Exploiting Weight-Level Sparsity in Channel Pruning with Low-Rank Approximation

Zhen Chen, Jianxin Lin, Sen Liu, Zhibo Chen, Weiping Li
CAS Key Laboratory of Technology in
Geo-spatial Information Processing and Application System
University of Science and Technology of China
Hefei, China

Jin Zhao, Wei Yan
Department of Integrated Circuits
and Intelligent Systems
Peking University
Beijing, China

***Abstract*—Acceleration and compression on Deep Neural Networks (DNNs) have become a critical problem to develop intelligence on resource-constrained hardware, especially on Internet of Things (IoT) devices. Previous works based on channel pruning can be easily deployed and accelerated without specialized hardware and software. However, weight-level sparsity is not well explored in channel pruning, which results in relatively low compression rate. In this work, we propose a framework that combines channel pruning with low-rank decomposition to tackle this problem. First, the low-rank decomposition is utilized to eliminate redundancy within filter, and achieves acceleration in shallow layers. Then, we apply channel pruning on the decomposed network in a global way, and obtains further acceleration in deep layers. In addition, a spectral norm-based indicator is proposed to balance low-rank approximation and channel pruning. We conduct a series of ablation experiments and prove that low-rank decomposition can effectively improve channel pruning by generating small and compact filters. To further demonstrate the hardware compatibility, we deploy the pruned networks on the FPGA, and the networks produced by our method have obviously low latency.**

***Index Terms*—deep learning, network acceleration, channel pruning, low-rank decomposition, hardware resources.**

I. Introduction

In recent decades, Internet of Things has been applied in various fields including industry, health and transport [1]–[3]. Meanwhile, deep learning, especially convolutional neural networks (CNNs), has shown superior performance on a series of artificial intelligence tasks, e.g. image classification [4] and object detection [5]. The trend is obvious that the intelligence of deep learning is indispensable for the development of IoT [6]. However, the high requirement on hardware resources hinders the prevailing of deep learning on IoT applications. In time-sensitive tasks, e.g. driving and health monitoring, uploading data to cloud servers would induce intolerable latency. For real-time reactions, efficient and hardware-friendly CNNs are significant to edge computing and embedded devices.

The hardware implementation of CNNs is mainly limited by model storage, computation cost and memory footprint. There are many works attempting to reduce the resource usage with negligible performance decay. But most of these approaches can only achieve one or two aforementioned goals. In order to directly obtain acceleration and compression benefits on general platforms, it's necessary to ensure the compact structure of the processed convolutional layer. Many existing works based on low-rank decomposition and channel pruning have tried to satisfy these requirements. However, there still exist some limitations of these works.

Low-rank decomposition generates a compact and efficient network by abandoning unimportant components in a layer-oriented way. These methods provide layer-wise optimal solution under a certain metric. However, the error of approximation is accumulated layer by layer. Due to the lack of global consideration, small errors in each layer may have a serious impact on the entire network. Besides, it needs burdensome trials to find the optimal ranks.

To pursue the compact structure, channel pruning utilizes the filter-level sparsity as an aggressive approach. But as a cost, the lack of flexibility becomes a bottleneck with some disadvantages. (1) Channel pruning cannot make use of weight-level sparsity, and results in relatively low compression rate. (2) Since each filter is either entirely deleted or kept, pruning wrong filters may discard lots of important weights and bring unrecoverable damage on performance.

To tackle these problems in channel pruning, one practicable way is to generate networks with small and compact filters before pruning. In this way, the redundancy within filter is removed, and pruning the wrong filters is likely to be avoided. Fortunately, this solution can be achieved by applying low-rank decomposition as a pre-processing. On the other hand, combining with channel pruning help low-rank decomposition to handle redundancy from a cross-layer perspective.

In this paper, we propose a pruning framework combined with low-rank decomposition. In the first step, we apply low-rank decomposition to eliminate redundancy especially in shallow layers, which makes the next channel pruning more robust to decision errors. In the second step, channel pruning is utilized to discard unimportant parameters in a global way, and obtain acceleration especially in deep layers. Besides, we introduce a spectral norm-based indicator to achieve a better balance between these two steps. Under our framework, processed networks retain the structural compactness. Experiments show that our method effectively achieves improvement

*Corresponding authors: Zhibo Chen (chenzhibo@ustc.edu.cn) and Wei Yan (yanwei@ss.pku.edu.cn). This work was supported in part by NSFC under Grant 61571413, 61632001, 61390514.

978-1-7281-0397-6/19/$31.00 ©2019 IEEE

on acceleration and compression simultaneously. In hardware deployment, we implement pruned networks on FPGA and our method obtains the minimum latency, which is crucial in further implementation on IoT devices.

II. Related Work

To obtain practical speedup, low-rank decomposition based approaches factorize each convolutional layer into several efficient ones. To keep network performance, they approximate the weight matrix or tensor with low-rank techniques, e.g., truncated Singular Value Decomposition (SVD) [7], CP-decomposition [8] and Tucker decomposition [9]. [10] provided a closed-form solution for SVD-based decomposition.

Pruning can be categorized by different granularity levels, including weight-level, kernel-level, channel-level and layer-level [11], [12]. Weight pruning has higher flexibility and obtains better storage compression by representing the sparse pruned network in compressed sparse column (CSC) [13], but it cannot reduce inference time and memory footprint without specialized hardware [14]. On the contrary, layer-level pruning is impracticable unless network depth is sufficient [15].

Compared to these two extremes, channel pruning provides a trade-off between flexibility and ease of implementation. Channel pruning regards each filter as the basic unit and discards the filters that are considered to be insignificant, and finally produces a smaller network with compact structure. The main challenge hinges on employing the right pruning criterion. Some works treated filter selection as an iterative optimization problem, with empirical trials to balance pruning ratio and accuracy decay layer by layer [16]. Other works use different criteria to estimate the importance of filters, e.g., norms of weights or activations [17], the entropy of filter [18], the percentage of zeros in output feature maps [19], scale factors in batch normalization layers [20] and even to pick out unimportant filters by reinforcement learning [21]. Taylor pruning regarded the network loss as a differentiable function of intermediate activation, and the retained first-order term of Taylor expansion indicates the importance of feature map and corresponding filter [22]. Their experiments had proved that Taylor pruning outperforms other channel pruning methods.

III. Approach

A. Small Filters Make Channel Pruning Better

[22] found that the network would suffer serious performance decay once some filters are pruned, and other filters seem inessential, which are called significant filters and insignificant ones respectively. Channel pruning wants to prune the least important filter at each iteration, but cannot avoid mistakes. For a certain pruning criterion, the probability that the pruning algorithm misjudges is assumed as a constant during pruning iterations, denoted as p. Noted that misjudging means picking any filter rather than the least important one.

Once pruning misjudges, to prune which wrong filter is regarded as an equal probability event under principle of maximum entropy [23]. For a network with m_0 significant filters and the rest m_1 insignificant ones, the probability of pruning a significant filter P_1 is:

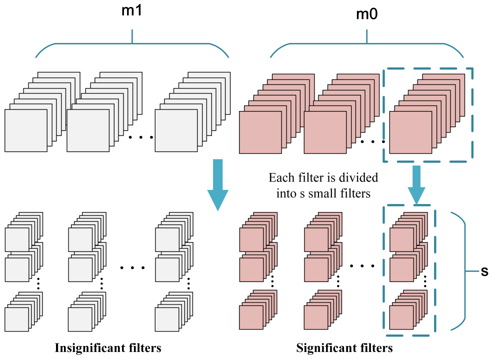

Fig. 1. Schema on reducing granularity. A network has m_0 significant filters and m_1 insignificant filters. Divide each filter into s smaller filters.

$$P_1 = \frac{m_0}{m_0 + m_1 - 1} p \,. \tag{1}$$

As shown in Fig. 1, each filter of the original network is converted into s smaller filters. As the decomposition methods, e.g. low-rank approximation, reserve major information, it is reasonable to assume the ratio between significant and insignificant filters doesn't change. For the decomposed network, the probability of pruning a significant filter $P_1^{'}$ is:

$$P_1^{'} = \frac{s m_0}{s(m_0 + m_1) - 1} p \,, \tag{2}$$

and $P_1^{'}$ is slightly smaller than P_1. More importantly, as the proportion of weights in each filter to the network is reduced by s times, severe mistakes are alleviated into recoverable.

Compared with pruning an important filter in the original network, pruning s corresponding important filters induces the equivalent degree of damage. But the probability $P_s^{'}$ that pruning s important filters in the decomposed network is much smaller than P_1 as follows:

$$P_s^{'} = \frac{C_{s m_0}^{s}}{C_{s(m_0+m_1)-1}^{s}} p^s \,, \tag{3}$$

where $s \ll m_0$, thus $P_s^{'}$ is approximate to $(\frac{m_0}{m_0+m_1})^s p^s$. As the analysis in [24], smaller filters relieve the damage in channel pruning and reduce the probability of severe mistake happens.

B. Combining Channel Pruning and Low-rank Decomposition Together

The aforementioned shortcomings limit the efficiency of channel pruning and low-rank decomposition. Fortunately, low-rank approximation and channel pruning can compensate limitations for each other. Besides, they both keep network compact, which is necessary for practical speedup.

We propose a hardware-friendly acceleration framework, which improves channel pruning with the help of low-rank approximation. These two steps of our proposed framework have clear functions. In this way, our method can avoid compressing the network in an aggressive way as well as the

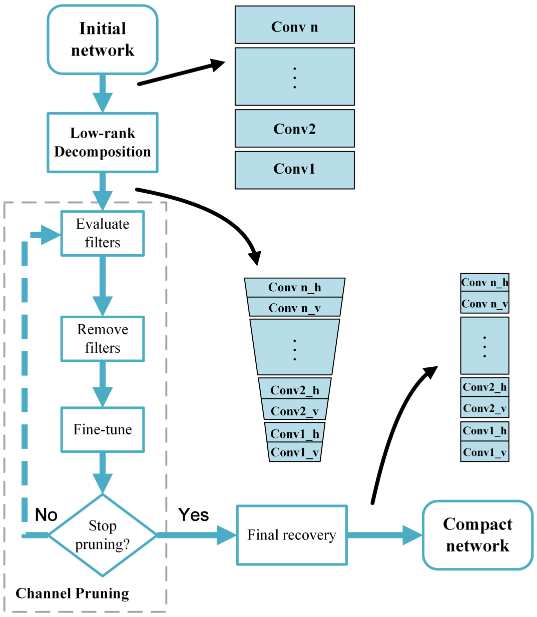

Fig. 2. The Low-rank decomposition-enhanced channel pruning framework. The area of the blue box stands for the FLOPs of the corresponding convolutional layer. The first step reduces most of computation in shallow layers, and the second step diminishes parameters globally as well as most of computation in deep layers. The method generates a more efficient network with compact structure.

bottleneck of performance. The whole framework is illustrated in Fig. 2.

Step 1: Low-rank decomposition method is implemented to diminish the redundancy within filter. As a result, the decomposed network consists of smaller filters, which improves the flexibility of the next channel pruning. By reducing the parameters in the shallow layers, we obtain impressive acceleration benefits in this step.

Step 2: We implement channel pruning on the decomposed network in the first step. In the previous analysis, the smaller filters will help channel pruning generate a more efficient network with given decay constraint. This step is employed to obtain global compression benefits as well as to eliminate the redundancy left in deep layers.

The computation reduction is more efficient in shallow layers. For a convolutional layer, the number of parameters and computation are calculated as follows:

$$\text{Param} = (C_{in}hw + 1)C_{out}, \tag{4}$$

$$\text{FLOPs} = 2HW(C_{in}hw + 1)C_{out}, \tag{5}$$

where H, W and C_{out} are the height, width and the number of channels of output feature maps, C_{in} is the number of channels of input feature maps, and h and w are height and width of the filter. As the resolution of feature maps generally decreases with the layer deepens, the computation is more sensitive to the parameters in the lower layers.

Previous works pointed out that channel pruning is likely to prune more filters in deeper layers, because shallow layers have relatively higher importance under global pruning metric [22]. Compared with deep layers, channel pruning cannot accomplish efficient speedup on shallow layers. For higher speedup ratio, we restrict the retained ranks in shallow layers to obtain satisfying acceleration benefits in shallow layers. And we retain moderate redundancy in deep layers, which is required for performance recovery. On the other hand, the left redundancy in deep layers is easily diminished in further channel pruning. In this way, the benefits induced by low-rank decomposition are maximized.

C. A Better Indicator for Rank Selection

In low-rank approximation, the retained rank determines the efficiency and the performance decay. However, it is impossible to evaluate the performance of network with every possible rank. Thus an indicator is needed for rank selection. When the indicator approaches a certain value, the corresponding rank results in a nice trade-off between efficiency and network performance. [25] proposed a Frobenius norm-based indicator for low-rank approximation. They computed the error between the approximated weights and original ones by normalized Frobenius norm. In experiment, we find that the spectral norm is a better metric. Denote that the weight matrix $W \in \mathbb{R}^{m\times n}(m > n)$ with singular values as $\sigma_1 \geq \cdots \geq \sigma_n \geq 0$. When the approximated matrix $\tilde{W}_k$ contains the largest-k singular values, the spectral norm of the approximation error is computed as follows:

$$\parallel W - \tilde{W}_k \parallel_{\text{spec}} = \sigma_{k+1}, \tag{6}$$

where $0 \leq k < n$. Particularly, the spectral norm of the original model is $\parallel W \parallel_{\text{spec}} = \sigma_1$. The indicator r is defined as the relative magnitude of the largest abandoned component.

$$r = \frac{\parallel W - \tilde{W}_k \parallel_{\text{spec}}}{\parallel W \parallel_{\text{spec}}} = \frac{\sigma_{k+1}}{\sigma_1} \tag{7}$$

We find that when the spectral norm ratio r is between 0.20 and 0.25, the network reaches a nice trade-off between performance and efficiency, as shown in Table I. At the same time, the Frobenius norm ratio changes dramatically from 0.02 to 0.23 in different layers. In our analysis, Frobenius norm describes the cumulative energy of singular values, and numerous small singular values may interfere with rank selection. In fact, these components with small magnitudes provide negligible contribution to the network, and some of them can be regarded as noise. But the spectral norm ratio r has non-maximum suppression over the noisy components. Once the component with the largest singular value is unimportant to the network, the other components are also negligible.

IV. Experiments

In this section, experiments are performed to show that our framework can effectively boost the performance of channel pruning, and we also present latency to further show our advantages on hardware deployment. We implement our experiments with a VGG-like network [26] on CIFAR-10 and AlexNet [4] on ImageNet. Widely-used low-rank decomposition method [10] and Taylor pruning method [22] are adopted in our framework.

TABLE I
SPECTRAL NORM INDICATOR VS. FROBENIUS NORM INDICATOR.

AlexNet	conv1	conv2	conv3	conv4	conv5
Rank	15	70	135	130	65
Acc. decay	-1.4%	-1.8%	-1.8%	-1.7%	-2.1%
F-norm ratio [25]	0.02	0.08	0.13	0.20	0.23
Spec-norm ratio	0.25	0.22	0.20	0.20	0.20

VGG-16	conv1_2	conv2_1	conv3_2	conv4_2	conv4_3
Rank	21	35	128	192	170
Acc. decay	-0.3%	-0.5%	-0.6%	-0.6%	-1.0%
F-norm ratio [25]	0.05	0.07	0.11	0.16	0.21
Spec-norm ratio	0.25	0.23	0.24	0.25	0.20

TABLE II
EXPERIMENT ON CIFAR-10 NETWORK.

	Original	Low-rank [10]	Prune [22]	Our method
Accuracy	93.16%	91.83%	92.03%	92.25%
Param($\times10^6$)/Ratio	9.29 / 1×	1.39 / 6.69×	1.14 / 8.15×	0.26 / 34.89×
FLOPs($\times10^8$)/Ratio	12.5 / 1×	2.01 / 6.25×	2.29 / 5.47×	0.75 / 16.65×
Latency($\times10^8$)/Ratio	34.6 / 1×	4.21 / 8.22×	1.91 / 18.12×	0.72 / 48.73×

TABLE III
EXPERIMENT ON ALEXNET.

	Original	Low-rank [10]	Prune [22]	Our method
Accuracy	78.64%	76.78%	75.59%	77.18%
Param($\times10^6$)/Ratio	2.47 / 1×	1.13 / 2.18×	1.53 / 1.61×	0.51 / 4.84×
FLOPs($\times10^8$)/Ratio	13.1 / 1×	5.71 / 2.30×	8.65 / 1.51×	3.08 / 4.25×
Latency($\times10^9$)/Ratio	3.54 / 1×	1.43 / 2.48×	2.01 / 1.76×	0.43 / 8.23×

A. Experiment on Acceleration Framework

We implement low-rank decomposition, channel pruning and our combined framework on a CIFAR-10 network and AlexNet respectively. Our experiments are consistent with training setting in [22]. We demonstrate the acceleration and compression performance on convolutional layers. In Table II, our method achieves 16.65× reduction on computation and 34.89× compression simultaneously with 0.91% accuracy decay. In the ImageNet based experiments, our method also outperforms either low-rank decomposition or channel pruning on both computation and model size, as shown in Table III. The experiments confirm that our framework effectively improves the performance of channel pruning.

B. Experiment for Extremely Resource-constrained Scenarios

The ADaPT method [27] is proposed for extremely efficient and tiny CNNs under IoT scenarios. We conduct our method and generate models with similar model size reported in the paper. Without the help of low-rank decomposition, Taylor pruning performs worse than ADaPT. But our model outperforms ADaPT with 1.44 percent rise in network performance with fewer parameters, as shown in Table IV.

TABLE IV
COMPARISON IN EXTREMELY RESOURCE-CONSTRAINED SCENARIOS.

	Prune [22]	ADaPT [27]	Our method
Accuracy	84.33%	86.00%	**87.44%**
Param Num	4.1×10^4	3.9×10^4	$\mathbf{3.7\times10^4}$

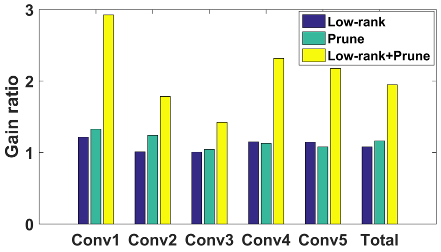

Fig. 3. The gain ratio is computed as latency ratio / FLOPs ratio in AlexNet. It represents the efficiency of theoretical acceleration into practical speedup. This shows that our method (yellow) is more suitable for hardware deployment.

C. Experiment on Rank Selection Methods

We compare the spectral norm-based indicator with Frobenius norm-based one. We implement low-rank decomposition on each single layer in AlexNet and VGG-16, and compute the Frobenius norm-based indicator and spectral norm-based indicator. Here we record the Top-5 accuracy decay, which is more robust for analysis. Considering VGG-16 has more layers, we limit the accuracy decay more strictly. Detailed information is shown in Table I. The variation range of Frobenius norm-based indicator spans orders of magnitude from 0.02 to 0.23, while our proposed indicator is between 0.20 and 0.25 on different layers and different networks.

D. Hardware Latency Evaluation

We deploy networks on Xilinx Zynq ZCU102 platform. The operations, including convolution operations and activation functions, are written in C++ and compiled by Vivado HLS. We keep clock frequency at 100.0 MHz. In Table II and Table III, our framework induces dramatic improvement on hardware latency, and also obtains the best latency reduction per FLOPs as shown in Fig. 3. In our analysis, our method generates more efficient filters with better flexibility, and results best performance in parallel computing. The consistency between FLOPs and practical latency is crucial to CNNs deployment on IoT devices.

V. CONCLUSION

In this work, we propose a two-step framework to boost channel pruning with the help of low-rank decomposition. In our framework, convolutional layers are first decomposed by low-rank approximation, which convert the network into smaller filters. Then we implement channel pruning to further compress and accelerate the low-rank network. In addition, we propose a spectral norm-based indicator to balance these two steps. Experiments show that our framework not only provides obvious improvement on both acceleration and compression theoretically, but also is more suitable for hardware deployment with lower latency.

REFERENCES

[1] L. Da Xu, W. He, and S. Li, “Internet of things in industries: A survey,” *IEEE Transactions on industrial informatics*, vol. 10, no. 4, pp. 2233–2243, 2014.

[2] S. R. Islam, D. Kwak, M. H. Kabir, M. Hossain, and K.-S. Kwak, “The internet of things for health care: a comprehensive survey,” *IEEE Access*, vol. 3, pp. 678–708, 2015.

[3] G. Kortuem, F. Kawsar, V. Sundramoorthy, and D. Fitton, “Smart objects as building blocks for the internet of things,” *IEEE Internet Computing*, vol. 14, no. 1, pp. 44–51, 2010.

[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[5] S. Ren, K. He, R. Girshick, and J. Sun, “Faster r-cnn: Towards real-time object detection with region proposal networks,” in *Advances in neural information processing systems*, 2015, pp. 91–99.

[6] H. Li, K. Ota, and M. Dong, “Learning iot in edge: deep learning for the internet of things with edge computing,” *IEEE Network*, vol. 32, no. 1, pp. 96–101, 2018.

[7] E. L. Denton, W. Zaremba, J. Bruna, Y. LeCun, and R. Fergus, “Exploiting linear structure within convolutional networks for efficient evaluation,” in *Advances in neural information processing systems*, 2014, pp. 1269–1277.

[8] V. Lebedev, Y. Ganin, M. Rakhuba, I. Oseledets, and V. Lempitsky, “Speeding-up convolutional neural networks using fine-tuned cp-decomposition,” *arXiv preprint arXiv:1412.6553*, 2014.

[9] Y.-D. Kim, E. Park, S. Yoo, T. Choi, L. Yang, and D. Shin, “Compression of deep convolutional neural networks for fast and low power mobile applications,” *arXiv preprint arXiv:1511.06530*, 2015.

[10] C. Tai, T. Xiao, Y. Zhang, X. Wang *et al.*, “Convolutional neural networks with low-rank regularization,” *arXiv preprint arXiv:1511.06067*, 2015.

[11] S. Anwar and W. Sung, “Coarse pruning of convolutional neural networks with random masks,” 2016.

[12] S. Anwar, K. Hwang, and W. Sung, “Structured pruning of deep convolutional neural networks,” *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, vol. 13, no. 3, p. 32, 2017.

[13] S. Han, H. Mao, and W. J. Dally, “Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding,” *arXiv preprint arXiv:1510.00149*, 2015.

[14] S. Han, X. Liu, H. Mao, J. Pu, A. Pedram, M. A. Horowitz, and W. J. Dally, “Eie: efficient inference engine on compressed deep neural network,” in *Computer Architecture (ISCA), 2016 ACM/IEEE 43rd Annual International Symposium on*. IEEE, 2016, pp. 243–254.

[15] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Q. Weinberger, “Deep networks with stochastic depth,” in *European Conference on Computer Vision*. Springer, 2016, pp. 646–661.

[16] Y. He, X. Zhang, and J. Sun, “Channel pruning for accelerating very deep neural networks,” in *International Conference on Computer Vision (ICCV)*, vol. 2, 2017, p. 6.

[17] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, “Pruning filters for efficient convnets,” *arXiv preprint arXiv:1608.08710*, 2016.

[18] B. Han, Z. Zhang, C. Xu, B. Wang, G. Hu, L. Bai, Q. Hong, and E. R. Hancock, “Deep face model compression using entropy-based filter selection,” in *International Conference on Image Analysis and Processing*. Springer, 2017, pp. 127–136.

[19] H. Hu, R. Peng, Y.-W. Tai, and C.-K. Tang, “Network trimming: A data-driven neuron pruning approach towards efficient deep architectures,” *arXiv preprint arXiv:1607.03250*, 2016.

[20] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, and C. Zhang, “Learning efficient convolutional networks through network slimming,” in *2017 IEEE International Conference on Computer Vision (ICCV)*. IEEE, 2017, pp. 2755–2763.

[21] A. Ashok, N. Rhinehart, F. Beainy, and K. M. Kitani, “N2n learning: Network to network compression via policy gradient reinforcement learning,” *arXiv preprint arXiv:1709.06030*, 2017.

[22] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz, “Pruning convolutional neural networks for resource efficient inference,” 2016.

[23] E. T. Jaynes, “Information theory and statistical mechanics,” *Physical review*, vol. 106, no. 4, p. 620, 1957.

[24] Z. Chen, J. Lin, S. Liu, J. Xia, and W. Li, “Decouple and stretch: a boost to channel pruning,” in *International Performance Computing and Communications Conference*. IEEE, 2018.

[25] K. Osawa and R. Yokota, “Evaluating the compression efficiency of the filters in convolutional neural networks,” in *International Conference on Artificial Neural Networks*. Springer, 2017, pp. 459–466.

[26] X. Chen, “Pytorch-playground,” https://github.com/aaron-xichen/pytorch-playground, last accessed: 2019/01/31.

[27] P. Maji, D. Bates, A. Chadwick, and R. Mullins, “Adapt: optimizing cnn inference on iot and mobile devices using approximately separable 1-d kernels,” in *Proceedings of the 1st International Conference on Internet of Things and Machine Learning*. ACM, 2017, p. 43.