A Report on Model Compression

Wang Nan
Team of Computer Vision
Sony China Research Lab

CONTENTS

I Introduction **1**
II Problem Definition **1**
II-A CondenseNet…….. 2
II-B MobileNet…….. 2
II-C Experiment…… 3
III Proposed Solutions **3**
III-A Weights Distributions in MobileNet…….. 3
III-A1 Original MobileNet…… 4
III-A2 MobileNet with Group Convolution……..... 4
III-A3 MobileNet with Learned Group Convolution…… 5
III-B Condensed MobileNet……... 9
III-B1 Experiment with different applying strategies……... 9
III-B2 Summary and Conclusion……. 9
III-C MobileNet with delayed Group Convolution…… 9
III-C1 Experiment on CIFAR-10/100……... 10
IV Conclusion and Recommendations **10**
References **11**

LIST OF FIGURES

1 Illustration of learned group convolutions……... 2
2 Bottleneck residual block…….. 3
3 The sparsity of weights of MobileNetV2 on group level…….. 6
4 The sparsity of weights of MobileNetV2-GC on group level……... 7
5 The sparsity of weights of MobileNetV2-LGC on group level……. 8
6 Comparison of weights sparsity between different settings…….. 9
7 Group delay on CIFAR-10/100 datasets…… .. 10

LIST OF TABLES

1 Experimental results of MobileNet and CondenseNet…… 3
2 MobileNetV2 network structure……. 4
3 Experimental results of applying group convolution on MobileNetV2…….. 4
4 Experimental results of applying learned group convolution on MobileNetV2…….. 5
5 Comparison of experimental results between GC/LGC…….. 5
6 Experimental results of applying LGC on expansion and projection layers……... 6
7 Experimental results with different applying strategies…….. 9

A Report on Model Compression

Abstract Deep neural networks are increasingly used on mobile devices, where computational resources are limited. In this report we did some work on model compression based on CondenseNet and MobileNetV2. It combines Learned Group Convolution which remove connections between layers by screen out some unimportant weights with depth-wise separable convolution. At test time, Learned Group Convolution can be implemented using standard group convolution, allowing for efficient computation in practice. Our experiments show that we can condense MobileNetV2 far more without significantly reduce accuracy.

I. INTRODUCTION

In recent years, deep neural networks have recently received lots of attentions, been applied to different applications and achieved dramatic accuracy improvements in many tasks. However, these works rely on deep networks with millions or even billions of parameters are computationally expensive and memory intensive. During the past few years, advanced techniques for compacting and accelerating CNNs model were developed. These improvements are based on the following three criteria:

i. No significant impact on model prediction accuracy.

ii. Reduce space complexity by compressing the number of parameters and the depth of the model.

iii. Does not significantly improve training time while reduce inference time.

These techniques are roughly categorized into five schemes: parameter pruning and sharing [1, 2, 3, 4, 5, 6, 7, 8, 26], low-rank factorization [9, 10, 11], transferred/compact convolution filters [16, 17], quantization [12, 13] and knowledge distillation [14, 15]. The parameter pruning and sharing based methods explore the redundancy in the model parameters and try to remove the redundant and unimportant ones. Low-rank factorization-based techniques use matrix/tensor decomposition to estimate the informative parameters of deep CNNs. The knowledge distillation methods learn a distilled model and train a more compact neural network to reproduce the output of larger networks. The quantization methods compress the original network by reducing the number of bits required to represent each weight. The transferred/compact convolutional filters-based approaches design special structural convolutional filters or efficient model to reduce the storage and computation complexity.

A range of recent studies has explored efficient convolutional networks that can be trained end-to-end [18, 19, 20, 21, 22, 23, 24, 25]. Three prominent examples of networks that are sufficiently efficient to be deployed on mobile devices are MobileNets [18, 19], CondenseNets [21], ShuffleNets [23, 24], which greatly reduce computational requirements without significantly reducing accuracy. CondenseNets use Learned Group Convolution based on DenseNets [20], while MobileNets and ShuffleNets use depth-wise separable convolutions.

Learned Group Convolution is a kind of weight pruning technique which form a standard group convolution by gradually filter out some unimportant weights during training process. However, how to find an effective evaluation method for the importance of parameters is particularly important in this method. We can also see that this evaluation standard is varied, and it is difficult to judge which method is better.

At present, the method [1, 2, 3, 4, 5, 6, 7, 8, 26, 32] based on weights pruning is the most simple and effective model compression method.

II. PROBLEM DEFINITION

Our works are mainly based on CondenseNet and

MobileNet. We want to efficiently and effectively combine learned group convolution / group convolution with depth-wise separable convolution in order to obtain more compact MobileNet. The following is a concise introduction to CondenseNet and MobilNet, as well as some related experiments.

A. CondenseNet

CondenseNet is an improved version based on DenseNet. The most prominent part is the use of learned group convolution for 1x1 convolution, which reduces the calculation by G (group number) times while does not harm accuracy. We learn group convolution through a multi-stage process, illustrated in Figure 1. During the training process we gradually screen out subsets of less important input features for each group. The importance of the *j-th* incoming feature map for the filter group g is evaluated by the averaged absolute value of weights between them across all outputs within the group.

$$\sum_{i=1}^{O/G} | F_{i,j}^{g} | \tag{1}$$

where O, G denote the number of output channels,

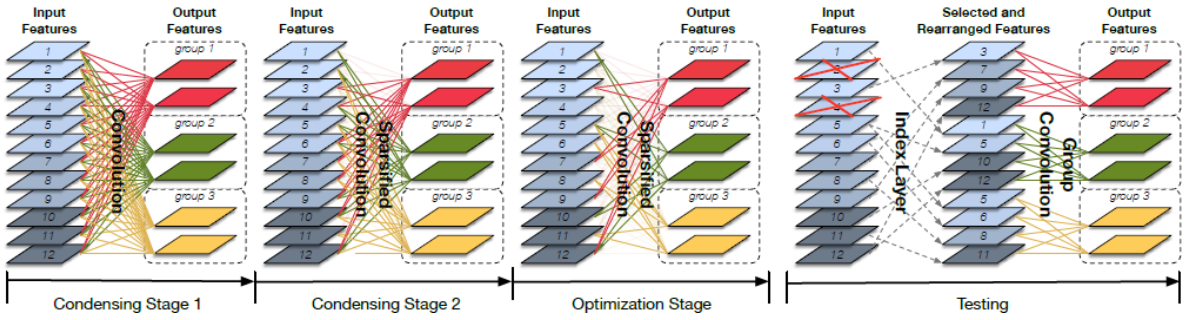


Figure1. Illustration of learned group convolutions with G=3 groups and a condensation factor of C=3. During training a fraction of (C - 1)/C connections are removed after each of the C - 1 condensing stages. Filters from the same group use the same set of features, and during test-time the index layer rearranges the features to allow the resulting model to be implemented as standard group convolutions.

the number of groups.

To reduce the negative effects on accuracy introduced by weight pruning, L_1 regularization is commonly used to induce sparsity [25, 26]. In CondenseNet, we induce group-level sparsity. To this end, we use the following group-lasso regularizer [27, 28] during training:

$$\sum_{g=1}^{G} \sum_{j=1}^{R} \sqrt{\sum_{i=1}^{O/G} F_{i,j}^{g\,2}} \tag{2}$$

where R represents the number of input channels. The group-lasso regularizer induces the group-level sparsity we aim for.

B. MobileNet

MobileNetV2 is based on an inverted residual structure where shortcut connections are between the thin bottleneck layers. The intermediate expansion layer uses lightweight depth-wise separable convolutions which reduce 3x3 convolution computation costs by nearly 9 times, to filter features as a source of non-linearity. The bottleneck structure is shown in Figure 2.

Expansion layer always has more output channels than input channels, exactly how much the data gets expanded is given by the expansion factor. This is one of the hyperparameters for experimenting with different architecture tradeoffs. The default expansion factor is 6. Then a depth-wise convolution that filters the inputs, followed by a 1x1 projection layer which projects data with a high number of dimensions(channels) into a tensor with a much lower number of dimensions.

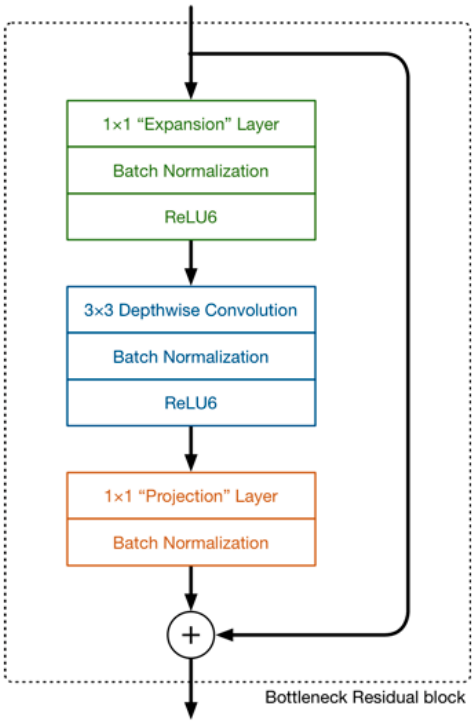


Figure2. There are three convolutional layers in the block. The last two are the ones we already know: a depth-wise convolution that filters the input, followed by a 1x1 pointwise convolution layer. The residual connection works just like in ResNet [29] and exists to help with the flow of gradients through the network. (The residual connection is only used when the number of channels going into the block is the same as the number of channels coming out of it, which is not always the case as every few blocks the output channels are increased.)

C. Experiment

We use TensorFlow, a deep learning framework to reproduce CondenseNet and MobileNet. For training time consideration, all experiments are done on CIFAR-10/100 dataset. Since the official did not perform experiments on CIFAR-10/100 datasets with MobileNetV2, we just used the same network structure proposed in official paper. Results are shown in Table 1.

It can be seen from the experimental results that after applying Learned Group Convolutions, both of DenseNet-94 and MobileNetV2 are compressed by about 2.5 times. But the accuracy rate of DenseNet-94 drops by 2.2%, while that of MobileNetV2 drops by 0.8%, better than DenseNet.

Model	Flops(M)	Params(M)	Acc(%)
DenseNet-94	345	0.947	77.98
CondenseNet-94	137	0.367	75.72
MobileNetV2	314	2.360	77.40
MobileNetV2-LGC	119	1.074	76.60
MobileNetV2 t=4	221	1.758	76.81
MobileNetV2 t=1	79	0.853	74.32

Table 1. Top-1 accuracy rate on CIFAR-100 datasets. DenseNet-94 denotes that CondenseNet-94 without Learned Group Convolutions. MobileNetV2-LGC represents that MobileNetV2 with Learned Group Convolutions on projection layers and Group Convolutions on expansion layers. t denotes expansion factor, which is one of the hyperparameters for experimenting with different architecture tradeoffs.

III. PROPOSED SOLUTIONS

The Problem is that how can we efficiently and effectively combine Learned Group Convolution / Group Convolution operations with MobilenetsV2 while does not significantly reduce accuracy, as well as condense the network.

A. Weights Distributions in MobileNet

According to the network structure of MobileNetV2, Learned Group Convolution or Group Convolution can only be applied to 1x1 Expansion layer and 1x1 projection layer.

If weights in these layers are already dense, which means that all weights are basically important. Either group convolution or learned group convolution will lead to greater loss of precision. But, if weights are sparse, which means only part of weights play a critical role. Then the accuracy loss caused by group convolution or learned group convolution is relatively small.

We determine the importance of input variable based on the absolute value of its corresponding weight, provided that the input variables have been normalized in some way. Otherwise weights corresponding to input variables that tend to have larger values will be proportionally smaller. Which is the most common and popular method currently.

We use Hoyer Index [30] to evaluate the sparsity of weights on group level. The mathematical formula of the Hoyer Index is as follows.

$$(\sqrt{N} - \frac{l^1}{l^2})(\sqrt{N} - 1)^{-1} \quad (3)$$

where, N is the length of the vector, l^1, l^2 are L1-norm and L2-norm corresponding to the vector. The result is between 0-1, with 0 being the least sparse and 1 being the sparsest. First, we calculate the weight on group level according to formula (1), then calculate the Hoyer Index according to formula (3) and visualize it, finally we get figures like Figure3~6.

A1. Original MobileNet

We first perform a set of experiments on CIFAR-100 dataset to validate the sparsity of weights in expansion layer and projection layer. Then we conduct the experiments of applying group convolution on expansion layer and projection layer respectively, as a verification experiment. The neural network structure for CIFAR-100 dataset is show in Table 2. The following experiments are based on this architecture.

Input	Operator	t	c	n	s
$32^2 \times 3$	Conv2d 3×3	-	32	1	1
$32^2 \times 32$	bottleneck	1	16	1	1
$32^2 \times 16$	bottleneck	6	24	2	1
$32^2 \times 24$	bottleneck	6	32	3	1
$32^2 \times 32$	bottleneck	6	64	4	2
$16^2 \times 64$	bottleneck	6	96	3	1
$16^2 \times 96$	bottleneck	6	160	3	2
$8^2 \times 160$	bottleneck	6	320	1	1
$8^2 \times 320$	Conv2d 1×1	-	1280	1	1
$8^2 \times 1280$	Avgpool 8×8	-	1280	1	1
$1^2 \times 1280$	Conv2d 1×1	-	k	-	-

Table 2. MobileNetV2: Each line describes a sequence of 1 or more identical (modulo stride) layers, repeated *n* times. All layers in the same sequence have the same number *c* of output channels. The first layer of each sequence has a stride *s* and all others use stride 1. All spatial convolutions use 3 × 3 kernels. The expansion factor *t* is always applied to the input size referred to "Expansion Layer" shown in Figure 2. The bottleneck is a compound operation referred to Figure 2.

We train all models with stochastic gradient descent (SGD) using similar optimization hyperparameters as in [20, 29]. We adopt Nesterov momentum with a momentum weight of 0.9 without dampening, and use a weight-decay of 10^{-4}. All models are trained with mini-batch size 64 for 400 epochs, unless otherwise specified. We use a cosine shape learning rate [31] which start from 0.1 and gradually reduces to 0.

The sparsity of weights distribution on group level (a special case of group equal to 1) is shown in Figure 3. We can clearly see from the figure that weights distribution on the projection layer is sparser than that on the expansion layer. So, if we conduct group convolution on projection layer, it should get better results than that on expansion layer, illustrated in Table 3.

Model	Flops(M)	Params(M)	Acc(%)
MobilenetV2	314	2.360	77.40
GC-Expansion	218	1.796	75.90
GC-Projection	215	1.638	76.60
GC-All	119	1.074	75.50

Table 3. Top-1 accuracy rate on CIFAR-100 datasets. GC-Expansion denotes that group convolution operations only apply to expansion layer, the same as GC-Projection. GC-All denotes that group convolution apply to both expansion and projection layers. Conducting group convolution operations on the layer whose weights are sparser will get better results.

A2. MobileNet with Group Convolution

We output weights at epoch of 66, 132 and 198, which corresponds to the three condensation stages under the conditions of Learned Group Convolution, and calculate the sparsity of their distribution, as shown in Figure 4.

Weights on the projection layer are much sparser than that on the expansion layer. According to the pruning criteria, on a sparser layer, we can more effectively find out those weights that are not important and filter them out. Conversely, if weights are dense, that is, the weights are of equal importance, so the weights filtered according to the pruning criteria are also important. This is also the

reason why we introduce sparsity in layer which Learned Group Convolution been applied to.

Therefore, using Learned Group Convolution on the projection layer is better than that on the expansion layer. Because weights on the projection layer are sparser, you can effectively filter out some unimportant weights., illustrated in Table 4. and Table 5.

Model	Flops(M)	Params(M)	Acc(%)
MobilenetV2	314	2.360	77.40
LGC-Expansion	218	1.796	76.30
LGC-Projection	215	1.638	77.20

Table 4. Top-1 accuracy rate on CIFAR-100 datasets. LGC-Expansion denotes that group convolution only apply to expansion layer, the same as LGC-Projection. Conducting learned group convolution on the layer whose weights are sparser will get better results with only a small accuracy loss.

A3. MobileNet with Learned Group Convolution

We conduct Learned Group Convolution on expansion layer and projection layer, then analyze how the weights distribution changes in these layers. We find that after some unimportant weights have been filtered, the remaining weights on expansion layer and projection layer are relatively important, and becomes denser, illustrated in Figure 5.

The results in Table 5. show that "Learning" has played a role compared to Group Convolution. Using Learned Group Convolution on the projection layer is better than that on the expansion layer. Some unimportant weights are filtered out, but the accuracy rate does not drop.

Model	Flops(M)	Params(M)	Acc(%)
GC-Expansion	218	1.796	75.90
LGC-Expansion	218	1.796	76.30
GC-Project	215	1.638	76.60
LGC-Projection	215	1.638	77.20
GC-All	119	1.074	75.50
LGC-Expansion/ GC-Projection	119	1.074	75.85
GC-Expansion /LGC-Projection	119	1.074	76.60
LGC-All	119	1.074	76.00

Table 5. Top-1 accuracy rate on CIFAR-100 dataset. GC-All denotes that group convolution apply to both expansion and projection layers, like LGC-All. LGC-Expansion/GC-Projection means using Learned Group Convolution on the expansion layer and Group Convolution on the projection layer. "Learning (a kind of pruning technique)" in projection layer is more effective than in expansion layer

In order to verify whether the group-lasso regularizer influences experimental results, we only use group-lasso regularizer on the layer which apply to Learned Group Convolution, and set the value of lasso decay as 1e-5, unless otherwise specified, other hyperparameters are the same as above. Then, we output the expansion layer and the weights on the projection layer at epoch 198, and compare the heatmap of Hoyer Index with the above two sets of experiments, as shown in Figure 6.

From the results (Figure 6.) we can see that after using Learned Group Convolution, the weights become denser, because some uncritical weights have been screened out, the remaining are equally important. After introducing the group-lasso regularizer with lasso decay of 1e-5, which slightly encourages the sparsity of weights. But when we only use Group convolution, the distribution of weights becomes sparser, which means that a greater loss of information caused by Group convolution.

And we studied the effect of different group-lasso decay values on the experimental results, we set the value of group-lasso decay to 1e-5, 1e-4, 1e-6 and 5e-6 respectively, experimental results are shown in Table 6. The experimental results show that when the value of group-lasso decay is 1e-4, 1e-6, 5e-6, the accuracy is reduced by 1.1%, 0.6%, 0.4% compared to "LGC-All" without group lasso. When the value of group-lasso decay is 1e-5, the accuray rate is slightly improved by 0.3%.

Since normal training produces variance, it is difficult to say that the group-lasso regularizer with decay of 1e-5 helps improve the accuracy. But it slightly promotes the sparsity of weights.

So, if you want to use group-lasso regularizer, you need to analyze it whether it is helpful or not based

on your specific application, and choose the value of group lasso decay carefully.

Model	Flops(M)	Params(M)	Acc(%)
LGC-Expansion	218	1.796	76.30
LGC-Expansion with group lasso (1e-5)	218	1.796	76.50
LGC-Expansion/GC-Projection	119	1.074	75.85
LGC-Expansion/GC-Projection with group lasso(1e-5)	119	1.074	76.20
LGC-All	119	1.074	76.00
LGC-All with group lasso(1e-5)	119	1.074	76.30
LGC-All with group lasso(1e-4)	119	1.074	74.90
LGC-All with group lasso(1e-6)	119	1.074	75.60
LGC-All with group lasso(5e-6)	119	1.074	75.40

Table 6. Top-1 accuracy rate on CIFAR-100 datasets. Using Learned group Convolution on expansion layer with group-lasso regularizer. The performance has been slightly improved after introduce group lasso in Learned Group Convolution.

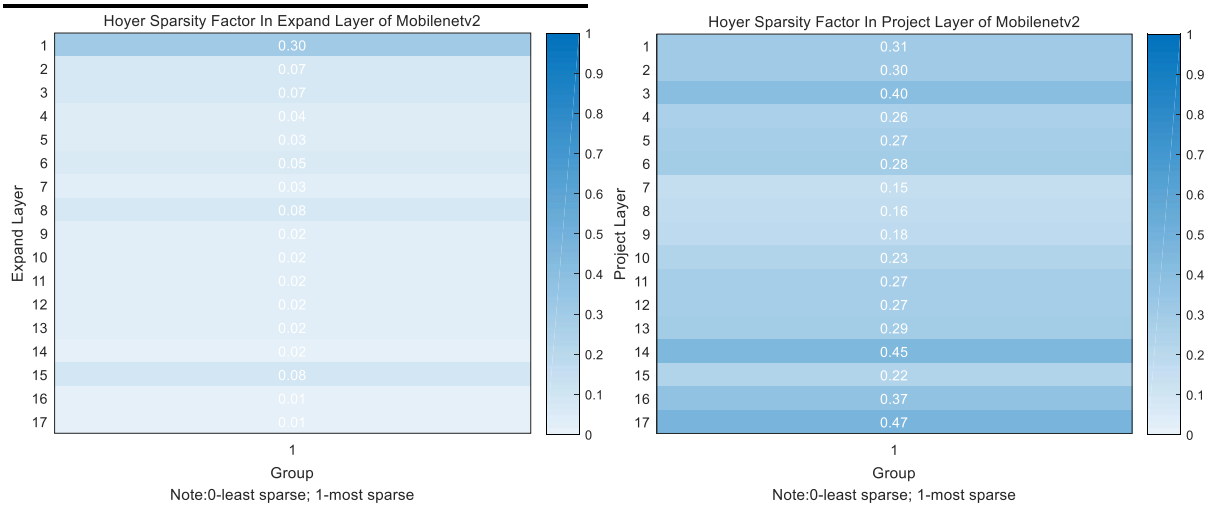
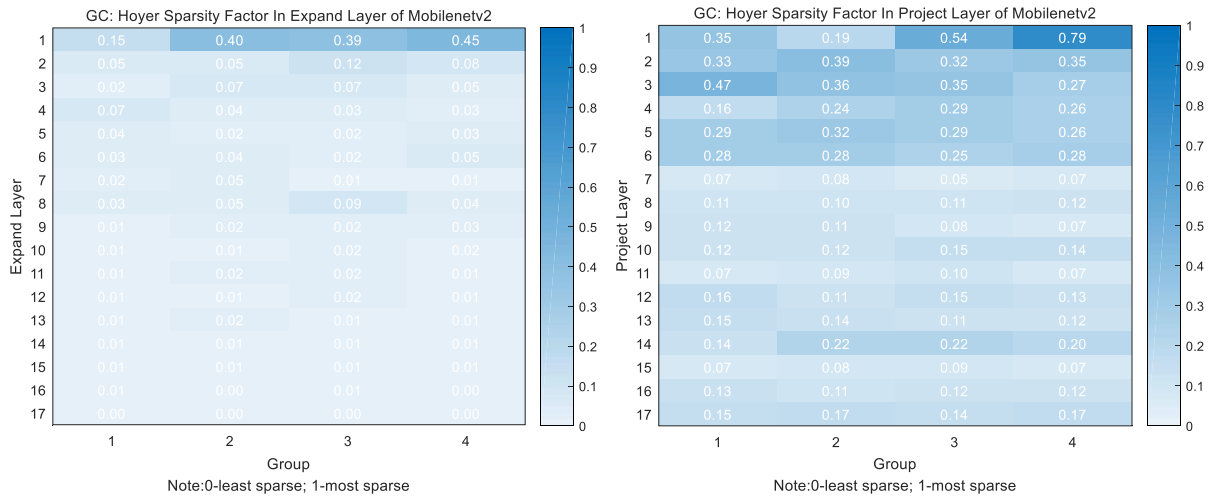
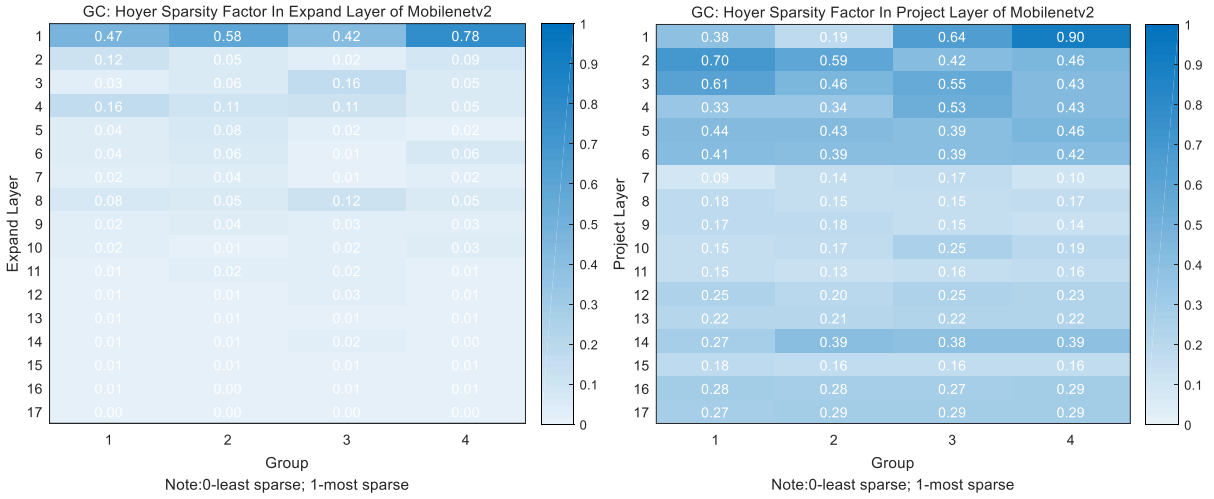


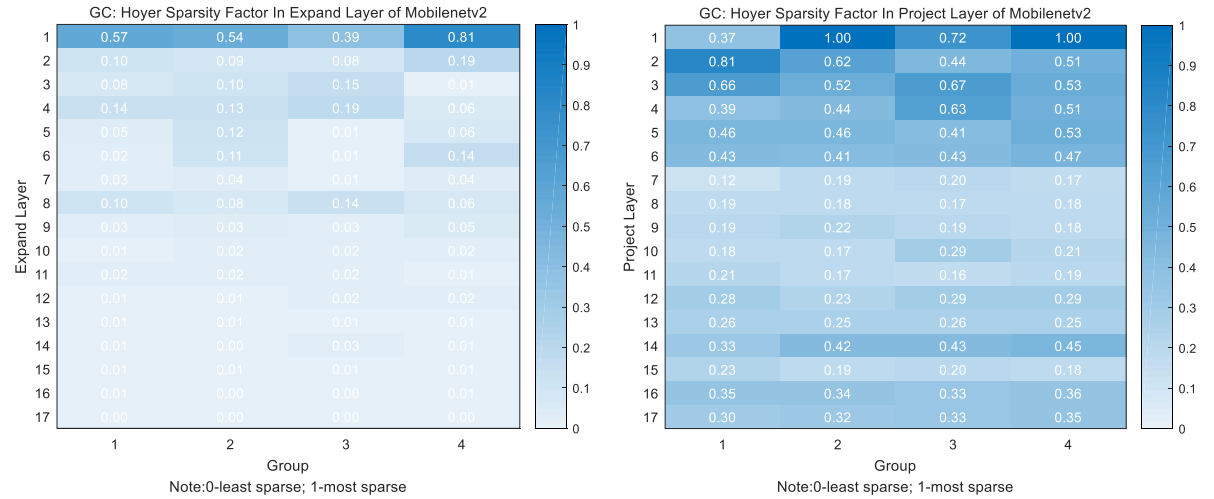
Figure 3. The sparsity of weights at epoch=400. The figure on the left shows the expansion layer while the right shows projection layer. The x-axis represents 1 group (a special case of group equal to 1), and the y-axis represents the layer in each block (the block contains an expansion layer and a projection layer). The color shade represents the sparsity of weight distribution on group level. The darker the color, the Sparser (1-most sparse), the lighter the color, the more compact (0-least sparse).



Epoch=66

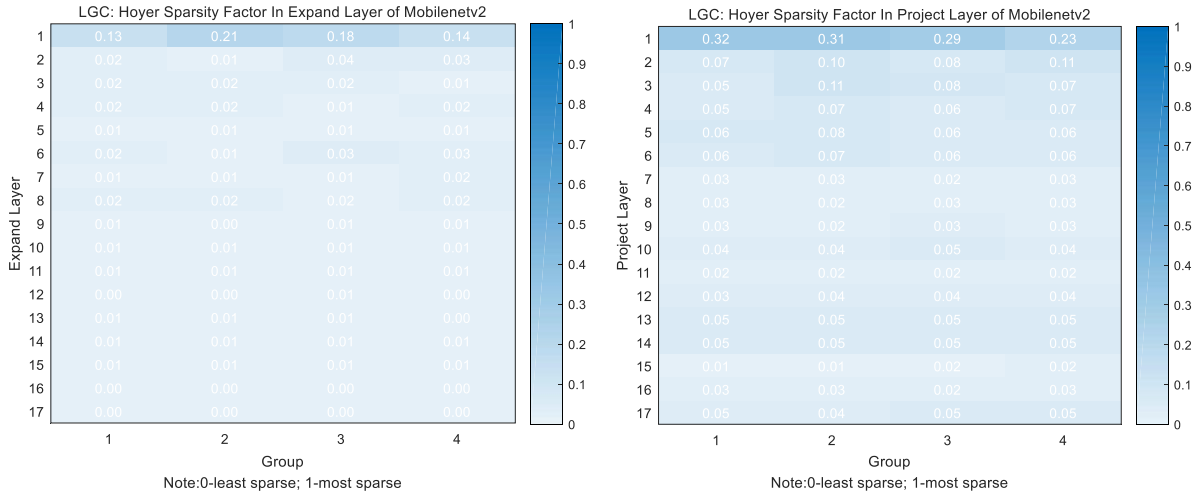


Epoch=132

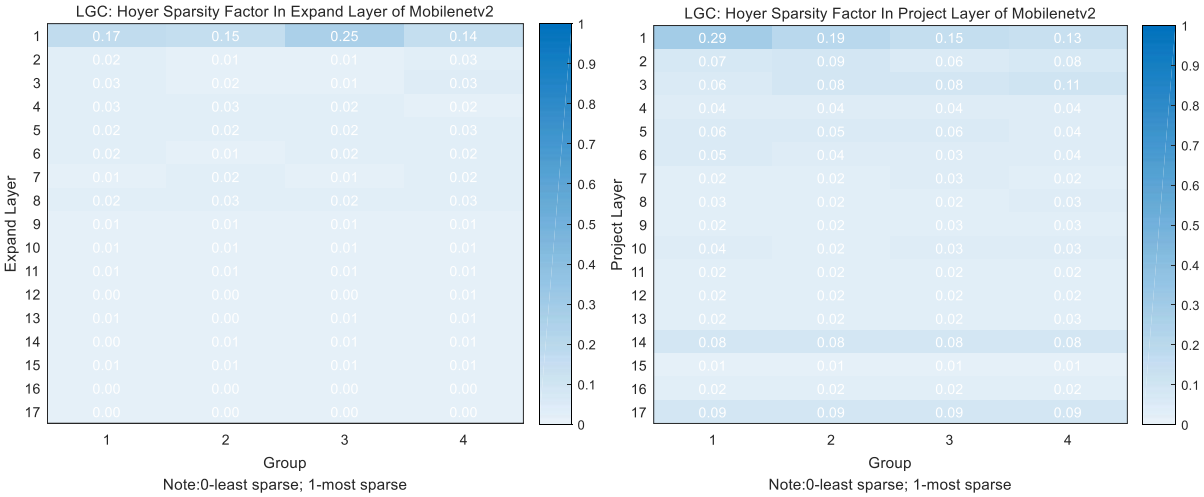


Epoch=198

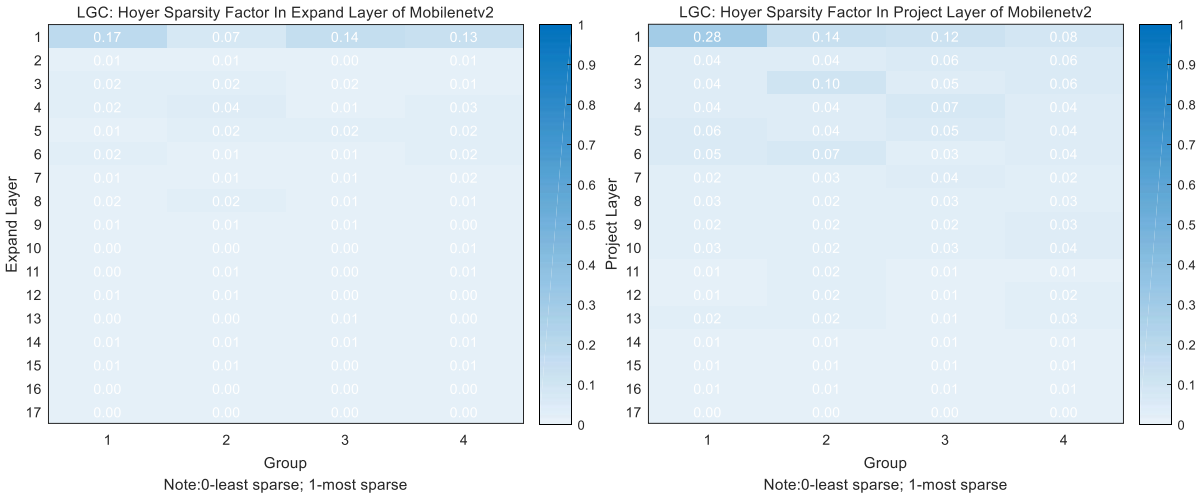
Figure 4. The sparsity of weights on expansion layer and projection layer at epoch of 66, 132, 198 after using GC on these layers. For the expansion layer, weights on the previous layers are sparse. For the projection layer, the overall weights are much sparser than expansion layer.



Epoch=66



Epoch=132



Epoch=198

Figure 5. The sparsity of weights on expansion layer and projection layer at epoch of 66, 132, 198 after using LGC on these layers. Weights on projection layer are slightly sparser than that on expansion layer at group level.

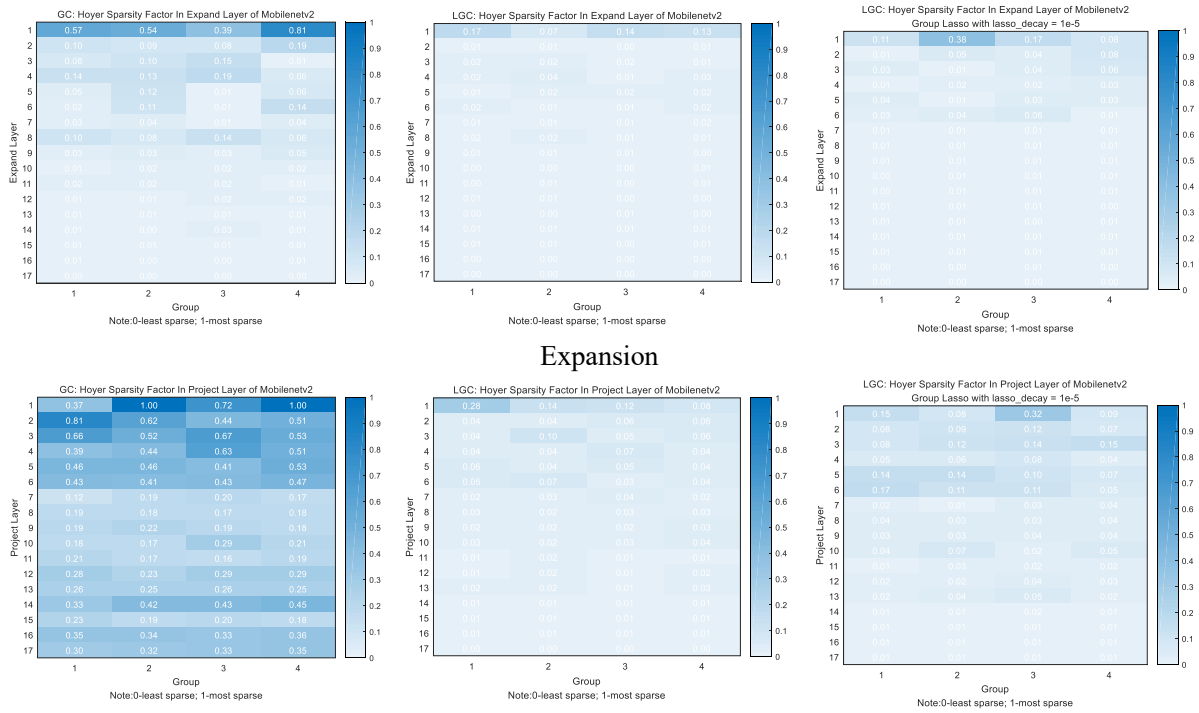


Figure 6. The sparsity of weights on expansion layer and projection layer at epoch of 198 with different settings. From left to right, using group convolution on MobileNetV2, using Learned Group Convolution on MobileNetV2, using Learned Group Convolution with group-lasso regularizer (the value of lasso decay is 1e-5).

B. Condensed MobileNet

MobileNetV2 has already been a compact neural network, but with Learned Group convolution and Group Convolution, you can further compress the model without significantly losing accuracy.

B1. Experiments with different applying strategies

The following experiments in Table 7. show that the performance on CIFAR-100 dataset with different experimental settings.

Using group convolutions on Expansion layer while learned group convolutions on Projection layer condense the original MobileNetV2 (2.5×) with only 0.8% accuracy loss. Using learned group convolutions on Projection layer condense the original MobileNetV2 (1.5 ×) with only 0.2% accuracy loss.

Model	Flops(M)	Params(M)	Acc(%)
MobileNetV2	314	2.360	77.40
LGC-Expansion	218	1.796	76.50
LGC-Projection	**215**	**1.638**	**77.20**
LGC-All	119	1.074	76.30
GC-Expansion	218	1.796	75.90
GC-Projection	215	1.638	76.60
GC-All	119	1.074	75.50
GC-Expansion/LGC-Projection	**119**	**1.074**	**76.60**
LGC-Expansion/ GC-Projection	119	1.074	76.20

Table 7. Top-1 accuracy rate on CIFAR-100 datasets with different applying strategies.

B2. Summary and Conclusion

From the above experimental results, we can infer that there are such conclusions:

i. Learned group convolution operations has a better performance than group convolution.

ii. Using Learned Group Convolution on Projection layer is better than on expansion layer.

iii. If you want to use group-lasso regularizer, you need to analyze it whether it is helpful or not based on your specific application, and choose the value of group lasso decay carefully.

C. MobileNet with delayed Group Convolution

The meaning of delay here is using group convolution operations from one of the layers, the previous layer uses traditional convolution operations. For example, delay-1 means that we use group convolution from the second block, while use traditional convolution on the first block.

C1. Experiment on CIFAR-10/100

We experimented on CIFAR-10/100 dataset using the same neural network with different delayed group convolution operations. The results are plotted in Figure 7.

The first data point of the red and blue lines indicates that group convolutions are applied to all blocks, and the second data point indicates that group convolutions are applied to all blocks except for the first one, and so on. The last data point indicates no group convolution is applied.

With the delay of group convolution, Flops and accuracy rate are both increasing, showing a linear growth trend.

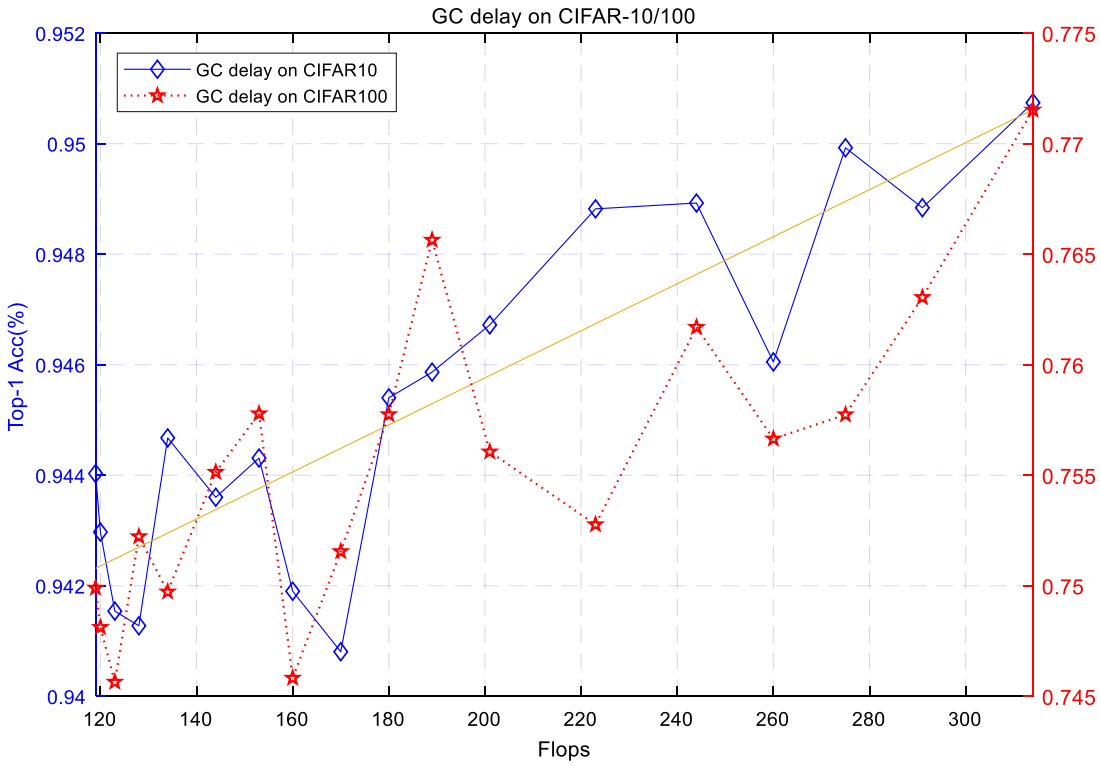


Figure 7. Delayed group convolution experiments result on CIFAR-10/100 datasets. The x-axis represents the Flops under different delayed group convolutions, the y-axis on the left represents the accuracy on cifar10, and the y-axis on the right represents the accuracy on cifar100. The blue thin line with diamonds represents the experimental results on cifar10, and the red dashed line with stars represents the experimental results on cifar100. With the delay of group convolution, Flops and accuracy rate are both increasing, showing a linear growth trend.

IV. CONCLUSION AND RECOMMENDATIONS

MobileNetV2 combined with Learned Group Convolution and Group Convolution operations are studied in this report, which shows the model becomes much more compact without losing too much accuracy by using Learned Group Convolution and Group Convolution on Expansion or Projection layers. Experimental results demonstrates that using LGC/GC can achieve about 1.5× compression of MobileNetV2 on CIFAR-100 with only 0.2% accuracy loss.

Learned Group Convolution is a kind of weights pruning technique which form a standard group

convolution by gradually filter out some unimportant weights during training process. Our experiments results indicate that if the weights are sparse in one layer, we can effectively screen out some uncritical weights.

Model compression is important for deploying deep neural networks on mobile devices. Efficient convolution computation combined with weight pruning may be a good idea.

REFERENCES

[1] LeCun Y, Denker J S, Solla S A. Optimal brain damage[C]//Advances in neural information processing systems. 1990: 598-605.

[2] Srinivas S, Babu R V. Data-free parameter pruning for deep neural networks[J]. arXiv preprint arXiv:1507.06149, 2015.

[3] Han S, Pool J, Tran J, et al. Learning both weights and connections for efficient neural network[C]//Advances in neural information processing systems. 2015: 1135-1143.

[4] Wen W, Wu C, Wang Y, et al. Learning structured sparsity in deep neural networks[C]//Advances in neural information processing systems. 2016: 2074-2082.

[5] Polyak A, Wolf L. Channel-level acceleration of deep face representations[J]. IEEE Access, 2015, 3: 2163-2175.

[6] Li H, Kadav A, Durdanovic I, et al. Pruning filters for efficient convnets[J]. arXiv preprint arXiv:1608.08710, 2016.

[7] Sun Y, Wang X, Tang X. Sparsifying neural network connections for face recognition[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2016: 4856-4864.

[8] He Y, Zhang X, Sun J. Channel pruning for accelerating very deep neural networks[C]//Proceedings of the IEEE International Conference on Computer Vision. 2017: 1389-1397.

[9] Wang M, Liu B, Foroosh H. Factorized convolutional neural networks[C]//Proceedings of the IEEE International Conference on Computer Vision. 2017: 545-553.

[10] Zhang X, Zou J, He K, et al. Accelerating very deep convolutional networks for classification and detection[J]. IEEE transactions on pattern analysis and machine intelligence, 2016, 38(10): 1943-1955.

[11] Jaderberg M, Vedaldi A, Zisserman A. Speeding up convolutional neural networks with low rank expansions[J]. arXiv preprint arXiv:1405.3866, 2014.

[12] Courbariaux M, Hubara I, Soudry D, et al. Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1[J]. arXiv preprint arXiv:1602.02830, 2016.

[13] Zhou A, Yao A, Guo Y, et al. Incremental network quantization: Towards lossless cnns with low-precision weights[J]. arXiv preprint arXiv:1702.03044, 2017.

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

[15] Yim J, Joo D, Bae J, et al. A gift from knowledge distillation: Fast optimization, network minimization and transfer learning[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2017: 4133-4141.

[16] Park J, Li S, Wen W, et al. Faster cnns with direct sparse convolutions and guided pruning[J]. arXiv preprint arXiv:1608.01409, 2016.

[17] Cho M, Brand D. MEC: memory-efficient convolution for deep neural network[C]//Proceedings of the 34th International Conference on Machine Learning-Volume 70. JMLR. org, 2017: 815-824.

[18] Howard A G, Zhu M, Chen B, et al. Mobilenets: Efficient convolutional neural networks for mobile vision applications[J]. arXiv preprint arXiv:1704.04861, 2017.

[19] Sandler M, Howard A, Zhu M, et al. Mobilenetv2: Inverted residuals and linear bottlenecks[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018: 4510-4520.

[20] Iandola F, Moskewicz M, Karayev S, et al. Densenet: Implementing efficient convnet descriptor pyramids[J]. arXiv preprint arXiv:1404.1869, 2014.

[21] Huang G, Liu S, Van der Maaten L, et al. Condensenet: An efficient densenet using learned group convolutions[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018: 2752-2761.

[22] Iandola F N, Han S, Moskewicz M W, et al. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and< 0.5 MB model size[J]. arXiv preprint arXiv:1602.07360, 2016.

[23] Zhang X, Zhou X, Lin M, et al. Shufflenet: An extremely efficient convolutional neural network for mobile devices[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018: 6848-6856.

[24] Ma N, Zhang X, Zheng H T, et al. Shufflenet v2: Practical guidelines for efficient cnn architecture design[C]//Proceedings of the European Conference on Computer Vision (ECCV). 2018: 116-131.

[25] Zhang T, Qi G J, Xiao B, et al. Interleaved group convolutions[C]//Proceedings of the IEEE International Conference on Computer Vision. 2017: 4373-4382.

[26] Liu Z, Li J, Shen Z, et al. Learning efficient convolutional networks through network slimming[C]//Proceedings of the IEEE International Conference on Computer Vision. 2017: 2736-2744.

[27] Yuan M, Lin Y. Model selection and estimation in regression with grouped variables[J]. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 2006, 68(1): 49-67.

[28] Simon N, Friedman J, Hastie T, et al. A sparse-group lasso[J]. Journal of Computational and Graphical Statistics, 2013, 22(2): 231-245.

[29] He K, Zhang X, Ren S, et al. Deep residual learning for image recognition[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 770-778.

[30] Hurley N, Rickard S. Comparing measures of sparsity[J]. IEEE Transactions on Information Theory, 2009, 55(10): 4723-4741.

[31] Loshchilov I, Hutter F. Sgdr: Stochastic gradient descent with warm restarts[J]. arXiv preprint arXiv:1608.03983, 2016.

[32] Cheng Y, Wang D, Zhou P, et al. A survey of model compression and acceleration for deep neural networks[J]. arXiv preprint arXiv:1710.09282, 2017.