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

Members : Jongmin-Woo, Bongwon-Jang, Chanhyun-Jung

INDEX

1. Each Members' Task

2. Simple summary of Swin

3. Method

4 Experiments

1. Each Members' Task

Jongmin-Woo : Paper Review
Speech

Bongwon-Jang : Code Review
Speech

Changhyung-Jeong : Write Jupyter Notebook & Run Experiment

2. Simple Summary of Swin

Linear Computational Complexity by Image Size

Use Shifted Window, Self Attention

Achieve better accuracy-time tradeoff

Make General Purpose Backbone. (for classification, object detection, segmentation ...)

2. Simple Summary

Merge Related Works' merit

CNN

- Good at find local feature

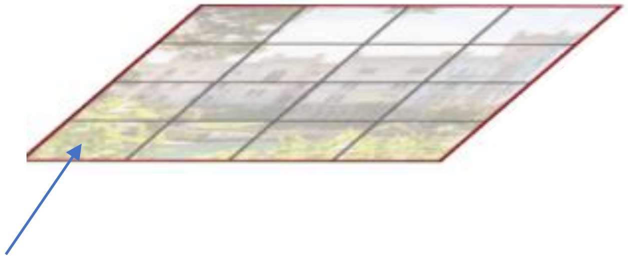
Transformer-Attention

- Good at find distant dependency

+ Shifted Windows

3. Method – Window to patch. To token

- First. Split RGB image into non-overlapping patches



patch

This example shows 16 patches
Each patch is treated as a token.
Token is concatenation raw pixel RGB values.

4

RGB	RGB	RGB	RGB
RGB	RGB	RGB	RGB
RGB	RGB	RGB	RGB
RGB	RGB	RGB	RGB

4
patch

RGBRGBRGBRGBRGBRGBRGBRGBRGBRGBRGBRGBRGBRGBRGBRGB

Token (patch width x patch height x channels)
Token's Length : 4 x 4 x 3 = 48
Token projected arbitrary dimension denoted as C

3. Method – Token to Arbitrary dimension

RGBRGBRGBRGBRGBRGBRGBRGBRGBRGBRGBRGBRGBRGBRGBRGB

Token length : Patch height x Patch width x Channels

project

FeatureFeatureFeatureFeatureFeatureFeatureFeatureFeature

Projected dimension Size : C

3. Method – Patch merging layer

Concat
2x2 Neighboring patches.

featuresfeaturesfeaturesfeaturesfeaturesfeaturesfeaturesfeatures

featuresfeaturesfeaturesfeaturesfeaturesfeaturesfeaturesfeatures

featuresfeaturesfeaturesfeaturesfeaturesfeaturesfeaturesfeatures

featuresfeaturesfeaturesfeaturesfeaturesfeaturesfeaturesfeatures

Patch Merging

Merged Patch's Length : 4C

PreviousFeatures…………PreviousFeatures

Down sampling

2C Dimensional Features

FeaturesFeaturesFeaturesFeatures

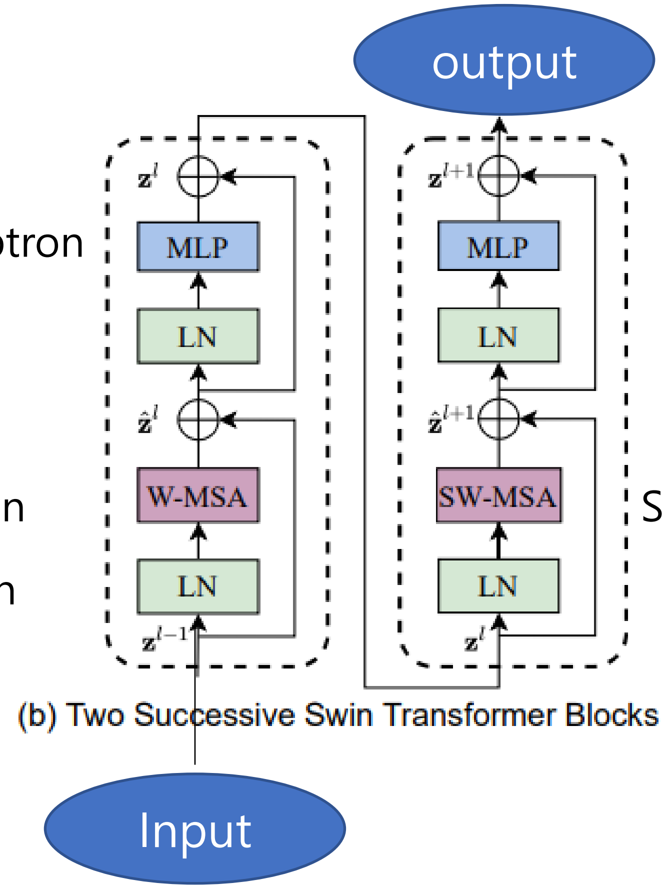
Length : 2C

3. Method – Swin Transformer blocks

Multiple Layer Perceptron

Window based Multi-head Self Attention

Layer Normalization



(b) Two Successive Swin Transformer Blocks

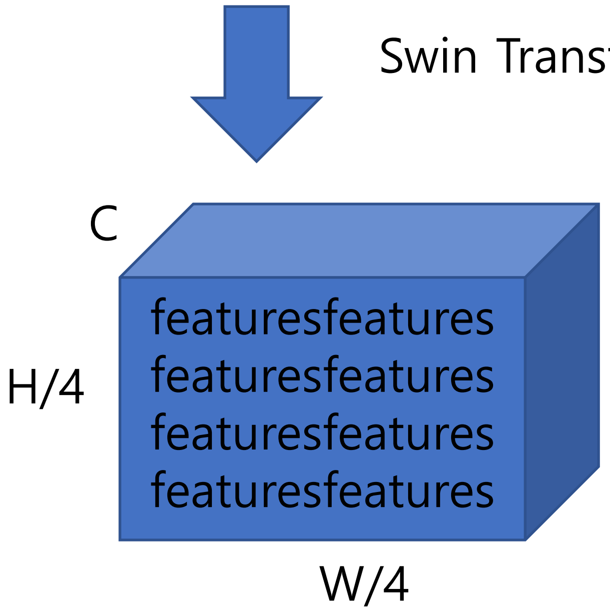
Shifted Window based Multi-head Self Attention

down-sampled features

3. Method – Swin Transformer blocks

2C Dimensional Features

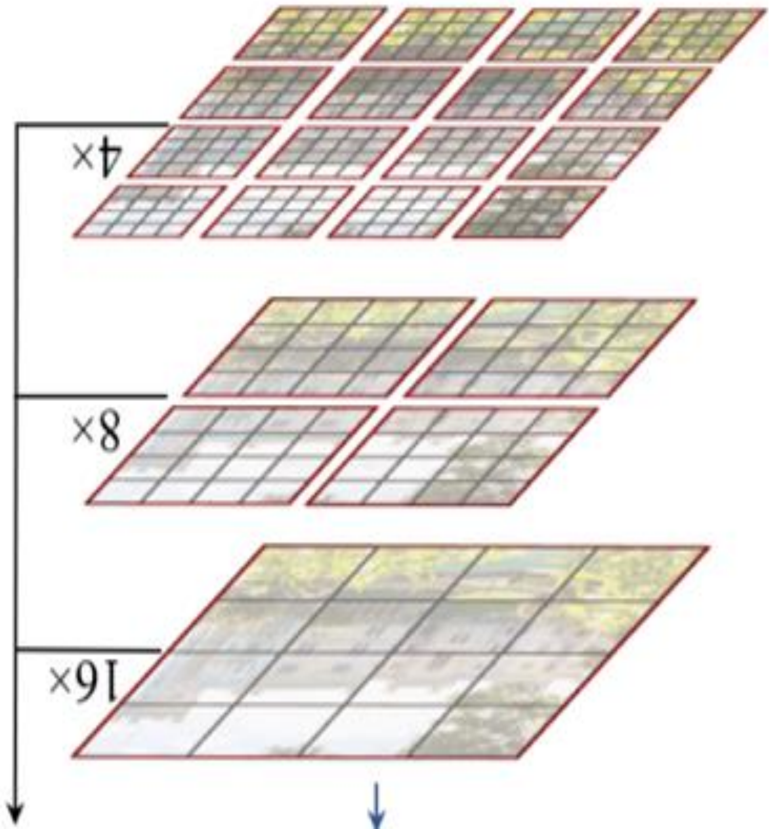
FeaturesFeaturesFeaturesFeaturesFeaturesFeaturesFeaturesFeatures



Swin Transformer block

Denotes this algorithm block → Stage 1

3. Method – stage name at Repeating



Input image (H x W)

token

After swin transformer

Patch split
Patch Merging
Swin Transformer. Repeat n times.

Stage 1 (H/4 x W/4)

Patch split

token

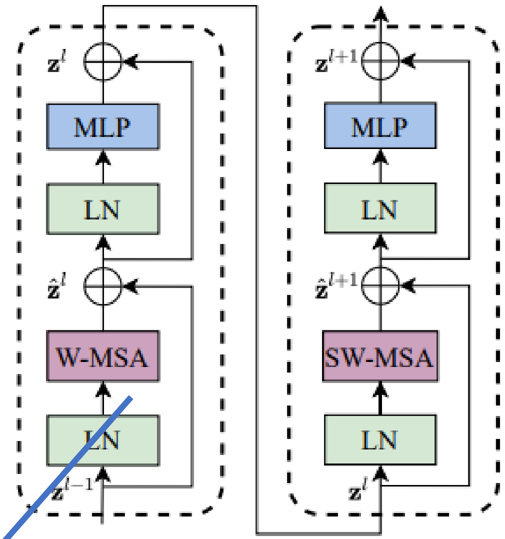
After swin transformer

Stage 2 (H/8 x W/8)

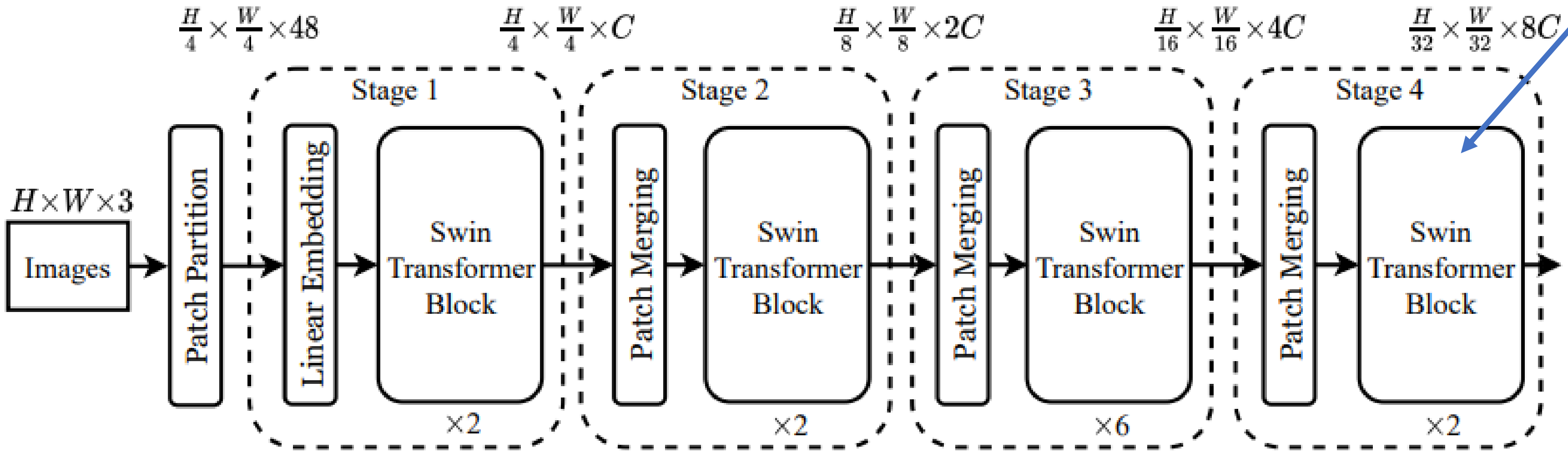
Swin Transformer. Repeat m times

Stage 3 (H/16 x W/16)

3. Method - Repeat



(b) Two Successive Swin Transformer Blocks



(a) Architecture

3. Method – problem of global self attention (computational complexity)

Input feature's shape : (h, w, C)

Q K V	QK^T	Softmax(QK^T) · V	Attention value
input Projection matrix (hw x C) · (C x C)	Q K^T (hw x C) · (C x hw)	QK^T V (hw x hw) · (hw x C)	QK^TV → projection matrix (hw x C) · (C x C)
Shape : (hw x C)	Shape : (hw x hw)	Shape : (hw x C)	Shape : (hw x C)
Ω(step1) : 3 * $(hw)C^2$	Ω(step2) : $(hw)^2C$	Ω(step3) : $(hw)^2C$	Ω(step4) : $(hw)\ C^2$

Total computational Complexity (summation of all step)

→ $\Omega(MSA) : 3(hw)C^2 + (hw)^2C + (hw)^2C + (hw)C^2$

$\Omega(MSA) : 4(hw)C^2 + 2(hw)^2C$

3. Method – Non-overlapped Windows

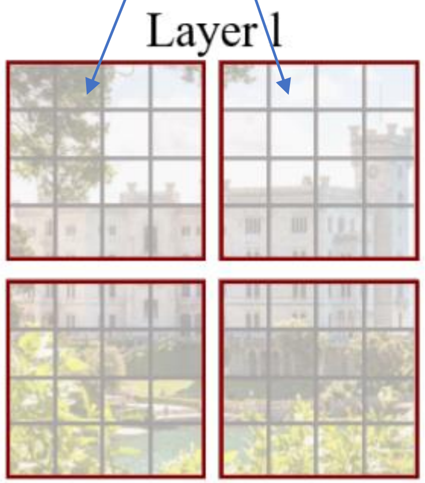
If patch size is M x M.
The number ov patches : $h/M \times w/M$

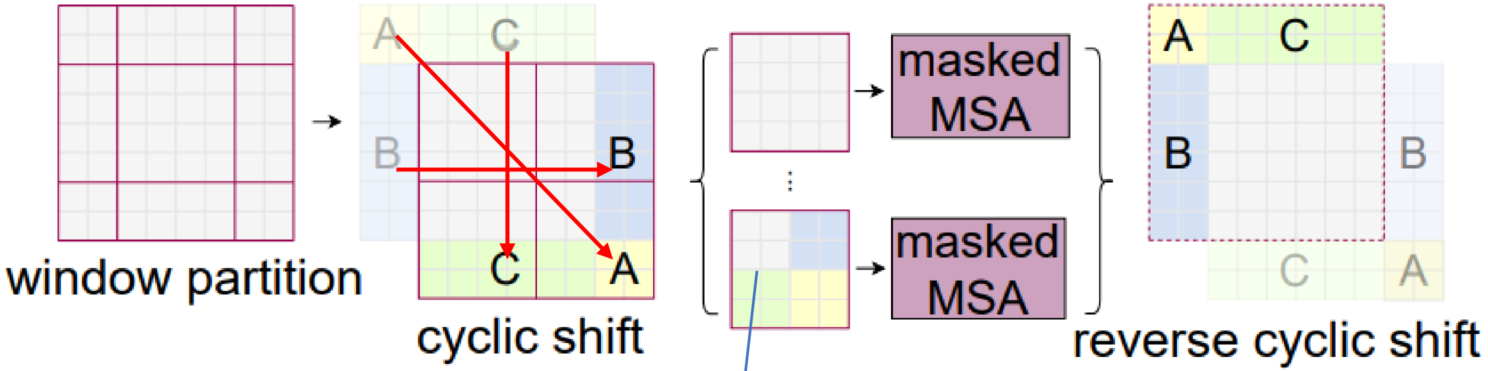
Q K V	QK^T	$\mathrm{Softmax}(QK^T) \cdot V$	Attention value
input Projection matrix $(M^2 \times C) \cdot (C \times C)$	Q K^T $(M^2 \times C) \cdot (C \times M^2)$	QK^T V $(M^2 \times M^2) \cdot (M^2 \times C)$	QK^TV → projection matrix $(M^2 \times C) \cdot (C \times C)$
Shape : $(M^2 \times C)$	Shape : $(M^2 \times M^2)$	Shape : $(M^2 \times C)$	Shape : $(M^2 \times C)$
Ω(step1) : $(h/M * w/M) * 3 * (M^2)C^2$ $= 3 * hwC^2$	Ω(step2) : $(h/M * w/M) * ((M^2)^2)C$ $=hw* M^2C$	Ω(step3) : $(h/M * w/M) * ((M^2)^2) C$ $=hw* M^2C$	Ω(step4) : $(h/M * w/M) * M^2 C^2$ $=hwC^2$

Total computational Complexity (summation of all step)
→ $\Omega(MSA) : 3 * hwC^2 + hw* M^2C + hw* M^2C + hwC^2$
$\Omega(MSA) : 4(hw)C^2 + 2M^2(hw)C$

3. Method – Cyclic Shift

Lack of connection between Windows





Return to original image

Originally this regions are not adjacent.
This region's features must not be globalized.

3. Method – Architecture Variants

Token length

Stage {1, 2, 3, 4} layer repeat number

- 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 =$\{2, 2, 18, 2\}$

4. Experiments

Dataset : ImageNet-1K
Task : Classification

method	image size	#param.	FLOPs	throughput (image / s)	ImageNet top-1 acc.
(a) Regular ImageNet-1K trained models					
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
(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

Table 1. Comparison of different backbones on ImageNet-1K classification. Throughput is measured using the GitHub repository of [68] and a V100 GPU, following [63].

4. Experiments

Dataset : COCO
Task : Object Detection

(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

(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

(c) System-level Comparison

Method	mini-val		test-dev		#param.	FLOPs
	AP^{box}	AP^{mask}	AP^{box}	AP^{mask}		
RepPointsV2* [12]	-	-	52.1	-	-	-
GCNet* [7]	51.8	44.7	52.3	45.4	-	1041G
RelationNet++* [13]	-	-	52.7	-	-	-
SpineNet-190 [21]	52.6	-	52.8	-	164M	1885G
ResNeSt-200* [78]	52.5	-	53.3	47.1	-	-
EfficientDet-D7 [59]	54.4	-	55.1	-	77M	410G
DetectoRS* [46]	-	-	55.7	48.5	-	-
YOLOv4 P7* [4]	-	-	55.8	-	-	-
Copy-paste [26]	55.9	47.2	56.0	47.4	185M	1440G
X101-64 (HTC++)	52.3	46.0	-	-	155M	1033G
Swin-B (HTC++)	56.4	49.1	-	-	160M	1043G
Swin-L (HTC++)	57.1	49.5	57.7	50.2	284M	1470G
Swin-L (HTC++)*	**58.0**	**50.4**	**58.7**	**51.1**	284M	-

Table 2. Results on COCO object detection and instance segmentation. †denotes that additional decovolution layers are used to produce hierarchical feature maps. * indicates multi-scale testing.

4. Experiments – Execution Time.

method	MSA in a stage (ms)				Arch. (FPS)		
	S1	S2	S3	S4	T	S	B
sliding window (naive)	122.5	38.3	12.1	7.6	183	109	77
sliding window (kernel)	7.6	4.7	2.7	1.8	488	283	187
Performer [14]	4.8	2.8	1.8	1.5	638	370	241
window (w/o shifting)	2.8	1.7	1.2	0.9	770	444	280
shifted window (padding)	3.3	2.3	1.9	2.2	670	371	236
shifted window (cyclic)	3.0	1.9	1.3	1.0	755	437	278

Table 5. Real speed of different self-attention computation methods and implementations on a V100 GPU.