Real-Time Lightweight CNN in Robots with Very Limited Computational Resources: Detecting Ball in NAO

Qingqing Yan(✉), Shu Li, Chengju Liu, and Qijun Chen

Tongji University, Shanghai 201804, China
qyan_0131@163.com

Abstract. This paper proposed a lightweight CNN architecture called *Binary-8* for ball detection on NAO robots together with a labelled dataset of 1000+ images containing balls in various scenarios to address the most basic and key issue in robot soccer games: detecting the ball. In contrast to the existing ball detection methods base on traditional machine learning and image processing, this paper presents a lightweight CNN object detection approach for CPU. In order to deal with the problems of tiny size, blurred image, occlusion and many other similar objects during detection, the paper designed a network structure with strong enough feature extraction ability. In order to achieve real time performance, the paper uses the ideas of depthwise separable convolution and binary weights. Besides, we also use SIMD (Single Instruction Multiple Data) to accelerate the operations. Full procedure and net structure have been given in this paper. Experimental results show that the proposed CNN architecture can run at full frame rate (140 Fps on CPU) with an accurate percentage of 97.13%.

Keywords: RoboCup · Ball detection · Depthwise separable convolution · Weight binarization · SIMD acceleration

1 Introduction

In the RoboCup Soccer Standard Platform League (SPL), ball detection is a fundamental and crucial ability for robotics, which is used to provide target distance and specific location for robots in various light environment. In addition, as the only standard device used in the SPL, the Softbank Robotics NAO has very limited resources, such as constrained computational abilities, and limited camera resolution. So, designing a real-time and efficient ball detection system has been a challenging task to address in the games: tiny size, blurred image, uneven illumination, occlusion and many similar objects. Traditional machine learning and image processing methods for ball recognition usually lead to a lot of and false positives and missed recognition.

State-of-the-art CNN shows excellent abilities of classification and object detection, but existing CNN-based detectors suffer from massive computational cost with server-class GPUs. When it comes to the application of CNN to mobile devices, there are several progresses in lightweight object detectors based on CNN, like YOLO-LITE, tiny-Yolo,

D. Tzovaras et al. (Eds.): ICVS 2019, LNCS 11754, pp. 24–34, 2019.
https://doi.org/10.1007/978-3-030-34995-0_3

Xception, MobileNet, XNOR-Net [1, 11–16], to improve the less computational cost on GPUs. However, the real-time operation performances are not able to meet the competition requirements when the nets are transferred to NAO robots vision application. In order to ensure the competitiveness of the game, we have to consider the balance between the real-time performance and detection accuracy. Furthermore, because the input resolutions are related to the model operation performance, we use the NAO's camera to capture a lot of images in various light conditions in our lab and competition field. In total, we captured 1008 unique images. While guaranteeing the detection performance, we pay more attention to the detection efficiency.

In this paper, we firstly provide a dataset, and then we investigate the effectiveness of depthwise convolution with binary weight in achieving real-time operational ability and desired detection accuracy on NAO robot. In the network design part, we described the process of building the network structure step by step in detail. The experimental results show that the computing time of the network structure designed by us decreases gradually without significant performance degradation. Satisfactory results have also been achieved in practical application.

The remainder of this letter is organized as follows. Section 2 introduces related works and analyses the existing shortcomings. Section 3 introduces our dataset in detail. Section 4 describes the procedure of developing the network structure step by step. Followed by Sect. 5 experiments and Sect. 6 conclusions.

2 Related Works

Lightweight CNN. As state-of-the-art one-stage object detection algorithms, YOLO [12–14] and SSD [18] enable to run in real time on GPUs with high accuracy. And YOLOv3-tiny [14] further improve efficiency of detection with acceptable accuracy on GPUs. But all of them suffer from massive computational cost. Recently, there have been several progresses in developing object detection algorithms to attribute to mobile and embedded vision applications, like MobileNet [15], and ShuffleNet [17]. However, these architecture designs are inspired by depthwise separable convolution which lacks efficient implementation. And other Pelee [3] enables to be executed on mobile devices at low frame rates. Compared with SSD MobileNet V1, YOLO-LITE [1] achieves the progress of computational speed improvement, but at the cost of losing the detection accuracy. So considering the balance between the real-time performance and detection accuracy on NAO robot vision application, we propose a real time lightweight CNN based on depthwise convolution with binary weight in NAO Robots for ball detection with better performance. Our design is mainly focused on efficiency.

Compression of CNNs. Generally, compression of CNNs enables to reduce the parameters and storage space of the model by means of related methods, such as pruning, quantization and approximation. Different methods have been proposed for pruning a network in [4–7]. Besides, quantization techniques were shown in [8, 9] for weights and representation of layers quantized in CNNs. With respect to approximation method, the authors proposed using FFT to compute the required convolutions in [10]. [11] Proposed a novel CNN which introduced two efficient approximations to CNNs by weight binary: Binary-Weight-Networks and XNOR Networks. In Binary-Weight-Networks, the weight values are approximated with the closest binary values, resulting in a 32x size smaller. Furthermore, XNOR Networks in which both the weights and the input of convolutional layers are binary values offers 58x speed up on a CPU, but 12.4% accuracy dropping in top-1 measure, by utilizing mostly XNOR and bit-counting operations. Inspired from the idea, our work utilizes the binary weight to compress our model.

Detection Algorithms on NAO. In response to the ball detection task, different recognition algorithms are proposed by the teams participating in the competition from all over the world. UChile, the Chilean team, has proposed a classification algorithm based on pentagonal recognition, which can better classify the positive and negative samples of the ball, but when the image is blurred, there are many missing recognition. German team HULK proposed a classification algorithm using Haar features. Although it enables to improve the accuracy of recognition to some extent, it takes a long time to compute and operation, resulting in the slow reaction of the robotics. Nao-Team HTWK and UT Austin Villa utilize a shallow CNN classifier, but they have to add other traditional image processing methods to generate Hypotheses first, and then use the CNN classifier to determine whether each hypothesis is a ball. However, in this way, the process of generating hypotheses with traditional methods will lead to leak recognition in all likelihood. Additionally, good features of the ball will be lost in the resize process. Except those, the shallow network with only 1–2 convolutional layers, generally consisting of convolution, Batch Normalization, ReLU activation and Max pooling layers, results in the weak feature extraction ability and poor generalization of the classifier.

3 Data Set

The proposed dataset was collected in our lab and real RoboCup competition fields, consisting of 1008 unique images with ball. Generally, the original images captured from the NAO's cameras is YUV format and the size of the images are lowered to 640 $\times$ 480 pixels and 320 $\times$ 240 pixels from the upper and lower camera, respectively. In order to speed up the operation process and improve the robustness under various scenarios, only the luminance (Y) channel of each image was extracted from NAO in action with various light conditions. Only when the original dataset obtained, were the ball pixels manually labelled. For the purpose of acceleration while ensuring detection accuracy, we resized the input images with label to a middle size of 416 $\times$ 416 pixels for later training and testing. An example of the proposed dataset is shown as Fig. 1. And the specific Dataset is online at https://github.com/qyan0131/Binary-8-DataSet.git.

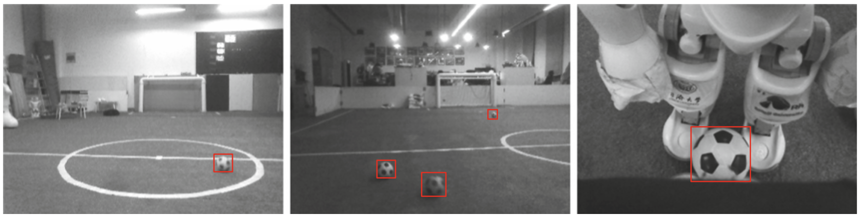

Fig. 1. Dataset for training and testing: It consists of 1008 Y channel images with ball captured from the NAO's cameras. The size of the images is lowered to 640 × 480 pixels and 320 × 240 pixels from the upper and lower camera, respectively

4 Network Design

In this section, we demonstrate the network design procedure in detail. The proposed network mainly focuses on the effectiveness and still maintain acceptable performances when transferred to NAO. And the design guidelines are composed of four parts. In the backbone part, we first design a small standard CNN as the fundamental network, which is enable to extract enough features for NAO vision detection. And under the premise of maintaining the accuracy for detection, we compress the standard CNN model as the backbone by reducing the number of layers and filters as much as possible. Then inspired from MobileNet ideas, we use the depthwise separable convolution based on the backbone to greatly reduce the number of parameters. Besides, a binary-weights approach is utilized in the point-wise convolution operation to further speed up the computational performance, because point-wise convolution operation takes more than 80% float type computation cost in MobileNet architecture while binary-weights use mostly XNOR and bit-counting operation. Finally, in view of the Intel CPU used by the NAO robot, we rewrite the convolution operation, batch normalization operation and ReLU none-linear activation operation of CNN network with SIMD instructions, which once again increases the speed by many times.

4.1 Backbone

In this paper, we no longer only apply CNN to classifier, we hope to use CNN to achieve end-to-end object detection. As consequence, we first build a backbone with sufficient feature extraction and generalization capabilities. Then, in order to deal with tiny size problem, we use anchor mechanism and design three anchors for different size objects. Finally, the output data structure of the network is given.

In the backbone design procedure, we adopt the sequential iteration method. We weigh the running time, accuracy against the number of layers and channels of the network. Because the number of network layers and layer filters will affect the network parameters and computation costs. The more network layers, the stronger the non-linear ability of the whole network, the stronger the ability to extract features, the stronger the robustness and generalization ability. The more layer filters, the more information flows between adjacent two layers of network, the richer and more accurate the extracted features are. However, increasing the number of network layers or increasing the number of layer filters will result in real-time performance degradation.

Darknet Reference Model is a small but efficient network proposed by [20]. Inspired by it, we prune Darknet Reference network layer by layer and keep training and testing. When the accuracy drops dramatically, we stop pruning the network layer. Then we start reducing layer filters. Similarly, when the accuracy on training and test sets declines significantly, we stop reducing the number of filters. In this way, the backbone containing 8 convolutional layers has been built, as shown in Table 1. We call it *Backbone-8*.

Table 1. *Backbone-8* architecture

Type/stride	Filter shape	Input size
Conv/s2	$3 \times 3 \times 3 \times 16$	$416 \times 416 \times 3$
Conv/s1	$3 \times 3 \times 16 \times 32$	$208 \times 208 \times 16$
Conv/s2	$3 \times 3 \times 32 \times 64$	$104 \times 104 \times 32$
Conv/s1	$3 \times 3 \times 64 \times 64$	$104 \times 104 \times 64$
Conv/s2	$3 \times 3 \times 64 \times 128$	$52 \times 52 \times 64$
Conv/s2	$3 \times 3 \times 128 \times 256$	$26 \times 26 \times 128$
Conv/s1	$3 \times 3 \times 256 \times 256$	$26 \times 26 \times 256$
Conv/s2	$3 \times 3 \times 256 \times 512$	$13 \times 13 \times 256$
Conv/s1	$1 \times 1 \times 512 \times 18$	$13 \times 13 \times 512$
Yolo		

4.2 Using Depthwise Convolution

MobileNet [15] uses depthwise separable convolutions, as opposed to YOLO's method, to lighten a model for real-time object detection. The idea of depthwise separable convolutions combines depthwise convolution and point-wise convolution. Depthwise convolution applies one filter on each channel then pointwise convolution applies a 1×1 convolution [15] to expand channels.

Based on [15], related to standard convolutions, using depthwise separable convolutions can get a reduction in computational cost of:

$$\frac{D_K \cdot D_K \cdot M \cdot D_F \cdot D_F + M \cdot N \cdot D_F \cdot D_F}{D_K \cdot D_K \cdot M \cdot N \cdot D_F \cdot D_F} = \frac{1}{N} + \frac{1}{D_K^2} \tag{1}$$

where $D_K \cdot D_K \cdot M \cdot N$ is the size of a parameterized convolution kernel K, and $D_F \times D_F \times M$ is the size of the feature map taken as input.

In order to make the network mentioned in Sect. 4.1 more real-time, we rewrite the backbone in MobileNet structure called *Depthwise-8*. In the model, we also use 3×3 depthwise separable convolutions to achieve 8 to 9 times less computation than *Backbone-8* (Table 2).

Table 2. *Depthwise-8* architecture

Type/stride	Filter shape	Input size
Conv/s2	3 × 3 × 3 × 16	416 × 416 × 3
Conv dw/s1	3 × 3 × 16 dw	208 × 208 × 16
Conv/s1	1 × 1 × 16 × 32	208 × 208 × 32
Conv dw/s2	3 × 3 × 32 dw	104 × 104 × 32
Conv/s1	1 × 1 × 32 × 64	104 × 104 × 64
Conv dw/s1	3 × 3 × 64 dw	104 × 104 × 64
Conv/s1	1 × 1 × 64 × 64	104 × 104 × 64
Conv dw/s2	3 × 3 × 64 dw	52 × 52 × 64
Conv/s1	1 × 1 × 64 × 128	52 × 52 × 128
Conv dw/s2	3 × 3 × 128 dw	26 × 26 × 128
Conv/s1	1 × 1 × 128 × 256	26 × 26 × 128
Conv dw/s1	3 × 3 × 256 dw	26 × 26 × 256
Conv/s1	1 × 1 × 256 × 256	26 × 26 × 256
Conv dw/s2	3 × 3 × 256 dw	13 × 13 × 128
Conv/s1	1 × 1 × 256 × 512	13 × 13 × 128
Conv/s1	1 × 1 × 512 × 18	13 × 13 × 512
Yolo		

4.3 Using Weight Binarization

Floating-point operation is time-consuming for device CPU, which is one of the most important factors restricting CNN running on CPU. Weights binarization can convert complex floating-point operations into simple XOR operations to accelerate the computation procedure. In the experiment we found that in Mobilenet architecture, point-wise convolution has limited feature extraction ability but it takes more than 80% of the total computation cost, while depthwise convolution can extract features effectively. Based on the above findings, we apply binary-weight operation to point-wise convolution to further accelerate the whole computation process. According to [11], the convolutional weight can be approximated by:

$$\mathcal{A}_{lk} = \frac{1}{n}\|\mathcal{W}_{lk}^{t}\|_{l1} \tag{2}$$

$$\mathcal{B}_{lk} = sign\left(\mathcal{W}_{lk}^{t}\right) \tag{3}$$

$$\widetilde{\mathcal{W}}_{lk} = \mathcal{A}_{lk}\mathcal{B}_{lk} \tag{4}$$

Where $\mathcal{W} \in \mathbb{R}^{n}$, ℓ, k represent kth filter in lth layer.

Since the number of network layers proposed in this paper is small and the input is 8-bit unsigned integer, if we binaries the network input i.e. the image or the output of each layer, it will take even more time to traverse the whole image. Therefore, this paper only binaries the network weight. The structure of binary-weighted network structure is shown in Table 3. We call it *Binary-8*.

Table 3. *Binary-8* architecture

Type/stride	Filter shape	Input size	Binary or not
Conv/s2	3 × 3 × 3 × 16	416 × 416 × 3	0
Conv dw/s1	3 × 3 × 16 dw	208 × 208 × 16	0
Conv/s1	1 × 1 × 16 × 32	208 × 208 × 32	1
Conv dw/s2	3 × 3 × 32 dw	104 × 104 × 32	0
Conv/s1	1 × 1 × 32 × 64	104 × 104 × 64	1
Conv dw/s1	3 × 3 × 64 dw	104 × 104 × 64	0
Conv/s1	1 × 1 × 64 × 64	104 × 104 × 64	1
Conv dw/s2	3 × 3 × 64 dw	52 × 52 × 64	0
Conv/s1	1 × 1 × 64 × 128	52 × 52 × 128	1
Conv dw/s2	3 × 3 × 128 dw	26 × 26 × 128	0
Conv/s1	1 × 1 × 128 × 256	26 × 26 × 128	1
Conv dw/s1	3 × 3 × 256 dw	26 × 26 × 256	0
Conv/s1	1 × 1 × 256 × 256	26 × 26 × 256	1
Conv dw/s2	3 × 3 × 256 dw	13 × 13 × 128	0
Conv/s1	1 × 1 × 256 × 512	13 × 13 × 128	1
Conv/s1	1 × 1 × 512 × 18	13 × 13 × 512	0
Yolo			

4.4 Boost Real Time Performance

The network in Sect. 4.3 already has strong real-time performance, but we can still use the SIMD instructions provided by Intel CPU to accelerate the operation on NAO robots to further enhance real-time performance. SIMD stands for Single Instruction Multiple Data. It can copy multiple operands and package them in a set of instructions in a single register. SSE is one of the instructions sets of SIMD which is supported by NAO's CPU. NAO uses 32-bit Intel CPU with 128-bit register length and 8-bit unsigned integer for CNN image input. Therefore, the operation of 32 pixel values can be processed at one time with SSE, leading to several times faster CNN calculation.

We rewrite the convolution operation, batch normalization operation and ReLU none-linear activation operation of CNN network with SIMD instructions.

5 Experimental Results

5.1 Comparison Among Proposed Networks

We evaluate the performance of our proposed approach on the task of NAO camera image. After successfully training models for our dataset, the network architectures accompany with their respective weights were test on the customized test set. The number of parameters, inference time and accuracy of the three proposed network is shown in Table 4. Figure 2 shows the APs during training phase.

Table 4. Comparison of three proposed network (Intel Atom 1.9 Hz CPU @ 320 * 240 pix)

Model	Million parameters	Inference time	Test set accuracy
Backbone-8	2.208	94 ms	98.17%
Depthwise-8	0.2612	17.5 ms	98.02%
Binary-8	0.2612	14.9 ms	97.13%
Binary-8-SSE	–	7.1 ms	–

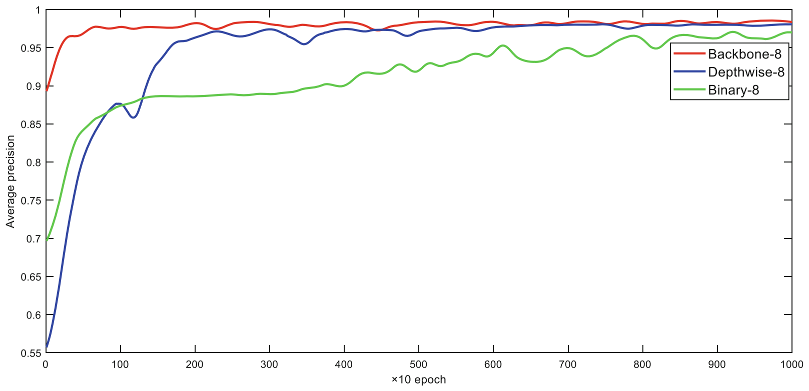

Fig. 2. Average precision of the three propose network

As shown in the table, every step of the network design improves the speed of computation while keep the correct rate is similar. And the final designed network with SSE optimization only takes 7.1 ms to process an image at the resolution of 320 × 240, which is fast enough to run on NAO robot.

Figure 3 shows the IoU and Loss during training phase of our proposed network. From the figure we can discover that the order of IoU rising speed is Backbone-8 > Depthwise-8 > Binary-8, however, with the increase of training epochs, the IoU of the three networks tends to be stable and the values are very similar. Loss in the figure is similar: although Backbone-8 declined the fastest, Binary-8 declined the slowest, they eventually tend to be stable. The difference is that, the Backbone-8 network's stable loss is the smallest, while the Binary-8 network is the highest. But considering the trade-off between computation time and performance, Binary-8 is the most efficient network.

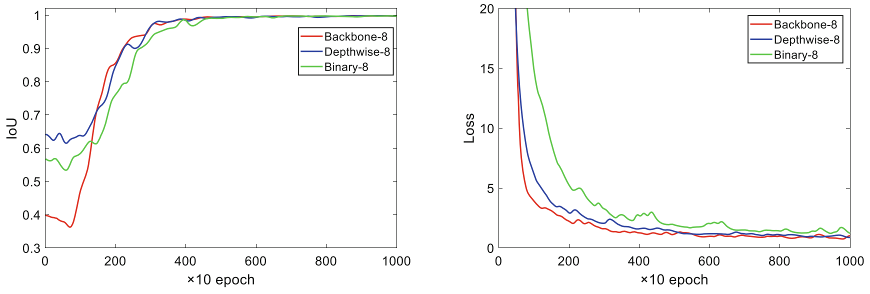

Fig. 3. IoU and Loss among proposed networks

5.2 Comparison Among Typical CNN Models

We also compare our proposed network with some of the famous or state-of-the-art lightweight network as shown in Table 5. Weights size, accurate percentage, BFLOPS and inference time on NAO robot are considered.

Table 5. Comparison of typical CNN models on Ball Dataset

Model	Weights size (MB)	AP@IOU0.5	BFLOPS	Inference time on NAO [ms]
AlexNet	238	98.78	2.862	1029
VGG	528	98.83	46.978	17310
Tiny-yolov3	35.19	98.86	1.773	566
MobilenetV1	12.22	98.65	3.798	113
Binary-8-SSE	1.05 M	97.22	0.189	7.1

According to the result in Table 5, compared with other typical lightweight models, the network we designed shows superior performance. The network structure proposed in this paper improves the computation speed greatly when accuracy is similar.

Experiments show that the proposed network has strong real-time performance (about 140 Fps on NAO robot CPU), and the accuracy (above 97%) can meet the recognition requirements.

6 Conclusion and Future Work

We propose a simple, efficient, and accurate CNN for object detection (ball) on NAO robots. We train a neural network that learns to find binary values for weights with depthwise convolution. In order to speed up execution on NAO CPU, we present a method of rewriting convolution layer, batch normalization layer and ReLU activation function using SSE. We also present a RoboCup Standard Platform image dataset with annotations, allowing other RoboCup researchers to train new models. The proposed network can detect balls accurately and run on NAO CPU in real-time.

In the future, we may continue focus on investigating more real-time CNNs on NAO robots with new tricks or new network architecture. We may research on the group point-wise convolution as proposed in ShuffleNet [17], as the point-wise convolution takes up a lot of computation in our network. We many also research on concatenating different layers to combine more feature information and further reduce the parameters. As for the RoboCup competition, we may use this network to detect all objects in games (i.e. ball, robot, obstacle, goalpost etc.). We may also use the backbone and similar tricks to build a real-time semantic segmentation algorithm on NAO robots to segment different objects/regions on the field.

Acknowledgements. The authors would like to acknowledge Team TJArk in RoboCup Standard Platform League for providing NAO robots and robot soccer field to capture dataset images. They also offer a lot of technical supports on operating and programming on NAO.

References

1. Pedoeem, J., Huang, R.: YOLO-LITE: a real-time object detection algorithm optimized for non-GPU computers. arXiv preprint arXiv:1811.05588 (2018)
2. Chollet, F.: Xception: deep learning with depthwise separable convolutions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1251–1258 (2017)
3. Wang, R.J., Li, X., Ling, C.X.: Pelee: a real-time object detection system on mobile devices. In: Advances in Neural Information Processing Systems, pp. 1963–1972 (2018)
4. Van Nguyen, H., Zhou, K., Vemulapalli, R.: Cross-domain synthesis of medical images using efficient location-sensitive deep network. In: Navab, N., Hornegger, J., Wells, W., Frangi, A. (eds.) MICCAI 2015. LNCS, vol. 9349, pp. 677–684. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24553-9_83
5. Han, S., Mao, H., Dall, W.J.: Deep compression: compressing deep neural networks with pruning, trained quantization and Huffman coding. arXiv preprint arXiv:1510.00149 (2015)
6. Chen, W., Wilson, J.T., Tyree, S., Weinberger, K.Q., Chen, Y.: Compressing neural networks with the hashing trick. arXiv preprint arXiv:1504.04788 (2015)
7. Jaderberg, M., Vedaldi, A., Zisserman, A.: Speeding up convolutional neural networks with low rank expansions. arXiv preprint arXiv:1405.3866 (2014)
8. Gong, Y., Liu, L., Yang, M., Bourdev, L.: Compressing deep convolutional networks using vector quantization. arXiv preprint arXiv:1412.6115 (2014)
9. Lin, Z., Courbariaux, M., Memisevic, R., Bengio, Y.: Neural networks with few multiplications. arXiv preprint arXiv:1510.03009 (2015)
10. Jaderberg, M., Vedaldi, A., Zisserman, A.: Speeding up convolutional neural networks with low rank expansions. arXiv:1405.3866 [cs.CV]

11. Rastegari, M., Ordonez, V., Redmon, J., Farhadi, A.: XNOR-Net: ImageNet classification using binary convolutional neural networks. arXiv preprint arXiv:1603.05279 (2016)
12. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: unified, real-time object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 779–788 (2016)
13. Redmon, J., Farhadi, A.: Yolo9000: better, faster, stronger. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 7263–7271 (2017)
14. Redmon, J., Farhadi, A.: Yolov3: an incremental improvement. arXiv preprint arXiv:1804.02767 (2018)
15. Howard, A.G., et al.: Mobilenets: efficient convolutional neural networks for mobile vision applications. CoRR, abs/1704.04861 (2017)
16. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.-C.: MobileNetV2: inverted residuals and linear bottlenecks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4510–4520 (2018)
17. Zhang, X., Zhou, X., Lin, M., Sun, J.: Shufflenet: an extremely efficient convolutional neural network for mobile devices. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 6848–6856 (2018)
18. Liu, W., et al.: SSD: single shot multibox detector. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9905, pp. 21–37. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46448-0_2
19. Iandola, F.N., Han, S., Moskewicz, M.W., Ashraf, K., Dally, W.J., Keutzer, K.: Squeezenet: alexnet-level accuracy with 50x fewer parameters and <0.5 MB model size. arXiv preprint arXiv:1602.07360 (2016)
20. Redmon, J., Lu, Y., Agirbau, L.: Huang ImageNet Classification [EB/OL], 3 November 2013. https://pjreddie.com/darknet/imagenet/