applied sciences

MDPI

Article

TCE-YOLOv5: Lightweight Automatic Driving Object Detection Algorithm Based on YOLOv5

Han Wang [1], Zhenwei Yang [2,*], Qiaoshou Liu [2], Qiang Zhang [1] and Honggang Wang [1]

[1] State Key Laboratory of Intelligent Vehicle Safety Technology, Chongqing 401122, China; wanghan01@caeri.com.cn (H.W.); zhangqiang2@caeri.com.cn (Q.Z.); wanghonggang@caeri.com.cn (H.W.)
[2] School of Communications and Information Engineering, Chongqing University of Posts and Telecommunications, Chongqing 400065, China; liuqs@cqupt.edu.cn
* Correspondence: yzhenwei0221@163.com

Abstract: In automatic driving systems, accurate and efficient object detection is essential to ensure driving safety and improve the driving experience. However, autonomous vehicles deal with large amounts of real-time data, which places extremely high demands on computing resources. Therefore, a lightweight object detection algorithm based on YOLOv5 is proposed to solve the problem of excessive network parameters in automatic driving scenarios. Firstly, the Bottleneck convolution kernel channels in the C3 module were grouped to greatly reduce the number of parameters. Secondly, the C3 module in the neck is replaced by the Res2Net module, which extracts features at different scales through multiple branches, not only ensuring rich details, but also enhancing the generalization ability of the network. Finally, the EIOU loss function is introduced to measure the overlap between the predicted box and the real box more accurately and improve the detection accuracy. The test results of KITTI and CCTSDB2021 public traffic datasets show that compared with the original YOLOv5 model, the improved algorithm reduces the number of parameters by 20%, the calculation amount by 21%, and mAP@0.5 by 1.0%. After TensorRT optimization, the inference speed of our model on Jetson Xavier NX reaches 61 frames/s, which is 15% higher than the original YOLOv5, and satisfies the requirements of real-time detection.

Keywords: YOLOv5; lightweight; autonomous driving; Res2Net

check for updates

Academic Editors: Giorgio De Nunzio, Grigorios Beligiannis and Paolino Di Felice

Received: 10 January 2025
Revised: 27 April 2025
Accepted: 30 April 2025
Published: 27 May 2025

Citation: Wang, H.; Yang, Z.; Liu, Q.; Zhang, Q.; Wang, H. TCE-YOLOv5: Lightweight Automatic Driving Object Detection Algorithm Based on YOLOv5. *Appl. Sci.* **2025**, *15*, 6018. https://doi.org/10.3390/app15116018

Copyright: © 2025 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

With the rapid development of the economy and the continuous improvement of people's pursuit of high-quality life, the demand for automobiles continues to rise, and automobile production also increases [1]. In the context of traffic congestion and frequent accidents, autonomous driving technology [2] has become a major topic of common concern for the government and enterprises.

Early in the history of autonomous driving technology, automatic driving systems relied on accurate sensor data, the parameter configuration often needed to be manually set by the developer, and it needed to pass field tests and repeated debugging [3]. Traditional methods have several disadvantages, the main challenge being that manually adjusted parameter configurations are inherently labor-intensive and have limited adaptability to new applications [4]. With the rapid development of artificial intelligence [5], machine learning [6], mobile internet [7] and other technologies, autonomous driving technology has made remarkable progress. As an important branch of artificial intelligence, deep learning [8] can extract useful feature information from driving data by building a complex

neural network model, achieve accurate positioning of vehicles, behavior prediction and other functions, greatly improve the accuracy and speed of object detection, and promote the development of automatic driving systems to a more intelligent direction [9].

Object detection is an important part of automatic driving systems. It is designed to identify specific objects in an image or video and determine the location of those objects. However, in practical applications, due to the different appearance and shape of objects, as well as the interference of lighting, occlusion and shooting angle, object detection is still facing an enormous challenge [10]. Traditional object detection methods usually extract features from areas that may contain objects, such as scale-independent feature transform (SIFT) [11], gradient-oriented graph (HOG) [12] and other feature extraction algorithms. However, these methods require a large computational load and have slow operation speeds, and the recognition effect is not good enough with low accuracy. Recent advances in deep learning have revolutionized the field of object detection. By using convolutional neural networks (CNN), the system can now extract features from images in layers, so as to detect key objects such as cars, pedestrians and road signs in real time with extremely high accuracy [13]. These algorithms not only improve the accuracy of recognition, but also significantly reduce the false positives and missed detection rates, providing more reliable environmental perception information for autonomous vehicles. At present, there are two kinds of object detection methods based on deep learning: two-level detection and single-level detection. The typical representative of two-level detection technology is R-CNN series [14]. Firstly, it proposes the region of the image, then extracts and classifies the feature of each proposed region, and finally performs boundary box regression on the classification results to obtain the exact target position. After the classification is complete, the post-processing steps are used to eliminate overlapping or redundant bounding boxes to ensure high precision detection results. In addition, a large amount of computing overhead hinders effective deployment on mobile devices with limited processing power. In contrast, single-stage detection technology takes a more direct approach. After the user inputs an image, the system can directly output the detection result, including the category of the target box, the confidence score and the object in the box. The detection and classification process can be completed by only one network, so it has the advantages of high real-time, fast inference, model parameter reduction, and low computational complexity. Although in the initial stage, the accuracy of single-stage detection technology is slightly lower than that of the two-stage method, with the continuous advancement of research, some advanced single-stage algorithms have successfully achieved exceptional accuracy while maintaining high speed and low parameters, making single-stage detection a prominent focus area in contemporary research. In 2015, Redmon J et al. proposed the You Only Look Once (YOLO) algorithm [15] for the first time, which adopted the architecture of full convolutional neural network to transform the object detection problem into a regression problem, and directly predicted the category and position of the object from the entire image. Although YOLO has improved greatly in speed and accuracy, it still performs poorly in small object detection and complex scenes. Afterward, Redmon J [16] et al. proposed YOLOv2 and YOLOv3 on the basis of YOLO, introduced multi-scale prediction and a deeper network structure, improved the detection ability of small objects, while maintaining a high detection speed, and effectively enhanced the detection accuracy of multiple objects in complex environments. Although YOLOv3 improves the detection of small objects, it still has limitations when dealing with extreme scenes and more occluders. Additionally, the inference performance of YOLOv3 is still limited by computing resources, especially in the case of high-resolution image input, which requires a large amount of computational power. Bochkovskiy [17] et al. improved YOLOv3 and proposed YOLOv4 by introducing CSPDarknet53 network, Mish activation function, and data enhancement

strategy, etc., which enabled the model to significantly improve detection accuracy while ensuring high speeds, especially in the application scenarios of highways and urban roads. YOLOv4 has improved both in speed and accuracy, but the processing speed depends on the computing resources, especially in the application of edge devices. YOLOv5 [18] uses a more flexible structure to optimize the speed and deployment efficiency of the model while maintaining high accuracy. It has carried out some targeted optimization for the application scenarios of automatic driving systems, especially in the aspect of real-time object detection. YOLOv7 [19] and YOLOv8 [20] have further improved their backbone networks based on their predecessors by introducing more complex modules and connection methods to enhance feature extraction capabilities. With refined loss functions, they have achieved higher detection performance. YOLOv10 [21] integrates an attention mechanism, which further enhances detection accuracy by highlighting key regions and suppressing background noise. To sum up, single-stage detection technology demonstrates significant promise for autonomous driving applications, enabling real-time environmental analysis for vehicles with precise situational awareness capabilities. In addition, with the continuous advancement of technology and the continuous optimization of algorithms, single-stage detection technology can also be used in more complex scenarios and higher-level autonomous driving systems.

2. Related Work

So far, the YOLO series has been updated in many versions, with YOLOv5 and YOLOv8 receiving much attention. The YOLOv5 model demonstrates exceptional performance in object detection tasks, achieving a balance between detection speed and accuracy. YOLOv5 has a relatively small model size, making it convenient for deployment on mobile devices and embedded systems. However, although YOLOv8 has further improved detection accuracy, its parameter count has also increased. The computational complexity of YOLOv8 creates deployment challenges for resource-limited edge devices. In contrast, the YOLOv5 has generated considerable research interest, with many researchers exploring architectural improvements and optimization strategies tailored for efficient inference in applications. Zhu [22] et al. integrated a feature fusion layer into YOLOv5, along with an attention mechanism, which aims to enhance feature discriminability by focusing on target-related information. However, the weights of channels and spaces need to be calculated separately, significantly increasing the computational load and being unfriendly to the real-time deployment of edge devices. Ning [1] et al. used the deformable convolutional network to optimize the backbone network of YOLOv5, and introduced the DIOU loss function, which effectively improved the detection accuracy and accelerated the convergence speed. Deformable convolution relies on offset learning. If the prediction is inaccurate, it will cause the detection box to shift. Ref. [23] introduced Ghost convolution in the neck C3 module of YOLOv5s, using standard convolution to obtain part of the feature map, and then generating more feature maps through linear operations and connecting them in specific dimensions to extract richer features with a small number of parameters and calculations, thus ensuring detection accuracy and achieving model lightweight. Ghost convolution requires specific hardware support and may not achieve the ideal inference speed on other devices. Ma [24] et al. improved YOLOv8 by replacing the C2f module in the backbone network with a VanillaBlock module, aiming to quantize the model and reduce the parameter count, thereby achieving faster and real-time detection. Liu [25] et al. incorporated the DWR module into YOLOv8 to enhance the multi-scale feature extraction capability of the backbone network, resulting in improved detection accuracy but increased computational load. However, such improvements have led to a significant decrease in detection accuracy. In the field of traffic target recognition, Cao et al. [26] proposed the

MCS-YOLO algorithm, adding the swin-Transformer and coordinate attention module to the backbone network to improve the feature extraction performance of the model. However, the swin-Transformer has a high computational complexity and will affect the real-time detection performance. Lai et al. [27] combined NWD with CIoU to improve the sensitivity to the position deviation of small objects. However, NWD's excessive focus on small targets may lead to insufficient feature extraction of large targets, resulting in a decrease in the accuracy of large targets. Feng et al. [28] proposed a light vehicle detection network based on the improved YOLOv5. By combining the detection head with ASFF, the multi-scale feature fusion has been improved. ASFF will increase the frequency of memory access, resulting in a decrease in the actual computing efficiency. Starting from the redundancy of information, Wang [29] et al. chose the FPGM algorithm to prune YOLOv5 to obtain a balance between recognition accuracy and detection speed. TensorRT inference accelerator further improves performance, achieving respectable frame rates on devices such as the Nvidia Jetson Nano and Xavier NX [30].

The above-mentioned studies have made significant progress in object detection in the field of autonomous driving. However, there is still a certain amount of computational complexity and latency, which can affect the real-time performance of autonomous driving systems. Additionally, autonomous driving systems require high-performance hardware to support the operation and detection of deep learning models [31]. Although the current hardware technology has made great progress, it still may not be able to satisfy the computing, storage and power requirements of autonomous driving systems. Therefore, we have optimized and improved the YOLOv5 series for object detection in autonomous driving, making it more suitable for our application scenarios.

To address the above challenges, this paper proposes an optimized object detection framework based on YOLOv5. The main contributions in this work can be summarized as follows:

1. A 3×3 convolution in the Bottleneck layer of the C3 module in the *backbone* network is grouped. This architectural design reduces model complexity by reducing the parameter count and computation, facilitating the creation of architectures optimized for deployment in resource-limited environments. Processing the input channels of each group independently not only reduces the redundancy within the model, but also promotes the locality and diversity of feature learning.
2. The C3 module in the neck is replaced by a Res2Net module, which uses group convolution and feature reuse to extract features at different scales. The introduction of grouping convolution further reduces the computational burden of the model. By dividing the input feature maps into different groups and applying subsequent independent convolution operations to each subgroup, this method effectively reduces parameter counts and computation power. The feature reuse mechanism enables the network to utilize the feature information extracted from previous layers more effectively, avoiding redundant computation. It not only improves the computational efficiency of the network, but also promotes the deep interaction between features and enhances the expressiveness of the model.
3. The EIOU loss function introduces more parameters to calculate the overlap area, which can measure the overlap degree of the predicted box and the real box more accurately. It not only takes into account the proportion of the intersection area between the predicted box and the real box to the union area, but also takes into account the shape, direction and center point distance and other factors. These factors work together to calculate the loss, which can more fully reflect the spatial relationship between the two boxes. This makes it more sensitive when dealing with small targets, thus improving the accuracy of target detection.

3. Method

3.1. YOLOv5

The algorithm presented in this paper is related to YOLOv5, so in this section we will briefly review YOLOv5. As shown in Figure 1, YOLOv5 architecture is divided into four distinct components. The first part is the input, which performs the necessary pre-processing operations on the input image to prepare for the subsequent stage of the network.

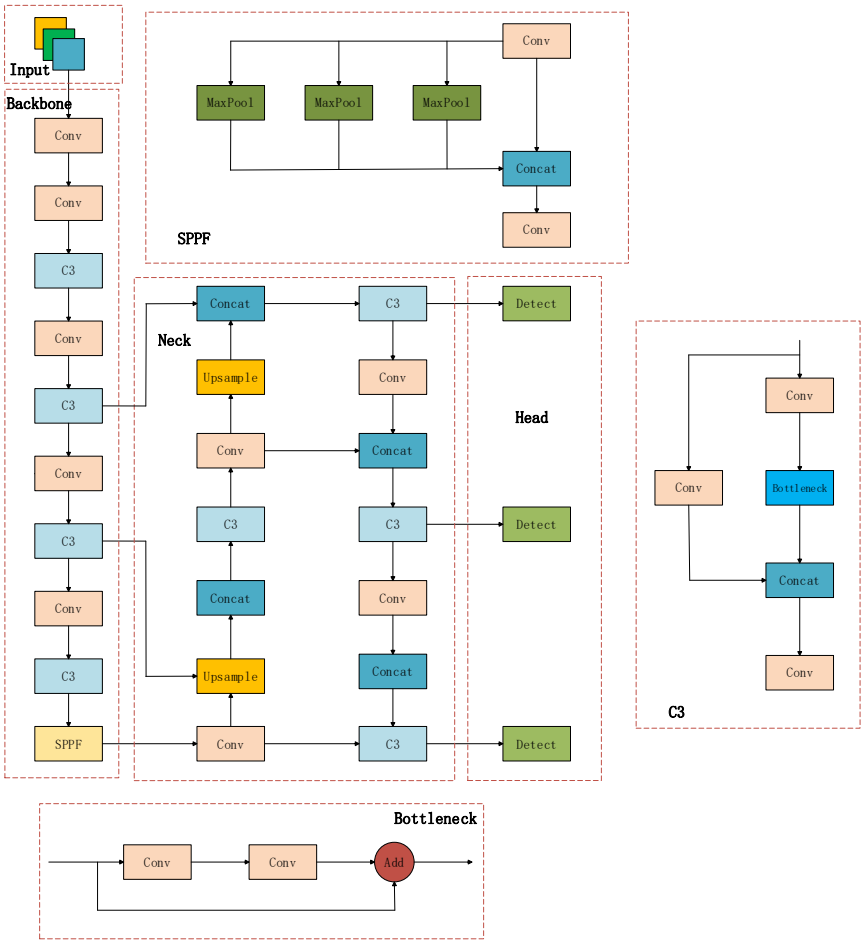


Figure 1. YOLOv5 network structure.

The second part is the *backbone* network, which is mainly composed of *Conv*, C3 and Spatial Pyramid Pooling (SPP) modules [32]. *Conv* is the basic convolution unit of YOLOv5, which performs Conv2d convolution, BatchNormalization layer (BN), and SiLU activation operations on input features. The C3 module consists of three Conv layers and multiple Bottleneck layers. It contains two branches: one for deep feature extraction through multiple Bottleneck layers, and the other through a *Conv*, and then the two branches are connected through residual connection, which effectively increases the depth and receptive field of the network, and improves the capability of feature extraction. SPP uses the maximum pooling method of different sizes, and then concatenates the feature maps of different scales to further enhance the feature extraction capability.

The third part is the *neck* network, which is mainly responsible for feature fusion. Using a FPN+PAN structure, features of different levels are fused to better detect targets of different scales. FPN (Feature Pyramid Network) conveys semantic information from the top down, and PAN (Pyramid Attention Network) [33] conveys positioning features from the bottom up.

The fourth part is the *head*, which is mainly responsible for outputting detection information, including the category, location and confidence of the target.

3.2. TCE-YOLOv5

Although YOLOv5 achieves a good balance between accuracy and parameter count, in order to be better deployed to edge devices, the number of parameters and computational costs of the overall network model must be reduced. The parameter count and computational load of the network model are mainly concentrated in the backbone network. Wang [34] et al. replaced the C3 module in the *backbone* network with S-Ghost. Although the model is lightweight, it has higher computational complexity and puts higher demands on hardware resources. This can be a problem on resource-limited edge devices, which often have limited computing power. Therefore, we proposed T-C3, inspired by Tree block [35]. After replacing C3 in the *backbone* network with T-C3, the parameter count and calculation cost of the model will be greatly reduced, but the ability of feature extraction will be reduced, resulting in a slight decrease in accuracy.

In order to solve the problem of reducing accuracy caused by modifying the backbone network, this paper uses the research of Res2Net for reference and replaces the Bottleneck of the C3 module in the neck network with Res2Net. By introducing more scale variations, the network can capture richer feature information, enhance feature extraction capability, and keep the parameter and computation cost low. At the same time, the loss function CIOU is changed to EIOU, which is improved on the basis of CIOU. EIOU can make the network pay more attention to the shape difference between the predicted box and the real box, and improve the positioning accuracy and the overall detection accuracy by introducing additional shape constraints. The TCE-YOLOv5 is shown in Figure 2.

The detailed descriptions of T-C3 and C3Res2Net are located in Sections 3.3 and 3.4, respectively.

3.3. T-C3

Although the C3 module performs well in feature extraction, it consumes a lot of computing resources due to its large number of network parameters and high computational complexity. In order to solve this problem, this paper, inspired by Tree block, improves the Bottleneck module of the C3 module in the *backbone* network to reduce the calculation overhead without reducing the accuracy as much as possible. Figure 3 is a comparison diagram before and after Bottleneck module modification. Figure 3a is the original Bottleneck, and Figure 3b is the modified one, which is named Tree Bottleneck. The Tree Bottleneck module is designed to balance the accuracy and computational efficiency. First, the H $\times$ W $\times$ C input passes through the first 3 $\times$ 3 convolution but does not directly proceed to the second 3 $\times$ 3 convolution. Instead, it is divided into two groups along the channel dimension, with one group passing through a 3 $\times$ 3 convolution and the other through a 1 $\times$ 1 convolution. The resulting outputs are then concatenated. The benefit of this approach is that only half of the channels undergo a 3 $\times$ 3 convolution, which can reduce the parameter count and computational load. The parameter count is reduced by about 38% and the amount of computation cost is reduced by about 44% for each C3. Meanwhile, the parallel use of convolution kernels of different sizes allows for the capture of multi-scale features. However, applying 1 $\times$ 1 convolution operations to half of the channels can reduce the receptive field, leading to a decrease in detection accuracy for large targets as well as improve the detection accuracy of small targets. This has also been proven in experimental results.

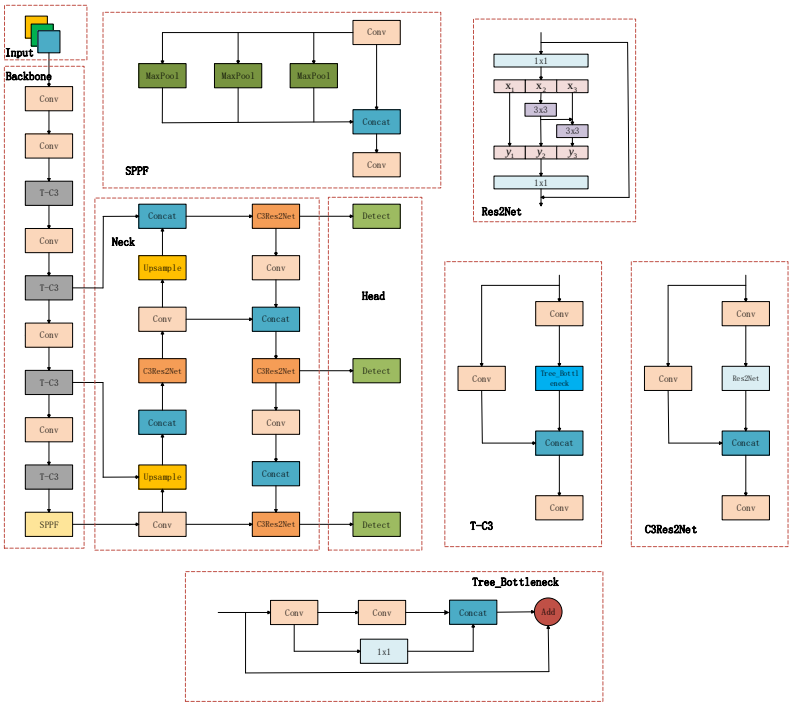
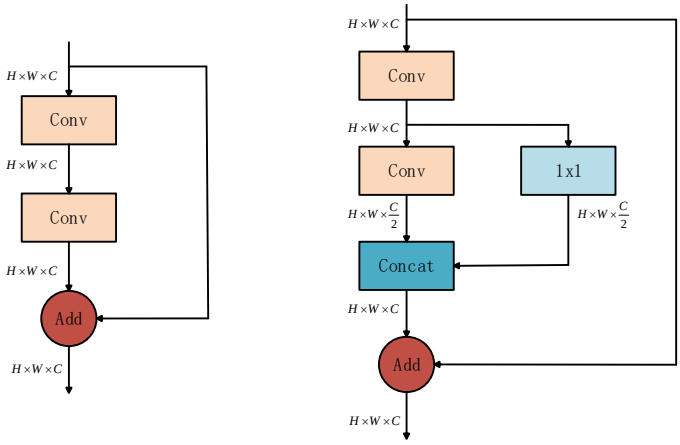


Figure 2. TCE-YOLOv5 network structure.



（a） （b）

Figure 3. Modified Bottleneck structure diagram. (**a**) The original Bottleneck; (**b**) the modified version.

3.4. C3Res2Net

After the *backbone* network has extracted the basic features of an image, the *neck* network should further fuse and enhance these features. In the *neck* module of YOLOv5, the C3 module is also an important part, which fuses feature information of different levels through multiple convolution and residual connection. However, to align with the improvements made to the backbone network and enhance the detection accuracy for small and medium-sized targets, this paper did not adopt a C3 module. Instead, it utilized C3Res2Net, which replaces the Bottleneck in the C3 module of the YOLOv5 neck network with a Res2Net [36] structure. Res2Net is an improved residual network structure, which is characterized by the introduction of more scale variations in each residual block. Traditional residual blocks typically contain only one path, but Res2Net enriches the feature representation by adding multiple branches. These branches have different convolution kernel sizes or channel numbers, allowing richer contextual information to be captured. In this way, Res2Net was able to significantly improve the model's performance while

keeping the parameter count low. In this paper, we replace the Bottleneck in the C3 module of the YOLOv5 *neck* network with a Res2Net structure. Specifically, we replace each Bottleneck unit in the C3 module with a Res2Net block with multiple branches. These branches extract features through different convolution operations and fuse them through residual connection. In this way, we can take advantage of Res2Net's scaling characteristics to enhance the feature extraction capability of the C3 module. The replacement C3 module maintains the original input–output interface, so it can be seamlessly integrated into YOLOv5's *neck* network. The structure of Res2Net is shown in Figure 4. By constructing hierarchical residual connections within a single residual block, this structure improves the multi-scale feature representation capability of the network at a finer level. Res2Net divides channels into S groups convolution operation. The larger the value of S, the stronger the receptive field's ability to learn multi-scale features.

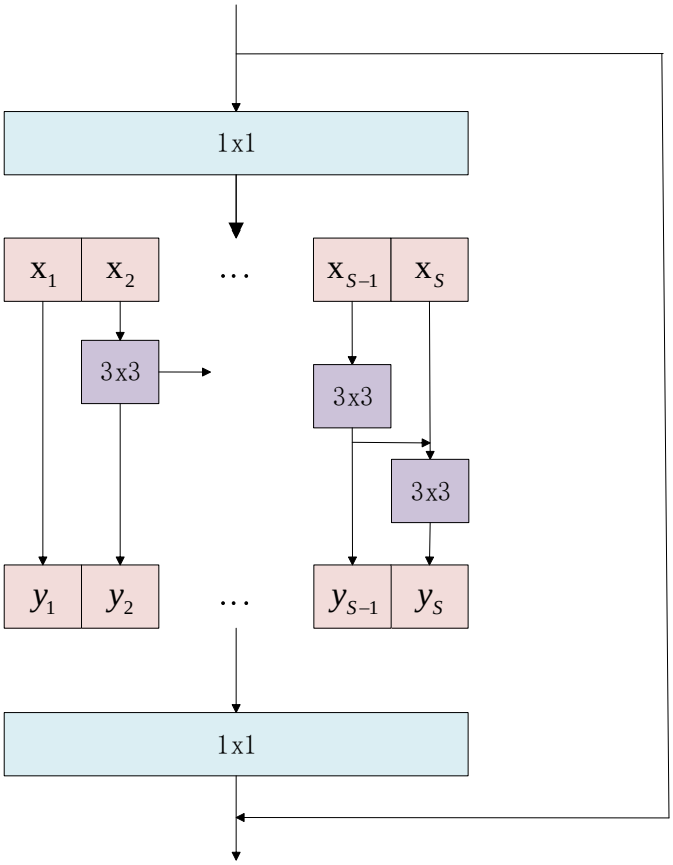


Figure 4. Replacement of the Bottleneck in C3 with Res2Net.

As shown in Figure 4, after the 1×1 convolution layer in the Res2Net module, the feature map is evenly divided into S feature map subsets, represented by x_i, where $i \in 1, 2, \ldots S$. The number of channels for x_i is $1/S$ of the original channels, and the feature map size remains the same as the input. Each x_i, except for x_1, has a corresponding 3×3 convolution that can accept and integrate feature information from all preceding subsets $(x_2, \ldots, x_{S-1})$. This design enables Res2Net to effectively integrate feature information of different scales, thus enhancing the feature extraction capability of the model. The Res2Net structure can be expressed by the following equation:

$$y_i = \begin{cases} x_i & i = 1, \\ K_i(x_i) & i = 2, \\ K_i(x_i + y_{i-1}) & 2 < i <= S, \end{cases} \quad (1)$$

where K_i represents the convolution operation of 3×3, x represents the input feature map, and y represents the output feature map.

In this paper, we set S to 3. First, when S increases, although richer multi-scale features can be captured, the parameter count and computational complexity of the model will increase accordingly. Choosing $S = 3$ can control the complexity and computation of the model to a certain extent, which is especially important for application scenarios where performance is maintained or hardware requirements are reduced. Secondly, selecting $S = 3$ can reduce unnecessary redundancy and improve feature fusion efficiency while ensuring feature diversity. Embed Res2Net into C3, as shown in Figure 5.

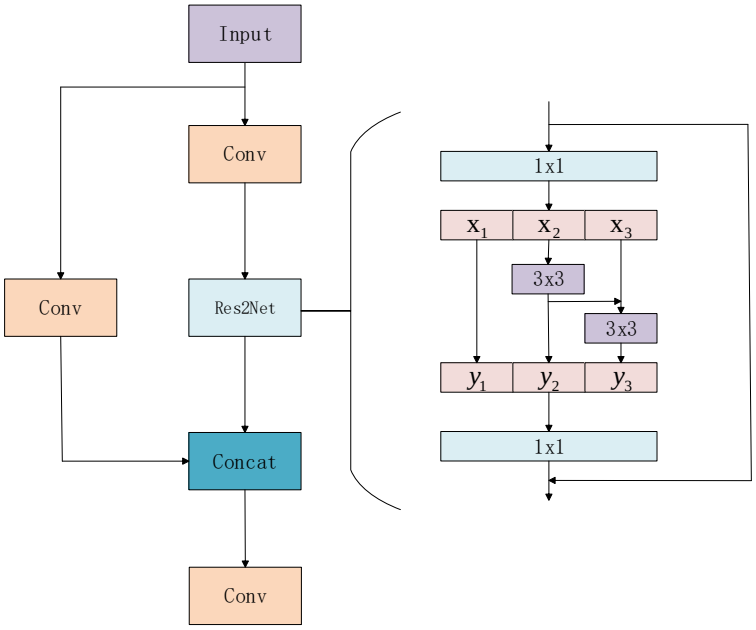


Figure 5. Replace the Bottleneck in C3 with Res2Net.

C3 and C3Res2Net were compared and the results are shown in Table 1. It can be observed that compared with C3, although the detection accuracy of C3Res2Net for large targets is decreased, it is improved for small and medium-sized targets to a certain extent, which is in line with the theory.

Table 1. Detection results of C3 and C3Res2Net for targets of different sizes.

Area	Small (IoU = 0.5:0.95)	Mediume (IoU = 0.5:0.95)	Large (IoU = 0.5:0.95)
C3	0.461	0.616	0.781
C3Res2Net	0.467	0.623	0.751

3.5. Loss Function

The loss function is very important in the training process because it can effectively measure the difference between the predicted results of the model and the real data. Well-designed loss functions play a dual role in improving model performance and training efficiency by accelerating convergence. In YOLOv5, the loss function integrates three key components: confidence loss, classification loss and localization loss.

Localization loss is an important concept used in object detection to measure the degree of overlap between the bounding box and the ground truth, and thus to evaluate the accuracy of object detection, known as the Intersection over Union (IOU). However, the traditional IOU loss function has a significant drawback: when the bounding box does

not intersect with the target frame, the value of IOU is 0, which leads to the loss function failing to provide effective gradient information for the model, thus hindering further optimization of the model. YOLOv5 uses GIOU [37] as a localization loss function to solve the problem that the bounding box and the ground truth do not intersect and the original IOU is not differentiable. GIOU encloses the bounding box and the ground truth using the area of the minimum external rectangle, then calculates the proportion of the minimum external rectangle that is not part of the bounding box and the ground truth, and finally subtracts this proportion from the IOU to obtain the GIOU. The definition of GIOU is shown in the equation:

$$GIOU = IOU - \frac{C - (A - B)}{C}. \tag{2}$$

GIOU is better able to focus on the degree of overlap between the bounding box and the ground truth, but when there is an inclusion relationship between the bounding box and the ground truth, GIOU has difficulty distinguishing the relative position of the two. DIOU [38] solves this problem by directly measuring the Euclidean distance between the predicted box and the ground truth. The definition of DIOU is shown in the equation:

$$DIOU = IOU - \frac{\rho^2(b, b^{gt})}{c^2}, \tag{3}$$

where b represents the center point of the prediction box, and b^{gt} represents the center point of the real box, p represents the Euclidean distance between the two center points, and c represents the diagonal length of the minimum external rectangle.

However, DIOU does not consider the aspect ratio of the bounding box and the ground truth, but the bounding box and the ground truth center point is the same, DIOU effect is not good. Therefore, CIOU [39] increases the aspect ratio on the basis of DIOU, which can effectively solve the problem of DIOU. The definition of CIOU is shown in the Equation (4), where v is the parameter used to compare the consistency of the aspect ratio, a is the weighting factor.

$$CIOU = IOU - \left(\frac{\rho^2(b, b^{gt})}{c^2} + av \right), \tag{4}$$

$$a = \frac{v}{(1 - IOU) + v}, \tag{5}$$

$$v = \frac{4}{\pi^2} \left(arctan \frac{w^{gt}}{h^{gt}} - arctan \frac{w}{h} \right)^2. \tag{6}$$

Although CIOU has increased the aspect ratio, the definition of the aspect ratio is vague. EIOU [38] proposes to disentangle the influence of aspect ratio by independently calculating the length and width of the predicted bounding box and the ground truth. The description of EIOU is shown in Figure 6. The EIOU loss consists of three components: overlap loss, central distance loss, and width–height loss. The first two components follow the methodology established in CIOU, while the width–height loss specifically minimizes the discrepancy between the dimensions of the predicted and ground truth boxes, thereby accelerating the convergence speed of the model training process. EIOU explicitly incorporates the differences in width and height into the loss calculation, enabling the model to more accurately locate and fit small and medium-sized targets, thereby improving detection accuracy. In contrast, although CIOU also considers the aspect ratio, its handling may not be as direct in optimizing width and height as EIOU, leading to slightly inferior performance on small and medium-sized targets. Although the design of

the EIOU loss function increases the complexity of training, potentially resulting in longer training times, this does not affect the model's inference efficiency. During the inference stage, the model has already completed its learning and no longer needs to calculate the loss function. Therefore, the improvements introduced by EIOU do not increase the latency in practical applications. This means that EIOU enhances detection performance while maintaining good real-time performance, making it particularly suitable for scenarios with high requirements for both accuracy and speed. The definition of EIOU is shown in the equation:

$$L_{EIOU} = 1 - IOU + \frac{\rho^2(b, b^{gt})}{(c_w)^2 + (c_h)^2} + \frac{\rho^2(w, w^{gt})}{(c_w)^2} + \frac{\rho^2(h, h^{gt})}{(c_h)^2}, \quad (7)$$

where c_w and c_h are the width and height of the minimum external rectangle.

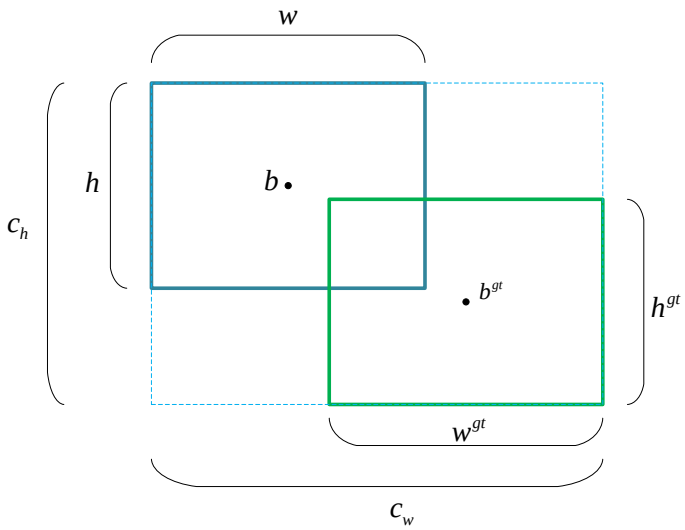


Figure 6. Description of EIOU.

Therefore, we use EIOU loss function as the loss function of the algorithm, which can maximize the detection accuracy between the predicted box and the ground truth.

4. Experiment and Result Analysis

4.1. Experiment Settings

In this paper, YOLOv5s-6.0 is used as the baseline for network training using Adam optimizer. The batch size is set to 32, and the number of epochs is 150. OneCycle learning rate scheduling strategy, 0.001 initial learning rate, 0.01 learning rate factor and 0.0005 weight decay factor are adopted. The momentum of the first three pre-training stages is 0.8, and then it becomes 0.937. In addition, the input image size is uniformly 640 × 640. The experiment was designed on a PC platform with a Xeon (R) W-2223 CPU@3.60 GHz, an NVIDIA GeForce RTX3080 GPU, running on the Ubuntu 18.04 operating system and developed using the PyTorch 1.13 framework. The edge platform is Jetson Xavier NX, as shown in Figure 7, featuring a six-core NVIDIA Carmel Arm® v8.2 64-bit CPU, 384 Core NVIDIA Volta™ architecture GPU with 48 Tensor Core, and it runs on Ubuntu 18.04. All models were trained on a same PC. Similarly, when deployed on the Jetson Xavier NX, all models were quantized by using the TensorRT method.



Figure 7. Jetson Xavier NX.

4.2. Datasets

In this paper, two public traffic datasets are selected to prove the effectiveness of the method. (1) KITTI dataset [40]: The dataset was developed by the Karlsruhe Institute of Technology (KIT) in Germany and the Toyota Technological Institute at Chicago, USA. TTI-C is one of the world's largest computer vision algorithm evaluation datasets for automated driving scenarios. The KITTI dataset contains real-world image data from urban, rural, and highway scenarios. Each image can contain up to 15 vehicles and 30 pedestrians, with varying degrees of occlusion and truncation. (2) CCTSDB2021 dataset [41]: China Traffic Sign Detection Benchmark 2021 is a traffic sign detection dataset produced by a team of scholars from Changsha University of Science and Technology. The dataset is broken down by traffic sign category, size and weather conditions. The traffic sign category includes ban signs, warning signs and so on. The dataset was divided into a training set, test set and verification set, with 12,499, 3572 and 1785 samples, respectively. These images are derived from real road driving scenes in China and have rich road background information.

4.3. Evaluation Metrics

This article uses mean Average Precision (mAP), number of parameters, FLOPs, P (Precise), R (Recall) and frames per second (FPS) as metrics to evaluate the model. Accuracy reflects the proportion of positive samples predicted by the model that are actually positive samples, and is an important indicator for evaluating the accuracy of model predictions. The recall rate measures the proportion of positive samples that the model can correctly identify to all actual positive samples, reflecting the model's ability to recognize positive samples. The formulas for calculating the accuracy and recall rate are given by Equation (8) and Equation (9), respectively. In the formula, TP is the true case (positive samples predicted as positive by the model), and FP is the false positive case (negative samples predicted as positive by the model), and FN is a false negative example (positive samples predicted as negative by the model). AP measures the performance of the model on a specific class goal and is specific to a specific class, as shown in Equation (10). Mean Average Precision (mAP)

is defined as the average accuracy at different recall rates, as shown in Equation (11). FPS is the number of images that can be processed per second.

$$P = \frac{TP}{TP + FP}, \tag{8}$$

$$R = \frac{TP}{TP + FN}, \tag{9}$$

$$AP = \int_0^1 P(r)dx, \tag{10}$$

$$mAP = \frac{1}{N}\sum_{j=1}^{N} AP_j. \tag{11}$$

4.4. Result Analysis

Table 2 shows the experimental results of different algorithm models in the KITTI dataset. In other words, the TCE-YOLOv5 algorithm is compared with YOLOv5, YOLOv7-tiny [19], YOLOv8 [42] and Improved YOLOv5 [43]. As can be seen from Table 3, the above five algorithms all show good performance. Among them, TCE-YOLOv5 is the algorithm with the fewest parameters and calculation. Compared with the benchmark model YOLOv5, the number of parameters is reduced by 18.5%, the calculation is reduced by 21%, and mAP@50 is increased by 1.0%. Compared with other algorithm models, the indicators of YOLOv7-tiny and Improved YOLOv5 are weaker than those of TCE-YOLOv5. Although YOLOv8 is higher than TCE-YOLOv5 in terms of accuracy, the parameter count and calculation amount of YOLOv8 are much higher than that of TCE-YOLOv5, so YOLOv8 is not suitable for deployment on edge devices. Overall, TCE-YOLOv5 is the most balanced and suitable for deployment in edge devices.

Table 2. Performance of different models of the KITTI dataset on PC.

Models	Params (M)	Flops (G)	mAP@50 (%)	mAP@50:95 (%)	P (%)	R (%)
YOLOv5	7	16	93.5	68. 8	94.4	87.1
TCE-YOLOv5	5.7	12.6	94.5	70.8	94.1	88.8
YOLOv7-tiny	6	13.2	92.8	64.5	92.2	86.2
Improved-YOLOv5 [43]	5.8	13.6	91.6	66.5	90.5	86.9
YOLOv8	11.1	28.7	95.2	78.0	95.0	90.8

To further demonstrate the superiority of TCE-YOLOv5 in autonomous driving detection, we performed an experimental evaluation on the CCTSDB2021 dataset. The CCTSDB2021 dataset provides traffic sign classification. Compared with the KITTI dataset, the CCTSDB2021 dataset has more small targets and is more challenging. Table 3 shows the detection capabilities of proposed method and other methods. It can be seen that the mAP@50 of TCE-YOLOv5 is still higher than that of YOLOv7-tiny and Improved YOLOv5, but it is almost the same as that of the benchmark model YOLOv5 and much lower than that of YOLOv8. However, considering the parameter count and the amount of calculations, TCE-YOLOv5 is still the best choice for edge devices.

Table 3. Performance of different models of the CCTSDB2021 dataset on PC.

Models	Params (M)	Flops (G)	mAP@50 (%)	mAP@50:95 (%)	P (%)	R (%)
YOLOv5	7	16	93.5	66.3	93.9	87.8
TCE-YOLOv5	5.7	12.6	93.4	66.5	93.2	87.9
YOLOv7-tiny	6	13.2	89.5	60.0	90.3	83.0
Improved-YOLOv5 [43]	5.8	13.7	92.0	66.1	92.7	86.3
YOLOv8	11.1	28.7	97.2	75.6	96.6	92.9

Considering that TCE-YOLOv5 is designed for embedded device deployment, we chose Jetson Xavier NX, quantized the model FP32 and FP16 using TensorRT, and performed comparative experiments. FP32, a 32-bit single-precision floating point number, is one of the most commonly used numerical representations in deep learning. FP16 is a 16-bit half-precision floating point number. Compared with FP32, it occupies half of the memory and bandwidth, and is often used in inference processes to speed up calculations and reduce memory usage. Using FP16 can significantly reduce the time to model inference, while in most cases having less impact on model accuracy. In the same test environment, we used the same dataset for evaluation and compared it with the benchmark model YOLOv5 and Improved YOLOv5. The experimental results of FP32 quantization are shown in Table 4.

Table 4. Experimental results of edge device model FP32 quantization.

Models (FP32)	P (%)	R (%)	mAP@50 (%)	mAP@50:95 (%)	FPS
YOLOv5	94.5	87.5	93.6	69.0	30
Improved-YOLOv5 [43]	90.1	87.0	92.0	66.5	20
TCE-YOLOv5	93.6	89.0	94.5	70.8	32

As can be seen from Table 4, TCE-YOLOv5 shows excellent performance in multiple performance indicators. Except having a p value slightly lower than YOLOv5, the other indicators are better than YOLOv5 and Improved YOLOv5. Especially in terms of FPS, TCE-YOLOv5 reached 32 frames, which is only 2 frames higher than YOLOv5, but it also reflects that the network structure of TCE-YOLOv5 is better than YOLOv5, and can provide faster response when processing images or videos. The experimental results of FP16 quantization are shown in Table 5.

Table 5. Experimental results of edge device model FP16 quantization.

Models (FP16)	P (%)	R (%)	mAP@50 (%)	mAP@50:95 (%)	FPS
YOLOv5	94.4	87.6	93.6	69.0	53
Improved-YOLOv5 [43]	89.9	87.2	92.0	66.4	51
TCE-YOLOv5	93.9	88.8	94.6	70.5	61

As can be seen from Table 5, after FP16 quantization, the FPS of TCE-YOLO is significantly Improved, which is 8 frames higher than the benchmark model YOLOv5, and 10 frames higher than the Improved YOLOv5. Meanwhile, other indicators have no obvious changes. It can be seen that the TCE-YOLOv5 after FP16 quantization is more suitable for deployment in edge devices.

Based on the analysis of Tables 4 and 5, it can be concluded that deploying models on resource-limited platforms requires simplified network architectures. Such a simplified network can reduce operator conflicts during TensorRT optimization. Compared with

YOLOv5, the improved design of TCE-YOLOv5 demonstrates better adaptability for edge deployment, with enhanced performance across all metrics.

Figure 8 shows the results of the different algorithms tested on the KITTI dataset. Specifically, Figure 8a shows the detection results of YOLOv5, Figure 8b shows the Improved detection effect of YOLOv5, and Figure 8c shows the results achieved by the proposed algorithm in this paper. Through comparative analysis, we can clearly observe that no matter YOLOv5 or Improved YOLOv5, there are cases of missing or wrong detection. In contrast, our algorithm demonstrated greater accuracy in detecting vehicles. Meanwhile, it can be seen that the detection accuracy of TCE-YOLOv5 is lower than that of YOLOv5 and Improved YOLOv5 for large targets, but the detection accuracy of TCE-YOLOv5 is higher than that of YOLOv5 and Improved YOLOv5 for medium targets and small targets. This is consistent with the previous theoretical analysis and the conclusions in Table 1.

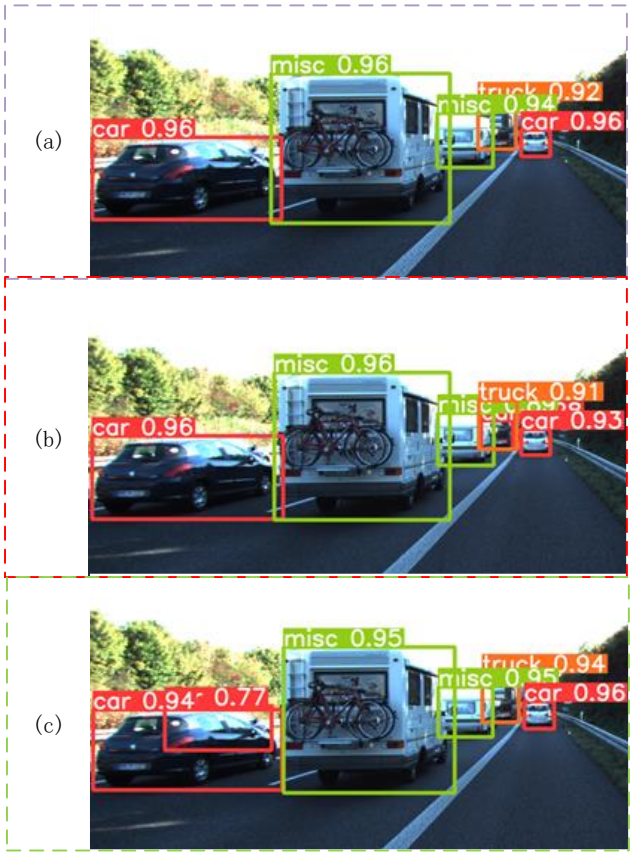


Figure 8. Actual detection results, (**a**) YOLOv5; (**b**) Improved YOLOv5; (**c**) TCE-YOLOv5. The dataset is KITTI.

In order to further explore the specific contribution of each module to the performance of the basic model, seven ablation experiments were carefully planned in this paper, all of which were performed under KITTI dataset and experimental conditions. The results are detailed in Table 6. The analysis results show that each improvement measure has a positive impact on the model. In particular, the introduction of the T-C3 module significantly reduced the number of parameters in the model, while its negative impact on accuracy was minimal. In addition, C3Res2Net not only further reduces the parameter size, but also successfully improves the performance mAP@0.5 by 0.9 percentage points, improving the detection performance. In addition, C3Res2Net not only further reduces the parameter size, but also improves the detection accuracy for targets of different sizes, successfully enhancing the performance of mAP@50 Improved by 0.9 percentage points, improving

detection performance. The addition of EIOU also played a positive role in the optimization of model performance, helping mAP@50 increase by 0.5%. These ablation experiments strongly verify the rationality and effectiveness of the proposed improvement measures.

Table 6. Ablation experiments of the KITTI dataset on PC.

T-C3	C3Res2Net	EIOU	P (%)	R (%)	mAP@50 (%)	mAP@50:95 (%)	Params (M)
			94.4	87.1	93.5	68.8	7
✓			93.6	87.0	93.5	68.4	6.2
	✓		94.2	90.1	94.4	71.9	6.5
		✓	94.1	87.6	94.0	70.0	7
✓	✓		94.4	88.8	94.2	70.3	5.7
✓		✓	94.3	88.6	94.2	70.4	6.2
✓	✓	✓	94.1	88.8	94.5	70.8	5.7

5. Conclusions

Object detection is an important component of autonomous driving, The deployment of object detection models based on deep neural networks must consider the limited resources of the vehicle platform. Prior methods have either focused on improving detection performance and could not be deployed on vehicle platforms due to the large network, or its deployment performance on vehicle platforms was too low. In this paper, a light-weight object detection model named TCE-YOLOv5 is introduced, specially designed for minimizing the number of model parameters and computational complexity while ensuring the accuracy of object detection. By replacing C3 in the backbone network with T-C3 and C3 in the neck with Res2Net, the number of parameters and computational complexity are greatly reduced. In order to balance the problem of decreased accuracy in detecting large targets caused by changes in the backbone network and neck, the EIOU loss function is introduced. The experimental results show that TCE-YOLOv5 is slightly lower than the original YOLOv5 in precision, but all other performance indicators are higher than the original YOLOv5 when performing on the CCTDSB2021 and KITTI datasets. Especially, as a key performance indicator, the mAP@50:90 of TCE-YOLOv5 is always greater than that of the original YOLOv5 in all tests. This is because our proposed modification aims to improve the detection accuracy of small and medium size objects by sacrificing the detection accuracy of large size objects, as small and medium targets detection are more difficult and important for autonomous driving. Due to the reduction in parameter count and computational complexity, the TCE-YOLOv5 runs at a higher FPS (61 frame/s—a 15% increase compared to the original YOLOv5) on the Jetson Xavier NX hardware platform. These results fully prove the superiority of TCE-YOLOv5 in the field of lightweight object detection, and provide a strong guarantee for the real-time and high efficiency of automatic driving systems. In future work, we plan to further explore the possibilities of improving its accuracy. Additionally, we will leverage the resources of State Key Laboratory of Intelligent Vehicle Safety Technology to increase the number of real-world testing scenarios. This will help us gain a more comprehensive understanding of the model's performance in practical applications and provide valuable feedback for further improvements.

Author Contributions: H.W. (Han Wang) put forward the initial conception of this study and led the design of the research plan. Z.Y. undertook most of the data collection work in this study. Q.L. recorded the experimental data and phenomena, and conducted in-depth analysis and processing of the experimental results. Q.Z. and H.W. (Honggang Wang) gave full play to their professional advantages in research methods and optimized and improved the research methods. All authors have read and agreed to the published version of the manuscript.

Funding: This work was funded by the State Key Laboratory of Intelligent Vehicle Safety Technology Open Project (IVSTSKL-202328).

Data Availability Statement: No new data were created in this study.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

List of abbreviations.

Name	Representation	Name	Representation
x	Input feature map	α	Weighting factor
y	Output feature map	v	Aspect ratio parameter
K_i	i-layer 3 × 3 Convolution operation	w^{gt}	Width of the real box
ρ	Euclidean distance	h^{gt}	Height of the real box
b	Center point of the prediction box	w	Width of the prediction box
b^{gt}	Center point of the real box	h	Height of the prediction box
[c]c	Diagonal length of the minimum external rectangle.	c_w	Width of the minimum external rectangle
c_h	Height of the minimum external rectangle		

References

1. Ning, J.; Wang, J. Automatic Driving Scene Target Detection Algorithm Based on Improved YOLOv5 Network. In Proceedings of the 2022 International Conference on Computer Network, Electronic and Automation (ICCNEA), Xi'an, China, 23–25 September 2022; pp. 218–222.
2. Shirmohammadi, S.; Ferrero, A. Camera as the instrument: The rising trend of vision based measurement. *IEEE Instrum. Meas. Mag.* **2014**, *17*, 41–47. [CrossRef]
3. Paden, B.; Čáp, M.; Yong, S.Z.; Yershov, D.; Frazzoli, E. A Survey of Motion Planning and Control Techniques for Self-Driving Urban Vehicles. *IEEE Trans. Intell. Veh.* **2016**, *1*, 33–55. [CrossRef]
4. Cao, Z.; Xu, L.; Niu, Z.; Zhang, C.; You, G.; Zhao, M.; Yang, Y. YOLOv7-Based Autonomous Driving Object Detection Algorithm. In Proceedings of the 2024 9th International Conference on Computer and Communication Systems (ICCCS), Xi'an, China, 19–22 April 2024; pp. 172–177.
5. Fu, Y.; Li, C.; Yu, F.R.; Luan, T.H.; Zhang, Y. A Survey of Driving Safety with Sensing, Vehicular Communications, and Artificial Intelligence-Based Collision Avoidance. *IEEE Trans. Intell. Transp. Syst.* **2022**, *23*, 6142–6163. [CrossRef]
6. Mahadevkar, S.V.; Khemani, B.; Patil, S.; Kotecha, K.; Vora, D.R.; Abraham, A.; Gabralla, L.A. A Review on Machine Learning Styles in Computer Vision—Techniques and Future Directions. *IEEE Access* **2022**, *10*, 107293–107329. [CrossRef]
7. Lv, Z.; Song, H. Mobile Internet of Things Under Data Physical Fusion Technology. *IEEE Internet Things J.* **2020**, *7*, 4616–4624. [CrossRef]
8. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
9. Shu, H.; Liu, T.; Mu, X.; Cao, D. Driving Tasks Transfer Using Deep Reinforcement Learning for Decision-Making of Autonomous Vehicles in Unsignalized Intersection. *IEEE Trans. Veh. Technol.* **2022**, *71*, 41–52. [CrossRef]
10. Ren, H.; Jing, F.; Li, S. DCW-YOLO: Road Object Detection Algorithms for Autonomous Driving. *IEEE Access* **2024**, *early access*. [CrossRef]
11. David, G. Object Recognition from Local Scale-invariant Features. In Proceedings of the International Conference on Computer Vision, Kerkyra, Greece, 20–27 September 1999; pp. 1150–1157.
12. Surasak, T.; Takahiro, I.; Cheng, C.H.; Wang, C.E.; Sheng, P.Y. His togram of Oriented Gradients for Human Detection in Video. In Proceedings of the International Conference on Business and Industrial Research (ICBIR), Bangkok, Thailand, 17–18 May 2018; pp. 172–176.
13. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. ImageNet classification with deep convolutional neural networks. *Adv. Neural Inf. Process. Syst.* **2012**, *25*, 1097–1105. [CrossRef]
14. Girshick, R.; Donahue, J.; Darrell, T.; Malik, J. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, 23–28 June 2014; pp. 580–587.
15. Redmon, J.; Redmon; Divvala, S.; Girshick, R.; Farhadi, A. You Only Look Once: Unified, Real-Time Object Detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 779–788.

16. Redmon, J.; Farhadi, A. Yolov3: An incremental improvement. *arXiv* **2018**, arXiv:1804.02767.
17. Bochkovskiy, A.; Wang, C.Y.; Liao, H.Y.M. Yolov4: Optimal Speed and Accuracy of Object Detection. *arXiv* **2020**, arXiv:2004.10934.
18. Zhu, X.; Lyu, S.; Wang, X.; Zhao, Q. TPH-YOLOv5: Improved YOLOv5 based on transformer prediction head for object detection on drone-captured scenarios. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Montreal, BC, Canada, 11–17 October 2021; pp. 2778–2788.
19. Wang, C.Y.; Bochkovskiy, A.; Liao, H.Y.M. YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Vancouver, BC, Canada, 17–24 June 2023; pp. 7464–7475.
20. Liu, Y.; Shen, S. Vehicle Detection and Tracking Based on Improved YOLOv8. *IEEE Access* **2025**, *13*, 24793–24803. [CrossRef]
21. Wang, A.; Chen, H.; Liu, L.; Chen, K.; Lin, Z.; Han, J. Yolov10: Real-time end-to-end object detection. *Adv. Neural Inf. Process. Syst.* **2024**, *37*, 107984–108011.
22. Zhu, L.; Geng, X.; Li, Z.; Liu, C. Improving YOLOv5 with attention mechanism for detecting boulders from planetary images. *Remote Sens.* **2021**, *13*, 3776. [CrossRef]
23. Dong, X.; Yan, S.; Duan, C. A lightweight vehicles detection network model based on YOLOv5. *Eng. Appl. Artif. Intell.* **2022**, *113*, 104914. [CrossRef]
24. Liu, X.; Wang, Y.; Yu, D.; Yuan, Z. YOLOv8-FDD: A Real-Time Vehicle Detection Method Based on Improved YOLOv8. *IEEE Access* **2024**, *12*, 136280–136296. [CrossRef]
25. Wang, H.; Ma, Z. Small Target Detection Algorithm of Edge Scene Based on Improved YOLOv8. In Proceedings of the 2024 13th International Conference of Information and Communication Technology (ICTech), Xiamen, China, 12–14 April 2024; pp. 124–128
26. Cao, Y.; Li, C.; Peng, Y.; Ru, H. MCS-YOLO: A multiscale object detection method for autonomous driving road environment recognition. *IEEE Access* **2023**, *11*, 22342–22354. [CrossRef]
27. Feng, J.; Yi, C. Lightweight detection network for arbitrary-oriented vehicles in UAV imagery via global attentive relation and multi-path fusion. *Drones* **2022**, *6*, 108. [CrossRef]
28. Lai, H.; Chen, L.; Liu, W.; Yan, Z.; Ye, S. STC-YOLO: Small object detection network for traffic signs in complex environments. *Sensors* **2023**, *23*, 5307. [CrossRef] [PubMed]
29. Wang, X.; Zhang, J.; Wang, Y.; Li, M.; Liu, D. Defect Detection of Track Fasteners Based on Pruned YOLO V5 Model. In Proceedings of the 2022 IEEE 11th Data Driven Control and Learning Systems Conference (DDCLS), Chengdu, China, 3–5 August 2022.
30. Farooq, M.A.; Shariff, W.; Corcoran, P. Evaluation of Thermal Imaging on Embedded GPU Platforms for Application in Vehicular Assistance Systems. *IEEE Trans. Intell. Veh.* **2023**, *8*, 1130–1144. [CrossRef]
31. Xia, W.; Li, P.; Huang, H.; Li, Q.; Yang, T.; Li, Z. TTD-YOLO: A Real-Time Traffic Target Detection Algorithm Based on YOLOV5. *IEEE Access* **2024**, *12*, 66419–66431. [CrossRef]
32. He, K.; Zhang, X.; Ren, S.; Sun, J. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* **2015**, *37*, 1904–1916. [CrossRef] [PubMed]
33. Liu, S.; Qi, L.; Qin, H.; Shi, J.; Jia, J. Path aggregation network for instance segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 8759–8768.
34. Wang, Y.; Jin, S.; Yin, C.; Wu, Z. Target Detection Algorithm Based on Improved YOLOv5 for Edge Computing Devices. In Proceedings of the 2023 5th International Conference on Robotics, Intelligent Control and Artificial Intelligence (RICAI), Hangzhou, China, 1–3 December 2023; pp. 965–970.
35. Rao, L. TreeNet: A lightweight one-shot aggregation convolutional network. *arXiv* **2021**, arXiv:2109.12342.
36. Gao, S.-H.; Cheng, M.-M.; Zhao, K.; Zhang, X.-Y.; Yang, M.-H.; Torr, P. Res2Net: A new multi-scale backbone architecture. *IEEE Trans. Pattern Anal. Mach. Intell.* **2021**, *43*, 652–662. [CrossRef]
37. Rezatofighi, H.; Tsoi, N.; Gwak, J.; Sadeghian, A.; Reid, I.; Savarese, S. Generalized intersection over union: A metric and a loss for bounding box regression. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 15–20 June 2019; pp. 658–666.
38. Zhang, Y.-F.; Ren, W.; Zhang, Z.; Jia, Z.; Wang, L.; Tan, T. Focal and efficient IOU loss for accurate bounding box regression. *Neurocomputing* **2022**, *506*, 146–157. [CrossRef]
39. Zheng, Z.; Wang, P.; Liu, W.; Li, J.; Ye, R.; Ren, D. Distance-IoU loss: Faster and better learning for bounding box regression. *Proc. AAAI Conf. Artif. Intell.* **2020**, *34*, 12993–13000. [CrossRef]
40. Tao, J.; Wang, H.; Zhang, X.; Li, X.; Yang, H. An object detection system based on YOLO in traffic scene. In Proceedings of the 2017 6th International Conference on Computer Science and Network Technology (ICCSNT), Dalian, China, 21–22 October 2017; pp. 315–319.
41. Zhang, J.; Zou, X.; Kuang, L.D. CCTSDB 2021: A more comprehensive traffic sign detection benchmark. In *Human-Centric Computing and Information Sciences*; Springer: Berlin, Germany, 2022; Volume 12.

42. Wang, X.; Gao, H.; Jia, Z.; Li, Z. BL-YOLOv8: An improved road defect detection model based on YOLOv8. *Sensors* **2023**, *23*, 8361. [CrossRef]
43. Yang, J.; Sun, T.; Zhu, W.; Li, Z. A Lightweight Traffic Sign Recognition Model Based on Improved YOLOv5. *IEEE Access* **2023**, *11*, 115998–116010. [CrossRef]

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.