Machine Intelligence Research
www.mi-research.net

20(1), February 2023, 109-120
DOI: 10.1007/s11633-022-1353-0

Pruning-aware Sparse Regularization for Network Pruning

Nan-Fei Jiang[1,2*] Xu Zhao[1*] Chao-Yang Zhao[1] Yong-Qi An[1,2]
Ming Tang[1,2] Jin-Qiao Wang[1,2]

[1] School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing 100049, China

[2] Institute of Automation, Chinese Academy of Sciences, Beijing 100049, China

Abstract: Structural neural network pruning aims to remove the redundant channels in the deep convolutional neural networks (CNNs) by pruning the filters of less importance to the final output accuracy. To reduce the degradation of performance after pruning, many methods utilize the loss with sparse regularization to produce structured sparsity. In this paper, we analyze these sparsity-training-based methods and find that the regularization of unpruned channels is unnecessary. Moreover, it restricts the network′s capacity, which leads to under-fitting. To solve this problem, we propose a novel pruning method, named MaskSparsity, with pruning-aware sparse regularization. MaskSparsity imposes the fine-grained sparse regularization on the specific filters selected by a pruning mask, rather than all the filters of the model. Before the fine-grained sparse regularization of MaskSparity, we can use many methods to get the pruning mask, such as running the global sparse regularization. MaskSparsity achieves a 63.03% float point operations (FLOPs) reduction on ResNet-110 by removing 60.34% of the parameters, with no top-1 accuracy loss on CIFAR-10. On ILSVRC-2012, MaskSparsity reduces more than 51.07% FLOPs on ResNet-50, with only a loss of 0.76% in the top-1 accuracy. The code of this paper is released at https://github.com/CASIA-IVA-Lab/MaskSparsity. We have also integrated the code into a self-developed PyTorch pruning toolkit, named EasyPruner, at https://gitee.com/casia_iva_engineer/easypruner.

Keywords: Deep learning, convolutional neural network (CNN), model compression and acceleration, network pruning, regula rization.

Citation: N. F. Jiang, X. Zhao, C. Y. Zhao, Y. Q. An, M. Tang, J. Q. Wang. Pruning-aware sparse regularization for network pruning. *Machine Intelligence Research*, vol.20, no.1, pp.109–120, 2023. http://doi.org/10.1007/s11633-022-1353-0

1 Introduction

Convolutional neural networks (CNNs) have demonstrated great success on a variety of computer vision tasks, such as image classification[1], detection[2], and semantic segmentation[3]. However, the increasing depth and width of CNNs also lead to higher computing resource demands and excessive memory footprint requirements. Typically, the widely used ResNet models[4] have millions of parameters, requiring billions of float point operations (FLOPs), making it a great challenge to deploy most state-of-the-art CNNs on edge devices. Network pruning is an effective way to compress and accelerate CNNs. It is attracting much attention from researchers. It can remove the parameters in the deep CNNs and reduce the required FLOPs and memory footprint while preserving the performance.

A typical scheme of network pruning consists of three stages: 1) training an over-parameterized model normally; 2) pruning the model under a certain criterion; and 3) fine-tuning the pruned model to reduce the degradation caused by pruning. Some of the existing network pruning methods apply a sparsity training stage after Step 1. These methods apply sparse regularization on the filter weights of the convolution layers[5, 6] or scaling factors[7, 8] of the batch normalization layers. After sparsity training, the corresponding filter weights or scaling factors of unimportant channels are considered to be near zero. Then, these channels could be safely pruned without affecting the output values of the corresponding layers too much. We call these methods sparsity-training-based methods.

In sparsity-training-based methods, to obtain an expected sparse rate of the model, they adopt global sparse regularization. However, in these methods, the weights of important channels are regularized in the sparsity training stage, although they are preserved after pruning. It is generally regarded that proper regularization achieves a good result by avoiding over-fitting. However, the model will be under-fitted when the regularization coefficient is too large. Since the weights of important channels are also regularized by sparse regularization, the magnitude of these weights usually decays towards 0. This prevents the coverage to a better local minimum of the network in the sparsity training stage, which affects the final performance of the fine-tuned pruned network. Since the network starts from a bad initialization, it is difficult for the

Research Article
Manuscript received March 30, 2022; accepted June 20, 2022
Recommended by Associate Editor Min-Ling Zhang
These authors contribute equally to this work

network to escape the bad local minimum. Moreover, the sparse rate of the globally trained network is difficult to control. This usually results in inconsistencies between the sparse mask and the pruning mask if we want to prune a model to a predefined FLOPs.

To address the problem mentioned above, we propose a novel sparsity-training-based channel pruning approach, MaskSparsity. Different from the previous sparsity-training-based methods that impose regularization on all channels of each layer, MaskSparsity imposes the regularization only on the specific channels selected by the pruning mask, which indicates the unimportant channels, as shown in Fig. 1. Through this mask, MaskSparsity can realize the strong correlation between pruning and regularization and carry out pruning-aware regularization. In other words, we only impose regularization on the channels to be pruned and prune the channels where regularization is applied. The perfect match between the sparse channels and the pruning channels allows us to minimize the impact of sparse regularization and maximize the accuracy of the pruned networks.

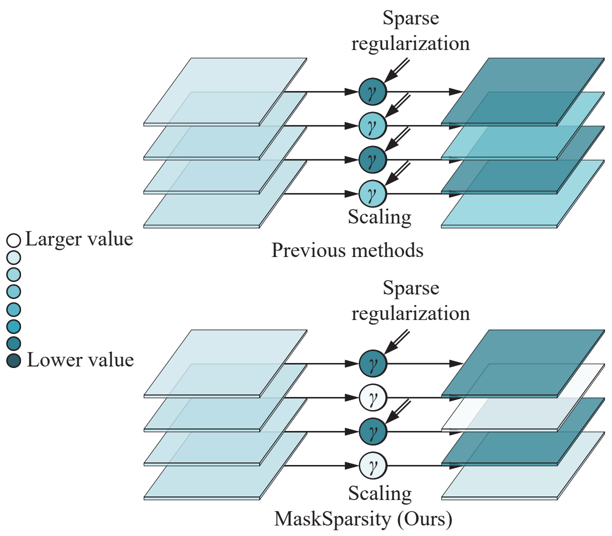


Fig. 1 Visual comparisons between the previous sparsity-training-based methods and the proposed MaskSparsity method. MaskSparsity applies sparse regularization only on the scaling factors of less-important channels.

Compared with the typical pruning methods that directly prunes the unimportant channels, the MaskSparsity can gradually push the unimportant parameters towards zero in a long period of iterations during the sparsity training stage. This prevents the model from a dramatic change in structure or weight in the pruning stage. It is regarded that the dramatic change may result in a certain amount of information loss, which is harmful to restoring the accuracy of the model in the fine-tuning stage.

To summarize, our main contributions are three-fold:

1) We analyze the previous sparsity-training-based methods in the previous work, which simply impose L1 regularization on all channels of the model. We find out the over-regularization problem on important channels.

2) We propose MaskSparsity to solve these problems by more fine-grained sparsity-training.

3) The extensive experiments on two benchmarks show the effectiveness and efficiency of MaskSparsity.

2 Related work

We mainly focus on the structural pruning methods in this paper. In this section, we first review the closely related works, the sparsity-training-based structural pruning methods. After that, we list other structural pruning methods.

2.1 Sparsity-training-based pruning methods

To make the network adaptively converge to a sparse structure and alleviate the damage of the pruning process to the network′s output, some sparsity-training-based pruning methods are proposed. There are mainly two categories of these methods according to the place where sparse regularization is applied.

The first category of methods is the group-sparsity-based methods that apply sparse regularization on the filter weights. Alvarez and Salzmann[5] proposed using a group sparsity regularizer to determine the number of channels of each layer. Wen et al.[6] proposed a structured sparsity learning (SSL) method to regularize the structure to obtain a hardware-friendly pruned structure. Alvarez and Salzmann[9] added a low-rank regularizer to improve the pruning performance. Li et al.[10] proposed the Hinge by combining filter pruning and low-rank decomposition into the group sparsity training framework.

The second category of methods is the indirect group-sparse methods, which apply sparse regularization on the scaling factors of each layer. The representative method is the NetSlim[8] method, which sparsely regularizes the scaling factors of batch normalization (BN) layers to get the sparse structure and remove less important channels. Huang and Wang[7] proposed adding a new scaling factor vector to each layer to apply sparse regularization. Srinivas et al.[11] proposed imposing a sparse constraint over each weight with additional gate variables and achieved high compression rates by pruning connections with zero gate values. Ye et al.[12] proposed pruning channels with layer-dependent thresholds according to the different weight distributions of each layer. Zhao et al.[13] developed norm-based importance estimation by taking the dependency between the adjacent layers into consideration.

These methods apply global sparse regularization on the network channels, which over-regularize the important channels and impose shrinkage for large values of weights. Yamada et al.[14] and Louizos et al.[15] also note that sparsity regularization may cause shrinkage in weights. They propose using $L0$ regularization or stochastic gates to avoid the shrinkage. Our MaskSparsity method solves this problem in another way and improves the performance of sparsity-training-based methods to the state-of-the-art level.

2.2 Non-sparsity-training-based pruning methods

Recently, many non-sparsity-training-based pruning methods have also shown good performance. These methods usually evaluate the importance of each channel with a handcraft criterion first. After that, they directly prune the unimportant channels and finetune the network. For instance, Li et al.[16] proposed pruning filters with smaller $L1$ norm values in a network. Based on the theory of geometric median (GM)[17], He et al.[18] proposed filter pruning via geometric median (FPGM) to prune the filters with the most replaceable contribution. Inspired by the discovery that the average rank of multiple feature maps generated by a single filter is always the same, Lin et al.[19] proposed pruning filters by exploring the high rank of feature maps (HRank). In this paper, we compare the performance of the proposed MaskSparsity with these methods and show good pruning performance.

3 Methodology

3.1 Notations

We assume that a convolutional neural network consists of multiple convolutional layers and each convolution layer is followed by a BN[20] layer. For the l-th convolutional layer, we use C_l and N_l to represent the numbers of input channels and output channels, and $k_l \times k_l$ represents the kernel size.

We use $\mathcal{W}^{(l)} = \{\mathcal{W}^{(l)}_{1,:,:,:}, \mathcal{W}^{(l)}_{2,:,:,:}, \cdots, \mathcal{W}^{(l)}_{N_l,:,:,:}\} \in \mathbf{R}^{N_l \times C_l \times k_l \times k_l}$ to represent the filters of the l-th convolutional layer. The input feature maps and the output feature maps to filters are denoted as $\mathcal{I}^{(l)} = \{\boldsymbol{i}^{(l)}_1, \boldsymbol{i}^{(l)}_2, \cdots, \boldsymbol{i}^{(l)}_{C_l}\} \in \mathbf{R}^{B \times C_l \times h_l \times w_l}$ and $\mathcal{O}^{(l)} = \{\boldsymbol{o}^{(l)}_1, \boldsymbol{o}^{(l)}_2, \cdots, \boldsymbol{o}^{(l)}_{N_l}\} \in \mathbf{R}^{B \times N_l \times {h_l}' \times {w_l}'}$. Here, h_l, w_l, ${h_l}'$ and ${w_l}'$ are the heights and widths of the input and output feature maps respectively. B is the batch size of the input images. The i-th channel feature map $\mathbf{o}^{(l)}_i \in \mathbf{R}^{B \times {h_l}' \times {w_l}'}$ is generated by $\mathcal{W}^{(l)}_{i,:,:,:} \in \mathbf{R}^{C_l \times k_l \times k_l}$ and $\boldsymbol{I}^{(l)} \in \mathbf{R}^{B \times C_l \times h_l \times w_l}$.

For the i-th channel of the l-th BN layer with mean $\mu^{(l)}_i$, standard deviation $\sigma^{(l)}_i$, learned scaling factor $\gamma^{(l)}_i$ and bias $\beta^{(l)}_i$, regardless of bias of the convolutional layer, we have

$$\boldsymbol{o}^{(l)}_i = (\mathcal{W}^{(l)}_{i,:,:,:} \otimes \boldsymbol{I}^{(l)} - \mu^{(l)}_i)\frac{\gamma^{(l)}_i}{\sigma^{(l)}_i} + \beta^{(l)}_i. \qquad (1)$$

3.2 Existing sparsity-training-based methods

Existing sparsity-training-based methods utilize sparse regularization loss to produce structured sparsity. Usually, sparse regularization is either applied on the filter weights of convolutions or the channel scaling factors of BNs.

Sparsity on filter weights. When sparse regularization is applied on the filter weights of convolutions, the training objective function of this category of methods is shown in (2):

$$L_{\text{Sparsity}}(\mathcal{I}^0, y, \mathcal{W}) = L(f(\mathcal{I}^0, \mathcal{W}), y) + \lambda \times \sum_{l=1}^{L}\sum_{i=1}^{N_l} \|\mathcal{W}^{(l)}_{i,:,:,:}\|_g \qquad (2)$$

where ($\mathcal{I}^0$, y) denotes the training samples and the labels, $\mathcal{W}$ denotes the trainable weights, the $L(f(\mathcal{I}^0, \mathcal{W}), y)$ is the objective function of normal training, $\|\mathcal{W}^{(l)}_{i,:,:,:}\|_g$ is a sparsity regularization penalty on the filter weights $\mathcal{W}$, here $\|\cdot\|_g$ is the group Lasso, $\|\mathcal{W}^{(l)}_{i,:,:,:}\|_g = \sqrt{\sum^{C_l}\sum^{k_1}\sum^{k_1}\left(\mathcal{W}^{(l)}_{i,j,k_1,k_2}\right)^2}$. λ is the factor of controlling the strength of sparsity.

Sparsity on channel scaling factors. When sparse regularization is applied on the channel scaling factors of the BN layer, the training objective function of this category of methods is shown in (3):

$$L_{\text{Sparsity}}(\mathcal{I}^0, y, \mathcal{W}) = L(f(\mathcal{I}^0, \mathcal{W}), y) + \lambda \times \sum_{l=1}^{L}\sum_{i=1}^{N_l} \|\gamma^{(l)}_i\|_g \qquad (3)$$

where ($\mathcal{I}^0$, y), $L(f(\mathcal{I}^0, \mathcal{W}), y)$, and λ denote the same mean as above. $\|\gamma^{(l)}_i\|_g$ is a sparsity regularization penalty on the scaling factors γ of BN layers. $\|\cdot\|_g$ is usually set as $L1$ regularization, and $L2$ regularization is available either as [21, 22].

In this paper, we choose the sparsity regularization penalty on the channel scaling factors for further investigation.

As with the normal regularization methods, the received gradients of the scaling factors from the normal training loss L and the sparsity regularization $\|\cdot\|_g$ are usually against each other during training. The former aims at improving the model performance on the training set. The latter aims to restricting the range of the parameters and increasing the structure sparsity, which tends to increase the loss on the training set. The sparsity-training-based pruning method thinks that the unimportant convolutional channels are easily pushed to near 0 by $\|\cdot\|_g$, while the value of important channels is kept large by $\|\cdot\|_g$.

3.3 Over-regularization problem of existing sparsity-training-based methods

Fig. 2 shows the statistical results of two sets of scaling factors (absolute value) collected from the normally-trained and sparsely-trained ResNet-50 on ILSVRC-2012. In Fig. 2, the purple histogram is the distribution of the normally-trained network, while the green histogram represents that of the sparse-trained network. Fig. 2 shows that the scaling factors of the normally-trained network form one peak and those of the sparsely-trained network form two peaks. This is consistent with the bimodal-distribution observation of optimal thresholding (OT)[12].

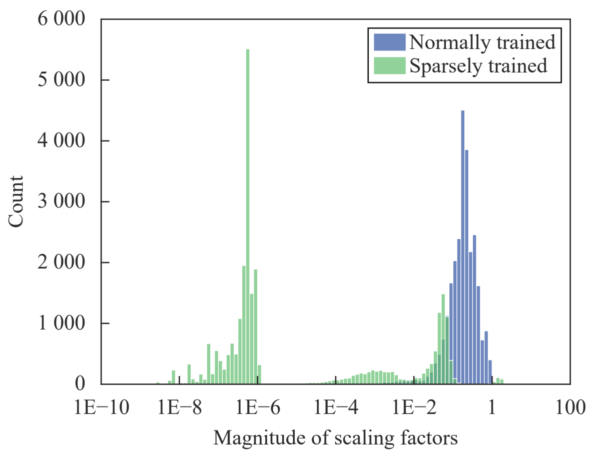


Fig. 2 Distribution of scaling factors of ResNet50 before and after global sparsity training

Obviously, the left scaling-factors peak of the sparsely-trained network represents the unimportant channels and the right peak represents the important channels. In Fig. 2, we demonstrate the over-regularization problem of the existing sparsity-training-based methods, which are mentioned in the introduction.

It can be seen that the right peak of the sparsely-trained network moves to 0 obviously, compared with their location in the histogram of the normally-trained network. This is a common phenomenon of the model regularization methods, i.e., the regularization causes a smaller magnitude of the model parameters. A small regularization usually leads to better generalization performance on the test set. However, excessively large regularization leads to under-fitting. This is because the regularization limits the network′s capacity.

Modern CNNs are usually trained with weight decay, which is widely regarded to be similar to $L2$ regularization, especially under the stochastic gradient descent (SGD) optimizer[23]. This is usually tuned to a proper magnitude to get the best performance. Moreover, sparse regularization is regarded to push a large partition of the channel to be near 0. This requires a large weight for the sparsity loss. Therefore, we think the newly applied sparse regularization on the unpruned channels is over-regularization. It should be avoided.

3.4 Pruning-aware sparse regularization

Therefore, we propose a fine-grained sparsity training method that only applies the sparse regularization on the unimportant channels to maintain the maximum representation ability of the important channels.

The task of sparsity training consists of two sub-tasks implicitly. The first sub-task identifies the unimportant channels. The second sub-task pushes the filter weights or scaling factors of unimportant channels to 0 by the sparse regularization loss. Existing sparsity-training-based methods accomplish the two sub-tasks simultaneously in the sparsity training stage. We propose to decouple the two sub-tasks. By doing this, we can apply the fine-grained sparse regularization, which only sparse out the unimportant channels.

Fig. 3 shows the training pipeline of the proposed MaskSparsity. We transform the sparsity training stage of the existing methods into two stages. The first stage is the sparsity training stage with global sparse regularization, which aims to get the indexes of the unimportant channels. The indexes are transformed into a binary

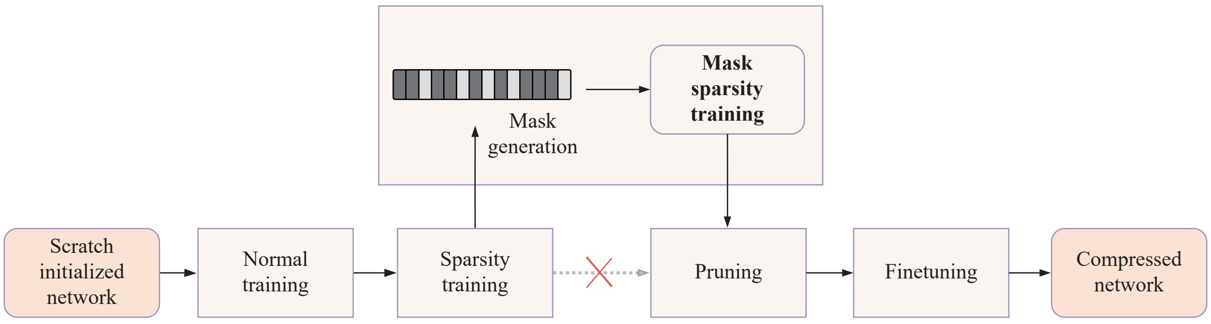


Fig. 3 Pipeline of the proposed MaskSparsity method

pruning mask in previous methods. In our method, we use the mask to identify which channels to apply the sparse regularization in the second stage. To get the pruning mask, we directly threshold the scaling factors of the normally trained network. The details are shown in (4):

$$\mathcal{M} = \{\mathbf{1}(\gamma < \theta) \mid \gamma \in \Gamma\} \tag{4}$$

where $\mathbf{1}$ is the indicator function, Γ is all the scaling factors of the network, and θ is the predefined pruning threshold of the pruning method. The pruning mask $\mathcal{M}$ consists of the unimportant-channel mask of each layer, $\mathcal{M} = \{\mathcal{M}^{(1)}, \mathcal{M}^{(2)}, \cdots, \mathcal{M}^{(L)}\}$, where L is the layer count of the network. According to (4), the pruning mask $\mathcal{M}_i$ is a binary vector consisting of 0 and 1.

Algorithm 1. Algorithm description of MaskSparsity

```
Require: Training data: $\{\mathcal{X}, y\}$, pruning threshold $\theta$.
1) Initialize: Pretrained model parameter $\mathcal{W} = \{\mathcal{W}_i, 0 \le i \le L\}$;
2) for $epoch = 1$; $epoch \le epoch_{\max}$; $epoch++$ do
3)     Update the model parameter $\mathcal{W}$ based on $\{\mathcal{X}, y\}$ and the global sparse regularization as in (3);
4) end for
5) Obtain the pruning mask $\mathcal{M}$ by thresholding $\gamma$ with $\theta$;
6) Reinitialize: Pretrained model parameter $\mathcal{W} = \{\mathcal{W}_i, 0 \le i \le L\}$;
7) for $epoch = 1$; $epoch \le epoch_{\max}$; $epoch++$ do
8)     Update the model parameter $\mathcal{W}$ based on $\{\mathcal{X}, y\}$, the mask-guided sparse regularization as shown in (5) with the mask $\mathcal{M}$;
9) end for
10) Obtain the compact model $\mathcal{W}^*$ from $\mathcal{W}$;
11) Finetune the compact model $\mathcal{W}^*$, the compact model and its parameters $\mathcal{W}^*$.
```

As discussed above, the over-regularization of the important channels limits the network capacity. Therefore, in this paper, we design a fine-grained sparse training strategy to alleviate the damage of the sparse regularization loss on important channels. Specifically, we propose applying the sparse regularization only on the unimportant channels. Based on (3), we can describe our MaskSparsity method as (5):

$$L_{\text{Sparsity}}(\mathcal{I}^0, y, \mathcal{W}) = L(f(\mathcal{I}^0, \mathcal{W}), y) + \lambda \times \sum_{l=1}^{L} \sum_{i=1}^{N_l} \mathcal{M}_i^{(l)} \|\gamma_i^{(l)}\|_g \tag{5}$$

where $\mathcal{M}$ denotes the binary mask, indicating the unimportant channels of the whole network. For the important channels, the values in the channel mask are 0. Therefore, these channels are not affected by the sparse regularization and are trained as normal.

4 Experiments

4.1 Experimental settings

Datasets. To demonstrate the effectiveness of MaskSparsity in reducing model complexity, we evaluate MaskSparsity on both small and large datasets, i.e., CIFAR-10[24] and ILSVRC-2012[1]. The CIFAR-10 dataset consists of natural images of 10 classes with a resolution of 32×32, and the train and test sets contain 50 000 and 10 000 images respectively. The ImageNet dataset consists of natural images of 1 000 classes with a resolution of 224×224, and the train and test sets contain 1.2 million and 50 000 images respectively. We experiment with ResNet-50[25] on ILSVRC-2012, and experiment with ResNet-56[4] and ResNet-110[4] on CIFAR-10.

Codebase and baseline. We directly deploy our algorithm on two popular codebases in GitHub[1, 2] for the experiments on CIFAR-10, CIFAR-100, and ILSVRC-2012. We hardly ever change any of the origin codes, except for adding the codes of our algorithm. Due to the difference in the training strategy of the baseline with other methods, we have a higher baseline accuracy on the evaluation datasets. It should be pointed out that a higher baseline makes it difficult for the pruning algorithms to keep the accuracy after pruning.

Evaluation protocols. We use the number of parameters and the FLOPs[4] to evaluate the complexity of the networks. To evaluate the accuracy, we use Top-1 and Top-5 scores of full-size models and pruned models on ILSVRC-2012 and Top-1 score only on CIFAR-10.

Training and pruning setting. All the training-related hyper-parameters follow the two above-mentioned Github repositories. We use the same hyper-parameters during the normal training stage, the two sparsity training stages, and the finetuning stage in Fig. 3. Specifically, on CIFAR-10, we train models for 200 epochs with a batch size of 128, a weight decay of 0.000 5, a Nesterov momentum of 0.9 without dampening in every stage, and an initial learning rate of 0.1, which is divided by 5 at epochs 60, 120 and 160 on four NVIDIA GTX 1080Ti GPUs; On ILSVRC-2012, we train models for 100 epochs with a batch size of 256, a weight decay of 0.000 1, a Nesterov momentum of 0.9 with dampening in every stage, and an initial learning rate of 0.1, which is divided by 10 at epochs 30, 60 and 90 on eight GPUs.

We set λ as 2E−4 and 5E−4 separately for the global sparsity training stage and the mask sparsity training stage. We only manually set them without too much tuning. We think the former should be set lower since it affects all channels of each layer. Moreover, in the two sparsity training stages, we reinitialize the network with a normally-trained model.

[1] https://github.com/weiaicunzai/pytorch-cifar100
[2] https://github.com/facebookresearch/pycls

For the pruning mask generation step after the global sparse regularization stage, we use a threshold of 1E−2. This thresholding step is to generate a sparse mask for the mask sparse regularization stage. Additionally, we use this sparse mask as our final pruning mask after the mask sparse regularization. In this way, we perform the pruning-aware sparse regularization on the network.

After pruning, we fine-tune the pruned models with an initial learning rate of 0.001, and keep other parameter settings the same as in the previous step, on both the datasets.

4.2 Results and analysis

4.2.1 Results on ILSVRC-2012

As shown in Table 1, our proposed MaskSparsity can achieve the comparable performance with the state-of-the-art methods. NetSlim (NS)[8] is the baseline method of MaskSparsity, which adopts the global sparse regularization on the scaling factors of BN layers and finetunes the pruned model without sparsity regularization. Other that the fine-grained sparsity, the proposed MaskSparsity carries out the same pruning training process with NS. Therefore, NS can be regarded as a baseline method to show the effect of different regularization ranges. OT[12] is the improved NS method, which sets an optimal threshold for each layer.

It can be seen that the MaskSparsity significantly outperforms them in the respect of accuracy drop (Top-1 ↓ and Top-5 ↓ in Table 1) under roughly the same level of FLOPs drop. This shows that with the fine-grained sparse regularization, we avoid the bad effect of the sparse regularization on the unpruned channels. Moreover, as shown in the ablation study at Section 4.3, with MaskSparsity, the pruning threshold on the scaling factors is easier to set.

In Table 1, it can be seen that we also outperform the non-sparsity-training-based methods under the same FLOPs decrease rate, discrimination-aware channel pruning (FPGM)[18] (53.5% FLOPs reduced), DCP[31] (55.76% FLOPs reduced), MetaPruning[29] (51.10% FLOPs reduced), and EagleEye[33] (50% FLOPs reduced). While the FLOPs reduction of MaskSparsity is less than HRank (53.76% VS. 62.1%), the accuracy of the pruned model is much higher than that of HRank (0.93 4.17 in Top-1 accuracy drop).

We note that the experimental data of the comparing approaches directly use the authors′ experimental results of their papers. The reason of different base performances is that they use different base codes and adopt different training strategies like data augmentation.

However, we think the performance drops are still comparable. It is regarded that the higher baseline accuracy of the unpruned network leads to higher difficulty in maintaining the accuracy. Therefore, the smaller accuracy drop of MaskSparsity is convincing.

4.2.2 Results on CIFAR-10

Table 2 shows the experimental results of ResNet-56

Table 1 Evaluation results using ResNet-50 on ILSVRC-2012

Method	Base Top-1 (%)	Base Top-5 (%)	Pruned Top-1 (%)	Pruned Top-5 (%)	Top-1 ↓ (%)	Top-5 ↓ (%)	FLOPs ↓ (%)
NS[8]	75.04	–	69.60	–	5.44	–	50.51
OT[12]	75.04	–	70.40	–	4.64	–	52.88
SFP[26]	76.15	92.87	74.61	92.06	1.54	0.81	41.8
GAL-0.5[27]	76.15	92.87	71.95	90.94	4.20	1.93	43.03
HRank	76.15	92.87	74.98	92.33	1.17	0.54	43.76
Hinge[10]	–	–	74.7	–	–	–	46.55
HP[28]	76.01	92.93	74.87	92.43	1.14	0.50	50
MetaP[29]	76.6	–	75.4	–	1.2	–	51.10
AutoP[30]	76.15	92.87	74.76	92.15	1.39	0.72	51.21
FPGM[18]	76.15	92.87	74.83	92.32	1.32	0.55	53.5
DCP[31]	76.01	92.93	74.95	92.32	1.06	0.61	55.76
ThiNet[32]	75.30	92.20	72.03	90.99	3.27	1.21	55.83
EagleEye*[1][33]	77.21	93.68	76.37	92.89	0.84	0.79	50
MaskSparsity	**76.44**	**93.22**	**75.68**	**92.78**	**0.76**	**0.44**	**51.07**
SCOP[34]	76.15	92.87	75.26	92.53	0.89	0.34	54.6
CHIP[35]	76.15	92.87	76.15	92.91	0.00	–0.04	48.7
CHIP[35]	76.15	92.87	75.26	92.53	0.89	0.34	62.8
HRank	76.15	92.87	71.98	91.01	4.17	1.86	62.10

[1] The baseline of EagleEye* is obtained by evaluating the weight provided by the authors.

Table 2 Evaluation results using ResNet-56 on CIFAR-10

Method	Base Top-1 (%)	Pruned Top-1 (%)	Top-1↓ (%)	FLOPs↓ (%)
NISP[36]	–	–	0.03	42.6
CHIP[35]	93.26	94.16	–0.9	47.4
Hinge[10]	93.69	92.95	0.74	50
AMC[37]	92.8	91.9	0.9	50
LeGR[38]	93.9	93.7	0.2	52
FPGM[18]	93.59	93.26	0.33	52.6
LFPC[39]	93.59	93.24	0.35	52.9
MaskSparsity (ours)	**94.50**	**94.19**	**0.31**	**54.88**
SCOP[34]	95.70	95.64	0.06	56.0
GAL-0.8[27]	93.26	90.36	2.9	60.2
HRank	93.26	90.72	2.54	74.1

on CIFAR-10. On this small dataset, MaskSparsity also achieves the comparable performance with the state-of-the-art methods. Under similar FLOPs reduction with FPGM[18] and Hinge[10], MaskSparsity achieves a 0.31% Top-1 accuracy drop with ResNet-56, which is slightly better than FPGM[18] (0.31% VS. 0.33%) and Hinge[10] (0.31% VS. 0.74%).

Table 3 shows the experimental results of another network, ResNet-110 on CIFAR-10. With this deeper network, MaskSparity achieves a better performance. As shown in Table 3, our MaskSparsity outperforms the other methods, like HRank (at 58.2% FLOPs reduction), under roughly the same ratio of FLOPs reduction. MaskSparsity has roughly the same accuracy increase as FPGM[18] (0.02 VS. 0.06), but MaskSparsity has a larger FLOPs reduction than these two methods (63.03% VS. 52.3% and 60.89%).

Table 4 shows the experimental results of VGG-16, which is a straight network structure that is different from ResNet. We compare MaskSparsity with NetSlim (NS)[8], FPGM[18], and PFEC. It shows that we outperform NS[8] and PFEC on both accuracy and FLOPs reduction. Compared with FPGM[18], we are 0.04% less than FPGM on the increase of the accuracy, which is very minor. However, we decrease FLOPs by 18.01% more than FPGM. Therefore, we also outperform FPGM in general pruning performance. Moreover, we also compare the accuracy drop of the pruned model without the fine-tuning stage. It can be seen in the third column of Table 4 that the MaskSparsity suffers a weaker accuracy drop than the other methods.

4.3 Ablation study

Visualization of the distribution of the scaling factors after using MaskSparsity. In Section 3.3 and Fig. 2, we show that the distribution of scaling factors meets the over-regularization problem that might damage the pruning performance. In Fig. 4, we draw the distribution of the same set of scaling factors after the mask-guided sparse regularization and compare it with that of the pre-trained networks. It can be seen that the two problems are alleviated significantly. The right peak of the sparsity trained network does not move towards 0, as in Fig. 2. Moreover, the two peaks are well distinguish-

Table 3 Evaluation results using ResNet-110 on CIFAR-10

Method	Base Top-1 (%)	Pruned Top-1 (%)	Top-1↓ (%)	FLOPs↓ (%)
Li and Kadav[16]	93.53	93.30	0.23	38.60
SFP[26]	93.68	93.86	–0.18	40.8
NISP-110[36]	–	–	0.18	43.78
GAL-0.5[27]	93.50	92.74	0.76	48.5
CHIP[35]	93.50	94.44	–0.94	52.1
FPGM[18]	93.68	93.74	–0.06	52.3
HRank[19]	93.50	93.36	0.14	58.2
LFPC[39]	93.68	93.07	0.61	60.3
MaskSparsity (ours)	**94.70**	**94.72**	**–0.02**	**63.03**
HRank	93.50	92.65	0.85	68.6
SASL[40]	93.83	93.80	0.03	70.2
CHIP[35]	93.50	93.63	–0.13	71.6

Table 4 Evaluation results using VGG-16 on CIFAR-10
FT: fine-tuning

Method	Base Top-1 (%)	before FT (%)	FT (%)	Top-1↓ (%)	FLOPs↓ (%)
PFEC	93.58	77.45	93.28	0.3	34.2
FPGM[18]	93.58	80.38	94.00	{–0.42}	34.2
NS[8]	93.66	–	93.80	–0.14	51
MaskSparsity (ours)	**93.86**	**94.16**	**94.24**	**–0.38**	**52.21**

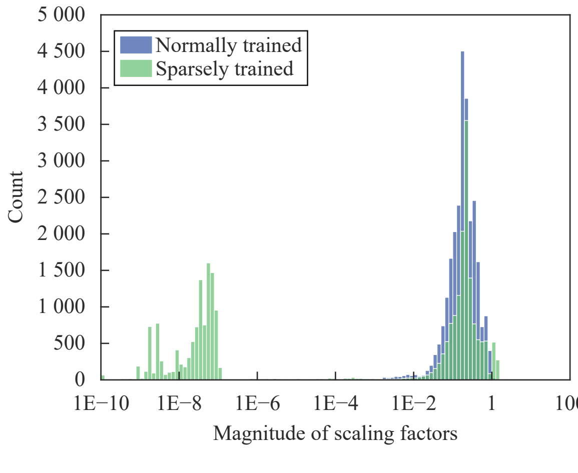


Fig. 4 Distribution of scaling factors of ResNet50 on ILSVRC-2012 before and after the MaskSparsity's sparsity training

able, with almost no in-between bars in the middle area. This demonstrates the effectiveness of the fine-grained sparse regularization of MaskSparsity, which would benefit the pruning performance. This comparison demonstrates the effect of different regularization methods. To rigorously investigate the effect of over-regularization, it is better to analyze the performance of the experimental results.

The convergence analysis by visualizing the gradients. To validate the statement that the sparse regularization on unpruned channels restricts the network's capacity, we visualize the gradients of important and unimportant channels after the sparsity training using global and fine-grained sparse regularization, respectively. The result is shown in Fig. 5. The gradients are collected by continuing training for 1 000 iterations from the end of the sparsity training stage of ResNet-50 on ILSVRC-2012. Fig. 5(a) shows that the gradient norm of the important channels is still large for the important channels with global sparse regularization, while Fig. 5(b) shows the important channel has a small gradient norm with our fine-grained MaskSparsity sparse regularization. Figs. 5(b) and 5(d) show that both the gradients of unimportant channels with the two kinds of sparse regularization methods are almost the same. According to Fig. 5, although the network's accuracy and sparsity have converged, the global sparse regularization leads to a larger gradient on the important channels. We infer that the large gradient prevents the network from converging to a better local minimum.

Comparing fine-tuning and train-from-scratch. In Table 5, we list the network model's accuracy and computational complexity at different pruning stages. Moreover, we also list the result that trains the pruned model from scratch without exploiting its weights. It can be seen that the train-from-scratch result is lower than the fine-tuning result. We think this demonstrates the effectiveness of the fine-grained sparsely-trained pre-trained weights.

The performance of the pruning mask generators. As discussed above, the above pruning process consists of two key elements, i.e. identifying the unimportant channels and pushing them to 0 by sparse regularization. This paper uses global sparsity training to generate the pruning mask. To show the MaskSparsity's generalization on other pruning mask generators, we apply it to two other pruning methods and show the superiority on the performance. Except for the pruning mask generating method, the other stage is the same as the pipeline in Fig. 3.

First, we directly use the codebase of EagleEye[33] and use the pruning mask after its searching process to con-

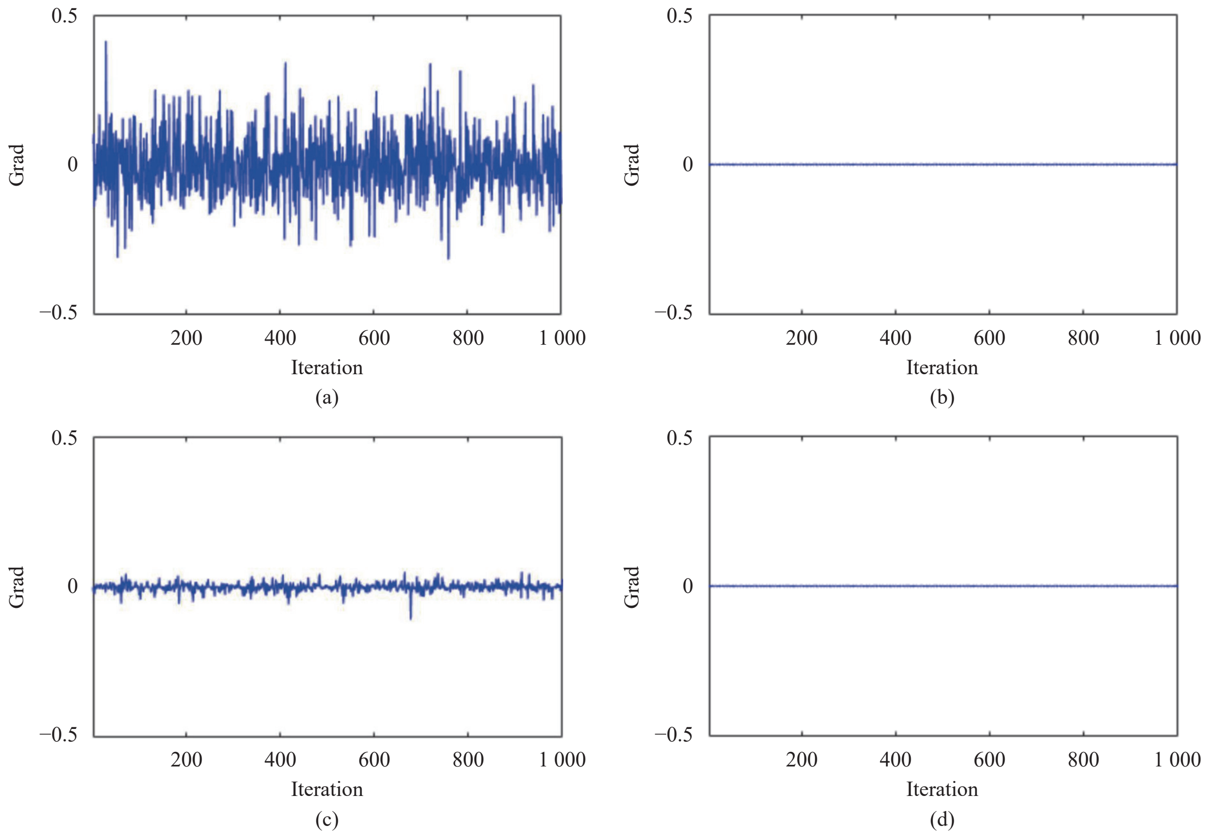


Fig. 5 Gradients of the scaling factors of a certain channel at the end of the sparsity training stage. Both important and unimportant channels are visualized. (a) Important channel, global sparsity; (b) Unimportant channel, global sparsity; (c) Important channel, MaskSparsity; (d) Unimportant channel, MaskSparsity.

duct our mask sparse regularization stage. We reuse the hyper-parameters of EagleEye's finetuning stage in our sparse regularization. Table 6 shows the experimental results. Under the same pruning mask after pruning, Top-1 accuracy increased by 0.32% over the original EagleEye. This shows the scalability of MaskSparsity on state-of-the-art methods.

Table 5 Stage-wise performance in the case of pruning ResNet-56 on CIFAR-10

Model state	Top-1 (%)	FLOPs	Parameters
Normally trained	94.50	126M	853K
MaskSparsity trained	94.02	126M	853K
Pruned	92.67	57M	419K
Finetune	94.19	57M	419K
Train from scratch	93.60	57M	419K

Table 6 Evaluation results of MaskSparsity using the pruning mask of EagleEye

Model state	Top-1 (%)	Top-5 (%)	FLOPs↓ (%)
Unpruned ResNet-50	77.21	93.68	–
EagleEye[33]	76.37	92.89	50.0
MaskSparsity + EagleEye	76.69	93.22	50.0

Second, we try the naive pruning method that directly prunes the same portion of the channels of each layer. We call this naive pruning method as uniform pruning. The experimental results on ResNet-50 are shown in Table 7. It can be seen that MaskSparsity improves this naive pruning method to the SOTA-level performance. This demonstrates the effectiveness of MaskSparsity on pushing the unimportant channels to near 0 without too much damage on the important channels.

Table 7 Evaluation results of MaskSparsity based on the uniform pruning mask of ResNet-50 on ImageNet

Model state	Top-1 (%)	Top-5 (%)	FLOPs↓ (%)
Unpruned ResNet-50	76.44	93.22	–
Direct pruning with uniform mask	74.23	92.28	53.46
MaskSparsity with uniform mask	75.62	92.68	53.46

MaskSparsity with different regularization. In this paper, we mainly use $L1$ regularization to generate the network sparsity. To show the compatibility of MaskSparsity with other specific forms of regularizations, we replace the $L1$ regularization with $L2$ regularization in the mask sparse training stage. We conduct this ablation study using ResNet-56 on CIFAR-10. The experimental results are shown in Table 8. It can be seen that there is little difference in accuracy between the result of $L1$ regularization in the MaskSparsity stage and $L2$ regularization. The accuracy of $L1$ regularization in the MaskSparsity stage is 0.2 points higher than $L2$ regularization.

Table 8 Evaluation results of MaskSparsity with different regularizations of ResNet-56 on CIFAR-10

Model state	Top-1 (%)	Top-5 (%)	FLOPs↓ (%)
Unpruned ResNet-56	94.50	99.79	–
MaskSparsity with L1	94.19	99.81	54.88
MaskSparsity with L2	93.99	99.8	54.88

4.4 Applications on other tasks

To validate the generalization ability, we apply the method to two different object detection tasks. The first is the face detection based on YOLOv5[3] evaluated on WiderFace[41]. The second is the car detection based on Faster-RCNN-FPN[42] evaluated on PASCAL VOC. The results are listed in Tables 9 and 10. For Faster-RCNN-FPN, we only prune the backbone part and report the backbone's FLOPs and parameters. From these experimental results, the pruned models of both tasks maintain roughly the same level of accuracy. It can be concluded that MaskSparsity is applicable to other tasks.

Table 9 Evaluation results of MaskSparsity on an YOLOv5s-based face detector on WiderFace

Model state	mAP[Easy]	FLOPs	Param
YOLOv5s	92.38%	4.1G	3.5M
YOLOv5s+MaskSparsity	91.86%	2.0G	1.6M

Table 10 Evaluation results of MaskSparsity on an FPN-based car detector on PASCAL VOC. The input size is 1 000×600 and the backbone is ResNet-50.

Model state	mAP	FLOPs	Param
Faster-RCNN-FPN	89.7%	49.95G	23.5M
Faster-RCNN-FPN+MaskSparsity	89.3%	23.73G	11.4M

4.5 Discussions of the models without BN

For the models without BN, it is a common practice that we impose group Lasso regularization on the weights of convolutional layers and fully connected layers to enforce the channel-level sparsity of models. Therefore, using our proposed method, we can impose the group Lasso regularization only on the unimportant channels or dimensions selected by the mask to apply MaskSparsity to the models without BN. For the transformers, there are also some sparsity-training based pruning methods like [43]. Based on [43], we can also impose the sparsity regularization only on the unimportant dimensions in the transformer selected by mask and prune unimportant di-

[3] https://github.com/ultralytics/yolov5

mensions of linear projections.

5 Conclusions

In this paper, to solve the problem that existing sparsity-training-based methods over-regularize the important channels, we design a pruning-aware sparse training method, named as MaskSparsity. MaskSparsity only applies the sparse regularization on the unimportant channels that are to be pruned. Therefore, MaskSparsity can minimize the negative impact of the sparse regularization on the important channels. The method is effective and efficient. The experimental results show that it outperforms the other sparsity-training-based pruning methods and achieves the comparable performance with the state-of-the-art methods on the benchmarks. In the future, we plan to work on how to obtain better pruning masks.

Acknowledgements

This work was supported by National Natural Science Foundation of China (Nos. 62176254, 61976210, 61876086, 62076235, 62002356, 62006230 and 62002357), and National Key R&D Program of China (No. 2021ZD0110403).

References

[1] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. A. Ma, Z. H. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, L. Fei-Fei. ImageNet large scale visual recognition challenge. *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015. DOI: 10.1007/s11263-015-0816-y.

[2] T. Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, C. L. Zitnick. Microsoft COCO: Common objects in context. In *Proceedings of the 13th European Conference on Computer Vision*, Springer, Zurich, Switzerland, pp. 740–755, 2014. DOI: 10.1007/978-3-319-10602-1_48.

[3] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, B. Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, Vegas, USA, pp. 3213–3223, 2016. DOI: 10.1109/CVPR.2016.350.

[4] K. M. He, X. Y. Zhang, S. Q. Ren, J. Sun. Deep residual learning for image recognition. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, Vegas, USA, pp. 770–778, 2016. DOI: 10.1109/CVPR.2016.90.

[5] J. M. Alvarez, M. Salzmann. Learning the number of neurons in deep networks. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, ACM, Barcelona, Spain, pp. 2270–2278, 2016. DOI: 10.5555/3157096.3157350.

[6] W. Wen, C. P. Wu, Y. D. Wang, Y. R. Chen, H. Li. Learning structured sparsity in deep neural networks. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, ACM, Barcelona, Spain, pp. 2082–2090, 2016. DOI: 10.5555/3157096.3157329.

[7] Z. H. Huang, N. Y. Wang. Data-driven sparse structure selection for deep neural networks. In *Proceedings of the 15th European Conference on Computer Vision*, Springer, Munich, Germany, pp. 317–334, 2018. DOI: 10.1007/978-3-030-01270-0_19.

[8] Z. Liu, J. G. Li, Z. Q. Shen, G. Huang, S. M. Yan, C. S. Zhang. Learning efficient convolutional networks through network slimming. In *Proceedings of IEEE International Conference on Computer Vision*, Venice, Italy, pp. 2755–2763, 2017. DOI: 10.1109/ICCV.2017.298.

[9] J. M. Alvarez, M. Salzmann. Compression-aware training of deep networks. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ACM, Long Beach, USA, pp. 856–867, 2017. DOI: 10.5555/3294771.329485.

[10] Y. W. Li, S. H. Gu, C. Mayer, L. Van Gool, R. Timofte. Group sparsity: The hinge between filter pruning and decomposition for network compression. In *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition*, IEEE, Seattle, USA, pp. 8015–8024, 2020. DOI: 10.1109/CVPR42600.2020.00804.

[11] S. Srinivas, A. Subramanya, R. V. Babu. Training sparse neural networks. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition Workshops*, Honolulu, USA, pp. 455–462, 2017. DOI: 10.1109/CVPRW.2017.61.

[12] Y. Ye, G. M. You, J. K. Fwu, X. Zhu, Q. Yang, Y. Zhu. Channel pruning via optimal thresholding. In *Proceedings of the 27th International Conference on Neural Information Processing*, Springer, Bangkok, Thailand, pp. 508–516, 2020. DOI: 10.1007/978-3-030-63823-8_58.

[13] K. Zhao, X. Y. Zhang, Q. Han, M. M. Cheng. Dependency aware filter pruning. [Online], Available: https://arxiv.org/abs/2005.02634, 2020.

[14] Y. Yamada, O. Lindenbaum, S. N. Negahban, Y. Kluger. 2020. Feature selection using stochastic gates. In *Proceedings of the 37th International Conference on Machine Learning*, pp. 10648–10659, 2020.

[15] C. Louizos, M. Welling, D. P. Kingma. Learning sparse neural networks through L_0 regularization. [Online], Available: https://arxiv.org/abs/1712.01312, 2018.

[16] H. Li, A. Kadav, I. Durdanovic, H. Samet, H. P. Graf. Pruning filters for efficient ConvNets. In *Proceedings of the 5th International Conference on Learning Representations*, Toulon, France, 2017.

[17] P. T. Fletcher, Suresh Venkatasubramanian, S. Joshi. Robust statistics on Riemannian manifolds via the geometric median. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, Anchorage, USA, 2008. DOI: 10.1109/CVPR.2008.4587747.

[18] Y. He, P. Liu, Z. W. Wang, Z. L. Hu, Y. Yang. Filter pruning via geometric median for deep convolutional neural networks acceleration. In *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition*, IEEE, Long Beach, USA, pp. 4335–4344, 2019. DOI: 10.1109/CVPR.2019.00447.

[19] M. B. Lin, R. R. Ji, Y. Wang, Y. C. Zhang, B. C. Zhang, Y. H. Tian, L. Shao. HRank: Filter pruning using high-

rank feature map. In *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Seattle, USA, pp. 1526–1535, 2020. DOI: 10.1109/CVPR42600.2020.00160.

[20] S. Ioffe, C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning*, ACM, Lille, France, pp. 448–456, 2015. DOI: 10.5555/3045118.3045167.

[21] S. Han, J. Pool, J. Tran, W. J. Dally. Learning both weights and connections for efficient neural network. In *Proceedings of the 28th International Conference on Neural Information Processing Systems*, ACM, Montreal, Canada, pp. 1135–1143, 2015. DOI: 10.5555/2969239.2969366.

[22] E. Tartaglione, S. Lepsøy, A. Fiandrotti, G. Francini. Learning sparse neural networks via sensitivity-driven regularization. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, ACM, Montréal, Canada, pp. 3882–3892, 2018. DOI: 10.5555/3327144.3327303.

[23] P. Goyal, P. Dollár, R. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Q. Jia, K. M. He. Accurate, large minibatch SGD: Training imageNet in 1 hour. [Online], Available: https://arxiv.org/abs/1706.02677, 2017.

[24] A. Krizhevsky. Learning Multiple Layers of Features From Tiny Images, Technical Report TR-2009, University of Toronto, Toronto, Canada, 2009.

[25] K. M. He, X. Y. Zhang, S. R. Ren, J. Sun. Identity mappings in deep residual networks. In *Proceedings of the 14th European Conference on Computer Vision*, Springer, Amsterdam, The Netherlands, pp. 630–645, 2016. DOI: 10.1007/978-3-319-46493-0_38.

[26] Y. He, G. L. Kang, X. Y. Dong, Y. W. Fu, Y. Yang. Soft filter pruning for accelerating deep convolutional neural networks. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, ACM, Stockholm, Sweden, pp. 2234–2240, 2018. DOI: 10.5555/3304889.3304970.

[27] S. H. Lin, R. R. Ji, C. Q. Yan, B. C. Zhang, L. J. Cao, Q. X. Ye, F. Y. Huang, D. S. Doermann. Towards optimal structured CNN pruning via generative adversarial learning. In *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition*, IEEE, Long Beach, USA, pp. 2785–2794, 2019. DOI: 10.1109/CVPR.2019.00290.

[28] X. F. Xu, M. S. Park, C. Brick. Hybrid pruning: Thinner sparse networks for fast inference on edge devices. [Online], Available: https://arxiv.org/abs/1811.00482, 2018.

[29] Z. C. Liu, H. Y. Mu, X. Y. Zhang, Z. C. Guo, X. Yang, K. T. Cheng, J. Sun. MetaPruning: Meta learning for automatic neural network channel pruning. In *Proceedings of IEEE/CVF International Conference on Computer Vision*, IEEE, Seoul, Republic of Korea, pp. 3295–3304, 2019. DOI: 10.1109/ICCV.2019.00339.

[30] J. H. Luo, J. X. Wu. AutoPruner: An end-to-end trainable filter pruning method for efficient deep model inference. *Pattern Recognition*, vol. 107, Article number 107461, 2020. DOI: 10.1016/j.patcog.2020.107461.

[31] Z. W. Zhuang, M. K. Tan, B. H. Zhuang, J. Liu, Y. Guo, Q. Y. Wu, J. Z. Huang, J. H. Zhu. Discrimination-aware channel pruning for deep neural networks. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, ACM, Montréal, Canada, pp. 883–894, 2018. DOI: 10.5555/3326943.3327025.

[32] J. H. Luo, H. Zhang, H. Y. Zhou, C. W. Xie, J. X. Wu, W. Y. Lin. ThiNet: Pruning CNN filters for a thinner net. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 10, pp. 2525–2538, 2019. DOI: 10.1109/TPAMI.2018.2858232.

[33] B. L. Li, B. W. Wu, J. Su, G. R. Wang, L. Lin. EagleEye: Fast sub-net evaluation for efficient neural network pruning. In *Proceedings of the 16th European Conference on Computer Vision*, Springer, Glasgow, UK, pp. 639–654, 2020. DOI: 10.1007/978-3-030-58536-5_38.

[34] Y. H. Tang, Y. H. Wang, Y. X. Xu, D. C. Tao, C. J. Xu, C. Xu, C. Xu. SCOP: Scientific control for reliable neural network pruning. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, 2020.

[35] Y. Sui, M. Yin, Y. Xie, H. Phan, S. A. Zonouz, B. Yuan. CHIP: CHannel independence-based pruning for compact neural networks. In *Proceedings of the Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems*, pp. 24604–24616, 2021.

[36] R. C. Yu, A. Li, C. F. Chen, J. H. Lai, V. I. Morariu, X. T. Han, M. F. Gao, C. Y. Lin, L. S. Davis. NISP: Pruning networks using neuron importance score propagation. In *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition*, IEEE, Salt Lake City, USA. pp. 9194–9203, 2018. DOI: 10.1109/CVPR.2018.00958.

[37] Y. H. He, J. Lin, Z. J. Liu, H. R. Wang, L. J. Li, S. Han. AMC: AutoML for model compression and acceleration on mobile devices. In *Proceedings of the 15th European Conference on Computer Vision*, Springer, Munich, Germany, pp. 815–832, 2018. DOI: 10.1007/978-3-030-01234-2_48.

[38] T. W. Chin, R. Z. Ding, C. Zhang, D. Marculescu. Towards efficient model compression via learned global ranking. In *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition*, IEEE, Seattle, USA, pp. 1515–1525, 2020. DOI: 10.1109/CVPR42600.2020.00159.

[39] Y. He, Y. H. Ding, P. Liu, L. C. Zhu, H. W. Zhang, Y. Yang. Learning filter pruning criteria for deep convolutional neural networks acceleration. In *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition*, IEEE, Seattle, USA, pp. 2006–2015, 2020. DOI: 10.1109/CVPR42600.2020.00208.

[40] J. Shi, J. F. Xu, K. Tasaka, Z. B. Chen. SASL: Saliency-adaptive sparsity learning for neural network acceleration. *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 31, no. 5, pp. 2008–2019, 2021. DOI: 10.1109/TCSVT.2020.3013170.

[41] S. Yang, P. Luo, C. C. Loy, X. O. Tang. WIDER FACE: A face detection benchmark. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, Las Vegas, USA, pp. 5525–5533, 2016. DOI: 10.1109/CVPR.2016.596.

[42] T. Y. Lin, P. Dollár, R. Girshick, K. M. He, B. Hariharan,

S. Belongie. Feature pyramid networks for object detection. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, Honolulu, USA, pp. 936–944, 2017. DOI: 10.1109/CVPR.2017.106.

[43] M. J. Zhu, Y. H. Tang, K. Han. Vision transformer pruning. [Online], Available: https://arxiv.org/abs/2014.08500, 2021.

Nan-Fei Jiang received the B. Eng. degree in telecommunications engineering from Qingdao University, China in 2017. He is currently a master student in computer technology at School of Artificial Intelligence, University of Chinese Academy of Sciences, China.

His research interests include network pruning, model compression, image and video processing.

E-mail: nanfei.jiang@nlpr.ia.ac.cn

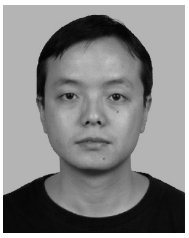
ORCID: 0000-0001-8919-884X

Xu Zhao received the B. Eng. degree in software engineering from Dalian University of Technology, China in 2014, the Ph. D. degree in pattern recognition and intelligence systems from National Laboratory of Pattern Recognition, Chinese Academy of Sciences, China in 2019. He is currently an assistant professor with National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, China.

His research interests include object detection, scene text detection, image and video processing, and intelligent video surveillance.

E-mail: xu.zhao@nlpr.ia.ac.cn (Corresponding author)

ORCID: 0000-0001-5888-8872



Chao-Yang Zhao received the B. Eng. degree in electronic information engineering and M. Sc. degree in circuit and system discipline from University of Electronic Science and Technology of China in 2009 and 2012 respectively, the Ph. D. degree in pattern recognition and intelligence systems from National Laboratory of Pattern Recognition, Chinese Academy of Sciences, China in 2016. He is currently an assistant professor with National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, China.

His research interests include object detection, image and video processing and intelligent video surveillance.

E-mail: chaoyang.zhao@nlpr.ia.ac.cn

Yong-Qi An received the B. Eng. degree in automation from Beijing Institute of Technology, China in 2021. He is currently a master student in pattern recognition and intelligence systems at School of Artificial Intelligence, University of Chinese Academy of Sciences, China.

His research interests include network pruning, knowledge distillation, model compression.

E-mail: yongqi.an@nlpr.ia.ac.cn

Ming Tang received the B. Eng. degree in computer science and engineering, the M. Sc. degree in artificial intelligence from Zhejiang University, China in 1984 and 1987, respectively, and the Ph. D. degree in pattern recognition and intelligent system from Chinese Academy of Sciences, China in 2002. He is currently a professor with National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, China.

His research interests include computer vision and machine learning.

E-mail: tangm@nlpr.ia.ac.cn

Jin-Qiao Wang received the B. Eng. degree from Hebei University of Technology, China in 2001, and the M. Sc. degree from Tianjin University, China in 2004, the Ph.D. degree in pattern recognition and intelligence systems from the National Laboratory of Pattern Recognition, Chinese Academy of Sciences, China in 2008. He is currently a professor with Chinese Academy of Sciences, China.

His research interests include pattern recognition and machine learning, image and video processing, mobile multimedia, and intelligent video surveillance.

E-mail: jqwang@nlpr.ia.ac.cn