BULLETIN OF THE POLISH ACADEMY OF SCIENCES
TECHNICAL SCIENCES, Vol. 70(6), 2022, Article number: e143644
DOI: 10.24425/bpasts.2022.143644

Vehicle detection in surveillance videos based on YOLOv5 lightweight network

Yurui WANG*, Guoping YANG and Jingbo GUO

Shanghai University of Engineering Science, School of Mechanical and Automotive Engineering, Shanghai, China

Abstract. The development of surveillance video vehicle detection technology in modern intelligent transportation systems is closely related to the operation and safety of highways and urban road systems. Yet, the current object detection network structure is complex, requiring a large number of parameters and calculations, so this paper proposes a lightweight network based on YOLOv5. It can be easily deployed on video surveillance equipment even with limited performance, while ensuring real-time and accurate vehicle detection. Modified MobileNetV2 is used as the backbone feature extraction network of YOLOv5, and DSC "depthwise separable convolution" is used to replace the standard convolution in the bottleneck layer structure. The lightweight YOLOv5 is evaluated in the UA-DETRAC and BDD100k datasets. Experimental results show that this method reduces the number of parameters by 95% as compared with the original YOLOv5s and achieves a good tradeoff between precision and speed.

Key words: YOLOv5; MobileNetV2; lightweight network; vehicle detection.

1. INTRODUCTION

With the rapid development of the economy, vehicle ownership also increases. Vehicles not only bring convenience to people, but also exert great pressure on the transportation system. Therefore, building an intelligent transportation system has become particularly important [1]. First-hand information on traffic conditions is mainly obtained through real-time road video surveillance. Information extraction from surveillance videos is critical for the construction of intelligent transportation, and object detection is a prerequisite for system decision-making. It can be seen that vehicle detection based on surveillance videos is of great significance [2].

The neural network algorithm promotes rapid development of object detection technology [3]. The current algorithm also performs well in the deployment of high-performance devices, but with the continuous improvement of detection accuracy requirements, the complexity of the model also increases. The resulting huge amount of computation makes it difficult to deploy in mobile or resource-limited devices [4].

This paper proposes a lightweight model based on YOLOv5 to solve the problems of limited computing performance of road monitoring equipment and high requirements for real-time performance and detection accuracy of the equipment. The main contributions of this paper are summarized as follows:

First, the lightweight MobileNetV2 network is used to replace the backbone feature extraction network of YOLOv5. MobileNetV2 is redesigned according to the task requirements. We reduce the number of layers of the network and make the backbone network more lightweight. This method combines the efficient feature extraction network of MobileNetV2 with the excellent detection framework of YOLOv5, effectively reduces parameters and computation, reduces the complexity of the model, and greatly improves the deployment capability of the model on embedded devices with limited performance.

Second, in the neck network, DSC is introduced to replace the 3×3 standard convolutional network in bottleneck block. And the number of bottlenecks in the C3 layers is set to 1 to reduce the convolution branch. We adopt an efficient convolution method to reduce redundant computing while ensuring information fusion between channels.

Finally, experiments are carried out on the UA-DETRAC and BDD100K datasets and compared with some existing methods. The results show that the detection network proposed in this paper not only has fewer model parameters and faster reasoning speed, but also maintains higher accuracy, which is conducive to wide application of the model on mobile terminals and small embedded devices. At the same time, it can also provide other researchers with integration ideas of different networks, which is conducive to creating a more efficient network structure.

2. RELATED WORK

As an important branch of computer vision, object detection is widely used in autonomous driving, intelligent transportation, etc. Traditional object detection is slow and low in accuracy as a result of manually extracting features [5–7].

The development of deep learning technology has greatly improved the performance of detection algorithms, but these models consume more computing resources and are difficult to deploy in environments with limited computing performance. Therefore, it is important to design smaller and more efficient networks for such limited environments [8].

*e-mail: wangyuruistar@163.com

Manuscript submitted 2022-05-31, revised 2022-10-18, initially accepted for publication 2022-10-22, published in December 2022.

© 2022 The Author(s). This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/)

2.1. Object detection based on deep learning

Object detection algorithms based on deep learning can be divided into three categories: object detection based on region proposal, object detection based on regression and object detection based on transformers. The first method divides the detection task into two parts, first extracting candidate regions, and then classifying and regressing the candidate regions. The second method is to directly regress the predicted object, mainly including SSD and YOLO series [9]. The third method is to convert the images to sequences for processing and prediction.

As a representative of the two-stage detection algorithm, R-CNN has proposed a series of detection algorithms after continuous improvement and research. In 2014, Girshick *et al*. applied CNN to the object detection task and proposed R-CNN [10], which combined AlexNet with a selective search algorithm to decompose the object detection task into several independent steps, which greatly improved the performance of the object detection task. The accuracy of object detection is higher, but computational efficiency is low. In the same year, He Kaiming's team proposed the spatial pyramid pooling network (SPP-Net) [11], where the input does not need a fixed size and avoids repeated feature extraction, greatly reducing the amount of computation. In 2015, Girshick was inspired by the SPP-Net algorithm, simplified the SPP layer, and improved feature utilization efficiency through shared convolution computation. Then he proposed fast R-CNN [12].

The object detection algorithm based on region proposal achieves high detection accuracy, but expansion of the model increases the amount of network computation, and real-time requirements cannot be well solved.

In 2016, Redmon *et al*. proposed the first one-stage object detection method based on deep learning, i.e. YOLO [13], which divides the input image into a fixed number of grids, and each grid is responsible for predicting that the center falls into the grid. The goal of the lattice is directly regressed to the bounding box and class probability. After that, Redmon improved YOLOv1 and proposed YOLOv2 [14], which introduced mechanisms such as anchor boxes, high resolution classifier and dimension clusters. In 2018, Redmon et al. proposed YOLOv3 [15] based on YOLOv2, modified the softmax loss function in YOLOv2 to cross entropy loss, introduced FPN, and deepened the network structure through residual connections. In 2020, Bochkovski proposed YOLOv4 [16], using Mosaic to achieve data augmentation, designed the CSPDarkNet53 network structure, added the PANet structure, and replaced Iou Loss with CIoU Loss, which further improved network performance. In the same year, Ultralytics released YOLOv5, adding auto learning bounding box anchors, GIOU loss, using Leaky ReLU and sigmoid activation functions, providing models of different sizes, making the application of the model more flexible. After that, many researchers presented some experienced improvements to the YOLO series, and proposed a variety of high-performance detectors. Chien-Yao Wang *et al*. proposed a multi-task unified network: YOLOR [17]. It can encode explicit knowledge and implicit knowledge at the same time, perform kernel space alignment, prediction refinement and multi-task learning in the network, and form a unified representation for multiple tasks to complete various tasks. Zheng Ge *et al*. proposed YOLOX [18], switching the YOLO detector to an anchor-free manner and conducting a decoupled head and the leading label assignment strategy SimOTA to achieve state-of-the-art results across a large scale range of models; Chien-Yao Wang *et al*. proposed a network scaling approach that modifies not only the depth, width and resolution, but also structure of the network, and released the scaled YOLOv4 [19].

Inspired by the power of the transformer in NLP, recently researchers extend the use of transformers to object detection tasks. Carion proposed a detection transformer (DETR) [20], which is a simple and fully end-to-end object detector. DETR treats the object detection task as an intuitive set prediction problem and gets rid of traditional hand-crafted components such as anchor generation and non-maximum suppression (NMS) post-processing. Zhu *et al*. proposed deformable DETR [21] to address longer training schedules and poor performance for small objects. The deformable attention module attends to a small set of key positions around a reference point. This way, computational complexity is greatly reduced and it also benefits from fast convergence. More importantly, the deformable attention module can be easily applied for fusing multi-scale features. Aiming at the high computation complexity problem of DETR, Zheng *et al*. propose an adaptive clustering transformer (ACT) [22] to replace the self-attention module of the pre-trained DETR model. ACT adaptively clusters the query features using a locality sensitivity hashing method and broadcasts the attention output to the queries represented by the prototypes selected.

2.2. Overview of lightweight network development

In order to satisfy the deployment of neural networks to platforms with limited storage space and computing resources, many researchers have devoted themselves to the research of small neural networks.

The SqueezeNet uses the Fire module for parameter compression, while SqueezeNext introduces separate convolutions on this basis for improvement [23, 24]. ShuffleNetV1 proposes a channel reorganization operation, which allows the network to fully use grouped convolution to accelerate, while ShuffleNetV2 proposes a channel split operation, which accelerates the network while reusing features and thus achieves good results [25, 26]. The MobileNet series is a lightweight network proposed by the Google team. MobileNetV1 uses DSC to replace standard convolutions and builds a lightweight network. MobileNetV2 innovatively proposes the inversed residual block with linear bottlenecks [27, 28]. MobileNetV3 is tuned to mobile phone CPUs through a combination of hardware aware network architecture search (NAS) complemented by the NetAdapt algorithm and then subsequently improved through novel architecture advances [29]. GhostNet proposes a novel ghost module, which can apply a series of linear transformations to generate many ghost feature maps that could fully reveal information underlying intrinsic features [30]. EfficientNet proposes a new scaling method that uniformly scales all dimensions of depth/width/resolution using a simple yet highly effective compound coefficient [31].

The design of a neural network needs to balance real-time performance and accuracy. Purchasing high-performance equipment will increase the cost of industrial production. Therefore, in order to adapt to low-performance equipment, this paper conducts research on efficient lightweight network structures.

3. LIGHTWEIGHT NETWORK DESIGN

The YOLO series has significant advantages in balancing accuracy and real-time performance. YOLOv5 can design different models by adjusting parameters, so that these models have higher flexibility and versatility in practical applications. Therefore, this chapter makes network modifications based on YOLOv5-v5.0 released by Ultralytics. At the same time, considering the limited computing power of traffic monitoring equipment, YOLOv5s is selected as the baseline network for lightweight design in order to reduce the amount of parameters and calculations while ensuring high precision.

3.1. YOLOv5 model

As shown in Fig. 1, the YOLOv5 network consists of three parts: backbone, neck and head.

The backbone network uses the C3 module to extract image features. First, the image is sliced through the focus module to obtain the initial multi-channel feature map. Then the features are extracted through the cascade connection of multiple convolutional layers and C3 modules, so that feature maps which have different sizes are obtained. The C3 module divides the basic feature map into two parts. One part passes through the multi-layer bottleneck, and then it merges with the other part across stages to alleviate the problem of gradient disappearance. The neck network fuses feature maps from different layers and passes them to the prediction layer to detect objects of different sizes. In the feature fusion process, the feature pyramid structure of FPN and PAN is used. The FPN structure transfers strong semantic features from top to bottom, and the PAN structure transfers strong localization features from lower feature maps to higher feature maps. In addition, YOLOv5 also uses the C3 module at this stage to enhance the feature fusion capability.

3.2. YOLOv5 network lightweight

In this section, due to the YOLOv5s model having a large number of parameters and high computational complexity, the original backbone network is replaced with modified MobilenetV2. Then channel pruning and convolution replacement are performed on the neck network to make the newly proposed model more lightweight. The schematic diagram of newly proposed model architecture is shown in Fig. 2.

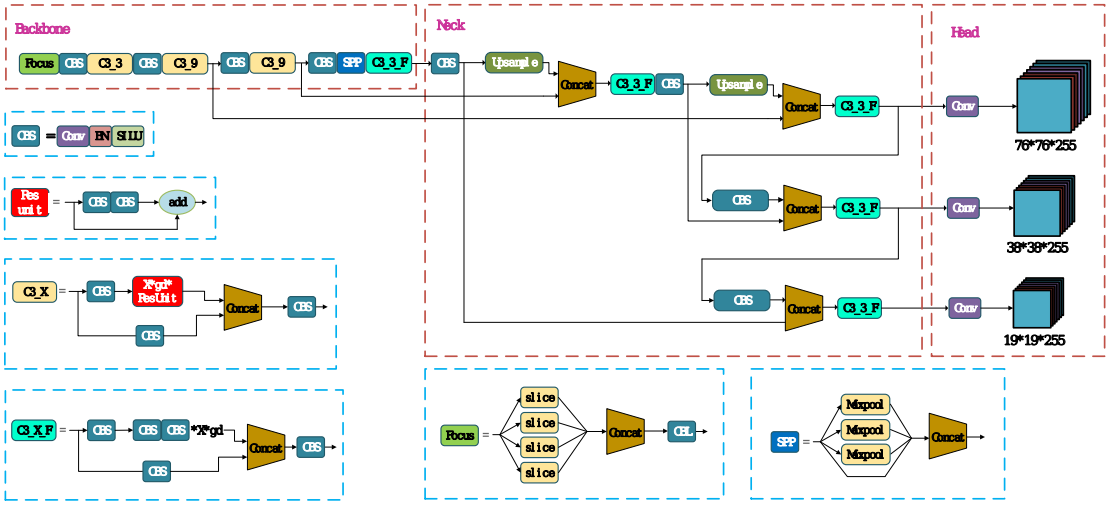

Fig. 1. Overall framework of the YOLOv5 model

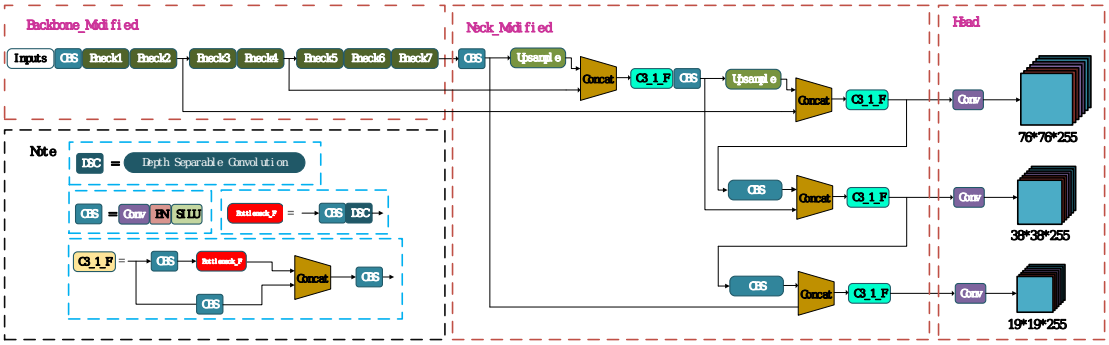

Fig. 2. Overall framework of the Mob_YOLOv5 Model

3.2.1. Backbone network lightweight

The YOLOv5 backbone network includes the focus layer and multiple C3 layers. The main function of the focus layer is to reduce parameters and accelerate the model without reducing the feature extraction capability of the model. The model performs well on GPU devices. But for embedded devices, the slicing operation will occupy a lot of cache and increase the burden of the computing processing. The C3 layer adopts multi-way branch convolution, which also occupies more cache space and reduces the running speed. Therefore, it is necessary to design a more lightweight backbone network.

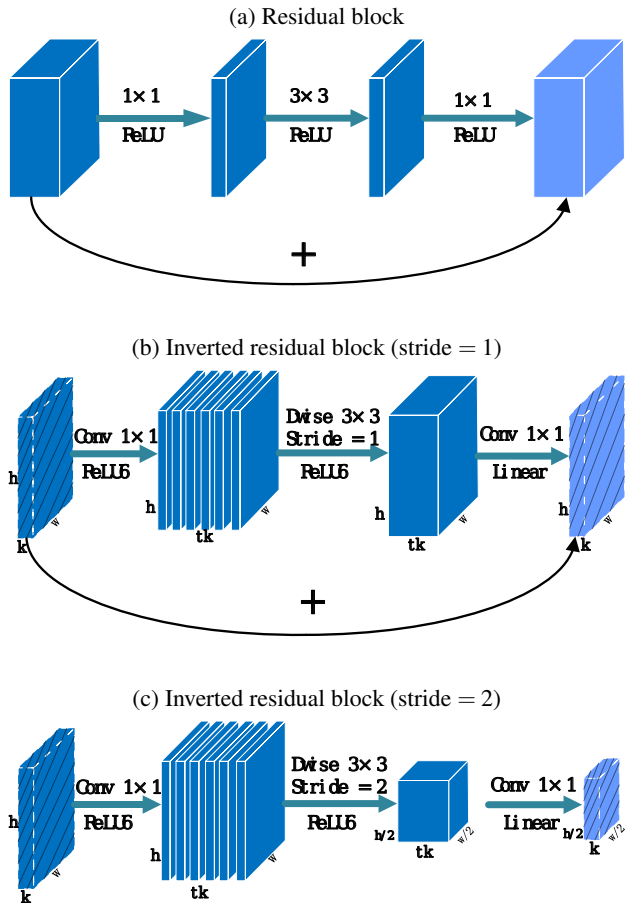

Fig. 3. Residual block and inverted residual block

MobileNetV2 is a lightweight network specially designed for mobile terminals. It refers to the idea of residual connection and proposes an inverse residual block with a linear bottleneck layer. The difference between the residual block [32] and the inverse residual block is shown in Fig. 3a, b. Different from the traditional residual block, the inverse residual block first dilates and then compresses the number of channels of the feature map. First it uses 1×1 convolution to expand the dimension of the feature map. Then standard convolution is replaced by depthwise convolution for feature extraction, and finally 1×1 convolution is used for dimensionality reduction output. Depthwise convolution uses multiple two-dimensional convolution kernels to operate each corresponding channel of the feature map, which can significantly reduce the amount of computation. After using 1×1 convolution to compress the channel, the ReLU activation function is replaced with a linear activation function, which can reduce the information loss of feature maps.

The schematic diagram of the modification of the backbone network is shown in Fig. 4. The backbone feature extraction network of YOLOv5 is replaced with the modified MobileNetV2. First, we set the number of repetitions of the bottleneck module to 1. This will reduce the original bottleneck layer number from 17 to 7. It can avoid extracting features repeatedly and greatly reduce the number of parameters and computation. Furthermore, due to the dataset in this paper not having complex classification tasks, we directly make feature fusion on the output of the last bottleneck layer. The last convolutional and pooling layers are removed, so that the backbone network becomes more lightweight.

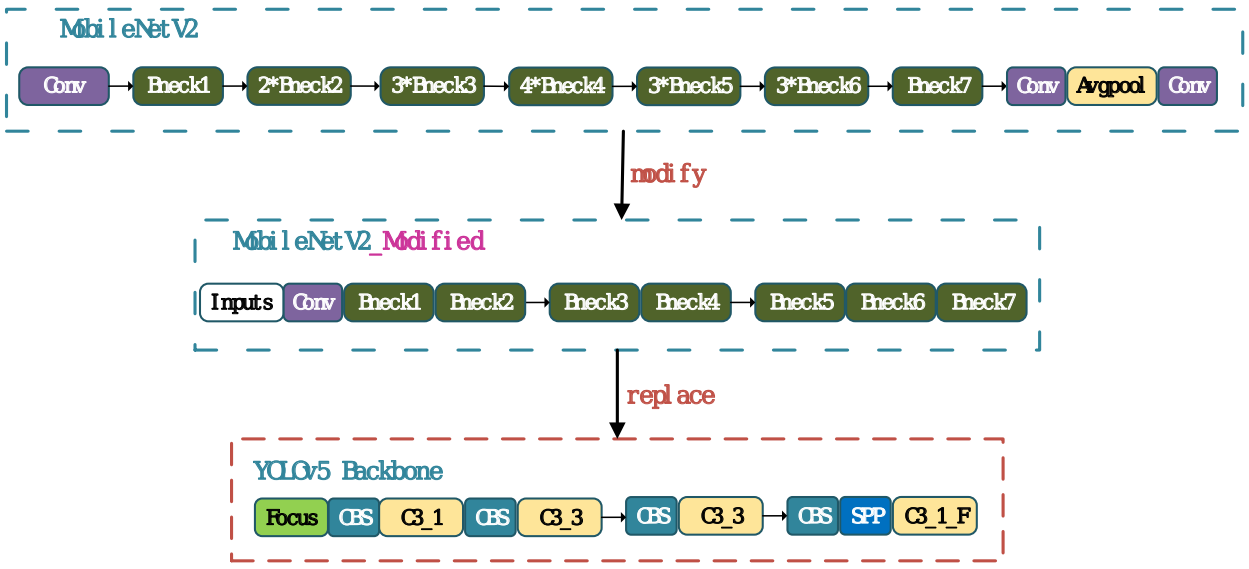

Fig. 4. The modified backbone network

3.2.2. Neck network optimization

In this section, the bottleneck module of the C3 layer, which is located in the neck of YOLOv5, is improved. The 3×3 convolutional standard is replaced with DSC to further reduce the network computational cost in practical applications. The C3 layer uses demultiplexed convolution. If the network is too fragmented (especially multi-channel), the degree of parallelism will be reduced. Therefore, the number of bottleneck modules in the C3 layer is set to 1. This can simplify the network and speed up the inference speed of the model. The modified area of the C3 module is shown in Fig. 5. Standard convolution uses weight matrixes to implement convolution calculations of position and channel dimensions, so the network has high computational complexity, high memory usage and many weight coefficients. Compared with standard convolution, DSC has the advantage of high computational efficiency and is usually used to build lightweight models.

The depthwise separable convolution consists of two parts: depthwise convolution and point convolution. The depthwise convolution is to convolve each channel of the input feature map separately, and then concatenate the outputs of all convolution kernels to obtain its final output. Point convolution is a 1×1 standard convolution. On the one hand, it is used to change the number of output channels. On the other hand, it can perform channel fusion on feature maps to make up for the lack of depthwise convolution cross-channel information interaction [33]. The process is shown in Fig. 6. Compared with standard convolution, using DSC can greatly reduce the computational cost. The following formula is the ratio of the calculation amount of DSC to the calculation amount of standard convolution.

$$\frac{D_K \cdot D_K \cdot M \cdot D_F \cdot D_F + M \cdot N \cdot D_F \cdot D_F}{D_K \cdot D_K \cdot M \cdot N \cdot D_F \cdot D_F} = \frac{1}{N} + \frac{1}{D_K^2}, \quad (1)$$

D_K is the size of the convolution kernel M is the number of input channels N is the number of output channels D_F is the size of the feature map.

4. EXPERIMENT AND ANALYSIS

4.1. Datasets and environments

First, this paper evaluates the model on the UA-DETRAC [34] dataset, and then it tests the generalization ability of the model on the BDD100k [35] dataset. The UA-DETRAC dataset is a screenshot from the monitoring of 24 different regional roads in Beijing and Tianjin. The dataset contains 138 252 training and test pictures, with a total of 1.21 million labeled bounding boxes of objects. BDD100K is currently the largest driving dataset, containing 100 000 images (train: 70%, val: 10%, test: 20%), and annotated with 13 categories. This paper selects the Car category for training.

The experimental environment is shown in Table 1. The hardware used for this experiment was an NVIDIA RTX 2070 graphics card, single GPU, 8G memory; win10 operating system; CUDA version 10.2; PyTorch version 1.10.0. The compiled virtual environment of Python was v3.6.

In order to ensure fairness of the experiments, the same initial training parameters were set for each group of experiments.

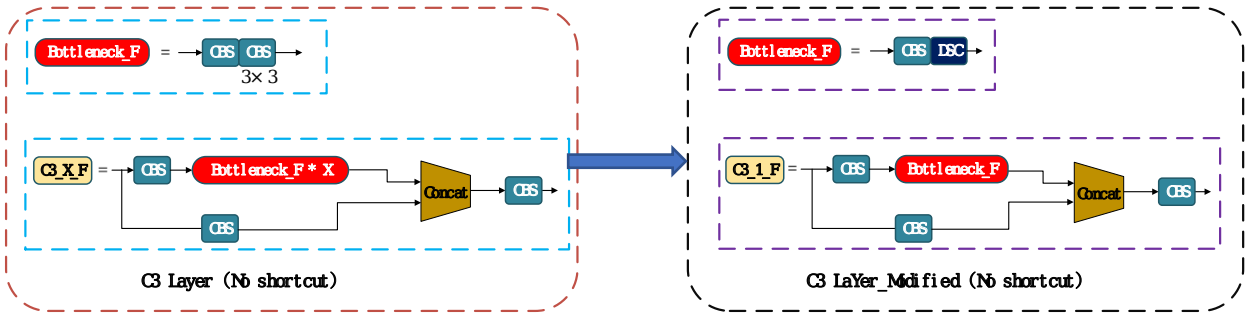

Fig. 5. C3 layer modification area

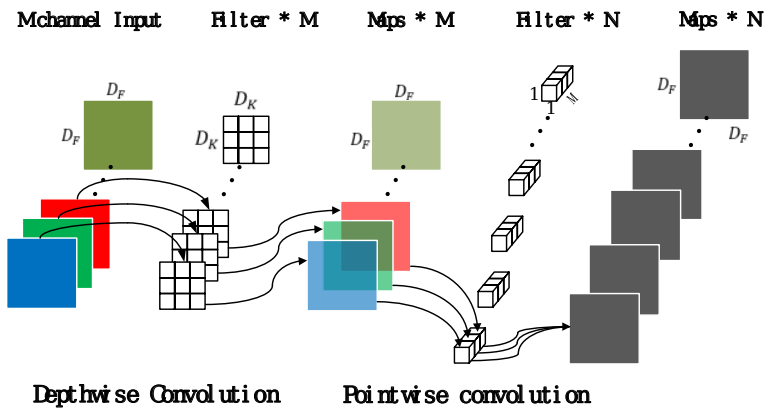

Fig. 6. Schematic diagram of Depthwise Separable Convolution

Table 1
Operating environment configuration

Component	Configuration
Operating system	Windows 10
CPU	Intel(R) Core(TM) i7-10750H
Memory	8
GPU	Nvidia GeForce RTX2070
GPU acceleration library	CUDA 10.2 cuDNN v8.2.0
Deep learning framework	Torch 1.10.1 torchvision 0.11.1
Programming language	Python3.6

The input resolution was uniformly resized to 640×640. The training batchsize and epoch are set to 16 and 250. The Adam algorithm was adopted to optimize the loss function. The initial learning rate was set to 0.001. Data augmentation and other parameters were kept the same as the default setting. None of the experiments adopted transfer learning.

4.2. Evaluation metrics

This paper uses AP@0.5, AP@0.5:0.95 and Fps as evaluation metrics. FPS means the number of images that can be processed per second. It can fully evaluate the real-time performance of the model. Precision refers to the probability of correct detection in all detected objects, and Recall refers to the probability of correct identification in all positive samples. AP is the area enclosed by Precision and Recall as the two axes, which represents the average value of the detector in each Recall case. 0.5 and 0.95 represent the IoU thresholds for determining positive and negative samples, and AP@0.5:0.95 represents the average AP over different IoU thresholds (from 0.5 to 0.95, step size 0.05). The specific formula is as follows:

$$\begin{aligned} \text{Precision} &= \frac{TP}{TP+FP}, \\ \text{Recall} &= \frac{TP}{TP+FN}, \\ \text{AP} &= \frac{1}{101}\sum_{i=0}^{100} \text{Precision}\left(\text{Recall} = \frac{i}{100}\right). \end{aligned} \tag{2}$$

In the above formula, TP is the number of accurately predicted labels; FP is the false detection of no object, or the false detection of an existing object; FN is the missed detection of the object.

4.3. Model training

The network training process of YOLOv5s, YOLOv5n and Mob_YOLOv5 is shown in Fig. 7. After 250 epochs of training, the metrics tend to stabilize. The comparison of confidence

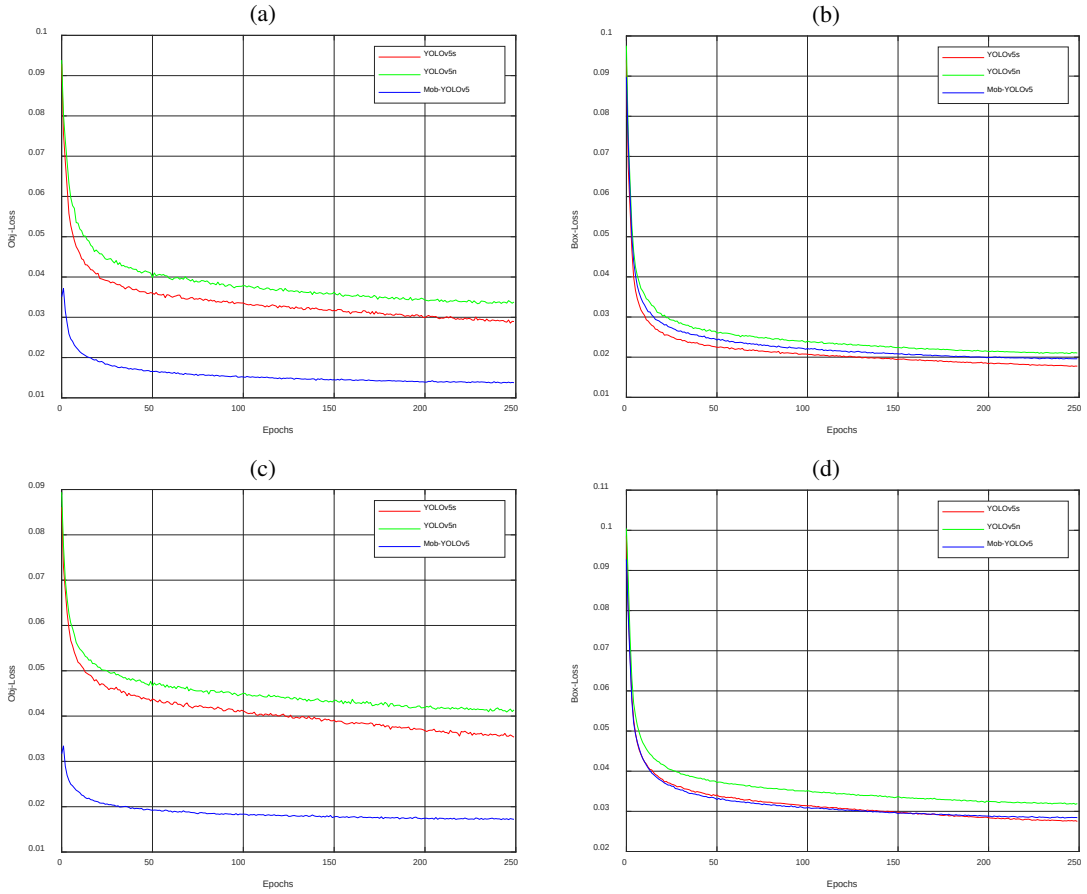

Fig. 7. Training process loss comparison

loss during training on UA-DETRAC and BDD100K datasets for three different methods is shown in Fig. 7a, c. It can be seen that Mob_YOLOv5 shows the smallest confidence loss in both datasets. This indicates that the predicted value of the improved model is closer to the true minimum bounding box of the object. The box loss comparison of three different methods during training on UA-DETRAC and BDD100K datasets is shown in Fig. 7b, d. It can be seen that compared with YOLOv5n, Mob_YOLOv5 shows smaller box loss, which is closer to the loss value of YOLOv5s. This shows that the predicted boxes of the newly proposed model have an accurate aspect ratio as compared to YOLOV5s. On the whole, the newly proposed method shows great improvement in the area of loss of confidence, and can complete the object detection task of this paper.

4.4. Experimental results

To verify the performance of Mob_YOLOv5 in detection tasks, comparative experiments are conducted on the UA-DETRAC dataset. Some detection examples of the improved algorithm are shown in Fig. 8. It can be seen that after the algorithm becomes lightweight, it performs well even in complex environments such as rainy days, nights, and overlapping objects, which shows the effectiveness of the designed method in the detection task.

Fig. 8. Detection examples of Mob_YOLOv5 in different scenarios

Table 2 shows the details of the parameters of the new network. It can be seen that by limiting the number of repetitions of the BottleneckMOB module, the parameters of the backbone network are greatly reduced. Therefore, the parameters of the backbone network using the BottleneckMOB module for feature extraction are controlled below 190, 000; the parameters of the neck part optimized by using depthwise separable convolution are about 130, 000. It can be seen that this paper constructs an ultra-lightweight object detection network.

To verify the lightness of the model, the total parameters of Mob_YOLOv5 and lightweight object detection methods such as YOLOv5n *et al*. are compared. It can be seen from Fig. 9 that the total parameters of Mob_YOLOv5 are 325, 766, while the number of parameters of YOLOv5s is 7, 063, 542 and YOLOv5n has nearly 2 million parameters. The number of parameters of the new network is about 1/6 of YOLOV5n. The general parameters of Mob_YOLOv5 are much smaller than those of the other models, which can prove that the proposed lightweight method can significantly reduce the parameter amount of the original network model.

Table 2
Detailed parameters of the new network

Module	Arguments	Params
Conv	[3, 16, 3, 2]	464
BottleneckMOB	[16, 8, 1, 1]	320
BottleneckMOB	[8, 16, 2, 6]	1 808
BottleneckMOB	[16, 16, 2, 6]	4 352
BottleneckMOB	[16, 32, 1, 6]	5 920
BottleneckMOB	[32, 48, 2, 6]	17 952
BottleneckMOB	[48, 80, 1, 6]	40 768
BottleneckMOB	[80, 160, 1, 6]	121 760
Conv	[160, 64, 1, 1]	10 368
Upsample	[None, 2, 'nearest']	0
Concat	[1]	0
C3	[96, 64, 1, False]	12 960
Conv	[64, 32, 1, 1]	2 112
Upsample	[None, 2, 'nearest']	0
Concat	[1]	0
C3	[48, 32, 1, False]	3, 408
Conv	[32, 32, 3, 2]	9 280
Concat	[1]	0
C3	[64, 64, 1, False]	10 912
Conv	[64, 64, 3, 2]	36 992
Concat	[1]	0
C3	[128, 128, 1, False]	42 304
yolo		1
SUM		325 766

Table 3 compares the metrics and model size of YOLOv5s, YOLOv5n and Mob_YOLOv5 in AP@50, AP@0.5:0.95, FPS.

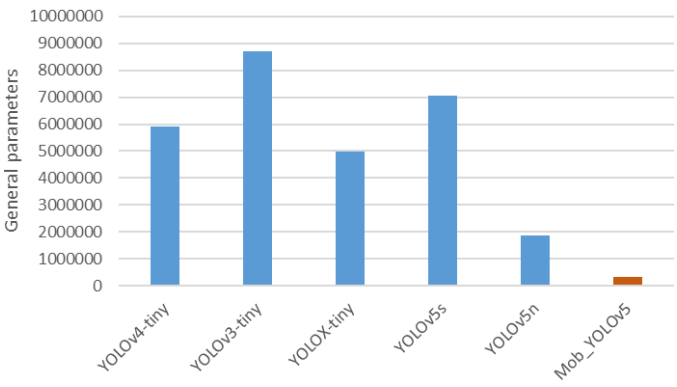

Fig. 9. Comparison of total parameters of different models

It can be seen that compared with YOLOv5s and YOLOv5n, the model size of Mob_YOLOv5 is reduced by 94% and 67%, respectively. This is beneficial to the deployment of the model on devices with limited memory. In addition, the response time of Mob_YOLOv5 is significantly shortened, and it can process 12 pictures per second on the CPU, which is 50% faster than that of YOLOv5s. Compared with YOLOv5n, FPS increased by 9%. The real-time performance of the model is further improved, which is suitable for real-time road monitoring scenarios. At the same time, on the test datasets of UA-DETRAC and BDD100K, average accuracy of the improved method shows a slight decrease. Compared with YOLOv5n, AP@0.5 did not drop by more than 1% in both datasets. Compared with YOLOv5s, AP@0.5:0.95 did not drop by more than 7.5% in both datasets. Although it has decreased, it still maintains good performance. In a word, the new network achieved a good tradeoff between precision and speed. Additionally, it can be seen from the evaluation results on two different datasets that the new network has strong generalization ability and can be applied to other object detection tasks.

Table 3
Comparison of experimental results

Method	Dataset	FPS (cpu)	Model Size/ (MB)	AP@0.5	AP@0.5: 0.95
YOLOv5s	UA-DETRAC	8	14	99.5	82.5
YOLOv5n	UA-DETRAC	11	2.7	98.5	79.4
Mob_YOLOv5	UA-DETRAC	***12***	***0.9***	98.1	75.7
YOLOv5s	BDD100k	8	14	88.5	62.4
YOLOv5n	BDD100k	11	2.7	84.7	59.5
Mob_YOLOv5	BDD100k	***12***	***0.9***	83.8	55.0

5. CONCLUSIONS

This paper analyzes the network structure parameters and calculation amount of YOLOv5, and optimizes the backbone and neck network of YOLOv5s. First, the lightweight MobileNetV2 network is used to replace the backbone feature extraction network of the original network, and the MobileNetV2 network is adaptively processed to reduce the convolution layer with high channel number and further simplify the network. Secondly, DSC is introduced. We reduce the model parameters by changing the convolution method of model neck network. Finally, the C3 module in the neck network is pruned to reduce memory usage and model complexity, and then an improved model, i.e. Mob_YOLOv5, is proposed. The model is trained and tested on the UA-DETRAC and BDD100K datasets. The experimental results show that:

1. Compared with the original YOLOv5s network, the model of the network is reduced by 94%, which greatly improves the application and deployment capability of the model in embedded devices;
2. The number of network layers and channels of the optimized network are reduced, which improves the inference speed of the model, while recognition speed of a single image is increased by about 50%, and real-time performance is further improved;
3. The solutions introduced greatly improve the real-time performance of the network and reduce the size of the model, but accuracy still maintains a high level, which fully meets the requirements of road monitoring scenarios.

REFERENCES

[1] L. Qiu *et al.*, "Deep learning-based algorithm for vehicle detection in intelligent transportation systems," *J. Supercomput.*, vol. 77, no. 10, pp 11083–11098, 2021, doi: 10.1007/s11227-021-03712-9.

[2] J. Zhao *et al.*, "Improved vision-based vehicle detection and classification by optimized YOLOv4," *IEEE Access*, vol. 10, pp. 8590–8603, 2022, doi: 10.1109/ACCESS.2022.3143365.

[3] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. 3rd Int. Conf. Learn. Represent. (ICLR)*, 2015, doi: 10.48550/ARXIV.1409.1556.

[4] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018, pp. 4510–4520, doi: 10.1109/CVPR.2018.00474.

[5] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, 2001, pp. I–I, doi: 10.1109/CVPR.2001.990517.

[6] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, 2005, vol. 1, pp. 886–893, doi: 10.1109/CVPR.2005.177.

[7] P.F. Felzenszwalb, R.B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 9, pp. 1627–1645, Sept. 2010, doi: 10.1109/TPAMI.2009.167.

[8] H. Fuhrmann, A. Boyko, M.H. Abdelpakey, and M.S. Shehata, "DETECTren: Vehicle object detection using self-supervised learning based on light-weight network for low-power devices," *2021 IEEE 7th World Forum on Internet of Things (WF-IoT)*, 2021, pp. 807–811, doi: 10.1109/WF-IoT51360.2021.9594927.

[9] L. Jiao *et al.*, "A survey of deep learning-based object detection," in *IEEE Access*, vol. 7, pp. 128837–128868, 2019, doi: 10.1109/ACCESS.2019.2939201.

[10] R. Girshick, J. Donahue, T. Darrell and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," *2014 IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 580–587, doi: 10.1109/CVPR.2014.81.

[11] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," in *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 9, pp. 1904–1916, 2015, doi: 10.1109/TPAMI.2015.2389824.

[12] R. Girshick, "Fast R-CNN," in *Proc. of 33rd IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 1440–1448, doi: 10.48550/arXiv.1504.08083.

[13] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, "You only look once: Unified, real-time object detection," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 779–788, doi: 10.1109/CVPR.2016.91.

[14] J. Redmon and A. Farhadi, "YOLO9000: Better, faster, stronger," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 6517–6525, doi: 10.1109/CVPR.2017.690.

[15] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," *arXiv*, 2018, doi: 10.48550/ARXIV.1804.02767.

[16] A. Bochkovskiy, Ch.-Y. Wang, and H.-Y.M. Liao, "YOLOv4: Optimal speed and accuracy of object detection," *arXiv*, 2020, doi: 10.48550/ARXIV.2004.10934.

[17] Ch.-Y. Wang, I-H. Yeh, and H.-Y.M. Liao, "You only learn one representation: Unified network for multiple tasks," *arXiv*, 2021, doi: 10.48550/ARXIV.2105.04206.

[18] Z. Ge, S. Liu, F. Wang, Z. Li, J. Sun, "YOLOX: Exceeding yolo series in 2021," *arXiv*, 2021, doi: 10.48550/ARXIV.2107.08430.

[19] Ch.-Y. Wang, A. Bochkovskiy and H.-Y.M. Liao, "Scaled-YOLOv4: Scaling cross stage partial network," in *Proc. 39th IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2021, pp. 13029–13038, doi: 10.48550/arXiv.2011.08036.

[20] N. Carion *et al.*, "End-to-end object detection with transformers," in *Proc. 17th European Conference on Computer Vision (ECCV)*, 2020, pp. 213–229, doi: 10.1007/978-3-030-58452-8_13.

[21] X. Zhu *et al.*, "Deformable DETR: Deformable transformers for end-to-end object detection," *arXiv*, 2020, doi: 10.48550/ARXIV.2010.04159.

[22] M. Zheng *et al.*, "End-to-end object detection with adaptive clustering transformer," *arXiv*, 2020, doi: 10.48550/ARXIV.2011.09315.

[23] F.N. Iandola *et al.*, "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and<0.5 MB model size," *arXiv*, 2016, doi: 10.48550/ARXIV.1602.07360.

[24] A. Gholami *et al.*, "SqueezeNext: Hardware-aware neural network design," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2018, pp. 1719–171909, doi: 10.1109/CVPRW.2018.00215.

[25] X. Zhang, X. Zhou, M. Lin, and J. Sun, "ShuffleNet: An extremely efficient convolutional neural network for mobile devices," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 6848–6856, doi: 10.1109/CVPR.2018.00716.

[26] N. Ma, X. Zhang, H.T. Zheng, and J. Sun, "Shufflenet v2: Practical guidelines for efficient CNN architecture design," in *Proc. 16th European Conference on Computer Vision (ECCV)*, 2018, pp. 122–138, doi: 10.1007/978-3-030-01264-9_8.

[27] A.G. Howard *et al.*, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv*, 2017, doi: 10.48550/arXiv.1704.04861.

[28] M. Sandler *et al.*, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proc. 36th IEEE conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2018, pp. 4510–4520, doi: 10.1109/CVPR.2018.00474.

[29] A. Howard *et al.*, "Searching for mobilenetv3," in *Proc. 17th IEEE Int. Conf. Comput. Vis. (ICCV)*, 2019, pp. 1314–1324, doi: 10.48550/arXiv.1905.02244.

[30] K. Han, Y. Wang, Q. Tian, J. Guo, C. Xu, and C. Xu, "GhostNet: More Features From Cheap Operations," *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 1577–1586, doi: 10.1109/CVPR42600.2020.00165.

[31] M. Tan and Q.V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *Proc. 36th International Conference on Machine Learning, 2019 (ICML)*, 2019, pp. 6105–6114, doi: 10.48550/arXiv.1905.11946.

[32] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778, doi: 10.1109/CVPR.2016.90.

[33] F. Chollet, "Xception: deep learning with depthwise separable convolutions," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 1800–1807, doi: 10.1109/CVPR.2017.195.

[34] L. Wen *et al.*, "UA-DETRAC: A new benchmark and protocol for multi-object detection and tracking," *Comput. Vision Image Understanding.*, vol. 193, p. 102907, 2020, doi: 10.1016/j.cviu.2020.102907.

[35] F. Yu *et al.*, "BDD100K: A Diverse Driving Dataset for Heterogeneous Multitask Learning," *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 2633–2642, doi: 10.1109/CVPR42600.2020.00271.