所在组别	2021 年中国高校大数据挑战赛	参赛编号
研究生组		bdc210717

基于改进 YOLOv4 的口罩佩戴检测

摘要

新冠疫情爆发以来，对全球人民的生活造成了巨大的影响。在人多密集的公共场所，正确佩戴口罩可有效阻断新冠病毒的传播。使用人工智能的方式实现口罩的自动检测，不仅克服人工效率低下的问题，也大大节约了公共资源。

本文在实现“佩戴口罩”、“未佩戴口罩”和“未正确佩戴口罩”3 种状态的人脸可视化检测的基础上，针对当前口罩检测算法难以区分是否正确佩戴口罩这一难点，提出了基于 YOLOv4 的模型融合算法，将传统图像处理方法融合进 YOLOv4 原始模型，首先将人脸的口罩部分提取出来，再根据口罩的像素块占整个人脸区域像素块的比例，以及口罩像素块的最小外接矩形长宽比来进行判断，优化了模型的检测效果。同时，采用 k-means 聚类方法选取 anchor box 形状，对人工标注的不同目标聚类计算出最具有代表性的一组 anchor box，从而减少预测框拟合实际框的时间，大大提高拟合的精度和速度。此外，从“未正确佩戴口罩”、“佩戴口罩的侧脸”、“人脸大比例占比图像面积”、“防毒面具等异形口罩”这 4 个方面扩充了官方数据集，提升了模型的鲁棒性。在人脸 3 种口罩佩戴状态的检测上，与原始 YOLOv4 算法相比，查准率平均提高了约 1.52%，“未正确佩戴口罩”状态的 AP 值提高了 10%，三类状态的 mAP 值提高了约 4%，漏检率平均下降了约 3.33%。

关键词：模型融合，图像处理，YOLOv4，口罩检测

一、问题重述

1.1 问题背景与提出

1.1.1 问题背景

2020 年新冠疫情大爆发给人类带来了全新的未知的挑战，病毒威胁着我们的生命安全，对我们的日常生活产生了重大影响。虽然现在国内疫情不像最初爆发时那么严重，但我们仍然不能放松警惕。飞沫传播是新冠病毒扩散的主要途径，因此戴口罩是当前国内外采用的主要预防措施，是抑制疾病在人员密集的公共场合快速传播和保护自身安全健康的重要手段，也是控制新冠肺炎传播最便捷、最有效的措施和方法。

疫情期间正确的佩戴口罩可以有效的防止病毒的传播，而当下通过人工的方式，在地铁、机场、车站等公共场所配置大量人员对人流的口罩佩戴情况进行监督，不仅效率低下，而且造成了大量公共资源的浪费。随着计算机视觉的进步，通过人工智能途径实现口罩佩戴检测的技术也取得了相应的发展。

1.1.2 口罩佩戴检测的目标与意义

人脸佩戴口罩的自动化识别可以有效检测人群佩戴口罩情况，通过运用机器学习及模式识别技术，对人流量大的公共场合进行实时检测，不需要人工操作，自动地对图片或者视频图像信息进行分类识别。在实际应用中，还可以将目标检测与预警系统相结合，当检测到人员没有佩戴口罩或者口罩佩戴不正确时，立即触发警报提示，可以有效地协助管理人员工作，减少人力监督的成本，在日常生活中具有重要意义。

1.1.3 国内外研究现状

人脸佩戴口罩识别属于计算机视觉中的目标检测范畴，是在人脸检测算法基础上进行的。早期的人脸检测算法使用了模板匹配技术，即用一个人脸模板图像与被检测图像中的各个位置进行匹配，来确定某个位置是否有人脸[1]。2001 年 Viola 和 Jones 设计了一种人脸检测算法（VJ 框架）[2]，使得人脸检测速度较之前的方法有了数量级的提高，奠定了基于 AdaBoost 目标检测框架的基础。

近年来，随着深度学习目标检测算法的迅猛发展[3]，目标检测主要可以分为两类：一类是基于候选框进行分类的 Two-stage 方法，如 Fast RCNN 算法，虽说准确率比较高，但检测速度较慢；以及直接通过卷积神经网络 CNN 检测目标的 One-stage 方法如 SSD 系列算法、YOLO 系列算法等，与 Two-stage 算法相反，检测速度比较快，准确率较低。

在 YOLO 系列算法中，YOLOv1 创造性地将一张图片完整地输入进网络，用一阶结构实现目标的定位与分类两个任务，相比于 Fast RCNN 算法在检测速度上得到大幅提升，但是精度不高，针对小目标的检测也不够理想；YOLOv2 是 YOLOv1 的升级版，通过去掉计算量大的全连接层和池化层，引入锚机制和 BN 层，在速度和精度上进一步提升；YOLOv3 是目前使用较广泛的算法，它将 backbone 网络架构从 darknet-19 换成了 darknet-53，将原来的单标签分类改进为多标签分类，同时采用多尺度预测输出，加强了对小目标检测的精确度。YOLOv4 的整体架构与 YOLOv3 类似，但多加入了 CSP 结构、PAN 结构，对算法的各个部位都做了改进优化[4]。而 YOLOv5 并没有对 v4 有很大的改进，所以本文基于 YOLOv4 算法模型，针对“未正确佩戴口罩”状态难以区分这一难点，融合传统图像处理方法提取口罩在人脸中的面积占比、宽高比及置信度来辅助决策，提高了检测算法的精度和效率。

1.2 问题要求与分析

问题 1：根据格式为 xml 的标签文件中的人脸信息，编写程序读取数据并将人脸框和口罩佩戴状态进行展示。其中，每个<object>元素代表一张人脸；<name>表示该人脸处于“佩戴口罩/未佩戴口罩/未正确佩戴口罩”的其中一种状态；<bndbox>记录了该人脸的包围盒的左上、右下两个顶点的坐标。

分析：该问题主要实现标签数据在图片上的可视化操作。可分为两步进行，第一步**读取数据**：拟采用 python 进行编程，利用其自带的函数，获取文件路径、读取图片信息和标签信息。第二步**标签可视化**：拟设计一个标签绘制函数，基于 python 自带的 cv2.rectangle 函数和 cv2.putText 函数来绘制人脸矩形框和文本框，实现人脸佩戴口罩状态的显示。

问题 2：根据 653 张图片和对应的标签信息设计人脸口罩检测算法，实现能检测任意一张图片中存在的人脸位置和口罩佩戴情况，并用如下量化指标 A 对算法进行检验：

$$\text{量化指标 A} = \frac{\text{被正确分类的人脸的数量}}{\text{标签文件中包含的所有人脸数量}}$$

对某张人脸来说，当且仅当 IoU 大于 0.45 时，才被认为被正确分类。

分析：该问题是设计口罩检测的核心算法，可分为设计和优化两个阶段。在**设计阶段**，首先需查阅相关文献，了解现有目标检测算法的优、缺点。再针对口罩检测的特点设计模型，将“附件 1 训练样本”以一定比例划分为训练集和验证集，并用训练集来训练所提出的模型。最后将验证集输入训练后的模型来预测分类结果，并将结果与标签相比较，得到每张图片的 IoU 值以检验所提出算法的性能。在**优化阶段**，分析算法模型在一些图片上没有正确分类的原因，通过融合传统图像处理方法提取特征来辅助决策，同时增大样本数据以优化模型参数。

问题 3：使用检测算法对“附件 2 测试样本”文件夹下 test_images 中包含的图片进行检测，并将检测结果汇总填写到“赛题 B 提交结果.xlsx”表格中。

分析：该问题是使用口罩检测算法对测试样本进行检测，将输出检测结果并做汇总。

二、模型假设

1.假设人员佩戴了口罩，但口罩没有遮住鼻子，视为“不正确佩戴口罩”；如果遮住了鼻子，则视为“正确佩戴口罩”。

2.假设光线条件充足，图片清晰可见。

三、符号说明

符号名	符号解释
t_x	预测框中心点的 x 坐标偏移量
t_y	预测框中心点的 y 坐标偏移量
t_h	预测框高度的尺度缩放变量
t_w	预测框宽度的尺度缩放变量
$b_x^{'}$	预测框中心点到图像左边界的距离

符号名	符号解释
$b_y^{'}$	预测框中心点到图像上边界的距离
$b_h^{'}$	预测框的高
$b_w^{'}$	预测框的宽
C_x	单元格到图像左边界的距离
C_y	单元格到图像上边界的距离
P_h	Anchor box 的高
P_w	Anchor box 的宽

四、模型建立与求解

4.1 问题 1 模型建立与求解

4.1.1 问题 1 分析与求解思路

问题 1 要求根据格式为 xml 的标签文件中的人脸信息，编写程序读取数据并将人脸框和口罩佩戴状态进行展示。通过对数据集的了解，我们拟分成两步进行。

第一步，读取数据：利用 Python 自带的函数获取图片和标签的文件路径，再使用 cv2.imread 函数读取图片信息，xml.etree.ElementTree 函数读取标签信息。

第二步，标签可视化：设计一个绘制标签的函数，使用 cv2.rectangle 函数在图像上绘制一个简单的矩形，使用 cv2.putText 函数在矩形上绘制一个文本框。

4.1.2 环境配置及实现过程

环境配置情况如表 1 所示：

表 1 环境配置情况

名称	配置参数
操作系统	Ubuntu 18.04
GPU 型号	RTX 2080
编程工具	PyCharm
编程语言	Python 3.7
深度学习框架	PyTorch 1.6.0

4.1.3 基本实现流程

根据分析，定义 draw_label()为标签绘制函数，采用 cv2.rectangle 函数绘制人脸矩形框，采用 cv2.putText 函数在矩形框上部显示人脸佩戴口罩状态的文本。其中，颜色与口罩佩戴状态对应关系如表 2 所示：

表 2 颜色与口罩佩戴状态对应关系

颜色	口罩佩戴状态
红色 RGB=（255, 0, 0）	佩戴口罩
绿色 RGB=（0, 255, 0）	未佩戴口罩
蓝色 RGB=（0, 0, 255）	未正确佩戴口罩

问题 1 基本流程框图所图 1 所示：

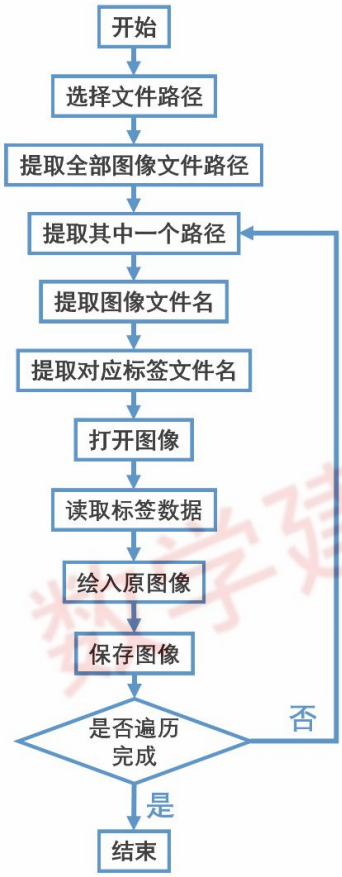
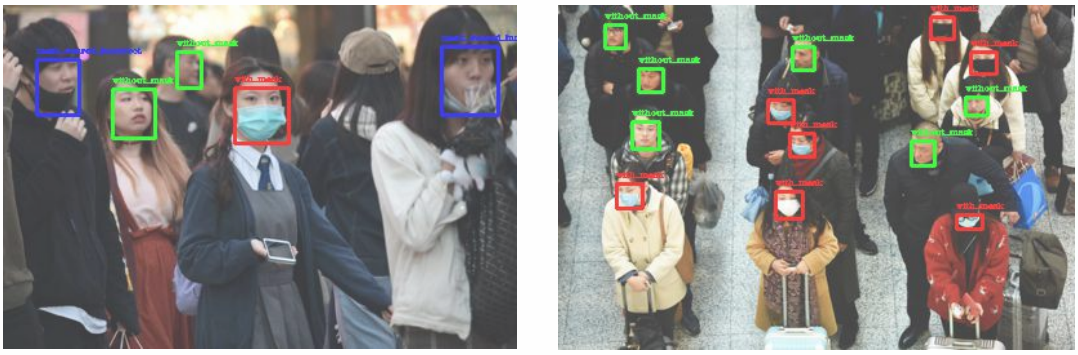


图 1　问题 1 实现流程框图

4.1.4 结果展示与分析

按照上述流程框图，在配置好的环境中编写代码，实现可视化的输出结果，这里以“250.png” 和“477.png”两张图片为例进行展示，如图 2 所示。



(a) 250.png　　(b) 477.png

图 2　可视化标签展示的示例图

分析：通过对“附件 1 训练样本”中 653 张图片的处理，使每一张图片都清晰地展示出对应的标签信息。在这个过程中，不仅熟悉了 Python 相关函数，在代码编写的过程中加深对赛题的理解，也为问题 2 算法设计打下了基础。

4.2 问题 2 模型建立与求解

4.2.1 问题 2 分析与求解思路

由于人脸口罩检测算法是在人脸检测算法的基础上进行的，故我们考虑各种人脸检测算法对于解决问题 2 的有效性。人脸口罩检测算法要解决以下几个核心问题：

1）人脸可能出现在图像中的任何一个位置

2）人脸口罩在图像中可能有不同的大小

3）口罩在图像中可能有不同的视角和姿态

4）口罩可能部分被遮挡

5）口罩是否正确佩戴

考虑到本项目的实际应用场景中需要处理大量数据、且对实时性要求较高，所以我们采用 One-stage 方法。YOLOv1 到 YOLOv4，每一版本都会比之前在某些方面有改进和优化，使得 YOLOv4 算法发展到今天已经成为目标检测算法的一个重要力量。而 YOLOv4 和 YOLOv3 相比，在输入端、backbone 主干网络、Neck 和 Prediction 各个部分都进行了优化改进。由于 YOLOv5 相比于 YOLOv4 没有本质上的改进，故本论文基于 YOLOv4 口罩佩戴检测算法[5]进行了改进，从而提高了检测的精度。

4.2.2 数据准备和处理

数据集一部分来源于举办方提供的训练样本，另一部分来源于网络上开源的模拟人脸佩戴口罩数据集，还有一部分样本为自己手动收集与标注。由于提供的数据中“未正确佩戴口罩”的样本数量较少，基于原有的训练样本得到的模型对于“未正确佩戴口罩”这一类别的识别效果并不是很好，所以我们采用了 Simulated Masked Face Recognition Dataset (SMFRD) [6]中正确/未正确佩戴口罩的一部分样本，样本数量为 181 个。同时，还在网络上选取了一些戴/没戴口罩的侧面人脸样本 50 个，以及一些脸部在图片中占较大面积的人脸样本 36 个，以增加训练模型的普遍性。

4.2.3 基于 YOLOv4 的目标检测模型

4.2.3.1 YOLOv4 的五个基本组件

1）CBM：由 ConV+BN+Mish 激活函数组成

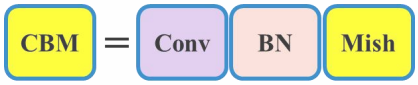


图 3　CBM 组件示意图

2）CBL：由 ConV+BN+Leaky_relu 激活函数组成

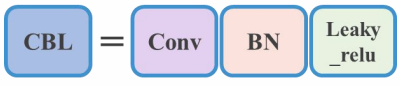


图 4　CBL 组件示意图

3）Res unit 模块：借鉴了 Resnet 网络中的残差结构

图 5　Res unit 模块示意图

4）CSPX：由卷积层和 x 个 Res unit 模块连接而成

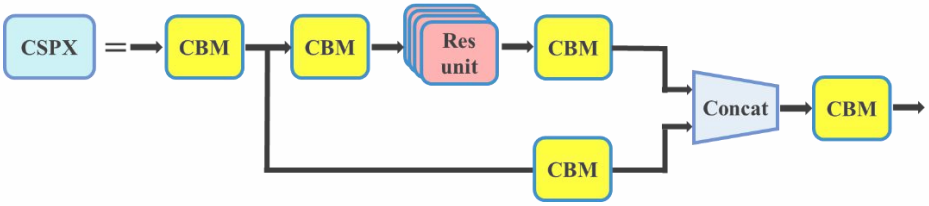


图 6　CSPX 组件示意图

5）SPP 模块：采用不同尺寸的 Maxpool 方式进行多尺度融合

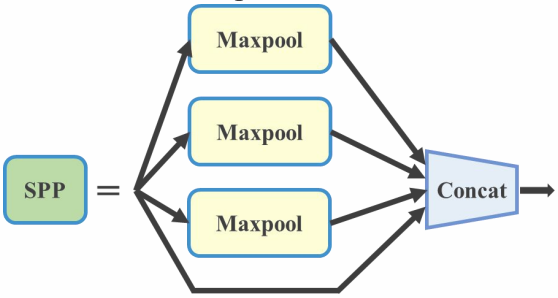


图 7　SPP 模块示意图

4.2.3.2 YOLOv4 算法核心基础内容

（1）输入端对数据的处理

选取官方所提供的“附件 1 训练样本”中的 322.png 进行说明。由于输入的图片尺寸大小不一，所以首先采用等比例变换的方式归一化图片尺寸。

图 8　“附件 1 训练样本”中 322.png 的等比例变换示意图

如图 8 所示，将图片等比例缩放后放到全零矩阵中间的位置，图片上多余的黑色部分会被当成背景处理。

由于数据集中的大中小目标占比并不均衡，所以接下来对图片进行 Mosaic 数据增强，将 4 张图片使用随机缩放、裁剪、排布的方式进行拼接，数据增强后的效果如图 9 所示：

图 9　Mosaic 数据增强效果图

（2）multi-scales 多尺度融合

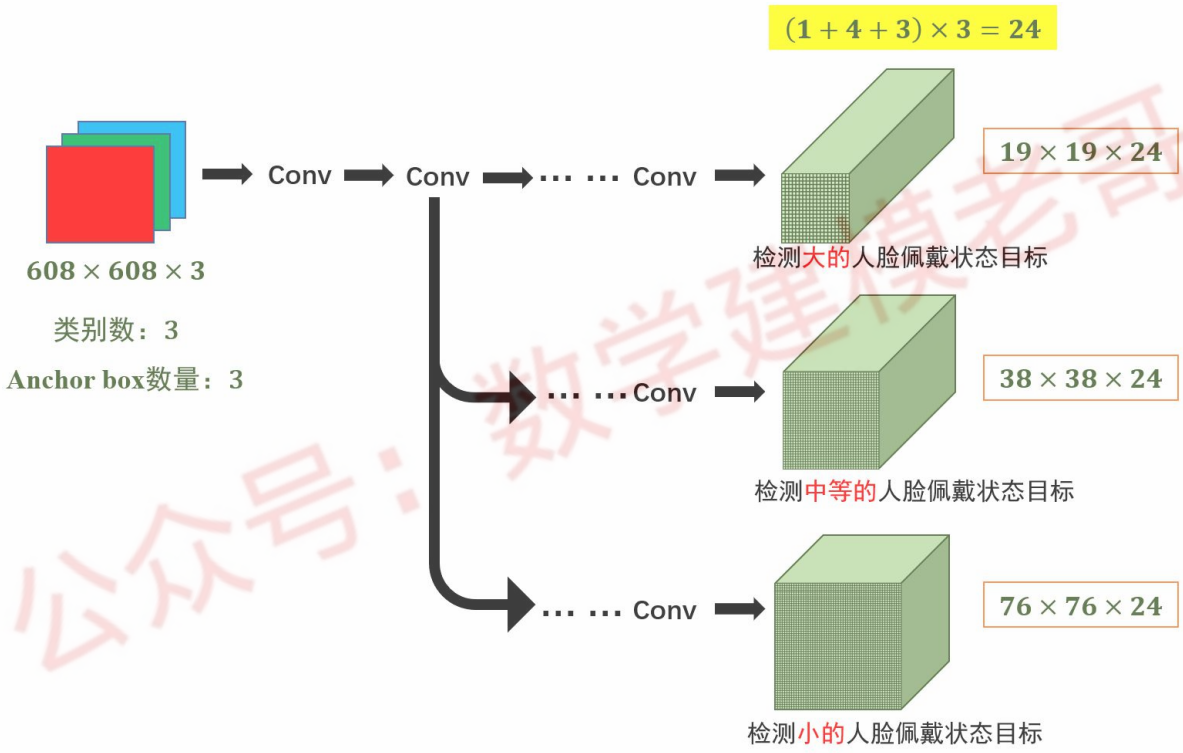


图 10　提取不同尺度的特征图

如图 10 所示，由于 608/19=32，所以右上角第一个长方体中的每个单元网格 1*1*24 的特征向量对应于原图上 32*32*3 的区域的信息，下面两个长方体同理，因此可以得到三种不同的特征图。19*19 的特征图相对于其他两个特征图来说有最大的感受野，因此使用的锚框尺寸也较大，适合检测较大的目标；38*38 的特征图具有中等的感受野，适合检测中等的目标；76*76 的特征图具有较小的感受野，适合检测较小的目标。

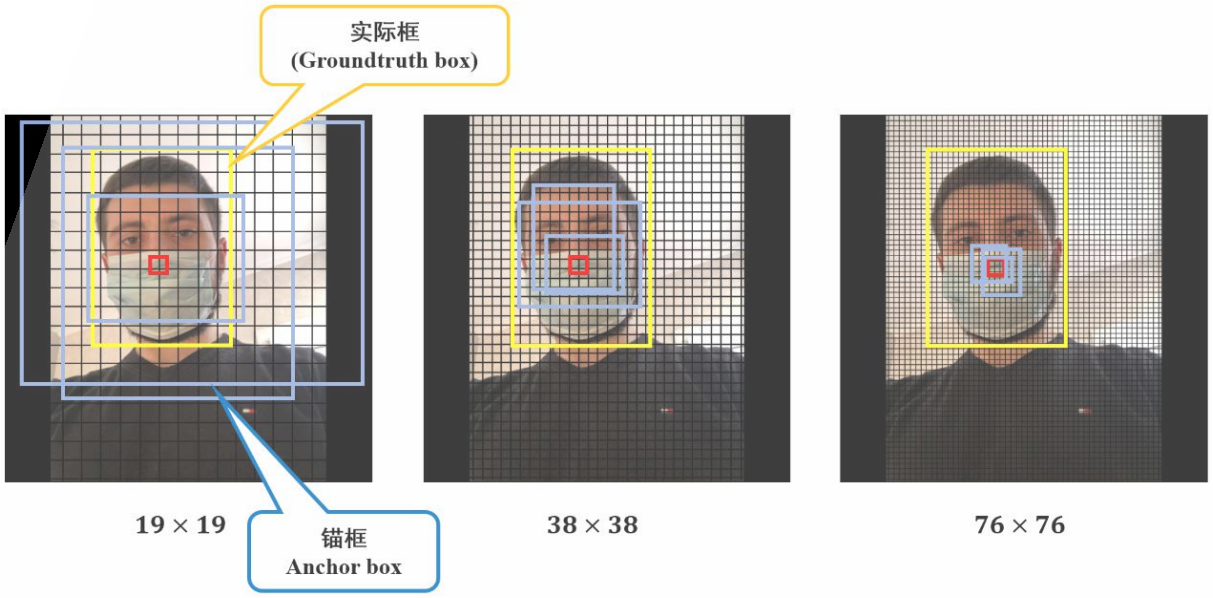


图 11　三种不同尺寸的特征图网格下锚框的效果图

由图 11 可知，每张图上红色的框是目标中心点所在，黄色的框是人脸实际的框，同时每张图都有三个不同尺寸的蓝色锚框。可以很明显地看到，该张图片中的人脸属于尺寸较大的目标，因此适合用 19*19 的特征图来检测人脸。

（3）Anchor box 计算机制

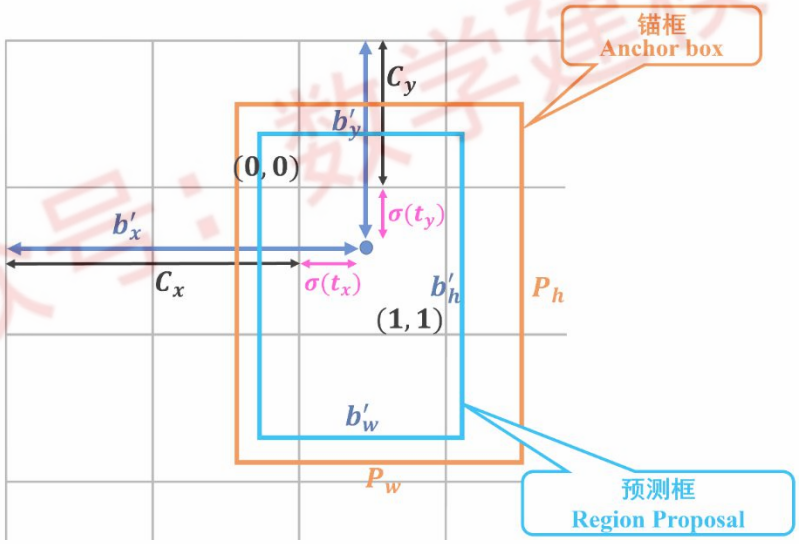


图 12　Anchor box 计算机制简单示意图

其中，$b_x^{'} = \sigma(t_x) + C_x, b_y^{'} = \sigma(t_y) + C_y, b_h^{'} = P_h e^{t_h}, b_w^{'} = P_w e^{t_w}$

由图 12 可知，橙色的框是提前设置好的锚框，蓝色的框是预测框，预测框在锚框的基础上和实际框通过 CIoU 损失函数进行关联，并且在锚框大小的基础上预测出目标框的大小，不断进行修正。

我们通常会将$b_x^{'}$和$b_y^{'}$延长，由到单元格两边的距离变成到图像边界的距离。而且通常会对t_x和t_y使用 sigmoid 函数进行处理，将其约束在 0-1 区间内。

（4）Anchor box 对应机制

首先计算各个锚框和实际框之间的 IoU 值，选择 IoU 值最大的锚框产生的预测框，计算其与实际框之间的差距，并通过梯度下降算法来不断修正。

（5）算法的损失函数

目标检测的损失函数一般来讲包括三个部分：边界框定位损失L_{loc}、预测类别损失L_{cls}和置信度损失L_{conf}，整体的网络损失 L 的计算公式[7]如下：

$$L = L_{loc} + L_{cls} + L_{conf} \tag{1}$$

边界框定位损失L_{loc}用于衡量预测框与真实框的位置（包括宽、高和中心坐标等）误差，交并比（intersection over union，IoU）是目前最常用的评估指标，其计算公式如下：

$$IoU = \frac{\left|B \cap B^{gt}\right|}{\left|B \cap B^{gt}\right|} \tag{2}$$

其中$B = (x, y, w, h)$表示预测框的位置，$B^{gt} = (x^{gt}, y^{gt}, w^{gt}, h^{gt})$表示真实框的位置。而 IoU 值仅在两边界框之间有重叠部分时有效，对于非重叠的情况，IoU 值则会失效。

本论文中采用 CIoU 损失函数，计算公式[8]如下：

$$CIoU = IoU - \frac{\rho^2\left(B, B^{gt}\right)}{d^2} - \alpha\upsilon \tag{3}$$

其中，$\rho^2(B, B^{gt})$表示预测框中心点和实际框中心点之间的欧氏距离，d 表示包含预测框和实际框的最小外接矩阵的对角线距离，α是权重参数，υ是衡量长宽比一致性的参数，α和υ的计算公式如下：

$$\alpha = \frac{\upsilon}{1 - IoU + \upsilon} \tag{4}$$

$$\upsilon = \frac{4}{\pi^2}\left(\arctan\frac{\omega^{gt}}{h^{gt}} - \arctan\frac{\omega}{h}\right)^2 \tag{5}$$

则边界框定位损失函数L_{loc}为：

$$L_{loc} = 1 - IoU + \frac{\rho^2\left(B, B^{gt}\right)}{d^2} + \alpha\upsilon \tag{6}$$

预测类别损失L_{cls}用于衡量预测框和真实框的类别误差，采用交叉熵函数进行衡量：

$$L_{cls} = -\sum_{i=0}^{K\times K} I_{ij}^{obj} \sum_{c\in classes}\left[\hat{p}_i(c)\log\left(p_i(c)\right) + \left(1 - \hat{p}_i(c)\right)\log\left(1 - p_i(c)\right)\right] \tag{7}$$

其中$K \times K$表示不同尺寸特征图上的网格数量；如果第 i 个网格的第 j 个锚框有需要预测的物体，那么$I_{ij}^{obj} = 1$，否则为 0；c 表示某一目标类别，$\hat{p}_i(c)$表示第 i 个网格的第 j 个锚框属于类别 c 的概率真实值，$p_i(c)$则表示概率预测值。

置信度损失L_{conf}采用交叉熵损失函数进行衡量，计算公式如下：

$$L_{conf} = \sum_{i=0}^{K\times K}\sum_{j=0}^{M} I_{ij}^{obj}\left[\hat{C}_i\log\left(C_i\right) + \left(1 - \hat{C}_i\right)\log\left(1 - C_i\right)\right] - \sum_{i=0}^{K\times K}\sum_{j=0}^{M} I_{ij}^{noobj}\left[\hat{C}_i\log\left(C_i\right) + \left(1 - \hat{C}_i\right)\log\left(1 - C_i\right)\right] \tag{8}$$

其中$K \times K$和I_{ij}^{obj}的含义与式（7）一致，M 表示先验框的个数，表示置信度的真实值和预测值；如果第 i 个网格的第 j 个锚框中没有需要预测的物体，那么$I_{ij}^{obj} = 1$，否则为 0。

4.2.3.3 算法网络结构解析

1.Backbone 主干网络

Backbone 主干网络部分采取了 CSPDarknet53 的结构，是在 Darknet53 网络的基础上借鉴了 CSPNet（Cross Stage Paritial Network）的优点产生的主干结构，其中包含了 5 个 CSP 模块[9]。

Type	Filters	Size	Output	
Conv	32	3×3	608×608	
Conv	64	3×3/2	304×304	
Conv	32	1×1		1*
Conv	64	3×3		
Res			304×304	
Conv	128	3×3/2	152×152	
Conv	64	1×1		2*
Conv	128	3×3		
Res			152×152	
Conv	256	3×3/2	76×76	
Conv	128	1×1		8*
Conv	256	3×3		
Res			76×76	
Conv	512	3×3/2	38×38	
Conv	256	1×1		8*
Conv	512	3×3		
Res			38×38	
Conv	1024	3×3/2	19×19	
Conv	512	1×1		4*
Conv	1024	3×3		
Res			19×19	
Avgpool		Global		
Connected		1000		
Softmax				

图 13　Darknet53 结构图

由于每个 CSP 模块前面的卷积核的大小都是 3×3，步长为 2，而输入图像是 608*608，因此经过 5 次 CSP 模块作用后可以得到大小为 19*19（608->304->152->76->38->19）的特征图。

由于 Mish 函数是光滑的非单调函数，故在 Backbone 中采用 Mish 激活函数来代替 YOLOv3 中的 Leaky_relu 激活函数，其公式如下：

$$f(x) = x * \tanh(\varsigma(x)), \varsigma(x) = \ln(1 + e^{x}) \quad (9)$$

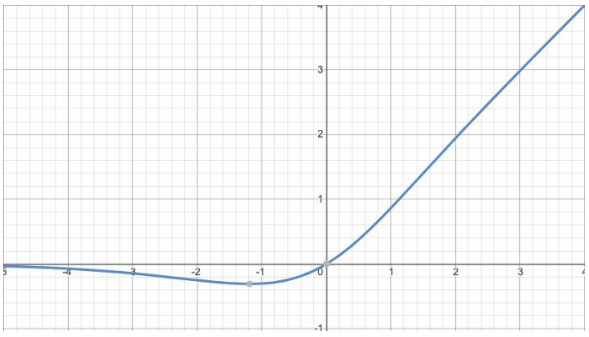


图 14　Mish 函数图

如图 14 所示，Mish 函数具有非单调的性质，有助于保持小的负值，从而稳定网络梯度流，其光滑性和无穷阶连续性可以提高模型的泛化能力。

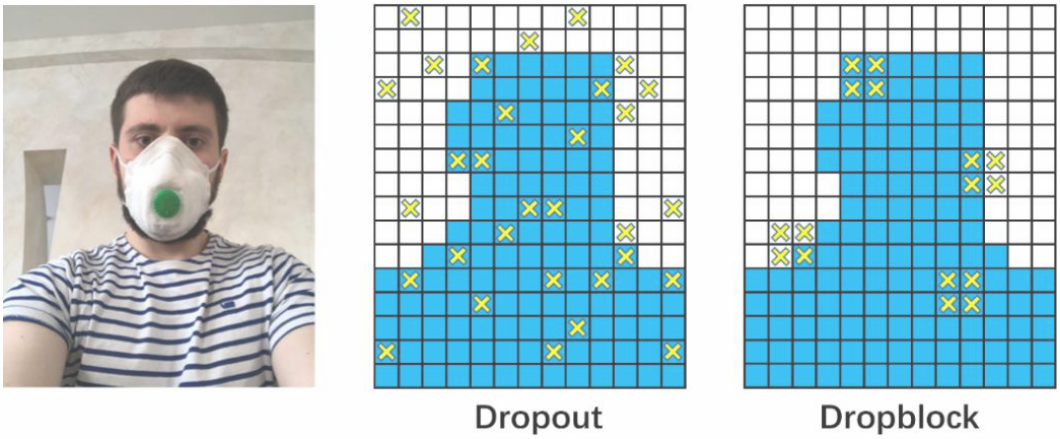


图 15　Dropout 和 Dropblock 示意图

如图 15 所示，对于常见网络中的 Dropout 功能提出进一步改进，不采用随即丢弃信息的方式，而是将局部区域进行删减丢弃，采用更优的 Dropblock 方式，对网络的正则化过程进行了全面的优化。

2.Neck 层

Neck 层作用在 Backbone 主干网络和 Prediction 输出层之间，主要采用 SPP 模块和 FPN+PAN 结构组合而成。在 SPP 模块中，使用 maxpooling 最大池化方式，再将不同尺寸的特征图进行 Concat 连接操作。

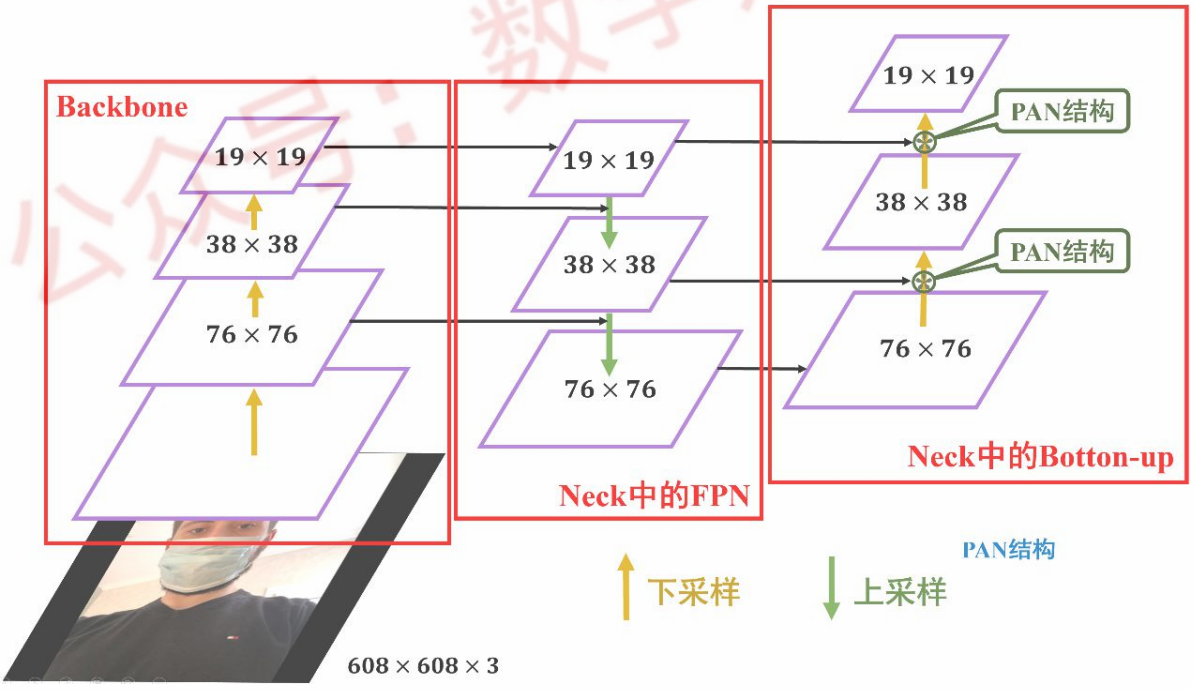


图 16　YOLOv4 算法的网络结构立体图

如图 16 所示，FPN 结构自顶向下进行上采样操作，并和 Backbone 主干网络中提取出来的特征图进行 Concat 连接，在此基础上添加了一个自底向上的特征金字塔，其中包含两个 PAN 结构。二者结合起来，可以实现从不同的主干层进行特征聚合，进而传播到输出层的三个特征图中。

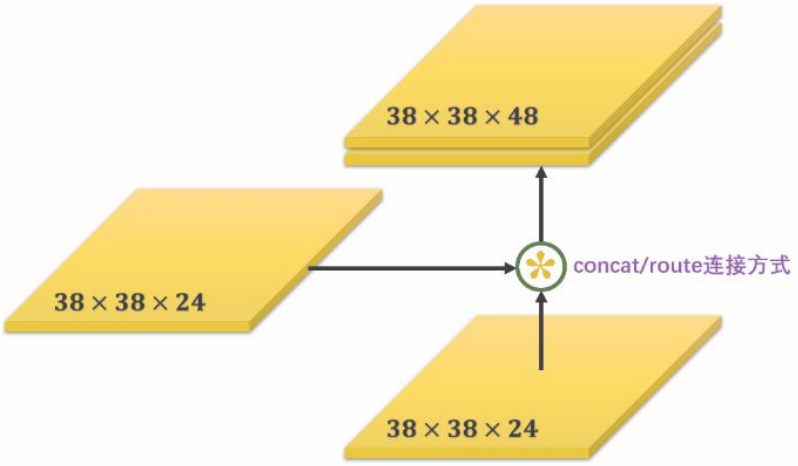


图 17　YOLOv4 中的 PAN 结构图

如图 17 所示，这里的 PAN 结构是借鉴了 PANet 网络中原 PAN 结构，并将 shortcut 连接方式改成了 concat/route 连接方式。

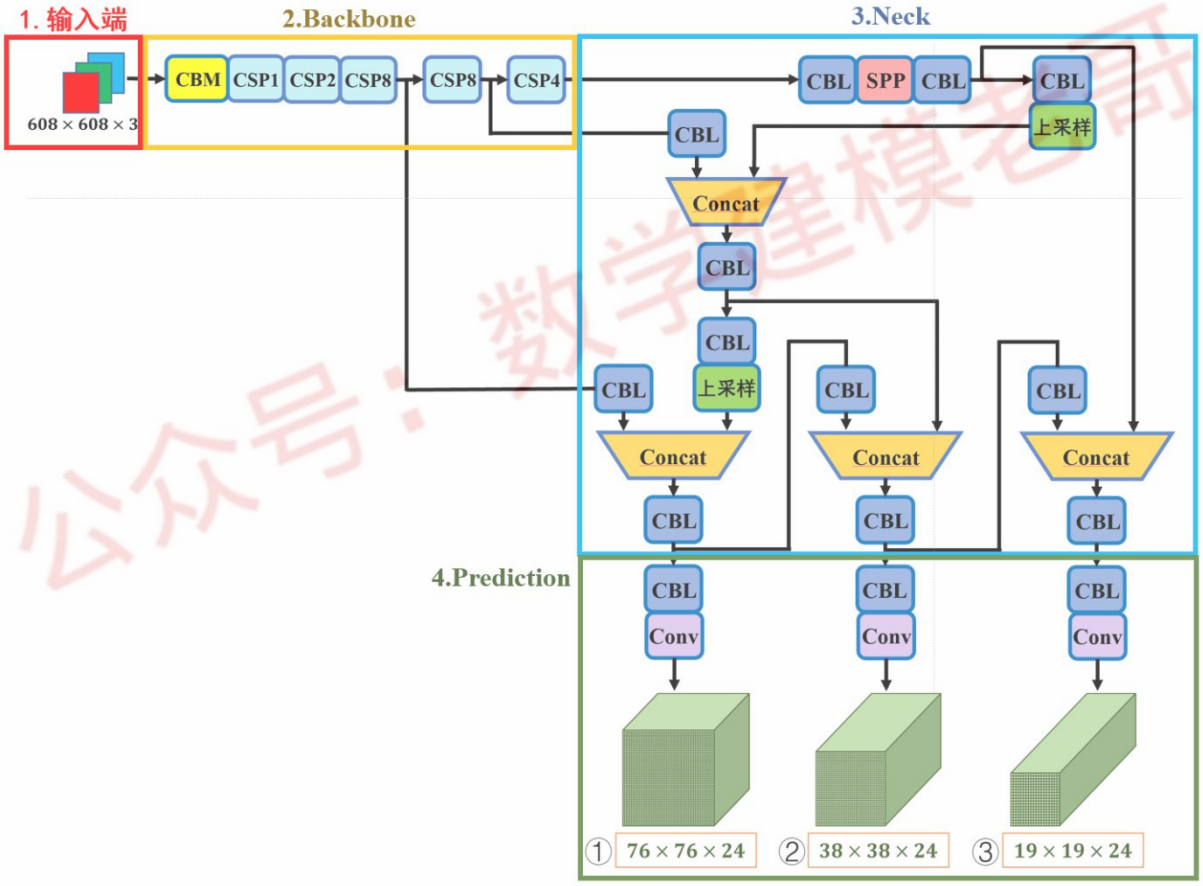


图 18　YOLOv4 算法的网络结构平面图

4.2.3.4 YOLOv4 算法测试过程

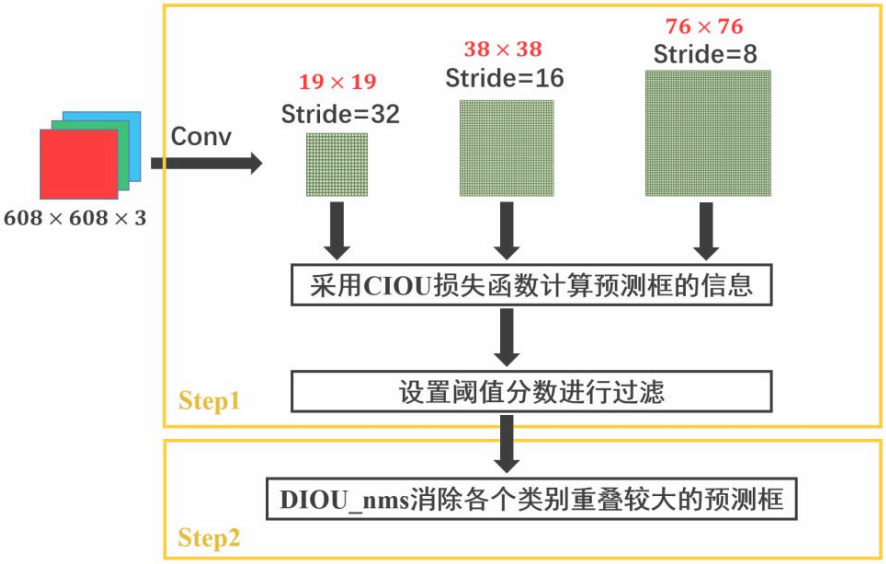


图 19　YOLOv4 算法的测试过程

如图 19 所示，第一部分采用 CIoU 损失函数计算预测框的信息，并设置阈值，过滤掉得分低的框；第二部分使用 DIoU_nms 消除各个类别重叠较大的预测框。

在实际应用场景中，如果使用经典的非极大值抑制（Non-Maximum Suppression，NMS）方式进行处理，当两个不同物体挨得很近时，由于 IoU 值比较大，往往只剩下一个预测框，从而会导致漏检的情况发生。DIoU_nms 在 IoU 的基础上考虑了两个框中心点之间的距离。如果两个框之间 IoU 比较大，而两个框的距离也比较大时，可能会将其判断为两个物体的框，从而大大减少漏检的情况。

4.2.4 基于 YOLOv4 算法的改进创新点

1.模型融合：

由于样本比例不均衡，“未正确佩戴口罩”的样本在总样本中所占比例极少，各样本分布情况如图 20 所示。

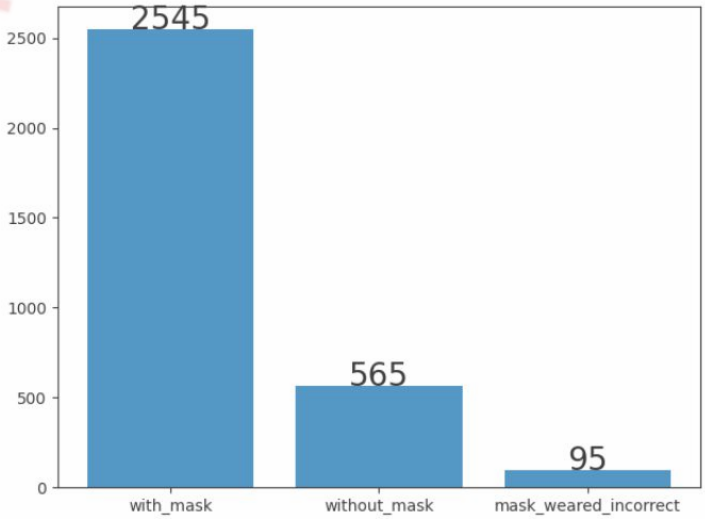


图 20　总样本中三个类别的样本分布图

在单独使用 YOLOv4 算法进行检测的情况下，我们观察发现模型容易将“未正确佩戴口罩”这一类别的样本错误识别为“佩戴口罩”这一类，因此除了基于 YOLOv4 的数据增强方式之外，本论文还加入了模型融合的方式进行共同决策。

图 21　使用 YOLOv4 算法检测结果的示例图

如图 21 所示，根据 YOLOv4 算法检测到的结果，预测框基本能正确预测出人脸的位置，虽然在少数情况下不能正确区分“未正确佩戴口罩”和“佩戴口罩”的样本，分析原因是在没遮住鼻子的情况下较难区分，分类的置信度会相对较低。

因此我们考虑将人脸的口罩部分提取出来，根据口罩的像素块占整个人脸区域像素块的比例，以及口罩像素块的最小外接矩形长宽比来判断是否正确佩戴口罩。

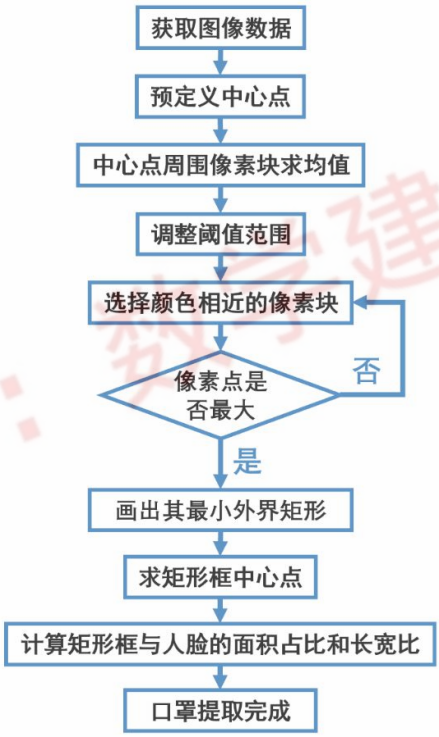


图 22　口罩提取流程图

图 22 展示了口罩的提取流程：首先根据得到的数据预定义一个口罩的中心点，再截取中心点周围一定大小的像素块，分三个通道分别求出均值，将该均值作为口罩颜色设定。之后调整范围的过滤值，将整幅图像中颜色与之相近的像素块全部提取出来，求出其中最大像素块的最小外接矩形，并画出矩形，再求出矩形框的中点为最终得到的口罩中心点位置，计算出矩形的面积，求出其所占人脸框的比例，最后算出长宽比。

I. 预定义口罩中心点位置

经过查找相关资料可以得知，口罩位置一般位于人脸中央偏下方的位置，我们预定义口罩中心点的公式如下：

$$\begin{cases} x_1 = 0.5 \times (x_{max} - x_{min}) \\ y_1 = 0.575 \times (y_{max} - y_{min}) \end{cases} \tag{11}$$

其中x_{max}为预测框（即截取的人脸框）右下角的 x 坐标值；x_{min}为预测框左上角的 x 坐标值；y_{max}为预测框右下角的 y 坐标值；y_{min}为预测框左上角的 y 坐标值；(x_1, y_1)为预定义的口罩中心点位置坐标。

图 23　截取人脸框并预定义口罩中心点位置

II. 颜色过滤

在口罩中心周围，分别截取人脸框大小五分之一的像素块，截取的范围如下所示：

$$\begin{cases} x_{left} = x_1 - 0.1 \times (x_{max} - x_{min}) \\ x_{right} = x_1 + 0.1 \times (x_{max} - x_{min}) \\ y_{left} = y_1 - 0.1 \times (y_{max} - y_{min}) \\ y_{right} = y_1 + 0.1 \times (y_{max} - y_{min}) \end{cases} \tag{12}$$

其中x_{left}为截取框的横坐标左边界，x_{right}为截取框的横坐标右边界，y_{left}为截取框的纵坐标左边界和y_{right}为截取框的纵坐标右边界。然后将截取到的图像转化为 HSV 格式，由于 HSV 格式只有一个通道代表颜色，故在颜色筛选时比 RGB 更高效。之后将三通道的图像分通道计算均值，通过查阅资料，了解到筛选范围一般取该均值周围三十五个 HSV 值。由于筛选后存在干扰像素块，故采用先腐蚀后膨胀的闭操作进行过滤，最终筛选结果如图 24 所示：

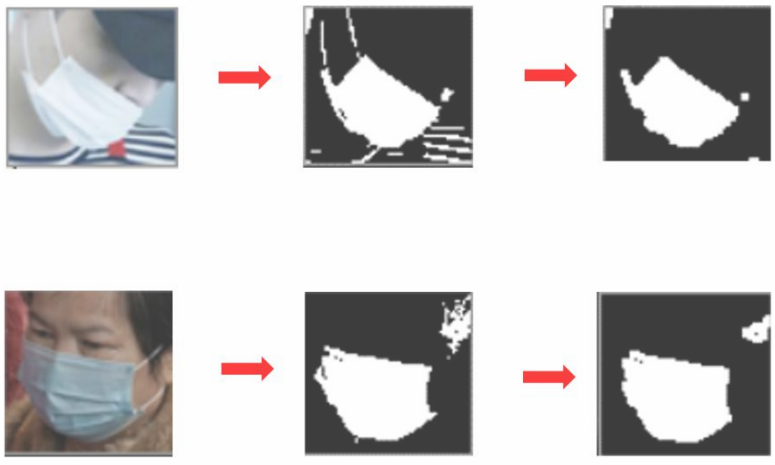


图 24　颜色筛选结果

III．求解最小外接矩形框

在颜色筛选后将图像二值化，以口罩为前景区域对图像求取轮廓，这里调用了opencv 中的 findcontours()函数得到口罩的外轮廓，在此基础上求解轮廓点集的凸包，然后使用旋转卡壳算法求出函数的最小外接矩形。求解示意图如图 25 所示：

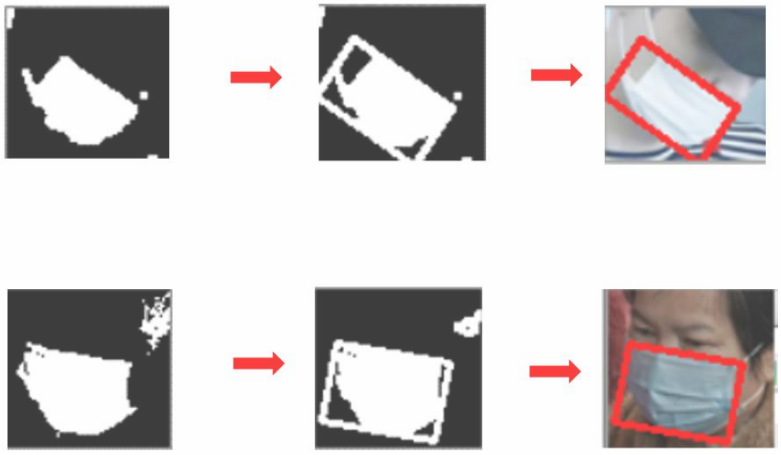


图 25　求解最小外接矩阵框示意图

IV．决策判断

主要利用三个特征进行决策判断：矩形的长宽比，像素占人脸的比例，以及使用基础 YOLOv4 算法计算出来的戴口罩这一类别的置信度。由于样本过度不均衡，且人工标注较为困难，无法采用机器学习的分类器进行训练，因此我们根据训练效果不断调整参数，最后设定长宽比和口罩占人脸的阈值分别为 0.6、0.4，并基于此参数进行最后的分类决策。

2.采用 k-means 聚类方法选取 Anchor box

在选取 Anchor box 时采用 k-means 聚类方法，对人工标注的不同目标的形状进行聚类，计算其中最具有代表性的一组 Anchor box，从而减少预测框拟合实际框的时间，大大提高拟合的精度和速度[10]。

使用 k-means 聚类方法选取 Anchor box 的步骤：

1）无差别地提取所有图片的 Bounding box 坐标

2）处理数据，获得所有 Bounding box 坐标的宽高大小

3）随机初始化 k 个 Anchor box

4）使用欧氏距离直接聚类 Bounding box 的宽和高，产生 k 个宽、高组合的 Anchor box

5）进行分类操作

6）Anchor box 更新

7）重复第 4-6 步直到所有 Bounding box 的分类不再更新，分类完成

4.2.5 算法评价指标

1. IoU 表示检测框与标签框的交并比，IoU 阈值为 0.45 及以上则判断为预测正确。

2. TP、TN、FP、FN 值

表 3　TP、TN、FP、FN 说明

	正样本	负样本
预测为正样本	TP（True Positives）	FP（False Positives）
预测为负样本	FN（False Negatives）	TN（True Negatives）

3. 查准率 Precision：衡量的是一个分类器分出来的正样本的确是正样本的概率。

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

查全率 Recall：衡量的是一个分类器能把所有的正样本都找出来的能力。

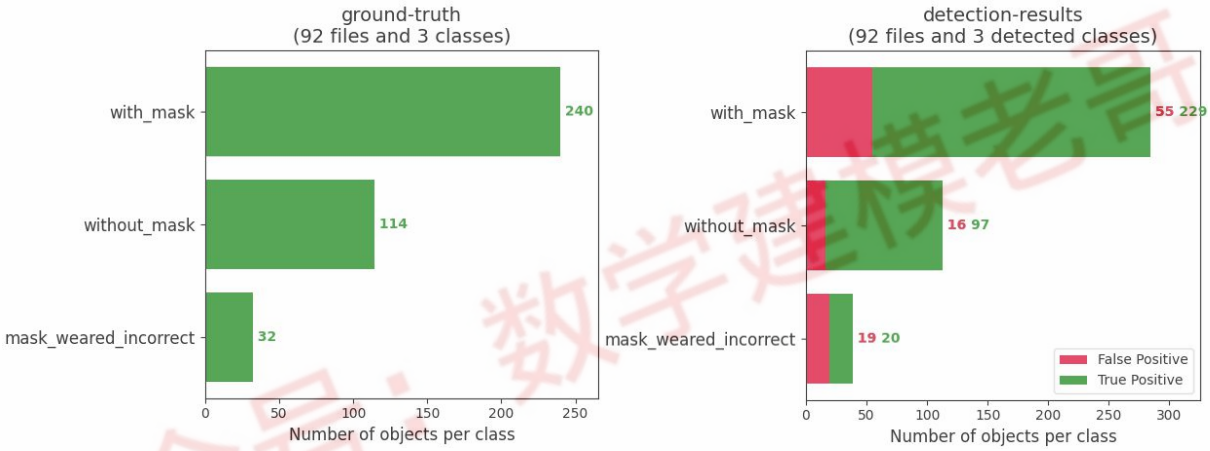
$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

4. 平均准确率 Average Precision：AP 衡量的是对一个类检测的好坏程度，mAP 衡量的是对多个类的检测好坏程度，是对各个类 AP 值求平均得来的。本论文使用 pascal voc 2012 的评价标准 mAP 来评价口罩检测算法的“好坏”程度。

4.2.6 训练效果说明

1.原始 YOLOv4 算法结果及分析

将训练样本和扩充样本共 920 张图片按照 9:1 的比例划分成训练集和测试集，测试集中包含 92 张图片，测试集中 3 种口罩的佩戴状态真实数量如图 26(a)所示，通过原始 YOLOv4 检测算法预测的结果如图 26(b)所示：



(a)口罩 3 种佩戴状态的真实数量　　(b)口罩 3 种佩戴状态的预测数量

图 26　原始 YOLOv4 预测前后数量对比图

如上图所示，在划分出的测试集中，图(a)表示标签标注的口罩 3 种佩戴状态的真实数量，共计 386 个；图(b)表示口罩 3 种佩戴状态的预测数量，共计 436 个。图(b)总数量比图(a)总数量多的原因是检测算法将未作标注的人脸也进行了识别，未标注的人脸多为小目标和模糊目标，可见该算法对这两种目标也具有较好的识别度。

根据图(b)，YOLOv4 算法在口罩 3 种佩戴状态上的查准率如下表所示：

表 4　YOLOv4 算法在口罩 3 种佩戴状态上的查准率

佩戴状态	查准率
佩戴口罩	80.63%
未佩戴口罩	85.84%
未正确佩戴口罩	51.28%

由上表可见，“未佩戴口罩”状态的查准率最高，为 85.84%；“未正确佩戴口罩”状态的查准率最低，为 51.28%。说明相比于其他两种佩戴情况，原始 YOLOv4 算法在“未正确佩戴口罩”状态的识别上表现较差。

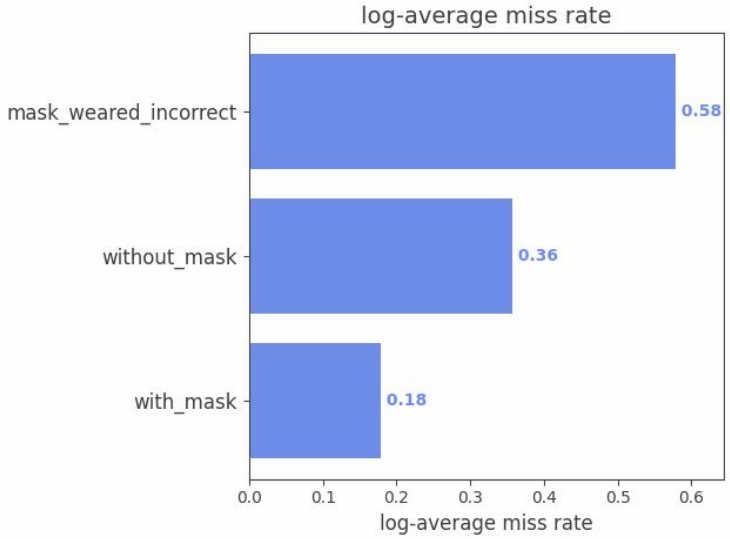
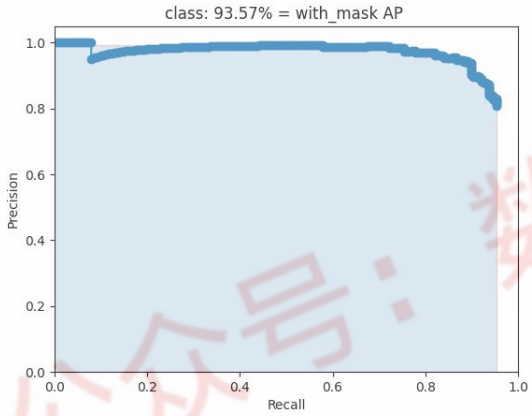


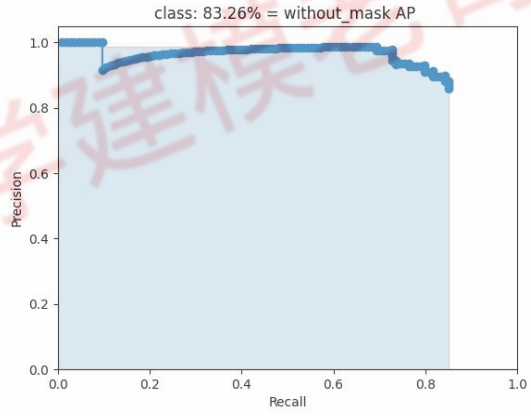
图 27　口罩 3 种佩戴状态的对数平均漏检率

如图 27 所示，“未正确佩戴口罩”的对数平均漏检率最高，为 0.58；“佩戴口罩”状态的对数平均漏检率最低，为 0.18，说明原始 YOLOv4 算法在“未正确佩戴口罩”状态的识别上表现较差。

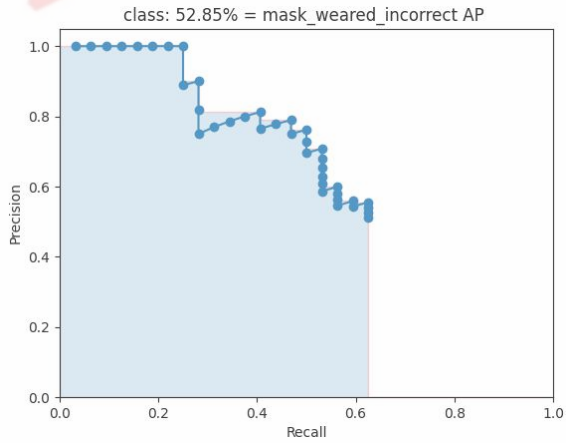
此外，本论文使用 mAP 指标来评价口罩检测算法的“好坏”程度，如图 28 所示：



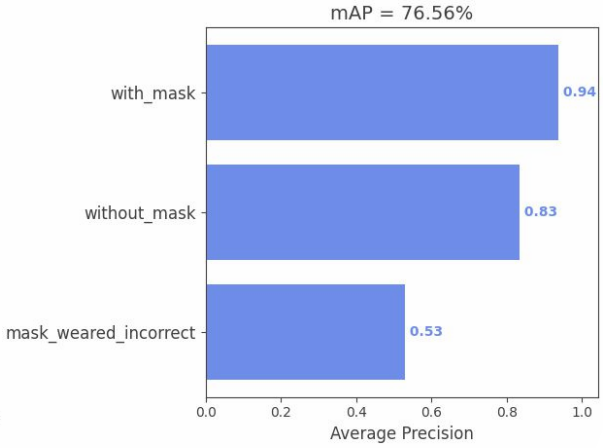
图(a)　“佩戴口罩”状态的 AP 值



图(b)　“未佩戴口罩”状态的 AP 值



图(c)　“未正确佩戴口罩”状态的 AP 值



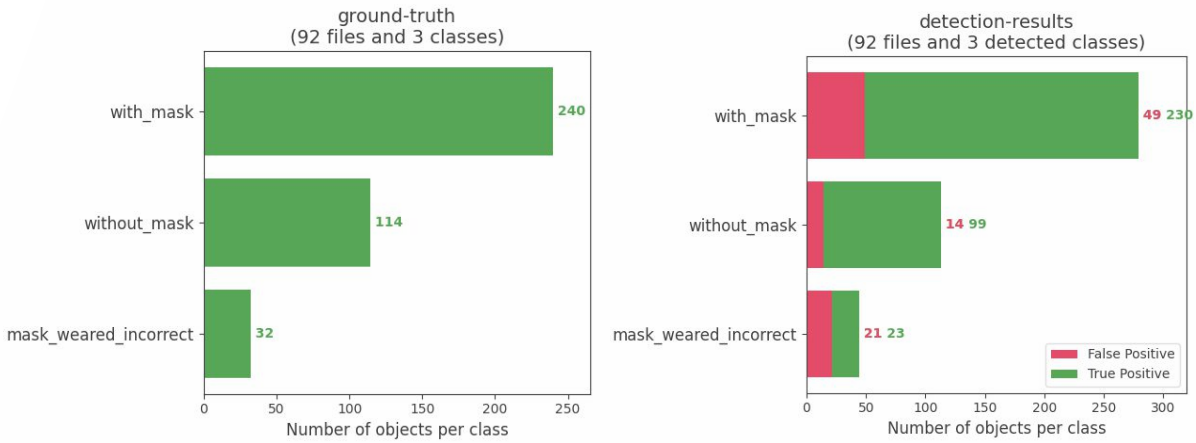
图(d)　口罩 3 种佩戴状态的 mAP 值

图 28　口罩 3 种佩戴状态的 AP 及 mAP 指标

由图 28 可见，原始 YOLOv4 算法在对“佩戴口罩”和“未佩戴口罩”两类的检测程度较好，在“未正确佩戴口罩”这一类上的检测程度较差。

2.改进 YOLOv4 算法结果及分析

测试集中 3 种口罩的佩戴状态真实数量如图 29(a)所示，通过原始 YOLOv4 检测算法预测的结果如图 29(b)所示：



(a)口罩 3 种佩戴状态的真实数量　　(b)口罩 3 种佩戴状态的预测数量

图 29　改进 YOLOv4 预测前后数量对比图

改进 YOLOv4 算法在口罩 3 种佩戴状态上的查准率如下表所示：

表 5　改进 YOLOv4 算法在口罩 3 种佩戴状态上的查准率

佩戴状态	原始 YOLOv4 查准率	改进 YOLOv4 查准率
佩戴口罩	80.63%	**82.44%**
未佩戴口罩	85.84%	**87.61%**
未正确佩戴口罩	51.28%	**52.27%**

由表 5 可见，改进 YOLOv4 算法在预测 3 种佩戴状态的查准率均有所提升。其中，“佩戴口罩”和“未佩戴口罩”状态的查准率提升较高，分别提升了 1.81%和 1.77%；在“未正确佩戴口罩”状态的查准率上提升了 0.99%，基本实现了设计口罩检测算法的要求。

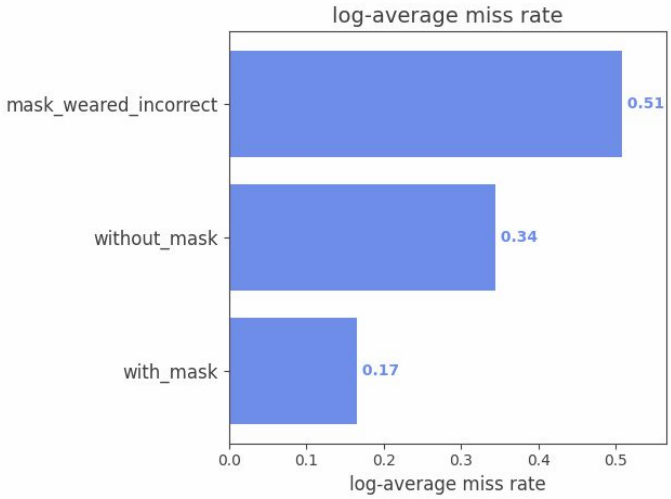


图 30　口罩 3 种佩戴状态的对数平均漏检率

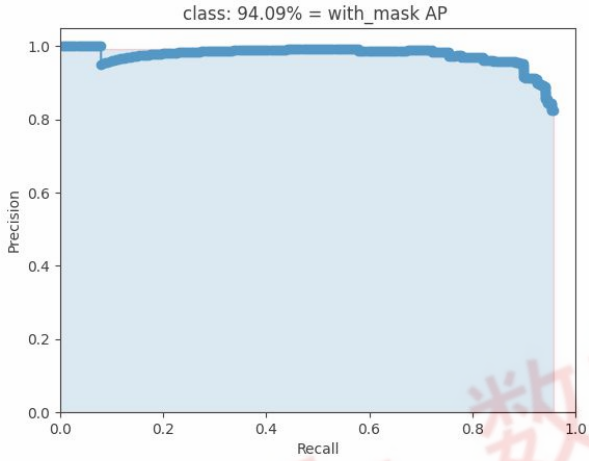
在对数平均漏检率指标上，改进 YOLOv4 算法的结果如图 30 所示，将该结果与原始算法的结果相比较，如表 6 所示：

表 6　改进 YOLOv4 算法与原算法结果的漏检率对比

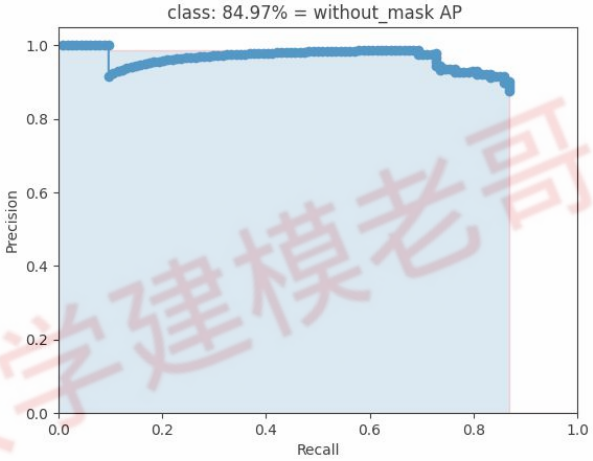
佩戴状态	原始 YOLOv4 漏检率	改进 YOLOv4 漏检率
佩戴口罩	0.58	**0.51**
未佩戴口罩	0.36	**0.34**
未正确佩戴口罩	0.18	**0.17**

由表 6 可见，改进 YOLOv4 算法在预测 3 种佩戴状态的漏检率上均有所下降。其中，“佩戴口罩”状态的漏检率下降最明显，下降了 7%；同时，“未佩戴口罩”和“未正确佩戴口罩”状态的漏检率分别下降了 2%和 1%，说明改进后的算法比原始算法表现更优。

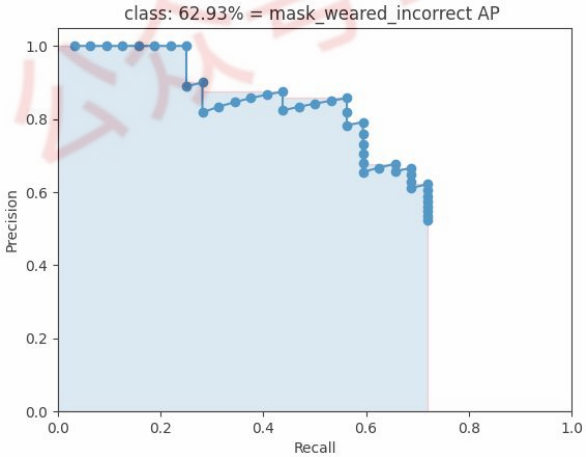
此外，在 mAP 指标方面，改进后算法的结果如图 31 所示：



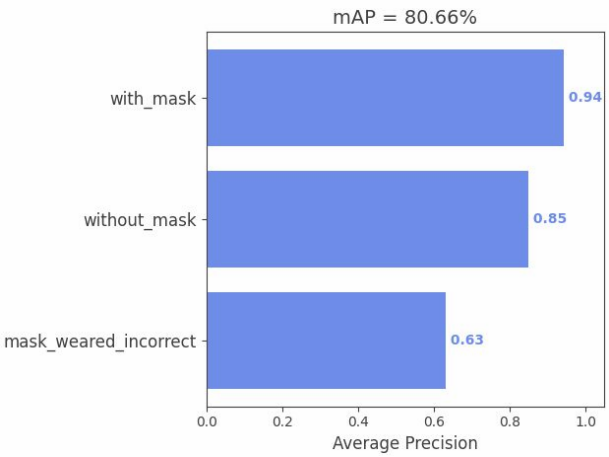
图(a)　“佩戴口罩”状态的 AP 值



图(b)　“未佩戴口罩”状态的 AP 值



图(c)　“未正确佩戴口罩”状态的 AP 值



图(d)　口罩 3 种佩戴状态的 mAP 值

图 31　口罩 3 种佩戴状态的 AP 及 mAP 指标

由上图可见，改进 YOLOv4 算法在口罩 3 种佩戴状态上的检测程度均有所提升，具体对比图如表 7 所示：

表 7　改进 YOLOv4 算法与原算法结果的 AP 值对比

佩戴状态	原始 YOLOv4 的 AP 值	改进 YOLOv4 的 AP 值
佩戴口罩	0.94	0.94
未佩戴口罩	0.83	**0.85**
未正确佩戴口罩	0.53	**0.63**
三类状态均值	76.56%	**80.66%**

由表 7 可见，在原始 YOLOv4 算法上表现最差的“未正确佩戴口罩”状态检测，改进 YOLOv4 算法大幅提升了该类的检测性能，使“未正确佩戴口罩”状态的 AP 值提高了 10%，使三类状态的 mAP 值提高了 4.1%，说明了改进算法的优越性。

4.3 问题 3 模型求解与分析

使用改进 YOLOv4 算法对“附件 2 测试样本”中 200 张图片的人脸口罩佩戴状态进行定位和预测分类，同时用问题 1 中算法对分类结果进行可视化操作，将输出结果与原 YOLOv4 算法输出结果对比，选取 3 张代表图片展示如图 32 所示：

图 32　测试样本预测结果对比图

由图 32 可看出，改进后的算法基本能区分出“佩戴口罩”、“未佩戴口罩”和“未正确佩戴口罩”三种状态，尤其是提高了对“未正确佩戴口罩”状态的区分能力，包括对帽檐遮挡人脸的情况也取得了较好的效果。

但也有少数样本预测错误的情况，如图 33 所示：

(a) (b)

(c)

图 33　预测误识别的结果示例图(黄色框内为误识别样本)

分析上图误识别的原因，用表 8 展示如下：

表 8　预测误识别情况与原因分析

	误识别情况	原因分析
图(a)	将孩子“佩戴口罩”错误识别为“未正确佩戴口罩”状态	算法误将鼻梁两侧的阴影识别成鼻孔从而导致判断错误
图(b)	将背景墙中佛祖也识别了出来	说明算法不能很好的区分出真人
图(c)	将女孩“未正确佩戴口罩”错误识别为“未佩戴口罩”状态	粉色口罩拉至下巴，口罩变形折叠后导致无法识别

五、模型优缺点分析和展望

5.1 模型优点分析

1）输入端使用数据增强 Mosaic 方式，丰富了数据集，使得网络具有更强的鲁棒性，并且减少了 GPU 使用量，增强了模型的普适性。

2）BackBone 主干网络采用 CSPDarknet53 结构，增强了模型的学习能力，降低了计算瓶颈和内存成本。

3）Neck 层采用 SPP 模块和 FPN+PAN 结构，有效增加了主干特征的接受范围，并可以实现不同检测层的特征聚合。

4）训练时采用 CIoU 损失函数，使得模型回归的速度和精度更高；预测框筛选采用 DIoU_nms 方式，减少了在重叠目标检测中漏检的情况。

5.2 模型缺点分析

1）Backbone 结构采用 Mish 激活函数，计算量较大，代价成本相对于 relu 函数来说比较高

2）直接对标签的宽高进行聚类得到的 anchor box，回归效果有较大的误差。

3）模型大小为 244MB，使得实际部署困难，应用普适性差。

4）在小目标样本的识别上仍有难度，难以区分更精细化的一些特征。

5.3 模型展望

实验结果表明，改进 YOLOv4 算法能较好地识别出样本中的人脸，并较为准确地预测出口罩佩戴状态，基本达到预期目标。但仍存在可改进的空间，如针对口罩挤压变形导致无法识别的问题，可以先将口罩佩戴状态按照“戴”和“没戴”进行二分类，在“戴”的样本中增加对鼻子的判断机制，就可以进一步区分出“正确佩戴”和“未正确佩戴”；针对算法不能很好的区分出真人的问题，可以融合真人人脸的检测算法以辅助决策。此外，还可以将算法扩展到视频领域，通过对空间管道的关联性分析可以实现进行口罩实时检测。

参考文献

[1]SIGAI，人脸检测算法综述， 2021.10.30.

[2]S.Z.Li, L.Zhu, Z.Q.Zhang, A.Blake, H.J.Zhang, H.Y.Shum. Statistical learning of multi-view face detection. In: Proceedings of the 7-th European Conference on Computer Vision. Copenhagen, Denmark: Springer, 2002.67-81.

[3]SIGAI，基于深度学习的目标检测算法综述，2021.10.30.

[4]丁培，阿里甫·库尔班，耿丽婷，韩文轩，自然环境下实时人脸口罩检测与规范佩戴识别，计算机工程与应用，1-10，2021.09.08.

[5]Song Zhuoxue(Bil369)，基于 PyTorch&YOLOv4 的口罩佩戴检测，2021.10.30.

[6]Wang Z , Wang G , Huang B , et al. Masked Face Recognition Dataset and Application. 2020.

[7]王兵,乐红霞,李文璟,张孟涵.改进 YOLO 轻量化网络的口罩检测算法[J].计算机工程与应用,2021,57(8):62-69.

[8] ZHENG Z，WANG P，LIU W，et al. Distance-IoU loss：Faster and better learning for bounding box regression［C］//Proceedings of the AAAI Conference on Artificial Intelligence. 2020， 34（07）：12993-13000.

[9]江大白，深入浅出 Yolo 系列之 YOLOv3&YOLOv4&YOLOv5&Yolox 核心基础知识完整讲解，2021.10.30.

[10]码农家园，YOLOV4 中 k-means 聚类获得 anchor boxes，2021.10.30.

附录

代码名称	代码解决的问题	代码电子版所在的位置
dataset.py	问题 1 的标签可视化	支撑材料中 MaskDetect-YOLOv4
eval.py	问题 2 模型评价，主要用于利用模型生成检测后的标签信息（已运行并生成了图片）	支撑材料中 MaskDetect-YOLOv4
main.py	生成评价指标，生成的图在 MaskDetect-YOLOv4\mAP\Output\文件夹下（可直接查看）	支撑材料中 MaskDetect-YOLOv4\mAP\main.py
predict.py	对 test 样本集进行预测，将结果放入表格并可视化	支撑材料中 MaskDetect-YOLOv4\predict

因代码量较大，可直接从支撑材料中打开，这里主要简述代码的运行方式。

一、数据准备

首先直接运行 dataset 文件，实现第一问，将所有的训练图片可视化，需要将数据放入./data/train 文件夹下，并新建一个 visualization 的文件(已建好)如图 34 所示：

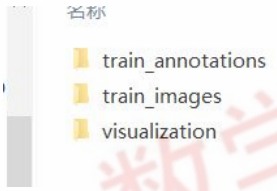


图 34　数据文件

此时，能够在 visualization 文件夹里看到所有的标签可视化结果。

接下来需要把图片生成训练集和验证集，在./train_split/new_train 文件夹下，将所有的数据以图 35 的数据文件进行存放，image 放入原始数据的图片，label 下放入对应标签。

图 35　train_split/new_train 数据存放方式

之后运行代码中的 data_partitioning.py 文件，图片会被自动按照 9：1 的方式划分到./data/train_split/train 和./data/train_split/val 文件夹下，如果后续要添加数据，也可以直接全部放入 new_train 下，之后再运行上述 py 文件即可（添加的数据文件不得和原来重复）。本论文中我们自己增加了一部分数据集，总的数据集可在百度网盘（提取码：pw05）直接提取，然后直接覆盖 new_train 即可。

之后运行 make_txt.py 文件，即可在./model_data 下文件夹下生成生成对于存放数据信息的 txt 文件，该 txt 文件在训练模型时会用到。

二、模型训练

以上数据处理全部完成，接下来是模型训练，从百度网盘下载模型（提取码：24qw），下载后得到一个 model 文件，里面放了两个模型 yolov4_maskdetect_weights1.pth 和 yolov4_maskdetect_weights4.pth, yolov4_maskdetect_weights1.pth 为预训练模型，如要重新训练模型，必须下载该模型，并放入./model_data 文件夹下，另一个模型是我们训练

好的，可供测试和验证使用。

之后运行 train_self 文件，即可开始训练。如输出如图 36 所示，开始训练。

```
----------Train one epoch.----------
Epoch:1/25
Start Training.
step:1/98 || Total Loss: 13416.5479 || 3.8960s/step
step:10/98 || Total Loss: 9011.9619 || 1.4416s/step
step:20/98 || Total Loss: 6523.8145 || 1.4155s/step
step:30/98 || Total Loss: 5048.5166 || 1.4631s/step
step:40/98 || Total Loss: 4120.8433 || 1.4772s/step
step:50/98 || Total Loss: 3490.8103 || 1.7355s/step
```

图 36　模型训练过程

当训练完成后会在./model_data 文件夹下生成 yolov4_maskdetect_weights7.pth 和 yolov4_maskdetect_weights8.pth，前者为第 25 个 epoch 下生成的模型，后者为第 50 个 epoch 下生成的模型，之后把模型文件名改为 yolov4_maskdetect_weights4.pth，即可进行下一步的验证。

也可跳过模型训练这一步，直接取出百度网盘下载的另一个模型 yolov4_maskdetect_weights4.pth，直接放入./model_data 文件夹下进行后面的步骤。

三、模型评估

在主文件夹下找 eval.py 文件，点击运行，运行完成后在./mAP/文件下运行 main.py 文件，即可生成模型的评价指标图，生成的图在./mAP/output 文件下。

四、模型测试

将提供使用的 test 文件放入./data/test 文件夹下，并创建图 37 所示的文件夹

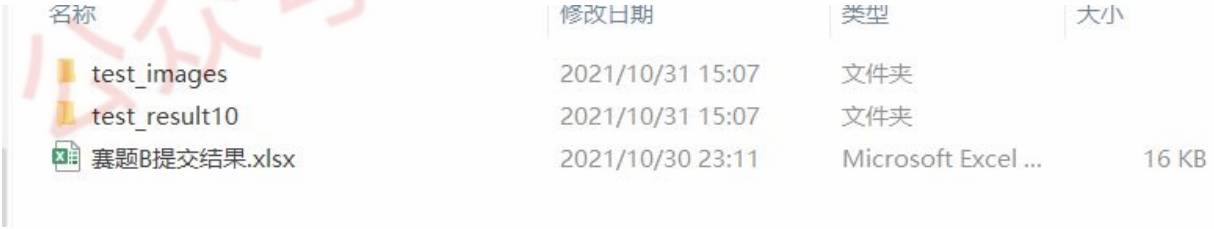


图 37　测试文件夹

直接运行 predict.py 文件，会把生成的图像可视化结果存入 test_result10(上传的支撑材料以包含了可视化的结果，可直接在改文件夹下查看)，将需要计算的指标存入 xlsx 文件下。