****

**毕 业 设 计**（**论 文**）

**论文题目：基于YOLOv11的交通标志检测**

**院 系：控制与计算机工程学院**

**计算机系**

**专 业：软件工程**

**班 级：软件2101**

**姓 名：陈博文**

**学 号：220211091002**

**指导教师：王平**

**二〇二五年 六 月**

摘 要

传统的交通标志检测技术存在较多缺陷，为解决传统模型精度低、速度慢、小目标漏检误检、模型参数量过大等问题，建立一种改进的YOLOv11交通标志检测模型，将原本的大目标检测层P5删除，减少高层抽象特征对计算资源的占用，以通过减少模型参数量来加快推理速度，考察驾驶过程中的实际情况，添加小目标检测层P2，增强对小型交通标志牌的检测能力和细节捕捉能力，缓解小目标丢失问题，减少误检漏检情况，保证参数量减少的同时精度损失较低。根据模型使用场景和国内道路情况，选用CCTSDB数据集和TT100K数据集进行模型微调。结果表明，改进后的模型n在数据集TT100K上参数量，精确度，召回率，平均精度分别是YOLOv11官方模型的34.5%，99.1%，94.7%，97.5%，在CCTSDB数据集上是官方模型的34.2%，97.8%，102.1%，100.8%，模型s在TT100K上是官方模型的30.7%，88.4%，91.7%，91.0%，在CCTSDB上是官方模型的28.6%，96.4%，99.4%，99.4%，模型在实现轻量化的同时保证了精度。开发了一个基于YOLOv11改进后的交通标志检测系统，提供了PyQt图形界面供用户操作，打包成为可执行文件便于安装，自由选择模型进行图片和视频检测，保存检测结果。

关键词：YOLO，交通标志检测，神经网络，深度学习，PyQt

**ABSTRACT​**​

Traditional traffic sign detection techniques exhibit several shortcomings. To address issues such as low accuracy, slow speed, false negatives/misses and false positives for small targets, and excessive model parameter count in conventional models, an enhanced YOLOv11 traffic sign detection model is proposed. The original large-target detection layer P5 is removed to reduce computational resource consumption by high-level abstract features, thereby accelerating inference speed through decreased parameter count. Considering real-world driving scenarios, a small-target detection layer P2 is added to enhance detection and detail capture capabilities for small traffic signs, mitigating small target loss and reducing false positives/misses while maintaining minimal accuracy compromise despite reduced parameters. Based on the model's application context and domestic road conditions, the CCTSDB and TT100K datasets are selected for model fine-tuning. Results indicate that the improved model-n achieves parameter count, precision, recall, and mAP of 34.5%, 99.1%, 94.7%, and 97.5% respectively of the official YOLOv11 model on TT100K; on CCTSDB, these metrics are 34.2%, 97.8%, 102.1%, and 100.8% of the official model. Model-s attains 30.7%, 88.4%, 91.7%, and 91.0% of the official model's performance on TT100K, and 28.6%, 96.4%, 99.4%, and 99.4% on CCTSDB. Both models ensure accuracy while achieving lightweight design. A traffic sign detection system based on the enhanced YOLOv11 is developed, featuring a PyQt graphical interface for user operation, packaged as an executable for easy installation. Users can freely select models for image/video detection and save results.

**Key words: YOLO, Traffic Sign Detection, Neural Network, Deep Learning, PyQt**

目 录

**[摘 要 I](#_Toc17539)**

**[ABSTACT II](#_Toc16454)**

**[第1章 绪论 1](#_Toc8712)**

[1.1 背景介绍 1](#_Toc25439)

[1.2 本文工作 2](#_Toc2002)

**[第2章 YOLOv11模型研究与改进 3](#_Toc19972)**

[2.1 模型介绍 3](#_Toc23033)

[2.1.1 C3k2模块 5](#_Toc28397)

[2.1.2 C2PSA模块 5](#_Toc18510)

[2.1.3 SPPF模块 5](#_Toc27491)

[2.1.4 深度可分离卷积 6](#_Toc19626)

[2.1.5 YOLOv11模型总结 6](#_Toc176)

[2.2 删除大目标检测层P5 7](#_Toc30399)

[2.3 添加小目标检测层P2 8](#_Toc20770)

[2.4 整合两种修改 9](#_Toc29560)

**[第3章 实验与结果分析 11](#_Toc614)**

[3.1 数据集 11](#_Toc5242)

[3.1.1 国内外开源交通标志数据集 11](#_Toc32658)

[3.1.2 选定实验数据集 12](#_Toc126)

[3.2 实验环境 13](#_Toc1166)

[3.3 模型部署与训练 13](#_Toc15255)

[3.3.1模型及预训练权重下载 13](#_Toc1471)

[3.3.2 开发环境配置 14](#_Toc28331)

[3.3.3 训练配置 16](#_Toc24104)

[3.4 模型评价指标 16](#_Toc20387)

[3.5 对比实验 18](#_Toc16209)

[3.5.1 TT100K数据集上的结果分析 21](#_Toc22552)

[3.5.2 CCTSDB数据集上的结果分析 21](#_Toc31164)

[3.5.3 检测结果展示 22](#_Toc9877)

[3.6 消融实验 24](#_Toc10672)

[3.6.1 消融实验概念 24](#_Toc1689)

[3.6.2 对改进进行消融实验 24](#_Toc29597)

[3.6.3 只删除P5层 27](#_Toc15875)

[3.6.4 只增加P2层 28](#_Toc5228)

[3.6.5 总结分析两种方案 28](#_Toc29331)

**[第4章 基于PyQt6的交通标志检测系统 30](#_Toc28015)**

[4.1 开发工具 30](#_Toc20630)

[4.1.1 开发环境及配置 30](#_Toc12507)

[4.1.2 PyQt6简介 30](#_Toc9229)

[4.1.3 PyInstaller简介 30](#_Toc15079)

[4.2 系统需求分析 30](#_Toc24508)

[4.3 系统设计 31](#_Toc17668)

[4.3.1 系统总体设计 31](#_Toc10761)

[4.3.2 用户交互界面模块设计 31](#_Toc19410)

[4.3.3 后台检测模块设计 32](#_Toc1436)

[4.4 界面开发实现 32](#_Toc1467)

[4.4.1 界面开发原则 32](#_Toc7188)

[4.4.2 界面开发 32](#_Toc30553)

[4.5 检测模型实现 33](#_Toc9039)

[4.5.1 数据集选择 33](#_Toc23566)

[4.5.2 训练参数设置 33](#_Toc23068)

[4.5.3 模型训练结果 34](#_Toc3203)

[4.6 代码实现和运行效果展示 37](#_Toc16205)

[4.7 程序打包分发 41](#_Toc6624)

[4.8 系统测试 42](#_Toc32386)

**[第5章 总结与展望 44](#_Toc19451)**

**[参 考 文 献 45](#_Toc10277)**

**[致 谢 47](#_Toc25044)**

# 第1章 绪论

## 1.1 背景介绍

近年来，随着人工智能技术的兴起，无人驾驶汽车以及智能辅助驾驶技术迅速发展，识别交通标志是自动驾驶中极为重要的一环，因错误识别交通标志导致的事故屡见不鲜，在实际道路中，交通标志的目标较小，有时会出现褪色、变形等情况，在恶劣天气如雾霾、暴雨下检测难度较高，想要在道路上正确识别交通标志，要求图像识别系统既要迅速，又要准确地完成任务，任何多余的检测延迟和误检漏检都可能危害驾驶员的生命安全。

交通标志识别系统的核心在于交通标志识别技术，深度学习兴起前，在传统的计算机视觉中，图像识别往往依赖人为标注的图像特征以及设计复杂的分类器，其精度较差，泛化能力不足，难以应对复杂的现实环境，对于雾霾、遮挡、破损的交通标志检测能力很差，计算速度不能满足高速行驶的车辆需求，为弥补传统技术的缺点，掩码-CNN[1]，Faster R-CNN[2],SSD[3],YOLO[4]等基于深度学习的目标检测模型被提出，相较于传统的目标识别模型，这些模型的检测精度与实时性大大提高，并取代传统模型成为主流，但是仍然存在计算量大、速度较慢的缺点，在要求实时性的车辆驾驶中难以接受[5]。Ultralytics公司在2023年推出了YOLOv8[6]模型，该模型计算速度较快，精度较高，抗干扰性强，2024年，YOLOv11模型推出，拥有相较于YOLOv8更高的精度与更少的参数量，适合作为辅助驾驶系统的图像识别模型。

国内外学者从不同角度对YOLO进行改进：例如将YOLOv4模型架构在英伟达Jetson Nano上，实时识别道路上交通标志，其准确率在54.68%-76.55%之间，速度能够达到32.5FPS[7];使用可优化边界框宽度与长度值的损失函数MPDIoU代替原模型中的CIoU损失函数，提升网络边界框回归性能，在精度没有明显降低的情况下参数量降低14.5%[8]；在特征提取模块中加入空洞卷积，然后在特征融合模块中加入CA注意力机制，并改进了损失函数，在精度提高了2%的同时未损失检测速度，FPS达到93[9]；设计了Conv-G7S和CSP-G7S模块,减少网络的参数量，设计了CSP-SwinTransformer模块，强化了使用窗口内特征信息进行上下文感知和建模的能力，并在颈部网络融合了卷积注意力机制，强化了对不同通道、空间权重信息的学习，最后对损失函数进行改进，提升了边界框回归性能，在精确率、平均精度提升了6.9%，3.7%的情况下参数量下降了75.4%，检测速度提升至136.99f/s[10]。

虽然经过如此多改良，在实际的交通标志检测中YOLO模型仍然出现了大量漏检误检问题，主要原因在于真实驾驶时拍摄到的交通标志往往尺寸较小，仅仅占据整张图片的极小一部分，且数量分布不均匀，仅某些位置密集分布大量目标，而对于大型交通标志牌，即使不使用太多计算资源也可以快速检出，

使用YOLO官方提供的更大尺寸的模型有助于减少相关问题，但其参数量过大，车载计算资源难以负担计算需求，导致检测速度过低，配备更高性能的计算芯片则会导致智能驾驶系统成本过高，难以推广。综上所述，本文旨在对于YOLO模型进行改进，在减少参数量的同时保证检测精度以适应现实条件。

## 1.2 本文工作

本文主要工作如下：

1. 使用中国开源数据集对YOLO模型进行微调，使其更加适应交通标志检测工作。
2. 删除对于大目标检测层P5，其原本工作交由中大型目标检测层P4代替，减少对计算资源的消耗，加快推理速度，减少参数量。
3. 增加小目标检测层P2，在更小的像素范围内进行特征提取，使模型对于小型交通标志有更强的检测能力，降低漏检率。
4. 使用PyQt制作图形界面，开发一个交通标志检测系统，提供图片检测和视频检测功能，自动保存检测结果，并配备多种模型供用户自由选择，将最终程序使用PyInstaller打包为可执行文件用于在其他设备上安装运行。

# 第2章 YOLOv11模型研究与改进

## 2.1 模型介绍

YOLO(you only look once)是Redmon等人于2015年提出的一个单阶段目标识别的对象检测模型，与之相对的是两阶段方法，即先扫描输入图片，划分出若干目标可能存在的区域，然后使用分类器对区域进行分类，YOLO作为单阶段方法，将对象检测任务转化为端到端的回归问题，在图像上直接通过卷积神经网络进行计算，一次输出所有识别到的目标信息，其速度优于同期的R-CNN，Ultralytics公司在后续几年内陆续推出了YOLOv1-YOLOv11等模型（YOLOv12及一些其他模型并非由Ultralytics公司提出），逐步改进，不断优化模型，为图像识别领域提供了性能优越的检测方法。下面对YOLO的发展历程简单总结。

YOLOv1的主干网络使用卷积层提取图像特征，最大池化层进行空间降采样，最终由连接层输出预测值，模型将图片分为若干栅格，对每个栅格产生预测框，预测框输出的预测结果含5维，分别是预测框在图片中的位置(x,y)，预测框的宽高(w,h)，以及置信度c，并预测类别概率，通过类别概率和置信度相乘来获得每个预测框中各类别的概率，得到最有可能的结果，作为初版YOLO模型，YOLOv1的精度和识别速度都较差。

YOLOv2[11]在YOLOv1的基础上使用VGGNet构建出DarkNet-19网络，通过归一化函数和高分辨率分类提升性能，但是一个分类器只能识别固定数量的物体。

YOLOv3采用了多尺度特征金字塔，将主干网络变为Darknet-53，使模型对目标的检测能力和特征提取能力提升，但也带来了耗费资源更大的问题[12]。

YOLOv4使用CSPDarknet 53作为主干网络，引入了空间金字塔池化技术，使用Mish激活函数，大大提升了性能和精确度、泛化能力，训练过程也更为复杂[13]。

YOLOv5采用了自适应锚框计算，和自适应图片放缩技术，减少了模型的计算量，同时需要更大的训练集来完成训练[14]。

YOLOv6对移动设备和嵌入式系统进行了推理速度的优化[15]。

YOLOv7引入了基于Transformer模型的预测头，将模型重参数化技术引入网络架构，使用了计划重参数化卷积等BoF方法，提高了模型精确度的同时不增加计算资源耗费以及训练成本，同时架构更加复杂[16]。

YOLOv8[17]将检测头从YOLOv5的耦合头更换为当前比较主流的解耦头，避免了推理过程中目标类别和位置相互耦合，使用正负样本匹配的方式来提取样本，使模型推理更加可靠[18]。

YOLOv9提出了可编程梯度信息(PGI)，主要由主分支、辅助可逆分支、多级辅助信息构成，具有较高的检测精度和处理重叠目标的能力，但对于小目标效率一般[19]。

YOLOv10架构类似YOLOv6，创新在于采用了无NMS一对多和一对一组合训练的方法，有效提高了推理速度，但是对重叠物体检测的准确度较低。

YOLOv11是Ultralytics公司于2024年9月发布的模型。其网络结构如图2-1所示。

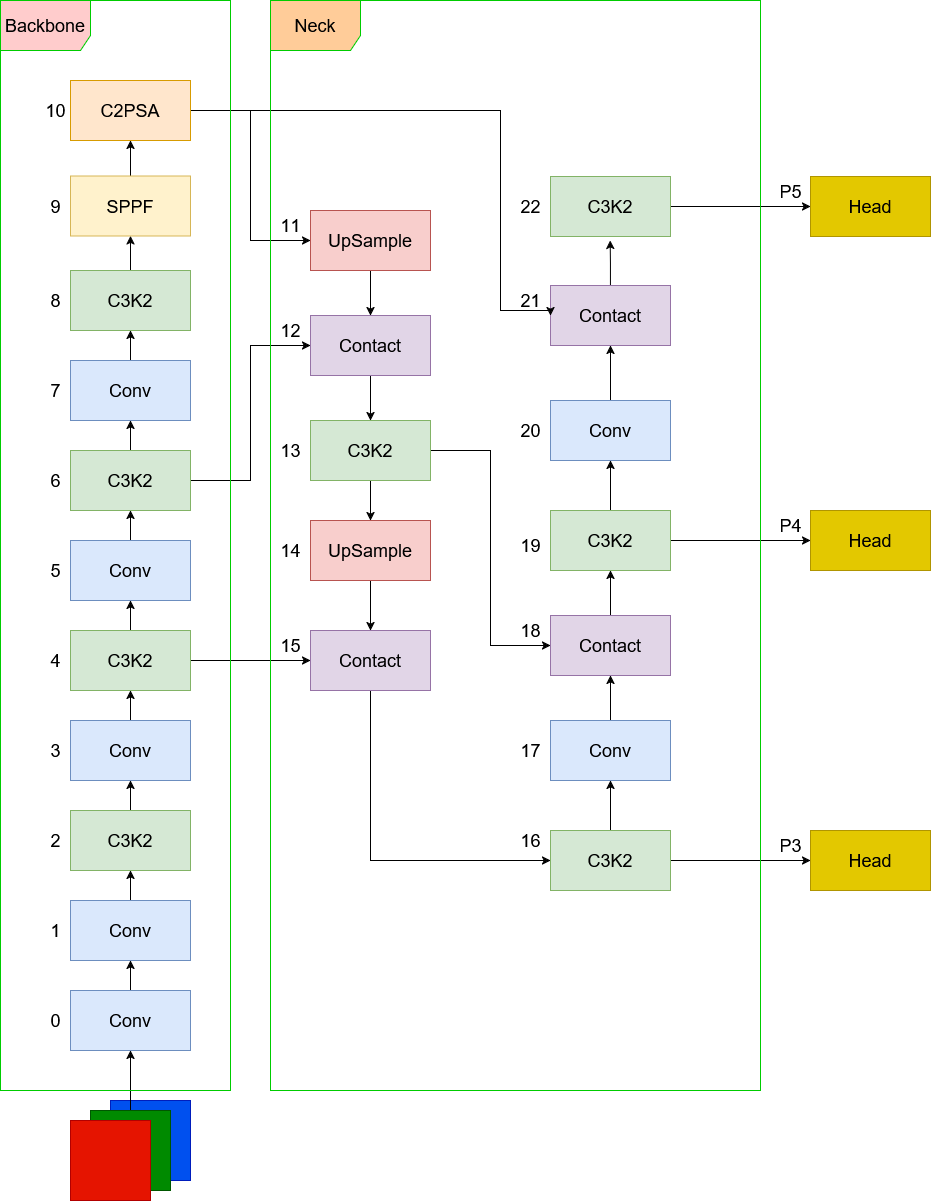


图2-1 YOLOv11网络结构图

2.1.1 C3k2模块

YOLOv11中的C3k2模块是该模型骨干网络的核心创新组件[20]，其设计融合了跨阶段部分卷积与双核动态调节机制，在保证检测精度的同时显著提升计算效率。

该模块的基本结构由特征分割层、双分支处理流和特征融合层共三个部分构成，首先输入特征图被均等分割为两部分，一路通过Identity操作直接传递原始特征以保留物体边缘纹理等浅层细节，另一路经过包含多级Bottleneck或C3K子模块的深度处理，目的在于提取如目标类别特征等高层语义信息。这种并行处理策略使模块能够同时兼顾特征复用与计算资源优化两个方面。

模块内部嵌入动态核选择机制，根据输入特征图的分辨率自动切换3×3与5×5卷积核尺寸，当在处理高分辨率特征图（如160×160）时采用大尺寸卷积核来增强全局特征捕获能力，而在处理低分辨率特征图（如20×20）阶段切换为小核减少计算量，C3k2采用深度可分离卷积替代标准卷积，可以将计算复杂度从传统方案的O(k²C\_inC\_out)降至O(k²C\_in + C\_inC\_out)。

2.1.2 C2PSA模块

C2PSA（Cross Stage Partial with Pyramid Squeeze Attention）模块是位于颈部网络的关键创新组件，其设计融合了跨阶段部分结构与金字塔挤压注意力机制，通过多尺度特征融合与空间动态加权显著提升模型在复杂场景下的目标检测精度，使其能够在雨雪，大雾天气下表现良好。

该模块采用了一种双分支并行处理架构，输入特征图首先经过1×1卷积压缩通道后被分割为两部分，Identity分支直接传递50%通道的原始特征来保留物体边缘纹理等浅层细节，另一分支通过金字塔空间注意力机制进行增强处理。在注意力分支中，特征图按坐标划分为4×4至8×8的多尺度子区域，并且结合距离衰减函数生成动态权重矩阵，这种操作会优先强化目标中心区域的特征响应。同时引入了通道维度的SE注意力机制，通过全局平均池化与全连接层生成通道权重向量，使关键特征通道的激活强度提升约28.6%。该模块通过3×3、5×5、7×7多尺寸卷积核并行提取特征，使多尺度目标适应性增强，适应不同距离下的交通标志检测。

2.1.3 SPPF模块

SPPF（空间金字塔池化快速版）模块是传统空间金字塔池化（SPP）的升级版本，设计目标是通过串行化池化操作在保持多尺度特征提取能力的同时显著提升计算效率[21]。该模块继承自YOLOv5的SPPF架构，在YOLOv11中进一步优化了池化层的参数配置与特征融合策略。输入特征图先经过1×1卷积压缩通道维度以降低计算负载，然后进行三次连续的最大池化操作（默认采用5×5池化核），每次池化后保留当前尺度的特征信息，最终将原始特征与三次池化结果在通道维度拼接，再通过1×1卷积恢复通道维度形成输出。这种串行池化设计相较传统SPP的并行池化结构，在计算量减少约40%的同时仍能等效捕获5×5、9×9、13×13三种尺度的特征响应。

在特征融合阶段，SPPF通过多尺度特征叠加增强了模型对于目标全局轮廓和局部细节的协同感知能力。

三次池化操作等效于特征图经过递进式感受野扩展，每次池化后的特征图分辨率保持不变，三次5×5池化的叠加效果与单次13×13池化的感受野等效，但计算复杂度大幅降低。

2.1.4 深度可分离卷积

标准的卷积操作将卷积核应用于输入特征的所有通道上，通过矩阵乘法累加运算来提取特征。但这种方式往往导致模型参数量巨大，且在特征提取过程中存在大量的冗余。因此研究者们致力于改进卷积结构，以在减少参数的同时提高效率[22]。

深度可分离卷积（depthwise separable convolution,DSConv)[23]巧妙地结合了深度卷积（depthwise convolution,DW）和逐点卷积（pointwise convolution,PW）两个过程,常规卷积结构如图2-2所示，深度可分离卷积结构如图2-3所示。在深度卷积阶段，每个输入特征通道都独立地与对应的卷积核进行卷积操作，这样可以精确提取特征图在长宽方向上的空间信息特征。随后，进入逐点卷积阶段，这里采用1×1的卷积核对深度卷积的输出进行卷积，以捕捉特征中不同通道之间的信息。通过这两个步骤的协同工作，深度可分离卷积在保持特征提取能力的同时，显著减少了模型的参数数量，从而实现了轻量化效果。

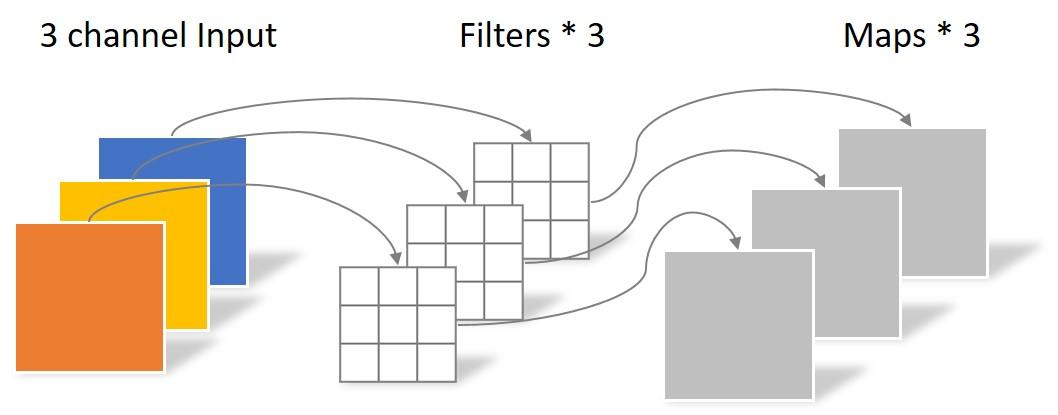


图2-2 常规卷积示意图

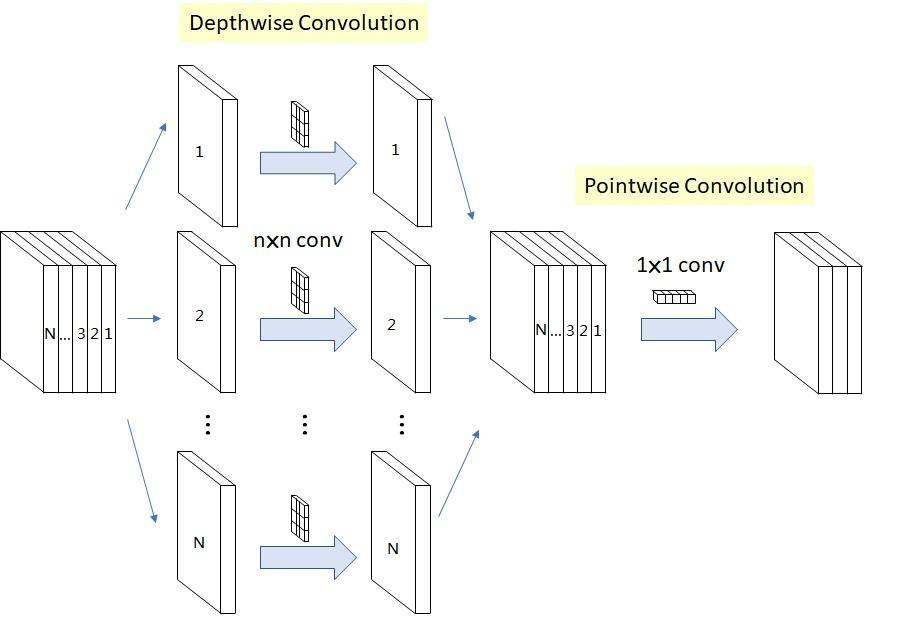


图2-3 深度可分离卷积示意图

深度可分离卷积的计算量仅约为常规卷积的1/k2（k表示卷积核的大小）。因此当神经网络采用这种卷积方式时，可以有效地减少参数数量和计算复杂度，在保持模型准确率的同时，极大地缩减了模型的参数量，有利于实际应用中的存储和部署。通过利用深度可分离卷积，可以构建更加轻量且高效的神经网络模型，适应对速度要求更高的场景。

在YOLOv11中，DWConv主要应用于**分类检测头（Classification Head）**的结构优化。YOLOv8的分类分支采用标准卷积层，在YOLOv11中被替换为两个串联的DWConv层，每个DWConv包含深度卷积与逐点卷积的组合。分类头的cv3模块被重构为DWConv(x, x, 3) → Conv(x, c3, 1) → DWConv(c3, c3, 3) → Conv(c3, self.nc, 1)的形式。使分类分支的参数量减少约22%，同时通过堆叠轻量化操作增强非线性表达能力，保证缩减模型大小的同时不失精度。回归分支（Bounding Box Regression）仍保留标准卷积以维持定位精度，来避免轻量化结构对空间敏感任务的影响。此外，YOLOv11的**C2PSA模块**中也嵌入了DWConv技术，用于局部特征提取与注意力机制的协同优化，从而增强对小目标的检测能力，避免漏检。

2.1.5 YOLOv11模型总结

这些改进使YOLOv11相比于v8版本具有更少的参数量和更高的准确性，此外，YOLOv11还具有更强的环境适应性，可以在不同环境中部署，例如边缘设备、云平台等，但是更复杂的网络结构使YOLOv11的训练需要消耗更多的计算资源与时间，要求训练者提供更昂贵的计算芯片，而最新推出的YOLOv12模型不但训练成本更加高昂，而且在某些场景下的精度较v11更差。YOLOv11拥有n,s,m,l,x五级训练完毕的模型，结合对速度的要求，本文选取了YOLOv11的n和s模型作为实验对象。

2.2 删除大目标检测层P5

P5层位于Backbone和Neck结合后的高层特征输出层，通过多次卷积和下采样操作，可以捕获图像中更全局的语义信息，通过高层融合增强了对目标上下文信息的理解，但是由于高层特征分辨率较低，对小型目标的细节识别捕捉能力较弱，主要用于处理大尺寸目标，例如车辆、建筑等物体，在交通标志检测中，交通标志往往较小[24]，待检测目标很少占据输入图像的大部分区域。尽管深层卷积能够提取更抽象和更丰富的特征信息，但在原图经过多次下采样后会丢失大量小目标信息。因此，在小目标为主要检测对象的情况下，P5检测头的作用相对有限。此外，由于深层卷积通常分配更多通道来表示抽象特征，移除P5检测头能够显著减少模型的参数量和计算复杂度。闫建红等人移除P5检测头，以减小新增小目标检测头所带来的额外计算量，实现了模型的轻量化[25]。由此可见，使用P5层不仅增加大量参数，降低推理速度，还无法获得较为明显的精度提升，故调整yaml文件，删除P5对应的C3k2层和卷积层，将原本连接的路径指向剩余的最高层P4，删除后网络结构如图2-4所示。

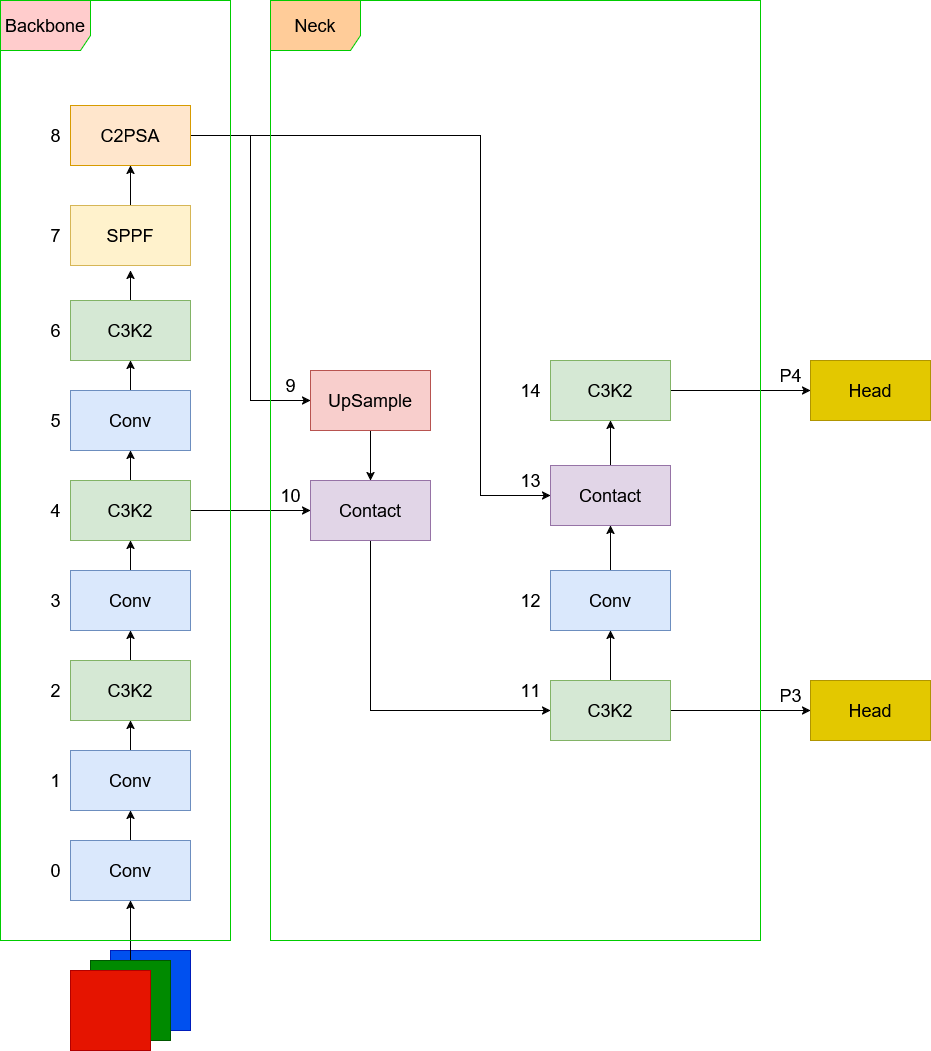


图2-4 删除P5层后的网络结构图

## 2.3 添加小目标检测层P2

P2层位于特征金字塔底层，相比于P3-P5层，P2层保留了更多空间的细节，对小型目标的细节识别捕捉能力强，特别适合检测像素占比小的目标，尽管在模型网络中引入P2层会带来额外计算开销[26]，但开销主要集中在P2检测头部分。在颈部网络中引入P2层仅会增加较少的计算量和参数量，而其提供的浅层特征信息能够显著提高小目标检测性能，例如远方或是树后的交通标志牌。故调整下采样策略，修改yaml配置文件，在早期卷积层增加P2分支，调整通道数，向上连接P3层，添加了小目标检测层，增加对于小目标的识别能力，减少误检漏检。添加后的网络结构如图2-5所示。

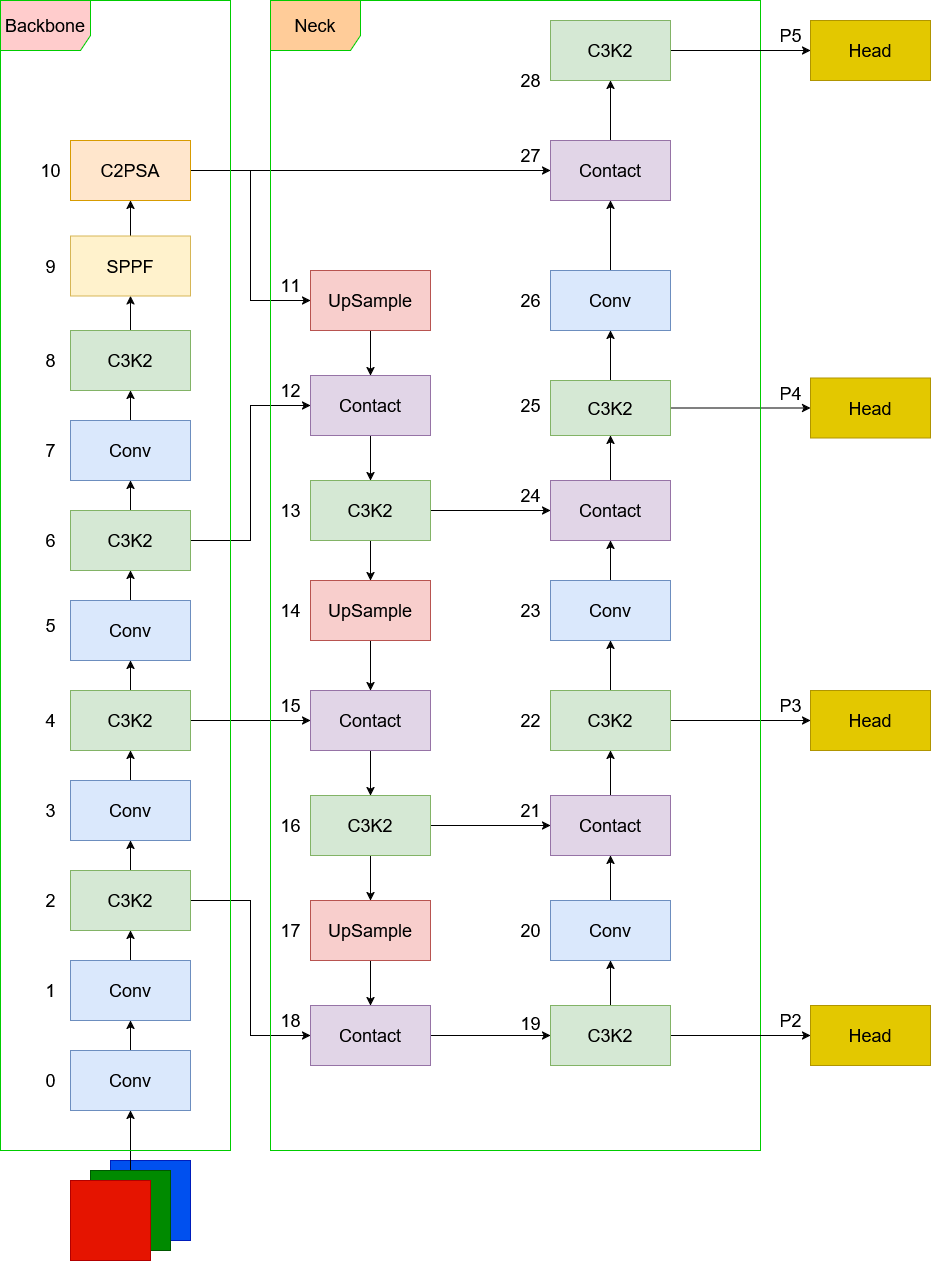


图2-5 增加P2后的网络结构图

## 2.4 整合两种修改

进行如下操作进行修改整合：在骨干网络**Backbone部分新增P2特征提取路径**，通过添加步长(stride)为2的3×3卷积层生成高分辨率特征图，并在**Head部分重构特征融合路径，**将原P3/8、P4/16、P5/32的跨层连接替换为P2/4、P3/8、P4/16的拼接结构，同时删除与P5相关的上采样及特征拼接操作。

将上述两种修改方法整合后的网络结构如下图2-6所示。

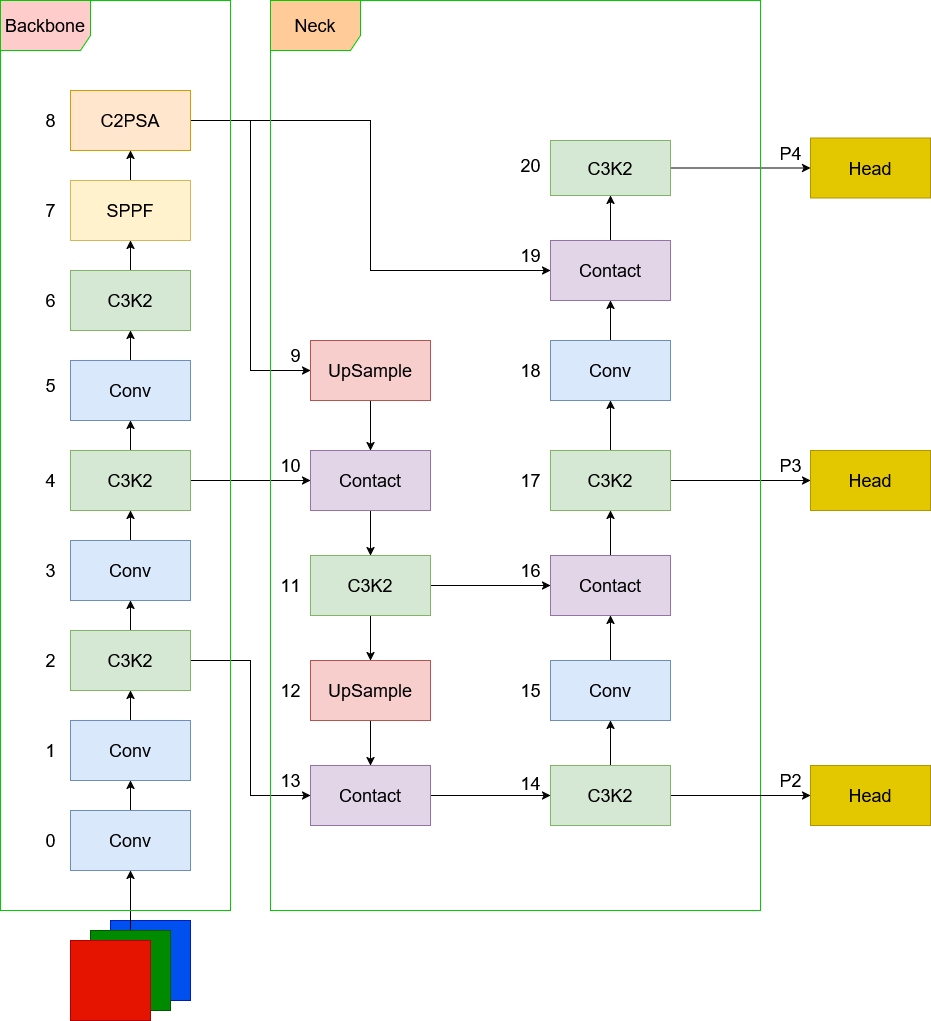


图2-6 整合后的网络结构图

第3章 实验与结果分析

## 3.1 数据集

## 3.1.1 国内外开源交通标志数据集

3.1.1.1 LaRa数据集

LaRA（LaRA Dataset）数据集是一个用于交通信号灯检测的开源数据集，主要服务于自动驾驶领域的研究。数据采集自法国巴黎的街道场景，具体时间不明，数据集规模与欧美主流城市交通场景训练需求相匹配。该数据集共包含**9,168张图像样本**，分辨率为**640×480像素**，数据集规模中等，交通信号灯状态被简单分类为四类通用标签：**行驶（Go）、警告（Warning）、停止（Stop）和危险（Danger）**，适用于基础交通信号识别任务。缺点在于该数据集缺乏对不同天气条件和多时间段的覆盖，且无倒计时显示标注、非机动车信号灯等细粒度信息，限制了其在真实复杂城市环境中的泛化能力，对于用于实际道路的自动驾驶系统价值有限。

3.1.1.2 LISA数据集

LISA（Laboratory for Intelligent and Safe Automobiles）是由美国德州大学奥斯汀分校的智能交通研究室开发的交通标志识别数据集，主要服务于自动驾驶和交通安全监控领域，包含早期版本的**7,369张图像**至扩展版本的**43,007帧视频图**，分辨率为**1280×960**，覆盖美国典型交通场景。数据主要采集自美国加利福尼亚州圣地亚哥，覆盖白天、夜间及不同天气条件，通过车载立体摄像头动态拍摄。但各类别数量极不均衡且有地域限制。

3.1.1.3 博世交通标志数据集

**德国博世交通标志数据集（Bosch Small Traffic Lights Dataset, BSTLD）**由德国博世公司（Bosch）开发，旨在为自动驾驶和智能交通系统提供高质量的交通信号灯检测数据，解决复杂城市环境中的小目标识别难题。共包含**13,427张图像**（训练集5,093张，测试集8,334张），分辨率为**1280×720像素**，标注了约24,000个交通信号灯实例。具有红灯（red）、绿灯（green）、黄灯（yellow）、关闭（off）四种类别。该数据集交通灯在图像中的尺寸极小，中位宽度仅约8.5像素（训练集8.6像素，测试集8.5像素），适用于对小目标交通灯识别的专门训练。

### **3.1.1.4 Mapillary Traffic Sign数据集**

Mapillary Traffic Sign Dataset 是由开源地图平台Mapillary发布的全球性交通标志数据集，是为自动驾驶和计算机视觉研究提供多样化的标注数据的数据集。包含**52,375张高分辨率图像，清晰度较高，**平均分辨率约1920×1080，标注了**313,611个交通标志实例**。覆盖全球六大洲，包含城市、乡村、山区、高速公路等复杂场景。共标注**300种交通标志类别**，涵盖禁令、警告、指示、临时标志等多种类型，包括各国特有标志。但是因依赖社区众包标注，部分图像存在标注噪声，且欧洲和北美数据较多，非洲、南美样本相对较少，均衡性较差。

### **3.1.1.5 D²-City数据集**

### **D²-City**由滴滴于2019年推出的首个大规模行车视频数据集，包含**10,000余段**行车记录仪视频，覆盖中国5个城市。视频以高清（720P）或超高清（1080P）录制，帧率25fps，每段视频时长30秒，共约1,000段视频提供逐帧标注，分为12类。该数据集每日产生大量原始视频数据，经由第三方数据标注公司处理，使用者必须拥有大容量数据存储设备，本文限于成本和模型大小，不考虑使用。

3.1.1.6 CCTSDB 2021数据集

CCTSDB（Chinese Traffic Sign Detection Benchmark）是由长沙理工大学张建明教授团队构建的专用于中国交通标志检测与识别的大规模基准数据集，覆盖城市道路、高速公路、乡村道路等多种场景，并包含不同光照（白天/夜间）和天气条件（雨雪/雾天）的图像，总量达**17,856张**（训练集16,356张，测试集1,500张），并加入500张负样本（无标志图像），约15,734张图像，标注为三大类：**禁令标志（prohibitory）**、**警告标志（warning）**、**指示标志（mandatory）[27]。**

**3.1.1.7 TT100K数据集**

TT100K（Tsinghua-Tencent 100K）数据集是由清华大学与腾讯联合实验室共同构建并持续维护的交通标志检测与识别领域权威基准数据集，数据采集自中国多个城市的腾讯街景全景图，经裁剪后生成**2048×2048**的单张图像，总计**10万张**，其中包含交通标志的图像约**1万张**（训练集6,105张，测试集3,071张）。共标注**30,000多个交通标志实例**，覆盖**221种交通标志类别**（如限速、禁止停车、注意行人等），标志中位尺寸仅**80×80像素**，部分极小目标（如远距离标志）仅占**5×5像素，**数据主要来自中国城市，缺乏国际标志类型。

3.1.2 选定实验数据集

考虑到本文研究模型主要用于国内自动驾驶系统，欧美数据集往往与中国道路状况不同，缺少中国特有的交通标志，而国内交通标志数据集的滴滴数据集，其数据量巨大，且大多缺少标注，故不选用；CCTSDB数据集和TT100K数据集2021版本均来自中国道路，图片量在10000到20000之间，且有清晰标注，图片质量较高，CCTSDB分类较粗放，仅有3种类别，而TT100K数据集的交通标志类别达到221种，两者可互为对比，故选用CCTSDB数据集和TT100K数据集，接下来所有关于YOLO模型的训练均以CCTSDB数据集和TT100K数据集为基础。考虑到TT100K数据集类别较多且类别名称为字母数字编码，图3-1为其类别名称编码，以便于对照。

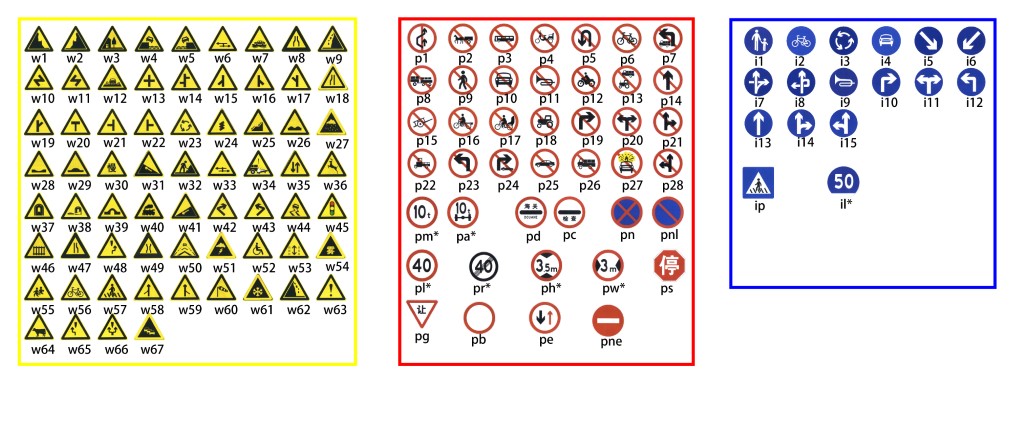


图3-1 TT100K的类别编码

## 3.2 实验环境

操作系统为Ubuntu 22.04，CPU使用60 vCPU Intel(R) Xeon(R) Platinum 8362 CPU @ 2.80GHz，GPU使用RTX 3090(24GB) \* 4。

开发环境为Pycharm 2024.1.1。Python版本3.10，Pytorch版本2.1.0，CUDA版本为12.1，cuDNN版本为8.9.0。

训练迭代次数初始均设定为300轮，使用官方提供的预训练参数，批次大小为200，线程数45，输入图像分辨率大小为640\*640，使用SGD优化器训练模型，开启自动混合精度训练和缓存选项，其他使用默认值。

3.3 模型部署与训练

3.3.1模型及预训练权重下载

Ultralytics公司在官方Github仓库提供了模型源代码，可以在仓库内拉取代码到本地，并在README文件中使用提供的链接跳转到预训练权重下载界面，官方提供了5个尺寸的预训练权重，分别是n（Nano）、s（Small）、m（Medium）、l（Large）、x（XLarge），参数规模逐级递增，适用于从边缘设备到云端的全场景部署，任务类型包括目标检测（Detection）、实例分割（Segmentation）、姿态估计（Pose），使用预训练权重可以大大减少本地训练所需时间，优化模型效果。每种预训练权重的相关信息如图3-2所示。

（该图由Ultralytics公司提供，为YOLOv11在COCO数据集上的训练示例，包含80个预训练类别）

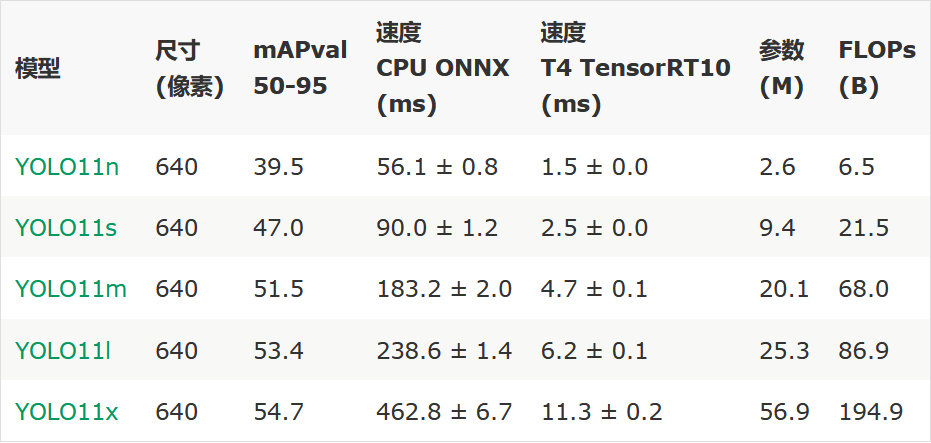


图3-2 YOLOv11在COCO数据集上的示例

3.3.2 开发环境配置

本文所需关键工具及依赖包括Anaconda,CUDA,cuDNN,Pytorch。

Anaconda是一个开源的Python发行版，专注于数据分析和科学计算。用户可以在Anaconda中配置虚拟环境以便控制版本以及版本隔离，其核心功能是包管理与环境管理。包管理与pip的使用类似，环境管理则允许用户方便地安装不同版本的Python并可以快速切换。

**CUDA**（Compute Unified Device Architecture）是由**NVIDIA**开发的并行计算平台和编程模型。它为软件开发人员和研究人员利用**NVIDIA**的**GPU**（图形处理单元）进行高性能计算提供了平台。**CUDA**提供了一系列 **API** 和工具，使开发者能够方便地编写和优化在**GPU**上运行的计算密集型任务。也就是说CUDA通过CPU任务分发和GPU并行处理的方式，把计算任务通过CPU分发给GPU进行并行计算加速。CUDA本身的编程接口和工具可以将开发者的高级语言转换为机器码，直接在NVIDIA公司生产的GPU上运行，大大加速了AI模型的训练与推理。

**cuDNN**（CUDA Deep Neural Network Library）是由NVIDIA公司提供的一个深度神经网络库，用于加速深度学习应用程序的开发和训练。cuDNN提供了一系列高性能的深度学习模型和函数集合供开发者使用，以及针对GPU硬件的优化实现，就相当于工作的工具，想要在CUDA上运行深度神经网络，就要安装cuDNN，插入式设计，把cuDNN数据库添加进CUDA中就能使用。它强调性能、易用性和低内存开销，使得开发者能够利用GPU的强大并行计算能力来加速相关程序。

PyTorch是一个开源的机器学习库，是一种用于构建深度学习模型的功能完备框架。由Facebook的人工智能研究团队开发，并于2016年发布。PyTorch以易用性、灵活性和强大的功能而广受欢迎，上手容易，开发简单。它基于已有的张量库Torch开发，并引入了动态计算图，使模型建立更加灵活，可以动态修改计算模型，将 Torch 中高效而灵活的 GPU 加速后端库与直观的 Python 前端相结合，极大提升了开发效率，非常适用于原型设计，且从早期版本开始，API一直保持一致，拥有一个完整的大型社区。

对于Anaconda，用户可以在Anaconda官方网站下载安装。安装完毕后即可以使用

conda create -n yolo\_env python=x.x

conda activate yolo\_env

命令创建并激活虚拟环境，此后如果在虚拟环境内安装配置Python环境及相关依赖，可以做到版本隔离，避免外界环境干扰，以及避免内部安装的环境干扰外界环境

建议使用GPU训练模型，在命令提示符窗口键入nvidia-smi查看本机对应的CUDA版本，在CUDA官网下载合适版本的CUDA后，到英伟达官网安装对应版本的cuDNN，安装完毕后，在Github拉取到的Ultralytics项目根目录下使用

pip install -r requirements.txt

自动开始下载所需依赖（包括Pytorch），下载完毕后使用

python detect.py --weights yolov11s.pt --source data/images/bus.jpg --device 0

若输出检测结果图像（runs/detect/exp），则环境配置成功，使用

python -c from ultralytics import YOLO;

print(YOLO('yolov11n.pt').info())

若输出版本信息，则环境正常。

如果对于自动安装的依赖版本不满意，可以手动对依赖单独进行删除与下载。

完成依赖项配置后，再配置好数据集文件即可训练模型。数据集文件通过data.yaml进行指示，在根目录下创建data.yaml文件指定数据集路径和类别信息，内容根据下列格式：

|  |
| --- |
| # 数据集根目录(可选)  path: ../dataset  # 训练集路径(相对path或绝对路径)  train: images/train  val: images/val  test: images/test # 测试集可选  # 类别数量  nc: 2  # 类别名称列表  names: ['cat', 'dog'] |

3.3.3 训练配置

进行训练时存在大量可自定义参数，常用参数及功能如表3-1。

表3-1 自定义参数表

|  |  |  |
| --- | --- | --- |
| 参数名称 | 描述 | 默认值 |
| ​​model​​ | 模型配置文件或预训练权重路径 | yolov11n.yaml |
| ​​data​​ | 数据集配置文件路径 | data.yaml |
| ​​epochs​​ | 总训练轮次 | 100 |
| ​​batch​​ | 批次大小 | 16 |
| ​​imgsz​​ | 输入图像尺寸 | 640 |
| ​​optimizer​​ | 优化器选择（SGD/Adam/AdamW等） | SGD |
| ​​lr0​​ | 初始学习率 | 0.01 |
| ​​lrf​​ | 最终学习率系数 | 0.01 |
| ​​weight\_decay​ | L2正则化系数，防止过拟合 | 0.0005 |
| ​​momentum​​ | 优化器动量参数 | 0.937 |
| ​​device​​ | 训练设备 | 0 |
| ​​workers​​ | 数据加载线程数 | 8 |
| ​​cos\_lr​​ | 是否启用余弦学习率调度 | FALSE |
| ​​patience​​ | 早停机制阈值 | 50 |
| ​​save\_period​ | 模型保存间隔轮次 | -1 |
| ​​pretrained​​ | 是否加载预训练权重 | TRUE |
| ​​hsv\_h/s/v​​ | 色调(0-0.1)/饱和度(0-0.9)/明度(0-0.9)增强幅度 | 0.015/0.7/0.4 |
| ​​fliplr​​ | 左右翻转概率 | 0.5 |
| ​​mosaic​​ | Mosaic数据增强概率 | 1 |
| ​​mixup​​ | Mixup增强概率 | 0 |
| ​​box/cls/dfl​​ | 边界框/分类/分布焦点损失的权重系数 | 7.5/0.5/1.5 |
| ​​label\_smoothin | 标签平滑系数 | 0 |

设定完毕后使用model.train(args)即可开始训练。

## 3.4 模型评价指标

本文实验使用精确度(Precision)、召回率(Recall)、平均精度值(mean Average Precision)、参数量(Parameters)来对模型进行评估。

其计算公式如下：

(3-1)

(3-2)

(3-3)

(3-4)

在上述公式中，TP表示模型预测结果为正样本且实际也是正样本的数量；FP表示模型预测结果为正样本但实际是负样本的数量；FN表示模型预测结果为负样本但实际为正样本的数量。N表示所使用数据集中类别数量，AP表示P-R曲线上精确度的平均值，mAP表示对数据集中所有类别取平均得到的平均精度值，便于全面评估模型的检测能力。使用参数量来衡量模型大小。

mAP是目标检测领域最核心的评估指标，它通过计算不同交并比（IoU）阈值下的平均精度（AP）后取类别平均值得出。

**精确率**衡量模型预测结果中正确检测的比例，高精确率意味着低误检率。交通标志检测中，误检可能导致自动驾驶系统误判。

**召回率**反映模型对真实目标的覆盖能力，漏检可能引发安全隐患。

参数量指模型中所有可学习变量的总数，直接表征网络复杂度与部署成本，包括权重（weights）和偏置（biases）[28]。权重决定输入信号在神经元间的传递强度，偏置则作为固定偏移量引入非线性，这两者共同构成模型的决策边界。大参数模型要求使用者为模型预留出足够的硬盘和显存容量，也需求更多的加载时间，例如deepseek作为不以参数量大著称的大语言模型，仍然具有6710亿参数。理论上参数量越大，模型上能拟合更复杂的函数。但受数据量和计算资源制约，并非参数量越大模型效果必然越好。当参数量远大于训练数据规模时，模型容易过度记忆训练数据的噪声和细节，发生过拟合现象，导致泛化能力下降，在陌生数据上表现不佳。如果训练数据仅覆盖少量交通标志类别，大参数量模型可能在小众类别上表现更差，而轻量化模型反而能通过数据增强提升效果。

边缘设备需在短时间内完成推理，出于车载设备存储能力和时间要求，其适用的模型参数量必须精简。所以说参数量是模型能力的基础，但其效果受**数据质量、架构效率、训练策略、任务需求**等多因素制约。所以要求自动驾驶系统配备的模型参数量不可过大。

综上所述，对于自动驾驶系统所用的图像识别模型的研究，应当尽可能提高其精度，召回率，平均精度值，尽可能减小参数量。

## 3.5 对比实验

为验证本文提出的删除P5、添加P2层的有效性，分别使用CCTSDB数据集和TT100K数据集，保持其他条件不变，对比YOLOv11n，YOLOv11s,，修改后的YOLOv11n，YOLOv11s的各项指标，同时加入了在其他场景下性能更加优越的YOLOv12n和YOLOv12s。

在CCTSDB数据集上实验指标对比如图3-3、3-4、3-5和3-6所示（\_dP5\_aP2表示删除P5层且添加P2层）。

图3-3 n模型在CCTSDB上的表现

图3-4 n模型在CCTSDB上的参数量

图3-5 s模型在CCTSDB上的表现

图3-6 s模型在CCTSDB上的参数量

在TT100K数据集上实验指标对比如图3-7、3-8、3-9和3-10所示。

图3-7 n模型在TT100K上的表现

图3-8 n模型在TT100K上的参数量

图3-9 s模型在TT100K上的表现

图3-10 s模型在TT100K上的参数量

### 3.5.1 CCTSDB数据集上的结果分析

在CCTSDB数据集上和YOLOv11n相比，精确度降低1.9%，召回率提升1.4%，平均精度值提升0.6%，参数量为原来的34.2%，和YOLOv12n相比，精确度提升1.3%，召回率提升4.5%，平均精度值提升5.2%，参数量为原来的34.5%，和YOLOv11s相比，精确度降低3.1%，召回率降低0.4%，平均精度降低0.5%，参数量为原来的30.6%，和YOLOv12s相比，精确度降低5.1%，召回率降低4.8%，平均精度值降低2.6%，参数量为原来的31.2%；

分析可得，对于n模型，改进后与YOLOv11n相比除了精确度，其他指标略微提升，参数量大幅减少，与YOLOv12n相比全面提升，对于s模型则略有下降，但整体仍然比n模型更加优秀。

### 3.5.2 TT100K数据集上的结果分析

删除P5层增加P2层的模型，在TT100K数据集上和YOLOv11n相比，精确度降低0.7%，召回率降低3.8%，平均精度值降低2.0%，参数量为原来的34.5%，和YOLOv12n相比，精确度降低1.8%，召回率降低1.7%，平均精度值降低0.3%，参数量为原来的34.8%，和YOLOv11s相比精确度降低10.4%，召回率降低6.6%，平均精度值降低7.9%，参数量为原来的30.7%，和YOLOv12s相比，精确度降低7.5%，召回率降低4.9%，平均精度值降低6.4%，参数量为原来的31.2%。

可以发现，对于n模型而言，所做改进效果显著，在大幅缩减参数量的同时精确度、召回率、平均精度值下降幅度较小，与YOLOv12精度接近；对于s模型，虽然与YOLOv11s相比各指标下降幅度更大，但是较n模型而言除精确度外的指标均有提升，同时参数量仍然远低于其他模型。分析认为，可能是删除P5层后丢失了一些高层语义特征，同时该数据集中大尺寸目标较多导致精度下降。

3.5.3 检测结果展示

模型训练完成后会自动生成的验证集第一个批次的可视化结果，包含真实标注（val\_batch1\_labels.jpg）和模型预测（val\_batch1\_pred.jpg）的对比。真实标注文件用绿色框显示人工标注的边界框和类别，预测文件用蓝色或红色框显示模型的检测结果，附带类别名称和置信度分数。这些图片的生成逻辑是训练过程中周期性抽取验证集的随机选取的样本进行推理，并将结果保存为图片输出。

通过对比预测框与真实框的重叠度，研究者可以方便地直观判断模型在定位和分类上的准确性。若预测框完全覆盖真实目标且置信度高，说明模型性能较好；如果预测框偏移较大或漏检，则需调整模型参数或数据增强策略。如果预测框出现在无真实标注区域，误判无关物体为目标，可能是数据标注质量差或模型过拟合所导致；某类别频繁漏检则应当增加该类别样本的训练样本数量。通过观察不同置信度阈值下的预测框数量变化，可调整阈值来平衡漏检和误检。结合训练日志中产生的损失曲线图片，若验证损失波动大但预测框质量稳定，有可能是数据分布不平衡或学习率设置不合理，事实上，当前开源的交通标志数据集绝大多数都存在样本类型数量不平衡的问题，可能是由于很多交通标志本身较为少见所导致。总之，可以通过自动产生的可视化结果修改训练策略，验证预期结果。下图为部分生成的预测图片。



图3-11 YOLOv11s在CCTSDB上的预测图片



图3-12 YOLOv12s在TT100K上的预测图片

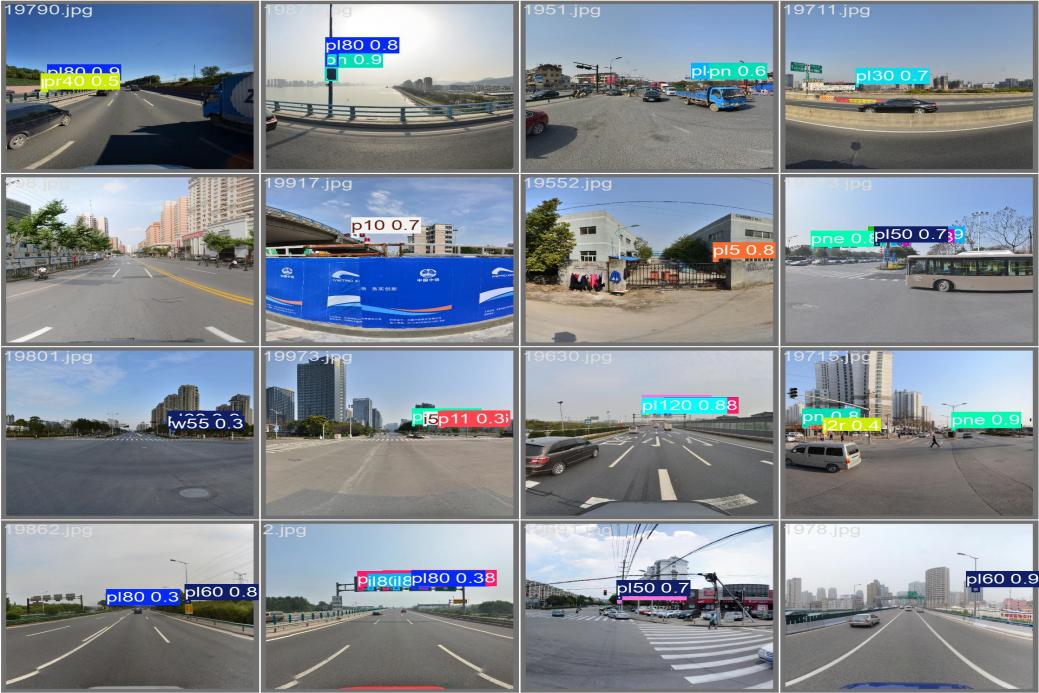


图3-13 YOLOv11n改进后在TT100K上的预测图片

## 3.6 消融实验

3.6.1 消融实验概念

消融实验（Ablation Study）是机器学习和深度学习中一种重要且常见的分析方法，目的在于通过系统性地移除或修改模型的特定组件、特征或策略，来评估其对整体效果的贡献，本质是控制变量法。这种实验方法不仅帮助开发人员理解模型设计的有效性，还为优化模型架构、提升可解释性提供了科学依据。

消融实验这个名字源于医学领域“消融术”的概念，即通过移除特定组织观察其对生理功能的影响。在人工智能领域，其核心思想是通过**控制变量法**，逐步“消融”（移除或禁用）模型的某些部分，量化分析这些部分对性能的影响，这样就可以发现改动对于模型的影响，不过也要注意，某些修改可能会互相作用，单纯进行某些控制变量实验不能充分体现其对整体的影响。

3.6.2 对改进进行消融实验

为进一步验证修改对YOLOv11模型的改进有效性，对所修改的部分在CCTSDB数据集和TT100K数据集上进行消融实验。分别增设只删除P5层，不增加P2层和只增加P2层，不删除P5层的两组模型，在同样的实验条件下进行对比。实验结果如下图所示。（dP5表示删除P5层，aP2表示增加P2层）

图3-14 所有n模型在CCTSDB上的表现

图3-15 所有n模型在CCTSDB上的参数量

图3-16 所有s模型在CCTSDB上的表现

图3-17 所有s模型在CCTSDB上的参数量

图3-18 所有n模型在TT100K上的表现

图3-19 所有n模型在TT100K上的参数量

图3-20 所有s模型在TT100K上的表现

图3-21 所有s模型在TT100K上的参数量

### 3.6.3 只删除P5层

只删除P5层不增加P2层的模型，在TT100K数据集上和YOLOv11n相比，精确度降低6.5%，召回率降低6.5%，平均精度值降低6.6%，参数量为原来的30.9%，和YOLOv11s相比精确度降低3.1%，召回率降低2.5%，平均精度值降低2.7%，参数量为原来的28.7%，在CCTSDB数据集上和YOLOv11n相比，精确度降低2.0%，召回率降低4.2%，平均精度值降低6.9%，参数量为原来的30.8%，和YOLOv11s相比，精确度降低0.8%，召回率降低1.3%，平均精度降低3.0%，参数量为原来的28.6%，虽然参数量大幅减少，模型显著缩小，但精度下降同样明显，漏检误检概率显著提高。

原因可能在于P5层作为特征金字塔的顶层，负责捕获全局语义信息与长距离上下文关联。图像的深层特征通过多次下采样积累了大尺度目标的形态学特征。删除P5层将导致大目标检测依赖P4捕获的中层特征，但中层特征缺乏足够的感受野，无法有效建模大目标的全局结构。删除P5层可能导致目标边界框回归误差增大，造成模型对密集场景中重叠的大目标的区分能力下降，例如在某些角度下多个交通标志牌重叠造成误检。

### 在模型中，特征金字塔的层级设计需保证相邻层级间的尺度过渡平滑。P5层在其中承担着特征抽象过渡功能。删除P5后，P4将直接连接至更高抽象模块，在跨层特征融合时会出现语义跳跃，导致目标尺度预测的连续性受损，造成精度损失。中尺度目标在P4层可能因缺乏高层语义引导而被误判为多个小目标，比如一个含有多个信息的大标志牌可能被当作大量小标志牌。

在目标运动过程中，特征表达需要随尺度变化平滑过渡。P5层的缺失使得模型无法通过高层特征修正中低层特征的误判，导致目标ID切换频率增加。远处车辆因尺度较小被P3层检测，随距离接近需切换至大目标检测层，但P5删除后缺乏中间层支持，易产生检测断层，小目标层的错误结果无法修正，即使接近仍然造成误判，影响检测结果。

### 3.6.4 只增加P2层

只增加P2层不删除P5层的模型，在TT100K数据集上和YOLOv11n相比，精确度降低5.7%，召回率降低6.1%，平均精度降低7.8%，参数量为原来的103.4%，和YOLOv11s相比，精确度降低6.3%，召回率降低6.2%，平均精度值降低6.2%，参数量为原来的101.5%，在CCTSDB数据集上和YOLOv11n相比，精确度降低4.5%，召回率降低4.9%，平均精度值降低4.0%，参数量为原来的103.0%，和YOLOv11s相比，精确度降低1.4%，召回率降低2.5%，平均精度值降低1.5%，参数量是原来的101.6%，在参数量略微增加的同时精度、召回率、平均精度值反而有所下降。

原因可能在于P2层位于网络浅层，其特征包含大量纹理、光照等低级细节。这些特征对小目标检测具有价值，但同时也容易引入背景噪声。树叶投下的阴影可能被误判为小目标边界，增加注意力机制可以对此进行过滤，但这又增加了模型复杂度与训练难度，导致推理速度变慢，增加参数量。

增加P2层后，特征金字塔将涵盖P2-P5等多级尺度，但层级间尺度差异扩大。P2与P3的跨度过大，导致中等尺度目标在进行跨层特征融合时出现信息丢失，发生错误。

浅层P2与深层检测头的梯度方向存在显著差异。P2层梯度侧重于局部细节优化，而检测头将综合全局语义，这种矛盾也会导致最终精度受到影响。

3.6.5 总结分析两种方案

删除深层P5检测层的本质目的是**消解高层语义冗余，避免不必要的计算**。而P5层负责的全局语义信息在自动驾驶摄像场景中易因大范围背景干扰产生特征模糊，移除后可以将计算资源重分配至中低层优化，更有效地利用计算资源。而新增P2层通过高分辨率特征图保留微小目标的边缘纹理与几何细节，结合跨层注意力路由机制与中层特征双向交互，实现细节与语义的双重保证。但P2层的高分辨率特征虽然增强了模型对于小目标检测的能力，但其浅层特征易受背景细节干扰，造成误判。新增P2检测头与保留的P3，P4层之间存在梯度传播方向差异：P2侧重局部细节优化，而P4依赖全局语义引导。这种矛盾会导致训练初期损失震荡加剧等问题；但就最终结果而言，带来的缺点是可以接受的，模型凭借较少的参数维持了较为优秀的精确度，可以认为改进行之有效。

第4章 基于PyQt6的交通标志检测系统

4.1 开发工具

4.1.1 开发环境及配置

本系统使用Pycharm 2024.1.1作为开发环境，Python版本使用3.8，在此前所训练完毕模型的基础上使用PyQt6开发图形界面，并使用PyInstaller将程序打包为可执行文件，便于用户在其他设备上简便安装运行。

4.1.2 PyQt6简介

PyQt是Qt框架的 Python 绑定库,由Digia公司开发，专为开发跨平台图形界面（GUI）应用而设计。它是一个多平台的工具包,支持 Windows、macOS、Linux 等操作系统，代码无需修改即可在不同平台运行。

本文选用PyQt6，原因在于Qt5 已于 2023 年底停止功能更新，而PyQt6使用了基于Qt6重构的RHI引擎，其图形性能提升了30%，尤其适合复杂 UI、3D 渲染或高频数据刷新场景，而且PyQt6引入了**事件循环（event loop）**机制，简化了事件处理逻辑，提升了响应速度，可以用于自动驾驶系统这类对速度要求较高的场景。优化 Windows/macOS/Linux 等主流操作系统的适配性，支持 Android/iOS 的混合开发，所以尤其适合汽车这类移动设备，大大减少了设备不兼容的可能性。

4.1.3 PyInstaller简介

PyInstaller 是一个跨平台的 Python 打包工具，能够将 Python 脚本及其依赖库转换为独立的可执行文件，支持 Windows、macOS 和 Linux 系统，同样能够减少设备不兼容的可能性。生成的可执行文件无需安装 Python 环境即可运行，适合分发给普通用户。支持单文件或文件夹两种打包模式，单文件模式便于分发，文件夹模式便于以文件形式获取程序运行所产生的输出结果，自由修改部分配置文件。且具有隐藏控制台窗口选项，软件开发者可以保留控制台窗口，便于查看程序输出信息进行调试。PyInstaller适配PyQt，避免开发时考虑兼容性问题。故选用PyInstaller进行打包。

4.2 系统需求分析

随着无人驾驶汽车以及智能辅助驾驶技术迅速发展，交通标志识别技术逐渐成为当前计算机视觉方向的研究热点，为了满足自动驾驶汽车的需求，需要开发一款交通标志检测系统，能够为用户提供检测图片与视频中交通标志的功能，由于实际路况往往比较复杂，环境多变，同一种识别模型不能适配所有任务，所以该系统需要能够支持用户自由切换识别模型，同时提供简洁美观，易于操作的图形界面供非专业人士使用。

该系统应当提供图片检查和视频检测功能：待检测样本来源多样，可能来自行车记录仪或是车载自动驾驶摄像头，其传入格式既可能为图片，又可能为视频，且文件后缀往往不同，需要适应多种格式，如png,jpg,mp4等。

自由选用配置的模型：系统应当配备多种可用模型并提供自由选择模型的功能，以适应不同道路场景的需要，切换方式应当流畅，通知用户切换完毕。

自动保存检测结果：对于检测后的结果必须保存至存储设备，以供开发者和普通用户观察评估检验效果，以及作为行车记录仪在交通事故后追责。

同时存在一些非功能需求：

精确性：交通标志检测作为自动驾驶技术中重要的一环，任何漏检误检行为都将严重危害乘客和公众的生命财产安全，该系统提供的检测服务必须具有高度精确性，保证检测结果的可信。

及时性：机动车行驶速度往往较快，其配备的摄像头为了捕捉到清晰画面，图像必定具有较大数量且清晰度较高，而在行车时任何不必要的时延都会减少驾驶员的反应时间，危害生命安全，所以要求检测必须快速稳定，及时处理到达的数据。

环境适应性：实际路况往往较为复杂，天气、光照环境都会影响检测精度，系统的检测服务必须适应各种天气和光照，对遮挡重叠，变色目标也具有较好的检测精度。

简便性：系统功能较少，其界面设计必须简洁易懂，减少用户的困惑和学习成本。

4.3 系统设计

4.3.1 系统总体设计

该系统总的来讲，具有两个大的模块，分别是用户交互界面模块和后台检测模块。用户交互界面展示给用户交互，显示检测信息，提供功能选择。后台检测模块为核心部分，接受控制对文件进行检测识别，并自动保存检测结果。

4.3.2 用户交互界面模块设计

用户交互界面模块采用PyQt6进行图形界面开发，设计一个简单，美观的界面，有如下规划设计：

设置选择图片，选择视频，检测图片，检测视频按钮，点击后触发选择文件和检测事件，激活检测模型。

模型选择下拉框，点击后出现模型选择列表，点击后检测模型切换至所选模型。

待检测图片预览，检测结果展示两个图片显示框，用于对图片进行预览和展示，让用户不必到保存文件夹中观看检测结果，对于视频不提供此功能；

若干提示文字文本框，用于提示用户当前操作是否完成。

4.3.3 后台检测模块设计

后台识别模块依赖前文提到的改进的YOLOv11模型，有如下规划设计：

接受图形界面传入信息，直接被调用激活，随点随用。

配备多种训练完毕的模型，涵盖经过专用数据集训练后的官方模型以及改进后的YOLOv11模型，接受到信息后即进行切换。

具有视频检测和图片检测两种功能，对于图片进行检测后告知文件路径，以便图形界面进行展示，对于视频同样提供检测，且由于视频的每一帧检测时无需模型装载卸载时间，所以速度更快。

自动保存，将图片的检测结果自动保存到程序附带的文件夹中，若文件夹不存在则自动创建；视频由于体积较大，则自动保存至原路径下。

4.4 界面开发实现

4.4.1 界面开发原则

为了保证用户的使用体验，本系统图形界面开发遵循以下原则：

界面简化：设计界面时避免过多功能，去除冗余元素。版面结构尽可能清晰简单，避免使用过多色彩搭配，避免提供过多不常用功能导致用户困惑。重点保留核心功能模块，通过视觉留白引导用户快速找到关键操作区域。

操作便捷：优化用户操作路径，将常用功能置于显眼区域。预判用户点击习惯，减少点击次数和光标移动距离，避免繁琐的点击操作，关键按钮放大处理并配有文字说明，使用区分度高的颜色引起关注。限制用户输入避免引发错误，在选择文件时不予显示不支持的文件类型。

风格统一：保持整体视觉风格一致，比如图标样式、字体大小、按钮形状等元素应当相同或相似。相同功能模块在不同页面应保证交互方式相同或相似，导航栏位置和操作反馈方式必须保持全局统一，避免用户感到明显差异感，降低用户的学习成本。

反馈信息：建立明确的交互响应机制，及时提示用户错误信息，操作成功应弹出明显提示，对于运行中的操作应当告知用户，尽可能提供进度条。关键操作需二次确认，文件相关操作应当注明文件路径。

4.4.2 界面开发

图形界面分为五个部分，包括标题、功能按钮、下拉菜单、图片显示区、提示文本。

界面主体使用PyQt6中的MainWindows窗体，在窗体正上方设置标题，下设文件操作区域，包含“选择图片”和“选择视频”两个按钮，点击后即弹出文件选择窗口，窗口来自PyQt自带的QFileDialog，受程序设计限制，显示出的文件仅包含本系统可以检测的格式，避免用户错误操作造成程序崩溃，按钮右侧有文本提示，汇报是否已选中文件。下方为模型选择区域，左侧为当前模型名称，右侧为选择下拉列表，点击后即弹出选择菜单。下方为图像检测结果，左侧框为待检测图像，用于比对检测结果，右侧在检测完毕后会出现检测结果，显示检测到的目标框，类型名称，置信度。最下方为检测控制区，包含图像检测与视频检测按钮，点击后即开始检测选中文件，右侧文本提示检测完毕的结果保存位置。图4-1为图形界面展示。

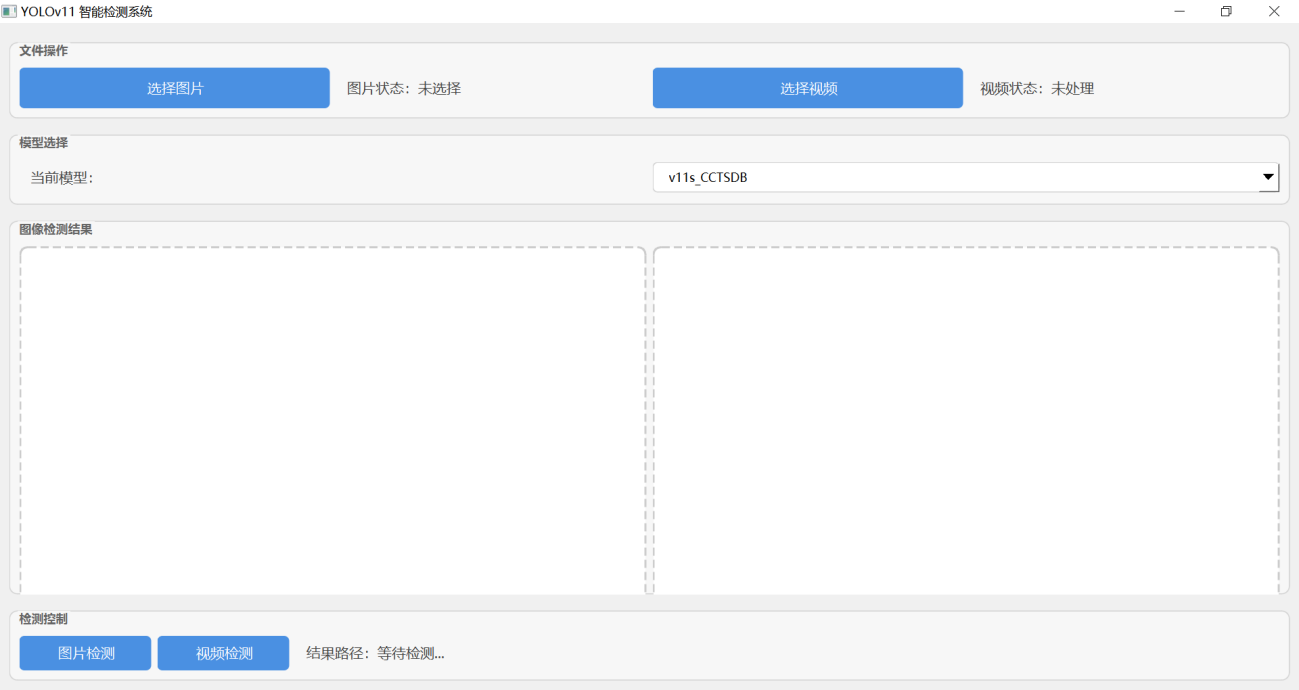


图4-1 图形界面展示

4.5 检测模型实现

4.5.1 数据集选择

数据集选择CCTSDB数据集和TT100K数据集。

CCTSDB数据集包括14664张训练集，1692张测试集，获取时已经转换为YOLO格式。

TT100K数据集包括5962张训练集，2979张测试集，获取时已经转换为YOLO格式。

4.5.2 训练参数设置

训练参数同3.2实验环境，训练迭代次数初始均设定为300轮，使用官方提供的预训练参数，批次大小为200，线程数45，输入图像分辨率大小为640\*640，使用SGD优化器训练模型，开启自动混合精度训练和缓存选项，其他使用默认值。

4.5.3 模型训练结果

训练得到共计8个模型，分别是YOLOv11n，YOLOv11s，YOLOv11n改进型，YOLOv11s改进型各自在CCTSDB和TT100K数据集上的训练结果。其对应的P-R曲线（准确度-召回率曲线）如图4-2、4-3、4-4、4-5、4-6、4-7、4-8和4-9所示。

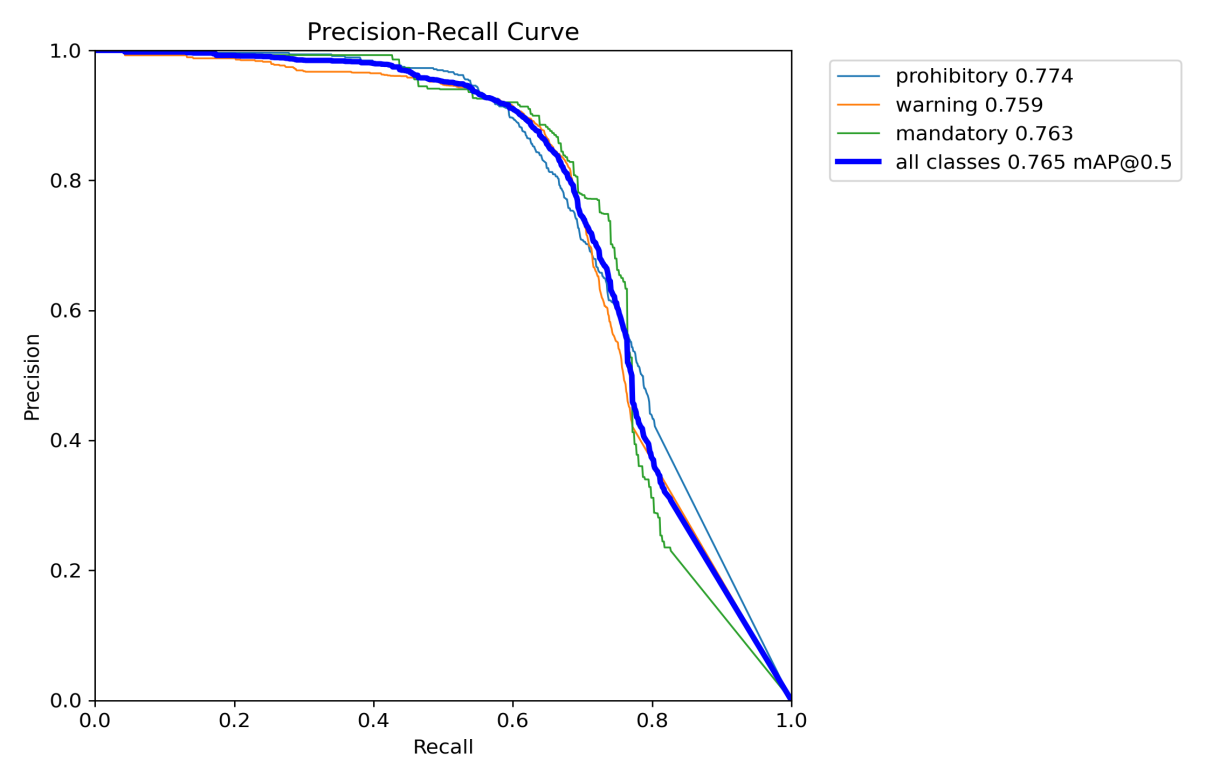


图4-2 n模型在CCTSDB上的P-R曲线

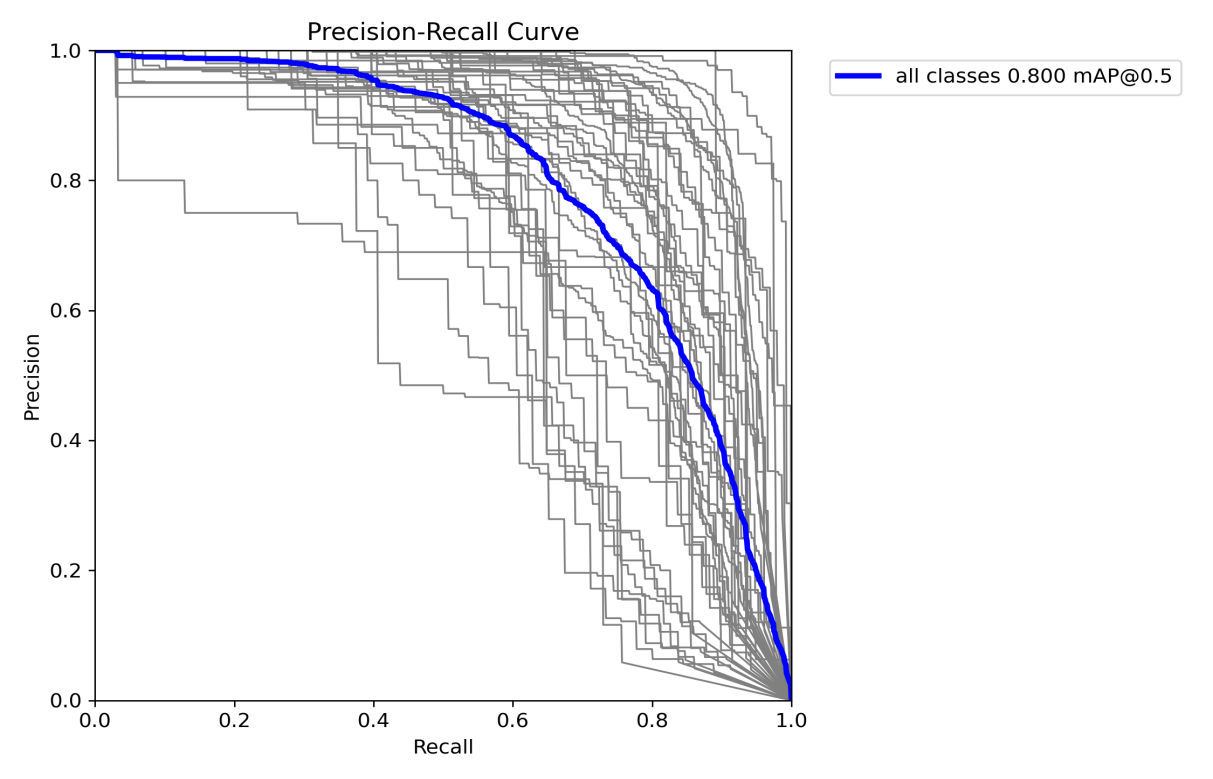


图4-3 n模型在TT100K上的P-R曲线

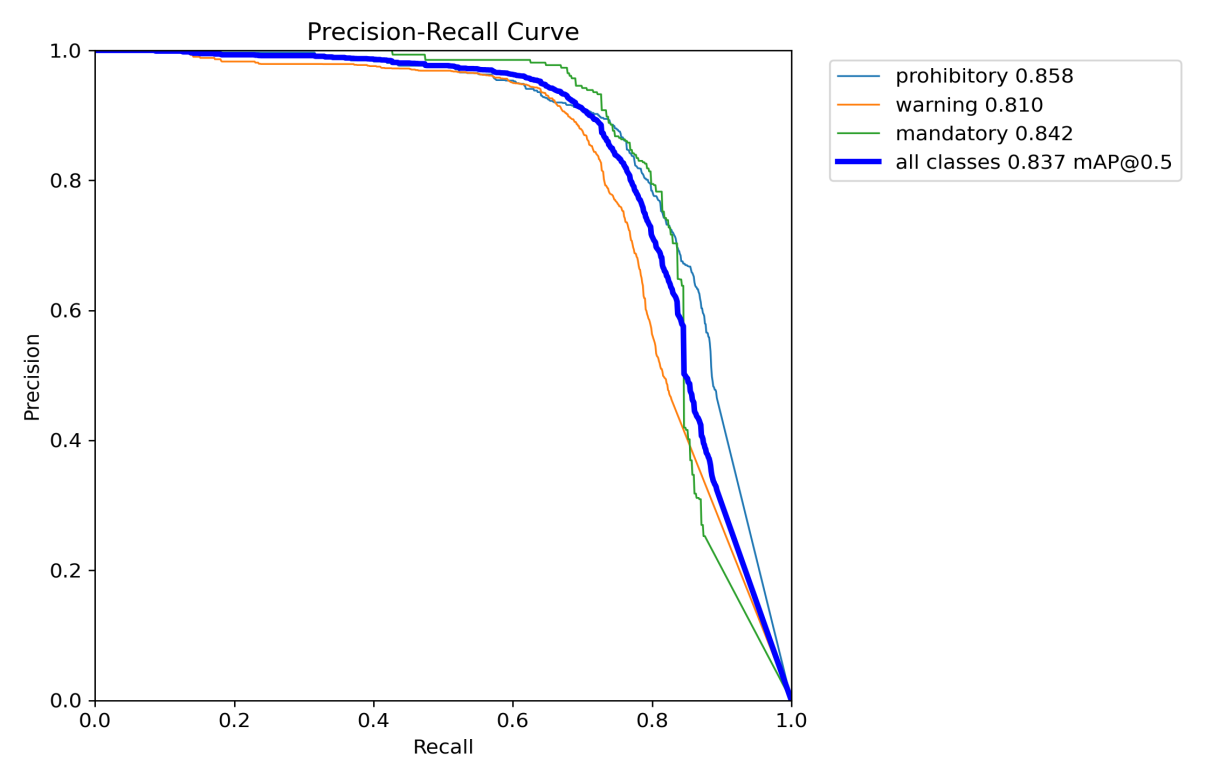


图4-4 s模型在CCTSDB上的P-R曲线

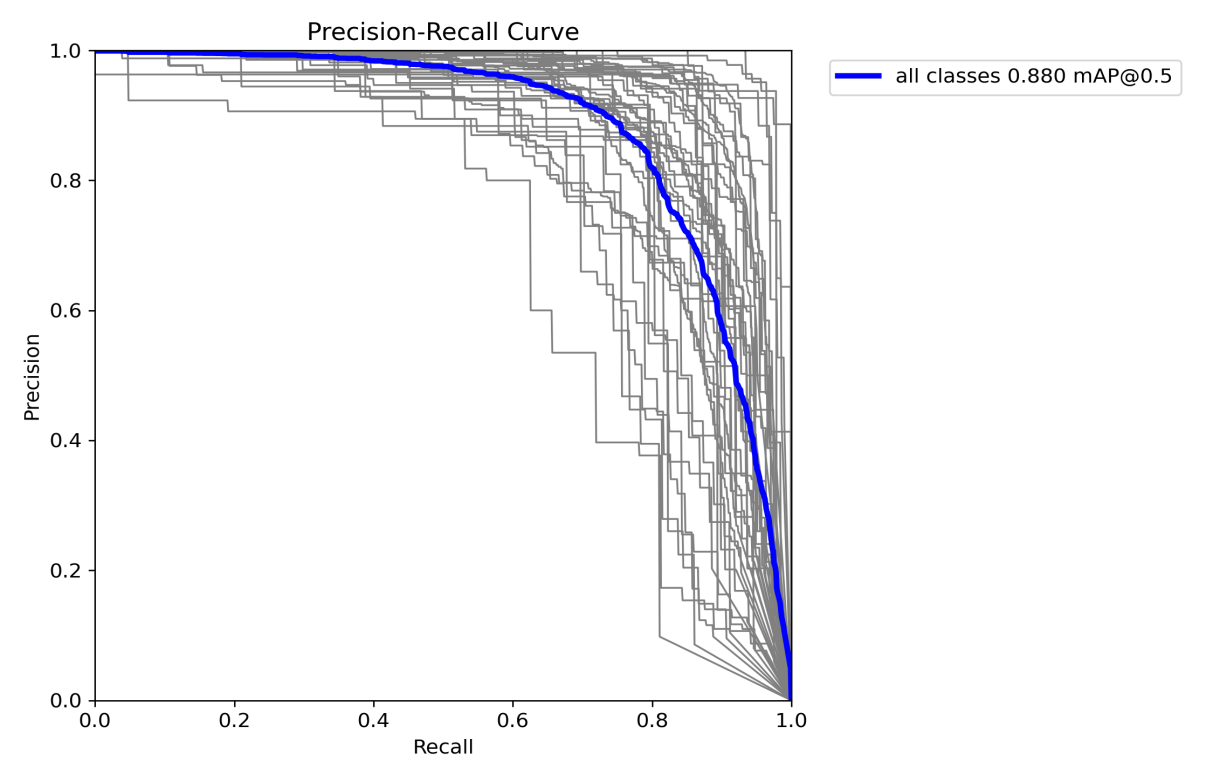


图4-5 s模型在TT100K上的P-R曲线

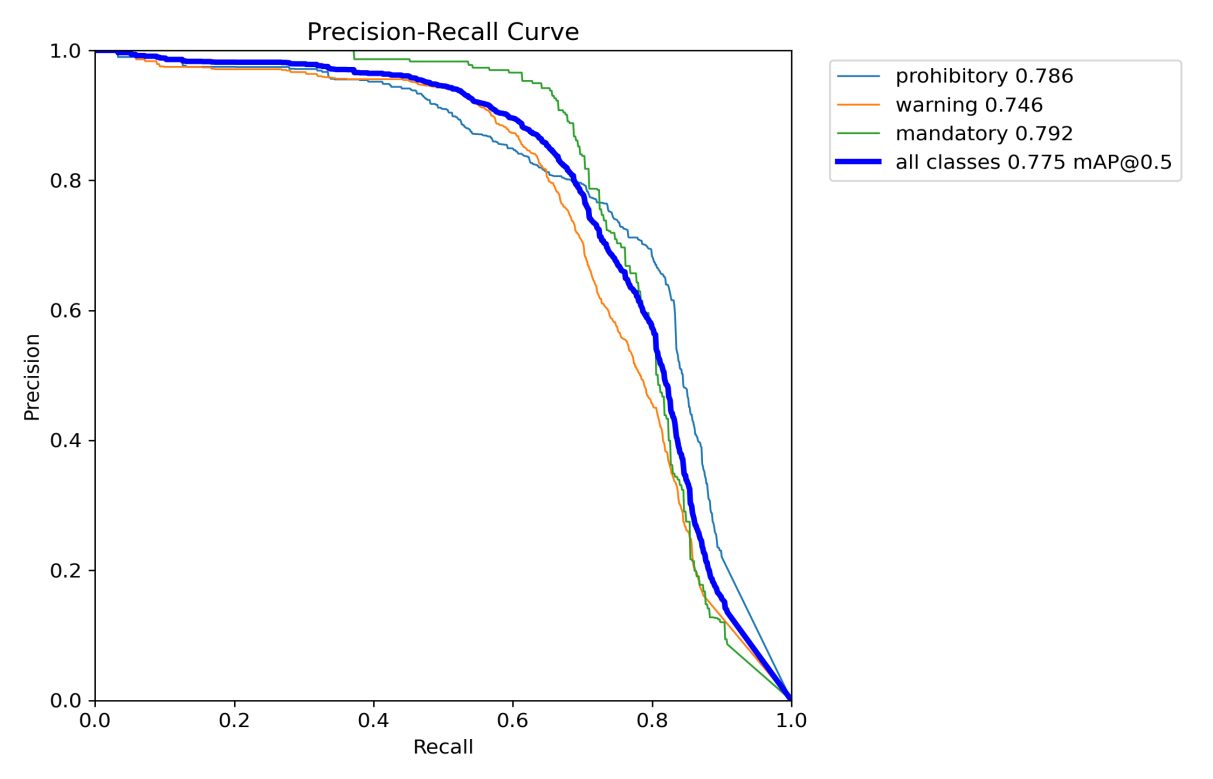


图4-6 改进后的n模型在CCTSDB上的P-R曲线

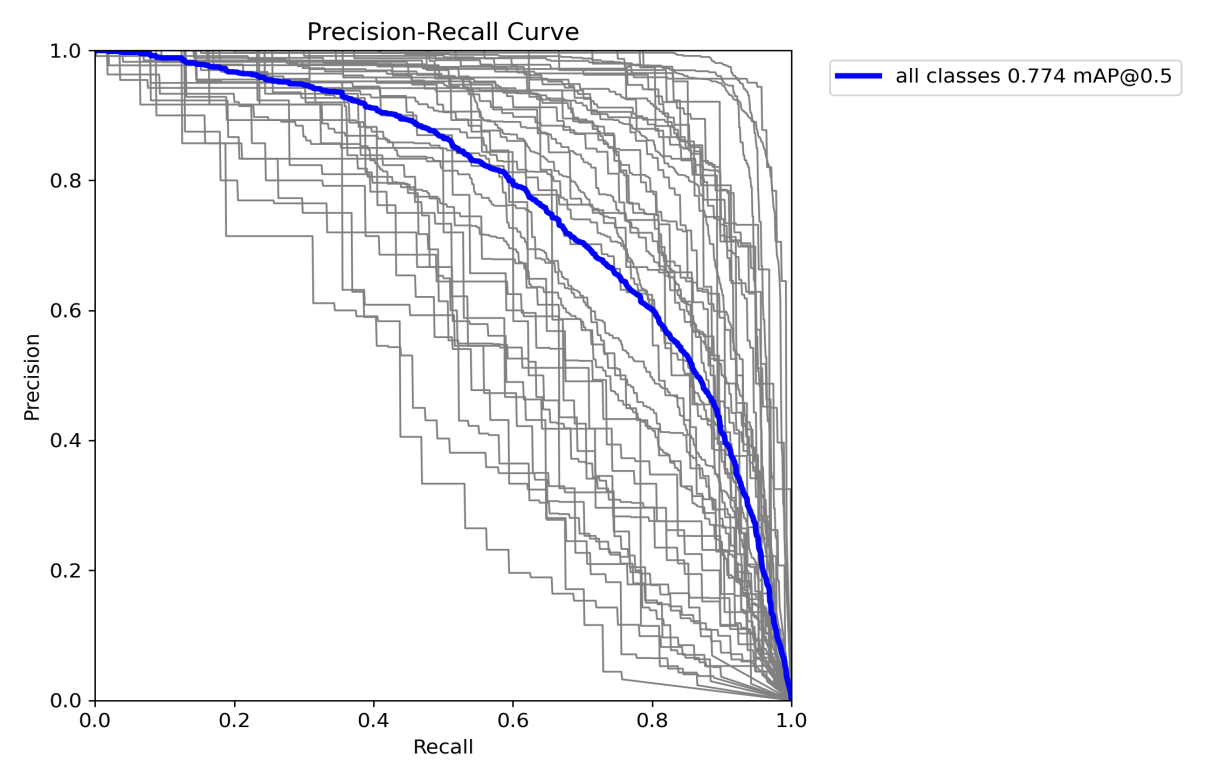


图4-7 改进后的n模型在TT100K上的P-R曲线

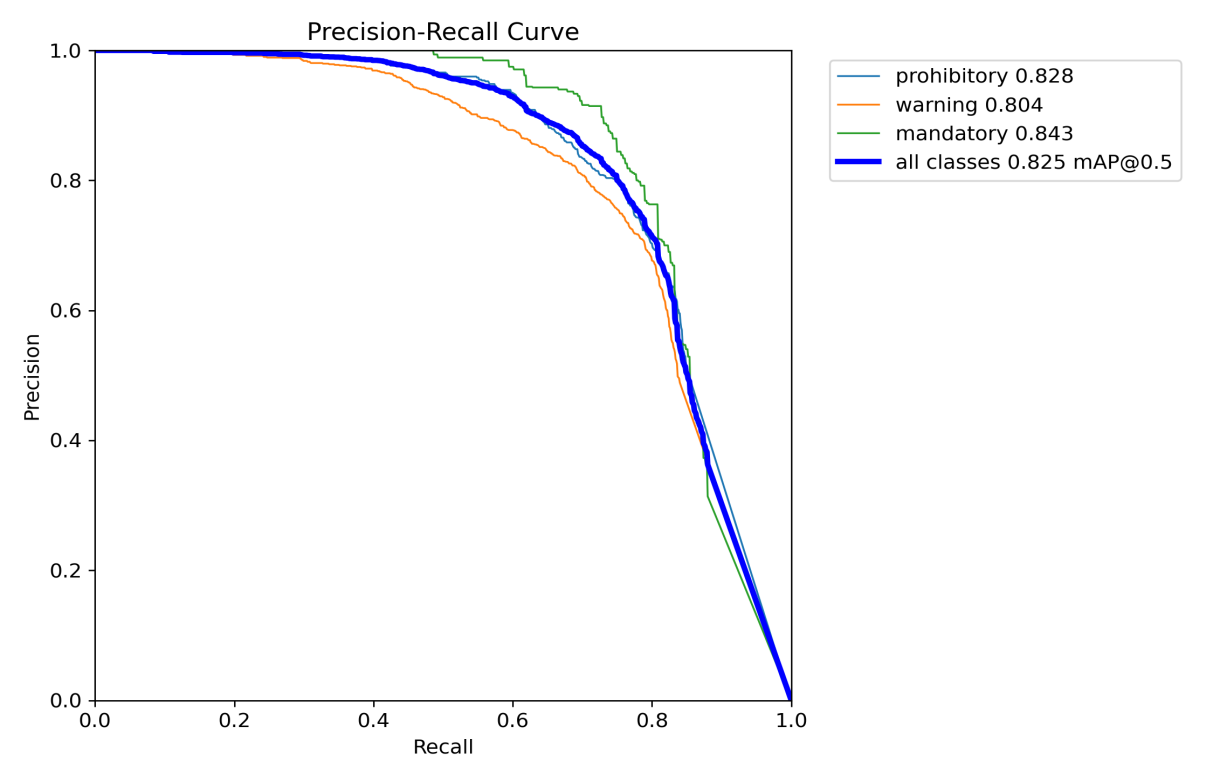


图4-8 改进后的s模型在CCTSDB上的P-R曲线

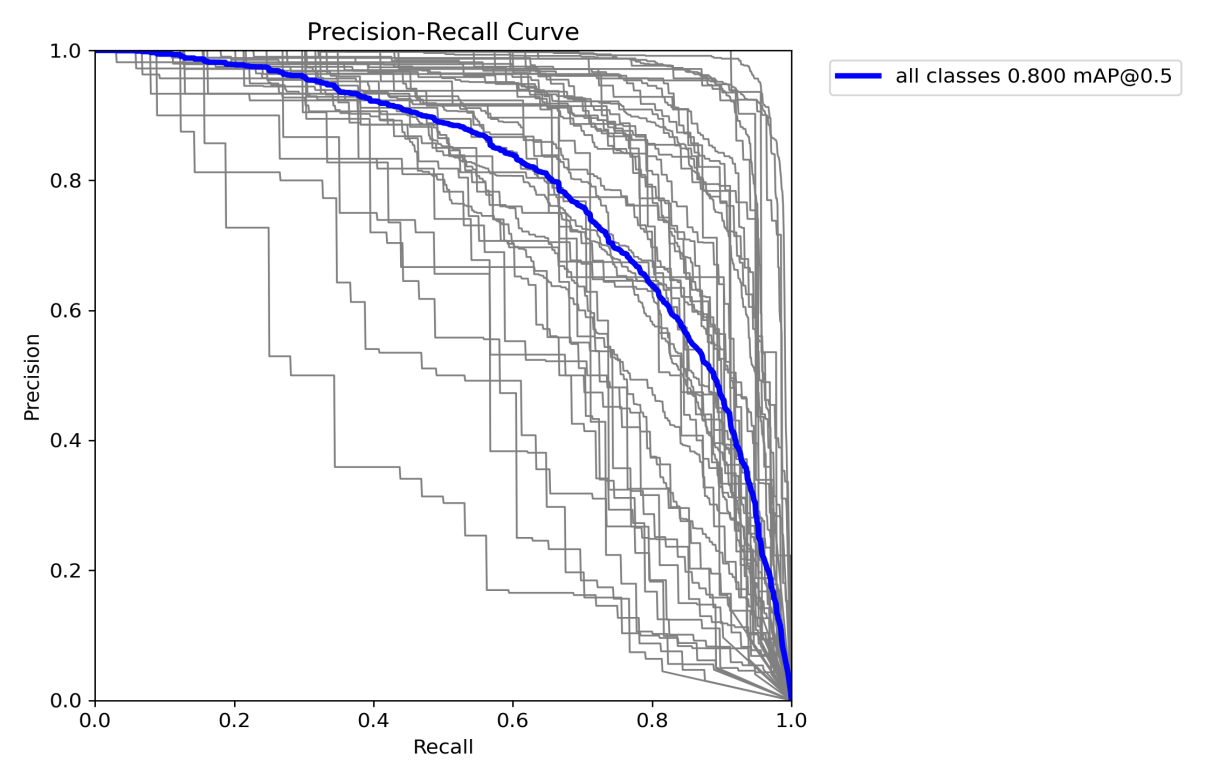


图4-9 改进后的s模型在TT100K上的P-R曲线

4.6 代码实现和运行效果展示

当用户点击“选择图片”按钮时，将会弹出文件选择窗口，其中只显示png,jpg,jpeg文件，点击“选择视频”按钮时，文件选择窗口将只显示mp4,avi,mov文件，选择目标文件后会在下方图像检测结果区域内预览图片（视频无法显示），QFileDialog将会获取用户选择的文件路径，自动修改图片尺寸适配显示框；下方的下拉列表可以通过点击切换装配的模型；点击“图片检测”按钮后将会按照选择的模型对图片进行检测，检测结果将在图像检测结果区域展示，包括目标框，置信度，类别名称在内的信息一并显示，并自动保存到按钮右方显示的路径中；点击“视频检测”后将会对视频进行检测，生成的结果将被直接保存到原视频目录下，不再进行展示；为保证程序可靠性，在选择文件时仅显示支持的格式，防止用户误操作造成崩溃，保证系统安全。

下面展示部分代码，限于篇幅，此处仅展示具有代表性的片段。

选择图片按钮后触发事件代码如下：

|  |
| --- |
| def select\_image(self):  options = QFileDialog.Option.DontUseNativeDialog  file\_path, \_ = QFileDialog.getOpenFileName(  self, "选择图片文件", "",  "图片文件 (\*.jpg \*.png \*.jpeg)", options=options  )  if file\_path:  self.source\_img = file\_path  self.img\_status.setText(f"图片状态：已选择")  pixmap = QPixmap(file\_path).scaled(  600, 400,  Qt.AspectRatioMode.KeepAspectRatio,  Qt.TransformationMode.SmoothTransformation  )  self.img\_preview.setPixmap(pixmap) |

该函数开始执行后，触发QFileDialog窗体，DontUseNativeDialog禁止使用本地系统自带的对话框，而是使用PyQt提供的对话框。设置对话框名称为“选择图片文件”，文件类型设置为“图片文件(\*.jpg \*.png \*jpeg)”，限制显示和可选中的文件仅限图片文件，避免误操作。选中后将文件路径记录入file\_path，如果file\_path被赋值，则将其值传递给窗体本身的图片路径，并使用QPixmap获取图片，设置其尺寸，最终布置到img\_preview中，显示给用户。

处理视频相关代码如下：

|  |
| --- |
| def process\_video(self):  if not self.source\_video:  self.save\_label.setText("错误：请先选择视频文件！")  return  try:  self.save\_label.setText(f"开始处理，正在配置模型")  base\_dir = os.path.dirname(os.path.abspath(\_\_file\_\_))  model\_dir = os.path.join(base\_dir, "pt")  model\_name = self.model\_combo.currentText() + ".pt"  self.model\_path = os.path.join(model\_dir, model\_name)  model = YOLO(self.model\_path)  self.save\_label.setText(f"模型加载完毕，获取视频参数")  cap = cv2.VideoCapture(self.source\_video)  # 视频参数获取  fps = cap.get(cv2.CAP\_PROP\_FPS)  width = int(cap.get(cv2.CAP\_PROP\_FRAME\_WIDTH))  height = int(cap.get(cv2.CAP\_PROP\_FRAME\_HEIGHT))  # 创建输出目录  save\_dir = os.path.join(os.path.dirname(self.source\_video), "custom\_predict\_vedio")  os.makedirs(save\_dir, exist\_ok=True)  output\_path = os.path.join(save\_dir, f"pred\_{os.path.basename(self.source\_video)}")  # 初始化视频写入器  fourcc = cv2.VideoWriter\_fourcc(\*'mp4v')  out = cv2.VideoWriter(output\_path, fourcc, fps, (width, height))  # 处理视频帧  self.save\_label.setText(f"视频处理中，请不要操作！")  while cap.isOpened():  ret, frame = cap.read()  if not ret: break  results = model(frame)  annotated\_frame = results[0].plot()  out.write(annotated\_frame)  # 释放资源  cap.release()  out.release()  self.result\_video = output\_path  self.video\_status.setText(f"视频处理完成\n保存路径：{output\_path}")  self.save\_label.setText(f"最新操作：视频检测完成")  except Exception as e:  self.video\_status.setText(f"处理失败：{str(e)}")  self.save\_label.setText(f"错误：视频检测失败") |

用户点击“视频检测”按钮后此函数执行，首先检查是否已经选中视频文件，若视频不存在则拒绝进行检测并修改文本提示。此后进入权重文件配置阶段，鉴于程序位置不定，这里不硬编码文件路径，而是在程序中获取当前所在的路径，将路径拼接，pt文件已经打包进入程序文件夹，路径拼接后加入pt文件夹，然后获取此前模型选择时的文本框内容，将其后缀拼入“.pt”，即可得到模型权重文件位置，然后即可在model = YOLO(self.model\_path)加载模型，此时模型已经载入，将视频转换为YOLO可以接受的形式即可检测。使用OpenCV库提供的函数获取视频文件句柄，通过句柄获取视频的帧率，尺寸等参数，由于视频往往占用存储空间较大，故不再将检测完毕的视频保存于程序文件夹内部，而是放置于原视频目录下，获取原视频目录，将其保存为视频输出目录。设置一个mp4v格式的视频编码器用来作为输出，将编码器，输出路径，原视频尺寸传入视频写入器。接下来处理视频帧，视频本质上是连续播放的大量图片，所以可以依据其帧率将视频拆分为图片，对拆出的图片进行预测，使用OpenCV提供的库函数将其送入视频写入器，作为视频保存至目标路径下。视频处理完毕后释放句柄，更新提示文本，结束处理。图4-10、4-11、4-12和4-13为运行结果展示。

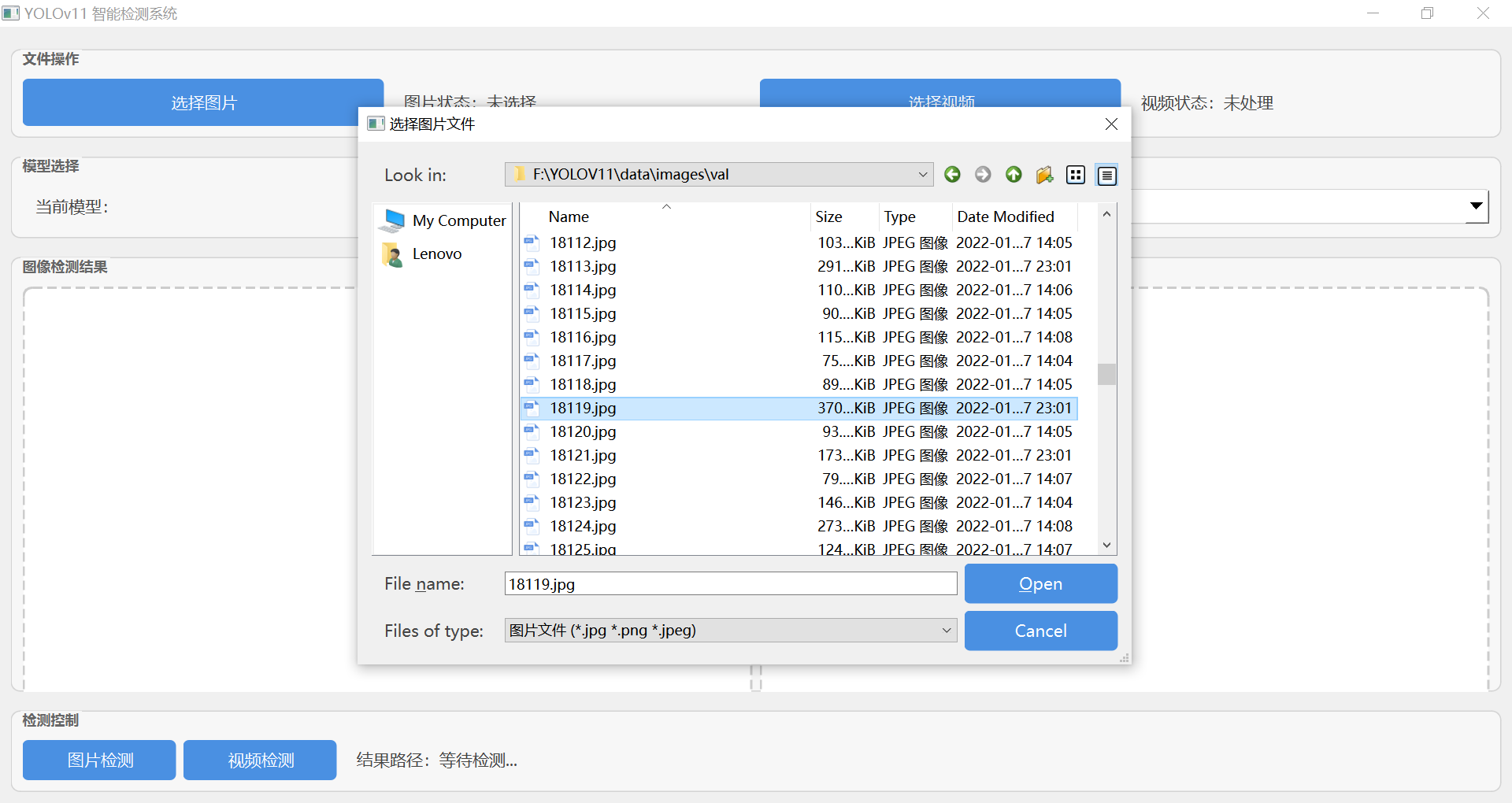


图4-10 选择图片文件界面

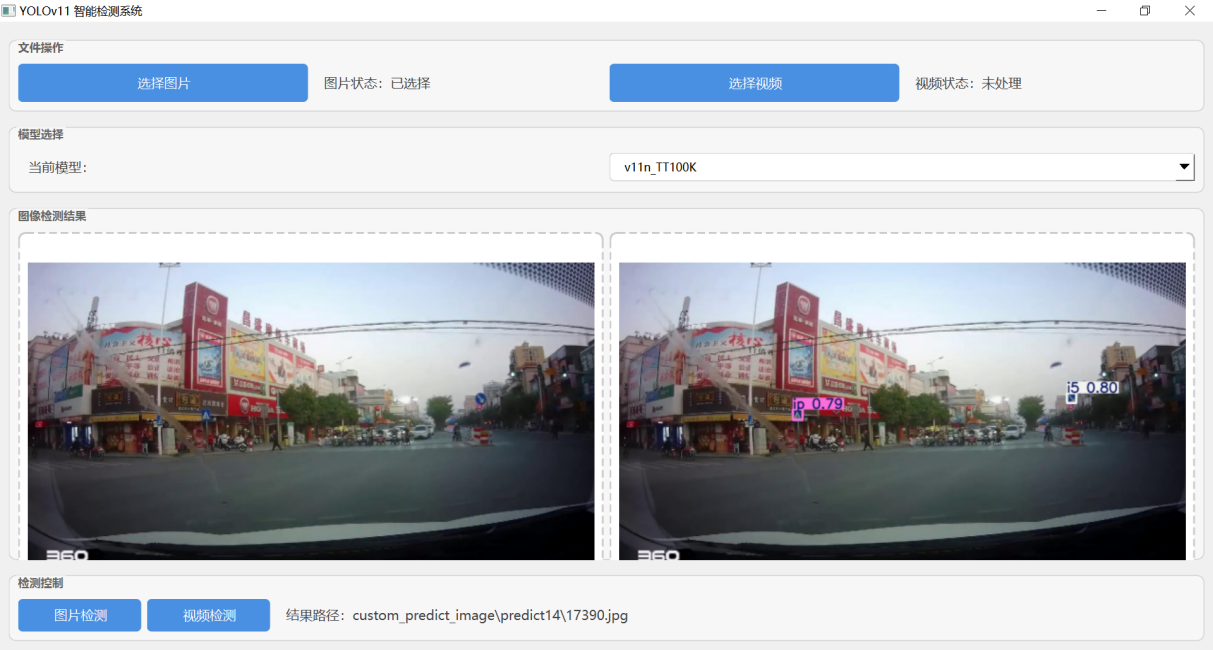


图4-11 图片检测效果

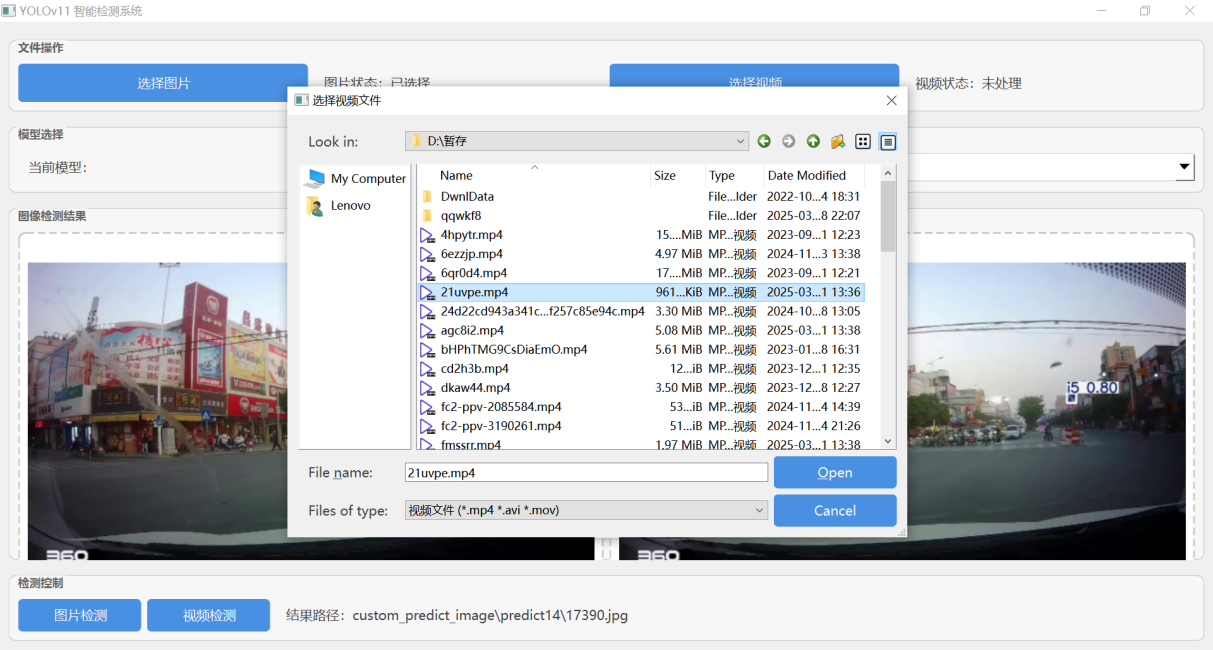


图4-12 视频选择界面

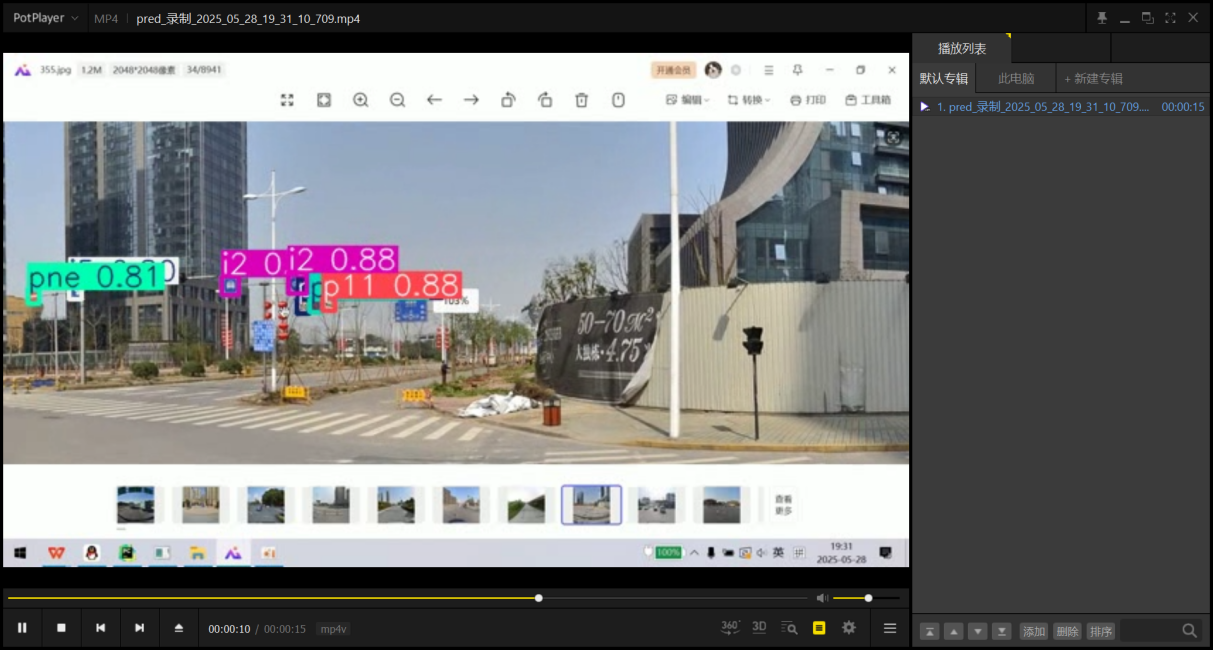


图4-13 视频检测结果

4.7 程序打包分发

系统开发完毕后，为了安装至实际车辆和便于开发人员调试，将该系统程序使用PyInstaller打包为可执行文件（.exe文件）。

PyInstaller支持单文件打包和文件夹打包方式，考虑到本系统会将图片检测结果保存至特定文件夹下，单文件方式将得不到图片且不便于更新版本，修改内部配置，采用文件夹打包，打包结果为build文件夹（记录运行日志，可删除），dist文件夹，该文件夹内部包括程序入口（.exe），点击后即可运行，和一个配置文件夹，该文件夹内即为相关配置文件。

PyInstaller同时支持隐藏控制台信息功能，考虑到使用者既有非专业人士又有相关开发者，故提供两个版本，分别是具有控制台输出和没有控制台输出的版本。

故使用以下命令输入终端进行打包：

pyinstaller --noconsole --add-data "F:\YOLOV11\ultralytics-main\QT\_yolo\pt;pt" --add-data "F:\YOLOV11\ultralytics-main\ultralytics\cfg;ultralytics\cfg" jiemian.py

pyinstaller --add-data "F:\YOLOV11\ultralytics-main\QT\_yolo\pt;pt" --add-data "F:\YOLOV11\ultralytics-main\ultralytics\cfg;ultralytics\cfg" jiemian.py

4.8 系统测试

本系统功能较少，可交互操作有限，但出于软件质量考虑，仍然进行少量测试。

测试目的：测试系统能否对视频和图片正确检测

测试方法：黑盒测试

测试环境：PyCharm

测试用例：如表4-1所示

测试效果：如图4-14和4-15所示

表4-1 测试用例表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 测试用例编号 | 图片是否选中 | 视频是否选中 | 点击操作 | 预期结果 |
| 1 | 是 | 是 | 点击图片检测 | 成功检测图片 |
| 2 | 是 | 是 | 点击视频检测 | 成功检测视频 |
| 3 | 否 | 是 | 点击图片检测 | 提示无图片 |
| 4 | 是 | 否 | 点击视频检测 | 提示无视频 |

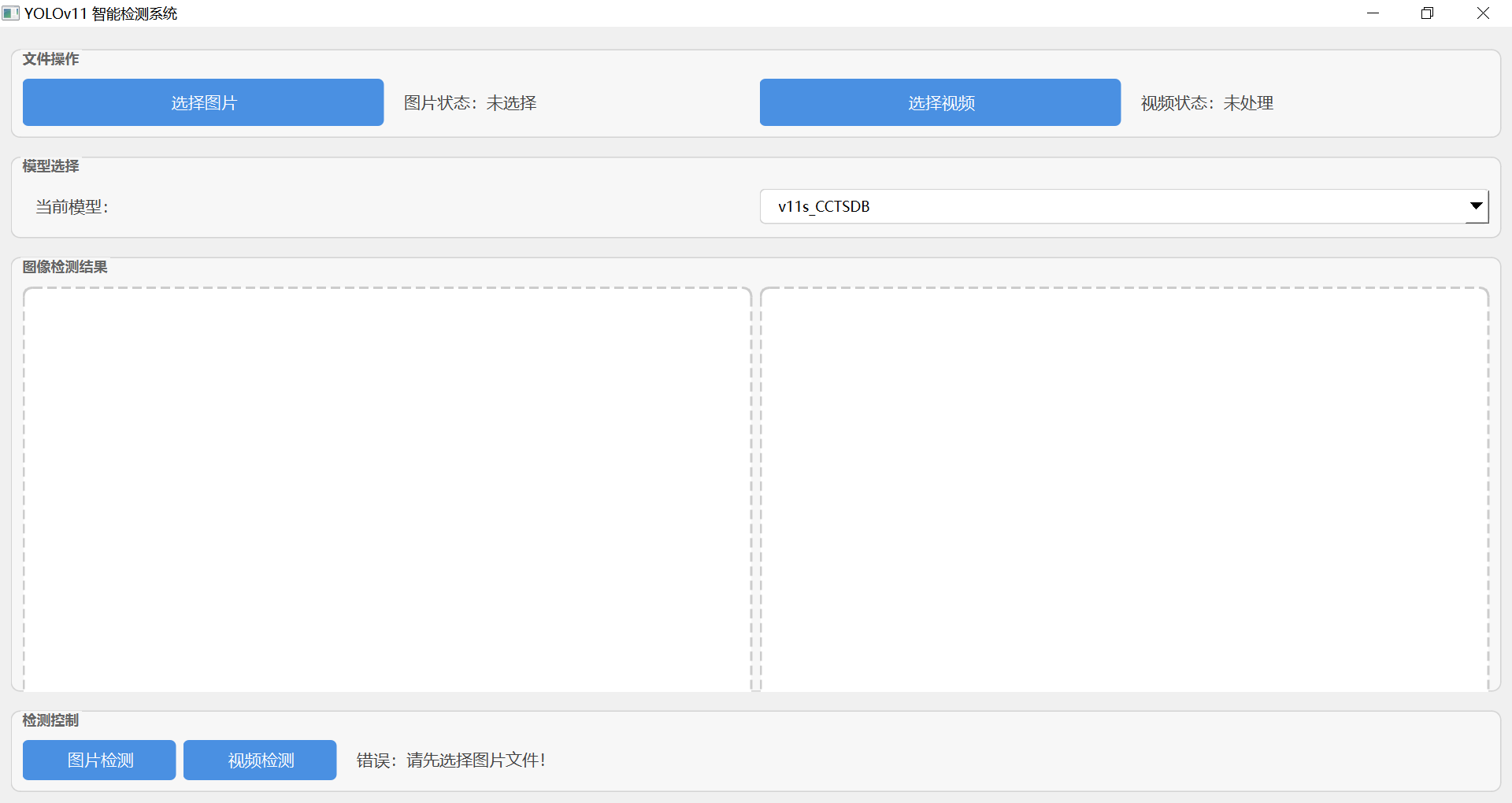


图4-14 图片为空

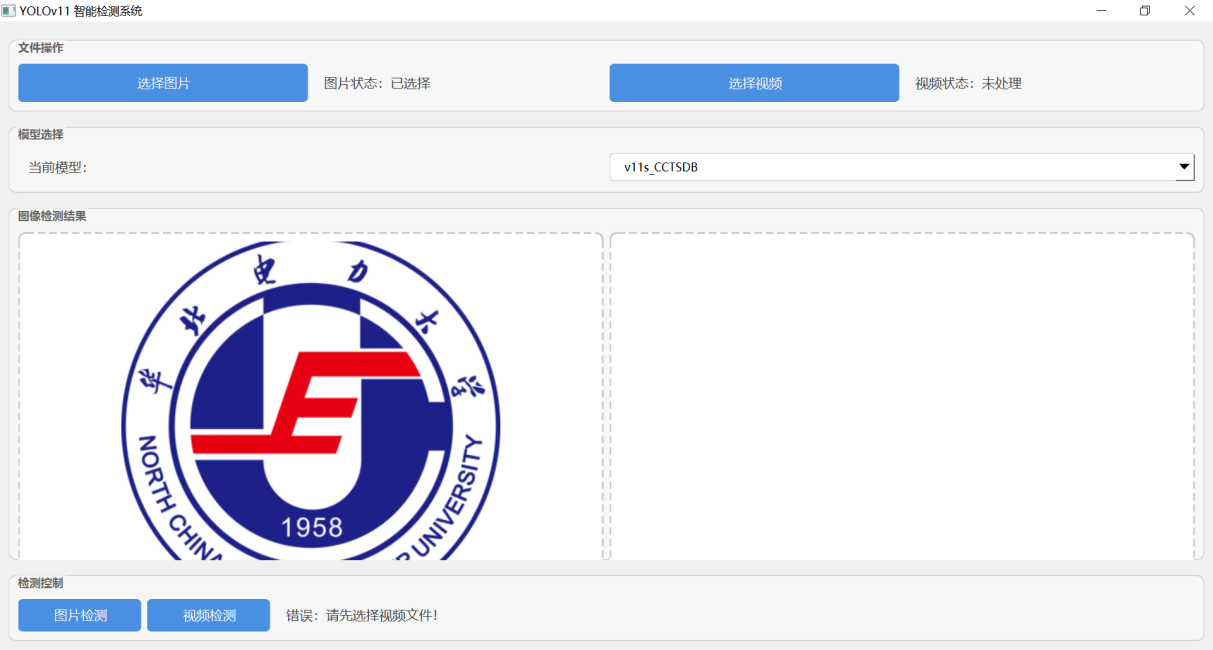


图4-15 视频为空

# 第5章 总结与展望

随着智能辅助驾驶系统的发展，交通标志的识别模型成为当前研究热点。本文设计了一种基于YOLOv11模型的交通标志检测模型。考虑到车载计算资源、行驶时对速度和精度的要求，需要模型的轻量化和性能之间达到平衡[29]。通过修改网络结构，在低层添加一个P2检测层，增强对小目标特征信息的提取能力，以适应实际路况；删除高层的P5检测层，降低对大目标的计算资源倾斜，以节省计算时间，减少参数量，保证检测结果的实时性。经过比对，认为删除P2层，添加P5层后的YOLOv11模型最能满足实际需求。并基于PyQt6，选中训练完毕的模型开发交通标志检测系统，为用户提供图形界面以便操作和调试，使用PyInstaller将程序打包为可执行文件，以便于分发至其他用户本地运行，以及安装于实车进行使用。

修改后的模型计算速度更快，但精度等指标略逊于原模型，仍然存在改进空间，例如恶劣天气、遮挡污损下的漏检，相似小目标误检等情况，未来的工作应当进一步改良网络结构，适度增加参数量来弥补缺点，优化数据集，添加更多更全面的样本来增强检测能力。

参 考 文 献

1. HE Kaiming, GKIOXARI G, DOLLÁR P, et al. Mask R-CNN[C]//2017 IEEE International Conference on Computer Vision. Venice: IEEE, 2017: 2980-2988.
2. REN Shaoqing, HE Kaiming, GIRSHICK R, et al. Faster R-CNN: towards real-time object detection with region proposal networks[J]. IEEE transactions on pattern analysis and machine intelligence, 2017, 39(6): 1137-1149.
3. LIU Wei, ANGUELOV D, ERHAN D, et al. SSD: single shot MultiBox detector[M]//Lecture Notes in Computer Science. Cham: Springer International Publishing, 2016: 21-37.
4. REDMON J, DIVVALA S, GIRSHICK R, et al. You only look once: unified, real-time object detection[C]//2016 IEEE Conference on Computer Vision and Pattern Recognition. Las Vegas: IEEE, 2016: 779-788.
5. 徐伟峰,雷耀,王洪涛,等.面向边缘设备的目标检测模型研究[J/OL].智能系统学报,1-11[2024-12-30].http://kns.cnki.net/kcms/detail/23.1538.TP.20241211.1852.004.html.
6. 杨子房,袁家政,徐成,等.基于YOLO系列的目标检测研究进展综述[C]//中国计算机用户协会网络应用分会.中国计算机用户协会网络应用分会2024年第二十八届网络新技术与应用年会论文集.北京联合大学北京市信息服务工程重点实验室;北京联合大学机器人学院脑与认知智能北京实验室;北京开放大学;,2024:5.DOI:10.26914/c.cnkihy.2024.047826.
7. SHIVAYOGI A B, DHARMENDRA N C M, RAMAKRISHNA A M, et al. Real-time traffic sign recognition using deep learning [J]. Pertanika Journal of Science and Technology, 2023, 31(1): 137-148.
8. 向雷,蒋文波.改进YOLOv8的城市行车道路障碍物检测算法研究[J/OL].电子测量技术,1-11[2024-12-30].http://kns.cnki.net/kcms/detail/11.2175.TN.20241220.1359.027.html.
9. 金超.基于YOLO的交通标志检测方法研究[D].杭州电子科技大学,2024.
10. 黄智渊,方遒,郭星浩.融合注意力机制的YOLOv8-TS交通标志检测网络[J].现代电子技术,2025,48(01):179-186.DOI:10.16652/j.issn.1004-373x.2025.01.030.
11. REDMON J.FARHADI A.YOLO9000:better,faster,stronger[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,2017:7263-7271.
12. REDMON J,FARHADI A.YOLOv3:an incremental improvement[EB/OL]. (2018-04-08). https://arxiv.org/abs/1804.02767v1.
13. BOCHKOVSKIY A.YOLOv4:Optimal Speed and Accuracy of Object Detection[EB/OL].arXiv:2004.10934,2020.
14. 李睿鑫,张应迁,吴嘉懿,等.YOLOv5改进综述[J].电脑知识与技术,2024,20(27):19-22.DOI:10.14004/j.cnki.ckt.2024.1388.
15. LI C,LI L,JIANG H,et al.YOLOv6:A single-stage object detection framework for industrial applications[J],arXiv:2209.02976,2022.
16. WANG G Y,BOCHKOVSKIY A,LIAO H Y M.YOLOv7:Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors[C]/Proceedings of the lEEE/CVE Conference on Computer Vision and Pattern Recognition,2023:7464-7475.
17. GE Zheng, LIU Songtao, WANG Feng, et al. Yolox: exceeding yolo series in 2021[EB/OL]. (2021-07-18). https://arxiv.org/abs/2107.08430.
18. 孙虎成,臧可,吕晓晓.基于改进YOLOv8m的道路目标检测算法[J/OL].电子设计工程, 1-9[2024-12-30].http://kns.cnki.net/kcms/detail/61.1477.TN.20241223.1724.002.html.
19. 王宁,智敏.深度学习下的单阶段通用目标检测算法研究综述[J/OL].计算机科学与探索,1-32[2025-04-30].http://kns.cnki.net/kcms/detail/11.5602.tp.20250117.2303.004.html.
20. WEI J F,NI L Y,LUO L,et al.GFS-YOLO11:a maturity detection model for multi-variety tomato[J].Agronomy,2024,14(11):1-21.
21. WANG C Y，LIAO H M，WU Y H，et al. CSPNet: a new backbone that can enhance learning capability of CNN [C]//2020 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). New York：IEEE，2020：1571-1580．
22. 周志飞,李华,冯毅雄,等.轻量化深度卷积神经网络设计研究进展[J].计算机工程与应用,2024,60(22):1-17.
23. HOWARD A G, ZHU M, CHEN B, et al. MobileNets:efficient convolutional neural networks for mobile vision applications[J]. arXiv:1704.04861, 2017.
24. 涂育智,王法翔,吴春霖.融合多注意力机制的轻量级无人机航拍小目标检测模型[J/OL].计算机工程与应用,1-15[2025-05-19].http://kns.cnki.net/kcms/detail/11.2127.TP.20250226.1447.019.html.
25. 闫建红,冉同霄.基于YOLOv8的轻量化无人机图像目标检测算法[J/OL].图学学报, 1-9 . http://kns. cnki.net/kcms/detail/10.1034.T.20241010.2055.004.html. YAN J H, RAN T X. Lightweight UAV image target detection algorithm based on YOLOv8 [J/OL]. JournalofGraphics,1-9.http://kns.cnki.net/kcms/detail/10.1034.T. 20241010.2055.004.html.
26. 范博淦,王淑青,陈开元.基于改进YOLOv8的航拍无人机小目标检测模型[J/OL].计算机应用, 1-11 .https://link.cnki.net/urlid/51.1307.tp.20241017.1040.004. FAN B J, WANG S Q, CHEN K Y. Small target detection model for aerial photography UAV based on improved YOLOv8 [J/OL]. Journal of Computer Applications, 1-11 . https://link.cnki.net/urlid/51.1307.tp. 2024 1017.1040.004.
27. 黄曼婷.基于卷积神经网络的中国交通标志检测研究[D].长沙理工大学,2018.
28. 宋晓茹,王嘉乐,刘通,等.改进YOLOv11的复杂场景下交通车辆检测方法[J/OL].激光杂志,1-9[2025-05-24].http://kns.cnki.net/kcms/detail/50.1085.TN.20250523.1108.012.html.
29. 唐彪,贾小军,骆淑云,等.BL-YOLO：无人机航拍图像目标检测算法[J/OL].光电子·激光,1-12[2025-05-15].http://kns.cnki.net/kcms/detail/12.1182.o4.20250423.1727.002.html.

致 谢

首先感谢我的指导老师王平，在我撰写论文的过程中一直给予我指导，帮助我把论文从简单的几个小章节扩充成两万字的一篇完整论文，并在休息日的晚上在微信上给我提出修改意见，没有她的指导，我的论文恐怕写上一个月也不会有眉目。

其次要感谢我的室友毕书尧，陈希文，张健阳，张永航，赵威鹏，我从他们那里学了很多文档编辑技巧和论文格式，还要感谢一下经常来宿舍串门的董旭旸同学，他告诉我怎么在外卖平台领券，让我省了好几块钱。

然后没了。