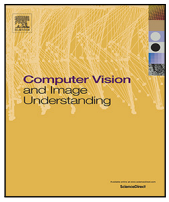
Computer Vision and Image Understanding 249 (2024) 104157

Contents lists available at ScienceDirect

Computer Vision and Image Understanding

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



# A lightweight convolutional neural network-based feature extractor for visible images

Xujie He, Jing Jin *, Yu Jiang, Dandan Li

*School of Astronautics, Harbin Institute of Technology, Harbin 150001, China*

ARTICLE INFO

Communicated by Maria Vakalalopoulou

*Keywords:*
Feature extraction network
Computer vision
Lightweight
Feature extraction
Memory efficiency

ABSTRACT

Feature extraction networks (FENs), as the first stage in many computer vision tasks, play critical roles. Previous studies regarding FENs employed deeper and wider networks to attain higher accuracy, but their approaches were memory-inefficient and computationally intensive. Here, we present an accurate and lightweight feature extractor (RoShuNet) for visible images based on ShuffleNetV2. The provided improvements are threefold. To make ShuffleNetV2 compact without degrading its feature extraction ability, we propose an aggregated dual group convolutional module; to better aid the channel interflow process, we propose a $\gamma$-weighted shuffling module; to further reduce the complexity and size of the model, we introduce slimming strategies. Classification experiments demonstrate the state-of-the-art (SOTA) performance of RoShuNet, which yields an increase in accuracy and reduces the complexity and size of the model compared to those of ShuffleNetV2. Generalization experiments verify that the proposed method is also applicable to feature extraction tasks in semantic segmentation and multiple-object tracking scenarios, achieving comparable accuracy to that of other approaches with more memory and greater computational efficiency. Our method provides a novel perspective for designing lightweight models.

## 1. Introduction

Digital multimedia industries have been thriving in the big data era, producing a massive amount of visible images that are expanding into all walks of life while facilitating the vigorous development of image processing and computer vision-related technologies. Feature extraction networks (FENs), which extract discriminative features from input images (usually as the first step), play exceptionally important roles in many computer vision tasks, such as visual image classification (Huo et al., 2023; Ke et al., 2022), visual image segmentation (Chen et al., 2017; Qu et al., 2022), object detection (Bonnaerens et al., 2022; Ren et al., 2017) and tracking (Sun et al., 2021; Tang et al., 2022). In some applications, feature extraction modules can impact or even determine the final task-oriented results. Therefore, investigating a feature extraction module with high accuracy at the front end is highly important. Convolutional neural network (CNN)-based feature extraction methods are object-agnostic and possess low color sensitivity and orientation isotropy, predominating other handcrafted features, such as Markov-based features (Han et al., 2016), Fourier-based features (Hjouji, 2022), and gray level co-occurrence matrix-based features (Wang and Sun, 2022). We therefore elaborate on the CNNs that have been proposed to extract features. Generally, CNNs can be categorized into 2 groups based on their parameter density and computational efficiency levels.

• **Deep and dense CNNs**. This group of CNNs achieves enhanced feature extraction capabilities by following deeper or wider structural paradigms. For example, Krizhevsky et al. (2012) combined a rectified linear unit (ReLU) activation layer, a dropout operation, and a series of data augmentation methods to mitigate overfitting and saturation problems. Based on the work of Krizhevsky et al. (2012), Zeiler and Fergus (2014) revealed the internal workings of CNNs for the first time. An efficient network named GoogLeNet and its successors were proposed by Szegedy et al. (2016) and Szegedy et al. (2017), which allowed for deeper and wider structures by simplifying the structures from fully connected to sparsely connected without increasing the incurred computational costs. In the work of Simonyan and Zisserman (2015), the relationships between various CNN depths and performances were first explored, and 9 types of structures named VGG11∼VGG19 with different numbers of layers were successfully built. To address the degradation problem resulting from a continually increasing network depth, 5 additional types of landmark networks named ResNet (He et al., 2016a) and its variants (He et al., 2016b) were proposed. To fully utilize the preceding network features, 4 additional types of networks named DenseNets were proposed. The development of highly accurate networks is critical for practical industrial applications; however, the

* Correspondence to: Intelligent Control Research Group, School of Astronautics, Harbin Institute of Technology, Room 502, Building 1, No. 92, Xidazhi Avenue, Nangang District, Harbin, Heilongjiang Province, China.
*E-mail addresses:* hexujie@stu.hit.edu.cn (X. He), jinjinghit@hit.edu.cn (J. Jin), hitjiangyu@hit.edu.cn (Y. Jiang), beauty@hit.edu.cn (D. Li).

https://doi.org/10.1016/j.cviu.2024.104157
Received 23 October 2022; Received in revised form 31 May 2024; Accepted 4 September 2024
Available online 12 September 2024

above deep and dense CNNs are usually not hardware-aware, and they demand high resource overhead levels. Therefore, it is more feasible to design lightweight and highly accurate CNN structures, which will likely become a trend in the coming years.

• **Lightweight CNNs**. Lightweight CNNs are usually designed for mobile devices because of their high computational efficiency levels. To date, several lightweight and highly accurate CNNs have been proposed in the literature. For example, Xception (Chollet, 2017) introduced the use of depthwise separable convolution (DSC) to replace traditional convolution and achieved competitive accuracy while also reducing the incurred computational costs. MobileNet (Howard et al., 2017) achieved higher accuracy than the approach of Szegedy et al. (2016), but the number of parameters was only 20% of that used by their method. Given the simple structure of MobileNet and the effectiveness of the residual blocks in the technique of He et al. (2016a), an improved MobileNetV2 (Sandler et al., 2018) consequently emerged, which overcame the gradient vanishing problem through inverted residuals and retained the diversity of features with linear bottlenecks. MobileNetV3 (Howard et al., 2019) used the AutoML technique to investigate a fast yet accurate network architecture. The 3×3 kernel of the DSC was separated into 1×3 and 3×1 kernels to further reduce the computational overhead in the work of Freeman et al. (2018). Furthermore, Face++ released ShuffleNet (Zhang et al., 2018) at the same time, which mainly improved and compressed the ResNet-like structure with a group convolution module and a channel shuffling operation and eventually reduced the model size and attained higher performance than that of previous methods. By comparing ShuffleNet and MobileNetV2, Ma et al. (2018) identified four networking criteria and released ShuffleNetV2. This architecture was more advantageous than the previous lightweight CNNs in terms of its accuracy and processing speed under the same computational burden. In the research of Han et al. (2020), a series of linear transformations were applied to produce ghost feature maps at low costs while achieving comparable state-of-the-art (SOTA) performance. Given that the method of Han et al. (2020) did not show much of an advantage on a GPU platform, an improved version, G-GhostNet (Han et al., 2022), was proposed; this method deploys inexpensive cross-layer operations to reduce the number of required computations while mitigating the transformation of memory data to make the best use of the GPU platform and accelerate the inference process.

In this article, we propose an accurate and lightweight feature extractor with a CNN, which enables further model size compression without impairing the resulting accuracy. Our method starts with the basic ShuffleNetV2 but differs in three ways. To compress ShuffleNetV2 without impairing its feature extraction ability, we propose an aggregated dual group convolutional (A-DGC) module. The A-DGC module has two variants: a small version, i.e., A-DGC(s), which focuses more on spatial-dimension features, and a large version, i.e., A-DGC(l), which focuses on both spatial- and planar-dimension features. To better facilitate the subsequent channelwise interflow process to better extract features, we propose a $\gamma$-weighted shuffling module ($\gamma$-WSM), which shuffles the input channels by the corresponding $\gamma$ vectors obtained from the batch normalization (BN) layers. Optionally, to further reduce the complexity and size of the model, we introduce slimming strategies.

To summarize, the contributions of our work include the following.

(1) A more accurate and lightweight feature extractor, termed RoShuNet, is proposed.

(2) An A-DGC module is proposed for feature extraction purposes.

(3) A $\gamma$-weighted shuffling module ($\gamma$-WSM) is proposed and employed for aiding the channel interflow process.

(4) Slimming strategies that address the DGC and shortcut structures are introduced.

(5) Extensive experimental results demonstrate the SOTA performance of the proposed method in terms of classification accuracy and validate its comparable performance with respect to addressing semantic segmentation and multiple-object tracking tasks.

## 2. Related work

The rapid increase in GPU processing speeds has recently made memory consumption a priority when building lightweight CNNs. The widely used strategies for constructing memory-efficient lightweight networks can be divided into two main categories: structure design (before) and structure compression (after) approaches.

**DSC:** DSC is a strategy that has been readily adopted in many lightweight structures (structure design); it aims to directly construct memory-efficient CNNs. This strategy was first introduced by Chollet (2017), where a two-stage paradigm composed of a group convolution and a pointwise convolution was utilized, substantially reducing the size and complexity of the constructed model. Therefore, due to its lightweight and excellent performance, DSC was adopted for all subsequently developed lightweight FENs. Some recent efforts have been made to optimize the deployment of DSCs. For example, the recently proposed open-source MegEngine method (Ding et al., 2022) overcomes the processing inefficiency of PyTorch when increasing the DSC kernel size. However, these efforts are all optimized on the hardware side instead of in the methods themselves; on the other hand, as the second stage of the DSC approach is still a pointwise convolution, it still consumes a large amount of memory.

**Slimming and Quantization:** Slimming and quantization are two prevailing structure compression methods that aim to further compress models from memory-inefficient to memory-efficient versions. Slimming (Liu et al., 2017) mainly refers to removing model layers that are less important by relying on a specific criterion, and it has been widely used in many methods due to its high reproducibility. Other slimming methods, such as knowledge distillation (Liu et al., 2020; Wang and Yoon, 2022), have also been adopted. Quantization (Chang et al., 2021) is mainly used to enable real-time network operations to be implemented on side platforms. This method transforms floating points into fixed points, for example, by converting a model from FP32 to FP16 or INT8, making it a hardware-oriented technique.

The current ShuffleNetV2 (Ma et al., 2018) is a highly efficient hierarchical network that stacks three stages composed of shuffling blocks. With the help of the utilized DSC, it consequently achieves favorable classification accuracy with low memory overhead; hence, our method starts with ShuffleNetV2. To make our work self-contained, the structure of the original ShuffleNetV2 is briefly introduced here. ShuffleNetV2 is mainly composed of a certain number of shuffling blocks. Therefore, in Fig. 1, (i) all of the $\gamma$-weighted shuffling modules ($\gamma$-WSMs) and fusion modules in the $\gamma$-weighted shuffling blocks are removed, (ii) all of the aggregated shuffling blocks are accordingly replaced with shuffling blocks, and (iii) no slimming operations are performed. Then, we can acquire the original ShuffleNetV2 and determine how it works. At this point, as mentioned above, it is first necessary to execute the slimming operation to further reduce the model size. In addition, ShuffleNetV2 still has room for further improvement, as shown below.

**Channel Shuffling Operation:** The networks in the ShuffleNet series are codenamed based on the shuffling operations used; these operations 'randomly' shuffle (rearrange) feature maps in the channel dimension. However, shuffling layers merely adjust the order of all the feature maps into a fixed order during the implementation process, which is more like handcrafted rearrangement than a random or learnable shuffling process, as all of the feature maps fed into the shuffling layers are rearranged in the same manner.

To address the issues described above, we propose RoShuNet. The merits of our method are that the proposed RoShuNet has competitive and even better classification performance while being more lightweight than the existing approaches. Additionally, this method not only exhibits powerful feature extraction capabilities for classification tasks but also is effective for semantic segmentation and multiple-object tracking scenarios, attaining roughly the same accuracy as that of the dense ResNet18 and VGG16 networks while further reducing the complexity and size of the model. Our method provides a novel perspective for designing lightweight models.

**Table 1**
Detailed layer structure. Note that C, B, R, M, A and FC denote the convolution, batch normalization, ReLU activation, max pooling, average pooling, and fully connected layers, respectively. *S* refers to the location at which the slimming strategies are imposed, and the corresponding number of channels is automatically adjusted.

| Layer name | Operation | Number of input channels | Number of output channels | Feature size |
|---|---|---|---|---|
| PrM | C | 3 | 24 | 1/2 |
| | M | 24 | 24 | 1/4 |
| FES-1 | $\gamma$-WSB | 24 | 116 | 1/8 |
| | ASBs×3 | 116 | 116 | 1/8 |
| FES-2 | $\gamma$-WSB | 116 | 232 | 1/16 |
| | ASB×7 (lssssls) | 232 | 232 | 1/16 |
| FES-3 | $\gamma$-WSB | 232 | 232 | 1/32 |
| | SB×3 | *S* | *S* | 1/32 |
| PoM | C-B-R | *S* | 1024 | 1/32 |
| | A | 1024 | 1024 | 1/224 |
| FC | Fully Connected | 1024 | 1 | *n* |

## 3. RoShuNet

The overall structure of RoShuNet is shown in Fig. 1. An input image is needed, and it is first passed to the preprocessing module (PrM). The PrM is composed of a convolutional layer and a max pooling layer, as shown in the figure below. This module is used to acquire low-level features and retain the principal features while reducing the number of required parameters and the imposed computational burden to prevent overfitting. The low-level features are fed into the 3 subsequent feature extraction stages (termed FES-1, FES-2 and FES-3) to convert the input features into advanced semantic-class information. Each feature extraction stage is composed of two portions: a $\gamma$-weighted shuffling block ($\gamma$-WSB, which is a shuffling block with a $\gamma$-weighted shuffling module) and a certain number of aggregated shuffling blocks (ASBs, shuffling blocks with A-DGC) and shuffle blocks. A postprocessing module (PoM) is used to convert the extracted features into the expected class information at the back end. Additionally, the FES-3 and PoM layers are slimmed with the proposed slimming strategies to make the model more compact and lightweight, thus satisfying some special situations. The detailed layer structure is described in Table 1. Notably, for both FES-1 and FES-3, we exploit 3 blocks to extract features; the difference is that the blocks exploited in FES-1 are ASBs with the small A-DGC version (A-DGC(s)), and the blocks exploited in FES-3 are the shuffling blocks. In FES-2, we exploit 7 ASBs to expand the intermediate features; the first and sixth blocks are large blocks, and the remainder are small blocks. In the following subsections, we describe the A-DGC module, $\gamma$-weighted shuffling module and slimming strategies that only reside in RoShuNet.

### 3.1. Aggregated dual group convolution module

The A-DGC module is proposed because conventional convolution often has difficulty satisfying practical requirements due to the large amount of computations performed on devices with restricted computing resources. That is, feature extraction should be implemented using a smaller model capacity without compromising the performance of the model. Inspired by Hu et al. (2020), to compact the model without impairing its feature extraction ability, we propose large and small versions of the A-DGC module. This module extracts features from both the planar and spatial dimensions in lieu of the DSC method, which was previously widely used for lightweight model structuring. Fig. 2 illustrates the large and small versions of the A-DGC module, which are termed A-DGC(l) and A-DGC(s), respectively. From a horizontal perspective, A-DGC(l) focuses on both planar and spatial features and is composed of two streams, which are shown with solid and dashed arrows. The solid arrows illustrate the main feature extraction stream, while the dashed arrows illustrate the aggregation stream. Additionally, from a vertical perspective, A-DGC(l) is composed of two group convolutions; therefore, we term it a dual group convolution method. This is why we call the whole structure the aggregated dual group convolution module.

Specifically, the input feature map $\boldsymbol{F}_1 \in \mathbb{R}^{H\times W\times 3}$ is first passed to the 1st group convolution. The 1st group convolution chiefly operates a groupwise convolution over the inputs with a kernel size of 3 × 3 and then outputs $\boldsymbol{F}_2 \in \mathbb{R}^{H\times W\times 3}$. $\boldsymbol{F}_2$ is separately transmitted into two streams. For the aggregation stream (shown with dashed arrows in Fig. 2), we first concatenate all the feature maps of $\boldsymbol{F}_2$ and then apply a dense convolution operation with a kernel size of 1 × 1 to acquire more inter-channel information, thus obtaining $\boldsymbol{F}_2' \in \mathbb{R}^{H\times W\times 3}$. Considering model capacity, we use three different convolutions with kernel sizes of 1 × 1 to form three different feature combinations from the feature maps of $\boldsymbol{F}_2$, thereby obtaining more expected higher-level features in the planar dimension. Then, $\boldsymbol{F}_2'$ is concatenated, and another dense convolution with a kernel size of 1 × 1 is applied to squash the superfluous channels, obtaining $\boldsymbol{F}_2'' \in \mathbb{R}^{H\times W\times 3}$. Next, the features of all channels are aggregated from the squashed features $\boldsymbol{F}_2''$, producing $\boldsymbol{F}_3 \in \mathbb{R}^{H\times W\times 1}$. The aggregation method is accomplished by simply computing the mean value in a channelwise manner (CMean) while adhering to the following equation:

$$\boldsymbol{F}_3(i,j) = \frac{\sum_{k=1}^{N} \boldsymbol{F}_2''(i,j,k)}{N} \tag{1}$$

where $i$, $j$ and $k$ denote the indices $\boldsymbol{F}_3$ and $\boldsymbol{F}_2''$. Finally, we apply the last dense convolution with a kernel size of 1 × 1 to further enhance the formalization ability of the extractor, obtaining an aggregated feature map $\boldsymbol{F}_4 \in \mathbb{R}^{H\times W\times 1}$. The features in $\boldsymbol{F}_2$ are channel-isolated, and the channelwise information is discarded. It is necessary to recapture the channelwise information before the second group convolution is performed. Therefore, for the feature extraction stream (shown with solid arrows in Fig. 2), after obtaining feature map $\boldsymbol{F}_4$, it is blended with $\boldsymbol{F}_2$ to obtain $\boldsymbol{F}_2''' \in \mathbb{R}^{H\times W\times 3}$ by using the Hadamard product to rescale the channel-isolated features in $\boldsymbol{F}_2$. At this point, we apply a second group convolution with a kernel size of 1 × 1 over $\boldsymbol{F}_2'''$ to obtain the final feature map $\boldsymbol{F}_5 \in \mathbb{R}^{H\times W\times 3}$. For the small version of A-DGC, termed A-DGC(s) and shown in Fig. 2 (b), which focuses more on spatial information, we simply remove the operations after $\boldsymbol{F}_2'$ and perform the CMean operation directly over $\boldsymbol{F}_2'$. For instance, as shown in Fig. 2 (b), assuming that the current number of input and output channels is 128 (128 is widely accepted in the networking field), when exploiting A-DGC(s), the number of parameters that A-DGC(s) produces is 3 × 3×128+1+1 × 128 = 1281. In contrast, when exploiting the DSC that resides in the original ShuffleNetV2, the number of parameters that the DSC produces is 3 × 3×128+1 × 128×128 = 17536. Therefore, A-DGC(s) can also compress the network. Considering model efficiency, given that the feature maps are larger (4× downsampling) in the front-end (FES-1) stage, which will impair the model efficiency, and smaller (16× downsampling) in the FES-3 stage, which cannot produce the optimal feature combinations in the planar dimension, after forming a tradeoff between performance and efficiency, we specifically insert A-DGC(l) into FES-2.

### 3.2. $\gamma$-Weighted shuffling module

Due to the group convolution used, it is necessary to conduct shuffling over the feature maps in the channel dimension to facilitate channel interflow and achieve a better feature extraction effect. However, the shuffling method used in ShuffleNetV2, as described above, is strictly a nondifferentiated rearrangement of the feature maps rather than a true feature shuffling process, and ShuffleNetV2 neglects the feature information contained in the feature maps. Moreover, we argue that it is necessary to consider the traits of the feature maps when conducting feature shuffling instead of merely executing a simple shuffling operation in a disordered and messy manner (as demonstrated in

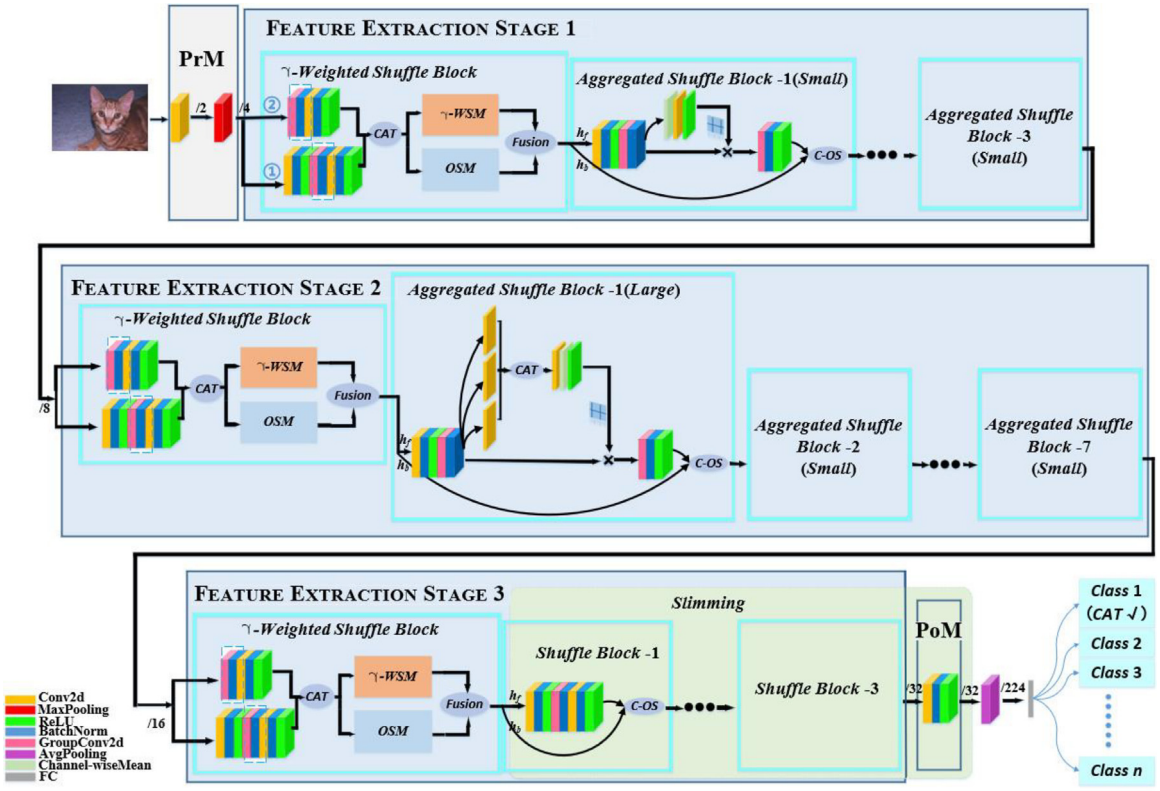
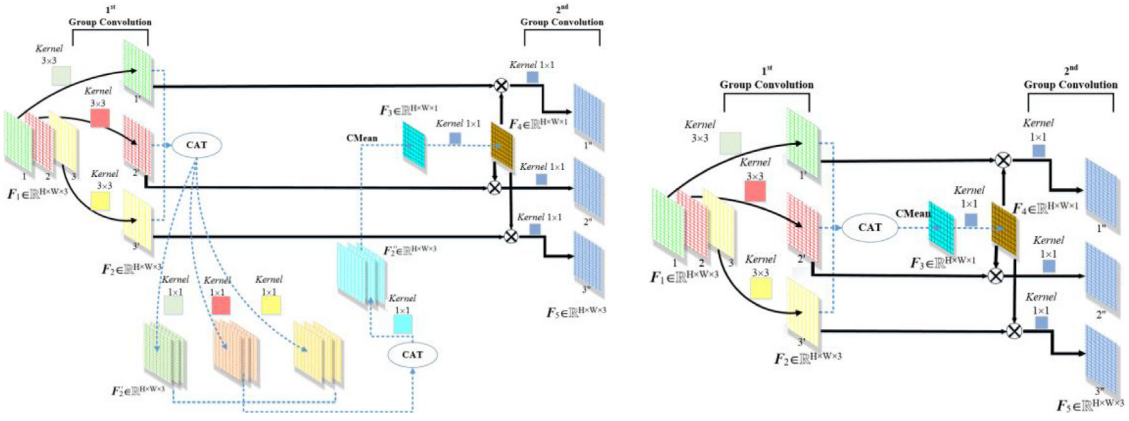



**Fig. 1.** The deployment workflow of RoShuNet for image classification. RoShuNet is composed of five parts, including the PrM; feature extraction stages 1, 2 and 3; and the PoM. The final output features are converted into class information. Note that the OSM represents the original shuffling module proposed in ShuffleNetV2, and C-OS indicates the combined operation consisting of concatenation and the OSM.




(a) A-DGC(l) (b) A-DGC(s)

**Fig. 2.** Illustration of the A-DGC process. The solid arrows represent the feature extraction stream, and the dashed arrows represent the aggregation stream.

Section 4.4, the traits in the feature maps impact the shuffling process). (1) Groups divided into different segments should not have relatively disparate importance weight levels. (2) More important channels containing more information can be employed to combine features more than once. Thus, we assign each feature map a scalar $\gamma$ representing its importance. Based on each $\gamma$, we propose the $\gamma$-WSM, as shown in Fig. 3. The $\gamma$-WSM can also be split into 2 streams according to the solid and dashed arrows displayed in the figure above. The stream shown with solid arrows is implemented over the feature maps, and the stream shown with dashed arrows is implemented over the importance scalar vectors. Specifically, the $\gamma$-WSM requires feature maps $\boldsymbol{G}_1 \in \mathbb{R}^{H\times W\times C1}$ and $\boldsymbol{G}_2 \in \mathbb{R}^{H\times W\times C2}$ along with their importance weight vectors $\boldsymbol{\gamma}_1 \in \mathbb{R}^{C1\times 1}$ and $\boldsymbol{\gamma}_2 \in \mathbb{R}^{C2\times 1}$, respectively, as inputs. First, $\boldsymbol{G}_1$ and $\boldsymbol{G}_2$ and their importance scalar vectors $\boldsymbol{\gamma}_1$ and $\boldsymbol{\gamma}_2$ are concatenated (CAT) to obtain the concatenated features $\boldsymbol{G}_3 \in \mathbb{R}^{H\times W\times (C1+C2)}$ and the importance vector $\boldsymbol{\gamma}_3 \in \mathbb{R}^{(C1+C2)\times 1}$. We sort $\boldsymbol{\gamma}_3$ in descending order and obtain $\boldsymbol{\gamma}_4 \in \mathbb{R}^{(C1+C2)\times 1}$. Then, we truncate $\boldsymbol{\gamma}_4$ at the center (half truncation) to obtain $\boldsymbol{\gamma}_5 \in \mathbb{R}^{(C1+C2)/2\times 1}$ (the first half) and $\boldsymbol{\gamma}_6 \in \mathbb{R}^{(C1+C2)/2\times 1}$ (the second half). $\boldsymbol{\gamma}_6$ is then reversed to form $\boldsymbol{\gamma}_7 \in \mathbb{R}^{(C1+C2)/2\times 1}$. At this point, the first shuffled importance vector $\boldsymbol{\gamma}_8 \in \mathbb{R}^{(C1+C2)\times 1}$ can be obtained by cross-piecing, which is implemented in a vectorwise and elementwise manner. Additionally, as the importance vector $\boldsymbol{\gamma}_5$ represents the feature maps possessing high importance levels, another shuffled importance vector $\boldsymbol{\gamma}_9 \in \mathbb{R}^{(C1+C2)\times 1}$ is thus produced by self-piecing. Then, the channels of $\boldsymbol{G}_3$ are aligned with the importance vectors $\boldsymbol{\gamma}_8$ and $\boldsymbol{\gamma}_9$, and the final shuffled features $\boldsymbol{G}_5$ are eventually obtained by merging the shuffled features $\boldsymbol{G}_{41} \in \mathbb{R}^{H\times W\times (C1+C2)}$ into $\boldsymbol{G}_{42} \in \mathbb{R}^{H\times W\times (C1+C2)}$. The merging method employed here is accomplished by using elementwise

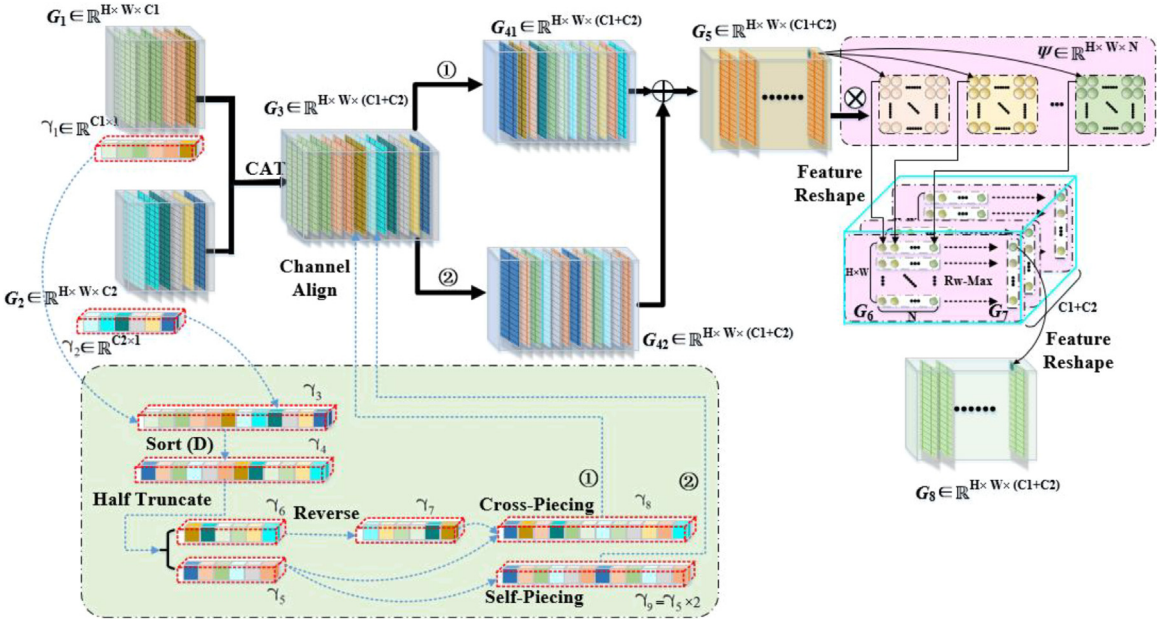



**Fig. 3.** Illustration of the $\gamma$-WSM. We add the CAT segment to the $\gamma$-WSM illustration for clarity. The importance vectors $\gamma_1$ and $\gamma_2$ of feature maps G1 and G2 correspond to the outputs of branches ① and ② in Fig. 1, respectively. Therefore, the real input of the $\gamma$-WSM is $G_3$, and the output is the shuffled feature map $G_8$. The whole shuffling process can also be divided into two streams, which are denoted above by the two different types of arrows (dashed and solid). The dashed arrows indicate the operations conducted over the importance vectors, and the solid arrows indicate the operations conducted over the feature maps; refer to the text for more details.

addition. At this point, $G_5 \in \mathbb{R}^{H\times W\times(C1+C2)}$ possesses features from both $G_{41}$ and $G_{42}$, and it is necessary to select the most discriminative feature at each feature location. Inspired by Goodfellow et al. (2013), we perform elementwise mapping operations $N$ times over $G_5$, and the mapping process adheres to the following equation:

$$G_{6:l,:,q} = \mathcal{F}(G_{5:i,j,q}, \Psi_{i,j,:}) \tag{2}$$

where $i$, $j$ and $q$ are the location indices of each feature. $\mathbf{F}(\cdot)$is the mapping function to be learned, and $\Psi \in \mathbb{R}^{H\times W\times N}$ represents its weights. $G_6 \in \mathbb{R}^{(H\times W)\times N\times(C1+C2)}$ is the output; therefore, $l$ is the index of the first dimension. The optimal mapped features of each row of $G_6$ are identified according to the equation given below:

$$y_{r,1,c} = f_{o(G_6)}(G_{6:c}) = \mathrm{Rw} - \underset{1\le k\le N}{\mathrm{Max}}(G_{6:r,k,c}) \tag{3}$$

where $c$ and $r$ are the channel and row indices, respectively, and $f_o(\cdot)$ is the optimal feature identification function. Here, we exploit the rowwise maximization (Rw-Max) operation in the implementation, and $y_{r,1,c} \in G_7 \in \mathbb{R}^{(H\times W)\times 1\times(C1+C2)}$ is the output. The final output of the $\gamma$-WSM $G_8 \in \mathbb{R}^{H\times W\times(C1+C2)}$ is obtained by adjusting the shape (i.e., applying feature reshaping) of $G_7$ back to that of the input.

As the aim of the $\gamma$-WSM is to further assist the channel interflow process, given that the A-DGC actually plays a role in channel interflow to a certain extent (via the blending method), we integrate the $\gamma$-WSM into the first block in each feature extraction stage, as shown in Fig . 1. In the $\gamma$-WSM, we retain the original shuffling module (OSM) of ShuffleNetV2 and fuse the shuffled channels acquired from the $\gamma$-WSM and OSM to form the final $\gamma$-WSB output. The fusion operation adheres to the following equation:

$$G = \eta \cdot G_8 + (1-\eta)\cdot G_{\mathrm{OSM}} \tag{4}$$

where $G_8$ and $G_{\mathrm{OSM}}$ are the outputs of the $\gamma$-WSM and OSM, respectively. $\eta$ is the balancing coefficient.

The whole process can be outlined as follows: sorting serves piecing, piecing guides the feature alignment process, and the aggregated features enable the selection of optimal features. At this point, one remaining issue is the acquisition of the importance weight vectors. As researched by Liu et al. (2017), the scale factor $\gamma$ of a BN layer can represent its importance, and this factor is given by:

$$BN_{\varpi,\rho}(z) = \varpi \cdot \hat{z} + \rho \tag{5}$$

where $\hat{z}$ is given by:

$$\hat{z} = \frac{z - E[z]}{\sqrt{Var(z) + \delta}} \tag{6}$$

where $z$ is the input. $\varpi$ and $\rho$ indicate the scaling and shift factors, respectively. We note that the output of the BN process is determined by the scaling and shift factors as well as the normalized data distribution. When all the data are normalized to a standard normal distribution, the scaling factor $\varpi$ has a greater impact on the output. At this point, a larger $\varpi$ represents a more distinct input feature selection process implemented via the convolution kernels (Liu et al. (2017)). This factor is finally chosen in our implementation for simplicity.

### 3.3. Slimming strategies

For special occasions requiring more lightweight models, we additionally apply a slimming operation in the proposed method. Our slimming operation is based on the work of Liu et al. (2017) but has two differences. (i) In the research of Liu et al. (2017), only dense convolution was considered. In our method, as we adopt a certain number of group convolutions, the operation proposed by Liu et al. (2017) cannot be exploited to slim our network. To address this issue, we introduce slimming strategy 1, namely, channel adjustment. (ii) Liu et al. (2017) proposed their method for networks with plain structures, such as the Visual Geometry Group (VGG) series. In RoShuNet, which is stacked with blocks, it is inappropriate to use the vanilla application (Liu et al., 2017) for our method. To resolve this issue, we introduce slimming strategy 2, namely, block articulation.

**Strategy 1: Channel adjustment.** To minimize the number of parameters that group convolution produces, we apply the group convolution operation in a channelwise manner, as mentioned above. It is inappropriate to exploit the approach of Liu et al. (2017), as their method would chop the number of channels before and after each group convolution into two different sizes, which goes against the initial intention

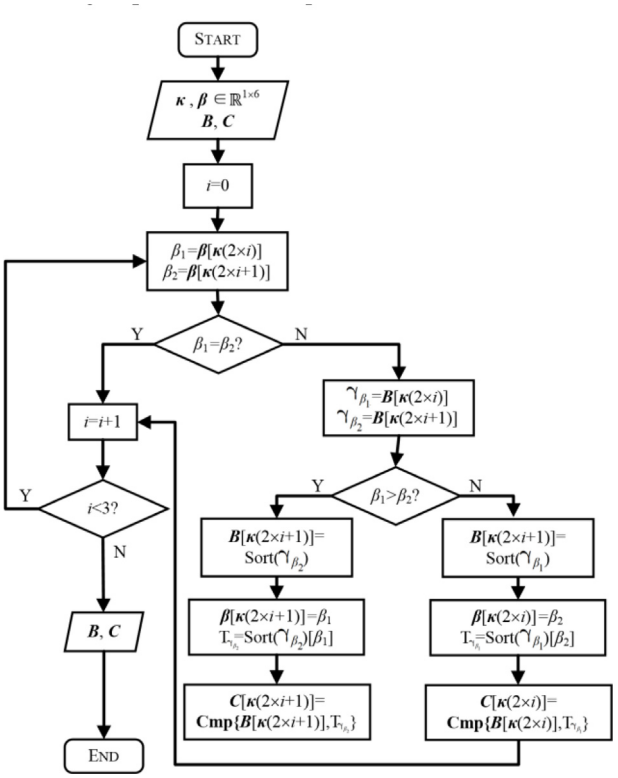



**Fig. 4.** A flowchart of the channel adjustment strategy. $\boldsymbol{B}$ and $\boldsymbol{C}$ are two containers in which the $\gamma$ vector and the corresponding mask vector of each BN layer are stored, respectively. $\kappa\in\mathbb{R}^{1\times 6}$ represents a row vector that stores the indices of the layers awaiting channel adjustment, and $\beta\in\mathbb{R}^{1\times 6}$ represents a row vector that stores the channels remaining after slimming. **Cmp** indicates the elementwise comparison operation.

of our network structure. To solve this issue, we propose a channel adjustment strategy. We first locate the group convolution in the whole pipeline, utilize a container $\boldsymbol{B}$ to store the $\gamma$ vectors of all BN layers of the unslimmed model, and use a container $\boldsymbol{C}$ to store the mask vectors accordingly (similarly hereinafter). The mask vectors refer to a set of binary vectors composed of only 0s and 1s, where 0s indicate that the corresponding BN layers should be ignored and removed. Furthermore, we utilize a row vector $\kappa\in\mathbb{R}^{1\times 6}$ to store all the BN locations before and after each group convolution in the last three shuffling blocks of FES-3 and employ $\beta\in\mathbb{R}^{1\times 6}$ to store the channels that should remain after performing slimming ($\beta$ is initialized with a threshold imposed over all layers of the model). A flowchart of the channel adjustment strategy is shown in Fig. 4. The basic idea is to compare the numbers of remaining channels before and after each group convolution is implemented and enlarge the smaller channel group and its slimming configuration to keep the numbers of channels that are present before and after each group convolution equal.

As shown in Fig. 4, the full number of iteration cycles is 3, as we only slim FES-3 in the current version. During each iteration, we first acquire the $\beta_1$ and $\beta_2$ of each group convolution, which indicate the numbers of remaining layers filtered according to the given threshold. If $\beta_1 = \beta_2$, we proceed to the next iteration; otherwise, we return to the $\gamma$ vectors ($\gamma_{\beta 1}$, $\gamma_{\beta 2}$) corresponding to the current location. If $\beta_1 > \beta_2$, we adjust the configurations corresponding to $\beta_2$. Specifically, we exclusively re-sort $\gamma_{\beta 2}$ in descending order, set the $\beta_1^{st}$ value in the re-sorted vector as the new threshold, and then adopt a new threshold to binarize the corresponding mask vectors in $\boldsymbol{C}$. If $\beta_2 > \beta_1$, the same operation is performed on the corresponding configurations of $\beta_1$. After performing channel adjustment, we can further refine the mask vectors $\boldsymbol{C}$ that are finally used to slim the model.

**Strategy 2: Block articulation.** After adjusting the channel configuration using strategy 1, the method of Liu et al. (2017) still cannot be exploited to conduct slimming due to the bypath structure between each pair of blocks. We thereby propose block articulation, which enables the method of Liu et al. (2017) and a bypath structure to be bridged. Block articulation refers to connecting the locations of two (shuffling) blocks. Similarly, a container $M_0$ is employed to store the kernel parameters of the convolution of the unslimmed model, and $M_1$ is used to accordingly store the parameters of the slimmed model. $\nu\in\mathbb{R}^{1\times 2}$ is a row vector that stores the articulation locations of the model. A flowchart of the block articulation process is shown in Fig. 5 . The main idea is to first investigate the numbers of input and output channels in each SB and then transplant the parameters of the unslimmed model to the slimmed model. Since only three shuffling blocks are contained in the slimming region (shown in Fig. 1) and the location of SB-1 is frontally articulated to the $\gamma$-WSM for good measure, two iterations are thus needed in total. For the first iteration, after obtaining the number of input channels in SB-2 and the number of output channels in the $h_f$ branch of SB-2, termed $\zeta_{2in}$ and $\mu_{2out}$, respectively, the preindexing range $\alpha$ for the next block articulation step is computed according to the following equation:

$$\alpha=\frac{\zeta_{2in}}{2}+\mu_{2out} \tag{7}$$

The indexing range of the current articulation procedure is therefore acquired by $\varphi=\lfloor \alpha/2 \rfloor$, where $\lfloor\cdot\rfloor$ indicates the rounding down operation. Next, the mask vector corresponding to the current articulation location is obtained. The obtained mask vector and the calculated $\varphi$ are used to transplant the parameters of the unslimmed model to the slimmed model. Because the number of input channels ($\zeta_{3in}$) in SB-3 is arbitrary due to the slimming process, we additionally judge the parity of the input channels in the second cycle. The remaining operations are consistent with those used previously. The remaining operations of the second iteration are the same as those employed before and are not repeated here.

**Network penalty.** The loss incurred by RoShuNet when applying slimming strategies comprises the three components given in the following equation:

$$L_{W/S}=\sum_{i=1}^{N_c} L_c(\boldsymbol{W}^{(i)}\otimes \boldsymbol{X}^{(i)},t)+\vartheta_n\sum_{i=1}^{N_n}\left\|\gamma^{(i)}\right\|_{L_1}+\vartheta_c\sum_{i=1}^{N_c}\left\|\mathrm{W}^{(i)}\right\|_{L_2} \tag{8}$$

where $\boldsymbol{W}^{(i)}$, $\boldsymbol{X}^{(i)}$ and $t$ denote the trainable weights, input feature maps and target vector, respectively. The superscript denotes the layer index. $\vartheta_n$ and $\vartheta_c$ are two coefficients that balance each penalty component. The first component originates from the backbone. In the implementation, we exploit the cross-entropy penalty. As researched by Liu et al. (2017), the model to be slimmed must first be sparsely pretrained to attain better performance; $\mathcal{L}_1$ normalization is therefore exploited as the second component of the network penalty. To prevent the network from overfitting, $\mathcal{L}_2$ normalization is used as the third component. After completing the pretraining process, it is necessary to fine-tune the slimmed model. During fine-tuning, the sparsity penalty is no longer needed. When the slimming strategies are not adopted, the loss ($\mathcal{L}_{\mathrm{WO/S}}$) is $\mathcal{L}_{\mathrm{W/S}}$ without the second sparsity component.

## 4. Experiments

In this section, we elaborate on the details of the comparison experiments, ablation experiments, and generalization experiments performed to investigate the feature extraction capabilities of the tested methods. Additionally, we reveal and compare other shuffling implementations, and to investigate the effectiveness resulting from the input image resolution, another group comparison experiment is carried out.

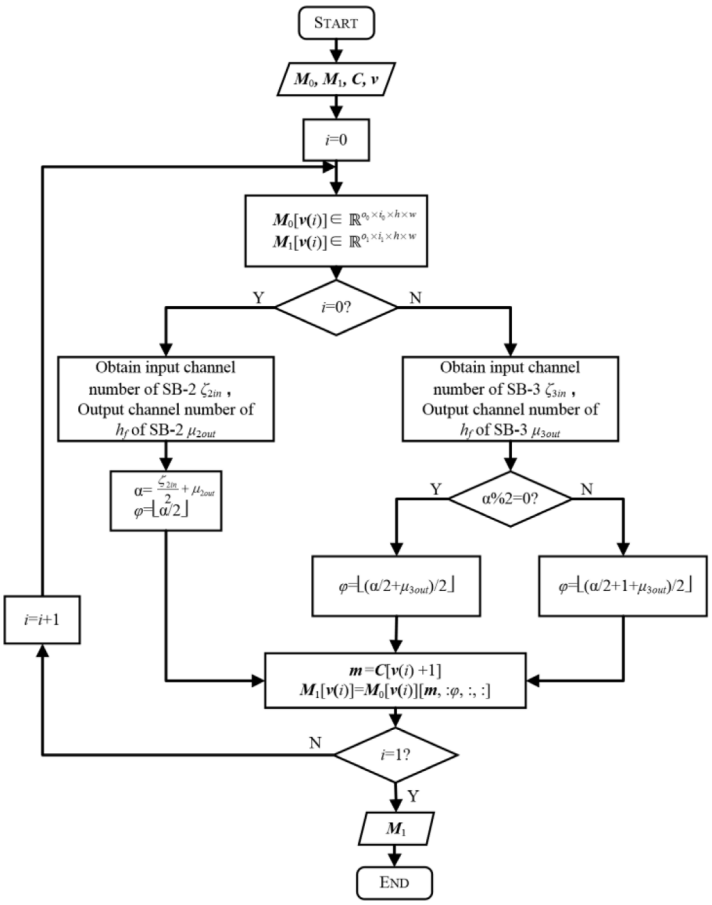



**Fig. 5.** A flowchart of the block articulation strategy. $M_0$, $M_1$, $C$ and $v$ indicate a container for storing the convolution parameters of the unslimmed model, a container for storing the convolution parameters of the slimmed model, a container for storing the mask vectors of the BN layer of the unslimmed model and a row vector for storing the articulation locations, respectively.

### 4.1. Comparison experiments

*Experimental Design*: To investigate the performance of our method compared to that of the baseline ShuffleNetV2 and other SOTA feature extraction methods, we first utilize the classification task to conduct a group of comparison experiments. For a fair comparison, the included comparison methods are all lightweight SOTA methods built on top of pure CNNs that have been validated in the literature in recent years as having excellent performance, including the MobileNet series (Howard et al., 2017, 2019; Sandler et al., 2018), the ShuffleNet series (Ma et al., 2018; Zhang et al., 2018), EffNet (Freeman et al., 2018) and GhostNet (Han et al., 2020).

*Datasets*: CIFAR-100,[1] Kaggle-CatsvsDogs,[2] Caltech-256 (Griffin et al., 2007) and miniImageNet (Vinyals et al., 2016) are the datasets adopted to quantitatively compare the tested methods, and they vary in terms of their image resolutions, object sizes and numbers of classes. Specifically, we first adopt the widely used CIFAR-100 and Caltech-256 datasets for comparison. The CIFAR-100 dataset comprises $6 \times 10^4$ red − green − blue (RGB) images with resolutions of $32 \times 32$. This dataset contains 20 superclasses and 100 subclasses. The Caltech-256 dataset comprises $3.06 \times 10^4$RGB images containing 257 classes with different (high) resolutions. EffNet, which was designed with limited classes, is adopted for comparison purposes. Therefore, the standardized Kaggle-CatsvsDogs dataset with 2 classes provided by the Kaggle community is also exploited to determine the effectiveness of each method in terms of its ability to address cases with limited classes. Kaggle-CatsvsDogs comprises $2.5 \times 10^4$ RGB images of different breeds of cats and dogs acquired under different scenes with high resolutions. Since Kaggle-CatsvsDogs does not provide annotation information for its testing set, 30% of the images are acquired from the training set and used as the testing set in our experiments. The miniImageNet dataset comprises $6 \times 10^4$ RGB images containing 100 classes with different (high) resolutions. Each class has 600 samples. In our experiments, we randomly split miniImageNet into training, validation and testing sets at a ratio of 6:1:3. Therefore, the numbers of data points contained training, validation and testing sets are $3.6 \times 10^4$, $0.6 \times 10^4$ and $1.8 \times 10^4$, respectively. The scenarios in the miniImageNet dataset are more complex than those in the other datasets and therefore valuable for conducting experimental studies as well.

*Training Protocols*: All the methods involved in the comparison experiments are trained for 550 epochs under CIFAR-100, 250 epochs under Caltech-256, 300 epochs under Kaggle-CatsvsDogs and 300 epochs under miniImageNet with the stochastic gradient descent (SGD) optimizer. All models, similar to many other reported methods (Liu et al., 2021), are trained without bells or whistles. The image size and batch size are set to $160 \times 160$ and 128, respectively, with the current GPU memory. The learning rate is set to 0.01, the balancing coefficient $\vartheta_c$ in Eq. (8) is set to 5e-4, and $\eta$ in Eq. (4) is set to 0.2 throughout the training process.

*Experimental Results*: The results of comparison experiments conducted on the four datasets described above are listed in Table 2. These results are divided into four groups that first rely on the employed datasets; then, each group is further divided according to the model sizes (KParams). The metrics we adopt to quantitatively evaluate the methods are accuracy, KFLOPs and KParams, which denote the correct

[1] http://www.cs.toronto.edu/~kriz/cifar.html
[2] https://www.kaggle.com/c/dogs-vs.-cats-redux-kernels-edition

**Table 2**
Comparison among the obtained experimental results. The results produced of ShuffleNetV2 and our RoShuNet for each subgroup are reported in bold font. ↑ indicates that higher values are expected, and ↓ indicates the opposite expectation.

| Dataset | Model | Accuracy (↑, %) | KFLOPs (↓) | KParams (↓) |
|---|---|---|---|---|
| CIFAR-100 | MobileNetV2-0.7X | 60.08 | 83 166.95 | 1319.08 |
| | MobileNetV3-Small | 60.67 | 32 203.52 | 1356.85 |
| | ShuffleNetV2-1.0X | **59.12** | **75 350.02** | **1356.08** |
| | RoShuNet-1.0X | **60.76** | **74 244.00** | **1254.19** |
| | MobileNetV2-1.0X | 61.35 | 159 745.60 | 2351.97 |
| | ShuffleNetV2-1.5X | **60.15** | **153 149.02** | **2581.10** |
| | RoShuNet-1.5X | **63.63** | **150 414.00** | **2344.66** |
| | MobileNetV3-Large | 64.94 | 131 050.30 | 3992.42 |
| | GhostNet | 65.75 | 76 724.43 | 4030.12 |
| | ShuffleNetV2-2.0X | **66.80** | **322 502.95** | **6520.07** |
| | RoShuNet-2.0X | **65.00** | **317 051.32** | **6063.68** |
| Kaggle-CatsvsDogs | EffNet | 82.95 | 153 088.00 | 314.98 |
| | MobileNetV2-0.7X | 90.53 | 83 041.51 | 1193.54 |
| | MobileNetV3-Small | 88.80 | 32 078.08 | 1231.31 |
| | ShuffleNetV2-1.0X | **91.48** | **75 249.67** | **1255.63** |
| | RoShuNet-1.0X | **91.83** | **74 143.65** | **1153.74** |
| | MobileNetV2-1.0X | 91.94 | 159 620.16 | 2226.43 |
| | ShuffleNetV2-1.5X | **91.98** | **153 048.67** | **2480.65** |
| | RoShuNet-1.5X | **92.86** | **150 313.65** | **2244.21** |
| | MobileNetV3-Large | 93.32 | 130 924.86 | 3866.88 |
| | GhostNet | 91.64 | 76 598.99 | 3904.58 |
| | ShuffleNetV2-2.0X | **94.32** | **322 214.04** | **6231.07** |
| | RoShuNet-2.0X | **94.03** | **316 762.42** | **5774.68** |
| Caltech-256 | EffNet | 38.01 | 179 200.00 | 26 427.23 |
| | MobileNetV2-0.7X | 47.67 | 83 367.91 | 1520.20 |
| | MobileNetV3-Small | 47.59 | 32 404.48 | 1557.97 |
| | ShuffleNetV2-1.0X | **41.10** | **74 410.79** | **1517.01** |
| | RoShuNet-1.0X | **42.37** | **74 404.77** | **1415.11** |
| | MobileNetV2-1.0X | 49.91 | 159 946.56 | 2553.09 |
| | ShuffleNetV2-1.5X | **44.00** | **153 309.79** | **2742.03** |
| | RoShuNet-1.5X | **45.68** | **150 574.77** | **2505.58** |
| | MobileNetV3-Large | 50.70 | 131 251.26 | 4193.54 |
| | GhostNet | 47.26 | 76 925.39 | 4231.23 |
| | ShuffleNetV2-2.0X | **43.40** | **322 965.78** | **6983.07** |
| | RoShuNet-2.0X | **46.48** | **317 514.16** | **6526.67** |
| miniImageNet | EffNet | 38.05 | 163 123.20 | 10 350.28 |
| | MobileNetV2-0.7X | 56.48 | 83 166.95 | 1319.08 |
| | MobileNetV3-Small | 52.33 | 32 203.52 | 1356.85 |
| | ShuffleNetV2-1.0X | **52.94** | **75 350.02** | **1356.08** |
| | RoShuNet-1.0X | **55.42** | **74 244.00** | **1254.19** |
| | MobileNetV2-1.0X | 59.48 | 159 745.60 | 2351.97 |
| | ShuffleNetV2-1.5X | **55.13** | **153 149.02** | **2581.10** |
| | RoShuNet-1.5X | **60.29** | **150 414.00** | **2344.66** |
| | MobileNetV3-Large | 57.89 | 131 050.30 | 3992.42 |
| | GhostNet | 62.69 | 76 724.43 | 4030.12 |
| | ShuffleNetV2-2.0X | **61.18** | **322 502.95** | **6520.07** |
| | RoShuNet-2.0X | **61.52** | **317 051.32** | **6063.68** |

classification rate, model complexity and model size, respectively. As reported in Table 2, we note that (i) overall, our RoShuNet (under the 1.0X and 1.5X versions) outperforms ShuffleNetV2 in terms of accuracy, KFLOPs, and KParams on the four datasets. RoShuNet_2.0X yields better results than ShuffleNetV2_2.0X on the Caltech-256 and miniImageNet datasets and achieves comparable results to ShuffleNetV2 on the Cifar100 and Kaggle datasets. (iii) Compared with the other SOTA methods, the RoShuNet series yields competitive results in terms of not only accuracy but also model size and model complexity. (iv) The MobileNet series yields better performance on the Caltech-256 dataset, and GhostNet yields the best results on the miniImageNet dataset. Furthermore, we note that EffNet increases to approximately 84 times the model size (from 314.98 to 26427.23) and approximately 1.2 times the model complexity (from 153088 to 179200) relative to the version utilized for Kaggle-CatsvsDogs while achieving only 38.01% accuracy.

### 4.2. Ablation experiments

*Experimental Design*: To determine how each newly proposed module affects the final results, we conduct ablation experiments by gradually adding these modules, producing variants including ShuffleNetV2_A (ShuffleNetV2 with the A-DGC module), ShuffleNetV2_$\gamma$ (ShuffleNetV2 with the $\gamma$-WSM), and RoShuNet (ShuffleNetV2 with the A-DGC module and $\gamma$-WSM). Additionally, we remove all OSMs from Fig. 1 to investigate the effectiveness of this module and term the model developed under this configuration "ShuffleNetV2_B". The training protocols remain consistent with those used in the comparison experiments. We also apply slimming strategies to RoShuNet (termed *s*-RoShuNet) and set the balance coefficients $\vartheta_n$ in Eq. (8) to 1e-4 with the help of previous research (Liu et al., 2017); the slimmed models are additionally fine-tuned for 30 epochs after training.

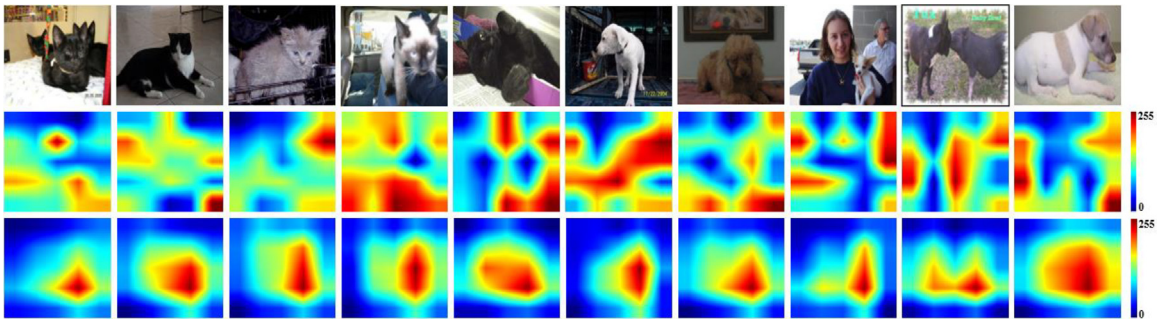



**Fig. 6.** Visualization of the location and extraction abilities of different models via heatmaps. The first row represents 10 groups of original images. The maps in the second row are all yielded by ShuffleNetV2, and the maps in the last row are all yielded by RoShuNet.

**Table 3**
Experimental ablation results obtained on the CIFAR-100 dataset. The best results are reported in bold font. The methods used in the ablation experiments are the 1.0X versions. The best results are reported in bold font.

| Model | Accuracy (↑, %) | KFLOPs (↓) | KParams (↓) |
|---|---|---|---|
| ShuffleNetV2 | 59.12 | 75 350.02 | 1356.08 |
| ShuffleNetV2_A | 60.25 | 74 244.00 | 1254.19 |
| ShuffleNetV2_B | 50.87 | 75 350.02 | 1356.08 |
| ShuffleNetV2_$\gamma$ | 59.82 | 75 350.02 | 1356.08 |
| RoShuNet | **60.76** | 74 244.00 | 1254.19 |
| $s_a$-RoShuNet | 60.01 | 72 857.05 | 1200.55 |
| $s_b$-RoShuNet | 59.42 | **67 776.22** | **997.71** |

*Experimental Results*: The results of the ablation experiments are listed in Table 3. We note that (i) when using ShuffleNetV2_A and ShuffleNetV2_$\gamma$, the accuracies of these two models are greater than that of ShuffleNetV2; (ii) when using RoShuNet, the accuracy is further enhanced and reaches the peak values (60.76%), and the model complexity (indicated by KFLOPs) and model size (indicated by KParams) both decrease compared to those of ShuffleNetV2; (iii) when removing the OSMs (ShuffleNetV2_B), the accuracy is substantially degraded; and (iv) when applying the proposed slimming strategies to RoShuNet, the accuracies of the slimmed RoShuNet versions ($s_a$-RoShuNet and $s_b$-RoShuNet, which are slimmed with two different thresholds $a = 0.1059$ and $b = 0.1371$, respectively) remain greater than that of ShuffleNetV2. Moreover, the model complexity is further decreased, with relative decreases of 3.3% and 10.1%, and the model size is decreased, with relative decreases of 11.5% and 26.4%, respectively.

Furthermore, when we adopt Kaggle-CatsvsDogs to perform the experiments, we find that the feature maps produced after using A-DGC can more precisely aggregate around the expected targets. We intuitively visualize the output feature maps yielded by SB-7/ASB-7 in FES-2 on Kaggle-CatsvsDogs to qualitatively investigate the contributions of the proposed A-DGC(l) and A-DGC(s) modules to the feature extraction process directly from their outputs. For ease of observation, we employ heatmaps to characterize the location and extraction abilities of the models, as shown in Fig. 6. We note that (i) as shown in the second row of Fig. 6, the location ability of ShuffleNetV2 seems implausible in some cases. However, RoShuNet (in the third row) can precisely locate the targets. (ii) As shown in the second row of Fig. 6, ShuffleNetV2 seems arbitrary in terms of its target feature extraction process at times. In contrast, RoShuNet can not only precisely extract the features of targets but also focus more on the more distinctive areas within the regions of interest (shown in orange and red in Fig. 6), such as faces and limbs. Unfortunately, the features produced by RoShuNet on CIFAR-100 and Caltech-256 do not possess this trait.

### 4.3. Generalization experiments

*Experimental Design*: To ascertain the feature extraction ability of our method for use in other tasks, we conduct two additional groups of generalization experiments involving two currently prevailing tasks, namely, semantic segmentation (SS) and multiple-object tracking (MOT). Specifically, we deploy DeepLabV3 (Chen et al., 2017) to address SS and a deep affinity network (DAN) (Sun et al., 2021) to address MOT, as these two methods have demonstrated SOTA performance in these respective tasks and accomplished feature extraction with CNNs. When implemented, to make RoShuNet compatible with DeepLabV3 and the DAN, RoShuNet undergoes some adjustments. For DeepLabV3, we remove the operations after the PoM (including the PoM) from RoShuNet, term the remaining part of RoShuNet RS, and let RS be the feature extraction backbone of DeepLabV3. For the DAN, (i) similarly, we utilize RS as the backbone to replace the VGG16 architecture used in the DAN; (ii) we feed the features output by RS into the extension part of the DAN; and (iii) we also collect the output features of FES-1, FES-2 and FES-3 of RS and let them be the input features of the first three feature dimensionality reduction layers of the DAN. Additionally, before feeding the features to the feature dimensionality reduction layers, the channel width values are correspondingly adjusted.

*Dataset*: *SS-PascalVOC2012*.[3] PascalVOC2012 provides a set of standardized and excellent data for conducting supervised learning in visual tasks. This dataset contains four major classes and twenty subclasses and can be used for visible image classification, object detection, and visible image segmentation tasks. For the segmentation task, PascalVOC2012 provides pixel-level segmentation masks for each object and class. The PascalVOC2012 training set employed for the segmentation task contains 10582 images, and the testing set contains 1449 images with different resolutions.

*Dataset*: *MOT*. MOT15/MOT17.[4] The MOT dataset series is widely used in multiple-object tracking scenarios. The MOT dataset mainly aims to provide more challenging visible videos for pedestrian and automobile tracking methods, such as clustered scenes and scenes with occlusions. We adopt the detection results produced by FRCNN (Ren et al., 2017) on MOT17 to finish the training task. As the annotations for the testing set are not publicly available, the videos in the MOT15 training dataset that exist in the MOT17 training set are removed and adopted as the testing set. The details of MOT17 and MOT15 with respect to our implementation are listed in Table 4. Each column in Table 4 presents the total numbers of videos and frames (videos/length), the total numbers of annotated boxes and trajectories (GTBoxes/trajectories), the object density of each frame (density) and the total number of boxes that are detected (DetBoxes).

*Training Protocols*: For semantic segmentation, the models are trained for 30 epochs under the SGD optimizer. The batch size is set to 16, and the initial learning rate ($lr_0$) is set to 2.5e−4. As the number of

[3] http://host.robots.ox.ac.uk/pascal/VOC/
[4] https://motchallenge.net/

**Table 4**
Attributes of the MOT15 and MOT17 training datasets.

| Phase | Series | Videos | Length | GTBoxes | Trajectories | Density | DetBoxes |
|---|---|---|---|---|---|---|---|
| Training | MOT17 Training Set | 7 | 5316 | 204 701 | 796 | 34.9 | 67 639 |
| Testing | MOT15 Training Set | 8 | 3993 | 30 589 | 386 | 6.9 | 30 436 |

**Table 5**
Quantitative experimental results obtained on a semantic segmentation task. The best results are reported in bold font.

| Model | MIoU (↑, %) | GFLOPs (↓) | MParams (↓) |
|---|---|---|---|
| DeepLabV3 (ResNet18) | 32.24 | 6.63 | 9.74 |
| DeepLabV3 (ResNet50) | 33.05 | 27.01 | 39.05 |
| DeepLabV3 (RS-1.0X) | 35.29 | **5.29** | **17.17** |
| DeepLabV3 (RS-1.5X) | 37.84 | 6.20 | 18.50 |
| DeepLabV3 (RS-2.0X) | **39.28** | 7.58 | 20.43 |

**Table 6**
Quantitative experimental results obtained for multiple-object tracking. Note that the RS used is the 1.0X version. Substantially improved metrics are reported in bold font.

| **Model** | **MOTP** (↑, %) | **IDs** (↓) | **IDF1** (↑, %) |
|---|---|---|---|
| DAN (VGG16) | 63.74 | 1979 | 45.29 |
| DAN (RS) | 63.76 | **1908** | **47.74** |
| **Model** | **FM** (↓) | **MOTA** (↑, %) | **CoT** (↑, %) |
| DAN (VGG16) | 1611 | 34.09 | 77.08 |
| DAN (RS) | **1598** | **34.28** | 77.09 |
| **Model** | **SMOTA** (↑, %) | **GFLOPs** (↓) | **MParams** (↓) |
| DAN (VGG16) | 57.96 | 297.37 | 25.79 |
| DAN (RS) | **58.50** | **25.97** | **15.86** |

epochs increases, the learning rate ($lr$) decreases according to $lr{=}lr_0$-$[1\text{-}(eph \times \max(itrs){+}itrs)/(30{\times}\max(itrs))]^{0.9}$, where $eph$ and $itrs$ denote the current number of epochs and the current number of iterations, respectively. The balance coefficient $\vartheta_c$ is set to 1e-4. All the input images are resized to resolutions of 513 × 513; for MOT, the models are trained for 20 epochs. The batch size and initial learning rate ($lr_0$) are set to 5 and 1e-2, respectively, and the $lr$ decreases according to $lr = lr_0 \times 0.1^{\mu}$, $\mu{\in}\{1, 2, 3\}$ every 5 epochs. The balance coefficient $\vartheta_c$ is set to 5e-4. All the input images are resized to resolutions of 650 × 650.

*Experimental Results Obtained for Semantic Segmentation*: The metric adopted to evaluate the segmentation results is the mean intersection over union (MIoU), which ranges from 0 to 1 and indicates the achieved segmentation performance in a coarse-to-fine manner. The experimental results are listed in Table 5. We note that (i) the DeepLabV3 versions backboned with three versions of RS all outperform the version of DeepLabV3 that is backboned with ResNet50 in terms of the MIoU metric, quantitatively yielding increases of 2.24%, 4.79% and 6.23% over the values produced by DeepLabV3 backboned with ResNet50; and (ii) the complexity levels (GFLOPs) and model sizes (MParams) of the DeepLabV3 versions backboned with RS are lower than those of DeepLabV3 (ResNet50). The model complexity produced by the largest version, DeepLabV3 (RS-2.0X), decreases from 27.01 for DeepLabV3 (ResNet50) to 7.58, and the model size produced by the largest version, DeepLabV3 (RS-2.0X), decreases from 39.05 for DeepLabV3 (ResNet50) to 20.43, yielding relative decreases of 72% and 48%, respectively.

*Experimental Results Obtained for Multiple-Object Tracking*: The metrics adopted to evaluate the MOT results are IDF1, IDS, FM, MOTA and MOTP from Bernardin and Stiefelhagen (2008) and Ristani et al. (2016) and CoT and SMOTA from Zhang et al. (2021)). Importantly, the last two metrics mainly characterize the continuity of the produced trajectories and the synthesized performance, enabling us to determine the overall performance of a tracker. The experimental results obtained for MOT are listed in Table 6. From the table, we note that (i) the DAN backboned with RS outperforms the DAN backboned with VGG16 in terms of the MOTA and SMOTA results. For the remainder of the metrics in Table 6, DAN (RS) yields comparable results to those of DAN (VGG16); (ii) the model complexity and model size are considerably reduced when backboning RoShuNet, and quantitatively, the model complexity and model size of DAN (RS) are just 8.7% and 61.5% of those of DAN (VGG16), respectively.

**Table 7**
Results of a shuffling-based ablation study conducted using different implementations. Note that S is ShuffleNetV2, and Asc and Dsc indicate ascending and descending, respectively. √ indicates that the corresponding operation is selected. The best results are reported in bold font.

| Shuffle model | Sorting | Piecing | Accuracy |
|---|---|---|---|
| S | – | – | 59.12 |
| S1 | Asc | | 57.54 |
| S2 | Dsc | | 59.23 |
| S3 | Dsc | ✓ | 59.53 |
| S4 | Asc+Dsc | | 58.84 |
| S5 | Asc+Dsc | ✓ | 58.95 |
| $\gamma$-WSM | Dsc | ✓ | **59.82** |

### 4.4. Shuffling with feature importance

As stated in Section 3.2, the traits existing in feature maps should be considered when shuffling channels. Table 7 lists some results investigated under different shuffling implementations using importance vectors. The difference between the $\gamma$-WSM and S1, S2, S3, S4 and S5 is that all of the latter modules lack mappings at the back end. For detailed implementations of the different models, please refer to Table 7.

As shown in Table 7, we note that (i) the worst accuracy is achieved when shuffling according to the importance vectors in ascending order, and this value is even worse than that of ShuffleNetV2; (ii) the accuracy is improved and starts to exceed that of ShuffleNetV2 when shuffling in descending order; and (iii) when shuffling the feature maps in descending order of the importance vectors along with piecing, the accuracy continues to rise, but when the ascending implementation is used together with the descending order, the accuracy worsens again. When further appending the piecing process, the accuracy exhibits a slight rebounding trend but is still below that of ShuffleNetV2. Therefore, we eventually choose the descending and piecing paradigm to shuffle the feature maps. Additionally, to select the optimal feature at each feature location, we perform the mapping operation at the back end, as stated previously.

### 4.5. Correlation experiments regarding the resolution

To investigate the correlations between the accuracy and resolution and between the model complexity and resolution, additional groups of correlation experiments are conducted by changing the resolution of the input images. The dataset used is Caltech-256. The number of training epochs is set to 50, and the remaining training protocols are all consistent with those used in the comparison experiments.

The experimental results obtained regarding the correlation between the accuracy and resolution are presented as a bar chart in Fig. 7. We note that (i) when the resolution changes from low to high, our proposed method always yields better accuracy than that of the original ShuffleNetV2; (ii) notably, our method can also yield better accuracy when the resolution is increased, meaning that increasing the

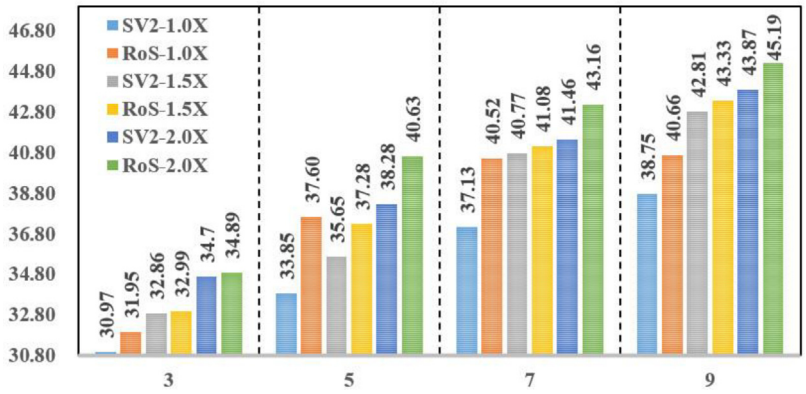



**Fig. 7.** A bar chart that presents the obtained experimental results for quantifying the correlation between the accuracy and resolution. SV2 is ShuffleNetV2, and RoS is RoShuNet.

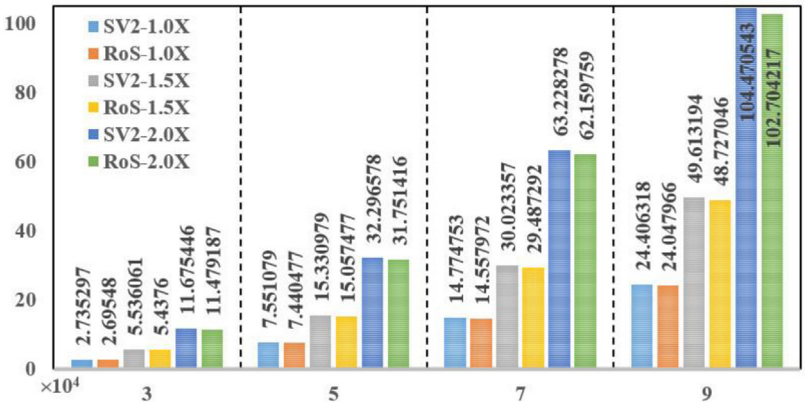



**Fig. 8.** A bar chart presenting the obtained experimental results for quantifying the correlation between the model complexity (KFLOPs) and resolution. SV2 is ShuffleNetV2, and RoS is RoShuNet.

resolution aids in improving the final accuracy. Additionally, within a certain range, boosting the resolution can increase the accuracy of the model, but outside this range, it can be observed from each strip that the accuracy increase rate gradually slows.

Similarly, the experimental results regarding the correlation between the model complexity and resolution are presented as a bar chart in Fig. 8. From the figure, we note that the model complexity represented by KFLOPs increases with increasing resolution; moreover, our method can always exhibit lower model complexity as the resolution increases.

## 5. Discussion

Table 2 shows that (i) the 1.0X and 1.5X versions of RoShuNet outperform ShuffleNetV2 on all four datasets. This means that the 1.0X and 1.5X versions of RoShuNet are better than ShuffleNetV2 in terms of addressing not only limited classes, multiple classes, and even multiple large classes but also small objects at low resolutions and large objects at high resolutions. The 2.0X version of RoShuNet demonstrates better classification performance on Caltech-256 and miniImageNet than does than ShuffleNetV2_2.0X in large multiclass classification scenarios with large objects at high resolutions; (ii) experiments conducted on the four datasets demonstrate the competitive performance of RoShuNet not only in terms of accuracy but also in model size and model complexity compared to the SOTA methods. That is, the proposed RoShuNet is competent in feature extraction tasks while being memory- and computationally-efficient, and (iii) experiments conducted on the Caltech-256 dataset demonstrate the significantly better performance achieved by the MobileNet series when addressing large multiclass classification scenarios involving large objects at high resolutions.

Table 3 further shows that (i) when introducing A-DGC to ShuffleNetV2, both the accuracy and the model size improve. The reason for this lies in the fact that A-DGC makes full use of group convolution to substantially replace the use of dense convolution in ShuffleNetV2; therefore, the consequent model size is compressed. Additionally, as group convolution can hinder channel interflowing, we further elicit an aggregation stream by aggregating the information derived from all of the channels of the current feature maps to boost the channel interflowing process, and the results prove the effectiveness of this strategy. (ii) When introducing the $\gamma$-WSM to ShuffleNetV2_A, the accuracy continues to increase and reaches a peak. This reveals that when facilitating channel interflow, the feature information existing in the given feature maps should not be neglected. (iii) When incorporating slimming strategies into RoShuNet, the model complexity and model size decrease, but the accuracy remains strong and is even higher than that of ShuffleNetV2. This indicates that well-trained models usually tend to possess redundant feature extraction layers, and pruning these layers can reduce the model size and lower the model complexity while retaining accuracy.

From the results of the generalization experiments, (1) as listed in Table 5, we note that DeepLabV3 backboned with RS outperforms DeepLabV3 (ResNet50) in terms of the MIoU, GFLOPs and MParams metrics (yielding an accuracy increase of 6.23%, while the GFLOPs and MParams account for just ~28% and ~52% of those of ResNet50, respectively); and (2) as listed in Table 6, we likewise note that the DAN backboned with RS yields the best SMOTA performance and achieves comparable performance in terms of the remaining metrics while producing a much less complex model with less memory overhead. These two groups of generalization experiments demonstrate both the competence and practicality of RoShuNet with respect to addressing semantic

segmentation and multiple-object tracking scenarios, which implies its powerful feature extraction ability for downstream tasks.

We also visualize the feature maps obtained after implementing the A-DGC module on Kaggle-CatsvsDogs, as shown in Fig. 6. We find that when addressing Kaggle-CatsvsDogs, RoShuNet exhibits more powerful object location and feature extraction capabilities, but these good traits disappear when employing CIFAR-100 and Caltech-256. We speculate that this phenomenon results from the number of present classes. When the number of classes is limited, RoShuNet can benefit from extracting the features of not only the spatial dimension but also the planar dimension of A-DGC, yielding better object location and feature extraction performance. When the number of classes is large, RoShuNet mainly focuses on how to correctly classify the objects because of its light weight; therefore, the accuracy at this point is higher than that of ShuffleNetV2.

From Table 7, we note that when sorting the importance vectors in ascending order or utilizing any other implementations containing ascending sorting, the resulting accuracy decreases. When the descending sorting method is utilized, the accuracy is exactly the opposite (i.e., it increases). This can be traced back to the structure shown in Fig. 1; the input of each ASB module is split into two parts, namely, $h_f$ and $h_b$. When performing shuffling according to the importance vectors in descending order, $h_f$ is fed with feature maps that possess high importance levels. In this case, after extracting the corresponding features through the $h_f$ branch, the model can better formalize the traits of the input. Furthermore, the remaining feature maps fed into $h_b$ are less important, and no more operations actually need to be carried out; the maps are simply concatenated at the back end of the ASB to guarantee the integrity of the features and prevent the degradation problem when training. However, when using the importance vectors in ascending order to shuffle the feature maps, the features passed to the $h_f$ branch are less important, and the model at this point does not further gain substantially important (discriminative) features from the input, which is why the implementations containing the importance vectors in ascending order yield inferior accuracy.

From Fig. 7, we note that increasing the resolution allows for better accuracy to be attained within a certain range, but continuing to increase the resolution has very little effect after reaching a certain point. As shown in Fig. 7, the rate of accuracy change slows after the resolution reaches 224 ($32 \times 7$). Unsurprisingly, the correlation between the model complexity and resolution is positive, as shown in Fig. 8. Therefore, it is very important to investigate the best resolution that best suits the model after constructing a new network.

## 6. Conclusion

In this article, we present an accurate and lightweight CNN-based feature extractor named RoShuNet for visual images by integrating A-DGC and a $\gamma$-WSM into a unified pipeline. To satisfy some specific situations, we also introduce two slimming strategies and apply them to RoShuNet. Extensive experiments conducted not only on classification but also on semantic segmentation and multiple-object tracking tasks demonstrate the competitive performance of our method and verify that it is more memory-efficient and computationally efficient than the competing approaches. We hope that our method could offer a novel perspective for the design of subsequent lightweight models.

The proposed A-DGC module, which possesses two sub-versions (A-DGC(l) and A-DGC(s)), requires less memory overhead while maintaining its feature extraction ability by extracting features from both the spatial and planar dimensions; thus, we hope that this module will provide a new option for constructing future lightweight networks in lieu of the current widely used DSC. The proposed $\gamma$-WSM further facilitates channelwise interflowing by taking channel importance into consideration, which implies that feature importance should not be neglected when constructing networks and that highly important features should be given priority and optimally utilized.

In the future, we will further consider simplifying the model structure to improve its parallelism and accelerate its inference time (RoShuNet currently achieves real-time performance of ~140 FPS under an Intel i5 CPU and ~4800 FPS under an Nvidia RTX3060 GPU with a resolution of 160 in image classification tasks). For example, the simple structure used in RepVGG can be adopted to further simplify the proposed implementation and accelerate its inference time.

## CRediT authorship contribution statement

**Xujie He:** Conceptualization, Investigation, Writing – original draft. **Jing Jin:** Funding acquisition, Writing – review & editing. **Yu Jiang:** Data curation, Validation. **Dandan Li:** Project administration.

## 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.

## Data availability

Data will be made available on request.

## Acknowledgments

This work was supported by of the National Natural Science Foundation of China [grant numbers 11973021, 12373107]; and the National Key R&D Program of China [grant number 2021ZD0110900].

## References

Bernardin, K., Stiefelhagen, R., 2008. Evaluating multiple object tracking performance: The CLEAR MOT metrics. Eurasip. J. Image Video Process. http://dx.doi.org/10.1155/2008/246309.

Bonnaerens, M., Freiberger, M., Dambre, J., 2022. Anchor pruning for object detection. Comput. Vis. Image Underst. 221, 103445. http://dx.doi.org/10.1016/j.cviu.2022.103445.

Chang, S.E., Li, Y., Sun, M., Shi, R., So, H.K.H., Qian, X., Wang, Y., Lin, X., 2021. Mix and match: A novel FPGA-centric deep neural network quantization framework. In: Proc. - Int. Symp. High-Performance Comput. Archit. 2021-Febru. pp. 208–220. http://dx.doi.org/10.1109/HPCA51647.2021.00027.

Chen, L.-C., Papandreou, G., Schroff, F., Adam, H., 2017. Rethinking atrous convolution for semantic image segmentation.

Chollet, F., 2017. Xception: Deep learning with depthwise separable convolutions. In: Proc. - 30th IEEE Conf. Comput. Vis. Pattern Recognition, CVPR 2017 2017-Janua. pp. 1800–1807. http://dx.doi.org/10.1109/CVPR.2017.195%2019.

Ding, X., Zhang, X., Han, J., Ding, G., 2022. Scaling up your kernels to 31 ×31: Revisiting large kernel design in CNNs. In: Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. 2022-June. pp. 11953–11965. http://dx.doi.org/10.1109/CVPR52688.2022.01166.

Freeman, I., Roese-Koerner, L., Kummert, A., 2018. Effnet: An efficient structure for convolutional neural networks. In: Proc. - Int. Conf. Image Process. ICIP 6–10. http://dx.doi.org/10.1109/ICIP.2018.8451339.

Goodfellow, I., Warde-Farley, D., Mirza, M., Courville, A., Bengio, Y., 2013. Maxout networks. In: Dasgupta, S., McAllester, D. (Eds.), Proc. 30th Int. Conf. Mach. Learn.. PMLR, Atlanta, Georgia, USA, pp. 1319–1327.

Griffin, G., Alex, H., Pietro, P., 2007. Caltech-256 object category dataset.

Han, J.G., Park, T.H., Moon, Y.H., Eom, I.K., 2016. Efficient Markov feature extraction method for image splicing detection using maximization and threshold expansion. J. Electron. Imaging 25, 023031. http://dx.doi.org/10.1117/1.jei.25.2.023031.

Han, K., Wang, Y., Tian, Q., Guo, J., Xu, Chunjing, Xu, Chang, 2020. GhostNet: More features from cheap operations. In: Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. pp. 1577–1586. http://dx.doi.org/10.1109/CVPR42600.2020.00165.

Han, K., Wang, Y., Xu, Chang, Guo, J., Xu, Chunjing, Wu, E., Tian, Q., 2022. GhostNets on heterogeneous devices via cheap operations. Int. J. Comput. Vis. 130, 1050–1069. http://dx.doi.org/10.1007/s11263-022-01575-y.

He, K., Zhang, X., Ren, S., Sun, J., 2016a. Deep residual learning for image recognition. In: Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. 2016-Decem. pp. 770–778. http://dx.doi.org/10.1109/CVPR.2016.90.

He, K., Zhang, X., Ren, S., Sun, J., 2016b. Identity mappings in deep residual networks. In: Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics) 9908 LNCS, pp. 630–645. http://dx.doi.org/10.1007/978-3-319-46493-0_38.

Hjouji, A., 2022. Orthogonal invariant Lagrange-Fourier moments for image recognition. Expert Syst. Appl. 199, 117126. http://dx.doi.org/10.1016/j.eswa.2022.117126.

Howard, A., Sandler, M., Chen, B., Wang, W., Chen, L.C., Tan, M., Chu, G., Vasudevan, V., Zhu, Y., Pang, R., Le, Q., Adam, H., 2019. Searching for mobileNetV3. In: Proc. IEEE Int. Conf. Comput. Vis. 2019-Octob. pp. 1314–1324. http://dx.doi.org/10.1109/ICCV.2019.00140.

Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H., 2017. MobileNets: Efficient convolutional neural networks for mobile vision applications.

Hu, J., Shen, L., Albanie, S., Sun, G., Wu, E., 2020. Squeeze-and-excitation networks. IEEE Trans. Pattern Anal. Mach. Intell. 42, 2011–2023. http://dx.doi.org/10.1109/TPAMI.2019.2913372.

Huo, X., Zeng, X., Wu, S., Shen, W., Wong, H.S., 2023. Collaborative learning with unreliability adaptation for semi-supervised image classification. Pattern Recognit. 133, http://dx.doi.org/10.1016/j.patcog.2022.109032.

Ke, X., Huang, Y., Guo, W.Z., 2022. Weakly supervised fine-grained image classification via two-level attention activation model. Comput. Vis. Image Underst. 218, 103408. http://dx.doi.org/10.1016/j.cviu.2022.103408.

Krizhevsky, A., Sutskever, I., Hinton, G.E., 2012. ImageNet classification with deep convolutional neural networks. In: Pereira, F., Burges, C.J., Bottou, L., Weinberger, K.Q. (Eds.), Advances in Neural Information Processing Systems. Curran Associates, Inc.

Liu, C.T., Chen, J.C., Chen, C.S., Chien, S.Y., 2021. Video-based person re-identification without bells and whistles. In: IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. Work. pp. 1491–1500. http://dx.doi.org/10.1109/CVPRW53098.2021.00165.

Liu, Z., Li, J., Shen, Z., Huang, G., Yan, S., Zhang, C., 2017. Learning efficient convolutional networks through network slimming. In: Proc. IEEE Int. Conf. Comput. Vis. 2017-Octob. pp. 2755–2763. http://dx.doi.org/10.1109/ICCV.2017.298.

Liu, Y., Zhang, W., Wang, J., 2020. Adaptive multi-teacher multi-level knowledge distillation. Neurocomputing 415, 106–113. http://dx.doi.org/10.1016/j.neucom.2020.07.048.

Ma, N., Zhang, X., Zheng, H.T., Sun, J., 2018. Shufflenet V2: Practical guidelines for efficient cnn architecture design. In: Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics) 11218 LNCS, pp. 122–138. http://dx.doi.org/10.1007/978-3-030-01264-9_8.

Qu, S., Tan, H., Li, Q., Peng, Z., 2022. Interactive image segmentation based on the appearance model and orientation energy. Comput. Vis. Image Underst. 217, 103371. http://dx.doi.org/10.1016/j.cviu.2022.103371.

Ren, S., He, K., Girshick, R., Sun, J., 2017. Faster R-CNN: Towards real-time object detection with region proposal networks. IEEE Trans. Pattern Anal. Mach. Intell. 39, 1137–1149. http://dx.doi.org/10.1109/TPAMI.2016.2577031.

Ristani, E., Solera, F., Zou, R., Cucchiara, R., Tomasi, C., 2016. In: Hua, G., Jégou, H. (Eds.), Performance Measures and a Data Set for Multi-Target, Multi-Camera Tracking BT - Computer Vision – ECCV 2016 Workshops. Springer International Publishing, Cham, pp. 17–35.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.C., 2018. MobileNetV2: Inverted residuals and linear bottlenecks. In: Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. pp. 4510–4520. http://dx.doi.org/10.1109/CVPR.2018.00474.

Simonyan, K., Zisserman, A., 2015. Very deep convolutional networks for large-scale image recognition. In: 3rd Int. Conf. Learn. Represent. ICLR 2015 - Conf. Track Proc. pp. 1–14.

Sun, S., Akhtar, N., Song, H., Mian, A., Shah, M., 2021. Deep affinity network for multiple object tracking. IEEE Trans. Pattern Anal. Mach. Intell. 43, 104–119. http://dx.doi.org/10.1109/TPAMI.2019.2929520.

Szegedy, C., Ioffe, S., Vanhoucke, V., Alemi, A.A., 2017. Inception-v4, inception-ResNet and the impact of residual connections on learning. In: 31st AAAI Conf. Artif. Intell. AAAI. pp. 4278–4284. http://dx.doi.org/10.1609/aaai.v31i1.11231.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z., 2016. Rethinking the inception architecture for computer vision. In: Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. 2016-Decem. pp. 2818–2826. http://dx.doi.org/10.1109/CVPR.2016.308.

Tang, Y., Liu, Y., Huang, H., 2022. Target-aware and spatial–spectral discriminant feature joint correlation filters for hyperspectral video object tracking. Comput. Vis. Image Underst. 223, 103535. http://dx.doi.org/10.1016/j.cviu.2022.103535.

Vinyals, O., Blundell, C., Lillicrap, T., Kavukcuoglu, K., Wierstra, D., 2016. Matching networks for one shot learning. Adv. Neural Inf. Process. Syst. 3637–3645.

Wang, Y., Sun, S., 2022. A rock fabric classification method based on the grey level co-occurrence matrix and the Gaussian mixture model. J. Nat. Gas Sci. Eng. 104, 104627. http://dx.doi.org/10.1016/j.jngse.2022.104627.

Wang, L., Yoon, K.J., 2022. Knowledge distillation and student-teacher learning for visual intelligence: A review and new outlooks. IEEE Trans. Pattern Anal. Mach. Intell. 44, 3048–3068. http://dx.doi.org/10.1109/TPAMI.2021.3055564.

Zeiler, M.D., Fergus, R., 2014. Visualizing and understanding convolutional networks. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (Eds.), Computer Vision – ECCV 2014. Springer International Publishing, Cham, pp. 818–833.

Zhang, W., He, X., Li, W., Zhang, Z., Luo, Y., Su, L., Wang, P., 2021. A robust deep affinity network for multiple ship tracking. IEEE Trans. Instrum. Meas. 70, http://dx.doi.org/10.1109/TIM.2021.3077679.

Zhang, X., Zhou, X., Lin, M., Sun, J., 2018. ShuffleNet: An extremely efficient convolutional neural network for mobile devices. In: Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. pp. 6848–6856. http://dx.doi.org/10.1109/CVPR.2018.00716.

**Xujie He** received an M.Eng. degree from Harbin Engineering University, Harbin, China, in 2021, and a B.Eng. degree from Shenyang Aerospace University, Shenyang, China, in 2018. He is currently pursuing a Ph.D. degree in Control Science and Engineering at Harbin Institute of Technology, China. His research interests primarily focus on computer vision, including defect segmentation and detection, multiple object tracking, and visual foundation models.

**Jing Jin** received her Ph.D. from Harbin Institute of Technology, Harbin, China, in 2008. She is currently a professor at the School of Astronautics, Harbin Institute of Technology, China. She has received research funding from the Chinese Natural Science Foundation and has been involved in several intelligent astronautics projects. Her research interests include computer vision, specifically intelligent detection and diagnosis for medical images; and navigation and localization technologies, including spacecraft pulsar navigation, SLAM, and smart curling robots.

**Yu Jiang** received his Ph.D. and bachelor's degrees in 2010 and 2003, respectively, from the Department of Control Science and Engineering at Harbin Institute of Technology, China. Since 2020, he has been a Senior Engineer in the Department of Control Science and Engineering. His current research interests include nonlinear control, underactuated control, artificial intelligence, and their applications in the guidance and control of flight vehicles.

**Dandan Li** obtained Ph.D. from Harbin Institute of Technology and then stayed on to work at the university. She currently also serves as the technical leader of the HIT-Nvidia AI and Control Joint Lab. She has extensive project experience in medical image processing and intelligent diagnosis, with her current research focusing on artificial intelligence applications and algorithm development. She has published numerous scientific papers in international journals, holds several patents, and, as a key member, has won the first prize in provincial science and technology progress.