DMQA Open Seminar

Backbone Network in Deep learning

2022-11-25

Korea University

Data Mining & Quality Analytics Lab.

심 세 진

발표자 소개

❖ 심세진 (Sejin Sim)

- 고려대학교 산업경영공학과 대학원 재학
- Data Mining & Quality Analytics Lab. (김성범 교수님)
- 석사 과정 (2022. 03 ~ Present)

❖ 관심 연구 분야

- Deep Learning for Multichannel Signal Analysis
- Time Series Representation

❖ 연락처

- ssj259@korea.ac.kr

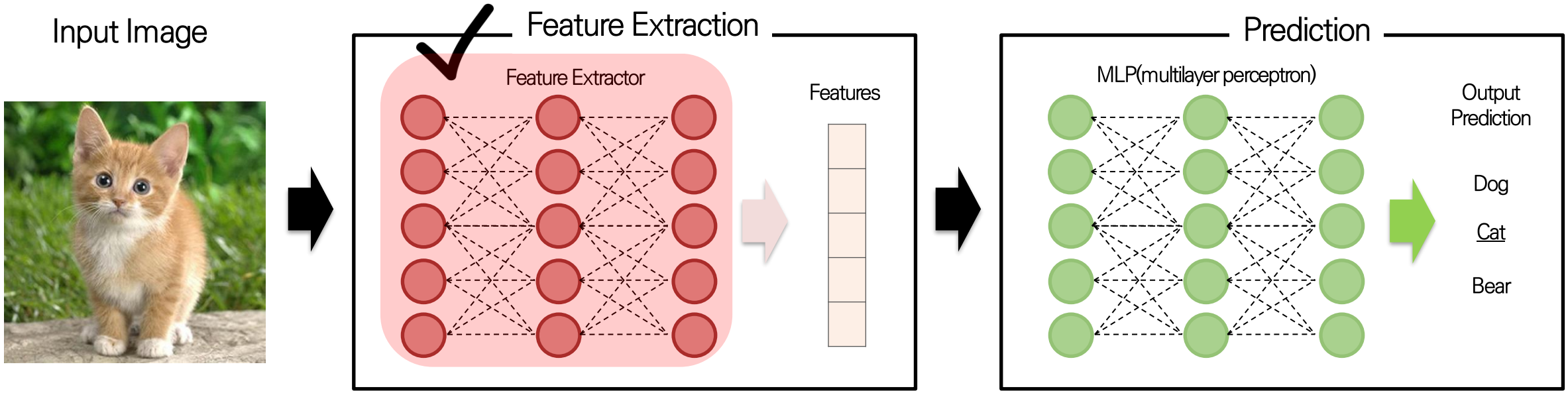
Contents

Introduction

Introduction

❖ What is Backbone Network?

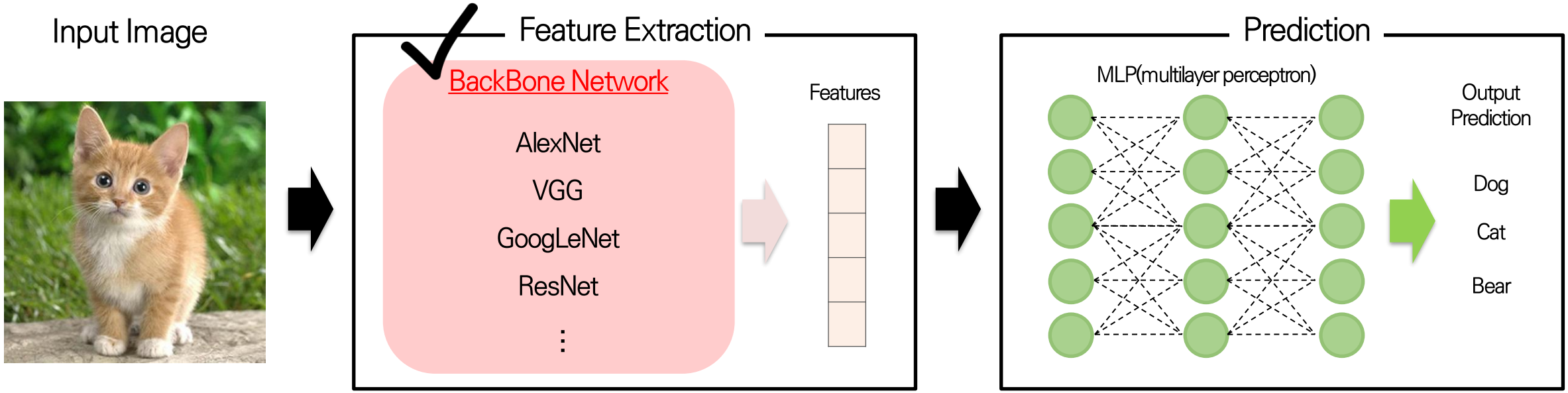
- 일반적인 이미지 분류를 위한 인공 신경망 구조



Introduction

❖ What is Backbone Network?

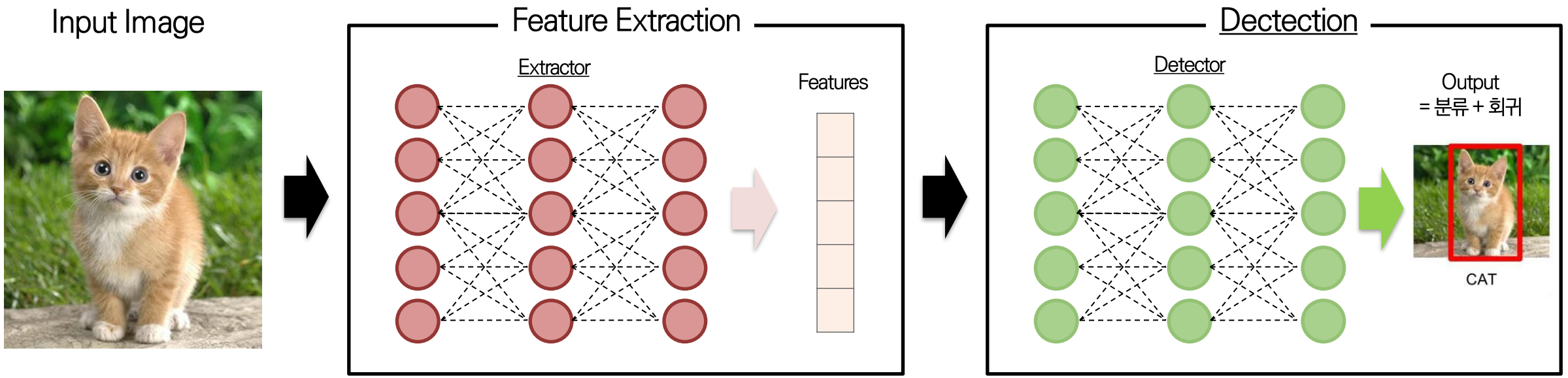
- 일반적인 이미지 분류를 위한 인공 신경망 구조



Introduction

❖ What is Backbone Network?

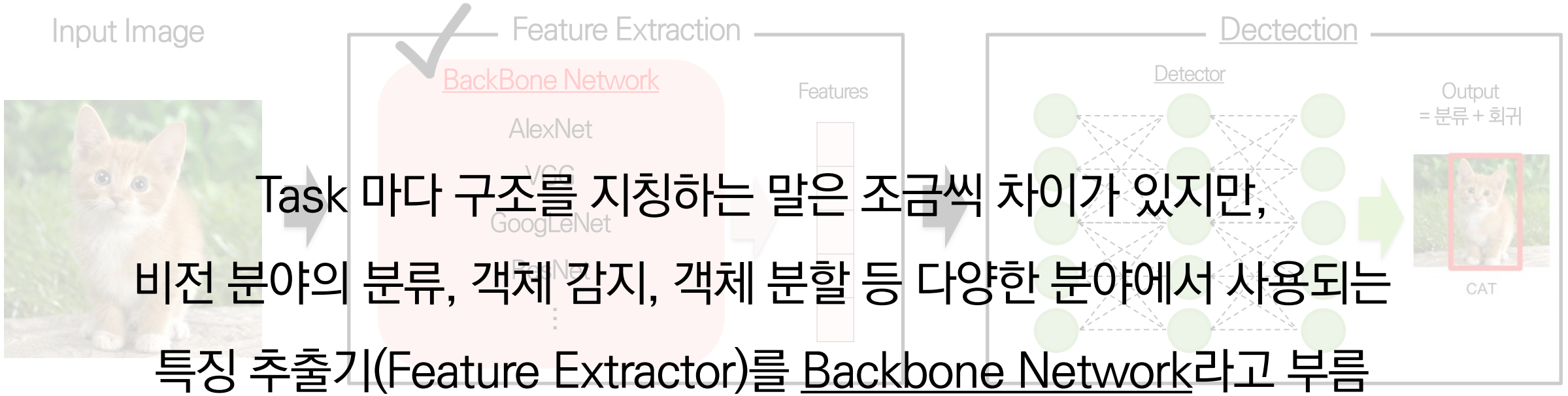
- Vision 분야 객체 검출(Object Detection) Task – 객체의 존재 유무 확인



Introduction

❖ What is Backbone Network?

- Vision 분야 객체 검출(Object Detection) Task – 객체의 존재 유무 확인



Task 마다 구조를 지칭하는 말은 조금씩 차이가 있지만,

비전 분야의 분류, 객체 감지, 객체 분할 등 다양한 분야에서 사용되는

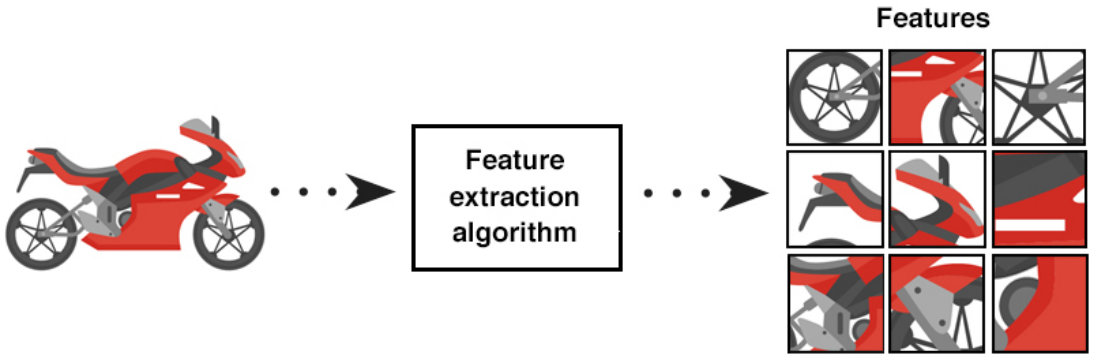
특징 추출기(Feature Extractor)를 Backbone Network라고 부름

Introduction

❖ What is Backbone Network?

- 중요한 이유
  - ✓ 입력 데이터의 패턴(=특징)을 잘 추출 해야함
  - ✓ 입력 데이터가 고차원, 대규모인 경우 계산 복잡도를 낮춰야 함

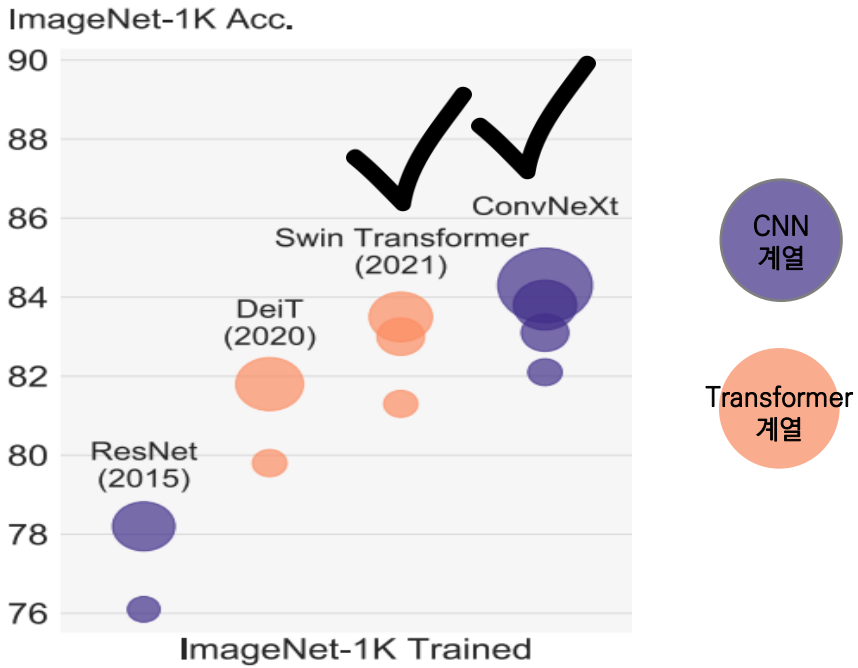
    → 궁극적으로 학습된 모델의 예측 정확도를 높이고 계산량을 낮추기 위해 중요



Introduction

❖ What is Backbone Network?

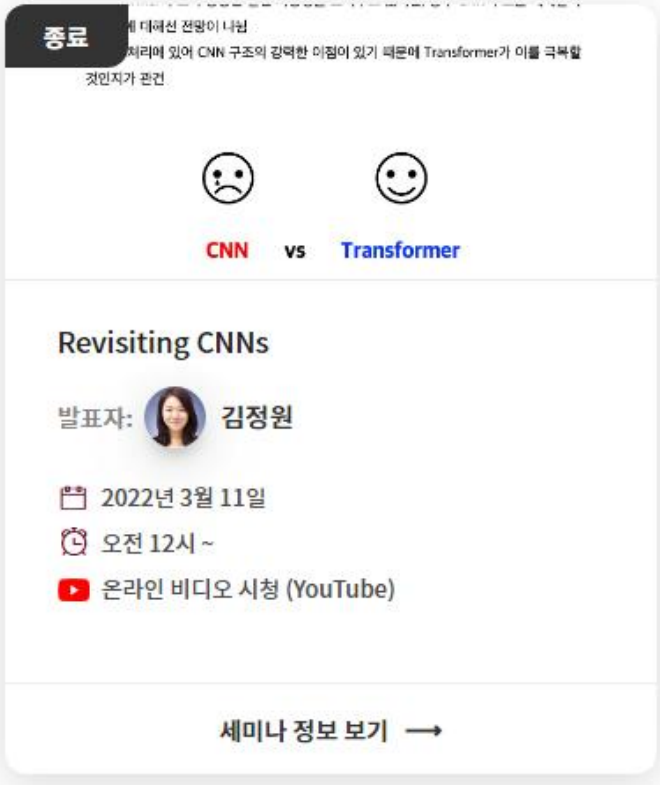
- 이미지 분류 분야(ImageNet)의 Backbone Network 간략한 발전 흐름
- CNN(합성곱 신경망) 계열 → ViT(Vision Transformer)의 등장
- 백본 네트워크의 발전은 효과적인 구조를 통해 비전 분야에서 일관된 성능 향상을 보임



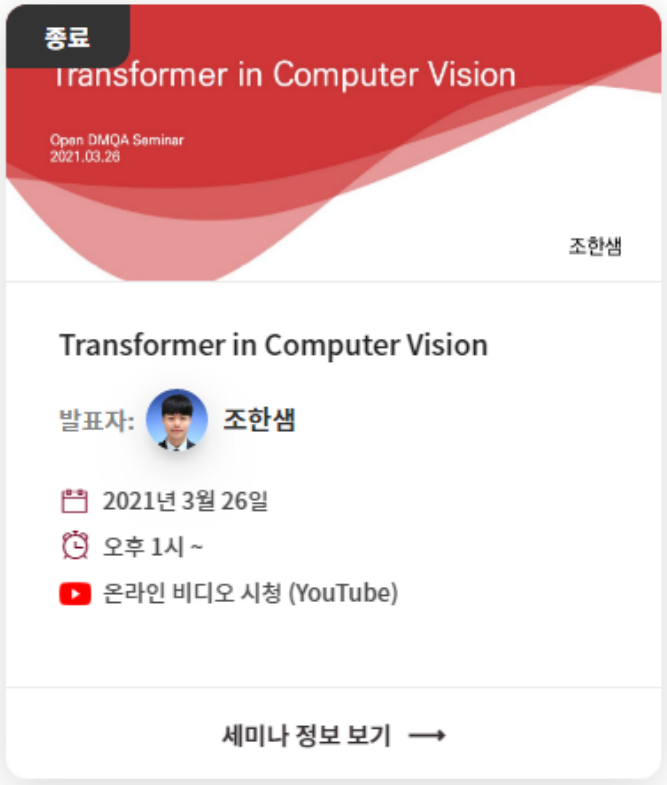
Introduction

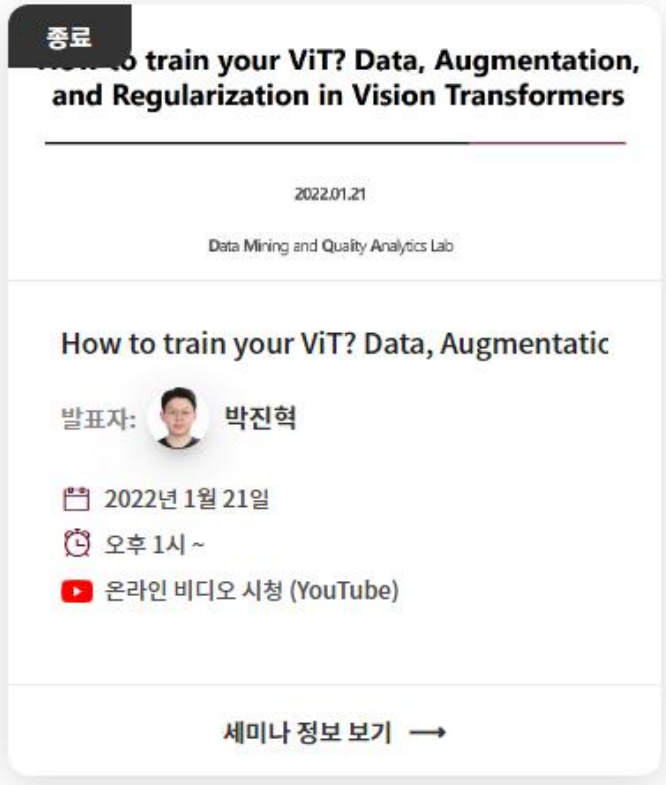
❖ 관련 세미나 소개

CNN 발전 관련



Vision Transformer(ViT) 관련





Backbone Network

Backbone Network

❖ Swin Transformer: Hierarchical Vision Transformer using Shifted Windows

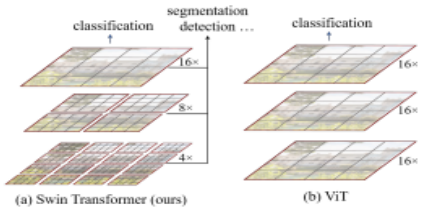
- ICCV (2021), 2022년 11월 18일 기준 3,712회 인용
- Transformer 계열 임에도 불구하고 CNN 기반 모델과 유사한 시간 소요량을 보이면서 높은 성능을 가짐

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-zeliul,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 (87.3 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 hierarchical design and the shifted window approach also prove beneficial for all-MLP architectures. The code and models are publicly available at https://github.com/microsoft/Swin-Transformer.



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

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 [20] produce feature maps of a single low resolution and have quadratic computation complexity to input image size due to computation of self-attention globally.

greater scale [30, 76], more extensive connections [34], and more sophisticated forms of convolution [70, 18, 84]. With 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

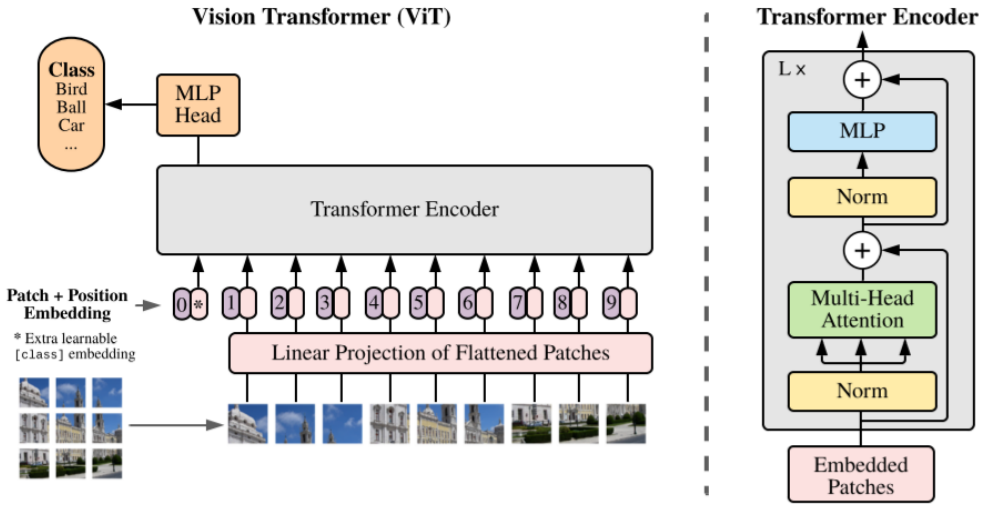
Backbone Network

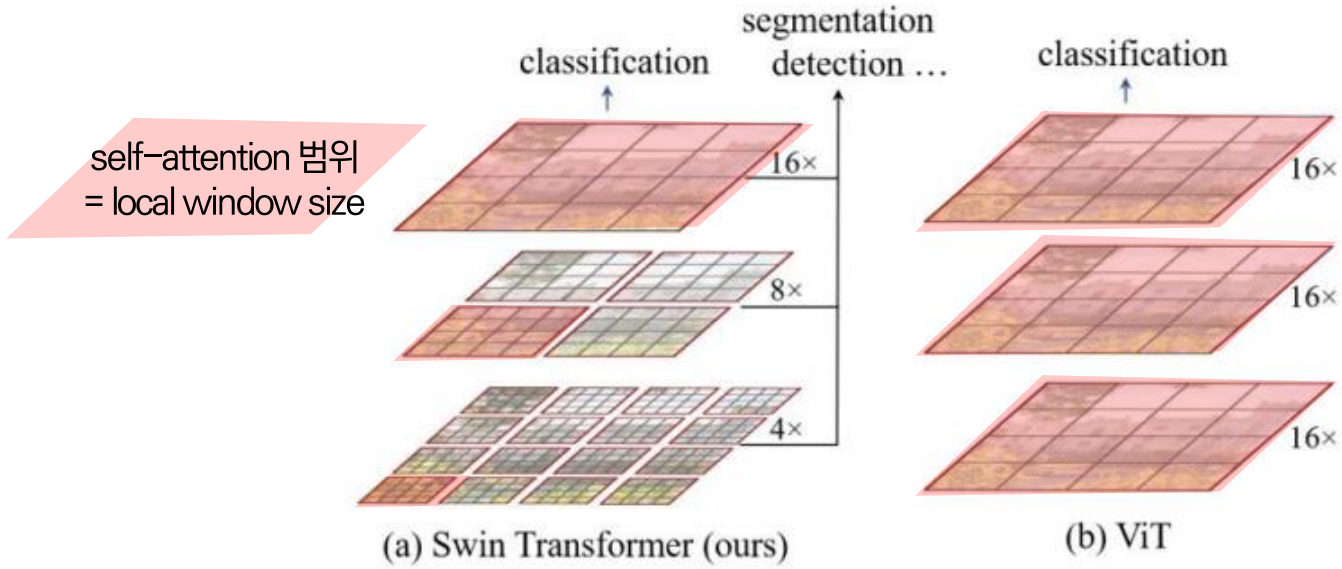
❖ Swin Transformer: Hierarchical Vision Transformer using Shifted Windows

- 등장 배경
  - ✓ 기존 Vision Transformer(ViT)에서 좋은 성능을 내기 어려운 이유

    → 이미지의 경우 텍스트(고정 된 크기의 token 사용)와 다르게 물체의 크기와 해상도가 매우 다양하기 때문

    → 전체 데이터를 통해 attention을 진행 하기 때문에 많은 양의 데이터가 필요



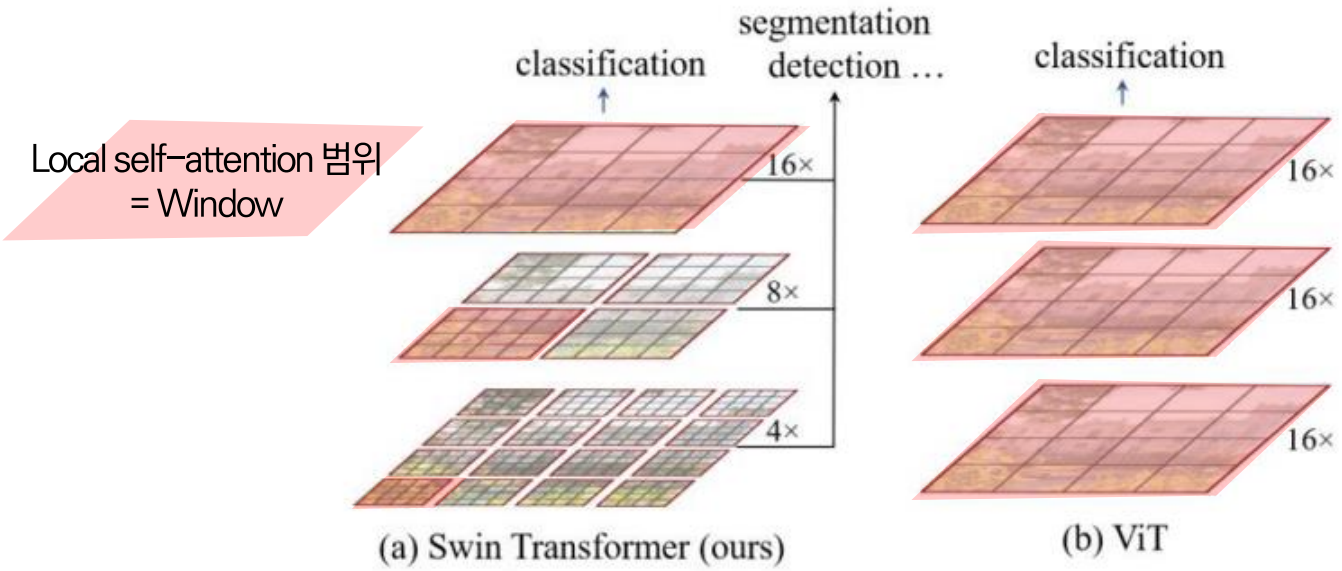


Backbone Network

❖ Swin Transformer: Hierarchical Vision Transformer using Shifted Windows

- 제안 방법론
  - ✓ 계층적인 Feature map을 구성하는 Hierarchical Shifted windows 방법론을 제안

    → ViT 대비 선형적인 계산 복잡도를 가지며 좋은 성능을 보임



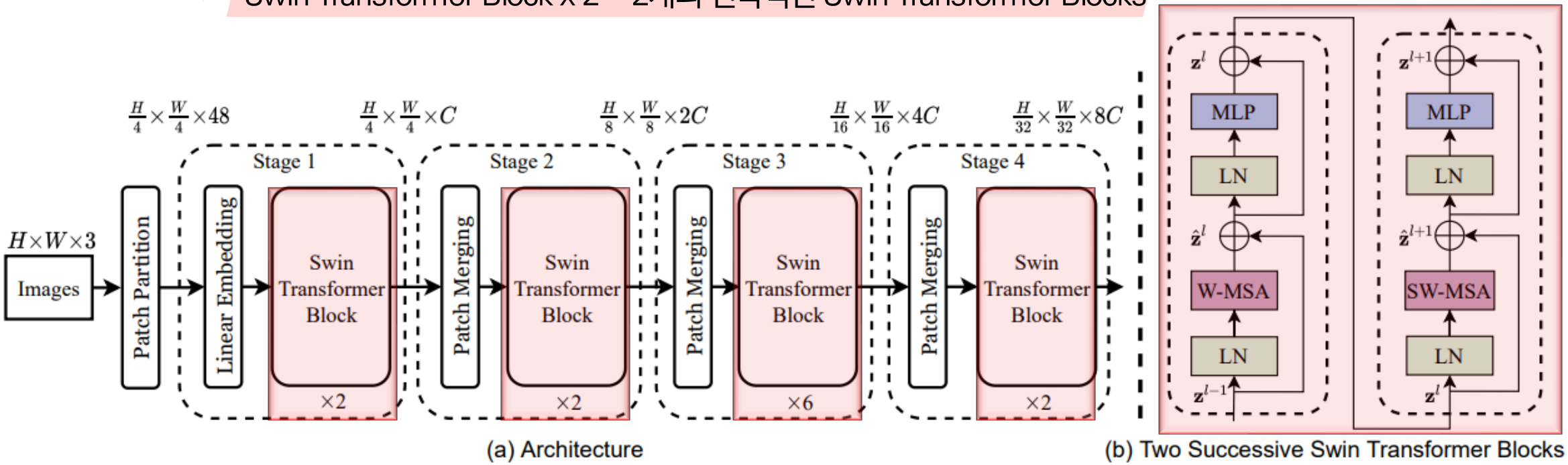
Window 크기 ↓

➡ Local self-attention 계산량 ↓

Backbone Network

❖ Swin Transformer: Hierarchical Vision Transformer using Shifted Windows

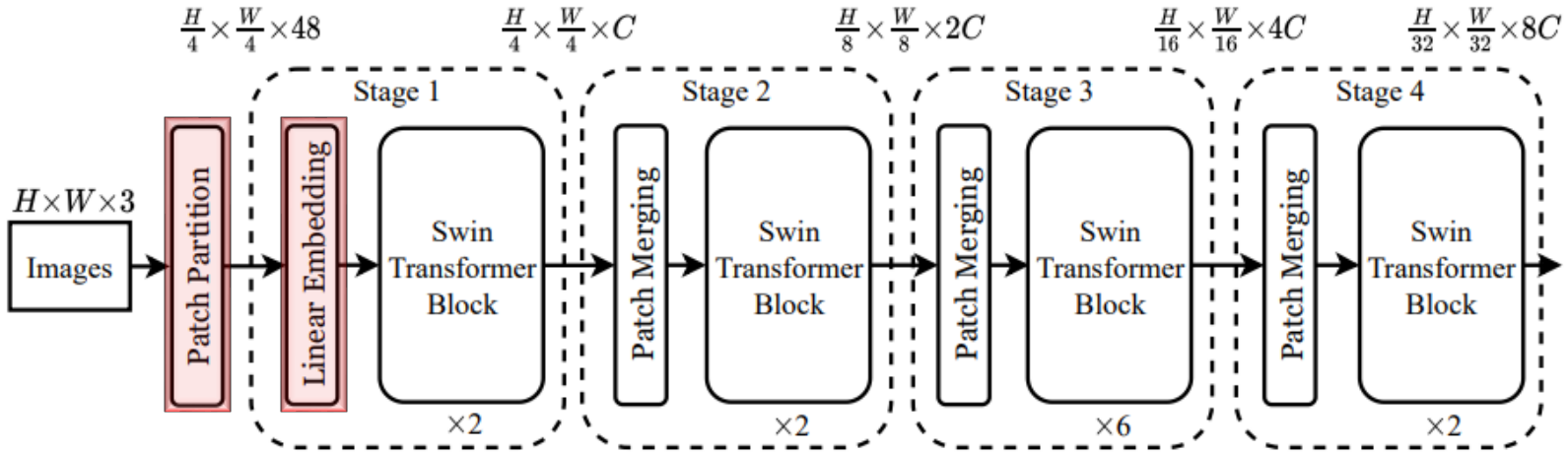
- Architecture (ex. Swin-tiny)
  - ✓ Stage 1~4
  - ✓ Swin Transformer Block x 2 = 2개의 연속적인 Swin Transformer Blocks



Backbone Network

❖ Swin Transformer: Hierarchical Vision Transformer using Shifted Windows

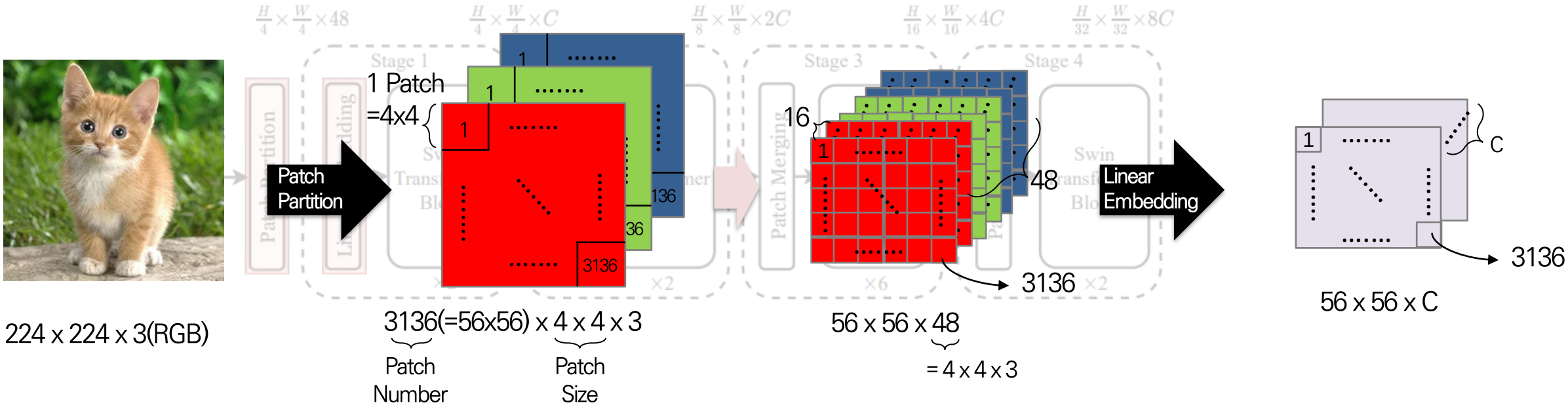
- Patch Partition & Linear Embeding
  - ✓ Patch Partition : 4 x 4 Pixel size의 Patch로 나눈 뒤, 채널 축으로 Concat
  - ✓ Linear Embedding : C차원으로 projection 시킴 (Swin-Tiny의 경우 C=96)



Backbone Network

❖ Swin Transformer: Hierarchical Vision Transformer using Shifted Windows

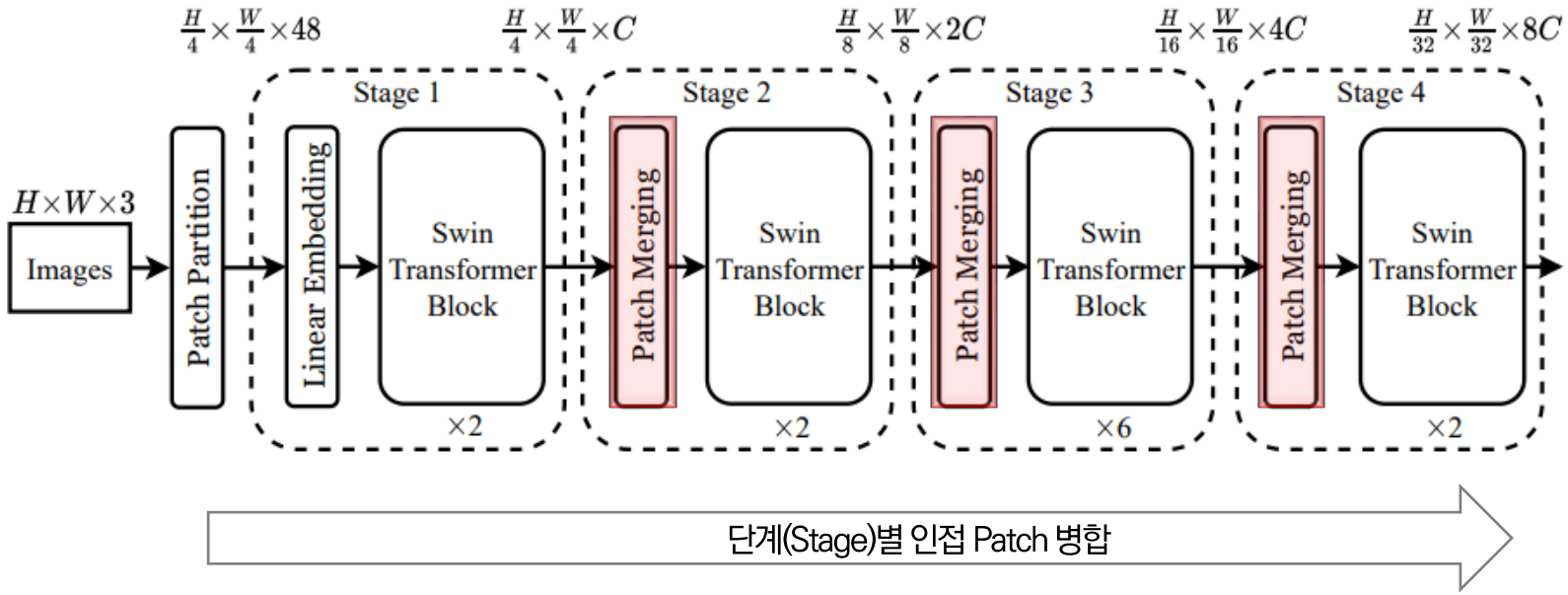
- Patch Partition & Linear Embeding
  - ✓ Patch Partition : 4 x 4 Pixel size의 Patch로 나눈 뒤, 채널 축으로 Concat
  - ✓ Linear Embedding : C차원으로 projection 시킴 (Swin-Tiny의 경우 C=96)

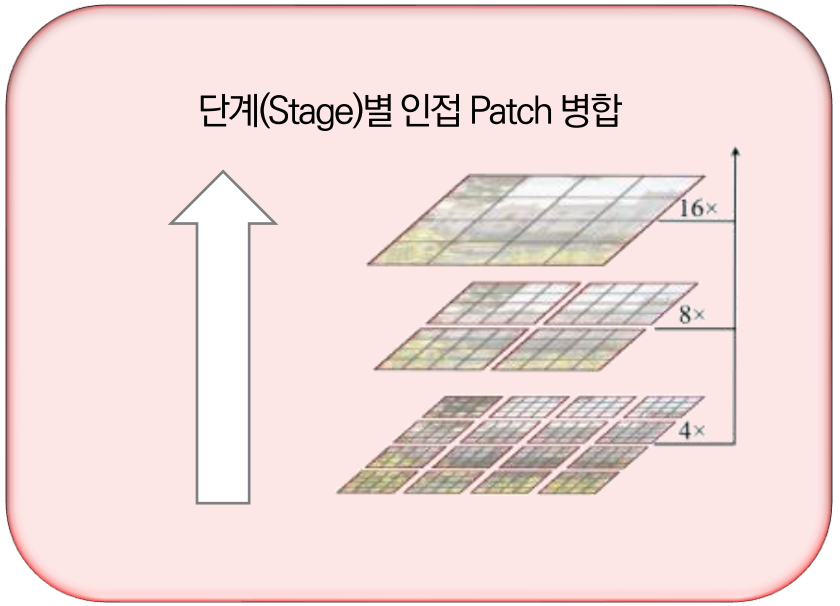


Backbone Network

❖ Swin Transformer: Hierarchical Vision Transformer using Shifted Windows

- Patch Merging
  - ✓ 단계(Stage)별 인접 Patch를 병합하여 해상도를 줄여주는 과정
  - ✓ Patch Merging = Merge + Linear Layer

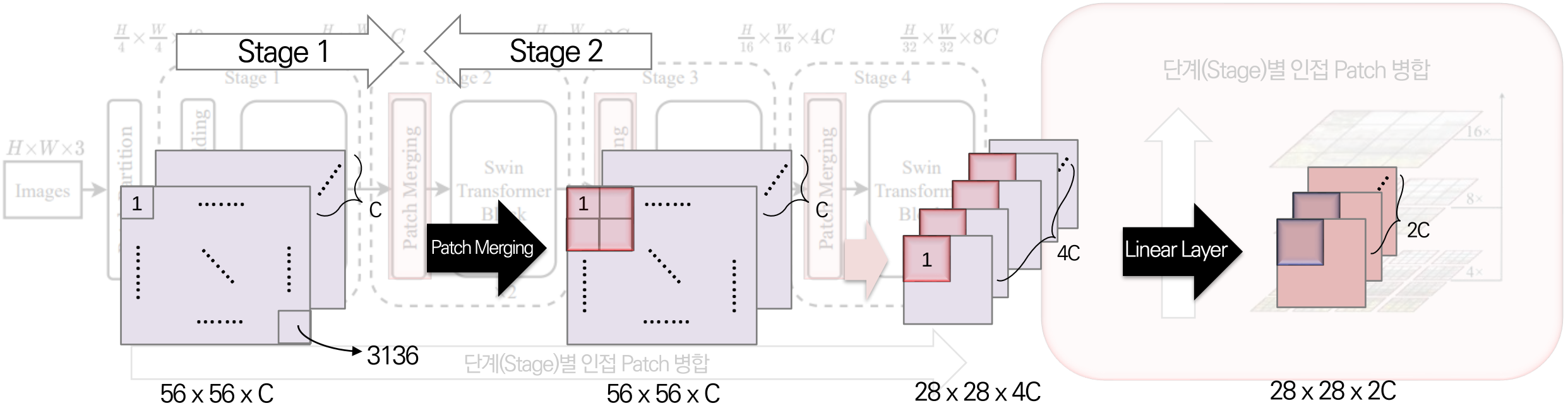




Backbone Network

❖ Swin Transformer: Hierarchical Vision Transformer using Shifted Windows

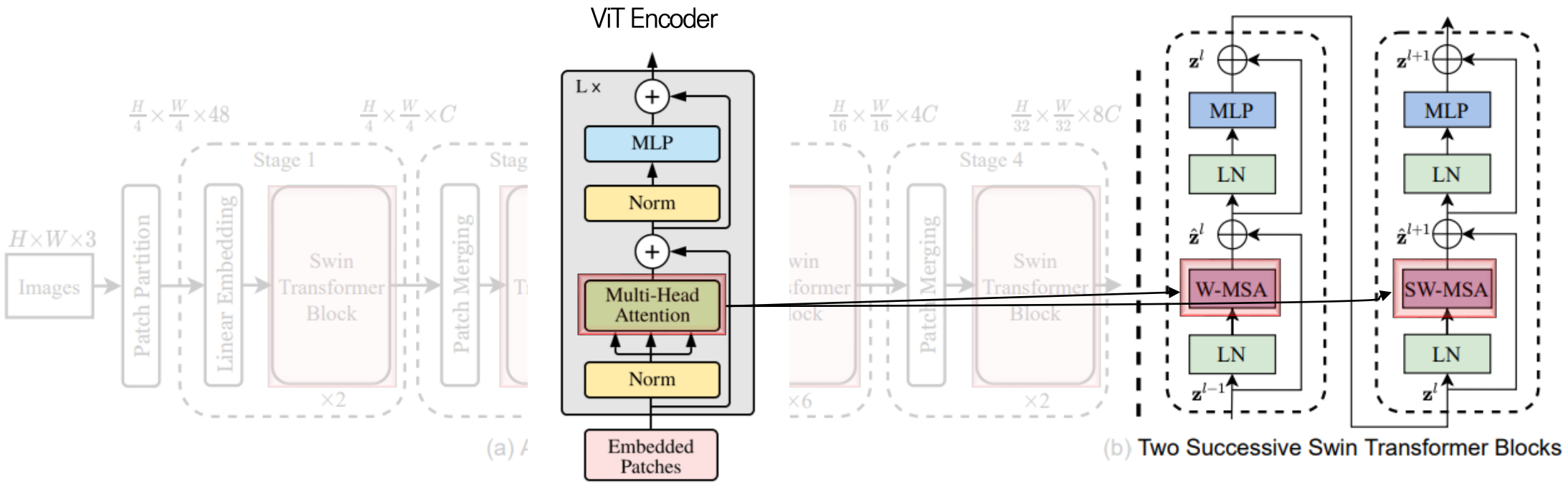
- Patch Merging
  - ✓ 단계(Stage)별 인접 Patch를 병합하여 해상도를 줄여주는 과정
  - ✓ Patch Merging = Merge + Linear Layer



Backbone Network

❖ Swin Transformer: Hierarchical Vision Transformer using Shifted Windows

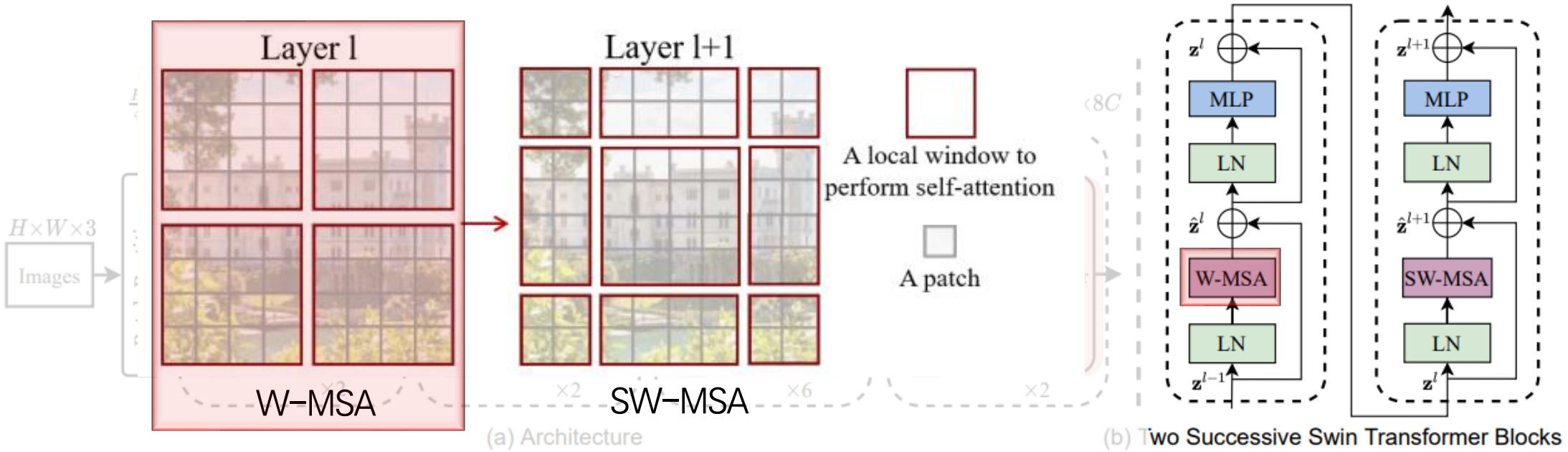
- Swin Transformer Block의 W-MSA & SW-MSA
  - ✓ 기존 ViT Encoder의 Multi-Head Attention을 대체 함



Backbone Network

❖ Swin Transformer: Hierarchical Vision Transformer using Shifted Windows

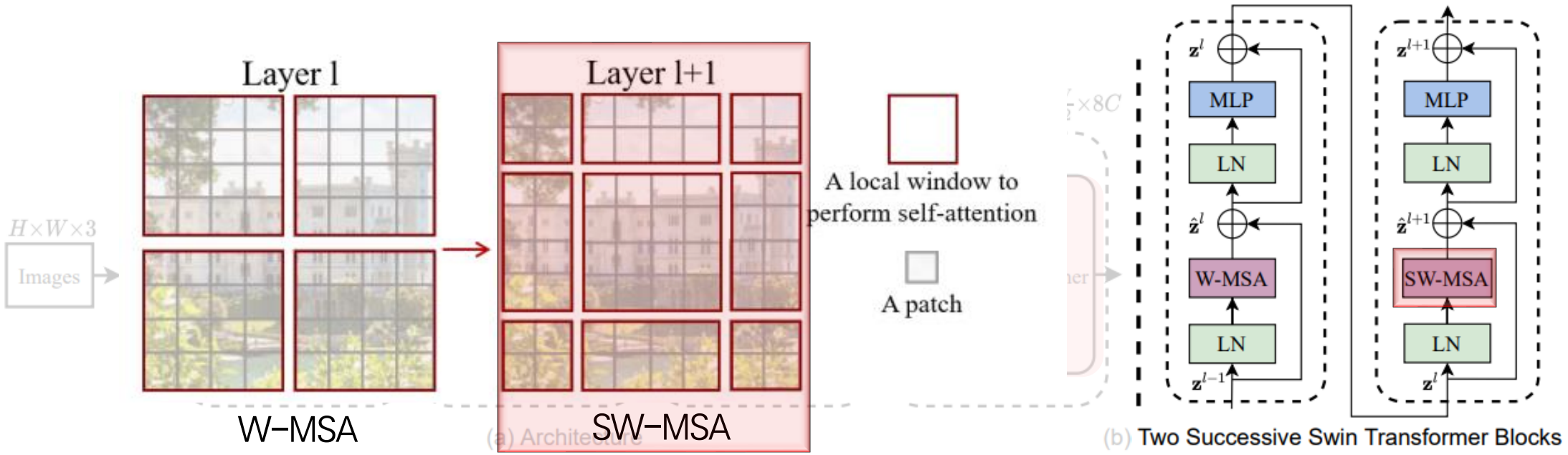
- Swin Transformer Block의 W-MSA & SW-MSA
  - ✓ W-MSA(Window Multi-head Self Attention) : Local Window 안에서의 self-attention
  - ✓ SW-MSA(Shifted Window Multi-head Self Attention) : Local Window 간의 self-attention



Backbone Network

❖ Swin Transformer: Hierarchical Vision Transformer using Shifted Windows

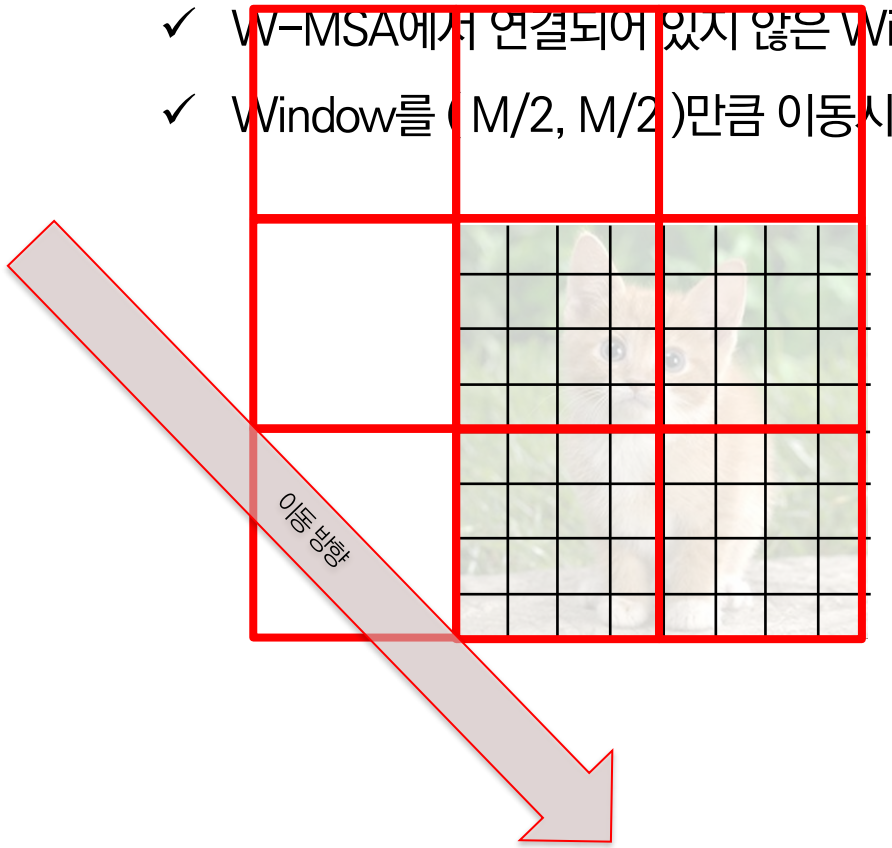
- Swin Transformer Block의 W-MSA & SW-MSA
  - ✓ W-MSA(Window Multi-head Self Attention) : Local Window 안에서의 self-attention
  - ✓ SW-MSA(Shifted Window Multi-head Self Attention) : Local Window 간의 self-attention

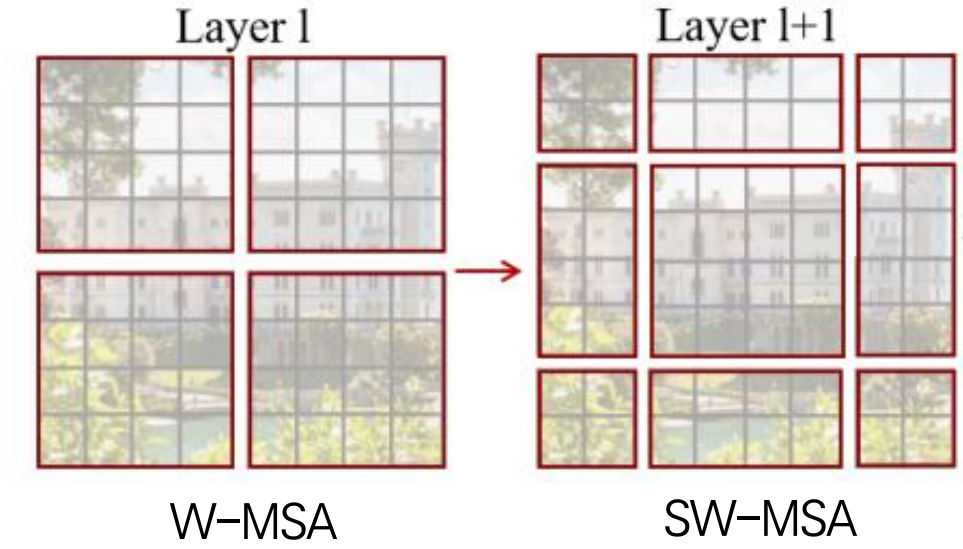


Backbone Network

❖ Swin Transformer: Hierarchical Vision Transformer using Shifted Windows

- W-MSA → SW-MSA 직관적인 이해
  - ✓ W-MSA에서 연결되어 있지 않은 Window들이 SW-MSA에서 연결되어 학습
  - ✓ Window를 (M/2, M/2)만큼 이동시켜 분할



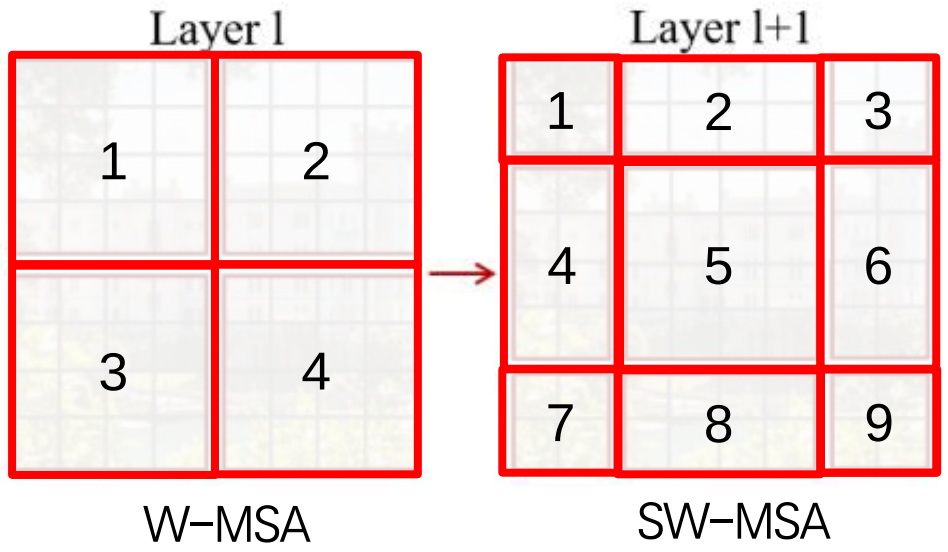


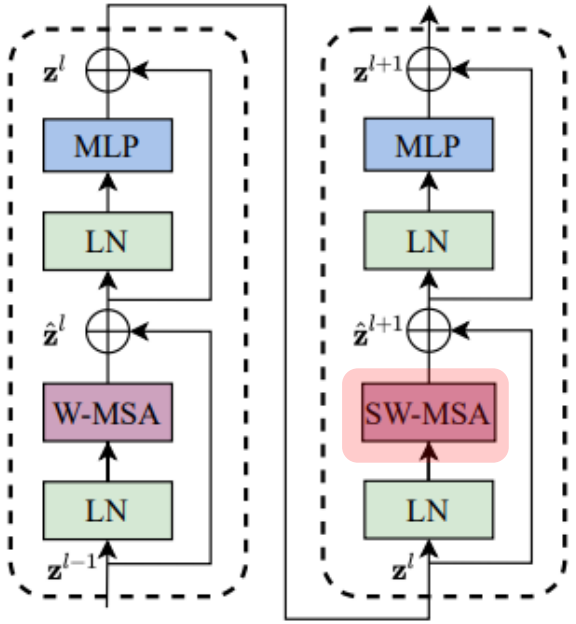
Backbone Network

❖ Swin Transformer: Hierarchical Vision Transformer using Shifted Windows

- Swin Transformer Block의 SW-MSA(Shifted Window Multi-head Self Attention)
  - ✓ W-MSA에서 연결되어 있지 않은 Window들이 SW-MSA에서 연결되어 학습
  - ✓ Window를 (M/2, M/2)만큼 이동시켜 분할

→ 문제 발생 : Local Window 4개 → 9개, 기존 윈도우 사이즈보다 작은 window가 생성 (계산 효율성 ↓)

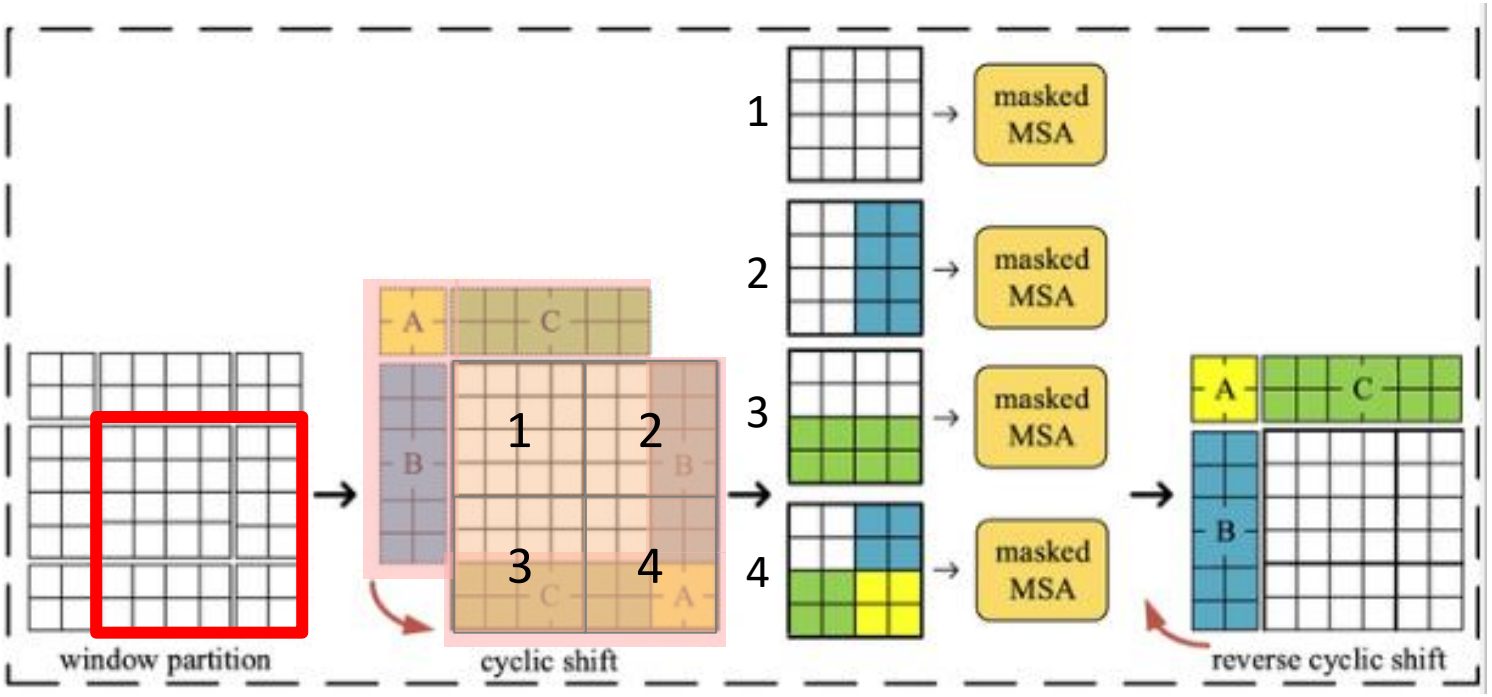


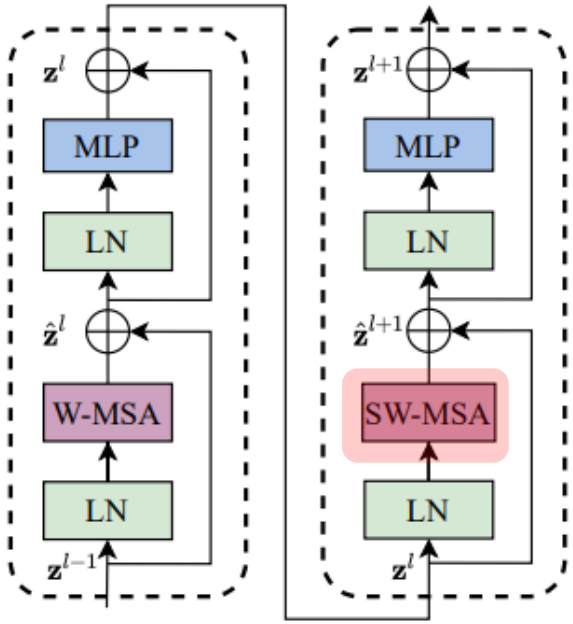


Backbone Network

❖ Swin Transformer: Hierarchical Vision Transformer using Shifted Windows

- SW-MSA의 Cyclic Shift & AttentionMask → W-MSA와 동일한 window개수 사용
  - ✓ Cyclic Shift : M x M 보다 작은 사이즈의 Window인 A, B, C를 모아서 기존과 동일한 크기로 만듬
  - ✓ AttentionMask : A, B, C는 Local Window Self-attention 계산 시 mask를 적용하여 연산이 수행되지 않게 함



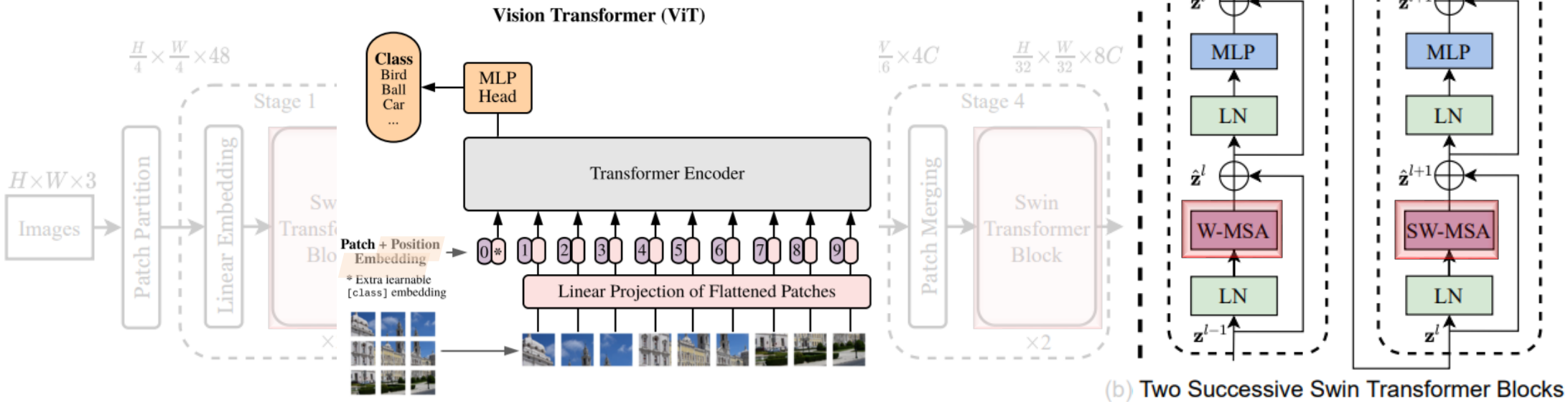


Backbone Network

❖ Swin Transformer: Hierarchical Vision Transformer using Shifted Windows

- W-MSA & SW-MSA의 Self-attention 계산 시
  - ✓ Relative position bias : 상대좌표를 더해주는 것으로 기존 절대 좌표인 Positional Encoding 보다 좋은 성능을 보임

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



Backbone Network

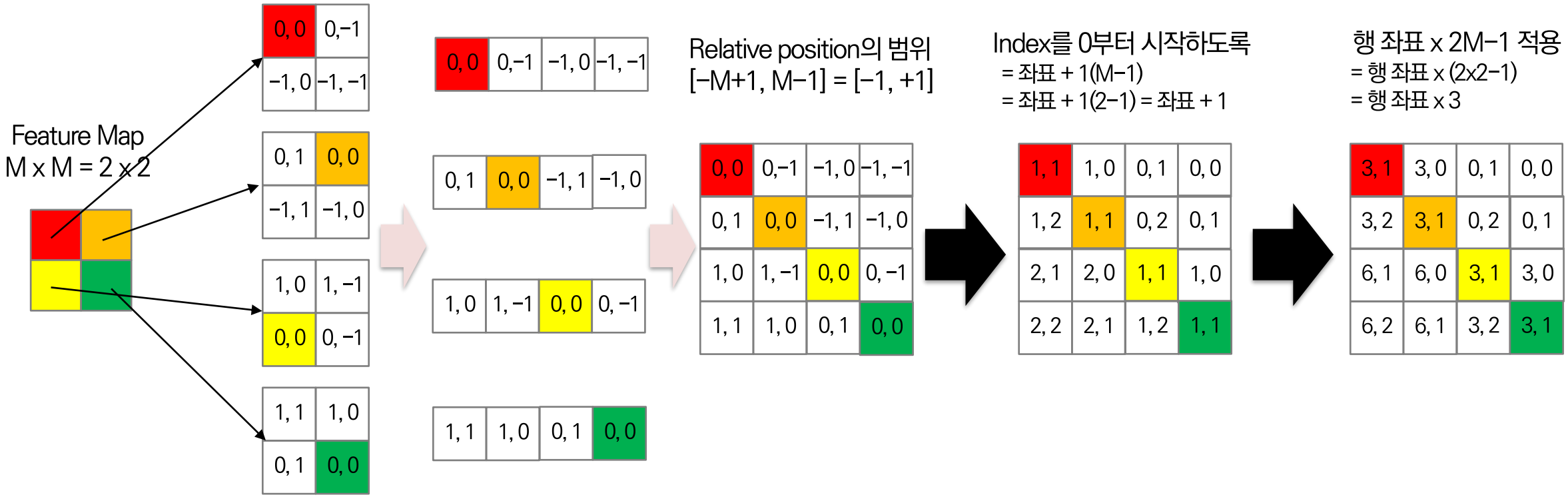
❖ Swin Transformer: Hierarchical Vision Transformer using Shifted Windows

- W-MSA & SW-MSA의 Self-attention 계산 시

✓ Relative position bias

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

(→, ↓ =−1 , ← ↑ =+1)



Backbone Network

❖ Swin Transformer: Hierarchical Vision Transformer using Shifted Windows

- W-MSA & SW-MSA의 Self-attention 계산 시
  - ✓ Relative position bias

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

행 좌표 x 2M-1 적용
= 행 좌표 x (2x2-1)
= 행 좌표 x 3

3, 1	3, 0	0, 1	0, 0
3, 2	3, 1	0, 2	0, 1
6, 1	6, 0	3, 1	3, 0
6, 2	6, 1	3, 2	3, 1

Relative position Index

4	3	1	0
5	4	2	1
7	6	4	3
8	7	5	4

Relative position bias table (=$\hat{B}$)

Index	bias
0	0.1
1	0.2
2	0.3
3	0.8
4	0.1
5	0.6
6	0.4
7	0.4
8	0.7

Relative position bias(=B)

0.1	0.8	0.2	0.1
0.6	0.1	0.3	0.2
0.4	0.4	0.1	0.8
0.7	0.4	0.6	0.1

Backbone Network

❖ Swin Transformer: Hierarchical Vision Transformer using Shifted Windows

- Data Shape Example

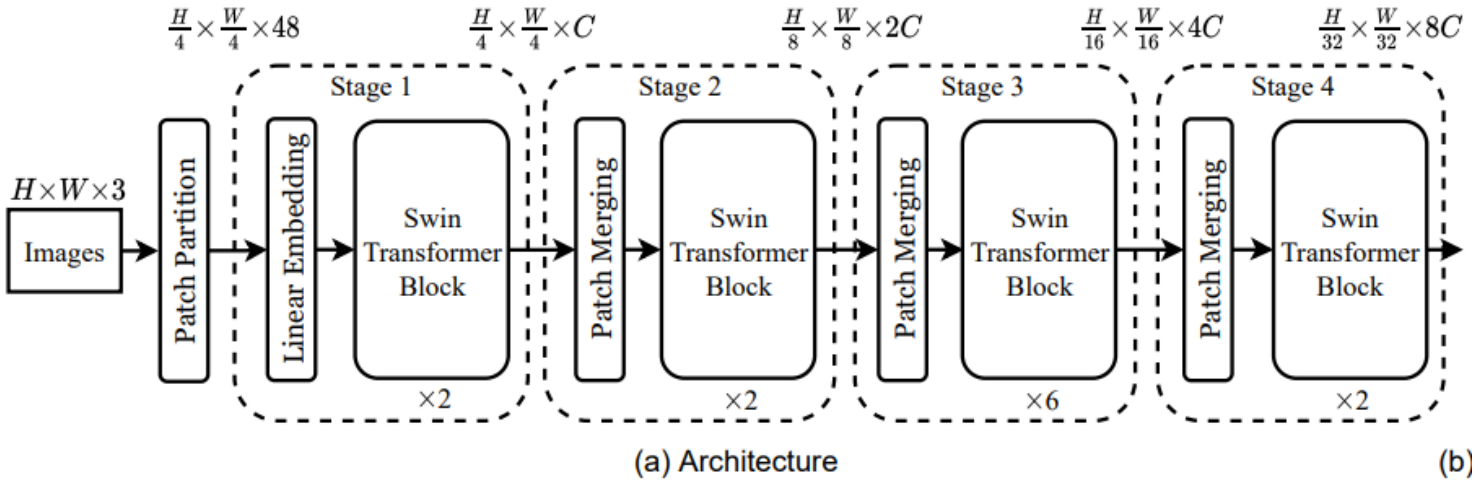


Image 224 x 224 x 3
→ (Patch Partition=Flatten) 56 x 56 x 48(= 4 x 4 x 3)

Stage 1.
→ (Linear Embedding=C 차원으로 투영) 56 x 56 x C

Stage 2.
→ (Patch Merging-1. 2x2 인접 patch끼리 merge) 28 x 28 x 4C
→ (Patch Merging-2. linear layer 통과해서 C 조정) 28 x 28 x 2C

Stage 3.
→ (Patch Merging-1. 2x2 인접 patch끼리 merge) 14 x 14 x 8C
→ (Patch Merging-2. linear layer 통과해서 C 조정) 14 x 14 x 4C

Stage 4.
→ (Patch Merging-1. 2x2 인접 patch끼리 merge) 7 x 7 x 16C
→ (Patch Merging-2. linear layer 통과해서 C 조정) 7 x 7 x 8C

Backbone Network

❖ Swin Transformer: Hierarchical Vision Transformer using Shifted Windows

- Architecture Variants
  - ✓ Image Size = 224x224 예시

	downsp. rate (output size)	Swin-T	Swin-S	Swin-B	Swin-L
stage 1	4× (56×56)	concat 4×4, 96-d, LN	concat 4×4, 96-d, LN	concat 4×4, 128-d, LN	concat 4×4, 192-d, LN
		[win. sz. 7×7, dim 96, head 3] × 2	[win. sz. 7×7, dim 96, head 3] × 2	[win. sz. 7×7, dim 128, head 4] × 2	[win. sz. 7×7, dim 192, head 6] × 2
stage 2	8× (28×28)	concat 2×2, 192-d , LN	concat 2×2, 192-d , LN	concat 2×2, 256-d , LN	concat 2×2, 384-d , LN
		[win. sz. 7×7, dim 192, head 6] × 2	[win. sz. 7×7, dim 192, head 6] × 2	[win. sz. 7×7, dim 256, head 8] × 2	[win. sz. 7×7, dim 384, head 12] × 2
stage 3	16× (14×14)	concat 2×2, 384-d , LN	concat 2×2, 384-d , LN	concat 2×2, 512-d , LN	concat 2×2, 768-d , LN
		[win. sz. 7×7, dim 384, head 12] × 6	[win. sz. 7×7, dim 384, head 12] × 18	[win. sz. 7×7, dim 512, head 16] × 18	[win. sz. 7×7, dim 768, head 24] × 18
stage 4	32× (7×7)	concat 2×2, 768-d , LN	concat 2×2, 768-d , LN	concat 2×2, 1024-d , LN	concat 2×2, 1536-d , LN
		[win. sz. 7×7, dim 768, head 24] × 2	[win. sz. 7×7, dim 768, head 24] × 2	[win. sz. 7×7, dim 1024, head 32] × 2	[win. sz. 7×7, dim 1536, head 48] × 2

Backbone Network

❖ Swin Transformer: Hierarchical Vision Transformer using Shifted Windows

- Experiments – Classification

✓ Transformer 계열 비교

: 가장 높은 성능 달성

✓ CNN 계열 비교

: throughput 즉, 성능과 학습속도의 Trade off가 적음

(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

Backbone Network

❖ Swin Transformer: Hierarchical Vision Transformer using Shifted Windows

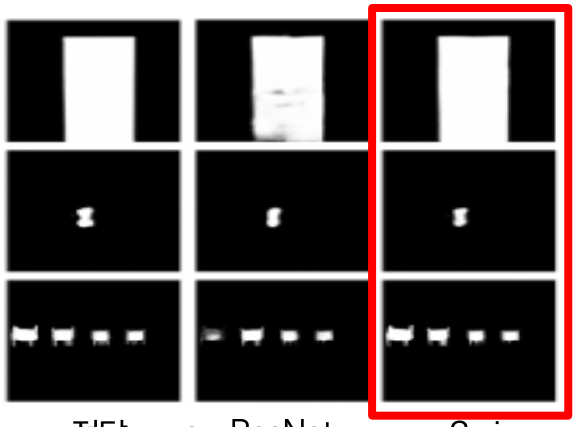
- Experiments – Object Detection

✓ 각 프레임 워크에서 백본 비교

: ReNet 50 대비 높은 성능 달성

✓ Cascade Mask R-CNN 구조에서 백본 비교

: 파라미터 개수로 구분 시 타 백본 대비 높은 성능 달성



정답 vs ResNet vs Swin

(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

Backbone Network

❖ Swin Transformer: Hierarchical Vision Transformer using Shifted Windows

- Experiments – Semantic Segmentation

✓ DeiT-S와 비교 시

: 적은 파라미터로 더 높은 성능 달성 및

✓ 다른 Method + Backbone 비교 시

: 가장 높은 성능

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

Backbone Network

❖ Swin Transformer: Hierarchical Vision Transformer using Shifted Windows

- 결론
  - ✓ Transformer 모델에 Local Window를 적용하여 성능 향상이 됨
  - ✓ Patch Merging을 통해 Stage별 계층적인 구조 형성

    → ViT보다 이미지 특성이 고려되어 더 좋은 특징을 추출

    → ViT 보다 더 적은 연산량을 가짐

Backbone Network

❖ ConvNext: A ConvNet for the 2020s

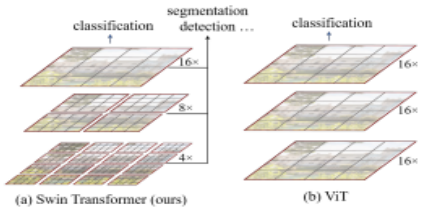
- CVPR (2022), 2022년 11월 22일 기준 485회 인용
- CNN 모델을 기반으로 하며, Swin Transformer의 장점을 반영한 구조로 높은 성능을 달성한 모델

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-zeliul,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 (87.3 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 hierarchical design and the shifted window approach also prove beneficial for all-MLP architectures. The code and models are publicly available at https://github.com/microsoft/Swin-Transformer.



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

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 [20] produce feature maps of a single low resolution and have quadratic computation complexity to input image size due to computation of self-attention globally.

greater scale [30, 76], more extensive connections [34], and more sophisticated forms of convolution [70, 18, 84]. With 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

Backbone Network

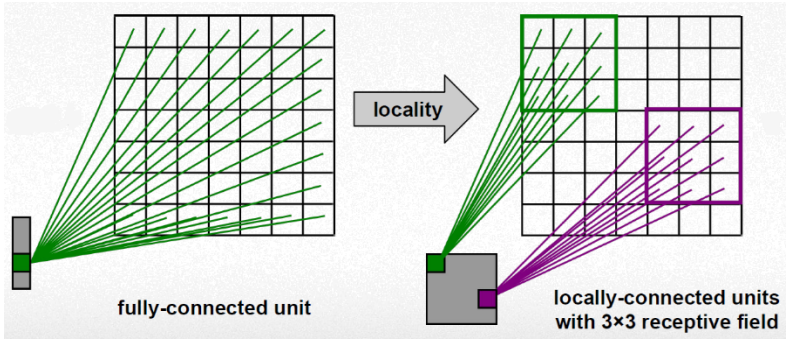
❖ ConvNext: A ConvNet for the 2020s

- 등장 배경
  - ✓ ViT 이후 Swin Transformer 등 Transformer에 연구가 집중되고 있음

    → CNN 모델은 Sliding Window를 통해 파라미터를 공유하여 Inductive bias를 통해 좋은 성능을 냄

    ➢ Inductive bias란?

    : 데이터에 대한 가정을 추가하여 더 좋은 성능을 유도 (ex. CNN의 경우 Locality, Transitional Invariance)



Locality
= 근접한 픽셀들끼리 종속성을 가짐

Transitional Invariance
= 어느 위치에 있건 출력은 불변

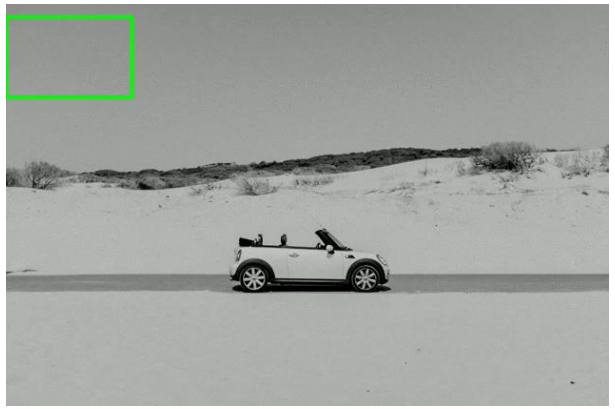
Backbone Network

❖ ConvNext: A ConvNet for the 2020s

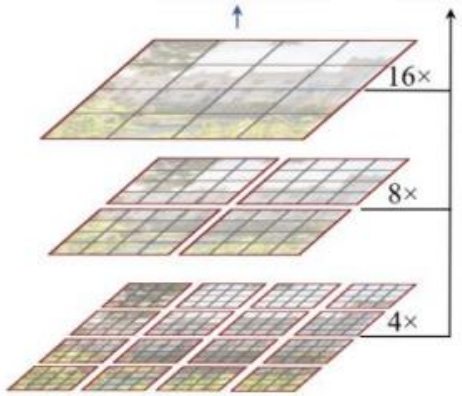
- 등장 배경
  - ✓ ViT 이후 Swin Transformer 등 Transformer에 연구가 집중되고 있음

    → CNN 모델은 Sliding Window를 통해 파라미터를 공유하여 Inductive bias를 통해 좋은 성능

    → 좋은 성능인 Swin Transformer의 Window 기법은 결국 CNN에서 가져온 컨셉으로 볼 수 있음



Sliding Window Approach for Object Detection



(a) Swin Transformer (ours)

https://towardsdatascience.com/understanding-regions-with-cnn-features-r-cnn-ec69c15f8ea7

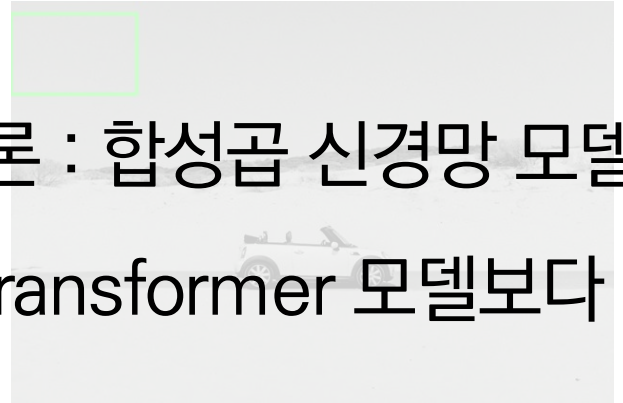
Backbone Network

❖ ConvNext: A ConvNet for the 2020s

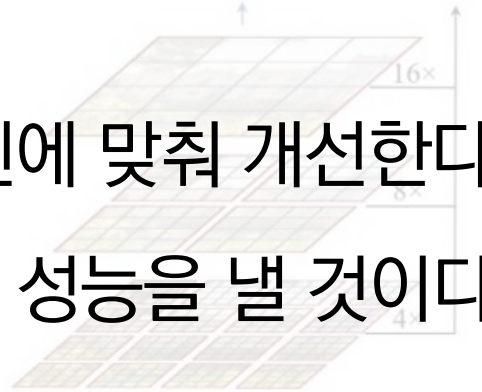
- 등장 배경
  - ✓ ViT 이후 Swin Transformer 등 Transformer에 연구가 집중되고 있음

    → CNN 모델은 Sliding Window를 통해 파라미터를 공유하여 Inductive bias를 통해 좋은 성능

    → 좋은 성능인 Swin Transformer의 Window 기법은 결국 CNN에서 가져온 컨셉으로 볼 수 있음



Sliding Window Approach for Object Detection



(a) Swin Transformer (ours)

결론 : 합성곱 신경망 모델을 최신에 맞춰 개선한다면

Transformer 모델보다 더 좋은 성능을 낼 것이다.

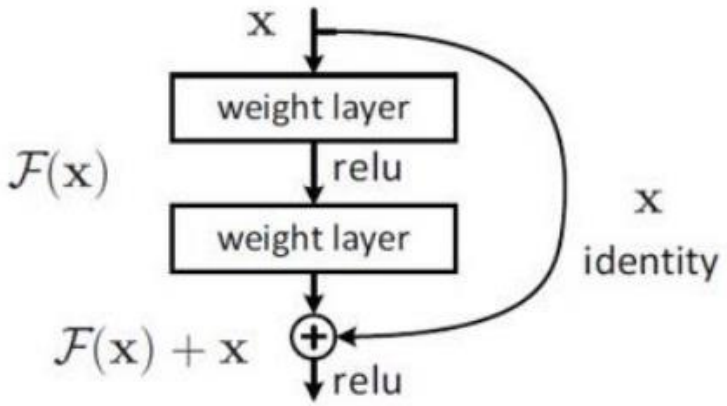
https://towardsdatascience.com/understanding-regions-with-cnn-features-r-cnn-ec69c15f8ea7

Backbone Network

❖ ConvNext: A ConvNet for the 2020s

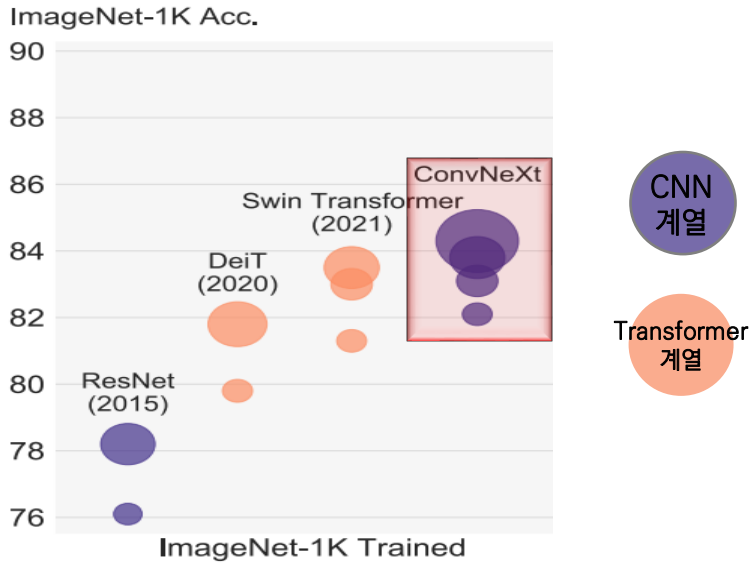
- 제안 방법론
  - ✓ ResNet50을 기반으로 Transformer와 유사하게 Modernizing(최신화)하여 성능을 높인 ConvNeXt 제안

    → Swin Transformer 과 비교했을 때 동일 연산량 대비 더 좋은 성능을 보임



CNN 기반 ResNet50

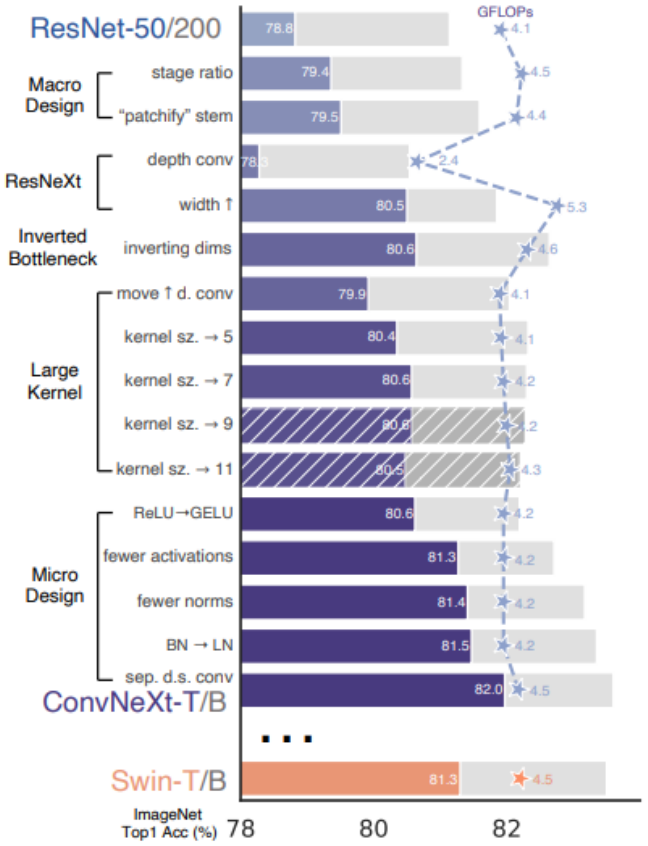
Modernizing



Backbone Network

❖ ConvNext: A ConvNet for the 2020s

- Modernizing
  - ✓ Attention modules을 넣지 않고 ResNet을 Swin Transformer와 같이 최신화 함
  - ✓ ResNet50에서 아래 전략을 적용해 Swin-T의 성능을 능가함
    - ① Macro Design
    - ② ResNeXt-ify
    - ③ Inverted Bottleneck
    - ④ Large Kernel Sizes
    - ⑤ Micro Design



Backbone Network

❖ ConvNext: A ConvNet for the 2020s

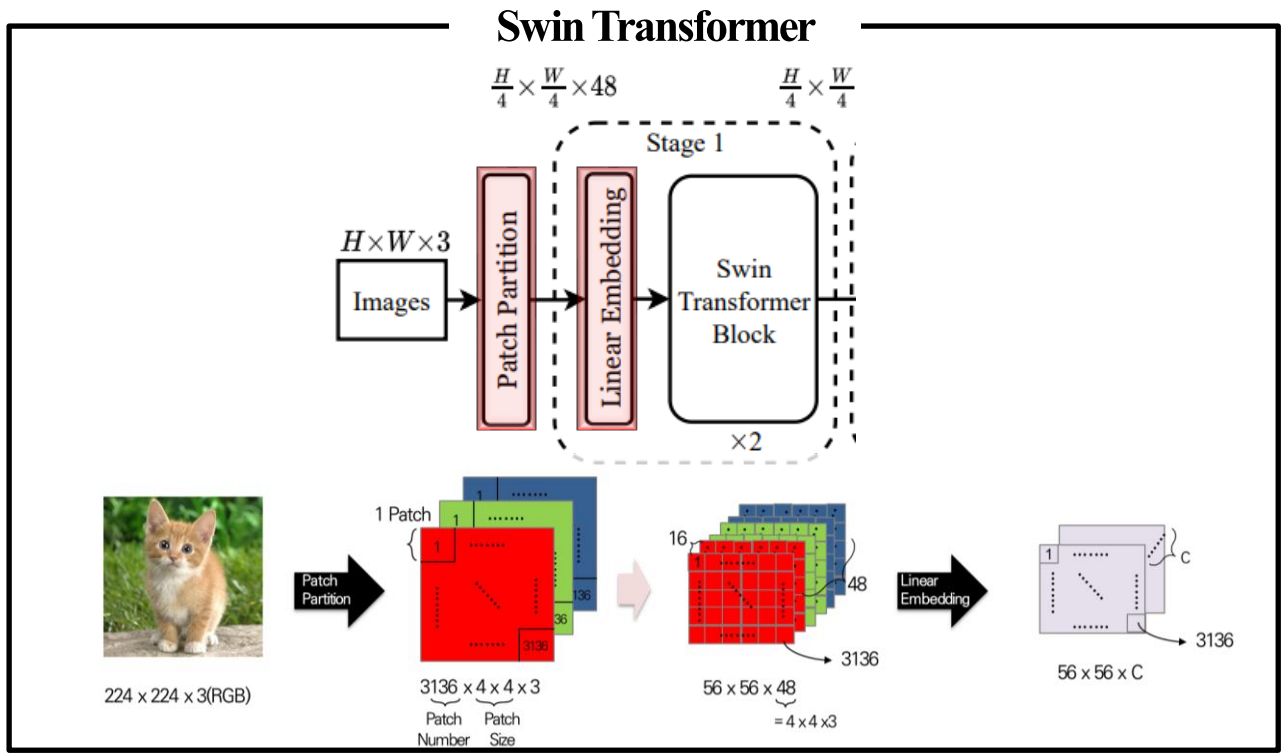
- Modernizing – ① Macro Design : stage ratio
  - ✓ Stage 구성을 Swin Transformer 비율인 1:1:3:1로 수정 (x 3, x 4, x 6, x 3 → x 3, x 3, x 9, x3)

	output size	• ResNet-50	• ConvNeXt-T	○ Swin-T
stem	56×56	7×7, 64, stride 2 3×3 max pool, stride 2	4×4, 96, stride 4	4×4, 96, stride 4
res2	56×56	[1×1, 64 3×3, 64 1×1, 256] × 3	[d7×7, 96 1×1, 384 1×1, 96] × 3	[1×1, 96×3 MSA, w7×7, H=3, rel. pos. 1×1, 96 1×1, 384 1×1, 96] × 2
res3	28×28	[1×1, 128 3×3, 128 1×1, 512] × 4	[d7×7, 192 1×1, 768 1×1, 192] × 3	[1×1, 192×3 MSA, w7×7, H=6, rel. pos. 1×1, 192 1×1, 768 1×1, 192] × 2
res4	14×14	[1×1, 256 3×3, 256 1×1, 1024] × 6	[d7×7, 384 1×1, 1536 1×1, 384] × 9	[1×1, 384×3 MSA, w7×7, H=12, rel. pos. 1×1, 384 1×1, 1536 1×1, 384] × 6
res5	7×7	[1×1, 512 3×3, 512 1×1, 2048] × 3	[d7×7, 768 1×1, 3072 1×1, 768] × 3	[1×1, 768×3 MSA, w7×7, H=24, rel. pos. 1×1, 768 1×1, 3072 1×1, 768] × 2
FLOPs		4.1×10^9	4.5×10^9	4.5×10^9
# params.		25.6×10^6	28.6×10^6	28.3×10^6

Backbone Network

❖ ConvNext: A ConvNet for the 2020s

- Modernizing – ① Macro Design : patchify stem
  - ✓ Swin Transformer의 Patch Partition과 같이 Patchify 수행 (7 x 7, stride 2→ 4 x 4, stride 4)

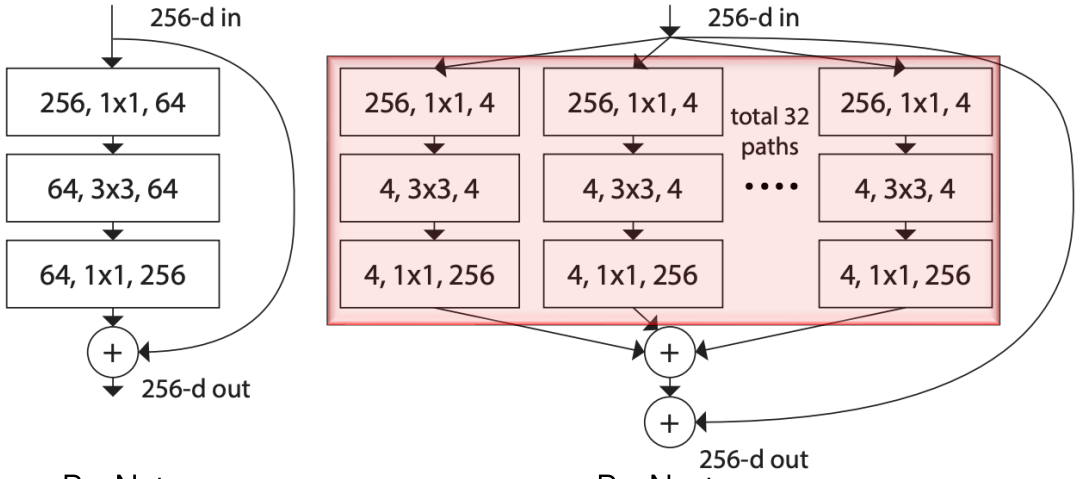


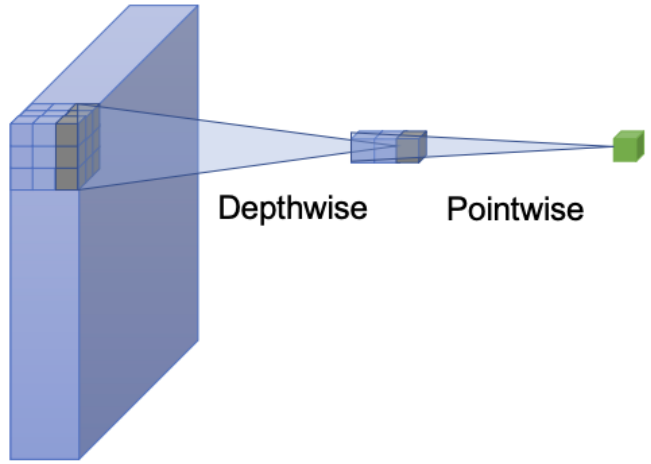
	output size	• ResNet-50	• ConvNeXt-T	○ Swin-T
stem	56×56	7×7, 64, stride 2 3×3 max pool, stride 2	4×4, 96, stride 4	4×4, 96, stride 4
res2	56×56	[1×1, 64 3×3, 64 1×1, 256] × 3	[d7×7, 96 1×1, 384 1×1, 96] × 3	[1×1, 96×3 MSA, w7×7, H=3, rel. pos. 1×1, 96 1×1, 384 1×1, 96] × 2
res3	28×28	[1×1, 128 3×3, 128 1×1, 512] × 4	[d7×7, 192 1×1, 768 1×1, 192] × 3	[1×1, 192×3 MSA, w7×7, H=6, rel. pos. 1×1, 192 1×1, 768 1×1, 192] × 2
res4	14×14	[1×1, 256 3×3, 256 1×1, 1024] × 6	[d7×7, 384 1×1, 1536 1×1, 384] × 9	[1×1, 384×3 MSA, w7×7, H=12, rel. pos. 1×1, 384 1×1, 1536 1×1, 384] × 6
res5	7×7	[1×1, 512 3×3, 512 1×1, 2048] × 3	[d7×7, 768 1×1, 3072 1×1, 768] × 3	[1×1, 768×3 MSA, w7×7, H=24, rel. pos. 1×1, 768 1×1, 3072 1×1, 768] × 2
FLOPs		4.1×10^9	4.5×10^9	4.5×10^9
# params.		25.6×10^6	28.6×10^6	28.3×10^6

Backbone Network

❖ ConvNext: A ConvNet for the 2020s

- Modernizing – ② ResNeXt : depth conv
  - ✓ Vanila ResNet이 아닌 ResNext을 사용 (Grouped convolution = ex. 32개의 그룹)
  - ✓ Depthwise seperable convolution을 사용해서 계산량 ↓
    - ➢ 차원 별 가중합이기 때문에 Self-attention과 유사한 연산을 수행



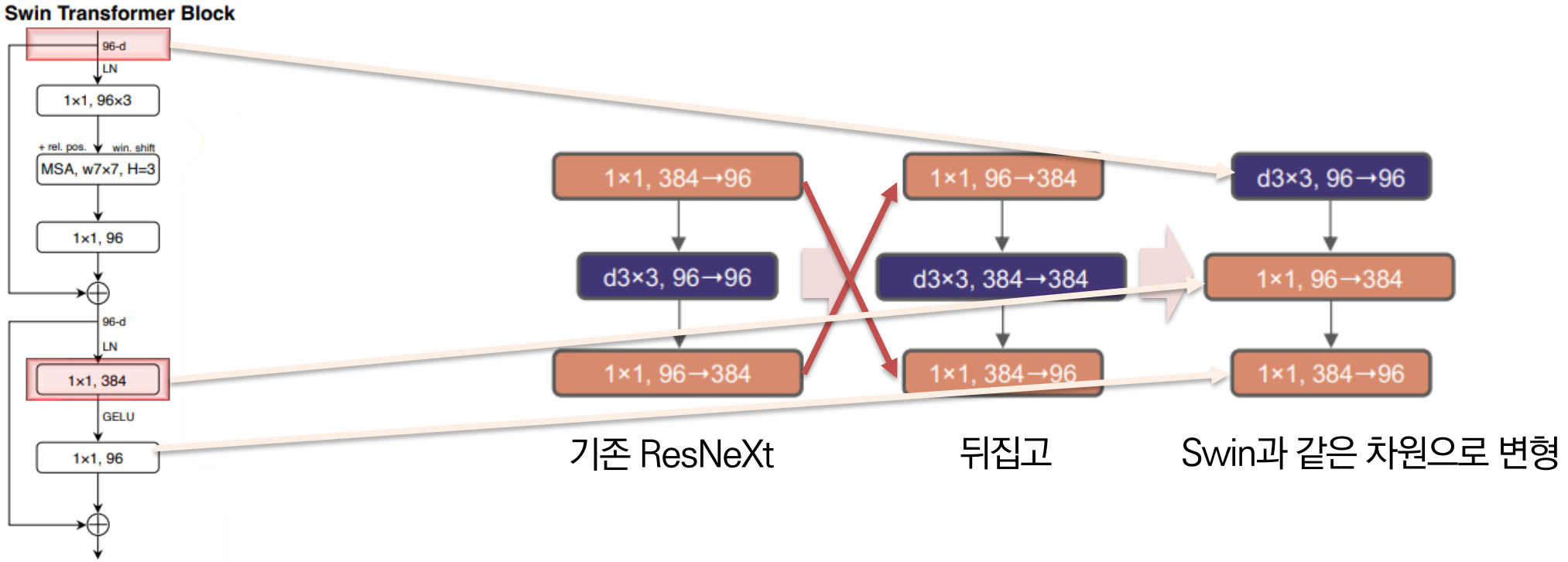


Depthwise seperable convolution

Backbone Network

❖ ConvNext: A ConvNet for the 2020s

- Modernizing – ③ Inverted Bottleneck & ④ Kernel Size
  - ✓ Swin Transformer는 MLP Block hidden dim이 Input dim의 4배
  - ✓ ResNext를 Swin과 동일하게 변형



Backbone Network

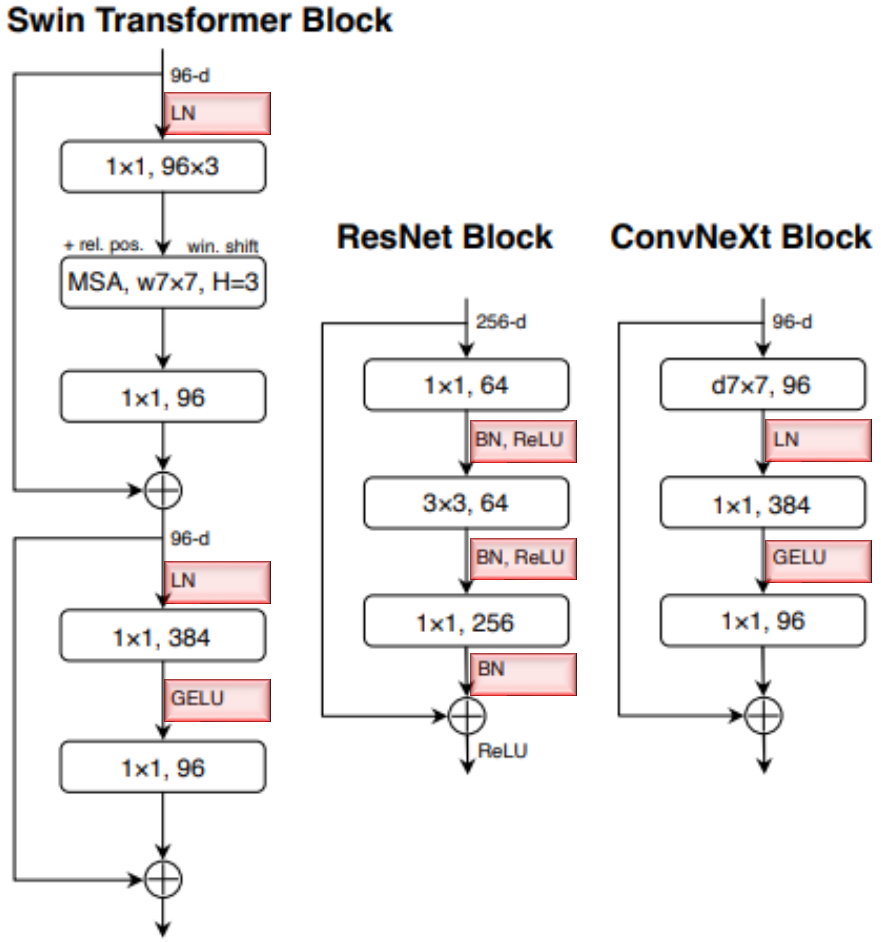
❖ ConvNext: A ConvNet for the 2020s

- Modernizing – ③ Inverted Bottleneck & ④ Kernel Size
  - ✓ ResNext에서 사용된 3 x 3 kernel size → Swin Window size인 7 x 7 로 변경

Backbone Network

❖ ConvNext: A ConvNet for the 2020s

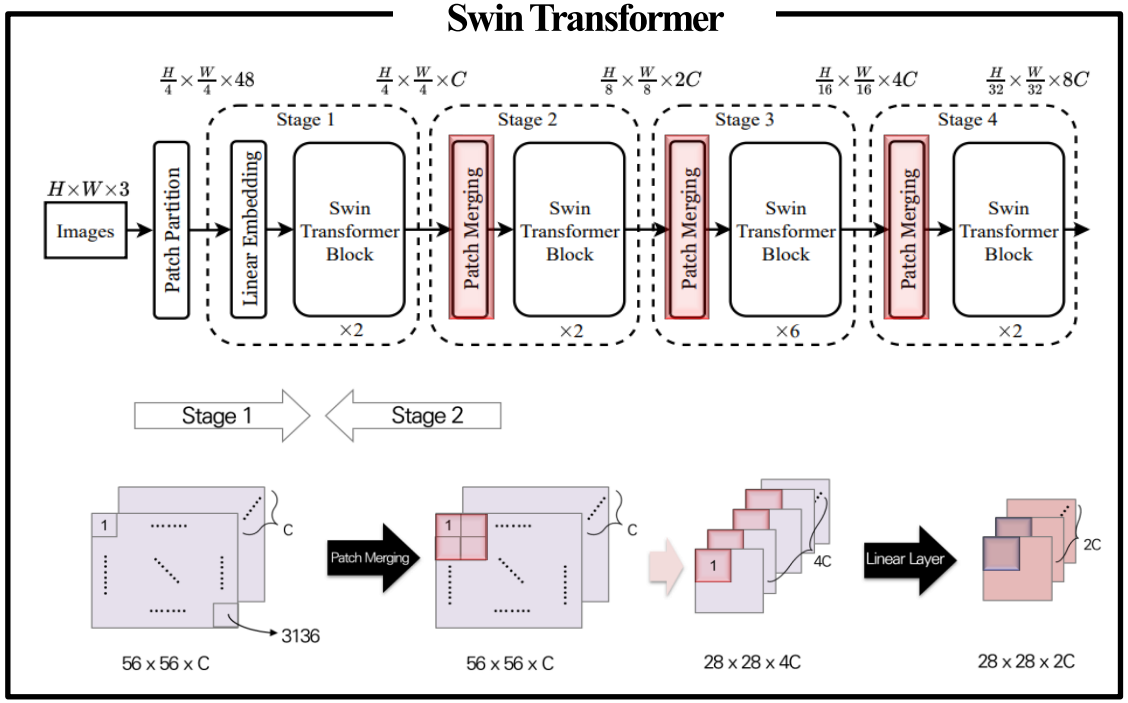
- Modernizing – ⑤ Micro Design
  - ✓ ReLU → GELU
  - ✓ Fewer Activations & Norm
    - ➢ 한번만 적용하는 것으로 변경
  - ✓ Batch Normalization → Layer Normalization



Backbone Network

❖ ConvNext: A ConvNet for the 2020s

- Modernizing – ⑤ Micro Design : Down-sampling Layer 를 매 Stage 도입
  - ✓ Swin의 Stage 사이 down sampling = Patch Merging
    - ➢ 유사한 2 x 2 conv with stride 2를 Convolution Layer 이후 적용



	output size	• ResNet-50	• ConvNeXt-T	○ Swin-T
stem	56×56	7×7, 64, stride 2 3×3 max pool, stride 2	4×4, 96, stride 4	4×4, 96, stride 4
res2	56×56	[1×1, 64 3×3, 64 1×1, 256] × 3	[d7×7, 96 1×1, 384 1×1, 96] × 3	[1×1, 96×3 MSA, w7×7, H=3, rel. pos. 1×1, 96 1×1, 384 1×1, 96] × 2
res3	28×28	[1×1, 128 3×3, 128 1×1, 512] × 4	[d7×7, 192 1×1, 768 1×1, 192] × 3	[1×1, 192×3 MSA, w7×7, H=6, rel. pos. 1×1, 192 1×1, 768 1×1, 192] × 2
res4	14×14	[1×1, 256 3×3, 256 1×1, 1024] × 6	[d7×7, 384 1×1, 1536 1×1, 384] × 9	[1×1, 384×3 MSA, w7×7, H=12, rel. pos. 1×1, 384 1×1, 1536 1×1, 384] × 6
res5	7×7	[1×1, 512 3×3, 512 1×1, 2048] × 3	[d7×7, 768 1×1, 3072 1×1, 768] × 3	[1×1, 768×3 MSA, w7×7, H=24, rel. pos. 1×1, 768 1×1, 3072 1×1, 768] × 2
FLOPs		4.1×10^9	4.5×10^9	4.5×10^9
# params.		25.6×10^6	28.6×10^6	28.3×10^6

Backbone Network

❖ ConvNext: A ConvNet for the 2020s

- Architecture Variants
  - ✓ C = channels in each stage
  - ✓ B = number of blocks in each stage

- ConvNeXt-T: $C = (96, 192, 384, 768), B = (3, 3, 9, 3)$
- ConvNeXt-S: $C = (96, 192, 384, 768), B = (3, 3, 27, 3)$
- ConvNeXt-B: $C = (128, 256, 512, 1024), B = (3, 3, 27, 3)$
- ConvNeXt-L: $C = (192, 384, 768, 1536), B = (3, 3, 27, 3)$
- ConvNeXt-XL: $C = (256, 512, 1024, 2048), B = (3, 3, 27, 3)$

	output size	• ConvNeXt-T
stem	56×56	4×4, 96, stride 4
res2	56×56	$\begin{bmatrix} \text{d}7\times7, 96 \\ 1\times1, 384 \\ 1\times1, 96 \end{bmatrix} \times 3$
res3	28×28	$\begin{bmatrix} \text{d}7\times7, 192 \\ 1\times1, 768 \\ 1\times1, 192 \end{bmatrix} \times 3$
res4	14×14	$\begin{bmatrix} \text{d}7\times7, 384 \\ 1\times1, 1536 \\ 1\times1, 384 \end{bmatrix} \times 9$
res5	7×7	$\begin{bmatrix} \text{d}7\times7, 768 \\ 1\times1, 3072 \\ 1\times1, 768 \end{bmatrix} \times 3$
FLOPs		4.5×10^9
# params.		28.6×10^6

Backbone Network

❖ ConvNext: A ConvNet for the 2020s

- Experiments – Classification

✓ Transformer 계열과 비교

: 유사한 파라미터를 가진 모델 버전에서 더 높은 성능 달성

✓ CNN 계열과 비교

: throughput 즉, 성능과 학습속도의 Trade off가 적음

model	image size	#param.	FLOPs	throughput (image / s)	IN-1K top-1 acc.
ImageNet-1K trained models					
• RegNetY-16G [54]	224^2	84M	16.0G	334.7	82.9
• EffNet-B7 [71]	600^2	66M	37.0G	55.1	84.3
• EffNetV2-L [72]	480^2	120M	53.0G	83.7	85.7
○ DeiT-S [73]	224^2	22M	4.6G	978.5	79.8
○ DeiT-B [73]	224^2	87M	17.6G	302.1	81.8
○ Swin-T	224^2	28M	4.5G	757.9	81.3
• ConvNeXt-T	224^2	29M	4.5G	774.7	**82.1**
○ Swin-S	224^2	50M	8.7G	436.7	83.0
• ConvNeXt-S	224^2	50M	8.7G	447.1	**83.1**
○ Swin-B	224^2	88M	15.4G	286.6	83.5
• ConvNeXt-B	224^2	89M	15.4G	292.1	**83.8**
○ Swin-B	384^2	88M	47.1G	85.1	84.5
• ConvNeXt-B	384^2	89M	45.0G	95.7	**85.1**
• ConvNeXt-L	224^2	198M	34.4G	146.8	**84.3**
• ConvNeXt-L	384^2	198M	101.0G	50.4	**85.5**

Backbone Network

❖ ConvNext: A ConvNet for the 2020s

- Experiments – Object Detection and Segmentation

✓ 각 프레임 워크에서 백본 비교

: 대등하거나 더 좋은 성능을 보임

backbone	FLOPs	FPS	AP^{box}	AP^{box}_{50}	AP^{box}_{75}	AP^{mask}	AP^{mask}_{50}	AP^{mask}_{75}
Mask-RCNN 3× schedule								
○ Swin-T	267G	23.1	46.0	68.1	50.3	41.6	65.1	44.9
● ConvNeXt-T	262G	25.6	**46.2**	67.9	50.8	**41.7**	65.0	44.9
Cascade Mask-RCNN 3× schedule								
● ResNet-50	739G	16.2	46.3	64.3	50.5	40.1	61.7	43.4
● X101-32	819G	13.8	48.1	66.5	52.4	41.6	63.9	45.2
● X101-64	972G	12.6	48.3	66.4	52.3	41.7	64.0	45.1
○ Swin-T	745G	12.2	50.4	69.2	54.7	43.7	66.6	47.3
● ConvNeXt-T	741G	13.5	**50.4**	69.1	54.8	**43.7**	66.5	47.3
○ Swin-S	838G	11.4	51.9	70.7	56.3	45.0	68.2	48.8
● ConvNeXt-S	827G	12.0	**51.9**	70.8	56.5	**45.0**	68.4	49.1
○ Swin-B	982G	10.7	51.9	70.5	56.4	45.0	68.1	48.9
● ConvNeXt-B	964G	11.4	**52.7**	71.3	57.2	**45.6**	68.9	49.5
○ Swin-B‡	982G	10.7	53.0	71.8	57.5	45.8	69.4	49.7
● ConvNeXt-B‡	964G	11.5	**54.0**	73.1	58.8	**46.9**	70.6	51.3
○ Swin-L‡	1382G	9.2	53.9	72.4	58.8	46.7	70.1	50.8
● ConvNeXt-L‡	1354G	10.0	**54.8**	73.8	59.8	**47.6**	71.3	51.7
● ConvNeXt-XL‡	1898G	8.6	**55.2**	74.2	59.9	**47.7**	71.6	52.2

Backbone Network

❖ ConvNext: A ConvNet for the 2020s

- 결론
  - ✓ 아래 주요 최신화 방법을 적용하여 Swin Transformer보다 높은 성능을 달성
    - ① Macro Design
    - ② ResNeXt-ify
    - ③ Inverted Bottleneck
    - ④ Large Kernel Sizes
    - ⑤ Micro Design
  - ✓ CNN의 Inductive bias를 유지할 수 있음

Conclusion

Conclusion

❖ Conclusion

- Backbone Network란 비전에서 딥러닝 구조의 Feature Extractor 부분을 말하며, 궁극적으로 학습된 모델의 예측 정확도를 높이고 계산량을 낮추기 위해 중요
- Swin Transformer는 Local window와 Patch Merging을 통해 ViT의 단점을 개선하여 Transformer 계열임에도 불구하고 좋은 성능과 학습속도를 보임
- ConvNext는 CNN을 기반으로 Inductive bias를 유지하면서 Swin Transformer에서 착안한 최신화 기법들을 통해 좋은 성능과 학습속도를 보임
- Backbone Network의 발전은 효과적인 구조를 통해 비전 분야에서 일관된 성능 향상을 보이기에 더 개선된 모델 등장 시 다양하게 적용할 수 있을 것으로 사료 됨

Conclusion

❖ 개인 의견

- 개인 연구인 Multichannel Signal 데이터에 ResNet, Swin Transformer, ConvNext를 적용 사례가 있음
  - ✓ ResNet 〉 Swin 〉 ConvNext 의 성능 순위가 나왔으며, ResNet와 Swin는 유사한 성능을 보임
  - ✓ 비전 Backbone Network가 타 분야에서도 좋은 성능을 보이지만, 동일한 성능 순위를 가지지는 않음
  - ✓ 도메인 별로 Backbone Network의 특징을 파악하여 활용하는 것이 중요할 것으로 사료 됨
    - ➢ ex. ResNet은 Swin이나 ConvNext보다 지역적인 특징에 더 집중 됨

감사합니다

References

❖ References

[1] Elharrouss, O., Akbari, Y., Almaadeed, N., & Al-Maadeed, S. (2022). Backbones-Review: Feature Extraction Networks for Deep Learning and Deep Reinforcement Learning Approaches. ArXiv preprint ArXiv: 2206.08016.

[2] Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., & Guo, B. (2021). Swin transformer: Hierarchical vision transformer using shifted window. In Proceedings of the IEEE/CVF International Conference on Computer Vision, 10012-10022.

[3] Liu, Z., Mao, H., Wu, C. Y., Feichtenhofer, C., Darrell, T., & Xie, S. (2022). A ConvNet for the 2020s. ArXiv preprint ArXiv: 2201.03545