

学号: 2021013149



西北农林科技大学

2025 届本科生毕业论文

摩托车驾乘人员头盔佩戴检测系统设计与实现

学院: 信息工程学院

专业: 计算机科学与技术

年级班级: 2021 级 4 班

学生姓名: 姜明宇

指导教师: 耿耀君

协助指导教师: \_\_\_\_\_

完成日期: 2025 年 5 月



## 本科生毕业论文的独创性声明

本人声明：所呈交的毕业论文是我个人在导师指导下独立进行的研究工作及取得的研究结果。尽我所知，除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含其他人和自己本人已获得西北农林科技大学或其它教育机构的学位或证书而使用过的材料。与我一同工作的同事对本研究所做的任何贡献均已在论文的致谢中作了明确的说明并表示了谢意。如违反此声明，一切后果与法律责任均由本人承担。

本科生签名： 时间： 年 月 日

## 关于本科生毕业论文知识产权的说明

本毕业论文的知识产权归属西北农林科技大学。本人同意西北农林科技大学保存或向国家有关部门或机构递交论文的纸质版和电子版，允许论文被查阅和借阅。

本人保证，在毕业离开西北农林科技大学后，发表或者使用本毕业论文及其相关的工作成果时，将以西北农林科技大学为第一署名单位，否则，愿意按《中华人民共和国著作权法》等有关规定接受处理并承担法律责任。

任何收存和保管本论文各种版本的其他单位和个人（包括作者本人）未经本论文作者的导师同意，不得有对本论文进行复制、修改、发行、出租、改编等侵犯著作权的行为，否则，按违背《中华人民共和国著作权法》等有关规定处理并追究法律责任。

本科生签名： 时间： 年 月 日

指导教师签名： 时间： 年 月 日



# 摩托车驾乘人员头盔佩戴检测系统设计与实现

**摘要：**随着经济的发展与出行需求的增长，摩托车凭借其便捷性，保有量不断增加。但由于交通管理难度增大、部分驾驶员安全意识不足等因素，涉及摩托车的交通事故也逐渐增多。头盔佩戴与否直接关系到驾乘人员在事故中的伤亡程度。本文旨在设计并实现一套摩托车驾乘人员头盔佩戴检测系统，实时检测驾乘人员的头盔佩戴情况，并将检测结果以数据可视化的形式呈现。本文的主要工作涵盖检测模型训练和系统开发两个重要部分。

检测模型训练方面，基于 YOLO 目标识别算法，构建了双模型协同架构：其中，第一个模型聚焦于快速、准确地识别图像或视频中摩托车及其驾乘人员整体的头盔佩戴状态；第二个模型则以前者输出的关键区域作为输入，精准定位并识别驾驶人员的 id。在研究过程中，对 YOLOv11n、YOLOv11s 和 YOLOv11m 三个不同规模的模型的训练效果进行了全面对比分析，并总结了各个模型所适用的检测场景。

系统开发方面，基于浏览器/服务器 (B/S) 架构开发前后端分离的系统，使用 Vue 框架搭建 B 端页面，SpringBoot 框架搭建 S 端服务器。前端浏览器为用户提供了头盔佩戴检测和历史记录查询两大核心功能。头盔佩戴检测功能允许用户根据自身需求自定义检测模型、检测交并比 (IoU) 和置信度，同时支持对检测结果的浏览和保存操作。历史记录查询功能则支持用户根据特定条件查询检测记录，并通过可视化图表的形式辅助用户进行数据分析。后端服务器主要分为目标检测和记录查询两个模块，分别与前端的两大功能进行对接。目标检测模块调用模型对用户上传的图片或视频进行检测，并存储检测结果；记录查询模块根据用户输入的过滤条件，对数据库中的历史记录进行查询并返回结果。

**关键词：**目标检测; YOLO; 摩托车头盔; 系统开发; 数据可视化



# **Design and implementation of helmet wearing detection system for motorcycle riders**

**Abstract:** As the economy develops and travel demand grows, the number of motorcycles rises due to their convenience. Yet, traffic accidents involving motorcycles have increased, driven by tougher traffic management and insufficient safety awareness among some riders. Helmet use significantly impacts accident outcomes for riders. This paper designs and implements a real - time helmet - wearing detection system for motorcycle riders, visualizing results. Its key tasks are detection model training and system development.

In terms of model training, a dual model collaborative architecture was constructed based on the YOLO object recognition algorithm. The first model focuses on quickly and accurately identifying the overall helmet wearing status of motorcycles and their occupants in images or videos; The second model uses the key areas output by the previous model as input to accurately locate and recognize the driver's ID. During the research process, a comprehensive comparative analysis was conducted on the training effects of three different scale models, YOLOv11n, YOLOv11s, and YOLOv11m, and the detection scenarios applicable to each model were summarized.

In terms of system development, a front-end and back-end separated system is developed based on the browser/server (B/S) architecture, using Vue framework to build B-end pages and SpringBoot framework to build S-end servers. The front-end offers two core functions: helmet wearing detection, allowing users to customize model, IoU, and confidence, and view/save results; and historical record query, enabling searches by specific conditions and using visual charts for analysis. The backend has two modules: the object detection module calls the model for user-uploaded media detection and stores results, while the record query module fetches historical data from the database based on user filters and returns them.

**Keywords:** Object Detection; YOLO; Motorcycle Helmet; System Development; Data Visualization



# 目 录

<b>第 1 章 绪论 .....</b>	<b>1</b>
1.1 研究目的与意义 .....	1
1.2 目标检测发展历程 .....	2
1.2.1 传统目标检测 .....	2
1.2.2 基于深度学习的目标检测 .....	3
1.2.3 YOLO 在交通安全领域的应用 .....	5
1.3 研究与设计内容 .....	6
1.4 章节安排 .....	6
<b>第 2 章 YOLO 算法理论及数据集预处理 .....</b>	<b>8</b>
2.1 YOLOv11 网络结构 .....	8
2.1.1 骨干网络 .....	8
2.1.2 颈部网络 .....	10
2.1.3 检测头 .....	11
2.2 YOLOv11 损失函数 .....	11
2.2.1 边界框回归损失 .....	11
2.2.2 分类损失 .....	12
2.2.3 分布损失 .....	12
2.3 数据集预处理 .....	12
2.4 实验环境 .....	14
2.5 参数设置 .....	15
2.6 本章小结 .....	16
<b>第 3 章 实验结果与分析 .....</b>	<b>17</b>
3.1 评价指标 .....	17
3.1.1 精度评价指标 .....	17
3.1.2 速度评价指标 .....	17
3.2 实验结果分析 .....	18
3.2.1 头盔佩戴情况模型 .....	18
3.2.2 驾驶员 id 模型 .....	23
3.3 本章小结 .....	27

<b>第 4 章 检测系统的设计与实现</b>	28
4.1 需求分析	28
4.2 系统整体架构	28
4.3 前端模块设计与实现	29
4.3.1 目标检测界面	29
4.3.2 记录查询界面	31
4.4 后端模块设计与实现	31
4.4.1 库表结构	32
4.4.2 检测模块	33
4.4.3 搜索模块	35
4.5 本章小结	35
<b>第 5 章 总结与展望</b>	37
5.1 本文工作总结	37
5.2 研究展望	37
<b>参考文献</b>	39
<b>致 谢</b>	41

## 第1章 绪论

### 1.1 研究目的与意义

当下，快节奏的生活方式催生出外卖配送、即时出行等需求，摩托车凭借其小巧灵活、通行便利的特性，在全球各地的城市交通体系中占据了愈发重要的地位。无论是穿梭于大街小巷的外卖骑手，还是追求通勤效率的上班族，都将摩托车视为短途出行的首要选择。

市场需求的增长，有力推动了摩托车产业的蓬勃发展，其保有量在全国范围内持续攀升。但由于我国交通体系复杂、管理难度大以及驾驶人员安全意识淡薄，涉及摩托车的交通事故逐渐增多。头盔作为摩托车驾驶员和乘客最关键的防护装置，能有效缓冲撞击、减轻伤害，尽可能地保护驾乘人员的生命安全(侯帅帅和欧秀丽 2023)。尽管国家出台了强制佩戴头盔的交通法规来保障骑行者的生命安全，但部分骑行者安全意识淡薄，依旧心存侥幸，不佩戴头盔就上路行驶，如图 1-1。

当前，针对摩托车驾乘人员头盔佩戴情况的监管，主要依赖交警人工检查。然而，我国道路系统错综复杂，交通流量庞大且情况瞬息万变，交警在维持交通秩序的同时，还要负责检查头盔佩戴情况，工作负担极为沉重。人工检查不仅效率低下、耗费大量人力物力，而且在复杂路况和密集车流中，极易出现漏检现象，难以确保监管工作的全面性和准确性。因此，一个摩托车驾乘人员头盔佩戴检测系统对减少人工工作量、提升检测速度和准确度有很大的意义。



图 1-1 城市道路上摩托车驾乘人员头盔佩戴情况

## 1.2 目标检测发展历程

图 1-2(图像来源: <https://www.frontiersin.org/journals/built-environment/articles/10.3389/fbuil.2025.1563483/full>) 展示了目标检测的发展。

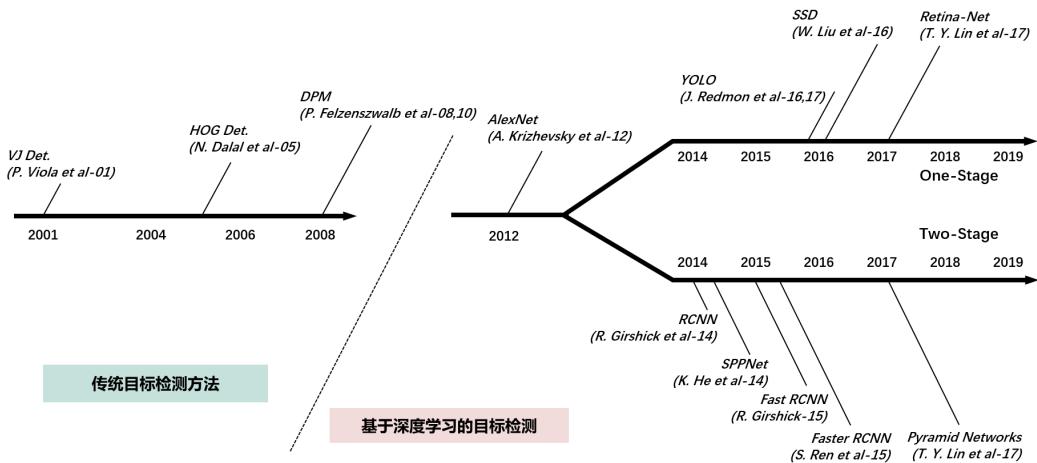


图 1-2 目标检测算法发展历程

### 1.2.1 传统目标检测

在深度学习兴起之前, 受限于图像表示能力, 研究者们需要设计复杂的特征表示。这一时期的代表成果深刻影响了后续目标检测技术的发展。

2001 年, [Viola and Jones \(2001\)](#)首次在通用场景下实现了无约束的人脸实时检测。以两位作者命名的 VJ 检测器在检测精度没有损失的情况下, 计算效率提高了 1-2 个数量级。该检测器运用滑动窗口, 对图像内所有尺寸所有位置的窗口进行遍历, 判别窗口中是否存在人脸。

[Dalal 与 Triggs 于 2005 年提出方向梯度直方图 \(HOG\) 描述器, 对当时的尺度不变特征变换和形状语境做出重要改进\(Dalal and Triggs 2005\)](#)。HOG 描述器最初针对行人检测任务而设计, 以均匀分布的单元网格作为计算基础, 引入重叠局部对比度归一化, 显著提升了特征表征的准确性。该描述器应用于多尺度目标检测时, 需保持检测窗口尺寸恒定, 对输入图像实施多尺度变换以实现不同尺寸目标的识别。

DPM(Deformable Part Models) 在目标检测领域具有重要地位, 该模型最早由 P. Felzenszwalb 在 2008 年提出([Felzenszwalb et al. 2008](#)), 是 HOG 检测器的优化版本, 后续经 R. Girshick 等人不断优化改进([Felzenszwalb et al. 2010](#))。

## 1.2.2 基于深度学习的目标检测

2012年，Krizhevsky et al. (2012)提出了一种经典的卷积神经网络 AlexNet，它的出现极大地推动了深度学习领域的技术革新。目标检测算法依据其架构与运行机制，主要可划分为一阶段算法与两阶段算法两大类别。一阶段检测方法通过回归策略直接在输入图像上预测目标类别和空间位置，省去了候选框生成过程，提升了检测效率，使其适用于自动驾驶系统、智能视频监控等对实时性要求严苛的应用场景，典型代表包括 YOLO 系列和 SSD 等。两阶段检测方法首先生成潜在目标区域，随后对这些候选区域执行细粒度的分类和边界框回归，虽然这种分阶段处理机制能够获得更精确的检测结果，但不可避免地会使得目标检测的速度变慢。下面将对两阶段的代表算法 R-CNN 系列展开介绍，然后详细介绍本文使用到的一阶段的代表算法 YOLO 系列。

### (1) R-CNN 系列算法

2014年，Girshick et al. (2014)提出的 R-CNN 的思路为：基于选择性搜索算法(Uijlings et al. 2013)生成可能存在目标的潜在区域，并统一各区域的尺度。采用 AlexNet 网络架构对标准化后的候选区域进行深度特征提取，最后运用 SVM 对各个候选区域进行目标分类，实现对不同目标的准确识别与判定。该模型需对数量众多且相互重叠的候选区域(单张图像通常生成超过 2000 个候选框)进行特征提取计算，检测速度极慢。同年，He et al. (2014)通过引入空间金字塔池化层 (SPP)，突破了传统 CNN 需要固定输入尺寸的约束，能够输入任意尺寸图像并输出固定长度特征表示，将检测速度提升至 R-CNN 的 20 倍以上。

2015 年，Girshick (2015)提出了 Fast R-CNN 检测算法。该模型通过创新性地将检测器与边界框回归器集成于统一网络架构中进行联合训练，不仅显著缩短了模型训练和推理过程的耗时，同时使检测性能获得显著提升。

根据Ren et al. (2015)的研究成果，Faster R-CNN 检测器作为首个实现端到端训练的深度学习检测框架，在检测效率方面取得了突破性进展，首次达到了接近实时的检测速度。该模型通过创新性地引入 RPN 机制，显著提升了目标检测的运算效率与整体性能表现。RPN 用于自动生成候选区域，在性能上要比选择性搜索算法好很多，推动目标检测系统从分散模块逐步整合为统一的端到端学习架构。

### (2) YOLO 系列算法

2016 年，Redmon et al. (2016)提出了 YOLO 算法，为目标检测任务提供了一种新的解决思路。其核心机制是利用单个卷积神经网络对整幅图像进行处理，将图像分割为多个区域后，直接预测各区域的边界框与对应类别概率，并通过概率加权对边界框进行筛

选，经阈值处理后输出高置信度检测结果。YOLO 算法显著提升了检测速度，但仍存在局限性：一是泛化能力弱，难以准确检测训练时未见过的新物体；二是空间约束大，每个网格单元仅预测两个框，不利于处理小目标群；三是定位误差大，经常误判物体位置。

YOLOv2 是在 YOLOv1 发布一年后推出的(Redmon and Farhadi 2017)，无论是分类还是检测，YOLOv2 都比其前身 YOLOv1 有了很大的改进。YOLOv2 的一大突破性进展在于开发了一种新的联合训练策略。该算法创造性地将检测数据集与分类数据集整合，使目标检测模型能够同时从两种类型的数据中获取信息，训练时同时提升检测精度与分类能力。通过标记检测图像学习目标精确定位，借助分类图像扩展模型识别类别范围、增强鲁棒性。

YOLOv3(Redmon and Farhadi 2018) 基于 YOLOv2 进行改进与创新：一方面，YOLOv3 修正了 YOLOv2 的数据加载缺陷，使模型平均精度均值 (mAP) 提升了两个点；另一方面，YOLOv3 采用 Darknet-53 骨干网络，引入多尺度预测架构 (三尺度特征金字塔) 优化跨尺寸目标检测能力。

相较于 YOLOv3，YOLOv4(Bochkovskiy et al. 2020) 在骨干网络和颈部网络两个关键环节进行了创新升级：骨干网络层面，摒弃原有架构，采用 CSPDarknet53 作为新的骨干网络，通过跨阶段部分网络 (CSPNet) 策略优化特征提取；颈部结构上，改进空间金字塔池化 (SPP) 与路径聚合网络 (PAN) 的引入，进一步强化多尺度特征融合能力。

YOLOv5 较 YOLOv4 在易用性上实现了提升。除了性能上面一些微小的提升之外，YOLOv5 基于 PyTorch 框架重构代码，简化了模型的部署，同时提供了更详尽的多语言文档支持，降低开发门槛。

YOLOv6 是 YOLO 系列中的一次重大演变，由美团视觉团队开发(Li et al. 2022)。YOLOv6 对 YOLOv4 和 YOLOv5 里的 PAN 拓扑结构进行了强化，借助 RepBlocks 与 CSPStackRep Blocks，更高效地从骨干网络的不同层级聚合特征。除此之外的创新还有：通过硬件感知架构设计与动态训练策略的协同创新，在速度精度平衡与部署效率层面实现突破性进展；核心架构采用 EfficientRep 主干网络，基于 RepVGG 重参数化思想构建分层模块化结构，显著提升 GPU 推理效率；特征融合模块重构为 Rep-PAN 拓扑，通过重参数化卷积增强跨尺度信息流，并结合解耦式预测头缩减冗余计算。

YOLOv7 基于 YOLOv6 提出了一系列细粒度的改进(Wang et al. 2022)。提出计划的重新参数化模型，将梯度传播路径概念应用于不同网络层；针对多输出层模型训练，引入标签分配的新方法；创新性地构建了“扩展”与“复合缩放”两种新型策略，通过优化参数配置与计算资源分配机制，显著提升了目标检测模型的性能表现。这些改进和优化策略，在不牺牲速度的情况下显著提高了准确率，是 YOLO 系列的重要进步。

YOLOv8 在检测精度和速度方面实现了更优的性能，延续了 YOLOv5 用户友好的特

点，进一步增强了易用性。YOLOv8 采用无锚分割 Ultralytics head，提升了检测的准确性与速度。YOLOv8 由 Ultralytics 维护，提供了针对检测、分割、分类和姿势检测等特定任务的多种专用模型。

相较于前代模型，2024 年提出的 YOLOv9 采用全新思路，解决了深度神经网络信息丢失的问题(Wang, Bochkovskiy, et al. 2024b)。YOLOv9 的核心创新在于两大关键技术：一方面，引入可编程梯度信息 (PGI)，通过辅助可逆分支完整保留输入信息，为目标函数计算提供充足依据，确保梯度更新更精准有效；另一方面，提出了 ELAN 架构的通用轻量化版本——GELAN，作为，基于梯度路径规划设计，最大化网络信息流，高效整合特征信息辅助预测。

同为 2024 年提出的 YOLOv10 相较于前代模型，刷新了速度与准确度上限，实现了真正的实时检测(Wang, Bochkovskiy, et al. 2024a)。YOLOv10 的核心创新在于：一是采用 NMS-Free 检测，基于双重标签分配（一对多和一对一）及一致匹配度量的训练策略，推理时仅用一对一 head，提升推理速度、简化部署、增强训练监督；二是运用整体效率-准确度驱动设计，通过轻量级分类 head、空间通道解耦下采样和等级引导块设计，在优化模型各组件的同时有效降低计算成本。

2024 年 9 月发布的 YOLOv11 历经一系列架构改良，聚焦于在无损检测准确性的前提下，全力提升计算效率。YOLOv11 创新性地引入了 C3k2 模块与 C2PSA 块等关键组件。C3k2 模块作为跨阶段部分 (CSP) 瓶颈的高效计算实现，取代了骨干网络和颈部网络中的 C2f 块。C2PSA 块紧跟 SPPF 模块之后，这种全新的注意力机制，使模型能够更为高效地聚焦于图像内的关键区域，精准识别目标物体。

### 1.2.3 YOLO 在交通安全领域的应用

YOLO 系列算法凭借出色的检测速度和准确性在交通安全领域得到了广泛的应用。在车辆检测、行人检测以及道路检测方面均表现出强大的应用潜力。

在车辆检测方面，张浩晨 等 (2025) 基于 YOLOv8 算法，提出了结合 Transformer 结构全局特征提取能力的模块 C2Former 代替 C2f 模块，提升了算法在目标被遮挡场景下对交通车辆检测的精度。Raza et al. (2024) 针对雾天场景下的车辆检测需求，引入注意力模块 (CBAM、NAM、SimAM) 和 BiFPN 结构优化 YOLO-V5s/V5l，并对比了 YOLO-V5/V8 系列模型的性能，优化了算法在雾天中对车辆的检测性能。

在行人检测方面，Wang, Yang, et al. (2024) 基于 YOLOv7 将 ELAN-SA 模块与 LGA 模块相结合，增强了特征提取能力，在遮挡和小目标行人检测方面表现出很强的性能。袁婷婷 等 (2025) 基于 YOLOv11，融合 RepConv 来改进 C3k2 模块，设计全新的颈部结构 MBFPN，提升特征提取与融合能力，提出了轻量化行人检测算法。

在道路检测方面,秦乐等(2025)提出基于YOLOv8n的轻量化改进算法EMF-YOLO,通过引入增强型特征融合金字塔EFFPN、可变形注意力机制和多尺度边缘敏感性增强模块MESA等,实现了模型的轻量化。

YOLO算法凭借其高效性与准确性,已在交通、工业、医疗等多个领域发挥其作用,提供了精准的实时监测能力,推动各个行业的智能化发展。

### 1.3 研究与设计内容

本文旨在设计并实现一套摩托车驾乘人员头盔佩戴检测系统,工作分为两个部分:模型训练和系统开发。

在模型训练阶段,主要开展两方面工作:一是优化数据标注,针对驾驶员及最多三名乘客(训练数据集中,最多只出现了一名驾驶员携带三名乘客)各自的头盔佩戴情况,细化设置了18个标签,相较于传统二分类标注,能提供更详尽的头盔佩戴信息;二是训练两个关键模型,分别用于识别摩托车驾乘人员的头盔佩戴状态,以及驾驶员id。训练过程中,采用YOLOv11n、YOLOv11s及YOLOv11m三种模型训练来比较效果,最后总结不同模型的优势与适用场景。

系统基于浏览器/服务器(B/S)架构开发。B端浏览器为用户提供了两个操作页面,一个是为了上传图像或视频以请求检测的页面,用户可通过该页面发起检测需求;另一个是数据查询页面,用户能从该页面向服务端数据库发送历史检测结果查询请求,还可以对驾驶人、记录时间、记录地点等字段进行过滤。查询得到的结果会以柱状图、折线图等可视化的形式在页面呈现,为后续制定执法策略提供数据支持。S端服务器负责处理浏览器传来的请求,当接收到用户上传的图像或视频后,先利用第一个模型预测图像中的头盔佩戴情况,之后对目标区域进行裁剪,再使用第二个模型检测目标区域的驾驶人员,并将检测结果保存到本地磁盘。**图1-3**和**图1-4**表示了输入图像和检测结果。



图 1-3 输入图像



图 1-4 检测结果

### 1.4 章节安排

本文共包含六个章节,每一章的主要内容如下:

第一章：绪论。本章首先介绍了本文的研究目的与意义，对目标检测的发展历程进行概述，详细介绍了 YOLO 系列算法的发展过程及其在交通安全领域的应用现状，最后说明了本文的研究内容。

第二章：YOLO 算法相关理论。本章深入阐述 YOLOv11 算法的核心构成，涵盖网络架构与损失函数两大关键模块。骨干网络通过 C3K2 和 C2PSA 两大创新点增强特征提取能力；颈部网络引入 C3K2 模块，并结合 C2PSA 模块的优点提升了特征融合能力；检测头以骨干网络和颈部网络生成的不同尺度的特征图像来生成预测框。损失函数主要介绍边界框回归损失函数、分类损失函数和分布损失函数。然后介绍了本文的数据集来源以及为解决类别不平衡对数据集做的增强处理，分析了增强之后的标签数量分布情况，然后介绍了实验所采用的硬件配置及软件环境，说明了模型训练过程中的具体参数设置。

第三章：实验结果与分析。本章首先介绍了本文使用到的目标检测模型检测精度和速度两方面的评价指标，展示了 YOLOv11n、YOLOv11s 和 YOLOv11m 这三个模型的训练结果，对比分析了上述三个模型的损失函数、精度、召回率、mAP 以及检测速度。并总结了不同模型各自适用的场景。

第四章：检测系统的设计与实现。本章主要介绍了摩托车驾乘人员头盔佩戴检测系统的设计与开发过程。从需求分析、系统架构、前端模块开发和后端模块开发这四个方面展开。

第五章：总结与展望。本章为本文的最后一张，对本文所做的工作进行总结，并展望目标检测技术在交通安全领域的未来的发展情况。

## 第 2 章 YOLO 算法理论及数据集预处理

### 2.1 YOLOv11 网络结构

YOLOv11 的网络架构采用骨干网络-颈部网络-检测头的分层设计。骨干网络作为特征提取的基石，引入 C3K2 模块、SPPF(快速空间金字塔池化) 和 C2PSA(具有注意力机制的卷积模块)组件，实现了高效的底层特征提取；颈部网络将 C2F 模块替换为了 C3K2 模块，提升了特征聚合过程的整体性能，通过 C2PSA 模块增强了对空间注意力机制的关注；检测头负责生成目标检测和分类的最终预测，它处理从颈部网络传递过来的特征图，最终输出图像中目标的边界框和类别标签。[图 2-1\(图像来源: <https://zhuanlan.zhihu.com/p/4251780251>\) 展示了 YOLOv11 的网络结构。](https://zhuanlan.zhihu.com/p/4251780251)

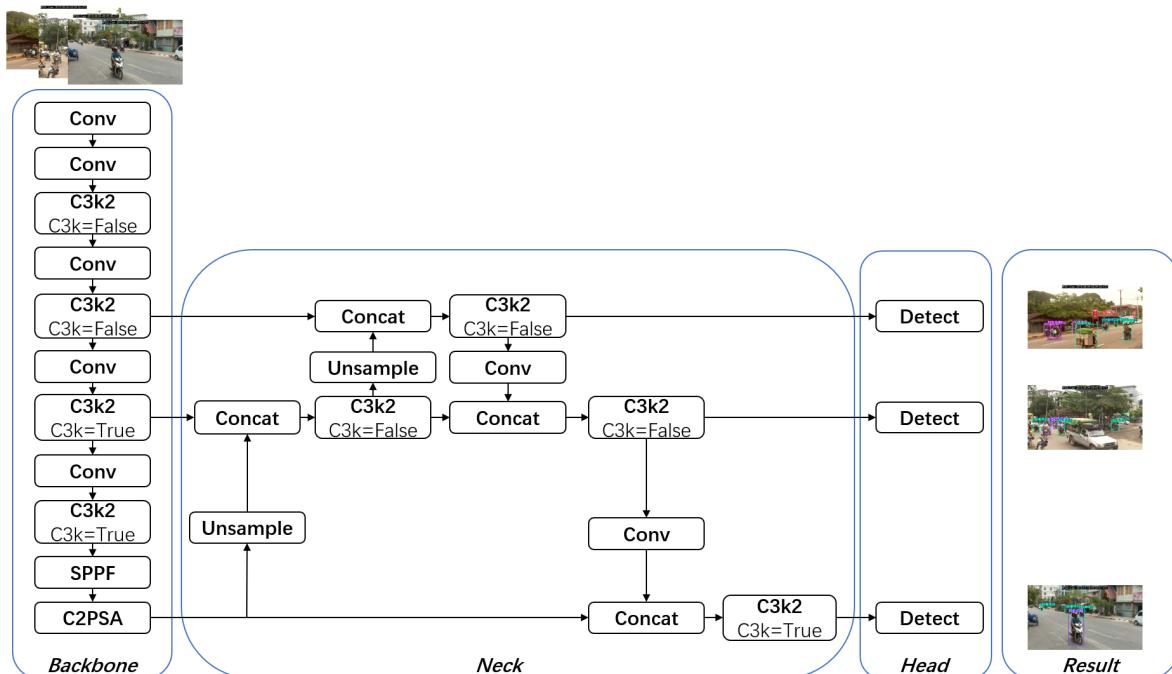


图 2-1 YOLOv11 网络模型

#### 2.1.1 骨干网络

YOLOv11 的骨干网络是整个架构的核心特征提取模块，包含 Conv、C3K2、SPPF 和 C2PSA 模块。

Conv 卷积模块的处理步骤可以概括为数据准备、参数配置、卷积运算和非线性激活四步。Conv 模块的输入通常是  $(B, C, W, H)$  四维张量， $B$  表示训练批次； $C$  表示数据通道数，彩色 RGB 图像的通道数为 3； $W$  表示图像的宽度， $H$  表示高度。卷积层的核心参数包括卷积核、步长 (stride)、填充 (padding)，卷积核决定特征提取的局部感知范围，

步长控制卷积核滑动间隔以实现下采样，填充通过边界补零调整输出尺寸以及输出通道数。卷积运算这一步将配置好的卷积核应用于输入数据，通过滑动窗口对每个局部区域进行加权求和，实现特征提取。非线性激活应用 ReLU、silu 等激活函数引入非线性变换，使模型能够学习复杂模式，最终输出处理后的特征图用于后续层的计算。

C3K2 模块是早期版本中引入的 CSP(Cross Stage Partial) Bottleneck 的演变，用来处理骨干网络不同阶段的特征提取。C3K2 模块通过分割特征图，并应用一系列较小的  $(3 \times 3)$  卷积核进行卷积操作，优化了网络中的信息流，这些卷积比较大的卷积核更快，计算成本更低。与 YOLOv8 的 C2F 模块相比，C3K2 模块能够以更少的参数提升特征表示能力。C3K2 模块使用 C3K 模块来处理信息。它在开始和结束时各有一个 Conv 模块，中间是一系列的 C3K 模块。将起始 Conv 模块的输出与最后一个 C3K 模块的输出进行拼接，并以一个最终的 Conv 模块结束。这个模块借助 CSP 结构，致力于在速度和准确性之间保持平衡。C3K 模块的结构与 C2F 模块类似，但在此模块中不会进行分割操作。输入数据先经过一个卷积模块，随后经过一系列 Bottleneck，并以最终的 Conv 块结束。这三个模块的结构如图 2-2(图像来源：<https://medium.com/@nikhil-rao-20/yolov11-explained-next-level-object-detection-with-enhanced-speed-and-accuracy-2dbe2d376f71>) 所示。

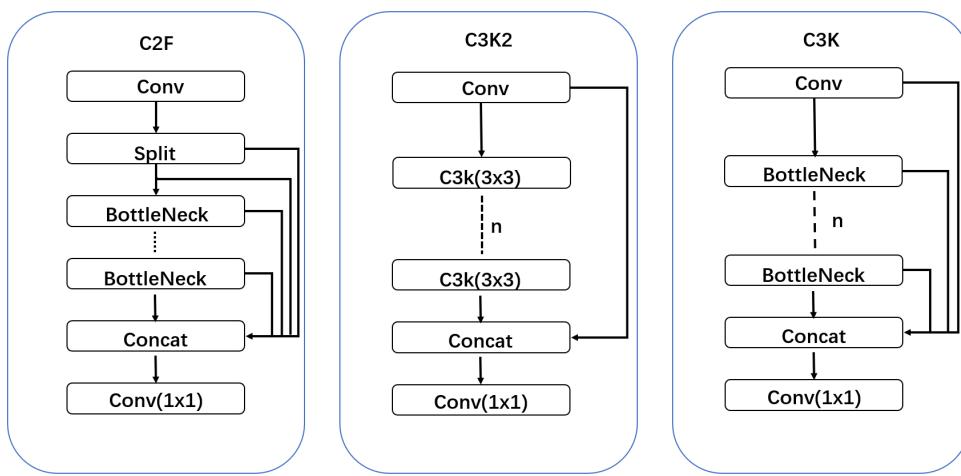


图 2-2 C2F 和 C3K2 模块的比较

SPPF(Spatial Pyramid Pooling - Fast) 模块是对 SPP(Spatial Pyramid Pooling) 模块的改进，主要用于增强模型对不同尺度目标的检测能力，其结构如图 2-3(图像来源：<https://zhuanlan.zhihu.com/p/4251780251>) 所示。SPPF 模块接收 C3K2 模块输出的特征图，对特征图同时应用多个不同大小的最大池化操作 (MaxPool)，然后将原始特征图和所有池化

后的特征图在通道维度上拼接，形成更丰富的多尺度特征表示，最后通过  $1 \times 1$  卷积压缩拼接后的特征通道数，减少计算量。

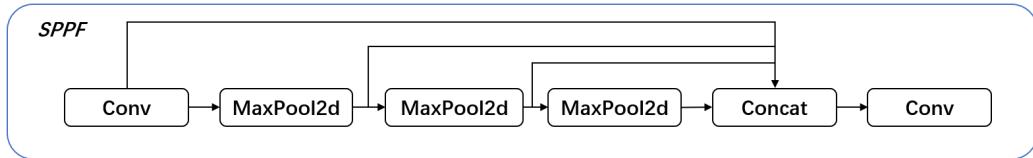


图 2-3 SPPF 模块结构

该模块采用多尺度最大池化核进行特征提取，有效实现对目标多尺度特征的全面表征，通过对不同尺寸池化核获取的特征图进行整合。

C2PSA 模块是 YOLOv11 的一大创新点，该模块结构如图 2-4(图像来源：<https://zhuanlan.zhihu.com/p/4251780251>) 所示。此模块引入了注意力机制，提高模型对图像中重要区域（例如较小或部分遮挡的对象）的关注。C2PSA 模块中的 Position-Sensitive Attention 封装了对输入张量应用位置敏感注意力和前馈网络的功能，提升了特征提取和处理能力。C2PSA 模块采用两个 PSA(Partial Spatial Attention) 模块，分别处理特征图分支后再拼接，类似 C2F 模块结构。这种设置在兼顾计算成本与检测精度的同时，让模型聚焦空间信息，使 YOLOv11 在需关注物体细节以实现精确检测的场景中优于 YOLOv8 等版本。

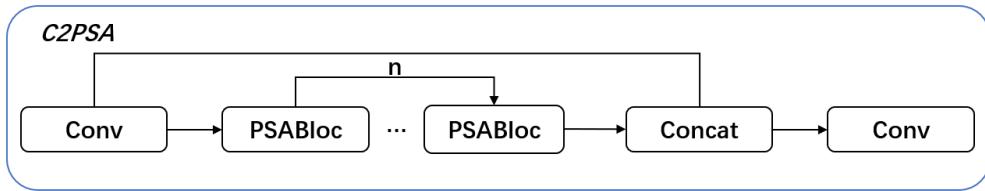


图 2-4 C2PSA 模块结构

### 2.1.2 颈部网络

颈部网络由多个 Conv 卷积层、C3K2 模块、Concat 操作和上采样模块组成，并结合了 C2PSA 机制的优势。

Conv 模块对特征图进行卷积运算，一方面可以调整特征图的通道数，使不同特征图在拼接前通道数匹配；另一方面能进一步提取特征，增强特征表达。C3K2 模块应用了一系列较小的卷积核，以更少的参数提升特征表示能力。Concat 模块沿着通道维度将多个特征图合并在一起，融合不同尺度的特征信息，增加特征的丰富度。上采样模块通过运

用插值等技术手段，实现特征图尺寸的扩充。这一操作的核心目的在于确保特征图维度与其他分支保持一致，借助这种方式，模型得以充分融合多尺度特征信息，提升了模型的检测精度。

颈部网络接收骨干网络不同层级输出的特征图，一些特征图会先经过 `Unsample` 上采样操作，使其尺寸与其他待拼接特征图一致，然后再进行 `Concat` 拼接。拼接后的特征图，再次经过若干 `C3K2` 模块和 `Conv` 卷积层，进一步融合与提炼特征。

### 2.1.3 检测头

与早期的 YOLO 版本类似，YOLOv11 使用多尺度预测头来检测不同大小的对象。头部使用由主干网络和颈部网络生成的特征映射输出三种不同比例（低、中、高）的检测框。检测头会输出来自三个特征映射（通常来自 P3、P4 和 P5）的预测，对应于图像中的不同尺度级别。这种方法可以确保在更精细的细节（P3）中检测到较小的对象，而在更高级别的特征（P5）中捕获较大的对象。

## 2.2 YOLOv11 损失函数

YOLOv11 的损失函数主要分为：边界框回归损失（BBox Loss）、分类损失（Classification Loss）和分布损失（Distribution Focal Loss, DFL）。

### 2.2.1 边界框回归损失

边界框回归损失函数为式 (2-1)， $S$  是网格的大小， $B$  是每个网格单元预测的边界框数量， $1_{ij}^{obj}$  表示第  $i$  个网格单元中第  $j$  个边界框是否负责预测目标。 $x, y$  是边界框中心点的坐标。 $w, h$  是边界框的宽度和高度。 $\lambda_{coord}$  是权重系数，用于平衡不同部分的损失。

$$\text{Box Loss} = \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^B 1_{ij}^{obj} ((x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2) \quad (2-1)$$

$$+ \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^B 1_{ij}^{obj} \left( (\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2 \right)$$

$$(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \quad (2-2)$$

$$(\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2 \quad (2-3)$$

式 (2-2) 衡量了预测边界框与真实边界框中心点之间的欧几里得距离平方。式 (2-3) 衡量了预测边界框与真实边界框的宽度和高度之间的差异。对宽度和高度取平方根减小了大尺寸边界框的影响，避免掩盖小尺寸边界框。

## 2.2.2 分类损失

分类损失函数见式(2-4)，其核心作用是衡量模型对目标类别的预测准确性，并指导模型通过优化算法（如梯度下降）调整参数，从而提升分类性能。

$$\text{Classification Loss} = \sum_{i=0}^{S^2} \mathbb{1}_i^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2 \quad (2-4)$$

$S$  是网格的大小。 $\mathbb{1}_i^{\text{obj}}$  表示第  $i$  个网格单元是否包含目标。 $p_i(c)$  是模型预测的第  $i$  个网格单元中目标属于类别  $c$  的  $\hat{p}_i(c)$  是真实的标签，表示第  $i$  个网格单元中目标是否属于类别  $c$ 。分类损失函数确保模型能正确识别图像中目标所属类别，最小化分类损失，使模型可处理多类别目标检测任务，提升模型分类准确性，助力模型学习类别特征差异，提高整体检测性能。

## 2.2.3 分布损失

目标检测领域普遍存在类别分布不平衡的现象，表现为特定类别的样本数量显著超过其他类别，这种数据分布失衡容易导致模型对高频类别表现出较好的识别性能，而对低频类别的检测准确率相对较低。分布损失函数用于优化模型在类别不平衡场景下的表现，其核心机制在于强化模型对困难样本的特征提取能力，从而显著提升对低频类别目标的检测效果。DFL 损失函数公式为式(2-5)。

$$\text{DFL} = - \sum_{i=1}^N \sum_{c=1}^C y_{ic} (\alpha(1 - p_{ic})^\gamma \log(p_{ic}) + (1 - \alpha)p_{ic}^\gamma \log(1 - p_{ic})) \quad (2-5)$$

$N$  是样本数量。 $C$  是类别的数量。 $y_{ic}$  是第  $i$  个样本的真实标签 (one-hot 编码，只有一个元素为 1，其余为 0)。 $p_{ic}$  是第  $i$  个样本属于类别  $c$  的预测概率。 $\alpha$  是平衡因子，用于调整正负样本之间的权重。 $\gamma$  是聚焦参数，用于控制对困难样本的关注程度。

## 2.3 数据集预处理

本文所使用的数据集来源于缅甸真实的交通道路照片，该数据集为公开资源 (数据集来源：<https://www.cvmart.net/dataSets/detail/627>)。首先使用 LabelImg 工具对数据集进行标注，生成 xml 格式的标注文件，再转换成 YOLO 格式的 txt 标注文件。YOLO 所使用的 txt 标注文件的格式如表 2-1 所示，文件的每一行有五个元素，其中中心点位置由 `x_center` 和 `y_center` 两个参数描述，分别对应于图像宽度和高度的相对比例值；而边界框的尺寸特征则通过 `width` 和 `height` 两个参数表征，同样表示为相对于图像原始宽度和高度的归一化比例值。上述五元组能够表示某一个类别的目标在图像中的位置。

本文的原数据集包含 5661 张摩托车驾乘人员头盔佩戴情况图像，由于其中某些类

表 2-1 YOLO 数据标注格式

class	x_center	y_center	width	height
5	0.18229123	0.72314815	0.09270814	0.19629641
14	0.40755224	0.68564817	0.05677574	0.18796234
15	0.84505233	0.61064834	0.04947934	0.12587944

别的样本数据非常少，对包含这些标签的原图像进行了图像增强，保证每一个类别至少有 100 张样本图像。具体的增强方法如图 2-5 所示。

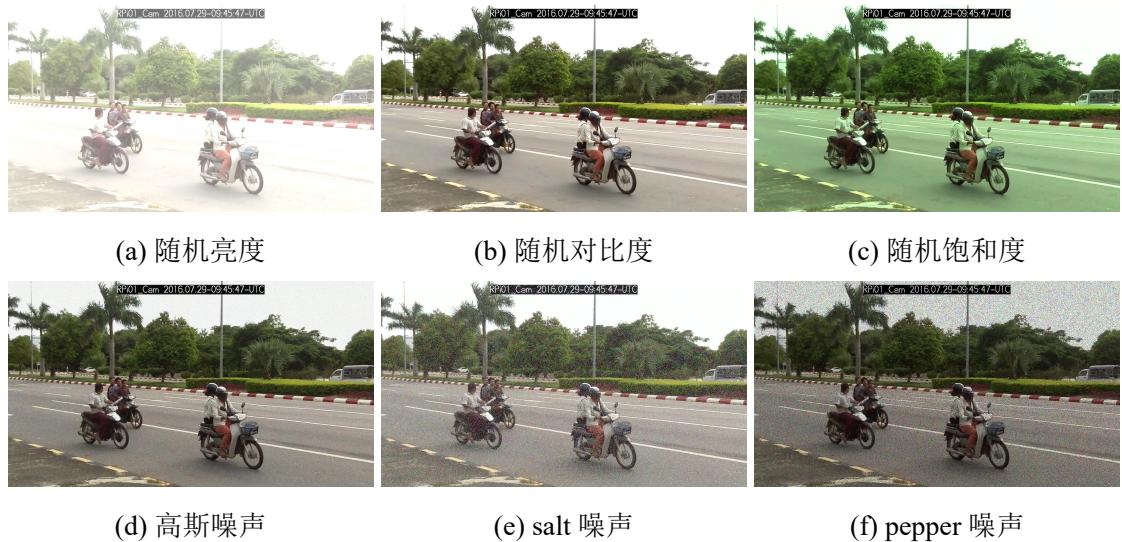


图 2-5 图像增强方式

对原数据集进行增强之后，共有 6282 张图像，采用 7:2:1 的比例对数据集进行随机划分，分别构建训练集、验证集和测试集。本文将目标类别分为 18 类，标签为 S01-S18，其对应关系以及含义如表 2-2 所示，其中 meaning 一列命名规则为：D(Driver) 代表摩托车驾驶员，P(Partner) 代表乘车人员。P1 代表驾驶员身后的一名乘车人员，P2 代表驾驶员身后的另一名乘车人员，P0 代表驾驶员身前的乘车人员（在原数据集中，某些小朋友会坐在驾驶员身前）。而 D、P0、P1 和 P2 后面紧跟着该乘车人员的头盔佩戴情况，Helmet 代表已佩戴头盔，NoHelmet 代表未佩戴头盔。例如，标签为 S09 的含义为 DNoHelmetP0NoHelmetP1NoHelmet，表示摩托车驾驶员 D 未佩戴头盔，驾驶员身前的乘车人员 P0 未佩戴头盔，驾驶员身后的一名乘车人员 P1 未佩戴头盔。

18 个类别的样本数量分布如图 2-6 所示，可以看出 S15 标签数量最多，共 6742 个，其表示单一驾驶员佩戴头盔；S06 标签数量次之，共 6346 个，其表示单一驾驶员未佩戴头盔，二者实例数量远高于其他标签。说明单一驾驶员驾驶摩托车是最常出现的情况，且佩戴头盔要比不佩戴头盔的情况稍多一些。除上述两个标签之外，S16 和 S04 是数量最

表 2-2 目标类别标签含义

class	label	meaning
0	S01	DHelmetP1NoHelmetP2NoHelmet
1	S02	DNoHelmetP1NoHelmetP2NoHelmet
2	S03	DHelmetP0NoHelmetP1NoHelmet
3	S04	DNoHelmetP1NoHelmet
4	S05	DHelmetP0NoHelmet
5	S06	DNoHelmet
6	S07	DNoHelmetP1NoHelmetP2Helmet
7	S08	DHelmetP1NoHelmet
8	S09	DNoHelmetP0NoHelmetP1NoHelmet
9	S10	DNoHelmetP1Helmet
10	S11	DHelmetP0NoHelmetP1Helmet
11	S12	DNoHelmetP0NoHelmetP1NoHelmetP2NoHelmet
12	S13	DHelmetP1NoHelmetP2Helmet
13	S14	DNoHelmetP0NoHelmet
14	S15	DHelmet
15	S16	DHelmetP1Helmet
16	S17	DHelmetP0Helmet
17	S18	DHelmetP1HelmetP2Helmet

多的两个标签，S16 标签共 3296 个，表示驾驶员和身后的一名乘客都佩戴头盔；S04 标签共 2503 个，表示驾驶员和身后的一名乘客都未佩戴头盔。

由表中数据可以看出，单一驾驶员驾驶摩托车的情况出现次数最多，其次是驾驶员携带一名乘客的情况。并且不管是单一驾驶员，还是驾驶员携带一名乘客，佩戴头盔的情况均多于不佩戴头盔的情况。而一名驾驶员携带一名以上乘客的情况比较少见。

对原数据集 5661 张图像中的每一个驾驶员目标框进行裁剪，得到 33571 张包含驾驶员信息的图像，共有 570 个不同驾驶员。由于驾驶员的图像样本也存在不平衡的情况，对这 33571 张图像同样进行图像增强，保证每一个驾驶员都至少有 100 张样本图像。

## 2.4 实验环境

操作系统：Ubuntu 20.04.3 LTS (Focal Fossa)

CPU：Intel(R) Xeon(R) Gold 5318Y CPU @ 2.10GHz

内存：1.5T

GPU：NVIDIA A40

显存：48G

CUDA 版本：12.2

Pytorch 版本：2.7.0+cu126

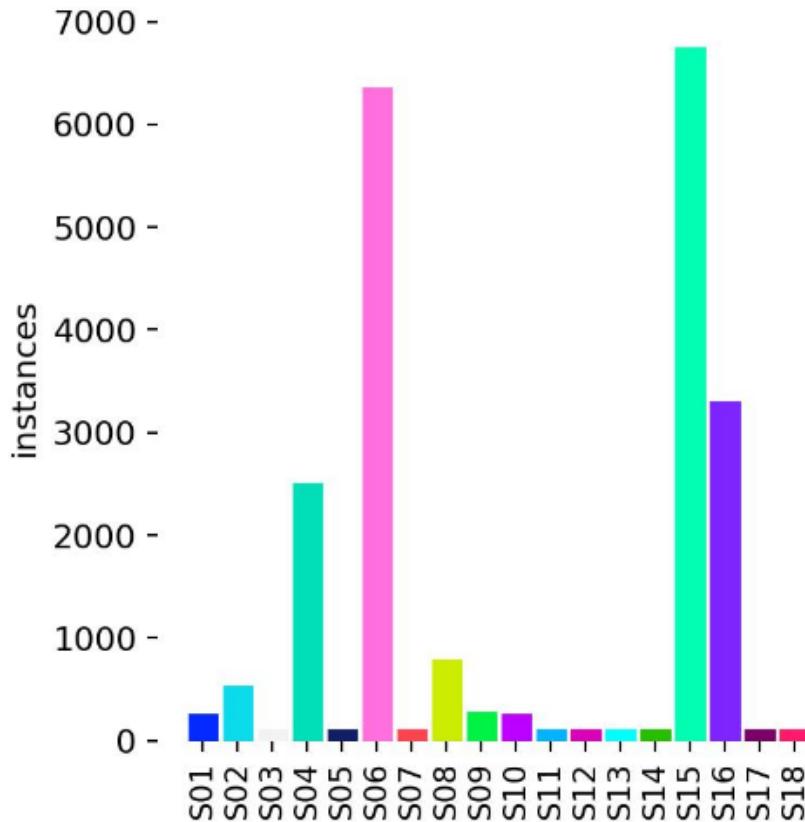


图 2-6 头盔佩戴情况标签数量分布

## 2.5 参数设置

本文分别使用 YOLOv11n.pt、YOLOv11s.pt、YOLOv11m.pt 进行训练，参数设置见表 2-3。本文进行头盔佩戴情况检测模型训练时，共有 18 个标签，在 yaml 文件里 nc 设置为 18。在训练模型中，将 batch 设置为 16，输入图像的 size 默认设置为 640x640，epochs 设置为 300。degrees 设置为 20，在指定度数范围内随机旋转图像，提升模型对摄像头拍摄到的不同角度的图像的识别能力。hsv\_v 设置为 0.6，将图像的亮度修改一部分，模拟不同的光照环境。translate 设置为 0.2，将图像进行水平和垂直平移，有助于检测部分可见物体。

**表 2-3 参数设置**

param	value	meaing
nc	18	类别数量
batch	16	训练批次
size	640x640	输入图像的尺寸
epochs	300	训练轮数
degrees	20	控制图像随机旋转的度数范围
hsv_v	0.6	控制图像亮度调整幅度
translate	0.2	控制图像在水平和垂直方向上的平移程度

## 2.6 本章小结

本章首先围绕 YOLOv11 算法相关理论展开，系统阐述了 YOLOv11 的网络结构，主要介绍了骨干网络、颈部网络和检测头，说明了 YOLOv11 对上述三个结构的改进点，并对 YOLOv11 中三个重要的损失函数做了详细的分析。然后介绍了原数据集的来源和本文使用到的六种数据增强方式，展示了需要训练的两个模型的标签及其数量分布。简要介绍了实验环境以及训练的参数设置。

## 第3章 实验结果与分析

### 3.1 评价指标

#### 3.1.1 精度评价指标

本文使用到的评价目标检测模型精度的指标有精度 (Precision)、召回率 (Recall)、平均精度 (Average Precision, AP) 及平均精度均值 (mean Average Precision, mAP)。

精度 (Precision) 和召回率 (Recall)，是衡量模型预测准确性与完整性的核心指标。精度的定义如式 (3-1)，其中 TP 表示模型预测为正例，且实际标签也是正例，FP 表示模型将负例误判为正例。精度数值越大，表明 FP 值越小，反映出模型预测结果中真正正例的占比更高，误检更少，预测的准确性更好。召回率 (Recall) 的定义如式 (3-2)，TP 同精度公式中的 TP，FN 表示模型漏掉的正例。召回率数值越大，代表 FN 值越小，说明模型对实际正例的捕捉能力更强，能够找到更多的正例样本。

$$Precision = \frac{TP}{TP + FP} \quad (3-1)$$

$$Recall = \frac{TP}{TP + FN} \quad (3-2)$$

作为评估模型性能的重要指标，平均精度 (AP) 采用 P-R 曲线与坐标轴围成区域的面积进行量化表征。从几何学视角分析，较高的 AP 值直观体现了模型在精度和召回率两个核心评价维度上的协同优化效果，表明其既能维持较高的正样本识别准确度，又具备较强的正样本覆盖能力。在综合评价体系方面，平均精度均值 (mAP) 通过对所有类别的 AP 计算结果求均值来表征模型的整体识别准确率。YOLO 模型主要采用 mAP50 和 mAP50-95 两种指标：前者设定 IoU 阈值为 0.5 时计算各类别平均精度的平均值；后者则在 IoU 阈值 0.5 至 0.95 范围内进行多尺度评估，最终取各阈值下 mAP 结果的平均值作为综合性能指标。

#### 3.1.2 速度评价指标

本文采用每秒帧率 (FPS) 与浮点运算量 (FLOPs) 作为目标检测模型速度性能的量化评估指标。其中，FPS 指标反映了模型在单位时间内能够检测的图像帧数，其数值越高，意味着一秒内能执行更多的目标检测任务；而 FLOPs 则用于量化模型执行目标检测任务过程中所涉及的浮点运算次数，该指标能够有效表征模型的计算量，FLOPs 越低，理论速度越快，但并不直接等于实际速度。

## 3.2 实验结果分析

### 3.2.1 头盔佩戴情况模型

本文基于 YOLOv11n、YOLOv11s 和 YOLOv11m 三个模型，各训练了 300 个 epoch，头盔佩戴情况模型训练结果如图 3-1 和图 3-2 所示。下面对边界框回归损失、分类损失、分布损失、精度、召回率、mAP50 和 mAP50-95 这七个指标进行详细对比分析。

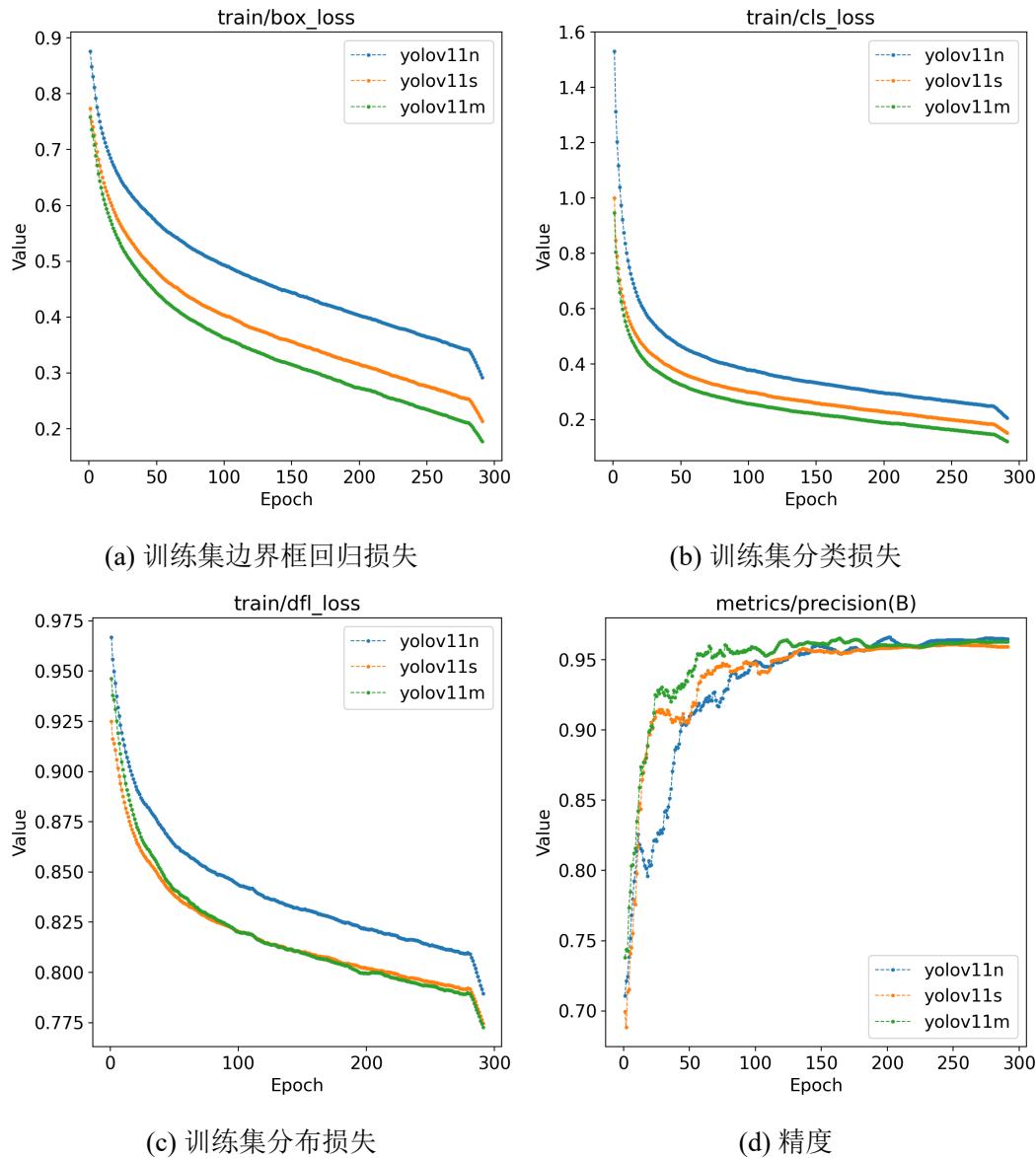


图 3-1 头盔佩戴情况模型训练结果（第一部分）

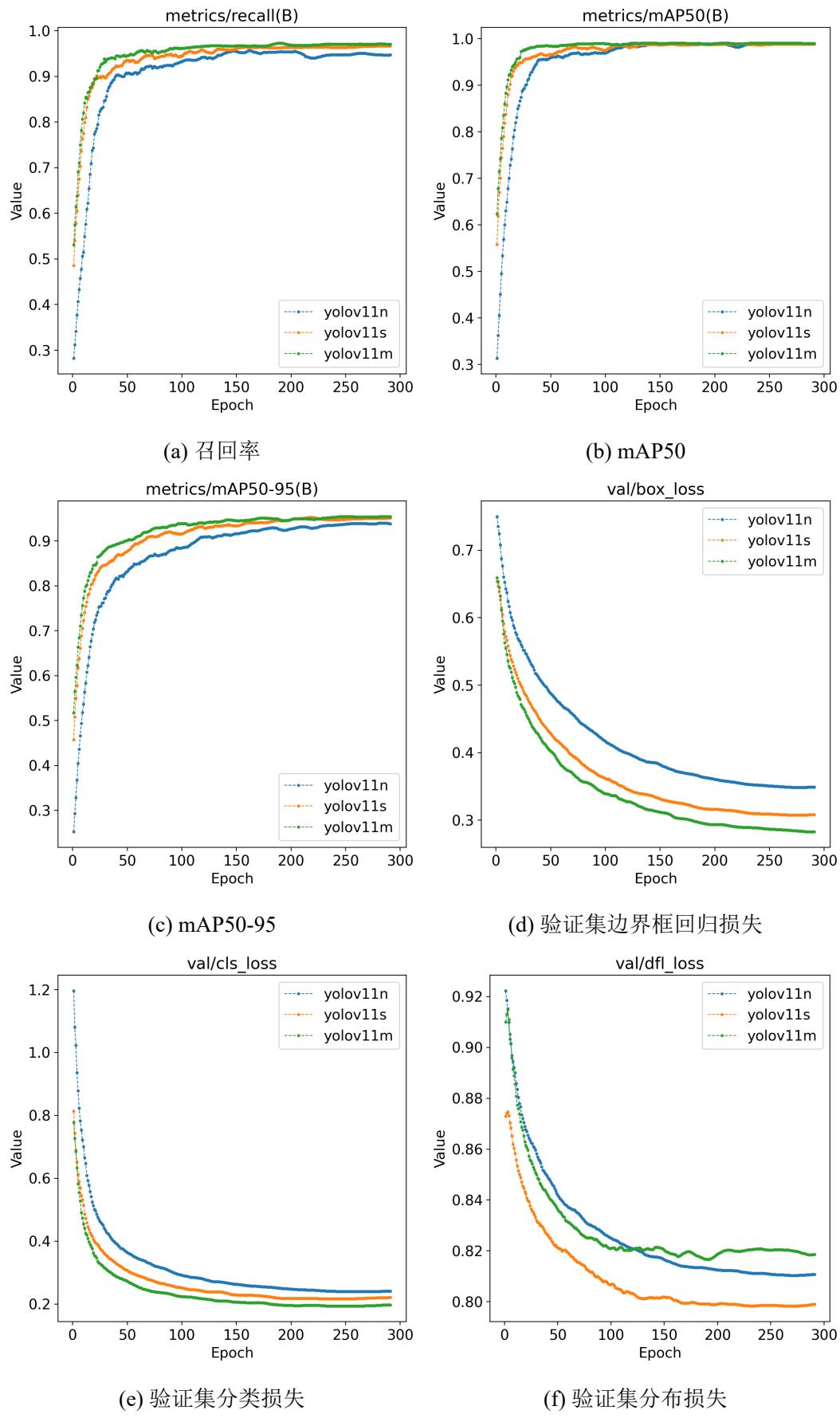


图 3-2 头盔佩戴情况模型训练结果（第二部分）

边界框回归损失：三个模型的边界框回归损失值在训练过程中逐渐降低。YOLOv11n 下降趋势相对较缓，YOLOv11s 和 YOLOv11m 前期下降速度较快且相近，在训练进行大约 50 轮之后，YOLOv11m 下降速度略快于 YOLOv11s，说明 YOLOv11m 在边界框回归损失的收敛上最具优势。在训练后期，YOLOv11m 的边界框回归损失最终值最低，YOLOv11s 次之，YOLOv11n 相对较高。这表明 YOLOv11m 在边界框定位的准确性上表现更好。

分类损失：三个模型分类损失收敛趋势相似，均在前期快速下降，后期平稳下降。前期 YOLOv11n 下降稍快，YOLOv11s 与 YOLOv11m 相近。中期 YOLOv11s 和 YOLOv11m 的下降速度与 YOLOv11n 相似，后期三者收敛速度相近。YOLOv11m 最终的分类损失值最低，YOLOv11s 和 YOLOv11n 相对较高且较为接近，意味着 YOLOv11m 在分类任务上的准确性更高。

分布损失：YOLOv11s 与 YOLOv11m 整体收敛趋势相似，前期都比 YOLOv11n 收敛速度略快，后期三者收敛速度近似。YOLOv11s 与 YOLOv11m 达到的最优值近乎相同，且都比 YOLOv11n 的最优值更好。

精度：三个模型精度均随训轮数增多逐渐变高。YOLOv11s 和 YOLOv11m 前期上升速度较快且相近，YOLOv11n 稍慢；后期三者趋势相似。YOLOv11m 达到的精度最优值最高，YOLOv11s 次之，YOLOv11n 相对较低，表明 YOLOv11m 在预测准确程度上表现更优。

召回率：YOLOv11s 与 YOLOv11m 前期收敛趋势相同且均比 YOLOv11n 收敛速度快；后期三个模型的召回率收敛速度都逐渐趋于平稳。YOLOv11m 召回率最优值最高，YOLOv11s 与之接近，YOLOv11n 相对较低。说明 YOLOv11m 在捕捉正例样本能力上表现最好。

mAP50：YOLOv11s 和 YOLOv11m 的 mAP50 值在训练前期上升迅速，YOLOv11n 稍慢；中后期 YOLOv11m 率先趋于稳定，YOLOv11s 次之。最后三个模型得到的 mAP50 最优值近似。

mAP50-95：YOLOv11s 和 YOLOv11m 前期收敛速度快且相近，YOLOv11n 比较缓慢；后期 YOLOv11m 收敛更平稳，更快达到较高精度水平。YOLOv11m 的 mAP50-95 最优值最高，YOLOv11s 与之相近，YOLOv11n 较低。表明 YOLOv11m 与 YOLOv11s 在不同 IoU 阈值下平均精度综合表现都比较好。

根据训练结果中的 csv 文件，能够得到各项指标在每一轮训练中的精确值，上述指标在训练过程中的最优值的对比如表 3-1 所示。从表格呈现的趋势来看，随着模型规模增大，三个损失函数的收敛速度更快，最终达到的损失值更低。其余四项指标，除检测精度外都随模型规模增大表现更优。

表 3-1 模型损失函数与检测精度指标对比 1

Model	box_loss (%)	cls_loss (%)	dfl_loss (%)	Precision (%)	Recall (%)	mAP50 (%)	mAP50-95 (%)
YOLOv11n	28.4	20.2	78.6	96.7	96.6	99.0	94.1
YOLOv11s	20.5	14.7	77.2	96.2	97.2	99.0	95.4
YOLOv11m	17.1	11.8	77.1	97.3	98.1	99.1	95.5

除了检测精度，模型的推理速度与计算复杂度同样是衡量性能的关键要素，表 3-2 对比了 YOLOv11n、YOLOv11s 和 YOLOv11m 三个版本基于实测数据的 FPS 与 FLOPs 数值。对于 FPS 指标，YOLOv11n 的值最高，达到 41.67，意味着该模型每秒能够处理 41.67 帧图像，实时性最强；YOLOv11s 的 FPS 为 36.63，仅次于 YOLOv11n；YOLOv11m 的 FPS 最低，为 31.75。由此可见，随着模型规模增大（从 n 到 m），检测速度逐渐降低，因为更大的模型通常包含更多的参数和计算操作，需要更长的推理时间。对于 FLOPs 指标，YOLOv11m 的 FLOPs 值最高，为 68.26G，表明其完成一次前向推理所需的浮点运算次数最多，计算复杂度最高；YOLOv11s 的 FLOPs 为 21.59G；而 YOLOv11n 的 FLOPs 最低，仅为 6.4G。综合两项指标来看，YOLOv11 模型的检测精度与计算速度呈现负相关性。YOLOv11n 在检测精度上略逊于其他版本，但其计算复杂度低、检测速度快；而 YOLOv11m 推理速度较慢，计算量最大，但有更高的检测精度。

表 3-2 模型检测速度对比 1

Model	FPS(1)	FLOPs(G)
YOLOv11n	41.67	6.4
YOLOv11s	36.63	21.59
YOLOv11m	31.75	68.26

三个模型的混淆矩阵如图 3-3、图 3-4、图 3-5 所示。YOLOv11n 对 S07 类别的检测效果较差，有 33% 的错误预测情况。对 S09 类别有 6% 的误判和 6% 的漏检。对剩余标签的检测准确度都在 94% 以上。YOLOv11s 相较于 YOLOv11n，对 S07 和 S09 两个标签的检测准确度有大幅提升，且对所有标签的检测精度都在 94% 以上。YOLOv11m 相较于 YOLOv11n，对各类别的检测精度近似，没有大幅变化，且对所有标签的检测精度也都在 94% 以上。

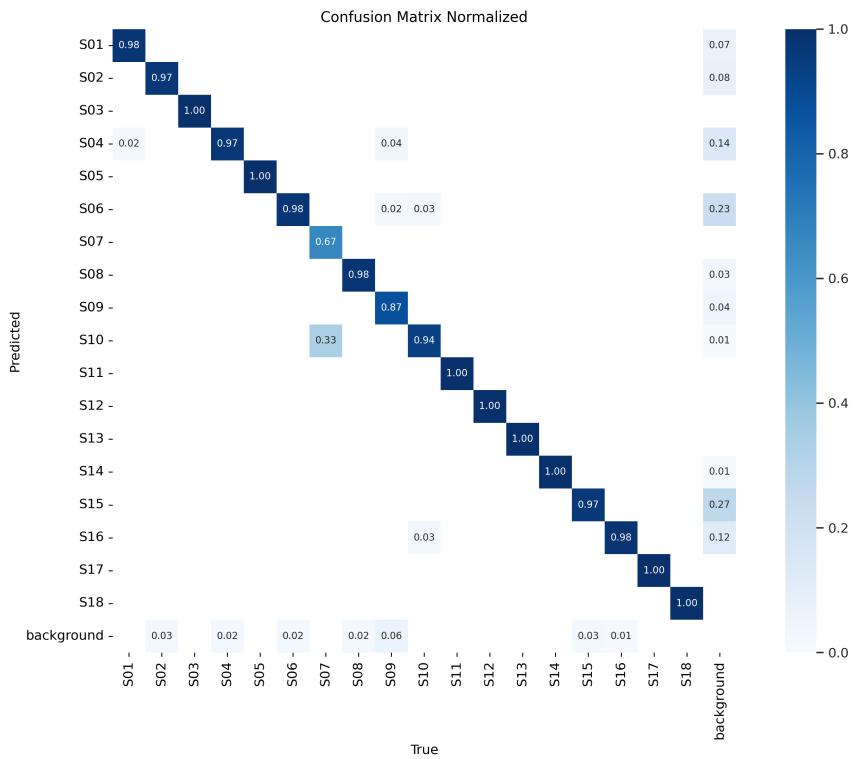


图 3-3 YOLOv11n 混淆矩阵

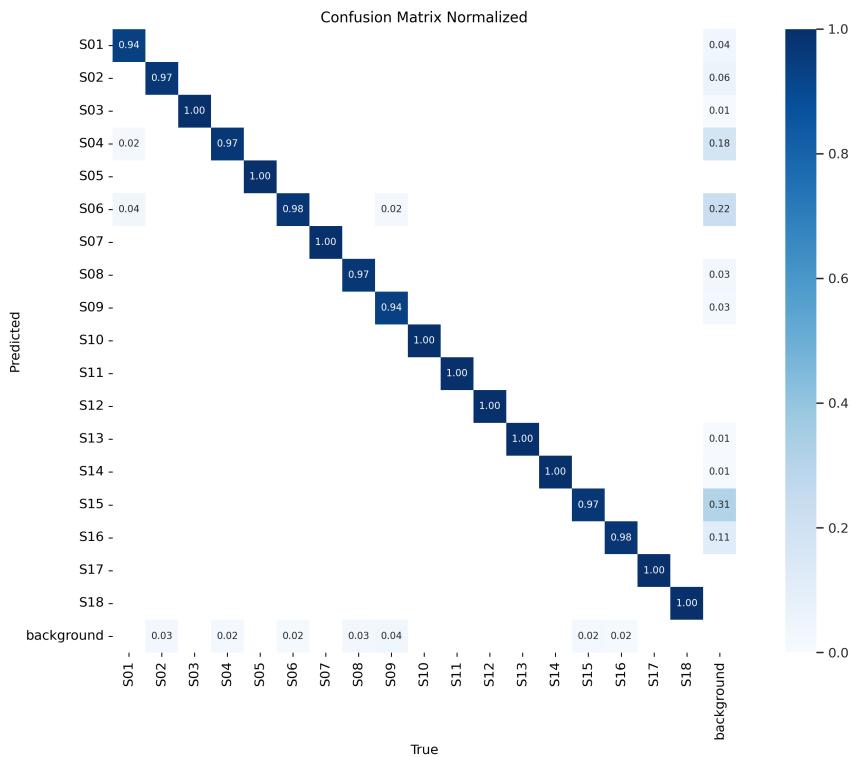


图 3-4 YOLOv11s 混淆矩阵

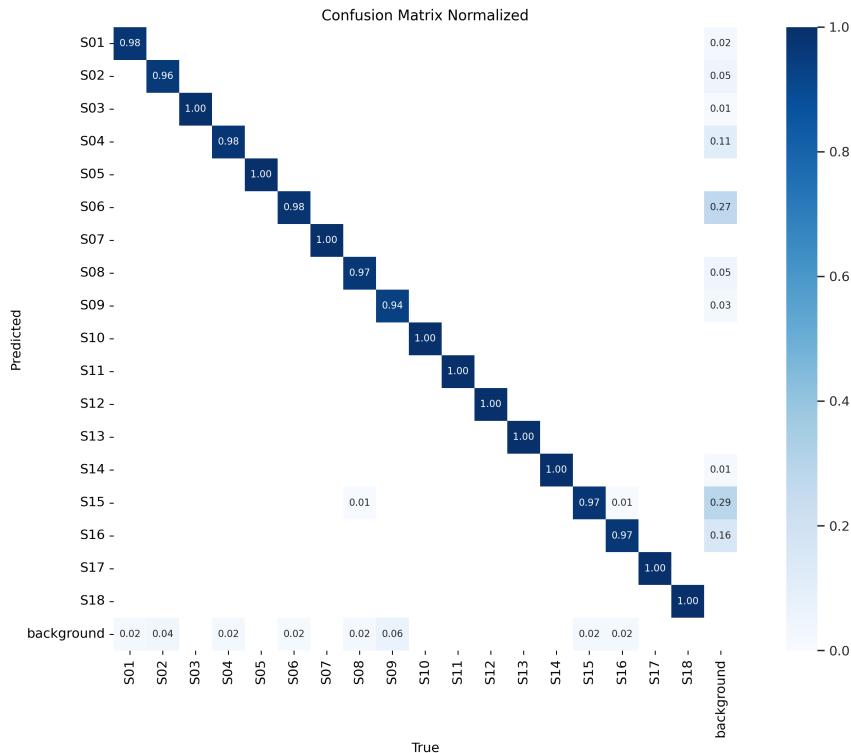


图 3-5 YOLOv11m 混淆矩阵

综合上文对三个模型在检测精度、检测速度和混淆矩阵三个方面的对比，YOLOv11n 模型的 FPS 为 41.67, FLOPs 为 6.4, 检测速度最快且计算量最小；召回率为 96.6%，mAP50 为 99.0%，mAP50-95 为 94.1%，检测精度相对较差，其适用于实时检测场景；YOLOv11s 模型的 FPS 为 36.63%，FLOPs 为 21.59%；精度为 96.2%，召回率为 97.2%，mAP50 为 99.0%，mAP50-95 为 95.4%，其在检测精度、检测速度和计算复杂度方面取得了平衡，适用于对实时性和精度都有一定要求的场景；YOLOv11m 模型的 FPS 为 31.75%，FLOPs 为 68.26%，检测速度慢且计算量大；精度为 97.3%，召回率为 98.1%，mAP50 为 99.1%，mAP50-95 为 95.5%，其牺牲了一定的检测速度，换来了更高的精度，适用于对检测精度要求较高的场景。

### 3.2.2 驾驶员 id 模型

本文基于 YOLOv11n、YOLOv11s 和 YOLOv11m 三个模型，各训练了 300 个 epoch，驾驶员 id 模型训练结果如图 3-6 和图 3-7 所示。下面同样对七个检测精度指标进行详细对比分析。

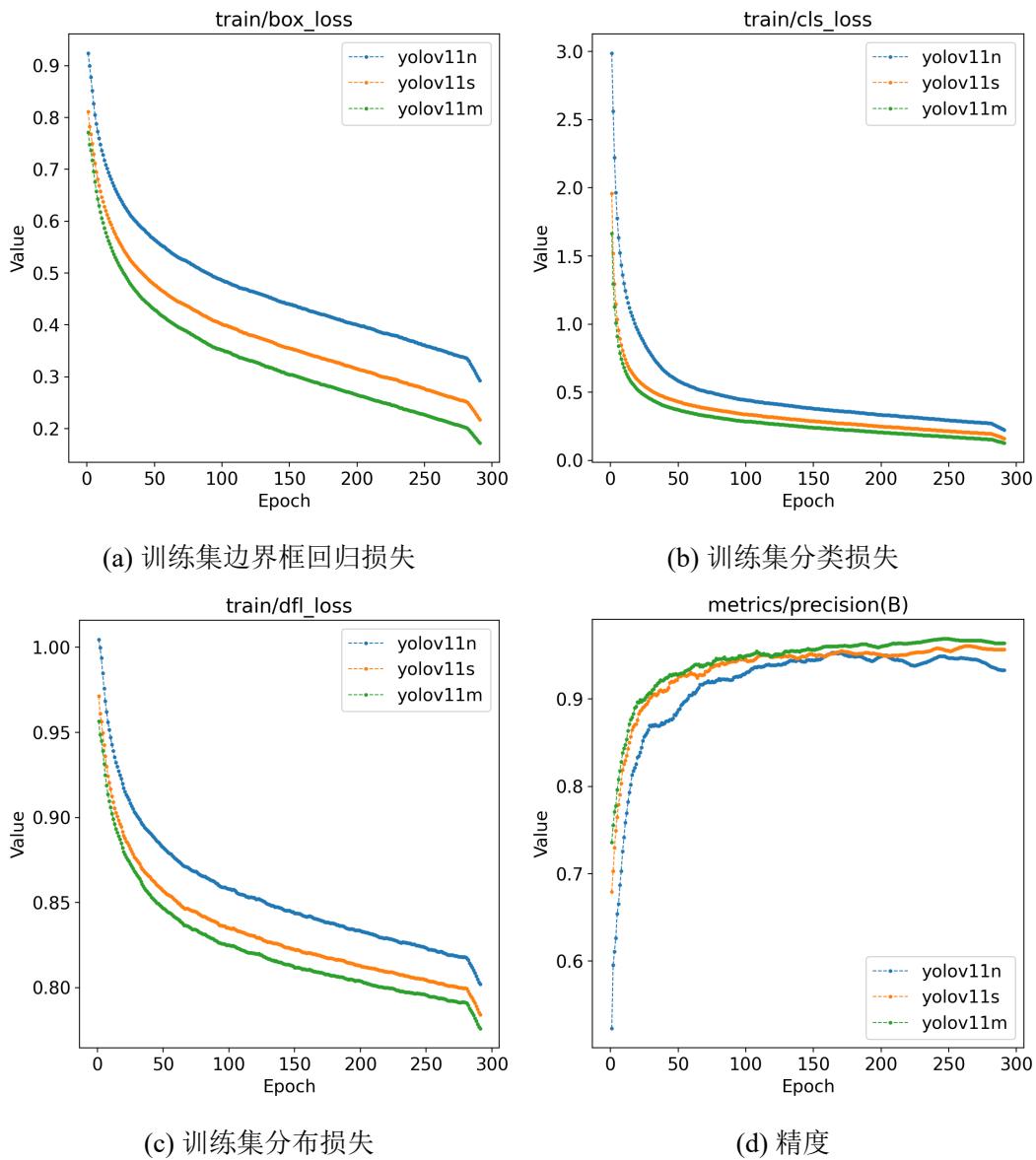


图 3-6 驾驶员 id 模型训练结果（第一部分）

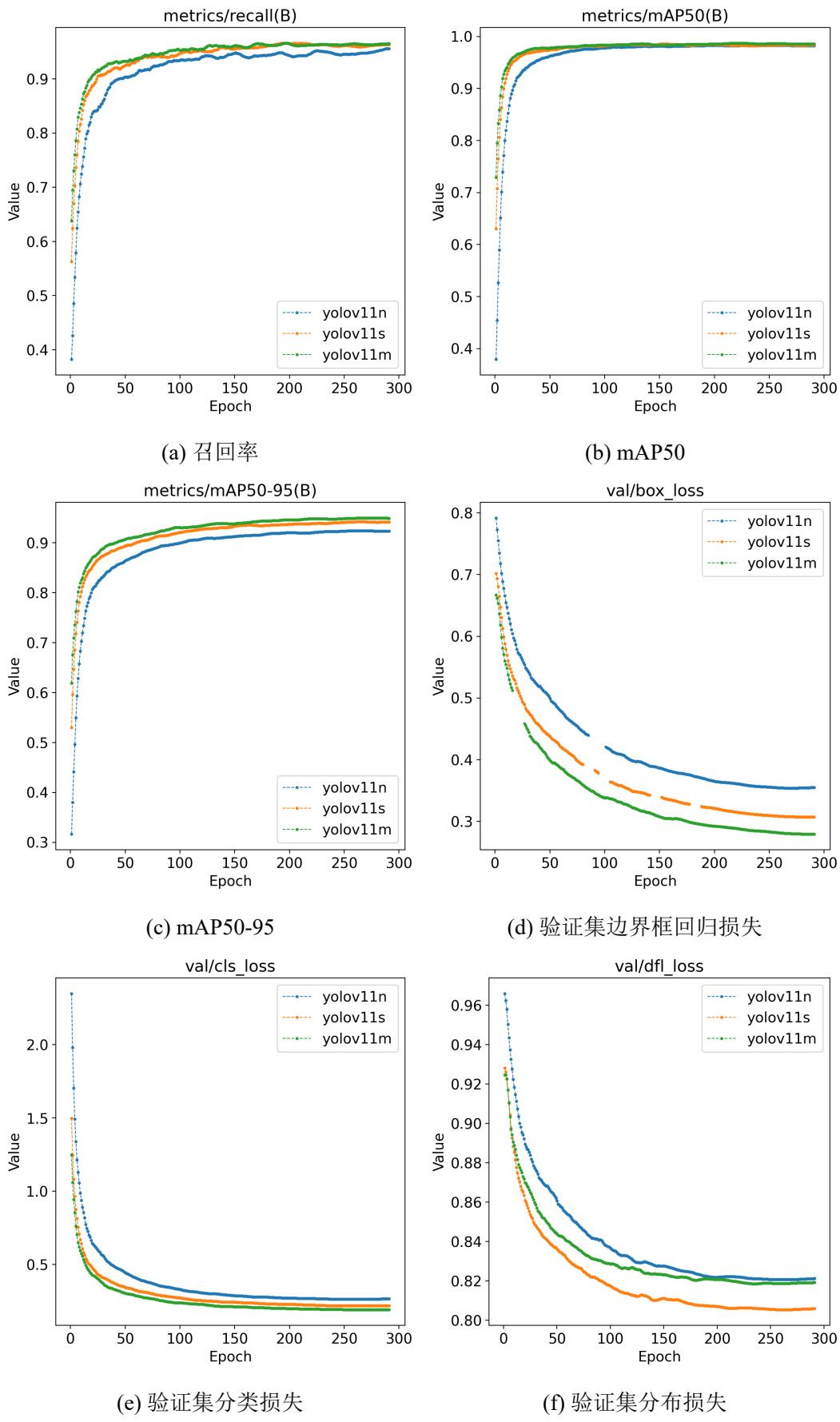


图 3-7 驾驶员 id 模型训练结果（第二部分）

边界框回归损失：YOLOv11s 和 YOLOv11m 两个模型的边界框回归损失在前期下降速度相较于 YOLOv11n 略快，中期以后三者下降速度相近。YOLOv11m 训练得到的边界框回归损失最优，数值最小，YOLOv11s 次之，YOLOv11n 相对较高。说明 YOLOv11m 模型预测驾驶员 id 的边界框更精确。

分类损失：YOLOv11s 和 YOLOv11m 的分类损失在前期下降较快，YOLOv11n 稍慢一些，中期以后三者下降速度近似。YOLOv11m 的分类损失值最小，其次是 YOLOv11s，YOLOv11n 的值最高。YOLOv11m 模型对驾驶员 id 的分类效果更好。

分布损失：YOLOv11m 和 YOLOv11s 的分类损失值收敛较快，YOLOv11n 收敛相对较慢。训练结束时 YOLOv11m 的分布损失值最小，效果最好，YOLOv11s 次之，YOLOv11n 的值最大。

精度：YOLOv11m 收敛速度最快，最早到达较高水平并趋于平稳。YOLOv11n 和 YOLOv11s 上升速度相对较慢。最终 YOLOv11m 达到的精度最高，其次是 YOLOv11s，YOLOv11n 的精度最低。这表明 YOLOv11m 对驾驶员 id 的预测更准确。

召回率：YOLOv11m 和 YOLOv11s 的收敛趋势近似，且都比 YOLOv11n 的收敛速度要快。YOLOv11m 和 YOLOv11s 最终的召回率相近，都比 YOLOv11n 的召回率要高。说明 YOLOv11m 和 YOLOv11n 能找到更多的正例驾驶员 id 目标。

mAP50：YOLOv11m 和 YOLOv11s 的 mAP50 值收敛趋势比较相似，都比 YOLOv11n 的收敛速度要快。最终三个模型的 mAP50 值几乎相同。

mAP50-95：YOLOv11m 的 mAP50-95 值收敛速度最快，其次是 YOLOv11s，YOLOv11n 的收敛速度最慢。训练结束时，YOLOv11m 达到了最高的 mAP50-95 值；YOLOv11s 次之；YOLOv11n 相对较低。

将上述七个指标训练过程中的最优值进行对比如表 3-3。三个损失函数 (box\_loss、cls\_loss、dfl\_loss) 随模型规模增大，最优值不断减小；四个精度指标 (precision、recall、mAP50、mAP50-95) 则随模型规模增大，最优值持续升高，模型检测精度随之提升。

表 3-3 模型损失函数与检测精度指标对比 2

Model	box_loss (%)	cls_loss (%)	dfl_loss (%)	Precision (%)	Recall (%)	mAP50 (%)	mAP50-95 (%)
YOLOv11n	28.5	21.9	79.9	95.6	95.8	98.3	92.5
YOLOv11s	21.0	15.7	78.2	96.1	96.7	98.7	94.3
YOLOv11m	16.6	12.4	77.3	96.9	96.9	98.7	95.0

三个模型的检测速度对比如表 3-4。随着模型规模增大，每秒处理帧数降低，实时性变差；且 FLOPs 数值增大，说明在预测中进行的浮点运算次数更多，理论上需要更长的推理时间。

表3-4 模型检测速度对比2

Model	FPS(1)	FLOPs(G)
YOLOv11n	67.11	7.49
YOLOv11s	66.67	22.68
YOLOv11m	55.87	70.45

综合上述对比结果，YOLOv11n 在检测驾驶员 id 场景下检测精度较低，但是检测速度最快，计算复杂度最低，适合对精度要求不是很高的实时检测场景；YOLOv11s 的检测精度和速度更加平衡，适合对精度和速度都有一定要求的场景；YOLOv11m 有着最高的检测精度，但带来的资源开销更大，适合高精度检测场景。

### 3.3 本章小结

本章首先介绍了目标检测模型检测精度和检测速度的几个重要评价指标，然后对本文进行的两个模型训练结果进行展示，对 YOLOv11n、YOLOv11s 和 YOLOv11m 三个模型的检测精度与检测速度进行了对比分析。YOLOv11 系列不同规模的模型具有不同的性能特征与适用场景：YOLOv11n 以牺牲部分检测精度为代价，换取了卓越的检测速度，特别适用于对实时性要求严苛的场景，如智能监控、自动驾驶的实时感知模块；YOLOv11s 在检测精度与速度之间取得了良好的平衡，能够满足多数场景下对二者的综合需求，是通用目标检测任务的优选方案；YOLOv11m 则有着更高的检测精度，尽管检测速度相对较慢，却在对精度要求极高的复杂场景中展现出显著优势，例如工业缺陷检测、医疗影像分析等对准确性要求苛刻的领域。

## 第4章 检测系统的设计与实现

### 4.1 需求分析

该系统需要为用户提供头盔佩戴情况检测和历史检测数据查询两个功能，本文为这两个功能各自设计并实现了前端页面。检测页面允许用户上传待检测图像、静态视频文件和实时监控视频，并需要支持用户自定义检测模型、IoU 和置信度参数。后端接收到参数之后调用模型进行头盔佩戴情况检测，并返回检测结果给前端。查询页面可供用户查询历史检测记录，允许用户对驾驶员、检测时间、检测地点等字段进行过滤。通过可视化图表展示查询结果，便于用户进行数据分析。

### 4.2 系统整体架构

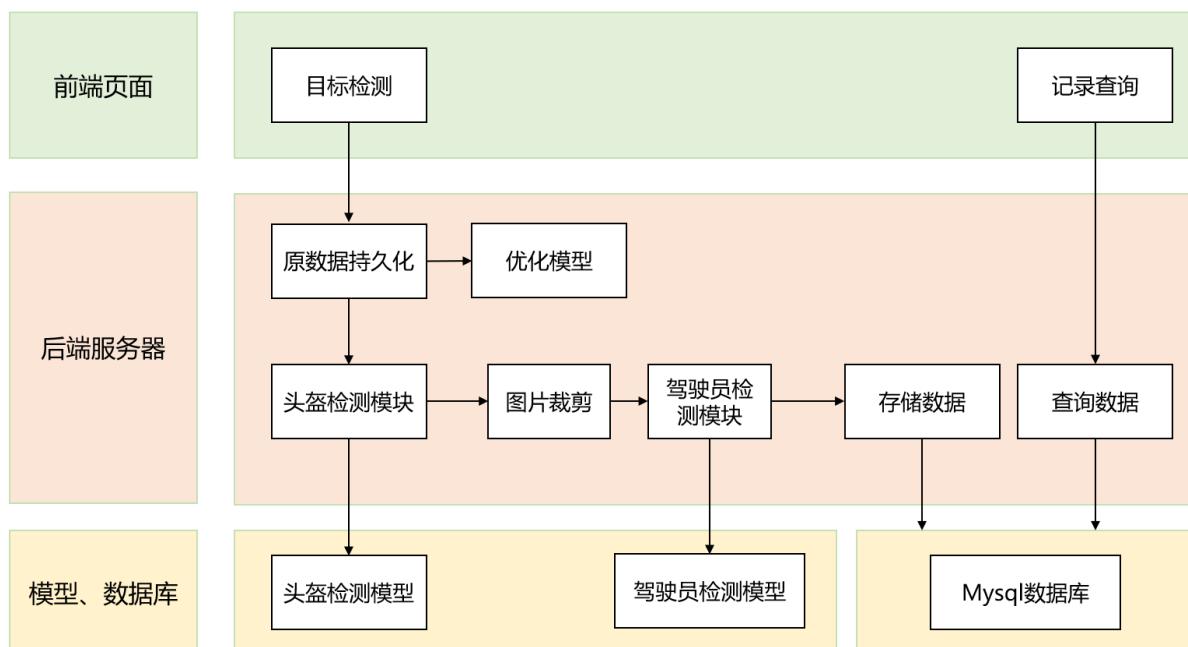


图 4-1 系统架构图

本系统整体架构如图 4-1 所示。本系统前端为用户提供目标检测和记录查询功能。目标检测界面展示后端返回的带检测框的结果并提供保存功能；记录查询界面通过 ECharts 组件以柱状图、折线图和饼图可视化展示后端查询的历史数据。

后端处理目标检测请求时：先持久化检测原数据，用于后续优化模型；再将用户传入的自定义检测模型、IoU 及置信度参数动态拼接到 python 脚本，调用训练好的头盔佩戴检测模型预测，并按边界框裁剪原数据；接着将裁剪图像作为驾驶员 id 检测模型的输入，检测驾驶员信息；最后把检测结果存至 MySQL 数据库，并将带检测框结果返回前端。

处理记录查询请求时，根据前端传来的过滤条件动态拼接 Sql 语句查询数据库。

### 4.3 前端模块设计与实现

前端界面分为目标检测界面和记录查询界面。两个界面都以灰、蓝色为主题色调，并且背景以及各个功能区的 CSS 样式几乎一样，使得两个页面非常协调。

#### 4.3.1 目标检测界面

目标检测界面用例设计如图 4-2。核心用例为头盔佩戴检测，其包含上传待检测资源、选择模型，设置 IOU 和置信度、请求后端检测、浏览检测结果和保存检测结果五个子用例。上传待检测资源包含上传视频和上传图像两个子用例，支持用户上传视频和图像资源。选择模型，设置 IOU 和置信度用例允许用户选择检测模型，自定义检测 IOU 和置信度阈值。请求后端检测用例会将待检测资源和用户设置参数上传至后端服务器请求头盔佩戴检测。检测结果浏览功能负责把检测所得结果展示在页面，并且提供保存功能，让用户可以把展示的检测结果保存至本地，实现数据的本地化存储。

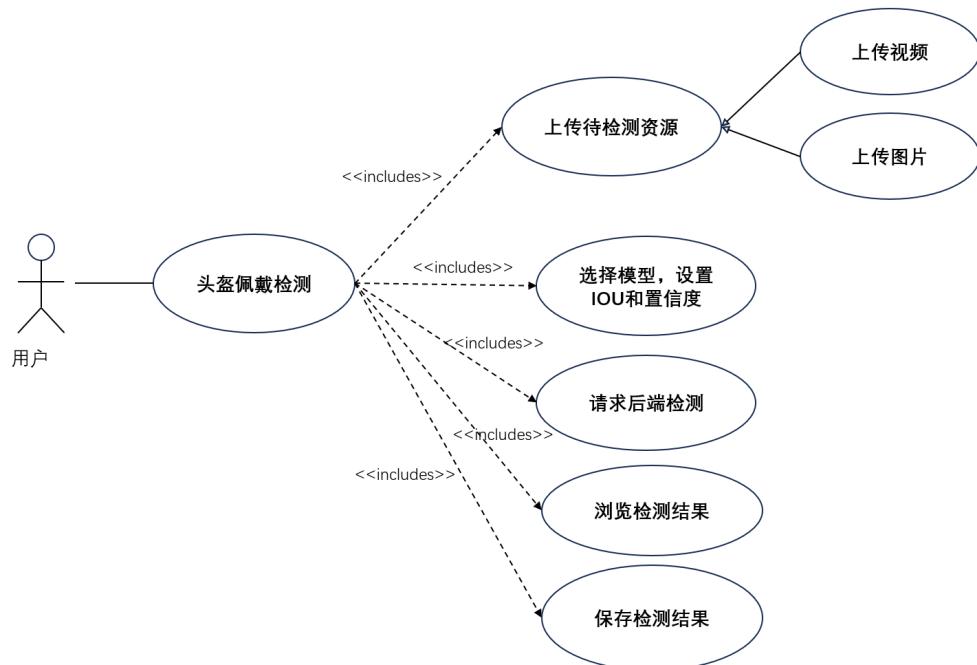


图 4-2 目标检测界面用例图

图 4-3展示了上述用例的交互顺序。用户需要先上传待检测资源，前端接收到资源后会展示在页面上。然后用户选择模型并调整参数，请求头盔佩戴检测功能。前端得到后端检测结果后，会将结果展示在页面上供用户浏览、保存。

目标检测界面的实现如图 4-4所示，核心是显示区、参数设置区、系统消息区和功能触发区。显示区用于浏览待检测资源和结果；功能触发区中，“上传图像”“上传视频”

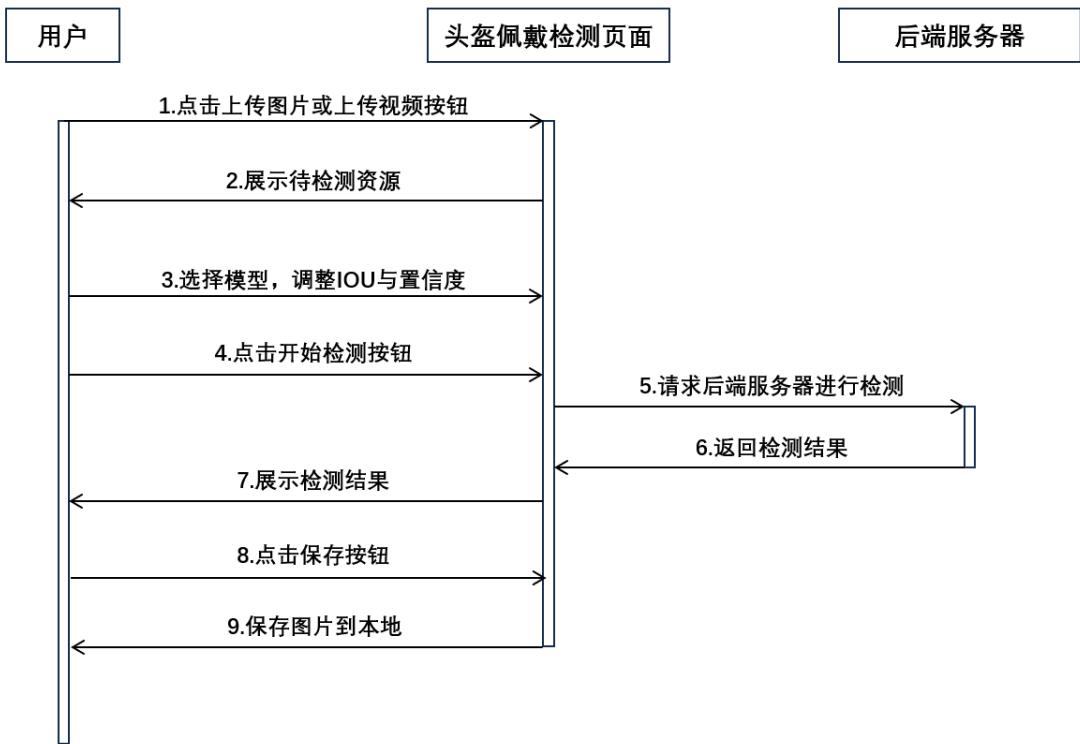


图 4-3 目标检测界面顺序图

按钮可上传待检测资源，“开始检测”按钮请求后端检测，“保存结果”按钮能将结果存至本地；参数设置区支持选择检测模型并设置 IOU 和置信度；系统消息区会展示用户操作情况。



图 4-4 目标检测界面布局

### 4.3.2 记录查询界面

记录查询界面用例设计如图 4-5。该页面以历史记录查询用例为核心。改用例包含设置查询过滤条件、请求后端查询和数据可视化三个子用例。设置查询过滤条件分为设置驾驶员、检测地点和检测时间三个子用例，允许用户对着三个字段设置过滤条件。通过请求查询用例发起查询请求。得到的查询结果会通过数据可视化用例通过图标的形式展示，提供了柱状图、折线图和饼图三种实现方式。

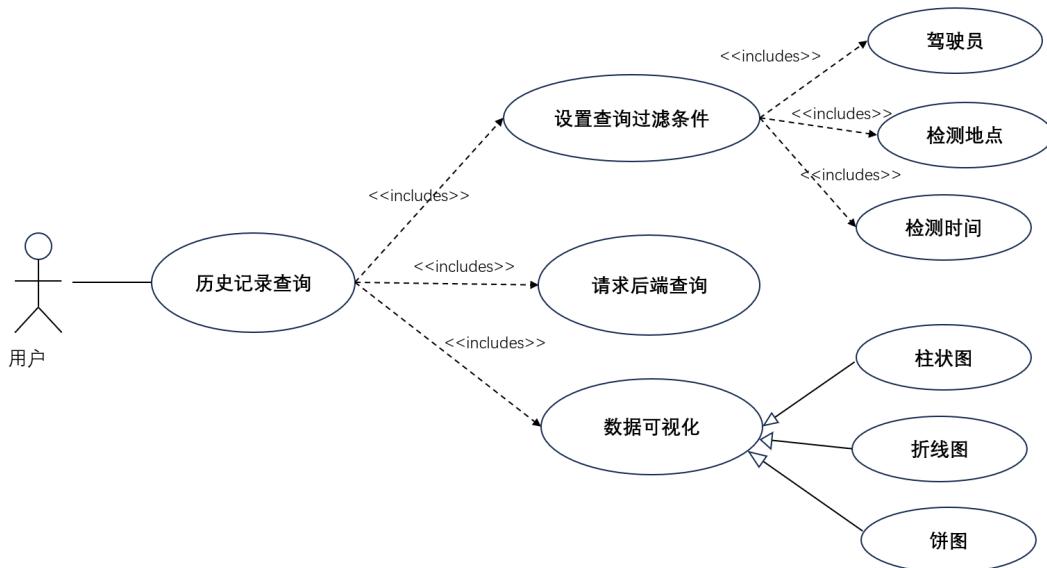


图 4-5 记录查询界面用例图

该页面的顺序图为图 4-6。用户需要先对驾驶员、检测地点和检测时间这三个字段设置查询过滤条件，不设置则认为不对该字段进行过滤，然后点击搜索按钮，前端页面会附带过滤条件向后端服务器发起查询请求。查询结果返回后，默认会展示柱状图类型的可视化图标，用户可以点击不同的图标类型选择不同的数据可视化方式。

记录查询界面的实现见图 4-7。用户通过记录查询区设置过滤条件并发起查询请求，查询结果会在数据可视化区以图标的形式展示，并允许用户选择不同的可视化方式。

## 4.4 后端模块设计与实现

后端模块在框架上选用 SpringBoot 进行开发，在架构上选择 MVC 三层架构。该系统主要分为两个模块：检测模块和搜索模块，负责实现前端页面提供的两大功能。

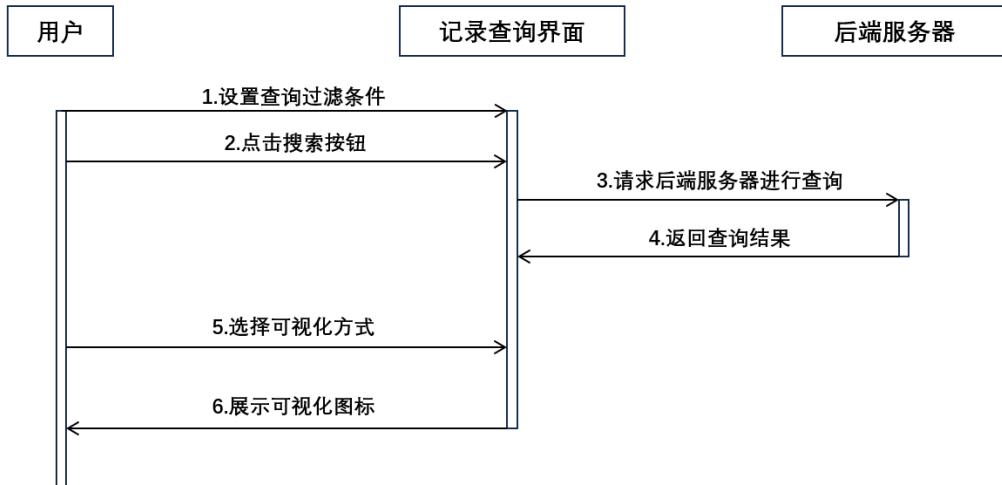


图 4-6 记录查询界面顺序图



图 4-7 记录查询界面布局

#### 4.4.1 库表结构

该系统使用的库表结构如表 4-1 所示。该表共有五个字段，除主键 id 外，记录了驾驶员信息、头盔佩戴情况、检测地区和检测时间。头盔佩戴情况在库表里面是以整型的方式记录的，节省空间且信息更加简洁，后端会对 label 和该字段的整型值做转换。由于在查询过程中，经常会对驾驶员信息和头盔佩戴情况这两个字段过滤，所以在设计的时候对这两个字段都创建了索引。

表 4-1 数据库表字段设计

字段	数据类型	说明	索引
id	int	主键 id	主键索引
driver	varchar	驾驶员	普通索引
detect_location	varchar	检测地区	普通索引
helmet	int	头盔佩戴情况	无
detect_time	datetime	检测时间	无

#### 4.4.2 检测模块

后端检测模块的用例设计如图 4-8。该模块的核心用例是检测头盔佩戴，其包含保存待测资源、检测头盔佩戴情况、检测驾驶员 id 和返回结果四个子用例。保存待测资源会将用户上传的图像或视频保存到本地磁盘；检测头盔佩戴用例将调用模型对上述资源进行预测；检测驾驶员 id 包含两个子用例，首先通过驾驶员 id 识别模型进行身份判定，随后将模型输出的检测结果持久化存储至数据库，最终封装结果数据返回给前端。

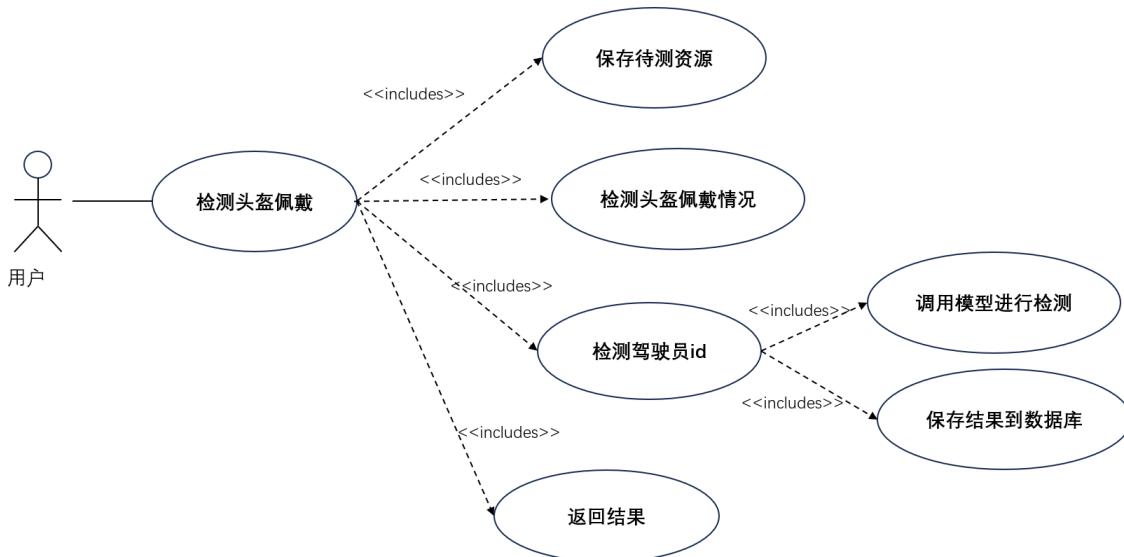


图 4-8 后端检测模块用例图

该模块执行顺序图如图 4-9。首先前端页面会上传待测资源到后端，后端将资源保存到本地磁盘，然后调用模型检测头盔佩戴情况，将检测结果保存到本地磁盘，再调用第二个模型检测驾驶员 id，将包含头盔佩戴情况、驾驶员以及检测时间等完整信息保存到数据库，最后封装结果返回给前端用户。

该模块类图如图 4-10 所示，主要由 DetectServiceImpl、ImageDTO 和 Detection 三个类构成。ImageDTO 类存储了用户输入的待检测资源和自定义检测模型、IOU 和置信度四个参数。DetectServiceImpl 的公共方法 detect 接收 ImageDTO 参数，其余私有方法分别

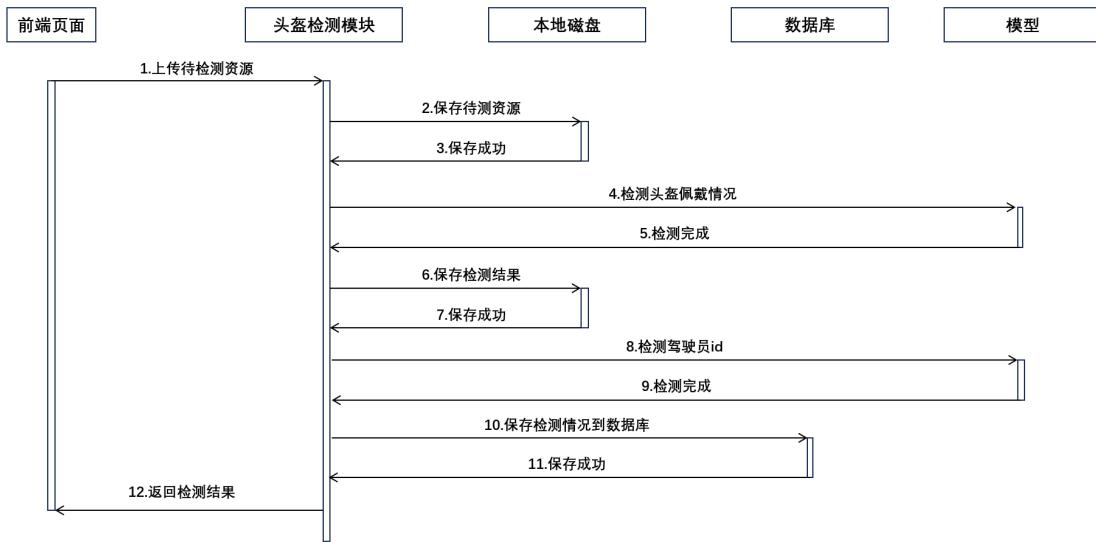


图 4-9 后端检测模块顺序图

负责保存待测资源、检测头盔佩戴、检测驾驶员 id、保存检测结果和构建返回结果，如果当前是视频资源，还会将格式从 avi 转换为 mp4。DetectServiceImpl 关联的 Detection 类作为 saveDetectResult() 方法的输入参数，负责保存检测结果，包括主键 id、驾驶员、头盔佩戴情况、检测地点和检测时间五个属性。

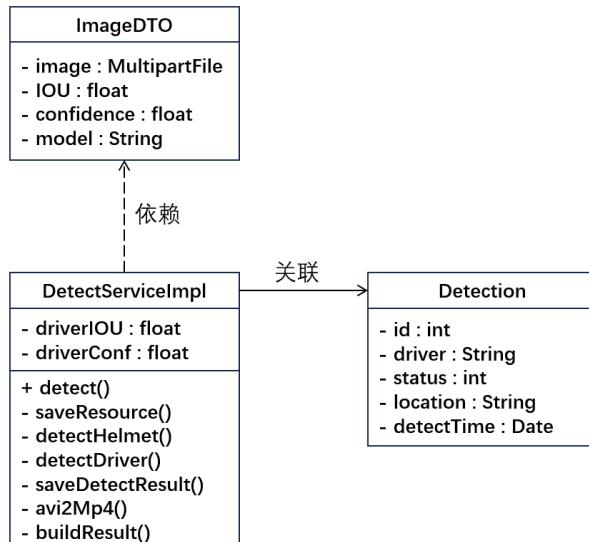


图 4-10 后端检测模块类图

#### 4.4.3 搜索模块

后端搜索模块的用例图如图 4-11。该模块的核心用例是查询历史记录。其包含查询数据库、构建可视化图表数据格式和返回结果三个子用例。查询数据库包含获取过滤条件和执行 sql 语句两个子用例，这一步可以将用户需要的数据从数据库查询出来。构建柱状图、折线图和饼图三种图表特定的数据格式，支持前端进行数据可视化。最后返回查询结果。

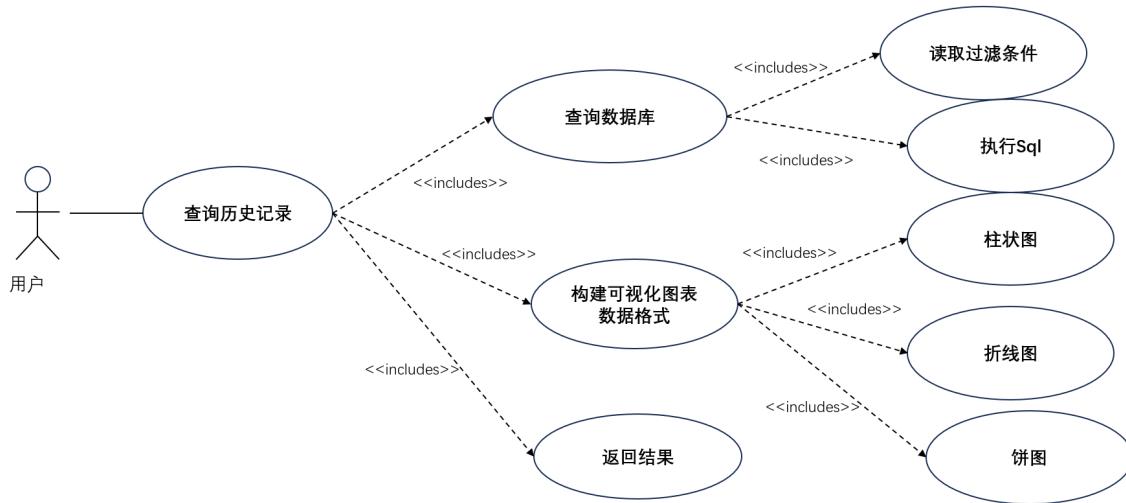


图 4-11 后端搜索模块用例图

图 4-12展示了后端搜索模块的操作顺序图。用户通过前端页面发起查询请求，该模块将过滤条件拼接到 sql 语句中去查询 Mysql 中的数据，根据 ECharts 文档执行数据格式转换操作，确保数据格式适配前端可视化图表。

搜索模块的类图见图 4-13，包含 SearchServiceImpl、SearchDTO 和 Detection 三个类。SearchDTO 作为 search 方法的输入参数，包含了用户自定义的过滤条件，包括对驾驶员、检测地点和检测时间这三个字段的过滤，对应用户在前端页面输入的过滤值。SearchServiceImpl 的 search 方法负责根据过滤条件从数据库中搜索历史检测记录，用 Detection 类接收返回值。transform 方法以查询出来的 Detection 结果集为输入，转换为可视化图表需要的特定数据格式。

### 4.5 本章小结

本章首先对基于 Vue 和 SpringBoot 框架的摩托车驾乘人员头盔佩戴检测系统进行需求分析，然后通过用例图、顺序图和类图介绍了前端和后端各个模块的设计思路与具体实现。

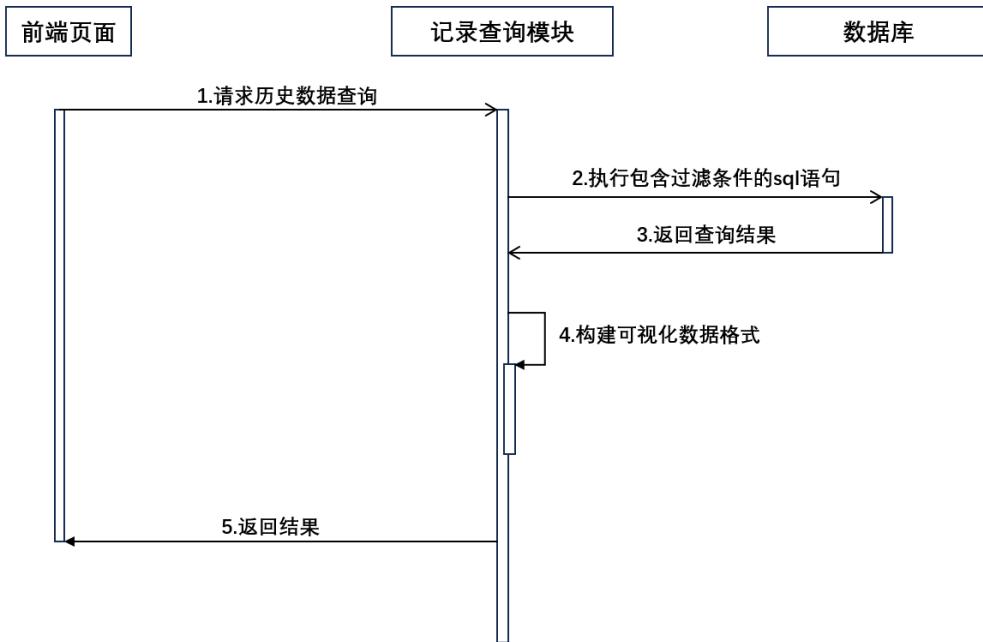


图 4-12 后端搜索模块顺序图

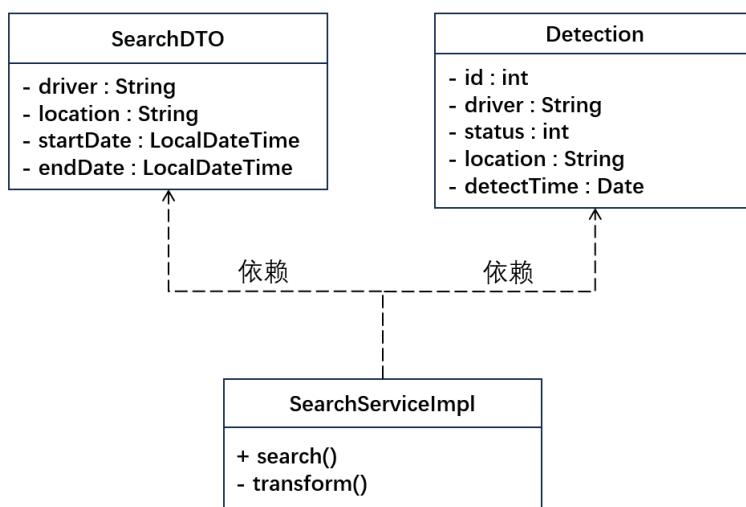


图 4-13 后端搜索模块类图

## 第5章 总结与展望

### 5.1 本文工作总结

本文基于 YOLOv11 算法，针对摩托车驾乘人员头盔佩戴数据集训练了两个模型：头盔佩戴情况检测模型和驾驶员检测模型，设计并实现了一套高效的检测系统，能够帮助执法人员减少人工工作量，推动交通执法向智能化方向发展。

在数据集构建过程中，对原始数据进行了细致处理，通过对原样本数据做图像增强保证每个类别和驾驶员都有足够数量的样本图像，采用 7:2:1 的比例对数据集进行随机划分，分别构建训练集、验证集和测试集。通过 YOLO 训练过程中的不同指标数据来对比不同模型的检测性能。

本文基于 YOLOv11n、YOLOv11s 和 YOLOv11m 三个不同权重的模型进行了训练并分析了结果。实验表明，三个模型在精度上都表现出色，均在 96% 以上，YOLOv11m 的精度最高，达到了 97.3%。召回率也都在 96% 以上，且也是 YOLOv11m 的召回率最高，为 98.1%。三个模型的 mAP50 值几乎一致，但 mAP50-95 指标 YOLOv11 和 YOLOv11m 最好，说明这两个模型在更严格的 IoU 阈值下性能更好。三个模型的检测速度从 YOLOv11n、YOLOv11s 到 YOLOv11m 依次变慢。YOLO 模型的检测精度和检测速度整体上是呈现负相关的，提升模型的检测精度会导致模型检测速度有下降。结合检测精度和检测速度两个方面各个指标来看，如果追求高检测速度，YOLOv11n 模型是比较好的选择；如果要求严格的检测精度，YOLOv11m 更合适，但是会牺牲一些检测速度；YOLOv11s 是介于以上两个模型之间比较适中的选择。

本文基于浏览器/服务器架构进行前后端分离的系统开发。前端为用户提供了目标检测和记录查询功能；后端则高效处理前端请求，完成图像或视频的检测、驾驶员信息识别以及结果存储等功能。同时，系统支持用户自定义检测模型、IoU 和置信度参数，历史检测结果可通过多种可视化图表呈现，满足了交管部门对数据的分析需求。

### 5.2 研究展望

随着科技的不断发展，目标检测技术在智能交通领域将迎来更广阔的发展空间。尽管 YOLOv11 模型在本文中表现优秀，但也存在着提升空间。

在模型性能方面，可以引入注意力机制、自监督学习等技术，增强模型对复杂场景和小目标的检测能力；针对头盔颜色与背景或衣服相近导致识别困难的问题，可扩充训练数据集，收集大量此类特殊样本，使模型学习对应特征模式，同时对现有数据进行颜色扰动增强，模拟多种相近情况，提升数据多样性。

在数据集方面，未来可收集更多不同场景下的摩托车驾乘人员数据，包括不同天气

条件、不同光照环境、不同地域的交通数据等，增强模型的泛化能力。根据本文的实验结果，可以看出 YOLO 模型在检测精度和速度上面呈现负相关性，在提升精度的同时，不可避免地会损失一些速度，反之亦然。随着深度学习技术的不断突破、硬件计算能力的不断提升，YOLO 有望实现速度和精度的协同突破，让 YOLO 模型既能在复杂场景中精准识别目标，又能以极快的速度完成实时检测任务。

在应用与落地方面，本文实现的摩托车驾乘人员头盔佩戴检测系统未来可广泛应用于智能交通监管，接入道路上的监控摄像头，实时检测驾乘人员的头盔佩戴情况，将违规驾驶员通知给执法部门，联动执法。

本文主要研究了摩托车驾乘人员头盔佩戴情况的检测系统，未来可将其拓展到其他领域，如汽车安全带佩戴检测、机动车违规行为检测等。同时，还可以与其他模块(如交通流量监测、违章行为自动抓拍等)进行深度融合，助力智能交通系统的发展。

## 参考文献

- 侯帅帅, 欧秀丽. 2023. 摩托车头盔产品质量安全风险分析. 质量与认证, (09): 91~92.
- 秦乐, 谭泽富, 雷国平, 陈秋伯. 2025. EMF-YOLO: 轻量化多尺度特征提取路面缺陷检测算法. 计算机工程与应用, 1~12.
- 袁婷婷, 赖惠成, 汤静雯, 张晞, 高古学. 2025. LMFI-YOLO: 复杂场景下的轻量化行人检测算法. 计算机工程与应用, 1~15.
- 张浩晨, 张竹林, 史瑞岩, 王文翰, 雷镇诺. 2025. YOLO-CDC: 优化改进 YOLOv8 的车辆目标检测算法. 计算机工程与应用, 1~15.
- Bochkovskiy A, Wang C.-Y, Liao H.-Y M. 2020. YOLOv4: Optimal Speed and Accuracy of Object Detection. *arXiv:2004.10934*,
- Dalal N, Triggs B. 2005. Histograms of Oriented Gradients for Human Detection. In: IEEE CVPR: 886–893.
- Felzenszwalb P F, Girshick R B, McAllester D. 2010. Cascade Object Detection with Deformable Part Models. In: IEEE CVPR: 2241–2248.
- Felzenszwalb P F, McAllester D, Ramanan D. 2008. A Discriminatively Trained, Multiscale, Deformable Part Model. In: IEEE CVPR: 1–8.
- Girshick R, Donahue J, Darrell T, Malik J. 2014. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition: 580–587.
- Girshick R B. 2015. Fast R-CNN. *CoRR*, abs/1504.08083.
- He K, Zhang X, Ren S, 0001 J S. 2014. Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. *CoRR*, abs/1406.4729.
- Krizhevsky A, Sutskever I, Hinton G E. 2012. ImageNet Classification with Deep Convolutional Neural Networks. *ImageNet Large Scale Visual Recognition Challenge*,
- Li C, Li L, Jiang H, Weng K, Wang Y. 2022. YOLOv6: A Single-Stage Object Detection Framework for Industrial Applications. Meituan (美团).
- Raza N, Habib M A, Ahmad M, Abbas Q, Aldajani M B, Latif M A. 2024. Efficient and Cost-Effective Vehicle Detection in Foggy Weather for Edge/Fog-Enabled Traffic Surveillance and Collision Avoidance Systems. *Computers, Materials & Continua*, 81(1): 911–931.
- Redmon J, Divvala S, Girshick R, Farhadi A. 2016. You Only Look Once: Unified, Real-Time Object Detection. In: IEEE CVPR: 779–788.
- Redmon J, Farhadi A. 2017. YOLO9000: Better, Faster, Stronger. In: IEEE CVPR: 6517–6525.
- Redmon J, Farhadi A. 2018. YOLOv3: An Incremental Improvement. *arXiv:1804.02767*,
- Ren S, He K, Girshick R, Sun J. 2015. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In: NeurIPS: 91–99.

- Uijlings J R R, Sande K E A, Gevers T, Smeulders A W M. 2013. Selective Search for Object Recognition. *International Journal of Computer Vision*, 104(2): 154–171.
- Viola P, Jones M. 2001. Rapid Object Detection using a Boosted Cascade of Simple Features. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR): vol. 1. IEEE Computer Society: 511–518.
- Wang C.-Y, Bochkovskiy A, Liao H.-Y M. 2022. YOLOv7: Trainable Bag-of-Freebies Sets New State-of-the-Art for Real-Time Object Detectors. *arXiv:2207.02696*,
- Wang C.-Y, Bochkovskiy A, Liao H.-Y M. 2024a. YOLOv10: Real-Time End-to-End Object Detection with Enhanced Efficiency-Accuracy Tradeoff. *arXiv:2405.14458*,
- Wang C.-Y, Bochkovskiy A, Liao H.-Y M. 2024b. YOLOv9: Learning What You Want to Learn Using Programmable Gradient Information. *arXiv:2402.13616*,
- Wang F, Yang X, Wei J. 2024. YOLO-ESL: An Enhanced Pedestrian Recognition Network Based on YOLO. *Applied Sciences*, 14(20): 9588–9588.

## 致 谢

行文至此，我的论文已经接近尾声，也意味着我在西北农林科技大学四年的校园生活将要结束，即将踏上新的旅途。回首这段时光，心中满是怀念与感慨，有太多人需要感谢。

感谢我的导师耿耀君老师在毕业设计上对我的悉心指导。从论文的选题、开题到初稿、修改，再到最终定稿，导师始终给予我悉心的指导和耐心的帮助，提供宝贵的建议。耿耀君老师严谨的治学态度和敬业的工作精神，让我不仅在学术上有所收获，更在为人处世方面深受教诲。

感谢我的大学室友、同学和所有朋友，在我因学业压力陷入情绪低谷的时候帮我排忧解难；在我因面试失败怀疑自己的时候帮我重振信心；在我因感冒发烧身体不适的时候帮我买药带饭。因为有你们，我的大学生活更加值得怀念。

感谢父亲、母亲、爷爷、奶奶和其他家人，在我学习和求职路上的鼓励，以及在生活上无微不至的关怀。每一份支持都让我更加有动力，朝着更好的未来前进。

最后感谢各位评审老师在百忙之中审阅论文并提出宝贵意见。

姜明宇

二〇二五年五月于 杨凌