Fully Learnable Group Convolution for Acceleration of Deep Neural Networks

Xijun Wang[1,2] Meina Kan[1] Shiguang Shan[1,2,3] Xilin Chen[1,2]
[1]Key Lab of Intelligent Information Processing of Chinese Academy of Sciences (CAS),
Institute of Computing Technology, CAS, Beijing 100190, China
[2]University of Chinese Academy of Sciences, Beijing 100049, China
[3]CAS Center for Excellence in Brain Science and Intelligence Technology, Shanghai 200031, China
xijun.wang@vipl.ict.ac.cn {kanmeina,sgshan,xlchen}@ict.ac.cn

Abstract

Benefitted from its great success on many tasks, deep learning is increasingly used on low-computational-cost devices, e.g. smartphone, embedded devices, etc. To reduce the high computational and memory cost, in this work, we propose a fully learnable group convolution module (FLGC for short) which is quite efficient and can be embedded into any deep neural networks for acceleration. Specifically, our proposed method automatically learns the group structure in the training stage in a fully end-to-end manner, leading to a better structure than the existing pre-defined, two-steps, or iterative strategies. Moreover, our method can be further combined with depthwise separable convolution, resulting in 5× acceleration than the vanilla Resnet50 on single CPU. An additional advantage is that in our FLGC the number of groups can be set as any value, but not necessarily 2^k as in most existing methods, meaning better tradeoff between accuracy and speed. As evaluated in our experiments, our method achieves better performance than existing learnable group convolution and standard group convolution when using the same number of groups.

1. Introduction

Since the Alexnet proposed by Krizhevsky *et al*. [23] achieved breakthrough results in the 2012 ImageNet Challenge, deeper and larger convolutional neural networks (CNNs) have become a ubiquitous setting for better performance, especially on tasks with big data [5, 26]. However, even an ordinary CNNs is usually with dozens, hundreds or even thousands of layers and thousands of channels [12, 35, 18]. Such huge parameters and high computational cost make it insupportable on devices with limited hardware resources or applications with strict latency requirements. In [6], Misha Denil *et al*. found that there is significant redundancy in CNNs, and the accuracy will not drop even many of the network parameters are not learned or removed. After that, various methods of reducing redundancy have emerged. These methods can be roughly grouped into two categories, post-processing methods such as pruning or quantizing a pre-trained deep model, and efficient architecture design methods attempting to design fast and compact deep network.

1.1. Post-processing methods

A straightforward strategy is to post-process a pre-trained model, such as pruning the parameters, or quantizing the model by using fewer bits.

Parameter Pruning. Some *fine-grained methods* attempt to prune the wispy connections between two neural nodes based on its importance, and thus convert a dense network to a sparse one [27, 11, 10, 24]. A typical one is [11], in which Han *et al*. proposed to learn the importance of each connection and then those unimportant connections are removed to reduce the operations. Furthermore, Guo *et al*. [10] proposed an on-the-fly connection pruning method named dynamic network surgery, which can avoid incorrect pruning and make it as a continual network maintenance by incorporating connection splicing into the whole process. The sparse network achieved from the fine-grained pruning methods has much lower computation cost theoretically. Unfortunately, there is no mature framework or hardware for sparse network, and thus only limited speed up can be obtained practically.

There are also some other methods attempting to do *coarse-grained pruning* by cutting off the entire filters, channels or even layers. In [14], He *et al*. proposed an iterative two-step algorithm to effectively prune each layer by using a LASSO regression, which based on the channel selection and least square reconstruction. In [25], Li *et al*. applied L1 regularization to prune filters to induce sparsity. More generally, Wen *et al*. [38] proposed a structured sparsity learning method to reduce redundant filters, channels, and layers in a unified manner. Coarse-grained prun-

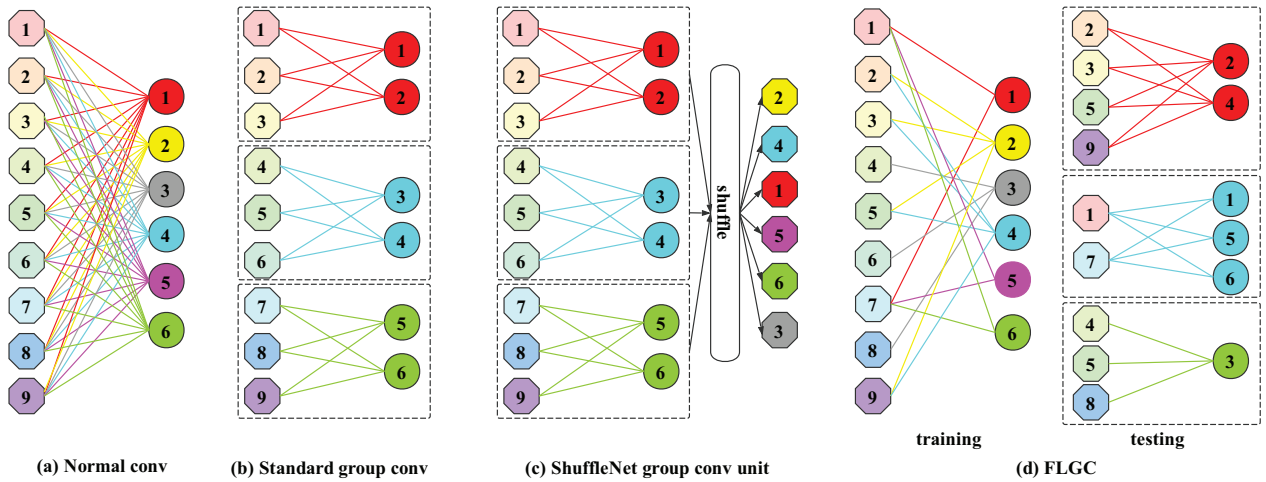

Figure 1. An overview of different group convolution mechanisms. (a) is a normal convolution. (b) is a standard group convolution, in which input channels and filters in each group are both fixed. (c) is ShuffleNet group convolution unit, in which input channels are fixed. (d) is our FLGC convolution, in which the grouping structure including both input channels and filters in each group is dynamically learnt. The octagons represent the input channels and the circles represent the filters.

ing methods directly remove the filters or channels and thus effectively accelerate the network.

Quantization. Network quantization is to reduce the number of bits used to represent the parameters or features. Han *et al*. [1] proposed a deep compression method that firstly pruned the insignificant connections, and secondly quantized the left connections by using weight sharing and Huffman coding. Moreover, INQ [47] and ShiftCNN [9] quantize a full-precision CNN model to a low-precision model whose parameters (i.e. weights) are either zero or powers of two. With the powers of two representation, the multiplication operations can be replaced by shift operations which is quite efficient. Besides these post-quantization methods, there are also some methods attempting to directly train a binary network, such as BinaryConnect [3], BNNs [4] and XNORNetworks [30]. As quite fewer bits used, all these methods can obtain faster networks, but correspondingly the accuracy usually is significantly decreased when dealing with large networks.

Other Methods. In addition to the methods described above, some other approaches explored how to use low-rank factorization, knowledge distillation for deep network acceleration. In *Low-rank decomposition* methods [7, 22], the convolutional filters structured in 4D tensors are decomposed to lower-rank filters, which removes the redundancy in convolution inducing fewer calculations. In *knowledge distillation* methods [15, 31], the knowledge learnt from a deep and wide network is shifted into shallower and narrower ones by making the output distribution of the two networks the same.

The post-processing methods are simple and intuitive, but obviously have some *limitations*. Most above methods are in two or multiple steps manner, the objective of the network (such as classification or detection) and the objective of acceleration are separately optimized. Therefore, the acceleration does not necessarily ensure excellent classification or detection accuracy. Besides, most pruning methods determine the importance of a connection or layer by only considering its magnitude and its contribution to several adjacent layers, but not its influence on the whole network. As verified in [43], pruning without considering the global impact will result in significant error propagation, causing performance degeneration especially in deep networks.

1.2. Design Efficient Architectures

Considering the above mentioned limitations, some researchers go other way to directly design efficient network architectures, such as smaller filters, separable convolution, group convolution, and etc.

Separable Convolution. Some early works straightforwardly employ small filters (e.g. 1×1, 3×3) to replace those large ones (e.g. 5×5, 7×7) for acceleration [33, 12, 20, 18]. However, even if only with 3×3 and 1×1 filters, an ordinary deep network is still time consuming, such as the ResNet50 needs about 4G MAdds[1] and VGG16 needs 16G MAdds for calculating a 224×224 image. In order to get further acceleration, some works explore separable convolution which uses multiple 2D con-

[1]In this paper, MAdds refers to the number of multiplication-addition operations.

volutions to replace the time-intensive 3D convolutions. In aspect of spatial separation, Inception V3 [37] factorizes a $h \times w \times c$ filter into two ones, i.e. one $h \times 1 \times c$ filter and another $1 \times w \times c$ filter. In aspect of channel separation, Xception [2] and MobileNets [16] employ depthwise separable convolution. This kind of separable convolution can speed up the computing exponentially, and thus they are widely used in many modern networks.

Group Convolution. Separable convolution achieves the acceleration by factorizing the filters. Differently, the group convolution speed up the network by dividing all filters into several groups, such as [21, 34, 40, 45, 41, 46]. The concept of group convolution was first proposed in Alexnet [23], and then it is further successfully adopted in ResNeXt [41], making it popular in recent network design. However, standard group convolutions do not communicate between different groups, which restricts their representation capability. To solve this problem, in ShuffleNet [46], a channel shuffle unit is proposed to randomly permute the output channels of group convolution to make the output better related to the input. In these methods, the elements (i.e. input channels and filters) in each group are fixed or randomly defined. Furthermore, in Condensenet [17] a learnable group convolution was proposed to automatically select the input channels for each group.

Although the existing group convolution methods have advanced the acceleration very effectively, there are still several limitations to solve: 1) The filters used for group convolution in each group are pre-defined and fixed, and this hard assignment hinders its representation capability even with random permutation after group convolution; 2) In some works the groups are learnable, but usually are designed as a tedious multiple-stage or iterative manner. In each stage, the network from previous stage is firstly pruned and then fine-tuned to recover the accuracy.

To deal with all above limitations once for all, in this work we propose a fully learnable group convolution (FLGC) method. In our proposed FLGC, the grouping structure including the input channels and filters in one group is dynamically optimized. What's more, this module can be embedded into any existing deep network and easily optimized in an end-to-end manner. At test time, the learnt model is calculated similar as the standard group convolution which allows for efficient computation in practice. A brief comparison of different group convolution methods are shown in Figure 1. Overall, the advantages of our method are as follows:

(1) The element including input channels and filters in each group are both learnable, allowing for flexible grouping structure and inducing better representation capability;

(2) The group structure in all layers are simultaneously optimized in an end-to-end manner, rather than a multiple-stage or iterative manner (i.e. pruning layer by layer.);

(3) The numbers of input channels and filters in each group are both flexible, while the two numbers must be divisible by the group number in conventional group convolution.

2. Fully Learnable Group Convolution(FLGC)

In modern deep networks, the size of filters is mostly 3×3 or 1×1, and the main computational cost is from the convolution layer. The 3×3 convolutions can be easily accelerated by using the depthwise separable convolution (DSC). And the separation of 3×3 convolutions will come along with additional 1×1 convolutions. After DSC, the 1×1 convolutions contribute the major time-cost, e.g. for a Resnet50 network, after applying DSC to the 3×3 convolutions, the computational cost of 1×1 convolutions accounts for 83.6% in the whole network. Therefore, how to speed up the 1×1 convolution becomes an urgent problem and attracts increasing attentions.

Since the 1×1 filters are non-separable, group convolution becomes a hopeful and feasible solution. However, simply applying group convolution will result in drastic precision degradation. As analyzed in [17], this is caused by the fact that the input channels to the 1×1 convolutional layer have an intrinsic order or they are far more diverse. This implies that the hard assignment grouping mechanism in standard group convolution is incompetent. For a better solution, our proposed method dynamically determines the input channels and filters for each group, forming a flexible and efficient grouping mechanism.

Briefly, in our FLGC the input channels and filters in one group (i.e. the group structure) are both dynamically determined and updated according to the gradient of the overall loss of the network through back propagation. And thus it can be optimized in an end-to-end manner.

2.1. Method

In a deep network, the convolution layer is computed as convolving the input feature maps with filters. Taking the k^{th} layer for an example, the input of the k^{th} layer can be denoted as $X^k = \{x_1^k, x_2^k, \cdots, x_C^k\}$, where C is the number of channels and x_i^k is the i^{th} feature map. The filters of the k^{th} layer are denoted as $W^k = \{w_1^k, w_2^k, \cdots, w_N^k\}$, where N denotes the number of filters, i.e. number of output channels, and w_i^k is the i^{th} 3D convolutional filter. The output[2] of this convolution layer is calculated as follows:

$$\begin{aligned} X^{k+1} &= W^k \otimes X^k \\ &= \{w_1^k * X^k, w_2^k * X^k, \cdots, w_N^k * X^k\}, \end{aligned} \quad (1)$$

where $\otimes$ in this work denotes the convolution between two sets, $*$ denotes the convolution operation between a filter and the input feature maps.

[2] We omit the bias b for simplicity.

In group convolution, the input channels and filters are divided into G groups respectively, denoted as $X^k = \{X_1^k, X_2^k, \cdots, X_G^k\}$[3] and $W^k = \{W_1^k, W_2^k, \cdots, W_G^k\}$[4]. Now, X^{k+1} is reformulated as below:

$$X^{k+1} = \{W_1^k \otimes X_1^k, W_2^k \otimes X_2^k, \cdots, W_G^k \otimes X_G^k\}. \quad (2)$$

Typically, in standard group convolution the input channels and filters are evenly divided into G groups in a hard assignment manner, i.e. $\frac{C}{G}$ input channels and $\frac{N}{G}$ filters in each group. Therefore, the number of channels used in each filter is reduced to $\frac{1}{G}$ of original ones, resulting in a acceleration rate as below:

$$\frac{\mathit{MAdds}(W^k \otimes X^k)}{\mathit{MAdds}(\sum_{i=1}^{G} W_i^k \otimes X_i^k)} = G. \quad (3)$$

As can be seen, this group convolution from hard assignment can easily bring about considerable acceleration of $G\times$. However, it is not necessarily a promising approach for accuracy. Therefore, *the goal of our method is to design a fully learnable grouping mechanism, where the grouping structure is dynamically optimized for favorable acceleration as well as accuracy.*

Firstly, we formulate the grouping structure in the k^{th} layer as two binary selection matrices for input channels and filters respectively, denoting as S^k and T^k.

The S^k is a matrix for channel selection in shape of $C \times G$, with each element defined as:

$$S^k(i,j) = \begin{cases} 1, \text{ if } x_i^k \in X_j^k, \\ 0, \text{ if } x_i^k \notin X_j^k, \end{cases} \quad i = [1, C]; j \in [1, G]. \quad (4)$$

in which $S^k(i,j) = 1$ means the i^{th} input channel is selected into the j^{th} group. As can be seen, the j^{th} column of S^k indicates which input channels belong to the j^{th} group. Then, X_j^k can be simply represented as follows:

$$X_j^k = X^k \odot S^k(:,j)^{\mathcal{T}}, j \in [1, G], \quad (5)$$

where $\odot$ denotes the element-wise selection operator and the element here means $\forall x_i^k \in X^k$, and $\mathcal{T}$ denotes the transpose of a vector.

Similarly, for filter selection we define a matrix T^k in shape of $N \times G$, with each element defined as:

$$T^k(i,j) = \begin{cases} 1, \text{ if } w_i^k \in W_j^k, \\ 0, \text{ if } w_i^k \notin W_j^k, \end{cases} \quad i = [1, N]; j \in [1, G]. \quad (6)$$

in which $T^k(i,j) = 1$ means the i^{th} filter is selected into the j^{th} group. The j^{th} column of T^k indicates which filters belong to the j^{th} group. Then the j^{th} group of filters, i.e. W_j^k, can be represented as:

$$W_j^k = W^k \odot T^k(:,j)^{\mathcal{T}}, j \in [1, G]. \quad (7)$$

As a result, the overall group convolution in Eq.(2) can be re-formulated as follows:

$$\begin{aligned} X^{k+1} =& W^k \otimes X^k \\ =& \{W_1^k \otimes X_1^k, W_2^k \otimes X_2^k, \cdots, W_G^k \otimes X_G^k\} \\ =& \{W^k \odot T^k(:,1)^{\mathcal{T}} \otimes X^k \odot S^k(:,1)^{\mathcal{T}}, \cdots, \\ & W^k \odot T^k(:,G)^{\mathcal{T}} \otimes X^k \odot S^k(:,G)^{\mathcal{T}}\}. \end{aligned} \quad (8)$$

With Eq.(8), the structure of group convolution is parameterized by two binary selection matrix S^k and T^k. Therefore, this parameterized group convolution can be embedded in any existing deep networks with the objective as:

$$\min_{W^k, S^k, T^k|_{k=1}^K} \frac{1}{n} \sum_{i=1}^{n} L(Y_i; \hat{Y} | X_i, W^k, S^k, T^k), \quad (9)$$

in which X_i denotes the i^{th} input sample, n indicates the number of training data, Y_i indicates the i^{th} sample's true category label, K is the number of layers, and $\hat{Y}$ is the label predicted from a network with our group convolution parameterized by W^k, S^k, T^k. $L(,)$ denotes the loss function (e.g. cross entropy loss) for classification or detection etc.

In the above objective, the filters W^k, the group structure including S^k and T^k can be all automatically optimized according to the overall objective function. However, binary variables are notorious for its non-differential feature. So, we further design an ingenious approximation to make it differentiable for better optimization in section 2.2.

As can be seen from Eq.(9), the group structure in our method is automatically optimized rather than manually defined. Furthermore, different from those methods only considering the magnitude and impact of the connection in one or two layers, the group structure in our method is determined according to the objective loss of the whole network. Therefore, the group structures of all layers in our method are jointly optimized implying a superior solution.

2.2. Optimization

In Eq.(9), the filters W^k can be easily optimized as most deep networks by using the stochastic gradient descent, while the binary parameters are hard to optimize as they are non-differentiable. To solve this problem, we approximate the S^k and T^k by applying a softmax function to a meta selection matrix to make it differentiable.

Specifically, we introduce a meta selection matrix $\bar{S}^k$ for channel selection with the same shape as S^k. And then the softmax function is applied to each row of $\bar{S}^k$, which can map it to (0,1) as below:

$$\hat{S}^k(i,:) = softmax(\bar{S}^k(i,:)), i \in [1, C]. \quad (10)$$

[3] $X_1^k \cup X_2^k \cup \cdots \cup X_G^k = X^k$
[4] $W_1^k \cup W_2^k \cup \cdots \cup W_G^k = W^k$

Here, the meta selection matrix $\bar{S}^k$ can be initialized as Gaussian distribution or results from other methods. After softmax, the i^{th} row of $\hat{S}^k$ indicates the probability that the i^{th} input channel belongs to each group. So, the i^{th} input channel can be simply selected into the group with highest probability. That is to say, the binary selection matrix S^k can be approximated as:

$$S^k(i,j) = \begin{cases} 1, \text{ if } \hat{S}^k(i,j) = max(\hat{S}^k(i,:)), \\ 0, \text{ otherwise.} \end{cases} \tag{11}$$

The reason of using softmax function is that with softmax operation the meta selection matrix $\bar{S}^k$ can be updated to make the output $\hat{S}^k$ approximating 0 or 1 as close as possible. As a result, the quantization error between $\bar{S}^k$ and S^k is largely narrowed.

Similarly, the binary selection matrix T^k is approximated by applying softmax function on a meta selection matrix $\bar{T}^k$ for filter selection as follows:

$$T^k(i,j) = \begin{cases} 1, \text{ if } \hat{T}^k(i,j) = max(\hat{T}^k(i,:)), \\ 0, \text{ otherwise,} \end{cases} \tag{12}$$

with

$$\hat{T}^k(i,:) \quad = \quad softmax(\bar{T}^k(i,:)),\ i \in [1,N]. \tag{13}$$

Here, the i^{th} row of $\hat{T}^k$ indicates the probability that the i^{th} filter belongs to each group.

In summary, with the above Eq.(10), Eq.(11), Eq.(12)and Eq.(13), the differentiation of the binary S^k and T^k are shifted to the differentiation of the meta selection variable $\bar{S}^k$ and $\bar{T}^k$ which are non-binary, yet with small quantization error.

Furthermore, for easy implementation, Eq.(8) is equivalently transformed to the following formulation:

$$\begin{aligned} X^{k+1} =& \{W^k \odot T^k(:,1)^{\mathcal{T}} \otimes X^k \odot S^k(:,1)^{\mathcal{T}}, \cdots, \\ & W^k \odot T^k(:,G)^{\mathcal{T}} \otimes X^k \odot S^k(:,G)^{\mathcal{T}}\} \\ =& (W^k \odot M^k) \otimes X^k, \end{aligned} \tag{14}$$

with $M^k = T^k(S^k)^{\mathcal{T}}$, and the shape of M^k is $N \times C$ that is the same as W^k.

Finally, the objective function is re-formulated as below:

$$\min_{W,\bar{S},\bar{T}} \frac{1}{n} \sum_{i=1}^{n} L(Y_i, X_i(W \odot M) + b), \tag{15}$$

where $W = \{W^k|_{k=1}^K\}$, $\bar{S} = \{\bar{S}^k|_{k=1}^K\}$, $\bar{T} = \{\bar{T}^k|_{k=1}^K\}$.

The objective in Eq.(15) can be easily optimized as most deep network by using the stochastic gradient descent method, with the parameters of each layer are updated as follows:

$$W^k_{(i,j)} \leftarrow W^k_{(i,j)} - \eta \frac{\partial L}{\partial \left(W^k_{(i,j)} \odot M^k_{(i,j)}\right)}, \forall i,j \in I, \tag{16}$$

$$\begin{aligned} \bar{S}^k_{(i,j)} \leftarrow & \bar{S}^k_{(i,j)} - \eta \frac{\partial L}{\partial \left(W^k_{(i,j)} \odot M^k_{(i,j)}\right)} \\ & \frac{\partial \left(W^k_{(i,j)} \odot M^k_{(i,j)}\right)}{\partial M^k_{(i,j)}} \frac{\partial M^k_{(i,j)}}{\partial \hat{S}^k_{(i,j)}} \frac{\partial \hat{S}^k_{(i,j)}}{\partial \bar{S}^k_{(i,j)}}, \end{aligned} \tag{17}$$

$$\begin{aligned} \bar{T}^k_{(i,j)} \leftarrow & \bar{T}^k_{(i,j)} - \eta \frac{\partial L}{\partial \left(W^k_{(i,j)} \odot M^k_{(i,j)}\right)} \\ & \frac{\partial \left(W^k_{(i,j)} \odot M^k_{(i,j)}\right)}{\partial M^k_{(i,j)}} \frac{\partial M^k_{(i,j)}}{\partial \hat{T}^k_{(i,j)}} \frac{\partial \hat{T}^k_{(i,j)}}{\partial \bar{T}^k_{(i,j)}}, \end{aligned} \tag{18}$$

in which η indicates the learning rate. The overall procedure is summarized in Algorithm 1.

Algorithm 1 FLGC: solving the optimization problem in Eq.(15) via SGD

Input: X: training data, Y: lable
Output: $\{W^k, S^k, T^k : k \in [1,K]\}$

Initialize $W^k \leftarrow msra$; $\bar{S}^k, \bar{T}^k \leftarrow Gaussian$; $S^k, T^k \leftarrow 0$
for each batch X_i **do**
  //Forward propagation:
  for $i = 1 \rightarrow C$ **do**
    $\hat{S}^k(i,:) \leftarrow softmax(\bar{S}^k(i,:))$
    $S^k(i,j) \leftarrow 1$,if $\hat{S}^k(i,j) = max(\hat{S}^k(i,:))$
  end for
  for $i = 1 \rightarrow N$ **do**
    $\hat{T}^k(i,:) \leftarrow softmax(\bar{T}^k(i,:))$
    $T^k(i,j) \leftarrow 1$,if $\hat{T}^k(i,j) = max(\hat{T}^k(i,:))$
  end for
  $M^k \leftarrow T^k(S^k)^T$
  Get loss: $L = L(Y_i, X_i(W^k \odot M^k) + b)$
  //Backward propagation:
  $W^k \leftarrow W^k - \eta \frac{\partial L}{\partial (W^k \odot M^k)}$
  $\bar{S}^k \leftarrow \bar{S}^k - \eta \frac{\partial L}{\partial (W^k \odot M^k)} \frac{\partial (W^k \odot M^k)}{\partial M^k} \frac{\partial M^k}{\partial \hat{S}^k} \frac{\partial \hat{S}^k}{\partial \bar{S}^k}$
  $\bar{T}^k \leftarrow \bar{T}^k - \eta \frac{\partial L}{\partial (W^k \odot M^k)} \frac{\partial (W^k \odot M^k)}{\partial M^k} \frac{\partial M^k}{\partial \hat{T}^k} \frac{\partial \hat{T}^k}{\partial \bar{T}^k}$
end for

2.3. Inference with Index-Reordering

After the group structure is learnt, the input channels and filters usually need to be re-organized for fast inference. A naive method is to add an index layer to re-order the input

channels according to the group information, and another index layer to re-order the filters. Then, the output channels are re-ordered back to the original order, as shown in Figure 2(a). Unfortunately, such frequent re-order operations on memory will significantly increase the inference time.

Therefore, we propose an efficient strategy for index re-ordering as shown in Figure 2(b). Firstly, the filters are re-ordered to make those filters in one group arranged together. Secondly, considering that the input channels are also the output channels of previous layer, we merge the index of the output from previous layer and index of input channels in this layer as single index to obtain correct order of input channels. The detail is shown in Figure 2(c). As designed like above, the operations on memory are reduced a lot and all these re-ordering index can be obtained offline, so it is quite efficient at the inference stage.

As a result, at the inference time our FLGC can be as efficient as the standard group convolution.

3. Experiments

In this section, we investigate the effectiveness of our proposed FLGC by embedding it into the existing popular CNNs networks including Resnet50 [13], MobileNetV2 [32] and Condensenet [17]. Firstly, we conduct ablation study of FLGC on CASIA-WebFace [42], and then compare it with existing competitive methods on CASIA-WebFace, CIFAR-10 and ImageNet (ILSVRC 2012) [5] in terms of face verification and image classification.

3.1. Embedding into the state-of-the-art CNNs

We select three state-of-the-art architectures including Resnet50, MobileNetV2 and CondenseNet to embed the proposed fully learnable group convolution(FLGC) for evaluation.

Resnet50 with FLGC. The Resnet50 is a powerful network which achieves prominent accuracy on many tasks. Nevertheless, it is quite time-consuming. As shown in Figure 3(blue line), the major computation cost falls on the 3×3

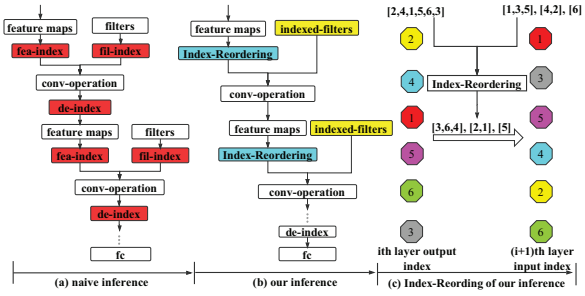

Figure 2. Illustration of index re-ordering for efficient inference. (a) is a naive inference method, (b) is our efficient inference method, (c) is Index-Reordering unit.

convolutions, and thus we firstly use the DSC to separate the 3×3 convolutions following MobileNet [16]. After DSC, there are a large number of 1×1 convolutions, which computational cost accounts for 83.6% of the whole network. Therefore, we further replace all the 1×1 layers in the network with our FLGC layers. Besides, we simply double the stride of the first layer and add a fc layer.

MobileNetV2 with FLGC. The MobileNetV2 is a state-of-the-art efficient architecture with elaborative design. This architecture achieves satisfactory accuracy on many tasks with favorable computational cost, e.g. classification, detection and segmentation. But still, the intensive 1×1 convolutions takes great majority of computational cost, leaving much room for further acceleration. Therefore, we replace those 1×1 convolution layers, of which the filters number is larger than 96, with our FLGC layer.

CondenseNet with FLGC. CondenseNet proposed a learnable group convolution which can automatically select the input channels for each group. However, the filters in each group are fixed, and the process are designed as a tedious multiple-stage or iterative manner. Besides, the importance of each input channel is determined according to the magnitude of the connections between the input and the filters, but without considering its impact on the overall network, leading to a sub-optimal solution. We substitute all the FLGC for the LGC in CondenseNet.

3.2. Ablation Study

The ablation experiment is conducted on CASIA-WebFace with Resnet50 in terms of face verification. The experimental setting on this dataset is the same as that in section 3.1.

Firstly, we analyze the speedup with DSC and our FGLC by comparing with the standard convolution. The time cost of each layer in all methods are shown in Figure 3. As can be seen, in the standard Resnet50 denoted in the blue line, 3×3 convolution layer is the major time-consuming part. After applying DSC, 3×3 convolution time cost is significantly reduced as shown in the orange line, and the orange line also highlights that 1×1 convolution layer is the major time cost part now. By further applying FLGC, the time cost of 1×1 convolution layer is successfully reduced as shown in the green line, resulting in a quite efficient architecture with comparable accuracy as the baseline(standard Resnet50). For overall procedure, our method achieves a significant improvement of time cost.

Besides the efficiency, we further explore the accuracy of standard group convolution and our FLGC w.r.t. different number of groups, and the result are shown in Figure 4, Table 1 and Table 2. As can be seen, the accuracy drops dramatically when the standard group convolution is applied to the 1×1 convolution, mainly due to the loss of representation capability from hard assignment. Differently, our

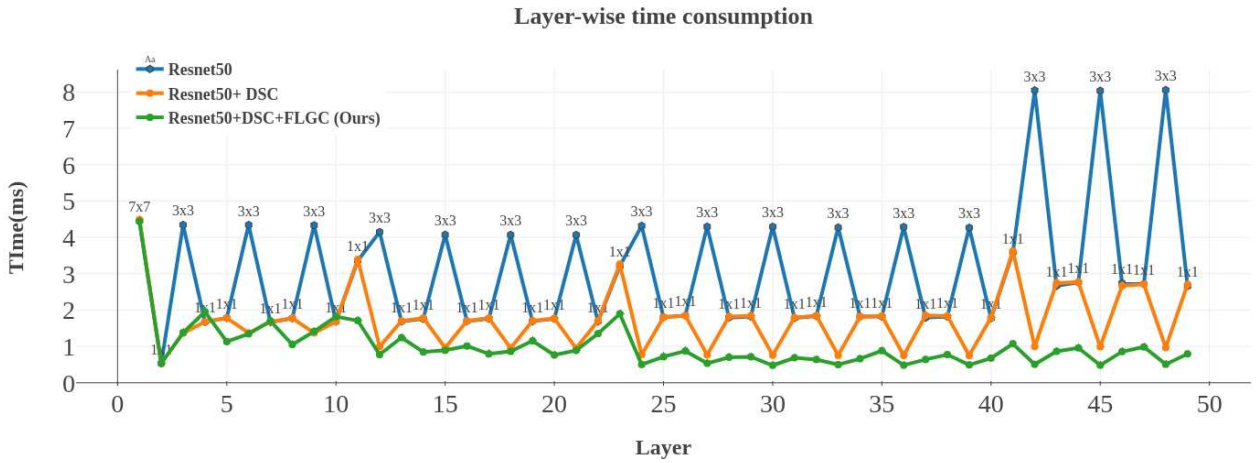

Figure 3. The time cost of each convolutional layer in Resnet50 with different convolution mechanisms on a single CPU. The blue line is the Standard Resnet50. The orange line is the Resnet50 with 3×3 convolutions replaced by DSC. The green line is the Resnet50 with 1×1 convolutions further replaced by FLGC.

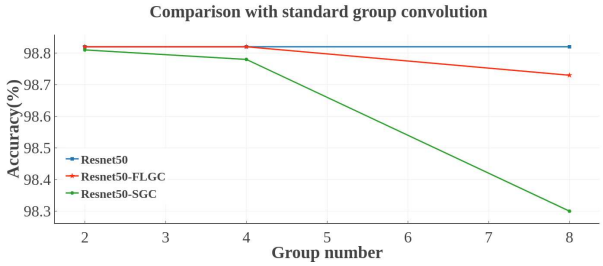

Figure 4. Compare our FLGC with standard group convolution(SGC) in terms of face verification accuracy of Resnet50 on CASIA-WebFace w.r.t. different group numbers.

FLGC successfully maintains the accuracy even with large number of groups, benefitted from the fully learnable mechanism for grouping structure.

3.3. Comparison with competitive approaches

Results on CASIA-WebFace. The CASIA-WebFace is a commonly used wild dataset for face verification, consisting of 494,414 face images from 10,575 subjects. All faces are detected and aligned by using [39] and [44], and then the detected faces are cropped out in resolution of 256×256. This dataset is used for training. Following the mainstream works, the well-known LFW [19] dataset is used for face verification evaluation. LFW includes 13,233 face images from 5749 different identities, and the standard protocol defines 3,000 positive pairs and 3,000 negative pairs for verification testing.

On this dataset, we embed the proposed FLGC into the Resnet50 as described in section 3.1. For optimization of our method, we initialize the meta selection matrix $\bar{S}^k$ and $\bar{T}^k$ with Gaussian distribution, and simply set the hyperparameters of momentum as 0.9, weight decay as 5×10^{-4}, batch size as 80, and iterations as 120,000. Two versions of our FLGC with group number as 4 and 8 are evaluated respectively.

Our accelerated network is compared with several state-of-the-art methods on this dataset including [42, 28, 8, 29]. All methods for comparison including ours employ softmax loss for optimization. The experimental results are shown in Table 1. As can be seen, the standard Resnet50 achieves better verification rate with giant architecture than [42, 28, 8, 29], inevitably leading to high computational cost. Expectedly, our modified Resnet50 achieves about 18x speed up over standard Resnet50 without accuracy drop, which is also much faster than [42, 28, 8, 29]. In practical evaluation on single CPU(Intel(R) Xeon(R) CPU E5-2620 v3 @2.40GHz), our modified Resnet50 runs 5x faster than standard Resnet50, demonstrating the effectiveness of our method.

Results on CIFAR-10. We further compare our FLGC with other acceleration approaches on CIFAR-10 dataset. CIFAR-10 consists of 10 classes and 60,000 images in resolution of 32×32 pixels. Among them, 50,000 images are used for training and 10,000 for testing.

Since the image resolution on this dataset is small, the modified Resnet50 in Section 3.1 used for 224×224 image is too large and redundant. So, we replace the 7×7 convolution layer with 3×3 convolution layer to suit the smaller input images. Based on this baseline architecture, we replace the 1x1 convolution layers with FLGC layers, and the number of group is set as 4. For clear comparison, two versions of FLGC with different MAdds by changing number of filters is proposed, referred to as ResNet50-FLGC1 and ResNet50-FLGC2. Besides Resnet50, we also embed our FLGC in the state-of-the-art acceleration architecture

Table 1. Face verification accuracy (%) and time complexity on LFW, all the medels are trained on CASIA-WebFace. The archtecture of ResNet50-FLGC and ResNet50-SGC are introduced in Section 3.1. (SGC: standard group convolution)

Model	MAdds	Params	Acc
Yi *et al*. [42]	770M	1.75M	97.73
64layer+Softmax [28]	28460M	37.16M	97.88
Ding *et al*. [8]	2874M	3.76M	98.43
Liu *et al*. [29]	10194M	6.78M	98.71
ResNet50(stardand)	3727M	20.69M	98.82
ResNet50-SGC(G=2)	363M	5.35M	98.81
ResNet50-FLGC(G=2)	**363M**	**5.35M**	**98.82**
ResNet50-SGC(G=4)	203M	2.70M	98.78
ResNet50-FLGC(G=4)	**203M**	**2.70M**	**98.82**
ResNet50-SGC(G=8)	124M	1.37M	98.30
ResNet50-FLGC(G=8)	**124M**	**1.37M**	**98.73**

MobileNetV2, referred to as MobileNetV2-FLGC. For optimization of our method, all hyperparameters is the same as that used on CASIA-WebFace.

On this dataset, we compare the FLGC with state-of-the-art filter level pruning methods and the state-of-the-art architecture MobileNetV2. The comparison results are shown in Table 2. Comparing with the pruning methods [14, 25] which also employ the Resnet architecture, we can get lower classification error with 3× fewer MAdds. Besides, our FLGC can be flexibly embedded into any efficient architectures such as MobileNetV2, leading to further speedup. As can be seen in Table 2, MobileNetV2 with FLGC achieves better accuracy w.r.t different group number, further demonstrating the superiority of our proposed FLGC.

Table 2. Image classification error(%) and time complexity of different methods on CIFAR-10.(G:group number)

Model	MAdds	Params	Err
ResNet56-pruned [14]	62M	—	8.2
ResNet50-FLGC1(ours)	**23M**	**0.22M**	**7.95**
ResNet56-pruned [25]	90M	0.73M	6.94
ResNet50-FLGC2(ours)	**44M**	**0.68M**	**6.77**
MobileNetV2-SGC(G=2)	158M	1.18M	6.04
MobileNetV2-FLGC(G=2)	158M	1.18M	**5.89**
MobileNetV2-FLGC(G=3)	122M	0.85M	**5.80**
MobileNetV2-SGC(G=4)	103M	0.68M	6.64
MobileNetV2-FLGC(G=4)	103M	0.68M	**5.84**
MobileNetV2-FLGC(G=5)	92M	0.58M	**6.12**
MobileNetV2-FLGC(G=6)	85M	0.51M	**6.33**
MobileNetV2-FLGC(G=7)	80M	0.46M	**6.34**
MobileNetV2-SGC(G=8)	76M	0.43M	7.51
MobileNetV2-FLGC(G=8)	76M	0.43M	**6.91**

Results on ImageNet. To further validate the effectiveness of our proposed FLGC, we compare our FLGC with state-of-the-art learnable group convolution which proposed in CondenseNet [17] on ImageNet.

Table 3. Comparison of Top-1 and Top-5 classification error rate (%) with other state-of-the-art compact models on ImageNet.

Model	MAdds	Params	Top1	Top5
Inception V1[36]	1448M	6.6M	30.2	10.1
1.0 MobileNet-224[16]	569M	4.2M	29.4	10.5
ShuffleNet 2x[46]	524M	5.3M	26.3	—
NASNet-A (N=4)[48]	564M	5.3M	26.0	8.4
NASNet-B (N=4)[48]	488M	5.3M	27.2	8.7
NASNet-C (N=4)[48]	558M	4.9M	27.5	9.0
CondenseNet (G=4)[17]	529M	4.8M	26.2	8.3
CondenseNet-SGC	529M	4.8M	29.0	9.9
CondenseNet-FLGC	**529M**	**4.8M**	**25.3**	**7.9**

For a fair comparison, we used the same network structure as CondenseNet. Based on this baseline architecture, we replace the LGC layers in CondenseNet with our FLGC layers and standard group convolution (SGC) layers respectively, and the number of group is set as 4. What's more, we keep the hyperparameters the same as that used in CondenseNet. All models are trained for 120 epochs, with a cosine shape learning rate which starts from 0.2 and gradually reduces to 0. As can be seen in Table 3, our FLGC achieves better accuracy than CondenseNet's LGC and SGC. Moreover, Our FLGC even achieves a favorable performance compared with competitive MobileNet, ShuffleNet and NASNet-A.

4. Conclusion

In this work, we propose a fully learnable group convolution module which is quite efficient and can be embedded into any layer of any deep neural networks for acceleration. Instead of the existing pre-defined, two-steps, or iterative acceleration strategies, FLGC can automatically learn the group structure at the training stage according to the overall loss of the network in a fully end-to-end manner, and run as efficient as standard group convolution at the inference stage. The number of input channels and filters in each group are flexible, which ensures better representation capability and well solves the problem of uneven information distribution encountered in standard group convolution. Furthermore, compared with LGC of CondenseNet and standard group convolution, our FLGC can better maintain the accuracy while achieve significant acceleration even with large number of groups.

Acknowledgements

This work is partially supported by the National Key R&D Program of China (No. 2017YFA0700800), Natural Science Foundation of China (Nos. 61650202, 61772496 and 61532018).

References

[1] W. Chen, J. Wilson, S. Tyree, K. Weinberger, and Y. Chen. Compressing neural networks with the hashing trick. In *International Conference on Machine Learning (ICML)*, pages 2285–2294, 2015.

[2] F. Chollet. Xception: Deep learning with depthwise separable convolutions. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1251–1258, 2017.

[3] M. Courbariaux, Y. Bengio, and J.-P. David. Binaryconnect: Training deep neural networks with binary weights during propagations. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3123–3131, 2015.

[4] 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.

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

[6] M. Denil, B. Shakibi, L. Dinh, N. De Freitas, et al. Predicting parameters in deep learning. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2148–2156, 2013.

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

[8] C. Ding and D. Tao. Robust face recognition via multimodal deep face representation. *IEEE Transactions on Multimedia (TMM)*, pages 2049–2058, 2015.

[9] D. A. Gudovskiy and L. Rigazio. Shiftcnn: Generalized low-precision architecture for inference of convolutional neural networks. *arXiv preprint arXiv:1706.02393*, 2017.

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

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

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

[13] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In *European conference on computer vision (ECCV), LNCS 9908, Part IV*, pages 630–645, 2016.

[14] Y. He, X. Zhang, and J. Sun. Channel pruning for accelerating very deep neural networks. In *IEEE International Conference on Computer Vision (ICCV)*, 2017.

[15] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

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

[17] G. Huang, S. Liu, L. Van der Maaten, and K. Q. Weinberger. Condensenet: An efficient densenet using learned group convolutions. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[18] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[19] G. B. Huang, M. Mattar, T. Berg, and E. Learned-Miller. Labeled faces in the wild: A database forstudying face recognition in unconstrained environments. In *Workshop on faces in'Real-Life'Images: detection, alignment, and recognition*, 2008.

[20] 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< 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016.

[21] Y. Ioannou, D. Robertson, R. Cipolla, A. Criminisi, et al. Deep roots: Improving cnn efficiency with hierarchical filter groups. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

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

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

[24] V. Lebedev and V. Lempitsky. Fast convnets using group-wise brain damage. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2554–2564, 2016.

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

[26] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *European Conference on Computer Vision (ECCV), LNCS 8693, Part V*, pages 740–755, 2014.

[27] B. Liu, M. Wang, H. Foroosh, M. Tappen, and M. Pensky. Sparse convolutional neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 806–814, 2015.

[28] W. Liu, Y. Wen, Z. Yu, M. Li, B. Raj, and L. Song. Sphereface: Deep hypersphere embedding for face recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[29] W. Liu, Y. Wen, Z. Yu, and M. Yang. Large-margin softmax loss for convolutional neural networks. In *International Conference on Machine Learning (ICML)*, pages 507–516, 2016.

[30] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European Conference on Computer Vision (ECCV), LNCS 9908, Part IV*, pages 525–542, 2016.

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

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

[33] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[34] K. Sun, M. Li, D. Liu, and J. Wang. Igcv3: Interleaved low-rank group convolutions for efficient deep neural networks. In *British Machine Vision Conference (BMVC)*, 2018.

[35] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2017.

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

[37] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2818–2826, 2016.

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

[39] S. Wu, M. Kan, Z. He, S. Shan, and X. Chen. Funnel-structured cascade for multi-view face detection with alignment-awareness. *Neurocomputing*, 2017.

[40] G. Xie, J. Wang, T. Zhang, J. Lai, R. Hong, and G.-J. Qi. Interleaved structured sparse convolutional neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8847–8856, 2018.

[41] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He. Aggregated residual transformations for deep neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5987–5995, 2017.

[42] D. Yi, Z. Lei, S. Liao, and S. Z. Li. Learning face representation from scratch. *arXiv preprint arXiv:1411.7923*, 2014.

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

[44] J. Zhang, M. Kan, S. Shan, and X. Chen. Occlusion-free face alignment: deep regression networks coupled with de-corrupt autoencoders. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3428–3437, 2016.

[45] T. Zhang, G.-J. Qi, B. Xiao, and J. Wang. Interleaved group convolutions. In *IEEE International Conference on Computer Vision (ICCV)*, 2017.

[46] X. Zhang, X. Zhou, M. Lin, and J. Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6848–6856, 2017.

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

[48] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le. Learning transferable architectures for scalable image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8697–8710, 2018.