SCConv: Spatial and Channel Reconstruction Convolution for Feature Redundancy

Jiafeng Li[1], Ying Wen[1]*, Lianghua He[2]
[1]School of Communication and Electronic Engineering, East China Normal University, Shanghai, China.
[2]Department of Computer Science and Technology, Tongji University, Shanghai, China.
51205904113@stu.ecnu.edu.cn;ywen@cs.ecnu.edu.cn;helianghua@tongji.edu.cn

Abstract

Convolutional Neural Networks (CNNs) have achieved remarkable performance in various computer vision tasks but this comes at the cost of tremendous computational resources, partly due to convolutional layers extracting redundant features. Recent works either compress well-trained large-scale models or explore well-designed lightweight models. In this paper, we make an attempt to exploit spatial and channel redundancy among features for CNN compression and propose an efficient convolution module, called SCConv (Spatial and Channel reconstruction Convolution), to decrease redundant computing and facilitate representative feature learning. The proposed SCConv consists of two units: spatial reconstruction unit (SRU) and channel reconstruction unit (CRU). SRU utilizes a separate-and-reconstruct method to suppress the spatial redundancy while CRU uses a split-transform-and-fuse strategy to diminish the channel redundancy. In addition, SCConv is a plug-and-play architectural unit that can be used to replace standard convolution in various convolutional neural networks directly. Experimental results show that SCConv-embedded models are able to achieve better performance by reducing redundant features with significantly lower complexity and computational costs.

1. Introduction

In recent years, convolutional neural networks (CNNs) have obtained widespread applications in computer vision tasks [24] due to its ability in obtaining representative features. However, such success relies heavily on intensive resources of computation and storage, which poses severe challenges to their efficient deployment on resource-constrained environments. Therefore, to address these challenges, various types of model compression strategies and network designs have been explored to improve network efficiency [1, 2, 26]. The former includes network pruning, weight quantization, low-rank factorization, and knowledge distillation. To be specific, network pruning [17, 22, 30] is a straightforward way to prune the uncritical neuron connections from an existing learned big model to make it thinner. Weight quantization [9] mainly focuses on converting network weights from floating-point types to integer ones to save computation resources. Low-rank factorization [5] applies the matrix decomposition techniques to estimate the informative parameters. Knowledge distillation [11, 34] generates small student networks with the guidance of a well-trained big teacher network. The common part of these compression techniques is that they have been regarded as post-processing steps, thus their performance is usually upper bounded by the given initial model. Meanwhile, the accuracy of these methods drastically drops while achieving a high compression rate.

Network design is another alternative way, which aims at reducing the inherent redundancy in dense model parameters and further developing a lightweight network model. For example, ResNet [10] and DenseNet [14] utilize an efficient shortcut connection to improve the network topology, which connects all preceding feature maps to diminish the redundant parameters. ResNeXt [31] replaces traditional convolutions with sparsely connected group convolutions to reduce inter-channel connectivity. Networks like Xception [4], MobileNet [12] and MobileNeXt [35] disentangle standard convolution into depth-wise convolution and point-wise convolution to further decrease the connection density between channels. MicroNet [19] adopts micro-factorized convolution to handle extremely low FLOPs by integrating sparse connectivity into convolution. In addition, EfficientNet [27] learns to automatically search optimal network architectures to lower the redundancy in dense model parameters.

Moreover, in CNN architecture design, bottleneck structure has been well adopted, in which 3×3 convolutional layers account for a majority of the model parameters and FLOPs. Therefore various efficient convolutional opera-

*Corresponding author (e-mail: ywen@cs.ecnu.edu.cn)

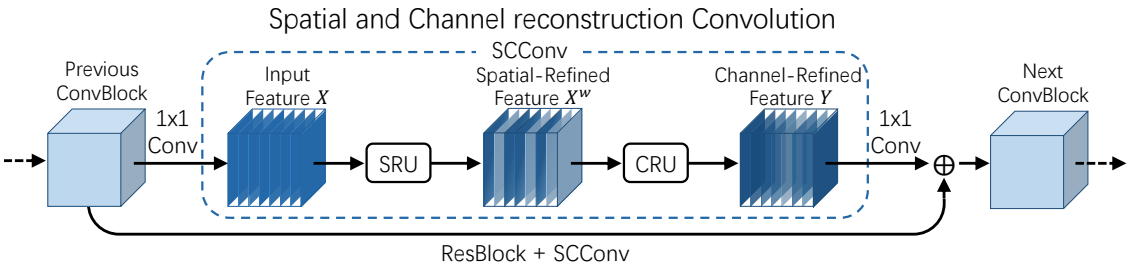

Figure 1. The architecture of SCConv integrated with Spatial Reconstruction Unit (SRU) and Channel Reconstruction Unit (CRU). This figure shows the exact position of our SCConv module within a ResBlock.

tions, such as group-wise convolution (GWC), depth-wise convolution (DWC) and point-wise convolution (PWC), a variant of the standard convolution, are proposed to replace the existing expensive convolutional operation. GWC, which was first introduced in AlexNet [16], can be regarded as a sparse convolution connection method that each output channel is connected to only a certain group of input channels. DWC [13] proposes to bring more efficiency by keeping each channel separately convolved with filters and without interaction between channels. PWC is used to keep the information flowing across channels and enable dimensionality reduction by reducing the number of filters. These operations are similar in sparse connectivity and have benefits in the number of parameters and FLOPs, which illustrates that redundancy in channel dimension can be reduced reasonably. Hence, a variety of convolution operations are proposed to explore redundancy reduction. For example, MobileNet [12] introduces inverted residual blocks using DWC and PWC to filter the features, which decreases the number of parameters while accelerating the training. ShuffleNet [33] resorts to point-wise group convolution and channel shuffle operation to improve the information flow between different channel groups. HetConv [25] designs heterogeneous convolutional filters where a 3×3 convolution kernels and a 1×1 convolution kernels are included in one single filter to extract features. TiedBlockConv [28] shares the same convolutional filter over equal blocks of channels to produce multiple responses within a single filter. SPConv [32] divides the input channels into two groups for different processing but needs a relatively large amount of calculation while extracting internal information. GhostNet [8] considers the redundancy between feature maps and uses cheap operations like DWC to learn redundant features. SilmConv [23] adopts the operation of reducing the channels of features and flipping the weights to reduce feature redundancy. In addition, orthogonal to channel redundancy, OctConv [3] proposes octave convolution to separate convolutional filters into high-frequency and low-frequency components, processing the latter in low resolution to alleviate the spatial redundancy, which reduces calculations while keeping the same number of parameters.

All these prior studies have proven that there indeed exists considerable redundancy in the deep neural networks, not only in dense model parameters but also in the spatial and channel dimension of feature maps. However, all the above methods either focus on reducing the redundancy in channel dimension or in spatial dimension, making the network still suffer from the problems of feature redundancy.

In this paper, different from prior work, we design a two-step procedure to exploit the redundancy of intermediate feature maps, with the goal of reducing the number of parameters and computation without performance loss. To this end, we propose a novel CNN compression approach to jointly reduce spatial and channel redundancy in the convolutional layers, termed as SCConv (Spatial and Channel reconstruction Convolution), which consists of two units, spatial reconstruction unit (SRU) and channel reconstruction unit (CRU). The proposed SCConv module, which can be embedded into various architectures without additional modifications, is designed to efficiently restrict feature redundancy. This module not only cuts down on the number of model parameters and FLOPs but enhances the capability of feature representation. We summarize our contributions as follows:

- We propose a spatial reconstruction unit, termed as SRU, which separates redundant features based on weights and reconstructs them to suppress the redundancy in spatial dimension and strengthen the representation of features.
- We propose a channel reconstruction unit, termed as CRU, which utilizes a split-transform-and-fuse strategy to diminish the redundancy in channel dimension as well as the computational costs and storage.
- We design a plug-and-play operation named SCConv combining SRU and CRU in a sequential manner to replace standard convolution for operating on a variety of backbone CNNs. It turns out that SCConv can substantially save the computing load yet enhance the model performance on challenging tasks.

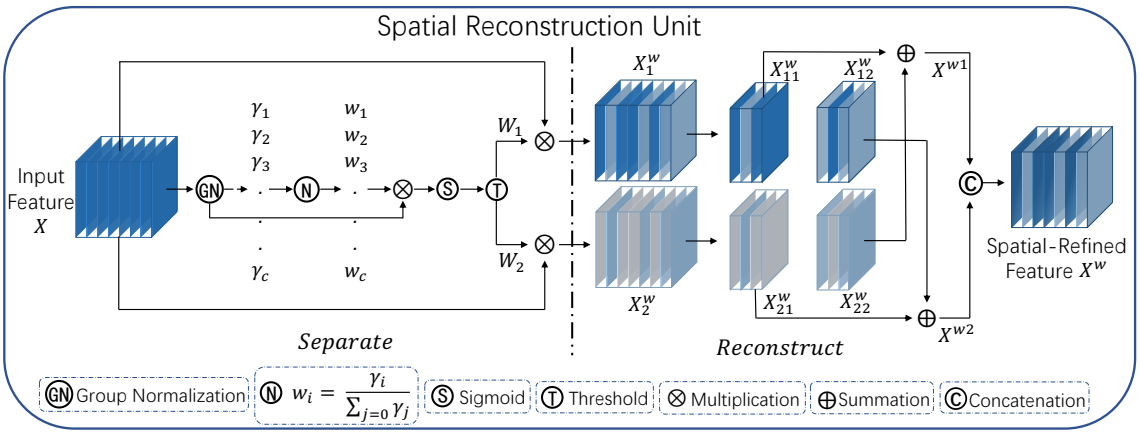

Figure 2. The architecture of Spatial Reconstruction Unit.

2. Methodology

In this section, we introduce the proposed SCConv as illustrated in Figure 1, which consists of two units, Spatial Reconstruction Unit (SRU) and Channel Reconstruction Unit (CRU), placed in a sequential manner. Concretely, for the intermediate input features X in the bottleneck residual block, we first obtain the spatial-refined features X^w through SRU operation and subsequently we utilize CRU operation to gain the channel-refined features Y. We exploit both spatial and channel redundancy among features in our SCConv module, which can be seamlessly integrated into any CNN architecture to decrease redundancy among intermediate feature maps and boost the feature representation of CNNs.

2.1. SRU for Spatial Redundancy

To exploit the spatial redundancy of features, we introduce Spatial Reconstruction Unit (SRU) as shown in Figure 2, which utilizes a *Separate*-and-*Reconstruct* operation. *Separate* operation aims to separate those informative feature maps from less informative ones corresponding to the spatial content. We leverage the scaling factors in Group Normalization (GN) [29] layers to assess the informative content of different feature maps. To be concrete, given an intermediate feature map $X \in \mathbb{R}^{N\times C\times H\times W}$, where N is the batch axis, C is the channel axis, H and W are the spatial height and width axes. We first standardize the input feature X by subtracting mean μ and dividing by standard deviation σ as follows:

$$X_{out} = GN(X) = \gamma\frac{X-\mu}{\sqrt{\sigma^2+\varepsilon}} + \beta \tag{1}$$

where μ and σ are the mean and standard deviation in X, ε is a small positive constant added for the sake of division stability, γ and β are trainable affine transformation.

Noted that we leverage the trainable parameters $\gamma \in R^C$ in GN layers as a way to measure the variance of spatial pixels for each batch and channel. The richer spatial information reflects more variation in spatial pixels contributing to a larger γ. The normalized correlation weights $W_\gamma \in R^C$ are obtained by equation 2, which indicates the importance of different feature maps.

$$W_\gamma = \{w_i\} = \frac{\gamma_i}{\sum_{j=1}^{C}\gamma_j},\ i,j = 1,2,\cdots,C \tag{2}$$

Then the weight values of feature maps reweighted by W_γ are mapped to the range $(0,1)$ by the sigmoid function and gated by a threshold. We set those weights above the threshold to 1 to obtain the informative weights W_1 while setting them to 0 to gain the non-informative weights W_2 (the threshold is set to 0.5 in the experiments). The whole process of acquiring W can be expressed as equation 3:

$$W = Gate(Sigmoid(W_\gamma(GN(X)))) \tag{3}$$

Finally, we multiply input features X by W_1 and W_2 respectively, yielding two weighted features: the informative ones X_1^w and less informative ones X_2^w. Thus we successfully separate the input features into two parts: X_1^w has informative and expressive spatial contents while X_2^w has little or no information, which is regarded as redundant.

In order to reduce the spatial redundancy, we further propose a *Reconstruct* operation that features with rich information sum up with less informative ones to generate features with richer information and save spatial space. Instead of adding these two parts directly, we adopt a cross reconstruct operation to sufficiently combine the weighted two different informative features and strengthen the information flow between them. Afterwards we concatenate the cross-reconstructed features X^{w1} and X^{w2} to obtain the spatial-refined feature maps X^w. The whole process of *Re-*

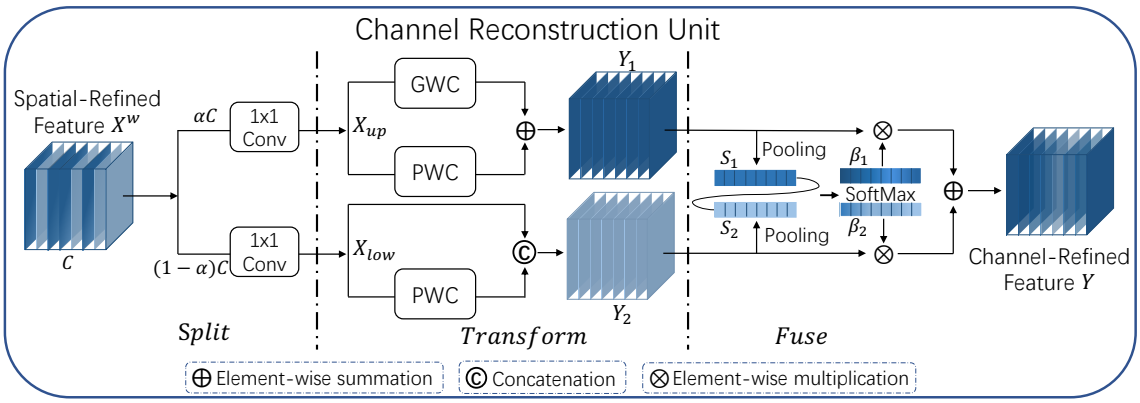

Figure 3. The architecture of Channel Reconstruction Unit.

construct operation can be expressed as :

$$\begin{cases} X_1^w = W_1 \otimes X, \\ X_2^w = W_2 \otimes X, \\ X_{11}^w \oplus X_{22}^w = X^{w1}, \\ X_{21}^w \oplus X_{12}^w = X^{w2}, \\ X^{w1} \cup X^{w2} = X^w. \end{cases} \tag{4}$$

where $\otimes$ is element-wise multiplication, $\oplus$ is element-wise summation, $\cup$ is concatenation. After the SRU is applied to the intermediate input features X, not only do we separate the informative features from less informative ones, but also we reconstruct them to enhance the representative features and suppress the redundant features in spatial dimension. Nevertheless, the spatial-refined feature maps X^w still remain redundant in channel dimension.

2.2. CRU for Channel Redundancy

To exploit the channel redundancy of features, we introduce Channel Reconstruction Unit (CRU) as shown in Figure 3, which utilizes a *Split-Transform*-and-*Fuse* strategy. Normally, we use repetitive standard $k \times k$ convolutions to extract features, resulting in some relatively redundant feature maps along the channel dimension. Let $M^k \in \mathbb{R}^{c\times\ k\times\ k}$ denotes a $k \times k$ convolution kernel and $X, Y \in \mathbb{R}^{c\times\ h\times\ w}$ denotes the input and convolved output features respectively. A standard convolution [1] can be defined as $Y = M^k X$. To be specific, we replace the standard convolution with CRU, which is implemented via three operators – *Split*, *Transform* and *Fuse*.

Split : For given spatial-refined features $X^w \in \mathbb{R}^{c\times h\times w}$, we first split the channels of X^w into two parts with αC channels and $(1-\alpha)C$ channels respectively, as shown in the *split* part of Figure 3, where $0 \le \alpha \le 1$ is a split ratio. Subsequently, we further utilize 1×1 convolutions to squeeze the channels of feature maps for its computing efficiency. Here we introduce a squeeze ratio r to control the feature channels to balance the computational cost of the CRU ($r = 2$ is a typical setting in the experiments). After the split and squeeze operations, we divide the spatial-refined features X^w into the upper part X_{up} and the lower part X_{low}.

Transform : X_{up} is fed into the upper transformation stage, serving as a "Rich Feature Extractor". We adopt efficient convolutional operations ($i.e.$ GWC and PWC) to replace the expensive standard $k \times k$ convolutions to extract high-level representative information as well as reduce the computational cost. Owing to sparse convolution connections, GWC reduces the amount of parameters and calculations but cuts off the information flow between channel groups. While PWC compensates for the information loss and helps the information flow across feature channels. Thus we perform $k \times k$ GWC (we set group size $g = 2$ in the experiments) and 1×1 PWC operations on the same X_{up}. Afterward, we sum up the output to form a merged representative feature maps Y_1 as shown in the *Transform* part of Figure 3. The upper transformation stage can be formulated as :

$$Y_1 = M^G X_{up} + M^{P_1} X_{up} \tag{5}$$

where $M^G \in \mathbb{R}^{\frac{\alpha c}{gr}\times\ k\times\ k\times c}$, $M^{P_1} \in \mathbb{R}^{\frac{\alpha c}{r}\times\ 1\times\ 1\times c}$ is a learnable weight matrix of GWC and PWC, $X_{up} \in \mathbb{R}^{\frac{\alpha c}{r}\times\ h\times\ w}$ and $Y_1 \in \mathbb{R}^{c\times\ h\times\ w}$ are the upper input and output feature maps respectively. In short, the upper transformation stage leverages a combination of GWC and PWC on the same feature maps X_{up} to extract rich representative features Y_1 with less computational cost.

X_{low} is fed into the lower transformation stage, where we apply cheap 1×1 PWC operations to generate feature maps with shallow hidden details as a supplementary to the Rich Feature Extractor. In addition, we reuse features X_{low} to obtain more feature maps without extra cost. Lastly, we

[1] For simplicity, we omitt the bias term.

concatenate the generated and reused features to form the output of the lower stage Y_2 as follows:

$$Y_2 = M^{P_2} \ X_{low} \cup X_{low} \tag{6}$$

where $M^{P_2} \in \mathbb{R}^{\frac{(1-\alpha)c}{r} \times 1 \times 1 \times (1-\frac{1-\alpha}{r})c}$ is a learnable weight matrix of PWC, $\cup$ is concatenation operation, $X_{low} \in \mathbb{R}^{\frac{(1-\alpha)c}{r} \times h \times w}$ and $Y_2 \in \mathbb{R}^{c \times h \times w}$ are the lower input and output feature maps respectively. In a word, the lower transformation stage reuses preceding features X_{low} and utilizes cheap 1×1 PWC to obtain features Y_2 with supplementary detailed information.

Fuse : After the transformation is performed, instead of direct concatenating or adding two types of features, we utilize the simplified SKNet method [18] to adaptively merge the output features Y_1 and Y_2 from upper and lower transformation stage as shown in the *Fuse* part of Figure 3. We first apply a global average pooling (Pooling) to collect global spatial information with channel-wise statistics $S_m \in \mathbb{R}^{c \times 1 \times 1}$, which is calculated as:

$$S_m = Pooling(Y_m) = \frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} Y_c(i,j), \ m = 1, 2 \tag{7}$$

Next, we stack the upper and lower global channel-wise descriptor S_1, S_2 together and use channel-wise soft attention operation to generate feature importance vector $\beta_1, \beta_2 \in \mathbb{R}^c$ as follows:

$$\beta_1 = \frac{e^{s_1}}{e^{s_1} + e^{s_2}}, \ \beta_2 = \frac{e^{s_2}}{e^{s_1} + e^{s_2}}, \beta_1 + \beta_2 = 1 \tag{8}$$

Finally, under the guidance of feature importance vector β_1, β_2, the channel-refined features Y can be obtained by merging the upper features Y_1 and the lower features Y_2 in a channel-wise manner as follows:

$$Y = \ \beta_1 Y_1 + \beta_2 Y_2 \tag{9}$$

In brief, we adopt CRU, using *Split-Transform*-and-*Fuse* strategy, to further diminish the redundancy of spatial-refined feature maps X^w along the channel dimension. Furthermore, CRU extracts rich representative features through lightweight convolutional operations while proceeds redundant features with cheap operation and feature reuse schemes. Overall, CRU can be used individually or in conjunction with SRU operation. By arranging the SRU and CRU in a sequential manner, the proposed SCConv is established, which is highly efficient and capable of replacing standard convolution operations.

2.3. Analysis on Complexities

Our SCConv is designed as a plug-and-play module that can be easily embedded into various existing well-designed neural architectures to reduce computation and storage costs. In the SCConv module, all of the parameters are concentrated on the transformation stage. Hence we analyze the reduction of theoretical memory usage. The parameters of standard convolution $Y = M^k X$ can be calculated as:

$$P_s = k \ \times \ k \times \ C_1 \times \ C_2 = k^2 C_1 C_2 \tag{10}$$

where k is kernel size of the convolution, C_1 and C_2 are the number of input and output feature channels.

The parameters of the proposed SCConv module consist of :

$$\begin{aligned} P_{sc} =& 1 \times 1 \times \alpha C_1 \times \frac{\alpha C_1}{r} + k \times k \times \frac{\alpha C_1}{gr} \times \frac{C_2}{g} \times \ g \\ &+ 1 \times 1 \times \frac{\alpha C_1}{r} \times C_2 + (1-\alpha) C_1 \times \frac{(1-\alpha) C_1}{r} \\ &+ 1 \times 1 \times \frac{(1-\alpha) C_1}{r} \times \left(C_2 - \frac{1-\alpha}{r} C_1 \right) \end{aligned} \tag{11}$$

where α denotes the split ratio, r refers to the squeeze ratio, g is the group size of GWC operation, C_1 and C_2 are input and output feature channels respectively. Here, we give a comparison to show the performance of the proposed SCConv. In the experiment, the general parameter set is $\alpha = \frac{1}{2}, \ r = 2, \ g = 2, \ k = 3, \ C_1 = C_2 = C$, the amount of parameters can be reduced by 5 times where $P_s / P_{sc} \approx 5$ while the model achieves better performance than standard convolution.

3. Experiments

To evaluate the effectiveness of the proposed SCConv, in this section, we perform a series of experiments on image classification and object detection with only the widely used 3×3 kernels being replaced by SCConv module. Image classification benchmarks includes CIFAR-10 [15], CIFAR-100 [15] and ImageNet-1K [16]. Object detection benchmarks include PASCAL VOC [6] and MS COCO [21]. Top-1 accuracy is reported as the evaluation metric for image classification and the mean average precision (mAP) is used to measure accuracy of object detection. For fair comparisons, all models in each experiment, including re-implemented baselines and SCConv-equipped models, are trained from scratch on 2 NVIDIA Tesla V100 GPUs with the default data augmentation and training strategy and no other tricks are used. In each experiment, we train several times with the same configuration to prevent the impact of fluctuations and report the median of results.

3.1. Experimental Settings

Dataset. CIFAR dataset, including CIFAR-10 and CIFAR-100, consists of 50k training images and 10k validation images, which are divided into 10 and 100 classes

respectively. ImageNet-1K dataset is a large-scale image classification dataset, containing 1.28 million training images and 50k validation images from 1k classes. PASCAL VOC dataset, which has 20 classes, contains more than 22k images for training and 5k images for validation. MS COCO dataset, which is divided into 80 classes, has more than 118k images for training and 5k images for validation.

Training and Inference. 1) For CIFAR-10 and CIFAR-100, we follow a similar training scheme in [10]. Networks are trained for 200 epochs with SGD optimizer with a weight decay of $5 \times e^{-4}$ and a momentum of 0.9. The learning rate is initialized to 0.05 and is decayed by 0.1 at 100 and 150 of the epochs. It trains with a mini-batch size of 128 on one GPU. Besides, according to different architectures of networks, we set (h, w) to $(8, 8)$ for ResNet, WideResNet, ResNeXt and set it to $(4, 4)$ for DenseNet. 2) For ImageNet-1K dataset, we follow standard practices and perform data augmentation with random cropping to size 224×224 pixels. We apply SGD with a momentum of 0.9 and a weight decay of $1 \times e^{-4}$. The initial learning rate is set to 0.1 and divided by every 30 epochs for a total of 100 epochs. 3) For PASCAL VOC dataset, we use SGD optimizer and set the batch size to 32. The initial learning rate is set to 0.1 with a 500 iterations warmup. We train 20k iterations totally and reduce the learning rate by a factor of 10 at 10k and 18k iterations. 4) For MS COCO dataset, we use SGD optimizer and set the initial learning rate to 0.1 and batch size to 16. The total iterations are set to 180k and the learning rate is divided by 10 at 120k and 160k iterations.

3.2. Ablation Studies

In this section, we conduct ablation studies to inspect the relative effectiveness of different components in the proposed SCConv module. We choose ResNet50 as the baseline network by replacing standard 3×3 convolution to conduct the following ablation experiments on CIFAR100.

SRU and CRU Our SCConv module includes a spatial reconstruction unit (SRU) and a channel reconstruction unit (CRU). Firstly we only apply SRU or CRU on ResNet50 to examine the efficiency of a single unit. As shown in Table 1, only embedding with SRU (ResNet50+S) achieves nearly 1% improvement without extra increment in FLOPs, while solely embedding with CRU (ResNet50+C) can save 38% parameters and FLOPs with 0.8% increase of Top-1 accuracy. The results imply that the single-unit-embedded model boosts the accuracy significantly. In addition, we compare three different ways of arranging the SRU and CRU: sequential spatial-channel (S+C), sequential channel-spatial (C+S), and parallel use of both units (C&S). We find that the spatial-first order (S+C) achieves the best accuracy

Table 1. Experimental results with different combination methods of SRU and CRU on CIFAR-100 dataset. (S and C is short for SRU and CRU respectively)

Description	Params	FLOPs	Top-1ACC(%)
ResNet50	23.71M	1.30G	78.60
ResNet50 + S	23.53M	1.30G	79.59
ResNet50 + C	14.74M	843.81M	79.21
ResNet50 + C & S	14.74M	843.81M	79.26
ResNet50 + C + S	14.74M	843.81M	79.54
ResNet50 + S + C	14.74M	843.81M	79.89

than other combination methods. Thus we adopt a sequential spatial-first combination (S+C) strategy to formulate our SCConv and further improve the model performance.

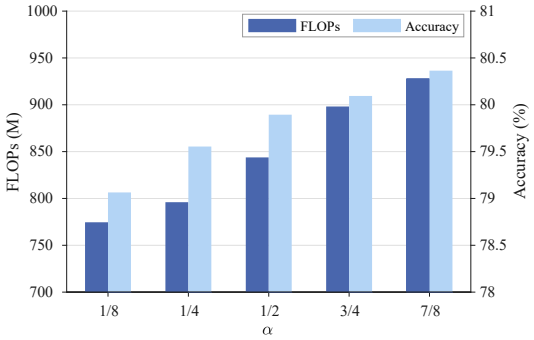

Figure 4. The trade-off between FLOPs and Accuracy on CIFAR-100 with different split ratios α in SCConv-embedded ResNet50.

Analysis on split ratio α To explore the effect of different split ratios α in CRU module, we vary the split ratio from 1/8 to 7/8 gradually to compare the accuracy versus FLOPs on CIFAR-100. As shown in Figure 4, the accuracy of SCConv-embedded ResNet50 rises with the increase of the split ratio α. The higher α represents the model can obtain richer feature information in the transformation stage of CRU, thereby improving the model's overall performance. When $\alpha = 1/2$, the whole network achieves the best flops-accuracy trade-off. Thus we adopt the optimal split ratio $\alpha = 1/2$ for SCConv in the following experiments for a better trade-off between performance and efficiency.

Visualization To explore the feature representation of the proposed SRU method, we visualize the feature maps of the first stage of the original ResNet50 and SRU-embedded ResNet50 in Figure 5. It can be observed that the pattern in features of SRU-embedded ResNet50 is enriched in comparison with original ResNet50. Not only the redundant features is diminished but the representative features are strengthened and diversified. The visualization demon-

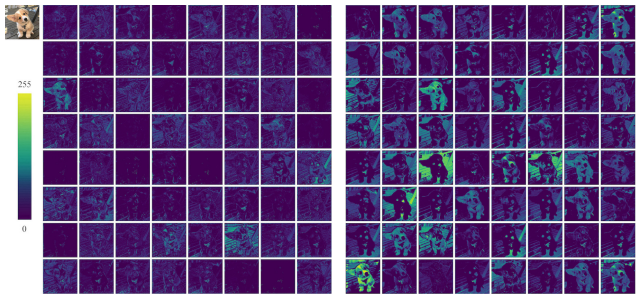

Figure 5. Left: Features from the first-stage of original ResNet50, Right: Features from the first-stage of SRU-embedded ResNet50.

strates the SRU is capable of generating representative and expressive features.

3.3. Image Classification on CIFAR

After studying the structure of the proposed SCConv module for efficient feature learning, we continue to evaluate the SCConv-embedded architecture on various baseline models and further make a comparison with the SOTA methods over classification accuracy, the number of parameters and FLOPs on CIFAR-10 and CIFAR-100 dataset. The related SOTA approaches include OctConv [3], GhostNet [8], SPConv [32], SlimConv [23], TiedConv [28]. All experiments are conducted by replacing the original convolutional layers with the corresponding convolution methods.

As can be seen in Table 2, in all cases, our SCConv-embedded models outperform all prior networks for accuracy. For ResNet-56 model, the SCConv-R56 requires only 62.7% parameters and FLOPs of the counterpart ResNet56 while bringing over a 1% accuracy increase on both datasets. For ResNet-50 model, the SCConv-R50 achieves better accuracy (nearly 1% and 1.3%) but around 37% parameters and 34% FLOPs reduction than the counterpart ResNet50 on CIFAR-10 and CIFAR-100. Besides, the SCConv-R50 requires the same computational costs as SlimConv-R50 while bringing higher (over 1%) promotion of accuracy. To show the generality of the proposed method, we apply SCConv and other SOTA methods to ResNeXt-29, WideResNet-28, and DenseNet-121. It can be observed that the SCConv-embedded models still achieve superior performance than other works with comparable model computations. For instance, the SCConv-RX29 achieves over 2.3% improvement of accuracy while the computation is on par with the GhostNet-RX29. The SCConv-WRN28 achieves better accuracy (nearly 1.3%) than the SlimConv-WRN28 while saving parameters and FLOPs by 11.7% and 15.5%.

3.4. Image Classification on ImageNet

We conduct experiments for ResNet50 on the ImageNet-1K dataset, comparing the performance of our approach

Table 2. Comparison of SOTA methods for common CNN architectures over Top-1 accuracy, the number of parameters and FLOPs on CIFAR-10 and CIFAR-100 dataset.

Network Architecture	FLOPs	Params	CIFAR-10 ACC(%)	CIFAR-100 ACC(%)
ResNet56 (R56)	126.84M	0.86M	93.27	71.50
OctConv-R56 (α=1/2)	126.84M	0.62M	93.11	70.93
SPConv-R56 (α=1/2)	87.95M	0.58M	93.49	71.51
GhostNet-R56 (s=2)	**68.35M**	**0.45M**	92.35	70.42
SlimConv-R56 (k=1)	88.82M	0.60M	93.51	71.71
TiedConv-R56 (b=2)	94.99M	0.54M	92.87	71.10
SCConv-R56 (Ours)	79.55M	0.52M	**94.12**	**72.56**
ResNet50 (R50)	1.30G	23.52M	95.09	78.60
OctConv-R50 (α=1/2)	727.26M	23.52M	95.25	78.91
GhostNet-R50 (s=2)	**632.45M**	**14.56M**	94.30	77.54
SlimConv-R50 (k=4/3)	853.50M	14.83M	95.05	78.85
SPConv-R50 (α=1/2)	972.31M	16.14M	95.32	79.23
SlimConv-R50 (k=1)	942.38M	16.78M	95.35	79.26
TiedConv-R50 (b=2)	998.78M	15.03M	95.44	79.52
SCConv-R50 (Ours)	831.18M	14.69M	**95.92**	**79.89**
ResNeXt-29 (RX29)	4.55G	28.27M	95.68	81.54
GhostNet-RX29 (s=2)	3.60G	**22.22M**	95.14	80.23
SPConv-RX29 (α=1/2)	4.79G	31.17M	96.03	81.76
SlimConv-RX29 (k=4/3)	4.38G	28.24M	95.85	81.93
TiedConv-RX29 (b=2)	5.96G	24.67M	95.66	81.32
SCConv-RX29 (Ours)	**3.57G**	22.31M	**96.20**	**82.56**
WideResNet-28 (WRN28)	5.96G	36.55M	95.21	79.40
GhostNet-WRN28 (s=2)	3.98G	22.12M	95.25	79.27
SPConv-WRN28 (α=1/2)	4.16G	24.20M	95.37	80.25
SlimConv-WRN28 (k=1)	4.25G	25.00M	95.43	79.52
TiedConv-WRN28 (b=2)	4.54G	22.03M	95.48	78.61
SCConv-WRN28 (Ours)	**3.75G**	**21.12M**	**95.64**	**80.83**
DenseNet-121 (D121)	898.23M	7.05M	95.09	79.43
GhostNet-D121 (s=2)	**517.36M**	**5.04M**	93.96	78.51
SPConv-D121 (α=1/2)	641.54M	5.69M	95.15	79.64
SlimConv-D121 (k=1)	670.21M	5.97M	94.63	78.90
TiedConv-D121 (b=2)	695.68M	5.45M	95.23	79.73
SCConv-D121 (Ours)	594.34M	5.45M	**95.37**	**80.24**

with the recent SOTA methods including OctConv [3], SPConv [32], GhostNet [8], SlimConv [23], PfLayer [7], TiedConv [28]etc. Noted that we only replace the bottleneck 3×3 convolutions with the corresponding convolution methods. As shown in Table 3, our SCConv-R50_α=1/2 can achieve 34.4% computation reductions with 0.26% accuracy increase over the ResNet50 model. When we further increase the split ratio α to 3/4, our approach gains superior performance to all other state-of-the-art methods in the growth of accuracy.

For instance, the SCConv-R50_α=3/4 achieves over 0.6% improvement of accuracy while the computation is on a par with the PfLayer-R50_max. In addition, the SCConv-R50_α=1/2 gains better performance than SlimConv-R50_k=4/3 with comparable calculations. To further prove the effectiveness of SCConv, we embed it with the deep model ResNet101. With nearly 62% of computation costs, our SCConv-R101 brings a 0.68% accuracy increase over the baseline model.

Besides, for better comparison, we select several state-of-the-art methods to draw the FLOPs v.s. Top-1 accu-

Table 3. Image classification results on ImageNet-1K dataset.

Network Architecture	FLOPs(G)	Params(M)	Top-1 (%)
ResNet50 (R50) (Baseline)	4.09	25.56	76.15
Versatile-R50 (NIPS2018)	**1.80**	18.7	75.50
GhostNet-R50_s=2 (CVPR2020)	2.15	13.95	75.18
SlimConv-R50_k=8/3 (TIP2021)	1.88	**12.10**	75.32
SPConv-R50_α=1/2 (IJCAI2020)	2.97	18.34	76.26
OctConv-R50_α=1/2 (CVPR2020)	2.40	25.56	76.34
SlimConv-R50_k=4/3 (TIP2021)	2.65	16.76	76.12
PfLayer-R50_max (ICLR2022)	2.90	18.00	76.15
SlimConv-R50_k=1 (TIP2021)	3.00	18.81	76.32
TiedConv-R50_b=2 (AAAI2021)	3.19	17.07	76.04
SCConv-R50_α=1/2	2.70	16.78	76.41
SCConv-R50_α=3/4	2.87	17.69	**76.79**
ResNet_101(R101)(Baseline)	7.83	44.55	77.25
SCConv-R101	**4.90**	**28.00**	**77.93**

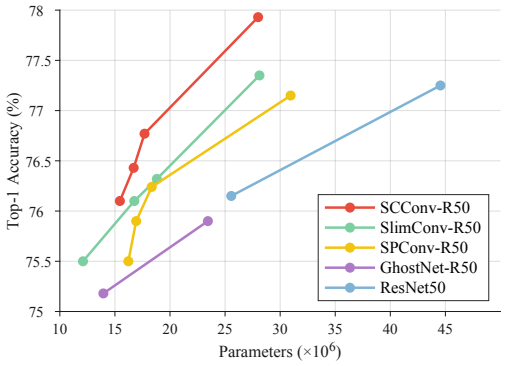

Figure 6. Top1-Accuracy v.s. FLOPs for ResNet50 on ImageNet.

racy curve. From an overall view of Figure 6, the curve of our proposed model is above all other methods including ResNet, GhostNet, SP-ResNet, and SlimConv. It shows that our proposed model gains higher accuracy with lower computation costs. As for the same performance, the model with our SCConv performs more compactly than others.

3.5. Object Detection

In order to further evaluate the generalization ability of SCConv, we conduct experiments on two datasets for object detection tasks. The one-stage RetinaNet [20] is used as a detection framework. We adopt the backbone network of ResNet-50, ResNet-101, and SCConv-embedded model acts as a drop-in replacement for the backbone feature extractor.

Table 4. Object detection experiments on the PASCAL VOC 2007 and 2012 dataset.

Backbone	Params(M)/FLOPs(G)	AP@.5	AP@.75	mAP@[.5,.95]
ResNet50(R50)	25.56/63.09	77.89	55.31	52.26
SPConv-R50	19.76/49.23	78.05	55.47	52.48
SlimConv-R50	18.81/47.12	77.96	55.38	52.42
SCConv-R50	**16.78/41.36**	**78.68**	**56.26**	**53.16**
ResNet101(R101)	44.55/121.3	79.23	56.31	53.32
SCConv-R101	**27.90/75.26**	**80.36**	**57.05**	**54.12**

Table 5. Object detection results on MS COCO val2017.

Backbone	Params(M)/FLOPs(G)	AP@.5	AP@.75	mAP@[.5, .95]
ResNet50(R50)	25.56/63.09	54.2	37.4	35.2
SPConv-R50	19.76/49.23	54.5	37.6	35.3
SlimConv-R50	18.81/47.12	54.0	37.1	35.0
SCConv-R50	**16.78/41.36**	**55.1**	**38.2**	**35.6**

For the PASCAL VOC dataset, as shown in Table 4, the AP@[.5] of RetinaNet with the SCConv-R50 and SCConv-R101 are 78.68% and 80.36%, exceeding original ResNet50 and ResNet101 by 0.8% and 1.1% while reducing parameters and FLOPs by 34.1% and 37%. For the MS COCO dataset, as shown in Table 5, the AP@[.5] of RetinaNet with SCConv-R50 is 55.1%, outperforming original ResNet-50 by 0.9% with over 22G FLOPs decreased.

In addition, our approach consistently surpasses current state-of-the-art methods on both datasets. For instance, the mAP@[.5,.95] of SCConv-R50 exceeds the SlimConv-R50 by nearly 0.8% and 0.6% on PASCAL VOC and MS COCO datasets. In short, these results prove that the SCConv module not only brings performance improvement but helps the network learn better representative features with a smaller amount of parameters, making it possible for object detection to be deployed on resource-limited devices.

4. Conclusion

In this paper, we have presented a novel spatial and channel reconstruction module (SCConv), an efficient architectural unit to decrease computational cost and model storage while improving the performance of CNN models by reducing spatial and channel redundancies that widely exist in standard convolution. We diminish the redundancy in feature maps with two distinctive modules, SRU and CRU, which achieve considerable performance improvement while cutting a substantial amount of computation loads. Besides, SCConv is a plug-and-play module and generic to replace the standard convolution without any model architecture adjustment. In addition, the extensive experiments with various SOTA methods on image classification and object detection have indicated the superiority of SCConv-embedded models for striking a much better trade-off between performance and model efficiency. Finally, we hope the proposed method can inspire research for more efficient architectural design.

Acknowledgments This work was supported in part by the 2030 National Key Research and Development Program of China (2018AAA0100500), National Nature Science Foundation of China (62273150, 62171323), Shanghai Natural Science Foundation (22ZR1421000), Shanghai Outstanding Academic Leaders Plan (21XD1430600), Science and Technology Commission of Shanghai Municipality (22DZ2229004).

References

[1] Jierun Chen, Tianlang He, Weipeng Zhuo, Li Ma, Sangtae Ha, and S-H Gary Chan. Tvconv: Efficient translation variant convolution for layout-aware visual processing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12548–12558, 2022. 1

[2] Yinpeng Chen, Xiyang Dai, Dongdong Chen, Mengchen Liu, Xiaoyi Dong, Lu Yuan, and Zicheng Liu. Mobile-former: Bridging mobilenet and transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5270–5279, 2022. 1

[3] Yunpeng Chen, Haoqi Fan, Bing Xu, Zhicheng Yan, Yannis Kalantidis, Marcus Rohrbach, Shuicheng Yan, and Jiashi Feng. Drop an octave: Reducing spatial redundancy in convolutional neural networks with octave convolution. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3435–3444, 2019. 2, 7

[4] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1251–1258, 2017. 1

[5] Emily L Denton, Wojciech Zaremba, Joan Bruna, Yann LeCun, and Rob Fergus. Exploiting linear structure within convolutional networks for efficient evaluation. *Advances in neural information processing systems*, 27, 2014. 1

[6] Mark Everingham, Luc van Gool, C.K.I. Williams, John Winn, and A. Zisserman. The PASCAL visual object classes challenge 2007 (VOC2007) results, 2007. 5

[7] Dongyoon Han, YoungJoon Yoo, Beomyoung Kim, and Byeongho Heo. Learning features with parameter-free layers. *arXiv preprint arXiv:2202.02777*, 2022. 7

[8] Kai Han, Yunhe Wang, Qi Tian, Jianyuan Guo, Chunjing Xu, and Chang Xu. Ghostnet: More features from cheap operations. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1580–1589, 2020. 2, 7

[9] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015. 1

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 1, 6

[11] Geoffrey Hinton, Oriol Vinyals, Jeff Dean, et al. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2(7), 2015. 1

[12] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017. 1, 2

[13] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. In *Computer Vision and Pattern Recognition*, 2017. 2

[14] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017. 1

[15] Alex Krizhevsky. Learning Multiple Layers of Features from Tiny Images. *Technical report, University of Toronto*, 2009. 5

[16] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet classification with deep convolutional neural networks. In *Neural Information Processing Systems*, pages 1097–1105, 2012. 2, 5

[17] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. *arXiv preprint arXiv:1608.08710*, 2016. 1

[18] Xiang Li, Wenhai Wang, Xiaolin Hu, and Jian Yang. Selective kernel networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 510–519, 2019. 5

[19] Yunsheng Li, Yinpeng Chen, Xiyang Dai, Dongdong Chen, Mengchen Liu, Lu Yuan, Zicheng Liu, Lei Zhang, and Nuno Vasconcelos. Micronet: Improving image recognition with extremely low flops. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 468–477, 2021. 1

[20] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017. 8

[21] Tsung Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft COCO: Common objects in context. In *European Conference on Computer Vision*, pages 740–755, 2014. 5

[22] Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. Learning efficient convolutional networks through network slimming. In *Proceedings of the IEEE international conference on computer vision*, pages 2736–2744, 2017. 1

[23] Jiaxiong Qiu, Cai Chen, Shuaicheng Liu, Heng-Yu Zhang, and Bing Zeng. Slimconv: Reducing channel redundancy in convolutional neural networks by features recombining. *IEEE Transactions on Image Processing*, 30:6434–6445, 2021. 2, 7

[24] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015. 1

[25] Pravendra Singh, Vinay Kumar Verma, Piyush Rai, and Vinay P Namboodiri. Hetconv: Heterogeneous kernel-based convolutions for deep cnns. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4835–4844, 2019. 2

[26] Xinglong Sun, Ali Hassani, Zhangyang Wang, Gao Huang, and Humphrey Shi. Disparse: Disentangled sparsification for multitask model compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12382–12392, 2022. 1

[27] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR, 2019. 1

[28] Xudong Wang and X Yu Stella. Tied block convolution: leaner and better cnns with shared thinner filters. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 10227–10235, 2021. 2, 7

[29] Yuxin Wu and Kaiming He. Group normalization. In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19, 2018. 3

[30] Mengzhou Xia, Zexuan Zhong, and Danqi Chen. Structured pruning learns compact and accurate models. *arXiv preprint arXiv:2204.00408*, 2022. 1

[31] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500, 2017. 1

[32] Qiulin Zhang, Zhuqing Jiang, Qishuo Lu, Jia'nan Han, Zhengxin Zeng, Shang-Hua Gao, and Aidong Men. Split to be slim: An overlooked redundancy in vanilla convolution. *arXiv preprint arXiv:2006.12085*, 2020. 2, 7

[33] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6848–6856, 2018. 2

[34] Borui Zhao, Quan Cui, Renjie Song, Yiyu Qiu, and Jiajun Liang. Decoupled knowledge distillation. In *Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition*, pages 11953–11962, 2022. 1

[35] Daquan Zhou, Qibin Hou, Yunpeng Chen, Jiashi Feng, and Shuicheng Yan. Rethinking bottleneck structure for efficient mobile network design. In *European Conference on Computer Vision*, pages 680–697. Springer, 2020. 1