Swin Transformer: Hierarchical Vision Transformer using Shifted Windows

Ze Liu, Yutong Lin, Yue Cao, Han Hu

2021

임동주

목차

Introduction

Method

1. Patch Partition
2. Patch Merging
3. Shifted Window based Self-Attention

Experiments

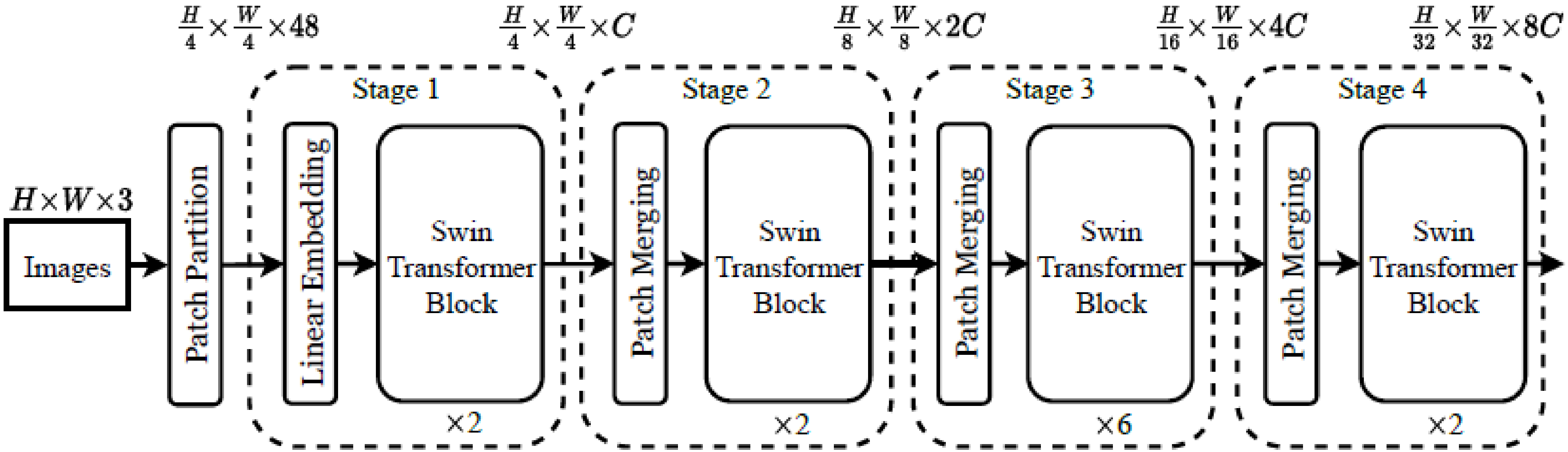
Q&A

Introduction

- ViT의 약점
  - Scale
    - NLP: 크기와 Scale이 일정
    - **Image: 크기와 Scale이 다양**
  - High Resolution
    - **이미지의 해상도**가 높아질수록 **연산량 급증**
    - 해상도 증가량의 **제곱의 비율**로 증가

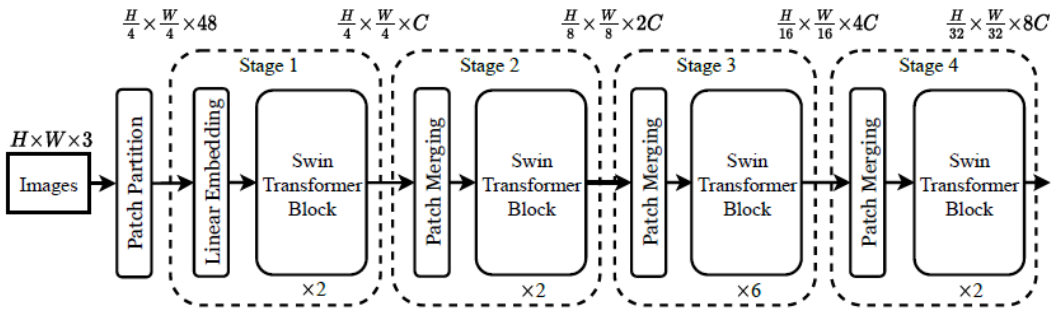
- Expectation
  - Visual Domain에서의 General-Purpose Backbone 모델

Model Architecture

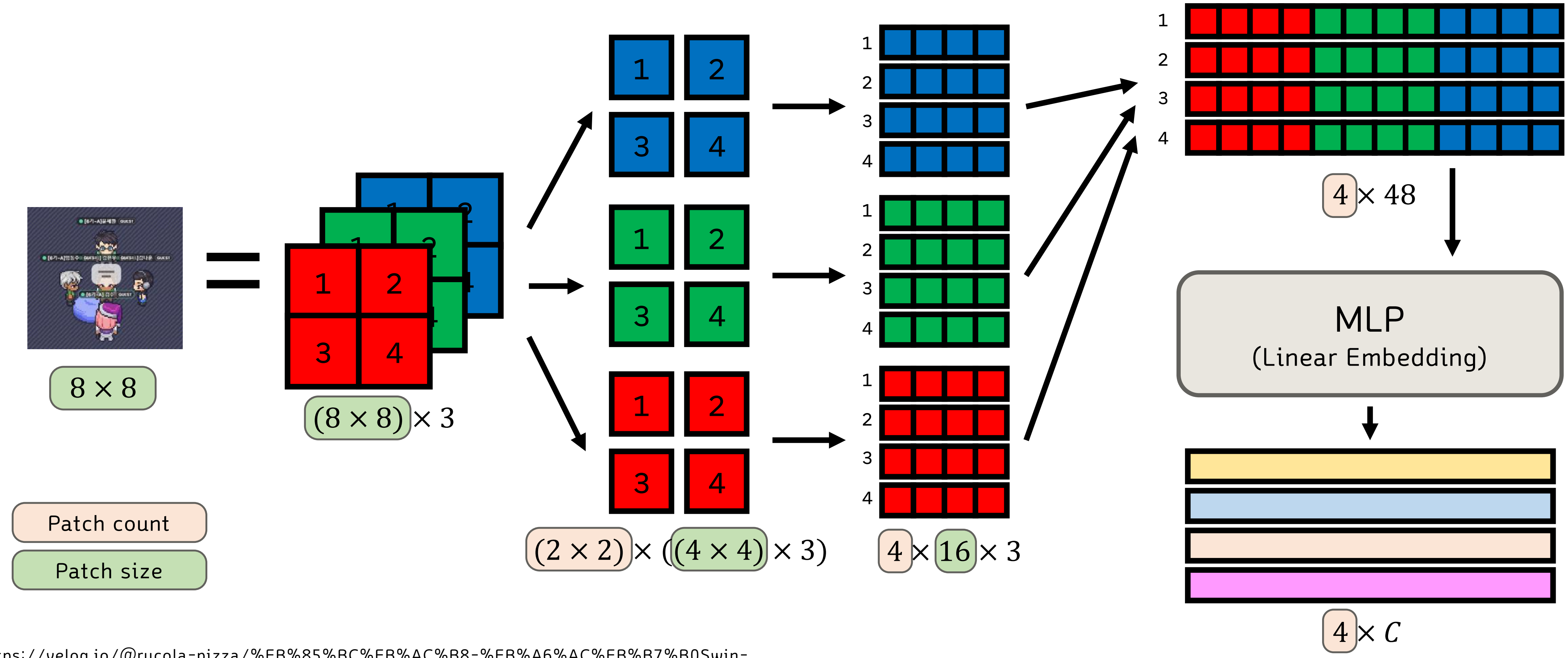


Model Architecture

- 간단한 Patch Partition



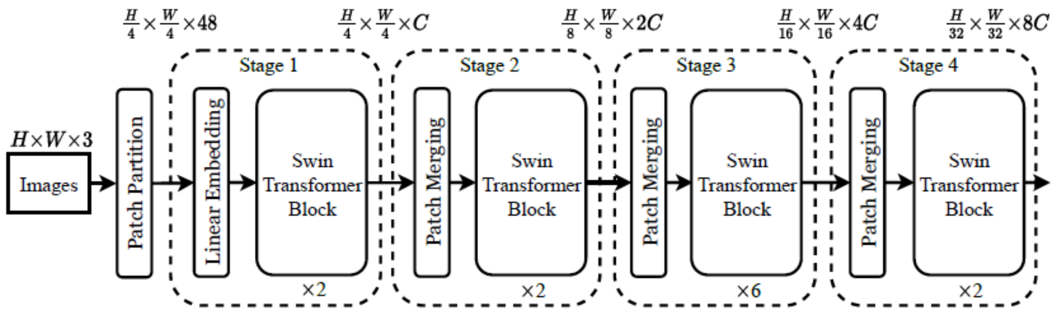
(a) Architecture



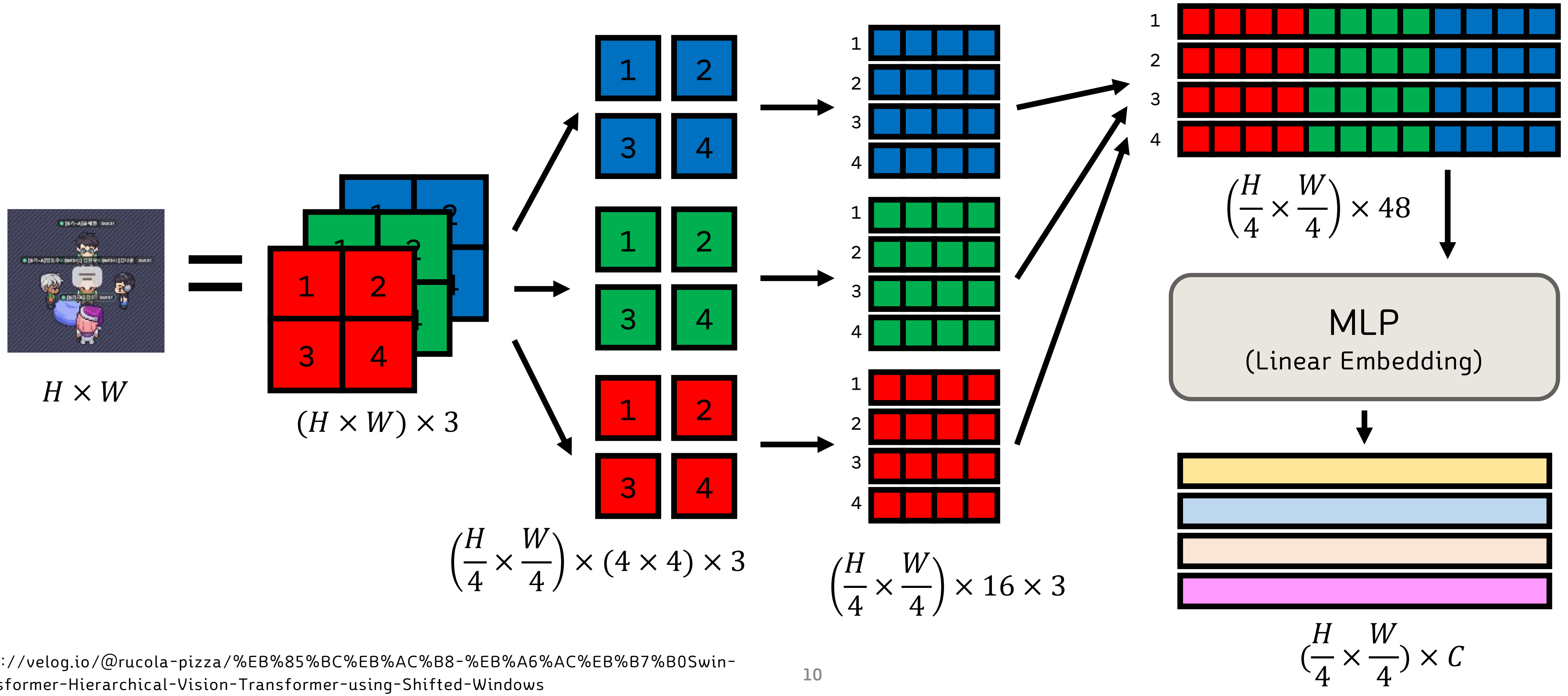
https://velog.io/@rucola-pizza/%EB%85%BC%EB%AC%B8-%EB%A6%AC%EB%B7%B0Swin-Transformer-Hierarchical-Vision-Transformer-using-Shifted-Windows

Model Architecture

- Patch Partition의 일반화



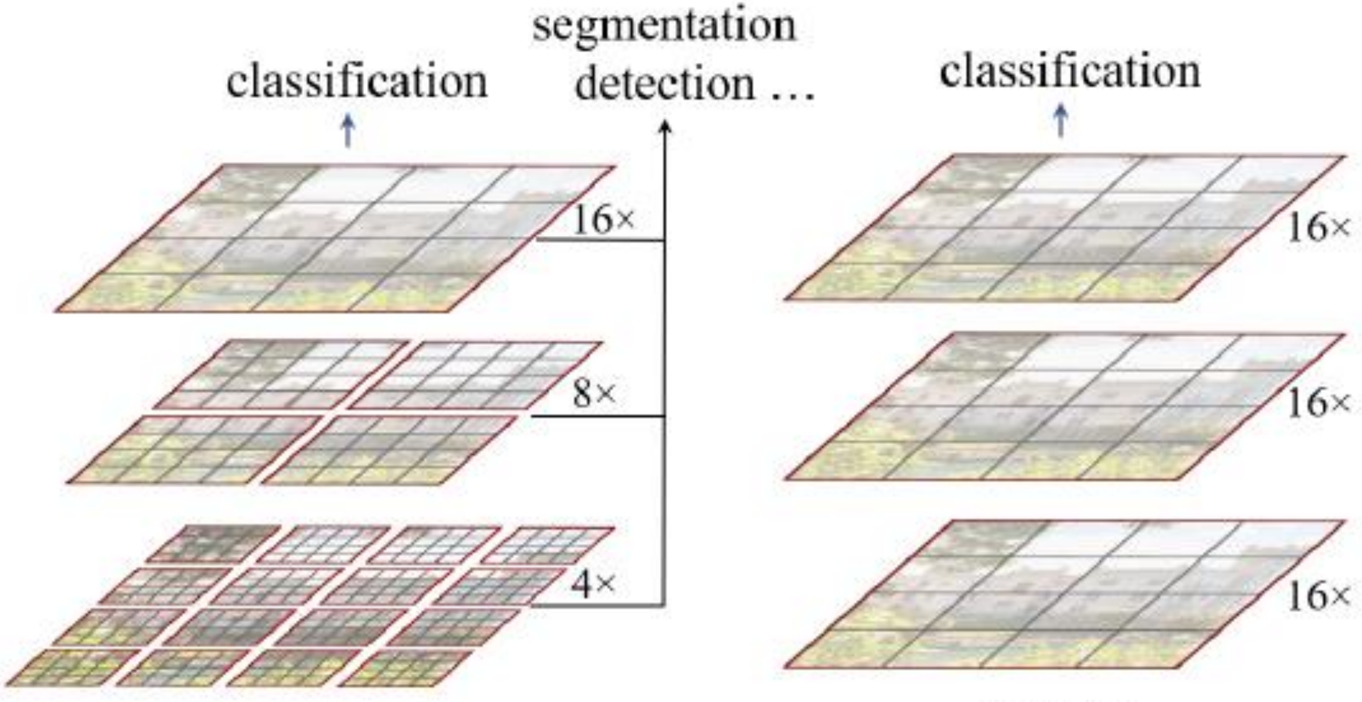
(a) Architecture



https://velog.io/@rucola-pizza/%EB%85%BC%EB%AC%B8-%EB%A6%AC%EB%B7%B0Swin-Transformer-Hierarchical-Vision-Transformer-using-Shifted-Windows

Model Architecture

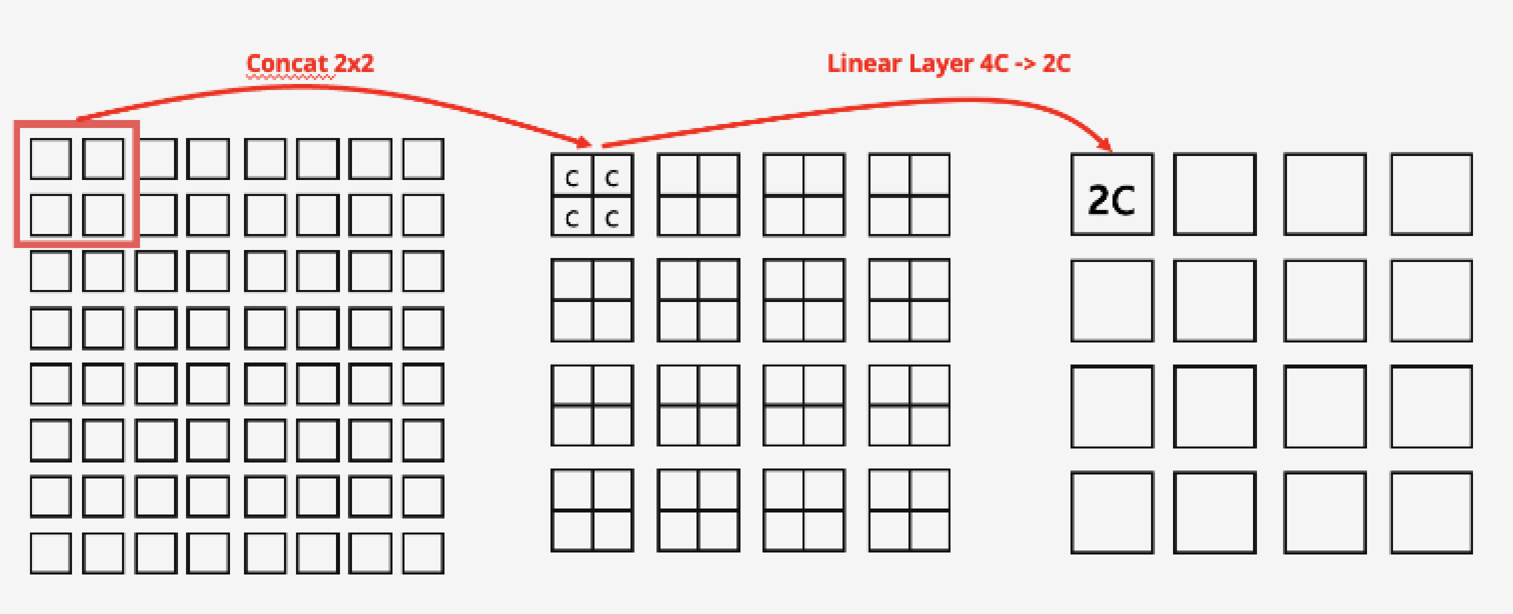
- Patch Merging

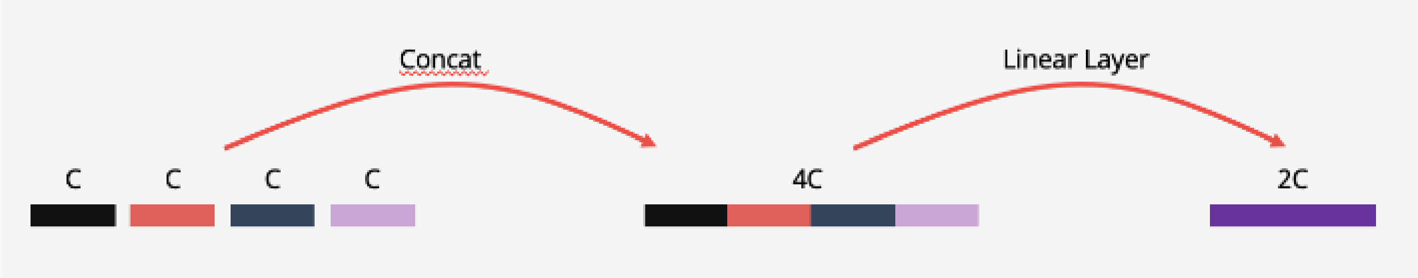


(a) Swin Transformer (ours) (b) ViT

Model Architecture

- Patch Merging

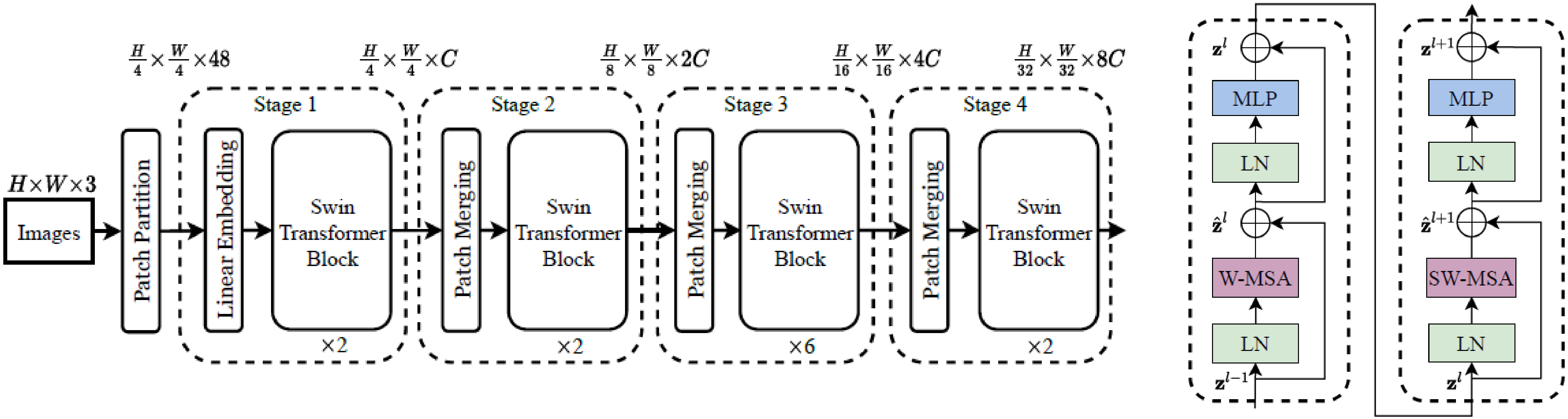




https://velog.io/@rucola-pizza/%EB%85%BC%EB%AC%B8-%EB%A6%AC%EB%B7%B0Swin-Transformer-Hierarchical-Vision-Transformer-using-Shifted-Windows

Method

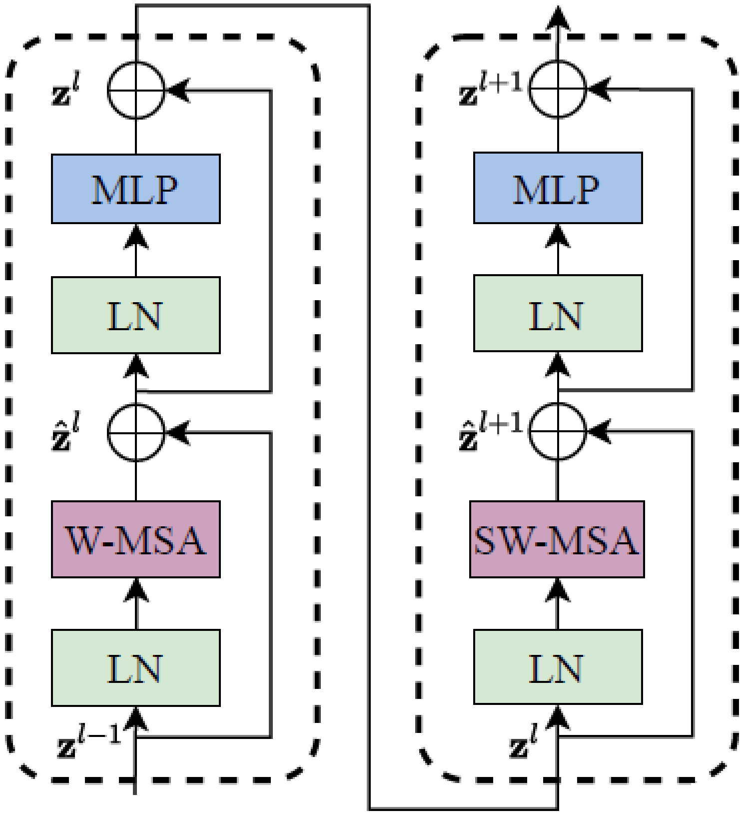
- Shifted Window based Self-Attention



Method

- Shifted Window based Self-Attention
  - W-MSA : Window-based Multihead Self-attention
  - SW-MSA : Shifted Window-based Multihead Self-attention

  - MLP : (Linear + GeLU) * 2
  - Residual Connection



Method

- Shifted Window based Self-Attention

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

- 기존의 MSA의 연산량
  - ➢ 이미지의 해상도(h*w)의 제곱의 비율로 증가

$$\Omega(\text{W-MSA}) = 4hwC^2 + 2M^2hwC, \qquad (2)$$

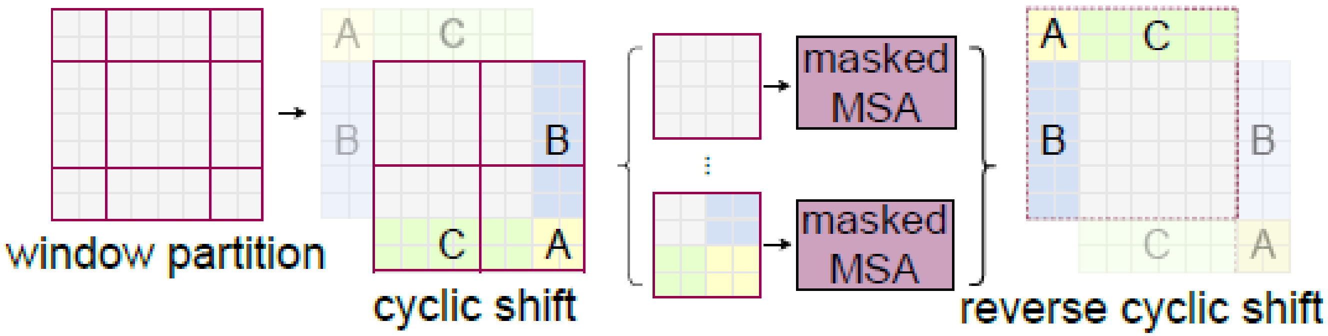
- W-MSA의 연산량
  - ➢ 정해진 상수인 Patch 개수의 제곱에 따라 증가
  - ➢ 이미지의 해상도(h*w)에 대해서는 선형적인 관계를 보임

➔ 연산량에 대한 이미지 해상도의 영향이 크지 않음

Method

- Shifted Window based Self-Attention

  = Cyclic-shifting



- 이미지의 Window를 이동시켜 패치 간의 연결성 확보
  - 논문에서는 (2, 2)만큼 이동. 최소 패치의 크기가 4*4인 것이 원인인 듯
  - 이동하고 남은 부분(그림에서 A, B, C)부분을 패딩으로 채우게 되면 Window의 개수가 증가
  - 이동시키고 Window 밖으로 나간 부분을 반대쪽에 연결

➔ 패치 간의 연결성 확보와 동시에 연산량 보존 가능

Method

- Shifted Window based Self-Attention
  - Relative position bias

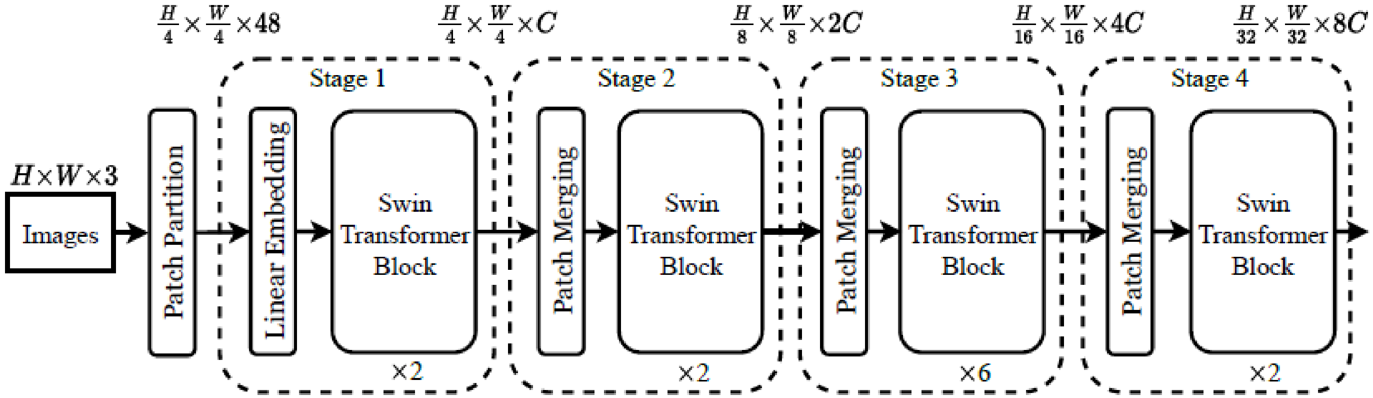
$$\text{Attention}(Q,K,V) = \text{SoftMax}(QK^T/\sqrt{d}\boxed{+B})V, \quad (4)$$

- Swin Transformer에는 Positional Encoding이 없음. ViT와 큰 차이점 중 하나
  - ➢ 대신 Self-attention을 계산할 때 Relative position bias를 추가함

➔ Absolute position encoding보다 성능 향상

Method

- Architecture Variants



Model Name	C (Embedding Dimension)	Layer Numbers
Swin-T	96	2, 2, 6, 2
Swin-S	96	2, 2, 18, 2
Swin-B	128	2, 2, 18, 2
Swin-L	192	2, 2, 18, 2

Experiments

- ImageNet-1K, 22K : Image Classification

(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

(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].

Experiments

- COCO : Object Detection

Method	Backbone	AP^{box}	AP^{box}_{50}	AP^{box}_{75}	#param.	FLOPs	FPS
(a) Various frameworks							
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	mini-val		test-dev			
Method	AP^{box}	AP^{mask}	AP^{box}	AP^{mask}	#param.	FLOPs
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.

Experiments

- ADE20K : Semantic Segmentation

ADE20K Method	Backbone	val mIoU	test score	#param.	FLOPs	FPS
DANet [23]	ResNet-101	45.2	-	69M	1119G	15.2
DLab.v3+ [11]	ResNet-101	44.1	-	63M	1021G	16.0
ACNet [24]	ResNet-101	45.9	38.5	-		
DNL [71]	ResNet-101	46.0	56.2	69M	1249G	14.8
OCRNet [73]	ResNet-101	45.3	56.0	56M	923G	19.3
UperNet [69]	ResNet-101	44.9	-	86M	1029G	20.1
OCRNet [73]	HRNet-w48	45.7	-	71M	664G	12.5
DLab.v3+ [11]	ResNeSt-101	46.9	55.1	66M	1051G	11.9
DLab.v3+ [11]	ResNeSt-200	48.4	-	88M	1381G	8.1
SETR [81]	T-Large‡	50.3	61.7	308M	-	-
UperNet	DeiT-S†	44.0	-	52M	1099G	16.2
UperNet	Swin-T	46.1	-	60M	945G	18.5
UperNet	Swin-S	49.3	-	81M	1038G	15.2
UperNet	Swin-B‡	51.6	-	121M	1841G	8.7
UperNet	Swin-L‡	**53.5**	**62.8**	234M	3230G	6.2

Table 3. Results of semantic segmentation on the ADE20K val and test set. † indicates additional deconvolution layers are used to produce hierarchical feature maps. ‡ indicates that the model is pre-trained on ImageNet-22K.

Experiments

- Ablation Study

	ImageNet		COCO		ADE20k
	top-1	top-5	AP^{box}	AP^{mask}	mIoU
w/o shifting	80.2	95.1	47.7	41.5	43.3
shifted windows	**81.3**	**95.6**	**50.5**	**43.7**	**46.1**
no pos.	80.1	94.9	49.2	42.6	43.8
abs. pos.	80.5	95.2	49.0	42.4	43.2
abs.+rel. pos.	81.3	95.6	50.2	43.4	44.0
rel. pos. w/o app.	79.3	94.7	48.2	41.9	44.1
rel. pos.	**81.3**	**95.6**	**50.5**	**43.7**	**46.1**

Table 4. Ablation study on the *shifted windows* approach and different position embedding methods on three benchmarks, using the Swin-T architecture. w/o shifting: all self-attention modules adopt regular window partitioning, without *shifting*; abs. pos.: absolute position embedding term of ViT; rel. pos.: the default settings with an additional relative position bias term (see Eq. (4)); app.: the first scaled dot-product term in Eq. (4).

Experiments

- Ablation Study

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.

Experiments

- Ablation Study

	Backbone	ImageNet top-1	ImageNet top-5	COCO AP^{box}	COCO AP^{mask}	ADE20k mIoU
sliding window	Swin-T	81.4	95.6	50.2	43.5	45.8
Performer [14]	Swin-T	79.0	94.2	-	-	-
shifted window	Swin-T	81.3	95.6	50.5	43.7	46.1

Table 6. Accuracy of Swin Transformer using different methods for self-attention computation on three benchmarks.

Q&A

[Github]
https://github.com/microsoft/Swin-Transformer