On the Integration of Self-Attention and Convolution
Supplementary Material

A. Model Architectures

We summarize the architectures of ResNet 26/38/50 [2], SAN 10/15/19 [9], PVT-T/S [7], Swin-T/S [5], and their respective ACmix version in Tab 2∼5. For fair comparison, we only substitute the original 3×3 convolution or self-attention module with our proposed operator in the modified models.

B. Dataset and Training Setup

ImageNet. ImageNet 2012 [1] comprises 1.28 million training images and 50,000 validation images from 1000 different classes. For ResNet-based models, we follow the training schedule in [9] and train all the models for 100 epochs. We use SGD with batchsize 256 on 8 GPUs. Cosine learning rate is adopted with the base learning rate set to 0.1. We apply standard data augmentation, including random cropping, random horizontal flipping and normalization. We use label smoothing with coefficient 0.1. For experiments on Transformer-based models, including PVT and Swin-Transformer, we follow training configurations in the original paper.

COCO. COCO dataset [4] is a standard object detection benchmark and we use a subset of 80k samples as training set and 35k for validation. For ResNet and SAN models, we train the network by SGD and 8 GPU are used with a batchsize of 16. For PVT and Swin-Transformer models, we train the network by adamw. Backbone networks are respectively pretrained on ImageNet dataset following the same training configurations in the original paper. We follow the "1x" learning schedule to train the whole network for 12 epochs and divide the learning rate by 10 at the 8th and 11th epoch respectively. For several transformer-based models, we follow the configurations in the original paper, and additionally experiment "3x" schedule with 36 epochs. We apply standard data augmentation, that is resize, random flip and normalize. Learning rate is set at 0.01 and linear warmup is used in the first 500 iterations. We follow the "1x" learning schedule training the whole network for 12 epochs and divide the learning rate by 10 at the 8th and 11th epoch respectively. For several transformer-based models, we follow the configurations in the original paper, and test with "3x" schedule. All mAP results in the main paper are tested with input image size (3, 1333, 800).

ADE20K. ADE20K [10] is a widely-used semantic segmentation dataset, containing 150 categories. ADE20K has 25K images, with 20K for training, 2K for validation, and another 3K for testing. For two baseline models, PVT and Swin-Transformer, we follow the training configurations in their original paper respectively. For PVT, we implement the backbone models on the basis of Semantic FPN [3]. We optimize the models using AdamW with an initial learning rate of 1e-4 for 80k iterations. For Swin-Transformer, we implement the backbone models on the basis of UperNet [8]. We use the AdamW optimizer with an initial learning rate of 6e-5 and a linear warmup of 1,500 iterations. Models are trained for a total of 160K iterations. We randomly resize and crop the image to 512 × 512 for training, and rescale to have a shorter side of 512 pixels during testing.

C. Hyper-parameters

For ResNet-ACmix models, we set $N = 4$ for all the experiments.

For SAN-ACmix models, the channel dimension for queries, keys and values are different in the original model [9]. Given input features with channel dimension C, queries and keys are projected to features with $C/4$ channels, while values are projected to features with C channels. Therefore, when implementing our ACmix operator, we divide values into 4 groups, where the divided groups have the same channel dimension $C/4$. The following self-attention and convolution operations follow the same *patchwise* attention in [9] and the same designing pipeline as we stated in Sec.4, respectively.

For PVT-ACmix and Swin-ACmix models, we follow the configurations in the original model [5].

$k_a = 7$ and $k_c = 3$ is set for all experiments, unless stated otherwise.

D. Positional Encoding

Positional encoding is widely adopted in self-attention modules, while not used in SAN and PVT models. Therefore, we follow this setting and only adopt positional encoding in the ResNet-ACmix models and Swin-ACmix. Specifically, the popular relative positional encoding [6] is adopted when computing the attention weights:

$$\mathrm{A}(q_{ij}, k_{ab}) = \underset{\mathcal{N}(i,j)}{\mathrm{softmax}}\left((q_{ij}^{\mathrm{T}} k_{ab} + B_{ij,ab})/\sqrt{d}\right), \quad (1)$$

where q, k, B represent queries, keys and relative positional encodings respectively. We didn't include positional encoding in the analysis for computation complexity in the Tab.1 of the main paper, as the *patchwise* attention proposed in [9] demonstrate the effectiveness of self-attention modules without adopting it. Nevertheless, the computation cost for positional encoding is also linear with respect to the channel dimension C, which is also comparably minor to the feature projection operations. Therefore, considering the positional encoding doesn't affect our main statement.

E. Practical Costs for Other Models

We also summarize the practical FLOPs and Parameters for convolution, self-attention and ACmix based on various models introduced in the Experiment section. The numbers are shown in Tab.1. Similar to ResNet 50, more than 60% of the computation are performed at Stage I of the self-attention module in SAN and Swin models. Meanwhile, it also demonstrates that ACmix only introduces minimum computational cost to integrate both convolution and self-attention modules based on various model structures.

Module	**Stage**	**ResNet 50 GFLOPs**	**SAN 19 GFLOPs**	**Swin-T GFLOPs**
Convolution	I	1.85 (99%)	-	-
	II	0.01 (1%)	-	-
Self-Attention	I	0.96 (83%)	1.29 (64%)	1.04 (68%)
	II	0.19 (17%)	0.72 (36%)	0.49 (32%)
ACmix	I	0.96 (73%)	1.29 (60%)	1.04 (62%)
	II	0.35 (27%)	0.89(40%)	0.64 (38%)

Table 1. Practical FLOPs and Parameters for different modules based on various models. Numbers within the brackets are their fractions of the whole module. SAN 19 and Swin-T models are designed with all self-attention modules, thus not applicable for the traditional convolution module.

References

[1] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 1

[2] 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

[3] Alexander Kirillov, Ross Girshick, Kaiming He, and Piotr Dollár. Panoptic feature pyramid networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6399–6408, 2019. 1

[4] 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 *ECCV*, 2014. 1

[5] 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. 1

[6] Prajit Ramachandran, Niki Parmar, Ashish Vaswani, Irwan Bello, Anselm Levskaya, and Jon Shlens. Stand-alone self-attention in vision models. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. 2

[7] 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. *arXiv preprint arXiv:2102.12122*, 2021. 1

[8] Tete Xiao, Yingcheng Liu, Bolei Zhou, Yuning Jiang, and Jian Sun. Unified perceptual parsing for scene understanding. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 418–434, 2018. 1

[9] Hengshuang Zhao, Jiaya Jia, and Vladlen Koltun. Exploring self-attention for image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10076–10085, 2020. 1, 2

[10] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ade20k dataset. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 633–641, 2017. 1

stage	output	ResNet-26 (**ACmix**)		ResNet-38 (**ACmix**)		ResNet-50 (**ACmix**)	
res1	112×112	7×7 conv, 64, stride 2		7×7 conv, 64, stride 2		7×7 conv, 64, stride 2	
res2	56×56	3×3 max pool, stride 2		3×3 max pool, stride 2		3×3 max pool, stride 2	
		$\begin{bmatrix} 1 \times 1 \text{ conv}, 64 \\ 3 \times 3 \text{ conv } (\textbf{ACmix}), 64 \\ 1 \times 1 \text{ conv}, 256 \end{bmatrix}$	$\times 1$	$\begin{bmatrix} 1 \times 1 \text{ conv}, 64 \\ 3 \times 3 \text{ conv } (\textbf{ACmix}), 64 \\ 1 \times 1 \text{ conv}, 256 \end{bmatrix}$	$\times 2$	$\begin{bmatrix} 1 \times 1 \text{ conv}, 64 \\ 3 \times 3 \text{ conv } (\textbf{ACmix}), 64 \\ 1 \times 1 \text{ conv}, 256 \end{bmatrix}$	$\times 3$
res3	28×28	$\begin{bmatrix} 1 \times 1 \text{ conv}, 128 \\ 3 \times 3 \text{ conv } (\textbf{ACmix}), 128 \\ 1 \times 1 \text{ conv}, 512 \end{bmatrix}$	$\times 2$	$\begin{bmatrix} 1 \times 1 \text{ conv}, 128 \\ 3 \times 3 \text{ conv } (\textbf{ACmix}), 128 \\ 1 \times 1 \text{ conv}, 512 \end{bmatrix}$	$\times 3$	$\begin{bmatrix} 1 \times 1 \text{ conv}, 128 \\ 3 \times 3 \text{ conv } (\textbf{ACmix}), 128 \\ 1 \times 1 \text{ conv}, 512 \end{bmatrix}$	$\times 4$
res4	14×14	$\begin{bmatrix} 1 \times 1 \text{ conv}, 256 \\ 3 \times 3 \text{ conv } (\textbf{ACmix}), 256 \\ 1 \times 1 \text{ conv}, 1024 \end{bmatrix}$	$\times 4$	$\begin{bmatrix} 1 \times 1 \text{ conv}, 256 \\ 3 \times 3 \text{ conv } (\textbf{ACmix}), 256 \\ 1 \times 1 \text{ conv}, 1024 \end{bmatrix}$	$\times 5$	$\begin{bmatrix} 1 \times 1 \text{ conv}, 256 \\ 3 \times 3 \text{ conv } (\textbf{ACmix}), 256 \\ 1 \times 1 \text{ conv}, 1024 \end{bmatrix}$	$\times 6$
res5	7×7	$\begin{bmatrix} 1 \times 1 \text{ conv}, 512 \\ 3 \times 3 \text{ conv } (\textbf{ACmix}), 512 \\ 1 \times 1 \text{ conv}, 2048 \end{bmatrix}$	$\times 1$	$\begin{bmatrix} 1 \times 1 \text{ conv}, 512 \\ 3 \times 3 \text{ conv } (\textbf{ACmix}), 512 \\ 1 \times 1 \text{ conv}, 2048 \end{bmatrix}$	$\times 2$	$\begin{bmatrix} 1 \times 1 \text{ conv}, 512 \\ 3 \times 3 \text{ conv } (\textbf{ACmix}), 512 \\ 1 \times 1 \text{ conv}, 2048 \end{bmatrix}$	$\times 3$
	1×1	global average pool 1000-d fc, softmax		global average pool 1000-d fc, softmax		global average pool 1000-d fc, softmax	

Table 2. Architectures of ResNet-based models with and without ACmix modules.

layers	output	SAN-10 (**ACmix**)	SAN-15 (**ACmix**)	SAN-19 (**ACmix**)
Input	224×224	1×1 conv, 64	1×1 conv, 64	1×1 conv, 64
Transition	112×112	2×2 max pool, stride 2 1×1 conv, 64	2×2 max pool, stride 2 1×1 conv, 64	2×2 max pool, stride 2 1×1 conv, 64
Block	112×112	$\begin{bmatrix} 3 \times 3 \text{ sa } (\textbf{ACmix}), 16 \\ 1 \times 1 \text{ conv}, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3 \text{ sa } (\textbf{ACmix}), 16 \\ 1 \times 1 \text{ conv}, 64 \end{bmatrix} \times 3$	$\begin{bmatrix} 3 \times 3 \text{ sa } (\textbf{ACmix}), 16 \\ 1 \times 1 \text{ conv}, 64 \end{bmatrix} \times 3$
Transition	56×56	2×2 max pool, stride 2 1×1 conv, 256	2×2 max pool, stride 2 1×1 conv, 256	2×2 max pool, stride 2 1×1 conv, 256
Block	56×56	$\begin{bmatrix} 7 \times 7 \text{ sa } (\textbf{ACmix}), 64 \\ 1 \times 1 \text{ conv}, 256 \end{bmatrix} \times 1$	$\begin{bmatrix} 7 \times 7 \text{ sa } (\textbf{ACmix}), 64 \\ 1 \times 1 \text{ conv}, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 7 \times 7 \text{ sa } (\textbf{ACmix}), 64 \\ 1 \times 1 \text{ conv}, 256 \end{bmatrix} \times 3$
Transition	28×28	2×2 max pool, stride 2 1×1 conv, 512	2×2 max pool, stride 2 1×1 conv, 512	2×2 max pool, stride 2 1×1 conv, 512
Block	28×28	$\begin{bmatrix} 7 \times 7 \text{ sa } (\textbf{ACmix}), 128 \\ 1 \times 1 \text{ conv}, 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 7 \times 7 \text{ sa } (\textbf{ACmix}), 128 \\ 1 \times 1 \text{ conv}, 512 \end{bmatrix} \times 3$	$\begin{bmatrix} 7 \times 7 \text{ sa } (\textbf{ACmix}), 128 \\ 1 \times 1 \text{ conv}, 512 \end{bmatrix} \times 4$
Transition	14×14	2×2 max pool, stride 2 1×1 conv, 1024	2×2 max pool, stride 2 1×1 conv, 1024	2×2 max pool, stride 2 1×1 conv, 1024
Block	14×14	$\begin{bmatrix} 7 \times 7 \text{ sa } (\textbf{ACmix}), 256 \\ 1 \times 1 \text{ conv}, 1024 \end{bmatrix} \times 4$	$\begin{bmatrix} 7 \times 7 \text{ sa } (\textbf{ACmix}), 256 \\ 1 \times 1 \text{ conv}, 1024 \end{bmatrix} \times 5$	$\begin{bmatrix} 7 \times 7 \text{ sa } (\textbf{ACmix}), 256 \\ 1 \times 1 \text{ conv}, 1024 \end{bmatrix} \times 6$
Transition	7×7	2×2 max pool, stride 2 1×1 conv, 2048	2×2 max pool, stride 2 1×1 conv, 2048	2×2 max pool, stride 2 1×1 conv, 2048
Block	7×7	$\begin{bmatrix} 7 \times 7 \text{ sa } (\textbf{ACmix}), 512 \\ 1 \times 1 \text{ conv}, 2048 \end{bmatrix} \times 1$	$\begin{bmatrix} 7 \times 7 \text{ sa } (\textbf{ACmix}), 512 \\ 1 \times 1 \text{ conv}, 2048 \end{bmatrix} \times 2$	$\begin{bmatrix} 7 \times 7 \text{ sa } (\textbf{ACmix}), 512 \\ 1 \times 1 \text{ conv}, 2048 \end{bmatrix} \times 3$
Classification	1×1	global average pool 1000-d fc, softmax	global average pool 1000-d fc, softmax	global average pool 1000-d fc, softmax

Table 3. Architectures of SAN-based models with and without ACmix modules.

stage	output	layer name	PVT-T (**ACmix**)		PVT-S (**ACmix**)	
res1	56×56	Patch Embedding	$P_1 = 4;\ C_1 = 64$		$P_1 = 4;\ C_1 = 64$	
		Transformer	$\begin{bmatrix} R_1 = 8 \\ N_1 = 1 \\ E_1 = 8 \end{bmatrix}$	$(\textbf{ACmix}) \times 2$	$\begin{bmatrix} R_1 = 8 \\ N_1 = 1 \\ E_1 = 8 \end{bmatrix}$	$(\textbf{ACmix}) \times 3$
res2	28×28	Patch Embedding	$P_2 = 2;\ C_2 = 128$		$P_2 = 2;\ C_2 = 128$	
		Transformer	$\begin{bmatrix} R_2 = 4 \\ N_2 = 2 \\ E_2 = 8 \end{bmatrix}$	$(\textbf{ACmix}) \times 2$	$\begin{bmatrix} R_2 = 4 \\ N_2 = 2 \\ E_2 = 8 \end{bmatrix}$	$(\textbf{ACmix}) \times 4$
res3	14×14	Patch Embedding	$P_3 = 2;\ C_3 = 320$		$P_3 = 2;\ C_3 = 320$	
		Transformer	$\begin{bmatrix} R_3 = 2 \\ N_3 = 5 \\ E_3 = 4 \end{bmatrix}$	$(\textbf{ACmix}) \times 2$	$\begin{bmatrix} R_3 = 2 \\ N_3 = 5 \\ E_3 = 4 \end{bmatrix}$	$(\textbf{ACmix}) \times 6$
res4	7×7	Patch Embedding	$P_4 = 2;\ C_4 = 512$		$P_4 = 2;\ C_4 = 512$	
		Transformer	$\begin{bmatrix} R_4 = 1 \\ N_4 = 8 \\ E_4 = 4 \end{bmatrix}$	$(\textbf{ACmix}) \times 2$	$\begin{bmatrix} R_4 = 1 \\ N_4 = 8 \\ E_4 = 4 \end{bmatrix}$	$(\textbf{ACmix}) \times 3$

Table 4. Architectures of PVT-based models with and without ACmix modules.

stage	output	Swin-T (**ACmix**)		Swin-S (**ACmix**)	
res1	56×56	concat 4×4, 96, LN		concat 4×4, 96, LN	
		$\begin{bmatrix} 7 \times 7 \text{ window} \\ \text{dim } 96, \text{head } 3 \end{bmatrix}$	$(\textbf{ACmix}) \times 2$	$\begin{bmatrix} 7 \times 7 \text{ window} \\ \text{dim } 96, \text{head } 3 \end{bmatrix}$	$(\textbf{ACmix}) \times 2$
res2	28×28	concat 2×2, 192, LN		concat 4×4, 192, LN	
		$\begin{bmatrix} 7 \times 7 \text{ window} \\ \text{dim } 192, \text{head } 6 \end{bmatrix}$	$(\textbf{ACmix}) \times 2$	$\begin{bmatrix} 7 \times 7 \text{ window} \\ \text{dim } 192, \text{head } 6 \end{bmatrix}$	$(\textbf{ACmix}) \times 2$
res3	14×14	concat 2×2, 384, LN		concat 4×4, 384, LN	
		$\begin{bmatrix} 7 \times 7 \text{ window} \\ \text{dim } 384, \text{head } 12 \end{bmatrix}$	$(\textbf{ACmix}) \times 6$	$\begin{bmatrix} 7 \times 7 \text{ window} \\ \text{dim } 384, \text{head } 12 \end{bmatrix}$	$(\textbf{ACmix}) \times 18$
res4	7×7	concat 2×2, 768, LN		concat 4×4, 768, LN	
		$\begin{bmatrix} 7 \times 7 \text{ window} \\ \text{dim } 768, \text{head } 24 \end{bmatrix}$	$(\textbf{ACmix}) \times 2$	$\begin{bmatrix} 7 \times 7 \text{ window} \\ \text{dim } 768, \text{head } 24 \end{bmatrix}$	$(\textbf{ACmix}) \times 2$

Table 5. Architectures of Swin-based models with and without ACmix modules.