IECE Transactions on Emerging Topics in Artificial Intelligence
http://dx.doi.org/10.62762/TETAI.2024.894227

RESEARCH ARTICLE

YOLOv8-Lite: A Lightweight Object Detection Model for Real-time Autonomous Driving Systems

Ming Yang[1,*] **and Xiangyu Fan**[1]

[1] College of Architecture and Design, Tongmyong University, Busan 608-711, Korea

Abstract

With the rapid development of autonomous driving technology, the demand for real-time and efficient object detection systems has been increasing to ensure vehicles can accurately perceive and respond to the surrounding environment. Traditional object detection models often suffer from issues such as large parameter sizes and high computational resource consumption, limiting their applicability on edge devices. To address this issue, we propose a lightweight object detection model called YOLOv8-Lite, based on the YOLOv8 framework, and improved through various enhancements including the adoption of the FastDet structure, TFPN pyramid structure, and CBAM attention mechanism. These improvements effectively enhance the performance and efficiency of the model. Experimental results demonstrate significant performance improvements of our model on the NEXET and KITTI datasets. Compared to traditional methods, our model exhibits higher accuracy and robustness in object detection tasks, better addressing the challenges in fields such as autonomous driving, and contributing to the advancement of intelligent transportation systems.

Keywords: autonomous driving, object detection, YOLOv8, real-time performance, intelligent transportation.

1 Introduction

Autonomous driving technology, as a significant breakthrough in today's technological realm, is leading us into a new era of transportation[1]. However, as its development delves deeper, we are gradually becoming aware of the myriad challenges and hurdles that autonomous driving faces. In practical applications, autonomous driving systems must navigate through various complex traffic scenarios and situations, including road conditions, weather conditions, traffic signals, and the behavior of other road users. The complexity of these factors sometimes makes it difficult for autonomous driving systems to make correct decisions, thus sparking concerns about their safety and reliability[2, 3]. Among the plethora of autonomous driving technologies, object detection serves as a crucial component, playing a pivotal role. Object detection systems perceive objects and obstacles in the surrounding environment, providing real-time environmental information and data to autonomous vehicles. Accurately and efficiently detecting and identifying various objects on the road, such as vehicles, pedestrians, and bicycles, is paramount for ensuring the safety and stability of autonomous driving systems. Therefore, research and improvement in object detection technology have become one of the

Academic Editor:
Teerath Kumar

Submitted: 17 December 2023
Accepted: 02 April 2024
Published: 07 April 2024

Vol. 1, **No.** 1, 2024.
10.62762/TETAI.2024.894227

***Corresponding author:**
✉ Ming Yang
yangming1995414@gmail.com

Citation
Yang, M., & Fan, X. (2024). YOLOv8-Lite: A Lightweight Object Detection Model for Real-time Autonomous Driving Systems. *IECE Transactions on Emerging Topics in Artificial Intelligence*, 1(1), 1–16.

crucial directions in the development of autonomous driving technology[4, 5].

In recent years, with the continuous development of autonomous driving technology, various algorithms have flourished in the field of object detection to meet the demand for precise and efficient object recognition in autonomous driving systems[6]. Firstly, the Yolo series of algorithms is one of the classic representatives in the field of object detection. The Yolo (You Only Look Once) algorithm is renowned for its high speed and concise network structure, capable of detecting and classifying multiple objects in an image in a single forward pass. Its model employs convolutional neural networks (CNNs) for feature extraction and utilizes fully connected layers for object detection[7]. However, due to the issue of inaccurate localization when dealing with small objects and dense scenes, the Yolo algorithm may exhibit lower detection accuracy in certain situations. Secondly, the Faster R-CNN algorithm is another commonly used object detection algorithm[8]. This algorithm extracts candidate object regions by introducing a Region Proposal Network (RPN), followed by object detection and classification through ROI pooling layers and fully connected layers. Faster R-CNN improves detection accuracy compared to the Yolo algorithm but suffers from a complex network structure and relatively slow speed due to its large computational overhead. Additionally, the SSD (Single Shot MultiBox Detector) algorithm predicts object locations and categories simultaneously on feature maps of different scales[9], achieving object detection in a single forward pass. With a simple model structure and fast detection speed, the SSD algorithm is suitable for real-time applications. However, there is still room for improvement in small object detection and object localization accuracy with the SSD algorithm. Lastly, the Mask R-CNN algorithm extends object detection into the realm of semantic segmentation. Building upon Faster R-CNN, this algorithm introduces additional segmentation branches to generate precise object boundaries[8, 10]. While Mask R-CNN excels in object detection and segmentation tasks, its complex network structure and higher computational costs limit its widespread use in practical applications[11].

The deployment of vehicle detection models also encounters certain challenges. High-precision models often have large-scale architectures, making them difficult to deploy. In practical applications, especially on embedded systems or edge devices with limited resources, deploying these large models becomes even more challenging. Furthermore, even with sufficient computational resources, these models may still have slow inference speeds, affecting their real-time performance[12, 13]. In past research, we have observed a lack of balance between accuracy, speed, and the number of parameters in previous works. Some models may prioritize achieving high accuracy but perform poorly in terms of speed and deployment, while others may have advantages in speed but fall short in accuracy. Therefore, we face the challenge of how to design and train vehicle detection models to maintain high accuracy while having fast inference speeds and fewer parameters[14]. The key to addressing this challenge lies in effective model design and optimization. We can reduce the model's scale and inference costs by exploring lightweight network architectures, quantization methods, and model compression techniques. Additionally, optimizing algorithms and leveraging hardware acceleration can improve the model's inference speed to meet real-time requirements. By considering accuracy, speed, and the number of parameters comprehensively, we can better balance the performance of vehicle detection models and achieve their effective deployment and application in various practical scenarios[15, 16].

In response to the existing challenges, we propose the YOLOv8-Lite solution aimed at enhancing the model's inference speed and reducing its complexity by introducing the lightweight network, FastDet. Additionally, we incorporate the CBAM (Convolutional Block Attention Module) attention mechanism to enhance feature extraction. YOLOv8-Lite is dedicated to addressing the balance between accuracy and efficiency in vehicle detection models, focusing on lightweight architectures and attention-based feature enhancement. Through the design of the FastDet backbone network, we streamline computations while maintaining high detection performance. The integration of the CBAM attention mechanism allows the model to capture richer feature information, thereby improving its ability to accurately detect vehicles in various scenarios. The introduction of this approach represents a significant advancement in achieving a better balance between accuracy, speed, and complexity in vehicle detection models, offering improved feasibility for deployment on resource-constrained devices and enhancing performance in real-world applications.

Here we introduce the three contributions of this paper:

- The introduction of the YOLOv8-Lite framework addresses the challenges faced by vehicle detection models by incorporating the lightweight backbone network, FastDet. This framework aims to improve model inference speed and reduce complexity while prioritizing efficiency without compromising accuracy, providing a practical solution for real-time vehicle detection applications.
- The integration of the Convolutional Block Attention Module (CBAM) attention mechanism into the feature extraction process further enhances the performance of the YOLOv8-Lite framework. By selectively focusing on informative features, the CBAM attention mechanism improves the model's ability to capture relevant information, thereby achieving more precise vehicle detection across various environmental conditions and scenarios.
- This model is more suitable for deployment on resource-constrained devices such as embedded systems or edge devices, expanding its applicability and effectiveness in real-world scenarios.

2 Related Works

2.1 The application of two-stage object detection algorithms in autonomous driving

The application of two-stage object detection algorithms in autonomous driving is a crucial component of the development of autonomous driving technology[17]. First is Fast R-CNN, which achieves end-to-end training of region proposal and object detection by introducing the Region of Interest (RoI) pooling layer, resulting in higher detection accuracy and improved speed. However, it is still constrained by the computational overhead of region proposal generation. Next is Faster R-CNN, which enhances inference speed and efficiency by introducing the Region Proposal Network (RPN) to share convolutional features, yet it still relies on selective search algorithms for region proposal generation, limiting its speed. Further, HyperNet improves localization accuracy by directly predicting bounding box parameters from shared feature maps, effectively handling objects of various scales and aspect ratios, but may increase complexity and computational cost. Mask R-CNN extends Faster R-CNN by adding parallel mask prediction branches, achieving instance segmentation and accurate object localization, albeit at a higher computational cost due to additional mask prediction tasks. Pedestrian FPN (PFN) is specifically designed for pedestrian detection in autonomous driving scenarios, utilizing Feature Pyramid Networks (FPN) to effectively handle pedestrians of various scales and orientations, although it may lead to longer inference times. CRAFT (Cascade R-CNN with Adaptive Feature Fusion) improves detection accuracy by iteratively refining bounding box predictions at multiple stages using a cascade architecture and adaptive feature fusion, albeit potentially requiring longer inference times[18]. In summary, these two-stage object detection algorithms strike a balance between accuracy and speed in autonomous driving, optimizing object localization and instance segmentation, yet they may incur higher computational costs and longer inference times. The selection of suitable algorithms depends on the specific requirements of the autonomous driving system[19].

2.2 The application of one-stage object detection algorithms in autonomous driving

In the field of autonomous driving, the application of one-stage object detection algorithms has garnered widespread attention[20]. Transformer-based object detection algorithms like DETR have attracted researchers' interest. These algorithms enable end-to-end object detection by introducing attention mechanisms and leveraging global context information, eliminating the need for candidate region generation steps seen in traditional two-stage methods and thus simplifying the process. However, these algorithms often suffer from noticeable performance degradation when dealing with a large number of objects, and they incur higher training and inference costs[21]. Furthermore, the evolution of algorithms from YOLOv5 to YOLOv8 has also been notable. YOLOv5 introduced a lightweight object detection framework that achieved faster inference speed and higher accuracy through improvements in backbone networks and feature extraction methods. YOLOv8 further optimized network structures and training strategies, enhancing detection performance and robustness. Additionally, RetinaNet is another noteworthy algorithm that addresses class imbalance issues by introducing focal loss but suffers from slower inference speeds. Similarly, SSD, known for its simple and efficient design, holds a significant position in the object detection domain; however, its performance diminishes when detecting small objects or objects with different aspect ratios.

In summary, these algorithms have their own strengths and weaknesses in terms of detection performance, speed, and applicability, thus requiring comprehensive consideration based on specific application requirements and scene constraints when selecting the appropriate algorithm[22].

3 Methodology

3.1 YOLOv8 Network

YOLOv8 is the latest advancement in the You Only Look Once (YOLO) series, boasting outstanding performance and innovative design[23]. This model enhances the accuracy and efficiency of object detection by introducing a series of new techniques and optimizations. The network architecture of YOLOv8 is illustrated in Figure 1.

The improvements in YOLOv8 can be attributed to several aspects:

- YOLOv8 draws inspiration from the design philosophy of YOLOv7 ELAN for its backbone network and Neck part. It replaces the C3 structure of YOLOv5 with a richer C2f structure and fine-tunes different channel numbers for models of different scales. These refinements and optimizations significantly enhance the model's performance and make it more suitable for various complex scenarios.

- Compared to YOLOv5, YOLOv8 introduces significant changes to its Head part. It adopts the mainstream decoupled head structure, separating classification and detection heads. Additionally, it transitions from an Anchor-Based approach to an Anchor-Free approach, further improving the model's detection accuracy and robustness.

- To better optimize the model's performance, we employ the TaskAlignedAssigner positive sample allocation strategy and introduce Distribution Focal Loss. These new loss computation schemes effectively enhance the model's training stability and convergence speed, further boosting its performance.

Through these innovative designs and optimization strategies, YOLOv8 has made significant advancements in the field of object detection, providing more reliable and efficient solutions for applications such as autonomous driving.

3.2 YOLOv8-Lite

Our overall network architecture takes into account the computational resource constraints of devices in scenarios such as autonomous driving. Therefore, we opted for a lightweight network model called YOLOv8-Lite. As shown in Figure 2, YOLOv8-Lite is based on the YOLOv8 framework and achieves efficient object detection through simplified design and optimized computation. Recognizing the demands for real-time performance and efficiency, we avoided large network structures like CSPDarkNet53 and instead adopted our proposed FastDet structure, which simplifies network complexity and is better suited to the computational resources of edge devices. Additionally, we introduced the TFPN (Top-Down Feature Pyramid Network) pyramid structure to address information loss during feature fusion. The TFPN structure effectively merges feature maps from different levels, enhancing the network's detection capability across different scales while reducing information loss. This pyramid structure design allows for a more comprehensive understanding of input images, thereby improving object detection accuracy. Furthermore, we introduced the CBAM (Convolutional Block Attention Module) mechanism to further enhance the network's representation capability. The CBAM mechanism adaptively adjusts attention within feature maps, enabling the network to focus more accurately on important target areas, thereby improving object detection performance. The introduction of this mechanism enhances the network's robustness in handling complex scenarios and occlusions, making it more suitable for practical applications such as autonomous driving.

3.3 CBAM

In the field of deep learning, attention mechanisms play a crucial role in allowing models to focus on relevant information while filtering out irrelevant details[24]. CBAM (Convolutional Block Attention Module) is a popular attention mechanism designed to enhance feature representations in deep neural networks. It consists of two complementary sub-modules: the Channel Attention Module (CAM) and the Spatial Attention Module (SAM). CAM recalibrates feature maps across channels to emphasize informative features, while SAM recalibrates feature maps spatially to capture contextual information. We introduced the CBAM attention module into the Feature Pyramid Network (FPN) to strengthen the feature extraction network, effectively extracting crucial features from the feature maps and improving

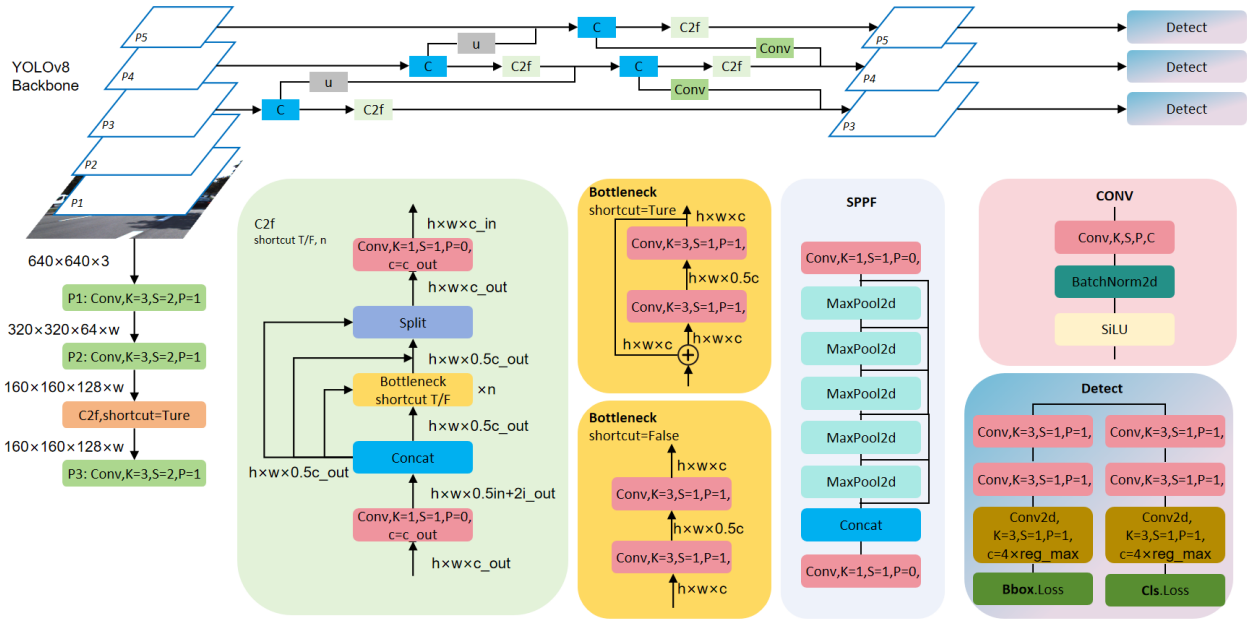

Figure 1. YOLOv8 Network Architecture Diagram[33].

the model's accuracy. Specifically, within the FPN architecture, we incorporated 6 ECA attention blocks. These attention blocks dynamically adjust the importance of features across channel and spatial dimensions, enabling the model to focus on relevant regions and enhance performance. The network architecture of the CBAM is depicted in Figure 3.

In the following equations, we introduce the key mathematical formulations of CBAM:

The Channel Attention Map C is computed as the sigmoid activation of the average channel-wise feature responses.

$$C = \sigma\left(\frac{1}{H \times W}\sum_{i=1}^{H}\sum_{j=1}^{W}\text{ReLU}(W_{\text{c}} \cdot X_{i,j})\right) \quad (1)$$

where: C is the channel attention map. H and W are the height and width of the feature map, respectively. $X_{i,j}$ is the feature map at position (i, j). W_{c} represents the weights used for channel-wise convolution. σ denotes the sigmoid activation function.

The Spatial Attention Map S is computed similarly, capturing spatial dependencies across the feature map.

$$S = \sigma\left(\frac{1}{C}\sum_{c=1}^{C}\text{ReLU}(W_{\text{s}} \cdot X_{:,:,c})\right) \quad (2)$$

where: S is the spatial attention map. C is the number of channels in the feature map. $X_{:,:,c}$ represents the c-th channel of the feature map. W_{s} represents the weights used for spatial-wise convolution.

The Output Feature Map Y is obtained by combining the channel and spatial attention maps element-wise with the original feature map X, enhancing the model's ability to focus on relevant regions.

$$Y = X \odot (C + S) \quad (3)$$

where: Y is the output feature map. $\odot$ denotes element-wise multiplication.

The Global Average Pooling operation computes the average value of each channel in the feature map X, providing a global context for the spatial features.

$$\text{GlobalAvgPool}(X) = \frac{1}{H \times W}\sum_{i=1}^{H}\sum_{j=1}^{W}X_{i,j} \quad (4)$$

where: $\text{GlobalAvgPool}(X)$ represents the global average pooling operation applied to the feature map X.

The Multi-Layer Perceptron (MLP) $\text{MLP}(X)$ is applied to the input feature map X to capture more complex patterns and relationships. Here, W_1 and W_2 represent the weights of the MLP layers.

$$\text{MLP}(X) = \text{ReLU}(W_2 \cdot (\text{ReLU}(W_1 \cdot X) + X)) \quad (5)$$

where: $\text{MLP}(X)$ denotes the multi-layer perceptron (MLP) applied to the input feature map X. W_1 and W_2 represent the weights of the MLP layers.

3.4 TFPN

Our TFPN network architecture is uniquely designed for object detection tasks, aiming to fully leverage the

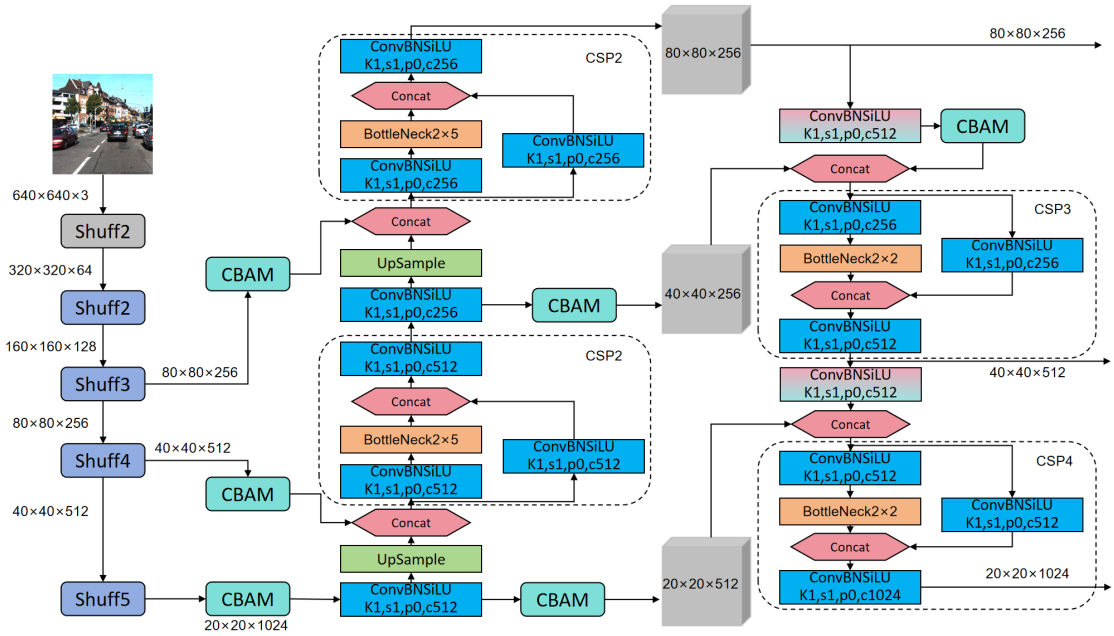

Figure 2. YOLOv8-Lite Network Architecture Diagram

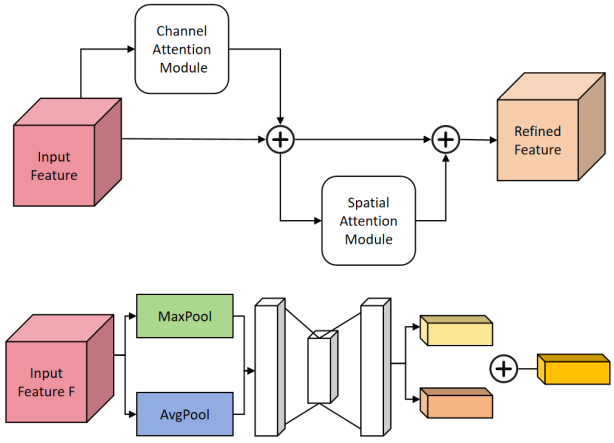

Figure 3. CBAM Network Architecture Diagram

information from feature maps at different scales [25]. Firstly, we enhance the output feature maps of Dark3, Dark4, and Dark5 in the backbone network, which contain target information at different scales. Next, we employ a top-down upsampling approach, using bilinear interpolation to resize the feature maps from deeper layers to match the sizes of shallower feature maps, facilitating feature fusion. This method ensures the effective fusion of feature maps at different levels and the comprehensive utilization of the information they carry. Subsequently, we perform bottom-up convolution or pooling operations to gradually downsample the feature maps from shallower to deeper layers, extracting more semantic information. This bidirectional feature extraction process helps the model better understand the semantic content of images, thereby improving the accuracy and robustness of object detection. The network architecture of TFPN is illustrated in Figure 4.

Below, we explain the main mathematical derivation formulas of TFPN:

$$H_l = \text{Upsample}(H_{l+1}) + W_l * H_l^{'} + b_l \tag{6}$$

where: H_l is the feature map at level l, H_{l+1} is the feature map at level $l+1$, W_l is the weight matrix for convolution at level l, $H_l^{'}$ is the feature map after bottom up processing at level l, b_l is the bias term at level l.

$$H_l^{'} = \text{ReLU}(H_l) \tag{7}$$

where: $\text{ReLU}(\cdot)$ is the rectified linear unit activation function.

$$W_l = \text{Conv}(H_{l-1}) \tag{8}$$

where: W_l is the weight matrix for convolution at level l, H_{l-1} is the feature map at level $l-1$, $\text{Conv}(\cdot)$ denotes the convolution operation.

$$\text{Upsample}(H_{l+1}) = \text{Bilinear\_Interpolation}(H_{l+1}) \tag{9}$$

where: $\text{Upsample}(H_{l+1})$ is the upsampled feature map from level $l+1$, $\text{Bilinear\_Interpolation}(\cdot)$ denotes the bilinear interpolation operation.

$$H_1 = \text{Conv\_Pooling}(X) \tag{10}$$

where: H_1 is the feature map at the first level, X is the input image or feature map, $\text{Conv\_Pooling}(\cdot)$ denotes the convolution and pooling operations.

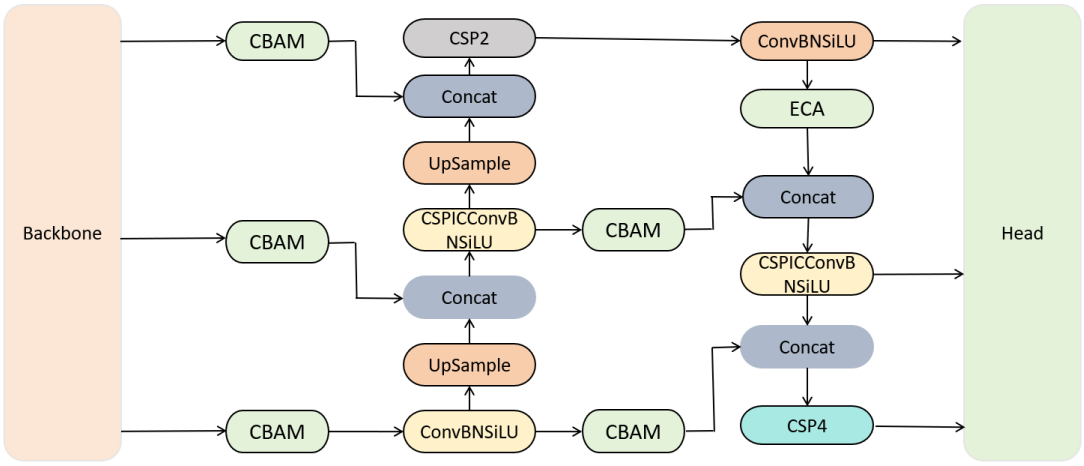

Figure 4. TFPN Network Architecture Diagram

3.5 FastDet

In the CSPDarkNet53 structure, the Dark module serves as a critical feature extraction module designed to extract image features for subsequent tasks. Adopting a deep residual structure, this module utilizes a series of convolutional and pooling layers to extract and abstract features from input images [26]. While the Dark module has been designed with lightweight principles in mind, it may still have some limitations under certain circumstances. For instance, when processing large-scale feature maps, the Dark module may require significant computational resources, resulting in slower inference speeds. Particularly in scenarios where real-time performance is crucial, the performance limitations of the Dark module may restrict the model's applicability. Therefore, despite making progress in lightweight design, further optimization and improvement are still needed to enhance its performance in fast object detection tasks.

We introduce the innovative FastDet structure, designed based on lightweight principles, to improve the inference speed and accuracy of object detection models. This structure begins with a simple 1×1 convolution operation, followed by the Fast module performing channel shuffling, dividing the feature map into two segments along the feature channels. This design enables the model to more efficiently extract and utilize feature information, significantly boosting inference speed while maintaining accuracy. Figure 5 illustrates the structures of Dark and FastDet. From the figure, it is evident that the Dark module consists of multiple convolutional layers and pooling layers, forming a deep residual network for feature extraction. In contrast, the FastDet structure employs a more simplified approach, with a simpler architecture. Figure 5 illustrates the network architecture of FastDet.

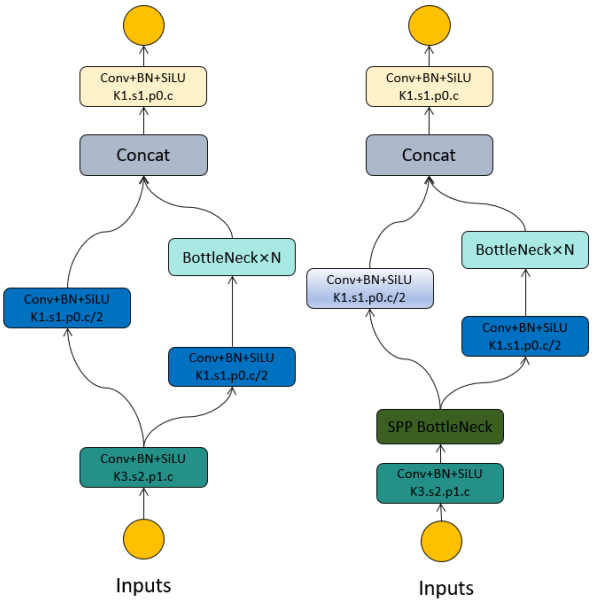

Figure 5. FastDet Network Architecture Diagram

Below, we explain the main mathematical derivation formulas of FastDet:

$$\mathbf{Y} = \mathbf{W}\mathbf{X} + \mathbf{b} \tag{11}$$

where: $\mathbf{Y}$ is the output feature map, $\mathbf{W}$ is the weight matrix, $\mathbf{X}$ is the input feature map, $\mathbf{b}$ is the bias vector.

$$\mathbf{Y} = \text{Shuffle}(\mathbf{X}) \tag{12}$$

where: $\mathbf{Y}$ is the output feature map after channel shuffling, $\text{Shuffle}(\cdot)$ denotes the channel shuffling operation.

$$\mathbf{Y}_1, \mathbf{Y}_2 = \text{Split}(\mathbf{X}) \tag{13}$$

where: $\mathbf{Y}_1$ and $\mathbf{Y}_2$ are the output feature maps split along the channel dimension, Split$(\cdot)$ denotes the feature map splitting operation.

$$\mathbf{Y} = \text{Conv}(\mathbf{X}) \tag{14}$$

where: $\mathbf{Y}$ is the output feature map after convolution, Conv$(\cdot)$ denotes the convolution operation.

$$\mathbf{Y} = \text{Pooling}(\mathbf{X}) \tag{15}$$

where: $\mathbf{Y}$ is the output feature map after pooling, Pooling$(\cdot)$ denotes the pooling operation.

4 Experiments

4.1 Dataset

To evaluate the performance of our proposed method in various scenarios, we selected two representative datasets: NEXET and the KITTI dataset.

The NEXET dataset is a large-scale video dataset widely used in the field of autonomous driving. It covers various urban streets [27], highways, and rural roads under different weather conditions and road conditions. The dataset provides high-resolution video clips along with rich annotation information, including bounding boxes, class labels, and motion trajectories of vehicles. This makes the NEXET dataset an ideal choice for studying and evaluating object detection algorithms.

The KITTI dataset is a classic dataset for autonomous driving [28], jointly released by the Karlsruhe Institute of Technology (KIT) and Toyota. It contains images, point clouds [29], and pose information collected from sensors mounted on vehicles. The dataset covers various tasks in urban and suburban road scenes, including object detection, object tracking, and stereo vision. Annotation information in the KITTI dataset includes bounding boxes and class labels for objects such as vehicles, pedestrians, and bicycles. Due to its realistic scenes and rich annotation information, the KITTI dataset is widely used for the development and evaluation of autonomous driving algorithms.

4.2 Experimental Environment

As shown in Table 1, our experimental setup is presented.

Table 1. Experimental environment demonstrated.

Parameter	Configuration
CPU	Intel Core i7-12700KF
GPU	NVIDIA GeForce RTX 4090 (24 GB)
CUDA version	CUDA 11.6
Python version	Python 3.9.13
Deep learning framework	Pytorch 2.0.0
Operating system	Ubuntu 22.04.2

4.3 Experimental Details

4.3.1 Data preprocessing

We selected a total of 53,460 images from the NEXET dataset. According to the partition criteria of 70% for training, 20% for validation, and 10% for testing, the training set comprises 37,422 images, the validation set comprises 10,692 images, and the test set comprises 5,346 images. Similarly, we selected 37,586 images from the KITTI dataset, with the training set containing 26,310 images, the validation set containing 7,517 images, and the test set containing 3,759 images.The specific results are shown in Table 2.

Table 2. Dataset Split

Dataset	Training Set	Validation Set	Test Set
NEXET	37422	10692	5346
KITTI	26310	7517	3759

During the data cleaning phase, we first identify and remove outliers and noise from the image data. Next, we assess the quality of the images and eliminate those with poor quality to ensure consistency and reliability of the data. Additionally, we perform standardization procedures on the images, such as resizing and background removal, to prepare a clean dataset for model training.

To increase the diversity and richness of the dataset, we employed data augmentation techniques to augment the image data. Data augmentation includes operations such as random rotation, flipping, cropping, and brightness adjustment to generate additional training samples, thereby enhancing the model's generalization ability and robustness.

4.3.2 Model training

Our training settings for the YOLOv8-Lite model took into account the practical requirements of the autonomous driving domain. We opted for a lightweight model suitable for edge devices and employed representative parameter configurations during training. As shown in Table 3, the learning rate was set to 0.001, and we utilized the Adam optimizer to enhance convergence speed and accuracy. Given the high real-time demands of autonomous driving tasks, we set the batch size to 640 to improve training efficiency. Additionally, to prevent model overfitting, we applied a small weight decay value of 0.001. The

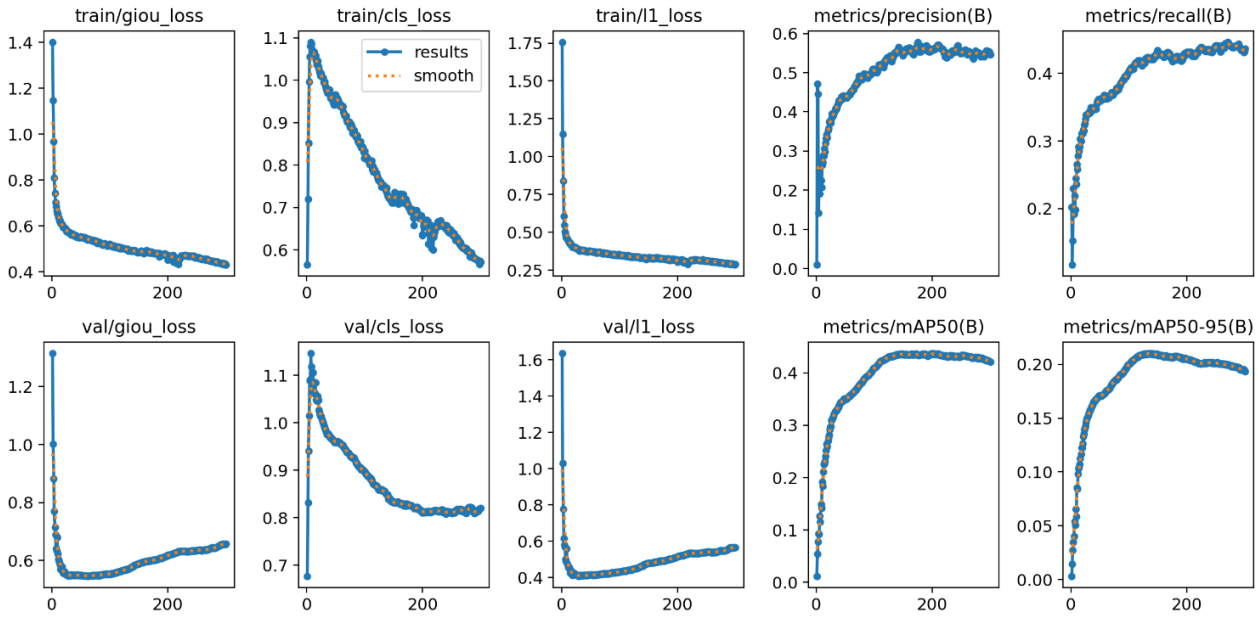

Figure 6. The performance of the YOLOv8-Lite model.

training epochs were set to 300 to ensure the model adequately learned the features and patterns within the dataset. Ultimately, our YOLOv8-Lite model, with these parameters, consisted of 4,385,523 parameters and 252 layers, providing an efficient and reliable solution for object detection tasks in the autonomous driving scenario.

Table 3. Model Parameter Settings

Parameter	Value
Learning Rate	0.001
Optimizer	Adam
Batch Size	640
Weight Decay	0.001
Training Epochs	300
Model Parameters	4,385,523
Number of Layers	252

During the training process, we closely monitored the changes in various metrics, which are crucial for evaluating model performance and training progress. Figure 6 illustrates the variations in bounding box loss, confidence loss, and class loss. Analyzing these loss curves enables us to gain a clearer understanding of the model's performance during training. Specifically, our model demonstrated outstanding performance, with gradual decreases in bounding box loss, confidence loss, and class loss, indicating a progressive improvement in the accuracy and precision of the model for object detection tasks. Our model can effectively detect objects in images and accurately classify and localize them. This steady improvement validates the effectiveness of our training process and the robustness of the model. Our model exhibited excellent performance on both the NEXET and KITTI datasets, providing a reliable solution for object detection tasks in practical applications such as autonomous driving.

Algorithm 1 shows the training process of our network.

4.3.3 Evaluation Metrics

The experiment adopts the following evaluation metrics to comprehensively assess the model's performance:

Accuracy: Evaluates the ratio of accurately classified instances among the total instances.

$$Accuracy = \frac{True\ Positives + True\ Negatives}{Total\ Instances} \tag{16}$$

Where $True\ Positives$ are the number of correctly predicted positive instances, $True\ Negatives$ are the number of correctly predicted negative instances, and $Total\ Instances$ is the total number of instances.

Recall: Gauges the ratio of accurately identified true positive instances by the model among all genuine positive instances.

$$Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives} \tag{17}$$

Where $True\ Positives$ are the number of correctly predicted positive instances, and $False\ Negatives$ are the number of incorrectly predicted negative

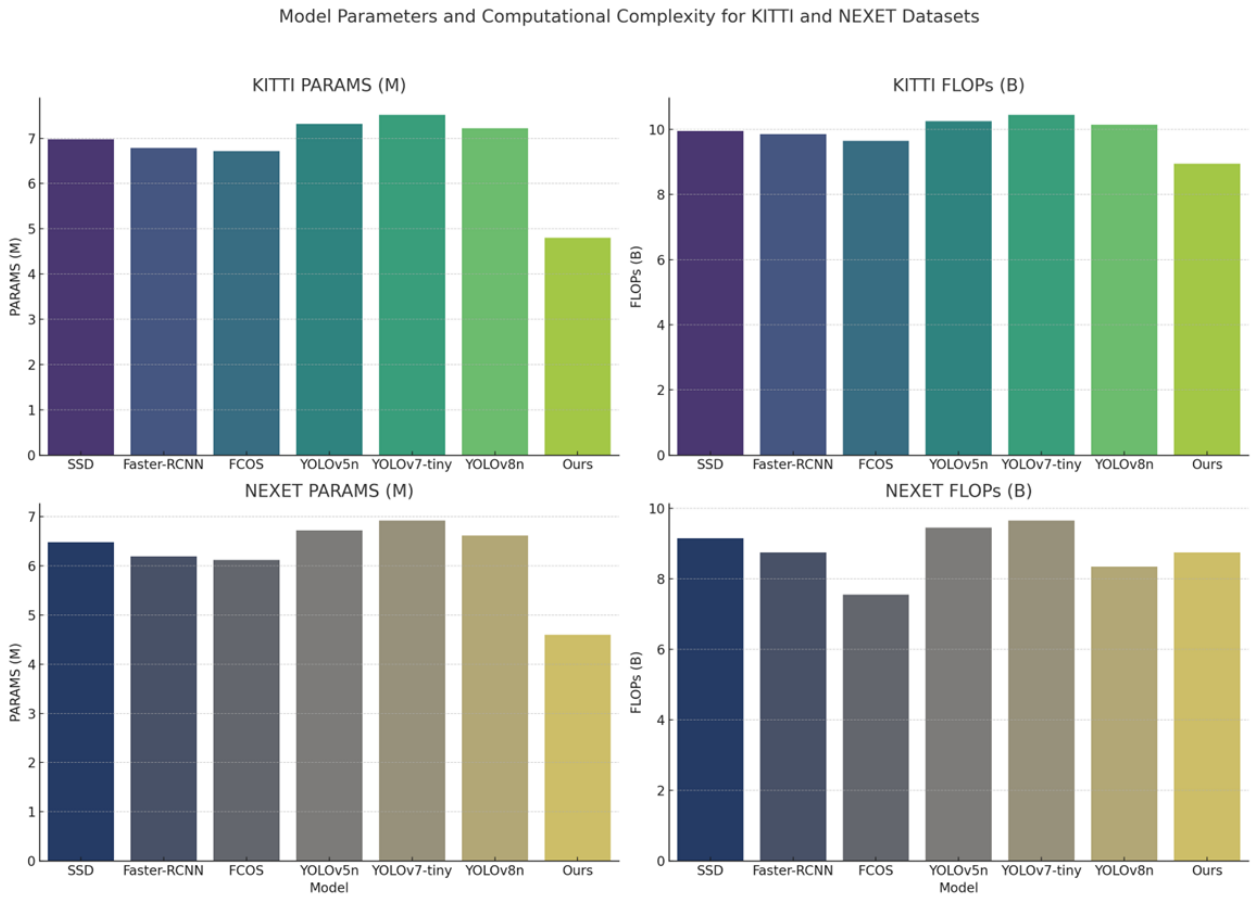

Figure 7. Model parameters and computational complexity for KITTI and NEXET datasets.

Algorithm 1 Training YOLOv8-Lite Network

Require: Training dataset: NEXET and KITTI
Ensure: Trained YOLOv8-Lite network

```
Initialize YOLOv8-Lite network parameters randomly
Set learning rate α, batch size B, and number of epochs E
for epoch = 1 to E do
    for i = 1 to N/B do
        Sample a mini-batch of size B from the training dataset
        Forward pass: Compute predicted bounding boxes and confidence scores
        Compute loss function ℒ using ground truth bounding boxes
        Backward pass: Update network parameters using gradient descent
    end for
    Evaluate precision and mAP@0.5 on the validation set
    if Validation mAP@0.5 has improved then
        Save current model parameters
    end if
end for
```

instances. Recall measures the proportion of true positive instances correctly identified by the model among all actual positive instances.

Precision: Assesses the ratio of accurately predicted true positive instances among all instances flagged as positive by the model.

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives} \tag{18}$$

Where $True\ Positives$ are the number of correctly predicted positive instances, and $False\ Positives$ are the number of incorrectly predicted positive instances. Precision measures the proportion of true positive instances among all instances predicted as positive by the model.

F1 Score: Represents the harmonic average of precision and recall, ensuring equilibrium between them.

$$F1\ Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{19}$$

F1 Score is the harmonic mean of precision and recall, providing a balance between them.

Table 4. Comparison of different methods on KITTI and NEXET datasets.

	KITTI				NEXET			
Methods	Precision	F1 Score	mAP@0.5	FPS	Precision	F1 Score	mAP@0.5	FPS
SSD[9]	58.37	54.09	56.29	18	56.19	51.86	53.09	9
Faster-RCNN[20]	64.16	52.07	49.09	20	61.90	51.42	47.88	15
FCOS[32]	71.58	52.62	59.82	21	68.86	49.98	57.63	19
YOLOv5n[30]	66.13	64.56	60.89	42	63.02	61.20	58.67	39
YOLOv7-tiny [31]	67.92	62.62	65.36	62	65.69	59.91	63.16	60
YOLOv8n [19]	70.82	66.63	66.35	57	68.58	64.36	64.22	54
Ours	76.72	74.62	75.32	85	74.52	71.82	73.12	72

mAP: Represents the average precision scores' mean across all classes or categories, a commonly utilized metric in object detection tasks.

$$mAP = \frac{1}{N}\sum_{i=1}^{N} AP_i \tag{20}$$

Where N is the number of classes or categories, and AP_i is the Average Precision for class i. mAP is the mean of the average precision (AP) scores across all classes or categories, commonly used in object detection tasks.

mAP@[IoU]: Denotes the average precision means at varying Intersections over Union (IoU) thresholds, assessing the model's effectiveness across diverse IoU requirements.

$$mAP@[IoU] = \frac{1}{N}\sum_{i=1}^{N} AP@[IoU]_i \tag{21}$$

Where $AP@[IoU]_i$ is the Average Precision at the IoU threshold for class i. mAP@[IoU] is the mean of the average precision at different Intersections over Union (IoU) thresholds, evaluating the model's performance under different IoU requirements.

4.4 Experimental Results and Analysis

Table 4 provides a comparative analysis of different methods applied on the KITTI and NEXET datasets. This table includes key performance indicators such as precision, F1 score, mAP@0.5, and frames per second (FPS), aiming to comprehensively evaluate the effectiveness of various methods in object detection tasks. In the evaluation on the KITTI dataset, the method proposed in this study performed the best, achieving scores of 76.72% in precision, 74.62% in F1 score, 75.32 in mAP@0.5, and 85 in FPS. These results not only demonstrate its superior accuracy but also its significant advantage in processing speed. For the NEXET dataset, our method also displayed outstanding performance, specifically achieving a precision of 74.52%, an F1 score of 71.82%, a mAP@0.5 of 73.12, and an FPS of 72. These outcomes further confirm the consistency of our method's efficiency and accuracy across different testing scenarios. In summary, by adopting the lightweight network model YOLOv8-Lite, we have not only made breakthroughs in accuracy and speed but also ensured the feasibility of the model in practical applications. Particularly in scenarios like autonomous driving, where fast and accurate object detection is required, a high FPS rate means faster response times, which is crucial for safety.

Table 5 presents a comparison of parameter counts and computational complexity for different models on the KITTI and NEXET datasets. It lists each model's number of parameters and floating-point operations (FLOPs), revealing that our proposed model demonstrates lower parameters and FLOPs on both datasets, specifically 4.80M/8.95B (KITTI) and 4.60M/8.75B (NEXET). This indicates that our model maintains high efficiency while reducing computational resource consumption. In contrast, other models like SSD, Faster-RCNN, FCOS, YOLOv5n, YOLOv7-tiny, and YOLOv8n show varying degrees of parameter counts and computational complexity. Figure 7 visualizes these comparisons, further highlighting the results and allowing us to see at a glance that our model has found an ideal balance between high-performance object detection and computational resource efficiency. This not only provides an effective solution for areas such as autonomous driving but also offers valuable insights for future efficient computing in resource-constrained environments.

4.5 Ablation study

As Table 6 demonstrates, the ablation study results compare the specific impacts of different module combinations on model performance. Through

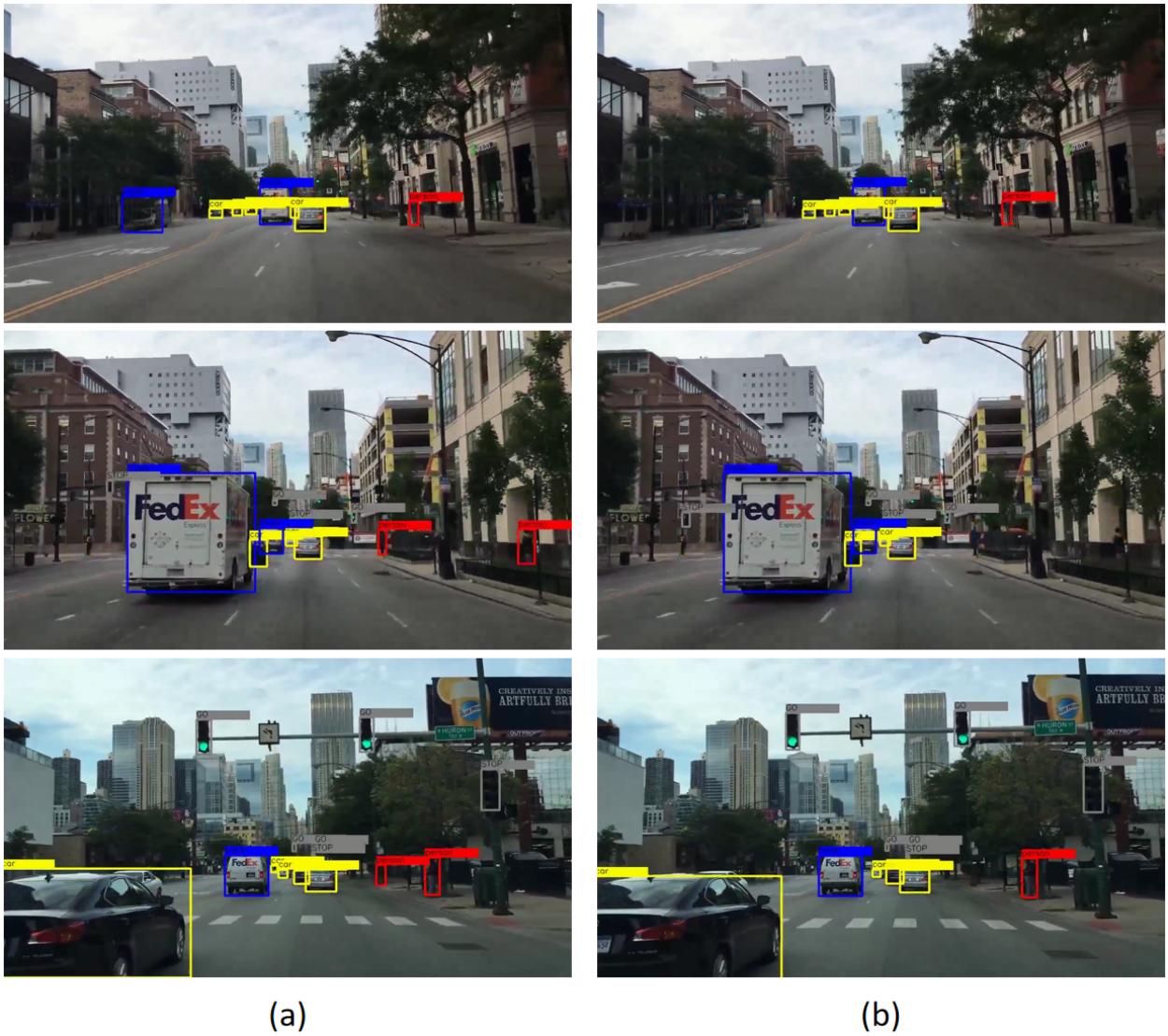

Figure 8. Visualization of detection results. (a) YOLOv8-Lite. (b) YOLOv8.

Table 5. Model parameters and computational complexity for KITTI and NEXET datasets.

Model	KITTI		NEXET	
	PARAMS	FLOPs	PARAMS	FLOPs
SSD	6.98M	9.95B	6.48M	9.15B
Faster-RCNN	6.79M	9.85B	6.19M	8.75B
FCOS	6.72M	9.65B	6.12M	7.55B
YOLOv5n	7.32M	10.25B	6.72M	9.45B
YOLOv7-tiny	7.52M	10.45B	6.92M	9.65B
YOLOv8n	7.22M	10.15B	6.62M	8.35B
Ours	4.80M	8.95B	4.60M	8.75B

detailed analysis of three core modules—CBAM, TFPN, and FastDet—the experiments reveal their individual and combined effects on precision, F1 score, mean Average Precision (mAP@0.5), and frames per second (FPS). The baseline experiment (Experiment 1) did not employ any of the new modules, while subsequent experiments progressively introduced CBAM, TFPN, and FastDet to explore the stepwise improvement in model performance. Ultimately, when all modules were integrated in Experiment (5), the model demonstrated optimal performance on both KITTI and NEXET datasets, particularly on the KITTI dataset, where precision, F1 score, mAP@0.5, and FPS were improved to 76.19%, 74.09%, 74.79%, and 85, respectively, with similar significant improvements observed on the NEXET dataset. This series of ablation studies not only verifies the importance of CBAM, TFPN, and FastDet in enhancing model performance but also provides strong evidence for designing efficient object detection models, emphasizing the effectiveness of our model in the field of autonomous driving.

4.6 Discussion

Based on the visualization results shown in Figure 8, we can observe that each detection box represents the

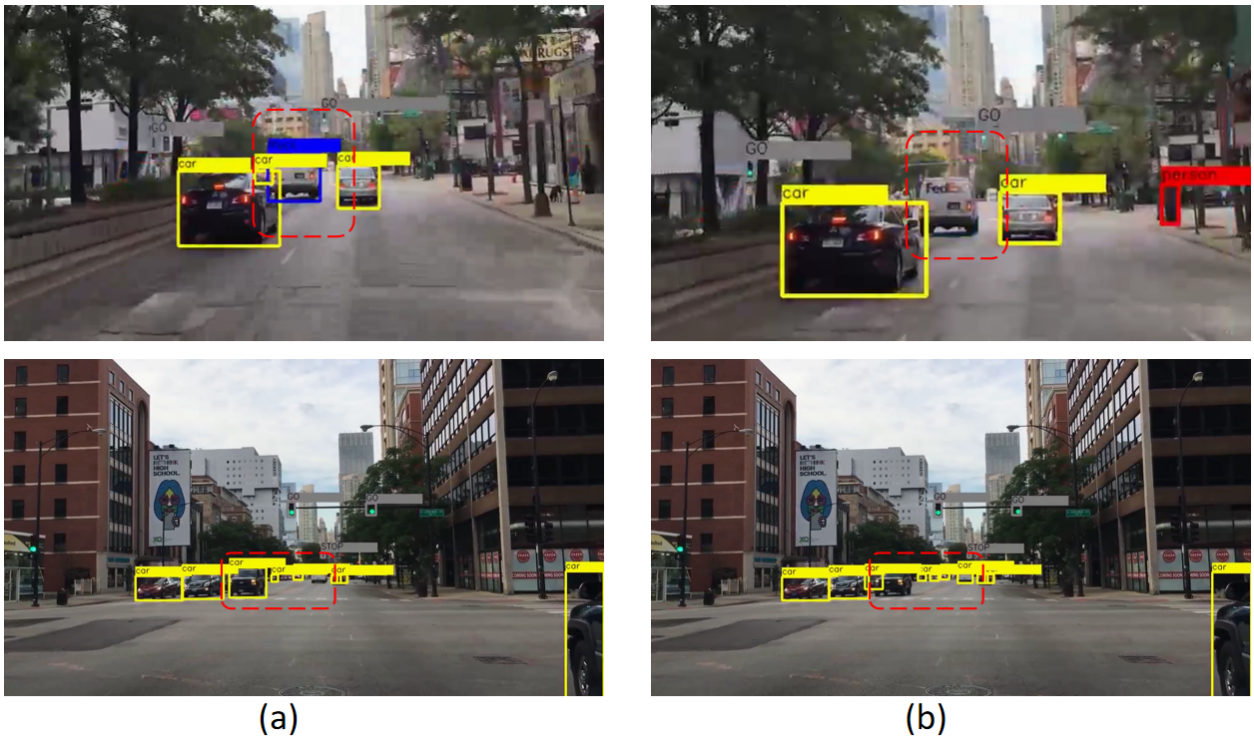

Figure 9. Comprehensive Detection Analysis. (a) YOLOv8-Lite. (b) isYOLOv8.

Figure 10. Visualization of Model Limitations.

actual predicted location of objects, with the top of the detection box displaying the predicted category and confidence score. These results demonstrate the outstanding detection performance of the proposed YOLOv8-Lite, especially in terms of recognizing small objects. Compared to the original YOLOv8 model, YOLOv8-Lite significantly improves overall precision while maintaining prediction accuracy. This achievement is primarily attributed to optimizations made in the model architecture, including the introduction of more efficient feature extraction and fusion mechanisms, as well as an optimized network structure designed to reduce computational complexity while enhancing the model's ability to recognize small objects.

Further analysis shows that our network is more comprehensive in object detection. As demonstrated in Figure 9, there are many missed detections by YOLOv8, which we have marked with red boxes. In contrast, our model effectively reduces missed detections, thereby improving the comprehensiveness and accuracy of detection. Additionally, the robustness of our model in handling complex scenes has significantly improved. In environments with dense objects, YOLOv8-Lite not only reduces missed detections but can also more accurately distinguish and locate objects that are close to each other, a task that is challenging for conventional models. This capability of our model is attributed to the in-depth optimization of the feature extraction layers and network structure, enabling the model to more effectively capture subtle feature differences and thus achieve more accurate object detection in complex environments.

However, our network also has some limitations. As

Table 6. Ablation Experiment Results on KITTI and NEXET Datasets

Method	Modules			KITTI				NEXET			
	CBAM	TFPN	FastDet	Precision	F1 Score	mAP@0.5	FPS	Precision	F1 Score	mAP@0.5	FPS
(1)				65.31	69.41	69.61	63	70.11	67.21	68.41	51
(2)	✓			66.33	70.01	71.51	70	73.11	67.81	70.31	62
(3)		✓		66.81	70.61	70.91	63	64.61	69.71	69.71	51
(4)	✓		✓	72.01	69.71	71.71	65	71.84	68.51	70.51	52
(5)	✓	✓	✓	76.19	74.09	74.79	85	73.99	71.29	72.59	72

shown in Figure 10, it is evident that in conditions of poor lighting or snowy weather, our model cannot comprehensively and accurately detect objects. This phenomenon is mainly due to the significantly reduced visibility of objects in these special environments, making it difficult for the model to capture sufficient feature information, thereby affecting the accuracy and comprehensiveness of detection.

To address this issue, future work needs to consider the introduction of more powerful feature extraction and enhancement techniques, such as using deeper network structures or introducing environment-adaptive algorithms, to improve the model's performance in complex environments. Additionally, leveraging multimodal data (such as radar and infrared images) may also be an effective means to enhance the model's robustness, as these technologies can provide additional environmental information when visual information is insufficient.

5 Conclusion

In our study, we proposed a lightweight object detection model based on the YOLOv8-Lite framework and validated it on the NEXET and KITTI datasets. Experimental results demonstrate that our model performs well in the task of object detection. This model significantly enhances efficiency and inference speed while reducing the number of parameters. Additionally, through comprehensive evaluation of various metrics, our model exhibits reliability and effectiveness in real-world applications such as autonomous driving. However, our model still has some limitations. Firstly, due to constraints in model parameters and network structure, our model may perform poorly in handling complex scenes and occlusion situations. Secondly, owing to dataset limitations and training strategy choices, the model may encounter overfitting or underfitting issues in specific scenarios, affecting its generalization ability and stability.

Looking ahead, we will continue to improve the model's design and training strategies to enhance its adaptability and robustness in complex scenes. Furthermore, we plan to expand the scale and diversity of the dataset to comprehensively evaluate the model's performance and generalization ability. Additionally, we aim to explore more advanced object detection technologies and algorithms to address the challenges in fields like autonomous driving, contributing to the development of intelligent transportation systems.

Conflicts of Interest

The authors declare no conflicts of interest.

Funding

This work was supported without any funding.

References

[1] Zhu, Y., & Yan, W. Q. (2022). Traffic sign recognition based on deep learning. *Multimedia Tools and Applications*, 81(13), 17779-17791.

[2] Du, W., Chen, L., Wang, H., Shan, Z., Zhou, Z., Li, W., & Wang, Y. (2023). Deciphering urban traffic impacts on air quality by deep learning and emission inventory. *Journal of environmental sciences*, 124, 745-757.[CrossRef]

[3] Wang, C., Wang, Y., Han, Y., Song, L., Quan, Z., Li, J., & Li, X. (2017, January). CNN-based object detection solutions for embedded heterogeneous multicore SoCs. In *2017 22nd Asia and South Pacific design automation conference* (*ASP-DAC*) (pp. 105-110). IEEE. [CrossRef]

[4] Babbar, S., & Bedi, J. (2023). Real-time traffic, accident, and potholes detection by deep learning techniques: a modern approach for traffic management. *Neural Computing and Applications*, 35(26), 19465-19479.

[5] Zhang, Y., Zhao, T., Gao, S., & Raubal, M. (2023). Incorporating multimodal context information into traffic speed forecasting through graph deep learning. *International Journal of Geographical Information Science*, 37(9), 1909-1935. [CrossRef]

[6] Sattar, K., Chikh Oughali, F., Assi, K., Ratrout, N., Jamal, A., & Masiur Rahman, S. (2023). Transparent deep machine learning framework for predicting traffic crash severity. *Neural Computing and Applications*, 35(2), 1535-1547.

[7] Zou, H., Zhan, H., & Zhang, L. (2022). Neural network based on multi-scale saliency fusion for traffic signs detection. Sustainability, 14(24), 16491. [CrossRef]

[8] Mittal, U., Chawla, P., & Tiwari, R. (2023). EnsembleNet: A hybrid approach for vehicle detection and estimation of traffic density based on faster R-CNN and YOLO models. *Neural Computing and Applications*, 35(6), 4755-4774.

[9] Chen, Z., Guo, H., Yang, J., Jiao, H., Feng, Z., Chen, L., & Gao, T. (2022). Fast vehicle detection algorithm in traffic scene based on improved SSD. *Measurement*, 201, 111655. [CrossRef]

[10] Li, X., Xie, Z., Deng, X., Wu, Y., & Pi, Y. (2022). Traffic sign detection based on improved faster R-CNN for autonomous driving. *The Journal of Supercomputing*, 1-21.

[11] Guillermo, M., Francisco, K., Concepcion, R., Fernando, A., Bandala, A., Vicerra, R. R., & Dadios, E. (2023, May). A Comparative Study on Satellite Image Analysis for Road Traffic Detection using YOLOv3-SPP, Keras RetinaNet and Full Convolutional Network. In *2023 8th International Conference on Business and Industrial Research* (*ICBIR*) (pp. 578-584). IEEE. [CrossRef]

[12] Wang, J., Li, F., An, Y., Zhang, X., & Sun, H. (2024). Towards robust lidar-camera fusion in bev space via mutual deformable attention and temporal aggregation. *IEEE Transactions on Circuits and Systems for Video Technology*. [CrossRef]

[13] Zhang, P., Yu, X., Bai, X., Wang, C., Zheng, J., & Ning, X. (2024). Joint discriminative representation learning for end-to-end person search. *Pattern Recognition*, 147, 110053. [CrossRef]

[14] Ning, E., Wang, C., Zhang, H., Ning, X., & Tiwari, P. (2024). Occluded person re-identification with deep learning: a survey and perspectives. Expert systems with applications, 239, 122419. [CrossRef]

[15] Ning, X., Yu, Z., Li, L., Li, W., & Tiwari, P. (2024). DILF: Differentiable rendering-based multi-view Image–Language Fusion for zero-shot 3D shape understanding. *Information Fusion*, 102, 102033. [CrossRef]

[16] Ning, X., He, F., Dong, X., Li, W., Alenezi, F., & Tiwari, P. (2024). ICGNet: An intensity-controllable generation network based on covering learning for face attribute synthesis. *Information Sciences*, 660, 120130. [CrossRef]

[17] Fakhrurroja, H., Pramesti, D., Hidayatullah, A. R., Fashihullisan, A. A., Bangkit, H., & Ismail, N. (2023, October). Automated License Plate Detection and Recognition using YOLOv8 and OCR With Tello Drone Camera. In *2023 International Conference on Computer, Control, Informatics and its Applications* (*IC3INA*) (pp. 206-211). IEEE. [CrossRef]

[18] Arora, N., Kumar, Y., Karkra, R., & Kumar, M. (2022). Automatic vehicle detection system in different environment conditions using fast R-CNN. *Multimedia Tools and Applications*, 81(13), 18715-18735.

[19] Soylu, E., & Soylu, T. (2024). A performance comparison of YOLOv8 models for traffic sign detection in the Robotaxi-full scale autonomous vehicle competition. *Multimedia Tools and Applications*, 83(8), 25005-25035.

[20] Othmani, M. (2022). A vehicle detection and tracking method for traffic video based on faster R-CNN. *Multimedia Tools and Applications*, 81(20), 28347-28365.

[21] Xia, J., Li, M., Liu, W., & Chen, X. (2023). Dsra-detr: An improved detr for multiscale traffic sign detection. *Sustainability*, 15(14), 10862. [CrossRef]

[22] Wei, H., Zhang, Q., Qin, Y., Li, X., & Qian, Y. (2024). YOLOF-F: you only look one-level feature fusion for traffic sign detection. *The Visual Computer*, 40(2), 747-760.

[23] Aboah, A., Wang, B., Bagci, U., & Adu-Gyamfi, Y. (2023). Real-time multi-class helmet violation detection using few-shot data sampling technique and yolov8. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 5350-5358).

[24] Liu, Z., Li, J., Song, R., Wu, C., Liu, W., Li, Z., & Li, Y. (2022). Edge guided context aggregation network for semantic segmentation of remote sensing imagery. *Remote Sensing*, 14(6), 1353. [CrossRef]

[25] Wang, X., Gao, H., Jia, Z., & Li, Z. (2023). BL-YOLOv8: An improved road defect detection model based on YOLOv8. *Sensors*, 23(20), 8361. [CrossRef]

[26] Ma, H., Yang, H., & Huang, D. (2021). Boundary guided context aggregation for semantic segmentation. *arXiv preprint arXiv:2110.14587*. [CrossRef]

[27] Unal, D., Catak, F. O., Houkan, M. T., Mudassir, M., & Hammoudeh, M. (2023). Towards robust autonomous driving systems through adversarial test set generation. ISA transactions, 132, 69-79. [CrossRef]

[28] Geiger, A., Lenz, P., & Urtasun, R. (2012, June). Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE conference on computer vision and pattern recognition* (pp. 3354-3361). IEEE. [CrossRef]

[29] Liu, Q., Ye, H., Wang, S., & Xu, Z. (2024). YOLOv8-CB: Dense Pedestrian Detection Algorithm Based on In-Vehicle Camera. *Electronics*, 13(1), 236. [CrossRef]

[30] Chen, X. (2022, October). Traffic lights detection method based on the improved yolov5 network. In *2022 IEEE 4th International Conference on Civil Aviation Safety and Information Technology* (*ICCASIT*) (pp. 1111-1114). IEEE. [CrossRef]

[31] Li, S., Wang, S., & Wang, P. (2023). A small object

detection algorithm for traffic signs based on improved YOLOv7. *Sensors*, 23(16), 7145. [CrossRef]

[32] Mohandas, P. (2023). Sad: Sensor-based anomaly detection system for smart junctions. IEEE Sensors Journal, 23(17), 20368-20378. [CrossRef]

[33] Talaat, F. M., & ZainEldin, H. (2023). An improved fire detection approach based on YOLO-v8 for smart cities. *Neural Computing and Applications*, 35(28), 20939-20954.