Real-Time Object Detection With Reduced Region Proposal Network via Multi-Feature Concatenation

Kuan-Hung Shih, Ching-Te Chiu, Jiou-Ai Lin, and Yen-Yu Bu

***Abstract*— In recent years, object detection became more and more important following the successful results from studies in deep learning. Two types of neural network architectures are used for object detection: one-stage and two-stage. In this paper, we analyze a widely used two-stage architecture called Faster R-CNN to improve the inference time and achieve real-time object detection without compromising on accuracy. To increase the computation efficiency, pruning is first adopted to reduce the weights in convolutional and fully connected (FC) layers. However, this reduces the accuracy of detection. To address this loss in accuracy, we propose a reduced region proposal network (RRPN) with dilated convolution and concatenation of multi-scale features. In the assisted multi-feature concatenation, we propose the intra-layer concatenation and proposal refinement to efficiently integrate the feature maps from different convolutional layers; this is then provided as an input to the RRPN. Using the proposed method, the network can find object bounding boxes more accurately, thus compensating for the loss arising from compression. Finally, we test the proposed architecture using ZF-Net and VGG16 as a backbone network on the image sets in PASCAL VOC 2007 or VOC 2012. The results show that we can compress the parameters of the ZF-Net-based network by 81.2% and save 66% of computation. The parameters of VGG16-based network are compressed by 73% and save 77% of computation. Consequently, the inference speed is improved from 27 to 40 frames/s for ZF-Net and 9 to 27 frames/s for VGG16. Despite significant compression rates, the accuracy of ZF-Net is increased from 2.2% to 60.2% mean average precision (mAP) and that of VGG16 is increased from 2.6% to 69.1% mAP.**

***Index Terms*— Multi-feature concatenation, object detection, region proposal network (RPN), weight pruning.**

I. Introduction

OBJECT detection includes the operations of object location and classification. With the proliferation of camera-enabled mobile devices, there has been an increase in the number of images and video clips. Those valuable data can be applied in deep learning. For example, convolutional neural networks (CNNs) [1]–[3] have been successfully used in object detection to improve the object location and classification accuracy [4]. With the rising popularity of autonomous driving, new applications, such as traffic sign detection [5] and pedestrian detection [6], have emerged, and object detection has gained significance [7]. Furthermore, the emergence of drone technology, in which drones are used to monitor the areas and places that people cannot reach, has also triggered interest in object detection techniques. Images taken by drones are usually captured from angles that are different from that used in ordinary cameras. References [8] and [9] have demonstrated that object detection can also process images from drones. Lightweight object detection systems suitable for use in drones will give rise to more practical applications using drones. At the airport, customs officers use X-ray equipment to check the baggage contents, but the generated images are not normal RGB images. Moreover, personnel must quickly inspect each X-ray image to determine whether dangerous items are present. References [10] and [11] show that object detection using deep learning can assist in identifying objects in X-ray images. For robotic [12], [13] applications, object detection is a crucial technique for the correct action of robots.

There are three tasks that object detection must solve: classification, localization, and finding all objects. Compared to simple image classification, there are many more aspects that object detection needs to address. As a result, the computational complexity is significantly higher than that of the image classification. However, most applications need to perform object detection in real-time; therefore, inference speed is an important consideration in object detection. In addition, many researchers have been working on decreasing the model size because smaller model sizes have a higher chance on being loaded as CNN models on mobile devices.

Girshick *et al.* [14] proposed a region-based rich feature hierarchy for accurate object detection using CNN (R-CNN). Following this, Faster R-CNN was proposed [15] to add region proposal networks (RPNs) to CNN for improving the computational speed. Finding regions of interest (ROIs) and classification are executed in different stages in Faster R-CNN. An important feature in Faster R-CNN is the RPN that is used to find all the ROIs. To achieve higher accuracy, VGG-16 is used as the backbone model in Faster R-CNN. Although Faster R-CNN is faster than the previous R-CNN versions [14], [16], the VGG-16-based Faster R-CNN has a frame rate of less than 10 frames/s [15]. There are two main reasons. First, high-resolution input images are required. For each ROI, max pooling is used in ROIPooling [3], [15]–[20] to crop and resize the dimension of feature maps. However, after several layers of pooling, the feature maps become too small for accurate bounding box regression and classification. The input image resolution has to be high enough; typically, the resolution is

Manuscript received October 2, 2018; revised March 1, 2019, April 29, 2019 and June 21, 2019; accepted July 5, 2019. This work was supported by the Ministry of Science and Technology (MOST), Taiwan, Optimizing and Compressing Deep Learning Networks With Applications in Computer Vision Project, under Grant MOST 107-2634-F-007-003. *(Corresponding author: Ching-Te Chiu.)*

The authors are with the Department of Computer Science, National Tsing Hua University, Hsinchu 30000, Taiwan (e-mail: chiusms@cs.nthu.edu.tw).

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNNLS.2019.2929059

set to 600×1000. Another reason is that too $n =$ many ROIs are generated by the RPN; each of which has to be processed through various fully connected (FC) layers and classifiers. The above-mentioned issues are also present in Faster R-CNN and lead to an increase in the number of computations.

Our goal is to reduce the number of computations to allow Faster R-CNN to achieve real-time object detection. Modern networks [1], [2] use several parameters that not only require extra storage space but also require a larger number of computations. We adopt weight pruning on the neural network to compress parameters, solving the memory and performance issues. However, pruning involves deleting some filters, which, in turn, results in information loss and loss of accuracy. Therefore, the challenge is to design an object detection system that is lightweight enough for use in mobile devices but also accurate so that people can rely on its results.

Our goal is to use pruning for reducing the number of parameters and computations without losing accuracy. First, we survey different pruning methods. After pruning the convolutional layers and FC layers, we propose the use of a reduced RPN and assisted multi-feature concatenation to compensate for the loss in accuracy. We introduce 1×1 convolution, a thinner convolution channel, and dilated convolution in the reduced RPN. Compared to the original RPN, our reduced RPN uses less than 10% of the parameters while improving accuracy. In the assisted multi-feature concatenation, we combine as many feature maps as possible to provide the reduced RPN with sufficient information to find all the ROIs more accurately. Finally, we test our proposed method on both ZF-Net [21] and VGG16 [2]; these are the same as that mentioned in the original Faster R-CNN paper [15].

Most pruning methods can reduce the number of parameters or computations in a neural network model at the cost of loss in accuracy. Our main contribution is an architectural design capable of recovering the accuracy loss caused by pruning. The proposed architectural changes include using a reduced RPN with dilated convolution and multi-feature concatenation. The reduced RPN with dilated convolution decreases the channel number of feature maps to reduce the model size while finding more accurate ROIs. The assisted multi-feature concatenation helps extract features from sub-layers with difference scales.

Our contributions in this paper are summarized as follows.

1) We apply pruning on ZF-Net and VGG16 to eliminate 79% and 71% of the parameters, respectively. After pruning, the mean average precision (mAP) values are 58.1% and 66.5%.
2) On using the proposed reduced RPN and intra-layer concatenation with proposal refinement, the model size can further eliminate approximately 2%, and the mAPs can be improved by 2.1%–60.2% for ZF-Net and by 2.6%–69.1% for VGG16.
3) Overall, we compress the parameters of ZF-Net by 81.2%, and the parameters of VGG16 are compressed by 73%. The high compression rates greatly benefit memory-limited systems.
4) With respect to the inference speed, 40- and 27-frames/s object detections are achieved for ZF-Net and VGG16, respectively, for input image resolution of 600×1000 and reduce the amount of computation by 34% and 23%, respectively, for ZF-net and VGG-16. Therefore, it is possible to incorporate object detection in real-time applications using the proposed architecture.
5) In this paper, we test the proposed method on both ZF-Net and VGG16, which implies that our method can be applied to the most modern CNN models.

II. Related Work

Our work uses pruning on CNN models in the context of object detection; therefore, we briefly introduce different pruning methods and describe other studies that deal with object detection.

A. Pruning

Owing to the presence of several filters, deep learning models can learn many patterns from data and outperform traditional algorithms. Many parameters in the neural networks are redundant. Deleting less important connections are the focus of early compression works. Optimal Brain Damage [22] and Optimal Brain Surgeon [23] adopt the Taylor expansion to rank the importance of weights, while absolute values are used to evaluate their importance in [24]. Researchers have suggested pruning entire filters recently. Data-independent and data-dependent are two common adopted approaches. In data-independent methods, the values of filters are analyzed directly; for instance, the absolute values of all weights in a filter are summed up and their significance are sorted in [25]. In contrast, data-dependent methods evaluate activation values. Molchanov *et al.* [26] use the Taylor expansion to evaluate activation. For pruning filters in neural network models, interactions between filters are considered in [27] and [28], and LASSO regression is used to find candidates in [29].

B. Object Detection

Object detection is an important field in computer vision that has been studied for a long time. Traditional detection methods comprise three major steps: finding proposals, extracting features, and classification [4] with several methods possible for each step. Methods for finding proposals include selective search [30], edge box [31], and multiscale combinatorial grouping [32]. Histogram of oriented gradient (HOG) [33], scale-invariant feature transform (SIFT) [34], and so on are some of the methods used in the extracting features phase. Classification methods include support vector machine (SVM) [35], adaboost [36], and so on.

With the introduction of deep learning models, many studies have applied these for object detection and have achieved successful results. One- and two-stage network architectures are predominantly used in object detection. The input images to the one-stage architecture are sliced into several grid cells. The classifier outputs a vector that encodes the information of each grid cell [37]–[40]. The two-stage architecture first finds the ROIs and then performs detection in every ROI [3], [15]–[20]. Comparing the two architectures, the one-stage

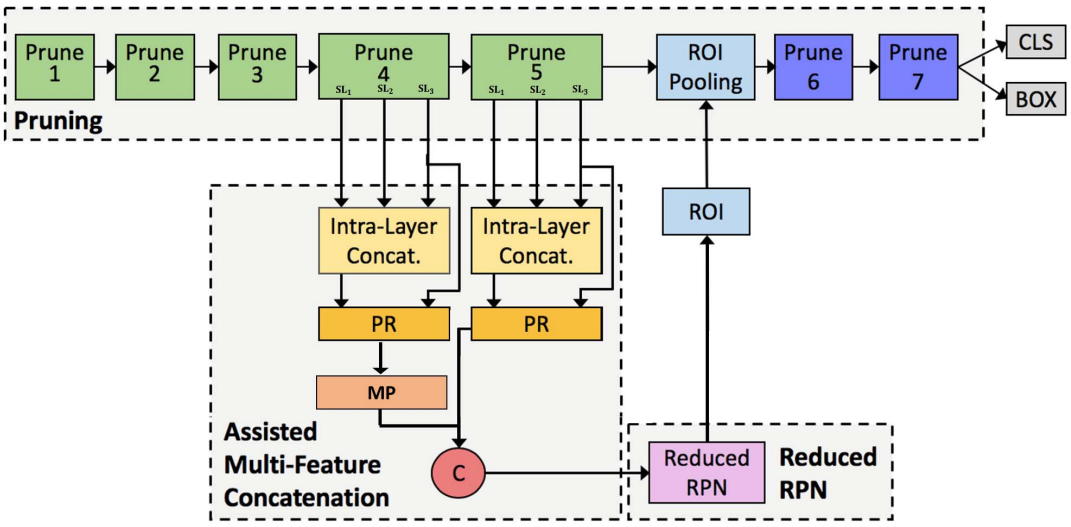

Fig. 1. Overall architecture, which includes pruning, reduced RPN, and assisted multi-feature concatenation. ML: Max-Pooling. SL_i: sub-layer i, where $i = 1, 2, 3$.

architecture predicts the classes and locations of objects directly, while the two-stage architecture finds ROI first and then performs the classifications on an ROI by the ROI basis. In this paper, we focus on two-stage object detection.

C. Series of R-CNN

The first CNN-based object detection study was R-CNN [14], in which a selective search [30] is used to find possible proposals. Convolution is used to extract the features, following which a representative vector was sent to the final SVM classifier [35]. However, this method is very time-consuming, as every proposal is processed by the entire CNN.

To improve upon the speed performance, Fast R-CNN [16] was developed. Inspired by SPPNet [41], this approach is based on ROIPooling, in which the feature maps are shared after convolution. Therefore, only one convolution is processed in the entire detection model. The problem with fast R-CNN is that the proposals are generated by a traditional time-consuming selective search.

Faster R-CNN [15] was proposed to further improve upon Fast R-CNN. In Faster R-CNN, an RPN is used to replace the selective search. The proposal task is converted into a binary problem, in which the network predicts if an object is present in a box. Therefore, in this approach, a small sub-network, the RPN, is used to solve the problem. Subsequently, the generated proposals are processed using Fast R-CNN. The important point is that the inputs to the RPN comprise the feature maps of convolution, so there is slightly more loading involved when using the RPN.

In terms of the inference speed, there is a great improvement from R-CNN to Faster R-CNN. Fast R-CNN is 146 times faster than R-CNN, and Faster R-CNN is 1460 times faster than R-CNN.

Recently, several studies have proposed techniques to increase the accuracy of Faster R-CNN. Instead of using VGG-16 as a backbone network for Faster R-CNN, adoption of different backbone networks, such as ResNet and Inception ResNet, has been proposed. He *et al.* [3] proposed the use of a deep residual network, such as ResNet-101, for image recognition. The authors showed that ResNet-101 has a lower complexity compared to VGG-16 and achieves good accuracy. Lin *et al.* [17] proposed using a feature pyramid network (FPN) for faster-RCNN. With feature sharing, the FPN-based Faster R-CNN system achieved better accuracy than [3]. Dai *et al.* [18] proposed the use of a Region-based Fully Convolution Network (R-FCN) based on positive-sensitive cropping to reduce the number of ROIs per image. R-FCN achieved comparable accuracy with a speed that was slighter higher than that of ResNet-101 [17]. Huang *et al.* [19] used an Inception ResNet v2 in the backbone of the Faster R-CNN to achieve better accuracy than that obtained using ResNet 101 with a slightly lower running time per frame. Shrivastava *et al.* [20] proposed a top–down modulation (TDM) network to incorporate fine details in the detection network for detecting small objects. They achieved higher accuracy compared to [19] with a slightly higher frame rate. Yauan *et al.* [42] proposed two refinement methods, iterative and LSTM refinement, for the Faster R-CNN model and improved the accuracy. Most of the above-mentioned works focus on increasing accuracy instead of improving the frame rate. Shih *et al.* [43] proposed a method to both accelerate the frame rate and improve the detection accuracy.

III. Proposed Architecture

There are three primary modules in the proposed architecture: pruning, reduced RPN, and assisted multi-feature concatenation. Fig. 1 shows the overall framework. Prune [1–5] in the green region denotes pruning in convolution layers while prune [6 and 7] in the blue region represents pruning in FC layers.

TABLE I
DIMENSION OF EACH LAYER IN ZF-NET AND CORRESPONDING NUMBER OF PARAMETERS AND AMOUNT OF COMPUTATION. C_{in}: CHANNEL$_{\text{in}}$ AND C_{out}: CHANNEL$_{\text{out}}$

ZF-Net Layer	X	Y	kernel _size	C_{in}	C_{out}	# parameter (M)		# computation (GFLOPs)	
conv1	300	500	7	3	96	0.014	0%	2.117	6%
conv2	76	126	5	96	256	0.614	1%	5.883	17%
conv3	39	64	3	256	384	0.885	1%	2.208	7%
conv4	39	64	3	384	384	1.327	2%	3.312	10%
conv5	39	64	3	384	256	0.885	1%	2.208	7%
fc6	1	1	6	256	4096	37.749	64%	11.325	34%
fc7	1	1	1	4096	4096	16.777	28%	5.033	15%
RPN	39	64	3	256	256	0.590	1%	1.472	4%
RPN_CLS	39	64	1	256	18	0.005	0%	0.012	0%
RPN_BOX	39	64	1	256	36	0.009	0%	0.023	0%
CLS	1	1	1	4096	21	0.086	0%	0.026	0%
BOX	1	1	1	4096	84	0.344	0%	0.103	0%
Total						59.825	100%	33.723	100%

TABLE II
DIMENSION OF EACH LAYER IN VGG16 AND CORRESPONDING NUMBER OF PARAMETERS AND AMOUNT OF COMPUTATION. C_{in}: CHANNEL$_{\text{in}}$ AND C_{out}: CHANNEL$_{\text{out}}$

VGG16 Layer	X	Y	kernel _size	C_{in}	C_{out}	# parameter (M)		# computation (GFLOPs)	
conv1_1	600	1000	3	3	64	0.002	0%	1.037	0%
conv1_2	600	1000	3	64	64	0.037	0%	22.118	10%
conv2_1	300	500	3	64	128	0.074	0%	11.059	5%
conv2_2	300	500	3	128	128	0.147	0%	22.118	10%
conv3_1	150	250	3	128	256	0.295	0%	11.059	5%
conv3_2	150	250	3	256	256	0.590	0%	22.118	10%
conv3_3	150	250	3	256	256	0.590	0%	22.118	10%
conv4_1	75	125	3	256	512	1.180	1%	11.059	5%
conv4_2	75	125	3	512	512	2.359	2%	22.118	10%
conv4_3	75	125	3	512	512	2.359	2%	22.118	10%
conv5_1	38	63	3	512	512	2.359	2%	5.648	3%
conv5_2	38	63	3	512	512	2.359	2%	5.648	3%
conv5_3	38	63	3	512	512	2.359	2%	5.648	3%
fc6	1	1	7	512	4096	102.760	75%	30.828	14%
fc7	1	1	1	4096	4096	16.777	12%	5.033	2%
RPN	38	63	3	512	512	2.359	2%	5.648	3%
RPN_CLS	38	63	1	512	18	0.009	0%	0.022	0%
RPN_BOX	38	63	1	512	36	0.018	0%	0.044	0%
CLS	1	1	1	4096	21	0.086	0%	0.026	0%
BOX	1	1	1	4096	84	0.344	0%	0.103	0%
Total						137.065	100%	225.574	100%

A. Pruning

We adopt filter pruning to remove less important weights in the network, including the convolutional and FC layers. The required number of parameters and the amount of computation in the network are analyzed by the following equations:

$$\#\text{parameter} = \text{kernel\_size}^2 \times \text{channel}_{\text{in}} \times \text{channel}_{\text{out}} \quad (1)$$

$$\#\text{computation} = \#\text{parameter} \times X \times Y \times \text{batch\_size} \quad (2)$$

where the kernel_size, channel$_{\text{in}}$, and channel$_{\text{out}}$ are defined as the kernel size of the weight filter, number of input channels, and number of output channels in each convolution layer. X and Y represent the horizontal and vertical dimensions of the feature maps in each convolution layer. batch_size denotes the number of training images in each iteration. Because the neurons in both FC and convolutional layers compute dot products, their functional form is identical. Therefore, (1) and (2) can be used as the FC layer for computational complexity analysis by doing FC to convolution conversion. For example, in VGG16-based Faster R-CNN, the first FC layer has channel$_{\text{out}}$ as 4096. Its input volume of size $7 \times 7 \times 512$ can be equivalently expressed as a CONV layer with kernel_size of 7, input channel number channel$_{\text{in}}$ of 512, output feature map of size 1×1, and output channel number channel$_{\text{out}}$ of 4096. Tables I and II show the analytical results of ZF-Net and VGG16, respectively. The batch_size is set to one in the analysis.

On analyzing Tables I and II, it is observed that FC layers dominate the model size, while convolutional layers dominate the amount of computation. In (2), the amount of computation is proportional to the number of parameters. Therefore, reducing the number of parameters in both convolution and FC layers helps achieve the goal. There are various strategies to decrease the model size, such as pruning [27] and quantization [44]. Pruning is a widely used method to compress a model, and it is parallel to quantization. However, quantization requires customized hardware [45] support during runtime; otherwise, it will only benefit storage. In this paper, we use pruning to decrease the model size and speed up the inference process.

1) Convolution: The characteristic feature of Faster R-CNN [15] in comparison with the previous work is the introduction of the ROIPooling operation. ROIPooling is placed after several convolutional layers and pooling layers, which allows it to generate discriminative and fixed-size features for each ROI. To achieve this, sufficiently large feature maps are required. Otherwise, there will be no detectable difference between two differently sized ROIs in the same location. Therefore, in Faster R-CNN, the input image size is typically enlarged to 600×1000. Larger feature maps greatly increase the amount of computation.

We use the pruning method proposed in [27] to prune our model. Both models are compressed by approximately 50% with respect to the convolutional part.

2) FC Layers: All the hidden neurons are weighted to add together for each element in the output in the FC layers. The number of parameters in the FC layers is more than that in the convolutional layers. To compress the model size, decreasing the size of the FC layers is essential. Another important item is the batch size of the FC layers in Faster R-CNN. Considering different input image sizes, Faster R-CNN accepts only one image as input, resulting in a batch size of 1. However, after generating ROIs, every ROI has to pass through the FC layers to obtain the final prediction. The batch size in FC layers is, thus, multiplied by the number of ROIs. In our experiment, the number of ROIs in the inference phase is set to 300, so having to perform 300 times the number of computations in a FC layer affects the inference speed.

To alleviate the inference time, we adopt the absolute sum method in [24]. Smaller summation values mean less

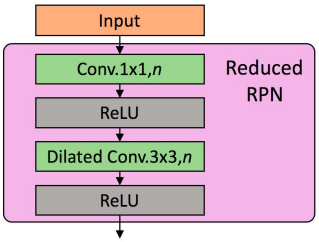

Fig. 2. Architecture of reduced RPN.

significant rows of weights. First, we sum up all the weights belonging to a particular hidden unit and then sort the values. Finally, we eliminate the kth smallest weights, where k depends on the target pruning ratio. In our model, the number of channels in the sixth FC layer (FC6) is reduced from 4096 to 854 and 1024 for ZF-Net and VGG16, respectively, and the compression rates of including the sixth and seventh FC layers for ZF-Net and VGG16 are 21% and 25%, respectively.

B. Reduced Region Proposal Network

In comparison to Fast R-CNN [16], the key component to improve the speed in Faster R-CNN [15] is RPN. The design of the RPN makes it possible to leverage the power of deep learning to find ROIs. In this case, we can solve the ROI task and detection task using the same network. This results in the Faster R-CNN method being ten times faster than Fast R-CNN. However, the Faster R-CNN model will find it impossible to complete the object detection task correctly without accurate ROIs. Hence, we can intuitively conclude that improving RPN can improve the overall accuracy. A reduction in the accuracy is inevitable after pruning; therefore, we propose using a reduced RPN to regain accuracy. There are two parts that we can modify in the RPN to achieve this: the architecture and the input. The reduced RPN is categorized as an architectural modification that is described in this section, and the reduced RPN architecture is shown in Fig. 2.

1) Compression: To compress the PRN, we reduce the number of channels in the RPN convolution layer to decrease the number of parameters in the subnet. Although we decrease the number of channels in the RPN convolution, there is still some chance for further improvement. Decreasing the input feature channels can further reduce the number of parameters in the subsequent convolutional layer. GoogLenNet [46] and ResNet [3] use a 1×1 kernel to reduce the number of parameters without losing accuracy while also gaining efficiency. The advantage of applying 1×1 convolution is in changing the number of channels. In short, we can use the simpler convolution to adjust the channel dimensions of feature maps. We set the same number of channels in the 1×1 convolution and the original convolution in RPN to make the architecture simple. From the experimental results, we find that the channel number of 64 is sufficient for obtaining good ROIs. For the backbone architecture of ZF-Net and VGG16, the reduced RPN uses only 3% and 9% of the number of parameters compared to the original RPN, respectively. Table III shows the comparison results.

TABLE III
COMPARISON OF NUMBER OF PARAMETERS AND CHANNELS USED IN RPN AND REDUCED RPN

# parameters	RPN	Reduced RPN	**Compression**	Channel (Orig/Reduced)
ZF-Net	590k	53k	**9%**	256/64
VGG16	2359k	70k	**3%**	256/64

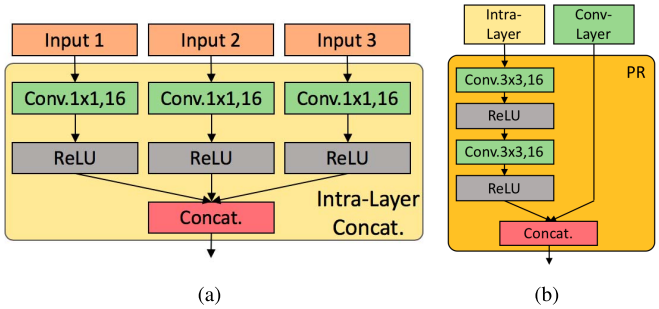

Fig. 3. Two concatenation blocks of (a) intra-layer concatenation and (b) proposal refinement.

2) Dilated Convolution: In [47] and [48], dilated convolution was proposed in the context of semantic segmentation. Using dilated convolution does not increase the number of parameters and only needs the pre-trained model to be fine-tuned. It does not increase the computation because the zero operations are skipped. However, the skipping operation is expensive in hardware implementation. The task of reduced RPN is to find bounding boxes for ROIs. While the details inside the box are unimportant, whether or not all the pixels inside the box belong to the same object is important. In the reduced RPN, the objective is to find a box that exactly surrounds an object; otherwise, irrelative features may interfere with the detection result. Dilated convolution is able to increase the receptive field. The larger receptive field helps a model to see a bigger picture and recognize the boundary of each object. Appropriately leveraged, more confined bounding boxes can be produced. To further increase the accuracy of bounding box prediction, we adopt the continuous dilated convolution in [47]. Continuous dilated convolution means that dilated kernels are applied in the previous convolutional layers. In this design, we extend the dilated convolution from reducing RPN to convolution layers, conv4 and conv5.

C. Assisted Multi-Feature Concatenation

In the original Faster R-CNN [15], the input to the RPN comprises of the feature maps from the last convolutional layer in the backbone network. The assisted multi-feature concatenation is proposed to provide reduced RPN with more specialized and informative feature maps. Fig. 3 shows the entire assisted multi-feature concatenation framework and its components. From Fig. 1, it is observed that the assisted multi-feature concatenation is only applied on the conv4 and conv5 blocks. In VGG16-based Faster R-CNN, sub-layers 1, 2, and 3 in both conv4 and conv5 are used in assisted multi-feature concatenation, as shown in Table II.

TABLE IV
COMPRESSION RATE, COMPUTATION RATIO, AND ACCURACY FOR ZF-NET AND VGG16 AFTER PRUNING

	Compression		Computation		mAP	
	ZF-net	VGG16	ZF-net	VGG16	ZF-net	VGG16
Baseline [15]	100%	100%	100%	100%	59.9%	68.7%
CONV pruning	81%	95%	70%	37%	58.6%	66.9%
CONV + FC Pruning	21%	29%	38%	25%	58.1%	66.5%

TABLE V
COMPRESSION RATE, COMPUTATION RATIO, AND ACCURACY FOR ZF-NET AND VGG16 ADDING ALL MODULES

	Compression		Computation		mAP	
	ZF-net	VGG16	ZF-net	VGG16	ZF-net	VGG16
Baseline [15]	100%	100%	100%	100%	59.9%	68.7%
Pruning	21%	29%	38%	25%	58.1%	66.5%
Pruning + Reduced RPN	20%	27%	34%	23%	59.3%	67.8%
Pruning + Reduced RPN + Multi-Feature	20%	27%	34%	23%	60.2%	69.1%
(Reduced RPN + Multi-Feature) No pruning	99%	98%	97%	96%	61.5%	69.4%

TABLE VI
COMPARING DIFFERENT FC6 CHANNEL DEPTHS

	Compression		Computation		mAP	
	ZF-net	VGG16	ZF-net	VGG16	ZF-net	VGG16
4096 (No Pruning)	81%	95%	70%	37%	58.6%	66.9%
2048	43%	51%	49%	29%	58.5%	67.3%
854	21%	29%	38%	25%	58.1%	66.5%

TABLE VII
COMPARING DIFFERENT CONVOLUTION CHANNEL DEPTHS IN REDUCED RPN. NC: NO CHANGE

	Compression		Computation		mAP	
	ZF-net	VGG16	ZF-net	VGG16	ZF-net	VGG16
512	-	29% (NC)	-	35% (NC)	-	66.5% (NC)
256	21% (NC)	28%	38% (NC)	35%	58.1% (NC)	66.5%
128	20%	28%	35%	23%	58.6%	66.8%
64	20%	27%	34%	23%	58.4%	66.9%

TABLE VIII
COMPARING WITH AND WITHOUT CONTINUOUS DILATED CONVOLUTIONS

	Compression		Computation		mAP	
	ZF-net	VGG16	ZF-net	VGG16	ZF-net	VGG16
w/o Continuous	20%	27%	34%	23%	58.4%	66.9%
w/ Continuous	20%	27%	34%	23%	59.3%	67.8%

The setting is based on experimental results, and more details are presented in Section IV-C. Assisted multi-feature concatenation is composed of two concatenation operations: intra-layer concatenation and proposal refinement. These components are described in detail in the following.

1) Concatenation: On examining the feature maps after the reduced RPN, we find that there are two types of feature maps, namely, shape and character. A few studies [49], [50] use summation to accumulate feature maps; however, to preserve feature maps independently, we replace the summation operator with a concatenation operator. For example, the ROIs need two types of features, such as shape and character features, for location predictions. In this way, the model can select the necessary channels automatically during the training phase, and we simply provide more informative feature maps. We perform max-pooling after conv4 to ensure that the size of the output feature map from cnonv4 is the same as that of the output feature map from cnonv5. Then, we concatenate them.

2) Intra-Layer Concatenation: Although we believe that the last layer has more dominant features, we expect a positive effect if we take advantage of all the feature maps in each layer. Therefore, we use a strategy similar to that followed in DenseNet [51] and name it as the intra-layer concatenation. The three convolution layers in conv4 and conv5 within the same convolutional block are concatenated together, as shown in Fig. 3(a). For example, in conv5 block, we concatenate conv5_1, conv5_2, and conv5_3 together. However, the number of output channels would be large if we directly concatenate all feature maps together. A large number of channels would result in computational issues, which would affect the inference speed. The 1×1 convolution is used for decreasing number of channels and aligning channel dimension, which is selected as 16. After two convolution layers, the concatenated result along with the last feature map (i.e., conv4_3 or conv5_3) are provided as inputs to the proposal refinement block.

3) Proposal Refinement: The purpose of the proposal refinement is to further improve the prediction accuracy near a boundary as in [49]. As shown in Fig. 3(b), there are two 3×3 convolutional layers in the proposal refinement block, and the output of the previous intra-layer concatenation is the input to the first convolutional layer in the proposal refinement block. After intra-layer concatenation, the feature maps are very informative and the convolutional layers can process them further. In other words, feature maps must be specialized for the subsequent reduced RPN; therefore, we design the proposal refinement stage to convert feature maps into the ROI related data. Considering that the feature maps from the last convolutional layer (e.g., conv4_3 and conv5_3) have more dominant features, we combine the dominant feature maps with the refined feature maps at the end of the proposal refinement block. In addition, we use concatenation to combine them to preserve information independently. The outputs of the proposal refinement block from conv4 are downsampled using maximum pooling (MP) first, and then, they are concatenated with the outputs of the proposal refinement block from conv5 before sending to the reduced RPN, as shown in Fig. 1. Yang *et al.* proposed the use of scale-dependent pooling (SDP) and cascaded rejection classifiers (CRCs) that use different scale convolution features to eliminate negative object proposals in a cascaded manner [52]. There are a few differences between their method and our proposed method. First, their method is based on fast R-CNN; therefore, the ROI in each of the images is provided externally. In our method, the ROI is generated by the RPN. The number of ROI provided externally is usually less than that generated from the RPN

TABLE IX
AVERAGE PRECISION ON PASCAL VOC 2007

Method	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	persn	plant	sheep	sofa	train	tv	mAP
Baseline(ZF-Net)	65.0	72.5	57.9	45.4	33.6	65.2	74.1	71.4	38.4	63.0	62.4	66.9	76.3	65.8	66.7	31.3	58.5	50.9	71.6	61.0	59.9
Pruning + Reduced RPN(ZF-Net)	65.6	72.1	51.6	47.8	30.2	67.2	72.2	69.8	38.1	62.6	66.9	64.8	76.9	70.1	69.0	34.1	58.7	55.1	69.3	62.6	60.2
Baseline(VGG16)	69.0	78.6	67.8	56.6	52.1	75.0	79.5	79.7	48.8	75.7	65.7	76.9	80.9	73.6	76.4	43.0	67.7	64.6	73.8	68.2	68.7
Pruning + Reduced RPN(ZF-Net)(VGG16)	68.8	79.3	64.8	58.7	50.8	77.3	80.0	83.3	52.7	73.0	69.1	75.5	80.8	75.8	78.3	43.0	57.7	65.5	78.0	70.5	69.1

in the Faster R-CNN model. Second, although the purpose of the approach in [52] is similar to our proposed assisted multi-feature concatenation method, our method is based on the concatenation of features of different layers instead of cascading features layer-by-layer. Their method focuses on classifying the ROI, such that the SDP and CRC method need extra fully convolution layers. In the proposed approach, concatenation is used to preserve the features of different scales, so the convolution layers are used.

IV. EXPERIMENTAL RESULTS

In this section, we provide details of our experimental results and discuss ablation. In Section IV-C, we show how we find the optimal settings for our proposed method. We also discuss the impact of different settings. At the end of this section, we compare our results with other work.

A. Experimental Environment and Data Sets

The proposed method is implemented in Caffe [53] with a Python 2.7 interface. The CPU is Intel Core i7-7800X @ 3.5 GHz, the main memory is 32-GB DDR4 RAM, and the GPU is an NVIDIA GeForce GTX 1080.

In this paper, we verify our proposed method on the PASCAL VOC 2007 data set [54]. There are 5011 images in the training set and 4952 images in the testing set. The data set has 20 classes, such as vehicles and animals. Following the instructions, we used the mAP@0.5 metric to evaluate accuracy. We did not change the training set and testing set for cross validation. However, for each of the mAP values in the experiments, we ran the simulation approximately eight times and use the average mAP as the results.

In the experiments, we pre-trained and fine-tuned all backbone networks on ImageNet [55]. In the training phase, we set the training iteration as 70k. A learning rate of 0.001 for the first 50k iterations and 0.0001 for the remaining iterations is adopted. Referring to the original paper [15], the input image is resized, such that the length of the shorter side of the image is 600 pixels. Horizontal flipping is used for data augmentation.

B. Results

Pruning on convolutional layers can reduce the amount of computation but does not impact compression greatly. When we further perform pruning on the FC layers, the number of parameters dramatically decreases and the computation decreases. Table IV shows the compression rate, computation ratio, and corresponding accuracy for ZF-Net and VGG16. The compression rate is greatly affected with and without pruning on the FC layers, and there is some accuracy drop.

TABLE X
APPLYING PROPOSAL REFINEMENT ON DIFFERENT CONVOLUTIONAL BLOCKS (VGG16)

VGG16	Baseline [15]	w/o PR	PR at CONV5	PR at CONV4,5	PR at CONV3,4,5
mAP	68.7%	67.8%	67.6%	68.9%	66.5%

TABLE XI
APPLYING INTRA-LAYER CONCATENATION AND DIFFERENT DATA COMBINING OPERATIONS (VGG16)

VGG16	Baseline [15]	w/o Intra-Layer	Concat.	Sum
mAP	68.7%	68.9%	69.1%	68.1%

After pruning, we propose the use of the reduced RPN and assisted multi-feature concatenation to regain accuracy.

In the reduced RPN, we propose the use of 1×1 convolution, thinner convolution, and continuous dilated convolution. Among these modifications, 1×1 convolution and thinner convolution greatly reduce the number of parameters used in RPN, while continuous dilated convolution improves accuracy. In the assisted multi-feature concatenation, we use 1×1 convolution to ensure that there is no extra computational loading required by this module. Table V shows the compression rate, computation ratio, and corresponding accuracy for ZF-Net and VGG16. Compared with a scenario using pruning only and the baseline, using the reduced RPN and assisted multi-feature concatenation simultaneously can completely compensate for the accuracy loss after pruning. As shown in Table V, the reduced RPN and multi-feature concatenation without pruning can increase the detection rate. The accuracy is highest with a 2.9% increase over the baseline model for VGG-16. However, the compression is only by 2%.

C. Ablation

In this section, we discuss how we selected the settings for our proposed method. The settings we experimented include the pruning ratio of FC layers, channel dimension of the convolutional layer in reduced RPN, the influence of continuous dilated convolution, effective proposal refinement area, and use of different combining operations in the intra-layer concatenation.

1) Pruning Ratio of FC Layers: After pruning the convolutional layers, we further prune the FC layers to reduce the number of parameters. The FC layers directly connect to classifiers. Therefore, the pruning ratio of FC layers is carefully selected. For ZF-Net, we evaluated pruning ratios of 50% and 80%, while for VGG16, we evaluated pruning ratios of 50% and 75%. The results are listed in Table VI. Compared with a scenario with no pruning, accuracy drops with the largest pruning ratio; however, the accuracy difference is acceptable,

TABLE XII
COMPARISON WITH OTHER WORK ON PASCAL VOC 2007 DATA SET

Method	Backbone	Dataset Train	mAP	Compression (Comparison/original)	Computation (Comparison/original)	FPS
Faster R-CNN [15] Nvidia K40	VGG16	Pascal VOC 07	68.7%	100% (523 MB/523 MB)	100% (226 G/226 G)	5
Channel Pruning [27]	VGG16	Pascal VOC 07	66.9%	95% (495 MB/523 MB)	37% (84 G/226 G)	19
faster R-CNN + RPR [42]	VGG16	Pascal VOC 07	67.7% (+0.7%)**	200% (1046 MB/523 MB)	137% (309 G/226 G)	–
R-FCN [18] train on Nvidia TITAN X	ResNet-101	Pascal VOC 07+12	79.5%	38% (185 MB/523 MB)	43% (97.74 G/226 G)	12
CoupleNet [57] train on Nvidia TITAN X	ResNet-101	Pascal VOC 07+12	82.7%	74% (385 MB/523 MB)	44% (100 G/226 G)	8
ERPN [58] train on Nvidia TITAN X	VGG16	Pascal VOC 07+12	78.6%	-	-	6
SAN [59] train on Nvidia TITAN X	ResNet-101	Pascal VOC 07+12	76.5%	-	-	11
Our method NVIDIA GTX 1080	VGG16	Pascal VOC 07	69.1% (+2.6%)	27% (144 MB/523 MB)	23% (52 G/226 G)	27
Our method NVIDIA GTX 1080	VGG16	Pascal VOC 07+12	75.3%	27% (144 MB/523 MB)	23% (52 G/226 G)	27

** The author re-implemented Faster R-CNN with PyTorch [60], and the baseline is 67.0%.

and both the number of parameters and amount of computation are greatly reduced. Therefore, the channel dimensions that we used are 854 for ZF-Net and 1024 for VGG16.

2) Channel Dimension of the Convolutional Layer in Reduced RPN: One of our tasks in this paper is to compress the model size as much as possible. Therefore, we also compress the RPN even though the sub-network is relatively small. In traditional RPN, the channels of convolution in the RPN are set to 256 for ZF-Net and 512 for VGG16, but there is no experimental evidence showing whether the setting is optimized. Therefore, different channel dimensions are checked to determine whether it is possible to use fewer parameters. Table VII shows the experimental results. From this table, it is observed that we can further compress the entire model size by using a smaller convolution channel in RPN. In terms of accuracy, both ZF-Net and VGG16 achieve better performance when using a depth of 64 rather than their original settings. The sub-network is small, and the task performed by the RPN is not difficult, but overfitting [56] might occur. Any overfitting issue can be addressed by using fewer parameters; this also helps to reduce the model size. In our proposed method, the channels of convolution in the reduced RPN are set to 64 for both ZF-Net and VGG16. In the reduced RPN, we add an extra layer with an extra non-linearity that is not present in the baseline. This could explain the mAP difference.

3) Influence of Continuous Dilated Convolution: In the reduced RPN, dilated convolution is used to enlarge the receptive field. Hence, we can include more information from other areas to help recognize the contours of objects. Dilated convolution is widely used in semantic segmentation tasks, and it is usually used continuously. The receptive field can be accumulated, and it grows exponentially with continuous dilated convolution. Here, we show the results with continuous dilated convolution in Table VIII for ZF-Net and VGG16. In the experiments, "w/o continuous" denotes that dilated convolution is only applied on reduced RPN, while "w/ continuous" signifies that we apply the dilated convolution on both the reduced RPN and convolutional layers in the feature extractor (i.e., conv4_3 and conv5_3). The results show that using continuous dilated convolution obviously helps to improve accuracy, and from the compression and computation aspects, there is no extra storage space or computation required.

4) Effective Proposal Refinement Area: Proposal refinement is proposed to refine the feature maps to features that are more suitable for finding ROIs. Furthermore, we wanted to utilize feature maps as much as possible; therefore, we try to connect the proposal refinement to multiple convolutional blocks in the feature extractor. Table IX shows our experimental results, and it is observed that connecting CONV4 and CONV5 produces the best result. It is surprising that the result becomes worse if the proposal refinement is connected to CONV3. A possible reason for this could be that the feature extractor is pruned, so some of the information is deleted. Therefore, in accordance with the findings from Table IX, we only connect our proposed assisted multi-feature concatenation to the CONV4 and CONV5 blocks.

5) Different Combining Operations in Intra-Layer Concatenation: There are several convolutional layers in a convolutional block in VGG16, and we wanted to collect information from all these feature maps. In the experimental result shown in Table X, we first compare scenarios with and without intra-layer concatenation and then compare the two operations used to combine data together (summation and concatenation). The results show that intra-layer concatenation can slightly improve the accuracy, and the result agrees with the fact that the last convolution in the block involves the most dominant features. On comparing different operations, it is found that summation yields a lower accuracy. The result helps to prove the necessity of keeping feature maps independent.

D. Comparison With Other Work

In this section, we compare our results with other models, focusing on the works on compression or RPN improvement. First, Table XI shows the results from comparing our proposed method with the original Faster R-CNN [15] for both ZF-Net and VGG16.

In Table XII, results from our experiment and other works are compared. Reference [42] used extra FC layers to refine RPN iteratively; therefore, many more parameters and computations are needed. Our pruning method further compresses the FC layers, so a significantly better compression rate compared with [27] can be achieved. Using the three proposed modules, all of the lost accuracies can be regained, and 2.6% and 2.1% improvements are achieved over our initial pruning results for VGG16 and ZF-Net, respectively.

CoupleNet [57] is similar to R-FCN [18] in that both adopt the ResNet 101 as backbone and position-sensitive ROI (PSROI) pooling to extract local information of bounding boxes. CoupleNet adds a global FCN to encode global and context information. Therefore, their accuracy is higher than that of R-FCN with a slightly higher number of parameters and computation. EFPN [58] uses a deconvolutional FPN (DFPN) to increase the resolution of the higher feature maps and then integrates it with the lower feature maps. Therefore, the resolution of the resulting feature maps is increased, and the detection rate is higher. The number of parameters in the last DFPN is less than that in the RPN. Therefore, the speed is slighted accelerated compared with Faster R-CNN. Yan *et al.* [59] proposed the spatial alignment network (SAN), in which the RoIs and scores are integrated into the form of the default boxes to reduce redundant boxes. They do not use the RoI-Pooling layer; this reduces the computational repeatability of the second stage in the Faster R-CNN.

V. Conclusion

In this paper, we proposed using a reduced RPN and assisted multi-feature concatenation. Using the proposed modules, we were able to compress the number of parameters of ZF-Net by 81.3% and that of VGG16 by 73%. With respect to the computational speed, ZF-Net was accelerated from 25 to 40 frames/s, and VGG16 was accelerated from 9 to 27 frames/s with image sizes of 600×1000. Furthermore, detection accuracy improved 2.2%–60.2% for ZF-Net and 2.6%–69.1% for VGG16. Our proposed architecture can be applied for real-time computer visions' applications.

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.

[2] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," Sep. 2014, *arXiv:1409.1556*. [Online]. Available: https://arxiv.org/abs/1409.1556

[3] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 770–778.

[4] W. Zhiqiang and L. Jun, "A review of object detection based on convolutional neural network," in *Proc. 36th Chin. Control Conf. (CCC)*, Jul. 2017, pp. 11104–11109.

[5] Z. Zhu, D. Liang, S. Zhang, X. Huang, B. Li, and S. Hu, "Traffic-sign detection and classification in the wild," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 2110–2118.

[6] W. Ouyang and X. Wang, "Joint deep learning for pedestrian detection," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2013, pp. 2056–2063.

[7] B. Wu, F. N. Iandola, P. H. Jin, and K. Keutzer, "SqueezeDet: Unified, small, low power fully convolutional neural networks for real-time object detection for autonomous driving," in *Proc. CVPR Workshops*, Jul. 2017, pp. 129–137.

[8] J. Park, D. H. Kim, Y. S. Shin, and S.-H. Lee, "A comparison of convolutional object detectors for real-time drone tracking using a PTZ camera," in *Proc. 17th Int. Conf. Control, Automat. Syst. (ICCAS)*, Oct. 2017, pp. 696–699.

[9] G. V. Konoplich, E. O. Putin, and A. A. Filchenkov, "Application of deep learning to the problem of vehicle detection in UAV images," in *Proc. 19th IEEE Int. Conf. Soft Comput. Meas. (SCM)*, May 2016, pp. 4–6.

[10] S. Akcay, M. E. Kundegorski, C. G. Willcocks, and T. P. Breckon, "Using deep convolutional neural network architectures for object classification and detection within X-ray baggage security imagery," *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 9, pp. 2203–2215, Sep. 2018.

[11] S. Akcay and T. P. Breckon, "An evaluation of region based object detection strategies within X-ray baggage security imagery," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2017, pp. 1337–1341.

[12] S. O'Keeffe and R. Villing, "Evaluating pruned object detection networks for real-time robot vision," in *Proc. IEEE Int. Conf. Auton. Robot Syst. Competitions (ICARSC)*, Apr. 2018, pp. 91–96.

[13] D. Guo, T. Kong, F. Sun, and H. Liu, "Object discovery and grasp detection with a shared convolutional neural network," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, May 2016, pp. 2038–2043.

[14] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 580–587.

[15] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 91–99.

[16] R. Girshick, "Fast R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2015, pp. 1440–1448.

[17] T.-Y. Lin, P. Dollar, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proc. CVPR*, Jul. 2017, pp. 2117–2125.

[18] J. Dai, Y. Li, K. He, and J. Sun, "R-FCN: Object detection via region-based fully convolutional networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 379–387.

[19] J. Huang *et al.*, "Speed/accuracy trade-offs for modern convolutional object detectors," in *Proc. CVPR*, vol. 4, Jul. 2017, pp. 3296–3297.

[20] A. Shrivastava, R. Sukthankar, J. Malik, and A. Gupta, "Beyond skip connections: Top-down modulation for object detection," Dec. 2016, *arXiv:1612.06851*. [Online]. Available: https://arxiv.org/abs/1612.06851

[21] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Proc. Eur. Conf. Comput. Vis.* Basel, Switzerland: Springer, 2014, pp. 818–833.

[22] Y. LeCun, J. S. Denker, and S. A. Solla, "Optimal brain damage," in *Proc. Adv. Neural Inf. Process. Syst.*, 1990, pp. 598–605.

[23] B. Hassibi and D. G. Stork, "Second order derivatives for network pruning: Optimal brain surgeon," in *Proc. Adv. Neural Inf. Process. Syst.*, 1993, pp. 164–171.

[24] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 1135–1143.

[25] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient convNets," Aug. 2016, *arXiv:1608.08710*. [Online]. Available: https://arxiv.org/abs/1608.08710

[26] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz, "Pruning convolutional neural networks for resource efficient inference," Nov. 2016, *arXiv:1611.06440*. [Online]. Available: https://arxiv.org/abs/1611.06440

[27] Y. He, X. Zhang, and J. Sun, "Channel pruning for accelerating very deep neural networks," in *Proc. Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, vol. 2, no. 6, pp. 1389–1397.

[28] W. Wen, C. Wu, Y. Wang, Y. Chen, and H. Li, "Learning structured sparsity in deep neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 2074–2082.

[29] R. Tibshirani, "Regression shrinkage and selection via the lasso," *J. Roy. Stat. Soc. B, (Methodological)*, vol. 58, no. 1, pp. 267–288, 1996.

[30] J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, and A. W. M. Smeulders, "Selective search for object recognition," *Int. J. Comput. Vis.*, vol. 104, no. 2, pp. 154–171, Sep. 2013.

[31] C. L. Zitnick and P. Dollár, "Edge boxes: Locating object proposals from edges," in *Proc. Eur. Conf. Comput. Vis.* Basel, Switzerland: Springer, 2014, pp. 391–405.
[32] P. Arbelaez, J. Pont-Tuset, J. T. Barron, F. Marques, and J. Malik, "Multiscale combinatorial grouping," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 328–335.
[33] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, Jun. 2005, vol. 1, no. 1, pp. 886–893.
[34] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vis.*, vol. 60, no. 2, pp. 91–110, 2004.
[35] C. Cortes and V. Vapnik, "Support-vector networks," *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, 1995.
[36] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *J. Comput. Syst. Sci.*, vol. 55, no. 1, pp. 119–139, Aug. 1997.
[37] J. Redmon and A. Farhadi, "YOLO9000: Better, faster, stronger," in *Proc. CVPR*, Jul. 2017, pp. 7263–7271.
[38] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," Apr. 2018, *arXiv:1804.02767*. [Online]. Available: https://arxiv.org/abs/1804.02767
[39] W. Liu *et al.*, "SSD: Single shot multibox detector," in *Proc. Eur. Conf. Comput. Vis.* Amsterdam, The Netherlands: Springer, 2016, pp. 21–37.
[40] C.-Y. Fu, W. Liu, A. Ranga, A. Tyagi, and A. C. Berg, "DSSD : Deconvolutional single shot detector," Jan. 2017, *arXiv:1701.06659*. [Online]. Available: https://arxiv.org/abs/1701.06659
[41] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," in *Proc. Eur. Conf. Comput. Vis.* Basel, Switzerland: Springer, 2014, pp. 346–361.
[42] P. Yuan, Y. Zhong, and Y. Yuan, "Faster R-CNN with region proposal refinement," Comput. Sci. Dept., Stanford Univ., Tech. Rep., 2017.
[43] K.-H. Shih, C.-T. Chiu, and Y.-Y. Pu, "Real-time object detection via pruning and a concatenated multi-feature assisted region proposal network," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, May 2019, pp. 1398–1402.
[44] J. Wu, C. Leng, Y. Wang, Q. Hu, and J. Cheng, "Quantized convolutional neural networks for mobile devices," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 4820–4828.
[45] S. Sharify, A. D. Lascorz, K. Siu, P. Judd, and A. Moshovos, "Loom: Exploiting weight and activation precisions to accelerate convolutional neural networks," in *Proc. 55th Annu. Des. Autom. Conf.*, Jun. 2018, pp. 1–8.
[46] C. Szegedy *et al.*, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2015, pp. 1–9.
[47] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," *arXiv preprint arXiv:1511.07122*, 2015.
[48] P. Wang *et al.*, "Understanding convolution for semantic segmentation," Nov. 2015, *arXiv:1511.07122*. [Online]. Available: https://arxiv.org/abs/1511.07122
[49] C. Peng, X. Zhang, G. Yu, G. Luo, and J. Sun, "Large kernel matters—Improve semantic segmentation by global convolutional network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 1743–1751.
[50] G. Lin, A. Milan, C. Shen, and I. D. Reid, "RefineNet: Multi-path refinement networks for high-resolution semantic segmentation," in *Proc. CVPR*, Jul. 2017, vol. 1, no. 2, pp. 5168–5177.
[51] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jul. 2017, vol. 1, no. 2, pp. 2261–2269.
[52] F. Yang, W. Choi, and Y. Lin, "Exploit all the layers: Fast and accurate CNN object detector with scale dependent pooling and cascaded rejection classifiers," in *Proc. CVPR*, Jun. 2016, pp. 2129–2137.
[53] Y. Jia *et al.*, "Caffe: Convolutional architecture for fast feature embedding," Jun. 2014, *arXiv:1408.5093*. [Online]. Available: https://arxiv.org/abs/1408.5093
[54] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The Pascal visual object classes (VOC) challenge," *Int. J. Comput. Vis.*, vol. 88, no. 2, pp. 303–338, Sep. 2009.
[55] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 248–255.
[56] R. Caruana, S. Lawrence, and C. L. Giles, "Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping," in *Proc. Adv. Neural Inf. Process. Syst.*, 2001, pp. 402–408.
[57] Y. Zhu, C. Zhao, J. Wang, X. Zhao, Y. Wu, and H. Lu, "CoupleNet: Coupling global structure with local parts for object detection," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, vol. 2, Oct. 2017, pp. 4126–4134.
[58] Y. P. Chen, Y. Li, and G. Wang, "An enhanced region proposal network for object detection using deep learning method," *PLoS ONE*, vol. 13, no. 9, Sep. 2018, Art. no. e0203897.
[59] C. Yan, W. Chen, P. C. Y. Chen, A. S. Kendrick, and X. Wu, "A new two-stage object detection network without RoI-pooling," in *Proc. Chin. Control Decis. Conf. (CCDC)*, Jun. 2018, pp. 1680–1685.
[60] A. Paszke *et al.*, "Automatic differentiation in pytorch," in *Proc. NIPS Autodiff Workshop, Future Gradient-Based Mach. Learn. Softw. Techn.*, Long Beach, CA, USA, 2017.

Kuan-Hung Shih received the B.S. degree from the Department of Chemistry Engineering, National Taiwan University, Taipei, Taiwan, in 2009, and the M.S degree from the Institute of Computer Science, National Tsing Hua University, Hsinchu, Taiwan, in 2018, where he is currently pursuing the Ph.D. degree with the Department of Computer Science.

His current research interests include deep learning, object detection, and computer vision. His research interest will focus on deep learning and 3-D object detection.

Ching-Te Chiu received the B.S. and M.S. degrees from National Taiwan University, Taipei, Taiwan, and the Ph.D. degree from the University of Maryland, College Park, MD, USA, all in electrical engineering.

She is currently a Professor with the Computer Science Department and the Institute of Communications Engineering, National Tsing Hua University, Hsinchu, Taiwan. She was a member of the Technical Staff at AT&T, Lucent Technologies, Murry Hill, NJ, USA, and Agere Systems, Murry Hill, NJ, USA. Her current research interests include machine learning, pattern recognition, high dynamic range image and video processing, super-resolution, high-speed SerDes design, multi-chip interconnect, and fault tolerance for network-on-chip design.

Dr. Chiu is a TC Member of the IEEE Circuits and Systems Society, Nanoelectronics and Gigascale Systems Group. She won the First Prize Award, the Best Advisor Award, and the Best Innovation Award of the Golden Silicon Award in 2006. She was the Program Chair of the first IEEE Signal Processing Society Summer School, Hsinchu, in 2011, and the Technical Program Chair of the 2013 IEEE Workshop on Signal Processing System (SiPS). She currently serves as the Chair of IEEE Signal Processing Society, Design and Implementation of Signal Processing Systems (DISPS) TC. She served as an Associate Editor for the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS I, *IEEE Signal Processing Magazine*, and *Journal of Signal Processing Systems*.

Jiou-Ai Lin received the B.S. degree from the Department of Mathematics Science, National Changhua University of Education, Changhua, Taiwan, in 2017. She is currently a Graduate Student with the Institute of Communication Engineering, National Tsing Hua University, Hsinchu, Taiwan.

Her current research interests include object detection and machine learning.

Yen-Yu Bu received the B.S. degree from the Department of Electronic Engineering, Chung Yuan Christian University, Taoyuan, Taiwan, in 2018. He is currently a Graduate Student with the Institute of Communication Engineering, National Tsing Hua University, Hsinchu, Taiwan.

His current research interests include image processing, computer vision, signal processing, and machine learning.