Pruning Networks With Cross-Layer Ranking & $k$-Reciprocal Nearest Filters

Mingbao Lin®, Liujuan Cao®, Yuxin Zhang®, Ling Shao®, Fellow, IEEE, Chia-Wen Lin®, Fellow, IEEE, and Rongrong Ji®, Senior Member, IEEE

Abstract—This article focuses on filter-level network pruning. A novel pruning method, termed CLR-RNF, is proposed. We first reveal a “long-tail” pruning problem in magnitude-based weight pruning methods and then propose a computation-aware measurement for individual weight importance, followed by a cross-layer ranking (CLR) of weights to identify and remove the bottom-ranked weights. Consequently, the per-layer sparsity makes up the pruned network structure in our filter pruning. Then, we introduce a recommendation-based filter selection scheme where each filter recommends a group of its closest filters. To pick the preserved filters from these recommended groups, we further devise a $k$-reciprocal nearest filter (RNF) selection scheme where the selected filters fall into the intersection of these recommended groups. Both our pruned network structure and the filter selection are nonlearning processes, which, thus, significantly reduces the pruning complexity and differentiates our method from existing works. We conduct image classification on CIFAR-10 and ImageNet to demonstrate the superiority of our CLR-RNF over the state-of-the-arts. For example, on CIFAR-10, CLR-RNF removes 74.1% FLOPs and 95.0% parameters from VGG16 with even 0.3% accuracy improvements. On ImageNet, it removes 70.2% FLOPs and 64.8% parameters from ResNet-50 with only 1.7% top-five accuracy drops. Our project is available at https://github.com/lmbxmu/CLR-RNF.

Index Terms—Efficient inference, filter pruning, model compression, network structure.

I. INTRODUCTION

THOUGH deep convolutional neural networks (CNNs) are prevailing, it comes at the cost of huge computational burden and large power consumption, which poses a great challenge for real-time deployments on resource-limited devices, such as cell phones and Internet-of-Things (IoT) devices. To address this problem, model compression has become an active research topic, which aims to reduce the model redundancy with a comparable or even better performance in comparison with the full model, such that the compressed model can be easily run on resource-limited devices. General methods for reducing the model size can be roughly categorized into five groups:

1) Low-bit quantization aims to compress a pretrained model by reducing the number of bits used to represent the weight parameters of the pretrained models [1]–[3].

2) Compact networks, such as ShuffleNets [4], [5], MobileNets [6]–[8] and GhostNet [9], directly design parameter-efficient neural network models.

3) Tensor factorization approximates the weight tensor with a series of low-rank matrices, which are then organized in a sum-product form [10], [11].

4) Network pruning removes a certain part of the network. According to the pruning granularity, existing methods include weight pruning [12], [13], block pruning [14], [15], row/column pruning [16], [17], kernel pruning [18], [19], pattern pruning [20], [21], filter pruning [22], [23], and so on.

In this article, we focus on filter pruning for efficient image classification, which has received ever-increasing focus due to the following advantages: 1) the pruned model is structured, which can be well supported by regular hardware and off-the-shelf basic linear algebra subprogram (BLAS) library; 2) the storage usage and computational cost are significantly reduced in online inference; and 3) it can be further combined with other compression methods, such as network quantization, tensor factorization, and weight pruning, to achieve a deeper compression and acceleration. Despite the extensive progress [24]–[28] made in the literature, two essential issues remain as open problems in the filter pruning, i.e., the pruned network structure and the filter importance measurement.

As the first issue, the pruned network structure is related to the per-layer pruning rate. Setting these pruning rates for different layers has been shown to significantly affect

Manuscript received 19 July 2021; revised 28 November 2021; accepted 28 February 2022. Date of publication 16 March 2022; date of current version 30 October 2023. This work was supported in part by the National Science Fund for Distinguished Young Scholars under Grant 62025603; in part by the National Natural Science Foundation of China under Grant U1705262, Grant 62176222, Grant 62176226, Grant 62072386, Grant 62072387, Grant 62002305, Grant 61772443, Grant 61702136, and Grant 61702136; in part by the Guangdong Basic and Applied Basic Research Foundation Grant 2021B1515120049; in part by the Fundamental Research Funds for the Central Universities under Grant 20720200097, Grant 20720200090, and Grant 20720200091. (Corresponding author: Liujuan Cao.)

Mingbao Lin is with the Media Analytics and Computing Laboratory, Department of Artificial Intelligence, School of Informatics, Xiamen University, Xiamen 361005, China, and also with the YouTu Laboratory, Tencent, Shangha 200233, China.

Liujuan Cao is with the Fujian Key Laboratory of Sensing and Computing for Smart City, Computer Science Department, School of Informatics, Xiamen University, Xiamen 361005, China (e-mail: caolj@xmu.edu.cn).

Yuxin Zhang is with the Media Analytics and Computing Laboratory, Department of Artificial Intelligence, School of Informatics, Xiamen University, Xiamen 361005, China.

Ling Shao is with the Inception Institute of Artificial Intelligence, Abu Dhabi, United Arab Emirates, and also with the Mohamed bin Zayed University of Artificial Intelligence, Abu Dhabi, United Arab Emirates.

Chia-Wen Lin is with the Department of Electrical Engineering and the Institute of Communications Engineering, National Tsing Hua University, Hsinchu 30013, Taiwan.

Rongrong Ji is with the Media Analytics and Computing Laboratory, Department of Artificial Intelligence, School of Informatics, Xiamen University, Xiamen 361005, China, and also with the Institute of Artificial Intelligence, Xiamen University, Xiamen 361005, China.

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TNNLS.2022.3156047.

Digital Object Identifier 10.1109/TNNLS.2022.3156047.
the final performance [26], [29], [30]. To this end, existing methods resort to a series of complex learning steps, many of which focus on training from scratch with additional sparsity constraints. For instance, methods in [31] and [32] employ joint-retraining with sparse requirements on the scaling factors of batch normalization layers, and the pruning rate in each layer relies on a given threshold. Huang and Wang [25] proposed to train CNNs with the 0-1 mask on each filter, and the percentage of 1s in each layer makes up of the pruned network structure. The method in [33] takes previous activation responses as inputs and generates a binary index code for pruning. Similar to Huang and Wang [25], the pruned network structure consists of the ratio of trained nonzero indexes. Dynamic pruning [34] incorporates a feedback scheme to reanimate the pruned filters, which, thus, achieves dynamic allocation of the sparsity in each layer. Another group [27], [35] requires human experts to designate the layerwise pruning strategy, which is simple but quantitatively suboptimal. More recent works [26], [30], [36], [37] focus on search-based strategies, typically through network architecture search [36], one-shot architecture search [37], or heuristic-based search algorithms, such as evolutionary algorithm [26] and artificial bee colony [30]. Although search-based methods generally result in a better network structure, their search progress is extremely time-consuming.

As the second issue, the filter importance measurement identifies which filters in the pretrained model should be preserved and inherited to initialize the pruned network structure. Existing works focus on measuring the individual filter importance. To this end, many of them resort to preserving the most “important” filters by a certain criterion to estimate the filter importance, such as magnitude-based [31], zero percentage of output activation [38], and rank of feature map [27]. However, the methods in [27] and [28] are data-driven and add complexity in evaluation, and the method in [31] is more effective in weight pruning [13], [35] rather than filter pruning, as demonstrated in [39]. Besides, these methods usually require layerwise fine-tuning to improve inference accuracy, which is also time-consuming. Training-from-scratch methods [25], [31]–[34] preserve the weights of nonzero masked filters or filters with fewer sparse factors for the follow-up fine-tuning. The methods in [26], [30], and [36] adopt a random measurement to assign filter weights with random Gaussian distribution or randomly pick up some of the pretrained filter weights. Besides, methods in [26] and [37] also require training a large auxiliary network to predict the weights of potential pruned network structure, making the pruning more complex.

In this article, we propose a novel pruning method, termed CLR-RNF, which consists of two components of CLR and RNF to, respectively, solve the above two problems. The former aims to efficiently find the optimal pruned network structure and the latter targets to select a subgroup of important filters to initialize the pruned network structure such that the pruned model performance can be effectively recovered. To find the optimal pruned network structure, we adopt the effective magnitude-based criterion in weight pruning [13], [35] and introduce a cross-layer ranking (CLR) of weights. As a result, the pruned network structure with our filter pruning scheme also benefits from the per-layer sparsity employed in weight pruning. For the first time, we reveal the “long-tail” pruning problem in the magnitude-based weight pruning, as illustrated in Fig. 1, and propose a computation-aware measurement of weight importance to effectively address the inefficiency in network pruning caused by the long tail. To select a subgroup of important filters, instead of selecting filters based on their individual importance, we prefer to measure the collective importance of a filter group for selection, which is based on our insight that per-layer filters are involved in a coalition to achieve the desired performance. Specifically, each filter in the pretrained model would recommend a group of its closest filters that have a high potential to be inherited by the pruned model. Correspondingly, a $k$-reciprocal nearest filter (RNF) selection is proposed to pick up filters that fall into the intersection of all the recommended groups as the final inherited filters. Both our pruned network structure and filter selection are nonlearning, which, thus, greatly simplifies the complexity in filter pruning.

To sum up, the main contributions of this work include the following.

1) For the first time, we reveal the “long-tail” pruning problem in the magnitude-based weight pruning, which degrades the efficacy of a pruned network, and propose a new computation-aware measurement to effectively address the problem.

2) We propose to treat the per-layer sparsity in the CLR of weight pruning as the per-layer pruning rate for filter pruning. To the best of our knowledge, this is the first work that utilizes the linkage between filter pruning and weight pruning.

3) We propose a novel recommendation-based filter selection scheme based on the $k$-reciprocal nearest neighbors recommended by individual filters in a layer. The method selects a group of filters by taking into account the overall collective importance of the filter group, rather than the importance of individual filters.

II. RELATED WORK

A. Weight Pruning

In contrast to filter pruning, weight pruning pursues to remove individual neurons in the weight tensors of a neural network by a certain criterion or training technique, such
as second-order Taylor expansion [40], second-order derivative [41], $\ell_2$-regularization [12], global sparse momentum SGD [42], and magnitude of weight value [13], [43]–[45]. After removing the neurons, the weight tensors become highly sparse, and the memory can be reduced by arranging the model in a sparse format. Specialized hardware and software are, thus, required to achieve practical speedups. Differently, we focus on filter pruning but aim to make full use of the magnitude-based weight pruning to derive the pruned network structure.

### B. Neural Architecture Search

Recently, neural architecture search (NAS) has attracted increasing attention. It aims to design a network architecture in an automated way with as little human intervention as possible, typically through reinforcement learning [46], evolutionary learning [47], differentiable search [48], and so on. Similar to NAS, recent arts resort to search-based strategies for the pruned network structure [26], [30]. Differently, the search space of NAS is broad (operations, the filter number, network depth, and so on) and is defined distinctively across different works. On the contrary, filter pruning focuses on the decision of per-layer filter number to produce a subset of a given network, which can be seen as a simplified version of architecture search.

### III. METHODOLOGY

#### A. Preliminary

Consider a pretrained CNN with $L$ convolutional layers $C = \{C_1, C_2, \ldots, C_L\}$, whose kernels are $\mathbf{K} = \{\mathbf{K}_1, \mathbf{K}_2, \ldots, \mathbf{K}_L\}$, where $C_i$ denotes the $i$th convolutional layer with kernel $\mathbf{K}_i = \{\mathbf{k}_1, \mathbf{k}_2, \ldots, \mathbf{k}_{n_i}\}$ consisting of $n_i$ filters. The $j$th filter of $\mathbf{K}_i$ can be represented by a three-way tensor $\mathbf{k}_j \in \mathbb{R}^{n_{i-1} \times h_i \times w_i}$, where $n_{i-1}$, $h_i$, and $w_i$ stand for the channel number, height, and width of the filter, respectively. As can be seen, the channel number in the $i$th layer is equal to the filter number in the $(i - 1)$th layer. For ease of presentation, we reformat each filter with the shape of $\mathbf{k}_j \in \mathbb{R}^{n_{i-1} \times h_i \times w_i}$. We denote the $q$th weight element in $\mathbf{k}_j$ as $\theta_{\mathbf{k}_j,q}$, and each weight in our setting would be assigned with an importance estimate denoted as $\theta_{\mathbf{k}_j,q}$.

Given a global pruning rate $p$, filter pruning aims to find and prune redundant filters in each layer of a network to obtain a compressed representation of the pruned network $\hat{\mathbf{K}}_i = \{\hat{\mathbf{k}}_1^i, \hat{\mathbf{k}}_2^i, \ldots, \hat{\mathbf{k}}_{n_i}^i\} \subseteq \mathbf{K}_i$ with $\hat{\mathbf{k}}_j^i \in \mathbb{R}^{n_{i-1} \times h_i \times w_i}$. By denoting the pruning rate in the $i$th layer as $p_i$, we have $\tilde{n}_i = \lceil (1 - p_i) \cdot n_i \rceil$, where $\lceil \cdot \rceil$ rounds its input to the nearest integer. $\hat{\mathbf{K}}_i$ is subsequently end-to-end fine-tuned to recover the accuracy performance.

As discussed in Section I, the pruned network structure and the filter importance measurement are two important factors impacting the final pruning performance: the former reflected in the value of $p_i$ (or $\tilde{n}_i$) and the latter reflected in the filters of $\hat{\mathbf{K}}_i$. To that effect, with $p$, prevalent methods resort to a series of complex learning in finding $p_i$ and focus on measuring the importance of individual filters to locate $\hat{\mathbf{K}}_i$. Instead, we aim to improve the filter pruning by proposing two nonlearning components for finding a better pruned network structure and identifying a filter subset with better collective importance.

#### B. Cross-Layer Ranking

Fig. 2 shows our policy for pruned network structure. Differently, our CLR dates back to the weight pruning [12], [13], [43]–[45], which directly measures the importance of each individual weight by its magnitude, i.e.,

$$
\left( \theta_{\mathbf{k}_j}^i \right)_q = \left| \mathbf{k}_j \right|_q
$$

(1)

where $| \cdot |$ returns the absolute value of its input. We then conduct a weight ranking across the whole network by the numerical order of $\left( \theta_{\mathbf{k}_j}^i \right)_q$. Given a global pruning rate $p$, it can be easily achieved by pruning out the lowest-rated weights. As a result, the weight pruning leads each filter $\mathbf{k}_j$ to a sparse $\hat{\mathbf{k}}_j^i$ whose elements are defined as

$$
\left( \hat{\mathbf{k}}_j^i \right)_q = \begin{cases} 0, & \text{if } \left( \theta_{\mathbf{k}_j}^i \right)_q \text{ is among the lowest-rated} \\ \left( \mathbf{k}_j \right)_q, & \text{otherwise} \end{cases}
$$

(2)

It has been demonstrated that more than 90% of network parameters can be safely removed by weight pruning without compromising performance [12], [13], [35], [40] since weight pruning considers the ranking relationship across different architectures.
structure by filter pruning is to determine the per-layer pruning the pruned network structure by filter pruning. Besides global redundancy, we believe that per-layer sparsity in weight pruning provides useful information for determining the pruned network structure by filter pruning.

A straightforward method for designing the pruned network structure by filter pruning is to determine the per-layer pruning rate \( p_i \) based on the per-layer sparsity after weight pruning as

\[
p_i = \frac{\sum_j \sum_{n_{i-1} \leq h_j \leq n_i} \delta \left( \left( k^j_i \right)_q \neq 0 \right)}{n_i \cdot n_{i-1} \cdot h_j \cdot w_i}
\]

where \( \delta(\cdot) \) is an indicator function, which returns 1 if the input is true and 0 otherwise.

The definition of \( p_i \) in (3) equalizes the sparsity of the \( i \)th layer in the weight pruning. However, the weight importance in (1) is closely related to the network compression while ignoring the FLOPs in the network, which is directly related to the acceleration. We empirically observe that the magnitudes of weights in the top layers are usually smaller than those in the bottom layers, as shown in Fig. 1. Consequently, the weight pruning [12], [13], [43]–[45] tends to remove more individual weights in the top layers, known as “long-tail” pruning.

In Fig. 3(a), we show the Kullback–Leibler divergence between the initial weights and the weights derived in different training stages. As can be seen, the trained weights in the top layers show a significant difference from the initial weights, while the bottom-layer weights are nearly unchanged. Fig. 3(b) shows the per-layer mean of weight magnitudes in different training epochs, and it is clear that the great changes in Fig. 3(a) result in smaller weight values in the top layers. We relate this phenomenon to the “gradient vanishing” in network learning, i.e., the bottom-layer gradient is vanishingly small; thus, the filter weights keep unchanged during training. As a result, a large portion of bottom-ranked weights is concentrated in the top layers, returning higher pruning rates, as shown in Fig. 3(c).

Besides, more FLOPs are typically consumed in the bottom convolutional layers due to the larger input feature maps. Only considering weight magnitude rarely accelerates inference after pruning. Moreover, different from the weight pruning that simply zeros out certain weight elements but does not change the network structure, filter pruning is more sensitive to the network structure since the whole filters are removed. It will make the network unstable if most filters are removed from the top layers. Thus, the per-layer sparsity should be well balanced while also considering per-layer computation.

To this end, we propose to retain the pruned network structure formulation in (3) for its easy implementation while redefining a computation-aware importance estimate for each individual weight \( (k^j_i)_q \) as

\[
(\theta^j_i)_q = \frac{\left| (k^j_i)_q \right|}{\#\text{FLOPs}_i} \quad (4)
\]

where \#\text{FLOPs}_i returns the FLOPs count in the \( i \)th layer and \( \lambda \geq 0 \) is a hyperparameter shared across the network.

Equation (4) is a generalization of (1). By setting \( \lambda = 0 \), it degenerates to the magnitude-based importance measurement of (1) widely used in [12], [13], and [43]–[45]. With a fixed \( \lambda \), smaller magnitude weights with more computation consumption lead to fewer importance estimates and then tend to be removed. Therefore, it can well tackle the “long-tail” pruning problem arising from (1). Moreover, our pruned network structure using (2) considers the CLR of pretrained weights. As validated in Section IV, better performance can be obtained since the global relationship is considered. Besides, it can be easily implemented without any complex learning requirement, which differs our method from existing search-based works [26], [30].

C. k-Reciprocal Nearest Filters

Given pruning rate \( p_i \) in (3) based on the importance estimate in (4), we have the number of preserved filters: \( \tilde{N}_i = \lfloor (1 - p_i) \cdot n_i \rfloor \). The next step lies in finding \( \tilde{N}_i \) most important filters, which would then be transferred to the pruned network structure for the follow-up fine-tuning. Our method lies in measuring the collective importance of a filter subset with the size of \( \tilde{N}_i \) rather than simply considering the individual filter importance in most previous methods [24], [27], [32], [35], [49]. Our insight is that, since the filters in each layer work collectively to achieve the desired outcome, we should consider the collective importance of all candidate filters. To this end, as outlined in Fig. 4, we propose a recommendation-based filter selection framework where each filter can suggest a group of \( k \) filters that have a higher potential to be inherited by the pruned network. The final selected filter set is picked up from these groups using our \( k \)-RNFs.

To that effect, we first build the similarity matrix \( S_{ij} \in \mathbb{R}^{n_i \times n_i} \) to model the normalized closeness among the \( i \)th layer pretrained filters \( K_i \), whose elements are defined as

\[
S_{ij}^h = \frac{\exp(-D^2(k^j_i, k^h_i))}{\sum_{q=1}^{n_i} \exp(-D^2(k^j_i, k^q_i))} \quad 1 \leq i \leq L, \quad 1 \leq j, \quad h \leq n_i
\]

(5)

where \( D(\cdot, \cdot) \) is a distance function. While other metrics can be used, we simply consider the \( \ell_2 \)-norm in our implementation, which can well reflect the closeness between filter \( k^j_i \) and filter \( k^h_i \) in our empirical observation. Based on the closeness
metric, we then further define the closeness rank of filter $k_i^h$ with respect to filter $k_j^j$ as follows:

$$\mathcal{CR}(k_i^h | k_j^j) = 1 + \sum_{g=1}^{n_i} \delta(S_{i}^{jg} > S_{i}^{gh})$$

(6)

where $\delta(\cdot)$ returns 1 if the input is true and 0 otherwise.

For the $t$th layer, each filter $k_i^j$ would recommend a group of its nearest-neighbor filters in $K_i$ as the candidates since these filters tend to be much closer to $k_i^j$. We can then construct a recommendation set with $k$ filters from $k_i^j$ as

$$\mathcal{N}_{k_i^j} = \{k_i^h | \mathcal{CR}(k_i^h | k_i^j) \leq k, h = 1, \ldots, n_i\}$$

(7)

which captures the $k$ nearest neighbors ($k$-NN) of filter $k_i^j$ in $K_i$. Although the $k$-NN filters of each filter in $K_i$ form good candidates for selecting filters in filter pruning, it is highly possible that different filters make different recommendations. Simply choosing one of the recommendation sets is inappropriate since the chosen recommendation may be close to the reference while being far away from others. To solve this, we propose the following $k$-RNF set:

$$\mathcal{K}_i = \mathcal{N}_{k_i^1} \cap \mathcal{N}_{k_i^2} \cap \cdots \cap \mathcal{N}_{k_i^{n_i}}.$$ 

(8)

As can be seen, the $k$-RNF set is defined as the intersection of the $k$-NNs of all filters in $K_i$. It puts a stricter requirement on the final selected filter set that each picked filter $k_i^j \in \mathcal{K}_i$ should fall into the $k$-NN of every pretrained filter rather than a single one. Thus, some of the low-value neighbors, i.e., close to a particular filter but far away from others, can be excluded.

The size of $\mathcal{K}_i$ may be smaller than the target number of preserved filters $\bar{n}_i$, i.e., $|\mathcal{K}_i| < \bar{n}_i$. To solve it, as shown in Fig. 4, starting with $k = \bar{n}_i$, we increase the value of $k$ with a step of 1 until the number of filters in $\mathcal{K}_i$ reaches $\bar{n}_i$.

We summarize our pruning steps in Algorithm 1. Lines 1–12 summarize our CLR for the pruned network structure, and Lines 13–19 outline our RNF for the filter selection.

IV. EXPERIMENTS

A. Implementation Settings

1) Training Details: All our pruned models are fine-tuned via stochastic gradient descent (SGD) optimizer with a momentum of 0.9 and a batch size of 256. On CIFAR-10, we fine-tune each pruned network for 150 epochs with a learning rate of 0.1, which is decayed to 0.01 and 0.001, respectively, after 50 and 100 epochs. Without specifications, on ImageNet, we train ResNet-50 for 90 epochs with a weight decay of 10−4. The initial learning rate is set to be 0.1 and is divided by 10 every 30 epochs. Without specifications, for all methods, we apply the random crop and horizontal flip to the input images, which are also official operations in Pytorch. To stress, other techniques for image augmentation, such as lightening and color jitter, can be applied to further improve the performance as done in the implementations of Liu et al. [26], He et al. [59], and Luo et al. [61], which, however, are not considered in
this article. Fig. 5 displays our pruning strategies for networks with/without the shortcut connections.

2) Performance Metrics: The numbers of FLOPs and parameters, and their corresponding pruning rate (denoted as PR) are reported to measure the efficacy of our CLR-RNF and compared methods. The numbers of FLOPs and parameters reflect the computation cost and storage consumption. Besides, for CIFAR-10, we report the top-one accuracy. For ImageNet, both the top-one and top-five accuracies are reported.

B. CIFAR-10

On CIFAR-10, we compare our CLR-RNF with several state-of-the-arts (SOTAs), including [24], [25], [27], [30], [32], [35], [50], and [52]. More detailed analyses are provided in the following.

1) VGGNet: We apply our CLR-RNF to prune the 16-layer VGGNet model, a popular sequential CNN for object detection and semantic segmentation. As shown in Table I, CLR-RNF significantly outperforms the SOTAs for all performance metrics mentioned in Section IV-A2. Our CLR-RNF can achieve about 20 \times parameters compression and boost the computation for 4 \times with even 0.3% top-one accuracy improvement, which greatly facilitates the VGGNet model to be deployed on resource-limited devices.

2) GoogLeNet: As shown in Table II, with negligible top-one accuracy drops (94.85% for CLR-RNF versus 95.03% for the baseline), our CLR-RNF can reduce 67.9% FLOPs and 64.7% parameters. In comparison with the best state of the art, i.e., HRank, CLR-RNF achieves higher accuracy performance while significantly reducing the numbers and FLOPs and parameters. Thus, CLR-RNF well shows its ability to reduce the redundancy of networks with the multibranch structure.

3) ResNet: We choose to prune ResNet-56 and ResNet-110 to demonstrate the effectiveness of our CLR-RNF for networks with residual blocks. In Table III, CLR-RNF takes the lead in both the top-one accuracy and the FLOPs/parameters compression rates. For ResNet-56, CLR-RNF reduces the numbers of parameters and FLOPs by 57.3% and 55.5%, respectively, without sacrificing the accuracy (93.27% for CLR-RNF and 93.26% for the baseline). For ResNet-110, CLR-RNF can reduce 66.0% FLOPs and 69.1% parameters while increasing the accuracy performance by 0.14% (93.71% for CLR-RNF and 93.57% for the baseline). These results show that CLR-RNF can effectively compress networks with residual blocks.

In Fig. 6, we further compare the top-one accuracy of the models compressed by GAL [54], FilterSketch [50], and our
CLR-RNF with different pruning rates using ResNet-56. GAL suffers severe accuracy drops as the complexity reduction goes deeper. In the case of a small pruning rate ($\leq 40\%$), our CLR-RNF and FilterSketch achieve similar accuracy performances. However, a large accuracy drop occurs with FilterSketch when the pruning rate is around 50%, whereas our CLR-RNF can still maintain a stable performance. Though the accuracy of CLR-RNF starts a clear drop when the pruning rate is more than 60%, it still outperforms FilterSketch by a margin and shows an overwhelming gain over GAL, thereby well demonstrating the superiority of CLR-RNF.

C. ImageNet

We further the results on ImageNet by comparing with several SOTAs [24]–[28], [30], [33], [54], [61]–[63]. We compare the accuracy performance under similar FLOPs/parameter reductions or compare the complexity reductions under similar accuracy performance.

1) ResNet-50: Table IV shows that CLR-RNF outperforms the other pruning methods in terms of both complexity reduction and accuracy. For example, when setting the global pruning rate $p$ to 0.52, CLR-RNF reduces the pretrained ResNet-50 to a smaller network with only 0.93 B FLOPs and 6.90 M parameters. Compared to the search-based ABCPruner-30% that has 0.94 B FLOPs and 7.35 M parameters, with more complexity reductions, CLR-RNF still achieves better performances (71.11\% for CLR-RNF and 70.42\% for ABCPruner in the top-one accuracy; 90.42\% for CLR-RNF and 89.63\% for ABCPruner in the top-five accuracy). Similar observations can be found with different values of $p$, such as 0.44 or 0.20.

Following the recent advances, e.g., AutoPruner [33], we further apply the learning rate with cosine scheduler, where the initial learning rate is set to 0.1 and the weight decay is set to $4 \times 10^{-5}$. A total of 100 training epochs are applied. As shown in Table IV, the superiority of CLR-RNF is evident. With significantly reduced model complexity, our CLR-RNF also achieves the top-one accuracy of 73.34\%, significantly better than AutoPruner of 73.05\%.

2) Efficiency of Cross-Layer Ranking: As stressed in Section I, prevalent methods resort to a series of complex learning steps in the decision of pruned network structure, such as search-based strategies [26], [30]. Our CLR lies in its simplicity by reranking the weight importance. Table V compares the runtime complexity between our CLR and ABCPruner [30] employing artificial colony bee as the search algorithm. As can be observed, our CLR, with a single CPU implementation, consumes only a few seconds to derive the pruned network structure, whereas it takes several hours or even days with ABCPruner on an NVIDIA Tesla V100 GPU platform. Note that ABCPruner would consume much more time to derive the pruned models on other lower computing devices, such as NVIDIA 1080 GPUs and CPUs. To analyze, the search-based strategy has to repeatedly apply search operations and measure the quality of each structure by a fitness function, both of which are computationally very expensive. Thus, the efficiency of our CLR is evident.
TABLE VI

| Model        | Top1 (%) | FLOPs (PR) | Parameters (PR) |
|--------------|----------|------------|-----------------|
| VGG-CLR      | 93.32    | 81.31M (74.1%) | 0.74M (95.0%)   |
| VGG-ABC      | 93.01    | 82.81M (73.7%) | 1.67M (88.7%)   |
| VGG-Human    | 92.91    | 82.93M (73.6%) | 1.23M (91.6%)   |
| ResNet-56-CLR| 93.27    | 54.00M (57.3%) | 0.38M (55.5%)   |
| ResNet-56-ABC| 93.13    | 58.54M (54.1%) | 0.39M (54.2%)   |
| ResNet-56-Human | 92.97  | 58.01M (54.2%) | 0.38M (55.5%)   |
| ResNet-110-CLR| 93.71    | 86.80M (66.0%) | 0.53M (69.1%)   |
| ResNet-110-ABC| 93.32    | 89.87M (65.0%) | 0.56M (67.4%)   |
| ResNet-110-Human | 93.27   | 96.49M (62.7%) | 0.62M (64.4%)   |
| GoogLeNet-CLR| 94.85    | 491.54M (67.9%) | 2.18M (64.7%)  |
| GoogLeNet-ABC| 94.47    | 513.19M (66.6%) | 2.46M (60.1%)  |
| GoogLeNet-Human | 94.01   | 520.37M (66.1%) | 2.29M (62.9%)  |
| ResNet-50-CLR| 71.11    | 0.93B (77.4%) | 6.90M (73.0%)   |
| ResNet-50-ABC| 70.53    | 0.94B (77.1%) | 7.35M (71.3%)   |
| ResNet-50-Human | 69.40   | 0.96B (76.7%) | 6.92M (72.9%)   |

Fig. 7. Top-one accuracy of pruned VGGNet, ResNet-56/110, and GoogLeNet on CIFAR-10 and ResNet-50 on ImageNet. All methods use the same pruned network structures given by our CLR.

D. Ablation Study

In this section, we show the ablation studies to, respectively, explore the effectiveness of our CLR and k-RNFs. All the experimental results are conducted on CIFAR-10 using VGGNet, ResNet-56/110, and GoogleNet and on ImageNet using ResNet-50.

1) Effectiveness of Cross-Layer Ranking: For comparisons, we also consider the pruned network structures given by the search-based artificial bee colony [30] and human-designated policy [27]. All strategies are fed with filter weights from our k-RNF. From Table VI, besides more complexity reductions, our CLR achieves better accuracy performances as well, validating the efficacy of our CLR to find a better pruned network structure.

2) Effectiveness of k-Reciprocal Nearest Filter: To show the effectiveness of our k-RNF, we further display the performances of different filter selection methods, including k-means, \( l_1 \) -norm, and randomness on top of the same pruned network structures given by our CLR. Fig. 7 shows that our k-RNF outranks other scenarios. This means that our filter selection can recommend more representative filters for better performance.

V. Conclusion

We proposed a novel filter pruning method, called CLR-RNF, involving two nonlearning methodologies, CLR and k-RNF, which aim to find the optimal pruned network structure and locate a filter subset with better collective importance. We first revealed the “long-tail” pruning problem in the magnitude-based weight pruning and proposed a CLR strategy to remove the least important weights. Instead of considering individual filter importance, we have devised a recommendation-based filter selection to pick filters with the best collective importance. Each filter would recommend a group of its closest filters as the potential candidates. Then, the k-RNFs fall into the intersection of different recommendation sets. Experiments on CIFAR-10 and ImageNet demonstrate the efficiency and effectiveness of our CLR-RNF.

REFERENCES

[1] V. Vanhoucke, A. Senior, and M. Z. Mao, “Improving the speed of neural networks on CPUs,” in Proc. Adv. Neural Inf. Process. Syst. Workshop, 2011, pp. 1–8.
[2] A. Zhou, A. Yao, Y. Guo, L. Xu, and Y. Chen, “Incremental network quantization: Towards lossless CNNs with low-precision weights,” in Proc. Int. Conf. Learn. Represent., 2017, pp. 1–14.
[3] M. Lin et al., “Rotated binary neural network,” in Proc. Adv. Neural Inf. Process. Syst., 2020, pp. 7474–7485.
[4] X. Zhang, X. Zhou, M. Lin, and J. Sun, “ShuffleNet: An extremely efficient convolutional neural network for mobile devices,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., Jun. 2018, pp. 6848–6856.
[5] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun, “ShuffleNet V2: Practical guidelines for efficient CNN architecture design,” in Proc. Eur. Conf. Comput. Vis., 2018, pp. 116–131.
[6] A. G. Howard et al., “MobileNets: Efficient convolutional neural networks for mobile vision applications,” 2017, arXiv:1704.04861.
[7] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, “MobileNetV2: Inverted residuals and linear bottlenecks,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., Jun. 2018, pp. 4510–4520.
[8] A. Howard et al., “Searching for MobileNetV3,” in Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV), Oct. 2019, pp. 1314–1324.
[9] K. Han, Y. Wang, Q. Tian, J. Guo, C. Xu, and C. Xu, “GhostNet: More features from cheap operations,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2020, pp. 1580–1589.
[10] S. Lin, R. Ji, C. Chen, D. Tao, and J. Luo, “Holistic CNN compression via low-rank decomposition with knowledge transfer,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 41, no. 12, pp. 2889–2905, Dec. 2019.
[11] K. Hayashi, T. Yamaguchi, Y. Sugawara, and S.-I. Maeda, “Exploring unexplored tensor network decompositions for convolutional neural networks,” in Proc. Adv. Neural Inf. Process. Syst., 2019, pp. 5552–5562.
[12] S. Han, J. Pool, J. Tran, and W. Dally, “Learning both weights and connections for efficient neural network,” in Proc. Adv. Neural Inf. Process. Syst., 2015, pp. 1135–1143.
[13] J. Frankle and M. Carbin, “The lottery ticket hypothesis: Finding sparse, trainable neural networks,” in Proc. Int. Conf. Learn. Represent., 2019, pp. 1–42.
[14] A. Zhou et al., “Learning N:M fine-grained structured sparse neural networks from scratch,” in Proc. Int. Conf. Learn. Represent., 2021, pp. 1–15.
[15] J. Choquette, W. Gandhi, O. Giroux, N. Stam, and R. Krashinsky, “NVIDIA A100 tensor core GPU: Performance and innovation,” IEEE Micro, vol. 41, no. 2, pp. 29–35, Mar. 2021.
[16] N. Liu, X. Ma, Z. Xu, Y. Wang, J. Tang, and J. Ye, “AutoCompress: An automatic DNN structured pruning framework for ultra-high compression rates,” in Proc. AAAI Conf. Artif. Intell., 2020, vol. 34, no. 4, pp. 4876–4883.
C. Zhao, B. Ni, J. Zhang, Q. Zhao, W. Zhang, and Q. Tian, “Varia-
Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, and C. Zhang, “Learning
M. Lin, R. Ji, Y. Zhang, B. Zhang, Y. Wu, and Y. Tian, “Channel
B. Li, B. Wu, J. Su, and G. Wang, “EagleEye: Fast sub-net evaluation
J.-H. Luo and J. Wu, “AutoPruner: An end-to-end trainable filter pruning
J. Liu, M. Sun, T. Zhou, G. Huang, and T. Darrell, “Rethinking the
Z. Huang and N. Wang, “Data-driven sparse structure selection for
E. Elsen, M. Dukhan, T. Gale, and K. Simonyan, “Fast sparse ConvNets,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2019, pp. 4943–4953.
Y. He, X. Zhang, and J. Sun, “Channel pruning for accelerating very deep neural networks,” in Proc. IEEE Int. Conf. Comput. Vis. (ICCV), Oct. 2017, pp. 1389–1397.
Z. Huang and N. Wang, “Data-driven sparse structure selection for deep neural networks,” in Proc. Eur. Conf. Comput. Vis., 2018, pp. 2410–2420.
Z. Liu et al., “MetaPruning: Meta learning for automatic neural network channel pruning,” in Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV), Oct. 2019, pp. 3296–3305.
M. Lin et al., “HRank: Filter pruning using high-rank feature map,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2020, pp. 4934–4939.
B. Li, B. Wu, J. Su, and G. Wang, “EagleEye: Fast sub-net evaluation for efficient neural network pruning,” in Proc. Eur. Conf. Comput. Vis., 2020, pp. 639–654.
Z. Liu, M. Sun, T. Zhou, G. Huang, and T. Darrell, “Rethinking the value of network pruning,” in Proc. Int. Conf. Learn. Represent. 2019, pp. 304–320.
M. Lin, R. Ji, Y. Zhang, B. Zhang, Y. Wu, and Y. Tian, “Channel pruning via automatic structure search,” in Proc. 29th Int. Joint Conf. Artif. Intell., Jul. 2020, pp. 673–679.
Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, and C. Zhang, “Learning efficient convolutional networks through network slimming,” in Proc. IEEE Int. Conf. Comput. Vis. (ICCV), Oct. 2017, pp. 2736–2744.
C. Zhao, B. Ni, J. Zhang, Q. Zhao, W. Zhang, and Q. Tian, “Variational convolutional neural network pruning,” in Proc. IEEE/CVF Int. Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2019, pp. 2780–2789.
J.-H. Luo and J. Wu, “AutoPruner: An end-to-end trainable filter pruning method for efficient deep model inference,” Pattern. Comput. Vis., vol. 107, Nov. 2020, pp. 107461.
T. Lin, U. Sinha, L. Barba, D. Dmitriev, and M. Jaggi, “Dynamic model pruning with feedback,” in Proc. Int. Conf. Learn. Represent., 2020, pp. 1–22.
H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, “Pruning filters for efficient ConvNets,” in Proc. Int. Conf. Learn. Represent., 2020, pp. 1–13.
X. Dong and Y. Yang, “Network pruning via transformable architecture search,” in Proc. Adv. Neural Inf. Process. Syst., 2019, pp. 760–771.
J. Yu and T. Huang, “AutoSlim: Towards one-shot architecture search for channel numbers,” 2019, arXiv:1903.11728.
H. Hu, R. Peng, Y.-W. Tai, and C.-K. Tang, “Network trimming: A data-driven neuron pruning approach towards efficient deep architectures,” 2016, arXiv:1607.07520.
J. Ye, X. Lu, Z. Lin, and J. Z. Wang, “Rethinking the smaller-norm less-informative assumption in channel pruning of convolution layers,” in Proc. Int. Conf. Learn. Represent., 2018, pp. 1–11.
Y. LeCun, J. S. Denker, and S. A. Solla, “Optimal brain damage,” in Proc. Adv. Neural Inf. Process. Syst., 1990, pp. 598–605.
X. Dong, S. Chen, and Y. Pan, “Learning to prune deep neural networks via layer-wise optimal brain surgeon,” in Proc. Adv. Neural Inf. Process. Syst., 2017, pp. 598–605.
X. Ding, X. Zhou, Y. Guo, J. Han, and J. Liu, “Global sparse momentum SGD for pruning very deep neural networks,” in Proc. Adv. Neural Inf. Process. Syst., 2019, pp. 6382–6394.
M. Zhu and S. Gupta, “To prune, or not to prune: Exploring the efficacy of pruning for model compression,” in Proc. Int. Conf. Learn. Represent., 2018, pp. 1–11.
Liujuan Cao received the B.S., M.S., and Ph.D. degrees from the School of Computer Science and Technology, Harbin Engineering University, Harbin, China, in 2005, 2008, and 2013, respectively. She is currently an Associate Professor with Xiamen University, Xiamen, China. She has authored over 40 papers in the top and major tired journals and conferences, including Conference on Computer Vision and Pattern Recognition (CVPR) and IEEE TRANSACTIONS ON IMAGE PROCESSING (TIP). Her research interests are computer vision and pattern recognition.

Dr. Cao is also the Financial Chair of the IEEE International Workshop on Multimedia Signal Processing (MMSP) 2015, the Workshop Chair of the ACM International Conference on Internet Multimedia Computing and Service (ICIMCS) 2016, and the Local Chair of the Visual and Learning Seminar 2017.

Yuxin Zhang is currently pursuing the B.S. degree with Xiamen University, Xiamen, China.

His research interests include computer vision, and neural network compression and acceleration.

Chia-Wen Lin (Fellow, IEEE) received the Ph.D. degree in electrical engineering from National Tsing Hua University (NTHU), Hsinchu, Taiwan, in 2000.

He was with the Department of Computer Science and Information Engineering, National Chung Cheng University, Chia-Yi, Taiwan, from 2000 to 2007. Prior to joining academia, he worked for the Information and Communications Research Laboratories, Industrial Technology Research Institute, Hsinchu, from 1992 to 2000. He is currently a Professor with the Department of Electrical Engineering and the Institute of Communications Engineering, NTHU. He is also the Deputy Director of the AI Research Center, NTHU. His research interests include image and video processing, computer vision, and video networking.

Dr. Lin has served as a Steering Committee Member of IEEE TRANSACTIONS ON MULTIMEDIA from 2014 to 2015. His articles received the Best Paper Award of IEEE International Conference on Visual Communications and Image Processing (VCIP) 2015 and the Young Investigator Award of VCIP 2005. He received the Outstanding Electrical Professor Award presented by the Chinese Institute of Electrical Engineering in 2019 and the Young Investigator Award presented by the Ministry of Science and Technology, Taiwan, in 2006. He has served as the Chair of the Multimedia Systems and Applications Technical Committee of the IEEE Circuits and Systems Society from 2013 to 2015. He has served as the Technical Program Co-Chair of IEEE International Conference on Multimedia and EXPO (ICME) 2010, the General Co-Chair of IEEE VCIP 2018, and the Technical Program Co-Chair of IEEE International Conference on Image Processing (ICIP) 2019. He is also the Chair of the Steering Committee of IEEE ICME. He has served as an Associate Editor for IEEE TRANSACTIONS ON IMAGE PROCESSING, IEEE TRANSACTIONS ON MULTIMEDIA, IEEE TRANSACTIONS ON SYSTEMS FOR VIDEO TECHNOLOGY, IEEE TRANSACTIONS ON MULTIMEDIA, IEEE MULTIMEDIA, and Journal of Visual Communication and Image Representation. He has served as a Distinguished Lecturer of the IEEE Circuits and Systems Society from 2018 to 2019.

Rongrong Ji (Senior Member, IEEE) is currently a Nanqiang Distinguished Professor, the Deputy Director of the Office of Science and Technology, and the Director of the Media Analytics and Computing Laboratory, Xiamen University, Xiamen, China. He has published more than 50 papers in ACM/IEEE TRANSACTIONS, including IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE (TPAMI) and International Journal of Computer Vision (IJCV), and more than 100 full papers on top-tier conferences, such as Conference on Computer Vision and Pattern Recognition (CVPR) and Conference on Neural Information Processing Systems (NeurIPS). His publications have got over 10k citations in Google Scholar. His research falls in the field of computer vision, multimedia analysis, and machine learning.

Dr. Ji is also an Advisory Member of Artificial Intelligence Construction Journal of Computer Vision (IJCV), and the Director of the Office of Science and Technology, and the Director of the Media Analytics and Computing Laboratory, Xiamen University, Xiamen, China. He has published more than 50 papers in ACM/IEEE TRANSACTIONS, including IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE (TPAMI) and International Journal of Computer Vision (IJCV), and more than 100 full papers on top-tier conferences, such as Conference on Computer Vision and Pattern Recognition (CVPR) and Conference on Neural Information Processing Systems (NeurIPS). His publications have got over 10k citations in Google Scholar. His research falls in the field of computer vision, multimedia analysis, and machine learning.

Dr. Ji is also an Advisory Member of Artificial Intelligence Construction in the Electronic Information Education Committee of the National Ministry of Education. He was awarded the National Science Foundation for Excellent Young Scholars in 2014, the National Ten Thousand Plan for Young Top Talents in 2017, and the National Science Foundation for Distinguished Young Scholars in 2020. He was a recipient of the Best Paper Award of the ACM Multimedia 2011. He has served as the Area Chair of top-tier conferences, such as CVPR and ACM Multimedia.