

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：随着摩托车保有量不断增加，摩托车交通事故频发，头盔佩戴与否直接关系到驾乘人员在事故中的伤亡程度。本系统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，并对检测结果实现数据可视化，减少人工介入，提高执法效率。

在核心算法层面，本系统基于 YOLO 目标识别算法，构建了双模型协同架构：其一，专注于快速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；其二，以前者输出的关键区域为输入，精准定位并识别驾驶人员身份。研究过程中，系统对比分析了 YOLOv11n、YOLOv11s 等不同权重模型的训练效果，通过不断优化参数，提升了头盔检测的精度与稳定性。

在系统设计上，通过 SpringBoot 搭建后端服务器，Vue 搭建前端 web 页面，支持用户自主选定模型，按需调节交并比（IoU）与置信度，以实现个性化的目标检测。历史检测结果可通过饼图、柱状图等丰富多样的可视化图表直观呈现，为交通管理部门提供清晰、便捷的数据分析手段，有效助力提升执法效能与交通安全管理水。

关键词：目标检测；YOLO；头盔佩戴检测；SpringBoot；Vue

Design and implementation of helmet wearing detection system for motorcycle riders

Abstract: With the increasing number of motorcycles, motorcycle traffic accidents occur frequently. Whether the helmet is worn or not is directly related to the degree of casualties of drivers and passengers in the accident. The purpose of this system is to design and implement a helmet wearing detection system for motorcycle drivers and passengers, which can detect the helmet wearing status of drivers and passengers in real time, visualize the detection results, reduce manual intervention, and improve law enforcement efficiency.

At the core algorithm level, the system builds a dual model collaborative architecture based on the Yolo target recognition algorithm: first, it focuses on quickly identifying the overall helmet wearing status of motorcycles and their riders in images or videos; Second, the key area of the former output is the input, which can accurately locate and identify the driver's identity. In the research process, the training effects of different weight models such as yolov11n and yolov11s were systematically compared and analyzed. Through continuous optimization of parameters, the accuracy and stability of helmet detection were improved.

In the system design, springboot is used to build the back-end server, Vue is used to build the front-end web page, which supports users to independently select the model, and adjust the intersection union ratio (IOU) and confidence as needed to achieve personalized target detection. The historical detection results can be visually presented through a variety of visual charts such as pie charts and histogram charts, providing clear and convenient data analysis means for traffic management departments, and effectively helping to improve law enforcement efficiency and traffic safety management level.

Keywords: Object Detection; YOLO; Helmet wearing detection; SpringBoot; Vue

目 录

| | |
|---------------------------------|----|
| 第 1 章 绪论 | 1 |
| 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 |
| 1.3 YOLO 在交通安全领域的应用 | 5 |
| 1.4 研究与设计内容 | 5 |
| 1.5 章节安排 | 6 |
| 第 2 章 YOLO 算法相关理论 | 7 |
| 2.1 YOLOv1 网络结构 | 7 |
| 2.1.1 骨干网络 | 7 |
| 2.1.2 颈部网络 | 9 |
| 2.1.3 检测头 | 9 |
| 2.2 YOLOv1 损失函数 | 9 |
| 2.2.1 边界框回归损失 | 9 |
| 2.2.2 分类损失 | 10 |
| 2.2.3 分布损失 | 10 |
| 2.3 YOLO 各版本性能对比 | 11 |
| 2.4 本章小结 | 11 |
| 第 3 章 数据集构建及训练参数设置 | 12 |
| 3.1 数据集构建 | 12 |
| 3.2 实验环境 | 13 |
| 3.3 参数设置 | 14 |
| 3.3.1 基本参数 | 14 |
| 3.3.2 超参数 | 14 |
| 3.4 本章小结 | 14 |
| 第 4 章 实验结果与分析 | 15 |
| 4.1 评价指标 | 15 |
| 4.1.1 精度评价指标 | 15 |
| 4.1.2 速度评价指标 | 16 |

| | |
|-------------------------------|-----------|
| 4.2 实验结果分析 | 17 |
| 4.2.1 YOLOv11n | 17 |
| 4.2.2 YOLOv11s | 18 |
| 4.2.3 YOLOv11m | 20 |
| 4.2.4 模型性能对比 | 21 |
| 4.3 本章小结 | 22 |
| 第 5 章 检测系统的设计与实现 | 23 |
| 5.1 需求分析 | 23 |
| 5.2 系统整体架构 | 23 |
| 5.3 前端模块设计与实现 | 24 |
| 5.3.1 目标检测界面 | 24 |
| 5.3.2 记录查询界面 | 26 |
| 5.4 后端模块设计与实现 | 26 |
| 5.4.1 库表结构 | 27 |
| 5.4.2 检测模块 | 27 |
| 5.4.3 搜索模块 | 28 |
| 5.5 本章小结 | 28 |
| 第 6 章 总结与展望 | 29 |
| 6.1 本文工作总结 | 29 |
| 6.2 研究展望 | 29 |
| 参考文献 | 30 |
| 致 谢 | 32 |

第1章 绪论

1.1 研究目的与意义

在城市化进程加速和生活节奏日益加快的当下，外卖配送、即时出行等需求呈爆发式增长，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优质选择。这一市场需求的增长，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。**图 1-1**展示了某城市道路上摩托车驾乘人员的头盔佩戴情况。

随着摩托车数量的急剧增长，涉及摩托车的交通事故也逐渐增多。头盔作为摩托车驾乘人员唯一的保护装置，能够极大减少交通事故给驾乘人员带来的伤害，尽可能地保护驾驶员和乘客的生命安全(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。

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

本文基于 YOLOv11 目标识别算法，训练了能够检测头盔佩戴情况的模型，该模型能够实时识别图片或视频中摩托车驾乘人员是否佩戴头盔。此外，通过 Vue 和 SpringBoot 框架搭建系统，为用户提供图形界面化的操作平台，将检测结果持久化到数据库，以便后续查询历史检测记录以及进行数据的可视化。

1.2 目标检测发展历程

目标检测是计算机视觉领域的核心任务之一，它的发展历程如图 1-2 所示。

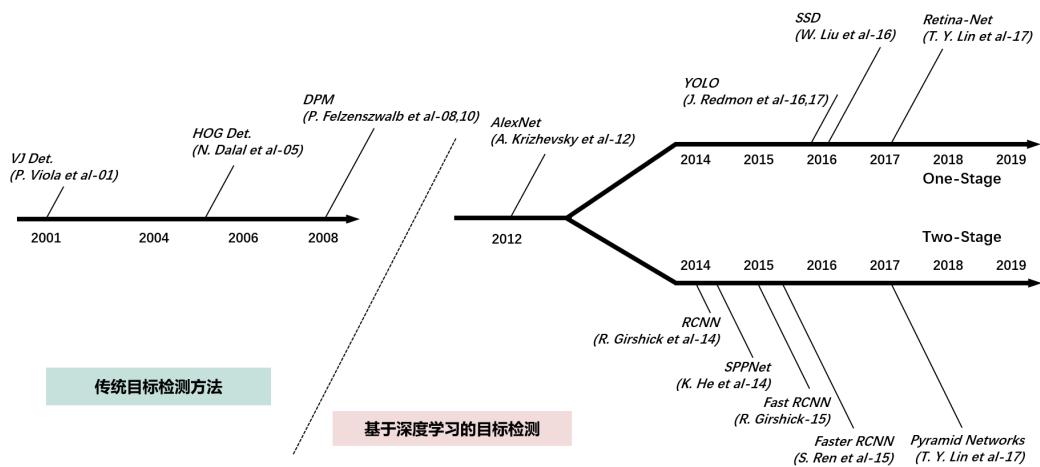


图 1-2 目标检测算法发展历程

1.2.1 传统目标检测

在深度学习兴起之前，目标检测算法高度依赖人工设计特征。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。相较于同期算法，Viola-Jones(VJ) 检测器在保持同等检测精度的前提下，运算速度实现了数十倍乃至数百倍的提升。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。VJ 检测器融合了“积分图像”、“特征选择”以及“检测级联”技术，大幅提升了检测效率。

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 描述器被设计为在密集的均匀间隔单元网格(区块)上计算，并使用重叠局部对比度归一化方法来提高精度。HOG

描述器的主要目标是行人检测，如若要检测不同大小的对象，则需要让 HOG 检测器在保持检测窗口大小不变的情况下，对输入图像进行多次重设尺寸。

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)，作为 HOG 检测器的延伸版本，后续经 R. Girshick 等人不断优化改进(Felzenszwalb et al. 2010)。DPM 采用“分而治之”的思想，通过检测对象的各个部件实现目标识别。

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，在 ImageNet 大规模视觉识别挑战赛中以压倒性优势刷新了记录，它的出现对深度学习发展具有里程碑式的意义。基于深度学习的目标检测算法主要分两类：基于回归的 One-Stage 与基于候选区域的 Two-Stage。

(1) R-CNN 系列算法

2014 年，Girshick et al. (2014) 提出的 R-CNN 的思路为：首先通过选择性搜索(Uijlings et al. 2013) 算法来提取可能包含目标的候选框，并将候选框调整成固定大小，然后通过 AlexNet 进行特征提取，最后利用 SVM 分类器识别每个区域内的目标。尽管 R-CNN 在目标检测领域实现了突破性进展，但其局限性也较为突出。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，实现对任意尺寸图像生成固定长度特征表示，将检测速度提升至 R-CNN 的 20 倍以上。

2015 年，Girshick (2015) 提出 Fast R-CNN 检测器，作为对 R-CNN 和 SPPNet 的进阶优化成果，其创新地实现了在同一网络配置下同步训练检测器与边界框回归器，降低了训练和推理时间，大大提升了模型的性能。

Ren et al. (2015) 提出的 Faster R-CNN 检测器是第一个端到端的，也是第一个接近实时的深度学习检测器。它引入了 RPN(Region Proposal Network) 来提升检测速度和性能。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) 提出的 YOLO 算法，是 One-Stage 算法及 YOLO 系列的开山之作。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。

选，经阈值处理后输出高置信度检测结果。YOLOv1 将输入图像划分为 $n \times n$ 网格，每个网格单元负责预测内部目标的边界框及类别信息，每个边界框均附带基于交并比 (IOU) 计算的置信度分数，用于评估框内存在目标的可能性。

在 YOLOv1 问世一年后，YOLOv2 应运而生(Redmon and Farhadi 2017)。该模型因能够检测超过 9000 个不同对象，也被称作 YOLO9000。YOLOv2 的核心是提出了一种独特的联合训练算法，该算法能够同时利用检测数据与分类数据对目标检测器进行训练。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 通过多尺度预测架构 (三尺度特征金字塔) 优化跨尺寸目标检测，并引入 Darknet-53 骨干网络，在 COCO 数据集上实现精度-速度双优平衡(Redmon and Farhadi 2018)。其采用独立 Logistic 分类器替代 Softmax，支持多标签分类，同时修正了 YOLOv2 的底层数据加载缺陷。

2020 年，YOLOv4 的主干网络采用 CSPDarknet53 骨干网络，融合了跨阶段局部连接 (CSPNet) 策略。颈部网络集成改进型空间金字塔池化 (SPP) 与路径聚合网络 (PAN)，实现多尺度特征融合与跨层级信息增强(Bochkovskiy et al. 2020)。YOLOv5 采用基于 PyTorch 的模块化架构重构检测框架，在提升了速度和精度的同时，使网络结构更加轻量级。

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 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了突破性进展。其核心架构采用 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；特征融合模块则重构为 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。同年提出的 YOLOv7 通过架构创新与动态训练机制的协同设计，在实时性与检测精度间实现突破性平衡(Wang et al. 2022)。其采用 E-ELAN 扩展型主干网络，通过多分支深度扩展与特征通道重组增强多尺度表征能力。YOLOv8 是 YOLO 系列实时对象检测器的迭代版本，在准确性和速度方面提供尖端性能。YOLOv8 由 Ultralytics 开发，引入了新功能和优化，使其成为各种应用中各种对象检测任务的理想选择。

2024 年提出的 YOLOv9 通过可编程梯度信息 (PGI) 重构梯度传播路径，利用辅助可逆分支生成可靠梯度，结合广义高效层聚合网络 (GELAN) 融合 CSPNet 与 ELAN 架构优势，显著缓解深度网络的信息衰减问题(Wang et al. 2024b)。同年提出的 YOLOv10 通过一致双分配策略实现无 NMS 训练，提升性能和推理效率；采用整体效率-精度驱动的模型设计策略，全面优化模型组件，降低计算开销、增强模型能力(Wang et al. 2024a)。同年晚些时候的 YOLOv11 进一步革新特征提取机制，引入 C3k2 轻量化卷积块替代传统 C2f 结构；并设计 C2PSA 跨阶段空间注意力模块强化遮挡与小目标检测能力，成为迄今最高效的 YOLO 迭代版本。YOLOv11 的网络架构将在下一章进行详细介绍。

1.3 YOLO 在交通安全领域的应用

YOLO(You Only Look Once) 是一系列目标检测算法，它基于深度学习技术，将目标检测问题转化为回归问题进行处理，这一独特的设计思路使得它在速度和精度方面都取得了很好的平衡，在交通安全领域得到了非常广泛的实际应用。

2025 年，张浩晨等人基于 YOLOv8 算法，首先提出了结合 Transformer 结构全局特征提取能力的模块 C2Former 代替 C2f 模块，设计了重感知的门控线性单元 GRMLP 优化 Transformer 分支非线性表达能力，提升了在小目标、遮挡目标等场景下对交通车辆检测的精度(张浩晨 等 2025)。肖振久等人设计动态特征融合网络，利用多样化卷积和通道缩放降低模型复杂度，并结合 SPDConv 重构颈部网络，增强小目标边缘信息提取的能力，在平衡检测精度的前提下提出了 DEL-YOLO 轻量化安全帽佩戴检测算法(肖振久 等 2025)。Raza et al. (2024) 针对雾天场景下的车辆检测需求，基于 DAWN 和 FD 数据集构建四类车辆标注，并对比了 YOLO-V5/V8 系列模型的性能。通过引入注意力模块 (CBAM、NAM、SimAM) 和 BiFPN 结构优化 YOLO-V5s/V5l，优化了算法在雾天中对车辆的检测性能。

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。

1.4 研究与设计内容

本文基于 YOLOv11 算法设计并实现了摩托车驾乘人员头盔佩戴检测系统，主要涉及模型训练、系统开发两个任务。

在模型训练这里，做了两方面工作。一方面，在数据标注环节，进行了细致且全面的处理。针对不同情况设置了多种标签，例如单一驾驶人佩戴、单一驾驶人未佩戴、驾驶人佩戴一位乘客未佩戴、驾驶人佩戴一位乘客佩戴等，并没有简单地将所有情况划分为佩戴和不佩戴两种，这样在检测时能为用户提供更详细的信息，让用户清楚了解当前驾驶人及乘客各自的头盔佩戴状况。另一方面，训练了两个关键模型，一个用于识别摩托车及驾乘人员整体头盔佩戴情况，另一个用于识别摩托车驾驶人员是谁。在训练过程中，使用 YOLOv11n、YOLOv11s、YOLOv11m 等不同模型进行尝试，并通过调整训练 epoch 来优化模型精度。

系统基于 BS 架构开发。B 端页面为用户提供了两个操作页面，一个是用于上传图片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 端负责处理 B 端页面传来的请求，当

接收到用户上传的图片或视频后，先利用第一个模型预测图片中的头盔佩戴情况，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，最后将检测结果保存到数据库中。**图 1-3**和**图 1-4**表示了输入图片和检测结果。



图 1-3 选择图片



图 1-4 检测结果

1.5 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。本章主要介绍了 YOLOv11 算法的网络结构和损失函数。网络结构方面主要介绍主干网络、颈部网络和检测头。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。最后对 YOLO 系列各版本的性能作了对比。

第三章：数据集构建及训练参数设置。本章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，然后介绍了本文的实验环境和训练参数设置。

第四章：实验结果与分析。本章首先介绍了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精度、召回率、mAP 以及检测速度，总结了各模型适用的场景。

第五章：检测系统的设计与实现。本章主要介绍了检测系统的软件设计与开发过程。从需求分析、系统架构、前端开发和后端开发这四个方面展开。

第六章：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 算法相关理论

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。通过模块化的结构创新与技术融合，在兼顾检测速度的同时显著提升了目标识别的精度与鲁棒性。骨干网络作为特征提取的基石，通过新型卷积模块与网络结构设计，实现了高效的底层特征提取；颈部网络采用改进的特征融合机制，将多尺度特征进行精细化处理与整合；检测头则依托优化的分类回归策略，实现了对目标位置、类别和置信度的精准预测。图 2-1 展示了 YOLOv11 的网络结构。

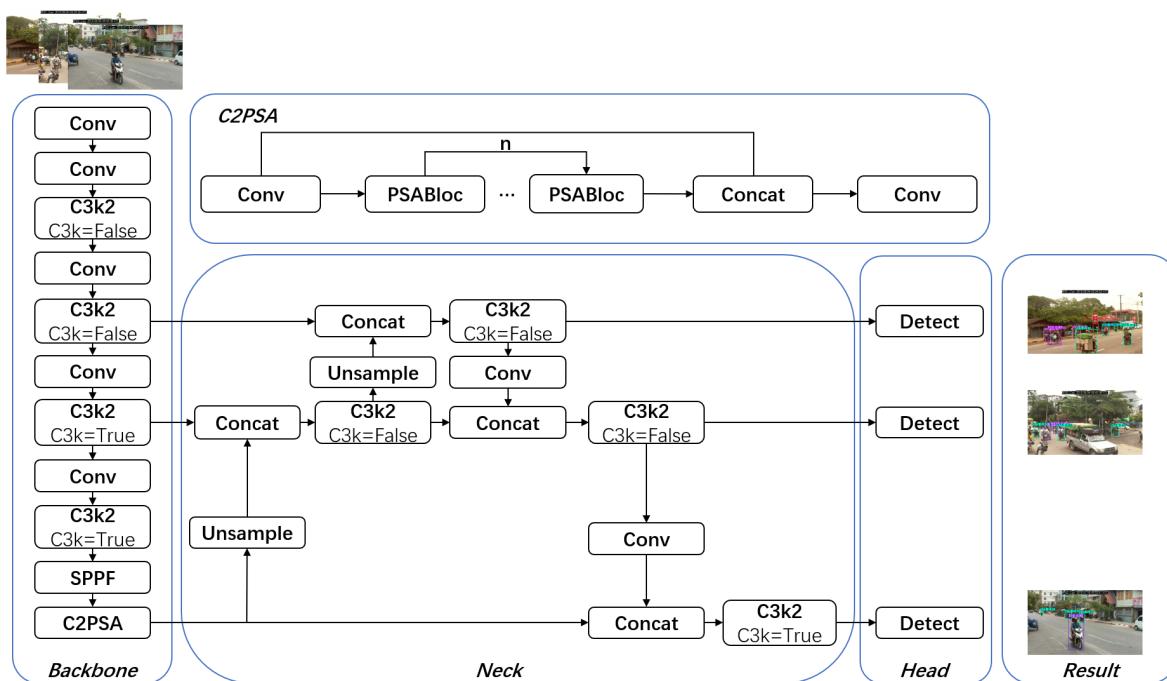


图 2-1 YOLOv11 网络模型

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 模块即卷积模块，它是深度学习里基础且关键的模块，在目标检测模型里起到特征提取的作用。在 YOLO 系列模型中，Conv 模块一般由卷积层、批量归一化层和激活函数层构成。卷积层借助不同的卷积核来提取输入特征图的不同特征，像边缘、纹理等。批量归一化层能加速模型的收敛速度，让模型训练更稳定。激活函数层为模型引入非线性因素，增强模型的表达能力。

YOLOv11 使用 C3K2 模块来处理骨干网络不同阶段的特征提取。较小的 3×3 内核可实现更高效的计算，同时保留模型在图像中捕获基本特征的能力。YOLOv11 骨干网络的核心是 C3K2 模块，它是早期版本中引入的 CSP(Cross Stage Partial) Bottleneck 的演变。C3K2 模块通过分割特征图，并应用一系列较小的 (3×3) 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。通过处理更小的单独特征图并在多次卷积后合并它们，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。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。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。

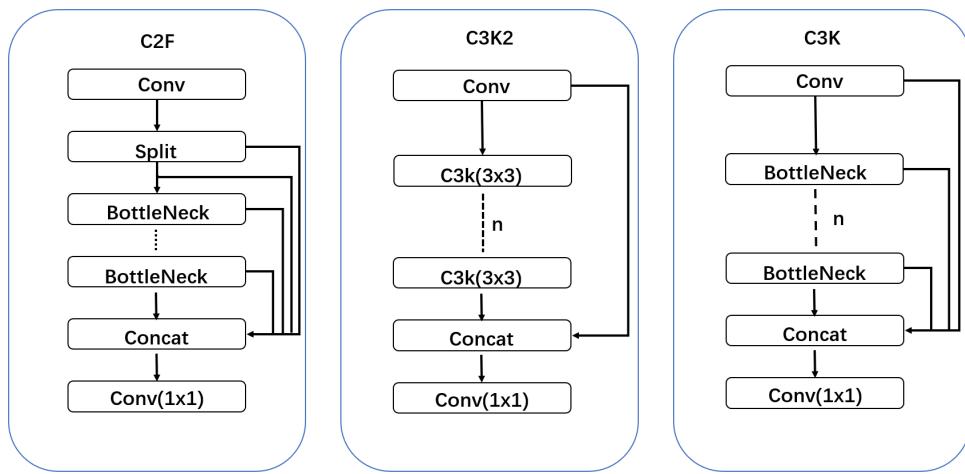


图 2-2 C2F 和 C3K2 模块的比较

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。该模块借助不同大小的最大池化核来提取不同尺度的特征，然后将这些特征进行拼接，从而使模型能捕捉到不同尺度的目标信息。通过融合多尺度特征，模型能够更好地检测出不同大小的目标，进而提高检测精度。

YOLOv11 的重大创新之一是增加了 C2PSA 块。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，经一系列处理提升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。

2.1.2 颈部网络

颈部网络由多个卷积层、C3K2 模块、Concat 操作和上采样模块组成，并结合了 C2PSA 机制的优势。颈部网络的主要作用是聚合不同尺度的特征，并将其传递给检测头。

颈部网络 Conv 模块利用卷积核对输入特征图进行卷积计算，进一步提取和变换特征。C3K2 模块使用了不同大小的卷积核，在处理复杂场景时能显著提高特征提取的精度。Concat 模块沿着通道维度将多个特征图合并在一起，可以融合不同尺度或不同来源的特征信息，增加特征的丰富度。上采样模块通常采用插值等方法增大特征图的尺寸，使特征图尺寸与其他分支匹配，方便后续的特征融合操作，让模型能结合不同尺度的特征进行检测。

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粒度级别。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 的损失函数通常由三部分构成：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)，用于优化预测边界框与真实边界框之间的差异。

$$\begin{aligned} \text{Box Loss} = & \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{obj}} \left((x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right) \\ & + \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{obj}} \left((\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2 \right) \end{aligned} \quad (2-1)$$

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

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

式(2-1)中, 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。 λ_{coord} 是权重系数, 用于平衡不同部分的损失。

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, 促使预测中心尽可能接近真实中心。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。

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

S 是网格的大小。 1_i^{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

在目标检测中, 经常会出现类别不平衡的问题, 即其中某些类别的样本数量远多于其他类别, 这有可能导致模型对常见类别的学习效果很好, 但对少数类别的学习效果略差。分布损失函数的主要目的是解决目标检测中类别不平衡的问题, 增强模型对困难样本的学习能力, 并提升模型对少数类别检测性能。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。 α 是平衡因子, 用于调整正负样本之间的权重。 γ 是聚焦参数, 用于控制对困难样本的关注程度。

2.3 YOLO 各版本性能对比

YOLOv11 与 YOLOv9、YOLOv8 等版本的性能对比见表 2-1。mAP 50-95 这个指标衡量的是模型在 IoU 值从 0.5 到 0.95 范围内的平均精度,它的评价范围更加广泛。Speed CPU ONNX 表示模型在 CPU 上基于 ONNX 推理时,处理一张图像所花费的时间,时间越短,说明模型在 CPU 上的推理效率越高,能够更快地给出检测结果。Speed T4 TensorRT10 指模型在 NVIDIA T4 TensorRT10 硬件平台上进行推理时,处理一张图像所需的时间。Params 表示模型的参数量,反映了模型的规模和复杂度,参数量越多,模型可学习的特征表示能力可能越强。FLOPs 表示浮点运算量,它衡量了模型在进行一次前向传播(推理)过程中所执行的浮点运算次数,能体现出模型计算的复杂度。

从下表可以看出,对于同一版本的 YOLO 模型,规模越大的模型检测精度越高,但是速度会有所下降。对于同一对规模的 YOLO 模型,新版本的模型要比旧版本的模型检测精度高。YOLOv8 相较于 YOLOv5 的同一规模的模型,检测速度更慢,但到了 YOLOv11 这里,同一规模的模型相较于之前的版本,检测速度都更快。即最新一代的 YOLOv11,在提升了检测精度的同时,也加快了检测速度。

表 2-1 YOLO 各版本模型性能对比

| Model | Size(pixels) | mAPval 50-95 | Speed CPU ONNX(ms) | Speed T4 TensorRT10(ms) | Params (M) | FLOPs(B) |
|----------|--------------|--------------|-----------------------|----------------------------|---------------|----------|
| YOLOv5n | 640 | 34.3 | 73.6 | 1.06 | 2.6 | 7.7 |
| YOLOv5s | 640 | 43.0 | 120.7 | 1.27 | 9.1 | 24 |
| YOLOv8n | 640 | 37.3 | 80.4 | 0.99 | 3.2 | 8.7 |
| YOLOv8s | 640 | 44.9 | 128.4 | 1.2 | 11.2 | 28.6 |
| YOLOv9n | 640 | 38.3 | NA | NA | 2.0 | 7.7 |
| YOLOv9s | 640 | 46.8 | NA | NA | 7.2 | 26.7 |
| YOLOv11n | 640 | 39.5 | 56.1 | 1.5 | 2.6 | 6.5 |
| YOLOv11s | 640 | 47.0 | 90.0 | 2.5 | 9.4 | 21.5 |

2.4 本章小结

本章围绕 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。并对 YOLOv11 中三个重要的损失函数做了详细的分析,最后展示了 YOLO 各版本性能对比, YOLOv11 在同等规模下实现检测精度与速度的双重提升。

第3章 数据集构建及训练参数设置

3.1 数据集构建

本文所使用的摩托车驾乘人员头盔佩戴情况数据集来源于某国家真实的交通道路照片。本文首先使用 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 3-1 所示，文件的每一行有五个元素， class 代表当前目标的类别， x_center(y_center) 代表边界框中心点的 x(y) 坐标，相对于图像宽度(高度)的归一化值， width(height) 代表边界框的宽度(高度)相对于图像宽度(高度)的归一化值。上述五元组能够表示某一个类别的目标在图像中的位置。

表 3-1 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 |

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片，由于其中某些类别的样本数据非常少，对包含这些标签的原图片进行了图像增强，保证每一个类别至少有 100 张样本图片。具体的增强方法如图 3-1 所示。

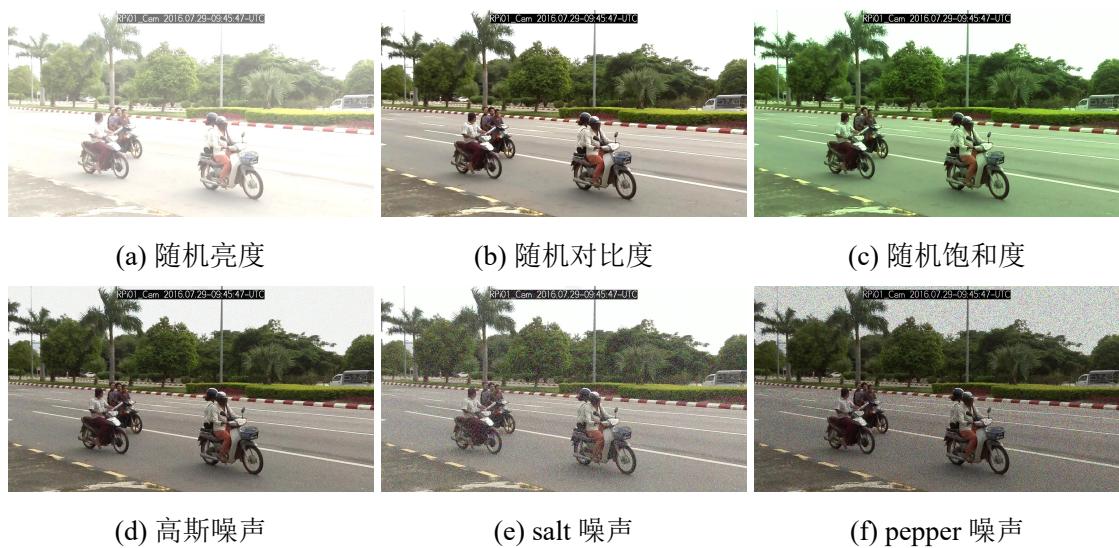


图 3-1 图像增强方式

最后得到了 6282 张图片，以 7:2:1 的比例，随机将图片划分为训练集、验证集和测试集。本文将目标类别分为 18 类，具体类别以及含义如表 3-2 所示。每一个类别的

样本数量分布如图3-2所示。现对表3-2中meaning一列做解释：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DNоХelmetP0NoHelmetP1NoHelmet，表示摩托车驾驶员未佩戴头盔，驾驶员身前的乘车人员未佩戴头盔，驾驶员身后的一名乘车人员未佩戴头盔。

对原数据集5661张图片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片，共有570个不同驾驶员。由于驾驶员的图片样本也存在不平衡的情况，对这33571张图片同样进行图像增强，保证每一个驾驶员都有100张样本图片。

表3-2 目标类别标签含义

| class | label | meaning |
|-------|-------|---|
| 0 | S01 | DHelmetP1NoHelmetP2NoHelmet |
| 1 | S02 | DNоХelmetP1NoHelmetP2NoHelmet |
| 2 | S03 | DHelmetP0NoHelmetP1NoHelmet |
| 3 | S04 | DNоХelmetP1NoHelmet |
| 4 | S05 | DHelmetP0NoHelmet |
| 5 | S06 | DNоХelmet |
| 6 | S07 | DNоХelmetP1NoHelmetP2Helmet |
| 7 | S08 | DHelmetP1NoHelmet |
| 8 | S09 | DNоХelmetP0NoHelmetP1NoHelmet |
| 9 | S10 | DNоХelmetP1Helmet |
| 10 | S11 | DHelmetP0NoHelmetP1Helmet |
| 11 | S12 | DNоХelmetP0NoHelmetP1NoHelmetP2NoHelmet |
| 12 | S13 | DHelmetP1NoHelmetP2Helmet |
| 13 | S14 | DNоХelmetP0NoHelmet |
| 14 | S15 | DHelmet |
| 15 | S16 | DHelmetP1Helmet |
| 16 | S17 | DHelmetP0Helmet |
| 17 | S18 | DHelmetP1HelmetP2Helmet |

3.2 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

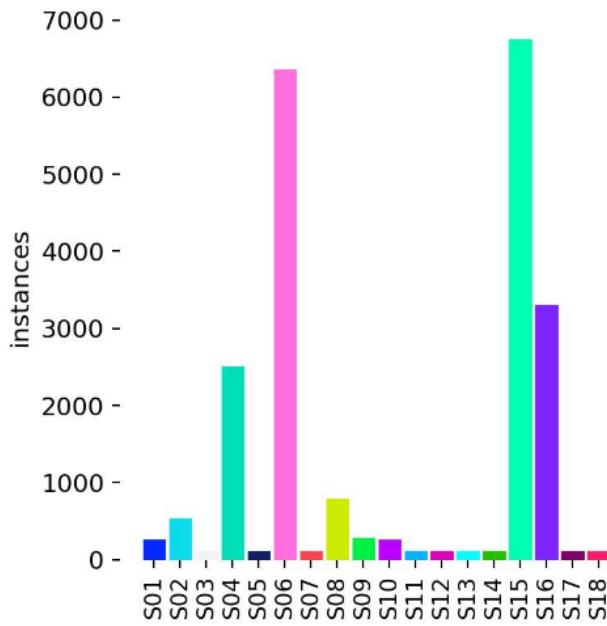


图 3-2 头盔佩戴情况标签数量分布

Pytorch 版本：2.7.0+cu126

3.3 参数设置

3.3.1 基本参数

本文基于 YOLOv11 进行模型训练，分别使用 YOLOv11n.pt、YOLOv11s.pt、YOLOv11m.pt 进行训练。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。

3.3.2 超参数

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。

3.4 本章小结

本章主要介绍了原数据集的状况以及后续处理过程，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，说明了本文将基于 YOLOv11 的三个不同量级的模型进行训练，比较各自的性能。

第4章 实验结果与分析

4.1 评价指标

4.1.1 精度评价指标

评价目标检测模型精度的指标有交并比 (IoU)、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) 等。

IoU 用来表示预测框的准确率，它在训练阶段反映的是标注框与预测框的重合程度。例如，用 A 表示预测目标框，用 B 表示真实目标框，IoU 则是 A、B 两个框的交集与并集的比值。IoU 值越大，代表 A、B 两个框重叠程度越高。IoU 要搭配 IoU 阈值一起使用。IoU 阈值默认被设置为 0.5，当两框的 IoU 大于阈值时，则判断预测框预测正确。

在深度学习中，将分类任务的预测结果分为以下四种，被称作混淆矩阵，见图 4-1：第一种是真正例 (True Positive, TP)，即模型预测为正例，且实际标签也是正例，预测正确；第二种是假负例 (False Negative, FN)，指模型预测为负例，但真实标签却是正例，预测错误；第三种是假正例 (False Positive, FP)，表示模型将负例误判为正例，同样是预测错误；最后一种是真负例 (True Negative, TN)，即模型预测为负例，同时实际标签也为负例，预测正确。这里的正例和负例其实只是针对某一类别而言的，针对 A 类而言，A 类别就是正例，其他类别就是负例。精度 (Precision) 的定义如式 (4-1)，该指标数值越大，表明误判为正例的数量 (即 FP 值) 越少，反映出模型预测结果中真正正例的占比更高，预测的准确性更好。Precision 越高，误检越少。召回率 (Recall) 的定义如式 (4-2)，其数值的越大，代表着漏判为负例的数量 (即 FN 值) 越少，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。

$$\text{Precision} = \frac{TP}{TP + FP} \quad (4-1)$$

$$\text{Recall} = \frac{TP}{TP + FN} \quad (4-2)$$

P-R 曲线即为分别以 Precision 与 Recall 为坐标围成的曲线。如图 4-2 所示，这是本文实验产生的一个 P-R 曲线。平均精度 (AP) 通过计算各类别 P-R 曲线与坐标轴所围成区域的面积来衡量模型性能。从几何意义上讲，若模型的 AP 值越高，意味着其 P-R 曲线与坐标轴所围合的区域面积越大，直观反映出该模型在精度 (Precision) 与召回率 (Recall) 两个关键指标上的综合表现更好，在整体数据预测中能够同时实现较高的正例预测纯度与正例捕捉能力。平均精度均值 (mAP) 是对所有类别的 AP 求平均值，用于反映整个模型的准确率。在 YOLO 模型中，常见的 mAP 评价指标有 mAP50 和 mAP50-95。mAP50 为

| | | Predict | |
|-----------|----------|--------------------|--------------------|
| | | Positive | Negative |
| Reference | Positive | True Positive(TP) | False Negative(FN) |
| | Negative | False Positive(FP) | True Negative(TN) |

图 4-1 混淆矩阵

IoU 阈值设为 0.5 时，所有类别的平均精度均值。mAP50-95 为 IoU 阈值从 0.5 到 0.95(间隔 0.05 递增) 的多个严格标准下，分别计算 mAP，再取平均值。

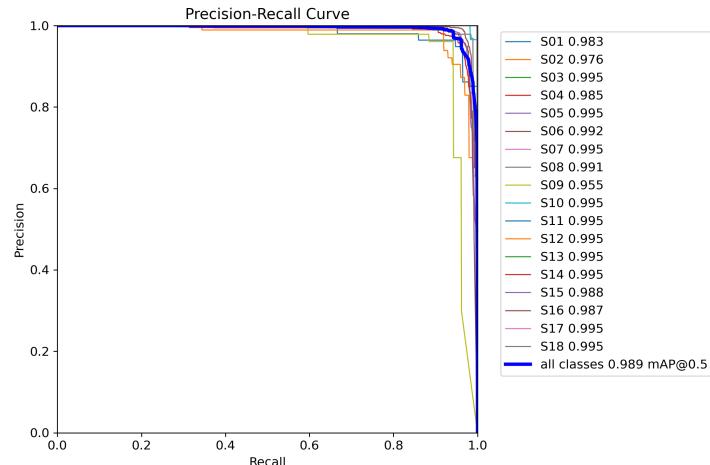


图 4-2 PR 曲线

4.1.2 速度评价指标

评价目标检测模型速度的指标有：每秒帧率 (Frame Per Second, FPS)，即每秒内可以处理的图片数量，能够衡量模型的实时性，FPS 越高，实时性越强。延迟 (Latency)，表示单张图像从输入到输出结果的总耗时。FLOPs 表示浮点运算量，它衡量了模型在进行

一次前向传播(推理)过程中所执行的浮点运算次数, FLOPs 越低, 理论速度越快, 但并不直接等于实际速度。

4.2 实验结果分析

4.2.1 YOLOv11n

本文基于 YOLOv11n 训练了 300 个 epoch, 训练结果如图 4-3 所示。从实验结果可知, 模型的最高精度为 96.7%, 最高召回率为 96.6%, mAP50 为 99.0%, mAP50-95 为 94.1%。

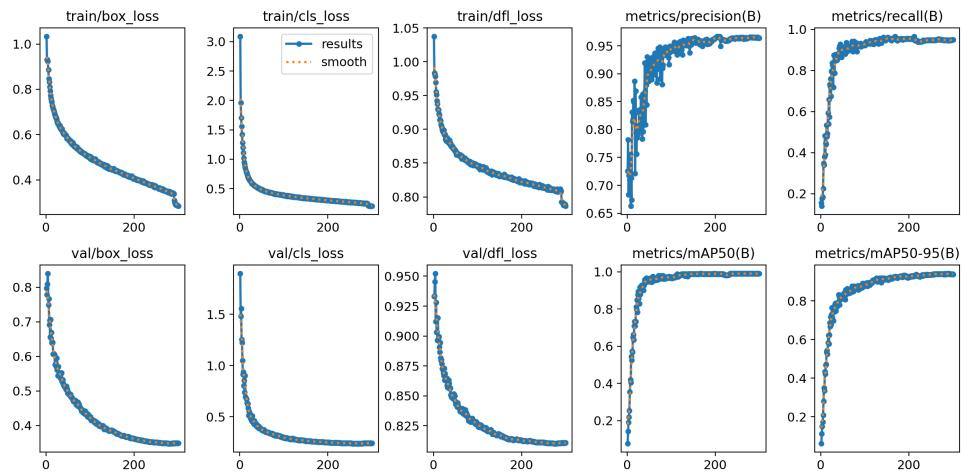


图 4-3 YOLOv11n 训练结果

基于 YOLOv11n 训练出的模型的混淆矩阵见图 4-4, 模型对大部分目标的分类效果都不错, 18 个标签中有 15 个标签的判断准确率达到了 97%。对 S07 这一类别的分类效果较差, 对其中 33% 的数据都预测为了 S10, 仅有 67% 的正确率。该模型的 P-R 曲线见图 4-5, 可以看出 mAP@0.5 为 99.0%。

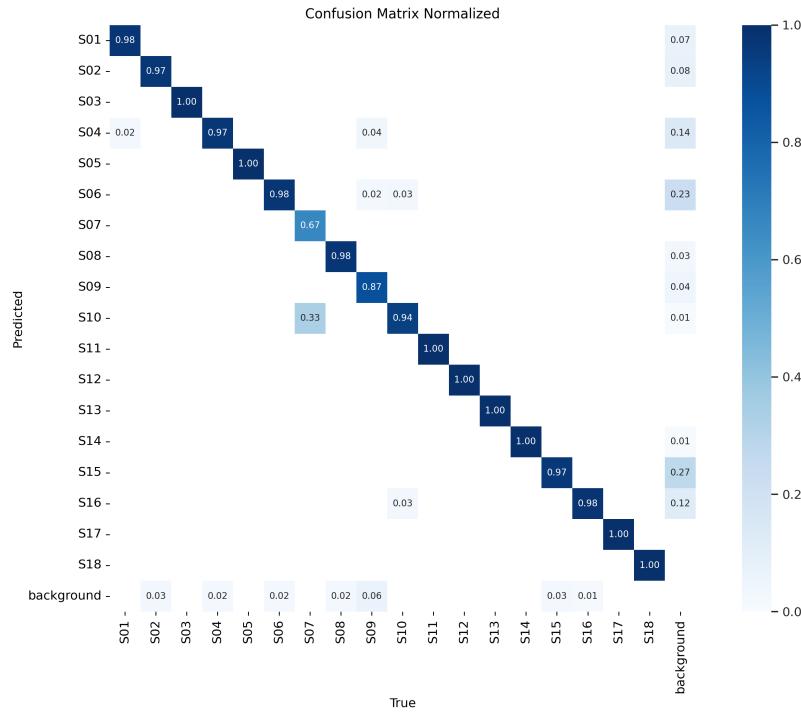


图 4-4 YOLOv11n 混淆矩阵

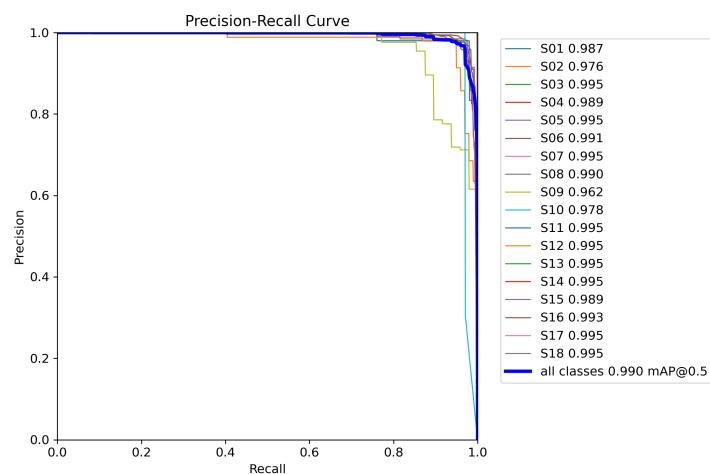


图 4-5 YOLOv11nP-R 曲线

4.2.2 YOLOv11s

本文基于 YOLOv11s 训练了 300 个 epoch，训练结果见图 4-6。该模型的最高精度为 96.2%，最高召回率为 97.2%，mAP50 为 99.0%，mAP50-95 为 95.4%。

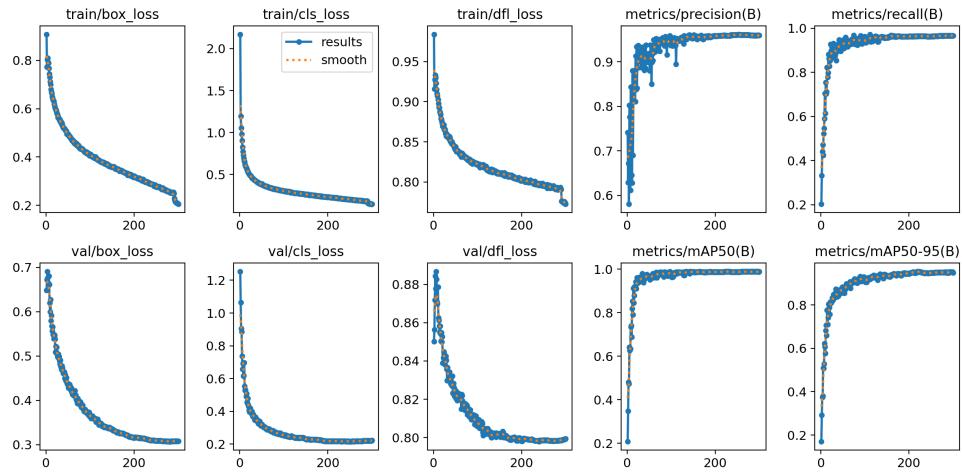


图 4-6 YOLOv11s 训练结果

图 4-7展示了该模型的混淆矩阵，可以看出模型对 S01 类别有 6% 的误判，其中 2% 预测为了 S04，4% 预测为了 S06。对 S09 类别有 2% 的误判，预测成了 S06，以及 4% 的漏判，其他的类别的准确率都达到了 97%。YOLOv11s 训练结果的 P-R 曲线如图 4-8，该模型的 mAP@0.5 为 99.0%。

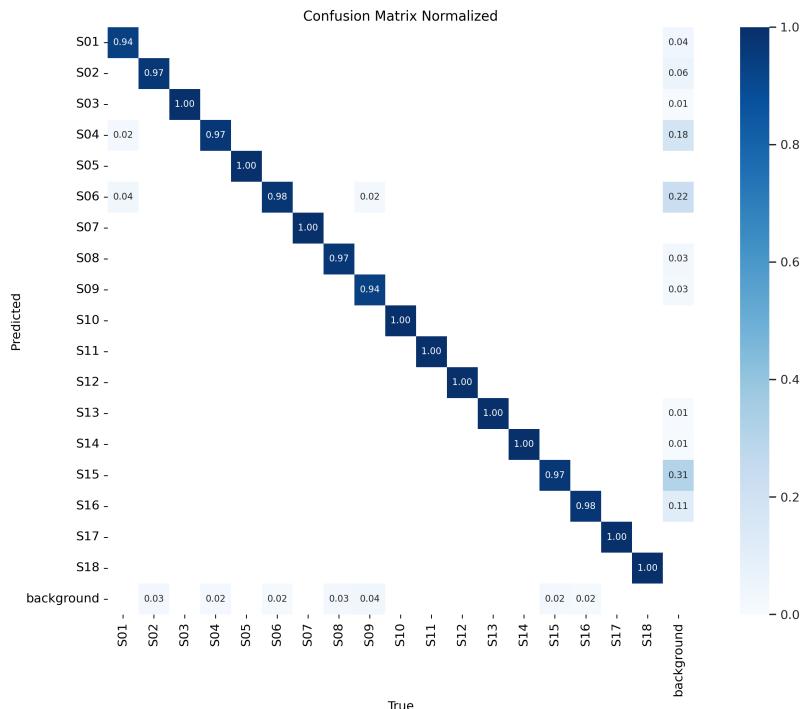


图 4-7 YOLOv11s 混淆矩阵

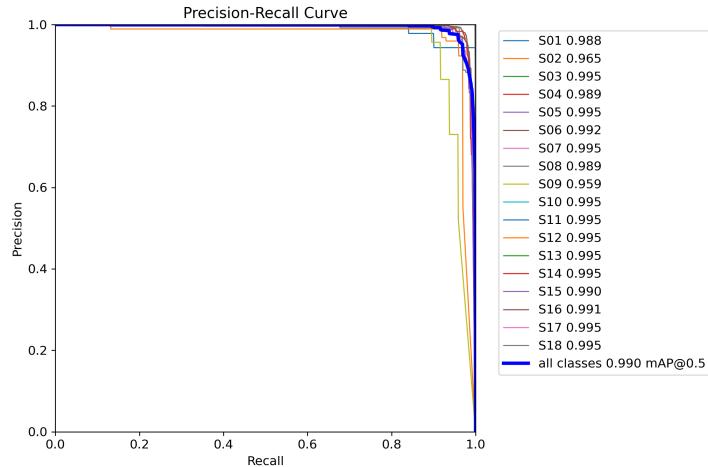


图 4-8 YOLOv11sP-R 曲线

4.2.3 YOLOv11m

本文基于 YOLOv11m 训练了 300 个 epoch，实验结果如图 4-9 所示，该模型的最高精度为 97.3%，最高召回率为 98.1%，mAP50 为 99.1%，mAP50-95 为 95.5%。

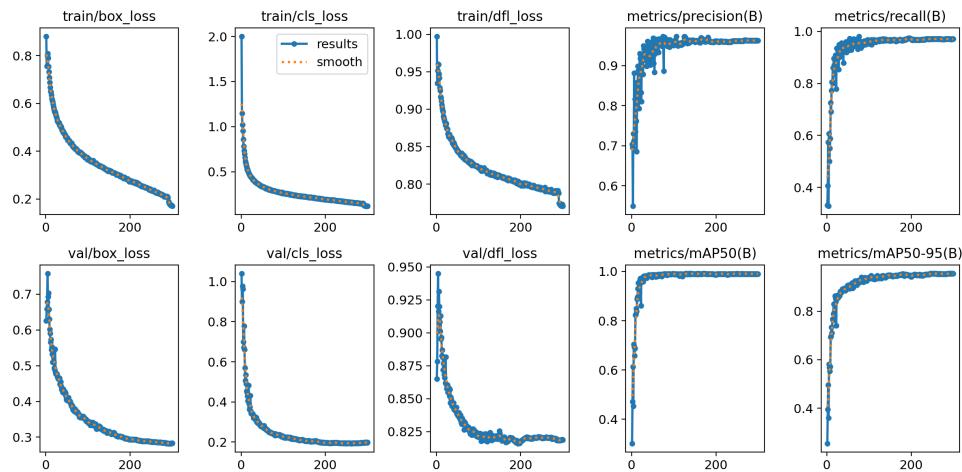


图 4-9 YOLOv11m 训练结果

混淆矩阵如图 4-10 所示，模型对 S04 类别有 4% 的漏判，对 S9 类别有 6% 的漏判，其他类别的准确率都达到了 97%。P-R 曲线为图 4-11，该模型的 mAP@0.5 为 99.1%。

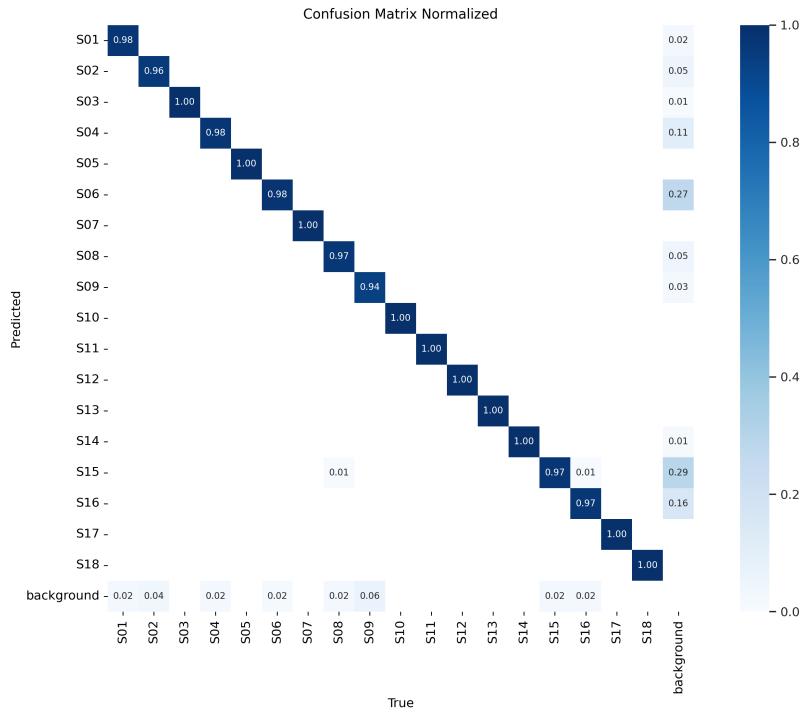


图 4-10 YOLOv11m 混淆矩阵

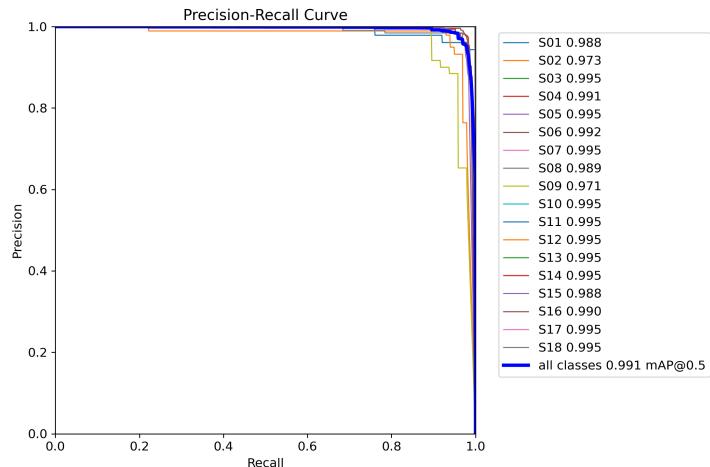


图 4-11 YOLOv11mP-R 曲线

4.2.4 模型性能对比

本文基于 YOLOv11 三种不同的模型对摩托车驾乘人员头盔佩戴数据集进行了训练，表 4-1 展示了各个模型在精度、召回率、mAP50、mAP50-95 以及检测速度这五个方面的对比。

三个模型的精度均在 96% 以上。其中，YOLOv11m 的精度最高，为 97.3%。三个模型的召回率也都处于较高水平，都在 96% 以上。YOLOv11m 的召回率最高，为 98.1%。

三个模型的 mAP50 值几乎一致，都在 99% 附近，最大仅相差 0.1%，说明在 IoU 阈值为 0.5 时，它们对目标物体的检测效果都非常好。在 mAP50-95 指标上，YOLOv11m 和 YOLOv11s 表现较好，分别为 95.5% 和 95.4%，仅相差 0.1%，都要高于 YOLOv11n 的 94.1%，这表明在更严格的 IoU 阈值范围内，YOLOv11m 和 YOLOv11s 的性能更优。

在检测速度方面，平均检测一张图片 YOLOv11m 耗时最长，为 31.5ms，YOLOv11s 为 27.3ms，比 YOLOv11m 快了 13.3%。YOLOv11n 最快，为 24ms，比 YOLOv11s 快了 12.1%，比 YOLOv11m 快了 23.8%。

根据分析可知，YOLOv11n 在速度上具有明显优势，适用于对实时性要求较高的场景。YOLOv11s 在精度和速度之间取得了较好的平衡，属于一个适中的模型。YOLOv11m 牺牲了一定的检测速度，换来了更高的精度，适合对检测精度要求极高的场景。

表 4-1 YOLO 数据标注格式

| Model | Precision(%) | Recall(%) | mAP50(%) | mAP50-95(%) | Speed(ms) |
|----------|--------------|-----------|----------|-------------|-----------|
| YOLOv11n | 96.7 | 96.6 | 99.0 | 94.1 | 24 |
| YOLOv11s | 96.2 | 97.2 | 99.0 | 95.4 | 27.3 |
| YOLOv11m | 97.3 | 98.1 | 99.1 | 95.5 | 31.5 |

4.3 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，然后对本文进行的模型训练结果进行展示，最后对本文三个模型的精度与速度进行了对比分析，总结了各个模型适用的场景。

第5章 检测系统的设计与实现

5.1 需求分析

该系统需要为用户提供两个模块的功能，一个是头盔佩戴情况检测功能，一个是历史检测数据查询功能，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片或视频，并需要支持用户自定义检测模型、IoU 和置信度参数。检测完成之后，需要将带有目标边界框的结果图片返回给前端并展示在页面上，供用户查看。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。检测结果要以可视化的形式展现出来，方便用户进行数据分析。

5.2 系统整体架构

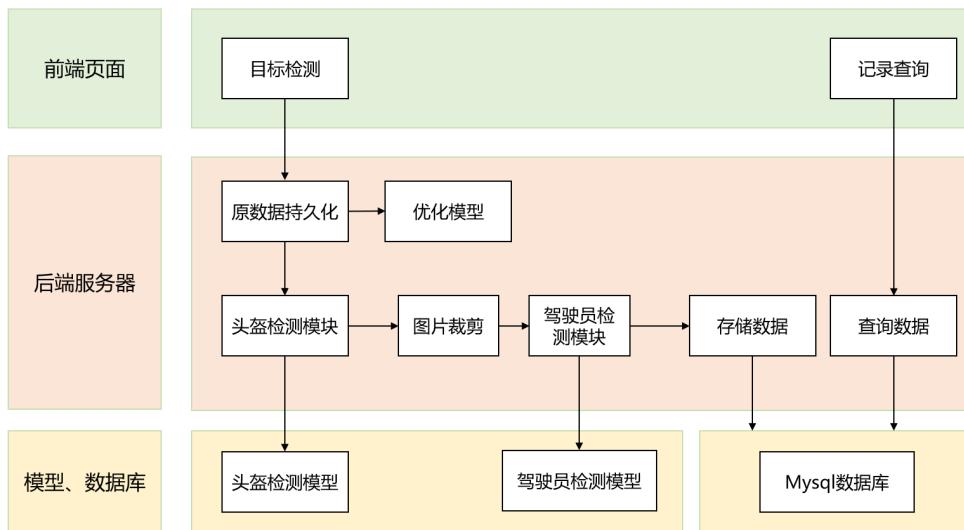


图 5-1 系统架构图

本系统整体架构如图 5-1 所示。前端为用户提供目标检测和记录查询两个功能，后端负责实现前端的这两个请求接口。后端在处理用户的目标检测请求时，首先会将用户输入的原数据做持久化，后续可以将这些数据作为新的数据集，不断优化模型。然后后端会把用户传入的自定义检测模型、IoU 以及置信度参数动态拼接到 python 脚本里，调用已经训练好的头盔佩戴检测模型进行预测。在 python 脚本里，设置 predict 参数 save_crop=True，会在预测之后根据目标检测框对原数据进行裁剪。后端会对这些裁剪之后的图片作为驾驶员检测模型的输入，进一步检测每一位驾驶员的信息。两步检测完成后，将检测结果，

包括驾驶员、头盔佩戴情况、检测时间等信息保存到 Mysql 数据库。最后将带有目标检测框的检测结果返回给前端，展示给用户。

用户请求记录查询接口时，前端将驾驶人、检测时间等参数传给后端，后端根据过滤条件去数据库查询历史数据。前端在这里通过 ECharts 组件实现了数据的可视化，将检测结果以柱状图、折线图和饼图的方式展示。

5.3 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。

5.3.1 目标检测界面

(1) 界面布局

目标检测界面的布局如图 5-2 所示。该界面包括显示区、参数设置区、系统消息区、功能触发区以及菜单。



图 5-2 目标检测界面布局

显示区是页面中最大，最主要的区域，用于展示用户上传的原数据，以及检测结果。该区域被设置为 flex 布局，左右两边的图像在整个显示区是左右对称的，且大小相同。右侧参数设置区允许用户调整检测模型、IoU 和置信度。右下方的系统消息区用来展示用户每一步的操作情况。下方的功能触发区给用户提供选择图片和视频、请求检测、终止当前检测以及保存检测结果的功能。最上方的菜单用于切换检测界面和查询界面。

(2) 功能测试

首先对检测界面的参数设置区进行检测。对同一张图片，设置不同置信度参数。图5-3和图5-4分别展示了置信度设置为0.5和0.95时的检测情况。结果表明，当置信度从0.5调整为0.95后，模型漏检了三个目标。通过查看后台python脚本执行日志，用户能够选择不同的模型进行目标检测。



图 5-3 置信度 0.5



图 5-4 置信度 0.95

然后对下方的功能触发区进行测试。该系统除了图片之外，还支持用户上传视频进行检测。经测试，上传视频并进行检测的功能可以正常使用，效果如图5-5所示。且用户点击保存结果后，会触发浏览器的下载功能，将检测结果保存到本地。在整个测试过程中，系统消息区可以将用户的操作结果正确展示出来。

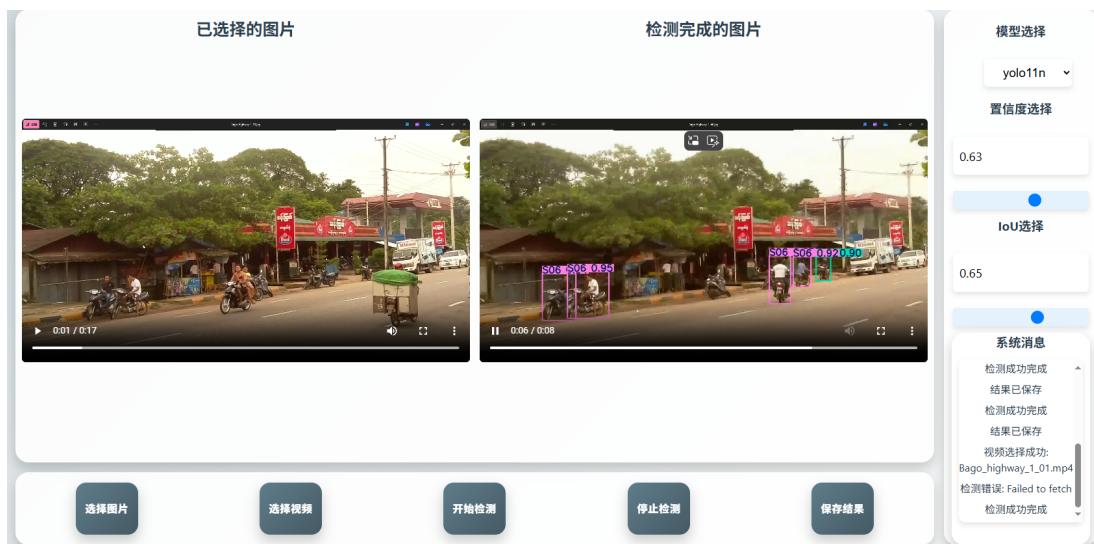


图 5-5 视频检测测试

5.3.2 记录查询界面

(1) 界面布局

记录查询界面布局见图 5-6。该界面为用户提供了记录查询功能，用户可以设置过滤条件，对历史记录进行查询。下方是查询结果的展示区域，这里提供了柱状图、折线图和饼图三种可视化方式。



图 5-6 目标检测界面布局

(2) 功能测试

对记录查询功能进行测试，设置为查询 3 月 21 日到 5 月 5 日的数据，可以通过下方的柱状图看出，过滤条件是生效的。切换不同的可视化图标，都可以展示正确结果。测试结果如图 5-7，图 5-8，图 5-9。



图 5-7 柱状图

图 5-8 折线图

图 5-9 饼图

5.4 后端模块设计与实现

由于 Java 语言强大的稳定性、跨平台性和丰富的第三方库，后端模块在框架上选用 SpringBoot 进行开发。为了便于系统的迭代和维护，在架构上选择 MVC 三层架构，即 Controller 层、Service 层和 Dao 层。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。

5.4.1 库表结构

该系统使用的库表结构如表 5-1 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 5-1 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 |

5.4.2 检测模块

该模块的执行流程图如图 5-10 所示，当接收到用户的检测请求时，首先会保存用户上传的原数据，然后会调用头盔检测模型检测数据中的头盔佩戴情况并保存检测结果到数据库。在检测驾驶员这一功能中，需要对原数据检测到的头盔佩戴目标框做裁剪，调用驾驶员检测模型检测每一个裁剪下来的小图片中的驾驶员，这个过程非常耗时，为了提高用户的使用体验，这里对驾驶员的检测会异步进行，不会阻塞用户的请求。YOLO 模型对视频的预测结果默认为 avi 格式，avi 格式的视频在目前各个浏览器上兼容性并不好且体积较大，所以后端这里会对视频检测结果做一个数据格式转换，由 avi 格式转换为兼容性更好的 mp4 格式。最后后端会构建好响应结果，返回数据给前端用户。

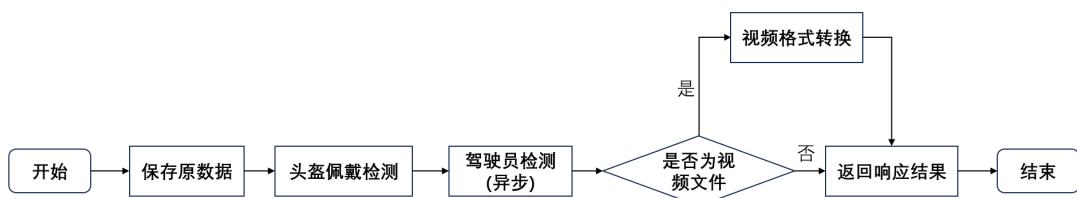


图 5-10 检测模块执行流程

5.4.3 搜索模块

该模块负责实现前端的记录查询功能，执行流程如图 5-11 所示。该模块会接收前端的搜索请求，并将过滤条件拼接到 Sql 语句中。如果前端传入的某一个字段过滤条件为空，则代表查询时不对该字段进行过滤。由于前端 ECharts 组件不同的可视化图标对数据的格式要求不同，还需要针对各个可视化图表的要求对数据作进一步处理。

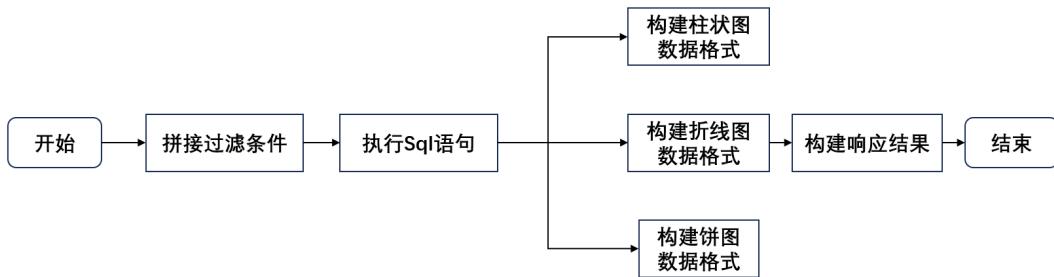


图 5-11 搜索模块执行流程

5.5 本章小结

本章详细介绍了基于 Vue 和 SpringBoot 框架的摩托车驾乘人员头盔佩戴检测系统的需求分析和界面实现。并对目标检测和结果查询两个前端页面的功能进行了测试。通过本章可知，该系统的前端和后端都已联通且功能正常。

第6章 总结与展望

6.1 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，提高检测精度，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片，以 7:2:1 的比例划分训练集、验证集和测试集，正确评估模型的性能。

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。可以得出结论，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 SpringBoot 和 Vue 框架设计并开发了前后端分离的系统。前端为用户提供了简洁美观的操作界面，包括目标检测界面和记录查询界面；后端则高效处理前端请求，完成图片或视频的检测、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。

6.2 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。YOLOv11 模型在本文中表现优秀，但也还存在提升空间。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。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。

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，还可以与智能交通系统的其他模块（如交通流量监测、违章行为自动抓拍等）进行深度融合，为城市交通管理提供更全面、高效的解决方案，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.
- 肖振久, 许子豪, 金海波, 李士博, 杨雅涵. 2025. DEL-YOLO: 安全帽佩戴检测的轻量化模型研究. 安全与环境学报, 1~11.
- 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)url+urldate.
- 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.

参考文献

- 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. 2024a. YOLOv10: Real-Time End-to-End Object Detection with Enhanced Efficiency-Accuracy Tradeoff. *arXiv:2405.xxxx*,
- 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 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 preprint arXiv:2207.02696*, arXiv: [2207.02696 \[cs.CV\]](https://arxiv.org/abs/2207.02696) <https://arxiv.org/abs/2207.02696>.

致 谢

致 谢