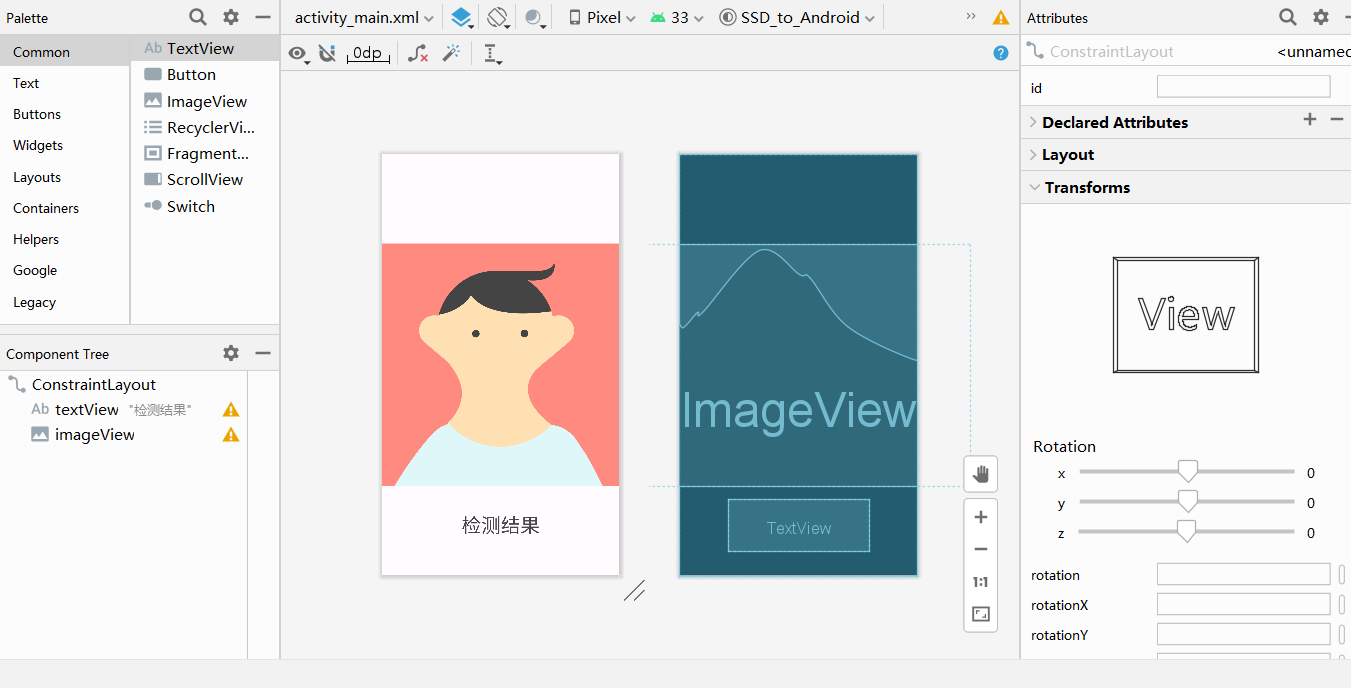
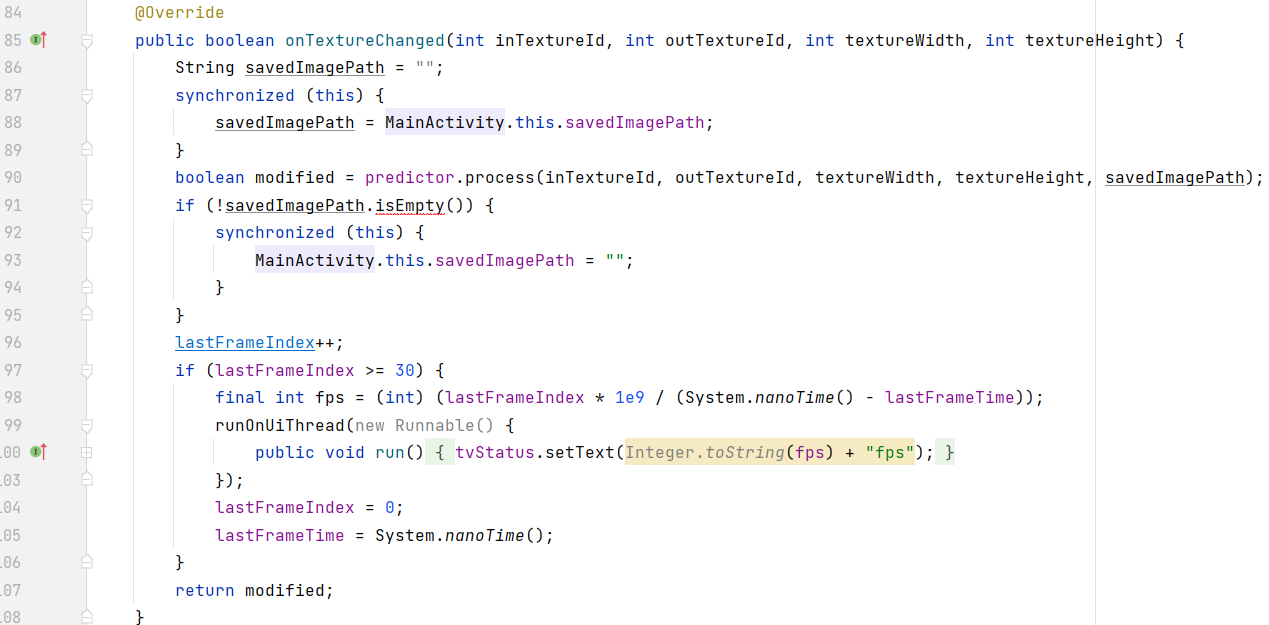
图6.2 activity\_main.xml的Design界面

图6.3 activity\_main.xml的Code界面

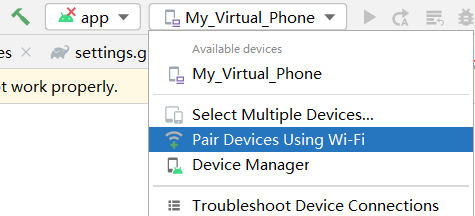
### 逻辑实现

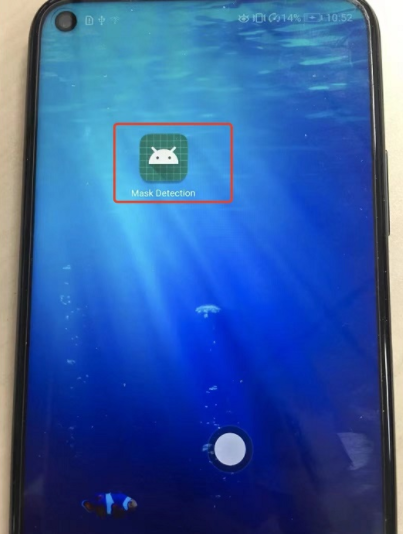
在MainActivity.java文件中编写程序的逻辑实现（如图6.4）。MainActivity实现APP的创建、运行、释放功能。在onTextureChanged函数中，我们通过调用predictor.process实现APP界面值传递和推理处理功能。

图6.4 MainActivity.java编写逻辑实现

### 打包为apk

在调试无误后，可运行程序编译并导出apk文件。

图6.5 选择连接的设备并进行编译

图6.6 打包成apk

## 结果展示

打开程序后会自动调用安卓端设备的摄像头进行实时推理，并可将推理结果保存到系统图库中，最后可在setting中设置模型检测的相应参数。

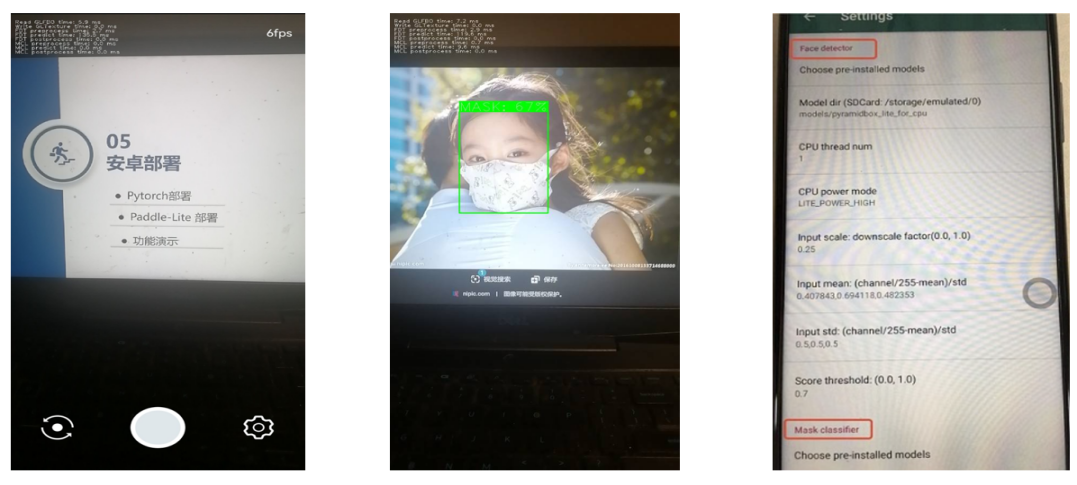
1.  （b） （c）

图6.7（a）程序主界面 （b）口罩检测 （c）setting设置参数

# 总结与展望

本设计总结了目标检测常用算法的网络结构、创新点和优缺点。跟随目标检测领域的发展历程总结了该领域的发展现状，并具体运用和加以改进来实现任务目标。具体来说，本项目参考相关文献，搭建了自己的SSD网络；利用信息检索收集了数量可观的数据集用于训练，对训练样本进行了正确重构和数据增强，提升训练效果；利用GPU对模型充分训练，并根据训练效果运用了dropout正则化、改进损失函数、使用半精度训练、合理地设置学习率，引入动量机制等对模型的训练进行了改进，进一步提升了模型的鲁棒性；使用matplotlib库对推理进行可视化展示，更加直观简洁；利用Flask和HTML相关知识搭建了前后端分离的口罩检测系统；支持图片、视频、摄像头等多种检测场景，具有应用的实际意义与需求。轻量级的模型结构，实现端到端推理，方便进一步部署到嵌入式设备、微信小程序等端侧设备。

鉴于本文水平有限，设备限制及时间等因素，本设计仍有许多改进的空间。首先，3200张的训练数据在目标检测领域仍属于较小规模，同时数据的质量还还可以进一步加强，且本实验的图像resize方式为等比例缩放，难免会一定程度上使目标形状失真，影响训练结果，可进一步使用补黑框等方法更好的对数据进行增强；其次，本设计实现的SSD模型的5层网络较小，与参考文献中的最优指标仍有一定差距，下一步可利用ResNet[20]等思想加强基础网络，还可融入特征金字塔的思想对特征进行融合，同时用更合适的方法设计锚框[21]；最后受限于时间等因素，本设计目前只实现了Web推理及安卓开发，后续可将模型部署到云端或其它端侧设备进行进一步开发。