所在组别	2021 年中国高校大数据挑战赛	参赛编号
（本科组）		（bdc210686）

口罩佩戴检测分析

摘要

新冠疫情的爆发对人类的生命安全以及全球经济发展造成了严重影响。戴口罩成为了预防新冠肺炎最有效、最便捷的措施。自动化识别人脸佩戴口罩可以有效检测人群口罩的佩戴情况，基于以上背景，本文建立了 YOLOv5 目标检测模型对图片中口罩佩戴情况进行快速检测。

针对问题一、首先利用 Python 软件的 xml 模块读取 annotations 中的相关数据，然后借助 OpenCV 将其绘制在图片上，并另存于目录中。

针对问题二、由于附件中的 annotation 文件缺少必要的标签数据，不能直接使用，本文首先利用 labelimg 软件标记图片中的待检测区域，并用 VOC 格式将数据保存于相应目录中，再将其转换为 YOLO 格式数据。数据集划分完毕后，建立 YOLOv5 目标检测模型，将处理后的数据输入至 YOLOv5 模型中供其训练，最后通过训练精度、预测精度等一系列指标对模型做出评估，将其训练结果可视化，并对最终的检测结果进行分析。

针对问题三、基于问题二中训练所得模型，为测试集进行检测，并将评估结果与实际情况相对比，从而得到最终的检测详情表格数据。

本文对图像数据做预处理后，构建了 YOLOv5 模型学习口罩的佩戴情况，通过训练精度、预测精度等一系列指标后，其 IOU 为 0.92，大于 0.45，证明该模型的检测结果符合实际情况，具有良好的推广应用价值。

关键词：口罩佩戴检测、Python、OpenCV、YOLOv5 目标检测模型、可视化

一、问题重述

1.1 问题背景

新冠疫情的爆发对人类生命安全及全球经济发展造成了重大影响。虽然现在国内疫情基本得到有效遏制，但日常防控仍不可松懈。戴口罩是预防新冠肺炎最便捷、最有效的措施和方法。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，是抑制疾病在人流量大的公共场合快速传播和保护身体健康的重要技术手段。

图 1 中，每个方框（包围盒）框选出一张人脸，不同的方框颜色用于区分是 否佩戴口罩。

图 1 检测示意图

附件 1 训练样本中包含 train_images 和 train_anotations 两个文件夹，分别提供了 653 张图片和对应的标签信息，查阅相关文献，讨论以下问题。

1.2 问题提出

1. 每个标签文件（xml 格式）中，记录了相应图片中所包含人脸信息。见图 2，以标签文件 250.xml 为例，每个<objiect>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>则记录了该人脸的包围盒的左上、右下两个顶点的坐标。用程序读取 xml 中保存的数据，并参考图 1 的方式将人脸框和口罩佩戴状态进行展示，论文中重点对“250.png”和“477.png”进行展示。

2. 运用 653 张图片和对应的标签信息，设计人脸口罩检测算法，检测任意一张图片中存在的人脸的位置和口罩佩戴情况，并对算法进行校验。量化指标为<被正确分类的人脸的数量>/<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。

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，单独上传到竞赛平台。所有图片的标记文件需要放到项目源文件中上传。

二、基本假设

1. 假设正常人脸数据与背景区别较大，分辨率高。
2. 假设使用了背景虚化的图片背景人脸不再计算范围内。
3. 假设被无效物体遮挡的人脸数据为背景。

三、符号说明

符号	说明
λ_{coord}	表示坐标损失权重系数
λ_{noobj}	表示不包含目标的置信度损失权重系数
x_i	预测框坐标信息
y_i	预测框坐标信息
C_i^j	预测框置信度
P_i^j	预测框类别概率
$\hat{x}_i^j$	目标真实框坐标信息
$\hat{y}_i^j$	目标真实框坐标信息
$\hat{C}_i^j$	目标真实框置信度
$\hat{P}_i^j$	目标真实框类别概率
I_{ij}^{obj}	表示第i个网格的第j个先验框是否符合
I_{ij}^{noob}	表示第i个网格的第j个先验框是否不符
TP	正确检验框
FP	误检框
FN	漏检框数量

四、问题分析

4.1 问题整体分析

对于整体的三个问题，分析流程如下图所示：

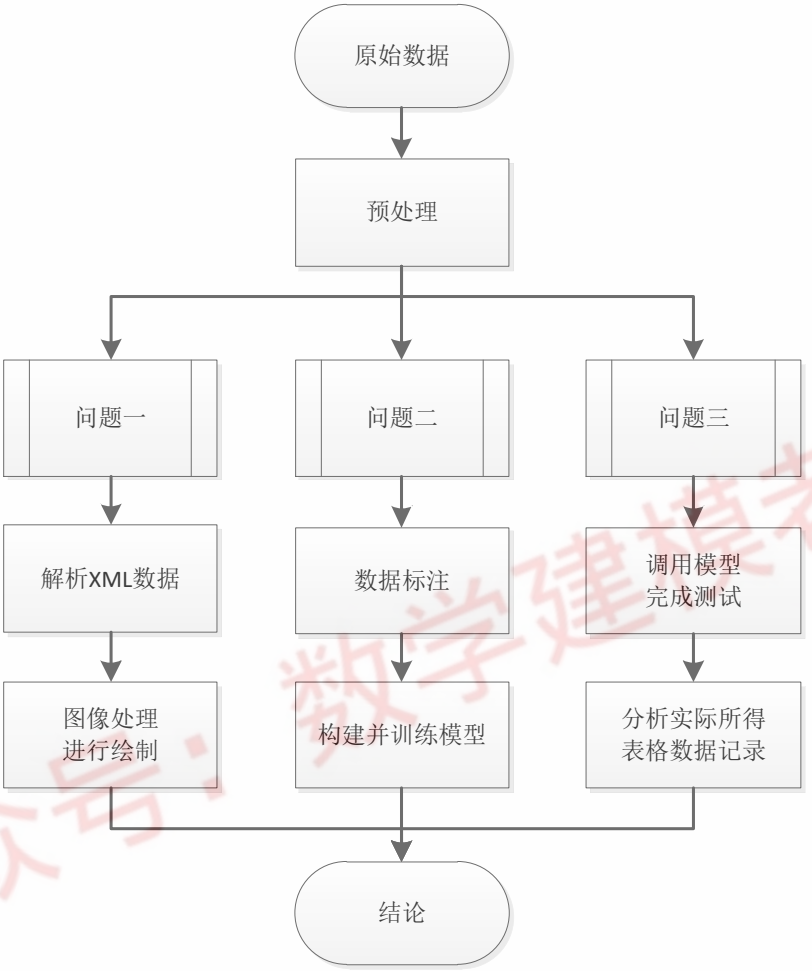

图 2 整体分析流程图

由图可知，首先解析 XML 数据，利用 OpenCV 将其绘制在图片上，其次进行数据标注，构建 YOLOv5 目标检测模型并训练该模型，最后通过调用模型的最佳权重，对测试集进行检测，并对其检测结果进行分析。

4.2 问题一分析

该问题需要根据已有的标签（附件中矩形检测区的顶点坐标、是否佩戴口罩的数据）为现有的 653 张图片进行待测区域边框的绘制。其步骤如下：

步骤一：解析 XML 数据文件，存储并解构所得对象，得到相关数据。

步骤二：采用 Python 的图像处理模块进行绘制。

步骤三：保存绘制所得结果。

4.3 问题二分析

该问题需要根据现有的图像数据，构建并训练 YOLOv5 目标检测模型，并通过训练精度、预测精度等一系列指标为其评估，其次选取合适的模型，其中，YOLO、SSD、FPN、SPP、R-CNN 等模型均为常用的目标检测模型，需根据不同的应用场合来选取最匹配的模型。

几种模型的性能对比如下表所示：

表 1 常用检测模型性能对比

Model	Fps	VOC 2007
R-CNN	0.077	48%-66%
SPP-net		63.1%-82.4%
Fast R-CNN		66.9%-70%
Faster R-CNN	15(ZF Model)	73.2%-85.6%
YOLO	45-150	58.8%
SSD	58-72	75.1%
R-FCN	6	83.6%

由上表显然可知，针对口罩佩戴情况的背景，由于 YOLO 模型的检测速度最快，可以很好的避免背景错误，且细微的定位错误、精度低等问题不影响整体的判别情况，故决定建立 YOLO 模型来检测图片中口罩的佩戴情况。

4.4 问题三分析

根据问题二中建立的 YOLO 模型，通过调用模型的最佳权重，将测试集输入该模型中，对测试集进行检测，并对测试结果进行分析。

五、问题一模型的建立与求解

5.1 问题一解题思路流程图

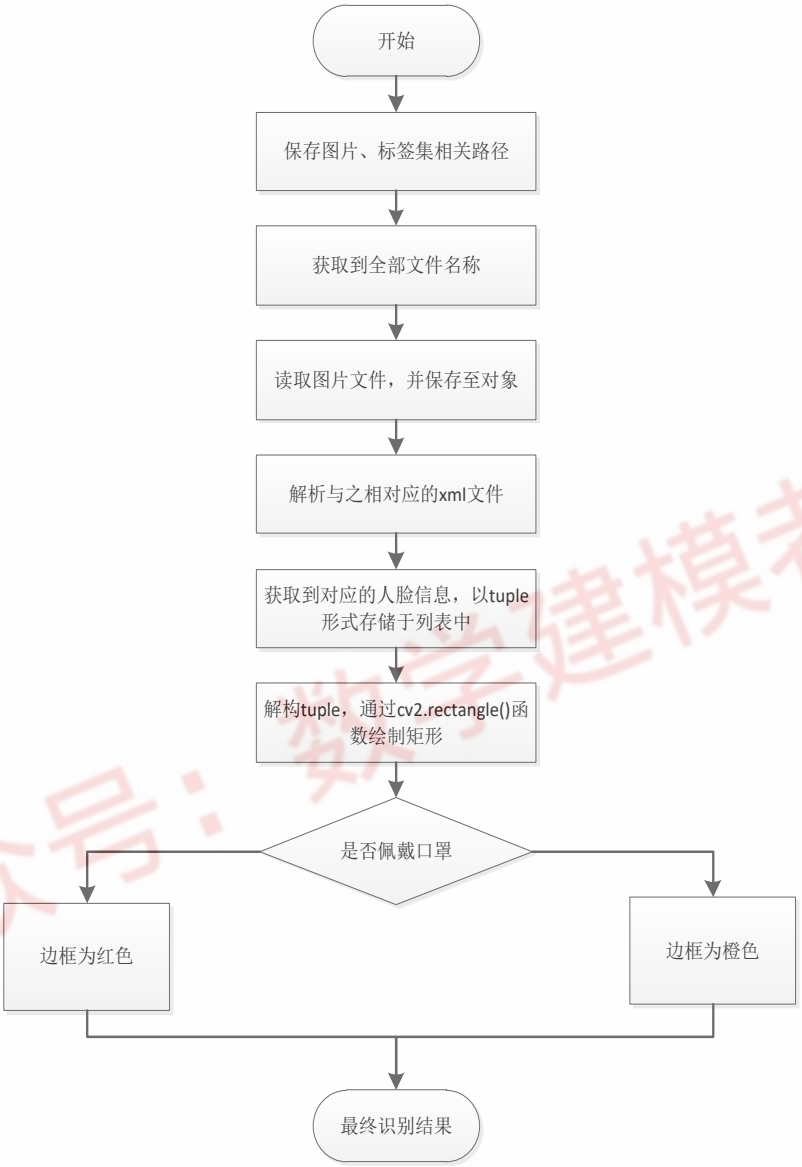

图 3 问题一流程图

首先将保存图片、标签集相关途径，获取其全部名称并读取附件图片信息，然后解析与之相对应的 XML 文件，获取与图片相对应的人脸信息，以 tupie 形式保存在列表中，最后解构 tuple，通过 cv2.rectangle（）函数绘制矩形（佩戴口罩为红色，未佩戴口罩或未正确佩戴口罩为橙色）。

5.2 数据预处理

5.2.1 转化格式

本文将数据集格的 VOC 格式转化为 YOLO 所需要的 txt 格式，其格式为（class_id,x,y,w,h），且为归一化值，因此需将其做相应转化，其运算规则如下，

$$x = x_{center}/width \quad (1)$$

$$y = y_{center}/height \quad (2)$$

$$w = (x_{max} - x_{min})/width \quad (3)$$

$$h = (y_{max} - y_{min})/height \quad (4)$$

其中$call_{id}$是类别 id 的编号，y_{max}，x_{max}，y_{min}，x_{min}分别代表 VOC 格式中 XML 的标记文件（相对于图片中标记物体所在位置右下角坐标值及左了上角的坐标值）。

5.2.2 数据统计

对部分数据进行描述性统计如下（完整数据见附件）：

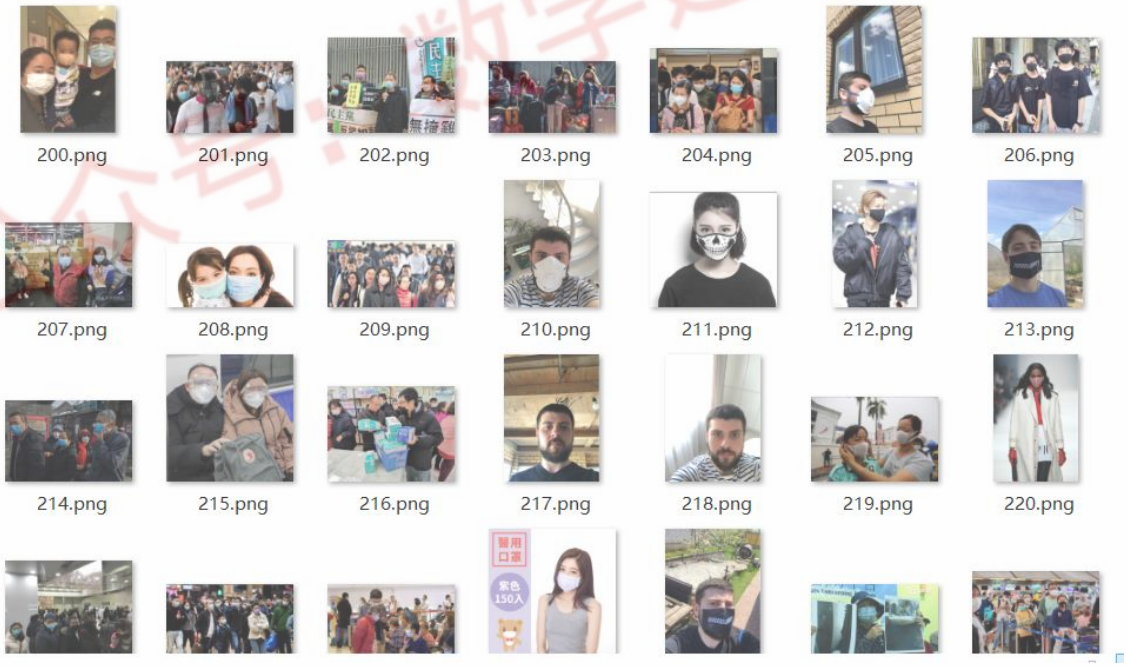

图 4 数据描述统计图

由上图可见，数据集为图像数据，其内容为在不同环境中若干佩戴口罩与否的人脸数据。由于需要对口罩进行检测，因此在本模型中，口罩为待检测的目标，需要使用 Annotations 中的 VOC 格式数据为口罩检测区进行标记。以 200.xml 为例，其 VOC 格式数据描述性统计如下：

表 2 数据描述统计表

	Name	Xmin	Ymin	Xmax	Ymax
1	With_mask	5	185	110	287
2	With_mask	115	130	180	192
3	With_mask	215	94	301	195

上述数据的每行对应着图像数据的一份人脸数据，即 xml 文件中的 object 标签。其中，name 为标识字段，With_mask 表示正确佩戴口罩，Without_mask 为未正确佩戴口罩或错误佩戴口罩；Xmin、Ymin 表示检测区矩形边框的左下角像素点坐标，Xmax、Ymax 表示检测区矩形边框的右上角像素点坐标。

5.3 模型求解

对于上述数据，首先需通过 xml 模块解析，得到数据后通过 xmin、ymin、xmax、ymax 获取待检测区域的区间，再通过 name 字段选择边框颜色（with_mask 对应红色，反之则对应橙色）。此处以 250.png 与 477.png 两张图片为例：

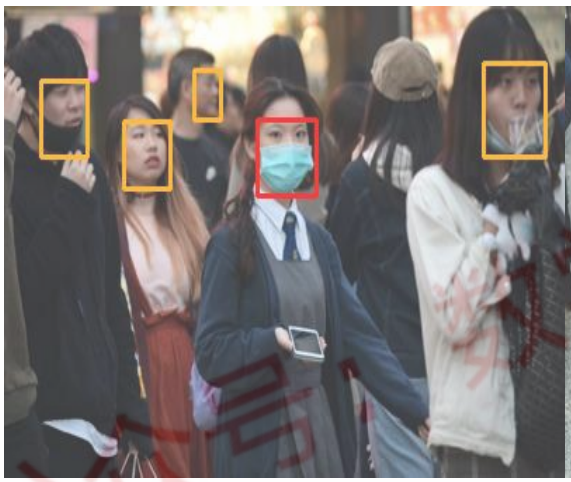

图 5 250.png 的绘制结果

图 6 477.png 的绘制结果

由上图可知，图 250.png 中存在不同状态如：正确佩戴口罩，未佩戴口罩，不正确佩戴口罩，侧脸等；图 477.png 中出现人脸数量较多（处于人流量较大的场合）。根据附件所提供的图片，及其对应的标签信息利用 OpenCV 软件可以将人脸框选出，并用不同颜色标记其佩戴口罩的情况；同时可知，在实际情况中，提高模型的检测精度以及检测速度十分重要。

六、问题二的模型建立与求解

6.1 问题二解题思路流程图

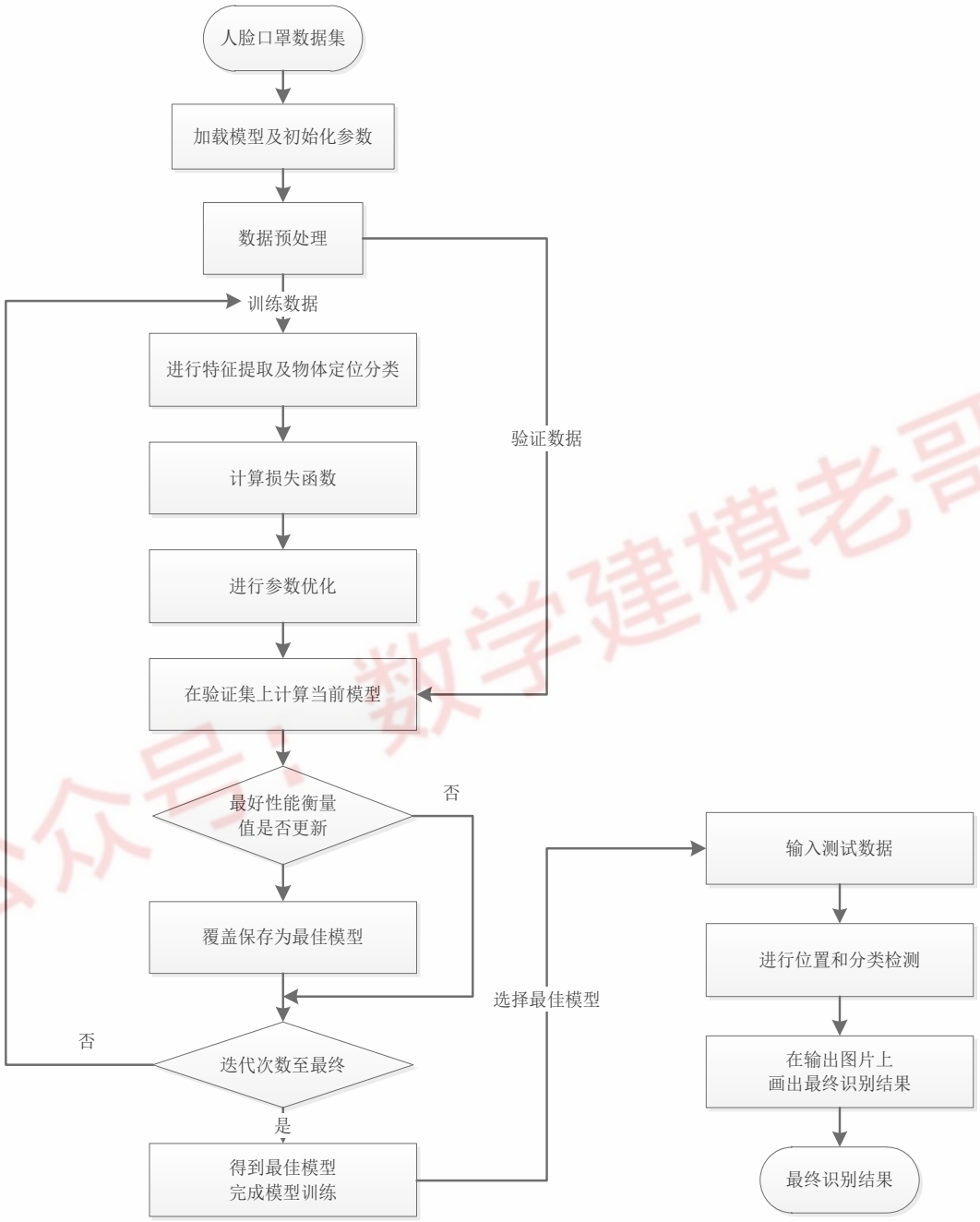

图 7 问题二流程图

由流程图可知，首先建立人脸口罩数据集，加载模块及初始化参数，数据预处理完成后，进行特征提取及人脸定位分类，计算其损失函数，将参数优化，再验证集上计算当前模型，将其保存为当前最佳模型，模型的训练次数达到后将得到当前最佳训练模型，最后输入测试数据对图片中人脸的位置和分类进行检测，在图片上标注其最终识别结果。

6.2 YOLO 模型的基本原理

YOLO 全称为 you only look once，即用来实现实时的目标检测的一种神经网络，其包括 24 个卷积层和 2 个全连接层。全连接层用来预测图像的位置和类别概率值，卷积层则用来提取图像的特征。其主要基本网络结构如下所示：

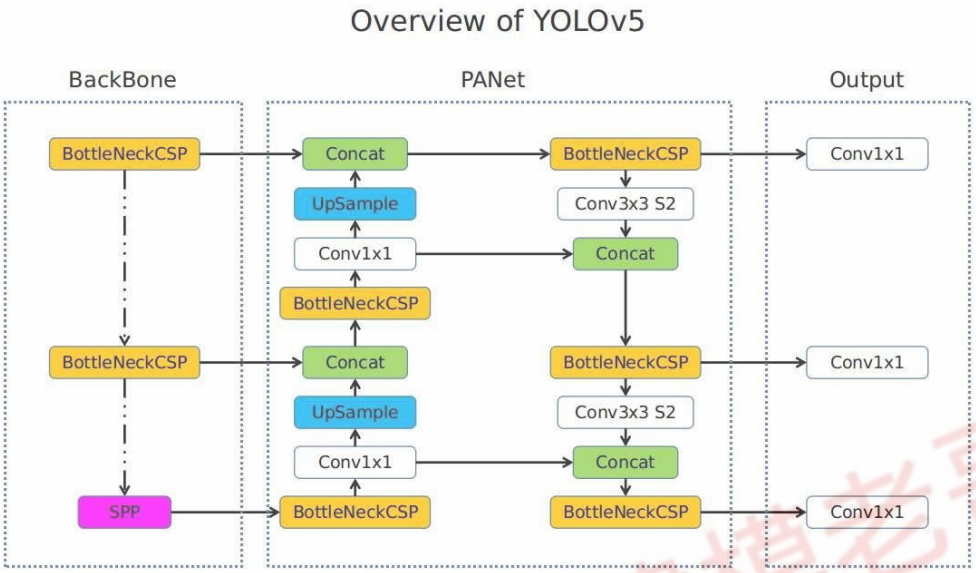

图 8 YOLO 模型基本网络结果示意图

其中，YOLOv5 的网络结构由 Neck、Backbone、Output(Prediction) 、输入端四部分组成。YOLOv5 的输入端采用了 Mosaic 数据增强，可实现自适应图像缩放与自适应锚框计算。Backbone 部分则采用了 CSP 网络结构和 Focus 网络结构，可以有效地减少参数量，从而达到降维效果，同时还可增加局部感受视野。相比使用的 step 为 2 的池化层或卷积层，Focus 网络层结构能有效地减少下采样带来的信息损失，以及减少计算量。Neck 部分中则使用了 FPN+PAN 结构，其中大量使用了 BottleneckCSP 模块，但由于本文采用了最新的 v5.5 版本，新的网络结构已将 CSP 模块修改为 C3 网络模块，但其结构相同。

最后的输出部分采用了 GIOU_loss 做 Bounding box 的损失函数，且仍保持了其加权 nms 非极大值抑制，因此对比 YOLOv4，使得原本被遮挡的目标不能被检测出来。

6.3 模型的构建

本文所设计的 YOLO 模型采用 v5 版本，其项目结构主要包括：data、utils、VOCdevkit、test.py 、weights、models、detect.py、train.py。其中：

表 3 名称说明表

名称	说明
data	目录，用于存放数据配置文件
models	目录，用于存放模型配置文件
utils	工具类模块
VOCdevkit	目录，用于放置训练数据

weights	目录，用于保存权重
detect.py	检测脚本文件
train.py	模型训练文件
test.py	模型测试脚本文件

使用框架前，需要对 yaml 文件作出修改，此处以 datasets 目录下的 voc.yaml 为例。将其另存为 mask.yaml 后，为其配置做出了如下修改：

```
# train and val data as 1) directory: path/images/, 2) file:
path/images.txt, or 3) list: [path1/images/, path2/images/]
train: VOCdevkit/images/train
val: VOCdevkit/images/val
# number of classes
nc: 2
# class names
names: ['without_mask', 'with_mask']
```

最后进行参数配置，将 train.py 中--weights、--data 、--cfg 字段的 default 属性改为自定义的文件：

```
parser.add_argument('--weights', type=str, default='weights/yolov5s.pt',
help='initial weights path')
parser.add_argument('--cfg', type=str, default='models/mask.yaml',
help='model.yaml path')
parser.add_argument('--data', type=str, default='data/mask.yaml',
help='data.yaml path')
```

配置完成后，即可开始训练，至此 YOLOv5 检测模型的搭建完成。

6.4 数据分布

可以根据 label 的相关性分布和分布图查看样本数据整体的情况，其中，label 的基本分布如下图所示：

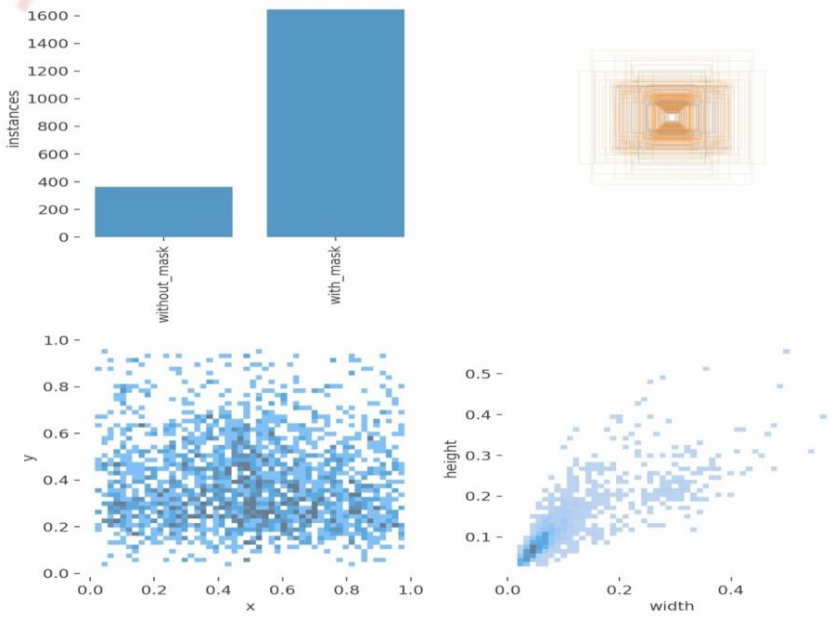

图 9 标签基本分布

由上图可知，第一行的两张图反映了数据集标签的分布情况，反例（即 without_mask）数量大致是 400，约为正例（即 with_mask）的四分之一。由此可见，数据集存在一定的正偏向性，但不会对最终的结果造成影响。

第二行的两张图反映了 y 与 x、height 与 width 的相关分布关系，像素点越密集、颜色越深的区域代表其分布密度越大。其两份密度的直方图的部分相关图如下：

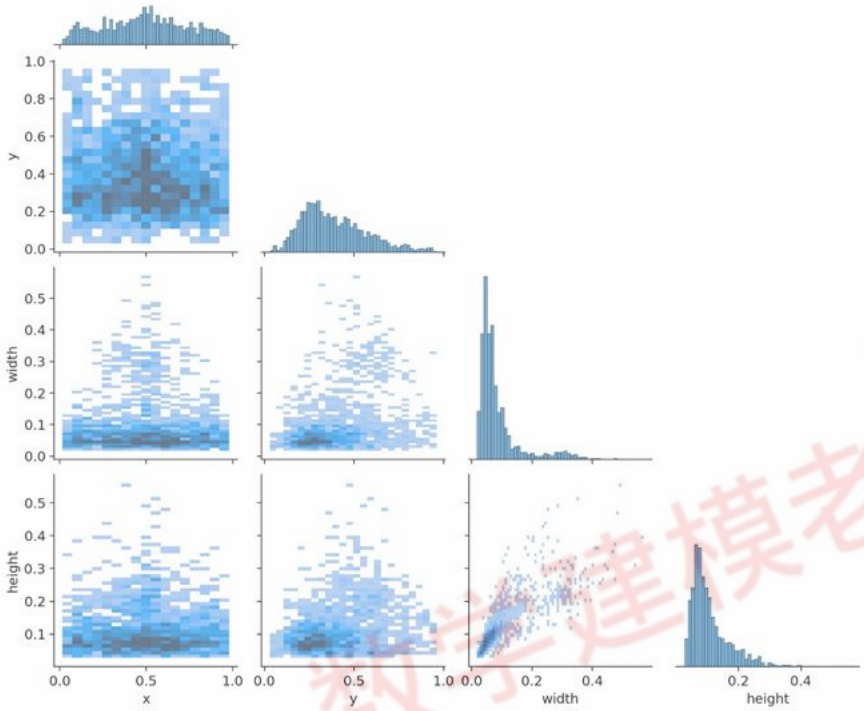

图 10 标签相关图

由上图可知，相关图刻画了不同标签之间相关的关系，通过相关分布可清晰地观察出各个标签间的相互关系及相互作用，如 height 与 x、height 与 y 等两者之间相关的关系。将下方密度的直方图堆叠起来，可得图标上方柱状图形式的密度分布的直方图，从而可以更加直观地观察其属性的分布情况。

6.5 基本网络结构

经上述步骤后，所构建出的模型网络结构如下表所示，这里只展示前五层网络结构，详细请移步至附录：

表 4 网络结构及其参数

	from	n	params	module	arguments
0	-1	1	3520	Focus	[3, 32, 3]
1	-1	1	18560	Conv	[32, 64, 3, 2]
2	-1	1	18816	C3	[64, 64, 1]
3	-1	1	73984	Conv	[64, 128, 3, 2]
4	-1	1	156928	C3	[128, 128, 3]
5	-1	1	295424	Conv	[128, 256, 3, 2]
…	…	…	…	…	…

6.6 训练结果可视化

模型训练完成后可通过其生成的图像查看模型评估指标的波动情况，其包括：recall 、mAP_0.5、precision 、mAP_0.5:0.95。其具体的波动情况如下：

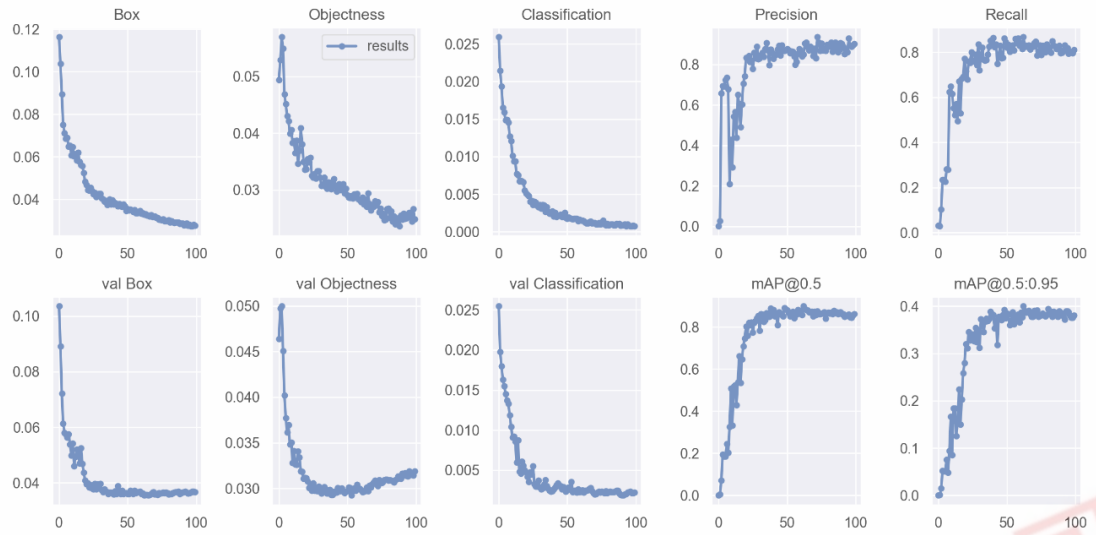

图 11 训练结果可视化

Tensorboard 所提供的评估指标可视化有限，但其可以进行交互，方便更好的观察相关变化情况，此处以静态图像为例进行分析。

由上图可知，左侧的 Box、Objectness、Classification 分别代表其三个损失函数在训练过程中的变化情况；下方带有前缀 Val 的代表其在预测过程中的变化。右侧四张图则分别代表其评估指标 Recall、PrecisionmAP@0.5、mAP@0.5:0.95 的变化情况。由此可见，在 100 轮迭代的学习过程中损失值的下降和评估指标到了一个允许范围内，因此，该模型可直接用于下一步工作中。

实验结果从标 Precision、Recall、mAP@0.5、mAP@0.5:0.95、模型大小、平均检测处理的时间、计算量、参数量八个角度综合衡量，其公式如下：

$$precision = \frac{TP}{TP+FP} = \frac{TP}{all\ detections} \tag{5}$$

$$\text{Recall} = \frac{TP}{TP+FN} = \frac{TP}{all\ ground\ tryths} \tag{6}$$

$$\text{A} = \int_0^1 pdr \tag{7}$$

$$mAP \frac{\sum_{i=1}^{N} AP_i}{N} \tag{8}$$

其中公式（5）、（6）中的 FN、TP、FP 和分别指漏检框数量、正确检验框及误检框。AP 值为 P-R 曲线的面积，并采用 101 个插值点计算的方法，因此结果更加精准。公式（8）中的 N 指检测类别总数，此处为 2。mAP@0.5:0.95 代表不同的 IOU 阈值所对应的平均 mAP，mAP@0.5 代表 IOU 设置为 0.5 时所有类别的平均 AP，IOU 取值从 0.5 至 0.95，步长 0.05。平均检测的处理时间包括了 NMS 处理及网络推理时间模型大小，所花费的时间代表了最终训练结束时得到的保存的模型大小。

6.7 损失可视化

该模型在训练过程中的损失波动如下：（具体的损失函数可视化，可通过

Tensorborad 查看，进入 localhost:6060 后即可查看如下可交互图表）

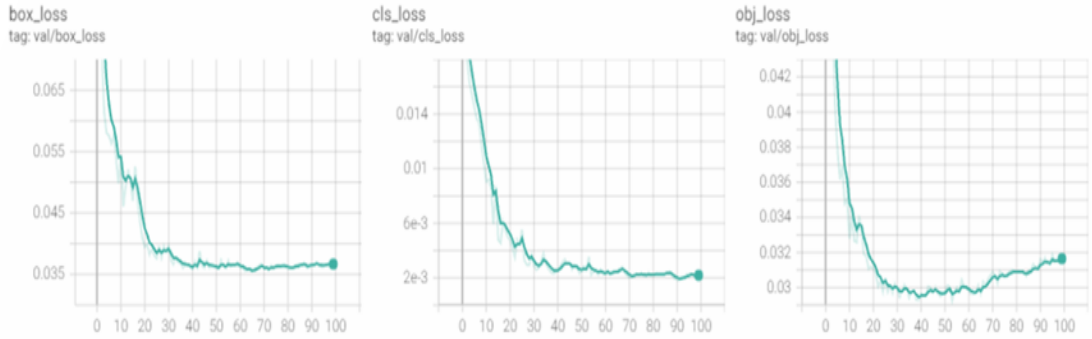

图 12 损失波动状况

上述可视化界面只能大致观测到以上三个损失函数的波动情况。YOLO 的损失函数由：分类损失、目标置信度损失和定位损失三部分组成。

6.8 改进损失函数

由上文可知，损失函数由分类损失、定位损失和目标置信度损失三部分组成，其中，损失函数如公式（9），边框在其回归过程采用了 MSE 损失函数，其公式如（10）所示，在分类损失和置信度上则采用了二元交叉熵的损失函数，其公式如(11)和式(12)所示：

$$Loss = Loss_{loc} + Loss_{conf} + Loss_{class} \tag{9}$$

$$Loss_{loc} = \lambda_{coord} \sum_{i=0}^{s^2} \sum_{j=0}^{B} I_{ij}^{obj} \left[\left(x_i - \hat{x}_i^j \right)^2 + \left(y_i - \hat{y}_i^j \right)^2 \right] +$$

$$\lambda_{coord} I_{ij}^{obj} \sum_{i=0}^{s^2} \sum_{j=0}^{B} I_{ij}^{obj} \left[\left(\sqrt{w_i^j} - \sqrt{\hat{w}_i^j} \right)^2 + \left(\sqrt{h_i^j} - \sqrt{\hat{h}_i^j} \right)^2 \right] \tag{10}$$

$$Loss_{conf} = -\sum_{i=0}^{s^2} \sum_{j=0}^{B} I_{ij}^{obj} \left[\hat{C}_i^j \lg\left(C_i^j \right) + \left(1 - \hat{C}_i^j \right) \lg\left(1 - C_i^j \right) \right] -$$

$$\lambda_{noobj} \sum_{i=0}^{s^2} \sum_{j=0}^{B} I_{ij}^{noobj} \left[\hat{C}_i^j \lg\left(C_i^j \right) + \left(1 - \hat{C}_i^j \right) \lg\left(1 - C_i^j \right) \right] \tag{11}$$

$$Loss_{cla} = -\sum_{i=0}^{s^2} I_{ij}^{obj} \sum_{c \epsilon classes} \left[\hat{P}_i^j \lg\left(P_i^j + \left(1 - \hat{P}_i^j \right) \lg 1 - P_i^j \right) \right] \tag{12}$$

其中，λ_{noobj}代表不包含目标置信度损失权重的系数，λ_{coord}代表坐标损失权重的系数w_i^j、y_i、x_i、h_i^j、C_i^j、P_i^j分别表示预测框的坐标信息、类别概率与置信度，$\hat{x}_i^j$、$\hat{h}_i^j$、$\hat{y}_i^j$、$\hat{w}_i^j$、$\hat{P}_i^j$、$\hat{C}_i^j$分别表示目标真实框的坐标信息、类别概率与置信度，I_{ij}^{obj}代表第i个网格中的第j个先验框是否负责此 2object（如果负责则I_{ij}^{obj}等于 1，否则就等于 0），I_{ij}^{noobj}代表第i个网格中的第j个先验框是否不负责此 object（如果不负责则I_{ij}^{noobj}则等于 1，否则等于 0）

6.9 学习率波动

首先介绍 YOLO 的余弦退火算法。其公式如下：

$$new\_lr = eta\_min + (initial - eat\_min) * \frac{1+\cos(T\_max - cur_{epoch} * \pi)}{2} \quad (13)$$

其中，$initial_lr$代表了初始的学习率；new_lr代表新得到的学习率；cur_epoch代表当前训练到某个$epoch$时对应的值；T_max代表示训练的总$epoch$数；eat_min代表了最小的学习率；$last_epoch$代表了最后一个$epoch$的$index$，如果是训练了多个$epoch$后中断了，这个值就是加载的模型的$epoch$，默认为-1，代表从头开始训练，即从$epoch = 1$开始。

其分析其学习率波动情况如下图所示：

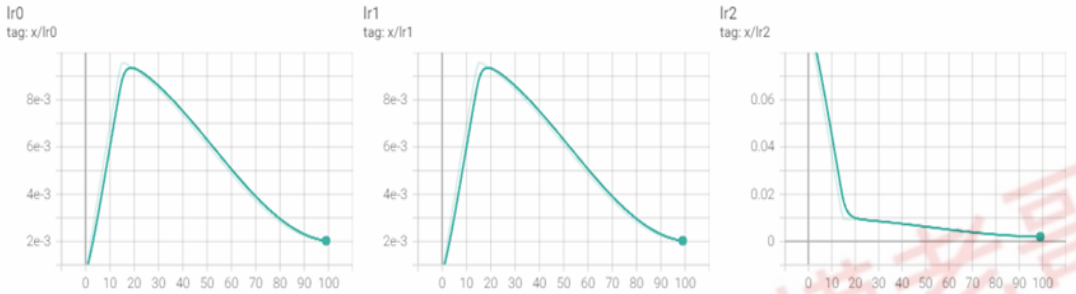

图 13 学习率波动状况

由图可见，其学习率由于使用了余弦退火策略，lr1 与 lr0 在经过训练前期的激增后，在训练后期得到了明显的下降；l2 则一直处于下降的状态，仅在极值点处变得更为缓和。

6.10 评价指标 IOU

目标检测中常用的评价指标为 IOU，其计算公式下所示，

$$\mathrm{IOU} = \frac{A \cap B}{A \cup B} \quad (14)$$

其中 B 代表真实框面积，A 代表预测框面积，IOU 是通过计算真实框和预测框的交集与并集的比值，表示其真实的边框与预测的边框的重叠程度，进而反应其检测结果是否可靠。

6.11 混淆矩阵

通过混淆矩阵，可以明显看出模型的拟合效果。其中，横、纵两轴分别表示实际输出与预测所得的结果，所围成的方形颜色越深，表明在这方面的预测工作置信度越高。

混淆矩阵具体如下图所示：

图 14 混淆矩阵

由上图可知混淆矩阵中心的 IOU 等于 0.92 大于 0.45 因此可认为图片中口罩佩戴情况被正确分类，且本模型在预测口罩佩戴情况时的置信度较高，而未佩戴口罩的置信度相对较低，而且犯假正例（FP）、假负例（FN）错误的概率非常低。导致前者出现的原因，可能如下：

1.在划分数据时，将未佩戴正确的样本划分为未佩戴；

2.划分时图像可能较为模糊，可能发生误判；

3.反例较少。

6.12 YOLOv5 模型结果检验分析

通过调用模型的最佳权重，从而对测试集进行预测，此处以部分具有代表性的图片为例：

图 15 80.png

图 16 91.png

图像 80.png（左侧图片）代表了图像中仅存在单个检测区的情况，且背景中

不存在模糊的人脸，因此得到了理想的结果，图像 91.png（右侧图片）代表了图像中包含多个检测区的情况。由图可以看出，部分检测区的 label（即图中的 with_mask 0.83 等信息）可能会造成检测结果的覆盖，但不会影响最终的检测结果，因此可以看出模型检测的准确性。

图 17 154.png

图 18 153.png

图像 153.png（左侧图片）则反映了标签重现问题。图片左侧人脸由于被背景影响，模型并没有将其识别为人脸；而图片右侧人脸则可被识别并准确检测出其佩戴了口罩。图像 154.png（右侧图片）代表了人脸未带口罩，且图像中仅存在一个检测区的数据，在有背景干扰。干扰且人脸区域有胡子的情况仍然准确识别其未带口罩。说明了模型检验的准确性。

七、问题三的模型建立与求解

问题三模型的检验与求解将问题二中建立的 YOLOv5 模型训练完成后，导入权重，对附件 2 测试样本文件夹下 test_images 中包含的图片进行检验，其部分数据见下表（完整数据见“赛题 B 提交结果”）。

图片名称	人脸数量	正确佩戴数量	未佩戴数量	未正确佩戴数量
0. png	3	1	2	0
1. png	4	3	1	0
2. png	4	4	0	0
3. png	9	8	1	0
4. png	1	1	0	0
5. png	3	2	1	0
6. png	1	1	0	0
…	…	…	…	…

由上文中 IOU 等于 0.92 可知，上表每张图片对应的人脸数量，正确佩戴口罩，未佩戴口罩数量，未正确佩戴口罩数量均可被认为知正确分类。

九、模型评价

9.1 模型优点

1.对于给定的图像数据，部分数据的分辨率较小，但数据集由自己标注，因此需要尽可能多的标注出分辨率低的样本，模型也能进行相应的学习。

2.本模型将目标检测任务转换成一个回归任务，极大提高了检测速度，且由于每个网格预测目标窗口时使用全图信息，会使得 FP 比例大幅降低。

3. 由于采取全图信息预测，相比于其他模型，本模型可以有效地降低背景预测错误率。

4.本模型可以学习到目标地概括信息，且比其他目标检测算法的准确率高很多。

9.2 模型缺点

1.对小目标的检测效果并不理想；

2.对相互靠近的物体检测效果不是很好；

3.容易产生定位错误。

十、参考文献

[1]叶茂，马杰，王倩，武麟. 多尺度特征融合的轻量化口罩佩戴检测算法[J/OL]. 计算机工程. https://doi.org/10.19678/j.issn.1000-3428.0062231
[2]王艺皓，丁洪伟，李波，等.复杂场景下基于改进 YOLOv3 的口罩佩戴检测算法[J]. 计算机工程， 2020， 46(11): 12-22.
[3]余阿祥，李承润，于书仪，等.多注意力机制的口罩检测网络[J].南京师范大学学报(工程技术版)，2021，21(01):23-29.
[4]王兵，乐红霞，李文璟，等.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(08):62-69.
[5]万子伦，张彦波，王多峰，等. 复杂环境下多任务识别的人脸口罩检测算法［J］. 微电子学与计算机，2021，31(10)：21-27.
[6]叶永雪，马鸿雁. 基于关键点与迁移学习的口罩佩戴检测研究. 计算机仿真. https://kns.cnki.net/kcms/detail/11.3724.TP.20210914.1901.002.html
[7]向志华，贺艳芳.深度迁移学习的相干斑噪声图像标注算法研究[J].计算机仿真，2020，37(4):397-401.
[8]谈世磊，别雄波，卢功林，谈小虎.基于 YOLOv5 网络模型的人员口罩佩戴实时检测[J].激光杂志，2021，42(2):147-150.
[9]许景辉，邵明烨，王一琛，韩文霆.基于迁移学习的卷积神经网络玉米病害图像识别[J].农业机械学报，2020，51(2):230-236+253.
[10]许德刚，王露，李凡.深度学习的典型目标检测算法研究综述[J].计算机工程与应用，2021，57(8):10-25