Improving Deep Neural Network Sparsity through Decorrelation Regularization

Xiaotian Zhu, Wengang Zhou, Houqiang Li
CAS Key Laboratory of Technology in Geo-spatial Information Processing and Application System,
EEIS Department, University of Science and Technology of China
zxt1993@mail.ustc.edu.cn, zhwg@ustc.edu.cn, lihq@ustc.edu.cn

Abstract

Modern deep learning models usually suffer high complexity in model size and computation when transplanted to resource constrained platforms. To this end, many works are dedicated to compressing deep neural networks. Adding group LASSO regularization is one of the most effective model compression methods since it generates structured sparse networks. We investigate the deep neural networks trained by group LASSO constraint and observe that even with strong sparsity regularization imposed, there still exists substantial filter correlation among the convolution filters, which is undesired for a compact neural network. We propose to suppress such correlation with a new kind of constraint called decorrelation regularization, which explicitly forces the network to learn a set of less correlated filters. The experiments on CIFAR10/100 and ILSVRC2012 datasets show that when combined our decorrelation regularization with group LASSO, the correlation between filters could be effectively weakened, which increases the sparsity of the resulting model and leads to better compressing performance.

1 Introduction

In recent years, deep neural networks (DNNs) have become the dominant tool for most computer vision tasks. Most classic DNN models are designed for pursuing the best performance. However, for many of the potential application scenarios such as mobile phones and embedded systems, the energy consumption and computation resources are extremely limited. The model size and computation complexity thus become a bottleneck when deploying modern DNN models to such platforms [Sun *et al.*, 2016; Zhou *et al.*, 2014; Cai *et al.*, 2016]. This difficulty motivates researchers to develop more compact and efficient DNN models without hurting the state-of-the-art performance.

The deep neural network compression is first explored by [Han *et al.*, 2015]. They find that removing most of the DNN parameters according to their magnitude does not affect the network performance. Since then, various network compression methods are proposed [Li *et al.*, 2017; Lin *et al.*, 2016; Luo *et al.*, 2017; Zhou *et al.*, 2018; Zhu *et al.*, 2018]. Among these methods, training with group LASSO regularization draws much attention [Wen *et al.*, 2016; Liu *et al.*, 2017; Huang and Wang, 2017]. Compared with other compression methods which modify a pretrained large network and finetune the compressed model, group LASSO guides the network to learn the sparse representation during training. Another advantage of group LASSO is its structured sparsity inducing property. Although removing non-structured weights can also result in a sparse network, the computation algorithms involved with non-structured sparse weights are not trivial. Models with structured sparsity can be easily implemented with most off-the-shelf deep learning libraries. However, by investigating the weights of networks trained with group LASSO regularization, we observe that even for a well trained sparse network, there still exist significant correlation among the convolution filters. The existence of correlation indicates that there is still room for further compression.

In this paper, we propose the filter decorrelation regularization for reducing the filter correlation and improving the network sparsity. During the training process, a sparse mask is maintained for each layer, and we minimize the correlation of the filters which have not been masked. This regularization explicitly forces the network to learn a set of less correlated filters, and the sparse mask ensures that decorrelation regularization is only imposed to those indispensable filters. The overall structure of our proposed regularization is shown in Figure 1.

We evaluate our proposed method on several popular DNN architectures. Experimental results show that by using our decorrelation regularization, the filter correlation could be effectively weakened. We also empirically demonstrate that weakening the correlation among convolution filters will result in a more sparse network, which makes our compression results superior than using the sparsity regularization solely and other comparison network compression methods.

2 Related Works

In this section, we briefly review the related works in network compression, which could be summarized into four aspects: network quantization, weight pruning and decomposition, sparsity regularization, and specially designed architectures.

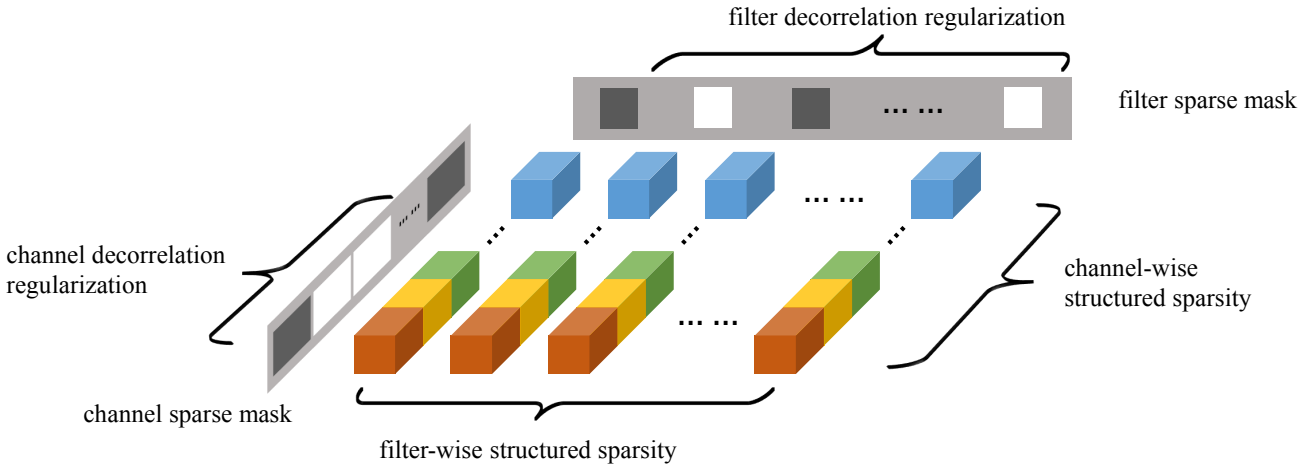

Figure 1: Illustration of the structured sparsity and decorrelation regularization. The structured sparsity regularization aims to remove unimportant weights, and decorrelation regularization forces non-sparse filters and channels less correlated.

2.1 Network Quantization

Observing that tiny perturbation on parameters makes little degradation on the network performance, many works try to compress a DNN by reducing the required number of bits for each parameter. In [Courbariaux *et al.*, 2015], sign function is used for weight quantization and the weights are constrained in $\{+1, -1\}$. Binary weights dramatically reduce the model storage size, but the acceleration effect is not obvious. XNOR-Net makes further acceleration by using 1-bit for both network weights and layer activations. Thus the convolution computation could be replaced with more efficient XNOR operations [Rastegari *et al.*, 2016].

2.2 Weight Pruning and Decomposition

Network pruning is a widely explored network compression method. In [Han *et al.*, 2015], weight magnitude is used as pruning guidance to prune out weights with small absolute value. Due to the difficulty of unstructured sparsity computation, Hu *et al.* define the average percentage of zeros (APoZ) as the filter importance measurement, if a filter always outputs zero, then the whole filter could be safely removed [Hu *et al.*, 2016]. Entropy based pruning uses the entropy of filter's output as filter importance measurement, and prunes the filter which has low output entropy [Luo and Wu, 2017]. Weight decomposition is another structural compression method. Lebedev *et al.* adopt CP decomposition to factorize a 4-D convolution kernel into four small 1-D convolution kernels [Lebedev *et al.*, 2015]. Similarly, in [Kim *et al.*, 2015], Tucker decomposition is used to transform a 4-D convolution kernel into two small 1-D kernels and a 2D kernel.

2.3 Sparsity Regularization

As mentioned before, adding sparsity regularization to network weights could learn a sparse network directly and usually achieves better compression rate than pruning a pre-trained network. In [Wen *et al.*, 2016], group LASSO is applied to the filters, channels, filter shapes and residual blocks as a structured sparsity regularization. The group LASSO regularization [Yuan and Lin, 2006] makes the parameters in some groups all zero, while parameters in other groups are all non-zero. Some works add a gate variable to the filters, channels, or residual blocks, then apply the LASSO regularization to these gate variables [Huang and Wang, 2017; Liu *et al.*, 2017]. After training, the filters, channels, or residual blocks which correspond to zero valued gates can be safely removed. In [Yoon and Hwang, 2017], it not only use group LASSO for network sparsity, but also introduces the exclusive LASSO on each layer's input channels. They claim that exclusive LASSO forces each filter to use different inputs, thus the hidden layers will learn more meaningful features.

2.4 Specially Designed Architectures

Many works try to design a more efficient network architecture directly. In [Howard *et al.*, 2017] the standard convolution layer is replaced by a cheap layer-wise convolution and a point-wise convolution to reduce the convolution computation. Wu *et al.* introduce a zero FLOP shift layer, which shifts feature maps to different directions [Wu *et al.*, 2017]. They claim that using this as an alternative to the spatial convolution can achieve state-of-the-art performance with much less computation.

3 Method

In this section, we investigate the DNN trained by group LASSO regularization, and show that strong filter correlation still exists in a sparse network. Then we describe our proposed decorrelation regularization algorithm in detail.

3.1 Motivation

For a convolutional neural network, the convolution kernel in layer l is a 4-dimensional tensor $\mathbf{W}^l \in R^{D_l \times D_l \times C_l \times N_l}$, where D_l is the spatial size of the convolution kernel in layer l, C_l is the number of input channels, and N_l is the number of filters

in layer l. The filter-wise structured sparsity in [Wen *et al.*, 2016] divides each weight tensor into N groups $\mathbf{W}^{(g)}, g = 1, \cdots, N$, where each group contains the weights of one single filter, and applies group LASSO to these filter groups as a regularization term R_S in the loss function:

$$R_S = \sum_{l=1}^{L} R_g(\mathbf{W}^l), \tag{1}$$

where

$$R_g(\mathbf{W}) = \sum_{g=1}^{N} \sqrt{\sum_{i=1}^{|\mathbf{W}_i^{(g)}|} \left(\mathbf{W}_i^{(g)}\right)^2}. \tag{2}$$

Similarly, the channel-wise structured sparsity applies the group LASSO regularization to C groups of weights $\mathbf{W}^{(g)}, g = 1, \cdots, C$ in each layer, where each group contains the weights that connected to a single input channel. The group LASSO forces some groups of weights all zero, while other groups are all non-zero, thus the network is regularized to have structured sparsity.

Group LASSO regularization effectively removes unimportant weights during training, but it remains uncertain if the resulting model does reach the limit of compression. To answer this question, we investigate the correlation between filters. A filter can be represented by other filters if it's correlated with these filters, thus the filter correlation indicates the redundancy in the network. To measure the filter correlation, we define the *mean filter correlation* $Corr_l$ of the filters in layer l as:

$$Corr_l = \frac{1}{N} \sum_{i=1}^{N} \max\left(|\mathbf{C}_i - \mathbf{I}_i|\right), \tag{3}$$

where $\mathbf{C}$ is the Pearson correlation coefficient matrix, and $\mathbf{C}_i$ is the ith row of matrix $\mathbf{C}$. The mean filter correlation of the whole network is then defined as:

$$Corr = \frac{1}{L} \sum_{l=1}^{L} Corr_l. \tag{4}$$

Intuitively, the mean filter correlation picks the most correlated filter pairs and measures the mean correlation value of these pairs. $Corr$ close to 1 means that most filters in the network are correlated. For a set of fully uncorrelated filters, the corresponding mean filter correlation should be zero.

We train a VGG-16 model on CIFAR10 dataset with the coefficient η of group sparsity regularization picked from a wide range, and measure the mean filter correlation of the trained networks. Table 1 shows the test accuracy on CIFAR10 as well as compressed weight size and mean filter correlation. It could be observed that although the weight size keeps decreasing as the coefficient η increases, the mean filter correlation does not decrease significantly. The mean filter correlation of sparsified networks stays close to the original dense model until the test accuracy begin to drop, which indicates that the group LASSO regularization does not effectively reduce the correlation among filters.

η	Accuracy	Weight size	$Corr$
0	93.58%	58.1 MB	0.471
0.00001	93.85%	55.0 MB	0.455
0.00005	93.68%	50.9 MB	0.461
0.0001	93.62%	45.4 MB	0.463
0.0005	93.39%	19.8 MB	0.458
0.001	93.43%	12.4 MB	0.441
0.005	91.63%	4.1 MB	0.350

Table 1: Compression rate and mean filter correlation of VGG-16 on CIFAR10.

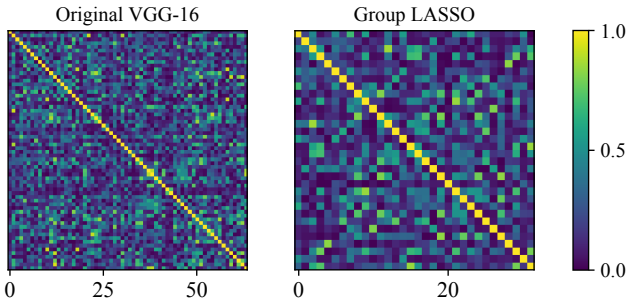

Figure 2: The filter correlation matrix of the first layer in VGG-16. For sparsity regularized VGG-16, the correlation matrix is computed on the non-sparse filters

We compute the Pearson correlation coefficient matrix of first layer filters in original VGG-16 and its sparse regularized counterpart. The absolute value of correlation matrix is shown in Figure 2. It could be seen that although many filters are removed by the group LASSO regularization, there still remains a noticeable correlation within the remaining filters. This phenomenon inspires us that removing filter correlation may be the key to the further network compression.

3.2 Filter Decorrelation

Motivated by removing filter correlation in a network, a straightforward solution is to perform low-rank tensor decomposition as in [Lebedev *et al.*, 2015; Kim *et al.*, 2015; Lin *et al.*, 2016]. Although these decomposition methods can find a compact representation of the network parameters, their acceleration results are not satisfactory. Since each convolution layer in the original network needs to be replaced with several convolution layers, it leads to dramatically increased network depth and temporary data size.

Instead of decomposing pretrained weights, we propose to minimize the filter correlation during training, and learn a set of less correlated filters directly. To achieve this goal, it's natural to extract the non-diagonal elements of filter correlation matrix, and add its L_2-norm to the loss function, which will act as another regularizer and reduce the filter correlation directly:

$$R_C^l = ||\mathbf{C^l} - \mathbf{I}||_2^2. \tag{5}$$

However, due to the structured sparsity induced by group LASSO regularization, the number of valid filters is changing dynamically during the training process. The filters that have values all close to zero will become uncorrelated to other fil-

ters. Thus the loss value will be drastically reduced by these "dying" filters. Another concern is that the dead filters will be removed right after training, reducing the correlation between these unnecessary filters does not benefits the final performance. Therefore we introduce a sparse mask M^l to the filters in layer l. If filter k's mask M^l_k has value 1, then it will be sent to compute the correlation matrix, while the filters with mask value 0 will be skipped in this iteration. We calculate the sparse mask according to the mean absolute values of each filter:

$$M^l_k = \begin{cases} 1 & \text{if } \frac{1}{|\mathbf{W}^l_k|}\sum_{i=1}^{|\mathbf{W}^l_k|} |(\mathbf{W}^l_k)_i| > \tau \\ 0 & \text{otherwise} \end{cases}, \tag{6}$$

where τ is a predefined threshold, and we use mean absolute values to prevent the undesired weighting effect caused by the different number of filters in different layers. With the sparse mask, the improved filter decorrelation regularization term is formulated as:

$$R^l_C = \frac{1}{\sum_k M^l_k}||\hat{\mathbf{C}}^l - \mathbf{I}||^2_2, \tag{7}$$

where $\hat{\mathbf{C}}^l$ is the correlation matrix of the unmasked filters in layer l. We also divide the squared L_2-norm by $\sum_k M^l_k$, which is the number of unmasked filters in layer l. Otherwise, when the number of valid filters changed, the dimension of correlation matrix will change, which will cause the loss value fluctuate drastically.

In structured sparsity regularization, the filter-wise structural sparsity works together with the channel-wise structural sparsity, because removing the input channels in current layer is equivalent to removing the corresponding filters in last layer. Such connection could also be inherited to our proposed method. The channel decorrelation regularization has the same form with filter decorrelation regularization, except that the correlation matrix is computed from input channels, and the sparse mask is determined by the mean absolute values of each input channel. In the experiments, we combine both two types of decorrelation regularizations.

The total loss for network training is represented as:

$$L = L_t + \lambda||\mathbf{W}||^2_2 + \eta \cdot R_S + \gamma \cdot \sum_{l=1}^{L} R^l_C, \tag{8}$$

where L_t is the task related loss (*e.g.*, cross entropy for image classification), the second term is the L_2 regularization on weights (*i.e.*, weight decay), the third term in Eq. 8 is the structured sparsity regularization, and the last term is the proposed filter decorrelation regularization. λ, η and γ are the hyper parameters that control the magnitude of these three regularization terms, respectively.

Apart from the decorrelation regularization for training, the influence of weight initialization should also be taken into consideration. We adopt the SVD decomposition to generate orthogonal random matrices for weight initialization. The weight tensor $\mathbf{W}^l$ of layer l is first generated from the initialization method described in [He *et al.*, 2015], and reshape the $D_l \times D_l \times C_l \times N_l$ tensor $\mathbf{W}^l$ into a $N_l \times (D_l \times D_l \times C_l)$ matrix $\hat{\mathbf{W}}^l$. Therefore each row of this matrix is the weight of one filter. Then the SVD decomposition is applied to the weight matrix $\hat{\mathbf{W}}^l$:

$$\left\{\mathbf{U}_l, \mathbf{S}_l, \mathbf{V}^\top_l\right\} = SVD(\hat{\mathbf{W}}^l). \tag{9}$$

If $N_l < D_l \times D_l \times C_l$, the matrix $\mathbf{V}_l$ will be a $N_l \times (D_l \times D_l \times C_l)$ unitary matrix, whose rows are all orthogonal to each other, and we use matrix $\mathbf{V}_l$ as the initialized weights in layer l. Notice that in some layers there are $N_l > D_l \times D_l \times C_l$, which means we can't find N orthogonal filters, in these layers we do not perform such orthogonal initialization.

4 Experiments

In this section, we report the experimental results of our proposed decorrelation regularization on several network architectures and datasets. The compression results are shown in terms of the weight size as well as FLOPs (floating-point operations). We also show the results of ablation studies in order to demonstrate the effectiveness of each individual component in our method.

4.1 Results on CIFAR10/100

CIFAR is a medium scale image classification dataset introduced in [Krizhevsky and Hinton, 2009]. The dataset has 50000 images for training and 10000 images for testing. The training images are mirror padded by 4 pixels, then random flipped and cropped for data augmentation.

We evaluate our proposed method on the widely used VGG-16 [Simonyan and Zisserman, 2014] and ResNet-56 [He *et al.*, 2016] architecture. For both architectures, the weight decay parameter λ is set to 0.0005, and the structured sparsity parameter η is 0.0015 for VGG-16, and 0.001 for ResNet-56. These are the values of η that result in highest sparsity without significant performance drop when only using structured sparsity regularization on VGG-16 and ResNet-56, respectively. We use the same η when combining structured sparsity with our decorrelation regularization to keep the comparison under the same condition. The decorrelation parameter γ is set to 5 and the sparsity threshold τ is set to 1e-4 according to grid search. We use stochastic gradient descent with momentum 0.9 for training. The initial learning rate is set to 0.1 and decays every 30 epochs with a factor of 0.5.

Evaluation on VGG-16 Table 2 shows the results of our proposed method compared with the naive group LASSO regularizaiton (SSL) [Wen *et al.*, 2016], filter pruning [Li *et al.*, 2017] and network slimming [Liu *et al.*, 2017]. We can observe that with the comparable test accuracy, combing our filter decorrelation regularization with group LASSO always ends up in a more sparsified network than SSL. The model trained by our regularization has 6.8 MB of weights, which is only 11.7% of the baseline model, and the FLOPs is 31.3% of the baseline model. Our method also has a much higher compression rate than the state-of-the-art filter pruning method and the network slimming, which is another sparsity regularization based method. For CIFAR100 experiments, the advantage in compression rate of our method is also noticeable. The compression rate of our method is not as significant as it in the CIFAR10 experiments, we attribute this to

Method	Accuracy	Weight size / Compression rate	FLOPs / Compression rate	$Corr$
CIFAR10				
Baseline	93.58%	58.1 MB / 1.0×	313.7 M / 1.0×	0.471
Filter pruning	93.40%	20.6 MB / 2.8×	206.0 M / 1.5×	-
Network slimming	93.80%	7.3 MB / 8.0×	156.9 M / 2.0×	-
SSL	93.49%	9.2 MB / 6.3×	140.0 M / 2.2×	0.415
Ours	93.31%	6.8 MB / 8.5×	98.4 M / 3.2×	0.136
CIFAR100				
Baseline	73.16%	58.1 MB / 1.0×	313.7 M / 1.0×	0.411
Network slimming	73.48%	14.5 MB / 4.0×	196.1 M / 1.6×	-
SSL	73.18%	22.5 MB / 2.6×	200.2 M / 1.6×	0.328
Ours	73.21%	14.9 MB / 3.9×	150.3 M / 2.1×	0.197

Table 2: Test accuracy and compression rate of VGG-16 on CIFAR10 and CIFAR100. Comparison methods: SSL [Wen *et al.*, 2016], filter pruning [Li *et al.*, 2017] and network slimming [Liu *et al.*, 2017].

the increased task complexity. The VGG-16 model is barely enough for the CIFAR100 dataset, thus there are not much redundancy in the original network.

To give a better illustration on the regularization effect, Figure 3 shows the correlation matrices of model regularized with and without decorrelation regularization, respectively. In contrast with SSL which only uses group LASSO, adding our decorrelation regularization causes most of the remaining filters less correlated after training. The number of remaining filters in each layer is shown in Figure 4. Most of the filters in last layers are removed during training, and our method reduce more filters than only using group LASSO in all layers. The rate of sparsity also indicates that the first several layers seem to be more important, where only few filters have been zeroed out.

Evaluation on ResNet-56 The baseline model of ResNet-56 has a 93.39% accuracy on CIFAR10 and 71.59% accuracy on CIFAR100. The results are shown in Table 3. Compared with SSL [Wen *et al.*, 2016], filter pruning [Li *et al.*, 2017] and channel pruning [He *et al.*, 2017], although the original ResNet model already has less redundancy than VGG-16, we still achieve a compression rate of 2.1× in weights and 1.9× in FLOPs. Again, our method outperforms the SSL and other comparison methods in compression rate of both weight size and computation.

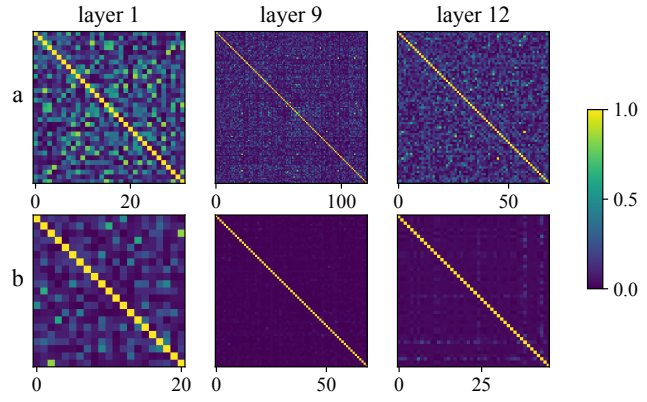

Figure 3: Correlation matrices on filters of difference layers in VGG-16. a) VGG-16 trained without decorrelation regularization. b) VGG-16 trained with decorrelation regularization.

4.2 Results on ILSVRC2012

ILSVRC2012 is a large scale image classification dataset with about 1.2 million images for training and 50000 images for validation [Deng *et al.*, 2009]. In our experiments, the training images are randomly resized and cropped into 224×224, then transformed by random horizontal flipping and mean subtracting. Both training and validation images are subtracted by mean value and divided by standard deviation per channel.

We adopt the AlexNet [Krizhevsky *et al.*, 2012] for the ILSVRC2012 experiments. Results of different ηs are listed to show the trade-off between compression rate and performance, γ and τ is also set to 5 and 1e-4, respectively. The network is trained by stochastic gradient descent with a momentum of 0.9. The initial learning rate is set to 0.01 and reduced by 0.1 at 60, 80 and 90 epochs.

Table 4 lists the experimental results on ILSVRC2012. On such a large scale dataset our method also outperforms SSL both in weights and FLOPs compression rate with an acceptable performance loss. It's worth noting that following by the settings in [Wen *et al.*, 2016], we do not compress the fully connected layers. Since over 97% of the computations in AlexNet come from the convolution layers, and over 94%

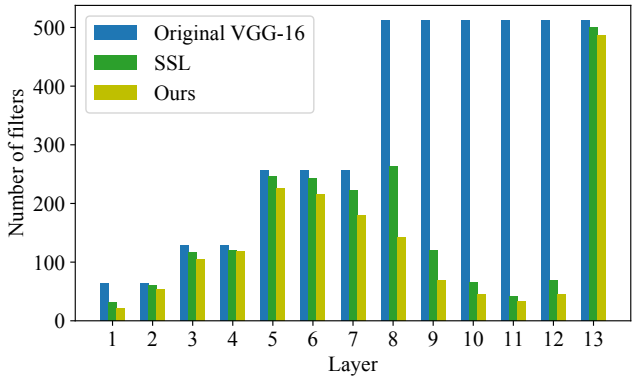

Figure 4: Number of filters in VGG-16 model.

Method	Accuracy	Weights	FLOPs	$Corr$
CIFAR10				
Baseline	93.39%	1.0×	1.0×	0.335
Filter pruning	93.06%	1.4×	1.2×	-
Channel pruning	91.80%	2.0×	-	-
SSL	93.43%	1.2×	1.2×	0.351
Ours	93.40%	2.1×	1.9×	0.194
CIFAR100				
Baseline	71.59%	1.0×	1.0×	0.300
SSL	71.19%	1.1×	1.1×	0.327
Ours	71.65%	1.2×	1.1×	0.122

Table 3: Test accuracy and compression rate of ResNet-56 on CIFAR10 and CIFAR100. Comparison methods: SSL [Wen *et al.*, 2016], filter pruning [Li *et al.*, 2017] and channel pruning [He *et al.*, 2017].

of the weights in AlexNet are in the fully connected layers, the compression rate on weights is not as significant as it is on FLOPs. Combining our regularization with other fully connected layer compression methods will leads to better weight compression result.

4.3 Ablation Study

In Section 3 we state that sparse masks and orthogonal weight initialization are introduced for better decorrealtion results. Here we evaluate the impact of these components and the ablation experimental results are listed in Table 5. From the results of experiments without sparse mask, it could be seen that not only the compression rates are inferior to the models trained with sparse mask, but also the mean filter correlations are higher than the comparison models. This indicates that including the dead filters into correlation calculation introduces negative effect on other remaining filters. A lower mean filter correlation is achieved by adding the orthogonal weight initialization, which demonstrates that orthogonal initialization does benefit the following decorrealtion training.

To verify that whether using the decorrelation regularization alone is already enough for inducing network sparsity, we conduct experiments with only the decorrelation regularization on VGG-16. From the results in Table 6 we could observe that although the filter correlation has been suppressed, the resulting network does not exhibit a good compression rate. This observation indicates that only forcing the network filters to be less correlated does not necessarily make the filters sparse, thus the decorrelation regularization should be combined with group sparsity regularization to take effect.

Method	η	Top-1 acc.	Top-5 acc.	Weights	FLOPs
Baseline	-	56.30%	79.07%	1.0×	1.0×
SSL	0.0005	56.50%	79.17%	1.1×	1.4×
	0.001	54.49%	76.97%	1.1×	1.5×
Ours	0.0005	55.78%	78.42%	1.1×	1.9×
	0.001	54.57%	77.20%	1.2×	2.4×

Table 4: Test accuracy and compression rate of AlexNet on ILSVRC2012. Comparison method: SSL [Wen *et al.*, 2016].

Mask	Init	η	Accuracy	Weights	FLOPs	$Corr$
✗	✗	0.0001	93.63%	1.8×	1.3×	0.185
✓	✗		93.33%	2.1×	1.4×	0.163
✓	✓		93.66%	2.7×	1.5×	0.156
✗	✗	0.0005	93.63%	4.2×	1.6×	0.221
✓	✗		93.74%	4.1×	1.9×	0.146
✓	✓		93.41%	5.0×	2.1×	0.141
✗	✗	0.001	93.54%	5.1×	1.8×	0.223
✓	✗		93.78%	5.8×	2.3×	0.147
✓	✓		93.22%	6.7×	2.5×	0.147
✗	✗	0.005	91.28%	14.9×	3.4×	0.251
✓	✗		92.61%	14.9×	6.2×	0.193
✓	✓		91.77%	16.6×	7.5×	0.176

Table 5: Ablation experiments of VGG-16 on CIFAR-10. The check mark means experiment with corresponding component, and the cross mark means without corresponding component

5 Conclusion

In this paper we focus on the filter correlation in DNNs, which indicates that redundancy exists even in a sparse network model. To reduce the filter correlation and squeeze the redundancy out, we propose the decorrelation regularization, which explicitly minimizes the correlation among the filters. A sparse mask is introduced to remove the side effect of unimportant filters, and orthogonal weight initialization is adopted to give the filters a uncorrelated initial state. Experiments on CIFAR10/100 and ILSVRC2012 datasets demonstrate that our decorrelation regularization effectively weakens the filter correlation, and effectively compresses the corresponding DNN models.

Acknowledgements

This work was supported in part to Dr. Houqiang Li by 973 Program under contract No. 2015CB351803 and NSFC under contract No. 61390514, and in part to Dr. Wengang Zhou by NSFC under contract No. 61472378 and No. 61632019, the Fundamental Research Funds for the Central Universities, and Young Elite Scientists Sponsorship Program By CAST (2016QNRC001).

Sparse	Decorr.	Accuracy	Weights	FLOPs	$Corr$
CIFAR10					
✗	✓	93.40%	1.1×	1.1×	0.189
✓	✓	93.31%	8.5×	3.2×	0.136
CIFAR100					
✗	✓	73.22%	1.1×	1.0×	0.263
✓	✓	73.21%	3.9×	2.1×	0.197

Table 6: The comparison of only use the decorrelation regularization and combine group sparsity with decorrelation regularization on VGG-16.

References

[Cai *et al.*, 2016] Xingyang Cai, Wengang Zhou, Lei Wu, Jiebo Luo, and Houqiang Li. Effective active skeleton representation for low latency human action recognition. *IEEE TMM*, 18(2):141–154, 2016.

[Courbariaux *et al.*, 2015] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. Binaryconnect: Training deep neural networks with binary weights during propagations. In *NIPS*, pages 3123–3131, 2015.

[Deng *et al.*, 2009] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR*, 2009.

[Han *et al.*, 2015] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In *NIPS*, pages 1135–1143, 2015.

[He *et al.*, 2015] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *ICCV*, pages 1026–1034, 2015.

[He *et al.*, 2016] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778. IEEE, 2016.

[He *et al.*, 2017] Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. In *ICCV*, volume 2, page 6, 2017.

[Howard *et al.*, 2017] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[Hu *et al.*, 2016] Hengyuan Hu, Rui Peng, Yu-Wing Tai, and Chi-Keung Tang. Network trimming: A data-driven neuron pruning approach towards efficient deep architectures. *arXiv preprint arXiv:1607.03250*, 2016.

[Huang and Wang, 2017] Zehao Huang and Naiyan Wang. Data-driven sparse structure selection for deep neural networks. In *NIPS*, 2017.

[Kim *et al.*, 2015] Yong-Deok Kim, Eunhyeok Park, Sungjoo Yoo, Taelim Choi, Lu Yang, and Dongjun Shin. Compression of deep convolutional neural networks for fast and low power mobile applications. *arXiv preprint arXiv:1511.06530*, 2015.

[Krizhevsky and Hinton, 2009] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Master's thesis, Department of Computer Science, University of Toronto, 2009.

[Krizhevsky *et al.*, 2012] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, pages 1097–1105, 2012.

[Lebedev *et al.*, 2015] Vadim Lebedev, Yaroslav Ganin, Maksim Rakhuba, Ivan Oseledets, and Victor Lempitsky. Speeding-up convolutional neural networks using fine-tuned cp-decomposition. In *ICLR*, 2015.

[Li *et al.*, 2017] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. In *ICLR*, 2017.

[Lin *et al.*, 2016] Shaohui Lin, Rongrong Ji, Xiaowei Guo, Xuelong Li, et al. Towards convolutional neural networks compression via global error reconstruction. In *IJCAI*, pages 1753–1759, 2016.

[Liu *et al.*, 2017] Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. Learning efficient convolutional networks through network slimming. In *ICCV*, pages 2755–2763. IEEE, 2017.

[Luo and Wu, 2017] Jian-Hao Luo and Jianxin Wu. An entropy-based pruning method for cnn compression. *arXiv preprint arXiv:1706.05791*, 2017.

[Luo *et al.*, 2017] Jian-Hao Luo, Jianxin Wu, and Weiyao Lin. Thinet: A filter level pruning method for deep neural network compression. In *ICCV*, 2017.

[Rastegari *et al.*, 2016] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *ECCV*, pages 525–542. Springer, 2016.

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

[Sun *et al.*, 2016] Shaoyan Sun, Wengang Zhou, Qi Tian, and Houqiang Li. Scalable object retrieval with compact image representation from generic object regions. *ACM TOMM*, 12(2):29, 2016.

[Wen *et al.*, 2016] Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Learning structured sparsity in deep neural networks. In *NIPS*, pages 2074–2082, 2016.

[Wu *et al.*, 2017] Bichen Wu, Alvin Wan, Xiangyu Yue, Peter Jin, Sicheng Zhao, Noah Golmant, Amir Gholaminejad, Joseph Gonzalez, and Kurt Keutzer. Shift: A zero flop, zero parameter alternative to spatial convolutions. *arXiv preprint arXiv:1711.08141*, 2017.

[Yoon and Hwang, 2017] Jaehong Yoon and Sung Ju Hwang. Combined group and exclusive sparsity for deep neural networks. In *ICML*, pages 3958–3966, 2017.

[Yuan and Lin, 2006] Ming Yuan and Yi Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(1):49–67, 2006.

[Zhou *et al.*, 2014] Wengang Zhou, Ming Yang, Houqiang Li, Xiaoyu Wang, Yuanqing Lin, and Qi Tian. Towards codebook-free: Scalable cascaded hashing for mobile image search. *IEEE TMM*, 16(3):601–611, 2014.

[Zhou *et al.*, 2018] Zhengguang Zhou, Wengang Zhou, and Houqiang Li. Online filter weakening and pruning for efficient convnets. In *ICME*, 2018.

[Zhu *et al.*, 2018] Xiaotian Zhu, Wengang Zhou, and Houqiang Li. Adaptive layerwise quantization for deep neural network compression. In *ICME*, 2018.