Exploration and Estimation for Model Compression

Yanfu Zhang[1*], Shangqian Gao[1*], Heng Huang[1,2]
[1]Electrical and Computer Engineering, University of Pittsburgh, [2]JD Explore Academy
yaz91@pitt.edu, shg84@pitt.edu, henghuanghh@gmail.com

Abstract

Deep neural networks achieve great success in many visual recognition tasks. However, the model deployment is usually subject to some computational resources. Model pruning under computational budget has attracted growing attention. In this paper, we focus on the discrimination-aware compression of Convolutional Neural Networks (CNNs). In prior arts, directly searching the optimal sub-network is an integer programming problem, which is non-smooth, non-convex, and NP-hard. Meanwhile, the heuristic pruning criterion lacks clear interpretability and doesn't generalize well in applications. To address this problem, we formulate sub-networks as samples from a multivariate Bernoulli distribution and resort to the approximation of continuous problem. We propose a new flexible search scheme via alternating exploration and estimation. In the exploration step, we employ stochastic gradient Hamiltonian Monte Carlo with budget-awareness to generate sub-networks, which allows large search space with efficient computation. In the estimation step, we deduce the sub-network sampler to a near-optimal point, to promote the generation of high-quality sub-networks. Unifying the exploration and estimation, our approach avoids early falling into local minimum via a fast gradient-based search in a larger space. Extensive experiments on CIFAR-10 and ImageNet show that our method achieves state-of-the-art performances on pruning several popular CNNs.

1. Introduction

With the advance of high-performance GPUs, Convolutional Neural Networks (CNNs) achieve great success in computer vision tasks [20, 38, 39]. Via exploiting deeper structure and over-parameterization, modern CNNs have strong generalization abilities. However, the complexity of CNNs is also growing, both in the computational cost and the size of model parameters. These requirements restrict the deployment of modern CNNs to resource-sensitive platforms, such as mobile devices and low-end facilities. Another severe issue also arises for mobile devices, as the computational burden typically leads to high energy costs, which depletes the battery quickly. To address these problems, pruning weights or structures [13, 23] from computational heavy models is an effective solution.

Among existing methods, structural pruning, particularly channel pruning, is an efficient scheme since post-processing is not necessary. Although various effective methods are proposed to target this problem, these methods still suffer from several problems. In some works, the pruned networks are obtained via solving the relaxation of the integer optimization. However, bias may be introduced into the reduced pruning criterion due to the small but ineluctable impacts of certain channels. Importance sampling [27, 30] attempts to estimate the global or relative importance of neurons, but the performance is heavily affected by the priors. Recently, discrimination power [46, 10] is used as an effective criterion for channel pruning. Nevertheless, the intra-neuron relation is delicate and difficult to deal with.

To address these problems, we propose a new continuous formulation for the pruning problem of pretrained deep convolutional networks, which can be solved via alternating exploration and estimation. We characterize the sub-network space with an interpretable probabilistic model, that shares the same optimum with the naive formulation and avoids the ambiguity of non-integer values and the potential bias in naive relaxation methods. Instead of directly searching the optimal sub-networks, we compute the parameter to generate high-quality sub-networks via alternating exploration and estimation. More specifically, in the exploration step, we sample sub-networks from the temporary distribution by virtue of stochastic gradient Hamiltonian Monte Carlo (SGHMC), which can explore more sub-networks and avoid falling into sub-optimal areas encoded by the temporary distribution. Of note, the exploration is gradient-based and computationally efficient, particularly for high-dimension parameter space. In the estimation step, we compute the optimal distribution that generates high-quality sub-networks,

∗ Equal contributions. This work was partially supported by NSF IIS 1845666, 1852606, 1838627, 1837956, 1956002, IIA 2040588.

which is then used as a warm-start for the next exploration, to guide the search focusing on the informative areas. Different from importance sampling, our prior is learned from the data, which has better generalization ability. To accelerate the computation, we further construct a proximal objective in the estimation, which implicitly considers the relation between neurons.

Our main contributions are summarized as follows:

- We propose a probabilistic formulation of the model compression problem. Our new formulation is interpretable and avoids potential sub-optimal or biased evaluation of pruned networks.
- We propose a method to solve the probabilistic model via alternating exploration and estimation. We also design a correction term to ensure the sampled sub-networks obeying the FLOPs budgets.
- The proposed algorithm can search the parameter space effectively. Guided by the estimation step, the search focuses on high-quality sub-networks. Via the exploration step, the search is more likely to jump out the local minimum, benefiting from both the larger search space of sampling methods and the computational efficiency of gradient methods.
- Extensive experimental results demonstrate our approach achieves state-of-the-art performances in pruning VGG and ResNet on CIFAR10 and ImageNet.

Notations: We use the bold capital and bold lowercase symbols to represent matrices and vectors, respectively. $\boldsymbol{I}_n$ denotes a $n \times n$-identity matrix. $\mathbf{1}_n$ is a n-dimension one vector, and $\mathbf{0}_n$ is a n-dimension zero vector. The weights of layer l are represented by $\mathcal{F}_l \in \mathcal{R}^{c_l \times w_l \times h_l}$, where c_l is the number of channels, w_l and h_l are height and width of the feature map. $\mathbb{E}(\cdot)$ represents the expectation.

2. Related Works

Channel Pruning: Channel pruning technologies achieve the model compression via reducing the channel-wise redundancy. Naturally, channels with low magnitude are regarded as less important, and Group Lasso [42] is an effective method to reduce the channel magnitude. GrOWL [44] further considers similar channels on top of sparsity. Given that exact zero values are difficult to achieve via ℓ_1 regularization, explicit ℓ_0 [29] is also considered. Similarly, batch-norm can be exploited in scaling the magnitude of feature maps [31]. Another related method is AutoSlim [43], which explicitly ranks each channel during training and only searches for width. The importance of neurons is explicitly considered in some methods. For example, importance estimation [34, 30] is available by utilizing first-order or second-order Taylor expansion. Recently, discrimination-aware pruning [46] suggests the local discrimination power is informative in pruning. Different from previous methods, our method represents the sub-networks with distribution and considers the discrimination of this distribution.

Other Pruning Techniques: There are other directions to reduce the computational cost of neural networks. The connection pruning suggests more zero weights will reduce the storage requirements. The weights are pushed to be sparse during training with the help of ℓ_1/ℓ_2 regularization [13]. The weight-wise importance is considered in SNIP [22] via back-propagation. To mining the fine-grained weight importance, second-order Taylor expansion is exploited by optimal brain damage [21] and optimal brain surgeon [14]. Another weight-wise importance criterion is via hashing [2], which avoids the costly computation of Hessian matrices. Via replacing the vanilla convolution with computational efficient operations, the spatial/spatial-temporal redundancy of neural networks can be reduced. Given the scenario, such computational-efficient operations include octave convolution [41], and multi-dimensional Pruning [11], and TSN [3], *etc.*. Different from previous methods, weight quantization [12] preserves the model structures but potentially decreases the precision. For example, the model weights are binary in Binary connect [4] and binary neural network [37]. To let the gradient pass the quantization, Straight-through estimator (STE) is exploited. MetaPruning [26] generate weights for each sub-network by using a meta network, which cannot prune pre-trained models. TAS [7] use a probability distribution to sample sub-networks, but it lacks exploration and requires more computational costs due to knowledge transfer. Compared to channel pruning, these methods either require special hardware or software for implementation or have significant performance loss against the full models. As such, in this paper, we focus our scope on structural pruning. Some Neural Architecture Search (NAS) approaches are also related to pruning. DARTS [25] jointly optimizing the structure and the weights. ENAS [35] exploits a policy gradient optimization framework. Our method shares some ideas of parameter sharing based NAS methods, but the approaches, costs, search space, and detailed settings are very different.

Sampling Algorithm: In this paper we exploit sampling techniques to help the search jump out of the local minimum. Typically, a Markov chain can be built *w.r.t.* the target distribution and the samples can be generated via the walks on the chain [32]. The Metropolis-Hastings algorithm is an important method, where the samples are drawn from a simple distribution and are rejected with some probability. For high-dimension sampling, the rejection rate usually is high. By exploiting Hamiltonian dynamics, Hamiltonian Monte Carlo (HMC) [8] is designed to solve the problem.

Based on HMC, SGHMC [1] is proposed where the rejection step is no longer necessary. In Bayesian neural networks, Hamiltonian methods are also frequently utilized. Simulated annealing (SA) [36], inspired by Metropolis-Hastings, is another algorithm related to our method, which introduces temperature parameters for the canonical distribution to guide the search.

3. Proposed Method

The idea of our method is illustrated in Fig. 1. The pruned pretrained networks can be viewed as sub-networks of the full model, and we propose a probability relaxation for channel pruning in §3.1. Our task is to compute the distribution parameters to generate high-quality sub-networks. More specifically, with fixed weights of the pretrained full model, we first sequentially sample sub-networks using the temporary distribution (the exploration step described in §3.2) and update the sampler parameters (the estimation step described in §3.3). Next, we obtain the optimal sub-network according to the computed distribution, and fine-tune the pruned model to obtain the final model. More implementation details are discussed in §3.4.

3.1. Parameterized Search Space

Suppose we have a full network, denoted as $\mathcal{F}$, optimized sufficiently for data $\mathcal{D}_x$ and objective $\mathcal{L}$. The channel pruning problem is to find the optimal sub-network *w.r.t.* $\mathcal{F}$ constrained by some computational budget b. In the context of channel pruning, the sub-networks are defined by removing part of the channels of $\mathcal{F}$ with the model weights frozen. Let $\mathcal{N}$ denote the neuron set of $\mathcal{F}$, then a pruned model can be represented using the subset $\mathcal{N}_s \subset \mathcal{N}$ preserved after the pruning. Let f represent a function computing the resource given $\mathcal{N}_s$. Formally, our task is,

$$\mathcal{N}_s^* = \arg\min_{\mathcal{N}_s} \mathcal{L}(\mathcal{N}_s, \mathcal{D}_x), \quad s.t. \quad f(\mathcal{N}_s) \leq b, \quad (1)$$

let $\mathbb{I}_{\mathcal{N}_s}(n_i)$ be the indicator function of the presence of neuron n_i in the sub-network $\mathcal{N}_s$, the above problem is transformed into an integer programming problem, which is non-smooth, non-convex and NP-hard.

To address this problem, we propose a probabilistic relaxation of Prob. (1). We use a Bernoulli distribution parameterized by $\boldsymbol{p} = [p_{n_1}, p_{n_2}, \cdots, p_{n_N}]$ to characterize sub-networks $\mathcal{N}_s$. p_{n_i} is the probability that n_i is preserved in $\mathcal{N}_s$. The objective *w.r.t.* $\boldsymbol{p}$ is defined as $\mathcal{L}_{\boldsymbol{p}}(\boldsymbol{p}) = \mathbb{E}_{\boldsymbol{p}}[\mathcal{L}(\mathcal{N}_s)]$. Finally, the sub-network search problem can be written as,

$$\boldsymbol{p}^* = \arg\min_{\boldsymbol{p}} \mathcal{L}_{\boldsymbol{p}}(\boldsymbol{p}, \mathcal{D}_x), \quad s.t. \quad \mathbb{E}_{\boldsymbol{p}}[f(\mathcal{N}_s)] \leq b. \quad (2)$$

Compared to Prob. (1), our new formulation Prob. (2) has two advantages. Firstly, Prob. (2) allows an interpretable non-integer optimum. Observe that $\mathcal{N}_s^*$ in (1) is potentially non-unique, as such, non-integer $\boldsymbol{p}^*$ indicates there are multiple candidate sub-networks with competing performance. Secondly, Prob. (2) can avoid the ambiguity of multiple optimal sub-networks. We can find that the solution set $\{\mathcal{N}_s^*\}$ of Prob. (1) is in the solution set $\boldsymbol{p}^*$ of the Prob. (2), and that their objectives have the same minimal values. Moreover, if $\boldsymbol{p}^*$ is unique and the entries are supported only on $\{0, 1\}$, the optimal sub-network is also unique and the solution degenerates to the unique $\mathcal{N}_s^*$. As depicted in Fig. 1, we solve Prob. (2) via alternating fixing $\boldsymbol{p}$ to explore sub-networks and updating $\boldsymbol{p}$ based on collected sub-networks.

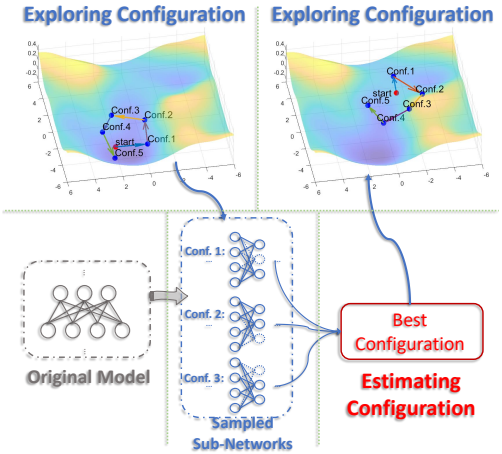


Figure 1: The illustration of exploration and estimation algorithm to solve (2). In exploration step, we generate several configurations (blue dots) from the start point (red dot), and sample sub-networks from the configurations. In estimation step, we compute the best configuration as the start point for next exploration.

3.2. Exploring Sub-Networks Sampling

In the exploration step, we sample representative sub-networks based on fixed $\boldsymbol{p}$. More specifically, in each sampled sub-network, neuron n_i and the associated weights are preserved with probability p_{n_i} independently. Prob. (2) defines a constrained optimization problem. However, there is no budget guarantee for the sub-networks sampled from $\boldsymbol{p}$. To address this problem, we propose a method to sample sub-networks aware of FLOPs constraints. We also insert randomness into $\boldsymbol{p}$ using SGHMC to enlarge the search space which empirically improves the pruning performance.

Sampling Sub-Networks with FLOPs Constraints In this paper, we consider the computational budget described in FLOPs. Given a data sample, the FLOPs $f_l(\mathcal{N}_s)$ associated with a single convolution layer l in $\mathcal{N}_s$ can be denoted as,

$$f_l(\mathcal{N}_s) = k_l^2(\mathbf{1}^\intercal \boldsymbol{p}_{l-1})(\mathbf{1}^\intercal \boldsymbol{p}_l) w_l h_l, \quad (3)$$

here k_l is the kernel size, w_l and h_l are width and height, $\boldsymbol{p}_l$ and $\boldsymbol{p}_{l-1}$ are vectors representing layer l and $l-1$ of the sampled $\mathcal{N}_s$. The unpruned FLOPs $f_l(\mathcal{N})$ is obtained by replacing $\mathbf{1}^\intercal \boldsymbol{p}_{l-1}$ and $\mathbf{1}^\intercal \boldsymbol{p}_l$ with c_{l-1} and c_l, the number of input and output channels.

To sample a sub-network subject to the FLOPs constraint, the probability $\boldsymbol{p}$ needs correction. A simple solution is to neglect the selected gates with probability $\min\left(1, \frac{f_l(\mathcal{N}_s)}{b \times f_l(\mathcal{N})}\right)$. However, from (3), the FLOPs associated with different layers are highly imbalanced, affected by both the layer structure and the number of connected neurons. To generate sub-networks with stable FLOPs, we add an alternative correction term to p_{n_i} in $\boldsymbol{p}$,

$$\hat{p}_{n_i} = p_{n_i} \min\left(\frac{T_\alpha g_l(\mathbf{1}^\intercal \boldsymbol{p}_l, \mathbf{1}^\intercal \boldsymbol{p}_{l+1})}{g_l(bc_l, bc_{l+1})}, 1\right), \tag{4}$$

here $T_\alpha \geq 1$ is a parameter to control the importance of the FLOPS constraint. $g_l(\cdot)$ is defined as,

$$g_l(x, y) = k_l^2 x w_l h_l + k_{l+1}^2 y w_{l+1} h_{l+1}. \tag{5}$$

Compared to the simple method, the sampled sub-networks by the proposed approach is more stable, because $\hat{p}_{n_i}$ is related to the associated FLOPs. We start from a relative large T_α and gradually reduce it, which is equivalent to decreasing the budget from 1 to b. Meanwhile, large T_α at the beginning can help avoiding skimping on some neurons from the early exploration.

Explorating Sub-Networks Sampling via SGHMC: A severe problem in gradient-based channel pruning methods is the instability introduced by the relaxation from discrete values to continuous values. The instability is then conveyed to the gradient-based update, and eventually twists the search space of sub-networks. This problem is alleviated in our probabilistic formulation. Now suppose we have sampled a set of $\mathcal{N}_s$. For high-dimension probabilistic models, the $\mathcal{N}_s$ set can be generated with different $\boldsymbol{p}$ with close probability. Note that each $\boldsymbol{p}$ implies different search spaces and different local optimum. If we exploit multiple potential $\boldsymbol{p}$, the generation of sub-networks will have better diversity and it is more likely to jump out of bad local minimum. To this end, we can insert randomness into $\boldsymbol{p}$ via Stochastic Gradient Hamiltonian Monte Carlo (SGHMC).

SGHMC is a fast sampling technique inspired by Hamiltonian dynamics. For self-containedness, we recap the necessary steps in SGHMC in the following. First, we construct the desired probability as $p(\boldsymbol{p}|\mathcal{D}_\mathcal{N}, b) \propto \exp(-U(\boldsymbol{p}))$. $\mathcal{D}_\mathcal{N} \triangleq \{(\mathcal{N}_s, \mathcal{L}(\mathcal{N}_s))\}$ refers to the set of observed sub-networks and their performances. U is the potential energy function given by,

$$U = -\sum \log p\left((\mathcal{N}_s, \mathcal{L}(\mathcal{N}_s))|\boldsymbol{p}\right) - \log p(\boldsymbol{p}), \tag{6}$$

here the first term can be viewed as the discrimination power of $\boldsymbol{p}$, the second term is the prior distribution of $\boldsymbol{p}$. Using (4), (6) can be written as,

$$U = -\sum \mathcal{L}_{\boldsymbol{p}}(\hat{\boldsymbol{p}}) - \log p(\hat{\boldsymbol{p}}), \tag{7}$$

where we abuse $\mathcal{L}_{\boldsymbol{p}}(\cdot)$ on the corrected $\hat{\boldsymbol{p}}$, representing the uncorrected $\mathcal{L}_{\boldsymbol{p}}(\boldsymbol{p})$. In this step $\boldsymbol{p}$ is not updated, instead we use it as the start point and sample new $\hat{\boldsymbol{p}}$, which is used for generating new sub-networks. For simplicity, we directly consider the priors on $\hat{\boldsymbol{p}}$. (4) considers the local FLOPs *w.r.t.* neurons of the the sampled sub-networks. We additionally consider the global stability of the sampled sub-networks. Specifically, let $p(\hat{\boldsymbol{p}}) \propto \exp(-\mathcal{L}_v(\hat{\boldsymbol{p}}))$, where $\mathcal{L}_v(\hat{\boldsymbol{p}}) \triangleq \mathbb{E}_{\boldsymbol{p}}\left[\max\left(\left(\frac{\mathbb{E}_{\boldsymbol{p}}[f(\mathcal{N}_s)]}{b \times \mathbb{E}_{\boldsymbol{p}}[f(\mathcal{N})]}\right)^{-T_\beta} - 1, 0\right)^2\right]$. $\mathcal{L}_v(\hat{\boldsymbol{p}})$ is then used to replace the second term of Eq. 7. T_β is a temperature parameter controlling the strength of the constraint. $\mathcal{L}_v(\hat{\boldsymbol{p}})$ can be viewed as the FLOPs variance of the sampled sub-networks. Finally, U is given by,

$$U = \mathcal{L}_{\boldsymbol{p}}(\hat{\boldsymbol{p}}, \mathcal{D}_x) + \lambda \mathcal{L}_v(\hat{\boldsymbol{p}}), \tag{8}$$

where λ is a parameter to compensate the normalization in $p(\hat{\boldsymbol{p}})$, $\mathcal{L}_{\boldsymbol{p}}(\hat{\boldsymbol{p}}, \mathcal{D}_x) = -\sum \mathcal{L}_{\boldsymbol{p}}(\hat{\boldsymbol{p}})$, and D_x is the dataset for pruning. Assuming $\hat{\boldsymbol{p}}$ follows a Hamiltonian dynamic system with friction, we have,

$$d\hat{\boldsymbol{p}} = \boldsymbol{r}dt, \quad d\boldsymbol{r} = -\triangledown U(\hat{\boldsymbol{p}})dt - \boldsymbol{r}dt + \mathcal{N}(\mathbf{0}, 2\boldsymbol{I}), \tag{9}$$

here $\boldsymbol{r} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ is a random momentum, $\mathcal{N}(\mathbf{0}, \boldsymbol{I})$ is a Gaussian distribution. We omit the noise term, and set the friction term to $\boldsymbol{I}$. The sample of $\hat{\boldsymbol{p}}$ is obtained by updating $\hat{\boldsymbol{p}}$ and $\boldsymbol{r}$ iteratively,

$$\hat{\boldsymbol{p}}^{(k+1)} = \hat{\boldsymbol{p}}^{(k)} + \epsilon \boldsymbol{r}^{(k)}, \tag{10}$$

$$\boldsymbol{r}^{(k+1)} = \boldsymbol{r}^{(k)} - \epsilon\left(\triangledown U(\hat{\boldsymbol{p}}) - \boldsymbol{r}^{(k)} + \mathcal{N}(\mathbf{0}, 2\boldsymbol{I})\right). \tag{11}$$

where ϵ is the step size. For simplicity we recursively compute $\hat{\boldsymbol{p}}^{(k)}$ and the associated sub-networks. The computation of $\mathcal{L}_{\boldsymbol{p}}(\hat{\boldsymbol{p}})$ is discussed in the next section.

3.3. Estimating Start Point

The exploration step elaborated in §3.2 allows larger search space *w.r.t.* sub-networks. To make the search more efficient, we utilize an estimation step to guide the search focusing on the area with high potential. Specifically, we compute a plausible $\boldsymbol{p}^*$ from the historically sampled sub-networks $\mathcal{D}_\mathcal{N}$, which is used as the start point in the exploration step.

In this step, we minimize the following objective,

$$\boldsymbol{p}^* = \arg\min_{\boldsymbol{p}} \frac{1}{Z_\mathcal{N}} \sum_{\mathcal{N}_s \in \mathcal{D}_\mathcal{N}} p_{\boldsymbol{p}}(\mathcal{N}_s)\mathcal{L}(\mathcal{N}_s), \tag{12}$$

here $Z_{\mathcal{N}} = \sum_{\mathcal{N}_s \in \mathcal{D}_{\mathcal{N}}} p_{\boldsymbol{p}}(\mathcal{N}_s)$ is a normalization term, and,

$$P_{\boldsymbol{p}}(\mathcal{N}_s) = \prod p_{n_j}^{\mathbb{I}_j}(1 - p_{n_j})^{1-\mathbb{I}_j}, \tag{13}$$

here $\mathbb{I}_j$ is the indicator function of neuron j in $\mathcal{N}_s$. Note that $\mathcal{F}$ is fixed, therefore $\mathcal{L}(\mathcal{N}_s)$ can be viewed as a constant given a sub-network $\mathcal{N}_s$. $\mathcal{L}_{\boldsymbol{p}}(\hat{\boldsymbol{p}})$ involves a similar $P_{\boldsymbol{p}}(\mathcal{N}_s)$ term, and the gradients *w.r.t.* $\hat{\boldsymbol{p}}$ can be computed based on sampled sub-networks.

However, $\mathcal{D}_{\mathcal{N}}$ is dynamically growing in the sampling which is inefficient in both maintaining and computation. To make the problem tractable, we propose a proximal objective function and compute $\boldsymbol{p}^*$ in an incremental manner. Denote ${\boldsymbol{p}^{(k)}}^*$ as the start point in round k, the $k+1$ start point can be estimated by,

$$\begin{aligned} {\boldsymbol{p}^{(k+1)}}^* = \arg\min_{\boldsymbol{p}} & \mathbb{D}_{KL}(\boldsymbol{p}|{\boldsymbol{p}^{(k)}}^*) + \\ & \frac{1}{Z_{\mathcal{N}}^{(k)}} \sum_{\mathcal{N}_s \in \mathcal{D}_{\mathcal{N}}^{(k)}} p_{\boldsymbol{p}}(\mathcal{N}_s)\mathcal{L}(\mathcal{N}_s), \end{aligned} \tag{14}$$

here $\mathbb{D}_{KL}(\cdot)$ is the KL divergence, $\mathcal{D}_{\mathcal{N}}^{(k)}$ is the samples start from ${\boldsymbol{p}^{(k)}}^*$. The optimization of (14) can be achieved via a standard stochastic gradient descent method. Note that in (14), ${\boldsymbol{p}^{(k)}}^*$ and $\mathcal{L}(\mathcal{N}_s)$ are deterministic, and the gradient of $\boldsymbol{p}$ is mainly computed on the KL divergence and $p_{\boldsymbol{p}}(\mathcal{N}_s)$ defined in (13).

The rationale behind the above approximation is twofold. Firstly, ${\boldsymbol{p}^{(k)}}^*$ bears the historical observations which is dominant in $\mathcal{D}_{\mathcal{N}}$, thus we expect ${\boldsymbol{p}^{(k+1)}}^*$ is close to ${\boldsymbol{p}^{(k)}}^*$. Secondly, $\mathcal{D}_{\mathcal{N}}^{(k)}$ is generated via two-step sampling, which can be viewed as generating from ${\boldsymbol{p}^{(k)}}^*$ but with larger variance. As such, the original problem can be reduced to the KL divergence term and the new observation term.

3.4. Algorithm and Implementation

Our algorithm is summarized in Algo. 1. During the search, the original model weights are remained untouched, and we only search the optimal combination of neurons from the full model. The finalized pruning is determined by deleting all neurons for $p_{n_i} \leq 0.5$ (step 18 in Algo. 1). Then, we finetune the pruned network to obtain the final model. Although we use the SGHMC method in the sampling, the optimized parameter is computed in the estimation step, which is more close to Maximum A Posterior estimation (MAP) with a Canonical distribution as the prior. SGHMC is an auxiliary method to enlarge the search space of sub-networks. On the other hand, the parameters of the sub-network distribution can be viewed as neuron-wise importance. Compared to the related method, we introduce the exploration step, which can be interpreted as a Bayesian model. Of note, our estimation explicitly considers the computational budgets, which is interpretable and avoids the bias by the threshold pruning in previous methods.

Algorithm 1: Exploration and Estimation for Model Compression

Input: Full model $\mathcal{F}$, pruning data $\mathcal{D}_x$, finetune data $\mathcal{D}_f$, budget b, parameter T_α, T_β, λ, K, m, ϵ.
Output: $\boldsymbol{p}^{(0)}$, pruned & finetuned model $\mathcal{F}_p$.

1 Initialization $\boldsymbol{p}^{(0)} = \mathbf{1}$.
`// Model pruning with` $\mathcal{F}$ `weights frozen`
2 **repeat**
3 Shuffle $\mathcal{D}_x$, $\mathcal{D}_{\mathcal{N}} \leftarrow \emptyset$
4 Sample $\mathcal{D}_{\mathcal{N}}^{(0)}$ from $\boldsymbol{p}^{(0)}$ using (4)
5 $\mathcal{D}_{\mathcal{N}} \leftarrow \mathcal{D}_{\mathcal{N}} \cup \mathcal{D}_{\mathcal{N}}^{(0)}$
6 Compute $\hat{\boldsymbol{p}}^{(0)}$ *w.r.t.* (4)
7 **for** $k \leftarrow 0$ **to** K **do**
8 $T_\alpha \leftarrow \max(0.999 \times T_\alpha, 1)$, $T_\beta \leftarrow 0.999 \times T_\beta$.
9 Sample momentum $\boldsymbol{r} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$
10 $\bar{\boldsymbol{p}}^{(0)} \leftarrow \hat{\boldsymbol{p}}^{(k)}$
11 **for** $i \leftarrow 0$ **to** m **do**
12 Compute $\bar{\boldsymbol{p}}^{(i+1)}$ *w.r.t.* (10)
13 $\hat{\boldsymbol{p}}^{(k+1)} \leftarrow \bar{\boldsymbol{p}}^{(m+1)}$
14 Sample $\mathcal{D}_{\mathcal{N}}^{(k+1)}$ from $\hat{\boldsymbol{p}}^{(k+1)}$ using (4)
15 $\mathcal{D}_{\mathcal{N}} \leftarrow \mathcal{D}_{\mathcal{N}} \cup \mathcal{D}_{\mathcal{N}}^{(k+1)}$
16 Update $\boldsymbol{p}^{(0)}$ on $\mathcal{D}_{\mathcal{N}}$ *w.r.t.* (14)
17 **until** *Converges.*;
`// Finetune pruned model`
18 Obtain pruned model $\mathcal{F}_p$ via deleting neurons in $\mathcal{F}$ using binarized $\boldsymbol{p}^{(0)}$
19 **repeat**
20 Finetune $\mathcal{F}_p$ on $\mathcal{D}_f$.
21 **until** *Converges.*;

4. Experimental Results

4.1. Experimental Settings

We verify our method on CIFAR-10 [20] and ImageNet [5]. We use $b \in (0, 1)$ as the FLOPs budget, denoting the percentage of FLOPs preserved in the pruning. For CIFAR-10, the target models include VGG-16 and ResNet 56, and the network weights are trained from scratch using PyTorch example codes. $\lambda = 0.001$ recommended by the original implementation [1].We set $T_\alpha = 2$ and $T_\beta = 3$, and the performance is not sensitive to T_α and T_α when they are not too large. The pruned models are finetuned for 160 epochs using SGD with start learning rate 0.1, weight decay 0.0001, and momentum 0.8, the learning rate is mul-

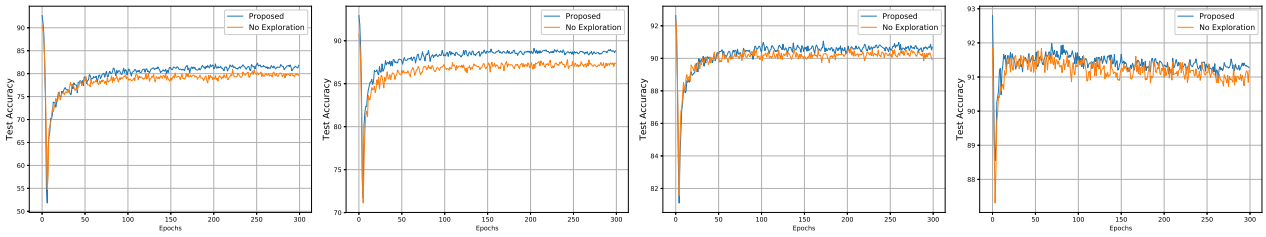
Method	Architecture	Baseline Acc	Pruned Acc	Δ-Acc	Pruned FLOPs
VP [45]	VGG-16	93.25%	93.18%	−0.07%	39.1%
Slimming [27]		93.85%	92.91%	−0.94%	48.1%
DCP [46]		93.99%	93.82%	−0.17%	50.0%
DCP-Adapt [46]		93.99%	93.41%	−0.58%	35.0%
SCP [18]		93.85%	93.79%	−0.06%	**66.23%**
Proposed		93.36%	93.63%	**+0.27%**	56.6%
DCP [46]	ResNet-56	93.80%	93.49%	−0.31%	50.0%
DCP-Adapt [46]		93.80%	93.81%	+0.01%	47.0%
SCP [18]		93.69%	93.23%	−0.46%	51.5%
FPGM [17]		93.59%	92.93%	−0.66%	52.6%
SFP [16]		93.59%	92.26%	−1.33%	52.6%
FPC [15]		93.59%	93.24%	−0.25%	52.9%
Proposed		93.62%	93.68%	**+0.06%**	**56.0%**

Table 1: Comparison results on CIFAR-10 dataset with ResNet-56 and VGG-16. Δ-Acc represents the performance changes before and after model pruning. ± indicates increase or decrease compared to baseline results. Baseline results are adopted from the original papers. The best results are in bold.

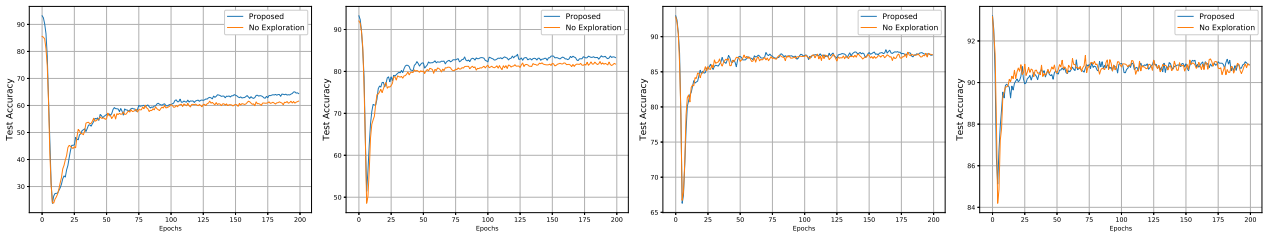
Method	Architecture	BL Top-1 Acc	BL Top-5 Acc	Δ-Acc Top-1	Δ-Acc Top-5	Pruned FLOPs
MIL [6]	ResNet-18	69.98%	89.24%	−3.65%	−2.30%	34.6%
SFP [16]		70.28%	89.63%	−3.18%	−1.85%	41.8%
FPGM [17]		70.28%	89.63%	**−1.87%**	**−1.15%**	41.8%
Proposed		70.28%	89.63%	−2.01%	−1.19%	**46.6%**
SFP [16]	ResNet-50	76.15%	92.87%	−1.54%	−0.81%	41.8%
IE [30]		76.15%	–	−1.68%	–	45.0%
SCOP-A [40]		76.15%	92.87%	−0.20%	**−0.08%**	45.3%
Proposed		76.15%	92.87%	**−0.10%**	**−0.02%**	**46.1%**
ABCP-70 [24]		76.01%	92.96%	−2.49%	−1.45%	**56.6%**
ABCP-80 [24]		76.01%	92.96%	−2.15%	−1.27%	54.29%
SCOP-B [40]		76.15%	92.87%	−0.89%	−0.34%	54.6%
FPGM [17]		76.15%	92.87%	−1.32%	−0.55%	53.5%
CCP [34]		76.15%	92.87%	−0.94%	−0.45%	54.1%
SCP [18]		75.89%	92.98%	−0.62%	−0.68%	54.3%
DMC [9]		76.15%	92.87%	−0.80%	−0.38%	55.0%
DCP [46]		76.01%	92.93%	−1.06%	−0.61%	55.6%
Proposed*		76.15%	92.87%	**−0.49%**	**−0.35%**	56.0%
Slimming [27]	VGG-11bn	70.84%	–	−2.22%	–	8.9%
Rethinking [28]		70.84%	–	−0.84%	–	8.9%
IE [30]		70.84%	–	−0.20%	–	8.9%
Proposed		70.84%	89.81%	**+0.31%**	**+0.67%**	**34.0%**

Table 2: Comparison results on ImageNet dataset with ResNet-18, ResNet-50 and VGG-11bn. BL refers to baseline. Δ-Acc represents the performance changes before and after model pruning. ± indicates increase or decrease compared to baseline results. For ResNet-50, we compare two different pruned rates, denoted as *Proposed* and *Proposed**.

tiplied by 0.1 at epoch 80 and 120. For ImageNet, the target models include VGG-11 with batchnorm, ResNet18, and ResNet 50, and we use the pre-trained model released by PyTorch [33]. In all experiments, $\lambda = 0.001$, $T_\alpha = 4$, and $T_\beta = 5$. After pruning, we finetune the model for 60 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 20 and 40. The finetuning hyperparameters are similar to those used in Collaborative Channel Pruning (CCP) [34] for both CIFAR-10 and ImageNet. We randomly choose 2500 and 10,000 samples from CIFAR-10 and ImageNet respectively as the training set for the proposed method. Our model is trained using ADAM [19] optimizer with a constant learning rate of 0.001 for 300 epochs. All the codes are implemented with PyTorch [33]. The experiments are conducted on a machine with 4 Nvidia Tesla P40 GPUs.



(a) VGG-16, 35% FLOPs (b) VGG-16, 50% FLOPs (c) VGG-16, 60% FLOPs (d) VGG-16, 80% FLOPs



(e) ResNet-56, 35% FLOPs (f) ResNet-56, 50% FLOPs (g) ResNet-56, 60% FLOPs (h) ResNet-56, 80% FLOPs

Figure 2: (a)-(d), test accuracy of pruned VGG-16 during training under different prune rate. (e)-(h), test accuracy of pruned ResNet-56 during training under different prune rate. The computational budgets are evaluated mainly on convolution operations, and are not strictly corresponding to the pruned FLOPs.

4.2. Results on CIFAR10

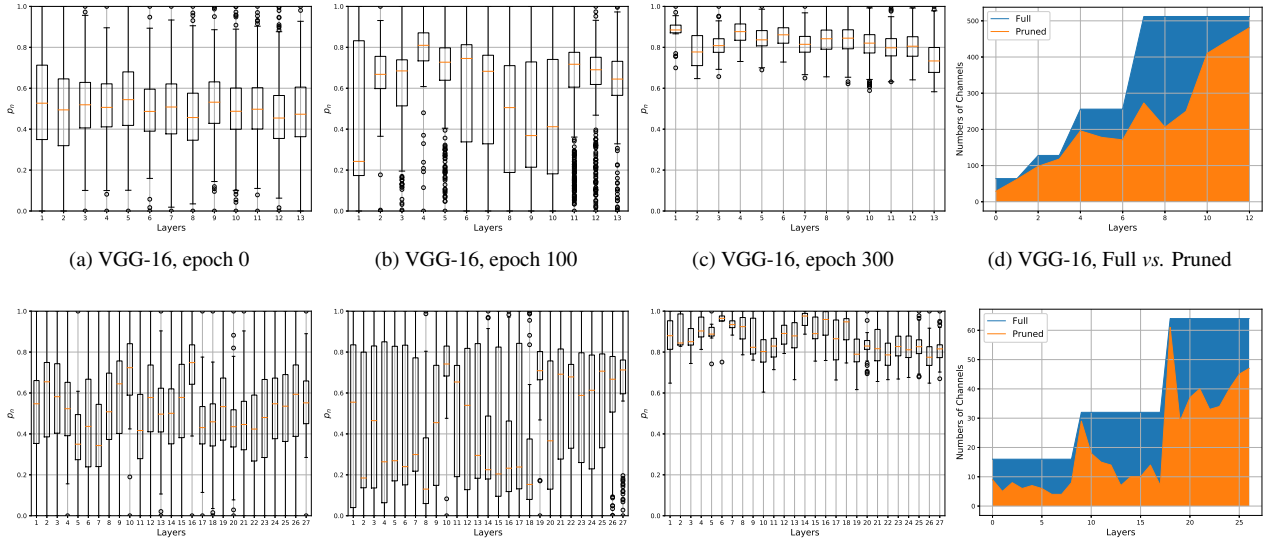
Table 1 presents the results of the proposed method and related baselines on CIFAR-10. For VGG-16, our method outperforms state-of-the-art baselines in both accuracies and pruned FLOPs. For example, the proposed method significantly outperforms DCP and DCP-Adapt [46]. DCP prunes more FLOPs by sacrificing performance compared to DCP-Adapt. Δ-Acc of our approach outperforms DCP-Adapt by 0.05% and prunes 6% more FLOPs compared to DCP. For ResNet-56, similar results can be observed. Besides SCP [18], our approach has the best Δ-Acc and the highest pruned FLOPs. DCP performs better than DCP-Adapt in general, and our approach outperforms DCP in both accuracies and pruned FLOPs. SCP [18] prunes more FLOPs, however, the proposed method achieves significantly higher Δ-Acc by up to 0.33%.

4.3. Results on ImageNet

Table 1 summarizes the comparison results on ImageNet. For ResNet-18, our approach is comparable to other related methods. Compared to FPGM which is the best-performed baseline, our method has slightly sacrificed accuracy for higher pruned FLOPs. For ResNet-50, our method shows state-of-the-art performances. For example, the accuracy of the sub-network found by our method is 0.13% higher in Δ-Acc Top-1 and 0.33% higher in Δ-Acc Top-5 compared to SCP under similar computational budgets. Of note, our method outperforms IE — a state-of-the-art importance-based method — in accuracy by up to 1.58%. Meanwhile, 1.2% more FLOPs are pruned. The generative parameter estimated by our method can also be interpreted as neuron importance, and we conjecture that the improvements are due to that the estimation of our approach is based on the temporary sub-networks instead of full networks, which is more accurate. For VGG-11, the proposed method significantly outperforms the related baselines. Compared to IE, the best-performed baseline, our method improves Δ-Acc Top-1 by 0.51%. Moreover, the improvement is achieved via pruning up to 34.0% FLOPs, which is much higher than the baselines where only 8.9% FLOPs are pruned.

4.4. Ablation Study and Discussions

In this section, we discuss some observations in the sub-network search of our approach. Firstly we examine the necessity of the exploration step. Specifically, we show the exploration step can help find better sub-networks. We compare the performances of several scenarios including *No Expl.*, referring to not using the exploration, *Expl. (a)*, where $K = 1$ and $m = 1$, *Expl. (b)*, where $K = 3$ and $m = 3$, *Expl. (c)*, where $K = 5$ and $m = 5$. The results are described in Tab. 3. For VGG-16, *Expl.(c)* clearly outperforms *No Expl.*, *Expl.(a)*, and *Expl.(b)* by a large margin. For ResNet-56, the improvements via introducing exploration is also observed, for example *Expl.(c)* outperforms *No Expl.* by 0.37%. These results verify our claim.



(a) VGG-16, epoch 0 (b) VGG-16, epoch 100 (c) VGG-16, epoch 300 (d) VGG-16, Full *vs.* Pruned

(e) ResNet-56, epoch 0 (f) ResNet-56, epoch 100 (g) ResNet-56, epoch 200 (h) ResNet-56, Full *vs.* Pruned

Figure 3: The dynamic of $\hat{\boldsymbol{p}}$ in Algo. 1. For VGG-16, (a)-(c), the layer-wise box plot of $\hat{\boldsymbol{p}}$ in epoch 0, 100, and 300; (d), the number of channels for the full network and the pruned network. In (c) we only plot $\hat{\boldsymbol{p}}$ of the preserved channels. (e)-(h), the results on ResNet-56, defined similar to (a)-(d).

Method	Architecture	Pruned Acc	Pruned FLOPs
No Expl.	VGG-16	92.88%	56.6%
Expl. (a)		93.17%	56.6%
Expl. (b)		93.33%	56.6%
Expl. (c)		**93.63%**	56.6%
No Expl.	ResNet-56	93.31%	56.0%
Expl. (a)		93.46%	56.0%
Expl. (b)		93.59%	56.0%
Expl. (c)		**93.68%**	56.0%

Table 3: Comparison of different exploration settings.

Secondly, we plot the test accuracy curve during the sub-network search of our approach in Fig. 2. To generalize our findings, we alter the computational budgets include 0.35, 0.5, 0.60, and 0.80. Here we highlight several noteworthy observations. Firstly, the proposed method is effective in different budget constraints. Secondly, in our approach, the search on sub-networks has two stages. In the first stage, the search is dominanted by the FLOPs constraints, and weakly-performed sub-networks are sampled. This is a very short stage. In the second stage, the optimal parameters are gradually discovered. Thirdly, the search is more difficult for more strict computational budges. For example, with 80% FLOPs we obtain satisfactory parameters after merely 50 epochs. But with 80% FLOPs, the sub-network sampling is continuously improving until 300 epochs. The test accuracy for *No Expl.* is also included in Fig. 2. The results show that the exploration step can help the sampling of good sub-networks instead of stuck in a sub-optimal area, particularly for high prune rates.

Lastly, to provide a closer view of the dynamics of the parameters, in Fig. 3 we present the layer-wise statistics of $\hat{\boldsymbol{p}}$ during the search. At the beginning of the search on VGG-16, $\hat{\boldsymbol{p}}$ is generally random. With more sampled sub-networks, part of neurons in each layer are identified as unimportant, which can be verified by the bottom part of Fig. 3 (b). Meanwhile, the larger variance also implies that informative neurons are differentiated from the unimportant. In the pruned networks, $\hat{\boldsymbol{p}}$ of the preserved neurons are close to 1 and have much less variance in each layer, which can be observed from Fig. 3 (c). These results indicate that the learned $\hat{\boldsymbol{p}}$ discovers the neuron-wise discriminator power. For ResNet-56, a similar tendency can be found. The layer-wise variance is growing during the search and the pruned network only preserves highly discriminating channels.

5. Conclusion

In this manuscript, we propose a novel method to prune CNN. We formulate the search of sub-networks as a probabilistic model and design an effective optimization method. We propose an alternating exploration and estimation scheme to solve the problem. Specifically, we enlarge the search space in the exploration step to avoid falling into bad local optimums. Meanwhile, the estimation step guides the search to focus on high potential sub-networks in the estimation step. Extensive experiments show our approach achieves state-of-the-art performances. The training dynamics of our algorithm also verifies the effectiveness of the search scheme.

References

[1] Tianqi Chen, Emily Fox, and Carlos Guestrin. Stochastic gradient hamiltonian monte carlo. In *International conference on machine learning*, pages 1683–1691, 2014. 3, 5

[2] Wenlin Chen, James Wilson, Stephen Tyree, Kilian Weinberger, and Yixin Chen. Compressing neural networks with the hashing trick. In *International conference on machine learning*, pages 2285–2294, 2015. 2

[3] Yunpeng Chen, Haoqi Fan, Bing Xu, Zhicheng Yan, Yannis Kalantidis, Marcus Rohrbach, Shuicheng Yan, and Jiashi Feng. Drop an octave: Reducing spatial redundancy in convolutional neural networks with octave convolution. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3435–3444, 2019. 2

[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. 2

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

[6] Xuanyi Dong, Junshi Huang, Yi Yang, and Shuicheng Yan. More is less: A more complicated network with less inference complexity. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5840–5848, 2017. 6

[7] Xuanyi Dong and Yi Yang. Network pruning via transformable architecture search. *arXiv preprint arXiv:1905.09717*, 2019. 2

[8] Simon Duane, Anthony D Kennedy, Brian J Pendleton, and Duncan Roweth. Hybrid monte carlo. *Physics letters B*, 195(2):216–222, 1987. 2

[9] S. Gao et al. Discrete model compression with resource constraint for deep neural networks. In *CVPR*, 2020. 6

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

[11] Jinyang Guo, Wanli Ouyang, and Dong Xu. Multi-dimensional pruning: A unified framework for model compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1508–1517, 2020. 2

[12] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015. 2

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

[14] Babak Hassibi and David G Stork. Second order derivatives for network pruning: Optimal brain surgeon. In *Advances in neural information processing systems*, pages 164–171, 1993. 2

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

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

[17] Yang He, Ping Liu, Ziwei Wang, Zhilan Hu, and Yi Yang. Filter pruning via geometric median for deep convolutional neural networks acceleration. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4340–4349, 2019. 6

[18] Minsoo Kang and Bohyung Han. Operation-aware soft channel pruning using differentiable masks. *International Conference on Machine Learning*, 2020. 6, 7

[19] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 6

[20] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009. 1, 5

[21] Yann LeCun, John S Denker, and Sara A Solla. Optimal brain damage. In *Advances in neural information processing systems*, pages 598–605, 1990. 2

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

[23] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. *ICLR*, 2017. 1

[24] Mingbao Lin, Rongrong Ji, Yuxin Zhang, Baochang Zhang, Yongjian Wu, and Yonghong Tian. Channel pruning via automatic structure search. *arXiv preprint arXiv:2001.08565*, 2020. 6

[25] Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: Differentiable architecture search. In *International Conference on Learning Representations*, 2019. 2

[26] Z. Liu et al. Metapruning: Meta learning for automatic neural network channel pruning. In *ICCV*, 2019. 2

[27] Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. Learning efficient convolutional networks through network slimming. In *ICCV*, 2017. 1, 6

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

[29] Christos Louizos, Max Welling, and Diederik P. Kingma. Learning sparse neural networks through l_0 regularization. In *International Conference on Learning Representations*, 2018. 2

[30] Pavlo Molchanov, Arun Mallya, Stephen Tyree, Iuri Frosio, and Jan Kautz. Importance estimation for neural network pruning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 11264–11272, 2019. 1, 2, 6

[31] Hyeonseob Nam and Hyo-Eun Kim. Batch-instance normalization for adaptively style-invariant neural networks. In *NIPS*, 2018. 2

[32] Radford M Neal et al. Mcmc using hamiltonian dynamics. *Handbook of markov chain monte carlo*, 2(11):2, 2011. 2

[33] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pages 8024–8035, 2019. 6

[34] 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. 2, 6

[35] Hieu Pham, Melody Y Guan, Barret Zoph, Quoc V Le, and Jeff Dean. Efficient neural architecture search via parameter sharing. *arXiv preprint arXiv:1802.03268*, 2018. 2

[36] Martin Pincus. Letter to the editor—a monte carlo method for the approximate solution of certain types of constrained optimization problems. *Operations research*, 18(6):1225–1228, 1970. 3

[37] 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. 2

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

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

[40] Yehui Tang, Yunhe Wang, Yixing Xu, Dacheng Tao, Chunjing Xu, Chao Xu, and Chang Xu. Scop: Scientific control for reliable neural network pruning. *arXiv preprint arXiv:2010.10732*, 2020. 6

[41] Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Van Gool. Temporal segment networks: Towards good practices for deep action recognition. In *European conference on computer vision*, pages 20–36. Springer, 2016. 2

[42] 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. 2

[43] Jiahui Yu and Thomas Huang. Autoslim: Towards one-shot architecture search for channel numbers. *arXiv preprint arXiv:1903.11728*, 2019. 2

[44] Dejiao Zhang, Haozhu Wang, Mario Figueiredo, and Laura Balzano. Learning to share: Simultaneous parameter tying and sparsification in deep learning. In *International Conference on Learning Representations*, 2018. 2

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

[46] 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. 1, 2, 6, 7