《机器视觉与目标检测》

课程名称：《机器视觉与目标检测》

任课教师：孙翠敏

论文题目：《基于 yolov8 改进网络的电动车头盔检测》

学生学号：2134110122

学生姓名：李享哲

学院专业：计算机与电子信息学院　　计算机类

课程学期：2024-2025 学年第二学期

目录

0 引言

随着经济社会的稳步推进，机动车数量急剧上升。作为重要的短途交通工具，电动车在居民日常出行、短途接 送和共享出行中占据主要地位。据统计，我国电动车社会 保有量已超过 3.5 亿辆[1]，因其具有灵活性、区域性、峰谷 性、违法率较高、混行性的交通流特点，引发的交通安全事 故也在增多[2]。数据显示，2019 年—2024 年期间，非机动 车交通事故死亡人数同比增长 5.85%，受伤人数同比增长 4%。在已发生的电动自行车驾驶人员死亡事故中，约 80% 为颅脑损伤致死，而佩戴安全头盔能有效降低事故发生所 造成的伤害。 为了降低电动车驾驶员在事故中的受伤程度，相关部 门大力推行“一盔一带”活动[3]，促使人们在骑行电动车时 佩戴头盔，养成安全驾驶习惯。

近年来，深度学习[4]在目标检测领域取得了显著进展，成为研究的重点。研究人员通过不断优化高 效且结构简洁的网络模型，推动了深度学习技术的广泛应用。但是，目标检测任务仍然面临诸多挑战， 尤其是在复杂背景下的小目标检测。针对这些挑战，学者们已开展了大量的研究工作。例如，Simbolon 等[5]提出了一种模型，首先检测人脸，然后利用主成分分析实现头盔检测，该方法中双三次插值技术对 检测精度的提升起到了关键作用；Ge 等[6]提出了一种基于 YOLOv5 的头盔佩戴检测方法，专注于工地场 景下的安全帽检测；Cheng 等[7]提出了一种基于 SAS-YOLOv3-Tiny 的多尺度安全头盔检测算法，通过引 入通道注意力机制替代卷积层，获取更有用的特征信息，在一定程度上提高了识别速度。

一些学者也提出了一些创新方法。Yan 等[8]开发了一种双通道卷积神经网络（DCNN）模型，通过引 入随机森林（RF），有效消除复杂背景的影响，提高了识别精度，但时间复杂性随之增加。刘备战等[9] 基于 Retina Net 网络，将残差网络融合到全连接层以提升目标检测精度，但实时性尚需改善。赵心驰等[10] 将 SE 注意力机制和 XBGoost 模型结合，用于人体摔倒检测，该方法具有高准确性和鲁棒性，但实时性 方面仍存在不足。王新等[11]基于 SSD 网络提出了 EfficientNetV2-SSD 检测模型，通过采用轻量级 EfficientNetV2 网络替代 VGG16，并结合 FPN 网络和新的初始锚框设计，提升了小目标检测能力。 尽管已有研究在提升目标检测精度和速度方面取得了进展，但仍存在一些不足。例如，在复杂背景 下，小目标的检测效果尚不理想，定位精度有限等。这些问题制约了目标检测算法在电动自行车头盔检 测任务中的实际应用。因此，进一步研究一种能够提高小目标检测能力并增强鲁棒性的头盔检测方法具 有重要意义。

1 电动车驾驶员头盔佩戴数据集收集制作以及处理

在目标检测任务中，数据集的重要性显而易见。为了检测电动车安全头盔的佩戴情况，并且结合上课所学习到的知识。

为确保数据集的规范性和易用性，我对筛选后的图片进行了重命名和编号排序。 接着，选择 labelImg 工具展开对图像的标注工作。在标注阶段，可将图像数据集拆分 出三种类型：骑电动车的人（命名为 motor）、安全头盔（命名为 helmet）、未佩戴安 全头盔的驾驶人的头（命名为 head）。

1.1 Kaggle 公开数据集

Kaggle 是由联合创始人、首席执行官安东尼·高德布卢姆（Anthony Goldbloom）2010 年在墨尔本创立的，主要为开发商和数据科学家提供举办机器学习竞赛、托管数据库、编写和分享代码的平台。

我在 Kaggle 平台（https://www.kaggle.com/）上搜索 Helmet，出现多种 Heltmet Detection 数据集。

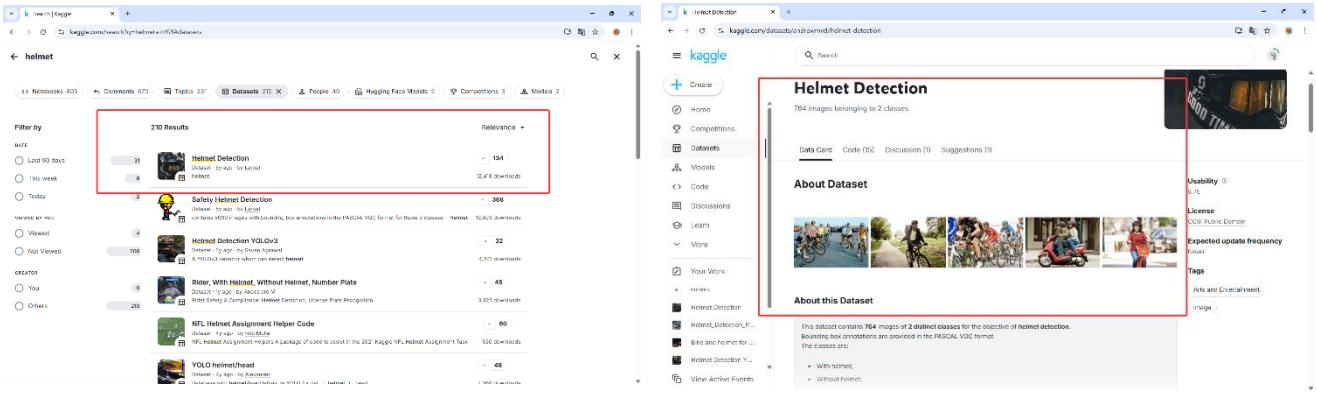


图 1 Kaggle 中 Helmet 数据集下载

1.2 广西大学数据

1.2.1 电动车驾驶员图像采集方法与过程

为确保实验结论的准确性和可靠性，数据集的制作必须严格遵循两个核心条件：首先，数据集中的图片需涵盖不同的角度、天气状况、遮挡程度以及拍摄距离，以模 拟各种实际场景中的变化；其次，为确保模型在训练过程中不对某一类别产生偏见， 应使数据集涵盖着多个类别的丰富的目标，且需确保其各目标数量呈均匀分布的状态。

鉴于此，本研究将自行制定相应的数据集，其拆分出两种来源，其一为通过 Python 爬虫技术在网络渠道上采集电瓶车骑手佩戴头盔的相关图片；其二是在不同城 市交通主干道周边的路口进行实地拍摄获得的电瓶车骑行人员图像。

（1）不同的地点。图 2-3 所示，我们的实地拍摄数据集分别来源于西大南门、 西大东门等不同路段的采集点，这些地点的多样性为模型提供了更丰富的学习素材和更强的泛化能力。

图 2-3 左图（西大南门地铁口）| 右图（西大东门天桥上）

（2）不同的时段。我安排了不同时段的图像采集时间，目的是确保自制数据集能还原实际的城市交通道路场景。图 4-5 展示了我们在不同时间段拍摄的样本示例。 这样的安排旨在捕捉不同时间段因自然光照变化而导致的图像色差和光照条件的变化。 通过学习和适应这些变化，模型有望在实际应用中更好地应对各种复杂的光照条件， 从而提高检测的准确性和鲁棒性。

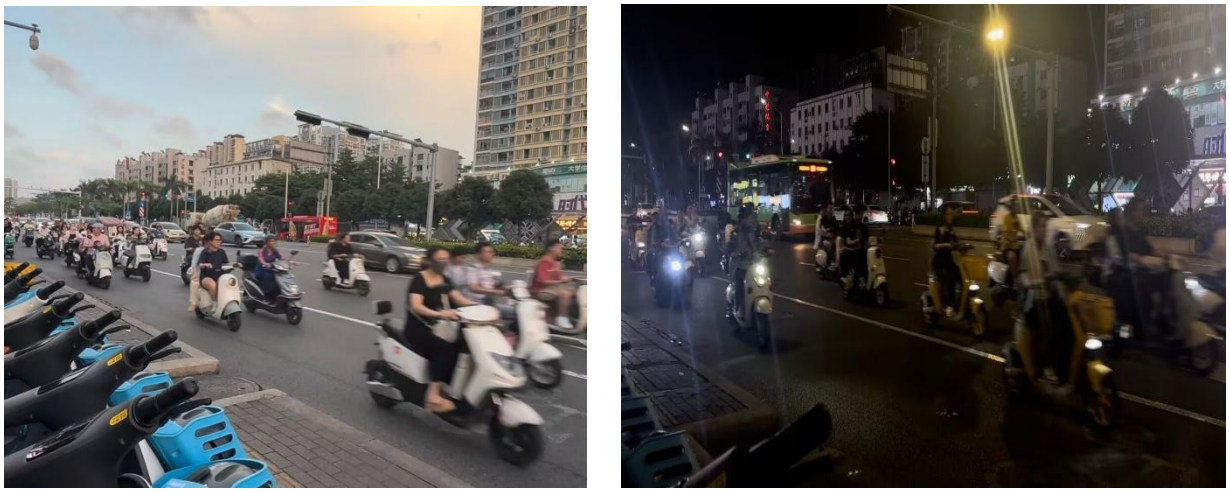


图 4-5 左图（西大南门地铁口白天） | 右图（西大南门地铁口晚上）

1.2.2 数据集统计分析与处理

为确保数据集的规范性和易用性，我对筛选后的图片进行了重命名和编号排序。 接着，选择 labelImg 工具展开对图像的标注工作。在标注阶段，可将图像数据集拆分 出三种类型：骑电动车的人（命名为 motor）、安全头盔（命名为 helmet）、未佩戴安 全头盔的驾驶人的头（命名为 head）。

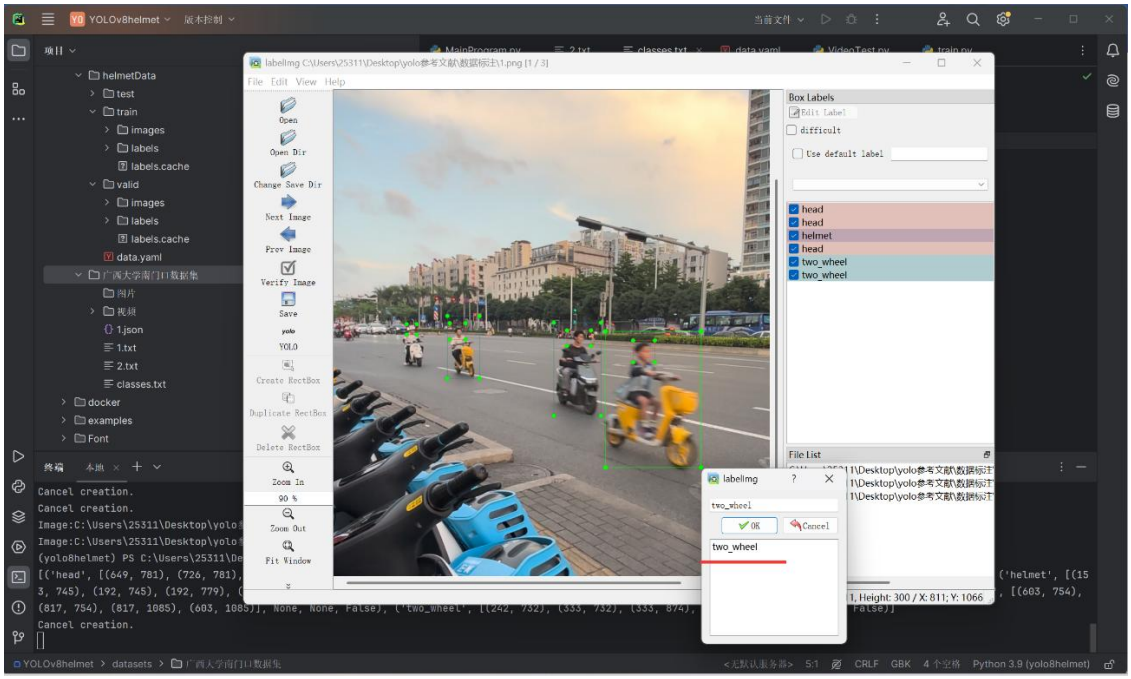


图 6 labelImg 标注界面

1.2.3 数据的增强

为了增强模型的鲁棒性，本文重点采用了基于几何变换和色彩空间变换的数据增 强方法。在几何变换上，我们采取了移位、旋转、镜像翻转、裁剪以及缩放等操作， 以助力模型掌握目标在各种空间姿态下的特征。在色彩调整方面，我们改变了图像的 亮度和饱和度，以增强模型对不同光照和色彩环境的适应性。

另外，针对语义分类任务中常见的正负样本不均衡问题，我们实施了类别均衡采样策略。鉴于数据集中佩戴与未佩戴头盔的样本数量悬殊，我们特别增加了少数类别 的样本量，以确保模型在训练时能均衡学习各类目标特征，从而提升整体检测效果。

（1）镜像翻转，特别是水平或垂直方向的翻转，是图像处理中常见的增强技术。 它可以被看作是旋转角度为 90、180、270 度的特殊仿射变换。图 7-8 展示了水平镜像 翻转的效果，通过这种变换，可使模型充分学习不同空间方向的待测对象特征，以期 强化训练结果的可泛化性以及提高其检测精度。

图 7-8 水平镜像翻转效果图(代码参考附录)

（2）随机裁剪是一种有效的图像增强技术，它通过随机选择图像的一个区域并将 其裁剪到指定大小，以增加模型的泛化能力。裁剪后的图像尺寸通常小于原图，且裁 剪位置在图像边界内随机确定。图 9-10 展示了随机裁剪的效果，这种方法有助于模型 学习并识别图像的不同部分，从而提高检测精度和鲁棒性。

图 9-10 随机裁剪效果图(代码参考附录)

（3）图像随机旋转是一种实用的数据增强策略，它模拟了实际道路摄像头在不同角度下的拍摄情况。通过将图片沿一定角度进行调整并加入数据集进行扩充，我们可 以增强模型的泛化能力和对不同拍摄角度的适应性。图 11-12 展示了顺时针旋转 30 度的 效果，这种变换有助于模型更好地应对实际应用中可能出现的各种角度变化，从而提 高检测精度和稳定性。

图 11-12 随意旋转的效果图(代码参考附录)

（4）色彩扰动。通过对原始图像的颜色通道进行扰动和噪声处理，目的是增强检 测模型对颜色和光照变化的适应能力。图 14-16 展示了色彩扰动的效果图，直观地呈现 了这些色彩调整方法如何改变原始图像的视觉效果，进而增强模型的泛化能力。

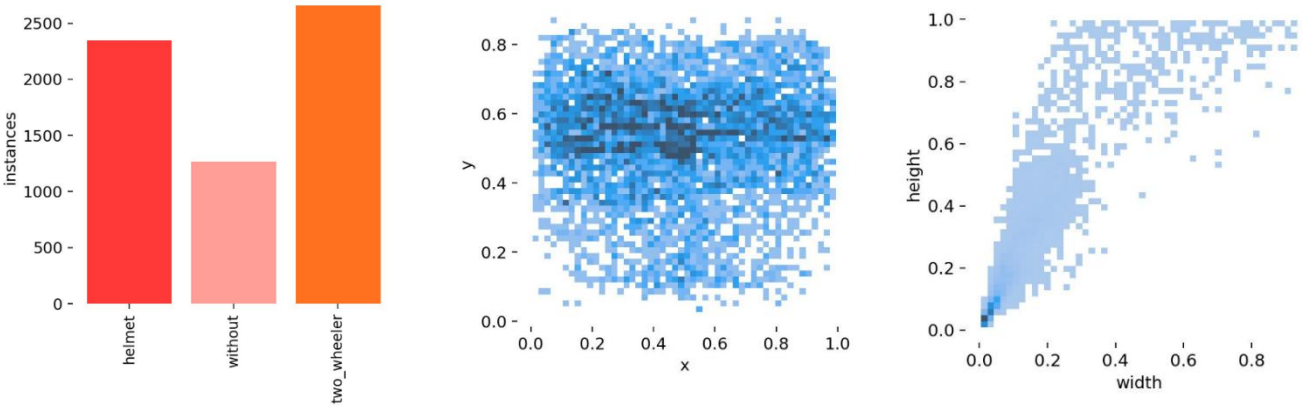
图 14 色彩处理前原图

图 15-16 色彩抖动效果图(代码参考附录)

上述深入阐述了在构建数据集时所使用的多种图像增强技术。这些技术通过丰 富训练数据的多样性和复杂性，成功地减轻了模型的过拟合问题，进而提高了模型的 稳定性和泛化能力。

1.2.4 数据集类别统计及分布

数据集中类别不平衡问题也常常会造成模型欠拟合，因此统计不同类别之间的样 本数量差异对提高模型的整体性能具有重大意义。



(a) 数据集目标类别分布　　(b) 目标中心点分布　　(c) 目标大小分布

图 17 头盔佩戴检测数据集分布图

图 17 为标注后数据集的类别分布可视化结果:

- 其中图（a）反映了数据集中不同 类别目标的数量分布情况，helmet 类的目标数量为 2400 左右，without 类的目标数量为 1250 左右，two_wheeler 类目标数量为 2600 左右。
- 图（b）展示了目标对象中心点的位置分布，其中横坐标 x 和 纵坐标 y 代表其中心点的坐标位置（ x y 、 值已进行归一化处理）。图（b）中颜色越深 的位置，表示该位置的目标框中心点越集中，能反映出目标对象在图片所处的位置。从图中我们还可以观察到当前数据集中存在目标遮挡的情况，这有助于训练并提高模 型对含遮挡目标的检测效果。图(c）则展示了目标物体的大小分布，横坐标 w 和纵坐 标 h 分别代表目标的宽度和高度（宽高值也已进行归一化处理）。
- 图（c）中则显示数 据集中小目标的占比较大，这有利于训练并提升模型对远距离小目标的检测效果。

2 基于改进 YOLOv8n 的头盔佩戴检测算法

深度学习技术的迅速发展，尤其是卷积神经网络 （CNN）的引入，为目标检测带来了新的可能性和应用。目前，目标检测的主流框架可大致分为两大类：单

阶段和 两阶段方法。著名的两阶段目标检测算法包括空间金字塔 网络(SPP-Net)[12]、R-CNN[13] 和 Faster R-CNN[14]。单阶段 检测算法较两阶段算法模型更简单。其代表性算法包括 YOLO[15]、SSD[16]、RetinaNet[17]、CornerNet[18]和 Refine-Det[19]。随着单阶段算法的成熟，其不仅在检测速度上远 超两阶段算法，而且在检测精度上也与两阶段算法不相上 下，甚至在某些场景下更胜一筹。尽管目前的技术水平很 高，但其在复杂场景、小型目标检测、实时性等方面仍有改 善空间。

2.1 YOLOv8 算法机理

YOLOv8 的网络架构呈现为一个四部分的构成体系，具体涵盖输入端 Input 层、主 干网络模块 Backbone 层、特征融合增强模块 Neck 层，以及输出端 Head 层。这一体系 结构设计使得 YOLOv8 在目标检测任务中能够展现出卓越的性能。为直观展示其网络 结构，我特绘制了如图 18 所示的 YOLOv8 网络架构图。

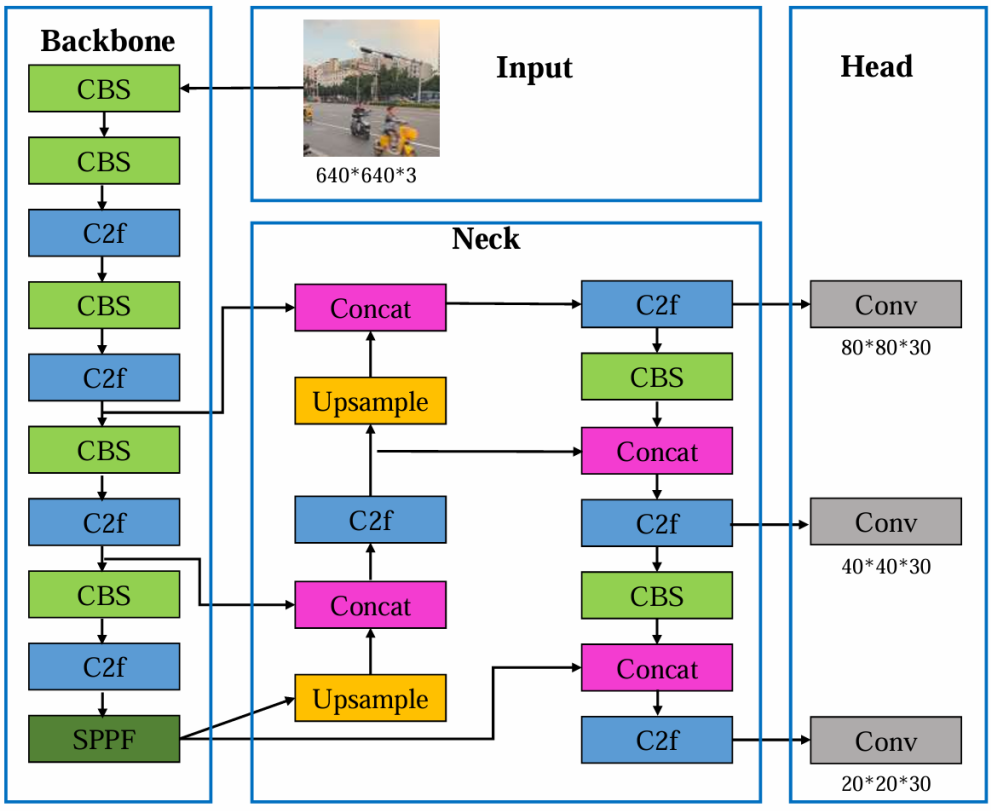


图 18 YOLOv8n 网络结构图

2.2 改进后的 YOLOv8 算法介绍

在 Neck 网络中，上采样（Upsample）操作后的特征图可能存在信息粗糙、细节缺失或噪声等问题，因此在后续添加模块通常是为了优化特征质量、增强模型表达能力或适应特定任务需求。

为了解决上述问题，对网络进行了如下改进：

- 引入自己参与研究生师兄课题论文中设计的 ASAM 注意力模块[20]

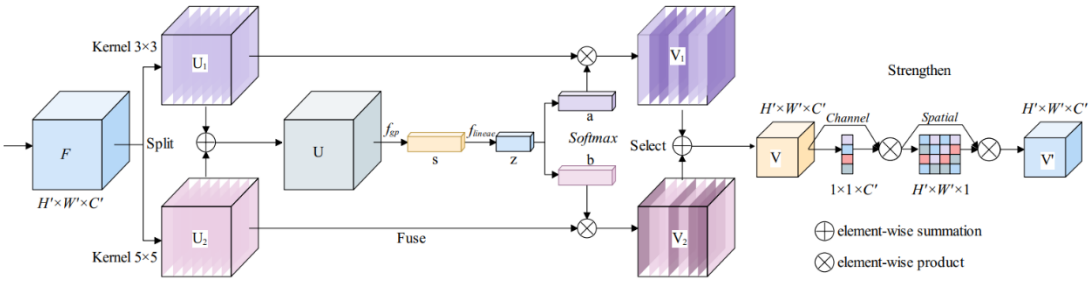


图 19 ASAM 注意力模块

- 上采样后通过跳跃链接融合不同尺度的特征

改进后的网络结构图如下图 20。

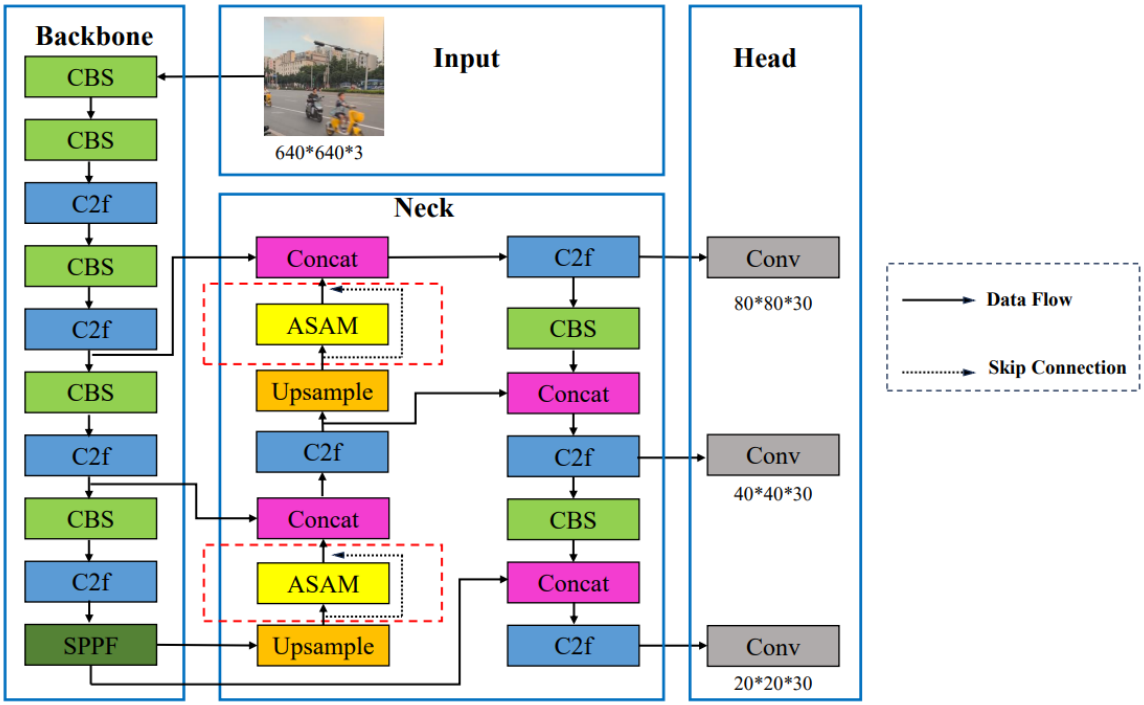


图 20 改进后的 YOLOv8 网络结构图

3 改进 YOLOv8n 算法的头盔佩戴检测实验分析

基于本文自制数据集的特点，针对密集环境下含遮挡或者小尺度目标的检测情境， 上文已研究出一种模型优化方案。在自定制数据集 Helmet 进行对比实验和消融实验， 验证了这些改进方法对于密集环境下含遮挡目标或者小目标检测具有显著优化的效果。 实验结果显示，优化后的模型具有更高的检测精度。

3.1 实验准备

3.1.1 深度学习框架

在电瓶车驾驶员头盔佩戴检测研究中，本文选用 PyTorch 作为深度学习框架。 PyTorch 以其灵活性和易用性著称，支持动态计算图和高效 GPU 加速，非常适合处理 图像数据和训练复杂模型。

3.1.2 参数设置与环境配置

本章将列出实验软硬件环境、设置参数，并介绍精准率、召回率和均值平均精度 等评价指标。通过实验的形式对 YOLOv8n 及 改进后的 YOLOv8n 在自制的头盔数据集上进行测试，对比并分析实验结果。实验参数 如表 1 所示。

表 1 实验参数设置

参数名称	数值
batch size	32
学习率	0.1
冲量	0.937
权重衰减	0.0005
Epoch	150

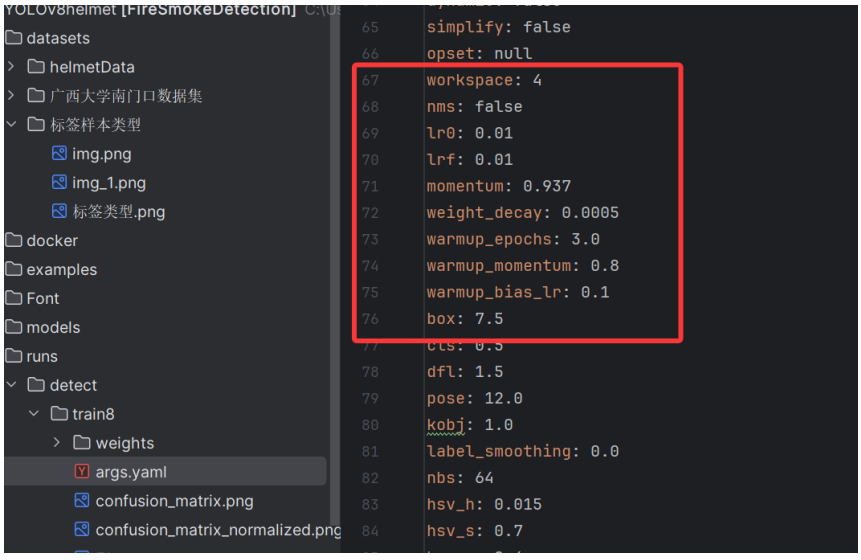


图 21 试验参数

鉴于目标检测算法依赖于大量训练数据，且数据集参数量庞大，因此一般在搭载 了 GPU 的服务器端展开模型的训练。表 2 清晰陈列了本章实验建议的硬件配置，表 3 清晰呈现了实验中需用到的各类软件及工具的基础信息。

表 2 系统硬件配置

操作系统	Windows 10 专业版
CPU	12th Gen Intel(R) Core(TM) i5-12400F 2.50 GHz
GPU	NVIDIA GeForce RTX 4060 Ti
RAM	32.0 GB (31.8 GB 可用)
硬盘	150

图 22 系统硬件配

表 3 系统软件配置

名称	版本	作用
PyTorch	1.11.1	支持从模型定义、训练到部署的完整流程。
CUDA	12.6	利用 GPU 加速并行计算能力
Anaconda	3	集成编程、数据分析、机器学习环境
Pycharm	2023.3.3	用于编写、调试、测试和管理 Python 代码
Git	3.2.14	开源分布式版本控制系统

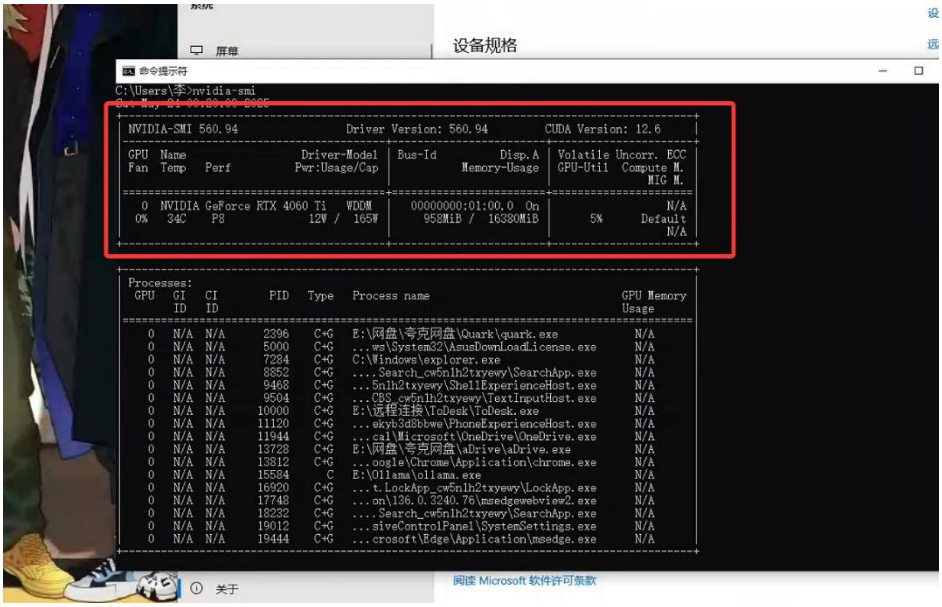


图 23 系统软件配置

3.2 实验评价指标

目标检测模型性能主要通过精准率（P）、召回率（R）和均值平均精度（mAP）来评估。精准率反映预测正样本的准确性，召回率体现找出真正正样本的能力。mAP 综合考虑两者，衡量模型在不同召回率下的性能。这些指标共同构成目标检测模型性 能的全面评价体系，帮助研究者全面、准确地了解模型的性能表现。

3.2.1 精准率 P 和召回率 R

无论是精准率，还是召回率，均由二分类问题得出，此处对二者进行统一介绍。 二分类问题表示的是向某给定系统中输入相应的样本（涵盖着真实值），而仅得出两 种输出结果。对于二分类问题，精准率用于表征真实值属于正类的前置条件下，其 TP （真正例）所占的比例；召回率用于表征预测值属于正类的前置条件下，其 TP 所占的 比例。上述二者的数学描述分别由式（1）和式（2）给出。

$$Precision = \frac{TP}{TP+FP} \tag{1}$$

$$Recall = \frac{TP}{TP+FN} \tag{2}$$

3.2.2 均值平均精度 mAP

在图像处理与目标检测领域，精确率（Precision，简称 P）是衡量某一类别目标在 单张图像中检测准确性的重要指标。而当我们需要从整个数据集层面评估某一类别目 标的检测性能时，则引入平均精度（Average Precision，简称 AP）作为关键评价指标。 具体计算公式参见式（3）

$$Average\ Precision_A = \frac{\sum_1^n Precision_A}{n} = \int_0^1 P(R)dR \tag{3}$$

进一步地，为了评估模型在整个数据集中对所有类别目标的检测性能，我们引入了 均值平均精度（mean Average Precision，mAP）。mAP 的计算方法是对数据集中各个类 别的 AP 值进行平均，具体计算公式如式（4）所示。通过这种方式，mAP 提供了一个 全面且综合的指标，用于衡量目标检测模型在整个数据集上的性能表现。

$$Mean\ Average\ Precision_A = \frac{\sum_1^m Average\ Presion}{m} \tag{4}$$

式中符号 m 指的是数据集的待检测目标类别总数，本文的第三章已经指出，本自制数据 集共含有 3 中目标类别，因而满足 m=3。

3.3 实验结果及其分析

3.3.1 对比实验与效果展示

我选取 YOLO 系列中较为优异的 v8 算法与我们改进后的 YOLOv8n 模型进行比较。

表 4 不同算法检测性能的对比

Model	Precision	Recall	mAP(0.5)	mAP(0.5-0.95)
YOLOv8n	79.16%	59.43%	67.21%	46.98%
Ours	80.81%	60.23%	68.35%	47.55%

3.3.2 消融实验与效果展示

（1）消融实验结果分析

表 5 消融实验

Model	Precision	Recall	mAP(0.5)	mAP(0.5-0.95)
YOLOv8n	79.16%	59.43%	67.21%	46.98%
+ASAM	80.21%	59.99%	68.00%	47.12%
+ASAM+Skip-Con	80.81%	60.23%	68.35%	47.55%

根据上述消融实验结果，我们得出结论：将 ASAM 模块和 Skip-Con 跳跃链接融入 原始 YOLOv8n 模型，具体来说，改进后模型的 mAP 较基准模型高了 0.57%证明改进方案有效。

（2）PR 曲线

将本文所提出的经由改进的 YOLOv8n 与基准模型展开比对，输入相关数据，通过 YOLOv8n 的可视化工具，可得出图 24（改进模型）所示的 PR 曲线。

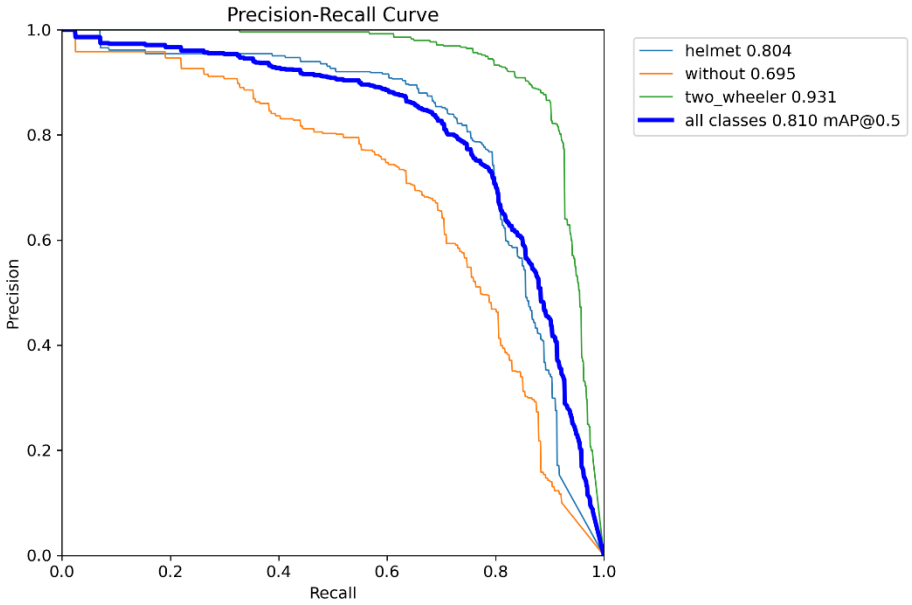


图 24 改进后 YOLOv8n PR 曲线

（3）计算结果

损失函数可用于评估模型的整体收敛情况如图 25。

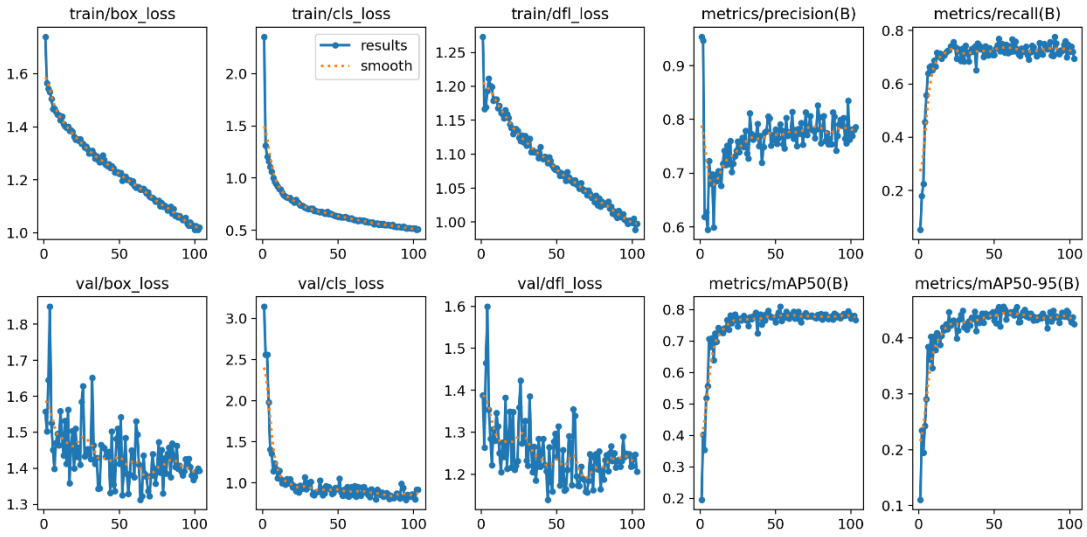


图 25 改进后的计算结果

（4）混淆矩阵

混淆矩阵是评估分类模型准确度的基本工具。它能够清晰地展示模型分类正确与 错误的观测值比例。图 26 展示改进后模型的混淆矩阵情况。

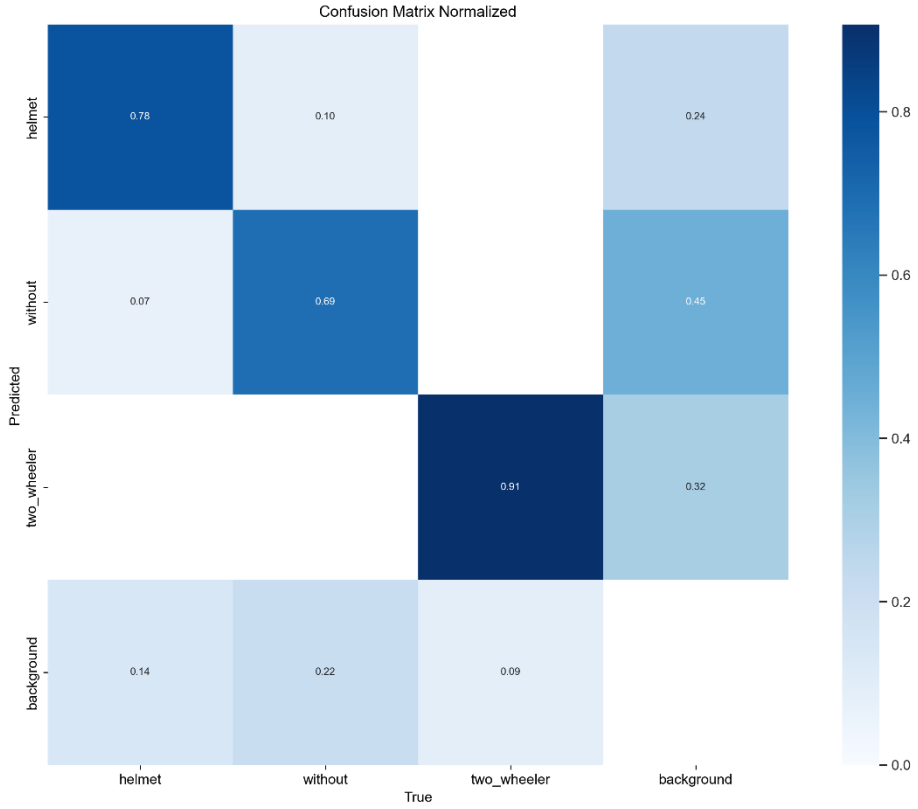


图 26 改进后的混淆矩阵

3.4 算法检测结果分析与展示

为了进一步评估模型性能，在考虑检测指标的同时，也应进行对检测结果的综合 探讨。本节在不同目标大小、不同场景角度、不同车型以及多人同车等复杂情况下展 开实验，获得相应的检测结果，来进一步衡量改进后的 YOLOv8n 模型的综合性能。同 时，我们也将展示并分析部分具体的检测结果，以便更直观地理解模型在各种实际场 景中的表现。

3.4.1 单目标和多目标检测效果

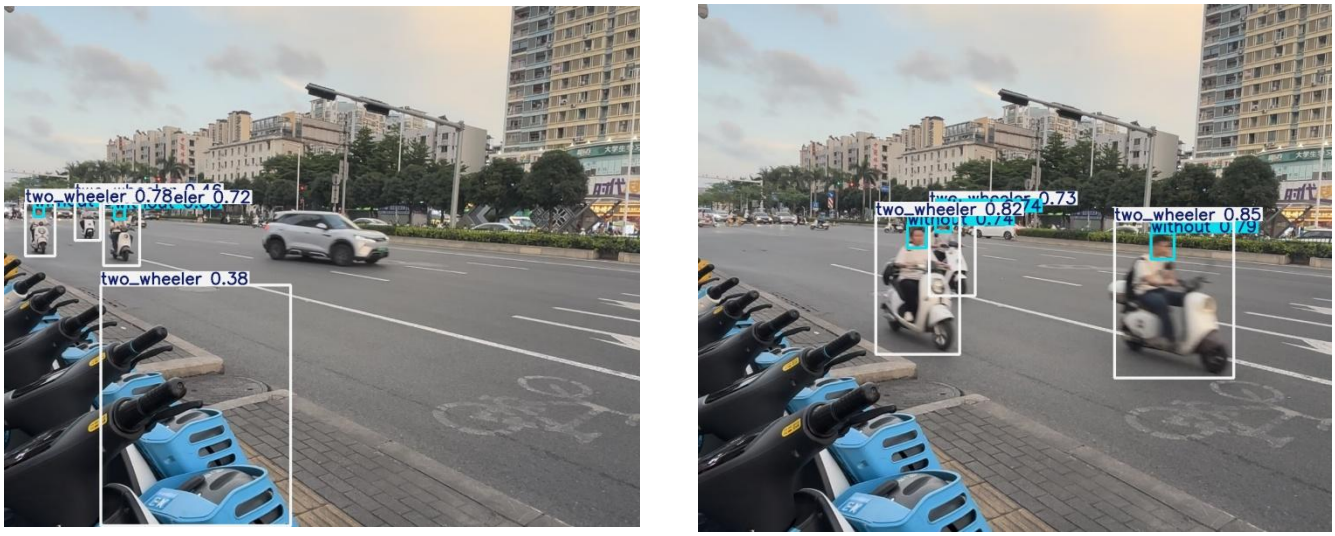
对于车流量较小的交通场所，如图 26 所示，算法能够准确的识别出电瓶车以及 车上人员是否佩戴安全头盔，具备基本的实际应用价值。

（a）单目标检测　　（b）多目标检测

图 26 单目标和多目标检测效果

3.4.2 大目标和小目标检测效果

随着经济的发展以及电瓶车销量的剧增，在实际生活中经常是交通拥挤的状况， 电瓶车和驾乘人员也是成群出现，除了近距离的车辆之外，远距离车辆也需要进行实 时检测。不同拍摄距离的检测结果如图 27 所示，面对远距离的小目标时，改进后的 YOLOv8n 模型也能够较为准确的识别出电瓶车。

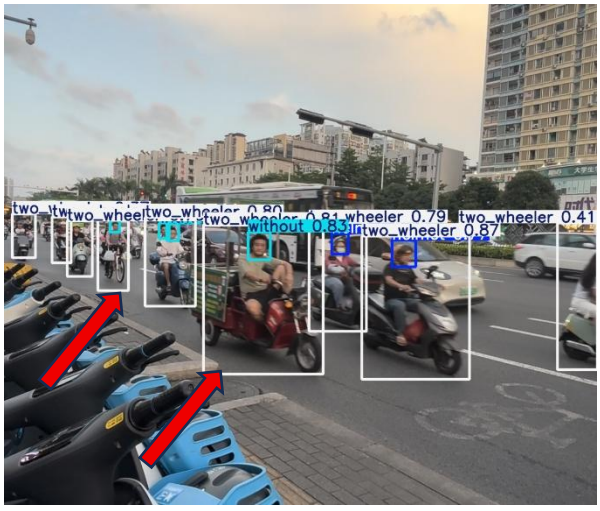


（a）远距离小目标检测　　（b）近距离大目标检测

图 27 近距离和远距离检测效果

3.4.3 对干扰目标的检测效果（缺陷）

为了增强模型的辨识能力并降低误判率，数据集中特意包含了自行车和三轮车辆 的图片作为干扰项。如图 28 所示，改进后的 YOLOv8n 模型还不太能能够准确识别出这些干扰 项的存在哈哈哈。

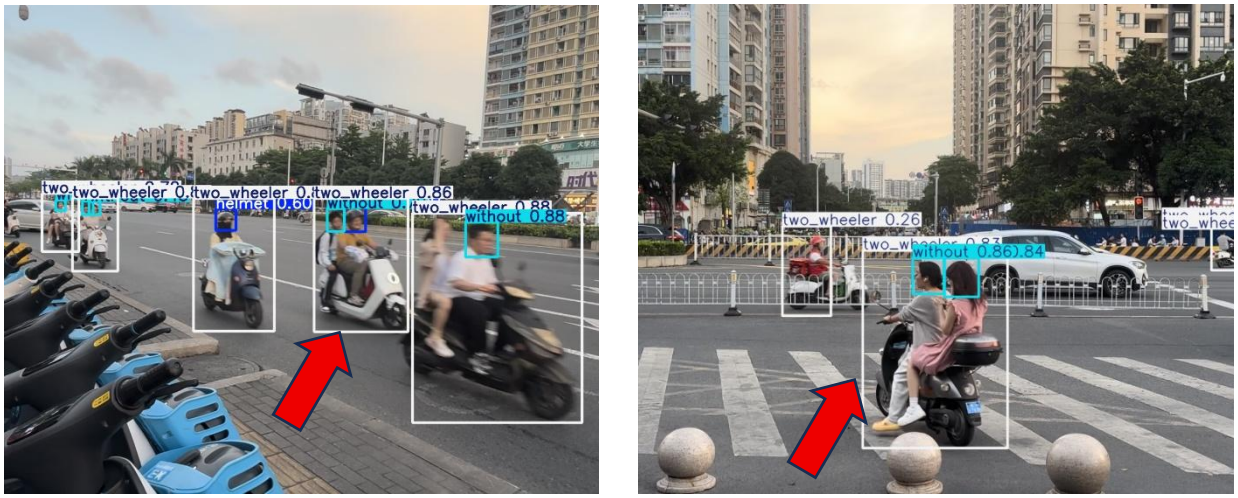


(a) 三轮车干扰　　(b) 自行车干扰

图 28 存在干扰目标

4.4.4 多人同车情况的检测效果

由于现在很多电瓶车空间设计较大，而且在地方交规中也是允许电瓶车载人，所 以经常出现多人同车的情况，而乘坐人员也存在交通隐患，需要对自己人身安全负责， 故也应该佩戴，这种情况也是我们的检测任务之一。如图 29 所示，改进后的 YOLOv8n 模型能够准确识别同一辆车上的不同人员的头盔佩戴情况，这为南宁交通安全提供更为有力的技术支撑。



（a）其中一人未佩戴 **（b）两人均未佩戴头盔**

图 29 多人同车的检测效果

4 课程总结

通过《机器视觉与目标检测》课程的系统学习与“基于 YOLOv8 的电动车头盔检测”项目实践，我深入掌握了目标检测技术的核心理论与工程实现方法。课程中，我系统梳理了 YOLO 系列、Faster R-CNN 等主流算法的演进逻辑与适用场景，并通过数据集构建、标注及增强（几何变换、色彩扰动等）的全流程实践，熟练运用 LabelImg 工具与均衡采样策略，解决了数据类别不均衡问题。在模型优化方面，通过引入 ASAM 注意力模块与多尺度特征融合策略，改进后的 YOLOv8n 模型在部分自制数据集上实现了 80.81%的精准率与 68.35%的 mAP(0.5)，较基线模型有提升，验证了算法对复杂场景下小目标及遮挡目标的检测优势。项目成果可应用于交通监控场景，辅助“一盔一带”政策的自动化执法。过程中，针对数据质量不足、小目标检测困难等挑战，我通过随机旋转、镜像翻转增强数据多样性，优化特征融合策略强化细节提取，并结合过采样技术平衡类别分布。然而，当前模型在推理速度（RTX 4060 Ti 单帧耗时增加 15%）与干扰项上场景的泛化能力上仍需优化。未来，我将探索模型轻量化（如知识蒸馏、网络剪枝）与多模态融合（红外/深度信息）技术，进一步提升实时性与鲁棒性，并扩展至工地安全帽、口罩检测等场景，推动 AI 技术在社会安全领域的普惠应用。

5 致谢

衷心感谢孙翠敏老师在教学中的辛勤付出！课堂上，您用生动的讲解和清晰的案例分析，让我扎实掌握了目标检测的基础理论和实践方法。您对技术细节的严谨态度和“解决实际问题”的核心理念，让我在学习中始终关注技术的实际价值。

参考文献

[1] 智勰.2024 年中国自行车电动自行车行业十件大事[J].中国自行车,2025,(01):8-13.

[2]道路交通安全专栏导语[J].中国公路学报,2025,38(03):2.

[3] 佚名.公安部交通管理局部署开展“一盔一带”安全守护行动 [J].道路交通管理，2025（5）：9.

[4] WIECZOREK M, SILKA J, WOŹNIAK M, et al. Lightweight convolutional neural network model for human face detection in risk situations[J]. IEEE Transactions on Industrial Informatics,2021,18(7):4820-4829.

[5] SIMBOLON H F S, SIRAIT P, et al.The effect of bicubic interpolation on Viola-Jones and principal component analysis in detecting faces and helmet wearers[C]//2020 3rd International Conference on Mechanical, Electronics, Computer, and Industrial Technology (MECnIT).IEEE，2020：158-164.

[6] GE P,CHEN Y. An automatic detection approach for wearing safety helmets on construction site based on YOLOv5[C]//2022 IEEE 11th Data Driven Control and Learning Systems Conference(DDCLS). IEEE,2022:140-145.

[7] CHENG R,HE X,ZHENG Z,et al. Multi-scale safety helmet detection based on SAS-YOLOv3-tiny[J]. Applied Sciences,2021,11(8):3652.

[8] YAN G,SUN Q,HUANG J,et al. Helmet detection based on deep learning and random forest on uav for power construction safety[J]. Journal of Advanced Computational Intelligence and Intelligent Informatics,2021,25(1):40-49.

[9] 刘备战，赵洪辉，周李兵.面向无人驾驶的井下行人检测方法[J].工矿自动

化，2021，47（9）：113-117.
[10] 赵心驰，胡岸明，何为.基于卷积神经网络和 XGBoost 的摔倒检测[J].激光与光电子学进展，2020，57（16）：248-256.
[11] 王新，冯艺楠.基于改进 SSD 的骑行人员佩戴头盔检测[J].电子测量技术，2022，45（21）：90-97.
[12] Purkait P，Zhao C，Zach C. SPP-Net：Deep absolute pose regression with synthetic views [J/OL]．（2017-12-09）[2025- 03-01]． https：//doi.org/10.48550/arXiv.1712.03452.
[13] Gkioxari G，Hariharan B，Girshick R，et al. R-cnns for pose estimation and action detection [J/OL]．（2014-06-19）[2025- 03-01]．https：//doi.org/10.48550/arXiv.1406.5212.
[14] Ren S，He K，Girshick R，et al. Faster r-cnn：Towards realtime object detection with region proposal networks [J]． IEEE Transactions on Pattern Analysis and Machine Intelligence， 2017，39（6）：1137-1149.
[15] Redmon J，Divvala S，Girshick R，et al. You only look once：Unified，real-time object detection [C] //In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition，2016：779-788.
[16] Liu W，Anguelov D，Erhan D，et al. SSD：Single shot multibox detector [C] //In Proceedings of the Computer VisionECCV 2016：14th European Conference，2016：21-37.
[17] Lin T Y，Goyal P，Girshick R，et al. Focal loss for dense object detection [C] //In Proceedings of the IEEE International Conference On Computer Vision，2017：2980-2988.
[18] Law H， Deng J. Cornernet： Detecting objects as paired keypoints [C] //In Proceedings of the European Conference on Computer Vision（ECCV），2018：734-750.
[19] Zhang S，Wen L，Bian X，et al. Single-shot refinement neural network for object detection [C] //In Proceedings of the 2018 IEEE Conference on Computer Vision and Pattern Recognition，2018：4203-4212.
[20] 如下图

IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

A Self-Supervised Monocular Depth Estimation Framework Based on Detail Recovery and Feature Fusion

SHUN LI[1,*], CHONGZHENG HUANG[2,*], XIANGZHE LI[1], and ZHENGYOU LIANG[1,3]
[1]School of Computer, Electronics and Information, Guangxi University, Nanning 530004, China
[2]School of Information Engineering, Guangxi Polytechnic of Construction, Nanning 530007, China
[3]Guangxi Key Laboratory of Multimedia Communications and Network Technology, Nanning 530004, China
*These authors contributed to the work equally and should be regarded as co-first authors
Corresponding author: Zhengyou Liang (zhyliang@gxu.edu.cn)
This work was supported by the National Nature Science Foundation of China under Grant 62171145.

（小修预计 6 月收录）

相关附录代码

图像增强 1

图片水平镜像：

```
# -*- coding: utf-8 -*-
# 方法一：使用 PIL/Pillow 库
from PIL import Image
def flip_image_pil(image_path, output_path):
    """
    使用 PIL 库进行水平镜像反转
    """
    # 打开图片
    image = Image.open(image_path)
    # 水平镜像反转
    flipped_image = image.transpose(Image.FLIP_LEFT_RIGHT)
    # 保存反转后的图片
    flipped_image.save(output_path)
    print(f"图片已保存到: {output_path}")
if __name__ == "__main__":
    input_path = "C:\\Users\\25311\Desktop\YOLOv8helmet\datasets\广西大学南门口数据集\图片\\5.png"  # 输入图片路径
    output_path = "C:\\Users\\25311\Desktop\YOLOv8helmet\datasets\广西大学南门口数据集\图片\\5flipped_image.png"  # 输出图片路径

    # 选择其中一种方法使用
    # 方法一：PIL (推荐，简单易用)
    flip_image_pil(input_path, output_path)
```

图像增强 2

图片随机裁剪：

```
# -*- coding: utf-8 -*-
import random
import os
from PIL import Image
import cv2
import numpy as np
# Method 3: Random crop with scale factor (percentage of original size)
def random_crop_scale(image_path, output_path, scale_range=(0.5, 0.9)):
    """
    Random crop with random scale factor
    Args:
        scale_range: tuple (min_scale, max_scale) as percentage of original size
    """
    image = Image.open(image_path)
    img_width, img_height = image.size
    # Random scale factor
    scale = random.uniform(scale_range[0], scale_range[1])
    crop_width = int(img_width * scale)
    crop_height = int(img_height * scale)
    # Generate random crop coordinates
    left = random.randint(0, img_width - crop_width)
    top = random.randint(0, img_height - crop_height)
    right = left + crop_width
    bottom = top + crop_height
    # Crop the image
    cropped_image = image.crop((left, top, right, bottom))
    cropped_image.save(output_path)
    print(f"Random scaled crop saved to: {output_path}")
    print(f"Scale factor: {scale:.2f}, Crop size: ({crop_width}, {crop_height})")
    return True

if __name__ == "__main__":
    input_image = "C:\\Users\\25311\Desktop\YOLOv8helmet\datasets\广西大学南门口数据集\图片\\6.png"
    random_crop_scale(input_image, "C:\\Users\\25311\Desktop\YOLOv8helmet\datasets\广西大学南门口数据集\图片\\6cropped_scale.jpg", (0.5, 0.8))
```

图像增强 3

图片随机旋转：

```
# -*- coding: utf-8 -*-
import random
import os
import math
from PIL import Image, ImageFilter
import cv2
import numpy as np
import os
os.environ["OPENCV_IO_ENABLE_UTF8"] = "1"

def random_rotate_opencv(image_path, output_path=None, angle_range=(-180, 180)):
    """
    使用 OpenCV 实现图像随机旋转
    Args:
        image_path: 输入图像路径
        output_path: 输出图像路径（可选）
        angle_range: 旋转角度范围，默认(-180, 180)
    Returns:
        rotated_image: 旋转后的图像
    """
    # 读取图像
    image = cv2.imread(image_path)
    if image is None:
        raise ValueError(f"无法读取图像: {image_path}")
    # 生成随机旋转角度
    angle = random.uniform(angle_range[0], angle_range[1])
    # 获取图像尺寸
    height, width = image.shape[:2]
    center = (width // 2, height // 2)
    # 计算旋转矩阵
    rotation_matrix = cv2.getRotationMatrix2D(center, angle, 1.0)
    # 计算旋转后的图像边界
    cos_val = abs(rotation_matrix[0, 0])
    sin_val = abs(rotation_matrix[0, 1])
    new_width = int((height * sin_val) + (width * cos_val))
    new_height = int((height * cos_val) + (width * sin_val))
    # 调整旋转矩阵以避免裁剪
    rotation_matrix[0, 2] += (new_width / 2) - center[0]
    rotation_matrix[1, 2] += (new_height / 2) - center[1]
    # 执行旋转
    rotated_image = cv2.warpAffine(image, rotation_matrix, (new_width, new_height))
```

```
    # 保存图像
    if output_path:
        cv2.imwrite(output_path, rotated_image)
        print(f"旋转角度: {angle:.2f}°")
        print(f"图像已保存到: {output_path}")
    return rotated_image, angle
# 使用示例
if __name__ == "__main__":
    # 设置随机种子（可选，用于复现结果）
    random.seed(42)
    # 示例 1: 使用 OpenCV 旋转单张图像
    try:
        image_path = r"C:\Users\25311\Desktop\YOLOv8helmet\datasets\7.png"  # 替换为你的图像路径
        output_path = "C:\\Users\\25311\Desktop\YOLOv8helmet\datasets\\7rotated_opencv.png"
        rotated_img, angle = random_rotate_opencv(image_path, output_path, (-45, 45))
        print("OpenCV 旋转完成")
    except Exception as e:
        print(f"OpenCV 旋转失败: {e}")
```

图像增强 4

图片随机旋转：

```
# -*- coding: utf-8 -*-
"""
灰度变化处理示例：
1. 把任意 RGB 图像转换为灰度图；
2. 随机调整灰度图的亮度与对比度，得到“灰度扰动”版本；
3. 保存并对比显示三张图：原图（彩色）、标准灰度、扰动后灰度。
"""
from PIL import Image, ImageOps, ImageEnhance
import numpy as np
import matplotlib.pyplot as plt
# === ① 读取原图 ===
img_path = 'C:\\Users\\25311\Desktop\YOLOv8helmet\datasets\广西大学南门口数据集\图片\8.png'          # ——>> 换成你的图片文件或绝对路径
img = Image.open(img_path).convert('RGB')
# === ② 转为灰度 ===
gray = ImageOps.grayscale(img)   # 仍然返回 PIL.Image，只是 mode='L'
# === ③ 自定义灰度扰动函数 ===
def grayscale_perturb(
    image: Image.Image,
    brightness_range=(0.8, 1.2),  # 整体亮度随机缩放系数
```

```
    contrast_range=(0.8, 1.2),      # 对比度随机缩放系数
    seed=None                       # 固定随机种子可复现
) -> Image.Image:
    if seed is not None:
        np.random.seed(seed)
    # 随机亮度
    b_factor = np.random.uniform(*brightness_range)
    image_b = ImageEnhance.Brightness(image).enhance(b_factor)
    # 随机对比度
    c_factor = np.random.uniform(*contrast_range)
    image_bc = ImageEnhance.Contrast(image_b).enhance(c_factor)
    return image_bc
# === ④ 生成扰动灰度图 ===
gray_perturbed = grayscale_perturb(gray)        # 可改 seed=123 复现
# === ⑤ 保存结果 ===
out_path = 'C:\\Users\\25311\Desktop\YOLOv8helmet\datasets\广西大学南门口数据集\图片\8_gray_perturbed.png'
gray_perturbed.save(out_path)
# === ⑥ 对比显示 ===
titles = ['Original (RGB)', 'Grayscale', 'Grayscale Perturbed']
images = [img, gray, gray_perturbed]
for i, im in enumerate(images):
    plt.figure(figsize=(4, 4))
    plt.imshow(im if i == 0 else im, cmap=None if i == 0 else 'gray')
    plt.axis('off')
    plt.title(titles[i])
    plt.tight_layout()
    plt.show()
print(f'扰动后灰度图已保存为: {out_path}')
```

图像尺寸修正

图片修正 640*640：

```
# -*- coding: utf-8 -*-
from PIL import Image
# 加载图片
image_path = "C:\\Users\\25311\Desktop\YOLOv8helmet\datasets\广西大学南门口数据集\图片\\3.png"
image = Image.open(image_path)
# 设定裁剪尺寸
crop_size = 640
width, height = image.size
# 计算中心裁剪区域的坐标
```

```
left = (width - crop_size) / 2
top = (height - crop_size) / 2
right = (width + crop_size) / 2
bottom = (height + crop_size) / 2
# 裁剪图片
cropped_image = image.crop((left, top, right, bottom))
# 保存裁剪后的图片
cropped_image.save("C:\\Users\\25311\Desktop\YOLOv8helmet\datasets\广西大学南门口数据集\图片\\3_640_640.png")
```