Journal of Network Intelligence
Taiwan Ubiquitous Information
 ISSN 2414-8105 (Online)
Volume **7**, Number **1**, February **2022**

A lightweight fast object detection method

Yan-Fei Jia

School of Electrical and Information Engineering
Beihua University
Jilin 132013, China
jia_yanfei@163.com

Guang-Da Chen*

School of Electrical and Information Engineering
Beihua University
Jilin 132013, China
Corresponding Author: 694864603@qq.com

Zi-Cong Jiang

Graduate school of Information Science and Electrical Engineering
Kyushu University
Fukuoka 819-0395, Japan
jiangzicong1234@gmail.com

Miao Yang

Engineering Training Center
Beihua University
Jilin 132013, China
215998452@qq.com

Li-Yun Xing

School of Electrical and Information Engineering
Beihua University
Jilin 132013, China
373158565@qq.com

Ying Cui

Zhuhai Power Supply Bureau
Guangdong Electric Power Corporation
Zhuhai 519000, China
cuiying794758706@126.com

Received May 2021; revised July 2021

Abstract. *To speed up the object detection on devices that have limited computing power, a fast object detection method is proposed on the basis of "You only look once (version 4)-tiny". It firstly uses two ResBlock-D modules in ResNet-D network instead of two CSPDBlock modules in "You only look once (version 4)-tiny", which reduces the computation complexity. Secondly, it designs an auxiliary residual network block to extract more feature information of object to reduce detection error. In the design of auxiliary network, two consecutive* 3×3 *convolutions are used to obtain* 5×5 *receptive fields to extract global features, and channel attention and spatial attention are also used to extract more effective information. In the end, it merges the auxiliary network and backbone network to construct the whole network structure of proposed method. Simulation results show that the proposed method has faster object detection than "You only look once (version 4)-tiny" and "You only look once (version 3)-tiny", and almost the same mean value of average precision as the "You only look once (version 4)-tiny". It is more suitable for real-time object detection.*

Keywords: Artificial intelligence; Object detection;YOLOv4-tiny; Real-time

1. **Introduction.** With development of artificial intelligence, deep leaning has been received increasing attention and many deep learning methods have bee proposed [1–3]. They have been applied in autonomous driving,stock price prediction and other many fields [4–7]. In this paper, we mainly focus on their application in object detection. Object detection method based on deep learning mainly includes two types: region proposal-based two-stage method and regression-based one-stage method. The typical two-stage methods include region-based convolution neural network (R-CNN) method. Faster R-CNN [8] method, Mask R-CNN [9], region-based fully convolutional networks (R-FCN) method [10], light head R-CNN method and other improve methods based on convolution neural network [11, 12]. Although two-stage method has higher accuracy than one-stage method, the one-stage method has faster detection speed than two-stage method. The one-stage method is more suitable for application in some conditions that require higher real-time.

The normal detection method requires devices that have larger memory and computing power. They cannot be deployed on embedded devices that have limited computing power. This affects the application of object detection methods. The lightweight object detection method has lower complexity than normal detection method, and they can be developed on embedded devices that have limited computing power, such as unmanned aerial vehicle and robot. Therefore, it is important to research the lightweight object detection method to improve its accuracy or detection speed. The You Only Look Once (YOLO) method [13] proposed by Redmon is the first regression-based one stage method. Redmon, et al. also proposed the You Only Look Once version 2 (YOLOv2) [14] based on YOLO by deleting fully connected layer and the last pooling layer, using anchor boxes to predict bounding boxes and designing a new basic network named DarkNet-19. The You Only Look Once version 3 (YOLOv3) [15] is the last version of YOLO method proposed by Redmon, et al. It introduces feature pyramid network, a batter basic network named darknet-53 and binary cross-entropy loss to improve the detection accuracy and the ability of detecting smaller object. Due to the type of information fusion employed by YOLOv3 does not make full use of low-level information, a weakness which restricts its potential application in industry. Therefore, Du et al. [16] have proposed the YOLO-Inception method, one which uses the inception structure with diversified receptive fields, which in turn can provide rich semantic information and improve the performance of small object detection. Tia et al. [17] has proposed the YOLOv3-dense method. It uses the dense

net method to process feature layers with low resolution, which effectively enhances feature propagation, promotes feature reuse, and improves network performance. Two years later, after the authors of YOLOv3 declared to give up updating it, Bochkovskiy et al. [18] proposed the YOLOv4 method that has been accepted by the authors of YOLOv3. It used CSPDarknet53 backbone, spatial pyramid pooling module, PANet path-aggregation neck and YOLO3 (anchor based) head as the architecture of YOLOv4. Besides, it also introduced a new method of data augmentation mosaic and self-adversarial training, applied genetic algorithms to select optimal hyper-parameters and modified some existing method to make the proposed method suitable for efficient training and detection.

YOLO serial methods and their improved methods have complex network structure and a larger number of network parameters. They require powerful GPU(graphic processing unit) computing power to realize the real-time object detection. However, they have limited computing power and limited memory, and require real-time object detection for some mobile devices and embedded devices (autonomous driving devices, augmented reality devices and other smart device) in real-world applications [19]. For example, such as real-time inference on smart phones and embedded video surveillance, the available computing resources are limited to a combination of low-power embedded GPUs or even just embedded CPUS with limited memory. Therefore, it is a big challenge to realize the real-time object detection on embedded devices and mobile devices. To solve the problem, lightweight object detection methods are proposed by many researchers. The lightweight methods have comparatively simpler network structure and fewer parameters. Therefore, they require lower computing resources and memory, and have faster detection speed. They are more suitable for deploying on mobile devices and embedded devices. Although they have lower detection accuracy, the accuracy can meet the actual demands. Lightweight object detection methods based on deep learning have been applied in many fields, including vehicle detection [20], pedestrian detection [21], bus passenger object detection [22], agricultural detection and human abnormal behavior detection [23, 24], etc.

A number of lightweight object detection methods have already been proposed to improve detection speed with the limitation of hardware platforms and meanwhile to meet the demand of high performance. Such MobileNet series [25–27], Squeezenet series [28,29], ShuffleNet series [30,31], lightweight YOLO series [32–34]. The MobileNetv1 method [25] constructs lightweight deep neural networks by using depth wise separable convolution instead of the traditional convolution to reduce parameters. Based on MobileNetv1, the MobileNetv2 bulids inverted residual module by adding the point-wise convolution layer in front of depthwise separable convolution to improve the ability of extracting features [26]. MobileNetv3 redesigns some computationally-expensive layers and introduces the hard swish nonlinearity to improve detection speed [27]. Squeezenet method design the new network architecture based on CNN by replacing 3×3 convolutions with 1×1 convolutions, using squeeze layers to decrease the number input channels to 3×3 convolutions and down sampling late in the network to improve detection speed. The SqueezeNext method is proposed based on squeezenet. Its neural network architecture is able to achieve AlexNet top-5 performance [29]. The Mobile Netseries, squeezenet series and shuffleNet series are directly designed to realize lightweight network. The lightweight YOLO series methods are designed based on complete YOLO. They are realized by suppressing the network of complete YOLO method. YOLOv2-tiny is one of lightweight YOLO series methods [33]. The complete YOLOv2 uses the Darknet19 as backbone network, which contains 19 convolution layers and 6 pooling layers. They YOLOv2-tiny method deletes convolution layers in Darknet19 network to 9 layers to reduce the network complexity. YOLOv3-tiny is proposed by compressing the network model of YOLOv3 [15]. It uses

seven layer convolution networks and six max pooling layers instead of the ResBlock structure in DarkNet53 network [35]. It also reduces the output branch from the three scale predictions($52 \times 52, 26 \times 26$ and 13×13) to two scale predictions (26×26 and 13×13).

YOLOv4-tiny [36] is also one of lightweight YOLO series methods, and also realized based on YOLOv4 [18]. It uses CSPDarknet53-tiny backbone network instead CSPDarknet53 backbone network of YOLOv4. The spatial pyramid pooling and path aggregation network are also be instead by feature pyramid networks (FPN) to reduce the detection time. Besides, it also uses two scale predictions (26×26 and 13×13) instead of three scale predictions. Compared with YOLOv3-tiny, the YOLOv4-tiny uses the CSPBlock network to extract feature without using the conditional con convolution networks, and introduces the complete intersection over union to select bounding boxes.The Yolov4-tiny method uses the CSPB lock module as residual module. Although it improves the accuracy, it also increases network complexity. To reduce the complexity of network, we propose a fast object detection method.

The main contributions of this paper are as follows:

1) To reduce network complexity, we use the ResBlock-D module instead of two CSPBlock modules to modify the backbone of Yolov4-tiny. The floating point operations of ResBlock-D module is about 6×10^7, and CSPBlock module is about 7×10^8. This means that the ResBlock-D module has lower complexity than CSPBlock module, and using ResBlock-D module can reduce the complexity of backbone of YOLOv4-tiny.

2) To reduce detection error, we design an auxiliary residual network block and add it into backbone network of YOLOv4-tiny. The improved backbone realized the fusion of deep and shallow features. This can extract more feature information of object.

The above two points realize the balance between detection speed and accuracy. Compared with YOLOv3-tiny and YOLOv4-tiny, the proposed method has faster detection speed and almost the same detection error with YOLOv4-tiny. Therefore, it is more suitable for developing on embedded devices.

2. **Proposed method.**

The Yolov4-tiny method uses the CSPBlock module as residual module. It improves the accuracy, but it also increases network complexity. This reduces the speed of object detection. To improve the speed of object detection with slight impacting accuracy, an improved Yolov4-tiny is proposed.

To speed up the object detection, we use the ResBlock-D module instead of two CSPBlock modules in Yolov4-tiny. The CSPBlock and ResBlock-D modules are shown in Figure 1. The ResBlock-D module directly uses two paths network to deal with the input feature map. It also has two paths network. The path A network contain three layers that are 1×1 convolutions, 3×3 convolutions with 2 strides and 1×1 convolutions. The path B network contains two layers that are 2×2 average poolings with 2 strides and 1×1 convolutions. Compared with CSPBlock module, the ResBlock-D module deletes the first layer with 3×3 convolutions in CSPBlock, and uses the 1×1 convolutions layer to instead the 3×3 convolutions layer in CSPBlock module in path A to further more reduce computation. Although it increases two layers in path B, the increased computation is smaller than reduced computation.

To analysis the computation of two modules, the floating point operations (FLOPs) are used to compute computation complexity. It can be expressed as follow:

$$FLOPs = \sum_{l=1}^{D} M_l^2 \cdot K_l^2 \cdot C_{l-1} \cdot C_l \tag{1}$$

where D is the sum of all convolution layers, M_l^2 is output feature map size in convolution layer, K_l^2 is the number of kernel size, C_{l-1} and C_l are the number of input channel and output channel, respectively. We suppose the size of input image is 104×104 and the number of channel is 64. Based on (1), the FLOPs of CSPBlock used in Yolov4-tiny is:

$$\begin{aligned} FLOPs=&104^2 \times 3^2 \times 64^2 + 104^2 \times 3^2 \times 64 \times 32 + 104^2 \times 3^2 \times 32^2 + 104^2 \times 1^2 \times 64^2 \\ =&7.421 \times 10^8 \end{aligned} \tag{2}$$

The FLOPs of ResBlock-D used in our proposed method is

$$\begin{aligned} FLOPs=&104^2 \times 1^2 \times 64 \times 32 + 52^2 \times 3^2 \times 32^2 + 52^2 \times 1^2 \times 32 \times 64 + \\ &64 \times 52^2 \times 2^2 + 52^2 \times 1^2 \times 64^2 \\ =&6.438 \times 10^7 \end{aligned} \tag{3}$$

Based on (2) and (3), the computation complexity rate of CSPBlock and ResBlock-D is about 10:1. It means that the computation complexity of ResBlock-D is smaller than CSPBlock.

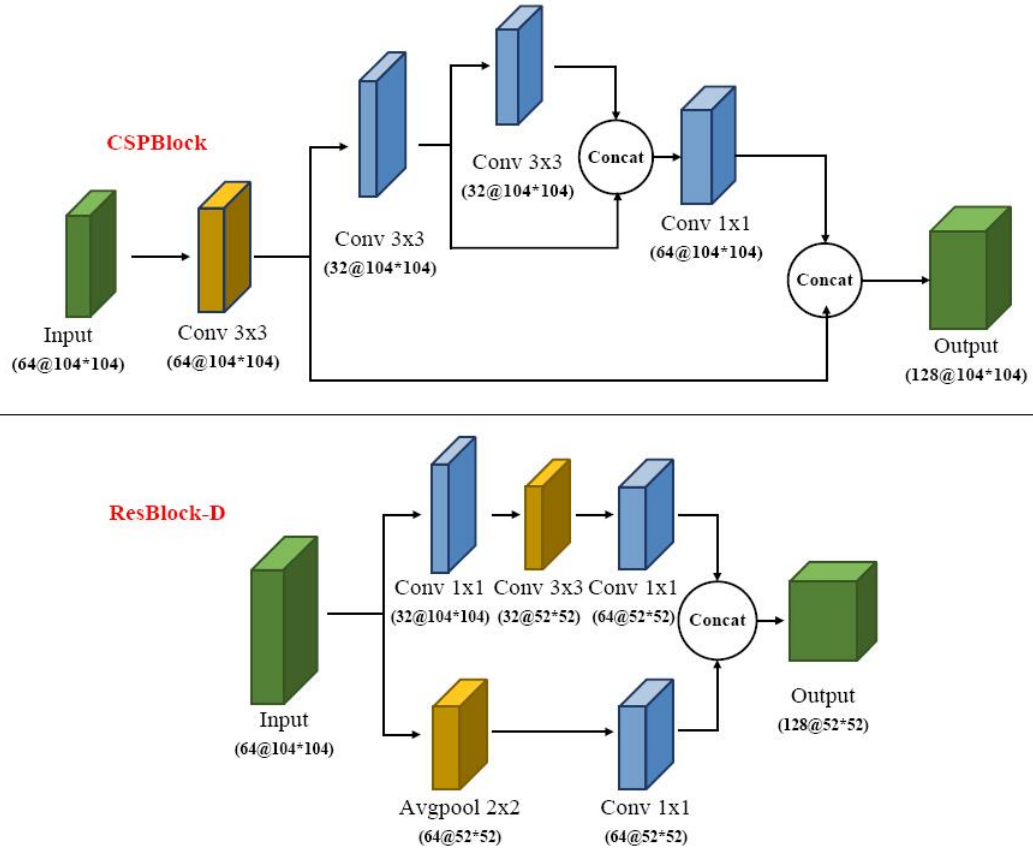

FIGURE 1. CSPBlock and ResBlock-D modules

Although we use the ResBlock-D module to replace CSPBlock module to improve object detection speed, it reduces the accuracy of object detection. To keep the balance between accuracy and speed, we design two same residual network blocks as auxiliary network block and add them into the ResBlock-D module to improve accuracy. In original backbone network, the residual network module uses 3×3 convolution kernels to extract feature.

The size of its receptive field is also 3×3. Although the smaller receptive field can extract more local information, it losses the global information that affects the accuracy of object detection. To extract more global feature, we use two consecutive same 3×3 convolutions to obtain 5×5 receptive fields in the auxiliary residual network block to extract more global information. Besides, we also introduce the channel attention module and spatial attention module into our designed auxiliary network module to extract more effective information. The channel attention module focuses on that what is meaningful given an input image. The spatial attention module focuses on where is an informative part, which is complementary to the channel attention.The attention mechanism can focus on processing and transmitting the effective features, and channel suppressing the invalid features. Our designed auxiliary network block is shown in Figure 2. It uses two 3×3 convolutions network to extract the global features, and channel attention and spatial attention to extract more effective information. The concatenate operation is used to combine the output feature obtained from the first convolution network and the output feature obtained from the spatial attention. The combined feature is used as the output feature of designed auxiliary network.

The output of channel attention is:

$$F' = M_c(F) \otimes F \tag{4}$$

Where $F \in R^{C \times H \times W}$ denotes input feature map, $\otimes$ expresses element-wise multiplication, $M_c(F)$ is channel attention operation that is expressed as:

$$M_c(F) = \sigma\left(MLP\left(AvgPool\left(F\right)\right) + MLP\left(MaxPool\left(F\right)\right)\right) \tag{5}$$

Where $AvgPool()$ and $MaxPool()$ denote average-pooling operation and max-Pooling operation, respectively. $MLP()$ denotes Multi-Layer perceptron network, $\sigma()$ is the sigmoid function. The output of spatial attention is:

$$F'' = M_s(F') \otimes F' \tag{6}$$

where the $M_s(F')$ is expressed as:

$$M_s(F') = \sigma(f^{7\times7}[MaxPool(F'); AvgPool(F')]) \tag{7}$$

where $f^{7\times7}$is the convolution operation with the kernel size of 7×7, $[\cdot;\cdot]$ denotes concatenate operation.

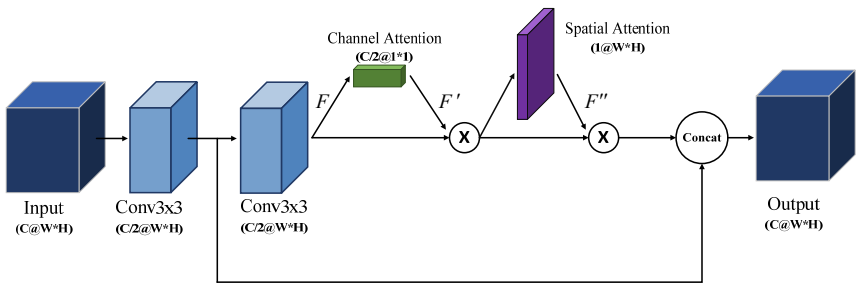

FIGURE 2. Auxiliary residual network block

In the end, the final output feature of auxiliary network will be combined with the output feature of residual network in backbone network. The combined feature will be used as the input feature of next residual network in backbone network. The auxiliary network transmits the extracted global information to backbone network. The backbone network combines the global information obtained by larger receptive field and local information obtained by smaller receptive field to obtain more object information. This makes the

improved backbone network can extract the global and local feature of detection object, and further more improve the accuracy of detection. The proposed backbone network is shown in Figure 3. The output feature of designed residual network block is fused with shallow feature of backbone network by element-wise summation operation, and the fused information is used as the input of next layer in the backbone network. The fused process can be expressed as:

$$x^l = F^l(x^{l-1}) + x^l_{assist}(l = 3, 4) \tag{8}$$

where l is the index of network layer, x^l is the output of $l-1$th layer and also input of lth layer, x^l_{assist} is the output of our designed residual network, $F^l()$ denotes the relationship between the input and output in the l th layer network. This realizes the convergence between deep network and shallow network. It makes the network learn more information to improve detection accuracy and avoids the large increase of calculation. Based on the above introduction, the whole network structure of our proposed YOLO v4-tiny is shown in Fig.4. The mainly difference between our proposed method and YOLOv4-tiny in network structure are that we use two ResBlock-D modules to replace two CSPBlock modules in the original CSPNet53-tiny network. Besides, we design auxiliary network block by using two 3×3 convolutions, channel attention, spatial attention and concatenate operation to extract global feature. In the end, we merge our designed auxiliary network into the backbone network to build new backbone network. Our proposed network is marked in red in the Fig.4.

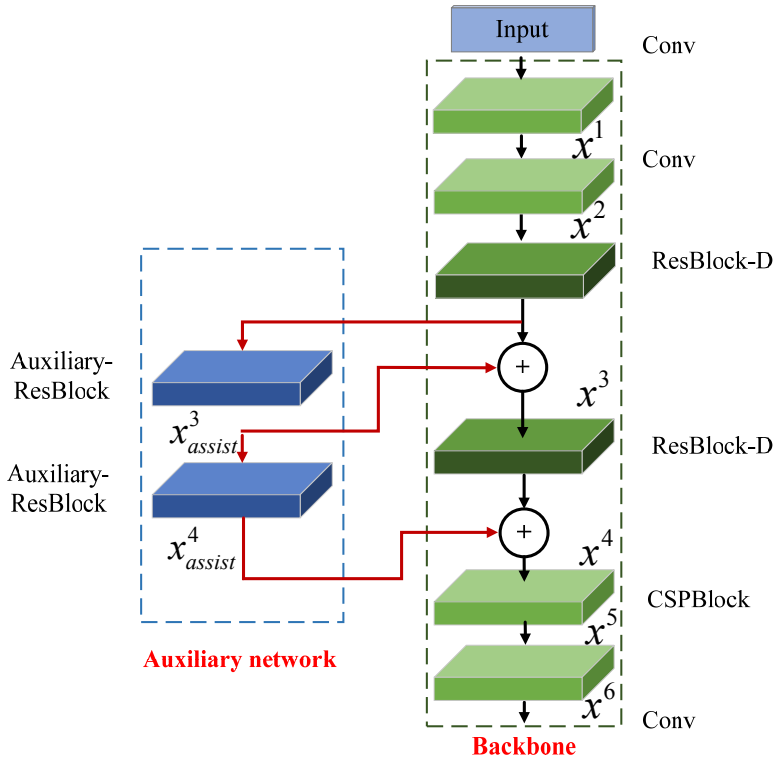

FIGURE 3. Proposed backbone network

3. **Experimental results and analysis.** In this paper, we use the MS COCO (Microsoft Common Objects in Context) dataset as train and test dataset. The MS COCO is an authoritative and significant benchmark used in the field of object detection and

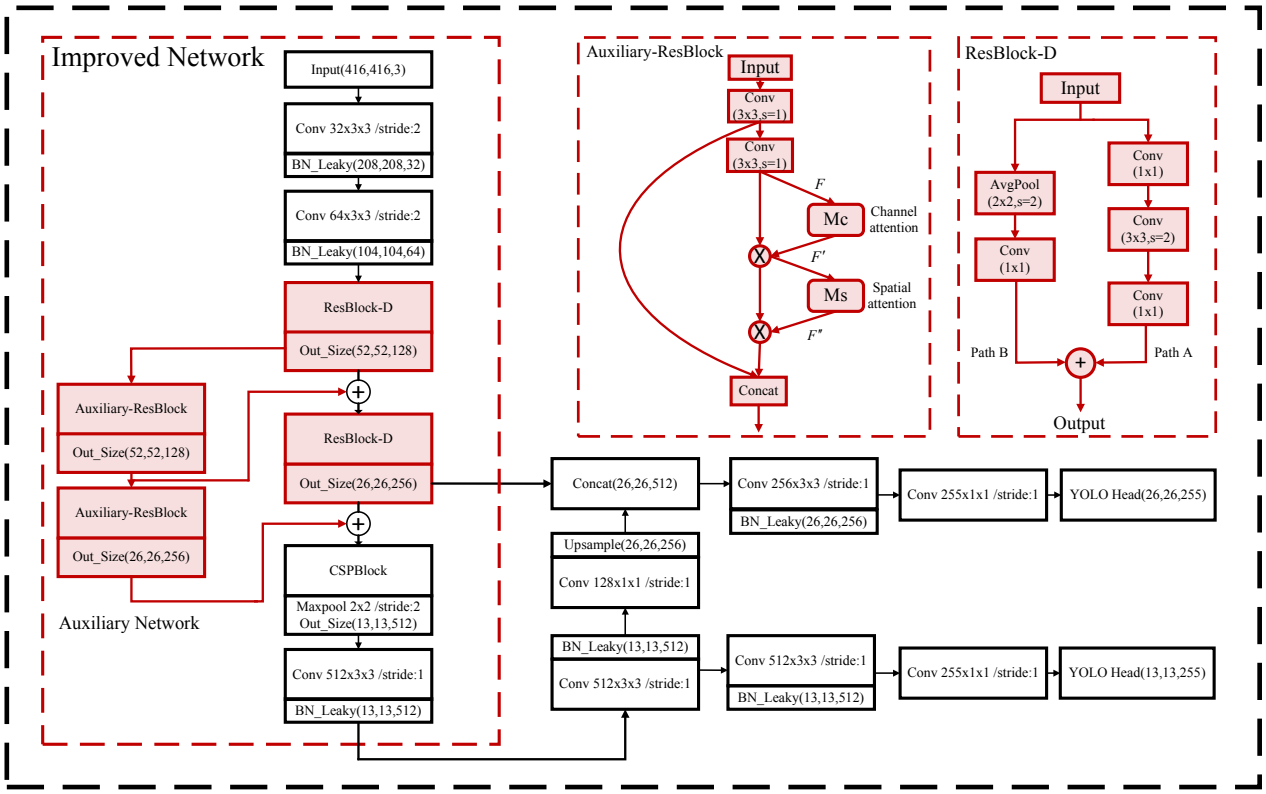

FIGURE 4. Network structure of our proposed method

recognition. It is widely used in many detection methods. It contains 117,264 training images and 5000 testing images with 80 classes. The experiments environment configured in this paper is as follows: The operating system is Ubuntu 18.04. The CPU is Intel Xeon E5-2678 v3 with 2.5 GHZ main frequency. The GPU is NVIDIA GeForce GTX 1080Ti. In order to make full use of the GPU to accelerate the network training, the CUDA 10.1 and its matching CUDNN are installed in the system. The deep learning framework is PyTorch. We use the same parameters for different methods. The batch size, epoch, learning rate, momentum and decay are 16, 273, 0.001, 0.973 and 0.0005 for all methods, respectively.

The mAP (mean value of average precision), FPS (Frames per second) and GPU utilization are used to quantitatively evaluate the performance of different methods. The mAP is the mean value of average precision for the detection of all classes. FPS denotes the number of images that can be detected successfully in one second. GPU utilization denotes used GPU memory in testing the different detection methods.

We firstly compare our proposed method with YOLOv3, YOLOv4, YOLOv3-tiny, YOLOv4-tiny to test their performance in mAP and FPS. The results are shown in Table 1. Although YOLOv4 and YOLOv3 methods have the larger mAP than other methods, they also have the smaller FPS than other methods. YOLOv4 and YOLOv4 methods have complex network structure and many parameters. This makes them have better performance in mAP and worse performance in FPS. They demand the platform to be very powerful. This limits to deploy them on the mobile and embedded devices that have limited computing power. YOLOv3-tiny, YOLOv4-tiny and our proposed method belong to lightweight deep learning method. They have relatively simple network structure and

few parameters. Therefore, they have better performance in FPS and worse performance in mAP, and more suitable for deploying on the mobile and embedded devices.

Due to the YOLOv3-tiny, YOLOv4-tiny and our proposed methods belong to lightweight deep learning method. The YOLOv3 and YOLOv4 methods do not belong to lightweight deep learning method, so we only compare our proposed method with YOLOv3-tiny and YOLOv4-tiny in the following analysis. Compared our proposed method with YOLOv3-tiny and YOLOv4-tiny, our proposed method has the largest FPS, and YOLOv4-tiny has the largest mAP followed by our proposed method. The mAP of our proposed method is 38% and YOLOv4-tiny method is 38.1%. The relative mAP only reduces by 0.1%. The FPS of our proposed method is 294 and YOLOv4-tiny method is 270. The relative FPS increases by 8.9%. Although the mAP of our proposed method is reduction compared with YOLOv4-tiny, the reduction is much smaller than the increase of FPS, and almost can be ignored.

Table 1. Comparison of different methods in FPS and mAP

Method	FPS	mAP(%)
YOLOv3	49	52.5
YOLOv4	41	64.9
YOLOv3-tiniy	277	30.5
YOLOv4-tiniy	270	38.1
Proposed method	294	38.0

The Table 2 shows the GUP utilization when different methods are used to detect object. GPU utilizations are 1123MB, 1055MB and 1003MB for YOLOv3-tiny, YOLOv4-tiny and proposed method, respectively. The proposed method has the smallest GPU utilization. Based on above analysis, our proposed method has faster detection speed and smaller GPU utilization than others, and is more suitable for developing on the mobile and embedded devices.

Table 2. Comparison of different methods in GPU utilization(MB)

Method	GPU utilization(MB)
YOLOv3-tiniy	1123
YOLOv4-tiniy	1055
Proposed method	1003

To test the performance of proposed method on CPU device, we simulate the different methods on CPU. The CPU model is Intel Xeon E5-2678 v3 with 2.5 GHZ main frequency. We also use the same MS COCO dataset as testset. The size of input images is 416 x 416 and the batch size is 16. The number of FPS(Frames Per Second)for YOLOv3-tiny, YOLOv4-tiny and our proposed method are 32, 25 and 37, respectively. Compared with YOLOv3-tiny and YOLOv4-tiny, the FPS of our proposed method increases by 15% and 48%, respectively. This means that our proposed method has faster detection speed than others on CPU. The Raspberry Pi is one of special embedded devices. It has been widely used in robot [37, 38]. To test the performance of proposed method on embedded devices, we simulate the different methods on Raspberry Pi and telephone mobile devices, respectively. The Raspberry Pi model is Raspberry Pi 3B with BC219M2835 processor. We also use the same MS COCO dataset as testset. The size of input images is 416×416 and the batch size is 16. Due to the limited of storage space in Raspberry Pi device,

we randomly select 40 images from MS COCO dataset to test different methods on the Raspberry Pi. Fig.5 is the Raspberry Pi device that is testing different methods. The time used to recognize the 40 images for YOLOv3-tiny, YOLOv4-tiny and our proposed method are 219s, 211s and 128s, respectively. We also transform the consumed time to FPS(Frame Per second) by dividing the time by the number of recognized images. The number of FPS for YOLOv3-tiny, YOLOv4-tiny and our proposed method are 0.18, 0.19 and 0.31, respectively. Compared with YOLOv3-tiny and YOLOv4-tiny, the FPS of our proposed method increases by 72% and 63%, respectively. This also shows that our proposed method has faster detection speed than others on Raspberry Pi.

We also simulate the different methods on telephone mobile. The telephone model is Redmi Note4X with Android 7.0. Its CPU is Snapdragon 625. We also use the same MS COCO dataset as testset. The size of input images is 416×416 , and the batch size is 16. The average detection times for one picture and corresponding FPS for different methods are shown in Fig.6. The average detection times for YOLOv3-tiny, YOLOv4-tiny and our proposed method are 675.1ms, 618.1ms and 505.5ms, respectively. The FPS for YOLOv3-tiny, YOLOv4-tiny and our proposed method are 1.48, 1.62 and 1.96, respectively. Compared with YOLOv3-tiny and YOLOv4-tiny, the FPS of our proposed method increases by 32%and 21%, respectively. This also shows that our proposed method has faster detection speed than others on telephone mobile.

Based on the above analysis, our proposed method still has faster detection speed than YOLOv3-tiny and YOLOv4-tiny methods when we test the three methods on the CPU, Raspberry Pi and mobile telephone, respectively.

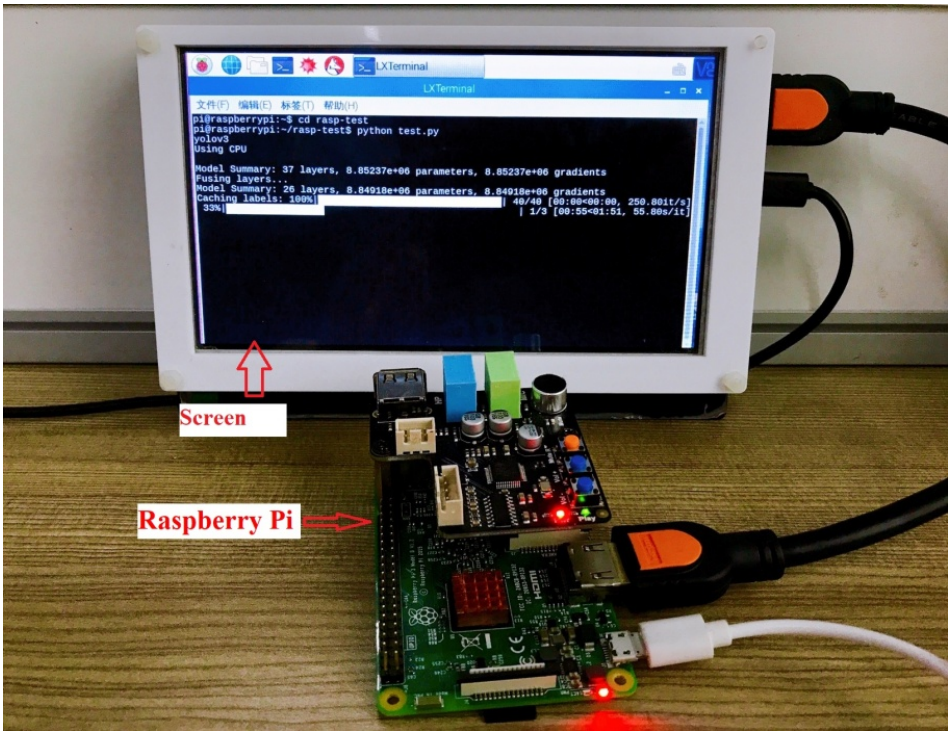

FIGURE 5. Raspberry Pi device

4. **Conclusions.** This paper proposes a fast object detection method that is suitable for devices that have limited computing power. There are two main contributions: 1) To reduce the consuming time of object detection, we use two same ResBlock-D modules to

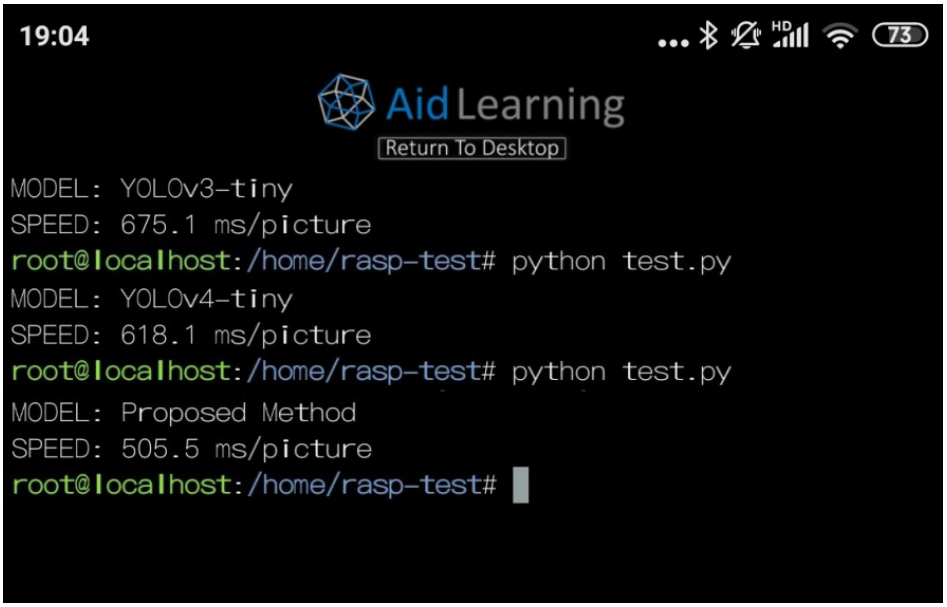

FIGURE 6. Simulation results obtained from telephone mobile

replace two CSPBlock modules in YOLOv4-tiny network to simple the network structure. 2) To balance the object detection time and accuracy, we design auxiliary network block by using two convolutions network, channel attention, spatial attention and concatenate operation to extract global feature. In the end, we merge our designed auxiliary network into the backbone network to build new backbone network. This realizes the convergence between deep network and shallow network. It makes the improved backbone network can extract the global and local feature of detection object, and further more improve the accuracy of detection without increasing large calculation. Compared with YOLOv3-tiny and YOLOv4-tiny, the proposed method has faster object detection speed and almost the same mean value of average precision with YOLOv4-tiny.

Acknowledgment. This work was supported by the National Natural Science Foundation of China (61901007), Scientific and Technological Developing Scheme of JiLin Province YDZJ202101ZYTS172), Research Foundation of Education Bureau of Jilin Province (JJKH20210042KJ,JJKH20210095KJ), Youth Science and technology innovation team of Beihua University(202016003)and Doctoral Scientific Research Foundation of Beihua University (20171424).

REFERENCES

[1] J.M.T. Wu, M. H. Tsai, Y. Z. Huang, S.K Hafizul Islam, M.M Hassan, A. Alelaiwi, G. Fortino, Applying an ensemble convolutional neural network with Savitzky-Golay filter to construct a phonocardiogram prediction model, *Applied Soft Computing*, vol. 78, pp. 29-40, 2019.

[2] K. Wang, C.M. Chen and M.S. Hossain, G. Muhammad, S. Kumar, S. Kumari, Transfer reinforcement learning-based road object detection in next generation IoT domain, *Computer Networks*, vol. 139, 108078, 2021.

[3] J.M.T. Wu , M. H. Tsai, S. H. Xiao ,Y.P. Liaw, A deep neural network electrocardiogram analysis framework for left ventricular hypertrophy prediction, *Journal of Ambient Intelligence and Humanized Computing*, 2020, https://doi.org/10.1007/s12652-020-01826-1.

[4] K.K. Tseng, J. Lin, C.M. Chen, M.M. Hassan, A fast instance segmentation with one-stage multitask deep neural network for autonomous driving, *Computers and Electrical Engineering*, vol. 93, 107194, 2021.

[5] J.M.T. Wu, Z. Li, N. Herencsar, B. Vo, C.W. Lin, A graph-based CNN-LSTM stock price prediction algorithm with leading indicators Multimedia Systems, *Multimedia Systems*, 2021, https://doi.org/10.1007/s00530-021-00758-w.

[6] J.M.T. Wu, Z. Li, N. Herencsar, B. Vo, C. W. Lin, A graph-based convolutional neural network stock price prediction with leading indicators, *Software: Practice and Experience*, vol. 51, no. 3, pp. 628-644, 2021.

[7] 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*, pp. 580-587, 2014.

[8] S. Ren, K. He, R. Girshick, and J. Sun, Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137-1149, 2017.

[9] Q. Chen, X. Gan, W. Huang, J. Feng, and H. Shim, Road damage detection and classification using mask r-cnn with densenet backbone, *Computers, Materials and Continua*, vol. 65, no. 3, pp. 2201-2215, 2020.

[10] J. Dai, Y. Li, K. He, and J. Su, R-FCN: object detection via region-based fully convolutional networks, *30th International Conference on Neural Information Processing Systems*, pp. 379-387, 2016.

[11] S. H. Park, H. S. Yoon, and K. R. Park, Faster R-CNN and Geometric Transformation-Based Detection of Driver Eyes Using Multiple Near-Infrared Camera Sensors, *Sensors*, vol. 19, no. 1, 197, 2019

[12] P. Vara, B.D. Souzakevin, and K. Bhargavavijay, A Downscaled Faster-RCNN Framework for Signal Detection and Time-Frequency Localization in Wideband RF Systems, *IEEE Transactions on Wireless Communications*, vol. 19, no. 7, pp. 4847-4862, 2020.

[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*, pp. 779-788, 2016.

[14] J. Redmon, and A. Farhadi, YOLO9000: Better, Faster, Stronger, *2017 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6517-6525, 2017.

[15] J. Redmon, and A. Farhadi, YOLOv3: An Incremental Improvement, *2017 arXiv:1804.02767*.

[16] P. Du, X. Qu, T. Wei, C. Peng, X. Zhong, and C. Chen, Research on Small Size Object Detection in Complex Background, *2018 Chinese Automation Congress (CAC)*, pp. 4216-4220, 2018.

[17] Y. Tian, G. Yang, and Z. Wang, Apple detection during different growth stages in orchards using the improved YOLO-V3 model, *Computer Electron Agriculture*, vol. 157, pp. 417-426, 2019.

[18] A. Bochkovskiy, C.Y. Wang, and H.Y.M. Liao, YOLOv4: Optimal Speed and Accuracy of Object Detection, *arXiv2020, arXiv:2004.10934*, 2020.

[19] Z.F. Xu, R.S. Jia, and H.M. Sun, Light-YOLOv3: fast method for detecting green mangoes in complex scenes using picking robots, *Applied Intelligence*, vol. 50, pp. 4670-4687, 2020.

[20] L. Chen, Q. Ding, Q. Zou, Z. Chen, and L. Li, DenseLightNet: A Light-Weight Vehicle Detection Network for Autonomous Driving, *IEEE Transactions on Industrial Electronics*, vol. 67, no. 12, pp. 10600-10609, 2020.

[21] Z. Yi, S. Yongliang, and Z. Jun, An improved tiny-yolov3 pedestrian detection algorithm, *Optik*, vol. 183, pp. 17-23, 2019.

[22] S. Zhang, Y. Wu , and C. Men, Tiny YOLO Optimization Oriented Bus Passenger Object Detection, *Chinese Journal of Electronics*, vol. 29, no. 1, pp. 132-138, 2020.

[23] A. Koirala, K. B. Walsh, Z. Wang, Deep learning for real-time fruit detection and orchard fruit load estimation: benchmarking of Mango YOLO, *Precision Agriculture*, vol. 20, pp. 1107-1135, 2018.

[24] J. Hongxia, Z. Xianlin, and L. Hongguang, Human abnormal behavior detection method based on T-TINY-YOLO, *5th International Conference on Multimedia and Image Processing*, pp. 1-5, 2020.

[25] A.G. Howard, M. Zhu, and B. Chen, MobileNets: efficient Convolutional Neural Networks for Mobile Vision Applications, *arXiv2017,arXiv:1704.04861*, 2017.

[26] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, MobileNetV2: Inverted Residuals and Linear Bottlenecks, *2018 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4510-4520, 2018.

[27] A.G. Howard, M. Zhu, and B. Chen, Searching for MobileNetV3, *2019 IEEE/CVF International Conference on Computer Vision*, pp. 1314-1324, 2019.

[28] F.N Iandola, S. Han, and M.W Moskewicz, Squeezenet: Alexnet-level accuracy with 50x fewer parameters and $<$ 0.5 MB model size, *arXiv preprint arXiv:1602.07360*, 2016.

[29] A. Gholami, K. Kwon,and B. Wu, SqueezeNext: Hardware-Aware Neural Network Design, *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 1719-1728, 2018.

[30] 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*, pp. 6848-6856, 2018.

[31] N. Ma, X. Zhang, and H. T Zheng, ShuffleNet V2: Practical Guidelines for Efficient CNN Architecture Design, *European Conference on Computer Vision*, pp. 122-138, 2018.

[32] H. Zhao, Y. Zhou, L. Zhang, Y. Peng, Mixed YOLOv3-LITE: A Lightweight Real-Time Object Detection Method, *Sensors*, vol. 20, no. 7, pp. 1861-1871, 2020.

[33] J. Redmon, Darkent: Open Source Neural Networks in C, Online]. https://pjreddie.com/darknet/yolov2 (accessed on 2November 2020)

[34] R. Huang, J. Pedoeem, and C. Chen, YOLO-LITE: A Real-Time Object Detection Algorithm Optimized for Non-GPU Computers, *2018 IEEE International Conference on Big Data (Big Data)*, pp. 2503-2510, 2018.

[35] J. Redmon, Darkent:Open Source Neural Networks in C, 2013-2016, *[Online]. https://pjreddie.com/darknet/* .

[36] A. Bochkovskiy, Darknet: Open Source Neural Networks in Python, *[Online]. https://github.com/AlexeyAB/darknet*, 2020.

[37] R. Harshitha, M. Hussain, Surveillance Robot Using Raspberry Pi and IoT, *2018 International Conference on Design Innovations for 3Cs Compute Communicate Control*, pp. 46-51, 2018.

[38] A. Seo, Mobile Robot Platform with Arduino Uno and Raspberry Pi for Autonomous Navigation, *Procedia Manufacturing*, vol. 32, pp. 572-577, 2019.