PCAS: Pruning Channels with Attention Statistics for Deep Network Compression

Kohei Yamamoto
yamamoto833@oki.com
Oki Electric Industry Co, Ltd., Japan
Kurato Maeno
maeno284@oki.com

Abstract

Compression techniques for deep neural networks are important for implementing them on small embedded devices. In particular, channel-pruning is a useful technique for realizing compact networks. However, many conventional methods require manual setting of compression ratios in each layer. It is difficult to analyze the relationships between all layers, especially for deeper models. To address these issues, we propose a simple channel-pruning technique based on attention statistics that enables to evaluate the importance of channels. We improved the method by means of a criterion for automatic channel selection, using a single compression ratio for the entire model in place of per-layer model analysis. The proposed approach achieved superior performance over conventional methods with respect to accuracy and the computational costs for various models and datasets. We provide analysis results for behavior of the proposed criterion on different datasets to demonstrate its favorable properties for channel pruning.

1 Introduction

Convolutional neural networks (CNNs) have brought about great advances in tasks such as object recognition, object detection, and semantic segmentation in several years. However, the number of parameters required for CNNs that have generally good performance tends to be very large, which imposes memory requirements and computational cost that exceed the capabilities of mobile and compact devices. To solve the problems, various techniques [6, 10, 14, 19, 29] have been proposed for making CNNs more efficient and increasing the speed of inference. In these works, network pruning is an important approach for removing redundant parameters from the models.

Research into the pruning methods are roughly divided at two levels: the neuron level and the channel level. At the neuron level, the number of parameters is reduced by severing connections between spatial neurons in the convolutional layer or the neurons in the fully connected layer. At the channel level (channel pruning), the connections of all structural elements that respond to a particular channel are dropped for input and output channels in the convolutional layer; pruning is performed in sets of groups. This method differs from the neuron-level pruning (*e.g.*, in [8]) in that it does not require any special implementation since the shape of the weight matrix is reduced. However, since deletions are performed in sets of groups, the influence on the precision is significant and the problem setup is more difficult than with neuron-level methods. The channel pruning methods have several difficulties that

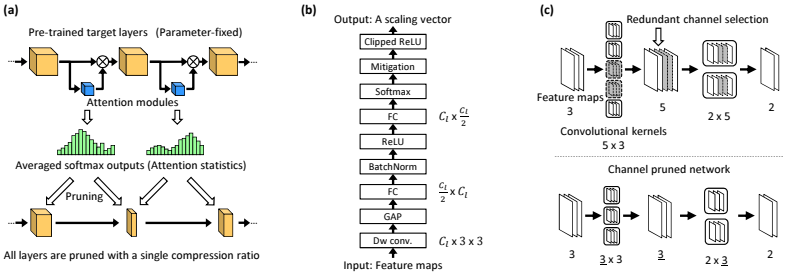


Figure 1: **(a)** Overview of our pruning approach. **(b)** The building blocks of a single attention module (see details in Section 3.1). **(c)** An example of channel pruning for convolutional layers.

require designing the criteria for evaluating the importance of channels and set the compression ratio for each layer. Especially, the latter is serious problem because the many existing methods [11, 12, 21, 23, 33] need the ratios as hyper-parameters for the pruning. In general, the problem will be more difficult when using deeper models.

In this paper, we propose a channel pruning method for pre-trained models. Figure 1a shows an overview of our approach. In this method, the importance of channels is evaluated using neural networks (we call attention modules) connected immediately before all target layers in the pre-trained model. Although these attention modules need to be trained, the modules are able to infer the importance of the channels. Furthermore, it is optimized in all levels of layers since the attention module for a lower level is trained by considering the gradient of the pre-trained model and the gradient of the upper-level attention module.

The major contributions of this paper are summarized as follows:

- We propose the attention statistics, a novel attention based criterion for channel pruning, to estimate redundant channels via optimizing the appended neural networks.
- We present a simple pruning technique that it requires only one compression ratio, which does not require the layer-by-layer compression ratio tuning that needs carefully controlling the trade-off between accuracy and the pruning performance.
- We evaluate our approach on various networks, VGG-10/16, ResNet-18/50/56, MobileNet and SegNet for image recognition/segmentation tasks. And the results show promising pruning performances on CIFAR-10/100, ImageNet and CamVid datasets.

2 Related Work

Non-pruning approaches. Network quantization [19, 27, 31] is a technique for replacing typical 16/32-bit weights/activations with fewer-bit ones. Hubara *et al*. [19] proposed a training scheme of binarized networks and Rastegari *et al*. [27] improved it by introducing scaling factors to minimize quantization error. Wan *et al*. [31] incorporated binary weight and ternary input to achieve better performance. Knowledge distillation [2, 14] is applied to train a small student model using a larger teacher one. Belagiannis *et al*. [2] presented a two-player adversarial learning scheme to train the student model. Factorization [4, 15] is an approach whereby a standard convolution is factored into more efficient operations. Chollet [4] developed a depth-wise separable convolution and Howard *et al*. [15] used it to design more efficient models with a width and resolution multiplier. Although such non-pruning approaches are based on different perspectives, they can be combined with the channel pruning approach to obtain even more compact models.

Channel pruning. Li *et al*. [23] performed selection of redundant channels using ℓ_1-norms of per-channel weights. To decide the compression ratio for each layer, they analyzed

the precision degradation depending on the number of channels that were deleted. However, since compression ratios were decided by the user, they were not necessarily optimal. As for using the norm of per-channel weights as a criterion, He *et al*. [10] proposed the soft-pruning approach that the less ℓ_2-norm kernels were zeronized for each epoch during training (allowing for updating from zeros in the next epoch). Luo *et al*. [21] found redundant channels by using the reconstruction error of each layer as the criterion, which compared the output before and after excluding certain channels and identifies channels with smaller error as more important. Since channels were selected in each layer, the relationships between layers could not be considered and the compression ratios needed to be set manually. Furthermore, a lot of time was needed for fine-tuning each layer. Yu *et al*. [33] were focused on the reconstruction error of the last layer before classification and estimated the less important neurons in the backward propagation of the scores that were derived from the error. Similarly, He *et al*. [12] solved the optimization problem of minimizing the reconstruction error in each layer by assigning a variable to each channel as a method that used ℓ_1 regularization. Huang *et al*. [18] introduced additional scaling factors to not only the output of channels but also the residual branches, and trained them to close 0 with the sparsity regularization for pruning, as in [12]. Both Huang *et al*. [17] and He *et al*. [13] used the reinforcement learning for channel pruning. They pruned unimportant channels selected by the agent networks that were trained to maximize the specialized reward functions for improving pruning performance. Furthermore, their methods had the property of the automatic channel selection in the same manner as for our proposed method.

Attention. The attention mechanism [26] that explicitly propagates positions to reference in spaces or in series are used for several applications. Recent image recognition research have worked to increase accuracy by applying the mechanism. Wang *et al*. [32] applied attention in spatial and channel directions for ResNet [10]. Hu *et al*. [16] introduced attention in only channel directions to increase the performance of recognizing features by emphasizing channels according to the input. The application of the mechanism to the model optimization or pruning has not yet been common.

3 Approach

In this section, we first provide a brief background on channel pruning. Then we discuss the whole scheme of our approach and its details (including the pruning criterion, the training trick and the technique for selecting redundant channels).

Background. In CNNs, the convolutional kernel (or filter) for the $l \in \{1,\ldots,L\}$-th layer is represented by a fourth-order tensor of the dimension $C_{l+1} \times C_l \times H_l \times W_l$, where C_l is the number of channels, and H_l and W_l are the width and height of the kernel, respectively. Note that C_{l+1} belongs to the output side when C_l belongs to the input side. In general channel pruning schemes as shown in Fig. 1c, redundant channels are first removed by some kind of criterion. By removing several channels from the feature maps in this way, the dimensions C_{l+1} and C_l of the corresponding convolutional kernels can also be removed. After removing part of the channels, damage from pruning can be restored through fine-tuning of the model using the training data. Although our approach also follows this scheme, we introduce a new strategy for the selection of redundant channels.

3.1 Pruning Strategy with Attention

Our goal is to estimate redundant channels precisely in pre-trained CNNs using the other neural networks (attention modules). First, as visualized in the Fig. 1a, we connect the attention modules to immediately before the all of the pre-trained layer. Next, we train

the modules without updating any of parameters of the pre-trained layers under the same conditions (the same training data and the same loss function are used). After training, we calculate our proposed pruning criterion (details in Sec. 3.2) from the modules, and convert a given single compression ratio into the per-layer compression ratios using that criterion (details in Sec. 3.4). Then we determine the redundant channels in the pre-trained layer based on both the criterion and the compression ratios, and prune them. Finally, we remove all the modules and then fine-tune the pruned network to restore pruning damage.

The role of the attention modules is to emphasize channels that contribute to reducing the loss function and to deemphasize others. Because deemphasizing a channel produces the same effect as deemphasizing a group of convolutional weights that correspond to the channel, we assume that the deemphasized groups of weights have less influence on the original network and can be pruned with less accuracy degradation. More specifically, the attention module generates a C_l-dimensional vector from the feature maps, and emphasis is achieved by a channel-wise multiplication of the vector by the feature maps again. This operation is also known as the self-gating or the self-attention mechanism [16, 32]. We expect that this mechanism can also be useful for evaluating the importance of channels.

Fixing the pre-trained weights causes the attention module to search for a solution that reduces the loss function under conditions of only being able to scale each of the input channels to nearly all layers of the pre-trained model. Although this means that multiple attention modules are connected when there are three or more convolutional layers in the pre-trained model, the training of these is performed simultaneously. Thus, the lower-layer attention module performs optimization based on the gradients both from the pre-trained model and from the upper-layer attention module. In other words, the attention layers are optimized overall since the relationships with other layers are considered.

We now describe the architecture of the attention modules as shown in Fig. 1b. First, depth-wise convolution is executed independently on the channels [4] for extracting spatially common feature maps from the parameter-fixed layer outputs. Next, global average pooling (GAP) [24], Fully connected (FC) layer, batch normalization [20], and a ReLU function are applied to emphasize the difference between channels [16]. Finally, the softmax function, mitigation function multiplication (details in Section 3.3), and clipped ReLU [9] are applied.

3.2 Pruning Criterion

Naturally, since the behavior of attention varies according to the input data, it cannot be used as-is as a channel pruning criterion. We therefore propose the attention statistic that is a quantity found by element-wise averaging of the softmax outputs of the attention modules over all training data, as a criterion for selecting important channels. It is defined as follows:

$$a_{l,c} = \frac{1}{|\mathcal{D}|} \sum_{i \in \mathcal{D}} s_{l,c,i}, \tag{1}$$

where $s_{l,c,i}$ is the output of the softmax function, $c \in \{1, \ldots, C_l\}$ is the index of the channel, and $\mathcal{D}$ is the set of training data. Furthermore, we take $a_{l,c} \in \mathcal{A}_l$, as described below. $\mathcal{A}_l$ is a set of channels in the attention statistics for the l-th layer, and $|\mathcal{A}_l| = C_l$. In fact, the difference between channels needs to be emphasized. For example, in [16, 32], although the sigmoid independently to each channel is used to construct the scaling vector, it is difficult to obtain a clear difference in the statistical quantity when comparing between channels. In the case of adjusting the inputs to the pre-trained model, this is because when the pre-trained model is assumed to be fully optimized, using nearly 1 for all values gives the best accuracy. We therefore introduce the softmax function as a constraint that emphasizes the difference

between channels. Due to the property of the softmax output, emphasizing certain channels requires deemphasizing others. This is a desirable property for emphasizing difference between channels.

3.3 Training the Attention Modules

Training the attention modules might perform poorly if the output of the softmax function is applied to the input feature maps as-is. This is because the constraints on the pre-trained model with fixed parameters are too tight. If we assume that all of the channels of the input feature maps that belong to the l-th layer have the same importance and the attention module is able to perfectly infer this, then the element values of output will all be $1/C_l$. In other words, the feature maps become smaller depending on the magnitude of C_l after multiplication. Now, since the gradient is kept low each time a module spans across multiple layers, the gradient of the attention modules near the input layer disappears.

We therefore relax this constraint by multiplying the mitigation function to the output elements of the softmax function. The mitigation function can be defined as

$$f_l(\alpha) = \frac{C_l}{1+\alpha(C_l-1)}, \tag{2}$$

where $\alpha \in [0,1]$ is a hyper-parameter for strength of the constraint on the softmax. If we assume that all the output elements of the attention module are $1/C_l$ when $\alpha = 0$, then all of the channels of the feature maps are completely unaffected by the attention module (i.e., an identity projection), and the gradient can also propagate without decreasing. However, in the case of $\alpha \to 1$, it approaches the output from the softmax alone. The important effect is that the value of α gradually increases each time the attention module weights are updated to strengthen the constraints. This effect allows the loss divergence to be avoided and solutions emphasizing the channel difference to be obtained. Although performance deteriorates from the original pre-trained model as α is increased, since the aim of this process is for training the attention modules, this degradation is not a problem as long as the optimization is successful under the given constraints. For stable learning, we provide the same range ($[0,1]$) as the softmax by applying the clipped ReLU [9]. It can prevent the loss value from increasing fairly in the case that a scaling vector heavily enlarges specific channels.

This approach can also be applied to the FC layers as a structured pruning method. A pruning evaluation for the FC layers is also included in Section 4.1.

3.4 A Single Compression Ratio for All Layers

Our method only needs a single compression ratio $r \in [0,1]$ that can be converted into the ratio for each layer l. Note that the ratio r is proportional to the total number of channels contained in all layers. First, we define the global threshold $t \in [0,T]$ where $T \in \mathbb{R}_{>0}$ is an arbitrary number, and the local threshold t/C_l. There is a clear relationship that the global threshold can also be converted into the local one. Here, the total number of channels whose attention statistics are under the local threshold can be represented as follows:

$$U_l(t) = |\{a\colon a < t/C_l \cap a \in \mathcal{A}_l\}|. \tag{3}$$

Next, we define the local compression ratio $U_l(t)/C_l$ for each layer l and the global compression ratio $g(t) = (\sum_{l=1}^{L} U_l(t))/(\sum_{l=1}^{L} C_l)$. The optimal global threshold t^* is found by solving the problem formulated as follows:

$$t^* = \arg\min_t |\, g(t) - r \,|. \tag{4}$$

Since it has only one parameter t and the convexity property, search methods (*e.g.*, the grid search) can solve this problem easily. Finally, the l-th layer's channels whose criteria are under t^*/C_l are selected as the pruning targets. We expect the softmax distribution to flatten as the number of important channels increases. This method aims to prevent important channels with flat distributions from being pruned and redundant channels with non-flat distributions from remaining. In general, when the criterion is used for evaluating the importance of channels, it is difficult to directly compare channels that belong to different layers. Although attention statistics are also not strictly comparable with those channels, we propose a roughly fair comparison technique using the properties of per-layer normalization by the softmax function. In many cases, important channels are kept even if the r value is higher. If all channels could be pruned in a layer, just keeping the most important channel is a simple workaround.

4 Experiments

In this section, we first evaluated effectiveness in comparison with conventional state-of-the-art pruning methods against the various models. Next, we report the ablation study results. Finally, we analyze the behavior of the attention statistics.

We evaluated the proposed method against the CIFAR-10/100 [22] and the ImageNet (ILSVRC-2012) [6] datasets for the object recognition task, and the CamVid road scenes [3] dataset for the semantic segmentation task. In all experiments, we used the SGD optimization algorithm with a momentum of 0.9 to train the attention modules and to fine-tune the pruned networks, and did not prune the first convolutional layer, where the influence was wide and significant. As when pruning the ResNet-type architecture in all datasets, a sampling technique [12] for discarding an arbitrary number of input channels at the start of the residual branch was introduced to expand the range of target channels. We implemented our proposed method on Chainer [30]. Note that all evaluated models in these experiments had 32-bit floating point weights. The experimental settings for each dataset are described below.

CIFAR-10/100. We evaluated VGG-10 and ResNet-18/56 on CIFAR-10, and evaluated VGG-10 and ResNet-50 on CIFAR-100. The attention modules were trained for 50 epochs with a learning rate of 10^{-2} and the value α was linearly increased from 0 to 6×10^{-2}, and then the rate was changed to 10^{-3} for another 50 epochs of training to stabilize the solution with the target value of α using a batch size of 128. We adopted some data augmentation techniques: horizontal flip, image expansion [25], and random crop (where images are cropped to 28 $\times$ 28 pixels).

ImageNet (ILSVRC-2012). We evaluated our pruning method on VGG-16 [28], ResNet-18/50 [10] and MobileNet [15]. During training of attention modules for all models, the value α was linearly increased from 0 to 2×10^{-3} for 5 epochs, then training continued for 5 more epochs for stabilization. The learning rate was fixed at 10^{-3} while training the attention modules. In the fine-tuning step, VGG-16 and MobileNet were trained for 45 epochs, dropping the learning rate from the initial 10^{-3} value by 10^{-1} every 15 epochs. In contrast, ResNet-18/50 were trained for 35 epochs and the learning rate was changed from 5×10^{-3} to 5×10^{-5} in the same manner as for VGG-16. We used batch sizes of 512 for VGG-16, ResNet-18 and MobileNet, and of 1024 for ResNet-50. We adopted two standard data augmentation techniques: 224×224 random cropping and horizontal flip.

CamVid. We evaluated with the SegNet [1] architecture. We trained the attention modules while increasing the value α from 0 to 2×10^{-3} and with a learning rate of 10^{-2} for 50 epochs, then trained with a fixed α and a learning rate of 10^{-3} for 50 more epochs using a

Table 1: Comparison among several different pruning methods for the object recognition.

Dataset	Model	Method	Top-1 Acc. %		Top-5 Acc. %		#Params.		#FLOPs	
CIFAR-10	ResNet-56	Pruned-B [23]	93.06	↑0.02	-		0.73M	↓13.7%	91M	↓27.6%
		NISP-56 [33]	-	↓0.03	-		0.49M	↓42.6%	71M	↓43.6%
		AMC [11]	91.90	↓0.90	-		-		63M	↓50.0%
		SFP (40%) [10]	93.35	↓0.24	-		-		59M	↓52.6%
		PCAS-35	93.58	↑**0.54**	-		0.39M	↓**53.7%**	56M	↓**54.8%**
CIFAR-100	ResNet-50	[23] (our impl.)	73.60	↓0.86	-		7.83M	↓54.2%	616M	↓56.3%
		PCAS-60	73.84	↓**0.62**	-		4.02M	↓**76.5%**	475M	↓**66.3%**
ImageNet	VGG-16	ThiNet-Conv [23]	69.80	↑**1.46**	89.53	↑**1.09**	131.44M	↓5.0%	9.58B	↓69.0%
		PCAS-45	69.41	↑1.00	89.22	↑0.85	128.95M	↓**6.8%**	8.59B	↓**72.2%**
		SSS [13]	68.53	↓3.93	88.20	↓2.64	130.50M	↓5.6%	7.67B	↓75.2%
		PCAS-50	68.83	↑**0.42**	88.82	↑**0.45**	128.05M	↓**7.4%**	7.49B	↓**75.8%**
		CP (5×) [10]	67.80	-	88.10	↓1.80	130.88M	↓5.4%	7.03B	↓77.2%
		PCAS-55	68.18	↓0.23	88.39	↑**0.02**	127.23M	↓**8.0%**	6.45B	↓**79.2%**
	ResNet-50	CP (2×) [10]	72.30	↓3.00	90.80	↓1.40	17.46M	↓31.5%	5.20B	↓32.8%
		ThiNet-70 [23]	72.04	↓0.84	90.67	↓0.47	16.94M	↓33.7%	4.88B	↓36.8%
		SSS-ResNet-26 [13]	71.82	↓4.30	90.79	↓2.07	15.60M	↓38.8%	-	↓43.0%
		NISP-50-B [33]	-	↓0.89	-		14.36M	↓43.8%	4.32B	↓44.0%
		PCAS-50	72.68	↓**0.04**	91.09	↑**0.03**	12.47M	↓**51.2%**	3.34B	↓**56.7%**

batch size of 8. We adopted a data augmentation technique: horizontal flip.

4.1 Comparison with Conventional Methods

For comparisons, we measured the number of floating point operations (FLOPs[1]), and the total number of parameters. We generated the pruned models with some compression ratios, and then selected the model that had the nearest accuracy to the compared method and noted the best pruning performance in the results.

Recognition model pruning. Table 1 compares the proposed method with conventional methods in the object recognition task that is commonly used for pruning performance evaluations. In Table 1, M/B means $10^6/10^9$, and arrows indicate absolute accuracy reductions or reduced ratios for the number of parameters and FLOPs. Our method is denoted as "PCAS", and the compression ratio usage is appended (*e.g.*, "-10" indicates that 10% of the compression ratio r is used). For ResNet-56 on CIFAR-10, PCAS outperformed the conventional methods that were based on the norm of weights [11, 23], the reconstruction error [33] and the reinforcement learning [13]. In the CIFAR-100 experiment, we used a ResNet-50 model whose reduction layers were replaced with zero-padding [7] to reduce the number of parameters. Although ResNet-50 has more parameters than ResNet-56, PCAS reduced redundant channels to a greater extent compared with the conventional methods. Regarding VGG-16, PCAS tended to reduce the number of the parameters more than other methods did, while also reducing the number of FLOPs at same levels of accuracy. As for ResNet-50, PCAS outperformed the other methods that included the depth pruning approach [18] and achieved the network with the fewest parameters and FLOPs.

Segmentation model pruning. Table 2 (*left*) shows pruning performance for the semantic segmentation task using SegNet on CamVid. PCAS showed competitive results. We confirmed that PCAS was able to exceed a 10% reduction in numbers of parameters compared with conventional methods that had the same level of accuracy and variations in FLOPs. Although PCAS reduced more than 60% of parameters with a relatively small compression ratio 30%, this indicated that most of the redundant channels had almost the same importance. This result suggests the effectiveness of PCAS for architectures with many channels in the middle of layers and for this task.

[1]In this study, FLOPs indicate only the number of operations in the convolutional or FC layer. Furthermore, FLOPs were calculated without considering the fused multiply-add (FMA) instruction for comparison, except for ResNet-56 on CIFAR-10.

Table 2: Pruning SegNet on CamVid (*left*). Pruning the FC layers of VGG-16 on ImageNet (*right*).

Method	Global Acc. %	#Params.	#FLOPs
[24] ([4]'s impl.)	83.50 ↓3.00	↓56.9%	↓**63.9%**
LTP [4]	88.60 ↑**2.10**	↓56.9%	↓**63.9%**
PCAS-30	88.57 ↑0.82	↓**67.8%**	↓63.8%

Method	Top-1 Acc. %	#Params.	#FLOPs
ThiNet-GAP [21]	67.34 ↓1.0	↓**94.0%**	↓69.8%
PCAS-L-55	67.91 ↓**0.5**	↓83.0%	↓**72.9%**

FC layer pruning. PCAS could be applied to structured pruning for FC layers by removing the depth-wise convolutional layer and the GAP operation from the attention modules. For ImageNet, we also evaluated this approach using VGG-16, which has numerous parameters in FC layers. Due to conceptual differences between the channels, we took a two-step pruning approach. Namely, we pruned an already pruned network using the same training methods. We chose the "PCAS-45" model in Table 1 as the target pruned network. As Table 2 (*right*) shows, PCAS (with appended suffix "-L") was competitive with the GAP approach using conventional methods [21]. Although the GAP approach reduced the larger number of parameters by removing FC layers, it is not capable to control the trade-off between accuracy and computational costs, unlike the pruning approach.

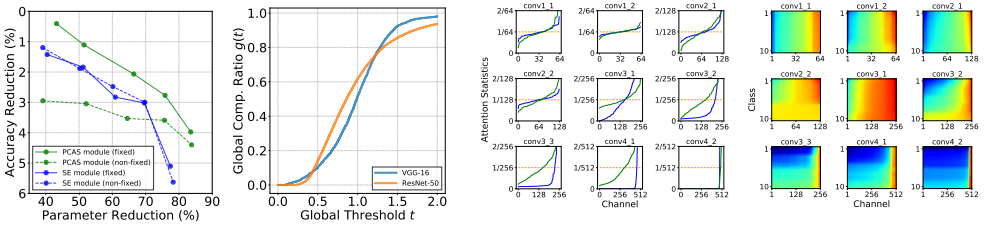
Comparison with non-pruning approaches. Table 3 shows the recognition results in comparison with the non-pruning compression approaches, including quantization, distillation and factorization techniques. Among the ResNet-18 models, PCAS with a compression ratio of 25% (PCAS-25) was superior to the quantization [27, 31] and distillation [2] approaches in terms of both the number of parameters and the accuracies. In contrast, quantization methods have potential for faster inference speed and lower memory consumption due to the utilization of more efficient bitwise operations through a dedicated implementation. MobileNet [15] with settings of 0.75 channel-width multipliers was more compact than the PCAS-25 model. Although MobileNet consists of depth-wise and 1 x 1 convolutional layers, PCAS can be also applied to such compact architectures by selecting only 1 x 1 convolutional layers as the channel pruning targets because of the channel independency of depth-wise convolutional layers. For MobileNet, we experimented with $r = 5\%, 10\%, 15\%$ (denoted as "+ PCAS") and it reduced about 15% of parameters and 30% of FLOPs with an accuracy degradation of 1.08%. We found that accuracy for MobileNet tends to be more sensitive to the compression ratio than do larger models (*e.g.*, VGG-16).

Table 3: Performance comparison with non-pruning approaches on ImageNet.

Method	Model	Top-1/5 Acc. %	#Params.	#FLOPs
Original	ResNet-18	68.98 / 88.69	11.68M	3.63B
XNOR-Net [24]	ResNet-18	51.20 / 73.20	11.68M	-
TBN [31]	ResNet-18	55.60 / 79.00	11.68M	-
ANC [1]	ResNet-18	67.11 / 88.28	13.95M	-
PCAS-25	ResNet-18	68.04 / 88.01	8.58M	2.59B
MobileNet [15] (our impl.)	0.75 MobileNet	68.19 / 88.39	2.59M	650M
MobileNet + PCAS-5	0.75 MobileNet	68.11 / 88.35	2.45M	571M
MobileNet + PCAS-10	0.75 MobileNet	67.51 / 87.88	2.32M	510M
MobileNet + PCAS-15	0.75 MobileNet	67.11 / 87.58	2.19M	458M

4.2 Ablation Study

Parameter-fixed training. We investigated the effectiveness of parameter-fixed training for attention modules, and differences in module architectures with conventional self-attention modules proposed in SE-Net [16]. The proposed method considers the importance of fixing parameters for the whole network without attention modules while training those for emphasizing the important differences between channels. To confirm this relationship, we conducted the experiment using ResNet-18, whose reduction layers were replaced with zero-padding and CIFAR-10. Figure 2a shows the performance results after fine-tuning with regard to accuracy and parameter losses in four cases. We denote PCAS as the use case of the proposed attention modules. Clearly, the pattern in the non-fixed case was inferior to that



(a) ResNet-18 on CIFAR-10 (b) VGG-16/ResNet-50 on ImageNet (c) VGG-10 on CIFAR-10/100 (d) VGG-10 on CIFAR-10

Figure 2: **(a)** Performance comparison with the training schemes and the modules on ResNet-18. **(b)** The relationship between t and $g(t)$ in Eq. 4 for VGG-16 and ResNet-50 on ImageNet. **(c)** Attention statistics on CIFAR-10 (blue) and CIFAR-100 (green). Channels in each layer are sorted by their values. **(d)** Attention statistics for each class. Values are converted to the log-scale for visibility and then colorized on each layer independently (red value is higher than blue). Channels and classes are sorted by the mean values over class axis and the minimum values over channel axis.

in the fixed case. Furthermore, we confirmed that actual attention statistics such as those shown in Fig. 2a in the non-fixed case had small discrepancies that were not sufficiently emphasized in the fixed case. From the above, we believe that this is because the weights in the original network are strongly dependent on the softmax constraint.

Attention module architecture. We evaluated differences in the attention module architectures. In contrast to the proposed modules using softmax, the SE modules in SE-Net independently apply the sigmoid function channel as the last blocks. Unlike our method, we trained the appended SE modules in a straightforward way as long as the SE-module outputs had nothing to do with between-channel constraints as in softmax (i.e., the training rarely broke down). The original network was then pruned using attention statistics, which were obtained from the sigmoid outputs. From the results in Fig. 2a, we confirmed that the SE modules were inferior to the parameter-fixed case of the proposed method. We also found no large differences between fixed and non-fixed trainings using the SE module. As the result, we consider that the procedure for mitigating the softmax constraints leads to effective extraction of channel importance.

Table 4: Pruning performance with different compression ratios on ImageNet.

Model	r	Top-1/5 Acc. %	#Params.	#FLOPs	Time (ms)
VGG-16	0%	68.41 / 88.37	138.34M	30.94B	71.41
	40%	69.91 / 89.65	129.89M	9.95B	27.62
	45%	69.41 / 89.22	128.95M	8.59B	22.13
	50%	68.83 / 88.82	128.05M	7.49B	19.03
	55%	68.18 / 88.39	127.23M	6.45B	16.94
	60%	67.35 / 87.88	126.51M	5.36B	14.99

Model	r	Top-1/5 Acc. %	#Params.	#FLOPs	Time (ms)
ResNet-50	0%	72.72 / 91.06	25.56M	7.72B	55.46
	45%	72.94 / 91.47	14.22M	3.85B	44.22
	50%	72.68 / 91.09	12.47M	3.34B	42.91
	55%	71.96 / 90.95	10.76M	2.82B	41.53
	60%	71.23 / 90.44	9.08M	2.37B	39.91
	65%	70.25 / 89.96	7.58M	1.94B	38.26

Pruning with different compression ratios. Figure 2b shows the relationship between t and $g(t)$ in Eq. 4 for both VGG-16 and ResNet-50 on ImageNet. As mentioned in Section 3.3, if all channels have the same importance for each layer, the relationship becomes the 0-1 step function switching at $t = 1.0$. However, since the curves slope upward, we confirmed that emphasizing the difference between channels was successfully achieved. We therefore experimented with different compression ratios r by 5%, and the results are summarized in Table 4. Note that the pruned models were generated from the same attention statistics, namely that the same curve in Fig. 2b was used for each model, and then they were fine-tuned under the same conditions. "Time" indicates the computational time of only a forward propagation, measured on a single GPU (NVIDIA Quadro GV100) with a batch size of 32. Although the performance goes worse with increasing the r value gradually, the pruned models gave better results than the original models till the r value of 0.50 and 0.45 for VGG-

16 and ResNet-50, respectively. And it also shows our method reduced 60% of channels with accuracy degradation of up to 1.5% for both models.

4.3 Analysis of Attention Statistics

For analysis, we used VGG-10, which is composed of only ten convolutional layers of VGG-16 by replacing FC layers with GAP. Figure 2c shows nine attention statistics corresponding to the channels for dimensions $C_1 = 64$ to $C_9 = 512$ in each convolutional layer in VGG-10 on CIFAR-10/100. These attention statistics directly show the importance of each layer. For example, since *conv1_2* has a shape that is closer to flat than other distributions, the importance of all of the channels is relatively high, and the effect on the accuracy is large. Furthermore, the distribution in *conv4_2* is heavily biased toward some channels, which indicates that there is a large number of redundant channels. The structure of VGG-10 is set to have more channels nearer the output-layer side, and these are clearly redundant.

By comparing the results of CIFAR-10 and CIFAR-100, we confirmed that the redundancy contained in the trained VGG-10 differs depending on the complexity of the problem. For example, from *conv4_1*, it is clear that the CIFAR-10 distribution is more biased than CIFAR-100, and the importance of some channels is high. In contrast, the CIFAR-100 distribution is only weakly biased, with other channels also contributing to the accuracy. Since our method decides the threshold value for pruning based on the ratio to the number of channels, it does not emphasize the channels in *conv4_1* for pruning in CIFAR-100 as much as in CIFAR-10. As a result, our method prunes independently of the complexity of the problem.

Figure 2d shows the attention statistics for each class. This visualization shows that the response to particular classes is weak. Furthermore, many of these are observed in output-side layers. Since the average is used in our method, channels that have weak response over all classes are pruned preferentially over channels that have specifically weak response to particular classes.

5 Conclusion

In this paper, we proposed a novel pruning method based on attention. Our goal was to more effectively extract the importance of channels from the original networks for pruning. The method trains attention modules inserted immediately before the target pre-trained convolutional or FC layers to learn the importance of the channels. After training, we can obtain pruning criteria from module inferences by taking statistics using the training data. Attention modules are one-shot trained, not in a layer-by-layer manner. We apply comparability to automate setting of the channel compression ratio for each layer in the entire model, whereas conventional methods set the ratio for each layer. We conducted various experiments using VGG-10/16, ResNet-18/50/56, MobileNet and SegNet models and the CIFAR-10/100, ImageNet, and CamVid datasets. The results showed that the proposed method can prevent accuracy degradation while achieving more effective compression than conventional methods by preferentially pruning the redundancy of these channels. The results of the ablation study suggest that our approach effectively extracts information for pruning from attention statistics. Analysis of attention statistics showed that there exist channels with a weak response to all classes and channels with a weak response to particular classes. We believe that the attention mechanism can also be a useful technique for channel pruning.

Acknowledgement

This paper is partly based on results obtained from a project commissioned by the New Energy and Industrial Technology Development Organization (NEDO).

References

[1] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2017.

[2] Vasileios Belagiannis, Azade Farshad, and Fabio Galasso. Adversarial network compression. In *European Conference on Computer Vision (ECCV) Workshops*, 2018.

[3] Gabriel J. Brostow, Julien Fauqueur, and Roberto Cipolla. Semantic object classes in video: A high-definition ground truth database. *Pattern Recognition Letters*, 2009.

[4] Francois Chollet. Xception: Deep learning with depthwise separable convolutions. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[5] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[6] Emily L Denton, Wojciech Zaremba, Joan Bruna, Yann LeCun, and Rob Fergus. Exploiting linear structure within convolutional networks for efficient evaluation. In *Advances in Neural Information Processing Systems (NIPS)*. 2014.

[7] Dongyoon Han, Jiwhan Kim, and Junmo Kim. Deep pyramidal residual networks. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[8] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. In *International Conference on Learning Representations (ICLR)*, 2016.

[9] Awni Y. Hannun, Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates, and Andrew Y. Ng. Deep speech: Scaling up end-to-end speech recognition. *CoRR*, abs/1412.5567, 2014.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[11] Yang He, Guoliang Kang, Xuanyi Dong, Yanwei Fu, and Yi Yang. Soft filter pruning for accelerating deep convolutional neural networks. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2018.

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

[13] Yihui He, Ji Lin, Zhijian Liu, Hanrui Wang, Li-Jia Li, and Song Han. Amc: Automl for model compression and acceleration on mobile devices. In *European Conference on Computer Vision (ECCV)*, 2018.

[14] Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. In *Deep Learning and Representation Learning Workshop in Neural Information Processing Systems (NIPS)*, 2015.

[15] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *ArXiv*, abs/1704.04861, 2017.

[16] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[17] Qiangui Huang, Shaohua Kevin Zhou, Suya You, and Ulrich Neumann. Learning to prune filters in convolutional neural networks. *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2018.

[18] Zehao Huang and Naiyan Wang. Data-driven sparse structure selection for deep neural networks. In *European Conference on Computer Vision (ECCV)*, 2018.

[19] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2016.

[20] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning (ICML)*, 2015.

[21] Jianxin Wu Jian-Hao Luo and Weiyao Lin. Thinet: A filter level pruning method for deep neural network compression. In *International Conference on Computer Vision (ICCV)*, 2017.

[22] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report.

[23] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. *International Conference on Learning Representations (ICLR)*, 2017.

[24] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *International Conference on Learning Representations (ICLR)*, 2014.

[25] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott E. Reed, Cheng-Yang Fu, and Alexander C. Berg. Ssd: Single shot multibox detector. In *European Conference on Computer Vision (ECCV)*, 2016.

[26] Volodymyr Mnih, Nicolas Heess, Alex Graves, and koray kavukcuoglu. Recurrent models of visual attention. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems (NIPS)*. 2014.

[27] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European Conference on Computer Vision (ECCV)*, 2016.

[28] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[29] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[30] Seiya Tokui, Kenta Oono, Shohei Hido, and Justin Clayton. Chainer: a next-generation open source framework for deep learning. In *Machine Learning Systems Workshop in Neural Information Processing Systems (NIPS)*, 2015.

[31] Diwen Wan, Fumin Shen, Li Liu, Fan Zhu, Jie Qin, Ling Shao, and Heng Tao Shen. Tbn: Convolutional neural network with ternary inputs and binary weights. In *European Conference on Computer Vision (ECCV)*, 2018.

[32] Fei Wang, Mengqing Jiang, Chen Qian, Shuo Yang, Cheng Li, Honggang Zhang, Xiaogang Wang, and Xiaoou Tang. Residual attention network for image classification. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[33] Ruichi Yu, Ang Li, Chun-Fu Chen, Jui-Hsin Lai, Vlad I. Morariu, Xintong Han, Mingfei Gao, Ching-Yung Lin, and Larry S. Davis. Nisp: Pruning networks using neuron importance score propagation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.