Enhanced network compression through tensor decompositions and pruning

Van Tien Pham, Yassine Zniyed, Thanh Phuong Nguyen

Enhanced network compression through tensor decompositions and pruning

Van Tien Pham, Yassine Zniyed, Thanh Phuong Nguyen

***Abstract*—Network compression techniques that combine tensor decompositions and pruning have shown promise in leveraging the advantages of both strategies. In this work, we propose NORTON (enhanced Network cOmpRession through TensOr decompositions and pruNing), a novel method for network compression. NORTON introduces the concept of filter decomposition, enabling a more detailed decomposition of the network while preserving the weight's multidimensional properties. Our method incorporates a novel structured pruning approach, effectively integrating the decomposed model. Through extensive experiments on various architectures, benchmark datasets, and representative vision tasks, we demonstrate the usefulness of our method. NORTON achieves superior results compared to state-of-the-art techniques in terms of complexity and accuracy. Our code is also available for research purposes.**

***Index Terms*—tensor decompositions, structured pruning, hybrid compression, efficient inference.**

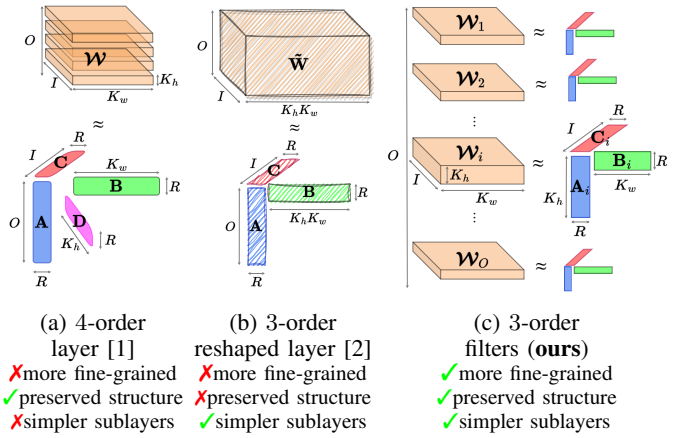


Fig. 1: Comparison of tensor decompositions approaches.

I. Introduction

NETWORK compression aims to reduce the computational and memory requirements of an existing model, enabling its deployment in resource-constrained environments without compromising its ability to generalize, *i.e.,* perform well on unseen data.

Among the existing paradigms for model compression, tensor decompositions [1], [2] and structured pruning [3] have demonstrated their effectiveness and practicality. Both approaches are built upon the hypothesis that the original model is over-parameterized, and thus, the redundant information can be eliminated either by representing the weights more efficiently using low-rank representations or by directly removing a part of them through filter pruning. In addition to their capabilities of high compression rate, these methods share the advantage of enabling the deployment of compressed models on resource-constrained devices without requiring any specialized support. However, it is noteworthy that these two approaches have mostly been developed independently in the literature, with few efforts [4], [5] to explore their combined potential and leverage their individual strengths in an orthogonal manner. This highlights the need for an integrated approach that harnesses the benefits of both tensor decompositions and structured pruning.

The motivation for combining tensor decompositions and structured pruning arises from the observation that the weights of CNNs possess both low-rank and sparse properties, as demonstrated in prior works [6]–[8]. These attributes are considered partially complementary [7]. Decomposition approaches are not designed to eliminate all redundant channels [8], while pruning approaches typically focus on removing redundant elements but may overlook the low-rank structure within the retained weights post-pruning. The natural inclination, therefore, is to integrate these two compression strategies to push the limits of network compression [6]. For instance, considering the VGG-16 architecture, RGP [9], a state-of-the-art (SOTA) filter pruning method, achieved a maximal pruning of 90.5% of MACs, while HALOC [10], representing SOTA in tensor decompositions, compressed a maximum of 86% of MACs. In other words, no reported work has effectively compressed more than 91% of MACs for this architecture. In contrast, by combining the two approaches, NORTON achieves a simultaneous compression of 99% of MACs and parameters with a modest drop, as depicted in Fig. 2. Specifically, applying only the decomposition part in NORTON can compress up to 88% of MACs (with the minimum rank 1, see Tab. VI). Conversely, applying only the pruning part, with a pruning ratio of around 90%, results in a significant accuracy drop, and the pruned model fails to recover the accuracy. However, by combining these two approaches, with rank 1 and a pruning ratio of 80%, NORTON achieves a 99% compression rate, evidently pushing the boundaries of network compression. Our work acknowledges a similar approach observed in image compression, akin to the widely used JPEG algorithm [11], where a combination of techniques enhances overall compression efficiency. Importantly, NORTON is not merely

All authors are with Université de Toulon, Aix Marseille Université, CNRS, LIS, UMR 7020, France (e-mails: van-tien-pham@etud.univ-tln.fr; zniyed@univ-tln.fr; tpnguyen@univ-tln.fr).

This work was granted access to the HPC resources of IDRIS under the allocation 2023-103147 made by GENCI. The work of T.P. Nguyen is partially supported by ANR ASTRID ROV-Chasseur.

a mechanical combination of these two approaches; it also capitalizes on their intrinsic advantages. It is noteworthy that in our approach, both stages share a common representation—the factor matrices—which are concurrently employed for both the decomposition and pruning phases.

Considering tensor decompositions, the typical pattern involves selecting a decomposition method and determining the rank for the weight tensor of the original layer. The decomposition is then applied to obtain a low-rank representation, which is used to construct the weights of the new layer. Previous works in the literature have primarily focused on the decomposition method [1], [2], [12], [13] and rank selection [14]–[17]. However, an unexplored aspect in the existing literature revolves around the decomposition of the weight tensor itself. Prior studies have predominantly focused on decomposing the entire layer as a 4-order tensor or reshaping it into a 3-order tensor before performing the decomposition. However, there is limited discussion on determining the most effective approach for decomposing the weight tensor. This aspect warrants further investigation to enhance and optimize network compression techniques.

The weight of a convolution layer is a 4-order tensor, which can be processed in different formats, each yielding distinct consequences. In Fig. 1, we illustrate three possible methods for handling this weight tensor, including layer decomposition [1], [12], [13] and reshaped-based decomposition [2], [17]–[20], while the proposed approach is referred to as *filters decomposition*. As a representative example, we consider the Canonical Polyadic decomposition (CPD) [21] to compare the differences among these approaches. The approach presented by [1] involves decomposing the entire layer, resulting in 4 factor matrices that correspond to 4 sublayers. Although this method preserves the multidimensional nature of the weight tensor, it processes the tensor at a coarse level. This opens up the potential for a more fine-grained treatment. In contrast, in [2] the authors suggested reshaping the 4-order weight tensor into a 3-order tensor, followed by CPD to obtain 3 factor matrices and 3 sublayers. The authors argued that the kernel size is relatively small so they can be ignored. However, this resizing process compromises the multidimensionality of the weight tensor, resulting in information loss during approximation. Moreover, this argument does not seem reasonable with recent modern architectures where the kernel size increases significantly [22].

Our proposed method operates on the original weight tensor in a filter-by-filter manner. The key insight lies in the fact that in a convolution layer, the input undergoes convolution separately with each filter, and the outputs are then aggregated to generate the final feature map. Therefore, it is intuitive to decompose each 3-order filter tensor individually. The filters decomposition approach offers a higher level of granularity compared to its counterpart, layer decomposition. With filters decomposition, not only the multidimensional property is strictly preserved, but also the layer's 4-order weight is spontaneously interpreted as a set of 3-order filters. Additionally, an interesting side effect of filters decomposition is that it leads to a narrower range of ranks, which simplifies the rank selection process, as demonstrated later

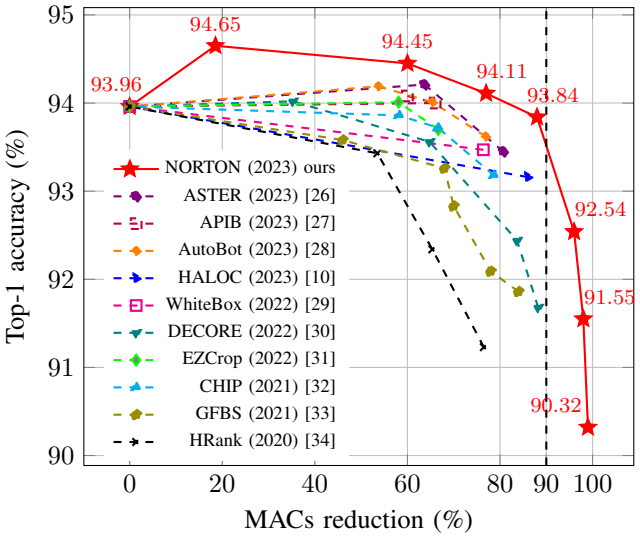


Fig. 2: The accuracy-MACs reduction Pareto curves of compressed VGG-16 models are compared on CIFAR-10.

in subsection III-B. Another notable difference is that filters decomposition replaces the original layer with 3 sublayers, while layer decomposition requires 4 sublayers, potentially increasing network depth unnecessarily and introducing the possibility of gradient vanishing issues. It is worth mentioning that the distinction between layer decomposition and filters decomposition aligns with a similar categorization seen in network pruning, specifically filter pruning [3] and weight pruning [23], emphasizing the fine-grained granularity aspect.

The second consideration of this work is to combine filters decomposition and filter pruning to leverage the independent advantages of each method. While previous studies [4], [8], [24], [25] have used low-rank representations and network pruning for compression purposes, they have not explored an orthogonal combination of these techniques. In these works, low-rank representations have been used only in the pruning step without directly contributing to the reduction of model size. In contrast, our approach takes a different direction by sequentially applying low-rank representations and pruning in two separate phases, enabling a dual compression process. To the best of our knowledge, this direction has not been extensively investigated in the literature.

There are two possible arrangements for combination: decomposing then pruning, and pruning then decomposing. Previous works [4], [5] have adopted the pruning then decomposing scheme, utilizing a Taylor expansion-based pruning criterion [3] and Tucker decomposition (TD) [21]. However, this work [4] lacks comprehensive analysis and experiments, leaving room for further investigation. To address this gap, our study explores the decomposing then pruning scheme, where the model is first decomposed using CPD and then subjected to filter pruning, as illustrated in Fig. 3. We argue that this order presents greater challenges compared to its counterpart, as the model's architecture becomes more complex after decomposition, with certain constraints imposed on the sublayers. To adapt to the decomposed components, we propose to use

the Principal Angles Between Subspaces (PABS) [35] as a suitable filter pruning metric which will be further discussed in subsection III-C.

Briefly, the contributions of this work are three-fold:

- Firstly, we introduce a novel filters decomposition method, promoting its differentiation with existing layer decomposition and reshaped decomposition methods.
- Secondly, we investigate the sequential combination of filters decomposition and filter pruning. We propose a novel filter pruning algorithm designed to address the challenges associated with this integration scheme.
- Thirdly, we evaluate the proposed framework on representative vision tasks including image classification, object detection, instance segmentation, and keypoint detection. We compare NORTON with SOTA methods in both low-rank representations and structured pruning domains to demonstrate the superiority of our method. Notably, our model with different compression levels can consistently outperform prior arts (see Fig. 2).

II. Related works

A. Low-rank Representations

Existing works can be categorized into two branches: matrix decompositions [15], [20], [36]–[40] and tensor decompositions [1], [2], [12], [13], [18], [19]. Matrix-based approaches consider the weight of the fully connected layer [36] or the convolution layer [15], [20], [37]–[40] as a 2-order matrix on which the SVD is performed to obtain two smaller matrices. A required step of this work line is that the 4-order weight tensor is flattened into a 2-order matrix [20]. On the other hand, tensor decompositions approaches [1], [2], [12], [13], [18], [19] execute directly with higher-order weight tensor of the convolution layer.

Based on the adopted decomposition, various factorizations have been examined, such as SVD [17], [20], [36], CPD [1], [41], TD [12], hierarchical TD (HTD) [13], tensor train (TT) [42], tensor ring (TR) [43] or a mixture of them [2], [44]. The pioneering work [36] employed the truncated SVD to approximate matrix weights of fully connected layers and the monochromatic approximation to compress the two first convolution layers. In the first work of layer decomposition style [1], CPD was used to exploit the redundancy of the 4-order weight tensor of convolution layers. However, this work was restricted to compressing a single layer. The authors of [12] applied TD along with the variational Bayesian matrix factorization [16] for rank selection to make a one-shot whole network compression. Similarly, the hierarchical TD was used in [13] with an energy-based algorithm for rank selection. Further, [2] combines CPD and TD to deal with the problem of degeneracy (*i.e.,* the difficulty of fine-tuning the factors). TT and TR were used in [42], [43] to obtain the sparse representation of the redundant weights. In their empirical investigation of dimension balance preferences, the authors of [44] proposed a hybrid consideration of TT for convolution layers and HTD for fully connected layers.

B. Filter Pruning

The criterion to measure the importance of filters is the foundation of filter pruning. Various approaches have been proposed to tackle this problem and they can be roughly categorized as follows: magnitude-based approaches [45]–[47], correlation-based approaches [48], activation-based approaches [27], [32], [34], regularization [26], [49]–[51], optimization [3], [52]–[55], and neural architecture search [9], [30], [56]. In [45], the $L1$ norm was used as a measure of saliency, assuming that filters with smaller magnitudes are less important. The authors of [32], [34] argued that the activation maps represent both the structure and data so the pruning criterion should be based on the feature map other than the filters. Regularization was used for learning structured sparse networks by applying different sparsity regularizers to different parameters [49]–[51], [57]. The first-order information of Taylor expansion is used to optimize the change in the loss of channel pruning in [3]. From the perspective of architecture search, [30] used reinforcement learning to find optimal compact structures. However, some drawbacks of these methods are the complex design, large search space, and high computational resource requirement which may be impractical for users.

C. Hybrid Approach

The notion of combining low-rank representations and network pruning has been suggested but lacks in-depth exploration in the existing literature [4]–[8], [24], [58]. In a previous work [8], a compressed-aware block was incorporated to acquire a low-rank weight basis and sparse channels; however, this method is data-dependent, requiring an additional training stage. Another approach [24] employed group sparsity constraints on sparsity-inducing matrices after reshaping the weight tensor to connect filter pruning and decomposition. A collaborative compression scheme was introduced in [6] to simultaneously learn model sparsity and low-rankness. In contrast, NORTON takes a distinctive approach by addressing tensor decompositions and filter pruning in two separate phases. This ensures that the advantages of both low-rank structure and filter removal contribute synergistically to the final compression by addressing sparse and redundant structures. The most closely related work to ours is [4], which employed Taylor pruning [3] and TD [12]. However, NORTON introduces the novel concept of filter decomposition and incorporates it with an adaptive filter pruning technique in the inverse order, a more challenging but promising approach. In contrast to [4], we provide comprehensive experiments and analysis to shed light on this hybrid approach.

III. NORTON Approach

Fig. 3 illustrates the overall pipeline of our approach, which comprises two main phases: decomposition and pruning. Firstly, the original model is decomposed into CPDBlocks (as explained in Subsection III-B). Next, the decomposed CPDBlocks undergo the filter pruning algorithm (as described in III-C). This algorithm selectively removes filters from the CPDBlocks based on certain criteria, effectively reducing the

model's computational and memory requirements. Finally, a fine-tuning process is conducted to refine the compact model. Empirically, only one process of fine-tuning as presented in our proposal is sufficient instead of considering it after every phase of decomposition and pruning. This choice simplifies the methodology, reduces computational requirements, and maintains satisfactory performance. This scheme is applied simultaneously to all convolution layers of the original model. However, for the sake of simplicity, subsequent sections will focus on discussing one layer, as shown in Fig. 4.

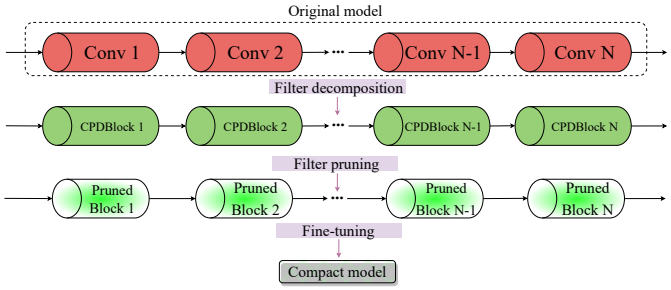


Fig. 3: Graphic illustration of the NORTON approach.

A. Preliminaries

This paper uses lower-case letters (e.g., a) to denote scalars, bold lowercase letters (e.g., $\mathbf{a}$) to represent vectors, bold uppercase letters (e.g., $\mathbf{A}$) to represent matrices, and bold calligraphic letters (e.g., $\boldsymbol{\mathcal{A}}$) to denote tensors. A tensor is a generalization of a matrix to a multi-way data array. A d-order tensor is a multi-way data array $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_d}$, where I_n is the size of mode n. $\|.\|_F$ stands for the Frobenius norm. We now introduce some definitions that will be useful in the sequel.

Definition 1: A rank-1 d-order tensor $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{I_1 \times \cdots \times I_d}$ can be expressed as a sequence of single outer product $\circ$ of d vectors as:

$$\boldsymbol{\mathcal{A}} = \mathbf{u}^{(1)} \circ \cdots \circ \mathbf{u}^{(d)}, \tag{1}$$

where $\mathbf{u}^{(n)} \in \mathbb{R}^{I_n}$ for $n = 1, \ldots, d$. Each entry of $\boldsymbol{\mathcal{A}}$ is given by $a_{i_1 \ldots i_d} = \prod_{n=1}^{d} u_{i_n}^{(n)}$, where $u_{i_n}^{(n)}$ is the i_n-th entry of the I_n-length vector $\mathbf{u}^{(n)}$.

Definition 2: The CP decomposition [21] expresses a tensor $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{I_1 \times \cdots \times I_d}$ as the sum of multiple rank-1 tensors:

$$\boldsymbol{\mathcal{A}} = \sum_{j=1}^{R} \mathbf{u}_j^{(1)} \circ \mathbf{u}_j^{(2)} \circ \cdots \circ \mathbf{u}_j^{(d)}. \tag{2}$$

The minimal integer R that ensures equality is called the canonical rank of $\boldsymbol{\mathcal{A}}$. An equivalent and compact representation of (2) is $\boldsymbol{\mathcal{A}} = [\![\mathbf{U}^{(1)}, \cdots, \mathbf{U}^{(d)}]\!]$, where $\mathbf{U}_{:,j}^{(n)} = \mathbf{u}_j^{(n)}$ and is of size $I_n \times R$. It is worth noting that CPD provides a compact representation requiring only $d \cdot I \cdot R$ parameters instead of the exponential I^d parameters in the full tensor representation, significantly reducing the memory and computational requirements.

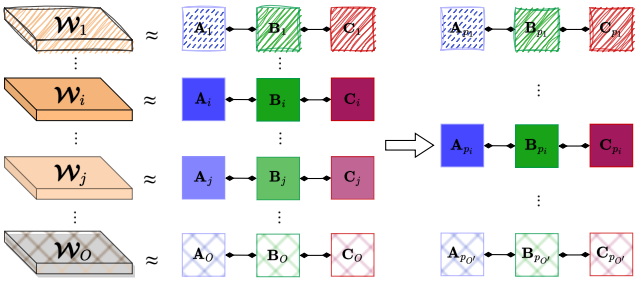


Fig. 4: The decomposition then pruning process for one layer.

Definition 3: For two linear subspaces $\boldsymbol{\mathcal{U}}, \boldsymbol{\mathcal{V}} \subset \mathbb{C}^I$, the smallest principal angle between them denoted by θ is defined as [35]:

$$\cos(\theta) = \max_{\mathbf{u} \in \boldsymbol{\mathcal{U}}, \mathbf{v} \in \boldsymbol{\mathcal{V}}} \frac{\mathbf{u}^H \mathbf{v}}{\|\mathbf{u}\|_2 \|\mathbf{v}\|_2}. \tag{3}$$

B. Filters Decomposition Using the CP Decomposition

Consider a convolutional layer with O filters of size $K_h \times K_w \times I$, where I represents the number of input channels, and K_h and K_w represent the height and width of the kernel, respectively. The weight tensor $\boldsymbol{\mathcal{W}}$ can be represented as a 4-order tensor of size $K_h \times K_w \times I \times O$. Alternatively, it can be viewed as a set of O individual 3-order filters denoted as $\{\boldsymbol{\mathcal{W}}_1, \boldsymbol{\mathcal{W}}_2, \ldots, \boldsymbol{\mathcal{W}}_O\}$. It is worth noting that the k-th filter can be obtained by extracting the sub-tensor $\boldsymbol{\mathcal{W}}_{:,:,:,k}$, which is equivalent to $\boldsymbol{\mathcal{W}}_k$. These weights map an input tensor $\boldsymbol{\mathcal{I}}$ of size $H_{in} \times W_{in} \times I$ into an output tensor $\boldsymbol{\mathcal{O}}$ of size $H_{out} \times W_{out} \times O$, where H_{in}, W_{in}, H_{out}, and W_{out} are the height and width of the input and output tensors, respectively. For simplicity, the dimension of the batch is disregarded.

In convolutional neural networks, the mapping of the input tensor $\boldsymbol{\mathcal{I}}$ to the output tensor $\boldsymbol{\mathcal{O}}$ is achieved through the convolution operation. This convolution is given by the following expression:

$$\boldsymbol{\mathcal{O}}_k(i,j) = \sum_{m=0}^{K_h-1} \sum_{n=0}^{K_w-1} \sum_{p=0}^{I-1} \boldsymbol{\mathcal{I}}(i+m, j+n, p) \cdot \boldsymbol{\mathcal{W}}_k(m,n,p), \tag{4}$$

where, for $0 \leq k \leq O-1$, $\boldsymbol{\mathcal{O}}_k = \boldsymbol{\mathcal{O}}_{:,:,k}$, and is of size $H_{out} \times W_{out}$. Based on (4) and the CPD definition in (2), we can apply the CPD to each individual filter $\boldsymbol{\mathcal{W}}_k$ in order to obtain a compact representation. By decomposing $\boldsymbol{\mathcal{W}}_k$ using CPD, we have

$$\boldsymbol{\mathcal{W}}_k(m,n,p) = \sum_{r=0}^{R-1} \mathbf{A}_k(m,r) \cdot \mathbf{B}_k(n,r) \cdot \mathbf{C}_k(p,r), \tag{5}$$

where $\mathbf{A}_k$, $\mathbf{B}_k$ and $\mathbf{C}_k$ are 3 factor matrices of size $K_h \times R$, $K_w \times R$ and $I \times R$, respectively. The complexity of this decomposition algorithm, using the alternating least squares algorithm, is $\mathcal{O}\left(R^2 \cdot I \cdot \max(K_h, K_w)\right)$ [21]. This approximation is graphically represented in the left half of Fig. 4.

By substituting (5) into (4), we obtain a new CPD-based approach to compute the convolution. This approach involves a sequence of mappings using the factor matrices instead of

high-order tensors. The resulting equation for this CPD-based convolution is:

$$\boldsymbol{\mathcal{O}}_k(i,j) = \sum_{m=0}^{K_h-1}\sum_{n=0}^{K_w-1}\sum_{p=0}^{I-1}\sum_{r=0}^{R-1}\boldsymbol{\mathcal{I}}(i+m,j+n,p)\cdot \mathbf{A}_k(m,r)\cdot\mathbf{B}_k(n,r)\cdot\mathbf{C}_k(p,r). \tag{6}$$

Starting from (6), we observe that the CPD-based convolution involves element-wise multiplications between the input tensor $\boldsymbol{\mathcal{I}}$ and the factor matrices $\mathbf{A}_k$, $\mathbf{B}_k$, and $\mathbf{C}_k$. It is important to note that the order of the convolutions can be rearranged without affecting the final result. This flexibility allows us to describe the computation as a sequential block of convolutions with smaller kernels, followed by a summation. First, the input tensor $\boldsymbol{\mathcal{I}}$ can be convolved with the kernel $\mathbf{C}_k$ along the input channel dimension. The resulting tensor is then convolved with the kernel $\mathbf{B}_k$ along the spatial dimensions. Finally, the output of the second convolution is convolved with the kernel $\mathbf{A}_k$ along the spatial dimensions. The outputs of these convolutions are summed up to compute the final output tensor $\boldsymbol{\mathcal{O}}_k$. The following set of equations captures these operations in a concise manner:

$$\boldsymbol{\mathcal{O}}^{\mathbf{C}}_k(i+m,j+n,r) = \sum_{p=0}^{I-1}\boldsymbol{\mathcal{I}}(i+m,j+n,p)\cdot\mathbf{C}_k(p,r), \tag{7}$$

where $\boldsymbol{\mathcal{O}}^{\mathbf{C}}_k$ is of size $H_{in}\times W_{in}\times R$.

$$\boldsymbol{\mathcal{O}}^{\mathbf{B}}_k(i+m,j,r) = \sum_{n=0}^{K_w-1}\boldsymbol{\mathcal{O}}^{\mathbf{C}}_k(i+m,j+n,r)\cdot\mathbf{B}_k(n,r), \tag{8}$$

where $\boldsymbol{\mathcal{O}}^{\mathbf{B}}_k$ is of size $H_{in}\times W_{out}\times R$.

$$\boldsymbol{\mathcal{O}}^{\mathbf{A}}_k(i,j,r) = \sum_{m=0}^{K_h-1}\boldsymbol{\mathcal{O}}^{\mathbf{B}}_k(i+m,j,r)\cdot\mathbf{A}_k(m,r), \tag{9}$$

where $\boldsymbol{\mathcal{O}}^{\mathbf{A}}_k$ is of size $H_{out}\times W_{out}\times R$. Finally, we have

$$\boldsymbol{\mathcal{O}}_k(i,j) = \sum_{r=0}^{R-1}\boldsymbol{\mathcal{O}}^{\mathbf{A}}_k(i,j,r). \tag{10}$$

One should note that equations (7), (8), and (9) can be seen as convolutions and can be easily implemented using common deep learning frameworks. Specifically, equation (7) can be computed using a classical 2D convolution operation, while equations (8) and (9) can be computed via group convolutions. One can refer to Fig. 5, which illustrates the structure of the CPDBlock. To ensure compatibility with classical frameworks, certain basic preprocessing operations including reshaping and mode permutations are required for adapting the kernel. Specifically, for the O factors $\mathbf{C}_k$ of dimensions $I\times R$, they need to be reshaped into a kernel of size $1\times1\times I\times(R\cdot O)$. This enables the application of a classical 2D convolution for equation (7). Additionally, for the group convolutions in equations (8) and (9), the kernels should be remodeled as $1\times K_w\times1\times(R\cdot O)$ and $K_h\times1\times1\times(R\cdot O)$, respectively. These modifications ensure the appropriate computation of $\boldsymbol{\mathcal{O}}^{\mathbf{B}}$ and $\boldsymbol{\mathcal{O}}^{\mathbf{A}}$. By performing these preprocessing steps, the CPD-based convolutions can be seamlessly integrated into existing deep learning frameworks. Note that in Fig. 5, the batch size B is depicted, and the O tensors $\boldsymbol{\mathcal{O}}^{\mathbf{C}}_k$, $\boldsymbol{\mathcal{O}}^{\mathbf{B}}_k$, and $\boldsymbol{\mathcal{O}}^{\mathbf{A}}_k$ are computed in a single operation, resulting in the corresponding output tensors $\boldsymbol{\mathcal{O}}^{\mathbf{C}}$, $\boldsymbol{\mathcal{O}}^{\mathbf{B}}$, and $\boldsymbol{\mathcal{O}}^{\mathbf{A}}$.

About rank and the compression. The choice of rank R in CPD plays a crucial role in balancing model compression and accuracy. Kruskal's theory provides a weak upper bound on the maximum rank [21], expressed as:

$$R \leq \min\{I\cdot K_h, I\cdot K_w, K_h\cdot K_w\}. \tag{11}$$

This upper bound offers a guideline for selecting an appropriate rank, ensuring a reasonable trade-off between model compression and preservation of critical features. By fixing a rank R, the CPDBlock achieves a significant reduction in the number of parameters compared to the original layer, which consists of $O\cdot I\cdot K_h\cdot K_w$ parameters. Through the CPD-based approach, NORTON achieves a compact representation of the filters, resulting in a reduced parameter count on the order of $O\cdot R\cdot(I+K_h+K_w)$. The computational complexity of the original layer is $\mathcal{O}(I\cdot O\cdot K_h\cdot K_w\cdot H\cdot W)$. In contrast, the three sub-layers exhibit complexities of $\mathcal{O}(I\cdot O\cdot R\cdot H\cdot W)$, $\mathcal{O}(O\cdot R\cdot K_w\cdot H\cdot W)$, and $\mathcal{O}(O\cdot R\cdot K_h\cdot H\cdot W)$, respectively. As a result, the CPDBlock significantly reduces the computational load to $\mathcal{O}(R\cdot O\cdot(I+K_h+K_w)\cdot H\cdot W)$. It's important to note that the rank constraint introduced in (11) guarantees a favorable reduction in both the number of parameters and computational complexity.

C. CPDBlock Pruning

After demonstrating the use of filter decomposition and its integration into the new architecture, we will now delve into the methodology of factor matrices pruning. This technique focuses on reducing the number of parameters by pruning the factor matrices obtained through the decomposition process.

First, it is important to note that each output $\boldsymbol{\mathcal{O}}_k$ is computed based on three factor matrices: $\mathbf{A}_k$, $\mathbf{B}_k$, and $\mathbf{C}_k$. Therefore, when pruning the kernel related to a specific output, all three matrices must be considered for removal. This necessitates the use of a pruning criterion that takes into account the interdependencies among the three matrices. Second, it is worth mentioning that when the CP decomposition is unique, it is unique up to scaling and permutation ambiguities. In other words, if two 3-order filters, $\boldsymbol{\mathcal{W}}_i$ and $\boldsymbol{\mathcal{W}}_j$, are strictly similar and satisfy the uniqueness conditions of the CP decomposition [21], then

$$\begin{cases}\boldsymbol{\mathcal{W}}_i = & [\![\mathbf{A}_i,\mathbf{B}_i,\mathbf{C}_i]\!],\\ \boldsymbol{\mathcal{W}}_j = & [\![\mathbf{A}_j,\mathbf{B}_j,\mathbf{C}_j]\!],\\ \boldsymbol{\mathcal{W}}_i = & \boldsymbol{\mathcal{W}}_j.\end{cases} \not\Rightarrow \begin{cases}\mathbf{A}_i=\mathbf{A}_j,\\ \mathbf{B}_i=\mathbf{B}_j,\\ \mathbf{C}_i=\mathbf{C}_j.\end{cases} \tag{12}$$

Instead, we have

$$\mathbf{A}_i=\mathbf{A}_j\boldsymbol{\Pi}\boldsymbol{\Lambda}^A,\ \mathbf{B}_i=\mathbf{B}_j\boldsymbol{\Pi}\boldsymbol{\Lambda}^B,\ \text{and}\ \mathbf{C}_i=\mathbf{C}_j\boldsymbol{\Pi}\boldsymbol{\Lambda}^C,$$

where $\boldsymbol{\Pi}$ is a permutation matrix, and the diagonal scaling matrices verify $\boldsymbol{\Lambda}^A\boldsymbol{\Lambda}^B\boldsymbol{\Lambda}^C=\mathbf{I}$. For all the aforementioned reasons, we have decided to choose the PABS [35] as a metric to measure the distance between two CP decompositions. The use of the PABS distance measure is justified in both the unique and non-unique cases of CP decomposition. In the

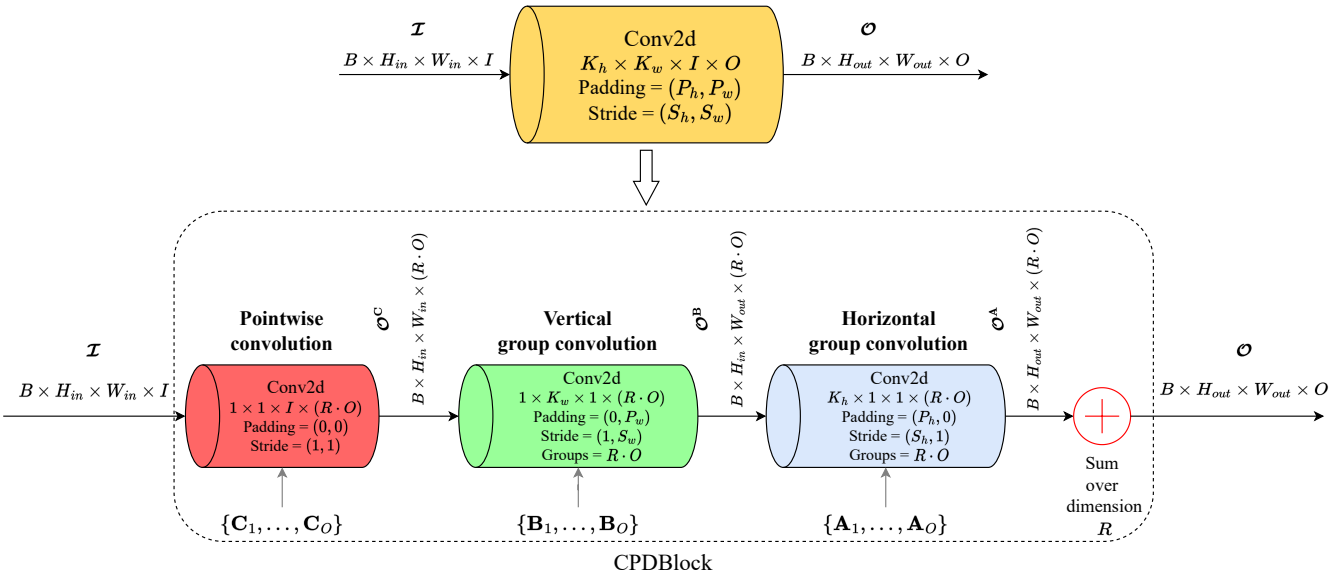


Fig. 5: Illustration of the CPDBlock structure for classical deep learning frameworks.

unique case, PABS allows for capturing distances between factor matrices, enabling the identification of redundant filters based on their distance patterns, while handling scaling and permutation ambiguities. Let $\phi(.,.)$ be a function that computes the PABS between two factor matrices, as defined in (3). If we reconsider the example in (12) in the case of unique CPDs, we have

$$\begin{cases} \mathcal{W}_i = & [\![\mathbf{A}_i, \mathbf{B}_i, \mathbf{C}_i]\!], \\ \mathcal{W}_j = & [\![\mathbf{A}_j, \mathbf{B}_j, \mathbf{C}_j]\!], \\ \mathcal{W}_i = & \mathcal{W}_j. \end{cases} \Rightarrow \begin{cases} \phi(\mathbf{A}_i, \mathbf{A}_j) = 0, \\ \phi(\mathbf{B}_i, \mathbf{B}_j) = 0, \\ \phi(\mathbf{C}_i, \mathbf{C}_j) = 0. \end{cases} \tag{13}$$

Even in the non-unique cases, PABS can still be effective in identifying redundancies, as it captures the distance between different sets of factor matrices representing the same tensor as will be confirmed in the simulations. This enables the pruning process to remove filters that contribute minimally to model performance or exhibit high similarity to other filters, resulting in a more compact model while preserving critical features and maintaining performance.

The core idea of CPDBlock pruning is to construct a distance matrix $\mathbf{D}$, where each element $\mathbf{D}_{ij}$ corresponds to the distance between the factor matrices $[\![\mathbf{A}_i, \mathbf{B}_i, \mathbf{C}_i]\!]$ and $[\![\mathbf{A}_j, \mathbf{B}_j, \mathbf{C}j]\!]$. The pruning process involves iteratively identifying the pair of decompositions, i and j, corresponding to the minimum value of $\mathbf{D}_{ij}$, and removing one of them. The removed decomposition in our strategy between i and j is the one that most closely resembles the rest of the decompositions, ensuring that the pruned one retains the representation that is most similar to the remaining ones. The distance matrix $\mathbf{D} \in \mathbb{R}^{O \times O}$ can be expressed as

$$\mathbf{D}_{ij} = \alpha \mathbf{D}^{\mathbf{A}}_{ij} + \beta \mathbf{D}^{\mathbf{B}}_{ij} + \gamma \mathbf{D}^{\mathbf{C}}_{ij}, \tag{14}$$

where $\mathbf{D}^{A}_{ij} = \phi(\mathbf{A}_i, \mathbf{A}_j)$ (similarly for $\mathbf{D}^{B}_{ij}$ and $\mathbf{D}^{C}_{ij}$), and α, β and γ are weight parameters, whose sum is equal to 1. The pruning strategy is outlined in Alg. 1, which is complemented by a visual representation in the right half of Fig. 4. The complexity of the CPDBlock Pruning algorithm is $\mathcal{O}\left(O^2 \cdot (O - O')\right)$, resulting from pairwise comparisons in the calculation of the distance matrix and subsequent iterations for factor deletion.

Algorithm 1 CPDBlock Pruning

Require: The decompositions of O filters $\{\mathbf{A}_1, \mathbf{B}_1, \mathbf{C}_1\}, \ldots, \{\mathbf{A}_O, \mathbf{B}_O, \mathbf{C}_O\}$ and the number of filters after pruning O'.
Ensure: Selected factors $\{\mathbf{A}_{p_1}, \mathbf{B}_{p_1}, \mathbf{C}_{p_1}\}, \ldots, \{\mathbf{A}_{p_{O'}}, \mathbf{B}_{p_{O'}}, \mathbf{C}_{p_{O'}}\}$.

```
Compute distance matrix D following (14).
for t = 1 to O − O' do
    Find the shortest distance: (i, j) = argmin_{(x,y), x≠y} D_{x,y}
    if Σ_{k=1, k≠i}^{O} D_{i,k} ≤ Σ_{k=1, k≠j}^{O} D_{j,k} then
        Delete factors of decomposition i.
    else
        Delete factors of decomposition j.
    end if
    Delete the row and column of the deleted
      decomposition from D.
end for
```

IV. Experiments

A. Experimental Settings

Architectures and Datasets. In order to demonstrate the adaptability of NORTON, we assess three representative architectures: VGG-16-BN [59], ResNet-56/110 with residual blocks [60], and DenseNet-40 with dense blocks [61]. These models are tested on the CIFAR-10 dataset [62]. Additionally, to validate the scalability of NORTON, experiments

are conducted on the challenging ImageNet dataset [63] using the ResNet-50 architecture. Furthermore, the compressed ResNet-50 model is employed as the backbone network for FasterRCNN-FPN [64], MaskRCNN, and KeypointRCNN [65] on the COCO-2017 dataset [66].

Baselines. NORTON is compared with 52 SOTAs in the fields of low-rank decompositions (D) [1], [2], [10], [12], [13], [17], [20], [39], [40], structured pruning (P) [3], [9], [25]–[34], [45]–[57], [67]–[78] and hybrid methods (H) [4]–[6], [8], [24], [58]. However, for certain methods where corresponding experiments were unavailable, we reproduced the approach of [1], [2], [4], [12] to ensure a more comprehensive comparison. To ensure fairness in the comparison, all original models are identical.

Evaluation Protocols. The model is assessed via three dimensions: accuracy, required Multiply Accumulate Operations (MACs), and the number of parameters (Params). The compression ratio (CR) is defined as the percentage reduction in MACs/Params when compared to the original model. Concerning the performance, top-1/top-5 accuracy is used on classification tasks while mean average precision (AP) and recall (AR) are used on detection/segmentation tasks.

Configurations. On CIFAR, fine-tuning was carried out for 400 epochs with a batch size of 256, momentum of 0.9, weight decay of 0.0005, and an initial learning rate of 0.01. On ImageNet, fine-tuning was conducted for 200 epochs with a batch size of 128, a momentum of 0.99, a weight decay of 0.0001, and a cosine learning rate scheduler with an initial learning rate of 0.1. On COCO, models are fine-tuned following the default recipe of torchvision [79]. In this work, we choose $\alpha = \beta = \gamma = \frac{1}{3}$.

B. Results and Analysis

In the ensuing simulations, we adjusted the rank and pruning ratios to match other methods for a comparable compression rate. This enables us to examine the trade-off between CR and accuracy, as well as vice versa. For further transparency and accessibility, our accompanying code includes detailed information regarding the rank and pruning rates used in each case, ensuring the reproducibility and comprehensiveness of our results. We delve deeper into the impact of varying rank and pruning ratio selections in Section V.

VGG-16-BN. Tab. I shows compression results of VGG on CIFAR-10. In all compression levels, compared with other methods, NORTON consistently achieves the highest accuracy while reducing much more computation costs and enjoying a similar number of parameters. In particular, NORTON proves to be very robust with high compression rates. It can reduce 88% of FLOPs and 87% parameters with almost no loss, just 0.12%; or 96% MACs and 98% parameters with a slight loss of 1.42%. Finally, to evaluate the resistance of NORTON, we subject it to an ultra-high compression rate of 99% reduction in both MACs and parameters, and the proposed method remains firm, yielding a modest loss.

ResNet-56/110. Tab. II presents results on the ResNet-56/110 architectures. Our approach has the capacity to generalize the original model as it boosts 1.11% and 1.35% of the accuracy while eliminating 37% and 38% of parameters for ResNet-56 and ResNet-110, respectively. NORTON proves to be efficient with high-level compression ($\approx 90\%$) where it outperforms [73] in all aspects. For ResNet-110, NORTON is better than DECORE [30] in every way.

TABLE I: Compression results of VGG-16-BN on CIFAR

Method	Type	Top-1	MACs (CR)	Params (CR)
VGG-16-BN [59]		93.96	313.73M (00)	14.98M (00)
HRank-1 [34]	P	93.43	145.61M (54)	2.51M (83)
CHIP [32]	P	93.86	131.17M (58)	2.76M (82)
EZCrop [31]	P	93.01	131.17M (58)	2.76M (82)
DECORE-500 [30]	P	94.02	203.08M (35)	5.54M (63)
APIB [27]	P	94.08	127.03M (60)	3.60M (76)
AutoBot [28]	P	94.19	145.61M (54)	7.53M (50)
NORTON (Ours)	H	**94.45**	**126.49M (60)**	**2.58M (83)**
HRank-2 [34]	P	92.34	108.61M (65)	2.64M (82)
DECORE-200 [30]	P	93.56	110.51M (65)	1.66M (89)
Yeom *et al.* [25]	P	93.48	104.67M (66)	2.86M (81)
EZCrop [31]	P	93.70	104.78M (67)	2.50M (83)
CHIP [32]	P	93.72	104.78M (67)	2.50M (83)
APIB [27]	P	94.00	106.67M (66)	3.30M (78)
AutoBot [28]	P	94.01	108.71M (65)	6.44M (57)
NORTON (Ours)	H	**94.16**	**101.91M (68)**	**2.34M (84)**
RGP-64_16 [9]	P	92.76	78.78M (75)	3.81M (75)
WhiteBox [29]	P	93.47	75.80M (76)	N/A
AutoBot [28]	P	93.62	72.60M (77)	5.51M (63)
NORTON (Ours)	H	**94.11**	**74.14M (77)**	**3.60M (76)**
QSFM [76]	P	92.17	79.00M (75)	3.68M (75)
DECORE-100 [30]	P	92.44	51.20M (82)	0.51M (96)
FSM [74]	P	92.86	59.61M (81)	1.50M (90)
ALDS [20]	D	92.67	66.95M (86)	1.90M (96)
Dai *et al.* [5]	H	93.03	37.76M (87)	0.43M (97)
Lebedev *et al.* [1]	D	93.07	68.53M (78)	3.22M (78)
EPruner-0.73 [75]	P	93.08	74.42M (76)	1.65M (89)
HALOC [10]	D	93.16	43.92M (86)	0.30M (98)
CHIP [32]	P	93.18	66.95M (79)	1.90M (87)
ASTER [26]	P	93.45	60.00M (81)	N/A
EDP [8]	H	93.52	62.40M (80)	0.66M (96)
HTP-URC-2 [67]	P	93.62	73.32M (77)	0.87M (94)
DF [58]	H	93.64	94.00M (70)	1.47M (90)
FSM [74]	P	93.73	106.67M (66)	2.10M (86)
NORTON (Ours)	H	**93.84**	**37.68M (88)**	1.94M (87)
HRank-3 [34]	P	91.23	73.70M (77)	1.78M(92)
RGP-64_6 [9]	P	91.45	31.37M (90)	1.43M (90)
DECORE-50 [30]	P	91.68	36.85M (88)	0.26M (98)
NORTON (Ours)	H	**92.54**	**13.54M (96)**	**0.24M (98)**
NORTON (Ours)	H	**90.32**	**4.58M (99)**	**0.14M (99)**

DenseNet-40. Tab. III shows the compression results of DenseNet-40 on CIFAR-10. With similar performance, NORTON usually gains more compression than other methods. For example, with an accuracy of about 94.86%, our method reduces more than 7% MACs and 9% parameters in comparison with [30].

ResNet-50. To evaluate the scalability of NORTON, we perform experiments on the extensive ImageNet dataset using the ResNet-50 architecture, as enumerated in Tab. IV. Across all evaluated scenarios, NORTON consistently outperforms other approaches in terms of both performance and complexity reduction. Our method can reduce 50% MACs while still enjoying an accuracy increment of 1.88% compared to Hinge [24], a hybrid method. In comparison to other hybrid methods [6], [8], NORTON yields higher accuracy while requiring lower computational cost and fewer parameters.

TABLE II: Compression results of ResNet-56/110 on CIFAR

Method	Type	Top-1	MACs (CR)	Params (CR)
ResNet-56 [60]		93.26	127.09M (00)	0.85M (00)
HRank-1 [34]	P	93.52	88.72M (29)	0.71M (17)
DECORE-450 [30]	P	93.34	92.48M (26)	0.64M (24)
FilterSketch [73]	P	93.65	88.05M (30)	0.68M (21)
TPP [46]	P	93.81	86.59M (31)	N/A
WHC [47]	P	93.91	90.35M (28)	N/A
NORTON (Ours)	H	**94.37**	**83.11M (35)**	**0.54M (37)**
TRP [39]	D	92.77	55.60M (56)	N/A
HRank-2 [34]	P	93.17	62.72M (50)	0.49M (42)
FilterSketch [73]	P	93.19	73.36M (41)	0.50M (41)
SOKS-40% [71]	P	93.22	81.40M (36)	0.51M (40)
DECORE-200 [30]	P	93.26	62.93M (50)	0.43M (49)
TPP [46]	P	93.46	62.75M (50)	N/A
BSR [17]	D	93.53	55.70M (56)	0.37M (56)
MFP [72]	P	93.56	59.40M (53)	N/A
EDP [8]	H	93.61	53.07M (58)	0.39M (54)
FSM [74]	P	93.63	61.49M (51)	0.48M (44)
CC-0.5 [6]	H	93.64	60.00M (52)	0.44M (48)
ResRep [49]	P	93.71	59.30M (53)	N/A
DCP [77]	P	93.72	56.47M (55)	0.43M (50)
AutoBot [28]	P	93.76	55.82M (56)	0.46M (46)
DeGraph [70]	P	93.77	58.98M (53)	N/A
WHC [47]	P	93.80	74.04M (41)	N/A
EZCrop [31]	P	93.80	65.94M (47)	0.48M (43)
NORTON (Ours)	H	**94.00**	73.22M (42)	0.44M (48)
QSFM [76]	P	91.88	50.62M (60)	0.25M (71)
CHIP [32]	P	92.05	34.79M (72)	0.24M (72)
TPP [46]	P	92.35	36.39M (71)	N/A
BSR [17]	D	92.51	32.10M (74)	0.21M (75)
SOKS-55% [71]	P	93.08	61.30M (52)	0.39M (54)
CLR-RNF-0.56 [48]	P	93.27	54.00M (57)	0.38M (56)
Yang *et .al* [40]	D	93.27	33.88M (73)	N/A
APIB [27]	P	93.29	41.41M (67)	0.29M (66)
DF [58]	H	93.44	32.20M (75)	0.22M (74)
NORTON (Ours)	H	**93.81**	37.52M (71)	**0.21M (75)**
HRank-3 [34]	P	90.72	32.52M (74)	0.27M (68)
DECORE-55 [30]	P	90.85	23.22M (81)	0.13M (85)
FilterSketch [73]	P	91.20	32.47M (74)	0.24M (72)
APIB [27]	P	91.53	23.84M (81)	0.15M (83)
NORTON (Ours)	H	**91.62**	**14.47M (89)**	**0.08M (91)**
ResNet-110 [60]		93.50	256.04M (00)	1.73M (00)
DECORE-500 [30]	P	93.88	163.30M (35)	1.11M (36)
NORTON (Ours)	H	**94.85**	**163.00M (35)**	**1.08M (38)**
NNCS-61.7 [55]	P	93.41	99.86M (61)	0.66M (62)
DECORE-300 [30]	P	93.50	96.66M (62)	0.61M (65)
NORTON (Ours)	H	**94.11**	**92.99M (64)**	**0.59M (65)**
DECORE-175 [30]	P	92.71	58.37M (77)	0.35M (80)
NORTON (Ours)	H	**92.77**	**47.34M (82)**	**0.30M (83)**

TABLE III: Compression results of DenseNet-40 on CIFAR

Method	Type	Top-1	MACs (CR)	Params (CR)
DenseNet-40 [61]		94.81	282.92M (00)	1.04M (00)
DECORE-175 [30]	P	94.85	228.96M (19)	0.83M (21)
NORTON (Ours)	H	**94.86**	**213.58M (26)**	**0.74M (30)**
HRank-1 [34]	P	94.24	167.41M (41)	0.66M (37)
DECORE-115 [30]	P	94.59	171.36M (39)	0.56M (46)
Yeom *et al.* [25]	P	94.62	167.41M (41)	0.66M (37)
AutoBot [28]	P	94.67	167.64M (42)	0.76M (28)
NORTON (Ours)	H	**94.67**	**168.23M (42)**	0.58M (45)
HRank-2 [34]	P	93.68	110.15M (61)	0.48M (54)
EZCrop [31]	P	93.76	113.08M (60)	0.39M (62)
DECORE-70 [30]	P	94.04	128.13M (55)	0.37M (65)
HT2-0.8 [13]	D	94.06	104.31M (63)	0.34M (67)
NORTON (Ours)	H	**94.17**	103.68M (64)	0.33M (69)

TABLE IV: Compression results of ResNet-50 on ImageNet

Method	Type	Top-1	Top-5	MACs (CR)	Params (CR)
ResNet-50 [60]		76.15	92.87	4.09G (00)	25.50M (00)
ABCPruner-100% [56]	P	72.84	92.97	2.56G (37)	18.02M (29)
CLR-RNF-0.2 [48]	P	74.85	92.31	2.45G (40)	16.92M (34)
EPruner-0.81 [75]	P	74.95	92.36	2.37G (42)	N/A
FilterSketch-0.7 [73]	P	75.22	92.41	2.64G (36)	16.95M (33)
Kim *et al.* [12]	D	75.34	92.68	N/A	17.60M (31)
PFP [53]	P	75.91	92.81	3.65G (11)	20.88M (18)
C-SGD-70 [68]	P	75.94	92.88	2.62G (36)	17.09M (33)
LeGR [50]	P	76.20	93.00	2.99G (27)	N/A
DECORE-8 [30]	P	76.31	93.02	3.54G (13)	22.69M (11)
CHIP [32]	P	76.30	93.02	2.26G (44)	15.10M (41)
TPP [46]	P	76.44	N/A	2.74G (33)	N/A
NORTON (Ours)	H	**76.91**	**93.57**	**2.32G (43)**	**14.51M (43)**
FilterSketch-0.6 [73]	P	74.68	92.17	2.23G (46)	14.53M (43)
Hinge [24]	H	74.70	N/A	2.17G (47)	N/A
HRank-1 [34]	P	74.98	92.33	2.30G (44)	16.15M (37)
DECORE-6 [30]	P	74.58	92.18	2.36G (42)	14.10M (45)
PFP [53]	P	75.21	92.43	2.29G (44)	17.82M (30)
RGP-64_36 [9]	P	75.30	92.55	2.30G (44)	14.34M (44)
WhiteBox [29]	P	75.32	92.43	2.22G (46)	N/A
MFP [72]	P	75.67	92.81	2.37G (42)	N/A
EZCrop [31]	P	75.68	92.70	2.26G (45)	15.09M (41)
LeGR [50]	P	75.70	92.70	2.37G (42)	N/A
C-SGD-60 [68]	P	75.80	92.65	2.19G (47)	14.58M (43)
DepGraph [70]	P	75.83	N/A	2.09G (49)	N/A
SCOP [51]	P	75.95	92.79	2.24G (45)	14.59M (43)
CATRO [52]	P	75.98	92.79	2.21G (46)	N/A
WHC [47]	P	76.06	92.86	2.37G (42)	N/A
CHIP [32]	P	76.15	92.91	2.10G (49)	14.23M (44)
DNCP [54]	P	76.30	N/A	2.20G (46)	N/A
NORTON (Ours)	H	**76.58**	**93.43**	**2.08G (50)**	**13.51M (47)**
HRank-2 [34]	P	71.98	91.01	1.55G (62)	13.77M (46)
TRP [39]	D	72.86	91.58	1.78G (56)	N/A
FilterSketch-0.4 [73]	P	73.04	91.18	1.51G (63)	10.40M (59)
WhiteBox [29]	P	74.21	92.01	1.50G (63)	N/A
EZCrop [31]	P	74.33	92.00	1.52G (63)	11.05M (57)
DAIS [69]	P	74.45	92.21	1.83G (55)	N/A
CC-0.6 [6]	H	74.54	92.25	1.53G (63)	10.58M (59)
RGP-64_30 [9]	P	74.58	92.09	1.92G (53)	11.99M (53)
Phan *et al.* [2]	D	74.68	92.16	1.56G (62)	N/A
MFP [72]	P	74.86	92.43	1.88G (54)	N/A
TPP [46]	P	75.12	N/A	1.60G (61)	N/A
Yeom *et al.* [25]	P	75.25	92.49	1.52G (63)	11.05M (57)
SCOP [51]	P	75.26	92.53	1.86G (55)	12.29M (52)
CHIP [32]	P	75.26	92.53	1.52G (63)	11.04M (57)
ASTER [26]	P	75.27	92.47	1.51G (63)	N/A
C-SGD-60 [68]	P	75.29	92.39	1.82G (55)	12.37M (52)
LeGR [50]	P	75.30	92.40	1.92G (53)	N/A
ResRep [49]	P	75.30	92.47	1.52G (62)	N/A
WHC [47]	P	75.33	92.52	1.88G (54)	N/A
EDP [8]	H	75.34	92.43	1.92G (53)	14.28M (44)
APIB [27]	P	75.37	N/A	1.56G (62)	10.71M (58)
HTP-URC [67]	P	75.81	N/A	1.88G (54)	15.81M (38)
NORTON (Ours)	H	**75.95**	**92.91**	**1.49G (64)**	**10.52M (59)**
HRank-3 [34]	P	69.10	89.58	0.98G (76)	8.27M (68)
DECORE-5 [30]	P	72.06	90.82	1.60G (61)	8.87M (65)
Yeom *et al.* [25]	P	72.28	90.93	0.95G (77)	8.02M (67)
ABCPruner-50% [56]	P	72.58	90.91	1.30G (68)	9.10M (64)
CHIP [32]	P	72.30	90.74	0.95G (77)	8.01M (69)
CLR-RNF-0.44 [48]	P	72.67	91.09	1.23G (70)	9.00M (65)
EPruner-0.81 [75]	P	72.73	91.01	1.29G (68)	N/A
NORTON (Ours)	H	**74.00**	**92.00**	**0.96G (77)**	**7.96M (69)**
FilterSketch-0.2 [73]	P	69.43	89.23	0.93G (77)	7.18M (72)
DECORE-4 [30]	P	69.71	89.37	1.19G (71)	6.12M (76)
RGP-64_16 [9]	P	72.68	91.06	1.02G (75)	6.38M (75)
CURL [78]	P	73.39	91.46	1.11G (73)	6.67M (74)
NORTON (Ours)	H	**73.65**	**91.64**	**0.92G (78)**	**5.88M (77)**

Faster/Mask/Keypoint-RCNN. To assess NORTON's efficacy in downstream tasks, we employed our compressed ResNet-50/Imagenet as the backbone for training Faster/Mask/Keypoint-RCNN on COCO, as detailed in Tab. V. Our method enhances both precision and recall relative to the baseline, achieving a remarkable reduction of 17% MACs and at least 19% parameters. Notably, NORTON substantially elevates inference throughput, delivering over a 2× FPS im-

TABLE V: Compression results of Faster/Mask/Keypoint-RCNN-ResNet50-FPN on COCO-2017

Model	$AP^{0.5:0.95}$	$AP^{0.5}$	$AP^{0.75}$	AR^{1}	AR^{10}	AR^{100}	MACs (CR)	Params (CR)	FPS	Latency(ms)
FasterRCNN [64], [79]	0.37	0.58	0.39	0.31	0.48	0.51	134.85G (00)	41.81M (00)	12	85
NORTON (Ours)	**0.38**	**0.59**	**0.42**	**0.32**	**0.50**	**0.52**	111.47G (17)	30.72M (27)	19	53
NORTON (Ours)	0.32	0.52	0.34	0.29	0.46	0.48	**93.39G (31)**	**22.01M (47)**	**25**	**41**
MaskRCNN [65], [79]	0.34	0.55	0.36	0.29	0.45	0.47	134.85G (00)	44.46M (00)	9	111
NORTON (Ours)	**0.35**	**0.57**	**0.37**	**0.30**	**0.46**	**0.48**	111.47G (17)	33.36M (25)	14	73
NORTON (Ours)	0.32	0.52	0.33	0.28	0.44	0.46	**93.39G (31)**	**24.65M (45)**	**20**	**50**
				$AR^{0.5:0.95}$	$AR^{0.5}$	$AR^{0.75}$				
KeypointRCNN [65], [79]	0.65	0.86	0.71	0.71	0.90	0.77	137.42G (00)	59.19M (00)	8	125
NORTON (Ours)	**0.65**	**0.86**	**0.71**	**0.72**	**0.91**	**0.77**	114.04G (17)	48.10M (19)	13	76
NORTON (Ours)	0.63	0.85	0.69	0.69	0.90	0.75	**95.97G (30)**	**39.39M (34)**	**17**	**59**

provement compared to the baseline models. For instance, FasterRCNN sees the end-to-end latency drop from 85 ms to 41 ms, reaching a real-time framerate of 25 FPS. It's essential to highlight that these performance assessments were conducted on an RTX 3060 GPU, providing robust evidence of our approach's real-world utility in complex tasks of computer vision based on NORTON's network compression.

V. Discussions

A. The Role of the Rank

As the rank directly relates to the approximation error and the gain of compression rate, we conduct additional experiments on VGG16/CIFAR10 to investigate the effect of the rank selection. First, we only apply the CPD testing all possible ranks (see Tab. VI). To measure the approximation error between the original model and the decomposed model, we employ the normalized mean square error (NMSE) defined as:

$$\text{NMSE} = \frac{1}{\sum_{i=1}^{N} O_i} \cdot \sum_{i=1}^{N} \sum_{j=1}^{O_i} \frac{\left\| \mathcal{W}_i^j - \hat{\mathcal{W}}_i^j \right\|_F^2}{\left\| \mathcal{W}_i^j \right\|_F^2}, \tag{15}$$

where $\mathcal{W}_i^j$ is the j-th 3-order filter in the i-th layer composed of O_i filters, $\hat{\mathcal{W}}_i^j$ is its approximation, and N is the number of layers.

TABLE VI: Complexity reduction, approximation error, and accuracy with and without fine-tuning with respect to the rank

Rank	MACs CR	Params CR	NMSE	Accuracy (%)	
				Without FT	With FT
1	88.03	87.06	0.6265	10.00	93.84
2	76.44	75.98	0.4114	10.00	94.11
3	64.85	64.91	0.2760	68.37	94.18
4	53.27	53.84	0.1837	88.30	94.07
5	41.69	42.76	0.1173	92.44	94.22
6	30.10	31.69	0.0698	93.45	94.15
7	18.51	20.61	0.0372	93.90	94.11
8	6.93	9.54	0.0137	94.03	94.03

Tab. VI shows the complexity reduction, approximation error, and accuracy before and after fine-tuning. Obviously, without fine-tuning, the higher the rank is, the better the weights are approximated but with less compression obtained. With $5 \leq R$, our filter decomposition method produces comparable accuracies (maximum 1.46% drop) with the original model without fine-tuning. This suggests that our decomposition step behaves well without fine-tuning. In addition, we demonstrate that our decomposition step can effectively work when followed by a fine-tuning step. The results show that after fine-tuning, the accuracy is completely restored for all cases. However, it should be noted that the achieved compression ratios are not as favorable as those obtained with the proposed hybrid strategy.

Notably, our proposed method has not met the degeneracy problem (*i.e.,* instability issue when training a CNN with decomposed layers in the CP format) as in a reshaped decomposition approach [2]. We suspect that the filter decomposition method is more fine-grained than the reshaped one. The trend of NMSE demonstrates a strong correlation between the approximation error and the accuracy (without fine-tuning). This observation aligns with recent work [80].

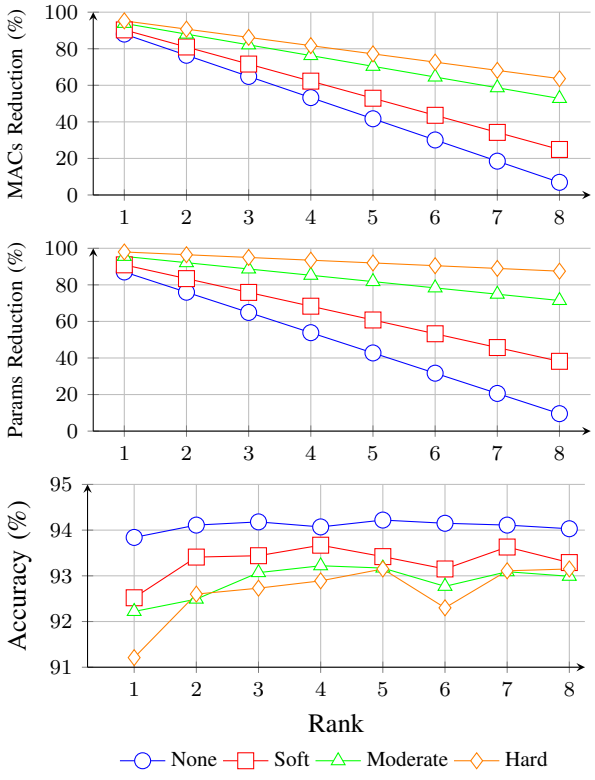


Fig. 6: Comparison of complexity reduction and accuracy with respect to the rank and the pruning ratio.

B. The Effect of Rank and Pruning Ratio Selection

To evaluate the effect of pruning ratios and rank on compression and accuracy, we conducted additional simulations depicted in Fig. 6. These simulations involve three compression levels, namely None (N), Soft (S), Moderate (M), and Hard (H) pruning ratios. Each level represents a different degree of pruning, allowing us to examine the impact of different compression settings on the decomposed models.

These experiments provide a comprehensive understanding of how the selection of rank and pruning ratio impacts complexity reduction and model performance. With a predefined complexity reduction goal, there are multiple choices for the rank and pruning ratio, resulting in different outcomes. For example, to eliminate about 65% of MACs, as shown in Fig. 6, possible choices are 3N, 4S, 6M, and 8H which produce 94.18%, 93.67%, 92.77%, and 93.15% accuracy after fine-tuning, respectively. Notably, these combinations, despite similar MACs reductions, lead to different levels of parameter reduction: 64.91%, 68.33%, 78.34%, and 87.49%, respectively. This observation raises questions about the optimal combination of rank and pruning ratio, warranting further investigation, particularly for hybrid compression approaches.

C. Component Analysis

Given that NORTON operates as a hybrid method, a detailed examination of the efficacy of its individual components becomes imperative. We conducted an experiment on the VGG-16-BN/CIFAR-10, focusing on the accuracy vs. MACs reduction Pareto curves of six approaches. These approaches were organized into three direct comparisons: our decomposition part versus Tucker decomposition [12], our pruning part versus Taylor pruning [3], and our hybrid approach versus the combination of Taylor pruning and TD [4]. It's worth noting that we selected these approaches based on their relevance to our work, especially [4], as detailed in subsection II-C, and their components (TD and Taylor pruning) being representative in network compression. Due to the absence of results for this architecture in the referenced papers, we reproduced the results for Taylor pruning [3], TD [12], and their combination [4]. The outcomes are presented in Fig. 7.

Effectiveness: In terms of pairwise comparisons, our components consistently outperform their counterparts from [4]. For instance, our CPD consistently achieves higher performance than TD at similar compression ratios. Additionally, CPD demonstrates greater stability than TD, meaning its accuracy does not drop significantly as the CR increases (up to 88%, see Tab. VI). TD, a layer decomposition approach, experiences a rapid loss of performance beyond a 50% compression ratio, suggesting potential information loss due to its neglect of spatial dimensions of the kernel (3rd and 4th modes). This comparison underscores the effectiveness of our proposed filter decomposition method compared to layer decomposition and reshaped-based decomposition approaches. The subspace-based pruning method proves more effective and efficient than the Taylor pruning method. However, both pruning methods experience a significant performance drop at a 70% compression ratio threshold. Notably, the combination of Taylor pruning and TD exhibits suboptimal performance, even worse than its individual components at many points. In contrast, NORTON effectively synthesizes the strengths of its components, yielding higher accuracy at various points and achieving higher compression (beyond 90%). This experiment not only strengthens the motivation of our hybrid approach but also reaffirms that our decomposition/pruning proposal can stand alone as a promising candidate for network compression.

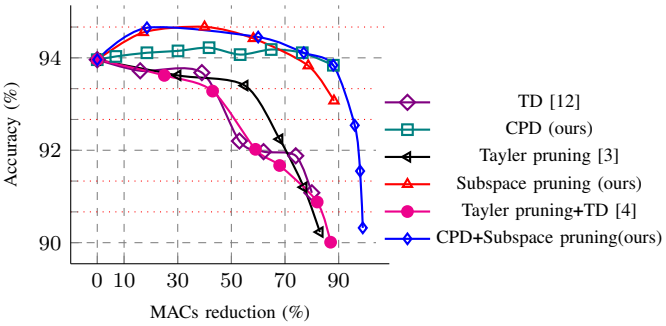


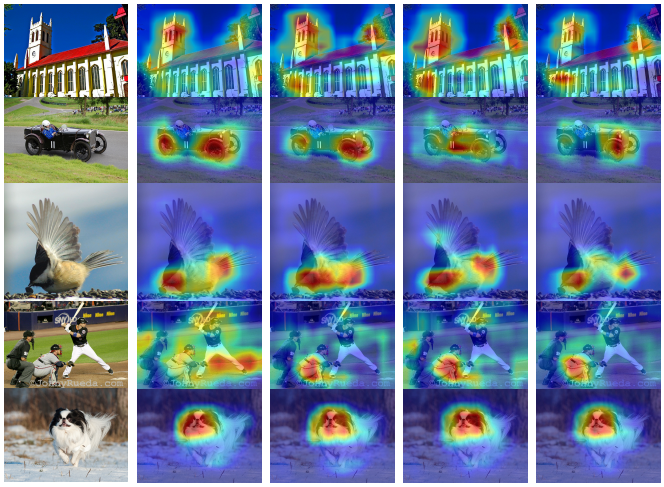
Fig. 7: The accuracy-MACs reduction Pareto curves of individual components in hybrid approaches.

Efficiency: Theoretical computational complexities for the proposed CPD and subspace pruning were detailed in Section III. Notably, the overheads associated with both the decomposition and pruning stages are deemed insignificant, each requiring less than 10 seconds in our simulation. This is notably less than the time spent on the fine-tuning step, which can extend up to 100 hours when using ResNet50/Imagenet. Additionally, our choice of a single-shot compression scheme ensures that the compression process occurs only once. In contrast, the hybrid method in [4] employs an iterative scheme, incurring higher costs as the decomposition and filters' saliency estimation need recalculation in each round. It's essential to emphasize that the critical metric lies in the inference speed during the deployment of the compressed model, aptly represented by the reduction in MACs, where our approach has proven to be particularly efficient.

This comprehensive evaluation shows that NORTON, as a hybrid approach, excels not only in synthesizing the strengths of its individual components but also in outperforming comparable methods in terms of accuracy and compression.

D. Visualizing Feature Preservation

We qualitatively assess feature preservation in NORTON, complementing the established efficiency from numerical results in Section IV. We randomly selected 5 images from the ImageNet validation dataset and evaluated three compression levels for the original ResNet-50 model: 50%, 64%, and 78% (see Table IV). Using GradCAM [81] for interpretation, we visualize and analyze feature maps in the original and compressed models in Fig. 8. It can be observed that at different CRs, NORTON consistently proves robust in capturing and preserving essential features across diverse classes. This robustness implies sustained effectiveness and reliability across varying scenarios and CR, establishing NORTON as a versatile choice for network compression.



(a) Input (b) CR = 0% (c) CR = 50% (d) CR = 64% (e) CR = 78%

Fig. 8: Qualitative assessment of feature preservation in compressed models.

VI. Conclusion

In this work, we have introduced NORTON, a novel hybrid network compression method that combines the strengths of tensor decompositions and structured pruning. Through extensive experiments and analyses, we have demonstrated the effectiveness and superiority of NORTON in achieving significant model compression while preserving critical features and maintaining performance. By leveraging filter decomposition and the integration of structured pruning, NORTON offers a powerful approach for reducing model complexity and the number of parameters, pushing the limits of network compression. For instance, our method achieved a simultaneous MACs/Params CR of 99% since current SOTAs only report to 91% to the best of our knowledge. Our results have shown that NORTON outperforms existing SOTA compression techniques in terms of compression ratios and accuracy retention. The combination of tensor decompositions and structured pruning in NORTON provides a versatile framework that allows for fine-grained control over the compression process. Moreover, NORTON's ability to handle scaling and permutation ambiguities further enhances its practicality and flexibility. The evaluation of NORTON on various architectures and datasets has demonstrated its scalability and generalizability. Furthermore, we have analyzed the impact of rank and pruning ratios on the compression and performance trade-offs, providing valuable insights for selecting optimal configurations.

References

[1] V. Lebedev, Y. Ganin, M. Rakhuba, I. V. Oseledets, and V. S. Lempitsky, "Speeding-up convolutional neural networks using fine-tuned cp-decomposition," in *ICLR*, 2015.

[2] A. Phan, K. Sobolev, K. Sozykin, D. Ermilov, J. Gusak, P. Tichavský, V. Glukhov, I. Oseledets, and A. Cichocki, "Stable low-rank tensor decomposition for compression of convolutional neural network," in *ECCV*, 2020.

[3] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz, "Pruning convolutional neural networks for resource efficient inference," in *ICLR*, 2017.

[4] S. Goyal, A. Roy Choudhury, and V. Sharma, "Compression of deep neural networks by combining pruning and low rank decomposition," in *IPDPSW*, 2019.

[5] C. Dai, X. Liu, H. Cheng, L. T. Yang, and M. J. Deen, "Compressing deep model with pruning and tucker decomposition for smart embedded systems," *IEEE Internet of Things Journal*, 2021.

[6] Y. Li, S. Lin, J. Liu, Q. Ye, M. Wang, F. Chao, F. Yang, J. Ma, Q. Tian, and R. Ji, "Towards compact cnns via collaborative compression," in *CVPR*, 2021.

[7] X. Yu, T. Liu, X. Wang, and D. Tao, "On compressing deep models by low rank and sparse decomposition," in *CVPR*, 2017.

[8] X. Ruan, Y. Liu, C. Yuan, B. Li, W. Hu, Y. Li, and S. Maybank, "Edp: An efficient decomposition and pruning scheme for convolutional neural network compression," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 10, pp. 4499–4513, 2021.

[9] Z. Chen, J. Xiang, Y. Lu, Q. Xuan, Z. Wang, G. Chen, and X. Yang, "Rgp: Neural network pruning through regular graph with edges swapping," *IEEE Transactions on Neural Networks and Learning Systems*, 2023.

[10] J. Xiao, C. Zhang, Y. Gong, M. Yin, Y. Sui, L. Xiang, D. Tao, and B. Yuan, "Haloc: Hardware-aware automatic low-rank compression for compact neural networks," in *AAAI*, 2023.

[11] G. Wallace, "The jpeg still picture compression standard," *IEEE Transactions on Consumer Electronics*, 1992.

[12] Y. 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," in *ICLR*, 2016.

[13] M. Gabor and R. Zdunek, "Compressing convolutional neural networks with hierarchical tucker-2 decomposition," *Applied Soft Computing*, 2023.

[14] C. Hawkins, X. Liu, and Z. Zhang, "Towards compact neural networks via end-to-end training: A bayesian tensor approach with automatic rank determination," *SIAM Journal on Mathematics of Data Science*, 2022.

[15] Y. Idelbayev and M. A. Carreira-Perpinan, "Low-rank compression of neural nets: Learning the rank of each layer," in *CVPR*, 2020.

[16] S. Nakajima, M. Sugiyama, S. D. Babacan, and R. Tomioka, "Global analytic solution of fully-observed variational bayesian matrix factorization," *JMLR*, 2013.

[17] M. Eo, S. Kang, and W. Rhee, "An effective low-rank compression with a joint rank selection followed by a compression-friendly training," *Neural Networks*, vol. 161, pp. 165–177, 2023.

[18] X. Zhang, J. Zou, K. He, and J. Sun, "Accelerating very deep convolutional networks for classification and detection," *TPAMI*, 2016.

[19] M. Astrid, S.-I. Lee, and B.-S. Seo, "Rank selection of cp-decomposed convolutional layers with variational bayesian matrix factorization," in *ICACT*, 2018.

[20] L. Liebenwein, A. Maalouf, O. Gal, D. Feldman, and D. Rus, "Compressing neural networks: Towards determining the optimal layer-wise decomposition," in *NIPS*, 2021.

[21] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *SIAM Review*, vol. 51, no. 3, pp. 455–500, 2009.

[22] S. Liu, T. Chen, X. Chen, X. Chen, Q. Xiao, B. Wu, T. Kärkkäinen, M. Pechenizkiy, D. C. Mocanu, and Z. Wang, "More convnets in the 2020s: Scaling up kernels beyond 51x51 using sparsity," in *ICLR*, 2023.

[23] S. Han, J. Pool, J. Tran, and W. J. Dally, "Learning both weights and connections for efficient neural networks," in *NIPS*, 2015.

[24] Y. Li, S. Gu, C. Mayer, L. Van Gool, and R. Timofte, "Group sparsity: The hinge between filter pruning and decomposition for network compression," in *CVPR*, 2020.

[25] S.-K. Yeom, K.-H. Shim, and J.-H. Hwang, "Toward compact deep neural networks via energy-aware pruning," in *tinyML*, 2022.

[26] Y. Zhang and N. M. Freris, "Adaptive filter pruning via sensitivity feedback," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–13, 2023.

[27] S. Guo, L. Zhang, X. Zheng, Y. Wang, Y. Li, F. Chao, C. Wu, S. Zhang, and R. Ji, "Automatic network pruning via hilbert-schmidt independence criterion lasso under information bottleneck principle," in *ICCV*, 2023.

[28] T. Castells and S.-K. Yeom, "Automatic neural network pruning that efficiently preserves the model accuracy," in *AAAI*, 2023.

[29] Y. Zhang, M. Lin, C.-W. Lin, J. Chen, Y. Wu, Y. Tian, and R. Ji, "Carrying out cnn channel pruning in a white box," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 34, no. 10, pp. 7946–7955, 2023.

[30] M. Alwani, V. Madhavan, and Y. Wang, "Decore: Deep compression with reinforcement learning," *CVPR*, 2022.

[31] R. Lin, J. Ran, D. Wang, K. Chiu, and N. Wong, "Ezcrop: Energy-zoned channels for robust output pruning," in *WACV*, 2022.

[32] Y. Sui, M. Yin, Y. Xie, H. Phan, S. Zonouz, and B. Yuan, "Chip: Channel independence-based pruning for compact neural networks," in *NeurIPS*, 2021.

[33] X. Liu, B. Li, Z. Chen, and Y. Yuan, "Exploring gradient flow based saliency for dnn model compression," in *ACM MM*, 2021.

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

[35] Åke Björck and G. H. Golub, "Numerical methods for computing angles between linear subspaces," *Mathematics of Computation*, 1973.

[36] E. Denton, W. Zaremba, J. Bruna, Y. LeCun, and R. Fergus, "Exploiting linear structure within convolutional networks for efficient evaluation," in *NIPS*, 2014.

[37] C. Tai, T. Xiao, X. Wang, and W. E, "Convolutional neural networks with low-rank regularization," in *ICLR*, 2016.

[38] C. Li and C. J. R. Shi, "Constrained optimization based low-rank approximation of deep neural networks," in *ECCV*, 2018.

[39] Y. Xu, Y. Li, S. Zhang, W. Wen, B. Wang, Y. Qi, Y. Chen, W. Lin, and H. Xiong, "Trp: Trained rank pruning for efficient deep neural networks," in *IJCAI*, 2020.

[40] H. Yang, M. Tang, W. Wen, F. Yan, D. Hu, A. Li, H. Li, and Y. Chen, "Learning low-rank deep neural networks via singular vector orthogonality regularization and singular value sparsification," in *CVPRW*, 2020.

[41] D. Wang, B. Wu, G. Zhao, M. Yao, H. Chen, L. Deng, T. Yan, and G. Li, "Kronecker cp decomposition with fast multiplication for compressing rnns," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 34, no. 5, pp. 2205–2219, 2023.

[42] T. Garipov, D. Podoprikhin, A. Novikov, and D. P. Vetrov, "Ultimate tensorization: compressing convolutional and fc layers alike," *NIPS Workshop*, 2016.

[43] W. Wang, Y. Sun, B. Eriksson, W. Wang, and V. Aggarwal, "Wide compression: Tensor ring nets," in *CVPR*, 2018.

[44] B. Wu, D. Wang, G. Zhao, L. Deng, and G. Li, "Hybrid tensor decomposition in neural network compression," *Neural Networks*, 2020.

[45] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient convnets," *ICLR*, 2017.

[46] H. Wang and Y. Fu, "Trainability preserving neural pruning," in *ICLR*, 2023.

[47] S. Chen, W. Sun, and L. Huang, "Whc: Weighted hybrid criterion for filter pruning on convolutional neural networks," in *ICASSP*, 2023.

[48] M. Lin, L. Cao, Y. Zhang, L. Shao, C.-W. Lin, and R. Ji, "Pruning networks with cross-layer ranking & k-reciprocal nearest filters," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 34, no. 11, pp. 9139–9148, 2023.

[49] X. Ding, T. Hao, J. Tan, J. Liu, J. Han, Y. Guo, and G. Ding, "Resrep: Lossless cnn pruning via decoupling remembering and forgetting," in *ICCV*, 2021.

[50] T.-W. Chin, R. Ding, C. Zhang, and D. Marculescu, "Towards efficient model compression via learned global ranking," *CVPR*, 2019.

[51] Y. Tang, Y. Wang, Y. Xu, D. Tao, C. XU, C. Xu, and C. Xu, "Scop: Scientific control for reliable neural network pruning," in *NeurIPS*, 2020.

[52] W. Hu, Z. Che, N. Liu, M. Li, J. Tang, C. Zhang, and J. Wang, "Catro: Channel pruning via class-aware trace ratio optimization," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–13, 2023.

[53] L. Liebenwein, C. Baykal, H. Lang, D. Feldman, and D. Rus, "Provable filter pruning for efficient neural networks," in *ICLR*, 2020.

[54] Y.-J. Zheng, S.-B. Chen, C. H. Q. Ding, and B. Luo, "Model compression based on differentiable network channel pruning," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–10, 2022.

[55] W. Gao, Y. Guo, S. Ma, G. Li, and S. Kwong, "Efficient neural network compression inspired by compressive sensing," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–15, 2022.

[56] M. Lin, R. Ji, Y. Zhang, B. Zhang, Y. Wu, and Y. Tian, "Channel pruning via automatic structure search," in *IJCAI*, 2021.

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

[58] M. Eo, S. Kang, and W. Rhee, "A differentiable framework for end-to-end learning of hybrid structured compression," *arXiv preprint arXiv:2309.13077*, 2023.

[59] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *ICLR*, 2015.

[60] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016.

[61] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *CVPR*, 2017.

[62] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," University of Toronto, Tech. Rep. 0, 2009.

[63] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. S. Bernstein, A. C. Berg, and L. Fei-Fei, "Imagenet large scale visual recognition challenge," *IJCV*, 2014.

[64] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *NIPS*, 2015.

[65] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *ICCV*, 2017.

[66] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *ECCV*, 2014.

[67] Y. Qian, Z. He, Y. Wang, B. Wang, X. Ling, Z. Gu, H. Wang, S. Zeng, and W. Swaileh, "Hierarchical threshold pruning based on uniform response criterion," *IEEE Transactions on Neural Networks and Learning Systems*, 2023.

[68] T. Hao, X. Ding, J. Han, Y. Guo, and G. Ding, "Manipulating identical filter redundancy for efficient pruning on deep and complicated cnn," *IEEE Transactions on Neural Networks and Learning Systems*, 2023.

[69] Y. Guan, N. Liu, P. Zhao, Z. Che, K. Bian, Y. Wang, and J. Tang, "Dais: Automatic channel pruning via differentiable annealing indicator search," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–12, 2022.

[70] G. Fang, X. Ma, M. Song, M. B. Mi, and X. Wang, "Depgraph: Towards any structural pruning," *CVPR*, 2023.

[71] G. Liu, K. Zhang, and M. Lv, "Soks: Automatic searching of the optimal kernel shapes for stripe-wise network pruning," *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[72] Y. He, P. Liu, L. Zhu, and Y. Yang, "Filter pruning by switching to neighboring cnns with good attributes," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 34, no. 10, pp. 8044–8056, 2023.

[73] M. Lin, L. Cao, S. Li, Q. Ye, Y. Tian, J. Liu, Q. Tian, and R. Ji, "Filter sketch for network pruning," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 12, pp. 7091–7100, 2022.

[74] Y. Duan, Y. Zhou, P. He, Q. Liu, S. Duan, and X. Hu, "Network pruning via feature shift minimization," in *ACCV*, 2022.

[75] M. Lin, R. Ji, S. Li, Y. Wang, Y. Wu, F. Huang, and Q. Ye, "Network pruning using adaptive exemplar filters," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 12, pp. 7357–7366, 2022.

[76] Z. Wang, X. Liu, L. Huang, Y. Chen, Y. Zhang, Z. Lin, and R. Wang, "Qsfm: Model pruning based on quantified similarity between feature maps for ai on edge," *IEEE Internet of Things Journal*, 2022.

[77] J. Liu, B. Zhuang, Z. Zhuang, Y. Guo, J. Huang, J. Zhu, and M. Tan, "Discrimination-aware network pruning for deep model compression," *TPAMI*, 2022.

[78] J.-H. Luo and J. Wu, "Neural network pruning with residual-connections and limited-data," in *CVPR*, 2020.

[79] T. maintainers and contributors, "Torchvision: Pytorch's computer vision library," https://github.com/pytorch/vision, 2016.

[80] J. Schuurmans, K. Batselier, and J. Kooij, "How informative is the approximation error from tensor decomposition for neural network compression?" in *ICLR*, 2023.

[81] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," *IJCV*, 2020.