Interleaved Structured Sparse Convolutional Neural Networks

Guotian Xie[1,2,*] Jingdong Wang[3†] Ting Zhang[3] Jianhuang Lai[1,2] Richang Hong[4] Guo-Jun Qi[5]
[1]Sun Yat-Sen University [2]Guangdong Key Laboratory of Information Security Technology
[3]Microsoft Research [4]Hefei University of Technology [5]University of Central Florida
guotian.xgt@alibaba-inc.com, {jingdw,tinzhan}@microsoft.com
stsljh@mail.sysu.edu.cn, hongrc.hfut@gmail.com, guojun.qi@ucf.edu

Abstract

In this paper, we study the problem of designing efficient convolutional neural network architectures with the interest in eliminating the redundancy in convolution kernels. In addition to structured sparse kernels, low-rank kernels and the product of low-rank kernels, the product of structured sparse kernels, which is a framework for interpreting the recently-developed interleaved group convolutions (IGC) and its variants (e.g., Xception), has been attracting increasing interests.

Motivated by the observation that the convolutions contained in a group convolution in IGC can be further decomposed in the same manner, we present a modularized building block, IGC-V2: interleaved structured sparse convolutions. It generalizes interleaved group convolutions, which is composed of two structured sparse kernels, to the product of more structured sparse kernels, further eliminating the redundancy. We present the complementary condition and the balance condition to guide the design of structured sparse kernels, obtaining a balance among three aspects: model size, computation complexity and classification accuracy. Experimental results demonstrate the advantage on the balance among these three aspects compared to interleaved group convolutions and Xception, and competitive performance compared to other state-of-the-art architecture design methods.

1. Introduction

Deep convolutional neural networks with small model size, low computation cost, but still high accuracy become an urgent request, especially in mobile devices. The efforts include (i) network compression: compress the pretrained model by decomposing the convolutional kernel matrix or removing connections or channels to eliminate redundancy, and (ii) architecture design: design small kernels, sparse kernels or use the product of less-redundant kernels to approach single kernel and train the networks from scratch.

Our study lies in architecture design using the product of less-redundant kernels for composing a kernel. There are two main lines: multiply low-rank kernels (matrices) to approximate a high-rank kernel, e.g., bottleneck modules [6], and multiply sparse matrices, which has attracted research efforts recently [43, 11, 1] and is the focus of our work.

We point out that the recently-developed algorithms, such as interleaved group convolution [43], deep roots [11], and Xception [1], compose a dense kernel using the product of two structured-sparse kernels. We observe that one of the two kernels can be further approximated. For example, the 1×1 kernel in Xception and deep roots can be approximated by the product of two block-diagonal sparse matrices. The suggested secondary group convolution in interleaved group convolutions contains two branches and each branch is a 1×1 convolution, which similarly can be further approximated. This is able to further reduce the redundancy.

Motivated by this, we design a building block, IGC-V2: Interleaved Structured Sparse Convolution, as shown in Figure 1, which consists of successive group convolutions. This block is mathematically formulated as multiplying structured-sparse kernels, each of which corresponds to a group convolution. We introduce the complementary condition and the balance condition, so that the resulting convolution kernel is dense and there is a good balance among three aspects: model size, computation complexity and classification performance. Experimental results demonstrate the advantage of the balance among these three aspects compared to interleaved group convolutions and Xception, and competitive performance compared to other state-of-the-art architecture design methods.

2. Related Work

Most existing technologies design efficient and effective convolutional kernels using various forms with redundancy

[*]This work was done when Guotian Xie was an intern at Microsoft Research, Beijing, P.R. China. Personal email: xieguotian1990@gmail.com
[†]Corresponding author.

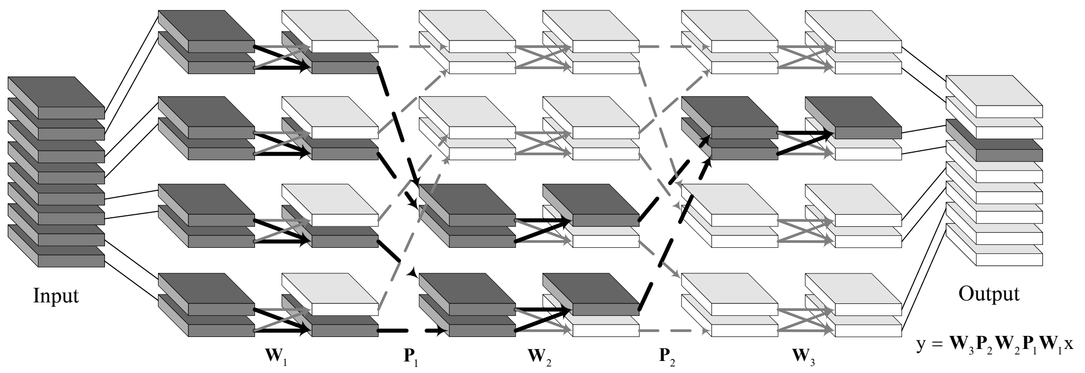

Input
Output
$\mathbf{W}_1$
$\mathbf{P}_1$
$\mathbf{W}_2$
$\mathbf{P}_2$
$\mathbf{W}_3$
$\mathbf{y} = \mathbf{W}_3\mathbf{P}_2\mathbf{W}_2\mathbf{P}_1\mathbf{W}_1\mathbf{x}$

Figure 1. IGC-V2: the Interleaved Structured Sparse Convolution. $\mathbf{W}_1$, $\mathbf{W}_2$, $\mathbf{W}_3$ (denoted as solid arrows) are sparse block matrices corresponding to group convolutions. $\mathbf{P}_1$ and $\mathbf{P}_2$ (denoted as dashed arrows) are permutation matrices. The resulting composed kernel $\mathbf{W}_3\mathbf{P}_2\mathbf{W}_2\mathbf{P}_1\mathbf{W}_1$ is ensured to satisfy the *complementary condition* which guarantees that for each output channel, there exists one and only one path connecting the output channel to each input channel. The bold line connecting gray feature maps shows such a path.

eliminated, by learning from scratch or approximating pretrained models. We roughly divide them into low-precision kernels, sparse kernels, low-rank kernels, product of low-rank kernels, product of structured sparse kernels.

Low-precision kernels. There exist redundancies in the weights in convolutional kernels represented by float numbers. The technologies eliminating such redundancies include quantization [45, 5], binarization [3, 30], and trinarization [21, 46, 47]. Weight-shared kernels in which some weights are equal to the same value, are in some sense low-precision kernels.

Sparse kernels. Sparse kernels, or namely sparse connections, mean that some weights are nearly zero. The efforts along this path mainly lie in how to perform optimization, and the technologies include non-structure sparsity regularization [24, 29], and structure sparsity regularization [39, 28]. The scheme of structure sparsity regularization is more friendly for hardware acceleration and storage. Recently, group convolutions, adopted in [41, 44] are essentially structured-sparse kernels. Different from sparsity regularization, the sparsity pattern of group convolution is manually pre-defined.

Low-rank kernels. Small filters, e.g., 3×3 kernels replacing 5×5 kernels, reduce the ranks in the spatial domain. Channel pruning [25] and filter pruning [40, 22, 26] compute low-rank kernels in the output channel domain and the input channel domain, respectively[1].

Composition from low-rank kernels. Using a pair of 1×3 and 3×1 kernels to approximate a 3×3 kernel [12, 13, 27] is an example of using the product of two small (low-rank) filters. Tensor decomposition uses the product of low-rank/small tensors (matrices) to approximate the kernel in the tensor form along the spatial domain [4, 13], or the input and output channel domains [4, 15, 13]. The bottleneck structure [6], if the intermediate ReLUs are removed, can be viewed as the low-rank approximation along the output channel domain.

Composition from sparse kernels. Interleaved group convolution [43] consists of two group convolutions, each of which corresponds to a structured-sparse kernel (the sizes are the same to that of the kernel to be approximated for the 1×1 convolutions). Satisfying the complementary property [43] leads to that the resulting composite kernel is dense. Xception [1] can be viewed as an extreme case of interleaved group convolutions: one group convolution is degraded to a regular convolution and the other one is a channel-wise convolution, an extreme group convolution. Deep roots [11] instead uses the product of a structured-sparse kernel and a dense kernel. Our approach belongs to this category and shows a better balance among model size, computation complexity and classification accuracy.

3. Our Approach

The operation in a convolution layer in convolutional neural networks relies on a matrix-vector multiplication operation at each location:

$$\mathbf{y} = \mathbf{W}\mathbf{x}. \tag{1}$$

Here the input $\mathbf{x}$, corresponding to a patch around the location in the input channels, is a SC_i-dimensional vector, with S being the kernel size (e.g., $S = 3 \times 3$), C_i being the number of input channels. The output $\mathbf{y}$ is a C_o-dimensional vector, with C_o being the number of output channels. $\mathbf{W}$ is formed from C_o convolutional kernels and each row corresponds to a convolutional kernel. For presentation clarity, we assume $C_i = C_o = C$, but all the formulations can be generalized to $C_i \neq C_o$.

3.1. A Review of IGC, Xception and Deep Roots

We show that recent architecture design algorithms, Xception [1], deep roots [11], and interleaved group convo-

[1]Small filters, channel and filter pruning in some sense can also be interpreted as sparse kernels: some columns or rows are removed.

lutions (IGC) [43], compose a dense convolution matrix $\mathbf{W}$ by multiplying possibly sparse matrices:

$$\mathbf{y} = \mathbf{P}^2\mathbf{W}^2\mathbf{P}^1\mathbf{W}^1\mathbf{x}, \qquad (2)$$

where $\mathbf{W}^1$ and $\mathbf{W}^2$ are both, or at least one matrix is block-wise sparse, $\mathbf{P}^i$ is a permutation matrix that is used to reorder the channels, and $\mathbf{W} = \mathbf{P}^2\mathbf{W}^2\mathbf{P}^1\mathbf{W}^1$ is a dense matrix.

Interleaved group convolutions. The interleaved group convolution block consists of primary and secondary group convolutions. The corresponding kernel matrices $\mathbf{W}^1$ and $\mathbf{W}^2$ are block-wise sparse,

$$\mathbf{W}^i = \begin{bmatrix} \mathbf{W}^i_1 & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{W}^i_2 & \mathbf{0} & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{W}^i_{G_i} \end{bmatrix}, \qquad (3)$$

where $\mathbf{W}^i_g$ ($i = 1$ or 2) is the kernel matrix over the corresponding channels in the gth branch, G_i is the number of branches in the ith group convolution. In the case suggested in [43], the primary group convolution is a group 3×3 convolution, $G_1 = \frac{C}{2}$, and $\mathbf{W}^1_g$ is a matrix of size $2 \times (2S)$. The secondary group convolution is a group 1×1 convolution, $G_2 = 2$, $\mathbf{W}^2_1$ and $\mathbf{W}^2_2$ are both dense matrices of size $\frac{C}{2} \times \frac{C}{2}$.

Xception. The Xception block consists of a 1×1 convolution layer followed by a channel-wise convolution layer. It is pointed out that the order of the two layers does not make effects. For convenience, we below discuss the form with the 1×1 convolution put as the second operation. $\mathbf{W}^2$ is a dense matrix of size $C \times C$. $\mathbf{W}^1$ is a sparse block matrix of size $C \times (SC)$, a degraded form of the matrix shown in Equation 3: there are C blocks and $\mathbf{W}^1_g$ is degraded to a row vector of size S.

Deep roots. In deep roots, $\mathbf{W}^2$ is a dense matrix of size $C \times C$, i.e., corresponding to a 1×1 convolution while $\mathbf{W}^1$ is a sparse block matrix as shown in Equation 3, corresponding to a group convolution.

Complexity. The computation complexity of Equation 2 is $O(|\mathbf{W}^1|_0 + |\mathbf{W}^2|_0)$ (with the complexity in permutation is ignored), where $|\mathbf{W}^i|_0$ is the number of non-zero entries. The sparse block matrix, as given in Equation 3, are storage friendly, and the storage/memory cost is also $O(|\mathbf{W}^1|_0 + |\mathbf{W}^2|_0)$.

3.2. Interleaved Structured Sparse Convolutions

Our approach is motivated by the observations: (i) the block $\mathbf{W}^i_g$ in Equation 3 and the 1×1 convolution in Xception are dense and can be composed by multiplying sparse matrices, thus further eliminating the redundancy and saving the storage and time cost; and (ii) such a process can be repeated more times.

The proposed Interleaved Structured Sparse Convolution (IGC-V2) is mathematically formulated as follows,

$$\mathbf{y} = \mathbf{P}_L\mathbf{W}_L\mathbf{P}_{L-1}\mathbf{W}_{L-1}\dots\mathbf{P}_1\mathbf{W}_1\mathbf{x} \qquad (4)$$

$$= (\prod\nolimits_{l=L}^{1}\mathbf{P}_l\mathbf{W}_l)\mathbf{x}. \qquad (5)$$

Here, $\mathbf{P}_l\mathbf{W}_l$ is a sparse matrix. $\mathbf{P}_l$ is a permutation matrix, and the role is to reorder the channels so that $\mathbf{W}_l$ is a sparse block matrix, as given in Equation 3 and corresponds to the lth group convolution, where the numbers of channels in all the branches are in our work set to be the same, equal to K_l, for easy design.

Construct a dense composed kernel matrix. We introduce the following *complementary condition*, which is generalized from interleaved group convolutions [43], as a rule for constructing the L group convolutions such that the resulting composed convolution kernel matrix is *dense*.

Condition 1 (Complementary condition) $\forall m$, $(\mathbf{W}_L\prod_{l=L-1}^{m}\mathbf{P}_l\mathbf{W}_l)$ *corresponds to a group convolution and* $(\mathbf{W}_{m-1}\prod_{l=m-2}^{1}\mathbf{P}_l\mathbf{W}_l)$ *also corresponds to a group convolution. The two group convolutions are thought complementary if the 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.*

Here is the sketch showing that an interleaved structured sparse convolution block satisfying the complementary condition is dense. The proof is based on two points: (i) for a group convolution, we have that any channel output from a branch is connected to the channels input to this branch and any channel input to a branch is connected to the channels output from this branch; (ii) for two complementary group convolutions, the channels output from any branch of the second group convolution are connected to the channels input to the corresponding branch, which are from all the branches of the first group convolution. As a result, the channels output from an IGC-V2 is connected to all the channels input to the IGC-V2, i.e., the IGC-V2 kernel is *dense*.

Let us look at the relation between the number of channels, C, and the number of channels in the branches of L group convolutions, $\{K_1, K_2, \dots, K_L\}$. We analyze the relation according to Equation 5: (i) An input channel is connected to K_1 intermediate channels output by the first group convolution. (ii) Let C_{l-1} be the number of intermediate channels output by the $(l-1)$th group convolution, to which an input channel is connected. The *complementary condition* indicates that through the lth group convolution an input channel is connected to *exactly* K_lC_{l-1} intermediate channels output by the lth group convolution. (iii) Finally, an input channel is connected to exactly $C_L = \prod_{l=1}^{L} K_l$ channels output from the L group convolutions. Since the composed kernel is dense, we have

$$\prod\nolimits_{l=1}^{L} K_l = C. \tag{6}$$

Because of the complementary property, there is no waste connection: there is only one path between each input channel and each output channel. Besides, the complementary condition is a sufficient condition yielding a dense composed kernel matrix, and not a necessary condition.

When the amount of parameters is the smallest? We further analyze when the number of parameters with L group convolutions, as given in Equation 5, satisfying the complementary condition, is the smallest.

We have that the number of parameters in the lth group convolution is CK_l for the 1×1 convolutions, and CSK_l for the spatial (e.g., $S = 3 \times 3$) convolution. It is easily shown that for consuming fewer parameters there is only one group spatial convolution and all others are 1×1. The spatial convolution lies in any group convolution, and without affecting the analysis, we assume it lies in the first group convolution[2]. Thus, the number of total parameters Q, smaller number of parameters in permutation matrices ignored, is:

$$Q = C \sum\nolimits_{l=2}^{L} K_l + CSK_1. \tag{7}$$

According to Jensen's inequality, we have

$$\begin{aligned} Q = {} & C \sum\nolimits_{l=2}^{L} K_l + CSK_1 & (8) \\ \geqslant {} & CL(SK_1 \prod\nolimits_{l=2}^{L} K_l)^{\frac{1}{L}} & (9) \\ = {} & CL(SC)^{\frac{1}{L}}. & (10) \end{aligned}$$

Here, the equality from the second line to the third line holds because of Equation 6. The equality in the second line holds, i.e., $Q = CL(SC)^{\frac{1}{L}}$, when the following *balance condition* is satisfied[3],

$$SK_1 = K_2 = \cdots = K_L (= (SC)^{\frac{1}{L}}). \tag{11}$$

Furthermore, let us see the choice of L, yielding the smallest amount of parameters ($Q = CL(SC)^{\frac{1}{L}}$), guaranteeing a dense composed kernel. We present a rough analysis by considering the derivative of Q with respect to L:

$$\begin{aligned} \frac{d \log Q}{dL} = {} & \frac{d}{dL}(\log C + \log L + \frac{1}{L}\log(SC)) & (12) \\ = {} & \frac{1}{L} - \frac{1}{L^2}\log(SC). & (13) \end{aligned}$$

When the derivative is zero, $\frac{1}{L} - \frac{1}{L^2}\log(SC) = 0$, we have that Q is the minimum if

$$L = \log(SC). \tag{14}$$

Examples. We take an example: separate the convolution along the spatial domain and the channel domain, to construct the IGC-V2 block. The first group convolution is an extreme group convolution, a channel-wise 3×3 convolution, followed by several group 1×1 convolutions. This can be regarded as decomposing the 1×1 convolution in Xception into group 1×1 convolutions. In this case, the balance condition becomes $K_2 = K_3 = \cdots = K_L = C^{\frac{1}{L-1}}$, for which the amount of parameters is the smallest. As we empirically validate in Section 4.3, under the same number of parameters, an IGC-V2 block satisfying such a balance condition leads to the maximum width and consistently superior performance: the best or nearly best. This consistency observation is different from [43] and might stem from that the balance condition is only used to 1×1 group convolutions and that there is no coupling with spatial convolutions.

We also study the construction from interleaved group convolutions: each submatrix $\mathbf{W}_g^2$ in Equation 3 in the secondary group convolution corresponds to a (dense) 1×1 convolution over a subset of channels, and thus can be further decomposed into group convolutions. The first group convolution is still a group 3×3 convolution (other than channel-wise). Consequently, the balance condition given in Equation 11 is deduced from the coupling of convolutions over the spatial and channel domains, which does not lead to the consistency between the width increase and the performance gain and makes the analysis uneasy. This is empirically validated in our experiments in Section 4.4. Thus, we suggest to separate the convolution along the spatial and channel domains and design an IGC-V2 over the channel domain.

3.3. Discussions

Non-structured sparse kernels. There is a possible extension: remove the structured sparsity requirement, i.e., replace the group convolution by a non-structured sparse kernel, and introduce the dense constraint (the composed kernel is dense) and the sparsity constraint. This potentially results in better performance, but leads to two drawbacks: the optimization is difficult and non-structured sparse matrices are not storage-friendly.

Complementary condition. The complementary condition is a sufficient condition guaranteeing the resulting composed kernel is dense. It should be noted that it is not a necessary condition. Moreover, it is also not necessary that the composed kernel is dense, and further sparsifying the connections, which remains as a future work, might be beneficial. The complementary condition is an effective guide to design the group convolutions.

Sparse matrix multiplication and low-rank matrix multiplication. Low-rank matrix (tensor) multiplication or decomposition has been widely studied in matrix analysis [19, 20] and applied to network compression and network architecture design. In comparison, sparse matrix (tensor) multiplication or decomposition is rarely studied in

[2]The formulation is not the same to Equation 5 if the group 3×3 convolution is not the first, but essentially they are the same.

[3]This can be regarded as an extension of the analysis in [43].

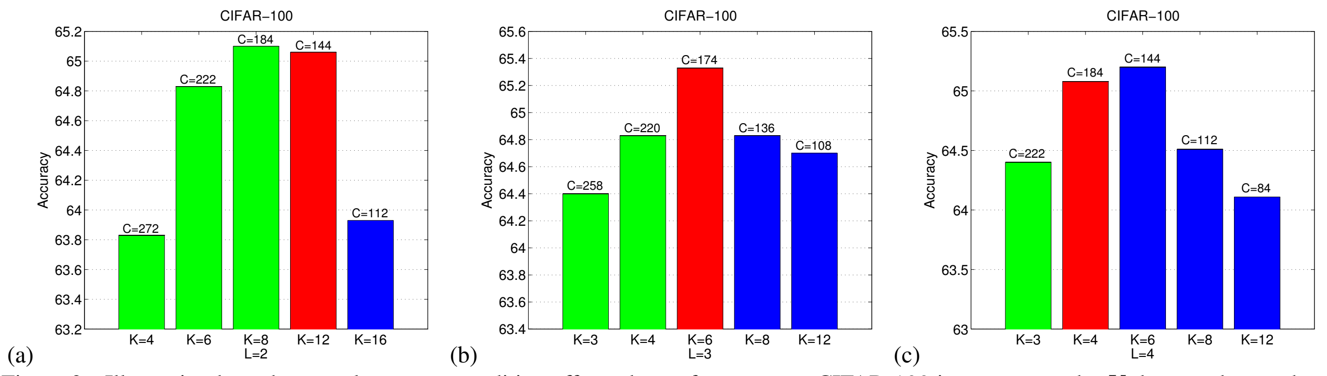


Figure 2. Illustrating how the complementary condition affects the performance on CIFAR-100 in our approach. K denotes the number of channels in each branch and C denotes the width of the network. With L fixed, the composed kernel is denser with a larger K. The red bar corresponds to the case in which the complementary condition is the most satisfied. The best performances corresponding to the red bar or the bars immediately near to the red bar show that the complementary condition is reasonable for IGC-V2 design.

matrix analysis. The future works include applying sparse matrix decomposition to compress convolutional networks, combining low-rank and sparse matrices together: low-rank sparse matrix multiplication or decomposition, and so on.

4. Experiment

4.1. Datasets and Training Settings

CIFAR. The CIFAR datasets [16], CIFAR-10 and CIFAR-100, are subsets of the 80 million tiny images [37]. Both datasets contain 60000 32×32 color images with 50000 images for training and 10000 images for test. The CIFAR-10 dataset consists of 10 classes, each of which contains 6000 images. There are 5000 training images and 1000 testing images per class. The CIFAR-100 dataset consists of 100 classes, each of which contains 600 images. There are 500 training images and 100 testing images per class. The standard data augmentation scheme we adopt is widely used for these datasets [6, 10, 18, 9, 17, 23, 31, 34, 35]: we first zero-pad the images with 4 pixels on each side, and then randomly crop them to produce 32×32 images, followed by horizontally mirroring half of the images. We normalize the images by using the channel means and standard deviations.

Tiny ImageNet. The Tiny ImageNet dataset[4] is a subset of ImageNet [32]. The image size is resized to 64×64. There are 200 classes, sampled from 1000 classes of ImageNet, and 500 training images, 50 validation images and 50 testing images per class. In our experiment, we adopt the data augmentation scheme: scale up the training images randomly to the size within $[64, 80]$, and randomly crop a 64×64 patch for training, randomly horizontal mirroring and normalize the cropped images by subtracting the channel means and standard deviations.

Training settings. For CIFAR, we adopt the same training settings as [43]. We use SGD with Nesterov momentum to update network, starting from learning rate 0.1 and multiplying with a factor 0.1 at 200 epochs, 300 epochs and 350 epochs. Weight decay is set as 0.0001 and momentum as 0.9. We train the network with batch size as 64 for 400 epochs and report the accuracy at the final iteration. The implementation is based on Caffe [14]. For Tiny ImageNet, we use the similar training settings as CIFAR, except that we train for totally 200 epochs and multiply the learning rate with a factor 0.1 at 100 epochs, 150 epochs and 175 epochs. To adapt Tiny ImageNet to the networks designed for CIFAR, we set the stride of the first convolution layer as 2, which is adopted in [8] as well.

[4]https://tiny-imagenet.herokuapp.com/

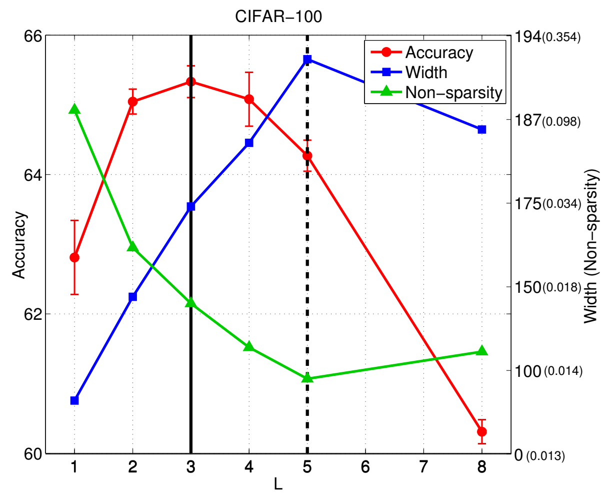


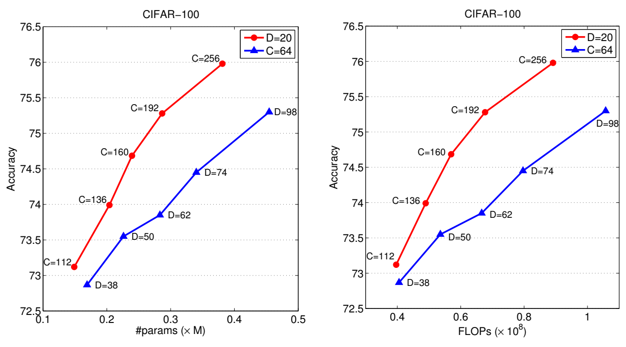
Figure 3. Illustrating how the number of layers L affects the performance on CIFAR-100. The number of channels in each branch of group convolution is chosen to satisfy both the balance condition and the (nearly) complementary condition, and to keep the #params the same. The maximum accuracy is achieved at some L, in which the width and the non-sparsity degree reach a balance.

4.2. Empirical Analysis

Complementary condition. We empirically investigate how the complementary condition affects the performance

Table 1. Illustrating the architectures of networks we used in the experiments. x is the number of channels at the first stage. B is the number of blocks and the skip connection is added every two blocks. $x \times (3 \times 3, 1)$ means a 3×3 channel-wise convolution with the channel number being x. L and K are the hyper-parameters of IGC-V2. $[L-1, x, (1 \times 1, K)]$ denotes the $(L-1)$ group 1×1 convolutions with each branch containing K channels. For IGC-V2 (Cx), $L = 3$, and for IGC-V2* (Cx), $K = 8$, $L^* = \lceil log_K(x) \rceil + 1$.

Output size	Xception (Cx)	IGC-V1 (Cx)	IGC-V2 (Cx)	IGC-V2* (Cx)
32×32	$(3 \times 3, x)$	$(3 \times 3, x)$	$(3 \times 3, x)$	$(3 \times 3, 64)$
32×32	$\begin{bmatrix} x \times (3 \times 3, 1) \\ (1 \times 1, x) \end{bmatrix} \times B$	$\begin{bmatrix} \frac{x}{2} \times (3 \times 3, 2) \\ 2 \times (1 \times 1, \frac{x}{2}) \end{bmatrix} \times B$	$\begin{bmatrix} x \times (3 \times 3, 1) \\ L-1, x, (1 \times 1, K_{s_1}) \end{bmatrix} \times B$	$\begin{bmatrix} x \times (3 \times 3, 1) \\ L^*-1, x, (1 \times 1, K) \end{bmatrix} \times B$
16×16	$\begin{bmatrix} 2x \times (3 \times 3, 1) \\ (1 \times 1, 2x) \end{bmatrix} \times B$	$\begin{bmatrix} x \times (3 \times 3, 2) \\ 2 \times (1 \times 1, x) \end{bmatrix} \times B$	$\begin{bmatrix} 2x \times (3 \times 3, 1) \\ L-1, 2x, (1 \times 1, K_{s_2}) \end{bmatrix} \times B$	$\begin{bmatrix} 2x \times (3 \times 3, 1) \\ L^*-1, 2x, (1 \times 1, K) \end{bmatrix} \times B$
8×8	$\begin{bmatrix} 4x \times (3 \times 3, 1) \\ (1 \times 1, 4x) \end{bmatrix} \times B$	$\begin{bmatrix} 2x \times (3 \times 3, 2) \\ 2 \times (1 \times 1, 2x) \end{bmatrix} \times B$	$\begin{bmatrix} 4x \times (3 \times 3, 1) \\ L-1, 4x, (1 \times 1, K_{s_3}) \end{bmatrix} \times B$	$\begin{bmatrix} 4x \times (3 \times 3, 1) \\ L^*-1, 4x, (1 \times 1, K) \end{bmatrix} \times B$
1×1	average pool, fc, softmax			
Depth	$3B + 2$			



(a) (b)

Figure 4. Illustrating how the performance changes when our network goes deeper or wider. We use the network structure IGC-V2* (Cx) in table 1 and conduct the experiments with various depths (denoted as D) and widths (denoted as C). Both going wider and going deeper increase the performance and the benefit of going wider is greater than going deeper.

over the 8-layer network without down-sampling, where there are 6 intermediate layers except the first convolutional layer and the last FC layer. We study an IGC-V2 building block, which consists of L group convolutions: a channel-wise 3×3 convolution, $(L-1)$ group 1×1 convolutions with each branch containing K channels. We conduct the studies over such blocks, where the IGC-V2 is only applied to 1×1 convolutions, for removing the possible influence of the coupling with spatial convolutions. The results are given in Figure 2 with various values of K and L under almost the same number of parameters.

An IGC-V2 block satisfying the complementary condition leads to a dense kernel with a large width under the same number of parameters. The red bars in Figure 2 depict the results for the IGC-V2 blocks that nearly satisfy this condition. The blue bars in Figure 2 show the results of the blocks that correspond to dense kernels but do not satisfy the complementary condition, thus with more redundancy. It can be seen that the networks with the IGC-V2 blocks *(nearly) satisfying the complementary condition* (the red bars and the bars next to the red bar in Figure 2 (a) and (c)) achieve the best performance. We also notice that the red bar in Figure 2 (b), satisfying the complementary condition performs better than the best results in Figure 2 (a) and (c).

In addition, we look at how the sparsity (density) of the composed convolution kernel affects the performance. The results correspond to the green bars in Figure 2. With L being fixed, the kernel is more sparse with smaller K. It can be seen that the denser kernel leads to higher performance. In Figure 2 (a), the performance for the block $K = 8$, which is quite *near to satisfy the complementary condition* is slightly better than and almost the same to the denser kernel $K = 12$. The reason might be that though the kernel for $K = 8$ is more sparse, but it corresponds to a larger width.

The effect of L. We empirically show how the number L of group convolutions affects the performance under the same number of parameters on the CIFAR-100 dataset as an example. We still use the 8-layer network and study the IGC-V2 block, which contains a channel-wise spatial convolution and $(L-1)$ group 1×1 convolutions satisfying the balance condition.

Figure 3 shows the accuracy curve, the width curve, and the non-sparsity curve, where the non-sparsity value is the ratio of the number of non-zero parameters to the size of the resulting dense kernel matrix. It can be seen that the function of accuracy w.r.t. L is concave, the function of width w.r.t. L is concave, and the function of non-sparsity w.r.t. L is convex. The accuracy depends on both the width and the non-sparsity degree. When width becomes larger, accuracy might be higher. On the other hand, accuracy might be lower when non-sparsity degree becomes smaller. The black dashed line denotes an extreme case that the width is the largest and the non-sparsity is the smallest, the performance however is not the best. Instead, the maximum accuracy is achieved at some L, in which the width and the non-sparsity degree reach a balance denoted by the black solid line.

Deeper and wider networks. We also conduct experiments to explore how the performance changes when our network goes deeper and wider. In this study, the IGC-V2 block is composed of a channel-wise 3×3 convolution and $(L-1)$ group 1×1 convolutions.

Table 2. Classification accuracy comparison between Xception and IGC-V2 over the 8-layer network with various widths. The number of parameters and FLOPs are calculated within each block.

Network	#Params / FLOPs	C	CIFAR-10	CIFAR-100
Xception	≈ 4700 /	64	88.82 ± 0.38	62.81 ± 0.53
IGC-V2	$\approx 4.8 \times 10^6$	144	$\mathbf{89.12 \pm 0.15}$	$\mathbf{65.05 \pm 0.18}$
Xception	≈ 10000 /	96	90.21 ± 0.62	65.21 ± 0.31
IGC-V2	$\approx 10^7$	256	$\mathbf{90.63 \pm 0.11}$	$\mathbf{67.68 \pm 0.28}$
Xception	≈ 17000 /	128	91.03 ± 0.30	66.51 ± 0.68
IGC-V2	$\approx 1.7 \times 10^7$	361	$\mathbf{91.35 \pm 0.12}$	$\mathbf{68.86 \pm 0.57}$

Table 3. Classification accuracy comparison between Xception, IGC, and our network under various widths with depth fixed as 20.

Network	#Params ($\times M$)	FLOPs ($\times 10^8$)	CIFAR-100	Tiny ImageNet
Xception ($C58$)	0.44	0.67	74.46 ± 0.24	57.34 ± 0.34
IGC-V1 ($C96$)	0.40	0.82	75.10 ± 0.36	58.37 ± 0.60
IGC-V2* ($C216$)	0.32	0.76	$\mathbf{75.35 \pm 0.56}$	$\mathbf{59.40 \pm 0.22}$
Xception ($C71$)	0.66	0.98	75.66 ± 0.11	58.28 ± 0.16
IGC-V1 ($C126$)	0.60	1.28	76.18 ± 0.20	59.67 ± 0.47
IGC-V2* ($C304$)	0.48	1.12	$\mathbf{76.30 \pm 0.19}$	$\mathbf{60.50 \pm 0.29}$
Xception ($C83$)	0.89	1.31	76.22 ± 0.05	59.01 ± 0.52
IGC-V1 ($C150$)	0.79	1.72	76.69 ± 0.51	60.73 ± 0.50
IGC-V2* ($C416$)	0.65	1.52	$\mathbf{77.02 \pm 0.20}$	$\mathbf{60.98 \pm 0.23}$

We study the performance over the networks with identity mappings as skip connections, where we replace the regular convolution in the residual network with our IGC-V2 building block. We do experiments on IGC-V2* (Cx) in Table 1, where Cx means that the network width C in the first stage is x. There are two types of experiments. (1) Go wider: we fix the depth as 20 ($B = 6$) and vary the width among $\{112, 136, 160, 192, 256\}$. (2) Go deeper: we fix the width as 64 and vary the depth among $\{38, 50, 62, 74, 98\}$. The results are shown in Figure 4 (a) and (b). One can see that both going wider and going deeper increase the network performance, and the benefit of going wider using a relative small depth is greater than that of going deeper, which is consistent to the observation for regular convolutions.

4.3. Comparison With Xception

We empirically show the comparison with Xception using various widths and various depths under roughly the same number of parameters.

Varying the width. We firstly conduct the experiments over the 8-layer network without down-sampling, where the 6 intermediate convolutional layers (except the first convolutional layer and the last FC layer) are replaced with Xception blocks and our IGC-V2 blocks. Our IGC-V2 block is composed of a channel-wise 3×3 convolution similar to Xception, and two group 1×1 convolutions corresponding to the 1×1 convolution in Xception. The comparison on CIFAR-10 and CIFAR-100 is given in Table 2. It can be seen that our networks consistently perform better than Xception on both CIFAR-10 and CIFAR-100 datasets. In particular on CIFAR-100, our networks achieve at least 2% improvement. The reason might be that our network using the IGC-V2 block is wider and thus improve the performance.

Table 4. Classification accuracy comparison between Xception and our network under various depths.

Network	#Params ($\times M$)	FLOPs ($\times 10^8$)	CIFAR-100	Tiny ImageNet
D= 8				
Xception ($C35$)	0.056	0.095	67.00 ± 0.29	49.91 ± 0.18
IGC-V2 ($C64$)	0.047	0.095	$\mathbf{67.83 \pm 0.35}$	$\mathbf{51.40 \pm 0.32}$
D= 20				
Xception ($C35$)	0.168	0.268	70.97 ± 0.10	55.29 ± 0.19
IGC-V2 ($C64$)	0.149	0.262	$\mathbf{71.69 \pm 0.09}$	$\mathbf{55.72 \pm 0.43}$
D= 26				
Xception ($C35$)	0.223	0.355	72.48 ± 0.18	55.39 ± 0.35
IGC-V2 ($C64$)	0.200	0.346	$\mathbf{72.94 \pm 0.26}$	$\mathbf{56.32 \pm 0.38}$

In addition, we report the results over 20-layer networks with various widths. The networks we used are IGC-V2* (Cx) and Xception (Cx) illustrated in Table 1, and we fix the channel number at the first convolutional layer in Xception as 35. The results are presented in Table 3. We can see that our network with fewer number of parameters, performs better than Xception, which shows the powerfulness of IGC-V2 block.

Varying the depth. We also compare the performances between Xception and our network IGC-V2 with various depths: $8, 20, 26$. The width C in Xception is fixed as 35, and the width in our network is set to 64 in order to keep the number of parameters smaller than Xception. For IGC-V2 ($C64$), the number of channels in each branch are the same within the stage and are different for the three stages. To satisfy the complementary condition, K_{s_1}, K_{s_2}, and K_{s_3} are set to $8, 16, 32$ respectively. The results over CIFAR-100 and Tiny ImageNet are given in Table 4. Our IGC-V2 ($C64$) network performs better than Xception ($C35$) with smaller numbers of parameters and smaller or similar computation complexity. For example, when depth is 26, our network consuming fewer number of parameters and less computation complexity gets 56.32% accuracy on Tiny ImageNet, 1% better than 55.39% of Xception.

4.4. Comparison With IGC.

Varying the width. Similar to the comparison with Xception, we perform comparison with IGC [43] (denoted by IGC-V1 in experiments) over the simple networks and over 20-layer networks with different widths. The results are presented respectively in Table 5 and Table 3, in which the observation is consistent to that from the comparison to Xception.

Let us look at the detailed results over the simple 8-layer network. The IGC-V1 block is designed by following [43]: the primary group convolution contains two branches. We study two designs of our IGC-V2 block. The first design follows the IGC-V1 design manner: the first group convolution contains two branches and the other two group convolution contains the same number of branches. In this case,

Table 5. Classification accuracy comparison between IGC-V1 and our networks with two designs, IGC-V2 I and IGC-V2 II, over the 8-layer network with various widths. The number of parameters and FLOPs are calculated within each block.

Network	#Params / FLOPs	C	CIFAR-10	CIFAR-100
IGC-V1	≈ 3000 / $\approx 3.2 \times 10^6$	64	87.93 ± 0.10	60.89 ± 0.46
IGC-V2 I		98	87.41 ± 0.35	61.35 ± 0.41
IGC-V2 II		100	$\mathbf{88.01 \pm 0.32}$	$\mathbf{62.32 \pm 0.57}$
IGC-V1	≈ 6000 / $\approx 6.1 \times 10^6$	96	89.29 ± 0.26	64.58 ± 0.19
IGC-V2 I		162	89.06 ± 0.08	65.39 ± 0.61
IGC-V2 II		169	$\mathbf{89.66 \pm 0.17}$	$\mathbf{66.00 \pm 0.78}$
IGC-V1	≈ 10000 / $\approx 1.1 \times 10^7$	128	90.16 ± 0.20	66.59 ± 0.48
IGC-V2 I		242	89.85 ± 0.22	67.21 ± 0.34
IGC-V2 II		256	$\mathbf{90.63 \pm 0.11}$	$\mathbf{67.68 \pm 0.28}$

Table 6. Classification accuracy comparison between IGC-V1 and our network under various depths.

Network	#Params ($\times M$)	FLOPs ($\times 10^7$)	CIFAR-100	Tiny ImageNet
D= 8				
IGC-V1 ($C48$)	0.046	1.00	66.52 ± 0.12	48.57 ± 0.53
IGC-V2 ($C80$)	0.046	1.18	$\mathbf{67.65 \pm 0.29}$	$\mathbf{51.49 \pm 0.33}$
D= 20				
IGC-V1 ($C48$)	0.151	2.89	70.87 ± 0.39	54.83 ± 0.27
IGC-V2 ($C80$)	0.144	3.20	$\mathbf{72.63 \pm 0.07}$	$\mathbf{56.40 \pm 0.17}$
D= 26				
IGC-V1 ($C48$)	0.203	3.83	71.82 ± 0.40	56.64 ± 0.15
IGC-V2 ($C80$)	0.193	4.21	$\mathbf{73.49 \pm 0.34}$	$\mathbf{57.12 \pm 0.09}$

$SK_1 = 18$, $K_2 = K_3 = 7$ (9, 11 for other two bigger networks), which is far from the balance condition given in Equation 11. In the second design, we use a channel-wise 3×3 convolution, two group 1×1 convolutions. In this case, $SK_1 = 9$, $K_2 = K_3 = 10$ (13, 16 for other two bigger networks), which satisfies the balance condition. The resulting networks are denoted as IGC-V2 I and IGC-V2 II respectively. The results given in Table 5 show (i) that the first design performs better than IGC-V1 on CIFAR-100 and a little worse than IGC-V1 on CIFAR-10, which might stem from different balance condition satisfaction degrees, and (ii) that the second design performs the best, which stems from the better satisfaction with the balance condition.

In addition, the comparison over 20-layer network shown in Table 3 verifies that our network IGC-V2 with smaller model size as well as less computation complexity, is able to achieve better performance under various widths.

Varying the depth. We also show the performance comparison between IGC-V1 and our network under various depths: $8, 20, 26$. The width C is set to 48 in IGC-V1 (corresponding to IGC-$L24M2$ in [43]) and the width in our network is set to 80. The network structure can be seen in Table 1 and here $K_{s_1}, K_{s_2}, K_{s_3}$ are set to $10, 16, 20$ respectively to satisfy the complementary condition. The results over CIFAR-100 and Tiny ImageNet are given in Table 6, showing superior performance of our network over IGC-V1, which demonstrates the effectiveness of IGC-V2 block.

4.5. Performance Comparison to Small-Model

We show the advantages of our network with small models by comparing to existing state-of-the-art architecture design algorithms. The results are presented in Table 7.

Table 7. Illustrating the advantages of our networks for the small model cases through the classification error comparison to existing state-of-the-art architecture design algorithms.

	D	#Params(M)	C10	C100	Tiny ImageNet
Swapout [33]	20	1.1	6.58	25.86	-
	32	7.4	4.76	22.72	-
DFN [38]	50	3.7	6.40	27.61	-
	50	3.9	6.24	27.52	-
FractalNet [17]	21	38.6	5.22	23.30	-
ResNet [10]	110	1.7	6.41	27.76	-
ResNet(pre-act) [7]	164	1.7	5.46	24.33	-
ResNet [8]	110	1.7	5.52	28.02	46.5
DFN-MR1 [44]	56	1.7	4.94	24.46	-
RiR [36]	18	10.3	5.01	22.90	-
ResNet34 [2]	34	21.4		-	46.9
ResNet18-2× [2]	18	25.7		-	44.6
WRN-32-4 [8]	32	7.4	5.43	23.55	39.63
WRN-40-4 [42]	40	8.9	4.53	21.18	-
DenseNet(k = 12) [8]	40	1.0	-	-	39.09
DenseNet(k = 12) [9]	40	1.0	5.24	24.42	-
DenseNet-BC(k = 12) [9]	100	0.8	4.51	22.27	-
IGC-V2*-$C416$	20	0.65	5.49	22.95	38.81

The observation is that our network with a smaller model achieves similar classification accuracies. For example, on CIFAR-100, the classification error of our approach with $0.65M$ parameters is 22.95%, while Swapout [33] reaches 22.72% with $7.4M$ parameters, FractalNet [17] reaches 23.30% with $38.6M$ parameters, WRN-40-4 [42] reaches 21.18 with $8.9M$ parameters and WRN-32-4 [8] reaches 23.55% with $7.4M$ parameters. On Tiny ImageNet, our network contains the smallest number of parameters, and achieves better performance compared to the reported results. On CIFAR-10, our network achieves competitive performance. In table 7, DenseNet-BC($k = 12$) [9] with more number of parameters achieves lower classification error on CIFAR-100 and CIFAR-10 compared with IGC-V2* (C416). Dense connection is a structure complementary to IGC-V1, and we can also combine densely connected structure with IGC-V1 to improve the performance. We believe that our approach potentially gets more improvement if dense connection and bottleneck design are exploited.

5. Conclusion

In this paper, we aim to eliminate the redundancy in convolution kernels and present an Interleaved Structured Sparse Convolution (IGC-V2) block to compose a dense kernel. We present the complementary condition and the balance condition to guide the design and obtain a balance among model size, computation complexity and classification accuracy. Empirical results show the advantage over IGC-V1 and Xception, and demonstrate that our network with smaller model size achieves similar performance compared with other network structures.

Acknowledgement

This project was supported by the NSFC (U1611461, 61573387). G.-J. Qi was supported by Damo Academy of Alibaba Group during the research was performed

References

[1] F. Chollet. Xception: Deep learning with depthwise separable convolutions. *CoRR*, abs/1610.02357, 2016.

[2] L. Cordeiro. Wide residual network for the tiny imagenet challenge.

[3] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio. Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. *arXiv preprint arXiv:1602.02830*, 2016.

[4] E. L. Denton, W. Zaremba, J. Bruna, Y. LeCun, and R. Fergus. Exploiting linear structure within convolutional networks for efficient evaluation. In *NIPS*, pages 1269–1277, 2014.

[5] S. Han, H. Mao, and W. J. Dally. Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. *CoRR*, abs/1510.00149, 2015.

[6] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.

[7] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In *ECCV*, pages 630–645. Springer, 2016.

[8] G. Huang, Y. Li, G. Pleiss, Z. Liu, J. E. Hopcroft, and K. Q. Weinberger. Snapshot ensembles: Train 1, get m for free. *arXiv preprint arXiv:1704.00109*, 2017.

[9] G. Huang, Z. Liu, and K. Q. Weinberger. Densely connected convolutional networks. *CoRR*, abs/1608.06993, 2016.

[10] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Q. Weinberger. Deep networks with stochastic depth. In *ECCV*, pages 646–661, 2016.

[11] Y. Ioannou, D. P. Robertson, R. Cipolla, and A. Criminisi. Deep roots: Improving CNN efficiency with hierarchical filter groups. *CoRR*, abs/1605.06489, 2016.

[12] Y. Ioannou, D. P. Robertson, J. Shotton, R. Cipolla, and A. Criminisi. Training cnns with low-rank filters for efficient image classification. *CoRR*, abs/1511.06744, 2015.

[13] M. Jaderberg, A. Vedaldi, and A. Zisserman. Speeding up convolutional neural networks with low rank expansions. In *BMVC*, 2014.

[14] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[15] Y. Kim, E. Park, S. Yoo, T. Choi, L. Yang, and D. Shin. Compression of deep convolutional neural networks for fast and low power mobile applications. *CoRR*, abs/1511.06530, 2015.

[16] A. Krizhevsky. Learning multiple layers of features from tiny images. *Technical report*, 2009.

[17] G. Larsson, M. Maire, and G. Shakhnarovich. Fractalnet: Ultra-deep neural networks without residuals. *CoRR*, abs/1605.07648, 2016.

[18] C. Lee, S. Xie, P. W. Gallagher, Z. Zhang, and Z. Tu. Deeply-supervised nets. In *AISTATS*, 2015.

[19] J. Lee, S. Kim, G. Lebanon, and Y. Singer. Local low-rank matrix approximation. In *ICML (2)*, pages 82–90, 2013.

[20] J. Lee, S. Kim, G. Lebanon, Y. Singer, and S. Bengio. LLORMA: local low-rank matrix approximation. *Journal of Machine Learning Research*, 17:15:1–15:24, 2016.

[21] F. Li, B. Zhang, and B. Liu. Ternary weight networks. *arXiv preprint arXiv:1605.04711*, 2016.

[22] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf. Pruning filters for efficient convnets. *CoRR*, abs/1608.08710, 2016.

[23] M. Lin, Q. Chen, and S. Yan. Network in network. *CoRR*, abs/1312.4400, 2013.

[24] B. Liu, M. Wang, H. Foroosh, M. Tappen, and M. Penksy. Sparse convolutional neural networks. In *CVPR*, pages 806–814. IEEE, 2015.

[25] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, and C. Zhang. Learning efficient convolutional networks through network slimming. In *ICCV*, pages 2755–2763, 2017.

[26] J. Luo, J. Wu, and W. Lin. Thinet: A filter level pruning method for deep neural network compression. In *ICCV*, pages 5068–5076, 2017.

[27] F. Mamalet and C. Garcia. Simplifying convnets for fast learning. In *ICANN*, pages 58–65, 2012.

[28] H. Mao, S. Han, J. Pool, W. Li, X. Liu, Y. Wang, and W. J. Dally. Exploring the regularity of sparse structure in convolutional neural networks. *arXiv preprint arXiv:1705.08922*, 2017.

[29] J. Park, S. Li, W. Wen, P. T. P. Tang, H. Li, Y. Chen, and P. Dubey. Faster cnns with direct sparse convolutions and guided pruning. 2016.

[30] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi. Xnornet: Imagenet classification using binary convolutional neural networks. In *ECCV*, pages 525–542. Springer, 2016.

[31] A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio. Fitnets: Hints for thin deep nets. *CoRR*, abs/1412.6550, 2014.

[32] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *IJCV)*, 115(3):211–252, 2015.

[33] S. Singh, D. Hoiem, and D. A. Forsyth. Swapout: Learning an ensemble of deep architectures. In *NIPS*, pages 28–36, 2016.

[34] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. A. Riedmiller. Striving for simplicity: The all convolutional net. *CoRR*, abs/1412.6806, 2014.

[35] R. K. Srivastava, K. Greff, and J. Schmidhuber. Training very deep networks. In *NIPS*, pages 2377–2385, 2015.

[36] S. Targ, D. Almeida, and K. Lyman. Resnet in resnet: Generalizing residual architectures. *CoRR*, abs/1603.08029, 2016.

[37] A. Torralba, R. Fergus, and W. 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.

[38] J. Wang, Z. Wei, T. Zhang, and W. Zeng. Deeply-fused nets. *CoRR*, abs/1605.07716, 2016.

[39] W. Wen, C. Wu, Y. Wang, Y. Chen, and H. Li. Learning structured sparsity in deep neural networks. In *NIPS*, pages 2074–2082, 2016.

[40] W. Wen, C. Xu, C. Wu, Y. Wang, Y. Chen, and H. Li. Coordinating filters for faster deep neural networks. *arXiv preprint arXiv:1703.09746*, 2017.

[41] S. Xie, R. B. Girshick, P. Dollár, Z. Tu, and K. He. Aggregated residual transformations for deep neural networks. *CoRR*, abs/1611.05431, 2016.

[42] S. Zagoruyko and N. Komodakis. Wide residual networks. *CoRR*, abs/1605.07146, 2016.

[43] T. Zhang, G. Qi, B. Xiao, and J. Wang. Interleaved group convolutions for deep neural networks. *CoRR*, abs/1707.02725, 2017.

[44] L. Zhao, J. Wang, X. Li, and Z. Tu. Deep convolutional neural networks with merge-and-run mappings. *CoRR*, abs/1611.07718, 2016.

[45] A. Zhou, A. Yao, Y. Guo, L. Xu, and Y. Chen. Incremental network quantization: Towards lossless cnns with low-precision weights. *arXiv preprint arXiv:1702.03044*, 2017.

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

[47] C. Zhu, S. Han, H. Mao, and W. J. Dally. Trained ternary quantization. *arXiv preprint arXiv:1612.01064*, 2016.