24th European Conference on Artificial Intelligence - ECAI 2020
Santiago de Compostela, Spain

# Tutor-Instructing Global Pruning for Accelerating Convolutional Neural Networks

**Fang Yu**[1,2] and **Li Cui**[1*]

**Abstract.** Model compression and acceleration has recently received ever-increasing research attention. Among them, filter pruning shows a promising effectiveness, due to its merits in significant speedup for inference and support on off-the-shelf computing platforms. Most existing works tend to prune filters in a layer-wise manner, where networks are pruned and fine-tuned layer by layer. However, these methods require intensive computation for per-layer sensitivity analysis and suffer from accumulation of pruning errors. To address these challenges, we propose a novel pruning method, namely **T**utor-**I**nstructing global **P**runing (TIP), to prune the redundant filters in a global manner. TIP introduces Information Gain (IG) to estimate the contribution of filters to the class probability distributions of network output. The motivation of TIP is to formulate filter pruning as a minimization of the IG with respect to a group of pruned filters under a constraint on the size of pruned network. To solve this problem, we propose a Taylor-based approximate algorithm, which can efficiently obtain the IG of each filter by backpropagation. We comprehensively evaluate our TIP on CIFAR-10 and ILSVRC-12. On ILSVRC-12, TIP reduces FLOPs for ResNet-50 by 54.13% with only a drop in top-5 accuracy by 0.1%, which significantly outperforms the state-of-the-art methods.

## 1 INTRODUCTION

Convolutional neural networks (CNNs) have yielded state-of-the-art results on a variety of applications such as image classification [7], object detection [21], and semantic segmentation [23]. However, the deep CNNs bring in prohibitively expensive computational and memory costs, making it a great burden to be deployed on the hardware devices with limited storage and computation resources, especially for the mobile and IoT systems. Therefore, efficient model compression and acceleration, aiming to reduce the number of computations and parameters, enables broader application of deep neural networks.

In this context, some efforts have been made for model compression and acceleration using specialized hardware implementation, including weight pruning [5], parameter quantization [4] and binarization [15]. Though theoretically plausible, the models compressed by these methods require specialized supports to be accelerated. In contrast, filter pruning [9, 1, 10], a.k.a. channel pruning [11] or network slimming [22], converts dense CNNs into smaller compact networks through directly removing filters as a whole. In this case, the compressed models can be well supported by various off-the-shelf computing platforms. In addition, filter pruning can effectively reduce the floating-point operations (FLOPs) and parameters of model, while maintaining the accuracy almost intact. Moreover, filter pruning is orthogonal to other accelerating techniques, which can be further accelerated by other techniques without extra operation. In this work, we focus on filter pruning.

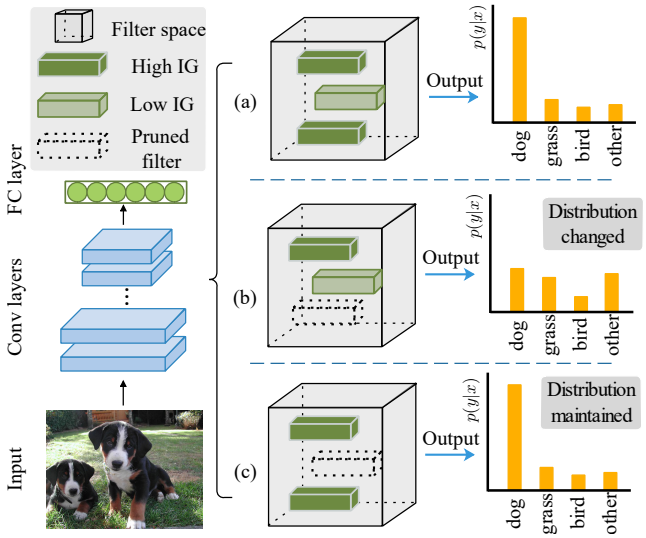


**Figure 1**: An illustration of pruning filters based on Information Gain (IG). (a): The class probability distribution $p(y|x)$ of an uncompressed network output w.r.t. an input image. (b): The class probability distribution $p(y|x)$ after pruning filters with high IG. (c): The $p(y|x)$ after pruning filters with low IG.

Current practices [19, 11, 32] perform filter pruning in a layer-wise manner, where networks are first evaluated for redundancy, then pruned and fine-tuned layer by layer. Unfortunately, these layer-by-layer based pruning approaches suffer from some drawbacks. Firstly, for various CNNs, layer-wise pruning approaches require expensive per-layer sensitivity analysis to determine the number of pruned filters at each layer. Otherwise, a fixed pruning rate applied to all convolutional layers will damage some layers sensitive to pruning, leading to a dramatical drop in performance. Secondly, when applied to extremely deep networks (e.g., ResNet-1001 [8]), the layer-wise methods are inefficient or even impractical to prune and fine-tune the deep CNNs layer by layer. Thirdly, some sophisticated networks have complicated structures, e.g., shortcut connection [7], where some layers have branches connected to other layers. It is intractable for layer-wise methods to handle such networks with complex connectivity. Lastly, but most significantly, to evaluate the redundancy of filter per layer, the layer-wise methods usually prune filters which have

[1] Institute of Computing Technology, Chinese Academy of Sciences, China, email: {yufang, lcui}@ict.ac.cn. * Corresponding author.
[2] University of Chinese Academy of Sciences, China.

little impact on the next layer. However, the errors caused by pruned filters will be accumulated and amplified when propagating through multiple layers, degrading the performance of the pruned networks.

In contrast to pruning filters layer by layer, we propose a global pruning method to prune the redundant filters, which can address the problems mentioned above. More specifically, we first calculate the Information Gain (IG) [26] of filters on the class probability distributions of network output with respect to input samples. The IG of filter quantifies the influence of filter removal on network output, instead of the influence on the next layer. Thus, it will not make substantial negative influence on network performance when pruning the group of filters with the lowest IG across all convolutional layers, as shown in Figure 1. In addition, the computation for IG of filters can be efficiently achieved by our proposed Taylor-based approximate algorithm (TBAA), which is able to be performed on any CNNs. It is noteworthy that the implementation of TBAA relies on a pre-trained prior network (we refer to it as *tutor network*) which provides prior class distributions w.r.t. the same samples (ground truth). In the sequel, we name our pruning method as **T**utor-**I**nstructing global **P**runing (TIP).

Our main contributions are summarized as follows:

- We introduce information gain (IG) to estimate the contribution of filters to class probability distributions, and propose a Taylor-based approximate algorithm (TBAA) to efficiently calculate it.
- Based on IG and TBAA, we propose the tutor-instructing global pruning method (TIP) to prune the redundant filters in CNNs.
- By TIP, we provide a pruning strategy for Residual Networks, which have some constraints for pruning.
- Extensive experiments on CIFAR-10 and ILSVRC-12 demonstrate the effectiveness and efficiency of our TIP.

## 2 RELATED WORKS

**Weight pruning** Weight pruning is an unstructured pruning method that targets at zeroing out some weights in filters. It was successively proposed in optimal brain damage [18] and optimal brain surgeon [6] to prune weights based on the second-order derivative of loss function. Recently, Han et al. [5] proposed an iterative approach to prune weights with small values below a given threshold. Guo et al. [3] proposed dynamic network surgery composed of pruning and splicing. Yu et al. [31] prune weights in filters to minimize the reconstruction error of the final response layer. Unfortunately, these methods only produce the pruned networks with irregular sparsity, which requires specialized library or hardware design to store a large number of indices for efficient speedup.

**Filter pruning** Filter pruning [19, 13, 11, 9, 20] belongs to structured pruning, aiming at producing regular sparsity in networks. In this way, the compressed networks can be accelerated on generic computing platforms without specialized support. In [19], Li et al. prune filters with small $\ell_1$ norm, where per-layer sensitivity analysis is used to determine which layers are sensitive to pruning and how many filters can be safely pruned at each layer. Similarly, a series of layer-wise pruning methods have been proposed. In [13], Hu et al. prune filters based on the average percentage of zero values (APoZ) in output feature maps. He et al. [11] proposed LASSO-based channel selection strategy and least square reconstruction to prune filters. ThiNet [24] was put forward to preserve the filters which minimize reconstruction error of output on each layer. However, sensitivity analysis which adds extensive computation is essential for these methods to achieve good performances. To avoid sensitivity analysis, Molchanov et al. [25] analyzed several greedy criteria for pruning, and proposed saliency estimation for filters based on the sensitivity of loss. More recently, Ding et al. [1] used binary filter search to simultaneously decide the pruning quantity for each layer, and He et al. [10] employed geometric median to recognize the replaceable filters for each layer. On the other hand, some works attempt to impose sparse constraints on filters [30] or the scale of batch-normalization [22] when training networks. These methods do not need sensitivity analysis and can be performed to prune globally. However, these methods require expensive training from scratch when obtaining the compact CNNs. In our work, the proposed TIP can globally prune filters without sensitivity analysis and expensive training. We need only choose a single compression ratio parameter, and TIP can automatically determine the appropriate filters for each layer.

**Other methods** Apart from network pruning, there are some other works on model compression and acceleration. Network quantization [4, 27, 15] aims to reduce the number of bits used to represent the weights and gradients. Furthermore, BinaryNet [15] constrains weights and activations as binary bit, which dramatically reduces the model size compared with the full-precision version. However, these methods also require hardware support for new numeric formats. Knowledge distillation [12] targets at improving a small compact network through transferring knowledge from a large teacher network, whose drawback is the need to select or manually design the decent compact network.

## 3 THE PROPOSED APPROACH

### 3.1 Preliminaries

**Formulation** Given a convolutional neural network with parameters $\mathbf{W} = \{\mathbf{W}_1, \mathbf{W}_2, ..., \mathbf{W}_L\}$ and a dataset $\mathcal{D} = \{(x_1, y_1), ..., (x_N, y_N)\}$ composed of N input-output pairs, the class probability distribution that maps inputs $x$ to their corresponding outputs $y$ can be denoted as $p(y|x, \mathbf{W})$. The parameter of the $l$-th convolutional layer is represented as $\mathbf{W}_l \in \mathbb{R}^{N_{l+1} \times N_l \times k_w \times k_h}$, where $N_{l+1}$ is the number of filters, $N_l$ is the number of filters' kernels, and $k_w \times k_h$ is the kernel size of filters. The $j$-th filter in $\mathbf{W}_l$ can be represented as $\mathbf{w}_l^j \in \mathbb{R}^{N_l \times k_w \times k_h}$, whose removal will result in deleting its corresponding bias term, batch-normalization term, output feature map and related input kernels at the next layer.

We use a binary mask $\boldsymbol{\beta} \in \{0, 1\}^m$ representing the condition of all filters, where 1 denotes filter is preserved, 0 denotes filter is removed and $m$ is the total number of filters. For image classification, the task of filter pruning can be formulated to minimize the cross-entropy loss on the dataset under sparsity constraints on filters, written as:

$$\begin{aligned} \min_{\mathbf{W}^*} \quad & -\sum\nolimits_{i=1}^{N} y_i \log p(y|x_i, \mathbf{W}^*) \\ \text{s.t.} \quad & \mathbf{W}^* = \boldsymbol{\beta} \odot \mathbf{W}, \\ & \|\boldsymbol{\beta}\|_0 = (1-\gamma)m, \beta_i \in \{0, 1\}, i = 1, 2, ..., m, \end{aligned} \tag{1}$$

where $\mathbf{W}^*$ is the remaining filters after $\mathbf{W}$ being pruned with filters whose $\beta$ is equal to 0, $\odot$ is multiplication with broadcasting, $\|\cdot\|_0$ is standard $\ell_0$ norm and $\gamma$ is the designated pruning rate.

**Information gain** Information gain [26] is a popular statistic tool for feature selection. It quantifies the change in information entropy of random variable $T$ from a prior state to a state that takes some condition as given. Formally, it is defined as:

$$\mathrm{IG}(T, a) = \mathbb{H}(T) - \mathbb{H}(T|a), \tag{2}$$

where $\mathbb{H}(\cdot)$ denotes entropy, and $\mathbb{H}(\cdot|\cdot)$ denotes conditional entropy of $T$ given the value of attribute $a$.

### 3.2 Information Gain of Filters

The optimization problem (1) is non-convex and NP-hard, so there is no effective way to solve such $\ell_0$ norm optimization problem. An alternative approach is to greedily prune an amount of $\gamma m$ filters with the least impact on network across all layers. Information gain can quantify how much information is changed about the class probability distribution $p(y|x)$ of network output given the condition of filters being removed from network. Importantly, the higher information gain of a certain filter, the more information is gained by this filter, which quantifies the more contribution of this filter to network output. In contrast, the filters with the lowest IG carry little information, whose removal will not potentially incur much information loss. Figure 1 provides an illustration of pruning filter based on IG.

We use the concept of IG to reformulate the problem (1). Considering that the ground truth $y_i$ has no effect on output distribution $p(y|x, \mathbf{W})$, we neglect $y_i$, and reformulate (1) as a minimization with respect to IG of the pruned filters on $p(y|x, \mathbf{W})$ under a sparsity constraint on filters:

$$\begin{aligned} \min_{\overline{\mathbf{W}}} \quad & \mathrm{IG}[p(y|x, \mathbf{W}), \overline{\mathbf{W}}] \\ \text{s.t.} \quad & \overline{\mathbf{W}} = \mathbf{W} - \mathbf{W}^*, \mathbf{W}^* = \boldsymbol{\beta} \odot \mathbf{W}, \\ & \|\boldsymbol{\beta}\|_0 = (1-\gamma)m, \beta_i \in \{0,1\}, i = 1, 2, ..., m, \end{aligned} \tag{3}$$

where $\overline{\mathbf{W}}$ is a set of pruned filters in the network. The IG objective in (3) can be written as:

$$\begin{aligned} \mathrm{IG}[p(y|x, \mathbf{W}), \overline{\mathbf{W}}] &= \mathbb{H}[p(y|x, \mathbf{W})] - \mathbb{H}[p(y|x, \mathbf{W})|\overline{\mathbf{W}} = \mathbf{0}] \quad (4) \\ &= \mathbb{H}[p(y|x, \mathbf{W})] - \mathbb{H}[p(y|x, \mathbf{W}^*)], \quad (5) \end{aligned}$$

where $\mathbb{H}[p(y|x, \mathbf{W})]$ is the entropy of output distribution when keeping all filters, and $\mathbb{H}[p(y|x, \mathbf{W}^*)]$ is the entropy of output distribution when pruning the set of filters $\overline{\mathbf{W}}$.

Compared with (1), the constrained optimization (3) is easier to solve as it does not require expensive training trial to minimize the cross-entropy loss on the dataset. However, the optimization (3) is still NP-hard. To relax it, we assume filters in network are *i.i.d.* random variables which are determined by distributions of training data, randomness of training data and randomness of training algorithm. Under this assumption, the IG objective in (3) can be computed by the sum of IG of individual filter. Hence, the sparsity-constrained optimization (3) can be rewritten as:

$$\begin{aligned} \min_{\mathbf{w}} \quad & \textstyle\sum_{\mathbf{w}} \mathrm{IG}[p(y|x, \mathbf{W}), \mathbf{w}] \\ \text{s.t.} \quad & \mathbf{w} \in \overline{\mathbf{W}}, \overline{\mathbf{W}} = \mathbf{W} - \mathbf{W}^*, \mathbf{W}^* = \boldsymbol{\beta} \odot \mathbf{W}, \\ & \|\boldsymbol{\beta}\|_0 = (1-\gamma)m, \beta_i \in \{0,1\}, i = 1, 2, ..., m, \end{aligned} \tag{6}$$

where $\mathbf{w}$ is an individual pruned filter from $\overline{\mathbf{W}}$. The IG objective in (6) can be written as:

$$\mathrm{IG}[p(y|x, \mathbf{W}), \mathbf{w}] = \mathbb{H}[p(y|x, \mathbf{W})] - \mathbb{H}[p(y|x, \mathbf{w}^*)], \tag{7}$$

where $\mathbf{w}^*$ is the remaining filters after an individual filter $\mathbf{w}$ is removed.

### 3.3 Approximate Algorithm for IG Computation

Optimization problem (6) can be solved by exhaustive search to find an amount of $\gamma m$ individual filters with the lowest IG via Eq. (7). However, the computation of Eq. (7) is prohibitively expensive, as the IG of a filter is measured by accumulating the entropy difference between the original network output $p(y|x, \mathbf{W})$ and the pruned network output $p(y|x, \mathbf{w}^*)$ over the entire dataset. Moreover, the time complexity of exhaustive search is linear to the number of filters. Hence, it is intolerable when the pruned network is very deep. To efficiently solve the problem (6), we propose a Taylor-based approximate algorithm (TBAA) for IG computation.

We assume that a prior class probability distribiton w.r.t. the same input sample is $p_t(y|x)$, which is provided by a pre-trained tutor network. The entropy in Eq. (7) can be efficiently obtained by borrowing from the property of entropy[3]. We convert the form of entropy in Eq. (7):

$$\mathbb{H}[p(y|x, \mathbf{W})] = H_{p_t}[p(y|x, \mathbf{W})] - D_{KL}[p(y|x, \mathbf{W})||p_t(y|x)], \tag{8}$$

$$\mathbb{H}[p(y|x, \mathbf{w}^*)] = H_{p_t}[p(y|x, \mathbf{w}^*)] - D_{KL}[p(y|x, \mathbf{w}^*)||p_t(y|x)], \tag{9}$$

where $H_{p_t}$ is the cross entropy with $p_t(y|x)$ and $D_{KL}$ is Kullback-Leibler divergence. Combining Eq. (7-9), we have

$$\begin{aligned} \mathrm{IG}[p(y|x, \mathbf{W}), \mathbf{w}] =& \{H_{p_t}[p(y|x, \mathbf{W})] - H_{p_t}[p(y|x, \mathbf{w}^*)]\} \\ & -\{D_{KL}[p(y|x, \mathbf{W})||p_t(y|x)] - D_{KL}[p(y|x, \mathbf{w}^*)||p_t(y|x)]\} \\ =& \Delta H_{p_t} - \Delta D_{KL}, \end{aligned} \tag{10}$$

where $\Delta$ denotes the value change. For better readability, we use $H_{p_t}(\cdot)$ to denote the cross entropy $H_{p_t}[p(y|x, \cdot)]$ and $D_{KL}(\cdot)$ to denote KL-divergence $D_{KL}[p(y|x, \cdot)||p_t(y|x)]$ in Eq. (10). Considering an individual filter $\mathbf{w}$, $\Delta H_{p_t}$ and $\Delta D_{KL}$ can be estimated by approximating $H_{p_t}(\mathbf{W})$ and $D_{KL}(\mathbf{W})$ with first-order Taylor expansion near $\mathbf{w}$=$\mathbf{0}$, respectively. Specifically, $\Delta H$ is simplified by expanding $H_{p_t}(\mathbf{W})$:

$$\begin{aligned} \Delta H_{p_t} &= \left( H_{p_t}(\mathbf{w}^*) + \frac{\partial H_{p_t}(\mathbf{W})}{\partial \mathbf{w}}^{\mathrm{T}} \mathbf{w} + R_1(\mathbf{w} = \mathbf{0}) \right) - H_{p_t}(\mathbf{w}^*) \\ &\approx \frac{\partial H_{p_t}(\mathbf{W})}{\partial \mathbf{w}}^{\mathrm{T}} \mathbf{w}, \end{aligned} \tag{11}$$

where $R_1(\mathbf{w} = \mathbf{0})$ is 1-th order remainder term which can be neglected. Similarly, $\Delta D_{KL}$ is simplified as:

$$\Delta D_{KL} \approx \frac{\partial D_{KL}(\mathbf{W})}{\partial \mathbf{w}}^{\mathrm{T}} \mathbf{w}. \tag{12}$$

With the above Eq. (10-12), we obtain the IG of filter:

$$\mathrm{IG}[p(y|x, \mathbf{W}), \mathbf{w}] \approx \left( \frac{\partial H_{p_t}(\mathbf{W})}{\partial \mathbf{w}} - \frac{\partial D_{KL}(\mathbf{W})}{\partial \mathbf{w}} \right)^{\mathrm{T}} \mathbf{w}. \tag{13}$$

Equation (13) reveals that the IG of a filter can be approximated by matrix product of its weight value $\mathbf{w}$ and the first-order derivative $\frac{\partial H_{p_t}(\mathbf{W})}{\partial \mathbf{w}} - \frac{\partial D_{KL}(\mathbf{W})}{\partial \mathbf{w}}$ w.r.t. this filter. We have more insights into this equation. Firstly, the weight value of filter can imply an indirect effect on the output distribution. For example, a filter with larger weight value has a more significant impact on its output feature map, indirectly affecting the subsequent layers by forward propagation. Secondly, $\frac{\partial H_{p_t}(\mathbf{W})}{\partial \mathbf{w}} - \frac{\partial D_{KL}(\mathbf{W})}{\partial \mathbf{w}}$ reveals the *saliency* of filter. As this derivative is derived from the Taylor expansion near $\mathbf{w}$=$\mathbf{0}$, it

[3] $\mathbb{H}[P] = -\sum_x p(x) \log [\frac{p(x)}{q(x)} q(x)] = -\sum_x p(x) \log q(x) - \sum_x p(x) \log \frac{p(x)}{q(x)} = H_Q[P] - D_{KL}[P||Q]$

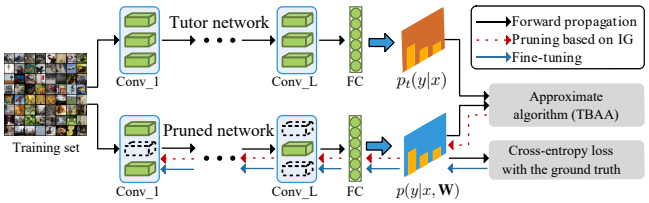


**Figure 2**: The workflow of TIP. The IG of each filter is computed over the entire training set by TBAA. TIP then prunes filters with the lowest average IG and fine-tunes the remaining filters by SGD, where pruning and fine-tuning are iterative.

can directly reflect the removal influence of a filter on the output distribution by backpropagation. For instance, if one filter has a large weight value but a small derivative value, this filter obtains low IG on $p(y|x, \mathbf{W})$ and can be pruned from the network.

## 3.4 Implementation Details

In this subsection, we point out several implementation details of our TIP with respect to Eq. (13). The workflow of our TIP is illustrated in Fig. 2.

**Information gain loss** To obtain the first-order derivative of Eq. (13) w.r.t. all filters, we define the information gain loss:

$$\mathcal{L}_{IG} = H_{p_t}[p(y|x, \mathbf{W})] - D_{KL}[p(y|x, \mathbf{W})||p_t(y|x)], \qquad (14)$$

where $p_t(y|x)$ is the ground truth. It can be converted from the logit output $z$ of tutor network into logit probability by softmax function

$$p_k(y|x) = \frac{\exp(z_k)}{\sum_{c=1}^{C} \exp(z_c)}, \qquad (15)$$

where $k$ is class index and $C$ is the total number of classes. After performing error backpropagation of $\mathcal{L}_{IG}$, we can easily obtained the IG of all filters via Eq. (13). Note that $\mathcal{L}_{IG}$ is only used to obtain IG, but not used to optimize the parameters of filters.

**Selection of the tutor network** In the pruning pipeline, the tutor network participates in computation of IG to determine which filters to be pruned. In general, a larger and deeper tutor network can provide a more accurate class distribution about ground truth, but the more inference wall-clock time is required. To balance pruning quality and pruning cost, we select the pre-trained original network of pruned network as its tutor network. The effect for selection of tutor network will be discussed in Section 4.4.

**Dimension conversion** Note that filter $\mathbf{w}$ is a tensor of $\mathbb{R}^{N_l \times k_w \times k_h}$, which is necessary to be flattened to the vector of $\mathbb{R}^{N_l k_w k_h \times 1}$. In this way, the result of IG in Eq. (13) is a single value rather than a high dimensional tensor, which avoids extra computational conversion.

**Averaging IG over entire training set** The measurement of IG needs to be evaluated over the entire training set since IG estimated by a small amount of data will have an information bias on data. Thus, we average Eq. (13) over the entire training set to decide which filters to be pruned.

**Pruning strategy** Our pruning approach takes a pre-trained network as input. To prune a ratio $\gamma$ of filters with the least impact on this network, we consider the information change of remaining filters after pruning, and thus adopt a prune-finetune iterative strategy. To start with, we traverse all the training samples, compute the average IG of each filter, and remove $p\%$ filters with the lowest average IG across all layers. We then re-organize the remaining filters to a smaller network and transfer the corresponding parameters into it. Afterwards, we fine-tune this re-organized network by stochastic gradient descent (SGD), during which we re-evaluate the average IG of remaining filters via Eq. (13). We repeat the pruning and fine-tuning process until the pruned network is converged or the maximum iteration epoch is achieved. Algorithm 1 summarizes the overall procedure.

**Algorithm 1:** Algorithm Description of TIP

**Input**: Training set $\mathcal{D}$, batch size $S$, pruning rate $\gamma$, pruning granularity $p\%$, maximum epoch $T$;
**Output**: The compact pruned network;

$t := 1$;
Compute average IG of each filter over $\mathcal{D}$ via Eq. (13);
Remove $p\%$ filters with the lowest average IG;
Construct a smaller network and transfer parameters;
**repeat**
  **for** $i \leftarrow 0$ **to** $\frac{|\mathcal{D}|}{S}$ **do**
    Obtain the logit output $z$ of both networks;
    Obtain $p(y|x, \mathbf{W})$ and $p_t(y|x)$ via Eq. (15);
    Obtain $\mathcal{L}_{IG}$ via Eq. (14);
    Obtain $\frac{\partial \mathcal{L}_{IG}}{\partial \mathbf{w}}$ for all filters by backpropagation;
    Compute average IG of each filter via Eq. (13);
    Fine-tune the re-organized network by SGD;
  **end**
  **if** $t \times p\% < \gamma$ **then**
    Remove $p\%$ filters with the lowest average IG;
    Construct a smaller compact network and transfer the corresponding parameters;
  **end**
  $t := t + 1$;
**until** convergence or $t$ reaches the maximum epoch $T$;

## 3.5 Pruning Residual Networks

For some classical plain CNNs, e.g., AlexNet [17] and VGGNet [29], they are simply stacked by the convolutional layers. Pruning these networks will not affect the connectivity patterns of architecture. However, for the advanced networks, i.e., Residual Networks [7], some constraints for pruning must be considered. For instance, feature maps of each residual block are element-wise added to form the *residual flows*. Therefore, the order and size of residual flows should be consistent with the feature maps of each block. In addition, the size of residual flows is expanded by *linear projection* or *subsampling* in different stages on ResNet, so it is necessary to consider the size matching of residual flows during pruning process. In practice, the layer-wise methods [19, 10] usually choose to prune the internal layers in each block to avoid pruning such constrained layers. In addition to the internal layers, our TIP can also prune the constrained layers on ResNet. We take ResNet-50 and ResNet-56 to illustrate our pruning strategy on ResNet with linear projection and with subsampling, respectively.

ResNet-50 is a typical ResNet, which contains four stages of residual blocks with different size. Note that the size of residual flows in each stage is matched by the linear projection $\mathbf{y} = W\mathbf{x}$. Due to this structure, it is flexible to expand or shrink the residual flows in each stage. Therefore, TIP removes the entire residual flow with the lowest IG in the same stage, resulting in removal of corresponding filters and feature maps along this shortcut flow, as shown in Fig. 3(a). More

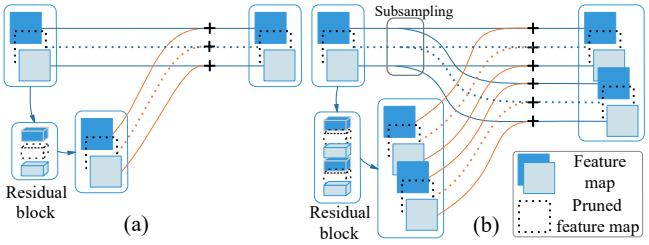


**Figure 3**: An illustration of pruning ResNet. For ResNet with linear projection, TIP removes the entire residual flows like (a) in the same stage, resulting in removal of corresponding filters in each block and feature maps. For ResNet with subsampling, TIP removes the entire residual flows, including the expanded flows by subsampling in all stages like (b).

specifically, if all filters along the residual flow in the same position (i.e., filters that produce feature maps added to the same shortcut flow) are the group with the lowest IG, this flow is considered to make little contribution to the output distribution. Hence, this residual flow and corresponding filters are safely deleted, which will not cause the error and disorder in residual mechanism. Meanwhile, the corresponding parameters in linear projection are removed to match the size change of residual flows.

ResNet-56 is performed with subsampling to expand the residual flows in each stage. In this case, the size of residual flows at the latter stage must be twice as large as the former. Owing to this constraint, TIP directly removes the entire residual flow whose filters are the group with the lowest IG, including the expanded flows by subsampling in all stages, as shown in Fig. 3(b).

### 3.6 Theoretical Acceleration Analysis

We provide a brief analysis to illustrate the effect of architecture hyper-parameters on the floating-point operations (FLOPs) consumption (which we will use in our experimental results to compare the various approaches). For a CNN model, the total number of FLOPs at the $l$-th convolutional layer with the batch size $B$ can be given as

$$FLOPs = (2 \times c_{in} k_w k_h) \times w_o h_o c_o \times B, \tag{16}$$

where $(c_{in}, w_{in}, h_{in})$ is the size of input tensor, $(N_l, k_w, k_h)$ is the size of filter and $(c_o, w_o, h_o)$ is the size of output tensor.

If a ratio $P_l$ of filters at the $l$-th convolutional layer are pruned from network, the reduction of FLOPs is

$$FLOPs_{reduce} = (2 \times c_{in} k_w k_h) \times w_o h_o c_o P_l \times B. \tag{17}$$

## 4 EXPERIMENT

We implement our method using Pytorch. The effectiveness validation is performed on two datasets, CIFAR-10 [16] and ILSVRC-12 [28]. CIFAR-10 contains 50k training images and 10k validating images, which are categorized into 10 classes for image classification. Compared with CIFAR-10, ILSVRC-12 is a larger scale image classification dataset, which comprises 1.28 million images from 1k categories for training and 50k images for validation.

### 4.1 Efficacy of Pruning via Information Gain

In this subsection, we first investigate the efficacy about pruning filters with IG and compare it with other pruning criteria using AlexNet

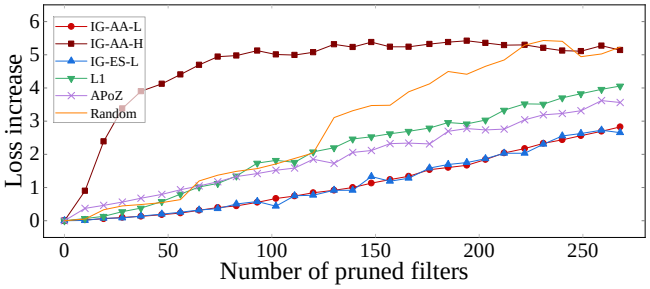


**Figure 4**: Comparison of different filter pruning criteria on AlexNet.

on ILSVRC-12. AlexNet is a plain CNN with five stacked convolutional layers containing a total of 1,152 filters, which is convenient to evaluate various pruning criteria. To illustrate the effectiveness of our method, considering that the cross-entropy loss can reflect the class distributions of network output, we record the increase of cross-entropy loss after globally pruning filters one by one without fine-tuning. We refer to pruning filters with the lowest IG and highest IG computed by our TBAA as IG-AA-L and IG-AA-H, respectively. For comparison, we conduct the following experiments: pruning filters with the lowest IG computed by exhaustive search (IG-ES-L), with the least $\ell_1$ norm (L1) [19], with the least APoZ (APoZ) [13] and random pruning (Random). Notably, as APoZ and IG methods are data-dependent, we use 100,000 random samples to determine the contribution of filters. We perform these methods to globally delete filters one by one without fine-tuning. The results are evaluated on the validation set of ILSVRC-12.

Figure 4 shows the increase of cross-entropy loss with respect to the number of pruned filters. We have three observations. Firstly, we observe that IG-AA-L and IG-ES-L incur the least loss increase, while IG-AA-H has the most significant loss increase among all results. It reveals that IG of filters can accurately measure the contribution of filters to network output. Secondly, the curve of IG-AA-L is similar to IG-ES-L, demonstrating that IG computed by our approximate algorithm has a comparable efficacy with the exhaustive search. Notably, our approximate algorithm is more efficient and time-saving. Thirdly, IG-AA-L outperforms the popular pruning criteria APoZ and L1. Their motivation is to assume that the removal of filters has a minimal impact on the next layer. However, the pruning errors are actually accumulated and amplified when propagating forward to the network output. In contrast, IG of filters directly quantifies their effect on the class distributions of network output, so pruning filters with the lowest IG can accurately remove the most redundant filters, minimizing the damage to the pruned networks.

### 4.2 Comparison with State-of-the-art Methods

We evaluate the effectiveness of our TIP on CIFAR-10 [16] and ILSVRC-12, and compare our results with the following state-of-the-art methods: L1-pruned [19], AFP [2], FPGM [10], VCNNP [32], AOFP [1] and GAL [20]. L1-pruned and VCNNP are traditional layer-wise pruning methods which are based on sensitivity analysis. AFP, FPGM and AOFP use various approaches to determine pruning quantity of filter for each layer but are still layer-by-layer. GAL utilizes generative adversarial learning for structural pruning in an end-to-end manner.

**Results on CIFAR-10** We experiment with VGG-16, ResNet-56, ResNet-110 and DenseNet-40 [14] on CIFAR-10 [16]. We train

the baseline models from scratch for 200 epochs to ensure convergence, and select them as the tutor networks. For pruning, we set the maximum epoch to 90 and batch size to 128. The learning rate is initialized to 0.01 and divided by 10 per 20 epochs. The momentum and weight decay are set to 0.9 and 0.0005, respectively. The pruning granularity $p\%$ is set to 1%. Note that the pruning rate $\gamma$ controls the ratio of pruned filters in initial filters. We use TIP-$\gamma$ to denote our results under different pruning rates.

The classification accuracy, FLOPs and the number of parameter of various pruning methods are reported in Table 1. As shown in Table 1, our TIP outperforms the state-of-the-art methods on CIFAR-10. For example, for pruning VGG-16, compared with L1-pruned-A, VCCNP and AOFP-A4, TIP-80% achieves a higher accuracy of 93.48% and obtains the higher reduction in FLOPs of 72.45% and parameters of 94.90%. Similarly, for pruning ResNet-56, ResNet-110 and DenseNet-40, our TIP also achieves a higher accuracy than these methods, while obtains more reduction in FLOPs and number of parameters. Compared with these state-of-the-art methods, our TIP utilizes IG to globally explore the least contributing filters, whose removal leads to little performance degradation, thus producing superior results.

**Table 1**: Pruning results of VGG-16, ResNet-56, ResNet-110 and DenseNet-40 on CIFAR-10. In all tables, our results are denoted as TIP-$\gamma$, where $\gamma$ is the pruning rate of filter. PR represents the pruned rate. "n.p.f." means TIP does not prune the residual flow of ResNet. "-" means the corresponding result is not reported. M/B means million/billion.

| Model | Accu.(%) | FLOPs(PR) | #Param.(PR) |
|---|---|---|---|
| VGG-16 | 93.73 | 314.16M(0%) | 14.72M(0%) |
| L1-pruned-A[19] | 93.40 | 206.00M(34.20%) | 5.40M(64.00%) |
| VCNNP[32] | 93.18 | 190.00M(39.10%) | 3.92M(73.34%) |
| AOFP-A4[1] | 93.47 | 108.00M(65.27%) | - |
| **TIP-80%** | **93.48** | **86.53M(72.45%)** | **0.75M(94.90%)** |
| AOFP-A5[1] | 93.28 | 77.00M(75.27%) | - |
| **TIP-82%** | **93.34** | **71.94M(77.10%)** | **0.70M(95.24%)** |
| AFP-E[2] | 92.94 | 63.70M(79.69%) | - |
| **TIP-84%** | **93.22** | **62.76M(80.02%)** | **0.46M(96.88%)** |
| ResNet-56 | 93.92 | 126.58M(0%) | 0.85M(0%) |
| L1-pruned-B[19] | 93.06 | 90.90M(27.60%) | 0.73M(13.70%) |
| TIP-15%(n.p.f.) | 93.78 | 81.12M(35.91%) | 0.70M(17.65%) |
| GAL-0.6[20] | 93.38 | 78.30M(37.60%) | 0.75M(11.80%) |
| **TIP-15%** | **94.04** | **76.56M(40.30%)** | **0.65M(23.25%)** |
| FPGM-only-40%[10] | 93.49 | 59.40M(52.60%) | - |
| **TIP-28%** | **93.59** | **58.07M(54.12%)** | **0.46M(45.89%)** |
| GAL-0.8[20] | 91.58 | 49.99M(60.20%) | 0.29M(65.90%) |
| **TIP-33%** | **92.99** | **48.85M(61.41%)** | **0.27M(68.23%)** |
| ResNet-110 | 94.47 | 255.01M(0%) | 1.73M(0%) |
| GAL-0.1[20] | 93.59 | 205.70M(18.70%) | 1.65M(4.10%) |
| TIP-30%(n.p.f.) | 93.42 | 161.20M(36.79%) | 0.76M(56.97%) |
| L1-pruned-B[19] | 93.30 | 155.00M(38.60%) | 1.16M(32.40%) |
| **TIP-30%** | **93.76** | **151.36M(40.64%)** | **0.67M(61.27%)** |
| DenseNet-40 | 94.25 | 290.11M(0%) | 1.06M(0%) |
| VCNNP[32] | 93.16 | 156.00M(44.78%) | 0.42M(59.67%) |
| **TIP-30%** | **93.67** | **119.02M(58.97%)** | **0.38M(64.15%)** |

**Results on ILSVRC-12** We experiment with ResNet-34 and ResNet-50 on ILSVRC-12. We use the pre-trained models provided by PyTorch[4] as baseline and tutor networks. We set the maximum epoch to 50 and use a mini-batch 128 to fine-tune. The learning rate is initialized to 0.1 and divided by 10 per 10 epochs. Other settings are the same as pruning on CIFAR-10.

[4] https://pytorch.org/docs/stable/torchvision/models.html

**Table 2**: Pruning results of ResNet-34 and ResNet-50 on ILSVRC-12.

| Model | Top-1(%) | Top-5(%) | FLOPs(PR) |
|---|---|---|---|
| ResNet-34 | 73.30 | 91.42 | 3.68B(0%) |
| L1-pruned-B[19] | 72.17 | - | 2.76B(24.20%) |
| FPGM-only-30%[10] | 72.54 | 91.13 | 2.17B(41.10%) |
| **TIP-30%** | **72.63** | **91.24** | **2.12B(42.39%)** |
| ResNet-50 | 76.15 | 92.87 | 4.12B(0%) |
| VCNNP[32] | 75.20 | 92.10 | 2.47B(40.00%) |
| TIP-30%(n.p.f.) | 75.38 | 92.67 | 2.40B(41.75%) |
| FPGM-only-30%[10] | 75.59 | 92.63 | 2.38B(42.20%) |
| GAL-0.5[20] | 71.95 | 90.94 | 2.33B(43.44%) |
| **TIP-30%** | **75.81** | **92.98** | **2.31B(43.93%)** |
| FPGM-only-40%[10] | 74.83 | 92.32 | 1.92B(53.50%) |
| **TIP-40%** | **75.23** | **92.77** | **1.89B(54.13%)** |

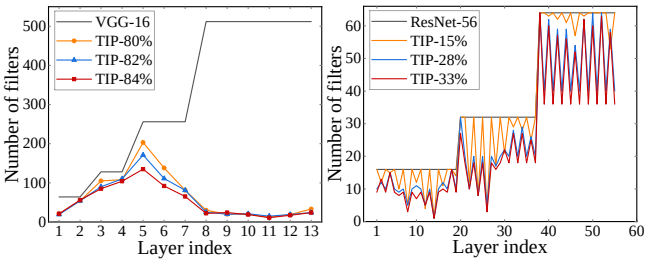


**Figure 5**: Number of remaining filters at each layer. Left: VGG-16 on CIFAR-10. Right: ResNet-56 on CIFAR-10.

In Table 2, we report the top-1/5 classification accuracy and FLOPs of various filter pruning methods. Results in Table 2 show that our TIP achieves the state-of-the-art results on ILSVRC-12. For example, for pruning the ResNet-50, compared with FPGM-only-40%, our TIP-40% achieves higher top-1 accuracy (75.23% vs. 74.83%) and top-5 accuracy (92.77% vs. 92.32%) with a higher reduction in FLOPs (54.13% vs. 53.50%). The results on two benchmarks indicate our TIP can produce more compact pruned networks with better performance than the state-of-the-art methods. Besides, pruning residual flow can yield better performance than not pruning residual flow in ResNet (denoted as n.p.f.). It reveals the residual flows exist redundancy, some of which can be safely removed by TIP.

## 4.3 Pruned Structure Analysis

We visualize the pruned structures of VGG-16 and ResNet-56 on CIFAR-10, as shown in Fig. 5. For VGG-16, we learn that filters at some shallow layers (layer 2, 3, 4, 5) are less pruned than those at the subsequent layers. This finding is consistent with the sensitivity analysis in [19] and [13], where they found that the shallow layers of VGG-16 on CIFAR-10 are more sensitive to pruning. We infer that at the shallow layers, the receptive field of filters is relatively small. These filters mainly extract the low-level features of input images (e.g., edge, corner), which are more important for classification, so these filters obtain higher IG on the output distribution. In contrast, considerable amount of filters are at the deep layers of VGG-16, and their receptive field cover the whole images from CIFAR-10. These filters capture the structured information and semantic context of images. However, these features are numerous and low-resolution, some of which are redundant for classification. The corresponding

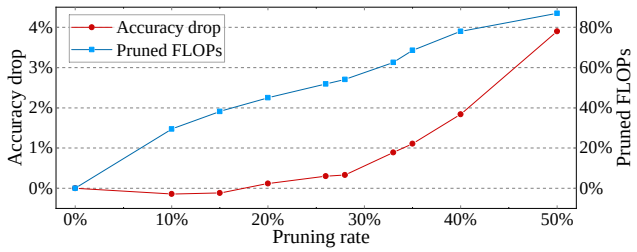


**Figure 6**: Accuracy drop for ResNet-56 on CIFAR-10 with varying pruning rates.

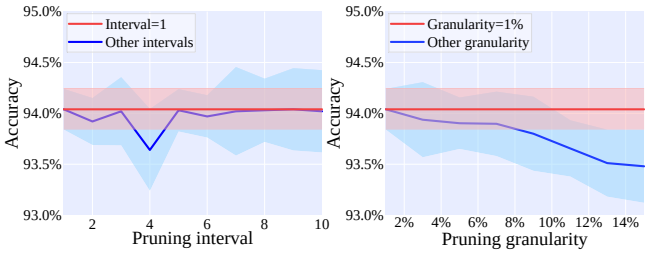


**Figure 7**: Accuracy for ResNet-56 on CIFAR-10 regarding pruning interval (Left) and pruning granularity (Right). Solid line and shadow denote the mean value and standard deviation, respectively.

filters obtain lower IG and are pruned from the network.

Different from the pruning pattern of VGG-16, considerable filters at the shallow layers are pruned on ResNet-56, as shown on the right of Fig. 5. We conjecture that due to the introduction of the residual mechanism, the low and high level semantic features can be combined for further analysis at these deeper layers. The filters at the deep layers obtain higher IG and thus are preserved. Moreover, we find that the first layers of each stage (layer 20 and 38) are preserved with relatively more filters, which is also consistent with the sensitivity analysis [19]. It can be considered that the size increase of residual flows needs more precise residual errors, which leads to filters at the first blocks in each stages with higher IG and prevents them from removal. For instance, at the layer 20, TIP-28% has larger size of residual flows than TIP-33% (10 vs. 9), and TIP-28% is preserved with more filters at the layer 20 (32 vs. 27).

To test the practical speedup of pruned structures, we measure the forward time of the pruned ResNet-56 on CIFAR-10 on one GTX2080Ti GPU with a batch size of 128. In Table 3, we report the practical speedup rate and theoretical speedup rate (i.e., pruned rate of FLOPs). We find that there is a gap between practical speedup and theoretical speedup. We infer this gap may come from the limitation of IO delay, buffer switch, and parallelization bottleneck on intermediate layers.

**Table 3**: Execution time for ResNet-56 on CIFAR-10. All models are tested on a Nvidia GTX 2080Ti GPU with a batch size of 128.

| Model | Measured time(ms) | Practical speedup rate | Theoretical speedup rate |
|---|---|---|---|
| ResNet-56 | 27.52 | - | - |
| TIP-15% | 17.22 | 37.43% | 40.30% |
| TIP-28% | 14.82 | 46.15% | 54.12% |
| TIP-33% | 13.05 | 52.58% | 61.41% |

**Table 4**: Accuracy and wall clock time for pruning ResNet-56 on CIFAR-10 by different tutor networks. In this table, "min" means minute.

| Tutor network (Accu.) | Pruned network (Accu.) | Wall clock time |
|---|---|---|
| ResNet-20 (92.20%) | 93.01% | **45min** |
| ResNet-32 (93.86%) | 93.56% | 57min |
| ResNet-56 (93.92%) | 94.04% | 87min |
| ResNet-110 (94.47%) | **94.16%** | 212min |

## 4.4 Parameter Study

**Effect of tutor network** In the previous experiments, the tutor network of all pruned networks were set as their pre-trained networks. To investigate the effect of tutor network, we use ResNet-56 with a pruning rate of $\gamma = 15\%$ on CIFAR-10 as baseline. We attempt to use pre-trained ResNet-20, ResNet-32 and ResNet-110 as tutor network to prune ResNet-56. The pruning setting is the same as baseline. We record the accuracy of pruned ResNet-56 and wall clock time of pruning process, as shown in Table 4. It reveals that the more powerful the tutor network, the higher the performance of pruned network is, but the more pruning cost it takes.

**Varying pruning rates** To study the effect of pruning rate on TIP, we prune ResNet-56 on CIFAR-10 with a pruning rate $\gamma$ varying from 0% to 50%. In Fig. 6, we report the accuracy drop and pruned FLOPs under different pruning rates. We observe when the pruning rate is less than 18%, the compressed network performs better than the uncompressed one, which can be considered as the regularization effect introduced by filter pruning. As the pruning rate increases, the compressed network have a more speedup, but its accuracy also drops, which indicates there is a trade-off between accuracy and speedup in reality.

**Effect of pruning interval** The pruning interval controls how many epochs our TIP conducts a pruning operation at the end of fine-tuning phase. In our experimental setting, the pruning interval is set to 1 by default. To study the influence of pruning interval, we attempt to change the pruning interval from 1 to 10. We use ResNet-56 on CIFAR-10 with a pruning rate of $\gamma = 15\%$ and interval $= 1$ as baseline, and results are averaged over 3 random seeds. As shown on the left of Fig. 7, the fluctuation in accuracy along with different interval is less than 0.5%. This result reveals the performance of pruned network is not sensitive to pruning interval.

**Effect of pruning granularity** In all experiments, the pruning granularity $p\%$ was set to 1%, which controls ratio of per pruning. To study its effect, we use ResNet-56 with a pruning rate of $\gamma = 15\%$ and $p\% = 1\%$ as baseline. We change $p\%$ from 1% to 15%, and results are averaged over 3 random seeds. Note that, the pruning granularity of 15% means *single-shot* global pruning for ResNet-56 under $\gamma = 15\%$. As shown on the right of Fig. 7, we find that although by a slight margin, pruning filters with a smaller pruning granularity yields better performances.

## 5 CONCLUSION

In this work, we have proposed TIP to effectively prune the redundant filters in CNNs. Without the need of per-layer sensitivity analysis, TIP calculates IG of filters by our proposed Taylor-based approximate algorithm (TBAA) to quantify the contribution of filters to class probability distribution, and iteratively prunes filters with the lowest IG. Extensive experiments have shown the outstanding performances of our proposed method.

## ACKNOWLEDGEMENTS

The paper is supported by the National Natural Science Foundation of China (NSFC) under Grant No. 61672498 and the National Key Research and Development Program of China under Grant No. 2016YFC0302300.

## REFERENCES

[1] Xiaohan Ding, Guiguang Ding, Yuchen Guo, Jungong Han, and Chenggang Yan, 'Approximated oracle filter pruning for destructive cnn width optimization', in *International Conference on Machine Learning*, pp. 1607–1616, (2019).

[2] Xiaohan Ding, Guiguang Ding, Jungong Han, and Sheng Tang, 'Auto-balanced filter pruning for efficient convolutional neural networks', in *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 6797–6804, (2018).

[3] Yiwen Guo, Anbang Yao, and Yurong Chen, 'Dynamic network surgery for efficient dnns', in *Advances in Neural Information Processing Systems*, pp. 1379–1387, (2016).

[4] Song Han, Huizi Mao, and William J Dally, 'Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding', in *International Conference on Learning Representations*, (2016).

[5] 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*, pp. 1135–1143, (2015).

[6] Babak Hassibi and David G Stork, 'Second order derivatives for network pruning: Optimal brain surgeon', in *Advances in Neural Information Processing Systems*, pp. 164–171, (1993).

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, 'Deep residual learning for image recognition', in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, (June 2016).

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, 'Identity mappings in deep residual networks', in *European Conference on Computer Vision*, pp. 630–645. Springer, (2016).

[9] 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)*, pp. 2234–2240, (2018).

[10] Yang He, Ping Liu, Ziwei Wang, Zhilan Hu, and Yi Yang, 'Filter pruning via geometric median for deep convolutional neural networks acceleration', in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4340–4349, (June 2019).

[11] Yihui He, Xiangyu Zhang, and Jian Sun, 'Channel pruning for accelerating very deep neural networks', *In Proceedings of International Conference on Computer Vision*, 1398–1406, (2017).

[12] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean, 'Distilling the knowledge in a neural network', in *Workshop on Advances in Neural Information Processing Systems*, (2014).

[13] Hengyuan Hu, Rui Peng, Yu-Wing Tai, and Chi-Keung Tang, 'Network trimming: A data-driven neuron pruning approach towards efficient deep architectures', *arXiv preprint arXiv:1607.03250*, (2016).

[14] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger, 'Densely connected convolutional networks', in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4700–4708, (2017).

[15] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio, 'Binarized neural networks', in *Advances in Neural Information Processing Systems*, pp. 4114–4122, (2016).

[16] Alex Krizhevsky and Geoffrey Hinton, 'Learning multiple layers of features from tiny images', (2009).

[17] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton, 'Imagenet classification with deep convolutional neural networks', in *Advances in Neural Information Processing Systems*, pp. 1097–1105, (2012).

[18] Yann LeCun, John S Denker, and Sara A Solla, 'Optimal brain damage', in *Advances in Neural Information Processing Systems*, pp. 598–605, (1990).

[19] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf, 'Pruning filters for efficient convnets', in *International Conference on Learning Representations*, (2017).

[20] Shaohui Lin, Rongrong Ji, Chenqian Yan, Baochang Zhang, Liujuan Cao, Qixiang Ye, Feiyue Huang, and David Doermann, 'Towards optimal structured cnn pruning via generative adversarial learning', in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2790–2799, (June 2019).

[21] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg, 'Ssd: Single shot multibox detector', in *European Conference on Computer Vision*, pp. 21–37, (2016).

[22] Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang, 'Learning efficient convolutional networks through network slimming', in *Proceedings of International Conference on Computer Vision*, pp. 2736–2744, (2017).

[23] Jonathan Long, Evan Shelhamer, and Trevor Darrell, 'Fully convolutional networks for semantic segmentation', in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3431–3440, (2015).

[24] Jian-Hao Luo, Jianxin Wu, and Weiyao Lin, 'Thinet: A filter level pruning method for deep neural network compression', in *Proceedings of International Conference on Computer Vision*, pp. 5058–5066, (2017).

[25] Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz, 'Pruning convolutional neural networks for resource efficient transfer learning', in *International Conference on Learning Representations*, (11 2017).

[26] J. Ross Quinlan, 'Induction of decision trees', *Machine learning*, **1**(1), 81–106, (1986).

[27] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi, 'Xnor-net: Imagenet classification using binary convolutional neural networks', in *European Conference on Computer Vision*, pp. 525–542. Springer, (2016).

[28] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al., 'Imagenet large scale visual recognition challenge', *International Journal of Computer Vision*, **115**(3), 211–252, (2015).

[29] K. Simonyan and A. Zisserman, 'Very deep convolutional networks for large-scale image recognition', in *International Conference on Learning Representations*, (2015).

[30] 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*, pp. 2074–2082, (2016).

[31] Ruichi Yu, Ang Li, Chun-Fu Chen, Jui-Hsin Lai, Vlad I Morariu, Xintong Han, Mingfei Gao, Ching-Yung Lin, and Larry S Davis, 'Nisp: Pruning networks using neuron importance score propagation', in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 9194–9203, (2018).

[32] 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*, pp. 2780–2789, (2019).