2023 IEEE International Conference on Multimedia and Expo (ICME) | 978-1-6654-6891-6/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICME55011.2023.00323

SCFormer: Integrating hybrid Features in Vision Transformers

Hui Lu[1*], Ronald Poppe[1], Albert Ali Salah[1,2]
[1] Department of Information and Computing Sciences, Utrecht University, Utrecht
[2] Department of Computer Engineering, Bogazici University, Istanbul, Turkey
{h.lu1, R.W.Poppe, a.a.salah}@uu.nl

***Abstract*—Hybrid modules that combine self-attention and convolution operations can benefit from the advantages of both, and consequently achieve higher performance than either operation alone. However, current hybrid modules do not capitalize directly on the intrinsic relation between self-attention and convolution, but rather introduce external mechanisms that come with increased computation cost. In this paper, we propose a new hybrid vision transformer called Shift and Concatenate Transformer(SCFormer), which benefits from the intrinsic relationship between convolution and self-attention. SCFormer roots in the Shift and Concatenate Attention (SCA) block, that integrates convolution and self-attention features. We propose a shifting mechanism and corresponding aggregation rules for the feature integration of SCA blocks such that generated features more closely approximate the optimal output features. Extensive experiments show that, with comparable computational complexity, SCFormer consistently achieves improved results over competitive baselines on image recognition and downstream tasks. Our code is available at: https://github.com/hotfinda/SCFormer.**

***Index Terms*—Vision transformer, hybrid module, feature integration**

I. Introduction

Convolutional neural networks (CNNs) and vision transformers have achieved remarkable performance. Considering the inherently different feature extraction process of CNNs and transformers, researchers are motivated to utilize the advantages of both and extract optimal features through the integration of features from both CNNs and transformers.

Previous work has explored the combination of self-attention (SA) and convolution operations in two ways. One option is to consider SA and convolution operations independently and aggregate their outputs through summation or multiplication. Examples of this approach include SAN [1], AA-ResNet [2], and Container [3]. A second line of approach focuses on integrating convolution and SA operations in a mixed path to exchange features through matrix projection or insert operations. This approach is pursued in Mixformer [4], Conformer [5], and MobileFormer [6]. While existing approaches to combine SA and convolutions achieve better performance than their uni-modal counterparts, they unavoidably introduce additional CNN modules to enrich SA modules. These additional modules make models computationally expensive. Moreover, these CNN modules are designed for certain model structures or integration methods. When integrating these CNN modules with SA modules in different model structures, feature details could deteriorate. Thus it remains a challenging task to optimally combine outputs of convolution and SA without heavily increasing the computation cost.

In this paper, we explore an alternative approach to integrate convolution and self-attention outputs. We utilize the intrinsic relationship between convolution and SA by reusing the value map of SA layers as the source of integrated CNN features, and develop a mixed Shift and Concatenate Attention (SCA) module. The SCA module elegantly integrates features from convolution and SA with minimal computational overhead. We first project the input feature maps with 1×1 convolutions to obtain a set of intermediate features. After the computation of the SA layer, these intermediate features are reused to generate CNN features. The CNN and SA layer features further pass through a novel shifting block that increases the receptive field. The final complementary features are obtained after aggregation. In this way, we benefit from the capabilities of convolution and SA operations, while reducing the computation effort. Our contributions include:

- We propose a computationally efficient Shift and Concatenate Attention (SCA) block to combine features of CNN and SA. We also introduce SCFormer, a vision transformer with integrated SCA blocks.
- A shifting mechanism is proposed in the SCA block to integrate the complementary contribution of self-attention and convolution.
- Our SCFormer achieves state-of-the-art performance on image classification, object detection and semantic segmentation tasks, demonstrating its use as an efficient general purpose vision transformer.

II. Related Work

Owing to the ability to capture long-range dependencies, self-attention is mainly introduced as non-local alternative to CNN blocks [7]. SAGAN [8] introduced SA modules into the generator and the discriminator of a GAN so that both efficiently model relationships between spatially distant regions. Relation Networks [9] use an object attention module as an embedding in existing networks. The object attention module processes a set of objects simultaneously through the interaction of their appearance features and geometry. Despite the performance improvement of transformers over CNNs, it is deemed necessary to complement transformer models with convolution operations to introduce additional inductive biases [10]. CvT [11] adopts convolution in the tokenization process and utilizes strided convolutions to reduce the computation complexity of SA. Xiao et al. [12] demonstrated that a standard, lightweight convolutional stem at the early stage of vision transformers is more robust. Swin Transformer [13] adopts a convolution-based positional encoding technique and shows improvements on downstream tasks such as image classification and segmentation.

Researchers have proposed different ways to integrate convolution and SA features. We distinguish between two main approaches. One way is to treat the convolution and SA operations independently and to aggregate features in a subsequent step. AA-Resnet [2] applies SA in parallel to a standard convolution operation and concatenates the outputs. Lu et al. [3] propose the Container block, which aggregates the outputs of both operations using an affinity matrix. Another line of research focuses on integrating convolution and SA operations in an embedded path to share or exchange both features. Mixformer [4] utilizes matrix projection to embed features of depth-wise convolutions into the SA operation. The features are integrated by concatenating the outputs. Conformer [5] adds features directly to the pipeline of the other path and finally integrates the features. MobileFormer [6] follows the same strategy while having fewer parameters.

While existing approaches improve the model performance, the improvement is based on additional CNN modules that enrich the SA features. This approach relies on CNN modules that are specifically designed for certain model structures or integration method. When used in a different model structure, feature details could deteriorate. Another drawback is that the additional CNN modules add parameters and thus make the network computationally more expensive.

We deviate from existing works and propose the Shift and Concatenate Attention(SCA) block to integrate features of convolution and SA without introducing additional convolution modules. The SCA block reusing the value map features of SA layers as the source of integrated CNN features. We further propose a shifting mechanism and corresponding aggregation rules to expand the receptive field. Such a structure not only naturally inherits the advantages of both CNNs and transformers but also remains computationally efficient.

III. Method

A. SCA block

Our proposed SCA block (see Figure 1) makes two key modifications to the standard self-attention block. First, the SCA block uses a shifting operation to remove the overlap between the focus regions of SA and CNN. Second, we introduce an aggregation method to generate features from SA and convolution.

Shifting block. SA and CNN will focus on partly overlapping regions, which we will refer to as "shared" regions. Except for these shared regions, SA tends to focus on textured regions to generate low-frequency features, while CNNs tend to focus on the boundary regions to generate high-frequency features. This could explain why a hybrid model uses convolutions in early stages and SA in later stages [14]. Because the features in shared regions are already processed by SA or CNN, our intuition is that we can move the focus area of SA from shared regions towards textured regions. This way, we can not only collect more features, but also obtain a larger receptive field. To achieve this shift, we propose a shifting mechanism for the self-attention output following the form:

$$y_i = \alpha_i - \gamma\left(\alpha_i, \beta_i\right) \odot \alpha_i \tag{1}$$

$$\gamma\left(\alpha_i, \beta_i\right) = Softmax(\alpha_i \odot \beta_i) \tag{2}$$

where y_i is the output of size $1 \times 1 \times C$, α_i is the feature of size $1 \times 1 \times C$ in the SA result, and β_i is the feature of size $1 \times 1 \times C$ in the value map. The relation function γ outputs a single vector that represents the relation between α_i and β_i, $\odot$ is the Hadamard product.

We take β_i as the CNN feature since β_i comes from a convolution block. y_i represents the new SA feature with no relation to β_i, which also means that Equations 1 and 2 remove the shared regions from the focus areas of SA. Instead, we then shift the self-attention towards the texture regions. Following [15], we keep query map Q and key map K unchanged, and the capacity of information that should be obtained at each position i of the SA result can be calculated as follows:

$$\hat{I}_i = \phi(Q_i) \sum_{j=1}^{m} \frac{\phi(K_j)}{O_j} \tag{3}$$

where m is the number of tokens in the key map and $\frac{\phi(K_j)}{O_j}$ is the normalization operation for the token at position j in key map.

We take $\alpha_i y_i^T$ as the similarity of features at position i, because the similarity value will be much smaller when the position is part of the shared region compared to the textured region. We can increase the information capacity of the textured region to get the final SA result r_i through:

$$r_i = LReLU(\alpha_i y_i^T * \hat{I}_i) \odot y_i \tag{4}$$

Aggregation. Instead of utilizing the simple summation operation to aggregate the features from CNN and SA, we firstly concatenate the features at the same position of CNN and SA in the channel dimension and then use a 1×1 convolution layer to aggregate the features. The advantage of this approach is that we not only increase the depth channel for the later stage, but also keep the opportunity for feature transformation in the next stage to better process the features.

B. SCFormer architecture

Overall architecture. Based on the SCA block, we propose our SCFormer architecture, which follows residual networks. SCFormer is efficient and easy to implement, as shown in Figure 1. Another advantage of this design is that we can directly compare to recent works that use a similar structure, e.g. EfficientNets [16], SAN [1], and Mixformer [4].

SCFormer consists of four parts: convolution stem, SCA block, projection layer, and classification head, see Figure 1. The convolution stem acts as an initial encoder. In four stages,

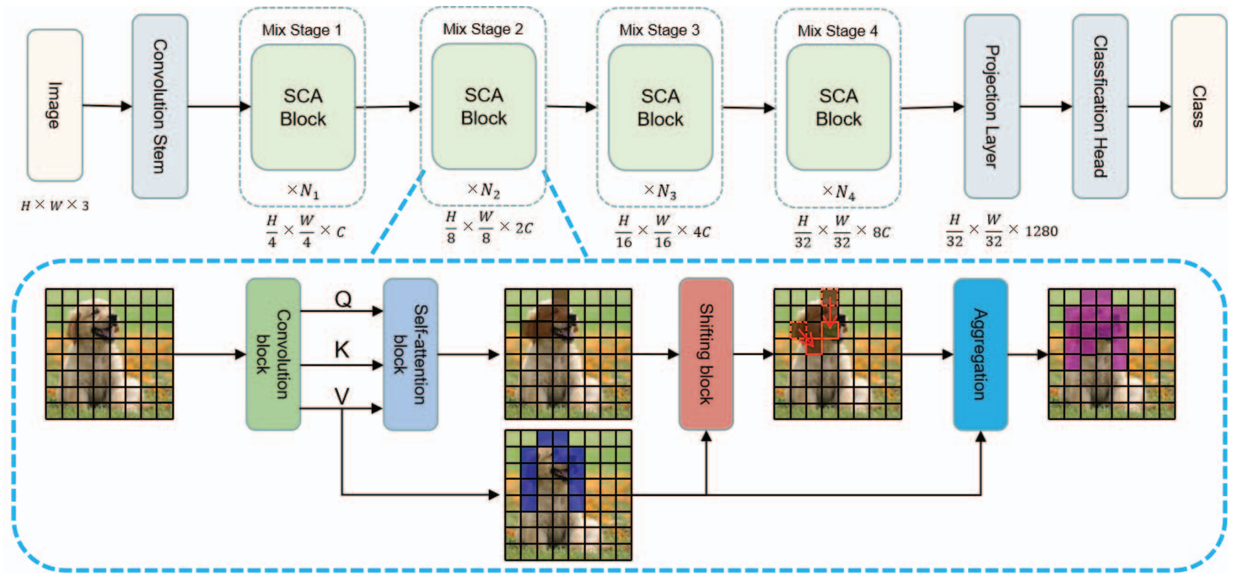

Fig. 1: **Overall pipeline of SCA block**. The convolution block processes the input feature map and generates initial features. The self-attention block processes features from the convolution block and generates features that mainly focus on textured regions. The self-attention regions move toward other textured regions after passing through the shifting block, and the final result is generated by aggregating the value map and the deformed self-attention output.

we stack SCA blocks with downsampling rates of 4, 8, 16, 32 inside the convolution block, respectively. We then use a linear projection layer to increase the number of channels to 1280 to preserve more detail in the channel. Finally the classification head provides the final classification.

Architecture variants. We define four architecture variants (B1-B4) by stacking blocks of different sizes. The number of blocks in four stages is set by following residual networks and SAN networks [1], and the channel number of the four stages is set as 256, 512, 1024, and 2048, respectively. Details of the four variants appear as follows:

- B1: head numbers=$\{2, 4, 8, 16\}$, blocks=$\{1, 2, 4, 1\}$
- B2: head numbers=$\{2, 4, 8, 16\}$, blocks=$\{2, 3, 5, 2\}$
- B3: head numbers=$\{3, 6, 12, 24\}$, blocks=$\{3, 4, 6, 3\}$
- B4: head numbers=$\{4, 8, 16, 32\}$, blocks=$\{3, 5, 8, 4\}$

IV. Experiments

We evaluate our SCA block and its use in SCFormer on image classification (Section IV-A), object detection (Section IV-B) and semantic segmentation (Section IV-C). We then compare to related architectures. In Section IV-E, we provide a qualitative evaluation of SCFormer. We further present the ablation study on the effects of aggregation method, feature map type, and number of blocks per stage. Finally, we validate the performance of our SCFormer on datasets of limited size.

A. Image classification

We first verify our method on the image classification task using the ImageNet-1K dataset. To make a fair comparison with previous works, the setting follows [4]. We use the AdamW optimizer with a cosine decay schedule. All models are trained for 300 epochs, and the input image size is 224×224. The results are shown in Table I.

Results. We compared the four SCFormer variants to the state-of-the-art, including ConvNets, vision transformers, and a combination of both. We explicitly evaluate several variants of the same architecture to better understand the relation between model size, number of computations and classification performance. Our method outperforms all baselines with comparable FLOPs and parameters. For example, compared to vision transformers, SCFormer-B4 obtains a 83.7% top-1 accuracy, which is 0.7% higher than Swin-S with 38% fewer parameters and 60% fewer FLOPs. With a combination of CNN and transformer, the formerly best performing works are MixFormer-B4 and Swin-ACmix-S. Our largest architecture variant SCFormer-B4 outperforms these networks by +0.7% and +0.2% respectively. SCFormer-B4 has a comparable computational complexity to MixFormer-B4 but has only 61% the number of parameters and only performs 39% of the number of FLOPs of Swin-ACmix-S. SCFormer is both more efficient and more effective.

B. Object detection

We continue with an experiment on COCO-2017 dataset to evaluate the effectiveness of SCFormer on object detection. We use our SCFormer-B4 as backbone with Mask R-CNN [19] and Cascade Mask R-CNN [20] as the detection heads. To better compare with other models, we follow [4] and adopt the $1\times$ and $3\times$ schedule to train different models.

Results. For Table II, we observe that SCFormer-B4 consistently shows better performance than the baselines under different train schedules and detection heads. For example,

TABLE I: **Image classification accuracy** on ImageNet-1K validation. Methods grouped into convolution, transformer and combination. Best top-1 accuracy in **bold**.

Method	#Params	FLOPs	Top-1
ConvNets			
EffNet-B1 [16]	8M	0.7G	79.1
EffNet-B2 [16]	9M	1.0G	80.1
EffNet-B3 [16]	12M	1.8G	81.6
EffNet-B4 [16]	19M	4.2G	82.9
Vision transformers			
DeiT-T [17]	6M	1.3G	72.2
DeiT-S [17]	22M	4.6G	79.9
DeiT-B [17]	87M	17.5G	81.8
PVT-T [18]	13M	1.8G	75.1
PVT-S [18]	25M	3.8G	79.8
PVT-M [18]	44M	6.7G	81.2
PVT-L [18]	61M	9.8G	81.7
CvT-13 [11]	20M	4.5G	81.6
CvT-21 [11]	32M	7.1G	82.5
Swin-T [13]	29M	4.5G	81.3
Swin-S [13]	50M	8.7G	83.0
Combination of CNN and transformers			
AA-ResNet-50 [2]	25.8M	4.2G	77.7
MixFormer-B1 [4]	8M	0.7G	78.9
MixFormer-B2 [4]	10M	0.9G	80.0
MixFormer-B3 [4]	17M	1.9G	81.7
MixFormer-B4 [4]	35M	3.6G	83.0
Swin-ACmix-T [10]	30M	4.6G	81.9
Swin-ACmix-S [10]	51M	9.0G	83.5
SCFormer-B1(**Ours**)	8M	0.7G	78.9
SCFormer-B2(**Ours**)	10M	0.9G	80.0
SCFormer-B3(**Ours**)	16M	1.9G	82.0
SCFormer-B4(**Ours**)	31M	3.5G	**83.7**

TABLE II: **Object detection on MS COCO.** We compare using two schedules and two backbones. Best results in **bold**.

Method	Backbone	#	FLOPs	AP^m	AP^m_{50}	AP^m_{75}
Mask R-CNN	ResNet50 [21]	1×	260G	34.4	55.1	36.7
	Swin-T [13]	1×	264G	39.1	61.6	42.0
	MixFormer-B4 [4]	1×	243G	41.2	64.3	44.1
	SCFormer-B4(**Ours**)	1×	240G	**42.0**	**65.3**	**45.0**
Cascade Mask R-CNN	ResNet50 [21]	3×	739G	40.1	61.7	43.4
	Swin-T [13]	3×	745G	43.7	66.6	47.1
	Shuffle-T [22]	3×	746G	44.1	66.9	48.0
	SCFormer-B4(**Ours**)	3×	717G	**45.5**	**68.5**	**49.4**

compared to Swin-T [13], SCFormer-B4 achieves +2.9 higher mAP under the $1\times$ schedule with Mask R-CNN, and +1.8 higher mask mAP in the $3\times$ schedule with Cascade Mask R-CNN. The number of FLOPs for all methods is comparable. Again, this demonstrates that the improvements are not achieved from using a more complex model, but instead from the ability to encode more informative features.

C. Semantic segmentation

We also evaluate the effectiveness of SCFormer on a challenging scene parsing dataset: ADE20K [23]. We use UPerNet [24] as the segmentation method with different backbones pretrained on ImageNet-1K. For training, we mainly follow the setting in [13], and a resolution of 512×2048 is used.

TABLE III: **Semantic segmentation results** on ADE20K validation split with single scale testing. Best mIoU in **bold**.

Backbone	#Params	FLOPs	mIoU
ResNet-101 [21]	86M	1029G	43.8
DeiT-S [17]	52M	1099G	44.0
Swin-T [13]	60M	945G	44.5
Focal-T [25]	62M	998G	45.8
Shuffle-T [22]	60M	949G	46.6
TwinsP-S [26]	55M	919G	46.2
ACmix-Swin-T [10]	60M	950G	45.3
MixFormer-B4 [4]	63M	918G	46.8
SCFormer-B4 (**Ours**)	60M	914G	**47.7**

Results. From Table III, we conclude that SCFormer outperforms other backbones with fewer or a comparable number of parameters and FLOPs. For example, with the same number of parameters, SCFormer-B4 outperforms Swin-T by +3.2 on mIoU, and ACmix by +2.1. These results confirm the merits of SCFormer.

TABLE IV: **Image classification results with other networks.** In ResNets and SAN, we replace the original blocks in the last stage with SCA blocks, and adjust the channel depth of the original network (denoted with *). Best results in **bold**.

Models	FLOPs	Params	Top-1
ResNet-26 [21]	2.4G	13.7M	75.5
ResNet-26 + Swin-T [13]	2.5G	15.8M	78.0
ResNet-26 + ACmix [10]	2.5G	15.9M	78.0
ResNet-26 + SCA	2.6G	16.1M	**78.7**
ResNet-26* + SCA	2.4G	13.6M	**78.4**
ResNet-50 [21]	4.1G	25.6M	79.0
ResNet-50 + Swin-T [13]	4.3G	28.4M	80.0
ResNet-50 + ACmix [10]	4.4G	28.8M	80.3
ResNet-50 + SCA	4.4G	29.0M	**81.1**
ResNet-50* + SCA	4.1G	23.7M	**80.9**
SAN-10 [1]	1.9G	11.8M	79.1
SAN-10 + Swin-T [13]	1.9G	12.1M	79.6
SAN-10 + ACmix [10]	1.9G	12.3M	79.6
SAN-10 + SCA	2.0G	12.4M	**80.0**
SAN-10* + SCA	1.9G	11.8M	**79.8**
SAN-19 [1]	3.3G	20.5M	80.2
SAN-19 + Swin-T [13]	3.4G	21.2M	80.6
SAN-19 + ACmix [13]	3.5G	21.9M	80.7
SAN-19 + SCA	3.5G	22.1M	**81.2**
SAN-19* + SCA	3.3G	19.3M	**80.9**

D. Generalization to other networks

We now evaluate the ImageNet-1K image classification performance of the SCA block in ResNet [21] and SAN [1] architectures. Following [4], we replace all the blocks in the last stage with our SCA block. The networks are trained as in Section IV-A. Results appear in Table IV.

Our SCA block can provide consistent gains. For example, SCA Block brings +3.2% and +2.1% top-1 accuracy over ResNet-26 and ResNet-50, respectively. By adjusting the depth

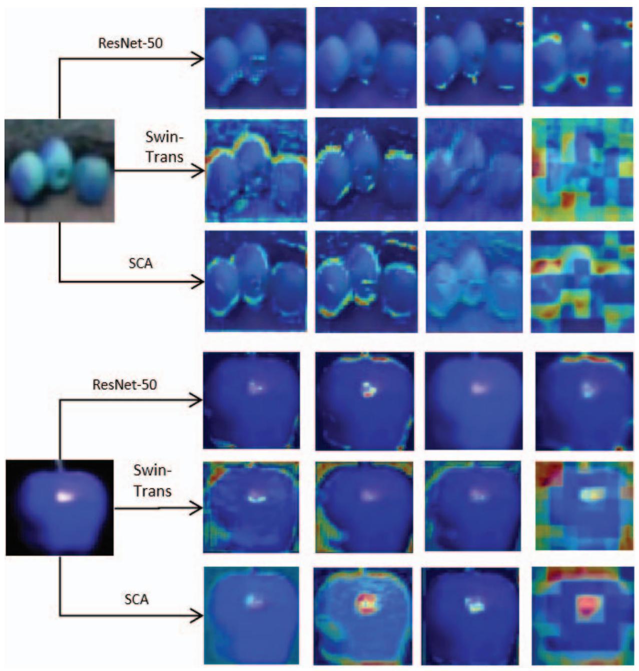

Fig. 2: **Grad-CAM visualization** of different layers of ResNet-50, ResNet-50+Swin-T and ResNet-50+SCFormer.

TABLE V: **Feature aggregation strategies.** We evaluate SCFormer-B4 on the image classification, object detect, and image segmentation tasks. Best results in **bold**.

Integration	ImageNet-1K		MS COCO	ADE20k
	Top-1	Top-5	AP^m	mIoU
Dot product	82.6	96.1	40.4	46.1
Subtraction	82.7	96.1	40.4	46.3
Summation	83.1	96.3	41.1	47.2
Concatenation	**83.7**	**96.8**	**42.0**	**47.7**

of feature maps, we obtain the ResNet* and SAN* with same FLOPs and similar numbers of parameters as ResNet and SAN while obtaining higher accuracy.

E. Qualitative evaluation

To better understand how the SCA module is working, we use Grad-CAM to visualize the feature maps of three ResNet-50 networks with the original convolution blocks in the last stage, and with blocks replaced by Swin-Transformers or SCA blocks. The networks are trained as in Section IV-A. The visualization results for two images are shown in Figure 2.

The networks focus on different regions at different stages. CNN layers focus on small regions in the feature maps, and few pixels in the objects are identified by the network, we might explain the lower accuracy. Compared to CNN layers, self-attention layers clearly have larger regions of attention, which results in higher accuracy. Our SCA layers have richer information of the object and combine useful attention areas from CNN layers and self-attention layers to obtain complementary information. This is consistent with the design of our SCA layer.

TABLE VI: **Different feature maps** in image classification, object detection, and image segmentation with SCFormer-B4.

Feature map	ImageNet-1K		MS COCO	ADE20k
	Top-1	Top-5	AP^m	mIoU
1×1 conv + 3×3 conv	81.5	95.4	39.4	45.0
Value map	83.7	96.8	42.0	47.7
Value map + 3×3 conv	83.9	96.9	42.4	47.8

TABLE VII: **Image classification with different numbers of blocks** in each stage SCFormer-B4, evaluated on ImageNet-1K.

Structure	Channels	FLOPs	# Param	Top-1
1	stage1:(64, 128, 256) stage2:(512*5) stage3:(1024*8) layer4: (2048*4)	3.5G	31.0M	83.7
2	stage1:(64, 128, 256, 256) stage2:(512*5) stage3:(1024*8) stage4: (2048*4)	3.5G	31.5M	83.7
3	stage1:(64, 128, 256) stage2:(512*6) stage3:(1024*8) stage4: (2048*4)	3.5G	32.2M	83.7
4	stage1:(64, 128, 256) stage2:(512*5) stage3:(1024*9) stage4: (2048*4)	3.6G	33.0M	83.8
5	stage1:(64, 128, 256) stage2:(512*5) stage3:(1024*8) stage4: (2048*5)	3.7G	35.1M	83.9

F. Ablation: Effect of aggregation method

We experiment with alternative feature aggregation methods: summation, subtraction, and dot product. We consider the same tasks image classification, object detection, and image segmentation. We use the SCFormer-B4 model and apply the same training procedures and datasets.

In Table V, self-concatenation is the most effective way of aggregation. The other strategies show comparable but lower performance. A reasonable assumption is that summation, subtraction and dot product cannot take into account the scale of the features, so that the network misses information.

G. Ablation: Value map versus other feature maps

To explore the design spirit of utilizing the value map and the output of SA, we utilize feature maps from different methods and summarize the results in Table VI.

In Table VI, the value map consistently performs better than utilizing features from another 1×1 conv + 3×3 convolution block. This verifies our hypothesis that the value map enables better feature representation learning. Introducing additional convolution modules after the value map also slightly increases the performance, but at increased computation cost.

H. Ablation: Number of blocks per stage

Our SCFormer utilizes the same structure as residual networks, which includes four stages. In each stage, a number of SCA blocks is used. To explore the sensitively to different numbers of blocks per stage, we take SCFormer-B4 as the

Models	FLOPs	Top-1	Top-5
ResNet-50	4.1G	82.6	94.5
ResNet-50 + ACmix	4.4G	83.2	95.8
ResNet-50 + Swin-Transformer	4.3G	83.1	95.6
ResNet-50* + SCA	4.1G	83.8	96.1
MobileNetV2	0.3G	70.8	90.4
MobileNetV2 + ACmix	0.3G	71.5	91.6
MobileNetV2 + Swin-Transformer	0.3G	71.4	91.4
MobileNetV2 + SCA	0.3G	71.9	92.0

TABLE VIII: **Image classification on CIFAR-100.** We substitute our SCA block in ResNet-50 and MobileNetV2.

baseline and systematically increase the number of blocks in each stage. Results for image classification on ImageNet-1K are shown in Table VII.

In Table VII, by comparison with first three rows we can see that increasing the number of blocks in the early stage has almost no influence on the performance. Comparing the fourth and fifth row, we can see that the optimal choice is to use more blocks in later stages.

I. Application to small-scale datasets

Here we validate the performance of our SCA block on datasets of limited size. To make comparison with other methods, we follow [14] and replace all blocks in the last stage of the original networks with our SCA block. We apply our SCA block to widely used ConvNets ResNet-50 and MobileNetV2, and utilize the CIFAR-100 dataset.

Table VIII shows that the SCA block acts as a suitable alternative to ConvNet blocks and provides gains on small dataset. For example, SCA brings +1.1% and +1.2% top-1 accuracy over MobileNetV2 and ResNet-50, respectively. Compared to ACmix, SCA also shows better performance with +0.4% and +0.6% over MobileNetV2 and ResNet-50, respectively.

V. Conclusion

We have introduced a Shift and Concatenate Attention (SCA) block to combine features from convolution operations and self-attention without an increase in the computational overhead. The block allows for an expanded receptive field, while the focus is directed to regions that are naturally more informative. We employ the SCA block in a novel transformer architecture SCFormer. Extensive experiments on image classification, object detection and image segmentation demonstrate the effectiveness and efficiency of our approach.

Acknowledgment

This work is supported in part by the scholarship from China Scholarship Council (CSC) under the Grant No.202106290068.

References

[1] Hengshuang Zhao, Jiaya Jia, and Vladlen Koltun, "Exploring self-attention for image recognition," in *CVPR*, 2020, pp. 10076–10085.
[2] Irwan Bello, Barret Zoph, Ashish Vaswani, Jonathon Shlens, and Quoc V Le, "Attention augmented convolutional networks," in *ICCV*, 2019, pp. 3286–3295.
[3] Jiasen Lu, Roozbeh Mottaghi, Aniruddha Kembhavi, et al., "Container: Context aggregation networks," *NeurIPS*, vol. 34, pp. 19160–19171, 2021.
[4] Qiang Chen, Qiman Wu, Jian Wang, Qinghao Hu, Tao Hu, Errui Ding, Jian Cheng, and Jingdong Wang, "Mixformer: Mixing features across windows and dimensions," in *CVPR*, 2022, pp. 5249–5259.
[5] Zhiliang Peng, Wei Huang, Shanzhi Gu, Lingxi Xie, Yaowei Wang, Jianbin Jiao, and Qixiang Ye, "Conformer: Local features coupling global representations for visual recognition," in *ICCV*, 2021, pp. 367–376.
[6] Yinpeng Chen, Xiyang Dai, Dongdong Chen, Mengchen Liu, Xiaoyi Dong, Lu Yuan, and Zicheng Liu, "Mobile-former: Bridging mobilenet and transformer," in *CVPR*, 2022, pp. 5270–5279.
[7] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He, "Non-local neural networks," in *CVPR*, 2018, pp. 7794–7803.
[8] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena, "Self-attention generative adversarial networks," in *ICML*. PMLR, 2019, pp. 7354–7363.
[9] Han Hu, Jiayuan Gu, Zheng Zhang, Jifeng Dai, and Yichen Wei, "Relation networks for object detection," in *CVPR*, 2018, pp. 3588–3597.
[10] Xuran Pan, Chunjiang Ge, Rui Lu, Shiji Song, Guanfu Chen, Zeyi Huang, and Gao Huang, "On the integration of self-attention and convolution," in *CVPR*, 2022, pp. 815–825.
[11] Haiping Wu, Bin Xiao, Noel Codella, Mengchen Liu, Xiyang Dai, Lu Yuan, and Lei Zhang, "CvT: Introducing convolutions to vision transformers," in *CVPR*, 2021, pp. 22–31.
[12] Tete Xiao, Mannat Singh, Eric Mintun, Trevor Darrell, Piotr Dollár, and Ross Girshick, "Early convolutions help transformers see better," *NeurIPS*, vol. 34, pp. 30392–30400, 2021.
[13] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," *arXiv preprint arXiv:2103.14030*, 2021.
[14] Namuk Park and Songkuk Kim, "How do vision transformers work?," *arXiv preprint arXiv:2202.06709*, 2022.
[15] Haixu Wu, Jialong Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long, "Flowformer: Linearizing transformers with conservation flows," *arXiv preprint arXiv:2202.06258*, 2022.
[16] Mingxing Tan and Quoc Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *ICML*. PMLR, 2019, pp. 6105–6114.
[17] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou, "Training data-efficient image transformers & distillation through attention," in *ICML*, 2021, pp. 10347–10357.
[18] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao, "Pyramid vision transformer: A versatile backbone for dense prediction without convolutions," in *ICCV*, 2021, pp. 568–578.
[19] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick, "Mask R-CNN," in *ICCV*, 2017, pp. 2961–2969.
[20] Zhaowei Cai and Nuno Vasconcelos, "Cascade R-CNN: Delving into high quality object detection," in *CVPR*, 2018, pp. 6154–6162.
[21] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," in *CVPR*, 2016, pp. 770–778.
[22] Zilong Huang, Youcheng Ben, Guozhong Luo, Pei Cheng, Gang Yu, and Bin Fu, "Shuffle transformer: Rethinking spatial shuffle for vision transformer," *arXiv preprint arXiv:2106.03650*, 2021.
[23] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba, "Scene parsing through ade20k dataset," in *CVPR*, 2017, pp. 633–641.
[24] Tete Xiao, Yingcheng Liu, Bolei Zhou, Yuning Jiang, and Jian Sun, "Unified perceptual parsing for scene understanding," in *ECCV*, 2018, pp. 418–434.
[25] Jianwei Yang, Chunyuan Li, Pengchuan Zhang, Xiyang Dai, Bin Xiao, Lu Yuan, and Jianfeng Gao, "Focal self-attention for local-global interactions in vision transformers," *arXiv preprint arXiv:2107.00641*, 2021.
[26] Xiangxiang Chu, Zhi Tian, Yuqing Wang, Bo Zhang, Haibing Ren, Xiaolin Wei, Huaxia Xia, and Chunhua Shen, "Twins: Revisiting the design of spatial attention in vision transformers," *NeurIPS*, vol. 34, pp. 9355–9366, 2021.