MixFormer

Mixing Features across Windows and Dimensions

2022

论文出发点或背景:

局部窗口的自注意力机制虽然在视觉领域中表现不错，但是却有着感受野受限，建模能力较弱的短板。主要是因为它在非重叠的窗口上计算自注意力，并且在通道维度上共享权重

ViT虽然展现了transformer结构应用于视觉任务的潜力，但是在下游任务中仍然存在挑战，特别是高分辨率视觉任务的低效和捕获局部关系的低效。

像PVT和CvT 这样的工作，在全局自注意之前插入空间缩减或卷积，兼顾了自注意和卷积的优点。

论文创新思路:

1.将局部自注意力机制与深度卷积并行设计跨窗口连接以扩大感受野

2.跨分支的双向交互支路，实现通道和空间维度上的互补

由于在非重叠窗口内的自注意力机制由于没有交叉的窗口连接，会导致感受野受限制，解决这个问题的方法有shift,expand,shuffle或者卷积来进行跨窗口的连接，考虑到卷积经常被用来建模局部关系，我们使用了深度卷积作为一种很有用的连接方案

局部窗口自注意在跨维度共享权值的同时，动态计算空间维度的权值，导致信道维度建模能力薄弱的问题。

论文方法介绍

并行设计有两个方面的好处：首先，将局部窗口的自注意和跨分支的深度卷积结合起来，建立跨窗口的模型连接，解决了有限的接受域问题。其次，并行设计同时建模窗

口内关系和跨窗口关系，为特征跨分支交织提供了机会，实现了更好的特征表示学习。

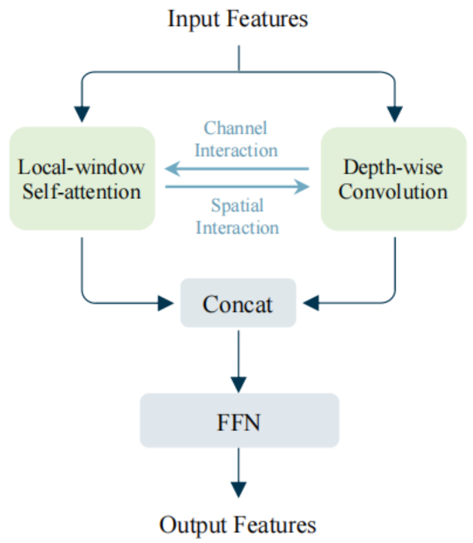


Figure 1. **The Mixing Block.** We combine local-window self-attention with depth-wise convolution in a parallel design. The captured relations within and across windows in parallel branches are concatenated and sent to the Feed-Forward Network (FFN) for output features. In the figure, the blue arrows marked with *Channel Interaction* and *Spatial Interaction* are the proposed bi-directional interactions, which provide complementary clues for better representation learning in both branches. Other details in the block, such as module design, normalization layers, and shortcuts, are omitted for a neat presentation.

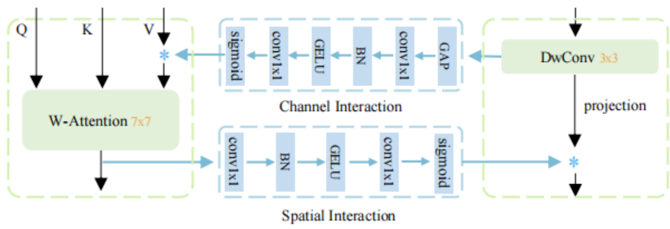
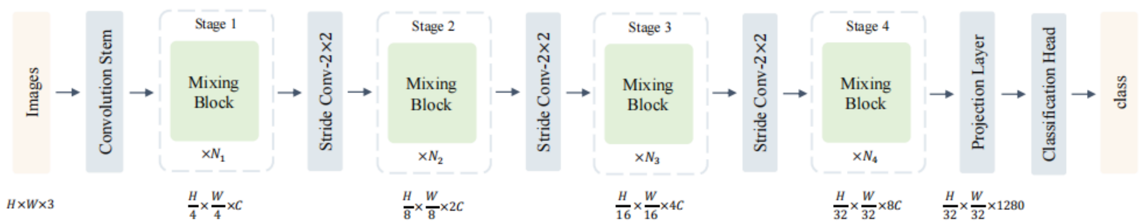


Figure 2. **Detailed design of the Bi-directional Interactions.** The channel/spatial interaction provides channel/spatial context extracted by depth-wise convolution/local-window self-attention to the other path.



虽然我们的通道交互与SE层[24]具有相同的设计，但它们在两个方面有不同：(1)attention模块的输入不同。我们的通道交互的输入来自另一个并行分支，而SE层

则在同一个分支中执行。(2)我们只将通道交互应用于局部窗口自注意中的值，而不是像SE层那样将其应用于模块的输出。

$$\hat{X}^{l+1} = \mathrm{MIX}(\mathrm{LN}(X^l), \mathrm{W\text{-}MSA}, \mathrm{CONV}) + X^l,$$
$$X^{l+1} = \mathrm{FFN}(\mathrm{LN}(\hat{X}^{l+1})) + \hat{X}^{l+1}$$

实际效果

	Attention	W-Attention	Conv	DwConv
Sharing Weights	Channel Dim	Channel Dim	Spatial Dim	Spatial Dim
FLOPs	$2NCH^2W^2$	$\mathbf{2NCHWK^2}$	NC^2HWK^2	$\mathbf{NCHWK^2}$

Table 1. **Sharing Weights Dimensions and FLOPs.** We provide comparison among four operations: global self-attention(Attention), local window self-attention(W-Attention), convolution(Conv) and depth-wise convolution(DwConv). In the table, we provide the dimension of weight sharing for all components in the first row. Besides, the FLOPs is calculated with a $N \times C \times H \times W$ input and a output with the same shape. The K in the table represents the window size in local-window self-attention or convolution. Note that the Attention operator adopts a window size of $H \times W$ as it models global dependencies in the spatial dimension.

Method	#Params	FLOPs	Top-1
ConvNets			
RegNetY-0.8G [40]	6M	0.8G	76.3
RegNetY-1.6G [40]	11M	1.6G	78.0
RegNetY-4G [40]	21M	4.0G	80.0
RegNetY-8G [40]	39M	8.0G	81.7
EffNet-B1 [43]	8M	0.7G	79.1
EffNet-B2 [43]	9M	1.0G	80.1
EffNet-B3 [43]	12M	1.8G	81.6
EffNet-B4 [43]	19M	4.2G	82.9
Vision Transformers			
DeiT-T [44]	6M	1.3G	72.2
DeiT-S [44]	22M	4.6G	79.9
DeiT-B [44]	87M	17.5G	81.8
PVT-T [49]	13M	1.8G	75.1
PVT-S [49]	25M	3.8G	79.8
PVT-M [49]	44M	6.7G	81.2
PVT-L [49]	61M	9.8G	81.7
CvT-13 [53]	20M	4.5G	81.6
CvT-21 [53]	32M	7.1G	82.5
TwinsP-S [6]	24M	3.8G	81.2
DS-Net-S [38]	23M	3.5G	82.3
Swin-T [34]	29M	4.5G	81.3
Swin-S [34]	50M	8.7G	83.0
Twins-S [6]	24M	2.9G	81.7
LG-T [31]	33M	4.8G	82.1
Focal-T [57]	29M	4.9G	82.2
Shuffle-T [26]	29M	4.6G	82.5
MixFormer-B1 (**Ours**)	8M	0.7G	78.9
MixFormer-B2 (**Ours**)	10M	0.9G	80.0
MixFormer-B3 (**Ours**)	17M	1.9G	81.7
MixFormer-B4 (**Ours**)	35M	3.6G	83.0

Table 3. **Classification accuracy on the ImageNet validation set.** Performances are measured with a single 224×224 crop. "Params" refers to the number of parameters. "FLOPs" is calculated under the input scale of 224×224.

Backbone	Method	#Params	FLOPs	mIoU$_{ss}$	mIoU$_{ms}$
ResNet-101 [19]	DANet [13]	69M	1119G	43.6	45.2
ResNet-101 [19]	DLab.v3+ [5]	63M	1021G	45.1	46.7
ResNet-101 [19]	ACNet [14]	-	-	45.9	-
ResNet-101 [19]	DNL [58]	69M	1249G	46.0	-
ResNet-101 [19]	OCRNet [60]	56M	923G	-	45.3
ResNet-101 [19]	UperNet [54]	86M	1029G	43.8	44.9
HRNet-w48 [47]	OCRNet [60]	71M	664G	-	45.7
DeiT-S [44]†	UperNet [54]	52M	1099G	44.0	-
TwinsP-S [6]	UperNet [54]	55M	919G	46.2	47.5
Swin-T [34]	UperNet [54]	60M	945G	44.5	45.8
Twins-S [6]	UperNet [54]	54M	901G	46.2	47.1
LG-T [31]	UperNet [54]	64M	957G	-	45.3
Focal-T [57]	UperNet [54]	62M	998G	45.8	47.0
Shuffle-T [26]	UperNet [54]	60M	949G	46.6	47.6
MixFormer-B1(**Ours**)	UperNet [54]	35M	854G	42.0	43.5
MixFormer-B2(**Ours**)	UperNet [54]	37M	859G	43.1	43.9
MixFormer-B3(**Ours**)	UperNet [54]	44M	880G	44.5	45.5
MixFormer-B4(**Ours**)	UperNet [54]	63M	918G	**46.8**	**48.0**

Backbones	#Params	FLOPs	AP^b	AP^b_{50}	AP^b_{75}	AP^m	AP^m_{50}	AP^m_{75}
ResNet50 [19]	82M	739G	46.3	64.3	50.5	40.1	61.7	43.4
Swin-T [34]	86M	745G	50.5	69.3	54.9	43.7	66.6	47.1
Shuffle-T [26]	86M	746G	50.8	69.6	55.1	44.1	66.9	48.0
MixFormer-B4(**Ours**)	91M	721G	**51.6**	**70.5**	**56.1**	**44.9**	**67.9**	**48.7**

Table 5. **COCO detection and segmentation with the Cascade Mask R-CNN.** The performances are reported on the COCO *val* split under a 3$\times$ schedule. Results show consistent improvements of MixFormer over Swin Transformer.

Backbones	COCO keypoint detection		
	AP^{kp}	AP^{kp}_{50}	AP^{kp}_{75}
ResNet50 [19]	71.8	89.8	79.5
Swin-T [34]	74.2	92.5	82.5
HRFormer-S [34]	74.5	92.3	82.1
MixFormer-B4(**Ours**)	**75.3** (+1.1)	**93.5** (+1.0)	**83.5** (+1.0)

Backbones	LVIS Instance Segmentation		
	AP^{mask}	AP^{mask}_{50}	AP^{mask}_{75}
ResNet50 [19]	21.7	34.3	23.0
Swin-T [34]	27.6	43.0	29.3
MixFormer-B4(**Ours**)	**28.6** (+1.0)	**43.4** (+0.4)	**30.5** (+1.2)

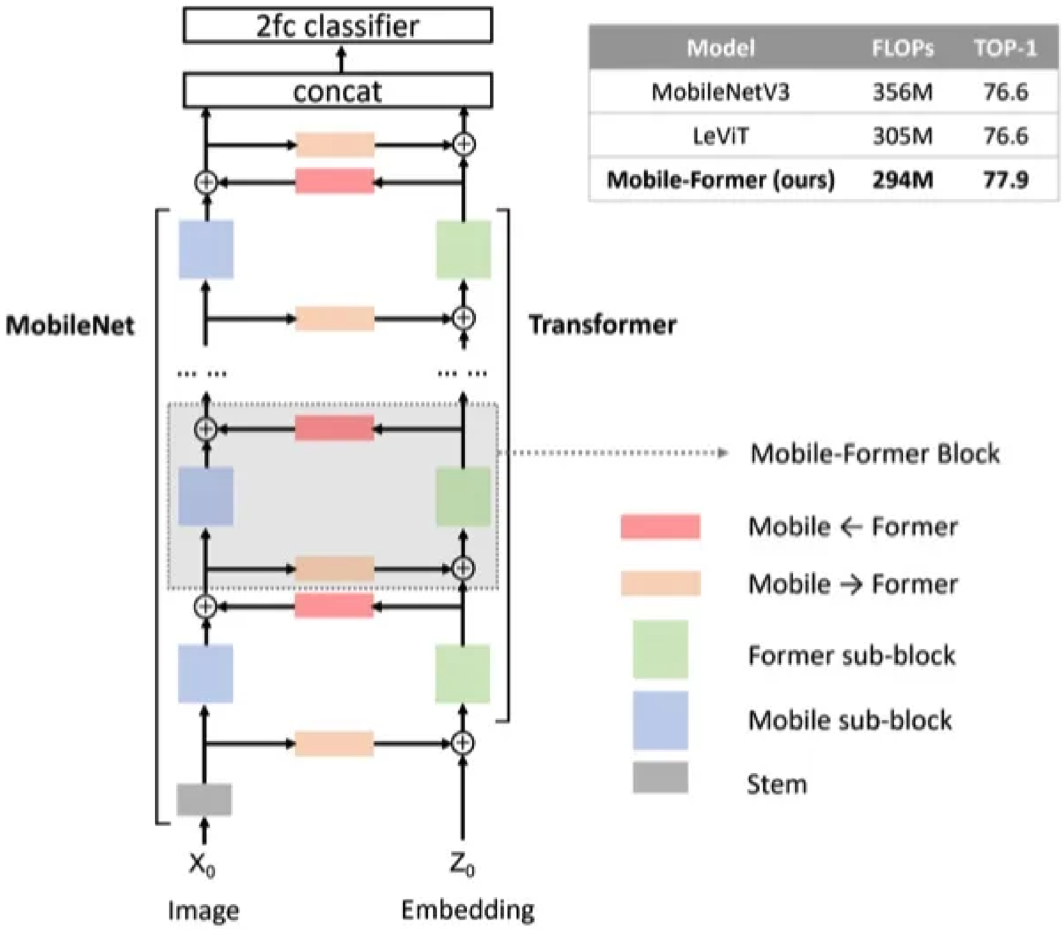
Table 11. **More Downstream Tasks.** We compare our MixFormer with ResNet50 [19] and Swin Transformer [34] on keypoint detection and long-tail instance segmentation.

Models	FLOPs	Top-1	Top-5
ResNet50 [44]	4.1G	78.4	-
ResNet50 [51]	4.1G	79.8	-
ResNet50*	4.1G	79.0	94.3
ResNet50 + Mixing Block	3.9G	**80.6** (+1.6)	**95.1** (+0.8)
MobileNetV2 [41]	0.3G	72.0	-
MobileNetV2*	0.3G	71.7	90.3
MobileNetV2+SE+Non-Local*	0.3G	72.5	91.0
MobileNetV2 + Mixing Block	0.3G	**73.6** (+1.9)	**91.6** (+1.3)

Table 12. **Apply Mixing Block to ConvNets on ImageNet-1K.** We introduce our Mixing Block to typical ConvNets, ResNet [19] and MobileNetV2 [41]. As different training recipes give variant accuracy [51], we also train ResNet50 [19] and MobileNetV2 [41] with the same setting as ours, denoted with $*$ in the Table.

个人理解

非常经典的一种思路，就是通过CNN的局部建模去弥补transformer的一些短板，之前看过一篇MobileFormer，也是设计的并行设计，中间通过双向桥实现局部特征和全局特征的双向融合，本文方法实现中和mobileformer不一样的就在于双向桥的实现，文章中提到自注意力机制欠缺通道之间的特征提取，而DW卷积得到的特征对于空间信息提取能力较弱，所以两者之间互补，通过sigmoid函数得到通道层面的权重，加权给transformer branch的channel。同时transformer branch的feature通过卷积之后经过sigmoid函数最终得到了HW方向上的权重（有点空间注意力机制的意思）。



Model	FLOPs	TOP-1
MobileNetV3	356M	76.6
LeViT	305M	76.6
Mobile-Former (ours)	**294M**	**77.9**

Figure 1. **Overview of Mobile-Former**, which parallelizes MobileNet [24] (on the left side) and Transformer [33] (on the right side). Different with vision transformer [8] that uses image patches to form tokens, the transformer in Mobile-Former takes *very few learnable tokens* as input that are randomly initialized. *Mobile* (refers to MobileNet) and *Former* (refers to transformer) are communicated by a bidirectional bridge, which is modeled by the proposed light-weight across attention. Best viewed in color.