引用:孙嘉豪,邹瑞滨,李和福,等. 优化实时交通流检测:一种将 YOLO 与图像预处理结合的研究[J]. 聊城大学学报(自然科学版),2026,39(2):238-248

Citation: SUN Jiahao, ZOU Ruibin, LI Hefu, GAO Yang, ZHANG Weilin, LIU Yuanxi. Optimizing real-time traffic flow detection: an integrated approach combining YOLO with advanced image preprocessing[J]. Journal of Liaocheng University (Natural Science Edition),2026,39(2): 238-248.

文章编号 1672-6634(2026)02-0238-11 **DOI** 10.19728/j.issn1672-6634.2025050002

优化实时交通流检测:一种将 YOLO 与图像预处理结合的研究

孙嘉豪,邹瑞滨,李和福,高扬,张葳琳,刘沅鑫

(聊城大学 物理科学与信息工程学院,山东 聊城 252059)

摘要 为提高城市交通管理效率,应用智能交通系统是当前较为高效可行的办法,车流量实时监测技术可为整个系统提供有效的实时数据,研究聚焦于提升车流量实时监测终端设备性能的关键需求,提出一种基于 YOLO 目标检测算法的方法,结合图像预处理技术提升系统准确度并减少了终端资源消耗。研究基于 COCO 公开数据集整理和处理部分素材图,构建符合本次测试的自训练数据集,分别训练 YOLOv5s 与 YOLOv8s 模型,在动态视频和实时视频流等多种场景下进行全面测试,同时引入背景差分、CLAHE 图像增强和中值滤波等图像预处理技术,有效验证了模型在复杂环境下对目标识别能力的提升,同时降低了设备资源占用。实验结果说明,图像预处理技术在不同测试方式和环境下提高约 1.2%~1.8%的检测精度,相应的资源占用情况也有所降低,研究设计并分析了从模型训练、图像处理及性能评估等主要环节,凭借一系列视频测试和现实模拟测试得出相应数据,有一定应用价值。

关键词 智能交通;YOLO;车流量实时监测;模型训练;图像预处理

中图分类号 O495;TP391.41 **文献标志码** A **开放科学(资源服务)标识码(OSID)**

Optimizing real-time traffic flow detection: an integrated approach combining YOLO with advanced image preprocessing

SUN Jiahao, ZOU Ruibin, LI Hefu, GAO Yang, ZHANG Weilin, LIU Yuanxin

(School of Physics Science and Information Technology, Liaocheng University, Liaocheng 252059, China)

Abstract To improve the efficiency of urban traffic management, the application of intelligent transportation systems (ITS) has become a practical and effective approach. Real-time traffic flow monitoring technology provides essential data support for such systems. This study focuses on enhancing the performance of terminal devices used for real-time traffic monitoring and proposes a method based on the YOLO object detection algorithm. By incorporating image preprocessing techniques, the system achieves improved de-

收稿日期:2025-05-07

基金项目:山东省自然科学基金项目(ZR2021MF097)资助

通信作者:邹瑞滨,男,汉族,博士,副教授,研究方向:图像处理、嵌入式系统开发、红外成像,E-mail:zouruibin@lcu.com。

tection accuracy while reducing computational resource consumption at the terminal. The research utilizes a subset of the COCO public dataset to construct a customized training dataset suitable for this task. YOLOv5s and YOLOv8s models were trained and comprehensively evaluated across various scenarios, including dynamic video and real-time video streams. Techniques such as background subtraction, Contrast Limited Adaptive Histogram Equalization (CLAHE), and median filtering were applied to enhance input image quality. Experimental results demonstrate that these preprocessing methods improve detection accuracy by approximately 1.2% to 1.8% under different testing conditions and environmental complexities, while also reducing resource usage. This study systematically analyzes key components including model training, image processing, and performance evaluation. Through a series of video-based and real-world simulation experiments, the proposed approach is shown to have practical value for intelligent traffic applications.

Keywords　intelligent transportation; YOLO; real-time vehicle flow monitoring; model training; image preprocessing

0　引言

智慧城市是指利用各种信息技术或创新概念，将城市系统和服务打通、集成，提升资源利用效率并改善市民生活质量[1]。我国自 2012 年正式提出智慧城市等概念，其中智能交通是智慧城市的重要组成之一，可以实时收集、分析与上传城市各道路情况等信息，为交通管理、规划及安全评估提供数据支撑。传统车流量监测方法安装成本比较昂贵，维修成本也比较高，对于检测到目标准确度比较差[2]。本研究可以利用现有的摄像头设备，采用深度学习的目标检测算法实现低成本、高效率的数据采集与目标检测方法。

最近几年，YOLO(You Only Look Once)系列算法凭借其计算快速和较高准确率[3]，通过及时优化检测目标能够有效在检测速度与精度中取得平衡的特点[4]，在该领域得到了广泛应用[5]。当前学界研究主要侧重点在于引入其他目标跟踪等算法来对 YOLO 检测出的目标进行优化降低误检率[6]，同时通过应用图像预处理技术(如背景差分、CLAHE 自适应直方图均衡及中值滤波)抑制噪声并增强检测目标特征[7]。

本文中提出了一种基于 YOLO 与图像预处理的车流量实时监测系统优化方法，针对关键理论与技术、方法试验与设计、实验结果与分析三部分进行论述[8]。通过收集与处理相应素材图，构建自训练数据集，分别训练了 YOLOv5s 与 YOLOv8s 模型。基于 OpenCV 视觉库引入背景差分、CLAHE 图像增强和中值滤波等图像预处理技术，对比分析图像预处理前后模型在动态视频与实时视频流下的检测性能。

1　关键理论与技术

1.1　深度学习技术

深度学习是一种基于人工神经网络(Artificial Neural Network，ANN)的机器学习方法[9]，利用多层神经网络自动提取目标的数据特征，是当前目标检测领域研究的首选技术[10]。传统的机器学习在进行数据预处理之后依然采用人工特征提取和选择分类器，工作量较大、错误率也比较高。相比之下，深度学习可以通过设计模型直接从训练中自主学习特征。传统机器学习和深度学习流程如图 1 所示：

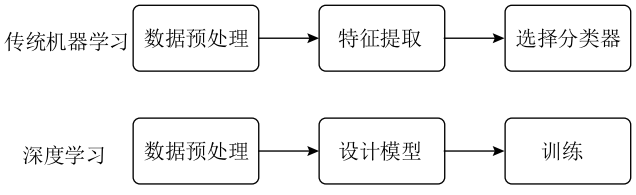


图 1　传统机器学习和深度学习流程

Fig. 1　Processes of traditional machine learning and deep learning

1.2 YOLO目标检测技术

YOLO是一组实时目标检测机器学习算法,简化了传统检测方法的流程,相比传统检测方法,YOLO不需要人工进行特征设计和分类器训练,检测速度快,计算量较低。YOLO将输入图像切割成多个网格,每个网格只预测一定数量的边界框,仅需要进行一次卷积神经网络(Convolutional Neural Network, CNN)进行特征提取[11],计算出目标物体的类别概率和坐标信息,将不影响检测结果的冗余边界框去除,减少计算量,根据预先设定的置信度阈值输出检测到的目标类别和位置信息[12]。

1.3 预处理技术

OpenCV是一个基于BSD(Berkeley Software Distribution)许可开发的跨平台目标检测库,可以运行在Windows、Linux等多种操作系统上[13],主要由C函数和部分C++函数构成,提供了例如Python、MATLAB等诸多语言的接口。OpenCV是广泛应用于目标检测的开源库,拥有丰富的图像处理函数库,非常适合用于目标检测的开发利用[14]。

对于本研究中,OpenCV额外提供了CUDA(Compute Unified Device Architecture)的使用接口,CUDA是英伟达公司设计的一种通用并行计算平台和编程模型,在OpenCV中使用CUDA可以显著提高图像处理的性能,并可以调用部分机器学习算法。在本次研究的图像预处理阶段,通过OpenCV的资源库中引入背景差分、图像增强和滤波处理等技术。

背景差分是作用于视频中的运动目标,将视频中当前帧与背景参考模型做减法,通过计算与背景图像差异超过一定阈值的区域作为运动区域,区分背景和运动区域这种方法可以有效减少计算量,排除背景因素对于推理结果的影响,适用于非移动摄像机的场景。在OpenCV中使用高斯混合模型(GMM)的改进算法,相较于改进前算法,高斯混合模型改进算法(MOG2)可支持一至五个动态数量,可减少冗余计算,每个像素使用多个高斯分布建模。高斯混合模型改进算法的计算方式如式(1)所示,其中 $p(x)$ 代表"背景模型总概率",取值0至1,概率越高,代表像素越可能是背景,用于判断是动态目标还是静态背景;x 为当前帧中某像素的灰度值;$\sum_{K=1}^{K}\pi_k$ 为权重归一化约束,所有权重之和为1;$N((x)\,|\mu_k,\sum_k)$ 为均值为 μ_k 、协方差为 $\sum_k$ 的高斯分布概率密度函数,用于描述当前像素 x 与某背景模式 μ_k 的相似度,相似度越高,N 越大。

$$p(x)=\sum_{k=1}^{K}\pi_k N((x)\,|\mu_k,\sum_k)。\tag{1}$$

图像增强利用CLAHE和伽马校正,通过将图像分割成小的像素块,各部分独立将直方图均衡化,对不同区域应用不同增强,改变像素值与最终输出之间非线性关系来调整图像亮度和对比度,对视频中某些低质量帧进行处理,减少曝光问题导致高对比度引起结果失真。CLAHE计算方法如式(2)所示,其中 s_k 代表原始灰度级 k 经过映射之后的新灰度级;T(k) 为灰度级映射函数;round()代表四舍五入为整数的取整函数;$M\times N$ 为分块的总像素数量,本次行与列均设置为8,总像素数量为64;$F(k)$表示灰度级 k 的累积分布函数(Cumulative Distribution Function,即CDF),$F_{\min}$ 为 $F(k)$排除纯黑像素后的最小值;$L-1$ 为最大灰度值,此处为255对应纯白像素。伽马校正计算如式(3)所示,其中 $g(x,y)$为伽马校正后坐标处的像素灰度值,取值0~255;$f(x,y)$伽马校正前坐标处的像素灰度值,取值0~255;γ 为伽马系数,本次实验处理夜晚图像取值小于1,用于提亮,处理白天图像取值大于1,用于压暗。

$$s_k=T(k)=\mathrm{round}\left(\frac{F(k)-F_{\min}}{(M\times N)-F_{\min}}\times(L-1)\right),\tag{2}$$

$$g(x,y)=255\times\left(\frac{f(x,y)}{255}\right)^{\gamma},\tag{3}$$

应用中值滤波和运动模糊补偿机制进行滤波处理,中值滤波是一种非线性平滑技术,该技术以每一个像素点相邻区域窗口内所有像素点灰度值的中值作为该点的灰度值,能大幅降低噪声对推理时的影响,数学公式如式(4)所示,其中 $g(x,y)$为中值滤波后该坐标处的像素灰度值;median()函数代表在排序后取中间位置数值;$f(x+i,y+i)$ 为滤波窗口内改坐标处的原始邻域像素灰度值;i,j 为领域像素相对于中心像素的偏

移量，窗口大小由 k 决定。运动模糊补偿机制可处理因高速运动物体产生的模糊和扭曲等现象，能有效恢复视频帧清晰度。

$$g(x,y)=\text{median}(\{f(x+i,y+i)\mid i,j\in[-k,k]\}),\tag{4}$$

优化实时交通流检测采用深度学习的方法，模型采用 YOLO 模型，预处理采用 OpenCV 工具平台实现。

2　方法与实验设计

2.1　硬件与软件平台介绍

2.1.1　*PC 端硬件配置*。本研究采用 PC 端进行车流量实时监测实验，模型训练、视频测试为同一平台，计算机处理器型号为英特尔 I9-13700F，内存为 DDR4，大小为 32 GB，利用英伟达 RTX3080 显卡，显卡内存大小为 8 GB 并搭载 NVIDIA CUDA 环境加速 YOLO 算法训练和目标识别推理。

2.1.2　*PC 端软件环境*。本研究所用操作系统为 Windows11，涉及到主要软件功能及对应版本型号如表 1。

表 1　实验主要软件功能及应用版本型号

Tab. 1　Main software functions and application versions and models in the experiment

功能	所用软件名称	版本型号
模型训练	Visual Studio Code	13.4
编程语言	Python	3.8
深度学习框架	PyTorch	1.12.1
模型转换	ONNX	1.12
图片标注	Labelimg	
推理加速	NVIDIA CUDA	11.3

2.2　YOLO 系列模型简介

YOLO 系列模型是一种高效的目标检测算法[15]，本研究选择了以下两个版本的 YOLO 模型进行分析。

2.2.1　*YOLOv5*。YOLOv5 是 Ultralytics 团队于 2020 年推出的一款相较于之前版本提升更大的一个版本。本研究选择了 YOLOv5 系列中深度最小，特征图宽度最小的 YOLOv5s 网络。系列模型特征如表 2 所示，因其具有较小的模型体积和较快的速度[16]，所以更适合部署在资源有限的设备中。

表 2　YOLOv5 系列模型特征

Tab. 2　Characteristics of YOLOv5 series models

模型	FP16 模型大小/MB	V100 响应时间/ms	COCO 精度/mAP
YOLOv5n(Nano)	4	6.3	28.4
YOLOv5s(Small)	14	6.4	37.2
YOLOv5m(Medium)	41	8.2	45.2
YOLOv5l(Large)	89	10.1	48.8
YOLOv5x(XLarge)	166	12.1	50.7

YOLOv5 的模型结构主要由五部分构成，分别是输入端、Backbone 网络、Neck 网络、输出端和激活函数。YOLOv5 可利用 Mosaic 图像提高增加数据多样性，自适应锚框计算能提高检测精度，CSP 和 SPP 结构可优化特征提取，利用 NMS 减少重复检测。该模型能借助自适应图片缩放适应不同尺寸目标，并结合 FocalLoss 和 Mish 激活函数提升检测性能和鲁棒性。

2.2.2　*YOLOv8*。Ultralytics 公司在 2023 年发布了最新一代实时目标检测模型，YOLOv8 采用了更先进的骨干网络和颈部架构，优化了自适应锚框机制，可自动调整锚框尺度和比例，以适应不同场景下目标的形状和大小，相对于之前版本提高了对动态场景中目标运动模糊和遮挡的鲁棒性[17]。新加入的“变形卷积”技术能在输入图像的不同区域按需求调整卷积核大小和形状[18]，可捕捉图像细节特征，新增加多尺度特征融合机制，使网络可同时考虑细节问题和全局信息，有效提升了对不同尺寸物体的检测能力，利用注意力机制，可在预测时聚焦于图像的关键部分，提高对重要特征的关注度[19]。为了此次研究控制额外的变量影响检测

结果,YOLOv8 系列模型同样选择了较小的 YOLOv8s。

2.3 数据集选择与模型训练

2.3.1 *数据集选择*。本研究使用基于 COCO 公开数据集中部分图片以及部分实拍场景构建成的自训练数据集模型,实拍场景来源为各城市道路监控图片以及街拍各种包含目标物体图片,数据集图片数量为 1 200 张,选取图像的尺寸有 1 920×1 080、1 280×650、1 080×960、640×640 等 4 种。其中针对光照不同的白天图片约 850 张,其余为光线较弱的夜晚图片;不同天气主要为晴天、阴天、雨天和下雪天气;大约 700 张图片角度为街道上方比较贴合实际街道摄像头俯视角度,其余角度为平视角度。利用 Labelimg 进行人工标注标签,标注结果以 YOLO 格式储存,包括目标类别与归一化后的边界框坐标,目标标签设置为家用客车(car)、卡车(truck)、公共汽车(bus)、摩托车(motorcycle)、自行车(bicycle)等 5 种。

2.3.2 *模型训练流程与参数配置*。本研究训练模型在 PyTorch 框架下进行,采用迁移学习方式微调 COCO 预训练权重,预训练权重使用 ultralytics 官方提供(YOLOv5s.pt、YOLOv8s.pt)。训练使用参数如表 3。

表 3 模型训练参数

Tab. 3 Model training parameters

训练参数	训练数值
框架	PyTorch
输入图像尺寸/像素	640×640
Batch size	16
训练轮次/Epochs	500

2.4 图像预处理模块和测试流程

OpenCV 是一个跨平台的目标检测库,利用其中提供的 Python 接口,可以通过设计代码应用图像处理和目标检测方面的很多通用算法。通过 COCO 数据集的网格搜索确定,基于验证集上精度与效率权衡实验,设置 CLAHE 的 clip Limit(对比度限制阈值)为 2.0,tile size(分块大小)采用 8×8 网格;中值滤波采用的窗口尺寸为 3×3;背景差分设置 200 帧历史长度与方差阈值 16。

本次实验在 Windows 环境下的 Visual Studio Code 软件中为主要控制端,通过设计 Python 代码调用 OpenCV 库中的 cv2 接口连接 USB 摄像头进行视频输入[20],并利用 CUDA 接口对输入视频进行加速推理。利用代码启用 OpenCV 库中背景差分、图像增强和滤波处理等图像预处理技术,将运算处理器设置为 GPU 加速推理。利用代码同时获得处理视频流时计算机中内存、GPU 等资源消耗情况并记录。项目中图像处理的流程如图 2 所示。

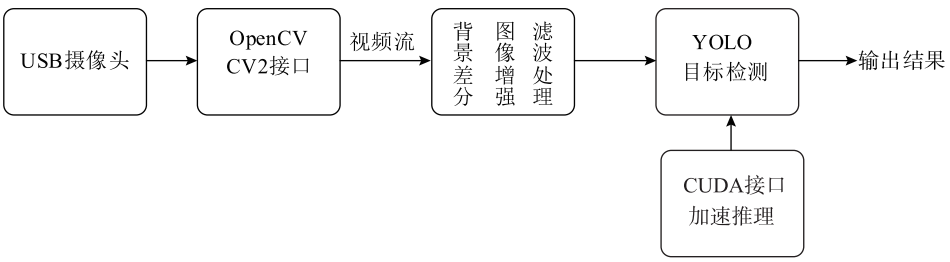


图 2 图像处理的流程

Fig. 2 Processes of image processing

2.5 评估指标

在机器学习领域常用混淆矩阵评估分类模型的性能,混淆矩阵为评估指标中的一级指标,其中主要参数有 TP、FP、TN 和 FN,在之后中分别用 T_p、F_p、T_n、F_n 表示。T_p 为模型正确地将正例预测为正例的数量,F_p 为模型错误地将负例预测为正例的数量,T_n 为模型正确地将负例预测为负例的数量,F_n 为模型错误地将正例预测为负例的数量。T_p 和 T_n 的值越大,对分类的效果越好,F_n 和 F_p 越小,对分类的效果越好。一级指标通常统计的为数量,但是在比较大的模型中,数量来评价性能不全面,通常以一级指标为基础利用二级指标来评估性能。

2.5.1 *精确率和召回率*。精确率(Precision)与召回率(Recall)分别反映了模型在预测和检出能力方面的表现,精确率(P)和召回率(R)计算方式如式(5)和式(6)所示,是评估目标检测性能的核心指标,

$$P=\frac{T_p}{T_p+F_p}, \tag{5}$$

$$R=\frac{T_{\mathrm{p}}}{T_{\mathrm{p}}+F_{\mathrm{n}}},\tag{6}$$

使用 PR 曲线和混淆矩阵对精确率与召回率进行可视化分析。对比 YOLOv5s、YOLOv8s、在不同置信度阈值下的精确率与召回率曲线，计算出平衡相对高精确率和高召回率合适点。为了更为准确评价模型性能，在精确率和召回率基础上引申出三级指标 $F_{1\,\mathrm{Score}}$ 用于平衡“不误检”和“不漏检”，在二级指标基础上进行算术运算，$F_{1\,\mathrm{Score}}$ 计算方式如式(7)所示，在 0 到 1 之内取值，越接近于 1 模型的输出最好。

$$F_{1\mathrm{Score}}=\frac{2PR}{P+R}。\tag{7}$$

2.5.2　准确率(精度)。AP(Average Precision)是用于衡量模型在某个类别上的识别效果，取值在 0 到 1 之间，值越大模型对此类别的检测性能就更强。

准确率是目标检测模型在测试数据集上的整体检测能力的综合评价指标，通常以平均精度(m_{AP}，Mean Average Precision)表示，精度(Precision)是指模型预测为正样本中实际为正样本的比例，通过计算所有类别的平均精度得出。

在模型推理结束后，比较检测结果与真实标注框之间的匹配情况。匹配标准为 IoU(Intersecti on over Union，交并比)大于设定阈值(0.5)。根据如式(8)所示公式计算每个类别的精度(AP，Average Precision)，并取所有类别的平均值

$$m_{\mathrm{AP}}=\frac{T_{\mathrm{p}}+T_{\mathrm{n}}}{T_{\mathrm{p}}+T_{\mathrm{n}}+F_{\mathrm{p}}+F_{\mathrm{n}}}。\tag{8}$$

m_{AP} 值越高，说明模型在整体检测任务中的准确性越高，尤其是在多类别目标检测中表现优异。

2.5.3　响应时间。推理时间是衡量目标检测模型实时性能的关键指标，是指从输入图像(或视频帧)到输出检测结果(包括检测框和分类标签)的时间延迟。

在 PC 端搭载训练完成后的 YOLOv5s、YOLOv8s 模型，分别凭借动态视频和实时视频流来进行测试，在 Visual Studio Code 中新建环境利用 Python 代码记录模型推理的起始时间与结束时间，并计算平均响应时间，计算方式如式(9)所示，其中 T_{avg} 表示平均响应时间，N 为帧数，$t_{\mathrm{end},i}$、$t_{\mathrm{start},i}$ 分别为第 i 帧的推理起始时间和结束时间。为保证测试的公平性，每个模型均运行相同的输入数据集。

$$T_{\mathrm{avg}}=\frac{1}{N}\sum_{i=1}^{N}(t_{\mathrm{end},i}-t_{\mathrm{start},i})。\tag{9}$$

2.5.4　资源消耗。在实时监测系统实际部署中，系统运行时需要调用大量 PC 端资源，主要在内存消耗和 GPU 占用。所有程序运行都在内存中进行，内存消耗量的大小直接决定了设备处理信息的速度，越小占用意味着整个系统的运行会更加流畅，同时也可以减少电量的消耗。GPU 为系统中主要推理运行的部分，AI 推理计算能力大小直接由 GPU 的占用空间限制。

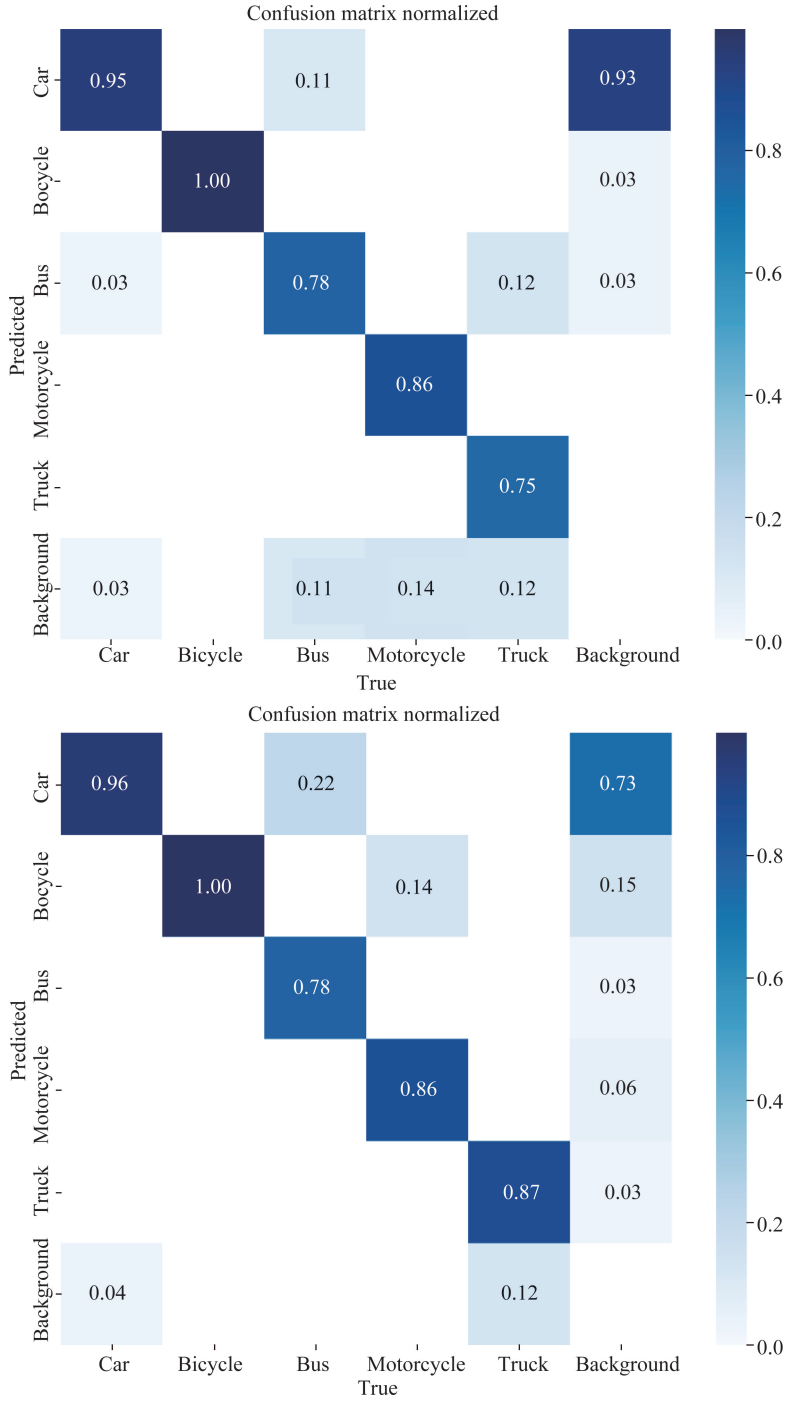
3　实验结果与分析

3.1　实验结果

本章基于 YOLOv5s 和 YOLOv8s 自训练模型，对未引入图像预处理技术和引入图像预处理技术在车流量实时监测任务中的表现进行实验评估，采用混淆矩阵、精确率、召回率以及平均精度(m_{AP})等指标对模型进行评价。为了进一步综合评估实验结果，引入 $F_{1\,\mathrm{Score}}$ 作为平衡指标，并基于 I_{oU} 阈值计算各类别的 AP，进而得到整体 mAP 值。实验采用动态视频测试和实时视频流测试两种方式，以确保评估的全面性。

3.1.1　YOLOv5s 和 YOLOv8s 模型训练结果及打印信息。经过 500 轮训练之后所得到的 YOLOv5s 的 best.pt 模型的归一化混淆矩阵如图 3(a)所示，YOLOv8s 的 best.pt 模型的归一化混淆矩阵如图 3(b)所示，横坐标为真实类别，纵坐标为预测类别，矩阵中元素颜色越深代表此类目标数值越大。YOLOv5s 对于 bicycle 标签目标检测数值达到 1.00；对于 car 标签目标检测准度达到 0.95，有 0.03 误检为 bus；对于 motorcycle 标签目标检测准度达到 0.86；对于 bus 标签目标检测准度达到 0.78，有 0.11 误检为 car；对于 truck 标签目标检测准度最低，为 0.75，有 0.12 误检为 bus；car、bus、motorcycle 和 truck 等四类目标漏检为背景概率分别为 0.03、0.11、0.14 和 0.12。

YOLOv8s 对于 bicycle 标签目标检测数值达到 1.00;对于 car 标签目标检测准度达到 0.96;对于 motorcycle 标签目标检测准度达到 0.86,有 0.14 误检为 bicycle;对于 truck 标签目标检测准度达到 0.87;对于 bus 标签目标检测准度最低,为 0.78,有 0.22 误检为 car;仅有 0.12 概率将 truck 漏检为背景。



注:(a) YOLOv5s;(b) YOLOv8s。

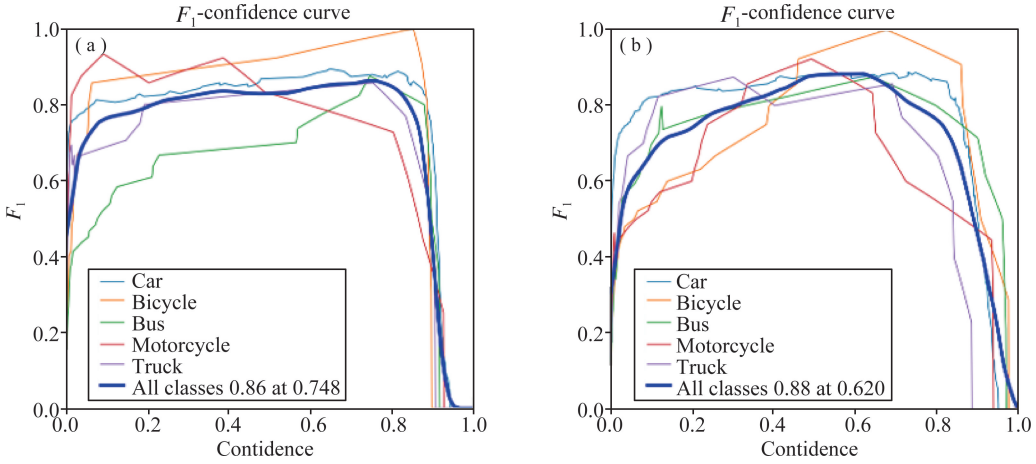
图 3 训练之后归一化混淆矩阵

Fig. 3 Normalized confusion matrix after the training

YOLOv5s 的 F1 Score 曲线如图 4(a)所示,YOLOv8s 的 F1 Score 曲线如图 4(b)所示,横坐标为置信度,纵坐标为 F1 分数。图中深蓝色的曲线为所有类别,浅蓝色曲线为 car 标签,黄色曲线为 bicycle 标签,绿色曲线为 bus 标签,红色曲线为 motorcycle 标签,紫色曲线为 truck 标签。

YOLOv5s 的 PR 曲线如图 5(a)所示,YOLOv8s 的 PR 曲线如图 5(b)所示。YOLOv5s 中 car 标签值为 0.929,bicycle 标签值为 0.995,bus 标签值为 0.785,motorcycle 标签值为 0.978,truck 标签值为 0.879,

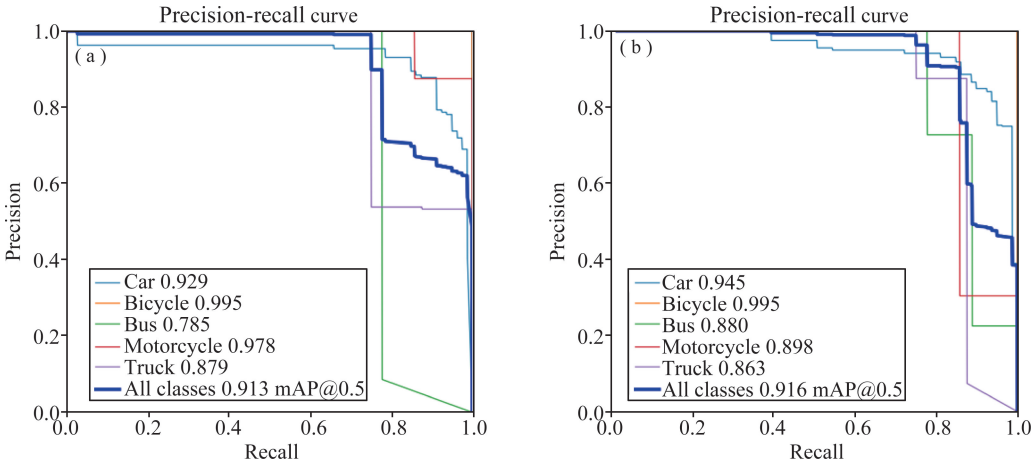
深蓝色曲线所围成的面积即 m_{AP} 的值在 0.5 置信度下为 0.913。YOLOv8s 中 car 标签值为 0.945，bicycle 标签值为 0.995，bus 标签值为 0.880，motorcycle 标签值为 0.898，truck 标签值为 0.863，深蓝色曲线所围成的面积即 m_{AP} 的值在 0.5 置信度下为 0.916。



注：(a) YOLOv5s；(b) YOLOv8s。

图 4　F_{1Score} 曲线

Fig. 4　F_{1Score} curve



注：(a) YOLOv5s；(b) YOLOv8s。

图 5　PR 曲线

Fig. 5　Precision-recall (PR) curve

注：(a) YOLOv5s；(b) YOLOv8s。

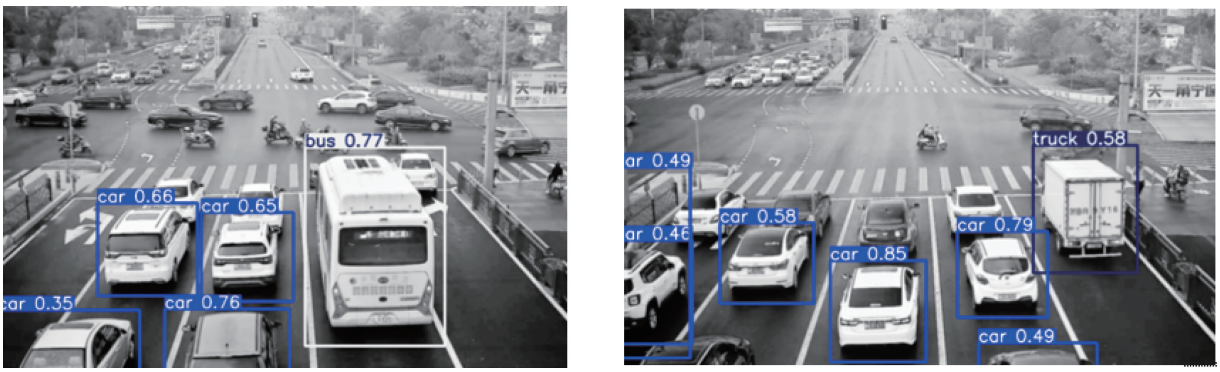
图 6　视频测试效果

Fig. 6　Video testing performance

3.1.2　动态视频测试结果。本实验动态视频选取公开城市交通监控视频，城市道路摄影等真实影像，

涵盖不同天气和光照条件,其中两部视频分辨率为 2 K,其余三部为 4 K。分别依次利用不同模型进行推理,利用 Python 代码抓取推理过程中各测试指标数值,利用公式计算每段视频检测精度[21]。将 I_{oU} 值统一设置为 0.5,未引入图像预处理技术,YOLOv5s 视频测试效果如图 6(a)所示,YOLOv8s 视频测试效果如图 6(b)所示。

保持其他设置不变,开启图像预处理技术进行视频测试,图像预处理后 YOLOv5s 测试效果如图 7(a)所示,图像预处理后 YOLOv8s 测试效果如图 7(b)所示。



注:(a) YOLOv5s;(b) YOLOv8s。

图 7 图像预处理后测试效果

Fig. 7 Testing performance after image preprocessing

动态视频测试检测获取到的数值如表 4 所示。

表 4 动态视频测试检测获取到的数值

Tab. 4 Acquired values from dynamic video testing detection

项目	未引入图像预处理模块		引入图像预处理模块	
	YOLOv5s	YOLOv8s	YOLOv5s	YOLOv8s
样本一测试精度/%	90.68	91.68	91.57	93.02
样本二测试精度/%	89.67	90.53	91.11	92.57
样本三测试精度/%	91.32	92.25	92.36	93.63
样本四测试精度/%	88.66	90.27	91.24	92.90
样本五测试精度/%	90.16	90.81	91.85	92.64
平均测试精度/%	90.01	91.11	91.63	92.95

3.1.3 实时视频流测试结果。本实验实时视频流利用可移动 PC 外接 USB 摄像头作输入端搭载不同训练模型进行推理,选取地点为城市道路天桥位置。为契合实际应用场所,本次测试均使用固定视角抓取 5 min 视频流,数据获取方式同动态视频一致,修改代码抓取运行过程中内存占用和 GPU 推理占用资源情况等,未引入图像预处理技术,YOLOv5s 实时视频流测试效果如图 8(a)所示,YOLOv8s 实时视频流测试效果如图 8(b)所示。

注:(a) YOLOv5s;(b) YOLOv8s。

图 8 实时视频流测试效果

Fig. 8 Real-Time video stream testing performance

保持其他设置不变，将图像预处理模块开启继续按照不同模型分别进行五次测试，图像预处理后 YOLOv5s 实时视频流测试效果如图 9(a)所示，图像预处理后 YOLOv8s 实时视频流测试效果如图 9(b)所示。

注：(a) YOLOv5s；(b) YOLOv8s。

图 9　图像预处理后实时视频流测试效果

Fig. 9　Testing performance in Real-Time video stream after image preprocessing

实时视频流测试检测获取到的数值如表 5 所示。

表 5　实时视频流测试检测获取到的数值

Tab. 5　Acquired values from real-time video stream testing and detection

项目	未引入图像预处理模块		引入图像预处理模块	
	YOLOv5s	YOLOv8s	YOLOv5s	YOLOv8s
样本一测试精度/%	90.68	91.68	91.57	93.02
样本一测试精度/%	89.82	90.45	90.55	91.52
样本二测试精度/%	88.16	90.87	90.91	92.16
样本三测试精度/%	90.05	89.95	90.24	91.86
样本四测试精度/%	89.26	90.15	91.37	90.99
样本五测试精度/%	90.52	90.23	90.98	91.95
平均测试精度/%	89.56	90.33	90.81	91.70
平均占用内存空间(GB)	1.14	1.27	0.98	1.14
平均响应时间(ms/帧)	14.5	15.7	19.6	19.9
平均 GPU 占用(GB)	1.60	1.83	1.55	1.79

3.2　结果分析

经训练之后模型在视频测试中，未引入图像预处理模块时用 YOLOv5s 和 YOLOv8s 推理相同视频识别平均精度分别为 90.01%和 91.11%；引入图像预处理之后 YOLOv5s 和 YOLOv8s 推理识别平均精度为 91.63%和 92.95%，相较于未引入图像预处理时平均精度分别提升约 1.62%和 1.84%。

在实时视频流测试中，未引入图像预处理模块时用 YOLOv5s 和 YOLOv8s 推理相同视频识别平均精度分别为 89.56%和 90.33%，平均占用内存空间为 1.14 GB 和 1.27 GB，平均响应时间为 14.5 ms 每帧和 15.7 ms 每帧，平均 GPU 资源占用为 1.60 GB 和 1.83 GB；引入图像预处理之后 YOLOv5s 和 YOLOv8s 推理识别平均精度为 91.63%和 92.95%，平均占用内存空间为 0.98 GB 和 1.14 GB，平均响应时间为 19.6 ms 每帧和 19.9 ms 每帧，平均 GPU 资源占用为 1.55 GB 和 1.79 GB，相较于未引入图像预处理时平均精度分别提升约 1.25%和 1.37%，平均内存占用空间分别减少约 0.16 GB 和 0.13 GB，平均响应时间分别增加 5.1 ms 每帧和 4.2 ms 每帧，平均 GPU 占用分别减少 0.05 GB 和 0.04 GB。

根据已公开专用车辆检测和跟踪的大规模数据集 UA-DETRAC 对比，输入图像尺寸/像素为 640×480。训练后的 YOLOv5 经重写 dataloader，即随机挑选训练视频文件夹随机取图片测试，识别帧数为 50 帧，9 轮测试中，最高 m_{AP}(测试精度)为 0.938，平均 m_{AP} 为 0.864，本次测试引入图像预处理之后 YOLOv5s 的平均识别精度相较之约提升 3 个百分点。

4　结论

本研究着重分析图像预处理技术对基于 YOLO 模型的车流量实时监测系统所产生的优化成效，借助分

别验证 YOLOv5s 和 YOLOv8s 在引入图像预处理技术前后的性能提升状况,全面且系统地剖析了图像预处理技术对检测精度、资源消耗以及反应实时性这几个方面的影响。

在动态视频以及实时视频流测试当中,引入图像预处理之后,借助背景差分有效地分离了运动目标与背景,结合图像提高对光照不均问题给予优化,并且利用中值滤波抑制噪声干扰,使得 YOLOv5s 和 YOLOv8s 的平均检测精度分别提高了 1.62%和 1.84%。整体上减少了计算量,基于 YOLO 模型的车流量实时监测系统引入图像预处理技术后,与 UA-DETRAC(车辆检测权威数据集)的 YOLOv5 模型精度相比提升较为较大,但基于训练模型数据集的差异和检测方法的不同,其余性能指标暂时无法比较;同时引入图像预处理技术后对于计算机的 GPU 和内存等资源消耗有一定的减少。

本次引入的图像预处理技术说明基于 YOLO 目标检测技术可借助图像预处理技术来提高准确率以及降低资源消耗,对于车流量监测系统有着一定的效果提升。因为 OpenCV 对于图像处理还有诸多预处理技术,以及本次研究两种算法对于 bus 和 truck 误检率都比较高,今后的研究将持续探索不同的图像预处理技术以及不同版本 YOLO 算法对于目标检测的提升效果,同时尝试将高端 PC 完成的任务尝试移植到更加适合实际交通场景中的边缘计算设备,将继续探讨模型轻量化或硬件加速方案的改进。

参 考 文 献

[1] 吴玉菲. 数字化背景下智慧城市建设和治理的问题思考[J]. 信息系统工程, 2025(2): 105-108.

[2] 曹华军. 车流量检测几种常用算法分析[J]. 信息技术与信息化, 2018(11): 83-86.

[3] WU W T, LIU H, LI L L, et al. Application of local fully convolutional neural network. combined with YOLO v5 algorithm in small target detection of remote sensing image[J]. PLoS One, 2021, 16(10): e0259283.

[4] ZHAO R, WANG X J, XIA J J, et al. Deep reinforcement learning based mobile edge computing for intelligent Internet of Things[J]. Physical Communication, 2020, 101184: 1874-4907.

[5] SHI W, CAO J, ZHANG Q, et al. Edge computing: vision and challenges[J]. IEEE Internet of Things Journal, 2016, 3(5): 637-646.

[6] 张韡,李永,刘涛,等. 基于深度学习的转弯车流量检测方法[J]. 科学技术与工程, 2025, 25(4): 1701-1710.

[7] 熊显名, 刘雨鑫, 黎恒. 基于改进 YOLOv5s 的监控视频车流量检测[J]. 桂林电子科技大学学报, 2024, 44(4): 416-426.

[8] LIM J, LEE J, AN C, et al. Enhancing real-time traffic volume prediction: a two-step approach of object detection and time series modelling[J]. IET Intelligent Transport Systems, 2024, 18(12): 2744-2758.

[9] MITTAL S. A survey on optimized implementation of deep learning models on the NVIDIA Jetson platform[J]. Journal of Systems Architecture, 2019, 97: 428-442.

[10] LI H, OTA K, DONG M X. Learning IoT in edge: deep learning for the Internet of Things with edge computing[J]. IEEE Network, 2018, 32(1): 96-101.

[11] CHEN Y H, T. KRISHNA T, EMER J S, et al. Eyeriss: an energy-efficient reconfigurable accelerator for deep convolutional neural networks[J]. IEEE Journal of Solid-State Circuits, 2017, 52(1): 127-138.

[12] HUSSAIN M. YOLO-v1 to YOLO-v8, the rise of YOLO and its complementary nature toward digital manufacturing and industrial defect detection[J]. Machines, 2023, 11(7): 677.

[13] CHEN W J, HUANG H B, PENG S, et al. YOLO-face: a real-time face detector[J]. The Visual Computer, 2021, 37(4): 805-813.

[14] 罗一中. 基于深度学习的 OpenCV 图像处理软件设计[J]. 软件, 2025, 46(1): 50-52.

[15] JIANG P Y, ERGU D, LIU F Y, et al. A review of yolo algorithm developments[J]. Procedia Computer Science, 2022, 199: 1066-1073.

[16] DIWAN T, ANIRUDH G, TEMBHURNE J V. Object detection using YOLO: challenges, architectural successors, datasets and applications[J]. Multimedia Tools and Applications, 2023, 82(6): 9243-9275.

[17] 姚若禹, 郑世玲, 史怡璇, 等. 基于改进 YOLOv8n 的钢材表面缺陷检测算法[J]. 聊城大学学报(自然科学版), 2025, 38(2): 177-189.

[18] Lee Y H, Kim Y. Comparison of CNN and YOLO for object detection[J]. Journal of the semiconductor & display technology, 2020, 19(1): 85-92.

[19] TERVEN J, CÓRDOVA-ESPARZA D M, ROMERO-GONZÁLEZ J A. A comprehensive review of YOLO architectures in computer vision: from YOLOv1 to YOLOv8 and YOLO-NAS[J]. Machine Learning and Knowledge Extraction, 2023, 5(4), 1680-1716.

[20] 刘忠瑞, 赵廷玉. 基于 OpenCV 的 4K HDMI 相机上的自动寻边体系的研究[J]. 智能计算机与应用, 2024, 14(11): 156-162.

[21] 高国人, 张博怀, 时啟璐, 等. 图像处理系统的可视化界面设计与应用[J]. 自动化应用, 2025, 66(7): 233-236,240.

(责任编辑:赵海军)