Channel Pruning Guided by Spatial and Channel Attention for DNNs in Intelligent Edge Computing

Mengran Liu[a], Weiwei Fang[a,b,*], Xiaodong Ma[a], Wenyuan Xu[a], Naixue Xiong[c], Yi Ding[d]

[a]*School of Computer and Information Technology, Beijing Jiaotong University, Beijing 100044, China*
[b]*Key Laboratory of Industrial Internet of Things & Networked Control, Ministry of Education, Chongqing 400065, China*
[c]*College of Intelligence and Computing, Tianjin University, Tianjin 300350, China*
[d]*School of Information, Beijing Wuzi University, Beijing 101149, China*

Abstract

Deep Neural Networks (DNNs) have achieved remarkable success in many computer vision tasks recently, but the huge number of parameters and the high computation overhead hinder their deployments on resource-constrained edge devices. It is worth noting that channel pruning is an effective approach for compressing DNN models. A critical challenge is to determine which channels are to be removed, so that the model accuracy will not be negatively affected. In this paper, we first propose Spatial and Channel Attention (SCA), a new attention module combining both spatial and channel attention that respectively focuses on "where" and "what" are the most informative parts. Guided by the scale values generated by SCA for measuring channel importance, we further propose a new channel pruning approach called Channel Pruning guided by Spatial and Channel Attention (CPSCA). Experimental results indicate that SCA achieves the best inference accuracy, while incurring negligibly extra resource consumption, compared to other state-of-the-art attention modules. Our evaluation on two benchmark datasets shows that, with the guidance of SCA, our CPSCA approach achieves higher inference accuracy than other state-of-the-art pruning methods under the same pruning ratios.

Keywords: Deep neural networks, Model compession, Channel pruning, Attention module

1. Introduction

Intelligent edge, which refers to the combination of edge computing with artificial intelligence (AI), machine learning, big data and other technologies, is regarded as a crucial step in the new revolution of the Internet of things (IoT) domain. Intelligent edge collects, stores, and processes data in real time by edge devices. It can not only make full use of the computing power of edge devices, but also reduce the bandwidth consumption, improve the response speed, and minimize the risk of data privacy leakage, as compared with traditional cloud computing. However, there exist many challenges for deploying AI techniques on edge devices. As the most popular AI technique, DNNs (Deep Neural Networks) have achieved remarkable success in many application scenarios, ranging from the initial handwriting recognition [1] to image classification [2], object detection [3], and visual tracking [4]. DNNs have displaced conventional computer vision approaches to a great extent, since they can provide near-human or even better-than-human accuracy in practice [5]. However, DNNs are known to be both compute-intensive and memory-intensive. For example, VGG16 has 138M weights and requires 15.5G multiply-and-accumulates for one input image, and ResNet50 has 25.5M weights and requires 3.9G multiply-and-accumulates per image [5]. Thus, most DNN models currently are difficult to be deployed on resource-constrained Internet-of-Things (IoT) devices and in performance-demanding edge-computing applications. A critical challenge is how to compress the DNN models to reduce computational requirements and resource consumption without negatively affecting their accuracy [6, 7].

In this context, DNN compression techniques have been intensively studied, e.g., parameter pruning [8], low-rank factorization [9], weight quantization [10], knowledge distillation [11], etc. Among them, the pruning-based methods aim to compress DNN models and accelerate DNN inference by removing redundancy in structures and parameters.

Except for early works on non-structured weight pruning [10], the structured pruning approach represented by channel pruning becomes more popular in recent years, since it does not require using specialized software and hardware. The basic idea of channel pruning is to reduce the number of input/output channels in the convolutional layer with negligible performance degradation [12, 13]. A key step in channel pruning is to measure the importance of each channel, which determines if the channel could be removed or not. Early studies only take the independent contribution of each channel to reconstruction loss into consideration [13, 14, 12], and neglect the impact of correlations between neighboring channels to inference performance.Besides, the tradeoff between accuracy and pruned ratio is a noteworthy problem. In order to achieve a better balance between the pruned ratio and accuracy, the work in [15] proposed a efficient approach to channel pruning, based on the genetic algorithm and sparse learning, and another work [16] proposed a scheme of network compression from the perspective of multi-objective evolution. However, the accuracy and pruned ratio of the existing methods need to be improved.

Attention mechanism is a good alternative for measuring the important level of channels and the inter-channel relationship of features [17]. It is inspired by human perception, in which our central nervous system tends to focus on salient parts from a series of partial sense-organ input so as to capture perceptual information better [18]. Attention not only tells us where to focus, but also helps us to improve the representation of subject of interest. This mechanism initially attracts widespread interest in the field of natural language processing [19], and then achieves a lot of promising results in emerging computer vision applications. For image classification tasks, there have been several attempts to incorporate attention processing to improve the inference performance of DNN models. Typical attention modules for image classification include Residual Attention Network [20], SENet [21], BAM [22], CBAM [17] and SGE [23].

In this paper, we propose a new channel pruning approach called Channel Pruning guided by Spatial and Channel Attention (CPSCA), in which we combine both spatial attention and channel attention together to determine the channel importance and guide the pruning operation. At first, we design a simple yet effective Spatial and Channel Attention module (SCA). This SCA module can emphasize useful features and suppress useless features along both the spatial and channel dimensions, so as to boost representation power of current DNN model. Meanwhile, it can generate a scale value for each individual channel to measure its important level to the classification task. Using this scale value, we develop a CPSCA pruning approach to prune redundant and unimportant channels from the original DNN model, so as to well reduce resource consumption in terms of computation and storage while incurring negligible accuracy loss. On the one hand, compared with the existing attention designs (e.g., [21, 17, 23]), SCA adopts pooling and group normalization to improve the performance of spatial and channel attention submodules, respectively. On the other hand, compared with the exiting pruning techniques (e.g., [8, 24, 25]), CPSCA is more capable to classify and prune unimportant channels so as to well preserve the inference accuracy.

The main contributions of this work are summarized as follows:

- We design a new light-weight attention module SCA, which combines both spatial and channel attention as a whole. It can not only be inserted into the DNN model in a plug-and-play fashion to enhance representation power, but also output the scale values as a more reliable measure of importance for the channels of the DNN model.

- Guided by SCA, we further propose a new attention-based channel pruning approach CPSCA. This approach measures the important level of channels based on the attention statistics from the SCA module. It then removes the unimportant channels from the DNN model, and retrains (fine-tunes) the model for loss recovery.

- We verify the effectiveness of the optimal structure design of SCA through ablation studies, and demonstrate the superiority of this module by comparing it with the state-of-the-art attention modules using CIFAR datasets and VGG/ResNet models.

- We conduct extensive experiments on CIFAR-10 and CIFAR-100 datasets, and the results show that our CPSCA pruning approach achieves higher inference accuracy than other state-of-the-art pruning methods under the same pruning ratios.

The remainder of this paper is organized as follows. In Section 2, we briefly review related works. Section 3 introduces the proposed SCA attention module and the CPSCA approach. In Sections 4, we evaluate the performance of SCA and CPSCA using the standard datasets CIFAR-10 and CIFAR-100 on image classification. Section 5 finally concludes the paper.

2. Related Work

Pruning approaches. The pruning approaches for DNNs can be generally classified as two categories, i.e., non-structured pruning and structured pruning. Early studies [10, 26, 27, 28] are mainly based on weight pruning, resulting in non-structured sparsity in the pruned model. The runtime acceleration is difficult to be achieved because of irregular memory access [29], unless using specialized hardware and libraries. Pioneered by [29, 12], structured pruning overcomes this problem by removing whole filters or channels and producing a non-sparse compressed model [29, 27, 30]. Structured pruning can be classified into four types according to pruning granularities, including layer pruning, filter pruning, channel pruning and kernel pruning. It is noteworthy that channel pruning and filter pruning are correlated, because pruning the channel of current layer will cause the corresponding filter of the upper layer to be removed [12]. In this paper, we focus on channel pruning, which targets at removing a certain number of channels and the relevant filters to compress DNN models. ℓ_1-norm based work [8] used the ℓ_1-norm of filters as the pruning criterion. Network Slimming [24] adopted the Batch Normalization (BN) layers to scale different channels, and identified the channels with relatively smaller scale factors as the unimportant ones to be pruned. Some studies proposed to prune the channels that have the smallest impact on the feature reconstruction error between the original model and the pruned model. The work in [13] proposed a greedy search based method to minimize the reconstruction error, and another work [12] retained the representative channels by solving a lasso regression problem about the reconstruction error. Both of them only considered local statistics of two consecutive layers, i.e., prune one layer to minimize the next layer's reconstruction error. Considering the effect of potential error propagation in the entire network, NISP [14] proposed to minimize the reconstruction errors using the global importance scores propagated from the second-to-last layer before classification. It must be noted that a good metric for channel pruning should take not only the channel importance from a global view but also the correlation between different channels [14, 21].

Attention mechanisms. The attention mechanism can effectively help to improve the classification performance of DNNs [31], by enhancing the representation of feature map with more important information and suppressing the interference of unnecessary information [32, 33]. Attention has been widely used in recent applications, e.g., neural machine translation [19], image captioning [34], object detection [35], and generative modeling [36]. To improve the performance of image classification, SENet (Squeeze-and-Excitation Networks) [21] proposed a light-weight gating mechanism to exploit the channel-wise relationships. It is actually an attention mechanism applied along the channel dimension, but neglects the importance of the spatial dimension. The SGE (Spatial Group-wise Enhance) module [23] is designed to enhance the ability of each of its feature groups to express different semantics while suppressing possible noise and interference. It is essentially a spatial attention module that misses the spatial attention. Both CBAM (Convolutional Block Attention Module) [17] and BAM (Bottleneck Attention Module) [22] exploit both spatial-wise and channel-wise attention, and verify that combining both is superior to using either of them. The structure design of these two modules are different. CBAM sequentially apply channel and spatial attention, while BAM computes the two attentions in a simultaneous way.

Attention-based pruning methods. In recent years, attention mechanisms have also been introduced for improving the performance of model pruning. For example, the work in [25] proposed to apply the SENet model to evaluate channel importance, so that the redundant channels with least importance can be identified and removed. However, the limitation of SENet itself makes the scale value generated by this model can not fully reflect the channel importance and improve the pruning performance. PCAS [37] designed a new attention model and evaluated the importance of channels based on attention statistics. Actually, the module in PCAS is only a channel attention module, and has two fully connected layers that incur additional overhead and complexity. Moreover, the operations of dimensionality reduction bring side effect [38, 39] on channel attention prediction. To address these problems, we propose a new attention module that exploits both spatial and channel-wise attention based on an efficient structure design, and verify its superior performance over state-of-the-art solutions in the process of channel pruning.

3. Our Proposed CPSCA Methodology

In this section, we first present an overview of our CPSCA approach. Next, we introduce the structural composition of the SCA attention module. Finally, we propose the CPSCA algorithm that prunes DNN models with the guidance of SCA.

3.1. Overview of Our CPSCA Approach

Fig. 1 depicts the overview of our CPSCA approach. Firstly, we insert our SCA modules which can reflect the importance of channels into the original network, and then train the resulting network. According to the statistical channel scales generated by SCA and the pre-defined channel pruning ratios, a certain number of channels with least importance are identified. After removing all inserted SCA modules, we then prune the network by removing the identified channels as well as the filters corresponding to these channels. At last, we finetune the pruned network to recover the accuracy loss caused by pruning.
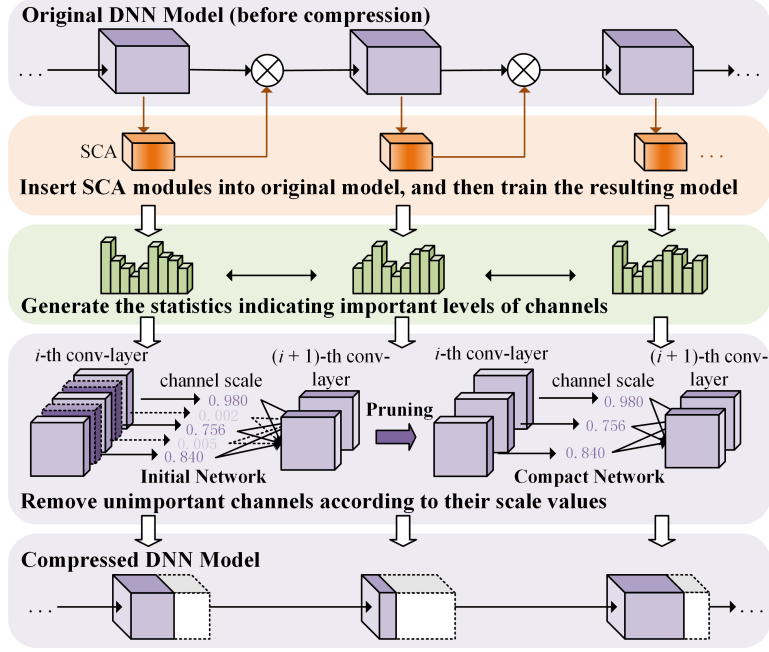

Fig. 1. Overview of the CPSCA approach.

Fig. 2 shows the overall structure design of our SCA (Spatial and Channel Attention) module. In fact, if we only employ spatial attention, the information in the channel dimension will be ignored, as it treats the features in different channels equally. Similarly, if we only employ channel attention, the information inside of channels will also be ignored. Thus, we believe the combination of spatial and channel attention modules as a whole module will achieve higher performance.
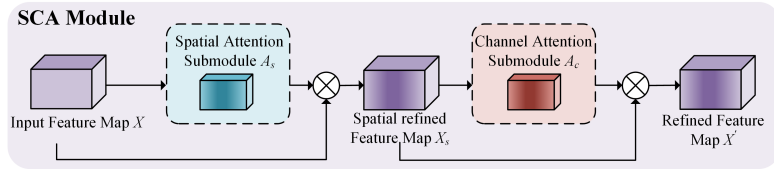

Fig. 2. Overall structure of the SCA module.

3.2. SCA Module

Given the input feature map $X \in \mathbb{R}^{C \times H \times W}$, the spatial attention submodule firstly infers the spatial attention map $A_s \in \mathbb{R}^{1 \times H \times W}$, then we can obtain the spatial refined feature map $X_s \in \mathbb{R}^{C \times H \times W}$ by:

$$X_s = X \otimes A_s(X), \tag{1}$$

4

where $\otimes$ denotes element-wise multiplication. Based on X_s, the channel attention submodule further deduces the channel attention map $A_c \in \mathbb{R}^{C \times 1 \times 1}$, and then generates the final refined feature map $X^{'} \in \mathbb{R}^{C \times H \times W}$ by:

$$X^{'} = X_s \otimes A_c(X_s). \tag{2}$$

The computation process of spatial and channel attention submodules are introduced in the following subsections.

3.2.1. Spatial Attention Submodule

The spatial attention submodule focuses on 'where' are the informative parts, and pays differentiated attention to different positions on the feature map. For a feature map $X \in \mathbb{R}^{C \times H \times W}$, the spatial attention map $A_s \in \mathbb{R}^{1 \times H \times W}$ is corresponding to a $H \times W$ matrix, in which each position denotes a weight corresponding to a pixel of original feature map.

The SGE attention module [23] has verified global avg-pooling is helpful to performance improvement. The reason is that avg-pooling calculates the mean value in the feature map region as the resulting pooled value, and can diminish the bias of estimated average value as well as improve the robustness of model. Based on this observation, we further introduce global max-pooling which calculates the maximum value in the feature map region as the resulting pooled value, as it is able to learn the edge information and texture structure of the feature map. We believe the combination of both these pooling methods can effectively aggregate the spatial information. Fig. 3 depicts the computation process of spatial attention submodule. The detailed computation process is as follows.
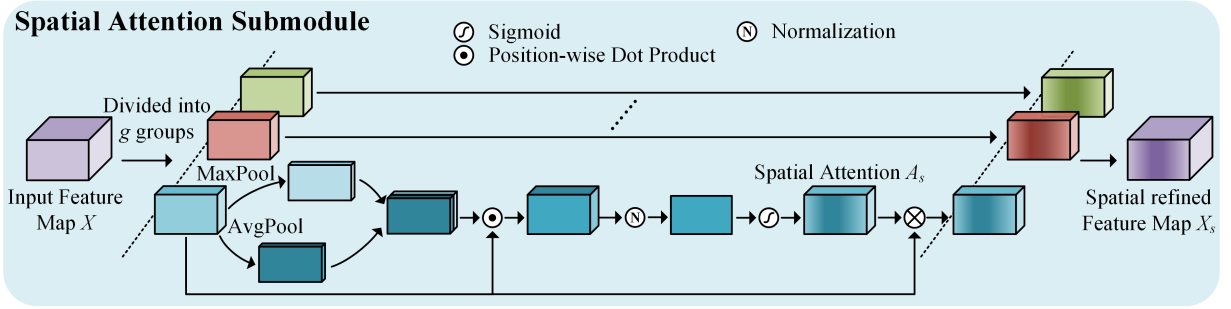

Fig. 3. Diagram of spatial attention submodule.

(1) The input feature map $X \in \mathbb{R}^{C \times H \times W}$ is divided into groups along the direction of the channel. The grouping strategy is adopted due to two reasons. Firstly, it can well reduce the model computation overhead [23]. Secondly, as revealed in [40], the learning process of DNNs can be regarded as a process of gradually capturing specific semantic responses, such as the bird's beak in the bird recognition task [41]. In this group space, ideally, we can obtain features with strong responses at the beak position, while other positions become zero vectors as no activation exists. However, it is generally difficult for DNN to obtain uniformly distributed feature responses, due to the existence of unavoidable noise and similar patterns [23]. Therefore, the overall information of the whole group space can be used to further enhance semantic feature learning at key regions.

Firstly, In the spatial dimension, each position of the group can be represented by the following vectors:

$$\boldsymbol{p} = \{\mathbf{P}_1, \ldots, \mathbf{P}_n\}, \mathbf{P}_i \in \mathbb{R}^{\frac{C}{g}}, n = H \times W, \tag{3}$$

where g is a pre-defined hyper-parameter denoting the number of groups, and $\mathbf{P}_i$ denotes the local statistical feature. The similarity between the global statistical feature and the local one at each position can generate the spatial attention map. The global statistical feature can be obtained through avg-pooling and max-pooling:

$$\mathbf{f} = cat(AvgPool(\boldsymbol{p}), MaxPool(\boldsymbol{p})), \tag{4}$$

where cat denotes the concatenate operation.

(2) For each position $i \in \{1, \ldots, n\}$, the similarity between the global and local statistical features can be obtained by the simple dot product:

5

$$W_i = \mathbf{f} \cdot \mathbf{P}_i = \|\mathbf{f}\| \times \|\mathbf{P}_i\| \times \cos \theta_i, \tag{5}$$

where θ_i is the angle between $\mathbf{f}$ and $\mathbf{P}_i$ [23]. To avoid bias among the coefficients of various samples, we then adopt normalization [42, 43, 44] to W_i:

$$\hat{N}_i = \frac{W_i - \mu_w}{\sigma_w + \varepsilon}, \tag{6}$$

$$\mu_w = \frac{1}{n} \sum_j^n W_j, \tag{7}$$

$$\sigma_w^2 = \frac{1}{n} \sum_j^n (W_j - \mu_w)^2, \tag{8}$$

where ε (e.g., 1e-5) is a constant added for numerical stability.

(3) The sigmoid function is used to calculate the final spatial attention map as follows:

$$A_s = Sigmoid(N_i). \tag{9}$$

3.2.2. Channel Attention Submodule

Different from spatial attention, the channel attention submodule focuses on 'what' are the informative parts and pays differentiated attention to different channels of feature map. For a feature map $X \in \mathbb{R}^{C \times H \times W}$, channel attention map $A_c \in \mathbb{R}^{C \times 1 \times 1}$ is corresponding to a $C \times 1 \times 1$ matrix, in which each position denotes a weight corresponding to a channel of original feature map.

The previously designed attention modules, e.g., SENet and CBAM, use two Fully-Connected (FC) layers to process channel information. There exist three drawbacks for such a design. Firstly, it limits the total number of attention modules that can be inserted into the original model [45]. Secondly, it becomes difficult to analyze the relationship between different layers of channels due to the complexity of parameters in FC layers. Actually, capturing the dependencies between all channels is inefficient and unnecessary. Thirdly, dimensionality reduction has to be involved to control model complexity, which has side effects on channel attention prediction [46].

To address these problems, we propose to use normalization to model the relationship of channels. Compared to the FC layers used in SENet and CBAM, Batch Normalization (BN) can generate competition relationship between channels, using much fewer resource cost while providing more stable training performance [45]. In this work, we choose Group Normalization (GN) [44], as a simple alternative of BN, to replace the design with two FC layers. In GN, the channels are divided into groups, and the mean and variance for normalization are computed within each group. As the computation of GN is independent of batch sizes, it can outperform BN and other normalization methods. Fig. 4 illustrates the computation process of channel attention submodule. The detailed computation process is as follows.
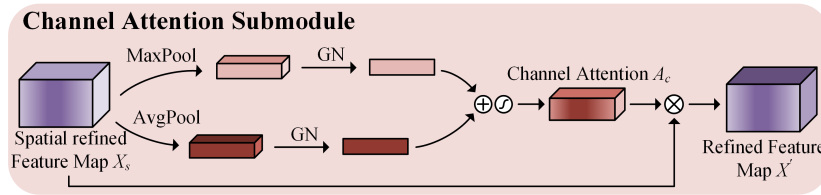

Fig. 4. Diagram of channel attention submodule.

(1) Due to similar aforementioned reasons, both avg-pooling and max-pooling are adopted in the channel attention submodule to aggregate the feature map in each channel, so as to generate two different spatial context descriptors as follows:

$$A = AvgPool(X_s), \tag{10}$$

$$M = MaxPool(X_s), \tag{11}$$

where A denotes the average-pooled features, and M denotes the max-pooled features.

(2) Then, A and M are normalized respectively by GN. Take A as an illustration. Given A, GN performs the following computation:

$$\hat{A}_i = \frac{1}{\sigma_i}(A_i - \mu_i), \tag{12}$$

where i denotes the index. For 2D images, $i = (i_N, i_C, i_H, i_W)$ is a 4D vector indexing the features in the order of (N, C, H, W), where N is the batch axis. Meanwhile, μ and σ in (12) are the mean and the standard deviation (std), respectively. They can be calculated as:

$$\mu_i = \frac{1}{m} \sum_{K \in \Gamma_i} A_K, \tag{13}$$

$$\sigma_i = \sqrt{\frac{1}{m} \sum_{K \in \Gamma_i} (A_K - \mu_i)^2 + \epsilon}, \tag{14}$$

where ϵ is a small constant, Γ_i is the set of pixels in which the mean and the std are computed, and m is the size of this set. Actually, the primary difference between the various feature normalization methods is the different definition of Γ_i [42, 47, 44]. In GN, this set is defined as:

$$\Gamma_i = \{K | K_N = i_N, \lfloor \frac{K_C}{C/G} \rfloor = \lfloor \frac{i_C}{C/G} \rfloor\}, \tag{15}$$

where G is a pre-defined hyper-parameter denoting the number of groups, and C/G is the number of channels in each group. The normalization operation for M follows a similar procedure, and therefore is omitted for simplicity.

(3) We merge the normalized output feature vectors using element-wise summation, and generate the 3D channel attention map A_c via a sigmoid function. The final channel attention map is computed as:

$$\begin{aligned} A_c &= Sigmoid(GN(MaxPool(X_s)) + GN(AvgPool(X_s))) \\ &= Sigmoid(GN(M) + GN(A)). \end{aligned} \tag{16}$$

3.2.3. How to Combine Two Attention Submodules

The spatial and channel attention submodules can complement each other very well on image classification, as they focus on 'where' and 'what' is meaningful respectively. How to arrange them has a dramatic impact on the final performance, and should be taken into account when design. Actually, they can be combined in a sequential manner (e.g., CBAM), or in a parallel manner (e.g., BAM). By experiments, we found that the sequential arrangement with the spatial-first order achieves the best result. That's why we name our module as "SCA". Detailed experimental results will be presented in the next section.

3.3. Channel Pruning Guided by SCA

Attention mechanism can explicitly describes the importance relationship between channels in the same layer, and constantly adjusts the parameters of fully-connected layers in the process of back-propagation [18, 17]. By inserting the attention module, the network is capable of showing the trend of enhancing or suppressing a portion of channels gradually. We propose to use the channel scale that is a statistical quantity found by element-wise averaging of the weight in the channel attention map over all training data, as a criterion for measuring channel importance. Note that for different input data (i.e.,images), the attention module will output different weight outputs for the same channel [48], as illustrated by the experimental results in Fig. 5. As a result, the channel importance in CPSCA is measured in a statistical fashion for fairness and precision [37]. The channel scale for a given channel j is given as follows:

$$w_j = \frac{1}{|D|} \sum_{d \in D} S_j(A_c^d), j \in \{1, \ldots, C_l\}, l \in \{1, \ldots, L\}. \tag{17}$$
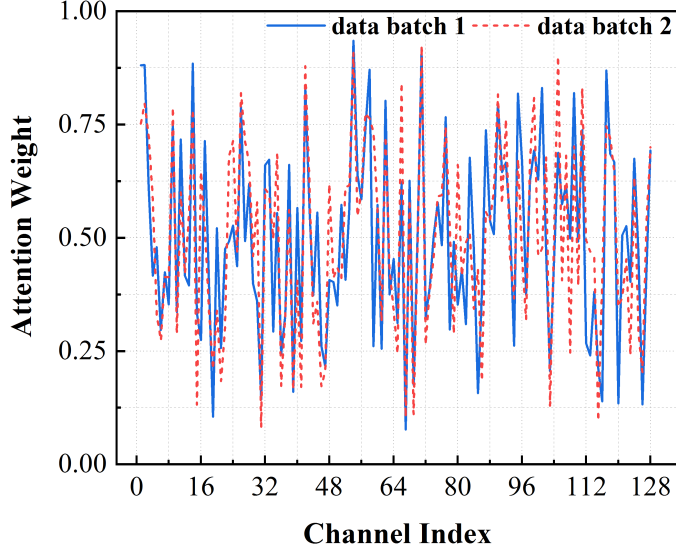
Fig. 5. An illustration of comparison results for the attention weights generated from different input data.

where D denotes the set of training data, A_c^d denotes the channel attention map for input data d, L denotes the number of layers, and S_j denotes the function that extracts the weight value for channel j from the $C \times 1 \times 1$ channel attention matrix.

The overall channel pruning process in CPSCA is summarized in Algorithm 1.

4. Performance Analysis

In this section, we evaluate SCA and CPSCA on the standard benchmarks CIFAR-10 and CIFAR-100 for image classification. The CIFAR-10 dataset consists of 60000 32×32 color images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images. The CIFAR-100 dataset is just like the CIFAR-10, except it has 100 classes containing 600 images each. There are 500 training images and 100 testing images per class. We perform channel pruning on VGGs and ResNets. Firstly, we perform extensive ablation experiments to fully evaluate the effectiveness of the final SCA module. Next, we demonstrate the applicability and performance of SCA across typical architectures and different datasets, as compared to the previously published attention modules. At last, we show that with the guidance of SCA, our CPSCA approach outperforms the state-of-the-art pruning methods. Moreover, we deploy the pruned model onto NVIDIA Jetson Nano [49] to demonstrate the execution speedup of DNN models on the real edge platform. All the evaluation models are trained on an 8 Nvidia Titan-X GPUs server using Pytorch.

4.1. Ablation studies

This subsection shows the effectiveness of our design choice for the attention module. We first carefully search for the optimal structure design of the spatial attention, and then the channel attention. Then, we find out the best way of arrange the spatial and channel attention submodules through comparison experiments. We will explain the experiments for this module design process in detail as follows.

4.1.1. Spatial Attention Submodule

In this part of experiments, we only place the spatial attention submodule in SCA.

Algorithm 1 CPSCA Algorithm

Input:
  M: the original DNN model
  L: the number of layers in M
  D: the set of training data
  w_j: the weight value for channel j output by SCA
  $\{p_l\%|l \in \{1, ..., L\}\}$: the pruning ratios for all L layers
Output:
  $M^{'}$: the pruned DNN model
 1: **Procedure**
 2: Insert SCA modules into model M
 3: **for** each batch $d \in D$ **do**
 4: Train model M with d
 5: **for** each layer $l \in \{1, ..., L\}$ **do**
 6:   **for** each channel $j \in \{1, ..., C_l\}$ **do**
 7:    Calculate the attention weight $S_j(A_c^d)$ for j
 8:   **end for**
 9: **end for**
10: **end for**
11: **for** each layer $l \in \{1, ..., L\}$ **do**
12: **for** each channel $j \in \{1, ..., C_l\}$ **do**
13:   $w_j = \frac{1}{|D|}\sum_{d \in D} S_j(A_c^d)$
14: **end for**
15: Sort all the C_l channels by $w_j, j \in \{1, \ldots, C_l\}$
16: **end for**
17: Remove all inserted SCA modules
18: Prune all layer in model M, i.e., removing a proportion $p_l\%$ of channels with the least scale values from layer l.
  Note that the filters in the next convolutional layer corresponding to the pruned channels are also removed
19: Finetune the model for accuracy recovery, and obtain M'
20: **return** $M^{'}$

Groups: In the spatial attention submodule, we first investigate the impact of the number of groups, g, which denotes the number of different semantic sub-features. When the number of channels in the same convolutional layer is fixed, too few groups are not conducive to semantic diversity; On the contrary, too many groups will make feature representation for each semantic response limited. From Fig. 6(a), we can observe that $g = 64$ tends to produce better prediction results than the others. It's a moderate value that can balance semantic diversity and representation ability of each semantic to optimize inference performance.

Pooling: In order to verify the effectiveness of the joint using of both poolings in spatial attention module, we compare three different pooling strategies: avg-pooling, max-pooling and joint use of both poolings. The results in Fig. 6(b) confirm that joint using both avg-pooling and max-pooling significantly enhances representation power of DNN models, resulting in higher accuracy than using each independently. That's because avg-pooling extracts features smoothly while max-pooling focuses only on the most salient features. It is better to use them simultaneously to make them compensate with each other.

As a brief summary, we use both poolings in our spatial attention submodule, and the number of groups $g = 64$ in the following experiments.

4.1.2. Channel Attention Submodule

After the spatial-wise refined features are given, we can further explore how to effectively compute the channel attention. In this part of experiments, we place the channel attention submodule just after the previously designed spatial attention submodule in SCA, since our ultimate goal is to combine them together.
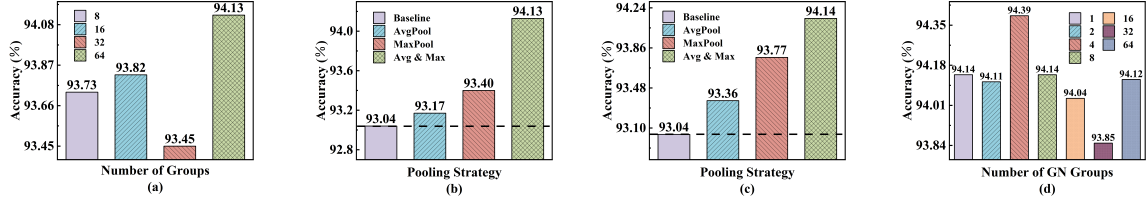
Fig. 6. (a) Number of groups g vs. performance. Evaluation uses VGG16 on CIFAR-10 dataset with both poolings. (b) Pooling strategies vs. performance. Evaluation uses VGG16 on CIFAR-10 dataset with $g = 64$. (c) Pooling strategies vs. performance. Evaluation uses VGG16 on CIFAR-10 dataset with $G = 1$. (d) Number of GN groups G vs. performance. Evaluation uses VGG16 on CIFAR-10 dataset with both poolings.

Pooling: Same as that in spatial attention, both avg-pooling and max-pooling are adopted in the channel attention module. Fig. 6(c) shows the experimental results with the typical pooling strategies. Similar to the results in spatial attention, all the three pooling strategies outperform the baseline which does not use pooling, and the best accuracy result can be achieved by joint using both poolings.

GN groups: In the channel attention submodule, we apply Group Normalization (GN) to both avg-pooled and max-pooled features simultaneously. In the experiments, the setting of group division is the same as that in [44]. From Fig. 6(d), GN performs well for all values of G, and the inference accuracies increase observably by 0.81%-1.35% as compared to the baseline. Meanwhile, the results indicate that setting $G = 4$ achieves the best performance among all the options.

As a brief summary, we also use both poolings in our channel attention submodule, and the number of GN groups $G = 4$ in the following experiments.

4.1.3. Arrangement of the spatial and channel attention

After determining the suitable settings for each of the two submodules, we compare five different ways of arranging these submodules in SCA (i.e., only channel, only spatial, sequential spatial and channel, sequential channel and spatial, and parallel use of both submodules), to investigate how the existence and the order of submodules could affect the overall performance. From the results summarized in Table 1, all the arranging patterns outperform the baseline, indicating the advantage of applying attention mechanisms. We can also observe that the sequential arrangement surpasses the parallel arrangement, and the spatial-first order achieves better results than the channel-first order. According to the comparison results, we choose to arrange the spatial and channel attention submodules sequentially in SCA.

Table 1: Comparison of different combining methods using VGG16 on CIFAR-10 dataset.

Description	Params	GFLOPs	Acc(%)
VGG16 (baseline)	16.87M	0.63163	93.04
VGG16 + Spatial	16.87M	0.63163	94.13
VGG16 + Channel	16.88M	0.63163	93.82
VGG16 + Spatial + Channel (ours)	16.88M	0.63163	**94.39**
VGG16 + Channel + Spatial	16.88M	0.63163	93.61
VGG16 + Channel & Spatial in parallel	16.88M	0.63163	93.57

4.2. Comparisons with State-of-the-art Attention Modules

We perform image classification experiments to evaluate our SCA module in two popular backbone architectures, i.e., VGG and ResNet. In VGG networks the SCA module is placed at every convolutional block, and in ResNet networks the SCA module is placed on the convolution outputs in each ResBlock. We compare SCA against several state-of-the-art attention modules, including SENet, CBAM and SGE, based on CIFAR-10 and CIFAR-100 datasets. The evaluation metrics consist of two aspects: efficiency (i.e., parameter size, and computation cost) and effectiveness (i.e., Top-1 accuracy and accuracy improvement). The comparison results are summarized in Table 2 and Table 3.

Table 2: Comparison of different attention modules on CIFAR-10 dataset.

Model	Variant	Params	GFLOPs	Acc(%)	
VGG16	baseline	16.87M	0.63163	93.04	—
	+ SENet	17.10M	0.63209	93.50	↑0.46
	+ CBAM	17.10M	0.63310	93.19	↑0.15
	+ SGE	16.87M	0.63163	93.12	↑0.08
	+ SCA	16.88M	0.63163	**94.39**	**↑1.35**
VGG19	baseline	22.18M	0.80158	92.87	—
	+ SENet	22.49M	0.80219	93.33	↑0.46
	+ CBAM	22.49M	0.80336	93.59	↑0.72
	+ SGE	22.18M	0.80158	92.92	↑0.05
	+ SCA	22.19M	0.80158	**93.97**	**↑1.10**
ResNet56	baseline	0.85M	0.25257	92.98	—
	+ SENet	0.87M	0.25260	93.53	↑0.55
	+ CBAM	0.87M	0.25742	93.61	↑0.63
	+ SGE	0.85M	0.25257	93.78	↑0.80
	+ SCA	0.86M	0.25257	**93.86**	**↑0.88**
ResNet110	baseline	1.73M	0.50893	93.21	—
	+ SENet	1.76M	0.50898	93.53	↑0.32
	+ CBAM	1.77M	0.51861	93.43	↑0.22
	+ SGE	1.73M	0.50893	93.81	↑0.60
	+ SCA	1.74M	0.50893	**93.87**	**↑0.66**

Table 3: Comparison of different attention modules on CIFAR-100 dataset.

Model	Variant	Params	GFLOPs	Acc(%)	
VGG16	baseline	17.24M	0.63237	73.14	—
	+ SENet	17.47M	0.63282	74.87	↑1.73
	+ CBAM	17.47M	0.63384	75.14	↑2.00
	+ SGE	17.24M	0.63237	73.25	↑0.11
	+ SCA	17.25M	0.63237	**75.24**	**↑2.10**
VGG19	baseline	22.55M	0.80232	73.02	—
	+ SENet	22.85M	0.80292	74.34	↑1.32
	+ CBAM	22.86M	0.80410	74.12	↑1.10
	+ SGE	22.55M	0.80232	73.62	↑0.60
	+ SCA	22.56M	0.80232	**74.74**	**↑1.72**
ResNet56	baseline	0.86M	0.25258	75.78	—
	+ SENet	0.87M	0.25261	76.12	↑0.34
	+ CBAM	0.88M	0.25743	76.09	↑0.31
	+ SGE	0.86M	0.25258	76.28	↑0.50
	+ SCA	0.86M	0.25258	**76.73**	**↑0.95**
ResNet110	baseline	1.73M	0.50894	75.87	—
	+ SENet	1.76M	0.50899	76.06	↑0.04
	+ CBAM	1.77M	0.51862	76.18	↑0.31
	+ SGE	1.74M	0.50894	76.15	↑0.28
	+ SCA	1.75M	0.50894	**76.27**	**↑0.40**

Based on the results, we can make several observations. Firstly, the networks with SCA outperform all the counterparts in terms of inference accuracy across all four architectures and both datasets. The performance improvement verifies that SCA is an effective attention module, benefiting from the combination of both attentions as well as the

11

Table 4: Comparison of different pruning approaches using VGG16 on CIFAR-10 and CIFAR-100 dataset.

Method	CIFAR-10						CIFAR-100					
	Params	Pruned	GFLOPs	Pruned	Time(ms)	Acc(%)	Params	Pruned	GFLOPs	Pruned	Time(ms)	Acc(%)
baseline	16.87M	—	0.63163	—	12.3	93.04 —	17.24M	—	0.63237	—	13.5	73.14 —
ℓ_1-norm						93.25 ↑0.21						72.97 ↓0.17
Slimming	11.43M	32.25%	0.47499	24.75%	10.2	93.35 ↑0.31	11.47M	33.47%	0.47509	24.87%	9.6	73.34 ↑0.20
CPSE						93.43 ↑0.39						73.36 ↑0.22
CPSCA						**93.45 ↑0.41**						**73.77 ↑0.63**
ℓ_1-norm						92.98 ↓0.06						71.76 ↓1.38
Slimming	8.52M	49.50%	0.21988	65.05%	4.5	93.01 ↓0.03	8.57M	50.29%	0.21997	65.21%	4.2	71.95 ↓1.19
CPSE						92.09 ↓0.95						72.13 ↓1.01
CPSCA						**93.10 ↑0.06**						**72.89 ↓0.25**
ℓ_1-norm						92.13 ↓0.91						70.19 ↓2.95
Slimming	3.38M	79.96%	0.14407	77.02%	3.0	92.78 ↓0.26	3.89M	77.44%	0.18677	70.47%	3.9	71.33 ↓1.81
CPSE						92.89 ↓0.15						71.09 ↓2.05
CPSCA						**92.96 ↓0.08**						**72.03 ↓1.11**
ℓ_1-norm						88.90 ↓4.14						69.87 ↓3.27
Slimming	0.72M	95.73%	0.09918	84.12%	2.1	89.79 ↓3.25	0.77M	95.53%	0.09927	84.30%	2.1	70.25 ↓2.89
CPSE						90.11 ↓2.93						70.87 ↓2.27
CPSCA						**91.39 ↓1.65**						**71.67 ↓1.47**

Table 5: Comparison of different pruning approaches using ResNet56 on CIFAR-10 and CIFAR-100 dataset.

Method	CIFAR-10						CIFAR-100					
	Params	Pruned	GFLOPs	Pruned	Time(ms)	Acc(%)	Params	Pruned	GFLOPs	Pruned	Time(ms)	Acc(%)
baseline	0.86M	—	0.25257	—	30.6	92.98 —	0.86M	—	0.25258	—	30.9	75.78 —
ℓ_1-norm						92.54 ↓0.44						74.89 ↓0.89
Slimming	0.45M	47.67%	0.16983	32.76%	20.7	92.26 ↓0.72	0.45M	47.67%	0.16984	32.76%	21.0	75.60 ↓0.18
CPSE						92.55 ↓0.43						75.55 ↓0.23
CPSCA						**93.06 ↑0.08**						**75.87 ↑0.09**
ℓ_1-norm						90.48 ↓2.50						73.18 ↓2.60
Slimming	0.17M	80.23%	0.06476	74.36%	8.1	91.09 ↓1.89	0.18M	79.07%	0.06477	74.36%	7.5	73.19 ↓2.59
CPSE						91.28 ↓1.70						73.22 ↓2.56
CPSCA						**91.31 ↓1.67**						**73.34 ↓2.44**
ℓ_1-norm						89.32 ↓3.66						71.23 ↓4.55
Slimming	0.04M	95.35%	0.01351	94.65%	1.5	89.17 ↓3.81	0.05M	94.19%	0.01352	94.65%	1.6	71.57 ↓4.21
CPSE						88.96 ↓4.02						71.86 ↓3.92
CPSCA						**90.18 ↓2.80**						**72.61 ↓3.17**

adoption of new pooling strategy and grouping operations. Secondly, the results show that SCA can efficiently raise predictive accuracy with negligible extra parameters and computation overhead. Though both SCA and CBAM exploit both spatial and channel-wise attention, CBAM actually incurs noticeable overheads but can't achieve accuracy as high as that of SCA. The reason is that, the grouping operation in the spatial submodule and the replacement of the FC layers with GN operation in the channel submodule can greatly reduce both the computation overhead and the redundant parameters of SCA. Thirdly, SCA is shown to be more capable of helping the shallower models (e.g., VGG16 and ResNet56) other than the deeper ones (e.g., VGG19 and ResNet110) to boost their representation power [17] and improve their inference performance.

4.3. Comparisons with State-of-the-art Pruning Methods

We have verified that our SCA achieves the best performance among all the presented attention modules. As a result, the scale values generated by SCA are more convincing, and can better describe the important levels of different channels. In this part, we compare the CPSCA pruning approach with the representative pruning schemes, including ℓ_1-norm [8] and Slimming [24]. Additionally, we also apply the most classic attention module SENet for

pruning [25] and name the approach CPSE (Channl Pruning guided by SENet), and use the result as a comparative reference. Compared with existing pruning algorithms, our CPSCA approach needs to embed the SCA modules into the original network before pruning, and remove all of them after obtaining the attention statistics. Thus, it may be more laborious in operation. Table 4 and Table 5 compare the obtained accuracy of different solutions under the same pruning ratios on CIFAR-10 and CIFAR-100, respectively. The column of "Time" in the tables is the average inference time for one image on the NVIDIA Jetson Nano edge platform.

From the results in Table 4 and Table 5, our CPSCA approach consistently outperforms the other state-of-the-art pruning methods, which demonstrates the effectiveness of CPSCA on the two datasets. It is also worth mentioning that when the pruning ratio is relatively small, CPSCA may have a higher accuracy than that of the original model. For example, CPSCA improves the accuracy of VGG16 by 0.41% on CIFAR-10 dataset when the pruning ratio is 32.25%, and improves the accuracy of ResNet56 by 0.09% on CIFAR-100 dataset when the pruning ratio is 47.67%. We hypothesize that the risk of model overfitting could be partially mitigated by pruning some unimportant channels. Though similar results are also reported for the other counterparts, these schemes are generally more sensitive to the increment of pruning ratio than CPSCA. As another extreme, when the pruning ratio is relatively very high (e.g., 95%), CPSCA still maintains competitive accuracy. The accuracy degradation in SCA is up to 1.65% for VGG16 and up to 3.17% for ResNet56 respectively on both datasets, much less than the corresponding results of the other methods. In particular, our CPSCA pruning algorithm can significantly accelerate the inference computation on the real edge device. All these observations clearly demonstrate that it is beneficial to prune channels with the guidance of our SCA module.

5. Conclusions and Future Work

In this work, we have proposed a new channel pruning approach called Channel Pruning guided by Spatial and Channel Attention (CPSCA) to compress DNN models in edge computing. The core idea of CPSCA is pruning channels with least importance based on the attention statistics provided by a new attention module SCA. By combining spatial and channel attention as a whole, the SCA module can not only enhance the representation power of DNN model, but also reveal the channels' existence to inference performance. Comprehensive experiments on two benchmark datasets verify the effectiveness of the SCA module and the CPSCA approach, as compared to other state-of-the-art solutions.

For future work, we will investigate the performance of our CPSCA with other popular datasets (e.g., ImageNet) and more DNN models for edge intelligence. We also plan to combine this approach with other model compression strategies (e.g., quantization) and other edge intelligence technologies (e.g, edge-cloud collaboration) to further reduce model size and inference cost.

References

[1] R. Plamondon, S. N. Srihari, Online and off-line handwriting recognition: a comprehensive survey, IEEE Transactions on pattern analysis and machine intelligence 22 (2000) 63–84.

[2] J. Wan, D. Wang, S. C. H. Hoi, P. Wu, J. Zhu, Y. Zhang, J. Li, Deep learning for content-based image retrieval: A comprehensive study, in: Proceedings of the 22nd ACM international conference on Multimedia, 2014, pp. 157–166.

[3] R. Girshick, Fast r-cnn, in: Proceedings of the IEEE international conference on computer vision, 2015, pp. 1440–1448.

[4] N. Wang, D.-Y. Yeung, Learning a deep compact image representation for visual tracking, in: Advances in neural information processing systems, 2013, pp. 809–817.

[5] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770–778.

[6] Q. Zhang, C. Zhou, N. Xiong, Y. Qin, X. Li, S. Huang, Multimodel-based incident prediction and risk assessment in dynamic cybersecurity protection for industrial control systems, IEEE Transactions on Systems, Man, and Cybernetics: Systems 46 (2015) 1429–1444.

[7] K. Huang, Q. Zhang, C. Zhou, N. Xiong, Y. Qin, An efficient intrusion detection approach for visual sensor networks based on traffic pattern learning, IEEE Transactions on Systems, Man, and Cybernetics: Systems 47 (2017) 2704–2713.

[8] H. Li, A. Kadav, I. Durdanovic, H. Samet, H. P. Graf, Pruning filters for efficient convnets, in: 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings, OpenReview.net, 2017.

[9] A. M. Grachev, D. I. Ignatov, A. V. Savchenko, Compression of recurrent neural networks for efficient language modeling, Applied Soft Computing 79 (2019) 354 – 362.

[10] S. Han, H. Mao, W. J. Dally, Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding, arXiv preprint arXiv:1510.00149 (2015).

[11] W. Zhang, G. Biswas, Q. Zhao, H. Zhao, W. Feng, Knowledge distilling based model compression and feature learning in fault diagnosis, Applied Soft Computing 88 (2020) 105958.

[12] Y. He, X. Zhang, J. Sun, Channel pruning for accelerating very deep neural networks, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 1389–1397.

[13] J.-H. Luo, J. Wu, W. Lin, Thinet: A filter level pruning method for deep neural network compression, in: Proceedings of the IEEE international conference on computer vision, 2017, pp. 5058–5066.

[14] R. Yu, A. Li, C.-F. Chen, J.-H. Lai, V. I. Morariu, X. Han, M. Gao, C.-Y. Lin, L. S. Davis, Nisp: Pruning networks using neuron importance score propagation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 9194–9203.

[15] J. Huang, W. Sun, L. Huang, Deep neural networks compression learning based on multiobjective evolutionary algorithms, Neurocomputing 378 (2020) 260–269.

[16] Z. Wang, F. Li, G. Shi, X. Xie, F. Wang, Network pruning using sparse learning and genetic algorithm, Neurocomputing (2020).

[17] S. Woo, J. Park, J.-Y. Lee, I. So Kweon, Cbam: Convolutional block attention module, in: Proceedings of the European conference on computer vision (ECCV), 2018, pp. 3–19.

[18] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, Y. Bengio, Show, attend and tell: Neural image caption generation with visual attention, in: International conference on machine learning, 2015, pp. 2048–2057.

[19] D. Bahdanau, K. Cho, Y. Bengio, Neural machine translation by jointly learning to align and translate, arXiv preprint arXiv:1409.0473 (2014).

[20] F. Wang, M. Jiang, C. Qian, S. Yang, C. Li, H. Zhang, X. Wang, X. Tang, Residual attention network for image classification, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 3156–3164.

[21] J. Hu, L. Shen, G. Sun, Squeeze-and-excitation networks, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 7132–7141.

[22] J. Park, S. Woo, J.-Y. Lee, I. S. Kweon, Bam: Bottleneck attention module, arXiv preprint arXiv:1807.06514 (2018).

[23] X. Li, X. Hu, J. Yang, Spatial group-wise enhance: Improving semantic feature learning in convolutional networks, arXiv preprint arXiv:1905.09646 (2019).

[24] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, C. Zhang, Learning efficient convolutional networks through network slimming, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 2736–2744.

[25] F. Song, Y. Wang, Y. Guo, C. Zhu, J. Liu, M. Jin, A channel-level pruning strategy for convolutional layers in cnns, in: 2018 International Conference on Network Infrastructure and Digital Content (IC-NIDC), IEEE, 2018, pp. 135–139.

[26] S. Han, J. Pool, J. Tran, W. Dally, Learning both weights and connections for efficient neural network, in: Advances in neural information processing systems, 2015, pp. 1135–1143.

[27] J.-H. Luo, J. Wu, An entropy-based pruning method for cnn compression, arXiv preprint arXiv:1706.05791 (2017).

[28] T. Zhang, S. Ye, K. Zhang, J. Tang, W. Wen, M. Fardad, Y. Wang, A systematic dnn weight pruning framework using alternating direction method of multipliers, in: Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 184–199.

[29] W. Wen, C. Wu, Y. Wang, Y. Chen, H. Li, Learning structured sparsity in deep neural networks, in: Advances in neural information processing systems, 2016, pp. 2074–2082.

[30] C. Min, A. Wang, Y. Chen, W. Xu, X. Chen, 2pfpce: Two-phase filter pruning based on conditional entropy, arXiv preprint arXiv:1809.02220 (2018).

[31] X. Li, W. Wang, X. Hu, J. Yang, Selective kernel networks, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2019, pp. 510–519.

[32] H. Larochelle, G. E. Hinton, Learning to combine foveal glimpses with a third-order boltzmann machine, in: Advances in neural information processing systems, 2010, pp. 1243–1251.

[33] V. Mnih, N. Heess, A. Graves, et al., Recurrent models of visual attention, in: Advances in neural information processing systems, 2014, pp. 2204–2212.

[34] Q. You, H. Jin, Z. Wang, C. Fang, J. Luo, Image captioning with semantic attention, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 4651–4659.

[35] H. Hu, J. Gu, Z. Zhang, J. Dai, Y. Wei, Relation networks for object detection, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 3588–3597.

[36] H. Zhang, I. Goodfellow, D. Metaxas, A. Odena, Self-attention generative adversarial networks, in: International Conference on Machine Learning, 2019, pp. 7354–7363.

[37] K. Yamamoto, K. Maeno, Pcas: Pruning channels with attention statistics for deep network compression, arXiv preprint arXiv:1806.05382 (2018).

[38] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, Z. Wojna, Rethinking the inception architecture for computer vision, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 2818–2826.

[39] Z. Gao, J. Xie, Q. Wang, P. Li, Global second-order pooling convolutional networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 3024–3033.

[40] S. Sabour, N. Frosst, G. E. Hinton, Dynamic routing between capsules, CoRR abs/1710.09829 (2017). URL: http://arxiv.org/abs/1710.09829. arXiv:1710.09829.

[41] J. Fu, H. Zheng, T. Mei, Look closer to see better: Recurrent attention convolutional neural network for fine-grained image recognition, 2017, pp. 4476–4484. doi:10.1109/CVPR.2017.476.

[42] S. Ioffe, C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, arXiv preprint arXiv:1502.03167 (2015).

[43] S. Qiao, H. Wang, C. Liu, W. Shen, A. Yuille, Weight standardization, arXiv preprint arXiv:1903.10520 (2019).

[44] Y. Wu, K. He, Group normalization, in: Proceedings of the European conference on computer vision (ECCV), 2018, pp. 3–19.

[45] Z. Yang, L. Zhu, Y. Wu, Y. Yang, Gated channel transformation for visual recognition, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 11794–11803.

[46] Q. Wang, B. Wu, P. Zhu, P. Li, W. Zuo, Q. Hu, Eca-net: Efficient channel attention for deep convolutional neural networks, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 11534–11542.

[47] J. L. Ba, J. R. Kiros, G. E. Hinton, Layer normalization, arXiv preprint arXiv:1607.06450 (2016).

[48] X. Gao, Y. Zhao, Ł. Dudziak, R. Mullins, C.-z. Xu, Dynamic channel pruning: Feature boosting and suppression, arXiv preprint arXiv:1810.05331 (2018).

[49] R. Hadidi, J. Cao, Y. Xie, B. Asgari, T. Krishna, H. Kim, Characterizing the deployment of deep neural networks on commercial edge devices, in: 2019 IEEE International Symposium on Workload Characterization (IISWC), 2019, pp. 35–48. doi:10.1109/IISWC47752.2019.9041955.

Mengran Liu received her B.S. degree from the Department of Internet of Things Engineering from TianGong University in 2018. Currently, she is a graduate student in the School of Computer and Information Technology at Beijing Jiaotong University. Her main current research interests include edge computing and Internet of Things.

Weiwei Fang received the B.S. degree from Hefei University of Technology, Hefei, China, and the Ph.D. degree from Beihang University, Beijing, China, in 2003 and 2010, respectively. He is currently an Associate Professor with the School of Computer and Information Technology, Beijing Jiaotong University, Beijing, China. His research interests include edge computing, cloud computing, and Internet of Things. He has published over 60 papers in journals, international conferences/workshops.

Xiaodong Ma received his B.S. degree from the School of Software Engineering from Zhengzhou University in 2020. Currently, he is a graduate student in the School of Computer and Information Technology at Beijing Jiaotong University. His current research interests include edge computing and distributed machine learning.

Wenyuan Xu received his B.S. degree from the School of Software Engineering from Zhengzhou University in 2019. Currently, he is a graduate student in the School of Computer and Information Technology at Beijing Jiaotong University. His current research interests include edge computing and DNN model compression.

Naixue Xiong received the Ph.D. degree in sensor system engineering from Wuhan University and in dependable sensor networks from the Japan Advanced Institute of Science and Technology. He was with Georgia State University, Wentworth Technology Institution, and Colorado Technical University (as a Full Professor for five years) about ten years. He is currently a Professor with the College of Intelligence and Computing, Tianjin University, China. He has published over 300 international journal articles and over 100 international conference papers. Some of his works were published in the IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS, IEEE or ACM transactions, ACM Sigcomm workshop, IEEE INFOCOM, ICDCS, and IPDPS. His research interests include cloud computing, security and dependability, parallel and distributed computing, networks, and optimization theory. Dr. Xiong has been a Senior member of the IEEE Computer Society, since 2012. He has received the Best Paper Award in the 10th IEEE International Conference on High Performance Computing and Communications (HPCC-08) and the Best student Paper Award in the 28th North American Fuzzy Information Processing Society Annual Conference (NAFIPS2009).

Yi Ding received the Ph.D. degree from Beihang University, Beijing, China, in 2014. Currently, he is an Assistant Professor in the School of Information at Beijing Wuzi University. His current research interests include edge computing and blockchain systems.