LESS: LEARNING TO SELECT A STRUCTURED ARCHITECTURE OVER FILTER PRUNING AND LOW-RANK DECOMPOSITION

Moonjung Eo
LG AI Research, Seoul
moonj@lgresearch.ai

Suhyun Kang
Department of Intelligence and Information
Seoul National University, Seoul
su_hyun@snu.ac.kr

Wonjong Rhee*
Department of Intelligence and Information, IPAI, and AIIS
Seoul National University, Seoul
wrhee@snu.ac.kr

ABSTRACT

Designing a deep neural network (DNN) for efficient operation in low-resource environments necessitates strategic application of compression techniques. Filter pruning and low-rank decomposition stand out as two prominent methods employed for DNN compression. While these techniques possess complementary properties, their integration has been only partially explored, resulting in limited reported gains thus far. In this study, we present a novel fully joint learning algorithm named **LeSS**, aiming to concurrently determine filters for filter pruning and ranks for low-rank decomposition. Unlike previous methods, LeSS simultaneously determines both filters and ranks, eliminating the need for iterative or heuristic processes. Notably, LeSS adheres strictly to the specified resource budget constraint, ensuring practical applicability in resource-constrained scenarios. LeSS outperforms state-of-the-art methods on a number of benchmarks demonstrating its effectiveness.

1 INTRODUCTION

Deep neural networks (DNNs) have achieved state-of-the-art performance in various fields, such as image classification, object detection, and video understanding. However, millions of parameters and high computational costs make their deployment on low-resource settings such as edge and mobile devices challenging. To overcome this problem, DNN compression has been extensively studied in recent years. Among various compression techniques, filter pruning and low-rank decomposition are two representative approaches, both of which do not require hardware modification. They aim to reduce a heavy network to a lightweight form with two different structural viewpoints.

Consider a weight matrix with n filters $\mathbf{W} = \{\mathbf{w_1}, \cdots, \mathbf{w_n}\} \in \mathbb{R}^{m \times n}$. Filter pruning removes uninformative column vectors (i.e., filters) from $\mathbf{W}$ at low compression rates but discards valuable information as rates increase. In contrast, low-rank compression reduces the column space dimension of $\mathbf{W}$ (i.e., rank), resulting in $\mathbf{W}$ with reduced rank. Given the distinct structural perspectives of the two approaches, there is potential to combine them effectively, offering a promising compression solution.

Recent studies (Ruan et al., 2020; Guo et al., 2019; Li et al., 2020) have rigorously examined the simultaneous optimization of filter pruning and low-rank compression, employing common criteria to unveil the most efficient lightweight structure for DNNs. These methodologies incorporate formalized regularized training with l_1 regularization (Ruan et al., 2020), strategically applied to both

*Correspondence to : wrhee@snu.ac.kr
Full version of this paper is available at https://arxiv.org/abs/2309.13077

the column and row vectors of the weight matrix $\mathbf{W}$. Despite these formalizations, the pivotal challenge persists in precisely determining effective filters and optimal ranks for the ultimate compressed model, predominantly due to the reliance on heuristic search algorithms.

To address these challenges, we propose a learning method called **Le**arning to **S**elect a **S**tructured architecture (**LeSS**) that jointly learns a mask and a threshold to select informative filters and an optimal rank within a desired resource budget constraint. While we adopt binary mask learning to select informative filters in the filter pruning field (He et al., 2018a; Huang & Wang, 2018; Kang & Han, 2020; Luo & Wu, 2020; Wang et al., 2020), we devise threshold learning to select optimal ranks. To the best of our knowledge, we are the first to develop a learning method for selecting optimal ranks. Through learned binary masks and thresholds, we can select informative filters and optimal ranks without additional heuristic search algorithms. Our work significantly contributes to devising an end-to-end learning process without heuristic search algorithms, resulting in enhanced model efficiency.

2 Related works

Previous research (Dubey et al., 2018; Chen et al., 2020) has proposed separate compression stages to integrate multiple compression techniques. These stages sequentially adopt one compression technique in each step and ignore the interrelations of the different compression methods. For instance, in Dubey et al. (2018), filter pruning is conducted first to reduce the weights, and the weights are then decomposed using a corset-based decomposition technique. In addition, several compression-aware training approaches are proposed using regularization to make a network compression-friendly (Chen et al., 2019; Ruan et al., 2020; Guo et al., 2019; Li et al., 2020). For example, in Li et al. (2020), they first introduce a sparsity-inducing matrix at each weight and then impose group sparsity constraints during training. However, determining a good balance between compression rate and accuracy is challenging under the desired compression rate with these compression-aware methods. Recently, Li et al. (2022) proposes a collaborative compression method to employ the global compression rate optimization method to obtain the compression rate of each layer and adopt a multi-step heuristic removal strategy. Our hybrid compression method does not need heuristics on selecting filters and ranks and does not require compression-aware regularized training.

3 Background

3.1 Tensor Matricization

In our work, *matricization* is used to transform the tensor of convolutional kernels into a matrix to conduct singular value decomposition (SVD) operation. *Matricization* is the process of reshaping the elements of an D-dimensional tensor $\boldsymbol{\mathsf{X}} \in \mathbb{R}^{I_1 \times \cdots \times I_D}$ into a matrix (Kolda & Bader, 2009; Kolda, 2006). Let the ordered sets $\mathcal{R} = \{r_1, ..., r_L\}$ and $\mathcal{C} = \{c_1, ..., c_M\}$ be a partitioning of the modes $\mathcal{D} = \{1, ..., D\}$. The matricization function ψ of an D-dimensional tensor $\boldsymbol{\mathsf{X}} \in \mathbb{R}^{I_1 \times \cdots \times I_D}$ is defined as:

$$\psi : \boldsymbol{\mathsf{X}} \longmapsto \mathbf{X}_{(\mathcal{R}\times\mathcal{C}:I_D)} \in \mathbb{R}^{J \times K}, \\ \text{where } J = \prod_{n \in \mathcal{R}} I_n \quad \text{and} \quad K = \prod_{n \in \mathcal{C}} I_n. \tag{1}$$

For example, the weight tensor of a convolutional layer is represented as a 4-D tensor ($\boldsymbol{\mathsf{W}} \in \mathbb{R}^{C_{out} \times C_{in} \times k \times k}$) where it is composed of kernels, and it can be unfolded into a matrix as six different forms. The two most common forms used in low-rank decomposition are as follows: ① $\mathbf{W} \in \mathbb{R}^{C_{out} \times (C_{in}kk)}$, ② $\mathbf{W} \in \mathbb{R}^{(C_{out}k) \times (C_{in}k)}$.

3.2 CNN decomposition scheme

To decompose a convolutional layer with C_{in}, C_{out} (input/output channels) and k (kernel size), one of the following low-rank structures is used depending on the matricization form.

Scheme 1 When we use the first reshaping form ① introduced in Section 3.1, the convolutional weights can be considered as a linear layer with the shape of $C_{out} \times C_{in}k^2$. Then, the rank-r approximation presents two linear mappings with weight shapes $C_{out} \times r$ and $r \times C_{in}k^2$. These linear mappings can be deployed as a sequence of two convolutional layers: $\boldsymbol{\mathsf{W}}_1 \in \mathbb{R}^{r \times C_{in} \times k \times k}$, and $\boldsymbol{\mathsf{W}}_2 \in \mathbb{R}^{C_{out} \times r \times 1 \times 1}$ (Wen et al., 2017; Xu et al., 2020; Li & Shi, 2018).

Scheme 2 When we use the second reshaping form ② introduced in Section 3.1, the convolutional weights can be considered as a linear layer of $C_{out}k \times C_{in}k$. For this scheme, an approximation of rank r has two linear mappings with weight shapes $C_{out}k \times r$ and $r \times C_{in}k$. These can be

implemented as a sequence of two convolutional layers as follows: $\mathbf{W}_1 \in \mathbb{R}^{r\times C_{in}\times k\times 1}$, and $\mathbf{W}_2 \in \mathbb{R}^{C_{out}\times r\times 1\times k}$ (Tai et al., 2015; Jaderberg et al., 2014).
When r of a layer is large and its matrix decomposition results in an increase of FLOPs, matrix decomposition is not applied to the layer. This has been a common convention in low-rank research (Idelbayev & Carreira-Perpinán, 2020; Phan et al., 2020; Xu et al., 2020). We use ***Scheme 1*** throughout all experiments in our study.

4 LEARNING TO SELECT A STRUCTURED ARCHITECTURE

In this section, we propose a new learning method called LeSS, which learns binary masks and thresholds to select informative filters and an optimal rank within a desired resource budget constraint. The general idea of LeSS is to transform the problem of solving over discrete variables $\mathbf{c}$ and $\mathbf{r}$ into minimizing a differentiable surrogate function h_{LeSS} over continuous variables $\mathbf{z_c}$ and $\mathbf{z_r}$. The discrete solution can be approximated using differentiable functions $g_\mathbf{c}$ and $g_\mathbf{r}$, and this allows us to use gradient-based optimization that is not available in discrete problems. More specifically, we re-define the problem as follows:

$$\begin{aligned} \min_{\mathbf{z_c},\mathbf{z_r}} \frac{1}{N}\sum_{i=1}^{N} &\mathcal{L}(f(x_i; h_{\text{LeSS}}(\mathbf{W}, \mathbf{z_c}, \mathbf{z_r}))), y_i) \\ &+\lambda \left\|\mathcal{B}(g_\mathbf{c}(\mathbf{z_c}), g_\mathbf{r}(\mathbf{z_r})) - \mathcal{B}_d\right\|^2 \end{aligned} \tag{2}$$

where $g_\mathbf{c}(\mathbf{z_c})$ is the number of filters for each channel, $g_\mathbf{r}(\mathbf{z_r})$ is the rank for each layer, and λ is a hyper-parameter used to regularize the budget constraint. For each layer, LeSS consists of the surrogate function h_{LeSS} which is composed of two modules, s_m and s_t.

Module s_m (mask learning for filter selection): To construct a function h_{LeSS}, we describe a module s_m used for filter selection. Let $\mathbf{z_c} = \{M_1, \cdots, M_L \mid M_l \in \mathbb{R}^{C^l_{out}\times(C^l_{in}kk)}\}$ be a set of diagonal matrices (i.e., mask matrix). To establish s_m, we first define a scheduled sigmoid function to generate the approximate binary masks as follows:

$$\phi_s(x) = \frac{1}{1+\exp(-1*\mu_i*(x-0.5))} \tag{3}$$

where $\mu_i = \min(\alpha, \mu_{i-1}+\beta)$.
Note that μ_i is the scheduling factor affecting the steepness of sigmoid in iteration i, and it is updated every iteration and does not exceed α. During the beginning phases of training, μ_i is kept at a very low value; it is then increased as the optimization process progresses. When μ_i grows large enough, the values of approximate binary masks will become almost 0 or 1. That is, it is completely determined which filter should be removed. For μ_i, its α is simply set as a large constant of 50 because LeSS's performance is not sensitive to the choice, and its β is explored using a light grid search. For each weight $\mathbf{W}_l$ of the l-th layer, we define a function s_m by Eq. (1) and Eq. (3) as follows:

$$s_m(\mathbf{W}_l, M_l) = \psi^{-1}(\phi_s(M_l)\cdot\psi(\mathbf{W}_l)) \tag{4}$$

To approximate the number of filters corresponding to the continuous variable $\mathbf{z_c}$, we define the set-valued function $g_\mathbf{c}$ as follows:

$$g_\mathbf{c}(\mathbf{z_c}) = \{\mathbf{1}^T\cdot\text{diag}(\phi_s(M_l))\}_{l=1}^{L} \tag{5}$$

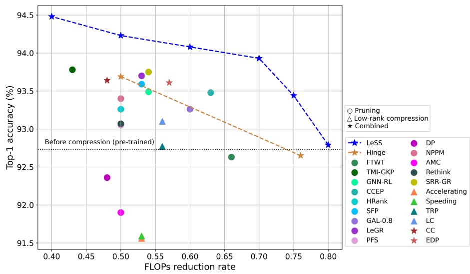
Since all functions constituting Eq. (4) and Eq. (5) are differentiable, we can easily confirm that s_m and $g_\mathbf{c}$ are differentiable.

Module s_t (threshold learning for rank selection): To proceed, we explain the s_t module used for rank selection. Let $\mathbf{z_r} = \{\gamma_1, \cdots, \gamma_L \mid \gamma_l \in \mathbb{R}\}$ be a threshold set. To construct s_t, we introduce a Singular Value Thresholding (SVT) function (Cai et al., 2010). For a matrix $M \in \mathbb{R}^{m\times n}$ and threshold $\gamma \in \mathbb{R}$, SVT is defined as follows:

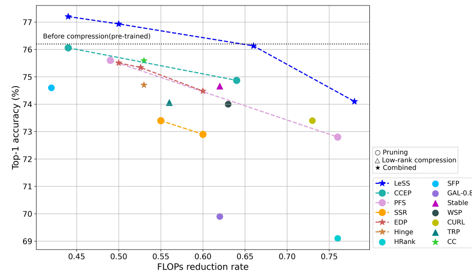
$$\text{SVT}(M,\gamma) = U\cdot\text{ReLU}(\Sigma-\gamma)\cdot V^T \tag{6}$$

where U is an $m\times m$ real unitary matrix, Σ is an $m\times n$ rectangular diagonal matrix with non-negative real numbers on the diagonal, V is an $n\times n$ real unitary matrix, and $\text{ReLU}(\cdot)$ is a rectified linear unit activation function. For each weight W_l of the l-th layer, we define a function s_t by (1) and (6) as follows:

$$s_t(\mathbf{W}_l, \gamma_l) = \psi^{-1}(\text{SVT}(\psi(\mathbf{W}_l), \gamma_l)) \tag{7}$$



(a) ResNet56 on CIFAR10



(b) ResNet50 on ImageNet

Figure 1: Comparison of our method with SOTA pruning, low-rank decomposition, and hybrid compression methods for (a) ResNet56 on CIFAR10 and (b) ResNet50 on ImageNet.

To approximate the rank corresponding to $\mathbf{z_r}$, we also define the set-valued function $g_{\mathbf{r}}$ as:

$$g_{\mathbf{r}}(\mathbf{z_r}) = \{\mathbf{1}^T \cdot (\tanh(\mathrm{ReLU}(\Sigma_l - \gamma_l) \cdot \tau))\}_{l=1}^{L} \tag{8}$$

where Σ_l is a diagonal matrix whose diagonal entries are singular values of the l-th layer weight matrix W_l and τ is a scaling hyper-parameter used to control the steepness of $\tanh$. Similar to s_m and $g_{\mathbf{c}}$, we can easily confirm that s_t and $g_{\mathbf{r}}$ are differentiable.

Budget $\mathcal{B}(g_{\mathbf{c}}(\mathbf{z_c}), g_{\mathbf{r}}(\mathbf{z_r}))$: FLOP is used for the resource budget in this paper and the formula of budget calculation is as follows:

$$\sum_{l=1}^{L} \frac{A_l \cdot k_l \cdot k_l \cdot g_{\mathbf{r}}(l)(\mathbf{z_r}) \cdot (g_{\mathbf{c}}(\mathbf{z_c})(l-1) + g_{\mathbf{c}}(\mathbf{z_c})(l))}{A_l \cdot k_l \cdot k_l \cdot C_{in}^l \cdot C_{out}^l} \tag{9}$$

A_l denotes the area of the l-th layer's feature maps, and k_l is the kernel size of the l-th layer. C_{in}^l and C_{out}^l denote l-th layer's input- and output-channels of the original model, respectively. $g_{\mathbf{r}}(\mathbf{z_r})(l)$ and $g_{\mathbf{c}}(\mathbf{z_c})(l)$ are the l-th layer's selected rank and number of selected filters, respectively. Because all elements in Eq. (9) are differentiable, the budget function $\mathcal{B}(g_{\mathbf{c}}(\mathbf{z_c}), g_{\mathbf{r}}(\mathbf{z_r}))$ is differentiable. We use FLOPs resource budget, but other resource budgets (e.g., number of parameters) also can be used.
Finally, we define the surrogate function, h_{LeSS}:

$$h_{\mathrm{LeSS}}(\mathbf{W}_l, M_l, \gamma_l) = s_t(s_m(\mathbf{W}_l, M_l), \gamma_l) \tag{10}$$

The two parameters $\mathbf{z_c}$ and $\mathbf{z_r}$ of LeSS are learned simultaneously in the training process. Therefore, LeSS can efficiently employ two distinct compression techniques by simultaneously considering the impact of reducing the number of filters and ranks on the model's performance.

Select the informative filters and rank: After training is completed, we require exact binary masks and ranks for each layer to directly compress the model. To acquire the informative filters and optimal ranks, we can select *Binary Mask Set* $\lceil \mathbf{z_c} \rceil = \{\lceil M_1 \rceil, \cdots, \lceil M_L \rceil\}$ and *Rank Set* $g_{\mathbf{r}}(\mathbf{z_r})$ without additional heuristic algorithms (e.g., Binary search). In the l-th layer weight, we prune the filter whose exact binary mask is zero. Subsequently, we perform low-rank decomposition with the exact rank on the pruned weight. Finally, as in previous studies (Alwani et al., 2022; Cai et al., 2021; Idelbayev & Carreira-Perpinán, 2020), we fine-tune the compressed model to improve performance further.

5 Experiments

We provide graphical summaries for the two cases (ResNet56 on CIFAR10 and ResNet50 on ImageNet) where a sufficiently large number of comparisons exist and provide table summaries where fewer comparison points are available.

ResNet56 on CIFAR10 Figure 1a shows the comparison results for ResNet56 on CIFAR10. LeSS outperforms the previous methods by a large margin across all FLOP reduction rates. In particular, the 50% FLOP reduction rate is investigated by a bunch of previous methods, and LeSS achieves the best performance under this constraint. Note that the compressed model produced by LeSS

Dataset	Model	Compression method	Algorithm	Baseline (%)	Test acc.(%)	Δ Test acc.(%)	GFLOPs (Reduction ratio)	Params (Compression ratio)
ImageNet	ResNet18	Low-rank	Stable (Phan et al., 2020)	69.76	68.62	- 1.14	1.00 (45 %)	N/A
			TRP (Xu et al., 2020)	69.10	65.51	- 3.59	0.73 (60 %)	N/A
			ALDS (Liebenwein et al., 2021)	69.62	69.24	- 0.38	0.64 (65 %)	N/A
		Pruning	SFP (He et al., 2018a)	70.28	67.10	- 3.18	1.06 (42 %)	N/A
			FPGM (He et al., 2019)	70.28	68.41	- 1.87	1.06 (42 %)	7.10 M (39 %)
			PFP (Liebenwein et al., 2019)	69.74	65.65	- 4.09	1.04 (43 %)	N/A
			DMCP (Guo et al., 2020)	N/A	69.00	N/A	1.04 (43 %)	N/A
			CHEX (Hou et al., 2022)	N/A	69.60	N/A	1.03 (43 %)	N/A
			SCOP (Tang et al., 2020)	69.76	68.62	- 1.14	1.00 (45 %)	N/A
			FBS (Gao et al., 2018)	69.76	68.17	- 1.59	0.91 (50 %)	N/A
			CGNET (Hua et al., 2019)	69.76	68.30	- 1.46	0.89 (51 %)	N/A
			GNN (Yu et al., 2022)	69.76	68.66	-1.10	0.89 (51 %)	N/A
			ManiDP (Tang et al., 2021)	69.76	68.88	- 0.88	0.89 (51 %)	N/A
			PGMPF (Cai et al., 2022)	70.23	66.67	- 3.56	0.84 (54 %)	N/A
		Hybrid	**LeSS**	69.76	**71.24**	+ 1.48	0.91 (50 %)	4.68 M (60 %)
			LeSS	69.76	**70.82**	+ 1.06	0.70 (62 %)	3.51 M (70 %)
			LeSS	69.76	**70.15**	+ 0.39	0.55 (70 %)	2.81 M (76 %)
	MobileNetV2	Low-rank	LC (Idelbayev & Carreira-Perpinán, 2020)	71.80	69.80	- 2.00	0.21 (30 %)	N/A
		Pruning	PFS (Wang et al., 2020)	71.80	70.90	- 0.90	0.21 (30 %)	2.60 M (26 %)
			AMC (He et al., 2018b)	71.80	70.80	- 1.00	0.22 (27 %)	2.30 M (34 %)
			MetaPruning (Liu et al., 2019)	72.00	71.20	- 0.80	0.22 (27 %)	N/A
			LeGR (Chin et al., 2020)	71.80	71.40	- 0.20	0.21 (30 %)	N/A
			NPPM (Gao et al., 2021)	72.02	72.04	+ 0.02	0.21 (30 %)	N/A
			GNN (Yu et al., 2022)	71.87	70.04	- 1.83	0.17 (42 %)	N/A
		Hybrid	EDP (Ruan et al., 2020)	N/A	71.00	N/A	0.22 (27 %)	N/A
			LeSS	71.80	**72.16**	+ 0.20	0.19 (35 %)	2.24 M (36 %)
			LeSS	71.80	**71.63**	- 0.17	0.14 (55 %)	1.54 M (56 %)

Table 1: Performance comparison for ResNet18 and MobileNetV2 on ImageNet.

consistently exhibits higher performance than that of the original (baseline) model across all FLOP reduction rates. LeSS reduces the FLOPs by 40% compared with the baseline model yet improves accuracy by 1.6%. This demonstrates that our compression method correctly eliminates redundant dimensions and filters, resulting in a generalizable compressed model.

ResNet50 and ResNet18 on ImageNet The result of ResNet50 on ImageNet can be founded in Figure 1b. The graphical summary confirms that **LeSS** shows superior performance than that of the other SOTA methods in all FLOP reduction rates. For instance, when we compare the difference in the FLOP rate between our method and the CC algorithm (Li et al., 2021) at the same performance (75.59%), our method can accelerate the inference time by 14% more than the CC method (Li et al., 2021) (0.53 *vs.* 0.68). In addition, even when ResNet50 is compressed by 50% FLOP reduction, our method exhibits higher performance than the baseline performance. The result of ResNet18 on ImageNet is summarized in Table 1. Because no experimental results of hybrid algorithms for ResNet18 on ImageNet are available, the performances of algorithms employing only a single compression method are compared. When compared with recent SOTA methods, **LeSS** outperforms them in all FLOP reduction rates, and even when a model is compressed up to 70%, the performance is higher than the baseline performance. That is, **LeSS** removes redundant dimensions and filters effectively.

MobileNetV2 on ImageNet Performance comparison result for ImageNet on light-weight MobileNetV2 is summarized in Table 1. MobileNetV2 is a well-known computationally efficient model, which makes it harder to compress. Nevertheless, our method surprisingly increases the model's top-1 accuracy up to 72% when the FLOP reduction rate is 35%. Furthermore, despite the fact that the inference time is accelerated more than twice that of the original model, the performance reduction is only 0.17 percentage points. From these results, it is concluded that our method can efficiently reduce the size of a network while keeping performance as high as possible, even if the model size is already small.

6 Conclusion

We introduce a novel fully joint learning algorithm, LeSS, designed to concurrently determine filters for pruning and ranks for low-rank decomposition. Integrating differentiable mask learning for filter pruning and differentiable threshold learning for low-rank decomposition, LeSS achieves joint optimization while adhering to specified resource constraints. Unlike previous methods, LeSS seamlessly determines both filters and ranks, eliminating the need for iterative or heuristic processes. Notably, LeSS strictly adheres to the specified resource budget constraint, ensuring practical applicability in resource-constrained scenarios. The superior performance of LeSS across various benchmarks underscores its effectiveness, positioning it as a promising advancement in the realm of DNN compression for low-resource environments.

Acknowledgements

This work was supported by NRF (NRF-2020R1A2C2007139), IITP [NO.2021-0-01343, Artificial Intelligence Graduate School Program (Seoul National University)], and the Korea Medical Device Development Fund grant funded by the Korea government (the Ministry of Science and ICT, the Ministry of Trade, Industry and Energy, the Ministry of Health Welfare, Republic of Korea, the Ministry of Food and Drug Safety) (Project Number: 202013B14).

REFERENCES

Manoj Alwani, Yang Wang, and Vashisht Madhavan. Decore: Deep compression with reinforcement learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12349–12359, 2022.

Jian-Feng Cai, Emmanuel J Candès, and Zuowei Shen. A singular value thresholding algorithm for matrix completion. *SIAM Journal on optimization*, 20(4):1956–1982, 2010.

Linhang Cai, Zhulin An, Chuanguang Yang, and Yongjun Xu. Soft and hard filter pruning via dimension reduction. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8. IEEE, 2021.

Linhang Cai, Zhulin An, Chuanguang Yang, Yangchun Yan, and Yongjun Xu. Prior gradient mask guided pruning-aware fine-tuning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 1, 2022.

Zhen Chen, Jianxin Lin, Sen Liu, Zhibo Chen, Weiping Li, Jin Zhao, and Wei Yan. Exploiting weight-level sparsity in channel pruning with low-rank approximation. In *2019 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–5. IEEE, 2019.

Zhen Chen, Zhibo Chen, Jianxin Lin, Sen Liu, and Weiping Li. Deep neural network acceleration based on low-rank approximated channel pruning. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 67(4):1232–1244, 2020.

Ting-Wu Chin, Ruizhou Ding, Cha Zhang, and Diana Marculescu. Towards efficient model compression via learned global ranking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1518–1528, 2020.

Abhimanyu Dubey, Moitreya Chatterjee, and Narendra Ahuja. Coreset-based neural network compression. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 454–470, 2018.

Shangqian Gao, Feihu Huang, Weidong Cai, and Heng Huang. Network pruning via performance maximization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9270–9280, 2021.

Xitong Gao, Yiren Zhao, Łukasz Dudziak, Robert Mullins, and Cheng-zhong Xu. Dynamic channel pruning: Feature boosting and suppression. *arXiv preprint arXiv:1810.05331*, 2018.

Kailing Guo, Xiaona Xie, Xiangmin Xu, and Xiaofen Xing. Compressing by learning in a low-rank and sparse decomposition form. *IEEE Access*, 7:150823–150832, 2019.

Shaopeng Guo, Yujie Wang, Quanquan Li, and Junjie Yan. Dmcp: Differentiable markov channel pruning for neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 1539–1547, 2020.

Yang He, Guoliang Kang, Xuanyi Dong, Yanwei Fu, and Yi Yang. Soft filter pruning for accelerating deep convolutional neural networks. *arXiv preprint arXiv:1808.06866*, 2018a.

Yang He, Ping Liu, Ziwei Wang, Zhilan Hu, and Yi Yang. Filter pruning via geometric median for deep convolutional neural networks acceleration. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 4340–4349, 2019.

Yihui He, Ji Lin, Zhijian Liu, Hanrui Wang, Li-Jia Li, and Song Han. Amc: Automl for model compression and acceleration on mobile devices. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 784–800, 2018b.

Zejiang Hou, Minghai Qin, Fei Sun, Xiaolong Ma, Kun Yuan, Yi Xu, Yen-Kuang Chen, Rong Jin, Yuan Xie, and Sun-Yuan Kung. Chex: Channel exploration for cnn model compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12287–12298, 2022.

Weizhe Hua, Yuan Zhou, Christopher M De Sa, Zhiru Zhang, and G Edward Suh. Channel gating neural networks. *Advances in Neural Information Processing Systems*, 32, 2019.

Zehao Huang and Naiyan Wang. Data-driven sparse structure selection for deep neural networks. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 304–320, 2018.

Yerlan Idelbayev and Miguel A Carreira-Perpinán. Low-rank compression of neural nets: Learning the rank of each layer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8049–8059, 2020.

Max Jaderberg, Andrea Vedaldi, and Andrew Zisserman. Speeding up convolutional neural networks with low rank expansions. *arXiv preprint arXiv:1405.3866*, 2014.

Minsoo Kang and Bohyung Han. Operation-aware soft channel pruning using differentiable masks. In *International Conference on Machine Learning*, pp. 5122–5131. PMLR, 2020.

Tamara G Kolda and Brett W Bader. Tensor decompositions and applications. *SIAM review*, 51(3): 455–500, 2009.

Tamara Gibson Kolda. Multilinear operators for higher-order decompositions. Technical report, Sandia National Laboratories (SNL), Albuquerque, NM, and Livermore, CA ..., 2006.

Chong Li and CJ Shi. Constrained optimization based low-rank approximation of deep neural networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 732–747, 2018.

Nannan Li, Yu Pan, Yaran Chen, Zixiang Ding, Dongbin Zhao, and Zenglin Xu. Heuristic rank selection with progressively searching tensor ring network. *Complex & Intelligent Systems*, 8(2): 771–785, 2022.

Yawei Li, Shuhang Gu, Christoph Mayer, Luc Van Gool, and Radu Timofte. Group sparsity: The hinge between filter pruning and decomposition for network compression. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 8018–8027, 2020.

Yuchao Li, Shaohui Lin, Jianzhuang Liu, Qixiang Ye, Mengdi Wang, Fei Chao, Fan Yang, Jincheng Ma, Qi Tian, and Rongrong Ji. Towards compact cnns via collaborative compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6438–6447, 2021.

Lucas Liebenwein, Cenk Baykal, Harry Lang, Dan Feldman, and Daniela Rus. Provable filter pruning for efficient neural networks. *arXiv preprint arXiv:1911.07412*, 2019.

Lucas Liebenwein, Alaa Maalouf, Dan Feldman, and Daniela Rus. Compressing neural networks: Towards determining the optimal layer-wise decomposition. *Advances in Neural Information Processing Systems*, 34:5328–5344, 2021.

Zechun Liu, Haoyuan Mu, Xiangyu Zhang, Zichao Guo, Xin Yang, Kwang-Ting Cheng, and Jian Sun. Metapruning: Meta learning for automatic neural network channel pruning. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 3296–3305, 2019.

Jian-Hao Luo and Jianxin Wu. Autopruner: An end-to-end trainable filter pruning method for efficient deep model inference. *Pattern Recognition*, 107:107461, 2020.

Anh-Huy Phan, Konstantin Sobolev, Konstantin Sozykin, Dmitry Ermilov, Julia Gusak, Petr Tichavskỳ, Valeriy Glukhov, Ivan Oseledets, and Andrzej Cichocki. Stable low-rank tensor decomposition for compression of convolutional neural network. In *European Conference on Computer Vision*, pp. 522–539. Springer, 2020.

Xiaofeng Ruan, Yufan Liu, Chunfeng Yuan, Bing Li, Weiming Hu, Yangxi Li, and Stephen Maybank. Edp: An efficient decomposition and pruning scheme for convolutional neural network compression. *IEEE Transactions on Neural Networks and Learning Systems*, 32(10):4499–4513, 2020.

Cheng Tai, Tong Xiao, Yi Zhang, Xiaogang Wang, et al. Convolutional neural networks with low-rank regularization. *arXiv preprint arXiv:1511.06067*, 2015.

Yehui Tang, Yunhe Wang, Yixing Xu, Dacheng Tao, Chunjing Xu, Chao Xu, and Chang Xu. Scop: Scientific control for reliable neural network pruning. *Advances in Neural Information Processing Systems*, 33:10936–10947, 2020.

Yehui Tang, Yunhe Wang, Yixing Xu, Yiping Deng, Chao Xu, Dacheng Tao, and Chang Xu. Manifold regularized dynamic network pruning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5018–5028, 2021.

Yulong Wang, Xiaolu Zhang, Lingxi Xie, Jun Zhou, Hang Su, Bo Zhang, and Xiaolin Hu. Pruning from scratch. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 12273–12280, 2020.

Wei Wen, Cong Xu, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Coordinating filters for faster deep neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 658–666, 2017.

Yuhui Xu, Yuxi Li, Shuai Zhang, Wei Wen, Botao Wang, Yingyong Qi, Yiran Chen, Weiyao Lin, and Hongkai Xiong. Trp: Trained rank pruning for efficient deep neural networks. *arXiv preprint arXiv:2004.14566*, 2020.

Sixing Yu, Arya Mazaheri, and Ali Jannesari. Topology-aware network pruning using multi-stage graph embedding and reinforcement learning. In *International Conference on Machine Learning*, pp. 25656–25667. PMLR, 2022.