Towards Better Structured Pruning Saliency by Reorganizing Convolution

Xinglong Sun[1, 2], Humphrey Shi[2, 3]
[1]Stanford Univeristy, [2]SHI Labs @ Georgia Tech & UIUC & U of Oregon, [3]Picsart AI Research (PAIR)

Abstract

*We present **SPSRC**, a novel, simple and effective framework to extract enhanced **S**tructured **P**runing **S**aliency scores by **R**eorganizing **C**onvolution. We observe that performance of pruning methods have gradually plateaued recently and propose to make better use of the learned convolutional kernel weights simply after a few steps of transformations. We firstly re-organize the convolutional operations between layers as matrix multiplications and then use the singular values and the matrix norms of the transformed matrices as saliency scores to decide what channels to prune or keep. We show both analytically and empirically that the long-standing kernel-norm-based channel importance measurement, like L1 magnitude, is not precise enough possessing a very obvious weakness of lacking spatial saliency but can be improved with SPSRC. We conduct extensive experiments across different settings and configurations and compare with the counterparts without our SPSRC along with other popular methods, observing obvious improvements. Our code is available at: https://github.com/AlexSunNik/SPSRC/tree/main.*

1. Introduction

Convolutional neural networks (CNNs) [31] have seen great success in computer vision in the recent years, especially in fundamental vision tasks such as image classification [6, 30], object detection [18, 40] and segmentation [4, 45]. While a plethora of new model architectures have been developed or continue to be proposed, convolutional operations usually stay a constant component in such architecture designs. Hence, how to effectively compress convolutional layers in deep neural networks for efficient storage and computation has been a very active research area, and various approaches have been proposed and developed [7] over the years.

In general, neural network compression techniques can be categorized [7] into pruning [2, 16, 17, 32, 36, 42], quantization [5, 44, 61], low-rank factorization [8, 38, 64] or knowledge distillation [23, 28, 41]. In this paper, we will focus on network pruning, the class of techniques where certain parts of model parameters are discarded to reduce model storage and computation while retaining accuracy as much as possible. Network pruning methods can be further divided into unstructured pruning and structured pruning. Unstructured pruning methods [13, 16, 17, 32, 33, 55, 56] removes individual neurons or weight connections of less importance without consideration for where they occur within each tensor. It can reduce the number of parameters by more than 90% without harming much accuracy. However, it additionally requires sparse BLAS libraries or even specialized hardware [15] and may not improve performance on commodity hardware until a large fraction of weights has been pruned [49]. On the other hand, structured pruning methods [12, 36, 42, 47, 48, 51–53, 57, 59] prune parameters under structure constraints. It involves pruning weights in groups, for example removing convolutional filters. Structurally-pruned models preserve dense computations thus enjoy immediate performance improvement without specialized hardware or library support. Our proposed work falls within the structured pruning category, identifies an everlasting issue of saliency score used in many structured pruning works [20, 36, 47, 52], and proposes a novel framework to better measure the saliency score used for pruning channels.

For structured network pruning, the most representative works are aiming at pruning channels (also called filters or kernels) in convolutional layers, i.e. to identify the best set of filters to discard, which should yield the highest compression ratio with the lowest decrease in accuracy. Channel pruning methods usually try to assign the best saliency score (also called importance score) to each channel and perform pruning based on these importance measurements. Existing channel pruning methods usually either explicitly calculate the channel importance scores based on some intrinsic properties of the pre-trained models (i.e. **property-based**) or get the scores implicitly by including the pruning requirement as part of the model training process (i.e. **learning-based**). Property-based methods [24, 28, 36, 37, 47, 52] are usually easy to implement and can sometimes provide useful insights into understanding pre-trained models. Learning-based methods [1, 22, 39, 42], though often better in accuracy, have many constraints and are difficult to be utilized in scenarios where specialized training is infeasible.

Despite the diversity of different kinds of pruning methods proposed these years, works like [10] have also discovered that they all lead to very similar accuracy-sparsity tradeoffs. Performance of standard pruning methods have gradually plateaued these years, and an increasing number of works began to focus on directions like hardware-aware pruning [27, 34, 52] and reducing the training cost [11, 63]. Following the joint efforts to exceed the limits of standard pruning works, in this work, we introduce **SPSRC**, a novel, simple and effective framework to extract better structured pruning saliency scores by a few simple steps of reorganizing convolution. Ever since works like L1 [36], convolution-kernel-based pruning scores have been widely adopted in many methods [20, 47, 52]. However, we study and find that some everlasting issues exist with such methods: (1) theoretical works and analysis of pruning effectiveness are very limited since convolution itself does not exhibit a clear information flow (2) the current scores do not capture the spatial saliency of a convolution kernel. For example, in case of zero-padded convolution, we use the central element of the kernel more often than the other elements, and we should weigh it more when calculating a pruning score. We aim to solve (1) and (2) simultaneously by introducing a transformation step to reorganize convolutional kernels at each layer as matrix multiplication. Many matrix analysis and decomposition methods hereby become useful to extract information. We argue that this organization step is significantly helpful to extract better importance measurement of each channel from the trained weights. From Singular Value Decomposition(SVD) of the reorganized matrices, we observe that the singular values of the transformed convolution matrix could serve as a reliable measurement and hint of channel importance. We thus propose three variants: **SPSRC-frobenius**, **SPSRC-spectral**, and **SPSRC-nuclear** corresponding to employing different norms of the transformed matrix as pruning metric. We demonstrate both analytically(3.4.1) and empirically(4) that the weaknesses of the convolution-kernel-based methods like the popular L1 [36] can be solved by SPSRC reorganization for more accurate importance measurement.

We conduct extensive experiments to verify our assumptions. We performed compression on three popular network architectures, VGGNet [54], ResNet [19], and DenseNet [25], and evaluate on three different benchmarks, CIFAR-10 [29], CIFAR-100 [29], and ImageNet [50]. The results demonstrate that SPSRC obtains better channel importance measurement and successfully improves the previous convolution-kernel-based scores [20, 36, 47]. We also include extensive ablation studies to further validate the effectiveness of our method.

We summarize our contributions as follows:

- We identify everlasting intrinsic issues of the long-standing convolution-kernel-based pruning scores like lacking spatial saliency.
- We propose a novel, simple, and effective channel pruning framework to transform convolution kernels at each layer into a matrix multiplication for better analysis and saliency score measurement, addressing the above identified issue.
- We conduct extensive experiments and show the effectiveness of SPSRC across many different settings and configurations.

2. Related Works

There have been various channel pruning methods proposed in recent years, either exploiting different properties of the trained model or devising a novel learning paradigm to adaptively make pruning decisions. We separate channel pruning methods into two categories based on the evaluation of the importance scores.

2.1. Property-based Methods

The importance scores are calculated based on the intrinsic properties of the models. These methods can directly operate on a pre-trained model without modifications of the target function or optimization settings. Therefore, they are usually easy to implement and have more universal applications. However, pruning is usually followed by a large decrease in accuracy, which is then compensated by additional retraining or finetuning epochs.

Hu *et al*. [24] proposes a data-driven method to prune filters with high sparsity in activations. Li *et al*. [36] assumed the channels with small absolute sum of convolution kernels are less important thus should be removed first. A similar assumption is also made in unstructured pruning [13, 16]. Molchanov *et al*. [47, 48] takes the first-order gradient into account when evaluating the channel importance scores to maximally preserve model loss [46]. He *et al*. [21] removed the layers closest to the geometric median of all channels to maximize the representation ability of the model. Similar to He *et al*. [19], Lin *et al*. [37] proposed another data-driven method, arguing that filters generating low-rank features should be pruned first. Some latest methods like HALP [51, 53] consider hardware-aware pruning. They leveraged property importance scores like Taylor [47] and solved a constrained optimization problem under a given hardware inference latency target.

2.2. Learning-based Methods

The calculation of the importance scores usually involve a learning process to make adaptive pruning decisions. These methods often encode the pruning requirement

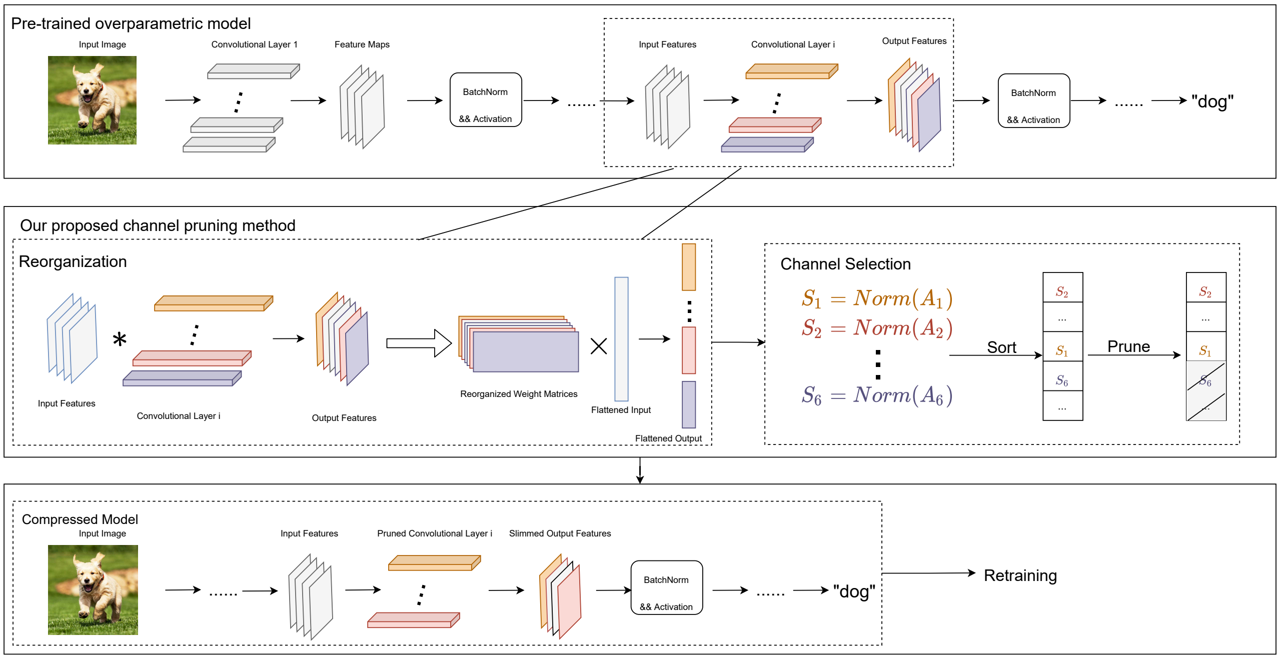

Figure 1. Schematics of our proposed method. Example of pruning a certain layer is shown here. We first reorganize the convolution kernel weights connecting the feature maps to a set of matrices. Then, we choose channels based on the norms of these matrices, creating a thinner compressed model. We additionally retrain the model to compensate for the accuracy loss.

as part of the objective function and perform joint optimization.

He *et al*. [22] enforced sparsity regularization into the training objective and used group LASSO to push all the weight parameters corresponding to the same channel towards zero simultaneously during training. Liu *et al*. [42] trained the model with channel sparsity regularization and removed filters with small scaling factors in the BN layer. Huang *et al*. [26] added mask parameter as scaling factors, and filters corresponding to a scaling factor of zero are removed. Lin *et al*. [39] used adversarial learning to aid the channel selection.

3. The Proposed Method

3.1. Notation and Formulation

Suppose we're given a model with K convolutional layers. Let's consider the i^{th} convolutional layer and represent the input feature maps at the i^{th} layer as $I^{(i)}$ and the output feature maps at the i^{th} layer as $O^{(i)}$. Moreover, suppose $I^{(i)} \in \mathbb{R}^{N_i \times h_i \times w_i}$ and $O^{(i)} \in \mathbb{R}^{N_{i+1} \times h_{i+1} \times w_{i+1}}$, where N_i and N_{i+1} represent the number of input and output channels and (h_i, w_i) and (h_{i+1}, w_{i+1}) represent the input and output feature map size. We can then represent the weights at the i^{th} layer as $W^{(i)} \in \mathbb{R}^{N_{i+1} \times N_i \times k_i \times k_i}$, where k_i denotes the kernel size. We can also represent the j^{th} filter corresponding to the j^{th} output channel, $O_j^{(i)} \in \mathbb{R}^{h_{i+1} \times w_{i+1}}$, as $W_j^{(i)} \in \mathbb{R}^{N_i \times k_i \times k_i}$.

We can thus show the convolution operation taken at the i^{th} layer by writing the relation between the m^{th} output channel and N_i input channels as:

$$O_m^{(i)} = \sum_{n=1}^{N_i} I_n^{(i)} * W_{m,n}^{(i)} \tag{1}$$

Obviously, we have N_{i+1} such equations with m taking value from 1 to N_{i+1}.

Now, suppose that we only want to keep P output feature maps in $O^{(i)}$. Our goal is thus to devise an importance function, $importance(.)$, such that we can decide whether to keep $O_m^{(i)}$ by evaluating $importance(W_m^{(i)})$, keeping P output channels that have the highest importance scores.

We can see that such pruning affects two convolutional layers. After pruning, the shape of tensor $W^{(i)}$ becomes (P, N_i, k_i, k_i), and the shape of tensor $W^{(i+1)}$ becomes $(N_{i+2}, P, k_{i+1}, k_{i+1})$. Furthermore, the number of operations at the i^{th} layer is reduced by $O(PN_ik_i^2h_{i+1}w_{i+1})$, and the number of operations at the $(i+1)^{th}$ layer is reduced by $O(PN_{i+2}k_{i+1}^2h_{i+2}w_{i+2})$, bringing the total operation reduction to $O(P(N_ik_i^2h_{i+1}w_{i+1} + N_{i+2}k_{i+1}^2h_{i+2}w_{i+2}))$.

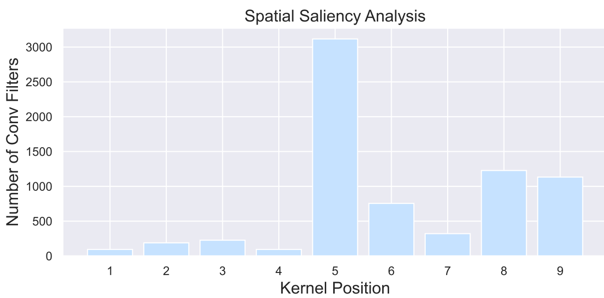

Figure 2. Times of each kernel location being the salient position.

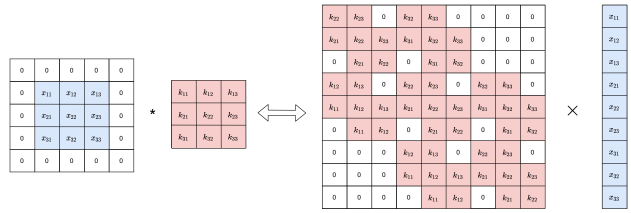

Figure 3. Example of reorganization of convolution to matrix multiplication.

3.2. Spatial Saliency

Now, we analyze a very intrinsic but also everlasting issue of many convolution-kernel-based pruning scores [20, 36, 47, 52]: the lack of consideration of spatial saliency. We consider a spatial position in a convolution kernel more important if it has a larger norm than other positions. Following the above notations, a regular filter kernel is represented as $W_m^{(i)} \in \mathbb{R}^{N_i \times k_i \times k_i}$. The norm at position x is calculated by $\|W_m^{(i)}[x]\| = \sqrt{\sum_{n=1}^{N_i} |W_{m,n}^{(i)}[x]|^2}$, for $x = 1, 2, 3, ..., k^2$. We examine the ResNet34 model and plot a histogram of the times of each kernel location being the salient position in Figure 2. We limit the scope to 3×3 kernel at this point. We could easily observe that the central kernel element gets to be the salient position most frequently among the filters, weighing more than the surrounding neighbors. Intuitively, in the case of zero-padding convolution, we do use the central elements more often than the edge and corner elements during computation. This also aligns with the findings in the very latest paper [3].

Equipped with the above knowledge, ***we can easily observe that standard convolution-kernel-based scores do not take this spatial saliency into consideration***. For example, L1 score [36] can be represented as $\|K\|^2 = \sum_{i=1}^{3} \sum_{j=1}^{3} k_{ij}^2$, which assigns uniform weights over all positions. Motivated by this, we propose the following reorganization mechanism to better measure the channel saliency. We also provide another detailed comparison in the following Sec.3.4.1

3.3. Reorganization of Convolution

In order to better analyze the pre-trained weights $W^{(i)}$ for importance calculation, we reorganize the convolution operation to a matrix multiplication. Specifically, we transform the convolution filter $W_{m,n}^{(i)}$ to matrix $A_{m,n}^{(i)}$ and rewrite equation 1 as:

$$\vec{O_m^{(i)}} = \sum_{n=1}^{N_i} A_{m,n}^{(i)} \vec{I_n^{(i)}} \tag{2}$$

, where $\vec{I_n^{(i)}} \in \mathbb{R}^{h_i w_i}$ and $\vec{O_m^{(i)}} \in \mathbb{R}^{h_{i+1} w_{i+1}}$ represent the flattened input and output feature maps, and $A_{m,n}^{(i)} \in \mathbb{R}^{h_{i+1} w_{i+1} \times h_i w_i}$ represents the transformed convolution matrix.

We can further reduce equation 2 by removing the summation as follows:

$$\vec{O_m^{(i)}} = [A_{m,1}^{(i)}, A_{m,2}^{(i)}, ..., A_{m,N_i}^{(i)}][\vec{I_1^{(i)}}, \vec{I_2^{(i)}}, ..., \vec{I_{N_i}^{(i)}}]^\top \tag{3}$$

$$= \hat{A_m^{(i)}} \hat{I^{(i)}} \tag{4}$$

, where $\hat{A_m^{(i)}} \in \mathbb{R}^{h_{i+1} w_{i+1} \times N_i h_i w_i}$ represents a single matrix constructed by stacking weight matrices $A_{m,n}^{(i)}$, and $\hat{I^{(i)}} \in \mathbb{R}^{N_i h_i w_i}$ represents a single vector constructed by stacking feature vectors $I_n^{(i)}$. Figure 3 provides an illustration of such process.

3.4. Decomposition and Metrics

Through Singular Value Decomposition(SVD), we can rewrite Equation 4 as:

$$\hat{A_m^{(i)}} = \sum_{k=1}^{r} \sigma_k u_k v_k^\top \tag{5}$$

$$\vec{O_m^{(i)}} = \sum_{k=1}^{r} \sigma_k u_k v_k^\top \hat{I^{(i)}} \tag{6}$$

, where r denotes the rank of matrix $\hat{A_m^{(i)}}$, and σ_k, u_k, v_k represent the singular value, left singular vector, and right singular vector of $\hat{A_m^{(i)}}$ respectively. We can see here that the set of singular values, $\{\sigma_k | 1 \le k \le r\}$, can serve as reliable measurement and hint of information richness and channel importance. We thus intuitively propose three metrics and name them as: **SPSRC-frobenius**, **SPSRC-spectral**, and **SPSRC-nuclear**. Note that in the following analysis, for simplicity, we denote $\hat{A_m^{(i)}}$ simply by A.

3.4.1 Frobenius Norm

After the reorganization, we could simply use the frobenius norm of matrix A to compute saliency score. Formally, frobenius norm is defined as:

$$\|A\|_F = \sqrt{trace(A^T A)} = \sqrt{\sum_k^r \sigma_k^2} = \sqrt{\sum_{i=1} \sum_{j=1} A_{i,j}^2} \quad (7)$$

We can see that the frobenius norm of the weight matrix A is simply the square root of the squared sum of all of its elements.

Note that the squared sum of the elements in A is not equal to the scaled squared sum of the convolution kernel filter which is employed in L1 [20, 36]. However, we can notice from the example shown in Figure 3 that, while calculating the squared sum of A, more central elements of the convolution kernel get more weighted. Concretely, for the weight matrix shown in Figure 3, we can write the square of its frobenius norm as:

$$\begin{aligned} \|A_{fig}\|_F^2 = 9k_{22}^2 + 6k_{23}^2 + 6k_{21}^2 + 6k_{32}^2 + 6k_{12}^2 \\ +4k_{13}^2 + 4k_{31}^2 + 4k_{33}^2 + 4k_{11}^2 \end{aligned} \quad (8)$$

, whereas the kernel squared sum metric employed in L1 and SFP [20, 36] can be written as:

$$\|K\|^2 = \sum_{i=1}^{3} \sum_{j=1}^{3} k_{ij}^2 \neq \frac{1}{C} \|A_{fig}\|_F^2, \forall C \in \mathbb{R} \quad (9)$$

We can note that the more central elements of the convolution kernel are considered more important while evaluating the frobenius norm of the transformed matrix, addressing the issue identified in Sec. 3.2.

3.4.2 Spectral Norm

Besides the Frobenious norm, we also explore the other norm choices on the transformed convolution matrix. Formally, spectral norm, or operator norm, is defined as:

$$\|A\|_{op} = \sqrt{\sigma_{max}(A^H A)} = \sqrt{\sigma_{max}(A^T A)} = \sigma_{max}(A) \quad (10)$$

, which is simply the largest singular value of the weight matrix A. It represents the magnitude of the most dominant component after decomposition of A. Another way to look at it is to consider the amount of change A brings to the input feature vector $I^{\hat{(i)}}$. Recall equation 4 and consider a perturbation $\triangle \hat{I}$ added on the input feature vectors $I^{\hat{(i)}}$ which also results a change in the output vector, $\vec{O_m^{(i)}}$:

$$\vec{O_m^{(i)}} + \triangle \vec{O_m^{(i)}} = A(I^{\hat{(}i)} + \triangle \hat{I}) \quad (11)$$

$$\triangle \vec{O_m^{(i)}} = A \triangle \hat{I} \quad (12)$$

$$\|\triangle \vec{O_m^{(i)}}\|_2^2 = \|A \triangle \hat{I}\|_2^2 \leq \|A\|_{op}^2 \|\triangle \hat{I}\|^2 \quad (13)$$

Thus, we can see that the larger the spectral norm of the matrix, the larger the output can change as input changes. Intuitively, $\|A\|_{op}$ implies the **sensitivity** of the convolutional layer. By keeping the most sensitive channels that have the highest spectral norm of their corresponding convolution matrix, we keep the model at its highest potential.

3.4.3 Nuclear Norm

Another intuitive choice of pruning metric is the nuclear norm of the weight matrix A, which is formally defined as:

$$\|A\|_* = trace(\sqrt{A^* A}) = \sum_{k=1}^{r} \sigma_k(A) \quad (14)$$

$\|A\|_*$ can be also viewed as a convex envelope of the rank function $rank(A)$. We utilize it to estimate the magnitude and importance of the output feature vector $\vec{O_m^{(i)}}$.

3.5. Pruning Procedure

We described our pruning method as a four-step procedure. For each layer: first, we calculated $importance(W_j^{(i)})$ for the j^{th} output channel by evaluating the norm of its transformed matrix; next, we sort the output channels based on their importance scores; then, we keep $P^{(i)}$ channels by selecting ones with the highest importance scores; finally, we make additional removal in the BN and next convolutional layer to match the shape of the output of the pruned channel.

Notably, we performed **all of our experiments in a single-shot manner**. We pruned channels at all the layers at once and only perform a single retraining session. Algorithmic description of SPSRC can be found in Algorithm 1.

4. Experiments

We conduct extensive experiments to validate SPSRC. We compare with many other compression methods, including both property-based and learning-based. We trained with various pruning configurations to demonstrate SPSRC's superiority with different compression ratios. We refer to them as the **first**, **second**, or **third** pruning configuration in later discussion based on the compression ratios and the order they appear in the result tables(Table 1, 2, 3, 4, and 5) for different models. Moreover, through ablation studies, we show that SPSRC is indeed an effective channel pruning method to reduce the model size, keeping the most essential part of the model. We also include direct comparisons with L1 [36] in all experiments to demonstrate that a simple transformation by SPSRC is effective and provides much better channel importance measurement than naively employing the convolution kernels.

4.1. Benchmarks and Evaluation Protocols

We demonstrated our result on CIFAR-10 [29], CIFAR-100, and ImageNet [50], showing SPSRC 's superiority on both small and large dataset. We studied the performance on different network architectures, including VGG-16-bn [54], ResNet-56 [19], ResNet-34, ResNet-50, and

Algorithm 1 Proposed Algorithm: SPSRC

Input: `model`, `prune_layers`, `prune_ratios`, `fm_sizes`

Output: `new_model`

▷ `model` is the trained network with K layers, `prune_layers` represents the list of indices of layers we try to prune, and `prune_ratios` represents the list of ratio of removed channels at each layer, and `fm_sizes` represents the list of feature map sizes. `new_model` is the compressed and thinner model we return after pruning

```
new_model ← model
for i ← 1 to K do
    if i not in prune_layers then continue
    end if
    W ← model.convs[i].weight
    W' ← model.convs[i+1].weight
    N ← W.num_in_channels
    M ← W.num_out_channels
    scores ← [ ]                //Saliency Scores
    os, is ← fm_sizes[i + 1], fm_sizes[i]
    for m ← 1 to M do
        A ← Null
        for n ← 1 to N do
            A_{m,n} ← ToMat(W_{m,n}, os, is)
            A ← ConCat(A, A_{m,n})
        end for
        score ← norm(A)
        scores.append(score)
    end for
    indices ← argsort(scores)
    kept_ratio ← 1 − prune_ratios[i]
    kept_inds ← indices[: kept_ratio]
    new_model.convs[i] ← W[kept_inds, :]
    new_model.convs[i+1] ← W'[:, kept_inds]
end for
finetune(new_model)
return new_model
```

DenseNet-40 [25] to demonstrate SPSRC's success on network with plain structure, residual connections, and dense modules.

We showed Top-1% accuracy for CIFAR-10 and CIFAR-100 for the recovered accuracy. For ImageNet, we show both Top-1% and Top-5% accuracies. In terms of evaluation of compression, we adopted two widely used metrics: Float Points Operations (FLOPs) and the number of parameters.

We used PyTorch to implement our method. For all of the experiments, we used Stochastic Gradient Descent(SGD) for optimization with weight decay of 0.0005 and momentum of 0.9.

Model	Top-1%↑	FLOPs(M)↓	Params(M)↓
Dense	93.51	321.35	14.73
L1 [36]	93.40	211.62	5.26
Zhao et al [65]	93.18	195.73	3.94
Spectral(ours)	93.78±0.05	211.62	5.26
Nuclear(ours)	93.81±0.03	211.62	5.26
Frobenius (ours)	**93.82**±0.07	211.62	5.26
EnergyAware [60]	93.48	107.33	2.81
HRank [37]	92.34	111.51	2.64
SSS [26]	93.02	187.05	3.87
GAL - 0.05 [39]	92.03	194.13	3.31
GAL - 0.1 [39]	90.73	176.13	2.63
Spectral (ours)	**93.90**±0.05	186.83	2.77
Nuclear (ours)	93.77±0.09	186.83	2.77
Frobenius (ours)	93.74±0.11	186.83	2.77

Table 1. Results on CIFAR-10 with VGG16-BN. We separated the table with dashline based on compression ratios.

4.2. Results on CIFAR-10

All of the following experiments conducted on CIFAR-10 are on a single RTX 2080 Ti GPU. We trained all the baseline(unpruned) networks from scratch with SGD optimizer and `batch_size` set to 128. For VGG16-bn, we trained a total of 164 epochs and an initial learning rate of 0.1, decaying by 10 at 81^{th} and 122^{th} epoch. For ResNet-56, we trained a total of 200 epochs and an initial learning rate of 0.1, decaying by 10 at the 60^{th}, 120^{th}, and 160^{th} epoch. For DenseNet-40, we trained a total of 350 epochs and an initial learning rate of 0.1, decaying by 10 at the 150^{th} and 250^{th} epoch. For finetuning after pruning, we used the same optimization setting as the baseline network training but with initial learning rate set to 0.001, decaying by 10 at the 20^{th} epoch.

VGG-16 In terms of pruning configuration, we pruned on exactly the same set of layers as L1 [36] for a fair comparison but with a larger pruning ratio. In Table 1, we demonstrated results of two different pruning configurations, retrained for 20 and 70 epochs respectively after pruning. Specifically, we observed that the model pruned with spectral norm achieves a recovered accuracy of 93.9, even higher than the baseline accuracy by nearly 0.4%, with a large reduction in the number of parameters(81.33%) and FLOPs(41.86%). This is a significant improvement compared to 93.02 [26], 92.03 [39], and 90.73 [39], pruned with similar or even fewer reduction ratios in parameters.

ResNet-56 In terms of pruning configuration, same as [20, 36], we do not consider pruning of the projection shortcuts for simplification. In Table 2, we demonstrated results of three different pruning configurations, retrained for 50, 80, and 80 epochs respectively after pruning. Table 2 shows that SPSRC is higher in accuracy than other methods with similar compression ratios with all of the three pruning configurations. Specifically, as we pruned out more parameters, the superiority of SPSRC becomes more significant. Compared with He *et al*. [22], we achieved 63.95% FLOPs

Model	Top-1%↑	FLOPs(M)↓	Params(M)↓
Dense	93.59	129.32	0.85
HRank [37]	**93.52**	91.43	0.72
L1 [36]	93.06	93.63	0.73
Spectral(ours)	93.4±0.01	89.22	0.73
Nuclear(ours)	**93.52**±0.01	89.22	0.73
Frobenius (ours)	93.46±0.01	89.22	0.73
HRank [37]	93.17	64.66	0.49
NISP [62]	93.01	83.67	0.49
GAL - 0.6 [39]	92.98	80.70	0.75
Spectral (ours)	92.82±0.02	73.66	0.47
Nuclear (ours)	**93.19**±0.01	73.66	0.47
Frobenius (ours)	92.89±0.03	73.66	0.47
HRank [37]	90.72	33.50	0.27
He *et al.* [22]	90.8	63.88	-
GAL - 0.8 [39]	90.36	51.47	0.29
Spectral (ours)	91.55±0.01	46.62	0.3
Nuclear (ours)	**91.65**±0.01	46.62	0.3
Frobenius (ours)	91.62±0.01	46.62	0.3

Table 2. Results on CIFAR-10 with ResNet56. We separated the table with dashline based on compression ratios.

Model	Top-1%↑	FLOPs(M)↓	Params(M)↓
Dense	94.82	282.04	1.04
EnergyAware [60]	94.62	167.41	0.66
HRank [37]	94.24	167.41	0.66
GAL - 0.01 [39]	94.29	182.92	0.67
Spectral(ours)	94.64±0.04	168.82	0.59
Nuclear(ours)	94.53±0.10	168.82	0.59
Frobenius(ours)	**94.69**±0.07	168.82	0.59

Table 3. Results on CIFAR-10 with DenseNet40. We separated the table with dashline based on compression ratios.

Model	Top-1%↑	FLOPs(M)↓	Params(M)↓
Dense	73.33	321.40	14.77
L1 [36]	72.83	211.64	5.29
Spectral(ours)	**73.16**±0.03	211.64	5.29
Nuclear(ours)	72.83±0.04	211.64	5.29
Frobenius(ours)	72.69±0.08	211.64	5.29
L1 [36]	71.85	186.85	2.77
Spectral(ours)	**71.89**±0.02	186.85	2.77
Nuclear(ours)	71.85±0.07	186.85	2.77
Frobenius(ours)	71.74±0.05	186.85	2.77

Table 4. Results on CIFAR-100 with VGG16-BN. We separated the table with dashline based on compression ratios.

reduction(v.s. 50.6%) with recovered accuracy 91.65(v.s. 90.8). Compared with GAL [39], we achieved 63.95% FLOPs reduction(v.s. 60.2%) and 64.71% parameters reduction(v.s. 65.9%) with recovered accuracy 91.65(v.s. 90.36).

DenseNet-40 In Table 3, we demonstrated result of SPSRC on DenseNet-40. We finetuned the model for 40 epochs after pruning. As shown in Table 3, SPSRC is higher in accuracy than all other competitive methods with similar or smaller compression ratios.

Model	Top-1%↑	Top-5%↑	FLOPs(G)↓	Params(M)↓
Dense	73.27	91.43	3.76	21.88
L1 [36]	72.17	90.89	2.82	19.47
LCCN [9]	72.99	91.19	2.83	-
Taylor [47]	72.83	-	2.93	17.26
Spectral(ours)	72.94±0.10	91.12	2.82	19.47
Nuclear(ours)	**73.18**±0.05	**91.33**	2.82	19.47
Frobenius(ours)	73.04±0.11	91.23	2.82	19.47
SFP [20]	71.83	90.33	2.22	-
Spectral(ours)	71.98±0.11	90.57	2.30	17.88
Nuclear(ours)	**72.12**±0.05	90.56	2.30	17.88
Frobenius(ours)	72.02±0.08	**90.59**	2.30	17.88

Table 5. Results on ImageNet with ResNet34. We separated the table with dashline based on compression ratios.

Method	Top-1%↑	FLOPs(G)↓	FPS(im/s)↑
Dense [34]	76.2	4.1	1019
EagleEye-1G [34]	74.2	1.0	2429
GReg-2 [58]	73.9	1.3	1514
DSNet [35]	74.6	1.2	–
Polarize [66]	74.2	1.2	–
MetaPruning [43]	73.4	1.0	2381
DMCP [14]	74.1	1.1	–
HALP-30% [53]	74.3	1.0	2755
HALP-SPSRC(Ours)	**75.0**±0.1	1.1	**3007**

Table 6. Results on ImageNet with ResNet50 for hardware-aware pruning. We instantiated our method SPSRC on top of HALP.

4.3. Results on CIFAR-100

All of the following experiments conducted on CIFAR-100 are on a single RTX 2080 Ti GPU.

VGG16 Training and optimization setting of the baseline network is the same as that of VGG16-bn for CIFAR10.

In Table 4, we demonstrated results of two different pruning configurations retrained for 50 and 70 epochs respectively after pruning. Table 4 demonstrates that, with both pruning configurations, SPSRC is higher in accuracy than other methods with similar compression ratios.

4.4. ImageNet

All of the following experiments conducted on ImageNet are on eight RTX 2080 Ti GPUs with parallel training.

ResNet-34 For a fair comparison, we performed pruning on the official torchvision version model(73.27% Top-1 accuracy). We also tried the model trained from scratch and observed similar accuracy. We followed the same optimization setting as PyTorch official example, with batch size 256, momentum 0.9, and weight decay 0.0001. For retraining, we used the same optimization setting as the baseline model training but with the initial learning rate set to 0.001, decaying by 10 every 10 epochs. We propose two pruning configurations for ResNet-34 on ImageNet. For both configurations, we retrained the model for 30 epochs and observed that the model achieved the best accuracy around the 23^{th} epoch. Table 5 demonstrates that, with both pruning configurations, SPSRC is higher in accuracy than other methods with a similar compression ratio.

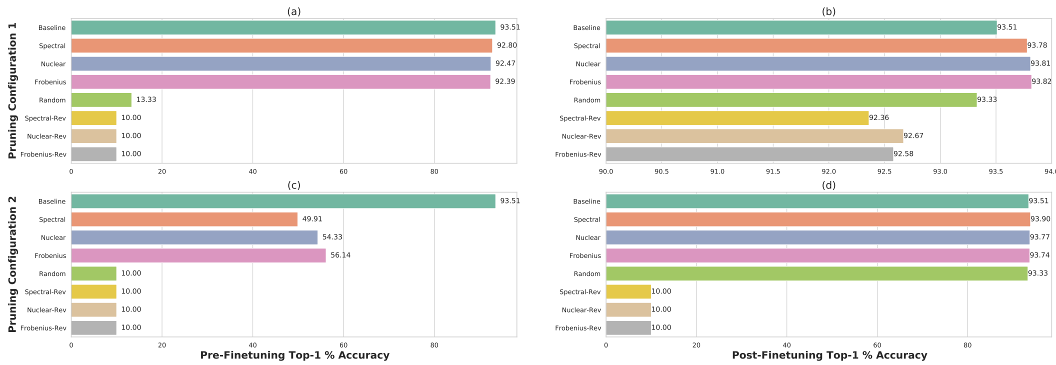

Figure 4. Ablation studies results. We demonstrated both pre-finetuning and post-finetuning accuracies on models pruned with both configurations. Check 4.2 and Table 1 for detail of pruning configurations.

4.4.1 ResNet50 & Hardware-Aware Pruning

Here, we demonstrate that our framework can also be ***combined with the latest state-of-the-art*** hardware-aware pruning framework HALP [53] to yield better results. Notably, HALP leveraged Taylor importance score [47], which is formulated as a product of gradient and magnitude of kernel weights. While instantiating SPSRC, we apply the reorganization to the final product which also considers gradient info. Results can be found in the Table 6. We improve HALP obviously with SPSRC, surpassing its Top-1 by 0.7 and FPS by around 300 IM/S.

4.5. Ablation Studies

We performed ablation studies on CIFAR-10 with VGG16. We showed the results in Figure 4. We demonstrated results of random pruning and pruning with the opposite of our metric(labeled with "-Rev" postfix in Figure 4) to show the effectiveness of SPSRC. For the opposite of our metric, filters with larger spectral, nuclear, or frobenius norm after reorganization are removed first. We showed results for both the first and second pruning configuration with same compression settings as described in 4.2 and Table 1.

With 34.15% FLOPs reduction and 64.29% parameters reduction, as shown in sub-figure (a) of Figure 4, SPSRC still retains decent accuracy (around 92.5%) even before finetuning. However, for random pruning and pruning with reverse metric, the accuracy drops sharply to 13.33% and 10%. The appearance of exactly 10.00% Top-1 accuracy(expected random guess accuracy on CIFAR-10) is because the network consistently produces **zero** due to insignificant weights, making the network predict one class regardless of the input. This phenomenon can also be seen in other sub-figures. After finetuning, we can see from sub-figure (b) that SPSRC even surpasses the baseline accuracy. Random pruning, though performing worse than SPSRC and the baseline, is better than pruning with the reverse metric.

With 41.86% FLOPs reduction and 81.33% parameters reduction, as shown in sub-figure (c) of Figure 4, accuracy of SPSRC drops to around 54% before finetuning. However, for pruning randomly and pruning with the reverse metric, the accuracy drops to 10%. Interestingly, we observe that, even with finetuning, the network pruned with the reverse metric can not improve as shown in sub-figure (d). The gradients are negligible, causing the network to be stuck at a very bad local optimum and consistently predicting one class regardless of the input. We even elongated retraining to 80 epochs and gradually increase the learning rate to 1 but still did not observe any improvement on test accuracy for reverse SPSRC. As expected, the model pruned randomly can recover with randomly kept important channels but performed worse than the baseline and SPSRC.

5. Conclusions

In this paper, we present a novel channel pruning framework called SPSRC, which employs the norm of the weight matrices after reorganization of convolution to matrix multiplication. We identify intrinsic and everlasting issues in previous methods which directly extract information from the plain convolution kernels like lack of convolution kernel spatial saliency information. We both theoretically and empirically demonstrate why this reorganization step helps us obtain better importance measurement of each channel. We conducted extensive experiments to show the efficiency and superiority of SPSRC compared to the other channel pruning methods. We hope that our work provide a new direction for pruning works in the future. In the following works, we will analyze the reorganized matrices from convolution kernels more deeply and conduct other types of compression, including tensor decomposition, after a more thorough theoretical analysis.

References

[1] Ali Alqahtani, Xianghua Xie, Mark W Jones, and Ehab Essa. Pruning cnn filters via quantifying the importance of deep visual representations. *Computer Vision and Image Understanding*, 208:103220, 2021. 1

[2] Davis W. Blalock, Jose Javier Gonzalez Ortiz, Jonathan Frankle, and John V. Guttag. What is the state of neural network pruning? In *Proceedings of the 3rd MLSys Conference*, 2020. 1

[3] Jierun Chen, Shiu-hong Kao, Hao He, Weipeng Zhuo, Song Wen, Chul-Ho Lee, and S-H Gary Chan. Run, don't walk: Chasing higher flops for faster neural networks. In *CVPR*, 2023. 4

[4] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 40(4):834–848, 2017. 1

[5] Wenlin Chen, James Wilson, Stephen Tyree, Kilian Weinberger, and Yixin Chen. Compressing neural networks with the hashing trick. In *International Conference on Machine Learning(ICML)*, pages 2285–2294. PMLR, 2015. 1

[6] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 1

[7] L. Deng, G. Li, S. Han, L. Shi, and Y. Xie. Model compression and hardware acceleration for neural networks: A comprehensive survey. *Proceedings of the IEEE*, 108(4):485–532, 2020. 1

[8] Emily L. Denton, Wojciech Zaremba, Joan Bruna, Yann LeCun, and Rob Fergus. Exploiting linear structure within convolutional networks for efficient evaluation. In *Neural Information Processing Systems(NIPS)*, pages 1269–1277, 2014. 1

[9] Xuanyi Dong, Junshi Huang, Yi Yang, and Shuicheng Yan. More is less: A more complicated network with less inference complexity. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5840–5848, 2017. 7

[10] Gale et el. The state of sparsity in deep neural networks. *arXiv preprint arXiv:1902.09574*, 2019. 2

[11] Hou et el. Chex: channel exploration for cnn model compression. In *CVPR*, 2022. 2

[12] Lin et el. Channel pruning via automatic structure search. 2020. 1

[13] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations (ICLR)*, 2019. 1, 2

[14] Shaopeng Guo, Yujie Wang, Quanquan Li, and Junjie Yan. Dmcp: Differentiable markov channel pruning for neural networks. In *CVPR*, pages 1539–1547, 2020. 7

[15] S. Han, X. Liu, H. Mao, J. Pu, A. Pedram, M. A. Horowitz, and W. J. Dally. Eie: Efficient inference engine on compressed deep neural network. In *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*, pages 243–254, 2016. 1

[16] Song Han, Jeff Pool, John Tran, and William J. Dally. Learning both weights and connections for efficient neural networks. *Neural Information Processing Systems (NIPS)*, 2015. 1, 2

[17] Babak Hassibi, David G Stork, and Gregory J Wolff. Optimal brain surgeon and general network pruning. In *IEEE international conference on neural networks*, pages 293–299. IEEE, 1993. 1

[18] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017. 1

[19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 770–778, 2016. 2, 5

[20] Yang He, Guoliang Kang, Xuanyi Dong, Yanwei Fu, and Yi Yang. Soft filter pruning for accelerating deep convolutional neural networks. In *International Joint Conference on Artificial Intelligence(IJCAI)*, pages 2234–2240, 2018. 1, 2, 5, 6, 7

[21] Yang He, Ping Liu, Ziwei Wang, Zhilan Hu, and Yi Yang. Filter pruning via geometric median for deep convolutional neural networks acceleration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR*, pages 4340–4349, 2019. 2

[22] Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 1389–1397, 2017. 1, 3, 6, 7

[23] Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. *Neural Information Processing Systems (NIPS)*, 2014. 1

[24] Hengyuan Hu, Rui Peng, Yu-Wing Tai, and Chi-Keung Tang. Network trimming: A data-driven neuron pruning approach towards efficient deep architectures. *CoRR*, abs/1607.03250, 2016. 1, 2

[25] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017. 2, 6

[26] Zehao Huang and Naiyan Wang. Data-driven sparse structure selection for deep neural networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 304–320, 2018. 3, 6

[27] Ryan Humble, Maying Shen, Jorge Albericio Latorre, Eric Darve, and Jose Alvarez. Soft masking for cost-constrained channel pruning. In *European Conference on Computer Vision*, pages 641–657. Springer, 2022. 2

[28] Jianbo Jiao, Yunchao Wei, Zequn Jie, Honghui Shi, Rynson WH Lau, and Thomas S Huang. Geometry-aware distillation for indoor semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition(CVPR*, pages 2869–2878, 2019. 1

[29] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 2, 5

[30] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012. 1

[31] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 1

[32] Yann LeCun, John S Denker, Sara A Solla, Richard E Howard, and Lawrence D Jackel. Optimal brain damage. In *Neural Information Processing Systems(NIPs)*, volume 2, pages 598–605. Citeseer, 1989. 1

[33] Namhoon Lee, Thalaiyasingam Ajanthan, and Philip H. S. Torr. Snip: single-shot network pruning based on connection sensitivity. In *International Conference on Learning Representations(ICLR)-Poster*, 2019. 1

[34] Bailin Li, Bowen Wu, Jiang Su, and Guangrun Wang. Eagleeye: Fast sub-net evaluation for efficient neural network pruning. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*, pages 639–654. Springer, 2020. 2, 7

[35] Changlin Li, Guangrun Wang, Bing Wang, Xiaodan Liang, Zhihui Li, and Xiaojun Chang. Dynamic slimmable network. In *CVPR*, pages 8607–8617, 2021. 7

[36] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. In *International Conference on Learning Representations(ICLR)-Poster*, 2017. 1, 2, 4, 5, 6, 7

[37] Mingbao Lin, Rongrong Ji, Yan Wang, Yichen Zhang, Baochang Zhang, Yonghong Tian, and Ling Shao. Hrank: Filter pruning using high-rank feature map. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition(CVPR)*, pages 1529–1538, 2020. 1, 2, 6, 7

[38] Shaohui Lin, Rongrong Ji, Chao Chen, Dacheng Tao, and Jiebo Luo. Holistic cnn compression via low-rank decomposition with knowledge transfer. *IEEE transactions on pattern analysis and machine intelligence(TPAMI)*, 41(12):2889–2905, 2018. 1

[39] Shaohui Lin, Rongrong Ji, Chenqian Yan, Baochang Zhang, Liujuan Cao, Qixiang Ye, Feiyue Huang, and David Doermann. Towards optimal structured cnn pruning via generative adversarial learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR*, pages 2790–2799, 2019. 1, 3, 6, 7

[40] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision (ECCV)*, pages 21–37. Springer, 2016. 1

[41] Yifan Liu, Ke Chen, Chris Liu, Zengchang Qin, Zhenbo Luo, and Jingdong Wang. Structured knowledge distillation for semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition(CVPR)*, pages 2604–2613, 2019. 1

[42] Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. Learning efficient convolutional networks through network slimming. In *Proceedings of the IEEE International Conference on Computer Vision(ICCV)*, pages 2736–2744, 2017. 1, 3

[43] Zechun Liu, Haoyuan Mu, Xiangyu Zhang, Zichao Guo, Xin Yang, Kwang-Ting Cheng, and Jian Sun. Metapruning: Meta learning for automatic neural network channel pruning. In *ICCV*, pages 3296–3305, 2019. 7

[44] Zechun Liu, Baoyuan Wu, Wenhan Luo, Xin Yang, Wei Liu, and Kwang-Ting Cheng. Bi-real net: Enhancing the performance of 1-bit cnns with improved representational capability and advanced training algorithm. In *Proceedings of the European conference on computer vision (ECCV)*, pages 722–737, 2018. 1

[45] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015. 1

[46] Ekdeep Singh Lubana and Robert P. Dick. A gradient flow framework for analyzing network pruning. *CoRR*, abs/2009.11839, 2020. 2

[47] Pavlo Molchanov, Arun Mallya, Stephen Tyree, Iuri Frosio, and Jan Kautz. Importance estimation for neural network pruning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11264–11272, 2019. 1, 2, 4, 7, 8

[48] Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. Pruning convolutional neural networks for resource efficient inference. In *International Conference on Learning Representations(ICLR)-Poster*, 2017. 1, 2

[49] Jongsoo Park, Sheng R. Li, Wei Wen, Ping Tak Peter Tang, Hai Li, Yiran Chen, and Pradeep Dubey. Faster cnns with direct sparse convolutions and guided pruning. In *International Conference on Learning Representations(ICLR)-Poster*, 2017. 1

[50] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. 2, 5

[51] Maying Shen, Lei Mao, Joshua Chen, Justin Hsu, Xinglong Sun, Oliver Knieps, Carmen Maxim, and Jose M Alvarez. Hardware-aware latency pruning for real-time 3d object detection. In *2023 IEEE Intelligent Vehicles Symposium (IV)*, pages 1–6. IEEE, 2023. 1, 2

[52] Maying Shen, Hongxu Yin, Pavlo Molchanov, Lei Mao, Jianna Liu, and Jose M Alvarez. Halp: hardware-aware latency pruning. *arXiv preprint arXiv:2110.10811*, 2021. 1, 2, 4

[53] Maying Shen, Hongxu Yin, Pavlo Molchanov, Lei Mao, Jianna Liu, and Jose M Alvarez. Structural pruning via latency-saliency knapsack. *NeurIPS*, 2022. 1, 2, 7, 8

[54] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In Yoshua Bengio and Yann LeCun, editors, *International Conference on Learning Representations (ICLR)*, 2015. 2, 5

[55] Xinglong Sun. Pruning for better domain generalizability. *arXiv preprint arXiv:2306.13237*, 2023. 1

[56] Xinglong Sun, Ali Hassani, Zhangyang Wang, Gao Huang, and Humphrey Shi. Disparse: Disentangled sparsification for multitask model compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12382–12392, 2022. 1

[57] Xinglong Sun, Maying Shen, Hongxu Yin, Lei Mao, Pavlo Molchanov, and Jose M Alvarez. Towards dynamic sparsification by iterative prune-grow lookaheads. 2022. 1

[58] Huan Wang, Can Qin, Yulun Zhang, and Yun Fu. Neural pruning via growing regularization. In *ICLR*, 2021. 7

[59] Paul Wimmer, Jens Mehnert, and Alexandru Condurache. Interspace pruning: Using adaptive filter representations to improve training of sparse cnns. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12527–12537, 2022. 1

[60] Seul-Ki Yeom, Kyung-Hwan Shim, and Jee-Hyun Hwang. Toward compact deep neural networks via energy-aware pruning. *arXiv preprint arXiv:2103.10858*, 2021. 6, 7

[61] Haichao Yu, Haoxiang Li, Humphrey Shi, Thomas S Huang, and Gang Hua. Any-precision deep neural networks. In *AAAI*, 2021. 1

[62] Ruichi Yu, Ang Li, Chun-Fu Chen, Jui-Hsin Lai, Vlad I Morariu, Xintong Han, Mingfei Gao, Ching-Yung Lin, and Larry S Davis. Nisp: Pruning networks using neuron importance score propagation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9194–9203, 2018. 7

[63] Xin Yuan, Pedro Savarese, and Michael Maire. Growing efficient deep networks by structured continuous sparsification. *ICLR*, 2021. 2

[64] Xiangyu Zhang, Jianhua Zou, Xiang Ming, Kaiming He, and Jian Sun. Efficient and accurate approximations of nonlinear convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and pattern Recognition(ICCV)*, pages 1984–1992, 2015. 1

[65] Chenglong Zhao, Bingbing Ni, Jian Zhang, Qiwei Zhao, Wenjun Zhang, and Qi Tian. Variational convolutional neural network pruning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2780–2789, 2019. 6

[66] Tao Zhuang, Zhixuan Zhang, Yuheng Huang, Xiaoyi Zeng, Kai Shuang, and Xiang Li. Neuron-level structured pruning using polarization regularizer. *NeurIPS*, 33:9865–9877, 2020. 7