计算机视觉导论期末报告

玻璃瓶检测及瓶口定位

姓名：颜帆

学号：20210980108

指导老师：陈雁秋教授

目录

玻璃瓶检测及瓶口定位

摘要：在计算机视觉众多的技术领域中，目标检测是一项非常基础的任务，图像分割、物体检测、关键点检测等通常都要依赖于目标检测，因此目标检测是一项基础而又非常重要的任务。目标检测的经典算法中，由于YOLO系列算法是单阶段的检测，在保持精度的同时速度非常快，在众多检测算法中脱颖而出。因此，本文章基于YOLOV3算法，在给定的带玻璃瓶的照片数据集中，自己标注真实标签，对网络进行训练并实现对自制数据集的玻璃瓶的检测和瓶口定位的任务。

关键词： 目标检测、自制数据集、YOLO

一.课题介绍

1.1 目标检测

在2012年的ImageNet图像识别竞赛中，Hinton率领的团队利用卷积神经网络搭建的AlexNet一举夺得了冠军，从此点燃了学术界、工业界等对于深度学习、卷积神经网络的热情。不到十年时间，深度学习在图像分类、目标检测、图像分割、人体姿态估计、目标追踪等一系列的计算机视觉领域中都收获了极大的成功，甚至在许多领域，计算机视觉算法都已经落地，为人们带来了极大的便利。

而在众多的计算机视觉技术领域中，目标检测又是一项非常基础的任务，图像分割、物体检测、关键点检测等通常都要依赖于目标检测。此外，由于每张图像中物体的数量、大小及姿态各不相同，也就是有非结构化的输出，这是与图像分类非常不同的一点，并且物体在图像中经常会有遮挡截断等问题，因此，目标检测极富挑战性，从诞生以来就始终是研究学者们最为关注的焦点领域之一。

目标检测的任务在二维图像中，是找到我们希望找到的目标及定位目标在图像中位置，即找到目标是谁和目标在哪儿，检测出目标物体出现的位置及对应的类别。通常对于图像中的一个目标，我们要求检测其输出五个量：物体的类别和物体边框的四个坐标，通常定义边框为一个矩形框，当然输出边框的中心点与宽高也能确定这个边框来确定目标物体的位置，两者是等价的。

1.2 发展历程

按照是否使用深度学习将目标检测分为传统的目标检测算法和基于深度学习的检测算法。

传统的目标检测算法：

传统的目标检测算法的步骤通常分为区域选取、特征提取与特征分类这三个阶段。

- 区域选取：首先选取图像中可能出现物体的位置，由于物体位置、大小都不固定，因此传统算法通常使用滑动窗口算法，但这种算法不可避免的带来了冗余框，导致计算量特别大。
- 特征提取：在得到目标位置后，通常使用人工精心设计的提取器进行特征提取，如SIFT和HOG算法等，由于提取器包含的参数少，为人工设计的，所以提取的特征往往鲁棒性较低，效果都不太好。
- 特征分类：最后，对上一步提取到的特征进行特征分类，使用的算法通常为支持向量机、Adaboost分类器等。

基于深度学习的检测算法：

基于深度学习的检测算法由于深度神经网络的大量参数可以提取出鲁棒性和语义性更好的特征，所以分类器性能也更优越。2014年的经典之作RCNN拉开了深度学习做物体检测的序幕。在RCNN的基础上，2015年的Fast RCNN实现了端到端的检测与卷积共享，Faster RCNN提出了锚框这一划时代的思想，将目标检测推向了一个高峰。而在2016年，YOLOv1实现了无锚框的一阶检测，SSD实现了多特征图的一

阶检测，对随后的目标检测算法产生了深远的影响。

基于是否使用锚框产生建议框，将基于深度学习的检测算法分为一阶段和两阶段检测算法。

- 两阶：两阶段算法的第一阶段通常专注于找出物体出现的位置得到建议框，保证准确的召回率，然后在第二个阶段专注于对建议框进行分类寻找更精确的位置。典型算法如Faster RCNN,两阶段算法通常精确度更高，但速度会比较慢。
- 一阶：一阶的算法通常将将两阶算法的两个阶段合二为一，在一个阶段里完成寻找物体出现位置与类别的预测，方法更为简单，速度一般比两阶网络更快，但精度会有所损失。典型算法如YOLO、SSD。

1.3 应用场景

由于检测性能的迅速提升，目标检测也是深度学习在工业界取得大规模应用的领域之一。以下列举了五个广泛应用的领域。

- **安防**：在安防领域中，有很大的趋势是将检测技术融入到摄像头中，形成智能摄像头，以海康威视、地平线等公司最为出名。广为人知的技术有人脸识别技术，在交通卡口，车站等已有了成熟的应用。
- **自动驾驶**：自动驾驶的感知任务中，行人、车辆和障碍物的检测尤为重要。由于涉及到安全性，对检测器的性能要求极高，尤其是召回率这个指标。此外，由于 车辆需要获取障碍物相对于自身的三维目标，通常在检测器后还需要增加许多的后处理感知模块。堪称人工智能应用的“珠穆朗玛峰”。

图1.1 自动驾驶领域中的目标检测

- **机器人**：工业机器人自动分拣中，系统需要识别出要分拣的各种部件，这是极为典型的机器人应用领域。
- **搜索推荐**：在互联网公司的各大应用平台中，物体检测无处不在。例如，对于包含指定物体的图像过滤、筛选、推荐和水印处理等等
- **医疗诊断**：基于人工智能和大数据，医疗诊断也迎来了新的春天，利用物体检测技术，我们可以更准确、更迅速地对CT、MR等医疗图像中特定的关节和病症进行诊断等等。

以分拣机器人为例：据新闻报道，京东在商品仓库里使用了700个分拣机器人，一天工作十小时就能正确的分拣20万件包裹，可以减少70%的分拣人力，也降低了分拣错误率。因此可以看出，精准的目标检测算法能给我们的生活带来极大的便利。

图1.2 京东仓库中的目标检测

1.4 课题目标

本次期末报告论文选择课题1，任务是在老师给定的带玻璃瓶的图像中（包含407张水滴型截面的玻璃瓶，361张椭圆形截面的玻璃瓶和181张圆形截面的玻璃瓶照片），检测出玻璃瓶的位置，并定位出瓶口所在的位置。并且对这个检测算法，追求更高的检测准确度和定位精度的同时，保证检测和定位的速度。

二.方法介绍

针对目标检测而言，经典之作Faster RCNN算法利用率两阶结构，先实现了感兴趣区域的生成，再进行精细的分类与回归，虽然出色地完成了目标检测任务，但是由于感兴趣区域的生成很耗时，限制了算法的速度，在更加追求速度的实际应用场景下，应用起来仍存在差距。

在此背景下，YOLO系列算法利用回归的思想，使用一阶网络直接完成了分类与位置定位两个任务，速度极快。并且YOLO从V1到V5，不断的改进，在保持速度的同时，还增长了检测的准确度。本次实验将采用YOLOV3算法，对玻璃瓶进行检测并对瓶口进行定位。接下来，详细介绍YOLO的思想和每一代的改进点。

YOLOv1

YOLOv1将输入的照片分为SxS个单元格，如果一个目标的中心落入格子，则该格子就负责检测该目标。在每个单元格中，预测B个边界框以及边界框的置信度。置信值代表该边界框包含一个目标的置信度，定义为$\Pr(\text{ Object })* \text{ IOU }_{\text{pred}}^{\text{truth}}$ ，即边界框含有这个物体的概率和这个边界框的准确度(IOU)的乘积。同时对每个边界框预测五个值(x,y,w,h,c)，分别为边界框的中心坐标和边界框的宽和高，边界框的置信度。 同时对每个格子还要预测概率值C，代表格子包含一个类别的物体的概率值，类别的置信分数定义为$\Pr(\text{ Class }_i|\text{ Object })*\Pr(\text{ Object })*IOU_{pred}^{\text{truth}}$ 。最终预测得到(SxSx(B*5+C))维的输出。然后针对预测出包含一个类别物体概率值高的边界框，使用NMS非极大值抑制的方法，即若两个框的重合度很高，大概率是一个目标，那就只取一个框，两个框的重合度的计算方法是利用交并比，两个框的交集面积/两个框的并集面积来计算的。这样，就避免了对同一个目标预测出多个框的错误。

网络结构图2.1如下：采用卷积网络来提取特征，然后使用全连接层来得到预测值。网络包含24个卷积层和2个全连接层，如图所示。对于卷积层，主要使用1x1卷积来做降维。对于卷积层和全连接层，除了最后一层采用线性激活函数，其它层的激活函数均用Leaky ReLU。 以Pascal VOC数据集为例，对每张图片裁剪到448x448，将输入的照片分为7x7个单元格，每个单元格预测2个边界框，即得到7x7x30的输出。

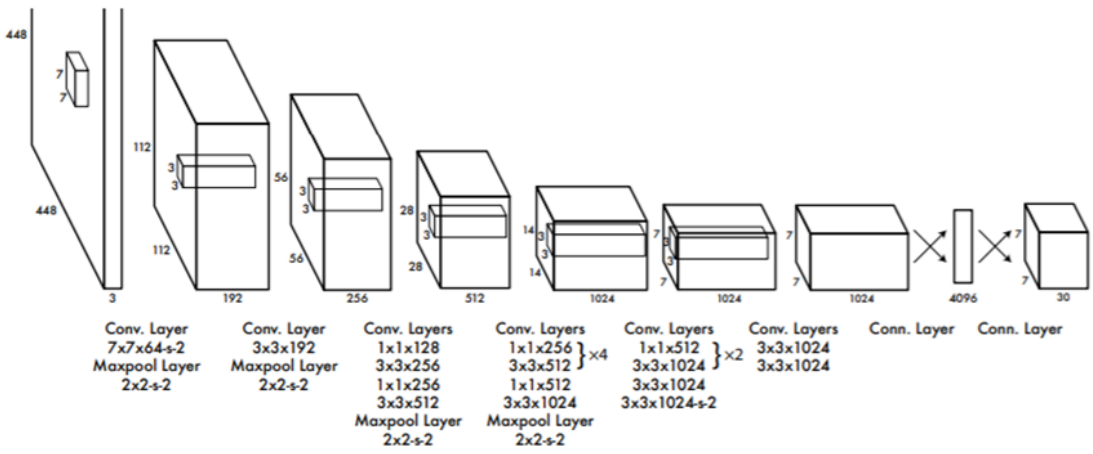

图2.1 YOLOv1模型网络结构图

网络的损失函数为：

$$
\begin{aligned}
&\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] \\
&\quad + \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i}\right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i}\right)^2 \right] \\
&+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{\text{obj}} \left(C_i - \hat{C}_i\right)^2 \\
&+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i\right)^2 \\
&+ \sum_{i=0}^{S^2} 1_{i}^{\text{obj}} \sum_{c \in \text{ classes}} \left(p_i(c) - \hat{p}_i(c)\right)^2
\end{aligned}
$$

损失函数一共由五个部分组成，均使用了均方差误差。公式中i代表第几个区域，一共S^2个区域；j代表某个区域的第几个预测边框，一共有B个预测框；obj代表该框对应了真实物体；noobj代表该框没有对应真实物体。五项分别代表的意义如下：

- 第一项为正样本中心点坐标的损失。λ_{coord}的目的是为了调节位置损失的权重，YOLOv1设置λ_{coord}为5，调高了位置损失的权重。
- 第二项为正样本宽高的损失。由于宽高差值受物体尺度的影响，因此这里先对宽高进行了平方根处理，在一定程度上降低了对尺度的敏感，强化了对小物体的损失权重。
- 第三、四项分别为正样本与负样本的置信度损失，正样本置信度真值为1，负样本置信度为0，λ_{noord}默认为0.5，目的是调低负样本置信度损失的权重。
- 最后一项为正样本的类别损失。

总体上，YOLOv1利用回归的思想，使用轻量化的一阶网络同时完成了物体的定位和分类，速度极快，出色的完成了目标检测的任务，该模型的优缺点如下：

模型优点：1)实时检测:在保证精确度的同时，达到了45 FPS(每秒传输帧率)，Fast YOLO达到了155 FPS并且精确度达到了当时最优实时检测SOTA的两倍。 2)普适性:在三个差别较大的数据集VOC和艺术品目标检测数据集(Picasso|People-Art）上都取得了很好的效果,说明模型鲁棒性强。

模型缺点：1）一个单元格只预测两个边界框和一种类别。导致模型对相邻目标预测准确度下降。这种限制会导致模型对于小物体，特别是对靠得近的小物体检测效果并不好。2)在损失函数中，大物体的位置损失权重和小物体的位置损失权重是一样的，会带来定位的不准确。因为我们知道对小物体而言，边界框的误差的影响是远大于大物体的边界框的误差带来的影响的。

YOLOv2

YOLOv2提出的DarkNet相比于v1版本的基础网络，进行了以下几点的改进：

- BN层：DarkNet使用了在卷积层后，激活函数之前加了一个BN层，有助于解决反向传播中的梯度消失与爆炸问题，可以加速模型的收敛，同时起到一定的正则化作用。
- 用连续的3X3卷积代替了v1版本中的7X7卷积，减少了计算量又增加了网络深度，此外还去掉了全连接层和Dropout层。
- Passthrough层：DarkNet还进行了深浅层特征的融合。这种特征融合有利于小物体的检测，也提升了模型性能。
- 除了模型上的改进，v2版本也是一个充满工程技巧的检测模型。包括使用多尺度训练和多阶段训练，使得模型在保持极快速度的同时大幅度提升了检测精度。

但是该模型也有一些不足，缺点如下：1）单层特征层：虽然使用了Passthrough层来融合浅层的特征，增强多尺度检测性能，但仅采用一层特征图做预测，细粒度仍然不够，对小物体检测性能提升有限。2）太工程化：v2整体的实现有较多工程化调参的过程，尤其是后续损失计算复杂，导致后续改进和扩展空间不足。

YOLOv3

针对v2的缺陷，2018年Joseph又推出v3版本，将当时一些较好的检测思想融入到了YOLO中，在保持速度优势的前提下，进一步提升了检测精度，尤其是对小物体的检测能力。提出的新网络结构DarkNet-53的改进点如下：

- 残差思想：DarkNet-53借鉴了ResNet的残差思想，在基础网络中大量使用了残差链接，因此网络结构可以设计得很深，并且缓解了训练中梯度消失的问题，使得模型更容易收敛。
- 多层特征图：通过上采样与通道拼接操作，融合了深浅层的特征，最终输出了三种尺寸的特征图，用于后续的预测。
- 无池化层：之前的网络用了最大池化层来缩小特征图的尺寸，而DarkNet-53没有采用池化的做法，通过步长为2的卷积核来达到缩小尺寸的效果。

值得注意的是，YOLOv3的速度并没有之前的版本快，而是在保证实时性的前提下，追求检测的精度。如果最求速度，YOLOv3提供了一个更轻量化的网络tiny-DarkNet，在模型大小与速度上，实现了SOTA(State of the Art)的效果。模型仍有些许的不足，主要体现在对于又遮挡和拥挤的这种较难的物体的检测，仍然做到很高的精度。

三.实验和讨论

3.1 实验步骤

1. 数据集制作

对于老师提供的带有玻璃瓶的照片，包含407张水滴型截面的玻璃瓶，361张椭圆形截面的玻璃瓶和181张圆形截面的玻璃瓶照片，一共949张照片放入images文件夹中，使用labelimg的软件对每张照片，找出玻璃瓶及瓶口所在的位置，用矩形框框出，并赋予标签，以得到训练数据的真实标签，用txt文档格式存储在labels文件夹中，由于一张图片只有一个玻璃瓶，因此一张图片对应的真实标签包括两行值，分别对应玻璃瓶和瓶口的位置。一行值代表五个数字，分别为对应的物体类别和边界框的四个坐标信息。由于数据集样本量较少，本文以所有的样本作为训练集输入网络进行训练，并且以所有样本为测试集进行测试。

2. 模型训练

使用yolov3作者提供的GitHub库的代码，按给定的实验设置来确定实验的超参数，先调用官网给出的预训练模型，在我们提供的数据集上进行训练，设定预测的种类为两种，分别为玻璃瓶和瓶口。模型在两个小时内就完成了949张照片的200个轮次的训练和测试，可以说是速度非常快了，并且效果也很不错，将在实验结果中进行展示。

图3.1为训练的模型的网络详细结构，每层的输入输出向量的大小，网络层次，参数多少和浮点数运算量如下：

```
Overriding model.yaml nc=80 with nc=2

                 from  n    params  module                                  arguments
  0                -1  1      3520  models.common.Focus                     [3, 32, 3]
  1                -1  1     18560  models.common.Conv                      [32, 64, 3, 2]
  2                -1  1     19904  models.common.BottleneckCSP             [64, 64, 1]
  3                -1  1     73984  models.common.Conv                      [64, 128, 3, 2]
  4                -1  1    161152  models.common.BottleneckCSP             [128, 128, 3]
  5                -1  1    295424  models.common.Conv                      [128, 256, 3, 2]
  6                -1  1    641792  models.common.BottleneckCSP             [256, 256, 3]
  7                -1  1   1180672  models.common.Conv                      [256, 512, 3, 2]
  8                -1  1    656896  models.common.SPP                       [512, 512, [5, 9, 13]]
  9                -1  1   1248768  models.common.BottleneckCSP             [512, 512, 1, False]
 10                -1  1    131584  models.common.Conv                      [512, 256, 1, 1]
 11                -1  1         0  torch.nn.modules.upsampling.Upsample    [None, 2, 'nearest']
 12          [-1, 6]  1         0  models.common.Concat                    [1]
 13                -1  1    378624  models.common.BottleneckCSP             [512, 256, 1, False]
 14                -1  1     33024  models.common.Conv                      [256, 128, 1, 1]
 15                -1  1         0  torch.nn.modules.upsampling.Upsample    [None, 2, 'nearest']
 16          [-1, 4]  1         0  models.common.Concat                    [1]
 17                -1  1     95104  models.common.BottleneckCSP             [256, 128, 1, False]
 18                -1  1    147712  models.common.Conv                      [128, 128, 3, 2]
 19          [-1, 14]  1         0  models.common.Concat                    [1]
 20                -1  1    313088  models.common.BottleneckCSP             [256, 256, 1, False]
 21                -1  1    590336  models.common.Conv                      [256, 256, 3, 2]
 22          [-1, 10]  1         0  models.common.Concat                    [1]
 23                -1  1   1248768  models.common.BottleneckCSP             [512, 512, 1, False]
 24      [17, 20, 23]  1     18879  models.yolo.Detect                      [2, [[10, 13, 16, 30, 33, 23], [30, 61, 62, 45, 59, 119], [116, 9
0, 156, 198, 373, 326]], [128, 256, 512]]
Model Summary: 283 layers, 7257791 parameters, 7257791 gradients, 16.8 GFLOPS
```

图3.1 YOLO模型网络结构

3. 实验设置

实验环境使用pytorch1.8.1版本，两块Tesla V100的GPU；

学习率设置为0.01；batch size 批尺寸设为16； epochs轮次设为200

4. 评价指标

对于一个检测器，我们需要制定一定的规则来评价其好坏，从而选择需要的检测器。对于目标检测而言，我们可以通过从预测框与真实框的贴合程度来判断检测的质量，通常使用**IoU(Intersection of Union)交并比**来量化贴合程度。交并比的计算方式如图所示，为蓝色预测框A与红色真实框B的交集部分黄色区域与并集部分绿色区域面积的比值，公式如下

$$IOU_{A,B} = \frac{A \cap B}{A \cup B}$$

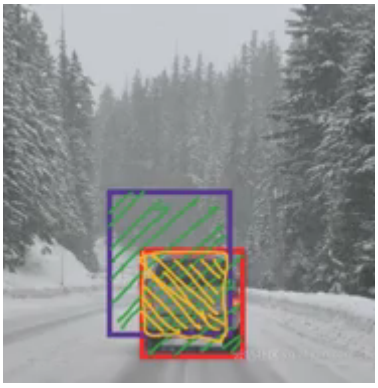

图3.2 交并比计算示例

可以知道交并比值大于等于0，小于等于1。并且我们知道，如果模型效果好，预测框应该和真实框特别接近，即交并比的值越大，说明两个框重合越好，预测效果越好。

对交并比而言，我们通常会设定一个阈值，如0.5，来判断预测框是正确的还是错误的。当两个框交并比大于0.5时，认为该预测框是一个有效的检测，否则属于无效的匹配。定义四个概念：

- 正确预测框：预测框与标签框正确匹配了，两个框的交并比大于阈值，记为TP(True Positive)。
- 误检框：将背景预测成了物体，这种框与所有的标签框的交并比都小于阈值，记为FP(False Positive)。
- 漏检框，为本来需要模型检测出的物体，但模型没有检测出，记为FN(False Negative)。
- 正确背景，本身是背景，且模型也没有检测出框,这种情况通常不需要考虑，记为TN(True Negative)。

由此，我们可以定义召回率：$R = \frac{TP}{TP+FN}$和准确率：$P = \frac{TP}{TP+FP}$,召回率表示一共检测出的标签框与所有标签框的比值；准确率表示当前遍历过的预测框中，属于正确预测边框的比值。对于一个检测器，可以用**mAP(mean Average Precision)**来评价模型的好坏，这里的AP指的是一个类别的检测精度，mAP则是多个类别的平均精度。

3.2 实验结果

1. 测试集上的检测结果展示

图3.3 模型预测边框

图3.4 照片真实边框

可以看到，对于模型而言，图3.3为模型预测出的边框，图3.4为照片真实的标签框。可以看出，模型基本上能检测出玻璃瓶和定位瓶口，由于预训练模型有在COCO数据集上训练时有识别瓶子这个种类，所以在我们的数据集上瓶子的定位十分准确，基本上置信度都为1且边界框和标签框基本重合，而对于较难检测的小物体瓶口，置信度也在0.9左右，但是也会出现如左上角图的情况：猜测是因为非极大值抑制时没能去掉那个重复框，导致了对一个瓶口预测了两个框出来。

模型也会出现将物体种类预测出错的情况，如图3.5，可能是因为训练样本过少的原因，因此模型仍有改善的空间。

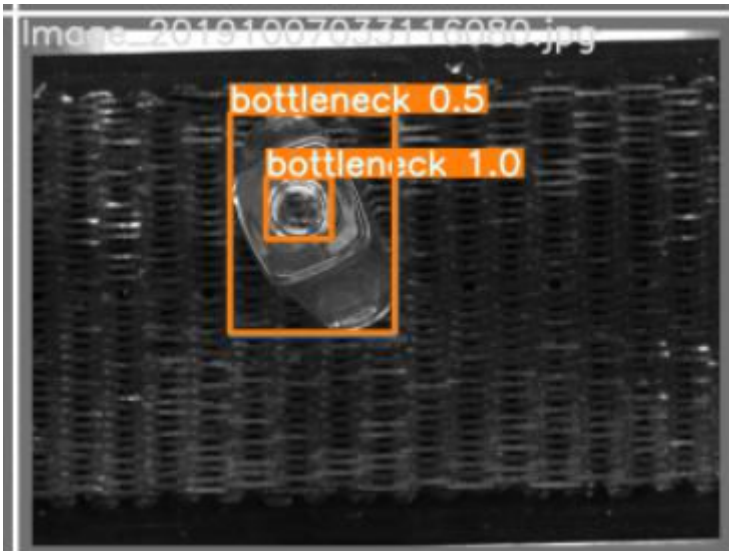

图3.5 模型预测失误的样本

2.模型评估

混淆矩阵：从图3.6的混淆矩阵中可以看出，模型的漏检框为0，说明模型把所有的标签框都框出来了，召回率很高，但是存在一些误检框，将背景预测为了物体；整体上看，标签为bottle预测为bottle的占0.92，对瓶子的预测准确度很高，对瓶口的预测准确度相对较低，可能是因为瓶口是更小的物体或者预训练模型没有这个类别的缘故，但总体上而言，模型的预测效果还是挺好的。

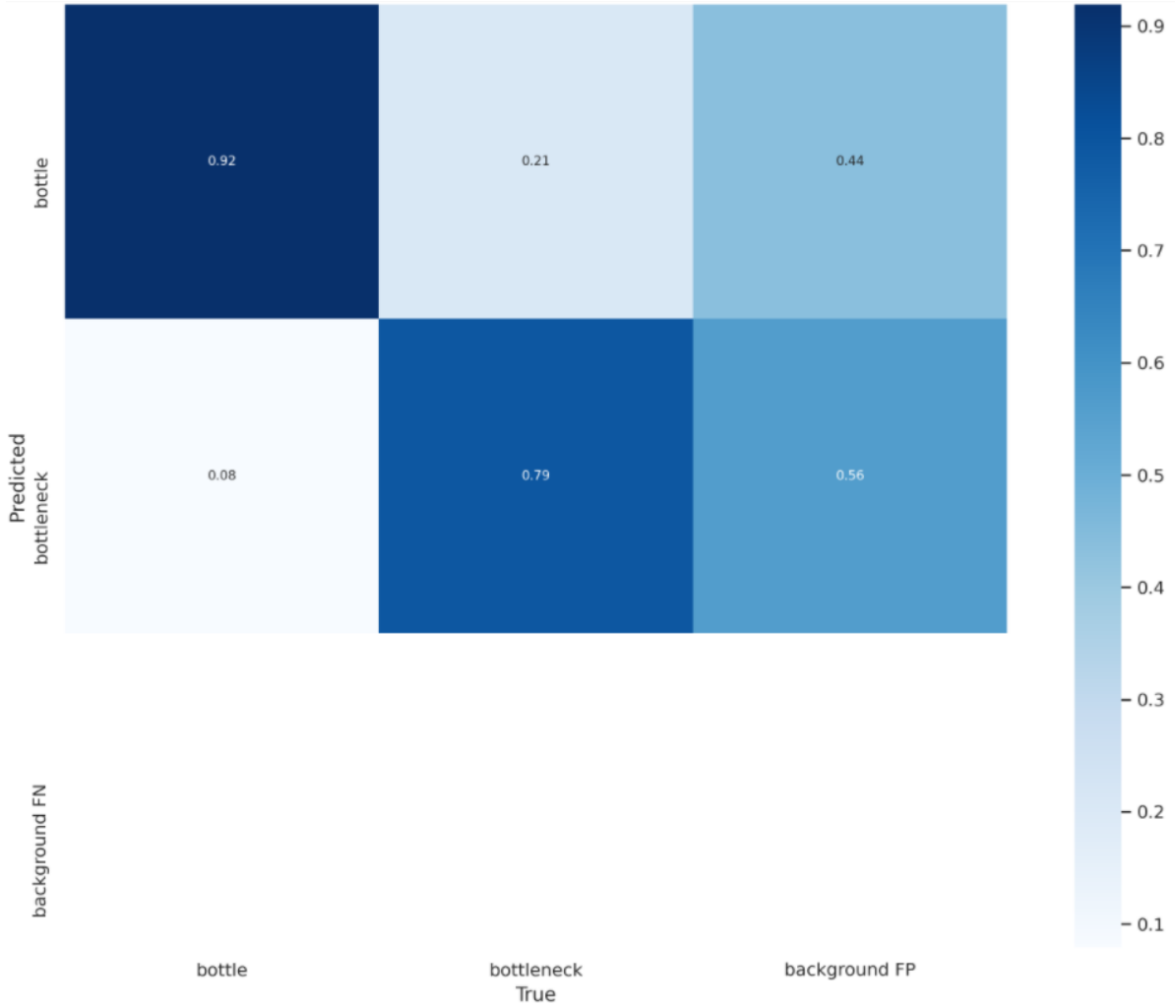

图3.6 模型结果的混淆矩阵

准确率、召回率和F1分数：

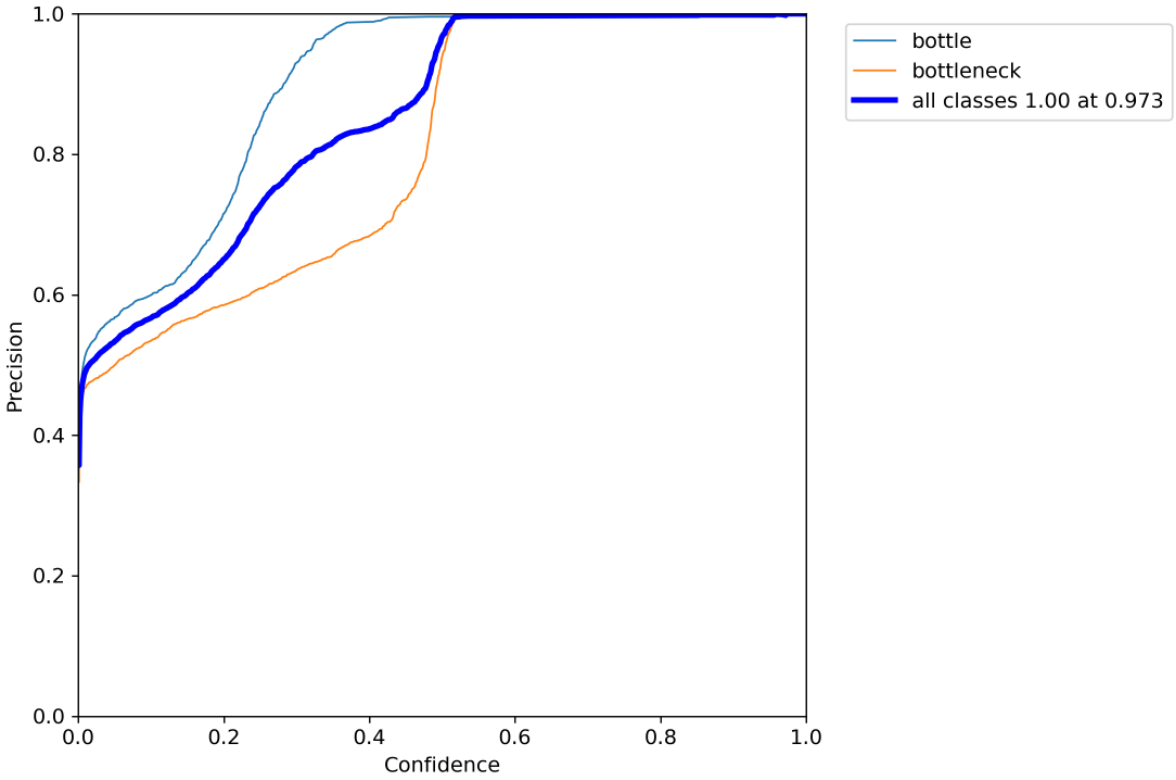

图3.7 准确率-置信度图

从准确率上看模型，由于准确率的定义$P = \frac{TP}{TP+FP}$,我们知道如果尽可能的减少FP误检框的数量，P值就会接近1。即当模型要求的置信度越高，（置信度要求越高，就越有把握认定预测框为标签框，但同时会减少预测框的数量），预测框为标签框的概率就越高，所以误检框就会越少，因此，随着置信度的提升，模型准确率就会越高，从图3.7中可以看到，在置信度逐渐大于0.5后，模型的准确率就基本上接近于1了，说明模型预测为标签框的准确度很高。

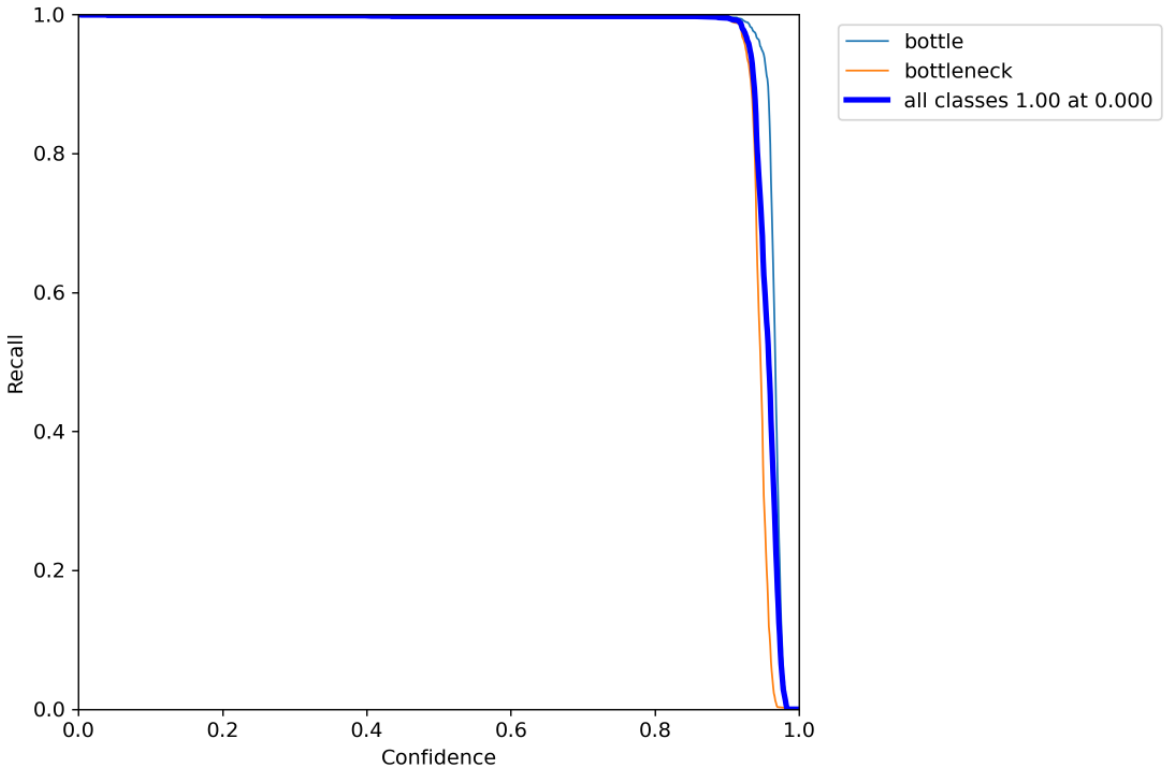

图3.8 召回率-置信度图

从召回率上看模型，由于召回率的定义$R = \frac{TP}{TP+FN}$，我们知道如果尽可能的减少FN漏检框的数量，R值就会接近1.即当模型要求的置信度越低，（置信度要求越低，就越不会漏检标签框，但同时会增加预测框的数量），漏检标签框的概率就越低，所以漏检框就会越少。随着置信度的下降，模型的召回率就会越高，从图3.8中可以看到，在置信度小于0.8时，模型的召回率都是接近1的，说明模型的漏检框很少。

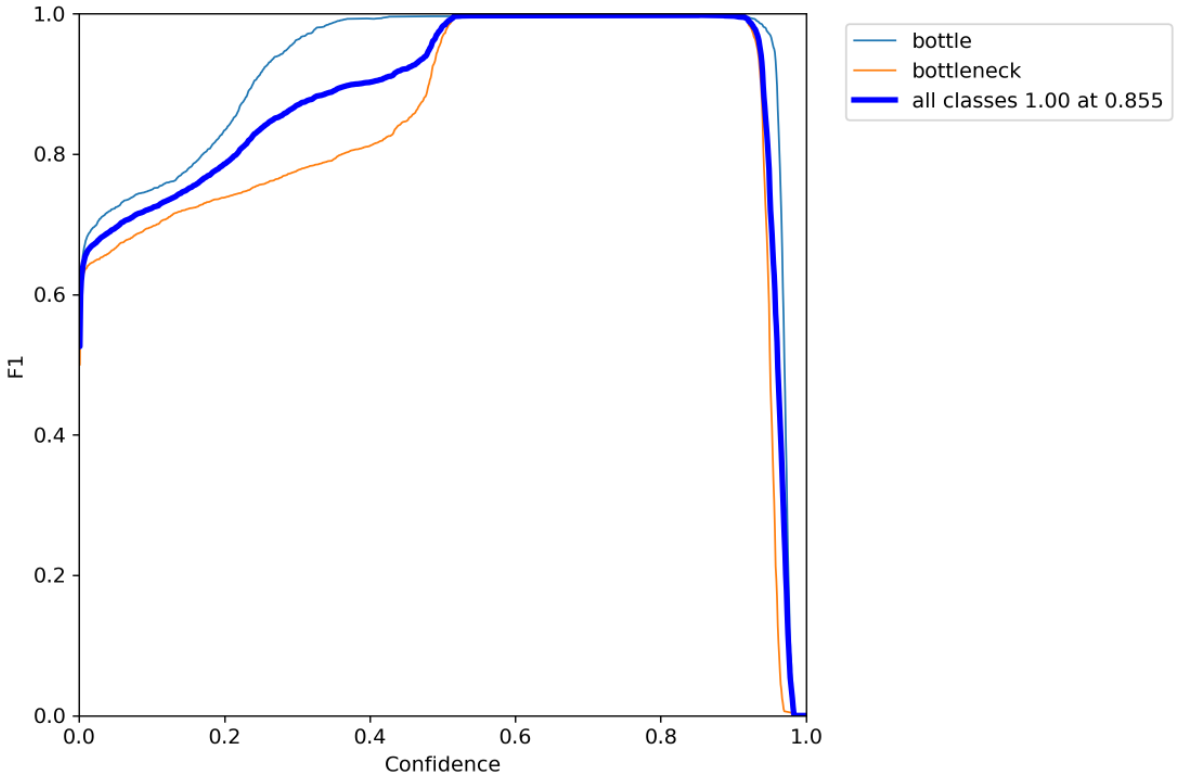

图3.9 F1分数-置信度图

从F1分数上看模型，由于F1分数的定义 $F_1 = 2 * \frac{PR}{P+R}$, 它兼顾了分类模型的准确率和召回率，取值在[0,1]之间，越接近1说明模型分类效果越好，从图3.9中可以看见，在置信度大于0.5和小于0.9之间时，模型的F1分数接近1，说明模型的预测效果非常好。

3.训练过程可视化

使用了Weights & Biases工具进行训练过程的可视化：

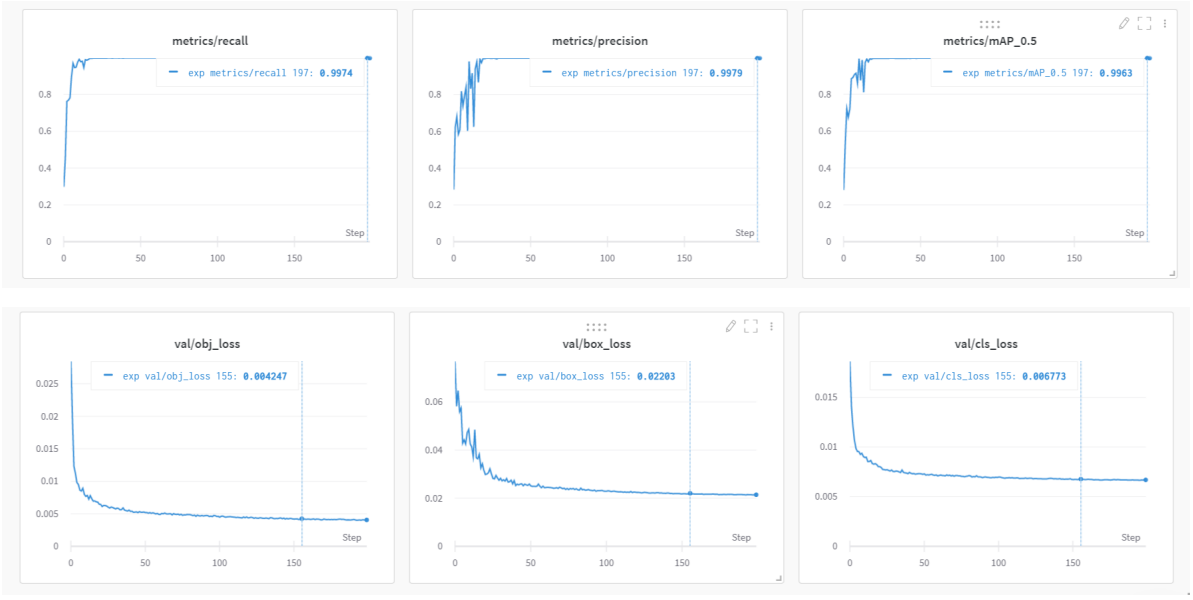

图3.10 WB训练过程可视化

从图3.10中可以看到，在第197轮次的训练中，模型的准确率达到了0.9979，召回率达到了0.9974，mAP达到了0.9963，说明模型的预测效果非常不错。

Tensorboard可视化：

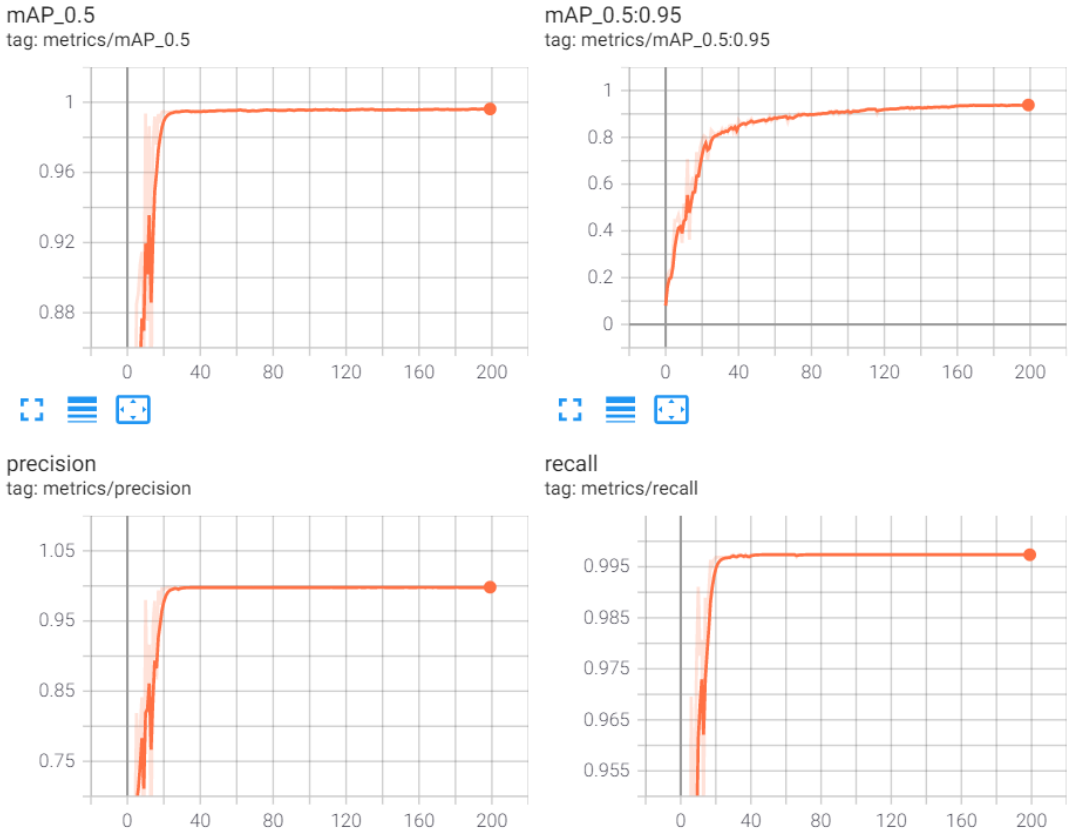

图3.11 Tensorboard训练过程可视化

3.3 选择方法的效果及优缺点

由上述结果可以看出，选择的方法YOLOv3很好的预测出了瓶子及准确的定位了瓶口的位置，各项评价指标比如mAP达到了快接近1的0.996，说明模型预测得非常准确，也有可能是因为训练集和测试集都是相同的照片，有过拟合的嫌疑。但是总体而言，YOLOv3模型在速度非常快的情况下（两个小时预测完九百多张图片的200个轮次的训练），很好的检测出了我们的目标玻璃瓶和定位了瓶口所在的位置。存在许多优点的同时，模型仍存在着一些缺点，比如模型仍然会分类错误，比如把瓶子预测成了瓶口，或者在一个瓶口处预测出了两个框，且对一些遮挡的物体的预测精度仍然有限，仍有许多改进的空间值得去探索和优化。

参考文献

[1]Redmon, Joseph, et al. "You only look once: Unified, real-time object detection." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.

[2]Redmon J, Farhadi A. YOLO9000: better, faster, stronger[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2017: 7263-7271.

[3]Redmon J, Farhadi A. Yolov3: An incremental improvement[J]. arXiv preprint arXiv:1804.02767, 2018.