Adding Before Pruning: Sparse Filter Fusion for Deep Convolutional Neural Networks via Auxiliary Attention

Guanzhong Tian, Yiran Sun, Yuang Liu, Xianfang Zeng, Mengmeng Wang, Yong Liu, Jiangning Zhang, and Jun Chen

***Abstract*— Filter pruning is a significant feature selection technique to shrink the existing feature fusion schemes (especially on convolution calculation and model size), which helps to develop more efficient feature fusion models while maintaining state-of-the-art performance. In addition, it reduces the storage and computation requirements of deep neural networks (DNNs) and accelerates the inference process dramatically. Existing methods mainly rely on manual constraints such as normalization to select the filters. A typical pipeline comprises two stages: first pruning the original neural network and then fine-tuning the pruned model. However, choosing a manual criterion can be somehow tricky and stochastic. Moreover, directly regularizing and modifying filters in the pipeline suffer from being sensitive to the choice of hyperparameters, thus making the pruning procedure less robust. To address these challenges, we propose to handle the filter pruning issue through one stage: using an attention-based architecture that adaptively fuses the filter selection with filter learning in a unified network. Specifically, we present a pruning method named adding before pruning (ABP) to make the model focus on the filters of higher significance by training instead of man-made criteria such as norm, rank, etc. First, we add an auxiliary attention layer into the original model and set the significance scores in this layer to be binary. Furthermore, to propagate the gradients in the auxiliary attention layer, we design a specific gradient estimator and prove its effectiveness for convergence in the graph flow through mathematical derivation. In the end, to relieve the dependence on the complicated prior knowledge for designing the thresholding criterion, we simultaneously prune and train the filters to automatically eliminate network redundancy with recoverability. Extensive experimental results on the two typical image classification benchmarks, CIFAR-10 and ILSVRC-2012, illustrate that the proposed approach performs favorably against previous state-of-the-art filter pruning algorithms.**

***Index Terms*— Deep neural networks (DNNs), effective feature fusion, feature selection, filter pruning.**

Manuscript received September 16, 2020; revised March 22, 2021 and June 22, 2021; accepted August 11, 2021. This work was supported in part by the National Key Research and Development Program of China under Grant 2018AAA0101503 and in part by the Science and Technology Project of State Grid Corporation of China (SGCC) (Fundamental theory of human-in-the-loop hybrid-augmented intelligence for power grid dispatch and control). *(Guanzhong Tian and Yiran Sun contributed equally to this work.) (Corresponding author: Yong Liu.)*

Guanzhong Tian is with the Institute of Cyber-Systems and Control, Zhejiang University, Hangzhou 310027, China, and also with Ningbo Research Institute, Zhejiang University, Ningbo 315000, China.

Yiran Sun, Yuang Liu, Xianfang Zeng, Mengmeng Wang, Yong Liu, Jiangning Zhang, and Jun Chen are with the Institute of Cyber-Systems and Control, Zhejiang University, Hangzhou 310027, China (e-mail: yongliu@iipc.zju.edu.cn).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TNNLS.2021.3106917.

Digital Object Identifier 10.1109/TNNLS.2021.3106917

I. Introduction

THE significant success of deep neural networks (DNNs) in diverse applications [21], [28], [31], [35] (image classification, semantic segmentation, object detection, etc.) is realized in part by advances of computing and storage hardware. However, existing representative fusion schemes including dense connection, residual learning, convolution operation tend to be memory-exhaustive and computationally costly, which blocks practical deployments to local devices: self-driving cars, smartphones, etc. Recent efforts, including developing more effective and efficient fusion schemes for DNNs, have attracted growing research attention [5], [11], [15], [17], [25], [33], [34], [37], [42] to reduce these overheads. And most of these pruning methods mainly focus on how to seek out the redundant features or parameters from a pre-trained model.

Existing methods for pruning fusion models mainly rely on using prior knowledge to improve the performance and reduce the inference time, in which different criteria or constraints demand delicately design for various architectures and datasets. These criteria are basically based on weight magnitude or original performance (such as the accuracy) of the primary convolutional neural network (CNN) model. According to the pruning criterion based on magnitude, weights below a certain threshold are pruned. However, the deduction that filters with smaller norms contribute less to performances is still a prior knowledge and cannot be rigorously verified. Moreover, for the layer-wise pruning [19], [25], [27], the pruned filters are unrecoverable once being pruned based on most pruning criteria, which means that temporarily inaccurate pruning results from non-optimal hyperparameters or batch noises cannot be corrected. Besides, these methods are of inferior generalization abilities due to the sensitivity to hyperparameters.

Liu *et al.* [26] indicate that network pruning is very similar to neural architecture search (NAS) in a way that they both need to find an efficient network architecture. However, we argue that NAS algorithms cannot replace pruning tasks entirely. For instance, Liu *et al.* [26] adopt additional indicators which are optimized by gradient descent to obtain a suitable network structure. Nevertheless, in the evaluation process, discrepancy still exists between the discretized sub-graph and the successive over-parameterized graph. In contrast,

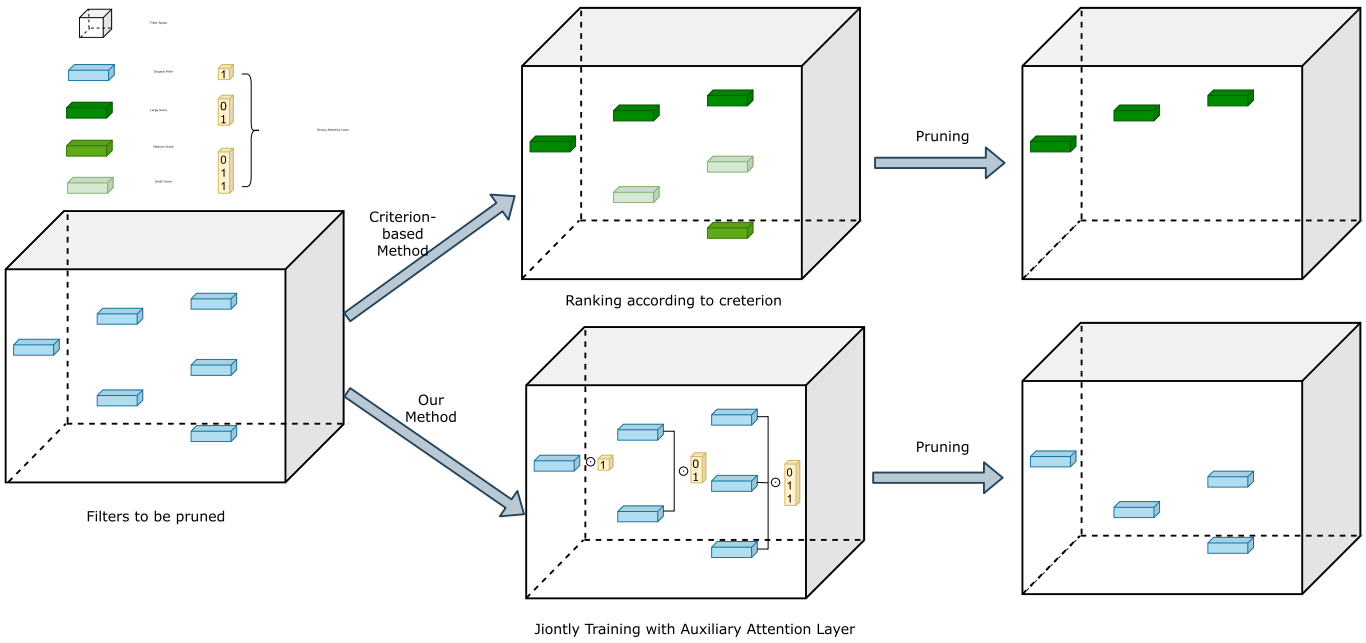


Fig. 1. Sketch of the pruning procedure for the criterion-based approach and our proposed approach. The black boxes indicate filters of the original model. The green color is used to measure the importance score of the filters based on specific criterion. A darker color means a larger score. Based on the assumption that filters with small score are less important, only the larger score filters will be retained. On the contrary, our ABP removes filters with corresponding "0" values in the binary auxiliary attention layer.

to reduce this discrepancy, we propose an iterative recoverable sub-graph.

To overcome the problems mentioned above, we introduce our one-stage method: adding before pruning (ABP) to adaptively fuse the filter selection with filter learning in a unified network. Introducing an auxiliary attention layer to filter pruning can be hyperparameter insensitive: We can regularize auxiliary parameters instead of original weights so that the gradient fluctuations (caused by dead neurons, noises, etc.) can be aggregated in the auxiliary attention layer. As such, temporarily inaccurate pruning results from non-optimal hyperparameters or instability will be corrected.

As shown in Fig. 1, a dot-product operation is applied between the standard convolutional filter and the attention layer. After training the model with an auxiliary attention layer, the original filters associated with 0-valued scalars are directly abandoned, as well as the attention layer, for obtaining a new compact network. One key problem for the training of attention-based filters lies in that we cannot directly adopt a gradient descent algorithm to train the attention layer due to the discontinuity of binary weights. Binarized neural networks (BNNs) [15] adopt straight-through-estimator (STE) to train binary weights and step functions to generate binary values. Inspired by their thought, we design another modified indicator function for attention function a_j^i parameterized with m_j^i. To testify our proposed algorithm, comprehensive experiments on various structures and benchmarks are conducted. Compared with the other studied pruning methods, our ABP achieves state-of-the-art pruning performance in the experiments. We make the following contributions in this work.

1) We propose an attention-based one stage pruning method that adaptively fuses the feature selection with original parameter training in a unified network. To our knowledge, we are the first to prune from scratch: combine training with pruning and obtain the final pruned model directly without fine-tuning.
2) We develop a novel gradient estimator for the update of binary auxiliary parameters via separating the original weights from the auxiliary parameters update processes and prove its rationality through mathematical derivation.
3) We demonstrate that, with the proposed model, our ABP can achieve considerable performance improvements, and it is robust under different initialization methods.

II. Related Work

We can mainly divide pruning methods for DNNs into two types: the structured pruning and the unstructured one. Both pruning methods can reduce the storage space, while structured pruning performs much better on computational cost reduction.

A. Unstructured Pruning

This is a weight-level pruning approach, which prunes weights in all layers. Deep compression [5] propose to prune weights with small values as well as non-significant connections in pre-trained models. Guo *et al.* [4] develop an interactive algorithm by introducing recoverability into the global pruning. Zhang *et al.* [39] fuse cardinality constraints with the weights and re-formulate the pruning problem into a non-convex optimization issue. However, a non-negligible

weakness of unstructured pruning methods is that these methods generate an out-of-order compressed model. Thus calculation cost and the storage usage of the pruned neural networks will not shrink that much.

B. Structured Pruning

This is a filter-level pruning approach that is designed for pruning channels. Compared to weight-level pruning schemes, the structured pruning methods remove the filters and the corresponding feature maps as well. It can favorably accelerate the inference by decreasing the memory footprint consumption as well as reducing parallel computing on hardware. The most common route for filter pruning is to employ a human-made evaluation criterion to calculate each filter's importance score and remove the filters with smaller scores. Li *et al.* [19] perform the selection of redundant channels using L1-norms of weights in each channel. He *et al.* [7] propose to select filters with an L2-norm based criterion, and those selected filters will be pruned in a recoverable manner. Both methods are formulated on the assumption that channels with smaller norms have fewer contributions to the model. Similarly, Network trimming [12] takes advantage of average zero output ratio to remove neurons, and NISP [38] indicates that, by back-propagating importance scores from the last layer, they can obtain other layers' score. For further increasing the compression rate, researchers usually adopt regularization constraints to remove more filters. He *et al.* [9] propose to select channels via a least-square reconstruction algorithm, and they present a filter pruning strategy based on LASSO. Liu *et al.* [25] introduce a penalty term into the loss function: the L1 regularization for scaling factors from the batch normalization (BN) layer, and channels with smaller factors will be removed. However, for all these structured pruning algorithms, a manually pre-designed selection criterion is required to perform the pruning procedure, and the network is pruned through the original weights.

C. Attention

Motivated by how a human focus on the keywords in context or various regions of a photo, numerous methods have been developed [13], [32] to improve the performance of DNNs by incorporating attention mechanism. Wang *et al.* [32] introduces an attention module based on an encoder decoder. They use the module to formulate the Residual Attention Network. Hu *et al.* [13] propose to compute channel-wise attention through global average-pooled features. They exploit the inter-channel relationship by a compact module called the Squeeze-and-Excitation module. Yamamoto and Maeno [36] propose to evaluate the importance of filters through attention statistics, and they develop a filter selection algorithm based on it. Inspired by their channel-wise attention mechanism, we develop our auxiliary attention layer, which indicates which filter is more important and should be retained.

III. Approach

A. Problem Formulation

Given training dataset consists of N samples $\{x_i, y_i\}_{i=1}^{N}$, where $x_i \in R_{m\times n}$ denotes input features and $y_i \in R_d$ is ground-truth labels of x_i. Suppose $f_w{:}R_{m\times n} \rightarrow R_d$ is a differentiable and continuous neural network model with parameters W, which maps input x_i to target y_i. We can formulate the channel selection problem as

$$\arg\min_{w} \frac{1}{N}\left(\sum_{i=1}^{N}\mathcal{L}(f(x_i, W), y_i)\right) + \rho O(W) \tag{1}$$

where $O(W)$ represents the number of filters that contain non-zero weights and ρ is used to balance the two parts in the optimization function. The object lies in seeking a minimum subset $W' \in W$ and formulates a sparse model without reducing the accuracy of the original model. Here are two challenges: The problem cannot be solved by gradient descent since the last term is non-differentiable. In addition, hyperparameter can be more sensitive and batch training can be more unstable due to the direct regularization on W'.

We relax these challenges by bringing in the auxiliary attention layer as an indicator of whether corresponding filters deserve more attention or not. Further, in order to prune filters that deserve low attention, we propose to constrain weights in the auxiliary attention layer to "0" and "1" through the indicator function, which can be defined as

$$a_j^i = \begin{cases} 1, & \text{if corresponding filter } \mathcal{F}_j^i \text{ needs attention} \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

These two values are very advantageous for indicating whether the corresponding filter should be paid more attention to. That is to say, "0" indicates that the corresponding filter will be pruned while "1" indicates that it will be preserved. We seek to parameterize a universal indicator function by introducing auxiliary attention layer with parameters M rather than deploying indicator functions for each $\mathcal{F}$. Then we can re-formulate the network filter selection problem as an optimization problem

$$\arg\min_{w,m} \frac{1}{N}\left(\sum_{i=1}^{N}\mathcal{L}(f(x_i, W \odot M), y_i)\right) \tag{3}$$

where $\hat{W} = W \odot M$ is the element-wise product, which represents the pruned weight matrix. The channel pruning process will be less sensitive by regularizing on auxiliary attention layers rather than targeting filters. That is because the gradient update of w_j^i will not be influenced directly by the change of m_j^i.

B. Gradient Estimator Design

As described above, the weights in the auxiliary attention layer are assigned to quantized values and the indicator function is a piecewise constant function. At this point, training activation binary layers tend to be unreachable for the standard chain rule or back-propagation. That is because the gradients of piecewise constant functions vanish all the time. To address this issue, some researchers choose to employ straight-through estimator (STE) [10] to estimate the vanished gradients. Original STE simply adopts the sign function (-1 when the input is negative, 1 otherwise) as an indicator function and identity function to back-propagate in the backward pass. In this way,

the "gradient" through the changed chain rule tends to be non-trivial. Although this method can handle functions with differentials full of zero, the weakness is that it can only deal with one binary decision.

In our case, the stochastic binary neurons in the auxiliary attention layers are designed to face more than one binary decision, and we attempt to fuse the pruning thresholding parameter t into the indicator function. Therefore, we no longer use the sign function in the forward pass of our models. Also, we should assume that another function related with t rather than the original identity function is adopted to back-propagate the gradients in the backward pass.

To address our problem without lossing generality, we emphasize channels by weights in the auxiliary attention layer. Therefore, instead of sign function, we propose our indicator function in auxiliary attention layer

$$a_j^i = \sigma\left(m_j^i\right) = \begin{cases} 0, & \text{if } \left|m_j^i\right| \le t \\ 1, & \text{if } \left|m_j^i\right| > t \end{cases} \tag{4}$$

where $t > 0$ is the threshold factor as well as a hyperparameter used to judge whether we keep the corresponding filters. Hyperparameter t can be different across different convolutional layers. Clearly, the value of t will affect the pruning rate of each convolutional layer. In fact, we adopt t to replace the conventional "pruning rate" used in other works. Considering the output of the indicator function is a_j^i and $g_a = (\partial l/\partial a_j^i)$ can be acquired by back-propagation algorithm, we propose our gradient estimator

$$g_m = g_a 1_{|m| \le t} \tag{5}$$

where g_m is the "gradient" of loss w.r.t. the variables in auxiliary attention layer (i.e., m_j^i) through our modified chain rule, which is referred to as estimated gradient in this article.

Although promising results in various tasks have proved the effectiveness of BNNs [15], the rationality for STE is still controversial. That is because the backward and forward passes do not match, and the estimated gradients are definitely not the gradients from the loss function. Therefore, the following questions come up: Since the selection of an estimated gradient is non-unique, how can we say that ours is good? And why does it work even when the estimated gradient uses a "faked" gradient? In the following subsection, we try to take a step forward, attempting to solve these questions from the optimization perspective.

C. Rationality of Our Gradient Estimator

In this subsection, we prove that the estimated partial gradient using our method and the population loss's true partial gradient generally has a positive correlation. This means that the gradient descent generated by our estimator can behave such as the real descent on loss function and yield good convergence.

We consider a simplified neural network case similar to [2] that outputs the prediction

$$f(\boldsymbol{Z}, \boldsymbol{p}, \boldsymbol{q}) = \sum_{i=1}^{m} \boldsymbol{p}_i \sigma\left(\boldsymbol{Z}_i^{\mathrm{T}} \boldsymbol{q}\right) = \boldsymbol{p}^{\mathrm{T}} \sigma(\boldsymbol{Z}\boldsymbol{q}). \tag{6}$$

The first layer in the case is a convolutional layer with trainable weights $\boldsymbol{q} \in R^n$, followed by a fully connected layer with trainable weights $\boldsymbol{p} \in R^m$ as the classifier. $\boldsymbol{Z} \in R^{m*n}$ is the input, and σ is the activation function. $f^*(\boldsymbol{Z}, \boldsymbol{p}, \boldsymbol{q}) = (\boldsymbol{p}^*)^{\mathrm{T}} \sigma(\boldsymbol{Z}\boldsymbol{q}^*)$ is used to generate the label and L2 loss is adopted as the loss function

$$\begin{aligned} l(\boldsymbol{Z}, \boldsymbol{p}, \boldsymbol{q}) &= \frac{1}{2}\left(f(\boldsymbol{Z}, \boldsymbol{p}, \boldsymbol{q}) - f^*(\boldsymbol{Z}, \boldsymbol{p}, \boldsymbol{q})\right)^2 \\ &= \frac{1}{2}\left(\boldsymbol{p}^{\mathrm{T}} \sigma(\boldsymbol{Z}\boldsymbol{q}) - \left(\boldsymbol{p}^*\right)^{\mathrm{T}} \sigma\left(\boldsymbol{Z}\boldsymbol{q}^*\right)\right)^2. \end{aligned} \tag{7}$$

Then we can obtain the partial gradient of L2 loss

$$\frac{\partial l}{\partial \boldsymbol{q}} = \boldsymbol{Z}^{\mathrm{T}}\left(\sigma'(\boldsymbol{Z}\boldsymbol{q}) \odot \boldsymbol{p}\right)\left(\boldsymbol{p}^{\mathrm{T}} \sigma(\boldsymbol{Z}\boldsymbol{q}) - \left(\boldsymbol{p}^*\right)^{\mathrm{T}} \sigma\left(\boldsymbol{Z}\boldsymbol{q}^*\right)\right) \tag{8}$$

$$\frac{\partial l}{\partial \boldsymbol{p}} = \sigma(\boldsymbol{Z}\boldsymbol{q})\left(\boldsymbol{p}^{\mathrm{T}} \sigma(\boldsymbol{Z}\boldsymbol{q}) - \left(\boldsymbol{p}^*\right)^{\mathrm{T}} \sigma\left(\boldsymbol{Z}\boldsymbol{q}^*\right)\right). \tag{9}$$

According to (4), σ' is zero almost everywhere. In order to pass the gradient, we employ a related non-trivial function μ' to replace these zero gradients, which is the estimator of the gradient. Using μ' to replace σ', we can get

$$\frac{\partial l}{\partial \boldsymbol{q}} = \boldsymbol{Z}^{\mathrm{T}}\left(\mu'(\boldsymbol{Z}\boldsymbol{q}) \odot \boldsymbol{p}\right)\left(\boldsymbol{p}^{\mathrm{T}} \sigma(\boldsymbol{Z}\boldsymbol{q}) - \left(\boldsymbol{p}^*\right)^{\mathrm{T}} \sigma\left(\boldsymbol{Z}\boldsymbol{q}^*\right)\right). \tag{10}$$

Assume $\boldsymbol{Z}$ is sampled from a Gaussian distribution. Following [1], [41], we can cast the L2 loss as the following population loss:

$$L_Z(\boldsymbol{p}, \boldsymbol{q}) = E_{\boldsymbol{Z}}[l(\boldsymbol{Z}, \boldsymbol{p}, \boldsymbol{q})]. \tag{11}$$

Since gradients of the object loss function is unavailable, we can access the expected sample gradient: $E_{\boldsymbol{Z}}[(\partial/\partial \boldsymbol{p}) l(\boldsymbol{Z}, \boldsymbol{p}, \boldsymbol{q})]$ and $E_{\boldsymbol{Z}}[(\partial/\partial \boldsymbol{q}) l(\boldsymbol{Z}, \boldsymbol{p}, \boldsymbol{q})]$. We will calculate the population loss and the expected sample gradients in the following part. The calculation process basically uses rotational invariant property and polar decomposition of Gaussian random variables. Besides, it needs to utilize some auxiliary lemmas. See section Appendix for details.

First, We need to obtain the expression of the population loss $L_Z(\boldsymbol{p}, \boldsymbol{q})$. From (9), it is easy to get that

$$\begin{aligned} &L_Z(\boldsymbol{p}, \boldsymbol{q}) \\ &= E_{\boldsymbol{Z}}[l(\boldsymbol{Z}, \boldsymbol{p}, \boldsymbol{q})] = E_{\boldsymbol{Z}}\left[\frac{1}{2}\left(\boldsymbol{p}^{\mathrm{T}} \sigma(\boldsymbol{Z}\boldsymbol{q}) - \left(\boldsymbol{p}^*\right)^{\mathrm{T}} \sigma\left(\boldsymbol{Z}\boldsymbol{q}^*\right)\right)^2\right] \\ &= \frac{1}{2}\Big(\boldsymbol{p}^{\mathrm{T}} E_{\boldsymbol{Z}}[\sigma(\boldsymbol{Z}\boldsymbol{q})\sigma(\boldsymbol{Z}\boldsymbol{q})^{\mathrm{T}}]\boldsymbol{p} - 2\boldsymbol{p}^{\mathrm{T}} E_{\boldsymbol{Z}}[\sigma(\boldsymbol{Z}\boldsymbol{q})\sigma\left(\boldsymbol{Z}\boldsymbol{q}^*\right)^{\mathrm{T}}]\boldsymbol{p}^* \\ &\qquad + \left(\boldsymbol{p}^*\right)^{\mathrm{T}} E_{\boldsymbol{Z}}[\sigma\left(\boldsymbol{Z}\boldsymbol{q}^*\right)\sigma\left(\boldsymbol{Z}\boldsymbol{q}^*\right)^{\mathrm{T}}]\boldsymbol{p}^*\Big). \end{aligned}$$

Assume that the ith row of $\boldsymbol{Z}$ is $\boldsymbol{Z}_i^{\mathrm{T}}$, we can have the following equations using Lemma 1:

$$\begin{aligned} E_{\boldsymbol{Z}}\left[\sigma(\boldsymbol{Z}\boldsymbol{q})\sigma(\boldsymbol{Z}\boldsymbol{q})^{\mathrm{T}}\right]_{ij} &= E_{\boldsymbol{Z}}\left[\sigma\left(\boldsymbol{Z}_i^{\mathrm{T}}\boldsymbol{q}\right)\sigma\left(\boldsymbol{Z}_j^{\mathrm{T}}\boldsymbol{q}\right)\right] \\ &= E_{\boldsymbol{Z}}\left[1_{\{\boldsymbol{Z}_i^{\mathrm{T}}\boldsymbol{q}>0\}}\right] E_{\boldsymbol{Z}}\left[1_{\{\boldsymbol{Z}_j^{\mathrm{T}}\boldsymbol{q}>0\}}\right] \\ &= \frac{1}{4}(i \ne j) \\ E_{\boldsymbol{Z}}[\sigma(\boldsymbol{Z}\boldsymbol{q})\sigma(\boldsymbol{Z}\boldsymbol{q})^{\mathrm{T}}]_{ii} &= E_{\boldsymbol{Z}}\left[\sigma\left(\boldsymbol{Z}_i^{\mathrm{T}}\boldsymbol{q}\right)\sigma\left(\boldsymbol{Z}_i^{\mathrm{T}}\boldsymbol{q}\right)\right] \\ &= E_{\boldsymbol{Z}}\left[1_{\{\boldsymbol{Z}_i^{\mathrm{T}}\boldsymbol{q}>0\}}\right] \\ &= \frac{1}{2}. \end{aligned}$$

Therefore, we can conclude the above two equations into

$$E_Z[\sigma(\boldsymbol{Zq})\sigma(\boldsymbol{Zq})^{\mathrm{T}}] = \frac{1}{4}(\boldsymbol{I} + \boldsymbol{1}\boldsymbol{1}^{\mathrm{T}}).$$

Similarly, we can check that

$$E_Z[\sigma(\boldsymbol{Zq}^*)\sigma(\boldsymbol{Zq}^*)^{\mathrm{T}}] = \frac{1}{4}(\boldsymbol{I} + \boldsymbol{1}\boldsymbol{1}^{\mathrm{T}}).$$

Moreover, invoking Lemma 1

$$\begin{aligned} E_Z[\sigma(\boldsymbol{Zq})\sigma(\boldsymbol{Zq}^*)^{\mathrm{T}}]_{ij} &= E_Z[1_{\{\boldsymbol{Z}_i^{\mathrm{T}}\boldsymbol{q}>0\}}]E_Z[1_{\{\boldsymbol{Z}_j^{\mathrm{T}}\boldsymbol{q}^*>0\}}] \\ &= \frac{1}{4} \\ E_Z[\sigma(\boldsymbol{Zq})\sigma(\boldsymbol{Zq}^*)^{\mathrm{T}}]_{ii} &= E_Z[1_{\{\boldsymbol{Z}_i^{\mathrm{T}}\boldsymbol{q}>0\}}]E_Z[1_{\{\boldsymbol{Z}_i^{\mathrm{T}}\boldsymbol{q}^*>0\}}] \\ &= \frac{\pi - \alpha(\boldsymbol{q},\boldsymbol{q}^*)}{2\pi}. \end{aligned}$$

Therefore

$$E_Z[\sigma(\boldsymbol{Zq})\sigma(\boldsymbol{Zq}^*)^{\mathrm{T}}] = \frac{1}{4}\left(\left(1 - \frac{2}{\pi}\alpha(\boldsymbol{q},\boldsymbol{q}^*)\right)\boldsymbol{I} + \boldsymbol{1}\boldsymbol{1}^{\mathrm{T}}\right).$$

Combine the above-mentioned result we can obtain the final claim

$$\begin{aligned} &L_Z(\boldsymbol{p},\boldsymbol{q}) \\ &\quad= \frac{1}{8}\boldsymbol{p}^{\mathrm{T}}(I + \boldsymbol{1}\boldsymbol{1}^{\mathrm{T}})\boldsymbol{p} + \frac{1}{8}(\boldsymbol{p}^*)^{\mathrm{T}}(\boldsymbol{I} + \boldsymbol{1}\boldsymbol{1}^{\mathrm{T}})\boldsymbol{p}^* \\ &\qquad-\frac{1}{4}\boldsymbol{p}^{\mathrm{T}}\left(\left(1 - \frac{2}{\pi}\alpha(\boldsymbol{q},\boldsymbol{q}^*)\right)\boldsymbol{I} + \boldsymbol{1}\boldsymbol{1}^{\mathrm{T}}\right)\boldsymbol{p}^* \end{aligned} \tag{12}$$

where α is the angel between vector $\boldsymbol{q}$ and vector $\boldsymbol{q}^*$, $\boldsymbol{I}$ and $\boldsymbol{1}\boldsymbol{1}^{\mathrm{T}}$ represents identity matrix and square matrix with all 1 element, respectively.

Second, we try to calculate the gradients of population loss w.r.t. p and q. First

$$\begin{aligned} \frac{\partial L_Z}{\partial \boldsymbol{p}} &= \frac{\partial \frac{1}{8}\boldsymbol{p}^{\mathrm{T}}(\boldsymbol{I} + \boldsymbol{1}\boldsymbol{1}^{\mathrm{T}})\boldsymbol{p}}{\partial \boldsymbol{p}} + \frac{\partial \frac{1}{8}(\boldsymbol{p}^*)^{\mathrm{T}}(\boldsymbol{I} + \boldsymbol{1}\boldsymbol{1}^{\mathrm{T}})\boldsymbol{p}^*}{\partial \boldsymbol{p}} \\ &\quad-\frac{\partial \frac{1}{4}\boldsymbol{p}^{\mathrm{T}}\left(\left(1 - \frac{2}{\pi}\alpha(\boldsymbol{q},\boldsymbol{q}^*)\right)\boldsymbol{I} + \boldsymbol{1}\boldsymbol{1}^{\mathrm{T}}\right)\boldsymbol{p}^*}{\partial \boldsymbol{p}} \\ &= \frac{1}{8}\left[(\boldsymbol{I} + \boldsymbol{1}\boldsymbol{1}^{\mathrm{T}}) + (\boldsymbol{I} + \boldsymbol{1}\boldsymbol{1}^{\mathrm{T}})^{\mathrm{T}}\right]\boldsymbol{p} \\ &\quad-\frac{1}{4}\left(\left(1 - \frac{2}{\pi}\alpha(\boldsymbol{q},\boldsymbol{q}^*)\right)\boldsymbol{I} + \boldsymbol{1}\boldsymbol{1}^{\mathrm{T}}\right)\boldsymbol{p}^* \\ &= \frac{1}{4}(\boldsymbol{I} + \boldsymbol{1}\boldsymbol{1}^{\mathrm{T}})\boldsymbol{p} - \frac{1}{4}\left(\left(1 - \frac{2}{\pi}\alpha(\boldsymbol{q},\boldsymbol{q}^*)\right)\boldsymbol{I} + \boldsymbol{1}\boldsymbol{1}^{\mathrm{T}}\right)\boldsymbol{p}^*. \end{aligned} \tag{13}$$

Then, since the gradients of population loss w.r.t. q is only related with $\alpha(\boldsymbol{q},\boldsymbol{q}^*)$, and we can easily obtain the differential of $\alpha(\boldsymbol{q},\boldsymbol{q}^*) = \arccos(\boldsymbol{q}^{\mathrm{T}}\boldsymbol{q}^*/||\boldsymbol{q}||)$ w.r.t. $\boldsymbol{q}$ at $\alpha(\boldsymbol{q},\boldsymbol{q}^*) \in (0,\pi)$. As a result

$$\frac{\partial L_Z}{\partial \boldsymbol{q}} = \frac{\boldsymbol{p}^{\mathrm{T}}\boldsymbol{p}^*}{2\pi}\frac{(\boldsymbol{q}^{\mathrm{T}}\boldsymbol{q}^*)\boldsymbol{q} - ||\boldsymbol{q}||^2\boldsymbol{q}^*}{||\boldsymbol{q}||^3\sqrt{1 - \frac{(\boldsymbol{q}^{\mathrm{T}}\boldsymbol{q}^*)^2}{||\boldsymbol{q}||^2}}}. \tag{14}$$

Third, we calculate the expected partial gradient of l w.r.t. $\boldsymbol{p}$ and $\boldsymbol{q}$

$$E_Z\left[\frac{\partial l}{\partial \boldsymbol{p}}\right] = \frac{\partial l}{\partial \boldsymbol{p}}. \tag{15}$$

Based on (13), $(\partial L_Z/\partial \boldsymbol{p})$ is linear. Then we can get

$$E_Z\left[\frac{\partial l}{\partial \boldsymbol{p}}\right] = \left[\frac{\partial E_Z[l]}{\partial \boldsymbol{p}}\right] = \frac{\partial l}{\partial \boldsymbol{p}}.$$

By (10), Let $\mu(m) = \min\{\max\{x,-t\},t\}(t > 0)$, $\mu' = 1_{\{-t<x<t\}}$, we have

$$\begin{aligned} &E_Z\left[\frac{\partial l}{\partial \boldsymbol{q}}\right] \\ &= E_Z\left[\left(\sum_{i=1}^m \boldsymbol{p}_i\mu'(\boldsymbol{Z}_i^{\mathrm{T}}\boldsymbol{q}) - \sum_{i=1}^m \boldsymbol{p}_i^*\mu'(\boldsymbol{Z}_i^{\mathrm{T}}\boldsymbol{q}^*)\right)\right. \\ &\qquad\left.\times\left(\sum_{i=1}^m \boldsymbol{Z}_i\boldsymbol{p}_i\sigma(\boldsymbol{Z}_i^{\mathrm{T}}\boldsymbol{q})\right)\right] \\ &= E_Z\left[\left(\sum_{i=1}^m \boldsymbol{p}_i 1_{\{|\boldsymbol{Z}_i^{\mathrm{T}}\boldsymbol{q}|<t\}} - \sum_{i=1}^m \boldsymbol{p}_i^* 1_{\{|\boldsymbol{Z}_i^{\mathrm{T}}\boldsymbol{q}^*|<t\}}\right)\right. \\ &\qquad\left.\times\left(\sum_{i=1}^m 1_{\{\boldsymbol{Z}_i^{\mathrm{T}}\boldsymbol{q}>0\}}\boldsymbol{p}_i\boldsymbol{Z}_i\right)\right] \\ &= \sum_{i=1}^m \boldsymbol{p}_i^2 E\left[\boldsymbol{Z}_i 1_{\{|\boldsymbol{Z}_i^{\mathrm{T}}\boldsymbol{q}|<t,\boldsymbol{Z}_i^{\mathrm{T}}\boldsymbol{q}>0\}}\right] \\ &\quad+\sum_{i=1}^m\sum_{\substack{i=1\\j\neq i}}^m \boldsymbol{p}_i\boldsymbol{p}_j E\left[\boldsymbol{Z}_i 1_{\{|\boldsymbol{Z}_i^{\mathrm{T}}\boldsymbol{q}|<t,\boldsymbol{Z}_j^{\mathrm{T}}\boldsymbol{q}>0\}}\right] \\ &\quad-\sum_{i=1}^m \boldsymbol{p}_i\boldsymbol{p}_i^* E\left[\boldsymbol{Z}_i 1_{\{|\boldsymbol{Z}_i^{\mathrm{T}}\boldsymbol{q}^*|<t,\boldsymbol{Z}_i^{\mathrm{T}}\boldsymbol{q}>0\}}\right] \\ &\quad-\sum_{i=1}^m\sum_{\substack{i=1\\j\neq i}}^m \boldsymbol{p}_i\boldsymbol{p}_j^* E\left[\boldsymbol{Z}_i 1_{\{|\boldsymbol{Z}_i^{\mathrm{T}}\boldsymbol{q}^*|<t,\boldsymbol{Z}_j^{\mathrm{T}}\boldsymbol{q}>0\}}\right]. \end{aligned}$$

Using Lemma 1 and 2, we have

$$E\left[\boldsymbol{Z}_i 1_{\{-t<\boldsymbol{Z}_i^{\mathrm{T}}\boldsymbol{q}<t,\boldsymbol{Z}_j^{\mathrm{T}}\boldsymbol{q}>0\}}\right] = \begin{cases} \upsilon(0,\boldsymbol{q})\dfrac{\boldsymbol{q}}{||\boldsymbol{q}||}, & i = j \\ \upsilon(0,\boldsymbol{q})\dfrac{\boldsymbol{q}}{2||\boldsymbol{q}||}, & i \neq j \end{cases}$$

$$E\left[\boldsymbol{Z}_i 1_{\{-t<\boldsymbol{Z}_i^{\mathrm{T}}\boldsymbol{q}<t,\boldsymbol{Z}_j^{\mathrm{T}}\boldsymbol{q}^*>0\}}\right] = \begin{cases} \left[\upsilon(\alpha,\boldsymbol{q}), w(\alpha,\boldsymbol{q}), 0^{\mathrm{T}}\right], & i = j \\ \upsilon(0,\boldsymbol{q})\dfrac{\boldsymbol{q}}{2||\boldsymbol{q}||}, & i \neq j. \end{cases}$$

Therefore

$$\begin{aligned} &E_Z\left[\frac{\partial L_Z}{\partial \boldsymbol{q}}\right] \\ &= \frac{1}{2}\upsilon(0,\boldsymbol{q})\left(||\boldsymbol{p}||^2 + \left(1^{\mathrm{T}}\boldsymbol{p}\right)^2\right)\frac{\boldsymbol{q}}{||\boldsymbol{q}||} - \left(\boldsymbol{p}^{\mathrm{T}}\boldsymbol{p}^*\right) \\ &\quad\times\left[\left(\upsilon(\alpha,\boldsymbol{q}) - \cot(\alpha/2)w(\alpha,\boldsymbol{q})\right)\frac{\boldsymbol{q}}{||\boldsymbol{q}||}\right. \\ &\qquad\left.+\csc(\alpha/2)w(\alpha,\boldsymbol{q})\frac{\frac{\boldsymbol{q}}{||\boldsymbol{q}||} + \boldsymbol{q}^*}{\left\|\frac{\boldsymbol{q}}{||\boldsymbol{q}||} + \boldsymbol{q}^*\right\|}\right] \\ &\quad-\frac{1}{2}\upsilon(0,\boldsymbol{q})\left(\left(1^{\mathrm{T}}\boldsymbol{p}\right)\left(1^{\mathrm{T}}\boldsymbol{p}^*\right) - \boldsymbol{p}^{\mathrm{T}}\boldsymbol{p}^*\right)\frac{\boldsymbol{q}}{||\boldsymbol{q}||} \\ &= \frac{1}{2}\upsilon(0,\boldsymbol{q})\left(||\boldsymbol{p}||^2 + \left(1^{\mathrm{T}}\boldsymbol{p}\right)^2 - \left(1^{\mathrm{T}}\boldsymbol{p}\right)\left(1^{\mathrm{T}}\boldsymbol{p}^*\right) + \boldsymbol{p}^{\mathrm{T}}\boldsymbol{p}^*\right)\frac{\boldsymbol{q}}{||\boldsymbol{q}||} \end{aligned}$$

$$-\left(\boldsymbol{p}^{\mathrm{T}}\boldsymbol{p}^{*}\right)\Bigg[\left(v(\alpha,\boldsymbol{q})-\cot(\alpha/2)w(\alpha,\boldsymbol{q})\right)\frac{\boldsymbol{q}}{||\boldsymbol{q}||}+\csc(\alpha/2)w(\alpha,\boldsymbol{q})\frac{\frac{\boldsymbol{q}}{||\boldsymbol{q}||}+\boldsymbol{q}^{*}}{\left\|\frac{\boldsymbol{q}}{||\boldsymbol{q}||}+\boldsymbol{q}^{*}\right\|}\Bigg]$$

where cot, sec and csc are cosine, tangent and cotangent trigonometric function, respectively.

$$v(\alpha,\boldsymbol{q})=\frac{1}{2\pi}\int_{-\frac{\pi}{2}}^{\frac{\pi}{2}+\alpha}\cos(\phi)\int_{-\frac{\sec(\phi)}{t||\boldsymbol{q}||}}^{\frac{\sec(\phi)}{t||\boldsymbol{q}||}}r^{2}e^{-\frac{r^{2}}{2}}drd\phi$$

$$w(\alpha,\boldsymbol{q})=\frac{1}{2\pi}\int_{-\frac{\pi}{2}}^{\frac{\pi}{2}+\alpha}\sin(\phi)\int_{-\frac{\sec(\phi)}{t||\boldsymbol{q}||}}^{\frac{\sec(\phi)}{t||\boldsymbol{q}||}}r^{2}e^{-\frac{r^{2}}{2}}drd\phi.$$

According to (14), we have

$$\frac{\partial L_{Z}}{\partial\boldsymbol{q}}=\frac{\boldsymbol{p}^{\mathrm{T}}\boldsymbol{p}^{*}}{2\pi}\frac{\left(\boldsymbol{q}^{\mathrm{T}}\boldsymbol{q}^{*}\right)\boldsymbol{q}-||\boldsymbol{q}||^{2}\boldsymbol{q}^{*}}{||\boldsymbol{q}||^{3}\sqrt{1-\frac{\left(\boldsymbol{q}^{\mathrm{T}}\boldsymbol{q}^{*}\right)^{2}}{||\boldsymbol{q}||^{2}}}}=-\frac{\boldsymbol{p}^{\mathrm{T}}\boldsymbol{p}^{*}}{2\pi||\boldsymbol{q}||}\frac{\left(\boldsymbol{I}-\frac{\boldsymbol{q}\boldsymbol{q}^{\mathrm{T}}}{||\boldsymbol{q}||^{2}}\right)\boldsymbol{q}^{*}}{\left\|\left(\boldsymbol{I}-\frac{\boldsymbol{q}\boldsymbol{q}^{\mathrm{T}}}{||\boldsymbol{q}||^{2}}\right)\boldsymbol{q}^{*}\right\|}.$$

In the end, we calculate the inner product of our estimated gradient and the real one w.r.t. $\boldsymbol{q}$: Notice that $(\boldsymbol{I}-(\boldsymbol{q}\boldsymbol{q}^{\mathrm{T}}/||\boldsymbol{q}||^{2})\boldsymbol{q}=0$, then

$$\left\langle E_{Z}\left[\frac{\partial l}{\partial\boldsymbol{q}}\right],\frac{\partial L_{Z}}{\partial\boldsymbol{q}}\right\rangle=\csc(\alpha/2)\frac{w(\alpha,\boldsymbol{q})}{2\pi}\left(\boldsymbol{p}^{\mathrm{T}}\boldsymbol{p}^{*}\right)^{2}\times\left\langle\frac{1}{||\boldsymbol{q}||}\frac{\left(\left(\boldsymbol{I}-\frac{\boldsymbol{q}\boldsymbol{q}^{\mathrm{T}}}{||\boldsymbol{q}||^{2}}\right)\boldsymbol{q}^{*}\right)}{\left\|\left(\left(\boldsymbol{I}-\frac{\boldsymbol{q}\boldsymbol{q}^{\mathrm{T}}}{||\boldsymbol{q}||^{2}}\right)\boldsymbol{q}^{*}\right)\right\|},\frac{\boldsymbol{q}^{*}}{\frac{\boldsymbol{q}}{||\boldsymbol{q}||}+\boldsymbol{q}^{*}}\right\rangle.$$

Among the above equation

$$\begin{aligned}&\left\langle\frac{1}{||\boldsymbol{q}||}\frac{\left(\boldsymbol{I}-\frac{\boldsymbol{q}\boldsymbol{q}^{\mathrm{T}}}{||\boldsymbol{q}||^{2}}\right)\boldsymbol{q}^{*})}{\left\|\left(\boldsymbol{I}-\frac{\boldsymbol{q}\boldsymbol{q}^{\mathrm{T}}}{||\boldsymbol{q}||^{2}}\right)\boldsymbol{q}^{*})\right\|},\frac{\boldsymbol{q}^{*}}{\frac{\boldsymbol{q}}{||\boldsymbol{q}||}+\boldsymbol{q}^{*}}\right\rangle\\&=\frac{||\boldsymbol{q}||^{2}-(\boldsymbol{q}^{\mathrm{T}}\boldsymbol{q}^{*})^{2}}{||\ ||\boldsymbol{q}||^{2}\boldsymbol{q}^{*}-\boldsymbol{q}(\boldsymbol{q}^{\mathrm{T}}\boldsymbol{q}^{*})||\ ||\boldsymbol{q}+||\boldsymbol{q}||\boldsymbol{q}^{*}||}\\&=\frac{||\boldsymbol{q}||^{2}-(\boldsymbol{q}^{\mathrm{T}}\boldsymbol{q}^{*})^{2}}{\sqrt{||\boldsymbol{q}||^{4}\boldsymbol{q}^{*}-||\boldsymbol{q}||^{2}(\boldsymbol{q}^{\mathrm{T}}\boldsymbol{q}^{*})^{2}}\sqrt{2(||\boldsymbol{q}||^{2}+||\boldsymbol{q}||(\boldsymbol{q}^{\mathrm{T}}\boldsymbol{q}^{*}))}}\\&=\frac{||\boldsymbol{q}||^{2}-(\boldsymbol{q}^{\mathrm{T}}\boldsymbol{q}^{*})^{2}}{\sqrt{2||\boldsymbol{q}||^{3}}\sqrt{||\boldsymbol{q}||^{2}-(\boldsymbol{q}^{\mathrm{T}}\boldsymbol{q}^{*})^{2}}\sqrt{||\boldsymbol{q}||+(\boldsymbol{q}^{\mathrm{T}}\boldsymbol{q}^{*})}}\\&=\frac{1}{\sqrt{2}||\boldsymbol{q}||}\sqrt{1-\frac{\boldsymbol{q}^{\mathrm{T}}\boldsymbol{q}^{*}}{||\boldsymbol{q}||}}\\&=\frac{1}{\sqrt{2}||\boldsymbol{q}||}\sqrt{1-\cos(\alpha)}.\end{aligned}$$

Therefore, the inner product between $E_{\boldsymbol{Z}}[(\partial l/\partial\boldsymbol{q})]$ and $(\partial L_{Z}/\partial\boldsymbol{q})$ is given by

$$\begin{aligned}&\left\langle E_{Z}\left[\frac{\partial l}{\partial\boldsymbol{q}}\right],\frac{\partial L_{Z}}{\partial\boldsymbol{q}}\right\rangle\\&=\csc(\alpha/2)\frac{w(\alpha,\boldsymbol{q})}{2\pi}\left(\boldsymbol{p}^{\mathrm{T}}\boldsymbol{p}^{*}\right)^{2}\frac{1}{\sqrt{2}||\boldsymbol{q}||}\sqrt{1-\cos(\alpha)}\\&=\frac{w(\alpha,\boldsymbol{q})}{2\pi||\boldsymbol{q}||}\left(\boldsymbol{p}^{\mathrm{T}}\boldsymbol{p}^{*}\right)^{2}\geq 0.\end{aligned}$$

□

Clearly, when $\langle E_{\boldsymbol{Z}}[(\partial l/\partial\boldsymbol{q})],(\partial L_{Z}/\partial\boldsymbol{q})\rangle>0$, $E_{\boldsymbol{Z}}[(\partial l/\partial\boldsymbol{q})]$ is roughly in the same direction as $(\partial L_{Z}/\partial\boldsymbol{q})$. Based on Theorem 3, $\langle E_{\boldsymbol{Z}}[(\partial l/\partial\boldsymbol{q})],(\partial L_{Z}/\partial\boldsymbol{q})\rangle\geq 0$ always establishes. Moreover, $E_{\boldsymbol{Z}}[(\partial l/\partial\boldsymbol{p})]=(\partial L_{Z}/\partial\boldsymbol{p})$ by Theorem 3. Therefore, our estimated gradient decent on $L_{Z}(\boldsymbol{p},\boldsymbol{q})$ is able to behave such as the real one directly on loss function, which means that our estimated gradient is reasonable.

D. Propagation and Updates

In this section, we consider the different steps of forward and backward passes with SGD updates and whether to discretize the weights in auxiliary attention layer at each of these steps. The entire training algorithm for our method is presented in Algorithm 1.

Algorithm 1 Training Algorithm With Auxiliary Attention Layer and BN. N Represents the Number of Layers and the Activation Function is θ. Binarize() Specifies Change the Input Tensors Into Binary Ones. Weights in Auxiliary Attention Layer are Binarized Based on (4). Update() Specifies Use Gradient Descent Algorithms (SGD, ADAM Etc.) to Update the Parameters Based on Former Values and the Corresponding Gradients. η_1 and η_2 are the Learning Rate of the Original Model and the Auxiliary Attention Layer, Respectively. BatchNorm() Means Batch-Normalize the Inputs While BackBatchNorm() Means Backpropagate by the BN.

Require: A mini batch of outputs and targets(a_n, y), previous Weights W_l and M_l, previous BN parameters (γ_l, β_l), learning rate η_1 for W and learning rate η_2 for M.

Ensure: Updated Weight Matrix W^{t+1} and Auxiliary Attention Layer Weight Matrix M^{t+1}

1. Gradient Calculation:

1.1 Forward Propagation:

for $l=1$ to n **do**

$\hat{M}_l \leftarrow Binarize(M_l)$

$\hat{W}_l = W_l \odot \hat{M}_l$

$z_l \leftarrow a_{l-1}\hat{W}_l$

$\tilde{z} \leftarrow BatchNorm(z_l, \gamma_l, \beta_l)$

$a_l \leftarrow \theta(\tilde{z})$

end for

1.2 Back Propagation:

Computing $g_{a_n} = \frac{\partial \mathcal{L}}{\partial a_n}$ based on a_n and y

for $l=n$ to 1 **do**

$(g_{\gamma_l}, g_{\beta_l}) \leftarrow BackBatchNorm(g_{a_l}, z_l, \gamma_l, \beta_l)$

$g^{M}_{a_{l-1}} \leftarrow g_{a_l}\hat{M}_l$

$g_{M_l} \leftarrow (g^{W}_{a_l})^{T} a_{l-1}$

$g^{W}_{a_{l-1}} \leftarrow g_{a_l} W_l$

$g_{W_l} \leftarrow (g^{W}_{a_l})^{T} a_{l-1}$

end for

2. Parameter Update:

for $l=1$ to n **do**

$(\gamma_l^{t+1}, \beta_l^{t+1}) \leftarrow Update(\gamma_l^{t}, \beta_l^{t}, \eta_1, g_{\gamma_l}, g_{\beta_l})$

$M_l^{t+1} \leftarrow Update(M_t, g_{M_l}, \eta_2)$

$W_l^{t+1} \leftarrow Update(W_t, g_{W_l}, \eta_1)$

end for

We first calculate each layer's activations layer by layer in the forward pass after feeding the input. Meantime we can

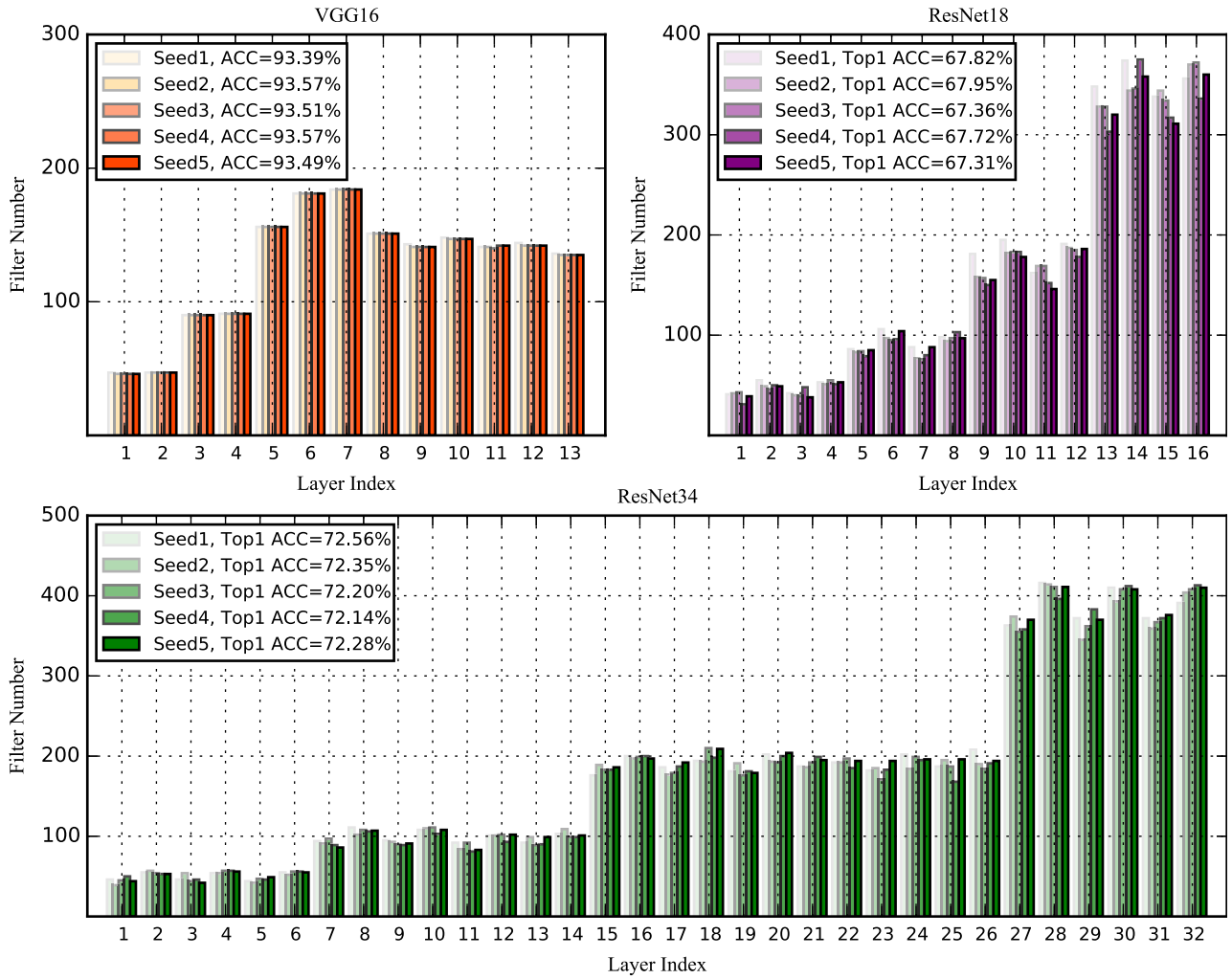


Fig. 2. Visualization of filter numbers for different pruned neural architectures. We use CIFAR-10 to train VGG16, while ResNet18 and ResNet34 are trained on ImageNet. For each model, we initialize the model with five different random seeds and then prune the original network using our ABP method.

TABLE I
EXPERIMENTAL RESULTS OF VGG-16 ON CIFAR-10

Model	Method	Ref[a]Acc. (%)	Pruned[b]Acc. (%)	Acc.↓ (%)	Params.↓ (%)	FLOPs↓ (%)
VGG-16	Pruning [19]	93.25	93.40	-0.15[c]	64.0	34.2
	SSS [14]	93.96	93.02	0.94	73.8	41.6
	Zhao [40]	93.25	93.18	**0.07**	73.3	39.1
	HRank [22]	93.96	93.43	0.78	82.9	53.5
	ABP-1 (ours)	93.96	93.75	0.21	**83.4**	**53.5**
	GAL-0.5 [23]	93.96	92.03	1.93	77.6	39.6
	HRank [22]	93.96	92.34	1.62	82.1	65.3
	ABP-2 (ours)	93.96	93.50	**0.46**	**86.2**	**66.2**
	GAL-0.1 [23]	93.96	90.73	3.23	82.2	45.2
	HRank [22]	93.96	91.23	2.73	**92.0**	76.5
	ABP-3 (ours)	93.96	92.65	**1.31**	89.8	**83.2**

[a] Ref Acc. represents the accuracy of the baseline neural network.
[b] Pruned Acc. represents the accuracy of the pruned neural network.
[c] Negative values specify the Ref Acc. is lower than the Pruned Acc.

obtain the output of the model, which is also the activation of the last layer. We refer to this step as forward propagation. Next, we use each layer's activations to calculate the training objective gradient based on the given label and each layer's parameters. Going down layer by layer, this step is conducted from the last layer until the first one. We refer to this step as backpropagation. Third, we employ the computed gradients and the previous weight values to update the parameters. We refer to this step as parameter update. BN [16] tend to be conducive for accelerating the overall training process as well

as reducing the impact of the weight perturbation. Therefore, we add it in to our actual training algorithm. As presented in Algorithm 1, one key point to be mentioned for the training procedures in our algorithm is that we only binarize the weights in the auxiliary attention layer and only during step 1 (Gradient Calculate). In step 2 (Parameter Update), the parameters in the auxiliary attention layer will not be binarized.

E. Hyperparameter Sensitivity Analysis

By designing an indicator function and inserting an auxiliary attention layer, we bring in two brand-new hyperparameters into the original neural network: the threshold parameter t the learning rate η_2 for the auxiliary attention layer. The regularization hyperparameter t is used to adjust the pruning ratio. Practically, compared to the learning rate for original weights, the learning rate η_2 of auxiliary attention layers is scheduled to be smaller than that of the original model (η_1) to avoid fluctuations of accuracy in practice.

Based on the following reasons, we conjecture that the pruning process will not be that sensitive to these hyperparameters. First, the bias of hyperparameter and our gradient estimator will not directly affect weights since we do not directly regularize on W. Second, even if the auxiliary parameters m_j^i fluctuate, our piecewise indicator function is tolerant of handling it. Third, the pruning filters are recoverable through the auxiliary attention layer when optimization proceeds. The robustness of our algorithm is presented in Fig. 2.

IV. Experiments

In this section, we present our experimental settings and evaluate the performances of our proposed method. The source code and trained models will be made available to the public.

A. Experimental Settings

1) Baselines and Datasets: To testify the effectiveness of our method, experiments on both large and small dataset, i.e., ILSVRC-2012 [30] and CIFAR-10 [18] are conducted. The ILSVRC-2012 dataset contains 1.28 million training images and 50 K validation images of 1000 classes, which is a large-scale dataset widely used in the classification task. And CIFAR-10 dataset is a small-scale dataset containing 60 000 32×32 color images of ten different classes, in which 50 000 training images and 10 000 testing images are included. We evaluate different state-of-the-art algorithms on the most prevailing CNN models, including single-branch network VGGNet [30] and the most prevailing multiple-branch network: ResNet [6].

2) Evaluation Metrics: We use floating point operations (FLOPs) and the number of parameters to represent the computational cost and model size of models, respectively, which are widely used protocols. To facilitate comparisons between different methods with different baseline accuracy, we further employ three metrics to measure the effectiveness in reducing complexity, including Params. ↓, FLOPs ↓ and Acc. ↓, which denotes the drop of parameters, computational cost, and accuracy, separately. Obviously, a better compressing performance corresponds to higher Params. ↓, FLOPs ↓, and lower Acc. ↓.

TABLE II
Layer-Wise Pruning Results and Pruned Models (ABP-1, ABP-2, and ABP-3) Statistics for VGG-16 on CIFAR-10

		Baseline	ABP-1	ABP-2	ABP-3
Input Size		32*32*3	32*32*3	32*32*3	32*32*3
Layers	CONV1_1	64	55	47	34
	CONV1_2	64	55	47	28
	CONV2_1	128	105	90	64
	CONV2_2	128	109	91	57
	CONV3_1	256	192	156	115
	CONV3_2	256	216	181	115
	CONV3_3	256	223	184	121
	CONV4_1	512	151	151	151
	CONV4_2	512	141	143	141
	CONV4_3	512	147	148	147
	CONV5_1	512	140	141	142
	CONV5_2	512	144	144	145
	CONV5_3	512	135	136	135
	FC6	512	512	512	512
	FC7	10	10	10	10
Total Parameters		15M	2.481M	2.076M	1.532M
Accuracy		93.96	93.75	93.50	92.65
FLOPs		314.95M	146.36M	106.71M	52.87M

3) Training Setting: We adopt the stochastic gradient descent (SGD) algorithm as our optimizer to minimize the loss. PyTorch [29] is used to implement our ABP. For fair comparisons, we evaluate the FLOPs and parameter reductions when the accuracy is fixed to be similar to baselines. Or in another way, we evaluate the accuracy under similar reductions of FLOPs or model sizes between algorithms. We use an NVIDIA Tesla V100 GPU as our training hardware, and the same experiments are conducted five times to get an averaged result. In the channel selection process, we set $\eta_2 = \eta_1 * 0.01$. Hyperparameter t is used to control the pruning rate of the filters, which is employed to balance between accuracy and acceleration. And t is set to be the same for all convolutional layers for simplicity. To compare more equally we adjust the value of t to generate similar pruning ratios over different architectures for comparison. It is worth noting that the network does not need fine-tuning after pruning. Instead, we simply remove the auxiliary attention layer and the corresponding filters with low attention and then obtain the compressed model.

B. CIFAR-10 Results

1) VGG: Table I presents the results with VGG-16. Several state-of-the-art methods are compared, including several adaptive importance based methods: Zhao *et al.* [40], SSS [14] and GAL [23], as well as Pruning [19], a method based on property importance. Our ABP is testified with different settings of t to obtain similar FLOPs or Parameters reduction with other methods, denoted as ABP-1, ABP-2, and ABP-3.

Compared to Pruning [19] and SSS [14] and Zhao *et al.* [40], our ABP achieves higher accuracy and most reduction in both parameters and FLOPs. As is shown in the table, ABP-1 achieves the highest accuracy (93.75%

TABLE III
EXPERIMENTAL RESULTS OF RESNET ON CIFAR-10

Model	Method	Ref Acc. (%)	Pruned Acc. (%)	Acc.↓ (%)	Params.↓ (%)	FLOPs↓ (%)
ResNet-20	SFP [7]	92.20	90.83	1.37	-[a]	42.2
	FPGM [8]	92.20	91.09	1.11	-	42.2
	CNN-FCF [20]	92.20	91.13	**1.07**	42.8	41.6
	ABP	92.15	91.03	1.12	**45.1**	**47.7**
ResNet-32	SFP [7]	92.63	92.08	0.55	-	41.5
	FPGM [8]	92.63	92.31	0.32	-	41.5
	CNN-FCF [20]	92.43	92.18	0.25	42.7	42.2
	ABP	92.63	92.55	**0.08**	**43.6**	**46.3**
ResNet-56	SFP [7]	93.59	93.78	-0.19	-	41.1
	GAL-0.6 [23]	93.26	92.98	0.28	11.8	37.6
	FPGM [8]	93.59	92.93	0.66	-	**52.6**
	NISP [38]	-	-	0.03	42.6	43.6
	CNN-FCF [20]	93.14	93.38	**-0.24**	43.1	42.8
	ABP	93.41	93.10	0.31	**45.7**	45.2
ResNet-110	SFP [7]	93.68	93.38	0.3	-	40.8
	Pruning-B [19]	93.53	93.30	0.2	32.4	38.6
	GAL-0.05 [23]	93.50	92.55	0.95	44.8	48.5
	FPGM [8]	93.68	93.73	-0.05	-	**52.3**
	NISP [38]	-	-	0.18	43.2	43.8
	CNN-FCF [20]	93.58	93.67	-0.09	43.2	43.1
	ABP	93.63	93.95	**-0.32**	**44.9**	46.2

[a] - means that the compared methods do not report the corresponding metric value.

versus 93.40% by Pruning, 93.02% by SSS, and 93.18% by Zhao) with the highest FLOPs (53.5% versus 34.2% by Pruning, 41.6% by SSS, and 39.1% by Zhao) and parameters (64.0% versus 73.8% by Pruning, 82.9% by Zhao and 73.3% by SSS). ABP is advantageous in every aspect compared with GAL [23]: For GAL-0.5, 93.39% versus 92.03% in accuracy, 86.2% versus 77.6% in parameters drop, 66.2% versus 39.6% in FLOPs drop.

Moreover, our pruned model could still give an excellent performance under extreme conditions (about 90% of the weights are removed): just 1.31% top-1 accuracy loss versus 2.73% by HRank, and 83.2% FLOPs reduction versus 76.5% by HRank, which demonstrates the superiority of pruning by training as a guiding thought for shrinking CNNs. The layer-wise pruning statistics for our method is shown in Table II. ABP-1, ABP-2, and ABP-3 denote pruned models with different sets of t.

2) ResNet20/32/56/110: We compare our ABP with different feature pruning approaches such as "filter pruning via geometric median" (FPGM) [8], "soft filter pruning" (SFP) [7], "compressing CNNs via factorized convolutional filters" (CNN-FCFs) [20], "Toward optimal structured CNN pruning via generative adversarial learning" (GAL) [23], "pruning networks using neuron importance score propagation" (NISP) [38], and "Pruning Filters for Efficient Convnets" (Pruning) [19].

We compress Residual Networks with multiple depths: 20, 32, 56, and 100, from shallow to deep. Results displayed in Table III show that under similar FLOPs or parameter reduction, our ABP could still achieve state-of-the-art performance. For ResNet-20, we give a small accuracy loss (1.12% Acc.↓) when the parameter and FLOPs reductions are much bigger (45.1% Params.↓ and 47.7% FLOPs↓). Similarly, on ResNet-32, our method achieves lower Acc.↓ (0.08%) with higher Params.↓ (43.6%) and FLOPs↓ (46.3%). Besides, we observe that our ABP achieves the biggest parameter reduction under similar FLOPs reduction and pruned accuracy compared with CNN-FCF on ResNet-56. Finally, in comparison with other methods on ResNet 110, Our method greatly reduces the model complexity (46.2% for FLOPs and 44.9% for parameters) with the best pruned accuracy raise (0.32%). The above results validate that our ABP is remarkably effective in shrinking model size and computational cost for models with residual blocks.

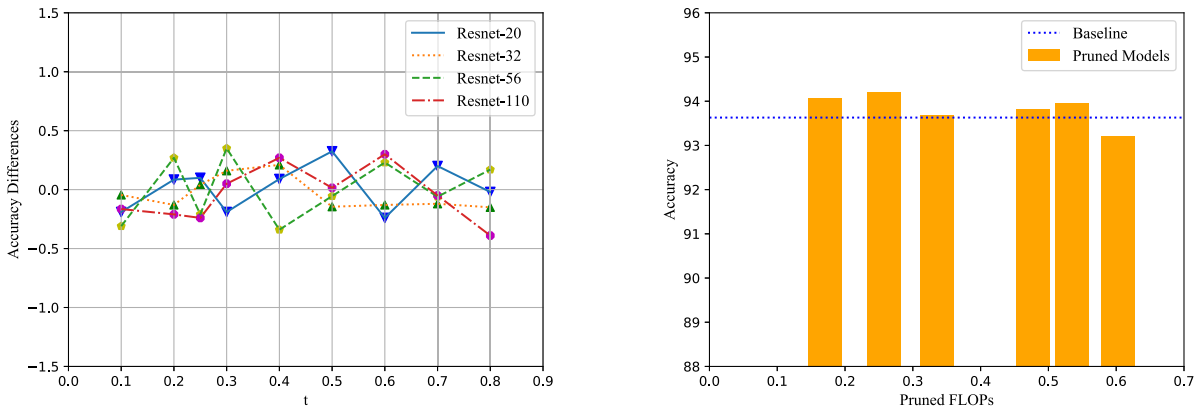
C. ILSVRC-2012 Results

ResNet18/34/50: We testify the performance of different methods on the challenging ILSVRC-2012 dataset for ResNet-18/34/50 as well. Table IV summarizes the overall results. Same with the test on CIFAR10, we do not finetune the compressed model after training.

ABP demonstrates its effectiveness again, not only on the parameters and FLOPs drop but also top-1 and top-5 accuracy. To be specific, our ABP could still yields 67.82% top-1 accuracy and 87.92% top5 accuracy on ResNet 18 while 1.86 × parameters (6.3 M versus 11.7 M) and 1.78 × FLOPs (1.02B versus 1.82B) are pruned. On ResNet-34, we achieve much higher Params.↓ (50.7%) and FLOPs↓ (47.2%) than all the other methods while the accuracy drop is basically at the same level. On ResNet-50, ABP can prune 49% of the parameters with the lowest accuracy drop. This indicates that training and selecting filters together is effective as well as scalable.

D. Robustness Test

Fig. 2 visualizes the remained filter numbers for different pruned neural architectures. For each model, we initialize the



(a) (b)

Fig. 3. (a) Accuracy differences of binary and ternary weight for ResNet-20, 32, 56, and 110 on CIFAR-10 regarding different t; and (b) ResNet-110 classification accuracy on CIFAR-10 dataset regarding varied Pruned FLOPs.

TABLE IV
EXPERIMENTAL RESULTS OF RESNET ON ILSVRC-2012

Model	Method	Top1[a]Ref.(%)	Top1↓ (%)	Top5[b]Ref.(%)	Top5↓ (%)	Params.↓ (%)	FLOPs↓ (%)
ResNet-18	SFP [7]	70.28	3.18	89.63	1.85	-	41.8
	FPGM(only) [8]	70.28	2.50	89.63	1.62	-	41.8
	FPGM(mix) [8]	70.28	2.47	89.63	1.52	-	41.8
	ABP	70.29	**2.46**	89.64	1.72	46.0	**43.7**
ResNet-34	Pruning [19]	73.23	1.06	-	-	10.8	24.2
	NISP [38]	-	0.92	-	-	43.7	43.8
	SFP [7]	73.92	2.09	90.33	1.29	-	41.8
	FPGM(only) [8]	73.92	2.13	91.62	0.92	-	41.8
	FPGM(mix) [8]	73.92	1.81	91.62	0.93	-	41.8
	CNN-FCF [20]	73.30	**0.51**	91.42	**0.47**	42.2	41.4
	ABP	73.86	1.71	91.37	0.60	**50.7**	**47.5**
ResNet-50	SFP [7]	76.15	1.54	92.87	0.81	-	41.8
	FPGM(only) [8]	76.15	1.12	92.87	0.47	-	42.2
	FPGM(mix) [8]	76.15	1.21	92.87	0.48	-	42.2
	HRank [22]	76.15	1.17	92.87	0.54	36.7	**43.7**
	ABP	75.88	**1.08**	92.76	**0.40**	**49.0**	42.8

[a] Top1 Ref. represents top1 accuracy of the baseline.
[b] Top5 Ref. represents top5 accuracy of the baseline.

model with five different random seeds and then prune the original network using our ABP method. For different seeds, the performance of our pruned model remains stable and the remained numbers of filters in the intermediate layers have few fluctuations, which indicates the robustness of our algorithm under different initialization circumstances.

E. Generalization Ability

We also conduct experiments on object detection benchmark Pascal VOC [3] dataset in order to further testify the generalization ability of our approach. We select an popular object detector: SSD [24]. For our experiments, we train the SSD architecture with the VGG-16 (uncompressed) as the base network.

Considering SSD extract intermediate feature maps for regression, we adopt two strategies to prune the filters: A. All the filters are pruned (denoted as ABP-a) B. Keep those filters that its corresponding feature maps are extracted and prune the others (denoted as ABP-b). The results are reported in Table V, where ABP-a1 and ABP-a2 denote different settings of t when using strategy A, and ABP-a2 and ABP-b share the same set up of t. From Table V, with the pruned architecture, we can achieve similar precision while reducing considerable parameters and computations. In addition, we find that strategy A could achieve higher Average Precision (75.7 versus 75.4) with more parameters reduction (57.7% versus 50.5%) and FLOPs drop (56.2% versus 52.5%).

Based on the above results, we can conclude that our method is good at shrinking feature fusion schemes while preserving original accuracy, as well as generalizing to higher-level computer vision tasks.

F. Ablation Study

1) Pruning With Ternary Weight: We analyze differences in the auxiliary attention layer parameters. In contrast to the proposed modules using binary weights (0, 1), we train the appended auxiliary attention layer with ternary weights $(-1, 0, 1)$

$$a_j^i = \begin{cases} 0, & \text{if } \left|m_j^i\right| \leq t \\ 1, & \text{if } m_j^i > t \\ -1, & \text{if } m_j^i < -t. \end{cases} \tag{16}$$

For the same set of t, the parameter and FLOPs reduction with ternary weight is the same as that of binary weight.

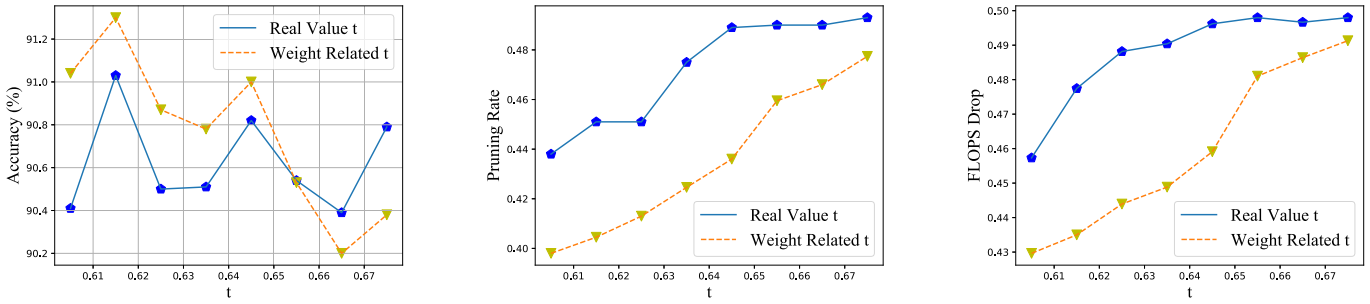


Fig. 4. Performance of ResNet-20 on CIFAR-10 regarding different t. black lines and yellow lines denote real-valued t and weight related t of three experiments, respectively: (a) accuracy with different t, (b) pruning rate with different t, and (c) FLOPs drop with different t.

TABLE V
LAYER-WISE PRUNING RESULTS AND PRUNED MODELS (ABP-A1, ABP-A2, AND ABP-B) STATISTICS FOR SSD-300 ON PASCAL VOC DATASET

		Baseline	ABP-a1	ABP-a2	ABP-b
Input Size		300*300	300*300	300*300	300*300
Layers	CONV1_1	64	59	43	43
	CONV1_2	64	59	44	44
	CONV2_1	128	114	86	86
	CONV2_2	128	118	88	88
	CONV3_1	256	171	154	154
	CONV3_2	256	193	175	175
	CONV3_3	256	206	176	176
	CONV4_1	512	384	339	339
	CONV4_2	512	384	328	328
	CONV4_3	512	376	329	512
	CONV5_1	512	373	339	339
	CONV5_2	512	369	311	331
	CONV5_3	512	385	337	337
	CONV6	1024	758	666	666
	CONV7	1024	779	676	1024
	CONV8_1	256	192	172	172
	CONV8_2	512	385	337	512
	CONV9_1	128	93	84	84
	CONV9_2	256	205	181	512
	CONV10_1	128	96	79	79
	CONV10_2	256	188	161	256
	CONV11_1	128	93	80	80
	CONV11_2	256	193	171	256
Parameters Drop		-	44.3%	57.7%	50.5%
Average Precision		76.4	75.9	75.7	75.4
FLOPs Drop		-	35.8%	56.2%	52.5%

We find that the accuracy differences of the pruned model between ternary weights and binary weights rise and fall as the change of t, which can be observed in Fig. 3(a). This indicates that adding more values in the appended auxiliary attention layer may not contribute much to the accuracy of the model. This is because new adding value (i.e., "-1") also indicates preserving the filter, which cannot bring much impact to the pruning process.

2) Effect of Threshold Hyperpatameter Design: We evaluate the effectiveness of the different designs of hyperparameter t. Normally, we adopt a real number as the threshold t to binarize the weights in the auxiliary attention layer. However, we also attempt another threshold design (weight-related)

$$t = \max(|m|) * t_0 \tag{17}$$

where m is the input filter tensor and t_0 is the coefficient to control the pruning rate of the filters.

We use the ResNet-20 with CIFAR-10 as a baseline to find out the differences. Fig. 4 shows that using real-valued t as hyperparameter to perform the binary procedure will relatively bring higher pruning rate and FLOPs drop, but weight-related t as threshold would achieve a slightly better result with the accuracy because of low sparsity.

3) Effect of Varying Pruned FLOPs: To comprehensively explore our ABP, We change the pruned ratio of FLOPs for ResNet-110. The results is summarized in Fig. 3(b). We can see that the accuracy of the pruned network even exceeds the baseline when the pruned FLOPs is less than 0.55. This indicates that our ABP may introduce a regularization effect to the original model.

V. CONCLUSION

We propose a novel filter pruning method called ABP for feature fusion schemes of DNNs. To that effect, we add an auxiliary layer and regularize parameters in an auxiliary layer instead of original weight values. Then, we mathematically prove that our gradient estimator is reasonable and the estimated gradient could behave such as the true descent on the loss function. In addition, we combine filter training with filter pruning by inserting our gradient estimator into the back-propagation framework. Experiment results on popular deep neural architectures illustrate our ABP's effectiveness in shrinking model size and computational complexity. In the future work, we plan to further design an automobile feature pruning framework that can adjust the hyperparameters to meet the requirements.

APPENDIX

ADDITIONAL SUPPORTING LEMMAS

Lemma 1: Let $\boldsymbol{Z}$ be a random vector sampled from Gaussian Distribution $\mathcal{N}(0, 1)$. Suppose the angel between vector $\boldsymbol{q}$ and $\hat{\boldsymbol{q}}$ is α, then

$$E\left[1_{\{\boldsymbol{Z}^{\mathrm{T}}\boldsymbol{q}>0\}}\right] = \frac{1}{2}$$

$$E\left[1_{\{\boldsymbol{Z}^{\mathrm{T}}\boldsymbol{q}>0, \boldsymbol{Z}^{\mathrm{T}}\hat{\boldsymbol{q}}>0\}}\right] = \frac{\pi - \alpha}{2\pi}$$

$$E\left[\boldsymbol{Z}1_{\{\boldsymbol{Z}_T\boldsymbol{q}>0\}}\right] = \frac{1}{\sqrt{2\pi}}\frac{\boldsymbol{q}}{||\boldsymbol{q}||}$$

$$E\big[\boldsymbol{Z}1_{\{\boldsymbol{Z}_T\boldsymbol{q}>0,\boldsymbol{Z}_T\hat{\boldsymbol{q}}>0\}}\big]=\frac{\cos(\alpha/2)}{\sqrt{2\pi}}\frac{\frac{\boldsymbol{q}}{||\boldsymbol{q}||}+\frac{\hat{\boldsymbol{q}}}{||\hat{\boldsymbol{q}}||}}{\left\|\frac{\boldsymbol{q}}{||\boldsymbol{q}||}+\frac{\hat{\boldsymbol{q}}}{||\hat{\boldsymbol{q}}||}\right\|}.$$

Proof: Without loss of generality, suppose $\boldsymbol{q}=[\boldsymbol{q}_1,0^{\mathrm{T}}]^{\mathrm{T}}$ with $\boldsymbol{q}_1>0$, $\hat{\boldsymbol{q}}=[\hat{\boldsymbol{q}}_1,\hat{\boldsymbol{q}}_2,0^{\mathrm{T}}]^{\mathrm{T}}$. Then

$$\begin{aligned}E\big[1_{\{\boldsymbol{Z}^{\mathrm{T}}\boldsymbol{q}>0\}}\big]&=P(\boldsymbol{Z}_1>0)=\frac{1}{2}\\E\big[1_{\{\boldsymbol{Z}^{\mathrm{T}}\boldsymbol{q}>0,\boldsymbol{Z}^{\mathrm{T}}\hat{\boldsymbol{q}}>0\}}\big]&=P\big(\boldsymbol{Z}^{\mathrm{T}}\boldsymbol{q}>0,\boldsymbol{Z}^{\mathrm{T}}\hat{\boldsymbol{q}}>0\big)\\&=\frac{\pi-\alpha}{2\pi}.\end{aligned}$$

The third claim can be proved by Lemma A.1 from [2]. Using the polar representation of 2-D Gaussian random variables

$$\begin{aligned}&E\big[\boldsymbol{Z}_1 1_{\{\boldsymbol{Z}_T\boldsymbol{q}>0,\boldsymbol{Z}_T\hat{\boldsymbol{q}}>0\}}\big]\\&\quad=\frac{1}{2\pi}\int_0^{+\infty}r^2e^{-\frac{r^2}{2}}dr\int_{-\frac{\pi}{2}+\alpha}^{\frac{\pi}{2}}\cos(\phi)d\phi=\frac{1+\cos\alpha}{2\sqrt{2\pi}}\\&E\big[\boldsymbol{Z}_2 1_{\{\boldsymbol{Z}_T\boldsymbol{q}>0,\boldsymbol{Z}_T\hat{\boldsymbol{q}}>0\}}\big]\\&\quad=\frac{1}{2\pi}\int_0^{+\infty}r^2e^{-\frac{r^2}{2}}dr\int_{-\frac{\pi}{2}+\alpha}^{\frac{\pi}{2}}\sin(\phi)d\phi=\frac{\sin\alpha}{2\sqrt{2\pi}}\\&E\big[\boldsymbol{Z}_i 1_{\{\boldsymbol{Z}_T\boldsymbol{q}>0,\boldsymbol{Z}_T\hat{\boldsymbol{q}}>0\}}\big]\\&\quad=0\ (i\geq 3).\end{aligned}$$

where ϕ is the angle and r is the radius. $dP\phi=(1/2\pi)d\phi$ and $dPr=re^{-(r^2/2)}dr$. Then

$$\begin{aligned}&E\big[\boldsymbol{Z}1_{\{\boldsymbol{Z}_T\boldsymbol{q}>0,\boldsymbol{Z}_T\hat{\boldsymbol{q}}>0\}}\big]\\&\quad=\frac{1}{\sqrt{2\pi}}\big[\cos^2(\alpha/2),\sin(\alpha/2)\cos(\alpha/2),0^{\mathrm{T}}\big]\\&\quad=\frac{\cos(\alpha/2)}{\sqrt{2\pi}}\big[\cos(\alpha/2),\sin(\alpha/2),0^{\mathrm{T}}\big]\\&\quad=\frac{\cos(\alpha/2)}{\sqrt{2\pi}}\frac{\frac{\boldsymbol{q}}{||\boldsymbol{q}||}+\frac{\hat{\boldsymbol{q}}}{||\hat{\boldsymbol{q}}||}}{\left\|\frac{\boldsymbol{q}}{||\boldsymbol{q}||}+\frac{\hat{\boldsymbol{q}}}{||\hat{\boldsymbol{q}}||}\right\|}\end{aligned}$$

where $(\boldsymbol{q}/||\boldsymbol{q}||)$ and $((\boldsymbol{q}/||\boldsymbol{q}||)+(\hat{\boldsymbol{q}}/||\hat{\boldsymbol{q}}||))/(||(\boldsymbol{q}/||\boldsymbol{q}||)+(\hat{\boldsymbol{q}}/||\hat{\boldsymbol{q}}||)||)$ are unit-normed vectors. The angle between them is $\alpha/2$. □

Lemma 2: Let $\boldsymbol{Z}$ be a random vector sampled from Gaussian Distribution $\mathcal{N}(0,1)$. Suppose the angel between vector $\boldsymbol{q}$ and $\hat{\boldsymbol{q}}$ is α

$$\begin{aligned}E\big[\boldsymbol{Z}1_{\{-t<\boldsymbol{Z}_T\boldsymbol{q}<t\}}\big]&=\upsilon(0,\boldsymbol{q})\frac{\boldsymbol{q}}{||\boldsymbol{q}||}\\E\big[\boldsymbol{Z}1_{\{-t<\boldsymbol{Z}_T\boldsymbol{q}<t,\boldsymbol{Z}_T\hat{\boldsymbol{q}}>0\}}\big]&=\left[(\upsilon(\alpha,\boldsymbol{q})-\cot(\alpha/2)w(\alpha,\boldsymbol{q}))\frac{\boldsymbol{q}}{||\boldsymbol{q}||}\right.\\&\quad\left.+\csc(\alpha/2)w(\alpha,\boldsymbol{q})\frac{\frac{\boldsymbol{q}}{||\boldsymbol{q}||}+\boldsymbol{q}^*}{\left\|\frac{\boldsymbol{q}}{||\boldsymbol{q}||}+\boldsymbol{q}^*\right\|}\right]\end{aligned}$$

where

$$\begin{aligned}\upsilon(\alpha,\boldsymbol{q})&=\frac{1}{2\pi}\int_{-\frac{\pi}{2}}^{\frac{\pi}{2}+\alpha}\cos(\phi)\int_{-\frac{\sec(\phi)}{t||\boldsymbol{q}||}}^{\frac{\sec(\phi)}{t||\boldsymbol{q}||}}r^2e^{-\frac{r^2}{2}}drd\phi\\w(\alpha,\boldsymbol{q})&=\frac{1}{2\pi}\int_{-\frac{\pi}{2}}^{\frac{\pi}{2}+\alpha}\sin(\phi)\int_{-\frac{\sec(\phi)}{t||\boldsymbol{q}||}}^{\frac{\sec(\phi)}{t||\boldsymbol{q}||}}r^2e^{-\frac{r^2}{2}}drd\phi.\end{aligned}$$

Proof: Following the above proof, suppose: $\boldsymbol{q}=[\boldsymbol{q}_1,0^{\mathrm{T}}]^{\mathrm{T}}$ with $\boldsymbol{q}_1>0$, $\hat{\boldsymbol{q}}=[\hat{\boldsymbol{q}}_1,\hat{\boldsymbol{q}}_2,0^{\mathrm{T}}]^{\mathrm{T}}$, we can get

$$\begin{aligned}&E\big[\boldsymbol{Z}_1 1_{\{-t<\boldsymbol{Z}_T\boldsymbol{q}<t\}}\big]\\&\quad=\frac{1}{2\pi}\int_{-\frac{\pi}{2}}^{\frac{\pi}{2}}\cos(\phi)\int_{-\frac{\sec(\phi)}{t||\boldsymbol{q}||}}^{\frac{\sec(\phi)}{t||\boldsymbol{q}||}}r^2e^{-\frac{r^2}{2}}drd\phi=\upsilon(0,\boldsymbol{q})\\&\quad E\big[\boldsymbol{Z}_2 1_{\{-t<\boldsymbol{Z}_T\boldsymbol{q}<t\}}\big]\\&\quad=\frac{1}{2\pi}\int_{-\frac{\pi}{2}}^{\frac{\pi}{2}}\sin(\phi)\int_{-\frac{\sec(\phi)}{t||\boldsymbol{q}||}}^{\frac{\sec(\phi)}{t||\boldsymbol{q}||}}r^2e^{-\frac{r^2}{2}}drd\phi=w(0,\boldsymbol{q})=0\\&E\big[\boldsymbol{Z}_i 1_{\{-t<\boldsymbol{Z}_T\boldsymbol{q}<t\}}\big]\\&\quad=0\ (i\geq 3).\end{aligned}$$

The second equation stands because the integrand is an odd function in ϕ. From the above, the first claim holds. Then we have

$$\begin{aligned}&E\big[\boldsymbol{Z}_1 1_{\{-t<\boldsymbol{Z}_T\boldsymbol{q}<t,\boldsymbol{Z}_T\hat{\boldsymbol{q}}>0\}}\big]\\&\quad=\frac{1}{2\pi}\int_{-\frac{\pi}{2}}^{\frac{\pi}{2}+\alpha}\cos(\phi)\int_{-\frac{\sec(\phi)}{t||\boldsymbol{q}||}}^{\frac{\sec(\phi)}{t||\boldsymbol{q}||}}r^2e^{-\frac{r^2}{2}}drd\phi=\upsilon(\alpha,\boldsymbol{q}).\end{aligned}$$

Similarly, $E[\boldsymbol{Z}_2 1_{\{-t<\boldsymbol{Z}_T\boldsymbol{q}<t,\boldsymbol{Z}_T\hat{\boldsymbol{q}}>0\}}]=w(\alpha,\boldsymbol{q})$. So

$$E\big[\boldsymbol{Z}1_{\{-t<\boldsymbol{Z}_T\boldsymbol{q}<t,\boldsymbol{Z}_T\hat{\boldsymbol{q}}>0\}}\big]=\big[\upsilon(\alpha,\boldsymbol{q}),w(\alpha,\boldsymbol{q}),0^{\mathrm{T}}\big].$$

So it is not difficult to testify that the second identity stands because

$$\begin{aligned}\frac{\boldsymbol{q}}{||\boldsymbol{q}||}&=\big[1,0^{\mathrm{T}}\big]\\\frac{\frac{\boldsymbol{q}}{||\boldsymbol{q}||}+\boldsymbol{q}^*}{\left\|\frac{\boldsymbol{q}}{||\boldsymbol{q}||}+\boldsymbol{q}^*\right\|}&=\big[\cos(\alpha/2),\sin(\alpha/2),0^{\mathrm{T}}\big].\end{aligned}$$

□

REFERENCES

[1] A. Brutzkus and A. Globerson, "Globally optimal gradient descent for a convnet with Gaussian inputs," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2017, pp. 605–614.

[2] S. Du, J. Lee, Y. Tian, A. Singh, and B. Poczos, "Gradient descent learns one-hidden-layer CNN: Don't be afraid of spurious local minima," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2018, pp. 1339–1348.

[3] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The Pascal visual object classes challenge: A retrospective," *Int. J. Comput. Vis.*, vol. 111, no. 1, pp. 98–136, Jan. 2015.

[4] Y. Guo, A. Yao, and Y. Chen, "Dynamic network surgery for efficient DNNs," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2016, pp. 1379–1387.

[5] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2016, pp. 1–14.

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

[7] Y. He, G. Kang, X. Dong, Y. Fu, and Y. Yang, "Soft filter pruning for accelerating deep convolutional neural networks," in *Proc. 27th Int. Joint Conf. Artif. Intell.*, Jul. 2018, pp. 2234–2240.

[8] Y. He, P. Liu, Z. Wang, Z. Hu, and Y. Yang, "Filter pruning via geometric median for deep convolutional neural networks acceleration," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 4340–4349.

[9] Y. He, X. Zhang, and J. Sun, "Channel pruning for accelerating very deep neural networks," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 1389–1397.

[10] G. Hinton, "Neural networks for machine learning," *Coursera*, vol. 264, no. 1, pp. 2146–2153, 2012.

[11] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," in *Proc. Neural Inf. Process. Systems (NIPS)*, 2015, pp. 1–9.

[12] H. Hu, R. Peng, Y.-W. Tai, and C.-K. Tang, "Network trimming: A data-driven neuron pruning approach towards efficient deep architectures," 2016, *arXiv:1607.03250*. [Online]. Available: http://arxiv.org/abs/1607.03250

[13] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 7132–7141.

[14] Z. Huang and N. Wang, "Data-driven sparse structure selection for deep neural networks," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 304–320.

[15] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2016, pp. 4107–4115.

[16] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," 2015, *arXiv:1502.03167*. [Online]. Available: http://arxiv.org/abs/1502.03167

[17] X. Jiang, Y. Pang, M. Sun, and X. Li, "Cascaded subpatch networks for effective CNNs," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 7, pp. 2684–2694, Jul. 2018.

[18] A. Krizhevsky *et al.*, "Learning multiple layers of features from tiny images," Citeseer, Princeton, NJ, USA, Tech. Rep., 2009.

[19] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient convnets," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2017, pp. 1–13.

[20] T. Li, B. Wu, Y. Yang, Y. Fan, Y. Zhang, and W. Liu, "Compressing convolutional neural networks via factorized convolutional filters," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 3977–3986.

[21] Y. Li, Y. Pang, J. Cao, J. Shen, and L. Shao, "Improving single shot object detection with feature scale unmixing," *IEEE Trans. Image Process.*, vol. 30, pp. 2708–2721, 2021.

[22] M. Lin *et al.*, "HRank: Filter pruning using high-rank feature map," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 1529–1538.

[23] S. Lin *et al.*, "Towards optimal structured CNN pruning via generative adversarial learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 2790–2799.

[24] W. Liu *et al.*, "SSD: Single shot multibox detector," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*. Berlin, Germany: Springer, 2016, pp. 21–37.

[25] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, and C. Zhang, "Learning efficient convolutional networks through network slimming," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 2736–2744.

[26] Z. Liu, M. Sun, T. Zhou, G. Huang, and T. Darrell, "Rethinking the value of network pruning," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2019, pp. 1–21.

[27] J.-H. Luo, J. Wu, and W. Lin, "ThiNet: A filter level pruning method for deep neural network compression," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 5058–5066.

[28] Y. Pang, J. Cao, Y. Li, J. Xie, H. Sun, and J. Gong, "TJU-DHD: A diverse high-resolution dataset for object detection," *IEEE Trans. Image Process.*, vol. 30, pp. 207–219, 2021.

[29] A. Paszke *et al.*, "Automatic differentiation in pytorch," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2017, pp. 1–4.

[30] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2015, pp. 1–14.

[31] S. A. Taghanaki, K. Abhishek, J. P. Cohen, J. Cohen-Adad, and G. Hamarneh, "Deep semantic segmentation of natural and medical images: A review," *Artif. Intell. Rev.*, vol. 54, no. 1, pp. 137–178, 2020.

[32] F. Wang *et al.*, "Residual attention network for image classification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 3156–3164.

[33] H. Wang, C. Qin, Y. Bai, and Y. Fu, "Dynamical isometry: The missing ingredient for neural network pruning," 2021, *arXiv:2105.05916*. [Online]. Available: http://arxiv.org/abs/2105.05916

[34] H. Wang, Q. Zhang, Y. Wang, and H. Hu, "Structured probabilistic pruning for convolutional neural network acceleration," in *Proc. Brit. Mach. Vis. Conf. (BMVC)*, 2018, p. 3.

[35] J. Xie, Y. Pang, H. Cholakkal, R. Anwer, F. Khan, and L. Shao, "PSC-Net: Learning part spatial co-occurrence for occluded pedestrian detection," *Sci. China Inf. Sci.*, vol. 64, no. 2, pp. 1–13, Feb. 2021.

[36] K. Yamamoto and K. Maeno, "PCAS: Pruning channels with attention statistics for deep network compression," in *Proc. Brit. Mach. Vis. Conf. (BMVC)*, 2019, pp. 1–13.

[37] A. Yang, B. Yang, Z. Ji, Y. Pang, and L. Shao, "Lightweight group convolutional network for single image super-resolution," *Inf. Sci.*, vol. 516, pp. 220–233, Apr. 2020.

[38] R. Yu *et al.*, "NISP: Pruning networks using neuron importance score propagation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 9194–9203.

[39] T. Zhang *et al.*, "A systematic DNN weight pruning framework using alternating direction method of multipliers," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 184–199.

[40] C. Zhao, B. Ni, J. Zhang, Q. Zhao, W. Zhang, and Q. Tian, "Variational convolutional neural network pruning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 2780–2789.

[41] K. Zhong, Z. Song, P. Jain, L. Peter Bartlett, and S. Inderjit Dhillon, "Recovery guarantees for one-hidden-layer neural networks," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2017, pp. 4140–4149.

[42] Z. Zhuang *et al.*, "Discrimination-aware channel pruning for deep neural networks," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2018, pp. 875–886.