Journal of Real-Time Image Processing (2024) 21:174
https://doi.org/10.1007/s11554-024-01553-2

RESEARCH

High-precision real-time autonomous driving target detection based on YOLOv8

Huixin Liu[1] · Guohua Lu[2] · Mingxi Li[3] · Weihua Su[4] · Ziyi Liu[1] · Xu Dang[1] · Dongyuan Zang[1]

Received: 25 April 2024 / Accepted: 6 September 2024

Abstract

In traffic scenarios, the size of targets varies significantly, and there is a limitation on computing power. This poses a significant challenge for algorithms to detect traffic targets accurately. This paper proposes a new traffic target detection method that balances accuracy and real-time performance—Deep and Filtered You Only Look Once (DF-YOLO). In response to the challenges posed by significant differences in target scales within complex scenes, we designed the Deep and Filtered Path Aggregation Network (DF-PAN). This module effectively fuses multi-scale features, enhancing the model's capability to detect multi-scale targets accurately. In response to the challenge posed by limited computational resources, we design a parameter-sharing detection head (PSD) and use Faster Neural Network (FasterNet) as the backbone network. PSD reduces computational load by parameter sharing and allows for feature extraction capability sharing across different positions. FasterNet enhances memory access efficiency, thereby maximizing computational resource utilization. The experimental results on the KITTI dataset show that our method achieves satisfactory balances between real-time and precision and reaches 90.9% mean average precision(mAP) with 77 frames/s, and the number of parameters is reduced by 28.1% and the detection accuracy is increased by 3% compared to the baseline model. We test it on the challenging BDD100K dataset and the SODA10M dataset, and the results show that DF-YOLO has excellent generalization ability.

Keywords DF-YOLO · Traffic target detection · Multi-scale fusion · Parameter sharing

1 Introduction

The continuous development of autonomous driving technology has made unmanned driving possible, with perception models playing a crucial role in this field. Perception is one of the core modules of autonomous driving systems, with traffic target detection being a vital component of the perception module. Rapid and accurate detection can assist drivers or autonomous vehicles in making decisions earlier, thereby enhancing safety.

Detection algorithms extract information related to objects, behaviors, key points, etc., from images and detect objects from the background. Traditional detection algorithms often employ sliding windows to identify candidate bounding boxes. Subsequently, relevant features are extracted from these regions. Common feature descriptors encompass Haar features [1], Histogram of Oriented Gradients (HOG) features [2], and others. Eventually, classifiers such as Support Vector Machines (SVM) [3] and AdaBoost [4] are employed to classify features, thus discerning the location and type of the object. Nevertheless, these conventional methodologies rely on prior knowledge and are primarily suited to uncomplicated scenes. In complex and evolving environments, their performance is deficient, falling short of meeting the demands of practical applications. In recent years, deep learning-based detection methods have exhibited notable performance in both accuracy and real-time processing. These methodologies can be divided into two primary categories: one includes single-stage detection algorithms like You Only Look Once (YOLO) [5–10] and Single Shot MultiBox Detector

✉ Weihua Su
13512062095@126.com

[1] School of Artificial Intelligence, Hebei University of Technology, Tianjin 300000, China

[2] Department of Military Biomedical Engineering, Air Force Medical University, Xian 710032, China

[3] Academy of Military Transportation, Tianjin 300000, China

[4] School of Mechanical Engineering, Hebei University of Technology, Tianjin 300000, China

Published online: 19 September 2024

(SSD) [11], which directly conduct detection on the image, achieving efficient real-time performance. Another category includes two-stage detection algorithms like Fast R-CNN [12] and Faster R-CNN [13]. These algorithms initially generate candidate regions and subsequently perform classification and localization tasks. These algorithms often demonstrate superior accuracy performance.

Target detection based on deep learning has been widely used in fields, such as underwater object detection [14], driver fatigue detection [15], and small target detection [16]. Nonetheless, in complex traffic scenes where there are significant differences in target sizes, especially for detecting small targets such as pedestrians and distant vehicles, the performance is unsatisfactory. In real-world scenes, where targets move quickly the time for capturing and detecting is short, and computational resources are limited. Detecting traffic objects rapidly and accurately becomes even more challenging. Di et al. [17] studied target detection and tracking in such dynamic scenes and achieved good results. The present study proposes an improved YOLOV8 algorithm to address the aforementioned issue, and significantly enhance the model's capability to detect multi-scale targets while balancing detection speed and precision. The contributions of this paper are as follows:

1. To address the significant scale differences between vehicles and pedestrians, we design a Deep and Filtered Path Aggregation Network (DF-PAN), which uses the characteristics of deep features and shallow features, and the shallow information is filtered through the weights generated by the deep features, filtering out redundant semantic information, and highlighting the underlying small targets; filtering the deep information through the weights generated by the shallow features, filtering out redundant positioning information and highlight the semantic information of the target, and achieving more efficient multi-scale fusion.

2. To address the challenge of limited computational resources, we devise a parameter-sharing detection, transmitting the feature map to the Conv_share module. We incorporate a scaling factor to modulate the feature map after its traversal through the Conv_share module, yielding three distinct outputs. Implementing parameter sharing in the detection component enhances the feature classification ability of the model while reducing the number of parameters.

3. To tackle the complexity and high computational cost of the YOLOV8 network, we use FasterNet[18], a rapid neural network, to improve feature extraction speed and reduce model complexity.

The DF-YOLO algorithm significantly improves precision while reducing the number of model parameters. We test our method on the KITTI dataset, which reaches 90.9% mAP with 77 frames/s, and the number of parameters is only 2.3 m. This is a 3% mAP improvement over the baseline model (YOLOv8-n) and a 28.1% reduction in the number of parameters. In the context of autonomous driving scenes, it is essential to consider key factors, such as model precision, speed, and the number of parameters to ensure vehicle safety during operation. Our method achieves satisfactory balances between real time and precision and is suitable for autonomous driving scenes.

The paper is structured as follows: Sect. 2 provides a detailed review of existing research work. Following that, Sect. 3 delves into the improvement details of the algorithm. In Sect. 4, a comprehensive description and analysis of the experimental procedures are provided. Finally, Sect. 5 summarizes the work carried out in this study.

2 Related work

In recent years, numerous scholars have conducted extensive research on detection tasks in traffic scenes [19–21]. Hu et al. [22] proposed a cascaded vehicle detection method that integrates multi-feature fusion with Convolutional Neural Networks (CNN), demonstrating exceptional robustness in complex driving environments. R. Ghosh [23] introduced a Faster R-CNN road vehicle detection approach employing multiple Region Proposal Networks (RPNs) of varying sizes to effectively detect vehicles of different scales, achieving promising results. HAN [24] proposed a convolutional neural network (CNN) enhanced with contextual information, which effectively improves the accuracy of detecting small-sized and occluded vehicles by progressively integrating shallow-layer information into deep-layer networks. Although the aforementioned detection models achieve high precision, they fail to meet real-time requirements due to the limited computational performance of onboard vehicle systems in real-world scenes. Single-stage detection networks are more suitable for real-time traffic object detection in terms of detection speed compared to two-stage networks. Goran Oreski [25] proposed a YOLO*C algorithm that incorporates the MCTX (Multi-Context) context-aware module, efficiently utilizing rich global context information and significantly improving the detection accuracy of small targets in complex traffic environments. KANG et al. [26] introduced a novel YOLO detector called YOLO-FA based on fuzzy attention, which utilizes fuzzy entropy to weight features and enables the network to focus on targets, thereby effectively enhancing the detection accuracy of vehicles. However, these scholars did not address the issue of reduced accuracy due to differences in target scales.

Scale problem lies at the heart of every object detection, and many researchers have made outstanding contributions in this area. Li et al. [27] introduced a depth-based segmentation method and a multi-scale detection network

aimed at substantially improving small object detection in vehicle detection systems, and validate its effectiveness through experimental verification. Yuan et al. [28] introduced a multi-scale feature network into the detection model to more accurately extract the features of small targets in traffic scenes. SD Khan et al. proposed a robust method for producing object proposals in a paper [29], encoding objects of different sizes at different scales and achieving satisfactory results. In another paper [30], they proposed addressing the issue of scale variation by utilizing feature maps from three dense blocks to construct three Region Proposal Networks (RPNs). Each RPN is designed to target objects of different sizes, thereby generating multi-scale object proposals. This approach enhances the detection capability for targets across a range of scales by integrating feature maps from different depths with RPNs tailored to various scales. Some researchers have also adopted multi-scale feature fusion methods to solve the scale problem. This technique is aimed at fusing deep features with shallow features to obtain sufficient semantic information. The Feature Pyramid Network (FPN) [31] achieves multi-scale fusion by upsampling features and adding them to bottom-level features, enabling the combination of feature maps with strong low-resolution semantic information and those with rich high-resolution spatial information at minimal computational cost. Building upon FPN, the Path Aggregation Network (PANet) [32] incorporates bottom–up path enhancement to leverage precise localization information for enhancing the entire feature set. The Bidirectional Feature Pyramid Network (BiFPN) [33] proposes a more efficient bidirectional feature fusion to mitigate the issues of information loss and redundancy in the traditional feature pyramid networks when extracting features of different scales.

Additionally, detection tasks in traffic scenes are significantly constrained by the following two challenges:

a. The scene is intricate and constantly changing, with small targets often existing alongside surrounding objects. When combined with the inherent characteristics of small targets, this results in inadequate extraction of feature information from these targets by the model.

b. Targets from different categories display significant variations in size, and even targets within the same category may differ in size due to their positional differences.

To tackle the challenges mentioned above, inspired by MFDS-DETR [34], we designed a Deep and Filtered Path Aggregation Network (DF-PAN) to better suit detection tasks in driving scenes. First, the top-level features filter the underlying information through weights generated by the Feature Coordinate Filtering Module (FCF), filtering out redundant semantic information to highlight small targets in lower layers; second, the bottom features filter the top features through the weights generated by FCF, filtering out redundant positioning information to highlight the semantic information of the target. DF-PAN aims to enhance the model's ability to fuse multi-scale features, thereby improving detection efficiency.

3 Method

3.1 The overall architecture of the improved algorithm

The DF-YOLO architecture consists primarily of three parts: a backbone network, a deep and filtered Path Aggregation Network, and a parameter-sharing detection head. In the backbone extraction network, we replace the original structure with FasterNet, a rapid neural network, to reduce redundant computations and improve spatial feature extraction. To tackle the multi-scale challenges arising from variations in object sizes, we propose the Deep Filtering Path Aggregation Network (DF-PAN), which effectively fuses deep features with shallow features. In addition, we design a parameter-sharing detection head (PSD), which reduces the number of parameters and improves detection precision. The overall structure of the algorithm is shown in Fig. 1.

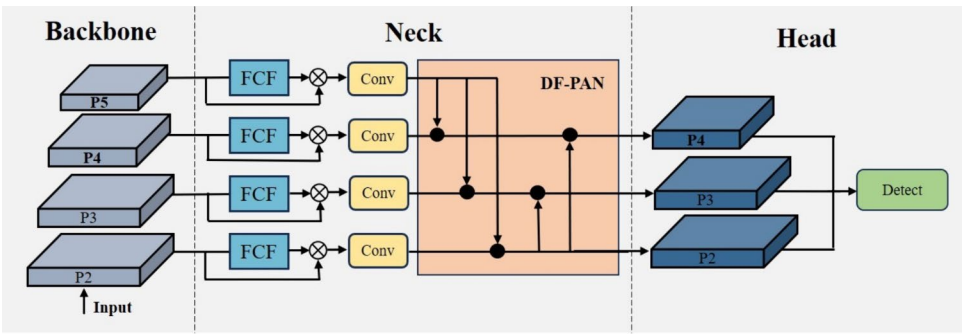

Fig. 1 The overall structure of DF-YOLO

3.2 Backbone

The backbone network of YOLOv8 is overly large. Utilizing FasterNet as the backbone network for feature extraction in DF-YOLO aligns more closely with practical requirements. FasterNet addresses the issue of redundancy in the convolutional neural networks by introducing Partial Convolution (PConv). This technique enhances spatial feature extraction while reducing unnecessary computations and memory access. The overall structure is shown in Fig. 2.

The proposed architecture consists of four hierarchical stages, each featuring a 4×4 regular convolutional layer with a stride of 4 for spatial downsampling and a 2×2 regular convolutional layer with a stride of 2 for expanding the channel number. Each stage has a FasterNet block, which contains a PConv layer. Unlike conventional convolution, the PConv layer selectively conducts convolution operations on specific channels to extract spatial features, while leaving the remaining channels unchanged. In the detection task of this paper, the input and output feature maps have the same number of channels. Therefore, the FLOPs of PConv are: $h \times \omega \times k^2 \times c_p^2$, and for $r = 1/4$, the FLOPs of PConv are only $1/16$ of standard convolution, with memory access being $1/4$ of standard convolution. To more effectively utilize information from all channels, pointwise convolution (PWConv) has been added after PConv. PWConv enables feature information to flow through all channels, thereby allowing the model to focus more on the central position.

Due to the small size of targets which only occupy a small portion of the image, models often overlook the spatial information of these targets. The FasterNet feature extraction network excels in spatial feature extraction, enhancing the model's ability to detect small targets. Additionally, FasterNet requires less memory access and is better suited for traffic scenes with limited computing resources.

3.3 DF-PAN

In complex traffic scenes, significant size variations exist among targets across different categories. Moreover, targets within the same category may also vary in size due to their distinct spatial locations. This inherent multi-scale diversity negatively impacts the model's detection and recognition capabilities. Feature Pyramid Networks (FPN) fuse extracted multi-scale information in a bottom–up manner, thereby enhancing the precision of models. However, this structure is less effective in transmitting location information. To address this limitation, the PANet introduces a top–down pathway, facilitating the transfer of low-level details to the upper layers, thereby obtaining a feature map with more comprehensive semantic and spatial information, consequently enhancing the feature representation capability. To better tackle the inherent multi-scale issue in driving scenes, we design a Deeper and Filtered Path Aggregation Network (DF-PAN). The structure is shown in Fig. 3 and consists of two modules:

(a) Feature Coordinate Filtering module (FCF)
(b) Deep Feature Fusion module (DF).

(a) Feature coordinate filtering module (FCF).

The feature coordinate filtering module extracts the importance of the feature map in different dimensions, thereby selectively filtering unimportant features. This process enhances the semantic content of the generated features, ultimately improving the model's ability to detect targets of various scales. The original channel-wise attention module only considers dependencies

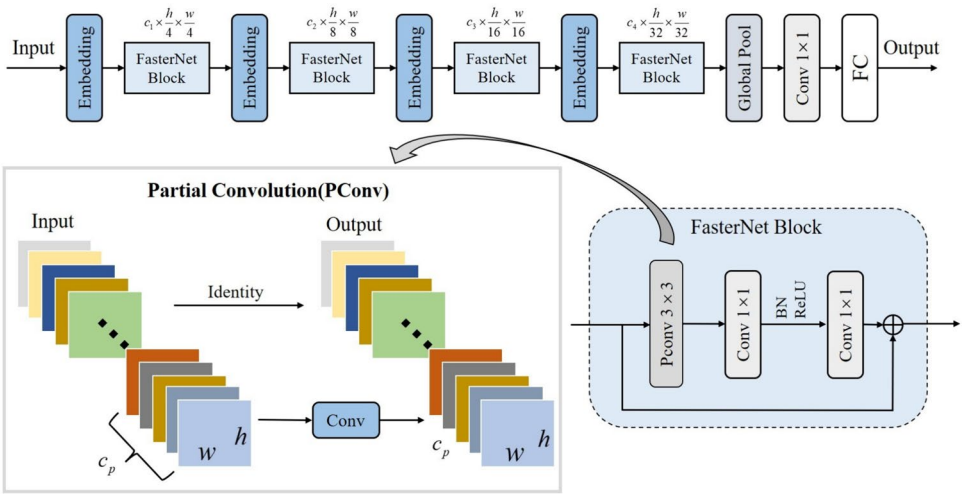

Fig. 2 The overall structure of FasterNet

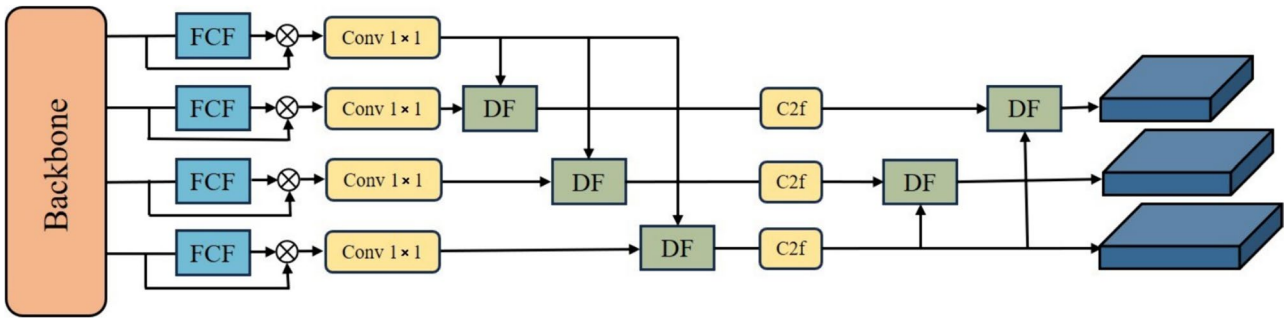

Fig. 3 The overall structure of DF-PAN

between channels, neglecting the importance of positional information in capturing target structures. To tackle this issue, FCF breaks down global pooling into two parallel feature encoding processes. This approach allows for the capture of inter-feature dependencies while also preserving accurate positional information. Consequently, it significantly enhances the model's ability to objects of different scales. As illustrated in Fig. 4, the feature coordinate filtering module processes the input feature map, where C denotes the number of channels, H represents the height of the feature map, and W represents its width. For a given input X, we encode each channel separately along the horizontal and vertical coordinates using pooling kernels with two spatial ranges: (H, 1) or (1, W). Then, we combine these encodings to obtain features and employ the sigmoid activation function to determine the weights for the horizontal and vertical coordinates. Next, multiply the obtained weights with the feature maps after global average pooling. Finally, multiply them with the corresponding proportion of feature maps to obtain the output. This transformation allows the module to capture long-range dependencies along one spatial direction while preserving positional information along another. The purpose of average pooling is to evenly obtain all data from the feature map and minimize information loss, which is particularly crucial for small target detection in traffic scenes.

(b) Deep feature fusion module (DF).

In the feature map extracted by the feature extraction network, the deep features have rich semantic information, but relatively little location information; the shallow features have accurate target positioning, but limited semantic information. In this regard, we propose a deep feature fusion module (Fig. 5), the deep features filter the underlying information through weights generated by FCF, filtering out redundant semantic information to highlight small targets in lower layers; The shallow features filter the deep features through the weights generated by FCF, filtering out redundant positioning information to highlight the semantic information of the target. Establishing connections between feature maps at different levels for more effective feature fusion, enabling the network to detect objects of various scales.

Moreover, smaller objects occupying fewer pixels often lead to frequent occurrences of missed and false detections during the detection process. To address this issue, we

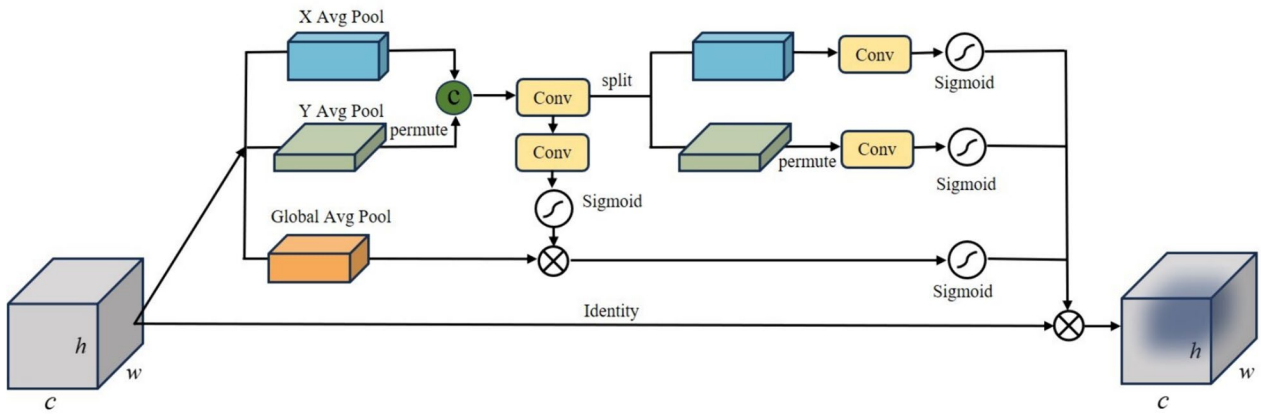

Fig. 4 The overall structure of FCF

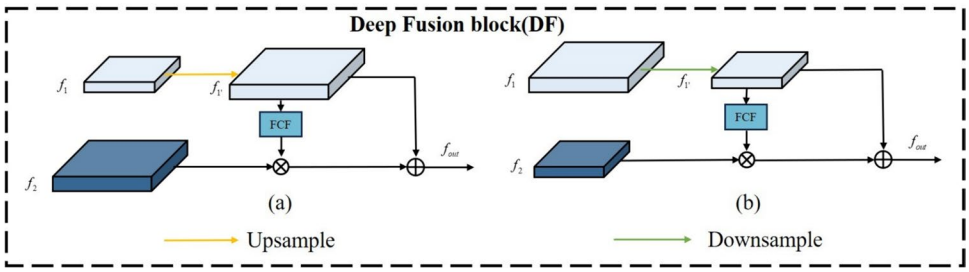

Fig. 5 Deep Fusion block structure diagram. Given two features f_1 and f_2, to achieve uniform dimensions, upsampling and downsampling are performed on the deep and shallow features respectively. Subsequently, weights are generated through the Feature Coordinate Filtering (FCF) to filter the corresponding features. Finally, the filtered features are fused together, and the output is expressed as: $f_{out} = f_2 \times FCF(f_{1'}) + f_{1'}$

incorporate deeper upsampling in the feature fusion module to enable more profound fusion.

3.4 Parameter sharing detection

The detection head of YOLOv8-n accounts for 40% of the computational workload in the model. To make the algorithm better match the autonomous driving scenario, we designed a lightweight detection head called Parameter Sharing Detection (PSD) (Fig. 6), which reduces computational effort by sharing parameters. The PSD module takes feature maps from different scales obtained from DF-PAN as input to the shared convolution module. Moreover, we introduce a learnable scaling factor to adjust the feature maps after Conv_share, resulting in three distinct outputs from the detection layers. This parameter-sharing approach not only reduces the number of parameters in the model but also enables feature extraction capabilities to be shared across different locations, thereby enhancing both the model's efficiency and generalization ability. Experimental results demonstrate that PSD effectively reduces computational workload while maintaining high precision.

4 Experiments and analysis

4.1 Datasets

We conducted experiments on the KITTI dataset [35], BDD100K dataset [36], and SODA 10 M dataset [37] to comprehensively evaluate the proposed method.

The KITTI dataset is one of the most commonly used benchmark datasets internationally for evaluating detection algorithms in autonomous driving scenarios. It comprises real image data collected from various environments like urban, rural, and highway scenes, featuring up to 15 cars and 30 pedestrians per image, along with varying degrees of occlusion and truncation, posing a significant challenge in the field of object detection. The dataset is annotated with nine

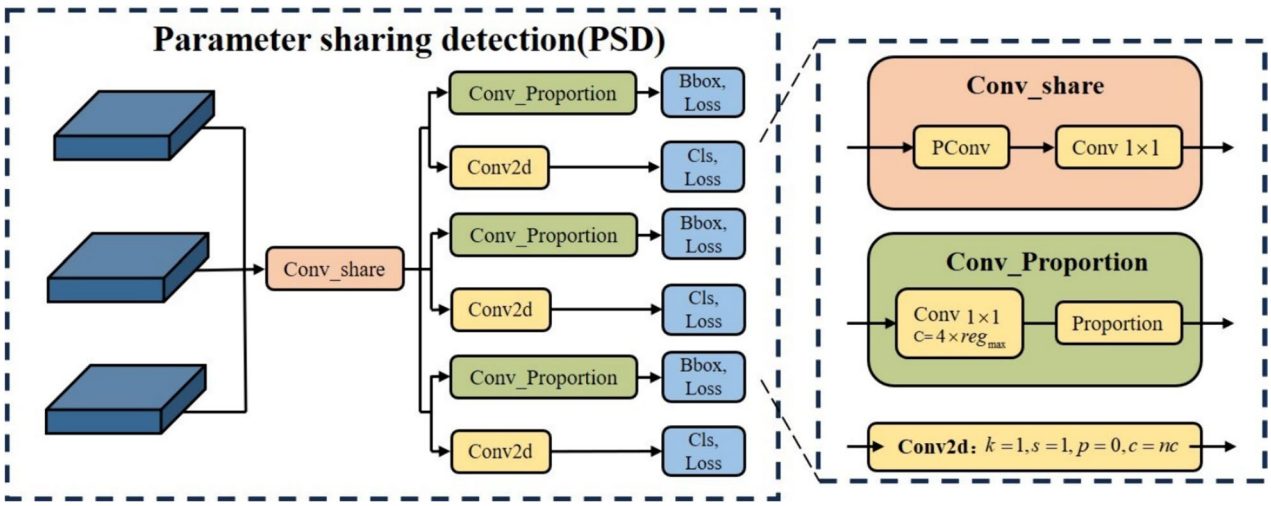

Fig. 6 Overall structure of PSD

primary scene categories (Car, Truck, Van, Tram, Pedestrians, Person_sitting, Cyclist, Misc).

The BDD100K dataset is a publicly available driving dataset released by the University of California, Berkeley. It comprises 100,000 annotated frames collected under various weather conditions (sunny, cloudy, overcast, rainy, snowy, foggy), different times of the day (daytime, nighttime, dawn/dusk), and diverse scenes (residential areas, urban streets, highways, etc.). Due to its diverse geographical, environmental, and meteorological conditions, the BDD100K dataset is an excellent choice for evaluating network reliability.

The SODA10M dataset covers a variety of different road scenes, taking into account diverse weather conditions and conducting data collection during various periods, including daytime, nighttime, early morning, and dusk. The dataset is annotated with six primary scene categories (Pedestrian, Cyclist, Car, Truck, Tram, Tricycle). The SODA10M dataset presents a variety of environmental conditions and can approximate the diversity of real driving environments.

4.2 Training equipment and parameters set

Throughout the training process, we optimize training parameters using stochastic gradient descent. We set the momentum at 0.937 and the learning rate during training to 0.01. We set the batch size as 64, and the epoch is 300. The size of the image in the dataset is 640×640 pixels (as shown in Table 1).

The loss value reflects the convergence state of the model during the training process. The loss functions used in this paper include classification loss function and bounding box loss function. The classification loss function includes binary cross-entropy loss, denoted as L_{BCE}. The bounding box loss function includes distribution focal loss (DFL), denoted as L_{DFL} and complete IoU (CIoU) loss, represented as L_{CIoU}. Thus, the loss L_{total} can be represented as

$$L_{total} = \lambda_{DFL}L_{DFL} + \lambda_{CIoU}L_{CIoU} + \lambda_{BCE}L_{BCE}, \tag{1}$$

where λ_{BCE}, λ_{DFL}, and λ_{CIoU} are corresponding coefficients

$$L_{BCE} = -\left[y_n \log x_n + (1 - y_n)\log(1 - x_n)\right], \tag{2}$$

Table 1 Training environment and hardware platform parameters table

Parameters	Configuration
Operational platform	Ubuntu 18.04
Compilers	Python 3.9
Network construction method	Pytorch 2.0
CPU	Intel(R) Xeon(R) Gold 6348
GPU	NVIDIAA30 24 GB

where x_n is the predicted classification of each object. y_n is. the ground truth of each object

$$\begin{aligned} L_{DFL}(S_i, S_{i+1}) &= -((y_{i+1} - y)\log(S_i) + (y - y_i)\log(S_{i+1})) \\ S_i &= \frac{y_{i+1} - y}{y_{i+1} - y_i}, S_{i+1} = \frac{y_i - y}{y_i - y_{i+1}}, \end{aligned} \tag{3}$$

where: y is the ground truth of the bounding box coordinate.

$$\begin{aligned} L_{CIoU} &= 1 - IoU + \frac{\rho^2(b, b^{gt})}{c^2} + \alpha v \\ v &= \frac{4}{\pi^2}(\arctan\frac{w^{gt}}{h^{gt}} - \arctan\frac{w}{h})^2 \\ \alpha &= \frac{v}{(1 - IoU) + v}, \end{aligned} \tag{4}$$

where b is the central point of the prediction box, b^{gt} is the central point of the ground truth box. ρ is the Euclidean distance between prediction and ground truth points. c is the diagonal length of the smallest enclosing rectangle of the two boxes. v and α are ratio coefficients. w^{gt} and h^{gt} are the width and height of the ground truth box, and w and h are the width and height of the prediction box.

4.3 Evaluation metrics

To assess the effectiveness of the model, we utilize mAP, the number of Parameters, Floating Point Operations (FLOPs), and Frames Per Second (FPS) as evaluation metrics. The number of parameters is used to describe the space complexity of the network, which reflects the small space occupied by the model. FLOPs and FPS are used to describe the time complexity of the network. FLOPs represent the number of floating-point arithmetic operations performed by the network during inference, providing insight into the computational workload. On the other hand, FPS indicates the rate at which the network processes frames or inputs per second, reflecting its real-time performance. mAP is primarily employed for evaluating Average Precision (AP), Precision (P), and Recall (R) in object detection tasks. Equations (5) (6), respectively, represent Precision (P) and Recall (R).

$$P = \frac{TP}{TP + EP} \tag{5}$$

$$R = \frac{TP}{TP + FN} \tag{6}$$

where: TP denotes true positives, FP denotes false positives, and FN denotes false negatives.

AP denotes the area under the precision–recall curve, which provides a comprehensive evaluation of model accuracy by considering both precision and recall for

each class. A higher precision indicates superior model performance. Equation (7) can be used to express AP, whereas mAP quantifies the average AP across all classes and is represented by Eq. (8)

$$AP = \int_0^1 P(R)dr \tag{7}$$

$$mAP = \frac{\sum_{i=1}^{N} AP_i}{N} \tag{8}$$

To provide a more detailed description of the model's performance, we use TIDE [38] evaluation indicators and categorize errors in the model into six types: Classification error (Cls), Localization error (Loc), Classification and Localization error (Cls + Loc), Duplicate detection error (Duplicate), Background error (Bkgd), and Missed detection error (Miss). We use IoU_{max} to represent the maximum IoU overlap between the predicted bounding box and the Ground Truth of the given class. The foreground IoU threshold is denoted as t_f, and the background threshold is denoted as t_b, which are set to 0.5 and 0.1 respectively. Details are shown in Fig. 7.

4.4 Performance comparison between DF-YOLO and YOLOv8

We validate our model using the KITTI dataset. To assess its performance, we compare the experimental results of DF-YOLO and YOLOv8 with those of YOLOv8-n as the baseline (as shown in Table 2). From the experimental results, it can be observed that compared to YOLOv8-n, DF-YOLO demonstrates a significant performance improvement. The mAP increases by 3%, while the number of parameters decreases by 28.1%. Moreover, the localization error decreases by 1.65%, and the missed detection error decreases by 1.1%. When the batch size is set to 1, the model achieves an FPS rate of 77, meeting the real-time requirements. The experiments confirm that our proposed model possesses superior object localization capabilities and enhances detection capabilities for small objects. To better illustrate the superiority of our proposed model, we conduct a visual analysis of the detection results of DF-YOLO on the KITTI dataset compared to the original YOLOv8-n model (as shown in Fig. 8). According to the figures shown, the left column displays the detection results of the original model. As shown in the first and third images, the model's performance in detecting heavily occluded objects is very poor. In addition, as shown in the second picture, the model cannot accurately detect long-distance small targets, resulting in missed detections and false detections. In contrast, the right column presents the detection results of DF-YOLO. Our proposed model effectively reduces the likelihood of missed and false detections, demonstrating strong detection capabilities even for occluded and distant small objects. In the third figure, although our method reduces the probability of missing and false detections for occluded and small objects, there are still occurrences of missed detections and false positives. The reason for this is that, due to the distance of the targets and occlusion issues, the detector is constrained by insufficient resolution, making it difficult to distinguish the boundaries and features of each

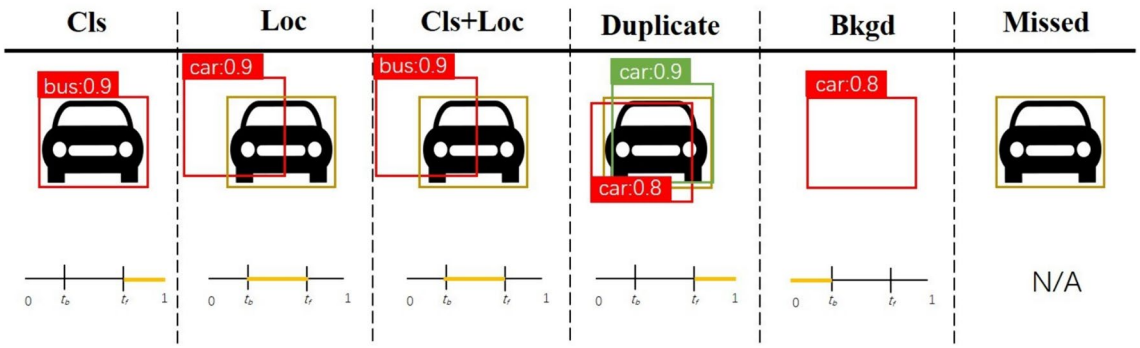

Fig. 7 Error type definitions. Red boxes represent false positives, green boxes represent ground truth; and orange boxes represent true positives. The IoU with ground truth for each error type is indicated by an orange highlight and shown in the bottom row

Table 2 Performance comparison between DF-YOLO(n) and YOLOv8-n (%)

Method	mAP50	Params (M)	FPS(bs = 1)	E_{cls}	E_{loc}	E_{both}	E_{Dup}	E_{bkgd}	E_{miss}
YOLOV8-n	87.9	3.2	102	0.22	4.29	0.26	1.26	1.43	1.80
DF-YOLO	90.9	2.3	77	0.26	2.64	0.25	0.95	1.23	0.70
Improve	3	0.9	–	–	1.65	0.01	0.31	0.2	1.1

Fig. 8 Visualization of detection results of YOLOv8-n detector (left) and DF-YOLO detector (right) based on the KITTI dataset. Green, blue, and red boxes represent true positives (TP), false positives (FP), and false negatives (FN), respectively. Yellow boxes indicate zoomed-in views

target, resulting in missed detections and false positives. Although our method may encounter issues with missed detections in certain scenarios, considering the diversity of real-world application needs and the comprehensive performance of the overall system, the impact of such situations on practical applications is relatively minor. The visualized results further confirm that DF-YOLO is better suited for complex autonomous driving scenes.

4.5 Ablation study

The effectiveness of different modules is validated through extensive ablation experiments. Among them, Baseline refers to YOLOv8-n, and √ indicates the inclusion of the corresponding module in this experiment. From the experimental results (Table 3), it can be observed that compared to YOLOv8-n, the addition of FasterNet, DF-PAN, and PSD modules results in an increase in mAP by 1.4%, 1.5%, and 0.5% respectively. Notably, DF-PAN exhibits the most significant improvement in model performance. Error analysis suggests that the FasterNet module reduces both classification and localization errors as well as background errors. The DF-PAN module significantly decreases missed detections, while the PSD module reduces localization errors probability-wise. When combined together with FasterNet, DF-PAN and PSD modules lead to a respective mAP improvement of 2.1% and 1.8%. Consequently, when combined with these modules, mAP reaches 90.9%, representing a remarkable 3% enhancement over YOLOv8-n's performance level. Furthermore, the number of parameters is reduced by an impressive 28.1% compared to YOLOv8-n while achieving an FPS rate of 77. However, during the process of improving accuracy, we observe an increase in FLOPs and a slight decrease in FPS. We believe that sacrificing some speed to achieve higher accuracy is reasonable in the detection task addressed in this paper.

Table 3 Ablation experiments of different modules (%)

	FasterNet	DF-PAN	PSD	mAP50	Params (M)	FLOPs(G)	FPS(bs = 1)	E_{cls}	E_{loc}	E_{both}	E_{Dup}	E_{bkg}	E_{miss}
Baseline				87.9	3.2	8.2	102.7	0.22	4.29	0.26	1.26	1.43	1.80
	√			89.3	4.4	10.7	86.3	0.25	4.37	0.14	1.27	1.14	2.20
		√		89.4	2.6	14.9	70.2	0.44	3.33	0.30	0.83	1.22	0.81
			√	88.4	2.5	6.7	112.7	0.29	3.14	0.19	1.98	1.35	2.09
	√	√		90	3.3	16.0	70.5	0.34	3.23	0.29	0.85	1.15	0.75
		√	√	89.7	2.2	11.7	82.8	0.41	3.30	0.29	1.12	1.33	0.76
	√	√	√	90.9	2.3	12.2	77.1	0.26	2.64	0.25	0.95	1.23	0.70

The above experiments fully demonstrate not only the effectiveness but also the suitability of these modules for deployment on resource-constrained mobile platforms.

4.6 Compare with mainstream methods

This section compares DF-YOLO with models, such as YOLOv5, YOLOP [39], YOLOPv2 [40], A-YOLOM [41], CF-YOLOX [42], Faster R-CNN, DINO-Deformable-DETR [43], and HybridNets [44] on the KITTI dataset, The experimental results are shown in Table 4. Among them, bs = 1 represents the FPS when the batch size is 1. The difference between DF-YOLO(n) and DF-YOLO(l) lies in the complexity of the backbone network, where 'n' represents a lightweight network. The design of DF-YOLO(n) reduces the complexity and is suitable for deployment on edge devices with limited computing resources. DF-YOLO(l) provides higher precision while increasing computational overhead. According to experimental results, DF-YOLO(n) achieves the most efficient results. Compared with YOLOP, A-YOLOM, and HybridNets, DF-YOLO(n) has fewer parameters, higher precision, and higher FPS. Compared to yolopv2, DINO-Deformable-DETR, and CF-YOLOX, DF-YOLO(n) has significant advantages in terms of model parameters and speed. With similar parameters and computation costs, our framework outperforms other object detectors, achieving a balance between real-time performance and accuracy.

4.7 Performance on the SODA 10 M dataset

This section compares DF-YOLO with models such as YOLOv5, YOLOP, YOLOPv2, A-YOLOM, TTD-YOLO [45], Faster R-CNN, DINO-Deformable-DETR, and HybridNets on the SODA10M dataset. The experimental results are shown in Table 5, that the DF-YOLO algorithm outperforms the other algorithms, has higher detection accuracy, and can meet real-time detection. To better illustrate the reliability of the proposed framework, we conduct a visual analysis (as shown in Fig. 9) of the model's detection results under various environmental conditions on the SODA10M dataset. Under three different environmental conditions (a), (b), and (c), the original model detected 6, 6, and 0 targets, respectively, while DF-YOLO detected 10, 9, and 3 targets, respectively. The visualization results show that the proposed framework has good generalization ability in different environments.

Table 4 Comparison of detection results of different algorithms on the KITTI dataset (%)

Method	mAP50	Recall	Params (M)	FPS(bs = 1)
YOLOv8-n (baseline)	87.9	79.9	3.2	**102**
Faster R-CNN	79.9	79.1	41.3	8.8
YOLOv5-s	88.2	80.7	7.2	53.4
HybridNets	87.7	86.5	12.8	15.2
A-YOLOM	89.3	83.5	4.43	39.9
YOLOP	86.3	81.3	7.9	26.5
YOLOPv2	92.5	84.9	38.9	48
DINO-Deformable-DETR	93.5	–	47	5
CF-YOLOX	93.07	–	9.5	33.9
DF-YOLO(n)	90.9	82.6	**2.3**	77
DF-YOLO(l)	**93.9**	**87.9**	31.4	39

Table 5 Comparison of detection results of different algorithms on the SODA10M dataset (%)

Method	mAP50	Recall	FPS
YOLOv8-n(Baseline)	42.6	50.2	**100.7**
Faster R-CNN	30.1	39.3	11.8
YOLOv5-s	44.5	52.7	52.8
HybridNets	43.1	55.8	13.4
A-YOLOM	45.7	51.4	39.7
YOLOP	44.2	56.8	25.4
YOLOPv2	46.6	61.2	47.6
DINO-Deformable-DETR	55.4	–	4.8
TTD-YOLO	46.5	62	55.9
DF-YOLO(n)	47.3	63.1	76.5
DF-YOLO(l)	**59.1**	**67.7**	38.2

4.8 Performance on the BDD100K dataset

This section compares DF-YOLO with models such as YOLOv5, YOLOP, YOLOPv2, A-YOLOM, CF-YOLOX, MCS_YOLO [46], Faster R-CNN, DINO-Deformable-DETR, and HybridNets on the BDD100K dataset. The experimental results are shown in Table 6, that the DF-YOLO algorithm outperforms the other algorithms, has higher detection accuracy, and can meet real-time detection.

5 Conclusion

High-precision and real-time performance are essential requirements for detection algorithms in autonomous driving systems. This paper proposes a DF-YOLO algorithm that can balance real-time performance and accuracy in complex scenes with limited computational resources, addressing the problem of poor accuracy caused by large differences in target scales in traffic scenarios. High-precision real-time detection results can aid autonomous vehicles in quickly identifying and predicting potentially dangerous situations, thereby making timely decisions and plans, reducing the risk

Fig. 9 Comparison of detection results of YOLOv8-n detector (left) and DF-YOLO detector (right) in different traffic scenarios

Table 6 Comparison of detection results of different algorithms on the BDD100K dataset (%)

Method	mAP50	Recall	FPS
YOLOv8-n(Baseline)	75.1	82.2	**103.5**
Faster R-CNN	64.9	81.2	9.4
YOLOv5-s	76.8	86.8	82
HybridNets	77.1	91.3	11.7
A-YOLOM	78.0	85.3	39.7
YOLOP	76.5	88.6	27
YOLOPv2	83.4	91.1	43.2
DINO-Deformable-DETR	82.7	–	5.6
CF-YOLOX	74.7	–	32.8
MCS-YOLO	80.8	–	55
DF-YOLO(n)	78.7	83.7	75.8
DF-YOLO(l)	**83.9**	**91.7**	42.1

of accidents, and improving safety. DF-YOLO comprises the FasterNet feature extraction network, DF-PAN, and PSD. FasterNet can extract multi-scale feature information more efficiently; DF-PAN can effectively fuse both shallow and deep features, significantly improving the model's detection capabilities for targets with large-scale differences; PSD enables the model to share feature extraction capabilities at different locations, thereby reducing the number of model parameters while enhancing the efficiency and generalization ability of the model. We conducted extensive ablation experiments on the proposed modules, and the results demonstrate that all three modules contribute to improving accuracy. After rigorous ablation experiments and comparative analysis, it was found that the mAP on the KITTI data set reached 90.9%, representing a 3% improvement over the baseline, significantly reducing the probability of missed and false detections. Additionally, the number of parameters decreased by 28.1%. Experimental results prove the effectiveness of DF-YOLO. Overall, this study demonstrates the potential application of the model in resource-constrained environments, assisting with embedded applications. However, exploration in this field remains challenging, and future work will focus on optimizing the model structure to improve algorithm efficiency and precision.

Author contributions HL designed the methodology and drafted the initial manuscript; WS was responsible for project management; ZL and DZ were responsible for conducting the review; XD investigated the research background; GL and ML designed experiments and performed analysis.

Funding This work is partially supported by the National Natural Science Foundation of China (No. 91948303) and Air Force Medical University Cross-integration Project (XJ2023000201).

Data availability The data supporting the findings of this study are publicly available datasets that can be accessed on the official website.

Declarations

Conflict of interest All authors declare that they have no conflicts of interest affecting the work reported in this article.

References

1. Viola, P., Jones, M.J.: Robust real-time face detection. Int. J. Comput. Vision **57**, 137–154 (2004)
2. Dalal N, Triggs B. Histograms of Oriented Gradients for Human Detection. IEEE Computer Society Conference on Computer Vision & Pattern Recognition2005.
3. Hearst, M.A., Dumais, S.T., Osuna, E., Platt, J., Scholkopf, B.: Support vector machines. IEEE Intell. Syst. Their Appl. **13**(4), 18–28 (1998)
4. Viola PA, Jones MJ. Rapid object detection using a boosted cascade of simple features. computer vision and pattern recognition, 2001 CVPR 2001 Proceedings of the 2001 IEEE Computer Society Conference on2001.
5. Redmon J, Divvala S, Girshick R, Farhadi A. You only look once: unified, real-time object detection. computer vision & pattern recognition 2016.
6. Redmon J, Farhadi A. YOLO9000: Better, Faster, Stronger. IEEE Conference on Computer Vision & Pattern Recognition 2017. p. 6517–25.
7. Redmon J, Farhadi A. YOLOv3: An Incremental Improvement. arXiv e-prints. 2018.
8. Bochkovskiy A, Wang CY, Liao HYM. YOLOv4: Optimal Speed and Accuracy of Object Detection. 2020.
9. Ge Z, Liu S, Wang F, Li Z, Sun J. YOLOX: Exceeding YOLO Series in 2021. 2021.
10. Wang CY, Bochkovskiy A, Liao HYM. YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. arXiv e-prints. 2022.
11. Liu W, Anguelov D, Erhan D, Szegedy C, Reed S, Fu C-Y, et al. Ssd: Single shot multibox detector. Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14: Springer; 2016. p. 21–37.
12. Girshick R. Fast r-cnn. Proceedings of the IEEE international conference on computer vision2015. p. 1440–8.
13. Ren, S., He, K., Girshick, R., Sun, J.: Faster R-CNN: towards real-time object detection with region proposal networks. IEEE Trans. Pattern Anal. Mach. Intell. **39**(6), 1137–1149 (2017)
14. Guo, A., Sun, K., Zhang, Z.: A lightweight YOLOv8 integrating FasterNet for real-time underwater object detection. J. Real-Time Image Proc. **21**(2), 49 (2024). https://doi.org/10.1007/s11554-024-01431-x
15. Li, X., Li, X., Shen, Z., Qian, G.: Driver fatigue detection based on improved YOLOv7. J. Real-Time Image Proc. **21**(3), 75 (2024). https://doi.org/10.1007/s11554-024-01455-3
16. Wang, H., Qian, H., Feng, S.: GAN-STD: small target detection based on generative adversarial network. J. Real-Time Image Proc. **21**(3), 65 (2024). https://doi.org/10.1007/s11554-024-01446-4
17. Di Y, Li R, Tian H, Guo J, Shi B, Wang Z, et al. A maneuvering target tracking based on fastIMM-extended Viterbi algorithm. Neural Computing and Applications. 2023:1–10.
18. Chen J, Kao S-h, He H, Zhuo W, Wen S, Lee C-H, et al. Run, Don't walk: Chasing higher FLOPS for faster neural networks. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition2023. p. 12021–31.
19. Chen, J., Wang, Q., Cheng, H.H., Peng, W., Xu, W.: A review of vision-based traffic semantic understanding in ITSs. IEEE Trans. Intell. Transp. Syst. **23**(11), 19954–19979 (2022)
20. Tang Y, He H, Wang Y, Mao Z, Wang H. Multi-modality 3D object detection in autonomous driving: A review. Neurocomputing. 2023:126587.
21. Wang, Z., Zhan, J., Duan, C., Guan, X., Lu, P., Yang, K.: A review of vehicle detection techniques for intelligent vehicles. IEEE Trans. Neural Netw. Learn. Syst. **34**(8), 3811–3831 (2022)
22. Hu, J., Sun, Y., Xiong, S.: Research on the cascade vehicle detection method based on CNN. Electronics **10**(4), 481 (2021)
23. Ghosh, R.: On-road vehicle detection in varying weather conditions using faster R-CNN with several region proposal networks. Multimedia Tools Appl. **80**(17), 25985–25999 (2021)
24. Han, X.: Modified cascade R-CNN based on contextual information for vehicle detection. Sens. Imaging. **22**(1), 19 (2021)
25. Oreski, G.: YOLO* C—adding context improves YOLO performance. Neurocomputing **555**, 126655 (2023)
26. Kang, L., Lu, Z., Meng, L., Gao, Z.: YOLO-FA: Type-1 fuzzy attention based YOLO detector for vehicle detection. Expert Syst. Appl. **237**, 121209 (2024)
27. Li, S., Chen, J., Peng, W., Shi, X., Bu, W.: A vehicle detection method based on disparity segmentation. Multimedia Tools Appl. **82**(13), 19643–19655 (2023)
28. Yuan, Z., Wang, Z., Zhang, R.: CCBA-NMS-YD: A vehicle pedestrian detection and tracking method based on improved YOLOv7 and DeepSort. World Electric Vehicle J. **15**(7), 309 (2024)
29. Khan SD, Ullah H, Ullah M, Conci N, Cheikh FA, Beghdadi A. Person head detection based deep model for people counting in sports videos. 2019 16th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS): IEEE; 2019. p. 1–8.
30. Khan, S.D., Basalamah, S.: Scale and density invariant head detection deep model for crowd counting in pedestrian crowds. Vis. Comput. **37**(8), 2127–2137 (2021)
31. Lin T-Y, Dollár P, Girshick R, He K, Hariharan B, Belongie S. Feature pyramid networks for object detection. Proceedings of the IEEE conference on computer vision and pattern recognition2017. p. 2117–25.
32. Liu S, Qi L, Qin H, Shi J, Jia J. Path aggregation network for instance segmentation. Proceedings of the IEEE conference on computer vision and pattern recognition2018. p. 8759–68.
33. Tan M, Pang R, Le QV. Efficientdet: Scalable and efficient object detection. Proceedings of the IEEE/CVF conference on computer vision and pattern recognition2020. p. 10781–90.
34. Chen, Y., Zhang, C., Chen, B., Huang, Y., Sun, Y., Wang, C., et al.: Accurate leukocyte detection based on deformable-DETR and multi-level feature fusion for aiding diagnosis of blood diseases. Comput. Biol. Med. **170**, 107917 (2024)
35. Geiger A, Lenz P, Urtasun R. Are we ready for autonomous driving? the kitti vision benchmark suite. 2012 IEEE conference

on computer vision and pattern recognition: IEEE; 2012. p. 3354–61.
36. Yu F, Chen H, Wang X, Xian W, Chen Y, Liu F, et al. Bdd100k: A diverse driving dataset for heterogeneous multitask learning. Proceedings of the IEEE/CVF conference on computer vision and pattern recognition2020. p. 2636–45.
37. Han J, Liang X, Xu H, Chen K, Hong L, Mao J, et al. SODA10M: A large-scale 2D self/semi-supervised object detection dataset for autonomous driving. arXiv preprint arXiv:11118. 2021.
38. Bolya D, Foley S, Hays J, Hoffman J. Tide: A general toolbox for identifying object detection errors. Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part III 16: Springer; 2020. p. 558–73.
39. Wu, D., Liao, M.-W., Zhang, W.-T., Wang, X.-G., Bai, X., Cheng, W.-Q., et al.: YOLOP: you only look once for panoptic driving perception. Mach. Intell. Res. **19**(6), 550–562 (2022). https://doi.org/10.1007/s11633-022-1339-y
40. Han C, Zhao Q, Zhang S, Chen Y, Zhang Z, Yuan J. Yolopv2: Better, faster, stronger for panoptic driving perception. arXiv preprint arXiv:11434. 2022.
41. Wang J, Wu Q, Zhang N. You Only Look at Once for Real-time and Generic Multi-Task. arXiv preprint arXiv:01641. 2023.
42. Wu, S., Yan, Y., Wang, W.: CF-YOLOX: An autonomous driving detection model for multi-scale object detection. Sensors. **23**(8), 3794 (2023)
43. Zhang H, Li F, Liu S, Zhang L, Su H, Zhu J, et al. Dino: Detr with improved denoising anchor boxes for end-to-end object detection. arXiv preprint arXiv:03605. 2022.
44. Vu D, Ngo B, Phan H. Hybridnets: End-to-end perception network. arXiv preprint arXiv:09035. 2022.
45. Xia W, Li P, Huang H, Li Q, Yang T, Li Z. TTD-YOLO: A Real-time Traffic Target Detection Algorithm Based on YOLOV5. IEEE Access. 2024.
46. Cao, Y., Li, C., Peng, Y., Ru, H.: MCS-YOLO: A multi-scale object detection method for autonomous driving road environment recognition. J IEEE Access. **11**, 22342–22354 (2023)