Scilight

Journal of Machine Learning and Information Security
https://www.sciltp.com/journals/jmlis

Article

Check for updates

An Enhanced YOLOv8 Algorithm with C2f_Parnet for Vehicle and Pedestrian Detection

Jun Peng [1,2,*], Wei Wang [1], Yuanmin He [3], Shangzhu Jin [2,4], Jingxin Liu [1,2] and Mou Zhou [2]

[1] School of Mathematical and Physical Sciences, Chongqing University of Science and Technology, Chongqing 401331, China
[2] Research Institute of Intelligent Mathematics and Autonomous AI (RI-IM·AI*), CQUST, Chongqing 401331, China
[3] Chongqing Communication Design Institute Co., Ltd, Chongqing 400041, China
[4] Informatization Office, Chongqing University of Science and Technology, Chongqing 401331, China
* Correspondence: jpeng@cqust.edu.cn

How To Cite: Peng, J.; Wang, W.; He, Y.; et al. An Enhanced YOLOv8 Algorithm with C2f_Parnet for Vehicle and Pedestrian Detection. *Journal of Machine Learning and Information Security* **2026**, *2*(3), 17. https://doi.org/10.53941/jmlis.2026.100017

Received: 8 April 2026
Revised: 26 June 2026
Accepted: 21 July 2026
Published: 3 August 2026

Abstract: This paper proposes enhancements to the YOLOv8s model for autonomous driving object detection tasks. Firstly, a C2f_Parnet module is designed based on the Parnet parallel network architecture to reduce network depth and computational latency. Secondly, GSConv is modified by employing dual 3 × 3 convolutions to decrease parameter count. Thirdly, L2-scored structural pruning is introduced to compress redundant channels. Finally, detection accuracy is improved using EIoU loss. Experiments on the KITTI dataset demonstrate that the improved model maintains a mAP50 of 0.917. The performance of this model is almost comparable to the original model (0.939), but with parameters reduced to 11 million and GFLOPs decreased to 26.2. Compared with other methods, L2 pruning achieved the highest mAP50-95 score of 0.753. In complex traffic environments, this model demonstrates robust real-time detection capabilities, can be flexibly deployed, and is a useful choice for lightweight detection in autonomous driving.

Keywords: object detection; C2f_Parnet; channel pruning; autonomous driving

1. Introduction

Object detection [1] is a core issue in computer vision and a key technology for machine-based visual perception. It supports a range of advanced applications, such as autonomous driving, video surveillance and human-computer interaction. The popular detection algorithms are divided into two types. The first type is the two-stage detector, such as the series of R-CNN [2]. These methods can generate region proposals, perform classification and regression. These methods have high accuracy, but they often have high computational costs and slow inference speed. The second type is the single-stage detectors, such as SSD [3] and YOLO [4]. These methods adopt a regression-based approach. In a single forward pass, they can simultaneously complete location and category prediction. This way makes them have faster processing speed and enables real-time detection, but the early accuracy was slightly inferior to that of the two-stage detectors. Released by Ultralytics in 2023, YOLOv8 introduces several innovations, including an anchor-free design, an efficient backbone network, and an improved loss function. On standard benchmark datasets, it achieves competitive performance and has become a widely adopted framework for real-time object detection. Varghese [5] further investigated its robustness in their recent study.

However, the use of YOLOv8 for the detection tasks of vehicle and pedestrians faces many challenges. These challenges are specifically manifested in the following notable shortcomings:

(1) Scale processing should be improved. The model uses PAN-FPN for multi-scale feature fusion [6], but it still has difficulties processing objects at extreme scales, especially those at small scale. In scenes with severe occlusion, it is common to have missed detections and false alarms.
(2) In harsh environmental conditions performance degrades. These factors such as sudden changes in lighting, inclement weather or motion blur can degrade object features, even lead to texture loss. This typically results in a significant decrease in detection accuracy.

(3) The computational efficiency can be further improved. This model still has some redundancy in feature computation. Without sacrificing accuracy or ideally even improving the accuracy, it is a major challenge for practical deployment to find ways to make the model lighter and faster.

2. Current State of Research

To address these challenges, many researchers have explored ways to improve YOLOv8. Their work aims to improve specific aspects, such as the model's accuracy, inference speed and robustness.

Some studies have focused on lightweight design and improved efficiency. Xie et al. used knowledge distillation to compress YOLOv8 to boost scene detection in autonomous driving [7]. For resource-limited devices, Duan et al. came up with a lightweight multi-scale vehicle detector [8]. In terms of accuracy and efficiency , Zhao et al. introduced YOLOv8-BASW [9]. For small object detection in drone-captured aerial views, Li et al. developed RLRD-YOLO [10]. In high-resolution remote sensing images, Shao et al. adjust YOLOv8 to boost vehicle detection performance [11]. It has drawn attention to improving robustness in complex environments. Ma et al. improved YOLOv8 to make it more stable in challenging surveillance scenarios [12]. For multi-scale pedestrian detection accuracy, Liu et al. proposed the lightweight YOLOv8-CB model [13]. Sun et al. built a robustness-enhanced version of YOLOv8 and advanced multi-object detection and tracking performance [14]. For ensemble methods and architectural innovations, Lv et al. combined EfficientDet with YOLOv8 to improve vehicle detection accuracy [15]. Bakirci evaluated several YOLOv8 variants to offer useful guidance for choosing architectures in intelligent aerial traffic monitoring [16]. With the development of end-to-end systems and practical deployment, Dou et al. researched an improved YOLOv8 [17] for traffic monitoring on drones. For autonomous driving scenarios, Du et al. proposed a lightweight YOLOv8 detector [18]. Liu et al. showed their improvements lead to higher traffic detection accuracy and demonstrated real-world usefulness [19].

Although efforts to improve YOLOv8 have shown great potential, a clear trade-off remains, namely that higher detection accuracy often means more complex networks and heavier memory usage. To solve this, we redesigned the architecture of YOLOv8. Our contributions are listed below:

(1) Based on the parallel design philosophy of ParNet, we introduced the C2f_ParNet module to shallower the network architecture and accelerate computation.
(2) The GSConv structure was improved to achieve comparable feature extraction with fewer parameters.
(3) We utilized the L2 norm of channel weights as an importance metric to globally remove redundant channels.
(4) By adopting the EIoU loss function, the regression accuracy of bounding boxes was significantly improved.

The structure of this paper is as follows: Section 3 presents the methods and implementation, Section 4 covers the dataset, experimental setup and comparative results, Section 5 offers the conclusions.

3. Methodology

3.1. Motivations

As an indispensable component of neural networks, the convolutional layer is responsible for extracting discriminative feature information from raw images. The size of the convolution kernel directly determines the overall parameter count of the network. Over-reliance on standard convolutional structures inevitably leads the model toward redundancy and bloat. To address the problem of excessive size in object detection networks, lightweight alternatives have attracted widespread attention. Among them, depthwise separable convolution has been extensively adopted due to its ability to effectively reduce the parameter count. However, its feature extraction performance still lags behind that of standard convolution. What's more, extensive use of depthwise separable convolution increases memory access frequency. This causes additional latency. We seek an optimization method to address this issue. The method balances compression and feature preservation. Runtime memory overhead is controlled. The network transitions smoothly toward lightweight design.

3.2. YOLOv8

YOLOv8 stands out among object detection algorithms. Its detection speed is fast, and its recognition accuracy is high. We adopt it as our baseline network. The research focuses on object detection in autonomous driving environments. The CBL module serves as the basic feature extraction unit in YOLOv8. This module has a simple structure consisting of three stacked layers. These three layers are, in sequence, a convolutional layer, a batch normalization layer, and a Leaky ReLU activation function. As shown in Figure 1a. The convolutional layer is responsible for extracting feature information from the input image. The batch normalization layer maintains the stability of input data distribution. This helps accelerate the model convergence process. The Leaky ReLU activation

function enhances the network's ability to fit non-linear features.

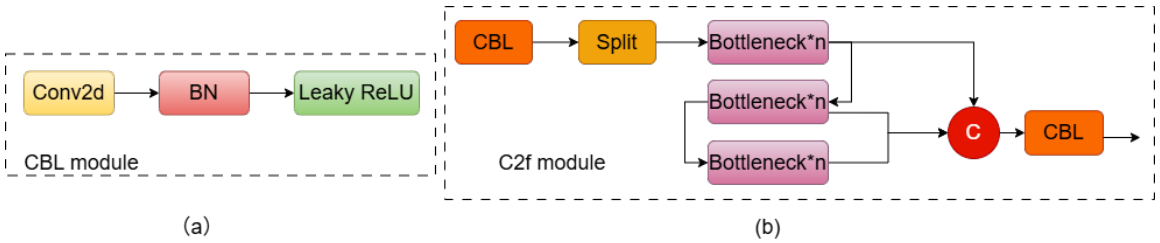


Figure 1. Illustration of network module structures: (**a**) CBL module; (**b**) C2f module.

The C2f module is an important component of YOLOv8. Its main function is to enhance the efficiency of feature fusion. The design of this module references two classic schemes. They are the C3 module in YOLOv5 and the feature aggregation approach of the ELAN architecture. The outcome is a more streamlined layer aggregation structure. Redundant intermediate layers are eliminated. As shown in Figure 1b, this structure integrates multi-scale features. This facilitates smooth propagation of gradient information. Detection accuracy and generalization performance are therefore significantly enhanced.

In the C2f architecture diagram, "C" denotes the Concat operation. The workflow of this module is as follows. The C2f module first receives the feature maps output from the CBL module. It then divides them into two branches. One branch goes through iterative processing by multiple bottleneck modules. The other branch remains unchanged. Finally, the Concat operation merges the features from the two branches back together.

3.3. Improved YOLOv8

The CBL and C2f modules together form the feature extraction backbone network of YOLOv8. However, when it comes to deployment in intelligent driving systems, the original version still has shortcomings. In this section, we will discuss several network structure optimization strategies. The goal of these strategies is to substantially reduce the parameter count while maintaining the model's original feature extraction capability. The model will thus be better able to balance the dual demands of speed and accuracy in practical applications.

3.3.1. Parallel Network

ParNet [20] proposes a network construction approach that differs from traditional methods. Instead of stacking layers vertically to increase depth, it arranges them horizontally to increase breadth. Figure 2 shows how the network is built. The key idea is to use several parallel processing streams. These streams run independently and only communicate, or merge, at the beginning and the end of the network. Each stream works on feature maps at different resolutions. With this parallel design, ParNet manages to cut the network depth down to just 12 layers. For comparison, mainstream models like ResNet usually go 50, 101, or even deeper.

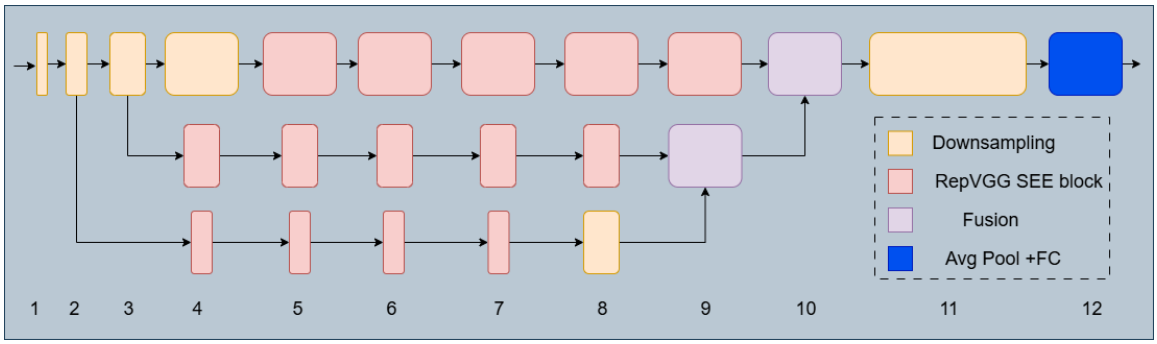


Figure 2. Parnet network architecture.

The ParNet Block employs an enhanced RepVGG block, which features complex architecture during training (with parallel branches) but can be "reparameterised" into a simpler structure during inference. During training, multiple branches may be utilised on a 3×3 convolutional block. Once trained, these multiple branches can be fused into a single 3×3 convolution. Consequently, the final architecture comprises only 3×3 blocks and nonlinearities.

This reparameterisation or block fusion reduces latency during inference. To address the limited receptive field of shallow networks, ParNet introduces the Skip-Squeeze-Excitation (SSE) module upon the SE attention

mechanism. Without increasing depth, it adaptively adjusts channel weights to expand the effective receptive field, as illustrated in Figure 3.

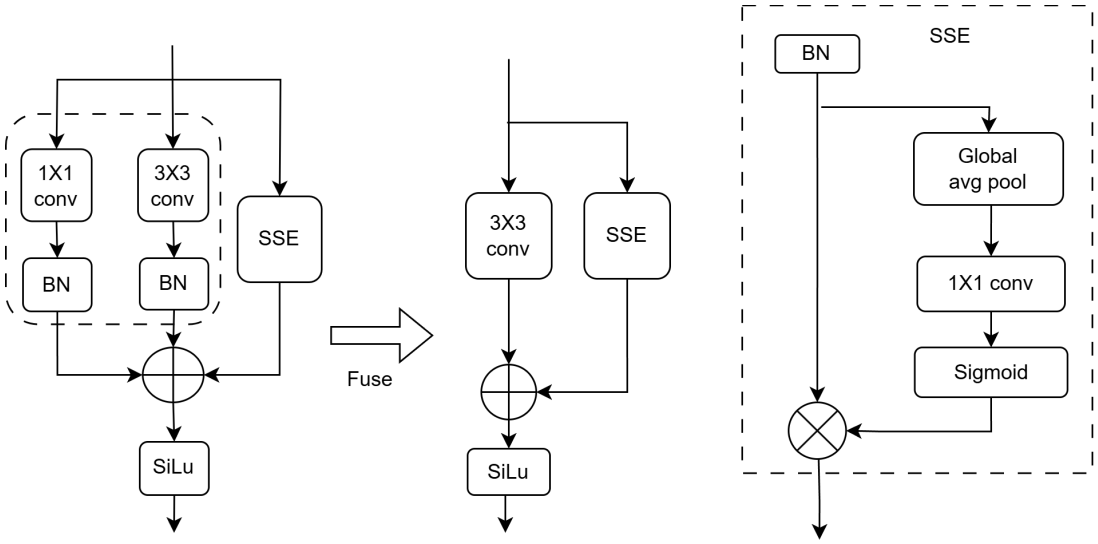


Figure 3. ParNet block.

As a non-deep network comprising only 12 layers, ParNet inherently lacks the multi-layer nonlinear stacking characteristic of deep networks. SiLU compensates for this deficiency in nonlinearity without increasing depth, while also circumventing the limitations ReLU imposes on gradient propagation in shallow networks. When combined with the SSE module, it further enhances model performance. Consequently, this paper employs SiLU in place of ReLU. The SiLU formula is as follows:

$$F(x) = x \cdot \text{sigmoid}(x) \tag{1}$$

In addition to RepVGG-SSE (whose inputs and outputs share identical dimensions), ParNet incorporates a fusion module and a downsampling module. The fusion block combines information from multiple resolutions, whilst the downsampling block reduces the resolution and increases the width to enable multi-scale processing, as illustrated in Figure 4 below:

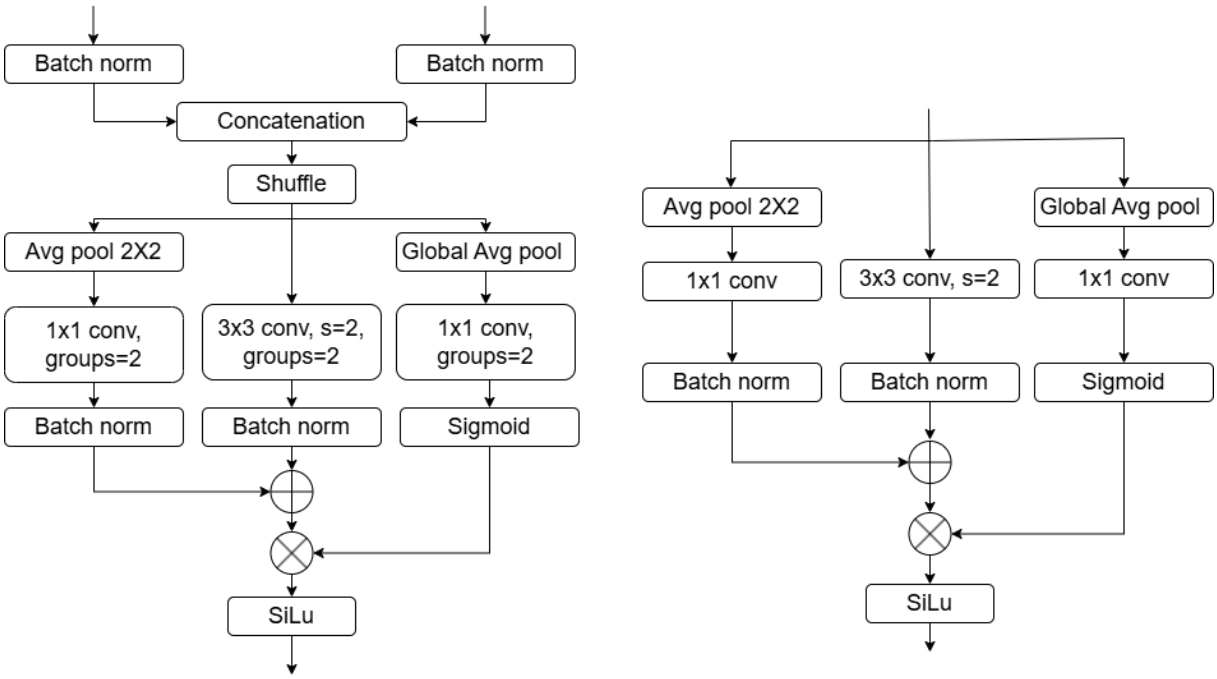


Figure 4. Fusion module (**left**) and downsampling module (**right**).

Fusion Module: This is employed to fuse information from different resolutions. It resembles the Downsampling Module but incorporates an additional concatenation layer. Beyond concatenation, it possesses a greater number of input channels. Consequently, to reduce the number of parameters, we adopt batch convolutions with groups = 2.

Downsampling Module: This reduces resolution while increasing width to facilitate multi-scale processing. It similarly omits the concatenation branch, instead incorporating a single-layer SE module (Global avg pool, 1 × 1 conv, Sigmoid) parallel to the convolution branch. Additionally, 2D average pooling (Avg pool 2 × 2) is added to the convolution branch.

Following this introduction to the ParNet architecture, this paper integrates it with YOLOv8's C2f layer to form C2f_ParNet. The critical "RepVGG-SSE block" within ParNet is encapsulated into a module compatible with the YOLOv8 framework. Results from the improved backbone network are presented in Figure 5.

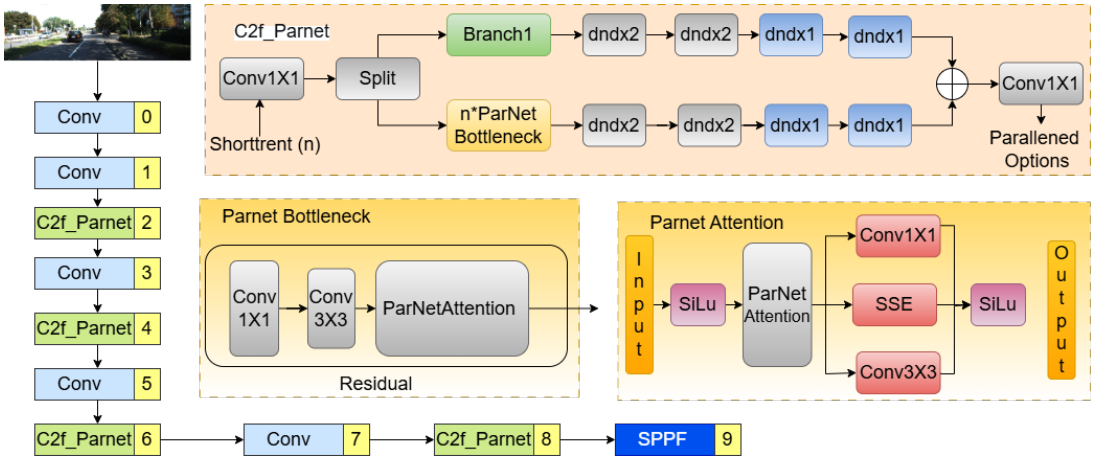


Figure 5. C2f_Parnet network architecture.

This diagram illustrates the position of C2f_ParNet within the backbone network and its architectural structure. The upper layer depicts the primary workflow of C2f_ParNet, where the input Shorttrent(n) undergoes initial channel consolidation via Conv1 × 1. Subsequently, the features are split into two branches through Split: Branch1 (original feature branch, retained directly); and n*ParNet Bottleneck (parallel enhancement branch, processed through multiple ParNet Bottlenecks). The outputs from both Branch1 and n-ParNet Bottleneck undergo a series of dndx2/dndx1 operations (i.e., dimension adjustment/downsampling to unify feature dimensions and channel counts), followed by a final channel merging via Conv1 × 1. This ultimately outputs the feature Parallened Options.

The feature enhancement part of C2f_ParNet is implemented by the ParNet Bottleneck submodule. This module starts with convolutional preprocessing. The Conv1 × 1 layer reduces the channel number of the input features. The Conv3 × 3 layer extracts spatial structure information from the features. The preprocessed features are sent to the ParNetAttention module. This module has two main design elements. One is the multi-branch structure. The other is the attention mechanism. The two work together to enhance the expression of important features. The output of ParNetAttention is fused with the original input features via residual connections (Residual), preventing gradient vanishing and enhancing training stability (corresponding to the shortcut parameter in the code).

ParNetAttention forms the core of ParNet. Input features first pass through the SiLU activation function to enhance non-linear expression. The activated features are then fed into ParNetAttention, comprising three parallel branches: Conv1 × 1 (integrating channel features and compressing redundancy), Conv3 × 3 (expanding the receptive field to extract spatial features), and SSE attention (allocating channel weights to enhance key channels). After fusing the three branch features, we apply SiLU activation again to finish the feature enhancement.

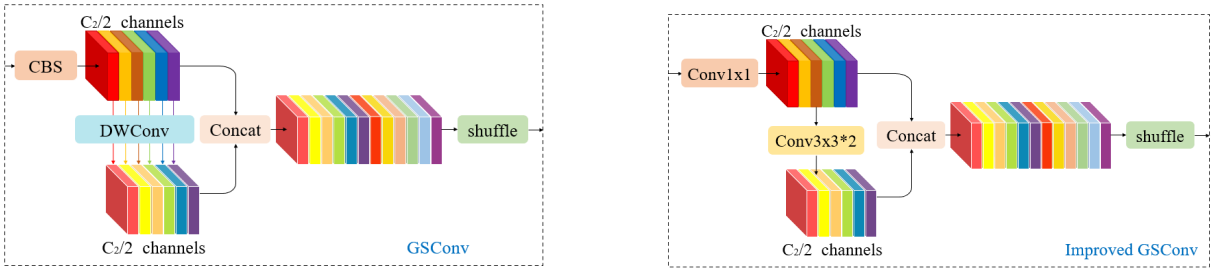
3.3.2. Slim-neck

Slim-neck [21] introduces a way to connect the Neck layer that's designed to hold onto as much information as possible during compression, while keeping time complexity low. GSConv, a core component of Slim-neck, is structured as depicted in Figure 6a. GSConv employs standard convolutions to halve the number of channels in the input feature map, subsequently merging it with the feature map generated by the DWConv operation. Finally, it performs channel reshuffling to preserve latent connections between potentially severed channels through sparse convolutions, thereby maintaining a degree of feature extraction capability. The time complexity of GSConv is

calculated as follows:

$$W \cdot H \cdot K_1 \cdot K_2 \cdot \frac{C_2}{2} \cdot (C_1 + 1) \tag{2}$$

where W and H denote the width and height of the output feature map respectively, while C_1 and C_2 denote the number of channels in the input and output feature maps respectively. Additionally, K_1 and K_2 denote the size of the convolutional kernel respectively. Compared to standard convolution, this requires lower time complexity.



(a) **(b)**

Figure 6. Structure of GSConv. (**a**) denotes the original connection; (**b**) denotes replacing the 5 × 5 convolution with two 3 × 3 convolutions.

Research has revealed that within the GSConv module, the feature map first undergoes a convolution with a kernel size of 5 and a stride of 1, followed by a concatenation operation. A 5 × 5 convolution effectively extracts semantic information from feature maps. However, concatenating the outputs of two 3 × 3 convolutions achieves equivalent results while employing fewer parameters. Based on this analysis, two 3 × 3 convolutions can reduce network parameters while preserving feature extraction capabilities. The original GSConv module used a 5 × 5 convolution. This design incurred high parameter count and computational cost. We modified it. We replaced the 5 × 5 convolution with two 3 × 3 convolutions. Both convolutions share a kernel size of 3 × 3. After the replacement, parameter count and computational burden are significantly reduced. Figure 6b presents the structural layout of the modified module. During implementation, we split the input tensor into two branches. Each branch passes through a 3 × 3 convolutional layer. The outputs from both branches are merged at the end. Overall computational efficiency is therefore improved. This can be written as:

$$x_{\text{out}} = \text{concat}(x_1, x_2, \text{dim} = 1) \tag{3}$$

$$y = \text{permute}\big(\text{reshape}(x_{\text{out}})\big) \tag{4}$$

$$\text{output} = \text{concat}(y[0], y[1], \text{dim} = 1) \tag{5}$$

The first branch adjusts the number of channels via a 1 × 1 convolutional layer, yielding x_1. Simultaneously, the second branch first undergoes a 1 × 1 convolutional layer to adjust the channel count, then feeds the output into two serial 3 × 3 convolutional layers to extract features, resulting in x_2. Subsequently, x_1 and x_2 are concatenated to produce x_{out}. Finally, channel replacement operations are performed on x separately, and the resulting tensors are summed to yield the final output. Building upon GSConv, Slim-Neck further introduces the GSBottleneck module, inspired by bottleneck architecture. Moreover, it incorporates the concept of aggregation by consolidating features from all layers at the module's final stage, thereby addressing inefficiencies arising from dense connections. Its structure is illustrated in Figure 7.

3.3.3. Research on Pruning Algorithms Based on L2 Scores

Currently, the most widely adopted pruning method is network slimming. Network slimming employs L1 regularisation to sparsify the training model, utilising the γ parameter from the Batch Normalisation (BN) layer as the pruning ratio factor. Following sparsity training, the majority of these ratio factors become sparse. Channels within the sparsely trained model are ranked according to their importance before pruning, thereby achieving network size reduction. Although slimming methods are currently relatively straightforward to implement, they carry a significant risk of erroneously removing crucial neurons or connections, leading to a marked decline in model

performance. Furthermore, continuous tuning of sparsity rate hyperparameters during sparse training is required to achieve optimal pruning results. Evci et al. [22] proposed the RigL pruning theory, suggesting that selecting specific layers across multiple neural network layers for weight pruning yields the most pronounced pruning effects. The RigL pruning theory is illustrated in Figure 8.

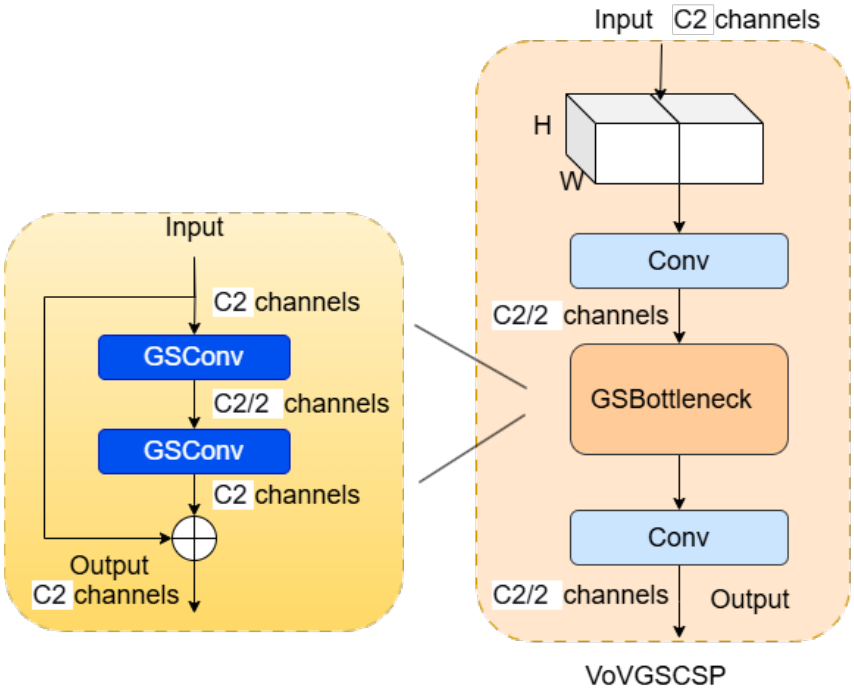


Figure 7. VoVGSCSP comprises GSConv that mimics residual structures.

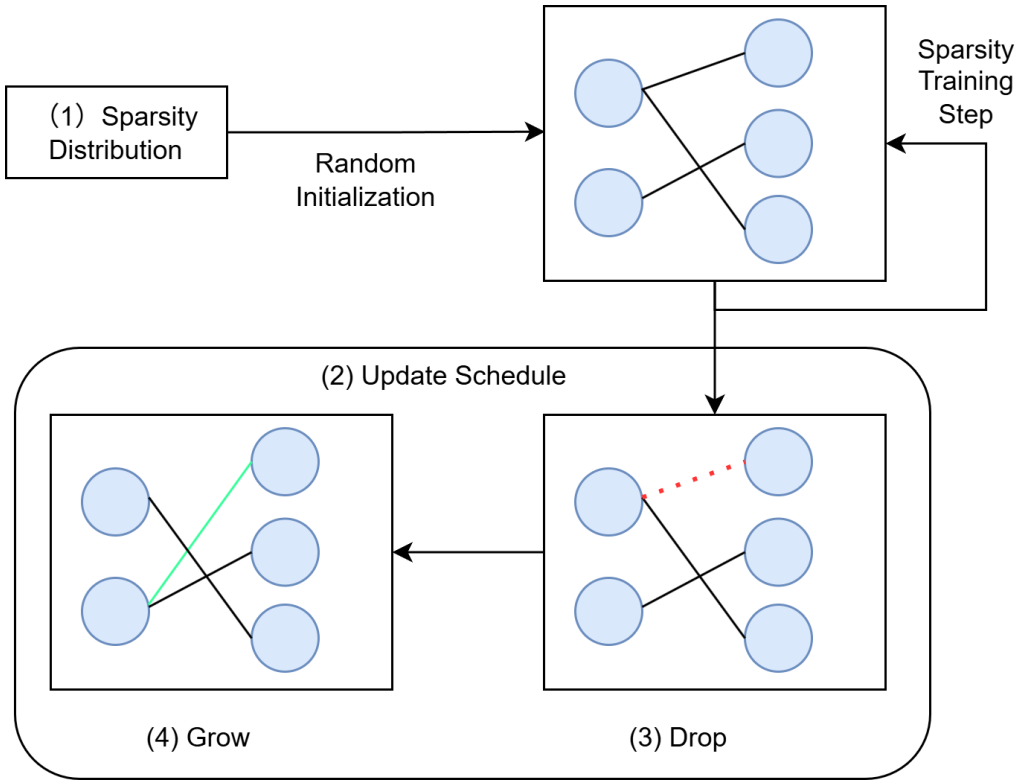


Figure 8. Rigl pruning theory.

RigL initialises the network using a random sparse topology. It periodically removes a proportion of the least significant connections based on the number and magnitude of their weights, whilst determining whether to add

new connections according to gradient information. Should a connection exhibit a substantial gradient, indicating significant contribution to the model, it is reinstated for further training and undergoes pruning iterations post-trimming. Gale et al. [23] took a close look at weight-based pruning by testing a wide range of hyper parameters. Experiments reveal that some structurally simple layer-level pruning techniques actually outperform algorithms with more complex designs. However, how to determine which layers should be pruned remains a non-negligible issue.

L2-score pruning [24] is a classic algorithm in structured channel pruning. Its core logic is as follows. Each output channel in a convolutional layer corresponds to an independent set of weights with a tensor shape of [out_channel, in_channel, k, k]. A smaller L2 norm of the channel weights indicates lower overall magnitude. This means the channel contributes less to the network's final output. Zeroing out or directly removing the weights of these "low-contribution" channels can effectively reduce FLOPs and parameter count. Model accuracy is not significantly affected. The inference speed of the model is therefore improved. The L2 norm is calculated as follows:

$$\left\|F_i^l\right\|_2 = \sqrt{\sum_{n=1}^{N_l}\sum_{k_1=1}^{K_1}\sum_{k_2=1}^{K_2}\left|F_i^l(n,k_1,k_2)\right|^2} \tag{6}$$

where: F_i^l denotes the i-th output channel (filter) of the l-th layer; N_l denotes the number of input channels in the l-th layer; K_1 and K_2 denote the kernel sizes.

3.3.4. EIoU Loss Function

Traditional IoU merely calculates the intersection-over-union ratio between predicted and ground-truth bounding boxes, yielding zero gradients and failing regression when boxes lack overlap. EIoU [25] decouples and optimises the aspect ratio loss in CIoU by directly regressing actual width and height, thereby enhancing convergence speed and regression accuracy. The formula is as follows:

$$\text{EIoU} = 1 - \text{IoU} + L_{\text{dist}} + L_{\text{wh}} \tag{7}$$

$$\text{IoU} = \frac{|B_p \cap B_g|}{|B_p \cup B_g|} \tag{8}$$

$$d = \sqrt{(x_p - x_g)^2 + (y_p - y_g)^2} \tag{9}$$

$$d_c = \sqrt{w_c^2 + h_c^2} \tag{10}$$

$$L_{\text{dist}} = \frac{d^2}{d_c^2} \tag{11}$$

$$L_{\text{wh}} = \frac{(w_p - w_g)^2}{w_c^2} + \frac{(h_p - h_g)^2}{h_c^2} \tag{12}$$

where, $B_p = (x_p, y_p, w_p, h_p)$ denotes the predicted bounding box, and $B_g = (x_g, y_g, w_g, h_g)$ denotes the ground truth bounding box, where x and y represent the centre coordinates, and w and h represent the width and height, respectively. d denotes the centre distance, d_c denotes the diagonal length of the minimum bounding rectangle, and w_c and h_c denote the width and height of the minimum bounding rectangles of the predicted and ground truth bounding boxes, respectively. L_{dist} denotes the centre distance loss, and L_{wh} denotes the width-height loss.

3.3.5. Modified Model Structure

Following the replacement of the backbone and neck networks, along with modifications to the loss function and pruning, the number of model parameters decreased by 1%, with GFLOPs dropping from 28.5 to 26.2. The revised network architecture is illustrated in Figure 9.

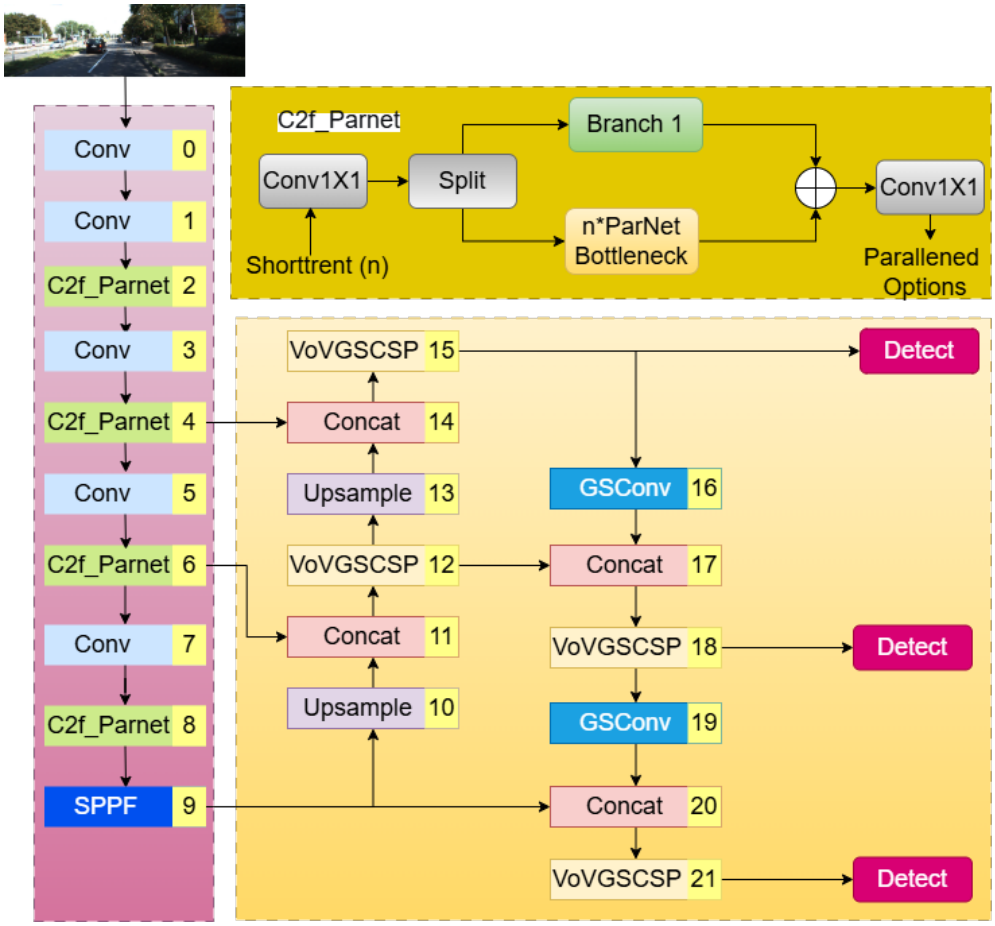


Figure 9. Modified network architecture.

4. Experiment

4.1. Dataset

This paper validates the effectiveness of the improved YOLOv8s model using the KITTI [26] public dataset. The KITTI dataset was jointly released by the Karlsruhe Institute of Technology in Germany and the Toyota Research Institute in the United States. It is considered one of the largest datasets globally for evaluating autonomous driving vision algorithms. The dataset contains data captured under various weather conditions across multiple real-world road scenarios, including urban areas, highways, and villages. Additionally, objects within images exhibit varying degrees of occlusion and cropping. Images have a resolution of 1272 × 375 pixels, with labels comprising pairs such as cars, trucks, vans, pedestrians, and cyclists. Figure 10 provides a snapshot of the data.

From this public dataset, this paper extracted 7481 image samples covering seven categories: pedestrians, trucks, cars, vans, trams, cyclists, and seated individuals. Furthermore, the dataset was split into training and validation sets at an 8:2 ratio. The number of targets per image varies. The specific counts of category labels are shown in Table 1.

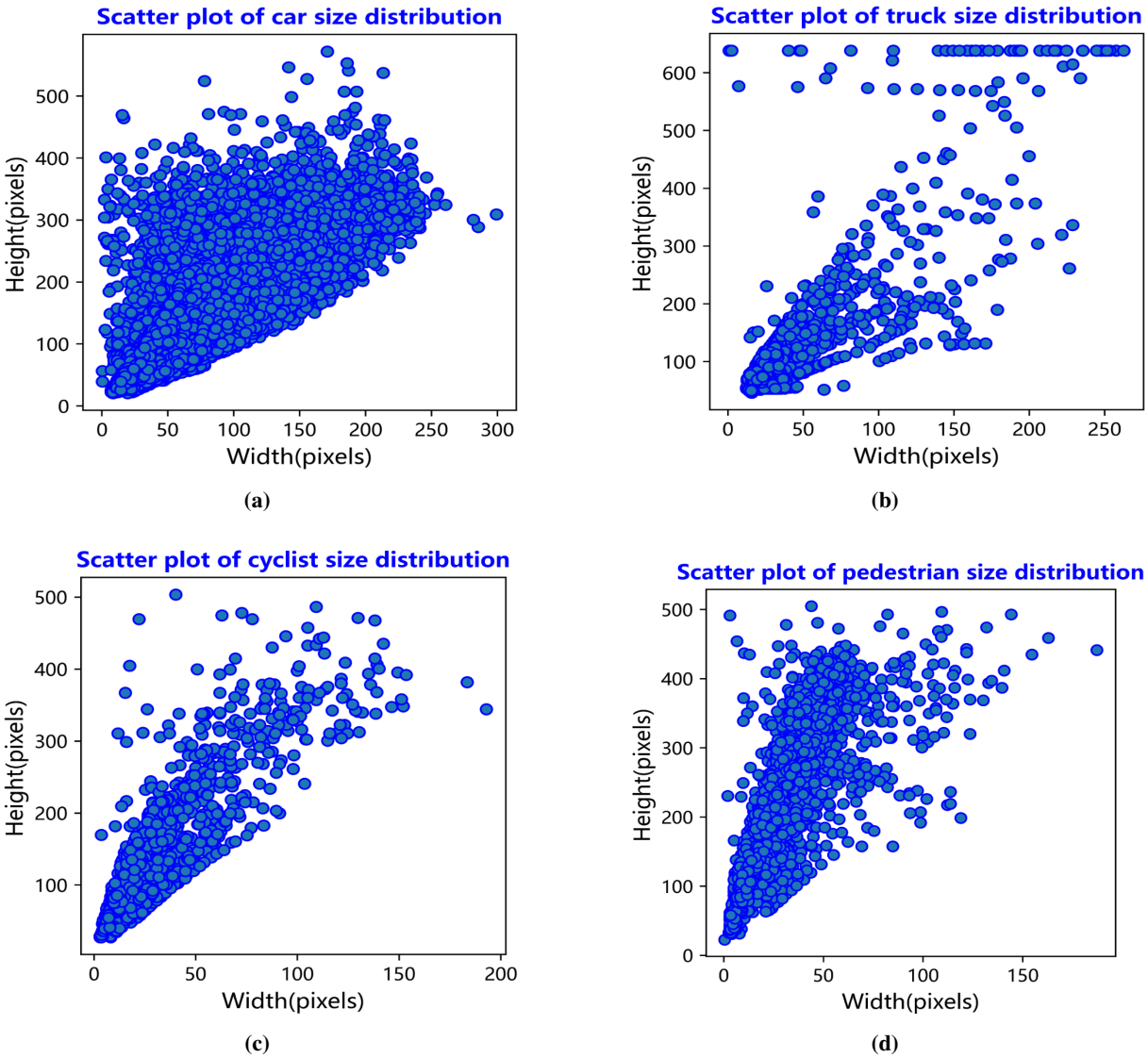
Table 1. Specific counts for category labels.

Label	Number
Pedestrian	4487
Truck	1094
Car	28,742
Cyclist	1627
Van	2914
Tram	511
Person_sitting	222

The height and width distributions of some bounding boxes are illustrated in Figure 11. Pedestrians and cyclists usually have much higher aspect ratios than cars and trucks. Their detection boxes look like tall, narrow rectangles, while those for cars and trucks are closer to squares. This difference means that in crowded scenes, it's harder to detect pedestrians and cyclists than it is to detect motor vehicles.

(a) **(b)**

Figure 10. Snapshots of sample data: (**a**) Snapshot 1; (**b**) Snapshot 2.



(a) **(b)** **(c)** **(d)**

Figure 11. Scatter plots of object bounding box size distribution: (**a**) car; (**b**) truck; (**c**) cyclist; (**d**) pedestrian.

4.2. Experimental Environment and Training Strategy

This experiment was conducted on the Windows 11 operating system with an NVIDIA RTX 3060 GPU. The software framework was configured with PyTorch 2.7.1 and CUDA 11.8. The SGD algorithm was selected for optimization. Among the training-related parameters, batch size was set to 8 and the total number of training epochs was 200. The initial learning rate was determined to be 0.01, with the aim of accelerating the model convergence process.

4.3. Experimental Results and Analysis

To evaluate the model's detection performance, we selected several commonly used metrics in the field of object detection. These specifically include Precision, Recall, mAP@50, and mAP@50-95.

In object detection, Precision measures the proportion of correct predictions made by the model. The formula is as follow:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{13}$$

where TP stands for the positive samples that the model correctly identifies, while FP is the positive samples it gets wrong.

Recall is the fraction of true positive samples that the model successfully detects. The formula is as follow:

$$\text{Recall} = \frac{TP}{TP + FN} \tag{14}$$

where FN stands for the true targets that the model failed to catch.

Mean Average Precision (mAP) is the average of the precision scores across all categories. The formula is as follow:

$$\text{mAP} = \frac{\sum_{k=1}^{K} \text{AP}_k}{K} \tag{15}$$

where K is the total number of categories in the dataset.

In addition to mAP, two other factors are important for evaluating model performance: the number of parameters and FLOPs. Parameters refer to the total number of trainable variables in the model and are commonly used as a measure of model size. FLOPs measure the computational cost of the model during inference. Their formula are as follows:

$$\text{Parameters} = C_{\text{out}} \times K_w \times K_h \times C_{\text{in}} + C_{\text{out}} \tag{16}$$

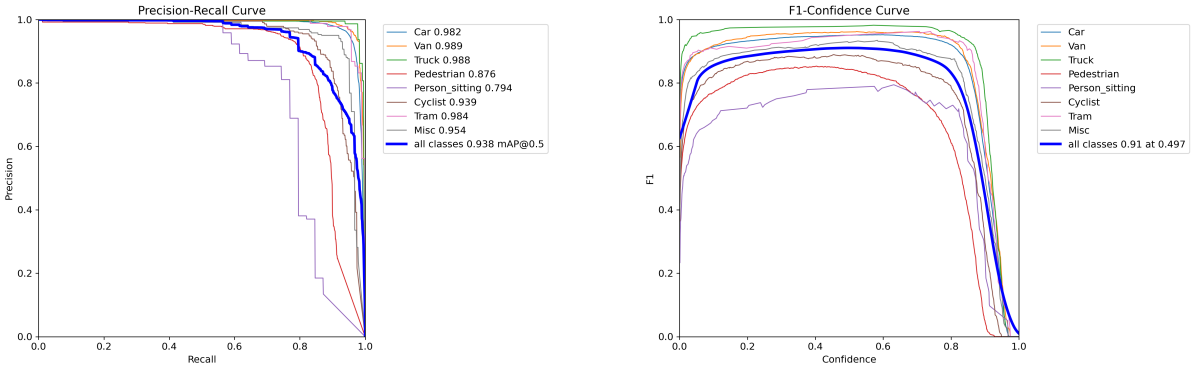
$$\text{FLOPs} = 2 \times C_{\text{out}} \times H_{\text{out}} \times W_{\text{out}} \times (C_{\text{in}} \times K_w \times K_h) \tag{17}$$

where K_w and K_h denote the convolution kernel width and height, and C_{in} and C_{out} denote the number of input and output channels, H_{out} and W_{out} denote the height and width of the output feature map respectively.

4.4. Evaluation Indicators

4.4.1. Original Model Results Curves

At the start of the experiment, the original model was trained using the KITTI dataset. Figure 12 displays the PR curves and F1 curves for the original model across each category.



(a) **(b)**

Figure 12. YOLOv8s's PR curve (**a**) and F1 curve (**b**).

As shown in Figure 12, the model's recall remains above 0.8 within the 0.0 to 0.8 confidence interval, while its precision consistently stays above 0.8. The F1 curve demonstrates an F1 score exceeding 0.9 across the entire 0.0 to

0.8 confidence range. This indicates that the model exhibits excellent overall performance under most practical confidence thresholds.

4.4.2. GSConv Comparison Experiments

To validate the effectiveness of the improved GSConv, the pre-improved GSConv was compared with the improved GSConv. The comparison results are shown in Table 2.

Table 2. Comparative experiments of improved GSConv.

Model	mAP50	mAP50-95	Parameters	GFlops
GSConv	0.922	0.704	981 4568	25.1
Improved GSConv	0.908	0.689	934 6984	22.9

As shown in Table 2, the improved GSConv requires fewer parameters while achieving nearly identical results, validating the effectiveness of the enhancement.

4.4.3. Comparison of Different Pruning Methods

To validate whether the L2-score-based pruning method outperforms the Network Slimming approach based on sparsity, pruning, and fine-tuning, we compared the former with various commonly used pruning schemes. The comparison results are shown in Table 3. In the table: Growing_reg employs L2 regularization sparse training with dynamically varying penalty terms; Slim represents the classic network slimming pruning method; Group_slim combines L1 and dual-norm regularization; Group_hessian utilizes second-order derivative information to balance network complexity and training set error during pruning.

Table 3. Comparison of different pruning methods.

Methods	mAP50	mAP50-95
L2	0.944	0.753
Growing_reg	0.938	0.757
Grphp_slim	0.935	0.750
Group_hessian	0.936	0.749
Random	0.935	0.750
Lamp	0.936	0.749

As shown in the table, the L2-pruned YOLOv8s achieves an mAP50 value of 0.944, maintaining the highest recognition accuracy.

The superiority of the L2 pruning algorithm is also evident from the comparison of accuracy, recall, and mAP during fine-tuning training. The comparison results are shown in Figure 13.

4.4.4. Pruning Experiment

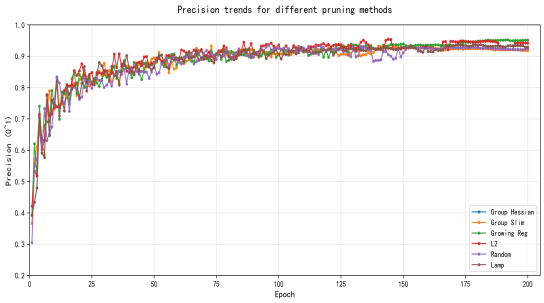
To determine the optimal pruning rate, we conducted pruning experiments with varying compression levels. The GFLOPs and mAP results are shown in Figure 14.

Through experimentation, we found that setting the pruning rate to 1.5 enables the fine-tuned model to achieve higher accuracy in the mAP50-95 metric. This demonstrates that the L2 pruning algorithm effectively avoids severing critical internal connections within the model, thereby ensuring lossless pruning at lower pruning rates. This underscores the algorithm's outstanding performance in compressing YOLOv8 models. Consequently, we set the compression ratio to 1.5 for all subsequent experiments to optimize model performance.

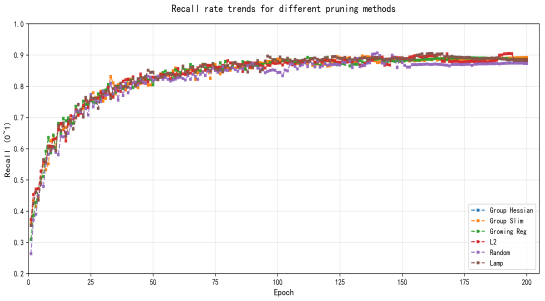
4.4.5. Ablation Experiment

To evaluate the impact of lightweighting on backbone networks, neck networks, and trunks, as well as on GFLOPs and mAP, ablation experiments are conducted. The lightweight techniques were introduced in a step-by-step manner. The performance changes after each adjustment were recorded in detail. All comparative results are summarized in Table 4. Model1 in the table represents the unmodified original YOLOv8s. This model serves as the benchmark reference for the entire experiment.

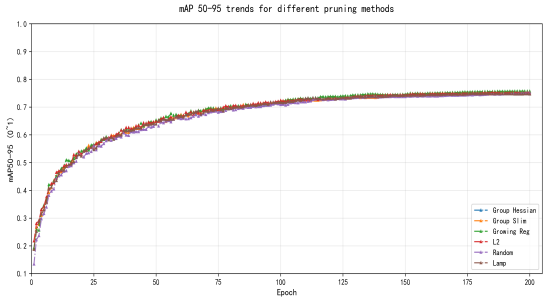
The ablation experiment results provide several important insights. The adoption of C2f_Parnet and the improved GSConv improves model efficiency but leads to a certain degree of performance degradation. The effect of L2 pruning is the opposite. It brings accuracy and mAP50 back up while further reducing computational cost. The synergy of the three strategies ultimately gives rise to Model6. This model achieves optimal performance in the trade-off between accuracy and efficiency. Its precision of 0.942 ranks first among all models and is nearly identical to the baseline. The only perceptible performance degradation is a slight drop in mAP50-95. Core evaluation metrics remain largely at their original levels. Model efficiency gains a substantial improvement. These advantages make Model6 more competitive in real-world engineering deployment.



(a) Precision



(b) Recall



(c) mAP50-95

Figure 13. Evaluation metrics for various pruning methods.

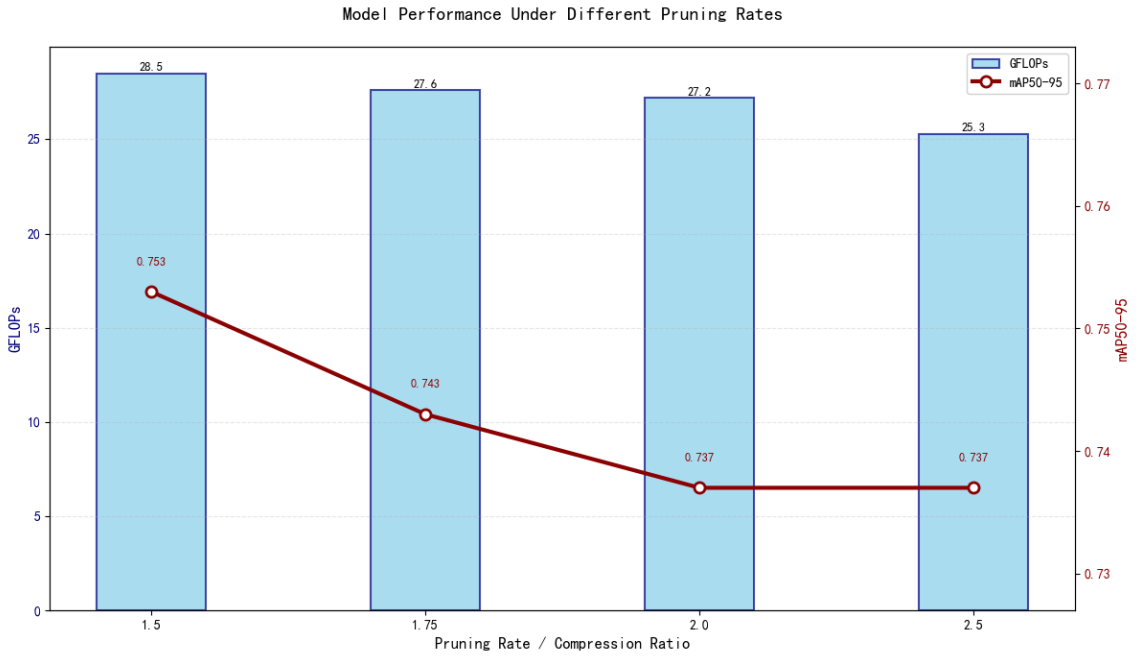


Figure 14. GFLOPs and mAP at different compression ratios.

Table 4. Ablation experiments.

Model	C2f Parnet	Improved GSConv	L2	EIoU	P	R	mAP 50	mAP50-95	Weights (MB)
Model1	×	×	×	×	0.930	0.894	0.939	0.761	22.5
Model2	✓	×	×	×	0.912	0.874	0.927	0.722	24.9
Model3	×	✓	×	×	0.912	0.841	0.908	0.689	19.9
Model4	✓	✓	×	×	0.894	0.865	0.911	0.699	22.4
Model5	✓	✓	✓	×	0.939	0.848	0.921	0.699	20.4
Model6	✓	✓	✓	✓	0.942	0.840	0.917	0.693	20.4

Figure 15 presents the changes in channel counts across layers before and after pruning. Layer 5, a high-channel layer, decreases from 1024 to 341 channels. Medium-channel layers like Layers 4, 6, 10 and 14 dropped from 512 to 170. Low-channel layers also experience reductions of similar magnitude. This distribution pattern indicates that pruning is applied to all layers, and the absolute reduction magnitude is positively correlated with the original channel count of each layer. The compression ratio for all layers stabilised around 0.33 (with no significant fluctuations), indicating that the pruning strategy uniformly compresses each layer rather than selectively targeting certain layers. This prevents excessive compression in individual layers, thereby avoiding the collapse of the model's feature extraction capabilities.

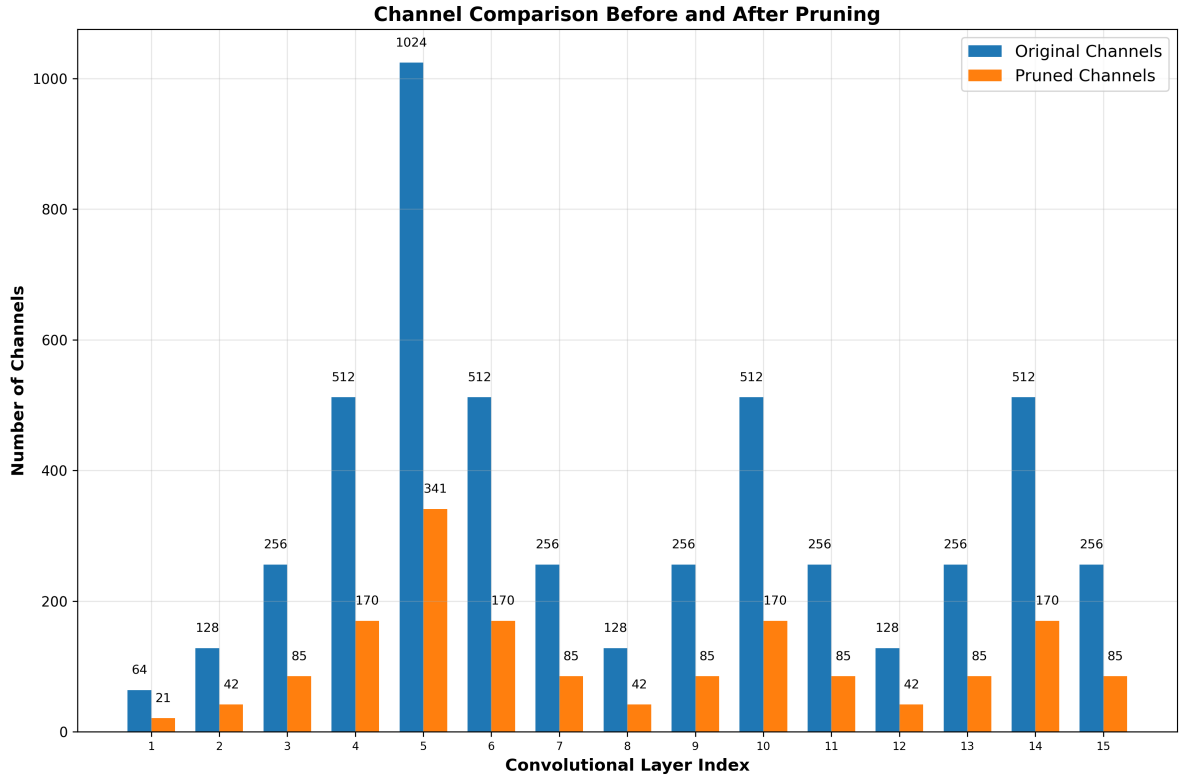


Figure 15. Channel comparison diagram.

The algorithm performed detailed statistical analysis of detection performance for each category on the validation set, with results summarised in Table 5. These results validate the algorithm's performance in detecting vehicle and pedestrian targets. However, the model exhibits relatively low recall rates for pedestrians, cyclists, and seated individuals (person_sitting), consistent with analyses in the dataset description. Given the relatively narrow contour widths of pedestrians, cyclists, and seated individuals, the model may omit certain targets during detection.

Table 5. Test results.

Class	Images	InstancesBox	P	R	mAP50	mAP50-95
All	1497	8231	0.942	0.840	0.917	0.693
Car	1338	5889	0.965	0.911	0.973	0.835
Van	427	570	0.961	0.918	0.971	0.904
Truck	220	227	0.982	0.956	0.982	0.843
Pedestrian	359	916	0.948	0.674	0.837	0.509
Person_sitting	20	39	0.834	0.692	0.785	0.472
Cyclist	222	308	0.944	0.779	0.884	0.623
Train	63	95	0.955	0.937	0.977	0.745
Misc	156	187	0.947	0.850	0.929	0.710

4.4.6. Improved Model Testing and Comparative Experiments

The performance of the improved model is compared with that of the original model. Figure 16a shows the original image, Figure 16b displays the detection results from the improved model, and Figure 16c presents the detection results from the original model.

(a) **(b)** **(c)**

Figure 16. Some test results show: (**a**) the original image; (**b**) results representing the model described herein; (**c**) originating from the original model.

Finally, this section compares the improved model with other widely used object detection models, as well as models proposed by other researchers on the KITTI dataset. The specific comparison results are shown in Table 6.

Table 6. Detection results of various models.

Model	mAP50	GFLOPs	Weights (MB)
SSD	0.724	183.4	52.6
Faster RCNN	0.847	194.3	83.0
YOLOv3	0.882	154	123.5
YOLOv5s	0.913	15.8	14.4
Mask RCNN	0.879	128.4	74.2
YOLOv11s	0.923	21.3	19.2
Our	0.943	26.2	20.4

It is evident that the model presented herein achieves optimal detection accuracy (0.943 mAP50) on the KITTI dataset. This approach incurs a modest computational overhead of 26.2 GFLOPs but achieves a 2% performance gain over the latest YOLOv11s model. This represents a distinct advantage for applications where accuracy is the primary concern.

5. Conclusions

In this paper, we looked at several ways to make YOLOv8s lighter and more efficient for detecting vehicles and pedestrians in autonomous driving scenarios. We introduced the C2f_Parmetre module to rebuild the backbone network, turning the usual depth-stacking approach into parallel processing streams. We improved GSConv through a kernel decomposition strategy and compressed the parameter count. On this basis, L2-score pruning was adopted to remove redundant channels according to the L2 norm of channel weights, while avoiding the cutting of important feature connections. The EIoU loss function further assisted in improving regression accuracy and accelerating convergence.

The model still has deficiencies in recall, mainly manifested as missed detections of strip-shaped targets such as pedestrians and cyclists. The root cause of this problem may lie in the physical characteristics of these targets and their occurrence patterns in the training data. Future work could focus on two directions: attention mechanisms and multi-scale feature fusion, to enhance the model's adaptability to small targets and occlusion scenarios. Deployment optimization on embedded platforms should not be overlooked either, as it will provide support for efficient end-to-end inference.

Author Contributions

J.P.: conceived and designed the study, supervised the project, and secured funding; W.W.: refined the methodology based on Y.H.'s initial work, performed software development, validation, and formal analysis, and drafted the manuscript; Y.H.: conducted the preliminary research and established the foundational implementation; M.Z. and S.J.: contributed to discussions and funding acquisition; J.L.: supervised the project. All authors have read and agreed to the published version of the manuscript.

Funding

This research received partial funding from several sources: the Natural Science Foundation of Chongqing (Grant No. CSTB2024NSCQ-MSX1087), the Science and Technology Research Program of the Chongqing Municipal Education Commission (Grants No. KJQN202501553 and KJQN202501544), the CQUST (No. CKRC20231224), and the Chongqing Municipal Key Lab of Autonomous AI Fundamental Theories and New Intelligent Industrial Innovation.

Institutional Review Board Statement

Not applicable.

Informed Consent Statement

Not applicable.

Data Availability Statement

The dataset used in this study is publicly available. The KITTI dataset can be accessed at: http://www.cvlibs.net /datasets/kitti.

Conflicts of Interest

The authors declare no conflict of interest.

Use of AI and AI-Assisted Technologies

No AI tools were utilized for this paper.

References

1. Zhou, Y.; Hu, D.; Wang, D.; et al. Review on Research and Application of Deep Learning-Based Target Detection Algorithms. *Comput. Eng. Appl.* **2023**, *59*, 1–13. (In Chinese)
2. Girshick, R.; Donahue, J.; Darrell, T.; et al. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. In Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, 23–28 June 2014; pp. 580–587.
3. Liu, W.; Anguelov, D.; Erhan, D.; et al. SSD: Single Shot Multibox Detector. In *Computer Vision—ECCV 2016*; Springer: Cham, Switzerland, 2016; pp. 21–37.
4. Redmon, J.; Divvala, S.; Girshick, R.; et al. You Only Look Once: Unified, Real-Time Object Detection. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 779–788.
5. Varghese, R.; Sambath, M. YOLOv8: A Novel Object Detection Algorithm with Enhanced Performance and Robustness. In Proceedings of the 2024 International Conference on Advances in Data Engineering and Intelligent Computing Systems (ADICS), Chennai, India, 18–19 April 2024; pp. 1–6.
6. Zhang, M.; Zhang, J.; Peng, Y.; et al. FreqDyn-YOLO: A High-Performance Multi-Scale Feature Fusion Algorithm for Detecting Plastic Film Residues in Farmland. *Sensors* **2025**, *25*, 4888.
7. Xie, Y.; Du, D.; Liu, Y. RE-YOLO: Vehicle Detector Based on Knowledge Distillation YOLOv8 for Autonomous Driving Scenarios. *J. Comput. Civ. Eng.* **2025**, *39*, 04025084.
8. Duan, H.; Zhang, Y.; Zhu, S.; et al. Lightweight Multi-Scale Vehicle Object Detection Algorithm Based on Optimized YOLOv8. *Pattern Anal. Appl.* **2025**, *28*, 183.
9. Zhao, Z.; Chen, S.; Zhang, P.; et al. Research on the Construction of Real-Time Vehicle Detection Model Based on YOLOv8-BASW. *IEEE Access* **2025**, *13*, 183519–183533.
10. Li, H.; Li, Y.; Xiao, L.; et al. RLRD-YOLO: An Improved YOLOv8 Algorithm for Small Object Detection from an Unmanned Aerial Vehicle (UAV) Perspective. *Drones* **2025**, *9*, 293.
11. Shao, Z.; He, K.; Yuan, B.; et al. Enhanced YOLOv8 Framework for Precision Vehicle Detection in High-Resolution Remote Sensing Images. *Signal Image Video Process.* **2025**, *19*, 218.
12. Ma, X.; Zhou, M.; Zhu, H.; et al. Research on Pedestrian and Vehicle Detection Method Based on Improved YOLOv8 Model. *Signal Image Video Process.* **2025**, *19*, 1167.
13. Liu, Q.; Ye, H.; Wang, S.; et al. YOLOv8-CB: Dense Pedestrian Detection Algorithm Based on In-Vehicle Camera. *Electronics* **2024**, *13*, 236.
14. Sun, F.; Du, L.; Dai, Y. PyQt5-Powered Frontend for Advanced YOLOv8 Vehicle Detection in Challenging Backgrounds. *IET Wirel. Sens. Syst.* **2025**, *15*, e70001.
15. Lv, C.; Mittal, U.; Madaan, V.; et al. Vehicle Detection and Classification Using an Ensemble of EfficientDet and YOLOv8. *PeerJ Comput. Sci.* **2024**, *10*, e2233.
16. Bakirci, M. Advanced Aerial Monitoring and Vehicle Classification for Intelligent Transportation Systems with YOLOv8 Variants. *J. Netw. Comput. Appl.* **2025**, *237*, 104134.
17. Dou, H.; Chen, S.; Xu, F.; et al. Analysis of Vehicle and Pedestrian Detection Effects of Improved YOLOv8 Model in Drone-Assisted Urban Traffic Monitoring System. *PLoS One* **2025**, *20*, e0314817.
18. Du, D.; Xie, Y. Vehicle and Pedestrian Detection Algorithm in an Autonomous Driving Scene Based on Improved YOLOv8. *J. Transp. Eng. Part A Syst.* **2025**, *151*, 04024095.
19. Liu, Y.; Shen, S. Vehicle Detection and Tracking Based on Improved YOLOv8. *IEEE Access* **2025**, *13*, 24793–24803.
20. Goyal, A.; Bochkovskiy, A.; Deng, J.; et al. Non-Deep Networks. *Adv. Neural Inf. Process. Syst.* **2022**, *35*, 6789–6801.
21. Li, H.; Li, J.; Wei, H.; et al. Slim-Neck by GSConv: A Better Design Paradigm of Detector Architectures for Autonomous Vehicles. *arXiv* **2022**, arXiv:2206.02424.
22. Evci, U.; Gale, T.; Menick, J.; et al. Rigging the Lottery: Making All Tickets Winners. In Proceedings of the 37th International Conference on Machine Learning, Online, 13–18 July 2020; pp. 2943–2952.
23. Gale, T.; Elsen, E.; Hooker, S. The State of Sparsity in Deep Neural Networks. *arXiv* **2019**, arXiv:1902.09574.
24. He, Y.; Xiao, L. Structured Pruning for Deep Convolutional Neural Networks: A Survey. *IEEE Trans. Pattern Anal. Mach. Intell.* **2024**, *46*, 2900–2919.
25. Zhang, Y.F.; Ren, W.; Zhang, Z.; et al. Focal and Efficient IOU Loss for Accurate Bounding Box Regression. *Neurocomputing* **2022**, *506*, 146–157.
26. Geiger, A.; Lenz, P.; Stiller, C.; et al. Vision Meets Robotics: The KITTI Dataset. *Int. J. Robot. Res.* **2013**, *32*, 1231–1237.