Discrimination-aware Network Pruning for Deep Model Compression

Jing Liu*, Bohan Zhuang*, Zhuangwei Zhuang*, Yong Guo, Junzhou Huang, Jinhui Zhu, Mingkui Tan*†

Abstract—We study network pruning which aims to remove redundant channels/kernels and hence speed up the inference of deep networks. Existing pruning methods either train from scratch with sparsity constraints or minimize the reconstruction error between the feature maps of the pre-trained models and the compressed ones. Both strategies suffer from some limitations: the former kind is computationally expensive and difficult to converge, while the latter kind optimizes the reconstruction error but ignores the discriminative power of channels. In this paper, we propose a simple-yet-effective method called discrimination-aware channel pruning (DCP) to choose the channels that actually contribute to the discriminative power. To this end, we first introduce additional discrimination-aware losses into the network to increase the discriminative power of the intermediate layers. Next, we select the most discriminative channels for each layer by considering the discrimination-aware loss and the reconstruction error, simultaneously. We then formulate channel pruning as a sparsity-inducing optimization problem with a convex objective and propose a greedy algorithm to solve the resultant problem. Note that a channel (3D tensor) often consists of a set of kernels (each with a 2D matrix). Besides the redundancy in channels, some kernels in a channel may also be redundant and fail to contribute to the discriminative power of the network, resulting in kernel level redundancy. To solve this issue, we propose a discrimination-aware kernel pruning (DKP) method to further compress deep networks by removing redundant kernels. To avoid manually determining the pruning rate for each layer, we propose two adaptive stopping conditions to automatically determine the number of selected channels/kernels. The proposed adaptive stopping conditions tend to yield more efficient models with better performance in practice. Extensive experiments on both image classification and face recognition demonstrate the effectiveness of our methods. For example, on ILSVRC-12, the resultant ResNet-50 model with 30% reduction of channels even outperforms the baseline model by 0.36% in terms of Top-1 accuracy. We also deploy the pruned models on a smartphone (equipped with a Qualcomm Snapdragon 845 processor). The pruned MobileNetV1 and MobileNetV2 achieve 1.93× and 1.42× inference acceleration on the mobile device, respectively, with negligible performance degradation. The source code and the pre-trained models are available at https://github.com/SCUT-AILab/DCP.

Index Terms—Channel Pruning, Kernel Pruning, Network Compression, Deep Neural Networks.

1 Introduction

Since 2012, deep neural networks (DNNs) have achieved great success in many computer vision tasks, *e.g.*, image classification [21], [36], [69], face recognition [9], [63], [70], object detection [44], [60], [61], image generation [5], [13], [15] and video analysis [65], [78], [85]. However, the large model size and high computational costs remain great obstacles for many applications, especially on some constrained devices with limited memory and computational resources. 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: quantization [19], [59], [87], sparse or low-rank compression [17], [19], [86], and network pruning [46], [51], [81], [83]. Network quantization seeks to represent weights and activations with low bitwidth fixed-point integers, thus convolution operations can be implemented by efficient XNOR-popcount bitwise operations for substantial speedup. However, the training can be very difficult since the non-differentiable quantizer that transforms the continuous weights/activations into the discrete ones would inevitably bring errors and hamper the model performance [59]. Sparse connections methods can obtain a high compression rate in theory, but they may generate irregular convolutional kernels that need carefully designed sparse matrix operations. Low-rank compression methods seek to approximate the original filters of a pre-trained model with low-rank filters. Nevertheless, they are often inefficient for the convolutions with small kernels sizes, *e.g.*, 1×1 [72]. In contrast, network pruning reduces the model size and speeds up the inference by removing the redundant modules (channels [25], [46], [92] or kernels [2], [53]). In particular, channel pruning can be well supported by existing deep learning libraries with little additional effort compared with network quantization and sparse or low-rank connections. More critically, most compression methods, such as quantization, can be easily applied on top of network pruning. For example, pruning redundant channels/kernels is able to further reduce the model size and accelerate the inference speed of the quantized models by reducing the number of parameters [19].

In network pruning, how to identify the informative (or important) channels/kernels (also known as channel/kernel selection) is an important problem. Existing methods can be divided into two categories, namely, training-from-scratch methods [1], [46],

- *Jing Liu is with the School of Software Engineering, South China University of Technology and also with the Key Laboratory of Big Data and Intelligent Robot, South China University of Technology, Ministry of Education. E-mail: seliujing@mail.scut.edu.cn*
- *Zhuangwei Zhuang, Yong Guo, Jinhui Zhu, and Mingkui Tan are with the School of Software Engineering, South China University of Technology. Mingkui Tan is also with the Pazhou Laboratory, Guangzhou, China. E-mail: {seliujing, z.zhuangwei, guo.yong}@mail.scut.edu.cn, {csjhzhu, mingkuitan}@scut.edu.cn.*
- *Bohan Zhuang is with the Faculty of Information Technology, Monash University, Australia. E-mail: bohan.zhuang@monash.edu.*
- *Junzhou Huang is with Tencent AI Lab, Shenzhen, Guangdong, China. E-mail: joehhuang@tencent.com.*
- ** Authors contributed equally. † Corresponding author.*

[79] and reconstruction-based methods [25], [27], [39], [51]. Training-from-scratch methods directly learn the importance of channels/kernels with sparsity regularization, but it is very difficult to train very deep networks on large-scale datasets [1], [46]. The reconstruction-based methods seek to perform network pruning by minimizing the reconstruction error of feature maps between the pruned model and the pre-trained one [25], [51]. However, the performance is highly affected by the quality of the pre-trained model. If the pre-trained model is not well trained, the pruning performance can be limited. More importantly, these methods suffer from a critical limitation: the redundant channels/kernels may be mistakenly kept to minimize the reconstruction error of feature maps. Consequently, these methods may incur severe performance degradation on more compact and deeper models, such as MobileNet [26], [62] for large-scale datasets.

In this paper, we aim to overcome the drawbacks of both strategies. In contrast to existing methods [25], [27], [39], [51], we assume and highlight that an informative channel/kernel, no matter where it is, should have sufficient discriminative power; otherwise, it should be removed. Based on this intuition, we propose a discrimination-aware channel pruning (DCP) method to find the channels that actually contribute to the discriminative power of the network. In DCP, relying on a pre-trained model, we first introduce multiple additional discrimination-aware losses into the network to increase the discriminative power of the intermediate layers. Then, we perform channel selection to find the most discriminative channels for each layer by considering both the discrimination-aware loss and the reconstruction error of feature maps. In this way, we are able to make a balance between the discriminative power of the channels and feature maps reconstruction. Note that a channel (3D tensor) consists of a set of kernels (each with a 2D matrix). In practice, some kernels in the selected channels may be redundant and fail to contribute to the discriminative power of the network, which leads to kernel level redundancy. To solve this issue, we propose a discrimination-aware kernel pruning (DKP) method to find the kernels with discriminative power.

Our main contributions are summarized as follows.

- We propose a discrimination-aware channel/kernel pruning (DCP/DKP) scheme to compress deep models with the introduction of additional discrimination-aware losses. The proposed methods first fine-tune the model with the additional losses and the final objective. Then, we conduct channel/kernel selection by simultaneously considering the additional loss and the reconstruction error of feature maps. In this way, the proposed method is able to find the channels/kernels that actually contribute to the discriminative power of the network.
- We formulate the channel/kernel selection problem as a constrained optimization problem and propose a greedy method by solving the resultant convex optimization problem to select informative channels/kernels.
- We propose a new adaptive stopping condition to prevent DCP/DKP from selecting too many channels/kernels when determining the number of selected channels/kernels. Specifically, the stopping condition incorporates an additional constraint which enforces the number of selected channels to be no more than a predefined value for each layer.
- Extensive experiments demonstrate the superior performance of our methods on a variety of architectures. For example, on ILSVRC-12 [8], when pruning 30% channels from ResNet-50, DCP improves the original model by 0.36% in terms of Top-1 accuracy. To demonstrate the effectiveness of our methods, we also deploy the pruned models on a smartphone (equipped with a Qualcomm Snapdragon 845 processor) and show significant acceleration on the mobile CPU.

This paper extends our preliminary version [92] from several aspects. 1) We propose two training techniques to reduce the computational overhead of DCP while still maintaining comparable or even better performance. 2) We apply the improved DCP to more compact models (i.e., MobileNetV1 and MobileNetV2) and achieve promising performance on ILSVRC-12. We further deploy the pruned models on a smartphone with a Qualcomm Snapdragon 845 processor to investigate the inference acceleration. 3) We apply the improved DCP to compress the latest face recognition models. 4) We extend DCP for kernel pruning to further compress models at kernel level. 5) We propose a new adaptive stopping condition for the optimization. 6) We provide more ablative studies to investigate the effectiveness of our methods.

2 Related Work

Network quantization. Quantization-based methods represent the network weights and/or activations with very low precision, which yields highly compact DNNs compared to their floating-point counterparts. The extreme case is the binary neural networks (BNNs) where both weights and activations are constrained to $\{+1, -1\}$ [4], [29], [59]. In this way, one can replace the matrix multiplication operations with the light-weighted bitwise XNOR-popcount operations. As a result, the 1-bit convolutional layer can achieve up to $32\times$ memory saving and $58\times$ speedup on CPUs [59], [91]. However, BNNs still suffer from significant accuracy decreases. To reduce this accuracy gap, fixed-point methods have been proposed to represent weights and activations with higher bitwidth. Uniform fixed-point approaches [88], [90] designed quantizers with a constant quantization step. To improve the precision of the discrete uniform quantizer, [7], [33] explicitly parameterized and optimized the quantization intervals.

Sparse or low-rank connections. To reduce the storage requirements of neural networks, Han *et al.* [20] suggested that neurons with zero input or output connections can be safely removed from the network. With the help of the ℓ_1/ℓ_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 [19]. Considering the importance of parameters that are changed during weight pruning, Guo *et al.* [17] proposed dynamic network surgery (DNS). Training with sparsity constraints [68], [79] has also been studied to reach a higher compression rate. Deep models often contain many correlations among channels. To remove such redundancy, low-rank approximation approaches have been widely studied [11], [12], [31], [67]. For example, Zhang *et al.* [86] sped up VGGNet for $4\times$ with negligible performance degradation on ILSVRC-12. However, low-rank approximation approaches are not efficient for the convolutions with small kernels size, *e.g.*, 1×1 kernel [72].

Network pruning. Network pruning aims at removing redundant modules, *e.g.*, channels or kernels, to accelerate the run-time inference. The resultant pruned models would have fewer parameters and lower computational overhead. In order to measure the importance of the network module, different metrics [24], [27], [38], [39], [52], [58], [82], [83] were proposed. With a sparsity regularizer in the objective function, training-based methods [1], [40], [42], [46], [79], [89] were proposed to learn the compact models in the training phase. Considering efficiency, reconstruction-methods [25],

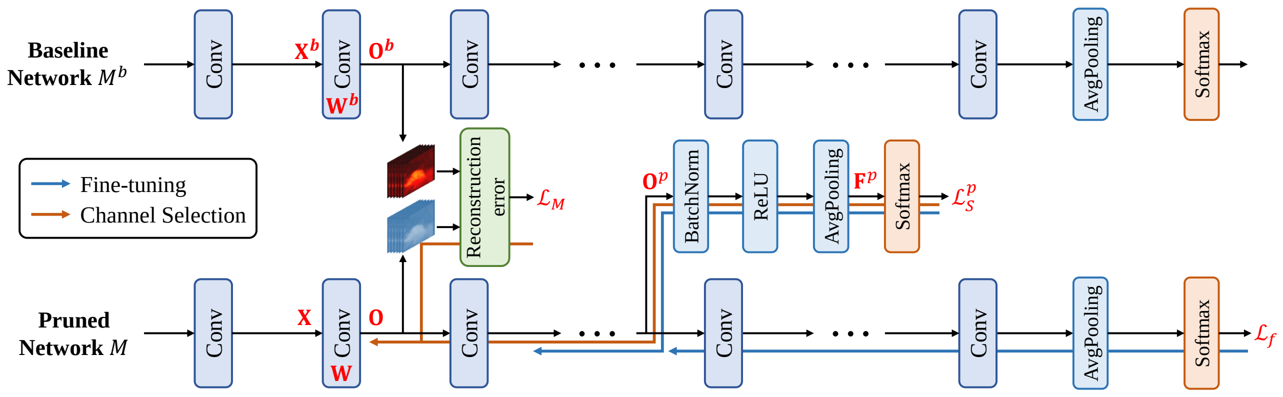


Fig. 1. Illustration of discrimination-aware channel pruning. Here, $\mathcal{L}_S^p$ denotes the discrimination-aware loss (*e.g.*, cross-entropy loss or additive angular margin loss) in the L_p-th layer, $\mathcal{L}_M$ denotes the reconstruction loss, and $\mathcal{L}_f$ denotes the final loss. DCP first updates the model M and learns the parameters $\{\boldsymbol{\theta}^p\}_{p=1}^P$ with $\{\mathcal{L}_S^p\}_{p=1}^P$ and $\mathcal{L}_f$. Then, DCP performs channel pruning with $P+1$ stages. At each stage, for example, in the p-th stage, DCP conducts the channel selection for each layer in $\{L_{p-1}+1,\ldots,L_p\}$ with the corresponding $\mathcal{L}_S^p$ and $\mathcal{L}_M$.

[27], [32], [39], [51] transformed the channel selection problem into the optimization of the reconstruction error. Recently, several methods [22], [41] have been proposed to prune the redundant filters in a dynamic such that the pruned filters can be recovered during training. Apart from these methods, the pruning rate for each layer can also be automatically determined by reinforcement learning [23], [73], greedy method [80] or evolutionary search [47]. Compared with the proposed methods, most existing methods use heuristic metrics to identify informative modules. Specifically, both our proposed methods and reconstruction-based methods use the reconstruction error during channel selection. However, unlike these methods, our proposed DCP/DKP introduce additional losses to select those channels/kernels that actually contribute to the discriminative power of deep networks. Moreover, compared with those methods that use reinforcement learning [23] or evolutionary algorithms [80], DCP/DKP use an adaptive stopping condition to automatically determine the sparsity for each layer.

3 Preliminary

Let $\{\mathbf{x}_i, y_i\}_{i=1}^N$ be the training samples, where N indicates the number of samples. Given an L-layer deep network M, let $\mathbf{W} \in \mathbb{R}^{n\times c\times h_f\times z_f}$ represents the model parameters w.r.t. the l-th convolutional layer (or block). Here, h_f and z_f denote the height and width of the filters, respectively; c and n denote the number of **input** channels and **output** filters, respectively. The parameter $\mathbf{W}$ contains $n \times c$ kernels in total. Each kernel, for example, the kernel $\mathbf{W}_{j,k} \in \mathbb{R}^{h_f\times z_f}$ w.r.t. the k-th input channel and j-th filter, is a matrix with the dimension of $h_f \times z_f$. Let $\mathbf{X} \in \mathbb{R}^{N\times c\times h_{in}\times z_{in}}$ and $\mathbf{O} \in \mathbb{R}^{N\times n\times h_{out}\times z_{out}}$ 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 $\mathbf{X}_{i,k} \in \mathbb{R}^{h_{in}\times z_{in}}$ be the input feature map of the k-th channel for the i-th sample. The output feature map w.r.t. the j-th filter of $\mathbf{W}$ (*i.e.*, $\mathbf{W}_j$) for the i-th sample, denoted by $\mathbf{O}_{i,j} \in \mathbb{R}^{h_{out}\times z_{out}}$, is obtained by performing convolution on the input feature maps $\mathbf{X}_i$ of the i-th sample using the kernel $\mathbf{W}_j$:

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

where $*$ denotes the convolutional operation.

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

$$||\mathbf{W}||_{2,0} = \sum_{k=1}^{c} \Omega(\sum_{j=1}^{n} ||\mathbf{W}_{j,k}||_F) \leq \kappa_c^l, \tag{2}$$

where κ_c^l denotes the desired number of channels at layer l. Or equivalently, given a predefined pruning rate $\eta \in (0,1)$, it follows that $\kappa_c^l = \lceil (1-\eta)c \rceil$. When some channels of $\mathbf{W}$ are removed, the computation w.r.t. these channels can be effectively avoided. As a result, the pruned models would have fewer parameters and lower computational costs than the original models.

4 Proposed Method

Identifying the informative channels, also known as channel selection, is an important problem in channel pruning. Most existing methods [25], [50] conduct channel pruning by minimizing the reconstruction error of feature maps between the pre-trained model and the pruned one. However, merely minimizing the reconstruction error may cause some redundant channels to be mistakenly selected, even though they are actually irrelevant to the discriminative power of the network. This issue will be even severer when the network becomes deeper.

In this paper, we highlight that an informative channel, no matter where it is, should contribute to the discriminative power of the network; otherwise, it should be removed. Based on this intuition, we propose a discrimination-aware channel pruning (DCP) scheme to find the channels that actually contribute to the discriminative power. To this end, relying on a pre-trained model, we first introduce multiple discrimination-aware losses into the network to increase the discriminative power of the intermediate layers. Then, we conduct channel selection to select the most

1. We can also use other norms to compute the number of selected channels.

discriminative channel by considering both the discrimination-aware loss and the reconstruction error of the feature maps. In the following, we will illustrate the details of our method.

4.1 Motivation

We seek to perform channel pruning by keeping those channels that actually contribute to the discriminative power of the network. In practice, however, it is very difficult to measure the discriminative power of channels due to the complex operations (such as ReLU activation and Batch Normalization) in CNNs. One may consider a channel as an important one if the final loss $\mathcal{L}_f$ would sharply increase without it. However, for deep models, its shallow layers often have little discriminative power due to the long path of propagation. As a result, it is not practical to evaluate the discriminative power when the network is very deep.

To increase the discriminative power of the intermediate layers, one can introduce additional losses to the intermediate layers of the deep networks [14], [37], [71]. In this paper, we insert P discrimination-aware losses $\{\mathcal{L}_S^p\}_{p=1}^P$ 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. 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.

4.2 Construction of discrimination-aware loss

The construction of discrimination-aware loss $\mathcal{L}_S^p$ is very important in our method. As shown in Figure 1, $\mathcal{L}_S^p$ 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 apply batch normalization [16], [30] and ReLU [55] before performing the average pooling. In this way, the input feature maps for the loss at layer L_p, denoted by $\mathbf{F}^p(\mathbf{W})$, can be computed by

$$\mathbf{F}^p(\mathbf{W}) = \mathrm{AvgPooling}(\mathrm{ReLU}(\mathrm{BN}(\mathbf{O}^p))), \tag{3}$$

where $\mathbf{O}^p$ represents the output feature maps of layer L_p. Let $\mathbf{F}^{(p,i)}$ 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

$$\mathcal{L}_S^p(\mathbf{W}) = -\frac{1}{N}\left[\sum_{i=1}^{N}\sum_{t=1}^{m} I\{y^{(i)} = t\} \log \frac{e^{\boldsymbol{\theta}_t^{p\top}\mathbf{F}^{(p,i)}}}{\sum_{k=1}^{m} e^{\boldsymbol{\theta}_k^{p\top}\mathbf{F}^{(p,i)}}}\right], \tag{4}$$

where $I\{\cdot\}$ is the indicator function, $\boldsymbol{\theta}^p \in \mathbb{R}^{n_p \times m}$ 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 also use other losses as the additional loss, such as the additive angular margin loss [9]. (See results in Section 6.2).

4.3 Optimization problem for channel pruning

Since a pre-trained model contains very rich information about the learning task, similar to [51], we hope to reconstruct the feature maps in the pre-trained model by minimizing the reconstruction error of feature maps between the pre-trained model M^b and the pruned one. Formally, the reconstruction error can be measured by the mean squared error (MSE) between the feature maps of the baseline network and the pruned one as follows:

$$\mathcal{L}_M(\mathbf{W}) = \frac{1}{2N \cdot n \cdot h_{out} \cdot z_{out}} \sum_{i=1}^{N}\sum_{j=1}^{n} ||\mathbf{O}_{i,j}^b - \mathbf{O}_{i,j}||_F^2, \tag{5}$$

Algorithm 1 Discrimination-aware channel pruning

```
Input: Pre-trained model $M^b$, training data $\{\mathbf{x}_i, y_i\}_{i=1}^N$, and hyper-
    parameters $\{\kappa_c^l\}_{l=1}^L$.
Output: Pruned model $M$.
1: Initialize $M$ using $M^b$.
2: Insert losses $\{\mathcal{L}_S^p\}_{p=1}^P$ to layers $\{L_1, ..., L_P\}$, respectively.
3: Learn $\{\boldsymbol{\theta}^p\}_{p=1}^P$ and Fine-tune $M$ with $\{\mathcal{L}_S^p\}_{p=1}^P$ and $\mathcal{L}_f$.
4: Initialize $M^b$ using $M$.
5: for $p \in \{1, ..., P+1\}$ do
6:   for $l \in \{L_{p-1}+1, ..., L_p\}$ do
7:     Do Channel Selection for layer $l$ using Algorithm 2.
8:   end for
9: end for
```

where $\mathbf{O}_{i,j}^b \in \mathbb{R}^{h_{out} \times z_{out}}$ denotes the feature maps of M^b. By considering both discrimination-aware loss and reconstruction error, we have a joint loss function as follows:

$$\mathcal{L}(\mathbf{W}) = \lambda \mathcal{L}_M(\mathbf{W}) + \mathcal{L}_S^p(\mathbf{W}), \tag{6}$$

where λ balances the two terms.

Proposition 1. (Convexity of the loss function) *Let* $\mathbf{W}$ *be the model parameters of a considered layer. Given the discrimination-aware loss and the mean square loss defined in Eqs. (4) and (5), the joint loss function* $\mathcal{L}(\mathbf{W})$ *is convex w.r.t.* $\mathbf{W}$.[2]

By introducing the $\ell_{2,0}$-norm constraint, the optimization problem for discrimination-aware channel pruning becomes

$$\min_{\mathbf{W}} \ \mathcal{L}(\mathbf{W}), \quad \text{s.t.} \ ||\mathbf{W}||_{2,0} \leq \kappa_c^l, \tag{7}$$

where κ_c^l is the number of channels to be selected. In our method, the sparsity of $\mathbf{W}$ can be either determined by a predefined pruning rate (See Section 3) or automatically adjusted by the adaptive stopping conditions in Section 4.6.

4.4 Discrimination-aware channel pruning

By introducing P losses $\{\mathcal{L}_S^p\}_{p=1}^P$ to the intermediate layers, the proposed discrimination-aware channel pruning (DCP) method is shown in Algorithm 1. Starting from a pre-trained model M^b, DCP first updates the model M and learns the parameters $\{\boldsymbol{\theta}^p\}_{p=1}^P$. Then, DCP performs channel pruning with $(P+1)$ stages. Algorithm 1 is called discrimination-aware in the sense that the additional losses and the final loss are considered to fine-tune the model. Moreover, the additional losses will be used to select channels, as discussed below.

At the beginning of channel pruning, we first construct additional losses $\{\mathcal{L}_S^p\}_{p=1}^P$ and insert them at layer $\{L_1, ..., L_P\}$ (See Figure 1). Next, we learn the parameters $\{\boldsymbol{\theta}^p\}_{p=1}^P$ and fine-tune the model M at the same time with both the additional losses $\{\mathcal{L}_S^p\}_{p=1}^P$ and the final loss $\mathcal{L}_f$. During fine-tuning, all the parameters in M are updated. Here, with fine-tuning, the parameters regarding the additional losses can be well learned. [3] After doing fine-tuning with $\{\mathcal{L}_S^p\}_{p=1}^P$ and $\mathcal{L}_f$, the discriminative power of the intermediate layers can be significantly improved. Then, we initialize the baseline model M^b with the fine-tuned model M and perform channel pruning with $(P+1)$ stages. At each stage, for example, in the p-th stage, we consider the layers of the current stage independently, and conduct channel selection

2. The proof can be found in Section 7 in [92].
3. The details of the fine-tuning algorithm can be found in [14].

Algorithm 2 Greedy algorithm for channel selection

```
Input: Training data $\{\mathbf{x}_i, y_i\}_{i=1}^N$, model $M$, hyperparameters $\kappa_c^l$, $\epsilon$.
Output: Selected channel index set $\mathcal{A}$ and model parameters $\mathbf{W}_\mathcal{A}$.
1: Initialize $\mathcal{A}_0 \leftarrow \emptyset$, $\mathbf{W}^0 = \mathbf{0}$, and $t = 1$.
2: while (stopping conditions are not achieved) do
3:     Compute gradients of $\mathcal{L}$ w.r.t. $\mathbf{W}^{t-1}$: $\mathbf{G}^{t-1} = \partial\mathcal{L}/\partial\mathbf{W}^{t-1}$.
4:     Find the $B$ largest $||\mathbf{G}_{:,k}^{t-1}||_F$ and record their indices in $\mathcal{J}_t$.
5:     Let $\mathcal{A}_t \leftarrow \mathcal{A}_{t-1} \cup \mathcal{J}_t$.
6:     Solve the following Subproblem to update $\mathbf{W}_{\mathcal{A}_t}^{t-1}$:
           $$\min_{\mathbf{W}^{t-1}} \mathcal{L}(\mathbf{W}^{t-1}), \ \text{s.t.} \ \mathbf{W}_{\mathcal{A}_t^c}^{t-1} = \mathbf{0}. \quad (8)$$
7:     Set $\mathbf{W}^t \leftarrow \mathbf{W}^{t-1}$ and let $t \leftarrow t + 1$.
8: end while
```

for the layers in $\{L_{p-1}+1, ..., L_p\}$ with corresponding $\mathcal{L}_S^p$ and $\mathcal{L}_M$. Following [25], [86], we perform channel selection from the shallower layers to the deeper layers in this paper.

4.5 Greedy algorithm for channel selection

Due to the non-convexity of $\ell_{2,0}$-norm, directly optimizing Problem (7) is very difficult. To address this issue, following the general greedy methods in [3], [43], [74], [75], [84], we propose a greedy algorithm to solve Problem (7).

We show the details of the proposed greedy algorithm in Algorithm 2. At the beginning of the channel selection, we remove all the channels by setting $\mathbf{W}^0 = \mathbf{0}$. In each iteration, we first select those channels that actually contribute to the discriminative power of the network. Then, we solve Subproblem (8) with the selected channels only. We will give the details of selecting the channels with discriminative power and the subproblem optimization in the following subsections.

4.5.1 *Discrimination-aware channel selection*

At each iteration of Algorithm 2, we compute the gradients $\mathbf{G}_{:,k}^{t-1} = \partial\mathcal{L}/\partial\mathbf{W}_{:,k}^{t-1}$, where $\mathbf{W}_{:,k}^{t-1}$ denotes the parameters for the k-th input channel at iteration t. Since we set $\mathbf{W}^0 = \mathbf{0}$ at the beginning of channel selection, the initial loss value will be very large. Apparently, selecting any channel at the t-th iteration will decrease the loss function, and the channel with the largest gradient $||\mathbf{G}_{:,k}^{t-1}||_F$ will decrease the loss function the most. With the selection criteria, we choose B channels corresponding to the B largest $||\mathbf{G}_{:,k}^{t-1}||_F$ as active channels and record their indices into $\mathcal{J}_t$. Let $\mathcal{A}_t \subset \{1, \ldots, c\}$ be the index set of the selected channels up to iteration t, *i.e.*, $\mathcal{A}_t = \cup_i \mathcal{J}_i, i = 1, \ldots, t$. In general, once a channel is added into $\mathcal{J}_t$, it is unlikely to be selected in the following iteration. However, if we do not solve Subproblem (8) accurately, some of the selected channels may have a large value of $||\mathbf{G}_{:,k}^{t-1}||_F$, thus they might be chosen again. To avoid this issue, we propose choosing channels from $\{1, \ldots, c\}\backslash\mathcal{A}_{t-1}$ to form $\mathcal{J}_t$. In this way, there will be no overlapping channels among the $\mathcal{J}_i$'s, where $i = 1, \ldots, t$.

4.5.2 *Subproblem optimization*

Once $\mathcal{A}_t$ is determined, we optimize $\mathbf{W}^{t-1}$ w.r.t. the selected channels by minimizing Subproblem (8). Here, $\mathbf{W}_{\mathcal{A}_t^c}^{t-1}$ denotes the subtensor indexed by $\mathcal{A}_t^c$, and $\mathcal{A}_t^c$ is the complementary set of $\mathcal{A}_t$, *i.e.*, $\mathcal{A}_t^c = \{1, \ldots, c\}\backslash\mathcal{A}_t$. To solve the Subproblem in Eq. (8), we apply stochastic gradient descent (SGD) and update $\mathbf{W}_{\mathcal{A}_t}^{t-1}$ by

$$\mathbf{W}_{\mathcal{A}_t}^{t-1} \leftarrow \mathbf{W}_{\mathcal{A}_t}^{t-1} - \gamma \frac{\partial\mathcal{L}}{\partial\mathbf{W}_{\mathcal{A}_t}^{t-1}}, \tag{9}$$

where $\mathbf{W}_{\mathcal{A}_t}^{t-1}$ denotes the subtensor indexed by $\mathcal{A}_t$, and γ denotes the learning rate.

Note that when optimizing Subproblem (8), $\mathbf{W}_{\mathcal{A}_t}^{t-1}$ is warm-started from the fine-tuned model M. As a result, the optimization can be completed very quickly. Moreover, since Subproblem (8) is convex with respect to $\mathbf{W}^{t-1}$ for one layer, we do not need to consider all the data for optimization. To make a trade-off between the efficiency and performance, we sample a subset of images randomly from the training data for optimization. [4] Last, since we use SGD to update $\mathbf{W}_{\mathcal{A}_t}^{t-1}$, the learning rate γ should be carefully adjusted to achieve an accurate solution.

4.6 Stopping conditions

Given a predefined hyperparameter κ_c^l in Problem (7), Algorithm 2 will be stopped if $||\mathbf{W}^t||_{2,0} > \kappa_c^l$. However, κ_c^l is hard to be determined in practice. To solve this issue, we adopt the following two stopping conditions to terminate Algorithm 2.

Stopping condition 1: Since $\mathcal{L}$ is convex, $\mathcal{L}(\mathbf{W}^t)$ will monotonically decrease with the iteration index t in Algorithm 2. Therefore, we can adopt the following stopping condition:

$$\frac{|\mathcal{L}(\mathbf{W}^{t-1}) - \mathcal{L}(\mathbf{W}^t)|}{\mathcal{L}(\mathbf{W}^0)} \leq \epsilon, \tag{10}$$

where ϵ is a tolerance value.

Stopping condition 2: Directly using **stopping condition 1** can automatically determine the number of selected channels. However, in practice, if most of the channels are relatively informative and each of them contributes to the loss function, then the objective function value w.r.t. Eq. (6) may decrease very slowly during the greedy selection process. In this case, **stopping condition 1** may tend to select more channels or even all channels to achieve sufficiently small loss function value w.r.t. Eq. (6). To address this, we propose **stopping condition 2** which incorporates an additional constraint to force the number of selected channels to be no more than a predefined value for each layer. Specifically, given a minimum pruning rate η_{min}, we use the following condition:

$$\begin{aligned} &\frac{|\mathcal{L}(\mathbf{W}^{t-1}) - \mathcal{L}(\mathbf{W}^t)|}{\mathcal{L}(\mathbf{W}^0)} \leq \epsilon, \ \text{or} \\ &||\mathbf{W}^t||_{2,0} > \lceil(1-\eta_{\text{min}})c\rceil. \end{aligned} \tag{11}$$

If the above condition is achieved, the pruning process will be stopped earlier, and the number of selected channels will be automatically determined, *i.e.*, $||\mathbf{W}^t||_{2,0}$. As a result, the pruned models tend to have lower computational costs. Note that **stopping condition 2** is built on top of **stopping condition 1**. In practice, **stopping condition 1** alone often works well and **stopping condition 2** is satisfied accordingly. The comparisons of different stopping conditions are shown in Section 7.6.

4.7 Techniques for efficient implementations

For model compression methods, training cost has always been a key factor in real-world applications. Regarding this issue, we propose two methods to improve the training efficiency of DCP.

Single round fine-tuning. In Algorithm 1, fine-tuning plays a crucial role in improving the discriminative power of the intermediate layers. However, it leads to high computational costs. In our conference version [92], we perform fine-tuning and channel selection stage-wisely. In total, we fine-tune the network $P + 1$

4. We explore the number of samples in Section 11 in [92].

times, which increases the computational cost of channel pruning. To improve the efficiency of channel pruning, we fine-tune the network only once in step 3 of Algorithm 1, which reduces the times of fine-tuning. Moreover, the fine-tuning process is the same for the network with the same architecture but different pruning rates. Therefore, we can store the model after fine-tuning. In this way, the pruned networks with the same architecture but different pruning rates can skip fine-tuning in step 3 of Algorithm 1, which greatly reduces the computational cost of channel pruning. We show the effect of single round fine-tuning in Section 7.3.1.

Feature reusing. In Algorithm 2, we need to compute the input feature maps of layer l to compute the loss function. To obtain the input feature maps, we have to feed N images into the network from layer 1 to layer $l-1$. In [92], this process is repeated for each iteration, which incurs high computational overhead. Since we do not change the input feature maps during channel selection, we can store and reuse the input feature maps once they have been computed. In this way, we avoid the repeated calculation of the input feature maps, which greatly reduces the computational cost of channel selection. An empirical study on the effect of feature reusing can be found in Section 7.3.2.

5 Discrimination-aware kernel pruning

The proposed DCP can select the channels that actually contribute to the discriminative power of the network. However, there exists a limitation for DCP. Specifically, channel pruning assumes that all kernels in a channel are equally important, which may hamper the performance of the pruned model. In fact, some kernels may not contribute to the discriminative power of the network, resulting in performance degradation of the pruned model. More critically, in DCP, once a channel is pruned (or not selected), even if there may still exist some informative kernels related to it, we are no longer able to find and keep those useful kernels related to the removed channel, which may result in suboptimal performance.

To solve this issue, we propose a kernel pruning method called discrimination-aware kernel pruning (DKP) to further compress deep networks by removing redundant kernels. Similar to DCP, we introduce a variant of the $\ell_{2,0}$-norm constraint on $\mathbf{W}$ to conduct kernel pruning:

$$||\mathbf{W}||_{2,0}^{ker} = \sum_{j=1}^{n} \sum_{k=1}^{c} \Omega(||\mathbf{W}_{j,k}||_F) \leq \kappa_{ker}^{l}, \qquad (12)$$

where κ_{ker}^{l} is the desired number of kernels at layer l. When some kernels are removed, the corresponding computational cost w.r.t. the kernels can be effectively reduced.

Starting from a pre-trained model, DKP introduces P additional losses $\{\mathcal{L}_S^p\}_{p=1}^P$ evenly to the intermediate layers. Then, DKP fine-tunes the model using the addition losses $\{\mathcal{L}_S^p\}_{p=1}^P$ and the final loss $\mathcal{L}_f$ to improve the discriminative power of the intermediate layers. After that, DKP conducts kernel selection for each layer in a layer/stage-wise manner.

Similar to DCP, we introduce a variant of the $\ell_{2,0}$-norm constraint into the loss function in Eq. (6). Thus, the optimization problem for DKP can be formulated as:

$$\min_{\mathbf{W}} \ \mathcal{L}(\mathbf{W}), \quad \text{s.t. } ||\mathbf{W}||_{2,0}^{ker} \leq \kappa_{ker}^{l}. \qquad (13)$$

To solve Problem (13), we propose a greedy algorithm for kernel selection similar to Algorithm 2. At the beginning of the kernel selection, DKP removes all the kernels by setting $\mathbf{W}^0 = \mathbf{0}$. Instead of choosing the channels, we choose B kernels with the largest gradients $||\mathbf{G}_{j,k}^{t-1}||_F = \partial\mathcal{L}/\partial\mathbf{W}_{j,k}^{t-1}$ and put their indices into $\mathcal{J}_t$. Then, we update $\mathcal{A}_t$ by $\mathcal{A}_t = \mathcal{A}_{t-1} \cup \mathcal{J}_t$. Once $\mathcal{A}_t$ is determined, we optimize $\mathbf{W}^{t-1}$ w.r.t. the selected kernels by minimizing Subproblem (8), which is similar to DCP. In practice, we may directly apply DKP to compress models (denoted by **DKP Only**) or sequentially perform kernel selection after performing channel pruning (denoted by **DCP+DKP**). We investigate the effectiveness of DKP in Section 6.3.

Comparisons with DCP. Unlike DCP that focuses on channel selection, DKP seeks to select the informative kernels related to each channel, making it possible for model compression in a finer way (channel level vs. kernel level). Note that in each greedy optimization step, we can include a relatively large number of kernels (namely, a large B). Moreover, similar to DCP, during the subproblem optimization, we do not need to solve the problem exactly. So the complexity of DKP is similar to DCP. In fact, DKP has a comparable training cost with DCP in practice. For example, when pruning 90% FLOPs from VGGNet on CIFAR-10, the training cost of DKP is 3.25 GPU hours while the cost of DCP is 2.92 GPU hours. More empirical comparisons between DKP and DCP can be found in Section 6.3.

6 Experiments

To demonstrate the effectiveness of the proposed method, we apply DCP to various architectures, such as ResNet [21], MobileNetV1 [26] and MobileNetV2 [62], etc. We conduct experiments on both image classification and face recognition. In order to verify the effectiveness of DKP, we apply DKP to ResNet [21] and VGGNet [66]. All implementations are based on PyTorch [57].

We organize the experiments as follows. First, we evaluate the proposed DCP on image classification in Section 6.1. Second, we apply DCP to face recognition in Section 6.2. Last, we evaluate the proposed DKP in Section 6.3.

6.1 Experiments on image classification

6.1.1 *Compared methods*

To investigate the effectiveness of the proposed methods, we include the following methods for study: **DCP:** The proposed channel pruning method with a pre-defined pruning rate η. **Adapt-DCP:** DCP with adaptive **stopping condition 2** introduced in Section 4.6. **WM:** We shrink the width of a network by a fixed ratio and train it from scratch, which is known as the width multiplier [26]. **WM+:** Based on WM, we evenly insert additional losses into the network and train it from scratch. **Random-DCP:** Relying on DCP, we randomly choose channels instead of using the gradient-based strategy introduced in Algorithm 2.

We also consider several state-of-the-art channel pruning methods for comparison. On CIFAR-10, we compare DCP with NISP [83], ThiNet [51], CP [25], AMC [23], SFP [22], FPGM [24], PREC [39], Network Slimming [46], NRE [32], and DR [89]. On ILSVRC-12, we compare DCP with SFP [22], GAL [42], Taylor-FO-BN [52], SPP [77], CP [25], GDP [41], FPGM [24], ThiNet [51], SSR-L2 [40], NetAdapt [80], AMC [23], and MetaPruning [47]. Following [46], [51], we measure the computational cost of the pruned models by the number of floating point operations (FLOPs).

6.1.2 *Datasets and implementation details*

We evaluate the proposed DCP on two image classification datasets, including CIFAR-10 [35] and ILSVRC-12 [8]. CIFAR-10 consists

TABLE 1
Performance comparisons on CIFAR-10. "-" denotes that the results are not reported. Top-1 Err. ↑ is the Top-1 error gap between the pruned model and the baseline model.

Model	Method	Baseline Top-1 Err. (%)	Pruned Top-1 Err. (%)	Top-1 Err. ↑ (%)	#Param. ↓ (%)	#FLOPs ↓ (%)
ResNet-56	NISP [83]	-	-	+0.03	42.60	43.61
	ThiNet [51]	6.20	7.02	+0.82	49.67	49.78
	CP [25]	7.20	8.20	+1.00	-	50.00
	AMC [23]	7.20	8.10	+0.90	-	50.00
	SFP [22]	6.41	6.65	+0.24	-	52.60
	FPGM [24]	6.41	6.51	+0.10	-	52.60
	WM* [26]	6.26	6.76	+0.50	49.67	49.78
	WM+	6.26	6.65	+0.39	49.67	49.78
	Random-DCP	6.26	6.66	+0.40	49.67	49.78
	DCP	6.26	6.28	+0.02	49.67	49.78
	Adapt-DCP	6.26	**6.23**	**-0.03**	**68.48**	**54.80**
VGGNet	PFEC [39]	6.75	6.60	-0.15	64.00	34.18
	ThiNet [51]	6.01	6.15	+0.14	48.29	50.08
	CP [25]	6.01	6.33	+0.32	48.29	50.08
	Network Slimming [46]	6.34	6.20	-0.14	88.52	50.94
	NRE [32]	6.54	6.60	+0.06	92.70	67.60
	DR [89]	6.42	6.69	+0.27	88.30	68.63
	WM* [26]	6.02	6.39	+0.37	48.29	50.08
	WM+	6.02	6.12	+0.10	48.29	50.08
	Random-DCP	6.02	5.99	-0.03	48.29	50.08
	DCP	6.02	5.71	-0.31	48.29	50.08
	Adapt-DCP	6.02	**5.45**	**-0.57**	**91.69**	**69.81**
MobileNetV1	WM* [26]	6.04	6.39	+0.35	43.41	48.63
	WM+	6.04	6.35	+0.31	43.41	48.63
	Random-DCP	6.04	6.36	+0.32	43.41	48.63
	DCP	6.04	5.54	-0.50	43.41	48.63
	Adapt-DCP	6.04	**5.43**	**-0.61**	**78.26**	**66.12**
MobileNetV2	WM* [26]	5.53	5.98	+0.45	23.59	27.07
	WM+	5.53	5.93	+0.40	23.59	27.07
	Random-DCP	5.53	5.96	+0.42	23.59	27.07
	DCP	5.53	5.37	-0.16	23.59	27.07
	Adapt-DCP	5.53	**5.28**	**-0.25**	**40.06**	**34.44**

* The results were obtained by our implementations.

of 50k training samples and 10k testing images with 10 classes. ILSVRC-12 contains 1.28 million training samples and 50k testing images for 1,000 classes.

Based on the pre-trained model, we apply our method to select informative channels. We first introduce additional losses to increase the discriminative power of the intermediate layers. In practice, we determine the number of additional losses according to the depth of the network (See Section 7.8). Specifically, we insert 3 additional losses to ResNet-50 and ResNet-56, and 2 additional losses to VGGNet, ResNet-18, MobileNetV1, and MobileNetV2. Then, we fine-tune the model with both the additional losses and the final loss. On CIFAR-10, we fine-tune 100 epochs using a mini-batch size of 128. The learning rate starts from 0.1 and is divided by 10 at epochs 40 and 60. On ILSVRC-12, we fine-tune 60 epochs using a mini-batch size of 256. The learning rate is initialized to 0.01 and divided by 10 at epochs 36, 48, and 54.

After doing fine-tuning, we conduct channel selection in a stage-wise manner by considering the corresponding additional loss and the reconstruction error of the feature maps. For ResNet-56 and ResNet-50, we set η and η_{min} to 0.5 and 0.4, respectively. For VGGNet, MobileNetV1 and MobileNetV2, η and η_{min} are set to 0.3 and 0.2, respectively. In our experiment, λ and B are set to 1.0 and 2.0, respectively.

After doing channel selection, we fine-tune the whole network with the selected channels only. We use SGD with nesterov [56] for optimization. The momentum and weight decay are set to 0.9 and 1×10^{-4}, respectively. 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 epochs 160 and 240. For ResNet-18 and ResNet-50 on ILSVRC-12, we fine-tune the network for 60 epochs with a mini-batch size of 256. The learning rate starts at 0.01 and is divided by 10 at epochs 36, 48 and 54. For MobileNetV1 and MobileNetV2 on ILSVRC-12, we fine-tune for 150 epochs and 250 epochs with a mini-batch size of 256. Following [26], [62], we set the weight decay to 4×10^{-5}. The learning rate is initialized to 0.09 and decreased to 0 following the cosine function [49].

6.1.3 Comparisons on CIFAR-10

We apply DCP to prune ResNet-56, VGGNet, MobileNetV1, and MobileNetV2 and compare the performance on CIFAR-10. We report the results in Table 1.

From Table 1, we have following observations. First, the models pruned by DCP significantly outperform those pruned by Random-DCP. For example, on VGGNet, DCP reduces the error by 0.28% compared with Random-DCP, which implies the effectiveness of the proposed channel selection strategy. Second, the inserted additional losses improve the performance of the networks. Specifically, WM+ of VGGNet achieves better performance than WM. Third, our proposed DCP shows much better performance than WM+. For example, on VGGNet, DCP outperforms WM+ by 0.41% on the

TABLE 2
Performance comparisons on ILSVRC-12. "-" denotes that the results are not reported. "Top-1 Err. ↑" and "Top-5 Err. ↑" denote the Top-1 and Top-5 error gap between the pruned model and the baseline model.

Model	Method	Baseline Top-1 Err. (%)	Pruned Top-1 Err. (%)	Top-1 Err. ↑ (%)	Baseline Top-5 Err. (%)	Pruned Top-5 Err. (%)	Top-5 Err. ↑ (%)	#Param. ↓ (%)	#FLOPs ↓ (%)
ResNet-50	SFP [22]	23.85	25.39	+1.54	7.13	7.94	+0.81	-	41.80
	GAL [42]	23.85	28.05	+4.20	7.13	9.06	+1.93	16.86	43.03
	Taylor-FO-BN [52]	23.82	25.50	+1.68	-	-	-	44.53	44.99
	SPP [77]	-	-	-	8.80	9.60	+0.80	-	50.00
	CP [25]	-	-	-	7.80	9.20	+1.40	-	50.00
	GDP [41]	24.87	28.11	+3.24	7.70	9.29	+1.59	-	51.30
	FPGM [24]	23.85	25.17	+1.32	7.13	7.68	+0.55	-	53.50
	ThiNet [51]	24.70	27.97	+3.27	7.80	9.01	+1.21	51.55	55.50
	SSR-L2 [40]	24.88	28.53	+3.65	7.70	9.81	+2.11	52.94	56.41
	WM* [26]	23.99	26.47	+2.48	7.07	8.52	+1.45	51.55	55.50
	WM*(Scratch-B) [48]	23.99	25.44	+1.45	7.07	8.03	+0.96	51.55	55.50
	DCP	23.99	25.01	+1.02	7.07	7.80	+0.73	51.55	**55.50**
	Adapt-DCP	23.99	**24.85**	**+0.86**	7.07	**7.70**	**+0.63**	**54.99**	52.41
MobileNetV1	NetAdapt [80]	29.10	30.90	+1.80	-	-	-	-	12.63
	AMC [23]	29.10	29.51	+0.41	10.10	10.69	+0.59	43.09	48.42
	MetaPruning [47]	29.10	29.60	+0.50	-	-	-	-	50.62
	WM [26]	29.10	31.60	+2.50	10.10	11.80	+1.70	38.90	42.86
	DCP	29.12	29.52	+0.40	10.22	**10.61**	**+0.39**	**45.47**	49.67
	Adapt-DCP	29.12	**29.50**	**+0.38**	10.22	10.78	+0.56	43.27	**51.25**
MobileNetV2	AMC [23]	28.20	29.15	+0.95	9.00	10.09	+1.09	33.51	30.06
	MetaPruning [47]	28.20	28.80	+0.60	-	-	-	-	27.67
	WM [62]	28.20	30.20	+2.00	9.00	10.40	+1.40	25.02	40.52
	DCP	28.12	28.79	+0.67	9.70	10.01	+0.31	34.24	**32.21**
	Adapt-DCP	28.12	**28.61**	**+0.49**	9.70	**9.91**	**+0.21**	**35.01**	30.67

* The results were obtained by our implementations.

TABLE 3
Inference acceleration of the pruned models on ILSVRC-12. We report the forward time tested on a mobile CPU (Qualcomm Snapdragon 845).

Network	η	#Param. (M)	#FLOPs (M)	Mobile CPU Time (ms)	Speedup Ratio
MobileNetV1	0.00	4.23	1132.44	107.25	1.00
	0.30	**2.31**	**569.92**	**55.65**	**1.93**
MobileNetV2	0.00	3.50	594.87	62.88	1.00
	0.30	**2.30**	**403.24**	**44.25**	**1.42**

testing accuracy. Fourth, compared with DCP, Adapt-DCP further improves the performance of the pruned models with much fewer parameters and FLOPs. Specifically, when applying Adapt-DCP on VGGNet, the pruned model with 91.69% and 69.81% reductions in parameters and FLOPs even lowers the testing error by 0.26% compared with DCP.

Compared with several state-of-the-art methods, our method achieves the best performance. For example, ThiNet [51] prunes VGGNet by 48.29% of the parameters and 50.08% of the FLOPs with a 0.14% drop in terms of the Top-1 accuracy. In contrast, our proposed DCP achieves the same speedup ratio with a **0.31%** improvement in terms of the Top-1 accuracy. Moreover, for VGGNet, Adapt-DCP outperforms NRE [32] and DR [89] and obtains a **91.69%** reduction in the model size and **69.81%** in FLOPs. These results show the superior performance of our proposed DCP and Adapt-DCP.

6.1.4 Comparisons on ILSVRC-12

To verify the effectiveness of the proposed method on the large-scale dataset, we evaluate our method on ILSVRC-12 and report the Top-1 and Top-5 errors with single view evaluation in Table 2. We first apply DCP to compress ResNet-50. Compared with WM (Scratch-B) [48], which leads to 1.45% increase in terms of the Top-1 error, DCP only results in **1.02%** degradation in terms of the Top-1 accuracy, which demonstrates the effectiveness of the pruning scheme in DCP. Compared with FPGM [24], DCP achieves 0.30% improvement in terms of the Top-1 accuracy with a 55.50% reduction in FLOPs. More critically, the model pruned by Adapt-DCP with the smaller model size even outperforms DCP by 0.16% and 0.10% on Top-1 and Top-5 accuracy, respectively.

We also apply DCP to prune compact and efficient neural networks, such as MobileNetV1 [26] and MobileNetV2 [62]. Compared with AMC [23] and MetaPruning [47], our proposed DCP achieves better performance. For example, on MobileNetV2, DCP outperforms AMC [23] by 0.36% in terms of the Top-1 accuracy. Moreover, Adapt-DCP pruned MobileNetV2 outperforms MetaPruning [47] by 0.11% in terms of the Top-1 accuracy. These results demonstrate the effectiveness of DCP and Adapt-DCP.

6.1.5 Model deployment to mobile devices

We further deploy the pruned models to a mobile device. We perform the evaluations on a Xiaomi 8 smartphone, which is equipped with a 2.8 GHz Qualcomm Snapdragon 845 mobile processor. The test-phase computation is carried out on a single large CPU core without GPU acceleration. We report the results in Table 3.

Compared with the pre-trained model, MobileNetV1 with a pruning rate of 30% achieves nearly 2× acceleration on the mobile phone. Moreover, the execution of our pruned MobileNetV2 only requires 44.25ms, which is much lower than the pre-trained one. These results show that our methods are able to compress the compact models to significantly reduce the inference time.

TABLE 4
Performance comparisons of different methods on face recognition. "VR" refers to Verification TAR (True Accepted Rate) and "FAR10^{-6}" refers to the False Accepted Rate at 10^{-6}. We do not report the inference time of SphereFace and CosFace due to its large model size and computational costs.

Model	Validation Accuracy (%)			VR@FAR10^{-6} (%)	#Param. (M)	#FLOPs (G)	Mobile CPU	Speedup
	LFW	CFP-FP	AgeDB-30	MegaFace			Times (ms)	Ratio
SphereFace [63]	99.76	93.70	97.56	97.43	65.16	24.17	-	-
CosFace [76]	99.80	94.40	97.91	98.36	65.16	24.17	-	-
LResNet34E-IR [10]	99.72	96.39	98.03	96.71	31.81	7.10	391.52	1.00
LResNet34E-IR (prune 25% channels)	99.70	**96.71**	97.63	**97.03**	27.12	5.35	308.72	1.27
LResNet34E-IR (prune 50% channels)	99.70	95.90	97.70	96.14	**22.42**	**3.60**	**232.61**	**1.68**
MobileFaceNet [6]	99.50	92.23	95.63	90.38	1.00	0.44	36.90	1.00
MobileFaceNet (prune 25% channels)	99.38	92.17	95.47	90.33	**0.79**	**0.33**	**28.71**	**1.29**

TABLE 5
Comparisons of DCP and DKP on CIFAR-10. The Top-1 error (%) of the pre-trained ResNet-56 and VGGNet are 6.26 and 6.02, respectively.

Model	Method	Pruned Top-1 Err. (%)	Top-1 Err. ↑ (%)	#Param. ↓ (%)	#FLOPs ↓ (%)
ResNet-56	DCP	6.28	+0.02	49.67	49.78
	DCP+DKP	6.23	-0.03	**51.53**	**50.59**
	DKP Only	**6.18**	**-0.08**	49.52	50.00
	DCP	7.02	+0.76	68.29	68.44
	DCP+DKP	6.96	+0.70	**68.62**	**68.84**
	DKP Only	**6.71**	**+0.45**	67.78	68.20
	Adapt-DCP	6.23	-0.03	68.48	54.80
	Adapt-DKP	**6.17**	**-0.08**	**68.72**	**58.42**
VGGNet	DCP	6.24	+0.22	72.01	74.30
	DCP+DKP	6.17	+0.15	73.16	74.26
	DKP Only	**6.15**	**+0.13**	**74.33**	**74.54**
	DCP	7.98	+1.96	88.42	90.24
	DCP+DKP	7.74	+1.72	89.39	90.42
	DKP Only	**7.52**	**+1.50**	**90.41**	**90.50**
	Adapt-DCP	**5.45**	**-0.57**	91.69	69.81
	Adapt-DKP	5.48	-0.54	**91.90**	**76.48**

TABLE 6
Comparisons of DCP and DKP on ILSVRC-12. The Top-1 and Top-5 error (%) of the pre-trained ResNet-18 are 30.36 and 11.02, respectively.

Model	Method	Top-1 Err. (%)	Top-5 Err. (%)	#Param. ↓ (%)	#FLOPs ↓ (%)
ResNet-18	DCP	32.64	12.38	47.01	46.22
	DCP+DKP	32.57	12.20	**47.31**	46.40
	DKP Only	**32.46**	**12.04**	47.12	**46.56**

6.2 Experiments on Face Recognition

6.2.1 Compared methods

We apply the proposed DCP to LResNet34E-IR [10] and MobileFaceNet [6] on face recognition. To evaluate the proposed DCP method, we consider several face recognition models for comparison, including SphereFace [45] and CosFace [76].

6.2.2 Datasets and implementation details

We evaluate the proposed DCP method on four benchmark datasets, including LFW [28], CFP-FP [64], AgeDB-30 [54], and MegaFace [34]. LFW [28] contains 13,233 face images from 5,749 identities. CFP [64] consists of 500 identities, each with 10 frontal and 4 profile images. AgeDB-30 [54] contains 12,240 images of 440 identities. MegaFace [34] is a very challenging benchmark dataset to evaluate the performance of face recognition methods at a scale of million distractors.

We use the refined MS-Celeb-1M [18] released by [9] for training. The training dataset consists of 5.8M face images from 85k individuals. With the same settings as [9], we first train LResNet34E-IR [10] and MobileFaceNet [6] from scratch. Then, we apply our proposed DCP to compress the pre-trained models.

Before channel pruning, we first insert 2 additive angular margin losses [9] into LResNet34E-IR and MobileFaceNet. Then, we fine-tune 15 epochs with both the discrimination-aware losses and the final loss. The learning rate is initialized to 0.01 and divided by 10 at epochs 4, 8, and 12.

After doing fine-tuning, we perform channel selection to select the informative channels. After doing channel selection, we fine-tune the whole network for 28 epochs. The learning rate is initialized to 0.01 and divided by 10 at epochs 8, 16 and 24. We use SGD with a mini-batch size of 512 for optimization.

6.2.3 Performance comparisons

We report the results in Table 4. From the results, we observe that the pruned models with small pruning rates achieve nearly the same performance as the pre-trained model. For example, for LResNet34E-IR, the pruned model with a pruning rate of 25% even outperforms the pre-trained model on CFP-FP and MegaFace. Moreover, for MobileFaceNet, the pruned model achieves comparable performance as the pre-trained model with only 0.79M parameters and 28.71ms for inference, which is suitable for resource-limited devices.

Compared with SphereFace [63] and CosFace [76], our pruned LResNet34E-IR achieves comparable performance with a much smaller number of parameters and FLOPs. Even with a pruning rate of 50%, our pruned LResNet34E-IR still achieves comparable performance to the pre-trained model. These results demonstrate the effectiveness of the proposed DCP on face recognition.

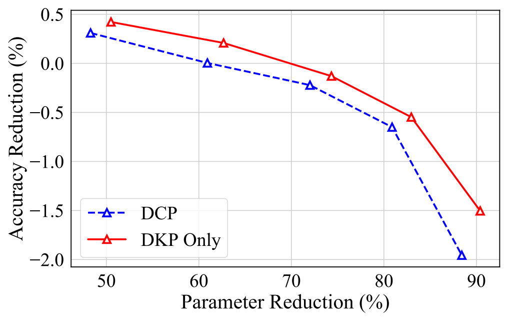
6.3 Effectiveness of DKP

6.3.1 Compared methods

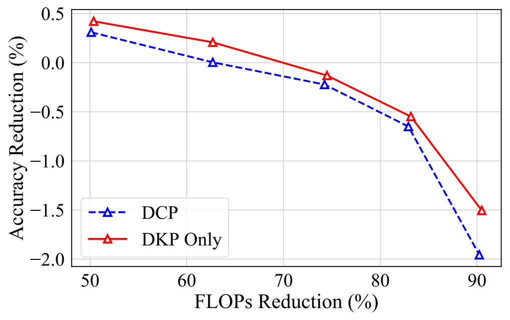
To investigate the effectiveness of DKP, apart from DCP and Adapt-DCP, we include the following methods for comparisons. **DKP Only**: The proposed kernel pruning method with a fixed pruning rate. **DCP+DKP**: We sequentially perform channel selection and kernel selection. **Adapt-DKP:** DKP Only with adaptive **stopping condition 2.**

6.3.2 Datasets and implementation details

We evaluate the performance of the pruned models on CIFAR-10 and ILSVRC-12. For kernel selection, we use the same additional losses introduced in DCP. We set B to a value that is equal to the



(a) Accuracy reduction vs. parameter reduction



(b) Accuracy reduction vs.. FLOPs reduction

Fig. 2. Comparisons of DCP and DKP in terms of Top-1 accuracy reduction vs. parameter/FLOPs reduction. We apply DCP/DKP to prune VGGNet on CIFAR-10 with different pruning rates.

number of kernels in a channel. After doing kernel selection, we fine-tune the whole network with the selected kernels only.

6.3.3 Performance comparisons

We apply DKP to select informative kernels from ResNet-56, VGGNet, and ResNet-18. From Table 5 and Table 6, directly adopting DKP to prune models achieve the best performance on both CIFAR-10 and ILSVRC-12. For example, for ResNet-18, DKP outperforms DCP by 0.18% in terms of the Top-1 accuracy with 47.12% and 46.56% reductions in the parameters and FLOPs. Moreover, with Adapt-DKP, the resultant models obtain comparable performance but lower computational costs than the ones of Adapt-DCP. Note that the improvement of DCP+DKP over DCP is smaller than that of DKP Only. In DCP+DKP, we simply perform kernel selection after doing channel selection. As a result, the performance highly depends on DCP. Specifically, once a channel is pruned (or not selected), even if there may still exist some informative kernels related to it, we are no longer able to find and keep those useful kernels related to the removed channel, which may result in suboptimal performance and the marginal improvement over DCP. From Figure 2, DKP Only consistently outperforms DCP under different pruning rates. To be specific, VGGNet pruned by DKP Only with a 90.41% reduction in parameters reduces the Top-1 error by 0.46% compared with the one pruned by DCP. These results demonstrate the effectiveness of DKP.

7 Ablation Studies

In this section, we conduct ablative studies for the proposed DCP. 1) We investigate the effect of using different pruning rates in Section 7.1. 2) We explore the effect of using different λ in Section 7.2. 3) We explore the effect of efficient training strategies in Section 7.3. 4) We explore the effect of using different B in Section 7.4. 5) We discuss the effect of the tolerance ϵ in the proposed stopping conditions in Section 7.5. 6) We compare different stopping conditions in Section 7.6. 7) We visualize the feature maps w.r.t. the pruned/selected channels of ResNet-18 in Section 7.7. 8) We explore the influence of the number of additional losses in Section 7.8.

TABLE 7
Comparisons on ResNet-18 and ResNet-50 with different pruning rates. We report the Top-1 and Top-5 error (%) on ILSVRC-12.

Model	η	Top-1 Err. (%)	Top-5 Err. (%)	#Param. ↓ (%)	#FLOPs ↓ (%)
ResNet-18	0%	30.36	11.02	-	-
	30%	**30.77**	**11.03**	28.05	27.49
	50%	32.64	12.38	47.01	46.22
	70%	35.89	14.22	**65.70**	**64.25**
ResNet-50	0%	23.99	7.07	-	-
	30%	**23.63**	**6.96**	33.40	35.72
	50%	25.01	7.80	51.55	55.50
	70%	27.28	8.83	**65.90**	**71.09**

TABLE 8
Pruning results of ResNet-56 and MobileNetV1 with different λ. We report the testing error (%) on CIFAR-10.

λ	0	0.1	0.5	1	1($\mathcal{L}_M$ only)	5	10
ResNet-56	5.94	5.96	5.87	**5.80**	5.99	5.85	5.93
MobileNetV1	5.66	5.46	5.59	5.54	5.43	**5.42**	5.56

7.1 Performance with different pruning rates

To study the effect of using different pruning rates η, we prune 30%, 50%, and 70% channels from ResNet-18 and ResNet-50, and evaluate the pruned models on ILSVRC-12. The experimental results are shown in Table 7.

From the results, the pruned models perform worse with the increase of the pruning rate. However, our pruned ResNet-50 with a pruning rate of 30% outperforms the pre-trained model with **0.36%** and **0.11%** improvement in terms of the Top-1 and Top-5 accuracy, respectively. It may attribute that DCP serves as a regularization as it effectively reduces model redundancy by selecting the most discriminative channels for each layer. Additionally, the performance degradation of ResNet-50 is smaller than that of ResNet-18 under the same pruning rate. For example, when pruning 50% of the channels, it only leads to 1.02% increase in terms of the Top-1 error for ResNet-50. In contrast, it results in 2.28% increase of the Top-1 error for ResNet-18. It can be attributed to that, compared with ResNet-18, ResNet-50 is more redundant with more parameters, thus it is easier to be pruned.

7.2 Effect of the trade-off hyperparameter λ

We prune 30% channels of ResNet-56 and MobileNetV1 on CIFAR-10 with different λ values. We report the testing error in Table 8. From the table, the performance of the pruned models first improves and then degrades with the increasing λ. Here, a larger λ implies that more emphasis is placed on the reconstruction error (See Eq. (6)). This demonstrates the effectiveness of the 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 models, which strongly supports the motivation to select the important channels by $\mathcal{L}_S^p$ and $\mathcal{L}_M$.

TABLE 9
Effect of efficient single round fine-tuning. We report the testing error (%) and the time of channel pruning on CIFAR-10 and ILSVRC-12.

Model	Method	Top-1 Err. (%)	Top-5 Err. (%)	Time (hour)
ResNet-56 (CIFAR-10)	DCP [92]	6.51	-	7.58
	DCP	**6.26**	-	**2.83**
ResNet-18 (ILSVRC-12)	DCP [92]	**35.88**	**14.32**	31.78
	DCP	35.89	14.34	**5.18**

TABLE 10
Effect of feature reusing. We report the testing error (%) and the time of channel pruning on CIFAR-10 and ILSVRC-12.

Model	Method	Top-1 Err. (%)	Top-5 Err. (%)	Time (hour)
ResNet-56 (CIFAR-10)	without feature reusing	**6.26**	-	2.83
	with feature reusing	6.28	-	**1.85**
ResNet-18 (ILSVRC-12)	without feature reusing	**35.89**	**14.34**	5.18
	with feature reusing	35.90	**14.34**	**3.02**

TABLE 11
Pruning results of ResNet-56 with different B. We report the testing error (%) and the time of channel pruning on CIFAR-10.

B	Testing Err. (%)	Time (hour)
1	**6.22**	1.85
2	6.28	0.90
4	6.32	**0.48**

Note that setting λ to 1.0 does not lead to the best performance considering different architectures and datasets. For simplicity, we set λ to 1.0 by default in our experiments.

7.3 Effect of the improved training techniques

7.3.1 Effect of the single round fine-tuning

To evaluate the effectiveness of the single round fine-tuning, we prune 50% channels of ResNet-56 on CIFAR-10 and 70% channels of ResNet-18 on ILSVRC-12. We compare the proposed DCP with the conference version [92] and report the testing error and time of channel pruning in Table 9. From the table, the proposed DCP with single round fine-tuning significantly reduces the training time while maintaining comparable performance to the previous DCP [92]. For example, DCP pruned ResNet-18 on ILSVRC-12 reduces the computational cost by **6.14**$\times$ compared with the conference version. These results demonstrate the effectiveness and efficiency of the improved fine-tuning strategy.

7.3.2 Effect of the feature reusing

To study the effect of the feature reusing, we prune 50% channels of ResNet-56 on CIFAR-10 and 70% channels of ResNet-18 on ILSVRC-12. We report the testing error and time of channel pruning in Table 10. As shown in the table, pruning with the feature reusing achieves comparable performance to the model without the feature reusing. However, pruning with the feature reusing greatly reduces the time of channel pruning. For example, DCP pruned ResNet-18 with the feature reusing reduces the computational cost by **1.72**$\times$. Due to the superior performance of the feature reusing, we use it by default in our experiments.

TABLE 12
Effect of ϵ for channel selection. We prune VGGNet with different ϵ and report the testing error (%) on CIFAR-10.

Model	ϵ	Top-1 Err. (%)	#Param. $\downarrow$ (%)	#FLOPs $\downarrow$ (%)
with **stopping condition 1**	5×10^{-4}	6.29	**96.16**	**67.87**
	3×10^{-4}	5.91	94.59	56.73
	1×10^{-4}	**5.31**	90.93	30.80
with **stopping condition 2**	5×10^{-4}	6.36	**95.95**	**84.61**
	3×10^{-4}	5.87	94.74	79.66
	1×10^{-4}	**5.45**	91.69	69.81

TABLE 13
Pruning results of ResNet-56 with different stopping conditions. We report the testing error (%) on CIFAR-10.

Model	Testing Err. (%)	#Param. $\downarrow$ (%)	#FLOPs $\downarrow$ (%)
with **stopping condition 1**	6.36	65.20	46.82
with **stopping condition 2**	**6.23**	**68.48**	**54.80**

7.4 Effect of the hyperparameter B

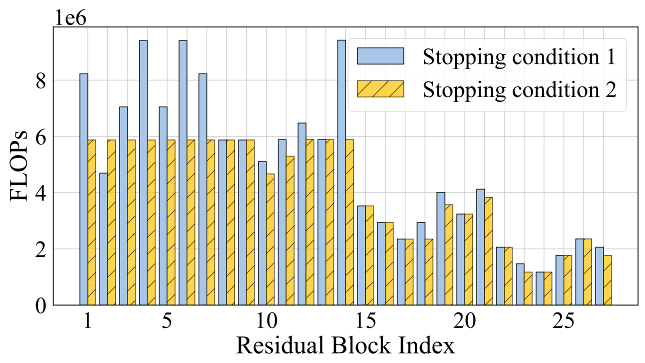
To evaluate the effect of B, we prune 50% channels of ResNet-56 on CIFAR-10 with different B and report the results in Table 11. Here, a larger B indicates that we select more channels at each iteration in Algorithm 2. As a result, fewer iterations are required. From Table 11, the channel pruning time decreases with the increase of B while the performance of DCP degrades. For example, DCP pruned ResNet-56 with $B = 4$ reduces the computational cost by 4.85$\times$. To make a trade-off between accuracy and complexity, we set B to 2 in our experiments.

7.5 Effect of the tolerance ϵ in stopping conditions

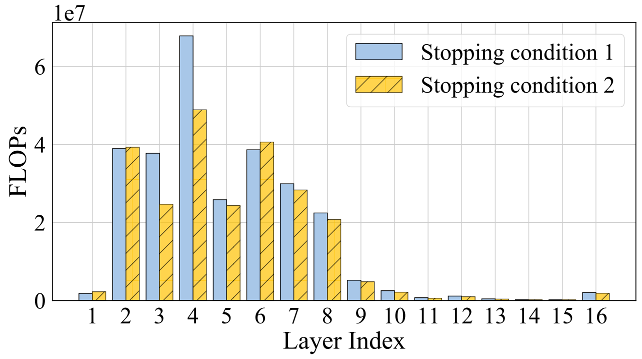
We test different tolerance values in Eq. (10) and Eq. (11). Here, we prune VGGNet on CIFAR-10 with $\epsilon \in \{1\times10^{-4}, 3\times10^{-4}, 5\times10^{-4}\}$. We show the experimental results in Table 12. In general, a smaller ϵ will lead to a more rigorous stopping condition. Hence, more channels will be selected. As a result, the performance of the pruned models is improved with the decrease of ϵ. This experiment demonstrates the effectiveness of stopping conditions for automatically determining the pruning rate.

7.6 Effect of different stopping conditions

We investigate the effect of different stopping conditions mentioned in Section 4.6. To this end, we prune VGGNet and ResNet-56 using Adapt-DCP with different stopping conditions and report the results in Tables 12 and 13. From the results, the resultant models with **stopping condition 2** obtain comparable performance but significantly lower computational costs than the ones of **stopping condition 1**. For example, VGGNet pruned with **stopping condition 2** reduces 69.81% FLOPs while the one pruned with **stopping condition 1** only reduces 30.80% FLOPs when $\epsilon = 1e^{-4}$. To further explore the effect of different stopping conditions, we visualize the number of FLOPs w.r.t. each layer of the pruned ResNet-56 and VGGNet. From Figure 3, the proposed **stopping condition 2** is able to prevent DCP from selecting too many channels, and significantly reduces the computational cost of the compressed models. Due to the superior performance of **stopping condition 2** in FLOPs reduction, we use it by default.



(a) FLOPs w.r.t. each layer in the pruned ResNet-56.



(b) FLOPs w.r.t. each layer in the pruned VGGNet.

Fig. 3. Number of FLOPs w.r.t. each layer in the pruned ResNet-56 and VGGNet with different stopping conditions.

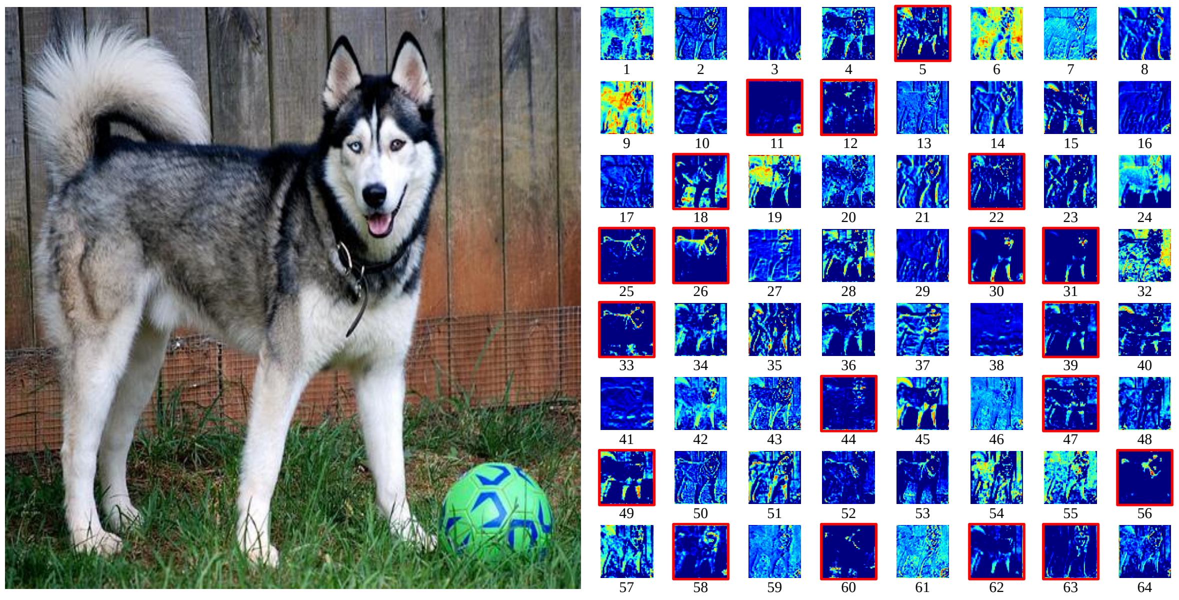


Fig. 4. Visualization of the feature maps from the second layer of the first residual block in ResNet-18 on ILSVRC-12. Feature maps with and without the red bounding boxes are the pruned and selected channels, respectively.

7.7 Visualization of feature maps

We further visualize all the feature maps w.r.t. the pruned and selected channels from the first residual block in ResNet-18 in Figure 4. From the results, the feature maps of the pruned channels are less informative than those of the selected channels. Moreover, the proposed DCP is able to remove those highly activated channels that are redundant or nearly identical along with the weakly activated channels. For example, DCP removes the 49-th channel since it is similar to the 45-th channel (cosine similarity: 0.82). Even a channel with high activation values (*e.g.*, the 5-th channel), DCP still removes it as it does not contribute to the discriminative power of the network. These results prove that the proposed DCP selects the channels with strong discriminative power for the network.

7.8 Effect of the number of additional losses

We prune 50% channels from ResNet-56 and 30% channels from MobileNetV1 with different number of additional losses on CIFAR-10. From Table 14, adding more losses results in better performance. For example, ResNet-56 with three additional losses outperforms the one with one additional loss. However, adding too many losses may lead to a little gain in performance but significantly increase the computational cost. Heuristically, we find that adding losses every 5-10 blocks is sufficient to make a good trade-off between accuracy and complexity.

TABLE 14
Effect of the number of additional losses. We prune 50% channels from ResNet-56 and 30% channels from MobileNetV1. We report the testing error on CIFAR-10. The testing error (%) of pre-trained ResNet-56 and MobilenNet v1 are 6.26 and 6.04, respectively.

#Additional Losses	Err. gap (%)	
	ResNet-56	MobileNetV1
1	+0.10	-0.28
3	+0.02	-0.54
5	+0.01	-0.65
7	-0.08	-0.70

8 Conclusion

In this paper, we have proposed a discrimination-aware channel pruning (DCP) method for the compression of deep neural networks. Specifically, we first introduce additional losses to improve the discriminative power of the network and then perform channel selection in a layer-wise manner. In order to select informative channels, we have formulated the channel pruning as a sparsity-induced optimization problem and proposed a greedy algorithm to solve it. Based on DCP, we have further proposed several techniques to accelerate the channel pruning process. Moreover, we have also proposed a discrimination-aware kernel pruning (DKP) method to perform model compression at the kernel level. Extensive results on both image classification and face recognition tasks have demonstrated the effectiveness of the proposed methods.

In the future, we may extend our method in two aspects. First, in the proposed DCP, we perform channel/kernel selection in a layer-wise manner. This strategy, however, may result in suboptimal performance as only a single layer is considered each time. To address this, we may consider multiple layers/blocks each time to improve the pruning performance. Second, we will extend the proposed DCP scheme for model quantization. Specifically, based on DCP, we can conduct the quantization for each selected channel/kernel, and use the reconstruction residual to choose the next channel/kernel. In this way, we can perform channel/kernel pruning and quantization simultaneously.

ACKNOWLEDGMENTS

This work was partially supported by Key-Area Research and Development Program of Guangdong Province 2018B010107001, National Natural Science Foundation of China (NSFC) 61836003 (key project), National Natural Science Foundation of China (NSFC) 62072190, and Program for Guangdong Introducing Innovative and Enterpreneurial Teams 2017ZT07X183.

REFERENCES

[1] J. M. Alvarez and M. Salzmann. Learning the number of neurons in deep networks. In *Advances in Neural Information Processing Systems*, pages 2270–2278, 2016.

[2] S. Anwar, K. Hwang, and W. Sung. Structured pruning of deep convolutional neural networks. *ACM Journal on Emerging Technologies in Computing Systems*, 13(3):32, 2017.

[3] S. Bahmani, B. Raj, and P. T. Boufounos. Greedy sparsity-constrained optimization. *Journal of Machine Learning Research*, 14:807–841, 2013.

[4] A. Bulat and Y. Tzimiropoulos. Hierarchical binary cnns for landmark localization with limited resources. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2018.

[5] J. Cao, Y. Guo, Q. Wu, C. Shen, J. Huang, and M. Tan. Adversarial learning with local coordinate coding. In *International Conference on Machine Learning*, volume 80, pages 707–715, 2018.

[6] S. Chen, Y. Liu, X. Gao, and Z. Han. Mobilefacenets: Efficient cnns for accurate real-time face verification on mobile devices. In *Chinese Conference on Biometric Recognition*, pages 428–438, 2018.

[7] J. Choi, Z. Wang, S. Venkataramani, P. I.-J. Chuang, V. Srinivasan, and K. Gopalakrishnan. Pact: Parameterized clipping activation for quantized neural networks. *arXiv preprint arXiv:1805.06085*, 2018.

[8] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[9] J. Deng, J. Guo, X. Niannan, and S. Zafeiriou. Arcface: Additive angular margin loss for deep face recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[10] J. Deng, J. Guo, X. Niannan, and S. Zafeiriou. Arcface: Additive angular margin loss for deep face recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 4690–4699, 2019.

[11] E. L. Denton, W. Zaremba, J. Bruna, Y. LeCun, and R. Fergus. Exploiting linear structure within convolutional networks for efficient evaluation. In *Advances in Neural Information Processing Systems*, pages 1269–1277, 2014.

[12] Y. Gong, L. Liu, M. Yang, and L. Bourdev. Compressing deep convolutional networks using vector quantization. *arXiv preprint arXiv:1412.6115*, 2014.

[13] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680, 2014.

[14] Y. Guo, J. Chen, Q. Du, A. Van Den Hengel, Q. Shi, and M. Tan. Multi-way backpropagation for training compact deep neural networks. *Neural networks*, 2020.

[15] Y. Guo, Q. Chen, J. Chen, Q. Wu, Q. Shi, and M. Tan. Auto-embedding generative adversarial networks for high resolution image synthesis. *IEEE Transactions on Multimedia*, 2019.

[16] Y. Guo, Q. Wu, C. Deng, J. Chen, and M. Tan. Double forward propagation for memorized batch normalization. In *AAAI Conference on Artificial Intelligence*, 2018.

[17] Y. Guo, A. Yao, and Y. Chen. Dynamic network surgery for efficient dnns. In *Advances in Neural Information Processing Systems*, pages 1379–1387, 2016.

[18] Y. Guo, L. Zhang, Y. Hu, X. He, and J. Gao. Ms-celeb-1m: A dataset and benchmark for large-scale face recognition. In *European Conference on Computer Vision*, pages 87–102, 2016.

[19] S. Han, H. Mao, and W. J. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. In *International Conference on Learning Representations*, 2016.

[20] S. Han, J. Pool, J. Tran, and W. Dally. Learning both weights and connections for efficient neural network. In *Advances in Neural Information Processing Systems*, pages 1135–1143, 2015.

[21] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[22] Y. He, G. Kang, X. Dong, Y. Fu, and Y. Yang. Soft filter pruning for accelerating deep convolutional neural networks. In *International Joint Conferences on Artificial Intelligence*, pages 2234–2240, 2018.

[23] Y. He, J. Lin, Z. Liu, H. Wang, L.-J. Li, and S. Han. Amc: Automl for model compression and acceleration on mobile devices. In *European Conference on Computer Vision*, pages 784–800, 2018.

[24] Y. He, P. Liu, Z. Wang, Z. Hu, and Y. Yang. Filter pruning via geometric median for deep convolutional neural networks acceleration. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[25] Y. He, X. Zhang, and J. Sun. Channel pruning for accelerating very deep neural networks. In *IEEE International Conference on Computer Vision*, pages 1389–1397, 2017.

[26] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[27] H. Hu, R. Peng, Y.-W. Tai, and C.-K. Tang. Network trimming: A data-driven neuron pruning approach towards efficient deep architectures. *arXiv preprint arXiv:1607.03250*, 2016.

[28] G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical report, Technical Report 07-49, University of Massachusetts, Amherst, 2007.

[29] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio. Binarized neural networks. In *Advances in Neural Information Processing Systems*, pages 4107–4115, 2016.

[30] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456, 2015.

[31] M. Jaderberg, A. Vedaldi, and A. Zisserman. Speeding up convolutional neural networks with low rank expansions. In *British Machine Vision Conference*, 2014.

[32] C. Jiang, G. Li, C. Qian, and K. Tang. Efficient dnn neuron pruning by minimizing layer-wise nonlinear reconstruction error. In *International Joint Conferences on Artificial Intelligence*, volume 2018, pages 2–2, 2018.

[33] S. Jung, C. Son, S. Lee, J. Son, J.-J. Han, Y. Kwak, S. J. Hwang, and C. Choi. Learning to quantize deep networks by optimizing quantization intervals with task loss. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 4350–4359, 2019.

[34] I. Kemelmacher-Shlizerman, S. M. Seitz, D. Miller, and E. Brossard. The megaface benchmark: 1 million faces for recognition at scale. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 4873–4882, 2016.

[35] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. *Tech Report*, 2009.

[36] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012.

[37] C.-Y. Lee, S. Xie, P. Gallagher, Z. Zhang, and Z. Tu. Deeply-supervised nets. In *International Conference on Artificial Intelligence and Statistics*, pages 562–570, 2015.

[38] N. Lee, T. Ajanthan, and P. H. Torr. Snip: Single-shot network pruning based on connection sensitivity. In *International Conference on Learning Representations*, 2019.

[39] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf. Pruning filters for efficient convnets. In *International Conference on Learning Representations*, 2017.

[40] S. Lin, R. Ji, Y. Li, C. Deng, and X. Li. Toward compact convnets via structure-sparsity regularized filter pruning. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–15, 2019.

[41] S. Lin, R. Ji, Y. Li, Y. Wu, F. Huang, and B. Zhang. Accelerating convolutional networks via global & dynamic filter pruning. In *International Joint Conferences on Artificial Intelligence*, pages 2425–2432, 2018.

[42] S. Lin, R. Ji, C. Yan, B. Zhang, L. Cao, Q. Ye, F. Huang, and D. Doermann. Towards optimal structured cnn pruning via generative adversarial learning. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2790–2799, 2019.

[43] J. Liu, J. Ye, and R. Fujimaki. Forward-backward greedy algorithms for general convex smooth functions over a cardinality constraint. In *International Conference on Machine Learning*, pages 503–511, 2014.

[44] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In *European Conference on Computer Vision*, pages 21–37, 2016.

[45] W. Liu, Y. Wen, Z. Yu, M. Li, B. Raj, and L. Song. Sphereface: Deep hypersphere embedding for face recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 212–220, 2017.

[46] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, and C. Zhang. Learning efficient convolutional networks through network slimming. In *IEEE International Conference on Computer Vision*, pages 2736–2744, 2017.

[47] Z. Liu, H. Mu, X. Zhang, Z. Guo, X. Yang, T. K.-T. Cheng, and J. Sun. Metapruning: Meta learning for automatic neural network channel pruning. In *IEEE International Conference on Computer Vision*, 2019.

[48] Z. Liu, M. Sun, T. Zhou, G. Huang, and T. Darrell. Rethinking the value of network pruning. In *International Conference on Learning Representations*, 2019.

[49] I. Loshchilov and F. Hutter. SGDR: stochastic gradient descent with warm restarts. In *International Conference on Learning Representations*, 2017.

[50] J.-H. Luo, J. Wu, and W. Lin. Thinet: A filter level pruning method for deep neural network compression. In *IEEE International Conference on Computer Vision*, pages 5058–5066, 2017.

[51] J.-H. Luo, H. Zhang, H.-Y. Zhou, C.-W. Xie, J. Wu, and W. Lin. Thinet: pruning cnn filters for a thinner net. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2018.

[52] P. Molchanov, A. Mallya, S. Tyree, I. Frosio, and J. Kautz. Importance estimation for neural network pruning. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[53] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz. Pruning convolutional neural networks for resource efficient inference. In *International Conference on Learning Representations*, 2016.

[54] S. Moschoglou, A. Papaioannou, C. Sagonas, J. Deng, I. Kotsia, and S. Zafeiriou. Agedb: the first manually collected, in-the-wild age database. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 51–59, 2017.

[55] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *International Conference on Machine Learning*, pages 807–814, 2010.

[56] Y. E. Nesterov. A method for solving the convex programming problem with convergence rate o (1/kˆ 2). In *Proceedings of the USSR Academy of Sciences*, volume 269, pages 543–547, 1983.

[57] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pages 8024–8035, 2019.

[58] Y. Rao, J. Lu, J. Lin, and J. Zhou. Runtime network routing for efficient image classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2018.

[59] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European Conference on Computer Vision*, pages 525–542, 2016.

[60] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 779–788, 2016.

[61] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1137–1149, 2016.

[62] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 4510–4520, 2018.

[63] F. Schroff, D. Kalenichenko, and J. Philbin. Facenet: A unified embedding for face recognition and clustering. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 815–823, 2015.

[64] S. Sengupta, J.-C. Chen, C. Castillo, V. M. Patel, R. Chellappa, and D. W. Jacobs. Frontal to profile face verification in the wild. In *Winter Conference on Applications of Computer Vision*, pages 1–9. IEEE, 2016.

[65] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *Advances in Neural Information Processing Systems*, pages 568–576, 2014.

[66] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

[67] V. Sindhwani, T. Sainath, and S. Kumar. Structured transforms for small-footprint deep learning. In *Advances in Neural Information Processing Systems*, pages 3088–3096, 2015.

[68] S. Srinivas, A. Subramanya, and R. V. Babu. Training sparse neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 455–462, 2017.

[69] R. K. Srivastava, K. Greff, and J. Schmidhuber. Training very deep networks. In *Advances in Neural Information Processing Systems*, pages 2377–2385, 2015.

[70] Y. Sun, D. Liang, X. Wang, and X. Tang. Deepid3: Face recognition with very deep neural networks. *arXiv preprint arXiv:1502.00873*, 2015.

[71] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.

[72] C. Tai, T. Xiao, X. Wang, and W. E. Convolutional neural networks with low-rank regularization. In *International Conference on Learning Representations*, 2016.

[73] M. Tan and Q. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*, pages 6105–6114, 2019.

[74] M. Tan, I. W. Tsang, and L. Wang. Towards ultrahigh dimensional feature selection for big data. *Journal of Machine Learning Research*, 15(1):1371–1429, 2014.

[75] M. Tan, I. W. Tsang, and L. Wang. Matching pursuit lasso part i: Sparse recovery over big dictionary. *IEEE Transactions on Signal Processing*, 63(3):727–741, 2015.

[76] H. Wang, Y. Wang, Z. Zhou, X. Ji, D. Gong, J. Zhou, Z. Li, and W. Liu. Cosface: Large margin cosine loss for deep face recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 5265–5274, 2018.

[77] H. Wang, Q. Zhang, Y. Wang, and H. Hu. Structured probabilistic pruning for convolutional neural network acceleration. In *British Machine Vision Conference*, 2018.

[78] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. Van Gool. Temporal segment networks: Towards good practices for deep action recognition. In *European Conference on Computer Vision*, pages 20–36, 2016.

[79] W. Wen, C. Wu, Y. Wang, Y. Chen, and H. Li. Learning structured sparsity in deep neural networks. In *Advances in Neural Information Processing Systems*, pages 2074–2082, 2016.

[80] T.-J. Yang, A. Howard, B. Chen, X. Zhang, A. Go, M. Sandler, V. Sze, and H. Adam. Netadapt: Platform-aware neural network adaptation for mobile applications. In *European Conference on Computer Vision*, pages 285–300, 2018.

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

[82] Z. You, K. Yan, J. Ye, M. Ma, and P. Wang. Gate decorator: Global filter pruning method for accelerating deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, 2019.

[83] R. Yu, A. Li, C.-F. Chen, J.-H. Lai, V. I. Morariu, X. Han, M. Gao, C.-Y. Lin, and L. S. Davis. Nisp: Pruning networks using neuron importance score propagation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 9194–9203, 2018.

[84] X. Yuan, P. Li, and T. Zhang. Gradient hard thresholding pursuit for sparsity-constrained optimization. In *International Conference on Machine Learning*, pages 127–135, 2014.

[85] R. Zeng, C. Gan, P. Chen, W. Huang, Q. Wu, and M. Tan. Breaking winner-takes-all: Iterative-winners-out networks for weakly supervised temporal action localization. *IEEE Transactions on Image Processing*, 28(12):5797–5808, 2019.

[86] X. Zhang, J. Zou, K. He, and J. Sun. Accelerating very deep convolutional networks for classification and detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(10):1943–1955, 2016.

[87] A. Zhou, A. Yao, Y. Guo, L. Xu, and Y. Chen. Incremental network quantization: Towards lossless cnns with low-precision weights. In *International Conference on Learning Representations*, 2017.

[88] S. Zhou, Y. Wu, Z. Ni, X. Zhou, H. Wen, and Y. Zou. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *arXiv preprint arXiv:1606.06160*, 2016.

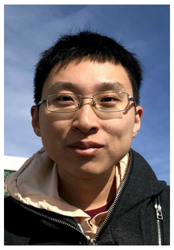
[89] X. Zhu, W. Zhou, and H. Li. Improving deep neural network sparsity through decorrelation regularization. In *International Joint Conferences on Artificial Intelligence*, pages 3264–3270, 2018.

[90] B. Zhuang, C. Shen, M. Tan, L. Liu, and I. Reid. Towards effective low-bitwidth convolutional neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 7920–7928, 2018.

[91] B. Zhuang, C. Shen, M. Tan, L. Liu, and I. Reid. Structured binary neural networks for accurate image classification and semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 413–422, 2019.

[92] Z. Zhuang, M. Tan, B. Zhuang, J. Liu, Y. Guo, Q. Wu, J. Huang, and J. Zhu. Discrimination-aware channel pruning for deep neural networks. In *Advances in Neural Information Processing Systems*, pages 881–892. 2018.

Jing Liu is a Ph.D. student in the Faculty of Information Technology, Monash University Clayton Campus, Australia. He received his Bachelor Degree in 2017 and Master Degree in 2020, both from the School of Software Engineering at South China University of Technology, China. His research interests include computer vision, model compression and acceleration.



Bohan Zhuang is an assistant professor with the Faculty of Information Technology, Monash University Clayton Campus, Australia. He received his Bachelor degree in Electronic and Information Engineering in 2014 from Dalian University of Technology, China. He has been with the School of Computer Science, University of Adelaide, Australia, where he received the Ph.D. degree in 2018 and worked as a Senior Research Associate from 2018-2020. His research interests include computer vision and machine learning.

Zhuangwei Zhuang is currently working as an Algorithm Engineer in RoboSense, Shenzhen, China. He received his Bachelor Degree in Automation and Engineering in 2016 and Master Degree in Software Engineering in 2018, both from South China University of Technology in Guangzhou, China. His research interests include computer vision and model compression.

Yong Guo is a Ph.D. Student in the School of Software Engineering at South China University of Technology. He also received his bachelor degree in Software Engineering from the same university in 2016. His research interests include deep learning and computer vision.

Junzhou Huang is an Associate Professor in the Computer Science and Engineering department at the University of Texas at Arlington. He received the B.E. degree from Huazhong University of Science and Technology, China, the M.S. degree from Chinese Academy of Sciences, China, and the Ph.D. degree in Rutgers university. His major research interests include machine learning, computer vision and imaging informatics. He was selected as one of the 10 emerging leaders in multimedia and signal processing by the IBM T.J. Watson Research Center in 2010. He received the NSF CAREER Award in 2016.

Jinhui Zhu received the B.S. degree in automatic control, the M.S. degree in mechanical design and theory, and the Ph.D. degree in computer science from the South China University of Technology, Guangzhou, China, in 1999, 2002, and 2010, respectively. He has been with the School of Computer Science and Engineering, South China University of Technology, since 2002, where he became an Associate Professor in 2011 and joined the School of Software Engineering, in 2012. His current research interests include robot software architecture, embedded operating system, machine learning, and their applications.

Mingkui Tan is currently a professor with the School of Software Engineering at South China University of Technology. He received his Bachelor Degree in Environmental Science and Engineering in 2006 and Master degree in Control Science and Engineering in 2009, both from Hunan University in Changsha, China. He received the Ph.D. degree in Computer Science from Nanyang Technological University, Singapore, in 2014. From 2014-2016, he worked as a Senior Research Associate on computer vision in the School of Computer Science, University of Adelaide, Australia. His research interests include machine learning, sparse analysis, deep learning and large-scale optimization.