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

Xiaoyi Dong, Jianmin Bao, Dongdong Chen, Weiming Zhang, Nenghai Yu, Lu Yuan, Dong Chen, Baining Guo
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-Shaped Window 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 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 52.2 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 high segmentation performance on ADE20K with 55.7 mIoU.

1 Introduction

Transformer-based architectures [12, 30, 42, 49] 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 [12] 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 [30, 44]. To bridge the connection between windows, 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 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 [18, 30, 45, 56] 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 [9, 35, 45], 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 high 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 [5, 16, 19–21, 28, 34, 36, 38–40]. Recently, the pioneering work ViT [12] 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 [4, 46], image classification [8, 10, 13, 14, 17, 24, 42, 43, 47, 49, 53–55], object detection [3, 59] and semantic segmentation [37, 48, 57]. Rather than concentrating on one special task, some recent works [30, 47, 56] try to design a general vision Transformer backbone for general-purpose vision tasks. They all follow the hierarchical 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...
Efficient Self-attentions. In the NLP field, many efficient attention mechanisms [1, 6, 7, 25, 27, 31, 33, 41] 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 [12, 42, 49, 55] still adopt the original full self-attention, whose computation complexity is quadratic to the image size. To reduce the complexity, the recent vision Transformers [30, 44] adopt the local self-attention mechanism [32] and its shifted/haloed version to add the interaction across different local windows. Besides, axial self-attention [18] and criss-cross attention [23] propose calculating attention within stripe windows along horizontal or vertical axis. While the performance of axial attention is limited by its sequential mechanism and restricted window size, criss-cross attention is inefficient in practice due to its overlapped window design and ineffective due to its restricted window size. They are the most related works with our CSWin, which could be viewed as a much general and efficient format of these previous works.

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) [45], relative positional encoding (RPE) [30, 35] and conditional positional encoding (CPE) [9]. 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. 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 [49] and leverage the overlapped convolutional token embedding ($7 \times 7$ 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 ($3 \times 3$, 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 $\frac{H}{2^3} \times \frac{W}{2^3}$ 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 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 [30, 44] 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 $X \in R^{(H \times W) \times C}$ 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 $[X^1, \ldots, X^M]$ 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:

$$X = [X^1, X^2, \ldots, X^M],$$

$$Y^i_k = \text{Attention}(X^iW^i_k, X^iW^k_k, X^iW^v_k),$$

$$H\text{-Attention}_k(X) = [y^1_k, y^2_k, \ldots, y^M_k]$$

where $X^i \in \mathbb{R}^{(sw \times W) \times C}$ and $M = H/sw$, $i = 1, \ldots, M$. $W^q_k \in \mathbb{R}^{C \times d_k}$, $W^k_k \in \mathbb{R}^{C \times d_k}$, $W^v_k \in \mathbb{R}^{C \times d_k}$ 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 $V\text{-Attention}_k(X)$.

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 performs horizontal stripes self-attention while the second group of heads performs vertical stripes self-attention. Finally the output of these two parallel groups will be concatenated back together.

$$\text{CSWin-Attention}(X) = \text{Concat}(\text{head}_1, \ldots, \text{head}_K)W^O$$

$$\text{head}_k = \begin{cases} H\text{-Attention}_k(X) & k = 1, \ldots, K/2 \\ V\text{-Attention}_k(X) & k = K/2 + 1, \ldots, K \end{cases}$$

Where $W^O \in \mathbb{R}^{C \times C}$ 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 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:

$$\Omega(\text{CSWin}) = HWC \ast (4C + sw \ast H + sw \ast W)$$

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 $224 \times 224$ input, we empirically set $sw$ to 1, 2, 7, 7 for four stages 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 [45] and CPE [9] add the positional information into the input token before feeding into the Transformer blocks, while RPE [35] 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. Meanwhile, we notice that RPE introduces bias in a per head manner, while our LePE is a per-channel bias, which may show more potential to serve as positional embeddings.

Mathematically, we denote the input sequence as $x = (x_1, \ldots, x_n)$ of $n$ elements, and the output of the attention $z = (z_1, \ldots, z_n)$ of the same length, where $x_i, z_i \in \mathbb{R}^C$. Self-attention computation could be formulated as:

$$z_i = \sum_{j=1}^{n} \alpha_{ij}v_j, \alpha_{ij} = \exp(q_i^T k_j / \sqrt{d})$$

where $q_i, k_i, v_i$ are the queue, key and value get by a linear transformation of the input $x_i$, and $d$ is the feature dimension. Then our Locally-Enhanced position encoding performs as a learnable per-element bias and Eq.4 could be formulated as:

$$z_i^k = \sum_{j=1}^{n} (\alpha_{ij}^k + \beta_{ij}^k)^v_j$$

where $z_i^k$ represents the $k^{th}$ element of vector $z_i$. To make the LePE suitable to varying input size, we set a distance
3.3. CSWin Transformer Block

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

\[ \hat{X}^l = \text{CSWin-Attention} \left( \text{LN} \left( X^{l-1} \right) \right) + X^{l-1}, \]
\[ X^l = \text{MLP} \left( \text{LN} \left( \hat{X}^l \right) \right) + \hat{X}^l, \] (6)

where \( \hat{X}^l \) denotes the output of \( l \)-th Transformer block or the precedent convolutional layer of each stage.

3.4. Architecture Variants

For a fair comparison with other vision Transformers under similar settings, we build four different variants of CSWin Transformer as shown in Table 1: CSWin-T (Tiny), CSWin-S (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 [11] classification, COCO [29] object detection, and ADE20K [58] semantic segmentation. We also perform comprehensive ablation studies to analyze each component of CSWin Transformer. As most of the methods we compared did not report downstream inference speed, we use an extra section to report it for simplicity.

4.1. ImageNet-1K Classification

For fair comparison, we follow the training strategy in DeiT [42] as other baseline Transformer architectures [30, 49]. Specifically, all our models are trained for 300 epochs with the input size of \( 224 \times 224 \). 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 [22] 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 \( 384 \times 384 \) 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.

In Table 2, we compare our CSWin Transformer with state-of-the-art CNN and Transformer architectures. With the limitation of pages, we only compare with a few classical methods here and make a comprehensive comparison in the supplemental materials.

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

<table>
<thead>
<tr>
<th>Models</th>
<th>#Dim</th>
<th>#Blocks</th>
<th>sw</th>
<th>#heads</th>
<th>#Param.</th>
<th>FLOPs</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSWin-T</td>
<td>64</td>
<td>1,2,21,1</td>
<td>2,4,8,16</td>
<td>23M</td>
<td>4.3G</td>
<td></td>
</tr>
<tr>
<td>CSWin-S</td>
<td>64</td>
<td>2,4,32,2</td>
<td>2,4,8,16</td>
<td>35M</td>
<td>6.9G</td>
<td></td>
</tr>
<tr>
<td>CSWin-B</td>
<td>96</td>
<td>2,4,32,2</td>
<td>2,4,8,16</td>
<td>78M</td>
<td>15.0G</td>
<td></td>
</tr>
<tr>
<td>CSWin-L</td>
<td>144</td>
<td>2,4,32,2</td>
<td>1,2,7,7,6</td>
<td>173M</td>
<td>31.5G</td>
<td></td>
</tr>
</tbody>
</table>

Table 1. Detailed configurations of different variants of CSWin Transformer. The FLOPs are calculated with \( 224 \times 224 \) input.

<table>
<thead>
<tr>
<th>Method</th>
<th>Image Size</th>
<th>#Param.</th>
<th>FLOPs Throughput</th>
<th>Top-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eff-B4</td>
<td>380^2</td>
<td>19M</td>
<td>4.2G</td>
<td>349/s</td>
</tr>
<tr>
<td>Eff-B5</td>
<td>456^2</td>
<td>30M</td>
<td>9.9G</td>
<td>169/s</td>
</tr>
<tr>
<td>Eff-B6</td>
<td>528^2</td>
<td>43M</td>
<td>19.0G</td>
<td>96/s</td>
</tr>
<tr>
<td>DeiT-S</td>
<td>224^2</td>
<td>22M</td>
<td>4.6G</td>
<td>940/s</td>
</tr>
<tr>
<td>DeiT-B</td>
<td>224^2</td>
<td>87M</td>
<td>17.5G</td>
<td>292/s</td>
</tr>
<tr>
<td>DeiT-B</td>
<td>384^2</td>
<td>86M</td>
<td>55.4G</td>
<td>85/s</td>
</tr>
<tr>
<td>PVT-S</td>
<td>224^2</td>
<td>25M</td>
<td>3.8G</td>
<td>820/s</td>
</tr>
<tr>
<td>PVT-T</td>
<td>224^2</td>
<td>44M</td>
<td>6.7G</td>
<td>526/s</td>
</tr>
<tr>
<td>PVT-L</td>
<td>224^2</td>
<td>61M</td>
<td>9.8G</td>
<td>367/s</td>
</tr>
<tr>
<td>T2T^6-14</td>
<td>224^2</td>
<td>22M</td>
<td>6.1G</td>
<td>–</td>
</tr>
<tr>
<td>T2T^6-19</td>
<td>224^2</td>
<td>39M</td>
<td>9.8G</td>
<td>–</td>
</tr>
<tr>
<td>T2T^6-24</td>
<td>224^2</td>
<td>64M</td>
<td>15.0G</td>
<td>–</td>
</tr>
<tr>
<td>CvT-13</td>
<td>224^2</td>
<td>20M</td>
<td>4.5G</td>
<td>–</td>
</tr>
<tr>
<td>CvT-21</td>
<td>224^2</td>
<td>32M</td>
<td>7.1G</td>
<td>–</td>
</tr>
<tr>
<td>CvT-21</td>
<td>384^2</td>
<td>32M</td>
<td>24.9G</td>
<td>–</td>
</tr>
<tr>
<td>Swin-T</td>
<td>224^2</td>
<td>29M</td>
<td>4.5G</td>
<td>755/s</td>
</tr>
<tr>
<td>Swin-S</td>
<td>224^2</td>
<td>50M</td>
<td>8.7G</td>
<td>437/s</td>
</tr>
<tr>
<td>Swin-B</td>
<td>224^2</td>
<td>88M</td>
<td>15.4G</td>
<td>278/s</td>
</tr>
<tr>
<td>Swin-B</td>
<td>384^2</td>
<td>88M</td>
<td>47.0G</td>
<td>85/s</td>
</tr>
<tr>
<td>CSWin-T</td>
<td>224^2</td>
<td>23M</td>
<td>4.3G</td>
<td>701/s</td>
</tr>
<tr>
<td>CSWin-S</td>
<td>224^2</td>
<td>35M</td>
<td>6.9G</td>
<td>437/s</td>
</tr>
<tr>
<td>CSWin-B</td>
<td>224^2</td>
<td>78M</td>
<td>15.0G</td>
<td>250/s</td>
</tr>
<tr>
<td>CSWin-B</td>
<td>384^2</td>
<td>78M</td>
<td>47.0G</td>
<td>80/s</td>
</tr>
</tbody>
</table>

Table 2. Comparison of different models on ImageNet-1K.
base model setting, our CSWin-S and CSWin-B also achieve the best performance. When finetuned on the $384 \times 384$ 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 [40] 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 ImageNet-21K dataset, which contains 14.2M images and 21K classes. Models are trained for 90 epochs with the input size of $224 \times 224$. 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 $1 \times 10^{-8}$, learning rate of $1 \times 10^{-5}$, batch size of 512. The increasing stochastic depth [22] 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%~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 [15] and Cascade Mask R-CNN [2] framework respectively. Specifi-
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 $512 \times 2048$. ResNet/ResNeXt results and Swin FPN results are copied from [47] and [8] respectively. 1 means the model is pre-trained on ImageNet-21K and finetuned with $640 \times 640$ resolution.

different model configurations.

4.3. ADE20K Semantic Segmentation

We further investigate the capability of CSWin Transformer for Semantic Segmentation on the ADE20K [58] dataset. Here we employ the semantic FPN [26] and Upernet [50] as the basic framework. For fair comparison, we follow previous works [30, 47] 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, +2.8, +3.0 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 pretrained model, our CSWin-L further achieves 55.7 mIoU and surpasses the previous best model by +2.2 mIoU, while using less computation complexity.

4.4. Inference Speed.

Here we report the inference speed of our CSWin and Swin works. For downstream tasks, we report the FPS of Cascade Mask R-CNN for object detection on COCO and UperNet for semantic segmentation on ADE20K. In most cases, the speed of our model is only slightly slower than Swin (less than 10%), but our model outperforms Swin by large margins. For example, on COCO, CSWin-S are +1.9% box AP and +1.7% mask AP higher than Swin-S with similar inference speed(11.7 FPS vs. 12 FPS). Note that our CSWin-T performs better than Swin-B on box AP(+0.6%), mask AP(+0.3%) with much faster inference speed(14.2 FPS vs. 11.2 FPS), indicating our CSWin achieves better accuracy/FPS trade-offs.

4.5. Ablation Study

To better understand CSWin Transformers, we compare each key component with the previous works under a completely fair setting that we use the same architecture and hyper-parameter for the following experiments, and only vary one component for each ablation. 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.

Parallel Multi-Head Grouping. We first study the effectiveness of our novel “Parallel Multi-Head Grouping” strategy. Here we compare Axial-Attention [18] and Criss-Cross-Attention [23] under the CSWin-T backbone. “Attention region” is used as the computation cost metric for detailed comparison. To simplify, we assume the attention is calculated on a square input that $H = W$.

In Table 8, we find that the “parallel multi-head grouping” is efficient and effective, especially for downstream tasks. When we replace the Parallel manner with Sequential, the performance of CSWin degrades on all tasks. When comparing with previous methods under the similar attention region constrain, our $sw = 1$ CSWin performs slightly better than Axial on ImageNet, while outperforming it by a large margin on downstream tasks. Our $sw = 2$ CSWin performs slightly better than Criss-Cross Attention, while the speed of CSWin is $2 \times \sim 5 \times$ faster than it on different tasks, this further proves that our “parallel” design is much more efficient.
Dynamic Stripe Width. In Fig.4 we study the trade off between stripe width and accuracy. We find that with the increase of stripe width, the computation cost (FLOPS) increase, and the Top-1 classification accuracy improves greatly at the beginning and slows down when the width is large enough. Our default setting [1,2,7,7] achieves a good trade-off between accuracy and FLOPS.

Attention Mechanism Comparison. Following the above analysis on each component of CSWin self-attention, we further compare with existing self-attention mechanisms. As some of the methods need even layers in each stage, for a fair comparison, we use the Swin-T [30] as backbone and only change the self-attention mechanism. In detail, we use 2, 2, 6, 2 blocks for the four stages with the 96 base channel, non-overlapped token embedding [12], and RPE [30]. The results are reported in Table 9. Obviously, our CSWin self-attention mechanism performs better than existing self-attention mechanisms across all the tasks.

Positional Encoding Comparison. The proposed LePE is specially designed to enhance the local positional information on downstream tasks for various input resolutions. Here we use CSWin-T as the backbone and only vary the positional encoding. In Table 10, we compare our LePE with other recent positional encoding mechanisms (APE [12], CPE [9], and RPE [35]) 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. We achieved the state-of-the-art performance on various vision tasks under constrained computation complexity. We are looking forward to applying it for more vision tasks.

Acknowledgement. This work was supported in part by the Natural Science Foundation of China under Grant U20B2047, 62072421, 62002334, and 62121002, Exploration Fund Project of University of Science and Technology of China under Grant YD348002001, and by Fundamental Research Funds for the Central Universities under Grant WK2100000011.
References


