Pruning from Scratch via Shared Pruning Module and Nuclear norm-based Regularization

Donghyeon Lee†, Eunho Lee†, Youngbae Hwang
Dept. of Intelligent Systems and Robotics, Chungbuk National University
{jsleeg98,ehlee,ybhwang}@cbnu.ac.kr

Abstract

Most pruning methods focus on determining redundant channels from the pre-trained model. However, they overlook the cost of training large networks and the significance of selecting channels for effective reconfiguration. In this paper, we present a "pruning from scratch" framework that considers reconfiguration and expression capacity. Our Shared Pruning Module (SPM) handles a channel alignment problem in residual blocks for lossless reconfiguration after pruning. Moreover, we introduce nuclear norm-based regularization to preserve the representability of large networks during the pruning process. By combining it with MACs-based regularization, we achieve an efficient and powerful pruned network while compressing towards target MACs. The experimental results demonstrate that our method prunes redundant channels effectively to enhance representation capacity of the network. Our approach compresses ResNet50 on ImageNet without requiring additional resources, achieving a top-1 accuracy of 75.25% with only 41% of the original model's MACs. Code is available at https://github.com/jsleeg98/NuSPM.

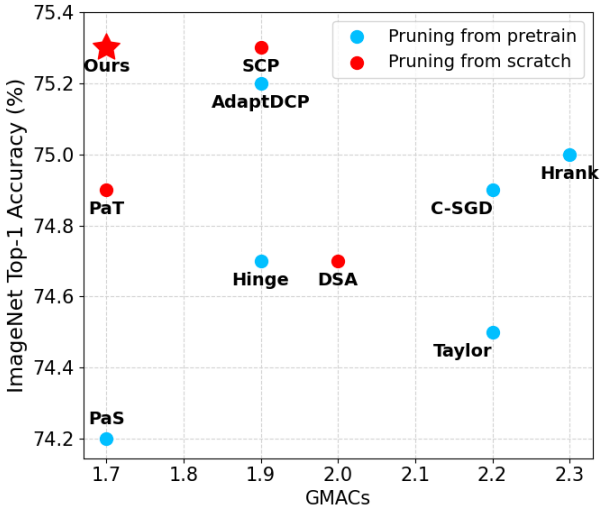

Figure 1. Comparison of Top-1 Accuracy according to MACs for compressed ResNet50 models on ImageNet. A blue color represents requiring a pre-trained network, while a red color represents not requiring one. Our approach achieves the best accuracy with minimal computational complexity compared to prior works without relying on pre-trained network.

1. Introduction

Convolutional neural networks (CNNs) have achieved remarkable success and are used in a wide range of AI applications. As CNNs offer high performance, their computational complexity also become higher, which can burden the deployment on edge devices. To mitigate these challenges, there are many methods to reduce the computational cost of a network, such as designing efficient networks [38, 41], knowledge distillation [12], network search [27, 49], quantization [46] and network pruning [5, 17, 19]. Among them, network pruning is one of effective methods for reducing the resource requirements of the network while minimizing performance degradation. To be specific, structured pruning reduces the inference time practically by removing entire redundant channels from layers.

Structured pruning can be categorized into two main approaches: importance-based and regularization-based methods. Importance-based methods [10, 19] use heuristic metrics, such as ℓ_1 norm or geometric median, to evaluate the significance of channels. On the other hand, regularization-based methods [28, 47] prune the network by incorporating regularization loss during the pruning process. These structured pruning methods usually follow a three-step pipeline. First, a large network is pre-trained until it reaches convergence and achieves a high performance on the target task. Second, the network is pruned to remove redundant channels based on specific pruning criteria or metrics. Finally, the pruned network is fine-tuned to recover from performance degradation caused by the pruning

† These authors contributed equally to this work.

process. This three-step pipeline has been widely used in structured pruning approaches and has shown promising results in reducing the computational cost and memory footprint of inference. Structured pruning methods primarily emphasize optimizing the last two processes to achieve efficient and effective network compression. However, training a large model until it converges is difficult and time-consuming, which reduces the overall efficiency. To deal with this inefficiency, we present a framework for pruning from scratch.

Pruning from scratch is compressing the network started from randomly initialized weights. In this framework, determining the significance of channels becomes challenging when using an importance-based approach. Instead, we adopt a regularization-based method, which gradually prunes the network over time. However, if the regularization is applied to network parameters directly, it has the potential loss of meaningful information and reduces network flexibility. To mitigate this issue, we use an indirect approach by applying regularization to the pruning module. This approach enables us to conduct recoverable network search during the pruning process.

Practical acceleration of a pruned network through structured pruning requires a reconfiguration process that actually removes masked channels. The widely-used residual blocks consist of convolutional layers and can be categorized into inner and outer layers based on their relationship to the shortcut connection. The inner layers can be reconfigured without any techniques because they operate independently. However, when the outer layers, which are related to shortcut connections, are reconfigured without considering channel alignment, the pruned network cannot maintain its performance after the reconfiguration process. In this paper, we introduce the Shared Pruning Module (SPM) as an approach for pruning the outer layers to achieve efficient network compression while preserving performance. By sharing the pruning module of the outer layers, we achieve more efficient network by properly pruning both inner and outer layers.

It is significant to guide the pruning module towards selecting appropriate channels. This should be done while maintaining the network's ability to effectively capture and express complex patterns and features, which we refer to as its *"representability"*. To preserve network representability while achieving the target compressed network, we introduce two types of losses: $\mathcal{L}_{mac}$ and $\mathcal{L}_{nuc}$. The $\mathcal{L}_{mac}$ loss is designed to guide the pruning modules towards achieving the target Multiply-Accumulate operations (MACs) for the pruned network [22]. On the other hand, $\mathcal{L}_{nuc}$ aims at maintaining the representability of the pruned network where the nuclear norm [40] measures the similarity in expression capacity between the pruned network and the large network. By applying both loss functions, we achieve a compressed network of the target ratio while maintaining the representability of the large network.

In our experiments, we show that our pruning method achieves an accuracy of 75.3% on the ImageNet dataset while utilizing only 41% of the original ResNet50 MACs. This result is obtained without requiring pre-training of a large network. Moreover, we prune ResNet56 by 51% of its original size while still achieving an accuracy of 93.5% on the CIFAR10 dataset. These results outperform prior works that utilize pre-trained networks despite pruning from scratch.

The contributions of this paper are highlighted as follows:

- The Shared Pruning Module (SPM) is introduced to address the channel alignment problem without increasing the network size during reconfiguration.
- A novel loss function that leverages the nuclear norm is proposed to preserve the expression capacity of pruned networks.
- Experiments show that the accuracy of 93.5% and 75.3% on CIFAR10 and ImageNet are achieved, respectively, without a pre-trained network.

2. Related Works

Network pruning methods have aimed to accelerate the inference of deep neural networks by eliminating redundant components, such as parameters or channels in the model. According to granularity, this method is categorized into unstructured pruning and structured pruning. Unstructured pruning methods individually identify and remove unimportant connections or weights from the network. Therefore, real acceleration on a general-purpose GPU requires specialized libraries. On the other hand, structured pruning method determines redundant elements as channels, enabling acceleration of inference time without requiring additional resources. Our approach applies structured pruning for practical acceleration.

2.1. Structured pruning

Structured pruning can be classified into two approaches based on the removal strategies: importance-based and regularization-based. Importance-based structured pruning uses various metrics to assess the significance of channels. For instance, Li *et al*. [19] prune channels based on the ℓ_1 norm of network's weight. However, channels having a large magnitude of weights does not always represent that they are important. Instead, He *et al*. [10] detect informative channels by assessing their correlation using the geometric median based on network's weight. In addition to using network's weights, the results of activation functions can serve as an accurate evaluation of importance. Lin *et*

al. [25] determine channel importance by its rank within the feature map. Alternatively, channels that receive fewer updates after backpropagation may be considered less important [34]. Furthermore, a Hessian matrix approximated by first-order derivatives can be utilized to assess channel importance [37]. However, these methods typically have limitations in that they determine importance from pre-trained networks.

Regularization-based structured pruning appends regularization loss during pruning to sparsify large networks. Liu *et al*. [28] use ℓ_1 regularization on the scaling factors in batch normalization for network pruning. As this approach can lead to overall weights being set to zero, causing performance degradation, Zhuang *et al*. [47] introduce "polarization" regularization to keep important weights intact.

There are several methods to indirectly apply regularization for better information preservation. Lin *et al*. [26] utilize detached soft mask from the network and apply ℓ_1 regularization to it. In another approach, channel pruning is achieved by employing regularization on an additional convolutional layer [22]. Xiao *et al*. [44] utilize ℓ_0 regularization on auxiliary parameters for network pruning. By using these indirect methods that involve regularization, it becomes possible to perform recoverable channel selection and to improve overall performance while mitigating the loss of valuable information in the original network.

2.2. Reconfiguration after pruning

Modern networks include residual blocks which have shortcut connections. These connections require an equal number of input and output feature maps, as well as equal indices. Given this requirement, careful selection of channels becomes crucial during reconfiguration after pruning. If channels in the outer layers are pruned without considering matching feature maps, applying a union operation can preserve network performance, but the network size is increased [16, 32]. To avoid this problem, several methods [3, 10, 24, 30, 31] focus on pruning only the inner layers of residual blocks. However, achieving an efficient network without pruning outer layers remains challenging. As alternatives to these methods, Wang *et al*. [42] compute the average importance of outer layers to ensure equal scoring, and Lin *et al*. [23] manually adjust the channels of outer layers. In contrast to these approaches, our proposed SPM automatically addresses this issue during the pruning and reconfigures without increasing network size.

2.3. Pruning from scratch

To alleviate the computationally demanding nature of pre-training over-parameterized networks, several methods have been introduced to enable pruning from scratch. Frankle and Carbin [4] involve training a neural network with random initialization for a short period. After this initial training, they prune the network to create a smaller sub-network. This pruned sub-network is then reinitialized with its original random weights and fine-tuned. In Lee *et al*. [18], important channels are determined by connection sensitivity, which is based on the impact of the network's output from the initial model. And then, they fine-tune the sub-network as regular training. Similarly, in Wang *et al*. [43], they prune a large network from scratch using a pruning method similar to Liu *et al*. [28]. After reaching the target network size, they train the sub-model. As another aspect, Shen *et al*. [39] determine when to prune based on sub-network similarity. If their sub-networks are similar, they prune the network and fine-tune during rest epochs. In these approaches, redundant channels are identified from an under-trained model allowing no opportunity for those pruned channels to update again. Consequently, this premature channel determination to prune can lead to misidentifying channels and result in performance degradation.

3. Methods

We introduce a framework for pruning from scratch that is free from pre-trained models. In particular, Shared pruning module (SPM) is presented in Sec. 3.1 to consider channel alignment problem for reconfiguration without increasing the network size. Then, we explain the proposed regularization in Sec. 3.2, aimed at achieving a model with desired computational cost while preserving the representability of the original network.

3.1. Shared Pruning Module

In a network with L convolutional layers, the filters of the l-th convolutional layer w^l can be represented by $\mathbb{R}^{C_{in} \times C_{out} \times k \times k}$, where k is the kernel size. C_{in} and C_{out} are the number of input channels and output channels, respectively. To reduce the size of the network, structured pruning removes the filters that have less impact on network performance. We apply the pruning module inspired by previous works [21, 22, 44, 45], as shown in Fig. 2 (c), to automatically determine the criterion as below:

$$o_c^l = (\mathbb{I}(a_c^l) \odot w_c^l) * o_c^{l-1} \tag{1}$$

where $\odot$ and $*$ are the element-wise multiplication and convolution operation, respectively. o_c^l represents the c-th output feature map of the l-th layer, w_c^l denotes the c-th filter of w^l. $\mathbb{I}$ is a indicator function as follows:

$$\mathbb{I}(a_c^l) = \begin{cases} 1, & \text{if } a_c^l > \tau \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

a_c^l is the pruning indicator for determining whether to remove the w_c^l. The value of a_c^l determined by the learning

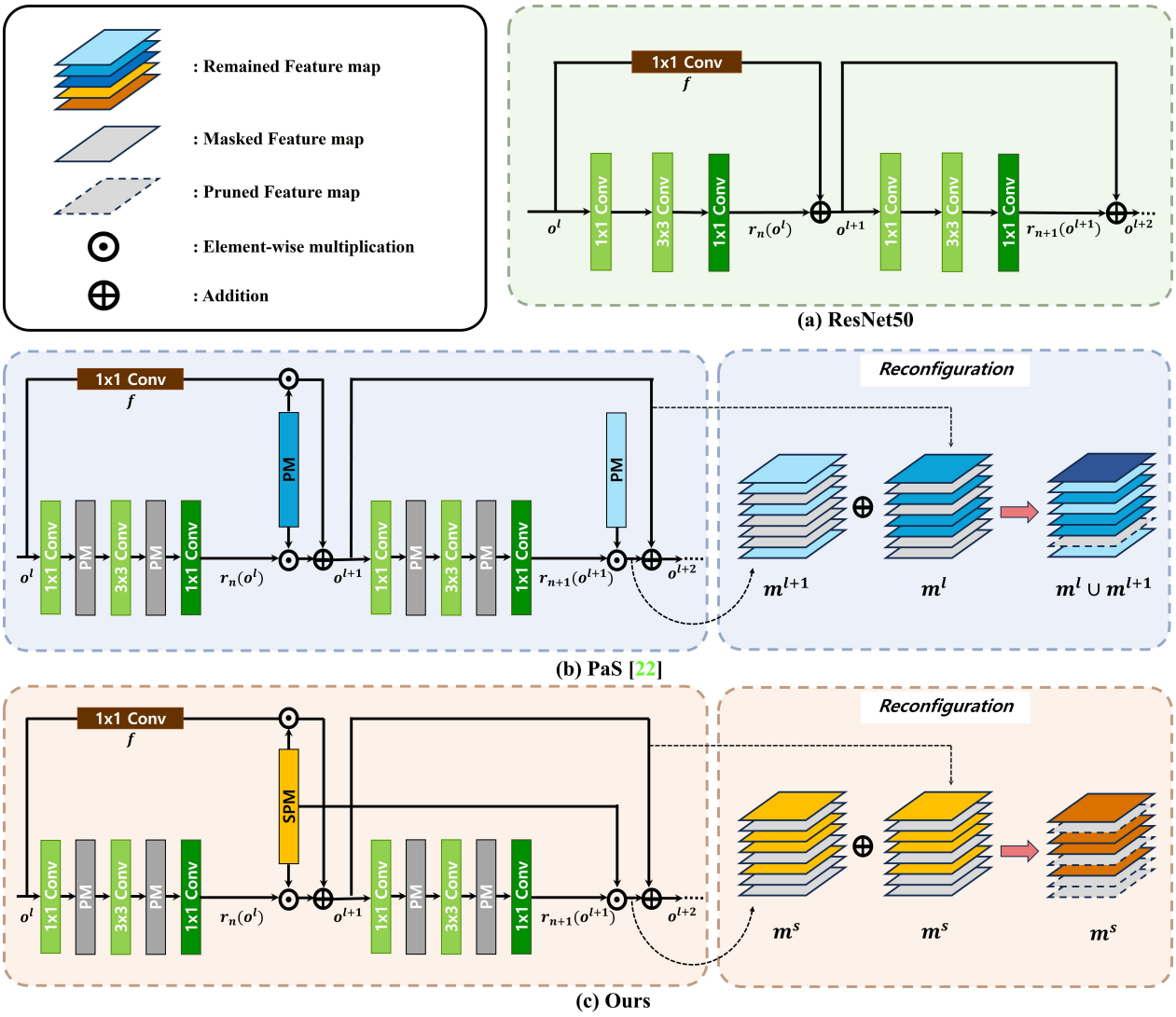

Figure 2. Illustratation of residual blocks applied with pruning methods and reconfiguration process. Light green and deep green convolutional layers represent the inner and outer layers, respectively. (a) This is a part of a residual block in ResNet50. (b) PaS [22] applies individual pruning modules (PM) to the outer layers, resulting in an increase in the non-masked channels of the feature map after reconfiguration. (c) Ours utilizes the Shared Pruning Module (SPM) to address the channel alignment problem, ensuring that the masked feature map remains unchanged.

process can be represented as a binary mask through the indicator function $\mathbb{I}$, which have a value of 1 when above the threshold τ and 0, otherwise. The mask is multiplied to the w_c^l, retaining the value of w_c^l if the corresponding mask value is 1, while pruning it if the mask value is 0.

Since Eq. (1) involves a binary operation, it is non-differentiable, which presents challenges for back-propagation. To solve this, straight through estimator (STE) is used for back-propagation as below:

$$\frac{\partial \mathcal{L}}{\partial a^l} = \frac{\partial \mathcal{L}}{\partial \mathbb{I}(a^l)} \tag{3}$$

where it allows for the direct passing of gradients from $\mathbb{I}(a^l)$ to a^l, resulting in the update of the binary convolutional layer. Therefore, the networks can be automatically pruned without relying on heuristic criteria.

Modern networks often include residual blocks to improve performance by mitigating the vanishing gradient problem. As shown in Fig. 2 (a), the residual blocks are typically stacked consecutively by:

$$o^{l+1} = f(o^l) + r_n(o^l). \tag{4}$$

The function f corresponds to a 1×1 convolutional layer used to match dimensions of output channels. Meanwhile,

$r(\cdot)$ represents bottleneck layers within the residual blocks, which comprise 1×1, 3×3, and 1×1 convolutional layers. o^l and o^{l+1} represent feature maps.

In Fig. 2 (b), the pruning modules are added individually after all convolutional layers including the outer layers. Additionally, there are stacked consecutive residual blocks. In the case of the first residual block, the shortcut connection operates as follows:

$$o^{l+1} = f(o^l) \odot m^l + r_n(o^l) \odot m^l. \tag{5}$$

The binary mask m^l is created by the pruning module's indicator. As m^l is applied to both the 1×1 convolutional layer f and the output of bottleneck layers, the addition operation is conducted using the same mask. Consequently, the pruned channels of output feature map are not changed after reconfiguration. However, it should be noted that the residual blocks following the first one differ from the initial block in this regard.

$$o^{l+2} = o^{l+1} \odot m^l + r_{n+1}(o^{l+1}) \odot m^{l+1}. \tag{6}$$

In the subsequent shortcut connection operation as shown in Eq. (6), the addition is performed using different binary masks, namely m^l and m^{l+1}. These masks are applied to the respective layers before the addition operation takes place. Since m^l and m^{l+1} are generated by different pruning indicators, they can have varying numbers of masked channels or even different indices of masked channels. We call this *the channel alignment problem*. This discrepancy in the binary masks can result in different sets of channels being retained or pruned for each layer, potentially leading to differences in the network's architecture and computation flow between the two layers involved in the addition operation. To reconfigure the network using these binary masks without any performance degradation, it should apply a union operation to combine them. This can be seen in the "Reconfiguration" part of Fig. 2 (b), where the binary masks m^l and m^{l+1} are merged using a union operation [16, 32]. It can remain all necessary channels, but it can increase network size as decrease masked channels of binary mask.

In this paper, we propose the Shared Pruning Module (SPM) as an approach to automatically address the challenge of channel alignment, as shown in Fig. 2 (c). The SPM is specifically designed to prune the outer layers, which are involved addition operations. By leveraging a shared pruning indicator, we ensure consistent pruning or holding of channels participating in these operations. This eliminates any mismatches or inconsistencies during the addition operation. Our approach based on the SPM provides an effective solution for automating channel alignment and achieving efficient reconfiguration in network architectures with shortcut connections.

$$o^{l+2} = o^{l+1} \odot m^s + r_{n+1}(o^{l+1}) \odot m^s. \tag{7}$$

In Eq. (7), the binary mask m^s is generated by the SPM. The pruned channels do not change after reconfiguration. Because the binary masks applied to the output feature maps are the same. This allows for channel removal while maintaining performance and without increasing network size.

3.2. Regularization for pruning from scratch

We introduce two regularization terms: $\mathcal{L}_{mac}$ and $\mathcal{L}_{nuc}$ for achieving the desired compression ratio and preserving the representability of the original network, respectively. To achieve the target ratio by reducing computational complexity of network, $\mathcal{L}_{mac}$ is calculated as follows:

$$\mathcal{L}_{mac} = \left| \sum_l C'_{out} \times C'_{in} \times \mathcal{F}_h \times \mathcal{F}_w \times k^2 - \zeta \right|^2 \tag{8}$$

where C'_{out} and C'_{in} denote the number of pruned input channels and output channels, respectively. $\mathcal{F}_h \times \mathcal{F}_w$ is the size of feature map, and ζ is the target MACs. It is defined as the squared ℓ_2 norm of the difference between current and target MACs. However, when determining which channels to retain through the pruning module, it is necessary to consider not only achieving the target ratio, but also improving performance.

Derived from the inherent nature of pruning, which can reduce the network's capacity and lead to performance degradation, we suggest maintaining network representability layer by layer. In mathematics, $rank$ can be used to express representability, as it refers to the dimension of the vector space that can be generated by a matrix. Using $rank$ directly in a loss term may be challenging [40] due to its limited range of expression as it can only be an integer value. Instead, we use the nuclear norm, which provides a more flexible and continuous representation compared to the discrete nature of $rank$, which is defined as the ℓ_1 norm of the singular values of a matrix. Nuclear norm regularization differs from L1 or L2 regularization in that it assesses the network's representation capacity rather than merely shrinking the weights.

The representability of the pruned layer can be obtained by its weight as shown below:

$$p^l = \mathbb{I}(a^l) \odot w^l \tag{9}$$

where p^l denotes the pruned weight of l-th convolutional layer.

$$I_p^l = ||p^l||_*, \tag{10}$$

$$I_o^l = ||w^l||_* \tag{11}$$

Methods	PT	MACs(M)	Top-1 acc.(%)
ThiNet [31]	Y	63.6	92.98
CP [11]	Y	63.6	92.80
DCP [48]	Y	63.6	93.49
AMC [9]	Y	63.6	91.90
SFP [7]	Y	63.6	93.35
Rethink [29]	N	63.6	93.07
PfS [43]	N	63.6	93.05
Ours	N	**63.2**	**93.50**

Table 1. Results of ResNet56 on CIFAR10 dataset. "PT" represents requiring pre-trained network, and "Y" and "N" means yes or no, respectively. "MACs" stands for the number of multiply and add operation for network, and less is better.

where $||\ ||_*$ denotes nuclear norm. I_p^l and I_o^l denote the representability of the pruned layer and the original layer, respectively.

$$\mathcal{L}_{nuc} = \sum_l \left| I_o^l - I_p^l \right| \tag{12}$$

where is defined as the sum of the ℓ_1 norm, which reflects the difference between the representability in the original and pruned layers. By using $\mathcal{L}_{nuc}$ regualrization, we improve the pruned network performance by minimizing representability with original network. The final regularization can be written as:

$$\mathcal{L}_{reg} = \alpha_{nuc}\mathcal{L}_{nuc} + \alpha_{mac}\mathcal{L}_{mac} \tag{13}$$

where α_{nuc} and α_{mac} are scale factors. We integrate regularization with the classification loss to form the final loss function. The final loss function is:

$$\min_{W,A} \mathcal{L}(W, A) + \mathcal{L}_{reg}. \tag{14}$$

W is the parameters of network and A is the pruning indicators. $\mathcal{L}(W, A)$ is a cross-entropy loss of the pruned network.

4. Experiments

We conduct experiments using ResNet [6] to show the effectiveness of our method for the classification task. We simply set a threshold of the pruning indicator, τ to 0.5. To demonstrate the applicability across datasets of varying sizes, we use CIFAR10 [15] and ImageNet ILSVRC 2012 datasets [1]. All experiments are executed on PyTorch framework using NVIDIA RTX A6000 GPUs. In order to demonstrate the efficiency of the Shared Pruning Module (SPM), we compare the MACs with PaS [22] after reconfiguration. We evaluate the impact of our nuclear norm-based regularization by examining accuracy across various values of α_{nuc} and analyzing the effect on the architecture of the pruned network.

Methods	PT	MACs(G)	Top-1 acc. (%)	Epochs
GAL [26]	Y	2.3	72.0	150
Hrank [25]	Y	2.3	75.0	570
SSS [13]	N	2.3	71.8	100
Taylor [33]	Y	2.2	74.5	-
C-SGD [2]	Y	2.2	74.9	-
DSA [35]	N	2.0	74.7	120
Hinge [20]	Y	1.9	74.7	-
AdaptDCP [48]	Y	1.9	75.2	210
SCP [14]	N	1.9	**75.3**	200
PaS [22]	Y	**1.7**	74.5	150
PaT [43]	N	**1.7**	74.9	**90**
Ours	N	**1.7**	**75.3**	**90**

Table 2. Results of ResNet50 on ImageNet dataset. "Epochs" represents the number of total epochs from randomly initialized weights to finetune after pruning. "PT" and "MACs" have the same meaning as stated in Tab. 1.

4.1. Results on CIRAR10

We experiment on the CIFAR10 dataset [15], which includes 50,000 training images and 10,000 test images across 10 classes. When performing pruning from a randomly initialized network on the CIFAR10 dataset, we use the SGD optimizer with a batch size of 128. The learning rate is warmed up linearly in the first 8 epochs, after which it follows a step scheduler with a division factor of 0.2 at epochs 60, 120, and 160. The network undergoes pruning and fine-tuning all together for a total of 200 epochs, which is consistent with the fine-tuning epochs used in other methods [8]. To be specific, we perform pruning for the initial 120 epochs, and then the network is fine-tuned for the remaining epochs.

Tab. 1 shows the top-1 accuracy of ResNet56 pruned by pre-training required and pruning from scratch methods on CIFAR10 dataset. Compared to the methods which requires a pre-trained model, the proposed method achieves better or similar performance with a smaller network size. In addition, our framework outperforms recent pruning from scratch methods by 0.43% in top-1 accuracy. These results demonstrate that our method can search the network more effectively and yield greater expression capacity than methods utilizing pre-trained networks. While other pruning from scratch methods may be limited in their expression capacity, our approach mitigates this limitation.

4.2. Results on ImageNet

ImageNet dataset [1] consists of 1.3M images across 1,000 classes. We employ the original training pipeline, utilizing PyTorch mixed-precision training based on DeepLearningExamples [36] with 90 epochs in total. An individual batch size of 128 is assigned to each GPU with using Distributed Data Parallel module. The learning rate is

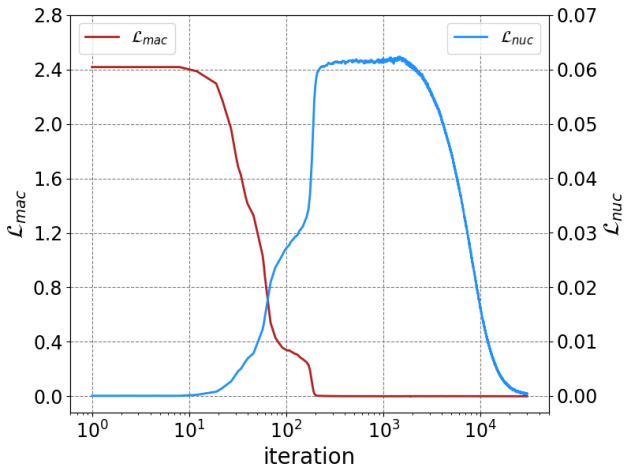

Figure 3. Loss change in pruning process. Red line denotes the MACs-based regularization loss $\mathcal{L}_{mac}$ and blue line is the nuclear norm-based regularization loss $\mathcal{L}_{nuc}$, respectively. Change of MACs-based and nuclear norm-based regularization.

initially warmed up linearly during the first 8 epochs, and then it follows a cosine decay throughout the entire process. We prune network using our proposed regularization loss for the first 60 epochs, followed by fine-tuning the pruned network. In Tab. 2, our pruned network achieves remarkable Top-1 accuracy while using fewer MACs and smaller epochs compared to other pruning methods. It is worthy note that the proposed method has fewer resources due to avoidance of pre-training the large network.

4.3. Ablation study

Regularization Effects. In this section, we demonstrate the impact of MACs and nuclear norm based regularization by showing the loss of each term during the pruning process in Fig. 3. Initially, the primary focus is on MACs-based loss function to achieve the target MACs ratio. As the pruned network approaches the target ratio, nuclear norm-based loss becomes more dominant. This is because the pruned network initially lacks representability compared to its original network. The observation highlights that MACs-based regularization has a significant impact at the initial steps, while nuclear norm-based regularization contributes to improving representability throughout the pruning process.

Effects of Nuclear Norm Regularization. We demonstrate the impact of $\mathcal{L}_{nuc}$ at various α_{nuc} values on the performance of the pruned network in Tab. 3. "Latency" is calculated as the average time required for processing one image, with a batch size of 1 and using only one GPU. The original network has 4.1G MACs. For fair comparison, we prune the network to achieve the same MACs for all networks by setting the same α_{mac}. As a result, the pruned networks

α_{nuc}	Params	MACs	Latency	Top-1 acc. (%)
0	16.3M	1.7G	8.34ms	74.4
0.0001	16.9M		8.50ms	74.5
0.0005	19.1M		8.74ms	74.9
0.001	19.2M		8.70ms	75.3

Table 3. Comparison of different α_{nuc} under fixed α_{mac} and MACs. α_{mac} and α_{nuc} represent the weights of MACs-based and nuclear norm-based regularizations, respectively. "Params" represents the number of parameters of pruned networks. α_{mac} is set to 0.5.

exhibit similar latency despite having different numbers of parameters. When α_{nuc} is set to zero, only MACs-based regularization is applied during pruning. As we increase α_{nuc}, the performance improves gradually because it can preserve more representability from the large network. This shows that the regularization based on the nuclear norm has a positive impact on the performance of the pruned network. Additionally, when evaluated on a CPU, it resulted in only about a 20% decrease in latency despite of 60% MACs reduction. This reduction in latency is not as much as the decrease in MACs, mainly because of constraints related to memory access time.

Fig. 4 provides an overview of the number of remaining output channels in convolutional layers after pruning. The green architecture represents the pruned network utilizing only MACs-based regularization, while the blue architecture incorporates nuclear norm-based regularization in addition. These two network architectures differ in terms of the number of parameters, yet they have a similar MACs size (1.7G), indicating comparable computational costs.

We can observe several features as follows. First, since we utilize our proposed SPM, the number of remaining channels in the outer layers of residual blocks is the same. Therefore, we enable lossless performance after reconfiguration without adding any computational complexity. Second, when we add nuclear norm-based regularization loss during pruning, pruned network has fewer channels in the initial layers of the network than without this regularization. Because the input feature maps of the initial layers are larger than those of the deeper layers, they involve higher computational complexity in terms of MACs, even though they contain fewer parameters. Hence, by retaining fewer channels in the initial layers, our method achieves an efficient network in terms of MACs. Third, when we incorporate nuclear norm-based regularization loss during pruning, it leads to fewer channels being pruned in the outer layers of the network's residual blocks. These outer layers are repeatedly employed in shortcut connections, making them potentially more informative than the inner layers. Therefore, our pruning method, which retains a greater number of channels in the outer layers, contributes to preserving the

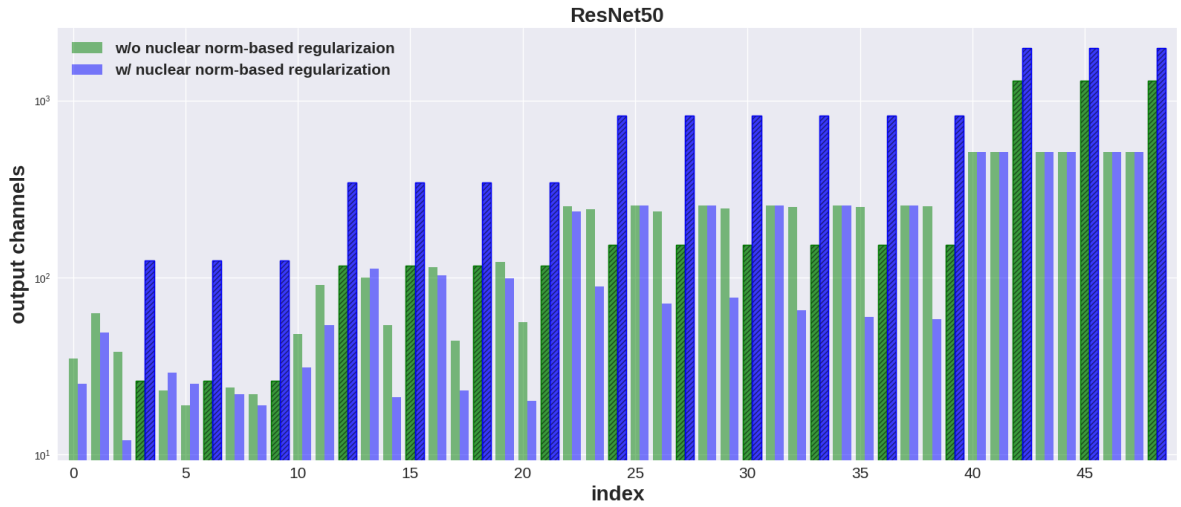

Figure 4. Channel pruning results for ResNet50. Green bars represents the number of remaining channels after applying MACs-based regularization alone, while blue bars represent the number of remaining channels after applying both MACs-based and nuclear norm-based regularization. Hatched bars indicate the remaining channels in the outer layers, while plain bars represent the remaining channels in the inner layers (excluding the first layer). The x-axis denotes the index of convolutional layers in ResNet50, and the y-axis represents the number of remaining output channels.

Methods	Before MACs	After MACs	Top-1 acc. (%)
PaS	1.7G	2.1G	74.5
PaS	1.4G	1.7G	74.1
Ours	1.7G	1.7G	75.3

Table 4. Comparison MACs with PaS: Before and After. Before MACs and After MACs indicates calculated MACs from remained channels in network and the MACs after reconfiguration for real acceleration, respectively.

network's representability.

Comparison of reconfiguration. To demonstrate the effectiveness of our reconfiguration process, we reproduce the pruning approach introduced in PaS [22]. Then, we calculate the network's MACs both before and after reconfiguration and compare the results with our approach as shown in Tab. 4. Because PaS individually applies the pruning module to all convolution layers including the outer layers of residual blocks, more channels of the outer layers should be retained to preserve information [16, 32]. Therefore, the MACs of PaS increase after reconfiguration to address the channel alignment problem. However, our method not only preserves MACs after reconfiguration due to the SPM, but also achieves higher accuracy compared to networks of the same size before reconfiguration. In case of the same MACs after reconfiguration, the performance gap is larger up to 1.2%. This method has more practical computational complexity for pruning network architecture by the SPM-based reconfiguration scheme.

5. Conclusion

In this paper, we proposed a new framework for pruning from scratch by considering the reconfiguration and enhancing the expression capacity. We introduced the Shared Pruning Module (SPM) to automatically address the channel alignment problem that arises with outer layers. It keeps the number of outer channels the same, enabling the reconfiguration without increasing the model size. Additionally, we presented MACs and nuclear norm based regularization terms to achieve target MACs and enhance the representability of the pruned network, respectively. Experiments on CIFAR10 and ImageNet showed that our framework achieved better performance at smaller model size than previous methods that require pre-training or perform pruning from scratch. It demonstrated that the proposed loss function can effectively reduce the model complexity while maintaining the performance. In addition, we validated that our framework can reconfigure the model without increasing the model size. In the future work, the generalizability of our framework is investigated by applying to other learning tasks instead of image classification.

Acknowledgement

This work is supported by Institute of Information & communications Technology Planning & Evaluation (IITP). (No. 2020-0-01077 and No. 2021-0-02068)

References

[1] 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. 6

[2] Xiaohan Ding, Guiguang Ding, Yuchen Guo, and Jungong Han. Centripetal sgd for pruning very deep convolutional networks with complicated structure. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4943–4953, 2019. 6

[3] Xiaohan Ding, Guiguang Ding, Yuchen Guo, Jungong Han, and Chenggang Yan. Approximated oracle filter pruning for destructive cnn width optimization. In *International Conference on Machine Learning*, pages 1607–1616. PMLR, 2019. 3

[4] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. 3

[5] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015. 1

[6] 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*, pages 770–778, 2016. 6

[7] Yang He, Guoliang Kang, Xuanyi Dong, Yanwei Fu, and Yi Yang. Soft filter pruning for accelerating deep convolutional neural networks. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*. International Joint Conferences on Artificial Intelligence Organization, 2018. 6

[8] Yang He, Guoliang Kang, Xuanyi Dong, Yanwei Fu, and Yi Yang. Soft filter pruning for accelerating deep convolutional neural networks. In Jérôme Lang, editor, *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*, pages 2234–2240. ijcai.org, 2018. 6

[9] Yihui He, Ji Lin, Zhijian Liu, Hanrui Wang, Li-Jia Li, and Song Han. Amc: Automl for model compression and acceleration on mobile devices. In *Proceedings of the European conference on computer vision (ECCV)*, pages 784–800, 2018. 6

[10] 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*, pages 4340–4349, 2019. 1, 2, 3

[11] 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*, pages 1389–1397, 2017. 6

[12] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. 1

[13] 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. 6

[14] Minsoo Kang and Bohyung Han. Operation-aware soft channel pruning using differentiable masks. In *International Conference on Machine Learning*, pages 5122–5131. PMLR, 2020. 6

[15] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images, 2009. 6

[16] Donghyeon Lee, Eunho Lee, and Youngbae Hwang. Lossless reconstruction of convolutional neural network for channel-based network pruning. *Sensors*, 23(4):2102, 2023. 3, 5, 8

[17] Eunho Lee and Youngbae Hwang. Layer-wise network compression using gaussian mixture model. *Electronics*, 10(1):72, 2021. 1

[18] Namhoon Lee, Thalaiyasingam Ajanthan, and Philip H. S. Torr. Snip: single-shot network pruning based on connection sensitivity. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. 3

[19] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. 1, 2

[20] Yawei Li, Shuhang Gu, Christoph Mayer, Luc Van Gool, and Radu Timofte. Group sparsity: The hinge between filter pruning and decomposition for network compression. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8018–8027, 2020. 6

[21] Yawei Li, Shuhang Gu, Kai Zhang, Luc Van Gool, and Radu Timofte. Dhp: Differentiable meta pruning via hypernetworks. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part VIII 16*, pages 608–624. Springer, 2020. 3

[22] Yanyu Li, Pu Zhao, Geng Yuan, Xue Lin, Yanzhi Wang, and Xin Chen. Pruning-as-search: Efficient neural architecture search via channel pruning and structural reparameterization. In Lud De Raedt, editor, *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, pages 3236–3242. International Joint Conferences on Artificial Intelligence Organization, 7 2022. Main Track. 2, 3, 4, 6, 8

[23] Mingbao Lin, Liujuan Cao, Yuxin Zhang, Ling Shao, Chia-Wen Lin, and Rongrong Ji. Pruning networks with cross-layer ranking & k-reciprocal nearest filters. *IEEE transactions on neural networks and learning systems*, 2022. 3

[24] Mingbao Lin, Rongrong Ji, Shaojie Li, Yan Wang, Yongjian Wu, Feiyue Huang, and Qixiang Ye. Network pruning using adaptive exemplar filters. *IEEE Transactions on Neural Networks and Learning Systems*, 33(12):7357–7366, 2021. 3

[25] 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, pages 1529–1538, 2020. 3, 6

[26] 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*, pages 2790–2799, 2019. 3, 6

[27] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*, 2018. 1

[28] 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*, pages 2736–2744, 2017. 1, 3

[29] Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. Rethinking the value of network pruning. In *International Conference on Learning Representations (ICLR)*, 2019. 6

[30] Jian-Hao Luo and Jianxin Wu. Autopruner: An end-to-end trainable filter pruning method for efficient deep model inference. *Pattern Recognition*, 107:107461, 2020. 3

[31] Jian-Hao Luo, Jianxin Wu, and Weiyao Lin. Thinet: A filter level pruning method for deep neural network compression. In *Proceedings of the IEEE international conference on computer vision*, pages 5058–5066, 2017. 3, 6

[32] Sangkug Lym, Esha Choukse, Siavash Zangeneh, Wei Wen, Sujay Sanghavi, and Mattan Erez. Prunetrain: fast neural network training by dynamic sparse model reconfiguration. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–13, 2019. 3, 5, 8

[33] 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*, pages 11264–11272, 2019. 6

[34] Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. Pruning convolutional neural networks for resource efficient inference. *arXiv preprint arXiv:1611.06440*, 2016. 3

[35] Xuefei Ning, Tianchen Zhao, Wenshuo Li, Peng Lei, Yu Wang, and Huazhong Yang. Dsa: More efficient budgeted pruning via differentiable sparsity allocation. In *European Conference on Computer Vision*, pages 592–607. Springer, 2020. 6

[36] NVIDIA. Deeplearningexamples, 2023. https://github.com/NVIDIA/DeepLearningExamples. 6

[37] Hanyu Peng, Jiaxiang Wu, Shifeng Chen, and Junzhou Huang. Collaborative channel pruning for deep networks. In *International Conference on Machine Learning*, pages 5113–5122. PMLR, 2019. 3

[38] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018. 1

[39] Maying Shen, Pavlo Molchanov, Hongxu Yin, and Jose M Alvarez. When to prune? a policy towards early structural pruning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12247–12256, 2022. 3

[40] Yang Sui, Miao Yin, Yi Xie, Huy Phan, Saman Aliari Zonouz, and Bo Yuan. Chip: Channel independence-based pruning for compact neural networks. *Advances in Neural Information Processing Systems*, 34:24604–24616, 2021. 2, 5

[41] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR, 2019. 1

[42] Wenxiao Wang, Cong Fu, Jishun Guo, Deng Cai, and Xiaofei He. Cop: Customized deep model compression via regularized correlation-based filter-level pruning. *arXiv preprint arXiv:1906.10337*, 2019. 3

[43] Yulong Wang, Xiaolu Zhang, Lingxi Xie, Jun Zhou, Hang Su, Bo Zhang, and Xiaolin Hu. Pruning from scratch. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 12273–12280, 2020. 3, 6

[44] Xia Xiao, Zigeng Wang, and Sanguthevar Rajasekaran. Autoprune: Automatic network pruning by regularizing auxiliary parameters. *Advances in neural information processing systems*, 32, 2019. 3

[45] Changdi Yang, Pu Zhao, Yanyu Li, Wei Niu, Jiexiong Guan, Hao Tang, Minghai Qin, Bin Ren, Xue Lin, and Yanzhi Wang. Pruning parameterization with bi-level optimization for efficient semantic segmentation on the edge. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15402–15412, 2023. 3

[46] Jiwei Yang, Xu Shen, Jun Xing, Xinmei Tian, Houqiang Li, Bing Deng, Jianqiang Huang, and Xian-sheng Hua. Quantization networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7308–7316, 2019. 1

[47] Tao Zhuang, Zhixuan Zhang, Yuheng Huang, Xiaoyi Zeng, Kai Shuang, and Xiang Li. Neuron-level structured pruning using polarization regularizer. *Advances in neural information processing systems*, 33:9865–9877, 2020. 1, 3

[48] Zhuangwei Zhuang, Mingkui Tan, Bohan Zhuang, Jing Liu, Yong Guo, Qingyao Wu, Junzhou Huang, and Jinhui Zhu. Discrimination-aware channel pruning for deep neural networks. *Advances in neural information processing systems*, 31, 2018. 6

[49] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016. 1