Swin Transformer: Hierarchical Vision Transformer using Shifted Windows

School of Industrial and Management Engineering, Korea University

Young Jae Lee

KOREA UNIVERSITY

DMQA

Contents

Research Purpose

❖ Swin Transformer: Hierarchical Vision Transformer using Shifted Windows (arXiv, 2021)

- Microsoft Research에서 연구하였고 2021년 08월 06일 기준으로 약 127회 인용

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
{v-zeliu1,v-yutlin,yuecao,hanhu,v-yixwe,zhez,stevelin,bainguo}@microsoft.com

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 (86.4 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 code and models will be made publicly available at https://github.com/microsoft/Swin-Transformer.

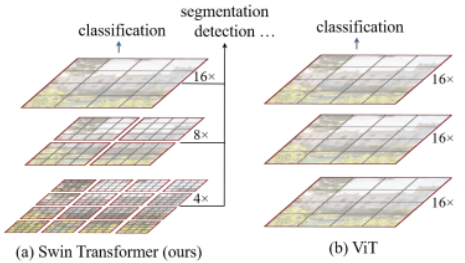


Figure 1. (a) The proposed Swin Transformer builds hierarchical feature maps by merging image patches (shown in gray) in deeper layers and has linear computation complexity to input image size due to computation of self-attention only within each local window (shown in red). It can thus serve as a general-purpose backbone for both image classification and dense recognition tasks. (b) In contrast, previous vision Transformers [19] produce feature maps of a single low resolution and have quadratic computation complexity to input image size due to computation of self-attention globally.

CNNs serving as backbone networks for a variety of vision tasks, these architectural advances have led to performance improvements that have broadly lifted the entire field.

On the other hand, the evolution of network architectures in natural language processing (NLP) has taken a different path, where the prevalent architecture today is instead the

DMQA

Research Purpose

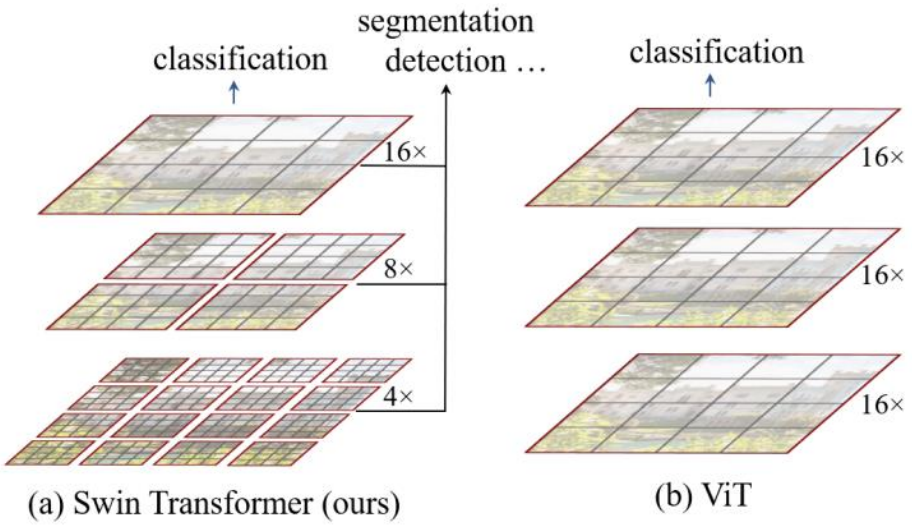
❖ Swin Transformer: Hierarchical Vision Transformer using Shifted Windows (arXiv, 2021)

- Shifted Window 기반 Self-Attention (Local) 연산을 도입한 Hierarchical Architecture 제안
  - ✓ Transformer를 Language에서 Vision 영역까지 Adaptation하는 어려움을 완화
  - ✓ Vision Transformer (ViT)에서 모든 Patch가 Self-Attention (Global) 연산을 수행할 때 발생하는 Computational Cost 감소
- Hierarchical Architecture는 다양한 Scale의 모델에 **유연성**을 갖게 되고 Image 사이즈에 대해 **Linear** Computational Cost를 갖게 됨
- Computer Vision의 General-Purpose Backbone 강조

Swin Transformer

❖ Hierarchical Feature Representation

- 작은 크기의 Patch (회색 윤곽선)로부터 시작함으로써 더 깊은 Transformer 레이어의 인접 Patch를 점진적으로 병합하여 계층적 표현을 구성
- 일반적인 Transformer와 달리 계층적 Feature Map을 통하여 Feature Pyramid Networks (FPN) 또는 U-Net과 같이 Object Detection, Segmentation에 활용 가능
- **Linear** Computational Cost는 겹치지 않은 이미지 (빨간색 윤곽선) 내에서 지역적으로 Self-Attention 연산을 수행함으로써 달성

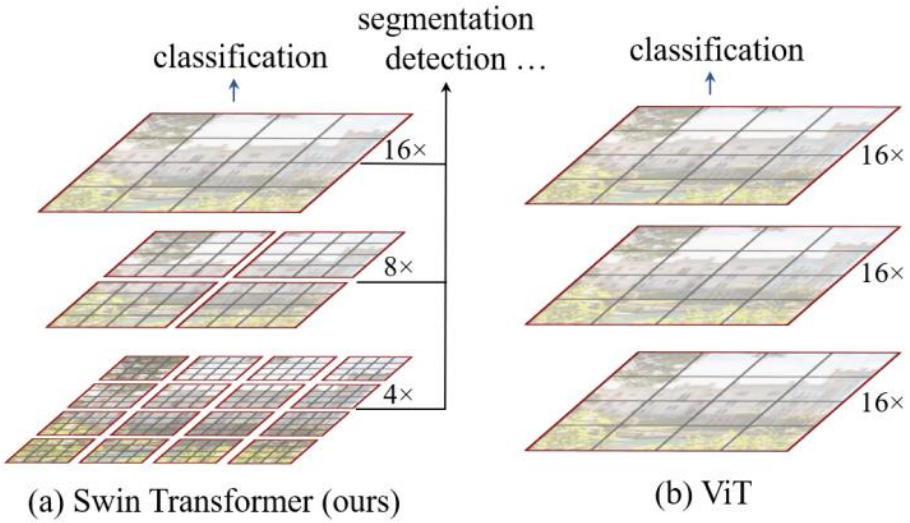


DMQA

Swin Transformer

❖ Hierarchical Feature Representation (Example)

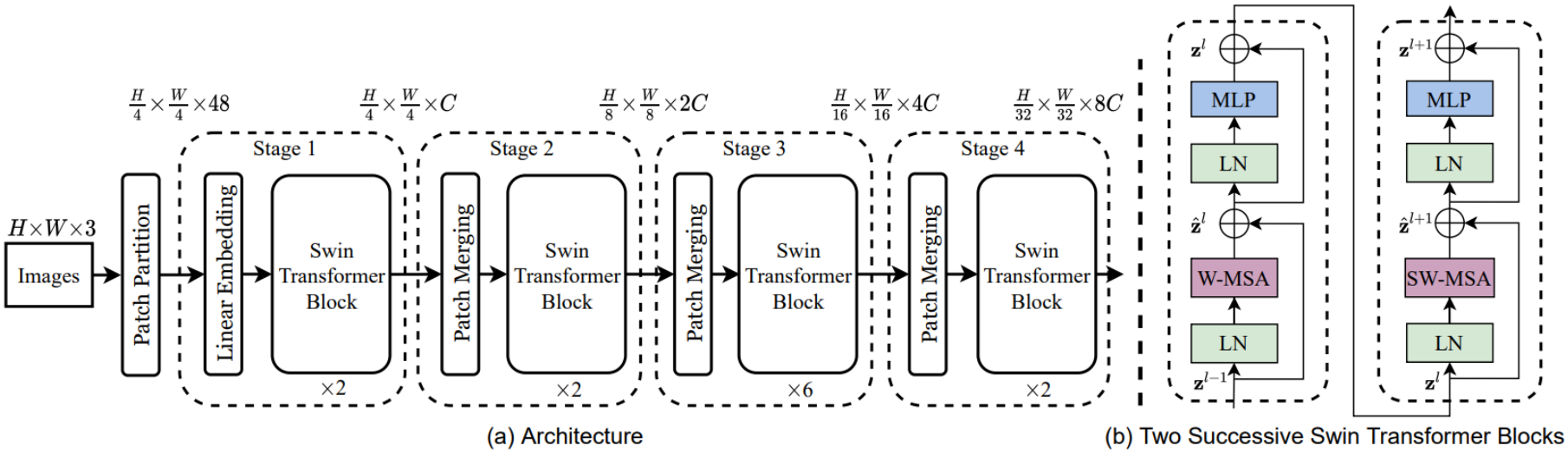
- 입력 이미지 사이즈: 224 X 224
- Window 사이즈(M): 7 X 7
- 첫번째 레이어에서 4 X 4 사이즈의 각 Patch가 56 X 56개가 존재할 때 Window 사이즈로 나누어 8 X 8개의 Window를 생성
- 첫번째 단계에서 각 Patch는 16개의 Pixel이 존재하며 각 Window에는 49개의 Patch가 존재



Swin Transformer

❖ Overall Architecture

- 크게 Patch Partition, Linear Embedding, Swin Transformer Block, Patch Merging으로 구분
- 4개의 단계로 이루어져 있음(Stage 1 ~ 4)
- 제안 방법론의 핵심인 그림 (b)는 두개의 Encoder로 이루어져 있으며 일반적인 Multi-Head Self-Attention (MSA)가 아닌 W-MSA, SW-MSA로 이루어져 있음
- 그림 (b)에서 2개의 Encoder를 하나로 보았을 때, 각 단계에서 Block 적용 횟수는 실제로 1, 1, 3, 1



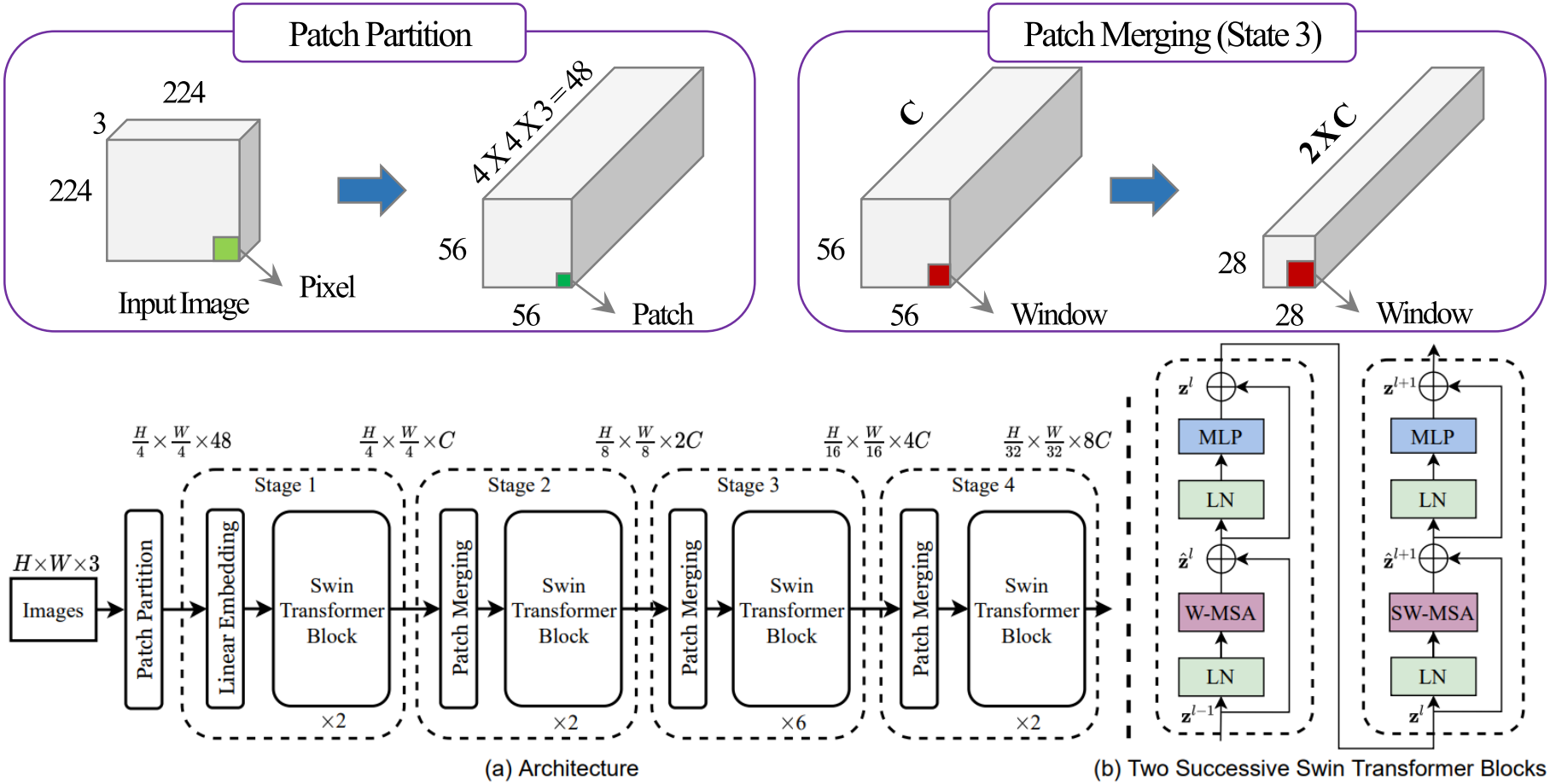
(a) Architecture (b) Two Successive Swin Transformer Blocks

DMQA

Swin Transformer

❖ Overall Architecture – Patch Partition, Patch Merging

- 이미지에서 Patch로 Partition하는 것과 Merging하는 것은 같은 의미를 가짐



(a) Architecture

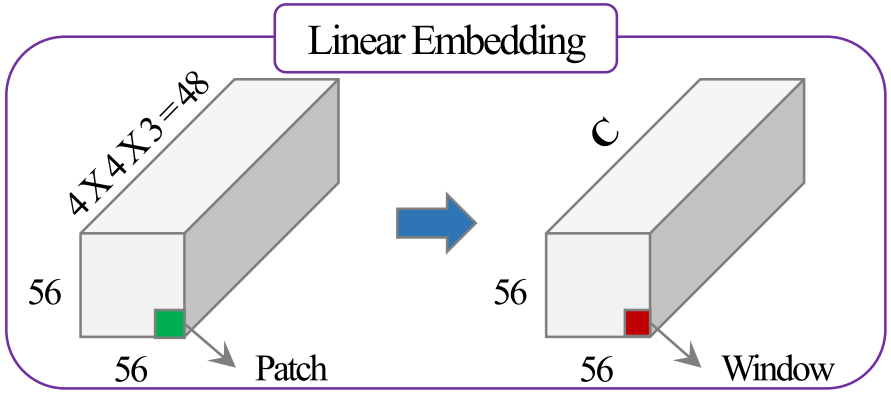
(b) Two Successive Swin Transformer Blocks

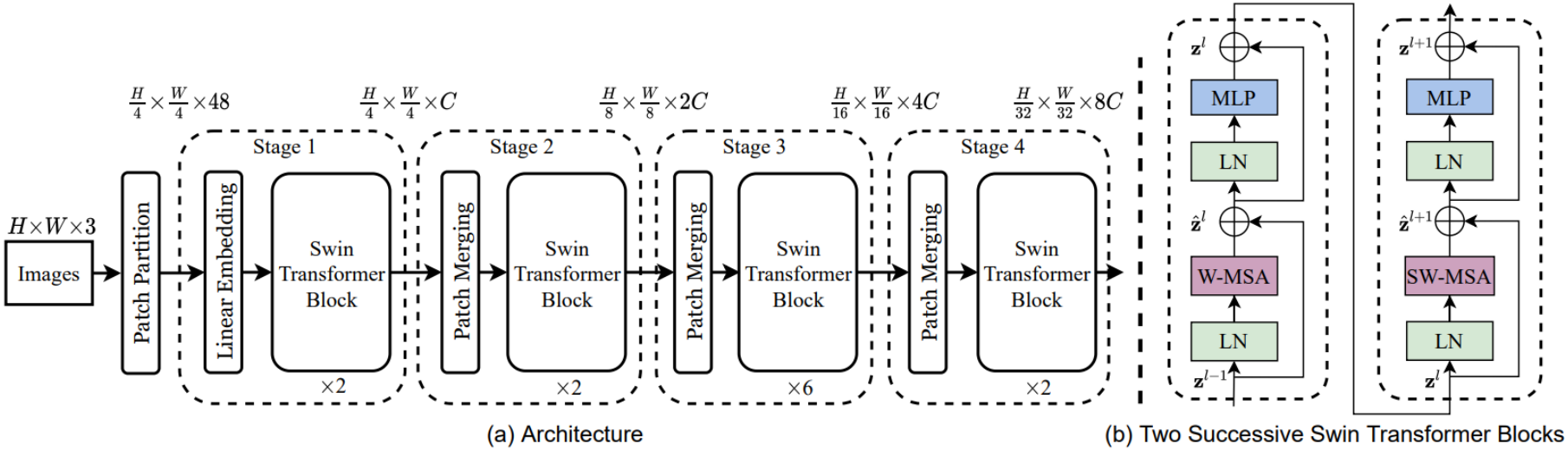
DMQA

Swin Transformer

❖ Overall Architecture – Linear Embedding

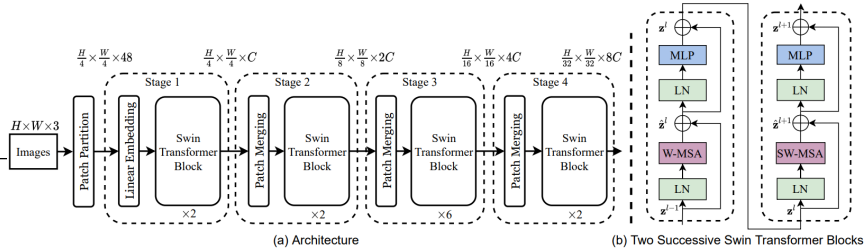
- Patch Partition 또는 Patch Merging 후에 Linear Layer로부터 Dimension C로 출력





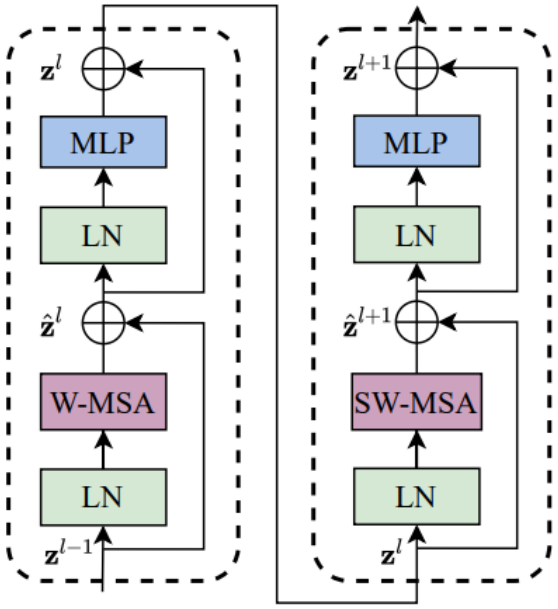
DMQA

Swin Transformer



❖ Overall Architecture – Swin Transformer Block

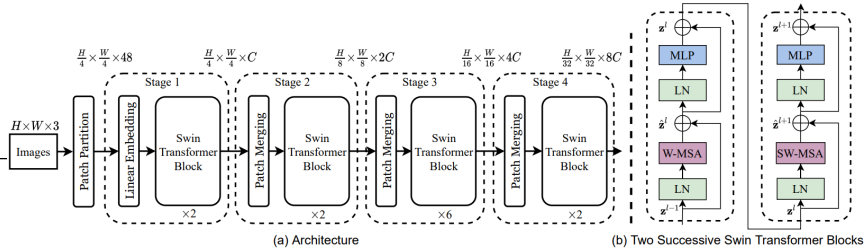
- 두개의 Encoder로 이루어져 있으며 일반적인 Multi-Head Self-Attention (MSA)가 아닌 W-MSA, SW-MSA로 이루어져 있음
- W-MSA는 현재 Window에 있는 Patch들로만 Self-Attention 연산함으로써 계산 비용 감소
- MSA는 Quadratic, W-MSA는 **Linear** Computation (논문에서는 M을 7로 고정)
  - ✓ W-MSA은 hw보다 M이 훨씬 작기 때문에 이미지 사이즈가 커져도 ViT보다 계산 비용을 줄일 수 있음



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

DMQA

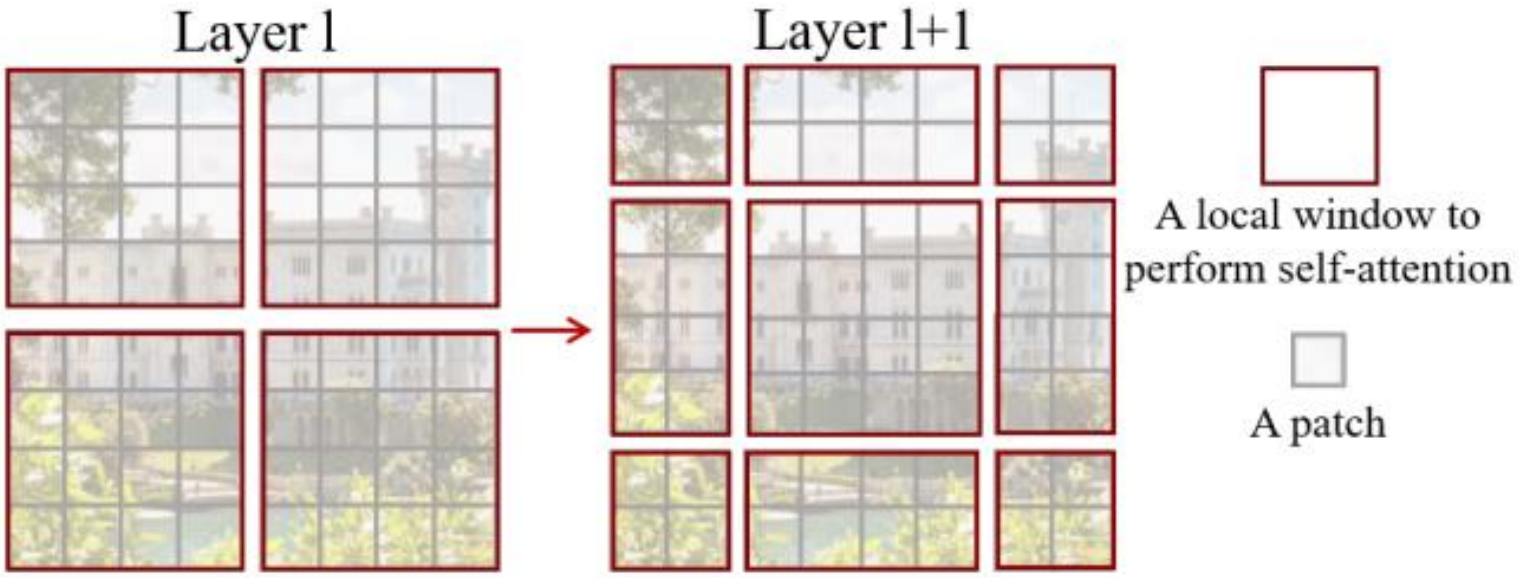
Swin Transformer



(a) Architecture (b) Two Successive Swin Transformer Blocks

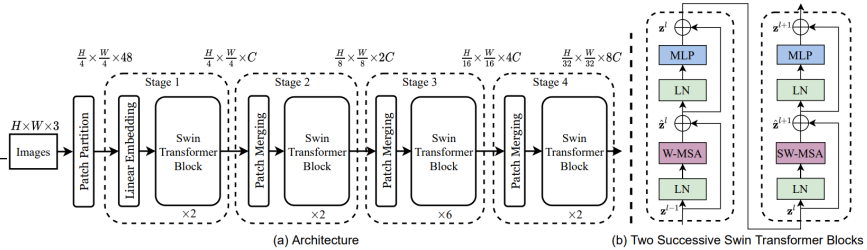
❖ Overall Architecture – Swin Transformer Block

- W-MSA는 Window가 고정되어 있어 고정된 부분에서만 Self-Attention을 연산하는 단점 존재
- Shifted Window 방법을 도입하여 위의 문제를 해결(SW-MSA)
  - ✓ Layer *l*은 Regular Window Partitioning Scheme
  - ✓ Layer *l+1*은 Window Partitioning이 이동되어 새로운 Window를 생성(Shifted Window 결과)



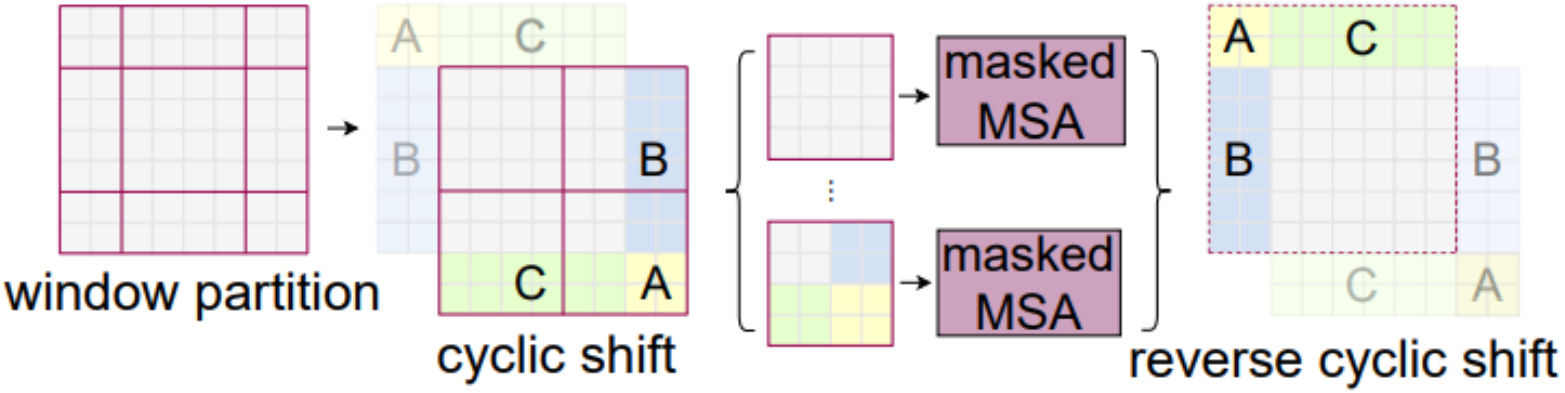
DMQA

Swin Transformer



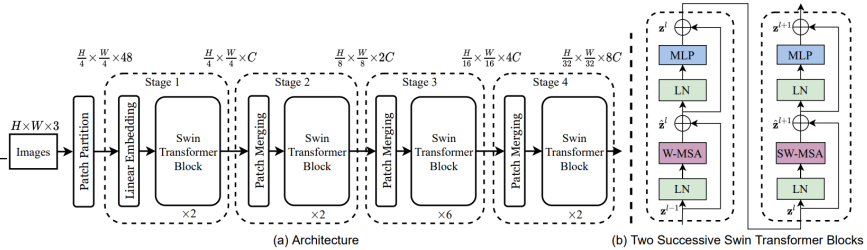
❖ Overall Architecture – Swin Transformer Block

- SW-MSA 연산은 **1) Cyclic Shift**로 Window를 Shift 시킴
- Window Size / 2 만큼 우측 하단으로 Shift하고 Shift된 A, B, C 구역에 **2) Mask**를 씌워 Self-Attention을 하지 못하도록 함
  - ✓ 원래 좌측 상단에 있던 정보이기 때문에 Self-Attention 연산의 의미가 없음
- **3) Reverse Cyclic Shift**: Mask 연산을 한 후 다시 원래 값으로 되돌림



DMQA

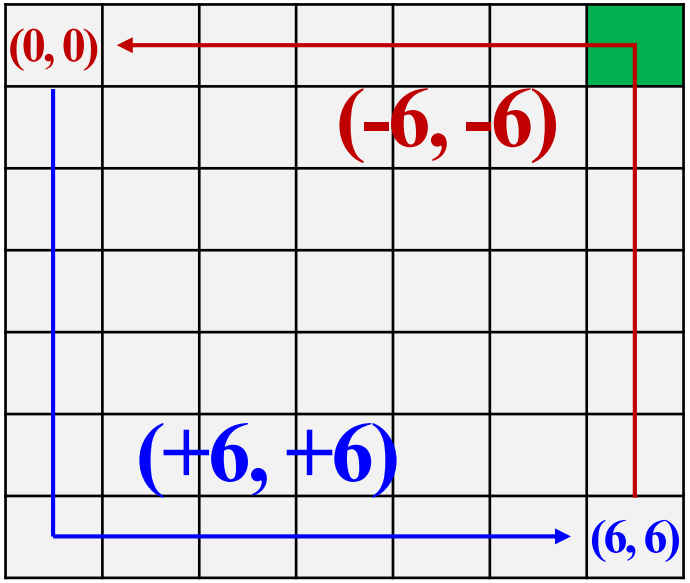
Swin Transformer



❖ Overall Architecture – Relative Position Bias

- Swin Transformer는 ViT와 다르게 Positional Encoding 정보를 더하지 않음
- Self-Attention 연산 과정에서 Relative Position Bias를 추가함(아래 수식에서 **B**를 의미)
  - ✓ 논문에서는 실험을 통해서 상대 좌표를 더해주는 것이 좋은 방법임을 증명

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



Patch

- Window Size: 7 X 7 (Patch 구성, 8 page 참고)
- (0, 0) Patch에서 (6, 6) Patch로 이동 시 (+6, +6)만큼 이동
- (6, 6) Patch에서 (0, 0) Patch로 이동 시 (-6, -6)만큼 이동
- Patch 중심 기준에 따라 이동해야 하는 값이 달라짐
- 따라서, 각 축 범위 [-6, 6] 기준으로 상대적 좌표를 Embedding하는 것이 효과적

DMQA

Experiments

❖ Image Classification Metric

- Top-1 Accuracy: Softmax의 Output에서 제일 높은 수치를 가지는 값이 정답일 경우에 대한 지표
- Float Point Operations Per Second (FLOPs): 컴퓨터의 성능을 표현하는 지표
- Parameters: Model의 Weight 또는 Parameter 수

❖ Object Detection Metric

- Average Precision (AP): IoU 계산 결과 값이 0.5 이상이면 True Positive (TP), 0.5 미만이면 False Positive (FP)로 판단하고 검출 결과들 중 옳게 검출한 비율을 의미(정확도)

❖ Semantic Segmentation Metric

- Mean Intersection over Union (mIoU): 예측 및 실제 픽셀 간 교집합에 포함되는 정도에 대한 지표
- Float Point Operations Per Second (FLOPs): 컴퓨터의 성능을 표현하는 지표

Experiments

Image Classification Results

❖ Results with Regular ImageNet-1K Training / with ImageNet-22K Pre-Training

- EfficientNet-B7과 대등한 결과

(a) Regular ImageNet-1K trained models

method	image size	#param.	FLOPs	throughput (image / s)	ImageNet top-1 acc.
RegNetY-4G [47]	224^2	21M	4.0G	1156.7	80.0
RegNetY-8G [47]	224^2	39M	8.0G	591.6	81.7
RegNetY-16G [47]	224^2	84M	16.0G	334.7	82.9
EffNet-B3 [57]	300^2	12M	1.8G	732.1	81.6
EffNet-B4 [57]	380^2	19M	4.2G	349.4	82.9
EffNet-B5 [57]	456^2	30M	9.9G	169.1	83.6
EffNet-B6 [57]	528^2	43M	19.0G	96.9	84.0
EffNet-B7 [57]	600^2	66M	37.0G	55.1	84.3
ViT-B/16 [19]	384^2	86M	55.4G	85.9	77.9
ViT-L/16 [19]	384^2	307M	190.7G	27.3	76.5
DeiT-S [60]	224^2	22M	4.6G	940.4	79.8
DeiT-B [60]	224^2	86M	17.5G	292.3	81.8
DeiT-B [60]	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	83.3
Swin-B	384^2	88M	47.0G	84.7	84.2

(b) ImageNet-22K pre-trained models

method	image size	#param.	FLOPs	throughput (image / s)	ImageNet top-1 acc.
R-101x3 [37]	384^2	388M	204.6G	-	84.4
R-152x4 [37]	480^2	937M	840.5G	-	85.4
ViT-B/16 [19]	384^2	86M	55.4G	85.9	84.0
ViT-L/16 [19]	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	86.0
Swin-L	384^2	197M	103.9G	42.1	86.4

Experiments

Object Detection Results

❖ Object Detection on COCO

- Backbone으로 사용했을 때 Detection에서 최고의 성능을 보임

(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 AP^{box}	mini-val AP^{mask}	test-dev AP^{box}	test-dev AP^{mask}	#param.	FLOPs
RepPointsV2* [11]	-	-	52.1	-	-	-
GCNet* [6]	51.8	44.7	52.3	45.4	-	1041G
RelationNet++* [12]	-	-	52.7	-	-	-
SpineNet-190 [20]	52.6	-	52.8	-	164M	1885G
ResNeSt-200* [75]	52.5	-	53.3	47.1	-	-
EfficientDet-D7 [58]	54.4	-	55.1	-	77M	410G
DetectoRS* [45]	-	-	55.7	48.5	-	-
YOLOv4 P7* [3]	-	-	55.8	-	-	-
Copy-paste [25]	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	-

Experiments

Semantic Segmentation Results

❖ Semantic Segmentation on ADE20K

- Backbone으로 사용했을 때 Segmentation에서 최고의 성능을 보임

ADE20K Method	Backbone	val mIoU	test score	#param.	FLOPs	FPS
DANet [22]	ResNet-101	45.2	-	69M	1119G	15.2
DLab.v3+ [10]	ResNet-101	44.1	-	63M	1021G	16.0
ACNet [23]	ResNet-101	45.9	38.5	-		
DNL [68]	ResNet-101	46.0	56.2	69M	1249G	14.8
OCRNet [70]	ResNet-101	45.3	56.0	56M	923G	19.3
UperNet [66]	ResNet-101	44.9	-	86M	1029G	20.1
OCRNet [70]	HRNet-w48	45.7	-	71M	664G	12.5
DLab.v3+ [10]	ResNeSt-101	46.9	55.1	66M	1051G	11.9
DLab.v3+ [10]	ResNeSt-200	48.4	-	88M	1381G	8.1
SETR [78]	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

Experiments

Ablation Study

❖ Shifted Window / Relative Position Bias

- SW-MSA 사용했을 때 / Relative Position Bias 사용했을 때 좋은 성능을 보임

	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**

Conclusion

- ❖ Shifted Window 기반 Self-Attention 연산을 도입한 Hierarchical Architecture 제안
- ❖ Hierarchical Architecture를 제안함으로써 다양한 Scale의 모델에 Flexibility를 갖게 되고 Image 사이즈에 대해 Linear Computational Cost를 갖게 됨
- ❖ 특히, 제안 방법론은 Image Classification 뿐만 아니라 Object Detection, Semantic Segmentation 등 다양한 방법론의 Backbone으로 사용할 수 있는 점을 강조(General-Purpose Backbone)
- ❖ Object Detection, Semantic Segmentation에 Backbone으로 사용함으로써 최고의 성능을 보임
  - 후기: 무거운 Transformer의 계산 비용을 줄이는 Window 단위의 Local 연산과 더불어 Hierarchical 구조를 제안함으로써 다양한 분야에 적용할 수 있다는 점이 Good!
    다만, 다양한 데이터로 실험 결과를 보여주었으면 더욱 Best!
    강화학습에 Tiny 버전으로 활용하여 State Foreground Extraction에 녹여볼 법함

- ❖ Reference

  - Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., ... & Guo, B. (2021). Swin transformer: Hierarchical vision transformer using shifted windows. arXiv preprint arXiv:2103.14030.


Thank You