Operation-Aware Soft Channel Pruning using Differentiable Masks

Minsoo Kang [1] **Bohyung Han** [1]

Abstract

We propose a simple but effective data-driven channel pruning algorithm, which compresses deep neural networks in a differentiable way by exploiting the characteristics of operations. The proposed approach makes a joint consideration of batch normalization (BN) and rectified linear unit (ReLU) for channel pruning; it estimates how likely the two successive operations deactivate each feature map and prunes the channels with high probabilities. To this end, we learn differentiable masks for individual channels and make soft decisions throughout the optimization procedure, which facilitates to explore larger search space and train more stable networks. The proposed framework enables us to identify compressed models via a joint learning of model parameters and channel pruning without an extra procedure of fine-tuning. We perform extensive experiments and achieve outstanding performance in terms of the accuracy of output networks given the same amount of resources when compared with the state-of-the-art methods.

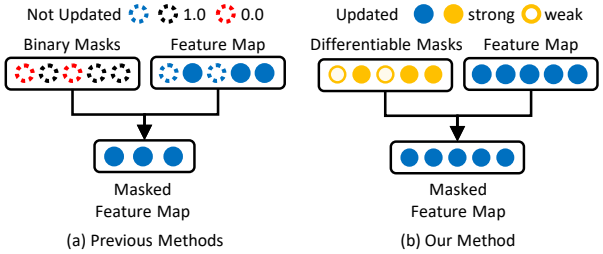


Figure 1. Illustration of the difference between the previous channel pruning methods and our approach. The previous methods often make hard decision for channel pruning and eliminate pruned channel permanently while our algorithm trains model parameters and masks jointly via soft channel pruning. Note that the pruned networks identified by the previous methods should be fine-tuned later but it may not be effective to achieve competitive accuracy because the pruned models are obtained from poor local optima due to greedy decisions during training procedure. On the other hand, the proposed method achieves good performance without separate procedure of fine-tuning because we learn the model parameters and the pruning masks in a unified way based on a differentiable optimization framework.

1. Introduction

Deep neural networks have achieved state-of-the-art performance on various visual recognition tasks including image classification (He et al., 2016a; Huang et al., 2017; Tan & Le, 2019), image segmentation (Noh et al., 2015; Chen et al., 2018), object detection (He et al., 2017a; Li et al., 2019), and visual tracking (Nam & Han, 2016). However, despite their outstanding accuracy, the applicability of the models based on deep neural networks to resource-hungry systems, *e.g.*, mobile or portable devices, is still limited due to their computational or physical costs in terms of model sizes, FLOPs, and power consumption. Consequently, model compression, a research problem to reduce the sizes of deep neural networks, has been investigated actively in the community.

There is a large volume of research related to the compression of deep neural networks. Early methods often rely on mathematical analysis of weight matrices such as matrix decomposition (Denton et al., 2014; Jaderberg et al., 2014; Tai et al., 2016). These approaches focus on reducing FLOPs of neural networks using the low-rank factorizations of pretrained weight matrices or tensors. Another line of research in model compression is network quantization (Courbariaux et al., 2015; Han et al., 2016; Rastegari et al., 2016; Zhu et al., 2017; Polino et al., 2018), which learns the low-bit representations of model parameters and is often employed for hardware integration of deep neural networks. On the other hand, pruning approaches (Han et al., 2015; Molchanov et al., 2017; Wen et al., 2016; Li et al., 2017; Luo et al., 2017; Liu et al., 2017; He et al., 2017b; 2018; Zhao et al., 2019; He et al., 2019) remove unnecessary weights or activations through heuristics, optimization techniques, or learning processes.

We are interested in a structured pruning technique that removes channels from deep neural networks. Most of the ex-

[1]Computer Vision Laboratory, Department of Electrical and Computer Engineering & ASRI, Seoul National University, Korea. Correspondence to: Bohyung Han <bhhan@snu.ac.kr>.

Proceedings of the 37^{th} International Conference on Machine Learning, Online, PMLR 119, 2020.

isting channel pruning methods rely on the two-step process for model compression—pruning followed by fine-tuning as illustrated in Figure 1(a). Since, at the time of fine-tuning, the networks are already pruned and have smaller capacity, the models may converge to poor local minima. Moreover, the channel pruning decision of the previous methods often depends only on the absolute values of activations in the corresponding channels, but it fails to consider the natural consequences happening inside deep neural networks; negative activations will be discarded after passing rectified linear units (ReLUs) regardless of their magnitudes.

To deal with these issues, we incorporate training and pruning stages by introducing learnable mask parameters instead of identifying static masks. To be specific, the model parameters and the pruning masks are learned jointly by a gradient-based optimization method, where the original capacity of the networks is preserved during training as illustrated in Figure 1(b). Furthermore, our framework considers the distribution of the activations in a channel together with the operations (batch normalization and ReLU) to be applied to the channel, and safely removes the channels that most of the values in the feature map are near zero or negative. To this end, we propose a probabilistic formulation to identify such cases and estimate how likely individual channels are to be pruned by the criteria.

The main contributions of the proposed algorithm are summarized below:

- We propose a novel framework of channel pruning for deep neural network compression, which jointly optimizes model parameters and masks by employing a differentiable formulation.
- We introduce a simple but effective channel pruning strategy, which estimates the importance of each channel probabilistically using the distribution of activations in accordance with the network operations.
- Experimental results show that the proposed approach outperforms the previous structured pruning methods. We also provide the empirical evidence that each component is helpful for improving accuracy.

The rest of the paper is organized as follows. Section 2 discusses the related work to model compression. The details of our approach is described in Section 3, and the experimental results are presented in Section 4. Finally, we conclude this paper in Section 5.

2. Related Work

The proposed technique is one of the model compression algorithms, and aims to prune the channels that are unlikely to make a critical impact on the accuracy of the network. This section first describes three types of model compression algorithms and then discusses several closely related works to our approach.

2.1. Matrix Decomposition

The main goal of matrix decomposition methods is to reduce computational cost in a deep neural network by approximating weight matrices. (Denton et al., 2014) approximates convolution filter weights to low-rank tensors by applying singular value decompositions. (Jaderberg et al., 2014) minimizes reconstruction error between the pretrained weights of original filters and a linear combination of basis filters while penalizing a nuclear norm for the approximated filters. On the other hand, (Tai et al., 2016) reduces the reconstruction error of filters by minimizing the Frobenius norm of the difference between pretrained filters and approximated ones with a rank constraint, where they find a closed form solution.

2.2. Network Quantization

Network quantization techniques reduce the precision of weights to accelerate deep neural networks. (Courbariaux et al., 2015) learns a network with binary weights, where real-valued gradients are employed to update the binarized weights. (Zhu et al., 2017) quantizes weights to $\{-W_l^n, 0, W_l^p\}$ using learnable parameters, W_l^n and W_l^p, which leads to a significantly smaller model with little accuracy drop. In (Polino et al., 2018), the authors perform a model compression using quantization and knowledge distillation, where the quantization points are optimized through backpropagation. Even though network quantization methods are effective to reduce computational cost conceptually, they require additional efforts in low-level processing to design practical systems.

2.3. Pruning

Weight pruning methods eliminate unnecessary connections based on heuristics or optimization processes. Optimal Brain Damage (LeCun et al., 1990) removes weight parameters based on the Hessian matrix of objective function and fine-tunes the updated network. (Han et al., 2015) prunes unimportant weights from a pretrained network based on the magnitude of the weights and retrain the model iteratively. (Molchanov et al., 2017) proposes an extension of Variational Dropout (Kingma et al., 2015), which addresses the challenge in training with high dropout rates and prunes the weights whose dropout rate is above a threshold. Although weight pruning methods are successful in reducing a large number of connections, the resulting networks tend to have unstructured random connectivity, which leads to irregular memory access and little gain in actual inference speed without a proper hardware specialization as discussed

in (Wen et al., 2016).

Contrary to such pruning methods, structured pruning approaches (Wen et al., 2016; Li et al., 2017; Luo et al., 2017; Liu et al., 2017; He et al., 2017b; Ye et al., 2018; He et al., 2018; Huang & Wang, 2018; Zhao et al., 2019; He et al., 2019) aim to reduce redundant filters, channels or layers and improve the actual inference time without the need of special library or hardware support. (Li et al., 2017) removes unnecessary filters in a pretrained network based on their ℓ_1 norms and fine-tunes the whole network after pruning. Soft Filter Pruning (SFP) (He et al., 2018) resets less important filters at every epoch while updating all filters including the reset ones to identify to-be-pruned channels. Sparse Structure Selection (SSS) (Huang & Wang, 2018) introduces scaling factors to prune specific structures such as neurons or residual blocks by adding sparsity regularizations on the structures. Filter Pruning via Geometric Median (FPGM) (He et al., 2019) removes the filters minimizing the sum of distances to others instead of discarding the filters with negligible weights, as opposed to (Li et al., 2017; He et al., 2018). On the other hand, (Zhao et al., 2019) provides a Bayesian model compression technique, which approximates BN scaling parameters to a fully factorized normal distribution using stochastic variational inference (Kingma & Welling, 2014) and then prunes the channels that have smaller mean and variance of the variational distribution.

2.4. Channel Pruning without Extra Fine-Tuning

The proposed algorithm is closely related to SFP (He et al., 2018), SSS (Huang & Wang, 2018), FPGM (He et al., 2019), and Variational Pruning (Zhao et al., 2019) in the sense that they prune filters or channels without an extra fine-tuning stage. Variational Pruning permanently removes some channels based on the predefined criteria at every epoch but such a greedy strategy may lead to a local optimum. SFP and FPGM reset all the weights in unnecessary filters to zeros during training instead of completely eliminating the filters for their potential needs in the later stage of training. However, the abrupt changes of the filter values make the training procedures unstable. Both our algorithm and SSS introduce learnable masks and trains the masks and model parameters jointly. The main difference between the two approaches is that ours estimates the masks based on BN and ReLU while SSS directly parameterizes and optimizes the masks by enforcing them to be zeros.

3. Proposed Method

This section describes our probabilistic framework of joint channel pruning and parameter optimization with differentiable deep neural network architectures in detail.

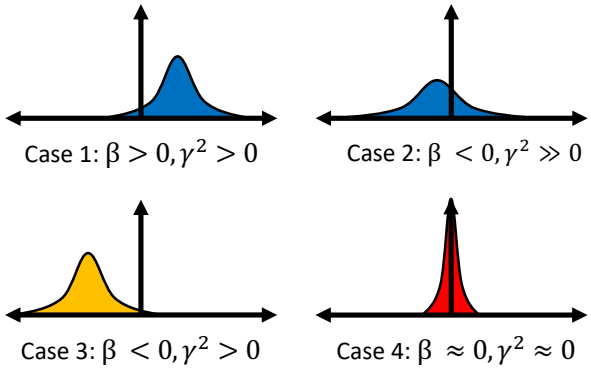


Figure 2. Illustration of the difference between the proposed method and the previous work (Liu et al., 2017). In case 1 and 2, both approaches would not prune the channel due to their BN statistics. Our algorithm and (Liu et al., 2017) will make a different decision for case 3; the proposed method prunes the channel because most of the activations will be zeros after applying a ReLU function. The channel shown in case 4 will probably be pruned by both methods because the activations have low variance.

3.1. Preliminary

Many recent deep neural networks (He et al., 2016a;b; Huang et al., 2017) adopt multiple identical building blocks that contain a batch normalization (BN) layer (Ioffe & Szegedy, 2015) followed by a rectified linear unit (ReLU) layer. Originally, the BN layer has been designed to accelerate convergence and facilitate stable training of deep neural networks. The main idea of BN is to prevent the input feature distribution in each layer from fluctuating despite the inherent variations of data representations across mini-batches. For the purpose, an output x^{out} of a BN layer is calculated by normalizing an input x^{in} and then performing an affine transformation as follows:

$$z = \frac{x^{\text{in}} - \hat{\mu}}{\sqrt{\hat{\sigma}^2 + \epsilon}} \tag{1}$$

$$x^{\text{out}} = \gamma \cdot z + \beta \tag{2}$$

where ϵ is a small positive value for numerical stability, and γ and β are learnable affine parameters in the BN layer. Note that $\hat{\mu} \equiv \text{E}[x^{\text{in}}]$ and $\hat{\sigma}^2 \equiv \text{Var}[x^{\text{in}}]$ are given by respectively calculating the sample mean and variance sequentially across mini-batches in the training stage while they are fixed during testing. By employing the BN layer, we assume that z follows a normal distribution, which makes x^{out} normally distributed with mean β and variance γ^2.

3.2. Mask for Channel Pruning

(Liu et al., 2017) proposes a channel pruning method using BN layers, where they prune channels from a pretrained

network if the learned scaling parameter γ is less than a predefined threshold. This strategy is motivated by the fact that a channel such that the output x^{out} is very close to a constant β can be removed safely. However, the performance gap between a pretrained network and the corresponding pruned network is often too high, which is partly because the method ignores the shifting parameters β totally and relies on a heuristic hard channel pruning strategy as mentioned in (Zhao et al., 2019).

The proposed algorithm considers both the channel scaling and shifting parameters for channel pruning. We claim that a feature map after a BN layer can be removed with low risk if most of its output activations are negative and then zeroed out by a subsequent ReLU layer eventually. Figure 2 illustrates the difference between the prior work (Liu et al., 2017) and the proposed approach; the channel pruning decision in (Liu et al., 2017) is based only on the γ while our algorithm utilizes both the learnable parameters in a BN layer, β and γ. Note that the criteria in our approach take advantage of a subsequent ReLU layer and have low risk for pruning.

We define a mask variable $m(\delta;\beta,\gamma)$ given the predefined thresholds, δ and c, as

$$m(\delta;\beta,\gamma) = \begin{cases} 0 & \text{if } \Phi(\delta;\beta,\gamma) \geq c \\ 1 & \text{otherwise} \end{cases}, \tag{3}$$

where $\Phi(\delta;\beta,\gamma)$ denotes the cumulative density function (CDF) of a Gaussian distribution $f(t;\beta,\gamma)$ parametrized by β and γ, which is given by

$$\begin{aligned}\Phi(\delta;\beta,\gamma) &= \int_{-\infty}^{\delta} f(t;\beta,\gamma)\,dt \\ &= \int_{-\infty}^{\delta} \frac{1}{\sqrt{2\pi\gamma^2}} \exp\left(-\frac{(t-\beta)^2}{2\gamma^2}\right) dt. \end{aligned} \tag{4}$$

The mask variable is not differentiable with respect to β and γ, and the next subsection discusses how to make the function differentiable and how to optimize the masks and the model parameters jointly.

3.3. Soft Channel Pruning

Our goal is to perform a joint optimization of model parameters and pruning masks by designing a differentiable deep neural network. The model parameters can be updated by a gradient-based method trivially. However, to learn the mask by the standard backpropagation, one can replace the mask function in (3), which is based on an indicator function, by a logistic function, $q(\cdot)$, for its continuous relaxation as

$$q(\delta;\beta,\gamma) = \frac{1}{1+\exp(-k(\Phi(\delta;\beta,\gamma)-c))}, \tag{5}$$

where k is a constant. Note that the logistic function becomes identical to the indicator function in (3) when k approaches to infinity. The partial derivative of the logistic function with respect to $\Phi(\delta;\beta,\gamma)$ is given by

$$\frac{\partial q(\delta;\beta,\gamma)}{\partial \Phi(\delta;\beta,\gamma)} = k \cdot q(\delta;\beta,\gamma) \cdot (1 - q(\delta;\beta,\gamma)), \tag{6}$$

which implies that, if k is getting larger, the gradient vanishes over a wide range because $q(\delta;\beta,\gamma)$ moves toward either 0 or 1 quickly. Instead of setting k to a large value, we consider a moderate value for k to avoid the vanishing gradient problem. Also, $q(\delta;\beta,\gamma)$ can be viewed as a probability mask to estimate how likely the corresponding channel is deactivated after going through a ReLU function. Note that $m(\cdot)$ is the *discrete* mask function adopted for hard pruning at test time, which potentially has a large discrepancy from the probabilistic *continuous* mask function $q(\cdot)$ employed during training time.

Directly sampling from a Bernoulli distribution is a reasonable option to tackle the issue, but the sampling procedure is not differentiable. So, we employ the Gumbel-Softmax (Jang et al., 2017) trick, which performs a differentiable sampling to approximate to a categorical random variable. Then, we define $n(\cdot)$ using the Gumbel-Softmax trick as

$$\begin{aligned} &n(\delta;\beta,\gamma) \\ &= \frac{\exp((\log\pi_1 + g_1)/\tau)}{\exp((\log\pi_1 + g_1)/\tau) + \exp((\log\pi_0 + g_0)/\tau)}, \end{aligned} \tag{7}$$

where g_0 and g_1 are samples drawn from Gumbel$(0,1)$ distribution, and π_1 and π_0 are given by $1 - q(\delta;\beta,\gamma)$ and $q(\delta;\beta,\gamma)$, respectively. Note that the output of $n(\cdot)$ becomes identical to a Bernoulli sample as τ approaches to 0. To exploit the sample $n(\cdot)$ for training, the proposed algorithm revises a BN output x^{out} to

$$z = \frac{x^{\text{in}} - \hat{\mu}}{\sqrt{\hat{\sigma}^2 + \epsilon}}, \tag{8}$$

$$x^{\text{out}} = (\gamma \cdot z + \beta) \cdot n(\delta;\beta,\gamma). \tag{9}$$

In the revised BN layer, the partial derivatives of x^{out} with respect to β and γ are calculated as follows:

$$\frac{\partial x^{\text{out}}}{\partial \gamma} = z \cdot n(\delta;\beta,\gamma) + (\gamma \cdot z + \beta) \cdot \frac{\partial n(\delta;\beta,\gamma)}{\partial \gamma} \tag{10}$$

$$\frac{\partial x^{\text{out}}}{\partial \beta} = n(\delta;\beta,\gamma) + (\gamma \cdot z + \beta) \cdot \frac{\partial n(\delta;\beta,\gamma)}{\partial \beta}, \tag{11}$$

where

$$\frac{\partial n(\delta;\beta,\gamma)}{\partial \gamma} = \frac{\partial n(\delta;\beta,\gamma)}{\partial q(\delta;\beta,\gamma)} \cdot \frac{\partial q(\delta;\beta,\gamma)}{\partial \Phi(\delta;\beta,\gamma)} \cdot \frac{\partial \Phi(\delta;\beta,\gamma)}{\partial \gamma}$$

$$\frac{\partial n(\delta;\beta,\gamma)}{\partial \beta} = \frac{\partial n(\delta;\beta,\gamma)}{\partial q(\delta;\beta,\gamma)} \cdot \frac{\partial q(\delta;\beta,\gamma)}{\partial \Phi(\delta;\beta,\gamma)} \cdot \frac{\partial \Phi(\delta;\beta,\gamma)}{\partial \beta}$$

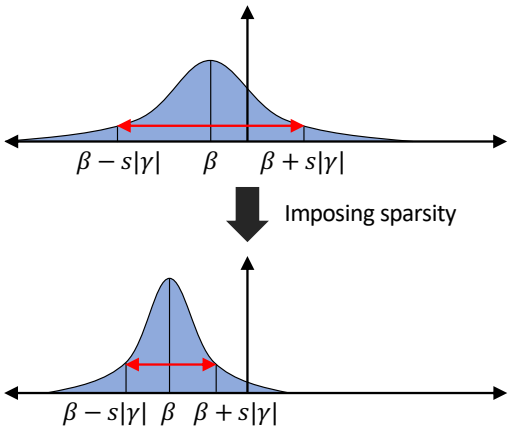


Figure 3. Illustration of confidence intervals in normal distributions, where β and γ^2 are mean and variance of the normal distribution, respectively, and s is a constant. The proposed sparsity regularization loss induces a larger value of CDF, $\Phi(\delta;\beta,\gamma)$ at the same threshold.

The derivatives of the above two terms also lead to the following equations:

$$\frac{\partial n(\delta;\beta,\gamma)}{\partial q(\delta;\beta,\gamma)} = -\frac{n(\delta;\beta,\gamma)(1-n(\delta;\beta,\gamma))}{\tau q(\delta;\beta,\gamma)(1-q(\delta;\beta,\gamma))} \tag{12}$$

$$\frac{\partial \Phi(\delta;\beta,\gamma)}{\partial \gamma} = -f(\delta;\beta,\gamma)\cdot\frac{\delta-\beta}{|\gamma|}\cdot\frac{\partial|\gamma|}{\partial\gamma} \tag{13}$$

$$\frac{\partial \Phi(\delta;\beta,\gamma)}{\partial \beta} = -f(\delta;\beta,\gamma), \tag{14}$$

where $\frac{\partial q(\delta;\beta,\gamma)}{\partial \Phi(\delta;\beta,\gamma)}$ is given by (6). Note that $\frac{\partial \Phi(\delta;\beta,\gamma)}{\partial \gamma}$ in (13) is differentiable almost everywhere except for 0. These equations illustrate that β and γ can be learned by the standard backpropagation.

3.4. Sparsity Loss with Confidence Interval

We have formulated the differentiable masks in Section 3.2 and 3.3, and the remaining question is how to define the loss function that makes networks compact by employing the differentiable masks. We impose a sparsity regularization of channels using the mask function in (3), which enforces $\Phi(\delta;\beta,\gamma) \geq c$ for each channel. The loss for the sparsity is given by

$$\mathcal{L}_{\text{sparse}}(\mathbb{B},\mathbb{C}) = \sum_{\beta_j\in\mathbb{B},\gamma_j\in\mathbb{C}} \beta_j + s|\gamma_j|, \tag{15}$$

where $\mathbb{B}$ and $\mathbb{C}$ are the sets of entire affine parameters in the BN layer in the original network and s is a predefined constant. This loss is motivated by the confidence interval of the normal distribution, which is illustrated in Figure 3. Since δ is fixed, the sparsity loss maximizes the CDF, $\Phi(\delta;\beta,\gamma)$. Thus, the optimization based on this loss encourages the network to be more sparse.

As mentioned earlier, our algorithm performs a joint optimization of the model parameters and the pruning masks. The full loss function $\mathcal{L}$ is expressed by a linear combination of the task-specific loss (classification loss in our paper) $\mathcal{L}_{\text{cls}}$ and the sparsity loss $\mathcal{L}_{\text{sparse}}$, which is given by

$$\mathcal{L}(\mathbf{W},\mathbb{B},\mathbb{C}) = \mathcal{L}_{\text{cls}}(\mathbf{W},\mathbb{B},\mathbb{C}) + \lambda\,\mathcal{L}_{\text{sparse}}(\mathbb{B},\mathbb{C}), \tag{16}$$

where $\mathbf{W}$ is a set of all parameters in the network excluding the affine parameters in BN layers. The optimization based on the loss function is straightforward because the whole network is differentiable. Note that we introduce no additional parameters to learn the masks. Instead, the mask function in (5) is determined by the parameters in BN layers, β and γ. Therefore, the gradient flow in the backward process for training is similar to the one in the ordinary deep neural networks with BN layers except that our approach performs the optimization of the model parameters and the masks jointly.

After training, we will obtain a deep neural network model for the target task (*i.e.*, classification) and a set of binary masks by thresholding CDF values based on $\mathbb{B}$ and $\mathbb{C}$. The final network is given by applying the masks to the individual channels of the network. Note that since the model parameters and pruning masks are learned at the same time, our algorithm does not require a separate fine-tuning stage followed by channel pruning.

4. Experiments

This section first presents details of datasets and our implementation, and then discusses the performance of the proposed algorithm in comparison to existing methods. We also analyze the characteristics of our approach via various ablative experiments.

4.1. Dataset

We employ CIFAR-10/100 and ILSVRC-12 in our experiment, which are the datasets widely accepted for evaluation of model compression techniques. The CIFAR-10/100 datasets consist of 50K and 10K color image splits for training and testing, where the size of each image is 32×32. The only difference between the two editions is the number of classes; CIFAR-10 and CIFAR-100 have 10 and 100 classes, respectively. On the other hand, the ILSVRC-12 dataset (Russakovsky et al., 2015) contains 1,281,167 training images in color and 50,000 color images for validation in 1,000 classes. For preprocessing images, we follow the techniques in (Liu et al., 2017) and (He et al., 2019) for CIFAR-10/100 and ILSVRC-12, respectively, to make fair comparisons.

Table 1. Performance comparison between the proposed algorithm, denoted by SCP, and other methods on CIFAR-10. "FT" indicates whether the pruned network identified by each algorithm performs fine-tuning or not. "Baseline Acc." and "Acc." mean the accuracies of the original and pruned networks, respectively. "Acc. Drop" represents the amount of accuracy degradation of the pruned networks with respect to the unpruned models. "Channels ↓", "Param. ↓", and "FLOPs ↓" denote the relative reductions in individual metrics compared to the unpruned networks. A bold-faced number indicates the best performance in each category while "–" denotes that the number is not available.

Method	Model	FT	Baseline Acc. (%)	Acc. (%)	Acc. Drop	Channels ↓ (%)	Param. ↓ (%)	FLOPs ↓ (%)
SFP (He et al., 2018)	ResNet-56	X	93.59	92.26	1.33	40	–	**52.60**
FPGM (He et al., 2019)	ResNet-56	X	93.59	92.93	0.66	40	–	**52.60**
SCP (Ours)	ResNet-56	X	93.69	93.23	**0.46**	**45**	48.47	51.50
Slimming (Liu et al., 2017)	DenseNet-40	O	94.39	92.59	1.80	80	73.53	68.95
Slimming (Liu et al., 2017)	DenseNet-40	X	94.39	12.78	81.61	80	73.53	68.95
Variational Pruning (Zhao et al., 2019)	DenseNet-40	X	94.11	93.16	0.95	60	59.76	44.78
SCP (Ours)	DenseNet-40	X	94.39	93.77	**0.62**	**81**	**75.41**	**70.77**
Slimming (Liu et al., 2017)	VGGNet-19	O	93.84	93.21	0.63	80	93.10	63.39
Slimming (Liu et al., 2017)	VGGNet-19	X	93.84	10.00	83.84	80	93.10	63.39
SCP (Ours)	VGGNet-19	X	93.84	93.82	**0.02**	**81**	**95.21**	**74.06**
Slimming (Liu et al., 2017)	VGGNet-16	O	93.85	92.91	0.94	70	87.97	48.12
Slimming (Liu et al., 2017)	VGGNet-16	X	93.85	10.00	83.85	70	87.97	48.12
Variational Pruning (Zhao et al., 2019)	VGGNet-16	X	93.25	93.18	0.07	62	73.34	39.10
SCP (Ours)	VGGNet-16	X	93.85	93.79	**0.06**	**75**	**93.05**	**66.23**

4.2. Implementation Details

The proposed method is implemented using TensorFlow library (Abadi et al., 2015). We train the network using SGD with Nesterov momentum (Sutskever et al., 2013) 0.9, weight decay parameter 0.0001, and initial learning rate 0.1. The setting for the experiment on the CIFAR datasets follows the one used in (Liu et al., 2017), where the batch size is set to 64, and the learning rate is reduced by the factor of 10 after the 80^{th} and 120^{th} epochs. For the ILSVRC-12 dataset, the network is learned for 100 epochs with 4 GPUs, where the total batch size is 256 and the learning rate is dropped by a factor of 10 at the 30^{th}, 60^{th}, and 90^{th} epochs. Fine-tuning the pruned network is based on the same setting except for the initial learning rate of 0.01. We set the temperature parameter τ for Gumbel-Softmax (Jang et al., 2017) to 0.5 and the threshold δ for CDF to 0.05 for all pruned models. All networks are trained from scratch.

4.3. Results on CIFAR-10/100 Datasets

We evaluate the performance of the proposed algorithm, denoted by SCP (Soft Channel Pruning), on the CIFAR-10/100 datasets using ResNet (He et al., 2016a), DenseNet (Huang et al., 2017), and VGGNet (Simonyan & Zisserman, 2015) since they are widely used for image recognition tasks. We compare our framework with Slimming (Liu et al., 2017) and Variational Pruning (Zhao et al., 2019), which also take advantage of BN layers to prune redundant channels. For CIFAR-10, we also compare the proposed method with SFP (He et al., 2018) and FPGM (He et al., 2019), which do not require fine-tuning like our algorithm. The results of Slimming are given by our reproduction from TensorFlow implementation.

CIFAR-10 Table 1 presents that the proposed approach, SCP, achieves the lowest accuracy drops compared to SFP, FPGM, Slimming, and Variational Pruning in all tested scenarios regardless of backbone networks. Also, the pruned models given by SCP based on DenseNet-40, VGGNet-19, and VGGNet-16 outperform the ones identified by Slimming with fine-tuning by 1.18%, 0.61%, and 0.88% points, respectively, although our models do not go through separate fine-tuning procedures. As mentioned in Section 3.2, Slimming suffers from significant accuracy drops without fine-tuning because of the ignorance of the shift parameter β in BN layers and the heuristic channel removal incurring error propagation over the network. In the case of VGGNet-19 and VGGNet-16, the pruned models identified by our algorithm have the almost same accuracy with the baselines while achieving substantial speed-ups. In addition, our method presents comparable or superior accuracy compared to the method based on Variational Pruning even with significantly smaller network sizes when DensNet-40 and VGGNet-16 are used as backbone networks.

CIFAR-100 Table 2 shows that our models given by SCP achieve less accuracy drop compared to the pruned networks identified by Slimming and Variational Pruning in most cases. Specifically, in the case of DenseNet-40 and VGGNet-16, our algorithm outperforms Variational Pruning by 1.95% and 0.28% points, respectively, even though our models are more compact. In addition, SCP achieves outstanding performance compared to Slimming on ResNet-164, DenseNet-40, and VGGNet-16 in terms of accuracy drop while the proposed method is comparable to Slimming on VGGNet-19. To demonstrate the effectiveness of our algorithm further, we also present the results with high

Table 2. Performance of our algorithm, SCP, with respect to Slimming and Variational Pruning on CIFAR-100.

Method	Model	FT	Baseline Acc. (%)	Acc. (%)	Acc. Drop	Channels ↓ (%)	Param. ↓ (%)	FLOPs ↓ (%)
Slimming (Liu et al., 2017)	ResNet-164	O	77.24	74.52	2.72	**60**	**29.26**	**47.92**
SCP (Ours)	ResNet-164	X	77.24	76.62	**0.62**	57	28.89	45.36
Slimming (Liu et al., 2017)	DenseNet-40	O	74.24	73.53	0.71	**60**	54.99	**50.32**
Variational Pruning (Zhao et al., 2019)	DenseNet-40	X	74.64	72.19	2.45	37	37.73	22.67
SCP (Ours)	DenseNet-40	X	74.24	73.84	**0.40**	**60**	**55.22**	46.25
Slimming (Liu et al., 2017)	VGGNet-19	O	72.56	73.01	**-0.45**	50	76.47	38.23
SCP (Ours)	VGGNet-19	X	72.56	72.99	-0.43	**51**	**77.52**	**40.92**
Slimming (Liu et al., 2017)	VGGNet-16	O	73.51	73.45	0.06	40	66.30	27.86
Variational Pruning (Zhao et al., 2019)	VGGNet-16	X	73.26	73.33	-0.07	32	37.87	18.05
SCP (Ours)	VGGNet-16	X	73.51	73.86	**-0.35**	**52**	**80.14**	**51.45**

Table 3. Performance of our algorithm, SCP, with respect to Slimming on CIFAR-100 with high pruning ratios.

Method	Model	FT	Baseline Acc. (%)	Acc. (%)	Acc. Drop	Channels ↓ (%)	Param. ↓ (%)	FLOPs ↓ (%)
Slimming (Liu et al., 2017)	ResNet-164	O	77.24	71.54	5.70	70	40.72	62.29
SCP (Ours)	ResNet-164	X	77.24	75.05	**2.19**	**71**	**53.30**	**64.93**
Slimming (Liu et al., 2017)	DenseNet-40	O	74.24	70.97	3.27	**80**	73.62	**68.20**
SCP (Ours)	DenseNet-40	X	74.24	73.17	**1.07**	**80**	**74.86**	67.82
Slimming (Liu et al., 2017)	VGGNet-19	O	72.56	67.81	4.75	**70**	89.10	59.67
SCP (Ours)	VGGNet-19	X	72.56	72.15	**0.41**	67	**89.37**	**61.94**
Slimming (Liu et al., 2017)	VGGNet-16	O	73.51	65.22	8.29	70	92.46	**80.49**
SCP (Ours)	VGGNet-16	X	73.51	69.96	**3.55**	**72**	**94.01**	79.24

Table 4. Performance of our algorithm, SCP, with respect to SSS and FPGM on ILSVRC-12 using ResNet-50.

Method	FT	Baseline Top-1 Acc. (%)	Baseline Top-5 Acc. (%)	Top-1 Acc. Drop	Top-5 Acc. Drop	FLOPs ↓ (%)
FPGM (He et al., 2019)	O	76.15	92.87	1.32	**0.55**	53.5
SCP (Ours)	O	75.89	92.98	**0.62**	0.68	**54.3**
SSS (Huang & Wang, 2018)	X	76.12	92.86	4.30	2.07	43.0
FPGM (He et al., 2019)	X	76.15	92.87	2.02	**0.93**	53.5
SCP (Ours)	X	75.89	92.98	**1.69**	0.98	**54.3**

pruning ratios in Table 3.

4.4. Results on ILSVRC-12

We compare the proposed method with SSS (Huang & Wang, 2018) and FPGM (He et al., 2019) on a large-scale dataset, ILSVRC-12, and present the results in Table 4. SCP accomplishes outstanding results compared to SSS for all measures and comparable performance to FPGM. Especially, the pruned model given by our approach without fine-tuning outperforms the one identified by SSS without fine-tuning by 2.61% and 1.09% points in top-1 and top-5 accuracy drop, respectively, even though our model is smaller. Also, SCP exceeds FPGM by 0.70% and 0.33% points in the top-1 accuracy drop with and without fine-tuning, respectively. Although our approach is marginally worse than FPGM in terms of top-5 accuracy drop, it has higher compression rates.

To observe the realistic and practical speed-up of the proposed method, we measure the wall clock inference time for the unpruned and pruned models on the NVIDIA TITAN Xp with a batch size of 64. SCP achieves about 24% reduction of inference time, from 104 ms without pruning to 79 ms with pruning.

4.5. Analysis

Consideration of ReLU for channel pruning One of the main ideas in this paper is to take advantage of BN and ReLU together for channel pruning, which is clearly differentiated from Slimming (Liu et al., 2017), which is based only on BN. So, we first analyze how critical ReLU in our framework by testing performance on CIFAR-100 with ResNet-164, DenseNet-40, VGGNet-19, and VGGNet-16. For the purpose, we evaluate a modified version of our algorithm with differential masks under consideration of BN operation only, which makes our pruning criteria similar to Slimming (Liu et al., 2017). In other words, the CDF $\Phi(\delta;\beta,\gamma)$ is replaced by $P(-\delta_{\text{new}} \leq x^{\text{out}} \leq \delta_{\text{new}})$, where δ_{new} is a small positive number; if the most activations in a channels are near-zeros, the channel is to be pruned in the modified formulation. In addition, we makes the objec-

Table 5. Results with and without consideration of ReLU operations on CIFAR-100. "SCP without ReLU" indicates that only BN is employed for channel pruning decisions and we only prune the channels when the absolute value of the channels are below a threshold (*e.g.*, case 4 in Figure 2).

Method	Model	Acc. Drop	FLOPs ↓ (%)
SCP (Ours)	ResNet-164	**0.62**	45.36
SCP without ReLU	ResNet-164	1.33	**48.89**
Slimming (Liu et al., 2017)	ResNet-164	2.72	47.92
SCP (Ours)	DenseNet-40	**0.77**	65.49
SCP without ReLU	DenseNet-40	1.79	62.31
Slimming (Liu et al., 2017)	DenseNet-40	3.27	**68.20**
SCP (Ours)	VGGNet-19	**0.41**	**61.94**
SCP without ReLU	VGGNet-19	1.33	54.96
Slimming (Liu et al., 2017)	VGGNet-19	4.75	59.67
SCP (Ours)	VGGNet-16	**3.55**	79.24
SCP without ReLU	VGGNet-16	5.21	78.37
Slimming (Liu et al., 2017)	VGGNet-16	8.29	**80.49**

Table 6. Sensitivity analysis about balancing term λ for the sparsity loss ($\mathcal{L}_{\text{sparse}}$) using DenseNet-40 on CIFAR-100 dataset.

λ	Acc. (%)	Acc. Drop	FLOPs ↓ (%)
1×10^{-6}	74.91	-0.67	28.99
5×10^{-6}	74.08	0.16	39.08
10×10^{-6}	73.84	0.40	46.25
30×10^{-6}	73.38	0.86	60.81
50×10^{-6}	73.17	1.07	67.82

tive function more appropriate for the new formulation by revising the sparsity loss in (15) to the following one,

$$\mathcal{L}_{\text{sparse}}(\mathbb{B}, \mathbb{C}) = \sum_{\beta_j \in \mathbb{B}, \gamma_j \in \mathbb{C}} |\beta_j + s|\gamma_j|| + |\beta_j - s|\gamma_j||. \quad (17)$$

Table 5 illustrates that our full model outperforms the modified version, denoted by "SCP without ReLU". Note that SCP without ReLU is still better than Slimming in terms of accuracy drop even though their criteria for channel pruning are similar to each other. The results highlight two advantages of our algorithm; 1) the consideration of BN and ReLU together for channel pruning is effective, and 2) our joint optimization framework with differentiable masks drives the pruned network to converge to a better model compared to Slimming, which determines masks by a heuristic.

Effect of λ for sparsity loss We analyze the effect of balancing term λ for sparsity loss on CIFAR-100 using DenseNet-40 to investigate the trade-off between accuracy and FLOPs reduction. Table 6 illustrates that the network becomes more compact as λ increases. Note that the network trained with $\lambda = 1 \times 10^{-6}$ even outperforms the baseline model by 0.67% points, which implies that our sparsity loss plays a role as a regularizer.

Table 7. Sensitivity analysis about s for the Gaussian confidence interval using DenseNet-40 on CIFAR-100 dataset with $\lambda = 30 \times 10^{-6}$.

s	Acc. (%)	Acc. Drop	FLOPs ↓ (%)
0	74.59	-0.35	45.73
1	74.47	-0.23	52.43
2	73.86	0.38	58.45
3	73.38	0.86	60.81

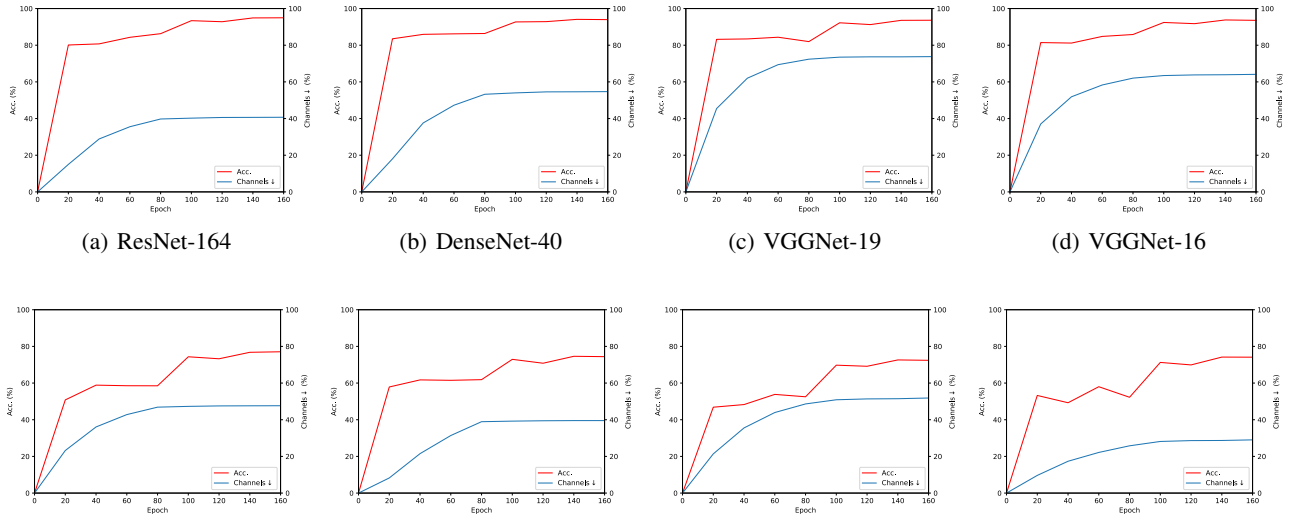
Table 8. Performance comparison between our original algorithm and its modified version given the target pruning ratio 0.4 on ILSVRC-12 using ResNet-50.

Method	FT	Top-1 Acc. Drop	Top-5 Acc.Drop	FLOPs ↓ (%)
SCP	O	**0.49**	**0.53**	**49.9**
SCP (modified training)	O	0.58	0.57	48.9
SCP	X	**1.33**	**0.92**	**49.9**
SCP (modified training)	X	1.43	1.04	48.9

Effect of s in confidence interval We study the effect of s in (15) on CIFAR-100 with DenseNet-40 to discuss accuracy and FLOPs trade-off. To this end, we test four different values of s, $\{0, 1, 2, 3\}$, because the range induced by the values covers up to very high CDF values (more than 99.8%) in the standard Gaussian distribution. Table 7 presents that larger values of s lead to lower FLOPs but lower accuracies.

Target channel pruning ratio The proposed algorithm needs to search for a proper value of λ to control target pruning ratio, which is a drawback for its applicability. However, such a limitation is addressed by a simple change of our training procedure. We apply the sparsity loss to the channels with high CDF values (the ones within the target pruning ratio) and update the model based only on the sparsity loss when it increases; otherwise, the model is updated with the total joint loss for classification and sparsity. To validate the effectiveness of this training scheme, we compare the model obtained from the new training strategy and the pruned network given by the original method with exhaustive search for λ on ILSVRC-12 using ResNet-50. Table 8 shows the new training algorithm achieves almost equivalent performance to the original version, even without time-consuming search for λ.

Balance between classification and sparsity loss Figure 4 illustrates both classification accuracy and pruning ratio with various backbone networks during their training procedures on CIFAR-10 and CIFAR-100 datasets. Note that all cases show similar tendency; 1) both classification accuracy and network sparsity improve rapidly in the early stage of training, 2) the sparsity of networks is saturated after about 80 epochs, and 3) the networks continue to en-



(a) ResNet-164 (b) DenseNet-40 (c) VGGNet-19 (d) VGGNet-16

(e) ResNet-164 (f) DenseNet-40 (g) VGGNet-19 (h) VGGNet-16

Figure 4. Plots for accuracy and pruning ratio of channels versus epochs on CIFAR-10 (first row) and CIFAR-100 (second row) datasets using ResNet-164, DenseNet-40, VGGNet-19 and VGGNet-16 networks.

hance their accuracy even in the last part of training.

5. Conclusion

This paper presents a soft channel pruning algorithm by jointly learning model parameters and pruning masks via a stochastic gradient-descent method. We formulate the channel pruning strategy in a principled way, based on the the property of a feature map derived by a sequence of BN and ReLU operations. The proposed algorithm achieves outstanding performance in terms of accuracy and efficiency consistently even without extra fine-tuning, on the multiple standard benchmarks and over several different backbone networks.

Acknowledgements

This work was partly supported by Samsung Advanced Institute of Technology (SAIT) and Institute for Information & Communications Technology Promotion (IITP) grant funded by the Korea government (MSIT) [2016-0-00563, 2017-0-01779].

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems, 2015. URL http://tensorflow.org/. Software available from tensorflow.org.

Chen, L.-C., Zhu, Y., Papandreou, G., Schroff, F., and Adam, H. Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation. In *ECCV*, 2018.

Courbariaux, M., Bengio, Y., and David, J.-P. Binaryconnect: Training Deep Neural Networks with Binary Weights during Propagations. In *NIPS*, 2015.

Denton, E. L., Zaremba, W., Bruna, J., LeCun, Y., and Fergus, R. Exploiting Linear Structure within Convolutional Networks for Efficient Evaluation. In *NIPS*, 2014.

Han, S., Pool, J., Tran, J., and Dally, W. Learning both Weights and Connections for Efficient Neural Network. In *NIPS*, 2015.

Han, S., Mao, H., and Dally, W. J. Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding. In *ICLR*, 2016.

He, K., Zhang, X., Ren, S., and Sun, J. Deep Residual Learning for Image Recognition. In *CVPR*, 2016a.

He, K., Zhang, X., Ren, S., and Sun, J. Identity Mappings in Deep Residual Networks. In *ECCV*, 2016b.

He, K., Gkioxari, G., Dollár, P., and Girshick, R. Mask R-CNN. In *ICCV*, 2017a.

He, Y., Zhang, X., and Sun, J. Channel Pruning for Accelerating very Deep Neural Networks. In *ICCV*, 2017b.

He, Y., Kang, G., Dong, X., Fu, Y., and Yang, Y. Soft Filter Pruning for Accelerating Deep Convolutional Neural Networks. In *IJCAI*, 2018.

He, Y., Liu, P., Wang, Z., Hu, Z., and Yang, Y. Filter Pruning via Geometric Median for Deep Convolutional Neural Networks Acceleration. In *CVPR*, 2019.

Huang, G., Liu, Z., Van Der Maaten, L., and Weinberger, K. Q. Densely Connected Convolutional Networks. In *CVPR*, 2017.

Huang, Z. and Wang, N. Data-Driven Sparse Structure Selection for Deep Neural Networks. In *ECCV*, 2018.

Ioffe, S. and Szegedy, C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *ICML*, 2015.

Jaderberg, M., Vedaldi, A., and Zisserman, A. Speeding up Convolutional Neural Networks with Low Rank Expansions. In *BMVC*, 2014.

Jang, E., Gu, S., and Poole, B. Categorical Reparameterization with Gumbel-Softmax. In *ICLR*, 2017.

Kingma, D. P. and Welling, M. Auto-Encoding Variational Bayes. In *ICLR*, 2014.

Kingma, D. P., Salimans, T., and Welling, M. Variational Dropout and the Local Reparameterization Trick. In *NIPS*, 2015.

LeCun, Y., Denker, J. S., and Solla, S. A. Optimal Brain Damage. In *NIPS*, 1990.

Li, H., Kadav, A., Durdanovic, I., Samet, H., and Graf, H. P. Pruning Filters for Efficient ConvNets. In *ICLR*, 2017.

Li, Y., Chen, Y., Wang, N., and Zhang, Z. Scale-Aware Trident Networks for Object Detection. In *ICCV*, 2019.

Liu, Z., Li, J., Shen, Z., Huang, G., Yan, S., and Zhang, C. Learning Efficient Convolutional Networks through Network Slimming. In *ICCV*, 2017.

Luo, J.-H., Wu, J., and Lin, W. Thinet: A filter Level Pruning Method for Deep Neural Network Compression. In *ICCV*, 2017.

Molchanov, D., Ashukha, A., and Vetrov, D. Variational Dropout Sparsifies Deep Neural Networks. In *ICML*, 2017.

Nam, H. and Han, B. Learning Multi-Domain Convolutional Neural Networks for Visual Tracking. In *CVPR*, 2016.

Noh, H., Hong, S., and Han, B. Learning Deconvolution Network for Semantic Segmentation. In *ICCV*, 2015.

Polino, A., Pascanu, R., and Alistarh, D. Model Compression via Distillation and Quantization. In *ICLR*, 2018.

Rastegari, M., Ordonez, V., Redmon, J., and Farhadi, A. XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks. In *ECCV*, 2016.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. ImageNet Large Scale Visual Recognition Challenge. *IJCV*, 2015.

Simonyan, K. and Zisserman, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *ICLR*, 2015.

Sutskever, I., Martens, J., Dahl, G., and Hinton, G. On the Importance of Initialization and Momentum in Deep Learning. In *ICML*, 2013.

Tai, C., Xiao, T., Zhang, Y., Wang, X., et al. Convolutional Neural Networks with Low-Rank Regularization. In *ICLR*, 2016.

Tan, M. and Le, Q. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. In *ICML*, 2019.

Wen, W., Wu, C., Wang, Y., Chen, Y., and Li, H. Learning Structured Sparsity in Deep Neural Networks. In *NIPS*, 2016.

Ye, J., Lu, X., Lin, Z., and Wang, J. Z. Rethinking the Smaller-Norm-Less-Informative Assumption in Channel Pruning of Convolution Layers. In *ICLR*, 2018.

Zhao, C., Ni, B., Zhang, J., Zhao, Q., Zhang, W., and Tian, Q. Variational Convolutional Neural Network Pruning. In *CVPR*, 2019.

Zhu, C., Han, S., Mao, H., and Dally, W. J. Trained Ternary Quantization. In *ICLR*, 2017.