Structural Alignment for Network Pruning through Partial Regularization

Shangqian Gao[1], Zeyu Zhang[2], Yanfu Zhang[1], Feihu Huang[1], and Heng Huang[*3]

[1]Department of Electrical and Computer Engineering, University of Pittsburgh
[2]School of Information, University of Arizona
[3]Department of Computer Science, University of Maryland at College Park
{shg84, yaz91}@pitt.edu, zeyuzhang@email.arizona.edu, huangfeihu2018@gmail.com,
heng@umd.edu

Abstract

In this paper, we propose a novel channel pruning method to reduce the computational and storage costs of Convolutional Neural Networks (CNNs). Many existing one-shot pruning methods directly remove redundant structures, which brings a huge gap between the model before and after network pruning. This gap will no doubt result in performance loss for network pruning. To mitigate this gap, we first learn a target sub-network during the model training process, and then we use this sub-network to guide the learning of model weights through partial regularization. The target sub-network is learned and produced by using an architecture generator, and it can be optimized efficiently. In addition, we also derive the proximal gradient for our proposed partial regularization to facilitate the structural alignment process. With these designs, the gap between the pruned model and the sub-network is reduced, thus improving the pruning performance. Empirical results also suggest that the sub-network found by our method has a much higher performance than the one-shot pruning setting. Extensive experiments show that our method can achieve state-of-the-art performances on CIFAR-10 and ImageNet with ResNets and MobileNet-V2.

1. Introduction

Convolutional Neural Networks (CNNs) have achieved many successes in different computer vision tasks [27, 46, 47, 50, 2, 6, 11]. To tackle real-world challenges, recent CNNs [27, 51, 14] become larger and larger regarding width, depth, etc. With such capacities, CNNs can obtain better performance on different benchmarks at the cost of computational and storage burdens. At the same time, with the recent developments of mobile and embedded devices, the demand for deploying CNNs on these devices has increased dramatically. However, there is a natural conflict between the size of CNNs and the hardware capability of these devices. To overcome these challenges, many works [13, 12] try to reduce the size of CNNs, and make them possible to be deployed on edge devices.

There are many directions to reduce the size of CNNs. Among them, weight pruning and structural pruning are two popular topics. Structural pruning, especially channel pruning, is more friendly to hardware than weight pruning since no post-processing steps are required to acquire savings in computational and storage costs. Thus, our paper focuses on channel pruning for CNNs. Many existing one-shot pruning works [44, 56, 9, 42, 17, 8] prune trained models directly. No matter what method is used, there will be a significant gap between the selected sub-network and the pruned model. Such a gap creates difficulties in regaining performance during the fine-tuning process. On the other hand, soft pruning methods [16, 23] softly remove structures during the training process, which can produce good results with a shorter fine-tuning process. However, soft pruning methods generally perform worse than typical one-shot pruning methods, probably because the weight space is restricted during the training process because of soft pruning.

To tackle the above problems, we introduce a novel partial regularization technique to align model weights and the discovered sub-network during the training process, which can produce a high performance sub-network and reduce the gap between the sub-network and the original model. In addition, unlike soft-pruning methods, all model structures are used for training. The partial regularization term contains a partial group lasso regularization on selected weights, and other weights remain intact without modifica-

*Corresponding author. This work was partially supported by NSF IIS 1838627, 1837956, 1956002, 2211492, CNS 2213701, CCF 2217003, DBI 2225775.

tions. An architecture generator is trained to select which weights should be aligned, and it is also updated during the training process. Our partial regularization formulation is related to partial regularization in lasso [40]. Inspired by the nonmonotone proximal gradient (NPG) method used in [40], we also use a proximal gradient method to solve the partial regularization problem in our setting. Note that our method dynamically changes which channels should be put in the partial regularization. As a result, we add a scalar to balance the regularization strength for different layers because the number of pruned channels is different for them. To maximally keep the capacity of the original model, we insert the partial regularization in the middle of the training process. This is because weights are vulnerable to pruning at the early training stage, and the FLOPs regularization will dominate updates of the architecture generator, which can create bad sub-networks and mislead the training process. Finally, we update model weights and the architecture generator periodically, and they are connected by the partial regularization term during training. To maintain similar training efficiency as the original model, we only use a small portion of samples to train the architecture generator. Thus, there is only a small overhead compared to the original training process. Our method successfully finds performant sub-networks from the original model with these techniques. In summary, the contributions of this paper can be summarized as follows:

1) We propose to align the sub-network in the original model with the final pruned model through partial regularization. By structural alignment, the gap between the selected sub-network and the pruned model is largely reduced, which naturally improves the performance of the pruned model.
2) We use an architecture generator parameterized by neural networks to select the proper sub-network structure and guide the partial regularization. Inspired by partial regularization in lasso [40], we propose to solve the partial regularization problem via proximal gradients, which facilitate the alignment process.
3) Empirical results show that the sub-network discovered by our method performs much better than the one-shot pruning setting. Extensive experiments on CIFAR-10 and ImageNet show that our method outperforms existing channel pruning methods on ResNets and MobileNet-V2.

2. Related Works

Weight-Level Pruning. Weight pruning removes redundant connections based on individual weights. High compression rates can be achieved by weight pruning, but they can not directly achieve acceleration, and specially designed sparse matrix libraries are required. One of the early works [13] proposes to measure the importance of weights with their L_1 or L_2 magnitude, and unimportant weights are removed. Instead of magnitude, SNIP [28] updates the importance of each weight by using gradients from the loss function. SNIP can be used at the initialization time. The assumption of the Lottery ticket hypothesis [7] suggests that there exist sparse sub-networks (winning tickets) that can achieve the performance of the full model. In addition, with repeated training and fine-tuning, it can achieve better results. On the other hand, rethinking network pruning [39] argues that the learned topology from pruning algorithms is the key to achieving better performance. In addition, weight rewinding [48] shows that resetting weights to values from the middle training process can also produce good results. Similar to weight rewinding, our method does not modify weight training at the beginning, the partial regularization is inserted in the middle training process.

Structural-Level Pruning. Different from weight pruning, structural pruning is more friendly to hardware since it requires little or no post-processing steps to achieve acceleration. Similar to weight pruning, one of the early structural pruning methods [29] measure the importance of filters by using the sum of the absolute value of kernel weights. Besides using the magnitude to measure channel or filter importance, other methods utilize the scaling factor of batchnorm [21] to indicate which channels are important because batchnorm [21] is popular for the design of recent CNNs [14, 49]. To prune channels, Liu *et al.* [37] apply the sparse regularization to the scaling factors of batchnorm, and the channel will be pruned if the corresponding scaling factor is small. Structure sparse selection [20] introduces scaling factor to specific structures, such as neurons, groups, or residual blocks, and the sparsity regularization is applied to these structures. Structures with small values will also be removed. Another line of research formulates channel pruning as a constrained optimization problem [24, 56, 9, 23, 59], and learnable parameters are used to control each channel. These parameters are end-to-end differentiable, which is amenable to gradient based optimization methods. Our method is also related to these researches. Different from these methods, the learning of sub-networks accomplishes the training of model weights in our method. In addition, we use partial regularization to promote the selected sub-network, which reduces the gap between the pruned model and the sub-network. Besides using gates, Automatic Model Compression (AMC) [17] uses policy gradient to update the policy network. This policy network is then used to decide the left width of each layer. Collaborative channel pruning [44] prunes channels by exploiting inter-channel dependency. Greedy forward selection [55] starts from an empty network and greedily adds Important channels from the full model. MetaPruning [38, 32] uses a hypernet to generate parameters for sub-

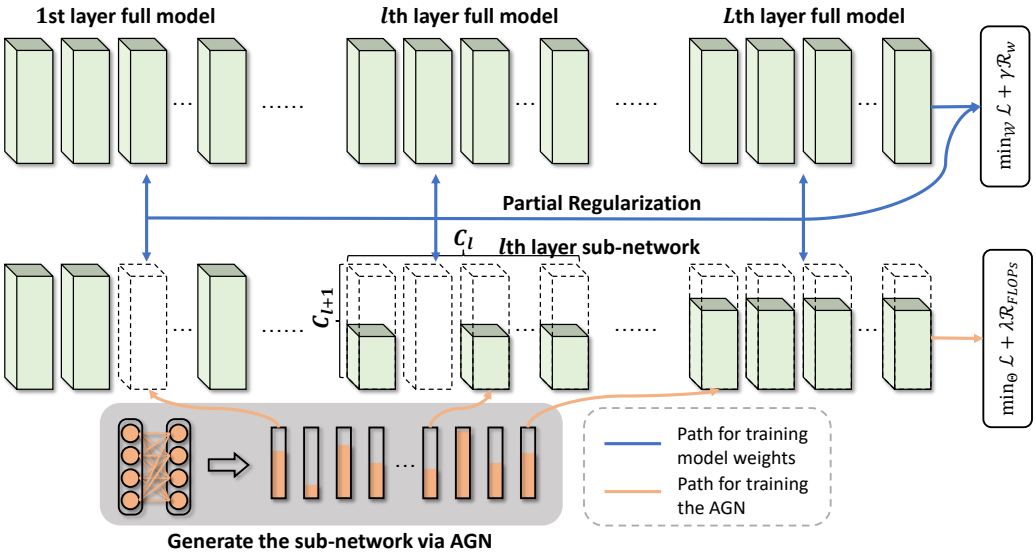


Figure 1: Overview of the proposed method. The AGN generates the sub-network to guide the partial regularization during the training process of model weights. The AGN is trained by evaluating sub-networks on a sub-set from the whole dataset. Both model weights and AGN are trained in an end-to-end differentiable manner.

networks, and evolutionary algorithms are utilized to find the best sub-networks. MetaPruning shows that pruning should accomplish trained model weights.

Besides these pruning methods, regularization based pruning methods can also be applied to structural pruning. Previous works [54, 31] use group sparsity to prune different structures. In addition to structural sparsity, structural sharing is considered in other works [58, 10]. Our method relates to regularization based methods; however, the formulation of our partial regularization only aligns selected channels dynamically, and other weights are intact.

Other Related Works. Besides the above-mentioned methods, there are works from other perspectives, including bayesian pruning [41, 43], weight quantization [4, 45], and knowledge distillation [19].

3. Proposed Method

3.1. Overview

Before introducing our method, we first describe notations and provide an overview of our method. In a CNN, the feature map of lth layer can be represented by $\mathcal{F}_l \in \Re^{C_l \times W_l \times H_l}$, $l = 1, \ldots, L$, where C_l is the number of channels, H_l and W_l are height and width of the current feature map, L is the number of layers. Similarly, the weights of lth layer can be written as $\mathcal{W}_l \in \Re^{C_l \times C_{l-1} \times k_l \times k_l}$, and k_l is the kernel size of this layer. The mini-batch dimension of feature maps is ignored to simplify notations.

The core motivation of our proposed method is to reduce the gap between the selected sub-network from the original model and the pruned model. To achieve this goal, we need two processes. First, we need to choose the desired sub-network from the original model, and this sub-network should also be updated when model weights are trained during the learning process. Secondly, we use the sub-network to guide the partial regularization term, which is used to decide which weights should be regularized. In addition, partial regularization should not be fixed to accommodate the changes in the selected sub-network.

3.2. Learning the Sub-network

We use an **A**rchitecture **G**enerator **N**etwork (AGN) to generate the desired sub-network architecture $\mathbf{v} \in \{0, 1\}^N$, where 0 or 1 is used to depict the removal or keep of a channel, and N is the total number of channels from all layers. The large parameterization space of AGN can facilitate the learning of sub-network structures. To generate $\mathbf{v}$, the following equation is used:

$$\mathbf{v} = \text{AGN}(\Theta), \tag{1}$$

and AGN is composed of gated recurrent unit (GRU) [3] and dense layers. In addition, Gumbel-Sigmoid [22] with STE [1] are used to produce the final binary vector $\mathbf{v}$, and they are placed after the output of dense layers. More details of AGN are provided in the supplementary materials.

Once we have $\mathbf{v}$, we can apply it to the feature maps to produce a sub-network. The feature map of the lth layer is then modified as follows:

$$\widehat{\mathcal{F}}_l = \mathbf{v}_l \odot \mathcal{F}_l, \tag{2}$$

where $\odot$ is element-wise multiplication, $\mathbf{v}_l$ is the architecture vector of lth layer, and $\mathbf{v}_l$ is resized to have the same size of $\mathcal{F}_l$. The feature map $\mathcal{F}_l$ is from the output of the activation function. The overall loss function for generating the desired sub-network is as follows:

$$\min_{\Theta} \mathcal{J}_\theta(\Theta) := \mathcal{L}\big(f(x; \mathcal{W}, \mathbf{v}), y\big) + \lambda \mathcal{R}_{\text{FLOPs}}(T(\mathbf{v}), pT_{\text{total}}) \tag{3}$$

where $T(\mathbf{v})$ is the current FLOPs decided by the vector $\mathbf{v}$, T_{total} is the total FLOPs of the original model, $p \in (0, 1]$ is a hyperparameter deciding the remaining fraction of FLOPs, λ is the hyper-parameter controlling the strength of FLOPs regularization, $f(x; \mathcal{W}, \mathbf{v})$ is a CNN parameterized by $\mathcal{W}$ and the sub-network structure is determined by the architecture vector $\mathbf{v}$, $\mathcal{L}$ is the task loss function and $\mathcal{R}_{\text{FLOPs}}$ is the regularization term for FLOPs. The regularization term $\mathcal{R}_{\text{FLOPs}}$ is $\mathcal{R}_{\text{FLOPs}}(x, y) = \log(\max(x, y)/y)$. With Eq. 3, we can find promising sub-networks when training the AGN and model weights periodically.

3.3. Partial Regularization

Given a sub-network $\mathbf{v}$ obtained from AGN, we can then reduce the gap between the sub-network and the pruned model and thus increase its performance. We can formulate the optimization problem with partial regularization as follows:

$$\min_{\mathcal{W}} \mathcal{J}_w(\mathcal{W}) := \mathcal{L}\big(f(x; \mathcal{W}), y\big) + \gamma \mathcal{R}_w(\mathcal{W}), \tag{4}$$

where $\mathcal{R}_w$ is the partial regularization term, and γ controls the strength of the partial regularization. $\mathcal{R}_w$ has the following form:

$$R_w(\mathcal{W}) = \sum_{l=1}^{L} \sum_{i \in S_l} \frac{\hat{N}_l}{\hat{N}} \|\mathcal{W}_{l[i,:,:,:]}\|_{\text{GL}}, \tag{5}$$

where $\hat{N}_l = \sum 1 - \mathbf{v}_l$, $\hat{N} = \sum 1 - \mathbf{v}$ and $S_l = \{i \mid \text{if } \mathbf{v}_{l[i]} = 0\}$. S_l contains the indices of pruned channels which are decided by AGN. $\frac{\hat{N}_l}{\hat{N}}$ is a scalar to adjust the regularization strength given different layers. The numerator $\hat{N}_l$ is the number of pruned channels of the lth layer, and the denominator $\hat{N}$ is the number of pruned channels from all layers. It is easy to see that $\sum_{l=1}^{L} \frac{\hat{N}_l}{\hat{N}} = 1$. $\|x\|_{\text{GL}}$ is the norm of grouped weights for Group Lasso, and $\|x\|_{\text{GL}} = \sqrt{\sum_{i=1}^{|x|} x_i^2}$ where $|x|$ represents the number of elements in x. In Eq. 5, we assume the corresponding layer is pruned across the output dimension. If it is pruned across the input dimension, we have $\sum_{i \in S_l} \|\mathcal{W}_{l[:,i,:,:]}\|_{\text{GL}}$.

The goal of using Group Lasso for our partial regularization is to reduce the distance between the dense model and the pruned model, other suitable functions may also be useful, but we found that the partial regularization with Group Lasso already produces good results. Another benefit of the partial regularization $\mathcal{R}_w$ is that it will not penalize weights that are not pruned. By doing so, we can avoid the problem of overpenalizing all weights' magnitude. In addition, $\mathbf{v}$ is updated during the training process, and $\mathcal{R}_w$ will dynamically regularize weights. As a result, $\mathbf{v}$ can flexibly change instead of falling into a fixed sub-network during the optimization process.

Algorithm 1: Structural Alignment for Network Pruning

Input: D, D_{AGN} ,p, λ, γ, E, E_{start},
Initialization: initialize $\mathcal{W}$ and θ.
for $e := 1$ *to* E **do**
  `/* Optimizing model weights. Freeze Θ of the AGN. */`
  for *a mini-batch* (x, y) *in* D **do**
    1. calculate the gradients w.r.t model weights: $\nabla_{\mathcal{W}} \mathcal{L}$.
    2. update model weights using any stochastic optimizer.
    3. **if** *if* $e \geq E_{start}$ **then**
      generate $\mathbf{v}$ from the AGN by using Eq. 1.
      apply the proximal gradient step following Eq. 8.
  end
  `/* Optimizing Θ of the AGN. Freeze model weights W. */`
  if *if* $e \geq E_{start}$ **then**
    for *a mini-batch* (x, y) *in* D_{AGN} **do**
      1. generate $\mathbf{v}$ from the AGN by using Eq. 1 and apply it to the model.
      2. calculate gradients w.r.t to $\mathcal{J}$ in Eq. 3: $\nabla_{\Theta} \mathcal{J}_\theta$
      3. update the AGN with ADAM
    end
end
Pruning the model with resulting $\mathbf{v}$, and fine-tuning it.

3.4. Proximal Gradients for Partial Regularization

Eq. 5 has a similar formulation of the partial regularization of lasso [40]. In [40], the related optimization problem is solved via a nonmonotone proximal gradient (NPG) method. However, NPG requires frequent evaluation of the loss function to ensure the loss value after proximal gradients is less or equal to the loss value before the update. With CNNs, the costs of NPG are too large due to frequent loss evaluations. As a result, we use a one-step proximal gradient update to solve the problem defined in Eq. 4.

The proximal operator of $\mathcal{R}_w$ is defined as:

$$\text{prox}_{\gamma \mathcal{R}_w}(x) = \arg\min_{y} \gamma \mathcal{R}_w(y) + \frac{1}{2}\|x - y\|^2. \tag{6}$$

In Eq. 6, $\|x - y\|^2$ defines the sum of the square difference

between x and y. We denote model weights after tth update as $\mathcal{W}^t$, and $\mathcal{W}^{t+1}$ can be obtained by:

$$\mathcal{W}^{t+1} = \text{prox}_{\alpha^{t+1}\gamma\mathcal{R}_w}(u(\mathcal{W}^t, \alpha^{t+1})), \tag{7}$$

where $u(\mathcal{W}^t, \alpha^{t+1})$ is an update rule that can be applied for many algorithms, and α^{t+1} is the learning rate at the current step. Take SGD as an example, $u(\mathcal{W}^t, \alpha^{t+1}) = \mathcal{W}^t - \alpha^{t+1}\nabla_{\mathcal{W}^t}\mathcal{L}$. With Eq. 6 and Eq. 7, model weights $\mathcal{W}$ can then be updated with proximal gradients.

We now present the analytical solution to Eq. 6, so that model weights can be efficiently updated. From the structure of $\mathcal{R}_w$, we know that channels with indices $i \in S_l$ will be regularized. This is equivalent to not regularizing channels if $i \notin S_l$. Consequently, we have the following form of the proximal gradient operator:

$$\text{prox}_{\alpha\gamma\mathcal{R}_w}(\mathcal{W}_l) = \begin{cases} \frac{\mathcal{W}_{l[i,:,:,:]}}{\|(1-\mathbf{v}_l)\odot\mathcal{W}_l\|_2} \max\left(0, -\frac{\hat{N}_l}{\hat{N}}\alpha\gamma \right. \\ \quad \left. +\|(1-\mathbf{v}_l)\odot\mathcal{W}_l\|_2\right), \text{ if } i \in S_l, \\ \mathcal{W}_{l[i,:,:,:]}, \text{ if } i \notin S_l. \end{cases} \tag{8}$$

In Eq. 8, we omit the step notation t to simplify the notations, and we still assume W_l is pruned along the output dimension. In this equation, it is easier to see that $\frac{\hat{N}_l}{\hat{N}}$ can balance the regularization strength given different layers. Due to the property of our proposed partial regularization, the term $\|(1-\mathbf{v}_l)\odot\mathcal{W}_l\|_2$ also changes dynamically, since $\hat{N}_l$ is changed after updates of the AGN. As a result, if no adjustment is applied, no matter how large or small $\hat{N}_l$ is, only a constant value $\frac{1}{L}\alpha\gamma$ will be used for the soft-thresholding, $\max(0, \|(1-\mathbf{v}_l)\odot\mathcal{W}_l\|_2 - \frac{1}{L}\alpha\gamma)$, in all circumstances, which is not reasonable. To accompany the changes of v_l, and consequently $\hat{N}_l$, we use $\frac{\hat{N}_l}{\hat{N}}$ to dynamically balance the soft-thresholding parameter between different layers. With Eq. 8, we can efficiently update $\mathcal{W}$ with our proposed partial regularization.

3.5. Network Pruning via Structural Alignment

We present the algorithm of our method in Algorithm. 1. In Algorithm. 1, D is the training dataset; D_{AGN} is a sub-dataset within D and it is used to train AGN; p decides how much FLOPs is preserved and it described in section 3.2; γ and λ are hyperparameters to control the strength of $\mathcal{R}_{\text{FLOPs}}$ and $\mathcal{R}_w$; E is the total number of epochs; E_{start} is the start epoch to train AGN and apply $\mathcal{R}_w$ when optimizing model weights. Note that, to reduce the overhead brought by training AGN, we only use a small sub-set sampled from D. As we discussed in section 1, we need to set a start epoch for AGN and $\mathcal{R}_w$. If we apply $\mathcal{R}_w$ when training starts ($E_{\text{start}} = 0$), it will largely restrict the final performance of the whole model before pruning. Instead, we can have deserved results if we start partial regularization in the middle of the training process. The reason for this problem can come from several perspectives. For example, at the beginning of the training, the classification loss can not produce accurate guidance for pruning since weights are not trained properly. Consequently, it will produce a bad sub-network, and following this sub-network only gives even worse results.

We summarize our method in Fig. 1. The inference path when training model weights and the AGN are different, and they are connected by partial regularization, which is different from current one-shot pruning and soft pruning methods. With this design, model weights are not directly affected by the sub-network architecture, which creates a smooth transition for the selected sub-network before and after pruning.

4. Experiments

4.1. Settings

We use CIFAR-10 [26] and ImageNet [5] to evaluate the performance of our method. Our method requires one hyper-parameter p to control the FLOPs budget. The detailed choices of p are listed in supplementary materials. We choose ResNets [14] and MobileNet-V2 [49] for comparison. For CIFAR-10, we compare our method with other methods on ResNet-56 and MobileNetV2. For ImageNet, we select ResNet-34, ResNet-50, ResNet-101, and MobileNetV2 as our target models.

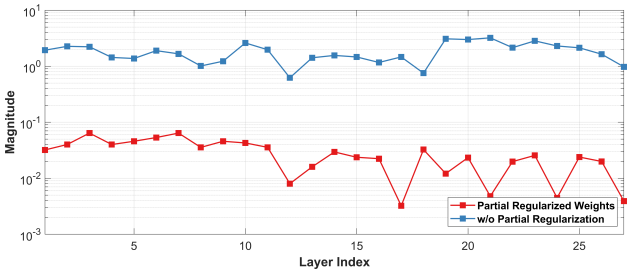
We set λ in Eq. 3 to 4.0 for all models and datasets. Similarly, we set γ to 0.0005 for all settings. We set E_{start} at 20% of the total training epochs. Detailed numbers of E_{start} are listed in supplementary materials. The range of E_{start} is quite large, which is hard to be explored thoroughly. The current setting already provides good results, but better settings may also exist. To reduce the training costs of the AGN, we random sample 5% of samples from the original dataset for constructing D_{AGN}. With this setup, the additional costs are less than 5% of the original training costs. We train the parameters Θ of the AGN using ADAM [25] with a start learning rate 0.001. Besides the training of the AGN, we follow the standard training recipe of ResNets for both CIFAR-10 and ImageNet. For MobileNet-V2, We follow the training settings in their original paper [49]. After training, we prune the model by using the sub-network generated from the AGN. The fine-tuning settings are similar to the training setting. Due to space limitations, details of training and fine-tuning are also presented in supplementary materials. In the experimental section, our method is abbreviated as **SANP**: **S**tructural **A**lignment for **N**etwork **P**runing.

4.2. CIFAR-10 Results

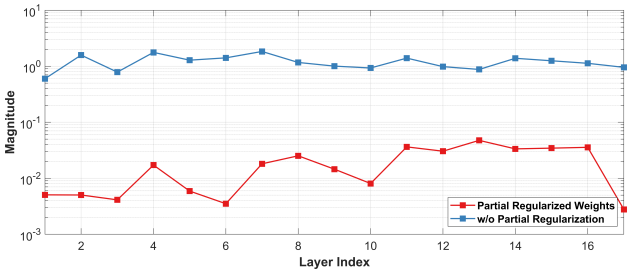
The results of CIFAR-10 are presented in Tab. 1. For ResNet-56, our method achieves the best performance (in terms of Δ-Acc) compared to other methods. Specifically,

Table 1: Comparison of results on CIFAR-10. Δ-Acc represents the performance changes relative to the baseline, and +/− indicates an increase/decrease, respectively.

Architecture	Method	Baseline Acc	Pruned Acc	Δ-Acc	Pruned FLOPs
ResNet-56	DCP-Adapt [60]	93.80%	93.81%	+0.01%	47.0%
	SCP [23]	93.69%	93.23%	−0.46%	51.5%
	FPGM [18]	93.59%	92.93%	−0.66%	52.6%
	SFP [16]	93.59%	92.26%	−1.33%	52.6%
	FPC [15]	93.59%	93.24%	−0.25%	52.9%
	HRank [35]	93.26%	92.17%	−0.09%	50.0%
	DMC [9]	93.62%	92.69%	+0.07%	50.0%
	GNN-RL [57]	93.49%	93.59%	+0.10%	**54.0%**
	SANP (ours)	93.49%	**93.81%**	**+ 0.32%**	52.0%
MobileNetV2	Uniform [60]	94.47%	94.17%	−0.30%	26.0%
	DCP [60]	94.47%	94.69%	+0.22%	26.0%
	DMC [9]	94.23%	94.49%	+0.26%	40.0%
	SCOP [53]	94.48%	94.24%	−0.24%	40.3%
	SANP (ours)	94.52%	**94.97%**	**+0.45%**	**46.0%**



(a) ResNet-56.



(b) MoibleNet-V2.

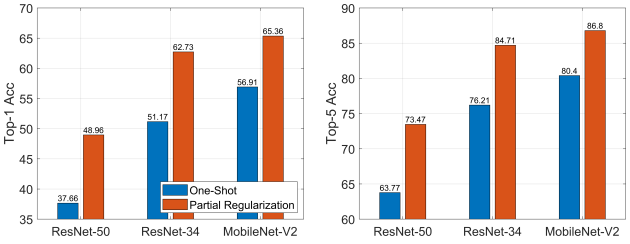
Figure 2: (a) and (b): the average norm of channels with or without partial regularization for ResNet-56 and MobileNet-V2.

our method outperforms the second best method GNN-RL by 0.22% regarding Δ-Acc (SANP +0.32% vs. GNN-RL +0.10%) when pruning similar FLOPs (SANP 52.0% vs. GNN-RL 54.0%). Our method also outperforms HRank and DMC by 0.25% and 0.41% separately (DMC +0.07% and HRank −0.09%). The gap between other methods and our method is even larger. For MobileNet-V2, our method prunes most FLOPs (46.0%) and achieves the best performance (Δ-Acc: +0.45%). Compared to the second best method, DMC, our method prunes 6% more FLOPs and outperforms it by 0.19%. SCOP also prunes 40% FLOPs, and the gap between our method and SCOP is even larger (0.69% better in terms of Δ-Acc). The uniform setting and DCP prunes around 26% FLOPs, but the performance is still worse than our method.

4.3. ImageNet Results

All results for the ImageNet dataset are shown in Tab. 2.

ResNet-34. Our method achieves 73.43% Top-1 accuracy and 91.48% Top-5 accuracy, which is better than the baseline by 0.19% and 0.16% for Top-1/Top-5 accuracy separately. At the same time, our method removes 44.1% FLOPs, which is on par with other methods. It is obvious



(a) (b)

Figure 3: Top-1 and Top-5 accuracy after pruning given different settings.

that the advantage of our method is clear compared to other methods. Our method prunes similar FLOPs to SCOP and DMC. However, the Δ Top-1 Acc of our method is 0.88% and 0.92% better than SCOP and DMC, respectively, and we have similar observations for Δ Top-5 Acc (0.60% and 0.47% better than SCOP and DMC). Taylor has the second best Δ Top-1 Acc, but the pruned FLOPs is much lower than our method (ours: 44.1% vs. Taylor: 24.2%). The advantage of our method compared to FPGM is more obvious.

ResNet-50. Our method achieves 76.47% Top-1 accuracy and 93.00% Top-5 accuracy, which is also better than the

Table 2: Comparison results on ImageNet with ResNet-34/50/101 and MobileNet-V2.

Architecture	Method	Baseline Top-1 Acc	Baseline Top-5 Acc	Δ Top-1 Acc	Δ Top-5 Acc	Pruned FLOPs
ResNet-34	FPGM [18]	73.92%	91.62%	−1.29%	−0.54%	41.1%
	Taylor [42]	73.31%	-	−0.48%	-	24.2%
	DMC [9]	73.30%	91.42%	−0.73%	−0.31%	43.4%
	SCOP [53]	73.31%	91.42%	−0.69%	−0.44%	**44.8%**
	SANP (ours)	73.24%	91.32%	**+0.19%**	**+0.16%**	**44.1%**
ResNet-50	DCP [60]	76.01%	92.93%	−1.06%	−0.61%	55.6%
	CCP [44]	76.15%	92.87%	−0.94%	−0.45%	54.1%
	FPGM [18]	76.15%	92.87%	−1.32%	−0.55%	53.5%
	ABCP [36]	76.01%	92.96%	−2.15%	−1.27%	54.3%
	DMC [9]	76.15%	92.87%	−0.80%	−0.38%	55.0%
	SCOP [53]	76.15%	92.87%	−0.89%	−0.34%	54.6%
	PFP [34]	76.13%	92.86%	−0.92%	−0.45%	44.0%
	CHIP [52]	76.15%	92.87%	+0.00%	+0.04%	48.7%
	NPPM [8]	76.15%	92.87%	−0.19%	+0.12%	56.0%
	Random-Pruning [30]	75.83%	92.92%	−0.75%	−0.40%	51.0%
	GNN-RL [57]	76.10%	-	−1.82%	–	53.0%
	SANP (ours)	76.06%	92.86%	**+ 0.41%**	**+ 0.17%**	**56.2%**
ResNet-101	FPGM [18]	77.37%	93.56%	−0.05%	0.00%	41.1%
	Taylor [42]	77.37%	-	−0.02%	-	39.8%
	DMC [9]	77.37%	93.56%	+0.04%	+0.03%	**56.0%**
	PFP [34]	77.37%	93.56%	−0.94%	−0.44%	45.1%
	SANP (ours)	77.53%	93.71%	**+ 0.61%**	**+ 0.29%**	55.4%
MobileNet-V2	Uniform [49]	71.80%	91.00%	−2.00%	−1.40%	30.0%
	AMC [17]	71.80%	-	−1.00%	-	30.0%
	CC [33]	71.88%	-	−0.97%	-	28.3%
	MetaPruning [38]	72.00%	-	−0.80%	-	**30.7%**
	Random-Pruning [30]	71.88%	-	−1.01%	-	29.1%
	SANP (ours)	71.91%	90.30%	**+ 0.14%**	**+ 0.07%**	29.1%

Settings	Architecture	Baseline Top-1 Acc	Δ Top-1 Acc	Pruned FLOPs
One-Shot	ResNet-34	73.31%	−0.50%	44.0%
SANP		73.24%	+0.19%	44.1%
One-Shot	ResNet-50	76.13%	−0.57%	56.0%
SANP		76.06%	+0.41%	55.2%
One-Shot	MobileNetV2	71.88%	−0.42%	29.3%
SANP		71.91%	+0.07%	29.1%

Table 3: Performance of pruned models given different pruning settings on ImageNet.

baseline Top-1/Top-5 accuracy. The second best method, CHIP, removes 48.7% FLOPs while maintaining the original performance. Our method outperforms CHIP by 0.41% in terms of the Δ Top-1 accuracy while pruning near 8% more FLOPs. NPPM is a strong baseline for ResNet-50, our method outperforms by 0.60% in Δ -Top-1 Acc. GNN-RL and Random-Pruning are two recent pruning works. Our method outperforms them by 2.23% and 1.16% separately, while our method prunes more FLOPs. PHP, DCP, CCP, and DMC have similar Δ-Top 1 accuracy. The gap between our method and these methods ranges from 1.21% to 1.47% regarding Δ-Top 1 accuracy. The advantage of our method compared to the rest methods is more apparent.

ResNet-101. Our method achieves 78.14% Top-1 accuracy and 94.00% Top-5 accuracy, which is 0.61% and 0.29% better than the baseline Top-1 and Top-5 accuracy. Although DMC prunes a little bit more FLOPs, it only achieves 77.41% Top-1 accuracy after pruning which is 0.73% lower than our method. FPGM, Taylor, and PFP remove less than 50% FLOPs. Our method prunes at least 10% more FLOPs and still has an advantage in terms of Δ-Top 1 accuracy (from 0.66% to 1.55%).

MobileNet-V2. MobileNet-V2 is generally harder to prune compared to ResNets. All comparison methods on MobileNet-V2 remove around 30% FLOPs. Our method achieves 72.05% Top-1 accuracy and 90.37% Top-5 accuracy, which is 0.14% and 0.07% better than the baseline Top-1 and Top-5 accuracy. Given the similar pruning rate, our method is 1.15%, 0.94%, 1.11% and 1.14% higher than Random-Pruning, MetaPruning CC, and AMC separately regarding Δ-Top-1 Acc. MetaPruning prunes most FLOPs, but the performance is much lower than our method. In short, our method can also be applied to lightweight CNNs, like MobileNet-V2.

4.4. Analysis of Our Method

The effectiveness of partial regularization. To verify whether our proposed partial regularization is effective, we plot the average channel norm of different groups within each block for ResNet-56 and MobileNet-V2 on CIFAR-10. The results are shown in Fig. 2. The average channel norm for the group with partial regularization and without partial regularization are obtained by $\frac{1}{\hat{N}_l}\sum_{i\in S_l}\|\mathcal{W}_{l[i,:,:,:]}\|_{\text{GL}}$ and $\frac{1}{C_l-\hat{N}_l}\sum_{i\notin S_l}\|\mathcal{W}_{l[i,:,:,:]}\|_{\text{GL}}$ separately. Weights with partial regularization are effectively aligned. On the contrary, weights without partial regularization are lightly affected, which justifies the strength of our proposed partial regular-

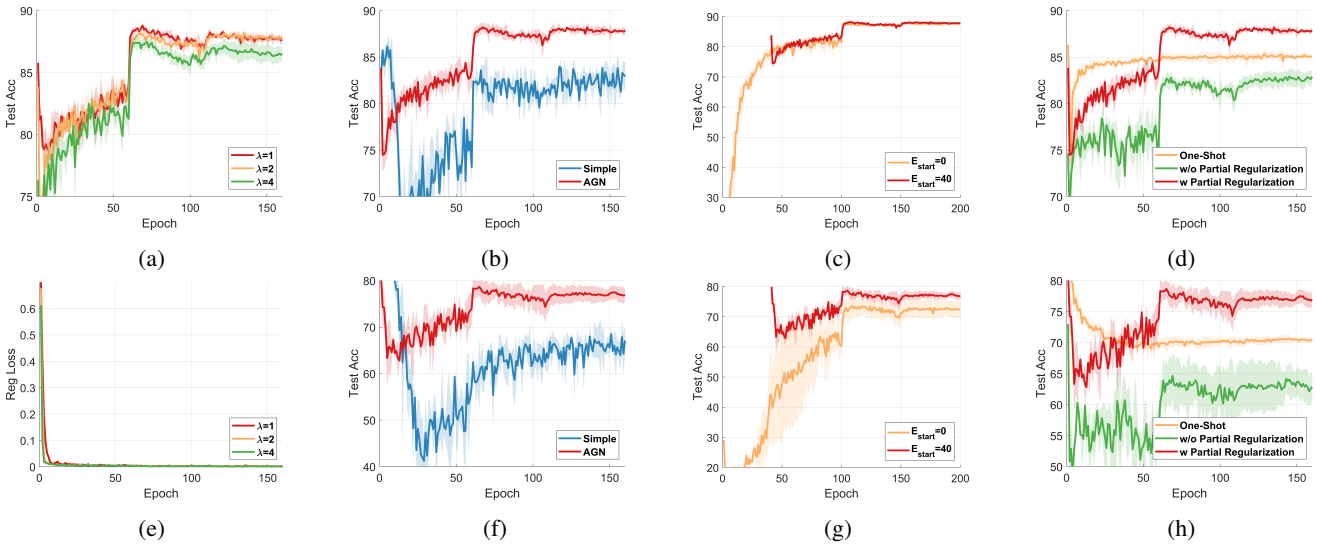


Figure 4: (a, e): the impact of λ in $\mathcal{R}_{\text{FLOPs}}$. (b, f): the effect of the architecture of AGN. (c, g): the effect of E_{start}. (d, h): the effect of different setups. Experiments are conducted on CIFAR-10 with ResNet-56 and $p = 0.5$ (a,b,c,d,e) and $p = 0.35$ (f,g,h).

ization.

The impact of λ. We study the impact of λ when training the AGN, and we plot the test accuracy and $\mathcal{R}_{\text{FLOPs}}$ in Fig. 4a and Fig. 4e. From the figures, we can see that if λ is too large, it will have negative impacts on learned sub-networks. Otherwise, our method is robust to λ,

The effect of AGN. In Fig. 4b and Fig. 4f, we plot the test accuracy when learning the AGN given different pruning rates. We construct a **Simple** baseline, which parameterizes each channel by using one learnable parameter. We can see that when the parameterization space shrinks, the performance of learned sub-networks will be affected severely. In addition, the learning is slower, and the best sub-network performance is also much worse than using AGN.

The effect of $\mathbf{E_{start}}$. In the method section, we argue that inserting partial regularization in the middle training process is beneficial. We plot the results of $E_{\text{start}} = 0$ and $E_{\text{start}} = 40$ in Fig. 4c and Fig. 4g. In general, they can find sub-networks with similar performance, but $E_{\text{start}} = 0$ is worse than $E_{\text{start}} = 40$ when the pruning rate is large ($p = 0.35$). More importantly, the full model accuracy of $E_{\text{start}} = 0$ is 92.96% (average across different runs), which is around 0.50% worse than $E_{\text{start}} = 40$, which suggests that using partial regularization at the beginning will limit the capacity of the full model.

The effect of different setups. We construct two additional baselines to see how partial regularization helps to produce a better sub-network within the full model. The **One-Shot** baseline directly trains the AGN on the pre-trained model. The **w/o Partial Regularization** baseline set $\gamma = 0$, and the rest settings are the same as our method. The related results are shown in Fig. 4d and Fig. 4h. From these figures, we can see that partial regularization always produces the best sub-network from the full model. In addition, 'w/o partial regularization' is worse than the one-shot setting. This is probably because it is hard to capture the changes of model weights without partial regularization, which makes the training of AGN much harder than the rest settings.

More comparisons on the ImageNet dataset. To further show how our method improves the one-shot setting, we present the sub-network performance before and after fine-tuning for one-shot and partial regularization settings in Fig. 3 and Tab. 3. From Fig. 3, we can see that partial regularization still produces better sub-networks on ImageNet, and it is around 10% better in terms of Top-1 accuracy than the one-shot setting for different models. The advantage of partial regularization naturally extends to results after fine-tuning, as shown in Tab. 3.

5. Conclusion

In this paper, we investigate how partial regularization helps to produce a better sub-network for network pruning. Specifically, our method uses AGN to guide partial regularization across the training process. We further provide an efficient way to update model weights through proximal gradients. With these designs, partial regularization effectively reduces the gap between the sub-network within the full model and the pruned model. Our method then starts from a better sub-network, thus resulting in a better final pruned model. Extensive experimental results on CIFAR-10 and ImageNet show the effectiveness of our method.

References

[1] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013. 3

[2] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016. 1

[3] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014. 3

[4] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. Binaryconnect: Training deep neural networks with binary weights during propagations. In *Advances in neural information processing systems*, pages 3123–3131, 2015. 3

[5] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. Ieee, 2009. 5

[6] Xin Dong, Junfeng Guo, Ang Li, Wei-Te Ting, Cong Liu, and H.T. Kung. Neural mean discrepancy for efficient out-of-distribution detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 19217–19227, June 2022. 1

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

[8] Shangqian Gao, Feihu Huang, Weidong Cai, and Heng Huang. Network pruning via performance maximization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9270–9280, 2021. 1, 7

[9] Shangqian Gao, Feihu Huang, Jian Pei, and Heng Huang. Discrete model compression with resource constraint for deep neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1899–1908, 2020. 1, 2, 6, 7

[10] Shangqian Gao, Burak Uzkent, Yilin Shen, Heng Huang, and Hongxia Jin. Learning to jointly share and prune weights for grounding based vision and language models. In *The Eleventh International Conference on Learning Representations*, 2023. 3

[11] Junfeng Guo, Yiming Li, Xun Chen, Hanqing Guo, Lichao Sun, and Cong Liu. SCALE-UP: An efficient black-box input-level backdoor detection via analyzing scaled prediction consistency. In *The Eleventh International Conference on Learning Representations*, 2023. 1

[12] 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

[13] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In *Advances in neural information processing systems*, pages 1135–1143, 2015. 1, 2

[14] 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. 1, 2, 5

[15] Yang He, Yuhang Ding, Ping Liu, Linchao Zhu, Hanwang Zhang, and Yi Yang. Learning filter pruning criteria for deep convolutional neural networks acceleration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2009–2018, 2020. 6

[16] 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, 6

[17] 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. 1, 2, 7

[18] 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 Conference on Computer Vision and Pattern Recognition*, pages 4340–4349, 2019. 6, 7

[19] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. 3

[20] 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. 2

[21] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ICML, pages 448–456. JMLR.org, 2015. 2

[22] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016. 3

[23] Minsoo Kang and Bohyung Han. Operation-aware soft channel pruning using differentiable masks. In *International Conference on Machine Learning*, pages 5122–5131. PMLR, 2020. 1, 2, 6

[24] Jaedeok Kim, Chiyoun Park, Hyun-Joo Jung, and Yoonsuck Choe. Plug-in, trainable gate for streamlining arbitrary neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020. 2

[25] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 5

[26] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009. 5

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

[28] Namhoon Lee, Thalaiyasingam Ajanthan, and Philip HS Torr. Snip: Single-shot network pruning based on connection sensitivity. *ICLR*, 2019. 2

[29] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. *ICLR*, 2017. 2

[30] Yawei Li, Kamil Adamczewski, Wen Li, Shuhang Gu, Radu Timofte, and Luc Van Gool. Revisiting random channel pruning for neural network compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 191–201, 2022. 7

[31] 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. 3

[32] 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. 2

[33] Yuchao Li, Shaohui Lin, Jianzhuang Liu, Qixiang Ye, Mengdi Wang, Fei Chao, Fan Yang, Jincheng Ma, Qi Tian, and Rongrong Ji. Towards compact cnns via collaborative compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6438–6447, 2021. 7

[34] Lucas Liebenwein, Cenk Baykal, Harry Lang, Dan Feldman, and Daniela Rus. Provable filter pruning for efficient neural networks. In *International Conference on Learning Representations*, 2020. 7

[35] Mingbao Lin, Rongrong Ji, Yan Wang, Yichen Zhang, Baochang Zhang, Yonghong Tian, and Ling Shao. Hrank: Filter pruning using high-rank feature map. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 6

[36] Mingbao Lin, Rongrong Ji, Yuxin Zhang, Baochang Zhang, Yongjian Wu, and Yonghong Tian. Channel pruning via automatic structure search. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 673 – 679, 2020. 7

[37] Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. Learning efficient convolutional networks through network slimming. In *ICCV*, 2017. 2

[38] 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 *Proceedings of the IEEE International Conference on Computer Vision*, pages 3296–3305, 2019. 2, 7

[39] Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. Rethinking the value of network pruning. In *International Conference on Learning Representations*, 2019. 2

[40] Zhaosong Lu and Xiaorui Li. Sparse recovery via partial regularization: Models, theory, and algorithms. *Mathematics of Operations Research*, 43(4):1290–1316, 2018. 2, 4

[41] Dmitry Molchanov, Arsenii Ashukha, and Dmitry Vetrov. Variational dropout sparsifies deep neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2498–2507. JMLR. org, 2017. 3

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

[43] Kirill Neklyudov, Dmitry Molchanov, Arsenii Ashukha, and Dmitry P Vetrov. Structured bayesian pruning via log-normal multiplicative noise. In *Advances in Neural Information Processing Systems*, pages 6775–6784, 2017. 3

[44] Hanyu Peng, Jiaxiang Wu, Shifeng Chen, and Junzhou Huang. Collaborative channel pruning for deep networks. In *International Conference on Machine Learning*, pages 5113–5122, 2019. 1, 2, 7

[45] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European Conference on Computer Vision*, pages 525–542. Springer, 2016. 3

[46] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016. 1

[47] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015. 1

[48] Alex Renda, Jonathan Frankle, and Michael Carbin. Comparing rewinding and fine-tuning in neural network pruning. In *International Conference on Learning Representations*, 2020. 2

[49] 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. 2, 5, 7

[50] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In *Advances in neural information processing systems*, pages 568–576, 2014. 1

[51] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 1

[52] 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, 2021. 7

[53] Yehui Tang, Yunhe Wang, Yixing Xu, Dacheng Tao, Chunjing Xu, Chao Xu, and Chang Xu. Scop: Scientific control for reliable neural network pruning. *Advances in Neural Information Processing Systems*, 33, 2020. 6, 7

[54] Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Learning structured sparsity in deep neural networks.

In *Advances in neural information processing systems*, pages 2074–2082, 2016. 3

[55] Mao Ye, Chengyue Gong, Lizhen Nie, Denny Zhou, Adam Klivans, and Qiang Liu. Good subnetworks provably exist: Pruning via greedy forward selection. In *International Conference on Machine Learning*, pages 10820–10830. PMLR, 2020. 2

[56] Zhonghui You, Kun Yan, Jinmian Ye, Meng Ma, and Ping Wang. Gate decorator: Global filter pruning method for accelerating deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 2130–2141, 2019. 1, 2

[57] Sixing Yu, Arya Mazaheri, and Ali Jannesari. Topology-aware network pruning using multi-stage graph embedding and reinforcement learning. In *International Conference on Machine Learning*, pages 25656–25667. PMLR, 2022. 6, 7

[58] Dejiao Zhang, Haozhu Wang, Mario Figueiredo, and Laura Balzano. Learning to share: Simultaneous parameter tying and sparsification in deep learning. 2018. 3

[59] Yanfu Zhang, Shangqian Gao, and Heng Huang. Exploration and estimation for model compression. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 487–496, 2021. 2

[60] 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. In *Advances in Neural Information Processing Systems*, pages 875–886, 2018. 6, 7