BALANCED STRIPE-WISE PRUNING IN THE FILTER

Zheng Huo[1], *Chong Wang*[1,2*], *Weiwei Chen*[1], *Yuqi Li*[1], *Jun Wang*[2] *and Jiafei Wu*[3]

[1] Faculty of Electrical Engineering and Computer Science, Ningbo University, China
[2] School of Information and Control Engineering, China University of Mining and Technology, China
[3] SenseTime Research, China

ABSTRACT

Neural network pruning offers a promising prospect to compress and accelerate modern deep convolution networks. The stripe-wise pruning method with a finer granularity than traditional methods has become the focus of research. Inspired by the previous work, a new balanced stripe-wise pruning, including the balanced pruning strategy and dynamic pruning threshold, is proposed to achieve higher performance. Specifically, the survived inter-filter stripes and the intra-filter stripes are redistributed by a balanced pruning strategy. Meanwhile the dynamic pruning threshold method makes survival rates will be further balanced across all layers. Comprehensive experiments are conducted on two public datasets (CIFAR-10 and TinyImageNet-200) for different models (ResNet and VGG). The experimental results show that the proposed model is capable of reducing the most parameters, yet achieving the highest accuracy. Our code is available at: https://github.com/ajdt1111/BSWP.

Index Terms— neural network pruning, stripe-wise pruning, efficient network architecture

1. INTRODUCTION

Deep neural network (DNN) has become a mainstream method in many fields, such as image recognition [1-3], speech recognition [4,5], object detection [6-8], etc. However, limited by the millions of parameters and heavy computational burdens, it is challenging for the deployment of neural networks on edge devices and mobile devices. To address this issue, extensive studies have been focused on compressing the model, aiming to reduce the computational redundancy with an acceptable accuracy. Among the various network compression methods, the ones of filter (channel) pruning and weights pruning are very active.

Filter (channel) pruning [9-11] nullifies the weak filters or channels within the convolution layers, which leaves the model with regular structures that facilitates acceleration. The $L1$-norm was used to prune unimportant convolutional filters [12]. Instead of pruning filters, the scaling factors of batch normalization layers were used to prune the corresponding channels with small scaling factors [9]. Meanwhile, least square reconstruction and lasso regression was applied to prune channels in [10]. Weights Pruning [13-15] is another group of fine-grained pruning methods, which directly deletes weight values in filters. However, it may cause unstructured sparsity due to the randomness of the deep neural network. This irregular structure makes it difficult to leverage the general hardware to achieve acceleration.

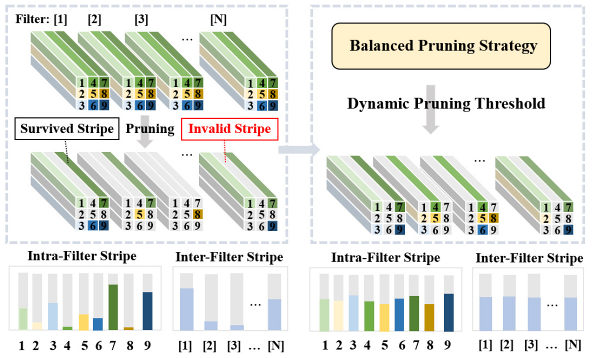

Fig. 1. The illustration of the balanced stripe-wise pruning. The number of survived stripes will be balanced.

Stripe-Wise Pruning (SWP) [16] combined the strength of aforementioned methods, which achieved more fine-grained pruning than traditional filter pruning and was still hardware friendly. By introducing a filter skeleton matrix to learn the importance of stripes in the filter, less significant stripes were pruned instead of the whole filter. However, SWP does not consider the local spatial distribution of the stripes and the correlation between different channels of the output feature maps, which results an imbalanced pruning result. In some cases, it may degrade the generalization ability or representation capacity of the model.

To address this issue, we propose a variant of SWP [16], namely balanced stripe-wise pruning (BSWP), to perform pruning in the filter. The key concept of BSWP shown in Fig. 1 is that stripes-wise pruning is performed under the constraint that the number of different types of stripes (intra-filter ones) are close, while the number of survived stripes in different filters (inter-filter ones) will also be balanced by a new pruning strategy. The main contribution of this work is twofold, (1) the idea of balanced pruning is firstly proposed for SWP; (2) the pruning threshold is dynamically adjusted according to the survival rate at each layer during the training process. With these improvements, the proposed BSWP model is able to achieve higher accuracy with less parameters and lower FLOPS for neural network pruning.

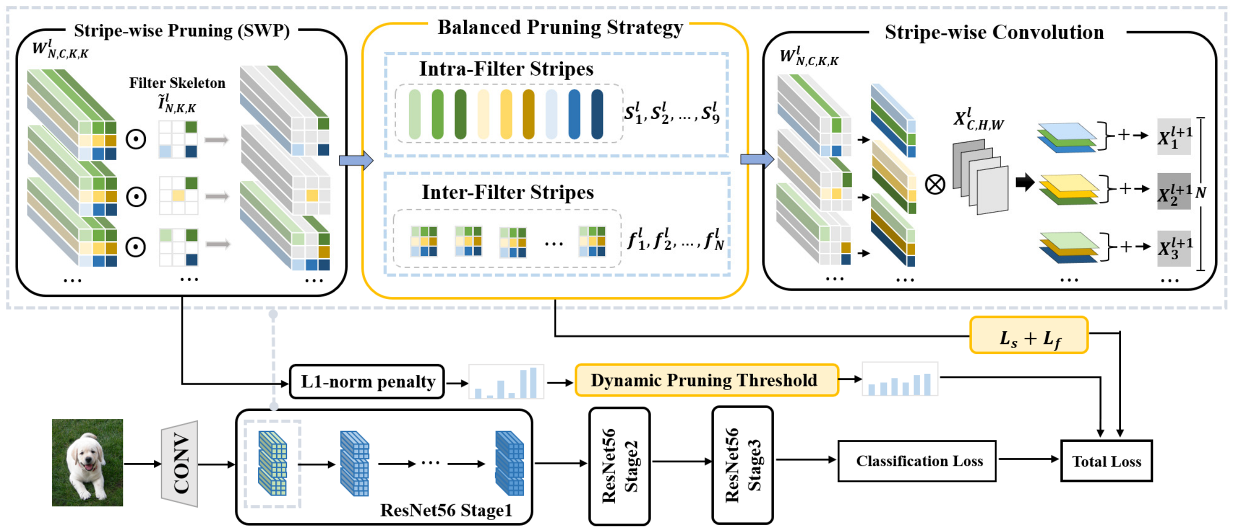

Fig. 2. The framework of the proposed balanced strip-wise pruning strategy in ResNet56. The main contributions of this work are marked in orange blocks, while the original stripe-wise pruning is denoted in black blocks.

2. METHODOLOGY

The main components of BSWP will be discussed in this section. As shown in Fig. 2, a new balanced pruning strategy is proposed to improve the vanilla SWP [16]. Not only the number of survived inter-filter stripes but also the number of intra-filter ones are constrained to be close. Furthermore, the survival rate at each layer is also dynamically adjusted to achieve a layer-wise balance.

2.1. Stripe-Wise Pruning

Generally, the filter, input and output feature map of l-th layer can be denoted as $W^l_{N,C,K,K}$, $X^l_{C,H,W}$ and $X^{l+1}_{N,H,W}$, respectively. N and K are the number and size of the filter, while H, W and C are the height, width and channel dimension. In SWP [16], a learnable matrix $I^l_{N,K,K}$, namely Filter Skeleton (FS), is proposed to represent the importance of each stripes in the filter, which is used to prune the filter weights $W^l_{N,C,K,K}$ as shown in Fig. 2. Then the convolution process with FS can be expressed as,

$$X^{l+1}_{n,h,w} = \sum_{c=1}^{C}\sum_{i=1}^{K}\sum_{j=1}^{K} I^l_{n,i,j} \times W^l_{n,c,i,j} \times X^l_{c,h+i-1,w+j-1} \quad (1)$$

where n and c denote n-th filter and c-th channel, h and w are the indices of height and width, i and j are the spatial offsets.

During the training process, FS $I^l_{N,K,K}$ is initialized as an all-one matrix. And a lasso loss is applied on it to perform sparsity regularization,

$$L_{lasso} = \sum_{l=1}^{L}\Big(\sum_{n=1}^{N}\sum_{i=1}^{K}\sum_{j=1}^{K}\big|I^l_{n,i,j}\big|\Big) \quad (2)$$

where L is the number of layers in total. Noting that, the lasso loss L_{lasso}, as part of the final objective function, will guide the learning of FS $I^l_{N,K,K}$ to only assign large values to the most important stripes.

Those elements in $I^l_{N,K,K}$ less than a given pruning threshold δ will be frozen and pruned afterwards. Thus, the pruned FS $\tilde{I}^l_{N,K,K}$ can be defined as,

$$\tilde{I}^l_{n,i,j} = \begin{cases} I^l_{n,i,j}, if\ I^l_{n.i.j} > \delta \\ 0, \quad else \end{cases}, \quad (3)$$

where $\tilde{I}^l_{n,i,j}$ and $I^l_{n,i,j}$ are the elements at location (i, j) in the pruned and original FS. Thus, the actual filter weights pruned for inference is $W^l_{n,c,i,j} \times \tilde{I}^l_{n,i,j}$.

To increase the inference efficiency of the pruned network, the standard convolution (1) can be reformulated in a stripe-wise manner,

$$X^{l+1}_{n,\mathrm{h},\mathrm{w}} = \sum_{i=1}^{K}\sum_{j=1}^{K} \tilde{I}^l_{n,i,j} \sum_{c=1}^{C} W^l_{n,c,i,j} \times X^l_{c,\mathrm{h}+i-1,\mathrm{w}+j-1} \quad (4)$$

The right summation in (4) can be viewed as the convolution over a given stripe $\bar{W}^l_{n,i,j} = \{W^l_{n,1,i,j}, W^l_{n,2,i,j}, \dots, W^l_{n,C,i,j}\}$, $i, j = 1, 2, \dots, K$, while some stripes are not used, i.e. pruned, if $\tilde{I}^l_{n,i,j} = 0$. These survived stripes can then be grouped together without changing (4) to perform a true stripe-wise convolution as shown in the right top part of Fig. 2. Noting that the size of pruned FS $\tilde{I}^l_{N,K,K}$ is negligible compared to the whole model.

2.2. Balanced Pruning Strategy

The basic assumption is that the value of $\tilde{I}^l_{n,i,j}$ indicates the importance of the corresponded stripe $\bar{W}^l_{n,i,j}$. That is why the pruning strategy in SWP [16] is simply relied on an empirical coefficient δ. However, the location (i, j) and filter index n also indicates some more information, such as the local spatial distribution of the stripes and the correlation

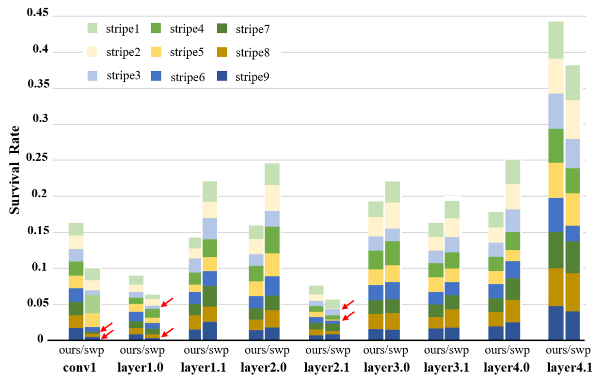

Fig. 3. The survival rate of intra-filter stripes in ResNet18.

between different channels of the output feature maps. To utilize such information in FS, a new pruning strategy considering the balance between inter-filter and intra-filter stripes is presented in this section, while and layer-wise balance is discussed in Section 2.3.

If the stripe $\bar{W}^l_{n,i,j}$ at location (*i, j*) is pruned, it means the information at (*i, j*) within the *K*×*K* local window is not used in producing the output feature maps $X^{l+1}_{N,H,W}$. If the stripes at certain locations are rarely survived as those marked by red arrows in Fig. 3, some useful patterns may be missing in $X^{l+1}_{N,H,W}$. It may even degrade the generalization ability of the model. To address this issue, a new constraint is imposed to keep the numbers $S^l = \{S^l_1, S^l_2, \dots, S^l_{K\times K}\}$ of different intra-filter stripes to be balanced. Specifically, a new loss function L_s is defined based on the variation,

$$L_s = \frac{1}{K \times K} \sum_{l=1}^{L} \sum_{m=1}^{K\times K} \left(S^l_m - \bar{S}^l\right)^2 \qquad (5)$$

where *m*=*i*×(*K*-*1*)+*j* is the 1-D index of intra-filter stripes and $\bar{S}^l$ denotes the average value of S^l.

Similarly, the stripe imbalance between *N* filters at *l*-th layer may also cause performance degradation. Since *N* is usually a larger number than *K*×*K*, some channels of the output feature map may be highly correlated in the extreme case that only 1 or 2 stripes survived in certain filters. It may reduce the representation capacity of the model with those redundant parameters. Thus, the second balance strategy is proposed in this work to avoid such extreme case for inter-filter stripes. The number of survived stripes in each filter can be denoted as $f^l = \{f^l_1, f^l_2, \dots, f^l_N\}$, while the second constraint is defined by the inter-filter loss L_f,

$$L_f = \frac{1}{N} \sum_{l=1}^{L} \sum_{n=1}^{N} \left(f^l_n - \bar{f}^l\right)^2 \qquad (6)$$

where $\bar{f}^l$ is the average value of f^l.

It is worth noticing that it is hard to compute the gradient of the thresholding function (3). Thus the following formula is used to simulate (3):

$$\sigma\left(I^l_{n,i,j}\right) = \frac{1}{e^{-q\times\left(I^l_{n,i,j}-\delta\right)} + 1} \qquad (7)$$

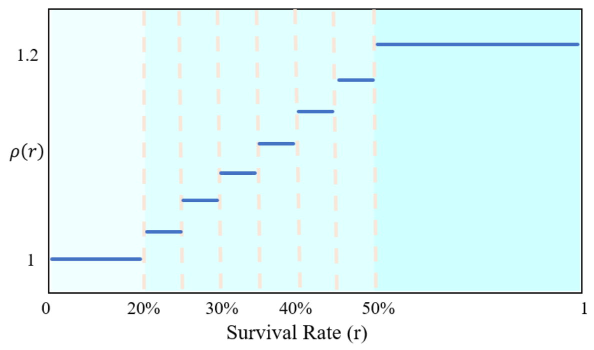

Fig. 4. The piecewise adjustment function $\rho(r)$.

where q is a manually set coefficient.

In summary, the final loss function can be written as,

$$L = L_{cla} + \lambda_1 L_{lasso} + \lambda_2\left(\mu L_s + (1-\mu)L_f\right), \qquad (8)$$

where L_{cla} is the specific task loss (e.g., cross-entropy loss), λ_1 and λ_2 control the magnitudes of regularization and μ is the weight balancing the contribution of L_s *and* L_f.

2.3. Dynamic Pruning Threshold

In the vanilla SWP [16], the pruning threshold δ is set to a fixed value for all convolutional layers. However, the value of δ is not directly connect to the pruning rate at each layer, which introduces another kind of layer-wise imbalance. Since it unable to control the exact pruning rate at specific layer, undesired poor perfomance may be triggered by a too low surviving rate in mid-layers.

Therefore, the threshold should be dynamically adjusted during the training stage for each layer, in order to make the layer-wise survival rates to be relatively balanced. To be specific, the survival rates of every layer in the entire model are counted every 10 training epochs. Different pruning thresholds are assigned according to their current survival rates. In other words, a higher pruning threshold will be given to those layers with a higher survival rate, and vice versa. This dynamic pruning threshold δ^l at l-th layer is determined as,

$$\delta^l = \delta_0 \times \rho(r), \qquad (9)$$

where δ_0 denotes initial pruning threshold, r is the survival rate at l-th layer, and $\rho(r)$ is an adjustment function. In the experiments, $\rho(r)$ is set a piecewise function shown in Fig. 4.

3. EXPERIMENTS

Different datasets (CIFAR-10 and TinyImageNet-200) and different network architectures (ResNet [15] and VGG nets [16]) have been conducted to evaluate the effectiveness of the proposed BSWP. The experimental results are also compared with other state-of-the-arts (SOTA) pruning methods. For a fair comparison, the network is not fine-tuned after pruning for all datasets and models.

3.1. Implementation Details

Table 1: Pruning results of ResNet56 on CIFAR-10.

ResNet56			
Metrics	Param (%) ↓	FLOPS (%) ↓	Acc.
Baseline [17]	0	0	**93.10**
L1 [12]	13.7	27.6	**93.12**
CP [10]	-	50	92.10
NISP [19]	42.6	43.6	93.07
DCP [20]	70.3	47.1	**93.11**
IR [21]	-	67.7	92.70
GBN [22]	66.7	70.3	93.07
HRnk [23]	68.1	74.1	90.72
SWP [16]	75.6	77.7	92.98
BSWP	**75.8**	**81.1**	**93.10**

The CIFAR-10 contains 60K color images of 32 × 32 pixels in 10 different classes, which is divided into 50K training images and 10K testing images. TinyImageNet-200 contains 110K color images of 200 different categories, while 100K images are used for training and 10K images for test. The size of the image is 64×64 pixels. On CIFAR-10, our method is applied to ResNet56[17] and VGG16[18], while ResNet18 is used on TinyImageNet-200.

Most hyper parameters are the same as SWP [16]. We used stochastic gradient descent (SGD) optimizer with momentum 0.9 and weight decay 1e-4 for training. For CIFAR-10, the mini batch size is set to 64 and the model is trained for 260 epochs. The initial learning rate is 0.1 and divide it by 10 after [0.5, 0.75] training epochs. The strength of sparsity-induced λ_1 is set to 1e-5 the same as [16]. λ_2 is set to 1e-6 and μ is set to 0.4. For TinyImageNet-200, the model is trained for 200 epochs with a batch size of 64. The initial learning rate is 0.1 and divide by 10 after 180 training epochs. Both λ_1 and λ_2 are set to 2e-5. And μ and q are set to 0.4 and 500. The initial pruning threshold δ_0 for ResNet56, VGG16 ResNet18 are 0.04, 0.03 and 0.04, respectively.

3.2. Compare with different pruning methods

Comparing with other recent state-of-arts pruning works, the results of the proposed BSWP with ResNet56 on CIFAR-10 is shown in Table 1. All the works except SWP are channel-wise or filter-wise pruning methods. It can be seen that our BSWP reduces the number of parameters by 75.8% and the number of Flops by 81.1% without losing network accuracy. As shown in Table 2, when VGG16 is used as the baseline model, the proposed model reduces the number of parameters by 93.83% and the FLOPS by 76.34%. It is worth noting that the network accuracy (93.84%) is even higher than the baseline.

The results with ResNet18 on TinyImageNet-200 is shown in Table 3. Similarly, the BWSP achieves a higher boost than both HGSR [28] and SWP. It shows that our model can reduce the number of parameters by 77.63% and the FLOPS by 50.07%, while still improving the network accuracy compared to baseline.

Table 2: Pruning results of VGG16 on CIFAR-10.

VGG16			
Metrics	Param (%) ↓	FLOPS (%) ↓	Acc.
Baseline [18]	0	0	93.26
ThiNet [24]	63.95	64.02	90.76
L1 [12]	64	34.3	93.40
SSS [13]	73.8	41.6	93.02
SFP [25]	63.95	63.91	92.08
GAL [26]	77.6	39.6	92.03
Hinge [27]	80.05	39.07	93.59
HRank [23]	82.9	53.5	93.43
SWP [16]	92.66	71.16	93.65
BSWP	**93.83**	**76.34**	**93.84**

Table 3: Pruning results of ResNet18 on TinyImageNet-200.

ResNet18			
Metrics	Param (%) ↓	FLOPS (%) ↓	Acc.
Baseline [17]	0	0	62.90
HGSR [28]	34.18	-	61.25
SWP [16]	73.91	45.24	63.06
SWP+DPT	**79.84**	**55.75**	62.60
SWP+L_s	76.48	47.21	63.11
SWP+L_f	74.09	47.01	63.15
BSWP	77.63	50.07	**63.35**

3.3. Ablation Study

To further demonstrate the effectiveness of the proposed balanced pruning strategy and dynamic pruning threshold (DPT). ResNet18 is chosen for the ablation experiments on TinyImageNet-200 as shown in Table 3. Three more sets of experiments (SWP+ L_s , SWP+ L_f and SWP+DPT) are presented. DPT achieves the highest parameter and FLOPS reduction, but its accuracy is the lowest. The balanced pruning strategy reduces parameters and FLOPS, as well as improves the accuracy. Combing all of them, the proposed BSWP achieves the highest accuracy.

4. CONCLUSION

In this work, we propose a new balanced stripe-wise pruning algorithm, which is a variant of the original SWP. Under the interaction of balanced pruning strategy and dynamic pruning threshold, the stripe distribution is balanced after pruning. Comprehensive experiments and analysis have been conducted on public data sets (CIFAR-10 and TinyImageNet-200), which shows the effectiveness of the proposed BSWP.

5. ACKNOWLEDGMENT

This work was supported by the Scientific Innovation 2030 Major Project for New Generation of AI, Ministry of Science and Technology of the People's Republic of China (No. 2020AAA0107300), Zhejiang Provincial Natural Science Foundation of China (No. LY20F030005) and National Natural Science Foundation of China (No. 61603202).

6. REFERENCES

[1] S. Ghosh, R. Shet, P. Amon, A. Hutter, and A. Kaup, "Robustness of Deep Convolutional Neural Networks for Image Degradations," In *ICASSP*, pp. 2916-2920, 2018.

[2] M. Zhu, K. Han, C. Zhang, J. Lin, and Y. Wang, "Low-resolution Visual Recognition via Deep Feature Distillation," In *ICASSP*, pp. 3762-3766, 2019.

[3] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," In *CVPR*, pp. 770-778, 2016.

[4] G. Srinivasan, A. Illa, and P. K. Ghosh, "A Study on Robustness of Articulatory Features for Automatic Speech Recognition of Neutral and Whispered Speech," In *ICASSP*, pp. 5936-5940, 2019.

[5] A. Graves, A. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," In *ICASSP*, pp. 6645-6649, 2013.

[6] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," In *NeurIPS*, pp. 91-99, 2015.

[7] Q. Mao, C. Wang, S. Yu, Y. Zheng, and Y. Li, "Zero-Shot Object Detection With Attributes-Based Category Similarity," In *TCAS-II*, pp. 921-925, 2020.

[8] W. Liu, et al., "Dynamic Relevance Learning for Few-Shot Object Detection," arXiv preprint arXiv: 2108.02235, 2021

[9] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, and C. Zhang, "Learning efficient convolutional networks through network slimming," In *ICCV*, pp. 2736-2744, 2017.

[10] Y. He, X. Zhang, and J. Sun, "Channel Pruning for Accelerating Very Deep Neural Networks," In *ICCV*, pp. 1398-1406, 2017.

[11] W. Chen, C. Wang, Z. Zhang, Z. Huo, and L. Gao, "Reweighted Dynamic Group Convolution," In *ICASSP*, pp. 3940-3944, 2021.

[12] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. Graf, "Pruning filters for efficient convnets," arXiv preprint arXiv:1608.08710, 2016.

[13] Y. He, G. Kang, X. Dong, Y. Fu, and Y. Yang, "Soft Filter Pruning for Accelerating Deep Convolutional Neural Networks," In *IJCAI*, pp. 2234–2240, 2018.

[14] S. Han, J. Pool, J. Tran, and W. Dally, "Learning Both Weights and Connections for Efficient Neural Networks," In *NeurIPS*, pp. 1135–1143, 2015.

[15] Z. Liu, J. Xu, X. Peng, and R. Xiong, "Frequency-Domain Dynamic Pruning for Convolutional Neural Networks," In *NeurIPS*, pp. 1043–1053, 2018.

[16] F. Meng, H. Cheng, K. Li, H. Luo, X. Guo, G. Lu, and X. Sun, "Pruning filter in filter," arXiv preprint arXiv:2009.14410, 2020.

[17] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," In *CVPR*, pp. 770–778, 2016.

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

[19] R. Yu, A. Li, C. Chen, J. Lai, V. Morariu, X. Han, M. Gao, C. Lin, and L. Davis, "Nisp: Pruning networks using neuron importance score propagation," In *CVPR*, pp. 9194–9203, 2018.

[20] Z. Zhuang, M. Tan, B. Zhuang, J. Liu, Y. Guo, Q. Wu, J. Huang, and J. Zhu, "Discrimination-aware channel pruning for deep neural networks," In *NeurIPS*, pp. 875–886, 2018.

[21] H. Wang, Q. Zhang, Y. Wang, L. Yu, and H. Hu, "Structured pruning for efficient convnets via incremental regularization," In *IJCNN*, pp.1–8, 2019.

[22] Z. You, K. Yan, J. Ye, M. Ma, and P. Wang, "Gate decorator: Global filter pruning method for accelerating deep convolutional neural networks," In *NeurIPS*, pp. 2130–2141, 2019.

[23] M.Lin, R.Ji, Y.Wang, Y.Zhang, B.Zhang, Y.Tian, and L. Shao, "Hrank: Filter pruning using high-rank feature map," In *CVPR* , pp. 1529-1538, 2020.

[24] J. Luo, J. Wu, and W. Lin, "Thinet: A filter level pruning method for deep neural network compression," In *ICCV*, pp. 5058-5066, 2017

[25] Z. Huang, N. Wang, "Data-driven sparse structure selection for deep neural networks," In *ECCV*, pp. 304-320, 2018

[26] S. Lin, R. Ji, C. Yan, B. Zhang, L. Cao, Q. Ye, F. Huang, and D. Doermann, "Towards optimal structured cnn pruning via generative adversarial learning," In *CVPR*, pp. 2790-2799, 2019.

[27] Y. Li, S. Gu, C. Mayer, L. Gool, and R. Timofte, "Group sparsity: The hinge between filter pruning and decomposition for network compression," In *CVPR*, pp. 8018-8027, 2020.

[28] K. Mitsuno and T. Kurita, "Filter Pruning using Hierarchical Group Sparse Regularization for Deep Convolutional Neural Networks," In *ICPR*, pp. 1089-1095, 2021.