Lightweight Vehicle Detection Algorithm Based on Improved YOLOv4

D. L. Yuan, Y. Xu

Abstract—**Vehicle detection is the first step and an important part of automatic traffic incident detection systems. It guarantees subsequent vehicle identification and vehicle counting accuracy and has crucial theoretical significance and practical value for traffic safety and control. The model obtained by the original YOLOv4 algorithm is too large to be used in embedded terminals in real time. To overcome this problem, this study replaces the original backbone network of YOLOv4, which is CSPDarknet53, with MobileNetv3 for the feature extraction. To further reduce the number of parameters, deep separable convolution is used to replace the common 3×3 convolution in the original model of the enhanced feature extraction networks SPP and PANet. Because of the imbalance in the object detection data, the loss function is redesigned using a weighting method. The research results show that in comparison to the original YOLO series algorithm, the optimized YOLOv4 algorithm improves the accuracy by 0.53% and reduces the number of model parameters by 78%. In comparison to the other algorithms, the improved YOLOv4 model is smaller and more accurate, which is the basis for realizing intelligent transportation systems.**

Index Terms—**KITTI, MobileNetv3, Object detection, YOLOv4.**

I. Introduction

OBJECT detection refers to the identification of all the regions of interest in an image and determining their positions and categories [1]. Recently, deep learning technology in combination with big data and efficient graphics processing unit (GPU) computing has surpassed and replaced traditional algorithms in the field of artificial intelligence [2]. Presently, the object detection methods that are based on deep learning algorithms can be divided into two categories according to whether the region proposal network is adopted. The first is the two-stage object detection algorithm: in the first stage, candidate regions are generated through the candidate region network, and in the second stage, candidate regions are classified and regressed. Representative algorithms include the region-based convolutional neural network (RCNN) [3], fast RCNN [4], faster RCNN [5], and feature pyramid network (FPN) [6]. The second category is the one-stage object detection algorithm, which classifies and regresses the target directly through a neural network. Representative algorithms include the You Only Look Once (YOLO) [7] series, single-shot detector (SSD) [8], and RetinaNet [9]. Real-time detection of vehicles on the road using object-detection techniques can be useful for criminal investigations and provide a reliable basis for traffic regulation and traffic light scheduling.

Among the many detection algorithms, the YOLO series may be the most popular object detection algorithm for practical applications. It is easy to improve a YOLO-based network to achieve the desired results. For example, Yang et al. studied traffic sign detection based on the YOLO lightweight network [10] using deep separable convolution in the backbone network to better extract small- and medium-sized targets. Hui et al. used the image pyramid structure to construct a helmet dataset to obtain a model with more industrial applications for detecting the wearing of safety helmets based on the YOLO lightweight network [11]. These improved algorithms are significant in the field of object detection, but they still do not explain the problem of feature extraction and utilization. YOLOv4 [12] is an open-source object detection network that has obvious advantages over other object detection networks that emerged during the same period in terms of speed and accuracy. YOLOv4 uses CSPDarknet53 as the backbone network and PANet instead of FPN for the feature aggregation. The detection accuracy is high, but it requires a high-performance hardware configuration; the detection speed is slow on small hardware platforms.

Therefore, YOLOv4-Tiny [12] is widely used for detection on embedded platforms. Although the detection speed is high, the detection performance is lower than that of YOLOv4 because of the simple network hierarchy and an insufficient feature extraction ability. Because YOLOv4 is a large model and is not suitable for resource-constrained hardware platforms, this study proposes an improved YOLOv4 lightweight network detection algorithm that is based on YOLOv4. The lightweight backbone network MobileNetv3 [13] is used to replace the original CSPDarknet53 backbone network of YOLOv4 for the feature extraction. To further reduce the number of parameters by strengthening the SPP and PANet feature extraction networks of YOLOv4, deep separable convolution is used to replace an ordinary 3×3 convolution in the original model. This is considered to be an unbalanced problem in the target detection data. Thus, the weighted method is used to redesign the loss function and the optimized lightweight vehicle detection model is employed. The results show that the improved YOLOv4 algorithm has the advantages of small models, high precision, fast speed, and better suitability for small hardware platforms.

II. YOLOv4 Algorithm Principle

The YOLOv4 algorithm is based on the original YOLO object detection framework. This was adopted in recent years as the best optimization strategy for CNNs. All these aspects have different degrees of optimization, but even though there

Manuscript received May 29, 2021; revised October 18, 2021.This work was supported by the National Natural Science Foundation of China under Grants 61575090 and 61775169, and the Scientific Study Project for Education Department of Liaoning Province, China under Grant LJKZ0310.

D. L. Yuan is a Master's Student of School of Computer Science and Software Engineering, University of Science and Technology LiaoNing, Anshan 114051, China (e-mail: ydl608@163.com).

Y. Xu is a Professor of School of Computer Science and Software Engineering, University of Science and Technology LiaoNing, Anshan 114051, China (corresponding author, phone: 86-13889785726; e-mail: xuyang_1981@aliyun.com).

was no theoretical innovation, it was still adopted by many engineers. In particular, a variety of optimization algorithms have been attempted [14]. The algorithm adopted by YOLOv4 retains the head part of YOLOv3 and modifies the backbone network to CSPDarkNet53, as shown in Fig. 1. At the same time, YOLOv4 adopts the idea of spatial pyramid pooling (SPP) to expand the field of experience, and PANet is used as the neck.

	Type	Filters	Size		Output	
	Convolutional	32	3	3	256	256
	Convolutional	64	3	3/2	128	128
1	Convolutional	32	1	1		
	Convolutional	64	3	3		
	Residual				128	128
	Convolutional	128	3	3/2	64	64
2	Convolutional	64	1	1		
	Convolutional	128	3	3		
	Residual				64	64
	Convolutional	256	3	3/2	32	32
8	Convolutional	128	1	1		
	Convolutional	256	3	3		
	Residual				32	32
	Convolutional	512	3	3/2	16	16
8	Convolutional	256	1	1		
	Convolutional	512	3	3		
	Residual				16	16
	Convolutional	1024	3	3/2	8	8
4	Convolutional	512	1	1		
	Convolutional	1024	3	3		
	Residual				8	8
	Avgpool		Global			
	Connected		1000			

Fig. 1. CSPDarkNet53 network structure.

Although the YOLOv4 and YOLOv4-tiny algorithms have advantages in terms of their accuracy and speed in comparison to YOLOv3 and other methods, they still have shortcomings. The trained YOLOv4 model is large and not suitable for embedded applications, whereas the YOLOv4-tiny model is small and fast [15]. However, it does not satisfy the industrial requirements for detection accuracy. Therefore, the aim of this study is to use the optimization method to greatly reduce the number of parameters while maintaining the original YOLOv4 accuracy and improving the speed of the model [16].

III. Improvement Strategy

Model optimization can be performed from the aspects of backbone network, optimizer, and model pruning optimization. In this study, by considering the implementation steps and the difficulty of different optimization methods, three aspects of optimization are carried out: backbone network adjustment, enhanced feature extraction network optimization, and loss function adjustment.

The network structure of YOLOv4 is divided into three main parts. In the first part, the function of the backbone feature extraction network is to extract the preliminary features. Using the backbone feature extraction network, we can obtain three preliminary effective feature layers. Because the original backbone network of YOLOv4, which is CSPDarkNet53, has deep network layers and we need to design a lightweight detection network, we can use the lightweight MobileNet [14] series network as the backbone feature extraction network. This retains the advantages of MobileNet without losing the original precision of YOLOv4. In the second part, the function of the enhanced feature extraction network is to extract the enhanced features. Using the enhanced feature extraction network, we can perform feature fusion of the three preliminary effective feature layers, extract better features, and obtain three more effective feature layers.

To design a lighter network, deep separable convolution can be used to replace the ordinary 3×3 convolution in the enhanced feature extraction network. By doing this, the number of network parameters is greatly reduced, and the extracted features are not significantly affected. In the third part, the function of the prediction network is to obtain the prediction result using the more effective feature layer. Of these three parts, parts 1 and 2 can be improved more easily. Part 3 does not require major modifications because it is just a combination of a 3×3 convolution and 1×1 convolution.

A. Backbone Network Adjustment

The MobileNet series network can be used for classification, and the network backbone extracts features. The aim of MobileNet is to create high-performance, low-resource networks that can be used on mobile phones. In addition, MobileNetv3 is an advanced version of MobileNet that mainly uses the neural architecture search (NAS) to design an algorithm. It improves the accuracy of the ImageNet classification by 6% over the MobileNetv2 [18] version, which is exactly what our lightweight detection network needs for a backbone extraction network. At the same time, MobileNetv3 mostly uses 1×1 and 3×3 convolution instead of 5×5 convolution. As a result, this greatly reduces the number of parameters.

In MobileNetv2, the 3×3 convolution is used first, and then the 1×1 convolution is used. In contrast, in MobileNetv3, the 1×1 convolution is used first, and then the 3×3 convolution is used. This not only preserves the high-dimensional feature space, it also reduces the delay in the backpropagation. In MobileNetv3, which includes a residual block and a lightweight attention mechanism, a lightweight attention module is introduced and integrated into the bottleneck structure to better extract the features [19]. Using H-Swish, the calculation speed is increased in mobile devices to improve the accuracy of the network.

For YOLOv4, we use the three effective features obtained from the backbone feature extraction network to build the enhanced feature pyramid. Specifically, the MobileNetv3 network is used to replace the CSPDarkNet53 in YOLOv4 for the feature extraction. Using a custom MobileNetv3 function, we obtain the effective feature layer corresponding to the MobileNet network. We use this effective feature layer to replace the effective feature layer of the original YOLOv4 backbone network, CSPDarkNet53, and strengthen the feature extraction with the three initial effective feature layers of the same shape. Subsequently, we integrate the MobileNet series network into YOLOv4. A schematic of the basic structure of MobileNetv3 is shown in Fig. 2 [13], and the detailed specifications of the entire MobileNetv3 network are given in Fig. 3.

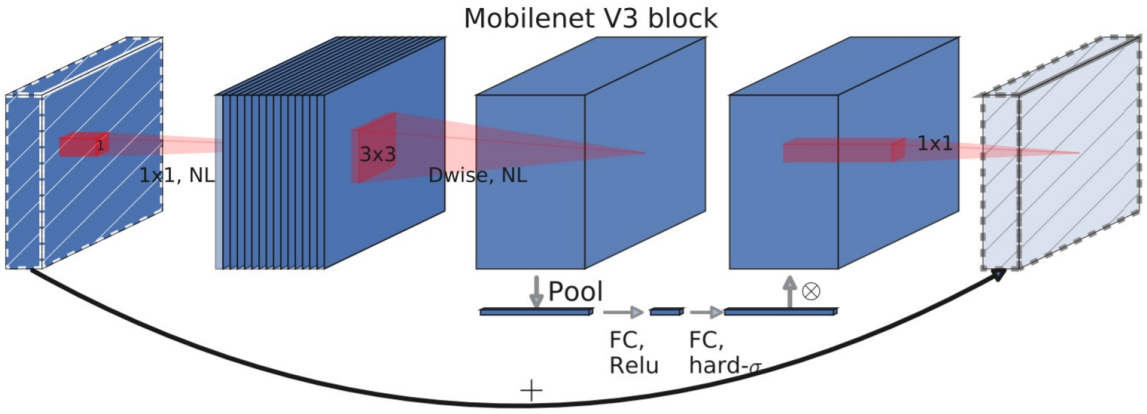


Fig. 2. Schematic diagram of the basic structure of MobileNetv3.

Input	Operator	exp size	#out	SE	NL	s
$224^2\times3$	conv2d	-	16	-	HS	2
$112^2\times16$	bneck,3×3	16	16	-	RE	1
$112^2\times16$	bneck,3×3	64	24	-	RE	2
$56^2\times24$	bneck,3×3	72	24	-	RE	1
$56^2\times24$	bneck,5×5	72	40	√	RE	2
$28^2\times40$	bneck,5×5	120	40	√	RE	1
$28^2\times40$	bneck,5×5	120	40	√	RE	1
$28^2\times40$	bneck,5×5	240	80	-	HS	2
$14^2\times80$	bneck,3×3	200	80	-	HS	1
$14^2\times80$	bneck,3×3	184	80	-	HS	1
$14^2\times80$	bneck,3×3	184	80	-	HS	1
$14^2\times80$	bneck,3×3	480	112	√	HS	1
$14^2\times112$	bneck,3×3	672	112	√	HS	1
$14^2\times112$	bneck,5×5	672	160	√	HS	2
$7^2\times160$	bneck,5×5	960	160	√	HS	1
$7^2\times160$	bneck,5×5	960	160	√	HS	1
$7^2\times160$	Conv2d,1×1	-	960	-	HS	1
$7^2\times960$	Pool,7×7	-	-	-	-	1
$1^2\times960$	Conv2d1×1,NBN	-	1280	-	HS	1
$1^2\times1280$	Conv2d1×1,NBN	-	k	-	-	1

Fig. 3. MobileNetv3 detailed specifications.

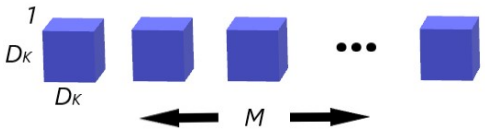
In Fig. 3, the input column lists the size changes in each feature layer of MobileNetv3. The operator column lists the block structure that each feature layer will go through. Feature extraction in MobileNetv3 goes through many bottlenecks. The third and fourth columns respectively specify the number of channels after the inverse residual structure rises in the bottleneck and the number of channels of the character layer when it is input into the bottleneck. The SE column indicates whether or not attention mechanisms are introduced at the various levels. The NL column indicates the type of activation function and the seventh column, s, indicates the step size of each block.

B. Strengthening the Feature Extraction Network Optimization

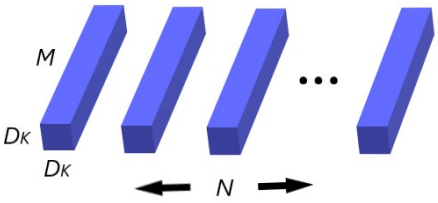
To further reduce the number of parameters, after using MobileNetv3 as the backbone extraction network, we use depth separable convolution to replace the conventional 3×3 convolution in the enhanced feature extraction networks SPP and PANet.

In conventional convolution, the upper layer of the connection generally has multiple channels; hence, in the convolution, a filter must have N kernels to correspond to the channels. A filter completes a convolution; in fact, multiple convolution kernels are convolved with the feature graph of the corresponding channel in the upper layer and then combined to output a feature graph of the channel in the next layer. In the next layer, if the characteristic graph of multiple channels is needed (m channels are assumed here), then the corresponding filter needs M.

In the case of depthwise convolution, the convolution of different input channels is performed using depthwise convolution, and then the outputs are combined using pointwise convolution. In fact, the overall effect is similar to that of a standard convolution, but it significantly reduces the computation and number of model parameters. Fig. 4 compares standard convolution and depth separable convolution kernels.



(a) Standard convolution filter



(b) Depthwise convolution filter

(c) Pointwise convolution filter

Fig. 4. Comparison of standard convolution and depth separable convolution.

In fact, depth separable convolution only makes a small change to the conventional convolution, but brings about a decrease in the number of true parameters, which invariably redounds to the network being lightweight. For the multi-channel feature maps from the upper layer, we first

split them into single-channel feature maps, respectively convolve them with a single channel, and then stack them together again. This is called depthwise convolution. This splitting action is very important; in this step, only the feature map from the previous layer is resized, and the number of channels does not change. Therefore, the second convolution of the previously obtained feature graph is performed, which takes the convolution kernels of size 1×1, and the filter contains the same number of convolution kernels as the number of channels in the previous layer. A filter outputs a feature graph, thus multiple channels require multiple filters.This is also called pointwise convolution. Fig. 5 shows a depth separable convolution structure.

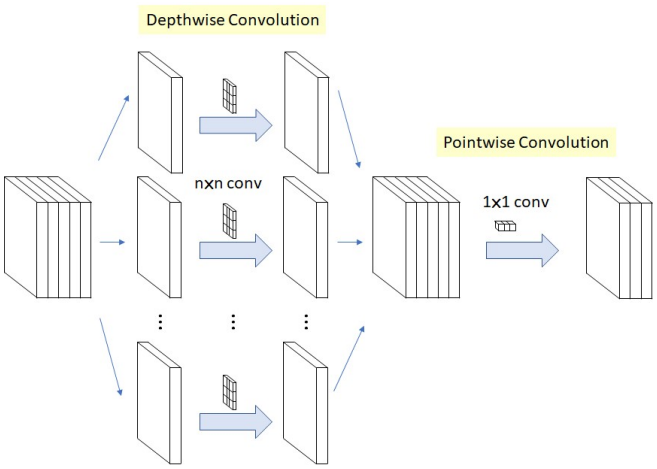


Fig. 5 Depth separable convolution structure.

In standard convolution, it is assumed that the image size is 5×5 and, as the number of color image channels is three, it can be seen as a 5×5×3 matrix. Assuming that there is no padding operation and the size of the convolution kernel is 3×3, because the number of output feature maps is four, the configuration of the convolution kernel is 3×3×3×4. Thus, we know that the number of parameters in this convolution is 3×3×3×4 = 108.

In depth separable convolution, a 3×3 convolution kernel is first used for the convolution operation for each channel, hence, the number of parameters should be 3×3×1×3 = 27. However, this processing method does not effectively utilize the feature information of different channels in the same space. Therefore, the second step is needed to integrate the three feature maps generated in this step

A 1×1 convolution is used to verify the three feature graphs obtained in the first step for the convolution operation. To achieve consistency of the output feature matrix shape with ordinary convolution, we need four convolution kernels. Thus, the number of parameters required in the second part is 1×1×3×4 = 12. With the same input, four feature graphs are finally obtained. The number of parameters for ordinary convolution is 108, whereas that for depth separable convolution is only 12+27 = 39, which is approximately two-thirds less than ordinary convolution.

It can be seen that the depth separable convolution significantly reduces the number of parameters and the computation required in the network, and the application of a neural network can effectively improve the running speed of the network.

C. *Optimization of the Loss Function*

Consider the problem of the loss being inaccurate because of the large number of negative samples when the positive and negative samples of the KITTI vehicle dataset after classification are imbalanced. The focal loss proposed in the RetinaNet [20] model was introduced to detect single-class vehicles. This is achieved by multiplying the original loss by the index of the weakened contribution of easily detected vehicle targets in network training. In comparison to YOLOv3, the original YOLOv4 only innovates the bounding box regression by replacing the MSE with CIOU, whereas the other two parts are not substantially changed. In the object-detection task, negative sample mining and sampling ratio control methods are often used to solve the imbalance in positive and negative samples. From this, a one-stage object detection network can also achieve two-stage accuracy and have a good detection speed. The focal loss is introduced mainly to solve the imbalance problem in terms of the number of difficult and easy samples.

The one-stage object detector usually produces candidate targets up to 100K. Only a small number of these are positive samples, and the number of positive and negative samples is very imbalanced. The formula for the cross-entropy, which is commonly used for classification, is shown in (1).

$$CE=\begin{cases} -log(p), & if \quad y=1 \\ -log(1-p), & if \quad y=0 \end{cases} \tag{1}$$

To solve the imbalance problem between the positive and negative samples, we usually add the parameter α in front of the cross-entropy loss, which is expressed as follows.

$$CE=\begin{cases} -\alpha log(p), & if \quad y=1 \\ -(1-\alpha) log(1-p), & if \quad y=0 \end{cases} \tag{2}$$

However, this does not solve the whole problem. The samples can be divided into the four categories shown in Table I.

TABLE I
SAMPLE CLASSIFICATION TABLE.

	D (difficult)	E (easy)
P (positive)	PD	PE
N (negative)	ND	NE

P = positive samples, N = negative samples, D = difficult samples, E = easy samples

Although it balances the positive and negative samples, it does nothing to help the difficult and easy sample imbalance. A large number of candidate targets in the object detection are easily divided into samples, as shown in Fig. 6.

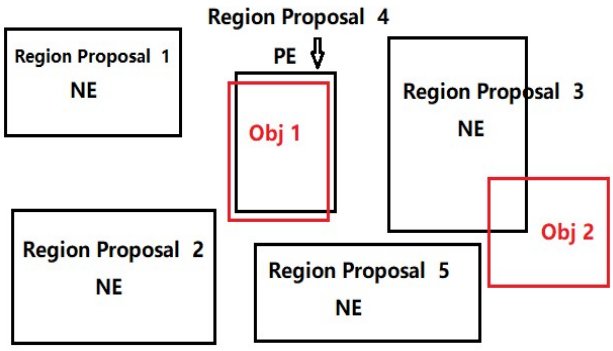


Fig. 6. Candidate boxes and samples.

The loss of these samples is low, but the number of easily divided samples is relatively large owing to the imbalance, and this ultimately dominates the total loss. However, the improvement in the model of the easily separable samples (i.e., samples with a high confidence) is very small; thus, the model should mainly focus on those samples that are difficult to separate. This problem can be solved by reducing the loss of the samples with a high confidence (P). Equation (3) is expressed as follows:

$$FL=\begin{cases}-(1-p)^{\gamma}\log(p), & if \quad y=1\\ -p^{\gamma}\log(1-p), & if \quad y=0\end{cases} \tag{3}$$

When γ is 2, if p=0.698, $(1\text{-}0.698)^2 \approx 0.001$, the loss is attenuated 1,000 times. In addition, the final form of the focal loss combines (2). Here, (3) solves the imbalance between the difficult and easy samples, and (2) solves the imbalance between the positive and negative samples. By combining (2) with (3), the two problems are solved simultaneously. The final focal loss is shown in (4).

$$FL=\begin{cases}-\alpha(1-p)^{\gamma}\log(p), & if \quad y=1\\ -(1-\alpha)p^{\gamma}\log(1-p), & if \quad y=0\end{cases} \tag{4}$$

The experimental results show that the effect is best when γ is 2 and α is 0.25. By doing this, the ordering of the objects in the training process is *PD* > *ND* > *PE* > *NE*. This is shown in Table II.

TABLE II
SAMPLE CLASSIFICATION TABLE.

	D (difficult)	E (easy)
P (positive)	① PD	③ PE, γ decline
N (negative)	② ND, α decline	④ NE, α, γ decline

The loss obtained in this manner induces the model to distinguish difficult-to-separate target categories. This effectively improves the overall object detection accuracy.

IV. Experimental Results and Analysis

A. Dataset Selection

In this study, public KITTI datasets were selected as training and testing samples, and actual road photos and part of the KITTI datasets were used for testing and verification. Further, the format of the KITTI dataset was changed to VOC format, the Labels categories were merged into only the CAR class, and the 7830 sorted images were divided into training set, test set, and verification set in a ratio of 8:1:1. Sample images from a part of the experimental dataset are shown in Fig. 7.

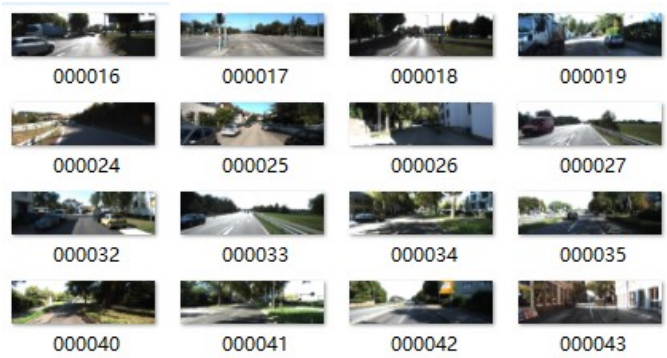


Fig. 7. Sample images from a part of the experimental dataset.

B. Experimental Results and Analysis

The experimental platform was composed of two parts: hardware and software. The hardware platform comprised an Intel Core i7-10700 CPU, an NVIDIA GeForce GTX 3070 GPU, and 8 GB RAM. The software environment comprised the Windows 10 operating system, the PyTorch1.8-GPU deep learning framework, and the PyCharm Community IDE.

The targets in the KITTI dataset were grouped into one class ("car"), and the two networks before and after the improvement, YOLOv4, and YOLOv4-MobileNetv3-best, were tested to calculate the target recall rate and detection accuracy. By changing the detection threshold on the test set, the model can detect the first Q pictures. In addition, changes in the threshold will also lead to changes in the accuracy and recall rate.

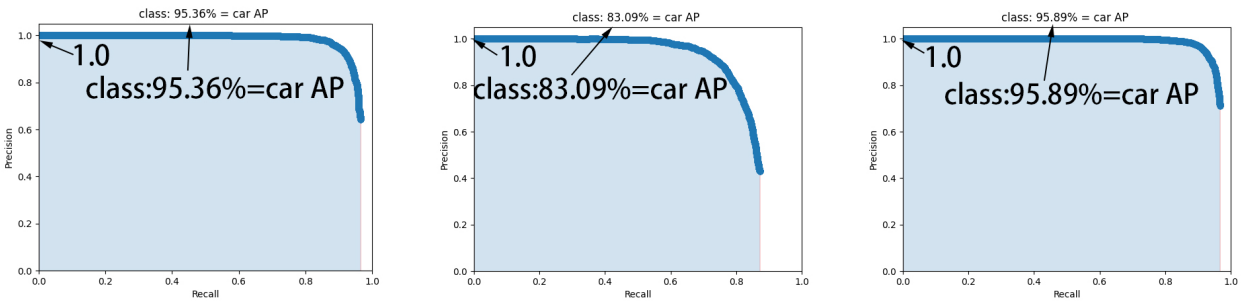
The P–R curve was established with recall rate as the abscissa and accuracy as the ordinate, and the area under the curve was defined as the average precision (AP). The higher the AP value, the better is the single-class object detection effect. The target recall rate R and detection accuracy rate P are respectively expressed in (5) and (6).

$$R=\frac{X_{\mathrm{TP}}}{X_{\mathrm{TP}}+X_{\mathrm{FN}}} \tag{5}$$

$$P=\frac{X_{\mathrm{TP}}}{X_{\mathrm{TP}}+X_{\mathrm{FP}}} \tag{6}$$

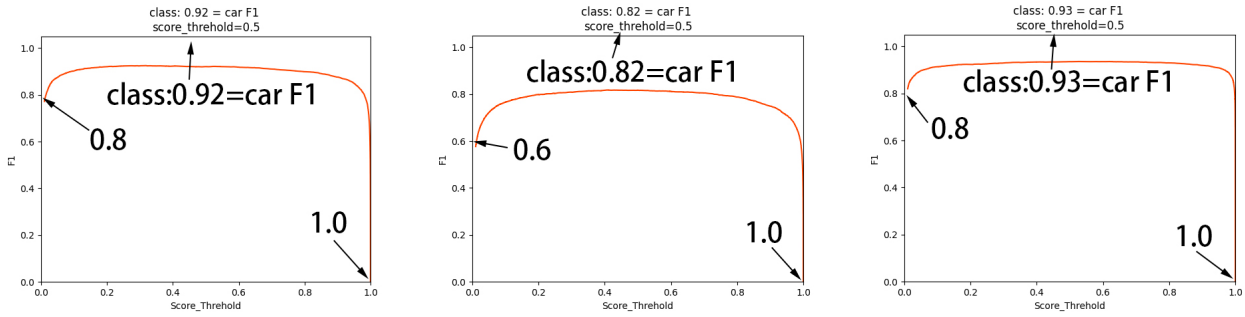
where X_{TP} denotes the number of correctly detected targets, X_{FN} denotes the number of targets that have not been detected, and X_{FN} denotes the number of targets that were wrongly detected.

There were 3,331 targets in 749 test images. YOLOv4 and our improved object detection algorithm were used to test for the KITTI dataset, and R and P were calculated, respectively. The P–R curves of the results are shown in Fig. 8.



(a) YOLOv4 (b) YOLOv4-Tiny (c) YOLOv4-MobileNetv3-best (Ours)

Fig. 8. Comparison of the P–R curves of three models.



(a) YOLOv4 (b) YOLOv4-Tiny (c) YOLOv4-MobileNetv3-best (Ours)

Fig. 9. Comparison of the F1 curves of three models.

F1 refers to the harmonic average of accuracy and recall, which are respectively calculated using (7) and (8):

$$\frac{2}{F1}=\frac{1}{P}+\frac{1}{R} \tag{7}$$

$$F1=\frac{2X_{\mathrm{TP}}}{2X_{\mathrm{TP}}+X_{\mathrm{FP}}+X_{\mathrm{FN}}} \tag{8}$$

With the confidence degree (Score_Threshold) on the x-axis, the F1 value on the y-axis, and the area under the curve of the F1 index to evaluate the model, the obtained F1 curve results of a single-vehicle class are shown in Fig. 9.

On the KITTI dataset, in comparison to YOLOv4 and YOLOv4-Tiny, the detection accuracy of the improved YOLOv4 algorithm increased from 95.55% to 95.63%. In comparison to YOLOv4-Tiny, its AP value increased by 12.8% and the recall rate increased from 87.84% to 91.50%. The average accuracy (AP) of the vehicle object detection was calculated using the two networks, respectively. The improved YOLOv4 network improves the AP of vehicle object detection from 95.36% to 95.89%. Table III shows the detection results for 3,331 targets in 749 test pictures.

As demonstrated in Table III, the recall rate of our optimized model was significantly improved. There were 3,331 targets in the test pictures. YOLOv4 correctly detected 3,190 and missed 441, whereas our optimized model correctly detected 3,221 and missed only 299. Our model is significantly better than the previous algorithm with respect to AP and F1. In addition, our chosen model greatly reduces the number of parameters. The sizes of CSPDarkNet-53 and MobileNetv3 are shown in Table IV. We tested the YOLOv4 basic model and the optimized YOLOv4-MobileNetv3 model in this study. Table V compares the model sizes.

According to the detection results, the AP value of the YOLOv4 detection algorithm, which adopts the improved MobileNet series as the backbone network, is above 93% for vehicle target detection. The model with the best performance was used for the target occlusion test. The detection results are shown in Fig. 10. On the right are two partially occluded vehicles, both of which have a detection confidence of 1.0.

(a) no shade

(b) covered

Fig. 10. YOLOv4-MobileNetv3-best detection results.

TABLE III
PERFORMANCE COMPARISON OF YOLOV4 AND THE VARIOUS VARIANTS.

Model	TP	FP	FN	Ap（%）	Recall（%）	Precision（%）	F1
YOLOv4	3190	148	441	95.36	87.84	95.55	0.92
YOLOv4-Tiny	2905	334	973	83.09	74.90	89.68	0.82
YOLOv4-MobileNetv1	3183	187	528	93.70	85.77	94.45	0.90
YOLOv4-MobileNetv2	3185	201	551	93.19	85.23	94.07	0.89
YOLOv4-MobileNetv3	3216	194	518	93.89	86.12	94.31	0.91
YOLOv4-MobileNetv3-best (Ours)	3221	147	299	95.89	91.50	95.63	0.93

TABLE IV
COMPARISON OF THE NUMBER OF NETWORK PARAMETERS.

Network	Parameters
CSPDarkNet	37.9M
MobileNetv3	3.4M

Table V shows the numbers of test parameters obtained after replacing all the 3×3 convolutions in the enhanced feature extraction network module of YOLOv4 with deep separable convolutions.

TABLE V
COMPARISON OF NUMBERS OF PARAMETERS.

MODEL	PARAMETERS
YOLOv4	64,040,001
YOLOv4-MobileNetv1	40,952,893
YOLOv4-MobileNetv2	39,062,013
YOLOv4-MobileNetv3	39,989,933
YOLOv4-MobileNetv3-best（Ours）	11,729,069

Compared with YOLOv4-MobileNetv3 of the enhanced feature weight enhancement network, the number of parameters of the proposed method is reduced by nearly 75%.

Our optimized model also greatly reduces the size of the weight file of the model. This not only reduces the memory consumption for some small devices, it also makes it easier to implement for embedded platforms.

Table VI compares the size of the weight file after the model is trained and the inference time on the CPU for weights trained by the improved model, the original YOLOv4 model, and the other improved models. Owing to the different computing methods of the CPU and GPU, the data read by the CPU comes from the memory and cache. In addition, the read speed from the cache is much faster than the memory, whereas the data read by the GPU differ.

TABLE VI
COMPARISON OF THE NUMBER OF PARAMETERS AND INFERENCE TIME.

Model	Parameters	INFERENCE TIME
YOLOv4	250.6M	166 ms
YOLOv4-Tiny	23.0M	49 ms
YOLOv4-MobileNetv1	72.3M	72 ms
YOLOv4-MobileNetv2	77.0M	65 ms
YOLOv4-MobileNetv3	75.0M	71 ms
YOLOv4-MobileNetv3-best (Ours)	52.0M	62 ms

Compared with the CPU, the GPU is hardly affected by the cache in operations such as convolution. MobileNetv3 utilizes the standard convolution decomposition operation to obtain a cache hit ratio at more levels. Therefore, MobileNetv3 is more friendly with respect to CPU computing. In this regard, this study takes the inference time on the CPU as the comparison index. Our optimized model not only reduces the memory usage for some small devices, it also makes it easier to implement for embedded platforms.

Considering the data in Table VI, the optimized YOLOv4-MobileNetv3-best model in this study reduced the number of weight parameters by 78% compared with the basic YOLOv4 model. The inference time on the CPU is reduced by 104 ms, which meets the real-time detection requirements. The visual detection results of the model on the KITTI dataset are shown in Fig. 11.

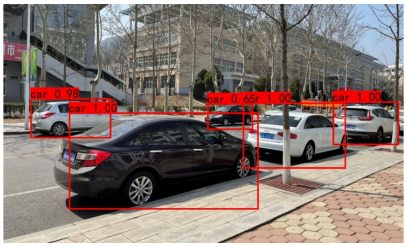
(a) Original image

(b) YOLOv4

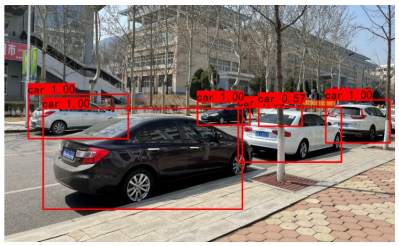
(c) YOLOv4-MobileNetv3-best (Ours)
Fig. 11. Visualization results of the KITTI dataset.

As illustrated in Fig. 11(b), the YOLOv4 detection results show six vehicles with a confidence level higher than 0.5, whereas Fig. 11(c) displays the optimized model detection results, which show a total of seven vehicles with a confidence level higher than 0.5. Our algorithm performs better in the case of vehicles with occlusion and a small target in the middle of the graph. When detecting vehicles in the

(a) Original image

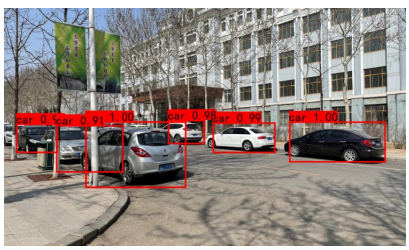


(b) YOLOv4



(c) YOLOv4-MobileNetv3-best (Ours)

(a) Original image



(b) YOLOv4

(c) YOLOv4-MobileNetv3-best (Ours)
Fig. 12. Visualization result of actual scene

KITTI dataset, the improved algorithm has a significantly better performance than the original YOLOv4 algorithm, and the rate of the missed detection is also greatly reduced. The visualization results obtained using our trained KITTI data to detect the actual road vehicles on the campus are shown in Fig. 12.

Fig. 12(a) presents the original image, Fig. 12(b) displays the test result of YOLOv4, and Fig. 12(c) shows the test result of our improved model. The detection confidence of our algorithm is 0.54 for the occluded vehicles at the entrance to the building in the scene on the left. However, YOLOv4 may miss some detections in this case. Similarly, on the right, the vehicle with a confidence of 0.57 indicates a missed detection vehicle for YOLOv4; thus, our model is more sensitive to occlusion. The improved model can still maintain a high recall rate and accuracy under the conditions of small targets and occlusion. Through a series of experimental comparisons, it can be observed that the improved model not only greatly reduces the number of model parameters, it also maintains, or even improves, the original detection accuracy. At the same time, it maintains a good performance in different environments, improves the application range of the algorithm, and has more practical value.

V. Conclusion

In this study, an improved YOLOv4-based vehicle target detection algorithm was proposed and applied to lightweight detection. Experimental results showed that, compared with the original YOLO series algorithm, the accuracy of our model improved by 0.53%, and the number of model parameters was reduced by 78%. In comparison to the other algorithms, the improved YOLOv4 model is smaller and more accurate, thus it can be used in lightweight vehicle detection networks. It provides a theoretical basis for vehicle detection on a smaller hardware platform. From the perspective of the current research in the field of intelligent connected vehicle environment perception, the research at the image level only considers the camera as a sensor, which includes target detection, target tracking, and semantic or instance segmentation. Owing to the fine precision degree and speed, there is room for improvement. The industry commonly uses sensor fusion that integrates a radar point cloud with camera information. This is done to obtain a higher degree of scene understanding. The novelty of this study is that it significantly optimizes the speed of resource-constrained systems when performing target detection. Meanwhile, the calculation time and space are minimized for the subsequent fusion. However, the problem of vehicle occlusion and target deformation without reducing the detection accuracy of the algorithm has not been completely solved; follow-up research will focus on this aspect.

References

[1] J. Li, X. Liang, Y. Wei et al., "Perceptual Generative Adversarial Networks for Small Object Detection," *in Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2017, pp. 1222-1230.

[2] Y. Zhang, S. Song, E. Yumer et al., "Physically-Based Rendering for Indoor Scene Understanding Using Convolutional Neural Networks," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2017, pp. 5287-5295.

[3] R. Girshick, J. Donahue, T. Darrell et al., "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2014, pp. 580-587.

[4] X. Wang, A. Shrivastava, A. Gupta, "A-Fast-RCNN: Hard Positive Generation via Adversary for Object Detection," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2017, pp. 2606-2615.

[5] S. Ren，K. He, R. Girshick et al., "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, 2017, vol. 39, no. 6, pp. 1-9.

[6] T. Y. Lin, P. Dollár, R. Girshick et al., "Feature Pyramid Networks for Object Detection," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2017, pp. 2117-2125.

[7] J. Redmon, S. Divvala, R. Girshick, "You Only Look Once: Unified, Real-Time Object Detection," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2016, pp. 779-788.

[8] W. Liu，D. Anguelov，D. Erhan et al．, "SSD: single shot multibox detector," *Eur. Conf. Comput. Vis.*，2016, pp. 21-37.

[9] T. Y. Lin，P. Goyal，R. Girshick et al., "Focal Loss for Dense Object Detection," in *Proc. IEEE Intl. Conf. Comput. Vis. (ICCV)*, 2017，pp. 2980-2988.

[10] X. Zhu, J. Pang, C. Yang, J. Shi, D. Lin, "Adapting Object Detectors via Selective Cross-Domain Alignment," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2019, pp. 687-696.

[11] P. A. Viola, M. J. Jones, "Rapid Object Detection using a Boosted Cascade of Simple Features," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2001, pp. 1-9.

[12] B. Alexey, W. Chien-Yao, M. L. Hong-Yuan, "YOLOv4:Optimal Speed and Accuracy of Object Detection," arXiv：2004.10934, 2020.

[13] H. Andrew, S. Mark, C. Grace et al., "Searching for MobileNetV3," in *Proc. IEEE/CVF Intl. Conf. Comput. Vis. (ICCV)*, 2019, pp. 1314-1324.

[14] H. Andrew, Z. Menglong, C. Bo et al., "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," arXiv:1704.04861, 2017.

[15] P. A. S. Mendes, M. Mendes, A. P. Coimbra, M. M. Crisostomo, "Movement Detection and Moving Object Distinction Based on Optical Flow," Lecture Notes in Engineering and Computer Science: Proceedings of The World Congress on Engineering 2019, 3-5 July 2019, London, U.K., pp. 48-53.

[16] G. Kosala, A. Harjoko, S. Hartati, "Robust License Plate Detection in Complex Scene using MSER-Dominant Vertical Sobel," *IAENG Intl. J. Comput. Sci*., vol. 47, no. 2, 2020, pp. 214-222.

[17] T. T. Yang, S. Y. Zhou, and A. J. Xu, "Rapid Image Detection of Tree Trunks Using a Convolutional Neural Network and Transfer Learning," *IAENG Intl. J. Comput. Sci.*, vol. 48, no. 2, 2021, pp. 257-265.

[18] S. Mark, H. Andrew, Z. Menglong, et al., "MobileNetV2：Inverted Residuals and Linear Bottlenecks,"in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.* (CVPR), 2018, pp. 4510-4520.

[19] H. Kaiming, G. Georgia, D. Piotr, et al., "Mask R-CNN," in *Proc. IEEE Intl. Conf. Comput. Vis. (ICCV)*, 2017, pp. 2961-2969.

[20] Y. Zhang, S. Song, E. Yumer et al., "Physically-Based Rendering for Indoor Scene Understanding Using Convolutional Neural Networks," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.* (CVPR), 2017, pp. 5287-5295.