

Swin-Transformer

Abstract

This paper presents a new vision Transformer, called Swin Transformer, that capably serves as a general-purpose backbone for computer vision. Challenges in adapting Transformer from language to vision arise from differences between the two domains, such as large variations in the scale of visual entities and the high resolution of pixels in images compared to words in text.

To address these differences, we propose a hierarchical Transformer whose representation is computed with Shifted windows. The shifted windowing scheme brings greater efficiency by limiting self-attention computation to non-overlapping local windows while also allowing for cross-window connection. This hierarchical architecture has the flexibility to model at various scales and has linear computational complexity with respect to image size.

These qualities of Swin Transformer make it compatible with a broad range of vision tasks, including image classification (87.3 top-1 accuracy on ImageNet-1K) and dense prediction tasks such as object detection (58.7 box AP and 51.1 mask AP on COCO test-dev) and semantic segmentation (53.5 mIoU on ADE20K val). Its performance surpasses the previous state-of-the-art by a large margin of +2.7 box AP and +2.6 mask AP on COCO, and +3.2 mIoU on ADE20K, demonstrating the potential of Transformer-based models as vision backbones. The hierarchical design and the shifted window approach also prove beneficial for all-MLP architectures.

Notes

Backbone

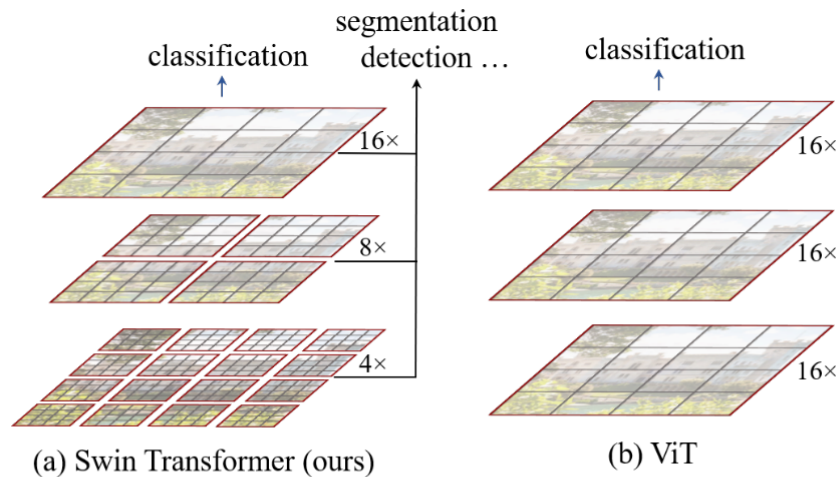


Figure 1. (a) The proposed Swin Transformer builds hierarchical feature maps by merging image patches (shown in gray) in deeper layers and has linear computation complexity to input image size due to computation of self-attention only within each local window (shown in red). It can thus serve as a general-purpose backbone for both image classification and dense recognition tasks. (b) In contrast, previous vision Transformers [20] produce feature maps of a single low resolution and have quadratic computation complexity to input image size due to computation of self-attention globally.

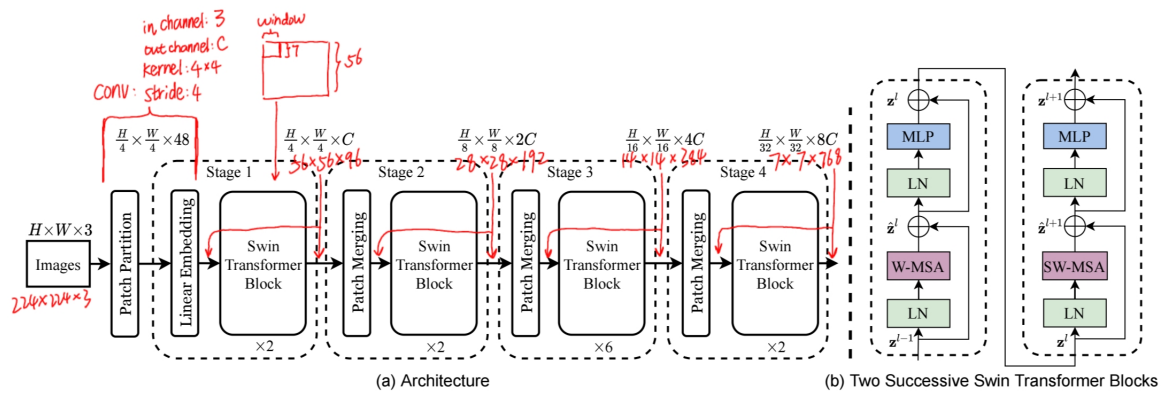
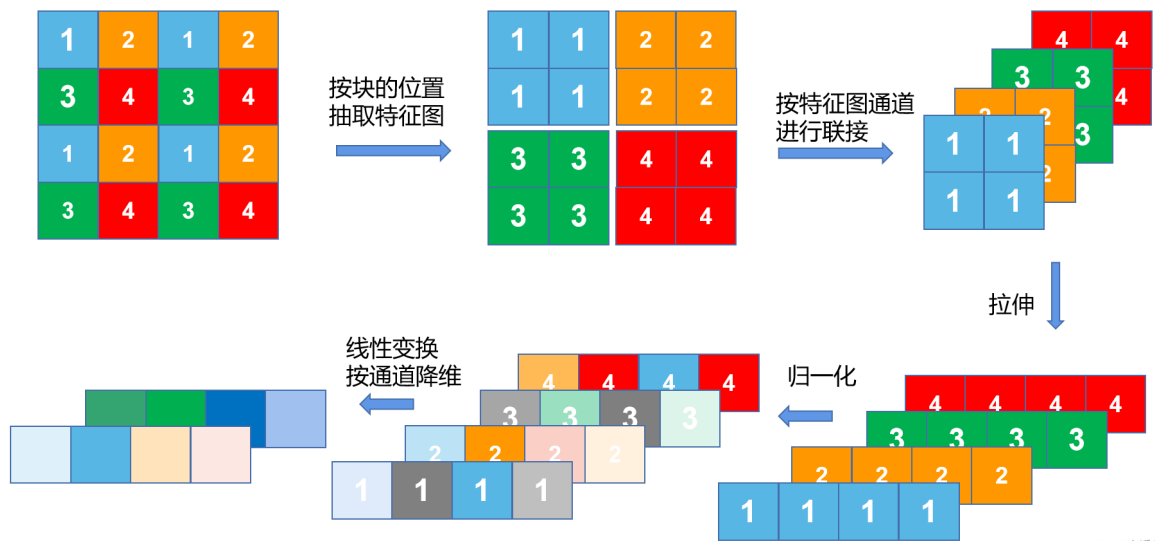


Figure 3. (a) The architecture of a Swin Transformer (Swin-T); (b) two successive Swin Transformer Blocks (notation presented with Eq. (3)). W-MSA and SW-MSA are multi-head self attention modules with regular and shifted windowing configurations, respectively.

Patch Merging

Patch Merging



CSDN @白话先生

<https://blog.csdn.net/liuweizj12/article/details/125094643>

Shifted Windows Multi-Head Self-Attention

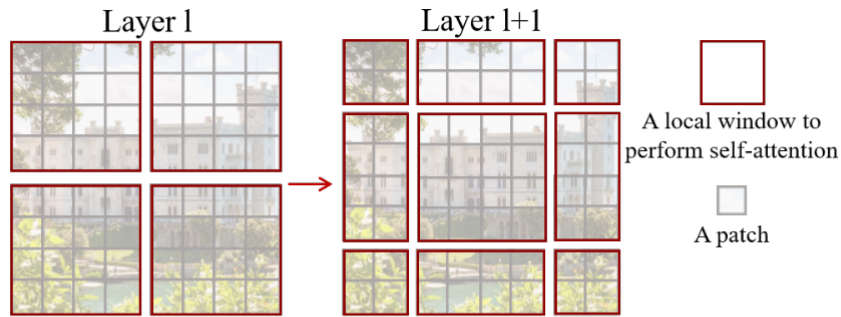


Figure 2. An illustration of the *shifted window* approach for computing self-attention in the proposed Swin Transformer architecture. In layer l (left), a regular window partitioning scheme is adopted, and self-attention is computed within each window. In the next layer $l + 1$ (right), the window partitioning is shifted, resulting in new windows. The self-attention computation in the new windows crosses the boundaries of the previous windows in layer l , providing connections among them.

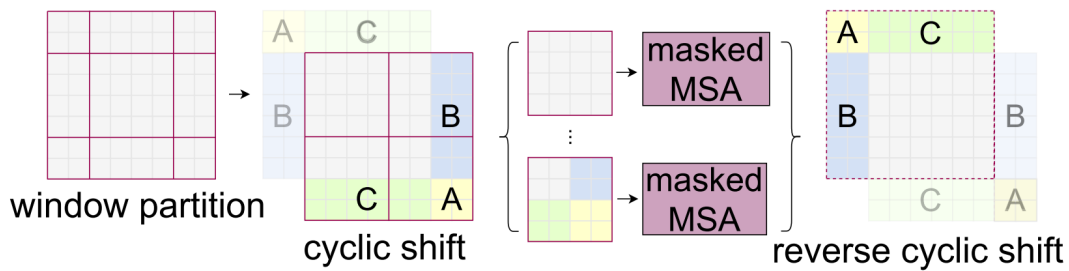


Figure 4. Illustration of an efficient batch computation approach for self-attention in shifted window partitioning.

Relative Position Bias

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

https://blog.csdn.net/qg_37541097/article/details/121119988

Config

T(Tiny), S(Small), B(Base), L(Large):

	downsp. rate (output size)	Swin-T	Swin-S	Swin-B	Swin-L
stage 1	4× (56×56)	concat 4×4, 96-d, LN	concat 4×4, 96-d, LN	concat 4×4, 128-d, LN	concat 4×4, 192-d, LN
		<div>win. sz. 7×7, dim 96, head 3</div> × 2	<div>win. sz. 7×7, dim 96, head 3</div> × 2	<div>win. sz. 7×7, dim 128, head 4</div> × 2	<div>win. sz. 7×7, dim 192, head 6</div> × 2
stage 2	8× (28×28)	concat 2×2, 192-d, LN	concat 2×2, 192-d, LN	concat 2×2, 256-d, LN	concat 2×2, 384-d, LN
		<div>win. sz. 7×7, dim 192, head 6</div> × 2	<div>win. sz. 7×7, dim 192, head 6</div> × 2	<div>win. sz. 7×7, dim 256, head 8</div> × 2	<div>win. sz. 7×7, dim 384, head 12</div> × 2
stage 3	16× (14×14)	concat 2×2, 384-d, LN	concat 2×2, 384-d, LN	concat 2×2, 512-d, LN	concat 2×2, 768-d, LN
		<div>win. sz. 7×7, dim 384, head 12</div> × 6	<div>win. sz. 7×7, dim 384, head 12</div> × 18	<div>win. sz. 7×7, dim 512, head 16</div> × 18	<div>win. sz. 7×7, dim 768, head 24</div> × 18
stage 4	32× (7×7)	concat 2×2, 768-d, LN	concat 2×2, 768-d, LN	concat 2×2, 1024-d, LN	concat 2×2, 1536-d, LN
		<div>win. sz. 7×7, dim 768, head 24</div> × 2	<div>win. sz. 7×7, dim 768, head 24</div> × 2	<div>win. sz. 7×7, dim 1024, head 32</div> × 2	<div>win. sz. 7×7, dim 1536, head 48</div> × 2

Table 7. Detailed architecture specifications.