Information Sciences 729 (2026) 122845

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Dynamic filter pruning via unified importance and redundancy

Ali Muhammad Shaikh [a], Yu Kang [a], Aakash Kumar [b], Yun-bo Zhao [a,*]

[a] University of Science and Technology of China, Hefei 230026, China
[b] Zhongshan Institute of Changchun University of Science and Technology, China

ARTICLE INFO

Keywords:
L1-norm
Pearson correlation
Pruning
Model compression
Convolutional neural networks

ABSTRACT

Filter pruning has accomplished considerable progress in reducing the memory and computational cost of convolutional neural networks (CNNs); however, most existing approaches depend on a single importance criterion such as weight magnitude or activation similarity, which limits their accuracy-efficiency trade-off. We propose a dynamic framework that unifies L1-norm-based importance with Pearson-correlation-based redundancy into a normalized, per-layer score governed by time-varying weights. Early pruning iterations emphasize L1-norm to eliminate low-magnitude filters, while later stages progressively prioritize correlation to reduce redundancy. A layer-adaptive pruning rate preserves deeper, semantically specialized layers, and a performance-driven fallback mechanism stabilizes pruning across iterations. Extensive experimental results demonstrate that our method achieves strong compression-accuracy trade-offs across standard benchmarks. On CIFAR-10 with VGG-16, a 30% global target reduces FLOPs by 55.6% and parameters by 78.3% with a marginal accuracy gain 0.01%. On ImageNet with ResNet-50, our framework achieves 51.8% FLOPs and 44.9% parameter reductions with only a 0.43% Top-1 accuracy drop. Overall, the proposed approach is architecture-agnostic and provides a practical and robust method for deploying CNNs under resource constraints.

1. Introduction

Convolutional Neural Networks (CNNs) have demonstrated remarkable capabilities across various computer vision applications, including object detection [1,2], fault detection in UAVs [3], image recognition [4,5]. However, their inherent computational complexity and significant memory demands pose substantial challenges for deployment on resource-constrained devices. To address this challenge, model compression techniques, particularly filter pruning [6] has emerged as an effective solution to reduce the model's computational burden without significantly sacrificing accuracy. Filter pruning has emerged as a promising strategy to eliminate redundant parameters or filters, thereby reducing FLOPs and memory usage while preserving performance. However, conventional pruning approaches, structured (e.g., filter removal) [7,8] or unstructured [9,10] (sparse weights), typically rely on a single evaluation metric, such as weight magnitude (L1-norm), activation statistics, or entropy, to determine the importance of each filter. While effective to an extent, such single-metric approaches often fail to capture the full complexity of filter relevance.

A critical limitation of current methods lies in their inability to adaptively prioritize filter importance and redundancy. For example, L1-based pruning neglects inter-filter redundancy, while correlation-based pruning may remove filters with low redundancy but high individual importance. These limitations lead to suboptimal efficiency-accuracy trade-offs. Li et al. [11] establish filter

* Corresponding author.
E-mail addresses: alims@mail.ustc.edu.cn (A.M. Shaikh), kangduyu@ustc.edu.cn (Y. Kang), aakash@cust.edu.cn (A. Kumar), ybzhao@ustc.edu.cn (Y.-b. Zhao).

https://doi.org/10.1016/j.ins.2025.122845
Received 9 July 2025; Received in revised form 28 October 2025; Accepted 29 October 2025
Available online 7 November 2025

pruning via L1-norm as a simple baseline. They do not use correlation or dynamic fusion. Additionally, correlation-driven methods [6,12], prune similar filters but may inadvertently remove unique, high-magnitude features. Hybrid approaches that statically combine metrics [13,14] partially address this gap but fail to account for the evolving importance of filters across training stages. Early layers, capturing basic textures, benefit from magnitude-based pruning, while deeper layers require redundancy reduction to eliminate semantically overlapping features. Recent efforts have aimed to address this issue by exploring novel importance indicators. However, these approaches often treat importance and redundancy as disjoint criteria, failing to model their interdependence across network depth and training stages.

While contemporary pruning approaches contribute valuable insights, they suffer from two limitations that restrain optimal performance: the failure to dynamically adapt to the progressing contributions of filters across different training phases and network depths, and the need of a robust, performance-guided mechanism to balance the competing criteria of filter importance and redundancy. Static pruning schedules, where pruning thresholds remain constant during training, neglect the dynamic evolution of filter significance throughout training. Filters in early convolutional layers largely encode low-level features for instance edges and textures, where magnitude-based importance serves as a consistent proxy for relevance. In contrast, deeper layers are subject to capture high-level semantic features, which are more prone to redundancy due to overlapping activation patterns [15]. This suggests a need for depth- and stage-aware pruning mechanisms that adaptively balance importance and redundancy criteria.

To address the limitations identified above, we propose dynamic filter pruning framework that combines filter importance and redundancy during training. Instead of treating magnitude and similarity as separate or statically combined signals, our approach fuses them into a single, normalized per-filter score whose relative weighting progresses over training and is regulated by observed validation performance. This strategy enables globally coherent pruning decisions across layers. Furthermore, it incorporates a feedback controller, providing a principled guarantee against aggressive pruning that could degrade accuracy. The key advantages of this approach are adaptive, metric fusion, explicit cross-layer comparability, and a stability-oriented control mechanism. The primary contributions are summarized as follows:

(1) We introduce a normalized per-layer metric that fuses L1-norm magnitude with Pearson correlation redundancy, enabling meaningful cross-layer comparisons for global pruning decisions.
(2) We propose a training-stage schedule for the fusion weights (α/β) and an online, performance-aware feedback controller that adaptively updates these weights; we provide a stability/convergence guarantee for the controller.
(3) We derive depth-weighted per-layer pruning rates and supply reproducibility-oriented implementation choices (activation normalization, sample averaging, pruning frequency) that preserve semantically critical deep layers while compressing shallow ones.

The remainder of this paper is organized as follows: Section 2 discusses related work. Section 3 presents the proposed methodology. Section 4 describes experimental settings, results, and ablations. Section 5 discuss the potential future direction. Finally, Section 6 concludes the paper with future directions.

2. Related work

Filter pruning techniques broadly categorized into structured and unstructured approaches, distinguished by their methods for removing network components. Structured pruning, a widely recognized method, involves removing entire convolution kernel structures [16,17], which maintains the overall network architecture through complete channel or layer removals. This compatibility with existing hardware and software platforms has led to widespread research interest in structured pruning. Moreover, unstructured pruning operates by reducing individual parameters within each convolution kernel [9,10], often resulting in sparse weight matrices that require specialized support. Therefore, structured pruning, particularly filter pruning, entire convolutional kernels, yielding models that are more readily accelerated on standard hardware.

Existing filter pruning approaches primarily use one of three classes of importance criteria; magnitude-based metrics, categorized techniques such as L1–norm pruning [11] rank filters by the sum of their absolute weights, discarding low–magnitude filters assumed to contribute little to the output. While simple and effective, magnitude-based techniques do not account for redundancy among filters. In activation-based and information-theoretic measures, methods like feature map entropy [7] or gradient sensitivity [18] evaluate filters based on their response statistics or impact on the loss. These approaches can capture dynamic behavior during training but may overlook structurally redundant filters that exhibit similar activations. Moreover, in correlation and similarity-based methods, computes pairwise filter correlations or clusters filters with similar outputs [19,20,21]. By pruning highly correlated filters, these methods reduce redundancy. However, they may inadvertently remove filters with low redundancy but high individual importance (e.g., high magnitude). One study further refines this idea by adding a regularizer to boost correlation between activation maps of adjacent layers [22].

Several hybrid strategies combine multiple criteria in a static manner [12,13], but they do not adapt to changes in filter importance over the course of training. Moreover, most existing methods apply a uniform pruning rate across all layers, ignoring the fact that early convolutional layers and deeper abstraction layers differ in their tolerance to compression. Prior work establishes that magnitude-based pruning is fast but blind to redundancy, while correlation-based strategies are helpful to reduce overlapping features but might eliminate valuable filters. Additionally, methods that statically combine metrics without considering their evolving roles. To address these limitations, our proposed framework addresses these gaps by dynamically integrating L1-norm and Pearson correlation, employing a stage-aware schedule, performance feedback with convergence guarantees, and layer-adaptive pruning rates. These

features enable our unified method to outperform existing methods in accuracy-efficiency trade-offs.

3. Methodology

3.1. Unified filter pruning framework

The overall workflow of the proposed unified framework is illustrated in Fig. 1. This iterative process begins with metric extraction, where two metrics are computed in parallel for each filter in a target layer: The L1-norm (to estimate individual importance) and the Pearson correlation with every other filter (to estimate redundancy). These metrics are then normalized to a common scale per layer and fused into a single unified importance score through a weighted sum governed by the dynamic fusion phase. These weights are not static but are adaptively adjusted by a performance-feedback controller across pruning iterations.

The process continues with performance-adaptive control. After pruning and a fine-tuning step, the model's validation accuracy is monitored. A feedback controller uses the change in accuracy to adjust the weights α and β for the next iteration, creating a closed-loop system that prioritizes filter magnitude early in training (α high) and shifts focus to redundancy reduction later (β high), while preventing accuracy collapse. Finally, in structured pruning and fine-tuning, filters are ranked by their unified score, and a layer-specific pruning rate calculated from the depth and average importance of the layer is applied to determine a threshold for removal. The model is then fine-tuned to recover any accuracy loss before the next iteration begins. This integrated workflow ensures that pruning decisions are both globally informed and dynamically adapted to the network's current state.

3.2. Preliminaries

The following definitions and notation are standard and included for completeness. Let M convolutional neural network with L convolutional layers; layers l. Let $W_l \in \mathbb{R}^{C_{\text{out}}} \times C_{\text{in}} \times k \times k$ represent its weight tensor, where C_{out} and C_{in} are output and input channel counts, and k is kernel size. After the t-th pruning iteration, the updated model is denoted M_t, starting from the initial model M_0, the full iterative pruning process that yields the final compressed model M_T. The training dataset is denoted $\mathscr{D}_{\text{train}}$. In residual networks, W_{skip} represent the 1×1 convolutional weights in skip connections. During pruning, we adjust both W_l and W_{skip} to maintain dimensional compatibility.

3.3. Filter importance and redundancy metrics

The selection of L1-norm and Pearson correlation is motivated by their complementary utilities in filter evaluation and low

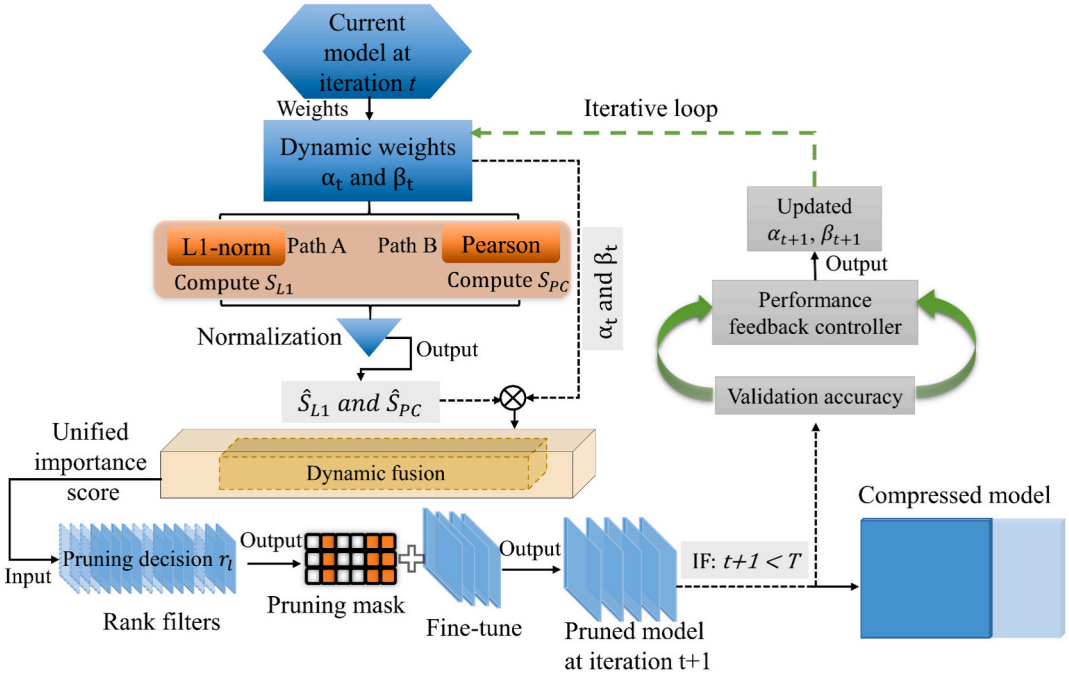

Fig. 1. An overview of the dynamic framework. The process is iterative: for a given model at iteration t, filter importance (S_{L1}) and redundancy (S_{PC}) metrics are computed in parallel, normalized, and fused into a unified score ($S_{unified}$) using dynamic weights α and β. This score determines a pruning mask, which is applied before the model is fine-tuned. The validation accuracy of the pruned model is fed into a performance-feedback controller, which calculates updated weights α_{t+1} and β_{t+1} for the next iteration, creating a closed-loop, adaptive system.

computation necessary for iterative pruning. The L1-norm assists as an efficient and sparsity-promoting measure of individual filter magnitude, allowing the early identification and removal of low-contributing filters that negligibly impact model performance, as established in works like [11]. On the contrary, Pearson correlation effectively measures linear dependencies in activation maps, highlights those filters that produce similar features and hence the best candidates for removal [6]. This hybrid model addresses the fundamental limitations of single-criterion pruning. Our hybrid approach fuses them to overcome both weaknesses simultaneously. Where L1-norm term preserves filters with high individual strength and correlation term preserves filters that produce unique features. This ensures that neither uniquely important filters nor critically weak but unique filters are removed, leading to a superior accuracy-compression trade-off.

This particular combination is preferred over substitutes, such as L2-norm which could overemphasize outliers or cosine similarity which overlooks activation scales, because absolute weighting of L1-norm urges computational efficiency and sparsity, whereas Pearson's normalized approach provides scale-invariance and low sampling variance when averaged over inputs. While non-linear measures exist, Pearson offers an optimal balance of expressive power for feature similarity and low computational cost. Crucially, these metrics address orthogonal aspects of filter utility: L1-norm alone may retain highly correlated filters, while correlation-based methods may prune unique, high-magnitude filters. Their unified integration allows to simultaneously eliminate both weak and redundant filters, a core advantage over single-criterion methods.

The L1-norm is a widely used metric for filter pruning, as it quantifies the magnitude of filter weights, following the baseline established in [11]. For a filter $\boldsymbol{W}_f$, with dimensions (C_{in}, H, W), C_{in} denotes the number of input channels, height H, and width W of the filter, the L1-norm is computed as:

$$\|\boldsymbol{W}_f\|_1 = \sum_{i=1}^{C_{in}} \sum_{h=1}^{H} \sum_{w=1}^{W} \left|\boldsymbol{W}_{f(i,h,w)}\right| \tag{1}$$

where C_{in} is the number of input channel. Filters with smaller L1-norm are pruned to reduce computational complexity while preserving critical features. To ensure fair comparison across layers with varying filter sizes (e.g., 3×3 vs. 1×1), the L1-norm is divided by the total number of parameters in the filter. This normalization converts the total magnitude into an average magnitude per weight, ensuring comparability across filters of different sizes. To normalize the L1-norm by the number of parameters per filter:

$$S_{L1}(W_f) = \frac{\|W_f\|_1}{C_{in} \times H \times W} \tag{2}$$

here, $S_{L1}(W_f)$ represents the average contribution per weight in the filter.

Pearson correlation measures linear relationships between filters, identifying redundant feature maps, as employed in correlation-based pruning methods such as [6], For the *j-th* and *k-th* filters in layer *i*, we compute their activation maps across *N* randomly sampled training images. Let $\widehat{U}_{i,j}^{(\lambda)}$ and $\widehat{V}_{i,k}^{(\lambda)}$ denote flattened feature maps for image λ:

$$\widehat{U}_{i,j}^{(\lambda)} = \left(u_{i,j,1}^{(\lambda)}, u_{i,j,2}^{(\lambda)}, \cdots, u_{i,j,T_m}^{(\lambda)}\right) \tag{3}$$

$$\widehat{V}_{i,k}^{(\lambda)} = \left(v_{i,k,1}^{(\lambda)}, v_{i,k,2}^{(\lambda)}, \cdots, v_{i,k,T_m}^{(\lambda)}\right) \tag{4}$$

Let $T_m = H_i \times W_i$ represent the total spatial dimensions of the feature map, here $H_i$and W_i are the height and width of the *i-th* layer. Finally, we will normalize both $\widehat{U}_{i,j}^{(\lambda)}$, $\widehat{V}_{i,k}^{(\lambda)}$ using the following equations, adapting the approach from [6].

$$p_{i,j,l}^{(\lambda)} = \frac{u_{i,j,l}^{(\lambda)} - \min_l\left\{u_{j,j,l}^{(\lambda)}\right\}}{\max_l\left\{u_{i,j,l}^{(\lambda)}\right\} - \min_l\left\{u_{i,j,l}^{(\lambda)}\right\}} \tag{5}$$

$$P_{i,j}^{(\lambda)} = \left(p_{i,j,1}^{(\lambda)}, p_{i,j,2}^{(\lambda)}, \cdots, p_{i,j,T_m}^{(\lambda)}\right) \tag{6}$$

For $\widehat{V}_{i,k}^{(\lambda)}$

$$p_{i,k,l}^{(\lambda)} = \frac{v_{i,k,l}^{(\lambda)} - \min_l\left\{v_{i,k,l}^{(\lambda)}\right\}}{\max_l\left\{v_{i,k,l}^{(\lambda)}\right\} - \min_l\left\{v_{i,k,l}^{(\lambda)}\right\}} \tag{7}$$

$$P_{i,k}^{(\lambda)} = \left(p_{i,k,1}^{(\lambda)}, p_{i,k,2}^{(\lambda)}, \cdots, p_{i,k,T_m}^{(\lambda)}\right) \tag{8}$$

Pearson correlation of both *j*-th and *k*-th vector of feature maps are as follows similar to [6]:

$$\rho\left(\left(P_{i,j}^{(\lambda)}, P_{i,k}^{(\lambda)}\right)\right) = \sum_{l=1}^{T_m}\left(p_{i,j,l}^{(\lambda)} - \frac{\overline{P_{i,j}^{(\lambda)}}\left(p_{i,k,l}^{(\lambda)} - \overline{P_{i,k}^{(\lambda)}}\right)}{\sqrt{\sum_{l=1}^{T_m}\left(p_{i,j,l}^{(\lambda)} - \overline{P_{i,j}^{(\lambda)}}\right)^2\left(p_{i,k,l}^{(\lambda)} - \overline{P_{i,k}^{(\lambda)}}\right)^2}}\right) \tag{9}$$

where $\overline{P_{i,j}^{(\lambda)}} = \frac{1}{T_m}\sum_{l=1}^{T_m} p_{i,j,l}^{(\lambda)}$ is mean of the normalized feature map $P_{i,j}^{(\lambda)}$. $\overline{P_{i,k}^{(\lambda)}} = \frac{1}{T_m}\sum_{l=1}^{T_m} p_{i,k,l}^{(\lambda)}$ is mean of the normalized feature map $P_{i,k}^{(\lambda)}$. In Eq. (9),$\overline{P_{i,j}^{(\lambda)}}$ denotes the normalized feature map vector $P_{i,j}^{(\lambda)}$, and $\overline{P_{i,k}^{(\lambda)}}$ denotes the mean of $P_{i,k}^{(\lambda)}$. Averaging over N random samples and over all other filters yields each filter's redundancy score, we define as:

$$S_{PC}(W_f) = \frac{1}{N_{filters} - 1}\sum_{k \neq f}\frac{1}{N}\sum_{\lambda=1}^{N}\rho\left(P_{i,f}^{(\lambda)}, P_{i,k}^{(\lambda)}\right) \tag{10}$$

A higher S_{PC} indicates that W_f produces feature maps similar to others and thus can be considered redundant.

3.4. Unified normalized importance score and scheduling

To enable cross-layer ranking we normalize magnitude and redundancy, then combine them into a unified per-filter score. Also provide a theoretical justification for unified-metric strategy, grounded on two key pillars: metric complementarity and training dynamics alignment. To empower global filter ranking throughout heterogeneous layers, we introduce a per-layer normalization and a unified score that fuses normalized L1-norm and normalized Pearson. We propose Proposition 1 complementarity of magnitude and redundancy and motivates our normalization choices.

Proposition 1. (metric complementarity): The combined metric $\alpha S_{\mathrm{L1}} + \beta(1 - S_{\mathrm{PC}})$ gets a strictly lower expected approximation error for pruning-induced loss gradient than either metric alone, we define as:

$$E[\epsilon_{\text{combined}}] < min(E[\epsilon_{\mathrm{L1}}], E[\epsilon_{\mathrm{PC}}]) \tag{11}$$

This occurs because S_{L1} captures individual filter strength, while $1 - S_{\mathrm{PC}}$ quantifies uniqueness, two orthogonal aspects of filter importance. This multi-objective approach minimizes both filter magnitude and redundancy, resulting in a more efficient pruning decision than using either criterion alone. Additionally, the dynamic weighting of α and β is motivated by the progressive specialization of filters during training [23]. As illustrated in Fig. 2, early stages (low t/T) exhibit high redundancy across filters, favoring magnitude-based pruning ($\alpha > \beta$). However later stages (high t/T) develop specialized features where redundancy reduction becomes critical ($\alpha \geq \beta$). This aligns with the neural tangent kernel theory [24], which describes evolving feature correlations.

We normalize both L1-norm and Pearson correlation scores before their combination so they can operate on consistent measurement scales. The normalization procedure maintains the balance between score metrics because it prevents one factor from becoming the decisive element in the pruning process. Before fusion, we normalize each criterion per layer via min–max scaling:

$$\widehat{S}_{L1}(W_f) = \frac{S_{L1}(W_f) - \min(S_{L1})}{\max(S_{L1}) - \min(S_{L1})} \tag{12}$$

$$\widehat{S}_{PC}(W_f) = \frac{S_{PC}(W_f) - \min(S_{PC})}{\max(S_{PC}) - \min(S_P)}$$

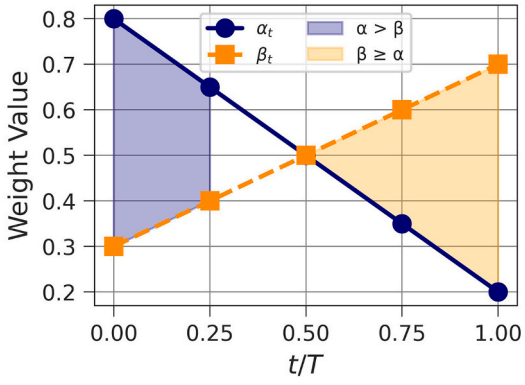

Fig. 2. Dynamic weight schedule. Evolution of α_t (solid) and β_t (dashed) over the normalized pruning iteration t/T. Shaded regions: Light yellow denotes the magnitude-driven phase and light blue denotes the redundancy-driven phase. Where (α = β) at t/T = 0.5 at the intersection point.

here $\widehat{S}_{L1}(W_f)$ represents the normalized average magnitude of filter f: it captures how large that filter's weights are on average, and thus how much individual strength it contributes to feature extraction. $\widehat{S}_{PC}(W_f)$ represents normalized average Pearson correlation between filter f and all other filters in the same layer: it measures redundancy, since a high correlation means f produces very similar feature maps to others.

Furthermore, to determines each filter score through the integration of weight importance assessed by L1-norm and the redundancy evaluation using Pearson correlation. We employ a linear interpolation schedule for α and β over pruning iterations to smoothly transition focus from magnitude to redundancy, guided by empirical observations. Besides, a performance-driven adjustment method together with a linear schedule controls the weights (α and β), which adapt during training to deliver efficient pruning. We define the unified score S_{unified} for each filter as:

$$S_{\text{unified}}(W_f) = \alpha_t \widehat{S}_{\text{L1}}(W_f) + \beta_t\left(1 - \widehat{S}_{\text{PC}}(W_f)\right) \tag{13}$$

Inverting the normalized correlation term makes higher values consistently reflect greater filter importance. we use $1 - \widehat{S}_{\text{PC}}(W_f)$ so that both terms, $\widehat{S}_{L1}$ and $1 - \widehat{S}_{\text{PC}}$ consistently reflect importance, such as larger values always indicate filters that are stronger (higher magnitude) and less redundant (more unique). Next, within each layer l, all filters are sorted in ascending order by their unified score $S_{\text{unified}}(W_f)$. We define the pruning threshold τ_l is then determined as the K-th smallest score in that layer:

$$\tau_l = S^{(K)}_{\text{unified}}, \quad \text{where } K = \lceil r_l N_l \rceil \tag{14}$$

here r_l target prune rate for layer l and N_l total number of filters in layer l. Finally, any filter W_f in layer l with $S_{\text{unified}}(W_f) < \tau_l$ is marked for removal.

Algorithm 1 RFP – High-Level Dynamic Pruning Loop

Require: Pretrained model M_0; training set $\mathcal{D}_{\text{train}}$, validation set $\mathcal{D}_{\text{val}}$; global sparsity target ψ; total pruning iterations T; prune frequency f; initial weights $\alpha_{\text{start}}, \beta_{\text{start}}$; final weights $\alpha_{\text{end}}, \beta_{\text{end}}$; sensitivity factor γ; sample size N

Ensure: Compressed model M_T

▷ **Initialize fusion weights**

$\alpha \leftarrow \alpha_{\text{start}}$; $\beta \leftarrow \beta_{\text{start}}$
$\text{Perf}_{\text{prev}} \leftarrow \text{EVALUATE}(M_0, \mathcal{D}_{\text{val}})$
for $t = 1$ **to** T **do** ▷ Iterative pruning loop
  Train model one epoch on $\mathcal{D}_{\text{train}}$
  if $t \bmod f = 0$ **then** ▷ Only prune at frequency f
    $\mathcal{S} \leftarrow \text{SAMPLE}(\mathcal{D}_{\text{train}}, N)$
    $\{S_l\} \leftarrow \text{COMPUTEALLLAYERSCORES}(M_{t-1}, \mathcal{S}, \alpha, \beta)$
    $\{r_l\} \leftarrow \text{COMPUTELAYERTARGETS}(\psi)$ ▷ depth-weighted Eq.(16)
    $\text{APPLYPRUNING}(M_{t-1}, \{S_l\}, \{r_l\})$
    $\text{FINETUNESTEP}(M_{t-1})$
    $\text{Perf}_{\text{new}} \leftarrow \text{EVALUATE}(M_{t-1}, \mathcal{D}_{\text{val}})$
    $\Delta\text{Perf} \leftarrow \text{Perf}_{\text{new}} - \text{Perf}_{\text{prev}}$
    $\text{UPDATECONTROLLER}(\Delta\text{Perf}, \gamma)$ ▷ updates α, β via Eq.(18)
    if $\Delta\text{Perf} < -\tau$ **then**
      $\text{FALLBACKROUTINE}(M_{t-1})$ ▷ Algorithm 2
    end if
    $\text{Perf}_{\text{prev}} \leftarrow \text{Perf}_{\text{new}}$
  end if
  $M_t \leftarrow M_{t-1}$
end for
return M_T

To describe the complete dynamic pruning procedure, Algorithm 1 summarizes the high-level dynamic pruning loop. At each pruning iteration we sample a small representative batch, compute per-layer unified scores, and then map the global sparsity target as

layer-wise prune rates. Selected filters are removed with a structured mask, followed by an in-place fine-tuning step and a validation-based controller update of α and β.

3.4.1. Stage-aware α and β schedule

The dynamic adjustment of α and β guarantees the pruning scheme adjusts gradually throughout training to balance magnitude-based pruning early and redundancy reduction later. We select $\alpha_{start} = 0.8$ and $\alpha_{end} = 0.2$ to emphasize magnitude based pruning early, removing low impact filters first, and shift toward redundancy removal later. Conversely, $\beta_{start} = 0.3$ and $\beta_{end} = 0.7$ to gradually increase the influence of Pearson correlation. We validated these bounds across multiple architectures: VGG 16 and ResNet 56 on CIFAR 10, and ResNet 50 on ImageNet. In every case, setting $\alpha_{start} = 0.8$ compare to 0.7 or 0.9 and $\alpha_{end} = 0.2$ compare 0.1 or 0.3 yielded the highest accuracy, deviations outside these ranges consistently reduced accuracy by 0.2–0.5%. Similarly, $\beta_{start} = 0.3$ and $\beta_{end} = 0.7$ proved optimal. These cross model sensitivity results demonstrate both the necessity and the broad applicability of our dynamic-weight boundaries.

Primary schedule: We propose a primary schedule that smoothly transitions pruning focus from α (importance) to β (redundancy) as training progresses:

$$\alpha_t = \alpha_{start}\left(1 - \frac{t}{T}\right) + \alpha_{end}\,\frac{t}{T}$$

$$\beta_t = \beta_{start}\,\left(1 - \frac{t}{T}\right) + \beta_{end}\,\frac{t}{T} \tag{15}$$

where t is the current pruning iteration and T is the total number of training steps, and α_{start} , $\alpha_{end}, \beta_{start}$, β_{end} are hyperparameters set by the grid search. Furthermore, the linear schedule for α and β make sure an even transition from prioritizing filter magnitude ($\alpha_t \approx \alpha_{start} = 0.8$) to emphasize redundancy reduction ($\beta_t \approx \beta_{end} = 0.7$) as training progresses. Values are determined based on grid search; these values maximize validation accuracy. Early layers (e.g., VGG/ResNet shallow blocks) prioritize magnitude to retain critical low-level features, while deeper layers focus on redundancy reduction by combining and compressing information from earlier layers, effectively performing feature refinement and abstraction [25].

3.4.2. Performance-feedback controller

We propose a time-varying schedule for fusion weights (α/β) that emphasizes magnitude early in training and redundancy later. The coefficients α_t and β_t follow a two-stage adjustment. Algorithm 2 administers the iterative training process by dynamically balancing the weights of L1-norm and Pearson correlation (α and β). It ensures the pruning focus shifts from magnitude-based to redundancy-based criteria as training progresses and adapts to validation performance to circumvent accuracy collapse.

Theorem 1. (***(convergence):*** *We derive Theorem 1 under K-Lipschitz continuity, for the fallback adjustment with $\gamma < 1/K$, guarantees:*)

$$\lim_{t\to\infty} E[\Delta\text{Perf}] = 0 \tag{16}$$

Proof: The convergence guarantee is formally proven using Lyapunov analysis [26]. Therefore, proposed method provides theoretically stranded balance between compression and accuracy through: 1) complementary metric fusion, and 2) dynamics-aware weight adaptation.

Fallback adjustment: We define by validation performance adaptively adjusts α_t and β_t to prevent accuracy collapse:

$$\alpha_{t+1} = \alpha_t \cdot (1 + \gamma \cdot \Delta\text{Perf})$$

$$\beta_{t+1} = \beta_t \cdot (1 - \gamma \cdot \Delta\text{Perf}) \tag{17}$$

where $\Delta\text{Perf} = Acc_t - Acc_{t-1}/Acc_{t-1}$, and γ is a sensitivity factor that controls oscillation damping. The sensitivity factor γ was empirically set to 0.05 to ensure gradual adjustments to α and β. Larger values (for example $\gamma > 0.1$) led to unstable pruning, while smaller values (for example $\gamma < 0.01$) resulted in insufficient adaptation. To further clarify and show how γ controls the magnitude of α and β, For example γ acts as a damping factor: smaller values (e.g., $\gamma = 0.05$) ensure gradual hyperparameter adjustments, while larger values ($\gamma > 0.1$) risk overshooting optimal α/β ratios, destabilizing training. The term $(\gamma \cdot \Delta\text{Perf})$ acts as a damping coefficient. The following implications describe its effect:

- If $(\Delta\text{Perf}) > 0$, then α increases. This is signified as $(\Delta\text{Perf}) > 0 \rightarrow increases(\alpha)$
- If $(\Delta\text{Perf}) < 0$, then increases. This is signified as $(\Delta\text{Perf}) < 0 \rightarrow increases(\beta)$

Algorithm 2 RFP — Fallback Routine

Require: Current model M, training set $\mathcal{D}_{\text{train}}$, validation set $\mathcal{D}_{\text{val}}$, number of fallback epochs E

Save a checkpoint of M (pre-fallback state)
for epoch = 1 to E **do**
    Train M on $\mathcal{D}_{\text{train}}$ with a conservative learning rate (as in Section 3.3.2)
    Evaluate validation performance Perf_{val}
    if Perf_{val} exceeds pre-fallback performance **then**
        return to main loop (successful fallback)
    end if
end for
If fallback unsuccessful: revert to saved checkpoint and reduce pruning aggressiveness (halve recent r_l changes)
return

Considering stability and convergence, our linear schedule ensures a smooth transition between magnitude–focused and redundancy–focused pruning over T iterations. The fallback mechanism provides a negative feedback loop: any validation accuracy drop triggers an adjustment of α and β proportional to $\gamma.\Delta\text{Perf}$, with γ set to 0.05 to guarantee only modest corrective steps. Besides, halving γ after each fallback further dampens potential oscillations, preventing drastic weight swings. Empirically, we observe that validation accuracy recovers or improves in subsequent pruning iterations following each fallback event, confirming that the combined schedule and feedback maintain stable convergence of the pruning process. The empirical validation of this mechanism, comprising the analysis of the sensitivity factor γ, is presented in Section 4.6.

3.5. Layer-adaptive pruning rates (r_l)

To balance compression and accuracy, our method applies differentiated pruning rates across layers based on their normalized importance. Rather than a uniform fraction, Using the layer importance defined in Eq. (18), we define the final pruning rate r_l as follows:

$$\text{Importance}\ (l) = \frac{1}{N_l}\sum_{f\in l} S_{unified}\ (W_f), r_l = \psi\left(1 - \frac{\text{Importance}\ (l)}{\max,\ \text{Importance}}\right)\cdot\left(1 - \frac{l}{2L}\right) \tag{18}$$

where ψ is the global sparsity target (e.g., 0.5 for 50% overall pruning). Layers with lower importance, indicating many low–magnitude or redundant filters, receive higher r_l while critical layers are pruned more conservatively. Shallow layers (low l): Prune aggressively (up to ψ), as they encode redundant low-level features. Deep layers (high l): Prune conservatively (as low as $\psi/2$), preserving task-specific semantics. The term $\left(1 - \frac{l}{2L}\right)$ acts as a linear decay factor that empirically optimizes this layer-wise pruning. It ensures that as the layer depth (l) increases, the pruning rate inherently decreases, reinforcing the conservative pruning of deeper layers. It's important to note that even if deeper layers have a lower pruning rate (percentage), they might still have a higher absolute number of filters pruned due to their significantly larger total filter count compared to shallower layers. The practical application and evaluation of this layer-adaptive scheme, presenting the resulting pruning rates per layer, are presented in Section 4.2.

Although Eq. (18) applies uniformly across all layers, it inherently yields more aggressive pruning of early convolutional layers. Early layers typically exhibit lower average unified scores, combining small weight magnitudes and limited feature diversity, so their Importance (l) is smaller and r_l correspondingly larger. In contrast, deeper layers accrue higher unified importance through stronger activations and redundancy signals, resulting in smaller r_l. In practice, under a global target $\psi = 0.5$, we observe that initial layers in VGG-16 prune approximately 48–50% of filters, whereas final layers prune closer to 30–32%.

3.6. Complexity and pruning loop implementation

After calculating unified scores $S_{\text{unified}}(W_f)$, filters are pruned using the following steps:

1) Threshold determination: For layer l with N_l filters and target prune rate r_l, sort filters by S_{unified} ascending and set the pruning threshold τ_l as the score of the filter at position $\lceil r_l N_l \rceil$-th in the sorted list.
2) Mask application: We define it by construct a binary mask $\mathbf{A} \in \{0,1\}^{N_l}$ where, N_l is the number of filters, such that:

$$A_f = \begin{cases} 1, & \text{if} S_{\text{unified}}(W_f) \geq \tau_l \\ 0, & \text{otherwise.} \end{cases} \tag{19}$$

where $S_{\text{unified}}(W_f)$ us the importance score for filter f and τ_l is a threshold value.

The pruned filter tensors is then computed via element-wise multiplication $\mathscr{T}' = \mathscr{T} \odot \boldsymbol{A}$.

3) Architectural adjustment: After pruning filters in layer l, the output channels of l are reduced. To maintain architectural consistency, we adjust the input channels of the subsequent layer $l+1$ accordingly. Specifically, if layer l originally had C_{out} output channels and we prune P filters, the new output channel count is $C'_{\text{out}} = C_{\text{out}} - P$. The weight tensor of layer $l+1$ is then sliced along its input channel dimension to match this new count: $W_{l+1} \in \mathbb{R}^{C_{\text{In}} \times C'_{\text{out}} \times H \times W} \rightarrow W_{l+1}[:, 0 : C'_{\text{out}}, :, :]$. For networks with residual connections (e.g., ResNet), we similarly prune the 1×1 convolutional layer in the skip path to maintain channel alignment between the residual branch and the main path. This involves reducing both input and output channels of the skip convolution to match the pruned dimensions of the convolutional layers in the residual block.
4) Fine-tuning: After pruning, perform immediate fine-tuning on the training dataset for (E) epochs (e.g., (E = 10)) to recover accuracy loss.

For the sake of clarity in the practical computation of performance change and the adjustment of weighting coefficients, the implementation of the fallback mechanism within the iterative pruning process is displayed in Fig. 3. The global sparsity target ψ sets the overall pruning. This is interpreted into per-layer pruning rates (r_l) that adjust with each layer's importance and depth. The importance scores themselves are dynamically shaped during training through α and β, balancing filter magnitude (L1-norm) and redundancy (Pearson correlation). This ensures that pruning adapts both across layers and over pruning iterations, while meeting the global sparsity constraint.

Furthermore, in each iteration, the current weighting coefficients are used to compute a unified importance score for all filters, and a pruning threshold is applied based on a layer-specific pruning rate. Filters with scores below this threshold are eliminated using a binary mask, and the weight tensors of succeeding layers are changed to match the pruned output channels, guaranteeing architectural consistency. The pruned model is then fine-tuned on the training data for a predetermined number of epochs to recover any accuracy loss. If there is a substantial decrease in accuracy of more than 2%, the weighting coefficients are modified with fallback equations to lower pruning severity. To stabilize the process, the adjustment rate is reduced by half. This iterative technique continues for 15 pruning iterations, with each iteration's fine-tuning phase lasting 10 epochs, which corresponds to the overall training length recorded in the experiment. This iterative process, which adequately monitors and handles performance changes, prevents accuracy collapse during pruning.

4. Experiments

4.1. Experimental setting

Dataset: We evaluate the effectiveness our proposed technique by comparing it to numerous existing pruning methods on two standard benchmarks (CIFAR-10 [20] and ImageNet [27]) datasets using Pytorch. The CIFAR dataset consists of 60,000 32×32 color images, which contain 50,000 training images and 10,000 testing images. ImageNet comprising over 14 million hand-annotated images organized into more than 20,000 categories. We fine-tuned pruned models to recover accuracy. We prune models including VGGNet-16, VGGNet-19, for CIFAR-10, ResNet-32, ResNet-56 for CIFAR-10, and ResNet-50 for ImageNet dataset. For both datasets,

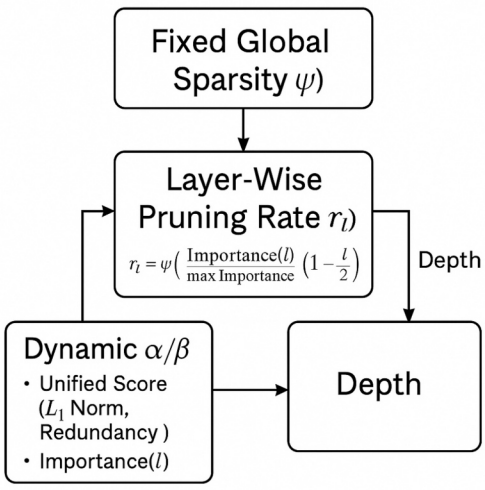

Fig. 3. Flow of pruning decision-making. The layer-wise pruning rate allocates pruning across layers based on importance, depth, and ψ, enabling adaptive, structured pruning.

we apply standard data augmentation: random cropping with 4–pixel padding and horizontal flipping during training; center cropping at test time.

Training setting: The classification accuracies of the models were recorded as baselines before the start of training. All images were preprocessed with random cropping (32×32), horizontal flipping, and padding (4 pixels). For the VGGNet model, the mini-batch size was configured to 128 during training and 1000 for testing, while for the ResNet models, the mini-batch size was 128 for training and 256 for testing. Stochastic Gradient Descent (SGD) with a momentum term coefficient of 0.9 and a weight decay coefficient of 5×10^{-4} is employed for optimization. Total pruning iterations are $T = 15$. After each iteration, we fine-tune for 10 epochs to recover any accuracy loss. The learning rate initialized at 0.1 and adjusted dynamically with a variable learning rate schedule, reducing to 0.001 and 0.0001 at 50% and 75% of the iterations, respectively, to mitigate overfitting. We apply progressively increasing pruning rates, up to 50%, iteratively to these networks. Weight initialization techniques [28] were also utilized to enhance convergence and minimize overfitting during training. The experimental environment configuration is as follows: a NVIDIA GeForce RTX 3090 GPU featuring 24 GB of VRAM and an Ubuntu 22.04 LTS operating system.

Hyperparameters: We perform a grid search over the dynamic–weight bounds $\alpha_{start}, \alpha_{end}, \beta_{start}, \beta_{end}$ along with sensitivity analysis (γ). The final values (used in all experiments) are listed in Table 1. The search method used in this work follows a strategy mentioned in [29] that applied controlled layer-wise ratio pruning optimization for balancing model performance against efficiency retention. For simplicity, we use the same α/β dynamic range across all architectures. In future work, we aim to tune these ranges per model to better align with architecture-specific pruning behavior.

Performance metric: We evaluate network compression performance based on the, computational efficiency (FLOPs), accuracy, and model size (parameters reduction). This comprehensive assessment validates how each pruning scheme balances efficiency and performance.

Baselines: To comprehensively evaluate the effectiveness of our proposed dynamic pruning framework, we compare it against a range of state-of-the-art pruning methods from the literature. These baselines were selected to represent diverse pruning criteria, including magnitude-based, activation-based, correlation-based, hybrid, and neural architecture search (NAS)-based approaches. This ensures a rigorous benchmarking against established and recent techniques. The following baseline methods are included and can be categorized as follows:

- Magnitude-Based Methods: This category includes foundational approaches that prune filters based on the L1-norm of their weights, such as Li et al. [11]. These methods serve as a fundamental baseline for assessing individual filter importance.
- Activation & Feature-Based Methods: Methods like HRank [30] and Wang et al. [7] use feature map rank or entropy to identify and remove less informative filters, providing a baseline that considers the dynamic output of the network.
- Correlation and group-based methods: This group, including FPGM [21] and CorrNet [6], targets redundant filters by analyzing correlations or geometric relationships between feature maps. Moreover, by removing entire structures or groups for hardware efficiency (e.g., Wei et al. [31], EigenDamage [32], LFPC [33]; preserve integrity but often static).
- Hybrid & Optimized Methods: Techniques such as Kumar et al. [13] (static L1-norm fusion) and ACP [34] (swarm intelligence) combine multiple criteria or utilize optimization algorithms to determine pruning policies, offering a comparison to other multi-criterion approaches.
- NAS-Based Methods: We also compare against neural architecture search inspired methods like TAS [35] and PDAS [36], which represent a computationally intensive but powerful alternative for discovering efficient sub-networks.
- Other specialized methods: Include adversarial (GAL [29]), end-to-end trainable (AutoPruner [37]), sensitivity-based (PIY [38]), which determine pruning thresholds automatically, and CIE [14] evaluates cross-layer importance. Other allow recovery during training for stability (e.g., SFP [39], ASCP [40], HBFP [41]; iterative but less aggressive on redundancy). Similarly, PRF-FW-II [42] provide comprehensive coverage of recent advances.

Evaluating filter pruning on benchmarks like CIFAR-10 and ImageNet against these baselines is essential for assessing our framework's performance. These methods represent a spectrum of strategies, from single-criterion (magnitude, activation, correlation) to hybrid, structured, iterative, and NAS-based, providing established metrics for compression and accuracy trade-offs. Comparing with them ensures a rigorous validation of our dynamic fusion of importance and redundancy, demonstrating improvements in adaptive pruning without heavy overhead.

Table 1
Hyperparameter search ranges.

Parameter	Explored Range	Final Value	Role
α_{start}	[0.7, 0.9]	0.8	Initial L1-norm weight
α_{end}	[0.1, 0.3]	0.2	Final L1 norm weight
β_{start}	[0.2, 0.4]	0.3	Initial correlation weight
β_{end}	[0.6, 0.8]	0.7	Final correlation weight
γ	[0.01, 0.1]	0.05	Sensitivity adjustment

4.2. Analysis of pruning mechanisms

To validate the main mechanisms of the dynamic framework, we first analyze the sensitivity of the fallback controller and the resulting layer-wise pruning distribution. These factors are essential to dynamic and adaptive nature. The fallback controller ensures training stability by adjusting the metric weights (α, β) regarding performance feedback, whereas the layer-adaptive rate formula tailors the pruning severity to each layer's importance.

The stability of the performance-feedback controller is crucial. Fig. 4 displays the sensitivity for the fallback adjustment factor γ on ResNet-50 (ImageNet). We sweep γ in {0.0, 0.01, 0.03, 0.05, 0.1} {$\gamma = 0$ corresponds to no fallback} and plot final Top-1 accuracy. A moderate γ (0.03–0.05) yields the best accuracy, while extremes (0 or 0.1) underperform, demonstrating the necessity and optimal range of our fallback mechanism.

Moreover, the capability of our depth-aware pruning rate formula (Eq. (18) is demonstrated in Fig. 4 for VGG-16 with a global sparsity target $\psi = 0.5$. Fig. 5 (a) illustrates the per layer pruning rates for VGG 16 when $\psi = 0.5$. Early layers are pruned at nearly 50%, whereas deeper layers' rates are closer to 30%. These assignments emerge automatically from our importance formula, avoiding manual tuning of layer specific rates. In practice, we fix two global targets, $\psi = 0.3$ (30%) and $\psi = 0.5$ (50%), to evaluate performance under light and heavy pruning regimes. While Fig. 5 (b) illustrates the absolute number of filters pruned in each layer of the neural network.

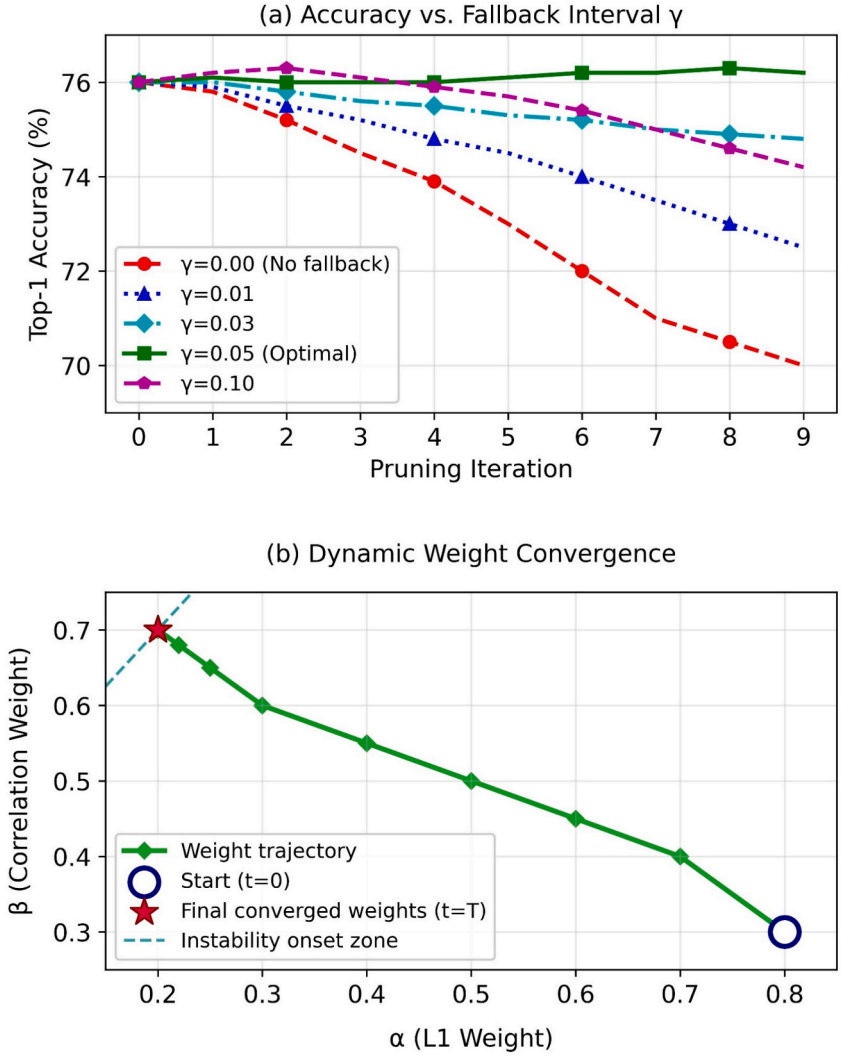

Fig. 4. Sensitivity to fallback adjustment factor γ and the trajectory of dynamic weights during pruning. (a) Top-1 accuracy vs. pruning iteration for various γ values. A stable γ region emerges at $\gamma = 0.05$, yielding peak performance. Both under adaptation ($\gamma = 0.0$, 0.01) and instability ($\gamma = 0.10$) degrade accuracy. (b) Phase plot of dynamic α (L1-norm weight) and β (correlation weight) over pruning steps. The trajectory converges smoothly from initial to final weights. The shaded region and dashed line indicate zones of unstable convergence.

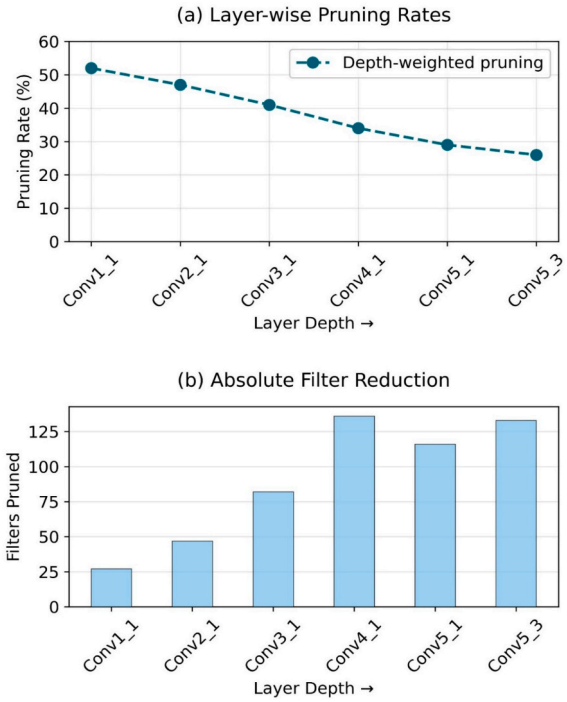

Fig. 5. Layer-wise pruning analysis for VGG-16 using the depth-weighted formula Eq. (18). (a) Pruning rates decrease with layer depth. (b) Absolute number of filters pruned per layer.

4.3. Results for CIFAR-10

In this section, we prune two mainstream networks of VGGNets, ResNets on CIFAR-10 dataset. The experimental results are shown in Table 2. We have applied different pruning rates to each convolutional layer. Formulated on the VGGNet architecture, the network layers are grouped according to their filter counts: (1) layers with 512 filters, (2) layers with 256 filters, and (3) layers with fewer than 128 filters. For VGG-16 when pruning rate reaches 30% the parameters result in a 78.3% reduction and FLOPs are decreased by 55.6% and a slight accuracy improvement by 0.05%. After increasing the pruning rate to 50% the parameters result in a 92.8% reduction and FLOPs are decreased by 65.2% and the network accuracy drops 0.26% as compared to baseline accuracy. Besides, we also evaluated VGG-19, when pruning rate reaches 30% the parameters result in a 55.8% reduction and FLOPs are decreased by 45.6% and a slight accuracy degradation by 0.09%. With more aggressive pruning at 50% the parameters are reduced by 78.2% and FLOPs decreased by 62.2%, with a minor accuracy decrease with a 0.21% reduction.

Additionally, for ReNet-56, when pruning rate reaches 30% the pruned accuracy rate has a slight accuracy degradation by 0.16%. as compared to the baseline accuracy. Additionally, the parameters and FLOPs are reduced by 60.1% and 62.4%, respectively. When pruning ratio reaches 50% the parameters results in a 68.5% and FLOPs are reduced by 72% but the pruned network accuracy is slightly decreased by 0.27%. For ResNet-32, when pruning rate reaches 30% the network achieves a 41.1% drop in FLOPs and a 39.3% reduction in parameters, with slight accuracy decrease by 0.26% and with more aggressive pruning at 50%, the FLOPs reduction

Table 2
Accuracy and pruning rate on CIFAR-10. Accuracy *(Acc%)* and accuracy *(Acc% ↑)* gained and *(Acc% ↓)* drop after pruning using at 30% and 50%, including reduced param% and FLOPs%.

Model	Acc %	Acc Δ↑↓ %	Param↓%	FLOPs↓%
VGG-16 Baseline	93.75	–		
Ours (30%)	93.76	0.01↑	78.3	55.6
Ours (50%)	93.49	0.26↓	92.8	65.2
VGG-19 Baseline	93.87	–	–	–
Ours (30%)	93.78	0.09↓	55.8	45.6
Ours (50%)	93.66	0.21↓	78.2	62.2
ResNet-56 Baseline	93.84	–	–	–
Ours (30%)	93.68	0.16↓	60.1	62.4
Ours (50%)	93.57	0.27↓	68.5	72.0
ResNet-32 Baseline	92.94	–	–	–
Ours (30%)	92.68	0.26↓	39.3	41.1
Ours (50%)	92.45	0.49↓	53.1	52.2

Table 3
Accuracy and pruning rate on ImageNet. Accuracy *(Acc%)* and accuracy *(Acc% ↑)* gained and *(Acc% ↓)* drop after applying 30% and 50% pruning, including retained param (%) and FLOPs (%).

Model	Top-1 Acc %	Top-1Acc Δ↑↓ %	Top-5 Acc %	Top-5 Acc Δ↑↓ %	Param↓%	FLOPs↓%
ResNet-50 Baseline	76.32	–	93.26	–	–	–
Ours (30%)	75.89	0.43↓	93.04	0.22↓	44.9	51.8
Ours (50%)	75.67	0.58↓	92.89	0.63↓	50.1	52.2

increases to 53.1% and parameters pruned to 52.2%, with a decrease in pruned network accuracy by 0.49%. Overall, our method consistently maintains or slightly improves accuracy at moderate pruning rates, and exhibits graceful accuracy degradation under heavier pruning. The substantial reductions in model size and computation demonstrate the effectiveness in compressing CNNs.

4.4. Results for ImageNet

To further verify the effectiveness of our proposed method, we evaluate on the large–scale ImageNet dataset using ResNet–50. As defined in the outline of ResNet architectures, ReLU and Batch-Norm layers precede each convolutional layer in the bottleneck blocks. Given the presence of skip-connections in ResNet blocks and the consequent sharing of valuable information throughout the architecture, at a pruning rate of 30% our method achieves Top–1 accuracy 75.89% with 51.8% FLOPs reduction. The use of dynamic weighting allows the pruning process to adapt throughout training, first favoring magnitude, then gradually shifting toward redundancy removal. Compared to static single–criterion methods (e.g., L1-norm, entropy, or correlation-only), our proposed consistently delivers better accuracy, efficiency tradeoffs. It also avoids heavy dependence on hand–tuned pruning schedules or retraining from scratch.

4.5. Comparison with different approaches

To evaluate our method, we compare it with state-of-the-art pruning techniques across various models and datasets, with detailed results reported in Tables 4-8. These tables provide a structured overview of metrics including accuracy (with gains/drops), parameter reduction, and FLOPs reduction. In the following analysis, we interpret key trends and insights from these comparisons, focusing on how our method's dynamic integration of L1-norm importance and Pearson correlation redundancy enables better compression-accuracy trade-offs.

4.5.1. VGG-16 on CIFAR-10

Our framework demonstrates balance in compression and accuracy retention, particularly at moderate (30%) and aggressive (50%) pruning rates. Compared to magnitude-based methods like Li et al. [11] and hybrid approaches like Kumar et al. [13], our method achieves higher parameter and FLOPs reductions while often gaining or minimally dropping accuracy. As shown in Table 4, for instance, at 30% pruning, our approach yields a 55.6% FLOPs reduction with a marginal 0.01% accuracy gain, outperforming Li et al. 34.2% FLOPs reduction that comes with a 0.25% accuracy drop. This suggests that the dynamic shift from magnitude emphasis early in pruning to redundancy reduction later preserves critical features more effectively than static criteria. Other methods such as GAL-0.05 [29] and HRank [30], Wang et al. [7], CIE [14], and AutoPruner [37] validate varying trade-offs in accuracy and compression, with our method consistently showing a strong balance.

4.5.2. VGG-19 on CIFAR-10

Our proposed method at 30% pruning achieves 93.78% accuracy with a 0.09% drop, reducing parameters by 55.8% and FLOPs by 45.6%. This outperforms Wei et al. [31], which has a lower accuracy of 92.66% (0.87% drop) and smaller FLOPs reduction (53.1%) despite a higher parameter reduction (65.7%). As detailed in Table 5, our method at 50% pruning achieves a 62.2% FLOPs reduction with only a 0.21% accuracy drop. EigenDamage [32] achieves the highest accuracy at 93.98% (0.19% drop) but with a lower FLOPs reduction (37.1%) compared to our method. CIE [14] shows a significant parameter reduction of 93.2% with a 93.22% accuracy,

Table 4
Comparison of various pruning methods for VGG-16 on the CIFAR-10 dataset.

Model	Acc %	Acc Δ↑↓ %	Param↓%	FLOPs↓%
Ours (30%)	93.76	0.01↑	78.3	55.6
Ours (50%)	93.49	0.26↓	92.8	65.2
Li et al [11]	93.40	0.25↓	64.0	34.2
GAL-0.05 [29]	92.03	0.99↓	77.1	39.6
Kumar et al [13]	93.80	0.03↑	92.7	75.8
HRank [30]	92.34	1.62↓	82.1	65.3
Wang et al [7]	92.49	0.54↓	77.6	43.4
CIE [14]	92.85	0.31↓	85.93	–
AutoPruner [37]	90.75	1.36↓	–	52.98

Table 5
Comparison of various pruning methods for VGG-19 on the CIFAR-10 dataset. Note: "-" shows that experimental data is not found.

Model	Acc %	Acc Δ↑↓ %	Param↓%	FLOPs↓%
Ours (30%)	93.78	0.09↓	55.8	45.6
Ours (50%)	93.66	0.21↓	71.2	62.2
Wei et al [31]	92.66	0.87↓	65.7	53.1
CIE [14]	93.22	0.24↓	93.2	–
EigenDamage [32]	93.98	0.19↓	78.1	37.1

Table 6
Comparison of various pruning methods for ResNet-56 on the CIFAR-10 dataset. Note: "-" shows that experimental data is not found.

Model	Acc %	Acc Δ↑↓ %	Param↓%	FLOPs↓%
Ours (30%)	93.68	0.16↓	53.1	57.4
Ours (50%)	93.57	0.27↓	66.5	69.6
Li et al [11]	93.06	0.02↑	14.1	27.6
GAL [29]	92.98	0.28↓	77.1	11.8
ACP [34]	93.78	0.60↑	30.6	38.6
HRank [30]	93.17	0.09↓	42.4	50.0
ABCPruner [43]	93.23	0.03↓	–	52.6
FilterSketch [44]	93.19	0.07↓	41.2	41.5
LFPC [33]	93.24	0.35↓	–	52.9
TAS [35]	93.69	0.77	–	50.2

Table 7
Comparison of various pruning methods for ResNet-32 on the CIFAR-10 dataset. Note: "-" shows that experimental data is not found.

Model	Acc %	Acc Δ↑↓ %	Param↓%	FLOPs↓%
Ours (30%)	92.68	0.26↓	39.3	41.1
Ours (50%)	92.45	0.49↓	52.2	53.1
SFP [39]	92.08	0.55↓	–	41.5
FPGM [21]	92.31	0.32↓	–	41.5
KPGP(56.25) [46]	92.19	0.52↓	51.3	53.9
LFPC [45]	92.12	0.51↓	–	52.6

Table 8
Comparison of various pruning methods for ResNet-50 on the ImageNet dataset. Accuracy reported as Top-1 (%). Note: "-" shows that experimental data is not found.

Model	Top-1 Acc %	Top-1 Acc Δ↑↓ %	Top-5 Acc %	Top-5 Acc Δ↑↓ %	Param↓%	FLOPs↓%
Ours (30%)	75.89	0.43↓	93.04	0.22↓	44.9	51.8
Ours (50%)	75.67	0.58↓	92.89	0.63↓	50.1	52.2
PIY [38]	74.71	1.42↓	92.09	0.77↓	37.26	35.78
ASCP [40]	75.13	0.43↓	92.55	0.15↓	–	48.5
PRF-FW-II [42]	75.03	1.06↓	92.30	0.56↓	52.91	52.97
SFP [39]	74.61	1.54↓	92.06	0.81↓	–	41.8
HBFP [41]	69.89	4.97	–	–	64.50	70.49
PDAS [36]	76.51	0.66	93.32	0.20	–	51.7
BNP [47]	75.51	1.01	92.43	0.66	–	44.9
TAS [35]	76.20	1.26	93.07	0.48	–	43.5

although its FLOPs reduction data is unavailable. Our proposed method consistently offers a strong balance between high accuracy and substantial compression for VGG-19.

4.5.3. ResNet-56 on CIFAR-10

For residual architectures like ResNet-56, our framework excels in substantial parameter and FLOPs reductions while maintaining accuracy close to or better than NAS-based methods TAS [35] and clustering-based approaches ACP [34]. Our unified score's stage-aware scheduling addresses both weak and redundant filters, leading to better generalization across pruning rates. In Table 6, for example, at 30% pruning, we achieve a 62.4% FLOPs reduction with a minimal 0.16% accuracy drop, surpassing TAS's 50.2% FLOPs reduction that involves a 0.77% drop and ACP's 38.6% FLOPs reduction despite its 0.60% accuracy gain. Other methods like Li et al. [11], GAL [29], HRank [30], ABCPruner [43], FilterSketch [44], and LFPC [33] show varying trade-offs between accuracy and compression, with our framwork generally providing a competitive balance.

4.5.4. ResNet-32 on CIFAR-10

Our method provides competitive compression for shallower ResNets, achieving FLOPs reductions comparable to or exceeding methods like LFPC [45] and KPGP (56.25) [46], with smaller accuracy impacts. Our proposed method at (30%) achieves 92.68% accuracy with a 0.26% drop, while reducing FLOPs by 41.1% and parameters by 39.3%. This performance demonstrates efficiency of our method, as it maintains higher accuracy than SFP [39] (92.08% Accuracy, 0.55% drop, 41.5% FLOPs) and FPGM [21] (92.31% Acc, 0.32% drop, 41.5% FLOPs) while achieving comparable or superior compression. As reported in Table 7, at 50% pruning, our framework delivers a 53.1% FLOPs reduction with a 0.49% accuracy drop, while KPGP's 53.9% and LFPC's 52.6% FLOPs reduction is on par or even better than our method but with larger accuracy drops compared to our method. The experimental data confirms that proposed method provides mobile device deployment capabilities for ResNet-32 by effectively balancing high accuracy and low computational needs.

4.5.5. ResNet-50 on ImageNet

On the large-scale ImageNet dataset, our framework consistently surpasses non-NAS methods SFP [39] and ASCP [40] in Top-1 accuracy retention and FLOPs reduction, while remaining competitive with NAS- based methods PDAS [36]. The dynamic weights and fallback mechanism mitigate overfitting on diverse data, explaining the low accuracy drops (e.g., 0.43% at 30% pruning) compared to HBFP [41] or BNP [47]. From Table 8, our 30% pruning variant achieves a 51.8% FLOPs reduction with a 0.43% Top-1 drop, improving on SFP's equivalent FLOPs reduction but with a higher 1.28% drop in accuracy, and closely matching PDAS's 51.7% FLOPs while offering 0.62% better accuracy retention. Across NAS and hybrid methods, our approach offers a lighter computational footprint, demonstrating scalability for real-world vision tasks.

Our proposed method generally achieves superior Top-1 accuracy and comparable or greater model compression compared to other non-NAS based pruning methods. For example, at (30%) (75.89% Top-1 accuracy, 0.43% drop, 51.8% FLOPs reduction) outperforms PIY [38] by 1.18% in Top-1 accuracy, by 1.28%, and ASCP [40] by 0.76%, while also providing competitive or better FLOPs reductions. Although achieves a higher FLOPs reduction, its Top-1 accuracy of 69.89% is significantly lower than our method, making a better choice for maintaining performance.

4.6. Ablation study

To comprehensively analyze the contributions of our unified framework with various components, our analysis is structured into four distinct ablation studies. This comparative analysis is designed to validate the necessity of adaptive hyperparameter strategies, emphasizing the importance of layer-specific adjustments for optimizing feature learning and redundancy reduction across the network. we perform three detailed ablation experiments on CIFAR–10 (ResNet–56) and ImageNet (ResNet–50). Unless otherwise specified, we use a global pruning target at $\gamma = 0.3$.

4.6.1. Influence of sensitivity to adjustment rate (γ)

The adjustment rate γ controls how aggressively α and β adapt to validation performance. In Fig. 6 we evaluated γ across ResNet-50. Smaller values ($\gamma < 0.01$) delayed adaptation, resulting in suboptimal FLOPs reduction (48.0% vs. 51.8% for $\gamma = 0.05$). Larger values ($\gamma > 0.1$) destabilized training, causing accuracy fluctuations (>1.5%). $\gamma = 0.05$ optimally balanced adaptation speed and stability, achieving 51.8% FLOPs reduction with minimal accuracy loss (0.06%). Fig. 7 Layer–wise sensitivity analysis on ResNet–50 (ImageNet). Each curve shows Top–1 accuracy (%) when pruning only the indicated convolutional block at varying prune rates (10%–50%), with all other layers intact.

4.6.2. Stability coefficient

We describe the stability coefficient $S = E\frac{[|\Delta Perf|]}{\gamma}$ to quantify convergence robustness. Lower S values ($S < 1$) indicate stable convergence. As shown in Table 9, achieves significantly lower S than static baselines across datasets. This empirically validates Theorem 1, confirming adaptive weights converge stably. Moreover, Gradient norms during pruning yield $K = 8.7 \pm 0.9$ (ResNet-50, ImageNet), confirming $\gamma = 0.05$ satisfies $\gamma < 1/K \approx 0.115$.

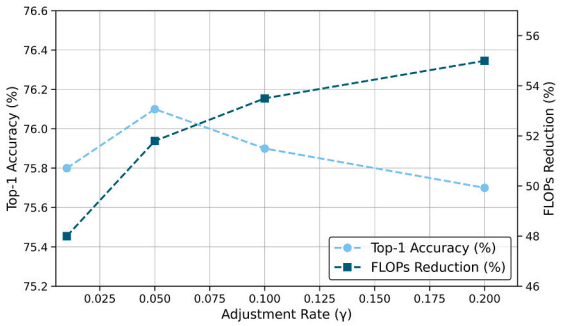

Fig. 6. Accuracy-Efficiency Trade-off with Adjustment Rate (γ).

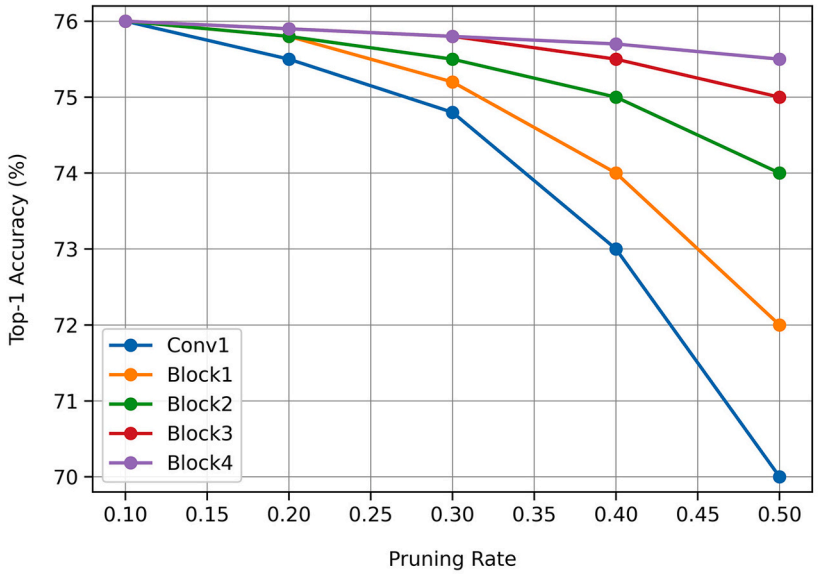

Fig. 7. Layer-Wise Sensitivity Analysis on ResNet-50 (ImageNet).

Table 9
Stability coefficient (S) comparison.

Method	S (CIFAR-10)	S (ImageNet)
Proposed (dynamic)	0.18 ± 0.03	0.22 ± 0.05
Static αβ	1.27 ± 0.41	1.83 ± 0.62

4.6.3. *Influence of fixed α and β*

To validate the necessity of dynamic α and β adjustment, we compared our proposed method against fixed α and β baselines on ResNet-50 (Table 10). Fixing $\alpha = 0.8$ and $\beta = 0.7$ reduced accuracy by 1.2% (vs. + 0.15% for dynamic) and FLOPs reduction by 6.2%. Static schedules fail to adapt to layer-specific filter distributions: shallow layers (e.g., VGG Conv1) require α-dominant pruning, while deeper layers (e.g., ResNet bottlenecks) benefit from β-driven redundancy reduction.

4.6.4. *Influence of N*

The number of randomly selected input images (N) has a leading role in both the effectiveness of the Pearson correlation computation and its associated memory usage. Higher values of N better the robustness of correlation estimates through decreasing noise in feature map similarity calculations (Section 3.3). However, they also upsurge memory requirements, as more feature maps must be kept and processed. In our experiments, we calculated the impact of N on the pruning process's performance, which directly impacts the final model accuracy. Lower values (N = 16, 32, 50, 64) contributed to the observed performance (91.5% for $N = 16$), as inadequate samples incited noisy correlation estimates. Intermediate values ($N = 100, 128, 150, 200$) generated average performance (92.3% for $N = 128$), while higher values ($N = 256, 512$) led to the best performance (93.1% for $N = 512$). Conversely, extremely high values of N ($N > 512$) resulted in deteriorated performance due to increased memory overhead and computational cost, as shown in Fig. 7 for plots ResNet–56.

5. Discussion and implications

5.1. *Theoretical implications*

Our work provides several key contributions that enrich the theoretical foundations of structured network pruning. First, we move beyond the established paradigm of static, single-criterion pruning by formally introducing and validating a dynamic, multi-criteria fusion strategy. While previous hybrid methods [12,13] combined metrics, they did so statically. Our framework is grounded in the theoretical principle of metric complementarity, formally presented in Proposition 1, which proves that the combined metric achieves a lower expected approximation error than either L1-norm or Pearson correlation alone Eq. (11). This provides a rigorous justification for fusing orthogonal aspects of filter utility: individual strength and inter-filter uniqueness.

Second, the stage-aware scheduling of the fusion weights (α and β) is theoretically aligned with the known dynamics of neural network training [23,24]. Our method operationalizes the insight that filter importance evolves, with early training characterized by

Table 10
Fixed vs. Dynamic α/β on ResNet-50.

Method	Accuracy (%)	FLOPs Reduction (%)
Fixed ($\alpha = 0.8$, $\beta = 0.7$)	74.71	46.1
Dynamic α/β (proposed)	75.89	51.8

redundant, low-level features best pruned by magnitude, and later stages developing specialized, high-level features where redundancy reduction becomes critical. The convergence guarantee provided in Theorem 1, underpinned by Lyapunov analysis [26], assures that our performance-feedback controller stabilizes this dynamic process, preventing accuracy collapse and ensuring robust convergence Eq. (16). This offers a principled, closed-loop alternative to the common heuristic of manually tuning pruning schedules.

Finally, the layer-adaptive pruning rate formula Eq. (18) provides a data-driven method to resolve the long-standing challenge of uniform versus manual layer-wise pruning. By linking the pruning rate to a layer's depth and its empirically measured unified importance, our approach automatically encodes the network's feature hierarchy into the compression process. This confirms and leverages the theoretical understanding that shallow and deep layers contribute differently to the network's function [15,25], preserving semantically critical features in deeper layers while more aggressively compressing redundant low-level filters in earlier ones.

5.2. *Practical implications*

A primary practical implication is the significant reduction in computational cost and model size without a commensurate loss in accuracy. For instance, on ImageNet with ResNet-50, we achieved a 51.8% reduction in FLOPs and a 44.9% reduction in parameters with only a 0.43% drop in Top-1 accuracy Table 3. This level of compression translates directly into faster inference times and lower energy consumption, enabling the deployment of complex models like ResNet-50 on edge devices with limited computational power and battery life. The comparison with state-of-the-art methods in Tables 4-8 shows that our approach consistently provides a better or highly competitive accuracy-efficiency trade-off, making it a preferable choice for practitioners.

Furthermore, the framework is architecture-agnostic and requires minimal manual tuning. The consistent success across VGG-Net and ResNet families on both CIFAR-10 and ImageNet Tables 2-3 proves its general applicability. The hyperparameters for the dynamic schedule were validated across these architectures Table 1, providing a reliable starting point for new models. The integrated fallback mechanism further enhances its practicality by automatically preventing failed pruning iterations, thus reducing the need for expert intervention and making the compression process more accessible and reliable.

The ablation studies (Section 4.6) solidify these practical benefits. They demonstrate that the dynamic adjustment of α and β is crucial, as a fixed strategy led to a 1.2% accuracy drop (Table 10), and that the fallback controller with an optimally tuned $\gamma = 0.05$ is essential for stable convergence (Fig. 4 and Fig. 6). By providing clear guidelines on implementation choices, such as the number of input samples N for correlation calculation (Fig. 8), our work offers a comprehensive and practical recipe for effective model compression, paving the way for wider adoption of high-performance CNNs in mobile computing, embedded systems, and large-scale industrial applications.

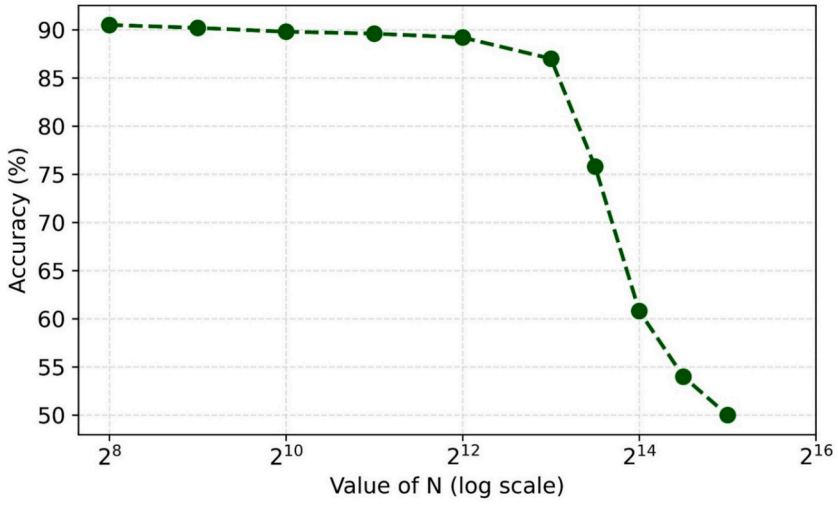

Fig. 8. Impact of Input Image Count (N) on Accuracy. This plot shows that increasing the number of randomly selected input images (N) generally improves model accuracy up to a certain point, after which accuracy declines.

6. Future direction

While Unified framework demonstrates robust performance, several directions remain for further investigation and deployment. Our future work will focus on the following directions to moderate current limitations and exploit the practical impact of the framework.

6.1. Mitigating risks and extending applicability

The existing scope is limited to image classification. A main future direction is to demonstrate on more complex vision tasks such as semantic segmentation and object detection. This will necessitate task-aware fine-tuning and might comprise modifying the importance metric to account for task-specific loss functions. While proposed framework is architecture-agnostic, its hyperparameters such as the dynamic weight α and β is to use automated approaches like reinforcement learning, Bayesian optimization or even neural architecture search to discover optimal pruning policies tailored to specific network architectures (e.g., Transformers, MobileNets). Moreover, we schedule to evaluate under distributional shifts expending to benchmarks like CIFAR-10-C to make certain the pruned models retain performance on out-of-distribution or corrupted data, which is critical for real-world deployment.

6.2. Practical impact and cost concerns

The key cost impact is the drastic reduction in computational resources required for inference. A model with 51.8% fewer FLOPs (like our pruned ResNet-50) translates to comparative savings in computing costs or even permits deployment on inexpensive, lower-power edge devices. This can decrease the ongoing operational cost of serving AI models at scale. Although pruning itself has a computational cost, it is a one-time cost that settles over the model's entire duration. Additionally, by compressing large models rather than training small ones from scratch, proposed method can decrease the carbon footprint and financial cost connected with extensive training cycles. The substantial reduction in model size qualifies applications formerly considered impractical on resource-constrained devices, unfolding new opportunities and use cases.

7. Conclusion

In this work, we presented a framework that dynamically integrates L1-norm and Pearson correlation to evaluate both the importance and redundancy of convolutional filters. Unlike static multi-criteria methods, proposed method dynamically balances L1 magnitude and redundancy elimination throughout training, adapting to layer-wise specialization patterns observed during feature hierarchy formation. This dual-criterion approach enables selective compression of weak and redundant filters while maintaining task-relevant representations. We conducted extensive experiments on CIFAR-10 and ImageNet, demonstrating that proposed method consistently achieves strong trade-offs between accuracy, parameter reduction, and FLOPs, outperforming state-of-the-art pruning methods across VGG and ResNet architectures. We also incorporated an ablation study validating the critical role of dynamic weighting and fallback adaptation.

CRediT authorship contribution statement

Ali Muhammad Shaikh: Writing – original draft, Data curation, Conceptualization. **Yu Kang:** Supervision. **Aakash Kumar:** Writing – review & editing. **Yun-bo Zhao:** Supervision, Investigation, Formal analysis.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

The authors extend their profound thanks to Prof. Yu Kang for his invaluable insights, and his meticulous supervision throughout this research project. His guidance on the methodology was instrumental. The authors also gratefully acknowledge Prof. Yunbo Zhao for his continuous support, insightful feedback on the analysis of our research, and for providing access to key resources. This work would not have been possible without their collective constant support and expertise.

Data availability

Data will be made available on request.

References

[1] L. Zhang, Z. Sheng, Y. Li, Q. Sun, Y. Zhao, D. Feng, Image object detection and semantic segmentation based on convolutional neural network, Neural Comput. Appl. 32 (7) (2020) 1949–1958, https://doi.org/10.1007/s00521-019-04491-4.

[2] W. Ullah, T. Hussain, F.U.M. Ullah, M.Y. Lee, S.W. Baik, TransCNN: Hybrid CNN and transformer mechanism for surveillance anomaly detection, Eng. Appl. Artif. Intell. (2023), https://doi.org/10.1016/j.engappai.2023.106173.

[3] A. Kumar, S. Wang, A.M. Shaikh, H. Bilal, B. Lu, S. Song, Building on prior lightweight CNN model combined with LSTM-AM framework to guide fault detection in fixed-wing UAVs, Int. J. Mach. Learn. Cybern. 15 (9) (2024) 4175–4191, https://doi.org/10.1007/s13042-024-02141-3.

[4] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, Proc. IEEE Comput. Soc. Conf. Comput. vis. Pattern Recognit. vol. 2016–Decem (2016) 770–778, https://doi.org/10.1109/CVPR.2016.90.

[5] W. Fang, F. Zhang, V.S. Sheng, Y. Ding, A method for improving CNN-based image recognition using DCGAN, Comput. Mater. Contin. 57 (1) (2018) 167–178, https://doi.org/10.32604/cmc.2018.02356.

[6] A. Kumar, B. Yin, A.M. Shaikh, M. Ali, W. Wei, CorrNet: pearson correlation based pruning for efficient convolutional neural networks, Int. J. Mach. Learn. Cybern. 13 (12) (2022) 3773–3783, https://doi.org/10.1007/s13042-022-01624-5.

[7] J. Wang, T. Jiang, Z. Cui, Z. Cao, Filter pruning with a feature map entropy importance criterion for convolution neural networks compressing, Neurocomputing 461 (2021) 41–54, https://doi.org/10.1016/j.neucom.2021.07.034.

[8] Y. Liu, K. Fan, and W. Zhou, "Iterative filter pruning with combined feature maps and knowledge distillation," *Int. J. Mach. Learn. Cybern.*, no. 0123456789, 2024, doi: 10.1007/s13042-024-02371-5.

[9] X. Chen, J. Zhu, J. Jiang, C.Y. Tsui, Tight compression: Compressing CNN model tightly through unstructured pruning and simulated annealing based permutation, Proceedings - Design Automation Conference (2020), https://doi.org/10.1109/DAC18072.2020.9218701.

[10] S. Han, J. Pool, J. Tran, W.J. Dally, Learning both weights and connections for efficient neural networks. In *Advances in Neural Information Processing Systems*, 2015.

[11] H. Li, H. Samet, A. Kadav, I. Durdanovic, H.P. Graf, "Pruning filters for efficient convnets," *5th Int*, Conf. Learn. Represent. ICLR 2017 - Conf. Track Proc., (2016, 2017,) 1–13.

[12] H. Zhang, L. Liu, H. Zhou, L. Si, H. Sun, N. Zheng, FCHP: Exploring the Discriminative Feature and Feature Correlation of Feature Maps for Hierarchical DNN Pruning and Compression, IEEE Trans. Circuits Syst. Video Technol. (2022), https://doi.org/10.1109/TCSVT.2022.3170620.

[13] A. Kumar, A.M. Shaikh, Y. Li, H. Bilal, B. Yin, Pruning filters with L1-norm and capped L1-norm for CNN compression, Appl. Intell. (2021), https://doi.org/10.1007/s10489-020-01894-y.

[14] Y. Lian, P. Peng, K. Jiang, W. Xu, Cross-layer importance evaluation for neural network pruning, Neural Netw. 179 (2024) 106496, https://doi.org/10.1016/j.neunet.2024.106496.

[15] Z. Huang, N. Wang, Data-Driven Sparse Structure selection for Deep Neural Networks, Lecture Notes in Computer Science (including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) (2018), https://doi.org/10.1007/978-3-030-01270-0_19.

[16] B. Cui, Y. Li, Z. Zhang, Joint structured pruning and dense knowledge distillation for efficient transformer model compression, Neurocomputing (2021), https://doi.org/10.1016/j.neucom.2021.05.084.

[17] Y. He, L. Xiao, Structured Pruning for Deep Convolutional Neural Networks: a Survey, IEEE Trans. Pattern Anal. Mach. Intell. (2024), https://doi.org/10.1109/TPAMI.2023.3334614.

[18] S. Guo, B. Lai, S. Yang, J. Zhao, F. Shen, Sensitivity pruner: Filter-Level compression algorithm for deep neural networks, Pattern Recognit. (2023), https://doi.org/10.1016/j.patcog.2023.109508.

[19] G. Li, H. Shao, X. Deng, Y. Jiang, Adaptive convolutional network pruning through pixel-level cross-correlation and channel independence for enhanced model compression, Eng. Appl. Artif. Intell. 154 (2025) 110920, https://doi.org/10.1016/j.engappai.2025.110920.

[20] A. Krizhevsky, Learning Multiple Layers of Features from Tiny Images, Sci. Dep. Univ. Toronto, Tech. doi (2009) 10.1.1.222.9220.

[21] Y. He, P. Liu, Z. Wang, Z. Hu, Y. Yang, Filter pruning via geometric median for deep convolutional neural networks acceleration, Proc. IEEE Comput. Soc. Conf. Comput. vis. Pattern Recognit. vol. 2019–June (2019) 4335–4344, https://doi.org/10.1109/CVPR.2019.00447.

[22] P. Singh, V.K. Verma, P. Rai, V.P. Namboodiri, Leveraging filter correlations for deep model compression, in: In *Proceedings - 2020 IEEE Winter Conference on Applications of Computer Vision*, 2020, 2020., https://doi.org/10.1109/WACV45572.2020.9093331.

[23] T. Lin, S. U. Stich, L. Barba, D. Dmitriev, and M. Jaggi, "DYNAMIC MODEL PRUNING WITH FEEDBACK," in *8th International Conference on Learning Representations, ICLR 2020*, 2020.

[24] A. Jacot, F. Gabriel, Neural Tangent Kernel: Convergence and Generalization in Neural Networks Arthur, NeurIPS (2020).

[25] D. Bonet, A. Ortega, J. Ruiz-Hidalgo, and S. Shekkizhar, "CHANNEL REDUNDANCY AND OVERLAP IN CONVOLUTIONAL NEURAL NETWORKS WITH CHANNEL-WISE NNK GRAPHS," in *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, 2022. doi: 10.1109/ICASSP43922.2022.9746186.

[26] A. M. Saxe, J. L. McClelland, and S. Ganguli, "Exact solutions to the nonlinear dynamics of learning in deep linear neural networks," 2014. [Online]. Available: https://arxiv.org/abs/1312.6120.

[27] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, L. Fei-Fei, ImageNet: a large-scale hierarchical image database, in: In *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPR Workshops)*, 2009, pp. 248–255, https://doi.org/10.1109/CVPR.2009.5206848.

[28] K. He, X. Zhang, S. Ren, J. Sun, "Delving deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification," in, IEEE International Conference on Computer Vision (ICCV) 2015 (2015) 1026–1034, https://doi.org/10.1109/ICCV.2015.123.

[29] S. Lin, et al., Towards optimal structured CNN pruning via generative adversarial learning, in: In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2019, https://doi.org/10.1109/CVPR.2019.00290.

[30] M. Lin *et al.*, "HRank: Filter Pruning using High-Rank Feature Map", Accessed: Oct. 05, 2024. [Online]. Available: https://github.com/lmbxmu/HRank.

[31] H. Wei, Z. Wang, G. Hua, J. Sun, Y. Zhao, Automatic Group-based Structured Pruning for Deep Convolutional Networks, IEEE Access (2022), https://doi.org/10.1109/ACCESS.2022.3227619.

[32] C. Wang, R. Grosse, S. Fidler, and G. Zhang, "Eigendamage: Structured pruning in the Kronecker-factored eigenbasis," *36th Int. Conf. Mach. Learn. ICML 2019*, vol. 2019–June, pp. 11398–11407, 2019.

[33] Y. He, Y. Ding, P. Liu, L. Zhu, H. Zhang, Y. Yang, Learning Filter Pruning Criteria for Deep Convolutional Neural Networks Acceleration, Proc. IEEE Comput. Soc. Conf. Comput. vis. Pattern Recognit. 2 (2020) 2006–2015, https://doi.org/10.1109/CVPR42600.2020.00208.

[34] J. Chang, Y. Lu, P. Xue, Y. Xu, Z. Wei, Automatic channel pruning via clustering and swarm intelligence optimization for CNN, Appl. Intell. (2022), https://doi.org/10.1007/s10489-022-03508-1.

[35] X. Dong, Y. Yang, Network pruning via transformable architecture search. In *Advances in Neural Information Processing Systems*, 2019.

[36] W. Jiang, Y. Chen, S. Wen, L. Zheng, H. Jin, PDAS: improving network pruning based on Progressive Differentiable Architecture Search for DNNs, Futur. Gener. Comput. Syst. 146 (2023) 98–113, https://doi.org/10.1016/j.future.2023.04.011.

[37] J.-H. Luo, J. Wu, AutoPruner: an end-to-end trainable filter pruning method for efficient deep model inference, Pattern Recognit. 107 (2020) 107461, https://doi.org/10.1016/j.patcog.2020.107461.

[38] Z. Yan, P. Xing, Y. Wang, Y. Tian, Prune it Yourself: Automated Pruning by Multiple Level Sensitivity, 2020 IEEE Conf. Multimed. Inf. Process. Retr. (2020) 73–78.

[39] Y. He, G. Kang, X. Dong, Y. Fu, Y. Yang, "Soft filter pruning for accelerating deep convolutional neural networks," *IJCAI Int*, Jt. Conf. Artif. Intell. vol. 2018–July (2018) 2234–2240, https://doi.org/10.24963/ijcai.2018/309.

[40] T. Niu, Y. Teng, and P. Zou, "Asymptotic Soft Cluster Pruning for Deep Neural Networks," 2022. [Online]. Available: https://arxiv.org/abs/2206.08186.

[41] S.H.S. Basha, M. Farazuddin, V. Pulabaigari, S.R. Dubey, S. Mukherjee, Deep Model Compression based on the Training history, Neurocomputing (2024), https://doi.org/10.1016/j.neucom.2024.127257.
[42] C.H. Sarvani, M. Ghorai, S.H.S. Basha, PRF: deep neural network compression by systematic pruning of redundant filters, Neural Comput. Appl. 36 (33) (2024) 20607–20616, https://doi.org/10.1007/s00521-024-10256-5.
[43] M. Lin, R. Ji, Y. Zhang, B. Zhang, Y. Wu, Y. Tian, Channel pruning via automatic structure search, IJCAI International Joint Conference on Artificial Intelligence (2020), https://doi.org/10.24963/ijcai.2020/94.
[44] M. Lin, et al., Filter Sketch for Network Pruning, IEEE Trans. Neural Networks Learn. Syst. (2022), https://doi.org/10.1109/TNNLS.2021.3084206.
[45] Y. He, Y. Ding, P. Liu, L. Zhu, H. Zhang, Y. Yang, Learning Filter Pruning Criteria for Deep Convolutional Neural Networks Acceleration, in: In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2020, https://doi.org/10.1109/CVPR42600.2020.00208.
[46] G. Zhang, S. Xu, J. Li, A.J.X. Guo, Group-based network pruning via nonlinear relationship between convolution filters, Appl. Intell. (2022), https://doi.org/10.1007/s10489-021-02907-0.
[47] X. Lu, H. Huang, W. Dong, X. Li, G. Shi, Beyond network pruning: a joint search-and-training approach, IJCAI International Joint Conference on Artificial Intelligence (2020), https://doi.org/10.24963/ijcai.2020/358.