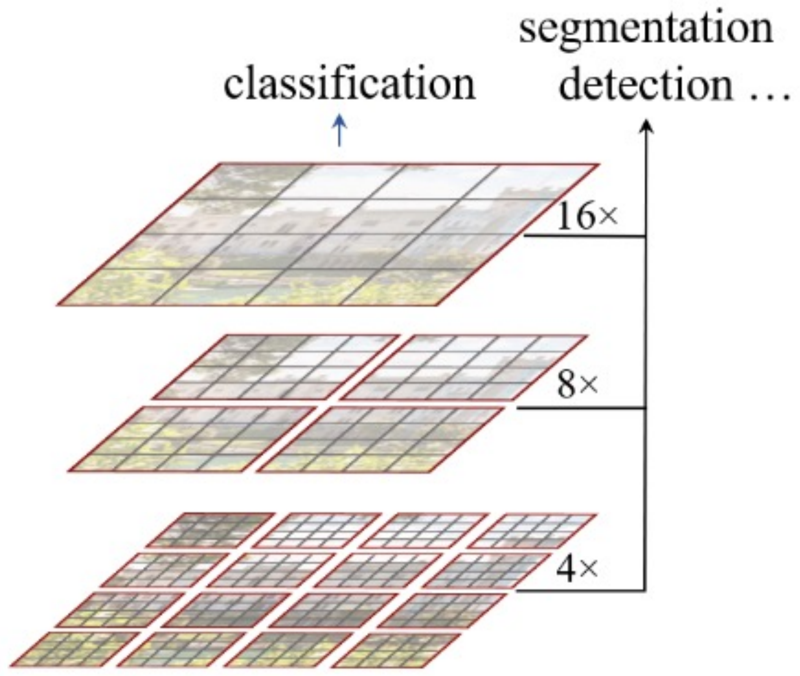
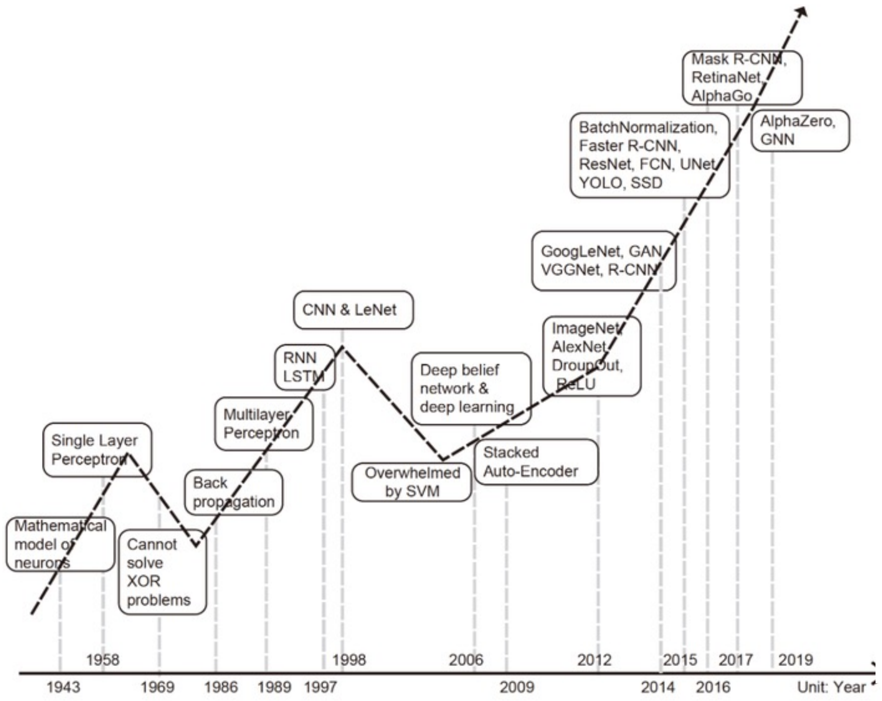
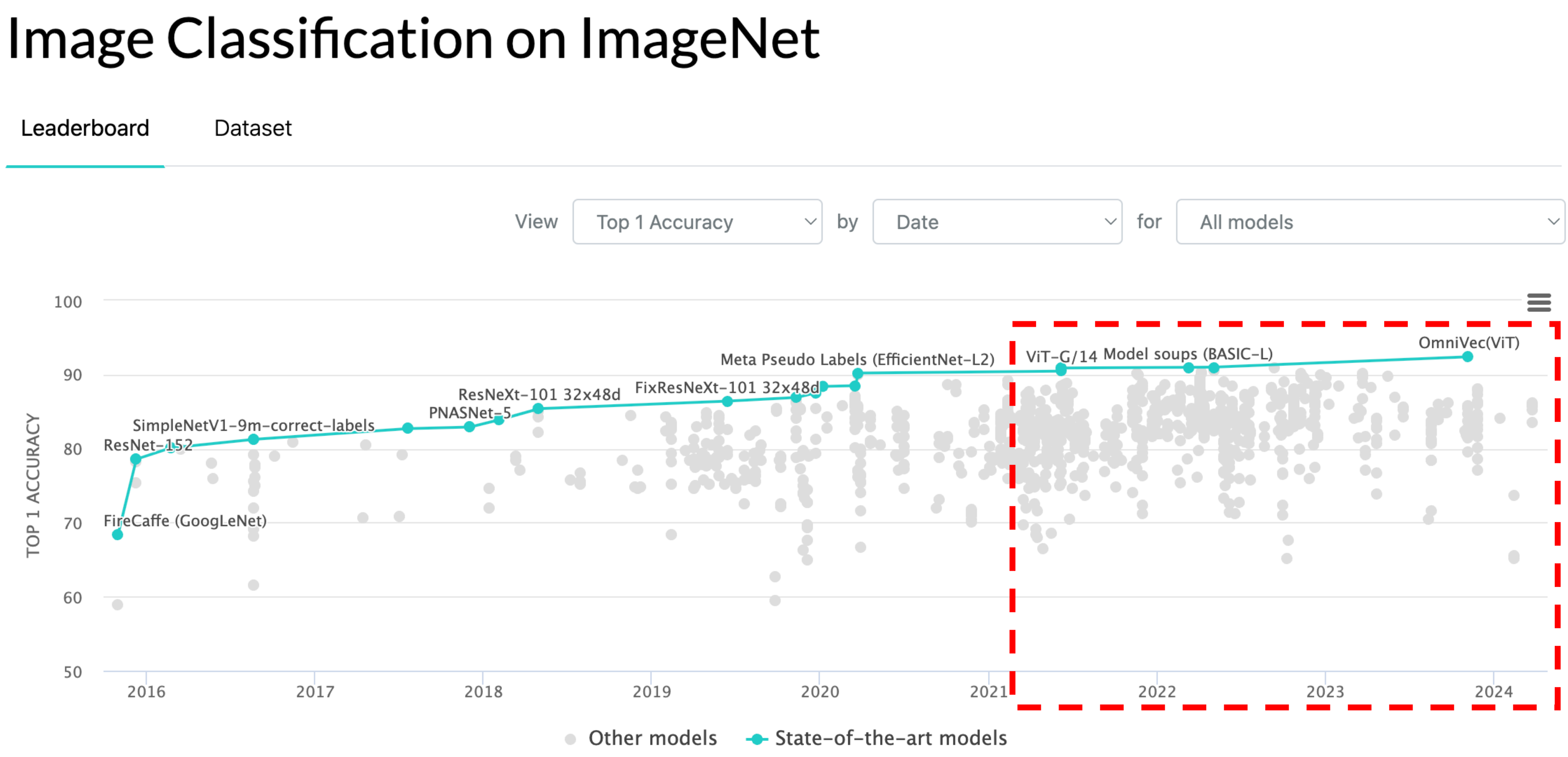
Swin Transformer: Hierarchical Vision Transformer using Shifted Windows

- 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: **high resolution of** pixels in **images** compared to **words in text**.
- The proposed Swin Transformer **merge image patches** in deeper layers. **Computational complexity** has **linear** to input **image size** due to computation of self-attention only within **each local window**.



Swin Transformer: Hierarchical Vision Transformer using Shifted Windows



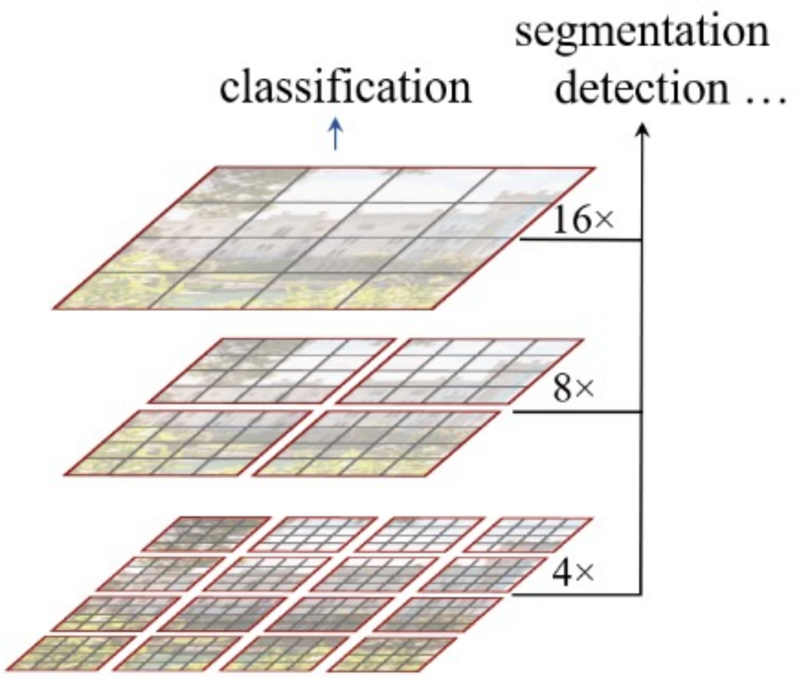


CNNs serve as backbone networks for a variety of vision tasks.

Its tremendous success in the language domain has led researchers to investigate its adaptation to computer vision (image classification and joint vision-language modeling)

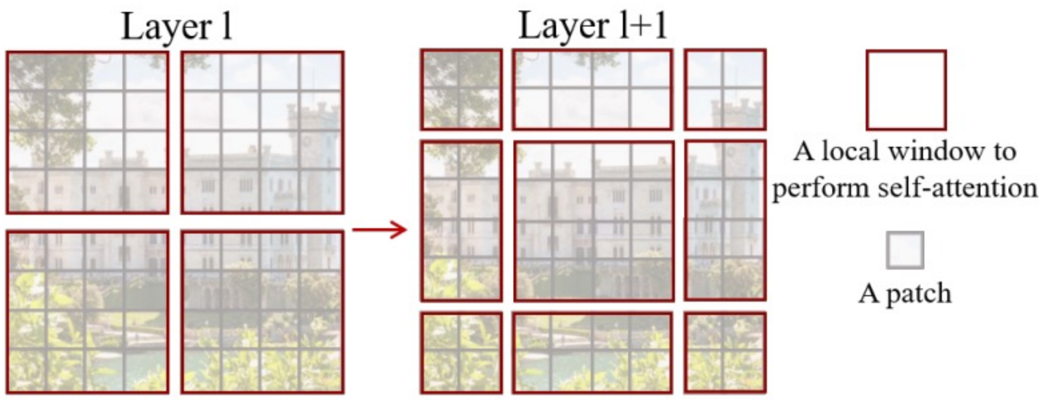
Swin Transformer: Hierarchical Vision Transformer using Shifted Windows

- Swin Transformer constructs a hierarchical representation by starting from small-sized patches and gradually merging neighboring patches in deeper Transformer layers.
- Number of patches in each window is fixed, and thus the complexity becomes linear to image size

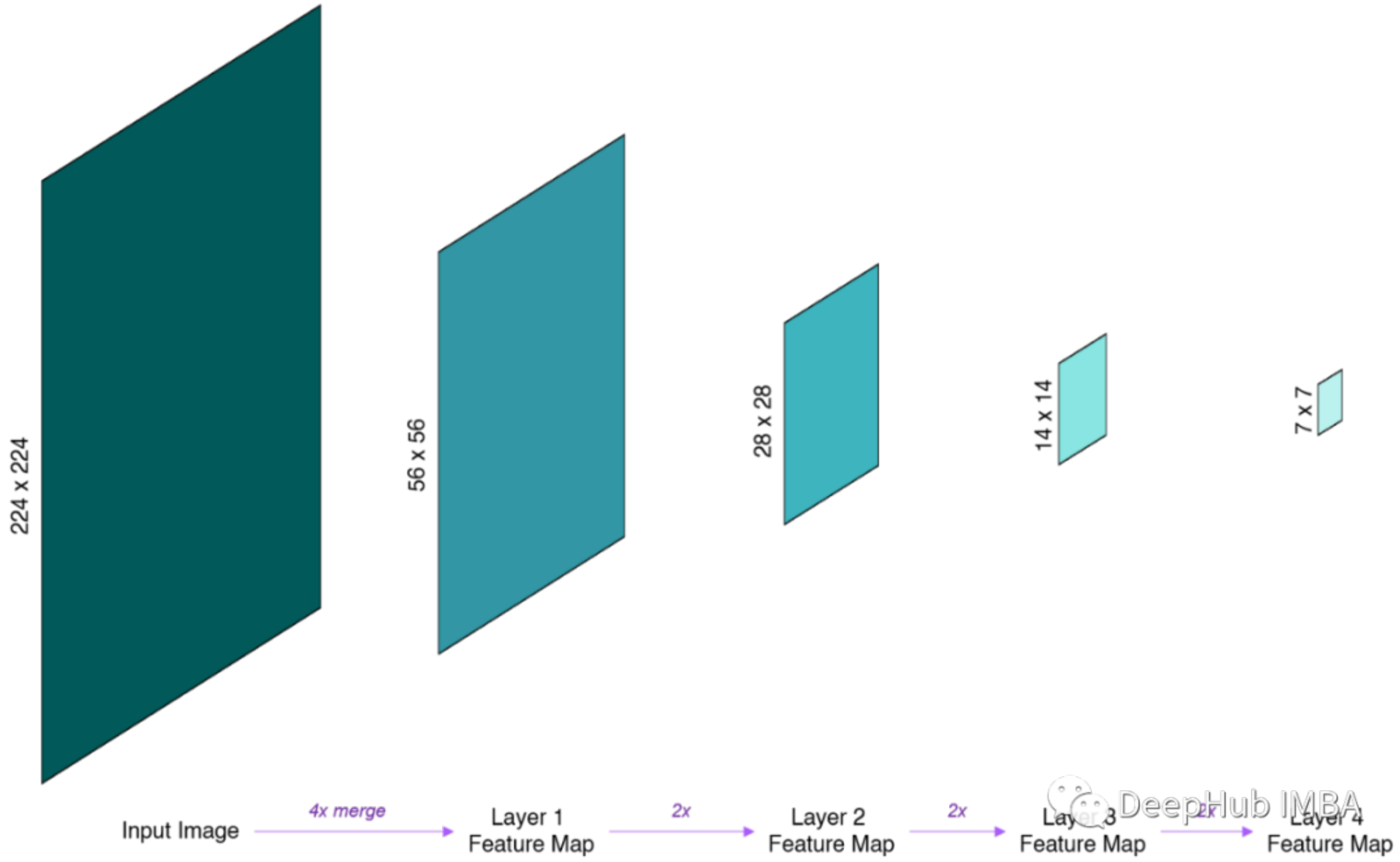


Swin Transformer: Hierarchical Vision Transformer using Shifted Windows

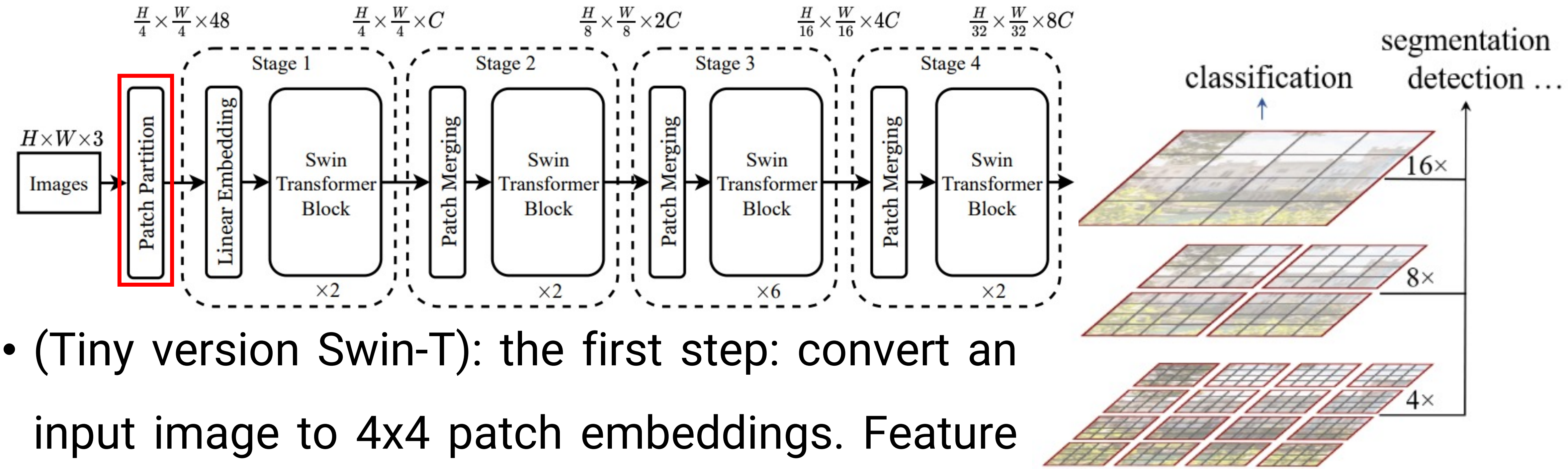
- A key design element of Swin Transformer is its shift of the window partition between consecutive self-attention layers.
- How to explain this figure? (look shifted window - multi self attention)



Hierarchical feature map

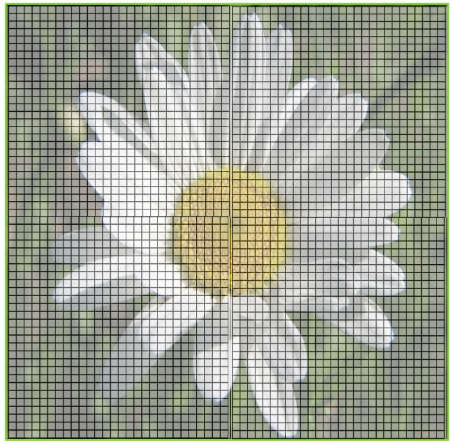


Swin Transformer: Hierarchical Vision Transformer using Shifted Windows

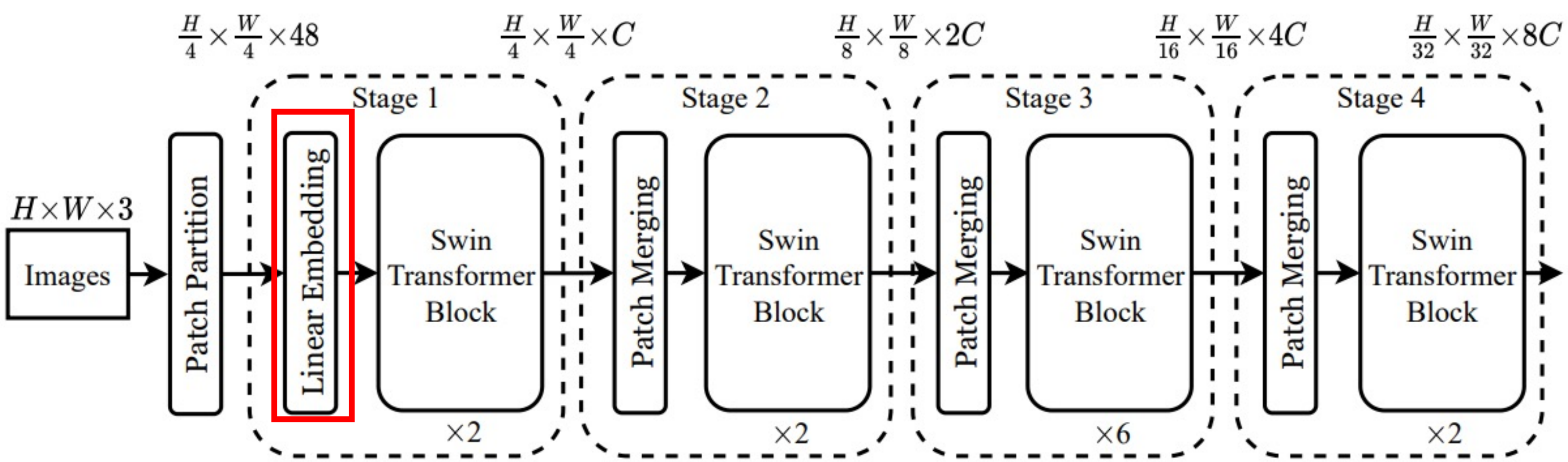


- (Tiny version Swin-T): the first step: convert an input image to 4x4 patch embeddings. Feature dimension of each patch is $4 \times 4 \times 3 = 48$

- image size: 224x224
- patch size: 4x4
- number of patches: 56x56
- total patches: 3136
- every patch: 4x4x3 values



Swin Transformer: Hierarchical Vision Transformer using Shifted Windows



Swin-T: $C = 96$, layer numbers = $\{2, 2, 6, 2\}$

Swin-S: $C = 96$, layer numbers $= \{2, 2, 18, 2\}$

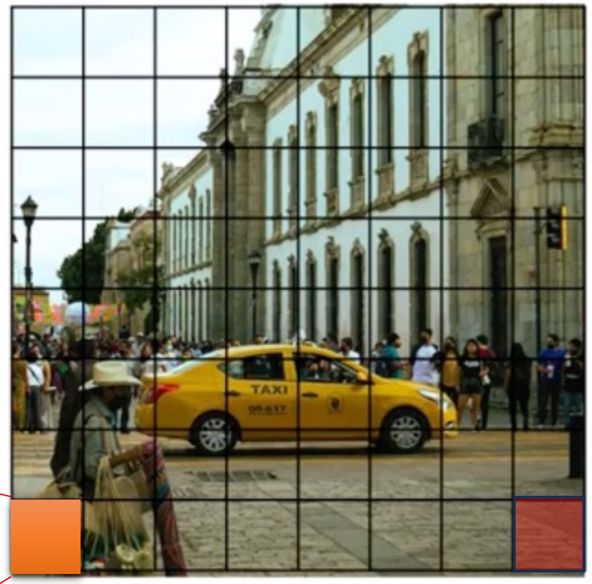
Swin-B: $C = 128$, layer numbers $= \{2, 2, 18, 2\}$

Swin-L: $C = 192$, layer numbers $= \{2, 2, 18, 2\}$

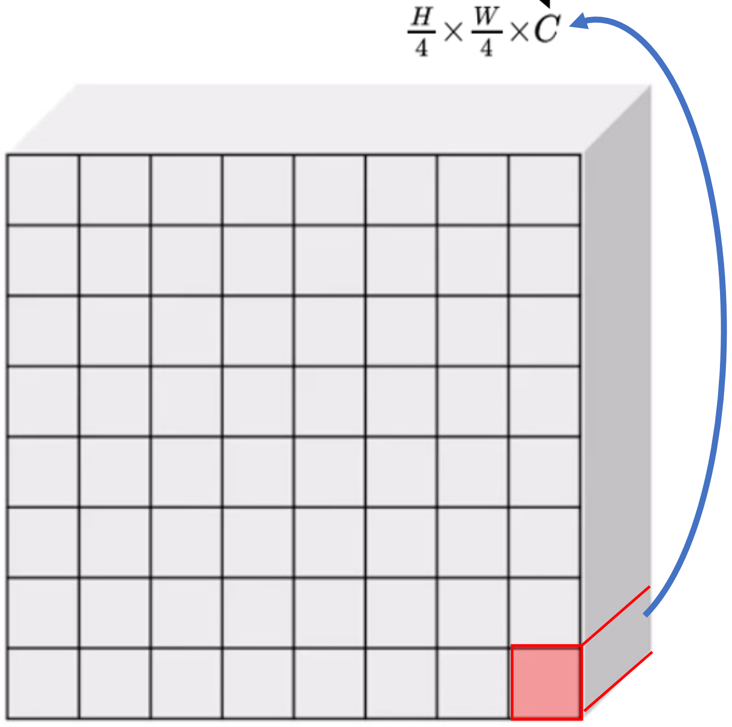
C is a hyper parameter, usually is 96 or 128

- A linear embedding layer is applied on **<u>this raw valued feature</u>** to project it to an arbitrary dimension (denoted as C)

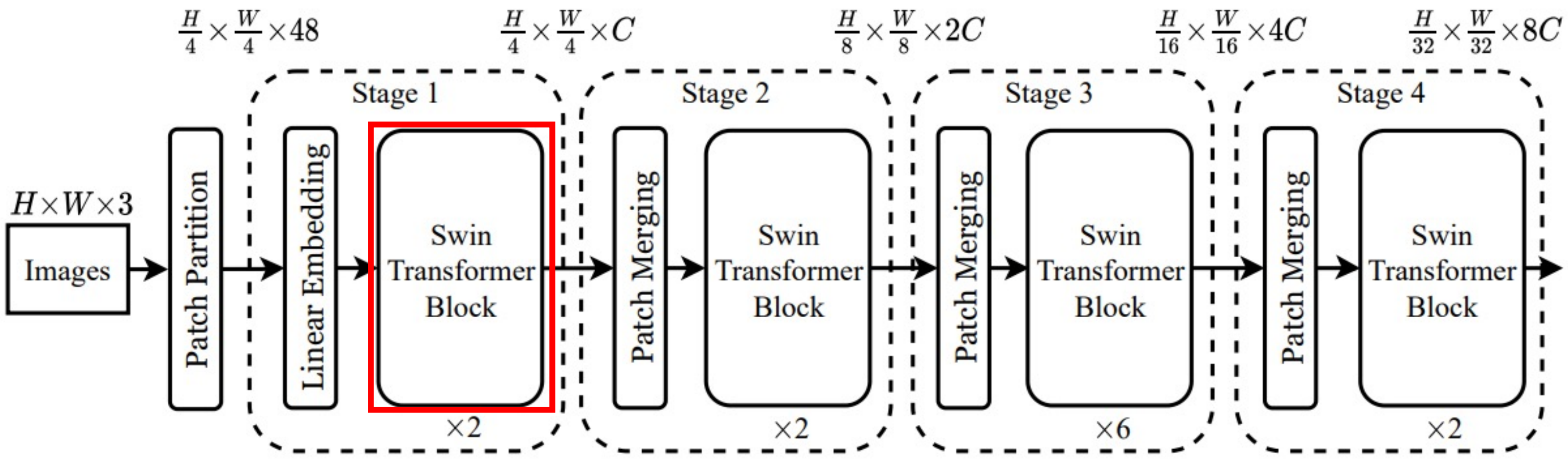
Conv2D (4x4)
Stride = 4
Kernel = C

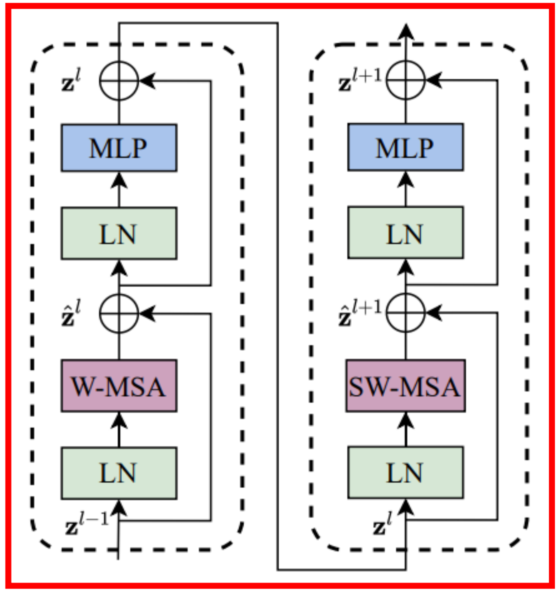


Multi Layer Perceptron



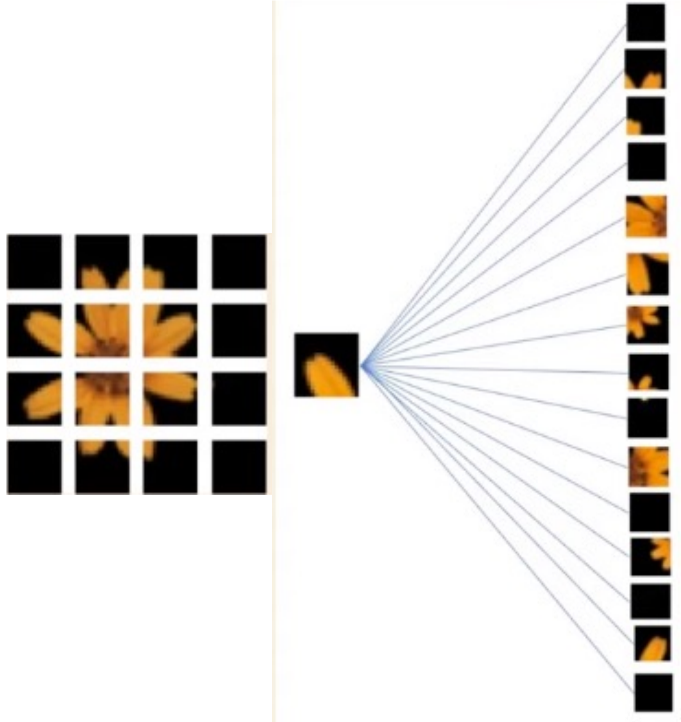
Swin Transformer: Hierarchical Vision Transformer using Shifted Windows



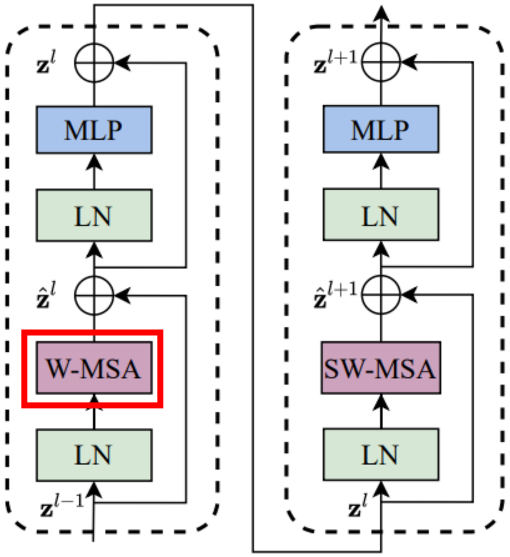


- Swin Transformer block consists of a shifted window based **MSA** module, followed by a 2-layer **MLP with GELU** nonlinearity in between. A **LayerNorm** (LN) layer is applied before each MSA module and each MLP, and a **residual connection** is applied after each module.
  - W-MSA: Window Multilayer Self Attention
  - SW-MSA: Shifted Window Multilayer Self Attention

Swin Transformer: Hierarchical Vision Transformer using Shifted Windows

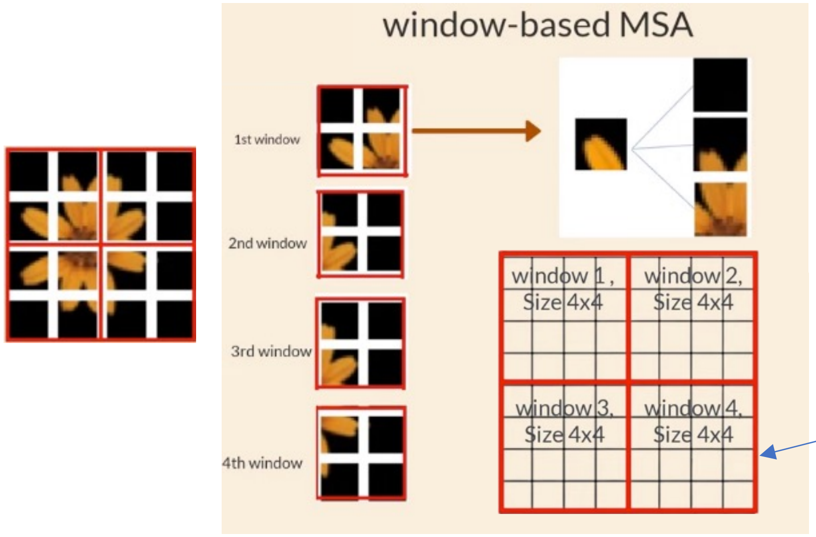


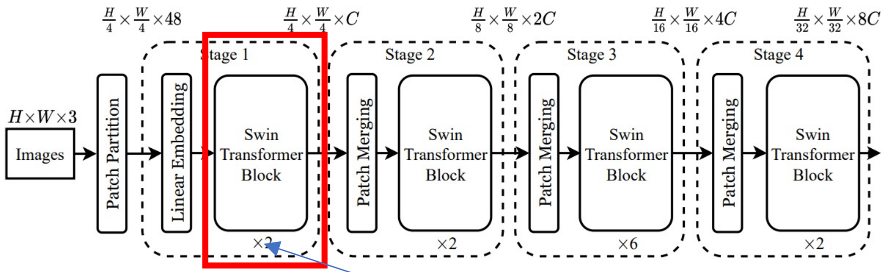
Attension in ViT: calculate the relationship between each patch and all the other patches. Computational complexity meaning unefficient in the high resolution images case. $O(N^2)$, N – number of patches.



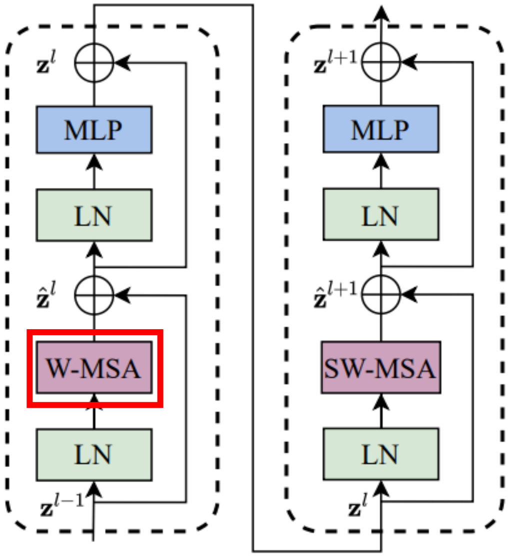
- Instead of computing self-attention on all the patches in an input image, window attention is limited to include patches that correspond to a window of predefined shape.

Swin Transformer: Hierarchical Vision Transformer using Shifted Windows



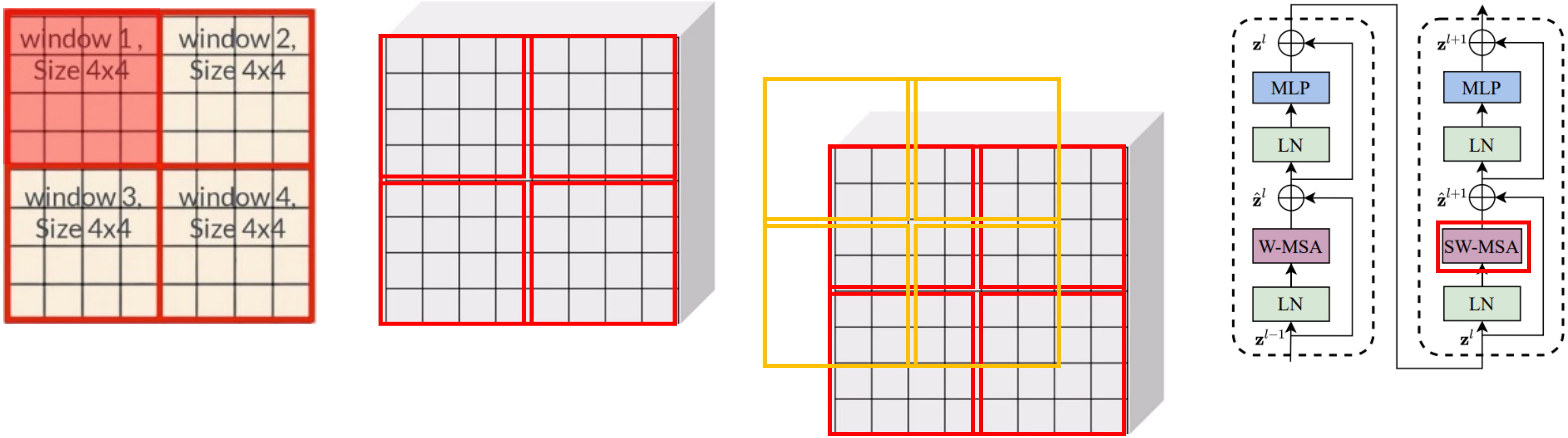


$$\Omega(\mathrm{MSA}) = 4hwC^2 + 2(hw)^2C, \qquad (1)$$



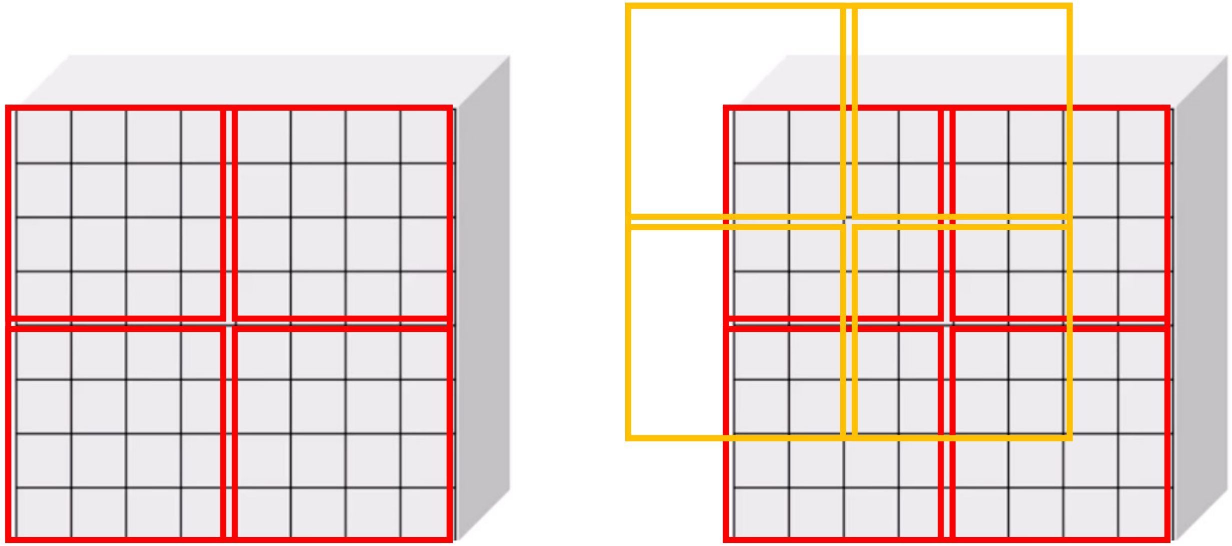
- W-MSA: collection of patches and attention is computed only within each window
- An input image is split into multiple non-overlapping windows of equal size for attention computation.
- The global computation leads to quadratic complexity with respect to the number of tokens.

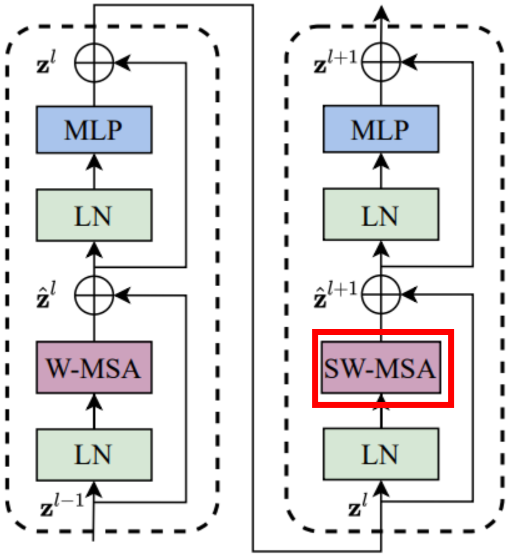
Swin Transformer: Hierarchical Vision Transformer using Shifted Windows

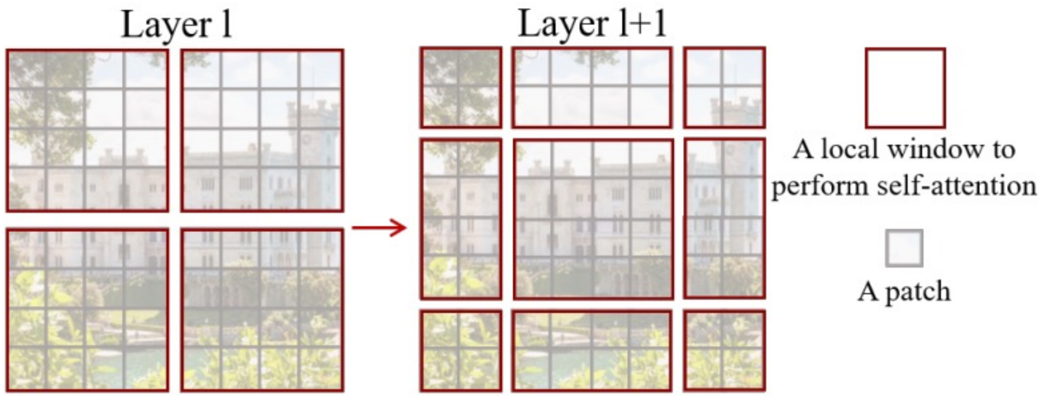


- This windows based self-attention cannot provide connections between windows.
- Windows are shifted in consecutive layers by M/2 and M/2 patches horizontally and vertically respectively. In the following figure windows 1, 2, 3 and 4 are shifted by 2 patches as M=4 in this case.

Swin Transformer: Hierarchical Vision Transformer using Shifted Windows

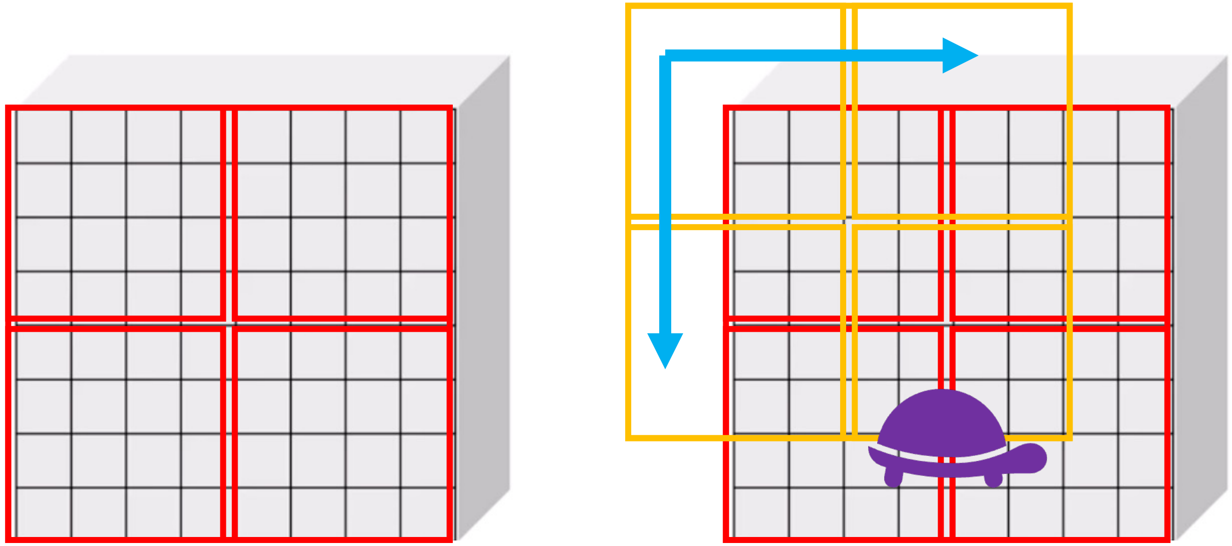


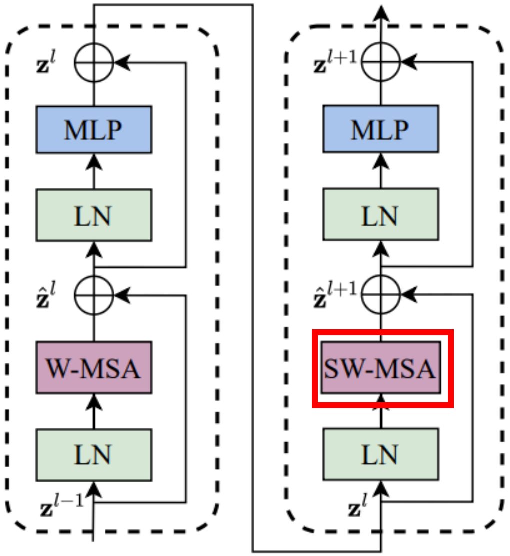




- shift size = M/2 (M: window size)
- Shift 2 times to the lefft
- Shift 2 times to the top

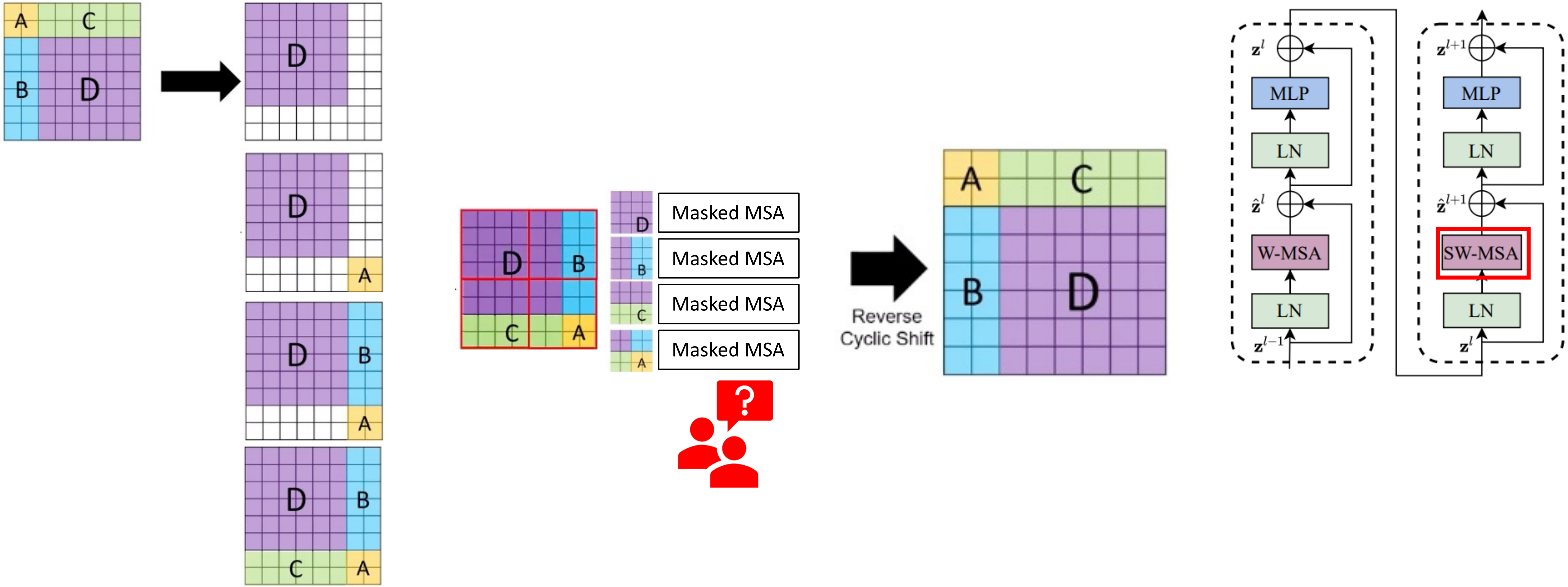
Swin Transformer: Hierarchical Vision Transformer using Shifted Windows





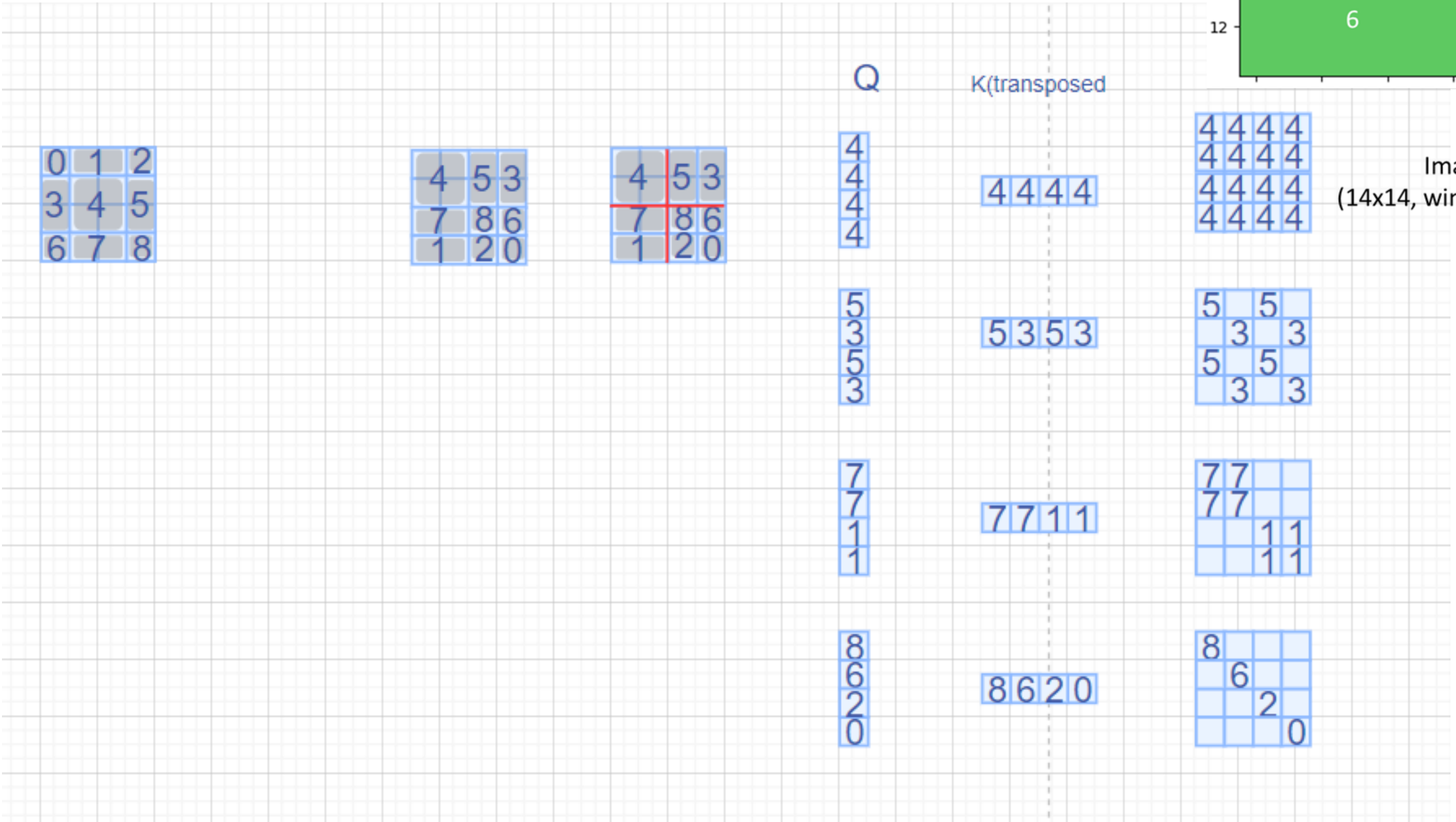
- **Problems:** zero patch is completely useless to compute relationship in MAS.
- WMSA/SWMSA cannot cover object.

Swin Transformer: Hierarchical Vision Transformer using Shifted Windows



- **Solution**: cyclic shifting (rolling) the additional windows then perform attention computation on cyclic shifted configuration, and then shifting the windows back to the original locations.

The Question about the mask of window attention #38



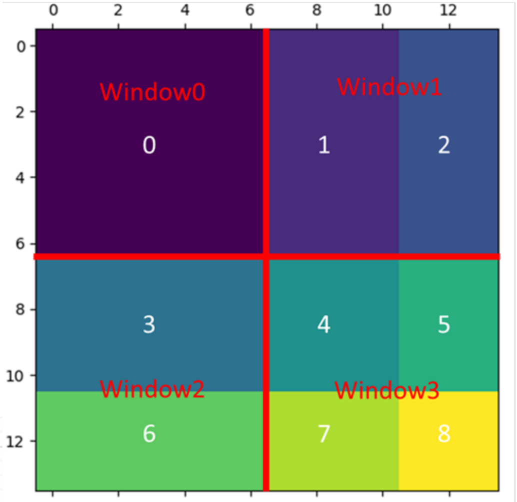
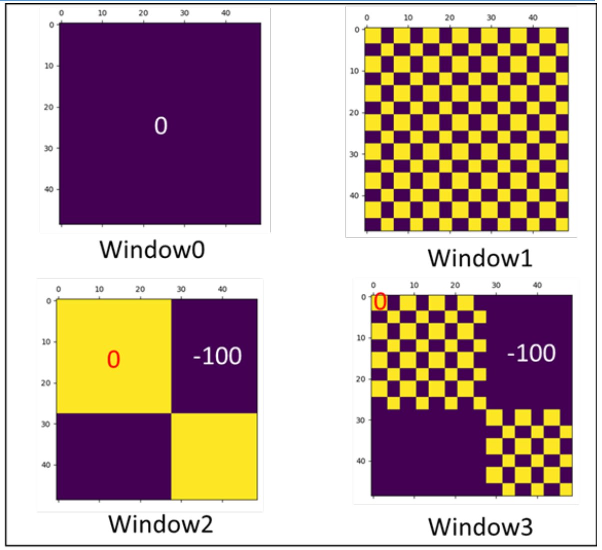
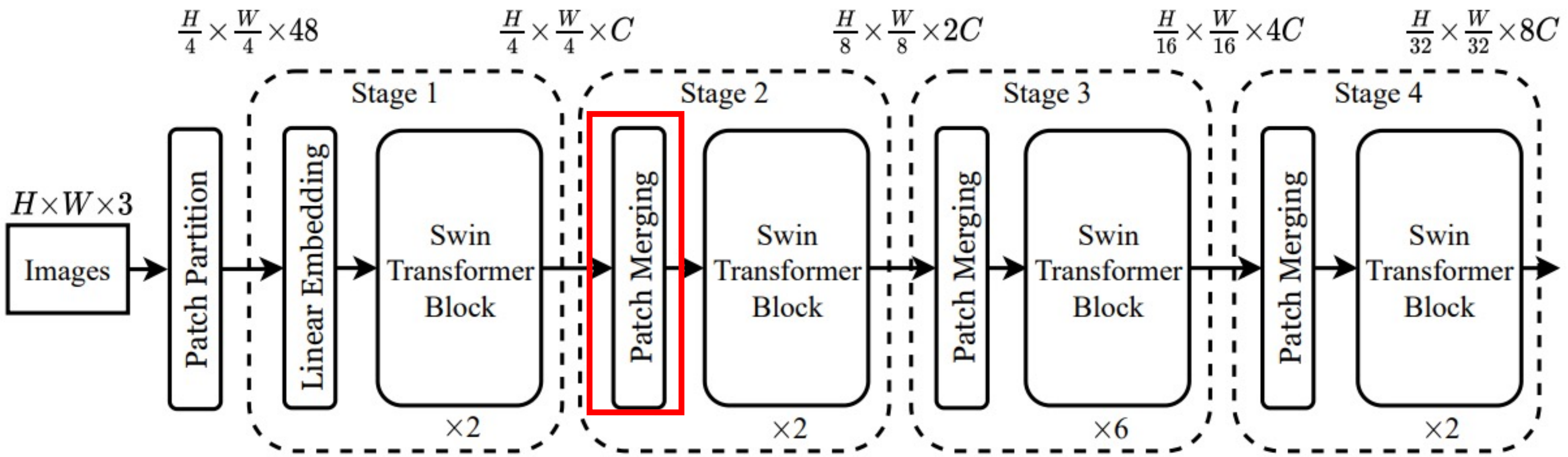


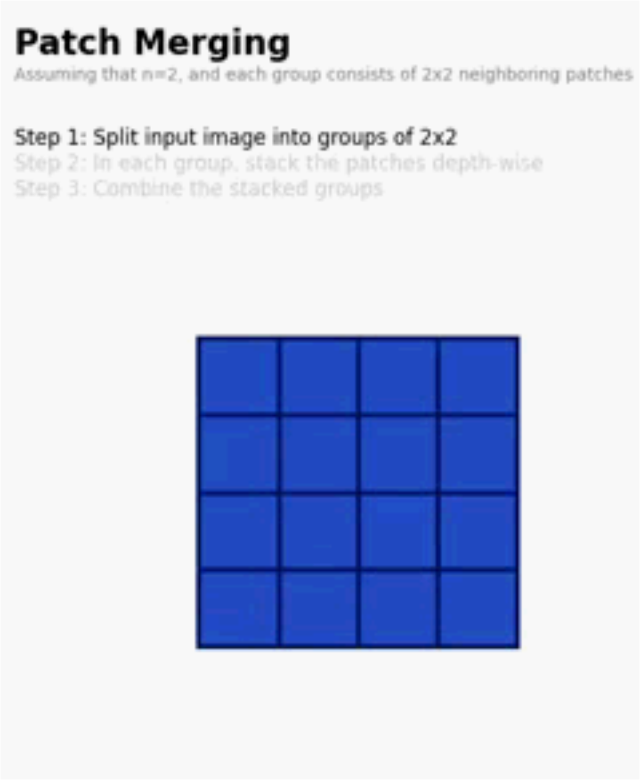
Image Mask
(14x14, window 7x7, shift 3)



Attn Mask

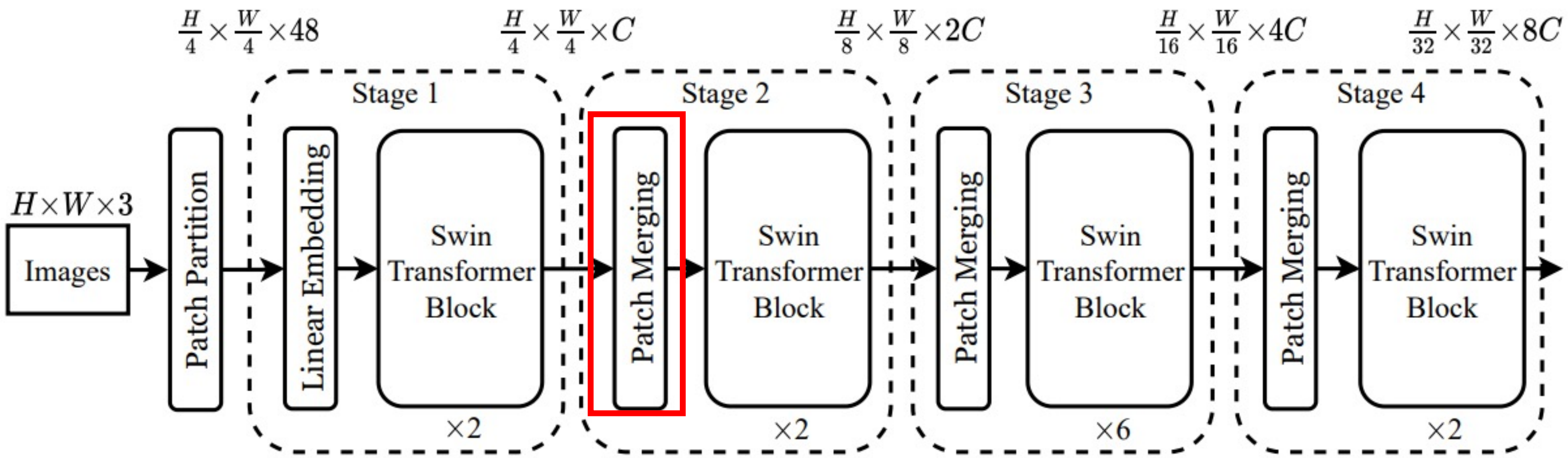
Swin Transformer: Hierarchical Vision Transformer using Shifted Windows

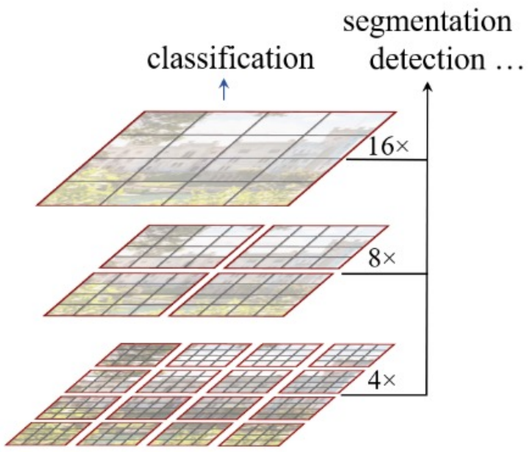


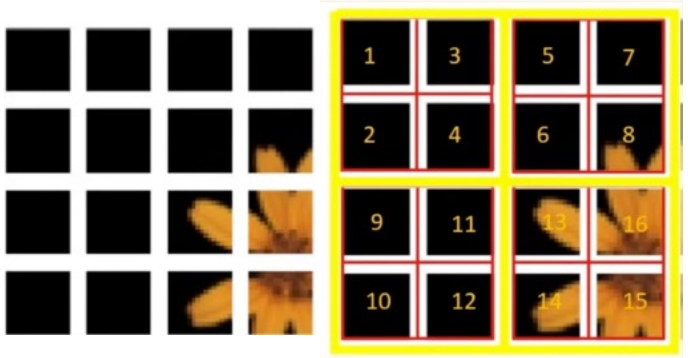


- The patch-merging layer merges four patches at a time, so with each merge the height and width of the image are reduced by a factor of 2.
- in stage 1 the input resolution is (H/4, W/4), but after patch merging the resolution becomes (H/8, W/8), which is the input for stage 2, for stage 3 is (H/16, W/16) and for stage 4 it is (H/32, W/32).

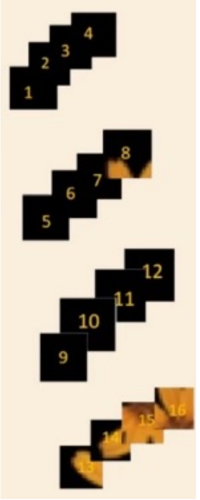
Swin Transformer: Hierarchical Vision Transformer using Shifted Windows

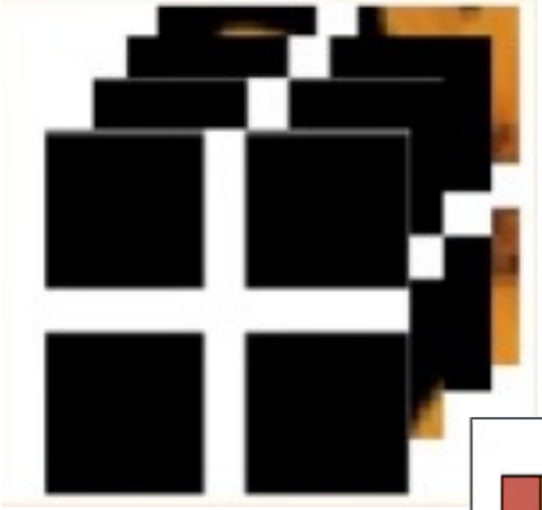


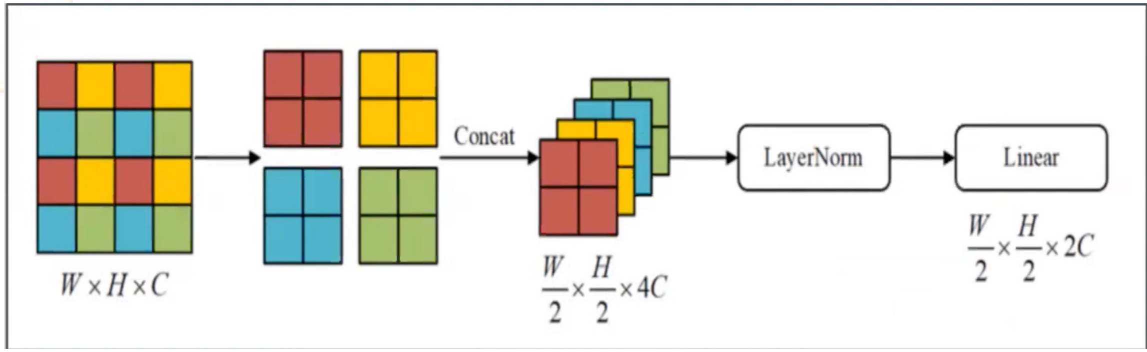




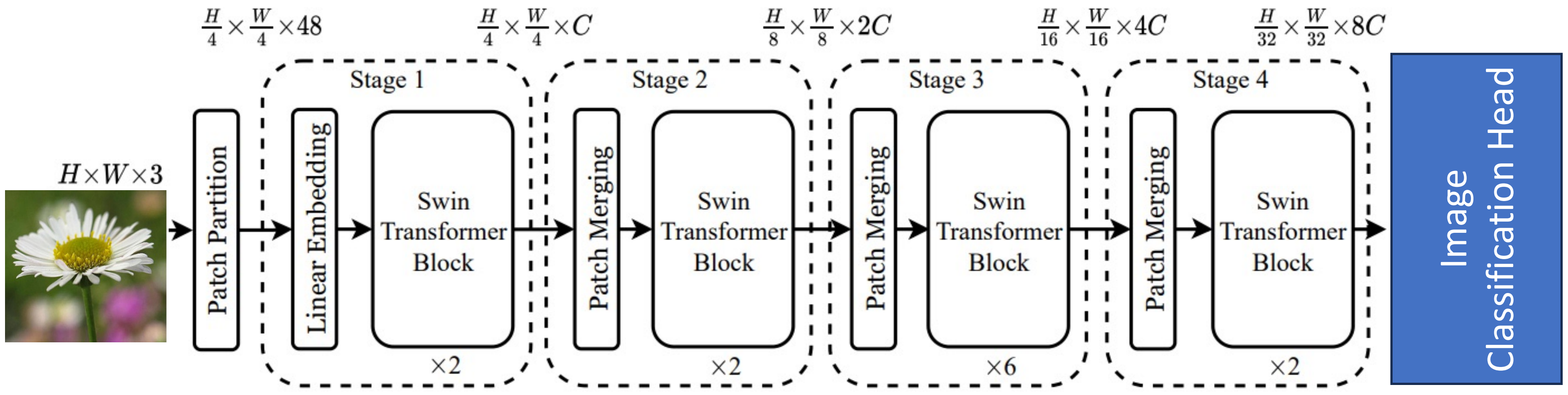
Split image
into 2x2 grid



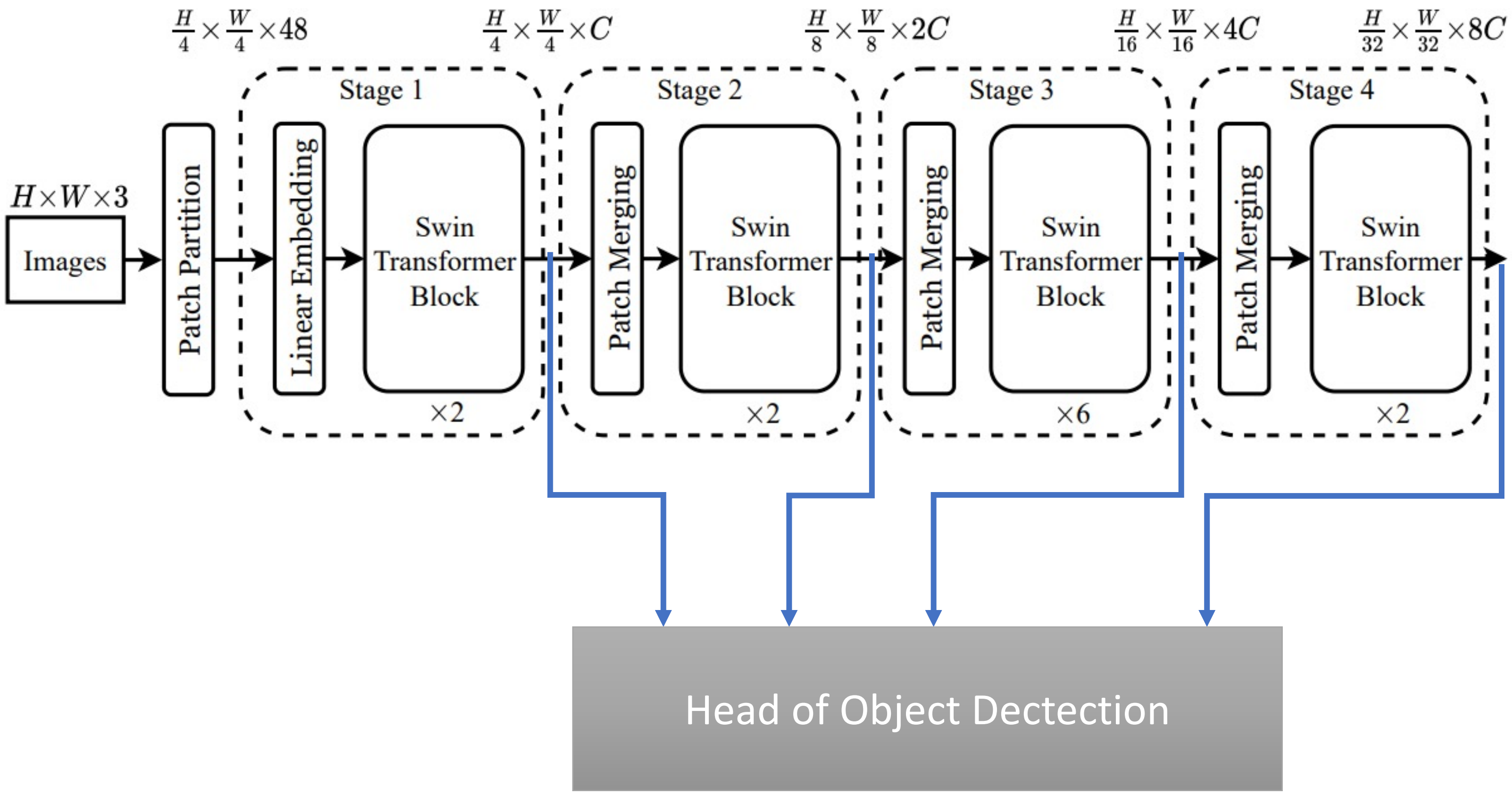




Swin Transformer: Hierarchical Vision Transformer using Shifted Windows

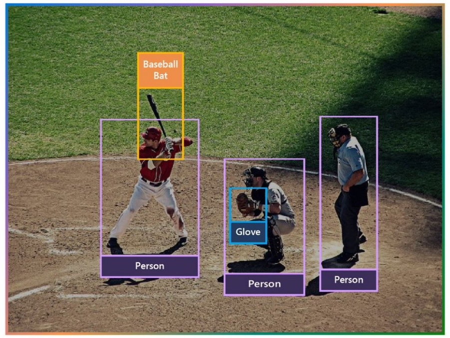


Swin Transformer: Hierarchical Vision Transformer using Shifted Windows



2000

1200



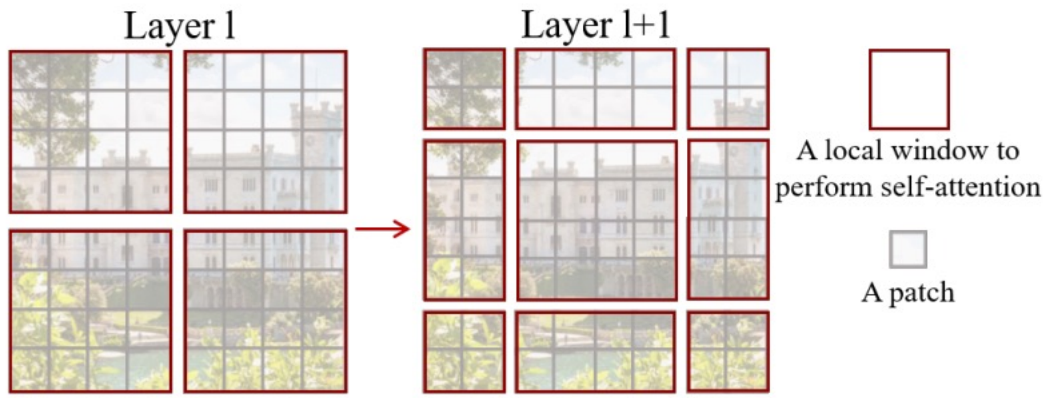
Object Detection (Fine-tuning)

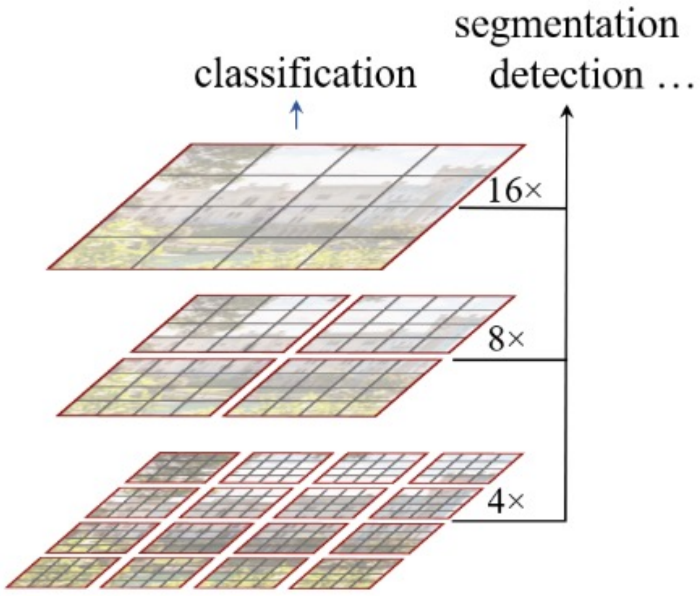
REF

- Swin Transformer: Hierarchical Vision Transformer Using Shifted Windows (thecvf.com)
- A Summary of the Swin Transformer: A Hierarchical Vision Transformer using Shifted Windows (ai-contentlab.com)
- Swin Transformer: Windows of Attention | by Subhash Nerella | Medium
- Swin Transformer - Paper Explained (youtube.com)
- Swin Transformer: Hierarchical Vision Transformer using Shifted Windows (youtube.com)
- The Question about the mask of window attention · Issue #38 · microsoft/Swin-Transformer (github.com)
- EfficientML.ai Lecture 14 - Vision Transformer (MIT 6.5940, Fall 2023) (youtube.com)
- Building Swin Transformer from Scratch using PyTorch: Hierarchical Vision Transformer using Shifted Windows | by Mishra | The Deep Hub | Medium

Implementation

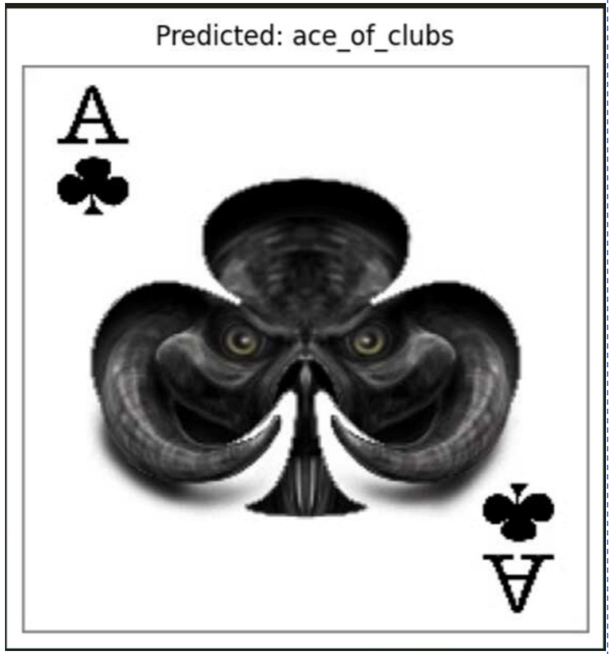
- Instead of doing attention computation over all tokens, window attention does attention computation within fixed-size (e.g., 7x7) local windows.
- The number of tokens in each window is fixed. Thus, the computational complexity is linear.
- Gradually downsample the feature map size.
- Cross window information exchange
- SViT requires large data and compute expensive





Train from transfer learning vs scratch

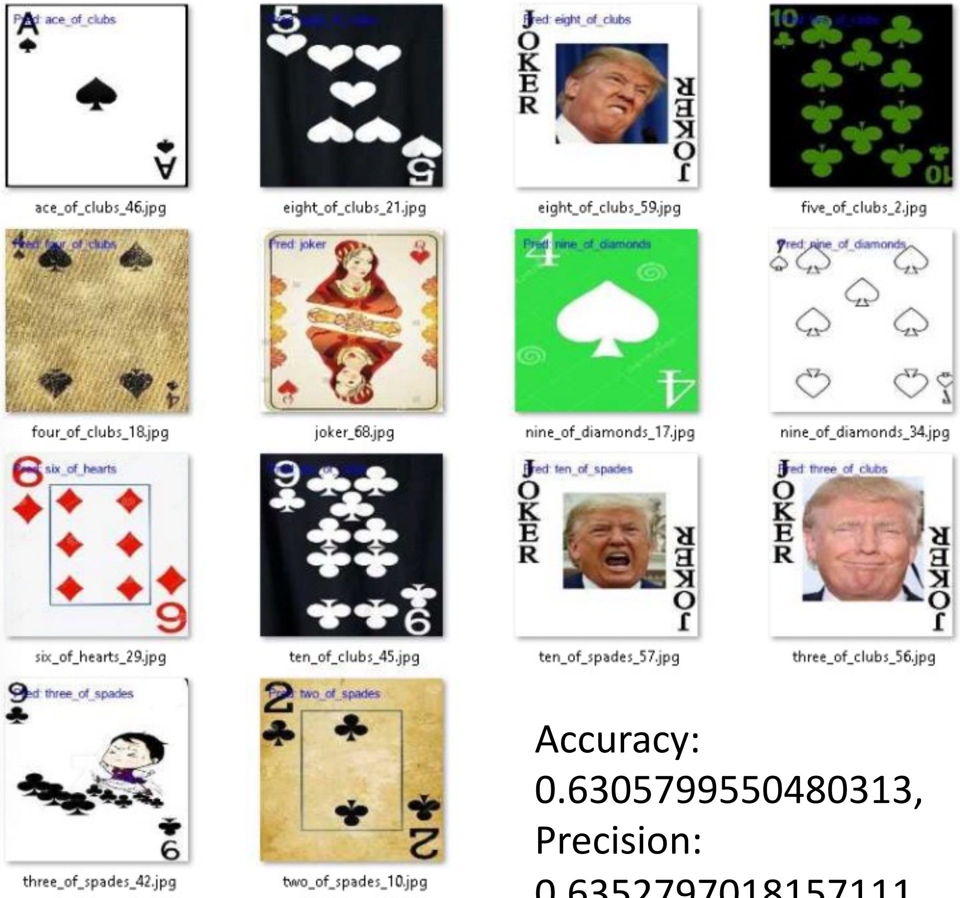
- svit_epoch_0_acc_0.0000.pt
- svit_epoch_1_acc_0.8555.pt
- svit_epoch_2_acc_0.8926.pt
- svit_epoch_3_acc_0.9552.pt
- svit_epoch_8_acc_0.9796.pt
- svit_epoch_13_acc_0.9812.pt



Accuracy: 0.982222232553694, Precision: 0.982222232553694, Recall: 0.982222232553694, F1 score: 0.982222232553694

with pretrain weight: err 0/70

Epoches: 30



Accuracy: 0.6305799550480313, Precision: 0.6352797018157111, Recall: 0.6305799550480313, F1 score: 0.6224562724431356

- svit_epoch_0_acc_0.0000.pt
- svit_epoch_1_acc_0.0619.pt
- svit_epoch_2_acc_0.1051.pt
- svit_epoch_3_acc_0.1566.pt
- svit_epoch_4_acc_0.1650.pt
- svit_epoch_5_acc_0.2104.pt
- svit_epoch_6_acc_0.2655.pt
- svit_epoch_7_acc_0.3060.pt
- svit_epoch_9_acc_0.3334.pt
- svit_epoch_10_acc_0.3558.pt
- svit_epoch_12_acc_0.4412.pt
- svit_epoch_13_acc_0.4776.pt
- svit_epoch_15_acc_0.5202.pt
- svit_epoch_20_acc_0.5293.pt
- svit_epoch_26_acc_0.5761.pt
- svit_epoch_28_acc_0.5765.pt
- svit_epoch_37_acc_0.6140.pt
- svit_epoch_43_acc_0.6168.pt
- svit_epoch_61_acc_0.6422.pt
- svit_epoch_65_acc_0.6617.pt
- svit_epoch_68_acc_0.6618.pt

from scratch: err 14/70

Epoches: 100