引用:薛培友,刘磊,刘瑞阳,等.基于树莓派和 YOLO 算法的车型检测[J].聊城大学学报(自然科学版),2025,38(6):840-852.

Citation: XUE Peiyou, LIU Lei, LIU Ruiyang, BIAN Wenyang, YAN Dongmei. Vehicle detection based on raspberry pi and YOLO algorithm[J]. Journal of Liaocheng University (Natural Science Edition), 2025,38(6):840-852.

文章编号 1672-6634(2025)06-0840-13 **DOI** 10.19728/j.issn1672-6634.2024090014

基于树莓派和 YOLO 算法的车型检测

薛培友[1],刘磊[1,2],刘瑞阳[2],卞文阳[2],闫冬梅[3]

(1.安徽省普通高校交通信息与安全重点实验室,安徽 合肥 230601;2.河海大学 数学学院,江苏 南京 211100;
3.南京邮电大学 现代邮政学院,江苏 南京 210003)

摘要 随着我国民用车辆保有量呈现出显著的增长态势,交通管理工作面临着前所未有的挑战。在此背景下,智能交通系统的研发显得尤为迫切且刻不容缓。车型检测作为智能交通系统的核心关键技术,在交通管理领域中发挥着举足轻重的作用。然而,现有的车型检测系统存在着资源消耗大、成本居高不下等明显弊端,这在很大程度上限制了系统的广泛推广与应用。针对这一现状,本文构建了 10 类常见汽车的 TenCar 数据集,并提出了一种基于树莓派和 YOLOv4-tiny 算法的低成本车型检测系统。经过实践验证,YOLOv4-tiny 模型在 TenCar 数据集上实现了 85.65%的精确率和 99.9%的召回率。该系统巧妙地借助图像识别技术以及精简的硬件设施,能够对车辆进出入场所的车型进行高效、自动的检测。这一特性在很大程度上简化了车辆的查找过程,使得管理工作更加便捷高效,并且该系统具有良好的普及性,易于在公司、小区等场所推广应用。

关键词 树莓派;车型检测; YOLOv4-tiny; YOLOv5s

中图分类号 U495;TP391.41 **文献标志码** A **开放科学(资源服务)标识码(OSID)**

Vehicle detection based on raspberry pi and YOLO algorithm

XUE Peiyou[1], LIU Lei[1,2], LIU Ruiyang[2], BIAN Wenyang[2], YAN Dongmei[3]

(1. Key Laboratory of Traffic Information and Safety of Anhui Higher Education Institutes, Hefei 230601, China;
2. College of Science, Hohai University, Nanjing 211100, China;3. School of Modern Posts,
Nanjing University of Posts and Telecommunications, Nanjing 210003, China)

Abstract As the number of civilian vehicles in China shows a remarkable growth trend, traffic management is facing unprecedented challenges. Against this backdrop, the development of intelligent transportation systems (ITS) is extremely urgent. As a core and crucial technology of ITS, vehicle type detection plays a pivotal role in the field of traffic management. However, the existing vehicle type detection systems have obvious drawbacks such as high resource consumption and exorbitant costs, which greatly limit their wide-spread promotion and application. In response to this situation, this paper carefully constructs the TenCar dataset containing 10 common types of vehicles and proposes a low-cost vehicle type detection system based on the Raspberry Pi and the YOLOv4-tiny algorithm. Through practical verification, the

收稿日期:2024-09-30

基金项目:安徽省重点实验室基金项目(KLAHEI18018);教育部重点实验室开放基金(Scip20240111)资助

通信作者:刘磊,男,汉族,博士,教授,博士生导师,研究方向:强化学习、多智能体系统,E-mail:liulei_hust@163.com。

YOLOv4-tiny model has achieved an accuracy rate of 85.65% and a recall rate of 99.9% on the TenCar dataset. This system skillfully utilizes image recognition technology and streamlined hardware facilities to efficiently and automatically detect the vehicle types when vehicles enter or exit a venue. This feature greatly simplifies the process of vehicle search, making management work more convenient and efficient. Moreover, the system has good popularization potential and is easy to be promoted and applied in places such as companies and residential communities.

Keywords　raspberry pi; vehicle detection; YOLOv4-tiny; YOLOv5s

0　引言

截至 2023 年初，我国民用车辆数量相较于去年呈现出显著增长，环比增量超一千万辆。汽车保有量的迅速攀升，在为民众出行提供便利的同时，也给社会交通管理带来了诸如车辆拥堵加剧、道路事故频发等一系列难题。面对这一新的交通态势，传统单纯依靠人工的处理模式愈发显得力不从心。故而，研发智能交通系统(Intelligent Traffic System，ITS)来取代传统人工管理方式已迫在眉睫。在智能交通系统中，车型检测技术扮演着极为关键的角色，是实现其各项功能的重要支撑。

车型检测属于目标检测的一个特定应用领域，目标检测技术为车型检测提供了基础和通用的方法。随着深度学习技术的不断普及，目标检测也由此进入基于深度学习算法的时期。基于深度学习算法的目标检测根据其进行过程主要分化为两种模式，分别为一阶段模式和二阶段模式。一阶段模式，被称为基于回归的目标检测，其算法检测过程是一步完成的，即其对目标的位置坐标定位、类别的区分以及类别概率的计算直接回归，一次性完成定位和分类过程。自 2013 年至今，基于回归的目标检测算法主要有 YOLO 算法系列[1-3]、SSD 算法系列[4]、基于中心点估计的 Center Net[5]、基于关键点来检测匹配的 Corner Net[6] 以及 2019 年提出的适合小目标检测的 Efficent Det[7] 等。在这些方法中，基于 YOLO 系列的检测算法优势显著。它将目标检测视为回归问题，一次卷积运算即可同时预测目标类别与位置，流程简洁，速度超过其他三者，且采用端到端架构，直接输入图像就能一次性输出检测结果，无需多步骤或多模型组合，架构简洁高效。2016 年，文献[2]首次将 YOLOv1 模型引入目标检测领域。随后，官方对其进行了多个版本的迭代[8-9]。而随着 YOLO 系列的不断发展，许多学者也对 YOLO 系列进行了不同方式的改进[10-12]。二阶段模式，又被称为基于候选框的目标检测，其命名来源于检测过程分为两个部分。此类算法的整体思路是首先对输入图像选取建议框，然后对建议框进行分类和位置回归，进而得到最终的检测结果。二阶段模式的目标检测算法主要有：R-CNN[13]，Faster R-CNN[14]，Mask R-CNN[15] 等。近年来，Transformer 架构被众多学者引入目标检测领域，涌现出众多新颖的结构，具体可参考文献[16-19]。

作为目标检测领域的重要细分，车型检测聚焦于车辆类型的精准识别，在智能交通体系里扮演着关键角色。在一阶段车型检测方法中，相关研究从多个维度对模型进行优化改进以提升车型检测性能：特征提取网络优化方面：文献[20]通过更换为提取能力更强的特征提取网络，文献[21]和[22]采用向原模型引入注意力机制的方式，增强模型的检测能力。模型整体改进方面：文献[23]针对 YOLOv3，运用改进归一化层、聚类算法等手段，提高了该模型对于车型检测的精度。损失函数改进方面，文献[24]对 YOLOv3 的损失函数进行优化，提升了模型的车型检测精度。基于二阶段目标检测方法的车型检测可参考文献[25-28]。

将车型检测模型部署于树莓派等边缘计算设备不仅能在本地实时处理数据，不仅大幅减轻网络传输负担，避免网络拥塞，而且具备搭建与维护的成本低廉的优势，可以按需灵活增减设备，轻松应对临时场景，扩展性强。因而被越来越广泛地地应用在智能交通系统上。文献[29]将 Yolov5 目标检测算法部署在边缘计算设备树莓派上，对交叉口信息进行采集，实现对交通信号灯的实时控制。文献[30]把在 PC 端上训练好的 YOLOv5s 与 YOLOv5-Lite 目标检测模型分别部署在搭载 Linux 系统的树莓派 4B 平台上，并在此平台上搭建深度学习环境，构建道路行人检测系统。在车型检测领域，2020 年陈璐等使用树莓派和神经元计算棒只针对卡车进行检测识别[31]。Tangkocharoe 等使用树莓派进行车辆检测过程中并未对车辆进行分类[32]；Daher 等使用树莓派对人类、摩托车、普通车型、卡车进行检测，检测粒度较粗[33]。基于树莓派进行车型检

测还有较大发展空间。鉴于此,本文基于树莓派搭建对常见车型的低成本分类检测系统,旨在将其广泛应用到各出入场所。本文构建的车型检测的工作流程由三部分构成,如图 1 所示。第一部分是数据集的构造,经研究调查选择 10 类常见汽车车型进行分类检测;第二部分是通过合适的模型进行特征提取,模型训练,以及模型测试;第三部分是树莓派的部署与测试。

1 车型检测

1.1 数据集的构造

数据集是目标检测成功的关键一步。针对不同的检测问题,其数据集也不尽相同,制作适合项目需求的数据集至关重要。通过对出入场所的车辆进行观察并结合研究目的,本文确定对 10 类常见汽车车型进行分类检测,其中包括客车、私家车、SUV、小型货车、大型卡车、面包车、油罐车、吉普车、出租车、以及消防车,并以 bus、familysedan、SUV、truck、heavytruck、mbc、oilcar、jeep、taxi、fireengine 作为其对应的数据标签。针对当前公开数据集中的车型数据并不完全满足本次实验需求的问题,先从 StanfordCars 数据库中获取部分数据,又通过人工拍摄、网络爬虫的方式完成数据集的初步构建。其次,使用 LabelImg 工具包对原始数据进行标注,生成 xml 标签文件。然后,通过几何空间转换和颜色空间转换的方法对原始数据和标签文件进行数据增强。最后,剔除不合格数据,构造出包含 12 400 张图片和 12 400 个 xml 文件的车型数据集 TenCars。

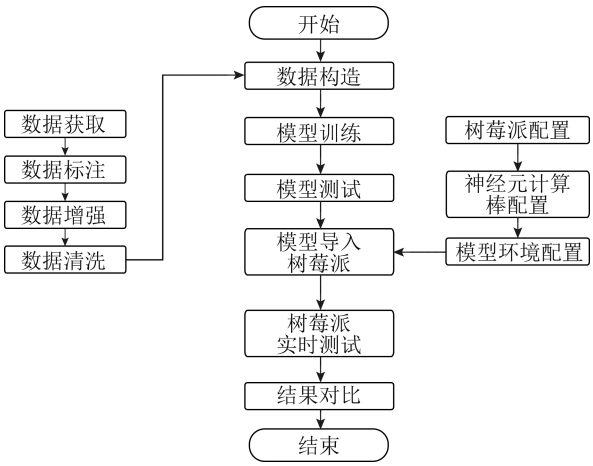


图 1 车型检测工作流程

Fig. 1 Vehicle model inspection workflow

1.1.1 几何空间变换。本文通过图像几何变换的方法对图像进行坐标系变换,从而达到图像扩增的目的。在坐标系变换中,旋转和偏移以图像中心为原点,具体变换过程如图 2 所示。在图像坐标系中通常以 A 为原点,以 AB、AC 方向作为原始坐标系方向(如图 3 所示)。然而,笛卡尔直角坐标系是将 D 点作为原点,坐标方向设定为 DE、DF 方向。在坐标转换过程中,可以将原始图像用 $M\times N$ 的矩阵进行表示。坐标转换后,原始坐标系中的原点 A 的坐标会从(0,0)转换为$(-N/2,M/2)$,原始图像的某一像素点(x,y)通过式(1)进行转换,得到笛卡尔坐标

$$\begin{cases} x=x'-N/2, \\ y=-y'+M/2, \end{cases} \tag{1}$$

逆变换转换关系式为

$$\begin{cases} x'=x+N/2, \\ y'=-y+M/2。 \end{cases} \tag{2}$$

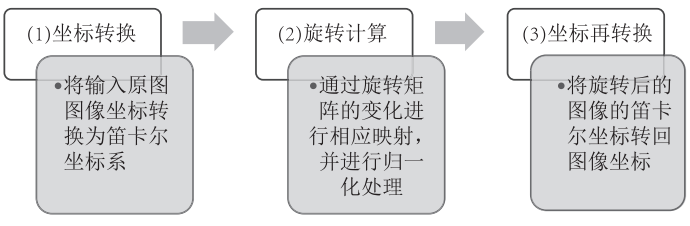


图 2 笛卡尔坐标转换流程图

Fig. 2 Flowchart of cartesian coordinate transformation

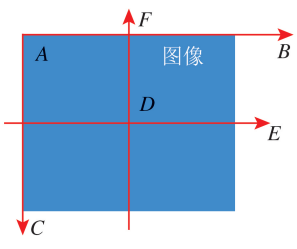
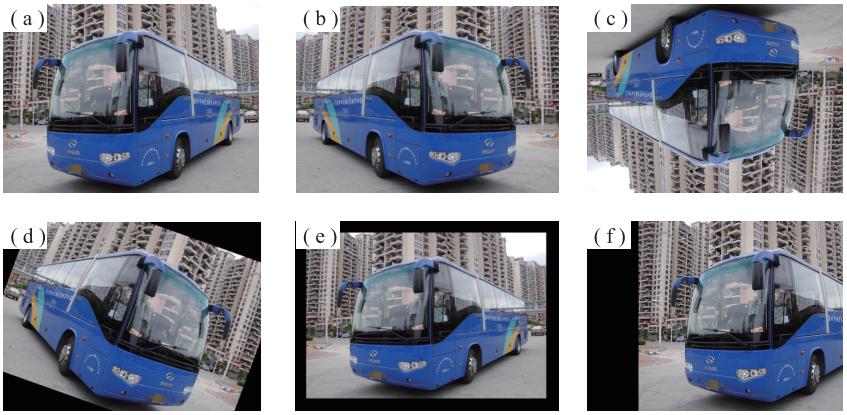


图 3 坐标转换图

Fig. 3 Diagram of coordinate transformation

根据笛卡尔坐标流程图的三个步骤进行三次变换,所得到的最终变换形式可以表示成式(3)

$$(x,y,1)=(x',y',1)\begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ -0.5\times N & 0.5\times M & 1 \end{bmatrix}\begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0.5\times N & 0.5\times M & 1 \end{bmatrix}。 \tag{3}$$



注：(a) 原图；(b) 水平翻转；(c) 垂直翻转；(d) 旋转 30 度；(e) 图像缩放；(f) 图像平移。

图 4　几何变换图像

Fig. 4　Geometric transformation image

常见的几何变换类型包含：翻转、平移、旋转、缩放、剪裁以及噪声注入的方法。本文选取了翻转、平移以及旋转的方式进行增强。图 4 是根据几何变换原理，使用 Python 语言对车辆图像进行了水平翻转、垂直翻转、旋转 30 度、缩放、平移操作后的图片。

1.1.2　*颜色空间变换*。光照偏差是图像检测常见挑战之一。在目标检测中，颜色空间变换行之有效。快速处理过亮或过暗图像的方法，是遍历图像像素，以恒定值增减像素值。为降低大雾天低能见度、避免正午强光曝光影响，对数据集实施对比度增强与亮度变换操作，增加数据多样性，提升模型训练的泛化能力。实验采用对比度增强和亮度调整手段，对数据集中图片进行颜色空间转换，减少光照与亮度对图像检测的干扰。

(1) 对比度增强。对比度增强算法主要通过反锐化掩模的技术进行增强，反锐化掩模的主要过程分为如图 5 所示的两个部分。进一步，由图 6 可以明显看到，对比度增强后的图片更加清晰，包含的图像特征更加明显，有助于之后在模型过程中的特征提取。

(2) 亮度调整。图像亮度调整是常用的数据增强手段。本次试验参照对比度、饱和度调整原理，在 RGB 空间下对图像进行亮度调整。基于 RGB 空间调整亮度时，计算亮度调整系数常采用式(4)所示方法对 RGB 值进行运算。并且在调整过程中，可对三个通道的值进行相同计算调整，无需考虑其数值大小。

$$\begin{cases} r = R_{GB} + R_{GB} \times (1/(1-v) - 1), & v > 0, \\ r = R_{GB} + R_{GB} \times v, & v \leqslant 0, \end{cases} \tag{4}$$

式中 r 为调整后图像的像素颜色值(rgb)，R_{GB} 表示原始图像的像素颜色值(RGB)，v 表示调整幅度的参数且取值范围为[−1,1]。

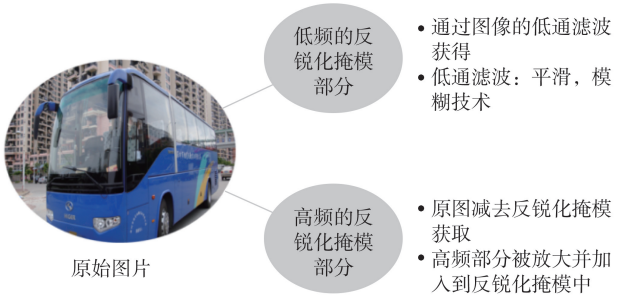


图 5　反锐化掩模技术

Fig. 5　Unsharp masking technique

注：(a) 原图；(b) 对比图。

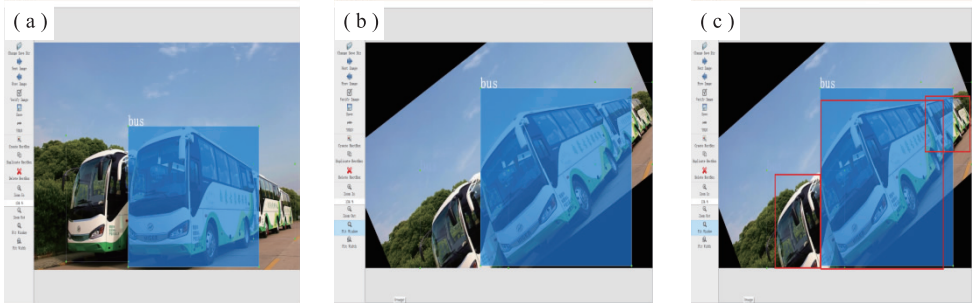
图 6　适应对比增强对比图

Fig. 6　Comparison diagram of adaptive contrast enhancement

1.1.3　*数据清洗*。通过数据获取和图像增强之后，本文共得到 1 400×9，即 12 600 张数据图片，因图片数据增强和 xml 文件增强同时进行，故在数据增强之后对 12 600 张图片进行图片校对、清洗，保证标签位置可以准确覆盖完整目标。经检验发现，对于旋转后的多目标图片，其与所覆盖标签存在一定的出入。图 7(a) 表示原始标记图片，图 7(b)表示旋转 30 度之后的图片，明显可以看出旋转后的图片的标记框超出了目标所

在区域,其正常应该标注区域应该如图 7(c)红色标记框所示。

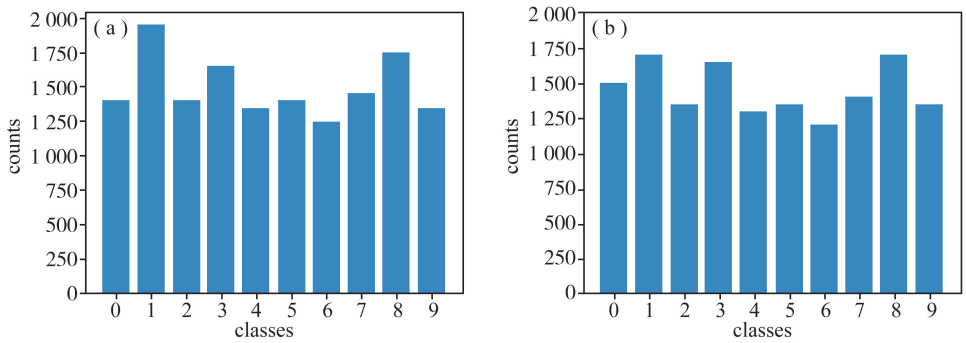
为保证数据均衡,在实验过程中对上述标签文件中的各类别标签个数进行了归纳整理,图 8 是转化成标签之后的各车型类别的标记框个数,在图 8(a)和图 8(b)中,横坐标均表示车型类别标签,其中标签 0 代表公共汽车,标签 1 代表私家车,标签 2 代表消防车,标签 3 代表大型卡车,标签 4 代表吉普车,标签 5 代表面包车,标签 6 代表油罐车,标签 7 代表 SUV,标签 8 代表出租车,标签 9 代表小型卡车;纵坐标表示对目标进行标记后所生成的标签的个数。



注:(a) 原始标图片;(b) 旋转 30 度图片;(c) 应矫正图片。

图 7 图像矫正对比图

Fig. 7 Image rectification comparison diagram



注:(a) 类别标签示意图;(b) 修改后的类别标签示意图。

图 8 数据清洗对比图

Fig. 8 Image rectification comparison diagram

通过图 8(a)可以看出标签 1 所代表的私家车的数量和标签 8 所代表的出租车数量明显大于其他类别数量,因此为了避免数据不均衡而产生的目标检测精度低的情况,在本次实验中采取人工清洗的方式对私家车和出租车的部分图片和 xml 标签文件进行了剔除操作,其中剔除私家车图片 160 张,出租车图片 40 张,另一方面针对图 8 中所出现的标注框不符合实际目标大小情况,进行了人工修改,最终得到的 12 400 张图片和 12 400 个 xml 标签文件构成本次车型数据集,并将其命名为 Ten Cars 数据集(如表 1 所示)。

1.2 基于 YOLO 的车型检测

1.2.1 模型简介。(1) YOLOv4-tiny 模型简介。YOLOv4-tiny 是 YOLOv4 的简化版,结构和 YOLOv4 相似,主要由输入端、主干框架(Backbone)、辅助框架(Neck)和输出端四个部分构成。其在 YOLOv4 的基础上减少了一些结构,从而在运行速度方面取得了大幅度的提升。YOLOv4 共有 168 层网络结构,约 6 000 万参数,YOLOv4-Tiny 则只有 38 层网络结构,约 600 万参数。图 9 详细描述了 YOLOv4-tiny 的网络结构组成,在训练过程中,把压缩到 416×416 或 608×608 像素的图片作为输入端,其次通过主干框架(Backbone)进行特征提取并构造 Neck 模块进一步提高特征提取的能力,最后通过非极大值抑制对目标进行预测。(2) Yolov5s 模型简介。YOLOv5s 算法的结构和 YOLOv4 相似,主要由输入端、Backbone、Neck 和输出端四个部分构成。图 10 详细描述了 YOLOv5s 的网络结构组成,在训练中,模型将图片压缩为 608×608 像素的图片作为输入端,然后通过主干框架进行特征提取,Neck 模块是 FPN+PAN 加成组成,以便进一步

提高特征提取的能力，最后通过非极大值抑制对目标进行预测。

表 1　车型数据库具体数量表

Tab. 1　Detailed quantity table of vehicle model database

车型	sfc 获取	爬虫清洗	人工拍摄	总计获取	数据增强	数据清洗
私家车	40	60	40	140	1 260	1 100
SUV	40	60	40	140	1 260	1 260
吉普车	40	100	0	140	1 260	1 260
小型货车	0	100	40	140	1 260	1 260
大型卡车	0	100	40	140	1 260	1 260
油罐车	0	100	40	140	1 260	1 260
消防车	0	100	40	140	1 260	1 260
出租车	0	100	40	140	1 260	1 220
面包车	0	100	40	140	1 260	1 260
客车	0	100	40	140	1 260	1 260
合计	120	920	360	1 400	12 600	12 400

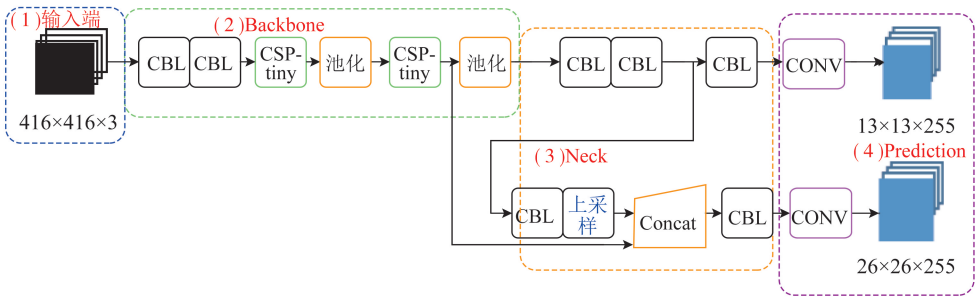


图 9　YOLOv4-tiny 网络结构图

Fig. 9　YOLOv4-tiny network architecture diagram

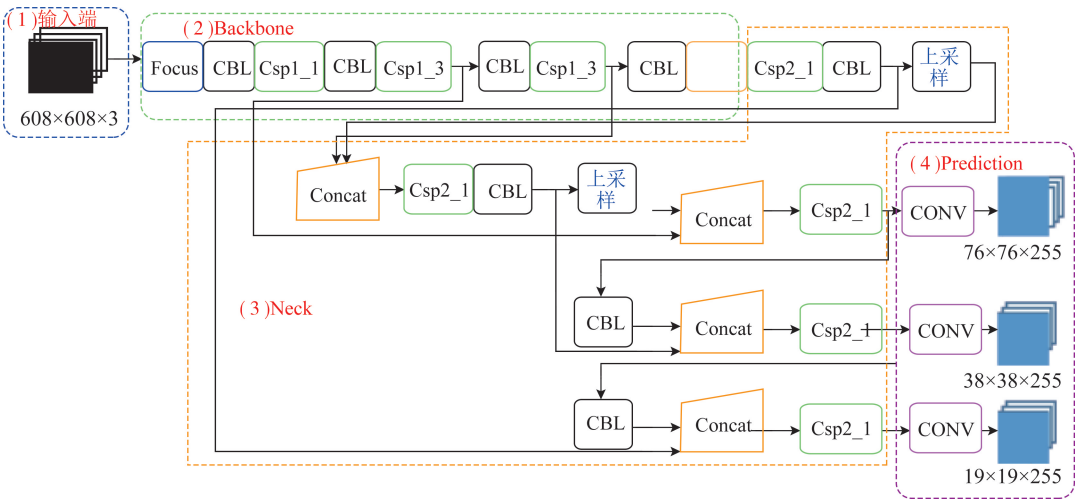


图 10　YOLOv5s 网络结构图

Fig. 10　YOLOv5s network architecture diagram

1.2.2　测试环境。此次 YOLOv4-tiny 和 YOLOv5s 是在 Windows 系统上实现的，具体环境为：硬件环境：Intel Core i5-9400F；NVIDIA GeForce RTx 1060；软件环境：Windows10 操作系统；操作系统环境：Python3.7.0；Pytorch1.6.0；CUDA 10.2；CUDNN7.6.5；Opencv4.4.0。

1.2.3　实验过程。针对树莓派开发板 AI 推理性能低的问题，选取 YOLOv4-tiny 和 YOLOv5s 在 TenCars 数据集上进行模型训练。本次实验在 Pytorch 框架上训练 YOLOv4-tiny 模型，在模型训练过程中首次选择

官方 YOLOv4-tiny. weights 权重文件,进行 YOLOv4-tiny 的网络的训练。在实验训练过程中,首先按 4∶1 的比例将数据集随机进行划分。在参数调节部分,将数据集导入到 YOLOv4-tiny 网络后,为保证模型具有较高的运行速度,实验选择将图片大小压缩到 416×416,同时在实验训练过程中不断调节模型参数。实验发现在 TenCars 数据集训练过程中,将学习率设置为 0.01,迭代次数(epoch)设置为 100 次,每批次大小设置为 16 时,模型会得到较高的召回率和准确率。整个模型在章节 1.2.2 中所描述的环境中的运行时长为 22 小时。将训练结果依次保存在 result.log 文件中,并通过 Matplotlib 工具包对 CIoU、目标损失(obj)、分类损失(cls)、总损失(total)、精确率(P)、召回率(R)以及两种平均准确率进行可视化分析,如图 11 所示。当迭代 100 次时,模型已经达到收敛。此时,边框损失误差为 0.012 2;置信度误差值为 0.022 896;分类误差为 0.003 073;总误差损失为 0.018 16。精确率为 85.65%,召回率为 99.9%,mAP_5 为 99.18%,mAP_5_95 为 86.05%。其中 mAP_5 表示交并比(Iou)设为 0.5 时的精度均值;mAP_5_95 表示 IoU 从 0.5 到 0.95 的精度均值,0.5 到 0.95 之间的步长为 0.05。通过训练结果可知,该训练模型达到了较高水平,特别是召回率达到了 99.9%,说明模型较为全面的找出目标样本。

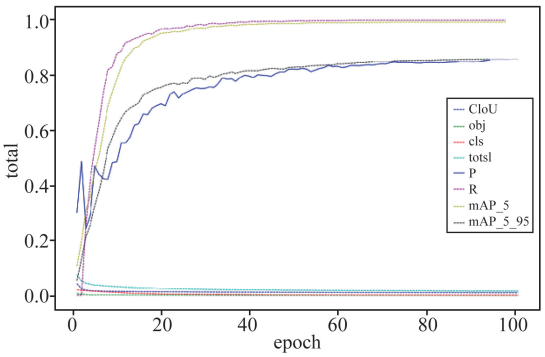


图 11 模型训练结果可视化

Fig. 11 Visualization of model training results

YOLOv5s 实验同样是在 Pytorch 框架上训练的,其选择官方 3.0 版本的 YOLOv5s.pt 作为首次训练的权重文件。在实验过程中,按照 4∶1 的比例随机划分训练集和验证集的图片,然后图片通过 Mosaic 数据增强,自适应图像调整到 608×608 的尺寸进行输入。在参数调节部分,设置迭代次数 epoch 为 120 次,并且将每批次大小设置为 16,学习率设置为 0.01,训练过程中设置各迭代的训练结果会依次保存在 result.txt 文件中,最后在 1.2.2 中所描述的环境中运行 30 h 后结束训练。通过 Matplotlib 工具包对保存的结果进行可视化。图 12 记录了数据集在训练过程中训练集的损失变化情况,其中包含 GIoU、目标损失(obj)、分类损失(cls)以及总损失(total),可以看到模型在迭代 120 次时,各 loss 数值已达到较低水平,且已经达到收敛状态。

通过图 13 可以清楚的发现模型训练精度也已达到较高的水平。在训练结束后,会在模型文件所指定的文件夹中得到两个权重文件,分别是 best.pt 和 last.pt,其中 best.pt 是训练迭代 120 次过程中得到的最好的一个权重文件,last.pt 是最后一次训练所得的权重文件。这样的操作有助于在测试阶段减少测试工作量,直接选取 best.pt 权重文件进行最终测试,究此原因本文中不对 YOLOv5 迭代第 120 次的损失值和精确值进行详细标注说明。

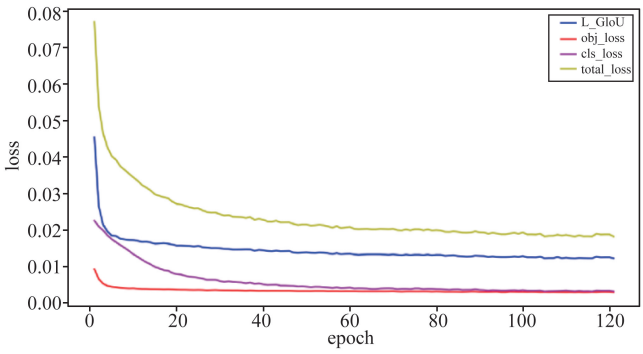


图 12 YOLOv5s 模型训练 loss 示意图

Fig. 12 YOLOv5s model training loss diagram

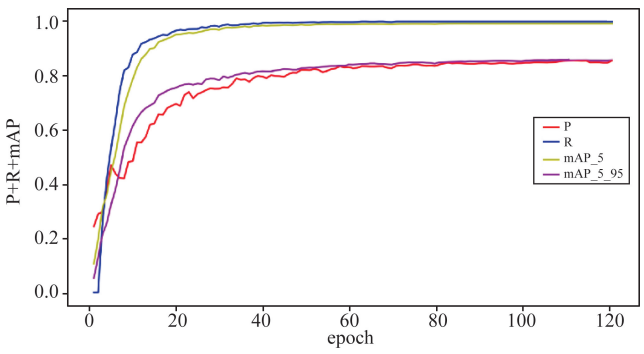


图 13 YOLOv5s 模型精度示意

Fig. 13 YOLOv5s model accuracy diagram

1.3 基于树莓派的车型检测

1.3.1 树莓派简介。树莓派(Raspberry PI)是只有信用卡大小的卡片式电脑,是目前最小的台式机,自其

问世以来，经历了 A 型、A＋型、B 型、B＋型、2B 型、3B 型、3B＋型、4B 型等型号的演进。为适应大众需求，树莓派也得到了不断的升级，目前为止树莓派已经从二代发展到树莓派四代。在发展过程中，其最大的进展在于处理器的不断升级。2019 年 6 月发行的树莓派 4B 型，其搭载 1.5GHz 的 64 位四核处理器，适用于 3A 的驱动电流，且支持多个操作系统。其中被广泛应用的操作系统包括 Raspbian；Snappy Ubuntu Core；OpenELEC；Ubuntu Mate 等。拥有诸多操作系统的树莓派可以凭借其自身性能完成很多设备无法完成的事情，所以运用树莓派与深度学习进行结合，非常有利于在不同场景下的目标检测。

1.3.2　测试环境。在完成基础的模型训练和测试之后，针对文件编译和模型移植过程中可能产生的损耗问题，本文将 YOLOv4-tiny 模型和 YOLOv5s 导入到图 14 所示的嵌入式开发设备——树莓派中进行测试。测试环境为，(1) 硬件环境：8G 处理器的树莓派开发板第四代。英特尔 Movidius 神经元计算棒第二代×2；64GB 的高速内存卡。(2) 软件环境：树莓派官方操作系统(Raspbian)。(3) 操作系统虚拟环境：Python 3.7.0；Pytorch1.6.0；Torchvision0.8.0；Opencv4.4.0；OpenVINO。

基于 YOLO v4-tiny 和 YOLO v5s 网络进行目标检测，在一定程度上提升了树莓派目标检测速度。但是通过测试发现，仅仅使用树莓派这种 CPU 处理器，并不能完全实现实时目标检测。

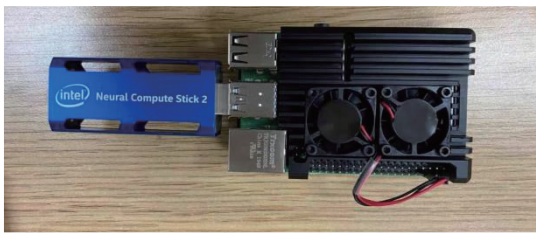


图 14　树莓派＋Movidius 装置示意图

Fig. 14　Schematic diagram of raspberry pi ＋ movidius device

在此基础上，本文提出用 Movidius 神经元计算棒来加速网络计算，进而提高检测速度[34-35]。英特尔开发的第二代神经计算棒(Neural Compute Stick 2/NCS 2)是一个深度学习推理工具和独立的人工智能加速器。该类加速器日常功耗极低，并且可以进行 AI 推理以及视觉运算，同时支持 Caffe、TensorFlow 等开发框架。

Movidius 神经元计算棒具有以下特性：(1) Movidius VPU 具备高能效 CNN 处理功能，支持 CNN 分析、原型构建和调优流程。(2) 所有数据和电源均通过单个 USBType A 端口提供，简单方便。(3) 实时设备上推理无需云连接，不仅支持在 Ubuntu 系统的开发应用，同时支持嵌入式设备树莓派的开发。(4) 在同一平台上可运行多台设备以扩展性能。(5) Open Model Zoo 上有预先训练的模型，可以快速部署现有 CNN 模型或经过训练的网络。

1.3.3　实验过程。(1) 树莓派部署过程。在实验过程中，本文选取树莓派官方镜像 Raspbian 作为树莓派开发板的操作系统，配置树莓派的主要操作流程为：1) 将从树莓派官网下载的树莓派镜像通过 Win32DiskImager 导入到准备好的干净的内存卡中。2) 将已成功写入 Raspbian 镜像的 SD 卡插入到准备好的树莓派中，然后对树莓派安装界面进行配置，即需要打开相应的 SSH 服务、无线网络连接和相机权限，以方便进行远程操控以及实时视频。3) 使用在同一局域网内的电脑设备远程连接树莓派，更换树莓派的官方源为清华或豆瓣源，以便提高所需软件源的下载速度。4) 使用 sudoapt-getupdate 和 sudoapt-getupgrade 命令，更新和升级软件源列表。5) 树莓派配置完成，重启树莓派。(2) Movidius 神经元计算棒配置。使用 Movidius 神经元计算棒来加速计算需要通过两个步骤来完成，第一步需要完成的是在树莓派上部署 OpenVINO，第二步则需要对上述实验得到的 YOLOv4-tiny 和 YOLOv5s 的权重文件进一步进行编译，使之可以被 Movidius 神经元计算棒所运行。主要过程是把训练得到的. Weights 格式的权重文件先转换为. pb 格式的静态文件，然后再转换为中间表示模型文件。

2　实验结果分析

在 YOLOv4-tiny 和 YOLOv5s 训练精度均较高的情况下，本节选取 100 张与训练集完全不同的图片作为测试集开展模型测试，并从测试速度和平均精度(mAP)两方面进行分析。同时，鉴于不同分辨率图片进行目标检测时运行速度有别，本节针对分辨率为 320×640 的车辆进出入视频展开速度测试。

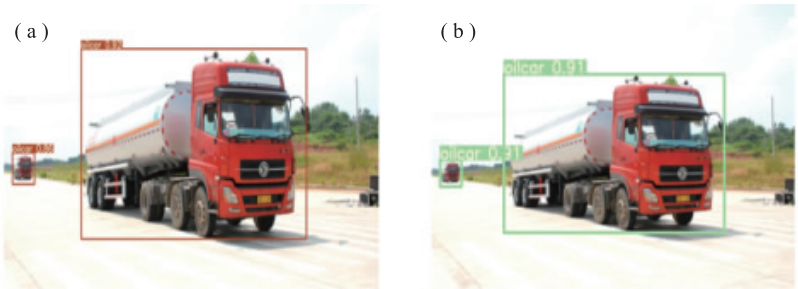
2.1 测试图片分析

本次实验目的是满足实时情况下对进出入口的车型进行检测,故在测试过程中需要对远近不同,角度不同的车辆图片进行测试。在本次测试过程中,本文选取了与训练集完全不同的 100 张图片进行测试,其中 SUV、油罐车、面包车、公共汽车、消防车、私家车、出租车、小型卡车、大型卡车、吉普车各 10 张图片。图 15 为随机选取的一组测试图片在 YOLOv4-tiny 模型下的检测情况。

图 15 十类车型检测

Fig. 15 Detection of ten vehicle categories

通过单纯的观察测试图片发现,用 YOLOv5s 训练得到的 best. pt 权重文件进行测试和 YOLOv4-tiny 选择迭代 100 次的权重文件进行测试时,测试集中均无误检情况,且对于近景车辆场景无漏检情况。图 16(a)为 YOLOv4-tiny 检测结果,图 16(b)为 YOLOv5s 检测结果,可以发现两者对远景小目标车辆也可以进一步检测,虽在置信度方面略有不同,但其数值均在 0.85 以上,相对较高。



注:(a) YOLO v4-tiny 测试图片;(b) YOLO v5s 测试图片。

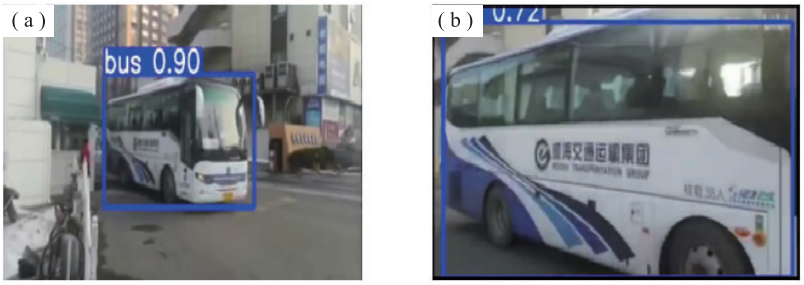
图 16 目标远景检测对比图

Fig. 16 Comparison diagram of distant target detection

在神经元计算棒支持下的树莓派对烟台公交汽车站的车辆进出入情况进行测试时,视频较为流畅,且模型对近距离车辆进行测试的置信度较高。图 17 是随机选取的 YOLOv4-tiny 模型视频测试图。通过观察发现,树莓派摄像头的安装位置对模型检测有一定的影响,即要充分考虑摄像头安装的高度、远度与角度。本次模型对车型展示较为完全的图片,识别效果更好。因此在实际应用中,应该注意摄像头的安装位置,避免因设备安装问题造成的精度降低情况。

2.2 mAP 精度分析

总类别精度均值(mAP) M_ρ 衡量的是训练好的模型在所有类别上的精确度,其具体的计算公式为 $M_\rho=\frac{\sum_{i=1}^{C}\rho_i}{C}$,式中 C 为类数目,ρ_i 为所有图片内的某一类的精度值(AP)。



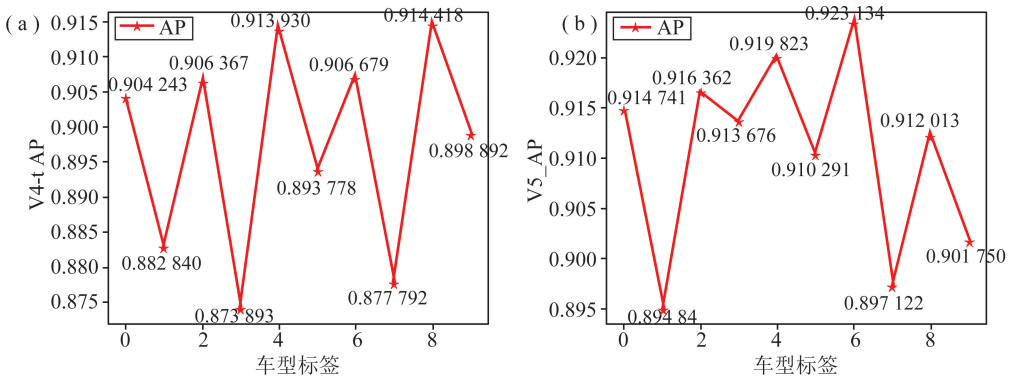
注：(a) 合适角度测试图；(b) 不合适角度测试图。

图 17　车辆进出入测试图片

Fig. 17　Vehicle ingress and egress test images

图 18 表示通过 YOLOv4-tiny 和 YOLOv5s 模型测试得到的各车型的检测精度。根据 mAP 计算公式可以得到，当 IoU 设置为 0.5 时，使用 YOLOv4-tiny 和 YOLOv5s 模型对 100 张与训练集完全不同图片进行测试，mAP 分别为 89.73% 和 91.04%。图 18 中横坐标展示的标签所对应的车型以及各车型在 YOLOv4-tiny 和 YOLOv5s 模型测试得到的具体 AP 值如表 2 所示。

根据 YOLOv4-tiny 和 YOLOv5s 模型测试得到的实验结果可以发现，在模型精度方面，两者的 mAP 相差不大，检测精度相对较高，YOLOv5s 模型精度略高于 YOLOv4-tiny。且根据前文测试图片对比分析初步判断，模型精度受到了数据集不够均衡的影响。



注：(a) YOLOv4-tinyAP 值；(b) YOLO v5sAP 值。

图 18　测试图片 AP 值对比图

Fig. 18　Comparison diagram of AP values on test images

表 2　模型 AP 对比图

Tab. 2　Comparison diagram of model AP

标签	车型	v4-tiny_AP/%	v5_AP/%
0	公共汽车	90.424 3	91.474 1
1	私家车	88.284	89.488 4
2	消防车	90.636 7	91.636 2
3	大型卡车	87.389 3	91.367 6
4	吉普车	91.393 0	91.982 3
5	面包车	89.377 8	91.029 1
6	油罐车	87.779 2	89.712 2
7	SUV	91.441 8	91.201 3
8	出租车	89.889 2	90.175 0
9	小型卡车	90.889 2	91.037 0

2.3 运行速度分析

在模型移植不会影响其精确度的情况下,本小节主要从运行速度方面来对本次实验的可行性进行分析。在实验过程中将每秒传输帧数(FPS)作为速度测试的评价指标。包含四种实验环境下 YOLOv4-tiny 模型的目标检测速度以及画面每秒传输帧数(FPS),实验环境分别为 Windows 系统、无神经元计算棒支持的的树莓派系统、有一个神经元计算棒支持的树莓派系统和有两个神经元计算棒支持的树莓派系统。此外,本文还选取 YOLOv5s 模型在 TenCars 车型数据集训练测试进行对照,实验对比结果如表 3 所示。

表 3 不同算法对比表

Tab. 3 Comparison table of different algorithms

模型	实验环境	t/s	FPS	mAP/%
YOLOv4-tiny	Windows	0.019	52.34	89.73
	Pi	0.91	1.10	89.73
	Pi+Movidius	0.15	6.69	89.73
	Pi+Movidius×2	0.078	12.76	89.73
YOLOv5s	Windows	0.026	37.86	91.04
	Pi	1.89	0.53	91.04
	Pi+Movidius	0.39	2.51	91.04
	Pi+Movidius×2	0.15	6.55	91.04

注:t 表示单张图像推理时间以秒为计算单位。

在两个神经元计算棒支持的树莓派环境中进行测试时,YOLOv4-tiny 的模型进行检测的运行速度为 12.76 f/s,测试视频较为流畅;而 YOLOv5s 测试的运行速度为 6.55 f/s,视频会出现卡顿的情况。因此虽然 YOLOv4-tiny 的模型精度略小于 YOLOv5s,但其在运行速度方面超过 YOLOv5s,更适合植入到树莓派这种嵌入式开发设备当中进行检测,因此本文最终选择 YOLOv4-tiny 模型移植到树莓派,以便其进一步的开发。

2.4 实验总结

根据 2.3 可知,基于 YOLOv5s 对车型进行检测的 mAP 已达到 91.04%,YOLOv4-tiny 的车型检测 mAP 为 89.73%,两者检测的精度相差不大。在模型移植不会影响其精确度的情况下,本小节主要从运行速度方面来对本次实验的可行性进行分析,并选择最合适的模型做最终实验输出。

本次实验同样对记录了烟台公交汽车站的车辆进出入情况的视频进行测试,选取每秒传输帧数(FPS)作为评价指标。实验结果如表 4 所示,包含以下信息:在 Windows 系统、无神经元计算棒支持的的树莓派和有神经元计算棒支持的树莓派三种实验环境下,YOLOv4-tiny 模型和 YOLO5s 模型的目标检测速度以及画面每秒传输帧数(FPS)。

表 4 运行速度对比表

Tab. 4 Comparison table of running speeds

模型	实验环境	t/s	FPS
YOLO v4-tiny	Windows	0.025 0	40
	Raspberry	1.113 0	0.884 0
	Raspberry+OpenVINO	0.149 3	6.698 0
YOLO v5s	Windows	0.032 3	31
	Raspberry	2.299 0	0.435 1
	Raspberry+OpenVINO	0.397 9	2.513 0

注:t 表示单张图像推理时间以秒为计算单位。

通过表 4 可以判断虽然在本次实验中 YOLOv4-tiny 的模型精度略小于 YOLOv5s，但其在运行速度方面超过 YOLOv5s，更适合植入到树莓派这种嵌入式开发设备当中进行实时检测，因此本文最终选择 YOLOv4-tiny 模型移植到树莓派，以便其进一步的开发。

3　总结和展望

本文收集了公共场所常见的十种车型图片构造成实验的车型数据集，选取 YOLOv4-tiny 和 YOLOv5s 模型对 Ten Cars 数据集进行训练，并将两种模型部署在由树莓派和 Movidius 神经元计算棒搭建的边缘性计算设备上。实验结果表明，在 Movidius 神经元计算棒的支持下，该系统可以进行流畅的车型检测，且 YOLOv4-tiny 比 YOLOv5s 更适合进行基于树莓派和神经元计算棒的实时检测。本文的工作在一定程度上有助于智能车型检测系统的普及，加快智能管理的发展步伐。但当前情况下，此类技术仍存在部分地方可以改进以进行后续研究，其中主要包括两方面：(1) 鉴于树莓派存在性能受限问题，只能选取轻量级网络进行实时检测。在模型精度方面，相对其他深度网络存在一定的欠缺。同时，为满足实时检测要求，此次只针对车型属性进行检测，暂无法应用到车辆颜色、车标等多属性同时检测。(2) 当前 YOLOv5s 的运行速度相对 YOLOv4-tiny 较低，其耗时集中在神经棒的推理中，同时转换后的中间表示模型文件在 CPU 上推理的置信度差异相对较高，存在一定的优化空间。

参　考　文　献

[1] REDMON J, DIVVALA S, GIRSHICK R, et al. You only look once: unified, real-time object detection[C] //Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, 2016: 779-788.

[2] REDMON J, FARHADI A. YOLO9000: better, faster, stronger[C]//Proceedings of IEEE Conference on Computer Vision and Pattern Recognition. 2017: 6517-6525.

[3] CHOI J, CHUN D, KIM H, et al. Gaussian YOLOv3: an accurate and fast object detector using localization uncertainty for autonomous driving[C]//Proceedings of IEEE/CVF International Conference on Computer Vision, 2019: 502-511.

[4] LIU W, ANGUELOV D, ERHAN D, et al. SSD: single shot multi-box detector[C]// European Conference on Computer Vision, 2016: 21-37.

[5] DUAN K W, BAI S, XIE L X, et al. CenterNet: keypoint triplets for object detection[C]//Proceedings of IEEE/CVF International Conference on Computer Vision, 2019: 6568-6577.

[6] LAW H, DENG J. Corner Net: detecting objects as paired key points[EB/OL]. arXiv: 1808.01244, 2018.

[7] TAN M X, PANG R M, LE Q V. EfficientDet: scalable and efficient object detection[C]//Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020: 10778-10787.

[8] ZHU X K, LYU S C, WANG X, et al. TPH-YOLOv5: improved YOLOv5 based on transformer prediction head for object detection on drone-captured scenarios[C]//Proceedings of IEEE/CVF International Conference on Computer Vision Workshops. 2021: 2778-2788.

[9] WANG C Y, BOCHKOVSKIY A, LIAO H M. YOLOv7: trainable bag-of-freebies sets new state-of-the-art for real-time object detectors[C]//Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2023: 7464-7475.

[10] ZHAO C, SHU X, YAN X, et al. RDD-YOLO: a modified YOLO for detection of steel surface defects[J]. Measurement, 2023, 214: 112776.

[11] 张杨，辛国江，王鑫，等. 基于改进的 YOLOv5 网络的舌象检测算法[J]. 计算机技术与发展，2024，34(2)：156-162.

[12] ELFWING S, UCHIBE E, DOYA K. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning [J]. Neural Networks, 2018, 107: 3-11

[13] GIRSHICK R, DONAHUE J, DARRELL T, et al. Rich feature hierarchies for accurate object detection and semantic segmentation [C]//Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, 2014: 580-587.

[14] REN S Q, HE K M, 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, 2016, 39(6): 1137-1149.

[15] HE K M, GKIOXARI G, DOLLÁR P, et al. Mask R-CNN[C]//Proceedings of IEEE International Conference on Computer Vision, 2017: 2961-2969.

[16] CARION N, MASSA F, SYNNAEVE G, et al. End-to-end object detection with transformers[C]//In: Computer vision-ECCV 2020,

Springer International Publishing, 2020: 213-229.

[17] ZHU X, SU W, LU L, et al. Deformable DETR: Deformable transformers for end-to-end object detection[EB/OL]. arXiv preprint arXiv:2010.04159, 2020.

[18] WANG T, YUAN L, CHEN Y P, et al. PnP-DETR: towards efficient visual analysis with transformers[C]//Proceedings of IEEE/CVF International Conference on Computer Vision, 2021: 4661-4670.

[19] DAI Z G, CAI B L, LIN Y G, et al. UP-DETR: unsupervised pre-training for object detection with transformers[C]//Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021: 1601-1610.

[20] 毛昭勇, 王亦晨, 王鑫,等. 面向高速公路的车辆视频监控分析系统[J]. 西安电子科技大学学报, 2021, 48(5): 178-189.

[21] 於积荣, 黄德启, 曾蓉,等. 基于改进 YOLOv4 的车型检测算法研究[J]. 激光杂志, 2022, 43(4): 52-59.

[22] 王乃洲,金连文,赵清利,等. 基于 Diou-Yolov3 的车型检测[J]. 电子技术与软件工程, 2020(4): 148-150.

[23] 林轶, 陈琳, 王国鹏,等. 改进的 YOLOv3 交通标志识别算法[J]. 科学技术与工程, 2022, 22(27): 12030-12037.

[26] ARORA N, KUMAR Y, KARKRA R, et al. Automatic vehicle detection system in different environment conditions using fast R-CNN [J]. Multimedia Tools and Applications, 2022, 81(13): 18715-18735.

[24] 章程军, 胡晓兵, 牛洪超. 基于改进 YOLOv5 的车辆目标检测研究[J]. 四川大学学报(自然科学版), 2022, 59(5): 73-81.

[25] 李致金,张亮,武鹏,等. 基于特征融合卷积神经网络的车型精细识别[J]. 计算机工程与设计, 2020,41(1): 226-230.

[26] ARORA N, KUMAR Y, KARKRA R, et al. Automatic vehicle detection system in different environment conditions using fast R-CNN [J]. Multimedia Tools and Applications, 2022,81(13): 18715-18735.

[27] 江志鹏,王自全,张永生,等. 基于改进 Deformable DETR 的无人机视频流车辆目标检测算法[J]. 计算机工程与科学, 2024, 46(1): 91-101.

[28] DESHMUKH P, SATYANARAYANA G S R, MAJHI S, et al. Swin transformer based vehicle detection in undisciplined traffic environment[J]. Expert Systems with Applications, 2023, 213: 118992.

[29] 王新宇. 基于 Yolo v5 目标检测算法的交通信号灯智能控制装置设计[J]. 公路交通技术, 2022, 38(1): 142-148.

[30] 郑尚坡,陈德富,邱宝象,等. 基于树莓派与 YOLOv5-Lite 模型的行人检测系统设计[J]. 计算机代, 2023(9): 116-119.

[31] 陈璐, 管霜霜, 谢艳芳. 基于树莓派与神经计算棒的特种车辆检测识别[J]. 计算机系统应用, 2020, 29(9): 142-148.

[32] TANGKOCHAROEN T, SRISUPHAB A. Vehicle detection on a pint-sized computer[C]//Proceedings of 9th International Conference on Knowledge and Smart Technology, 2017, Chonburi, Thailand,2017:40-44.

[33] DAHER A W, RIZIK A, RANDAZZO A, et al. Pedestrian and multi-class vehicle classification in radar systems using rulex software on the raspberry pi[J]. Applied Sciences, 2020, 10(24):9113.

[34] 李倩, 原建平. Movidius™ 神经计算棒的测评[J]. 网络新媒体技术, 2020, 9(2): 51-59.

[35] 张海清, 张生. 基于 Movidius 神经计算棒的物体检测研究[J]. 计算机技术与发展, 2020, 30(10): 31-36.

(责任编辑:赵海军)