Towards Compact CNNs via Collaborative Compression

Yuchao Li[1,2*], Shaohui Lin[3*], Jianzhuang Liu[4], Qixiang Ye[5], Mengdi Wang[2]
Fei Chao[1], Fan Yang[6], Jincheng Ma[6], Qi Tian[4], Rongrong Ji[1,7,8†]

[1]Media Analytics and Computing Laboratory, Department of Artificial Intelligence,
School of Informatics, Xiamen University, [2]Alibaba Group, [3]East China Normal University
[4]Huawei Noah's Ark Lab, [5]University of Chinese Academy of Sciences, [6]Huawei TechnologiesCo.,Ltd
[7]Institute of Artificial Intelligence, Xiamen University, [8]Peng Cheng Laboratory

{laiyin.lyc, didou.wmd}@alibaba-inc.com, shlin@cs.ecnu.edu.cn, qxye@ucas.ac.cn,
{liu.jianzhuang, yangfan74}@huawei.com, majincheng1@hisilicon.com, {fchao, rrji}@xmu.edu.cn

Abstract

Channel pruning and tensor decomposition have received extensive attention in convolutional neural network compression. However, these two techniques are traditionally deployed in an isolated manner, leading to significant accuracy drop when pursuing high compression rates. In this paper, we propose a Collaborative Compression (CC) scheme, which joints channel pruning and tensor decomposition to compress CNN models by simultaneously learning the model sparsity and low-rankness. Specifically, we first investigate the compression sensitivity of each layer in the network, and then propose a Global Compression Rate Optimization that transforms the decision problem of compression rate into an optimization problem. After that, we propose multi-step heuristic compression to remove redundant compression units step-by-step, which fully considers the effect of the remaining compression space (i.e., unremoved compression units). Our method demonstrates superior performance gains over previous ones on various datasets and backbone architectures. For example, we achieve 52.9% FLOPs reduction by removing 48.4% parameters on ResNet-50 with only a Top-1 accuracy drop of 0.56% on ImageNet 2012.

1. Introduction

Remarkable achievements have been attained by convolutional neural networks (CNNs), such as object classification [16, 36, 7], detection [34, 33] and segmentation [1]. However, the explosive growth of parameters and computational cost in CNN models has restricted their deployment on resource-limited devices, such as mobile or wearable devices. To this end, extensive efforts have been made for CNN compression and acceleration, including but not limited to, parameter pruning [40, 28, 19], tensor decomposition [17, 22, 41] and quantization [13, 45].

Parameter pruning and tensor decomposition are two widespread directions in CNN compression, which both aim to remove intrinsic redundancy in parameters with different removing strategies. Parameter pruning removes correlated weight connections [5, 6] or structured neurons [28, 10, 24] based on importance measurement methods, resulting in sparse weight structures. In contrast, tensor decomposition approximates weights of low-rank filters based on the intrinsic low-rankness of parameters [44, 41, 22, 14]. It is thus a natural thought to combine these two compression strategies, which might lead to the significant accuracy drop when pursuing high compression rate. For instance, Dubey *et al.* [4] proposed to compress the weights by sequentially employing pruning and tensor decomposition, which assumes that they are complementary without any mutual influence. However, as demonstrated in previous work [43], although the pruning and decomposition explore different redundancy in parameters, they are not completely orthogonal. Thus, the above method [4] does not exploit the complementary nature of pruning and decomposition, which is sub-optimal as exploring only within each sub-task (*i.e.,* each compression method), not from the global compression scope.

To leverage the benefits of both compression operations, training-aware methods [43, 29, 20] use two regularizations to separately handle the sparsity on channel pruning

[*]Equal contribution.
[†]Corresponding author.

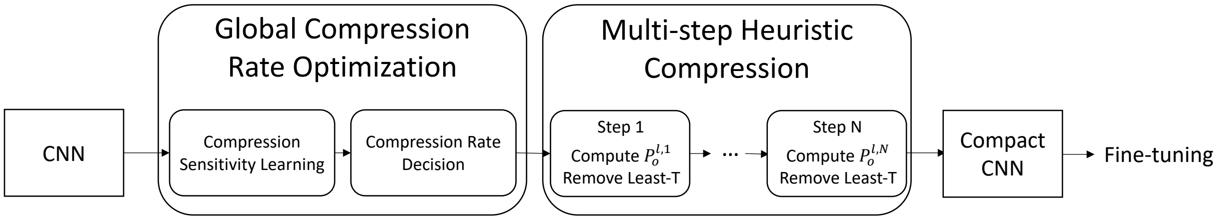

CNN
Global Compression Rate Optimization
Compression Sensitivity Learning
Compression Rate Decision
Multi-step Heuristic Compression
Step 1
Compute $P_o^{l,1}$
Remove Least-T
···
Step N
Compute $P_o^{l,N}$
Remove Least-T
Compact CNN
Fine-tuning

Figure 1: The framework of our method. We first compute the group values between information loss and compression rate in each layer and then obtain the compression rate of each layer by solving a global compression rate optimization problem based on them. After that, we compress each layer independently by removing the less important compression units step-by-step based on the removed units and the exploration of remaining compression space.

and the low-rankness on tensor decomposition, which are jointly minimized the regularization loss during training. They show that simultaneously handle sparsity and low-rankness in weights can explore the richer structure information of parameters. However, these methods are difficult to control the compression rate, which needs to adjust hyper-parameters by trial-and-error to achieve the trade-off between compression rate and model accuracy.

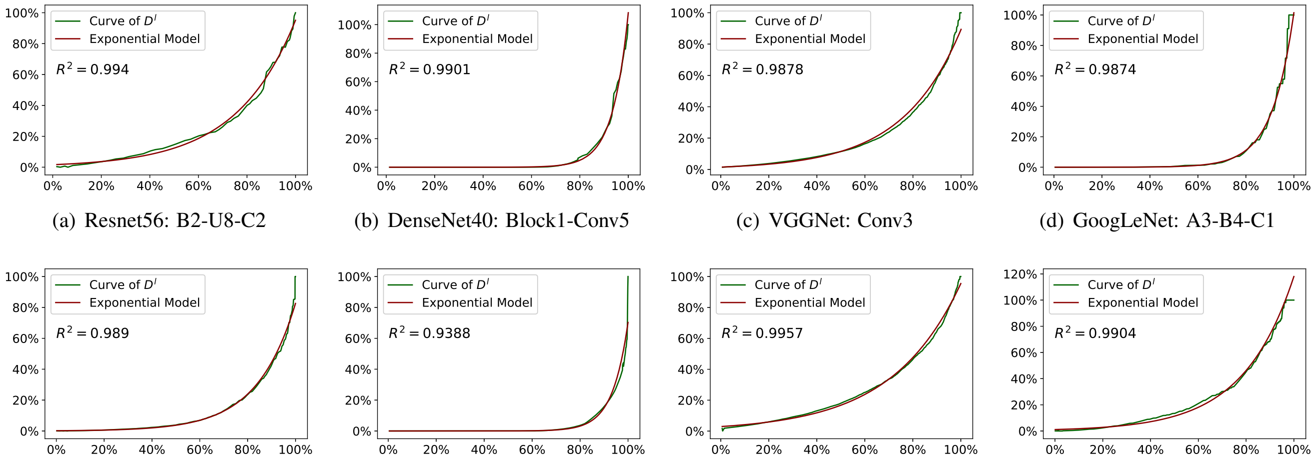
In this paper, we propose a novel unified compression framework, named *collaborative compression* (CC), to simultaneously deal with the sparsity and low-rankness in weights, with an essential innovation in automatic compression rate control. Our method is a post-training compression algorithm that simultaneously explores the sparsity and low-rankness in pre-trained networks, which is easier to apply than the training-aware methods. As shown in Fig. 1, towards a fast and practical compression, we first determine the compression rate of each layer by *global compression rate optimization* and then compress each layer independently by *multi-step heuristic compression*. It avoids determining the compression rate and the compression strategy (*i.e.,* which compression units should be removed) simultaneously. In particular, the global compression rate optimization analyzes the compression sensitivity of each layer by constructing the relationship between the proposed information loss and the compression rate. We find that this relationship can be fitted well by an exponential function, and the compression sensitivity can be viewed as the first-order derivative of this exponential function. Based on the exponential model, we construct an optimization process to search the best compression rate of each layer. After determining the compression rate, we compress each layer synchronously and independently. Considering that removing units will affect the importance of the remaining compression units, we propose a multi-step heuristic compression method, which removes less important units step-by-step. At each step, the importance calculation of each unit fully considers the effect of removed compression units and the remaining compression space.

In experiments, we demonstrate the effectiveness of the proposed CC framework using four widely-used networks (VGGNet, GoogleNet, ResNet and DenseNet) on two datasets (CIFAR-10 and ImageNet 2012). Compared to the state-of-the-art methods, CC achieves superior performance. For example, we obtain 52.9% FLOPs reduction by removing 48.4% parameters, with only a Top-1 accuracy drop of 0.56% on ResNet-50. Meanwhile, the compressed MobileNet-V2 obtained by our method achieves performance gains over the state-of-the-art pruning methods based on AutoML.

2. Related Work

Pruning can be categorized into either unstructured or structured pruning. Unstructured pruning [5, 6] aims to remove unimportant weights independently, while structured pruning removes structured parts (*e.g.,* filters, channels or layers) that are well supported by various off-the-shelf deep learning libraries. Structured pruning, especially filter pruning, removes redundant filters by different importance measurements, such as ℓ_1-norm on filters [19, 8], sparsity of output feature maps [11], sparsity regularization [40, 2, 23] and loss drop *w.r.t.* filter removal [30, 28]. Similar to filter pruning, channel pruning [26, 12] targets at removing input channels, which can avoid the dimensional mismatch in various multi-branch networks, *e.g.,* ResNets and DenseNets. Apart from this, layer pruning has been proposed for data-dependent inference [38, 39] or static block removal [25].

Tensor decomposition [41, 18, 22, 14] aims to reduce the memory and computation cost by decomposing the convolutional filters into a sequence of tensors with fewer parameters. Unlike pruning, it explores the low-rank structure of the original weights, which keeps the dimension of convolutional outputs unchanged. Early works directly introduce different tensor decomposition methods on the original weights, such as SVD-decomposition [44], Tucker-decomposition [15] and CP-decomposition [17]. Later on, Lin *et al.* [22] proposed a novel closed-form low-rank decomposition for fast decomposition. Different from direct decomposition, Wen *et al.* [41] proposed a force regularization to coordinate and learn more weight information into a low-rank space. Compared to these single compres-



(a) Resnet56: B2-U8-C2 (b) DenseNet40: Block1-Conv5 (c) VGGNet: Conv3 (d) GoogLeNet: A3-B4-C1

(e) Resnet56:B3-U8-C2 (f) DenseNet40:Block2-Conv5 (g) VGGNet:Conv4 (h) GoogLeNet:A4-B4-C1

Figure 2: Compression sensitivity analysis of different convolutional layers in different networks on CIFAR-10. For each subfigure, the x-axis represents the compression rate and the y-axis is the information loss based on Eq. 6, both of which are normalized to $[0, 1]$. R^2 represents the coefficient of determination between $\mathcal{D}^l$ and the learned exponential model, which measures the result of model fitting. R^2 closer to 1 is better.

sion operation-based methods (*i.e., only using one compression operation*), we propose a collaborative compression method, which focuses on simultaneously reducing the sparsity and low-rankness in the weights and fully considers the mutual influence between them.

To combine multiple compression operations, previous studies [5, 4] proposed multiple compression stages, which independently adopt one compression operation at each step and ignore the mutual influence between different compression operations. For instance, Dubey *et al.* [4] first adopted filter pruning to compress the weights and then decompose them based on a Coreset-based decomposition method. Moreover, some training-aware compression methods [43, 37, 20] compress networks during training by using regularization. Li *et al.* [20] first introduced a sparsity-inducing matrix followed at each weight and then added group sparsity constraints on them during training. However, these training-aware methods are difficult to search a good trade-off between compression rate and accuracy under the target compression rate. In contrast, our collaborative compression employs the global compression rate optimization method to easily obtain the compression rate of each layer under the target compression rate.

3. Proposed Method

3.1. Preliminaries

Generally, the l-th convolutional layer in CNNs transforms an input tensor $\mathcal{X}^l \in \mathbb{R}^{c^l\times h^l_{in}\times w^l_{in}}$ into an output tensor $\mathcal{Y}^l \in \mathbb{R}^{n^l\times h^l_{out}\times w^l_{out}}$ by using a weight tensor $\mathcal{W}^l \in \mathbb{R}^{n^l\times c^l\times k^l\times k^l}$, where c^l and n^l denote the number of input channels and filters (output channels), respectively, and $k^l \times k^l$ is the spatial size of the filters. The convolution operation is represented as:

$$\mathcal{Y}^l = \mathcal{W}^l \otimes \mathcal{X}^l, \tag{1}$$

where $\otimes$ denotes the convolution operation. The biases are omitted for simplicity. Both channel pruning and tensor decomposition seek a compact approximated representation $\overline{\mathcal{W}}^l \in \mathbb{R}^{n^l\times c^l\times k^l\times k^l}$ to replace $\mathcal{W}^l$. Thus, the compression process is regarded as a function $\overline{\mathcal{W}}^l = f(\mathcal{W}^l, o)$, where o represents a single unit index, whose corresponding unit should be removed.

Channel Pruning. We introduce input channel pruning, where each input channel is regarded as a compression unit. Therefore, the compression function is defined as:

$$\overline{\mathcal{W}}^l_{:,i,:,:} = \begin{cases} 0, & i = o, \\ \mathcal{W}^l_{:,i,:,:}, & i \neq o. \end{cases} \tag{2}$$

Tensor Decomposition. We decompose the network's weights based on the Singular Value Decomposition (SVD), where the original $\mathcal{W}^l$ is first mapped from a tensor to a matrix $M^l \in \mathbb{R}^{n^l\times(c^lk^lk^l)}$. After that, let $M^l = U^l\Sigma^l V^{l\top}$ be the SVD of M^l, where $U^l \in \mathbb{R}^{n^l\times n^l}$ and $V^l \in \mathbb{R}^{c^lk^lk^l\times c^lk^lk^l}$ are two orthogonal matrices. The diagonal elements in $\Sigma^l \in \mathbb{R}^{n^l\times(c^lk^lk^l)}$ are singular values of M^l, whose number is $r^l = min(n^l, c^lk^lk^l)$. Finally, the weight matrix M^l is decomposed into two light matrices $M^l_1 = \sqrt{\Sigma^l}V^{l\top}$ and $M^l_2 = U^l\sqrt{\Sigma^l}$ when the number of non-zero singular values $\bar{r}^l$ is much smaller than r^l. For implementation, M^l_1 and M^l_2 are reshaped back into $\mathcal{W}^l_1 \in \mathbb{R}^{\bar{r}^l\times c^l\times k^l\times k^l}$ and $\mathcal{W}^l_2 \in \mathbb{R}^{n^l\times\bar{r}^l\times 1\times 1}$, respectively. Thus, two light convolutions are computed to approximate the original convolution by $\mathcal{W}_2^l \otimes \mathcal{W}_1^l \otimes \mathcal{X}^l$. The compression rate is controlled by the number of non-zero singular values $\overline{r}^l$ in Σ^l. Therefore, the compression unit index is the index of the singular values, and the compression function is defined as:

$$f(\mathcal{W}^l, o) = \overline{\mathcal{W}}^l = \phi(U^l \overline{\Sigma}^l V^{l^\top}),\ \overline{\Sigma}^l_{i,i} = \begin{cases} 0, & i = o, \\ \Sigma^l_{i,i}, & i \neq o, \end{cases} \tag{3}$$

where $\phi : \mathbb{R}^{n^l \times (c^l k^l k^l)} \mapsto \mathbb{R}^{n^l \times c^l \times k^l \times k^l}$ is a mapping function (*a.k.a.* reshaping operator).

In summary, W^l has $c^l + r^l$ candidate compression units. We can obtain compressed weights $\widetilde{\mathcal{W}}^l$ if removing t_1 input channels and t_2 singular values:

$$\widetilde{\mathcal{W}}^l = \begin{cases} \begin{matrix} \widetilde{\mathcal{W}}_1^l \in \mathcal{R}^{(r^l - t_2) \times (c^l - t_1) \times k^l \times k^l}, \\ \widetilde{\mathcal{W}}_2^l \in \mathcal{R}^{n^l \times (r^l - t_2) \times 1 \times 1}, \end{matrix} & t_2 \neq 0 \\ \widetilde{\mathcal{W}}^l \in \mathcal{R}^{n^l \times (c^l - t_1) \times k^l \times k^l}, & t_2 = 0, \end{cases} \tag{4}$$

where $t_2 = 0$ represents that we only adopt channel pruning to compress the convolutional weights. In addition, the compression rate at the l-th layer is defined as:

$$R^l = \begin{cases} 1 - \dfrac{(r^l - t_2) * [(c^l - t_1) * k^l * k^l + n^l]}{n^l * c^l * k^l * k^l}, & t_2 \neq 0, \\ \dfrac{t_1}{c^l}, & t_2 = 0. \end{cases} \tag{5}$$

3.2. Global Compression Rate Optimization

As presented in [19, 23], since the compression sensitivity of each layer is different, the compression rate of the sensitive layers should be set to a small value while the compression rate of the insusceptible layers should be set to a large one. To obtain the compression sensitivity of the layers, the method [19] first removes a certain ratio of less important compression units in the weights and then evaluates the information loss. Such process needs to be evaluated iteratively with different compression rates to obtain the curve between compression rate and information loss, which is used to visualize the compression sensitivity. It inevitably makes the method labor-intensive and costs plenty of time with human analysis to determine the best compression rate. In this paper, we first fast evaluate the compression sensitivity of each layer and then search the best compression rate by solving a simple optimization problem.

Firstly, the information loss at the l-th layer is defined to indicate how much the network loss increases when removing the compression units in this layer. Inspired by [31, 30], we measure the information loss at the l-th layer by adopting the first-order Taylor-based approximation of the network loss on the compressed weights $\mathcal{W}^l$:

$$I^l = [L(\overline{\mathcal{W}}^l) - L(\mathcal{W}^l)]^2 \approx \sum_{i \in Q^l} (\frac{\partial \mathcal{L}}{\partial \mathcal{W}_i^l} * (\overline{\mathcal{W}}_i^l - \mathcal{W}_i^l))^2 = \mathrm{S}[(\mathcal{G}^l * (\overline{\mathcal{W}}^l - \mathcal{W}^l))^2], \tag{6}$$

where $\mathcal{G}^l \in \mathbb{R}^{n^l \times c^l \times k^l \times k^l}$ denotes the gradient computed by the average gradient of the loss *w.r.t.* the pre-trained weights W^l over the entire dataset, Q^l is all element indices in $\mathcal{W}^l$, and $\mathrm{S}[\cdot]$ denotes the sum of all elements in the tensor.

Algorithm 1 Compression sensitivity learning algorithm

```
Require: The pre-trained weights $\mathcal{W}^l$ at the $l$-th layer and the corresponding average gradients $\mathcal{G}^l$.
Ensure: An exponential model $I^l = a^l e^{b^l R^l}$.
Initialize the set of compression unit indices $\mathcal{U}^l$, whose corresponding unit number is $c^l + r^l$.
for each compression unit index $o$ in $\mathcal{U}^l$ do
    $I_o^l = ||\mathcal{G}^l * (f(\mathcal{W}^l, o) - \mathcal{W}^l)||_2^2$.
end for
Sort $\mathcal{U}^l$ based on $I^l$ ascendingly.
Initialize $\overline{\mathcal{W}}^l = \mathcal{W}^l$, $\mathcal{D}^l = \emptyset$.
for each compression unit index $o$ in the sorted $\mathcal{U}^l$ do
    $\overline{\mathcal{W}}^l = f(\overline{\mathcal{W}}^l, o)$.
    $I^l = \frac{||\mathcal{G}^l * (\overline{\mathcal{W}}^l - \mathcal{W}^l)||_2^2}{||\mathcal{G}^l * \mathcal{W}^l||_2^2}$.
    Compute $R^l$ via Eq. 5.
    Add $(R^l, I^l)$ into $\mathcal{D}^l$.
end for
Using the least squares method to fit $\mathcal{D}^l$ by the exponential model $I^l = a^l e^{b^l R^l}$.
```

Algorithm 2 Compression rate decision algorithm

```
Require: The evaluation model of each layer $\{I = a^1 e^{b^1 R}, ..., I = a^L e^{b^L R}\}$, the FLOPs of each layer $\{F^1, ..., F^L\}$, the FLOPs of the entire network $F$, the target compression rate $C$, and learning rate $\eta$.
Ensure: The target compression rate of each layer $\{R^1, ..., R^L\}$.
Initialize $\overline{I}' = 0.1$.
while $(\sum_{i=1}^{L} \frac{F^i}{b^i} log(\frac{\overline{y}'}{a^i b^i}) - CF)^2 > 10^4$ do
    $g = 2(\sum_{i=1}^{L} \frac{F^i}{b^i} log(\frac{\overline{I}'}{a^i b^i}) - CF)(\sum_{i=1}^{L} \frac{F^i}{b^i \overline{I}'})$.
    $\overline{I}' = \overline{I}' - \eta g$.
end while
return $\{R^i = \frac{1}{b^i} log(\frac{\overline{I}'}{a^i b^i})\}_{i=1}^{L}$.
```

After that, we propose a greedy algorithm shown in Alg. 1 to fast evaluate the information loss under different compression rates, which is significantly fast to measure compression sensitivity of each layer. We first compute the importance of each compression unit using Eq. 6. Then, to obtain the information loss under different compression rates (*i.e.,* a point set $\mathcal{D}^l$), we remove candidate compression units, based on their importance progressively, and compute the corresponding information loss I^l and compression rate R^l after each unit removed. As shown in Fig. 2, the green lines are from the point set $\mathcal{D}^l$ where R^l and I^l are normalized to $[0, 1]$. The results show that the compression sensitivity of different layers are possibly variable, such as Fig. 2(a) vs. Fig. 2(e) in ResNet-56. Through calculating all the layers across various networks, we empirically find that the curve of $\mathcal{D}^l$ can be estimated by an exponential function, $I^l = a^l e^{b^l R^l}$. Therefore, we fit the function to learn the parameters a^l and b^l by using the least-squares method. Also in Fig. 2, we find that the curve of $\mathcal{D}^l$ is well approximated by our exponential function with extremely small reconstruction error. More results are shown in the supplementary materials.

The compression sensitivity at the l-th layer can be defined by the gradient of the exponential function *w.r.t* the compression rate R^l as $\frac{\partial I^l}{\partial R^l} = a^l b^l e^{b^l R^l}$. We construct the following optimization problem to decide the compression rate of each layer when given the whole network compression rate C:

$$\begin{aligned} &\min_{\mathcal{R}} (\sum_{i=1}^{L} F^i \cdot R^i - C \cdot F)^2, \\ &s.t.\ \ a^i b^i e^{b^i R^i} = a^j b^j e^{b^j R^j},\ \forall i, j \in \{1, 2..., L\}, \end{aligned} \tag{7}$$

where $\mathcal{R} = \{R^1, R^2, ..., R^L\}$ denotes the set of compression rates for the layers, F^l and F represents the FLOPs of the l-th layer and the entire network, respectively. Under the same compression rate, the compression sensitivity $\frac{\partial I}{\partial R}$ of the sensitive layers is higher than that of the insusceptible layers. Correspondingly, the sensitive layers have a smaller compression rate than the insusceptible layers under the same compression sensitivity. Therefore, in the constraint term, the compression sensitivity of all layers are set to the same to make the compression rate of the sensitive layers smaller than that of the insusceptible layers. To simplify the optimization, we formulate the following equation as:

$$\overline{I}' = \frac{\partial I^l}{\partial R^l} = a^l b^l e^{b^l R^l} \Rightarrow R^l = \frac{1}{b^l} log(\frac{\overline{I}'}{a^l b^l}). \tag{8}$$

Therefore, we can transform the optimized variables from $\mathcal{R}$ to $\overline{I}'$ according to the constraint term in Eq. 7. Thus, Eq. 7 is rewritten as:

$$\min_{\overline{I}'} (\sum_{i=1}^{L} \frac{F^i}{b^i} log(\frac{\overline{I}'}{a^i b^i}) - C \cdot F)^2. \tag{9}$$

This optimization problem can be directly solved by a gradient descent algorithm. The detailed optimization process is presented in Alg. 2. After obtaining the optimal sensitivity (*i.e.,* final result of $\overline{I}'$), we can also obtain the compression rate of each layer via Eq. 8.

3.3. Multi-Step Heuristic Compression

After determining the compression rate, we propose a heuristic method to compress each layer's weights. As discussed above, the importance of units will change due to the removal of other units because of the mutual influence between different compression operations. Therefore, the previous importance metric [9, 21, 30] is no longer effective by only taking fixed weights into consideration and only computing the importance metric once. Inspired by the value function in the Markov Decision Process, we propose a novel importance metric, in which the importance of the o-th compression unit in the l-th layer at the t-th step can be computed as:

$$P_o^{l,(t)} = I_o^{l,(t)} + \gamma \frac{1}{|\mathcal{U}^{l,(t)}| - 1} \sum_{i \in \mathcal{U}^{l,(t)} \backslash o} I_{i|o}^{l,(t)}, \tag{10}$$

where $\mathcal{U}^{l,(t)}$ represents the remaining compression units (*i.e.,* a set of not removed compression units) after $t - 1$ steps and γ is a hyper-parameter to trade-off the influence of removed units and remaining compression space. Besides,

$$I_o^{l,(t)} = \mathrm{S}[(\mathcal{G}^l * (\overline{\mathcal{W}}_o^{l,(t)} - \mathcal{W}^l))^2]\ ,\ \overline{\mathcal{W}}_o^{l,(t)} = f(\overline{\mathcal{W}}^{l,(t-1)}, o), \tag{11}$$

and

$$I_{i|o}^{l,(t)} = \mathrm{S}[(\mathcal{G}^l * (\overline{\mathcal{W}}_{i|o}^{l,(t)} - \mathcal{W}^l))^2],\ \overline{\mathcal{W}}_{i|o}^{l,(t)} = f(\overline{\mathcal{W}}_o^{l,(t)}, i). \tag{12}$$

The first item in Eq. 10 represents the importance of the o-th compression unit based on the compressed weight $\overline{\mathcal{W}}^{l,(t-1)}$ at the $(t - 1)$-th step. The second item represents the potential information loss for searching the remaining compression space, which is an average information loss of the weights generated by removing the remaining compression units of $\overline{\mathcal{W}}_o^{l,(t)}$. Based on proposed important metric, we compute the importance of each compression unit and remove the least important one step-by-step. The compression process stops when the compression rate of $\overline{\mathcal{W}}^{l,(t)}$ is larger than the target compression rate at the l-th layer. After finishing the compression of all layers parallelly, the approximated weight $\overline{\mathcal{W}}^{l,(t)}$ will be transformed to $\widetilde{\mathcal{W}}^l$ via Eq. 4. Finally, fine-tuning is used to improve the accuracy of the compressed network.

However, the time complexity of the above computation process is $O((c^l + r^l)^3)$[1], which significantly affects the efficiency of offline compression. Instead, we propose to remove the T compression units after measuring the importance metric each time to accelerate the compression process, where T is a balance hyper-parameter. As such, the

[1] We omit the time complexity of SVD and matrix multiplication to simplify the analysis.

Model	Method	FLOPs(PR)	#Param.(PR)	Top-1 Acc%
ResNet-56	Baseline	125M	0.85M	93.33
	L1[19]	112M(10.4%)	0.77M(9.4%)	93.10
	HRank[21]	89M(28.8%)	0.71M(16.5%)	93.52
	Nisp[42]	81M(35.2%)	0.49M(42.4%)	93.01
	GAL[25]	78M(37.6%)	0.75M(11.8%)	92.98
	CC ($C = 0.4$)	**72M(42.4%)**	**0.54M(36.5%)**	**93.87**
	ENC[14]	63M(49.6%)	-	93.09
	CP[10]	62M(50.4%)	-	91.80
	CC ($C = 0.5$)	**60M(52.0%)**	**0.44M(48.2%)**	**93.64**
	FPGM[9]	59M(52.8%)	-	93.49
	C-SGD[2]	49M(60.8%)	-	93.44
DenseNet-40	Baseline	283M	1.04M	94.81
	GAL[25]	183M(35.3%)	0.67M(35.6%)	94.61
	HRank[21]	167M(41.0%)	0.66M(36.5%)	94.24
	CC ($C = 0.4$)	**150M(47.0%)**	**0.50M(51.9%)**	**94.67**
	slimming[26]	120M(57.6%)	0.35M(66.3%)	94.35
	C-SGD[2]	113M(60.1%)	-	94.37
	CC ($C = 0.6$)	**112M(60.4%)**	**0.37M(64.4%)**	**94.40**
	GAL[25]	78M(72.4%)	0.75M(27.9%)	92.98
VGGNet	Baseline	313M	14.72M	93.70
	L1[19]	206M(34.2%)	5.40M(63.3%)	93.60
	GAL[25]	189M(39.6%)	3.36M(77.2%)	93.77
	AOFP[3]	186M(40.6%)	-	94.03
	CC ($C = 0.5$)	**154M(50.8%)**	**5.02M(65.9%)**	**94.15**
	HRank[21]	145M(53.7%)	2.51M(82.9%)	93.42
	CC ($C = 0.6$)	**123M(60.7%)**	**4.02M(72.7%)**	**94.09**
GoogLeNet	Baseline	1.52B	6.15M	95.05
	L1[19]	1.02B(32.9%)	3.51M(42.9%)	94.54
	GAL[25]	0.94B(38.2%)	3.12M(49.3%)	93.93
	CC ($C = 0.5$)	**0.76B(50.0%)**	**2.83M(54.0%)**	**95.18**
	HRank[21]	0.69M(54.6%)	2.74M(55.4%)	94.53
	CC ($C = 0.6$)	**0.61B(59.9%)**	**2.26M(63.3%)**	**94.88**

Table 1: Comparison with single compression operation-based methods on CIFAR-10. In this table and all following tables and figures, M/B means million/billion, and PR denotes pruning rate.

computational complexity for the importance metric $P_o^{l,(t)}$ is $O(c^l + r^l)$, we re-formulate the second item in the importance metric as:

$$\begin{aligned} &\gamma \frac{1}{|\mathcal{U}^{l,(t)}| - 1} \sum_{i \in \mathcal{U}^{l,(t)} \setminus o} I_{i|o}^{l,(t)} \\ &= \gamma \frac{1}{|\mathcal{U}_o^{l,(t)}|} \sum_{i \in \mathcal{U}_o^{l,(t)}} \mathrm{S}[(\mathcal{G}^l * (\overline{\mathcal{W}}_{i|o}^{l,(t)} - \mathcal{W}^l))^2] \\ &= \gamma \frac{1}{|\mathcal{U}_o^{l,(t)}|} \sum_{i \in \mathcal{U}_o^{l,(t)}} \mathrm{S}[(\mathcal{G}^l * (\theta_{i|o}^{l,(t)} + \theta_o^{l,(t)}))^2] \end{aligned} \tag{13}$$

where $\theta_{i|o}^{l,(t)} = \overline{\mathcal{W}}_{i|o}^{l,(t)} - \overline{\mathcal{W}}_o^{l,(t)}$ and $\theta_o^{l,(t)} = \overline{\mathcal{W}}_o^{l,(t)} - \mathcal{W}^l$, $*$ represents element-wise multiplication and $\mathcal{U}_o^{l,t} = \mathcal{U}^{l,t} \setminus o$. The detailed computation process is presented in the supplementary materials. Finally, we can compute the importance metric as:

$$\begin{aligned} P_o^{l,(t)} = &(1 + \gamma)\mathrm{S}[(\mathcal{G}^l * (\overline{\mathcal{W}}_o^{l,(t)} - \mathcal{W}^l))^2] \\ &- \gamma \frac{4}{|\mathcal{U}_o^{l,(t)}|} \mathrm{S}[(\mathcal{G}^l)^2 * \theta_o^{l,(t)} * \overline{\mathcal{W}}_o^{l,(t)}] \\ &+ \gamma \frac{1}{|\mathcal{U}_o^{l,(t)}|} \mathrm{S}[(\mathcal{G}^l * \overline{\mathcal{W}}_o^{l,(t)})^2] \\ &+ \gamma \frac{1}{|\mathcal{U}_o^{l,(t)}|} \mathrm{S}[(\mathcal{G}^l)^2 * \phi((U_o^{l,(t)})^2 (\Sigma_o^{l,(t)})^2 (V_o^{l,(t)^\top})^2)], \end{aligned} \tag{14}$$

where $U_o^{l,(t)}$, $\overline{\Sigma}_o^{l,(t)}$ and $V_o^{l,(t)^\top}$ is the SVD result of $\phi^{-1}(\overline{\mathcal{W}}_o^{l,(t)})$. Thus, the computational complexity for the importance metric is reduced to $O(1)$, which significantly accelerates the calculation of the importance metric. Our heuristic compression algorithm is summarized in Alg. A in the supplementary materials.

4. Experiments

The proposed CC scheme is implemented using Pytorch [32] and evaluated on two datasets, CIFAR-10 and ImageNet ILSVRC 2012. The input images in CIFAR-10 are classified into 10 classes, whose sizes are all 32 × 32. The training set contains 50,000 images and the test set contains 10,000 images. ImageNet ILSVRC 2012 consists of 1.28 million training images and 50,000 validation images for testing over 1,000 classes.

4.1. Experimental Settings

All networks are trained via the stochastic gradient descent (SGD) with momentum 0.9. On CIFAR-10, we train the networks for 300 epochs using the mini-batch size of 128. The initial learning rate is set to 0.01 and is multiplied by 0.1 at 50% and 75% of the total epoch num-

Model	Method	FLOPs (PR)	#Param. (PR)	Top-1 Acc%	Top-5 Acc%
VGG-16	Baseline	15.48B	138M	71.59	90.38
	LRSD[43]	-	9.70M(93.0%)	68.75	-
	Coreset[4]	-	9.81M(92.9%)	68.56	-
	Coreset[4]	-	8.70M(93.7%)	68.16	-
	CC-GAP ($C = 0.5$)	**7.37B(52.4%)**	**8.35M(93.9%)**	**68.81**	**88.70**
ResNet-50	Baseline	4.10B	25.56M	76.15	92.87
	Hinge[20]	1.91B(53.4%)	-	74.70	-
	Rethinking[27]+SVD[44]	1.93B(52.9%)	-	74.75	92.17
	FPGM [9]+SVD[44]	1.93B(52.9%)	-	75.47	92.55
	CC ($C = 0.5$)	**1.93B(52.9%)**	**13.2M(48.4%)**	**75.59**	**92.64**
	CC ($C = 0.6$)	**1.53B(62.7%)**	**10.58M(58.6%)**	**74.54**	**92.25**

Table 2: Comparison with multiple compression operations-based methods on ImageNet2012.

ber. On ImageNet, we train the networks for 90 and 120 epochs with the initial learning rate is set to 0.001 and 0.01 for VGG and ResNet, respectively. Both mini-batch sizes on VGG and ResNets are set to 256. The learning rate is divided by 10 at epoch 30, 60 and 90. For fine-tuning the compressed networks, we follow the previous work[2] for training MobileNet-V2. We do not compress the first and the last layers of the networks. In our method, we use $T^l = \lfloor 0.01 * (c^l + r^l) \rfloor$ to control the calculation interval of the importance metric and set γ to 0.5.

4.2. Comparison with State-of-the-Art Methods

CIFAR-10. We compare our CC scheme with other methods with a single compression operation (either channel pruning or tensor decomposition) on ResNet-56, DenseNet-40, VGGNet, and GoogLeNet. As shown in Table 1, our method achieves the largest reductions of parameters and FLOPs, but with better performance, compared to other SOTA methods. For instance, compared to HRank [21] based on channel pruning, CC achieves higher reductions of FLOPs (59.9% vs. 54.6%) and parameters (63.3% vs. 55.4%) with higher accuracy (94.88% vs. 94.53%) on GoogLeNet. Meanwhile, compared to ENC [14] based on tensor decomposition, CC has better performance on ResNet-56 (52.0% vs. 49.6% in FLOPs reduction, and 93.64% vs. 93.09% in top-1 accuracy).

ImageNet 2012. We further compare our CC scheme with other methods based on multiple compression operations on VGG-16 and ResNet-50. "CC-GAP" represents that the convolutional layers are compressed by CC and all the fully-connected (FC) layers in VGG-16 are replaced by a global average pooling (GAP) and one FC layer. We also employ the same training parameters to fine-tune the compressed model by CC-GAP. As shown in Table 2, our method achieves the best performance with only a decrease of 0.56% in Top-1 accuracy by a factor of 1.94× compression and 2.12× theoretical speedup on ResNet-50. The result of "Rethinking+SVD" is obtained by compressing the pruned network "ThiNet ResNet50-70% Scratch-B" in [27] and then using SVD-decomposition [44] to conduct further compression. This method is regraded as the intuitive combination of two compression operations. Meanwhile, the result of "FPGM+SVD" is also obtained by compressing the pruned network "FPGM-only 30%" in [9] and then performing SVD-decomposition [44]. More results are given in the supplementary materials. In Table 3, we compare the proposed CC scheme with the state-of-the-art AutoML based pruning methods on MobileNet-V2. We find that our method achieves the best performance with 70.91% top-1 accuracy and only 215M FLOPs on ImageNet 2012.

Method	FLOPs	Top-1 Acc%
MobileNet-V2	300M	71.88
0.75× MobileNet-v2[35]	220M	69.80
AMC[8]	220M	70.80
CC($C = 0.12$)	**215M**	**70.91**

Table 3: Results of MobileNet-V2 on ImageNet2012.

Model	Method	Batch Size		
		1	8	32
VGG-16	Baseline	225ms	1224ms	4603ms
	CC($C = 0.5$)	**173ms**	**784ms**	**2983ms**
ResNet-50	Baseline	96ms	584ms	2420ms
	CC($C = 0.5$)	**88ms**	**420ms**	**1776ms**

Table 4: Results of latency on CPU.

Moreover, we demonstrate the effectiveness of our method in wall-clock time speedup using VGG-16 and ResNet-50 on PyTorch using the CPU AMD Ryzen Threadripper 1900X. As shown in Table 4, our method achieves 1.54× and 1.36× acceleration rates with batch size 32 on VGG-16 and ResNet-50, respectively.

4.3. Ablation Study

4.3.1 Effect of the Two Components in CC

In this section, we evaluate the effectiveness of CC's Global Compression Rate Optimization (GCRO) and Multi-Step Heuristic Compression (MSHC). We use the same compression rate for each layer to replace GCRO (*i.e.,* CC w/o GCRO), and only compute the importance metric once by setting $T = r^l + c^l$ to replace MSHC (*i.e.,* CC w/o MSHC),

[2] https://github.com/d-li14/mobilenetv2.pytorch

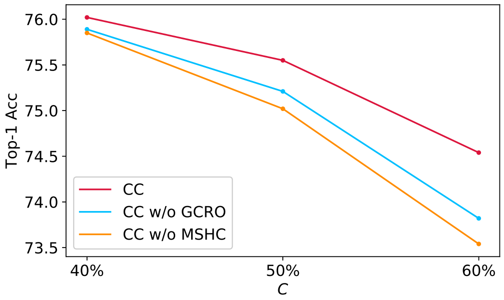


Figure 3: Ablation study of CC with ResNet-50 on ImageNet 2012.

separately. As shown in Fig. 3, both GCRO and MSHC indeed increase the performance of the compressed network. It demonstrates the effectiveness of GCRO and MSHC. We also observe that MSHC is more important in our CC scheme, compared to GCRO. This reveals the compression strategy is more important than the compression rate decision in the union compression.

4.3.2 Effect of the Compression Order

We propose two different variants of our method to investigate the effect of the compression order for the performance of the compressed network under a fixed structure. Our CC method simultaneously prunes and decomposes the weights during compression. Therefore, we can obtain the architecture of the compressed network, including the pruned t_1 input channels and t_2 singular values. In Fig. 4, "Pruning→Decomposition" represents that we first prune the weights and then fine-tune the pruned network, and continue to decompose the pruned network with the following fine-tuning. "Decomposition→Pruning" is in reverse order to "Pruning→Decomposition". Note that these two methods can obtain the same compressed network structure (i.e. the same t_1 and t_2) as that by our CC scheme, but only with different compression orders. The result demonstrates that the compression order (*i.e.,* "Pruning→Decomposition", "Decomposition→Pruning", and Collaborative Compression) has less effect on the final performance. In other words, the different compression strategies influence the performance of the compressed network by producing different network's structures rather than the network's weights. The result demonstrates that the network's structure is more important than network's weights in the compression algorithm, which is also shown in [27].

4.3.3 Compression Percentage Analysis.

Fig. 5 shows the proportion of removed compression units in channel pruning and tensor-decomposition from different layers in ResNet-50. Each bottleneck block has three convolutional layers. In our CC, the first convolution layer in each bottleneck block tends to generate sparse weights rather than low-rank weights. It reveals that the diversity of input features is more important for each bottleneck block in ResNet-50. In contrast, the final layer of the block pays more attention to produce compact features.

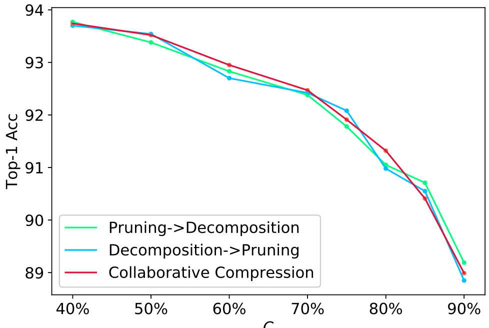


Figure 4: Comparison of different variants of CC with ResNet-56 on CIFAR-10.

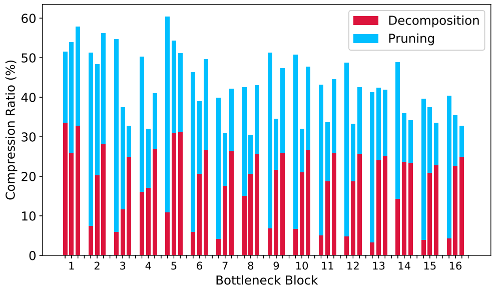


Figure 5: Compression rates of different compression operations in different layers of ResNet-50.

5. Conclusion

In this paper, we first investigate the problem of post-training union compression and propose a novel unified compression framework *Collaborative Compression* (CC) for CNN. Our method is divided into two stages: Global Compression Rate Optimization and Multi-Step Heuristic Compression. The first stage decides the compression rates of the layers by solving an optimization problem based on their compression sensitivity. After that, we compress each layer separately by removing the less important compression units step-by-step. Therefore, our compression process is more reliable, which fully takes the effect of the removed compression units and remaining compression space into account. Extensive experiments on various modern CNNs demonstrate the effectiveness of CC for reducing the computational complexity and model size.

Acknowledgments

This work is supported by the National Science Fund for Distinguished Young Scholars (No.62025603), the National Natural Science Foundation of China (No.U1705262, No.62072386, No.62072387, No.62072389, No.62002305, No.61772443, No.61802324 and No.61702136) and Guangdong Basic and Applied Basic Research FoundationNo.2019B1515120049).

References

[1] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. *European Conference on Computer Vision (ECCV)*, 2018. 1

[2] Xiaohan Ding, Guiguang Ding, Yuchen Guo, and Jungong Han. Centripetal sgd for pruning very deep convolutional networks with complicated structure. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2, 6

[3] Xiaohan Ding, Guiguang Ding, Yuchen Guo, Jungong Han, and Chenggang Yan. Approximated oracle filter pruning for destructive cnn width optimization. *International Conference on Machine Learning*, 2019. 6

[4] Abhimanyu Dubey, Moitreya Chatterjee, and Narendra Ahuja. Coreset-based neural network compression. *European Conference on Computer Vision (ECCV)*, 2018. 1, 3, 7

[5] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *International Conference on Learning Representations (ICLR)*, 2016. 1, 2, 3

[6] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. *Advances in Neural Information Processing Systems (NeurIPS)*, 2015. 1, 2

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 1

[8] Yihui He, Ji Lin, Zhijian Liu, Hanrui Wang, Li-Jia Li, and Song Han. Amc: Automl for model compression and acceleration on mobile devices. *European Conference on Computer Vision (ECCV)*, 2018. 2, 7

[9] Yang He, Ping Liu, Ziwei Wang, Zhilan Hu, and Yi Yang. Filter pruning via geometric median for deep convolutional neural networks acceleration. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 5, 6, 7

[10] Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. *International Conference on Computer Vision (ICCV)*, 2017. 1, 6

[11] Hengyuan Hu, Rui Peng, Yu-Wing Tai, and Chi-Keung Tang. Network trimming: A data-driven neuron pruning approach towards efficient deep architectures. *arXiv preprint arXiv:1607.03250*, 2016. 2

[12] Gao Huang, Shichen Liu, Laurens Van der Maaten, and Kilian Q Weinberger. Condensenet: An efficient densenet using learned group convolutions. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 2

[13] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 1

[14] Hyeji Kim, Muhammad Umar Karim Khan, and Chong-Min Kyung. Efficient neural network compression. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 1, 2, 6, 7

[15] Yong-Deok Kim, Eunhyeok Park, Sungjoo Yoo, Taelim Choi, Lu Yang, and Dongjun Shin. Compression of deep convolutional neural networks for fast and low power mobile applications. *International Conference on Learning Representations (ICLR)*, 2015. 2

[16] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems (NeurIPS)*, 2012. 1

[17] Vadim Lebedev, Yaroslav Ganin, Maksim Rakhuba, Ivan Oseledets, and Victor Lempitsky. Speeding-up convolutional neural networks using fine-tuned cp-decomposition. *International Conference on Learning Representations (ICLR)*, 2014. 1, 2

[18] Chong Li and CJ Richard Shi. Constrained optimization based low-rank approximation of deep neural networks. *European Conference on Computer Vision (ECCV)*, 2018. 2

[19] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. *International Conference on Learning Representations (ICLR)*, 2016. 1, 2, 4, 6

[20] Yawei Li, Shuhang Gu, Christoph Mayer, Luc Van Gool, and Radu Timofte. Group sparsity: The hinge between filter pruning and decomposition for network compression. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 1, 3, 7

[21] Mingbao Lin, Rongrong Ji, Yan Wang, Yichen Zhang, Baochang Zhang, Yonghong Tian, and Ling Shao. Hrank: Filter pruning using high-rank feature map. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 5, 6, 7

[22] Shaohui Lin, Rongrong Ji, Chao Chen, Dacheng Tao, and Jiebo Luo. Holistic cnn compression via low-rank decomposition with knowledge transfer. *IEEE transactions on pattern analysis and machine intelligence (TPAMI)*, 2018. 1, 2

[23] Shaohui Lin, Rongrong Ji, Yuchao Li, Cheng Deng, and Xuelong Li. Toward compact convnets via structure-sparsity regularized filter pruning. *IEEE transactions on neural networks and learning systems (TNNLS)*, 2019. 2, 4

[24] Shaohui Lin, Rongrong Ji, Yuchao Li, Yongjian Wu, Feiyue Huang, and Baochang Zhang. Accelerating convolutional networks via global & dynamic filter pruning. *IJCAI*, pages 2425–2432, 2018. 1

[25] Shaohui Lin, Rongrong Ji, Chenqian Yan, Baochang Zhang, Liujuan Cao, Qixiang Ye, Feiyue Huang, and David Doermann. Towards optimal structured cnn pruning via generative adversarial learning. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2, 6

[26] Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. Learning efficient convolutional networks through network slimming. *International Conference on Computer Vision (ICCV)*, 2017. 2, 6

[27] Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. Rethinking the value of network pruning. *International Conference on Learning Representations (ICLR)*, 2018. 7, 8

[28] Jian-Hao Luo, Jianxin Wu, and Weiyao Lin. Thinet: A filter level pruning method for deep neural network compression. *International Conference on Computer Vision (ICCV)*, 2017. 1, 2

[29] Yuzhe Ma, Ran Chen, Wei Li, Fanhua Shang, Wenjian Yu, Minsik Cho, and Bei Yu. A unified approximation framework for compressing and accelerating deep neural networks. *2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 376–383, 2019. 1

[30] Pavlo Molchanov, Arun Mallya, Stephen Tyree, Iuri Frosio, and Jan Kautz. Importance estimation for neural network pruning. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2, 4, 5

[31] Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. Pruning convolutional neural networks for resource efficient transfer learning. *arXiv preprint arXiv:1611.06440*, 3, 2016. 4

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

[33] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. 2018. 1

[34] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: towards real-time object detection with region proposal networks. *Advances in Neural Information Processing Systems (NeurIPS)*, 2015. 1

[35] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 7

[36] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 1

[37] Frederick Tung and Greg Mori. Clip-q: Deep network compression learning by in-parallel pruning-quantization. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 3

[38] Andreas Veit and Serge Belongie. Convolutional networks with adaptive inference graphs. *European Conference on Computer Vision (ECCV)*, 2018. 2

[39] Andreas Veit, Michael J Wilber, and Serge Belongie. Residual networks behave like ensembles of relatively shallow networks. *Advances in Neural Information Processing Systems (NeurIPS)*, 2016. 2

[40] Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Learning structured sparsity in deep neural networks. *Advances in Neural Information Processing Systems (NeurIPS)*, 2016. 1, 2

[41] Wei Wen, Cong Xu, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Coordinating filters for faster deep neural networks. *International Conference on Computer Vision (ICCV)*, 2017. 1, 2

[42] Ruichi Yu, Ang Li, Chun-Fu Chen, Jui-Hsin Lai, Vlad I Morariu, Xintong Han, Mingfei Gao, Ching-Yung Lin, and Larry S Davis. Nisp: Pruning networks using neuron importance score propagation. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 6

[43] Xiyu Yu, Tongliang Liu, Xinchao Wang, and Dacheng Tao. On compressing deep models by low rank and sparse decomposition. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 1, 3, 7

[44] Xiangyu Zhang, Jianhua Zou, Kaiming He, and Jian Sun. Accelerating very deep convolutional networks for classification and detection. *IEEE transactions on pattern analysis and machine intelligence (TPAMI)*, 38(10):1943–1955, 2015. 1, 2, 7

[45] Shuchang Zhou, Yuxin Wu, Zekun Ni, Xinyu Zhou, He Wen, and Yuheng Zou. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 1