Swin Transformer:

Hierarchical Vision Transformer using Shifted Windows

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, Baining Guo
Microsoft Research Asia
2021. 03. 25.

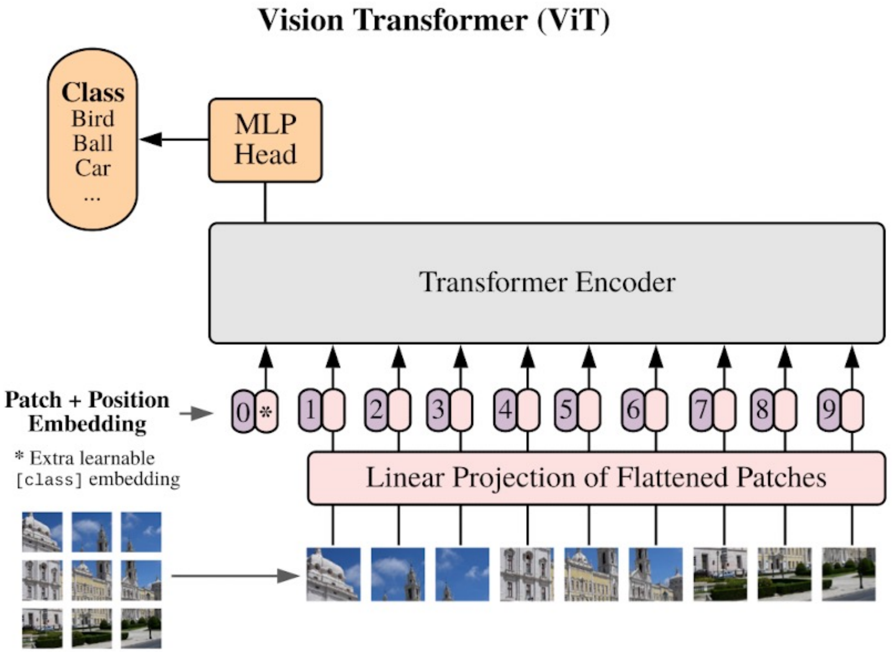
Presentation by

Jihun Kim
2022. 01. 13.

Introduction

Introduction

- Transformers has recently demonstrated promising results on certain tasks, specifically image classification and joint vision-language modeling.



Motivation

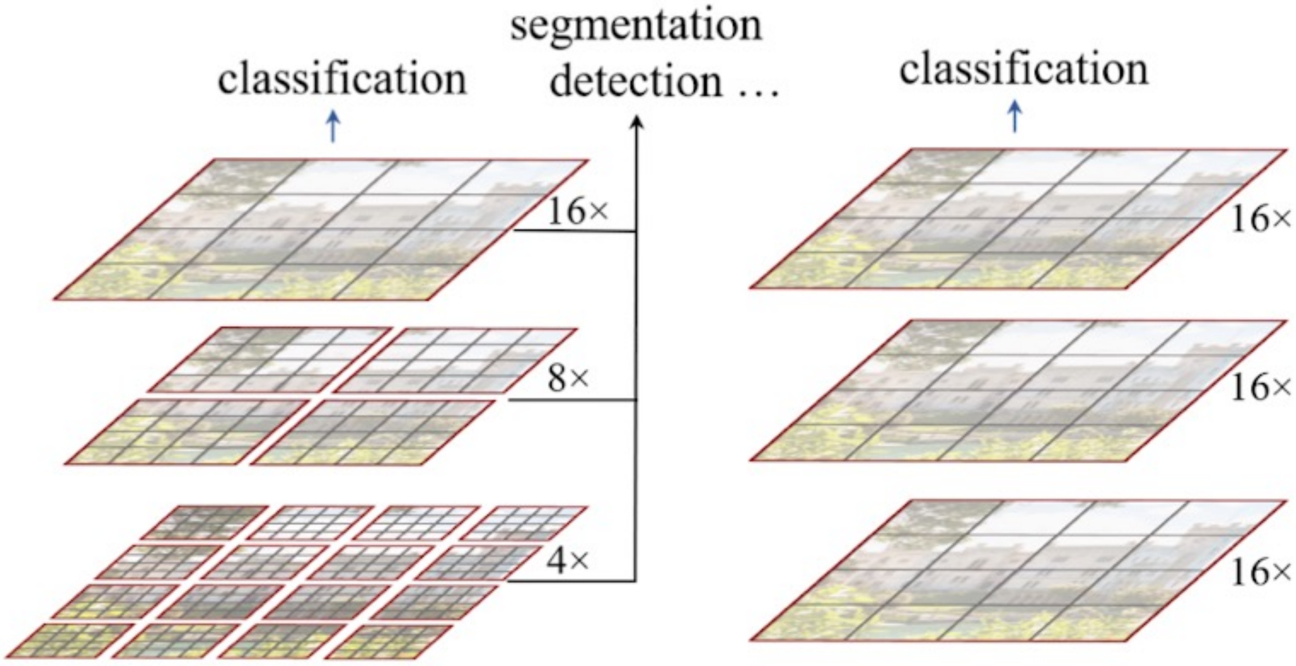
- Significant challenges in transferring its high performance in the language domain to the visual domain can be explained by differences between the two modalities.

Motivation

- Significant challenges in transferring its high performance in the language domain to the visual domain can be explained by differences between the two modalities.
- Scale
  - Visual elements can vary substantially in scale
  - In existing Transformer-based models, tokens are all of a fixed scale

Motivation

- Significant challenges in transferring its high performance in the language domain to the visual domain can be explained by differences between the two modalities.
- Scale
  - Visual elements can vary substantially in scale
  - In existing Transformer-based models, tokens are all of a fixed scale
- Much higher resolution of pixels in images
  - Dense prediction (e.g., semantic segmentation)
    requires dense prediction at the pixel level
  - This would be intractable for Transformer on high-resolution images

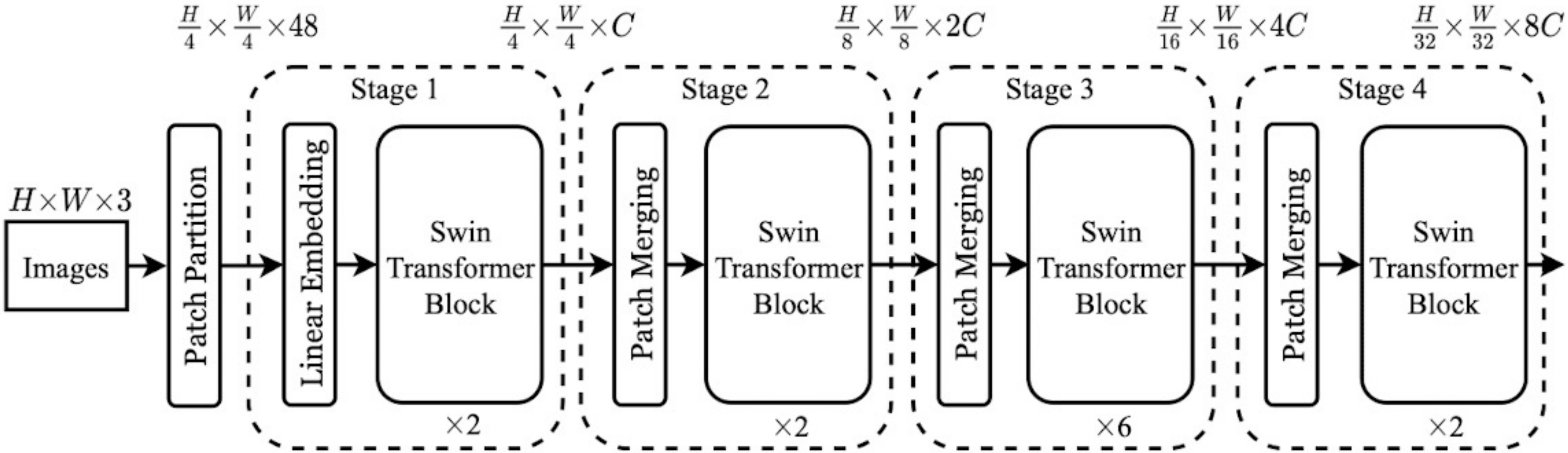


(a) Swin Transformer (ours)

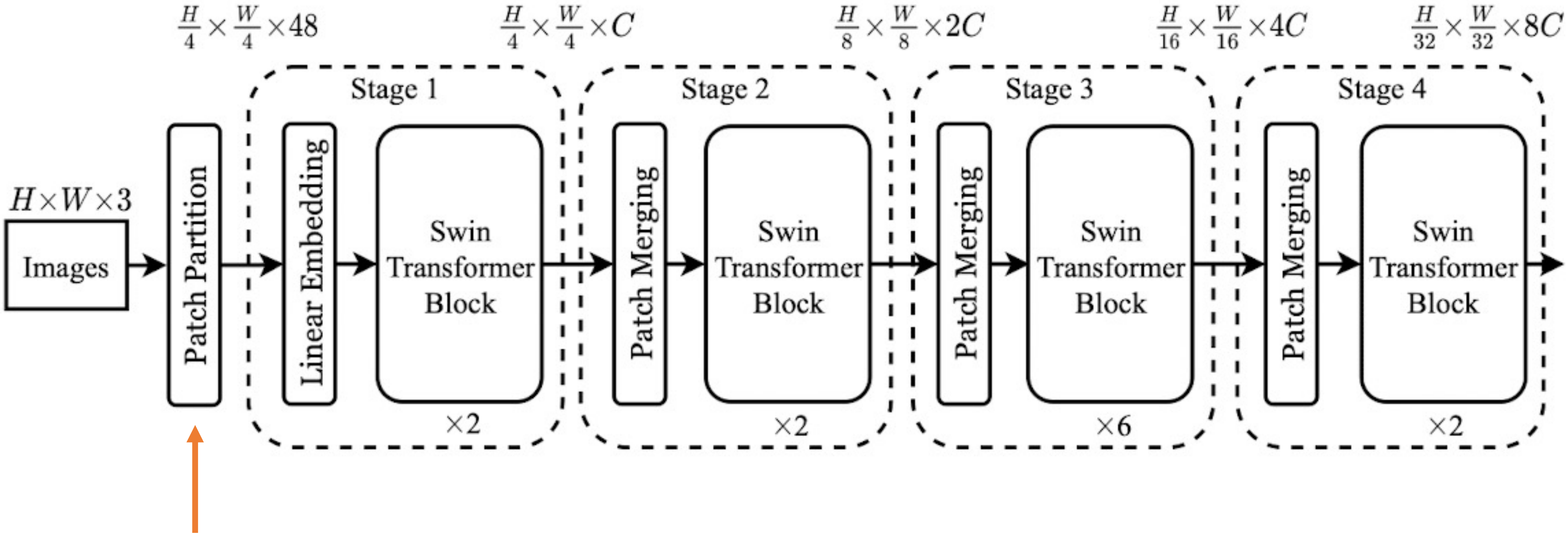
(b) ViT

Swin Transformer

Swin Transformer: Overall Architecture



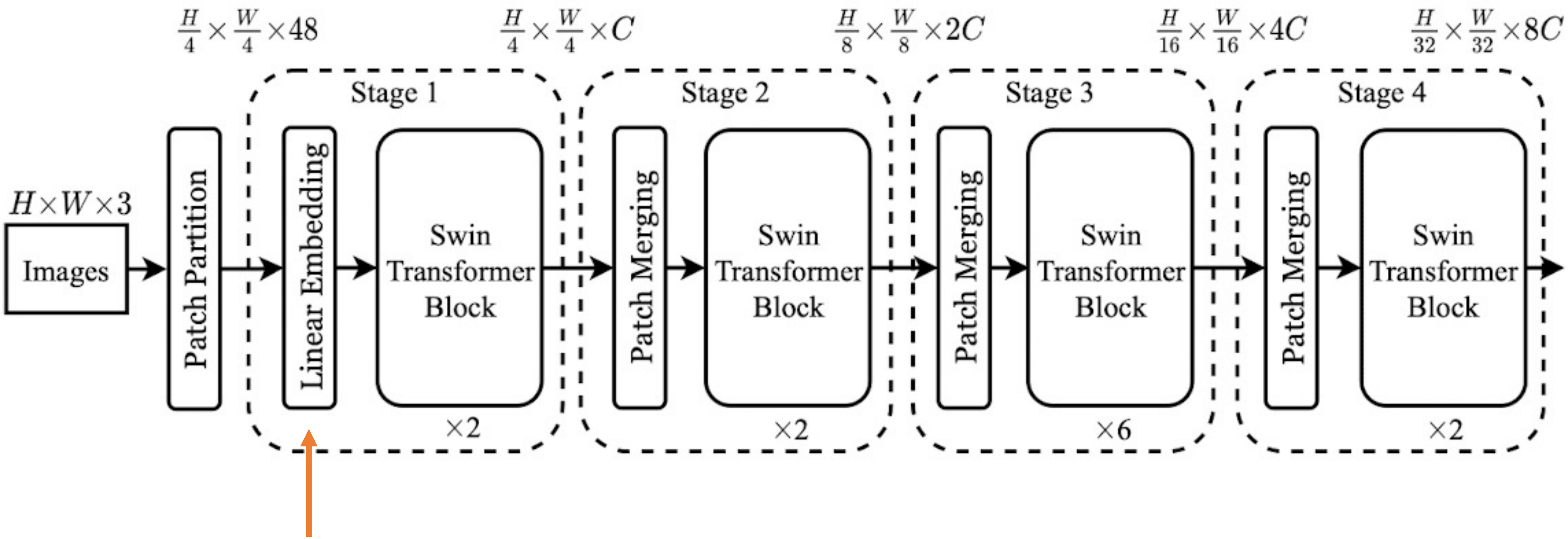
Swin Transformer: Overall Architecture



It first splits an input RGB image into non-overlapping patches.

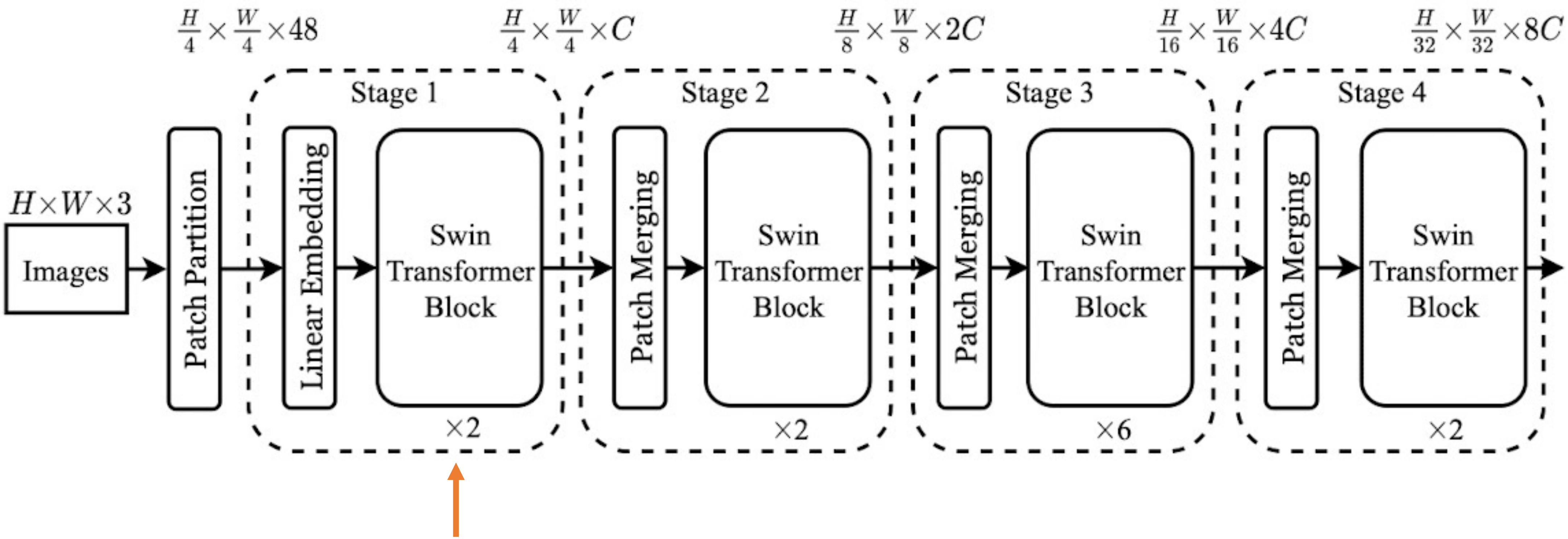
We use a patch size of 4×4 and thus the feature dimension of each patch is $4 \times 4 \times 3 = 48$.

Swin Transformer: Overall Architecture



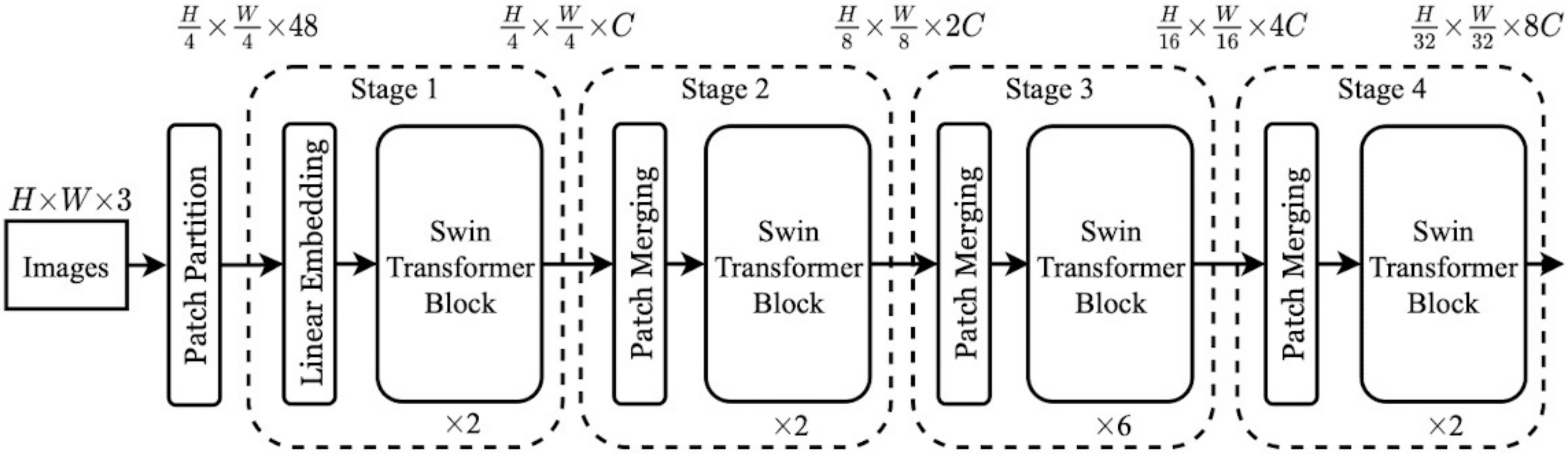
A linear embedding layer is applied on this raw-valued feature to project it to an arbitrary dimension (denoted as C).

Swin Transformer: Overall Architecture



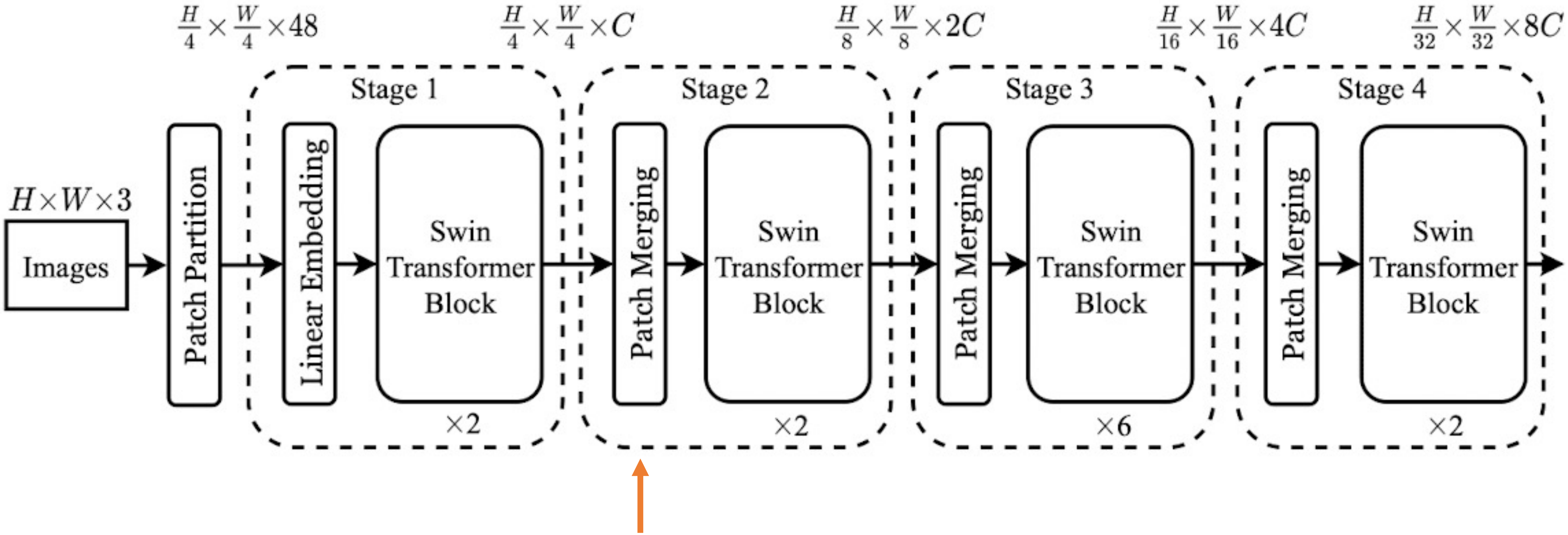
Several Transformer blocks with modified self-attention computation are applied on these patch tokens.
The Transformer blocks maintain the number of tokens ($\frac{H}{4}\times\frac{W}{4}$).

Swin Transformer: Overall Architecture



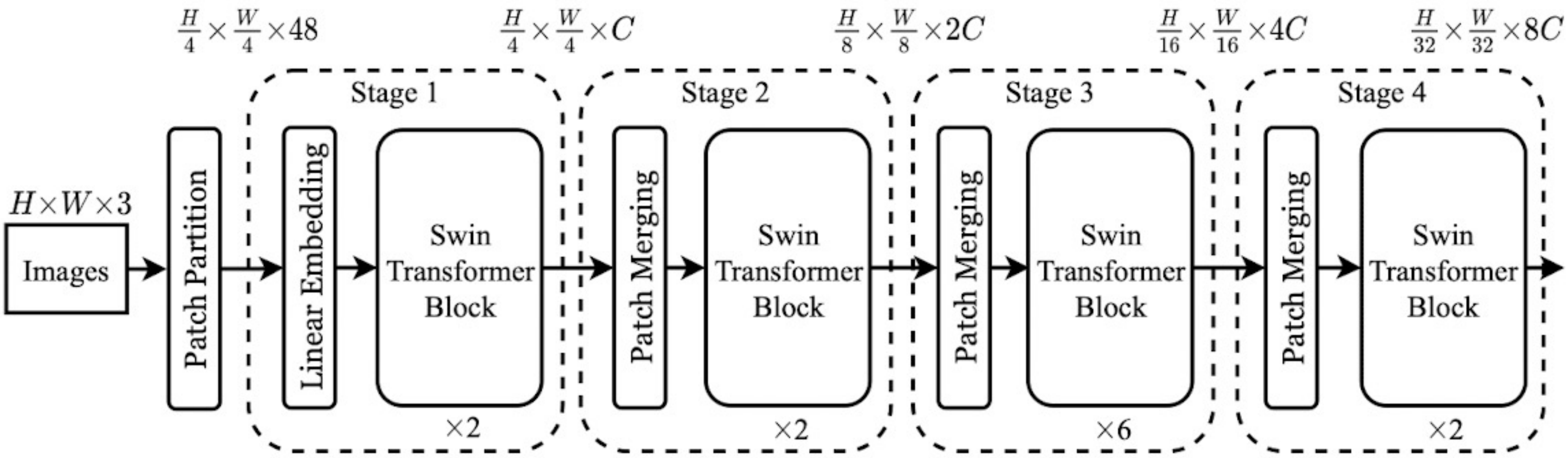
To produce a hierarchical representation,
the number of tokens is reduced by patch merging layers as the network gets deeper.

Swin Transformer: Overall Architecture



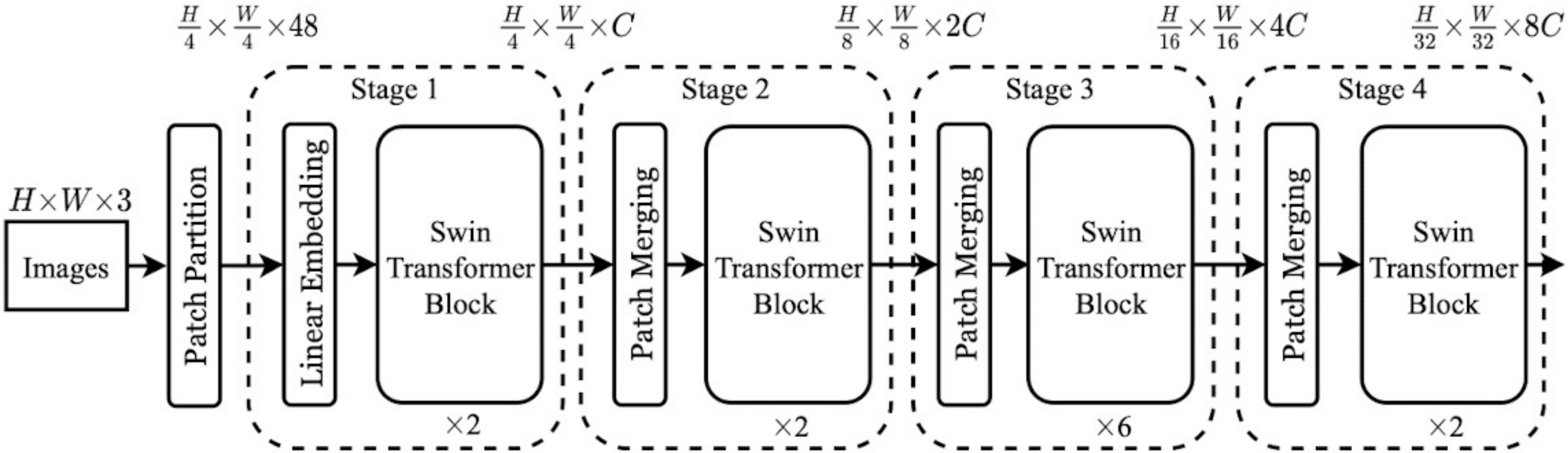
Patch merging layer concatenates the features of each group of 2×2 neighboring patches, and applies a linear layer on the $4C$-dimensional concatenated features.
The output dimension is set to $2C$.

Swin Transformer: Overall Architecture



And so on..

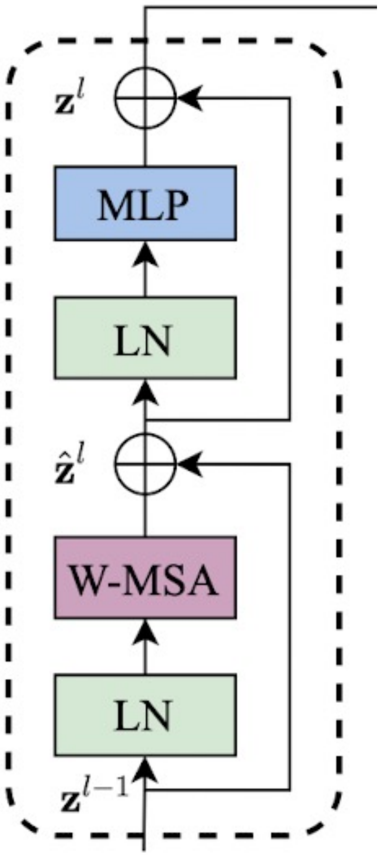
Swin Transformer: Overall Architecture



These stages jointly produce a hierarchical representation.
As a result, the proposed architecture can conveniently replace the backbone networks in existing methods for various vision tasks.

Swin Transformer Block

- Swin Transformer is built by replacing the standard multi-head self attention (MSA) module
  in a Transformer block,
  with other layers kept the same.



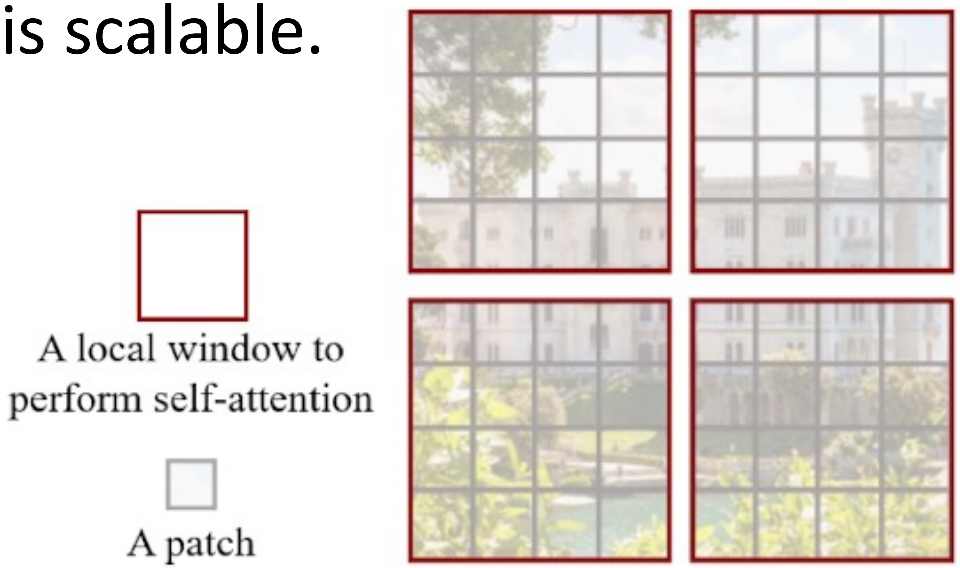
Shifted Window based Self-Attention
(W-MSA Block)

Shifted Window based Self-Attention

- The standard Transformer architecture
  and its adaptation for image classification (ViT)
  both conduct global self-attention.
- The global computation leads to quadratic complexity
  with respect to the number of tokens,
  making it unsuitable for many vision problems.

Shifted Window based Self-Attention

- For efficient modeling,
  we propose to compute self-attention within local windows.
- Global self-attention computation is generally unaffordable for a large number of patches,
  while the window based self-attention is scalable.

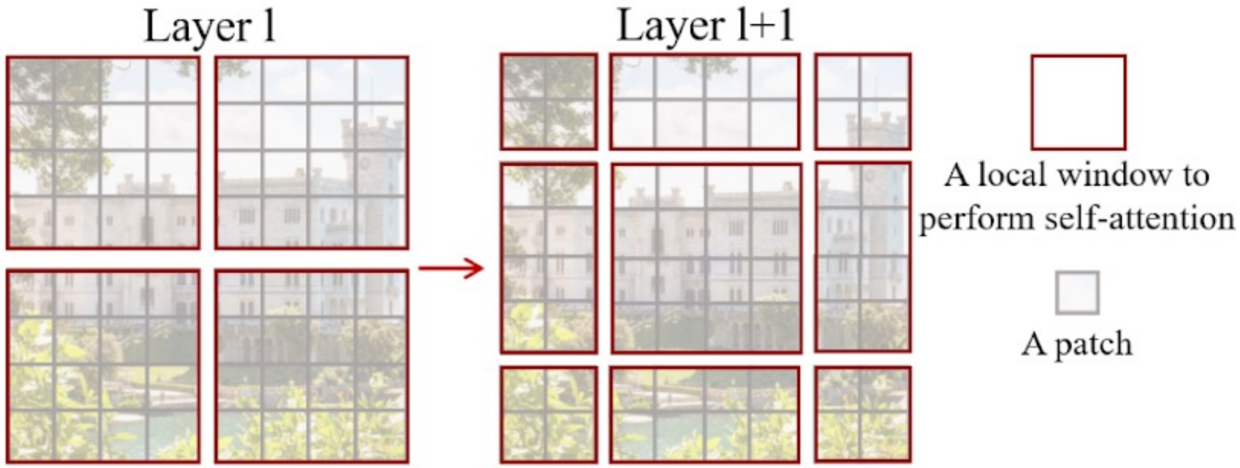


Shifted Window based Self-Attention

- Why not using sliding window?
- This strategy is efficient in regards to real-world latency.
  - All query patches within a window share the same key set, which facilitates memory access in hardware.
- In contrast, sliding window based self-attention approaches suffer from low latency on general hardware.
  - Due to different key sets for different query pixels.

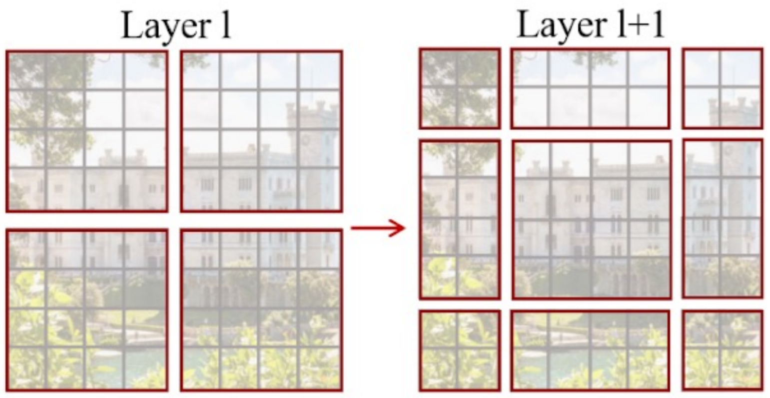
Shifted Window based Self-Attention

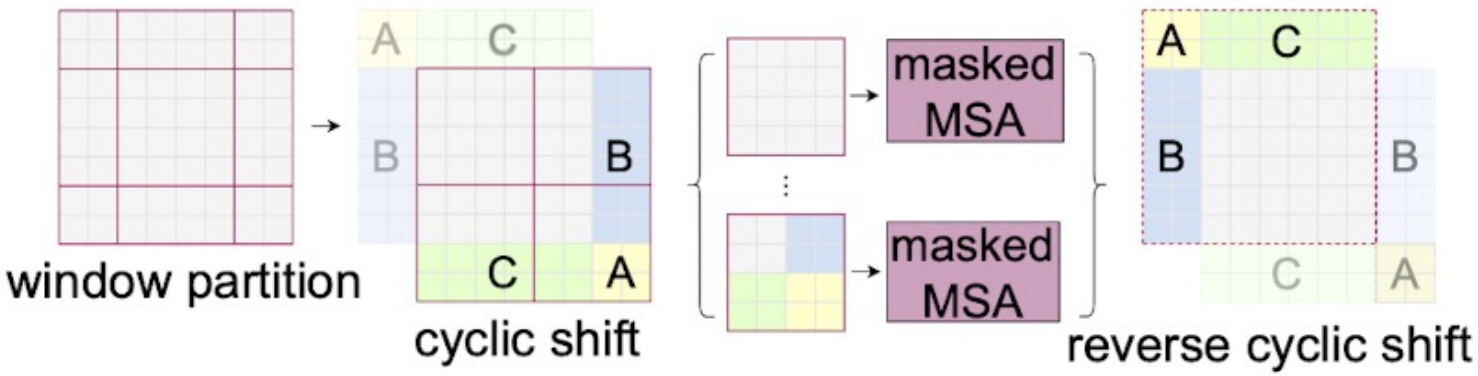
- The window-based self-attention module lacks connections across windows, which limits its modeling power.
- We propose a shifted window partitioning approach which alternates between two partitioning configurations in consecutive Swin Transformer blocks.



Shifted Window based Self-Attention

- Shifted window partitioning will result in more windows.
- We propose a more efficient batch computation approach
  by cyclic-shifting toward the top-left direction.
- A masking mechanism is employed
  to limit self-attention computation to within each sub-window.





Shifted Window based Self-Attention

- In computing self-attention, include a relative position bias $B \in \mathbb{R}^{M^2 \times M^2}$ to each head in computing similarity:

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

$Q, K, V \in \mathbb{R}^{M^2 \times d}$	query, key and value matrices
d	query/key dimension
M^2	number of patches in a window

Results

Image Classification: Regular ImageNet-1K training

ConvNets

Transformer based

(a) Regular ImageNet-1K trained models

method	image size	#param.	FLOPs	throughput (image / s)	ImageNet top-1 acc.
RegNetY-4G [48]	224^2	21M	4.0G	1156.7	80.0
RegNetY-8G [48]	224^2	39M	8.0G	591.6	81.7
RegNetY-16G [48]	224^2	84M	16.0G	334.7	82.9
EffNet-B3 [58]	300^2	12M	1.8G	732.1	81.6
EffNet-B4 [58]	380^2	19M	4.2G	349.4	82.9
EffNet-B5 [58]	456^2	30M	9.9G	169.1	83.6
EffNet-B6 [58]	528^2	43M	19.0G	96.9	84.0
EffNet-B7 [58]	600^2	66M	37.0G	55.1	84.3
ViT-B/16 [20]	384^2	86M	55.4G	85.9	77.9
ViT-L/16 [20]	384^2	307M	190.7G	27.3	76.5
DeiT-S [63]	224^2	22M	4.6G	940.4	79.8
DeiT-B [63]	224^2	86M	17.5G	292.3	81.8
DeiT-B [63]	384^2	86M	55.4G	85.9	83.1
Swin-T	224^2	29M	4.5G	755.2	81.3
Swin-S	224^2	50M	8.7G	436.9	83.0
Swin-B	224^2	88M	15.4G	278.1	83.5
Swin-B	384^2	88M	47.0G	84.7	84.5

- vs. Transformer-based
  - Surpass counterpart DeiT archs with similar complexities.
- vs. Conv-based
  - Slightly better speed-accuracy trade-off.
  - These ConvNets are obtained via a thorough arch. search, while Swin Transformers are not.

Image Classification: ImageNet-22K pre-training

(b) ImageNet-22K pre-trained models					
method	image size	#param.	FLOPs	throughput (image / s)	ImageNet top-1 acc.
R-101x3 [38]	384^2	388M	204.6G	-	84.4
R-152x4 [38]	480^2	937M	840.5G	-	85.4
ViT-B/16 [20]	384^2	86M	55.4G	85.9	84.0
ViT-L/16 [20]	384^2	307M	190.7G	27.3	85.2
Swin-B	224^2	88M	15.4G	278.1	85.2
Swin-B	384^2	88M	47.0G	84.7	86.4
Swin-L	384^2	197M	103.9G	42.1	87.3

- Significantly better speed-accuracy trade-offs, compared with the previous best results.

Object Detection on COCO

(a) Various frameworks								
Method	Backbone	AP^{box}	AP^{box}_{50}	AP^{box}_{75}	#param.	FLOPs	FPS
Cascade Mask R-CNN	R-50	46.3	64.3	50.5	82M	739G	18.0
	Swin-T	**50.5**	**69.3**	**54.9**	86M	745G	15.3
ATSS	R-50	43.5	61.9	47.0	32M	205G	28.3
	Swin-T	**47.2**	**66.5**	**51.3**	36M	215G	22.3
RepPointsV2	R-50	46.5	64.6	50.3	42M	274G	13.6
	Swin-T	**50.0**	**68.5**	**54.2**	45M	283G	12.0
Sparse R-CNN	R-50	44.5	63.4	48.2	106M	166G	21.0
	Swin-T	**47.9**	**67.3**	**52.3**	110M	172G	18.4

- vs. ResNet-50 backbone
  - consistent +3.4~4.2 box AP gains,
    with slightly larger model size, FLOPs and latency.

Object Detection on COCO

(b) Various backbones w. Cascade Mask R-CNN	AP^{box}	AP^{box}_{50}	AP^{box}_{75}	AP^{mask}	AP^{mask}_{50}	AP^{mask}_{75}	param	FLOPs	FPS
DeiT-S†	48.0	67.2	51.7	41.4	64.2	44.3	80M	889G	10.4
R50	46.3	64.3	50.5	40.1	61.7	43.4	82M	739G	18.0
Swin-T	**50.5**	**69.3**	**54.9**	**43.7**	**66.6**	**47.1**	86M	745G	15.3
X101-32	48.1	66.5	52.4	41.6	63.9	45.2	101M	819G	12.8
Swin-S	**51.8**	**70.4**	**56.3**	**44.7**	**67.9**	**48.5**	107M	838G	12.0
X101-64	48.3	66.4	52.3	41.7	64.0	45.1	140M	972G	10.4
Swin-B	**51.9**	**70.9**	**56.5**	**45.0**	**68.4**	**48.7**	145M	982G	11.6

- vs. ResNeXt-101
  - Higher detection accuracy
    with similar model size, FLOPs, and latency.
  - Note that ResNext is built by highly optimized Cudnn functions.

Object Detection on COCO

(b) Various backbones w. Cascade Mask R-CNN

	AP^{box}	AP^{box}_{50}	AP^{box}_{75}	AP^{mask}	AP^{mask}_{50}	AP^{mask}_{75}	param	FLOPs	FPS
DeiT-S†	48.0	67.2	51.7	41.4	64.2	44.3	80M	889G	10.4
R50	46.3	64.3	50.5	40.1	61.7	43.4	82M	739G	18.0
Swin-T	**50.5**	**69.3**	**54.9**	**43.7**	**66.6**	**47.1**	86M	745G	15.3
X101-32	48.1	66.5	52.4	41.6	63.9	45.2	101M	819G	12.8
Swin-S	**51.8**	**70.4**	**56.3**	**44.7**	**67.9**	**48.5**	107M	838G	12.0
X101-64	48.3	66.4	52.3	41.7	64.0	45.1	140M	972G	10.4
Swin-B	**51.9**	**70.9**	**56.5**	**45.0**	**68.4**	**48.7**	145M	982G	11.6

- vs. DeiT
  - Higher accuracy, and significantly higher inference speed.
  - The lower inference speed of DeiT is mainly due to its quadratic complexity to input image size.