REAL-TIME OBJECT DETECTION VIA PRUNING AND A CONCATENATED MULTI-FEATURE ASSISTED REGION PROPOSAL NETWORK

Kuan-Hung Shih, Ching-Te Chiu, Yen-Yu Pu

Department of Computer Science, National Tsing Hua University, Hsinchu, Taiwan

ABSTRACT

Object detection is an important research area in the field of computer vision. Its purpose is to find all objects in an image and recognize the class of each object. Since the development of deep learning, an increasing number of studies have applied deep learning in object detection and have achieved successful results. For object detection, there are two types of network architectures: one-stage and two-stage. This study is based on the widely-used two-stage architecture, called Faster R-CNN, and our goal is to improve the inference time to achieve real-time speed without losing accuracy.

First, we use pruning to reduce the number of parameters and the amount of computation, which is expected to reduce accuracy as a result. Therefore, we propose a multi-feature assisted region proposal network composed of assisted multi-feature concatenation and a reduced region proposal network to improve accuracy. Assisted multi-feature concatenation combines feature maps from different convolutional layers as inputs for a reduced region proposal network. With our proposed method, the network can find regions of interest (ROIs) more accurately. Thus, it compensates for loss of accuracy due to pruning. Finally, we use ZF-Net and VGG16 as backbones, and test the network on the PASCAL VOC 2007 dataset.

Index Terms— Object Detection, Convolutional Neuron Network Compression, Pruning, Region Proposal Network (RPN), Feature Maps Concatenation

1. INTRODUCTION

Object detection is an important field in computer vision, which is a key to understanding an image. With more and more convenient mobile devices, there are abundant images and video clips collected every day. It is valuable to obtain information from these data, but it is difficult for humans to process them efficiently. Therefore, we can utilize object detection to help us. Recently, deep learning has developed rapidly, especially with recent successes in image processing using convolutional neural networks (CNNs)[1][2][3]. Object detection can leverage the CNN to improve object recognition performance[4]. There are numerous applications relying on object detection techniques that are beyond the scope of our discussions.

The first CNN based object detection algorithm was R-CNN (Region-based Convolutional Neural Network)[5], and after some evolution, Faster R-CNN[6] has become one of the more popular object detection models. Faster R-CNN is a two-stage object detection method, where two-stage means finding regions of interest (ROIs) followed by ROI prediction. An important contribution in Faster R-CNN is the Region Proposal Network (RPN) to find all ROIs. Although Faster R-CNN is faster than previous R-CNN versions [5][7], it cannot achieve real-time performance. One of the reasons is that the RPN generates many ROIs, and every ROI has to go through various fully connected layers and classifiers. Another reason is large input image size. ROIPooling is an operation introduced in Fast R-CNN[7] that uses max pooling to crop and resize the feature maps for each ROI. Therefore, every ROI has its own representative feature map. However, after several pooling layers, the dimensions of feature maps become too small, so that two ROIs at the same location but with different sizes have similar feature maps. To prevent this issue, the input image size has to be large enough, where image size is set to 600×1000 in general.

Our work aims to solve the speed issue to allow Faster R-CNN to achieve real-time performance. Modern networks [1][2] use abundant parameters, which require not only excessive storage space but increased computation, so we evaluate our method on these types of networks. Intuitively, we can use pruning to reduce the number of parameters. Therefore, we can solve the storage and computation issues simultaneously. However, pruning means deleting some filters, which implies information loss. Hence, it is not surprising that the accuracy drops. With our proposed model, we can have the object detection system lightweight and accurate.

2. RELATED WORK

2.1. Pruning

Many parameters in the CNN model are thought to be redundant, and we can safely remove their associated weights. There are many researches studies on how to find the candidates to prune, ensuring model size and the required computation can be reduced.

Early pruning works focus on removing connections. Optimal Brain Damage[8] use the Taylor expansion to rank the

importance of weights, and [9] uses absolute values to evaluate importance. Recently, researchers prune entire filters, and there are two types of methods: data-independent, and data-dependent. Data-independent methods directly analyze values of filters, for instance, [10] sums up the absolute value of all weights in a filter and ranks its importance. In contrast, data-dependent methods analyze activation values. [10] uses activation means, and [11] uses the Taylor expansion to evaluate activation. [12] and [13] consider interactions between filters, and they use LASSO regression to find candidates.

2.2. Object Detection

Since the development of deep learning, more studies have applied deep learning in object detection and have achieved successful results. For object detection, there are two types of network architectures: one-stage, and two-stage. The one-stage architecture assumes that objects are located everywhere, and it directly predicts the classes of objects. Comparing the two architectures, the one-stage architecture directly predicts all results, while the two-stage architecture predicts results on an ROI–by-ROI basis. Therefore, one-stage is faster than two-stage; however, two-stage is more accurate than one-stage because the ROIs are carefully selected.

3. PROPOSED ARCHITECTURE

In our proposed method, there are three major blocks: pruning, reduced RPN, and assisted multi-feature concatenation. The overall architecture is shown in Figure 1, and each block will be discussed in the following sections.

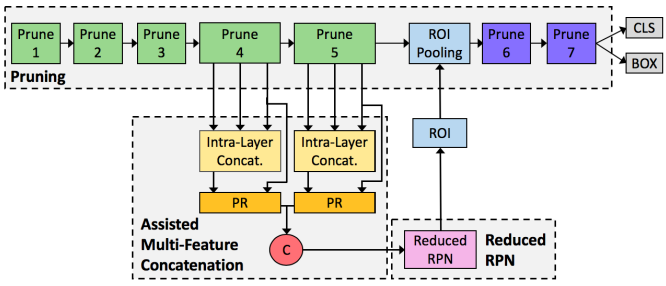

Fig. 1: Overall architecture, which includes pruning, reduced RPN, and assisted multi-feature concatenation.

3.1. Pruning

Our work aims to make the model small and fast. First, we analyze both the number of parameters and the amount of computation. Reducing the number of parameters in both convolution and fully connected layers helps to achieve the goal. In this work, we use pruning to decrease model size and speed up inferencing.

3.1.1. *Convolution*

Owing to larger input feature maps, shallower convolution contributes most in terms of computation. It is more efficient to prune at shallower convolutional layers than deeper convolutional layers. We use the pruning method proposed in [12] to prune our model. Both models are compressed by approximately 50% regarding the convolutional part.

3.1.2. *Fully Connected Layers*

A fully connected layer connects all hidden units together. Therefore, there are many more parameters in fully connected layers than in convolutional layers. To mitigate inference computation, we refer to [9] to prune fully connected layers. We consider that weights with small values have less importance. The sixth fully connected layer (FC6) is pruned to 854 and 1024 for ZF-Net and VGG16, respectively, and the compression rate of all fully connected layers for ZF-Net and VGG16 are 21% and 25%, respectively.

3.2. Reduced Region Proposal Network

RPN is the most significant improvement from Fast R-CNN[7] to Faster R-CNN[6]. When we study the RPN, we find that RPN is a simple sub-network. Although the sub-network is very simple, it greatly influences accuracy. In both the training phase and testing phase, the detection model relies on the RPN to generate ROIs. If the RPN cannot generate ROIs accurately, the detection model will receive many negative samples. Consequently, there are no useful samples fed to the detection model, and it will have difficulty converging during training. The reduced RPN is shown in Figure 2.

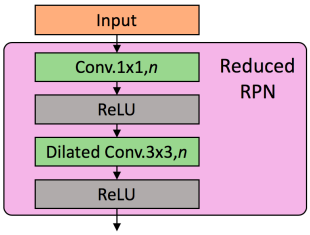

Fig. 2: Architecture of reduced RPN.

3.2.1. *Compression*

Our work is to accelerate and compress the model in object detection as much as possible, so we also try to make RPN smaller. There is only one convolution in RPN and the number of channels is originally set to 256 and 512 for ZF-Net and VGG16, respectively. However, whether it is necessary to use the abundant parameters in RPN has not been verified, and we found that using 64 channels does not harm the accuracy but does compress the RPN several times. We add a convolutional layer with 1×1 kernel size at the beginning of the reduced

RPN to reduce the input feature channels. To simplify the architecture, we make the two convolutional layers in reduced RPN have a consistent output channel dimension, which is set to 64 in our proposed architecture. With the compression strategy we proposed, the reduced RPN uses less than 10% the number of parameters compared to original RPN, and the results are shown in Table 1.

Table 1: Comparison of number of parameters between RPN and Reduced RPN.

# parameters	RPN	Reduced RPN	**Compression**
ZF-Net	590k	53k	**9%**
VGG16	2359k	70k	**3%**

3.2.2. Dilated Convolution

Dilated convolution is proposed in [14] and [15], which focus on semantic segmentation. Reduced RPN is responsible for predicting an ROI location, which is described by a bounding box. Looking into the purpose of finding ROIs, it is similar to the boundary problem in semantic segmentation. One of the advantages of dilated convolution is increasing the receptive field, as having a larger receptive field benefits in recognizing the boundary of each object. Appropriately leveraged, this advantage helps to generate more accurate ROIs. In [14], the authors mention that continuous dilated convolution enhances accuracy. Therefore, we also apply dilated convolution on the convolutional layer before the reduced RPN.

3.3. Assisted Multi-Feature Concatenation

In the original Faster R-CNN[6], the input of the RPN are the feature maps from the last convolutional layer in the backbone network. However, we believe more informative feature maps can help reduced RPN to recognize ROIs more accurately. Therefore, we design assisted multi-feature concatenation to provide reduced RPN with more specialized feature maps. Figure 3 shows the entire assisted multi-feature concatenation framework and its components. Assisted multi-feature concatenation is composed of several concatenation operations: intra-layer concatenation and proposal refinement. The following sections describe all components.

3.3.1. Concatenation

Regarding the feature maps after reduced RPN, we find that there are two types of feature maps, shape and character. Some works [16][17] use summation to accumulate feature maps, but, considering preserving feature maps independently, we replace summation with concatenation. At the same time, concatenation is also more flexible than summation because it only requires that two inputs have identical width and height. For these reasons, we use concatenation in our work when combining of data is needed.

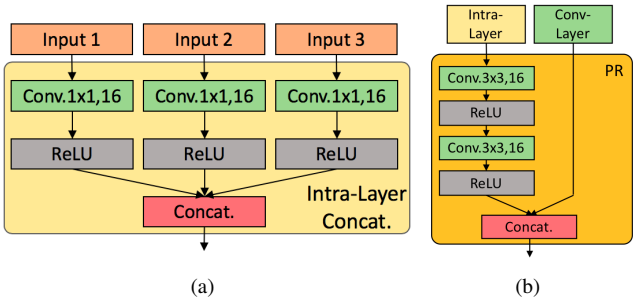

Fig. 3: Diagram of (a) Intra-Layer Concatenation and (b) Proposal Refinement.

3.3.2. Intra-Layer Concatenation

In VGG16, there are five convolutional blocks in the entire network, and each block is composed of two or three convolutional layers, which are identical but output different characteristics. Although we believe the last layer has more dominant features, we still expect a positive effect if we take advantage of all feature maps of each layer. As shown in Figure 3(a), we concatenate all feature maps together within the same convolutional block. However, if we directly concatenate all feature maps together, the channels of the output would be very large. To solve this problem, we apply a convolutional layer with kernel size 1 first, and the channel dimension of the output feature maps is adjusted to 16. We then concatenate the three sets of feature maps together, and the output of the intra-layer concatenation is the input of the proposal refinement block.

3.3.3. Proposal Refinement

The last part of assisted multi-feature concatenation is the proposal refinement. In [16], boundary refinement is proposed to improve prediction accuracy near a boundary. We use the same concept to propose a proposal refinement block to help improve reduced RPN. As shown in Figure 3(b), there are two convolutional layers in the proposal refinement block, and the output of the previous intra-layer concatenation is the input of these two convolutional layers. Because our goal is to compress the object detection model, we set the channel dimension of the two convolutional layers in proposal refinement to 16, ensuring there is no great difference in the amount of computation. At the end of proposal refinement block, considering that the feature maps from the last convolutional layer have more dominant features, we combine the dominant feature maps with the refined feature maps. Finally, all outputs of the proposal refinement block are concatenated and sent to the reduced RPN to generate ROIs.

Table 2: Comparison with other work on PASCAL VOC 2007 dataset.

Method	mAP	Compression	Computation	FPS	Backbone
[6] (original)	68.7%	100% (523 MB)	100%	9	VGG16
[6] (original)	59.9%	100% (227 MB)	100%	25	ZF-Net
[12]	66.9%	95% (495 MB)	37%	19	VGG16
[18]	67.7% (+0.7%)**	200%	136%	–	VGG16
Our Pruning	66.5%	29% (55 MB)	25%	26	VGG16
Our Pruning	58.1%	21% (48 MB)	38%	39	ZF-Net
Proposed	69.1% (+2.6%)	27% (144 MB)	23%	27	VGG16
Proposed	60.2% (+2.1%)	20% (45 MB)	34%	40	ZF-Net

** The author re-implemented Faster R-CNN with PyTorch[19], and the baseline is 67.0%.

4. EXPERIMENTAL RESULTS

4.1. Environment and Datasets

The proposed method is implemented in Caffe[20] 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 work, we verify our proposed method on the PASCAL VOC 2007 dataset[21]. Following the instructions, we use mAP@0.5 to evaluate accuracy. In this work, all backbone networks are pre-trained and fine-tuned on ImageNet[22]. In the training phase, the training iteration is set to 70k, and we use a learning rate of 0.001 for the first 50k iterations and 0.0001 for the remaining iterations. Referring to the original paper[6], the input image is resized such that the shorter side is 600 pixels.

4.2. Results

Table 3: Detection results of adding all modules (ZF-Net).

ZF-Net	Baseline[6]	Pruning	Pruning Reduced RPN	Pruning Reduced RPN Multi-Feature
Compression	100%	21%	20%	20%
Computation	100%	38%	34%	34%
mAP	59.9%	58.1%	59.3%	60.2%

Table 4: Detection results of adding all modules (VGG16).

VGG16	Baseline[6]	Pruning	Pruning Reduced RPN	Pruning Reduced RPN Multi-Feature
Compression	100%	29%	27%	27%
Computation	100%	25%	23%	23%
mAP	68.7%	66.5%	67.8%	69.1%

We propose three modules: pruning, reduced RPN, and assisted multi-feature concatenation. Pruning is used to compress model size and reduce the amount of computation, but it negatively affects accuracy. Additionally, pruning on convolutional layers and fully connected layers creates different results. After pruning, we propose the reduced RPN and assisted multi-feature concatenation to regain the accuracy.

Table 3 and Table 4 show the compression rate, computation ratio, and corresponding accuracy for ZF-Net and VGG16, respectively. Compared to pruning only and the baseline, using reduced RPN and assisted multi-feature concatenation at the same time can completely compensate for accuracy loss after pruning. With its comparable accuracy to the baseline, our model is significantly compressed, and the amount of computation is also reduced considerably.

4.3. Comparison with Other Work

In this section, we compare our results with other models, focusing on the work for compression purposes or improving RPN purposes. In Table 2, a comparison between our experimental results and other works is shown. [18] use extra fully connected layers to refine RPN iteratively; therefore, many more parameters and computations are needed. Our pruning method further prunes the fully connected layers, so we achieve a significantly better compression rate compared to [12]. With all of our proposed modules, all accuracy can be regained, and 2.6% and 2.1% improvements are achieved compared to our initial pruning results for VGG16 and ZF-Net, respectively. Compared to original Faster R-CNN, higher accuracy is achieved for both VGG16 and ZF-Net, verifying that our proposed method can really help RPN to find ROIs more accurately and quickly.

5. CONCLUSION

In this work, we first use a pruning technique on Faster R-CNN to reduce the number of parameters and computations. As expected, the accuracy drops; therefore, we propose reduced RPN and assisted multi-feature concatenation to regain the accuracy. With all of our proposed modules, we can compress ZF-Net from 227 MB to 48 MB, and compress VGG16 from 523 MB to 144 MB. Regarding speed, ZF-Net is accelerated from 25 fps to 40 fps, and VGG16 is accelerated from 9 fps to 27 fps. Furthermore, accuracy is improved from 58.1% to 60.2% and from 66.5% to 69.1% for ZF-Net and VGG16, respectively. Overall, we successfully achieve 40 fps with image sizes of 600×1000, which can be used for real-time applications. All proposed methods are tested on ZF-Net and VGG16, which are modern CNN architectures, and thus our method should be applicable to most CNN models.

6. REFERENCES

[1] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[2] Karen Simonyan and Andrew Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[4] Wang Zhiqiang and Liu Jun, "A review of object detection based on convolutional neural network," in *Control Conference (CCC), 2017 36th Chinese*. IEEE, 2017, pp. 11104–11109.

[5] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.

[6] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015, pp. 91–99.

[7] Ross Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.

[8] Yann LeCun, John S Denker, and Sara A Solla, "Optimal brain damage," in *Advances in neural information processing systems*, 1990, pp. 598–605.

[9] Song Han, Jeff Pool, John Tran, and William Dally, "Learning both weights and connections for efficient neural network," in *Advances in neural information processing systems*, 2015, pp. 1135–1143.

[10] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf, "Pruning filters for efficient convnets," *arXiv preprint arXiv:1608.08710*, 2016.

[11] Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz, "Pruning convolutional neural networks for resource efficient transfer learning," *CoRR, abs/1611.06440*, 2016.

[12] Yihui He, Xiangyu Zhang, and Jian Sun, "Channel pruning for accelerating very deep neural networks," in *International Conference on Computer Vision (ICCV)*, 2017, vol. 2.

[13] Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li, "Learning structured sparsity in deep neural networks," in *Advances in Neural Information Processing Systems*, 2016, pp. 2074–2082.

[14] Fisher Yu and Vladlen Koltun, "Multi-scale context aggregation by dilated convolutions," *arXiv preprint arXiv:1511.07122*, 2015.

[15] Panqu Wang, Pengfei Chen, Ye Yuan, Ding Liu, Zehua Huang, Xiaodi Hou, and Garrison Cottrell, "Understanding convolution for semantic segmentation," *arXiv preprint arXiv:1702.08502*, 2017.

[16] Chao Peng, Xiangyu Zhang, Gang Yu, Guiming Luo, and Jian Sun, "Large kernel matters—improve semantic segmentation by global convolutional network," in *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*. IEEE, 2017, pp. 1743–1751.

[17] Guosheng Lin, Anton Milan, Chunhua Shen, and Ian D Reid, "Refinenet: Multi-path refinement networks for high-resolution semantic segmentation.," in *Cvpr*, 2017, vol. 1, pp. 5168–5177.

[18] Peng Yuan, Yangxin Zhong, and Yang Yuan, "Faster r-cnn with region proposal refinement," Tech. Rep., Computer Science Department, Stanford University, 2017.

[19] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer, "Automatic differentiation in pytorch," 2017.

[20] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell, "Caffe: Convolutional architecture for fast feature embedding," *arXiv preprint arXiv:1408.5093*, 2014.

[21] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman, "The pascal visual object classes (voc) challenge," *International journal of computer vision*, vol. 88, no. 2, pp. 303–338, 2010.

[22] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. Ieee, 2009, pp. 248–255.