



Article

Improved Taillight Detection Model for Intelligent Vehicle Lane-Change Decision-Making Based on YOLOv8

Ming Li ¹, Jian Zhang ¹, Weixia Li ², Tianrui Yin ³, Wei Chen ³, Luyao Du ³, Xingzhuo Yan ³ and Huiheng Liu ^{4,*}

¹ Jiangxi Transportation Institute Co., Ltd., Nanchang 330200, China; lddwlm@163.com (M.L.); 15170063689@163.com (J.Z.)

² Jiangxi Provincial Intelligent Transportation Affairs Center, Nanchang 330003, China; 15180465785@163.com

³ School of Automation, Wuhan University of Technology, Wuhan 430070, China; 320973@whut.edu.cn (T.Y.); greatchen@whut.edu.cn (W.C.); duluyao@whut.edu.cn (L.D.); brianbox@whut.edu.cn (X.Y.)

⁴ School of Physics and Electronic Engineering, Hubei University of Arts and Science, Xiangyang 441053, China

* Correspondence: lhh@hbuas.edu.cn

Abstract: With the rapid advancement of autonomous driving technology, the recognition of vehicle lane-changing can provide effective environmental parameters for vehicle motion planning, decision-making and control, and has become a key task for intelligent vehicles. In this paper, an improved method for vehicle taillight detection and intent recognition based on YOLOv8 (You Only Look Once version 8) is proposed. Firstly, the CARAFE (Context-Aware Reassembly Operator) module is introduced to address fine perception issues of small targets, enhancing taillight detection accuracy. Secondly, the TriAtt (Triplet Attention Mechanism) module is employed to improve the model's focus on key features, particularly in the identification of positive samples, thereby increasing model robustness. Finally, by optimizing the EfficientP2Head (a small object auxiliary head based on depth-wise separable convolutions) module, the detection capability for small targets is further strengthened while maintaining the model's practicality and lightweight characteristics. Upon evaluation, the enhanced algorithm demonstrates impressive results, achieving a precision rate of 93.27%, a recall rate of 79.86%, and a mean average precision (mAP) of 85.48%, which shows that the proposed method could effectively achieve taillight detection.

Keywords: intelligent vehicle; taillight detection; lane-changing recognition; deep learning



Citation: Li, M.; Zhang, J.; Li, W.; Yin, T.; Chen, W.; Du, L.; Yan, X.; Liu, H. Improved Taillight Detection Model for Intelligent Vehicle Lane-Change Decision-Making Based on YOLOv8. *World Electr. Veh. J.* **2024**, *15*, 369. <https://doi.org/10.3390/wevj15080369>

Academic Editor: Guoxing Bai

Received: 14 July 2024

Revised: 4 August 2024

Accepted: 13 August 2024

Published: 15 August 2024



Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

In recent years, the rapid increase in global vehicle ownership has dramatically heightened the complexity of traffic scenarios, posing severe challenges to driving safety. Intelligent driving is considered as a key innovation to transform transportation [1], aiming to enhance driving safety and efficiency. In the process of vehicle motion planning and control, drivers need to acquire abundant information to perform lane-changes, turns, and overtaking maneuvers. Among these semantic cues, vehicle taillights are the most commonly used features, allowing drivers to infer the turning and lane-changing intentions of the preceding vehicle. Additionally, various combinations of front vehicle taillights can convey driving intentions to the following vehicles. Therefore, visual perception is an important auxiliary tool for drivers to plan and control their movements, which helps to ensure safe driving and reduce traffic accidents. Against this backdrop, vision-based vehicle lane-change detection technology has become a critical component of intelligent driving systems. Given the large number of motor vehicles and the complex traffic environment, improving the safety of vehicle motion planning and control has become an urgent research topic for intelligent vehicles. To reduce accident rates, vehicles must provide timely warnings before accidents occur, and even autonomously intervene in advance. Therefore, the recognition of vehicle lane-changing is crucial in vehicle motion planning and control, making it an important research hotspot.

Computer vision, as a significant application area of deep learning, encompasses several key technologies, such as semantic segmentation and object detection. The advancement of these technologies provides robust visual perception capabilities for assisted driving systems. However, current vehicle lane-change detection still faces challenges, particularly in highly dynamic and complex traffic scenarios. This paper aims to improve vehicle lane-change detection performance through deep learning techniques. Taillight detection and dynamic object detection are key tasks of interest in this study. By integrating the information from these tasks, this paper aims to develop an efficient fusion detection model, offering more comprehensive and accurate lane-change judgments for assisted driving systems. Utilizing visual perception technology, combining taillight signals and lane information enables a more thorough analysis of the vehicle's surrounding environment, enhancing the accuracy of lane-change intent detection and effectively assisting drivers in avoiding traffic accidents. This paper aims to systematically study taillight detection methods to construct a reliable and precise vehicle lane-change fusion detection system. The contributions of this paper can be summarized as follows.

1. To address issues such as image distortion and high computational load, this paper introduces the CARAFE lightweight universal upsampling operator. By utilizing the upsampling kernel prediction module and the content-aware feature reassembly module, this method enhances the fine-grained perception of small objects, thereby improving the accuracy of taillight detection.
2. By incorporating attention mechanisms, this paper enables the network to selectively focus on taillight information while disregarding irrelevant background details, thus better adapting to the task of vehicle taillight detection. The introduction of the TriAtt triple attention module enhances the model's ability to capture the intrinsic features of taillight data, thereby improving the accuracy of the model in taillight detection tasks.
3. In addressing complex vehicle taillight detection scenarios, this study enhances the robustness of the model by modifying the original detection head. By removing the channel compression step from the native detection head and introducing the EfficientP2Head module, the model's accuracy and robustness in detecting small targets within taillight detection tasks are significantly improved. This modification effectively enhances the convolutional layer's ability to capture feature information.

The structure of the paper is as follows. The first section provides an overview of the background of taillight detection models for intelligent vehicle lane-changing recognition. The second section reviews the existing literature on lane-change intention recognition, highlighting the strengths and limitations of various approaches. In the third section, an improved YOLOv8 algorithm with three enhancements is proposed. The fourth section presents the experimental validation of the proposed algorithm in real-world scenarios, along with an in-depth analysis of the results. The final section offers a comprehensive summary of the study.

2. Related Works

In the field of vehicle taillight detection, researchers have employed various methods, primarily including traditional computer vision techniques, deep learning approaches, and hybrid methods combining traditional vision with deep learning.

In 2010, the authors of [2] proposed a taillight image recognition rule and method based on the Sobel operator, targeting the color and spatial distribution characteristics of six types of taillights. However, the Sobel operator exhibited instability in complex scenes, particularly under varying lighting conditions and high noise interference, potentially reducing the detection performance of taillight signals. The authors of [3] successfully achieved the localization and identification of vehicle taillights by analyzing the combination of illumination, size, and color gamut parameters. This method utilized the unique color space characteristics of taillights, employing a combination of chroma and illumination threshold ratios to identify taillights. However, it was susceptible to ambient light and

other interfering light sources, leading to ambiguous recognition and high false detection rates, with poor generalizability limited to specific scenarios.

In 2014, the authors of [4] introduced a method for nighttime front vehicle recognition based on monocular vision, aiming to accurately locate and identify various nighttime driving vehicles through taillight features. The core idea of this method was to use the distribution characteristics of the HSV color space to set specific color thresholds, effectively separating front vehicle taillights from the background image. While this method could exclude falsely detected vehicles and estimate missed ones, it was highly sensitive to lighting changes, resulting in limited application scenarios and a poor robustness.

In 2015, Zhang Jun et al. [5] proposed a taillight shape method based on halo hierarchy, addressing the issue of misdetection caused by environmental interference in color gamut and size-based machine learning methods. This method had the advantages of real-time processing, high accuracy, and low false detection rates. However, in complex nighttime traffic scenes, especially those with multiple light sources, it still had certain limitations. Almagambetov A et al. [6] presented a lightweight algorithm based on an embedded smart camera for real-time vehicle taillight tracking and warning signal detection. This algorithm utilized soft color thresholds for preprocessing, and tracked taillights using a Kalman filter and codebook, achieving vehicle count statistics. It effectively tracked taillights and detected warning signals both at night and during the day, but there was still room for improvement in counting surrounding vehicles.

In 2016, Tian Qiang et al. [7] introduced a new method for interpreting vehicle taillight signal transmission information. This method accurately located and continuously tracked taillights using their unique color characteristics and symmetrical distribution, followed by parameter optimization using the least squares support vector machine (LS-SVM) for the accurate recognition of taillight semantic information. However, this recognition algorithm exhibited significant missed detections and mismatches in practical applications, heavily relying on the quality of taillight detection. Inspired by multi-sensor fusion strategies, Jin Lisheng et al. [8] proposed a method combining machine vision and radar waves, establishing a conversion relationship between millimeter-wave radar world coordinates and visual image coordinates to form a region of interest for the front vehicle. Image processing techniques effectively reduced interference noise in the region of interest, and the Dempster–Shafer evidence theory was used to fuse feature information, yielding vehicle detection results meeting confidence thresholds. Although this method was robust to various vehicle shapes, it failed to perform well in dense and congested scenes. Chien C L et al. [9] addressed the issue of red taillight detection under overexposure by combining morphological and logical operations to effectively extract overexposed areas, ensuring more reliable taillight detection. This method could effectively detect almost all vehicles under non-rainy nighttime driving conditions, but had limitations in range estimation for single-taillight vehicles, such as motorcycles.

In 2017, Vancea F I et al. [10] proposed a method combining vehicle detection and taillight segmentation for taillight detection and tracking in daytime scenes. This method first performed vehicle detection, followed by identifying taillight candidate regions within the vehicle. Two methods were used for detecting taillight candidate regions—explicit thresholding to extract red regions and deep learning for taillight segmentation. To reduce false alarms, Kalman filtering was introduced for taillight tracking. While this method reliably detected and tracked vehicle taillights in daytime scenes, its performance was suboptimal in specific scenarios, such as complex traffic conditions.

In 2019, Li Xiang [11] replaced traditional vision algorithms with deep learning detection algorithms, choosing the YOLOv3-tiny (You Only Look Once version 3) network as the base framework. He introduced the SPP (Spatial Pyramid Pooling) pyramid pooling network structure and the focal loss function to enhance the continuous real-time detection of vehicles and taillight signals in videos. However, there is still significant room for optimization in terms of parameter count and detection speed. Li Guojie [12] proposed and implemented a taillight detection algorithm based on deep learning. This algorithm, built

on convolutional neural networks and using a regression-based structure, effectively fused high-level and low-level features to detect multi-scale feature information and enhance the robustness of small target feature capture. Additionally, this network was capable of real-time tracking and capturing the taillight features of the preceding vehicles and interpreting the signal information contained therein. Nevertheless, the algorithm exhibited a lower recall rate for distant vehicles and showed instability in taillight classification when dealing with large vehicles and vehicles with occlusions. Vancea F I et al. [13] addressed the problem of vehicle taillight detection in collision prevention systems and autonomous driving by proposing a method based on semantic information for vehicle relative positioning and taillight detection. They utilized Faster RCNN (Faster Region-Convolutional Neural Network) for vehicle detection and classification, introduced a sub-network for taillight pixel segmentation, and employed ERFNet (Efficient Residual Factorized ConNet) for semantic segmentation. However, there is room for improvement in increasing the taillight segmentation IoU (Intersection over Union), expanding the number of training images, and using feature pyramid networks (FPNs). Gao F et al. [14] tackled the issue of vehicle detection at nighttime at multi-car intersections by proposing a saliency detection-based method. They used the frame difference method to detect moving objects, extracted vehicle taillights using saliency maps and color information, and achieved vehicle detection by matching taillight pairs. This method effectively met the real-time requirements of nighttime vehicle detection. However, there are some shortcomings in complex nighttime scenarios, particularly with buses, such as confusion between taillights and red route indicators, and the wide spacing of taillights on the same vehicle, which require further research to address.

In 2022, Liu Jingkai [15] proposed a technique for recognizing and interpreting the signal information of preceding vehicle taillights based on deep learning and color gamut space. This method first utilizes the YOLOv3 model to locate and capture the features of the preceding vehicle's taillights, subsequently classifying the signal information contained in the taillights into five driving states—braking, turning, warning, passing, and others. Furthermore, Liu Jingkai designed a method based on color space to identify whether the preceding vehicle's taillights are illuminated, which demonstrates good robustness under different weather conditions. However, both methods exhibit certain limitations in robustness in nighttime scenes and extreme weather conditions. Li Q [16] proposed a real-time vehicle taillight detection method based on an improved YOLOv3-tiny model. This method enhances the accuracy of taillight detection by adding output layers, introducing a spatial pyramid pooling module, and employing focal loss. Experimental results showed that significant improvements in the detection of brake lights, left turn signals, and right turn signals. Additionally, a taillight detection dataset was constructed, providing strong support for research in this field. However, issues such as the confusion between taillights and red route indicators, and the wide spacing between two taillights on the same bus, persist when dealing with scenarios involving buses at night.

In recent years, Parvin S [17] addressed the issue of nighttime vehicle detection in intelligent transportation systems by proposing a real-time detection and tracking system based on taillight and headlight features. This method utilizes computer vision and image processing techniques to identify vehicles through taillight and headlight features, employing a centroid tracking algorithm for vehicle tracking. Two flexible regions of interest (ROIs) are used for the headlight and taillight, respectively, to accommodate different image and video resolutions. The method reliably and effectively detects and tracks vehicles in nighttime environments, including both dual-light and single-light (e.g., motorcycle light) vehicles. Despite its outstanding performance in nighttime vehicle detection, further improvements are needed for vehicle detection in complex road scenarios and under conditions such as fog. Jeon H J et al. [18] proposed a high-accuracy, high-speed vehicle taillight detection model based on deep learning. This model includes three main modules—lane detection, vehicle detection, and taillight detection. Using a data-driven approach, the lane and taillight detection modules utilize a Recurrent Rolling Convolution (RRC)

architecture. However, the model is limited to taillight detection for passenger cars and does not consider other vehicle types, indicating certain limitations. Additionally, the model's detection methods need further enhancement for performance in extreme weather conditions. Oh G [19] proposed a method for detecting moving vehicles and their brake light states. This method employs a one-stage brake light state detection network based on YOLOv8, achieving the accurate detection of moving vehicles and brake light states through transfer learning on a custom dataset. The researchers also specifically constructed a dataset for this task and conducted comprehensive evaluations of the method's performance on edge devices, achieving significant results in high accuracy and short inference times. However, the model's generalization ability needs improvement for handling other vehicle types and performing in complex scenarios.

3. Improved Taillight Detection Model

In this paper, YOLOv8 is selected as the benchmark detection model [20]. Compared to previous versions, YOLOv8 incorporates design inspirations from earlier object detection algorithms, including YOLOX (You Only Look Once version X), YOLOv6, YOLOv7, and PPYOLOE (PaddlePaddle YOLO Enhanced). The chosen benchmark model, YOLOv8, demonstrates significant improvements in speed and accuracy and provides a unified framework that supports various tasks such as object detection, instance segmentation and image classification. The algorithm framework is illustrated in Figure 1.

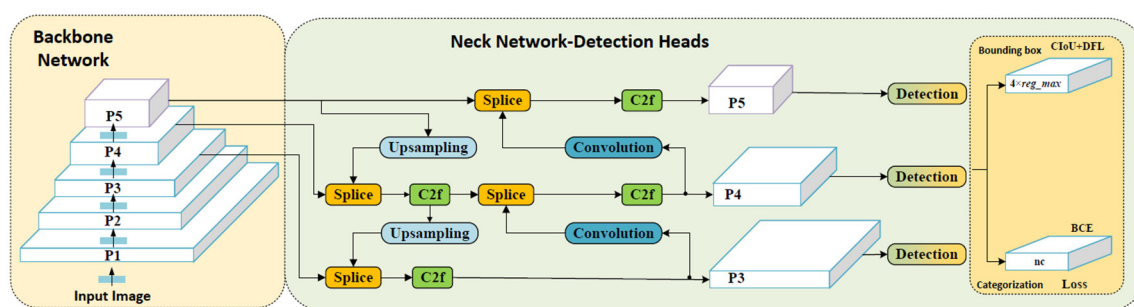


Figure 1. The algorithm framework of YOLOv8.

Due to the relatively small size of taillights, the performance of the original YOLOv8 model may degrade when processing such small targets. Additionally, YOLOv8 still has room for improvement in capturing feature information and recognizing small targets. Therefore, this study optimizes YOLOv8 by introducing three improvements—an upsampling operator at Section 3.1, an attention mechanism at Section 3.2, and an optimized detection head at Section 3.3. These enhancements are designed to increase the model's accuracy and efficiency in taillight detection tasks, making it more suitable for complex real-world scenarios and improving the overall performance in taillight detection.

3.1. Context-Aware Reassembly Operator (CARAFE): An Improved Characteristic Upsampling Operator

Feature upsampling is a critical operation in convolutional network architectures, and its design is essential for dense prediction tasks. Upsampling operations are widely used in various network structures, as illustrated in Figure 2. However, traditional upsampling methods suffer from issues such as image distortion caused by nearest-neighbor interpolation, feature map blurring due to transposed convolutions, and the significant computational load introduced by bilinear interpolation.

To overcome the limitations of traditional upsampling methods, this paper introduces a lightweight and versatile upsampling operator called CARAFE (Context-Aware Reassembly Operator) [21]. CARAFE achieves improvements across various tasks with minimal additional parameters and computational cost; it consists of two main modules—the kernel

prediction module and the content-aware feature reassembly module, as illustrated in Figure 3.

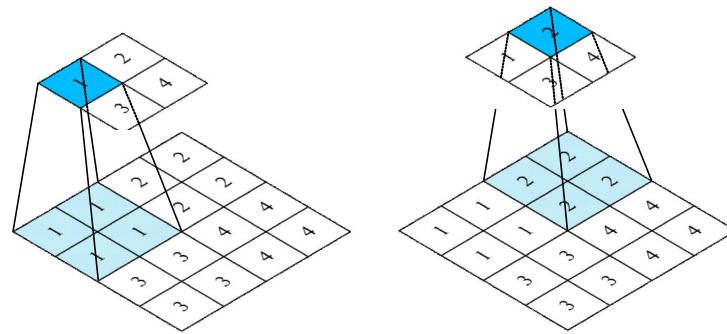


Figure 2. Schematic diagram of upsampling operation.

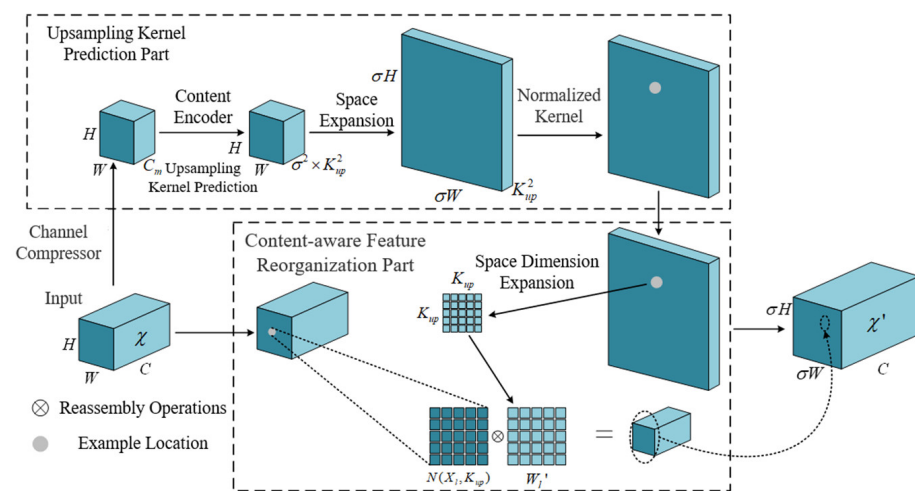


Figure 3. The overall framework diagram of CARAFE, where C is the channel dimension, H is the height of the input feature map, W is the width of the input feature map, σH is the height of the input feature map after upsampling operation and σW is the width of the input feature map after upsampling operation.

The upsampling kernel prediction module generates a reassembly kernel in real-time for each position based on the target. For an input feature map of size $H \times W \times C$ it compresses the channel dimension to C_m using a 1×1 convolution, resulting in a feature map of size $H \times W \times C_m$. This method is similar to the channel attention mechanism, reducing the parameter operations within the module. Assuming the size of the convolutional kernel for the upsampling operation is $k_{encoder} \times k_{encoder}$ and the upsampling ratio is σ , CARAFE performs a convolution operation to change the output channels from C_m to $C_{up} = \sigma^2 k_{up}^2$, achieving content encoding and obtaining an upsampling kernel of size $H \times W \times C_{up}$. Experimental results demonstrate that the model achieves an ideal balance between performance and computational complexity when this specific relationship $k_{encoder} = k_{up} - 2$ is satisfied. After spatial dimension expansion, the predicted upsampling kernel is flattened and rearranged along the channel dimension, resulting in a size of $\sigma H \times \sigma W \times k_{up} \times k_{up}$. Finally, the softmax function is applied to normalize the upsampling kernel across channels, ensuring the sum of the convolution kernel weights equals 1. The predicted upsampling kernel is then input into the feature reassembly part.

On the other hand, in the feature reassembly module, each position in the output feature map X' is mapped to the corresponding position in the input feature map X , selecting the original $k_{up} \times k_{up}$ feature map region centered at that position. Subsequently, a dot product operation is performed with the predicted upsampling kernel at that position,

ensuring that different channels at the same location share the same upsampling kernel. The final result is a new feature map with dimensions $\sigma H \times \sigma W \times C$.

For any target position $l' = (i', j')$ in the output feature map X' , there exists a corresponding source position $l = (i, j)$ in the input feature map X , as described using Equation (1). Here, $N(X_l, k)$ represents the $k \times k$ subregion of the input feature map X centered at position l .

$$i = \frac{i'}{\sigma}, j = \frac{j'}{\sigma} \quad (1)$$

Accordingly, the kernel prediction module ψ predicts the position kernel $X_{l'}$ for each location l' based on X_l , as shown in Equation (2):

$$W_{l'} = \psi(N(X_l, k_{encoder})) \quad (2)$$

For the feature reassembly module ϕ , it reassembles X_l with the reassembly kernel $W_{l'}$, as shown in Equation (3):

$$X'_{l'} = \phi(N(X_l, k_{up}), W_{l'}) \quad (3)$$

After passing through the kernel prediction module and the feature reassembly module, the computation formula of CARAFE is as shown in Equation (4):

$$CARAFE = \left[\text{softmax}(\sigma H \times \sigma W \times k_{up}^2) \right] \otimes N(X'_{l'}, k_{up}) \quad (4)$$

The parameter computation of this lightweight universal upsampling operator is given in Equation (5):

$$Params_{CARAFE} = 2(C_{in} + 1)C_m + 2\left(C_m k_{encoder}^2 \sigma^2 k_{encoder}^2 + 1\right) \sigma^2 k_{up}^2 + 2\sigma^2 k_{up}^2 C_{in} \quad (5)$$

The pseudocode for implementing this algorithm is shown in Algorithm 1:

Algorithm 1: CARAFE lightweight universal upsampling operator

Input: X is the dimensions of the input feature map, $h \times w \times c \times b$, where c is the number of channels, h is the height, w is the width, and b is the batch size.

Output: X is the dimensions of the output feature map, c is the encoder convolution kernel size, K_{enc} , K_{up} , c_{mid} , $scale = 2$

```

1:  $W \leftarrow \text{Comp}(X) = \text{Conv}(X, c_{mid})$ 
2:  $W \leftarrow \text{Enc}(W) = \text{Conv}(W, (k_{up} \times scale)^2, k = k_{enc}, act = \text{False})$ 
3:  $W \leftarrow \text{PixelShuffle}(W, scale)$ 
4:  $X \leftarrow \text{Softmax}(W, dim = 1)$ 
5:  $X \leftarrow \text{Upsample}(X, scale, mode = 'nearest')$ 
6:  $X \leftarrow \text{Unfold}(X, k_{up}, dilation = scale, padding = \frac{k_{up}}{2} \times scale)$ 
7:  $X \leftarrow \text{Reshape}(X, (b, c, -1, h', w'))$ 
8:  $X \leftarrow \text{Einsum}('bkhw, bckwh \rightarrow bchw', W, W)$ 
9: return  $X$ 

```

The initial convolution operation, denoted as $\text{Conv}(X, c_{mid})$, is a convolution operation with output channels c_{mid} . The subsequent convolution, $\text{Conv}(W, (k_{up} \times scale)^2)$, is another convolution operation. This is followed by the $\text{PixelShuffle}(W, scale)$ operation, which rearranges the tensor elements. The $\text{Softmax}(W, dim = 1)$ function along the channel dimension applies is applied. The $\text{Upsample}(X, scale, mode = 'nearest')$ function upsamples the input tensor. The $\text{Unfold}(X, k_{up}, dilation = scale, padding = \frac{k_{up}}{2} \times scale)$ operation extracts sliding local blocks from a batched input tensor. The reshaping operation $\text{Reshape}(X, (b, c, -1, h', w'))$ reshapes the tensor. Finally, the $\text{Einsum}('bkhw, bckwh \rightarrow bchw', W, W)$ operation is an Ein-

stein summation convention for tensor operations, effectively performing a batch matrix multiplication.

3.2. Introducing the Triplet Attention Mechanism (TriAtt)

To enhance detection performance, this paper introduces attention mechanisms to enable the network to selectively focus on taillight information while disregarding irrelevant background data, thereby improving its suitability for vehicle taillight detection tasks. Common attention mechanisms include channel attention, spatial attention, and mixed-domain attention, as exemplified by the dual attention network (DANet) and the convolutional block attention module (CBAM) [22].

Although CBAM's channel attention significantly improves performance, it does not account for cross-dimensional interactions and involves dimensional reduction, making channel attention calculations redundant. To address these issues, this paper proposes a more lightweight and efficient triplet attention mechanism. The architecture of triplet attention, as shown in Figure 4, consists of three parallel branches responsible for different tasks. The first and second branches achieve cross-dimensional interactions between the channel layer (C) and the spatial layers (H and W), while the third branch captures spatial attention through the spatial attention module (SAM). Finally, the outputs from all branches are aggregated through weighted averaging via pooling, integrating information across different dimensions.

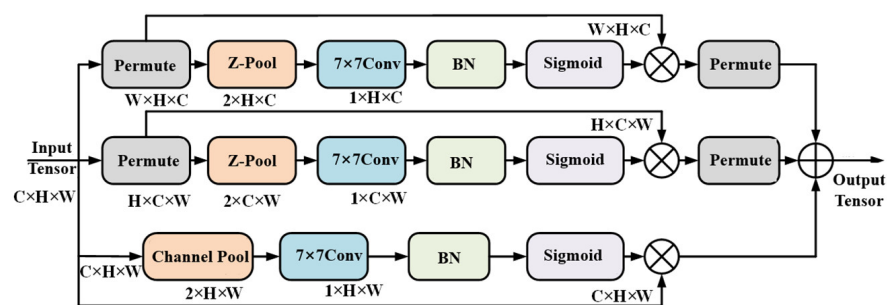


Figure 4. Diagram of the triplet attention mechanism structure, where Z-Pool represents ponding layer, Conv represents convolution, BN represents normalization layer.

The Z-pool layer in triplet attention is designed to reduce computational burden by downscaling the tensor along the C dimension. This implementation includes performing both average and max pooling on the features in the C dimension, followed by concatenating these pooled features. The purpose of this design is to retain the rich representation of the original tensor, while enhancing computational efficiency through dimensionality reduction. The expression is shown in Equation (6):

$$\text{Z-pool}(\chi) = [\text{MaxPool}_{0d}(\chi), \text{AvgPool}_{0d}(\chi)] \quad (6)$$

where $0d$ represents the 0th dimension, indicating that both max pooling and average pooling operations interact across the 0th dimension. For instance, an input tensor with the shape $C \times H \times W$ will be transformed into a tensor with the size $2 \times H \times W$ after passing through the Z-pool layer.

The first branch is responsible for establishing interactions between the H dimension and the C dimension. As shown in Figure 5, by rotating the input tensor χ counterclockwise by 90 degrees along the H axis, we obtain the rotated tensor $\hat{\chi}_1$, which has a shape of $H \times W \times C$. Subsequently, the tensor $\hat{\chi}_1^*$, which has dimensions $2 \times W \times C$ after Z-pooling, is processed through convolutional and batch normalization layers to generate an intermediate output with dimensions of $1 \times W \times C$. The final output is obtained by multiplying the attention weights generated by the sigmoid function with the input tensor χ , preserving its shape while being rotated clockwise by 90 degrees along the H axis.

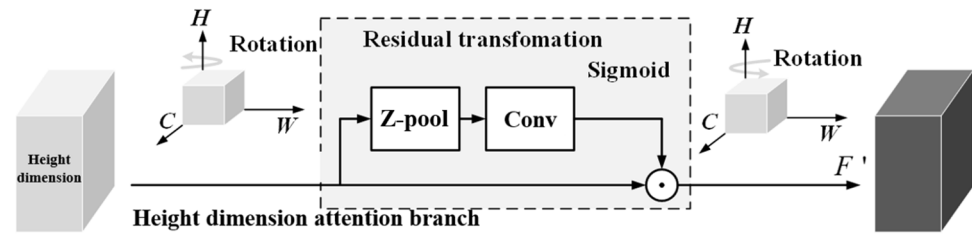


Figure 5. Diagram of the first branch of the triplet attention mechanism.

The attention computation process of this branch is illustrated using Equation (7), where F' denotes the output feature vector in the channel dimension (C, H). This branch vector is used to capture the dimensional dependencies between the height and channel of the input tensor.

$$F' = M_{(C,H)}(F) \otimes F = \hat{\chi}_1 \sigma(\psi_1(\hat{\chi}_1^*)) \quad (7)$$

The second branch introduces a rotation operation to facilitate interaction between the C dimension and the W dimension. As shown in Figure 6, by rotating the input tensor χ counterclockwise by 90 degrees along the W axis, we obtain the rotated tensor $\hat{\chi}_2$, which has a shape of $H \times C \times W$. The tensor $\hat{\chi}_2^*$, after Z-pool operation, has a shape of $2 \times C \times W$. Through standard convolutional layers and batch normalization layers, a dimension of $1 \times C \times W$ is generated as the intermediate output. Multiplying the attention weights generated by the sigmoid function with the input tensor χ , the final output is obtained, maintaining consistency in shape by rotating 90 degrees clockwise along the W axis.

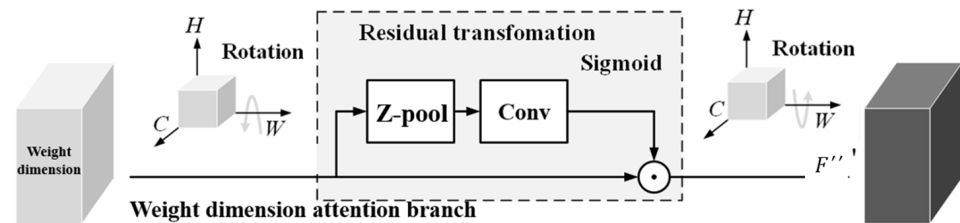


Figure 6. Diagram of the second branch of the triplet attention mechanism.

The attention computation process of this branch is illustrated using Equation (8), where F'' denotes the output feature vector in the channel dimension (C, W). This branch vector is used to capture the dimensional dependencies between the weight and channel of the input tensor.

$$F'' = M_{(C,W)}(F) \otimes F = \hat{\chi}_2 \sigma(\psi_2(\hat{\chi}_2^*)) \quad (8)$$

The third branch achieves an interaction between the H and W dimensions. As shown in Figure 7, by applying Z-pool, the channels of the input tensor χ are reduced to 2. After passing through a standard convolutional layer with a kernel size of $k \times k$, a simplified tensor with a shape of $2 \times H \times W$ is obtained. Through a batch normalization layer, the output is processed by a sigmoid activation layer to generate an attention weight $\hat{\chi}_3$ with a shape of $1 \times H \times W$.

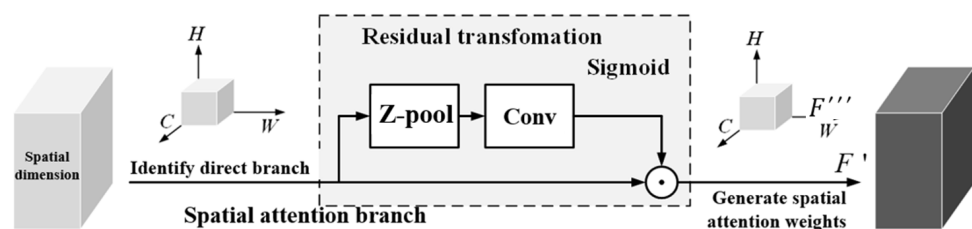


Figure 7. Diagram of the third branch of the triplet attention mechanism.

The attention calculation process for this branch is shown in Equation (9). F''' represents the output feature vector extracted using the (H, W) spatial dimension attention mechanism. This branch vector is used to capture the spatial dimension dependencies of the input tensor.

$$F''' = M_{(H,W)}(F) \otimes F = \sigma \left\{ f^{7 \times 7} ([AvgPool(F); MaxPool(F)]) \right\} = \sigma \left\{ f^{7 \times 7} \left(\begin{bmatrix} F_{avg}^S \\ F_{max}^S \end{bmatrix} \right) \right\} = \chi \sigma(\psi_3(\hat{\chi}_3)) \quad (9)$$

By averaging the fine tensors $C \times H \times W$ generated by the three branches, the final output tensor is formed. The specific calculation is shown in Equation (10):

$$y = \frac{1}{3}(F' + F'' + F''') = \frac{1}{3} \left(\overline{\hat{\chi}_1 \sigma(\psi_1(\hat{\chi}_1^*))} + \overline{\hat{\chi}_2 \sigma(\psi_2(\hat{\chi}_2^*))} + \chi \sigma(\psi_3(\hat{\chi}_3)) \right) = \frac{1}{3} (\overline{\hat{\chi}_1 \omega_1} + \overline{\hat{\chi}_2 \omega_2} + \chi \omega_3) \quad (10)$$

$$= \frac{1}{3} (\overline{y_1} + \overline{y_2} + y_3)$$

The pseudocode for the triplet attention mechanism algorithm is presented in Algorithm 2. The `PermuteAndContiguous(x, (0, 2, 1, 3))` operation is the permutation and contiguity operation. The `cw.AttentionGate(x_perm1)` and `hc.AttentionGate(x_perm2)` operations are both operations of `hc.AttentionGate(x_perm2)`.

Algorithm 2: Triplet attention mechanism

Input: X —Input feature map, `no_spatial`—indicates whether to enable the spatial attention mechanism

Output: Output feature map after triplet attention processing

```

1:  function TripletAttention(x, no_spatial)
2:    cw ← AttentionGate()
3:    hc ← AttentionGate()
4:    if not no_spatial then
5:      hw ← AttentionGate()
6:    end if
7:    x_perm1 ← PermuteAndContiguous(x, (0, 2, 1, 3))
8:    x_out1 ← cw.AttentionGate(x_perm1)
9:    x_out11 ← x_out1.permute(0, 2, 1, 3).contiguous()
10:   x_perm2 ← PermuteAndContiguous(x, (0, 3, 2, 1))
11:   xinyax_out2 ← hc.AttentionGate(x_perm2)
12:   xinyax_out22 ← x_out2.permute(0, 3, 2, 1).contiguous()
13:   if not no_spatial then
14:     x_out ← hw.AttentionGate(x)
15:     x_out ← 1/3 × (x_out + x_out11 + x_out22)
16:   else
17:     x_out ← 1/2 × (x_out11 + x_out22)
18:   end if
19:   return x_out

```

3.3. Introducing a Small Object Auxiliary Head Based on Depth-Wise Separable Convolutions (EfficientP2Head)

To enhance the robustness of the model, this study proposes modifications to the original detection head. By removing the channel compression step in the native detection head, the ability of the convolutional layers to capture feature information is effectively improved. However, this modification significantly increases the computational load. To maintain high detection accuracy while avoiding redundant computations, we introduce depth-wise separable convolutions [23] to reconstruct the detection head. This approach optimizes the parameter computation logic of the native detection head, making it more lightweight, while preserving its high-performance efficiency. The structure of the improved detection head is illustrated in Figure 8.

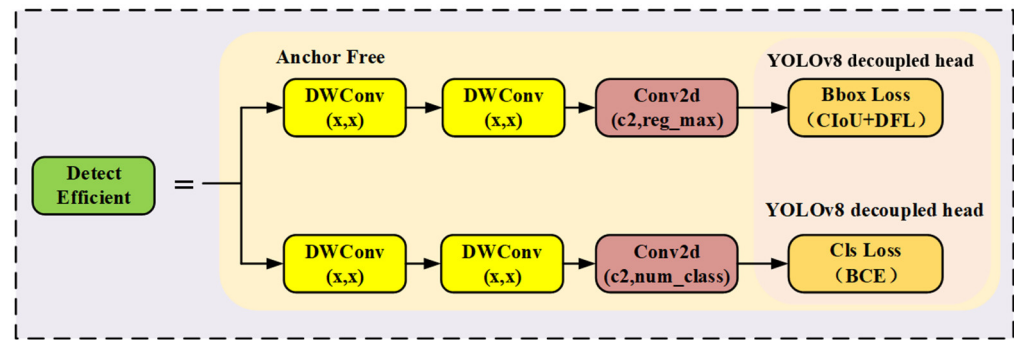


Figure 8. Structure of the improved detection head, where DWConv represents Depth-wise Convolution and Conv2d represents 2 dimension convolutional layer.

In detect efficient, depth-wise separable convolutions are utilized to reduce network parameters and enhance computational efficiency [24]. The core idea behind this technique is to decompose a standard convolution operation into two separate steps—Depth-wise Convolution (DWConv) and Pointwise Convolution (PWConv). The detailed structure of this convolution is depicted in Figure 9.

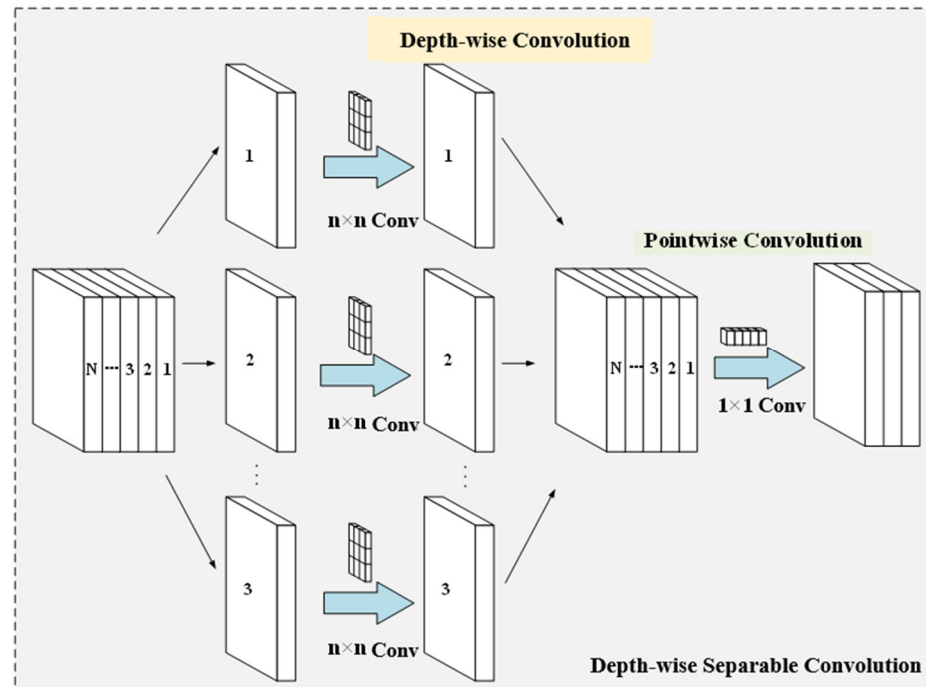


Figure 9. Diagram of the depth-wise separable convolution module.

Considering a standard convolution, given an input tensor X with size $D_F \times D_F \times M$, when using a convolution kernel of size $D_K \times D_K \times M$ with N filters, the parameter computation for a standard convolution is defined as follows in Equation (11):

$$\begin{cases} GFLOPs_{Conv} = D_K \times D_K \times M \times D_g \times D_g \times N \\ Params_{Conv} = D_K \times D_K \times M \times N \end{cases} \quad (11)$$

A depth-wise separable convolution consists of a DWConv and a PWConv. Given the same input as a standard convolution, it achieves the same output through a two-step sequential operation but with a reduced computational cost. When the input tensor X has a size $D_F \times D_F \times M$, and a convolution kernel size of $D_K \times D_K \times 1$ is used, the parameter computation for the depth-wise separable convolution is defined as follows in Equation (12):

$$\begin{cases} GFLOPS_{Depthwise} = D_K \times D_K \times 1 \times D_g \times D_g \times M \\ Params_{Depthwise} = D_K \times D_K \times M \times 1 \end{cases} \quad (12)$$

After the depth-wise convolution, a tensor of size $D_g \times D_g \times M$ is obtained. This tensor is then processed through a pointwise convolution (1×1 size, depth M convolution kernel) to achieve the desired depth tensor. Assuming there are N pointwise convolution kernels, the parameter computation is defined as follows in Equation (13):

$$\begin{cases} GFLOPS_{Pointwise} = M \times 1 \times 1 \times D_g \times D_g \times N \\ Params_{Pointwise} = N \times M \times 1 \times 1 \end{cases} \quad (13)$$

Since a depth-wise separable convolution is composed of a Depth-wise Convolution and a Pointwise Convolution in series, the final parameter computation for a depth-wise separable convolution is given by Equation (14). Ultimately, the depth-wise separable convolution performs the convolution operation with a significantly lower computational overhead, while achieving the same computational objectives. The parameters saved by using a depth-wise separable convolution are shown in Equations (15) and (16).

$$\begin{cases} Params_{DSCnv} = D_K \times D_K \times 1 \times D_g \times D_g \times M \\ \quad + M \times 1 \times 1 \times D_g \times D_g \times N \\ Params_{DSCnv} = D_K \times D_K \times M \times 1 + N \times M \times 1 \times 1 \end{cases} \quad (14)$$

$$\frac{Params_{DSCnv}}{Params_{Conv}} = \frac{D_K \times D_K \times M \times 1 + N \times M \times 1 \times 1}{D_K \times D_K \times M \times N} = \frac{1}{N} + \frac{1}{D_k^2} \quad (15)$$

$$\frac{GFLOPS_{DSCnv}}{GFLOPS_{Conv}} = \frac{D_k \times D_k \times 1 \times D_g \times D_g \times M + M \times 1 \times 1 \times D_g \times D_g \times N}{D_k \times D_k \times M \times D_g \times D_g \times N} = \frac{1}{N} + \frac{1}{D_k^2} \quad (16)$$

Overall, depth-wise separable convolutions provide an efficient convolution method for vehicle taillight detection, balancing computational efficiency and model performance. Compared to traditional convolution, they offer a higher computational efficiency and a greater adaptability. Although they reduce the number of parameters, they typically maintain a relatively good detection performance in practical applications. For the detection of specific objects like vehicle taillights, a depth-wise separable convolution can provide a sufficient receptive field and retain the necessary feature information to achieve efficient and accurate detection. The complete improved model is shown in Figure 10.

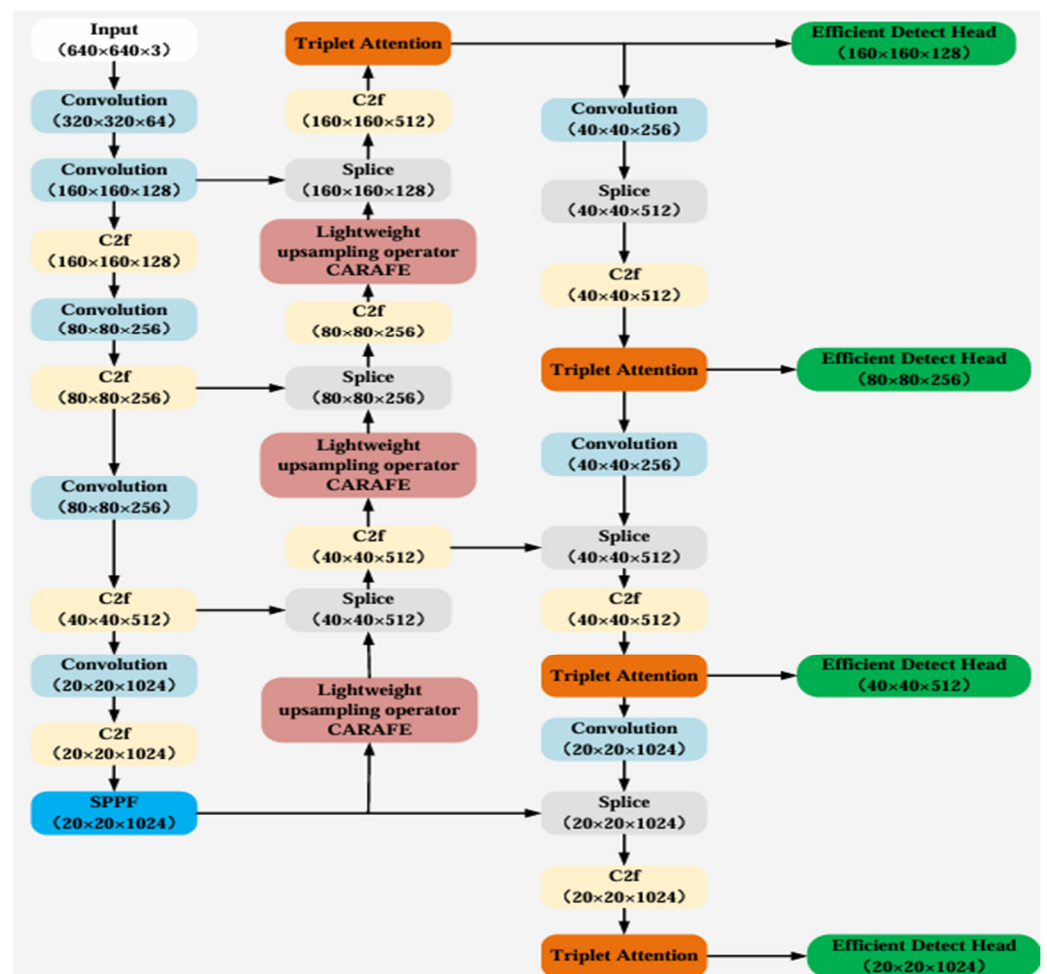


Figure 10. Diagram of the depth-wise separable convolution module, where C2f represents cross stage partial networks with factorized convolution and SPPF represents spatial pyramid pooling-fast.

4. Experimental Results

4.1. Data Feature Analysis

This study mainly focuses on the lane-changing behavior of vehicles while driving, and therefore focuses on the braking and turn signals of the preceding vehicle. Several common taillight flashing states during driving are shown in Figure 11.



Figure 11. Several types of common taillight style diagrams.

Brake lights are typically symmetrically mounted at the rear of the vehicle and some vehicles further augment their warning capabilities by adding a brake light in the rear window. When the driver presses the brake pedal, the braking system triggers the brake lights to illuminate. The red light source of the brake lights remains clearly visible even in low-visibility weather conditions, ensuring reliable communication of the braking signal [25]. Under different lighting conditions, vehicle brake lights exhibit varying degrees of scattering, which may affect the accurate observation and recognition of their status, as shown in Figure 12. The changes in lighting conditions pose a challenge to the robustness of taillight detection algorithms, especially in low-visibility scenes where stronger feature extraction and processing capabilities are required.

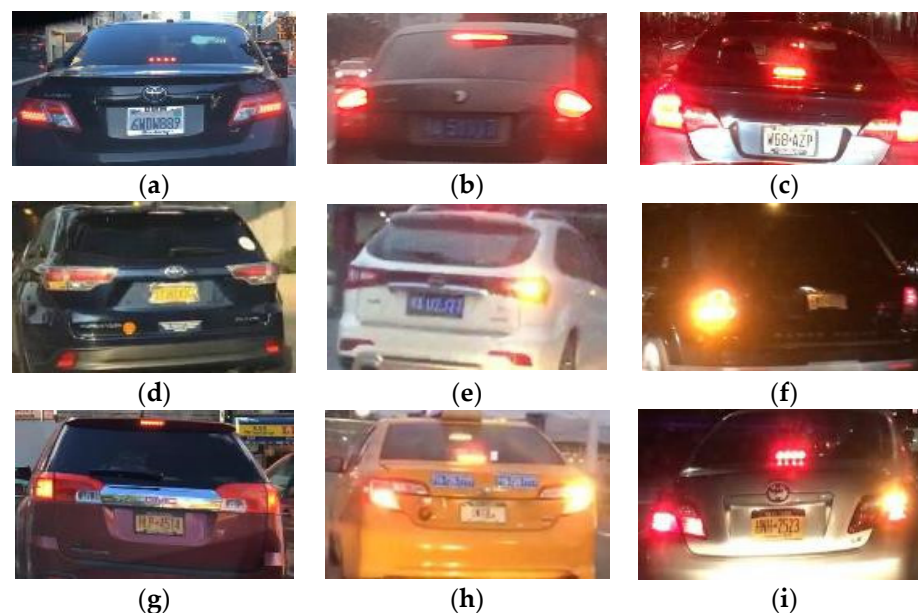


Figure 12. Light characteristics in different environments. (a–c) are brake lights in the high-intensity light scenario, low-light scenario, and nighttime scenario, respectively; (d–f) are turning lights in the high-intensity light scenario, low-light scenario, and nighttime scenario, respectively; (g–i) are both lights illuminated simultaneously in the high-intensity light scenario, low-light scenario, and nighttime scenario, respectively.

Turning signals are typically located either inside or outside the vehicle's taillight assembly [26]. Turning signals usually employ orange bulbs or LED lights, which are easier to recognize on the road, and they flash at a fixed frequency to enhance the communication of lane-changing information. This creates a sharp contrast with the red brake lights and white headlights. The performance of turn signals varies significantly under different climatic and lighting conditions, especially at night, where diffraction halos and image glare greatly impact the readability of taillight features. Designing robust turn signal detection algorithms thus requires consideration of diverse environmental factors.

In different driving contexts, the morphological characteristics of taillights and their signal semantics display diversity, influenced by factors such as vehicle type, road conditions, and lighting situations [27]. In the sparse road scenario shown in Figure 13a, the distant vehicles and their taillights appear at smaller scales due to the perspective effect. Conversely, in the dense road scenario depicted in Figure 13b, the occlusion and overlapping of taillight features lead to frequent signal loss and missed detections. In the nighttime road scenario illustrated in Figure 13c, the diffraction-induced halos and traffic light interference can cause confusion and false detections in interpreting taillight signals. Additionally, urban road driving scenes typically exhibit rapid and continuous changes, further increasing the complexity of taillight detection and lane-change recognition tasks.



Figure 13. Example of a road driving scenario. (a) A sparse road scenario, (b) a dense road scenario, and (c) a nighttime road scenario.

4.2. Experimental Environment and Evaluation Indicators

The simulation experiments in this study were designed using Python 3.8 and the PyTorch 1.1.0 framework, running on a 64-bit Linux system equipped with an Intel(R) Core (TM) i9-11900K CPU. To enhance training efficiency, the experimental platform utilized an NVIDIA GeForce RTX 3090 GPU with CUDA 11.3 and CuDNN 10.0 for graphical acceleration. The computations were performed using resources from Baidu AutoDL cloud servers. To optimize loss reduction based on prior training experience, the stochastic gradient descent algorithm was employed, with a batch size set to 128, an initial learning rate of 0.01, and training was conducted over 200 epochs.

The metrics used for evaluating the performance of the proposed detection model include precision (P), recall (R), average precision (AP), mean average precision (mAP), number of parameters (Params), and frames per second (FPS) [28]. Among these, higher values of P and R indicate better model detection accuracy. mAP is used to measure the overall performance of the model; the larger the mAP, the better the model's training performance. Compared to P and R, mAP provides a more comprehensive reflection of the algorithm's performance. Therefore, in this experiment, mAP @0.5 is selected for a comprehensive evaluation of the algorithm's accuracy. The calculation formulas for the evaluation metrics used in this paper are shown in Equations (17)–(20).

$$P = \frac{T_P}{T_P + F_p} \quad (17)$$

$$R = \frac{T_P}{T_P + F_N} \quad (18)$$

$$R = \frac{T_P}{T_P + F_N} \quad (19)$$

$$mAP = \frac{1}{n} \sum_{j=1}^n AP_j \quad (20)$$

Additionally, since mAP only reflects the accuracy of the model, other metrics of the model need to be considered, such as the number of parameters and inference speed [29]. Params denotes the size of the model's parameters, directly determining the inference speed of the model. FPS can be understood as the execution speed of the algorithm. Its expression is given in Equation (21):

$$FPS = frameNum / elapsedTime \quad (21)$$

where *elapsedTime* represents a fixed period, and *frameNum* denotes the number of frames transmitted during that period.

4.3. Analysis of Ablation Experiments

In the task of vehicle taillight detection, we conduct ablation experiments by progressively integrating the components designed in this study, which are CARAFE, TriAtt, and EfficientP2Head, into the improved YOLOv8s model. This was conducted to systematically analyze the impact of each component on the model's accuracy and robustness. The perfor-

mance data are presented in Table 1, and a detailed analysis of the effect of each component is provided below.

Table 1. Ablation experiment results.

Method	mAP (%)	AP (%)			
		Vehicle	Brake	Turn Left	Turn Right
YOLOv8s	79.77	89.67	82.63	71.42	75.36
YOLOv8s-CARAFE	81.32	90.83	84.16	73.91	76.38
YOLOv8s-CARAFE-TriAtt	82.91	93.54	85.65	74.88	77.57
YOLOv8s-CARAFE-TriAtt-EfficientP2Head	85.48	95.65	87.63	76.75	81.89

The introduction of the CARAFE module aims to address the issue of the fine-grained perception of small objects in taillight detection. Taillights often have relatively small sizes, and traditional upsampling algorithms may lead to information loss that affects detection performance. CARAFE significantly improves the perception ability of small objects with minimal parameters and computation cost through its lightweight universal upsampling operator and feature reassembly module, enhancing the accuracy of taillight detection. In the ablation study data, the introduction of this module increased the mAP of YOLOv8s from 79.77% to 81.32%. Specifically, for the key categories of left and right turns, the AP improved by 2.49% and 1.02%, respectively. This indicates that the CARAFE module significantly aids in the recognition of important targets and positively addresses the issue of information loss for small objects in taillight detection.

To better capture the intrinsic features of taillights and to enhance the recognition of positive samples, the TriAtt triplet attention module is introduced. Comparing experimental data, the overall mAP increased from 81.32% to 82.91% after integrating the TriAtt module, with corresponding improvements in the AP of various categories. For the critical categories of braking and left and right turns, the AP increased by 1.49%, 0.97%, and 1.19%, respectively. This demonstrates that the TriAtt module has a positive impact on model performance by better capturing the intrinsic features of taillight data, thereby improving the accuracy of the taillight detection task and validating the effectiveness of the TriAtt module in this application.

To address the performance degradation of traditional detection heads when handling small objects, the EfficientP2Head is introduced to enhance the accuracy and robustness of taillight detection tasks involving small objects. The integration of EfficientP2Head resulted in significant improvements. Specifically, the AP for the vehicle category increases from 93.54% to 95.65%; for the braking category, it increases from 85.65% to 87.63%; for the left turn category, it increases from 74.88% to 76.75%; and for the right turn category, it increases from 77.57% to 81.89%. This indicates that the EfficientP2Head module has a positive impact on the detection of various taillight categories, improving the model's accuracy in recognizing taillights and small objects.

In summary, the integration of these three modules results in an increase in the mAP from 79.77% to 85.48%, with improvements in the AP of each category, validating the effectiveness of the introduced components. This demonstrates that the final model achieves effective optimization in addressing small object taillight detection and complex feature extraction.

4.4. Analysis of Different Model Experiments

This section evaluates five recognition methods based on YOLOv4-tiny, YOLOv5s, YOLOv7-tiny, YOLOv8, and the improved taillight detection algorithm carried out in this paper on a test set and tests them in comparison with those based on the improved detection model; the results of the comparison are shown in Table 2.

Table 2. Comparison of experiment results.

Detection Model	Precision (%)	Recall Rate (%)	mAP (%)	Model Size (M)	Computational Volume GFLOPs	FPS/(f.s ⁻¹)
YOLOv4-tiny	64.72	44.49	51.40	22.5	6.83 G	50.82
YOLOv5s	77.64	54.79	60.57	7.2	16.66 G	84.74
YOLOv7-tiny	74.49	52.18	58.67	6.02	5.53 G	136.67
YOLOv8s	88.53	75.31	79.77	11.2	28.67 G	108.8
Improved Model	93.27	79.86	85.48	9.69	25.91 G	112.3

The focus task of this paper is vehicle taillight detection. Through in-depth comparison and analysis of different target detection models, this paper focuses on investigating and optimizing the performance of the improved model in the task of vehicle taillight detection. In this paper, the superiority of the improved model over YOLOv4-tiny, YOLOv5s, YOLOv7-tiny, and YOLOv8s in terms of accuracy, recall, and mAP will be explored in detail.

Firstly, YOLOv4-tiny is relatively small in terms of model size and computational effort, making it suitable for resource-constrained environments. However, its relatively low precision (64.72%) and recall (44.49%) may constrain its application in vehicle taillight detection tasks in complex scenarios. This suggests that the model may be able to satisfy some simple scenarios under ideal conditions, but has limited performance in complex scenarios in real-world environments. YOLOv5s shows a significant improvement in accuracy and recall compared to YOLOv4-tiny, but the increased model size and computational effort may limit its applicability in resource-constrained vehicle environments. The model needs to weigh the relationship between resource consumption and performance, while achieving performance gains. YOLOv7-tiny provides a more balanced option for vehicle taillight detection tasks by improving accuracy and recall, while maintaining a relatively small model size and computational effort. It has advantages in real-time performance and resource utilization and is suitable for in-vehicle scenarios, but there is some room for improvement in its accuracy.

Comparatively, the improved model (ours) in this paper excels in the vehicle taillight detection task. With high precision (93.27%) and recall (79.86%), it ensures the precise identification of vehicle taillights, providing strong support for vehicle safety. The relatively small model size (9.69M) and efficient computation (25.91G GFLOPs) lay the foundation for real-time performance. In addition, the model in this paper achieves 85.48% on the mAP metric, which illustrates the excellent performance on each taillight category.

In summary, the model in this paper has better characteristics in several performance indicators. With relatively small model parameters and computational volume, it ensures that the real-time performance requirements of lane-change intention detection can be met in real vehicle systems, and provides a high-frame-rate real-time detection capability for driver assistance systems.

4.5. Comparison Analysis of Detection Results

To validate the impact of the proposed model optimizations on convergence performance, a thorough comparison is conducted between the original YOLOv8s model and the improved YOLOv8s model on the dataset. In terms of the loss curves for bounding box loss, confidence loss, and class loss, both models maintained relatively low loss levels. However, the improved YOLOv8s model demonstrates superiority in class loss and confidence loss compared to the original YOLOv8s model. A further analysis of the convergence curves for four performance metrics, which are precision, recall, mAP@0.5, and mAP@[0.5:0.95], reveals that the improved YOLOv8s model exhibits a better performance in precision and recall, with smoother convergence curves, as shown in Figures 14 and 15. This indicates that the optimized model achieved significant improvements in overall performance.

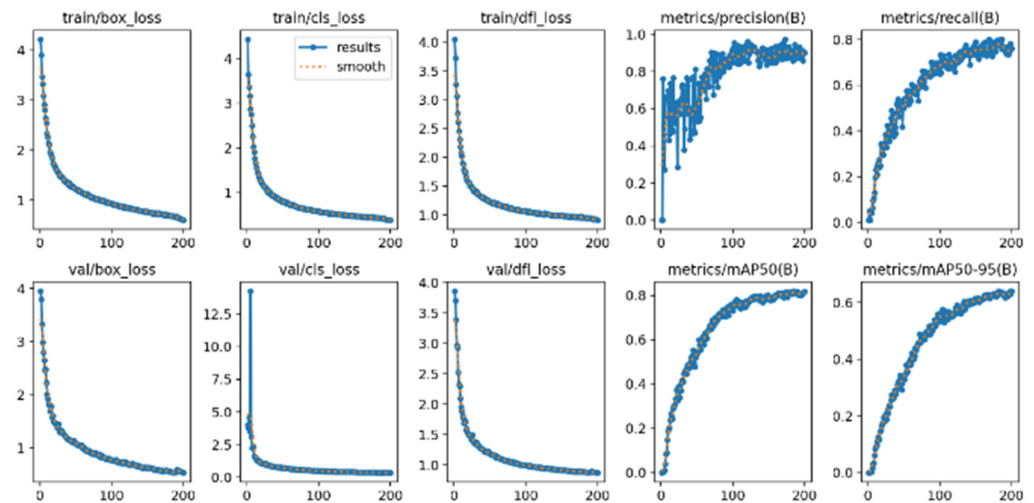


Figure 14. Convergence plot of the original model.

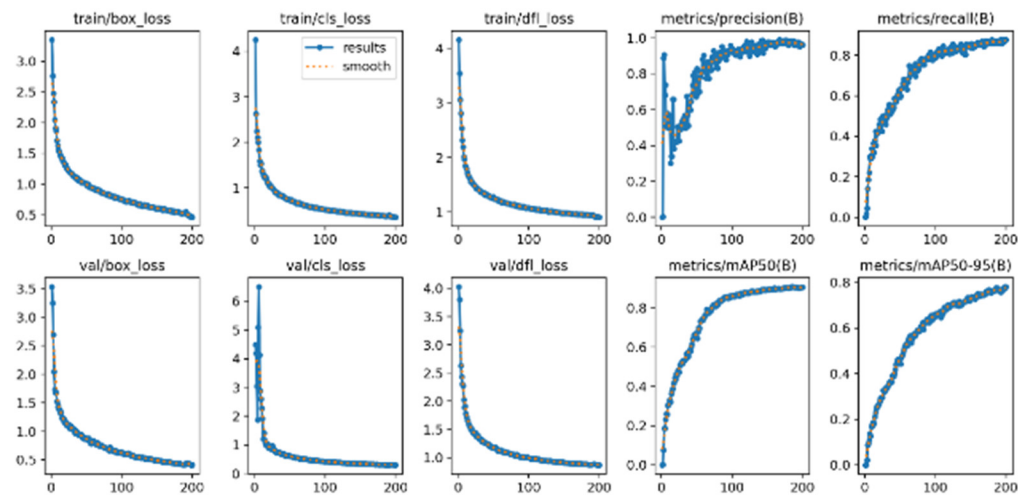


Figure 15. Convergence plot of the improved model.

Therefore, based on the comprehensive comparison results, the improved YOLOv8s model demonstrates an enhanced convergence performance, providing more reliable support for practical applications in taillight detection tasks.

The optimized detection model's semantic recognition results of taillights, which include comprehensive improvements, are depicted in Figure 16 across various driving conditions. Braking signal lights are marked with red bounding boxes; left and right turn signals with orange and pink bounding boxes, respectively; and vehicles are marked with yellow bounding boxes. In Figure 16a, despite the presence of diffraction halos and chaotic light sources, the optimized model accurately identifies the semantic information of the taillights. Figure 16b illustrates a vehicle turning right in bright daylight, where the turn signal features are weak. Figure 16c shows frequent vehicle occlusion in congested roads. Figure 16d demonstrates the detection results for right-turning vehicles and small target vehicles in a tunnel scenario with multi-source light interference. Figure 16e,f depict detection results in dense driving conditions, where taillight features are often obscured and overlapped, potentially leading to information loss and missed detections. Nonetheless, the optimized model effectively captures taillight information even in these challenging scenarios.

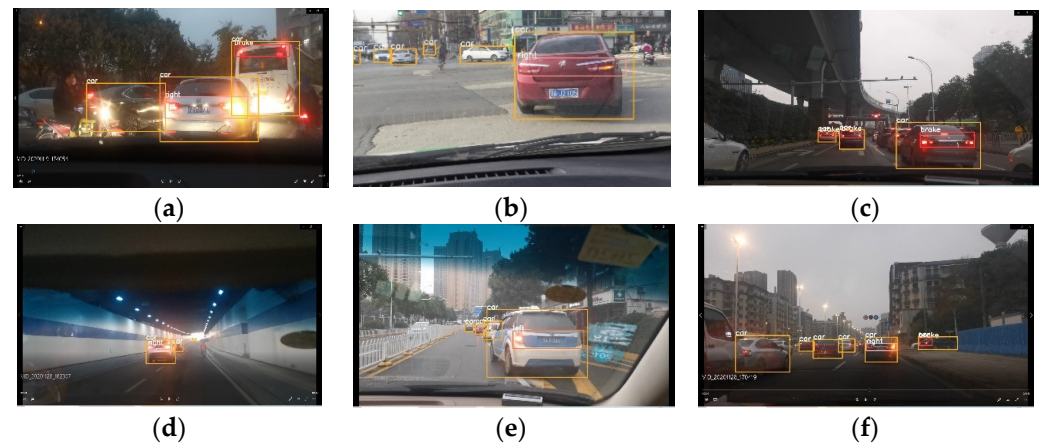


Figure 16. Different traffic detection results. (a) is the low-light congested scene; (b) is the right turn under strong daylight conditions; (c) is the evening congested scene; (d) is the multi-light-source interference environment in tunnels; and (e,f) are dense driving conditions, orange boxes are detected vehicle taillights.

Figure 17 illustrates a sequence of images used to detect taillight flashing conditions. Turn signals flash at fixed periodic frequencies, posing a real-time challenge for the algorithm. As shown in Figure 17a–d, the consecutive frames accurately capture the on–off changes in the front vehicle’s turn signals. Additionally, in Figure 17b–d, crowded vehicles on both sides, small target vehicles, and occluded vehicles are detected, but also the brake and turn signals of adjacent vehicles are captured. This demonstrates that the model can still capture taillight information even in crowded, multi-feature scenarios.



Figure 17. Consecutive figure detection results. (a) is the first frame; (b) is the second frame; (c) is the third frame; and (d) is the forth frame, orange boxes are detected vehicle taillights.

Figure 18 presents a visual comparison between the baseline detection model and the improved model. It can be observed that under complex and adverse conditions such as dense congestion, ambient light interference, and perspective effects, the improved detection model accurately captures taillight information from small targets, overlapping occlusions, and sparse information.

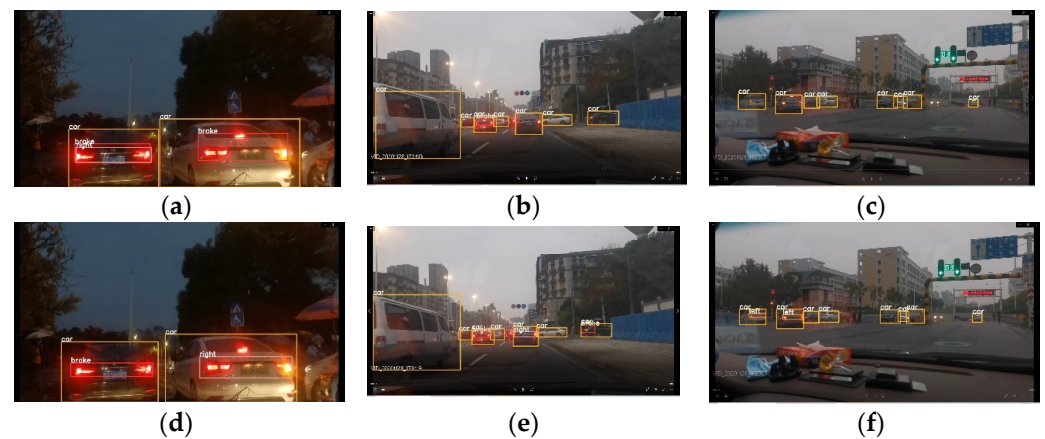


Figure 18. Consecutive figure detection results. (a–c) are the detection results of the baseline detection model; (d–f) are the detection results of the improved detection model, orange boxes are detected vehicle taillights.

To demonstrate the enhanced effect of the improved model on taillight feature information, this study incorporates GradCAM heatmaps to visually display the areas of interest identified by the model. As shown in Figure 19, the improved taillight detection model gradually shifts its focus from the vehicle center to the bilateral taillight regions, indicating a significant enhancement in the model's ability to perceive taillight semantic information.

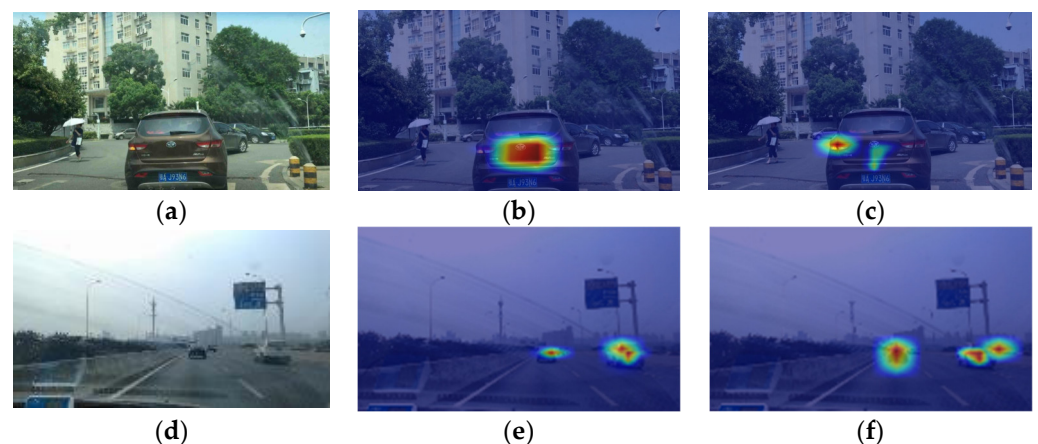


Figure 19. Visualization of detection areas of interest. (a,d) are the scenes to be detected; (b,e) are the detection results of the baseline detection model; and (c,f) are the detection results of the improved detection model.

Through a series of quantitative and qualitative evaluations presented earlier, it is confirmed that the improved model exhibits robustness and efficiency in practical application scenarios. By optimizing the upsampling operator, reconstructing the decoupled detection head, and employing a multi-dimensional interactive attention strategy, the model demonstrates superior detection performance and evaluation metrics in diverse driving scenarios compared to the original YOLOv8s model. Additionally, the improved model shows better performance in capturing small and overlapping targets, and places greater emphasis on the semantic information of vehicle taillights.

5. Conclusions

By integrating the CARAFE, TriAtt, and EfficientP2Head modules, this study significantly enhances the performance of the model in the task of taillight detection. The CARAFE module, through its lightweight universal upsampling operator and feature reorganization design, improves the perception of small targets. This integration increased

the mAP of YOLOv8s from 79.77% to 81.32%, with the AP for left turn and right turn categories improving by 2.49% and 1.02%, respectively. This indicates the positive role of the CARAFE module in addressing information loss in taillight detection. The TriAtt triple attention module further enhances the model's ability to capture intrinsic taillight features, boosting the overall mAP from 81.32% to 82.91%, with the AP for the brake and left and right turn categories increasing by 1.49%, 0.97%, and 1.19%, respectively. This demonstrates the beneficial impact of the TriAtt module on model performance, improving its accuracy. To address the decline in performance when handling small targets with traditional detection heads, the EfficientP2Head module is introduced, significantly improving the AP in various taillight detection categories. The AP for vehicle, brake, left turn, and right turn categories increased to 95.65%, 87.63%, 76.75%, and 81.89%, respectively, illustrating the enhanced robustness and small target recognition capability provided by EfficientP2Head. Collectively, these improvements raised the final model's mAP from 79.77% to 85.48%, with notable AP enhancements across all categories. The proposed improved model excels in vehicle taillight detection tasks, achieving a high precision rate of 93.27% and a recall rate of 79.86%, while maintaining a small model size and efficient computational load, thus meeting the real-time performance requirements of practical in-vehicle systems.

In future work, an exploration of the newly developed YOLOv9 model could further enhance the results achieved in this study. YOLOv9 introduces several advancements over YOLOv8, including a more refined architecture, improved data augmentation techniques, and a novel loss function tailored for the better handling of complex object detection scenarios. These enhancements are expected to offer significant benefits, such as increased accuracy and robustness, particularly in detecting small and densely packed objects. Additionally, YOLOv9's architecture is designed to be more efficient, potentially reducing the computational load while maintaining or improving detection speed, which is critical for real-time in-vehicle systems. Therefore, future research will focus on evaluating the benefits of YOLOv9 in comparison to the current implementation, aiming to push the boundaries of taillight detection performance even further.

Author Contributions: Conceptualization, M.L., J.Z. and W.L.; methodology, M.L. and T.Y.; validation, M.L. and J.Z.; formal analysis, W.L. and W.C.; investigation, M.L. and X.Y.; resources, M.L.; data curation, J.Z. and W.L.; writing—original draft preparation, M.L. and T.Y.; writing—review and editing, H.L., L.D. and X.Y.; visualization, T.Y. and X.Y.; supervision, H.L., W.C. and L.D.; project administration, M.L., H.L. and L.D.; funding acquisition, M.L., W.C. and L.D. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded in part by the Hubei Province Technological Innovation Major Project (No. 2019AAA025), in part by the Fundamental Research Funds for the Central Universities (104972024KFYd0012), and in part by the Jiangxi Provincial Department of Transportation Science and Technology Project (No. 2022X0043).

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

Conflicts of Interest: Ming Li, Jian Zhang are employees of Jiangxi Transportation Institute Co., Ltd, Weixia Li is employee of Jiangxi Provincial Intelligent Transportation Affairs Center. The paper reflects the views of the scientists, and not the company.

References

1. Traffic Management Bureau, Ministry of Public Security. *Nationwide Motor Vehicles Reach 430 Million and Drivers Reach 520 Million*; [EB/OL]; Traffic Management Bureau: Beijing, China, 2023.
2. Fan, H.; Han, K.; Sun, L.; Cui, R. Research on tail light language recognition method based on visual autonomous vehicle. *Comput. Knowl. Technol.* **2010**, *6*, 9790–9792.
3. Liu, Z.; Ye, Q.; Li, F.; Zhao, M.; Nie, J.; Sun, X. Tail light detection algorithm based on four thresholds of luminance and colour. *Comput. Eng.* **2010**, *36*, 202–203+206.
4. Guo, J.; Wang, J.; Yi, S.; Li, K. A monocular vision-based method for detecting vehicles ahead at night. *Automot. Eng.* **2014**, *36*, 573–579+585.
5. Zhang, J.; Xu, X.; Li, J. Nighttime tail light extraction method based on halo level feature verification. *Comput. Age* **2015**, *08*, 6–8+11.

6. Almagambetov, A.; Velipasalar, S.; Casares, M. Robust and computationally lightweight autonomous tracking of vehicle taillights and signal detection by embedded smart cameras. *IEEE Trans. Ind. Electron.* **2015**, *62*, 3732–3741. [[CrossRef](#)]
7. Tian, Q.; Kong, B.; Sun, C.; Wang, C. Detection and Recognition of Vehicle Tail Light Lamp Phrases. *Comput. Syst. Appl.* **2015**, *24*, 213–218.
8. Jin, L.; Cheng, L.; Cheng, B. Nighttime forward vehicle detection based on millimetre wave radar and machine vision. *J. Automot. Saf. Energy Conserv.* **2016**, *7*, 167–174.
9. Chien, C.-L.; Hang, H.-M.; Tseng, D.-C.; Chen, Y.-S. An image based overexposed taillight detection method for frontal vehicle detection in night vision. In Proceedings of the 2016 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA), Jeju, Korea, 13–15 December 2016; IEEE: New York, NY, USA, 2016; pp. 1–9.
10. Vancea, F.I.; Costea, A.D.; Nedevschi, S. Vehicle taillight detection and tracking using deep learning and thresholding for candidate generation. In Proceedings of the 2017 13th IEEE International Conference on Intelligent Computer Communication and Processing (ICCP), Cluj-Napoca, Romania, 7–9 September 2017; IEEE: New York, NY, USA, 2017; pp. 267–272.
11. Li, X. Video Vehicle and Tail Light Language Recognition Based on Deep Learning. Master's Thesis, Guangdong University of Technology, Guangzhou, China, 2020.
12. Li, G.J. Research on Deep Learning-Based Algorithm for Forward Vehicle Detection and Tail Light State Judgement. Master's Thesis, Shandong University of Science and Technology, Qingdao, China, 2021.
13. Vancea, F.I.; Nedevschi, S. Semantic information based vehicle relative orientation and taillight detection. In Proceedings of the 2018 IEEE 14th International Conference on Intelligent Computer Communication and Processing (ICCP), Cluj-Napoca, Romania, 6–8 September 2018; IEEE: New York, NY, USA, 2018; pp. 259–264.
14. Gao, F.; Ge, Y.; Lu, S.; Zhang, Y. On-line vehicle detection at nighttime-based tail-light pairing with saliency detection in the multi-lane intersection. *IET Intell. Transp. Syst.* **2019**, *13*, 515–522. [[CrossRef](#)]
15. Liu, J. Forward Vehicle Detection under Urban Road Conditions and Its Tail Light Lamp Language Recognition. Master's Thesis, Xi'an University of Technology, Xi'an, China, 2023.
16. Li, Q.; Garg, S.; Nie, J.; Garg, S.; Nie, J.; Li, X.; Liu, R.W.; Cao, Z.; Hossain, M.S. A highly efficient vehicle taillight detection approach based on deep learning. *IEEE Trans. Intell. Transp. Syst.* **2020**, *22*, 4716–4726. [[CrossRef](#)]
17. Parvin, S.; Rozario, L.J.; Islam, M.E. Vision-based on-road nighttime vehicle detection and tracking using taillight and headlight features. *J. Comput. Commun.* **2021**, *9*, 29–53. [[CrossRef](#)]
18. Jeon, H.J.; Nguyen, V.D.; Duong, T.T.; Jeon, J.W. A deep learning framework for robust and real-time taillight detection under various road conditions. *IEEE Trans. Intell. Transp. Syst.* **2022**, *23*, 20061–20072. [[CrossRef](#)]
19. Oh, G.; Lim, S. One-Stage Brake Light Status Detection Based on YOLOv8. *Sensors* **2023**, *23*, 7436. [[CrossRef](#)] [[PubMed](#)]
20. Xu, S.; Wang, X.; Lv, W.; Chang, Q.; Cui, C.; Deng, K.; Wang, G.; Dang, Q.; Wei, S.; Du, Y.; et al. PP-YOLOE: An evolved version of YOLO. *arXiv* **2022**, arXiv:2203.16250.
21. Wang, J.; Chen, K.; Xu, R.; Liu, Z.; Loy, C.C.; Lin, D. Carafe: Content-aware reassembly of features. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Seoul, Republic of Korea, 27 October–2 November 2019; pp. 3007–3016.
22. Woo, S.; Park, J.; Lee, J.-Y.; Kweon, I.S. Cbam: Convolutional block attention module. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 3–19.
23. Chollet, F. Xception: Deep learning with depthwise separable convolutions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 1251–1258.
24. Yu, F.; Chen, H.; Wang, X.; Xian, W.; Chen, Y.; Liu, F.; Madhavan, V.; Darrell, T. BDD100K: A Diverse Driving Dataset for Heterogeneous Multitask Learning. In Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 13–19 June 2020; IEEE: New York, NY, USA, 2020; pp. 2633–2642.
25. Tong, B.; Chen, W.; Li, C.; Du, L.; Xiao, Z.; Zhang, D. An improved approach for real-time taillight intention detection by intelligent vehicles. *Machines* **2022**, *10*, 626. [[CrossRef](#)]
26. Lee, D.H.; Liu, J.L. End-to-end deep learning of lane detection and path prediction for real-time autonomous driving. *Signal Image Video Process.* **2023**, *17*, 199–205. [[CrossRef](#)]
27. Tabelini, L.; Berriel, R.; Paixao, T.M.; Badue, C.; De Souza, A.F.; Oliveira-Santos, T. Keep your eyes on the lane: Real-time attention-guided lane detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Nashville, TN, USA, 20–25 June 2021; pp. 294–302.
28. Qin, Z.; Wang, H.; Li, X. Ultra fast structure-aware deep lane detection. In Proceedings of the Computer Vision-ECCV 2020: 16th European Conference, Glasgow, UK, 23–28 August 2020, Proceedings, Part XXIV 16; Springer International Publishing: Cham, Switzerland, 2020; pp. 276–291.
29. Chen, J.; Mai, H.S.; Luo, L.; Chen, X.; Wu, K. Effective feature fusion network in BIFPN for small object detection. In Proceedings of the 2021 IEEE International Conference on Image Processing (ICIP), Anchorage, AK, USA, 19–22 September 2021; IEEE: New York, NY, USA, 2021; pp. 699–703.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.