CSWin Transformer: A General Vision Transformer Backbone with Cross-Shaped Windows

Xiaoyi Dong[1]*, **Jianmin Bao**[2], **Dongdong Chen**[3], **Weiming Zhang**[1], **Nenghai Yu**[1], **Lu Yuan**[3], **Dong Chen**[2], **Baining Guo**[2]
[1]University of Science and Technology of China
[2]Microsoft Research Asia [3]Microsoft Cloud + AI
{dlight@mail., zhangwm@, ynh@}.ustc.edu.cn cddlyf@gmail.com
{jianbao, luyuan, doch, bainguo }@microsoft.com

Abstract

We present CSWin Transformer, an efficient and effective Transformer-based backbone for general-purpose vision tasks. A challenging issue in Transformer design is that global self-attention is very expensive to compute whereas local self-attention often limits the field of interactions of each token. To address this issue, we develop the ***Cross-S****haped* ***Win****dow* self-attention mechanism for computing self-attention in the horizontal and vertical stripes in *parallel* that form a *cross-shaped* window, with each stripe obtained by splitting the input feature into stripes of equal width. We provide a detailed mathematical analysis of the effect of the stripe width and vary the stripe width for different layers of the Transformer network which achieves strong modeling capability while limiting the computation cost. We also introduce *Locally-enhanced Positional Encoding* (LePE), which handles the local positional information better than existing encoding schemes. LePE naturally supports arbitrary input resolutions, and is thus especially effective and friendly for downstream tasks. Incorporated with these designs and a hierarchical structure, CSWin Transformer demonstrates competitive performance on common vision tasks. Specifically, it achieves **85.4%** Top-1 accuracy on ImageNet-1K without any extra training data or label, **53.9** box AP and **46.4** mask AP on the COCO detection task, and **51.7** mIOU on the ADE20K semantic segmentation task, surpassing previous state-of-the-art Swin Transformer backbone by **+1.2**, **+2.0**, **+1.4**, and **+2.0** respectively under the similar FLOPs setting. By further pretraining on the larger dataset ImageNet-21K, we achieve **87.5%** Top-1 accuracy on ImageNet-1K and state-of-the-art segmentation performance on ADE20K with **55.7** mIoU. The code and models will be available at `https://github.com/microsoft/CSWin-Transformer`

1 Introduction

Transformer-based architectures [18, 54, 39, 61] have recently achieved competitive performances compared to their CNN counterparts in various vision tasks. By leveraging the multi-head self-attention mechanism, these vision Transformers demonstrate a high capability in modeling the long-range dependencies, which is especially helpful for handling high-resolution inputs in downstream tasks, e.g., object detection and segmentation. Despite the success, the Transformer architecture with full-attention mechanism [18] is computationally inefficient.

To improve the efficiency, one typical way is to limit the attention region of each token from full-attention to local/windowed attention [39, 56]. To bridge the connection between windows,

*Work done during an internship at Microsoft Research Asia.

Preprint. Under review.

researchers further proposed halo and shift operations to exchange information through nearby windows. However, the receptive field is enlarged quite slowly and it requires stacking a great number of blocks to achieve global self-attention. A sufficiently large receptive field is crucial to the performance especially for the downstream tasks(e.g., object detection and segmentation). Therefore it is important to achieve large receptive filed efficiently while keeping the computation cost low.

In this paper, we present the *Cross-Shaped Window* (CSWin) self-attention, which is illustrated in Figure 1 and compared with existing self-attention mechanisms. With CSWin self-attention, we perform the self-attention calculation in the horizontal and vertical stripes in parallel, with each stripe obtained by splitting the input feature into stripes of equal width. This stripe width is an important parameter of the cross-shaped window because it allows us to achieve strong modelling capability while limiting the computation cost. Specifically, we adjust the stripe width according to the depth of the network: small widths for shallow layers and larger widths for deep layers. A larger stripe width encourages a stronger connection between long-range elements and achieves better network capacity with a small increase in computation cost. We will provide a detailed mathematical analysis of how the stripe width affects the modeling capability and computation cost.

It is worthwhile to note that with CSWin self-attention mechanism, the self-attention in horizontal and vertical stripes are calculated in parallel. We split the multi-heads into **parallel** groups and apply different self-attention operations onto different groups. This parallel strategy introduces no extra computation cost while enlarging the area for computing self-attention within each Transformer block. This strategy is fundamentally different from existing self-attention mechanisms [57, 39, 70, 26] that apply the same attention operation across multi-heads((Figure 1 b,c,d,e), and perform different attention operations **sequentially**(Figure 1 c,e). We will show through ablation analysis that this difference makes CSWin self-attention much more effective for general vision tasks.

Based on the CSWin self-attention mechanism, we follow the hierarchical design and propose a new vision Transformer architecture named "CSWin Transformer" for general-purpose vision tasks. This architecture provides significantly stronger modeling power while limiting computation cost. To further enhance this vision Transformer, we introduce an effective positional encoding, *Locally-enhanced Positional Encoding* (LePE), which is especially effective and friendly for input varying downstream tasks such as object detection and segmentation. Compared with previous positional encoding methods [57, 47, 13], our LePE imposes the positional information within each Transformer block and directly operates on the attention results instead of the attention calculation. The LePE makes CSWin Transformer more effective and friendly for the downstream tasks.

As a general vision Transformer backbone, the CSWin Transformer demonstrates strong performance on image classification, object detection and semantic segmentation tasks. Under the similar FLOPs and model size, CSWin Transformer variants significantly outperforms previous state-of-the-art (SOTA) vision Transformers. For example, our base variant CSWin-B achieves **85.4%** Top-1 accuracy on ImageNet-1K without any extra training data or label, **53.9** box AP and **46.4** mask AP on the COCO detection task, **51.7** mIOU on the ADE20K semantic segmentation task, surpassing previous state-of-the-art Swin Transformer counterpart by **+1.2**, **+2.0**, **1.4** and **+2.0** respectively. Under a smaller FLOPs setting, our tiny variant CSWin-T even shows larger performance gains, *i.e.,*, **+1.4** point on ImageNet classification, **+3.0** box AP, **+2.0** mask AP on COCO detection and **+4.6** on ADE20K segmentation. Furthermore, when pretraining CSWin Transformer on the larger dataset ImageNet-21K, we achieve **87.5%** Top-1 accuracy on ImageNet-1K and state-of-the-art segmentation performance on ADE20K with **55.7** mIoU.

2 Related Work

Vision Transformers. Convolutional neural networks (CNN) have dominated the computer vision field for many years and achieved tremendous successes [36, 48, 51, 23, 29, 7, 28, 52, 27, 46, 50]. Recently, the pioneering work ViT [18] demonstrates that pure Transformer-based architectures can also achieve very competitive results, indicating the potential of handling the vision tasks and natural language processing (NLP) tasks under a unified framework. Built upon the success of ViT, many efforts have been devoted to designing better Transformer based architectures for various vision tasks, including low-level image processing [5, 58], image classification [54, 67, 65, 14, 21, 59, 12, 61, 12, 66, 31, 55, 19, 24], object detection [3, 74] and semantic segmentation [60, 71, 49]. Rather than concentrating on one special task, some recent works [59, 70, 39] try to design a general vision Transformer backbone for general-purpose vision tasks. They all follow the hierarchical

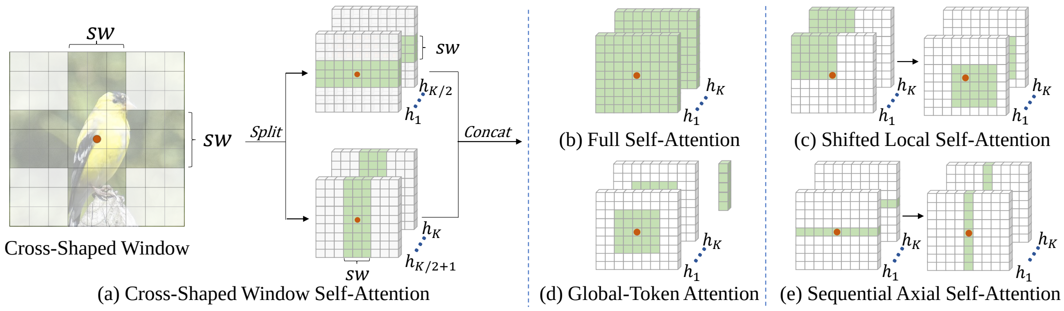

Figure 1: (a) Left: the illustration of the Cross-Shaped Window (CSWin) with stripe width sw for the query point(red dot). Right: the computing of CSWin self-attention, where multi-heads ($\{h_1, \ldots, h_K\}$) is first split into two groups, then two groups of heads perform self-attention in horizontal and vertical stripes respectively, and finally are concatenated together. (b), (c), (d), and (e) are existing self-attention mechanisms.

Transformer architecture but adopt different self-attention mechanisms. The main benefit of the hierarchical design is to utilize the multi-scale features and reduce the computation complexity by progressively decreasing the number of tokens. In this paper,we propose a new hierarchical vision Transformer backbone by introducing cross-shaped window self-attention and locally-enhanced positional encoding.

Efficient Self-attentions. In the NLP field, many efficient attention mechanisms [9, 43, 11, 32, 34, 53, 45, 1] have been designed to improve the Transformer efficiency for handling long sequences. Since the image resolution is often very high in vision tasks, designing efficient self-attention mechanisms is also very crucial. However, many existing vision Transformers [18, 54, 67, 61] still adopt the original full self-attention, whose computation complexity is quadratic to the image size. To reduce the complexity, the recent vision Transformers [39, 56] adopt the local self-attention mechanism [44] and its shifted/haloed version to add the interaction across different local windows. Another efficient self-attention mechanism used in image Transformers is axial self-attention [26], it applies the local window along horizontal or vertical axis sequentially to achieve global attention. However, its sequential mechanism and restricted window size limit its performance for representation learning.

Positional Encoding. Since self-attention is permutation-invariant and ignores the token positional information, positional encoding is widely used in Transformers to add such positional information back. Typical positional encoding mechanisms include absolute positional encoding (APE) [57], relative positional encoding (RPE) [47, 39] and conditional positional encoding (CPE) [13]. APE and RPE are often defined as the sinusoidal functions of a series of frequencies or the learnable parameters, which are designed for a specific input size and are not friendly to varying input resolutions. CPE takes the feature as input and can generate the positional encoding for arbitrary input resolutions. Then the generated positional encoding will be added onto the input feature before feeding it into the self-attention block. Our LePE shares a similar spirit as CPE, but proposes to add the positional encoding as a parallel module to the self-attention operation and operates on projected *values* in each Transformer block. This design decouples positional encoding from the self-attention calculation, and can enforce stronger local inductive bias.

3 Method

3.1 Overall Architecture

The overall architecture of CSWin Transformer is illustrated in Figure 2. For an input image with size of $H \times W \times 3$, we follow [61] and leverage the overlapped convolutional token embedding (7×7 convolution layer with stride 4)) to obtain $\frac{H}{4} \times \frac{W}{4}$ patch tokens, and the dimension of each token is C. To produce a hierarchical representation, the whole network consists of four stages. A convolution layer (3×3, stride 2) is used between two adjacent stages to reduce the number of tokens and double the channel dimension. Therefore, the constructed feature maps have $\frac{H}{2^{i+1}} \times \frac{W}{2^{i+1}}$ tokens for the i^{th} stage, which is similar to traditional CNN backbones like VGG/ResNet. Each stage consists of N_i sequential *CSWin Transformer Blocks* and maintains the number of tokens. CSWin Transformer Block has the overall similar topology as the vanilla multi-head self-attention Transformer block with two differences: 1) It replaces the self-attention mechanism with our proposed Cross-Shaped Window

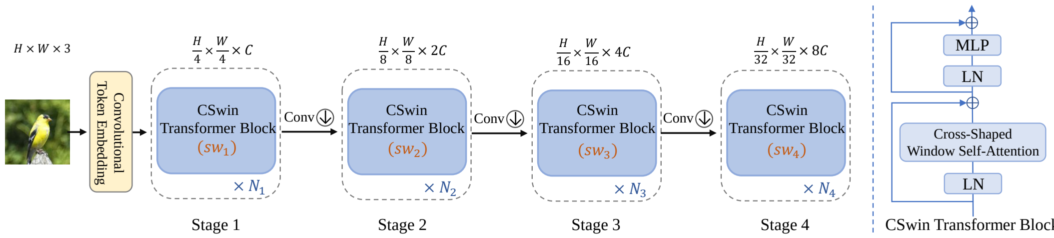

Figure 2: Left: the overall hierarchical architecture of our proposed CSWin Transformer, Right: the illustration of our proposed CSWin Transformer block.

Self-Attention; 2) In order to introduce the local inductive bias, LePE is added as a parallel module to the self-attention branch.

3.2 Cross-Shaped Window Self-Attention

Despite the strong long-range context modeling capability, the computation complexity of the original full self-attention mechanism is quadratic to feature map size. Therefore, it will suffer from huge computation cost for vision tasks that take high resolution feature maps as input, such as object detection and segmentation. To alleviate this issue, existing works [39, 56] suggest to perform self-attention in a local attention window and apply halo or shifted window to enlarge the receptive filed. However, the token within each Transformer block still has limited attention area and requires stacking more blocks to achieve global receptive filed. To enlarge the attention area and achieve global self-attention more efficiently, we present the cross-shaped window self-attention mechanism, which is achieved by performing self-attention in horizontal and vertical stripes in parallel that form a cross-shaped window.

Horizontal and Vertical Stripes. According to the multi-head self-attention mechanism, the input feature $X \in \boldsymbol{R}^{(H\times W)\times C}$ will be first linearly projected to K heads, and then each head will perform local self-attention within either the horizontal or vertical stripes.

For horizontal stripes self-attention, X is evenly partitioned into non-overlapping horizontal stripes $[X^1, .., X^M]$ of equal width sw, and each of them contains $sw \times W$ tokens. Here, sw is the stripe width and can be adjusted to balance the learning capacity and computation complexity. Formally, suppose the projected queries, keys and values of the k^{th} head all have dimension d_k, then the output of the horizontal stripes self-attention for k^{th} head is defined as:

$$
\begin{aligned}
&X = [X^1, X^2, \dots, X^M], \text{ where } X^i \in \boldsymbol{R}^{(sw\times W)\times C} \text{ and } M = H/sw \\
&Y_k^i = \text{Attention}(X^i W_k^Q, X^i W_k^K, X^i W_k^V), \text{ where } i = 1, \dots, M \\
&\text{H-Attention}_k(X) = [Y_k^1, Y_k^2, \dots, Y_k^M]
\end{aligned}
\tag{1}
$$

Where $W_k^Q \in \boldsymbol{R}^{C\times d_k}$, $W_k^K \in \boldsymbol{R}^{C\times d_k}$, $W_k^V \in \boldsymbol{R}^{C\times d_k}$ represent the projection matrices of queries, keys and values for the k^{th} head respectively, and d_k is set as C/K. The vertical stripes self-attention can be similarly derived, and its output for k^{th} head is denoted as V-Attention$_k(X)$.

Assuming natural images do not have directional bias, we equally split the K heads into two parallel groups (each has $K/2$ heads, K is often an even value). The first group of heads perform horizontal stripes self-attention while the second group of heads perform vertical stripes self-attention. Finally the output of these two parallel groups will be concatenated back together.

$$
\begin{aligned}
\text{CSWin-Attention}(X) &= \text{Concat}(\text{head}_1, ..., \text{head}_\text{K})W^O \\
\text{where head}_\text{k} &= \begin{cases} \text{H-Attention}_k(X) & k = 1, \dots, K/2 \\ \text{V-Attention}_k(X) & k = K/2+1, \dots, K \end{cases}
\end{aligned}
$$

Where $W^O \in \boldsymbol{R}^{C\times C}$ is the commonly used projection matrix that projects the self-attention results into the target output dimension (set as C by default). As described above, one key insight in our self-attention mechanism design is splitting the multi-heads into different groups and applying different self-attention operations accordingly. In other words, *the attention area of each token within one Transformer block is enlarged via multi-head grouping*. By contrast, existing self-attention

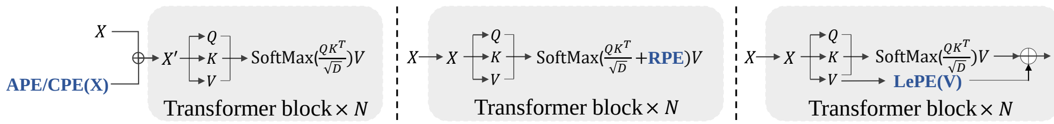

Figure 3: Comparison among different positional encoding mechanisms: APE and CPE introduce the positional information before feeding into the Transformer blocks, while RPE and our LePE operate in each Transformer block. Different from RPE that adds the positional information into the attention calculation, our LePE operates directly upon V and acts as a parallel module. * Here we only draw the self-attention part to represent the Transformer block for simplicity.

mechanisms apply the same self-attention operations across different multi-heads. In the experiment parts, we will show that this design will bring better performance.

Computation Complexity Analysis. The computation complexity of CSWin self-attention is:

$$\Omega(\text{CSWin-Attention}) = HWC * (4C + sw * H + sw * W) \tag{2}$$

For high-resolution inputs, considering H, W will be larger than C in the early stages and smaller than C in the later stages, we choose small sw for early stages and larger sw for later stages. In other words, *adjusting sw provides the flexibility to enlarge the attention area of each token in later stages in an efficient way*. Besides, to make the intermediate feature map size divisible by sw for 224×224 input, we empirically set sw to $1, 2, 7, 7$ for four stages respectively by default.

Locally-Enhanced Positional Encoding. Since the self-attention operation is permutation-invariant, it will ignore the important positional information within the 2D image. To add such information back, different positional encoding mechanisms have been utilized in existing vision Transformers. In Figure 3, we show some typical positional encoding mechanisms and compare them with our proposed locally-enhanced positional encoding. In details, APE [57] and CPE [13] add the positional information into the input token before feeding into the Transformer blocks, while RPE [47] and our LePE incorporate the positional information within each Transformer block. But different from RPE that adds the positional information within the attention calculation (i.e., Softmax(QK^T)), we consider a more straightforward manner and impose the positional information upon the linearly projected *values*. Suppose the edge between the *value* elements v_i and v_j is represented by vectors $e_{ij}^V \in E$, then

$$\text{Attention}(Q, K, V) = \text{SoftMax}(QK^T/\sqrt{d})V + EV, \tag{3}$$

However, if we consider all connections in E, it will require a huge computation cost. We hypothesize that, for a specific input element, the most important positional information is from its local neighborhood. So we propose locally-enhanced positional encoding (LePE) and implement it with a depth-wise convolution operator [10] applying on *value* V:

$$\text{Attention}(Q, K, V) = \text{SoftMax}(QK^T/\sqrt{d})V + \text{DWConv}(V), \tag{4}$$

In this way, LePE can be friendly applied to the downstream tasks that take arbitrary input resolutions as input.

3.3 CSWin Transformer Block

Equipped with the above self-attention mechanism and positional embedding mechanism, CSWin Transformer block is formally defined as:

$$\begin{aligned} \hat{X}^l &= \text{CSWin-Attention}\left(\text{LN}\left(X^{l-1}\right)\right) + X^{l-1}, \\ X^l &= \text{MLP}\left(\text{LN}\left(\hat{X}^l\right)\right) + \hat{X}^l, \end{aligned} \tag{5}$$

where X^l denotes the output of l-th Transformer block or the precedent convolutional layer if exists at the beginning of each stage.

3.4 Architecture Variants

For fair comparison with other vision Transformer baselines under similar parameters and FLOPs, we build four different variants of CSWin Transformer as shown in Table 1: CSWin-T (Tiny), CSWin-S

Models	#Channels	#Blocks in 4 stages	sw in 4 stages	#heads in 4 stages	#Param.	FLOPs
CSWin-T	64	[1, 2, 21, 1]	[1, 2, 7, 7]	[2, 4, 8, 16]	23M	4.3G
CSWin-S	64	[2, 4, 32, 2]	[1, 2, 7, 7]	[2, 4, 8, 16]	35M	6.9G
CSWin-B	96	[2, 4, 32, 2]	[1, 2, 7, 7]	[2, 4, 8, 16]	78M	15.0G
CSWin-L	144	[2, 4, 32, 2]	[1, 2, 7, 7]	[6,12,24,48]	173M	31.5G

Table 1: Detailed configurations of different variants of CSWin Transformer. Note that the FLOPs are calculated with 224×224 input.

ImageNet-1K 224^2 trained models			
Method	#Param.	FLOPs	Top-1
Reg-4G [42]	21M	4.0G	80.0
Eff-B4* [52]	19M	4.2G	**82.9**
DeiT-S [54]	22M	4.6G	79.8
PVT-S [59]	25M	3.8G	79.8
T2T-14 [67]	22M	5.2G	81.5
ViL-S [70]	25M	4.9G	82.0
TNT-S [21]	24M	5.2G	81.3
CViT-15 [4]	27M	5.6G	81.0
Visf-S [8]	40M	4.9G	82.3
LViT-S [37]	22M	4.6G	80.8
CoaTL-S [65]	20M	4.0G	81.9
CPVT-S [13]	23M	4.6G	81.5
Swin-T [39]	29M	4.5G	81.3
CvT-13 [61]	20M	4.5G	81.6
CSWin-T	23M	4.3G	82.7
ImageNet-1K 384^2 finetuned models			
CvT-13 [61]	20M	16.3G	83.0
T2T-14 [67]	22M	17.1G	83.3
CViT$_c$-15 [4]	28M	21.4G	83.5
CSWin-T	23M	14.0G	**84.3**

(a) Tiny Model

ImageNet-1K 224^2 trained models			
Method	#Param.	FLOPs	Top-1
Reg-8G [42]	39M	8.0G	81.7
Eff-B5* [52]	30M	9.9G	**83.6**
PVT-M [59]	44M	6.7G	81.2
PVT-L [59]	61M	9.8G	81.7
T2T-19 [67]	39M	8.9G	81.9
T2T$_t$-19 [67]	39M	9.8G	82.2
ViL-M [70]	40M	8.7G	83.3
MViT-B [20]	37M	7.8G	83.0
CViT-18 [4]	43M	9.0G	82.5
CViT$_c$-18 [4]	44M	9.5G	82.8
Twins-B [12]	56M	8.3G	83.2
Swin-S [39]	50M	8.7G	83.0
CvT-21 [61]	32M	7.1G	82.5
CSWin-S	35M	6.9G	**83.6**
ImageNet-1K 384^2 finetuned models			
CvT-21 [61]	32M	24.9G	83.3
CViT$_c$-18 [4]	45M	32.4G	83.9
CSWin-S	35M	22.0G	**85.0**

(b) Small Model

ImageNet-1K 224^2 trained models			
Method	#Param.	FLOPs	Top-1
Reg-16G [42]	84M	16.0G	82.9
Eff-B6* [52]	43M	19.0G	84.0
DeiT-B [54]	87M	17.5G	81.8
PiT-B [25]	74M	12.5G	82.0
T2T-24 [67]	64M	14.1G	82.3
T2T$_t$-24 [67]	64M	15.0G	82.6
CPVT-B [13]	88M	17.6G	82.3
TNT-B [21]	66M	14.1G	82.8
ViL-B [70]	56M	13.4G	83.2
Twins-L [12]	99M	14.8G	83.7
Swin-B [39]	88M	15.4G	83.3
CSWin-B	78M	15.0G	**84.2**
ImageNet-1K 384^2 finetuned models			
ViT-B/16 [18]	86M	49.3G	77.9
DeiT-B [54]	86M	55.4G	83.1
Swin-B [39]	88M	47.0G	84.2
CSWin-B	78M	47.0G	**85.4**

(c) Base Model

Table 2: Comparison of different models on ImageNet-1K classification. * means the EfficientNet are trained with other input sizes. Here the models are grouped based on the computation complexity.

(Small), CSWin-B (Base), CSWin-L (Large). They are designed by changing the base channel dimension C and the block number of each stage. In all these variants, the expansion ratio of each MLP is set as 4. The head number of the four stages is set as $2, 4, 8, 16$ in the first three variants and $6, 12, 24, 48$ in the last variant respectively.

4 Experiments

To show the effectiveness of CSWin Transformer as a general vision backbone, we conduct experiments on ImageNet-1K [17] classification, COCO [38] object detection, and ADE20K [73] semantic segmentation. We also perform comprehensive ablation studies to analyze each component of CSWin Transformer.

4.1 ImageNet-1K Classification

For fair comparison, we follow the training strategy in DeiT [54] as other baseline Transformer architectures [61, 39]. Specifically, all our models are trained for 300 epochs with the input size of 224×224. We use the AdamW optimizer with weight decay of 0.05 for CSWin-T/S and 0.1 for CSWin-B. The default batch size and initial learning rate are set to 1024 and 0.001, and the cosine learning rate scheduler with 20 epochs linear warm-up is used. We apply increasing stochastic depth [30] augmentation for CSWin-T, CSWin-S, and CSWin-B with the maximum rate as 0.1, 0.3, 0.5 respectively. When reporting the results of 384×384 input, we fine-tune the models for 30 epochs with the weight decay of $1e$-8, learning rate of $1e$-5, batch size of 512.

Method	#Param.	Input Size	FLOPs	Top-1	Method	#Param.	Input Size	FLOPs	Top-1
R-101x3 [35]	388M	384^2	204.6G	84.4	R-152x4 [35]	937M	480^2	840.5G	85.4
ViT-B/16 [18]	86M	384^2	55.4G	84.0	ViT-L/16 [35]	307M	384^2	190.7G	85.2
ViL-B [70]	56M	384^2	43.7G	86.2	—	—	—	—	—
Swin-B [39]	88M	224^2 384^2	15.4G 47.1G	85.2 86.4	Swin-L [39]	197M	224^2 384^2	34.5G 103.9G	86.3 87.3
CSWin-B(ours)	78M	224^2 384^2	15.0G 47.0G	85.9 **87.0**	CSWin-L(ours)	173M	224^2 384^2	31.5G 96.8G	86.5 **87.5**

Table 3: ImageNet-1K fine-tuning results by pre-training on ImageNet-21K datasets.

In Table 2, we compare our CSWin Transformer with state-of-the-art CNN and Transformer architectures. The models are split into three groups based on the computation complexity: Tiny models (around 4.3G FLOPs); Small Models (around 6.8G FLOPs), and Base models (around 15G FLOPs).

It shows that our CSWin Transformers outperform previous state-of-the-art vision Transformers by large margins. For example, CSWin-T achieves 82.7% Top-1 accuracy with only 4.3G FLOPs, surpassing CvT-13, Swin-T and DeiT-S by 1.1%, 1.4% and 2.9% respectively. And for the small and base model setting, our CSWin-S and CSWin-B also achieve the best performance. When finetuned on the 384×384 input, a similar trend is observed, which well demonstrates the powerful learning capacity of our CSWin Transformers.

Compared with state-of-the-art CNNs, we find our CSWin Transformer is the only Transformer based architecture that achieves comparable or even better results than EfficientNet [52] under the small and base settings, while using less computation complexity . It is also worth noting that neural architecture search is used in EfficientNet but not in our CSWin Transformer design.

We further pre-train CSWin Transformer on the larger ImageNet-21K dataset, which contains 14.2M images and 21K classes. Models are trained for 90 epochs with the input size of 224×224. We use the AdamW optimizer with weight decay of 0.1 for CSWin-B and 0.2 for CSWin-L, and the default batch size and initial learning rate are set to 2048 and 0.001. When fine-tuning on ImageNet-1K, we train the models for 30 epochs with the weight decay of $1e$-8, learning rate of $1e$-5, batch size of 512. The increasing stochastic depth [30] augmentation for both CSWin-B and CSWin-L is set to 0.1.

Table.3 reports the results of pre-training on ImageNet-21K. Compared to the results of CSWin-B pre-trained on ImageNet-1K, the large-scale data of ImageNet-21K brings a 1.6%$\sim$1.7% gain. CSWin-B and CSWin-L achieve 87.0% and 87.5% top-1 accuracy, surpassing previous methods.

4.2 COCO Object Detection

Next, we evaluate CSWin Transformer on the COCO objection detection task with the Mask R-CNN [22] and Cascade Mask R-CNN [2] framework respectively. Specifically, we pretrain the backbones on the ImageNet-1K dataset and follow the finetuning strategy used in Swin Transformer [39] on the COCO training set.

Backbone	#Params (M)	FLOPs (G)	Cascade Mask R-CNN 3x +MS					
			AP^b	AP^b_{50}	AP^b_{75}	AP^m	AP^m_{50}	AP^m_{75}
Res50 [23]	82	739	46.3	64.3	50.5	40.1	61.7	43.4
Swin-T [39]	86	745	50.5	69.3	54.9	43.7	66.6	47.1
CSWin-T	80	757	**52.5**	**71.5**	**57.1**	**45.3**	**68.8**	**48.9**
X101-32 [64]	101	819	48.1	66.5	52.4	41.6	63.9	45.2
Swin-S [39]	107	838	51.8	70.4	56.3	44.7	67.9	48.5
CSWin-S	92	820	**53.7**	**72.2**	**58.4**	**46.4**	**69.6**	**50.6**
X101-64 [64]	140	972	48.3	66.4	52.3	41.7	64.0	45.1
Swin-B [39]	145	982	51.9	70.9	56.5	45.0	68.4	48.7
CSWin-B	135	1004	**53.9**	**72.6**	**58.5**	**46.4**	**70.0**	**50.4**

Table 5: Object detection and instance segmentation performance on the COCO val2017 with Cascade Mask R-CNN.

We compare CSWin Transformer with various backbones: previous CNN backbones ResNet [23], ResNeXt(X) [63], and Transformer backbones PVT [59], Twins [12], and Swin [39]. Table 4 reports the results of the Mask R-CNN framework with "1×" (12 training epoch) and "3 × +MS" (36 training epoch with multi-scale training) schedule. It shows that our CSWin Transformer variants clearly outperforms all the CNN and Transformer counterparts. In details, our CSWin-T outperforms Swin-T by **+4.5** box AP, **+3.1** mask AP with the 1× schedule and **+3.0** box

Backbone	#Params (M)	FLOPs (G)	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}
Res50 [23]	44	260	38.0	58.6	41.4	34.4	55.1	36.7	41.0	61.7	44.9	37.1	58.4	40.1
PVT-S [59]	44	245	40.4	62.9	43.8	37.8	60.1	40.3	43.0	65.3	46.9	39.9	62.5	42.8
ViL-S [70]	45	218	44.9	67.1	49.3	41.0	64.2	44.1	47.1	68.7	51.5	42.7	65.9	46.2
TwinsP-S [12]	44	245	42.9	65.8	47.1	40.0	62.7	42.9	46.8	69.3	51.8	42.6	66.3	46.0
Twins-S [12]	44	228	43.4	66.0	47.3	40.3	63.2	43.4	46.8	69.2	51.2	42.6	66.3	45.8
Swin-T [39]	48	264	42.2	64.6	46.2	39.1	61.6	42.0	46.0	68.2	50.2	41.6	65.1	44.8
CSWin-T	42	279	**46.7**	**68.6**	**51.3**	**42.2**	**65.6**	**45.4**	**49.0**	**70.7**	**53.7**	**43.6**	**67.9**	**46.6**
Res101 [23]	63	336	40.4	61.1	44.2	36.4	57.7	38.8	42.8	63.2	47.1	38.5	60.1	41.3
X101-32 [64]	63	340	41.9	62.5	45.9	37.5	59.4	40.2	44.0	64.4	48.0	39.2	61.4	41.9
PVT-M [59]	64	302	42.0	64.4	45.6	39.0	61.6	42.1	44.2	66.0	48.2	40.5	63.1	43.5
ViL-M [70]	60	261	43.4	—	—	39.7	—	—	44.6	66.3	48.5	40.7	63.8	43.7
TwinsP-B [12]	64	302	44.6	66.7	48.9	40.9	63.8	44.2	47.9	70.1	52.5	43.2	67.2	46.3
Twins-B [12]	76	340	45.2	67.6	49.3	41.5	64.5	44.8	48.0	69.5	52.7	43.0	66.8	46.6
Swin-S [39]	69	354	44.8	66.6	48.9	40.9	63.4	44.2	48.5	70.2	53.5	43.3	67.3	46.6
CSWin-S	54	342	**47.9**	**70.1**	**52.6**	**43.2**	**67.1**	**46.2**	**50.0**	**71.3**	**54.7**	**44.5**	**68.4**	**47.7**
X101-64 [64]	101	493	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-L[59]	81	364	42.9	65.0	46.6	39.5	61.9	42.5	44.5	66.0	48.3	40.7	63.4	43.7
ViL-B [70]	76	365	45.1	—	—	41.0	—	—	45.7	67.2	49.9	41.3	64.4	44.5
TwinsP-L [12]	81	364	45.4	—	—	41.5	—	—	—	—	—	—	—	—
Twins-L [12]	111	474	45.9	—	—	41.6	—	—	—	—	—	—	—	—
Swin-B [39]	107	496	46.9	—	—	42.3	—	—	48.5	69.8	53.2	43.4	66.8	46.9
CSWin-B	97	526	**48.7**	**70.4**	**53.9**	**43.9**	**67.8**	**47.3**	**50.8**	**72.1**	**55.8**	**44.9**	**69.1**	**48.3**

Table 4: Object detection and instance segmentation performance on the COCO val2017 with the Mask R-CNN framework. The FLOPs (G) are measured at resolution 800×1280, and the models are pre-trained on the ImageNet-1K dataset. ResNet/ResNeXt results results are copied from [59].

AP, **+2.0** mask AP with the $3\times$ schedule respectively. On the small and base configurations, the similar performance gain can also be achieved.

Table 5 reports the results with the Cascade Mask R-CNN framework. Though Cascade Mask R-CNN is overall stronger than Mask R-CNN, we observe CSWin Transformers still surpass the counterparts by promising margins under different model configurations.

4.3 ADE20K Semantic Segmentation

We further investigate the capability of CSWin Transformer for Semantic Segmentation on the ADE20K [73] dataset. Here we employ the semantic FPN [33] and Upernet [62] as the basic framework. For fair comparison, we follow previous works [59, 39] and train Semantic FPN 80k iterations with batch size as 16, and Upernet 160k iterations with batch size as 16, more details are provided in the supplementary material. In Table 6, we report the results of different methods in terms of mIoU and Multi-scale tested mIoU (MS mIoU). It can be seen that, our CSWin Transformers significantly outperform previous state-of-the-arts under different configurations. In details, CSWin-T, CSWin-S, CSWin-B achieve **+6.7**, **+4.0**, **+3.9** higher mIOU than the Swin counterparts with the Semantic FPN framework, and **+4.8**, **+2.4**, **+2.7** higher mIOU with the Upernet framework. Compared to the CNN counterparts, the performance gain is very promising and demonstrates the potential of vision Transformers again. When using the ImageNet-21K pre-trained model, our CSWin-L further achieves **55.7** mIoU and surpass the previous best model by +2.2 mIoU, while using less computation complexity.

4.4 Ablation Study

To better understand CSWin Transformers, we ablate each key component and evaluate the performance on both classification and downtream tasks. For time consideration, we use Mask R-CNN with 1x schedule as the default setting for detection and instance segmentation evaluation, and Semantic FPN with 80k iterations and single-scale test for segmentation evaluation.

Component Analysis. As explained above, there are two key designs in CSWin self-attention, *i.e.*,, adjusting sw along the network depth and parallel multi-head grouping. To demonstrate their importance, 1) We evaluate the baseline that fixes $sw = 1$ for the first three stages, and observe dramatic performance drop on ImageNet classification and COCO detection, as shown in Table 7. This indicates that adjusting sw to enlarge the attention area is very crucial; 2) We also change the parallel self-attention design into the sequential counterpart without multi-head grouping like

Backbone	Semantic FPN 80k			Upernet 160k		
	#Param.(M)	FLOPs(G)	mIoU(%)	#Param.(M)	FLOPs(G)	mIoU/MS mIoU(%)
Res50 [23]	28.5	183	36.7	—-	—-	—-/—-
PVT-S [59]	28.2	161	39.8	—-	—-	—-/—-
TwinsP-S [12]	28.4	162	44.3	54.6	919	46.2/47.5
Twins-S [12]	28.3	144	43.2	54.4	901	46.2/47.1
Swin-T [39]	31.9	182	41.5	59.9	945	44.5/45.8
CSWin-T (ours)	26.1	202	**48.2**	59.9	959	**49.3/50.4**
Res101 [23]	47.5	260	38.8	86.0	1029	—-/44.9
PVT-M [59]	48.0	219	41.6	—-	—-	—-/—-
TwinsP-B [12]	48.1	220	44.9	74.3	977	47.1/48.4
Twins-B [12]	60.4	261	45.3	88.5	1020	47.7/48.9
Swin-S [39]	53.2	274	45.2	81.3	1038	47.6/49.5
CSWin-S (ours)	38.5	271	**49.2**	64.6	1027	**50.0/50.8**
X101-64 [64]	86.4	—	40.2	—-	—-	—-/—-
PVT-L [59]	65.1	283	42.1	—-	—-	—-/—-
TwinsP-L [12]	65.3	283	46.4	91.5	1041	48.6/49.8
Twins-L [12]	103.7	404	46.7	133.0	1164	48.8/50.2
Swin-B [39]	91.2	422	46.0	121.0	1188	48.1/49.7
CSWin-B (ours)	81.2	464	**49.9**	109.2	1222	**50.8/51.7**
Swin-B† [39]	—-	—-	—-	121.0	1841	50.0/51.7
Swin-L† [39]	—-	—-	—-	234.0	3230	52.1/53.5
CSWin-B† (ours)	—-	—-	—-	109.2	1941	51.8/52.6
CSWin-L† (ours)	—-	—-	—-	207.7	2745	**54.0/55.7**

Table 6: Performance comparison of different backbones on the ADE20K segmentation task. Two different frameworks semantic FPN and Upernet are used. FLOPs are calculated with resolution 512×2048. ResNet/ResNeXt results and Swin FPN results are copied from [59] and [12] respectively. † means the model is pretrained on ImageNet-21K and finetuned with 640×640 resolution.

	ImageNet			COCO				ADE20K		
	#Param.	FLOPs	Top1(%)	#Param.	FLOPs	AP^b	AP^m	#Param.	FLOPs	mIoU(%)
CSWin-T	23M	4.3G	**82.7**	42M	279G	**46.7**	**42.2**	26M	202G	**48.2**
Increasing sw → Fixed $sw = 1$	23M	4.1G	81.9	42M	258G	45.2	40.8	26M	179G	47.5
Parallel SA → Sequential SA	23M	4.3G	82.4	42M	279G	45.1	41.1	26M	202G	46.2
Deep-Narrow → Shallow-Wide Arch	30M	4.8G	82.2	50M	286G	45.8	41.8	34M	209G	46.6
Overlapped→Non-Overlapped CTE	21M	4.2G	82.6	41M	276G	45.4	41.3	25M	199G	47.0

Table 7: Ablation study of each component to better understand CSWin Transformer. “SA”, “Arch”, “CTE” denote “Self-Attention”, “Architecture”, and “Convolutional Token Embedding” respectively.

other self-attention mechanisms [26, 39]. The consistent performance drop on three different tasks demonstrates the effectiveness of multi-heads grouping.

In the last two rows of Table 7, we further ablate the overall architecture design and the token embedding part. Specifically, we find “Deep-Narrow” Transformer architecture is better than the “Shallow-Wide” counterpart. To verify it, we design one shallow-wide variant that has 2,2,6,2 blocks for four stages and the base channel dimensions C as 96. Obviously, even with larger model size and FLOPs, the shallow-wide model performs worse than our deep-narrow design across all the tasks. For the token embedding part, when replacing the overlapped token embedding [18] with the non-overlapped token embedding used in [18], the performance will also get worse, especially on downstream tasks, showing the importance of overlapping token embedding.

Attention Mechanism Comparison. The CSWin self-attention mechanism is the key element in our models. It achieves a strong modeling capability while limiting the computation cost. To verify this, we compare our cross-shaped window self-attention mechanism with existing self-attention mechanisms, including sliding window self-attention [44], shifted window self-attention [39], spatially separable self-attention [12], and sequential axial self-attention [26]. As most of the above methods need even layers in each stage, for a fair comparison, we use the shallow-wide design used in the above subsection ($2, 2, 6, 2$ blocks for the four stages and the base channel is 96). Meanwhile, we apply non-overlapped token embedding [18] and RPE [39] in all these models to reduce the influence

	ImageNet Top1(%)	COCO AP^b	AP^m	ADE20K mIoU(%)
Sliding window [44]	81.4	—	—	—
Shifted window [39]	81.3	42.2	39.1	41.5
Spatially Sep [12]	81.5	42.7	39.5	42.9
Sequential Axial [26]	81.5	40.4	37.6	39.8
Cross-shaped window(ours)	**82.2**	**43.4**	**40.2**	**43.4**

(a) Comparison of different self-attention mechanisms.

	ImageNet Top1(%)	COCO AP^b	AP^m	ADE20K mIoU(%)
No PE	82.5	44.8	41.1	47.0
APE [18]	82.6	45.1	41.1	45.7
CPE [13]	82.2	45.8	41.6	46.1
CPE* [13]	82.4	45.4	41.3	46.6
RPE [47]	**82.7**	45.5	41.3	46.6
LePE	**82.7**	**46.7**	**42.2**	**48.2**

(b) Comparison of different positional encoding mechanisms.

Table 8: Ablation study of different self-attention mechanisms and positional encoding mechanisms.

caused by other factors. The results are reported in Table 8a. Obviously, our CSWin self-attention mechanism performs better than existing self-attention mechanisms across all the tasks. Especially for the sequential axial self-attention, though it can also capture global receptive field with only two blocks, it performs very badly on the downstream tasks because of the smaller attention area within each Transformer block from the sequential and small stripe width ($sw = 1$) design.

Positional Encoding Comparison. The proposed LePE is specially designed to enhance the local positional information on downstream tasks for various input resolutions. In Table 8b, we compare our LePE with other recent positional encoding mechanisms(APE [18], CPE [13], and RPE [47]) for image classification, object detection and image segmentation. Besides, we also test the variants without positional encoding (No PE) and CPE*, which is obtained by applying CPE before every Transformer block. According to the comparison results, we see that: 1) Positional encoding can bring performance gain by introducing the local inductive bias; 2) Though RPE achieves similar performance on the classification task with fixed input resolution, our LePE performs better (+1.2 box AP and +0.9 mask AP on COCO, +0.9 mIoU on ADE20K) on downstream tasks where the input resolution varies; 3) Compared to APE and CPE, our LePE also achieves better performance.

5 Conclusion

In this paper, we have presented a new Vision Transformer architecture named CSWin Transformer. The core design of CSWin Transformer is the CSWin Self-Attention, which performs self-attention in the horizontal and vertical stripes by splitting the multi-heads into *parallel* groups. This multi-head grouping design can enlarge the attention area of each token within one Transformer block efficiently. On the other hand, the mathematical analysis also allows us to increase the stripe width along the network depth to further enlarge the attention area with subtle extra computation cost. We further introduce locally-enhanced positional encoding into CSWin Transformer for downstream tasks. Extensive experiments demonstrate the effectiveness and efficiency of our proposed CSWin Transformer. We achieved the state-of-the-art performance on various vision tasks under constrained computation complexity. We are excited about the results of CSWin Transformer and looking forward to applying it for more vision tasks.

Acknowledgement

We thank many colleagues at Microsoft for their help and useful discussions, including Bin Xiao, Xiyang Dai, Yue Cao, Zheng Zhang and Han Hu.

References

[1] Iz Beltagy, Matthew E Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.

[2] 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.

[3] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European Conference on Computer Vision*, pages 213–229. Springer, 2020.

[4] Chun-Fu Chen, Quanfu Fan, and Rameswar Panda. Crossvit: Cross-attention multi-scale vision transformer for image classification, 2021.

[5] Hanting Chen, Yunhe Wang, Tianyu Guo, Chang Xu, Yiping Deng, Zhenhua Liu, Siwei Ma, Chunjing Xu, Chao Xu, and Wen Gao. Pre-trained image processing transformer. *arXiv preprint arXiv:2012.00364*, 2020.

[6] Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, Zheng Zhang, Dazhi Cheng, Chenchen Zhu, Tianheng Cheng, Qijie Zhao, Buyu Li, Xin Lu, Rui Zhu, Yue Wu, Jifeng Dai, Jingdong Wang, Jianping Shi, Wanli Ouyang, Chen Change Loy, and Dahua Lin. MMDetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*, 2019.

[7] Yunpeng Chen, Jianan Li, Huaxin Xiao, Xiaojie Jin, Shuicheng Yan, and Jiashi Feng. Dual path networks. *arXiv preprint arXiv:1707.01629*, 2017.

[8] Zhengsu Chen, Lingxi Xie, Jianwei Niu, Xuefeng Liu, Longhui Wei, and Qi Tian. Visformer: The vision-friendly transformer, 2021.

[9] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019.

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

[11] Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, et al. Rethinking attention with performers. *arXiv preprint arXiv:2009.14794*, 2020.

[12] Xiangxiang Chu, Zhi Tian, Yuqing Wang, Bo Zhang, Haibing Ren, Xiaolin Wei, Huaxia Xia, and Chunhua Shen. Twins: Revisiting spatial attention design in vision transformers. *arXiv preprint arXiv:2104.13840*, 2021.

[13] Xiangxiang Chu, Zhi Tian, Bo Zhang, Xinlong Wang, Xiaolin Wei, Huaxia Xia, and Chunhua Shen. Conditional positional encodings for vision transformers. *arXiv preprint arXiv:2102.10882*, 2021.

[14] Xiangxiang Chu, Bo Zhang, Zhi Tian, Xiaolin Wei, and Huaxia Xia. Do we really need explicit position encodings for vision transformers? *arXiv e-prints*, pages arXiv–2102, 2021.

[15] MMSegmentation Contributors. Mmsegmentation, an open source semantic segmentation toolbox. `https://github.com/open-mmlab/mmsegmentation`, 2020.

[16] Ekin D. Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V. Le. Randaugment: Practical automated data augmentation with a reduced search space, 2019.

[17] 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.

[18] 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.

[19] Alaaeldin El-Nouby, Natalia Neverova, Ivan Laptev, and Hervé Jégou. Training vision transformers for image retrieval. *arXiv preprint arXiv:2102.05644*, 2021.

[20] Haoqi Fan, Bo Xiong, Karttikeya Mangalam, Yanghao Li, Zhicheng Yan, Jitendra Malik, and Christoph Feichtenhofer. Multiscale vision transformers. *arXiv preprint arXiv:2104.11227*, 2021.

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

[22] 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.

[23] 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.

[24] Shuting He, Hao Luo, Pichao Wang, Fan Wang, Hao Li, and Wei Jiang. Transreid: Transformer-based object re-identification. *arXiv preprint arXiv:2102.04378*, 2021.

[25] Byeongho Heo, Sangdoo Yun, Dongyoon Han, Sanghyuk Chun, Junsuk Choe, and Seong Joon Oh. Rethinking spatial dimensions of vision transformers, 2021.

[26] Jonathan Ho, Nal Kalchbrenner, Dirk Weissenborn, and Tim Salimans. Axial attention in multidimensional transformers. *arXiv preprint arXiv:1912.12180*, 2019.

[27] 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.

[28] 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.

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

[30] 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.

[31] 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.

[32] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International Conference on Machine Learning*, pages 5156–5165. PMLR, 2020.

[33] 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.

[34] Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*, 2020.

[35] Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Joan Puigcerver, Jessica Yung, Sylvain Gelly, and Neil Houlsby. Big transfer (bit): General visual representation learning, 2020.

[36] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.

[37] Yawei Li, Kai Zhang, Jiezhang Cao, Radu Timofte, and Luc Van Gool. Localvit: Bringing locality to vision transformers, 2021.

[38] 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.

[39] 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.

[40] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019.

[41] Boris T Polyak and Anatoli B Juditsky. Acceleration of stochastic approximation by averaging. *SIAM journal on control and optimization*, 30(4):838–855, 1992.

[42] Ilija Radosavovic, Raj Prateek Kosaraju, Ross Girshick, Kaiming He, and Piotr Dollár. Designing network design spaces, 2020.

[43] Jack W Rae, Anna Potapenko, Siddhant M Jayakumar, and Timothy P Lillicrap. Compressive transformers for long-range sequence modelling. *arXiv preprint arXiv:1911.05507*, 2019.

[44] Prajit Ramachandran, Niki Parmar, Ashish Vaswani, Irwan Bello, Anselm Levskaya, and Jonathon Shlens. Stand-alone self-attention in vision models. *arXiv preprint arXiv:1906.05909*, 2019.

[45] Aurko Roy, Mohammad Saffar, Ashish Vaswani, and David Grangier. Efficient content-based sparse attention with routing transformers. *Transactions of the Association for Computational Linguistics*, 9:53–68, 2021.

[46] 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.

[47] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. Self-attention with relative position representations. *arXiv preprint arXiv:1803.02155*, 2018.

[48] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[49] Robin Strudel, Ricardo Garcia, Ivan Laptev, and Cordelia Schmid. Segmenter: Transformer for semantic segmentation. *arXiv preprint arXiv:2105.05633*, 2021.

[50] Ke Sun, Bin Xiao, Dong Liu, and Jingdong Wang. Deep high-resolution representation learning for human pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5693–5703, 2019.

[51] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

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

[53] Yi Tay, Dara Bahri, Liu Yang, Donald Metzler, and Da-Cheng Juan. Sparse sinkhorn attention. In *International Conference on Machine Learning*, pages 9438–9447. PMLR, 2020.

[54] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. *arXiv preprint arXiv:2012.12877*, 2020.

[55] Hugo Touvron, Matthieu Cord, Alexandre Sablayrolles, Gabriel Synnaeve, and Hervé Jégou. Going deeper with image transformers. *arXiv preprint arXiv:2103.17239*, 2021.

[56] Ashish Vaswani, Prajit Ramachandran, Aravind Srinivas, Niki Parmar, Blake Hechtman, and Jonathon Shlens. Scaling local self-attention for parameter efficient visual backbones. *arXiv preprint arXiv:2103.12731*, 2021.

[57] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017.

[58] Ziyu Wan, Jingbo Zhang, Dongdong Chen, and Jing Liao. High-fidelity pluralistic image completion with transformers. *arXiv preprint arXiv:2103.14031*, 2021.

[59] 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.

[60] Yuqing Wang, Zhaoliang Xu, Xinlong Wang, Chunhua Shen, Baoshan Cheng, Hao Shen, and Huaxia Xia. End-to-end video instance segmentation with transformers. *arXiv preprint arXiv:2011.14503*, 2020.

[61] 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.

[62] 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.

[63] 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.

[64] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks, 2017.

[65] Weijian Xu, Yifan Xu, Tyler Chang, and Zhuowen Tu. Co-scale conv-attentional image transformers. *arXiv preprint arXiv:2104.06399*, 2021.

[66] Kun Yuan, Shaopeng Guo, Ziwei Liu, Aojun Zhou, Fengwei Yu, and Wei Wu. Incorporating convolution designs into visual transformers. *arXiv preprint arXiv:2103.11816*, 2021.

[67] 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.

[68] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features, 2019.

[69] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization, 2018.

[70] Pengchuan Zhang, Xiyang Dai, Jianwei Yang, Bin Xiao, Lu Yuan, Lei Zhang, and Jianfeng Gao. Multi-scale vision longformer: A new vision transformer for high-resolution image encoding. *arXiv preprint arXiv:2103.15358*, 2021.

[71] Sixiao Zheng, Jiachen Lu, Hengshuang Zhao, Xiatian Zhu, Zekun Luo, Yabiao Wang, Yanwei Fu, Jianfeng Feng, Tao Xiang, Philip HS Torr, et al. Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers. *arXiv preprint arXiv:2012.15840*, 2020.

[72] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation, 2017.

[73] 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.

[74] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection. *arXiv preprint arXiv:2010.04159*, 2020.

Appendix

We provide more detailed experimental settings in this section.

ImageNet-1K Classification. For a fair comparison, we follow the training strategy in DeiT [54]. Specifically, all our models are trained for 300 epochs with the input size of 224×224. We use the AdamW optimizer with weight decay of 0.05 for CSWin-T/S and 0.1 for CSWin-B. The default batch size and initial learning rate are set to 1024 and 0.001 respectively, and the cosine learning rate scheduler with 20 epochs linear warm-up is used. We adopt most of the augmentation in [54], including RandAugment [16] (rand-m9-mstd0.5-inc1) , Mixup [69] ($prob = 0.8$), CutMix [68] ($prob = 1.0$), Random Erasing [72] ($prob = 0.25$) and Exponential Moving Average [41] (ema-$decay = 0.99992$), increasing stochastic depth [30] ($prob = 0.1, 0.3, 0.5$ for CSWin-T, CSWin-S, and CSWin-B respectively).

When fine-tuning with 384×384 input, we follow the setting in [39] that fine-tune the models for 30 epochs with the weight decay of $1e$-8, learning rate of $1e$-5, batch size of 512. We notice that a large ratio of stochastic depth is beneficial for fine-tuning and keeping it the same as the training stage.

COCO Object Detection and Instance Segmentation. We use two classical object detection frameworks: Mask R-CNN [22] and Cascade Mask R-CNN [2] based on the implementation from mmdetection [6]. For Mask R-CNN, we train it with ImageNet-1K pretrained model with two settings: $1\times$ schedule and $3\times$+MS schedule. For $1\times$ schedule, we train the model with single-scale input (image is resized to the shorter side of 800 pixels, while the longer side does not exceed 1333 pixels) for 12 epochs. We use AdamW [40] optimizer with a learning rate of 0.0001, weight decay of 0.05 and batch size of 16. The learning rate declines at the 8 and 11 epoch with decay rate 0.1. The stochastic depth is also same as the ImageNet-1K setting that 0.1, 0.3, 0.5 for CSWin-T, CSWin-S, and CSWin-B respectively. For $3\times$+MS schedule, we train the model with multi-scale input (image is resized to the shorter side between 480 and 800 while the longer side is no longer than 1333) for 36 epochs. The other settings are same as the $1\times$ except we decay the learning rate at epoch 27 and 33. When it comes to Cascade Mask R-CNN, we use the same $3\times$+MS schedule as Mask R-CNN.

ADE20K Semantic segmentation. Here we consider two semantic segmentation frameworks: UperNet [62] and Semantic FPN [33] based on the implementation from mmsegmentaion [15]. For UperNet, we follow the setting in [39] and use AdamW [40] optimizer with initial learning rate $6e^{-5}$, weight decay of 0.01 and batch size of 16 (8 GPUs with 2 images per GPU) for 160K iterations. The learning rate warmups with 1500 iterations at the beginning and decays with a linear decay strategy. We use the default augmentation setting in mmsegmentation including random horizontal flipping, random re-scaling (ratio range [0.5, 2.0]) and random photo-metric distortion. All the models are trained with input size 512×512. The stochastic depth is also set as the same in the ImageNet-1K setting that 0.1, 0.3, 0.5 for CSWin-T, CSWin-S, and CSWin-B respectively. When it comes to testing, we report both single-scale test result and multi-scale test ([0.5, 0.75, 1.0, 1.25, 1.5, 1.75]$\times$ of that in training).

For Semantic FPN, we follow the setting in [59]. We use AdamW [40] optimizer with initial learning rate $1e^{-4}$, weight decay of $1e^{-4}$ and batch size of 16 (4 GPUs with 4 images per GPU) for 80K iterations.

More analysis of Stripe Width (sw). In our paper, we provide a detailed mathematical analysis

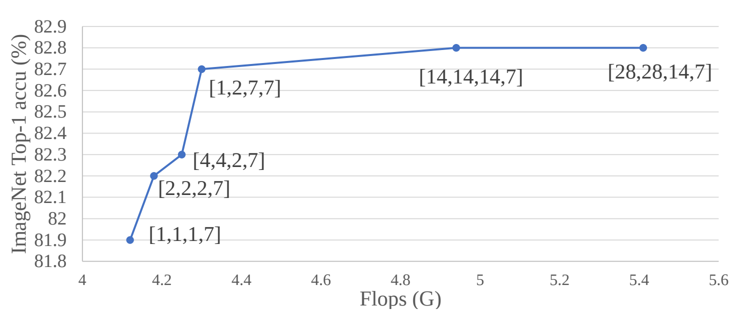

Figure 4: Ablation study on different stripes width. We show the sw of each stage with the form $[sw_1, sw_2, sw_3, sw_4]$ beside each point and X axis is its corresponding Flops.

of how the stripe width affects the modeling capability and computation cost. To further verify this, we design different settings for sw in four stages. Specifically, we vary the $[sw_1, sw_2, sw_3]$ of the first three stages of our CSWin-T and keep the last stage (only one block) with $sw_4 = 7$. As shown in Figure 4, with the increase of sw, the computation cost (FLOPs) increases, and the Top-1 classification accuracy improves greatly at the beginning and slows down when the $[sw_1, sw_2, sw_3]$ are large enough. Our default setting $[1, 2, 7, 7]$ for $[sw_1, sw_2, sw_3, sw_4]$ achieves a better trade-off for accuracy and computation cost.