УДК 004.896 DOI: 10.35803/1694-5298.2021.3.358-365

YUNSEOK JANG, JAESEOK KIM
[1]School of Electrical and Electronic Engineering, Yonsei University, South Korea

ЮНСЕК ЛАНГ, ЖАСЕОК КИМ
[1]Школа электротехники и электроники, Университет Йонсей, Южная Корея
yunejang@yonsei.ac.kr, jaekim@yonsei.ac.kr

FILTER PRUNING IN ORDER OF IMPORTANCE OF THE LAYER GROUPS TO ACCELERATE CONVOLUTIONAL NEURAL NETWORKS

ОТСЕЧЕНИЕ ФИЛЬТРА В ПОРЯДКЕ ОЦЕНКИ ВАЖНОСТИ ГРУПП СЛОЕВ ДЛЯ УСКОРЕНИЯ РАБОТЫ СЕТЕЙ CNN

CNN ТҮЙҮНДӨРҮН ЫЛДАМДАТУУ ҮЧҮН, КАТМАР ТОПТОРУНУН МААНИЛҮҮЛҮК УПАЙЫ БОЮНЧА КЫСКАРТЫП ЧЫПКАЛОО

Конволюциялык нейрон тармактарында камтылган түзмөктөрдү жайгаштыруу үчүн ар кандай ылдамдануу ыкмалары изилденген. Алардын ичинен фильтрди кесүү эң активдүү изилдөө болуп саналат, анткени аны аппараттык камсыздоодо ишке ашыруу оңой жана эсептөө жана эс тутум наркын төмөндөтүүдө жогорку тактык сакталат. Бул макалада биз жогорку тактыкты сактап, жогорку FLOP азайтууга жетишүү үчүн катмарларды топтоо, ар бир топтун маанилүүлүгүн табуу жана маанилүүлүккө жараша топ менен кыркуу ыкмасын сунуштайбыз. Алгач, алдын-ала даярдалган тармактын катмарларын чыгуучу өзгөчөлүктөр картасынын көлөмүнө жараша топторго бөлөбүз. Андан кийин, биринчи даражадагы Тейлордун кеңейишин колдонуп, бир топко маанилүүлүктү эсептейбиз. Акыр-аягы, чыпкаларды кыркуу эң жогорку мааниге ээ болгон топтун ирети менен жүргүзүлөт. CIFAR-10 боюнча окутулган VGG жана ResNet бутактарын кыйганда, биздин сунуш кылган ыкма заманбап ыкмаларга салыштырмалуу тактык жана FLOPs боюнча эң жогорку көрсөткүчтөрдү көрсөтөт. Белгилей кетүүчү нерсе, ResNet-50де 50% чыпкаларды алып салуу менен, FLOPs көлөмүнүн 70,85% азайышына жетишитик, баштапкы тактыгында 0,41% жоготуу.

Өзөк сөздөр *- фильтрди кыскартуу, маанилүүлүгүн баалоо, моделди кысуу, кыскартуу тартиби.*

Были изучены различные подходы к ускорению для развертывания сверточных нейронных сетей во встроенных устройствах. Среди них наиболее активным исследованием является отсечение фильтра, поскольку его легко реализовать на оборудовании и поддерживать высокую точность при одновременном снижении вычислительных затрат и затрат на память. В этой статье мы предлагаем метод группировки слоев, определения важности каждой группы и групповой обрезки в соответствии с порядком важности для достижения высокого сокращения FLOP при сохранении высокой точности. Сначала мы разделяем слои предварительно обученной сети на группы в соответствии с размером выходной карты функций. Затем мы рассчитываем оценку важности для каждой группы, используя разложение Тейлора первого порядка. Наконец, выполняется отсечение фильтра в порядке от группы с наивысшим баллом важности. При сокращении VGG и ResNet, обученных на CIFAR-10, предложенный нами метод показывает

превосходную производительность по точности и FLOP по сравнению с современными методами. Примечательно, что в ResNet-50 мы достигаем снижения FLOP на 70,85% за счет удаления 50% фильтров с небольшой потерей базовой точности на 0,41%.
Ключевые слова *- сокращение фильтра, оценка важности, сжатие модели, порядок сокращения.*

Various acceleration approaches have been studied to deploy convolutional neural networks in embedded devices. Among them, filter pruning is the most active research because it is easy to implement in hardware and keeps high accuracy while reducing the computational and memory cost. In this paper, we propose a method of grouping layers, finding the importance of each group, and group- wise pruning according to the order of importance to achieve high FLOPs reduction while retaining high accuracy. First, we divide the layers of the pre-trained network into groups according to the size of the output feature map. Next, we calculate the importance score per group using first-order Taylor expansion. Finally, filter pruning is performed in order from the group with the highest importance score. When pruning VGG and ResNet trained on CIFAR-10, our proposed method shows superior performance in accuracy and FLOPs compared to the state-of-art methods. Notably, on ResNet-50, we achieve 70.85% FLOPs reduction by removing 50% of the filters, with a slight loss of 0.41% in the baseline accuracy.
Index Terms *– Filter pruning, Importance score, model compression, Pruning order.*

Introduction. Convolutional neural network (CNNs) has shown excellence in performance in variety of fields, particularly in computer vision such as image classification [1],[2], object detection [3],[4], and semantic segmentation [5],[6]. As deep learning structures evolve and high-performance Graphic Processing Units (GPUs) are developed, CNN performance is getting better. However, as CNNs become deeper and more complex, they have increased computational cost and memory usage. As a result, it is difficult to deploy CNN to an embedded device or to inference in real-time. Therefore, several methods to accelerate CNN such as low-rank decomposition [7], quantization [8], and pruning [9]-[15] have been proposed. Among them, the pruning method has been actively studied recently because of the merits of implementing CNNs with limited memory. Pruning methods can be categorized into two groups, weight pruning and filter pruning. Weight pruning [9],[10] is a method of removing neurons inside the filter that do not significantly affect accuracy. The weight pruning method removes the neurons irregularly. Therefore, it is difficult to achieve high speed-up without specialized hardware because the memory access is irregular at online inference after pruning. Filter pruning [11]- [15] is a method of removing filters as a whole. This method provides regular memory access during the online inference phase, resulting in speedup without specialized hardware. Therefore, most of the recent pruning methods use the filter pruning method.

There are several ways to choose filters to prune. In the early studies, the norm-based method was proposed, which calculates the norm of a weight [11], activation [12], or gradient [13] and removes filters with low values. Recently, however, several studies have pruned differently, questioning the “small-norm-less-importance” criterion. In particular, a method of eliminating the low score filters has been proposed by calculating the importance score of the filters. In [14], Yu et. el. proposed a method of calculating the importance scores of neurons in a layer immediately beforeclassification and calculating the importance of all filters in the network by back-propagating these scores. In [15], Molchanov et. el. proposed a method of approximating the effect of the filter on the final loss using Taylor expansion to obtain the importance scores of filters in all layers, and then globally pruning the filters with small scores.

Existing importance-based pruning methods are layer-by-layer methods that prune filters for each layer or global methods that prune filters of all layers simultaneously with the same criterion. The layer- by-layer pruning methods are time consuming and have a drawback in that the pruning ratio of each layer must be determined in advance. The global pruning methods are removing filters globally across

all network layers. However, most of the pruned filters are in a layer with a small output feature map size, which does not reduce enough FLOPs.

In this paper, we propose a group-by-group pruning method which groups layers according to the output feature map size and then prunes them in order of group with the highest total importance. Unlike the previous importance-based pruning methods, our method sets the order of pruning with importance score so that the accuracy is retained as high as possible. For VGG and ResNet on CIFAR-10 dataset, our proposed pruning method leads to improvement in terms of accuracy, and FLOPs with a small accuracy drop.

We highlight out main contributions below. First, by group wise pruning, a certain percentage of filters can be pruned even in groups with large output feature map sizes. Therefore, more FLOPs can be reduced than existing global filter pruning. Second, by pruning in the order of the most important groups, we can keep the network capacity high when pruning more important groups. This method allows us to recover accuracy in fine-tuning during pruning. The proposed group-wise pruning method is verified in CIFAR-10 dataset and implemented in VGG and ResNet model. Compared to the recently proposed importance-based method, Taylor pruning, the FLOPs are much reduced and the accuracy is well retained.

The rest of the article is organized as follows: Section II describes our proposed filter pruning method. In Section III, various experimental results are presented. Finally, Section IV concludes the paper.

1. Proposed Method

Figure 1 shows a flowchart of our proposed group-wise pruning method.

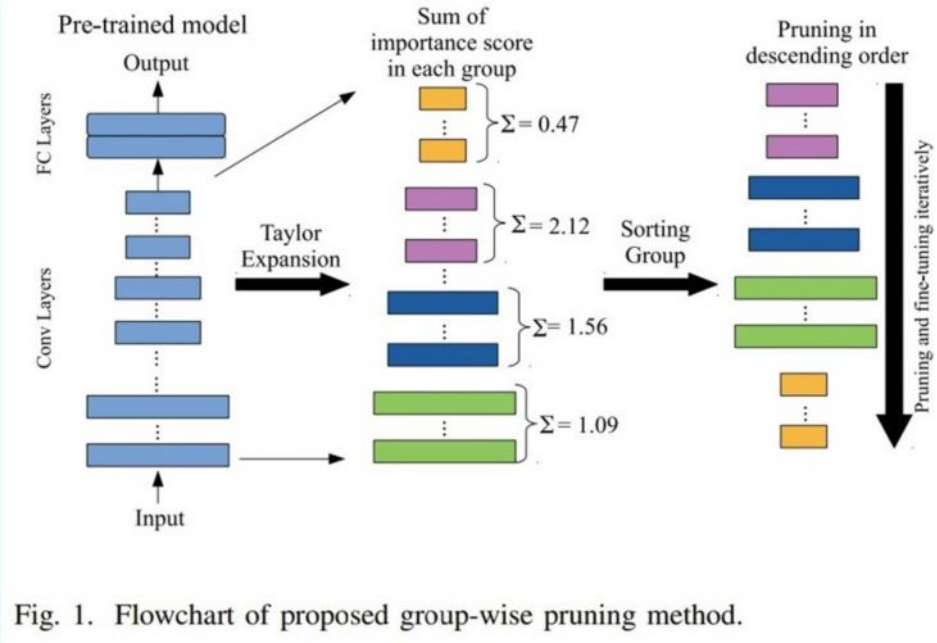

Fig. 1. Flowchart of proposed group-wise pruning method.

First, we approximate the importance score of all layers in the pre-trained model using first- order Taylor expansion. Next, we group the layers with the same output feature map size into one group, add all the importance scores of the filters in the group, and sort them in descending order. Finally, we repeat pruning and fine-tuning until the desired number of filters are removed, starting with the group with the largest import score.

Let $W = \{\omega_{1,s}, \ldots, \omega_{l,s}, \ldots, \omega_{n,s}\}$ be the parameter of a trained CNN model with n convolutional layers, where s_l is the output feature map size of the l-th convolutional layer. And, let $\omega_{l,s_l} = \{f_1^l, f_2^l, \ldots, f_{c_l}^l\}$ be the parameter of the l-th convolution layer with cl filters, and

$D=\{(x_0, y_0), (x_1, y_1), ..., (x_K, y_K)\}$ for the dataset, where (i, y_i) is input and output pair.

When the loss function is called L, the importance of the k-th filter in the l-th layer (I_{f^l}) can be regarded as the squared difference of the loss function values when the filter is pruned. Then, I_{f^l} can be expressed as follows:

$$I_{f_k^l} = (\mathcal{L}(\mathcal{D}, \mathbf{W}|f_k^l = 0) - \mathcal{L}(\mathcal{D}, \mathbf{W}))^2 \quad (1)$$

Computing Eq. (1) for all filters in network is computationally expensive, so we approximate Eq.(1) with first-order Taylor expansion at $f^l = 0$:

$$I_{f_k^l} \approx \left(\frac{\partial \mathcal{L}(\mathcal{D}, \mathbf{W})}{\partial f_k^l} f_k^l\right)^2. \quad (2)$$

Eq.(2) is easily computed because the gradient $\frac{\partial L(D, W)}{\partial f^l}_k$ is already obtained from backpropagation. Since filter f^l is a $s^2 c$ -dimensional vector, importance I_{f^l} can be thought of as $k \quad l \quad l-1 \quad k$ the squared value after accumulating the product of the gradient and the own value of filter s $s^2 c\ l-1$ times as in the following equation.

$$I_{f_k^l} \approx (\sum_{r=1}^{s_l^2 c_{l-1}} \frac{\partial \mathcal{L}(\mathcal{D}, \mathbf{W}))}{\partial f_{k,r}^l} f_{k,r}^l)^2, \quad (3)$$

where f^l denotes r-th component of filter f^l.

$k,r \qquad k$

Next, we group convolutional layers with the same output feature map size (O_S) into a group. If there are m types of O_S in a given CNN model ($S_1, ..., S_m$), the convolutional layers can be divided into m groups as follows:

$$\begin{aligned}
&G_1 = \{w_{1,s_1}, ..., w_{g_1,s_{g_1}}\},\ where\ s_1, ...,\ s_{g_1} = S_1 \\
&G_2 = \{w_{g_1+1,s_{g_1+1}}, ..., w_{g_2,s_{g_2}}\},\ where\ s_{g_1+1}, ...,\ s_{g_2} = S_2 \\
&... \\
&G_m = \{w_{g_{m-1}+1,s_{g_{m-1}+1}}, ..., w_{g_n,s_{g_n}}\}, \\
&where\ s_{g_{m-1}+1}, ..., s_{g_n} = S_m
\end{aligned} \quad (4)$$

We then express the importance of all groups as the sum of the importance of the filters in all the convolutional layers in each group. Therefore, we can express the importance score of the t-th group as:

$$I_{G_t} = \sum_{l=g_{t-1}+1}^{g_t} \sum_{k=1}^{c_l} I_{f_k^l} \quad (5)$$

Given the pruning ratio is γ and the number of filters per group is $f\,g\,1\,,...\,,f\,gm$, we remove$f\,g\,1\,\times\gamma\,\,,...\,f\,gm\times\gamma$ filters from each group. Next, we sort the importance scores of each groupobtained by (5) in descending order in order to prune the group with high importance score first, where $Ind_1, ..., Ind_m$ denote sorted group indices as follows:

$$Ind_1, ..., Ind_m = indexsort(I_{G_1}, ..., I_{G_m}), \quad (6)$$

A large sum of the importance scores of the filters in a group means that the group has a

large impact on accuracy. When pruning a group which has a large sum of importance scores, the capacity of the network should be maintained at the highest level. This is because the higher the network capacity, the better the recovery of the accuracy loss due to pruning when fine-tuning for several mini-batches after pruning. The experimental results in Section IV.A show that the highest accuracy is maintained when pruning in the order of high importance score when the pruning order is set in various ways.

Algorithm 1 Proposed Filter Pruning Method

Input: **W**: pretrained network, γ: pruning ratio, tf: total filter number in **W**, pf: pruning frequency, $f_{Ind_1}, f_{Ind_2}, \ldots f_{Ind_m}$: filter numbers in group $I_{G_{Ind_1}}, I_{G_{Ind_2}}, \ldots, I_{G_{Ind_m}}$;

Output: Pruned network

Find the importance score of each filter of **W** using (3).
Use (4) to group the layers.
Find the importance score per group with (5) and sort by (6).
$tn = 0$, $mb = 0$
repeat
    Update **W** using gradient obtained from backproga-tion.
    For each minibatch, the importance score of each filter is calculated using (3), and the importance score is accumulated for each filter.
    if $\text{Mod}((mb+1), pf) == 0$, **and** $tn < tf \times \gamma$ **then**
        Average the importance socre of each filter over the predefined minibatches.
        if $tn < t1 \times \gamma$ **then**
            Remove the N filters from $I_{G_{Ind_1}}$ with the smallest importance scores, $tn := tn + N$
        else if $tn < (t1 \times \gamma + t2 \times \gamma)$ **then**
            Remove the N filters from $I_{G_{Ind_2}}$ with the smallest importance scores, $tn := tn + N$
            ...
        else
            Remove the N filters from $I_{G_{Ind_m}}$ with the smallest importance scores, $tn := tn + N$
        end if
    end if
    $mb := mb + 1$
until predefined epoches

Algorithm 1 shows the pseudo code of the proposed pruning method. Each time a pre- defined mini-batch is performed, a pre-defined number of filters are pruned in order of the group with the highest importance score. After pre-defined number of filters are pruned in each group, we fine-tune the network for the rest of the epoch.

A. Theoretical Speedup Analysis.

Suppose there are c_l filters in the l-th layer and c_p filters are pruned, $c_{l+1} - c_{p_{l+1}}$ filters remain unchanged, and suppose the size of the input, output feature map and kernel of l-th layer is $s_{l-1} \times s_{l-1}$, $s_l \times s_l$, k. After pruning, the dimension of the output feature map is reduced from $c_l \times s_l \times s_l$ to $(c_l - c_p) \times s_l \times s_l$. Since the output of the l-th layer becomes the input of the l+1-th layer, the calculation of l+1-th layer decreases from $k^2 \times c_l \times c_{l+1} \times s_l \times s_l$ to $k^2 \times (l - c_{p_l}) \times (l+1 - c_{p_{l+1}}) \times s_l \times s_l$ when pruning c_p filters in l+1th layer. In other words, a proportion of $1 - (1 - \frac{c_{p_l}}{c_l}) \times (1 - \frac{c_{p_{l+1}}}{c_{l+1}})$ is reduced.

2. Experimental Results

A. Datasets and Experimental Setting

In order to evaluate the performance of the proposed method, the CIFAR-10 dataset is tested on VGG and Resnet models. CIFAR-10 is a 32×32 colored image with 10 classes, 60,000 training images and 10,000 test images.

For CIFAR-10 dataset, pruning and fine-tuning are performed for a total of 30 epochs, and 100 filters are set to prune for every 20 mini-batches. To calculate the average of the importance score, an exponential moving average is applied between iterations with coefficient 0.9. Initial learning rate is set to 0.01 and decreased to 0.001 after 15 epochs. Mini-batch size, momentum and weight decay are set to 128, 0.9 and 0.0001, respectively. In all experiments, the pruning ratio of the all layer groups is set to 0.5.

We run each experiment five times and averaged performance. All the experiments were conducted using pyTorch on single GTX1080Ti GPU.

TABLE I
COMPARISION OF THE PROPOSED PRUNING STRATEGY WITH OTHER STRATEGIES ON ACCURACY.

Network	Baseline (%)	Strategy *1) (%)*	Strategy *2) (%)*	Strategy *3) (%)*	Strategy *4) (%)*	Proposed (%)
VGG-16	93.80	92.50	92.34	**92.65**	92.57	92.62
ResNet-18	94.73	94.49	94.43	94.45	94.46	**94.68**
ResNet-50	95.60	95.01	95.08	94.99	95.01	**95.19**

TABLE II
COMPARISON OF FLOPS AND ACCURACY OF NETWORKS PRUNED BY PROPOSED PRUNING AND TAYLOR PRUNING METHODS.

Network	Method	Baseline Accu. (%)	Accelerated Accu. (%)	FLOPs	Pruned FLOPs (%)
VGG-16	Taylor Pruning (50%)	93.80	93.45	2.33E8	26.34
	Proposed (50%)		92.62	7.73E7	75.31
ResNet-18	Taylor Pruning (50%)	94.73	94.59	2.36E8	58.02
	Proposed (50%)		94.61	1.04E8	81.49
ResNet-34	Taylor Pruning (50%)	95.42	95.02	4.91E8	57.91
	Proposed (50%)		94.78	2.94E8	74.76
ResNet-50	Taylor Pruning (50%)	95.60	95.17	6.36E8	58.02
	Proposed (50%)		95.19	4.23E8	70.85

B. Effect of Pruning Order on Network Performance.

We compare four different pruning strategies to show that the accuracy is best retained when pruning is done from the group which has highest sum of importance score to the lowest. Below are four different pruning strategies.

1) From lowest sum of importance score to highest: Pruning is done in the reverse order of the proposed strategy.
2) From lowest output dimension to highest: We prune network from the furthest group to the closest group to the network input.
3) From highest output dimension to lowest: We prune network from the closest group to the furthest group to the network input.
4) Prune all groups simultaneously: We prune a predefined filters for each group to make the sum of the filters pruned for every 20 mini-batches to 100. The ratio of the filters pruned per group for every 20 mini-batches equals to the ratio of the total filters per group of baseline network.

We compared the proposed pruning strategy with four strategies using VGG-16 and ResNet-18, a relatively shallow networks, and ResNet-50, a relatively deep network, in terms of accuracy. As shown in Table II, the proposed pruning strategy showed the second highest accuracy in VGG-16 and the highest accuracy in ResNet-18 and ResNet-50. Except for the pruning strategy, all conditions are the same, so the proposed pruning strategy is the best for both shallow and deep networks.

TABLE III
COMPARISION ON THE THEORETICAL AND REALISTIC SPEEDUP. T AND P MEANS TAYLOR PRUNING AND PROPOSED METHOD, RESPECTIVELY.

Network	Mtd.	Baseline time (ms)	Pruned time (ms)	Realistic speedup(%)	Theoretical speedup(%)
VGG-16	T	5.31	3.76	29.19	26.34
	P		2.38	55.18	75.31
ResNet-18	T	7.70	4.41	42.73	58.02
	P		2.95	61.69	81.49
ResNet-34	T	14.62	8.26	43.50	57.91
	P		6.32	56.77	74.76
ResNet-50	T	30.54	22.28	27.04	58.02
	P		18.34	39.95	70.85

A. Comparison with Taylor Pruning method

The proposed pruning method is compared against Taylor pruning trained by CIFAR-10 dataset. Both methods compute the importance score of all filters by first order Taylor expansion. Pre-trained VGG-16, ResNet-18, ResNet-34, and ResNet-50 networks are pruned with mentioned two methods, and accuracy, FLOPs and speed up are compared.

Third and fourth column of Table III shows the results of comparing the accuracy and FLOPs of the two pruning methods. Like the Taylor pruning method, the proposed method prunes the filter by 50% while retaining the accuracy of the baseline network. Notably, when pruning the ResNet-18 and ResNet-50 networks with proposed method, the accelerated accuracy is 0.02% higher than Taylor pruning method. Compared with the accuracy of the baseline network, accuracy drop is only 0.12% and 0.41%, respectively.

The fifth column in Table III shows the FLOPs of the pruned network, and the sixth column shows percentage pruned FLOPs when compared to the baseline network. The proposed method show significantly lower FLOPs than the Taylor method for all four networks. In the case of VGG16, the proposed method pruned 75.31% of FLOPs, which is 48.97% higher than Taylor pruning method. The proposed method pruned FLOPs of ResNet-18, ResNet-34, and ResNet-50 networks by 81.49, 74.76, and 70.82%, respectively, which are 23.47, 16.85, and 12.80% higher than Taylor pruning method, respectively.

These results validate the effectiveness of proposed pruning method, which can prune more FLOPs while pruning same number of filters and retaining accuracy of the baseline network.

To see how much the realistic speedup ratio is, we measure the forward time of the network pruned by the proposed and Taylor pruning method. Batch size is set to 64 and forward time is measured for 10 epochs. After calculating the average time for each epoch, the pruned time is obtained by averaging 8 average times except the epoch with the highest and lowest average time. Table IV shows the results. In all networks, realistic speedup tends to be similar to theoretical speedup. However, realistic speedup is less than theoretical speedup because it does not include batch normalization and pooling layers, which need the inference time on GPU, in the calculation of FLOPs. In addition, the limitation of IO delay, buffer switch and efficiency of BLAS libraries also lead to the gap between realistic and theoretical speedup.

3. Conclusion

In this work, we propose a novel filter pruning method to accelerate the CNNs. We approximate each filter's contribution using first-order Taylor expansion to obtain importance score. Group layers with the same output feature map dimension. After adding together the importance scores of the filters in the group, group-wise pruning is performed in order of the highest total importance score. Experimental results show that our proposed pruning method is comparable in terms of accuracy and outperformed in terms of FLOPs and time reduction when compared with the Taylor pruning method, which is known to have the highest correlation with reliable estimate of the true importance.

References

[1] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale imagerecognition", in Int. Conf. Learn. Representations, 2015.
[2] K. He, X. Xhang, S. Ren, and J. Sun, "Deep residual learning for image recognition", arXiv:1512.03385, 2015.
[3] R. Girshick, J. Donahue, T. Darrall, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in Proc. IEEE Conf. Comput. Vis. Pattern Recog., 2014, pp. 580–587
[4] L. Zhang, Y. Yan, L. Cheng, and H. Wang, "Learning Object Scale With Click Supervision for Object Detection," in IEEE Signal Process. Lett., vol. 26, no. 11, pp. 1618–1622, Nov. 2019.
[5] J. Long, E. Shelhamer, and T. Darrall, "Fully convolutional networks for semantic segmentation," in Proc. IEEE Conf. Comput. Vis. Pattern Recog., 2015, pp. 3431–3440
[6] Y. Zhuge, G. Yang, P. Zhang, and H. Lu, "Boundary-guided feature aggregation network for salient object detection," IEEE Signal Process. Lett., vol. 25, no. 12, pp. 1800–1804, Dec. 2018.
[7] M. Jaderberg, A. Vedaldi, and A.Zisserman, "Speeding up convolutional neural networks with low rank expansions," in Proc. British Mach. Vis. Conf., 2014.
[8] D. D. Lin, S. S. Talathi, and V. S. Annapureddy, "Fixed point quantization of deep convolutional networks, arXiv preprint arXiv:1511.06393, Nov. 2015.
[9] S. Han, H. Mao, and W. J. Dally. "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," in Int. Conf. Learn. Representations, 2016.
[10] X. Dong, S. Chen, and S. J. Pan, "Learning to prune deep neural networks via layer-wise optimal brain surgeon, in Proc. Advances in Neural Information Processing Systems, pp. 48574867. 2017.
[11] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient convnets, in Proc. Int. Conf. on Learning Representations , 2017.
[12] J.-H. Luo, J. Wu, and W. Lin, "ThiNet: A filter level pruning method for deep neural network compression, in Proc. IEEE Int. Conf. on Computer Vision, pp. 50585066. 2017.
[13] C. Liu, H. Wu, "Channel pruning based on mean gradient for accelerating convolutional neuralnetworks", Signal Processing, vol 156, 2019, pp. 84–91.
[14] R. Yu, A. Li, C.-F. Chen, J.-H. Lai, V. I. Morariu, X. Han, M. Gao, C.-Y. Lin, and L. S. Davis, "NISP: Pruning networks using neuron importance score propagation", in Proc. IEEE Conf. Comput. Vis. Pattern Recog., 2018, pp. 9194–9203.
[15] P. Molchanov, A. Mallya, S. Tyree, I. Frosio, and J. Kautz. "Importance estimation

ABBYY PDF Transformer 2.0 Click here to buy www.ABBYY.com

for neural network pruning", in Proc. IEEE Conf. Comput. Vis. Pattern Recog., 2019, pp. 11264–11272.