

STRIPEPRUNER: TOWARDS MULTIGRANULARITY NETWORK COMPRESSION WITH DYNAMIC FILTER SLIMMING

LILI SONG^{1,2}, YONGCHAO LIU², SUMAIRA HUSSAIN², HAOHAN CAO², HANG QU²
AND XINFENG LIU^{2,*}

¹Computational Intelligence Center

²School of Computer and Artificial Intelligence
Shandong Jianzhu University

No. 1000, Fengming Road, Lingang Development Zone, Jinan 250101, P. R. China
{ songlili22; sumairahussain }@sdjzu.edu.cn; { 2024110115; 2025115025; 2025115027 }@stu.sdjzu.edu.cn

*Corresponding author: liuxinfeng18@sdjzu.edu.cn

Received December 2025; revised March 2026

ABSTRACT. *Deep convolutional neural networks (CNNs) are the backbone of modern intelligent perception systems; however, their remarkable accuracy comes at the cost of substantial computational and memory overhead. To facilitate efficient deployment on resource-constrained devices, we propose StripePruner – a novel compression approach that performs dynamic filter pruning at multiple granularities. In contrast to conventional filter or channel pruning, our method decomposes filters into stripes and adaptively prunes redundant stripes by optimizing a jointly learned L_1 -regularized filter skeleton matrix (FSm) alongside scaling factors. This enables fine-grained structural compression without significant accuracy loss. In addition, an early stopping mechanism is used to prevent overfitting and shorten the training duration. Extensive experiments demonstrate the efficacy of StripePruner: on VGG16 and ResNet56, it achieves high parameters reduction rate of 96.68% and 82.76%, on CIFAR-10, respectively, with only marginal accuracy degradation. The proposed method also demonstrates excellent performance on the deeper ResNet110 architecture; furthermore, it exhibits strong effectiveness on the CIFAR-100 dataset, which has a larger number of categories and higher complexity. Notably, our approach substantially outperforms strong baselines like Network Slimming in terms of FLOPs reduction, reaching up to 83.89% and 86.37% for VGG16 and ResNet56 on CIFAR-10, underscoring its superior practical deploy ability for efficient deep learning models.*

Keywords: Convolutional neural networks, Model compression, Stripe pruning, Dynamic optimization, Parameter tuning, Resource-constrained devices

1. Introduction. Deep convolutional neural networks (CNNs) have catalyzed revolutionary advances across diverse fields, including computer vision, speech recognition, and natural language processing [1, 2, 3]. However, the pursuit of higher accuracy has led to dramatically increased model complexity, with top-performing architectures often comprising hundreds of millions of parameters (Params) and requiring billions of floating point operations (FLOPs). This trend has created significant deployment challenges for resource-constrained mobile and embedded systems [4]. Although existing pruning and compression techniques offer partial solutions, maintaining prediction accuracy while substantially reducing computational overhead remains an open research problem. The evolution of key DNN architectures clearly illustrates this complexity escalation: from AlexNet with 60M Params and 0.7G FLOPs [5], to VGG16 with 138M Params and 15.5G FLOPs [6], to ResNet50 with 25.6M Params and 3.8G FLOPs [7]. The Transformer architecture

further escalated these demands with 110M Params and 4.1B FLOPs [8]. This trajectory not only demonstrates remarkable progress in feature extraction capabilities but also underscores the urgent need for innovative pruning methodologies to address the ongoing computational arms race.

The current escalation of challenges stems from the convergence of two profound technological trends. On the one hand, the proliferation of Internet of Things (IoT) technology has led to the widespread deployment of massive edge devices. These devices are often subject to stringent resource constraints – including limited computational power, storage capacity, and energy supply. Yet, with the growing demand for intelligent capabilities, they are increasingly required to process and interpret complex sensor data in real time. This shift drives the migration of high-performance deep neural network (DNN) models, traditionally hosted in the cloud, toward the network edge. On the other hand, advanced DNN architectures such as Transformers and large-scale vision models are evolving toward greater parameter scales and higher computational complexity to meet the escalating performance demands in tasks like image recognition and natural language processing. This trend of “model expansion” fundamentally conflicts with the inherent lightweight, low-power nature of edge devices.

These two forces interact to intensify the tension between model performance and deployment feasibility. IoT scenarios urgently require efficient localized intelligent inference to ensure real-time responsiveness, data privacy, and system reliability. At the same time, the substantial computational overhead of state-of-the-art models makes them difficult to run directly on resource-constrained devices. This contradiction has spurred and accelerated the research wave in model compression and optimization technologies. The core objective is to bridge this gap: while preserving model accuracy as much as possible, techniques such as pruning, quantization, knowledge distillation, and neural architecture search are employed to significantly reduce the model’s demand for computational resources, storage, and energy consumption. This enables advanced intelligent algorithms to be deployed in real-world edge computing environments, thereby promoting the widespread and inclusive adoption of artificial intelligence across broader application scenarios.

Pruning methods, such as filter pruning or channel pruning, can generate structured and hardware-friendly sparse patterns, which facilitate practical acceleration. However, they may sacrifice valuable spatial information encoded within the filters. On the other hand, fine-grained weight pruning better preserves the model’s representational capacity, but it typically leads to unstructured sparsity and fragmented computational patterns, which are difficult to execute efficiently on general-purpose hardware. Therefore, how to retain the accuracy advantages of fine-grained pruning while maintaining the structured sparsity required for hardware acceleration remains a central challenge in the field of model pruning.

Recent advances in fine-grained pruning have attempted to bridge this gap by exploring fine-grained structured and semistructured sparsity patterns. For example, Meng et al. [9] proposed stripe-wise pruning (SWP) to achieve fine-grained sparsity while maintaining filter independence. Building on this, Liu et al. [10] introduced IEESP in 2025, which utilizes internal network feature information to guide sparse stripe selection, thereby further enhancing precision. Similarly, Xiang et al. [11] proposed MaxQ, a multi-axis query methodology for N:m sparsity that dynamically identifies weight importance during training. In the realm of large language models, Zhang et al. [12] developed FinerCut to perform interpretable fine-grained layer pruning. Despite their effectiveness, these methods often rely on additional mechanisms or static evaluation criteria, and in many cases lack a unified way to adapt pruning structures in real time to the evolving training dynamics.

To address these challenges, we propose StripePruner, which aims to construct a novel compression paradigm by deeply integrating fine-grained stripe-wise pruning with dynamic filter slimming (SP-DFS). The core innovations of this paper are mainly reflected in the following two aspects. First, a dynamic adaptive filter importance evaluation mechanism is proposed to drive multi-granularity structured pruning. Addressing the limitation that traditional static evaluation cannot capture feature evolution, we combine the filter skeleton matrix (FSm) with trainable scaling factors (SR) to construct a dynamic stripe importance metric. This approach allows for a more accurate assessment of filter stripe importance across different training stages and network depths, thereby achieving dynamic, adaptive, and precise stripe pruning. Second, for specific pruning tasks, an intelligent early-stopping strategy is adopted to automatically determine the optimal stopping point for model pruning. This mechanism achieves high compression rates while effectively preventing performance degradation and resource waste caused by excessive pruning, significantly improving the stability and automation level of the pruning process.

Extensive experiments on VGG16, ResNet56, and ResNet110 demonstrate that the proposed method effectively improves compression rates across various models while maintaining high accuracy. These research results hold significant practical value for deploying real-time inference tasks on resource-constrained edge devices. The structure of this paper is organized as follows: Section 2 reviews related work in model pruning; Section 3 systematically elaborates the proposed SP-DFS framework, with a focus on the dynamic filter slimming mechanism; Section 4 presents the complete experimental setup and result analysis; Section 5 summarizes the paper and outlines directions for future research.

2. Related Work. Deep convolutional neural networks exhibit significant parameters redundancy, where a substantial portion of weights contribute minimally to model performance. This observation has spurred the development of various model compression techniques, which can be broadly categorized into four approaches: pruning, quantization, decomposition, and knowledge distillation [13, 14, 15, 16]. Among these techniques, network pruning is one of the most effective and widely studied approaches, showing particular efficacy for CNNs, Transformer models, and YOLO-based detectors in computer vision tasks [1, 17]. Pruning methods can be systematically classified based on their granularity and methodology, as discussed in the following sections.

Unstructured pruning operates at the finest granularity by removing individual weights without considering the network structure. The pioneering work of Han et al. established the foundation for this approach through their three-step pipeline: 1) learning connection importance, 2) pruning unimportant connections, and 3) retraining sparse networks. It demonstrated remarkable compression ratios of 9-13 times on AlexNet and VGG16 while maintaining baseline accuracy. This breakthrough showed that neural networks have significant redundancy that can be exploited for model compression [18].

Subsequent research has enhanced pruning along several dimensions. For importance criteria, Li et al. proposed to use L_1 -norms of filters to measure importance, removing filters with smaller magnitudes [19]. Liu et al. introduced network slimming, which imposes L_1 regularization on the scaling factors of batch normalization layers to automatically identify and prune insignificant channels [20]. For dynamic pruning, He et al. proposed soft filter pruning (SFP), a method that enables pruned filters to remain in a latent state and be reactivated during training, thus preserving the model's representational capacity [21]. The geometric median filter pruning (FPGM) method advanced the field by identifying redundant filters based on their geometric relationships rather than magnitude, proving particularly effective for ResNet architectures [22].

Structured pruning aims to remove entire network components while preserving hardware-compatible structures [23, 24, 25, 26, 27, 28]. Representative methods such as DepGraph [29] and HSC [30] achieve this by eliminating redundant structures while maintaining efficient hardware layouts. To determine pruning criteria, Huang and Wang [31] proposed data-driven sparse structure selection (SSS) by regularizing scaling factors. Lin et al. [32] introduced generative adversarial learning (GAL) to optimize the pruned structure, while Lin et al. [33] developed HRank, which prunes filters based on the rank of feature maps. Fang et al. [4] enhanced a structured pruning framework by incorporating dual attention modules to generate masks with ultra-high sparsity.

Recently, a series of fine-grained pruning methods have been proposed to further improve the trade-off between sparsity and accuracy. Specifically, Liu et al. developed SOKS [34], a method that automatically searches for optimal kernel shapes for stripe-wise pruning using a differentiable mask. Additionally, Li et al. introduced group sparsity (Hinge) [35], utilizing a hinge-based sparsity constraint to balance filter pruning and decomposition. These methods represent the state of the art in the pruning field, but they face certain limitations: First, they rely on static evaluation criteria, unable to adapt to the dynamic evolution of feature importance; Second, they employ fixed training schedules, lacking adaptive control over the pruning progress. This directly leads to accuracy loss under extreme compression and a waste of computational resources. Clearly, the field currently lacks an efficient pruning mechanism capable of simultaneously balancing dynamic structural adaptability and training convergence awareness.

To address the aforementioned challenges of static nature and inefficiency, we construct a dynamic FSM learning mechanism, empowering the network with the ability to reconstruct its own structure in real time during training; addressing the resource waste caused by fixed training schedules, this paper adopts an intelligent early-stopping strategy, achieving adaptive perception and regulation of the training convergence state. Through this dual-pronged multi-granularity optimization, SP-DFS effectively bridges the gap between dynamic importance evaluation and structured pruning. While maintaining high accuracy, it significantly breaks through the bottlenecks of compression efficiency and training costs.

3. The Proposed Method. In this section, we first elucidate the effectiveness of stripe pruning and then present two fundamental concepts: filter stripes (defined as 1×1 slices of individual kernels) and the FSM, which adaptively determines stripe retention [10]. Building upon this conceptual framework, we introduce the overall model architecture, elaborate on the systematic pruning procedure, and explain the core principles underlying our pruning methodology. As illustrated in Figure 1, stripe pruning involves both weight and shape attributes, with each stripe contributing variably to model performance.

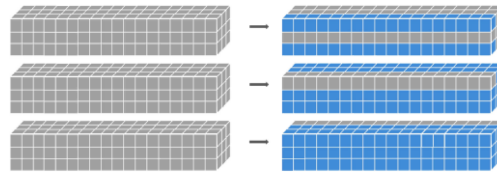


FIGURE 1. The stripe pruning process

Building upon this conceptual framework, we introduce the overall model architecture of our proposed SP-DFS method. As illustrated in Figure 2, the dynamic pruning mechanism operates in a sequential pipeline. First, the standard convolution kernels in the

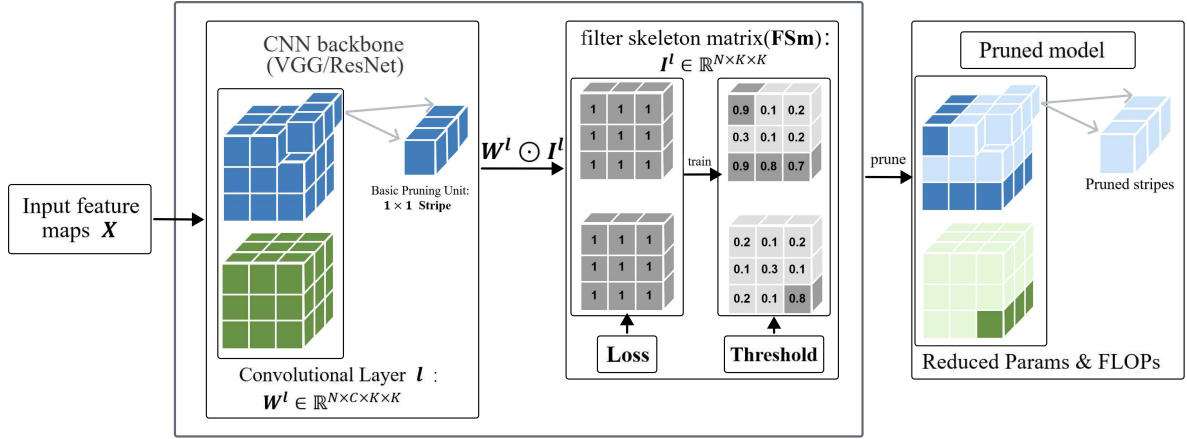


FIGURE 2. The overall framework and dynamic pruning process of the proposed SP-DFS method

backbone network (e.g., VGG/ResNet) are logically viewed as a collection of independent 1×1 stripes. During the training phase, the FSm is initialized to an all-ones matrix and participates in the forward propagation via element-wise multiplication with the weight tensor W^l . As training progresses, the values in the FSm diverge based on their importance gradient: the FSm values corresponding to critical stripes retain high magnitudes (depicted as dark blocks in Figure 2), while those for redundant stripes decay toward zero. Finally, stripes corresponding to FSm elements smaller than the threshold are permanently removed from the network, and the remaining weights are reconstructed into a compact, shift-based convolution format for efficient inference.

Following this overview, we elaborate on the systematic pruning procedure and explain the core principles in the subsequent sections.

3.1. Filter stripe pruning. Consider the weight tensor $W^l \in \mathbb{R}^{N \times C \times K \times K}$ for the l -th convolutional layer, where N , C and K represent the number of filters, the number of input channels, and the kernel size, respectively. Note that for notational simplicity, we define the kernel dimensions as $K \times K$ assuming square kernels (standard in VGG and ResNet), though our method applies to rectangular kernels. The size of FSm is $\mathbb{R}^{N \times K \times K}$ in the l -th convolutional layer. The FSm assigns an individual $K \times K$ mask to each output channel. More precisely, each element in FSm corresponds to a stripe in the filter. Before learning the filters, we initialize each filter matrix to all ones. Although this adds some computational overhead, it ensures that critical stripes are not pruned. The overall loss function is formulated as follows:

$$L = \sum_{(x,y)} \text{loss}(f(x, W \odot I), y), \quad (1)$$

where I is the FSm, and $\odot$ denotes the Hadamard product. The standard forward process is as follows:

$$X_{n,h,w}^{l+1} = \sum_{c=1}^C \sum_{i=1}^K \sum_{j=1}^K I_{n,i,j}^l \times W_{n,c,i,j}^l \times X_{n,h+i-\frac{K+1}{2},w+j-\frac{K+1}{2}}^l. \quad (2)$$

$X_{n,h,w}^l$ represents the feature value at spatial position (h, w) of channel n in the output feature map of layer l . $I_{n,i,j}^l$ represents the FSm value at spatial position (i, j) for channel n in layer l . $W_{n,c,i,j}^l$ represents the weight value at spatial position (i, j) of input channel c in convolution kernel n of layer l . The stripe-wise convolution can be represented as

$$X_{n,h,w}^{l+1} = \sum_{i=1}^K \sum_{j=1}^K \left(\sum_{c=1}^C I_{n,i,j}^l \times W_{n,c,i,j}^l \times X_{n,h+i-\frac{K+1}{2},w+j-\frac{K+1}{2}}^l \right). \quad (3)$$

Instead of following the standard convolutional computation order, stripe pruning reorders the computational sequence. It is worth noting that no additional operations are added to the network. The gradients of W and I are as follows:

$$\frac{\partial L}{\partial W_{n,c,i,j}^l} = I_{n,i,j}^l \times \sum_{h=1}^{M_H} \sum_{w=1}^{M_W} \frac{\partial L}{\partial X_{n,h,w}^{l+1}} \times X_{c,h+i-\frac{K+1}{2},w+j-\frac{K+1}{2}}^l, \quad (4)$$

$$\frac{\partial L}{\partial I_{n,i,j}^l} = \sum_{c=1}^C \left(W_{n,c,i,j}^l \times \sum_{h=1}^{M_H} \sum_{w=1}^{M_W} \frac{\partial L}{\partial X_{n,h,w}^{l+1}} \times X_{c,h+i-\frac{K+1}{2},w+j-\frac{K+1}{2}}^l \right), \quad (5)$$

where M_H and M_W represent the height and width of the feature map, respectively.

F_{Sm} is introduced to learn the skeleton characteristics of the convolution kernel that guide the convolution process through the convolution kernel skeleton. The shape of the convolution kernel can also be pruned during the weight pruning of the convolution kernel. In the process of stripe pruning, each convolution kernel is kept independent of the other, the F_{Sm} is used to learn the filter shape, and the optimal shape is learned through F_{Sm}, to achieve finer granularity of the filter and improve the pruning ratio.

The method of stripe pruning transforms how convolution is calculated to achieve a stripe-by-stripe process. The SP-DFS method retains more valid information than pruning the entire filter. In addition, according to the stripe (i.e., a filter of 1×1), the filter is pruned at the stripe level, which is more detailed than the traditional pruning and can achieve a higher pruning rate. Each filter adaptively learns its optimal morphology guided by F_{Sm}, ensuring that the pruning process removes redundant stripes while preserving most critical information. Consequently, the model maintains near-original accuracy.

Assume that a convolutional layer contains N convolutional kernels, each denoted by F , with a width of K , a height of K , and C channels. In this context, a stripe refers to each individual 1×1 filter within a kernel, meaning that every kernel F comprises $K \times K$ stripes. In this work, we employ a learnable matrix, referred to as the F_{Sm}, to guide the pruning of these stripes in order to achieve a more fine-grained pruning effect. Other network pruning methods include kernel pruning, which removes entire convolutional kernels; channel pruning, which eliminates selected channels; stripe pruning, which removes specific stripes; and group pruning, which removes all filters at the same spatial locations. In comparison, stripe pruning offers a more precise pruning strategy than both kernel pruning and channel pruning, thereby enabling a higher overall pruning rate.

3.2. Filter skeleton matrix. F_{Sm} is used to learn the “shape” property of each convolutional filter, that is, each convolutional filter contains multiple stripes. The size of F_{Sm} is $N \times K \times K$. Each element in the F_{Sm} represents a corresponding stripe in the convolution filter. By applying a predefined hyperparameter threshold and SR, the F_{Sm} values are improved to reflect the importance of each stripe. The detailed strategy for selecting these hyperparameters, specifically a two-stage tuning process involving grid search, is elaborated in Section 4.2. A higher value indicates a more critical stripe in the subsequent pruning process, while stripes with values below the threshold are masked out. In this manner, the F_{Sm} effectively learns which stripes should be retained by each convolution kernel.

The learning process of F_{Sm} is structured as follows. Initially, the F_{Sm} is set to a matrix of ones, representing all stripes within each filter. In the training process, the F_{Sm} multiplies the convolutional filter weights to induce sparsity, thereby improving the

filter's shape. To promote sparsity in the FSm, L_1 regularization is applied, which helps quantify the importance of each stripe. Establishing specific thresholds makes it possible to identify and subsequently prune less significant stripes.

FSm is integrated into the backpropagation process in the update mechanism, where the gradient of the loss function with respect to the sparse filter weights is computed, enabling the FSm to learn the significance of each stripe within the filters adaptively. This adaptive learning facilitates the determination of the filter's optimal shape, allows for more precise pruning, and enhances overall pruning efficiency. Unlike methods that prune entire filters, SP-DFS retains more critical information, thereby preserving the filter's effectiveness. The operational dynamics of the FSm during the training phase are intuitively depicted, with its values providing the criteria for selecting the most significant stripes (see Figure 2). In contrast to random pruning, the SP-DFS approach is shown to be more effective in eliminating redundant stripes.

To execute the pruned model, the final inference weights are computed as the product of the trained FSm and the original filter weights. Since only this product is retained and deployed, the FSm mechanism itself does not introduce any computational overhead during inference. Moreover, the method fundamentally retains filter-level independence, a key advantage over other pruning methods which creates dependencies between filters, thereby violating the standard operational principles of convolutional layers.

3.3. Pruning process. For model pruning, our methodology involves the optimization of the SR and threshold parameters before model training. We introduce a learnable FSm during the training phase while maintaining the original network structure. In each training iteration, we adaptively adjust both the stripe patterns within filters and BN layer parameters based on the evolving filter weights and dynamic thresholds. The FSm undergoes continuous optimization throughout this process.

Pruning operations are withheld in the training process. Instead, we employ the FSm as a multiplicative mask on the filters, updating the FSm parameters while maintaining standard convolution throughout. This approach preserves all network connections while learning optimal pruning patterns, ultimately retaining the model parameters that demonstrate the best predictive performance. During the inference phase, we initialize the process by loading the enhanced model parameters. Then, we permanently integrate the learned feature selection mask through matrix multiplication with the corresponding weight matrices. Pruning is then executed sequentially following the model's forward propagation order, targeting convolutional BN, and linear layers. The standard convolution operations are replaced with a shift-based convolution operator, effectively implementing 1×1 kernel operations. The final pruned model is then ready for prediction tasks.

In this paper, we modify the pruning approach by first decomposing each original filter into $K \times K$ spatial stripes. These stripes are then pruned independently, which inherently disrupts the filter's original structural integrity but enables more fine-grained compression. To address this, each independent stripe is treated as a 1×1 convolution kernel, enabling the use of the shift convolution operator. During forward propagation, the convolution operations are performed on each stripe and the input feature map to generate the corresponding output feature map. The output feature maps from all stripes are then aggregated to produce the final output feature map. In backpropagation, the gradient of each stripe is computed independently, and the corresponding stripe weights are updated accordingly. This approach achieves independent convolution for each stripe while preserving the overall structure of the convolutional layer.

It is well-established that the contributions from individual stripes within the network are heterogeneous. The basic idea of pruning is to build a compact network; therefore,

the FSm should be sparse. More precisely, the corresponding stripes can be pruned when the values in FSm are close to 0. Therefore, we introduce FSm in the network and impose regularization on FSm to make it sparse by defining the loss as follows:

$$L = \sum_{(x,y)} \text{loss}(f(x, W \odot I), y) + \alpha g(I), \quad (6)$$

where α corresponds to the SR and $g(I)$ denotes the L_1 -norm penalty on I , a common practice in pruning methods [19, 20, 36]. The SR regulates the intensity of sparse regularization, thereby promoting sparsity in the FSm and enabling effective filter stripe pruning. Specifically, $g(I)$ is written as

$$g(I) = \sum_{l=1}^L g(I^l) = \sum_{l=1}^L \left(\sum_{n=1}^N \sum_{i=1}^K \sum_{j=1}^K |I_{n,i,j}^l| \right). \quad (7)$$

In the training process, by optimizing the loss function augmented with the L_1 regularization term (see Equation (6)), the I is learned as a sparse matrix. During inference, for any FSm element satisfying $\|I_{n,i,j}^l\| < \theta$, where θ is the threshold, the corresponding stripe weights $W_{n,:,i,j}^l$ are permanently zeroed out and removed, completing the dynamic stripe pruning process.

3.4. Early stopping. During pruning-aware training, the effective model capacity decreases as stripes are progressively removed. In this setting, using a fixed number of epochs can be inefficient, and validation accuracy may even drop after a plateau. Therefore, we employ an early-stopping rule based on validation accuracy: training stops when the validation accuracy does not improve for a predefined number of consecutive epochs.

Let $\mathcal{A}_{val}^{(s)}$ denote the validation accuracy at epoch s . We record the best validation accuracy $\mathcal{A}_{best}$ and the corresponding model parameters Θ_{best} :

$$\mathcal{A}_{best} = \max_{1 \leq s \leq t} \mathcal{A}_{val}^{(s)}, \quad \Theta_{best} = \arg \max_{\Theta_s} \mathcal{A}_{val}^{(s)}. \quad (8)$$

A patience parameter P_{limit} and a counter $cnt^{(t)}$ are used to track consecutive epochs without improvement. This parameter is determined empirically based on validation convergence; for example, we set $P_{limit} = 30$ in our experiments (see Section 4.4):

$$cnt^{(t)} = \begin{cases} 0, & \text{if } \mathcal{A}_{val}^{(t)} > \mathcal{A}_{best} \\ cnt^{(t-1)} + 1, & \text{otherwise.} \end{cases} \quad (9)$$

Training is stopped when

$$cnt^{(t)} \geq P_{limit}. \quad (10)$$

Upon termination, the model state is reverted to Θ_{best} to initialize the subsequent pruning phase. This strategy not only eliminates computational redundancy once performance saturation is detected but also acts as a regularization mechanism, effectively mitigating the risk of overfitting and preventing accuracy degradation associated with prolonged training in later epochs.

4. Experiments and Results.

4.1. Experimental dataset. CIFAR-10 and CIFAR-100 datasets are used in this paper which contain 60,000 32×32 RGB images. CIFAR-10 comprises 10 classes while CIFAR-100 comprises 100 classes; both follow the standard split of 50,000 training and 10,000 test images. Input images are normalized per channel to zero mean and unit variance. In the training process, we apply standard data augmentation-zero-padding followed by random cropping and horizontal flipping to improving generalization.

4.2. Experimental setup. In the experiments, the proposed SP-DFS method is evaluated on standard benchmark architectures, including VGG16, ResNet56, and ResNet110. Dynamic stripe pruning is achieved by introducing the filter skeleton matrix. To effectively balance model accuracy and sparsification, we design a two-stage parameter tuning strategy, which separately optimizes the scaling coefficients and the pruning threshold to avoid cumbersome joint search. First, the pruning threshold is fixed to a conservative value (e.g., 0.05), and a grid search for SR is conducted over a moderate range (typically within $[0.5, 3.0] \times 10^{-5}$, tailored to specific architectures and datasets) to identify a stable SR value that can introduce effective sparsity regularization while maintaining convergence. Subsequently, based on the selected SR, the threshold is fine-tuned within a narrow interval (e.g., 0.05-0.09) to adjust the final pruning ratio and achieve specific compression targets. This decoupled process simplifies hyperparameter selection and forms a reproducible tuning workflow across different datasets. It is first preliminarily evaluated on VGG16 and ResNet56 using the CIFAR-10 dataset. Subsequently, the pruning experiments are further extended to the deeper ResNet110 architecture, and performance verification is conducted on VGG16 and ResNet56 using the CIFAR-100 dataset.

FLOPs, Params and model accuracy are used as the evaluation indicators of model performance. In this paper, the pruning method's effectiveness is evaluated by benchmarking its experimental results against several existing model compression studies. Given the variability in Params, FLOPs, and accuracy among the unpruned models reported in various studies, the comparison is anchored on three key metrics: pruned Params (Params Pruned), pruned FLOPs (FLOPs Pruned) and accuracy difference (i.e., the change in accuracy relative to the baseline model).

4.3. Experimental results and analysis on CIFAR-10. The cross-ablation analysis in Table 1 and Table 2 shows that increasing the epochs, the SR and the threshold values enhance pruning effectiveness, resulting in fewer Params and reduced FLOPs. While the overall trend shows a reduction in Params and FLOPs, it is worth noting that minor fluctuations (e.g., at epoch 260 in Table 1) may occur. These fluctuations are attributed to the stochastic nature of the dynamic optimization process. Table 2 illustrates the impact of varying SR and thresholds on model performance. While the optimal trade-off for VGG16 is achieved at $SR = 1.0 \times 10^{-5}$ and $Threshold = 0.05$, it is worth noting that the proposed method consistently yields competitive results across a broader range of configurations

TABLE 1. Performance comparison with different epochs on VGG16

Backbone	Epochs	FLOPs (M)	Params (M)	Accuracy (%)
VGG16	1	627.36	14.76	36.70
	10	627.36	14.76	85.07
	100	191.23	0.97	93.15
	130	161.44	0.79	92.77
	160	140.17	0.68	93.28
	200	121.88	0.59	92.45
	220	109.21	0.52	91.82
	230	106.28	0.51	90.25
	240	101.11	0.49	93.45
	250	107.54	0.50	92.21
	260	77.53	0.34	90.65
	270	96.75	0.44	91.86
	280	84.78	0.42	91.40

TABLE 2. Performance comparison with different SR and threshold on VGG16

Backbone	SR	Threshold	Params (M)	FLOPs (M)	Accuracy (%)
VGG16	0.8e-5	0.05	0.54	112.66	92.35
	1e-5	0.05	0.49	101.11	93.45
	1.2e-5	0.05	0.44	95.28	92.78
	1.4e-5	0.05	0.62	134.38	93.05
	1e-5	0.07	0.32	75.77	91.35
	1e-5	0.09	0.23	56.42	91.79

TABLE 3. Pruning performance of SP-DFS for VGG16 and ResNet56 on CIFAR-10 dataset

Backbone	Params (M)	Params Pruned (%)	FLOPs (M)	FLOPs Pruned (%)	Accuracy (%)
VGG16 (Baseline)	14.76	–	627.36	–	93.25
VGG16 (Pruned)	0.49	96.68	101.11	83.89	93.45
ResNet56 (Baseline)	0.87	–	252.59	–	93.10
ResNet56 (Pruned)	0.15	82.76	34.44	86.37	92.10

(e.g., $SR \in [0.8, 1.4] \times 10^{-5}$). Even with suboptimal settings, our method outperforms most baseline approaches, demonstrating that SP-DFS is robust to hyperparameter variations and does not rely on a single, brittle configuration to achieve effectiveness.

In particular, when applied to VGG16, SP-DFS yielded a 96.68% reduction in Params and an 83.89% reduction in FLOPs while maintaining excellent accuracy (see Table 3). Similarly, for ResNet56, the method reduced the Params by 82.76% and FLOPs by 86.37%, with only a negligible 1% drop in accuracy. These consistent results validate the effectiveness of SP-DFS in substantially reducing both the model’s parameters and computational complexity while keeping performance within acceptable limits. Notably, the outstanding performance achieved on VGG16 underscores the immense potential of this method for improving conventional CNN architectures.

To comprehensively evaluate the effectiveness of StripePruner, we benchmark it against a diverse set of state-of-the-art pruning methods. Our comparison includes classic structured pruning approaches such as PFEC [19], network slimming [20], SSS [31], GAL [32], FPGM [22], and HRank [33]. More importantly, to demonstrate the advantages of our fine-grained strategy, we also compare against advanced fine-grained pruning methods, including SWP [9], SOKS [34], and Hinge [35], which serve as our most direct competitors sharing similar objectives of fine-grained structural optimization. The performance of different compression methods is analyzed in terms of the following aspects: Params and Params Pruned, FLOPs and FLOPs Pruned, accuracy and accuracy difference (i.e., the change in accuracy relative to the baseline model). Results for the compared methods are taken from their original papers and may use different training settings; thus, baseline accuracies can differ across methods.

Compared to methods like PFEC and GAL, the proposed method prunes most of the redundant parameters, resulting in a much more compact model with significantly reduced storage requirements (See Table 4, Figure 3 and Figure 4). Despite the high pruning rate, SP-DFS not only preserves accuracy but also yields a 0.20% absolute improvement. This improvement shows that removing redundant components can reduce overfitting and enhance feature representation, thereby improving overall generalization. While many pruning techniques sacrifice some accuracy for high pruning rates (for example, GAL and

TABLE 4. Various pruning techniques for the VGG16 model on CIFAR-10 dataset

Backbone	Params Pruned (%)	FLOPs Pruned (%)	Baseline accuracy (%)	Accuracy (%)	Accuracy difference
SSS [31]	30.00	50.00	—	—	—
PFEC [19]	64.00	34.20	93.25	93.40	+0.15
GAL [32]	82.20	45.20	93.96	93.42	−0.54
SWP [9]	<u>92.66</u>	71.16	93.25	93.62	+0.37
SOKS-80 [34]	78.30	72.20	93.53	94.01	+0.48
Hinge [35]	80.05	39.07	94.02	94.45	+0.43
HRank [33]	76.50	92.00	93.96	91.23	−2.73
Slimming [20]	88.50	51.00	93.66	93.80	+0.14
FPGM [22]	—	34.00	93.58	94.00	<u>+0.42</u>
SP-DFS	96.68	<u>83.89</u>	93.25	93.45	+0.20

Note: For SOKS [34], the notations “SOKS-80” and “SOKS-55” refer to the specific configurations where the pruning ratio of convolution weights is set to 80% and 55%, respectively, as defined in the original paper.

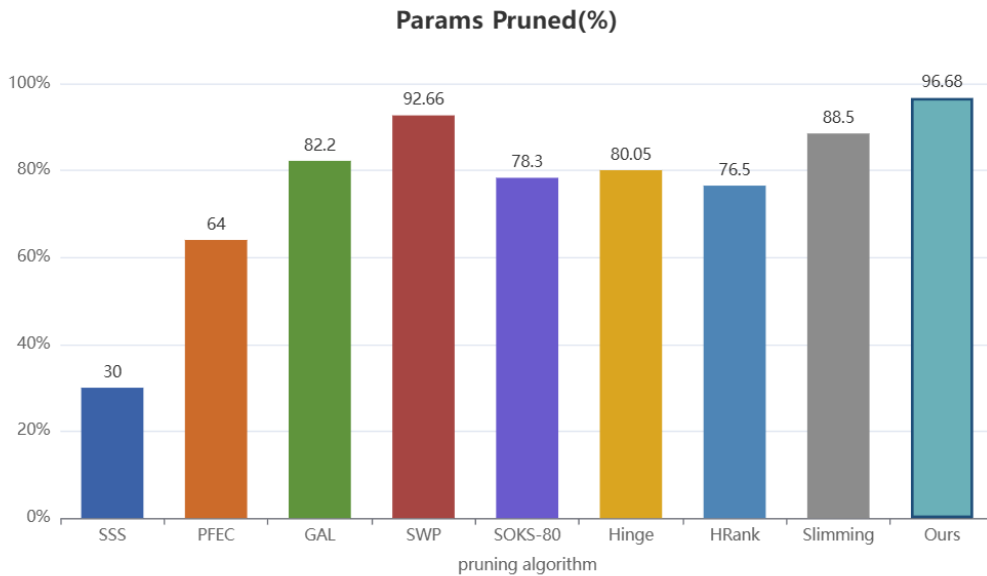


FIGURE 3. Params Pruned by applying various pruning methods to VGG16 on CIFAR-10

HRank lose 0.54% and 2.73% in accuracy, respectively), the proposed method still boosts accuracy even at very high pruning levels. This demonstrates its ability to effectively filter out irrelevant information and redundant stripes while maintaining or even enhancing performance.

According to the results in Table 5, our proposed SP-DFS demonstrates significant advantages over other approaches for pruning the ResNet56 network. Values shown in bold and underlined denote the best and second-best results, respectively, in Table 5. The method achieves a notable compression ratio, reducing Params by 82.76% and FLOPs by 86.37%, substantially outperforming alternatives such as PFEC, GAL, and SWP. Although this high-ratio compression entails a modest accuracy decrease (within an acceptable range), it confirms the SP-DFS’s ability to simplify the model while preserving

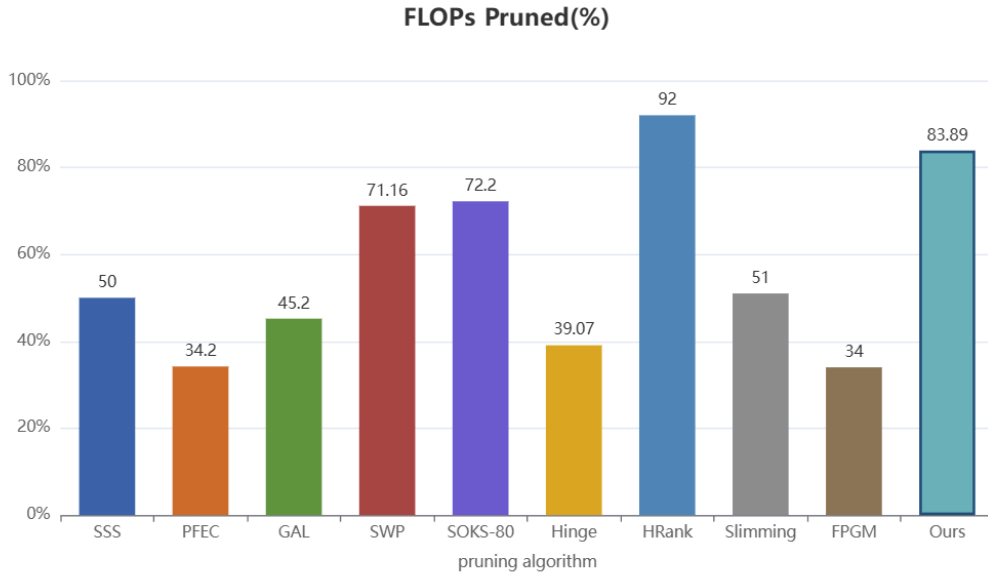


FIGURE 4. FLOPs reduced by applying different pruning methods to VGG-16 on CIFAR-10

TABLE 5. Params and FLOPs Pruned by applying various pruning techniques to the ResNet56 model on CIFAR-10 dataset

Backbone	Params Pruned (%)	FLOPs Pruned (%)	Baseline accuracy (%)	Accuracy (%)	Accuracy difference
PFEC [19]	13.70	27.60	93.04	93.06	+0.02
GAL [32]	65.90	60.20	93.26	91.58	-1.68
SWP [9]	<u>72.00</u>	67.00	93.10	92.67	-0.34
SOKS-55 [34]	54.10	51.70	93.06	92.02	-1.04
Hinge [35]	27.54	44.40	94.74	94.67	-0.07
HRank [33]	16.28	29.30	93.26	93.52	+0.26
Slimming [20]	68.10	<u>80.22</u>	93.26	90.72	-2.54
FPGM [22]	—	52.60	93.59	93.10	-0.49
SP-DFS	82.76	86.37	93.10	92.10	-1.00

Note: Results for SSS [31] are omitted due to the lack of ResNet56 experiments. For FPGM [22], the parameter reduction metric is marked as “—” because it was not reported in the original publication.

competitive performance. Consequently, for applications demanding maximal compression on ResNet architectures with minimal accuracy trade-off, the SP-DFS provides a highly competitive solution. In summary, the SP-DFS delivers considerable resource savings with an effective compression-accuracy balance, enabling efficient model deployment in resource-constrained environments.

4.4. Experimental results and analysis on CIFAR-100. Building on the experiments conducted on CIFAR-10, the evaluation was extended to CIFAR-100. More specifically, training will be stopped after 30 consecutive epochs with no improvement in model accuracy. Stripe pruning was applied to VGG16, ResNet56, and ResNet110, with baseline models established through ablation studies on the pruned architectures. According to Table 6, all the architectures achieved a Params reduction of over 40% and FLOPs reduction exceeding 50%, while experiencing only approximately a 1% decrease in accuracy.

TABLE 6. Pruning performance of SP-DFS on VGG16, ResNet56 and ResNet110 on CIFAR-100 dataset

Backbone	Params (M)	Params Pruned (%)	FLOPs (M)	FLOPs Pruned (%)	Accuracy (%)
VGG16 (Baseline)	14.81	—	627.45	—	71.71
VGG16 (Pruned)	2.61	82.38	302.68	51.76	70.58
ResNet56 (Baseline)	0.88	—	252.60	—	70.34
ResNet56 (Pruned)	0.51	42.05	89.23	64.68	69.76
ResNet110 (Baseline)	1.77	—	508.43	—	71.62
ResNet110 (Pruned)	0.79	55.37	174.05	65.77	70.70

VGG16 achieved the most aggressive compression, reducing Params by over 80%, yet incurred an accuracy loss of only around 1%, demonstrating its suitability for severely resource-constrained environments. Under a moderate compression ratio, ResNet56 exhibited the smallest accuracy degradation (0.58%), suggesting that shallower residual networks retain greater robustness to pruning. ResNet110 attained intermediate compression rates, with approximately 55% Params reduction and 66% FLOPs reduction, while maintaining an accuracy loss below 1%, thus striking an effective balance between model compactness and performance. These results indicate that the proposed pruning strategy performs well not only on shallow architectures but also generalizes effectively to deeper networks such as ResNet110, highlighting its practical utility for real-world deployment.

The pruning effects for ResNet110, ResNet56, and VGG16 under different sparsity rates (SR) for the models are illustrated in Figure 5. The first row compares the accuracy and FLOPs of the baseline models against the pruned models across various intensities, where the blue and red curves represent the accuracy and FLOPs of the baseline and pruned models, respectively. The second row shows the corresponding changes in accuracy and Params, with blue and green curves indicating the accuracy and Params quantity of the baseline and pruned models, respectively. The first to third columns correspond to the changes in accuracy, FLOPs, and Params before and after pruning for ResNet110, ResNet56, and VGG16, in order. The experimental results indicate that within a certain range of pruning intensities (i.e., as the sparsity rate increases), both FLOPs and Params are significantly reduced, while the accuracy of the pruned models remains largely

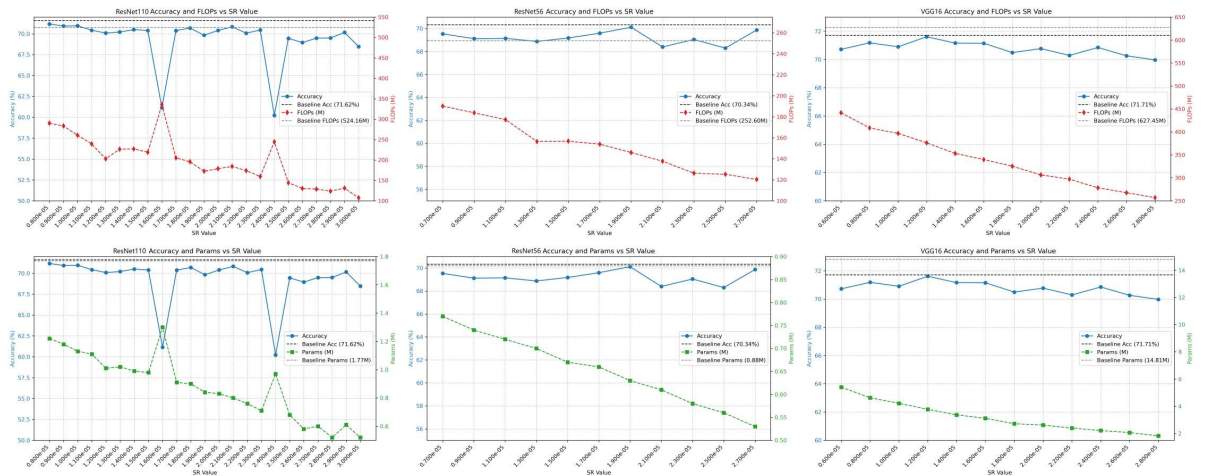


FIGURE 5. Accuracy, FLOPs, and Params comparison between baseline and SP-DFS across ResNet110, ResNet56, and VGG16 on CIFAR-100

consistent with that of the baseline models. This demonstrates that the proposed SP-DFS method effectively compresses the model size – significantly reducing both FLOPs and Params – while maintaining stable model performance, thereby offering a viable technical pathway for deploying large-scale models in resource-constrained environments.

4.5. Analysis of consistency in accuracy progression. To further investigate the relationship between the number of training epochs and model accuracy, FLOPs as well as Params, additional experiments were conducted using the CIFAR-100 dataset on the ResNet110. Figure 6 illustrates the changes in model accuracy, Params and FLOPs per epoch under the unpruned setting ($SR = 0$). The experimental results indicate that within the first 20 epochs, the model accuracy increases rapidly. Between 20 and 80 epochs, the rate of accuracy improvement slows, with a noticeable surge occurring around the 80th epoch. Beyond 80 epochs, the accuracy gradually converges and stabilizes. Both the Params and the FLOPs per training epoch remain constant throughout the training process as no pruning was applied.

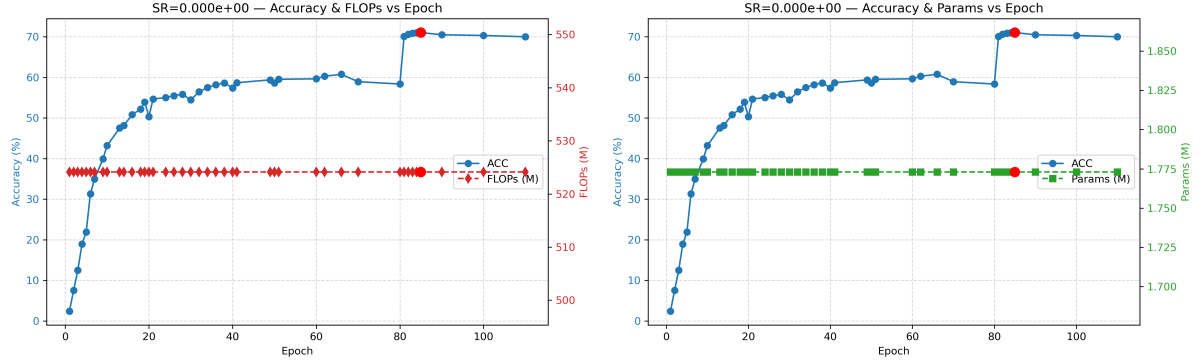


FIGURE 6. The variation in accuracy, FLOPs, and Params of ResNet110 on CIFAR-100 without pruning

Subsequently, further experiments were conducted on the CIFAR-100 dataset using ResNet110 under varying pruning intensities controlled by the SR parameter, where a higher SR corresponds to a greater pruning ratio. The results are presented in Figure 7. It can be observed that with the proposed SP-DFS method, the trend of accuracy change across different pruning intensities remains highly consistent with that of the unpruned model: a noticeable surge in accuracy occurs around epoch 80, after which it stabilizes at a constant value. In terms of FLOPs, as shown by the red curve in the top row of

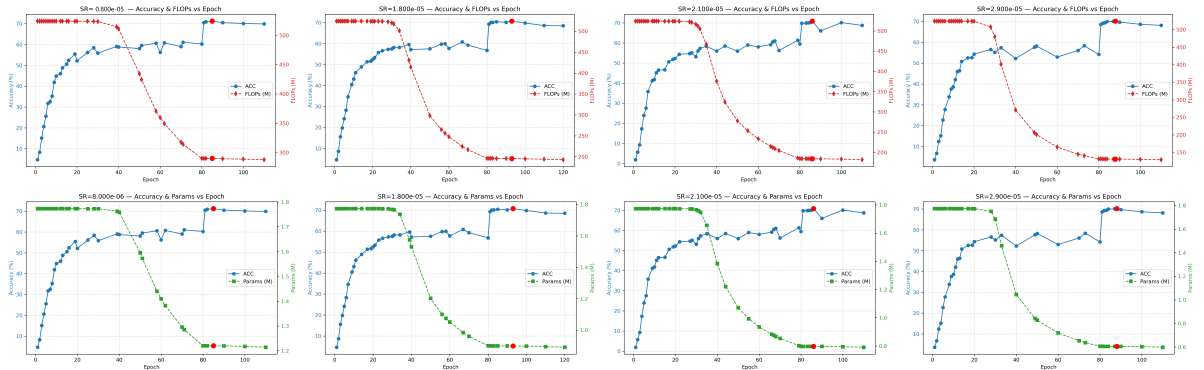


FIGURE 7. The variation in accuracy, FLOPs, and Params of ResNet110 with SP-DFS on CIFAR-100

Figure 7, a higher SR leads to a more rapid decrease in FLOPs, ultimately achieving a reduction of approximately 65.8%. Moreover, different SR settings have minimal impact on the final FLOPs reduction. Similarly, the effect of varying pruning intensities on the Params, indicated by the green curve in the second row of Figure 7, is largely consistent with that on FLOPs. Through pruning, the Params can also be reduced by around 55.4%.

To validate the generality of the proposed method, identical experiments were conducted on ResNet56 and VGG16 models, with selected results illustrated in Figures 8 and 9. The results demonstrate that after pruning with SP-DFS, the models retain nearly identical accuracy to the original ones, while significantly reducing both Params and FLOPs. These findings confirm that the proposed SP-DFS method can effectively identify and remove redundant parameters across different network architectures. It substantially decreases computational and storage overhead while preserving the training effectiveness of critical weights, thereby maintaining accuracy nearly consistent with that of the original models after pruning.

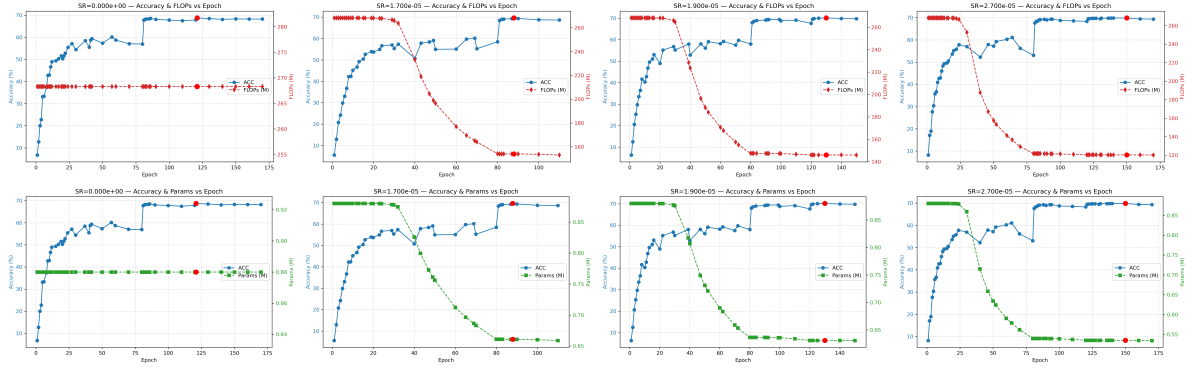


FIGURE 8. The variation in accuracy, FLOPs, and Params of ResNet56 with SP-DFS on CIFAR-100

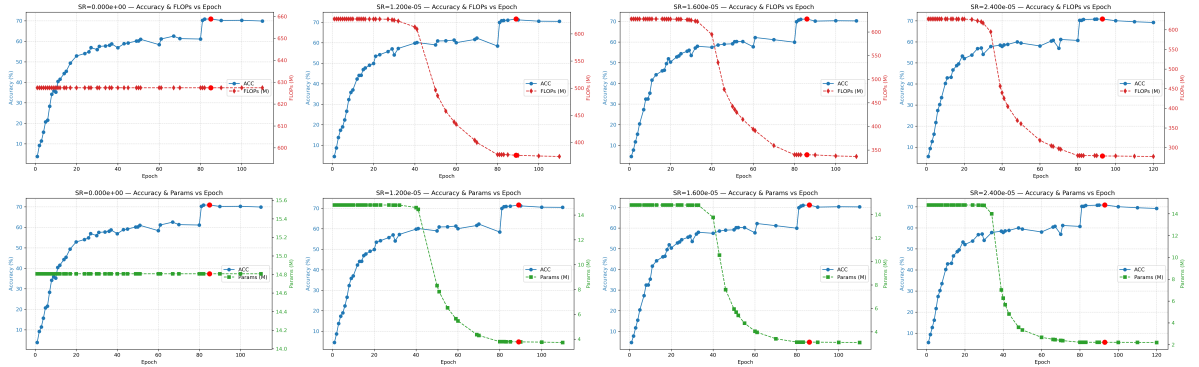


FIGURE 9. The variation in accuracy, FLOPs, and Params of VGG16 with SP-DFS on CIFAR-100

5. Conclusion. In this paper, we propose a new dynamic stripe pruning method called StripePruner to address the key challenge of balancing computational efficiency and model accuracy in deep convolutional neural networks. By integrating the filter skeleton matrix with trainable scaling coefficients, it achieves precise evaluation of filter importance across different training phases and network depths, thereby driving multi-granularity structured pruning. Furthermore, an early stopping strategy tailored for pruning tasks is introduced,

which intelligently determines the optimal pruning stopping point. This ensures high compression rates while effectively preventing performance degradation due to over-pruning, significantly enhancing the stability and automation level of the pruning process.

Extensive experiments on multiple architectures and datasets validate StripePruner's efficacy. On CIFAR-10, it achieved Params reduction rates of 96.68% for VGG16 and 82.76% for ResNet56 with only marginal accuracy loss. The method also demonstrated strong performance on the deeper ResNet110 and the more complex CIFAR-100 dataset. Notably, StripePruner substantially outperformed strong baselines such as Network Slimming in FLOPs reduction, with rates reaching 83.89% for VGG16 and 86.37% for ResNet56 on CIFAR-10. These results highlight the method's superior deployability for efficient deep learning, positioning it as an effective solution for resource-constrained environments.

In the future, we will explore deeper integration of the proposed method with techniques such as quantization and knowledge distillation. We also aim to expand its applicability to more complex model architectures and challenging vision tasks (e.g., object detection and semantic segmentation), and conduct systematic validation on larger-scale datasets, in order to continuously improve inference efficiency and optimize memory consumption on edge devices.

Acknowledgment. This work was supported by the Doctoral Grant of Shandong Jianzhu University (X24047), the Startup Foundation for Advanced Talents (X25102), and the Shandong Province Youth Innovation Team (2023KJ071).

REFERENCES

- [1] H. Zhang, Y. Zhou and G. H. Wang, Dense vision transformer compression with few samples, *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp.15825-15834, 2024.
- [2] H. Kheddar, M. Hemis and Y. Himeur, Automatic speech recognition using advanced deep learning approaches: A survey, *Information Fusion*, vol.109, 102422, 2024.
- [3] J. Cao, P. Ye, S. Li, C. Yu, Y. Tang, J. Lu et al., MADTP: Multimodal alignment-guided dynamic token pruning for accelerating vision-language transformer, *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp.15710-15719, 2024.
- [4] Y. C. Fang, W. Z. Li, Y. Zeng, Q. N. Lu and S. L. Lu, Pushing to the limit: An attention-based dual-prune approach for highly-compacted CNN filter pruning, *Journal of Computer Science and Technology*, vol.40, no.3, pp.805-820, 2025.
- [5] A. Krizhevsky, I. Sutskever and G. E. Hinton, ImageNet classification with deep convolutional neural networks, *Advances in Neural Information Processing Systems*, vol.25, pp.1097-1105, 2012.
- [6] K. Simonyan and A. Zisserman, Very deep convolutional networks for large-scale image recognition, *International Conference on Learning Representations*, 2015.
- [7] K. He, X. Zhang, S. Ren and J. Sun, Deep residual learning for image recognition, *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp.770-778, 2016.
- [8] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez et al., Attention is all you need, *Advances in Neural Information Processing Systems*, vol.30, pp.5998-6008, 2017.
- [9] F. Meng, H. Cheng, K. Li, H. Luo, X. Guo, G. Lu et al., Pruning filter in filter, *Conference on Neural Information Processing Systems*, 2020.
- [10] J. Liu, L. Huang, M. Feng, A. Guo, L. Yin and J. Zhang, IESSP: Information extraction-based sparse stripe pruning method for deep neural networks, *Sensors*, vol.25, no.7, 2025.
- [11] J. Xiang, S. Li, J. Chen, Z. Chen, T. Huang, L. Peng and Y. Liu, MaxQ: Multi-axis query for N:m sparsity network, *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp.15845-15854, 2024.
- [12] Y. Zhang, Y. Li, X. Wang, Q. Shen, B. Plank, B. Bischl, M. Rezaei and K. Kawaguchi, Finer-Cut: Finer-grained interpretable layer pruning for large language models, *arXiv Preprint*, arXiv: 2405.18218, 2024.
- [13] 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.10558-10578, 2024.

- [14] E. Frantar, S. Ashkboos, T. Hoefler and D. Alistarh, GPTQ: Accurate post-training quantization for generative pre-trained transformers, *International Conference on Learning Representations*, 2023.
- [15] K. Xie, C. Liu, X. Wang, X. Li, G. Xie, J. Wen et al., Neural network compression based on tensor ring decomposition, *IEEE Transactions on Neural Networks and Learning Systems*, vol.36, no.3, pp.5388-5402, 2025.
- [16] A. M. Mansourian, R. Ahmadi, M. Ghafouri, A. M. Babaei, E. B. Golezani, Z. Y. Ghamchi et al., A comprehensive survey on knowledge distillation, *arXiv Preprint*, arXiv: 2503.12067, 2025.
- [17] P. Lu, Y. Ding and C. Wang, Multi-small target detection and tracking based on improved YOLO and SIFT for drones, *International Journal of Innovative Computing, Information and Control*, vol.17, no.1, pp.205-224, 2021.
- [18] S. Han, J. Pool, J. Tran and W. Dally, Learning both weights and connections for efficient neural network, *Advances in Neural Information Processing Systems*, pp.1135-1143, 2015.
- [19] H. Li, A. Kadav, I. Durdanovic, H. Samet and H. P. Graf, Pruning filters for efficient ConvNets, *International Conference on Learning Representations*, 2016.
- [20] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan and C. Zhang, Learning efficient convolutional networks through network slimming, *Proc. of the IEEE International Conference on Computer Vision*, pp.2736-2744, 2017.
- [21] Y. He, G. Kang, X. Dong, Y. Fu and Y. Yang, Soft filter pruning for accelerating deep convolutional neural networks, *International Joint Conference on Artificial Intelligence*, pp.2234-2240, 2018.
- [22] Y. He, P. Liu, Z. Wang, Z. Hu and Y. Yang, Filter pruning via geometric median for deep convolutional neural networks acceleration, *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, pp.4340-4349, 2019.
- [23] Z. Liu, H. Mao, C. Wu, C. Feichtenhofer, T. Darrell and S. Xie, A ConvNet for the 2020s, *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, pp.11966-11976, 2022.
- [24] P. Wimmer, J. Mehnert and A. P. Condurache, Dimensionality reduced training by pruning and freezing parts of a deep neural network: A survey, *Artificial Intelligence Review*, vol.56, no.12, pp.14257-14295, 2023.
- [25] X. Ma, G. Fang and X. Wang, LLM-Pruner: On the structural pruning of large language models, *Advances in Neural Information Processing Systems*, vol.36, pp.21702-21720, 2023.
- [26] B. Wang and V. V. Kindratenko, RL-Pruner: Structured pruning using reinforcement learning for CNN compression and acceleration, *arXiv Preprint*, arXiv: 2411.06463, 2024.
- [27] H. Wang, S. Wang, W. Q. Zhang, H. Suo and Y. Wan, Task-agnostic structured pruning of speech representation models, *International Symposium on Computer Architecture*, pp.231-235, 2023.
- [28] M. Intraraprasit and O. Chitsobhuk, Filter pruning based on local gradient activation mapping in convolutional neural networks, *International Journal of Innovative Computing, Information and Control*, vol.19, no.6, pp.1697-1715, 2023.
- [29] G. Fang, X. Ma, M. Song, M. Mi and X. Wang, DepGraph: Towards any structural pruning, *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp.16091-16101, 2023.
- [30] H. Lin, Y. Peng, Y. Zhang, L. Bie, X. Zhao and Y. Gao, Filter pruning by high-order spectral clustering, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol.47, no.4, pp.2402-2415, 2025.
- [31] Z. Huang and N. Wang, Data-driven sparse structure selection for deep neural networks, *European Conference on Computer Vision*, 2017.
- [32] S. Lin, R. Ji, C. Yan, B. Zhang, L. Cao, Q. Ye et al., Towards optimal structured CNN pruning via generative adversarial learning, *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019.
- [33] M. Lin, R. Ji, Y. Wang, Y. Zhang, B. Zhang, Y. Tian et al., HRank: Filter pruning using high-rank feature map, *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.
- [34] G. Liu, K. Zhang and M. Lv, SOKS: Automatic searching of the optimal kernel shapes for stripe-wise network pruning, *IEEE Transactions on Neural Networks and Learning Systems*, vol.34, no.12, pp.9912-9924, 2023.
- [35] Y. Li, S. Gu, C. Mayer, L. Van Gool and R. Timofte, Group sparsity: The hinge between filter pruning and decomposition for network compression, *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp.8015-8024, 2020.
- [36] Y. He, X. Zhang and J. Sun, Channel pruning for accelerating very deep neural networks, *Proc. of the IEEE International Conference on Computer Vision*, pp.1389-1397, 2017.

Author Biography



Lili Song received the B.Sc. degree in Mathematics and Applied Mathematics from Shandong Normal University, China, in 2013; the M.Sc. degree in Operations Research and Cybernetics from Shandong Normal University, in 2016; and the Ph.D. degree in Applied Mathematics from the University of Coimbra, Portugal, in 2022. During her Ph.D., she visited the Department of Systems and Industrial Engineering, Lehigh University, USA, from 2018 to 2019.

Dr. Song is currently a lecturer at the School of Computer and Artificial Intelligence, Shandong Jianzhu University, China, where she also serves as the Head of the Department of Computer Science. She works at the Computational Intelligence Center in the team of Professor Yan Shi. Her research interests include computational intelligence, algorithm optimization, model compression, and data analysis. She has published many papers in international journals and conferences and holds three patents. She actively serves as a reviewer for several journals. Additionally, she holds the position of Academic Vice-Chair of the Jinan Young Computer Scientists & Engineers Forum.



Yongchao Liu received the B.S. degree in Computer Science and Technology from Shandong Jianzhu University, China, in 2024. During his undergraduate studies, he focused on embedded software development. He is currently pursuing the Master's degree in Software Engineering at Shandong Jianzhu University. His research interests focus on model pruning and time series prediction. Specifically, he specializes in stripe pruning for convolutional neural networks (CNNs), dedicated to exploring superior pruning criteria and optimized pruning structures.



Sumaira Hussain received the B.S. and M.S. degrees in Computer Science from Virtual University, Pakistan, in 2007 and 2014, respectively, and the Ph.D. degree in Computer Science from Shandong University, China, in 2022. She previously served at a renowned university in Karachi, Pakistan. Since 2022, she has worked as a lecturer and researcher in Shandong Jianzhu University, China. Her research interests include computer vision, pattern recognition, medical image processing, and artificial intelligence, with a specific focus on tumor segmentation using deep learning.



Haohan Cao received his B.S. degree in Data Science and Big Data Technology from Shandong Jianzhu University, China, in 2025. He is currently pursuing the M.S. degree in Computer Technology under the supervision of Lili Song at Shandong Jianzhu University. His research interests focus on large model parameter optimization and agent optimization.



Hang Qu obtained his B.S. degree in Internet of Things Engineering from Shandong Agricultural and Engineering University, China, in 2024. Throughout his undergraduate education, he concentrated on embedded systems and sensor networks, engaging in research projects including intelligent agricultural environmental monitoring. Presently, he is pursuing an M.S. degree in Computer Technology at Shandong Jianzhu University, China, under the supervision of Lili Song. His research interests mainly center around large model parameter optimization, specifically delving into lightweight algorithms such as pruning and quantization to strike a balance between performance and efficiency, along with agent optimization.



Xinfeng Liu received the Ph.D. degree in Materials Science and Engineering from Shandong University, China, in 2009. He is currently an associate professor with the School of Computer and Artificial Intelligence, Shandong Jianzhu University, China. His research interests include development of big data systems and the optimization of equipment operations.