Article

An Efficient Object Detection Algorithm Based on Compressed Networks

Jianjun Li [1,*], Kangjian Peng [1] and Chin-Chen Chang [2,*]

[1] School of Computer Science and Engineering, Hangzhou Dianzi University, 310018 Hangzhou, China; pengkangjian@gmail.com
[2] Department of Information Engineering and Computer Science, Feng Chia University, Taichung 40724, Taiwan
* Correspondence: lijjcan@gmail.com (J.L.); alan3c@gmail.com (C.-C.C.); Tel.: +886-6527986

Received: 29 May 2018; Accepted: 20 June 2018; Published: 22 June 2018

Abstract: For a long time, object detection has been a popular but difficult research problem in the field of pattern recognition. In recent years, object detection algorithms based on convolutional neural networks have achieved excellent results. However, neural networks are computationally intensive and parameter redundant, so they are difficult to deploy on resource-limited embedded devices. Especially for two-stage detectors, operations and parameters are mainly clustered on feature fusion of proposals after the region of interest (ROI) pooling layer, and they are enormous. In order to deal with these problems, we propose a subnetwork—efficient feature fusion module (EFFM) to reduce the number of operations and parameters for a two-stage detector. In addition, we propose a multi-scale dilation region proposal network (RPN) to further improve detection accuracy. Finally, our accuracy is higher than Faster RCNN based on VGG16, the number of operations is only half of the latter, and the number of parameters is only one third.

Keywords: object detection; neural network; efficient feature fusion; multi-scale dilation network

1. Introduction

Artificial neural networks and pattern recognition have accomplished remarkable things up to now. Object detection is an important subproblem in the field of pattern recognition. The development of neural networks and object detection algorithms has a long history. Since AlexNet was proposed [1], which was based on ImageNet in 2012 and obtained state-of-the-art results, neural networks have once again caught researcher's attention. Various network structures have emerged from time to time. The Region Convolutional Neural Network (RCNN) was presented by Ross Girshick [2] in 2014. Benefitting from the powerful feature extraction capability of neural networks, the field of object detection has achieved a landmark breakthrough, with world records set by PASCAL VOC, COCO and other datasets being surpassed one by one. Meanwhile, Faster RCNN [3], YOLO [4], SSD [5] and other excellent object detection algorithms [6–10] are constantly presented within a short time. Then, VGG16 [11] and GoogLeNet [12] surpassed the ImageNet challenge record again in 2014. In particular, object detection algorithms reached a higher level of accuracy after ResNet [13] was released in 2016. However, as network structures are becoming more accurate, operations and parameters are also increasing rapidly.

Object detection algorithms are mainly divided into one-stage detectors and two-stage detectors. Two-stage detectors start with generating object proposals, and then localize and classify numbers of proposals. Two-stage detectors are more accurate than one-stage detectors and have stronger generalization abilities. Two-stage detectors include Faster RCNN/Region-based Fully Convolutional Networks (R-FCN)/HyperNet/Networks on Convolutional feature maps (NoCs) [3,6–8]. In a variety of object detection algorithms, NoCs adopted GoogLeNet as a feature extractor. Faster RCNN adopted ZFNet [14] and VGG16 as a feature extractor. In the R-FCN [6], the author replaced VGG16 with ResNet and adopted fully convolutional networks, achieving higher precision. Some research shows that VGGNet and ResNet is not an efficient network model [15]. They consume larger operations and more redundant parameters than other models.

In addition, two-stage detectors often extract hundreds or even thousands of object proposals, and each object proposal needs to be classified and localized. This module mainly uses two large fully connected layers or large convolutional layers [3,6], which often consume plentiful operations and parameters. They have the disadvantage of computationally intensive and parameter redundant, which focuses on the feature fusion module after the region of interest (ROI) pooling layer.

One-stage detectors consider object detection as a regression problem. They usually achieve real-time detection. One-stage detectors mainly include YOLO/YOLOv2/SSD. In the last two years, some progress has been made in improving efficiency [4,5,16] in the field of object detection and convolutional neural networks. YOLO adopted darknet, similar to GoogLeNet, as a feature extractor and has achieved good results in speed and accuracy. In YOLOv2 [17], the author proposed DarkNet19, which runs faster, and SSD adopted VGG16 as a feature extractor. However, they have deficiencies that detection accuracy is not high enough, especially for small targets.

Moreover, some researchers have proposed methods to quantify weight, making it possible to compress neural network weight to eight or even one bit [18], as well as hash coding [19]. There are also some methods for pruning weight [20] or filter [21] that greatly reduce the number of parameters. Some researchers presented a new algorithm for training the PNN neural network with fewer neurons [22]. Of course, there are also some hardware acceleration methods proposed [23]. In this paper, we mainly talk about designing efficient and compressed network structures based on two-stage detector—Faster RCNN.

In the meantime, most of the two-stage detectors generate object proposals by the last convolutional layer of feature extractor, resulting in a low recall rate of object proposals. HyperNet proposed a Hyper feature to fuse multi-scale features [7]. Gated Bi-Directional CNN presented a two-way gated network to selectively pass information in different scale context windows [24]. Some researchers detected areas of interest with heuristic methods [25]. However, they bring about too many operations and parameters.

In this paper, in order to reduce a large number of operations and parameters, we present an efficient feature fusion module (EFFM) for two-stage object detectors to classify and localize object proposals. We adopt pointwise convolution to fuse local channel information and reduce channel dimensionality, and adopt grouped convolution to extract features and grouped fully connected layers to fuse features. In order to improve recall rate of object proposals, we propose multi-scale dilation region proposal network (RPN). We adopt dilated convolution and multi-scale features to expand the receptive field. Furthermore, we adopt global feature weighting to enhance the network's feature extraction capability. In order to improve the efficiency of the feature extractor, we prune GoogLeNet to inception4e as a feature extractor.

The remainder of the paper is organized as follows. In Section 2, the related works are presented. Section 3 describes our methodology. Section 4 presents our experimental results. Section 5 concludes with a summary of our work.

2. Related Work

Knowledge Distillation [26]: The method is known as "teacher–student network". The "teacher network" is often a more complex network with excellent performance and generalization capabilities. This network can be used as the soft targets and teach the "student network", which is simpler and less computationally intensive. By learning the category distribution of the "teacher network", the knowledge of the "teacher network" is refined into a smaller model. Then, "student network" has similar performance to the "teacher network". This method significantly reduces the number of operations and parameters. However, there are also some deficiencies. This method can only be used for classification tasks with softmax loss function, hindering its application (e.g., object detection). Another disadvantage is that the assumption of the model is too strict, which degrades its performance.

NoCs [8]: The authors proposed a object detection framework based on the region proposal network. NoCs adopted GoogLeNet and ResNet as feature extractor. With the neural networks deepening, the accuracy of object detection algorithms is also improving. However, researchers do not pay much attention to feature fusion after the ROI pooling layer. The role of feature fusion module is to classify and localize object proposals. The feature fusion module are usually multi-layer perceptrons in the Fast/Faster RCNN. Therefore, NoCs explores the effects of different feature fusion modules and find they are as important as generating object proposals. However, NoCs only explores the importance of classifying and localizing object proposals. The feature fusion modules they proposed are more complex convolutional neural network and consumes larger operations and more redundant parameters than Faster RCNN.

HyperNet [7]: In order to combine multi-scale feature information in the region proposal network, HyperNet proposed different sampling strategies for different layers. They added a pooling layer at the lower convolutional layer. For higher convolutional layers, they added deconvolution operation (Deconv) for upsampling. A convolutional layer is applied after each sampling. The convolutional layers not only extract feature information, but also integrate them into hyper feature maps. Finally, multi-scale feature maps are normalized using local response normalization (LRN) and concatenate them to one single output cube. However, although HyperNet improves the detection accuracy, it brings a large number of operations and parameters in the region proposal network, and is unable to achieve real-time detection.

Deep Compression [20]: Deep compression combines pruning, trained quantization and Huffman coding to dramatically reduce the number of parameters. Pruning is cutting unimportant connections to reduce redundancy. Generally, if the absolute value is less than a certain threshold, it is treated as insignificant weight and set to zero. Quantization uses the K-means algorithm to cluster weight, and homogeneous weight are replaced with the same quantified weight. Huffman coding uses different lengths to represent different weights. Although deep compression can greatly reduce the number of parameters, the pruned matrix is a large number of sparse matrices, especially for the convolutional layer, which currently lacks of good support for sparse basic linear algebra subprograms and corresponding hardware. At the same time, the quantization could not accelerate the training process.

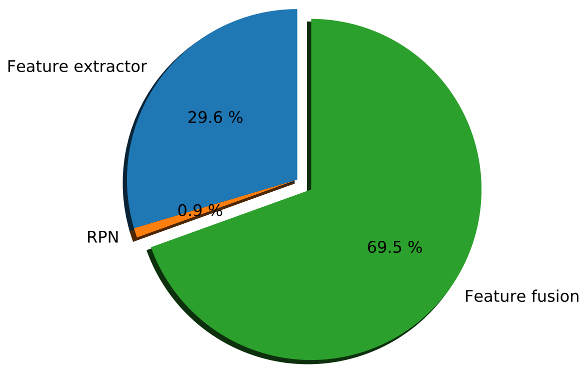
3. Methods

3.1. Faster R-CNN Analysis

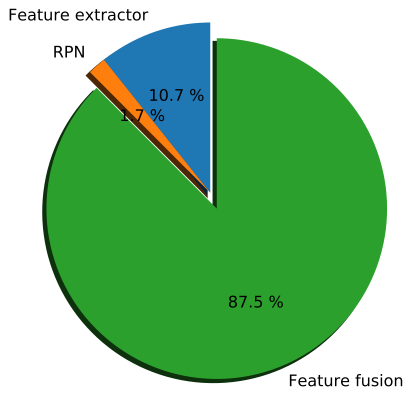
According to the design methods for Faster RCNN [3], it mainly contains three parts: feature extractor, region proposal network (RPN), and feature fusion module after the region of interest (ROI) pooling. Feature extractors are actually convolutional neural networks. RPN mainly outputs a set of rectangular object proposals, which is inspired by fully convolutional networks [27]. ROI pooling layers turn feature maps into fixed-length feature vector and greatly reduce the number of operations [28,29]. The feature fusion module is mainly to extract and fuse the feature information after ROI pooling, and then finally classify and localize the targets.

In the Faster RCNN, the feature extractor adopts ZFNet, VGG_CNN_M_1024 and VGG16. VGG_CNN_M_1024 is pruned by VGG16. RPN generates anchors based on convolutional layers in order to obtain object proposals. The feature fusion module uses two large fully connected layers.

Next, we will analyze the proportion of operations and parameters of three parts. We know that Faster RCNN needs to extract 300 proposals for classification and localization, while one-stage detector only takes one step. Our study found that two-stage detectors have enormous operations and parameters due to a large number of object proposals. Here, we take Faster RCNN based on VGG16 as a example. As can be seen in Figure 1, the number of operations and parameters of feature extractor, RPN, and feature fusion module accounts for 29.6%, 0.9%, 69.5%, and 10.7%, 1.7%, 87.5%, respectively. Figure 1 shows that the number of operations and parameters are mainly clustered in the feature fusion module after the ROI pooling layer. Thus, we want to significantly reduce the number of operations and parameters for the two-stage object detectors, we should design a more efficient feature fusion module.



(**a**)



(**b**)

Figure 1. Operations and parameters proportion of Faster RCNN. The feature fusion module consists of two fully connected layers. RPN is mainly a layer of convolution. The feature extractor is VGG16. (**a**) operations proportion; (**b**) parameters proportion.

3.2. Network Architecture

In order to reduce the number of operations and parameters and ensure detection accuracy simultaneously, we propose a subnetwork—efficient feature fusion module (EFFM) to classify and localize targets and multi-scale dilation RPN to generate object proposals, as shown in Figure 2. In our network structure, we prune GoogLeNet to an inception4e layer as our feature extractor. In the multi-scale dilation RPN, we fuse the feature information of multi-scale feature maps, and use dilated convolution to expand the receptive field, and then connect global feature weighting module to improve the recall rate of object proposals. We generate 2000 proposals by the multi-scale dilation RPN and select 300 proposals to classify and localize. In the EFFM, convolution layers are composed of grouped convolution, local channel fusion and grouped fully connected layer. Below, we will detail the EFFM and multi-scale dilation RPN.

3.3. Efficient Feature Fusion Module

At present, the feature fusion methods mainly include fully connected layers and convolutional layers plus global average pooling.

The roles of fully connected layers are: first, high-level features extracted by convolutional layers are dimensionally transformed into the target space for classification. Second, in the process of transfer learning, fully connected layers achieve higher accuracy when transferring from the source domain to the target domain. However, fully connected layers have the disadvantage of parameter redundancy [20].

As for fully convolution networks, the first few convolutional layers extract shallow features, while the middle convolutional layers mainly extract more abstract high-level features, and the last convolutional layers fuse the previous feature information and reduce dimensions through global average pooling layer. However, convolutional layers have less parameters but larger operations.

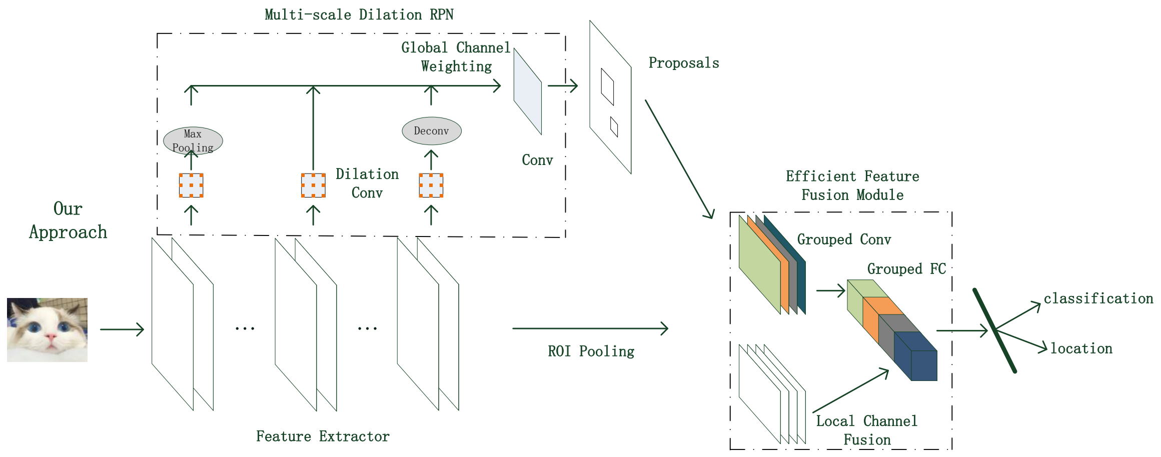


Figure 2. Network architecture. Our structure is mainly divided into multi-scale dilation RPN, efficient feature fusion module and feature extractor (GoogLeNet). FC means fully connected layer, conv means convolutional layer.

Many network structures are only considered for reducing the number of parameters, and they neglected to improve inference speed. In order to make two-stage detector's accuracy, operations and parameters reach a good trade-off, we combine the advantage of convolutional layers and fully connected layers and propose a subnetwork—efficient feature fusion module (EFFM). This module effectively fuses feature information and greatly reduces a large number of operations and parameters simultaneously.

Next, we will introduce EFFM in detail. In the two-stage detectors, we only have a 7×7 feature map after the ROI pooling layer, and the size of the feature map is relatively small. Therefore, as shown in Figure 3, we first adopted a pointwise convolution with 512 channels to fuse channel information and 480 channels to reduce dimensions. We greatly reduce the number of operations by choosing pointwise convolution rather than larger convolution kernel. Naturally, we also adopt a 3×3 grouped convolution with 512 channels to extract spatial information after dimensionality reduction. In order to reduce the number of operations and parameters, we divide the 3×3 convolution into four groups. Each group has 128 channels. If we choose 5×5 convolution kernel to extract spatial information, we would need to fill the feature map to size of 9×9. The feature map becomes 81 pixels after being filled, and 32 useless pixels are added compared to before. Therefore, in this case, we should consider more about local channel information fusion. Grouped convolution has little loss of accuracy, but it can greatly reduce the number of operations.

For convolutional neural networks, the number of parameters is mainly clustered on the first fully connected layer. For example, if you input a image with size of 224×224, after five times of pooling, the size of the feature map will reduce to size of 7×7. Assume that the number of channels in the last convolutional layer is 512, and the number of channels in the first fully connected layer is 4096. There are 102,760,448 parameters in the first fully connected layer. Moreover, research shows a large number of parameters of fully connected layers close to zero due to regularization [30]. As shown in Figure 3, we design more compact fully connected layers to ensure parameters utilization are more efficient. We divide the first fully connected layer with 1024 channels into four groups. Each group only has 256 channels. The number of first fully connected layer's parameters will be reduced to a quarter of the original. To ensure the integrity of information, we then connect a larger fully connected layer with 4096 channels to merge the final feature information. The second fully connected layer will not bring too many operations and parameters compared to the first fully connected layer. For example, the first fully connected layer is of size 4096, the second fully connected layer is of size 4096, and there are only 16,777,216 parameters in the second fully connected layer. Compared with pruning [20], grouped fully connected layers not only greatly reduce the number parameters, but also obtain a dense matrix instead of a sparse matrix, which is more suitable for hardware deployment.

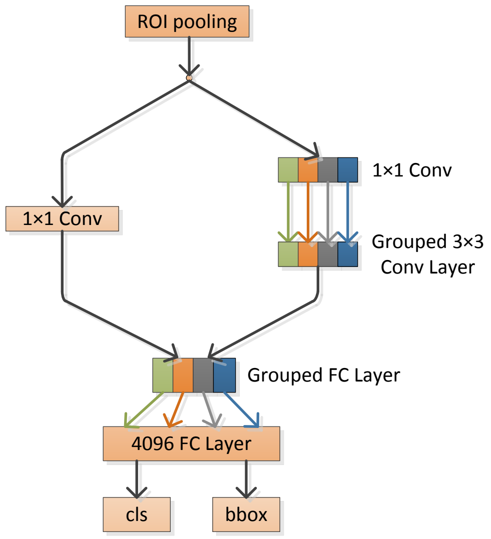


Figure 3. Efficient feature fusion module (EFFM) details. EFFM is mainly divided into convolutional layers and fully connected layers. We use dimensionality reduction and grouping to reduce a lot of operations and parameters.

3.4. Multi-Scale Dilation RPN

Faster RCNN only considers high-level feature information. This makes it easy to neglect low-level and high-resolution information, which leads to missing small targets. RPN is inspired by fully convolutional networks. Therefore, we adopt dilated convolution and multi-scale feature information, which make it possible to integrate more scale information and improve the recall rate of object proposals. Compared to the normal convolution kernel, dilated convolution has a hyper-parameter called dilation rate, which refers to the size of kernel intervals, as shown in Figure 4. Dilated convolution can expand the local receptive field. For example, a convolution kernel size of 3×3 (dilation: 2) can obtain a 7×7 receptive field [31]. Compared with max pooling, dilated convolution does not loss the feature information, which is beneficial to image segmentation and proposal generation.

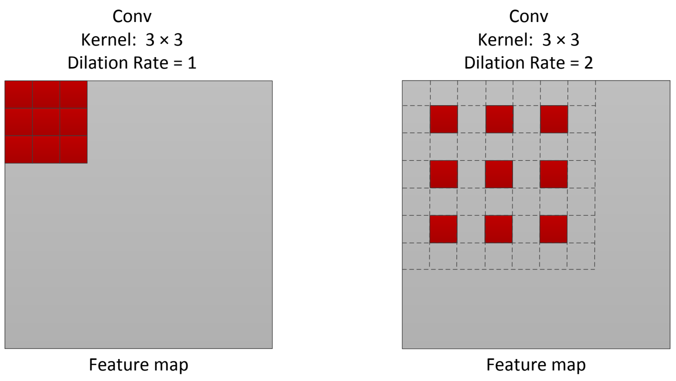


Figure 4. Dilated convolution with size of 3×3 and different dilation rates. (**Left**): dilation rate = 1, receptive field = 3×3; (**Right**): dilation rate = 2, receptive field = 7×7. The entire gray box represents the feature map, and the red boxes represent the convolutional kernel.

In addition, we can improve detection accuracy by incorporating deep but highly semantic, intermediate but really complementary, and shallow but naturally high-resolution features of the image [7]. In the multi-scale dilation RPN, we merge the feature maps of Conv2, Inception3b, and Inception4e by max pooling or deconvolution to the same size. After the Conv2 layer, we used a convolutional layer with 256 channels and a 3 × 3 kernel to extract feature and max pooling to reduce the feature maps to half of the original. After the inception3b layer, we directly connected a convolutional layer with 320 channels and 3 × 3 kernel to extract feature. After the inception4e layer, we connected a convolutional layer with 512 channels and 3 × 3 kernel to extract features, and then adopt deconvolution to upsample the feature maps to twice of the original. Then, we concatenate all the layers into the global feature weighting module.

The convolutional kernel is usually regarded as a local receptive field that fuses spatial information and channel information. We often acquire different sizes of spatial information by changing the size of the convolutional kernel. For channel information fusion, there are many ways, such as adding bitwise, grouping, and shuffle. In the RPN, Faster RCNN adopted convolutional layers with 512 channel and 3 × 3 kernel to generate object proposals. In order to enhance feature extraction capabilities of RPN, we performed a global feature weighting module between multi-scale feature maps and the first convolutional layer of RPN, as shown in Figure 5. The global feature weighting module uses a global average pooling layer to represent the weight of each feature channel and becomes a 1088-dimensional vector, and then nonlinearly transforms the weight through a 64-dimensional fully connected layer. Finally, a sigmoid gate function predicts the final weight of each channel [32]. We significantly improve detection accuracy while hardly increasing operations and parameters by adopting global feature weighting module. After the global feature weighting module, we connect two convolutional layers with 512 channels and 3 × 3 kernel. In order to reduce the size of feature map, the second convolutional layer's stride is 2.

In the multi-scale dilation RPN, We first generated about 2000 anchors in the last convolutional layer. The loss function is the same as the Faster RCNN. The last two convolutional layers connect to the loss layer and predict anchors. Each point of the feature map is nine different anchors. We randomly select 300 proposals in all anchors to classify and localize.

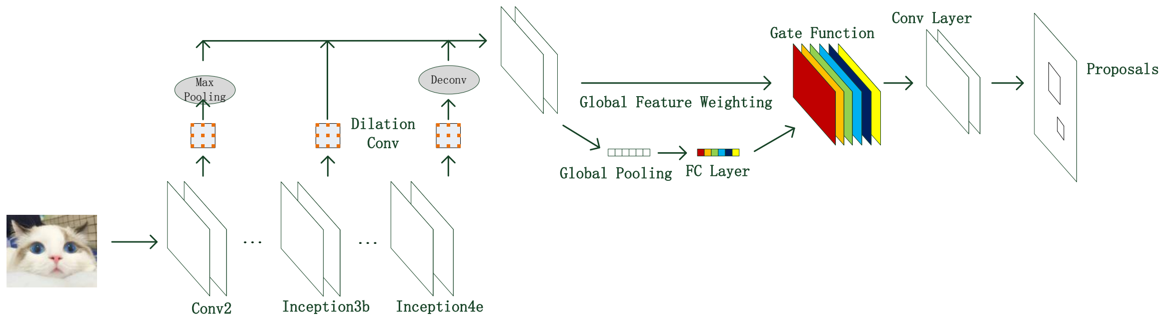


Figure 5. Multi-scale dilation region proposal network (RPN). This module incorporates feature information of Conv2/ Inception3b/Inception4e, and adopts dilated convolution to expand the receptive field, and finally connects a global feature weighting module.

4. Experiments and Results

4.1. Comparison with Faster RCNN

We performed a large number of experiments on the PASCAL VOC2007 and PASCAL VOC2012. PASCAL VOC2007 contains 5 k trainval images, 5 k test images and 20 categories. PASCAL VOC2012 contains more data on the basis of PASCAL VOC2007.

In order to take full advantage of the PASCAL VOC dataset, we combined VOC2007 with VOC2012 train/val images. The test process is based on VOC2007 test images. We implemented the experiment with Caffe (version: 1.0.0-rc3, University of California, Berkeley, CA, US). Training parameters are basically the same as Faster RCNN. Our initial learning rate is 0.001, which is reduced by one tenth every 30,000 iterations and weight decay is 0.0005. In order to ensure fair comparison, our test images are resized to 224 × 224. In the following, we will analyze the experimental results for detection accuracy, operations, model size, speedup rate, compression ratio, power consumption, parameter utilization and memory usage.

Table 1 shows the detection accuracy of various network structures. In Table 1, the CNN Model is the feature extractor and represents the network model. For example, GoogLeNet is our structure's feature extractor, and it represents our network structure. ZF, VGG_M_1024 and VGG16 are Faster RCNN's feature extractor, and they represent Faster RCNN based on ZF, VGG_M_1024 and VGG16, respectively. We only pruned GoogLeNet to the layer whose feature map is of size 14 × 14, mainly to ensure a large enough receptive field. The proposals refer to the number of object proposals extracted by a region proposal network. Our network model's proposals extracted by the multi-scale dilation RPN. VOC07 refers to our testing on the PASCAL VOC2007 testing set and training on the PASCAL VOC2007 training set. VOC07 + VOC12 refers to that we our testing on the PASCAL VOC2007 testing set and training on the PASCAL VOC2007 training set plus the PASCAL VOC2012 training set. The mAP refers to mean average precision.

From Table 1, we can see that our network structure's accuracy is 69.8% and 72.6% in the PASCAL VOC2007 training set and PASCAL VOC2007 plus 2012 training set, respectively. Our accuracy is slightly higher than Faster RCNN based on VGG16 (69.45%, 72.08%) and far higher than Faster RCNN based on ZF (59.01%, 60.7%) and VGG_M_1024 (60.27%, 62.19%). The VOC07 + VOC12 training set is generally 2% more accurate than the VOC07 training set. From the comparison of accuracy, we can see that our EFFM effectively classify and localize object proposals, and multi-scale dilation RPN effectively generate object proposals.

Table 1. Detection accuracy (test dataset: VOC2007). We extract 300 object proposals, the training sets are PASCAL VOC07 trainval and PASCAL VOC07 + 12 trainval respectively.

CNN Model	Proposals	mAP (VOC07) (%)	mAP (VOC07 + VOC12) (%)
ZF	300	59.01	60.7
VGG_M_1024	300	60.27	62.19
GoogLeNet	300	69.8	72.6
VGG16	300	69.45	72.08

Next, we will analyze the operations and model size of each neural network. Here, we do not calculate the runtime directly, mainly because Caffe does not provide good support for some neural computing modules, such as groups and deconvolution. Therefore, theoretical operations should be a better comparison. At the same time, we mainly calculate the sum of multiply-add because this is the main part of operations on neural networks. The *x*-coordinate of Figure 6 shows the operations. As shown in Figure 6, the operations of our structure are only 30.5 G-Ops and similar to the Faster RCNN based on VGG_M_1024 (25.72 G-Ops) and only half of the Faster RCNN based on VGG16 (51.8 G-Ops). From the experimental results, we can see that our EFFM greatly reduced the number of operations.

In Figure 6, the size of the blobs is proportional to the model size. Model size represents the number of parameters in each network. A legend is described in the bottom right corner, spanning from 20 × 10^6 to 500 × 10^6 bytes. Model size mainly comes from the size of the Caffemodel. As we can see from Figure 6, the model size of Faster RCNN based on ZFNet, VGG_CNN_M_1024, and VGG16, which use a large number of fully connected layers, is of size 227M, 334M, 523M, respectively, and go far beyond our network structure. They consume lots of memory and storage space, while our structure is only 156 M, one third of the Faster RCNN based on VGG16. From the experimental results, we can see that our EFFM greatly reduced the number of parameters.

Next, we will analyze speedup rate and compression ratio of our network structure based on the above experimental results. Compared with Faster RCNN based on VGG16, the speedup rate of our network structure is approximately 1.7. This result is calculated based on the operations, not actual runtime. The specific runtime will be discussed in Section 4.2. Our network structure's compression ratio is about 3.35 compared with the Faster RCNN based on VGG16. This result is calculated based on the size of the Caffemodel.

From Figure 6, we can see there exists a positive correlation between operations and model size. Faster RCNN based on VGG16 and ZFNet, whose operations and model size are quite enormous relative to detection accuracy, are not efficient enough. According to previous experience, the larger network has better detection accuracy. However, from our experimental results and analysis, we can greatly reduce the number of operations and parameters without losing accuracy by using an efficient feature fusion module and multi-scale dilation RPN.

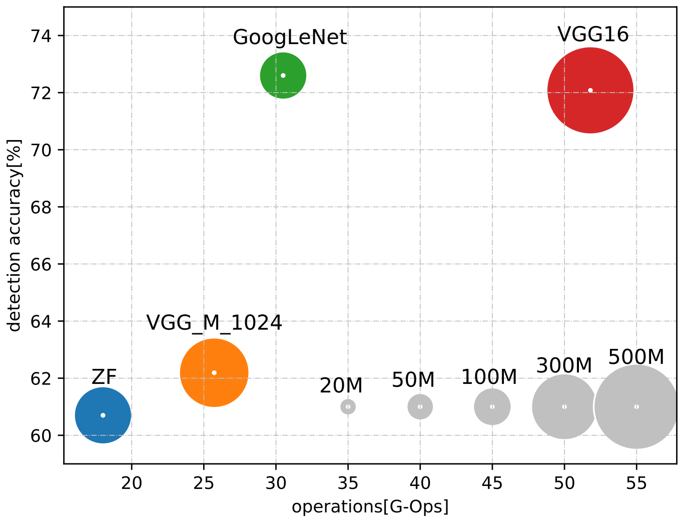


Figure 6. Comparison of operations and model size. This chart show the comparison between detection accuracy, operations and parameters. ZF, VGG_M_1024, VGG16 and GoogLeNet are feature extractors and represent corresponding object detection algorithms. ZF, VGG_M_1024, VGG16 are based on the Faster RCNN. GoogLeNet represents our network model.

Figure 7a compares power consumption in the forward calculation process. The experimental results were recorded every second. We used a NVIDIA TitanX graphics card (Santa Clara, CA, USA). Hundreds of experimental data were obtained. The final results consist of average value. From Figure 7a, we can see that the power consumption of Faster RCNN based on VGG16 is relatively high, about 165 W. In addition, our network structure maintains a lower level, only about 137 W. Due to the high power consumption of the NVIDIA TitanX, the power consumption can be significantly reduced with a low power NVIDIA Jetson TK1 or Field Programmable Gate Array (FPGA). Here, we only make a relative comparison based on our hardware platform.

Next, we will analyze the parameter utilization further. We adopted accuracy contribution per unit parameters (Acpup) to qualitatively analyze the relationship between detection accuracy and the number of parameters. Acpup is the ratio of detection accuracy to the number of parameters, and its unit is percent per Megabit (%/M). Figure 7b shows the comparison of Acpup among different network structures. As can be seen from Figure 7b, parameter utilization of our structure is much higher than other models, about 0.465. Furthermore, Faster RCNN based on VGG16 is the lowest, just 0.138. From the comparison of two figures, we can find that power consumption is a negative correlation with parameter utilization. Figure 8 mainly shows the comparison of memory usage. Here, the main reference to the experimental data is memory usage of the graphics card in the forward calculation process. Our network structure's memory usage is about 3000 M, much less than Faster RCNN based on VGG16. Since memory optimization of Caffe is not good enough. In fact, we can reduce memory usage further according to the specific network structure and hardware during actual deployment.

From the above experimental analysis, we can see that our network structure can achieve good progress in terms of power consumption, parameter utilization and memory usage compared with the Faster RCNN.

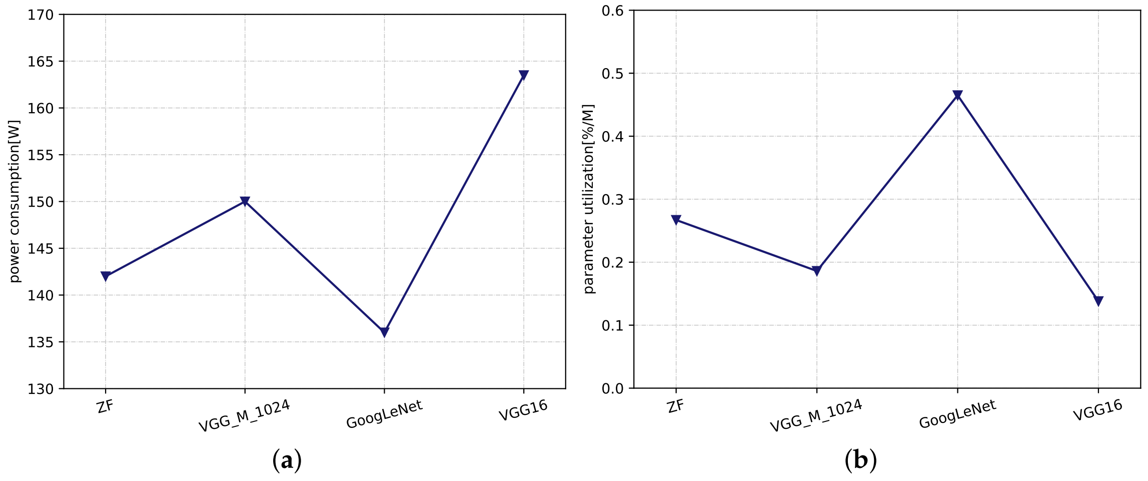


Figure 7. The relationship between power consumption and parameter utilization. (**a**) power consumption: comparison of power consumption with various network structures; (**b**) parameter utilization: comparison of parameter utilization with various network structures.

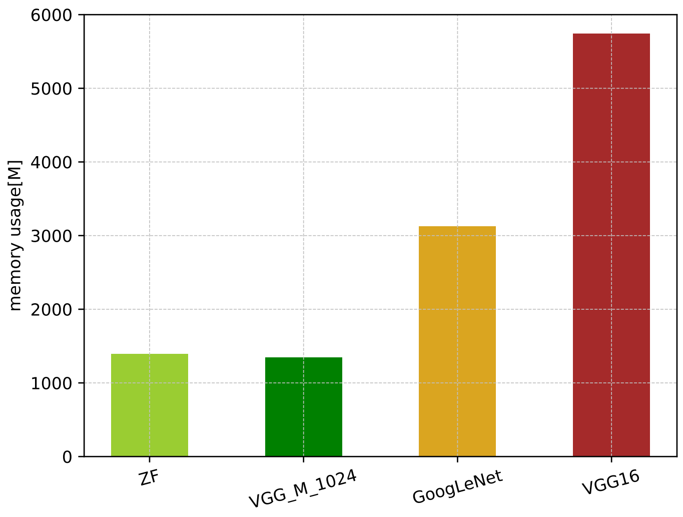


Figure 8. Comparison of memory usage (forward calculation process) with various network structures.

4.2. Comparison with Other Two-Stage Detectors

Here, we will make a comparison with other two-stage detectors and analyze similarities and differences. Two-stage detectors are mainly divided into feature extractor, object proposal network and feature fusion module. We mainly discuss them based on these three parts.

As can be seen from Table 2, compared to our efficient feature fusion module (EFFM), NoCs uses one or two convolutional layers with 256 channels and 3×3 kernel, plus two fully connected layers with 4096 channels. NoCs explored different design methods of convolutional neural networks in the feature fusion module. HyperNet has achieved higher detection accuracy by designing a hyper feature in the region proposal network. However, due to the large number of operations and parameters in the hyper feature and feature fusion module, it runs slower. The HyperNet also uses convolutional layers plus large fully connected layers in the feature fusion module, like NoCs. The Faster RCNN only adopts the last layer of the feature extractor to generate proposals in the region proposal network. The feature fusion module uses two large fully connected layers of size 4096 × 4096.

As can be seen from Table 3, our network has achieved a good balance between accuracy, speed and model size. Our detection accuracy is 72.6%, only second to HyperNet (74.8%). The runtime of our network structure is about 11 frames per second, which is higher than other two-stage detectors. Moreover, because Caffe does not provide good support for some neural computing units (e.g., grouped convolution, grouped fully connected layers) and our network's operations are only about half of other network structures, our structure's runtime can optimize further. Our network structure's model size is only 156 M, which is much smaller than other detectors.

Table 2. Comparison of network structures with other two-stage detectors. We mainly divide two-stage detectors into three parts: feature extractor, proposal network and feature fusion module.

Detector	Feature Extractor	Proposal Network	Feature Fusion
Our Network	GoogLeNet	multi-scale dilation RPN	EFFM
NoCs (1)	GoogLeNet	RPN	c256-f4096-f4096-f21
NoCs (2)	GoogLeNet	RPN	c256-c256-f4096-f4096-f21
Faster RCNN	VGG16	RPN	f4096-f4096-f21
HyperNet	VGG16	Hyper feature	c256-f4096-f4096-f21

Table 3. Comparison of operations, fps and model size with other two-stage detectors. The Detector corresponds to the network structure in Table 2, respectively. We trained on PASCAL VOC07 + 12 trainval set and tested on PASCAL VOC2007 test set. The mAP refers to mean average precision. The fps means frame per second.

Detector	mAP (%)	Operations (G-Ops)	fps (s)	Model Size (M)
Our Network	72.6	30.5	11	156
NoCs (1)	72.6	58.3	7	298
NoCs (2)	68.9	66.97	6	300
Faster RCNN	72.08	51.8	7	523
HyperNet	74.8	-	5	-

5. Conclusions

In order to solve the low efficiency problem of object detection algorithms based on convolutional neural networks, this paper proposed an effective feature fusion module after an ROI pooling layer and multi-scale dilation region proposal network. We reduced a large number of operations and parameters while improving detection accuracy simultaneously and achieved a good trade-off in terms of detection accuracy, operations and parameters. We also analyzed the detection accuracy, operations, model size, speedup rate, compression ratio, power consumption, parameter utilization, and memory usage of various network structures. In many aspects, we surpassed other two-stage detectors. However, there are still some issues that need further study on achieving a higher compression ratio and giving full play to a speedup rate for specific hardware. In the future, we will continue to research the method of model compression (e.g., combine channel pruning [21] and weight pruning [20]) and deploy to specific hardware (e.g., Nvidia Jetson TK1, FPGA even Advanced RISC Machine (ARM)) so that object detection algorithms based on neural networks can be more widely applied on mobile devices and embedded terminals without losing detection accuracy.

Author Contributions: In this paper, all of the authors participated in the data analysis and experiment, algorithm design, as well as the manuscript's preparation. All of the authors have approved the submitted manuscript. All of the authors equally contributed to the writing of the paper.

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

References

1. Kirzhevsky, A.; Sutskever, I.; Hinton, G. ImageNet classification with deep convolutional neural networks. In Proceedings of the International Conference on Neural Information Processing Systems, Lake Tahoe, NV, USA, 3–8 December 2012; Volume 25, pp. 1097–1105.
2. Girshick, R.B.; Donahue, J.; Darrell, T.; Malik, J. Rich feature hierarchies for accurate object detection and semantic segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, 23–28 June 2014; pp. 580–587.
3. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster R-CNN: Towards real time object detection with region proposal networks. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *39*, 1137–1149. [CrossRef] [PubMed]
4. 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, Las Vegas, NV, USA, 27–30 June 2016; pp. 779–788.
5. Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.; Berg, A.C. SSD: Single shot multibox detector. In Proceedings of the European Conference on Computer Vision, Amsterdam, The Netherlands, 8–16 October 2016; pp. 21–37.
6. Dai, J.; Li, Y.; He, K.; Sun, J. R-FCN: Object Detection via Region-based Fully Convolutional Networks. In Proceedings of the Conference on Neural Information Processing Systems, Barcelona, Spain, 5–6 December 2016; pp. 379–387.
7. Kong, T.; Yao, A.; Chen, Y.; Sun, F. HyperNet: Towards Accurate Region Proposal Generation and Joint Object Detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 845–853.
8. Ren, S.; He, K.; Girshick, R.B.; Zhang, X.; Sun, J. Object Detection Networks on Convolutional Feature Maps. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *39*, 1476–1481. [CrossRef] [PubMed]
9. Wozniak, M.; Polap, D.; Komider, L.; Clapa, T. Automated fluorescence microscopy image analysis of Pseudomonas aeruginosa bacteria in alive and dead stadium. *Eng. Appl. Artif. Intell.* **2018**, *67*, 100–110. [CrossRef]
10. Lai, W.W.L.; Chang, R.K.W.; Sham, J.F.C. Detection and imaging of cityś underground void by GPR. In Proceedings of the 2017 9th International Workshop on Advanced Ground Penetrating Radar (IWAGPR), Edinburgh, UK, 28–30 June 2017; pp. 1–6.
11. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv* **2014**, arXiv:1409.1556.
12. Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; Rabinovich, A. Going deeper with convolutions. In Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 7–12 June 2015; pp. 1–9.
13. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
14. Zeiler, M.D.; Fergus, R. Visualizing and Understanding Convolutional Networks. In Proceedings of the European Conference on Computer Vision, Portland, OR, USA, 23–28 June 2013; pp. 818–833.
15. Canziani, A.; Paszke, A.; Culurciello, E. An Analysis of Deep Neural Network Models for Practical Applications. *arXiv* **2016**, arXiv:1605.07678.
16. Yan, C.; Xie, H.; Liu, S.; Yin, J.; Zhang, Y.; Dai, Q. Effective Uyghur Language Text Detection in Complex Background Images for Traffic Prompt Identification. *IEEE Trans. Intell. Transp. Syst.* **2018**, *19*, 220–229. [CrossRef]
17. Redmon, J.; Farhadi, A. YOLO9000: Better, Faster, Stronger. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Hawaii, HI, USA, 21–26 July 2017; pp. 6517–6525.
18. Courbariaux, M.; Hubara, I.; Soudry, D.; Elyaniv, R.; Bengio, Y. Binarized Neural Networks: Training Deep Neural Networks with Weights and Activations Constrained to +1 or -1. *arXiv* **2016**, arXiv:1602.02830.
19. Yan, C.; Xie, H.; Yang, D.; Yin, J.; Zhang, Y.; Dai, Q. Supervised hash coding with deep neural network for environment perception of intelligent vehicles. *IEEE Trans. Intell. Transp. Syst.* **2018**, *19*, 284–295. [CrossRef]
20. Song, H.; Mao, H.; Dally, W.J. Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding. *arXiv* **2016**, arXiv:1510.00149.

21. He, Y.; Zhang, X.; Sun, J. Channel Pruning for Accelerating Very Deep Neural Networks. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October, 2017; pp. 1398–1406.
22. Wozniak, M.; Polap, D.; Capizzi, G.; Sciuto, G.L.; Kosmider, L.; Frankiewicz, K. Small lung nodules detection based on local variance analysis and probabilistic neural network. *Comput. Methods Programs Biomed.* **2018**, *161*, 173–180. [CrossRef] [PubMed]
23. Yan, C.; Zhang, Y.; Xu, J.; Dai, F.; Zhang, J.; Dai, Q. Efficient parallel framework for hevc motion estimation on many-core processors. *IEEE Trans. Circuits Syst. Video Technol.* **2014**, *24*, 2077–2089. [CrossRef]
24. Zeng, X.; Ouyang, W.; Yan, J.; Li, H.; Xiao, T.; Wang, K. Crafting gbd-net for object detection. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *39*. [CrossRef] [PubMed]
25. Wozniak, M.; Polap, D. Adaptive neuro-heuristic hybrid model for fruit peel defects detection. *Neural Netw.* **2018**, *98*, 16–33. [CrossRef] [PubMed]
26. Hinton, G.E.; Vinyals, O.; Dean, J. Distilling the Knowledge in a Neural Network. *arXiv* **2015**, arXiv:1503.02531.
27. Long, J.; Shelhamer, E.; Darrell, T. Fully convolutional networks for semantic segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 3431–3440.
28. Girshick, R. Fast R-CNN. In Proceedings of the IEEE International Conference on Computer Vision, Santiago, Chile, 13–16 December 2015; pp. 1440–1448.
29. He, K.; Zhang, X.; Ren, S.; Sun, J. Spatial pyramid pooling in deep convolutional networks for visual recognition. In European Conference on Computer Vision, Proceedings of the 13th European Conference, Zurich, Switzerland, 6–12 September 2014; Springer: Cham, Switzerland, 2014; pp. 346–361.
30. Han, S.; Pool, J.; Tran, J.; Dally, W.J. Learning both weights and connections for efficient neural networks. In Proceedings of the International Conference on Neural Information Processing Systems, Montreal, Canada, 7–12 December 2015; Volume 25, pp. 1135–1143.
31. Yu, F.; Koltun, V. Multi-Scale Context Aggregation by Dilated Convolutions. In Proceedings of the International Conference on Learning Representations, San Juan, Puerto Rico, 2–4 May 2016.
32. Hu, J.; Shen, L.; Sun, G. Squeeze-and-Excitation Networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018.