The Effect of Pruning Order: A Case Study on CNN Filter Pruning

Lorenzo Nikiforos
DET, Politecnico di Torino
Turin, Italy
lorenzo.nikiforos@polito.it

Luciano Prono
DET, Politecnico di Torino
Turin, Italy
luciano.prono@polito.it

Fabio Pareschi
DET, Politecnico di Torino
Turin, Italy
fabio.pareschi@polito.it

Gianluca Setti
DET, Politecnico di Torino
Turin, Italy
gianluca.setti@polito.it

***Abstract*—Progressive structured pruning is a widely adopted strategy for compressing convolutional neural networks, where model capacity is gradually reduced through multiple prune–fine-tune stages. While most existing methods focus on defining effective importance criteria for convolutional filters, the order in which they are removed is typically fixed and the effect of pruning order is largely unexplored. In this work, we investigate different pruning order strategies for structured pruning of convolutional filters. Specifically, we introduce two alternative scheduling policies, tail-based and center-based, that modify the order of filter removal while preserving the same final pruning budget. Unlike classic approaches, which remove the least important filters first, the proposed scheduling strategies modify the pruning order of the components. In this work, we demonstrate the significant impact of pruning scheduling through extensive experiments with ResNet and ResNeXt architectures on CIFAR-100 and ImageNet-1K datasets. In particular, center-based scheduling consistently outperforms the standard progressive strategy, with larger gains at higher pruning rates and on more challenging datasets. Additional analyses show that these improvements are robust with a varying number of pruning stages and importance criteria. These results highlight how changing pruning scheduling is a simple yet effective method to improve structured pruning performance.**

***Index Terms*—convolutional neural networks, structured pruning, filter pruning, pruning order, iterative pruning, model compression.**

I. Introduction

In recent years, deep neural networks have achieved remarkable success across a wide range of application domains, including computer vision [1], autonomous driving [2], natural language processing [3], and other fields. In many scenarios, models must operate under limited hardware resources while still providing reliable performance, making efficiency-aware network design and optimization a critical requirement. Within this context, Convolutional Neural Networks (CNNs) are highly relevant for visual resource-constrained applications.

To address this challenge, extensive research has focused on model compression and acceleration techniques aimed at reducing the complexity of deep neural networks while preserving their predictive performance. Approaches include quantization [4], low-rank approximation [5], knowledge distillation [6], and network pruning [7]. In particular, network pruning consists in identifying and eliminating redundant or less informative components of a network. In CNNs, filter-level pruning is particularly effective, as it preserves the regular network structure and can be directly executed using existing hardware and deep learning frameworks.

Many pruning approaches adopt a progressive pruning strategy, i.e., the amount of removed components is gradually increased during the training process [8]–[11]. The underlying motivation of this approach is to allow the network to progressively adapt to the reduction in capacity, avoiding abrupt structural changes that may severely degrade performance. Despite their effectiveness, most progressive pruning methods rely on an implicit and fixed pruning scheduling, where parameters are removed according to a monotonically increasing importance criterion. In practice, less important components are always pruned at early stages, while the increasingly important ones are progressively removed as the pruning rate increases. Such an assumption constrains the pruning process to a specific removal order, which not necessarily results in the optimal configuration.

To better elaborate on this, let us consider a CNN layer and divide its filters in four different groups based on their significance, namely the low, medium, high and most important ones. The most important filters are the ones to survive at the end of the progressive pruning process. Standard approaches remove the low importance group first, then fine-tune the CNN model. Since the removed filters were not important, the model easily recovers the lost performance. However, recovery relies on the update of the remaining filters, meaning that the medium, high and most important filters become even more significant. At this point, when medium and then high importance filters are removed, the damage to the CNN is increasingly high and difficult to recover.

On the contrary, let us start the progressive pruning operation by removing the high importance filters. In this case, the recovery effort is sustained by the most important filters, while low and medium importance filters marginally increase their significance. This means that removing the medium and then the low importance filter does not dramatically damage the performance of the model.

Nonetheless, the removal of the high importance filters right at the beginning of the process may disrupt too much the model performance, nullifying the benefit of gradual pruning. A middle ground solution is the removal of the medium importance filters first, then proceeding with the gradual removal of both increasingly and decreasingly important filters.

These observations highlight the limitations of the current methods, which primarily focus on what to prune rather than how pruning should be scheduled during training. Motivated by this, we investigate the role of pruning order in CNN filter pruning, analyzing how different scheduling strategies affect the final performance.

The remainder of the paper is structured as follows. Section II reviews related work on relevant pruning techniques. Section III presents in detail the investigated pruning scheduling strategies. Section IV reports the experimental setup, while Section V illustrates and analyzes the results. Finally the conclusion is drawn. *Reproducibility:* code is available at https://github.com/Foros15/pruning_scheduling.

II. Related work

A. Incremental Pruning

Iterative pruning is the most popular approach for incremental pruning, with most methods relying on repeated prune–fine-tune cycles to progressively remove convolutional filters. Specifically, many works [12]–[15] iteratively remove a fixed fraction of filters at each stage and retrain the network to recover accuracy.

Building on this, several studies [16] progressively increase sparsity during training, showing improved optimization stability. Regarding structured approaches, [17] allows filters to be temporarily zeroed while remaining trainable, effectively enabling a smooth increase of the pruning rate across stages. This idea is further formalized in [18] through an explicit pruning-rate scheduling strategy that asymptotically reaches a target compression level.

However, several works observe that removing certain parameters too early can be suboptimal. Dynamic pruning approaches [19], [20] address this by allowing previously pruned parameters to be reintroduced, motivated by the fact that connections deemed unimportant at early stages may become significant later. Although these methods were initially developed for unstructured sparsity, similar effects have been reproduced in structured convolutional networks. Specifically, authors in [21] show how channel importance evolves throughout training, while in [22] filters are suppressed rather than removed, implicitly accounting for the uncertainty of pruning decisions.

B. Pruning scores

Filter pruning relies on importance scores to rank and remove filters in convolutional layers. Common approaches use magnitude-based scores [12], [17], which depend only on the filter weights, gradient-aware scores [23]–[25], that use the training loss function to estimate the impact of filter removal, and activation-based scores [26]–[28], which measure the influence of filters on forward activations. Other criteria are based on batch-normalization scaling factors, attention maps, or hybrid combinations of these [29].

In this work, we focus on three representative scoring systems. First, the magnitude ℓ_1 score [12] measures filter importance as the sum of the absolute values of the weights as

$$s_{\mathrm{m}}(\boldsymbol{K}) = \|\boldsymbol{K}\|_1, \tag{1}$$

where $\boldsymbol{K}$ denotes the filter tensor, containing all its weights, and $\|\cdot\|_1$ is the ℓ_1-norm of the flattened tensor.

Second, we use the mean activation contribution of a convolutional filter over a dataset [30], computed as

$$s_{\mathrm{a}}(\boldsymbol{K}) = \mathbb{E}_{\boldsymbol{X}}\left[\frac{1}{HW}\|\boldsymbol{K} * \boldsymbol{X}\|_1\right], \tag{2}$$

where $\boldsymbol{X}$ is the input feature map, and H and W are the height and width of the output of the convolution operation.

Finally, we use the first-order Taylor score [31] to estimate the variation that the removal of the filter produces in the training loss as

$$s_{\mathrm{t}}(\boldsymbol{K}) = \mathbb{E}_{(\boldsymbol{X},y)}\left[\left\|\boldsymbol{K} \odot \nabla_{\boldsymbol{K}}\mathcal{L}(\boldsymbol{X},y)\right\|_1\right], \tag{3}$$

where y is the target classification label, $\mathcal{L}$ is the training loss, $\odot$ denotes element-wise multiplication. These three criteria span simple weight-based, activation-dependent, and gradient-aware perspectives.

III. Proposed Approach

In this section, we formalize the progressive filter pruning framework and introduce different pruning scheduling policies (tailed-based and center-based).

We consider a CNN layer that we prune progressively across T pruning stages, each stage labeled as $t \in \{1,\dots,T\}$. We define the starting set of N convolutional filters at $t = 1$ as

$$\mathcal{K}^{(1)} = \left\{\boldsymbol{K}_1^{(1)}, \boldsymbol{K}_2^{(1)}, \dots, \boldsymbol{K}_N^{(1)}\right\}. \tag{4}$$

For $t > 1$, $\mathcal{K}^{(t)}$ contains the filters of $\mathcal{K}^{(t-1)}$ minus the ones that have been pruned, i.e., $\mathcal{K}^{(t)} \subset \mathcal{K}^{(t-1)}$ with cardinality $|\mathcal{K}^{(t)}| = N_t < |\mathcal{K}^{(t-1)}| = N_{t-1}$, where $|\mathcal{K}^{(1)}| = N_1 = N$. At each stage t, filters are sorted according to their current importance score, so

$$\forall\, \boldsymbol{Q}, \boldsymbol{R} \in \mathcal{K}^{(t)}, \quad \boldsymbol{Q} \prec \boldsymbol{R} \Longleftrightarrow s(\boldsymbol{Q}) < s(\boldsymbol{R}) \tag{5}$$

where $s(\cdot)$ is any of the importance scoring functions from Section II-B and $\boldsymbol{Q} \prec \boldsymbol{R}$ indicates that $\boldsymbol{Q}$ precedes $\boldsymbol{R}$ in an ordered list. To explicitly represent the ordered list of filters in $\mathcal{K}^{(t)}$, we introduce the indexing function $\sigma_t : \{1,\dots,N_t\} \to \{1,\dots,N\}$ such that

$$\boldsymbol{K}^{(t)}_{\sigma_t(1)} \prec \boldsymbol{K}^{(t)}_{\sigma_t(2)} \prec \cdots \prec \boldsymbol{K}^{(t)}_{\sigma_t(N_t)}. \tag{6}$$

So, $\boldsymbol{K}^{(t)}_{\sigma_t(i)}$ denotes the filter occupying the i-th position in the ranking.

Of these, we prune P filters. The process is iterated T times during training, resulting in the removal of a total of PT filters. At the end of the pruning process, the number of surviving filters is $L = N - PT$. Therefore, at each stage, we target only the $M_t = N_t - L$ least important filters, which define the *pruning budget*.

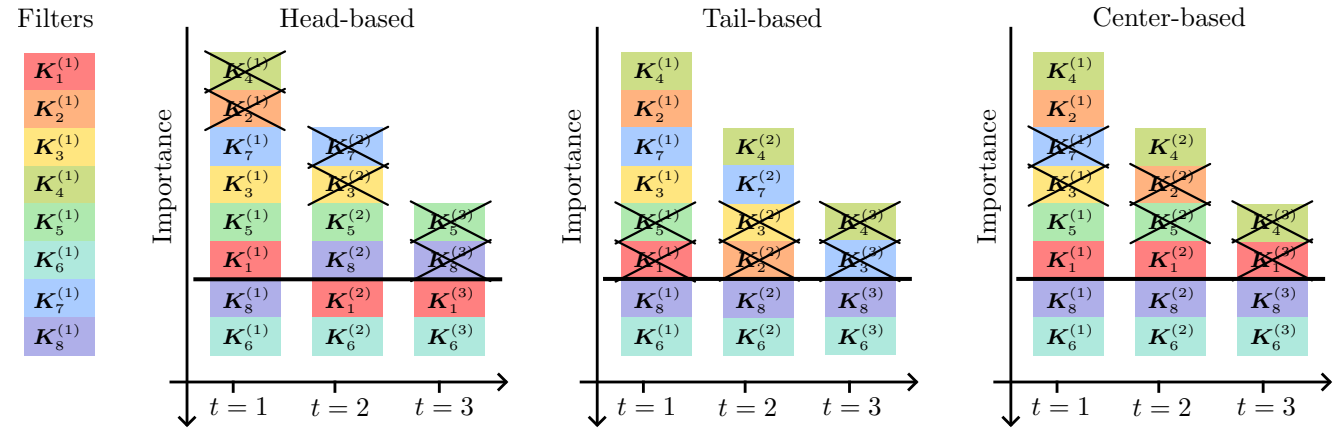

Fig. 1. Illustration of different progressive pruning scheduling strategies applied to a convolutional layer. Each colored block represents a filter. At the beginning of each pruning stage, filters are sorted from the least to the most important. Filters removed at each stage are cross-marked. Blocks below the solid black line are excluded from the pruning budget.

In standard iterative pruning strategies (that we refer to as *head-based* scheduling), filters are removed following the current ranking from the least important going upwards. Specifically, at stage t, the set of filters to be pruned is

$$\mathcal{P}^{(t)}_{\text{head}} = \left\{ \boldsymbol{K}^{(t)}_{\sigma_t(1)}, \boldsymbol{K}^{(t)}_{\sigma_t(2)}, \ldots, \boldsymbol{K}^{(t)}_{\sigma_t(P)} \right\}. \tag{7}$$

This procedure results in the gradual elimination of increasingly higher-ranked filters, until the target pruning rate is reached. While effective in practice, this approach is not necessarily optimal.

To study the effect of pruning order, we consider alternative scheduling strategies that operate within the same pruning budget.

We introduce *tail-based* scheduling, that reverses the removal order of head-based scheduling. Here, the pruning operation starts removing the most important filters within the pruning budget and then targets the filters with decreasingly lower ranking. Formally, at stage t, the pruning set is defined as

$$\mathcal{P}^{(t)}_{\text{tail}} = \left\{ \boldsymbol{K}^{(t)}_{\sigma_t(M_t-P+1)}, \boldsymbol{K}^{(t)}_{\sigma_t(M_t-P+2)}, \ldots, \boldsymbol{K}^{(t)}_{\sigma_t(M_t)} \right\}. \tag{8}$$

Finally, we define *center-based* scheduling, which starts removing filters of intermediate importance within the pruning budget. . This means that we initially preserve both the lowest and highest ranked filters and we prune them only in later stages. In this case the pruning set at stage t is defined as

$$\mathcal{P}^{(t)}_{\text{center}} = \left\{ \begin{array}{c} \boldsymbol{K}^{(t)}_{\sigma_t\left(\frac{M_t}{2}-\frac{P}{2}+1\right)}, \boldsymbol{K}^{(t)}_{\sigma_t\left(\frac{M_t}{2}-\frac{P}{2}+2\right)}, \cdots \\ \cdots, \boldsymbol{K}^{(t)}_{\sigma_t\left(\frac{M_t}{2}+\frac{P}{2}-1\right)}, \boldsymbol{K}^{(t)}_{\sigma_t\left(\frac{M_t}{2}+\frac{P}{2}\right)} \end{array} \right\}, \tag{9}$$

where M_t and P are assumed to be even without loss of generality.

Fig. 1 provides a graphical illustration of the different pruning scheduling strategies. Here, we consider a layer with $N = 8$ filters. We apply $T = 3$ pruning stages, each removing $P = 2$ filters, leaving $L = 2$ filters at the end of the process. As an example, at $t = 1$, filters are sorted by importance score such that $\sigma_1(i) = (4, 2, 7, 3, 5, 1, 8, 6)_i$ for $i = 1, \ldots, 8$. With head-based scheduling, we remove filters 4 and 2. Fine-tuning is performed, then at $t = 2$ we evaluate the importance scores again ($\sigma_2(i) = (7, 3, 5, 8, 1, 6)_i$ for $i = 1, \ldots, 6$) and we remove filters 7 and 3. With tail-based scheduling, we keep the $L = 2$ most important filters and remove filters 5 and 1. Fine tuning is performed, then at $t = 2$ we evaluate the importance scores again ($\sigma_2(i) = (4, 7, 3, 2, 8, 6)_i$ for $i = 1, \ldots 6$), keep the $L = 2$ most important filters and remove filters 3 and 2. Finally, with center-based scheduling, we remove central filters (within the pruning budget) 7 and 3. Fine tuning is performed, then at $t = 2$ we evaluate the importance scorse again ($\sigma_2(i) = (4, 2, 5, 1, 8, 6)_i$ for $i = 1, \ldots 6$), and we remove central filters (within the pruning budget) 2 and 5.

All considered, scheduling strategies remove the same total number of filters, but differ in how pruning decisions are distributed and what filters are preserved in the end. This formulation enables a controlled investigation of how pruning order influences optimization dynamics and final performance.

IV. Experiments

In this section, we describe the experimental setup used to evaluate the proposed pruning scheduling strategies. We conduct experiments on CIFAR-100 [32] and ImageNet-1K [33], two widely used benchmarks for image classification.

- CIFAR-100 consists of 60 000 color images of size 32×32, divided into 100 fine-grained classes. The dataset is split into 50 000 training images and 10 000 test images, with 500 training samples and 100 test samples per class. During training, data augmentation is applied and includes random horizontal flipping, random cropping with padding of 4 pixels, random rotations up to 15 degrees. All the inputs are normalized using the dataset-specific mean and standard deviation.
- ImageNet-1K is a large-scale image classification dataset containing approximately 1.28 million training images

and $50\,000$ validation images, spanning 1000 object categories. Training data augmentation consists of random resized cropping to 224×224 and random horizontal flipping. Normalization is employed using the ImageNet mean and standard deviation.

We evaluate our pruning strategies on ResNet-18, ResNet-101 [34], and ResNeXt-50 [35], which are representative convolutional neural network architectures with increasing depth and complexity. For ImageNet-1K, we use the standard ResNet-18, ResNet-101, and ResNeXt-50 architectures as originally proposed. For CIFAR-100, we employ CIFAR-style variants of ResNet architectures, adapted to the smaller input resolution. These models use a 3×3 convolution as the first layer instead of the 7×7 convolution and max-pooling layer used in ImageNet models.

We adopt different training and fine-tuning procedures for CIFAR-100 and ImageNet-1K. On CIFAR-100, we train all models from scratch starting from random initialization. We pretrain the networks for 200 epochs using stochastic gradient descent (SGD) with a batch size of 128. We use an initial learning rate of 0.1, momentum set to 0.9, and weight decay of 5×10^{-4}. Learning rate decays with a multi-step scheduling strategy, with decay milestones at epochs 60, 120, and 160, and a decay factor of 0.2. After each pruning stage, we recover the model's performance by fine-tuning the pruned model for 40 epochs using the same batch size and optimizer configuration, with an initial learning rate of 0.02. We adopt a cyclic learning rate scheduling, where the learning rate is scaled down twice by a factor of 0.2 after 25 and after 35 epochs. The learning rate scheduling resets at each pruning stage.

On ImageNet-1K, we conduct experiments starting from pretrained models provided by the official PyTorch repository. We perform initial training for 90 epochs using SGD with a batch size of 256. The optimizer uses an initial learning rate of 0.1, momentum of 0.9, and weight decay of 1×10^{-4}. The learning rate decays by a factor of 0.1 every 30 epochs using a step-based scheduling. After each pruning stage, we fine-tune the model for 20 epochs using the same batch size and optimizer configuration. During fine-tuning, the learning rate scales down by a factor of 0.2 after 12 epochs and then after 18 epochs. As before, we reinitialize the learning rate scheduling at each pruning step.

V. Results and Discussion

A. Comparison of Pruning Scheduling Strategies

In this section, we compare the effect of different pruning scheduling strategies with four target pruning rates, 50.00%, 62.50%, 75.00%, and 87.50%. We perform pruning progressively over five prune-fine-tuning stages, each one removing the same amount of filters. We sort filters using the magnitude ℓ_1 scoring system as in (1). We always remove filters locally in a layer-wise fashion, pruning each convolutional layer independently according to its own importance scores.

Table I and Table II show the results on CIFAR-100 and ImageNet-1K, respectively. We observe that, compared to the head-based pruning strategy, the tail-based scheduling generally leads to slightly improved performance. More notably, the center-based scheduling strategy yields the maximum gains in a majority of cases. In more challenging settings, such as ImageNet-1K or with higher pruning rates, performance gains become more pronounced, suggesting that the pruning order plays a more significant role.

TABLE I
CIFAR-100 accuracy with varying pruning rate

Model	Pruning Rate	Scheduling Strategy		
		Head-based	Tail-based	Center-based
ResNet-18	50.00%	70.21%	70.52%	**71.85%**
	62.50%	70.14%	70.39%	**71.44%**
	75.00%	67.70%	67.71%	**70.74%**
	87.50%	56.85%	62.54%	**64.62%**
ResNet-101	50.00%	**74.82%**	74.21%	74.49%
	62.50%	74.05%	74.10%	**74.91%**
	75.00%	72.84%	73.22%	**74.35%**
	87.50%	68.70%	70.44%	**73.31%**
ResNeXt-50	50.00%	75.14%	74.83%	**75.56%**
	62.50%	74.04%	74.13%	**75.23%**
	75.00%	72.16%	72.62%	**74.62%**
	87.50%	64.48%	69.32%	**72.90%**

TABLE II
ImageNet accuracy with varying pruning rate

Model	Pruning Rate	Scheduling Strategy		
		Head-based	Tail-based	Center-based
ResNet-18	50.00%	54.84%	56.35%	**58.19%**
	62.50%	51.28%	52.75%	**56.87%**
	75.00%	45.46%	49.11%	**52.16%**
	87.50%	31.73%	35.02%	**44.68%**
ResNet-101	50.00%	67.96%	67.11%	**68.50%**
	62.50%	65.51%	66.62%	**68.29%**
	75.00%	61.98%	64.47%	**67.56%**
	87.50%	49.73%	53.28%	**58.69%**
ResNeXt-50	50.00%	67.75%	67.98%	**68.66%**
	62.50%	65.73 %	65.67%	**68.48%**
	75.00%	59.97%	63.19%	**67.87%**
	87.50%	45.39%	50.31%	**59.60%**

B. The Effect of the Number of Pruning Stages

To further analyze the impact of pruning scheduling strategies and disentangle their effect from the number of pruning stages, we conduct an additional set of experiments varying the number of prune-fine-tuning stages with a fixed target pruning rate. The goal of these experiments is to verify that the observed performance differences are genuinely induced by the pruning scheduling, rather than being a side effect of using too many or too few pruning steps.

In this setting, we fix the target pruning rate to 0.75 and we progressively prune through 3, 5, or 10 stages. As in the previous experiments, we evaluate filter importance layer-wise with a magnitude-based criterion. Table III and Table IV summarize the results on CIFAR-100 and ImageNet-1K, respectively. Across both datasets, the performance trends observed in the previous experiments are largely preserved. In particular, the

TABLE III
CIFAR-100 ACCURACY WITH A VARYING NUMBER OF PRUNING STAGES

Model	Number of Stages	Scheduling Strategy Head-based	Tail-based	Center-based
ResNet-18	3	67.37%	67.38%	**69.28%**
	5	67.70%	67.71%	**70.74%**
	10	67.80%	67.80%	**71.96%**
ResNet-101	3	72.18%	73.04%	**74.61%**
	5	72.84%	73.22%	**74.35%**
	10	73.35%	73.71%	**75.18%**
ResNeXt-50	3	71.59%	72.25%	**74.94%**
	5	72.16%	72.62%	**74.62%**
	10	72.65%	72.83%	**75.35%**

TABLE IV
IMAGENET ACCURACY WITH A VARYING NUMBER OF PRUNING STAGES

Model	Number of Stages	Scheduling Strategy Head-based	Tail-based	Center-based
ResNet-18	3	43.60%	**47.57%**	46.68%
	5	45.46%	49.11%	**52.16%**
	10	46.84%	48.24%	**53.26%**
ResNet-101	3	59.95%	**62.05%**	61.83%
	5	61.98%	64.47%	**67.56%**
	10	62.89%	64.71%	**67.89%**
ResNeXt-50	3	58.40%	60.91%	**61.53%**
	5	59.97%	63.19%	**67.87%**
	10	60.75%	63.59%	**68.28%**

TABLE V
IMAGENET ACCURACY WITH DIFFERENT SCORING STRATEGIES

Model	Scoring	Scheduling Strategy Head-based	Tail-based	Center-based
ResNet-18	Activation	45.11%	46.73%	**52.12%**
	Taylor	44.70%	45.52%	**51.15%**
ResNeXt-50	Activation	60.77%	**62.64%**	62.04%
	Taylor	60.14%	63.13%	**64.88%**

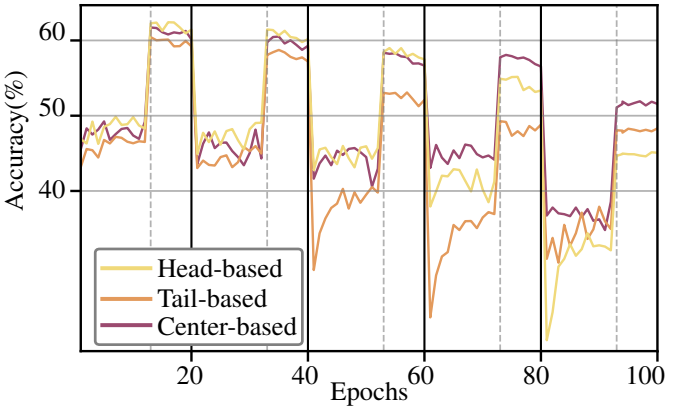

Fig. 2. Validation accuracy of ResNet-18 on ImageNet-1K with a fixed set of surviving filters. Pruning is applied at the beginning of training and at each pruning stage, indicated by the solid black vertical lines. Dotted vertical lines mark the first learning rate decay, which causes an accuracy increase.

tail-based and center-based scheduling strategies consistently outperform the head-based approach. These results indicate that the benefits of scheduling are not tied to a specific choice of pruning steps, but persist across both coarse and fine-grained pruning regimes, confirming that the pruning order constitutes a key factor in progressive pruning.

C. The Effect of Different Filter Importance Criteria

In a final set of experiments, we investigate whether the benefits of the proposed pruning scheduling strategies are consistent across different filter importance criteria. While all previous experiments rely on magnitude-based scoring, this analysis aims to assess the robustness of the scheduling effect when using other common importance scores. Specifically, we consider the activation-based scoring criterion as in (2) and the first-order Taylor score as in (3).

We conduct these experiments on ImageNet-1K using ResNet-18 and ResNeXt-50. We progressively reach the target pruning rate of 0.75 through 5 pruning stages, with the same procedure of earlier experiments. When required by the scoring metric, we extract a calibration set of 50 000 samples from the training set. Table V presents the results, showing that the overall performance trends observed in previous experiments are preserved. These findings suggest that the sensitivity to pruning scheduling is not tied to a specific importance metric.

D. Ablation Study: Isolating the Effect of Scheduling

To verify that the performance gains observed with the proposed pruning scheduling strategies are indeed attributable to the removal order of the filters rather than to the specific set of pruned filters, we conduct an ablation study on ResNet-18 using the ImageNet-1K dataset. As before, we reach the layer-wise target pruning rate of 0.75 through 5 pruning stages.

This time, we start by fixing once for all the set of filters that will survive. Using Fig. 1 as an example, if filters 8 and 6 are excluded from the pruning budget at $t = 1$, they are guaranteed to survive the whole pruning process. In this way, the final topology of the model does not depend on the pruning order and all pruning scheduling result in the same set of surviving filters. Any observed differences in performance can thus be directly attributed to how scheduling affects parameters updates during training, independent of filter selection.

Fig. 2 illustrates the validation accuracy during training with this setup, from which emerge several observable trends. In head-based scheduling, the network initially experiences small drops in accuracy at the early pruning stages, followed by larger decreases in later stages. In contrast, the tail-based scheduling strategy induces large accuracy drops earlier, while the effect decreases in the last stages. The center-based scheduling strategy shows a more balanced trend, with accuracy decreasing more evenly across all pruning stages. Notably, the tail-based strategy still consistently outperforms the head-based scheduling, and the center-based scheduling strategy achieves the best overall performance. This confirms that the observed gains are not simply due to the set of removed filters, but also to how pruning order affects the fine-tuning process.

VI. Conclusion and Future Work

In this work, we investigated the role of pruning scheduling in progressive structured filter pruning, showing that the order of filter removal represents an additional and impactful design dimension. We introduced tail-based and center-based scheduling and demonstrated through extensive experiments that they consistently outperform the standard head-based progressive approach. These gains are robust across different pruning rates, numbers of pruning stages, and importance criteria.

Future work could explore adaptive scheduling strategies guided by training dynamics or gradient information, as well as integrations with soft or gradual pruning methods to further improve optimization stability during model compression.

References

[1] U. Shah and A. Harpale, "A Review of Deep Learning Models for Computer Vision," in *2018 IEEE Punecon*, Nov. 2018, pp. 1–6. doi:10.1109/PUNECON.2018.8745417

[2] J. Ren, H. Gaber, and S. S. Al Jabar, "Applying Deep Learning to Autonomous Vehicles: A Survey," in *2021 4th International Conference on Artificial Intelligence and Big Data (ICAIBD)*, May 2021, pp. 247–252. doi:10.1109/ICAIBD51990.2021.9458968

[3] F. Hu, "Survey on Neural Networks in Natural Language Processing," in *2023 IEEE International Conference on Control, Electronics and Computer Technology (ICCECT)*, Apr. 2023, pp. 591–594. doi:10.1109/ICCECT57938.2023.10141113

[4] U. Kulkarni, A. S. Hosamani, A. S. Masur, S. Hegde, G. R. Vernekar, and K. Siri Chandana, "A Survey on Quantization Methods for Optimization of Deep Neural Networks," in *2022 International Conference on Automation, Computing and Renewable Systems (ICACRS)*, Dec. 2022, pp. 827–834. doi:10.1109/ICACRS55517.2022.10028742

[5] R. Mishra, H. P. Gupta, and T. Dutta, "A Survey on Deep Neural Network Compression: Challenges, Overview, and Solutions," Oct. 2020. doi:10.48550/arXiv.2010.03954

[6] J. Gou, B. Yu, S. J. Maybank, and D. Tao, "Knowledge Distillation: A Survey," *International Journal of Computer Vision*, vol. 129, no. 6, pp. 1789–1819, Jun. 2021. doi:10.1007/s11263-021-01453-z

[7] H. Cheng, M. Zhang, and J. Q. Shi, "A Survey on Deep Neural Network Pruning: Taxonomy, Comparison, Analysis, and Recommendations," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 46, no. 12, pp. 10 558–10 578, Dec. 2024. doi:10.1109/TPAMI.2024.3447085

[8] S.-K. Yeom, K.-H. Shim, and J.-H. Hwang, "Toward Compact Deep Neural Networks via Energy-Aware Pruning," Mar. 2022. doi:10.48550/arXiv.2103.10858

[9] W. Hu, P. Henderson, and J. Cano, "ICE-Pruning: An Iterative Cost-Efficient Pruning Pipeline for Deep Neural Networks," in *2025 International Joint Conference on Neural Networks (IJCNN)*, Jun. 2025, pp. 1–8. doi:10.1109/IJCNN64981.2025.11227410

[10] K. Zhu, F. Hu, Y. Ding, Y. Dong, and R. Wang, "Incremental Soft Pruning to Get the Sparse Neural Network During Training," in *2024 International Joint Conference on Neural Networks (IJCNN)*, Jun. 2024, pp. 1–9. doi:10.1109/IJCNN60899.2024.10650747

[11] J. Liu, Z. Xu, R. Shi, R. C. C. Cheung, and H. K. H. So, "Dynamic Sparse Training: Find Efficient Sparse Network From Scratch With Trainable Masked Layers," May 2020. doi:10.48550/arXiv.2005.06870

[12] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning Filters for Efficient ConvNets," Mar. 2017. doi:10.48550/arXiv.1608.08710

[13] Y. He, X. Zhang, and J. Sun, "Channel Pruning for Accelerating Very Deep Neural Networks," in *2017 IEEE International Conference on Computer Vision (ICCV)*, Oct. 2017, pp. 1398–1406. doi:10.1109/ICCV.2017.155

[14] J.-H. Luo, J. Wu, and W. Lin, "ThiNet: A Filter Level Pruning Method for Deep Neural Network Compression," in *2017 IEEE International Conference on Computer Vision (ICCV)*, Oct. 2017, pp. 5068–5076. doi:10.1109/ICCV.2017.541

[15] X. Gao, Y. Zhao, Ł. Dudziak, R. Mullins, and C.-z. Xu, "Dynamic Channel Pruning: Feature Boosting and Suppression," Jan. 2019. doi:10.48550/arXiv.1810.05331

[16] M. Zhu and S. Gupta, "To prune, or not to prune: Exploring the efficacy of pruning for model compression," Nov. 2017. doi:10.48550/arXiv.1710.01878

[17] Y. He, G. Kang, X. Dong, Y. Fu, and Y. Yang, "Soft Filter Pruning for Accelerating Deep Convolutional Neural Networks," Aug. 2018. doi:10.48550/arXiv.1808.06866

[18] Y. He, X. Dong, G. Kang, Y. Fu, C. Yan, and Y. Yang, "Asymptotic Soft Filter Pruning for Deep Convolutional Neural Networks," *IEEE Transactions on Cybernetics*, vol. 50, no. 8, pp. 3594–3604, Aug. 2020. doi:10.1109/TCYB.2019.2933477

[19] H. Mostafa and X. Wang, "Parameter Efficient Training of Deep Convolutional Neural Networks by Dynamic Sparse Reparameterization," May 2019. doi:10.48550/arXiv.1902.05967

[20] U. Evci, T. Gale, J. Menick, P. S. Castro, and E. Elsen, "Rigging the Lottery: Making All Tickets Winners," Jul. 2021. doi:10.48550/arXiv.1911.11134

[21] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, and C. Zhang, "Learning Efficient Convolutional Networks through Network Slimming," Aug. 2017. doi:10.48550/arXiv.1708.06519

[22] X. Ding, G. Ding, Y. Guo, and J. Han, "Centripetal SGD for Pruning Very Deep Convolutional Networks With Complicated Structure," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2019, pp. 4938–4948. doi:10.1109/CVPR.2019.00508

[23] Z. You, K. Yan, J. Ye, M. Ma, and P. Wang, "Gate Decorator: Global Filter Pruning Method for Accelerating Deep Convolutional Neural Networks," Sep. 2019. doi:10.48550/arXiv.1909.08174

[24] P. Molchanov, A. Mallya, S. Tyree, I. Frosio, and J. Kautz, "Importance Estimation for Neural Network Pruning," Jun. 2019. doi:10.48550/arXiv.1906.10771

[25] H. Peng, J. Wu, S. Chen, and J. Huang, "Collaborative Channel Pruning for Deep Networks," in *Proceedings of the 36th International Conference on Machine Learning*. PMLR, May 2019, pp. 5113–5122.

[26] H. Hu, R. Peng, Y.-W. Tai, and C.-K. Tang, "Network Trimming: A Data-Driven Neuron Pruning Approach towards Efficient Deep Architectures," Jul. 2016. doi:10.48550/arXiv.1607.03250

[27] X. Ding, G. Ding, Y. Guo, J. Han, and C. Yan, "Approximated Oracle Filter Pruning for Destructive CNN Width Optimization," May 2019. doi:10.48550/arXiv.1905.04748

[28] Z. Zhuang, M. Tan, B. Zhuang, J. Liu, Y. Guo, Q. Wu, J. Huang, and J. Zhu, "Discrimination-aware Channel Pruning for Deep Neural Networks," Jan. 2019. doi:10.48550/arXiv.1810.11809

[29] Y. He and L. Xiao, "Structured Pruning for Deep Convolutional Neural Networks: A Survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 46, no. 5, pp. 2900–2919, May 2024. doi:10.1109/TPAMI.2023.3334614

[30] J. T. C. Min and M. Motani, "DropNet: Reducing Neural Network Complexity via Iterative Pruning," Jul. 2022. doi:10.5555/3524938.3525805

[31] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz, "Pruning Convolutional Neural Networks for Resource Efficient Inference," Jun. 2017. doi:10.48550/arXiv.1611.06440

[32] A. Krizhevsky, "Learning Multiple Layers of Features from Tiny Images," 2009.

[33] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2009, pp. 248–255. doi:10.1109/CVPR.2009.5206848

[34] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016, pp. 770–778. doi:10.1109/CVPR.2016.90

[35] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated Residual Transformations for Deep Neural Networks," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul. 2017, pp. 5987–5995. doi:10.1109/CVPR.2017.634