PeerJ Computer Science

MLPruner: pruning convolutional neural networks with automatic mask learning

Sihan Chen and Ying Zhao

School of Digital and Intelligence Industry, Inner Mongolia University of Science and Technology, Bao tou, Inner Mongolia Autonomous Region, China

ABSTRACT

In recent years, filter pruning has been recognized as an indispensable technique for mitigating the significant computational complexity and parameter burden associated with deep convolutional neural networks (CNNs). To date, existing methods are based on heuristically designed pruning metrics or implementing weight regulations to penalize filter parameters during the training process. Nevertheless, human-crafted pruning criteria tend not to identify the most critical filters, and the introduction of weight constraints can inadvertently interfere with weight training. To rectify these obstacles, this article introduces a novel mask learning method for autonomous filter pruning, negating requirements for weight penalties. Specifically, we attribute a learnable mask to each filter. During forward propagation, the mask is transformed to a binary value of 1 or 0, serving as indicators for the necessity of corresponding filter pruning. In contrast, throughout backward propagation, we use straight-through estimator (STE) to estimate the gradient of masks, accommodating the non-differentiable characteristic of the rounding function. We verify that these learned masks aptly reflect the significance of corresponding filters. Concurrently, throughout the mask learning process, the training of neural network parameters remains uninfluenced, therefore protecting the normal training process of weights. The efficacy of our proposed filter pruning method based on mask learning, termed MLPruner, is substantiated through its application to prevalent CNNs across numerous representative benchmarks.

Subjects Computer Vision, Data Mining and Machine Learning, Neural Networks
Keywords Filter pruning, Mask learning, Straight-through estimator

Submitted 14 November 2024
Accepted 29 July 2025
Published 25 August 2025

Corresponding author
Ying Zhao, amengs@imust.edu.cn

Academic editor
Stefano Cirillo

Additional Information and Declarations can be found on page 16

DOI **10.7717/peerj-cs.3132**

INTRODUCTION

In recent years, the vision community has swiftly enhanced the performance of deep convolution neural networks (CNNs) across various tasks, including image classification (*He et al., 2016*; *Jangra et al., 2023*), object detection (*He et al., 2017*), and semantic segmentation (*Girshick et al., 2014*). Predominantly, these advancements are propelled by an escalating parameter load and computational expense. This trend, unfortunately, renders deep neural networks (DNNs) onerous for deployment on resource-limited edge devices, such as smartphones and Internet of Things (IoT) apparatuses. In response, there has been a burgeoning interest in model compression research (*Hubara et al., 2016*; *Howard et al., 2017*; *Lin et al., 2020a*), aiming to diminish the model's size while maintaining comparable efficacy to the original model, thereby mitigating the deployment challenges of DNNs.

Broadly speaking, DNN compression strategies can coalesce into four distinct categories: (1) Network quantization achieves compression of an already trained model through a reduction in the number of bits utilized for the weight parameter representation (*Hubara et al., 2016*; *Liu et al., 2018*; *Lin et al., 2020b*). (2) Tensor factorization seeks to approximate the weight tensor with a sequence of low-rank matrices organized in a sum-product outline (*Lin et al., 2018*; *Hayashi et al., 2019*). (3) Compactly designed networks such as ShuffleNets (*Zhang et al., 2018*) and MobileNets (*Howard et al., 2017*; *Sandler et al., 2018*) that involves various light-weight convolution module. (4) Network pruning discards superfluous weights of CNNs in filter/weight/block manner (*Han et al., 2015*; *Lin et al., 2021*; *Liu et al., 2019*).

In this article, we focus on the last category for CNNs compression, particularly for CNNs filter pruning (*Lin et al., 2021*; *Liu et al., 2019*; *Zhang et al., 2022*). Filter pruning removes entire convolutional filters in the original CNNs to yield a structured pruned model, aiming at retaining the performance of the original model while drastically reducing the FLoat-Point Operations (FLOPs) and parameter burden. It has received ever-increasing focus due to the practical compression effect as the compressed network can be well supported by regular hardware and off-the-shelf basic linear algebra subprograms (BLAS) library. The existing research on filter pruning can be roughly divided into two categories, which we specifically depict below.

The first group adheres to a three-step pruning pipeline, which encompasses pre-training the model, removing unimportant filters, and meticulously fine-tuning the resulting pruned model. Generally, the majority of methodologies in this domain tend to prioritize the second stage, deploying a wide range of filter importance estimation strategies such as ℓ_1-norm (*Li et al., 2017*), geometric data (*He et al., 2019*), and activation sparsity (*Hu et al., 2016*). Though characterized by their simplicity, these hand-crafted criteria for the quantification of filter importance often fail to accurately excise filters crucial to the overall performance. In sharp divergence, the secondary category effectuates filter pruning *via* training the network with additional sparse constraints on individual filters (*Huang & Wang, 2018*; *Luo & Wu, 2020*). Consequently, the pruned model becomes accessible upon the elimination of zero-valued filters or those becoming beneath a pre-determined threshold. However, these methods often disrupt the normal training process of the network due to the imposed penalties on weight norms, thereby also resulting in sub-optimal effects. In summary, how to automatically identify and eliminate redundant filters during the training process, while preserving the stable optimization of model parameters, remains a urgent issue in the community.

In response to the above obstacle, we propose MLPruner, a novel method that performs automatic mask learning to enable end-to-end filter pruning without any penalty on weights during training. MLPruner assigns a learnable mask to each convolutional filter, where the mask length is basically the same as the filter number. In the forward propagation process, we employ a pre-defined threshold to round the mask to either 0 or 1, which instructs whether to prune or retain the corresponding filter. During backpropagation, given the non-differentiable nature of the round function, we further

employ the straight-through estimator (STE) to update the mask's gradient. We proved that such trained masks can effectively reflect the importance of the corresponding filter. To elaborate, if the mask value is high, the removal of the corresponding filter will have a significant impact on the network training loss, and vice versa. Consequently, filter pruning can be seamlessly executed during training, even without the imposition of any ℓ_2 penalty on weights or interference in forward propagation, given that the filter is always multiplied by masks in binary form.

Extensive experiments on image classification tasks using many popular classification networks including VGG-Net (*Simonyan & Zisserman, 2015*), GoogLeNet (*Szegedy et al., 2015*), ResNet-56/110 (*He et al., 2016*), and Mobilenet-V2 (*Sandler et al., 2018*) demonstrate the superiority of our MLPruner over many state-of-the-arts. For instance, MLPruner removes 54.8% FLOPs of ResNet-56 while still achieving 93.31% top-1 accuracy on CIFAR-10, surpassing the recent baseline HRank (*He et al., 2018a*) that reaches 93.17% accuracy while removing less FLOPs of 50.0%. Our contributions in this article are summarized as follows:

- We propose MLPruner, a novel CNNs filter pruning method that assigns learnable masks to automatically prune filters in an end-to-end training manner, without any weight penalty during training as previous works do.
- We proved that the masks learned by MLPruner can well can well tell the relative importance between filters, enabling both accurate and automatic filter pruning.
- Extensive experiments on pruning representative models have demonstrated the advantages of our proposed MLPruner for compressing CNNs when compared with a wide range of pruning methods.

RELATED WORK

This section covers the spectrum of studies on pruning CNNs that closely related to our work. A more comprehensive overview can be found in the recent survey (*Hoefler et al., 2021*).

Network sparsity

By eliminating superfluous network weights (*LeCun, Denker & Solla, 1989*; *Han et al., 2015*; *He, Zhang & Sun, 2017*), network sparsity/pruning has evolved into a contemporary tool for acquiring lightweight sparse models. The practice of discarding individual weights at random positions, fine-grained sparsity, successfully boasts a high sparse ratio whilst maintaining performance assurance (*Han et al., 2015*; *Evci et al., 2020*). RigL (*Evci et al., 2020*) periodically alternates between the removal and reinstatement of weights, determined by magnitudes and gradients. Sparse Momentum (*Dettmers & Zettlemoyer, 2019*) takes into account the average momentum magnitude within each layer to reallocate weights. Regrettably, the consequent unstructured sparse weights struggle to effectuate acceleration on standard hardware. Unstructured sparsity proved successful in preserving performance, even with a high sparsity exceeding 95% (*Liu et al., 2021*). However, the ensuing irregular sparse tensors resulted in negligible speed improvements on common

hardware (*Wang, 2020*). Coarse-grained sparsity, which typically removes an entire weight block (*Ji et al., 2018*; *Meng et al., 2020*) or convolution filter (*Liu et al., 2019*; *Lin et al., 2020a*), bears more hardware compatibility. Contrasted with fine-grained sparsity, the compressed model enjoys a substantial speed increase, albeit at the cost of considerable performance degradation. In this article, our focus lies on filter pruning, an aspect of coarse-grained sparsity, with the objective of simultaneously preserving the performance of DNN models and enabling hardware acceleration.

Filter pruning

Filter pruning eliminates entire convolutional filters from the initial CNNs, yielding a structured pruned model (*Liu et al., 2017*; *Lin et al., 2020a*; *He et al., 2019*; *Lin et al., 2021*). Filter pruning aims to uphold the original model's performance while significantly decreasing both the FLOPs and parameter burden. It draws mounting attention for the practical compression effect as the compressed network receives adequate support from conventional hardware and readily accessible basic linear algebra subprograms (BLAS). The mainstram research on filter pruning bifurcates approximately into two distinct classes, as illustrated further.

The first group adheres to a three-step pruning pipeline, which encompasses pre-training the model, removing unimportant filters, and meticulously fine-tuning the resulting pruned model. Predominantly, most methodologies in this sphere tend to underscore the second stage, deploying a diverse array of filter importance estimation strategies. For instance, *Li et al. (2017)* elected to prune filters exhibiting smaller ℓ_1 norm values. *He et al. (2019)* considered the next layer's construction error as a pivotal criterion and proceeded with layer-by-layer pruning. *Wang et al. (2019)* employed attention modules to derive the scaling factor for each filter and leveraged it as a benchmark for determining filter importance. Despite their innate simplicity, these hand-crafted benchmarks for the quantification of filter importance frequently stumble in accurately eliminating filters integral to the comprehensive performance.

Contrarily, the secondary category catalyzes filter pruning by training the network, supplementing the sparse constraints on individual filters (*Huang & Wang, 2018*; *Luo & Wu, 2020*). As a result, the pruned model becomes available through the removal of zero-valued filters or those falling below a specified threshold. For instance, *Huang & Wang (2018)* presented a scaling factor to augment the output of a defined structure, thereby enhancing sparsity on such factors. *Luo & Wu (2020)* incorporated an "automatic pruner" layer into the convolutional layer to execute filter pruning autonomously. *Xiao, Wang & Rajasekaran (2019)* pruned CNN models utilizing gradient-based update rules. *Tang et al. (2021)* further delved into prevalent information to dynamically identify filter redundancy within CNNs. However, these methods routinely interfere with the standard training procedure of the network due to the penalties imposed on weight norms, leading to sub-optimal outcomes. Our proposed MLPruner aims to execute filter pruning during training without incurring any weight penalty, thus alleviating the restrictions imposed by the mentioned sets of pruning methods.

Besides, an emerging class of methods primarily focuses on the architecture of pruned networks, specifically the pruning rate of each layer (*Liu et al., 2019*; *He et al., 2018b*; *Lin et al., 2020c*). For instance, *He et al. (2018b)* proposed utilizing reinforcement learning to automate the search process. Similarly, *Liu et al. (2019)* pre-trained a PruningNet to predict the weights of potential networks and incorporated an evolutionary algorithm to scout for the finest candidate. In addition, several methodologies automatically identify optimal network architectures by designing differentiable operators (*Ning et al., 2020*; *Guo et al., 2020*; *Li et al., 2023*). For instance, DMCP models channel pruning as a differentiable Markov process to learn the significance of each layer. Our proposed MLPruner is orthogonal to these architecture search-based methods. To explain, MLPruner can apply a global pruning based on the learned masks to automatically determine the pruning rate for each layer or directly utilize the structure obtained from existing architecture search-based methods and perform local pruning from each layer.

Recent surveys (*Verma et al., 2024*) comprehensively analyze the evolution from static to dynamic neural architectures. *Gao et al. (2024)* proposed a unified dynamic and static channel pruning method based on two-layer optimization (BilevelPruning/UDSP), which evaluates the static subnetwork and jointly learns the channel selection of both from dynamic subnetwork to end-to-end, which retains the storage-saving advantage of static pruning and compensates for the performance constraints of the fixed subnetwork with the help of dynamic pruning, and at the same time, directly sets the parameter and computational constraints without manually fusing the functions. The parameters and computational constraints are set directly, eliminating the need for manual fusion functions. In parallel, some recent works attempt to unify pruning with architectural search or dynamic mask generation. *Wu et al. (2024)* proposed Auto-Train-Once (ATO), a single-shot framework that uses a controller network to dynamically generate pruning masks. ATO removes the need for fine-tuning and comes with convergence guarantees. Our MLPruner contributes to this direction by incorporating dynamic mask learning into training while maintaining a static, hardware-efficient architecture for deployment. Our MLPruner contributes to this evolving landscape by incorporating dynamic mask learning into the pruning process, while retaining a static and hardware-friendly deployment structure.

METHOD

Background

We first describe necessary preliminaries for CNNs filter pruning. Denote an L-layer CNN model's parameters as $\mathbf{W} = \{\mathbf{W}^1, \mathbf{W}^2, \mathbf{W}^3, \ldots, \mathbf{W}^L\}$, the convolution kernel weights of a distinct layer l can be represented as $\mathbf{W}^l \in \mathbb{R}^{C^l_{out} \times C^l_{in} \times K^l \times K^l}$, where C^l_{out}, C^l_{in}, K^l symbolize the number of output filters, input filters, and the kernel size, respectively. Let $I^l \in R^{B \times C^l_{in} \times H^l \times W^l}$ be the input feature map for layer l. Here, B acts as the batch size, H^l and W^l represent the dimensions of the input image. The output from layer l is predicated upon:

$$\mathcal{O}^l = \mathcal{I}^l \circledast \mathrm{W}^l, \tag{1}$$

where the convolution operator is represented as $\circledast$ [1]. Considering a batch of training image set X paired with class labels set Y, the training loss manifests as:

$$\min_{\mathbf{W}} \mathscr{L}(X, \mathbf{W}; Y), \tag{2}$$

where $\mathscr{L}(\cdot)$ operates as the loss function, usually taking the form of cross-entropy loss. For filter pruning, a subset of output filters in $\mathbf{W}^l$ is excised to yield the pruned kernel $\widehat{\mathbf{W}}^l \in \mathbb{R}^{\widehat{C}^l_{out} \times \widehat{C}^l_{in} \times K^l \times K^l}$, subject to $\widehat{C}^l_{out} \leq C^l_{out}$ and $\widehat{C}^l_{in} \leq C^l_{in}$ constraints. It is worthwhile to highlight that the associated input channels of $\mathbf{W}^{l+1}$ are also expelled. We can hence transpose Eqs. (1) and (2) for the pruned network as follows:

$$\widehat{\mathscr{O}}^l = \widehat{\mathscr{I}}^l \circledast \widehat{\mathrm{W}}^l, \tag{3}$$

$$\min_{\widehat{\mathbf{W}}} \mathscr{L}(X, \widehat{\mathbf{W}}; Y). \tag{4}$$

However, restoring the performance of pruned models, *i.e.*, minimizing Eq. (4), is challenging due to the elimination of pre-trained weights. Early research often utilized various criteria to remove insignificant filters, such as ℓ_1-norm (*Li et al., 2017*), geometric data (*He et al., 2019*), and activation sparsity (*Hu et al., 2016*). Recent research have developed training-aware pruning methods have been proposed to automatically identifying non-essential filters (*Ding et al., 2018*; *Zhang et al., 2022*). This class of methods typically achieves filter pruning by gradually penalizing some filter weights during the training process *via* the imposition of ℓ_2 regularization constraints on the original weights. However, such regularization constraint may affect the normal training objective in Eq. (4), thereby influencing the final performance of the pruned network.

[1] The bias and batch normalization layer are omitted for simplicity.

MLPruner

In this article, we present MLPruner to mitigate the aforementioned issue, with its core innovation falls into optimizing a learnable mask $\mathbf{M} = \{\mathbf{M}^1, \mathbf{M}^2, \mathbf{M}^3, \ldots, \mathbf{M}^L\}$ to automatically locate essential filters without weight penalty. The framework of MLPruner is illustrated in Fig. 1. In the following, we particularly introduce the optimization process for $\mathbf{M}$, as well as validate how the learned mask aptly reflects the corresponding filter's significance.

Mask forward. During forward propagation within a specific layer l, we leverage the mask $\mathbf{M}^l$ to obtain a binary indicator $\mathbf{B}^l \in \{0, 1\}^{C^l_{out} \times C^l_{in} \times K^l \times K^l}$ that guides the removal or preserve of corresponding filters. This is achieved using a pre-defined pruning rate p^l to round $\mathbf{M}^l$ to 0 or 1 s, where 0 indicate pruning corresponding weights, and vice versa. Formally, we first accumulate $\mathbf{M}^l$ in a filter-wise manner as:

$$\widehat{\mathbf{M}^l_i} = ||\mathbf{M}^l_{i,:,:,:}||_1, i = 1, 2, \ldots, C^l_{out}. \tag{5}$$

Then, the binary indicator is derived as

$$\mathbf{B}^l_{i,:,:,:} = \begin{cases} 0, & \text{if } \widehat{\mathbf{M}^l_i} \text{ in the top-}p^l \text{ smallest values of } \widehat{\mathbf{M}}^l, \\ 1, & \text{otherwise.} \end{cases} \tag{6}$$

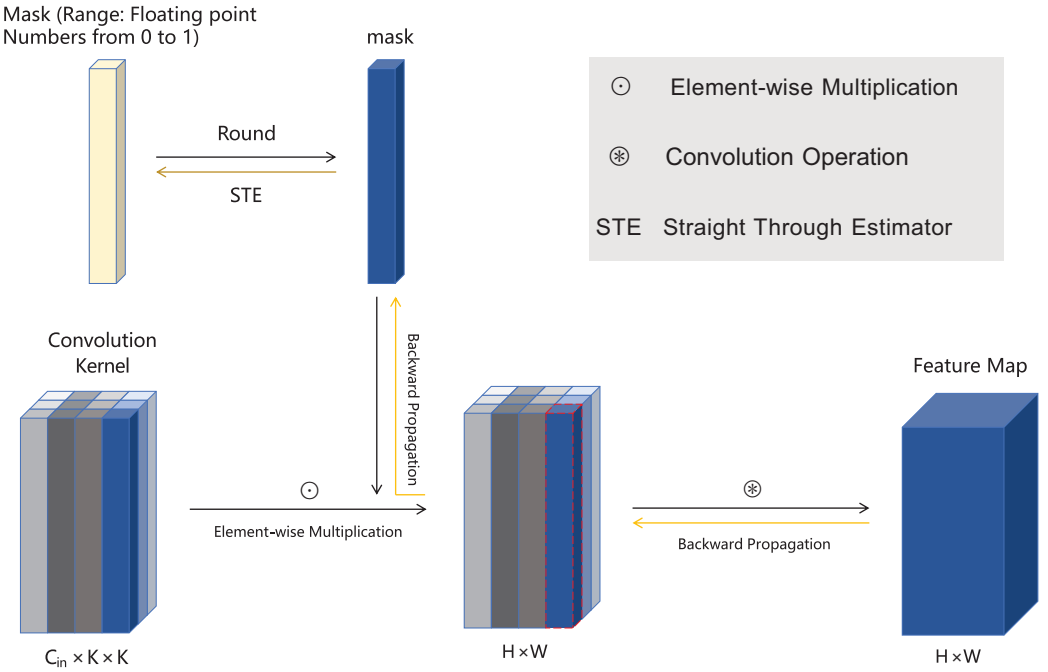

Figure 1 The framework of MLPrunner method. For better representation, we have chosen one of the channels for description. Our method assigns a learnable mask to the channel, performs a round operation on the mask using a pre-defined pruning rate, and multiplies it with the weight W to determine whether the channel needs to be pruned. Then, a feature map is generated through convolution operation to determine the model. (Note: Considering the non differentiable nature of the round function, which prevents the gradient accepted by mask from being updated in a timely manner, we have adopted STE for backpropagation to solve this problem). Full-size DOI: 10.7717/peerj-cs.3132/fig-1

Finally, we use the binary indicator to prune the dense weights during forward propagation as:

$$\mathcal{O}^l = \mathscr{L} \circledast (\mathrm{B}^l \odot \mathrm{W}^l), \tag{7}$$

where $\odot$ denotes the point-wise multiplication.

Mask backward. The above forward phase enables automatic filter pruning by looking at value of mask $\mathbf{M}^l$, yet Eq. (7) is non-differentiable due to the round operation, making the mask unable to be updated, *i.e.*, the gradient can not passed from $\mathbf{B}^l$ to $\mathbf{M}^l$. To address this, we use the straight-through estimator (STE) (*Bengio, Léonard & Courville, 2013*) as an alternative to approximate the mask gradient of $\mathbf{M}^l$ as:

$$\begin{aligned}\frac{\partial \mathscr{L}}{\partial \mathbf{M}^l} &= \frac{\partial \mathscr{L}}{\partial (\mathbf{B}^l \odot \mathbf{W}^l)} \frac{\partial (\mathbf{B}^l \odot \mathbf{W}^l)}{\partial \mathbf{B}^l} \frac{\mathbf{B}^l}{\mathbf{M}^l} \\ &\approx \frac{\partial \mathscr{L}}{\partial (\mathbf{B}^l \odot \mathbf{W}^l)} \frac{\partial (\mathbf{B}^l \odot \mathbf{W}^l)}{\partial \mathbf{B}^l} \cdot \mathbf{1} \,. \\ &= \frac{\partial \mathscr{L}}{\partial (\mathbf{B}^l \odot \mathbf{W}^l)} \mathbf{W}^l\end{aligned} \tag{8}$$

In this manner, the mask can be jointly optimized with the weight during training. Notably, such mask learning process brings no gradient distortion to the weights as traditional training-aware pruning methods do. Furthermore, such learned masks can well tell the importance of corresponding filters, which we analyze below.

Tractability of optimization. We take a specific filter $\mathbf{W}^l_{i,:,:,:}$ to show that our learned mask score can well reflect the relative importance of filter weights.

Omitting the update schedule including learning rate and momentum, we can represented the learned mask $\mathbf{M}^l_{i,:,:,:}$ as

$$\mathbf{M}^l_{i,:,:,:} \approx \sum_{t=1}^{T} \frac{\partial \mathscr{L}^t}{\partial (\mathbf{B}^l \odot \mathbf{W}^l)_{i,:,:,:}} \mathbf{W}^l_{i,:,:,:}, \tag{9}$$

where T denotes the total training iterations.

Now, let us turn our focus to the loss variation $\Delta\mathscr{L}(\mathbf{W}^l_{i,:,:,:})$ when removing $\mathbf{W}^l_{i,:,:,:}$. It can be derived using the Taylor expansion (*Molchanov et al., 2017*) as

$$\begin{aligned} \Delta\mathscr{L}(\mathbf{W}^l_{i,:,:,:}) &= \mathscr{L}(\mathbf{W}^l_{i,:,:,:} = 0) - \mathscr{L}(\mathbf{W}^l_{i,:,:,:}) \\ &\approx \mathscr{L}(\mathbf{W}^l_{i,:,:,:}) - \mathbf{W}^l_{i,:,:,:} \frac{\partial \mathscr{L}}{\partial (\mathbf{W}^l_{i,:,:,:})} \\ &\quad + R_1(\mathbf{W}^l_{i,:,:,:} = 0) - \mathscr{L}(\mathbf{W}^l_{i,:,:,:}) \\ &\approx -\mathbf{W}^l_{i,:,:,:} \frac{\partial \mathscr{L}}{\mathbf{W}^l_{i,:,:,:}} \end{aligned}, \tag{10}$$

which leads Eq. (9) to

$$\mathbf{M}^l_{i,:,:,:} = \sum_{t=1}^{T} \Delta\mathscr{L}(\mathbf{W}^l_{i,:,:,:}). \tag{11}$$

Here we can observe that the trained mask $\mathbf{M}^l_{i,:,:,:}$ is the accumulation of loss change for removing weights $\mathbf{W}^l_{i,:,:,:}$. A small score denotes less loss increase thus the corresponding candidate can be safely removed. Thus, our introduced mask learning can be used to measure the relative importance between convolution filters, and preserving filters with larger mask value can well decrease the loss after pruning. The learned masks directly correlate with filter importance through the accumulated loss gradient Eq. (9). Unlike weight-penalty methods that distort gradients *via* L2 regularization, MLPruner's masks provide unbiased importance estimates while preserving original weight optimization.

The overall pipeline of MLPruner is outlined in Algorithm 1. In particular, we set a mask training epochs $\mathscr{T}$ to learn the masks for filter pruning. After that, we stop optimizing masks and obtain the pruned weight $\widehat{\mathbf{W}}$ by extracting the corresponding filter from $\mathbf{W}$ according to the binary indicator $\mathbf{B}$. Finally, we fine-tune the pruned model to further recover its performance. It can be inferred that MLPruner does not bring any weight penalty during training, but automatically learns masks that can well reflect the importance of filters with respect to the training loss, achieving holistic filter pruning in an end-to-end manner. MLPruner fundamentally differs from weight-penalty approaches in

Algorithm 1 **Algorithm description of MLPruner.**

Input: An L-layer CNN with weight $\mathbf{W}$, mask training epochs $\mathscr{T}$
Output: The pruned model weight $\widehat{\mathbf{W}}$
// `Joint mask and weight training`
for $h = 1 \rightarrow \mathscr{T}$ **do**
  for *each training iteration* **do**
    Get binary indicator $\mathbf{B}$ *via* Eqs. (6) and (5);
    Forward propagation *via* Eq. (7);
    Backward propagation *via* Eq. (8);
  end
end
// `Obtain the pruned weights`
for $l = 1 \rightarrow l$ **do**
  $\widehat{\mathbf{W}^l}$ = squeeze($\mathbf{B}^l \odot \mathbf{W}^l$) // `squeeze () denotes extract non-zero filters`
end
Fine-tune $\widehat{\mathbf{W}}$ to recover performance.

three ways: (1) Our masks provide unbiased importance estimates without distorting weight gradients as $\mathbf{M}_i^l$ directly measures $\mathbf{W}_i^l$'s true contribution to loss minimization, uncontaminated by auxiliary penalties, (2) The pruning decision is based on accumulated loss impact rather than instantaneous magnitude, and (3) We maintain the original training objective without additional regularization terms, delivering better generalization ability as weight updates follow the original loss gradient $\nabla_{\mathbf{W}}\mathscr{L}$ (Eq. 10), maintaining the model's intrinsic generalization dynamics.

Remark. In our description of MLPruner, we take the layer-wise pruning rate p_l to be a known quantity. In practical scenarios, we can employ global sorting for the learned mask to acquire the binary indicator for each layer. In such case, the layerwise pruning rate is automatically derived, thereby showcasing the ease-of-use proffered by MLPruner. Furthermore, it is also worth noting that the existing structure-searching pruning methods (*Lin et al., 2020c*; *Liu et al., 2019*) hold orthogonality to MLPruner, which means that we could utilize the pruning rates identified from each layer to locally sort the mask and obtain the binary indicator. We manifest the comparative performance achieved through diverse layerwise pruning rates in the experiment section.

EXPERIMENT

Experimental settings

Datasets and networks. We conducted experiments on representative image classification datasets CIFAR-10, CIFAR-100 (*Krizhevsky & Hinton, 2009*), and ImageNet (*Deng et al., 2009*) to demonstrate the effectiveness of the proposed MLPruner. CIFAR-10 is comprised of 60,000 distinct 32×32 color images, spread evenly across 10 various classes, with each class housing 6,000 images. The CIFAR-100 dataset, designed parallel to CIFAR-10,

departs only in its increased class count, featuring 100 distinct classes, each containing 600 images. We elect representative convolutional networks to validate the efficacy of MLPruner, including VGGNet-16 (*Simonyan & Zisserman, 2015*), GoogLeNet (*Szegedy et al., 2015*), ResNet (*He et al., 2016*), and MobileNet-V2 (*Sandler et al., 2018*).

Implementation details. We set the mask training epochs $\mathscr{T} = 30$ for CIFAR-10/100 and $\mathscr{T} = 10$ for ImageNet, with all masks initialized to 1s. After training masks, we directly conduct global sort on the masks to obtain layer-wise pruning rates and then prune filters based on the learned masks. All our pruned models are fine-tuned employing the stochastic gradient descent (SGD) optimizer, paired with a momentum of 0.9 and a batch size of 256. On CIFAR-10, Each pruned network is fine-tuned for 300 epochs, incorporating a weight decay factor of 5×10^{-3} and an initial learning rate of 0.1. This rate undergoes a subsequent reduction to 0.01 and then to 0.001 after specific intervals of 50 and 100 epochs respectively. On ImageNet, we give 90 epochs for tine-tuning with weight decay of 1×10^{-4}. The learning rate is initialized to 0.1 and decayed by cosine annealing schedule. Besides, we employ random crop and horizontal flip to the input images. All experiments is implemented based on PyTorch and executed on NVIDIA 3090 GPUs.

Performance metrics and baselines. MLPruner is juxtaposed with several state-of-the-art pruning methods (*He, Zhang & Sun, 2017*; *Huang & Wang, 2018*; *Li et al., 2017*; *Lin et al., 2020a*, *2020c*, *2019*; *Yu et al., 2018*; *Zhao et al., 2019*). We report the top-1 accuracy to facilitate a quantitative comparison among various methods. The computing costs and storage demands are reflected in the number of FLOPs and parameters, accompanied by their corresponding pruning rates (PR), forming the basis for assessing the efficacy of our MLPruner and other methods used for comparison.

CIFAR-10

VGGNet-16. We first incorporate our MLPruner to prune the 16-layer VGGNet model, a classic CNN comprising 13 sequential convolutional layers and 3 fully-connected layers. Evident from Table 1, MLPruner markedly surpasses the current state-of-the-art methods across all performance indicators. Our MLPruner accomplishes a parameter compression of 89.1%, thereby enhancing the computation by 71.9% along with a negligible accuracy decrease of 0.03%. This substantial improvement proves instrumental in deploying the VGGNet model on resource-restricted devices.

GoogLeNet. We further demonstrate the effectiveness of MLPruner in pruning GoogLeNet, a prevailing network with inception structure. Table 2 substantiates that notwithstanding marginal drops in top-1 accuracy (94.78% for MLPruner *vs*. 95.03% for the baseline), MLPruner significantly curtails the FLOPs by 62.5% and parameters by 61.3%. When juxtaposed with the superior state-of-the-art, *i.e*., HRank, MLPruner not only surpasses in accuracy performance but also substantially reduces the quantity of FLOPs and parameters. Consequently, MLPruner excels in diminishing the superfluity of networks with intricate multi-branch structures.

Table 1 Pruning results of VGG-16 (*Simonyan & Zisserman, 2015*) on CIFAR-10.

Model	Top-1 (%)	FLOPs (PR)	Parameters (PR)
Baseline	93.02	314.04M (0.0%)	14.73M (0.0%)
SSS (*Huang & Wang, 2018*)	93.02	183.13M (41.6%)	3.93M (73.8%)
Zhao et al. (2019)	93.18	190.00M (39.1%)	3.92M (73.3%)
GAL-0.05 (*Lin et al., 2019*)	92.03	189.49M (39.6%)	3.36M (77.6%)
HRank (*Lin et al., 2020a*)	92.34	108.61M (65.3%)	2.64M (82.1%)
ABC (*Lin et al., 2020c*)	92.51	92.45M (70.6%)	1.75M (88.2%)
RGP (*Chen et al., 2023*)	92.76	78.8M (74.89%)	3.68M (75.00%)
MLPruner	92.99	87.98M (71.9%)	1.62M (89.1%)

Table 2 Pruning results of GoogLeNet (*Szegedy et al., 2015*) on CIFAR-10.

Model	Top-1 (%)	FLOPs (PR)	Parameters (PR)
Baseline	95.03	1.53B (0.0%)	6.17M (0.0%)
ℓ_1 (*Li et al., 2017*)	94.54	1.02B (32.9%)	3.51M (42.9%)
GAL-0.05 (*Lin et al., 2019*)	93.93	0.94B (38.2%)	3.12M (49.3%)
HRank (*Lin et al., 2020a*)	94.53	0.69B (54.9%)	2.74M (55.4%)
ABC (*Lin et al., 2020c*)	94.55	0.71B (53.9%)	2.89M (52.1%)
RGP (*Chen et al., 2023*)	94.61	0.54B (64.8%)	2.55M (58.7%)
MLPruner	94.78	0.59B (62.5%)	2.45M (61.3%)

ResNet. We also evaluate the network pruning performances of various methods on ResNet (*He et al., 2016*), a predominant deep CNN with residual modules, as shown in Table 3. Notably, our MLPruner enhances the original ResNet-56 performance by 0.05%, successfully eliminating approximately 54.8% of FLOPs. This efficiency contrasts sharply with other methods, which invariably experience an accuracy reduction to varying degrees, despite achieving fewer reductions in FLOPs. Furthermore, our MLPruner exhibits remarkable superiority when applied to ResNet-110. Despite a robust 65.8% reduction in FLOPs, it generates a performance improvement of 0.08%, significantly outpacing competing methods.

MobileNet. MobileNet-v2 (*Sandler et al., 2018*), an advancing network, exemplifies compact design *via* its depth-wise separable convolution. Given its minuscule computational cost, pruning MobileNet-v2 proves to be an exceptionally challenging task. Nevertheless, as illustrated in Table 4, in comparison with its competitors, MLPruner impressively maintains a superior top-1 accuracy of 95.31%, whilst achieving a more significant pruning of FLOPs by 31.5%. Therefore, the results underscore the notable preeminence of our MLPruner in CNN pruning, unflinchingly handling even highly compact network models.

CIFAR-100

We further demonstrate the efficacy of MLPruner for pruning VGGNet-19 on the more complex dataset CIFAR-100. Table 5 shows the comparative performance of MLPruner

Table 3 Pruning results of ResNet-56/110 (*He et al., 2016*) on CIFAR-10.

Model	Top-1(%)	FLOPs (PR)	Parameters (PR)
ResNet-56	93.26	126.56M (0.0%)	0.85M (0.0%)
ℓ_1 (*Li et al., 2017*)	93.06	90.90M (27.6%)	0.73M (14.1%)
GAL-0.6 (*Lin et al., 2019*)	92.90	78.30M (37.6%)	0.75M (11.8%)
FPGM (*He et al., 2019*)	93.26	59.40M (52.6%)	–
ABC (*Lin et al., 2021*)	93.19	73.36M (41.5%)	0.50M (41.2%)
LFPC (*He et al., 2020*)	93.24	59.10M (52.9%)	–
HRank (*Lin et al., 2020a*)	93.17	62.72M (50.0%)	0.49M (42.4%)
RGP (*Chen et al., 2023*)	92.92	57.99M(54.2%)	0.47M (44.8%)
MLPruner	93.31	57.23M (54.8%)	0.43M (49.5%)
ResNet-110	93.57	254.99M (0.0%)	1.73M (0.0%)
ℓ_1 (*Li et al., 2017*)	93.30	155.00M (38.7%)	1.16M (32.6%)
GAL-0.5 (*Lin et al., 2019*)	92.55	130.20M (48.5%)	0.95M (44.8%)
HRank (*Lin et al., 2020a*)	93.36	105.70M (58.2%)	0.70M (59.2%)
LFPC (*He et al., 2020*)	93.07	101.00M (60.3%)	–
ABC (*Lin et al., 2020c*)	93.44	92.84M (63.3%)	0.69M (59.9%)
RGP (*Chen et al., 2023*)	93.51	91.59M (64.1%)	0.63M (63.7%)
MLPruner	93.65	87.11M (65.8%)	0.61M (64.8%)

Table 4 Pruning results of MobileNet-v2 (*Sandler et al., 2018*) on CIFAR-10.

Model	Top-1 (%)	FLOPs (PR)	Parameters (PR)
Baseline	94.47	98.05M (0.0%)	2.29M (0.0%)
WM (*Howard et al., 2017*)	94.02	71.57M (27.0%)	1.81M (20.9%)
DCP (*Zhuang et al., 2018*)	94.69	71.57M (27.0%)	1.81M (20.9%)
MDP (*Guo, Ouyang & Xu, 2020*)	95.14	69.90M (28.7%)	–
White-Box (*Zhang et al., 2022*)	95.28	69.41M (29.2%)	1.77M (22.3%)
MLPruner	95.31	67.16M (31.5%)	1.56M (23.4%)

under analogous pruning rates *vs.* other methodologies. Observable from these comparisons, MLPruner outperforms the state-of-the-art (SOTA) alternatives across varying pruning rates. Emphasizing further, MLPruner successfully minimizes the FLOPs to a fraction of 70.7%, simultaneously achieving a top-1 accuracy of 73.45%. Conversely, the recent SOTA, HRank (*Lin et al., 2020a*), demands greater computation to yield subpar top-1 accuracy at 72.91%. Undeniably, these results accentuate the distinct advantage of employing our MLPruner in pruning CNNs on more challenging datasets.

ImageNet

In the challenging ImageNet dataset, we compared MLPruner with current state-of-the-art pruning technologies. Key performance indicators, as observed in Table 6, reveal that MLPruner demonstrates a significant advantage in many core metrics. Firstly, MLPruner achieved a Top-1 accuracy of 75.62%, which not only surpasses most of the listed methods

Table 5 Pruning results of VGGNet-19 (*Szegedy et al., 2015*) on CIFAR-100.

Model	Top-1 (%)	FLOPs (PR)	Parameters (PR)
Baseline	73.43	399.12M (0.0%)	20.04M (0.0%)
ℓ_1 (*Li et al., 2017*)	72.52	171.51M (42.9%)	5.17M (74.2%)
HRank (*Lin et al., 2020a*)	72.91	154.06M (61.4%)	4.08M (80.2%)
ABC (*Lin et al., 2020c*)	73.04	145.67M (63.5%)	3.54M (82.3%)
White-Box (*Zhang et al., 2022*)	73.33	127.31M (68.1%)	2.38M (86.4%)
MLPruner	73.45	116.94M (70.7%)	2.18M (88.1%)

Table 6 Pruning results of ResNet-50 (*He et al., 2016*) on ImageNet. † means we follow EagleEye to start from a pre-trained model and give 120 epochs for fine-tuning.

Model	Top-1 (%)	FLOPs (PR)	Parameters (PR)
Baseline	76.15	4.10G (0%)	25.56M (0%)
ThiNet (*Luo, Wu & Lin, 2017*)	72.04	1.62G (60.2%)	14.55M (43.0%)
SFP (*He et al., 2018a*)	74.61	2.38G (58.2%)	–
GAL (*Lin et al., 2019*)	71.95	2.33G (43.0%)	21.25M (16.9%)
HRank (*Lin et al., 2020a*)	74.98	2.29G (44.1%)	16.17M (36.7%)
White-Box (*Zhang et al., 2022*)	75.32	2.23G (45.5%)	16.49M (35.5%)
DSA (*Ning et al., 2020*)	75.10	2.46G (40.0%)	–
CLR-RNF (*Lin et al., 2022*)	74.85	2.44G (40.3%)	16.91M (33.8%)
MLPruner	75.62	2.21G (46.0%)	14.47M (43.4%)
EagleEye (*Li et al., 2020*)	76.40	2.00G (51.2%)	–
Pas (*Li et al., 2022*)	76.70	2.00G (51.2%)	–
MLPruner†	76.89	1.97G (51.9%)	13.98M (45.6%)

such as White-Box, DSA, and CLR-RNF—with accuracies of 75.32%, 75.10%, and 74.85%, respectively—but also shows considerable improvements over ThiNet at 72.04% and GAL at 71.95% Top-1 accuracy. This enhancement is particularly crucial as it demonstrates the capability to balance the reduction of computational resources while maintaining model performance during the pruning process. Further, MLPruner outperforms EagleEye by 1.2% accuracy at similar FLOPs because: (1) Our mask learning adapts to training dynamics rather than using fixed heuristics, (2) Joint optimization prevents the greedy local decisions EagleEye makes, and (3) The gradient-based importance measure better preserves critical filters.

Moreover, in terms of computational efficiency, MLPruner's performance is equally impressive. It reduced the total number of floating-point operations (FLOPs) by 46.0%, achieving only 2.21G. This figure is not only lower than the baseline model's 4.10G but also slightly better compared to other methods like DSA's 2.46G and CLR-RNF's 2.44G, indicating that MLPruner effectively lessens the computational load while preserving a relatively high accuracy. To sum up, Compared to other pruning methods, MLPruner not only reduces parameters more efficiently but also maintains better performance, even on more challenging dataset.

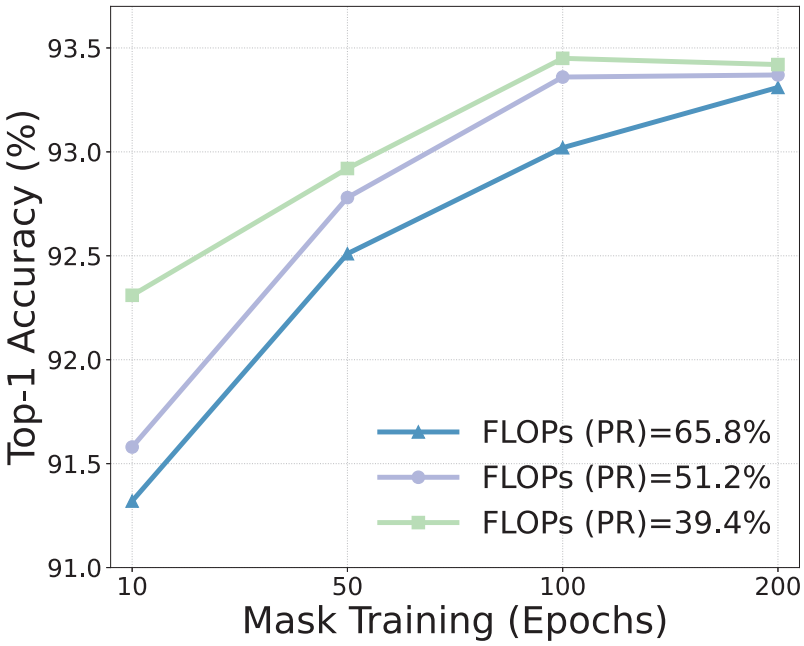

Figure 2 **Effect of the mask learning epochs within MLPruner.**
Full-size DOI: 10.7717/peerj-cs.3132/fig-2

We conclude with a discussion on the pruning efficiency. Traditional methods, such as ThiNet and HRank, typically determine layer-wise pruning rates through manual design, which incurs significant human effort. On the other hand, search-based approaches, such as EagleEye and ABCPruner, often employ evolutionary algorithms or specific evaluation metrics to explore optimal architectures, which still demand substantial computational time. In contrast, our proposed MLPruner achieves automatic mask training with only one-tenth of the fine-tuning cost, enabling end-to-end optimization of both layer-wise pruning rates and critical filter identification, thereby enhancing pruning efficiency compared to existing methods.

Performance analysis

In this section, we prune ResNet-56 and test its performance on CIFAR-10 as an example to analysis the performance of MLPruner *w.r.t.* its different components.

Mask learning epochs. Initially, we scrutinize the import of learning epochs $\mathscr{T}$ for masks. Figure 2 delineates the performance nuances when utilizing varying epochs at diverse pruning ratios. Manifestly, enhancing the epochs of mask learning significantly augments the model's efficacy, particularly under conditions of high sparsity. This phenomenon is intuitively understandable as a higher sparsity demands the removal of multiple filters, thereby necessitating an amplified number of mask training epochs to discern the priority between diverse filters. Concurrently, it should be noted that the performance gains of increasing epochs possess a ceiling effect, and an abundance of epochs can incur inflated training overhead. Therefore, we configure the mask training for 30 epochs.

Table 7 Top-1 accuracy comparison for pruning ResNet-56 on CIFAR-10 under different layerwise pruning rate configuration. 'Manual', 'ABCPruner', 'Global' means we use manually-defined (*Lin et al., 2020a*), searched results by ABCPruner (*Lin et al., 2020c*), and directly conduct global sorting for the masks, respectively.

Setting	Top-1 accuracy(%)	FLOPs(PR)
Manual	93.02	53.3%
ABCPruner	93.49	41.2%
Global	93.31	54.8%

Table 8 Top-1 accuracy comparison for pruning ResNet-56 on CIFAR-10 with or without weight penalty.

Setting	Top-1 accuracy(%)	FLOPs(PR)
w. Penalty	92.89	53.9%
w.o. Penalty (MLPruner)	93.31	54.8%

Layerwise pruning rate. We further investigated the impact of layerwise pruning rate on MLPruner's performance. Table 7 exemplifies the effect on FLOPs reduction and overall performance when employing different strategies to distribute the layerwise pruning rate. Interestingly, direct global ranking based on the learned mask can even yield better results than a manually set procedure, which underscores MLPruner's efficiency. Furthermore, the use of a searched layerwise pruning rate can lead to enhanced performance, albeit at increased costs. This demonstrates MLPruner's scalability, remaining orthogonal to the pruning methods predicated on architecture search.

Weight penalty. At last, we examine the impact of weight penalties on mask training within MLPruner, a widely adopted approach in training-aware filter pruning methods (*Huang & Wang, 2018*; *Xiao, Wang & Rajasekaran, 2019*). Specifically, we imposed an L1 penalty on the mask during its training course, with results displayed in Table 8. As can be seen, the introduction of a sparsity regularization term resulted in performance degradation. This can be attributed to the fact that our mask training strategy already accurately identifies critical filters, and the penalty item inevitably hinders the optimization of the mask.

FUTURE WORK

In this section, we discuss some future works that warrant exploration. First, the current mask learning process in MLPruner relies on a fixed threshold for binarization, which may not optimally adapt to varying layer-wise pruning sensitivities. Future work could explore dynamic or learnable threshold mechanisms to enhance flexibility. Second, the method primarily focuses on filter pruning for CNNs, leaving room for extension to other architectures (*e.g.*, transformers), particularly considering the structural differences in attention mechanisms and positional encodings. Third, the increased computational overhead during mask training requires optimization for larger-scale models. Fourth, while demonstrated on classification tasks, applicability to dense prediction tasks

(*e.g.*, segmentation) needs further validation. Lastly, the theoretical connection between mask values and filter importance could be further formalized to guide more principled pruning decisions. Addressing these limitations could broaden the applicability and robustness of the approach.

CONCLUSION

Filter pruning emerges as a research hotspot in both industrial and academic realms, intended to minimize the deployment and inference overhead of CNNs. In this article, we introduce a novel MLPruner methodology, which autonomously learns mask through threshold-based rounding forward and STE-based backward. The learned masks by MLPruner has been demonstrated to reflect a corresponding filter's significance, even in the absence of weight penalty that previous methods impose. Extensive experiments highlights the effectiveness of MLPruner in pruning popular CNN architectures as compared to a series of state-of-the-art filter pruning methods.

ACKNOWLEDGEMENTS

The authors thank all participants.

ADDITIONAL INFORMATION AND DECLARATIONS

Funding

This work is supported by the National Natural Science Foundation of China (81460279, 61841204); the Inner Mongolia Natural Science Foundation (2024LHMS08045, 2018LH08066); the Inner Mongolia Autonomous Region High School Science Research Foundation (NJZY145). The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Grant Disclosures

The following grant information was disclosed by the authors:
National Natural Science Foundation: 81460279, 61841204.
Inner Mongolia Natural Science Foundation: 2024LHMS08045, 2018LH08066.
Inner Mongolia Autonomous Region High School Science Research Foundation: NJZY145.

Competing Interests

The authors declare that they have no competing interests.

Author Contributions

- Sihan Chen conceived and designed the experiments, performed the experiments, analyzed the data, performed the computation work, prepared figures and/or tables, authored or reviewed drafts of the article, and approved the final draft.
- Ying Zhao conceived and designed the experiments, prepared figures and/or tables, authored or reviewed drafts of the article, and approved the final draft.

Data Availability

The following information was supplied regarding data availability:

Code and raw data are available in the Supplemental Files.

The CIFAR-10 dataset is available at the University of Toronto: https://www.cs.toronto.edu/~kriz/cifar.html.

The CIFAR-100 dataset is available at: https://www.image-net.org/download.php.

Supplemental Information

Supplemental information for this article can be found online at http://dx.doi.org/10.7717/peerj-cs.3132#supplemental-information.

REFERENCES

Bengio Y, Léonard N, Courville A. 2013. Estimating or propagating gradients through stochastic neurons for conditional computation. ArXiv DOI 10.48550/arXiv.1308.3432.

Chen Z, Xiang J, Lu Y, Xuan Q, Wang Z, Chen G, Yang X. 2023. RGP: neural network pruning through regular graph with edges swapping. *IEEE Transactions on Neural Networks and Learning Systems* **35(10)**:14671–14683 DOI 10.1109/tnnls.2023.3280899.

Deng J, Dong W, Socher R, Li L-J, Li K, Fei-Fei L. 2009. ImageNet: a large-scale hierarchical image database. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Piscataway: IEEE, 248–255.

Dettmers T, Zettlemoyer L. 2019. Sparse networks from scratch: faster training without losing performance. In: *Advances in Neural Information Processing Systems (NeurIPS)*.

Ding X, Ding G, Han J, Tang S. 2018. Auto-balanced filter pruning for efficient convolutional neural networks. In: *AAAI Conference on Artificial Intelligence (AAAI)*, Vol. 32.

Evci U, Gale T, Menick J, Castro PS, Elsen E. 2020. Rigging the lottery: making all tickets winners. In: *International Conference on Machine Learning (ICML)*, 2943–2952.

Gao S, Zhang Y, Huang F, Huang H. 2024. BilevelPruning: unified dynamic and static channel pruning for convolutional neural networks. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. Piscataway: IEEE, 16090–16100.

Girshick R, Donahue J, Darrell T, Malik J. 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. In: *IEEE International Conference on Computer Vision (ICCV)*. Piscataway: IEEE, 580–587.

Guo J, Ouyang W, Xu D. 2020. Multi-dimensional pruning: a unified framework for model compression. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Piscataway: IEEE, 1508–1517.

Guo S, Wang Y, Li Q, Yan J. 2020. DMCP: differentiable markov channel pruning for neural networks. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. Piscataway: IEEE, 1539–1547.

Han S, Pool J, Tran J, Dally W. 2015. Learning both weights and connections for efficient neural network. *Advances in Neural Information Processing Systems* **28**:1135–1143.

Hayashi K, Yamaguchi T, Sugawara Y, Maeda S-I. 2019. Exploring unexplored tensor network decompositions for convolutional neural networks. *Advances in Neural Information Processing Systems* **32**:5552–5562.

He Y, Ding Y, Liu P, Zhu L, Zhang H, Yang Y. 2020. Learning filter pruning criteria for deep convolutional neural networks acceleration. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Piscataway: IEEE, 2009–2018.

He K, Gkioxari G, Dollar P, Girshick R. 2017. Mask R-CNN. In: *IEEE International Conference on Computer Vision (ICCV)*. Piscataway: IEEE.

He Y, Kang G, Dong X, Fu Y, Yang Y. 2018a. Soft filter pruning for accelerating deep convolutional neural networks. In: *International Joint Conference on Artificial Intelligence (IJCAI)*, 2234–2240.

He Y, Lin J, Liu Z, Wang H, Li L-J, Han S. 2018b. AMC: automl for model compression and acceleration on mobile devices. In: *Proceeding European Conference on Computer Vision*, 784–800.

He Y, Liu P, Wang Z, Hu Z, Yang Y. 2019. Filter pruning via geometric median for deep convolutional neural networks acceleration. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Piscataway: IEEE, 4340–4349.

He K, Zhang X, Ren S, Sun J. 2016. Deep residual learning for image recognition. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Piscataway: IEEE, 770–778.

He Y, Zhang X, Sun J. 2017. Channel pruning for accelerating very deep neural networks. In: *IEEE International Conference on Computer Vision (ICCV)*. Piscataway: IEEE, 1389–1397.

Hoefler T, Alistarh D, Ben-Nun T, Dryden N, Peste A. 2021. Sparsity in deep learning: pruning and growth for efficient inference and training in neural networks. *Journal of Machine Learning Research* **22**:1–124 DOI 10.48550/arXiv.2102.00554.

Howard AG, Zhu M, Chen B, Kalenichenko D, Wang W, Weyand T, Andreetto M, Adam H. 2017. Mobilenets: efficient convolutional neural networks for mobile vision applications. ArXiv DOI 10.48550/arXiv.1704.04861.

Hu H, Peng R, Tai Y-W, Tang C-K. 2016. Network trimming: a data-driven neuron pruning approach towards efficient deep architectures. ArXiv DOI 10.48550/arXiv.1607.03250.

Huang Z, Wang N. 2018. Data-driven sparse structure selection for deep neural networks. In: *European Conference on Computer Vision*, 304–320.

Hubara I, Courbariaux M, Soudry D, El-Yaniv R, Bengio Y. 2016. Binarized neural networks. *Advances in Neural Information Processing Systems* **29**.

Jangra M, Dhull SK, Singh KK, Singh A, Cheng X. 2023. O-WCNN: an optimized integration of spatial and spectral feature map for arrhythmia classification. *Complex & Intelligent Systems* **9(3)**:2685–2698 DOI 10.1007/s40747-021-00371-4.

Ji Y, Liang L, Deng L, Zhang Y, Zhang Y, Xie Y. 2018. TETRIS: tile-matching the tremendous irregular sparsity. *Advances in Neural Information Processing Systems* **31**.

Krizhevsky A, Hinton G. 2009. Learning multiple layers of features from tiny images. Technical report, University of Toronto.

LeCun Y, Denker J, Solla S. 1989. Optimal brain damage. *Advances in Neural Information Processing Systems* **2**:598–605.

Li H, Kadav A, Durdanovic I, Samet H, Graf HP. 2017. Pruning filters for efficient convnets. In: *Proceeding International Conference on Learning Representations*.

Li Y, van Gemert JC, Hoefler T, Moons B, Eleftheriou E, Verhoef B-E. 2023. Differentiable transportation pruning. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. Piscataway: IEEE, 16957–16967.

Li B, Wu B, Su J, Wang G. 2020. EaglEeye: fast sub-net evaluation for efficient neural network pruning. In: *Proceeding European Conference on Computer Vision*, 639–654.

Li Y, Zhao P, Yuan G, Lin X, Wang Y, Chen X. 2022. Pruning-as-search: efficient neural architecture search via channel pruning and structural reparameterization. In: *International Joint Conference on Artificial Intelligence (IJCAI)*.

Lin M, Cao L, Li S, Ye Q, Tian Y, Liu J, Tian Q, Ji R. 2021. Filter sketch for network pruning. *IEEE Transactions on Neural Networks and Learning Systems* **33(12)**:7091–7100 DOI 10.1109/tnnls.2021.3084206.

Lin M, Cao L, Zhang Y, Shao L, Lin C-W, Ji R. 2022. Pruning networks with cross-layer ranking & k-reciprocal nearest filters. *IEEE Transactions on Neural Networks and Learning Systems* **34(11)**:9139–9148 DOI 10.1109/TNNLS.2022.3156047.

Lin S, Ji R, Chen C, Tao D, Luo J. 2018. Holistic CNN compression via low-rank decomposition with knowledge transfer. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **41(12)**:2889–2905 DOI 10.1109/tpami.2018.2873305.

Lin M, Ji R, Wang Y, Zhang Y, Zhang B, Tian Y, Shao L. 2020a. Hrank: filter pruning using high-rank feature map. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Piscataway: IEEE, 1529–1538.

Lin M, Ji R, Xu Z, Zhang B, Wang Y, Wu Y, Huang F, Lin C-W. 2020b. Rotated binary neural network. In: *Proceeding Advances in Neural Information Processing Systems*, 7474–7485.

Lin S, Ji R, Yan C, Zhang B, Cao L, Ye Q, Huang F, Doermann D. 2019. Towards optimal structured CNN pruning via generative adversarial learning. In: *Proceeding Advances in Neural Information Processing Systems*, 2790–2799.

Lin M, Ji R, Zhang Y, Zhang B, Wu Y, Tian Y. 2020c. Channel pruning via automatic structure search. In: *International Joint Conference on Artificial Intelligence (IJCAI)*, 673–679.

Liu S, Chen T, Chen X, Atashgahi Z, Yin L, Kou H, Shen L, Pechenizkiy M, Wang Z, Mocanu DC. 2021. Sparse training via boosting pruning plasticity with neuroregeneration. *Advances in Neural Information Processing Systems* **34**:9908–9922.

Liu Z, Li J, Shen Z, Huang G, Yan S, Zhang C. 2017. Learning efficient convolutional networks through network slimming. In: *Proceedings of the IEEE International Conference on Computer Vision*. Piscataway: IEEE, 2736–2744.

Liu Z, Mu H, Zhang X, Guo Z, Yang X, Cheng TK-T, Sun J. 2019. MetaPruning: meta learning for automatic neural network channel pruning. In: *IEEE International Conference on Computer Vision (ICCV)*. Piscataway: IEEE, 3296–3305.

Liu Z, Wu B, Luo W, Yang X, Liu W, Cheng K-T. 2018. Bi-Real Net: enhancing the performance of 1-bit CNNs with improved representational capability and advanced training algorithm. In: *Proceedings of the European Conference on Computer Vision (ECCV)*, 722–737.

Luo J-H, Wu J. 2020. AutoPruner: an end-to-end trainable filter pruning method for efficient deep model inference. *Pattern Recognition (PR)* **107(10)**:107461 DOI 10.1016/j.patcog.2020.107461.

Luo J-H, Wu J, Lin W. 2017. ThiNet: a filter level pruning method for deep neural network compression. In: *IEEE International Conference on Computer Vision (ICCV)*. Piscataway: IEEE, 5058–5066.

Meng F, Cheng H, Li K, Luo H, Guo X, Lu G, Sun X. 2020. Pruning filter in filter. *Advances in Neural Information Processing Systems* **33**:17629–17640.

Molchanov P, Tyree S, Karras T, Aila T, Kautz J. 2017. Pruning convolutional neural networks for resource efficient inference. In: *International Conference on Learning Representations (ICLR)*.

Ning X, Zhao T, Li W, Lei P, Wang Y, Yang H. 2020. DSA: more efficient budgeted pruning via differentiable sparsity allocation. In: *European Conference on Computer Vision (ECCV)*. Cham: Springer, 592–607.

Sandler M, Howard A, Zhu M, Zhmoginov A, Chen L-C. 2018. MobileNetV2: inverted residuals and linear bottlenecks. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Piscataway: IEEE, 4510–4520.

Simonyan K, Zisserman A. 2015. Very deep convolutional networks for large-scale image recognition. In: *International Conference on Learning Representations (ICLR)*.

Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, Erhan D, Vanhoucke V, Rabinovich A. 2015. Going deeper with convolutions. In: *Proceeding IEEE Conference on Computer Vision and Pattern Recognition*. Piscataway: IEEE.

Tang Y, Wang Y, Xu Y, Deng Y, Xu C, Tao D, Xu C. 2021. Manifold regularized dynamic network pruning. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. Piscataway: IEEE, 5018–5028.

Verma PR, Singh NP, Pantola D, Cheng X. 2024. Neural network developments: a detailed survey from static to dynamic models. *Computers and Electrical Engineering* **120(1)**:109710 DOI 10.1016/j.compeleceng.2024.109710.

Wang Z. 2020. SparseRT: accelerating unstructured sparsity on GPUs for deep learning inference. In: *Proceedings of the ACM International Conference on Parallel Architectures and Compilation Techniques (ICPACT)*, 31–42.

Wang W, Fu C, Guo J, Cai D, He X. 2019. COP: customized deep model compression via regularized correlation-based filter-level pruning. In: *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI-19)*, 3785–3791.

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. ArXiv DOI 10.48550/arXiv.2403.14729.

Xiao X, Wang Z, Rajasekaran S. 2019. Autoprune: automatic network pruning by regularizing auxiliary parameters. In: *Proceeding Advances in Neural Information Processing Systems*, 13681–13691.

Yu R, Li A, Chen C-F, Lai J-H, Morariu VI, Han X, Gao M, Lin C-Y, Davis LS. 2018. Nisp: Pruning networks using neuron importance score propagation. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Piscataway: IEEE, 9194–9203.

Zhang Y, Lin M, Lin C-W, Chen J, Wu Y, Tian Y, Ji R. 2022. Carrying out CNN channel pruning in a white box. *IEEE Transactions on Neural Networks and Learning Systems (TNNLS)* **34(10)**:7946–7955 DOI 10.1109/tnnls.2022.3147269.

Zhang X, Zhou X, Lin M, Sun J. 2018. ShuffleNet: an extremely efficient convolutional neural network for mobile devices. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Piscataway: IEEE, 6848–6856.

Zhao C, Ni B, Zhang J, Zhao Q, Zhang W, Tian Q. 2019. Variational convolutional neural network pruning. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Piscataway: IEEE, 2780–2789.

Zhuang Z, Tan M, Zhuang B, Liu J, Guo Y, Wu Q, Huang J, Zhu J. 2018. Discrimination-aware channel pruning for deep neural networks. In: *Proceeding Advances in Neural Information Processing Systems*, 875–886.