Electronic Research Archive

https://www.aimspress.com/journal/era

ERA, 33(9): 5591–5615.
DOI: 10.3934/era.2025249
Received: 04 June 2025
Revised: 19 August 2025
Accepted: 03 September 2025
Published: 17 September 2025

Research article

SDOD: An efficient object detection method for self-driving cars based on hierarchical cross-scale features

Jingwen Qi[1,*] and Jian Wang[1,2]

[1] School of Automation and Intelligence, Beijing Jiaotong University, Beijing, China
[2] Beijing Engineering Research Center of EMC and GNSS Technology for Rail Transportation, Beijing Jiaotong University, Beijing, China

* **Correspondence:** Email: 21111077@bjtu.edu.cn.

Abstract: With the increasing prominence of autonomous vehicles in recent years, rapid and accurate environmental perception has become crucial for the operational safety and decision-making capabilities. To address the challenge of achieving an optimal balance between accuracy and real-time performance under in-vehicle computational constraints, this paper presented an efficient object detection algorithm for self-driving cars, which extracted the hierarchical cross-scale features based on a shifted-window attention mechanism. By integrating this improved feature representation with the more efficient feature fusion neck and detection head based on depth-wise separable convolution, the proposed approach significantly reduced model complexity and improves detection speed while maintaining near-identical detection accuracy. Experimental results demonstrate that this method simultaneously enhances processing speed and reduced model complexity while maintaining high detection precision, with floating-point operations reduced from 21.5 G to 6.0 G, a decrease of 15.5 G and an increase of 139 frames per second compared to YOLOv11s. This combination of efficiency and accuracy made the proposed algorithm particularly adaptable for resource-constraint self-driving systems.

Keywords: object detection; self-driving; hierarchical features; cross-scale features

1. Introduction

The number of vehicles has increased rapidly in recent years, which leads to numerous issues including traffic collisions, traffic jam, and environmental pollution etc. Intelligent vehicles are becoming increasingly appealing since they demonstrate the capabilities like avoiding collision automatically, controlling cruise, and auto-driving functionalities through the use of multiple sensors, control algorithms, and other sophisticated systems [1–3]. They play a crucial role in increasing traffic safety and optimizing driving efficiency [4]. Recently, there is a growing trend of developing self-driving vehicles. The system for self-driving cars primarily comprises three fundamental capabilities: perceiving the environment, automatically making decisions, and controlling the movement. Among these procedures, perceiving the environment serves as the cornerstone and provides necessary condition for achieving self-driving. It requires techniques for monitoring the surrounding environment, detecting objects and obstacles rapidly and accurately, and then transmitting the processed information to the decision-making system for determining the optimal path [5]. Hence, the investigation and application of object detection technology in traffic scenarios hold substantial research importance and practical value. During the initial stage of developing autonomous driving technology, the environment perception was conducted by expensive single-sensor or a fusion of multiple sensors. Besides, technicians need to configure the vehicle settings manually and fine-tune them through multiple simulations and tests in real-world scenarios [6]. This early approach for gathering data from objects in the environment relied heavily on human intervention, making it challenging to be deployed in different scenarios. The emergence of affordable onboard sensors like radars, lidars and cameras [7, 8], and ongoing progress in computation ability as well as breakthroughs in deep learning methodologies have brought significant evolution to the field of autonomous driving [9]. To guarantee the smooth operation of self-driving cars in complicated environments, designing a reliable sensing system with a particular focus on object detection algorithms is necessary. However, considerable challenge still remains in achieving this goal.

Deep learning algorithms play a vital role in various domains. In the transportation sector, they demonstrate remarkable capabilities in critical applications such as identifying and tracking objects in traffic environments and detecting lane boundaries for vehicle navigation systems [10, 11]. For object detection research, deep learning algorithms have become mainstream technology, attracting widespread attention from both academic and industrial communities. Researchers are paying more attention to explore and optimize various deep neural network architectures to bring more precision and efficiency to the object detection. Presently, the most widely used object detection algorithms are classified into two primary branches: two-stage detection methods and one-stage detection methods. The one-stage detectors are exceptionally suitable for detecting objects in real-time, since they predict object positions and categories directly, eliminating the requirement for a separate region proposal phase [12]. On the other hand, two-stage detectors emphasize on the accuracy by utilizing a sequential two-stage schema [13]. This process first generates region proposals and then enhances categorization and localization [14]. The choice between these two detection methods depends on the particular requirements of the application, requiring careful consideration to strike a balance between processing speed and detection precision. It is worth noting that the You Only Look Once (YOLO) family, particularly YOLOv11 developed in 2024, has made significant and rapid progress, attaining leading performance in application of detecting objects [15]. However, despite the excellent performance of mainstream algorithms in general object detection, they still face specific challenges in the context of autonomous driving. For instance, autonomous vehicles must process scenes with a vast dynamic range of object scales. Traditional convolutional neural network (CNN) backbones, with their fixed local receptive fields, struggle to effectively capture features for both very large and very small objects simultaneously. A critical subset of the multi-scale problem is detecting objects that occupy very few pixels, such as distant pedestrians, traffic lights, and road signs. During the down-sampling process in a typical deep network, the sparse feature information from these small objects is often lost, making them invisible to the model. What's more, standard models can be easily interrupted by complex backgrounds, like adverse weather, varying lighting and occlusions.

In general, in the object detection algorithms like YOLO, feature map is compressed by utilizing down-sampling via convolution operations and, hence, the receptive field is broaden. Then, up-sampling is applied to reestablish the dimensions of the image to the original ones, allowing for the identification of objects at various scales. However, the information may get lost in these steps, causing impairments to detecting certain targets. In order to mitigate this information loss, this research proposes a novel object detection structure, which employs a multi-head self-attention mechanism that enables both local and global context modeling in a hierarchical manner. By splitting the input feature maps into non-overlapping windows and shifting these windows to apply attention mechanism within and across the windows, this approach helps maintain and refine critical features throughout the entire down-sampling and up-sampling processes. This approach capitalizes on both local and global information without the high computational cost of simple global attention. This layered, progressive design leads to a reduced floating-point operation count and faster inference speed, which is essential for developing autonomous driving applications where real-time, resource-efficient detection is considerably important. It strikes a balance between efficiency, the extension of the receptive field, and the accurate identification of multi-scale features for self-driving object detection tasks.

The primary contributions of this work can be summarized as follows:

- We propose an object detection framework for self-driving cars in which the YOLOv11 backbone is replaced with a Swin Transformer backbone. The Swin Transformer's shifted window attention mechanism and hierarchical architecture explicitly enable the capture of hierarchical cross-scale features. This design integrates multi-scale contextual information more effectively, facilitating feature fusion across different abstraction levels. Consequently, the model can better handle objects of varying sizes and complex scenes, which is critical in dynamic driving environments.

- The use of patch-based self-attention in the backbone, along with reduced channel dimensions in the neck, significantly lowers computational resource usage while maintaining detection accuracy. This makes it particularly well-suited for autonomous driving systems, which have high demands for both response speed and resource efficiency.

- A detection head based on depth-wise separable convolution (DSDH) is proposed for efficient object prediction, which effectively reduces the computational cost and parameter scale of the model, improves the inference speed, and provides a highly efficient feature processing pipeline that better aligns with the hierarchical feature representation.

The subsequent sections are structured as follows: A brief overview of previous research related to object detection methods and self-driving technologies are provided in Section II. Section III continues with a detailed explanation of our newly developed model's architecture and methodologies. In Section IV, the experimental results obtained from testing on the BDD100K dataset are analyzed. The last section concludes the findings of our experiment and discusses the future research directions.

2. Related work

In line with the growing trend of artificial intelligence and deep learning research, object detection algorithms have been steadily progressing and multiplying. At the initial phase, histogram of oriented gradient (HOG) and support vector machine (SVM) were combined by [16] to detect people and cars, which is prone to false detection in complicated environments and demonstrates poor generalization ability. Then, spatiotemporal deformable part models (SDPM) was introduced by [17] for pose detection, and [18] proposed a fastest DPM for object detection, optimizing the detection speed and accuracy; however, it is still restrained by certain detection scenarios and it is computationally costly.

Subsequently, researchers replaced traditional feature extraction methods with CNNs, with the classic approach being the region-based convolutional neural networks (RCNN) series. In 2014, RCNN was presented by [19], which first segments the picture into small regions and use the selective search algorithm to generate the potential object region candidate. Then, extracting features of selected regions is conducted by CNN, and SVM is applied for determining the object category. In the next year, [20] made some improvements on the feature extraction procedure of RCNN, which utilized CNN to extract feature from the entire picture and then used the region of interest (RoI) pooling layer to obtain features for each candidate region from the feature map. This algorithm is called Fast RCNN, achieving 10 times training and testing speed than RCNN and improving in accuracy. Then, [21] introduced Faster-RCNN, which replaced the selective search algorithm by region proposal network (RPN). The RPN shares convolutional features with the detection network, which not only improves the quality of the candidate regions but also significantly accelerates the speed of region generation. However, as the series of RCNN algorithms require generation of candidate regions before performing classification, although the accuracy has increased, the inference speed is significantly slower, making it difficult to meet real-time requirements. [22] implemented a multi-stage detection framework called cascade RCNN, where each stage uses progressively higher IoU thresholds for more precise object detection. By using the output of each stage to train the next stage and specializing detectors for different quality levels, the problems of both the overfitting and quality mismatch in traditional detectors are solved efficiently, achieving considerable performance improvements.

Transformer architecture was initially introduced and broadly implemented in the natural language processing (NLP) field, where long sequences of data can be modeled by self-attention mechanism, and it allows for transmitting information to extended distance [23]. The object detection field was revolutionized by the Google Lab in 2020, who represented the vision Transformer (ViT) model, a milestone for the computer vision field. This model makes Transformer architecture adaptive for image processing tasks, bridging the gap between NLP and computer version (CV) [24]. In the next year, Microsoft Research addressed the computational limitations of transformers by introducing the Swin Transformer model [25]. This model incorporates two key innovations: it utilizes mechanism similar to the sliding convolutional kernel for feature extraction from the respective field based on hierarchical transformer architecture, and it introduces the conception of "shifted window", making it possible to interact with neighboring windows and, therefore, strengthening the model's ability to better understand the overall context and semantic relationship of the entire image.

The YOLO algorithm family was originally implemented by [26], which has gained significant attention from researchers due to its excellent performance for real-time detection. The object detection was innovatively transferred into a regression problem, using a unified convolutional neural network to enable end-to-end training. However, this initial version of YOLO has certain limitations in handling small objects and localization accuracy, which provides a subsequent improving direction for the following YOLO versions. YOLOv2 incorporated multi-scale training, anchor boxes, and a deeper network architecture, which had a better detection performance than the initial version [27]. Reference [28] proposed a method for detecting tiny vehicles based on improvement on YOLOv2, which provides application value for self-driving systems. YOLOv3 provides further improvement by integrating feature pyramid networks (FPN) and residual connections with the Darknet-53 backbone, a deeper Darknet-53 enhancing its ability to detect objects of varying sizes and boosting model robustness [29]. The Gaussian YOLOv3 algorithm was proposed by [30], which introduces Gaussian modeling and uncertainty prediction for localization. This significantly enhances the object detection accuracy of YOLOv3 in autonomous driving scenarios and reduces false positive rate. YOLOv4 integrates several advanced feature enhancement techniques, such as Mosaic data augmentation, and optimizes the network structure, showing significant improvement in model performance [31]. YOLOv5 is widely adopted by industry for its lightweight design and efficient implementation. In the self-driving area, [32] enhanced the YOLOv5 model to improve its capability in detecting small objects, but the model complexity is also increased. YOLOv8 was developed in 2023, which was extended to a broader range of visual tasks by optimizing head design, loss functions, and with a deeper backbone [33]. The latest version, YOLOv11, was introduced in 2024. It further refines the model architecture, with reduced parameters and achieving simultaneous improvements in detection precision and inference speed [15]. In recent years, researchers also made improvements on the YOLO algorithm and applied them to traffic and industrial scenarios. For example, Tao et al. introduced the CDFF-YOLO network, which integrates cross-dimensional and dual-domain feature fusion to effectively improve the accuracy and speed of small traffic sign detection in complex environments [34]. Tao et al. [35] presented the EFE-YOLO algorithm, which incorporates multi-module feature enhancement and optimization, improving the small object detection performance in industrial scenarios with blur and occlusion significantly. Moreover, Sun et al. [36] developed the PG-DCDAN framework, which leverages a pseudo-label guided dual classifier, emphasizing domain adaptation and cluster boundary optimization. This method provides insights for the potential application in small sample scenarios, but its focus is on fault diagnosis rather than visual object detection.

Although the series of YOLO algorithms demonstrate excellent performance in object detection tasks, their application in resource-constrained environments is limited due to the higher resource consumption that comes with increased model depth and complexity. They utilize traditional convolutional network as backbone for feature extraction, which is highly computationally complex. The large number of parameters makes it challenging to deploy these models efficiently in embedded systems or in autonomous driving scenarios, where real-time processing is crucial. To address this weakness, we effectively optimize the feature extraction process based on the hierarchical window self-attention mechanism, which significantly reduces the overall complexity of the model while maintaining prediction accuracy and computation efficiency.

3. Proposed object detection model

3.1. Data processing

We utilized 640 × 640 color images as input, aiming to balance computational cost while ensuring rich feature representation. Specifically, an input image $J \in \mathbb{R}^{H\times W\times 3}$ is given , where H represents the height, W denotes the width, and the number of channels is 3 (RGB). In this task, both H and W are set to 640. The image is then partitioned into a P * P grid where P = H/S = W/S = 160, with each patch of dimension is S * S * 3, where S is set as four. The patch size was set to S = 4 because it strikes the best balance between accuracy and efficiency. Specifically, it preserves fine details necessary for detecting small objects, aligns naturally with the shallow feature map in YOLO, and avoids the excessive computational overhead that would arise from smaller patch sizes such as S = 2. Moreover, compared with larger patch sizes (e.g., S = 8 or S = 16), which tend to lose small-object information, S = 4 provides an optimal balance between spatial resolution, computational cost, and real-time performance for autonomous driving scenarios. These sub-patches are treated as a sequence of "tokens" with the features of each token formed by concatenating the RGB values of its original pixels. After that step, for each sub-patch, we flatten its three-dimensional structure S * S * 3 into a one-dimensional vector, resulting in the feature dimension of a single token as N = S * S * 3 (here, it equals 64). The flatten sequence $Z \in \mathbb{R}^{M\times N}$, where $M = \frac{H\times W}{S^2}$ (here, it equals 25600) represents the sequence length. A learnable linear transformation $T : Z_k \to Y_k \in \mathbb{R}^N \quad (k \in \{1, \dots, m\})$ projects the sequence to obtain embedded representation $Y \in \mathbb{R}^{M\times N}$. Finally, this embedded sequence feeds into the Swin Transformer backbone. The data processing procedure is displayed in Figure 1.

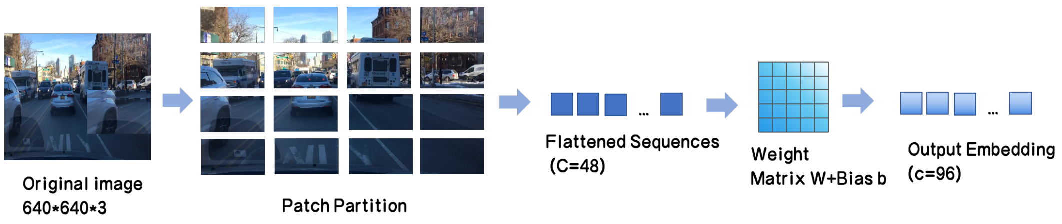


Figure 1. Data processing process.

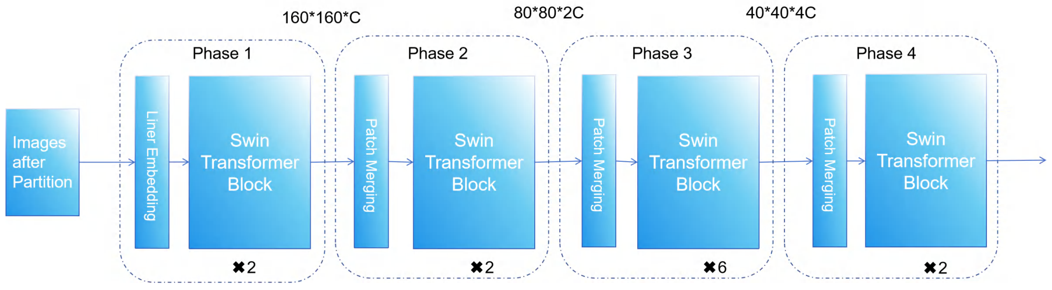


Figure 2. Structure of feature extraction process.

3.2. Hierarchical feature extraction

The hierarchical features are extracted by Swin-Transformer architecture. This model begins by processing the tiniest image patches and progressively increasing the downsampling rate as the network deepens, and multi-scale feature representations are constructed through the combining of neighboring feature blocks in deeper layers. Additionally, to have a better balance of computational efficiency and feature extraction capability, the model computes self-attention within restricted fixed-size windows and employs a window-shifting mechanism to enable feature interaction between different windows, thereby global contextual information can be obtained. Figure 2 displays the structure of the feature extraction process. In the data processing step, the original input pictures with 640 * 640 * 3 dimensions have been partitioned into non-overlapping 160 *160 patches. These patches, measuring 4×4 pixels with 3 channels, form a H/4 $\times$ W/4 grid, where each patch encodes 48 features (derived from $4 \times 4 \times 3$). These processed patches are subsequently fed into the initial stage of the feature extraction process. In phase one of data extraction, each patch first undergoes linear embedding, where its features are projected to an arbitrary dimension C before being processed by the Swin Transformer block for self-attention computation. Then, hierarchical features are generated through a patch merging operation, where neighboring 2×2 patches are down-sampled and their concatenated features are fused, reducing spatial dimensions from H/4 $\times$ W/4 to H/8 $\times$ W/8 and transforming feature dimension to 2C through learned feature fusion. Consequently, in stage 2, the patch numbers are decreased by a factor of 4 while the feature dimensions are doubled from stage 1. Following the same pattern as stage 2, the subsequent stages (3 and 4) further reduce spatial dimensions by a factor of 4 while doubling feature dimensions. This progressive transformation generates a hierarchical representation where each stage contributes to the multi-scale feature extraction.

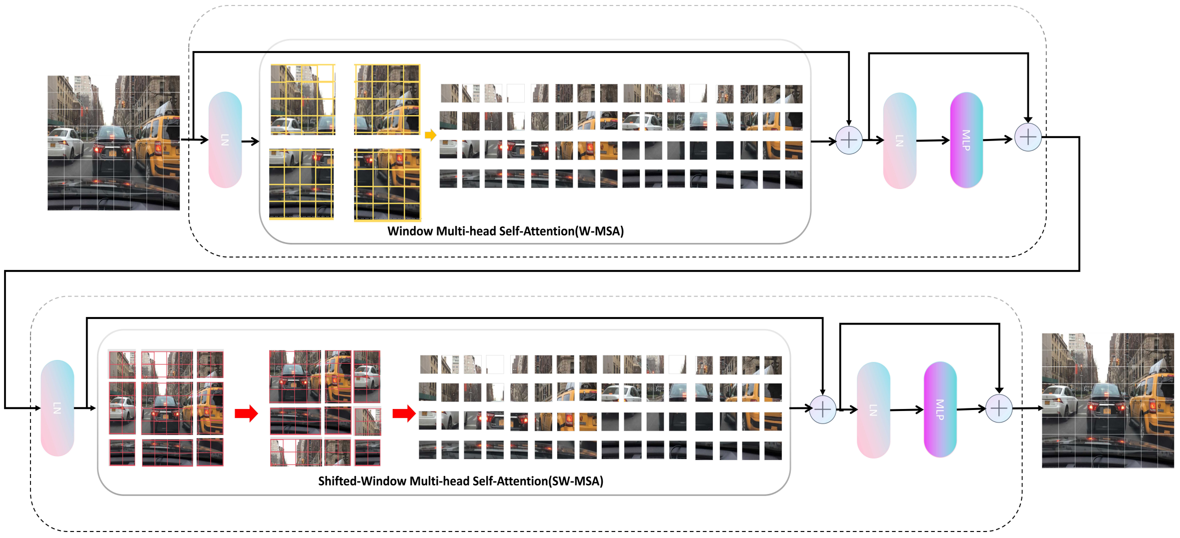

LN
Window Multi-head Self-Attention(W-MSA)
LN
MLP
LN
Shifted-Window Multi-head Self-Attention(SW-MSA)
LN
MLP

Figure 3. Structure of two sequential Swin-Transformer blocks.

The two sequential Swin Transformer blocks are illustrated in Figure 3, incorporating window multi-head self-attention (W-MSA) and its shifted variant (SW-MSA) that enable cross-window feature interactions. For the lst block, $\hat{Z}^l$ donates the output from the W-MSA or SW-MSA module, while Z^l indicates the output from the multi-layer perceptron (MLP) module. The architecture also employs layer normalization before each MSA and MLP block and residual connections afterward. The computational flow through these blocks can be expressed as:

$$\begin{aligned} \hat{z}^l &= \text{W-MSA}(\text{LN}(z^{l-1})) + z^{l-1}, && (3.1)\\ z^l &= \text{MLP}(\text{LN}(\hat{z}^l)) + \hat{z}^l, && (3.2)\\ \hat{z}^{l+1} &= \text{SW-MSA}(\text{LN}(z^l)) + z^l, && (3.3)\\ z^{l+1} &= \text{MLP}(\text{LN}(\hat{z}^{l+1})) + \hat{z}^{l+1}, && (3.4) \end{aligned}$$

The proposed architecture replaces the global self-attention calculation by calculating self-attention within local windows. As displayed in Figure 4, in layer l, it initially employs W-MSA, which divides the image into fixed local windows and performs self-attention calculations within each partitioned area. For an image consisting of p × q patches, which is divided into windows of size n × n, the formulation of calculating computation complexity is listed as follows, where d represents feature dimension.

$$\begin{aligned} \Omega(\text{MSA}) &= 4pqd^2 + 2(pq)^2 d, && (3.5)\\ \Omega(\text{W-MSA}) &= 4pqC^2 + 2n^2 pqd, && (3.6) \end{aligned}$$

with a constant window size, W-MSA maintains linear complexity relative to image size as each window contains significantly fewer patches than the total number of image patches. The (l+1)-th layer employs SW-MSA, which shifts the window downward and to the right to form a new window. Self-attention is then computed within these shifted windows, facilitating feature interaction between patches that were previously isolated in different windows of layer l. However, the window shifting operation initially results in an increased number of windows. To address this, Swin Transformer introduces an efficient cyclic shift strategy. As illustrated in Figure 4, when the shifted windows create incomplete regions at the lower-right boundary, sub-windows A, B, and C from the upper-left are cyclically shifted to form complete windows, enabling self-attention computation in all windows. Since the newly formed windows comprise spatially disconnected sub-windows, it is necessary to implement a masking mechanism to constrain self-attention computation within appropriate sub-windows, preventing attention calculation between non-adjacent patches. After completing calculating self-attention within each window, the cyclic-shifted sub-windows are returned to their original positions. This process maintains the original window count for calculating self-attention, reducing computation complexity while enabling feature interaction and information transferring between adjacent regions.

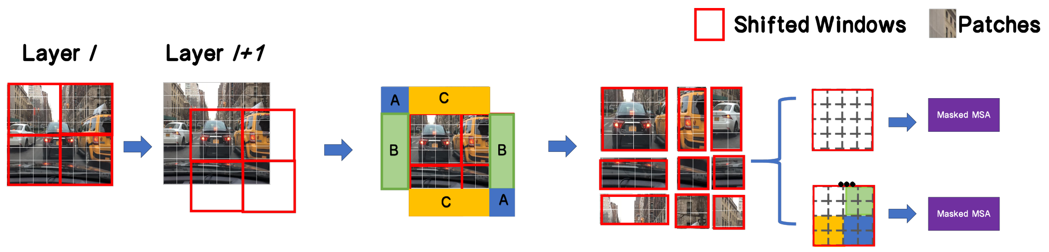


Figure 4. Self-attention calculation based on shifted-window mechanism.

The use of multi-head attention mechanism with window-shifting makes it possible to extract hierarchical and cross-scale features, significantly enhancing the model's ability to capture both local and global information efficiently. This architectural design allows for scalable feature representation across different layers while maintaining low computational complexity, ensuring faster inference speed that meets the requirements for self-driving applications.

The overall structure of the proposed object detection model is displayed in Figure 5. The model processes images through three main components: backbone, neck, and head modules. Specifically, the backbone is responsible for extracting hierarchical cross-scale features from the raw input images, the YOLO-based neck merges and refines these features at multiple scales, and the head performs the final classification and bounding box regression to produce detection results. The Swin Transformer backbone leverages shifted window attention and a hierarchical structure to capture layered cross-scale features, enhancing multi-scale context integration. To strengthen feature representation, SPFF, C2PSA, and C3K2 modules are incorporated, improving feature fusion and extraction while maintaining inference speed. The spatial pyramid pooling fast (SPFF) module aggregates multi-scale contextual information through maximum pooling operations with varying kernel sizes. The cross-stage partial spatial attention (C2PSA) module consists of two partial spatial attention (PSA) components operating on different feature map branches, with their outputs ultimately concatenated. The PSA can be expressed as:

$$\mathrm{PSA}(X) = X \odot \sigma\left(\mathrm{Conv}_{1\times1}\left(\frac{1}{HW}\sum_{i=1}^{H}\sum_{j=1}^{W} X_{i,j} + \max_{i,j} X_{i,j}\right)\right) \tag{3.7}$$

where $\frac{1}{HW}\sum_{i=1}^{H}\sum_{j=1}^{W} X_{i,j}$ represents global average pooling and $\max_{i,j} X_{i,j}$ represents global max pooling.

The C3K2 module is an advanced feature extraction component based on CSPNet (cross stage partial network), with "K" indicating variable kernel sizes, and "2" signifying its dual-branch architecture. The core formula can be expressed as:

$$\mathrm{C3k2}(X) = \mathrm{Conv}_{1\times1}\left(\mathrm{Concat}\left[\mathrm{Bottleneck}^{k}\left(\mathrm{Conv}_{1\times1}(X_1)\right), X_2\right]\right) \tag{3.8}$$

where $X_1, X_2 \in \mathbb{R}^{C/2\times H\times W}$ are split feature maps along channel dimension, and $Bottleneck^k$ are iterations of Bottleneck blocks.

3.3. Depth-wise separable convolution head

As illustrated in Figure 6, the detection head implements a dual-branch architecture, where the input features are routed through two parallel branches: one relies on standard convolution layers, while the other leverages depth-wise separable convolutions (DWConv) to decrease computational overhead and parameter count. A depth-wise separable convolution typically breaks down a regular convolution into two stages; first, a depth-wise convolution that applies filters to each input channel independently, and second, a point-wise (1×1) convolution which fuses channel information.

To highlight the efficiency of DSDH, we provide a formal comparison with the standard convolution-based detection head. For a convolution operation with kernel size $K \times K$, input channels C_{in}, and output channels C_{out}:

$$Params_{\mathrm{std}} = K^2 \times C_{\mathrm{in}} \times C_{\mathrm{out}} \tag{3.9}$$

$$\text{FLOPs}_{\text{std}} = H \times W \times K^2 \times C_{\text{in}} \times C_{\text{out}} \tag{3.10}$$

$$\text{Params}_{\text{ds}} = K^2 \times C_{\text{in}} + C_{\text{in}} \times C_{\text{out}} \tag{3.11}$$

$$\text{FLOPs}_{\text{ds}} = H \times W \times (K^2 \times C_{\text{in}} + C_{\text{in}} \times C_{\text{out}}) \tag{3.12}$$

The ratio of parameter reduction can thus be expressed as:

$$r_{\text{params}} = 1 - \frac{K^2 \times C_{\text{in}} + C_{\text{in}} \times C_{\text{out}}}{K^2 \times C_{\text{in}} \times C_{\text{out}}} \tag{3.13}$$

The ratio reduction of parameter count can be calculated by Eq (3.13), leading to a significant gain in inference speed. In addition, the DSDH is specifically designed to align with the hierarchical feature maps produced by the Swin Transformer backbone and the lightweight neck. The depth-wise stage preserves channel-specific spatial patterns from each scale, while the point-wise stage enables cross-channel fusion, ensuring that multi-scale semantic information is effectively integrated before prediction. By separating these two operations, the model performs fewer multiply-accumulate operations and considerably cuts the number of parameters needed. After extracting features, the outputs from both branches are fused and passed through additional convolutional layers, where classification is optimized with binary cross-entropy (BCE) loss and bounding-box regression is optimized with distribution focal loss (DFL) and complete intersection over union (CIoU) loss. By incorporating depth-wise separable convolutions, this detection head effectively strikes a balance between inference speed and detection accuracy, making it especially valuable for real-time or resource-constrained deployment scenarios like self-driving.

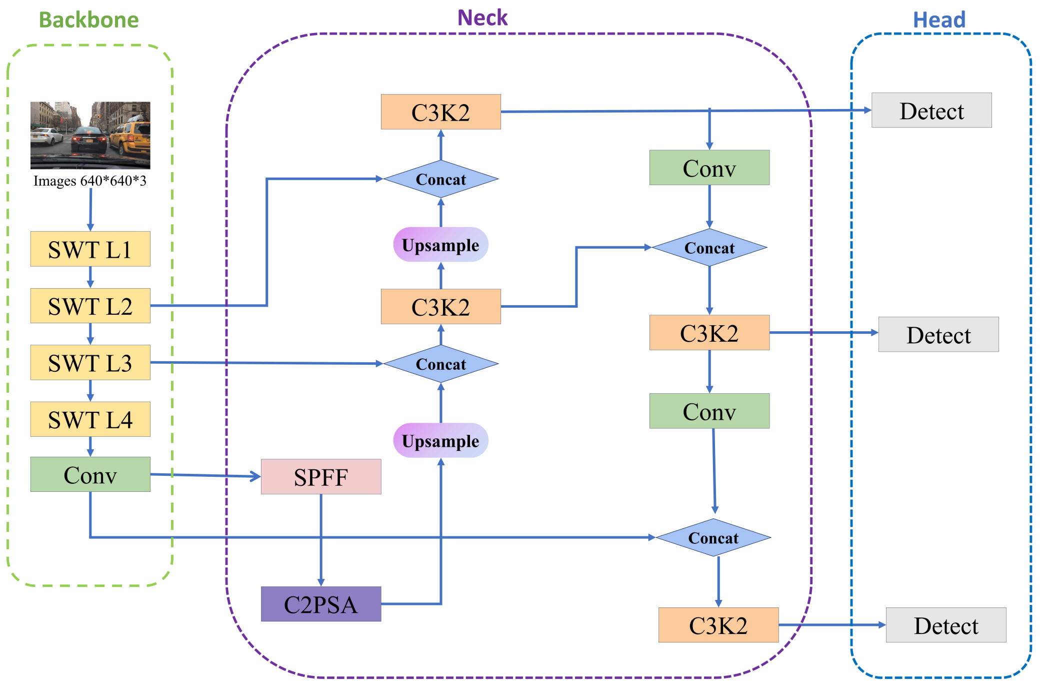
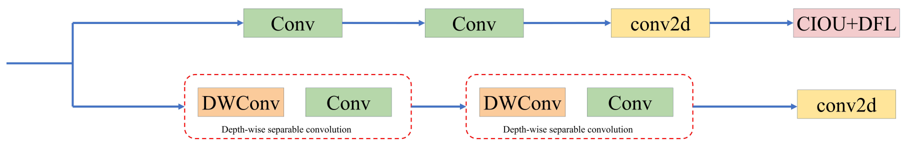


Figure 5. Structure of proposed object detection framework.



Conv
Conv
conv2d
CIOU+DFL
DWConv
Conv
Depth-wise separable convolution
DWConv
Conv
Depth-wise separable convolution
conv2d

Figure 6. Architecture of detect head.

3.3.1. Loss function

We employ *BCE*, *CIOU*, and *DFL* within its detection head to achieve a balanced optimization of classification and regression. In particular, *BCE* is used for robust classification, ensuring clear distinction between target and untargeted regions by minimizing

$$BCE(p, y) = -[y \log(p) + (1 - y) \log(1 - p)] \tag{3.14}$$

which p indicates predicted probability and $y \in \{0, 1\}$ depicts ground-truth label. Meanwhile, *CIOU* helps the model converge faster and more accurately by incorporating overlap, center distance, and aspect ratio into a single metric, typically formulated as

$$\text{CIoU} = 1 - \text{IoU} + \frac{\rho^2(\mathbf{b}, \mathbf{b}^g)}{c^2} + \alpha v. \tag{3.15}$$

Here, $\rho^2(\mathbf{b}, \mathbf{b}^g)$ is the center point displacement between predicted and target bounding boxes, c indicates the diagonal length of the tiniest enclosing box containing both boxes, and α serves as weight function and measures the dimensional proportions between boxes. *DFL* models bounding-box coordinates as a discrete distribution, thereby enabling finer-grained boundary predictions and improving localization quality. By combining these three losses, our model ensures that both the categorization of objects and the precision of their boundaries are optimized in a synthetic manner.

4. Experiments

In this part, we describe the dataset, model performance, and experiment environment. Subsequently, a range of experiments have been performed to prove the effectiveness of the SDOD model.

4.1. Introduction of dataset and experiment setup

The BDD100K dataset is used in this study. This dataset includes a wide range of complex scenarios that may be encountered in autonomous driving like highways with heavy traffic, residential neighborhood with pedestrians and automobiles, as well as city streets with complex traffic signals, etc. The videos captured at various times of the day and under diverse weather conditions consist of this dataset, including a balanced view of daytime and nighttime scenes all over the world. It is divided into 70% training set, 20% test set, and 10% validation set. The validation dataset is used for validating the performance of model training. These diverse training pictures make it possible to transfer the object detection model to other generalized novel datasets. The dataset provides statistical distributions of images, classified by label class, weather, scene, and time, as shown in Figure 7.

The system hardware setup of the experiment is represented in Table 1, which highlights the principal units. The training parameters are shown in Table 2.

4.2. Baseline model

The YOLOv11 is selected as the primary baseline because it represents one of the most recent and widely adopted versions of the YOLO family, offering an improved balance of accuracy and efficiency compared with earlier versions (e.g., YOLOv3, YOLOv8). This choice allows us to benchmark SDOD against a strong and up-to-date detector that is highly relevant for real-time autonomous driving scenarios.

4.3. Evaluation metrics

The evaluation method chosen in this study includes precision, recall, frames per second (FPS), mean average precision (mAP), and floating-point operations (FLOPs). These evaluation metrics can be used for evaluating the efficiency and effectiveness of the proposed model. Precision and Recall reflect the reliability of object detection by balancing false positives and false negatives, while mAP is adopted as a comprehensive metric that aggregates the average precision across multiple categories, thus representing the overall performance of the model on the 10 object classes in the BDD100K dataset. To further validate the practicality of the model, FPS is used to measure real-time processing speed, while FLOPs quantify computational complexity. These two indicators together demonstrate the model's advantages in real-time performance and lightweight deployment under the resource constraints of vehicular systems.

Precision can be described as the percentage of instances that are correctly predicted as positive samples to all the positive-predicated samples. It is used for assessing the positive prediction accuracy, which can be calculated by Eq (4.1).

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}. \tag{4.1}$$

Here, TP represents positive samples that are predicted truly, and FP represents samples that are wrongly predicted as positive.

Recall measures the proportion of cases correctly predicted as positive out of all the true positive samples. It can be expressed as Eq (4.2).

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{4.2}$$

The *mAP* is presented as a metric that integrates both precision and recall, which evaluates how well the algorithm performs across all classes, representing the overall classification performance of the model. It provides an assessment of the model's capability to accurately detect objects in specific situations. A higher *mAP* score indicates better detection performance. The calculation is represented in Eq (4.3), where N is the category number and AP_i means the average precision of class i.

$$\text{mAP} = \frac{\sum_{i=1}^{N} \text{AP}_i}{N} \tag{4.3}$$

The calculation of average precision of each class is presented in AP$_i$, where AP$_i$ is calculated by Eq (4.4).

$$\mathrm{AP}_i = \frac{\sum_{u=1}^{C_i} I(u)\mathrm{PrecAt}(u)}{C_i} \tag{4.4}$$

where C$_i$ donates the total number of ground truth for class *i*, and *I(u)* is an indicator function, which equals one if the u-th detection is correct (True Positive) and it equals zero if it's incorrect (False Positive). *PrecAt(u)* represents the precision at position *u*.

FPS is an indicator used for measuring the processing velocity of a system or model. It indicates how many image frames can be handled in one second, which reflects the system's ability of performing real-time processing.

FLOPs is a metric that quantifies a model's computational complexity by calculating the overall floating-point calculations number performed. Higher FLOPs values represent the model is richer in complexity. For the convolution layer, it can be calculated by:

$$\mathrm{FLOPs} = 2 \times (C_{\mathrm{in}} \times K \times K) \times C_{\mathrm{out}} \times H_{\mathrm{out}} \times W_{\mathrm{out}} \tag{4.5}$$

where K denotes kernel size, C_{in} refers input channels, C_{out} indicates output channels, and $H_{\mathrm{out}} \times W_{\mathrm{out}}$ specifies output dimensions.

Table 1. Configuration of system and experiment setup.

Memory	GPU	CPU	Cache size	Cuda	CudNN
32 GB	Nvidia RTX 4070	Intel XEON 6780E	108 MB	11.2	9.6

Table 2. Training parameters.

Parameter	**Value**
Learning rate	0.001 (cosine annealing)
Optimizer	Adam
Number of iterations	300

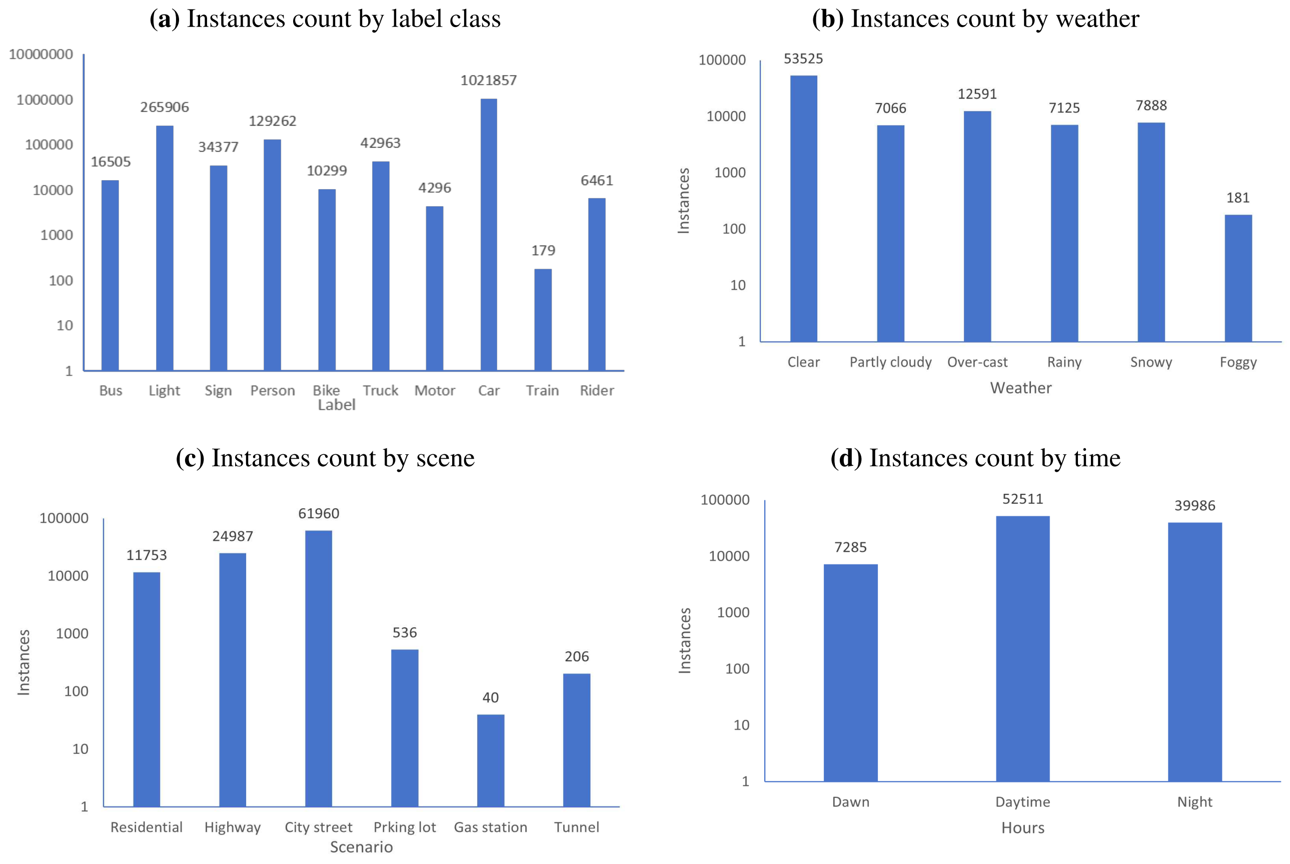


Figure 7. Statistical distributions of dataset.

4.4. Results

In this section, we will display the findings from experiments using the object detection model we developed. Then, we evaluate how well the proposed SDOD model performs by comparing it to other object detection algorithms across different metrics on the BDD100K dataset.

4.4.1. Detecting outputs achieved by the proposed model

This section exhibits the intermediate processing steps and final results of the model. Specifically, the input image is converted to pixels by the model, and then the model implements the feature detection process through neural networks to identify patterns in the input image, followed by classifying and detecting process using the further-refined features. Figure 8 illustrates heat maps produced by the model after incorporating the attention mechanism. The heatmap illustrates how the model focuses on predictions, highlighting the regions the model emphasizes during categorization.

Table 3 compares the overall performance of the SDOD model with other existing models in terms of accuracy, computational speed, and model complexity. It is evident that our SDOD model demonstrates remarkable performance by achieving an outstanding balance between accuracy and computational efficiency. While maintaining highly competitive detection accuracy with a precision of 0.72 and mAP of 0.47, which significantly surpass other comparison models like single shot multibox detector (SSD), faster R-CNN, and cascade R-CNN, and approaching the best-performing YOLOv11s. Although its precision and mAP are slightly lower than YOLOv11s (0.77 and 0.52), SDOD significantly outperforms in terms of inference speed and computational efficiency, achieving the highest FPS (222) among all models and requiring only 6.0 G FLOPs, which is only nearly one-third of YOLOv11s's computational cost (21.5 G FLOPs). This strategic balance between accuracy and efficiency makes SDOD particularly suitable for real-world applications where both reliable detection performance and real-time processing are essential requirements.

Detecting small objects like traffic lights, traffic signs, and people is always a tough problem; however, it can be seen from Table 4 that the proposed model excels in detecting small objects in complex scenes. It exhibits excellent performance across a wide range of object scales, and it proves effective in handling diverse self-driving scenarios, with the fewest computational complexity. The table compares the average precision (AP) results of the proposed SDOD model and YOLOv11n for various object classes on the BDD100K dataset. Overall, SDOD outperforms YOLOv11n in most categories, particularly for Traffic Light (0.51 vs. 0.32) and Traffic Sign (0.55 vs. 0.38); SDOD also shows slightly better performance for detecting Bike (0.52 vs. 0.49). Car achieves the highest AP scores in both models (0.74 and 0.69, respectively), while both models show low performance on Train detection, with SDOD achieving 0.02 and YOLOv11n 0.01, which is due to the unbalance data of BDD100K. Although YOLOv11n slightly outperforms SDOD in several categories, the differences are minor. In general, SDOD demonstrates competitive or superior performance in various object categories, highlighting its potential for effective and efficient real-world applications in autonomous driving.

The correlation between precision and recall metrics of different thresholds can be illustrated by the precision-recall curve (P-R Curve). It visualizes how well the model balances these two key performance metrics, with recall on the horizontal axis and precision on the vertical axis. Starting from an initial threshold value, the curve traces how these two metrics shift in response to threshold adjustments. Generally, a higher threshold leads to improvement in precision but reduction in recall. This occurs since the increasing in threshold makes the model more conservative in predictions, prioritizing confident detection, resulting in lower false positive rate but some valid cases may be missed. The precision-recall curve is displayed in Figure 9. Each colored line corresponds to one category, illustrating how precision changes as we adjust the threshold. As shown, classes like car and traffic sign exhibit relatively high P–R curves, indicating better detection performance, while categories such as train remain near the bottom, reflecting lower precision across different recall levels. The P-R curve demonstrates the SDOD model performs better in reducing false positive rates, maintaining consistent performance across all object categories and detecting objects of various sizes.

Loss function curves depict how the model's predicted error changes throughout its training process. It monitors the difference between what the model predicts and the actual ground truth values, which helps us understand the learning progress of the model. The loss function curve provides insights of the model's convergence, which can help find problems like overfitting or underfitting and make adjustments to improve performance of the model. Figure 10 displays how the bounding box loss changes over 300 training epochs, comparing the performance between our proposed model and others. As training progresses, the loss steadily decreases, indicating that the model is learning to refine object localization more accurately. Initially, the loss values start relatively high, then drop sharply within several epochs as the network quickly learns fundamental features. Afterward, the curve continues to

Table 3. Comparative results of the SDOD framework on BDD100K.

Method	Precision	Recall	mAP	FPS	FLOPs (G)
SSD	0.26	0.21	0.22	31	70.0
Faster-R-CNN	0.31	0.22	0.28	17	134.0
Cascade-R-CNN	0.39	0.29	0.33	15	86.0
Sparse-R-CNN	0.42	0.27	0.36	24	79.0
CenterNet	0.48	0.30	0.40	28	43.0
YOLO-v3-tiny [37]	0.61	0.35	0.39	45	19.2
YOLO-v8n [37]	0.65	0.42	0.45	50	8.1
YOLO-v11n	0.68	0.35	0.42	200	6.5
YOLO-v11s	0.77	0.45	0.52	83	21.5
SDOD	0.72	0.38	0.47	222	6.0

decline more gradually, which is further fine-tuning of bounding-box positions. By the later epochs, the loss converges toward a relatively low value, reflecting the model's improved ability to precisely enclose and identify target objects. It is evident that SDOD converges smoothly and maintains a near-consistently low bounding-box regression loss throughout most of the training process. Although its final loss is marginally higher than YOLOv11s, the difference is small relative to SDOD's real-time and low complexity advantages.

SDOD exhibits a faster convergence speed compared to YOLOv11s, which can be attributed to the hierarchical representation capability of the Swin Transformer backbone. However, a slight accuracy gap remains between SDOD and YOLOv11s for certain categories of small objects. This phenomenon may arise from the shifted-window attention mechanism in the Swin Transformer, which can create boundary artifacts when small objects are located near window edges. Such artifacts can compromise local feature integrity and ultimately reduce detection precision in these regions. To further address these issues and enhance the accuracy of small object detection, several optimization directions can be considered including refining multi-scale attention through adaptive window partitioning, enhancing multi-scale shallow features, applying targeted data augmentation, etc.

The precision curve is shown in Figure 11, which provides a comparison of the improvement in precision for our newly developed model and others. Precision serves as a key measurement in object detection that indicates how accurately the model can correctly identify targets. Among the models shown, SDOD stands out with its consistent and relatively smooth increase in precision throughout the epochs. Starting from a lower point, SDOD steadily rises and reaches a final precision of 0.74. Compared to other models, SDOD performs more stable. Although the final precision is lower than YOLOv11s, SDOD's smoother and steadier rise suggests that it may offer a more reliable solution.

4.5. Ablation study

In this section, we represent the results of ablation study to present the role of each part in our model. We choose mAP, loss-function curve, precision curve for detailed performance comparison. The results of the ablation study in Table 5 highlight the impact of different components on the performance of the SDOD model. The complete SDOD model achieves the most outstanding

Table 4. Comparative results of different classes on BDD100K.

Class	AP (SDOD)	AP (YOLO-v11n)
Bike	0.52	0.49
Bus	0.51	0.52
Car	0.74	0.69
Motor	0.42	0.43
Person	0.45	0.46
Rider	0.41	0.31
Traffic Light	0.51	0.32
Traffic Sign	0.55	0.38
Train	0.02	0.01
Truck	0.52	0.59
Small objects	0.47	0.42

performance across all metrics, with a precision of 0.72, recall of 0.38, and mAP of 0.47. Notably, it also displays superior computational efficiency, achieving the highest inference speed at 222 FPS while requiring only 6.0 Giga FLOating-point Operation (GFLOPs). This indicates that the full architecture successfully integrates all components for optimal performance. When examining the impact of the DSDH, its removal leads to a noticeable performance decrease, with precision dropping to 0.65 and mAP to 0.38. More significantly, without DSDH's efficient convolution operations, the computational cost increases dramatically to 71.0 GFLOPs and speed decreases significantly to 127 FPS, which indicates the considerable growth in computational complexity and highlights the depth-wise separable structure is crucial for both accuracy and computational efficiency. Moreover, when the DSDH is combined with the Darknet backbone (Darknet + DSDH), the precision, recall, and mAP also drop 0.19, 0.16, and 0.16, respectively, which demonstrates the effect of promoting detection accuracy for our newly developed backbone. The reduction in computation speed and increase in computational cost also exhibits the efficiency benefits of our backbone. Compared to the baseline model YOLOv11n, there is a marginal improvement in the precision, recall, and mAP, and the superior computation efficiency is validated. Overall, SDOD strikes an excellent balance between accuracy and efficiency, while removing DSDH or using a less efficient backbone leads to a decline in performance.

Figure 12 is a visualization of the loss function trends of four different models in the ablation experiment. It is clear that all models exhibit a gradual decline in loss as training progresses, among which the SDOD remains consistently low and smooth. YOLOv11n exhibits somewhat reduced performance. The SDOD without DSDH experiences a worse loss reduction compared to YOLOv11n, and Darknet + DSDH converges with the highest final loss, pointing out that the base Darknet architecture does not provide as effective feature extraction as SDOD. This comparison clearly emphasizes the value of incorporating our feature extraction backbone and detection head.

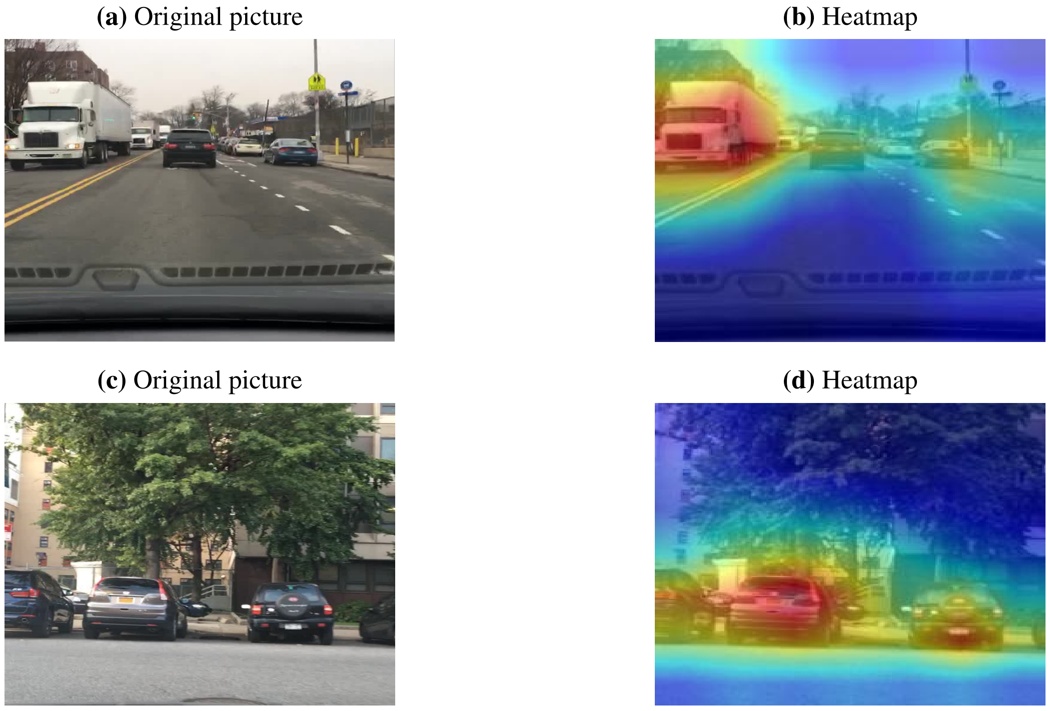


Figure 8. Heatmaps of the object detection process.

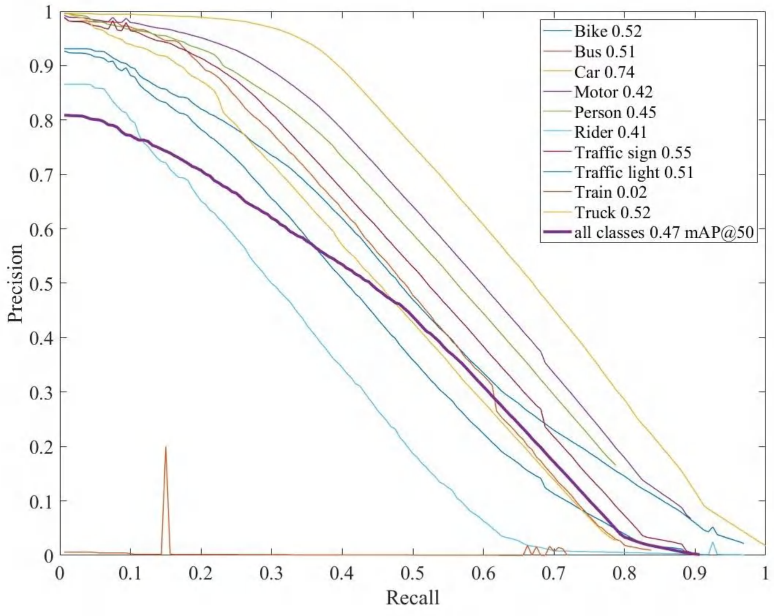


Figure 9. P-R curve of SDOD model for all categories.

Table 5. Results comparison of ablation study.

Method	Precision	Recall	mAP	FPS	GFLOPs
YOLO-v11n	0.68	0.35	0.45	200	6.5
SDOD without DSDH	0.65	0.33	0.38	127	71.0
Darknet + DSDH	0.53	0.22	0.31	43	45.0
SDOD	0.72	0.38	0.47	222	6.0

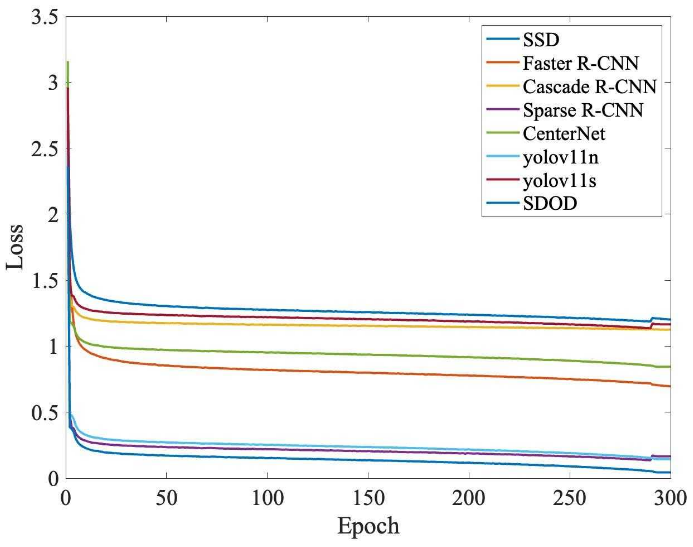


Figure 10. Loss comparison of different models.

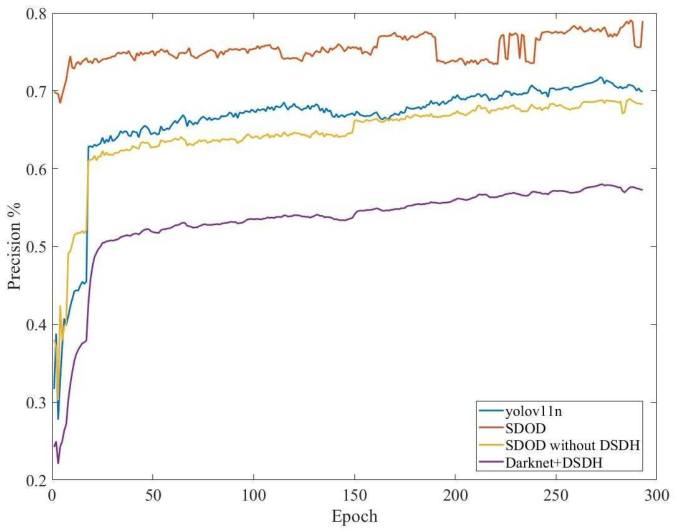
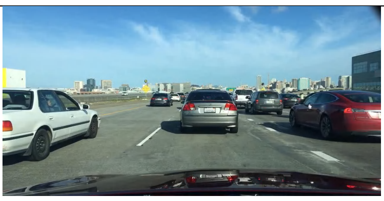
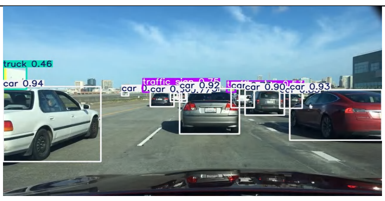
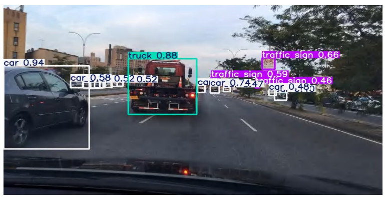
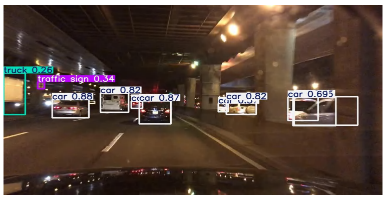
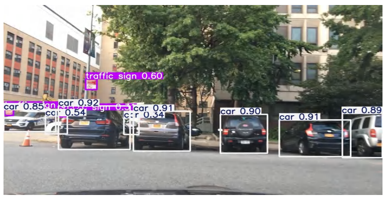
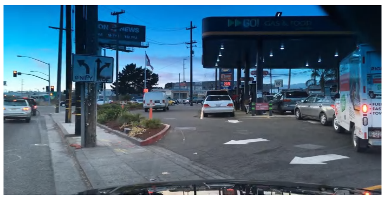
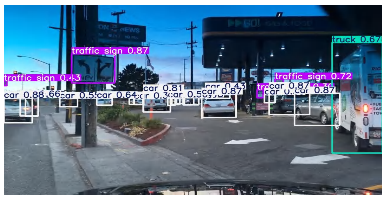
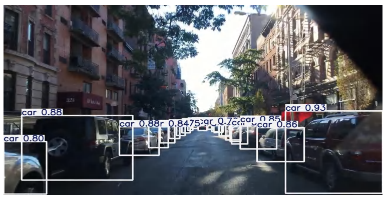
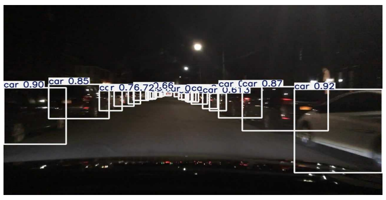


Figure 11. Precision comparison of different models.

Table 6. Comparison of original images and SDOD detection results across different scenarios.

Scenario	Original Image	SDOD Detection
Highway		
City Street		 car 0.94 truck 0.88 traffic_sign 0.66 traffic_sign 0.59 car 0.58
Tunnel		 traffic_sign 0.34 car 0.88 car 0.82 car 0.87 car 0.82
Parking Lot		 traffic_sign 0.60 car 0.85 car 0.92 car 0.91 car 0.90 car 0.91 car 0.89
Gas Station		 traffic_sign 0.87 truck 0.67 traffic_sign 0.72 car 0.81
Residential		
Night		

The precision curves represented in Figure 13 compare the training progression of precision of different model variants in the ablation study. It is evident that the SDOD model exhibits superior performance, achieving the highest final precision with rapid initial improvement and stable

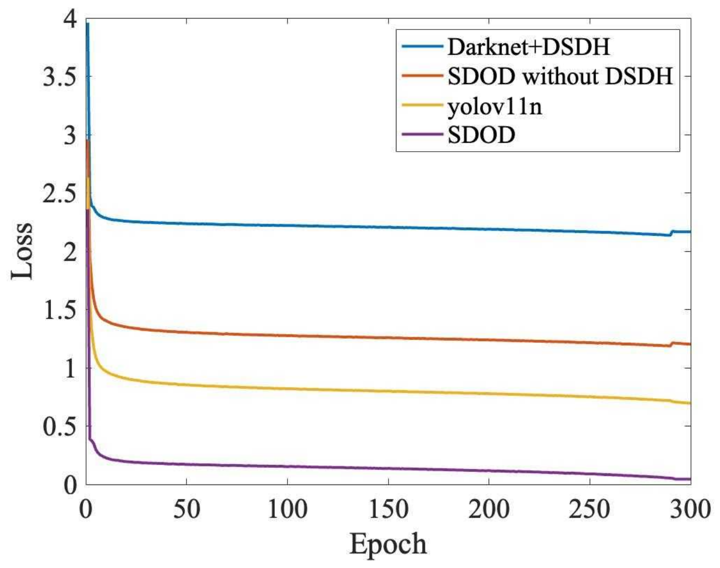


Figure 12. Comparison of loss function curve for ablation study.

convergence. The model SDOD without DSDH shows a noticeable drop in performance compared to the integrated SDOD model, indicating that the DSDH detection head is crucial for achieving the higher precision. The Darknet+DSDH configuration performs significantly worse than the other variants, which suggests that while the detection head contributes to better detection capabilities, the use of a weaker backbone, like Darknet, limits the model's overall precision. Following the model

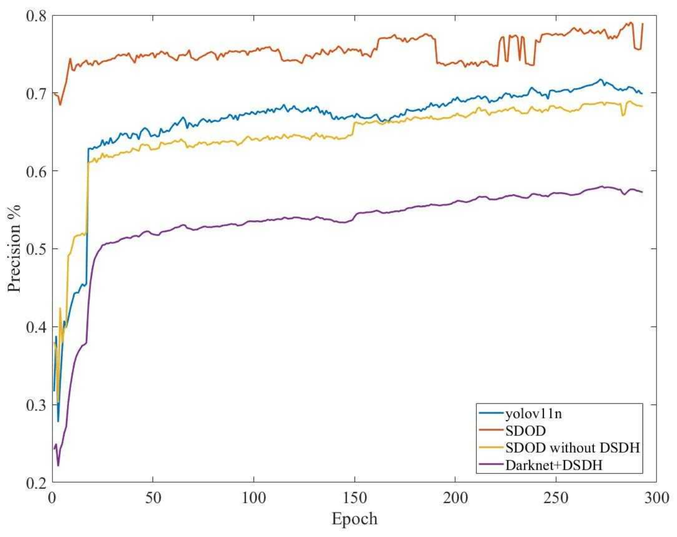


Figure 13. Comparison of Precision curve for ablation study.

training process, we evaluated the performance of object detection in different scenes using different models. The detection results visualized in Table 6 depict the ability of detecting objects under diverse environmental challenges. It is obvious that the proposed model can achieve higher accuracy in detecting objects from pictures in real-word scenes. We selected seven typical and common driving scenarios to demonstrate the object detection capabilities of the proposed model. These scenarios include highways, city street, tunnel, parking lot, gas station, residential areas, and night-time driving conditions. Each of these environments presents unique challenges for object detection, such as varying light conditions, complex backgrounds, and diverse object sizes. The detection results include the predicted bounding box for different categories along with confidence scores indicating the certainty of model prediction. It can be seen from Table 6 that the SDOD model can detect various kinds of objects in different driving environments with high confidence, which shows robustness in challenging conditions like night-time driving and tunnels. Additionally, the model adapts well to different object scales, for example, it can not only identify targets nearby but also locate long-distance cars accurately.

4.6. Technical discussion of results

The superior detection accuracy achieved by the proposed method can be attributed to the enhanced feature representation provided by the Swin Transformer backbone. Its shifted-window attention mechanism enables global context modeling and effective cross-scale feature aggregation, which is particularly beneficial for small and distant object detection in complex driving scenes. Compared to the original YOLOv11s backbone, this design reduces the risk of missing low-contrast targets under challenging conditions.

The notable improvement in processing speed is primarily due to the depth-wise separable convolution in the detection head, which reduces redundant computations by decomposing standard convolutions into depth-wise and point-wise operations. This design aligns well with the hierarchical multi-scale features produced by the Swin Transformer backbone, as confirmed in the ablation study. These findings validate the adaptability and robustness of the proposed approach for real-world autonomous driving environments.

5. Conclusions

In this work, we proposed SDOD, an efficient object detection model specifically tailored for autonomous driving scenarios. By integrating the Swin Transformer backbone, SDOD achieves stronger hierarchical feature representation, which improves detection accuracy under complex road conditions. The lightweight depth-wise separable convolution detection head further reduces computational cost and parameters, enabling faster inference while maintaining robust precision. Experimental results on the BDD100K dataset confirm the advantages of SDOD: the model requires only 6.0 GFLOPs, which is far lower than YOLOv11s at 21.5 GFLOPs, while still reaching a competitive mAP of 0.47. Moreover, SDOD achieves 222 FPS, outperforming all compared models and proving its superior real-time capability. These results highlight the clear advantages of SDOD: a unique combination of accuracy, efficiency, and adaptability that makes it highly suitable for deployment in resource-constrained embedded vehicular systems.

The future research will further investigate ways to improve the accuracy and efficiency of detecting targets under self-driving scenes, make improvement for the model, especially in strengthening the ability for detecting small objects, long-distance objects, and increasing the precision of detection in challenging environments, such as extreme weather conditions and tunnel entrances.

Use of AI tools declaration

The authors declare they have not used Artificial Intelligence (AI) tools in the creation of this article.

Acknowledgements

This work was supported by the National Natural Science Foundation of China (U2468202), National Key Research and Development Program of China (2022YFB4300500), Technological Research and Development Program of China State Railway Group Co., Ltd. (P2024G003).

Conflict of interest

The authors declare there is no conflicts of interest.

References

1. Y. Dai, S. G. Lee, Perception, planning and control for self-driving system based on on-board sensors, *Adv. Mech. Eng.*, **12** (2020), 1687814020956494. https://doi.org/10.1177/1687814020956494
2. C. Qiu, H. Tang, Y. Yang, X. Wan, X. Xu, S. Lin, et al., Machine vision-based autonomous road hazard avoidance system for self-driving vehicles, *Sci. Rep.*, **14** (2024), 12178. https://doi.org/10.1038/s41598-024-62629-4
3. M. Reda, A. Onsy, A. Y. Haikal, A. Ghanbari, Path planning algorithms in the autonomous driving system: A comprehensive review, *Robot. Auton. Syst.*, **174** (2024), 104630. https://doi.org/10.1016/j.robot.2024.104630
4. S. M. Hosseinian, H. Mirzahossein, Efficiency and safety of traffic networks under the effect of autonomous vehicles, *Iran. J. Sci. Technol. Trans. Civ. Eng.*, **48** (2024), 1861–1885. https://doi.org/10.1007/s40996-023-01291-8
5. Q. Chen, Y. Xie, S. Guo, J. Bai, Q. Shu, Sensing system of environmental perception technologies for driverless vehicle: A review of state of the art and challenges, *Sens. Actuators A Phys.*, **319** (2021), 112566. https://doi.org/10.1016/j.sna.2021.112566
6. M. Kuderer, S. Gulati, W. Burgard, Learning driving styles for autonomous vehicles from demonstration, in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, (2015), 2641–2646. https://doi.org/10.1109/ICRA.2015.7139555
7. A. Charroud, K. El-Moutaouakil, V. Palade, A. Yahyaouy, Enhanced autoencoder-based lidar localization in self-driving vehicles, *Appl. Soft Comput.*, **152** (2024), 111225. https://doi.org/10.1016/j.asoc.2023.111225
8. Z. Liu, Y. Cai, H. Wang, L. Chen, H. Gao, Y. Jia, et al., Robust target recognition and tracking of self-driving cars with radar and camera information fusion under severe weather conditions, *IEEE Trans. Intell. Transp. Syst.*, **23** (2021), 6640–6653. https://doi.org/10.1109/TITS.2021.3059674
9. Z. S. Dhaif, N. K. El-Abbadi, A review of machine learning techniques utilised in self-driving cars, *Iraqi J. Comput. Sci. Math.*, **5** (2024), 1. https://doi.org/10.52866/ijcsm.2024.05.01.015

10. A. Boukerche, Z. Hou, Object detection using deep learning methods in traffic scenarios, *ACM Comput. Surv.*, **54** (2021), 1–35. https://doi.org/10.1145/3434398

11. N. J. Zakaria, M. I. Shapiai, R. Abd-Ghani, M. N. M. Yassin, M. Z. Ibrahim, N. Wahid, Lane detection in autonomous vehicles: A systematic review, *IEEE Access*, **11** (2023), 3729–3765. https://doi.org/10.1109/ACCESS.2023.3234442

12. L. Yang, Y. Xu, S. Wang, C. Yuan, Z. Zhang, B. Li, et al., PDNet: Toward better one-stage object detection with prediction decoupling, *IEEE Trans. Image Process.*, **31** (2022), 5121–5133. https://doi.org/10.1109/TIP.2022.3193223

13. C. Y. Wang, A. Bochkovskiy, H. Y. M. Liao, Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, (2023) ,7464–7475.

14. L. Du, R. Zhang, X. Wang, Overview of two-stage object detection algorithms, *J. Phys. Conf. Ser.*, **1544** (2020), 012033. 10.1088/1742-6596/1544/1/012033

15. G. Jocher, J. Qiu, *Ultralytics YOLO11, Version 11.0.0*, Computer software, 2024.

16. Z. Chen, K. Chen, J. Chen, Vehicle and pedestrian detection using support vector machine and histogram of oriented gradients features, in *2013 International Conference on Computer Sciences and Applications*, (2013), 365–368. https://doi.org/10.1109/CSA.2013.92

17. Y. Tian, R. Sukthankar, M. Shah, Spatiotemporal deformable part models for action detection, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, (2013), 2642–2649. https://doi.org/10.1109/CVPR.2013.341

18. J. Yan, Z. Lei, L. Wen, S. Z. Li, The fastest deformable part model for object detection, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, (2014), 2497–2504.

19. R. Girshick, J. Donahue, T. Darrell, J. Malik, Rich feature hierarchies for accurate object detection and semantic segmentation, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, (2014), 580–587. https://doi.org/10.1109/CVPR.2014.81

20. R. Girshick, Fast R-CNN, in *Proceedings of the IEEE International Conference on Computer Vision*, (2015), 1440–1448. https://doi.org/10.1109/ICCV.2015.169

21. S. Ren, K. He, R. Girshick, J. Sun, Faster R-CNN: Towards real-time object detection with region proposal networks, *Adv. Neural Inf. Process. Syst.*, **28**, 2015.

22. Z. Cai, N. Vasconcelos, Cascade R-CNN: Delving into high quality object detection, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, (2018), 6154–6162. https://doi.org/10.1109/CVPR.2018.00644

23. A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. Gomez, et al., Attention is all you need, *Adv. Neural Inf. Process. Syst.*, **30**, 2017.

24. A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, et al., An image is worth 16×16 words: Transformers for image recognition at scale, preprint, arXiv:2010.11929.

25. Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, et al., Swin transformer: Hierarchical vision transformer using shifted windows, in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, (2021), 10012–10022. https://doi.org/10.1109/ICCV48922.2021.00986

26. J. Redmon, S. Divvala, R. Girshick, A. Farhadi, You only look once: Unified, real-time object detection, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, (2016), 779–788.

27. J. Redmon, A. Farhadi, Yolo9000: Better, faster, stronger, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, (2017), 7263–7271. https://doi.org/10.1109/CVPR.2017.690

28. X. Han, J. Chang, K. Wang, Real-time object detection based on YOLO-v2 for tiny vehicle object, *Proc. Comput. Sci.*, **183** (2021), 61–72. https://doi.org/10.1016/j.procs.2021.02.031

29. J. Redmon, A. Farhadi, Yolov3: An incremental improvement, preprint, arXiv:1804.02767.

30. J. Choi, D. Chun, H. Kim, H. J. Lee, Gaussian Yolov3: An accurate and fast object detector using localization uncertainty for autonomous driving, in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, (2019), 502–511. https://doi.org/10.1109/ICCV.2019.00059

31. A. Bochkovskiy, C. Y. Wang, H. Y.M. Liao, Yolov4: Optimal speed and accuracy of object detection, preprint, arXiv:2004.10934. https://doi.org/10.48550/arXiv.2004.10934

32. B. Mahaur, K. K. Mishra, Small-object detection based on Yolov5 in autonomous driving systems, *Pattern Recognit. Lett.*, **168** (2023), 115–122. https://doi.org/10.1016/j.patrec.2023.03.009

33. M. Sohan, T. Sai-Ram, C. V. Rami-Reddy, A review on Yolov8 and its advancements, in *International Conference on Data Intelligence and Cognitive Informatics*, (2024), 529–545. https://doi.org/10.1007/978-981-99-7962-2_39

34. H. Tao, Z. Huang, Y. Wang, J. Qiu, V. Stojanovic, Efficient feature fusion network for small objects detection of traffic signs based on cross-dimensional and dual-domain information, *Meas. Sci. Technol.*, **36** (2025), 035004. https://doi.org/10.1088/1361-6501/adb2ad

35. H. Tao, Y. Zheng, Y. Wang, J. Qiu, V. Stojanovic, Enhanced feature extraction yolo industrial small object detection algorithm based on receptive-field attention and multi-scale features, *Meas. Sci. Technol.*, **35** (2024), 105023. https://doi.org/10.1088/1361-6501/ad633d

36. Y. Sun, H. Tao, V. Stojanovic, Pseudo-label guided dual classifier domain adversarial network for unsupervised cross-domain fault diagnosis with small samples, *Adv. Eng. Inform.*, **64** (2025), 102986. https://doi.org/10.1016/j.aei.2024.102986

37. J. Li, Q. W. Deng, W. X. Gao, B. Yang, L. Jia, J. Zhou, et al., Dsf-yolo for robust multiscale traffic sign detection under adverse weather conditions, *Sci. Rep.*, **15** (2025), 24550. https://doi.org/10.1038/s41598-025-02877-0