Discrete Model Compression with Resource Constraint for Deep Neural Networks

Shangqian Gao[1,3], Feihu Huang[1], Jian Pei[2], and Heng Huang*[1,3]

[1]Electrical and Computer Engineering Department, University of Pittsburgh, PA, USA
[2]School of Computing Science, Simon Fraser University, BC, Canada
[3]JD Finance America Corporation, Mountain View, CA, USA
shg84@pitt.edu, feh23@pitt.edu, jpei@cs.sfu.ca, heng.huang@pitt.edu

Abstract

In this paper, we target to address the problem of compression and acceleration of Convolutional Neural Networks (CNNs). Specifically, we propose a novel structural pruning method to obtain a compact CNN with strong discriminative power. To find such networks, we propose an efficient discrete optimization method to directly optimize channel-wise differentiable discrete gate under resource constraint while freezing all the other model parameters. Although directly optimizing discrete variables is a complex non-smooth, non-convex and NP-hard problem, our optimization method can circumvent these difficulties by using the straight-through estimator. Thus, our method is able to ensure that the sub-network discovered within the training process reflects the true sub-network. We further extend the discrete gate to its stochastic version in order to thoroughly explore the potential sub-networks. Unlike many previous methods requiring per-layer hyper-parameters, we only require one hyper-parameter to control FLOPs budget. Moreover, our method is globally discrimination-aware due to the discrete setting. The experimental results on CIFAR-10 and ImageNet show that our method is competitive with state-of-the-art methods.

1. Introduction

Convolutional Neural Networks (CNNs) have achieved great success in computer vision tasks [23, 39, 40, 42, 2]. With more and more sophisticated GPU support on CNNs, the complexity of CNN grows dramatically from several layers [23, 43] to hundreds of layers [9, 17]. Although these complex CNNs can achieve strong performance on vision tasks, there is an unavoidable growth of the computational cost and model parameters. Such a huge computational burden prohibits the model from being deployed on mobile devices and resource-limited platforms. Even if the model can be deployed on mobile devices, the battery will be depleted quickly due to huge computational costs. To tackle such problems, many efforts [8, 7] have been devoted for getting compact sub-networks from the original computational heavy model.

Structural pruning, especially channel pruning, is an efficient way to reduce computational cost since it doesn't require any post-processing steps to acquire acceleration. One of the most challenging parts of structural pruning is how to deal with the natural discrete configuration of channels in each layer. Many existing works [31, 37] try to solve this problem by relaxing discrete values to continuous values. However, such relaxation may lead to a biased estimation of the corresponding pruning criterion, since you can't completely remove the impact of channels with small importance value. Some other methods [28, 34] use the estimation of channel importance to decide whether to prune a channel, nonetheless, the relative importance of a channel can be changed due to the choice of the sub-network. Recently, discrimination-aware pruning [49] has been proposed to explore the impact of discriminative power on channel pruning. Although this method considers the discriminative power of CNNs by adding classifiers on intermediate layers, it does not consider CNN as a whole, which may result in sub-optimal compression results.

To deal with these challenges, we propose a new method of using the discrete gate to turn off or turn on certain channels. By doing so, we can always get the exact model output given different sub-network architectures. Thanks to the precise output estimation of a sub-network, our method

*Corresponding Author. This work was partially supported by U.S. NSF IIS 1836945, IIS 1836938, IIS 1845666, IIS 1852606, IIS 1838627, IIS 1837956.

is able to consider the discriminative power of a complete sub-network. At the same time, we do not take the magnitude of a channel into consideration, the global discriminative power is the only criterion. Moreover, we propose an efficient optimization method to obtain the sub-network. Although directly optimizing discrete variables is often non-smooth non-convex and NP-hard, our optimization method can circumvent these difficulties by using the straight-through estimator (STE) [1].

Automatic Model Compression (AMC) [11] is a pioneer method using the discrete channel setting, which is optimized by reinforced learning. Different from AMC, our method is guided by gradients of the loss function when exploring sub-networks from the original CNN . Our method can obtain a sub-network efficiently due to its differentiable nature. On ImageNet dataset, our method can discover a high performance sub-network satisfying given budget within 2% of time for regular training (finetuning time is excluded). From this perspective, our method is also related to differentiable architecture search (DARTS) [27].

Our main contributions are summarized as follows:

1) To compress a model, we apply the discrete gate on each channel, which ensures that the output from any sub-networks is correct and unbiased.
2) We use STE to enable back-propagation through discrete variables. To further enlarge the search space of sub-networks, we replace the discrete gate with its stochastic version.
3) To ensure we can get the model with the given computational budget, we further propose resource regularization when exploring potential sub-networks.
4) Extensive experimental results show that our method achieves state-of-the-art results on CIFAR10 and ImageNet with ResNet and MobileNetV2.

2. Related Works

Model compression recently has drawn a lot of attention from the compute vision community. In general, current model compression methods can be separated into the following four categories: weight pruning, structural pruning, weight quantization, and knowledge distillation [14].

Weight pruning eliminates model parameters without any assumption on the structure of weights. One of the early works [8] uses L_1 or L_2 magnitude as criterion to remove weights. Under this setting, parameters lower than a certain threshold are removed, and weights with small magnitude are considered not important. A systematic DNN weight pruning framework [48] has been proposed by using alternating direction method of multipliers (ADMM) [3, 15, 16]. Different from the aforementioned works, SNIP [24] updates the importance of weights by backpropagating gradients from the loss function. Lottery ticket hypothesis [6] is another very interesting weight pruning algorithm, which manifests that small high-performance sub-networks exist within the overparameterized large network at initialization time. Different from the lottery ticket hypothesis, in rethinking the value of network pruning [29], they argue that fine-tuning a pre-trained model is not necessary and show that the pruned model with random initialization achieves better performance. he major drawback of weight level pruning is that they often require specially designed sparse matrix multiplication library to achieve acceleration.

Different from weight pruning, structural pruning provides a natural way to reduce computational costs. The development of structural pruning is similar to weight pruning in that channels with low magnitude are often regarded as not important [25]. Similar to the idea of magnitude pruning, Group Lasso [45] is also applied on CNNs to structurally make channels or filters to have all 0 values, thus those channels can be safely removed. GrOWL [47] focuses on exploring inter-channel relations on top of sparsity, and argues that similar channels can share the same weights. The following researches show that weights with small magnitude could be important [30], and it's difficult for channels under L_1 regularization to achieve exact zero values. To compensate this, they propose to get exact zero values for each channel by using explicit L_0 regularization [30]. Besides simply using channel magnitude as pruning standards, other methods utilize batchnorm to achieve the similar target, since batchnorm [19] is an indispensable component in recent neural network designs [9, 17]. For each channel, batchnorm uses a scaling factor γ to adjust the magnitude of corresponding feature maps. To achieve the goal of channel pruning, γ is regularized to be sparse and γ fell below a predefined threshold will be set to 0 during model pruning [28]. Other works related to this idea includes [46, 18, 20]. Unlike previous methods relying on the magnitude of channels, discrimination-aware pruning [49] utilizes local discriminative power to help channel pruning. AMC [11] achieves the goal of channel pruning in a discrete setting by taking the advantages of reinforcing learning. Collaborative channel pruning [37] focuses on pruning channels by utilizing Taylor expansion of the loss function. Our method belongs to this category, the major difference between our method and previous model pruning methods is that our method strictly uses the discrete setting of channels while it can be optimized through gradient descent.

Weight quantization is another direction for model compression, which focuses on reducing the numerical precision of weights from 32-bit float point value to low bit value. Binary connects [4] and binary neural network [38] push the full precision weight to binary weight values, making the model weights become binary values. The connection between our method and weight quantization is that both of them use STE to estimate the gradient for discrete value.

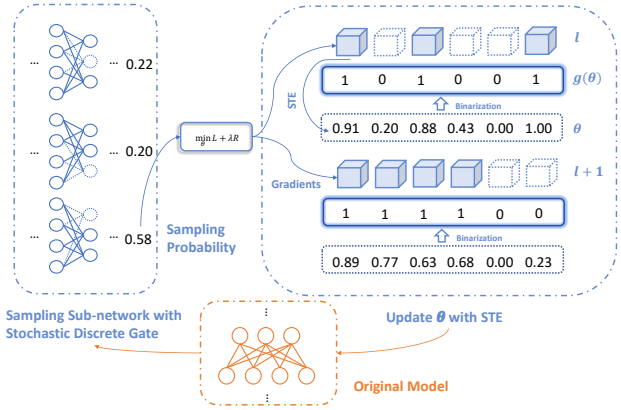


Figure 1: The gate training process of the proposed method. A sub-network is sampled according to Eq. (4). Then θ is optimized through STE with gradient descent. At the next iteration, the sub-network is sampled with the updated θ.

Besides the aforementioned methods, there are works from other directions. There is a range of methods focus on pruning weights [33] or structures [35] by utilizing uncertainty in weights. EigenDamge [44] can achieve structural pruning by using Kronecker-factored eigenbasis.

3. Proposed Method

3.1. Notations

To better describe proposed approach, we first define some notations. The feature map of each layer can be represented by $\mathcal{F}_l \in \Re^{C_l \times W_l \times H_l}$ where C_l is the number of channels, W_l and H_l are height and width of the current feature map. $\mathcal{F}_{l,c}$ is the feature map of c-th channel from l-th layer. The mini-batch dimension of feature map is ignored to simplify notations. Throughout the paper, *w.p.* means with probability. $\mathbf{1} = [1, \ldots, 1]^T$ is a vector with all ones. $\text{sign}(\cdot)$ is the popular sign function.

3.2. Differentiable Discrete Gate

In this paper, to incorporate the discrete nature of channel pruning, we explicitly consider using discrete-valued gates to represent open or close of a channel. The discrete gate function can be described as follows:

$$g(\theta) = \begin{cases} 1 & \text{if } \theta \in [0.5, 1], \\ 0 & \text{if } \theta \in [0, 0.5), \end{cases} \tag{1}$$

where $\theta \in [0, 1]$ is a learnable parameter in our setting. The discrete gate function is applied after the output feature map of a layer:

$$\widehat{\mathcal{F}}_{l,c} = g(\theta_{l,c}) \cdot \mathcal{F}_{l,c}, \tag{2}$$

where $\widehat{\mathcal{F}}_{l,c}$ is the feature map after pruning.

Since the binary gate function is not differentiable, STE [1] is used to enable gradient calculation, which can be described as follows:

$$\frac{\partial \mathcal{L}}{\partial \theta} = \frac{\partial \mathcal{L}}{\partial g(\theta)}, \tag{3}$$

where $\mathcal{L}$ is the loss function. Here, the backward propagation of $g(\theta)$ can be understood as an identity function within certain range. If $\theta \notin [0, 1]$, the gradient will not be calculated, and the θ will be clipped to range $[0, 1]$.

In fact, there are some limitations of the deterministic discrete gate function. For example, once the $g(\theta)$ of certain channels become 0, then these channels are probably remained pruned, and they may never be selected as candidate channels. To compensate such situations, we further propose the stochastic discrete gate to ensure gates with θ lower than 0.5 can be considered as candidate channels again. Specifically, the stochastic discrete gate is achieved by applying stochastic rounding:

$$g(\theta) = \begin{cases} 1 & \textit{w.p. } \theta \\ 0 & \textit{w.p. } 1 - \theta \end{cases} \tag{4}$$

where θ is within $[0, 1]$ to satisfy the definition of probability. From this definition, we can see that a channel always has a chance to be sampled if $\theta \neq 0$. Since our method uses the discrete setting, sampling from the stochastic discrete gate is equivalent to sample a sub-network.

3.3. Model Compression as Constrained Optimization

There are many different ways to represent the pruning objective for model compression. In this paper, we mainly focus on the following channel pruning problem:

$$\begin{aligned} &\min_{\Theta} \mathcal{L}\big(f(x; \mathcal{W}, \Theta), y\big) \\ &s.t.\ \mathbf{1}^T\mathbf{g} - p\mathbf{1}^T\mathbf{C} = 0 \\ &\mathbf{g} \in \{0, 1\}^n, \end{aligned} \tag{5}$$

where $\mathbf{g} = (g_1, \ldots, g_L)$ is a vector containing all gate values, $\mathbf{g}_l = [g(\theta_{l,1}), \ldots, g(\theta_{l,C_l})]^T$ is the vector containing gate values in l-th layer, Θ are the parameters of discrete gates following the definition in section 3.2, $\mathcal{W}$ is the model parameters, p is a predefined threshold working as the pruning rate, and $\mathbf{C} = (C_1, \cdots, C_L)$. Here, $\mathbf{1}^T\mathbf{g}$ is the sum of all gate values, which represents the number of remained channels, n is the total number of gates and $\mathbf{1}^T\mathbf{C}$ is the total number of channels. Note that not all layers are included in Eq. (5), and the vector $\mathbf{g}$ and $\mathbf{C}$ only contain layers are used for pruning. There are several remarks on this pruning objective: **1)** The pruning of channels only depends on the discriminate power of its own, channel magnitude is irrelevant during model pruning; **2)** There exists no layer-wise

hyper-parameter, only a global hyper-parameter is used to control pruning rate; **3)** During training of the parameters for the gate function, model parameters $\mathcal{W}$ are fixed.

Under this setting, the major advantage of the discrete gate setting is that the impact of pruned channels are precisely reflected in the value of loss function. If the gates are relaxed in continuous values, then it doesn't possess such good property. In addition, continuous relaxed gate function causes severe difficulty to solve the optimization problem defined in Eq. (5).

For simplicity, we replace the equality constraint with a regularization term and redefine the optimization problem as follows:

$$\min_{\Theta} \mathcal{F}(\Theta) := \mathcal{L}\big(f(x; \mathcal{W}, \Theta), y\big) + \lambda \mathcal{R}(\mathbf{1}^T\mathbf{g},\ p\mathbf{1}^T\mathbf{C}), \quad (6)$$

where $\mathcal{R}(\cdot,\cdot)$ is the specific regularization function used, which can be a typical regression loss function such as MAE or MSE. In practice, both MAE and MSE are not used, we will talk about the choice of $\mathcal{R}(\cdot,\cdot)$ later. The constraint $\mathbf{g} \in \{0,1\}^n$ is absorbed into the optimization problem due to the definition of discrete gate (Eq. (1) and (4)).

Recently, there are increasing interests in reducing the float point operations (FLOPs) in model pruning literature. The definition used in Eq. (6) along with the definition of deterministic discrete gate in Eq. (1) can easily transform the pruning rate constraint in Eq. (5) to FLOPs constraint. Recall that for a single convolution layer l with one sample, the FLOPs calculation can be down as follows:

$$(\text{FLOPs})_l = k_l \cdot k_l \cdot \frac{c_{l-1}}{\mathcal{G}_l} \cdot c_l \cdot w_l \cdot h_l, \quad (7)$$

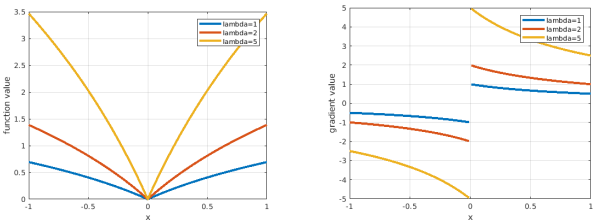
where k_l is the kernel size, $\mathcal{G}_l$ is the number of groups, c_{l-1} and c_l are the number of input and output channels, w_l and h_l are width and height, $(\text{FLOPs})_l$ is the FLOPs of l-th layer. By replacing c_l and c_{l-1} with $\mathbf{g}_l$ and $\mathbf{g}_{l-1}$, we get a new representation of FLOPs during searching for sub-networks:

$$\widehat{(\text{FLOPs})}_l = k_l \cdot k_l \cdot \frac{\mathbf{1}^T\mathbf{g}_{l-1}}{\mathcal{G}_l} \cdot \mathbf{1}^T\mathbf{g}_l \cdot w_l \cdot h_l, \quad (8)$$

Then combining Eq. (7), Eq. (8) with Eq. (6), the original $\mathcal{R}(\mathbf{1}^T\mathbf{g},\ p\mathbf{1}^T\mathbf{C})$ is replaced by the FLOPs regularzation:

$$\mathcal{R}(\hat{T},\ pT), \quad (9)$$

where $\hat{T} = \sum_{l=1}^{L} \widehat{(\text{FLOPs})}_l$ and $T = \sum_{l=1}^{L} (\text{FLOPs})_l$ are remained FLOPs and total FLOPs of the model. Note that we still use p as the only global hyper-parameter to represent the remaining fractions of the FLOPs. By incorporating FLOPs regularization and the objective in Eq. (6), we can prune the model to arbitrary level of FLOPs.



(a) function value of the regularization

(b) gradients of the regularization

Figure 2: The values and gradients of the resource regularization, y is set to 0 for better visualization.

3.4. Choice of the Regularization Loss

In order to train the parameters of gates properly, the value of the regularization term should be decreased to near 0 in the early stage of gate training, and remain near 0 for the rest of the training process. The reason we want to keep the regularization term near 0 for most time is to ensure the algorithm have enough time to discover the best possible sub-architecture with the given constraint. Regular regression loss like MAE and MSE can't satisfy this requirement, since their gradient is either constant or decreasing when close to 0. To overcome this issue, we propose the following regularization loss:

$$\mathcal{R}_{\log}(x, y) = \log(|x - y| + 1). \quad (10)$$

The plot of gradient and value of this function is shown in Fig. 2. The benefit of this function is that when x is close to target y, the gradient will increase and keep x close to the target value y. The loss function is not differentiable at the point $x = y$ by definition, but sub-gradient can be used here which has been implemented in major deep learning frameworks. In practice, the Eq. (10) works well for appropriate choice of λ, on the contrary, MAE and MSE often fail to keep the value of the regularization term close to 0.

3.5. Symmetric Weight Decay

To further expand search space, we propose a symmetric weight decay on the weights of gates, which is inspired by the subgradient of the regularization loss:

$$\frac{\partial \mathcal{R}_{\log}}{\partial \theta_{l,c}} = \begin{cases} \eta_l \cdot \frac{1}{|\hat{T}-pT|+1} \cdot \frac{\hat{T}-pT}{|\hat{T}-pT|}, & \text{if } \hat{T} \neq pT \\ 0, & \text{if } \hat{T} = pT \end{cases} \quad (11)$$

where $\eta_l = k_l^2 \cdot \frac{\mathbf{1}^T\mathbf{g}_{l-1}}{\mathcal{G}_l} \cdot w_l \cdot h_l$. Eq. (11) indicates that $\mathcal{R}_{\log}$ works like weight decay with different decay value for each layer while the decay value is also different for each training iteration. Since the impact of $R_{\log}$ on θ can be expressed by weight decay, the stochasticity of the gates can be increased in a similar way. Based on above arguments, we can explore

Algorithm 1: Discrete Model Compression

input: dataset for training gate, D_{gate}; remaining rate of FLOPs, p; regularization hyper-parameter, λ; symmetric weight decay parameter β; gate training epochs num-E; pre-trained model f.

Freeze $\mathcal{W}$ and batchnorm parameters in f.

Initialize all $\theta_{l,c}$ to 1.

for $e := 1$ *to num-E* **do**

  shuffle(D_{gate})

  foreach x, y *in* D_{gate} **do**

    1. forward calculation: $\min_{\Theta} \mathcal{F}(\Theta) = \mathcal{L}(f(x; \mathcal{W}, \Theta), y) + \lambda \mathcal{R}_{\log}$

    2. calculate gradient w.r.t $\theta_{l,c}$.

    3. update each $\theta_{l,c}$ by ADAM optimizer.

    4. apply symmetric weight decay on $\theta_{l,c}$ defined in Eq. (12).

    5. clip each $\theta_{l,c}$ to [0,1].

  end

end

return model f with the final Θ.

larger search spaces by applying symmetric weight decay on each $\theta_{l,c}$. The goal of doing this is to slow down the pace of gate parameters to become deterministic (approach 0 or 1). As a result, the search space is enlarged:

$$\theta_{l,c} = \theta_{l,c} - \beta \text{sign}(\theta_{l,c} - 0.5), \tag{12}$$

where β is the hyper-parameter to control the strength of weight decay.

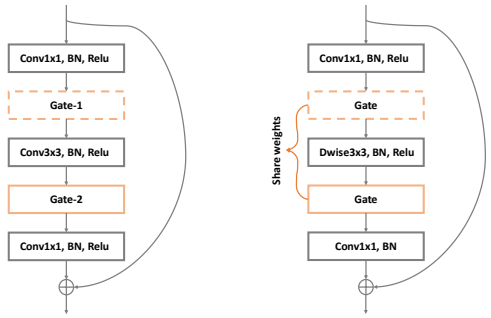
3.6. Our DMC Algorithm

We have introduced the core idea of our method, and the discrete model compression (DMC) algorithm is presented in Algorithm 1. During freezing trainable parameters, the batchnorm running statistics are also frozen. It should be emphasized again that the gate learning process is isolated from the training of the model parameters. In this way, we can prune any pre-trained models without modifications. In the calculation of $\mathcal{L}$, the sub-networks are drawn from the stochastic discrete gate. When calculating FLOPs regularization and during model pruning, the deterministic version of the gate is used. Both stochastic and deterministic calculation share the same $\theta_{l,c}$.

4. Experiments

4.1. Settings

Implementation Detail. We use CIFAR-10 [22] and ImageNet [5] to verify the performance of our method. Our method only requires one hyperparameter p to control the FLOPs budget. For all experiments, we use resource regularization with $\mathcal{R}_{\log}$ defined in Eq. (10). As a result, p decides how much FLOPs are preserved for each experiment. λ decides the regularization strength in our method. We choose $\lambda = 4$ in all CIFAR-10 experiments and $\lambda = 8$ for all ImageNet experiments. For CIFAR-10, we compare with other methods on ResNet-56 and MobileNetV2. For ImageNet, we select ResNet-34, ResNet50, ResNet101 and MobileNetV2 as our target models. The reason we choose these models is because that ResNet [9] models and MobileNetV2 [41] are much harder to prune than earlier models like AlexNet [23] and VGG [43]. For CIFAR-10 models, we train it from scratch following the code from PyTorch examples. After pruning the model, we finetune the model for 160 epochs using SGD with start learning rate 0.1, weight decay 0.0001 and momentum 0.8, the learning rate is multiplied by 0.1 at epoch 80 and 120. For ImageNet models, we directly use the pre-trained models released from pytorch [36]. After pruning, we finetune the model for 100 epochs using SGD with start learning rate 0.01, weight decay 0.0001 and momentum 0.9, and the learning rate is scaled by 0.1 at epoch 30, 60 and 90. For MobileNetV2 on ImageNet, we choose weight decay as 0.00004 which is usd in the original paper [41]. Both CIFAR-10 and ImageNet finetuning hyperparameters are similar to those used in Collaborative Channel Pruning (CCP) [37]. During gate training, we choose the β of symmetric weight decay (Eq. (12)) as 0.0001. We randomly choose $2,500$ and $10,000$ samples as the dataset for training gate (D_{gate}) for CIFAR-10 and ImageNet separately. We didn't create a standalone validation set for training gate in order to directly use pre-trained models. In the gate training process, we use ADAM [21] optimizer with a constant learning rate 0.001 and train gate parameters for 300 epochs. All the codes are implemented with pytorch [36]. The experiments are conducted on a machine with 4 Nvidia Tesla P40 GPUs.



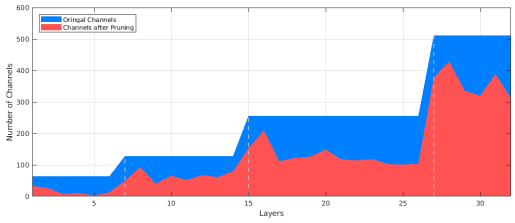
(a) Bottleneck Block (b) Inverted Residual Block

Figure 3: Gate placement for different architectures. (a) Bottleneck Block for ResNet. (b) Inverted Residual Block for MobileNetV2.

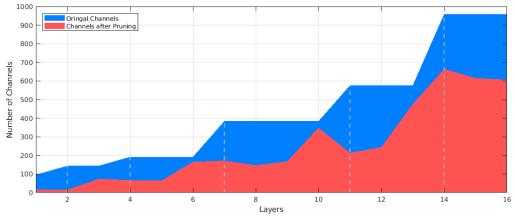
Placement of Gate. Where to put gates is a crucial problem to best approximate the output from actual sub-networks

Method	Architecture	Baseline Acc	Pruned Acc	Δ-Acc	Pruned FLOPs
Channel Pruning [13]	ResNet-56	92.80%	91.80%	-1.00%	50.0%
AMC [11]		92.80%	91.90%	-0.90%	50.0%
Pruning Filters [25]		93.04%	93.06%	+0.02%	27.6%
Soft Prunings [10]		93.59%	93.35%	-0.24%	52.6%
DCP [49]		93.80%	93.59%	-0.31%	50.0%
DCP-Adapt [49]		93.80%	93.81%	+0.01%	47.0%
CCP [37]		93.50%	93.42%	-0.08%	**52.6%**
DMC(ours)		93.62%	93.69%	**+0.07%**	50.0%
WM* [49]	MobileNetV2	94.47%	94.17%	-0.30%	26.0%
DCP [49]		94.47%	94.69%	+0.22%	26.0%
DMC(ours)		94.23%	94.49%	**+0.26%**	**40.0%**

Table 1: Comparison results on CIFAR-10 dataset with ResNet-56 and MobileNetV2. Δ-Acc represents the performance changes before and after model pruning. +/- indicates increase or decrease compared to baseline results. WM represents width multiplier used in original design of MobileNetV2, this result is from DCP [49] paper.



(a) ResNet-50



(b) MobileNetV2

Figure 4: Networks discovered by our method from ResNet-50 and MobileNetV2. Dashed line indicates channel number changes in the original model.

when corresponding channels are pruned. Following the settings in NAS works [27, 50], we regard Conv-BN-Relu as a complete block, thus the gates are always placed after Relu activation functions. To better simulate the results of a sub-network, we place two individual gates for a bottleneck block in a ResNet. For MobileNetV2, we place two gates in an inverted residual block, and the two gates share the same set of parameters due to the nature of depth-wise convolution. Details are shown in Fig. 3. Following the above settings, outputs from sampled sub-networks can well approximate the outputs from the actual compact network.

4.2. Results on CIFAR-10

In Tab. 1, we show all the comparison results on CIFAR-10. For ResNet-56, our method performs much better than early methods [13, 11, 25, 10]. Specially, when compared with Soft Pruning, our method is better than their results by 0.31% on Δ-Acc given similar pruned FLOPs (52.6% vs 50%). Discrimination-aware pruning utilizes local discrimination criterion when pruning the model. Our method outperforms DCP [49] by 0.38% on Δ-Acc with the same pruned FLOPs. Moreover, our method outperforms DCP-adapt (stronger version of DCP) by 0.06% give similar pruned FLOPs. Collaborative filter pruning [37] is one of the most recent works on channel pruning which considers the correlation between different weights when applying Taylor expansion on the loss function. Our method still outperforms their result by 0.15% on Δ-Acc. Such observation may indicate that our method also implicitly considers weights correlation during the search of optimal sub-network. For MobileNetV2, our method outperforms DCP by 0.04% on Δ-Acc, while pruning 14% more FLOPs than DCP. This shows that global discrimination-aware is better than local discrimination-aware.

4.3. Results on ImageNet

In Tab. 2, we presents all the comparison results on ImageNet. All results are adopted from their original papers except for ThiNet on MobileNetV2. To establish a high-quality baseline, the comparison methods are mainly chosen from recently published papers. Specially, DCP [49], CCP [37], IE [34], FPGM [12] and GAL [26] are from this category. Such high-quality baselines can help us better understand the benefit of using discrete channel settings.

For ResNet-34, our method can prune 43.4% FLOPs while only result in 0.73% and 0.31% performance drops on Top-1 accuracy and Top-5 accuracy separately. FPGM prunes slightly less FLOPs compared with our method (41.1% vs 43.3%), however, it causes larger damage to the final performance than our method (0.56% worse with Top-

Method	Architecture	Baseline Top-1 Acc	Baseline Top-5 Acc	Δ-Acc Top-1	Δ-Acc Top-5	Pruned FLOPs
Pruning Filters [25]	ResNet-34	73.23%	-	-1.06%	-	24.8%
Soft Prunings [10]		73.93%	91.62%	-2.09%	-1.92%	41.1%
IE [34]		73.31%	-	**-0.48%**	-	24.2%
FPGM [12]		73.92%	91.62%	-1.29%	-0.54%	41.1%
DMC(ours)		73.30%	91.42%	-0.73%	-0.31%	**43.4%**
Soft Pruning [10]	ResNet-50	76.15%	92.87%	-1.54%	-0.81%	41.8%
IE [34]		76.18%	-	-1.68%	-	45%
FPGM [12]		76.15%	92.87%	-1.32%	-0.55%	53.5%
GAL [26]		76.15%	92.87%	-4.35%	-2.05%	55.0%
DCP [49]		76.01%	92.93%	-1.06%	-0.61%	**55.6%**
CCP [37]		76.15%	92.87%	-0.94%	-0.45%	54.1%
DMC(ours)		76.15%	92.87%	**-0.80%**	**-0.38%**	55.0%
Rethinking [46]	ResNet-101	77.37%	-	-2.10%	-	47.0%
IE [34]		77.37%	-	-0.02%	-	39.8%
FPGM [12]		77.37%	93.56%	-0.05%	0.00%	41.1%
DMC(ours)		77.37%	93.56%	**+0.03%**	**+0.04%**	**56.0%**
ThiNet* [32]	MobileNetV2	70.11%	-	-6.40%	-4.60%	44.7%
DCP [49]		70.11%	-	-5.89%	-3.77%	44.7%
DMC(ours)		71.88%	90.29%	**-3.51%**	**-1.83%**	**46.0%**

Table 2: Comparison results on ImageNet dataset with ResNet-34, ResNet-50, ResNet-101 and MobileNetV2. Δ-Acc represents the performance changes before and after model pruning. +/- indicates increase or decrease compared to baseline results. ThiNet on MobileNetV2 results are from DCP [49] paper.

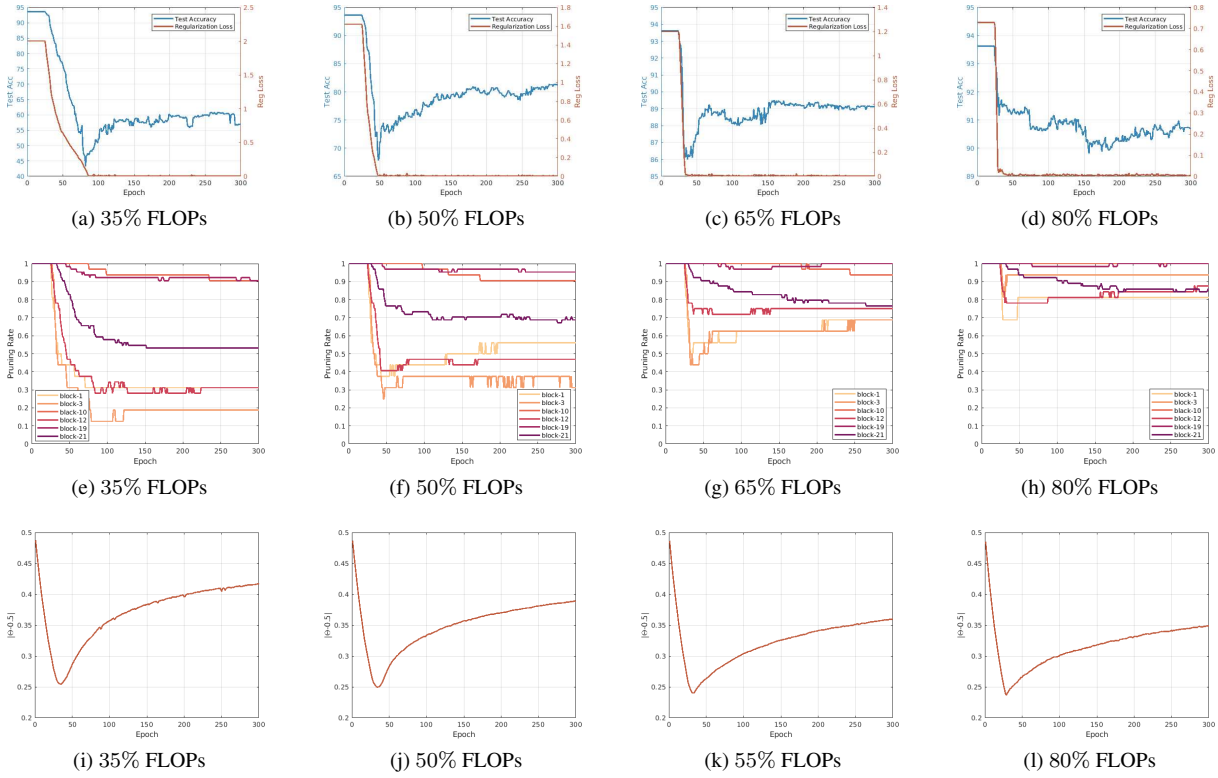
Gate Setting	Architecture	Top-1 Acc	Top-5 Acc	Pruned FLOPs
1-Gate	ResNet-50	75.06%	92.41%	55.2%
2-Gate		75.35%	92.49%	55.0%
1-Gate	MobileNetV2	67.73%	88.14%	45.3%
2-Gate-Shared		68.37%	88.46%	46.0%

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

1 accuracy and 0.23% worse with Top-5 accuracy). The other two methods only prune a small amount of FLOPs (around 25%). Our method has a lower Top-1 accuracy compared with IE, but we prune 1.8 times as much FLOPs as IE. For ResNet-50, our method achieves the best Δ-Acc Top-1 and Δ-Acc Top-5 accuracy compared with all other methods. Among all comparison methods, CCP has the smallest performance gap with our method. Specifically, our DMC algorithm outperforms the state-of-the-art pruning algorithm CCP by 0.14% at Top-1 accuracy with slightly more pruned FLOPs (55.0% vs 54.1%). For other comparison methods, our DMC algorithm has advantages on Top-1 accuracy varying from 0.26% to 3.55%. For ResNet-101, our method also achieves the best Δ-Acc Top-1 and Δ-Acc Top-5 accuracy. Moreover, our DMC algorithm prunes a much larger amount of FLOPs than all the other methods (56% vs 47% the second largest). After pruning 56% of FLOPs, the pruned ResNet-101 only has 3.43 GFLOPs which is even less than the vanilla ResNet-50 (4.09 GFLOPs). At such a high pruning rate for FLOPs, the performance of pruned ResNet-101 even increases by 0.03% and 0.04% for Top-1 and Top-5 accuracy respectively. For MobileNet-V2, our method largely increases the results obtained by DCP. With similar amount of pruned FLOPs (46% vs 44.7%), DMC outperforms DCP by 2.38% and 1.94% for Δ-Acc Top-1 and Δ-Acc Top-5 accuracy. Such significant improvement further shows that the global discrimination-aware pruning utilized in our method has obvious advantages when compared with local discrimination-aware pruning utilized in DCP. Global discrimination-aware is a direct consequence of discrete channel settings introduced in our algorithm. We further plot the channel configurations for each layer of ResNet-50 and MobileNetV2 after pruning in Fig. 4. Since early layers often have a large impact on FLOPs, most early layers are aggressively pruned. But there are some exceptions, layer-1 and layer-8 have a much higher preserved number of channels compared to adjacent layers in ResNet-50. Similar observations hold for layer-3 and layer-6 in MobileNetV2. These layers are regarded as important components in our method.

4.4. Impact of Gate Placement

In Tab. 3, we show how gate placement impacts the performance of the pruned model. 1-Gate represents removing gates with the dashed line in Fig 3. 2-Gate and 2-Gate-Shared are the original settings described in section 4.1. From this table, we can see that precise estimation of sub-network performance results in improvements in the final model ($+0.29\%$ on Top-1 for ResNet, $+0.64\%$ on Top-1 for MobileNetV2). This experiment clearly shows that the



(a) 35% FLOPs (b) 50% FLOPs (c) 65% FLOPs (d) 80% FLOPs

(e) 35% FLOPs (f) 50% FLOPs (g) 65% FLOPs (h) 80% FLOPs

(i) 35% FLOPs (j) 50% FLOPs (k) 55% FLOPs (l) 80% FLOPs

Figure 5: (a)-(d): Test set performance and regularization loss with the progress of gate training. (e)-(h): Pruning rate on different layers with the progress of gate training. (i)-(l): Evolution of gate randomness during the progress of gate training.

discrepancy between the precise and imprecise estimation of sub-network during gate training.

4.5. Understanding the Training of Discrete Gates

We draw related figures about the gate training process in Fig. 5. Four experiments are conducted given different values of p from 0.35 to 0.8 on CIFAR-10 with ResNet-56. Fig. 5 (a)-(d) show the test performance along with regularization loss. It's quite clear that there are two stages for gate training. In the first stage, the test accuracy of the sub-network sharply drops, at the same time, regularization loss drops too. In the second stage, regularization loss is near 0, the test performance continues to increase until the end of the training process. This shows that our DMC algorithm can consistently find sub-networks with better performance. Fig. 5 (e)-(h) show the progress of pruning rate for different layers. We can see the pruning rate of a layer dramatically decreases at the beginning. However, within the process of the second training stage, some pruned channels of certain layers are recovered. This observation shows that our method indeed explores the search space instead of stacking at a trivial solution with a fixed sub-network. In Fig.5 (i)-(l), we plot the value of $s = \frac{\sum_{l=1}^{L}\sum_{c=1}^{C_l}|\theta_{l,c}-0.5|}{n}$. This value can roughly measure whether the discrete gate is inclined to deterministic (θ is close to 0 or 1) or stochastic (θ is close to 0.5). At the beginning of training, s dramatically drops, depicting that increased stochasticity when sampling a sub-network. After training for a while, there is a turning point that the gates start to become more deterministic. This is mainly because that the information from the samples is utilized to reduce the stochasticity and make it more confident towards the final sub-network.

5. Conclusion

In this paper, we proposed an effective discrete model compression method to prune CNNs given certain resource constraints. By turning deterministic discrete gate to stochastic discrete gate, moreover, our method can explore larger search space of sub-networks. To further enlarge the space, we introduced the symmetric weight decay on the gate parameters inspired by the fact that regularization loss can be regarded as weight decay. Our method also benefits from the exact estimation of sub-networks' outputs because of a combination of the precise placement of gates and the discrete setting. Extensive experiments results on ImageNet and CIFAR-10 show that our method outperforms state-of-the-art methods.

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.

[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.

[3] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, Jonathan Eckstein, et al. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine learning*, 3(1):1–122, 2011.

[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.

[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.

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

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

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

[15] Feihu Huang, Songcan Chen, and Heng Huang. Faster stochastic alternating direction method of multipliers for nonconvex optimization. In *International Conference on Machine Learning*, pages 2839–2848, 2019.

[16] Feihu Huang, Shangqian Gao, Jian Pei, and Heng Huang. Nonconvex zeroth-order stochastic admm methods with lower function query complexity. *arXiv preprint arXiv:1907.13463*, 2019.

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

[18] 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.

[19] 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.

[20] Jaedeok Kim, Chiyoun Park, Hyun-Joo Jung, and Yoonsuck Choe. Plug-in, trainable gate for streamlining arbitrary neural networks. *CoRR*, abs/1904.10921, 2019.

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

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

[23] 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.

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

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

[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 Conference on Computer Vision and Pattern Recognition*, pages 2790–2799, 2019.

[27] Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: Differentiable architecture search. In *International Conference on Learning Representations*, 2019.

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

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

[30] Christos Louizos, Max Welling, and Diederik P. Kingma. Learning sparse neural networks through l_0 regularization. In *International Conference on Learning Representations*, 2018.

[31] Jian-Hao Luo and Jianxin Wu. Autopruner: An end-to-end trainable filter pruning method for efficient deep model inference. *arXiv preprint arXiv:1805.08941*, 2018.

[32] Jian-Hao Luo, Hao Zhang, Hong-Yu Zhou, Chen-Wei Xie, Jianxin Wu, and Weiyao Lin. Thinet: pruning cnn filters for a thinner net. *IEEE transactions on pattern analysis and machine intelligence*, 2018.

[33] 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.

[34] 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.

[35] 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.

[36] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[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, 2019.

[38] 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.

[39] 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.

[40] 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.

[41] 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.

[42] 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.

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

[44] Chaoqi Wang, Roger B. Grosse, Sanja Fidler, and Guodong Zhang. Eigendamage: Structured pruning in the kronecker-factored eigenbasis. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019*, pages 6566–6575, 2019.

[45] 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.

[46] Jianbo Ye, Xin Lu, Zhe Lin, and James Z Wang. Rethinking the smaller-norm-less-informative assumption in channel pruning of convolution layers. In *International Conference on Learning Representations*, 2018.

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

[48] Tianyun Zhang, Shaokai Ye, Kaiqi Zhang, Jian Tang, Wujie Wen, Makan Fardad, and Yanzhi Wang. A systematic dnn weight pruning framework using alternating direction method of multipliers. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 184–199, 2018.

[49] 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.

[50] Barret Zoph and Quoc V. Le. Neural architecture search with reinforcement learning. 2017.