SlimConv: Reducing Channel Redundancy in Convolutional Neural Networks by Features Recombining

Jiaxiong Qiu, *Student Member, IEEE*, Cai Chen, *Student Member, IEEE*, Shuaicheng Liu, *Member, IEEE*, Heng-Yu Zhang, and Bing Zeng, *Fellow, IEEE*

***Abstract*— The channel redundancy of convolutional neural networks (CNNs) results in the large consumption of memories and computational resources. In this work, we design a novel Slim Convolution (SlimConv) module to boost the performance of CNNs by reducing channel redundancies. Our SlimConv consists of three main steps: *Reconstruct*, *Transform*, and *Fuse*. It aims to reorganize and fuse the learned features more efficiently, such that the method can compress the model effectively. Our SlimConv is a plug-and-play architectural unit that can be used to replace convolutional layers in CNNs directly. We validate the effectiveness of SlimConv by conducting comprehensive experiments on various leading benchmarks, such as ImageNet, MS COCO2014, Pascal VOC2012 segmentation, and Pascal VOC2007 detection datasets. The experiments show that SlimConv-equipped models can achieve better performances consistently, less consumption of memory and computation resources than non-equipped counterparts. For example, the ResNet-101 fitted with SlimConv achieves 77.84% top-1 classification accuracy with 4.87 GFLOPs and 27.96M parameters on ImageNet, which shows almost 0.5% better performance with about 3 GFLOPs and 38% parameters reduced.**

***Index Terms*— Slim convolution, channel redundancy, image classification, model compression.**

I. Introduction

IN MOST studies of deep learning, convolutional neural networks (CNNs) have been emphasized with attention given to their impactful modeling for various vision tasks, such as image classification [12], object detection [19] and semantic segmentation [15]. Vanilla convolutional layers are increasingly deeper and more complicated for better accuracy, but these models bring massive parameters and floating-point operations (FLOPs). Hence, the accuracy and cost tradeoffs are ubiquitous in efficient CNN design, especially for mobile and edge devices (e.g., smartphones, drones, and self-driving cars). Over the past several decades, the field has spawned a giddy mix of methods to compress models while preserving accuracies, such as quantizing [48] and pruning [28], [30]. Besides, hand-craft or automatic designs have also made appreciable success.

MobileNet [7] exerts depth-wise and point-wise convolutions to build a small network with low latency, which achieves over 71% top1-accuracy with merely 3.5M parameters and 0.3 GFLOPs on ImageNet [12]. On the other hand, Shufflenet [8] resorts to the channel shuffle operation to improve the performance of tiny networks owing to the sufficient process to the inter-channel information. In addition, Neural architecture search (NAS) methods [33], [34] learn to tune network architectures and gain efficient models with high potentials.

These methods are primarily dedicated to explicitly diminishing the redundant parameters and decreasing the computational cost. However, the lightweight models have fewer layers or channels than large models, causing losses of high-layer or channel-wise information. According to [1], this information is significant for representing stronger non-linear expression ability and learning more complex transformations, which helps to fit more general and difficult inputs. Thus, this information loss brought by many of the lightweight models hampers the effort to the performance. Researchers have investigated plenty of redundant features in the network channels, which determine the model size and influence the implicit representation of models. OctConv [23] discussed the effectiveness of dropping the low-frequency part of features by interleaving connections in the convolutional layers, thus reducing the spatial redundancy. Besides, the series of Network Slimming [22], [38], [39] conducted nice explorations to directly prune the redundant channels despite the difficulty in dealing with the batch normalization. In particular, Autoslim [22] achieved state-of-the-art performance with the help of NAS [33].

Redundant features often waste storage and computational resources. We notice that simple linear operations can reduce redundant features. Here, we visualize all features in Fig. 1 from the first layer second stage of ResNet50 trained on the ImageNet. Some of the cells contain almost empty contents,

Manuscript received August 7, 2020; revised March 11, 2021 and May 17, 2021; accepted June 24, 2021. Date of publication July 7, 2021; date of current version July 15, 2021. This work was supported in part by the National Natural Science Foundation of China (NSFC) under Grant 61872067, Grant 62031009, and Grant 61720106004; in part by the 111 Projects under Grant B17008; and in part by the Sichuan Science and Technology Program under Grant 2019YFH0016. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Zhu Li. *(Corresponding authors: Shuaicheng Liu; Heng-Yu Zhang.)*

Jiaxiong Qiu, Cai Chen, Shuaicheng Liu, and Bing Zeng are with the School of Information and Communication Engineering, Institute of Image Processing, University of Electronic Science and Technology of China, Chengdu, Sichuan 611731, China (e-mail: liushuaicheng@uestc.edu.cn).

Heng-Yu Zhang is with the Department of Cardiology, West China Hospital, Sichuan University, Chengdu, Sichuan 610041, China (e-mail: zhanghe8635_cn@sina.com).

Digital Object Identifier 10.1109/TIP.2021.3093795

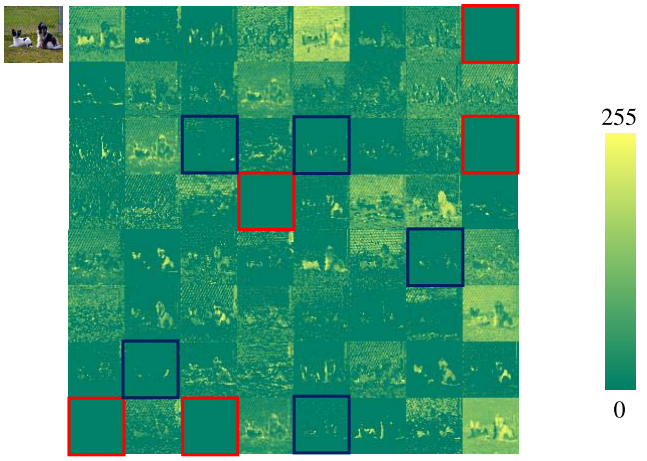


Fig. 1. Deep features are not compact in CNN Networks. We visualize all feature maps at the first layer second stage in ResNet-50 trained on the ImageNet for a classification task. Some redundant feature maps marked with red boxes are shown nearly empty contents, and other feature maps marked with blue boxes have little information.

and some of them have little information. One way to reduce such redundancies is splitting and summation. Inspired by group convolutions, we propose first to split features into several groups. Then, different groups are merged into one group, such that redundant feature groups are reduced. For example, in Fig 1 we show 8×8 feature cells, which can be split into two 4×8 groups. Each cell in one group can sum up with the cell at the same location in the other group, merging to one cell to save space. In this way, features with rich information can sum with less informative ones (marked with red and blue boxes in Fig 1). As a result, only informative cells are retained.

On the one hand, the split and sum operation can reduce the feature redundancy if adding proper cells, e.g., adding one informative with one less informative. On the other hand, when we weight two different informative features and then sum them to one feature, each pixel of this image has coded more information from this pixel of the original feature with the larger weight, and the other is damaged. So the representation of features could be damaged if adding two informative cells or become less effective when adding two empty cells. This could happen because cells are paired up for summation based on their locations instead of their contents. If we look at their contents, then the implementation is non-trivial. To solve this problem, we propose a novel operation **weights flipping**, which can essentially reduce the probability of improper summations.

To this end, we propose a novel convolutional operation, which we term as Slim Convolution (SlimConv), to obtain good performances while saving the computational resources simultaneously. The SilmConv can reduce and reform feature channels to improve the quality of feature representations. In addition, SlimConv is designed as a plug-and-play module that can be embedded into various popular CNN models. The SlimConv reduces calculations and maintains the capability of feature representations, within which the weights flipping operation plays an important role. It is a light-cost operation but can maintain representations significantly during the channel reduction.

The configuration of SlimConv is depicted in Fig. 2, which mainly consists of three operations: **reconstruct**, **transform**, and **fuse**. For reconstructing, we introduce SE-module [2] to obtain channel-wise weights of full channels (Fig. 2 (a)) initially. Second, we input feed-forward feature maps into two pathways. For the top pathway, we multiply the input features by the weights and halve the weighted features, followed by the element-wise summation of the pieces (Fig. 2 (c)). For the lower pathway (Fig. 2 (d)), we first flip the weights (Fig. 2 (b)) and then conduct the same process as the top pathway. As a result, the feature channels are 2-fold after being reduced in each pathway. The "transform" stage means transforming the reconstructed features through convolutional layers. For transforming, we follow the strategy of SKNet [10] which also uses the bottleneck structure for two pathways. Specifically, a convolution layer with 3×3 kernel serves as the transformer for the top pathway (Fig. 2 (e)). Meanwhile, a 1×1 convolution layer and a following layer with 3×3 kernel are adopted for the bottom (Fig. 2 (f)). Here, the transformer with a small kernel size also reduces channels by half. Finally, the features from two pathways are concatenated for the feature fusion (Fig. 2 (g)). In this way, as demonstrated by our experiments, our SlimConv can reduce the number of channels substantially while retaining the representation capability in theory. More importantly, SlimConv is a plug-and-play module that can be applied to enhance the efficiency of various backbone architectures by simply replacing their original convolutions.

To evaluate the performance of the SlimConv module, we conduct various experiments for common visual tasks on leading benchmarks, including ImageNet [12], MS COCO2014 [17], Pascal VOC2012 segmentation [16], and Pascal VOC2007 detection [18]. Experimental results show that the performance of our model is competitive compared with most available state-of-the-art networks concerning accuracy and efficiency. We also re-implement all lightweight models by the same training strategy for fair comparisons. Remarkably, we re-implement the MobileNet(v2) [7] equipped with SlimConv achieves 71.7% top-1 accuracy with only 0.239 GFLOPs and 3.38M parameters, which shows 0.3% higher performance with over 20% lower computational cost and 4% less parameters. Likewise, when equipping ShuffleNet(v2) [8], the accuracy improves nearly 1% while still reducing 40 MFLOPs and $30K$ parameters.

In summary, our main contributions are:

- We design a plug-and-play module named SlimConv that can compress models and enhance the representation ability of CNNs.
- We propose to reconstruct features to reduce the channel redundancy, within which a weights flipping operation can largely strengthen the diversity of features.
- We integrate various CNN backbones with the proposed SlimConv. Meaningful improvements have been achieved through experiments on challenging tasks, such as image classification, semantic segmentation, and object detection.

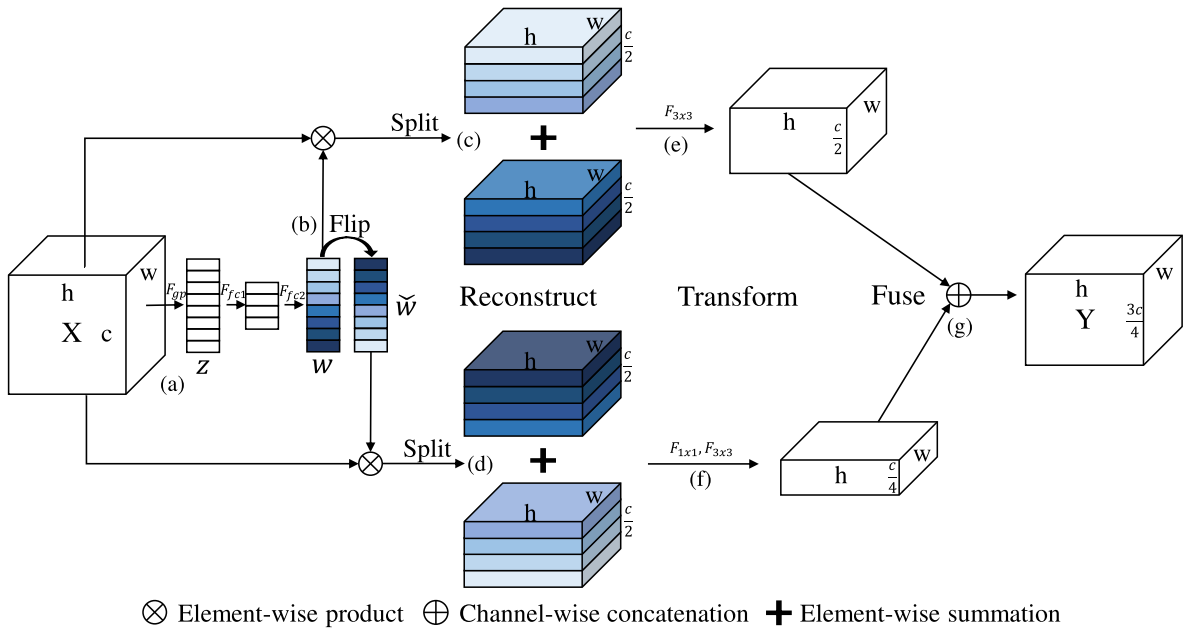


Fig. 2. Pipeline of Slim Convolution. For the reconstruct phase, (a) the input features X with size $w \times h \times c$ is fed into the SE-module [2] to obtain channel-wise weights w. (b) The X is then fed into two pathways. (c) For the top pathway, the weights w multiplied with the feature X. (d) For the lower pathway, we flip the weights w to obtain $\hat{w}$ and then multiplied with X. The features in each pathway are split into two groups. Then, we apply element-wise summation on two groups, yielding features with size $w \times h \times c/2$ in each pathway. For the transform phase, (e) a convolution layer $F_{3\times3}$ with kernel size 3×3 is applied for the top pathway and (f) two convolution layers $F_{1\times1}$ and $F_{3\times3}$ are adopted for the lower pathway, producing features with sizes $h \times w \times c/2$ and $h \times w \times c/4$ accordingly. For the fuse phase, (g) features from two pathways are concatenated as the output with size $h \times w \times 3/4c$.

The rest of the paper is organized as follows: Sec. II introduces the related work in the area. Then Sec. III explains the proposed SlimConv and its example bottleneck architecture. Experiments and analysis are presented in Sec. IV, and finally in Sec. V, concluding remarks are discussed.

II. Related Work

A. Efficient Network Architecture Design

Pioneering works on computer vision tasks achieved higher accuracy every year by prompting the network architecture to be deeper and more complex, such as AlexNet [25] and VGG [26] on ImageNet competition. From hand-craft designs, the increase in the number of parameters and computational complexity made the improvement of accuracy less beneficial. InceptionNet [13] proposed an Inception module to deepen the network with few added parameters. ResNet [1] and DenseNet [21] utilized the efficient residual block by adopting shortcut connections. ResNeXt [5] replaced traditional convolutions with group convolutions and introduced cardinality to increase model capacity. Res2Net [41] combined ResNet and ResNeXt and proposed an unusual multi-scale method. As the deployment for neural networks on terminal devices requires more lightweight models, the networks are encouraged to mobile-size such as SqueezeNet [24], ShuffleNet [8], Xception [27], and MobileNet [7]. Except for modifying backbones, some methods attempted to prune the trained models, such as [28]–[32], [56], [58], [60], which pruned the inconsequential connections and weight to decrease the model size at a moderate accuracy loss. Binary neural networks, such as PCNN [59] push network quantization to the extreme, which dramatically accelerating both forward and backward computation. ACNet [57] trained accessorial asymmetric kernels to re-parameterize the regular kernels and improved the performance without adding inference-time computations. Recently, neural architecture search (NAS) has became a trend for the efficient CNN field. These methods, such as NAS [33], PNAS [34], and MNASNet [35], obtained the best network architecture by learning to explore the network structures, including width, depth, convolution kernels, and connections. EfficientNet [36] and EffiecientDet [37] produced experimental evidence for scaling normal models to larger ones as the backbone with the method of NAS [33], aiming to maximize accuracy with limited resources. However, they need to mobilize a large number of computing resources to fulfill the task automatically. Besides, the series of Slimmable Network [22], [38], [39] was another kind of approach that proposed to learn a scale factor for each layer to reduce the network width while preserving the performance. It is restrictive for the network architecture to search for ideal models for many tasks, and it needs iterative training procedures.

B. Computational Cost Reduction

For designing an effective lightweight CNN, the evolution of the core convolution operation is the most direct way. It can significantly reduce the channel redundancy in the network so that it does not need to prune or compress the model with extra computations after training. Furthermore, it can reorganize the features for performance improvements. Based on the group convolutions [5], [25], ShuffleNet [8] proposed channel shuffle operation to build a desirable lightweight

model due to its improvement of the information flow across feature channels. MobileNet [7] introduced depth-wise separable convolutions, which demonstrated good representative ability than regular convolutions, thus reducing the number of parameters and accelerating the training. OctConv [23] and Multi-grid CNN [40] respectively proposed octave convolution and multi-grid convolution to exploit multi-scale representations. Especially, octave convolution took more consideration regarding the efficient design to reduce the feature redundancy and strengthen the information exchange between channels between high or low frequencies. GhostNet [50] discussed that some redundant features are useful, which can help networks understand the input data. As such, it proposed to use cheap operations to embrace redundant features. It uses 1×1 convolutional layer to generate a part of output feature maps and then uses the next convolutional layer to generate the other part of output feature maps like DenseNet. This is similar to the transform step in the bottom pathway of our module. However, we do not make features from the 1×1 convolutional layer as the part of output features in the bottom pathway of our module. Moreover, the main contribution of our method is on the reconstruction step, which uses the Split and Sum operation to compress the original input feature information. And then, we integrate branches of different arranged weights to improve the feature diversity and supplement the information loss.

However, these methods still suffer from the problem of feature redundancies. In contrast, our SlimConv can decrease the computational cost and storage at the same time by pruning a part of the channels of a convolution layer.

C. Attention Mechanisms

Attention has been widely used in many research fields such as salient object detection [42], depth completion [43], image super-resolution, [44] and facial expression recognition [45]. Wang *et al.* [46] proposed an attention module with Encoder-Decoder style for the image classification task. SENet [2] introduced a lightweight attention module to re-calibrate the feature map by channel-wise weights. Besides channel-wise importance, CBAM [47] considered spatial attention and designed two sequential sub-modules, including channel-wise and spatial attention. SKNet [10], following InceptionNet and SENet, adopted a soft-attention mechanism to make networks select the features with different receptive fields automatically. In contrast, we just use the attention mechanism (i.e., a general SE-module) to generate weights for our proposed features reconstructing. The motivation is that we want to add attentive weights to the channel features to avoid the potential excessive values in feature groups after the Split and Sum operation. Also, it benefits our architecture as performing the feature recalibration to learn to use global information to emphasize information features and suppress useless features selectively.

III. Method

A. Slim Convolution

Figure 2 shows our pipeline of SlimConv, which includes two pathways and consists of three steps: *Reconstruct*, *Transform*, and *Fuse*. Note that Fig. 2 only shows our default setting, and it is flexible to expand to fit different bottlenecks.

1) Reconstruct: Given an input feature map: $X \in \mathbb{R}^{C \times H \times W}$, we use SE-module [2] to obtain channel-wise weights w. We replace the fully connected (fc) layer with the convolutional layer whose kernel size is 1×1, and use larger reduction ratio 32 as the default setting. The whole process of acquiring w can be expressed as:

$$\begin{cases} z = F_{gp}(X) = \dfrac{1}{H \times W} \displaystyle\sum_{i=1}^{H} \sum_{j=1}^{W} X(i, j), \\ w = \sigma(F_{fc2}(\delta(F_{fc1}(z)))). \end{cases} \tag{1}$$

where $z \in \mathbb{R}^C$ contains channel-wise statistics, σ refers to the sigmoid function and δ is the ReLU [3] activation. F_{fc1} and F_{fc2} are convolution operations, F_{fc1} includes the Batch Normalization [4].

In the top pathway, we multiply features by w, yielding weighted features X_w. Then, we split X_w into two parts (X_w^1, X_w^2), and sum them to compress the number of features to half:

$$\begin{cases} X'_w = X_w^1 + X_w^2, \\ X_w = w * X, \\ X_w^1 \cup X_w^2 = X_w. \end{cases} \tag{2}$$

The compression can reduce redundant features but results in the loss of valuable information. To deal with it, we propose the bottom pathway.

In the bottom pathway, we disrupt the order of feature weights through weights flipping. Further, we use flipped channel-wise weights $\breve{w}$ to go through the same operations as the top pathway to obtain the half-channel features $X'_{\breve{w}}$.

2) Transform: We follow the bottleneck design rules of ResNet [1] and conduct two transformers. The top transformer $F_{3\times3}$ is a convolution layer with kernel size 3. The bottom transformer contains two convolution layers $F_{1\times1}$, $F_{3\times3}$ with kernel sizes 1 and 3 respectively. The convolution layer with kernel size 1 reduces the number of channels by half, followed by the convolution layer with kernel size 3.

3) Fuse: We concatenate different features from two pathways to integrate information. Our SlimConv outputs feature Y with channel number $\frac{3c}{4}$ (c is the original channel number).

Notably, the classic group convolution [25] does not reduce the channel. Unlike that, our method intends to merge features to reduce the channel of features after grouping. Secondly, the group convolution convolutes each feature group separately and then concatenates the shared parameters. While our method does a simple convolution after merging the feature groups, resulting in a better generalization ability than that of group convolution.

B. Network Architecture

Since our proposed SlimConv takes N-channel features as input and output $\frac{N}{k}$-channel ($k > 1$, default is $\frac{4}{3}$) features, it has only one extra hyperparameter k. It can be easily integrated into many state-of-the-art CNN models, such as

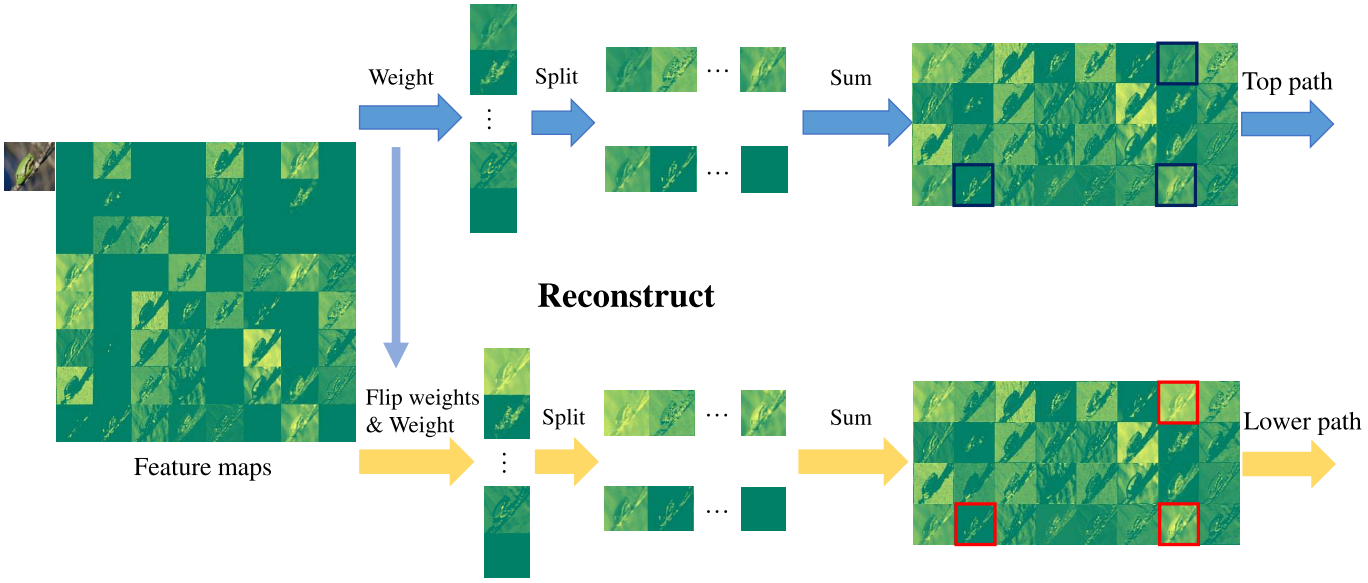


Fig. 3. Visualization of feature maps at the first layer second stage of SlimConv-equipped ResNet-50 during Reconstruct phase. The input feature maps contain 8×8 feature cells, many of which are less informative. Then, the features are feed into the SE-Module to produce weights. Next, the features are feed into two pathways. For the top pathway, the weights are directly multiplied with the features. For the lower pathway, the weights are flipped before multiplying with the features to introduce diversities. For each pathway, the feature cells are split into two 4×8 cells. Each cell in a group sums with the cell at the same location in the other group, yielding 4×8 cells for each pathway. The final output of reconstructing phase is still 8×8 cells from two pathways. However, the output cells become more informative, and the features of the two pathways are also different due to weights flipping. We highlight some obvious differences with boxes.

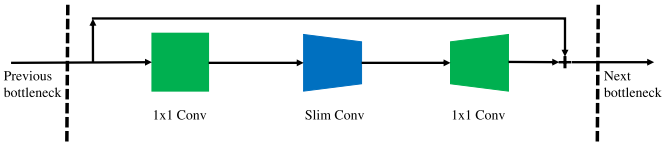


Fig. 4. **Modified Res-Bottleneck**.

ResNet [1], ResNeXt [5], DLA [6], MobileNet [7], ShuffleNet [8], Big-Little Net [9]. We take ResNet [1] as an example.

As illustrated in Fig. 4, the proposed SlimConv displaces the previous ordinary convolution layer, which with kernel size 3 and decreases the number of output channels. So the input of the last convolution layer with kernel size 1 needs to be changed accordingly.

Particularly, when embedding SlimConv in DLA46-C [6] and MobileNet V2 [7] which only have few channels in a convolution layer, we replace $\frac{C}{32}$ with $max(C/r, L)$ (r and L are fixed values) in the SE-Module.

C. Analysis on Complexities

We can easily integrate SlimConv into existing well-designed neural networks to decrease the computation and storage costs. Most of the parameters in SlimConv are from **transform**. Here, we analyze the reduction of theoretical memory usage by using the SlimConv module. The 1×1 conv layer in the lower pathway is fixed, and the other $n \times n$ conv layers can be modified according to the basic model. The parameters of the basic convolution layer can be calculated as:

$$P_b = C_1 \times n \times n \times C_2 = n^2 C_1 C_2 \tag{3}$$

where n is kernel size of the convolution layer, C_1 and C_2 are the number of input and output feature channels. From Fig. 2, we set $k = \frac{4}{3}$ as example, parameters of our proposed SlimConv module consists of:

$$\begin{aligned} P_s &= \frac{1}{2}C_1 \times 1 \times 1 \times \frac{1}{4}C_1 + \frac{1}{4}C_1 \times n \times n \times \frac{1}{4}C_2 \\ &\quad + \frac{1}{2}C_1 \times n \times n \times \frac{1}{2}C_2 \\ &= (\frac{5}{16}n^2 + \frac{1}{8}\frac{C_1}{C_2})C_1 C_2 \end{aligned} \tag{4}$$

We can adjust k to change input and output feature channels for flexible SlimConv modules. Because we reduce the number of channels, the computation for convolution is also reduced.

D. Effects of Flipping Weights

Our method of reducing channel dimension is a lossy compression method, which could cause some loss of feature information. We adopt a weights flipping operation to obtain rearranged weights. These weights introduce more possible situations for feature summation in the same channel and then help to generate more diverse features after feature summation. More diverse features can enhance the representation ability of neural networks in the reduced channel. Therefore, the core of our model is a weights flipping operation that can vastly improve the feature diversities, contributing to the performance crucially.

To better understand what happens inside weights flipping operation, we visualize the neural cells of our SlimConv-equipped ResNet-50 at the second layer trained on the task of ImageNet in Fig. 3. We choose a frog image and show its 8×8 feature cells. As seen, most of the cells are

less informative. Then, we feed the feature cells into two pathways. Therefore, each pathway still contains 8×8 cells. For the top pathway, the weights produced by SE-Module are directly multiplied with the features. For the lower pathway, flipped weights are multiplied with the features to introduce diversities. A detailed example is depicted in Fig. 3. For each pathway, the cells are split into two groups. Each group contains 4×8 cells, according to their physical locations. Each cell sums with the cell of the same coordinate in the other group to complete the reconstruction phase. As a result, the output features are more informative for each pathway as no empty cells can be observed. Moreover, due to the weights flipping operation, the two output features are different and both informative, which means the feature diversity is expanded. In Fig 3, we highlight some cell differences with blue boxes in the top pathway and red boxes in the lower pathway. After reconstruction, the outputs are still 8×8 cells, with each pathway contains 4×8 cells. However, they become more informative compared with the inputs.

IV. Experiments

To verify the effectiveness of our method, we conduct comprehensive experiments regarding various vision tasks on several benchmark datasets. Firstly, we introduce the environment of implementation and all the datasets used for experiments in the subsection IV-A. Then, as image classification is the typical task of CNN, we compare the state-of-the-art methods on ImageNet and CIFAR datasets in subsection IV-B and IV-C respectively. Also, we analyze the compressibility of our module in the subsection IV-D. Afterwards, we demonstrate our method on semantic segmentation and object detection in subsection IV-E and IV-F respectively. It follows by extensive ablation studies on ImageNet dataset in subsectionIV-G. Finally, in the subsection IV-H, we discuss how redundant features are compressed.

A. Implementation Details

Our proposed model and other state-of-the-art CNN-based models we used are all implemented by PyTorch [11]. Similar to [5], these models are trained on less than 8 GeForce RTX 2080 Ti GPUs. We mainly validate the effectiveness of our proposed model on 5 challenging datasets:

1) ImageNet: We use the most popular dataset ImageNet [12] for all the experiments on image classification. ImageNet is also a common benchmark, containing 1.28 million images for training and 50k images for validation. All these images have labels from 1000 categories. We train the SlimConv-equipped models on training images and pick the model with the best top-1 error performance on validation images. We conduct the random-size cropping to 224×224 and random horizontal flipping [13]. For fair comparisons on middle-sized models, we use the same data augmentation and training strategy as [1], [7], and [9] respectively. Specifically, we train middle-sized and multi-scale models using SGD with weight decay 0.0001, momentum 0.9, and a mini-batch of 256 on 8 GeForce RTX 2080Ti GPUs. The learning rate is initially set to 0.1 and divided by 10 every 30 epochs.

TABLE I

Performance Comparison for ResNet [1], ResNeXt [5], DenseNet [21], SENet [2], Ghost-ResNet-50 [50], SPConv-ResNet-50 [51] and Our Integrated Models on ImageNet

Model	Top-1 Error	FLOPs(10^9)	Params(10^6)
ResNet-50	23.85	4.09	25.56
SE-ResNet-50	**23.29**	4.12	28.09
SPConv-ResNet50-α1/2	23.74	2.97	18.34
Sc-ResNet-50(ours)	**23.29**	**2.65**	**16.76**
SPConv-ResNet50-α1/8	24.60	2.62	16.22
Ghost-ResNet-50(s=2)	24.99	2.15	13.95
Sc-ResNet-50(k=⅝)(ours)	**24.48**	**1.88**	**12.10**
ResNet-101	22.63	7.80	44.55
DenseNet-161	22.35	7.73	28.68
SE-ResNet-101	22.38	7.85	49.33
Sc-ResNet-101(ours)	**22.16**	**4.82**	**27.96**
ResNeXt-50(32×4d)	22.38	4.23	25.03
SE-ResNeXt-50(32×4d)	**21.78**	4.23	27.56
Sc-ResNeXt-50(32×4d)(ours)	22.03	**3.82**	**22.49**
ResNeXt-101(32×4d)	21.20	7.97	44.18
SE-ResNeXt-101(32×4d))	**21.00**	7.97	48.96
Sc-ResNeXt-101(32×3d,k=2)(ours)	21.18	**4.58**	**23.70**

For lightweight models, we use SGD with weight decay 4e-5. The learning rate is set to 0.05 initially with the cosine strategy.

2) CIFAR: To validate the performance of our SlimConv on a small-sized dataset, we conduct experiments on the CIFAR-100 dataset [49], which contains 50k images for the training and 10k images for the testing with 100 classes. The size of testing images is 32×32. We replace the corresponding layer of the basic block with our proposed SlimConv module. For fair comparisons, we keep the same training and testing strategy unchanged.

3) Pascal VOC2012 Aug: We evaluate the performance of our proposed model on semantic segmentation by using PASCAL VOC12 dataset [14]. Following previous works, we use the augmented version of the PASCAL VOC12 dataset [16] which contains $10,582$ training images and $1,449$ validating images from 21 classes. We use the state-of-the-art method Deeplab v3+ [15] as the segmentation framework and the exact implementation details for all models.

4) MS COCO2014 & PASCAL VOC2007: For object detection, we evaluate our SlimConv on MS COCO dataset [17] and PASCAL VOC2007 dataset [18]. We take the widely used method Faster RCNN [19] as the detection framework and use the same strategy to train and test models.

B. ImageNet

1) Comparing With Middle-Sized Models: Table I reports 4 group of results according to the complexity. Models equipped with our SlimConv contain the prefix 'Sc' in all the tables. In the first group, our integrated Sc-ResNet-50 achieves almost 0.6% better accuracy, 35% fewer FLOPs and parameters than non-equipped original ResNet-50 [1]. When we adjust k to $\frac{8}{3}$, Sc-ResNet-50 reduces 54% FLOPs and 53% parameters from ResNet-50. With the modified SE-Mouble, our Sc-ResNet-50 shows the same accuracy performance as SE-ResNet-50 [2], but cost over 40% fewer parameters. When compared with other manual efficient modules, our Sc-ResNet-50s have better accuracy, fewer FLOPs, and parameters than SPConv-ResNet-50s [51] and Ghost-ResNet-50 [50]. These modules both use group

TABLE II

PERFORMANCE COMPARISON FOR MULTI-SCALE MODELS: BL-RESNET-50 [9], OCT-RESNET-50 [23] AND OUR SC-BL-RESNET-50 ON IMAGENET

Model	Top-1 Error	FLOPs(10^9)	Params(10^6)
ResNet-50	23.85	4.12	25.56
bL-ResNet-50($\alpha = 2, \beta = 4$)	22.69	2.85	26.69
Oct-ResNet-50($\alpha = 0.5$)	22.60	2.40	25.60
Sc-bL-ResNet-50($\alpha = 2, \beta = 4$)(ours)	**22.34**	**2.16**	**17.77**

TABLE III

PERFORMANCE COMPARISON FOR LIGHTWEIGHT MODELS(SQUEEZENET [24], DLA [6], MOBILENET V2 [7] AND SHUFFLENET V2 [8]) AND OUR INTEGRATED MODELS ON IMAGENET

Model	Top-1 Error	FLOPs(10^8)	Params(10^6)
Squeezenet-B	39.60	7.20	1.20
DLA-46-C	35.96	5.90	1.31
Sc-DLA-46-C(ours)	**35.69**	**4.83**	**0.97**
1.0 ShuffleNet(v2)	32.01	1.45	2.28
1.0 Sc-ShuffleNet(v2)(ours)	**31.08**	**1.42**	**2.25**
1.0 MobileNet(v2)	28.54	3.00	3.51
1.0 Sc-MobileNet(v2, k=$\frac{8}{3}$)(ours)	**28.26**	**2.39**	**3.38**

convolution operations for better performance, and SPConv-ResNet-50s also utilize channel attention like SKNet [10].

In the second group, we consider deeper models and take ResNet-101 [1] as the basic model. Our integrated Sc-ResNet-101 also achieves nearly 0.5% less top-1 error than the basic model while reducing FLOPs and parameters simultaneously by over 38%. Our Sc-ResNet-101 is also more efficient than DenseNet-161 [21] and SE-ResNet-101 [2].

In the third group, we add our module to ResNeXt-50 [5]. We do the same group convolution operation in the last two convolution layers with kernel size 3×3. Due to our SlimConv can reduce channels, the width of each group is decreased. Our integrated Sc-ResNeXt-50 achieves almost 0.4% better accuracy while also reduces the computational cost and storage by 10%. In the last group, we change our hyperparameter k to 2, reducing the width from 128 to 96 during the integration to ResNeXt-101 [5]. Even though having a thinner architecture, our Sc-ResNeXt-101 achieves a slightly improved accuracy, with over 42% fewer FLOPs and 46.4% fewer parameters than the wider basic model. Our Sc-ResNeXts has less accuracy gain than SE-ResNeXt because of the significant reduction in parameters and FLOPs for ResNeXts.

2) Comparing With Multi-Scale Models: Multi-scale strategy [9], [23] is effective for image classification, Table II reports the results. Here, we chose a model from Big-Little Net [9], named bL-ResNet-50, to be our basic model, where 'bL' stands for Big-little. As seen, SlimConv equipped model, Sc-bL-ResNet-50, achieves nearly 0.4% better accuracy, 24.2% fewer FLOPs, and over one-third fewer parameters than the basic model. Compared with the SOTA model Oct-ResNet-50, our model also achieves almost 0.3% less top-1 error, fewer FLOPs, and over 30% fewer parameters.

3) Comparing With Lightweight Models: We conduct three groups of experiments to test our performances on lightweight models. Table III reports the results. In the first group, we choose DLA-46-C [6] as the baseline and implement an efficient model with parameters less than 1 M. Our integrated Sc-DLA-46-C achieves almost 0.3% better accuracy and 18.1% fewer FLOPs with only 0.97 MB parameters than the baseline. Compared to Squeezenet [2], the SlimConv embedded model has improvements as nearly 4% better accuracy, 32.9% fewer FLOPs, and 19.2% fewer parameters. We also choose the most popular lightweight models MobileNet [7] and ShuffleNet [8] as the baseline models. In the second group, our integrated Sc-ShuffleNet performs nearly 1% better accuracy, less computational cost and parameters than ShuffleNet [8]. In the third group, our Sc-MobileNet also achieves almost 0.3% less top-1 error, 20% fewer FLOPs, and fewer parameters than the basic MobileNet [7].

TABLE IV

INFERENCE LATENCY TIME (MS/BATCH), GPU MEMORY (GB) AND COMPRESSED RATIO (%) OF RESNET-50 AND SC-RESNET-50S ON IMAGENET. THE SIZE OF ONE BATCH IS 128

Model	latency(ms/batch)	Memory(GB)	Compressed ratio(%)
ResNet-50	205	2.14	0
Sc-ResNet-50(k=4/3)(ours)	218	2.28	34.4
Sc-ResNet-50(k=8/3)(ours)	197	2.23	52.7

4) Comparing With Network Pruning Methods: As shown in Fig. 6, we compare our Sc-ResNets with other network pruning methods including AutoSlim [22], Taylor-FO-BN [54], DMCP [52], SSS [53], MetaPruning [55] and Slimmable [38]. Without any searching, SlimConv-equipped ResNets surpass most of these network pruning methods on both accuracy and FLOPs. Because some network pruning methods aim to reduce FLOPs, we show the results of different FLOPs. Note that Sc-ResNets also have meaningful parameter reduction.

5) Class Activation Mapping: We present some results of class activation mapping by Grad-CAM [20] to visualize where CNNs pay more attention to image classification. The visualization examples are shown in Fig. 5. The lighter the area, the more attention the network has. Compared to ResNet-50 [1], the results of our integrated Sc-ResNet-50 concentrate on small objects such as 'Basketball' and 'Ice cream'. Even in complex scenes, the SlimConv equipped models can still concentrate on the area close to the object while ResNet-50 has been distracted. For large objects such as 'Ballpoint', 'Airship', 'Elephant' and 'Mosque', our activation maps are more accurate and cover objectives more comprehensively than the basic results.

6) Time: Table IV shows the real inference latency, GPU memory cost on a GeForce RTX 1080 Ti GPU, and storage consumption for ResNet-50 and our Sc-ResNet-50s. When increasing the compression ratio of ResNet-50, the SlimConv-equipped model can achieve faster inference.

C. CIFAR

1) Comparing With Lightweight Models: Table V reports 3 groups of results according to the complexity. When equipped with our SlimConv, the most widely used lightweight models ShuffleNet [8] and MobileNet [7]) both achieve over 1% better accuracy with less computational cost and parameters. Our integrated Sc-ResNet-56 achieves better accuracy than the

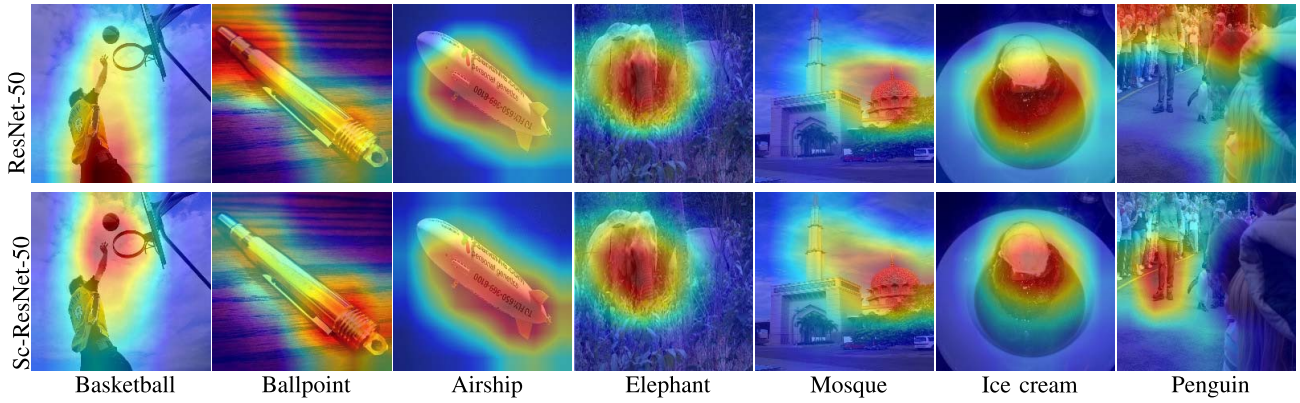


Fig. 5. Grad-CAM [20] visual comparison for ResNet-50 [1] and our Sc-ResNet-50 on ImageNet.

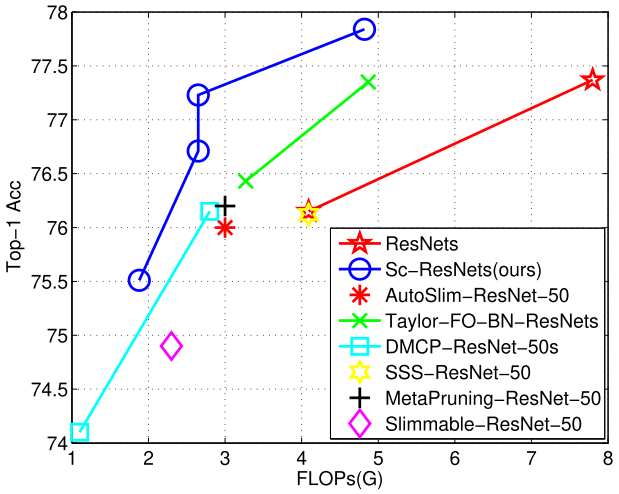


Fig. 6. Top-1 accuracy and FLOPs results comparing with network pruning methods on ImageNet dataset. Our ResNets including: Sc-ResNet-50($k = \frac{8}{3}$), Sc-ResNet-50, Sc-ResNet-50(cosine) and Sc-ResNet-101.

TABLE V

PERFORMANCE COMPARISON FOR SHUFFLENET [8], MOBILENET [7], NASNET [33], RESNET-56 [1] AND OUR INTEGRATED MODELS ON CIFAR-100

Model	Top-1 Error	FLOPs(10^6)	Params(10^6)
1.0 ShuffleNet(v2)	29.46	46.22	1.36
1.0 Sc-ShuffleNet(v2)(ours)	**27.74**	**44.58**	**1.34**
1.0 MobileNet(v2)	28.15	94.72	2.41
1.0 Sc-MobileNet(v2)(ours)	**26.73**	**74.80**	**2.19**
ResNet-56	28.41	126.56	0.86
Sc-ResNet-56(ours)	**28.29**	**67.28**	**0.46**
NasNet	20.85	681.88	5.22
Sc-NasNet(ours)	**20.56**	**644.71**	**5.05**

baseline ResNet-56 [1] with nearly half parameters. In particular, we further integrate the neural-architecture-search model (NasNet [33]) with our SlimConv module, yielding improved top-1 error as well as less consumption of resources than the original non-equipped basic model.

2) Comparing With Middle-Sized Models: We conduct 4 groups of experiments for middle-sized models to test our performances with different network design mechanisms. Table VI reports the results. In the first group, our

TABLE VI

PERFORMANCE COMPARISON FOR RESNET-50 [1], OCT-RESNET-50 [23], SKNET-50 [10], SE-RESNET [2], RESNEXT [5] AND OUR INTEGRATED MODELS ON CIFAR-100

Model	Top-1 Error	FLOPs(10^9)	Params(10^6)
ResNet-50	22.88	1.305	23.71
Sc-ResNet-50(ours)	**20.63**	**0.836**	**14.91**
Oct-ResNet-50($\alpha = 0.5$)	19.68	0.936	23.71
Sc-Oct-ResNet-50($\alpha = 0.5$)(ours)	**19.41**	**0.876**	**20.04**
SKNet-50	19.84	1.568	28.16
Sc-ResNeXt-50(ours)	**19.79**	**1.221**	**20.64**
SE-ResNet-50	22.26	1.317	26.50
Sc-SE-ResNet-50(ours)	**21.63**	**0.847**	**17.61**
SE-ResNet-101	21.75	2.538	47.77
Sc-SE-ResNet-101(ours)	**20.96**	**1.564**	**31.08**

integrated Sc-ResNet-50 achieves almost 2.3% better accuracy, almost 36% fewer FLOPs, and 37% fewer parameters than non-equipped original ResNet-50 [1]. In the second group, we embed our SlimConv into the last stage of Oct-ResNet-50 [23], which achieves almost 0.3% better accuracy than the baseline while reducing about 60 MFLOPs and 15.5% parameters. Compared to the SOTA method SKNet-50 [10] which takes ResNeXt-50 [5] as the basic model, our integrated Sc-ResNeXt-50 also achieves better accuracy, over 22% fewer FLOPs and 26.7% fewer parameters. Next, we study the impact of feature channel calibration by using SE-ResNet [2] as the basic model. The SlimConv-equipped SE-ResNet-50 achieves about 0.6% less top-1 error, 35.7% fewer FLOPs, and 33.5% fewer parameters than the original model. As for deeper networks, our integrated Sc-SE-ResNet-101 achieves almost 0.8% better accuracy, 38.4% computational cost, and 34.9% parameters compared with SE-ResNet-101. It is worth mentioning that Sc-ResNet-50 already has better performance than SE-ResNet-101.

D. Compressibility

Our SlimConv is a plug-and-play module, which can be easily integrated into CNNs to effectively compress models by only changing one hyperparameter (k). We conduct several experiments with different values of k. We report the results in Fig. 8. As k increases, the compression ratio also increases with slight drops of the performance. It suggests that the k

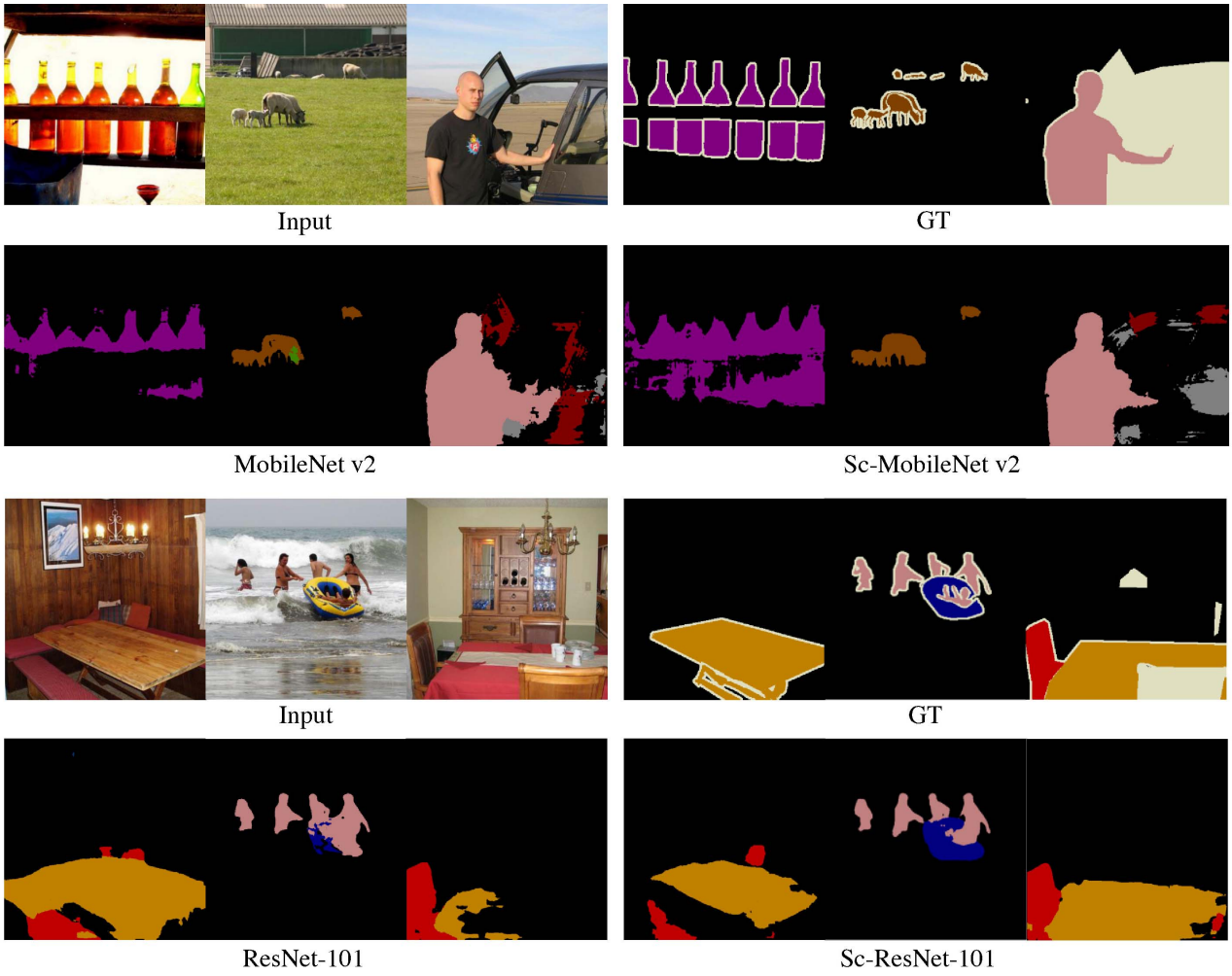


Fig. 7. Visual comparison on semantic segmentation for MobileNet v2 [7], ResNet [1] and our integrated models on Pascal VOC2012.

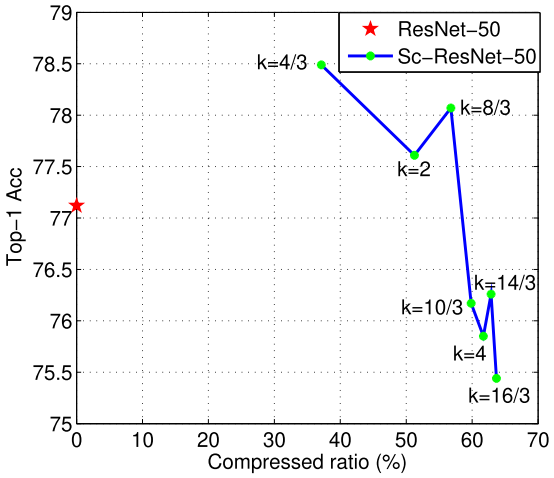


Fig. 8. Results of Top-1 accuracy and compressed ratio for our Sc-ResNet-50 with different hyperparameters (k) on CIFAR-100.

is a trade-off value tuned for different applications according to the computational resources. Specifically, our integrated Sc-ResNet-50 can still achieve nearly 1% better accuracy than the basic model with k set to $\frac{8}{3}$ while reducing parameters by 56.77%.

TABLE VII
PERFORMANCE COMPARISON ON SEMANTIC SEGMENTATION FOR MOBILENET V2 [7], RESNET [1] AND OUR INTEGRATED MODELS ON PASCAL VOC2012

Backbone	Mean IoU (%)	FLOPs(10^9)	Params(10^6)
1.0 MobileNet(v2)	**69.3**	14.20	5.23
1.0 Sc-MoblieNet(v2)(ours)	69.2	**13.80**	**5.10**
ResNet-50	76.4	62.60	39.76
Sc-ResNet-50(ours)	**76.8**	**50.18**	**30.96**
ResNet-101	77.8	83.31	58.75
Sc-ResNet-101(ours)	**78.5**	**62.32**	**42.16**

E. Semantic Segmentation

Figure 5 also shows that our module can strengthen networks for precisely localizing the region of objects. This characteristic makes our SlimConv has the potential to improve the performance of models in the semantic segmentation task.

We replace the backbone network of Deeplab v3+ [15] with MobileNet [7], ResNet [1], and our integrated models and conduct three groups of comparisons. The results are shown in Table VII. In group 1, our Sc-MobileNet achieves

TABLE VIII

DETECTION PERFORMANCE ON PASCAL VOC2007 DATASET [18]. NOTE THAT FLOPS AND PARAMS ARE CALCULATED WHEN THE SIZE OF THE INPUT IMAGE IS 850 × 600

Backbone	mAP	FLOPs(10^9)	Params(10^6)
ResNet-101	75.0	148.93	45.26
Sc-ResNet-101(ours)	**75.2**	**98.16**	**29.44**

TABLE IX

DETECTION PERFORMANCE ON COCO2014 MINIVAL DATASET [17]. NOTE THAT FLOPS AND PARAMS ARE CALCULATED WHEN THE SIZE OF THE INPUT IMAGE IS 600 × 899

Backbone	AP@(IoU=0.50:0.95)	FLOPs(10^9)	Params(10^6)
ResNet-101	33.9	153.59	45.86
Sc-ResNet-101(ours)	**35.3**	**101.36**	**30.04**

slightly worse mean IoU but fewer FLOPs and parameters than MobileNet [7]. As shown in Fig. 7, the SlimConv equipped model tends to handle occluded scenes correctly. In group 2, our Sc-ResNet-50 outperforms its baseline by 0.4% on mean IoU and also achieves almost 20% fewer FLOPs and over 22% fewer parameters than ResNet-50. In the last group, our integrated Sc-ResNet-101 achieves 0.7% better mean IoU, 25.2% less computational cost, and 28.2% less storage than the reference ResNet-101. As illustrated in Fig. 7, our Sc-ResNet-101 has the closest results to ground-truth while the coverage of objects is either more or less for ResNet-101 based results.

F. Object Detection

We evaluate our module on two kinds of the dataset for the object detection task. The results are shown in Table VIII and Table IX. We adopt the backbone network of the widely used ResNet-101 [1] v.s. our Sc-ResNet-101. Our SlimConv based model achieves 0.2% better average precision than the original ResNet-101, reducing FLOPs by 34.1% and parameters by 35% on the PASCAL VOC2007 dataset [18]. On the other popular dataset COCO2014 [17], our Sc-ResNet-101 achieves 1.4% better average precision with over 52G computational cost decreased and 34.5% fewer parameters than ResNet-101.

G. Ablation Studies

To explore the effectiveness of our different design choice, we conduct ablation studies on the ImageNet dataset [12] with ResNet-50 [1] as the baseline. The experimental results are shown in Table X.

Firstly, we drop the flipping operation and make the bottom pathway use the same weights as the top pathway. We find that Sc-ResNet-50 without flipping can still achieve 0.14% better accuracy than the baseline. It shows that our method possesses strong robustness. Secondly, we replace the learned weights w with the flipped weights $\breve{w}$ so that all weights used are flipped. In this case, the SlimConv with only flipped weights makes a little worse accuracy than the baseline. The first case converges faster but also over-fits earlier than the second one during training. Hence, our proposed model

TABLE X

PERFORMANCE COMPARISON FOR OUR SC-RESNET-50 WITH DIFFERENT SETTINGS ON IMAGENET DATASET. FOR @256, WE DIRECTLY CHANGE THE SIZE OF INPUT IMAGE FROM 224 × 224 TO 256 × 256 ON IMAGENET

Model	Top-1 Error	FLOPs(10^9)	Params(10^6)
ResNet-50	23.85	4.09	25.56
ResNet-50@256	23.28	5.34	25.56
Sc-ResNet-50(w/o flipping)	23.71	2.65	16.76
Sc-ResNet-50(only flipping)	23.93	2.65	16.76
Sc-ResNet-50	23.29	2.65	16.76
Sc-ResNet-50(cosine)	22.77	2.65	16.76
Sc-ResNet-50@256	**22.52**	3.47	16.76

TABLE XI

PERFORMANCE COMPARISON FOR OUR SC-RESNET-50 WITH DIFFERENT SETTINGS ON CIFAR-100

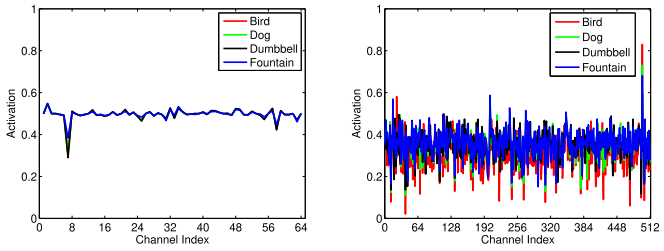
Model	Top-1 Error	FLOPs(10^9)	Params(10^6)
ResNet-50	22.88	1.31	23.71
Sc-ResNet-50	**20.63**	**0.84**	**14.91**
Sc-ResNet-50-sf	21.25	**0.84**	**14.91**
Sc-ResNet-50-3b-sf	20.73	0.96	17.03

outperforms other settings because of the weights flipping. Furthermore, we try to use the cosine learning rate. Our Sc-ResNet-50(cosine) achieves better results than before. Last but not least, we increase the size of the input image to 256 × 256 and test the performance of the pre-trained models. Our Sc-ResNet-50@256 achieves 0.76% better accuracy and almost 35% less computational cost than ResNet-50@256 as illustrated in Table X.

Finally, we further study the effectiveness of our module with other settings. The experimental results are shown in Table XI. It is demonstrated that the accuracy downgrades about 0.6% with the replacement to the shuffling weights (Sc-ResNet-50-sf), while the original flipping weights method works better. Then we add another branch with shuffling learned weights to our module (Sc-ResNet-50-3b-sf). The computational costs apparently increase with adding more branches while the performance even drops.

H. Discussion

We analyze the learned weights by inputting different objects to investigate how information of features is compressed. The results are shown in Fig. 9. At the first block of the stage, the values of output weights are varied from 0.2 to 0.6, and they are almost stable. It is hard to distinguish between different classes. Then through flipping these weights, a small part of weights at the same index changes large. It suggests that the weighted input features of the two pathways in our module are similar to each other at the same index. However, with the layers increase, especially at the last block named Sc_5_3, the weights of different objects are more diversified and easier to be distinguished. It indicates that the two weighted input features start to differ from the corresponding index and gradually diverse by the splitting and summation operation.



(a) Sc_2_1 (b) Sc_5_3

Fig. 9. Last activations(w) of the SE-module in our Sc-ResNet-50 on ImageNet dataset. Two set of activations are named by following the scheme: *Sc_stageID_blockID*.

V. Conclusion

In this paper, we have designed a novel SlimConv module, an efficient architectural unit to decrease computational cost and model storage while improving the performance of deep CNN models by reducing channel redundancies. The SlimConv consists of three steps, namely Reconstruct, Transform, and Fuse. A weights flipping operation has been proposed, which can generate richer features. The extensive experiments on multiple challenging tasks have shown the effectiveness of our SlimConv. The existing state-of-the-art methods integrated with SlimConv possess performance improvements while reducing the computations and saving the storage. In addition, the discussion section has indicated that the SlimConv equipped models have the potentials for further compression. Finally, we hope our proposed method can inspire the research for more efficient architectural design.

References

[1] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. CVPR*, Jun. 2016, pp. 770–778.
[2] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proc. CVPR*, Jun. 2018, pp. 7132–7141.
[3] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *Proc. ICML*, 2010, pp. 807–814.
[4] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," 2015, *arXiv:1502.03167*. [Online]. Available: http://arxiv.org/abs/1502.03167
[5] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Proc. CVPR*, Jul. 2017, pp. 1492–1500.
[6] F. Yu, D. Wang, E. Shelhamer, and T. Darrell, "Deep layer aggregation," in *Proc. CVPR*, Jun. 2018, pp. 2403–2412.
[7] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *Proc. CVPR*, Jun. 2018, pp. 4510–4520.
[8] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun, "ShuffleNet V2: Practical guidelines for efficient CNN architecture design," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 116–131.
[9] C.-F. Chen, Q. Fan, N. Mallinar, T. Sercu, and R. Feris, "Big-little net: An efficient multi-scale feature representation for visual and speech recognition," 2018, *arXiv:1807.03848*. [Online]. Available: http://arxiv.org/abs/1807.03848
[10] X. Li, W. Wang, X. Hu, and J. Yang, "Selective kernel networks," in *Proc. CVPR*, Jun. 2019, pp. 510–519.
[11] A. Paszke *et al.*, "Pytorch: An imperative style, high-performance deep learning library," in *Proc. NIPS*, 2019, pp. 8024–8035.
[12] O. Russakovsky *et al.*, "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, Dec. 2015.
[13] C. Szegedy *et al.*, "Going deeper with convolutions," in *Proc. CVPR*, Jun. 2015, pp. 1–9.
[14] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The Pascal visual object classes challenge: A retrospective," *Int. J. Comput. Vis.*, vol. 111, no. 1, pp. 98–136, Jan. 2015.
[15] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with Atrous separable convolution for semantic image segmentation," in *Proc. ECCV*, 2018, pp. 801–818.
[16] B. Hariharan, P. Arbeláez, L. Bourdev, S. Maji, and A. Malik, "Semantic contours from inverse detectors," in *Proc. ICCV*, 2011, pp. 991–998.
[17] T.-Y. Lin *et al.*, "Microsoft COCO: Common objects in context," in *Proc. ECCV*, 2014, pp. 740–755.
[18] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The Pascal visual object classes (VOC) challenge," *Int. J. Comput. Vis.*, vol. 88, no. 2, pp. 303–338, Jun. 2010.
[19] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Proc. NIPS*, 2015, pp. 91–99.
[20] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-CAM: Visual explanations from deep networks via gradient-based localization," in *Proc. ICCV*, 2017, pp. 618–626.
[21] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. CVPR*, Jul. 2017, pp. 4700–4708.
[22] J. Yu and T. Huang, "AutoSlim: Towards one-shot architecture search for channel numbers," 2019, *arXiv:1903.11728*. [Online]. Available: http://arxiv.org/abs/1903.11728
[23] Y. Chen *et al.*, "Drop an octave: Reducing spatial redundancy in convolutional neural networks with octave convolution," in *Proc. ICCV*, Oct. 2019, pp. 3435–3444.
[24] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size," 2016, *arXiv:1602.07360*. [Online]. Available: http://arxiv.org/abs/1602.07360
[25] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. NIPS*, 2012, pp. 1097–1105.
[26] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*. [Online]. Available: http://arxiv.org/abs/1409.1556
[27] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proc. CVPR*, Jul. 2017, pp. 1251–1258.
[28] S. Han *et al.*, "DSD: Dense-sparse-dense training for deep neural networks," 2016, *arXiv:1607.04381*. [Online]. Available: http://arxiv.org/abs/1607.04381
[29] 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 Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 10, pp. 2525–2538, Oct. 2019.
[30] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient ConvNets," 2016, *arXiv:1608.08710*. [Online]. Available: http://arxiv.org/abs/1608.08710
[31] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," in *Proc. NIPS*, 2015, pp. 1135–1143.
[32] Y. He, X. Zhang, and J. Sun, "Channel pruning for accelerating very deep neural networks," in *Proc. CVPR*, Oct. 2017, pp. 1389–1397.
[33] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *Proc. CVPR*, Jun. 2018, pp. 8697–8710.
[34] C. Liu *et al.*, "Progressive neural architecture search," in *Proc. ECCV*, 2018, pp. 19–34.
[35] M. Tan *et al.*, "MnasNet: Platform-aware neural architecture search for mobile," in *Proc. CVPR*, Jun. 2019, pp. 2820–2828.
[36] M. Tan and Q. V. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," 2019, *arXiv:1905.11946*. [Online]. Available: http://arxiv.org/abs/1905.11946
[37] M. Tan, R. Pang, and Q. V. Le, "EfficientDet: Scalable and efficient object detection," 2019, *arXiv:1911.09070*. [Online]. Available: http://arxiv.org/abs/1911.09070
[38] J. Yu, L. Yang, N. Xu, J. Yang, and T. Huang, "Slimmable neural networks," 2018, *arXiv:1812.08928*. [Online]. Available: http://arxiv.org/abs/1812.08928
[39] J. Yu and T. S. Huang, "Universally slimmable networks and improved training techniques," in *Proc. CVPR*, Oct. 2019, pp. 1803–1811.
[40] T.-W. Ke, M. Maire, and S. X. Yu, "Multigrid neural architectures," in *Proc. CVPR*, Jul. 2017, pp. 6665–6673.
[41] S.-H. Gao, M.-M. Cheng, K. Zhao, X.-Y. Zhang, M.-H. Yang, and P. Torr, "Res2Net: A new multi-scale backbone architecture," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 2, pp. 652–662, Feb. 2021.

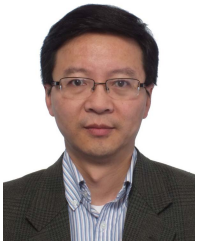
[42] D.-P. Fan, W. Wang, M.-M. Cheng, and J. Shen, "Shifting more attention to video salient object detection," in *Proc. CVPR*, Jun. 2019, pp. 8554–8564.
[43] J. Qiu *et al.*, "DeepLiDAR: Deep surface normal guided depth prediction for outdoor scene from sparse LiDAR data and single color image," in *Proc. CVPR*, Jun. 2019, pp. 3313–3322.
[44] Y. Zhang, K. Li, K. Li, L. Wang, B. Zhong, and Y. Fu, "Image super-resolution using very deep residual channel attention networks," in *Proc. ECCV*, 2018, pp. 286–301.
[45] P. D. M. Fernandez, F. A. G. Pena, T. I. Ren, and A. Cunha, "FERAtt: Facial expression recognition with attention net," in *Proc. CVPRW*, Jun. 2019, pp. 1–10.
[46] F. Wang *et al.*, "Residual attention network for image classification," in *Proc. CVPR*, Jul. 2017, pp. 3156–3164.
[47] S. Woo, J. Park, J.-Y. Lee, and I. S. Kweon, "CBAM: Convolutional block attention module," in *Proc. ECCV*, 2018, pp. 3–19.
[48] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "XNOR-Net: ImageNet classification using binary convolutional neural networks," in *Proc. ECCV*, 2016, pp. 525–542.
[49] A. Krizhevsky *et al.*, "Learning multiple layers of features from tiny images," Univ. Toronto, Toronto, ON, Canada, Tech. Rep., 2009, p. 7, vol. 1, no. 4.
[50] K. Han, Y. Wang, Q. Tian, J. Guo, C. Xu, and C. Xu, "GhostNet: More features from cheap operations," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 1580–1589.
[51] Q. Zhang *et al.*, "Split to be slim: An overlooked redundancy in vanilla convolution," 2020, *arXiv:2006.12085*. [Online]. Available: http://arxiv.org/abs/2006.12085
[52] S. Guo, Y. Wang, Q. Li, and J. Yan, "DMCP: Differentiable Markov channel pruning for neural networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 1539–1547.
[53] Z. Huang and N. Wang, "Data-driven sparse structure selection for deep neural networks," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 304–320.
[54] P. Molchanov, A. Mallya, S. Tyree, I. Frosio, and J. Kautz, "Importance estimation for neural network pruning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 11264–11272.
[55] Z. Liu *et al.*, "MetaPruning: Meta learning for automatic neural network channel pruning," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 3296–3305.
[56] X. Ding, G. Ding, Y. Guo, and J. Han, "Centripetal SGD for pruning very deep convolutional networks with complicated structure," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 4943–4953.
[57] X. Ding, Y. Guo, G. Ding, and J. Han, "ACNet: Strengthening the kernel skeletons for powerful CNN via asymmetric convolution blocks," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 1911–1920.
[58] X. Ding, G. Ding, Y. Guo, J. Han, and C. Yan, "Approximated oracle filter pruning for destructive CNN width optimization," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 1607–1616.
[59] J. Gu *et al.*, "Projection convolutional neural networks for 1-bit CNNs via discrete back propagation," in *Proc. AAAI Conf. Artif. Intell.*, vol. 33, 2019, pp. 8344–8351.
[60] X. Ding, G. Ding, J. Han, and S. Tang, "Auto-balanced filter pruning for efficient convolutional neural networks," in *Proc. AAAI Conf. Artif. Intell.*, vol. 32, 2018, pp. 1–8.

Jiaxiong Qiu (Student Member, IEEE) received the bachelor's degree from Dalian Maritime University in 2017. He is currently pursuing the master's degree with the University of Electronic Science and Technology of China. His research interests include computer vision, computer graphics, robotics, and deep learning.

Cai Chen (Student Member, IEEE) received the B.E. degree in network engineering from the University of Electronic Science and Technology of China (UESTC) in 2019. He is currently pursuing the master's degree with the School of Information and Communication Engineering, Institute of Image Processing, UESTC. His research interests lie in computer vision, computational photography, and 3D scene perception.

Shuaicheng Liu (Member, IEEE) received the B.E. degree from Sichuan University, Chengdu, China, in 2008, and the M.S. and Ph.D. degrees from the National University of Singapore, Singapore, in 2010 and 2014, respectively. Since 2014, he has been an Associate Professor with the School of Information and Communication Engineering, Institute of Image Processing, University of Electronic Science and Technology of China (UESTC). His research interests include computer vision and computer graphics.

Heng-Yu Zhang received the master's and Ph.D. degrees from the West China Medicine School, Sichuan University, in 1996 and 2010, respectively. From February 2008 to February 2009, he was a Visiting Scientist with the School of Medicine, National University of Singapore. Since 2021, he has been the Director of the West China Syncope Center, where medical images are heavily involved. He is currently a professor with the school and the Deputy Chief Physician of the West China Hospital, engaging cardiac pacing, electrophysiology, and pacemaker implantation with international leading level.



Bing Zeng (Fellow, IEEE) received the B.Eng. and M.Eng. degrees in electronic engineering from the University of Electronic Science and Technology of China (UESTC), Chengdu, China, in 1983 and 1986, respectively, and the Ph.D. degree in electrical engineering from the Tampere University of Technology, Tampere, Finland, in 1991. He was a Postdoctoral Fellow with the University of Toronto from 1991 to 1992 and as a Researcher with Concordia University from 1992 to 1993. He then joined The Hong Kong University of Science and Technology (HKUST). After 20 years of service at HKUST, he returned to UESTC in 2013, through China's 1000-Talent-Scheme. At UESTC, he leads the Institute of Image Processing to focus on image and video processing, 3D and multiview video technology, and visual big data. During his tenure at HKUST and UESTC, he has graduated over 30 master's and Ph.D. students, received about 20 research grants, filed eight international patents, and authored or coauthored over 250 articles. He received the Class Natural Science Award (the first recipient) from the Chinese Ministry of Education in 2014 and elected as an IEEE Fellow in 2016 for contributions to image and video coding. He was the General Co-Chair of IEEE VCIP-2016, Chengdu, in 2016. He served as an Associate Editor for the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY for eight years and received the Best Associate Editor Award in 2011. He is currently on the Editorial Board of the *Journal of Visual Communication and Image Representation* and serves as the General Co-Chair of PCM-2017.