MixFormer: Mixing Features across Windows and Dimensions

Qiang Chen[1*], Qiman Wu[1*], Jian Wang[1*], Qinghao Hu[2†], Tao Hu[1]
Errui Ding[1], Jian Cheng[2], Jingdong Wang[1]
[1]Baidu VIS
[2]NLPR, Institute of Automation, Chinese Academy of Sciences
{chenqiang13,wuqiman,wangjian33,hutao06,dingerrui,wangjingdong}@baidu.com
huqinghao2014@ia.ac.cn, jcheng@nlpr.ia.ac.cn

Abstract

While local-window self-attention performs notably in vision tasks, it suffers from limited receptive field and weak modeling capability issues. This is mainly because it performs self-attention within non-overlapped windows and shares weights on the channel dimension. We propose MixFormer to find a solution. First, we combine local-window self-attention with depth-wise convolution in a parallel design, modeling cross-window connections to enlarge the receptive fields. Second, we propose bi-directional interactions across branches to provide complementary clues in the channel and spatial dimensions. These two designs are integrated to achieve efficient feature mixing among windows and dimensions. Our MixFormer provides competitive results on image classification with EfficientNet and shows better results than RegNet and Swin Transformer. Performance in downstream tasks outperforms its alternatives by significant margins with less computational costs in 5 dense prediction tasks on MS COCO, ADE20k, and LVIS. Code is available at https://github.com/PaddlePaddle/PaddleClas.

1. Introduction

The success of Vision Transformer (ViT) [11, 44] in image classification [9] validates the potential to apply Transformer [46] to vision tasks. Challenges remain for downstream tasks, especially the inefficiency in high-resolution vision tasks and the ineffectiveness in capturing local relations. One possible solution is to use local-window self-attention. It performs self-attention within non-overlapped windows and shares weights on the channel dimension. Although this process improves efficiency, it poses the issues of limited receptive field and weak modeling capability.

*Equal Contribution.
†Corresponding author.

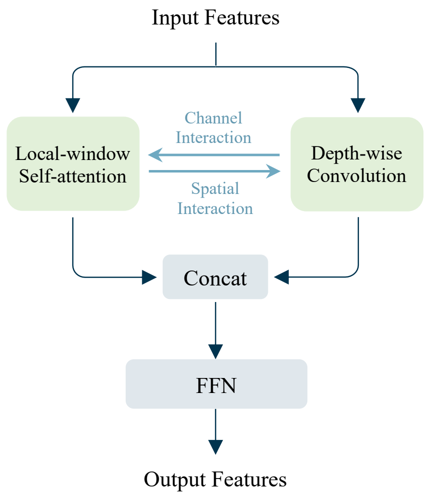

Figure 1. **The Mixing Block.** We combine local-window self-attention with depth-wise convolution in a parallel design. The captured relations within and across windows in parallel branches are concatenated and sent to the Feed-Forward Network (FFN) for output features. In the figure, the blue arrows 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.

A common approach to expand receptive field is to create cross-window connections. Windows are connected by shifting [34], expanding [45, 57], or shuffling [26] operations. Convolution layers are also employed as they capture natural local relations. Researches [26, 61] combine local-window self-attention with depth-wise convolution base on this and provide promising results. Still, the operations capture intra-window and cross-window relations in successive steps, leaving these two types of relations less interweaved. Besides, neglect of modeling weakness in these attempts hinders further advances in feature representation learning.

We propose Mixing Block to address both these issues (Figure 1). First, we combine local-window self-attention with depth-wise convolution, but in a parallel way. The parallel design enlarges the receptive fields by modeling intra-window and cross-window relations simultaneously. Second, we introduce bi-directional interactions across branches(illustrated as blue arrows in Figure 1). The interactions offset the limits caused by the weight sharing mechanism[1], and enhance the modeling ability in channel and spatial dimensions by providing complementary clues for local-window self-attention and depth-wise convolution respectively. The above designs are integrated to achieve complementary feature mixing across windows and dimensions.

We present MixFormer to verify the block's efficiency and effectiveness. A series of MixFormers with computational complexity ranging from 0.7G (B1) to 3.6G (B4) are built to perform distinguished in multiple vision tasks, including image classification, object detection, instance segmentation, semantic segmentation, etc. On ImageNet-1K [9], we achieve competitive results with EfficientNet [43], surpassing RegNet [40] and Swin Transformer [34] by a large margin. MixFormer markedly outperforms its alternatives in 5 dense prediction tasks with lower computational costs. With Mask R-CNN [18](1$\times$) on MS COCO [33], MixFormer-B4 shows a boost of 2.9 box mAP and 2.1 mask mAP on Swin-T [34] while requiring less computational cost. Substituting the backbone in UperNet [54], MixFormer-B4 delivers a 2.2 mIoU gain over Swin-T [34] on ADE20k [66]. Plus, MixFormer is effective in keypoint detection [33] and long-tail instance segmentation [15]. In brief, our MixFormer achieves state-of-the-art performance on multiple vision tasks as an efficient general-purpose vision transformer.

2. Related Works

Vision Transformers. The success of the pioneering work, ViT [11, 44], shows great potentials to apply transformer to the computer vision community. After that, various methods [2, 16, 29, 44, 59, 67] are proposed to improve the performance of vision transformers, demonstrating competitive results on the image classification task. As the self-attention [46] is different from the convolution in nature: self-attention models long-range dependencies while convolution captures relations in local windows, there are also works aiming for integrating convolution and vision transformer. Works like PVT [49] and CvT [53] insert spatial reduction or convolution before global self-attention, yielding the merits of self-attention and convolution.

[1]Local-window self-attention shares weights on the channel dimension while depth-wise convolution shares weights on the spatial one [17]. From the weight sharing perspective, sharing weights results in limited modeling ability in the correspond dimension.

Window-based Vision Transformers. Although global Vision Transformer shows its success on image classification; challenges remain for downstream tasks. For high-resolution vision tasks, the computation cost of the Vision Transformer is quadratic to image size, making it unaffordable for real-world applications. Recently, researchers have proposed plenty of methods [6, 10, 34, 49, 53, 56] to make vision transformers become general-purpose backbones as ConvNets [19, 22, 55]. Among them, Window-based Vision Transformer [6, 26, 34] adopts the local window attention mechanism, making its computational complexity increase linearly to image size.

Receptive Fields. Receptive fields are important for the downstream vision tasks. However, Window-based Vision Transformer computes self-attention within non-overlapping local windows, which limits the receptive fields in local windows. To solve the problem, researchers propose to use shifting [34], expanding [45, 57], or shuffling [26] operations to connect nearby windows.There are also works [26, 61] using convolutions to enlarge the receptive fields efficiently. Convolution layers are used to create connections because they capture local relations in nature. We combine local-window self-attention and depth-wise convolution in our block design.

Dynamic Mechanism. Dynamic networks here [8, 24, 28, 32, 46, 52] refer to networks whose parts of weights or paths are data-dependent. Generally speaking, the dynamic network achieves higher performance than its static alternative as it is more flexible in modeling relations. In ConvNets, the dynamic mechanism is widely used to better extract customized features given different inputs. There are various types of dynamic networks that focus on the channel [24, 32] and the spatial dimension [8, 28, 52]. These works promote many tasks to new state-of-the-art. For Transformer [46], the self-attention module is a dynamic component, which generates attention maps based on the inputs. In this paper, we also adopt the dynamic mechanism in the network design, while our application is based on the finding that the two efficient components share their weights on different dimensions [17]. To construct a powerful block while maintaining efficiency, we introduce dynamic interactions across two branches, which are light-weighted and improve the modeling ability in both channel and spatial dimensions.

3. Method

3.1. The Mixing Block

Our Mixing Block (Figure 1) adds two key designs upon the standard window-based attention block: (1) adopt a parallel design to combine local-window self-attention and depth-wise convolution, (2) introduce bi-directional interactions across branches. They are proposed to address the

	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.

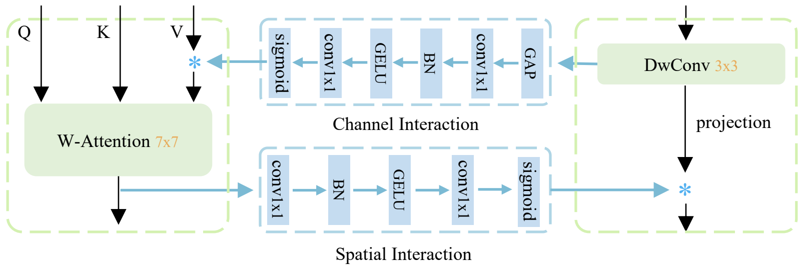

Figure 2. **Detailed design of the Bi-directional Interactions.** The channel/spatial interaction provides channel/spatial context extracted by depth-wise convolution/local-window self-attention to the other path.

limited receptive fields and weak modeling ability issues in local-window self-attention. We first present these two designs then integrate them to build the Mixing Block. Details are described next.

The Parallel Design. Although performing self-attention inside non-overlapped windows brings computational efficiency[2], it results in a limited receptive field due to no cross-window connections being extracted. Several methods resort to shift [34], expand [45, 57], shuffle [26], or convolution [26, 61] to model connections across windows. Considering that convolution layers are designed to model local relations, we choose the efficient alternative (depth-wise convolution) as a promising way to connect windows.

Attention then moves to adopt a proper way to combine local-window self-attention and depth-wise convolution. Previous methods [26, 34, 45, 57, 61] fill the goal by stacking these two operators successively. However, capturing intra-window and cross-window relations in successive steps make these two types of relations less interweaved.

In this paper, we propose a parallel design that enlarges the receptive fields by simultaneously modeling intra-window and cross-window relations. As illustrated in Figure 1, local-window self-attention and depth-wise convolution lie in two parallel paths. In detail, they use different window sizes. A 7×7 window is adopted in local-window self-attention, following previous works [23, 34, 45, 64]. While in depth-wise convolution, a smaller kernel size 3×3 is applied considering the efficiency[3]. Moreover, as their FLOPs are different, we adjust the number of channels according to the FLOPs proportion in Table 1. Then, their outputs are normalized by different normalization layers [1, 27] and merged by concatenation. The merged feature is sent to the successive Feed-Forward Network (FFN) to mix the learned relations across channels, generating the final output feature.

The parallel design benefits two-folds: First, combining local-window self-attention with depth-wise convolution across branches models connections across windows, addressing the limited receptive fields issue. Second, parallel design models intra-window and cross-window relations simultaneously, providing opportunities for feature interweaving across branches and achieving better feature representation learning.

Bi-directional Interactions. In general, sharing weights limits the modeling ability in the shared dimension. A common way to solve the dilemma is to generate data-dependent weights as done in dynamic networks [4, 24, 30, 52]. Local-window self-attention computes weights on the fly on the spatial dimension while sharing weights across channels, resulting in the weak modeling ability issue on the channel dimension. We focus on this issue in this subsection.

To enhance the modeling capacity of local-window self-attention on the channel dimension, we try to generate channel-wise dynamic weights [24]. Given that depth-wise convolution shares weights on the spatial dimension while focusing on the channel. It can provide complementary clues for local-window self-attention and vice versa. Thus, we propose *bi-directional interactions* (in Figure 1 and Figure 2) to enhance modeling ability in the channel and spatial dimension for local-window self-attention and depth-

[2]It has linear computational complexity concerning image size, as shown in Table 1.

[3]The results in Table 8 show that 3×3 is a good choice to achieve balance in accuracy and efficiency.

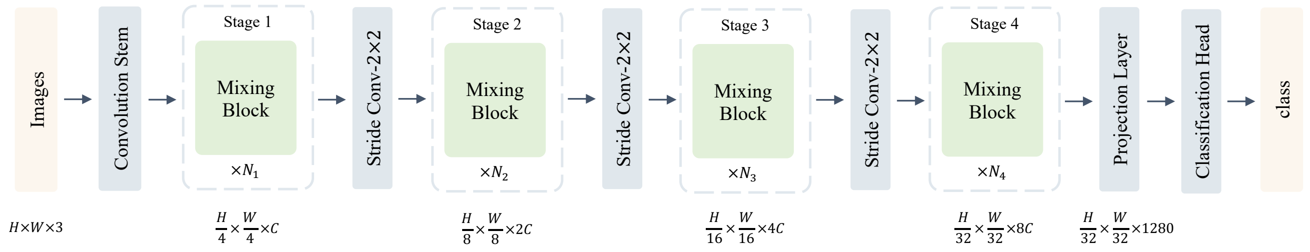

Figure 3. **Overall Architecture of MixFormer.** There are four parts in MixFormer: Convolution Stem, Stages, Projection Layer, and Classification Head. In Convolution Stem, we apply three successive convolutions to increase the channel from 3 to C. In Stages, we stack our Mixing Block in each stage and use stride convolution ($stride = 2$) to downsample the feature map. For Projection Layer, we use a linear layer with activation to increase the channels to 1280. The Classification Head is for the classification task.

wise convolution respectively. The bi-directional interactions consist of *the channel and spatial interaction* among the parallel branches. The information in the depth-wise convolution branch flows to the other branch through the channel interaction, which strengthens the modeling ability in the channel dimension. Meanwhile, the spatial interaction enables spatial relations to flow from the local-window self-attention branch to the other. As a result, the proposed bi-directional interactions provide complementary clues for each other. Next, we present the designs of the channel and spatial interactions in detail.

For the channel interaction, we follow the design of the SE layer [24], as shown in Figure 2. The channel interaction contains one global average pooling layer, followed by two successive 1×1 convolution layers with normalization (BN [27]) and activation (GELU [20]) between them. At last, we use sigmoid to generate attention in the channel dimension. Although our channel interaction shares the same design with the SE layer [24], they differ in two aspects: (1) The input of the attention module is different. The input of our channel interaction comes from another parallel branch, while the SE layer is performed in the same branch. (2) We only apply the channel interaction to the value in the local-window self-attention instead of applying it to the module's output as the SE layer does.

For the spatial interaction, we also adopt a simple design, which consists of two 1×1 convolution layers with followed BN [27] and GELU [20]. The detailed design is presented in Figure 2. These two layers reduce the number of channels to one. At last, a sigmoid layer is used to generate the spatial attention map. Same as we did in the channel interaction, the spatial attention is generated by another branch, where the local-window self-attention module is applied. It has a larger kernel size (7×7) than the depth-wise 3×3 convolution and focuses on the spatial dimension, which provides strong spatial clues for the depth-wise convolution branch.

The Mixing Block. Thanks to the above two designs, we mitigate two core issues in local-window self-attention. We integrate them to build a new transformer block, Mixing Block, upon the standard window attention block. As shown in Figure 1, the Mixing Block consists of two efficient operations in a parallel design, bi-directional interactions (Figure 2), and an FFN (Feed-Forward Networks) [46]. It can be formulated as follow:

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

$$X^{l+1} = \mathrm{FFN}(\mathrm{LN}(\hat{X}^{l+1})) + \hat{X}^{l+1} \quad (2)$$

Where MIX represents a function that achieves feature mixing between the W-MSA (Window-based Multi-Head Self-Attention) branch and the CONV (Depth-wise Convolution) branch. The MIX function first projects the input feature to parallel branches by two linear projection layers and two norm layers. Then it mixes the features by following the steps shown in Figure 1 and Figure 2. For FFN, we keep it simple and follow previous works [34,44], which is an MLP that consists of two linear layers with one GELU [20] between them. Moreover, we also try to add depth-wise convolution as done in PVTv2 [48] and HRFormer [61], which does not give many improvements over the MLP design (Table 9). Thus, to keep the block simple, we use MLP in FFN.

3.2. MixFormer

Overall Architecture. Based on the obtained block, we design an efficient and general-purpose vision transformer, MixFormer, with pyramid feature maps. There are four stages with downsampling rates of $\{4, 8, 16, 32\}$ respectively. MixFormer is a hybrid vision transformer, which uses convolution layers in both stem layers and downsampling layers. Besides, we introduce a projection layer in the tail of the stages. The projection layer increases the feature's channels to 1280 with a linear layer followed by an activation layer, aiming to preserve more details in the channel before the classification head. It gives a higher performance in classification, especially works well with smaller models. Same design can be found in previous efficient networks, such as MobileNets [22, 41] and EfficeintNets [43]. The sketch of our MixFormer is given in Figure 3.

Models	#Channels	#Blocks	#Heads
MixFormer-B1	$C = 32$	$[1, 2, 6, 6]$	$[2, 4, 8, 16]$
MixFormer-B2	$C = 32$	$[2, 2, 8, 8]$	$[2, 4, 8, 16]$
MixFormer-B3	$C = 48$	$[2, 2, 8, 6]$	$[3, 6, 12, 24]$
MixFormer-B4	$C = 64$	$[2, 2, 8, 8]$	$[4, 8, 16, 32]$

Table 2. **Architecture Variants.** Detailed configurations of architecture variants of MixFormer.

Architecture Variants. We stack the blocks in each stage manually and format several models in different sizes, whose computational complexities ranges from 0.7G (B1) to 3.6G (B4). The number of blocks in different stages is set by following a recipe: putting more blocks in the last two stages, which is roughly verified in Table 10. As shown in Table 2, we present the detailed settings of the models.

4. Experiments

We validate MixFormer on ImageNet-1K [9], MS COCO [33], and ADE20k [66]. We first present the accuracy on image classification. Then we do transfer learning to evaluate the models on three main tasks: object detection, instance segmentation, and semantic segmentation. Besides, ablations of different design modules in MixFormer and results with more vision tasks are provided.

4.1. Image Classification

Setup. We first verify our method by classification on ImageNet-1K [9]. To make a fair comparison with previous works [34, 44, 49], we train all models for 300 epochs with an image size of 224×224 and report Top-1 validation accuracy. We apply an AdamW optimizer using a cosine decay schedule. By following the rule that smaller models need less regularization, we adjust the training settings gently when training models in different sizes. Details are in Appendix C.

Results. Table 3 compares our MixFormer with efficient ConvNets [40, 43] and various Vision Transformers [26, 31, 34, 44, 49, 53, 57]. MixFormer performs on par with EfficientNet [43] and outperforms RegNet [40] by significant margins under various computational budgets (from B1 to B4). We note that it is *nontrivial* to achieve such results for vision transformer-based models, especially on small models (FLOPs $<$ 1.0G). Previous works such as DeiT [44] and PVT [49] show dramatic performance drops when reducing model complexities (-7.7% from DeiT-S to DeiT-T and -4.7% from PVT-S to PVT-T). Compared with Swin Transformer [34] and its variants [26, 31, 57], MixFormer shows better performance with less computational costs. In detail, MixFormer-B4 achieves 83.0% Top-1 accuracy with only 3.6G FLOPs. It outperforms Swin-T [34] by 1.7% while saving 20% computational costs and gives comparable results with Swin-S [34] but being $2.4\times$ efficient.

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.

The competitive advantage of MixFormer maintains when it comes to LG-Transformer [31], Focal Transformer [57] and Shuffle Transformer [26]. Moreover, our MixFormer also scales well to smaller and larger models. More results are provided in Appendix A.

4.2. Object Detection and Instance Segmentation

Setup. We validate the effectiveness of MixFormer on downstream tasks. We train Mask R-CNN [18] on the COCO2017 *train* split and evaluate the models on the *val* split. Two training schedules ($1\times$ and $3\times$) are adopted to show a consistent comparison with previous methods [19, 31, 34, 44, 57]. For the $1\times$ schedule, we train for 12 epochs with a single size (resizing the shorter side to 800 while keeping its longer side no more than 1333) [18]. While in $3\times$ schedule (36 epochs), we use multi-scale training by randomly resizing the shorter side to the range of $[480, 800]$ (See Appendix C for more details). Expect for Mask R-CNN [18], we also provide comparisons with pre-

Backbones	#Params	FLOPs	Mask R-CNN 1x schedule						Mask R-CNN 3x + MS schedule					
			AP^b	AP^b_{50}	AP^b_{75}	AP^m	AP^m_{50}	AP^m_{75}	AP^b	AP^b_{50}	AP^b_{75}	AP^m	AP^m_{50}	AP^m_{75}
ResNet18 [19]	31M	-	34.0	54.0	36.7	31.2	51.0	32.7	36.9	57.1	40.0	33.6	53.9	35.7
ResNet50 [19]	44M	260G	38.0	58.6	41.4	34.4	55.1	36.7	41.0	61.7	44.9	37.1	58.4	40.1
ResNet101 [19]	63M	336G	40.4	61.1	44.2	36.4	57.7	38.8	42.8	63.2	47.1	38.5	60.1	41.3
ResNeXt101-64×4d [55]	101M	493G	42.8	63.8	47.3	38.4	60.6	41.3	44.4	64.9	48.8	39.7	61.9	42.6
PVT-T [49]	33M	-	36.7	59.2	39.3	35.1	56.7	37.3	39.8	62.2	43.0	37.4	59.3	39.9
PVT-S [49]	44M	245G	40.4	62.9	43.8	37.8	60.1	40.3	43.0	65.3	46.9	39.9	62.5	42.8
PVT-M [49]	64M	302G	42.0	64.4	45.6	39.0	61.6	42.1	44.2	66.0	48.2	40.5	63.1	43.5
PVT-L [49]	81M	364G	42.9	65.0	46.6	39.5	61.9	42.5	44.5	66.0	48.3	40.7	63.4	43.7
TwinsP-S [6]	44M	245G	42.9	65.8	47.1	40.0	62.7	42.9	46.8	69.3	51.8	42.6	66.3	46.0
DS-Net-S [38]	43M	-	44.3	-	-	40.2	-	-	-	-	-	-	-	-
Swin-T [34]	48M	264G	42.2	64.6	46.2	39.1	61.6	42.0	46.0	68.2	50.2	41.6	65.1	44.8
Twins-S [6]	44M	228G	43.4	66.0	47.3	40.3	63.2	43.4	46.8	69.2	51.2	42.6	66.3	45.8
Focal-T [57]	49M	291G	-	-	-	-	-	-	47.2	69.4	51.9	42.7	66.5	45.9
Shuffle-T [26]	48M	268G	-	-	-	-	-	-	46.8	68.9	51.5	42.3	66.0	45.6
MixFormer-B1(**Ours**)	26M	183G	40.6	62.6	44.1	37.5	59.7	40.0	43.9	65.6	48.1	40.0	62.9	42.9
MixFormer-B2(**Ours**)	28M	187G	41.5	63.3	45.2	38.3	60.6	41.2	45.1	66.9	49.2	40.8	64.1	43.6
MixFormer-B3(**Ours**)	35M	207G	42.8	64.5	46.7	39.3	61.8	42.2	46.2	68.1	50.5	41.9	65.6	45.0
MixFormer-B4(**Ours**)	53M	243G	**45.1**	**67.1**	**49.2**	**41.2**	**64.3**	**44.1**	**47.6**	**69.5**	**52.2**	**43.0**	**66.7**	**46.4**

Table 4. **COCO detection and segmentation with the Mask R-CNN.** The performances are reported on the COCO *val* split under 1× and 3× schedules. The FLOPs (G) are measured at resolution 800 × 1280, and all models are pre-trained on the ImageNet-1K [9]. In the table, '-' means that the result is not reported by the original paper.

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× schedule. Results show consistent improvements of MixFormer over Swin Transformer.

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**

Table 6. **ADE20K semantic segmentation.** We report mIoU on the ADE20K [66] val split with single scale (ss) testing and multi-scale (ms) testing . A resolution 512 × 2048 is used to measure the FLOPs (G) in various models.

vious works based on a stronger model, Cascade Mask R-CNN [3, 18], where a 3× schedule is conducted.

Comparison on Mask R-CNN. Table 4 shows that MixFormer consistently outperforms other competitors [19, 31, 34, 49, 53, 57] under various model sizes with Mask R-CNN [18]. In particular, MixFormer-B4 achieves **+2.9**(**+1.6**) higher box mAP and **+2.1**(**+1.4**) higher mask mAP than the Swin-T [34] baseline with 1× (3×) schedule. Moreover, MixFormer keeps its efficiency in detection and instance segmentation, enabling higher performance with less computational costs than other networks [19, 34]. It is a surprise that our MixFormer-B1 (only with 0.7G) performs strongly with Mask R-CNN (1×), which exceeds ResNet-50 (with 4.1G) [19] by 2.3 box mAP and 2.9 mask mAP. The results suggest that implications for designing high-performance small models on detection are highlighted in MixFormer.

Comparison on Cascade Mask R-CNN. We also evaluate MixFormer with Cascade Mask R-CNN [3, 18], which is a stronger variant of Mask R-CNN [18]. MixFormer-B4 provides robust improvements over Swin-T [34] (Table 5) regardless of different detectors, as it shows similar gains (+1.1/1.2 box/mask mAP v.s. +1.6/1.4 box/mask mAP) with the ones on Mask R-CNN (3×) (Table 4).

4.3. Semantic Segmentation

Setup. Our experiments are conducted on ADE20K [66] using UperNet [54]. For training recipes, we mainly follow the settings in [34]. We report mIoU of our models in single scale testing (ss) and multi-scale testing (ms). Details are provided in Appendix C.

Results. In Table 6, MixFormer-B4 consistently achieves better mIoU performance than previous networks. It seems

Parallel	Interactions		ImageNet		COCO		ADE20k
	Channel	Spatial	Top-1	Top-5	AP^{box}	AP^{mask}	mIoU
			77.4	93.8	38.2	35.7	38.9
✓			78.1	94.1	39.4	36.6	39.8
✓	✓		78.3	94.1	40.1	37.1	40.6
✓		✓	78.3	94.1	39.7	36.6	40.5
✓	✓	✓	**78.4**	**94.3**	**40.3**	**37.3**	**40.9**
Δ			**+1.0**	**+0.5**	**+2.1**	**+1.6**	**+2.0**

Table 7. **Parallel Design with Bi-directional Interactions.** The baseline model in this table adopts a successive design and has no interactions in the block.

Window Sizes	ImageNet		COCO		ADE20k
	Top-1	Top-5	AP^{box}	AP^{mask}	mIoU
1×1	77.1	93.6	36.3	34.3	37.6
3×3	**78.4**	**94.3**	**40.3**	**37.3**	**40.9**
5×5	78.4	94.3	40.3	37.2	40.8

Table 8. **Window Sizes in DwConv.** We investigate various window sizes for DwConv. MixFormer uses the 3×3 window size for DwConv by default.

that the connections across windows and dimensions in the Mixing Block provide more benefits on semantic segmentation as the gains are larger than the ones on detection tasks (Table 4,Table 5) with the same backbones. In particular, MixFormer-B4 outperforms Swin-T [34] by **2.2** mIoU.

Moreover, other variants of MixFormer (from B1 to B3) also achieve higher performance while being more efficient than previous networks. Notably, MixFormer-B3 obtains 45.5 mIoU (comparable with Swin-T [34] but less FLOPs), which achieves on par results with OCRNet [60] with HRNet-W48 [47] (45.7 mIoU). Note that HRNet [47] is carefully designed to aggregate the features in different stages, while MixFormer simply constructs pyramid feature maps, indicating the strong potential for further improvements on dense prediction tasks.

4.4. Ablation Study

Setup. We provide ablations with respect to our designs on MixFormer-B1. We report all variations of different designs on ImageNet-1K [9] classification, COCO [33] detection and segmentation, and ADE20K [66] semantic segmentation. To make quick evaluations, we only train MixFormer-B1 for 200 epochs on ImageNet-1K [9]. Then, the pre-trained models are adopted by Mask R-CNN [18] (1×) on MS COCO [33] and UperNet [54] ($160k$) on ADE20K [66]. Note that, the differences in pre-train models provide slightly different results with the ones in Table 4 and Table 6.

Ablation: Parallel or Not. Table 7 provides the comparison of the ways (successive design or parallel design) to combine local-window self-attention and depth-wise convolution. Our parallel design consistently outperforms the successive design across various vision tasks, which verifies the hypothesis that the parallel design enables better feature representation learning in Section 1. The models below use parallel design by default.

Techniques	ImageNet		COCO		ADE20k
	Top-1	Top-5	AP^{box}	AP^{mask}	mIoU
MixFormer-B1(**Ours**)	78.4	94.3	40.3	37.3	40.9
+shifted windows	78.3	94.1	40.5	37.3	40.7
+DwConv in FFN	78.6	94.4	40.5	37.4	40.9

Table 9. **Other Techniques.** We combine two techniques with our MixFormer. When inserting DwConv in FFN, we only consider 3×3 DwConv.

#Blocks #Channels	FLOPs	ImageNet		COCO		ADE20k
		Top-1	Top-5	AP^{box}	AP^{mask}	mIoU
[2, 2, 8, 2] [32, 64, 160, 256]	0.9G	77.7	93.9	40.1	37.3	40.6
[2, 2, 6, 4] [32, 64, 128, 256]	0.9G	77.5	93.7	39.6	36.7	39.8
[1, 2, 6, 2] [32, 64, 160, 320]	0.8G	77.2	93.5	39.3	36.6	40.4
[1, 2, 6, 6] [32, 64, 128, 256]	**0.7G**	**78.4**	**94.3**	**40.3**	**37.3**	**40.9**

Table 10. **Number of Blocks in Stages.** In the table, the first two models and the last two models share similar computational complexities with each other.

Ablation: Bi-directional Interactions. Table 7 shows the results of the proposed interactions. According to the results, we see that both channel and spatial interactions outperform the model without interactions across all different vision tasks. Combining two interactions promotes better performance, resulting in consistent improvements by 0.3% Top-1 accuracy on ImageNet-1K, 0.9/0.7 box/mask mAP on COCO, and 1.1 mIoU on ADE20K. Given that we only use simple and light-weighted designs for bi-directional interactions, the gains are nontrivial, which indicates the effectiveness of providing complementary clues for local-window self-attention and depth-wise convolution.

Ablation: Window Sizes in DwConv. Table 8 shows that the performance will drop significantly on various vision tasks (-1.3 Top-1 accuracy on ImageNet-1K, $-4.0/-3.0$ box/mask mAP on COCO, and -3.3 mIoU on ADE20K) if we reduce the window size of the depth-wise convolution from 3×3 to 1×1. This phenomenon means that it's necessary for depth-wise convolution to use a window size (at least 3×3) with the ability to connect across-window. Besides, when we increase the window size to 5×5, no clear further gains are observed. Thus, we use a window size of 3×3 regarding the efficiency.

Ablation: Other Techniques. We also investigate other designs in MixFormer, including applying shifted windows

Backbones	COCO keypoint detection		
	AP^{kp}	AP^{kp}_{50}	AP^{kp}_{75}
ResNet50 [19]	71.8	94.9	49.2
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.

and inserting 3 × 3 depth-wise convolution in FFN, which play significant roles in previous works [34, 61]. As presented in Table 9, shifted window fails to provide gains over MixFormer. We hypothesize that the depth-wise convolution builds connections among windows, removing the need for shift operation. Besides, although inserting 3 × 3 depth-wise convolution in FFN can provide further gains, the room for improvements is limited with MixFormer. Thus, we use MLP in FFN by default.

Ablation: Number of Blocks in Stages. Previous works usually put more blocks in the third stage and greatly increase the number of blocks in that stage when scaling models [19, 34, 49]. We show an alternative way that can achieve the goal. We roughly conduct experiments on the way of stacking blocks. In Table 10, we achieve slightly higher performance on various vision tasks under less computational complexities by putting more blocks in both the last two stages. We follow this recipe to build our MixFormer.

4.5. Generalization

More Downstream Tasks. In Table 11, we conduct experiments on two more downstream tasks: keypoint detection and long-tail instance segmentation. Detailed experimental settings are provided in Appendix C.

COCO keypoint Detection: In Table 11, MixFormer-B4 outperforms baseline models [19, 34] by significant margins in all metrics. Moreover, MixFormer also shows clear advantages compared with HRFormer [61], which is specifically designed for dense prediction tasks.

LVIS 1.0 *Instance Segmentation:* This task has ∼ 1000 long-tailed distribution categories, which relies on the discriminative feature learned by the backbone. Results in Table 11 show that MixFormer outperforms the Swin-T [34] by 1.0 AP^{mask}, which demonstrates the robustness of the learned representation in MixFormer.

Summary: Considering the promising results given by MixFormer in previous tasks: object detection, instance segmentation, and semantic segmentation, MixFomer can serve as a general-purpose backbone and outperform its alternatives in **5** dense prediction tasks.

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.

Apply Mixing Block to ConvNets. We apply our Mixing Block to typical ConvNets, ResNet50 [19] and MobileNetV2 [41]. Following [42], we replace all the blocks in the last stage with our Mixing Block in ConvNets. To make a fair comparison, we adjust the number of blocks to maintain the overall computational cost. Table 12 shows that the Mixing Block can provide gains on ConvNets [19, 41] as a alternative to ConvNet blocks. Specifically, Mixing Block brings 1.9% and 1.6% Top-1 accuracy on ImageNet-1K [9] over MobileNetV2 [41] and ResNet50 [19]. Moreover, we also provide the result of MobileNetV2 [41] with SE layer [24] and Non-Local [50] in Table 12. It gives inferiror performance than our mixing block.

5. Limitations

Our MixFormer is proposed to mitigate the issues in local-window self-attention [34, 45]. Thus it may be limited to window-based vision transformers in this paper. Although the parallel design and the bi-directional interactions can be applied to the global self-attention [11, 44], it is not clear that how many gains can the above designs bring. We conduct a simple experiment on DeiT-Tiny [44]. But the result becomes slightly worse, as shown in Table 16. More efforts are needed to apply our mixing block to global attention. We leave this for future work. Moreover, we build the MixFormer series manually, limiting MixFormer in existing instances. Other methods such as NAS (Network Architecture Search) [43] can be applied to further improve the results.

6. Conclusion

In this paper, we propose MixFormer as an efficient general-purpose vision transformer. Addressing issues in Window-based Vision Transformer, we seek to alleviate

Models	#Channels	#Blocks	#Heads
MixFormer-B0	$C = 24$	$[1, 2, 6, 6]$	$[3, 6, 12, 24]$
MixFormer-B1	$C = 32$	$[1, 2, 6, 6]$	$[2, 4, 8, 16]$
MixFormer-B2	$C = 32$	$[2, 2, 8, 8]$	$[2, 4, 8, 16]$
MixFormer-B3	$C = 48$	$[2, 2, 8, 6]$	$[3, 6, 12, 24]$
MixFormer-B4	$C = 64$	$[2, 2, 8, 8]$	$[4, 8, 16, 32]$
MixFormer-B5	$C = 96$	$[1, 2, 8, 6]$	$[6, 12, 24, 48]$
MixFormer-B6	$C = 96$	$[2, 4, 16, 12]$	$[6, 12, 24, 48]$

Table 13. **Architecture Variants.** Detailed configurations of architecture variants of MixFormer.

limited receptive fields and weak modeling capability on the channel dimension. Our MixFormer enlarges receptive fields efficiently without shifting or shuffling windows, thanks to a parallel design coupling local window and depth-wise convolution. The bi-directional interactions boost modeling ability in the channel and spatial dimension for local-window self-attention and depth-wise convolution, respectively. Extensive experiments show that MixFormer outperforms its alternatives on image classification and various downstream vision tasks. We expect the designs in MixFormer to serve as a base setup for designing efficient networks.

Acknowledgements

The authors would like to thank Jiaying Guo, Zhe Li, and Fanrong Li for their helpful discussions and feedback. This work was supported in part by National Natural Science Foundation of China (No.62106267)

Appendix

A. More Variants of MixFormer

We scale our MixFormer to smaller and larger models. In this section, we provide two instantiated models (MixFormer-B0 and MixFormer-B5). Their detailed settings are provided in Table 13, along with previous methods (from B1 to B4). Note that MixFormer-B0 and MixFormer-B5 are two examples. More variants can be obtained with further attempts following the design of MixFormer. Then, we validate their effectiveness on ImageNet-1K [9]. The results are illustrated in Table 14.

On one side, MixFormer-B0 achieves competitive result (76.5% Top-1 accuracy on ImageNet-1K [9]) even with 0.4G FLOPs, which lies in the mobile level [37, 41]. While other vision transformer variants [34, 48, 49, 53, 57] did not provide a range of model sizes like our MixFormer, especially in mobile level. We believe that further efforts can be made to give higher performance to achieve state-of-the-art results [21, 43] in mobile level models. On the other side, MixFormer-B5 shows an example to scale our MixFormer to larger models. It has 6.8G FLOPs, while it can achieve on par results with Swin-B (15.4G) [34], Focal-S (9.1G) [57], Shuffle-S (8.9G) [26], and EfficientNet-B5 (9.9G) [43], which demonstrates the computational efficiency of MixFormer. MixFormer-B6 achieves **83.8%** top-1 accuracy on ImageNet-1K [9]. It maintains the superior performance to Swin-B(15.4G) [34] and is comparable to other models with less flops.

The above results verify the scalability of MixFormer to smaller and larger models. Moreover, it has the potential for further improvements.

Method	#Params	FLOPs	Top-1
ConvNets			
RegNetY-4G [40]	21M	4.0G	80.0
RegNetY-8G [40]	39M	8.0G	81.7
RegNetY-16G [40]	84M	16.0G	82.9
EffNet-B0 [43]	5M	0.4G	77.1
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
EffNet-B5 [43]	30M	9.9G	83.6
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
Swin-B [34]	88M	15.4G	83.5
Twins-S [6]	24M	2.9G	81.7
Twins-B [6]	56M	8.6G	83.2
LG-T [31]	33M	4.8G	82.1
LG-S [31]	61M	9.4G	83.3
Focal-T [57]	29M	4.9G	82.2
Focal-S [57]	51M	9.1G	83.5
Shuffle-T [26]	29M	4.6G	82.5
Shuffle-S [26]	50M	8.9G	83.5
MixFormer-B0 (**Ours**)	5M	0.4G	76.5
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
MixFormer-B5 (**Ours**)	62M	6.8G	83.5
MixFormer-B6 (**Ours**)	119M	12.7G	83.8

Table 14. **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.

B. Additional Experiments

Window Sizes in Local-Window Self-Attention. We conduct ablation study on the window size in local-window

Window Sizes	ImageNet	
	Top-1	Top-5
7 × 7	78.4	94.3
12 × 12	78.4	94.5

Table 15. **Window Sizes in Local-window Self-attention.** We investigate various window sizes for Local-window Self-attention in MixFormer.

DeiT-Tiny [44]	ImageNet	
	Top-1	Top-5
Baseline	72.2	91.1
Baseline+Mixing Block	71.3	90.5

Table 16. **Apply Mixing Block to DeiT-Tiny.** We apply our mixing block to global attention.

self-attention with MixFormer-B1. The experimental settings are follow the ones in the ablation studies. The results in Table 15 show that larger window size (ws=12) achieves on par performance with ws=7 (78.4%) on ImageNet-1K [9]. Based on the above result, We follow the con- ventional design of Swin Transformer (ws=7) [34] in all variants of MixFormer.

Apply Mixing Block to DeiT. Although our mixing block is proposed to solve the window connection problem in local-window self-attention [34]. It can also be applied to global attentions [11, 44]. We simply apply our mixing block to Deit-Tiny [44]. But the result is slightly lower than baseline (71.3% vs. 72.2%) on ImageNet-1K [9]. We conjecture that global attention (ViT-based model) may not share the same problem and detailed design for global attention may need further investigating. We leave this for future work.

C. Detailed Experimental Settings

Successive Design and Parallel Design. In Figure 4, we give the details on how to combine local-window self-attention and depth-wise convolution in the successive design and the parallel design. To make a fair comparison, we adjust the channels in the blocks to keep the computational complexity the same in the two designs.

Image Classification. We train all models for 300 epochs with an image size of 224 × 224 on ImageNet-1K [9]. We adjust the training settings gently when training models in different sizes. The detailed setting is in Table 17.

Object Detection and Instance Segmentation. When transferring MixFormer to object detection and instance segmentation on MS COCO [33], we consider two typical frameworks: Mask R-CNN [18] and Cascade Mask R-CNN [3, 18]. We adopt AdamW [36] optimizer with an initial learning rate of 0.0002 and a batch size of 16. To make a fair comparison with other works, we make all normalization layers trainable in MixFormer[4]. When training different sizes of models, we adjust the training settings gently according to their settings used in image classification. Table 18 shows the detailed hyper-parameters used in training models on MS COCO [33].

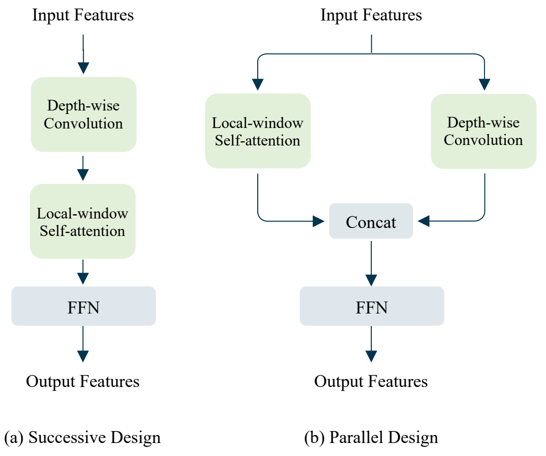

Figure 4. **Successive Design and Parallel Design.** We combine local-window self-attention with depth-wise convolution in two different ways. Other details in the block, such as module design, normalization layers, and shortcuts, are omitted for a neat presentation.

config	value
optimizer	AdamW [36]
base learning rate	8e-4 (B0-B3), 1e-3 (B4, B5, B6)
weight decay	0.04 (B0-B3), 0.05 (B4, B5, B6)
optimizer momentum	β_1, β_2=0.9, 0.999
batch size	1024
learning rate schedule	cosine decay [35]
minimum learning rate	1e-6
warmup epochs	20 (B0-B4), 40 (B5, B6)
warmup learning rate	1e-7
training epochs	300
augmentation	RandAug(9, 0.5) [7]
color jitter	0.4
mixup [63]	0.2
cutmix [62]	1.0
random erasing [65]	0.25
drop path [25]	[0.0, 0.05, 0.1, 0.2, 0.3, 0.5] (B0-B6)

Table 17. **Image Classification Training Settings.**

config	value
optimizer	AdamW
base learning rate	0.0002
weight decay	0.04 (B0-B3), 0.05 (B4, B5)
optimizer momentum	β_1, β_2=0.9, 0.999
batch size	16
learning rate schedule	steps:[8, 11] (1×), [27, 33] (3×)
warmup iterations (ratio)	500 (0.001)
training epochs	12 (1×), 36 (3×)
scales	(800, 1333) (1×), Multi-scales [34] (3×)
drop path	0.0 (B0-B3), 0.1 (B4, B5)

Table 18. **Object Detection and Instance Segmentation Training Settings.**

[4]Wherever BN is applied, we use *synchronous* BN across all GPUs.

Semantic Segmentation. On ADE20K [66], we use the AdamW optimizer [36] with an initial learning rate 0.00006, a weight decay 0.01, and a batch size of 16. We train all models for 160K on ADE20K. For testing, we report the results with single-scale testing and multi-scale testing on main comparisons, while we only give single-scale testing results on ablation studies. In multi-scale testing, the resolutions used are the $[0.5, 0.75, 1.0, 1.25, 1.5, 1.75]\times$ of that in training. The settings mainly follow [34]. For the path drop rates in different models, we adopt the same hyper-parameters as in MS COCO [33].

Keypoint Detection. We conduct experiments on the MS COCO human pose estimation benchmark. We train the models for 210 epochs with an AdamW optimizer, an image size of 256×192, and a batch size of 256. The training and evaluation hyper-parameters are mostly following the ones in HRFormer [61].

Long-tail Instance Segmentation. We use the hyper-parameters of Mask R-CNN [18] on MS COCO [33] when training models for long-tail instance segmentation on LVIS [15]. We report the results with a $1\times$ schedule. The training augmentations and sampling methods are the same for all models, which adopt a multi-scale training and use balanced sampling by following [15].

D. Discussion with Related Works

In MixFormer, we consider two types of information exchanges: (1) across dimensions, (2) across windows.

For the first type, Conformer [39] also performs information exchange between a transformer branch and a convolution branch. While its motivation is different from ours. Conformer aims to couple local and global features across convolution and transformer branches. MixFormer uses channel and spatial interactions to address the weak modeling ability issues caused by weight sharing on the channel (local-window self-attention) and the spatial (depth-wise convolution) dimensions [17].

For the second type, Twins (strided convolution + global sub-sampled attention) [6] and Shuffle Transformer (neighbor-window connection (NWC) + random spatial shuffle) [26] construct local and global connections to achieve information exchanges, MSG Transformer (channel shuffle on extra MSG tokens) [12] applies global connection. Our MixFormer achieves this goal by concatenating the parallel features: the non-overlapped window feature and the local-connected feature (output of the dwconv3x3).

References

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. 3

[2] Song Bai, Philip Torr, et al. Visual parser: Representing part-whole hierarchies with transformers. *arXiv preprint arXiv:2107.05790*, 2021. 2

[3] Zhaowei Cai and Nuno Vasconcelos. Cascade r-cnn: Delving into high quality object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6154–6162, 2018. 6, 10

[4] Jin Chen, Xijun Wang, Zichao Guo, Xiangyu Zhang, and Jian Sun. Dynamic region-aware convolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8064–8073, 2021. 3

[5] Liang-Chieh CHEN, Yukun ZHU, George PAPANDREOU, F Schroff, CV Aug, and H Adam. Deeplabv3+: Encoder-decoder with atrous separable convolution for semantic image segmentation [m]. *FERRARI V, HEBERT M, SMINCHISESCU C, et al. ECCV (7). Springer*, pages 833–851, 2018. 6

[6] 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. *arXiv preprint arXiv:2104.13840*, 1(2):3, 2021. 2, 5, 6, 9, 11

[7] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 702–703, 2020. 10

[8] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 764–773, 2017. 2

[9] 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, 5, 6, 7, 8, 9, 10

[10] Xiaoyi Dong, Jianmin Bao, Dongdong Chen, Weiming Zhang, Nenghai Yu, Lu Yuan, Dong Chen, and Baining Guo. Cswin transformer: A general vision transformer backbone with cross-shaped windows. *arXiv preprint arXiv:2107.00652*, 2021. 2

[11] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 1, 2, 8, 10

[12] Jiemin Fang, Lingxi Xie, Xinggang Wang, Xiaopeng Zhang, Wenyu Liu, and Qi Tian. Msg-transformer: Exchanging local spatial information by manipulating messenger tokens. *arXiv preprint arXiv:2105.15168*, 2021. 11

[13] Jun Fu, Jing Liu, Haijie Tian, Yong Li, Yongjun Bao, Zhiwei Fang, and Hanqing Lu. Dual attention network for scene segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3146–3154, 2019. 6

[14] Jun Fu, Jing Liu, Yuhang Wang, Yong Li, Yongjun Bao, Jinhui Tang, and Hanqing Lu. Adaptive context network for

scene parsing. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6748–6757, 2019. 6

[15] Agrim Gupta, Piotr Dollar, and Ross Girshick. Lvis: A dataset for large vocabulary instance segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5356–5364, 2019. 2, 11

[16] Kai Han, An Xiao, Enhua Wu, Jianyuan Guo, Chunjing Xu, and Yunhe Wang. Transformer in transformer. *arXiv preprint arXiv:2103.00112*, 2021. 2

[17] Qi Han, Zejia Fan, Qi Dai, Lei Sun, Ming-Ming Cheng, Jiaying Liu, and Jingdong Wang. Demystifying local vision transformer: Sparse connectivity, weight sharing, and dynamic weight. *arXiv preprint arXiv:2106.04263*, 2021. 2, 11

[18] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017. 2, 5, 6, 7, 10, 11

[19] 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. 2, 5, 6, 8

[20] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016. 4

[21] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. Searching for mobilenetv3. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1314–1324, 2019. 9

[22] 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. 2, 4

[23] Han Hu, Zheng Zhang, Zhenda Xie, and Stephen Lin. Local relation networks for image recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3464–3473, 2019. 3

[24] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018. 2, 3, 4, 8

[25] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger. Deep networks with stochastic depth. In *European conference on computer vision*, pages 646–661. Springer, 2016. 10

[26] 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. 1, 2, 3, 5, 6, 9, 11

[27] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015. 3, 4

[28] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. *Advances in neural information processing systems*, 28:2017–2025, 2015. 2

[29] Zihang Jiang, Qibin Hou, Li Yuan, Daquan Zhou, Xiaojie Jin, Anran Wang, and Jiashi Feng. Token labeling: Training a 85.5% top-1 accuracy vision transformer with 56m parameters on imagenet. *arXiv preprint arXiv:2104.10858*, 2021. 2

[30] Duo Li, Jie Hu, Changhu Wang, Xiangtai Li, Qi She, Lei Zhu, Tong Zhang, and Qifeng Chen. Involution: Inverting the inherence of convolution for visual recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12321–12330, 2021. 3

[31] Jinpeng Li, Yichao Yan, Shengcai Liao, Xiaokang Yang, and Ling Shao. Local-to-global self-attention in vision transformers. *arXiv preprint arXiv:2107.04735*, 2021. 5, 6, 9

[32] 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. 2

[33] 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. Springer, 2014. 2, 5, 7, 10, 11

[34] 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, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11

[35] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016. 10

[36] I. Loshchilov and F. Hutter. Fixing weight decay regularization in adam. 2017. 10, 11

[37] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *Proceedings of the European conference on computer vision (ECCV)*, pages 116–131, 2018. 9

[38] Mingyuan Mao, Renrui Zhang, Honghui Zheng, Peng Gao, Teli Ma, Yan Peng, Errui Ding, and Shumin Han. Dual-stream network for visual recognition. *arXiv preprint arXiv:2105.14734*, 2021. 5, 6, 9

[39] 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 *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 367–376, 2021. 11

[40] Ilija Radosavovic, Raj Prateek Kosaraju, Ross Girshick, Kaiming He, and Piotr Dollár. Designing network design spaces. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10428–10436, 2020. 2, 5, 9

[41] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018. 4, 8, 9

[42] Aravind Srinivas, Tsung-Yi Lin, Niki Parmar, Jonathon Shlens, Pieter Abbeel, and Ashish Vaswani. Bottleneck

transformers for visual recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16519–16529, 2021. 8

[43] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*, pages 6105–6114. PMLR, 2019. 2, 4, 5, 8, 9

[44] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International Conference on Machine Learning*, pages 10347–10357. PMLR, 2021. 1, 2, 4, 5, 6, 8, 9, 10

[45] Ashish Vaswani, Prajit Ramachandran, Aravind Srinivas, Niki Parmar, Blake Hechtman, and Jonathon Shlens. Scaling local self-attention for parameter efficient visual backbones. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12894–12904, 2021. 1, 2, 3, 8

[46] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017. 1, 2, 4

[47] Jingdong Wang, Ke Sun, Tianheng Cheng, Borui Jiang, Chaorui Deng, Yang Zhao, Dong Liu, Yadong Mu, Mingkui Tan, Xinggang Wang, et al. Deep high-resolution representation learning for visual recognition. *IEEE transactions on pattern analysis and machine intelligence*, 2020. 6, 7

[48] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pvtv2: Improved baselines with pyramid vision transformer. *arXiv preprint arXiv:2106.13797*, 2021. 4, 9

[49] 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. 2, 5, 6, 8, 9

[50] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7794–7803, 2018. 8

[51] Ross Wightman, Hugo Touvron, and Hervé Jégou. Resnet strikes back: An improved training procedure in timm. *arXiv preprint arXiv:2110.00476*, 2021. 8

[52] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. Cbam: Convolutional block attention module. In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19, 2018. 2, 3

[53] Haiping Wu, Bin Xiao, Noel Codella, Mengchen Liu, Xiyang Dai, Lu Yuan, and Lei Zhang. Cvt: Introducing convolutions to vision transformers. *arXiv preprint arXiv:2103.15808*, 2021. 2, 5, 6, 9

[54] 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. 2, 6, 7

[55] 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. 2, 6

[56] Haotian Yan, Zhe Li, Weijian Li, Changhu Wang, Ming Wu, and Chuang Zhang. Contnet: Why not use convolution and transformer at the same time? *arXiv preprint arXiv:2104.13497*, 2021. 2

[57] 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. 1, 2, 3, 5, 6, 9

[58] Minghao Yin, Zhuliang Yao, Yue Cao, Xiu Li, Zheng Zhang, Stephen Lin, and Han Hu. Disentangled non-local neural networks. In *European Conference on Computer Vision*, pages 191–207. Springer, 2020. 6

[59] Li Yuan, Yunpeng Chen, Tao Wang, Weihao Yu, Yujun Shi, Zihang Jiang, Francis EH Tay, Jiashi Feng, and Shuicheng Yan. Tokens-to-token vit: Training vision transformers from scratch on imagenet. *arXiv preprint arXiv:2101.11986*, 2021. 2

[60] Yuhui Yuan, Xilin Chen, and Jingdong Wang. Object-contextual representations for semantic segmentation. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part VI 16*, pages 173–190. Springer, 2020. 6, 7

[61] Yuhui Yuan, Rao Fu, Lang Huang, Weihong Lin, Chao Zhang, Xilin Chen, and Jingdong Wang. Hrformer: High-resolution transformer for dense prediction. *Advances in Neural Information Processing Systems*, 2021. 1, 2, 3, 4, 8, 11

[62] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6023–6032, 2019. 10

[63] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017. 10

[64] 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. 3

[65] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, number 07, pages 13001–13008, 2020. 10

[66] Bolei Zhou, Hang Zhao, Xavier Puig, Tete Xiao, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Semantic understanding of scenes through the ade20k dataset. *International Journal of Computer Vision*, 127(3):302–321, 2019. 2, 5, 6, 7, 11

[67] Daquan Zhou, Bingyi Kang, Xiaojie Jin, Linjie Yang, Xiaochen Lian, Zihang Jiang, Qibin Hou, and Jiashi Feng. Deepvit: Towards deeper vision transformer. *arXiv preprint arXiv:2103.11886*, 2021. 2