A pruning method based on the dissimilarity of angle among channels and filters

1stJiayi Yao*, 2ndPing Li*[†], 3rdXiatao Kang*, 4thYuzhe Wang*

*Changsha University of Science and Technology, China

[†]Hunan Provincial Key Laboratory of Intelligent Processing of Big Data on Transp, China

email: yjy_toka@163.com, lping9188@163.com, kangxiatao@gmail.com, zheeln@qq.com

Abstract-Convolutional Neural Network (CNN) is more and more widely used in various fileds, and its computation and memory-demand are also increasing significantly. In order to make it applicable to limited conditions such as embedded application, network compression comes out. Among them, researchers pay more attention to network pruning. In this paper, we encode the convolution network to obtain the similarity of different encoding nodes, and evaluate the connectivity-power among convolutional kernels on the basis of the similarity. Then impose different level of penalty according to different connectivity-power. Meanwhile, we propose Channel Pruning base on the Dissimilarity of Angle (DACP). Firstly, we train a sparse model by GL penalty, and impose an angle dissimilarity (AD) constraint on the channels and filters of convolutional network to obtain a more sparse structure. Eventually, the effectiveness of our method is demonstrated in the section of experiment. On CIFAR-10, we reduce 66.86% FLOPs on VGG-16 with 93.31% accuracy after pruning, where FLOPs represents the number of floating-point operations per second of the model. Moreover, on ResNet-32, we reduce FLOPs by 58.46%, which makes the accuracy after pruning reach 91.76%. Our code is made public at: https://github.com/kangxiatao/prune_tf2_master. Index Terms—Convolutional neural network(CNN); network

pruning; angle dissimilarity; FLOP

I. INTRODUCTION

Convolutional Neural Networks (CNNs) bring excellent performance, which makes great achievements in image processing, speech recognition etc. Nevertheless, for devices with limited computation and memory, such as mobile embedded devices, even with the latest high-efficiency architecture, the size and over-parameterization of its model are still burdensome to deploy on neural networks, which will also affect the combination of CNNs and many traditional industries. Thus, network pruning provides the possibility and necessity for neural network compression.

Early studies have proposed many methods of network compression. Such as weight quantization [1], low-rank decomposition [2], knowledge distillation [3], pruning [4]–[7]. In this paper, we focus on network pruning. After simplification, the network not only reduces amounts of computation, but also improves its generalization capability. Early pruning methods are weight pruning mostly [8], which is practiced by directly deleting unimportant parameters. However it will cause unstructured sparsity of the model. Hence, by removing filters [4]–[6], [9], [10] or channels [11] from the convolutional

kernels, it will become a more widespread choice to leave a compact and coherent model by filter-level pruning.

At present, a great deal of pruning techniques base on redundancy and importance. Redundancy-based pruning usually counts the number of filters/channels as its redundancy. Important-based pruning depends on different definitions of importance for pruning. Some of them attach importance to norms, and nevertheless, this "norm-only" pruning has great limitations. For instance: (1) Remove the weight of the filters according to the norm value [12]. (2) Directly remove part of the filters [13]. (3) Sparse networks are left by pruning connections with redundancy or low weights [7]. Some of them do not take into account the diversity of distribution of filters among layers, so they will have a negative impact on accuracy [11]. Meanwhile, there are many pruning methods based on "smaller-norm-less-informative". In fact, small values are not equal to unimportant values [13]. Filters with small norm values in the front may play an important role behind [11].



Fig. 1. The connectivity-power of channels and filters on VGG-16. Horizontal axis represents the number of layers, and vertical axis represents the similarity between channels (filters) of each layer. Blue lines denote channels and yellow lines denote filters. Inspired by the performance ability of related models, we try to increase the difference and improve their performance ability.

We try to prune the model while maintaining its performance, and encode the convolutional layer into a hypercube network. When the m-base strings of the encoding nodes differ by at most one bit, they are defined as a pair of adjacent nodes. As shown in Fig. 1, we calculate the mean of adjacent nodes to evaluate the connectivity-power of channels and filters between 2D convolutional kernels. In addition, we attempt to impose different LASSO penalties on the model to make the model structured sparsity. However, LASSO can only zero out the parameters of a single feature. Features appear in the form of groups, and a whole group of parameters need to be zeroed out at the same time. To solve this problem, Yuan [14] proposed Group LASSO (GL) in 2006. As can be seen in Fig. 1(b), after compression the connectivitypower of filters and channels in the middle layer increases significantly. Meanwhile, the performance of model is weak. Among them, the connectivity-power is the similarity between the channels/filters of each layer. The dispersion of channels in each layer is measured to facilitate dynamic pruning, and the filters are pruned in the same way [13].



Fig. 2. Clustering analysis of GL constraint and AS constraint. A comparative experiment is conducted on VGG-16, with a total of 13-layers CNN, and each coordinate axis represents a single layer. The horizontal axis indicates the Euclidean distance between the channels or filters to the origin, and the vertical axis indicates AS between the channels/filters and their mean vector. The Euclidean distance and AS value are standardized between [0, 1]. Different colors represent different categories. n is the number of categories, and '+' represents the center point of category.

In order to solve the problems above, we propose a novel model penalty method, Channel Pruning Based on the Dissimilarity of Angle (DACP). Firstly, a sparse model is trained by GL penalty, and then the Similarity of Angle (AS) constraint is constructed on the channels/filters of the CNN to increase the difference between the channels/filters and further screen out important filters to obtain a more sparse structure. As can be seen in Fig. 2, DACP separates the similarity between channels/filters and indirectly changes its norm value combined with GL constraint. So that filters with small but important values can be retained and the performance of convolutional kernels can be enriched. Besides, channels/filters with larger but similar values are penalized. In this case, it not only improves the generalization but also makes the model sparse.

As for filter selection, DACP is somewhat similar to filter pruning [4], but the latter is obtained by penalty-inducedsparsity and considering that filters close to geometric median have more redundancy. As for processing, it is resemblance to Dynamic Pruning [15], which increases the connectivitypower of the model and further inhibits the connectivity-power of the redundancy. Besides, since our method is based on GL to penalize the sparsity parts, our method does not need to achieve with extra pre-training and to achieve a compact model structure in the end of training. Above all, traditional pruning methods can also be applied to our trained models.

In this paper, we make a new assumption and propose ours for the existing limitations and shortcomings, which solves the disadvantages of poor performance and generalization. Finally, for verifying the effectiveness of DACP, we use multiple image recognition datasets on multiple network architectures. We compare it with a variety of widely used pruning methods, and describe in detail in the third section of the paper.

II. RELATED WORK

As mentioned above, previous works can be divided into unstructured pruning and structured pruning. Unstructured pruning [7], [8], [16], [17] mainly includes weight pruning and neuron pruning, which results in unstructured sparse model. Structured pruning [4]–[6], [9], [10], [18] mainly contains filter pruning and channel pruning. The compact network obtained after pruning can maintain the original structure, but usually leads to a significant decrease in accuracy [10].

A. Unstructured Pruning

Originally, LeCun [8] trimmed all unimportant weights in the network to improve the accuracy and generalization of the network by considering the weight parameters in the network as a single parameter. Later, Hassibi [16] proposed OBS technique based on LeCun's [8] to improve the restoration of the weight updating. Compared with the former, OBS can prune more weight as the same error. Recently, pruning focuses on reserving important connections in network according to norm value [7], so as to reduce the number of parameters and computation consuming of the model without affecting the final accuracy of the network [7]. Neuron pruning sets the row/column of the matrix to zero, and the size of the matrix does not change. Hu [17] defined "average percentage of zero" to measure the number of zero activated in each filter, and regards its neurons as redundant and prunes as a whole.

B. Structured Pruning

Contrary to unstructured pruning, sparsity brought by structured pruning is regular, and will obtain structured model eventually. So it is more feasible currently. Filter pruning and channel pruning belong to structured pruning. Previous pruning based on norm value include the filters pruning on account of L1-norm [10], L2-norm [9] and Lp-norm [5]. Besides, there are also pruning methods based on redundancy, such as He's [4]. However, these methods always only lay emphasis on how to prune in the same layer, without noticing the relationship among different layers. Aiming at the limitations above, Luo [6] attaches more importance to the relationship among layers, and he supposed that statistics calculated in next layer are used as the benchmark for pruning. Further, Xie [18] proposed the concept of Extended Filter Group (EFG), which solves the training of filters of the current layer and corresponding channels of the next layer according to the penalty of EFG, and conducted induced-sparsity of the model. Our approach sorts out the above problems and put forward a more effective method to achieve better performance.

III. METHOD

Our pruning method can be summarized as the following steps: (1) Impose GL penalty on the model until what we set. (2)Reduce the GL penalty, and add the penalty of AD in channels until the final convergence. (3) Prune the channels according to the norm value of 3D filter. In this section, we will elaborate the setting of penalty terms and the learning process of structured sparsity.

A. Preliminaries

We formally introduce the symbols and notations in this subsection. We assume that a neural network has L layers, and in the i-th layer we parameterize CNN as $W = \{W^1, W^2, \ldots, W^L\}$ as $\{W^{(i)} \in \mathbb{R}^{k \times k \times c^l \times n^l}\}$, where k denotes kernel size, c^l and n^l denote l-th layer's number of channels and filters respectively. In addition, the convolutional kernel of the *n*-th filter in the *c*-th channel at the *l*-th layer is expressed as $W_{c,n}^l$.

B. Convolutional Kernels Grouping

First we divide the convolutional kernels of each layer into two groups: channel-filter and 3D-filter, and then impose L2norm penalty on each group in the objective function:

$$R_{g} = \sum_{l=1}^{\mathcal{L}} \left(\sum_{c=1}^{c^{l}} \left\| W_{c,n^{l}}^{l} \right\|_{2} + \sum_{n=1}^{n^{l}} \left\| W_{c^{l},n}^{l} \right\|_{2} \right)$$
(1)

Indeed we can also design more complex groups to obtain better sparse effect in the training process, but the penalty based on AS only needs group constraints to work simply.

C. Channel Pruning Based on the Dissimilarity of Angle

Channel is a group of multidimensional data in CNN's convolutional kernels. AS is used to impose constrains on the model according to the difference. AS is widely used in natural language processing and data mining to measure cohesion within data clustering. AS S(A, B) of vectors A and B is demonstrated as follows:

$$S(A,B) = 1 - \left(\frac{\cos^{-1}\left(f_{\text{similarity}}\left(A,B\right)\right)}{\pi}\right)$$
(2)

$$f_{\text{similarity}}(A, B) = \cos(\theta) = \frac{A \cdot B}{\|A\| \|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} (A_i)^2} \times \sqrt{\sum_{i=1}^{n} (B_i)^2}}$$
(3)

where A and B are vectors to calculate AS A_i and B_i denote the component of vector A and B respectively.

We calculate the norm value of each convolutional kernel in the channel, and transform the channels and filters into the vectors. Then calculate the value of DACP according to AS:

$$R_{c} = \sum_{l=1}^{\mathcal{L}} \sum_{i=1}^{c^{l}-1} \sum_{j=i+1}^{c^{l}} S\left(X_{i}^{l}, X_{j}^{l}\right)$$
(4)

where X_i^l and X_j^l denotes the *i*-th and *j*-th channel vector of the *l*-th layer respectively.

D. Loss function

In supervised learning, y represents the target tag. C(x, W) represents the forward propagation result of input data x in CNN. And $\mathcal{L}(\cdot)$ represents the target tag and the loss of output result. We use cross-entropy to calculate the fitting-error, and add DACP penalty \mathcal{R}_c and LASSO penalty \mathcal{R}_g to construct our channel pruning based on AD, which could be formulated as:

$$\hat{\mathcal{L}}(y, C(x, W)) = \mathcal{L}(y, C(x, W)) + \lambda \mathcal{R}_c + \beta \mathcal{R}_q \qquad (5)$$

where λ and β denote the hyperparameter of the penalty term based on AD and LASSO penalty term respectively, which is conducive to adjust the level of penalty.

As for DACP penalty term, we use cosine-similarity instead of angle's. The larger the angle is, the smaller the similarity will be. The cosine-similarity among channels and filters plays a decisive role in our parameter penalty. The derivative of $f_{similarity}$ by the single vector X_i is as follows:

$$\frac{\partial(f)}{\partial X_{i}} = \frac{X_{j} \left(\|X_{i}\| \|X_{j}\| \right) - \left(X_{i}^{T}X_{j}\right) \frac{X_{i}\|X_{j}\|}{\|X_{i}\|}}{\|X_{i}\|^{2} \|X_{j}\|^{2}} = \left(\frac{1}{\left(X_{i}^{T}\right)} - \frac{X_{i}}{\|X_{i}\|^{2}}\right) f_{\text{similarity}}$$
(6)

where the $f_{similarity}$ denotes the cosine-similarity of channel vector X_i and X_j . Thus the gradient of penalty term with AD can be simplified as:

$$\nabla_{w} \lambda \mathcal{R}_{c} \propto \left(\left(\mathbf{W}^{T} \right)^{-1} - \frac{\mathbf{W}}{\|\mathbf{W}\|^{2}} \right) f_{w}$$
(7)

where f_w is the matrix of AS between vectors of channel.

It can be concluded from the gradient that the similarity between channels changes during the optimization processing. Combined with the constraints of GL, the similarity of multiple channels and filters with high similarity is penalized by AD. The similarity decreases gradually, and the norm value of some channels decreases sharply. After pruning, the model will become more compact, and generalization even improved.

E. Simplify the Calculation of Similarity

On VGG-16 model, the number of channels and filters reaches 512 after the 8-th layer, which means that the calculation of AS is huge. After calculating, the time complexity is $O(n^3)$. The larger the value of n is, the larger the floatingpoint calculation resource is.

Considering the complexity to calculate AS among channels, we attempt to calculate the mean vector B of channels and regard it as the base vector instead. We calculate AS between each channel and the base vector for approximation.

Then the penalty term \mathcal{R}_c of AD can be replaced by \mathcal{R}_c' in Eq.8.

$$\mathcal{R}_{c}^{\prime} = \sum_{l=1}^{L} \sum_{i=1}^{c^{*}} \mathrm{S}\left(X_{i}^{l}, B^{l}\right)$$
(8)

where B^l denotes the mean vector of channel of the *l*-th layer. Meanwhile, with the calculating of the channel vector, the time complexity after approximation processing is $O(n^2)$, which seems not to be significantly improved. However, when calculating AS with the base vector. We can directly calculate AS in the form of tensor, and the actual calculation reduces.

IV. EXPERIMENTS

A. Databases and Experimental Settings

We evaluated our approach on VGG-16, and ResNet Deep Neural Networks (DNN) with the classical datasets: MNISIT, Caltech-101, CIFAR-10, and CIFAR-100.

We selected the following pruning methods that have worked well in both industry and academia to compare them with our experiments in turn. (1)Network Slimming (NS) [19] applies L1 regularization to the scaling factor of batch normalization (BN) layer. L1 regularization inclines the scaling factor of BN layer to zero, so as to distinguish unimportant channels or neurons. (2) Soft Filter Pruning (SFP) [9], a dynamic pruning method, which can enable the pruning filter to participate in certain training to improve the efficiency of pruning. (3) Filter Pruning via Geometric Median (FPGM) [4], a network-compression method based on geometric center pruning. (4) Stripe-wise Pruning (SWP) [20]. A learnable matrix is introduced to reflect the shape of each filter, and the matrix is used to guide model pruning.

Before model training, in addition to MNIST, we make data augmentation of random trimming and random mirror for other datasets. During the training, we have an arrangement for cosine learning rate decay. Finally, we use the decline rate of FLOPs as the pruning rate to evaluate the performance after pruning.

B. Experiments on VGG-16

VGG-16 is a 16-layer-single-branch CNN with 13 convolutional layers. We tested the performance of VGG-16 network on CIFAR-10, CIFAR-100 and Caltech-101 by using various pruning methods. Table I reveals the relevant experiment results. Among them, NS [19] and SFP [9] are the data in the initial papers by others, and others are from our experiments.

In order to analyze the influence of GL penalty-intensity on AD, we have done a lot of experiments on CIFAR-10. As shown in Table II, the performance of AD on strong GL penalty is relatively stable in pruning and it generalizes better. When AD acts on the GL with weak penalty, its performance is stable in generalization and pruning improves.

C. Experiments on ResNet

ResNet is a multi-branch neural network structure composed of multiple residual blocks. Different from single-branch networks, pruning tends to cause a mismatch between the number

TABLE IPerformance of VGG-16.

Datasets	Method ¹	Pruned FLOPs(%)	Pruned Accuracy(%)
	baseline with L2*	0	74.23
	L1*	33.73	71.59
CIFAR-100	NS [19]	37.1	72.09
CITAR-100	GL^*	39.84	72.18
	SFP [9]	41.8	70.28
	Ours	42.18	73.12
	baseline with L2*	0	93.73
	L1*	60.20	92.45
CIEAP 10	NS [19]	51.0	92.8
CITAR-10	GL^*	66.57	92.51
	SWP [20]	71.16	92.85
	Ours	66.86	93.31
	baseline with L2*	0	95.28
Caltech 101	L1*	20.3	95.1
Cancell-101	GL^*	16.3	93.5
	Ours	22.95	95.55

¹ In the "Method" column, "baseline with L2" means that L2 regularization is used in the model; "Pruned FLOPs" means that the model reduces the number of FLOPs to be lost.

^{*} The experiment is implemented by us.

TABLE II The influence of different level of penalty of GL on the dissimilarity of angle.

Method	Pruned FLOPs (%)	Pruned accuracy (%)
ORI	0	$91.20(\pm 0.30)$
L1	$60.20(\pm 6.80)$	$92.45(\pm 0.35)$
L2	$0.06(\pm 0.06)$	93.73 (±0.42)
$GL(1)^{1}$	$72.10(\pm 4.50)$	$92.40(\pm 0.30)$
$GL(2)^{2}$	$56.83(\pm 3.10)$	$93.10(\pm 0.32)$
$GL(1)^{1}+ ad^{3}$	75.30 (±4.25)	$92.81(\pm 0.44)$
$GL(2)^{2}+ ad^{3}$	$66.86(\pm 4.50)$	$93.31(\pm 0.26)$

¹ Group LASSO with strong penalty.

² Group LASSO with weak penalty.

³ Our AD penalty.

of shortcuts filters for the residual block and the number of output filters. We solve this problem by taking their union.

From Table III, we can compare the pruning of various methods on CIFAR-10 and CIFAR-100s. ResNet itself is a compact network. There is no obvious difference in accuracy between ours and others, but our method has distinct advantages in pruning rate. For shortcuts in ResNet reduce the number of effective filters in the residual blocks, and our approach screens out important filters from a similarity term.

D. Feature Visualization

Further, we realize visualization analysis of output features in the model. On CIFAR-10, make visualization analysis and comparation on the output feature of second convolutional kernel on VGG-16, which is shown in Fig. 3. We select a car picture as the input feature. After the convolutional operation of convolutional kernel of the second layer, we intercepted the output feature of GL method and DACP, as depicted in Fig. 3(b) and Fig. 3(c) respectively. It is obvious that GL has a higher similarity of adjacent features, while AD has a more

Datasets	Model	Method	Pruned FLOPs(%)	Pruned
CIFAR -10		baseline with L2*	0.2	<u>92.37</u>
	ResNet	$L1^*$	22.62	90.55
	-18	GL^*	37.08	90.58
		Ours	52.5	90.51
		baseline with L2*	0	92.63
		$L1^*$	24.5	90.75
	ResNet	GL^*	40.26	90.56
	-20	SFP [9]	42.2	90.83
		FPGM [4]	54	90.44
		Ours	62.5	90.61
		baseline with L2*	0	93.65
	ResNet	$L1^*$	38.85	91.25
		GL^*	42.65	91.64
	-32	SFP [9]	41.5	92.08
		FPGM [4]	53.2	91.93
		Ours	58.46	91.76
CIFAR -100		baseline with L2*	0	74.65
	ResNet	$L1^*$	21.6	73.02
	-18	GL^*	34.83	73.28
		Ours	43.16	73.24
		baseline with L2*	0	74.86
	ResNet	$L1^*$	21.63	73.1
	-34	GL^*	48.11	72.47
		Ours	65.49	72.51

TABLE III Performance of ResNet.

* The experiment is implemented by us. Some methods do not show the best performance due to the influence of hyperparameters, but we try our might to get the best results and then conduct correlative experiments.

well-stacked feature map. The feature map of the red box in Fig. 3(b) can be replaced by the feature map in green box in Fig. 3(c) after AD penalty is applied. The features of the orange box in Fig. 3(b) are changed to those of the blue box in Fig. 3(c), which can be regarded as the restoration of filters.



Fig. 3. Feature visualization. There are 64 filters in the second layer of VGG-16. Totally black blocks in the figure indicate that the filters have been cut off.

V. CONCLUSION

In the experiment, it can be seen that 3D-filters show different levels of clustering while applying different level of LASSO penalties. Compared with L1 penalty, GL penalty has better clustering effect. The distribution of 3D-filters is relatively uniform, and the degree of model simplification is higher. Combined with this circumstance, DACP is proposed. The relationship of channels and filters in convolutional kernel is discussed, and the corresponding DACP is constructed with cosine similarity. Experiment results show that the DACP can make the 3D-filter clustering in the model more uniform and improve the sparsity of the model.

In this paper, the calculation of DACP adopts basis vector approximation, which will be more convictive if a more suitable way can be found to express it. In the future, we will start with the connectivity-power that mentioned in the beginning of this article. We attempt to guide model sparsity based on connectivity-power, and play a role of regularization to push the performance to a higher stage.

REFERENCES

- W. Chen, J. Wilson, S. Tyree, K. Weinberger, and Y. Chen, "Compressing neural networks with the hashing trick," in *International conference* on *Machine Learning*, pp. 2285–2294, PMLR, 2015.
- [2] E. L. Denton, W. Zaremba, J. Bruna, and LeCun, "Exploiting linear structure within convolutional networks for efficient evaluation," Advances in Neural Information Processing Systems, vol. 27, 2014.
- [3] P. Chen, S. Liu, H. Zhao, and J. Jia, "Distilling knowledge via knowledge review," in *Proceedings of the IEEE/CVF Conference on Computer Vision*, pp. 5008–5017, 2021.
- [4] Y. He, P. Liu, Z. Wang, Z. Hu, and Y. Yang, "Filter pruning via geometric median for deep convolutional neural networks acceleration," in *Proceedings of the IEEE/CVF Conference on Computer Vision*, pp. 4340–4349, 2019.
- [5] Y. He, Y. Ding, P. Liu, and L. Zhu, "Learning filter pruning criteria for deep convolutional neural networks acceleration," in *Proceedings of the IEEE/CVF conference on computer vision*, pp. 2009–2018, 2020.
- [6] J.-H. Luo, J. Wu, and W. Lin, "ThiNet: A filter level pruning method for deep neural network compression," in *Proceedings of the IEEE international conference on computer vision*, pp. 5058–5066, 2017.
- [7] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," Advances in Neural Information Processing Systems, vol. 28, 2015.
- [8] Y. LeCun, J. Denker, and S. Solla, "Optimal brain damage," Advances in neural information processing systems, vol. 2, 1989.
- [9] Y. He, G. Kang, X. Dong, Y. Fu, and Y. Yang, "Soft filter pruning for accelerating deep convolutional neural networks," *ArXiv Prepr* arXiv:1808.06866, 2018.
- [10] H. Li, A. Kadav, I. Durdanovic, and H. Samet, "Pruning filters for efficient convnets," ArXiv Prepr arXiv:1608.08710, 2016.
- [11] Y. He, X. Zhang, and J. Sun, "Channel pruning for accelerating very deep neural networks," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1389–1397, 2017.
- [12] Z. Mariet and S. Sra, "Diversity networks: Neural network compression using determinantal point processes," *Computer Science*, 2015.
- [13] B. Bartoldson, A. Morcos, A. Barbu, and G. Erlebacher, "The generalization-stability tradeoff in neural network pruning," *Advances* in Neural Information Processing Systems, vol. 33, pp. 20852–20864, 2020.
- [14] M. Yuan and Y. Lin, "Model selection and estimation in regression with grouped variables," *Journal of the Royal Statistical Society: Series B*, 2006.
- [15] Y. Guo, A. Yao, and Y. Chen, "Dynamic network surgery for efficient dnns," Advances in neural information processing systems, vol. 29, 2016.
- [16] B. Hassibi and D. Stork, "Second order derivatives for network pruning: Optimal brain surgeon," Advances in Neural Information Processing Systems, vol. 5, pp. 164–171, 1992.
- [17] R. Yu, A. Li, C.-F. Chen, J.-H. Lai, V. I. Morariu, X. Han, and M. Gao, "NISP: Pruning networks using neuron importance score propagation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 9194–9203, 2018.
- [18] Z. Xie, P. Li, F. Li, and C. Guo, "Pruning filters base on extending filter group lasso," *IEEE Access*, vol. 8, pp. 217867–217876, 2020.
- [19] Z. Liu, J. Li, Z. Shen, G. Huang, and S. Yan, "Learning efficient convolutional networks through network slimming," in *Proceedings of* the IEEE international conference on computer vision, pp. 2736–2744, 2017.
- [20] F. Meng, H. Cheng, K. Li, H. Luo, X. Guo, G. Lu, and X. Sun, "Pruning filter in filter," Advances in Neural Information Processing Systems, vol. 33, pp. 17629–17640, 2020.