Discrimination-aware Channel Pruning for Deep Neural Networks

Zhuangwei Zhuang1*, Mingkui Tan1†, Bohan Zhuang2*, Jing Liu1*, Yong Guo1, Qingyao Wu1, Junzhou Huang3,4, Jinhui Zhu1†

1South China University of Technology, 2The University of Adelaide, 3University of Texas at Arlington, 4Tencent AI Lab
{z.zhuangwei, seliu.jing, guo.yong}@mail.scut.edu.cn, jzhuang@uta.edu
{mingkuitan, qyw, csjhzhu}@scut.edu.cn, bohan.zhuang@adelaide.edu.au

Abstract

Channel pruning is one of the predominant approaches for deep model compression. Existing pruning methods either train from scratch with sparsity constraints on channels, or minimize the reconstruction error between the pre-trained feature maps and the compressed ones. Both strategies suffer from some limitations: the former kind is computationally expensive and difficult to converge, whilst the latter kind optimizes the reconstruction error but ignores the discriminative power of channels. In this paper, we investigate a simple-yet-effective method called discrimination-aware channel pruning (DCP) to choose those channels that really contribute to discriminative power. To this end, we introduce additional discrimination-aware losses into the network to increase the discriminative power of intermediate layers and then select the most discriminative channels for each layer by considering the additional loss and the reconstruction error. Last, we propose a greedy algorithm to conduct channel selection and parameter optimization in an iterative way. Extensive experiments demonstrate the effectiveness of our method. For example, on ILSVRC-12, our pruned ResNet-50 with 30% reduction of channels outperforms the baseline model by 0.39% in top-1 accuracy.

1 Introduction

Since 2012, convolutional neural networks (CNNs) have achieved great success in many computer vision tasks, e.g., image classification [19, 38, 7], face recognition [34, 39], object detection [32, 33], image captioning [44, 47] and video analysis [35, 45]. However, deep models are often with a huge number of parameters and the model size is very large, which incurs not only huge memory requirement but also unbearable computation burden. As a result, deep learning methods are hard to be applied on hardware devices with limited storage and computation resources, such as cell phones. To address this problem, model compression is an effective approach, which aims to reduce the model redundancy without significant degeneration in performance.

Recent studies on model compression mainly contain three categories, namely, quantization [31, 53], sparse or low-rank compressions [8, 9], and channel pruning [25, 26, 50, 48]. Network quantization seeks to reduce the model size by quantizing float weights into low-bit weights (e.g., 8 bits or even 1 bit). However, the training is very difficult due to the introduction of quantization errors. Making sparse connections can reach high compression rate in theory, but it may generate irregular convolutional kernels which need sparse matrix operations for accelerating the computation. In
contrast, channel pruning reduces the model size and speeds up the inference by removing redundant channels directly, thus little additional effort is required for fast inference. On top of channel pruning, other compression methods such as quantization can be applied. In fact, pruning redundant channels often helps to improve the efficiency of quantization and achieve more compact models.

Identifying the informative (or important) channels, also known as channel selection, is a key issue in channel pruning. Existing works have exploited two strategies, namely, training-from-scratch methods which directly learn the importance of channels with sparsity regularization [1,22,25], and reconstruction-based methods [12,14,22,26]. Training-from-scratch is very difficult to train especially for very deep networks on large-scale datasets. Reconstruction-based methods seek to do channel pruning by minimizing the reconstruction error of feature maps between the pruned model and a pre-trained model [12,26]. These methods suffer from a critical limitation: an actually redundant channel would be mistakenly kept to minimize the reconstruction error of feature maps. Consequently, these methods may result in apparent drop in accuracy on more compact and deeper models such as ResNet [11] for large-scale datasets.

In this paper, we aim to overcome the drawbacks of both strategies. First, in contrast to existing methods [12,14,22,26], we assume and highlight that an informative channel, no matter where it is, should own discriminative power; otherwise it should be deleted. Based on this intuition, we propose to find the channels with true discriminative power for the network. Specifically, relying on a pre-trained model, we add multiple additional losses (i.e., discrimination-aware losses) evenly to the network. For each stage, we first do fine-tuning using one additional loss and the final loss to improve the discriminative power of intermediate layers. And then, we conduct channel pruning for each layer involved in the considered stage by considering both the additional loss and the reconstruction error of feature maps. In this way, we are able to make a balance between the discriminative power of channels and the feature map reconstruction.

Our main contributions are summarized as follows. First, we propose a discrimination-aware channel pruning (DCP) scheme for compressing deep models with the introduction of additional losses. DCP is able to find the channels with true discriminative power. DCP prunes and updates the model stage-wisely using a proper discrimination-aware loss and the final loss. As a result, it is not sensitive to the initial pre-trained model. Second, we formulate the channel selection problem as an \( \ell_{2,0} \)-norm constrained optimization problem and propose a greedy method to solve the resultant optimization problem. Extensive experiments demonstrate the superior performance of our method, especially on deep ResNet. On ILSVRC-12 [3], when pruning 30% channels of ResNet-50, DCP improves the original ResNet model by 0.39% in top-1 accuracy. Moreover, when pruning 50% channels of ResNet-50, DCP outperforms ThiNet [26], a state-of-the-art method, by 0.81% and 0.51% in top-1 and top-5 accuracy, respectively.

## 2 Related studies

**Network quantization.** In [31], Rastegari et al. propose to quantize parameters in the network into +1/−1. The proposed BWN and XNOR-Net can achieve comparable accuracy to their full-precision counterparts on large-scale datasets. In [54], high precision weights, activations and gradients in CNNs are quantized to low bit-width version, which brings great benefits for reducing resource requirement and power consumption in hardware devices. By introducing zero as the third quantized value, ternary weight networks (TWNs) [21,55] can achieve higher accuracy than binary neural networks. Explorations on quantization [55] show that quantized networks can even outperform the full precision networks when quantized to the values with more bits, e.g., 4 or 5 bits.

**Sparse or low-rank connections.** To reduce the storage requirements of neural networks, Han et al. suggest that neurons with zero input or output connections can be safely removed from the network [10]. With the help of the \( \ell_1/\ell_2 \) regularization, weights are pushed to zeros during training. Subsequently, the compression rate of AlexNet can reach \( 35 \times \) with the combination of pruning, quantization, and Huffman coding [9]. Considering the importance of parameters is changed during weight pruning, Guo et al. propose dynamic network surgery (DNS) in [8]. Training with sparsity constraints [37,46] has also been studied to reach higher compression rate.

Deep models often contain a lot of correlations among channels. To remove such redundancy, low-rank approximation approaches have been widely studied [4,5,17,36]. For example, Zhang et
al. speed up VGG for $4 \times$ with negligible performance degradation on ImageNet \cite{52}. However, low-rank approximation approaches are unable to remove those redundant channels that do not contribute to the discriminative power of the network.

**Channel pruning.** Compared with network quantization and sparse connections, channel pruning removes both channels and the related filters from the network. Therefore, it can be well supported by existing deep learning libraries with little additional effort. The key issue of channel pruning is to evaluate the importance of channels. Li et al. measure the importance of channels by calculating the sum of absolute values of weights \cite{22}. Hu et al. define average percentage of zeros (APoZ) to measure the activation of neurons \cite{14}. Neurons with higher values of APoZ are considered more redundant in the network. With a sparsity regularizer in the objective function, training-based methods \cite{1125} are proposed to learn the compact models in the training phase. With the consideration of efficiency, reconstruction-methods \cite{1226} transform the channel selection problem into the optimization of reconstruction error and solve it by a greedy algorithm or LASSO regression.

### 3 Proposed method

Let $\{x_i, y_i\}_{i=1}^N$ be the training samples, where $N$ indicates the number of samples. Given an $L$-layer CNN model $M$, let $W \in \mathbb{R}^{n \times (n \times h_f \times z_f)}$ be the model parameters w.r.t. the $l$-th convolutional layer (or block), as shown in Figure 1. Here, $h_f$ and $z_f$ denote the height and width of filters, respectively; $c$ and $n$ denote the number of input and output channels, respectively. For convenience, hereafter we omit the layer index $l$. Let $\mathbf{X} \in \mathbb{R}^{N \times c \times h_l \times z_l}$ and $\mathbf{O} \in \mathbb{R}^{N \times c' \times h_{out} \times z_{out}}$ be the input feature maps and the involved output feature maps, respectively. Here, $h_{in}$ and $z_{in}$ denote the height and width of the input feature maps, respectively; $h_{out}$ and $z_{out}$ represent the height and width of the output feature maps, respectively. Moreover, let $\mathbf{X}_{i,k,:,:}$ be the feature map of the $k$-th channel for the $i$-th sample. $\mathbf{W}_{j,k,:,:}$ denotes the parameters w.r.t. the $k$-th input channel and $j$-th output channel. The output feature map of the $j$-th channel for the $i$-th sample, denoted by $\mathbf{O}_{i,j,:,:}$, is computed by

$$\mathbf{O}_{i,j,:,:} = \sum_{k=1}^{c} \mathbf{X}_{i,k,:,:} \ast \mathbf{W}_{j,k,:,:},$$

where $\ast$ denotes the convolutional operation.

Given a pre-trained model $M$, the task of Channel Pruning is to prune those redundant channels in $W$ to save the model size and accelerate the inference speed in Eq. (1). In order to choose channels, we introduce a variant of $\ell_{2,0}$-norm $\|W\|_{2,0} = \sum_{k=1}^{c} \Omega(\sum_{j=1}^{c} \|W_{j,k,:,:}\|_F)$, where $\Omega(a) = 1$ if $a \neq 0$ and $\Omega(a) = 0$ if $a = 0$, and $\|\cdot\|_F$ represents the Frobenius norm. To induce sparsity, we can impose an $\ell_{2,0}$-norm constraint on $W$:

$$\|W\|_{2,0} = \sum_{k=1}^{c} \Omega(\sum_{j=1}^{c} \|W_{j,k,:,:}\|_F) \leq \kappa_l,$$

where $\kappa_l$ denotes the desired number of channels at the layer $l$. Or equivalently, given a predefined pruning rate $\eta \in (0, 1)$ \cite{1125}, it follows that $\kappa_l = \lceil \eta c \rceil$. 

![Figure 1: Illustration of discrimination-aware channel pruning. Here, $\mathcal{L}_p^\prime$ denotes the discrimination-aware loss (e.g., cross-entropy loss) in the $L_p$-th layer, $\mathcal{L}_M$ denotes the reconstruction loss, and $\mathcal{L}_f$ denotes the final loss. For the $p$-th stage, we first fine-tune the pruned model by $\mathcal{L}_S^p$ and $\mathcal{L}_f$, then conduct the channel selection for each layer in $\{L_{p-1} + 1, \ldots, L_p\}$ with $\mathcal{L}_S^p$ and $\mathcal{L}_M$.](image-url)
3.1 Motivations

Given a pre-trained model $M$, existing methods [12][26] conduct channel pruning by minimizing the reconstruction error of feature maps between the pre-trained model $M$ and the pruned one. Formally, the reconstruction error can be measured by the mean squared error (MSE) between feature maps of the baseline network and the pruned one as follows:

$$L_M(W) = \frac{1}{2Q} \sum_{i=1}^{N} \sum_{j=1}^{n} ||O^b_{i,j} - O_{i,j}||^2_F,$$

where $Q = N \cdot n \cdot h_{out} \cdot z_{out}$ and $O^b_{i,j}$ denotes the feature maps of the baseline network. Reconstructing feature maps can preserve most information in the learned model, but it has two limitations. First, the pruning performance is highly affected by the quality of the pre-trained model $M$. If the baseline model is not well trained, the pruning performance can be very limited. Second, to achieve the minimal reconstruction error, some channels in intermediate layers may be mistakenly kept, even though they are actually not relevant to the discriminative power of the network. This issue will be even severer when the network becomes deeper.

In this paper, we seek to do channel pruning by keeping those channels that really contribute to the discriminative power of the network. In practice, however, it is very hard to measure the discriminative power of channels due to the complex operations (such as ReLU activation and Batch Normalization) in CNNs. One may consider one channel as an important one if the final loss $L$ increases without it. However, it is not practical when the network is very deep. In fact, for deep models, its shallow layers often have little discriminative power due to the long path of propagation.

To increase the discriminative power of intermediate layers, one can introduce additional losses to the intermediate layers of the deep networks [40][20][6]. In this paper, we insert $P$ discrimination-aware losses $\{L_S^{p} \}_{p=1}^P$ evenly into the network, as shown in Figure 1. Let $\{L_1, ..., L_P, L_{P+1}\}$ be the layers at which we put the losses, with $L_{P+1} = L$ being the final layer. For the $p$-th loss $L_S^p$, we consider doing channel pruning for layers $l \in \{L_{p-1} + 1, ..., L_p\}$, where $L_{p-1} = 0$ if $p = 1$. It is worth mentioning that, we can add one loss to each layer of the network, where we have $L_l = l$. However, this can be very computationally expensive yet not necessary.

3.2 Construction of discrimination-aware loss

The construction of discrimination-aware loss $L_S^p$ is very important in our method. As shown in Figure 1, each loss uses the output of layer $L_p$ as the input feature maps. To make the computation of the loss feasible, we impose an average pooling operation over the feature maps. Moreover, to accelerate the convergence, we shall apply batch normalization [16] and ReLU [27] before doing the average pooling. In this way, the input feature maps for the loss at layer $L_p$, denoted by $F^p(W)$, can be computed by

$$F^p(W) = \text{AvgPooling}(\text{ReLU} (\text{BN}(O^p))),$$

where $O^p$ represents the output feature maps of layer $L_p$. Let $F^{(p-i)}$ be the feature maps w.r.t. the $i$-th example. The discrimination-aware loss w.r.t. the $p$-th loss is formulated as

$$L_S^p(W) = -\frac{1}{N} \left[ \sum_{i=1}^{N} \sum_{t=1}^{m} I\{y^{(i)} = t\} \log \frac{e^{\theta_{y^{(i)}}^{(p-i)}}}{\sum_{k=1}^{m} e^{\theta_{k}^{(p-i)}}} \right],$$

where $I\{\cdot\}$ is the indicator function, $\theta \in \mathbb{R}^{n_p \times m}$ denotes the classifier weights of the fully connected layer, $n_p$ denotes the number of input channels of the fully connected layer and $m$ is the number of classes. Note that we can use other losses such as angular softmax loss [24] as the additional loss.

In practice, since a pre-trained model contains very rich information about the learning task, similar to [26], we also hope to reconstruct the feature maps in the pre-trained model. By considering both cross-entropy loss and reconstruction error, we have a joint loss function as follows:

$$L(W) = L_M(W) + \lambda L_S^p(W),$$

where $\lambda$ balances the two terms.

**Proposition 1 (Convexity of the loss function)** Let $W$ be the model parameters of a considered layer. Given the mean square loss and the cross-entropy loss defined in Eqs. (3) and (5), then the joint loss function $L(W)$ is convex w.r.t. $W$.

The proof can be found in Section 4.1 of the supplementary material.
Last, the optimization problem for discrimination-aware channel pruning can be formulated as
\[
\min_{W} \ L(W), \quad \text{s.t. } \|W\|_{2,0} \leq \kappa _{i},
\]  
(7)
where \( \kappa _{i} < c \) is the number of channels to be selected. In our method, the sparsity of \( W \) can be either determined by a pre-defined pruning rate (See Section \[3\]) or automatically adjusted by the stopping conditions in Section \[5.5\]. We explore both effects in Section \[4\].

### 3.3 Discrimination-aware channel pruning

By introducing \( P \) losses \( \{L_{S}^{p}\}_{p=1}^{P} \) to intermediate layers, the proposed discrimination-aware channel pruning (DCP) method is shown in Algorithm \[1\]. Starting from a pre-trained model, DCP updates the model \( M \) and performs channel pruning with \((P + 1)\) stages. Algorithm \[1\] is called discrimination-aware in the sense that an additional loss and the final loss are considered to fine-tune the model. Moreover, the additional loss will be used to select channels, as discussed below. In contrast to GoogLeNet \[40\] and DSN \[20\], in Algorithm \[1\] we do not use all the losses at the same time. In fact, at each stage we will consider two losses only, i.e., \( L_{S}^{p} \) and the final loss \( L_{f} \).

#### Algorithm 1: Discrimination-aware channel pruning (DCP)

**Input:** Pre-trained model \( M \), training data \( \{x_{i}, y_{i}\}_{i=1}^{N} \), and parameters \( \{\kappa _{i}\}_{i=1}^{c} \).

**Algorithm:**

1. **for** \( p \in \{1, \ldots, P + 1\} \) **do**
   1. Construct loss \( L_{S}^{p} \) to layer \( L_{p} \) as in Figure \[1\].
   2. Learn \( \theta \) and Fine-tune \( M \) with \( L_{S}^{p} \) and \( L_{f} \).
   **for** \( l \in \{L_{p-1} + 1, \ldots, L_{p}\} \) **do**
   3. Do Channel Selection for layer \( l \) using Algorithm \[2\].
   **end for**
   **end for**

At each stage of Algorithm \[1\] for example, in the \( p \)-th stage, we first construct the additional loss \( L_{S}^{p} \) and put them at layer \( L_{p} \) (See Figure \[1\]). After that, we learn the model parameters \( \theta \) w.r.t. \( L_{S}^{p} \) and fine-tune the model \( M \) at the same time with both the additional loss \( L_{S}^{p} \) and the final loss \( L_{f} \). In the fine-tuning, all the parameters in \( M \) will be updated. Here, with the fine-tuning, the parameters regarding the additional loss can be well learned. Besides, fine-tuning is essential to compensate the accuracy loss from the previous pruning to suppress the accumulative error. After fine-tuning with \( L_{S}^{p} \) and \( L_{f} \), the discriminative power of layers \( l \in \{L_{p-1} + 1, \ldots, L_{p}\} \) can be significantly improved. Then, we can perform channel selection for the layers in \( \{L_{p-1} + 1, \ldots, L_{p}\} \).

#### Algorithm 2: Greedy algorithm for channel selection

**Input:** Training data, model \( M \), parameters \( \kappa _{i} \), and \( \epsilon \).

**Output:** Selected channel subset \( \mathcal{A} \) and model parameters \( W_{\mathcal{A}} \).

**Algorithm:**

1. Initialize \( \mathcal{A} \leftarrow \emptyset \), and \( t \leftarrow 0 \).
2. **while** (stopping condition is not achieved) **do**
   1. Compute gradients of \( L \) w.r.t. \( W \), \( G = \partial L / \partial W \).
   2. Find the channel \( k = \arg \max _{j \notin \mathcal{A}} \{\|G_{j}\|_{F}\} \).
   3. Let \( \mathcal{A} \leftarrow \mathcal{A} \cup \{k\} \).
   4. Solve Problem \( (8) \) to update \( W_{\mathcal{A}} \).
   5. Let \( t \leftarrow t + 1 \).
3. **end while**

### 3.4 Greedy algorithm for channel selection

Due to the \( \ell _{2,0} \)-norm constraint, directly optimizing Problem \( (7) \) is very difficult. To address this issue, following general greedy methods in \[23,24,51,43,42\], we propose a greedy algorithm to solve Problem \( (7) \). To be specific, we first remove all the channels and then select those channels that really contribute to the discriminative power of the deep networks. Let \( \mathcal{A} \subset \{1, \ldots, c\} \) be the index set of the selected channels, where \( \mathcal{A} \) is empty at the beginning. As shown in Algorithm \[2\] the channel selection method can be implemented in two steps. First, we select the most important channels of input feature maps. At each iteration, we compute the gradients \( G_{j} = \partial L / \partial W_{j} \), where \( W_{j} \) denotes the parameters for the \( j \)-th input channel. We choose the channel \( k = \arg \max _{j \notin \mathcal{A}} \{\|G_{j}\|_{F}\} \) as an active channel and put \( k \) into \( \mathcal{A} \). Second, once \( \mathcal{A} \) is determined, we optimize \( W \) w.r.t. the selected channels by minimizing the following problem:
\[
\min _{W} \ L(W), \quad \text{s.t. } W_{\mathcal{A}^{c}} = 0,
\]  
(8)

where \( W_{\mathcal{A}^{c}} \) denotes the submatrix indexed by \( \mathcal{A}^{c} \) which is the complementary set of \( \mathcal{A} \). Here, we apply stochastic gradient descent (SGD) to address the problem in Eq. \( (8) \), and update \( W_{\mathcal{A}} \) by
\[
W_{\mathcal{A}} \leftarrow W_{\mathcal{A}} - \gamma \frac{\partial L}{\partial W_{\mathcal{A}}},
\]  
(9)

\[\text{The details of fine-tuning algorithm is put in Section S2 in the supplementary material.}\]
where $W_A$ denotes the submatrix indexed by $A$, and $\gamma$ denotes the learning rate.

Note that when optimizing Problem (8), $W_A$ is warm-started from the fine-tuned model $M$. As a result, the optimization can be completed very quickly. Moreover, since we only consider the model parameter $W$ for one layer, we do not need to consider all data to do the optimization. To make a trade-off between the efficiency and performance, we sample a subset of images randomly from the training data for optimization. Last, since we use SGD to update $W_A$, the learning rate $\gamma$ should be carefully adjusted to achieve an accurate solution. Then, the following stopping conditions can be applied, which will help to determine the number of channels to be selected.

### 3.5 Stopping conditions

Given a predefined parameter $\kappa_{l}$ in problem (7). Algorithm 2 will be stopped if $\|W\|_{2,0} > \kappa_{l}$. However, in practice, the parameter $\kappa_{l}$ is hard to be determined. Since $L$ is convex, $L(W^t)$ will monotonically decrease with iteration index $t$ in Algorithm 2. We can therefore adopt the following stopping condition:

$$|L(W^{t-1}) - L(W^t)|/L(W^0) \leq \epsilon,$$

where $\epsilon$ is a tolerance value. If the above condition is achieved, the algorithm is stopped, and the number of selected channels will be automatically determined, i.e., $\|W^t\|_{2,0}$. An empirical study over the tolerance value $\epsilon$ is put in Section 5.3.

### 4 Experiments

In this section, we empirically evaluate the performance of DCP. Several state-of-the-art methods are adopted as the baselines, including ThiNet [26], Channel pruning (CP) [12] and Slimming [25]. Besides, to investigate the effectiveness of the proposed method, we include the following methods for study: DCP: DCP with a pre-defined pruning rate $\eta$. DCP-Adapt: We prune each layer with the stopping conditions in Section 3.5. WM: We shrink the width of a network by a fixed ratio and train it from scratch, which is known as width-multiplier [13]. WM+: Based on WM, we evenly insert additional losses to the network and train it from scratch. Random DCP: Relying on DCP, we randomly choose channels instead of using gradient-based strategy in Algorithm 2.

**Datasets.** We evaluate the performance of various methods on three datasets, including CIFAR-10 [18], ILSVRC-12 [3], and LFW [15]. CIFAR-10 consists of 50k training samples and 10k testing images with 10 classes. ILSVRC-12 contains 1.28 million training samples and 50k testing images for 1000 classes. LFW [15] contains 13,233 face images from 5,749 identities.

### 4.1 Implementation details

We implement the proposed method on PyTorch [30]. Based on the pre-trained model, we apply our method to select the informative channels. In practice, we decide the number of additional losses according to the depth of the network (See Section S4 in the supplementary material). Specifically, we insert 3 losses to ResNet-50 and ResNet-56, and 2 additional losses to VGGNet and ResNet-18. We fine-tune the whole network with selected channels only. We use SGD with nesterov [28] for the optimization. The momentum and weight decay are set to 0.9 and 0.0001, respectively. We set $\lambda$ to 1.0 in our experiments by default. On CIFAR-10, we fine-tune 400 epochs using a mini-batch size of 128. The learning rate is initialized to 0.1 and divided by 10 at epoch 160 and 240. On ILSVRC-12, we fine-tune the network for 60 epochs with a mini-batch size of 256. The learning rate is started at 0.01 and divided by 10 at epoch 36, 48 and 54, respectively.

### 4.2 Comparisons on CIFAR-10

We first prune ResNet-56 and VGGNet on CIFAR-10. The comparisons with several state-of-the-art methods are reported in Table 1. From the results, our method achieves the best performance under the same acceleration rate compared with the previous state-of-the-art. Moreover, with DCP-Adapt, our pruned VGGNet outperforms the pre-trained model by 0.58% in testing error, and obtains 15.58× reduction in model size. Compared with random DCP, our proposed DCP reduces the performance

---

\[\text{We study the effect of the number of samples in Section 5.5 in the supplementary material.}\]
Table 1: Comparisons on CIFAR-10. "-" denotes that the results are not reported.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>VGGNet (Baseline 6.01%)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#Param. ↓</td>
<td>1.92×</td>
<td>1.92×</td>
<td>8.71×</td>
<td>1.92×</td>
<td>1.92×</td>
<td>1.92×</td>
<td>1.92×</td>
<td>15.58×</td>
</tr>
<tr>
<td>#FLOPs ↓</td>
<td>2.00×</td>
<td>2.00×</td>
<td>4.04×</td>
<td>2.00×</td>
<td>2.00×</td>
<td>2.00×</td>
<td>2.00×</td>
<td>2.86×</td>
</tr>
<tr>
<td>Err. gap (%)</td>
<td>+0.14</td>
<td>+0.32</td>
<td>+0.19</td>
<td>+0.38</td>
<td>+0.10</td>
<td>+0.14</td>
<td>-0.17</td>
<td>-0.58</td>
</tr>
<tr>
<td>ResNet-56 (Baseline 6.20%)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#Param. ↓</td>
<td>1.99×</td>
<td>1.99×</td>
<td>1.99×</td>
<td>1.99×</td>
<td>1.99×</td>
<td>1.99×</td>
<td>1.99×</td>
<td>1.89×</td>
</tr>
<tr>
<td>#FLOPs ↓</td>
<td>2.86×</td>
<td>2.86×</td>
<td>2.86×</td>
<td>2.86×</td>
<td>2.86×</td>
<td>2.86×</td>
<td>2.86×</td>
<td>2.86×</td>
</tr>
<tr>
<td>Err. gap (%)</td>
<td>+0.82</td>
<td>+1.0</td>
<td>+0.56</td>
<td>+0.45</td>
<td>+0.63</td>
<td>+0.31</td>
<td>-0.01</td>
<td></td>
</tr>
</tbody>
</table>

We further conduct experiments on LFW [15], which is a standard benchmark dataset for face recognition. We use CASIA-WebFace [29] (which consists of 494,414 face images from 10,575 individuals) for training. With the same settings in [24], we first train SphereNet-4 (which contains 4 convolutional layers) from scratch. And then, we adopt our method to compress the pre-trained SphereNet model. Since the fully connected layer occupies 87.65% parameters of the model, we also prune the fully connected layer to reduce the model size.

4.3 Comparisons on ILSVRC-12

To verify the effectiveness of the proposed method on large-scale datasets, we further apply our method on ResNet-50 to achieve 2× acceleration on ILSVRC-12. We report the single view evaluation in Table 3. Our method outperforms ThiNet [26] by 0.81% and 0.51% in top-1 and top-5 error, respectively. Compared with channel pruning [12], our pruned model achieves 0.79% improvement in top-5 error. Compared with WM+, which leads to 2.41% increase in top-1 error, our method only results in 1.06% degradation in top-1 error.

Table 3: Comparisons on ILSVRC-12. The top-1 and top-5 error (%) of the pre-trained model are 23.99 and 7.07, respectively. "-" denotes that the results are not reported.

<table>
<thead>
<tr>
<th>Model</th>
<th>ThiNet</th>
<th>CP [12]</th>
<th>WM</th>
<th>WM+</th>
<th>DCP</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResNet-50</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#Param. ↓</td>
<td>2.06×</td>
<td>-</td>
<td>2.06×</td>
<td>2.06×</td>
<td>2.06×</td>
</tr>
<tr>
<td>#FLOPs ↓</td>
<td>2.25×</td>
<td>2×</td>
<td>2.25×</td>
<td>2.25×</td>
<td>2.25×</td>
</tr>
<tr>
<td>Top-1 gap (%)</td>
<td>+1.87</td>
<td>-</td>
<td>+2.81</td>
<td>+2.41</td>
<td>+1.06</td>
</tr>
<tr>
<td>Top-5 gap (%)</td>
<td>+1.12</td>
<td>+1.40</td>
<td>+1.62</td>
<td>+1.28</td>
<td>+0.61</td>
</tr>
</tbody>
</table>

4.4 Experiments on LFW

We further conduct experiments on LFW [15], which is a standard benchmark dataset for face recognition. We use CASIA-WebFace [29] (which consists of 494,414 face images from 10,575 individuals) for training. With the same settings in [24], we first train SphereNet-4 (which contains 4 convolutional layers) from scratch. And then, we adopt our method to compress the pre-trained SphereNet model. Since the fully connected layer occupies 87.65% parameters of the model, we also prune the fully connected layer to reduce the model size.
We report the results in Table 4. With the pruning rate of 50%, our method speeds up SphereNet-4 for 3.66\times with 0.1\% improvement in ten-fold validation accuracy. Compared with huge networks, e.g., FaceNet [34], DeepFace [41], and VGG [29], our pruned model achieves comparable performance but has only 45.15M FLOPs and 5.89M parameters, which is sufficient to be deployed on embedded systems. Furthermore, pruning 65% channels in SphereNet-4 results in a more compact model, which requires only 24.16M FLOPs with the accuracy of 98.02\% on LFW.

5 Ablation studies

5.1 Performance with different pruning rates

To study the effect of using different pruning rates $\eta$, we prune 30\%, 50\%, and 70\% channels of ResNet-18 and ResNet-50, and evaluate the pruned models on ILSVRC-12. Experimental results are shown in Table 5. Here, we only report the performance under different pruning rates, while the detailed model complexity comparisons are provided in Section S8 in the supplementary material.

From Table 5 in general, performance of the pruned models goes worse with the increase of pruning rate. However, our pruned ResNet-50 with pruning rate of 30\% outperforms the pre-trained model, with 0.39\% and 0.14\% reduction in top-1 and top-5 error, respectively. Besides, the performance degradation of ResNet-50 is smaller than that of ResNet-18 with the same pruning rate. For example, when pruning 50\% of the channels, while it only leads to 1.06\% increase in top-1 error for ResNet-50, it results in 2.29\% increase of top-1 error for ResNet-18. One possible reason is that, compared to ResNet-18, ResNet-50 is more redundant with more parameters, thus it is easier to be pruned.

We prune 30\% channels of ResNet-56 on CIFAR-10 with different $\lambda$. We report the training error and testing error without fine-tuning in Table 6. From the table, the performance of the pruned model improves with increasing $\lambda$. Here, a larger $\lambda$ implies that we put more emphasis on the additional loss (See Equation (6)). This demonstrates the effectiveness of discrimination-aware strategy for channel selection. It is worth mentioning that both the reconstruction error and the cross-entropy loss contribute to better performance of the pruned model, which strongly supports the motivation to select the important channels by $L_S$ and $L_M$. After all, as the network achieves the best result when $\lambda$ is set to 1.0, we use this value to initialize $\lambda$ in our experiments.
5.3 Effect of the stopping condition

To explore the effect of stopping condition discussed in Section 3.5, we test different tolerance value \( \epsilon \) in the condition. Here, we prune VGGNet on CIFAR-10 with \( \epsilon \in \{0.1, 0.01, 0.001\} \). Experimental results are shown in Table 7. In general, a smaller \( \epsilon \) will lead to more rigorous stopping condition and hence more channels will be selected. As a result, the performance of the pruned model is improved with the decrease of \( \epsilon \). This experiment demonstrates the usefulness and effectiveness of the stopping condition for automatically determining the pruning rate.

Table 7: Effect of \( \epsilon \) for channel selection. We prune VGGNet and report the testing error on CIFAR-10. The testing error of baseline VGGNet is 6.01%.

<table>
<thead>
<tr>
<th>Loss ( \mathcal{L} )</th>
<th>( \epsilon )</th>
<th>Testing err. (%)</th>
<th>#Param. ( \downarrow )</th>
<th>#FLOPs ( \downarrow )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \mathcal{L} )</td>
<td>0.1</td>
<td>12.68</td>
<td>152.25×</td>
<td>27.39×</td>
</tr>
<tr>
<td>( \mathcal{L} )</td>
<td>0.01</td>
<td>6.63</td>
<td>31.28×</td>
<td>5.35×</td>
</tr>
<tr>
<td>( \mathcal{L} )</td>
<td>0.001</td>
<td>5.43</td>
<td>15.58×</td>
<td>2.86×</td>
</tr>
</tbody>
</table>

5.4 Visualization of feature maps

We visualize the feature maps w.r.t. the pruned/selected channels of the first block (i.e., res-2a) in ResNet-18 in Figure 2. From the results, we observe that feature maps of the pruned channels (See Figure 2(b)) are less informative compared to those of the selected ones (See Figure 2(c)). It proves that the proposed DCP selects the channels with strong discriminative power for the network. More visualization results can be found in Section S10 in the supplementary material.

Figure 2: Visualization of the feature maps of the pruned/selected channels of res-2a in ResNet-18.

6 Conclusion

In this paper, we have proposed a discrimination-aware channel pruning method for the compression of deep neural networks. We formulate the channel pruning/selection problem as a sparsity-induced optimization problem by considering both reconstruction error and channel discrimination power. Moreover, we propose a greedy algorithm to solve the optimization problem. Experimental results on benchmark datasets show that the proposed method outperforms several state-of-the-art methods by a large margin with the same pruning rate. Our DCP method provides an effective way to obtain more compact networks. For those compact network designs such as MobileNet v1&v2, DCP can still improve their performance by removing redundant channels. In particular for MobileNet v2, DCP improves it by reducing 30% of channels on CIFAR-10. In the future, we will incorporate the computational cost per layer into the optimization, and combine our method with other model compression strategies (such as quantization) to further reduce the model size and inference cost.

Acknowledgements

This work was supported by National Natural Science Foundation of China (NSFC) (61876208, 61502177 and 61602185), Recruitment Program for Young Professionals, Guangdong Provincial Scientific and Technological funds (2017B090901008, 2017A010101011, 2017B090910005), Fundamental Research Funds for the Central Universities D2172480, Pearl River S&T Nova Program of Guangzhou 201806010081, CCF-Tencent Open Research Fund RAGR20170105, and Program for Guangdong Introducing Innovative and Enterpreneurial Teams 2017ZT07X183.
References


