Efficient tensor decomposition-based filter pruning

Van Tien Pham, Yassine Zniyed, Thanh Phuong Nguyen

Efficient tensor decomposition-based filter pruning

Van Tien Pham[a,*], Yassine Zniyed[a], Thanh Phuong Nguyen[a]

[a] *Université de Toulon, Aix Marseille University, CNRS, LIS, France*

Abstract

In this paper, we present CORING, which is short for effiCient tensOr decomposition-based filteR prunING, a novel filter pruning methodology for neural networks. CORING is crafted to achieve efficient tensor decomposition-based pruning, a stark departure from conventional approaches that rely on vectorized or matricized filter representations. Our approach represents a significant leap forward in the field by introducing tensor decompositions, specifically the HOSVD, which preserves the multidimensional nature of filters while providing a low-rank approximation, thus substantially reducing complexity. Furthermore, we introduce a versatile method for calculating filter similarity by using the low-rank approximation offered by the HOSVD. This obviates the need for using full filters or reshaped versions and enhances the overall efficiency and effectiveness of our approach. Extensive experimentation across diverse architectures and datasets spanning various vision tasks, including image classification, object detection, instance segmentation, and keypoint detection, validates CORING's prowess. Remarkably, it outperforms state-of-the-art methods in reducing MACs and parameters, consistently enhancing validation accuracy. Furthermore, we supplement our quantitative results with a comprehensive ablation study, providing substantial evidence of the efficiency of our tensor-based approach. Beyond quantitative outcomes, qualitative results vividly illustrate CORING's ability to retain essential features within pruned neural networks. Our code is available for research purposes.

Keywords: network compression, tensor decompositions, filter pruning

1. Introduction

Network pruning is an important technique for designing efficient models of convolutional neural networks (CNNs) because it reduces the memory footprint and computation requirements while maintaining or improving the overall performance. This is particularly crucial when deploying CNNs on resource-constrained devices such as mobile phones or embedded systems. The hypothesis behind network pruning is that many models are over-parameterized, *i.e.,* they contain a large number of unnecessary or redundant parameters [56, 83]. Pruning redundant parameters can lead to a smaller and more efficient model that can be deployed on resource-constrained devices, while also improving the model's generalization in some cases [5].

Among existing pruning techniques, filter pruning and weight pruning are both popular approaches. Weight pruning is a form of unstructured pruning [14], where individuals deemed insignificant weights are pruned without considering any specific structure or pattern. On the other hand, filter pruning is an example of structured pruning [1, 3, 16, 29, 31, 38, 63, 83], where entire filters are pruned based on some criteria while maintaining the overall structure of the network. Compared to its counterpart, filter pruning is more interpretable, less sensitive to initialization, more computationally efficient, and allows direct deployment on terminal devices [56].

Undoubtedly, the choice of filters is the foundation of filter pruning. Early works [29, 38, 75, 83] determine the filter importance by measuring only the information of individual filters themselves. However, these approaches neglect the correlation between inter-filters, resulting in high redundancy. Recent advancements [61, 62, 63, 71, 79, 85] have demonstrated the potential benefits of leveraging the correlations or similarities between filters/feature maps

*Corresponding author

Email addresses: van-tien-pham@etud.univ-tln.fr (Van Tien Pham), zniyed@univ-tln.fr (Yassine Zniyed), tpnguyen@univ-tln.fr (Thanh Phuong Nguyen)

to reduce redundancy. This is based on the hypothesis that similar filters may generate duplicate features, and that eliminating this redundancy can be compensated during the fine-tuning process. Similarity-based pruning approaches evaluate the importance of filters by measuring the pairwise distance between filters in order to construct the similarity matrix of each network's layer. This distance/similarity metric quantifies the correlation or similarity between each pair of filters in the layer. Based on this similarity matrix, the importance or saliency of filters can be determined and filters that are deemed less important can be pruned.

Despite their promising results, many state-of-the-art (SOTA) methods for filter pruning suffer from some limitations that have yet to be fully addressed. Existing works [8, 16, 61, 71, 79, 85] often flatten 3-order tensor filters into 2-D matrices or 1-D vectors, which can result in loss of spatial or temporal information. The multidimensional structure of filters is important, and neglecting it through flattening can lead to loss of crucial information [27, 59]. Moreover, inter-filter approaches [8, 61, 71, 79] can require more complex and computationally expensive analysis compared to methods that only consider intra-filter information. Iterative pruning can be computationally expensive due to the need to compute the similarity matrix at each iteration. Thus, there is a demand for computationally efficient methods to address this issue. Lastly, several works [26, 30, 31, 36, 38, 49, 58, 63, 66, 73, 78] try to rank the filters through their corresponding feature maps or channels. With their feature-guided nature, such data-dependent methods are sensitive to the distribution of input data, making it difficult to accurately estimate average feature map values. This can require a large set of input images [31, 36, 38, 63] to estimate the average ranks of each feature map on a fixed dataset, further complicating the pruning process.

In this paper, we propose a novel approach, called CORING, for effiCient tensOr decomposition-based filteR prunING. It is a filter pruning technique using tensor decompositions [27]. Specifically, we decompose each layer's filters using the higher-order singular value decomposition (HOSVD) [9] and use this representation to measure similarity between filters, rather than considering the entire filter in its tensor, matrix, or vector form. This method allows to (i) preserve the multidimensional structure of the filters and their essential information while providing a low-rank approximation, and (ii) reduce computational time as we will show later. This approach is general and can work with any similarity metric. In our experiments, we assess our approach using the Euclidean and cosine distances, as well as VBD (Variance-Based Distance), which is an adaptation of the Signal to Noise Ratio (SNR) distance [77]. We show that our approach is effective across all these metrics.

This work brings the following contributions:

- Introduces tensor decompositions, specifically HOSVD, for filter pruning. This method preserves the multidimensional structure of filters while providing a low-rank approximation, effectively reducing complexity. This decomposition allows a novel and versatile way of calculating filter similarity using the representation derived from HOSVD, avoiding the need for the entire filter or its reshaped versions. The proposed method is tested with Euclidean, cosine, and VBD distances.
- Presents a straightforward and efficient filter selection method based on the similarity matrix. This method considers relationships between filters through a distance measure between the corresponding HOSVD factors of each pair of filters.
- Evaluates the proposed framework across various computer vision tasks, including image classification, object detection, instance segmentation, and keypoint detection. Through extensive experiments, the effectiveness of CORING is demonstrated in terms of accuracy, parameter reduction, and MACs reduction, showcasing its superiority over SOTA.

2. Related Works

Network pruning can be classified into three primary types based on the granularity of the pruning process: unstructured pruning, N:M sparsity, and structured pruning. Unstructured pruning, often termed weight pruning, involves removing individual weights deemed insignificant [14, 44, 82]. This typically entails setting small weights to zero, leading to a sparse network structure without altering the overall architecture. Despite its effectiveness in compression, weight pruning can be sensitive to the network's initial weights and often results in an unstructured sparsity pattern, introducing inefficiencies in hardware acceleration. N:M sparsity [84] involves setting N out of M contiguous weights to zero, offering advantages like reduced computational complexity and improved memory efficiency [81]. However, it may pose challenges during training and generalization, potentially introducing overhead in sparse matrix operations. On the other hand, structured pruning, exemplified by filter pruning, aims to eliminate unimportant

filters from the network [1, 29, 38, 63]. Unlike unstructured pruning, this method directly reduces the number of computations needed during the inference phase, leading to significant reductions in the network's memory footprint by directly decreasing the number of parameters.

Pioneering works in filter pruning determine the filter's importance by measuring the information of the filter itself. By supposing that filters with smaller norms are less important, [7, 29, 72] adopt the $L1$ or $L2$-norm to measure the filter saliency. A regularization-based amplitude saliency pruning evaluation criterion was investigated in [83]. However, methods that rely only on individual filter information neglect the correlation between filters, leading to redundancy in the resulting pruned models. Recently, similarity-based filter pruning [62, 71, 73, 79] has become an emerging approach to eliminate redundant filters that contribute similarly to the final output. To achieve this, filters are compared to each other based on a similarity metric, and those with the highest similarity are removed during the pruning process. Several works tried to apply common similarity metrics to measure filters correlation, such as Euclide [8, 79], Manhattan [79], Chebyshev [61, 79], Hamming [31] distances, cosine similarity [16, 30, 61, 71, 85], Pearson correlation [62, 71], to mention a few. Most of these works overlook the multidimensional structure of filters and rely on flattened versions of filters to compute similarity, resulting in information loss. Notably, the k-Reciprocal Nearest Filter (RNF) selection scheme proposed in [37] shares a core idea with our approach in considering the collective importance of a filter with other candidates rather than focusing solely on individual importance, as seen in previous works [29, 38, 75, 83]. However, RNF employs the entire filter version to calculate the similarity matrix, whereas our approach relies on the low-rank representation containing essential information.

When comparing filters for pruning, there are two main approaches: inter-layer comparison and intra-layer comparison. The filter selection is a crucial factor in filter pruning, as it determines the scope for filter removal, and refers to whether the comparison is limited to filters within a single layer, or extends across all layers in the network. It is worth mentioning that some researchers [63, 71] argue that inter-layer comparison is more reasonable for filter pruning. However, in this work, we use an intra-layer comparison, as each layer of a neural network serves a unique purpose in feature extraction.

Concerning pruning strategy, there exist two main approaches: one-shot pruning [38, 63, 79] and multi-shot pruning [4, 14]. With the first one, the entire network is pruned in a single shot, and the filters or weights in each layer are either preserved or pruned at once. This method is faster than the last one because it only requires one round of training and pruning. In contrast, multi-shot pruning is a pruning strategy in which the network is iteratively pruned, and each iteration involves pruning a portion of the redundant filters or weights. As discussed in [4, 14], this approach allows the network to adapt to the pruning process, preserving the most important elements and improving the overall accuracy. This method allows for more flexibility in choosing which filters to prune, which results in more effective pruning.

Tensor decompositions find widespread applications in network compression, as evidenced by various studies [33, 51, 52, 69, 70, 74]. The tensor ring factorization, for instance, was proposed to compress the weights of convolutional networks in [70] and recurrent networks in [51]. In [74], the hierarchical Tucker decomposition was introduced to convert weight matrices and compress convolutional kernels. Notably, [33] presented a progressive genetic algorithm to determine the optimal rank for decomposition. A recent survey by Wang et al. [69] provides a comprehensive overview of tensorial neural networks, highlighting the multilinearity structure of weight tensors. Despite the natural representation of filters in CNNs as multidimensional arrays, current practices in structured pruning [38, 63, 10, 79] often involve treating 3-order tensor filters by flattening them into matrices or vectors for ease of use. This approach, commonly applied in filter pruning [10, 79] and feature pruning [38, 63], results in information loss and may not fully exploit the multidimensional structure of the data, as far as our knowledge extends.

3. The CORING Framework

3.1. Notations and Preliminaries

The notations used throughout the rest of this paper are now defined. The outer product is denoted by $\circ$. The variance is denoted as $\mathrm{Var}(.)$. The floor function $\lfloor x \rfloor$ denotes the greatest integer less than or equal to x. Tensors are represented by bold calligraphic capital letters, e.g., $\boldsymbol{\mathcal{X}}$. The norm of a tensor $\boldsymbol{\mathcal{X}}$ is the square root of the sum of the squares of all its elements, *i.e.,* $||\boldsymbol{\mathcal{X}}|| = \sqrt{\sum_{i=1}^{N1}\sum_{j=1}^{N2}\sum_{k=1}^{N3}\mathcal{X}_{i,j,k}^2}$. $\mathrm{unfold}_q\boldsymbol{\mathcal{X}}$ refers to the unfolding of tensor $\boldsymbol{\mathcal{X}}$ over its q-th mode [27]. We now recall some definitions that will be useful in the sequel. The q-mode product is one

of the most important operations in tensor processing. It is defined as a product of a tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{N_1 \times \ldots \times N_Q}$ and a matrix $\mathbf{U} \in \mathbb{R}^{J \times N_q}$ as:

$$(\boldsymbol{\mathcal{X}} \times_n \mathbf{U})_{i_1 \ldots i_{q-1} j i_{q+1} \ldots i_D} \triangleq \sum_{i_q=1}^{N_q} X_{i_1 i_2 \ldots i_Q} u_{j i_q}. \tag{1}$$

Definition 1. *Based on the definition of the q-mode product, we can recall the definition of a Tucker decomposition (TD) [67]. Consider a core tensor $\boldsymbol{\mathcal{S}} \in \mathbb{R}^{R_1 \times R_2 \times R_3}$ and 3 factor matrices $\mathbf{A} \in \mathbb{R}^{N_1 \times R_1}, \mathbf{B} \in \mathbb{R}^{N_2 \times R_2}, \mathbf{C} \in \mathbb{R}^{N_3 \times R_3}$. The Tucker decomposition of $\boldsymbol{\mathcal{T}} \in \mathbb{R}^{N_1 \times N_2 \times N_3}$ is given by:*

$$\boldsymbol{\mathcal{T}} = \boldsymbol{\mathcal{S}} \times_1 \mathbf{A} \times_2 \mathbf{B} \times_3 \mathbf{C} \triangleq [\![\boldsymbol{\mathcal{S}}; \mathbf{A}, \mathbf{B}, \mathbf{C}]\!]. \tag{2}$$

The triplet of minimal values of $\{R_1, R_2, R_3\}$ forms the multilinear rank *of $\boldsymbol{\mathcal{T}}$. An easy way to obtain the TD of a tensor is to use the HOSVD algorithm [9]. In the case of the HOSVD, the factor matrices in (2) are constrained to be orthonormal, which can simplify the computation and interpretation of the factor matrices.*

Let us consider a pre-trained CNN model with L layers, denoted as $\mathscr{C}_1, \mathscr{C}_2, \ldots, \mathscr{C}_L$. The parameters of $\mathscr{C}_l$ can be represented as a set of 3-order filters $\boldsymbol{\mathcal{F}}^{l_1}, \boldsymbol{\mathcal{F}}^{l_2}, \ldots, \boldsymbol{\mathcal{F}}^{l_{c_l}}$ containing c_l filters $\boldsymbol{\mathcal{F}}^{l_i} \in \mathbb{R}^{c_{l-1} \times h_l \times w_l}$, where c_l, c_{l-1}, h_l and w_l denote the number of output channels, the number of input channels, the kernel height and the kernel width, respectively. In general, if we define a 4-order tensor s.t. $\boldsymbol{\mathcal{W}}^l_{:,:,:,i} = \boldsymbol{\mathcal{F}}^{l_i}$, then the objective of filter pruning is to optimize the following loss function:

$$\min_{\{\boldsymbol{\mathcal{W}}^l\}_{l=1}^L} \mathcal{L}(\boldsymbol{\mathcal{Y}}, f(\boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{W}}^l)), \quad \text{s.t.} \quad g(\boldsymbol{\mathcal{W}}^l) \leq \kappa_l, \tag{3}$$

where $\mathcal{L}(\cdot, \cdot)$ is the loss function, $\boldsymbol{\mathcal{Y}}$ is the ground-truth labels, $\boldsymbol{\mathcal{X}}$ is the input data, $f(\cdot, \cdot)$ is the output of the CNN model, κ_l is the number of filters to be kept in the l^{th} layer, and $g(.)$ is the number of non-zero filters of its argument.

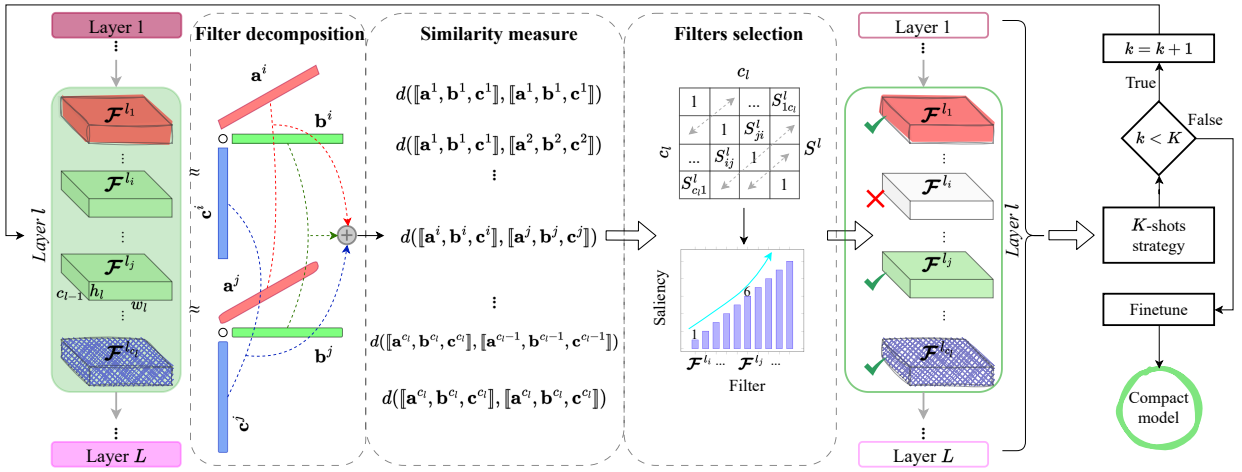


Figure 1: The CORING approach for filter pruning in one layer, summarized in three steps.

Fig. 1 provides an overview of the proposed methodology. Our approach begins with a pre-trained model, as defined in Section 3.1. The model is systematically processed layer by layer. Firstly, the tensor decomposition module, as elaborated in Subsection 3.2, transforms third-order filters into a low-rank representation through approximation. Subsequently, in Subsection 3.3, we calculate the distance between two filters by performing pairwise computations on their corresponding representative vectors, resulting in the formation of a similarity matrix. To determine the importance of filters and eliminate redundant ones, we introduce a filter selection algorithm in Subsection 3.4. The pruned model at iteration k undergoes calibration and serves as the base model for the subsequent iteration, as elucidated in Subsection 3.5. Ultimately, CORING produces a compact model that maintains comparable accuracy while significantly improving computation time and reducing memory usage.

3.2. Filter Decomposition

In this section, for the sake of clarity and to simplify the notation, we consider a filter without its subscript as $\boldsymbol{\mathcal{F}}$ of size $c_{l-1} \times h_l \times w_l$. Now, if we apply the TD in (2) to $\boldsymbol{\mathcal{F}}$ by considering that $R_1 = R_2 = R_3 = 1$, the model reduces to

$$\boldsymbol{\mathcal{F}} \approx s \times_1 \mathbf{a} \times_2 \mathbf{b} \times_3 \mathbf{c} = [\![s; \mathbf{a}, \mathbf{b}, \mathbf{c}]\!], \tag{4}$$

where $\mathbf{a} \in \mathbb{R}^{c_{l-1}}$, $\mathbf{b} \in \mathbb{R}^{h_l}$, $\mathbf{c} \in \mathbb{R}^{w_l}$, and s is a scalar that can also be seen as a 3-order tensor $s \in \mathbb{R}^{1\times1\times1}$. Without loss of generality, we can now denote $\boldsymbol{\mathcal{F}}$ simply as

$$\boldsymbol{\mathcal{F}} \approx [\![\mathbf{a}, \mathbf{b}, \mathbf{c}]\!]. \tag{5}$$

To decompose $\boldsymbol{\mathcal{F}}$ as in (5), there are multiple tensor decomposition methods available. In this work, we choose to use the HOSVD algorithm [9]. This choice is justified by several factors. Firstly, the HOSVD is a non-iterative algorithm, unlike ALS-based techniques [20], which can be computationally expensive. Secondly, it is based on the SVD, which ensures that the approximation with respect to the decomposed matrices is optimal [17]. Finally, the HOSVD is relatively easy to implement, making it a practical choice for many applications. Indeed, the HOSVD of $\boldsymbol{\mathcal{F}}$ involves the computation of the SVD of the three unfolding matrices [27] $\text{unfold}_1\boldsymbol{\mathcal{F}}$, $\text{unfold}_2\boldsymbol{\mathcal{F}}$ and $\text{unfold}_3\boldsymbol{\mathcal{F}}$, of size, respectively, $c_{l-1} \times (h_l \cdot w_l)$, $h_l \times (w_l \cdot c_{l-1})$ and $w_l \times (h_l \cdot c_{l-1})$. Following the HOSVD algorithm, the three vectors $\mathbf{a}$, $\mathbf{b}$, and $\mathbf{c}$ in (5) represent, respectively, the first dominant left singular vector of the matrices mentioned before. By applying rank-1 SVD to the matrices $\text{unfold}_q\boldsymbol{\mathcal{F}}$ (for $1 \leq q \leq 3$), it is guaranteed, by the Eckart-Young theorem [13], that the obtained approximation is the best rank-1 approximation in the Frobenius norm. However, the overall low-rank tensor approximation is generally not optimal. Nevertheless, it has been shown that the obtained decomposition is a good approximation of $\boldsymbol{\mathcal{F}}$, and it is bounded by a certain limit [27]. From a computational complexity standpoint, the choice of $R_1 = R_2 = R_3 = 1$ provides the most efficient approximation method for a given tensor. While it is true that increasing the multilinear rank can potentially lead to a better approximation of the original tensor, we decide to use multilinear rank all equal to 1. This choice is motivated by considerations of computational complexity as well as the results we obtained during our experiments. Specifically, we find that using this approximation provides a good trade-off between approximation accuracy and computational efficiency, which makes it a practical choice for our method. In this way, not only the multidimensional information is preserved, but also computational efficiency is attained as shown in Subsection 5.1.

3.3. Similarity Measure

The aim of this subsection is twofold. First, we present VBD, a measure of similarity between two filters. Second, we describe the process by which we calculate the distance using the low-rank approximations of the filters.

The authors of [77] proposed an SNR-based metric to measure the similarity of image pairs for deep metric learning. While this *quasi*-metric has been shown to be effective, it has one major caveat that should be noted: it does not satisfy the symmetry property, which is important for distance functions. To remedy this, we define the VBD as

$$d_{VBD}\left(\boldsymbol{\mathcal{F}}^i, \boldsymbol{\mathcal{F}}^j\right) = \frac{\text{Var}\left(\boldsymbol{\mathcal{F}}^i - \boldsymbol{\mathcal{F}}^j\right)}{\text{Var}\left(\boldsymbol{\mathcal{F}}^i\right) + \text{Var}\left(\boldsymbol{\mathcal{F}}^j\right)}. \tag{6}$$

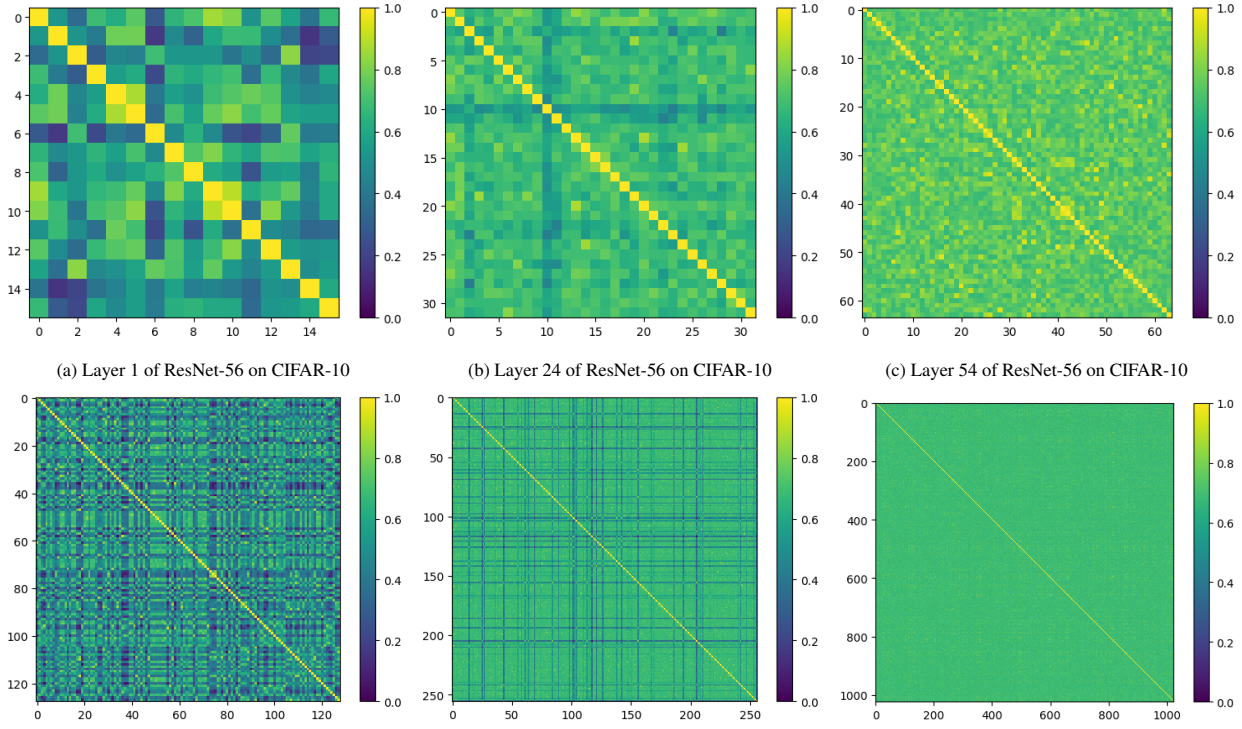
Let us now consider $d(.,.)$ as a general distance function. To calculate the distance between a pair of filters $\boldsymbol{\mathcal{F}}^i$, $\boldsymbol{\mathcal{F}}^j$, we will use their HOSVD as in (5). Let us assume that $\boldsymbol{\mathcal{F}}^i = [\![\mathbf{a}^i, \mathbf{b}^i, \mathbf{c}^i]\!]$ and $\boldsymbol{\mathcal{F}}^j = [\![\mathbf{a}^j, \mathbf{b}^j, \mathbf{c}^j]\!]$. In this case, we can calculate the distance between $\boldsymbol{\mathcal{F}}^i$ and $\boldsymbol{\mathcal{F}}^j$ as follows.

$$d(\boldsymbol{\mathcal{F}}^i, \boldsymbol{\mathcal{F}}^j) = d\big([\![\mathbf{a}^i, \mathbf{b}^i, \mathbf{c}^i]\!], [\![\mathbf{a}^j, \mathbf{b}^j, \mathbf{c}^j]\!]\big). \tag{7}$$

A simple way to calculate this distance is to take the average of the distances between the corresponding factors of the two filters as follows.

$$d(\boldsymbol{\mathcal{F}}^i, \boldsymbol{\mathcal{F}}^j) = \frac{d(\mathbf{a}^i, \mathbf{a}^j) + d(\mathbf{b}^i, \mathbf{b}^j) + d(\mathbf{c}^i, \mathbf{c}^j)}{3}. \tag{8}$$

Depending on the chosen distance metric, a similarity matrix $\mathbf{S}$ of size $c \times c$ can be constructed such that $S_{ij} = d(\boldsymbol{\mathcal{F}}^i, \boldsymbol{\mathcal{F}}^j)$. S_{ij} represents the similarity between the i-th and j-th filters, and $d(\cdot, \cdot)$ is the chosen distance function. Fig. 2 illustrates the cosine similarity among three layers of the ResNet-56 model [22] on CIFAR-10 [28] and ResNet-50 on ImageNet [54]. Notably, the presence of numerous yellow points indicates the existence of similarity in the models, aligning with our hypothesis of pruning based on similarity. Additionally, the redundancy tends to be more prevalent and pronounced in the later layers of the architecture. This observation is consistent with findings from recent research on automatic pruning rate search [64].



(a) Layer 1 of ResNet-56 on CIFAR-10

(b) Layer 24 of ResNet-56 on CIFAR-10

(c) Layer 54 of ResNet-56 on CIFAR-10

(d) Layer 12 of ResNet-50 on Imagenet

(e) Layer 25 of ResNet-50 on Imagenet

(f) Layer 42 of ResNet-50 on Imagenet

Figure 2: Visual representation of the similarity matrix across three layers of the ResNet-56 [22] model on CIFAR-10 dataset [28] and ResNet-50 on ImageNet [54] dataset.

3.4. Filters Selection

Algorithm 1 presents the filter selection procedure used in CORING. The algorithm takes as input a similarity matrix between all pairs of filters, the set of filters, and a sparsity target. The output of the algorithm is a set of κ selected filters. The core idea is to consider the collective importance of a filter with other candidates rather than focusing solely on individual importance, as seen in previous works [29, 38, 75, 83]. The procedure works by iteratively deleting filters that are most similar to the other filters. The algorithm starts by finding the pair of filters with the highest similarity and deleting one of the filters. The choice of the filter to delete is based on the sum of its similarities with the other filters in the layer. The algorithm then updates the similarity matrix by deleting the row and column of the deleted filter and continues to delete filters until the desired sparsity target is reached. The efficacy of our proposed algorithm is demonstrated through a representative example in Subfigure 2a. The algorithm detects the saliency of filters, yielding a sequence of indices such as $[5, 14, 7, 13, 4, 1, 15, 6, 10, 0, 3, 8, 9, 2, 12, 11]$. This sequence effectively identifies the most (index 0, darkest-blue) and least (index 9, brightest-yellow) important filters. In contrast, the $L1$ norm-based method [29] produces a different sequence, e.g., $[1, 11, 15, 0, 6, 7, 12, 5, 4, 8, 2, 13, 3, 14, 9, 10]$, ranking filter index 2 as the most important. However, it is evident that this filter is highly similar to filter index 13, indicating redundancy between them.

Algorithm 1 Filters selection

Require: Similarity matrix $\boldsymbol{S} \in \mathbb{R}^{c \times c}$, filters $\boldsymbol{\mathcal{F}}^1, \boldsymbol{\mathcal{F}}^2, \ldots, \boldsymbol{\mathcal{F}}^c$, sparsity target κ.
Ensure: Selected filters $\boldsymbol{\mathcal{F}}^{p_1}, \boldsymbol{\mathcal{F}}^{p_2}, \ldots, \boldsymbol{\mathcal{F}}^{p_\kappa}$.

for $t = 1$ to $c - \kappa$ **do**
  Find the highest similarity:
  $$(i, j) = \underset{\substack{(x,y) \\ x \neq y}}{\operatorname{argmax}} \quad S_{x,y}$$
  if $\sum_{k=1}^{c} S_{i,k} \geq \sum_{k=1}^{c} S_{j,k}$ **then**
    Delete $\boldsymbol{\mathcal{F}}^i$.
  else
    Delete $\boldsymbol{\mathcal{F}}^j$.
  end if
  Delete the row, column of the deleted filter in $\mathbf{S}$.
end for

For the sake of simplicity, we omitted the index l in the pseudo-code, but the pruning is addressed for each convolutional layer by considering κ_l filters to preserve at layer l. The most different κ_l filters, named $\boldsymbol{\mathcal{F}}^{p_1}, \boldsymbol{\mathcal{F}}^{p_2}, \ldots, \boldsymbol{\mathcal{F}}^{p_{\kappa_l}}$, are preserved, while the rest is pruned. This process is parallelly executed on all layers.

3.5. Pruning Strategy

Fig. 3 illustrates our proposed K-shots pruning strategy. It is a variation of multi-shot pruning, which prunes the network in K rounds with the possibility of fine-tuning between rounds. By fixing the number of pruning rounds, K-shots pruning provides better control over the pruning process and allows for a more efficient use of computational resources. In K-shots pruning, the number of filters to prune in each round is calculated based on the total number of filters in the network and the desired overall sparsity level. After each round of pruning, the pruned network can be fine-tuned to recover any loss in accuracy. This fine-tuning step can be repeated between rounds to further improve the performance of the pruned network. As a form of the multi-shot strategy, a major benefit of K-shots is that it allows one to benefit from the advantages of multi-shot pruning while maintaining a controlled computational complexity. By fixing the number of pruning rounds, we can ensure that the pruning process terminates after a specific number of steps, which can be useful in scenarios where computational resources are limited.

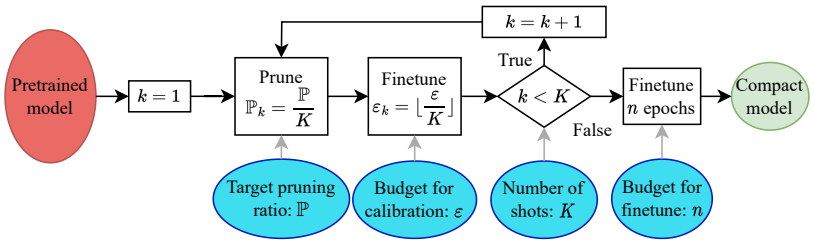


Figure 3: K-shots pruning strategy.

We denote by ε the predefined total number of calibrating epochs, $\mathbb{P}$ the target pruning ratio. The number of fine-tuning epochs ε_k (resp. the pruning ratio $\mathbb{P}_k$) after each shot is defined as:

$$\varepsilon_k = \lfloor \frac{\varepsilon}{K} \rfloor, \tag{9}$$

$$\mathbb{P}_k = \frac{\mathbb{P}}{K}. \tag{10}$$

Our strategy can improve performance by iterative fine-tuning like the multi-shot strategy while respecting the predefined budget for pruning as the one-shot strategy.

4. Experiments

4.1. Experimental Settings

Datasets and Architectures: CORING is evaluated on various benchmark datasets [28, 54, 42] with well-known and representative architectures including the classic plain structure VGG-16-BN [60], the GoogLeNet with inception modules [65], the ResNet-56 with residual blocks [22], the DenseNet-40 with dense blocks [24] and the MobileNetV2 with inverted residuals and linear bottlenecks [55]. Due to a large number of simulations, these models are all considered on CIFAR-10 [28]. Also, to validate the scalability of CORING, we conduct experiments on the challenging ImageNet dataset [54], with ResNet-50. Furthermore, the compressed ResNet-50 model is employed as the backbone network for FasterRCNN-FPN [53], MaskRCNN, and KeypointRCNN [21] on the COCO-2017 dataset [42].

Comparison. CORING is compared with 44 SOTAs in the fields of structured pruning [1, 3, 5, 6, 7, 11, 12, 15, 16, 18, 19, 23, 26, 29, 30, 31, 32, 34, 35, 36, 37, 38, 39, 40, 41, 43, 45, 46, 47, 48, 49, 56, 58, 63, 64, 66, 68, 73, 76, 78, 79, 80, 83, 85]. For a fair comparison, all available baseline models are identical. The reduction in parameters and MACs are kept similar and the accuracy is compared or vice versa.

Evaluation Protocols: The performance of the pruned model is assessed via three criteria: accuracy, required Float Points Operations (MACs), and number of parameters. The FLOP and parameter counts reflect the cost of computation and storage consumption. 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.

Configuration: The experiments were conducted on A40s using PyTorch. SGD is used as the optimizer for fine-tuning. After pruning, fine-tuning is performed for 300 epochs on CIFAR-10 with a batch size of 256, momentum of 0.9, weight decay of 0.05, and an initial learning rate of 0.01. On ImageNet, fine-tuning was performed for 180 epochs with a batch size of 512, momentum of 0.99, weight decay of 0.0001, and an initial learning rate of 0.1. In the case of K-shots strategy, the total number of calibration epochs was fixed at 100. On COCO, models are fine-tuned following the default recipe of torchvision [50].

4.2. Results and Analysis

Our proposed framework is flexible, enabling the combination of various distance metrics and the number of shots. For consistency, we present the results of CORING deployed with VBD and $K = 15$ in Tables 1-7. Ablation studies on the influences of distance metrics and K are discussed in Subsections 5.3 and 5.4, respectively.

VGG-16-BN. Table 1 shows the pruning results of VGGNet on CIFAR-10. In all compression levels, compared with other methods, CORING consistently gets the highest accuracy while maintaining the same level of pruning. Specifically, our method improves the model generalization by increasing the accuracy score from 93.96% to 94.42% while decreasing more than 81% of parameters. To provide a visual representation, we depict the accuracy-MACs reduction Pareto curves in Fig. 4, illustrating the effectiveness of CORING.

Table 1: Pruning results of VGG-16-BN on CIFAR-10

Model	Top1	Params (↓%)	MACs (↓%)
VGG-16-BN	93.96	14.98M(00.0)	313.73M(00.0)
CHIP [63]	93.86	2.76M(81.6)	131.17M(58.1)
EZCrop [40]	93.01	2.76M(81.6)	131.17M(58.1)
DECORE-500 [1]	94.02	5.54M(63.0)	203.08M(35.3)
FPAC [76]	94.03	2.76M(81.6)	131.17M(58.1)
DMPP [32]	94.18	N/A	166.59M(54.1)
CORING (Ours)	**94.42**	**2.76M(81.6)**	**131.17M(58.1)**
HRank-2 [38]	92.34	2.64M(82.1)	108.61M(65.3)
EZCrop [40]	93.70	2.50M(83.3)	104.78M(66.6)
CHIP [63]	93.72	2.50M(83.3)	104.78M(66.6)
FPAC [76]	93.86	2.50M(83.3)	104.78M(66.6)
APIB [19]	94.00	3.30M(77.9)	106.67M(66.0)
AutoBot [3]	94.01	6.44M(57.0)	108.71M(65.3)
CORING (Ours)	**94.20**	**2.50M(83.3)**	**104.78M(66.6)**
LAP [5]	89.95	N/A	75.01M(76.1)
QSFM [73]	92.17	3.68M(75.0)	79.00M(74.8)
RGP-64_16 [6]	92.76	3.81M(74.6)	78.78M (74.8)
CHIP [63]	93.18	1.90M(87.3)	66.95M(78.6)
FSM [12]	93.73	2.05M(86.3)	108.24M(66.0)
CORING (Ours)	**93.83**	**1.90M(87.3)**	**66.95M(78.6)**
RASP-70M [83]	92.81	1.13M(92.4)	70.15M(77.7)
CLR-RNF-0.86 [37]	93.32	0.74M(95.0)	81.31M(74.1)
DECORE-100 [1]	92.44	0.51M(96.6)	51.20M(81.5)
WhiteBox [80]	93.47	1.80M(87.8)	75.80M (75.9)
CORING (Ours)	**93.52**	**0.51M(96.6)**	**48.52M(84.6)**
HRank-3 [38]	91.23	1.78M(88.1)	73.70M(76.5)
FSM [12]	92.86	1.40M(90.6)	67.45M(81.0)
CORING (Ours)	**93.07**	**1.40M(90.6)**	**37.06M(88.3)**

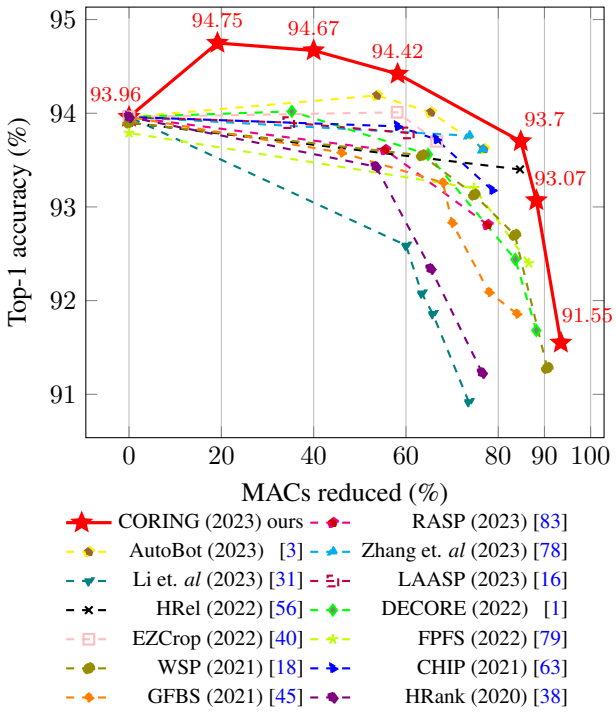


Figure 4: Pruning methods for VGG-16 baseline on CIFAR-10.

GoogLeNet. Table 2 shows pruning results of GoogLeNet on CIFAR-10. In all conducted cases, CORING consistently outperforms other methods [35, 37, 46] in every way. Therefore, inception modules can use our method to achieve high performance in comparison to cutting-edge techniques.

Table 2: Pruning results of GoogLeNet on CIFAR-10

Model	Top1	Params (↓%)	MACs (↓%)
GoogLeNet	95.05	6.15M(00.0)	1.52B(00.0)
DECORE-500 [1]	95.20	4.73M(23.0)	1.22B(19.8)
CORING (Ours)	**95.30**	**4.72M(23.3)**	**1.21B(20.4)**
L1 [29]	94.54	3.51M(42.9)	1.02B(32.9)
GAL-0.05 [41]	93.93	3.12M(49.3)	0.94B(38.2)
HRank-1 [38]	94.53	2.74M(55.4)	0.69M(54.9)
FPAC [76]	95.04	2.85M(53.5)	0.65B(57.2)
CC-0.5 [35]	95.18	2.83M(54.0)	0.76B(50.0)
CORING (Ours)	**95.32**	2.85M(53.5)	**0.65B(57.2)**
FPAC [76]	94.42	2.09M(65.8)	0.40B(73.9)
EACP($k = 30\%$) [46]	94.80	2.49M(59.6)	0.58B(62.3)
CLR-RNF-0.91 [37]	94.85	2.18M(64.7)	0.49B(67.9)
CC-0.6 [35]	94.88	2.26M(63.3)	0.61B(59.9)
CORING (Ours)	**95.03**	**2.10M(65.9)**	**0.39B(74.3)**

ResNet-56. Table 3 shows pruning results of ResNet on CIFAR-10. We can see that CORING can boost the accuracy by 1.5% compared to the baseline model with 22.4% and 27.3% model size and MACs reductions, respectively. In the scenario involving significant compression, with approximately 70% compression, CORING outperforms a

recent SOTA method [34] in all aspects.

Table 3: Pruning results of ResNet-56 on CIFAR-10

Model	Top1	Params (↓%)	MACs (↓%)
ResNet-56	93.26	0.85M(00.0)	125.49M(00.0)
HRank-1 [38]	93.52	0.71M(16.8)	88.72M(29.3)
DECORE-450 [1]	93.34	0.64M(24.2)	92.48M(26.3)
TPP [68]	93.81	N/A	86.50M(31.1)
CORING (Ours)	**94.76**	0.66M(22.4)	91.23M(27.3)
HRank-2 [38]	93.17	0.49M(42.4)	62.72M(50.0)
DECORE-200 [1]	93.26	0.43M(49.0)	62.93M(49.9)
TPP [68]	93.46	N/A	62.99M(49.8)
FSM [12]	93.63	.048M(43.6)	61.24M(51.2)
CC-0.5 [35]	93.64	0.44M(48.2)	60M(52.0)
FPAC [76]	93.71	0.48M(42.8)	65.94M(47.4)
ResRep [11]	93.71	N/A	59.3M(52.7)
DCP [43]	93.72	0.43M(49.7)	56.72M(54.8)
EZCrop [40]	93.80	0.48M(42.8)	65.94M(47.4)
Zhang et. *al* [78]	93.88	0.48M(42.8)	65.94M(47.4)
RASP-60M [83]	94.02	0.67M(22.2)	80.66M(36.5)
CHIP [63]	94.16	0.48M(42.8)	65.94M(47.4)
CORING (Ours)	**94.22**	0.48M(42.8)	65.94M(47.4)
HRank-3 [38]	90.72	0.27M(68.1)	32.52M(74.1)
LAP [5]	91.72	N/A	29.01M(76.9)
QSFM [73]	91.88	0.25M(71.3)	50.62M(60.0)
CHIP [63]	92.05	0.24M(71.8)	34.79M(72.3)
TPP [68]	92.35	N/A	36.89M(70.6)
FPAC [76]	92.37	0.24M(71.8)	34.79M(72.3)
Li et. *al* [34]	92.46	0.26M(70.0)	37.02M(70.5)
Zhang et. *al* [78]	92.48	0.24M(71.8)	34.79M(72.3)
CORING (Ours)	**92.84**	**0.24M(71.8)**	34.79M(72.3)

Table 4: Pruning results of DenseNet-40 on CIFAR-10

Model	Top1	Params (↓%)	MACs (↓%)
DenseNet-40	94.81	1.06M(00.0)	290.14M(00.0)
DECORE-175 [1]	94.85	0.83M(20.7)	228.96M(19.1)
CORING (Ours)	**94.88**	**0.80M(24.2)**	**224.12M(22.8)**
GAL-0.01 [41]	94.29	0.67M(35.6)	182.92M(35.3)
HRank-1 [38]	94.24	0.66M(36.5)	167.41M(40.8)
FPAC [76]	94.51	0.62M(40.1)	173.39M(38.5)
CORING (Ours)	**94.71**	**0.62M(41.2)**	173.39M(39.6)
FPAC [76]	93.66	0.39M(61.9)	113.08M(59.9)
HRank-2 [38]	93.68	0.48M(53.8)	110.15M(61.0)
EZCrop [40]	93.76	0.39M(61.9)	113.08M(59.9)
DECORE-70 [1]	94.04	0.37M(65.0)	128.13M(54.7)
CORING (Ours)	**94.30**	0.45M(57.3)	134.86M(53.5)

Table 5: Pruning results of MobileNetv2 on CIFAR-10

Model	Top1	Params (↓%)	MACs (↓%)
MobileNetv2	94.43	2.24M(00.0)	94.54M(00.0)
DCP [43]	94.02	1.71M(23.6)	69.58M(26.4)
CATRO [23]	94.27	N/A	55.21M(41.6)
CORING (Ours)	**94.81**	**1.26M(43.8)**	**55.16M(42.0)**
QSFM-PSNR [73]	92.06	1.67M(25.4)	57.27M(39.4)
DMC [15]	94.49	N/A	56.72M(40.0)
SCOP [66]	94.24	1.43M(36.1)	56.44M(40.3)
GFBS [45]	94.25	N/A	54.83M(42.0)
CORING (Ours)	**94.44**	**0.77M(65.6)**	**38.00M(60.0)**

DenseNet-40. Managing DenseNet architecture can be challenging because removing a single channel from the architecture requires removing that channel from all subsequent layers [1]. Table 4 shows the pruning results of DenseNet-40 on CIFAR-10. Again, with a soft compression, CORING allows to increase in the baseline accuracy while achieving 23.3% parameter compression and 20.4% MACs reduction. Compared with FPAC [76], CORING has advantages in all aspects.

MobileNetv2. Pruning MobileNet-v2 presents a significant challenge due to its exceedingly low computational cost. However, CORING exhibits superior performance compared to other candidate methods [45, 66], achieving a top-1 accuracy of 94.81% while pruning 42% of the network's MACs, as illustrated in Table 5. Even when compressing more than 60% of the network, the accuracy is not reduced, which suggests that CORING can be applied to optimize the hand-crafted design networks.

ResNet-50. To assess the scalability of CORING, we conduct experiments on the extensive dataset ImageNet by addressing ResNet-50 as shown in Table 6. Our approach excels at moderate compression ratios, achieving over 40% reductions while increasing accuracy by 0.63%. In contrast, DECORE [1] provides a mere 0.16% accuracy gain but is four times less efficient in terms of compression (11% reduction). When we further increase the compression ratio, our approach still achieves superior performance than SOTA. For instance, under the high MACs compression of 77%, we obtain an accuracy of 74%, outperforming very recent works including RASP [83] (71.05%), HRel [56] (73.67%) and the method presented by Zhang et. *al* [78] (73.18%).

Table 6: Pruning results of ResNet-50 on ImageNet

Model	Top1	Top5	Params (↓%)	MACs (↓%)
ResNet-50	76.15	92.87	25.55M(00.0)	4.11B(00.0)
AutoPruner-0.3 [48]	74.76	92.15	N/A	3.76B(08.1)
ABCPruner-100% [39]	72.84	92.97	18.02M(29.3)	2.56B(37.4)
CLR-RNF-0.2 [37]	74.85	92.31	16.92M(33.6)	2.45B(40.1)
WSP-40 [18]	75.49	92.57	17.12M(33.0)	2.51B(38.6)
APRS [64]	75.58	N/A	16.17M(35.4)	2.29B(44.0)
PFP [36]	75.91	92.81	20.88M(18.1)	3.65B(10.8)
LeGR [7]	76.20	93.00	N/A	3.01B(27.0)
MetaPruning-0.85 [47]	73.20	N/A	N/A	3.00B(27.0)
DECORE-8 [1]	76.31	93.02	22.69M(11.0)	3.54B(13.4)
CHIP [63]	76.30	93.02	15.10M(40.8)	2.26B(44.8)
TPP [68]	76.44	N/A	N/A	2.76B(32.9)
CORING (Ours)	**76.78**	**93.23**	**15.10M(40.8)**	**2.26B(44.8)**
AutoPruner-0.5 [48]	73.05	91.25	N/A	2.64B(35.5)
HRank-1 [38]	74.98	92.33	16.15M(36.7)	2.30B(43.8)
DECORE-6 [1]	74.58	92.18	14.10M(44.7)	2.36B(42.3)
PFP [36]	75.21	92.43	17.82M(30.1)	2.29B(44.0)
FPAC [76]	75.62	92.63	15.09M(40.9)	2.26B(45.0)
EZCrop [40]	75.68	92.70	15.09M(40.9)	2.26B(45.0)
LeGR [7]	75.70	92.70	N/A	2.38B(42.0)
SCOP [66]	75.95	92.79	14.59M(42.8)	2.24B(45.3)
DMPP [32]	75.44	92.69	N/A	2.21B(46.3)
Zhang et. *al* [78]	75.83	92.76	14.23M(44.2)	2.10B(48.7)
CLCS [85]	76.06	N/A	N/A	2.14B(48.1)
CHIP [63]	76.15	92.91	14.23M(44.2)	2.10B(48.7)
CORING (Ours)	**76.34**	**93.06**	**14.23M(44.2)**	**2.10B(48.7)**
HRank-2 [38]	71.98	91.01	13.77M(46.0)	1.55B(62.1)
MFMI [58]	72.02	90.69	11.41M(55.2)	1.84B(55.0)
WSP-60 [18]	73.91	91.66	11.60M(54.6)	1.55B(62.1)
FPAC [76]	74.17	91.84	11.05M(56.7)	1.52B(62.8)
EZCrop [40]	74.33	92.00	11.05M(56.7)	1.52B(62.8)
RASP-1G [83]	74.48	92.02	16.29M(36.3)	1.50B(63.6)
HRel-2 [56]	74.54	92.12	13.23M(48.2)	1.69B(58.9)
APRS [64]	74.72	N/A	N/A(52.9)	1.76B(57.2)
DMPP [32]	74.78	92.40	N/A	1.81B(56.1)
Zhang et. *al* [78]	74.80	92.39	11.04M(56.7)	1.52B(62.8)
TPP [68]	75.12	N/A	N/A	1.61B(60.9)
Li et. *al* [34]	75.24	N/A	N/A	1.61B(60.9)
SCOP [66]	75.26	92.53	12.29M(51.8)	1.86B(54.6)
CHIP [63]	75.26	92.53	11.04M(56.7)	1.52B(62.8)
LeGR [7]	75.30	92.40	N/A	1.93B(53.0)
ResRep [11]	75.30	92.47	N/A	1.52B(62.1)
CORING (Ours)	**75.55**	**92.61**	**11.04M(56.7)**	**1.50B(63.6)**
MFMI [58]	69.91	89.46	8.51M(66.6)	1.41B(34.4)
RASP-1G [83]	71.05	90.13	8.81M(65.6)	1.00B(75.7)
WSP-70 [18]	72.04	90.82	9.07M(64.5)	1.12B(72.6)
DECORE-5 [1]	72.06	90.82	8.87M(65.2)	1.60B(60.9)
FPAC [76]	72.30	90.74	8.02M(68.6)	0.95B(76.7)
HRank-3 [38]	72.30	90.74	8.27M(67.6)	0.98B(76.0)
ABCPruner-50% [39]	72.58	90.91	9.10M(64.3)	1.30B(68.2)
CLR-RNF-0.44 [37]	72.67	91.09	9.00M(64.7)	1.23B(69.9)
Zhang et. *al* [78]	73.18	91.32	8.01M(68.6)	0.95B(76.7)
CHIP [63]	73.54	90.58	8.01M(68.6)	0.95B(76.7)
MetaPruning-0.5 [47]	73.40	N/A	N/A	1.00B(75.7)
HRel-3 [56]	73.67	91.70	9.10M(64.4)	1.38B(66.4)
CORING (Ours)	**74.00**	**91.71**	**8.01M(68.6)**	**0.95B(76.7)**

Faster/Mask/Keypoint-RCNN. The results presented in Table 7 underscore the practical advantages of the CORING method when applied to downstream tasks like Faster/Mask/Keypoint-RCNN on the COCO dataset. By employing our compressed ResNet-50/Imagenet model as the backbone network, we achieved substantial reductions in

Table 7: 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(↓%)	Params(↓%)	FPS
FasterRCNN [53, 50]	36.91	58.53	39.61	30.73	48.46	50.84	134.85G(00)	41.81M(00)	12
CORING (Ours)	35.57	56.05	37.81	30.21	48.28	50.79	**92.23G(32)**	**24.04M(43)**	**25**
MaskRCNN [21, 50]	34.54	55.97	36.81	29.53	45.47	47.45	134.85G(00)	44.46M(00)	10
CORING (Ours)	32.77	53.53	34.57	28.93	44.94	47.11	**92.23G(32)**	**26.68M(40)**	**22**
				$AR^{0.5:0.95}$	$AR^{0.5}$	$AR^{0.75}$			
KeypointRCNN [21, 50]	65.05	86.08	71.37	71.75	90.71	77.42	137.42G(00)	59.19M(00)	9
CORING (Ours)	64.24	86.01	69.91	70.94	90.57	76.01	**96.59G(30)**	**41.42M(30)**	**17**

computational complexity and the number of parameters, highlighting the efficiency of our approach. Particularly noteworthy is the significant boost in inference throughput, with CORING achieving up to a $2\times$ improvement in Frames Per Second (FPS) compared to the baseline models. It's worth emphasizing that these performance measurements were conducted on an RTX 3060 GPU, providing tangible evidence of the real-world applicability of our approach. These outcomes underscore CORING's potential as a valuable tool for enhancing the efficiency and effectiveness of neural network models across challenging tasks such as object detection, instance segmentation, and keypoint detection in real-world scenarios.

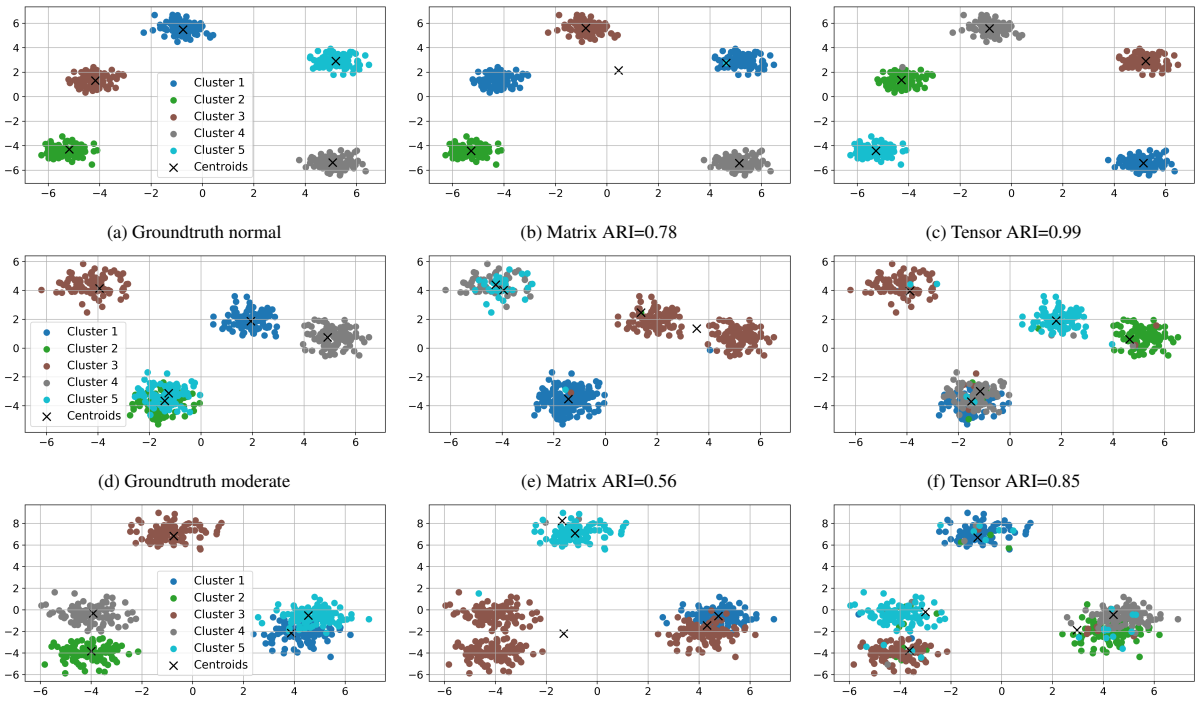
5. Discussions

5.1. The Advantages of CORING in Comparison to Flatten-based Approaches

Our low-rank tensor approximation method outperforms conventional techniques that flatten filters in terms of computational efficiency and effectiveness [8, 61, 71, 79]. Generally, to obtain the similarity matrix of a layer with N filters, the complexity of these methods is $\mathcal{O}(Nchw) + \mathcal{O}(N^2chw)$, while our method requires $\mathcal{O}(Nchw) + \mathcal{O}(N^2max(c,h,w))$. The justification for these complexities lies in the detailed computation of the pairwise distances. When considering the flattened filters, the pairwise distance computation incurs a complexity of $\mathcal{O}(chw)$ for each pair, leading to a total complexity of $\mathcal{O}(N^2chw)$ for all pairwise. In contrast, our approach, after applying HOSVD, simplifies this process. Given that the complexity for computing the r-truncated SVD for a matrix of size $m\times n$ is $\mathcal{O}\left(r(m+n)+rmn\right)$ [17], and considering that the HOSVD involves three rank-1 SVDs on matrices resized according to the tensor dimensions c, h and w, the HOSVD complexity is $\mathcal{O}(chw)$. The subsequent pairwise distance calculation, then, is $\mathcal{O}(c+h+w)$ for each pair, culminating in a total complexity of $\mathcal{O}(Nchw)+\mathcal{O}\left(N^2max(c,h,w)\right)$ after including the HOSVD step. Note that for common architectures, the number of input channels of each filter is usually higher than the kernel size. For example, for the 10-th convolution layer of VGG-16, $\{c,h,w\}=\{512,3,3\}$, so our method's complexity is $\mathcal{O}(N^2c)$. In this case, CORING is 9 times faster than the other methods. This proves that our approach is computationally more efficient in dealing with larger and deeper models with increasing numbers of layers and filters. Furthermore, when applying iterative pruning or pruning during training, which requires updating the similarity matrix at each iteration, our method can significantly reduce computational costs. In terms of effectiveness, the low-rank representation in CORING preserves the multidimensional nature of filters. This preservation offers a more efficient and accurate means of measuring similarity compared to traditional methods that rely on vectorized or matricized filter representations. Consequently, CORING facilitates more efficient filter pruning without compromising valuable information. It's crucial to note that low-rank matrix approximation methods are not as effective as low-rank tensor approximation methods, and may result in loss of information. This is an easy and intuitive example that demonstrates the advantages of using tensor decompositions over matrix decompositions for higher-dimensional data in the case of rank decomposition: a 3-order $6\times6\times6$ tensor can be decomposed uniquely [27] into 8 rank-1 tensors, while flattening the tensor into a matrix yields a 6×36 matrix that can be approximated by only 6 rank-1 matrices. This illustrates that tensor decompositions are able to extract more meaningful components or factors from the data, which is particularly important for higher-dimensional data.

5.2. Comparative Evaluation of Tensor-Based and Matrix-Based Approaches

To assess the comparative efficacy of tensor-based and matrix-based methodologies, we conduct an experimental study utilizing the K-means clustering algorithm with custom distance metrics. We generate a set of synthetic datasets



(a) Groundtruth normal (b) Matrix ARI=0.78 (c) Tensor ARI=0.99

(d) Groundtruth moderate (e) Matrix ARI=0.56 (f) Tensor ARI=0.85

(g) Groundtruth hard (h) Matrix ARI=0.48 (i) Tensor ARI=0.72

Figure 5: Visualizing the performance of tensor-based and matrix-based methods.

(D in total) characterized by c clusters, where each cluster features a centroid tensor of dimensions $C_{in} \times h \times w$, along with n filters of identical dimensions. The dataset complexity is governed by the standard deviations of centroids ($\sigma_{centroids}$) and filters ($\sigma_{filters}$). We evaluate the K-means algorithm's performance using the Adjusted Rand Index (ARI) $\alpha \in [-1, 1]$, a measure of similarity between ground truth and predictions [25]. Higher α values indicate superior predictive accuracy (e.g., $\alpha = 0$ implies predictions are no better than random chance concerning ground truth). Due to the initialization sensitivity of the K-means algorithm, we perform i initializations and select the one yielding the best inertia value, defined as the sum of distances between samples and their corresponding centroids, upon convergence, following sklearn guidelines [2]. All random processes in our study follow continuous uniform distributions. Our experimental settings are as follows: $D = 1000$, $c = 5$, $n = 100$, $C_{in} = 64$, $h = w = 3$, and $i = 100$. We design datasets with three difficulty levels: normal ($\sigma_{centroids} \in [1.7, 1.8]$, $\sigma_{filters} \in [0.2, 0.3]$), moderate ($\sigma_{centroids} \in [1.5, 2.0]$, $\sigma_{filters} \in [0.1, 0.3]$), and hard ($\sigma_{centroid} \in [1.0, 2.0]$, $\sigma_{filters} \in [0.1, 0.5]$). The mean ARIs, as presented in Table 8, unequivocally establish the superior performance of the tensor-based approach. Both approaches exhibit proficiency on less intricate datasets; however, in scenarios where dataset complexity escalates, our tensor-based methodology consistently demonstrates heightened consistency and effectiveness. This enhanced performance can be attributed to our method's ability to preserve the multidimensionality of the filters, allowing it to capture and retain crucial information effectively.

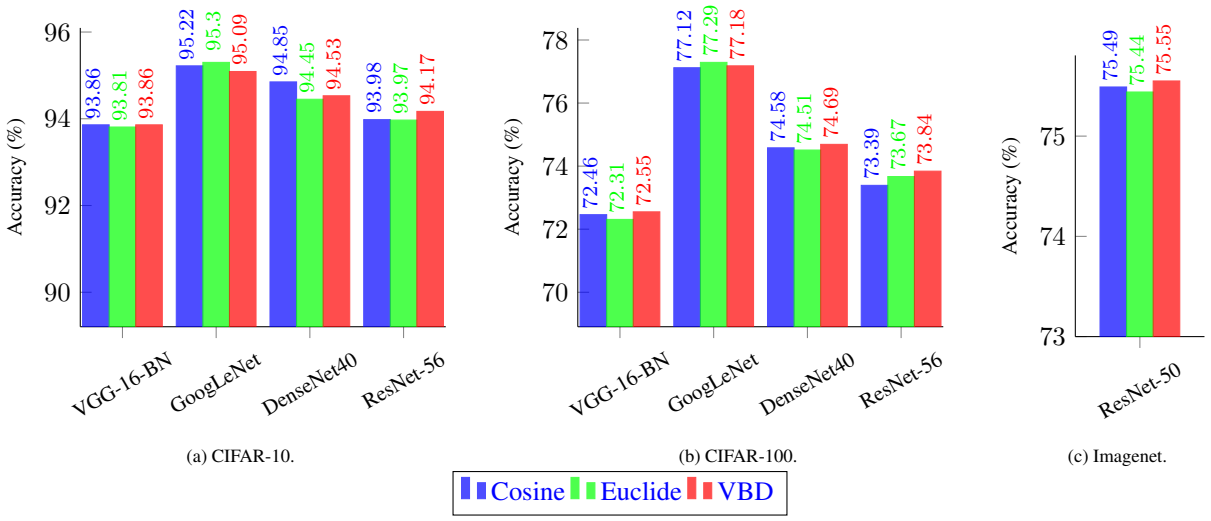
We choose representative datasets based on the ARI of predictions that closely align with the mean ARI across the entire dataset (3 scenarios, each containing 1000 datasets). These representative datasets are employed to illustrate typical outcomes of both methods, as presented in Fig. 5. It is worth noting that the Principal Component Analysis (PCA) technique is employed for visualizing the three-dimensional samples. These visualizations not only align with the numerical results but also showcase the prowess of the tensor-based method, further reinforcing the evidence of its superiority.

Table 8: ARI of the tensor-based and matrix-based method.

Method	Dataset difficulty		
	Normal	Moderate	Hard
Matrix	0.83	0.61	0.54
Tensor	**0.91**	**0.82**	**0.73**

5.3. Distance Metrics

Figure 6 provides a comprehensive visualization of the influence of different distance metrics on the accuracy of diverse neural network architectures across various datasets. In Subfigure 6a, we assess the accuracy on the CIFAR-10 dataset for prominent architectures, including VGG, GoogLeNet, DenseNet, and ResNet. The chosen metrics, encompassing Cosine, Euclidean, and VBD, exhibit distinct impacts on the model accuracies. Extending this analysis to the CIFAR-100 and Imagenet datasets in Subfigure 6b and Subfigure 6c, respectively, provides valuable insights into the robustness of these metrics across diverse datasets. This empirical exploration underscores the importance of choosing appropriate metrics tailored to specific neural network architectures and datasets. While VBD, derived from SNR, consistently demonstrates promising results in the majority of cases (6 out of 9), a theoretical justification for the superiority of a particular metric remains elusive. It is crucial to clarify that VBD is not the primary focus of our contribution. Instead, our work centers around incorporating a multidimensional structure for more accurate tensor filter metrics, as demonstrated in comparison with SOTA methods in Section 4. It is imperative to emphasize that this empirical evidence does not offer a conclusive verdict on the optimal distance metric for use with CORING. The selection of a specific metric is contingent on various factors, including the application context and dataset characteristics.



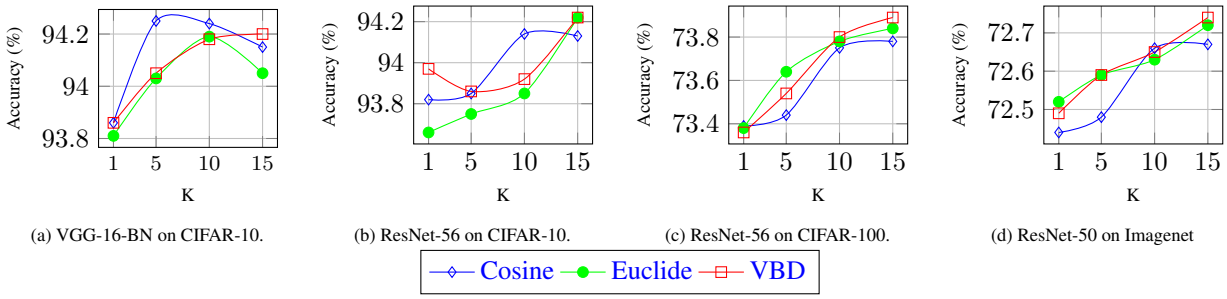
(a) CIFAR-10. (b) CIFAR-100. (c) Imagenet.

Figure 6: The influence of distance metrics on model accuracy for different architectures and datasets.

5.4. K-shots Analysis

To thoroughly examine the impact of the K-shots strategy, with a specific emphasis on the influence of the parameter K, we undertake a comprehensive ablation study. This investigation delves into the ramifications of varying the parameter K on the accuracy of various architectures across diverse datasets. The results, depicted in Fig. 7, are expounded in Subfigures 7a, 7b, and 7c, illustrating outcomes for the VGG-16-BN and ResNet-56 architectures on CIFAR-10 and CIFAR-100 datasets, respectively. Furthermore, Subfigure 7d provides insights into the performance of the ResNet-50 architecture on the ImageNet dataset. In each subfigure, three distinct distance metrics (Cosine,

Euclidean, and VBD) are plotted against the achieved accuracy at different K values (1, 5, 10, 15). The outcomes underscore that the adoption of multi-shot pruning strategies, denoted by K-shots, consistently outperforms single-shot pruning in the majority of cases (11 out of 12). Notably, on CIFAR-10, where the dataset size is relatively modest, the accuracy trend concerning K appears ambiguous, as observed in subfigures 7a and 7b. Conversely, on more intricate datasets, such as CIFAR-100 and ImageNet, the accuracy trend implies that higher values of K correlate with enhanced accuracy, evident in subfigures 7c and 7d. This can be attributed to the complexity of the task, where each round of the pruning process may incur errors, and a heightened value of K proves beneficial in mitigating such errors. In summary, these visualizations offer nuanced insights into how the selection of K significantly influences model accuracy across varying distance metrics, facilitating the optimization of model performance for specific tasks and datasets. Nevertheless, it is imperative to acknowledge that determining the optimal number of shots is a nuanced undertaking, as the efficacy of this parameter may fluctuate based on specific contextual factors and requirements.



(a) VGG-16-BN on CIFAR-10. (b) ResNet-56 on CIFAR-10. (c) ResNet-56 on CIFAR-100. (d) ResNet-50 on Imagenet

Figure 7: The influence of K, the number of shots, on model accuracy for different architectures and datasets.

5.5. *Pruning Efficiency Analysis*

To assess the time overheads and accuracy trade-offs associated with the K-shots pruning strategy, we conducted an experiment on VGG-16/CIFAR, measuring both pruning and fine-tuning times. As indicated in Table 9, pruning time increases with K since the importance of filters needs recalculation in each round. However, this increment is relatively small compared to fine-tuning time. Given that the number of epochs for fine-tuning (e.g., 300 epochs) remains invariant to K, the difference in fine-tuning time is insignificant. Consequently, while the total time increases with K, it remains comparable in all cases. The accuracy achieved with the K-shots strategy surpasses that of the one-shot strategy. Nevertheless, determining the optimal number of shots is non-trivial, as its effectiveness may vary depending on specific contextual factors and requirements.

Table 9: Time overhead vs accuracy trade-off varied by K - the number of shots

K	Pruning time (s)	Finetuning time (s)	Total time (s)	Accuracy (%)
1	138	3925	4063	93.63
5	492	3931	4423	93.71
10	810	3935	4745	93.68
15	975	3942	4917	93.83

Considering the pruning strategy, there exist two representative approaches: Pruning-and-Fine-tuning (PaF) [38, 63] and Training-aware-Pruning (TaP) [80]. Each approach has its pros and cons. While TaP brings many benefits such as dynamic adaptation and reduced fine-tuning overheads, it also has some inconveniences, such as increased training complexity, potential for slow training convergence, and compatibility issues. In this work, we adopted the PaF scheme due to its conceptual straightforwardness, allowing for ease of implementation, and its compatibility with pre-trained models, making it a practical choice for compression. We conduct an experiment on VGG-16-BN/CIFAR, comparing the time overheads of CORING and WhiteBox [80], a representative TaP method. The main overheads arise from the number of training epochs. Using the baseline model from HRank [38], CORING requires 300 epochs

to fine-tune the pruned network, while WhiteBox first trains the full network (heavier than the target sparse network) for 30 epochs and then continues fine-tuning the pruned network for 300 epochs. In the pruning phase, WhiteBox's training is more complicated due to embedding the class-aware mask into the baseline network, making it take 2901 seconds for only 30 epochs, compared to CORING's 138 seconds for the pruning step. The total time consumption of CORING and WhiteBox is 4063 and 6905 seconds, respectively. CORING achieves a pruned model with better performance (93.63% vs 93.47%) and higher MACs reduction (79% vs 76%). Although CORING might take more time if it has to train the baseline model from scratch, pre-trained models are readily available, such as those provided by PyTorch [50]. Consequently, in our view, if the baseline model is available, CORING incurs less overhead than WhiteBox, and vice versa.

5.6. Qualitative Assessment of Feature Preservation

In this subsection, we provide a qualitative assessment of filter preservation within the context of CORING. While the numerical results in Section 4 have already established the efficiency of CORING, our aim is to offer deeper insights by exploring the preservation of crucial filters and features.

To achieve this, we randomly selected 8 images from the ImageNet validation dataset and conducted a comprehensive evaluation. Our approach involved three levels of pruning ratios for the original ResNet-50 model, corresponding to 49%, 64%, and 77% reduction in MACs operations. The baseline model, ResNet-50, initially boasted an accuracy of 76.15%. After applying CORING and the respective pruning techniques, we observed accuracy scores of 76.34%, 75.55%, and 74% for the three levels of compression, respectively (see Table 6).

To gain a deeper understanding of filter preservation, we utilize GradCAM [57], a standard tool for neural network explanation and interpretation, to visualize and analyze feature maps in the original and compressed models. In Fig. 8, we showcase the effectiveness of CORING in retaining vital features of the original models across a diverse range of classes, including humans, animals, vehicles, objects, and architectures. Additionally, it's worth noting that under different levels of pruning ratios, CORING consistently demonstrates its robustness in capturing and preserving essential information. This robustness implies that CORING maintains its effectiveness and reliability in different scenarios and under varying pruning ratios, making it a versatile choice for filter pruning across a wide range of applications and datasets.

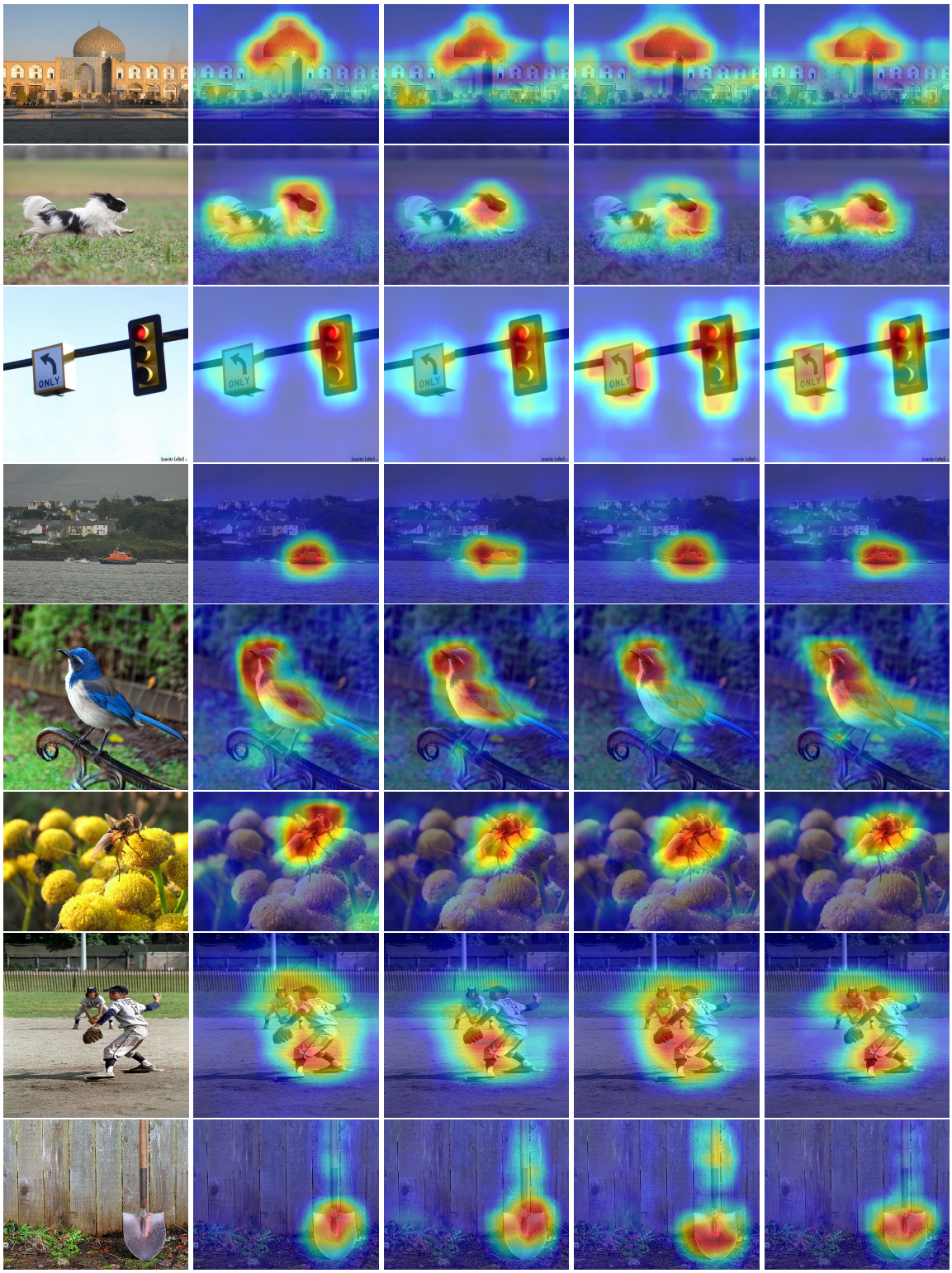
These observations align with our quantitative results, emphasizing the robustness and effectiveness of CORING in neural network compression. By leveraging GradCAM, we substantiate our argument that CORING is capable of retaining important filters and features, contributing to its superior performance.

6. Conclusion

In conclusion, we have introduced CORING, a novel filter pruning method designed around a tensor decomposition approach that effectively preserves the multidimensional nature of filters and seamlessly integrates with various metrics and pruning strategies. Our introduction of the K-shots pruning strategy achieves a balanced trade-off between accuracy and predetermined pruning budgets. Key contributions of our work include the development of a tensor-based approach for network compression and the introduction of a filter selection algorithm grounded in the similarity matrix. Furthermore, our method has demonstrated a remarkable ability to enhance model generalization through pruning, showcasing its versatility across different datasets and a wide range of network architectures. The extensive evaluation results underscore the efficiency and effectiveness of this innovative approach to network pruning, providing valuable insights for advancing neural network performance.

Acknowledgment

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) Query image (b) Original model (c) 49% pruned (d) 64% pruned (e) 77% pruned

Figure 8: Qualitative assessment of feature preservation in pruned networks.

References

[1] Alwani, M., Madhavan, V., Wang, Y., 2022. Decore: Deep compression with reinforcement learning. CVPR .

[2] Buitinck, L., Louppe, G., Blondel, M., Pedregosa, F., Mueller, A., Grisel, O., Niculae, V., Prettenhofer, P., Gramfort, A., Grobler, J., Layton, R., VanderPlas, J., Joly, A., Holt, B., Varoquaux, G., 2013. API design for machine learning software: experiences from the scikit-learn project, in: ECML PKDD Workshop: Languages for Data Mining and Machine Learning, pp. 108–122.

[3] Castells, T., Yeom, S.K., 2023. Automatic neural network pruning that efficiently preserves the model accuracy, in: AAAI.

[4] Chang, J., Lu, Y., Xue, P., Xu, Y., Wei, Z., 2023. Iterative clustering pruning for convolutional neural networks. Knowledge-Based Systems 265, 110386. URL: https://www.sciencedirect.com/science/article/pii/S0950705123001363, doi:https://doi.org/10.1016/j.knosys.2023.110386.

[5] Chen, Z., Liu, C., Yang, W., Li, K., Li, K., 2022. Lap: Latency-aware automated pruning with dynamic-based filter selection. Neural Networks 152, 407–418. URL: https://www.sciencedirect.com/science/article/pii/S0893608022001745, doi:https://doi.org/10.1016/j.neunet.2022.05.002.

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

[7] Chin, T.W., Ding, R., Zhang, C., Marculescu, D., 2019. Towards efficient model compression via learned global ranking. CVPR , 1515–1525.

[8] Chu, C., Chen, L., Gao, Z., 2020. Similarity based filter pruning for efficient super-resolution models, in: 2020 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB), pp. 1–7.

[9] De Lathauwer, L., De Moor, B., Vandewalle, J., 2000. A multilinear singular value decomposition. SIAM Journal on Matrix Analysis and Applications , 1253–1278.

[10] Ding, X., Ding, G., Han, J., Tang, S., 2018. Auto-balanced filter pruning for efficient convolutional neural networks, in: AAAI Conference on Artificial Intelligence.

[11] Ding, X., Hao, T., Tan, J., Liu, J., Han, J., Guo, Y., Ding, G., 2021. Resrep: Lossless cnn pruning via decoupling remembering and forgetting, in: ICCV, pp. 4490–4500.

[12] Duan, Y., Zhou, Y., He, P., Liu, Q., Duan, S., Hu, X., 2022. Network pruning via feature shift minimization, in: Proceedings of the Asian Conference on Computer Vision, pp. 4044–4060.

[13] Eckart, C., Young, G.M., 1936. The approximation of one matrix by another of lower rank. Psychometrika 1, 211–218.

[14] Frankle, J., Carbin, M., 2019. The lottery ticket hypothesis: Training pruned neural networks. ICLR .

[15] Gao, S., Huang, F., Pei, J., Huang, H., 2020. Discrete model compression with resource constraint for deep neural networks, in: CVPR, pp. 1896–1905.

[16] Ghimire, D., Lee, K., heum Kim, S., 2023. Loss-aware automatic selection of structured pruning criteria for deep neural network acceleration. Image and Vision Computing 136, 104745. URL: https://www.sciencedirect.com/science/article/pii/S0262885623001191, doi:https://doi.org/10.1016/j.imavis.2023.104745.

[17] Golub, G.H., Van Loan, C.F., 1996. Matrix Computations. Third ed., The Johns Hopkins University Press.

[18] Guo, Q., Wu, X.J., Kittler, J., Feng, Z., 2021. Weak sub-network pruning for strong and efficient neural networks. Neural Networks 144, 614–626. URL: https://www.sciencedirect.com/science/article/pii/S0893608021003658, doi:https://doi.org/10.1016/j.neunet.2021.09.015.

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

[20] Harshman, R.A., 1970. Foundations of the parafac procedure: Models and conditions for an 'explanatory' multi-modal factor analysis. UCLA Working Papers in Phonetics 16, 1–84.

[21] He, K., Gkioxari, G., Dollár, P., Girshick, R., 2020. Mask r-cnn. IEEE Transactions on Pattern Analysis and Machine Intelligence 42, 386–397.

[22] He, K., Zhang, X., Ren, S., Sun, J., 2016. Deep residual learning for image recognition, in: CVPR, pp. 770–778.

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

[24] Huang, G., Liu, Z., Van Der Maaten, L., Weinberger, K.Q., 2017. Densely connected convolutional networks, in: CVPR, pp. 2261–2269.

[25] Hubert, L., Arabie, P., 1985. Comparing partitions. Journal of Classification 2, 193–218.

[26] Jiang, P., Xue, Y., Neri, F., 2023. Convolutional neural network pruning based on multi-objective feature map selection for image classification. Applied Soft Computing 139, 110229. URL: https://www.sciencedirect.com/science/article/pii/S1568494623002478, doi:https://doi.org/10.1016/j.asoc.2023.110229.

[27] Kolda, T.G., Bader, B.W., 2009. Tensor decompositions and applications. SIAM Review , 455–500.

[28] Krizhevsky, A., Hinton, G., 2009. Learning multiple layers of features from tiny images. Technical Report 0. University of Toronto. Toronto, Ontario.

[29] Li, H., Kadav, A., Durdanovic, I., Samet, H., Graf, H.P., 2016. Pruning filters for efficient convnets. ICLR .

[30] Li, H., Ma, C., Xu, W., Liu, X., 2020. Feature statistics guided efficient filter pruning, in: IJCAI.

[31] Li, J., Shao, H., Zhai, S., Jiang, Y., Deng, X., 2023a. A graphical approach for filter pruning by exploring the similarity relation between feature maps. Pattern Recognition Letters 166, 69–75. URL: https://www.sciencedirect.com/science/article/pii/S0167865522003968, doi:https://doi.org/10.1016/j.patrec.2022.12.028.

[32] Li, J., Zhao, B., Liu, D., 2022. Dmpp: Differentiable multi-pruner and predictor for neural network pruning. Neural Networks 147, 103–112. URL: https://www.sciencedirect.com/science/article/pii/S0893608021004998, doi:https://doi.org/10.1016/j.neunet.2021.12.020.

[33] Li, N., Pan, Y., Chen, Y., Ding, Z., Zhao, D., Xu, Z., 2021a. Heuristic rank selection with progressively searching tensor ring network. Complex & Intelligent Systems , 1–15.

[34] Li, Y., van Gemert, J.C., Hoefler, T., Moons, B., Eleftheriou, E., Verhoef, B.E., 2023b. Differentiable transportation pruning, in: ICCV.

[35] Li, Y., Lin, S., Liu, J., Ye, Q., Wang, M., Chao, F., Yang, F., Ma, J., Tian, Q., Ji, R., 2021b. Towards compact cnns via collaborative compression, in: CVPR, pp. 6434–6443.
[36] Liebenwein, L., Baykal, C., Lang, H., Feldman, D., Rus, D., 2020. Provable filter pruning for efficient neural networks, in: ICLR.
[37] Lin, M., Cao, L., Zhang, Y., Shao, L., Lin, C.W., Ji, R., 2022a. Pruning networks with cross-layer ranking & k-reciprocal nearest filters. IEEE Transactions on Neural Networks and Learning Systems , 1–10.
[38] Lin, M., Ji, R., Wang, Y., Zhang, Y., Zhang, B., Tian, Y., Shao, L., 2020. Hrank: Filter pruning using high-rank feature map. CVPR .
[39] Lin, M., Ji, R., Zhang, Y., Zhang, B., Wu, Y., Tian, Y., 2021. Channel pruning via automatic structure search, in: Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence.
[40] Lin, R., Ran, J., Wang, D., Chiu, K., Wong, N., 2022b. Ezcrop: Energy-zoned channels for robust output pruning, in: WACV, pp. 3595–3604.
[41] Lin, S., Ji, R., Yan, C., Zhang, B., Cao, L., Ye, Q., Huang, F., Doermann, D.S., 2019. Towards optimal structured cnn pruning via generative adversarial learning. CVPR , 2785–2794.
[42] Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L., 2014. Microsoft coco: Common objects in context, in: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (Eds.), Computer Vision – ECCV 2014, Springer International Publishing, Cham. pp. 740–755.
[43] Liu, J., Zhuang, B., Zhuang, Z., Guo, Y., Huang, J., Zhu, J., Tan, M., 2022. Discrimination-aware network pruning for deep model compression. IEEE Transactions on Pattern Analysis and Machine Intelligence , 4035–4051.
[44] Liu, S., Chen, T., Chen, X., Atashgahi, Z., Yin, L., Kou, H., Shen, L., Pechenizkiy, M., Wang, Z., Mocanu, D.C., 2021a. Sparse training via boosting pruning plasticity with neuroregeneration. Advances in Neural Information Processing Systems 34, 9908–9922.
[45] Liu, X., Li, B., Chen, Z., Yuan, Y., 2021b. Exploring gradient flow based saliency for dnn model compression, in: Proceedings of the 29th ACM International Conference on Multimedia, p. 3238–3246.
[46] Liu, Y., Wu, D., Zhou, W., Fan, K., Zhou, Z., 2023. Eacp: An effective automatic channel pruning for neural networks. Neurocomputing 526, 131–142. URL: https://www.sciencedirect.com/science/article/pii/S0925231223000255, doi:https://doi.org/10.1016/j.neucom.2023.01.014.
[47] Liu, Z., Mu, H., Zhang, X., Guo, Z., Yang, X., Cheng, K.T., Sun, J., 2019. Metapruning: Meta learning for automatic neural network channel pruning, in: Proceedings of the IEEE/CVF international conference on computer vision, pp. 3296–3305.
[48] Luo, J.H., Wu, J., 2020a. Autopruner: An end-to-end trainable filter pruning method for efficient deep model inference. Pattern Recognition 107, 107461. URL: https://www.sciencedirect.com/science/article/pii/S0031320320302648, doi:https://doi.org/10.1016/j.patcog.2020.107461.
[49] Luo, J.H., Wu, J., 2020b. Neural network pruning with residual-connections and limited-data, in: CVPR, pp. 1455–1464.
[50] maintainers, T., contributors, 2016. Torchvision: Pytorch's computer vision library. https://github.com/pytorch/vision.
[51] Pan, Y., Xu, J., Wang, M., Ye, J., Wang, F., Bai, K., Xu, Z., 2019. Compressing recurrent neural networks with tensor ring for action recognition, in: Proceedings of the AAAI Conference on Artificial Intelligence, pp. 4683–4690.
[52] Pham, V.T., Zniyed, Y., Nguyen, T.P., 2024. Enhanced network compression through tensor decompositions and pruning. IEEE Transactions on Neural Networks and Learning Systems , 1–13doi:10.1109/TNNLS.2024.3370294.
[53] Ren, S., He, K., Girshick, R., Sun, J., 2017. Faster r-cnn: Towards real-time object detection with region proposal networks. IEEE Transactions on Pattern Analysis and Machine Intelligence 39, 1137–1149.
[54] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M.S., Berg, A.C., Fei-Fei, L., 2014. Imagenet large scale visual recognition challenge. IJCV , 211–252.
[55] Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.C., 2018. Mobilenetv2: Inverted residuals and linear bottlenecks, in: CVPR, pp. 4510–4520.
[56] Sarvani, C., Ghorai, M., Dubey, S.R., Basha, S.S., 2022. Hrel: Filter pruning based on high relevance between activation maps and class labels. Neural Networks 147, 186–197. URL: https://www.sciencedirect.com/science/article/pii/S0893608021004962, doi:https://doi.org/10.1016/j.neunet.2021.12.017.
[57] Selvaraju, R.R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., Batra, D., 2020. Grad-cam: Visual explanations from deep networks via gradient-based localization. Int. J. Comput. Vision 128, 336–359.
[58] Shao, L., Zuo, H., Zhang, J., Xu, Z., Yao, J., Wang, Z., Li, H., 2021. Filter pruning via measuring feature map information. Sensors (Basel) , 6601.
[59] Sidiropoulos, N.D., De Lathauwer, L., Fu, X., Huang, K., Papalexakis, E.E., Faloutsos, C., 2017. Tensor decomposition for signal processing and machine learning. IEEE Transactions on Signal Processing , 3551–3582.
[60] Simonyan, K., Zisserman, A., 2015. Very deep convolutional networks for large-scale image recognition. ICLR .
[61] Singh, A., Plumbley, M.D., 2022. A passive similarity based cnn filter pruning for efficient acoustic scene classification. arXiv preprint arXiv:2203.15751 .
[62] Singh, P., Verma, V.K., Rai, P., Namboodiri, V.P., 2020. Leveraging filter correlations for deep model compression. WACV .
[63] Sui, Y., Yin, M., Xie, Y., Phan, H., Zonouz, S., Yuan, B., 2021. Chip: Channel independence-based pruning for compact neural networks, in: NeurIPS.
[64] Sun, Q., Cao, S., Chen, Z., 2022. Filter pruning via automatic pruning rate search, in: ACCV, pp. 4293–4309.
[65] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A., 2015. Going deeper with convolutions, in: CVPR, pp. 1–9.
[66] Tang, Y., Wang, Y., Xu, Y., Tao, D., XU, C., Xu, C., Xu, C., 2020. Scop: Scientific control for reliable neural network pruning, in: NeurIPS.
[67] Tucker, L.R., 1966. Some mathematical notes on three-mode factor analysis. Psychometrika 31, 279–311.
[68] Wang, H., Fu, Y., 2023. Trainability preserving neural pruning, in: ICLR.
[69] Wang, M., Pan, Y., Yang, X., Li, G., Xu, Z., 2023. Tensor networks meet neural networks: A survey. arXiv preprint arXiv:2302.09019 .
[70] Wang, W., Sun, Y., Eriksson, B., Wang, W., Aggarwal, V., 2018. Wide compression: Tensor ring nets, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 9329–9338.
[71] Wang, W., Yu, Z., Fu, C., Cai, D., He, X., 2021a. Cop: customized correlation-based filter level pruning method for deep cnn compression.

Neurocomputing 464, 533–545. URL: https://www.sciencedirect.com/science/article/pii/S0925231221012959, doi:https://doi.org/10.1016/j.neucom.2021.08.098.
[72] Wang, Z., Li, C., Wang, X., 2021b. Convolutional neural network pruning with structural redundancy reduction, in: CVPR, pp. 14908–14917.
[73] Wang, Z., Liu, X., Huang, L., Chen, Y., Zhang, Y., Lin, Z., Wang, R., 2022. Qsfm: Model pruning based on quantified similarity between feature maps for ai on edge. IEEE Internet of Things Journal , 24506–24515.
[74] Wu, B., Wang, D., Zhao, G., Deng, L., Li, G., 2020. Hybrid tensor decomposition in neural network compression. Neural Networks 132, 309–320. URL: https://www.sciencedirect.com/science/article/pii/S0893608020303294, doi:https://doi.org/10.1016/j.neunet.2020.09.006.
[75] Yang, C., Liu, H., 2022. Channel pruning based on convolutional neural network sensitivity. Neurocomputing 507, 97–106. URL: https://www.sciencedirect.com/science/article/pii/S0925231222009110, doi:https://doi.org/10.1016/j.neucom.2022.07.051.
[76] Yang, H., Liang, Y., Liu, W., Meng, F., 2023. Filter pruning via attention consistency on feature maps. Applied Sciences .
[77] Yuan, T., Deng, W., Tang, J., Tang, Y., Chen, B., 2019. Signal-to-noise ratio: A robust distance metric for deep metric learning. CVPR .
[78] Zhang, S., Gao, M., Ni, Q., Han, J., 2023a. Filter pruning with uniqueness mechanism in the frequency domain for efficient neural networks. Neurocomputing 530, 116–124. URL: https://www.sciencedirect.com/science/article/pii/S0925231223001364, doi:https://doi.org/10.1016/j.neucom.2023.02.004.
[79] Zhang, W., Wang, Z., 2022. Fpfs: Filter-level pruning via distance weight measuring filter similarity. Neurocomputing 512, 40–51. URL: https://www.sciencedirect.com/science/article/pii/S092523122201164X, doi:https://doi.org/10.1016/j.neucom.2022.09.049.
[80] Zhang, Y., Lin, M., Lin, C.W., Chen, J., Wu, Y., Tian, Y., Ji, R., 2022a. Carrying out cnn channel pruning in a white box. IEEE Transactions on Neural Networks and Learning Systems , 1–10.
[81] Zhang, Y., Lin, M., Lin, Z., Luo, Y., Li, K., Chao, F., Wu, Y., Ji, R., 2022b. Learning best combination for efficient n: M sparsity. Advances in Neural Information Processing Systems 35, 941–953.
[82] Zhang, Y., Lin, M., Zhong, Y., Chao, F., Ji, R., 2023b. Lottery jackpots exist in pre-trained models. IEEE Transactions on Pattern Analysis and Machine Intelligence .
[83] Zhen, C., Zhang, W., Mo, J., Ji, M., Zhou, H., Zhu, J., 2023. Rasp: Regularization-based amplitude saliency pruning. Neural Networks URL: https://www.sciencedirect.com/science/article/pii/S0893608023004963, doi:https://doi.org/10.1016/j.neunet.2023.09.002.
[84] Zhou, A., Ma, Y., Zhu, J., Liu, J., Zhang, Z., Yuan, K., Sun, W., Li, H., 2021. Learning n: m fine-grained structured sparse neural networks from scratch. ICLR .
[85] Zu, X., Li, Y., Yin, B., 2023. Consecutive layer collaborative filter similarity for differentiable neural network pruning. Neurocomputing 533, 35–45. URL: https://www.sciencedirect.com/science/article/pii/S0925231223002114, doi:https://doi.org/10.1016/j.neucom.2023.02.063.