Indonesian Journal of Electrical Engineering and Computer Science
Vol. 41, No. 2, February 2026, pp. 753∼763
ISSN: 2502-4752, DOI: 10.11591/ijeecs.v41.i2.pp753-763

RAC: a reusable adaptive convolution for CNN layer

Nguyen Viet Hung[1], Phi Dinh Huynh[1], Pham Hong Thinh[2], Phuc Hau Nguyen[3], Trong-Minh Hoang[4]
[1]International Training and Cooperation Institute, East Asia University of Technology, Bacninh, Vietnam
[2]Quy Nhon University, Quynhon, Vietnam
[3]Electric Power University, Hanoi, Vietnam
[4]Posts and Telecommunications Institute of Technology, Hanoi, Vietnam

Article Info

Article history:

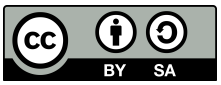
Received Dec 1, 2025
Revised Jan 2, 2026
Accepted Jan 11, 2026

Keywords:

Convolutional neural networks
Filter sharing
Lightweight deployment
Memory efficiency
Model compression
Reusable adaptive convolution

ABSTRACT

This paper proposes reusable adaptive convolution (RAC), an efficient alternative to standard 3×3 convolutions for convolutional neural networks (CNNs). The main advantage of RAC lies in its simplicity and parameter efficiency, achieved by sharing horizontal and vertical $1\times k/k\times1$ filter banks across blocks within a stage and recombining them through a lightweight 1×1 mixing layer. By operating at the operator design level, RAC avoids post-training compression steps and preserves the conventional Conv–BN–activation structure, enabling seamless integration into existing CNN backbones. To evaluate the effectiveness of the proposed method, extensive experiments are conducted on CIFAR-10 using several architectures, including ResNet-18/50/101, DenseNet, WideResNet, and EfficientNet. Experimental results demonstrate that RAC significantly reduces parameters and memory usage while maintaining competitive accuracy. These results indicate that RAC offers a reasonable balance between accuracy and compression, and is suitable for deploying CNN networks on resource-constrained platforms.



Corresponding Author:

Nguyen Viet Hung
International Training and Cooperation Institute, East Asia University of Technology
Bacninh, Vietnam
Email: hungnv@eaut.edu.vn

1. INTRODUCTION

Convolutional neural networks (CNNs) have driven major progress in vision, with strong results across image classification, detection, and tracking [1]–[6]. Recent backbones continue to scale depth and width to push accuracy, from ConvNeXt/ConvNeXt-V2 in the CNN family to hierarchical transformers like swin and swin-V2 [7]–[11]. However, the price of these gains is larger models and higher computational cost, which complicates training and deployment on resource-limited devices [12]–[14].

A large body of work reduces this cost along three main lines. Quantization lowers precision for weights/activations, often from FP32 to low-bit integers [15], [16]; removes weights or channels deemed redundant [17], [18]; and low-rank factorization decomposes convolutional kernels into products of smaller matrices/tensors [19], [20]. While effective, each line has trade-offs: quantization and pruning may require careful hyperparameter tuning or fine-tuning and can be sensitive to distribution shift [21]–[23]; pruning's theoretical sparsity does not always translate to proportionate wall-clock speedups [24], [25]; and low-rank methods depend strongly on rank choices and hardware locality for practical speed [26], [27]. In parallel, lightweight architectures (e.g., MobileNet, ShuffleNet, EfficientNet) redesign blocks to balance accuracy and efficiency [28]–[30].

In this paper we follow a complementary direction: instead of compressing a trained network, we reorganize the convolutional layer itself. Figure 1 contrasts two views. In Figure 1(a), each layer learns its own 3×3 kernels independently. In Figure 1(b), filters are assembled from shared components; layers no longer relearn the same structures from scratch but compose them. This motivates our method, which restructures the 3×3 operator into shared directional bases and a light mixing step, aiming to keep accuracy while reducing parameters and compute.

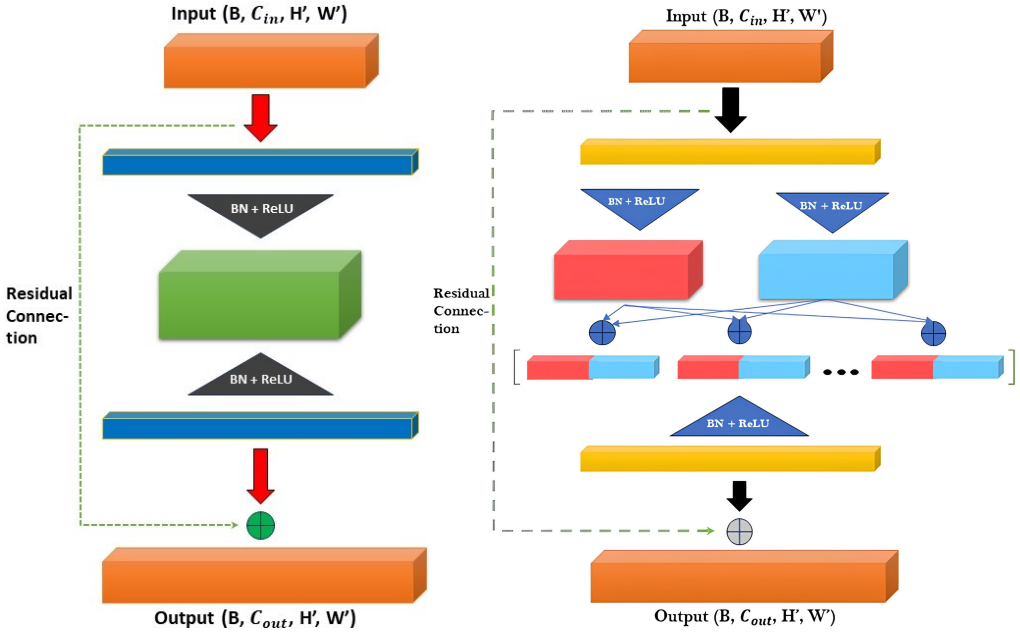


Figure 1. Comparison of convolutional organizations: (a) conventional block where each convolution learns independent filters and (b) an example of reorganized design using shared components and compositional filters. This line of work inspires methods that restructure convolution itself, beyond quantization, pruning, or factorization

We introduce reusable adaptive convolution (RAC), a drop-in replacement for 3×3 conv. RAC builds two shared banks of $1\times k$ and $k\times1$ filters (horizontal/vertical). Within a stage, blocks reuse these banks and form block-specific virtual filters by selecting and fusing bank responses; a 1×1 projection then mixes channels. This simple change keeps spatial resolution, promotes feature reuse across blocks, and reduces redundancy, while remaining compatible with standard layers (Conv/BN/ReLU) and typical toolchains. RAC is architecture-agnostic and can be plugged into common backbones such as ResNet, WideResNet, and DenseNet without altering their overall topology. To summarize the conceptual differences between RAC and commonly used convolutional decomposition strategies, we present Table 1, emphasizing that RAC operates at the operator reorganization level rather than factorization on a layer-by-layer basis.

Table 1. Comparison between RAC and related convolution designs

Aspect	Std. Conv	Depthwise+pointwise	Low-rank	RAC
Decomposition level	None	Per layer	Per layer	Stage-wise
Parameter sharing	No	No	No	Yes
Training paradigm	End-to-end	End-to-end	Often post-hoc	End-to-end
Structural reuse	None	Limited	Limited	Explicit
Design objective	Accuracy	Efficiency	Compression	Reusable operator

On CIFAR-10, RAC delivers accuracy close to the corresponding baselines while reducing memory footprint and training time. Beyond aggregate numbers, we also include diagnostics such as stage×block heatmaps to show where parameters concentrate and how RAC shifts load away from the heaviest regions.

We summarize our main contributions as follows:

- We introduce RAC, an operator-level alternative to standard 3×3 convolutions that reorganizes spatial filtering into stage-wise shared $1\times k/k\times1$ banks followed by a lightweight 1×1 mixing layer, enabling parameter reuse across blocks.
- We clarify the relationship between RAC and existing decomposition-based approaches, showing that RAC differs from depthwise separable and low-rank convolutions by operating as a reusable, end-to-end trainable operator rather than a per-layer or post-training factorization.
- We demonstrate the effectiveness of RAC by integrating it into multiple canonical CNN backbones and evaluating on CIFAR-10, where RAC achieves competitive accuracy with reduced memory consumption and favorable efficiency–performance trade-offs.

The remainder of this paper is organized as follows: section 2 presents the proposed RAC architecture, detailing the row-column bank design and virtual convolutional block (VCB) construction. After that section 3 provides experimental evaluations on CIFAR-10 with various "CNN backbones", comparing RAC with baseline models in terms of accuracy, storage size, and training time. Finally, section 4 concludes the paper and discusses potential future research directions.

2. METHOD

This section will concentrate on the design of RAC, its benefits and drawbacks, and the operation of the RAC.

2.1. On the reordering of CNN layers

To motivate RAC, we inspect the structure of widely used CNN backbones and observe that 3×3 convolutions are repeatedly applied with similar configurations across many blocks. For example, ResNet families rely on bottleneck blocks that recur multiple times within a stage [31], [32], while DenseNet employs 3×3 kernels throughout dense blocks [33], [34]. These repeated 3×3 layers contribute a large portion of the parameter and computation budget and may learn overlapping patterns, suggesting an opportunity to improve efficiency by enabling reuse rather than treating each layer as fully independent.

Based on this observation, we propose RAC. Instead of instantiating many separate 3×3 kernels, RAC learns two shared prototype banks that produce directional 1D filters, i.e., $1\times k$ (horizontal) and $k\times1$ (vertical), within each stage. Their responses are then combined through a VCB recomposition module and a lightweight 1×1 mixing layer to generate the final output.

Figure 2 illustrates the overall architecture. Compared to the conventional design in Figure 3 that stacks many *3×3* convolutions, RAC starts from the two shared banks to produces multiple intermediate responses, concatenates R fused components along the channel dimension, and finally applies the 1×1 mix layer for channel blending. The mechanism of shared-bank is shown in Algorithm 1. This two-part design consists of shared-bank creation (section 2.2) and mixer and virtual recomposition (section 2.3), which we detail next.

2.2. How shared-bank works

We construct the stage-wise prototype banks in Algorithm 1 by learning two shared operator sets: a horizontal bank of $1\times k$ filters and a vertical bank of $k\times1$ filters, instead of learning an independent 3×3 kernel for every block. Given an input feature map x, the banks produce two response stacks U and V (horizontal/vertical), each stacking m responses and preserving the spatial size of x. Because the same banks are reused by all blocks within a stage, their parameters receive gradients from multiple blocks, encouraging cross-block reuse and typically improving optimization stability while reducing parameters versus per-block 3×3 convolutions. The bank size m controls the expressiveness of the factor sets (more prototypes for U and V) at the cost of additional computation and parameters.

Using two 1D banks (row and column) provides a set of directional spatial primitives. Many local 2D patterns can be expressed by combining horizontal and vertical responses, while the subsequent 1×1 mixer learns how to blend multiple recombinations to match the target feature channels. Therefore, RAC does not claim an exact theoretical equivalence to a full 3×3 kernel, but offers a practical structured basis that works well in our empirical setting.

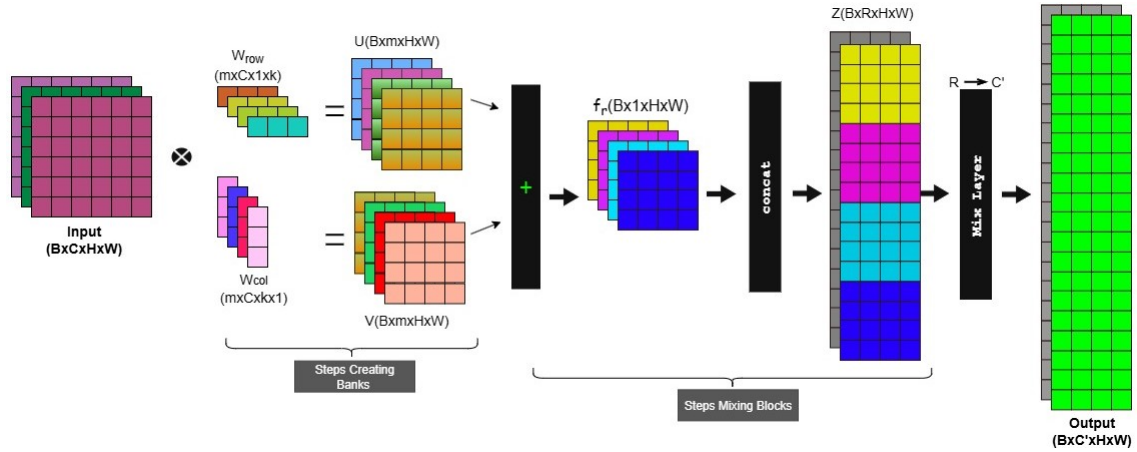


Figure 2. Our re-construction method

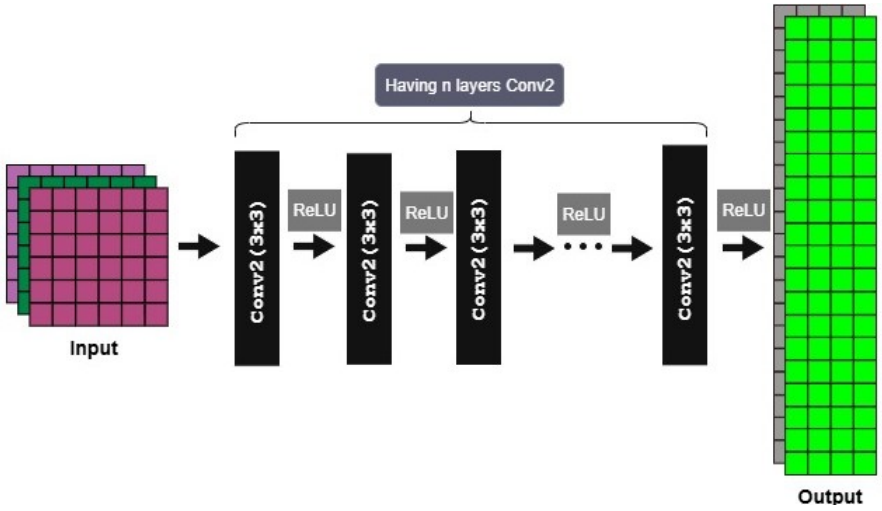


Figure 3. Old multi-conv3x3 structure. Each conv3x3 contains individual kernels, which cannot share information between layers, and each conv3x3 layer is also very parameter-heavy $((3 \times 3 \times C_{in}) + 1) \times C_{out}$

Algorithm 1: Shared-bank mechanism

function SHAREDBANK(x, m, k, W_{row}, W_{col})
  Input: $x \in \mathbb{R}^{B \times C \times H \times W}$; bank size m; kernel size k;
    $W_{\text{row}} \in \mathbb{R}^{m \times C \times 1 \times k}$, $W_{\text{col}} \in \mathbb{R}^{m \times C \times k \times 1}$.
  Output: $U, V \in \mathbb{R}^{B \times m \times H \times W}$.
  $p \leftarrow \lfloor k/2 \rfloor$
  $U \leftarrow \text{Conv2D}(x, W_{\text{row}};\ \text{stride} = 1,\ \text{padding} = (0, p))$
  $V \leftarrow \text{Conv2D}(x, W_{\text{col}};\ \text{stride} = 1,\ \text{padding} = (p, 0))$
  return (U, V)
end function

2.3. Mixer and virtual recomposition

In this part, we explain what the RAC block does after obtaining the two response stacks U and V as shown in Algorithm 2. Instead of learning a full 3×3 kernel in each block, RAC builds R virtual components by repeatedly picking one channel from U and one channel from V using the index pairs (α_r, β_r). For each pair, the two selected maps are fused (in our implementation, a simple element-wise sum) to form one component. These R components are then concatenated along the channel dimension to form an intermediate tensor with R channels, and a lightweight 1×1 mixing layer produces the final C' output channels. In short, RAC reuses

the same directional primitives across blocks and only learns a small mixer to combine them, which reduces redundant parameters while keeping the spatial output unchanged.

Algorithm 2: Mixing and virtual recomposition (RAC block)

function BLOCK($U, V, \boldsymbol{\alpha}, \boldsymbol{\beta}, W_{mix}$)**:**
  Input: $U, V \in \mathbb{R}^{B\times m\times H\times W}$; $R = |\boldsymbol{\alpha}| = |\boldsymbol{\beta}|$;
  $\boldsymbol{\alpha}, \boldsymbol{\beta} \in \{1, \ldots, m\}^R$;
  W_{mix} (1×1 mixer producing C' channels).
  Output: $y \in \mathbb{R}^{B\times C'\times H\times W}$.
  $R \leftarrow |\boldsymbol{\alpha}|$
  $\mathcal{Z} \leftarrow [\,]$
  for $r \leftarrow 1$ **to** R **do**
    $u \leftarrow U[:, \alpha_r : \alpha_r{+}1, :, :]$
    $v \leftarrow V[:, \beta_r : \beta_r{+}1, :, :]$
    $z_r \leftarrow \texttt{Fuse}(u, v)$
    append z_r to $\mathcal{Z}$
  $Y \leftarrow \text{Concat}(\mathcal{Z})$ // $Y \in \mathbb{R}^{B\times R\times H\times W}$
  return $\texttt{Conv}_{1\times1}(Y;\, W_{\text{mix}})$

3. RESULTS AND DISCUSSION

This part of the paper will present our experimental setup including: device configuration, dataset used and how we build the models (ResNet18/ResNet50/ResNet101/WideResNet/DenseNet/EfficientNet) and plug RAC into them. Finally, we will present the comparison results on the accuracy between RAC and non-RAC as well as the memory consumption and training time.

3.1. Experimental setup

The experiments were carried out on a 64-bit Windows 11 Pro and an NVIDIA GeForce RTX 3060 GPU. The implementation was developed in Python 3.10, utilizing essential libraries (e.g., PyTorch). Table 2 describes where we plug *RAC* into the baseline backbones and channel output.

Table 2. Location of RAC usage

Backbone	Location	C' (channel output)
ResNet–50	C4 (`layer3`).conv2 (6×)	256
ResNet–101	C4 (`layer3`).conv2 (23×)	256
ResNet–18	C4 (`layer3`).conv2 (2×)	256
WideResNet	group3.conv2 (each block)	group3 width
DenseNet	All `DenseLayer` conv2 (3×3)	growth rate (32)
EfficientNet	MBConv blocks in stage 4 (depthwise 3×3)	stage width

We conduct all experiments on the CIFAR-10 dataset [35], [36], a benchmark consisting of 60,000 color images at resolution 32×32 spanning 10 classes (50 k train, 10 k test). For our procedures, images are resized to 128×128 and trained with standard enhancements (random cropping/flipping) and normalization; evaluation uses the formal test split without label noise or additional data. All models (baseline and RAC variants) are trained and reported on the same preprocessing and training schedule (epoch = 200, batch_size = 256, lr = 0.1, seed = 42) for fair comparisons.

3.2. Performance evaluation

This section will present the results we obtained after the experiment but before that we will talk about the metrics used as evaluation measures. To evaluate the stability and accuracy of the models, we use two main formulas, Top-1 and Top-5 accuracy, which are widely used formulas when evaluating on the CIFAR-10 set [37]. They are stated very clearly both in terms of formula and efficiency in [38], [39].

Next, we perform the experiment and get the results as shown in Figure 4. On CIFAR-10, RAC-ResNet50 achieves 92.82%, while the baseline ResNet50 achieves 94.68%, the accuracy of RAC is only 2% lower than the baseline, but the benefits are less memory and training time ($\approx$ 77 MB vs. 90 MB and 300 seconds less). On other backbones (ResNet18/101, WideResNet, DenseNet, and EfficientNet), the instances show

the same results: slightly lower accuracy (≈1-2%) than the baseline but with parameter/storage savings (in the WideResNet case, the computational parameters are almost half lower than the baseline). Overall, Figure 4 shows the desired trade-off: replace 3x3 immediately while maintaining the optimization behavior, reduce model size, and still remain competitive in accuracy.

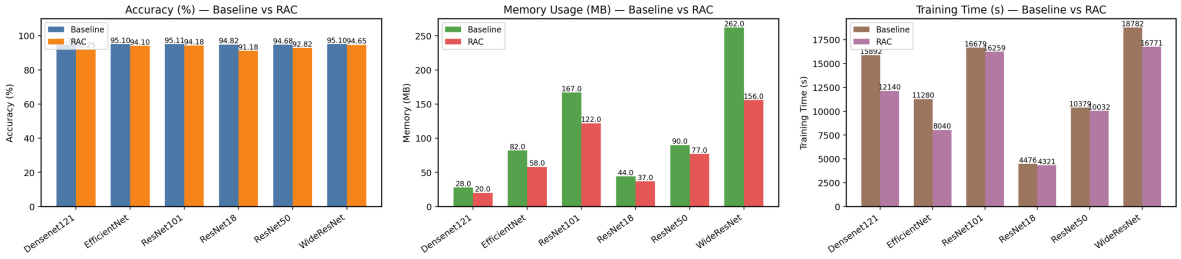


Figure 4. The results after the changes are illustrated as follows: the left panel shows the accuracy comparison, the middle shows memory consumption, and the right displays the training time of the models

Figure 5 compares the inference latency of the base models and their corresponding RAC-based models under the same experimental conditions. Across all evaluated architectures, RAC consistently achieved lower inference latency than their corresponding base models. The latency reduction was more pronounced for deeper and heavier networks such as ResNet-50, ResNet-101, DenseNet, and WideResNet, where standard convolutions contributed a significant portion to the computational cost. For lighter architectures like EfficientNet, the latency difference between the base variants and RAC was smaller but still consistently skewed toward RAC. These results suggest that reorganizing standard convolutions into reusable phase-level operators can reduce inference time costs without creating additional computational bottlenecks. It is important to note that the reported latency values are measured at the frame level under controlled conditions and are intended to reflect relative performance trends rather than fully optimized deployment latency on specific hardware platforms.

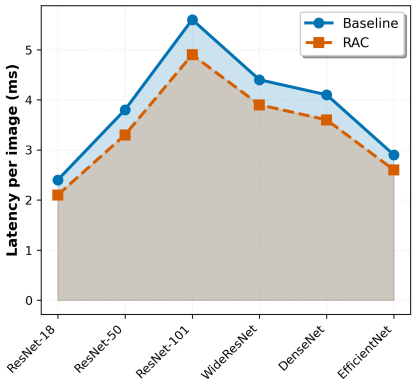


Figure 5. Inference latency per image (ms) of baseline vs RAC (batch=1) on RTX 3060; lower is better

Figure 6 presents the training dynamics of the baseline and RAC-based models over 200 epochs, including accuracy and loss curves for different architectures. Figures 6(a) to 6(c) show the results for ResNet-18, ResNet-50, and ResNet-101, respectively, where RAC exhibits convergence behaviors comparable to the baselines while generally displaying reduced fluctuations in the loss curves. For deeper models, such as ResNet-101 in Figure 6(c) and DenseNet in Figure 6(e), the RAC variants demonstrate noticeably smoother convergence trajectories, particularly during the early and middle training stages. Similar trends can be observed for WideResNet and EfficientNet in Figures 6(d) and 6(f), where RAC maintains stable training without degrading the final accuracy. Overall, these results indicate that introducing RAC does not adversely affect convergence and may lead to more stable optimization behavior, especially in deeper architectures.

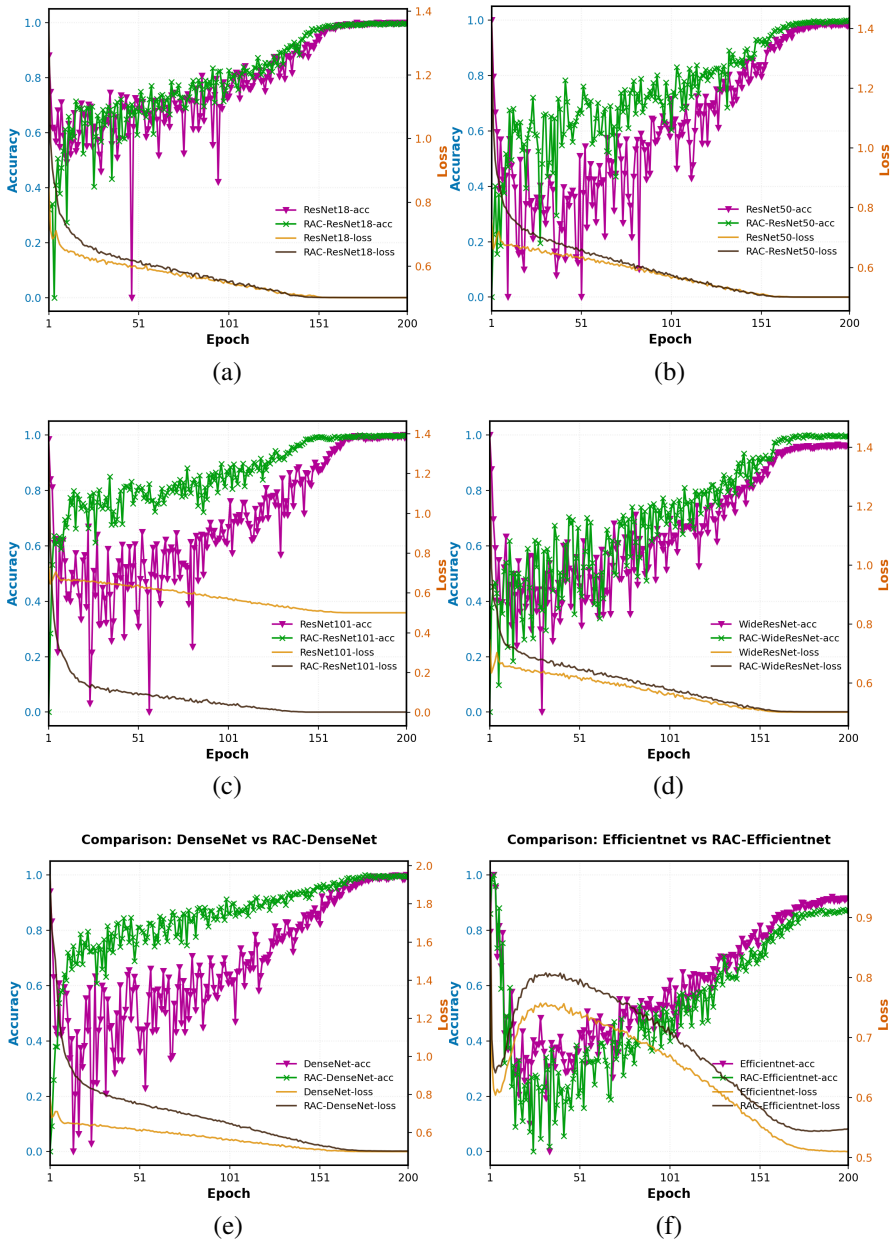


Figure 6. The charts show the improvement trend of the baseline models and RACs over 200 epochs: (a) ResNet18 vs. RAC_ResNet18, (b) ResNet50 vs. RAC_ResNet50, (c) ResNet101 vs. RAC_ResNet101, (d) WideResNet vs. RAC_WideResNet, (e) DenseNet vs. RAC_DenseNet, and (f) EfficientNet vs. RAC_EfficientNet

Figure 7 provides a visualization of the parameter distribution across different network stages, allowing a direct comparison between the baseline WideResNet and its RAC-enhanced counterpart. The heatmaps illustrate how parameters are allocated among layers after training, with color intensity indicating relative parameter density. As shown in Figure 7(a), the baseline WideResNet exhibits a highly unbalanced distribution, where the majority of parameters are concentrated in the deeper layers, particularly layer 4, followed by layer 3. In contrast, Figure 7(b) shows that the RAC-WideResNet significantly reduces the parameter density in layer 4. The noticeably lower color intensity in this stage indicates that parameter sharing and recombination effectively alleviate the computational burden of the deepest layers.

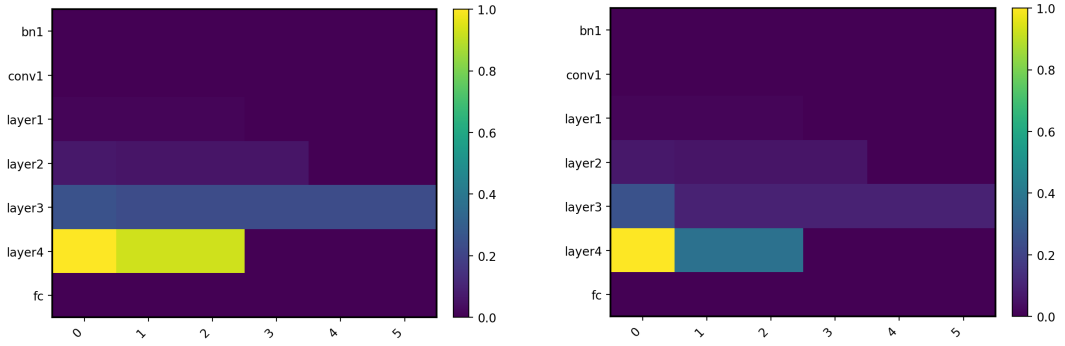


Figure 7. An example comparing the parameter load in each layer before and after RAC plugging: (a) parameter distribution chart of the WideResNet baseline model and (b) parameter distribution plot of WideResNet model after plugging in RAC

In addition, we examine the sensitivity of RAC to its two main hyperparameters, the bank size m and the number of virtual combinations R, and report the results in Table 3. In the upper part of the table, we vary $m \in \{4, 8, 16, 32\}$ while fixing R=2; accuracy typically improves when moving from small banks to moderate ones, then changes only marginally at larger m, suggesting that the shared banks become sufficiently expressive beyond a certain size. In the lower part, we vary $R \in \{1, 2, 3, 4\}$ with m=8 fixed; increasing R brings a small accuracy gain, but the benefit quickly saturates, indicating that only a few recombinations are needed in practice. Across different backbones, these trends are consistent and the variations are modest, so we adopt moderate settings (e.g., m=8–16 and R=2–3) as the default configuration in the main experiments unless stated otherwise.

Table 3. Ablation study of RAC hyperparameters on all backbones. Top-1 accuracy (%) on CIFAR-10

Effect of bank size m (fixed R=2)					
Model	Baseline	m=4	m=8	m=16	m=32
ResNet-18	94.82	92.80	93.20	93.40	93.35
ResNet-50	94.68	92.42	92.82	93.02	93.00
ResNet-101	95.11	93.20	93.60	93.80	93.72
WideResNet	95.10	93.90	94.30	94.50	94.49
DenseNet	94.78	93.10	93.50	93.70	93.71
EfficientNet	95.10	93.60	94.00	94.20	94.20
Effect of virtual combinations R (fixed m=8)					
Model	Baseline	R=1	R=2	R=3	R=4
ResNet-18	94.82	92.85	93.20	93.31	93.38
ResNet-50	94.68	92.47	92.82	92.94	93.00
ResNet-101	95.11	93.25	93.60	93.72	93.70
WideResNet	95.10	93.95	94.30	94.41	94.48
DenseNet	94.78	93.15	93.50	93.62	93.68
EfficientNet	95.10	93.65	94.00	94.12	94.20

3.3. Discussion

Experimental results show that RAC provides a practical balance between accuracy and efficiency by reorganizing standard 3×3 convolution operations into reusable stage-level operators. Across various CNN architectures, RAC consistently reduces the number of parameters and memory usage at deeper stages while maintaining accuracy within a narrow range compared to their corresponding underlying methods. This suggests that sharing spatial filter banks between blocks can effectively minimize redundant learning without significantly reducing performance. It is important to note that the performance improvements achieved by RAC should be interpreted within the scope of the experiments performed. All evaluations were performed on CIFAR-10, a small-scale and low-resolution dataset, and the generalizability of RAC to larger benchmarks such as ImageNet has yet to be established.

Furthermore, the experiments were limited to image classification tasks, and the behavior of RAC in more complex settings such as object detection or semantic segmentation remains an open question. From an efficiency perspective, while RAC reduces parameter storage and shows a favorable latency trend, the current implementation does not explicitly optimize kernel combination or memory access patterns across specific hardware platforms. Therefore, the reported runtime benefits reflect measurements at the frame level rather than fully optimized implementation scenarios. Additionally, the hyperparameters controlling RAC, specifically the bank size m and the number of virtual combinations R, are manually selected and fixed across stages, which may not be optimal for all architectures.

Overall, these observations underscore that RAC should be viewed as an operator-level structural design, supplementing rather than replacing existing compression and optimization techniques. Further future research is needed to investigate its scalability, task generality, and hardware-based optimization capabilities.

4. CONCLUSION

In this paper, we introduced RAC block as an alternative to standard 3×3 convolutions. Instead of letting each block in a stage learn independent full-rank 3×3 kernels, RAC builds stage-level shared $1\times k/k\times1$ banks and reconstructs virtual filters via a lightweight 1×1 mixing layer. This design preserves the conventional Conv–BN–Act interface while encouraging parameter sharing across blocks. We instantiated RAC in several backbones, including ResNet-18/50/101, WideResNet, DenseNet-121, and EfficientNet-B0, and evaluated them on CIFAR-10. Across these models, RAC reduces parameters and memory footprint (especially in deeper stages) with a modest accuracy trade-off, while the convergence curves, parameter heatmaps, and latency measurements provide an interpretable view of its training and efficiency behavior. Our current evaluation is limited to CIFAR-10 and framework-level runtime measurements; broader validation on larger datasets and real-device deployment remains future work.

Future directions include hardware-friendly fusion for the $1\times k/k\times1$ banks, automated tuning of (m, R) and stage-wise selection policies, and combining RAC with quantization, pruning, or distillation. We also plan to scale to ImageNet-1k and assess RAC on downstream detection and segmentation tasks.

FUNDING INFORMATION

The authors state no funding is involved.

CONFLICT OF INTEREST STATEMENT

The authors state no conflict of interest.

DATA AVAILABILITY

We will provide the data if anyone needs it.

REFERENCES

[1] L. Chen, S. Li, Q. Bai, J. Yang, S. Jiang, and Y. Miao, "Review of image classification algorithms based on convolutional neural networks," *Remote Sensing*, vol. 13, no. 22, p. 4712, Nov. 2021, doi: 10.3390/rs13224712.

[2] X. Zhao, L. Wang, Y. Zhang, X. Han, M. Deveci, and M. Parmar, "A review of convolutional neural networks in computer vision," *Artificial Intelligence Review*, vol. 57, no. 4, p. 99, 2024, doi: 10.1007/s10462-024-10721-6.

[3] N. Hung, T. Loi, N. Huong, T. T. Hang, and T. Huong, "AAFNDL - an accurate fake information recognition model using deep learning for the Vietnamese language," *Informatics and Automation*, vol. 22, no. 4, pp. 795–825, Jul. 2023, doi: 10.15622/ia.22.4.4.

[4] T. Turay and T. Vladimirova, "Toward performing image classification and object detection with convolutional neural networks in autonomous driving systems: a survey," *IEEE Access*, vol. 10, pp. 14076–14119, 2022, doi: 10.1109/ACCESS.2022.3147495.

[5] H. Nguyen Viet and P. D. Phong, "Building a new crowd-counting architecture," *Journal of Computer Applications in Technology*, Jul. 2023, doi: 10.36227/techrxiv.23691351.v1.

[6] Z. Li, F. Liu, W. Yang, S. Peng, and J. Zhou, "A survey of convolutional neural networks: analysis, applications, and prospects," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 12, pp. 6999–7019, Dec. 2022, doi: 10.1109/TNNLS.2021.3084827.

[7] Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie, "A ConvNet for the 2020s," *arXiv preprint arXiv:2201.03545*, 2022.

[8] S. Woo *et al.*, "ConvNeXt V2: co-designing and scaling ConvNets with masked autoencoders," in *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2023, vol. 2023-June, pp. 16133–16142, doi: 10.1109/CVPR52729.2023.01548.

[9] Z. Liu *et al.*, "Swin transformer: hierarchical vision transformer using shifted windows," in *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, Oct. 2021, pp. 9992–10002, doi: 10.1109/ICCV48922.2021.00986.

[10] Z. Liu *et al.*, "Swin transformer V2: scaling up capacity and resolution," in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2022, vol. 2022-June, pp. 11999–12009, doi: 10.1109/CVPR52688.2022.01170.

[11] N. Hung, T. Loi, N. Binh, N. Nga, T. Huong, and D. Luu, "Building an online learning model through a dance recognition video based on deep learning," *Informatics and Automation*, vol. 23, no. 1, pp. 101–128, Jan. 2024, doi: 10.15622/ia.23.1.4.

[12] D. Ngo, H. C. Park, and B. Kang, "Edge intelligence: a review of deep neural network inference in resource-limited environments," *Electronics (Switzerland)*, vol. 14, no. 12, p. 2495, 2025, doi: 10.3390/electronics14122495.

[13] C. Chen *et al.*, "Deep learning on computational-resource-limited platforms: a survey," *Mobile Information Systems*, vol. 2020, no. 1, p. 8454327, 2020, doi: 10.1155/2020/8454327.

[14] G. Bai *et al.*, "Beyond efficiency: a systematic survey of resource-efficient large language models," *arXiv preprint arXiv:2401.00625*, 2024.

[15] M. Chen *et al.*, "INT v.s. FP: a comprehensive study of fine-grained low-bit quantization formats," *arXiv preprint arXiv:2510.25602*, 2025.

[16] R. Gong *et al.*, "A survey of low-bit large language models: basics, systems, and algorithms," *Neural Networks*, vol. 192, Nov. 2025, doi: 10.1016/j.neunet.2025.107856.

[17] Z. Wang, C. Li, and X. Wang, "Convolutional neural network pruning with structural redundancy reduction," in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2021, pp. 14908–14917, doi: 10.1109/CVPR46437.2021.01467.

[18] Y. He and L. Xiao, "Structured pruning for deep convolutional neural networks: a survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 46, no. 5, pp. 2900–2919, May 2024, doi: 10.1109/TPAMI.2023.3334614.

[19] Y. Panagakis *et al.*, "Tensor methods in computer vision and deep learning," *Proceedings of the IEEE*, vol. 109, no. 5, pp. 863–890, 2021, doi: 10.1109/JPROC.2021.3074329.

[20] G. Wang, B. Tao, X. Kong, and Z. Peng, "Infrared small target detection using nonoverlapping patch spatial-temporal tensor factorization with capped nuclear norm regularization," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 60, pp. 1–17, 2022, doi: 10.1109/TGRS.2021.3126608.

[21] T. Dettmers, M. Lewis, Y. Belkada, and L. Zettlemoyer, "LLM.int8(): 8-bit matrix multiplication for transformers at scale," *Advances in Neural Information Processing Systems*, vol. 35, Nov. 2022.

[22] G. Xiao, J. Lin, M. Seznec, H. Wu, J. Demouth, and S. Han, "SmoothQuant: accurate and efficient post-training quantization for large language models," *Proceedings of Machine Learning Research*, vol. 202, pp. 38087–38099, 2023.

[23] E. Frantar, S. Ashkboos, T. Hoefler, and D. Alistarh, "GPTQ: accurate post-training quantization for generative pre-trained transformers," *11th International Conference on Learning Representations, ICLR 2023*, 2023.

[24] A. Tyagi, A. Iyer, W. H. Renninger, C. Kanan, and Y. Zhu, "Dynamic sparse training of diagonally sparse networks," *arXiv preprint arXiv:2506.11449*, 2025.

[25] T. Dao, D. Y. Fu, S. Ermon, A. Rudra, and C. Ré, "FLASHATTENTION: fast and memory-efficient exact attention with IO-awareness," *Advances in Neural Information Processing Systems*, vol. 35, pp. 16344–16359, 2022.

[26] A. Gural, P. Nadeau, M. Tikekar, and B. Murmann, "Low-rank training of deep neural networks for emerging memory technology," *arXiv preprint arXiv:2009.03887*, 2020.

[27] J. Xiao *et al.*, "HALOC: hardware-aware automatic low-rank compression for compact neural networks," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 9, pp. 10464–10472, Jun. 2023, doi: 10.1609/aaai.v37i9.26244.

[28] S. R. Aleti and K. Kurakula, "Evaluation of lightweight CNN architectures for multi-species animal image classification." 2024.

[29] M. Li, "Application of lightweight convolutional neural networks in image classification," LUT University, 2025.

[30] H. Briouya, A. Briouya, and A. Choukri, "Surveying lightweight neural network architectures for enhanced mobile performance," in *Communications in Computer and Information Science*, vol. 2168 CCIS, 2024, pp. 187–199.

[31] X. Jin, "Analysis of residual block in the ResNet for image classification," in *Proceedings of the 1st International Conference on Data Analysis and Machine Learning, Changsha, China*, 2024, pp. 253–257, doi: 10.5220/0012800400003885.

[32] F. Li, T. Sun, P. Dong, Q. Wang, Y. Li, and C. Sun, "MSF-CSPNet: a specially designed backbone network for faster R-CNN," *IEEE Access*, vol. 12, pp. 52390–52399, 2024, doi: 10.1109/ACCESS.2024.3386788.

[33] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul. 2017, pp. 4700–4708, doi: 10.1109/CVPR.2017.243.

[34] Y. Hou, Z. Wu, X. Cai, and T. Zhu, "The application of improved densenet algorithm in accurate image recognition," *Scientific Reports*, vol. 14, no. 1, p. 8645, 2024, doi: 10.1038/s41598-024-58421-z.

[35] A. Krizhevsky, "Learning multiple layers of features from tiny images," 2009. [Online]. Available: https://www.cs.toronto.edu/ kriz/learning-features-2009-TR.pdf.

[36] "CIFAR-10 and CIFAR-100 datasets." https://www.cs.toronto.edu/ kriz/cifar.html (accessed Oct. 20, 2025).

[37] A. V. Chuiko, V. V. Arlazarov, and S. A. Usilin, "The impact of dataset size on the reliability of model testing and ranking," *Bulletin of the South Ural State University. Series "Mathematical Modelling, Programming and Computer Software,"* vol. 18, no. 2, pp. 102–111, 2025, doi: 10.14529/mmp250209.

[38] S. B. Vinay and S. Balasubramanian, "A comparative study of convolutional neural networks and cybernetic approaches on CIFAR-10 dataset," *International Journal of Machine Learning and Cybernetics (IJMLC)*, vol. 1, no. 1, pp. 1–13, 2023.

[39] J. Chen, Z. Wu, Z. Wang, H. You, L. Zhang, and M. Yan, "Practical accuracy estimation for efficient deep neural network testing," *ACM Transactions on Software Engineering and Methodology (TOSEM*), vol. 29, no. 4, pp. 1–35, Oct. 2020, doi: 10.1145/3394112.

BIOGRAPHIES OF AUTHORS

Nguyen Viet Hung received the B.Sc. degree in informatics pedagogy from the Faculty of Engineering Technology, Ha Tinh University, Vietnam, in 2012, and the M.Sc. degree in information technology from the Faculty of Information Technology, Ho Chi Minh City University of Technology, Vietnam, in 2016. He is currently pursuing the Ph.D. degree in telecommunications engineering at Hanoi University of Science and Technology, Vietnam, in 2025. He is the Director of Information Technology Programs at the Training International and Cooperation Institute (ITCI), East Asia University of Technology. His research interests include multimedia communications, network security, artificial intelligence, traffic engineering in next-generation networks, machine learning, deep learning, computer vision, data and web mining, autonomous vehicles, quality of experience (QoE)/quality of service (QoS) assurance for network services, green networking, and related applications. He can be contacted at email: hungnv@eaut.edu.vn.

Phi Dinh Huynh is an undergraduate student at the Faculty of Information Technology, East Asia University of Technology, and a research assistant at the Mobilab Laboratory since 2023. His research focuses on computer networking, computer vision, data mining, and intelligent systems. He is particularly interested in integrating autonomous vehicles with AI-driven technologies to enhance smart mobility and intelligent communication systems. With strong enthusiasm for research, she continues to explore the synergy between machine learning, computer vision, and networked systems. He can be contacted at email: 20222072@eaut.edu.vn.

Pham Hong Thinh received the B.E. and M.E. degrees in electronics and telecommunications from the Hanoi University of Technology, Vietnam, in 2002 and 2006, respectively. He obtained his Ph.D. degree in telecommunications engineering from the Hanoi University of Science and Technology, Vietnam, in 2021. Since 2002, he has been working at Quy Nhon University, Vietnam. His research interests include QoE optimization, software-defined networking (SDN), AI, machine learning, virtual reality (VR), and content adaptation over HTTP. He can be contacted at email: phamhongthinh@qnu.edu.vn.

Phuc Hau Nguyen received his Bachelor's degree in information technology from Hai Phong Private University (2005), his Master's degree in computer networks and data communication from the Posts and Telecommunications Institute of Technology (2011), and his Ph.D. in informatics and computational engineering from M. I. Platov Southern Polytechnic University, Russian Federation (2021). His research interests include multimedia communications, information and network security, AI and machine learning, expert systems, software engineering, computer networks, digital society, sustainable IT, traffic engineering, and QoE/QoS in next-generation networks. He can be contacted at email: phuchauptit@gmail.com.

Trong-Minh Hoang (Senior Member, IEEE) received the bachelor's degree in physic engineering (1994) and electronic and telecommunication engineering (1999) from Hanoi University of Science and Technology, the master's degree in electronic and telecommunication engineering (2003), and the Ph.D.'s degree in telecommunication engineering (2014) from Posts and Telecommunications Institute of Technology. He is currently a lecturer in the Telecommunication Faculty of Posts and Telecommunications Institute of Technology. He is the head of the telecommunication network department. He has been working on the issues surrounding the performance of wireless networks. His research interests include routing, security, and network performance in mobile edge computing, flying ad hoc networks, wireless mobile networks, and beyond 5G. He applies mathematical analysis to model and analyze the behavior of complex systems and uses discrete event simulation tools to provide comprehensive solutions. He is a member of the IEEE ComSoc and IEEE Circuits and Systems Society. He can be contacted at email: hungnv@eaut.edu.vn.