Singular values-driven automated filter pruning

Van Tien Pham, Yassine Zniyed, Thanh Phuong Nguyen

▶ **To cite this version:**

Van Tien Pham, Yassine Zniyed, Thanh Phuong Nguyen. Singular values-driven automated filter pruning. Neural Networks, inPress, pp.107857. ⟨10.1016/j.neunet.2025.107857⟩. ⟨hal-05157087⟩

HAL Id: hal-05157087

https://hal.science/hal-05157087v1

Submitted on 10 Jul 2025

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

HAL Authorization

Singular values-driven automated filter pruning

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

[a] *Université de Toulon, Aix Marseille University, CNRS, LIS, UMR 7020, France*
[b] *University of Côte d'Azur, CNRS, I3S, UMR 7271, France*

Abstract

In this paper, we present SLIMING (Singular vaLues-drIven autoMated filter prunING), an automated filter pruning method that uses singular values to formalize the pruning process as an optimization problem over filter tensors. Recognizing that this original formulation poses a combinatorial challenge, we propose to replace it with a two-step process that consistently uses singular values in each phase: (i) determining the pruning configuration, which specifies the number of filters to retain in each layer, and (ii) selecting the filters themselves. We show that this approach ensures the preservation of the filters' multidimensional structure throughout the pruning process. For each of these steps, we propose a straightforward algorithm to solve them. To validate each part of our approach, we performed a numerical simulation on an overparameterized synthetic toy example. Additionally, we conducted extensive simulations across eight architectures, four benchmark datasets, and four vision tasks, validating the efficacy of our framework. Our code is available for research purposes at sliming-ai.github.io.

Keywords: matrix and tensor decompositions, HOSVD, structured pruning, automated pruning, network compression

1. Introduction

Network compression is an important area in machine learning that focuses on reducing the computational and memory demands of neural networks while maintaining their performance. The objective is to develop more efficient models that can be deployed on resource-constrained devices, enabling faster inference and greater autonomy. To achieve network compression, various techniques have been introduced, including pruning, quantization, neural architecture search (NAS), and low-rank factorization [9, 37, 56, 69]. These approaches contribute to reductions in model size, number of parameters, inference computation, and energy consumption.

Among these techniques, network pruning stands out as a particularly effective approach for supporting model deployment by reducing neural network redundancy [68]. The underlying principle behind network pruning is that many models are over-parameterized, containing numerous unnecessary or redundant parameters [42]. Pruning these redundant parameters yields smaller and more efficient models suitable for deployment on resource-constrained devices, often improving generalization performance [19, 57]. Network pruning can be categorized into two primary types based on the granularity of the pruning process, unstructured or structured pruning. Unstructured pruning, also known as weight pruning, involves removing individual insignificant weights. While offering advantages like reduced computational complexity and improved memory efficiency, this approach introduces inefficiencies in hardware acceleration, requiring specialized hardware or software [19]. In contrast, structured pruning overcomes these limitations by removing entire structured components such as layers [46], channels [14], and filters [19], making it more adaptable across different hardware platforms [37].

In the process of filter pruning, we are confronted with two questions that directly impact the efficacy of the pruning procedure. Firstly, we must establish a pruning configuration—i.e., devising a strategic approach to ascertain

*Corresponding author
Email addresses: van-tien-pham@etud.univ-tln.fr (Van Tien Pham), zniyed@univ-tln.fr (Yassine Zniyed), tpnguyen@i3s.unice.fr (Thanh Phuong Nguyen)

the number of filters to be pruned across each layer. Secondly, once the pruning ratio for each layer is defined (equivalently, the pruning configuration is established), the challenge shifts to identifying the optimal set of filters for pruning. Numerous methods have been proposed in the literature to address these questions. Among them, intra-channel approaches [1, 42, 66, 30, 8, 68, 49] assess filter importance solely based on individual filter or feature map characteristics. Metrics like the rank [42], norm [1, 66, 30], entropy [68, 49, 75] and energy [8] are commonly used to gauge filter saliency. However, such approaches overlook filter correlations, often resulting in redundant pruned models. For illustration, let us consider a convolutional layer containing three filters: two of them are structurally similar, while the third is noticeably different. Without loss of generality, suppose that the feature maps produced by the similar filters have higher ranks than the one generated by the distinct filter. According to its underlying criterion, HRank [42] would, for instance, remove the filter associated with the lowest-rank feature map—that is, the distinct one. However, this choice leads to preserving two highly redundant filters while discarding the more informative one, potentially reducing the diversity of the remaining filter set. To address this limitation, inter-channel methods [35, 68, 50, 74, 21, 36, 28, 40, 18, 65, 34] exploit correlations or similarities between filters or feature maps to reduce redundancy. Techniques such as similarity-based pruning [35, 68, 50, 36, 4, 40, 34] evaluate filter importance through pairwise distance measurements, while others introduce metrics like channel independence [68, 21, 28, 65] and attention map similarity [74, 45].

While inter-channel methods mitigate some shortcomings of magnitude-based approaches, they still have limitations. First, despite the multidimensional nature of the filter/feature map tensors, several works [42, 65, 40, 34] often flatten 3-order filter tensors into matrices or 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 the loss of crucial information [55]. Second, certain methodologies [35, 68, 21, 66, 55, 36, 54, 65] examine filters or features in isolation within each layer, thus overlooking the global relationship between layers and failing to provide a comprehensive interpretation of network pruning from a holistic perspective. Third, several methods estimate the importance of filters through their corresponding feature maps or channels [50, 19, 74, 38, 21, 36, 4, 58, 7, 42, 65, 8]. With their feature-guided nature, such data-dependent methods are computationally inefficient and sensitive to the distribution of input data, making it difficult to accurately estimate average feature map values [73].

The primary challenge in pruning lies in establishing the pruning configuration, i.e., the pruning ratio for each layer. There are two approaches: manual [50, 73, 30, 68, 26, 27] and automatic [63, 19, 37, 72, 13, 74] pruning configuration. With the former approach, the sparsity of each layer is manually defined based on experience. As manual search is a time-consuming process that requires human expertise, this approach lacks automation and thus has low adaptability and practicality in realistic engineering applications [5]. With the latter approach, the pruning rate is automatically determined to meet a given constraint such as the number of parameters, or the computational complexity [19, 37, 72, 13, 74, 18, 16, 52, 76, 34]. Recent works rely on various similar or saliency metrics, such as attention map ranking [74], sensitivity-based analysis [76, 17, 34], architecture generator [72], and reinforcement learning (RL) [13, 37] to discover tensor redundancy (i.e. filter redundancy). However, these methods also contain several disadvantages. Most of them are computationally and time-consuming because they either require retraining the model from scratch [6, 77], finetuning the model while pruning [72, 17, 34], or calculating the feature map [19, 74, 38, 18, 77]. On the other hand, some NAS-based [6, 16, 52, 33, 43] or RL-based methods [13, 37], which train the model on a proxy dataset in a calibration process to search for the pruning configuration, depend on the proxy dataset and are computationally expensive. Additionally, some methods reshape 4-order filters [34, 73, 40] or 3-order feature maps [18] to 2-order matrices, potentially compromising multidimensional information richness.

This paper introduces a method that leverages tools from linear and multilinear algebra to provide a new solution for automated filter pruning. We propose to detect network redundancy hinging on the dynamics of singular values. In this work, we illuminate the relationship between filter redundancy within neural networks and the observable variations in their singular values. This observation underpins our approach, which we articulate as a constrained optimization problem. Recognizing the inherent complexity and combinatorial nature of filter pruning, we strategically decompose the overarching challenge into two manageable sub-problems. Both are methodically designed to leverage the same underlying principle—the variation of singular values—thereby ensuring consistency in our methodology. Central to our formulation is the consideration of the filters' multidimensional structure, a critical aspect that preserves the integrity of information throughout the pruning process. We show that our approach guarantees that the essential information encoded within the filters is retained, preventing any loss of crucial information. To navigate the complexities of this optimization landscape, we introduce two heuristic methods, each tailored to effectively tackle one of

the sub-problems. The first method focuses on estimating the optimal pruning configuration, determining the precise number of filters to be retained across different layers under a global constraint. The second method is dedicated to the selection of filters, identifying which filters to preserve in order to maximize filter independence. Together, these methods embody a comprehensive solution to automatic filter pruning, called SLIMING, for Singular vaLues-drIven autoMated filter prunING.

Our approach distinguishes itself from existing methods through several key characteristics. First, it handles the multidimensional structure of the weight tensor without the reshaping process commonly seen in prior works [65, 18, 34, 20, 73]. Second, it adopts a layer-global perspective, establishing inter-layer relationships based on a desired resource budget, which contrasts with the layer-local approaches [35, 68, 21, 66, 55, 36, 65] that often ignore these interactions. In our method, all layers participate in the optimization process to jointly determine the optimal pruning configuration. Third, it uses an inter-filter strategy, leveraging information across all filters within a layer to better estimate filter saliencies, addressing the limitations found in intra-filter approaches [42, 66, 30]. Lastly, our approach is data-free and directly operates on filters, unlike feature map-based methods [50, 19, 74, 38, 21, 36, 4, 58, 7, 42, 65, 8] which require supplementary input data for estimating filter redundancy. This data independence eliminates the need for proxy data [19, 36, 58] or a calibration phase [34, 7, 77], thereby enhancing compression efficiency. Briefly, the contributions of this work are three-fold:

- First, we highlight the link between the variation in singular values of filter tensors and network redundancy, allowing us to formulate automated structured pruning as a constrained optimization problem aimed at maximizing the independence of pruned model filters.
- Second, acknowledging the combinatorial complexity of the initial formulation, we simplify it into two sub-problems and introduce two straightforward algorithms to solve them.
- Third, we assess our framework across diverse vision tasks, including image classification, object detection, instance segmentation, and keypoint detection. Through a comprehensive comparison with existing pruning methods, we validate the effectiveness of our approach.

2. Notations and background

2.1. Notations and 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. The $(i_1, i_2, \cdot, i_d)$-th element of $\boldsymbol{\mathcal{A}}$ is denoted as $a_{i_1 i_2 \ldots i_d}$. The Frobenius norm is denoted as $\|.\|_F$. $\mathrm{unfold}_n \boldsymbol{\mathcal{A}}$ refers to the unfolding of tensor $\boldsymbol{\mathcal{A}}$ over its n-th mode [10]. $\mathbf{a}^T$ is the transpose of $\mathbf{a}$. The cardinality of set $\mathcal{A}$ is denoted as $\big|\mathcal{A}\big|$.

Definition 1. *The singular value decomposition (SVD) of a matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ is a factorization of the form*

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T = \sum_{i=1}^{r} \sigma_i \mathbf{u}_i \mathbf{v}_i^T, \tag{1}$$

where $r \leq \min(m, n)$ is the rank of $\mathbf{A}$, $\mathbf{\Sigma} \in \mathbb{R}^{m \times n}$ is a diagonal matrix whose diagonal entries are non-negative and represent the singular values, $\mathbf{U} \in \mathbb{R}^{m \times m}$ is an orthogonal matrix, containing the left singular values of $\mathbf{A}$, and $\mathbf{V} \in \mathbb{R}^{n \times n}$ is an orthogonal matrix, containing the right singular values of $\mathbf{A}$. The singular values adopt the following convention

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r \geq 0 = \sigma_{r+1} = \cdots = \sigma_{\min(m,n)}. \tag{2}$$

Definition 2. *The nuclear norm of a matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$, denoted as $\|\mathbf{A}\|_*$, is defined as the sum of its singular values:*

$$\|\mathbf{A}\|_* = \sum_{i=1}^{r} \sigma_i. \tag{3}$$

Definition 3. *The mode-n product of a tensor $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_d}$ with a matrix $\mathbf{U} \in \mathbb{R}^{J \times I_n}$ is defined as:*

$$\boldsymbol{\mathcal{B}} = \boldsymbol{\mathcal{A}} \times_n \mathbf{U} \Longleftrightarrow b_{i_1 \ldots i_{n-1} j i_{n+1} \ldots i_d} = \sum_{i_n=1}^{I_n} a_{i_1 \ldots i_d} u_{j i_n}, \tag{4}$$

where $\boldsymbol{\mathcal{B}} \in \mathbb{R}^{I_1 \times \cdots \times I_{n-1} \times J \times I_{n+1} \cdots \times I_d}$.

Definition 4. *The Tucker decomposition [70] expresses a d-order tensor $\boldsymbol{\mathcal{A}}$ as a series of mode-n products:*

$$\boldsymbol{\mathcal{A}} = \boldsymbol{\mathcal{G}} \times_1 \mathbf{U}^{(1)} \times_2 \mathbf{U}^{(2)} \times_3 \ldots \times_d \mathbf{U}^{(d)}, \tag{5}$$

where $\boldsymbol{\mathcal{G}} \in \mathbb{R}^{R_1 \times R_2 \times \cdots \times R_d}$ is the core tensor, and $\mathbf{U}^{(n)} \in \mathbb{R}^{I_n \times R_n}$ is the n-th mode factor matrix. The tuple $\{R_1, R_2, \ldots, R_d\}$ is called the multilinear rank of $\boldsymbol{\mathcal{A}}$. In the special case when the factor matrices $\mathbf{U}^{(n)}$ are orthogonal, the decomposition in (5) *is called the higher-order singular value decomposition (HOSVD) [10]. This latter is considered as a generalization of the SVD of matrices defined in* (1)*.*

2.2. Relationship between singular values and redundancy

To motivate the use of singular values and the nuclear norm as criteria for filter selection, we now provide an interpretation of their relationship with redundancy. The singular value decomposition and nuclear norm were formally defined in Subsection 2.1 (see Definitions 1 and 2). Here, we revisit these notions to analyze how the spectrum of singular values reveals structural redundancy in a set of convolutional filters. According to the Eckart–Young theorem, the best rank-r approximation $\mathbf{A}_r$ of a matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$, in terms of the Frobenius norm, is obtained by truncating its singular value decomposition to the r largest singular values. The approximation error is given by the squared sum of the discarded singular values [12]:

$$\|\mathbf{A} - \mathbf{A}_r\|_F^2 = \sum_{i=r+1}^{\min(m,n)} \sigma_i^2. \tag{6}$$

Large singular values correspond to dominant directions capturing most of the variance in the data, while small singular values indicate directions that contribute little to the matrix's structure. When singular values decay rapidly, it means that the matrix's energy is concentrated along only a few directions. In such cases, the row vectors (e.g., convolutional filters) lie close to a low-dimensional subspace, and many of them convey overlapping or redundant information. Conversely, when the singular values are more evenly distributed, the information is spread across multiple independent directions, indicating greater diversity and reduced redundancy. The nuclear norm, defined as the sum of singular values, quantifies this spread. Maximizing the nuclear norm therefore promotes a more balanced contribution of multiple orthogonal components, thereby encouraging filter diversity and minimizing redundancy.

2.3. Classical problem formulation of filter pruning

Let us consider a convolutional neural network (CNN) model comprising L convolutional layers. The l-th layer possesses a weight tensor $\boldsymbol{\mathcal{W}}^l$ of dimensions $C_l \times C_{l-1} \times h_l \times w_l$, where C_l, C_{l-1}, h_l, and w_l represent, respectively, the number of output channels, the number of input channels, the height of the kernel, and the width of the kernel. The primary objective of filter pruning can be expressed as

$$\underset{\left\{\boldsymbol{\mathcal{W}}^l\right\}_{l=1}^{L}}{\arg\min}\ \mathcal{L}\Big(\left\{\boldsymbol{\mathcal{W}}^l\right\}_{l=1}^{L}, \mathcal{D}\Big), \quad \text{s.t.} \quad \mathcal{C}\left(\left\{\boldsymbol{\mathcal{W}}^l\right\}_{l=1}^{L}\right) \leq \mathcal{C}_{\text{desired}}, \tag{7}$$

where $\mathcal{L}$ is the loss function, and $\mathcal{D}$ is the considered dataset. $\mathcal{C}(\cdot)$ is the function that computes the model's configuration, and $\mathcal{C}_{\text{desired}}$ is the desired resource budget, representing flexible constraints such as the number of retained filters, target parameters, or MACs. Without loss of generality, let $\mathcal{C}_{\text{desired}}$ represent the desired total number of filters to retain N, and the function $\mathcal{C}(\cdot)$ now computes the sum of the number of filters to be kept across all layers. For the l-th layer, let $\mathcal{K}^l = \left\{k_1^l, k_2^l, \ldots, k_{N_l}^l\right\}$ be the set of the indices of the retained filters, where k_n^l is the index of the n-th retained filter, $1 \leq n \leq N_l$, and N_l is the total number of the retained filters. $\boldsymbol{\mathcal{W}}^l_{\mathcal{K}^l,:,:,:}$ is the weight tensor of the l-th pruned layer. Thus, the problem formulated in (7) can be expressed as

$$\underset{\left\{\mathcal{K}^l\right\}_{l=1}^{L}}{\arg\min}\ \mathcal{L}\Big(\left\{\boldsymbol{\mathcal{W}}^l_{\mathcal{K}^l,:,:,:}\right\}_{l=1}^{L}, \mathcal{D}\Big), \text{s.t.} \sum_{l=1}^{L} \left|\mathcal{K}^l\right| \leq N. \tag{8}$$

3. SLIMING approach

In this section, we present SLIMING, our proposed method, for addressing the automated filter pruning problem using singular values. First, in subsection 3.1, we analyze the relationship between singular values of filter tensors and their dependencies. Building upon this analysis, we formulate the singular values-driven pruning problem, characterized as a combinatorial optimization problem. Due to the intractable nature of directly solving this combinatorial formulation over a large search space, we propose a relaxation by decomposing the problem into two subproblems, as described in subsection 3.2. We then respectively propose two tailored solutions for these subproblems. The first component, *pruning configuration estimation* (detailed in subsection 3.3), takes as input the weights of a pretrained model and a target total number of filters to retain. It outputs an optimal layer-wise pruning configuration that satisfies the global filter budget. The second component, *filters selection* (discussed in subsection 3.4), iteratively removes redundant filters in each layer until the target number of retained filters—determined in the previous step—is achieved. Finally, the pruned model is fine-tuned to recover any information loss. The overall pipeline of the SLIMING pipeline is concisely summarized in subsection 3.5.

3.1. Singular values of filter tensors and their dependencies

In the case of matrices, the concept of redundancy, i.e., the dependency among columns and/or rows, culminates in rank degeneracy, signifying that the matrix does not possess full rank. This phenomenon of rank degeneracy is palpable through the lens of singular values. Specifically, if the tally of non-zero singular values falls short of $\min(m, n)$ for a matrix of dimensions $m \times n$, it heralds the existence of dependencies between the matrix's columns and/or rows. Transitioning to our context, where the weights $\boldsymbol{\mathcal{W}}^l$ manifests as multidimensional 4-order tensors rather than matrices, the conventional approach [34, 73, 40, 42, 65] of flattening these tensors into matrices of size $C_l \times (C_{l-1} \cdot h_l \cdot w_l)$, which encapsulates the vectorized rendition of the C_l filters across their rows, poses the risk of obfuscating the filters inherent multidimensional characteristics [56].

It appears most prudent, therefore, to explore the variation of a certain "multilinear" singular values counterpart pertaining to the 4-order tensor, particularly across the filters mode, *i.e.,* mode-1 of tensor $\boldsymbol{\mathcal{W}}^l$. This mode quintessentially embodies the filters variation. However, the application of the HOSVD does not yield "multilinear" singular values per se, given the core tensor's non-diagonal structure. Nevertheless, this core tensor boasts two important properties [10]: (i) the all orthogonality property (i.e., all the slice matrices of a given mode of the core tensor are mutually orthogonal), and (ii) the ordering property. The latter ensures that the Frobenius norm of the mode-1 matrix slices of the core tensor $\boldsymbol{\mathcal{G}}$ in (5), articulated as:

$$\left\|\boldsymbol{\mathcal{G}}(i,:,:,:)\right\|_F \geq \left\|\boldsymbol{\mathcal{G}}(j,:,:,:)\right\|_F \quad \text{if} \quad i \leq j. \tag{9}$$

This insight remains true for other modes as well, and allows an analysis of filter dependencies via the enduring Frobenius norms of the mode-1 slices of the core tensor $\boldsymbol{\mathcal{G}}$. This quantity, namely the Frobenius norm of the slice matrices, can be seen as a kind of "multilinear" singular values. Moreover, an interesting property highlighted in [10] reveals that, given the HOSVD of the 4-order tensor $\boldsymbol{\mathcal{W}}^l$ as

$$\boldsymbol{\mathcal{W}}^l = \boldsymbol{\mathcal{G}} \times_1 \mathbf{U}^{(1)} \times_2 \mathbf{U}^{(2)} \times_3 \mathbf{U}^{(3)} \times_4 \mathbf{U}^{(4)}, \tag{10}$$

and concurrently, the SVD of mode-1 unfolding of $\boldsymbol{\mathcal{W}}^l$ as

$$\text{unfold}_1 \boldsymbol{\mathcal{W}}^l = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T = \sum_{i=1}^{r} \sigma_i \mathbf{u}_i \mathbf{v}_i^T, \tag{11}$$

then it follows that

$$\left\|\boldsymbol{\mathcal{G}}(i,:,:,:)\right\|_F = \sigma_i. \tag{12}$$

In essence, this analysis elucidates that pursuing the variation of "multilinear" singular values relative to a specific mode, is equivalent to analyzing the singular values fluctuation of the tensor's unfolding in accordance with that mode. Leveraging these properties, we propose to formulate our pruning problem using the singular values variation, without

compromising their intrinsic dimensional interdependencies. In the sequel and in light of these explanations, we now define the nuclear norm of a tensor $\mathcal{W}^l$ as the nuclear norm of its mode-1 unfolding matrix as

$$\left\|\mathcal{W}^l\right\|_* = \left\|\text{unfold}_1 \mathcal{W}^l\right\|_*. \tag{13}$$

Before expressing the proposed pruning problem formulation, we simulate, in a toy example, a synthetic dataset, dubbed as SVGG, for Synthetic VGG, which includes an original model that mimics the architecture of the VGG network [64] with $L = 5$, $h_l = w_l = 3$ and $\{C_l\}_{l=1}^L = \{64, 128, 256, 512, 512\}$. We choose the redundant rates (the ratio of the number of redundant filters to the total number of filters, i.e., $\frac{C_l - N_l}{C_l}$) of these layers sequentially as $\{0.25, 0.3, 0.35, 0.4, 0.45\}$, thus $\{N_l\}_{l=1}^L = \{48, 90, 166, 307, 282\}$, and the number of retained filters $N = 893$. In the l-th layer, we init N_l core filters with the standard normal distribution while the remaining redundant filters are copied from the core filters with a small noise of variance $\epsilon = 0.01$. It should be noted that the SVGG test is only designed as a proof of concept in a controlled environment, where synthetic redundancy is deliberately introduced. For experiments on real-world models and datasets, including comparisons with state-of-the-art methods, we refer the reader to Section 4. The singular values are visualized in Figure 1. One should note that the overparameterized model contains many near-zero singular values, indicating redundancy. The result of our proposed algorithm on SVGG can be previewed in Figure 2, showing the number of non-redundant selected filters. Our algorithm accurately identifies 100% of the core filters across all layers. In the sequel, we explain in detail our proposed pruning problem formulation and corresponding algorithms to obtain this result.

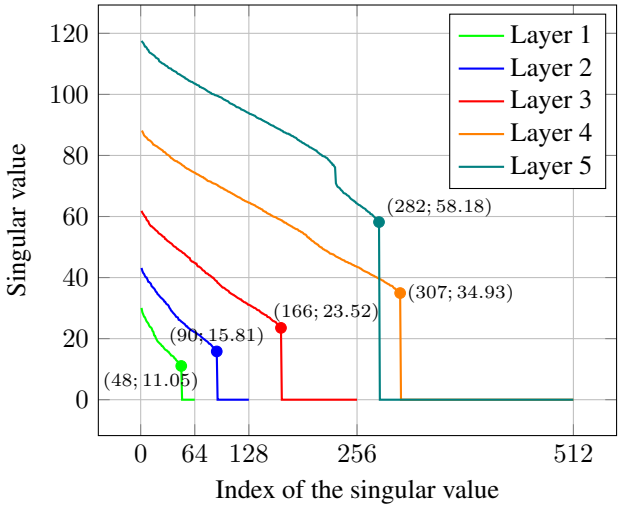


Figure 1: Singular values variation through the layers of SVGG.

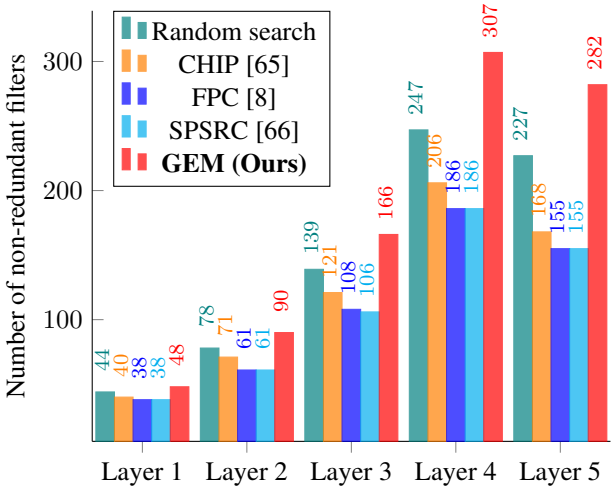


Figure 2: Non-redundant selected filters with SVGG.

3.2. Proposed problem formulation

Drawing on the insights from the preceding subsection, it now appears intuitive that if we establish a general pruning ratio or a fixed number N of filters to retain, then the following two assertions become equivalent: "Retaining the least redundant filters in a layer of a CNN" is synonymous with "Preserving the set of filters that maximizes the sum of singular values". This equivalence arises because filters exhibiting similarities drive their corresponding singular values toward zero, as illustrated in the toy example provided earlier. Consequently, we can express the singular values-driven pruning problem as follows:

$$\underset{\{\mathcal{K}^l\}_{l=1}^L}{\arg\max} \sum_{l=1}^{L} \left\|\mathcal{W}^l_{\mathcal{K}^l,:,:,:}\right\|_*, \quad \text{s.t.} \quad \sum_{l=1}^{L} \left|\mathcal{K}^l\right| \leq N. \tag{14}$$

This optimization challenge seeks to identify a set of sets, $\mathcal{K}^l$ for $1 \leq l \leq L$, that maximizes the nuclear norm (thereby ensuring maximal independence among filters) while ensuring that the aggregate cardinalities of these sets

do not exceed a predetermined value N. Notably, the search space for this combinatorial problem is immense; the number of possible combinations in (14) is bounded by $\sum_{n=1}^{N} \binom{\sum_{l=1}^{L} C_l}{n}$. For instance, in the synthetic SVGG model with $\sum_{l=1}^{L} C_l = 1472$, selecting filters that match the ground-truth $N = 893$ yields $\binom{1472}{893} \approx 6 \times 10^{426}$ possible configurations. This combinatorial complexity is even much greater in real models like VGG [64] or ResNet [24].

Given its combinatorial nature and the vast search space, we relax this problem by decomposing it into two subproblems: (i) determining the optimal cardinalities for each layer that maximize the sum of nuclear norms across all layers, and (ii) once the cardinalities for each layer are established, solving L independent optimization problems where, for each layer, we seek the filter selection that maximizes the nuclear norm subject to the given cardinality constraint.

For the first challenge, namely, finding the optimal configuration, the optimization problem can succinctly be represented as:

$$\underset{\left\{N_l\right\}_{l=1}^{L}}{\arg\max} \sum_{l=1}^{L} \sum_{i=1}^{N_l} \sigma_i^l, \quad \text{s.t.} \quad \begin{cases} \sum_{l=1}^{L} N_l \leq N, \\ 1 \leq N_l \leq C_l \ \text{ for } \ 1 \leq l \leq L, \end{cases} \tag{15}$$

where σ_i^l are the i-th singular values of $\text{unfold}_1 \boldsymbol{\mathcal{W}}^l$, and N_l is the cardinal of the set $\mathcal{K}^l$, essentially the number of filters to retain in each layer.

Upon determining the optimal configuration, the problem of filter selection then diverges into L distinct optimization challenges, one for each layer, defined as follows:

$$\underset{\mathcal{K}^l}{\arg\max} \left\| \boldsymbol{\mathcal{W}}^l_{\mathcal{K}^l,:,:,:} \right\|_*, \quad \text{s.t.} \quad \left|\mathcal{K}^l\right| = N_l, \tag{16}$$

for $1 \leq l \leq L$, with N_l being known. This phase shifts focus to identifying, for each layer, the precise subset of filters $\mathcal{K}^l$ that, when retained, maximizes the nuclear norm, hence adhering to the pre-established cardinality N_l.

To succinctly summarize the two sub-problems delineated in (15) and (16), which collectively supplant the initial problem posited in (14), it can be articulated that the problem in (15) addresses the query, *"How many filters will each layer contribute?"*, whereas the problem in (16) resolves the inquiry, *"Which filters will each layer contribute?"*. The count of possibilities for (15) is is bounded by $\sum_{i=L}^{N} \binom{i-1}{L-1}$. For (16), the search space for the l-th layer and across all layers are $\binom{C_l}{N_l}$ and $\prod_{l=1}^{L} \binom{C_l}{N_l}$, respectively.

It is worth noting that the two subproblems defined in (15) and (16) are not equivalent to the original problem in (14). While breaking down (14) into these subproblems does not constitute an exact replacement, this relaxation offers significant benefits, particularly in terms of computational efficiency and practical interpretability. First, the original problem in (14) has an extraordinarily large search space that grows combinatorially with the total number of filters across all layers. Decomposing it into two stages substantially reduces the search space, making the problem more manageable and feasible to solve, even for large-scale models with many layers. Second, this decomposition clearly separates decision-making stages, allowing us to address high-level resource allocation across layers independently from the detailed selection of filters within each layer. This modularity enables a more structured approach to problem-solving. Third, by tackling the problem in two stages, we gain flexibility in adjusting and interpreting the pruning strategy. The solution to (15) provides an interpretable layer-wise allocation of filters that can be easily adjusted to meet different model-size or inference-time requirements without needing to solve the entire problem from scratch. For instance, if a tighter constraint is introduced, we can simply reduce the filter count in specific layers by adjusting the outcome from (15) and then rerun the filter selection for those affected layers only. This flexibility allows for quick adaptation to new constraints. Furthermore, with the layer-wise filter allocation known, we can employ specialized filter selection techniques for each layer in (16). Fourth, the second subproblem in (16) can be solved independently for each layer once the allocation in (15) is determined. This independence allows the filter selection problem to be parallelized across layers, which can dramatically reduce the overall computation time in practice. This aspect is particularly beneficial in large networks with numerous layers, where parallel computing resources can be leveraged to expedite the pruning process. We showcase the pruning cost efficiency of SLIMING in subsection 5.1. Finally, each subproblem is simpler with a more constrained search space, making it easier to apply heuristic or approximation algorithms that converge quickly to an effective solution—something that may not be feasible in the original global problem. In the next subsections 3.3 and 3.4, we illustrate this advantage by showing that using two proposed heuristic algorithms for (15) and (16) can yield the correct solution for (14) on the toy example. Achieving this result would

be far more complex if we attempted to solve (14) directly. In summary, the relaxation of (14) into the subproblems in (15) and (16) not only reduces the computational burden but also improves flexibility and scalability.

3.3. Pruning configuration estimation

We introduce Algorithm 1, a method aimed at determining the optimal filter configuration across all layers while adhering to a constraint on the total number of retained filters, as described by (15). The core concept revolves around iteratively selecting the largest singular values from all layers until the desired number of retained filters is met. Initially, the algorithm initializes the retained filter count for each layer to 1. The set $\mathcal{S}$ is used to store the remaining largest singular value from each layer, initialized with the second singular values σ_2^l in step 5. This set serves as the pool of candidate singular values to be considered in each iteration. The algorithm iteratively selects the largest candidate singular value from $\mathcal{S}$ and increments the retained filter count of the corresponding layer. This simple process repeats until the desired total number of retained channels is achieved. Applied to SVGG, Algorithm 1 demonstrates an accurate estimation of the pruning ratios. The iterative process of selecting singular values exploits the decreasing property of singular values, resulting in an efficient computational complexity of $\mathcal{O}(N - L)$. This efficiency is achieved by iteratively selecting the largest singular values from each layer's set of candidate singular values until the desired number of retained filters is reached. Consequently, the computational burden of this iterative process is directly tied to the desired number of retained filters and remains independent of the size of the weight tensors.

Algorithm 1 Pruning configuration estimation

Input: $\left\{\boldsymbol{\mathcal{W}}^l\right\}_{l=1}^L$, N.

Output: Optimal configuration $\left\{N_l\right\}_{l=1}^L$ for (15).

1: **parfor** $l = 1$ to L **do** ▷ *In parallel for all* $l = 1$ *to* L
2: $N_l = 1$
3: $\text{unfold}_1 \boldsymbol{\mathcal{W}}^l \overset{\text{SVD}}{=} \mathbf{U}_l \boldsymbol{\Sigma}_l \mathbf{V}_l^T$, with $\sigma_i^l = \boldsymbol{\Sigma}_l(i,i)$.
4: **end parfor**
5: $\mathcal{S} = \left\{\sigma_2^l\right\}_{l=1}^L$
6: **for** $n = 1$ to $N - L$ **do**
7: $j = \arg\max \mathcal{S}_j$ ▷ j is the index of the largest value in $\mathcal{S}$.
8: $N_j = N_j + 1$
9: $\mathcal{S}_j = \sigma_{N_j+1}^j$
10: **end for**
11: Return $\left\{N_l\right\}_{l=1}^L$

3.4. Filters selection

Within this subsection, we introduce Algorithm 2, designed to address the solution to the optimization problem formulated in (16). It is imperative to note that this stage assumes a predefined pruning configuration, with the optimal number of filters to be retained across each layer already determined. The objective of Algorithm 2 is to identify, for each layer, the set $\mathcal{K}^l$ of size N_l, which epitomizes maximal filters independency. This aims to preserve the core functionality of the network by retaining a subset of filters that collectively contribute to its predictive capacity, devoid of redundancy. Given the combinatorial essence of the problem, an exhaustive search approach for deep CNNs is impractical due to the colossal search space it necessitates. To circumvent this challenge, Algorithm 2 employs a heuristic strategy that iteratively eliminates filters until only the N_l most significant filters, as measured by our objective function, remain. The underlying idea of our filter elimination methodology hinges on the observation that the removal of a filter, which results in negligible or no reduction in the overall nuclear norm, implies substantial redundancy of this filter with respect to the retained set. Hence, we start with the complete set of C_l filters for a given layer. Through successive iterations, the algorithm methodically prunes the least contributory filters, converging on the desired subset of N_l filters. This iterative culling process ensures that each step towards achieving the target configuration is informed by the strategic goal of minimizing redundancy while maximizing the collective independence and contribution of the

retained filters. The computational complexity of Algorithm 2 is $\mathcal{O}\left(\sum_{i=1}^{C_l-N_l}\left(C_l-i+1\right)^2 C_{l-1}h_l w_l\right)$, bounded by $\mathcal{O}\left(C_l^3 C_{l-1}h_l w_l\right)$. Note that Algorithm 2 is applied in parallel on all layers of the original model to solve (16).

Algorithm 2 Greedy Elimination Method (GEM)

Input: $\boldsymbol{\mathcal{W}}^l$, number of retained filters N_l.
Output: Optimal set $\mathcal{K}^l = \left\{k_1^l, k_2^l, \ldots, k_{N_l}^l\right\}$ for (16).

Initialize $\mathcal{K}^l = \{1, 2, \ldots, C_l\}$
for $i = 1$ to $(C_l - N_l)$ **do**
  Initialize $\Delta_{\text{min}} = \infty$
  for δ_{current} in $\mathcal{K}^l$ **do**
    $\mathcal{R}^l = \mathcal{K}^l \setminus \{\delta_{\text{current}}\}$
    $\Delta_{\text{current}} = \left\|\boldsymbol{\mathcal{W}}^l_{\mathcal{K}^l,:,:,:}\right\|_* - \left\|\boldsymbol{\mathcal{W}}^l_{\mathcal{R}^l,:,:,:}\right\|_*$
    if $\Delta_{\text{current}} < \Delta_{\text{min}}$ **then**
      $\Delta_{\text{min}} = \Delta_{\text{current}}$
      $\delta_{\text{min}} = \delta_{\text{current}}$
    end if
  end for
  $\mathcal{K}_l = \mathcal{K}^l \setminus \{\delta_{\text{min}}\}$
end for
Return $\mathcal{K}^l$

Here again, we proceed to validate the efficacy of the proposed Algorithm 2 by initially applying it to our synthetic toy example. This preliminary step precedes its application to deep CNNs from the state-of-the-art, which will be elaborated upon in the subsequent section. In our comparative analysis, we juxtapose GEM against 5 related techniques: the exhaustive search method, which, despite its guarantee of yielding optimal results by evaluating every possible combination, proves impractical for larger models due to its computational demands; a randomized search approach, which, by sampling a limited subset of the entire search space, risks overlooking the optimal solution; CHIP [65]; FPC [8]; and SPSRC [66]. In CHIP, FPC, and SPSRC, the authors did not formulate filter pruning as an optimization of the nuclear norm and it employed the calculation on feature maps, while the filters are addressed in our approach. However, CHIP proposed an approach to maximize channel independence, in which the impact of channels is calculated once, and then the most N_l independent channels are retained. Meanwhile, both FPC and SPSRC adopt an intra-channel strategy, assessing the importance of channels through metrics such as the nuclear norm and the largest singular value, respectively. In Table 1 and Figure 2, we compare them via three aspects, namely the computational complexity, the nuclear norm of the pruned model, and the number of non-redundant filters in l-th layer of the pruned model. τ is the number of selected candidates in the range search. Among all the above competing methods, only GEM and complete search manage to identify all unique filters, achieving 100% accuracy and the highest nuclear norm (41917), which corresponds to maximal retained information. The complexity of GEM is given by $\mathcal{O}\left(\sum_{i=1}^{C_l-N_l}\left(C_l-i+1\right)^2 C_{l-1}h_l w_l\right)$, upper-bounded by $\mathcal{O}\left(C_l^3 C_{l-1}h_l w_l\right)$. In contrast, complete search incurs a much higher cost of $\mathcal{O}\left(\binom{C_l}{N_l} C_l^2 C_{l-1}h_l w_l\right)$—approximately $\frac{\binom{C_l}{N_l}}{C_l}$ times more expensive than GEM. Instead of this exponential gap, GEM attains the same performance with cheaper computational cost, making it a computationally efficient yet highly effective alternative. Other heuristic-based methods significantly exhibit lower performance. Random search, with reduced complexity $\mathcal{O}(\tau C_l^2 C_{l-1}h_l w_l)$, only achieves 82.31% accuracy and fails to retain many informative filters (735 vs. 893). CHIP [65], while slightly more efficient, drops to 67.86% accuracy with 606 non-redundant filters retained. FPC [8] and SPSRC [66], both relying on significantly simpler complexity $\mathcal{O}(C_{l-1}^2 h_l w_l)$, perform the worst with accuracies around 61% and the fewest non-redundant filters. These results highlight the effectiveness of GEM in balancing computational efficiency and selection quality. Nevertheless, we note that GEM is a greedy algorithm and does not guarantee optimal solutions in all cases, which remains an open perspective for future research.

Table 1: Filters selection methods on SVGG.

Method	Complexity	Nuclear norm	Non-redundant filters	Accuracy
Complete search	$\mathcal{O}\left(\binom{C_l}{N_l} C_l^2 C_{l-1} h_l w_l\right)$	41917	893	100.00
Random search	$\mathcal{O}\left(\tau C_l^2 C_{l-1} h_l w_l\right)$	37456	735	82.31
CHIP [65]	$\mathcal{O}\left(C_l^2 C_{l-1} h_l w_l\right)$	33789	606	67.86
FPC [8]	$\mathcal{O}\left(C_{l-1}^2 h_l w_l\right)$	32403	548	61.37
SPSRC [66]	$\mathcal{O}\left(C_{l-1}^2 h_l w_l\right)$	32197	546	61.14
GEM (Ours)	$\mathcal{O}\left(\sum_{i=1}^{C_l - N_l} \left(C_l - i + 1\right)^2 C_{l-1} h_l w_l\right)$	41917	893	100.00

3.5. Overall pipeline

Algorithm 3 outlines the complete SLIMING pipeline. The algorithm takes as input a pretrained model consisting of convolutional layers with weights $\left\{\boldsymbol{\mathcal{W}}^l\right\}_{l=1}^L$, the total number of filters to retain N, and a set of fine-tuning hyperparameters. The process begins by determining the per-layer pruning configuration $\{N_l\}_{l=1}^L$ using Algorithm 1, which solves the global filter allocation problem in equation (15). Next, in parallel across all convolutional layers, SLIMING selects the N_l most important filters in each layer based on the greedy elimination method provided by Algorithm 2, which solves equation (16). The filters not selected are then pruned, resulting in a reduced network. Finally, the pruned model is fine-tuned on the training dataset $\mathcal{D}$ using the specified fine-tuning configuration (number of epochs E, learning rate η, learning rate scheduler Sched, batch size B, optimizer Opt, momentum m, weight decay λ) to recover or even improve the model's performance.

Algorithm 3 SLIMING algorithm

Input: Pretrained model with the weights $\left\{\boldsymbol{\mathcal{W}}^l\right\}_{l=1}^L$, total number of retained filters N, fine-tuning hyperparameters: number of epochs E, learning rate η, learning rate scheduler Sched, batch size B, optimizer Opt, momentum m, weight decay λ, training dataset $\mathcal{D}$

Output: Compact model that satisfies equation (8).

1: Estimate the pruning configuration $\left\{N_l\right\}_{l=1}^L$ for (15) using Algorithm 1.
2: **parfor** $l = 1$ to L **do** ▷ *In parallel for all* $l = 1$ *to* L
3: Find indices of the filters to be kept $\mathcal{K}^l = \left\{k_1^l, k_2^l, \ldots, k_{N_l}^l\right\}$ for (16) using Algorithm 2.
4: Keep $\boldsymbol{\mathcal{W}}^l_{\mathcal{K}^l,:,:,:}$, remove redundant filters.
5: **end parfor**
6: Fine-tune the pruned model on $\mathcal{D}$ using $\{E, \eta, \mathrm{Sched}, B, \mathrm{Opt}, m, \lambda\}$

4. Experiments

4.1. Experimental settings

Architectures and Datasets: To demonstrate the adaptability of SLIMING, we assess five architectures: VGG-16-BN [64], GoogLeNet with inception modules [67], ResNet-20/32/56/110 with residual blocks [24], DenseNet-40 with dense blocks [29], and MobileNetV2 with inverted residual block [61]. These models are tested on the CIFAR-10/100 dataset [32]. Additionally, to validate the scalability of SLIMING, experiments are conducted on the ImageNet dataset [11] using ResNet-34/50 [24] and MobileNetV2 [61] architectures. Furthermore, the compressed ResNet-50 model is employed as the backbone network for FasterRCNN-FPN [60], MaskRCNN, and KeypointRCNN [23] on the COCO-2017 dataset [44]. It is worth mentioning that the pretrained models on CIFAR-10/100 and Imagenet are taken from CHIP [65], and Faster/Mask/Keypoint-RCNN from Pytorch [53]. All these models are available on our code repository: sliming-ai.github.io.

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 quantified as the percentage

reduction in MACs/Params 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.

Fine-tuning Configuration: The hyperparameters used during fine-tuning are summarized in Table 2, with their notations defined in Algorithm 3. Stochastic gradient descent (SGD) is employed as the optimizer across all three datasets. Before training, the datasets are preprocessed as follows. For CIFAR-10/100, the training images are augmented using random cropping to 32×32 pixels with 4-pixel padding, followed by random horizontal flipping. The images are then converted to tensors and normalized using the channel-wise mean $(0.4914, 0.4822, 0.4465)$ and standard deviation $(0.2023, 0.1994, 0.2010)$, corresponding to standard CIFAR-10 statistics. Test images are only converted to tensors and normalized, without augmentation. For ImageNet, training images undergo random resized cropping, random horizontal flipping, tensor conversion, and normalization using the standard mean $[0.485, 0.456, 0.406]$ and standard deviation $[0.229, 0.224, 0.225]$. Validation images are resized to 256 pixels on the shorter side, center-cropped to 224×224 pixels, and then normalized. For COCO, training data is augmented using horizontal flipping and copy-paste augmentation from Torchvision [53]. We conducted fine-tuning using NVIDIA Tesla V100 SXM2 16GB GPUs and Intel Cascade Lake 6248 processors (20 cores at 2.5 GHz).

Table 2: Fine-tuning hyperparameters

Dataset	Hyperparameters						
	E	η	Sched	B	Opt	m	λ
CIFAR-10/100	300	0.1	CosineAnnealingLR [51]	128	SGD	0.9	0.005
ImageNet	150	0.5	CosineAnnealingLR [51]	128	SGD	0.9	0.0001
COCO	Faster/Mask-RCNN: 26 Keypoint-RCNN: 46	0.02	MultiStepLR [53]: (16, 22) MultiStepLR [53]: (36, 43)	16	SGD	0.9	0.0001

Baselines: SLIMING is compared with 58 related works, including 31 hand-crafted [39, 35, 50, 73, 30, 68, 56, 55, 26, 27, 78, 21, 66, 20, 36, 2, 57, 48, 41, 22, 9, 49, 7, 4, 25, 8, 3, 42, 65, 15, 47] and 27 automatic pruning rate approaches [38, 63, 19, 37, 72, 46, 13, 1, 14, 74, 71, 45, 6, 28, 52, 58, 5, 18, 76, 33, 17, 75, 34, 77, 59, 40, 16]. We classify the selected baselines into groups and provide a brief introduction to them as follows:

- Intra-channel approaches [1, 42, 66, 30, 8, 49] assess filter importance solely based on individual filter or feature map characteristics. Rank: HRank [42]. Norm: SPWB [1], SPSRC [66], ResPrune [30]. Entropy: FSIM-E [49], REAF [75]. Energy: FPC [8]. For example, SPWB [1] uses the L2 norm to define the filters' importance; the filters with L2 norms that are below a pruning threshold are removed during training.
- Inter-channel methods [35, 68, 50, 74, 21, 36, 28, 40, 18, 65, 34, 73, 49] exploit correlations or similarities between filters or feature maps to reduce redundancy. Channel independence: CHIP [65], Tang et. *al* [68], TAILOR [21], CATRO [28], FSIM-E [49], CLR-RNF [40], APIB [18], FiltDivNet [34], SFI-FP [73]. Feature map or filter similarity: ARPruning [74], ACSC [45], SFP [35], FPWT [50].
- NAS-based approaches [6, 16, 52, 33, 43]. EvoFC [6], DAIS [16], Lu *et al.* [52], Lee et. *al* [33], ABCPruner [43], RLAL [13], MCMC [37].
- RL-based methods: RLAL [13], MCMC [37]. For example, RLAL [13] jointly learns the weights and structurally prunes the architectures of CNN models during training.
- Other approaches. Sensitivity-based analysis: ASTER [76], SP [17], FiltDivNet [34]. Architecture generator ATO [72]. Structural reparameterization: UPDP [46]. Interpretable pruning: Guo *et al.* [19], WhiteBox [77]. Tensor-based: CORING [55], NORTON [56]. Attention-based: ACSC [45], GlobalPru [71], ARPruning [74]. Graph-based: HSC [39], RGP [9], Li et. *al* [36]. Clustering-based: SFP [35], ICP [4]. Dynamic pruning: UDSP [14].

We adjust the target compression ratios to align with other methods for a comparable compression rate, facilitating an analysis of the trade-off between compression rate and accuracy, and vice versa.

4.2. Results on CIFAR-10

VGG-16-BN. Table 3 shows the pruning results of VGG-16-BN on CIFAR-10. Compared to hand-crafted pruning rate approaches such as CHIP [65], RGP [9], and SPSRC [66], the pruned models from SLIMING give higher accuracy while consuming fewer resources. In comparison with RL-based automatic pruning rate approaches including MCMC [37], SLIMING compresses more MACs and parameters and gives higher accuracy. This advantage also extends to advanced attention-based automatic pruning methods, such as ACSC [45], GlobalPru [71], and ARPruning [74]. For instance, ARPruning [74], a channel pruning method based on attention map ranking, achieves a comparable parameter reduction (89% vs. 92%) but still lags behind SLIMING in accuracy (92.69% vs. 93.74%).

Table 3: Pruning results of VGG-16-BN on CIFAR-10

Method	Auto	Top-1	MACs (CR)	Params (CR)
VGG-16-BN [64]		93.96	313.73M (00)	14.98M (00)
HRank [42]	✗	92.34	108.61M (65)	2.64M (82)
GlobalPru [71]	✓	93.49	135.85M (57)	2.99M (80)
Guo *et al.* [19]	✓	93.68	109.68M (65)	3.62M (75)
FiltDivNet [34]	✓	93.76	104.66M (66)	4.50M (70)
MCMC [37]	✓	93.77	154.97M (51)	N/A
CHIP [65]	✗	93.86	131.17M (58)	2.76M (82)
SPSRC[66]	✗	93.90	186.83M (40)	2.77M (82)
FPWT [50]	✗	93.94	140.28M (55)	3.96M (73)
SFI-FP [73]	✗	94.02	104.78M (67)	2.50M (83)
SLIMING (Ours)	✓	**94.22**	**99.05M (69)**	**2.49M (83)**
GFBS [48]	✗	92.09	69.02M (78)	N/A
HBFP [2]	✗	92.30	62.30M (80)	2.90M (80)
ACSC [45]	✓	92.69	71.53M (77)	1.60M (89)
ARPruning [74]	✓	93.18	85.02M (73)	2.27M (85)
RGP [9]	✗	92.76	78.78M (75)	3.81M (75)
TAILOR [21]	✗	93.38	73.89M (76)	1.20M (92)
Tang et. *al* [68]	✗	93.67	65.94M (79)	1.77M (88)
SLIMING (Ours)	✓	**93.74**	**59.23M (81)**	**1.16M (92)**
EvoFC [6]	✓	88.71	42.35M (87)	0.99M (93)
RGP [9]	✗	91.45	31.37M (90)	1.43M (90)
SLIMING (Ours)	✓	**91.69**	**31.37M (90)**	**0.23M (99)**

ResNet-56/110. Table 4 presents the pruning outcomes for ResNet-56/110 on CIFAR-10. With ResNet-56, our method achieves over 57% compression in both MACs and parameters compared to the baseline model, while enhancing the accuracy. This contrasts with several other methods [55, 66] that struggle to recover baseline accuracy at such high compression levels. Compared to RLAL [13], a RL-based method which jointly learns the weights and structurally prunes the architectures during training, SLIMING gives higher accuracy (94.15% vs. 93.86%) while still enjoying a better MACs reduction (58% vs. 50%). For ResNet-110, SLIMING outperforms the method proposed by Guo *et al.* [19] (interpretable task-inspired adaptive filter pruning), NNCS [15] (compressive sensing), and SFI-FP [73] (soft independence guided filter pruning) across all metrics. Notably, compared to HSC [39], a very recent state-of-the-art pruning method based on high-order spectral clustering with manually tuned pruning ratios, SLIMING consistently outperforms it across two compression scenarios, offering better parameter savings (84% vs. 70%) and greater MACs reduction (79% vs. 72%) while maintaining comparable accuracy (93.64% vs. 93.56%).

Table 4: Pruning results of ResNet-56/110 on CIFAR-10

Method	Auto	Top-1	MACs (CR)	Params (CR)
ResNet-56 [24]		93.26	127.09M (00)	0.85M (00)
SPSRC[66]	✗	93.19	73.66M (42)	0.47M (45)
DCFF [41]	✗	93.26	55.84M (56)	0.38M (55)
TAILOR [21]	✗	93.55	55.13M (56)	0.42M (51)
MFP [25]	✗	93.56	59.40M (53)	N/A
ResPrune [30]	✗	93.59	53.40M (58)	N/A
DCP [47]	✗	93.72	56.47M (55)	0.43M (50)
GFBS [48]	✗	93.75	73.71M (42)	N/A
RLAL [13]	✓	93.86	63.20M (50)	N/A
Lee et. *al* [33]	✓	94.00	63.41M (50)	0.61M (29)
NORTON [57]	✗	94.00	73.22M (42)	0.44M (48)
SLIMING (Ours)	✓	**94.15**	**53.35M (58)**	**0.37M (57)**
SPSRC[66]	✗	91.65	46.62M (63)	0.30M (65)
CHIP [65]	✗	92.05	34.79M (72)	0.24M (72)
Hu et. *al* [27]	✗	92.21	35.00M (72)	0.26M (69)
Tang et. *al* [68]	✗	92.49	34.09M (73)	0.22M (74)
SFI-FP [73]	✗	92.74	34.79M (72)	0.24M (72)
CORING [55]	✗	92.84	34.79M (72)	0.24M (72)
Torque [20]	✗	93.26	46.72M (63)	0.28M (67)
SLIMING (Ours)	✓	**93.33**	**32.19M (75)**	**0.21M (75)**
ResNet-110 [24]		93.50	256.04M (00)	1.73M (00)
REAF [75]	✓	93.37	105.80M (58)	0.70M (59)
ResPrune [30]	✗	93.68	93.63M (63)	N/A
HSC [39]	✗	94.01	88.26M (65)	0.69M (60)
TAILOR [21]	✗	94.26	113.60M (55)	0.93M (46)
NNCS [15]	✗	93.41	99.86M (61)	0.66M (62)
Chen *et al.* [7]	✗	94.42	88.33M (66)	N/A
SFI-FP [73]	✗	94.43	121.09M (52)	0.89M (48)
SLIMING (Ours)	✓	**94.52**	**87.59M (66)**	**0.61M (65)**
HBFP [2]	✗	91.96	63.30M (75)	0.43M (75)
Guo *et al.* [19]	✓	92.81	76.40M (70)	0.34M (80)
SFI-FP [73]	✗	93.41	67.78MM (73)	0.48M (72)
FPWT [50]	✗	93.45	68.37M (73)	0.52M (70)
HSC [39]	✗	93.56	71.31M (72)	0.51M (70)
SLIMING (Ours)	✓	**93.64**	**54.50M (79)**	**0.28M (84)**

Table 5: Pruning results of DenseNet-40 on CIFAR-10

Method	Auto	Top-1	MACs (CR)	Params (CR)
DenseNet-40 [29]		94.81	292.81M (00)	1.06M (00)
HRank [42]	✗	94.24	167.41M (41)	0.66M (37)
ARPruning [74]	✓	94.42	168.42M (41)	0.56M (46)
FPWT [50]	✗	94.44	165.36M (44)	0.58M (44)
NORTON [56]	✗	94.67	168.23M (42)	0.58M (45)
SPSRC[66]	✗	94.69	168.82M (42)	0.59M (44)
CORING [55]	✗	94.71	173.39M (40)	0.62M (41)
SLIMING (Ours)	✓	**94.80**	**157.62M (46)**	**0.52M (51)**
EvoFC [6]	✓	93.23	134.99M (54)	0.50M (53)
HRank [42]	✗	93.68	110.15M (61)	0.48M (54)
FPWT [50]	✗	93.81	127.00M (57)	0.40M (44)
FPC [8]	✗	94.14	121.08M (59)	0.50M (53)
NORTON [56]	✗	94.17	103.68M (64)	0.33M (69)
ARPruning [74]	✓	94.20	108.81M(61)	0.27M (74)
SLIMING (Ours)	✓	**94.25**	**93.07M (68)**	**0.27M (74)**

Table 6: Pruning results of GoogLeNet on CIFAR-10

Method	Auto	Top-1	MACs (CR)	Params (CR)
GoogLeNet [67]		95.05	1.52B (00)	6.15M (00)
EvoFC [6]	✓	94.28	0.71M (53)	3.44M (44)
HRank [42]	✗	94.53	0.69B (55)	2.74M (55)
FPWT [50]	✗	95.17	0.61M (60)	2.73M (55)
FSIM-E [49]	✗	95.32	0.86B (43)	3.26M (47)
CORING [55]	✗	95.32	0.65B (57)	2.85M (54)
SLIMING (Ours)	✓	**95.48**	**0.62B (60)**	**2.68M (57)**
FPWT [50]	✗	94.49	0.37M (76)	2.00M (67)
CLR-RNF [40]	✓	94.85	0.49B (68)	2.18M (65)
FSIM-E [49]	✗	94.92	0.43B (72)	2.19M (64)
DCFF [41]	✗	94.92	0.46B (70)	2.08M (66)
CORING [55]	✗	95.03	0.39B (74)	2.10M (66)
ICP [4]	✗	95.09	0.44B (71)	2.03M (67)
Zheng et *al.* [78]	✗	95.30	0.52B (66)	N/A
SLIMING (Ours)	✓	**95.31**	**0.29B (81)**	**1.41M (77)**

DenseNet-40. Handling DenseNet architecture can be intricate due to the interconnected nature of channels across layers. The pruning results for DenseNet-40 on CIFAR-10 are illustrated in Table 5. Despite achieving comparable accuracy, our method typically achieves greater compression than other methods [50, 6, 66, 55, 74]. For instance, it reduces more than 21% of the parameters compared to EvoFC [6] while still maintaining higher accuracy. Remarkably, compared to NORTON [56], a hybrid compression approach that combines tensor decomposition with pruning, SLIMING slightly achieves higher accuracy in both moderate and high compression settings (94.80% vs. 94.67% and 94.25% vs. 94.17%). It also provides improved reductions in MACs (46% vs. 42% and 68% vs. 64%) and parameter count (51% vs. 45% and 74% vs. 69%), suggesting its ability to maintain performance under tighter resource constraints.

GoogLeNet. We validate the versatility of our method through an experiment conducted on a multi-branch architecture, GoogLeNet, on CIFAR-10, as detailed in Table 6. SLIMING exhibits superior accuracy with reduced overhead when compared to similarity-based and hand-crafted methods [55, 41, 49, 4]. Notably, SLIMING surpasses similarity-based approaches like FPWT [50], CORING [55], FSIM-E [49], and CLR-RNF [40] across all evaluated metrics. For instance, FPWT [50], a manually configured pruning method leveraging frequency-domain information through wavelet transforms, achieves competitive accuracy (94.49%); however, SLIMING provides higher accuracy (95.31%) along with greater reductions in MACs (81% vs. 76%) and parameters (77% vs. 67%), indicating a more efficient compression strategy.

MobileNetV2. Table 7 presents the pruning results for MobileNetV2 on CIFAR-10, demonstrating that our method outperforms other competing approaches [35, 50, 55, 78, 13, 48, 47]. For instance, compared to the clustering-

based approach SFP [35], SLIMING achieves an additional 23% reduction in parameters and a 14% greater reduction in MACs, all while maintaining higher accuracy. Compared to RLAL [13], SLIMING achieves a slightly higher accuracy (95.01% vs. 94.85%) while also providing a greater reduction in MACs (54% vs. 49%). This suggests that our method can be applied to optimize compact architectures.

Table 7: Pruning results of MobileNetV2 on CIFAR-10

Method	Auto	Top-1	MACs (CR)	Params (CR)
MobileNetV2 [61]		94.51	94.54M (00)	2.24M (00)
GFBS [48]	✗	94.28	54.83M (42)	N/A
DCP [47]	✗	94.72	61.98M (34)	1.34M (40)
CORING [55]	✗	94.81	55.16M (42)	1.26M (44)
FPWT [50]	✗	95.20	54.69M (42)	1.45M (35)
SLIMING (Ours)	✓	**95.28**	**53.81M (43)**	**1.21M (46)**
RLAL [13]	✓	94.85	48.22M (49)	N/A
SLIMING (Ours)	✓	**95.01**	**43.07M (54)**	**1.01M (55)**
Zheng et *al.* [78]	✗	93.29	35.55M (62)	N/A
FPWT [50]	✗	93.76	42.30M (55)	1.06M (53)
SFP [35]	✗	94.06	48.54M (49)	1.28M (43)
SLIMING (Ours)	✓	**94.17**	**34.91M (63)**	**0.76M (66)**

4.3. Results on CIFAR-100

Table 8 displays the pruning outcomes for VGG-16-BN and ResNet-20/32/56/110 on CIFAR-100. Our method outperforms both manual and automatic approaches [66, 9, 63, 52, 76, 16] in terms of accuracy drop and computational overhead reduction.

On the VGG-16-BN model, SLIMING attains the highest top-1 accuracy of 74.18%, while also yielding the most compact model—with a 50% reduction in MACs and a 90% reduction in parameters. In contrast, SPSRC [66] and FPC [8], which manually follow pre-defined pruning heuristics and lack automatic configuration capabilities, deliver lower compression and accuracy. SPSRC [66], in particular, reformulates convolutional layers into matrix multiplications and uses matrix norms to assess filter importance. However, by evaluating filters independently, it overlooks correlations between them, potentially discarding useful information and degrading performance (73.16% accuracy, 65% parameters reduction). On the other hand, HTP-URC [58], an automated method that selects filters based on feature discrimination, slightly achieves lower accuracy (74.03%) and a smaller parameter reduction (81%) than SLIMING. For a higher compression level, SLIMING achieves the highest top-1 accuracy of 72.18% on VGG-16-BN while attaining the highest compression rates—reducing MACs by 86% and parameters by 94%. Notably, Tang et *al.* [68], a very recent state-of-the-art method, evaluates filter importance by estimating the information capacity using entropy, capturing contributions from both individual filters and the overall network. Despite this sophisticated strategy, it only attains 70.54% accuracy with higher MACs and parameters compared to SLIMING. RGP [9], another manual pruning method using graph theory, also falls behind with both lower accuracy and compression.

On the ResNet-20 model, SLIMING also achieves the best trade-off between accuracy and efficiency. It reduces MACs by 60% and parameters by 63%, while maintaining a top-1 accuracy of 67.83%, outperforming prior methods such as PGMPF [3] and ResPrune [30]. Notably, ResPrune employs a two-phase pruning strategy that selects filters based on a combination of L2-norm and redundancy measures. Despite this dual-criterion approach, it achieves a lower accuracy (67.62%) and compresses the model less effectively than SLIMING.

On ResNet-32, SLIMING achieves the best performance with 72.37% accuracy, along with 44% MAC and 45% parameter reduction. In comparison, DAIS [16]—which introduces a differentiable annealing indicator search mechanism inspired by NAS—automatically explores pruning configurations under computation constraints and slightly achieves lower accuracy (72.20%) with marginally higher MACs.

For ResNet-56, SLIMING outperforms both manual and automatic methods, achieving a top-1 accuracy of 73.69%, which is the highest among all approaches. This performance comes with a 30% reduction in MACs and a 29% re-

duction in parameters. Notably, ASTER, an adaptive sensitivity-based pruning method, offers reasonable accuracy (72.26%) but with slightly lower MACs reduction (27%). When we further increase the compression ratio, SLIMING maintains its superiority with a top-1 accuracy of 72.61%, better than other methods like FGC [63], which adopts a self-supervised feature-gate coupling strategy for dynamic pruning (72.00% accuracy).

Overall, these results demonstrate that SLIMING offers a more effective trade-off between model compactness and performance, even compared to advanced recent baselines [68, 9, 30, 63].

Table 8: Pruning results on CIFAR-100

Method	Auto	Top-1	MACs (CR)	Params (CR)
VGG-16-BN [64]		73.45	314.62M (00)	15.04M (00)
SPSRC [66]	✗	73.16	211.64M (33)	5.29M (65)
FPC [8]	✗	73.44	175.63M (44)	1.83M (88)
HTP-URC [58]	✓	74.03	179.63M (43)	2.82M (81)
SLIMING (Ours)	✓	**74.18**	**157.19M (50)**	**1.53M (90)**
RGP [9]	✗	69.33	49.55M (84)	2.39M (84)
Tang et. *al* [68]	✗	70.54	47.82M (85)	1.63M (89)
SLIMING (Ours)	✓	**72.18**	**44.88M (86)**	**0.85M (94)**
ResNet-20 [24]		68.92	41.25M (00)	278.32K (00)
PGMPF [3]	✗	67.62	28.74M (30)	N/A
ResPrune [30]	✗	67.62	28.74M (58)	N/A
SLIMING (Ours)	✓	**67.83**	**16.42M (60)**	**102.33K (63)**
ResNet-32 [24]		71.16	69.78M (00)	472.76K (00)
DAIS [16]	✓	72.20	39.40M (43)	N/A
SLIMING (Ours)	✓	**72.37**	**38.96M (44)**	**261.66K (45)**
ResNet-56 [24]		72.89	126.86M (00)	861.62K (65)
HTP-URC [58]	✓	71.45	95.15M (25)	636.74K (26)
FPC [8]	✗	72.26	90.91M (28)	400.00K (54)
ASTER [76]	✓	72.23	92.00M (27)	N/A
SLIMING (Ours)	✓	**73.69**	**88.45M (30)**	**612.77K (29)**
Li et. *al* [36]	✗	70.12	68.89M (46)	726.00K (16)
ASTER [76]	✓	71.78	58.00M (54)	N/A
PGMPF [3]	✗	70.21	59.40M (53)	N/A
Lu *et al.* [52]	✓	71.15	59.80M (53)	N/A
FGC [63]	✓	72.00	74.76M (41)	N/A
SLIMING (Ours)	✓	**72.61**	**54.19M (57)**	**385.30K (55)**

4.4. Results on ImageNet

ResNet-34/50. To assess the scalability of SLIMING, we conduct experiments on the ImageNet dataset by addressing ResNet-34/50 as shown in Table 9. Across all compression levels, our method consistently outperforms other approaches in terms of both performance and complexity reduction. Notably, on ResNet-34, SLIMING outperforms UPDP [46]—an automated layer pruning method using reparameterization—by achieving a 0.99% higher Top-1 accuracy (73.39% vs. 72.40%) and a 21% greater reduction in MACs (51% vs. 30%). For ResNet-50, compared to SPWB [1], a recent automated channel pruning approach using weight blending, the compressed model of SLIMING has 13% fewer parameters and requires less than 12% MACs, while achieving 0.34% higher top-1 accuracy.

Table 9: Compression results of ResNet-50 on ImageNet

Method	Auto	Top-1	Top-5	MACs (CR)	Params (CR)
ResNet-34 [24]		73.45	91.48	3.67B (00)	21.80M (00)
SPSRC [66]	✗	72.12	90.56	2.30B (37)	17.88M (18)
UPDP [46]	✓	72.40	N/A	2.51B (30)	N/A
RLAL [13]	✓	73.31	91.25	1.85B (50)	N/A
SLIMING (Ours)	✓	**73.39**	**91.42**	**1.81B (51)**	**10.76M (51)**
ResNet-50 [24]		76.15	92.87	4.12G (00)	25.56M (00)
REAF [75]	✓	75.17	92.44	2.16G (48)	14.57M (43)
RGP [9]	✗	75.30	92.55	2.30G (44)	14.34M (44)
Chen *et al.* [7]	✗	75.60	92.58	2.21G (46)	N/A
C-SGD [22]	✗	75.80	92.65	2.19G (47)	14.58M (43)
CHIP [65]	✗	76.15	92.91	2.10G (49)	14.23M (44)
SFI-FP [73]	✗	76.29	93.08	2.10G (49)	14.23M (44)
PEEL [26]	✗	76.50	N/A	2.20G (46)	N/A
SLIMING (Ours)	✓	**76.74**	**93.43**	**2.09G (49)**	**13.27M (48)**
CIE [38]	✓	74.06	91.87	1.56G (62)	9.98M (61)
RGP [9]	✗	74.58	92.09	1.92G (53)	11.99M (53)
MFP [25]	✗	74.86	92.43	1.88G (54)	N/A
FPWT [50]	✗	75.01	92.45	1.89G (54)	12.86M (50)
Torque [20]	✗	75.07	N/A	1.99G (51)	9.68M (62)
OTOv2 [5]	✓	75.20	92.22	1.53G (63)	N/A
FiltDivNet [34]	✓	75.23	92.50	1.66G (59)	15.62M (39)
ASTER [76]	✓	75.27	92.47	1.51G (63)	N/A
C-SGD [22]	✗	75.29	92.39	1.82G (55)	12.37M (52)
Hu et. *al* [27]	✗	75.30	92.40	1.81G (56)	17.86M (30)
SPWB [1]	✓	75.62	N/A	2.01G (51)	12.94M (49)
DCFF [41]	✗	75.60	92.55	1.52G (63)	11.05M (57)
HTP-URC [58]	✓	75.81	N/A	1.88G (54)	15.81M (38)
SLIMING (Ours)	✓	**75.96**	**93.29**	**1.51G (63)**	**9.68M (62)**
HBFP [2]	✗	69.17	N/A	0.94G (76)	8.09M (68)
CHIP [65]	✗	72.30	90.74	0.95G (77)	8.01M (69)
SNACS [59]	✓	72.60	N/A	1.98G (52)	7.92M (69)
RGP [9]	✗	72.68	91.06	0.94G (77)	8.13M (68)
FPWT [50]	✗	72.82	91.14	1.02G (75)	6.38M (75)
SFI-FP [73]	✗	73.48	92.87	0.96G (77)	8.03M (69)
ACSC [45]	✓	73.68	N/A	1.03G (75)	6.31M (75)
DCFF [41]	✗	73.81	91.59	1.02G (75)	6.56M (74)
Guo *et al.* [19]	✓	73.84	92.07	1.19G (71)	6.25M (75)
SLIMING (Ours)	✓	**73.88**	**92.07**	**0.87G (79)**	**5.68M (78)**

MobileNetV2. Table 10 shows the pruning results of MobileNetV2 on ImageNet. Our method exhibits superior performance compared to other competing methods [19, 14, 58, 13, 52, 18, 41]. For example, compared to DCFF [41], a manual pruning ratio configuration approach using dynamic-coded filter fusion, SLIMING achieves a similar reduction in MACs (approximately 53%) while pruning over 23% more parameters and achieving higher accuracy. Compared to RLAL [13], a very recent method in automatic structured pruning using RL, SLIMING gives comparable accuracy (71.89% vs. 71.82%) while reducing more than 9% of MACs.

Table 10: Pruning results of MobileNetV2 on Imagenet

Method	Auto	Top-1	MACs (CR)	Params (CR)
MobileNetV2 [61]		71.89	300.78M (00)	3.51M (00)
SP [17]	✓	70.40	200.29M (33)	N/A
DCP [47]	✗	71.39	208.53M (31)	2.28M (35)
Lu et *al.* [52]	✓	71.61	212.00M (30)	N/A
HTP-URC [58]	✓	71.68	210.55M (30)	N/A
UDSP [14]	✗	71.72	190.69M (37)	2.97M (15)
RLAL [13]	✓	71.82	226.09M (29)	N/A
SLIMING (Ours)	✓	**71.89**	**187.11M (38)**	**2.25M (36)**
EvoFC [6]	✓	61.63	198.21M (34)	3.14M (11)
SFP [35]	✗	67.89	187.99M (38)	2.68M (24)
DCFF [41]	✗	68.60	140.46M (53)	2.62M (25)
SLIMING (Ours)	✓	**69.62**	**140.09M (53)**	**1.83M (48)**

4.5. Results on COCO

To evaluate SLIMING's effectiveness in downstream tasks, we used our compressed ResNet-50/ImageNet as the backbone for training Faster/Mask/Keypoint-RCNN on COCO, as outlined in Table 11. Our method shows promising results in terms of precision and recall, along with achieving relatively higher compression levels compared to other approaches [55, 56]. Remarkably, SLIMING significantly enhances inference throughput, resulting in over a $2\times$ improvement in frames per second (FPS) compared to the baseline models. For instance, MaskRCNN exhibits a reduction in end-to-end latency from 100 ms to 42 ms, achieving a real-time framerate of 24 FPS. It is worth emphasizing that these performance evaluations were conducted on a GTX 3060 GPU, providing robust evidence of the real-world applicability of our approach. These results highlight SLIMING's potential as a valuable tool for enhancing neural network efficiency and effectiveness in demanding tasks such as real-world object detection, instance segmentation, and keypoint detection.

Table 11: Pruning results of Faster/Mask/Keypoint-RCNN-ResNet50-FPN on COCO-2017

Model	Auto	$AP^{0.5:0.95}$	$AP^{0.5}$	$AP^{0.75}$	AR^{1}	AR^{10}	AR^{100}	MACs (CR)	Params (CR)	FPS	Latency (ms)
FasterRCNN [60]		36.91	58.54	39.61	30.73	48.46	50.84	134.85G (00)	41.81M (00)	12	84
CORING [55]	✗	35.57	56.05	37.81	30.21	48.28	50.79	92.23G (32)	24.04M (43)	25	40
SLIMING (Ours)	✓	**35.64**	**56.56**	**38.22**	**30.71**	**49.05**	**51.63**	**90.56G (33)**	**22.52M (46)**	**28**	**36**
MaskRCNN [23]		34.52	56.02	36.73	29.66	45.56	47.56	134.85G (00)	44.46M (00)	10	100
CORING [55]	✗	32.77	53.53	34.57	28.93	44.94	47.11	92.23G (32)	26.68M (40)	22	45
SLIMING (Ours)	✓	**33.14**	**54.02**	**35.09**	**29.41**	**45.83**	**48.11**	**90.56G (33)**	**25.16M (43)**	**24**	**42**
				$AR^{0.5:0.95}$	$AR^{0.5}$	$AR^{0.75}$					
KeypointRCNN [23]		65.06	86.11	71.38	71.76	90.74	77.42	137.42G (00)	59.64M (00)	9	112
NORTON [56]	✗	62.81	85.04	68.57	69.34	89.56	74.62	95.97G (30)	**39.39M (34)**	17	59
SLIMING (Ours)	✓	**63.62**	**85.13**	**69.35**	**69.90**	**89.82**	**74.94**	**90.56G (33)**	39.90M (33)	**18**	**53**

5. Discussions

5.1. Pruning efficiency

To empirically analyze the pruning efficiency, we conduct experiments comparing the compression time of our method, SLIMING, with five other approaches, including WhiteBox [77], ABCPruner [43], DCFF [41], CORING [55], and NORTON [56]. Table 12 presents the time cost (compression and fine-tuning time) and compression results (top-1 accuracy, MACs reduction, and parameters reduction) across six models (VGG-16-BN/VGG-19-BN [64], ResNet-56/110/50 [24], and GoogLeNet [67]) on three datasets (CIFAR-10/100 [32] and ImageNet [11]).

WhiteBox [77] belongs to the family of feature map-based methods [77, 50, 19, 74, 38, 21, 4, 58, 7, 42, 65, 8], which assess the importance of channels based on their corresponding feature maps. WhiteBox requires several training epochs (e.g., 30 epochs on CIFAR-10) to evaluate the contribution of each channel, making its pruning process time-intensive, especially for larger models and datasets. In contrast, SLIMING enables parallel filter selection across all layers—unlike feature map-based approaches, which require sequential, layer-wise processing. As a result, SLIMING not only compresses models faster but also achieves higher accuracy in three out of three experiments compared to WhiteBox, as evidenced by the results on ResNet-56/110/50 on CIFAR-10/ImageNet in Table 12.

ABCPruner [43] is a method tailored to tackle the combinatorial challenge of the automatic channel pruning problem using the artificial bee colony algorithm. This approach belongs to the family of NAS-based approaches [6, 16, 52, 33, 43]. Building based on a complex numerical optimization method [31], ABCPruner consumes a lot of time to search for the optimal pruning structure (e.g., 26968 seconds for GoogLeNet/CIFAR-10). In the cases of VGG-16-BN and GoogLeNet on CIFAR-10, SLIMING demonstrates shorter pruning times and superior compression results compared to ABCPruner.

We also compare SLIMING with DCFF [41], CORING [55], and NORTON [56], three recent approaches in network compression. DCFF employs a pruning-during-training strategy, which significantly increases its training time. SLIMING proves to be more time-efficient (9754 seconds vs. 15180 seconds) while yielding better compression results on GoogLeNet/CIFAR-10. NORTON, a hybrid compression technique combining tensor decomposition and pruning, is also outperformed by SLIMING in both compression time and efficacy on VGG-19-BN/CIFAR-100. Compared to CORING, a tensor decomposition-based filter pruning approach, SLIMING achieves comparable top-1 accuracy (93.74% vs. 93.68%) while significantly reducing both pruning time (485 seconds vs. 810 seconds) and fine-tuning time (1548 seconds vs. 3935 seconds).

In conclusion, SLIMING delivers competitive or superior compression performance across six models and three datasets, while consistently achieving significantly lower compression time compared to the five evaluated methods. These results highlight SLIMING's practicality for real-world deployment scenarios where pruning efficiency is considered alongside pruning effectiveness.

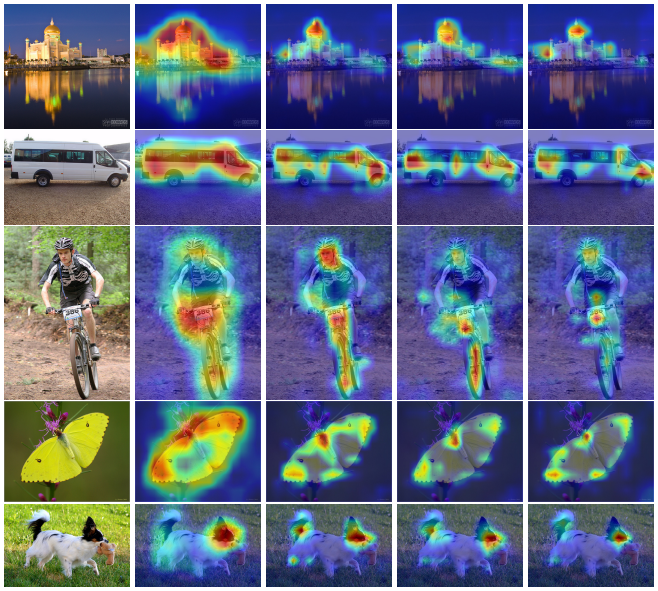
Table 12: Pruning efficiency comparison

Model	Dataset	Method	Time cost (seconds)			Acc.	Compression (%)	
			Prune	Finetune	Total		MACs	Params
VGG-16-BN [64]	CIFAR-10 [32]	ABCPruner [43]	5387	1712	7099	93.08	74	89
		CORING [55]	810	3935	4745	93.68	79	87
		SLIMING (ours)	**485**	**1548**	**2033**	**93.74**	**81**	**92**
ResNet-56 [24]	CIFAR-10 [32]	WhiteBox [77]	2809	5231	8040	93.54	56	45
		SLIMING (ours)	**652**	**4597**	**5249**	**94.25**	**58**	**57**
ResNet-110 [24]	CIFAR-10 [32]	WhiteBox [77]	5213	9547	14760	94.12	65	62
		SLIMING (ours)	**943**	**9102**	**10045**	**94.52**	**66**	**65**
GoogLeNet [67]	CIFAR-10 [32]	ABCPruner [43]	26968	9881	36849	94.84	67	60
		DCFF [41]	N/A	15180	15180	94.92	70	66
		SLIMING (ours)	**1372**	**9754**	**11126**	**95.31**	**81**	**77**
VGG-19-BN [64]	CIFAR-100 [32]	NORTON [56]	1793	4159	5952	74.62	55	56
		SLIMING (ours)	**852**	**3043**	**3895**	**75.01**	**58**	**62**
ResNet-50 [24]	ImageNet [11]	WhiteBox [77]	100473	972278	1072751	75.32	46	31
		SLIMING (ours)	**15924**	**871295**	**887219**	**76.74**	**49**	**48**

5.2. Visualizing feature preservation

We conduct a qualitative assessment of feature preservation in SLIMING, complementing the established efficiency demonstrated by numerical results. Specifically, we randomly select 5 images from the ImageNet validation dataset [11] and evaluate three compression levels for the original ResNet-50 [24] model: 49%, 63%, and 79% (refer

to Table 9). Utilizing GradCAM [62] for interpretation, we visualize and analyze feature maps in both the original and compressed models, as shown in Figure 3. The visual illustration showcases SLIMING's effectiveness in retaining crucial features across diverse classes. Notably, at different compression levels, SLIMING consistently demonstrates robustness in capturing and preserving essential information across a variety of patterns.



(a) Input (b) CR = 0% (c) CR = 49% (d) CR = 63% (e) CR = 79%

Figure 3: Assessment of feature preservation in pruned models.

6. Conclusion

This paper tackles the challenge of automatic structured pruning by framing it as a constrained combinatorial optimization problem within the realm of multilinear algebra. Our method, grounded in tensor decomposition and the analysis of singular values, effectively exposes the relationship between network redundancy and these singular values. The incorporation of two proposed algorithms extends the adaptability of our approach, facilitating the automatic determination of pruning configuration and efficient filter selection. Through extensive experiments spanning diverse architectures, datasets, and tasks, our proposed framework consistently demonstrates its efficacy and utility in network compression.

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, France.

References

[1] Agarwal, P., Mathew, M., Patel, K.R., Tripathi, V., Swami, P., 2024. Prune efficiently by soft pruning, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 2210–2217.

[2] Basha, S.S., Farazuddin, M., Pulabaigari, V., Dubey, S.R., Mukherjee, S., 2024. Deep model compression based on the training history. Neurocomputing 573, 127257.

[3] Cai, L., An, Z., Yang, C., Yan, Y., Xu, Y., 2022. Prior gradient mask guided pruning-aware fine-tuning. Proceedings of the AAAI Conference on Artificial Intelligence 36, 140–148.

[4] Chang, J., Lu, Y., Xue, P., Xu, Y., Wei, Z., 2023. Iterative clustering pruning for convolutional neural networks. Knowledge-Based Systems 265, 110386.

[5] Chen, T., Liang, L., Ding, T., Zhu, Z., Zharkov, I., 2023a. Otov2: Automatic, generic, user-friendly, in: The Eleventh International Conference on Learning Representations, ICLR, pp. 1–13.
[6] Chen, X., Liu, C., Hu, P., Lin, J., Gong, Y., Chen, Y., Peng, D., Geng, X., 2024a. Evolving filter criteria for randomly initialized network pruning in image classification. Neurocomputing , 127872.
[7] Chen, Y., Li, R., Li, W., Wang, J., Li, R., 2023b. Three-stage global channel pruning for resources-limited platform. IEEE Transactions on Neural Networks and Learning Systems , 1–14.
[8] Chen, Y., Wen, X., Zhang, Y., He, Q., 2022. Fpc: Filter pruning via the contribution of output feature map for deep convolutional neural networks acceleration. Knowledge-Based Systems 238, 107876.
[9] Chen, Z., Xiang, J., Lu, Y., Xuan, Q., Wang, Z., Chen, G., Yang, X., 2024b. Rgp: Neural network pruning through regular graph with edges swapping. IEEE Transactions on Neural Networks and Learning Systems 35, 14671–14683.
[10] De Lathauwer, L., De Moor, B., Vandewalle, J., 2000. A multilinear singular value decomposition. SIAM Journal on Matrix Analysis and Applications 21, 1253–1278.
[11] Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L., 2009. Imagenet: A large-scale hierarchical image database, in: 2009 IEEE conference on computer vision and pattern recognition, Ieee. pp. 248–255.
[12] Eckart, C., Young, G.M., 1936. The approximation of one matrix by another of lower rank. Psychometrika 1, 211–218.
[13] Ganjdanesh, A., Gao, S., Huang, H., 2024. Jointly training and pruning cnns via learnable agent guidance and alignment, in: 2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 16058–16069.
[14] Gao, S., Zhang, Y., Huang, F., Huang, H., 2024a. Bilevelpruning: Unified dynamic and static channel pruning for convolutional neural networks, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 16090–16100.
[15] Gao, W., Guo, Y., Ma, S., Li, G., Kwong, S., 2024b. Efficient neural network compression inspired by compressive sensing. IEEE Transactions on Neural Networks and Learning Systems 35, 1965–1979.
[16] Guan, Y., Liu, N., Zhao, P., Che, Z., Bian, K., Wang, Y., Tang, J., 2023. Dais: Automatic channel pruning via differentiable annealing indicator search. IEEE Transactions on Neural Networks and Learning Systems 34, 9847–9858.
[17] Guo, S., Lai, B., Yang, S., Zhao, J., Shen, F., 2023a. Sensitivity pruner: Filter-level compression algorithm for deep neural networks. Pattern Recognition 140, 109508.
[18] Guo, S., Zhang, L., Zheng, X., Wang, Y., Li, Y., Chao, F., Wu, C., Zhang, S., Ji, R., 2023b. Automatic network pruning via hilbert-schmidt independence criterion lasso under information bottleneck principle, in: Proceedings of the IEEE/CVF international conference on computer vision, pp. 17458–17469.
[19] Guo, Y., Gao, W., Li, G., 2024. Interpretable task-inspired adaptive filter pruning for neural networks under multiple constraints. International Journal of Computer Vision , 1–17.
[20] Gupta, A., Bau, T., Kim, J., Zhu, Z., Jha, S., Garud, H., 2024. Torque based structured pruning for deep neural network, in: 2024 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV), pp. 2699–2708.
[21] Han, X., Chu, Y., Wang, K., Wang, L., Yue, L., Ding, W., 2024. Tailor: Inter-feature distinction filter fusion pruning. Information Sciences .
[22] Hao, T., Ding, X., Han, J., Guo, Y., Ding, G., 2023. Manipulating identical filter redundancy for efficient pruning on deep and complicated cnn. IEEE Transactions on Neural Networks and Learning Systems , 1–14.
[23] He, K., Gkioxari, G., Dollár, P., Girshick, R., 2020. Mask r-cnn. IEEE Transactions on Pattern Analysis and Machine Intelligence 42, 386–397.
[24] He, K., Zhang, X., Ren, S., Sun, J., 2016. Deep residual learning for image recognition, in: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 770–778.
[25] He, Y., Liu, P., Zhu, L., Yang, Y., 2023. Filter pruning by switching to neighboring cnns with good attributes. IEEE Transactions on Neural Networks and Learning Systems 34, 8044–8056.
[26] Hou, Y., Ma, Z., Liu, C., Wang, Z., Loy, C.C., 2024. Network pruning via resource reallocation. Pattern Recognition 145, 109886.
[27] Hu, F., Zhang, J., Gao, S., Lin, Y., Zhou, W., Wang, R., 2024a. An efficient training-from-scratch framework with bn-based structural compressor. Pattern Recognition , 110546.
[28] Hu, W., Che, Z., Liu, N., Li, M., Tang, J., Zhang, C., Wang, J., 2024b. Catro: Channel pruning via class-aware trace ratio optimization. IEEE Transactions on Neural Networks and Learning Systems 35, 11595–11607.
[29] Huang, G., Liu, Z., Van Der Maaten, L., Weinberger, K.Q., 2017. Densely connected convolutional networks, in: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2261–2269.
[30] Jayasimhan, A., Pabitha, P., 2024. Resprune: An energy-efficient restorative filter pruning method using stochastic optimization for accelerating cnn. Pattern Recognition , 110671.
[31] Karaboga, D., et al., 2005. An idea based on honey bee swarm for numerical optimization .
[32] Krizhevsky, A., Hinton, G., et al., 2009. Learning multiple layers of features from tiny images .
[33] Lee, S., Song, B.C., 2024. Fast filter pruning via coarse-to-fine neural architecture search and contrastive knowledge transfer. IEEE Transactions on Neural Networks and Learning Systems 35, 9674–9685.
[34] Lei, P., Liang, J., Zheng, T., Wang, J., 2024. Compression of convolutional neural networks with divergent representation of filters. IEEE Transactions on Neural Networks and Learning Systems 35, 4125–4137.
[35] Li, G., Li, R., Li, T., Shen, C., Zou, X., Wang, J., Wang, C., Li, N., 2025. Sfp: Similarity-based filter pruning for deep neural networks. Information Sciences 689, 121418.
[36] Li, J., Shao, H., Deng, X., Jiang, Y., 2024a. Efficient filter pruning: Reducing model complexity through redundancy graph decomposition. Neurocomputing , 128108.
[37] Li, S., Chen, J., Liu, S., Zhu, C., Tian, G., Liu, Y., 2024b. Mcmc: Multi-constrained model compression via one-stage envelope reinforcement learning. IEEE Transactions on Neural Networks and Learning Systems , 1–13.
[38] Lian, Y., Peng, P., Jiang, K., Xu, W., 2024. Cross-layer importance evaluation for neural network pruning. Neural Networks 179, 106496.
[39] Lin, H., Peng, Y., Zhang, Y., Bie, L., Zhao, X., Gao, Y., 2025. Filter pruning by high-order spectral clustering. IEEE Transactions on Pattern Analysis and Machine Intelligence 47, 2402–2415.

[40] Lin, M., Cao, L., Zhang, Y., Shao, L., Lin, C.W., Ji, R., 2023a. Pruning networks with cross-layer ranking & k-reciprocal nearest filters. IEEE Transactions on Neural Networks and Learning Systems 34, 9139–9148.
[41] Lin, M., Chen, B., Chao, F., Ji, R., 2023b. Training compact cnns for image classification using dynamic-coded filter fusion. IEEE Transactions on Pattern Analysis and Machine Intelligence 45, 10478–10487.
[42] Lin, M., Ji, R., Wang, Y., Zhang, Y., Zhang, B., Tian, Y., Shao, L., 2020a. Hrank: Filter pruning using high-rank feature map, in: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 1529–1538.
[43] Lin, M., Ji, R., Zhang, Y., Zhang, B., Wu, Y., Tian, Y., 2020b. Channel pruning via automatic structure search, in: Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, pp. 1–7.
[44] 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: Computer vision–ECCV 2014: 13th European conference, zurich, Switzerland, September 6-12, 2014, proceedings, part v 13, Springer. pp. 740–755.
[45] Liu, J., Liu, W., Li, Y., Hu, J., Cheng, S., Yang, W., 2024a. Attention-based adaptive structured continuous sparse network pruning. Neurocomputing 590, 127698.
[46] Liu, J., Tang, D., Huang, Y., Zhang, L., Zeng, X., Li, D., Lu, M., Peng, J., Wang, Y., Jiang, F., et al., 2024b. Updp: A unified progressive depth pruner for cnn and vision transformer, in: Proceedings of the AAAI Conference on Artificial Intelligence, pp. 13891–13899.
[47] 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 44, 4035–4051.
[48] Liu, X., Li, B., Chen, Z., Yuan, Y., 2023a. Generalized gradient flow based saliency for pruning deep convolutional neural networks. International Journal of Computer Vision 131, 3121–3135.
[49] Liu, Y., Fan, K., Wu, D., Zhou, W., 2023b. Filter pruning by quantifying feature similarity and entropy of feature maps. Neurocomputing 544, 126297.
[50] Liu, Y., Fan, K., Zhou, W., 2024c. Fpwt: Filter pruning via wavelet transform for cnns. Neural Networks 179, 106577.
[51] Loshchilov, I., Hutter, F., 2017. SGDR: Stochastic gradient descent with warm restarts, in: International Conference on Learning Representations, pp. 1–16.
[52] Lu, X., Dong, W., Li, X., Wu, J., Li, L., Shi, G., 2023. Adaptive search-and-training for robust and efficient network pruning. IEEE Transactions on Pattern Analysis and Machine Intelligence 45, 9325–9338.
[53] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S., 2019. Pytorch: an imperative style, high-performance deep learning library, in: Advances in Neural Information Processing Systems, pp. 1–12.
[54] Pham, V.T., Zniyed, Y., Nguyen, T.P., 2023. Élagage efficace des filtres basé sur les décompositions tensorielles, in: XXIXème Colloque Francophone de Traitement du Signal et des Images, GRETSI - Groupe de Recherche en Traitement du Signal et des Images. pp. 937–940.
[55] Pham, V.T., Zniyed, Y., Nguyen, T.P., 2024a. Efficient tensor decomposition-based filter pruning. Neural Networks , 106393.
[56] Pham, V.T., Zniyed, Y., Nguyen, T.P., 2024b. Enhanced network compression through tensor decompositions and pruning. IEEE Transactions on Neural Networks and Learning Systems , 1–13.
[57] Pham, V.T., Zniyed, Y., Nguyen, T.P., 2024c. Hybrid network compression through tensor decompositions and pruning, in: 32nd European Signal Processing Conference, EUSIPCO 2024, pp. 1–5.
[58] Qian, Y., He, Z., Wang, Y., Wang, B., Ling, X., Gu, Z., Wang, H., Zeng, S., Swaileh, W., 2024. Hierarchical threshold pruning based on uniform response criterion. IEEE Transactions on Neural Networks and Learning Systems 35, 10869–10881.
[59] Ravi Ganesh, M., Blanchard, D., Corso, J.J., Sekeh, S.Y., 2024. Slimming neural networks using adaptive connectivity scores. IEEE Transactions on Neural Networks and Learning Systems 35, 3794–3808.
[60] 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.
[61] Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.C., 2018. Mobilenetv2: Inverted residuals and linear bottlenecks, in: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 4510–4520.
[62] 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. International Journal of Computer Vision 128, 336–359.
[63] Shi, M., Liu, C., Jiao, J., Ye, Q., 2024. Self-supervised feature-gate coupling for dynamic network pruning. Pattern Recognition , 110594.
[64] Simonyan, K., Zisserman, A., 2015. Very deep convolutional networks for large-scale image recognition, in: 3rd International Conference on Learning Representations, pp. 1–14.
[65] Sui, Y., Yin, M., Xie, Y., Phan, H., Aliari Zonouz, S., Yuan, B., 2021. Chip: Channel independence-based pruning for compact neural networks. Advances in Neural Information Processing Systems 34, 24604–24616.
[66] Sun, X., Shi, H., 2024. Towards better structured pruning saliency by reorganizing convolution, in: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, pp. 2204–2214.
[67] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A., 2015. Going deeper with convolutions, in: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1–9.
[68] Tang, X., Ye, S., Shi, Y., Hu, T., Peng, Q., You, X., 2024. Filter pruning based on information capacity and independence. IEEE Transactions on Neural Networks and Learning Systems .
[69] Tokcan, N., Sofi, S.S., Pham, V.T., Prévost, C., Kharbech, S., Magnier, B., Nguyen, T.P., Khoshnam, A., Zniyed, Y., de Lathauwer, L., et al., 2025. Tensor decompositions for signal processing: Theory, advances, and applications. Signal Processing .
[70] Tucker, L.R., 1966. Some mathematical notes on three-mode factor analysis. Psychometrika 31, 279–311.
[71] Wang, Y., Guo, S., Guo, J., Zhang, J., Zhang, W., Yan, C., Zhang, Y., 2024. Towards performance-maximizing neural network pruning via global channel attention. Neural Networks 171, 104–113.
[72] Wu, X., Gao, S., Zhang, Z., Li, Z., Bao, R., Zhang, Y., Wang, X., Huang, H., 2024. Auto- train-once: Controller network guided automatic network pruning from scratch, in: 2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 16163–16173.
[73] Yang, L., Gu, S., Shen, C., Zhao, X., Hu, Q., 2024. Soft independence guided filter pruning. Pattern Recognition 153, 110488.

[74] Yuan, T., Li, Z., Liu, B., Tang, Y., Liu, Y., 2024. Arpruning: An automatic channel pruning based on attention map ranking. Neural Networks , 106220.
[75] Zhang, X., Xie, W., Li, Y., Jiang, K., Fang, L., 2023a. Reaf: Remembering enhancement and entropy-based asymptotic forgetting for filter pruning. IEEE Transactions on Image Processing 32, 3912–3923.
[76] Zhang, Y., Freris, N.M., 2024. Adaptive filter pruning via sensitivity feedback. IEEE Transactions on Neural Networks and Learning Systems 35, 10996–11008.
[77] Zhang, Y., Lin, M., Lin, C.W., Chen, J., Wu, Y., Tian, Y., Ji, R., 2023b. Carrying out cnn channel pruning in a white box. IEEE Transactions on Neural Networks and Learning Systems 34, 7946–7955.
[78] Zheng, Y., Sun, P., Ren, Q., Xu, W., Zhu, D., 2024. A novel and efficient model pruning method for deep convolutional neural networks by evaluating the direct and indirect effects of filters. Neurocomputing 569, 127124.