

# IGCV3: Interleaved Low-Rank Group Convolutions for Efficient Deep Neural Networks

Ke Sun<sup>1</sup>

sunk@mail.ustc.edu.cn

Mingjie Li<sup>1</sup>

homles11@mail.ustc.edu.cn

Dong Liu<sup>1</sup>

dongeliu@ustc.edu.cn

Jingdong Wang<sup>2</sup>

jingdw@microsoft.com

<sup>1</sup> University of Science and Technology of China  
Anhui, China

<sup>2</sup> Microsoft Research Asia,  
Beijing, China

## Abstract

In this paper, we are interested in building lightweight and efficient convolutional neural networks. Inspired by the success of two design patterns, composition of structured sparse kernels, e.g., interleaved group convolutions (IGC), and composition of low-rank kernels, e.g., bottle-neck modules, we study the combination of such two design patterns, using the composition of structured sparse low-rank kernels, to form a convolutional kernel. Rather than introducing a complementary condition over channels, we introduce a loose complementary condition, which is formulated by imposing the complementary condition over super-channels, to guide the design for generating a dense convolutional kernel. The resulting network is called IGCV3. We empirically demonstrate that the combination of low-rank and sparse kernels boosts the performance and the superiority of our proposed approach to the state-of-the-arts, IGCV2 and MobileNetV2 over image classification on CIFAR and ImageNet and object detection on COCO. Code and models are available at <https://github.com/homles11/IGCV3>.

## 1 Introduction

There have been imperative demands for portable and efficient deep convolutional neural networks with high accuracies in vision applications. The recent efforts include two paths, network compression: approximate pre-trained models by pruning superfluous connections and channels or decomposing convolutional matrices; and lightweight network design: design less-redundant kernels for constructing networks trained from scratch.

We are interested in designing lightweight networks using less-redundant kernels to form a dense convolutional kernel. There are two main representative schemes, either of which has independently shown the success in building small networks. One scheme is to use a sequence of low-rank convolutional kernels to compose a linear kernel or nonlinear kernel with intermediate nonlinear activations included in the sequence of low-rank kernels, e.g.,

bottleneck [10] and inverted residual block [6]. The other scheme is to use a sequence of sparse (and possible dense) kernels to compose a kernel, e.g., interleaved group convolutions, MobileNetV1 and Xception, and deep roots [2, 22, 16, 41].

In this paper, we design a modularized convolutional block, which simultaneously explores both low-rank and sparse kernels to compose a dense convolutional kernel. We start from IGCv2 [39], which is composed of a channel-wise spatial convolution, group point-wise convolutions, and intermediate permutation operations. We introduce the low-rank design pattern into group point-wise convolutions, where one group convolution expands the feature dimension, and the other group convolution projects the feature back. We introduce a *loose* complementary condition over channels for constructing *dense* composed kernels, which is equivalent to the *strict* complementary condition, presented in IGCv2 [39], imposed over super-channels (a set of channels), so as to handle the case that the number of input and output channels are different and avoid over-wide convolutions. The resulting network is called IGCv3. We empirically demonstrate that the combination of low-rank and sparse kernels boosts the performance and the superiority of our proposed approach to the state-of-the-arts, IGCv2 and MobileNetV2, over image classification on CIFAR and ImageNet and object detection on COCO.

## 2 Related Work

To obtain portable and efficient models, most of works devote great efforts to reduce the redundancy in the convolutional kernels, which mainly lie in two extents: high numerical precision and superfluous weights.

**Low-precision kernels.** Approaches in this field quantize the weights of CNNs from floating point into lower bit-depth representations. These methodologies reduce the redundancies in the numerical precision, such as binarization [9] constraining the weights to either  $+1$  or  $-1$ , trinarization [21, 45, 46] making weights to be ternary-valued and quantization [6, 44] a general form to convert weights into a low-precision version.

**Sparse or Low-rank kernels.** In this field, the methods adopt one sparse or low-rank kernel as an approximation of the original dense or high-rank kernel. (i) *Sparse kernels*: Many regularizations are designed to adding the sparse constraint on the convolutional kernels, such as  $L1$  or  $L2$  regularization [6, 7] and structured sparsity regularizer [11, 22, 38]. Group convolution is a pre-defined structured-sparse matrix, which is widely used in the mobile models [40, 43]. (ii) *Low-rank kernels*. Pruning redundant weights or channels from pre-trained models [11, 22, 26, 28] is a main branch in network compression. What's more, many works [8, 33, 36] replace a large kernel with small kernels is to reduce the ranks in the spatial domain.

**Composition from multiple sparse or low-rank kernels.** Design the composition of sparse or low-rank kernels to keep the original dense connections. (i) *Multiplying the sparse kernels*: IGCv1 [40] is the product of two structured-sparse matrices and IGCv2 [39] decomposes the dense kernel into more sparse kernels. And the permutation operations [41] are adopted to keep dense connectivity between input and output channels, which can increase the expressive capacity of networks [32]. Xception [2] is an extreme case of IGCv1: a pointwise convolution followed by a channel-wise convolution. (ii) *Composition from low-rank kernels*. In a clear case, a  $3 \times 3$  kernel can be decomposed into a  $3 \times 1$  convolution followed by a  $1 \times 3$  convolution [15, 17, 27], which approximates the dense kernel along the spatial domain. Bottleneck [10, 24, 6], if neglecting the intermediate ReLUs, can be

viewed as the low-rank approximation along the output channel domain.

### 3 Our Approach

In this section, we firstly review prior works, IGC and MobileNets, and then describe our proposed IGCV3. Finally, we analyze the loose complementary condition and give a discussion on the architecture of IGCV3 blocks.

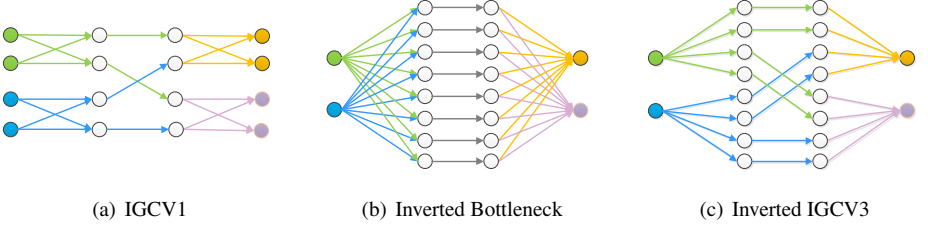


Figure 1: The architectures of different blocks. Each circle denotes a channel and an arrow denotes a forward path. A set of arrows between two columns of circles can be viewed as a weight matrix. (a) IGCV1, the first and third matrices are sparse; (b) Inverted Bottleneck, the first and third matrices are low-rank; (c) Inverted IGCV3, the first and third matrices are sparse and low-rank. The second permutation in IGCV1 and IGCV3 is not shown here. Bottleneck and IGCV3 have the skip connection between the input and output.

#### 3.1 Prior Works

**Interleaved Group Convolution (IGCV1).** The IGCV1 block consists of primary and secondary group convolutions, which is mathematically formulated as follows:

$$\mathbf{y} = \mathbf{P}^2 \mathbf{W}^2 \mathbf{P}^1 \mathbf{W}^1 \mathbf{x}. \quad (1)$$

Here, the input response maps include  $C$  channels and each channel has  $K$  pixels. Let  $\mathbf{x} \in \mathbb{R}^{(K \times C) \times 1}$  denote the corresponding input vector and  $\mathbf{y} \in \mathbb{R}^{C \times 1}$  denotes the output vector. The primary convolution is a group spatial convolution, whose kernel size is  $K$ , e.g.,  $K = 9$  for  $3 \times 3$  kernels.  $\mathbf{P}^1$  and  $\mathbf{P}^2$  are permutation matrices [41]. The kernel matrices  $\mathbf{W}^1$  and  $\mathbf{W}^2$  are block-wise sparse,

$$\mathbf{W}^i = \begin{bmatrix} \mathbf{W}_1^i & 0 & 0 & 0 \\ 0 & \mathbf{W}_2^i & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \mathbf{W}_{G_i}^i \end{bmatrix}, \mathbf{x} = \begin{bmatrix} \mathbf{x}^1 \\ \mathbf{x}^2 \\ \vdots \\ \mathbf{x}^C \end{bmatrix}, \quad (2)$$

where  $\mathbf{x}^c = [x_1^c, x_2^c, \dots, x_K^c]$  is a column vector.  $\mathbf{W}_g^i$  ( $i = 1$  or  $2$ ) is the kernel matrix over the corresponding channels in the  $g$ th branch,  $G_i$  is the number of branches in the  $i$ th group convolution. In the case suggested in [41], the primary group convolution is a group  $3 \times 3$  convolution with  $G_1 = \frac{C}{2}$  groups, and  $\mathbf{W}_g^1$  is a matrix of size  $2 \times (2K)$ . The secondary group

convolution is a group  $1 \times 1$  convolution with  $G_2 = 2$  groups, where  $\mathbf{W}_1^2$  and  $\mathbf{W}_2^2$  are both dense matrices of size  $\frac{C}{2} \times \frac{C}{2}$ .

**Interleaved Structured Sparse Convolution (IGCV2).** IGCV2 extends IGCV1 by decomposing the convolution matrix into more structured sparse matrices:

$$\mathbf{y} = \mathbf{P}_L \mathbf{W}_L \mathbf{P}_{L-1} \mathbf{W}_{L-1} \dots \mathbf{P}_1 \mathbf{W}_1 \mathbf{x} \quad (3)$$

$$= \left( \prod_{l=L}^1 \mathbf{P}_l \mathbf{W}_l \right) \mathbf{x}. \quad (4)$$

Here  $\mathbf{W}_1$  corresponds to a channel-wise spatial convolution, and  $\mathbf{W}_l, l \in \{1, 2, \dots, L\}$  corresponds to group point-wise convolutions.

**MobileNetV1.** A MobileNetV1 block consists of a channel-wise spatial convolution and a point-wise convolution. The mathematical formulation is given as follows:

$$\mathbf{y} = \mathbf{W}^2 \mathbf{W}^1 \mathbf{x}, \quad (5)$$

where  $\mathbf{W}^1$  and  $\mathbf{W}^2$  corresponds to the channel-wise and point-wise convolution respectively. It is an extreme case of IGCV1 [14]: both channel-wise and point-wise convolutions are extreme group convolutions.

**MobileNetV2.** A MobileNetV2 block consists of a dense pointwise convolution, a channel-wise spatial convolution, and a dense pointwise convolution. It uses an inverted bottleneck: the first pointwise convolution increases the width and the second one reduces the width.

For convenience, we convert the input  $\mathbf{x}$  to  $\hat{\mathbf{x}}$ , and accordingly the bottleneck is mathematically formulated:

$$\mathbf{y} = \mathbf{W}^2 \mathbf{W}^1 \mathbf{W}^0 \hat{\mathbf{x}}, \quad (6)$$

$$\mathbf{W}^0 = \begin{bmatrix} \hat{\mathbf{w}}_{1,1}^T & 0 & 0 & 0 \\ 0 & \hat{\mathbf{w}}_{1,2}^T & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \hat{\mathbf{w}}_{C_{int},K}^T \end{bmatrix}, \hat{\mathbf{x}} = \begin{bmatrix} \hat{\mathbf{x}}_{1,1} \\ \hat{\mathbf{x}}_{1,2} \\ \vdots \\ \hat{\mathbf{x}}_{C_{int},K} \end{bmatrix}, \quad (7)$$

where  $\mathbf{W}^1 \in \mathbb{R}^{C_{int} \times (K \times C_{int})}$  corresponds to the channel-wise  $3 \times 3$  convolution, the kernel including  $K = 9$  spatial positions, and  $\mathbf{W}^0 \in \mathbb{R}^{(K \times C_{int}) \times (K \times C_{int} \times C)}$  and  $\mathbf{W}^2 \in \mathbb{R}^{C \times C_{int}}$  are two low-rank matrices, which correspond to the two dense point-wise convolutions.  $C_{int}$  represents the intermediate width.  $\hat{\mathbf{x}} \in \mathbb{R}^{(K \times C_{int} \times C) \times 1}$  is a column vector and  $\hat{\mathbf{x}}_{j,k} = [x_k^1, x_k^2, \dots, x_k^C]$ ,  $j \in \{1, 2, \dots, C_{int}\}$  is the  $k$ th spatial position across all the input channels for the  $j$ th output channel.  $\hat{\mathbf{w}}_{j,k} = [w_j^1, w_j^2, \dots, w_j^C]$ ,  $k \in \{1, 2, \dots, K\}$  represents the  $j$ th kernel of the first convolution, which is shared by  $K$  spatial positions.

### 3.2 Interleaved Low-Rank Group Convolutions

The proposed Interleaved Low-Rank Group Convolutions, named IGCV3, extend IGCV2 by using low-rank group convolutions to replace group convolutions in IGCV2. It consists of a channel-wise spatial convolution, a low-rank group point-wise convolution with  $G_1$  groups that reduces the width and a low-rank group point-wise convolution with  $G_2$  groups which expands the width back. The mathematical formulation is given as follows,

$$\mathbf{y} = \mathbf{P}^2 \mathbf{W}^2 \mathbf{P}^1 \mathbf{W}^1 \hat{\mathbf{W}}^0 \hat{\mathbf{x}}. \quad (8)$$

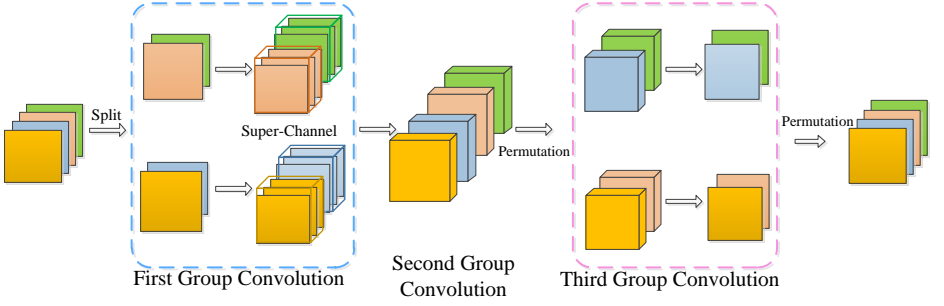


Figure 2: Illustrating the interleaved branches in IGCv3 block. The first group convolution is a group  $1 \times 1$  convolution with  $G_1 = 2$  groups. The second is a channel-wise spatial convolution. The third is a group  $1 \times 1$  convolution with  $G_2 = 2$  groups.

Here,  $\mathbf{P}^1$  and  $\mathbf{P}^2$  are permutation matrices similar to permutation matrices given in [44].  $\mathbf{W}^1$  corresponds to the channel-wise  $3 \times 3$  convolution.  $\hat{\mathbf{W}}^0$  and  $\mathbf{W}^2$  are low-rank structured sparse matrices. The two low-rank sparse matrices are mathematically formulated as follows,

$$\hat{\mathbf{W}}^0 = \begin{bmatrix} \hat{\mathbf{w}}_{1,1}^1 & 0 & 0 & 0 \\ 0 & \hat{\mathbf{w}}_{1,2}^1 & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \hat{\mathbf{w}}_{C_{int},K}^{G_1} \end{bmatrix}, \mathbf{W}^2 = \begin{bmatrix} \mathbf{W}_1^2 & 0 & 0 & 0 \\ 0 & \mathbf{W}_2^2 & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \mathbf{W}_{G_2}^2 \end{bmatrix}. \quad (9)$$

Here,  $\hat{\mathbf{w}}_{j,k}^g \in \mathbb{R}^C$ , a row vector including  $\frac{C}{G_1}$  non-zero weights, corresponds to the kernels for the  $g$ th branch in the first group convolution, e.g.  $\hat{\mathbf{w}}_{j,k}^1 = [w_j^1, w_j^2, \dots, w_j^{\frac{C}{G_1}}, 0, \dots, 0]^T$ , which increases the number of channels.  $\mathbf{W}_g^2$  corresponds to the  $g$ th branch in the second group convolution, which reduces the width back to the initial width.

**Construct a dense composed kernel.** Similar to IGCv2, we also aim to design the two group convolutions such that the composed pointwise convolutional kernel,  $\mathbf{P}^2 \mathbf{W}^2 \mathbf{P}^1 \mathbf{W}^1$ , is dense. Different from IGCv2, the number of input and output channels in the low-rank group convolution are not the same, so the permutation operation cannot work as before.

We divide input, output and intermediate channels into a set of (e.g.,  $C_s$ ) super-channels, where the super-channel for input and output response maps contains  $\frac{C}{C_s}$  channels and for intermediate response maps the super-channel contains  $\frac{C_{int}}{C_s}$  channels. The sketch is shown in Fig 2, where super-channels are represented as 3-D boxes in different colors. Then, we review the *strict* complementary condition proposed in [89] and describe the *loose* complementary condition based on the concept of super-channel.

**Condition 1 (Strict complementary condition)** *The two group convolutions are thought complementary if the channels in the input and output response maps and in the intermediate response maps lying in the same branch in one group convolution lie in different branches and come from all the branches in the other group convolution.*

**Loose complimentary condition.** We empirically observed that over-sparse convolutions lead to over-wide response maps and thus too large memory cost, but do not lead to higher accuracy, which is consistent to the empirical results in IGCv1 [44] and IGCv2 [49]. Therefore, we relax the strict complementary condition to a loose version.

**Condition 2 (Loose complementary condition)** *The two group convolutions are thought complementary if the super-channels lying in the same branch in one group convolution lie in different branches and come from all the branches in the other group convolution.*

### 3.3 Discussions and Analysis

**Linear, nonlinear, and inverted IGCv3.** It is shown in IGCv1 [44] that the linear version, i.e., no intermediate ReLU is included in the block, performs better than the very deep networks which include lots of ReLU activations. Similar observations are also obtained in the bottleneck block [44] and Xception [9]. On the other hand, for not very deep networks, e.g., there are only 10+ IGCv3 blocks, the non-linear version, i.e., there are one or more intermediate ReLUs in the block, performs better. We provide the ablation study in Sec. 5.

In our experiments, we do not see difference between the normal version and the inverted version of bottleneck. We adopt the inverted IGCv3 block to save the memory footprints during the training and inference process (shown in Fig 1(c)): low-rank group convolution (width increasing), channel-wise spatial convolution, and low-rank group convolution (width reduction) and accordingly the identity connection is built between two low-dimensional representations, which is similar to inverted bottleneck presented in MobileNetV2 [8].

**Wider and deeper IGCv3 networks.** We stack our IGCv3 blocks to form a deep network and provide two versions for fair comparisons with MobileNetV2. One is to let the depth (the number of blocks) be the same and to widen the network. In our implementation,  $C_s = 4$  super-channels is used to form group convolutions ( $G_1 = 4$  and  $G_2 = 4$ ). The other one is to let the width (of the corresponding block) be the same and to deepen the network. In our implementation,  $C_s = 2$  super-channels is used ( $G_1 = 2$  and  $G_2 = 2$ ). The empirical study is given in Sec. 5.

## 4 Experiments

### 4.1 Datasets

**CIFAR.** The CIFAR datasets, CIFAR-10 [8] and CIFAR-100 [67], are subsets of the 80 million tiny images. Both datasets contain 60000  $32 \times 32$  color images with 50000 images for training and 10000 images for test. The standard data augmentation scheme we adopt is widely used for these datasets [8, 14, 19, 20, 23, 30, 34, 35]: we zero-pad the images with 4 pixels on each side, and then randomly crop them to produce  $32 \times 32$  images, followed by horizontally mirroring half of the images and normalizing them by using the channel means.

For training, we use the SGD algorithm and train all networks from scratch. We initialize the weights similar to [8, 9], and set the weight decay as 0.0001 and the momentum as 0.9. The initial learning rate is 0.1 and is reduced by a factor 10 at the 200, 300 and 350 training epochs. Each training batch consists of 64 images on four asynchronous GPUs.

**ImageNet.** The ILSVRC 2012 classification dataset [9] contains over 1.2 million training images and 50,000 validation images, and each image is labeled from 1000 categories. We adopt the same data augmentation scheme as in [8, 9]. We use SGD to train the networks

with the same hyperparameters (batch size = 96, weight decay = 0.00004 and momentum = 0.9) on 4 GPUs. We train the models for 480 epochs with extra 50 epochs for retraining. We start from a learning rate of 0.045, and then scale it by 0.98 every epochs.

**MSCOCO.** We evaluate and compare the performance of IGCv3 and MobileNetV2 for object detection on COCO dataset [24], which are used as a backbone for SSDLite and SSDLite2 respectively. As in previous work [51], we train using the union of 80k training images and a 35k subset of val images (trainval35k) and evaluate on test-dev. We train the models for 240 epochs on MXNet. We start from a learning rate of 0.004, and then divide it by 10 every 60 epochs. The input resolution is shown in Table 4.

## 4.2 Comparisons with IGCv1 and IGCv2.

We compare IGCv3 with IGCv1 and IGCv2 on CIFAR and ImageNet. The models of IGCv1 and IGCv2 replace the blocks in MobileNetV1 with their proposed blocks. Please check the details in [89, 41]. IGCv3 adopts two group convolutions with  $G_1 = 2$  and  $G_2 = 2$  respectively for the deeper version (IGCV3-D).

	#Params	CIFAR-10	CIFAR-100	ImageNet
IGCV1 (our impl.)	2.2M	$91.77 \pm 0.14$	$70.07 \pm 0.17$	-
IGCV2 (our impl.)	2.2M	$94.76 \pm 0.11$	$77.45 \pm 0.35$	70.7
IGCV3-D	2.2M	<b><math>94.96 \pm 0.07</math></b>	<b><math>77.95 \pm 0.39</math></b>	<b>72.1</b>

Table 1: Comparisons of different IGC blocks for classification accuracy on CIFAR and ImageNet. All the models keep the same number of parameters. Experiments are repeated five times, and the results are shown as *mean*  $\pm$  *std*. The parameter numbers are calculated without counting the final FC-layer. The results of IGCv1 and IGCv2 on CIFAR are reproduced by ourselves.

The comparisons of IGC blocks are shown in Table 1. IGCv3 outperforms the prior works slightly on CIFAR datasets, and achieves significant improvement about 1.5% on ImageNet. By introducing low-rank design into the sparse kernels, IGCv3 reduces the redundancies in feature dimensions and increases the depth of network to improve the generalization and the overall performance.

## 4.3 Comparisons with Other Mobile Networks.

We compare IGCv3 with MobileNetV2 and other mobile networks for classification and detection on several benchmarks. IGCv3 adopts two group convolutions with  $G_1 = 2$  and  $G_2 = 2$  for the deeper version (IGCV3-D) and with  $G_1 = 4$  and  $G_2 = 4$  for the wider version (IGCV3-W). Due to the limitation, we reproduce MobileNetV2 on 4 GPUs not proposed 16 GPUs. We try our best to train MobileNetV2 and provide the reproduced results as (our impl.). We adopt the settings of the best reproduced MobileNetV2 to train our models.

**CIFAR classification.** The classification accuracy comparisons of MobileNetV2 and IGCv3 are presented in Table 2. Introducing sparse design into low-rank kernels, IGC V3 further reduces the superfluous weights in convolutional kernels, and also makes the best use of parameters to improve the performance. IGCv3 outperforms MobileNetV2 a lot with the similar number of parameters. Moreover, IGCv3 with 50% parameters still achieves a better

	#Params	CIFAR-10	CIFAR-100
MobileNetV2 (our impl.)	2.1M	94.56 $\pm$ 0.16	77.09 $\pm$ 0.17
IGCV3-D 0.5 $\times$	1.1M	94.73 $\pm$ 0.14	77.29 $\pm$ 0.25
IGCV3-D 0.7 $\times$	1.5M	94.92 $\pm$ 0.16	77.83 $\pm$ 0.38
IGCV3-D 1.0 $\times$	2.2M	<b>94.96 <math>\pm</math> 0.07</b>	<b>77.95 <math>\pm</math> 0.39</b>

Table 2: Classification accuracy comparisons of MobileNetV2 and IGCv3 on CIFAR datasets. "Network  $s\times$ " means reducing the number of parameters in "Network 1.0 $\times$ " by  $s$  times. For IGCv3-D, we reduce the number of blocks to control the count of parameters. The parameter numbers are calculated without counting the final FC-layer.

performance, which has the same depth as MobileNetV2. The reason may be that the number of ReLU is half of MobileNet V2.

	#Params	MAdds	ImageNet
MobileNetV2 (0.7)(our impl.)	2.8M	210M	66.51
IGCV3-D (0.7)	2.8M	210M	<b>68.45</b>
MobileNetV1 [42]	4.2M	575M	70.6
ShuffleNet (1.5) [42]	3.4M	292M	71.5
MobileNetV2 [51]	3.4M	300M	72.0
MobileNetV2 (our impl.)	3.4M	300M	71.3
IGCV3-D	3.5M	318M	<b>72.2</b>
ShuffleNet (2.0) [42]	5.4M	524M	73.7
NasNet-A [47]	5.3M	564M	74.0
MobileNetV2 (1.4) [51]	6.9M	585M	<b>74.7</b>
MobileNetV2 (1.4)(our impl.)	6.9M	585M	73.76
IGCV3-D (1.4)	7.2M	610M	74.55

Table 3: Comparisons of different mobile networks for classification on ImageNet. We count the total number of Multiply-Adds for ops. "Network ( $\alpha$ )" means scaling the number of filters in "Network (1.0)" by  $\alpha$  times thus overall complexity will be roughly  $\alpha^2$  times of "Network ( $\alpha$ )".

**ImageNet classification.** We compare IGCv3 with other mobile models on ImageNet, shown in Table 3. With the similar computation complexity, IGCv3 always outperforms other networks. IGCv3-D (1.4) achieves a superior performance under the same training settings (74.55% vs 73.76%), though marginally worse than the original MobileNetV2 (1.4). Due to the difference of the final FC-layer, IGCv3-D has the different number of parameters in Table 3 from Table 1 and 2 (1.3M (1000 classes), 13k (100 classes) and 1.3k (10 classes)).

**Detection on COCO.** We extend IGCv3 to be a backbone for detection networks. SSDLite is proposed in [51], we follow the original framework, but replace all the feature extraction blocks with IGCv3, denoted by "SSDLite2". The comparisons of different methods for detection are shown in Table 4. We adopt IGCv3-W as our backbone for extracting the features from the layers at the same depth as MobileNetV2. IGCv3 is slightly better than MobileNetV2 with fewer parameters, and outperforms YOLOV2 0.6% mAP with much fewer number of parameters.



	Input Size	#Params	mAP
SSD-300 [25]	300x300	36.1M	23.2
SSD-512 [25]	512x512	36.1M	26.8
YOLOV2 [24]	-	50.7M	21.6
MobileNetV1 SSDLite [12]	320x320	5.1M	22.2
MobileNetV2 SSDLite [13]	320x320	4.3M	22.1
MobileNetV2 SSDLite (our impl.)	320x320	4.3M	22.1
IGCV3-W SSDLite2	320x320	4.0M	22.2

Table 4: Comparisons of different methods for detection on COCO. The models are evaluated with the different input size. Reproduced MobilenetV2 and IGCv3 are trained and evaluated with the input resolution of  $320 \times 320$ .

## 5 Ablation Study

We explore three main design choices: (i) deeper and wider networks; (ii) intermediate ReLUs; (iii) #branches in group convolutions. These three elements determine the architecture of the IGCv3 block and the whole network.

**Deeper and wider networks.** Most of related works enlarge the width of networks, which output high-dimensional features capturing more information.

	#Params	CIFAR-10	CIFAR-100	ImageNet
IGCV3-W	2.1M	$94.68 \pm 0.21$	$76.48 \pm 0.25$	71.1
IGCV3-Widest	2.1M	$94.62 \pm 0.19$	$76.39 \pm 0.27$	-
IGCV3-D	2.2M	<b><math>94.88 \pm 0.08</math></b>	<b><math>77.95 \pm 0.39</math></b>	<b>72.2</b>

Table 5: Comparisons of deeper and wider networks on CIFAR and ImageNet. IGCv3-Widest is designed by following the strict complementation condition.

Comparisons between deeper and wider networks are shown in Table 5. It can be observed that the deeper version significantly outperforms the wider version, which is as expected: (i) there are redundancies in feature dimensions, so further enlarging the width cannot bring about gains; (ii) the networks built by stacking bottlenecks improve the final performance with the increasing of depth, consistent to the observations in [8, 13].

	#Params	CIFAR-10	CIFAR-100
G1x1_ReLU_Cw3x3_ReLU_G1x1	2.2M	$94.44 \pm 0.11$	$76.61 \pm 0.29$
G1x1_Cw3x3_ReLU_G1x1	2.2M	<b><math>94.88 \pm 0.07</math></b>	<b><math>77.95 \pm 0.39</math></b>
G1x1_Cw3x3_G1x1_ReLU	2.2M	$93.76 \pm 0.08$	$75.53 \pm 0.25$

Table 6: Comparisons of different IGCv3 blocks for classification on CIFAR. The first column represents the architectures,  $G1 \times 1$  and  $Cw$  denote the group  $1 \times 1$  convolution and the channel-wise convolution respectively. The positions of ReLUs in the first block is the same as the MobileNetV2 block and the second is adopted in our IGCv3 block. Each convolution is followed by a BN.

**Intermediate ReLUs.** We explore the positions of non-linearities and the results are shown in Table 6. The second block (IGCV3 block) has obvious advantages over other blocks. The first block includes two ReLUs, and thus the resulting network has two times ReLUs more

than others, which may be a reason for its worse performance. The third block is equivalent to a convolutional kernel followed by a ReLU, which is worse than IGC blocks.

**#Branches in group convolutions.** It’s easy to find that there are many possible combinations of  $G_1$  and  $G_2$  for two group convolutions satisfying the loose complementary condition, only if the product of  $G_1$  and  $G_2$  is the *common divisor* of the number of input and output channels. Table 7 provides another two cases.

	#Params	CIFAR-10	CIFAR-100
IGCV3-D( $G_1 = 2, G_2 = 2$ )	2.2M	$94.88 \pm 0.07$	$77.95 \pm 0.39$
IGCV3-D( $G_1 = 2, G_2 = 4$ )	2.2M	<b><math>94.89 \pm 0.11</math></b>	<b><math>78.12 \pm 0.36</math></b>
IGCV3-D( $G_1 = 4, G_2 = 2$ )	2.2M	$94.71 \pm 0.26$	$77.84 \pm 0.40$

Table 7: Comparisons of IGCv3 blocks with different branches for classification on CIFAR.

From the results, shown in Table 7, we can find that the first group convolution prefers to be denser. The third group convolution projects the high-dimensional features back to the low-dimensional space, which results in information loss. Therefore, reducing more kernels may have little effect on its performance. In these cases, the IGCv3 blocks in the resulting networks share the same settings, e.g.  $G_1 = 2, G_2 = 2$ . Actually, IGCv3 blocks can adopt the different settings independently, which leads to a deeper network and achieves a better performance. In our experiments, we adopt  $G_1 = 2, G_2 = 2$  to reduce the memory cost, which also achieves a good performance.

## 6 Conclusion

In this paper, we propose a novel block, interleaved low-rank group convolutions, which enjoys the benefits of both low-rank and sparse convolutions. We introduce super-channels and the loose complementary condition to guide the design. Empirical results demonstrate the efficient of our models and the superiority over the state-of-the-arts, MobileNetV2 and IGCv2, on several benchmarks.

## References

- [1] Jose M Alvarez and Mathieu Salzmann. Learning the number of neurons in deep networks. In *Advances in Neural Information Processing Systems*, pages 2270–2278, 2016.
- [2] François Chollet. Xception: Deep learning with depthwise separable convolutions. *CoRR*, abs/1610.02357, 2016.
- [3] Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. *arXiv preprint arXiv:1602.02830*, 2016.
- [4] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009.

- 
- [5] Emily L Denton, Wojciech Zaremba, Joan Bruna, Yann LeCun, and Rob Fergus. Exploiting linear structure within convolutional networks for efficient evaluation. In *Advances in neural information processing systems*, pages 1269–1277, 2014.
  - [6] Song Han, Huizi Mao, and William J. Dally. Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. *CoRR*, abs/1510.00149, 2015.
  - [7] Song Han, Jeff Pool, John Tran, and William J. Dally. Learning both weights and connections for efficient neural network. In *NIPS*, pages 1135–1143, 2015.
  - [8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016. doi: 10.1109/CVPR.2016.90.
  - [9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *ECCV*, pages 630–645, 2016. doi: 10.1007/978-3-319-46493-0\_38.
  - [10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
  - [11] Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. *arXiv preprint arXiv:1707.06168*, 2017.
  - [12] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.
  - [13] Gao Huang, Zhuang Liu, and Kilian Q. Weinberger. Densely connected convolutional networks. *CoRR*, abs/1608.06993, 2016.
  - [14] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and < 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016.
  - [15] Yani Ioannou, Duncan P. Robertson, Jamie Shotton, Roberto Cipolla, and Antonio Criminisi. Training cnns with low-rank filters for efficient image classification. *CoRR*, abs/1511.06744, 2015.
  - [16] Yani Ioannou, Duncan P. Robertson, Roberto Cipolla, and Antonio Criminisi. Deep roots: Improving CNN efficiency with hierarchical filter groups. *CoRR*, abs/1605.06489, 2016.
  - [17] Max Jaderberg, Andrea Vedaldi, and Andrew Zisserman. Speeding up convolutional neural networks with low rank expansions. In *BMVC*, 2014.
  - [18] Alex Krizhevsky. Learning multiple layers of features from tiny images. *Technical report*, 2009.
  - [19] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. Fractalnet: Ultra-deep neural networks without residuals. *CoRR*, abs/1605.07648, 2016.

- [20] Chen-Yu Lee, Saining Xie, Patrick W. Gallagher, Zhengyou Zhang, and Zhuowen Tu. Deeply-supervised nets. In *AISTATS*, 2015.
- [21] Fengfu Li, Bo Zhang, and Bin Liu. Ternary weight networks. *arXiv preprint arXiv:1605.04711*, 2016.
- [22] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. *CoRR*, abs/1608.08710, 2016.
- [23] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *CoRR*, abs/1312.4400, 2013.
- [24] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.
- [25] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.
- [26] Jian-Hao Luo, Jianxin Wu, and Weiyao Lin. Thinet: A filter level pruning method for deep neural network compression. In *ICCV*, pages 5068–5076, 2017. doi: 10.1109/ICCV.2017.541.
- [27] Franck Mamalet and Christophe Garcia. Simplifying convnets for fast learning. In *ICANN*, pages 58–65, 2012. doi: 10.1007/978-3-642-33266-1\_8.
- [28] Jongsoo Park, Sheng Li, Wei Wen, Ping Tak Peter Tang, Hai Li, Yiran Chen, and Pradeep Dubey. Faster cnns with direct sparse convolutions and guided pruning. 2016.
- [29] Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. *arXiv preprint*, 2017.
- [30] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. *CoRR*, abs/1412.6550, 2014.
- [31] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation. *arXiv preprint arXiv:1801.04381*, 2018.
- [32] Or Sharir and Amnon Shashua. On the expressive power of overlapping architectures of deep learning. *arXiv preprint arXiv:1703.02065*, 2017.
- [33] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.
- [34] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin A. Riedmiller. Striving for simplicity: The all convolutional net. *CoRR*, abs/1412.6806, 2014.
- [35] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Training very deep networks. In *NIPS*, pages 2377–2385, 2015.

- [36] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *CVPR*, pages 2818–2826, 2016. doi: 10.1109/CVPR.2016.308.
- [37] Antonio Torralba, Robert Fergus, and William T. Freeman. 80 million tiny images: A large data set for nonparametric object and scene recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 30(11):1958–1970, 2008. doi: 10.1109/TPAMI.2008.128.
- [38] Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Learning structured sparsity in deep neural networks. In *NIPS*, pages 2074–2082, 2016.
- [39] Guotian Xie, Jingdong wang, Ting Zhang, Jianhuang Lai, Richang Hong, and Guo-Jun Qi. Igcv2: Interleaved structured sparse convolutional neural networks. In *CVPR*, 2018.
- [40] Saining Xie, Ross B. Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. *CoRR*, abs/1611.05431, 2016.
- [41] Ting Zhang, Guo-Jun Qi, Bin Xiao, and Jingdong Wang. Interleaved group convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4373–4382, 2017.
- [42] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. *arXiv preprint arXiv:1707.01083*, 2017.
- [43] Liming Zhao, Jingdong Wang, Xi Li, and Zhuowen Tu. Deep convolutional neural networks with merge-and-run mappings. *CoRR*, abs/1611.07718, 2016.
- [44] Aojun Zhou, Anbang Yao, Yiwen Guo, Lin Xu, and Yurong Chen. Incremental network quantization: Towards lossless cnns with low-precision weights. *arXiv preprint arXiv:1702.03044*, 2017.
- [45] Shuchang Zhou, Yuxin Wu, Zekun Ni, Xinyu Zhou, He Wen, and Yuheng Zou. Dorefanet: Training low bitwidth convolutional neural networks with low bitwidth gradients. *arXiv preprint arXiv:1606.06160*, 2016.
- [46] Chenzhuo Zhu, Song Han, Huizi Mao, and William J Dally. Trained ternary quantization. *arXiv preprint arXiv:1612.01064*, 2016.
- [47] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le. Learning transferable architectures for scalable image recognition. *CoRR*, abs/1707.07012, 2017.