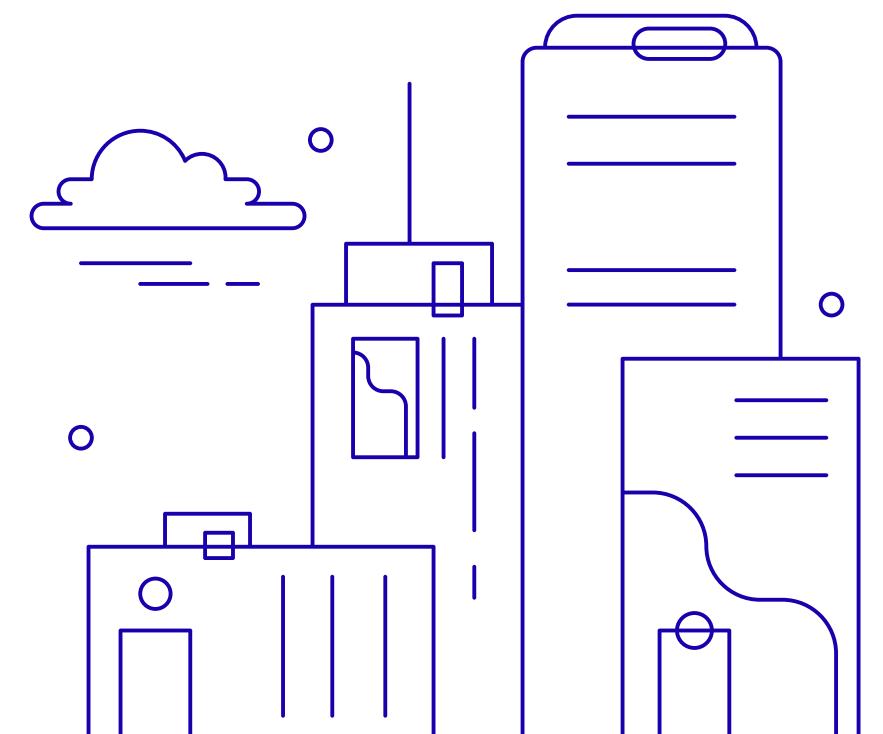


VIT - SWIN - SWIN V2

데이터 효율성과 확장성으로 진화한 비전 트랜스포머

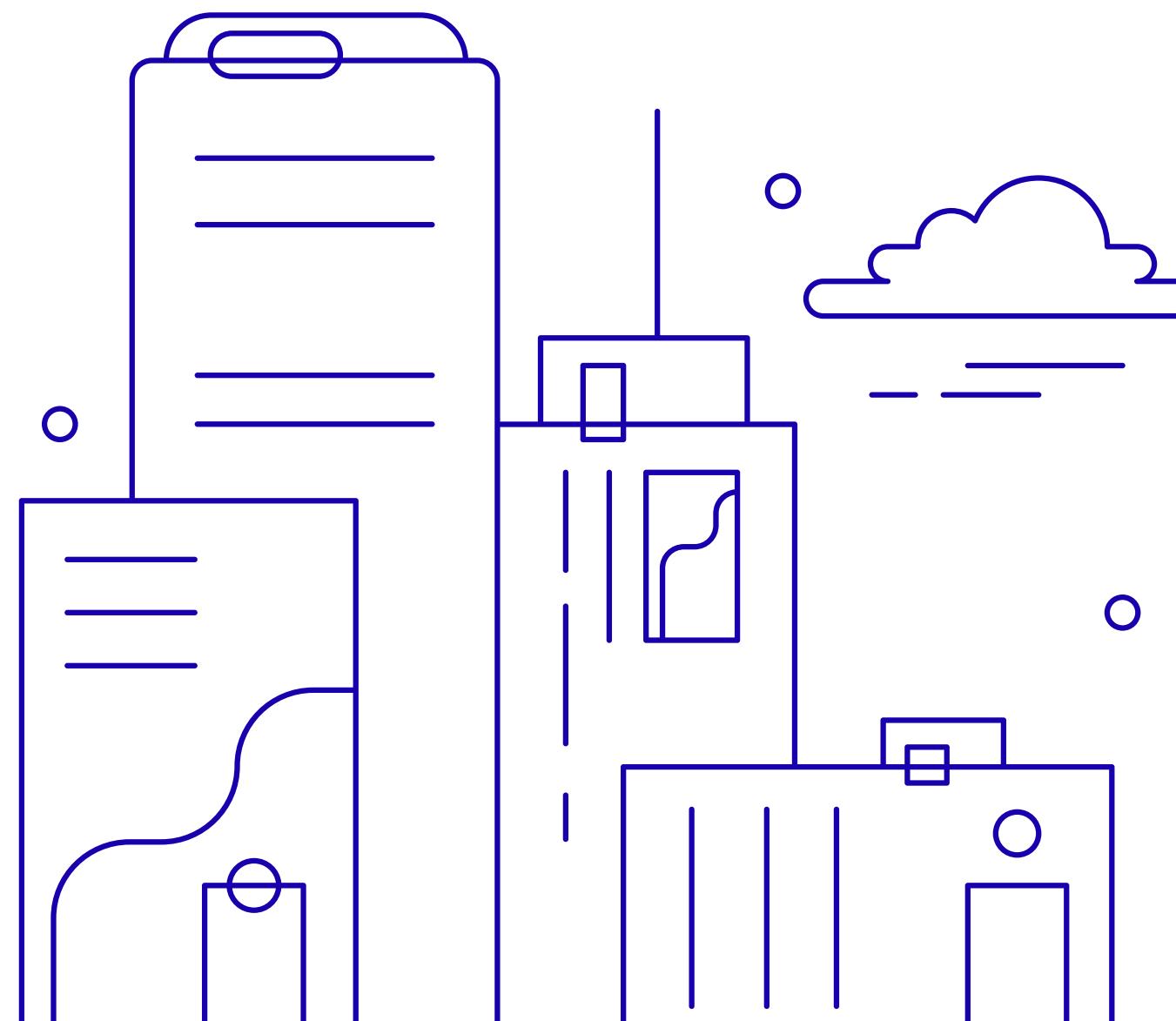


팀: 레모네이드

|

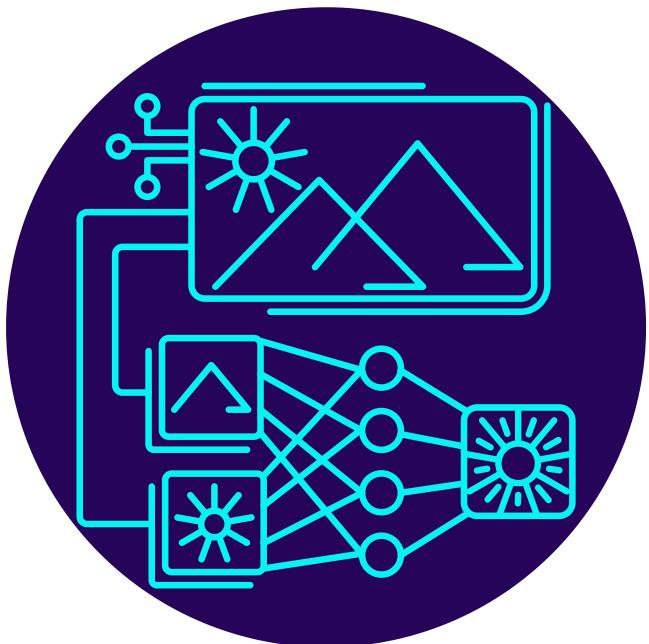
장우진 김선준 장재우 강희준 박은수

목차



- 01** 개요
- 02** Vision Transformer
- 03** Swin Transformer
- 04** Swin Transformer v2
- 05** 요약
- 06** 코드구현
- 07** 참조자료

01 개요 (1/2)



CNN(합성곱 신경망, Convolutional Neural Network)

이미지처럼 격자(그리드) 구조의 데이터를 대상으로,
작은 필터(커널)로 지역적 특징을 합성곱해
계층적으로 추출·압축하며 분류·검출·인식을 수행하는 딥러닝 모델이다.



오랫동안 컴퓨터 비전 분야는 합성곱 신경망 = CNN의 시대

01 개요 (2/2)

귀납적 편향(Inductive Biases)

딥러닝 알고리즘의 일반화를 돋기 위해
추가적으로 가정하는 개념

지역성(Locality)

이미지에서 중요한 정보(특징)는 가까이
있는 픽셀들끼리 뭉쳐서 만들어진다는 가정

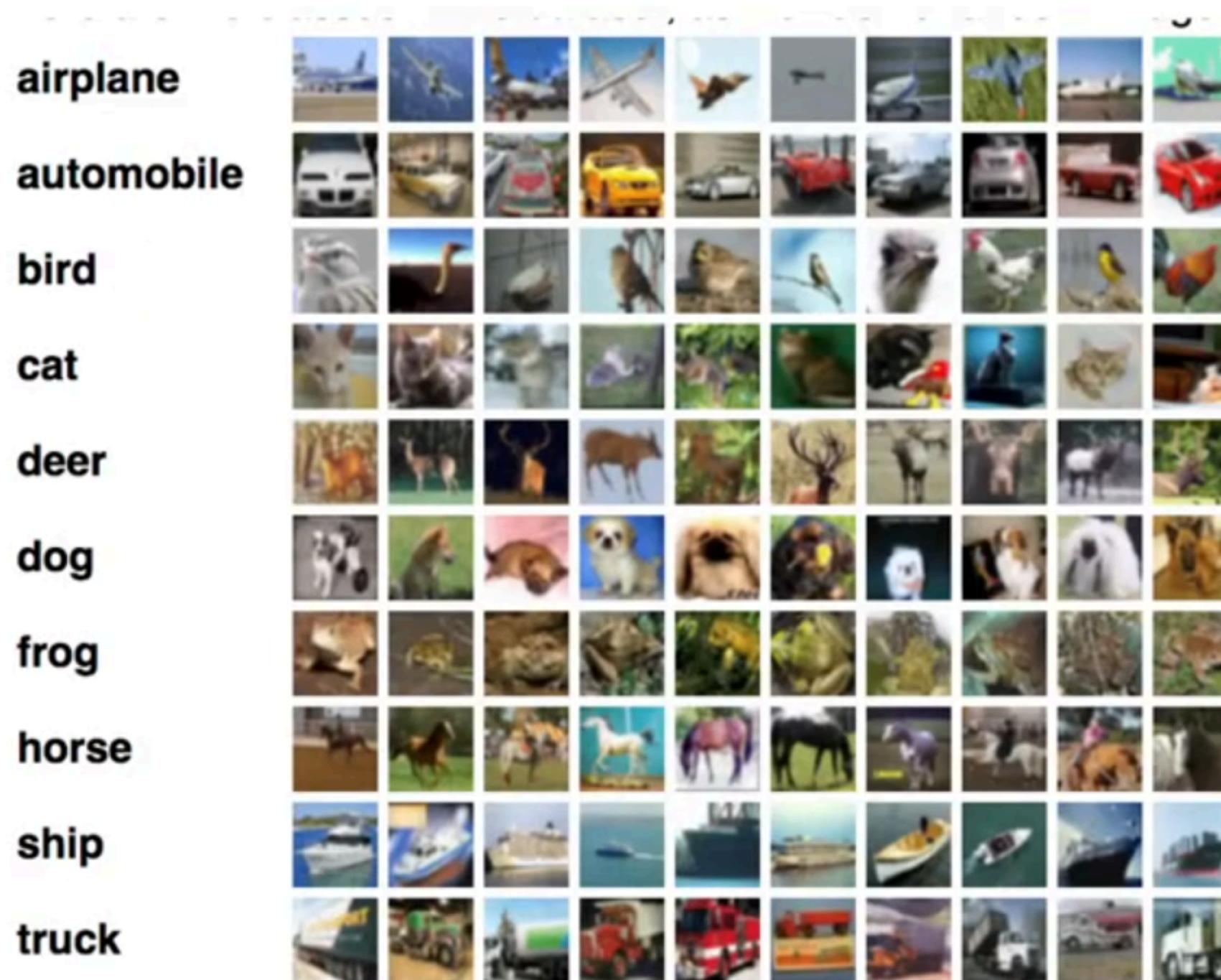
평행 이동 등변성(Translation Equivariance)

이미지 속의 물체가 움직여도 (평행 이동해도),
그 물체를 감지하는 방식(필터)은 변하지 않는다는 가정

➤ CNN의 귀납적 편향의 역할 = 자전거 보조 바퀴 달아주는 것

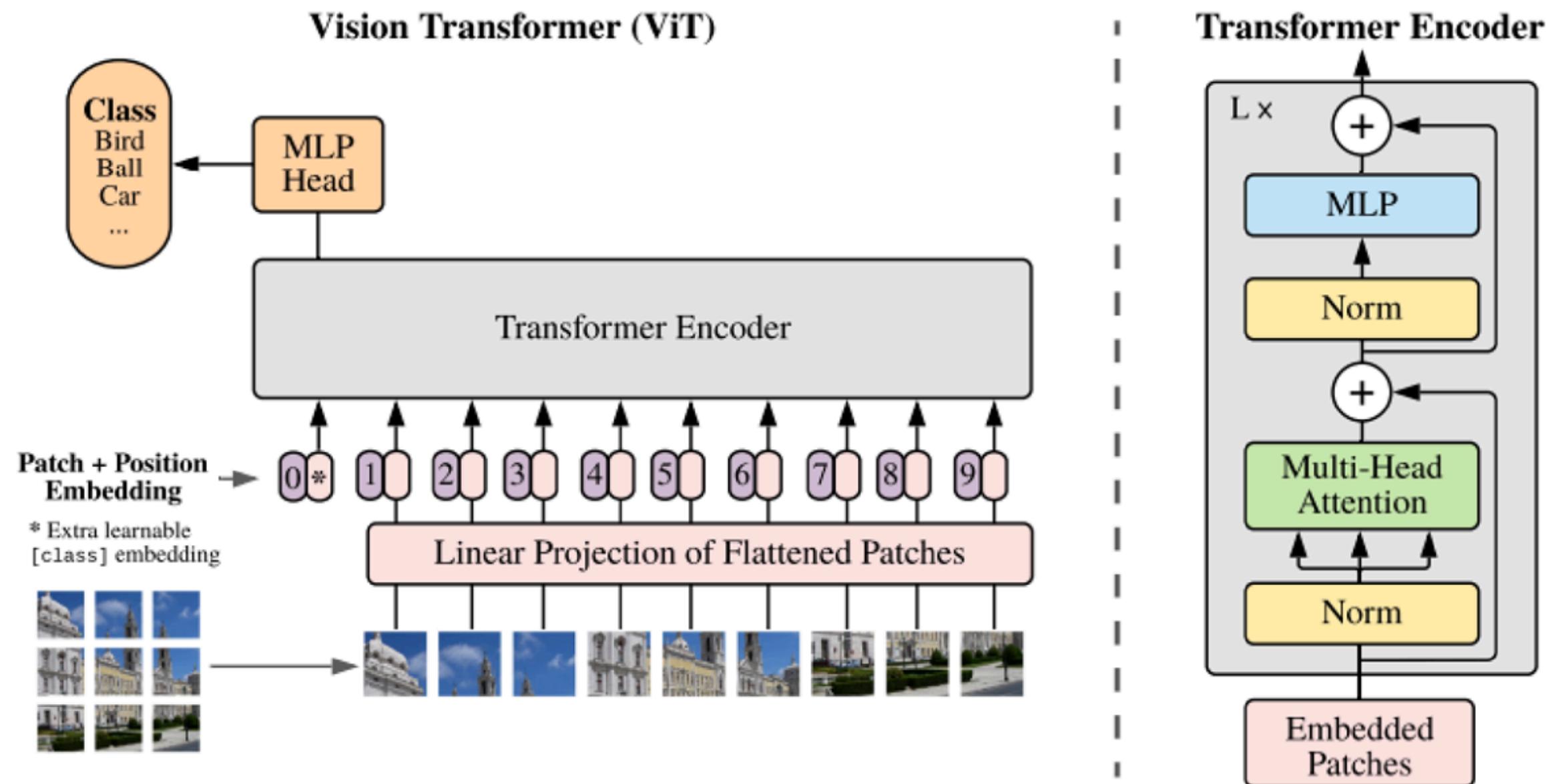
02 Vision Transformer, ViT (1/9)

▶ Image Classification 이미지 분류



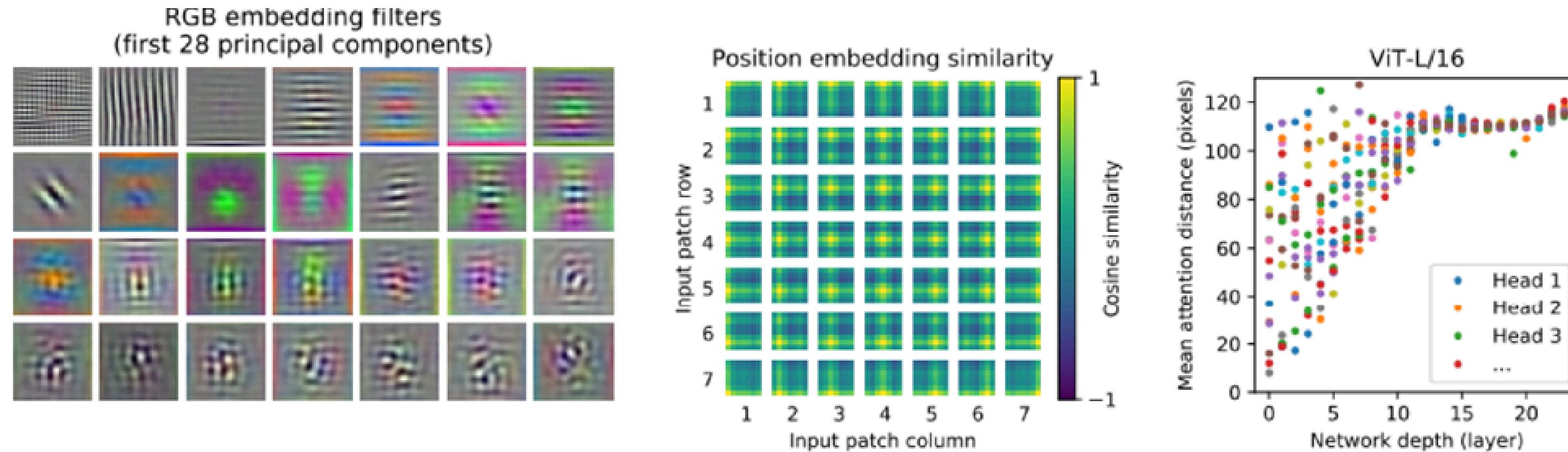
02 Vision Transformer, ViT (2/9)

AN IMAGE IS WORTH 16x16 WORDS:
TRANSFORMERS FOR IMAGE RECOGNITION AT SCALE



02 Vision Transformer, ViT (3/9)

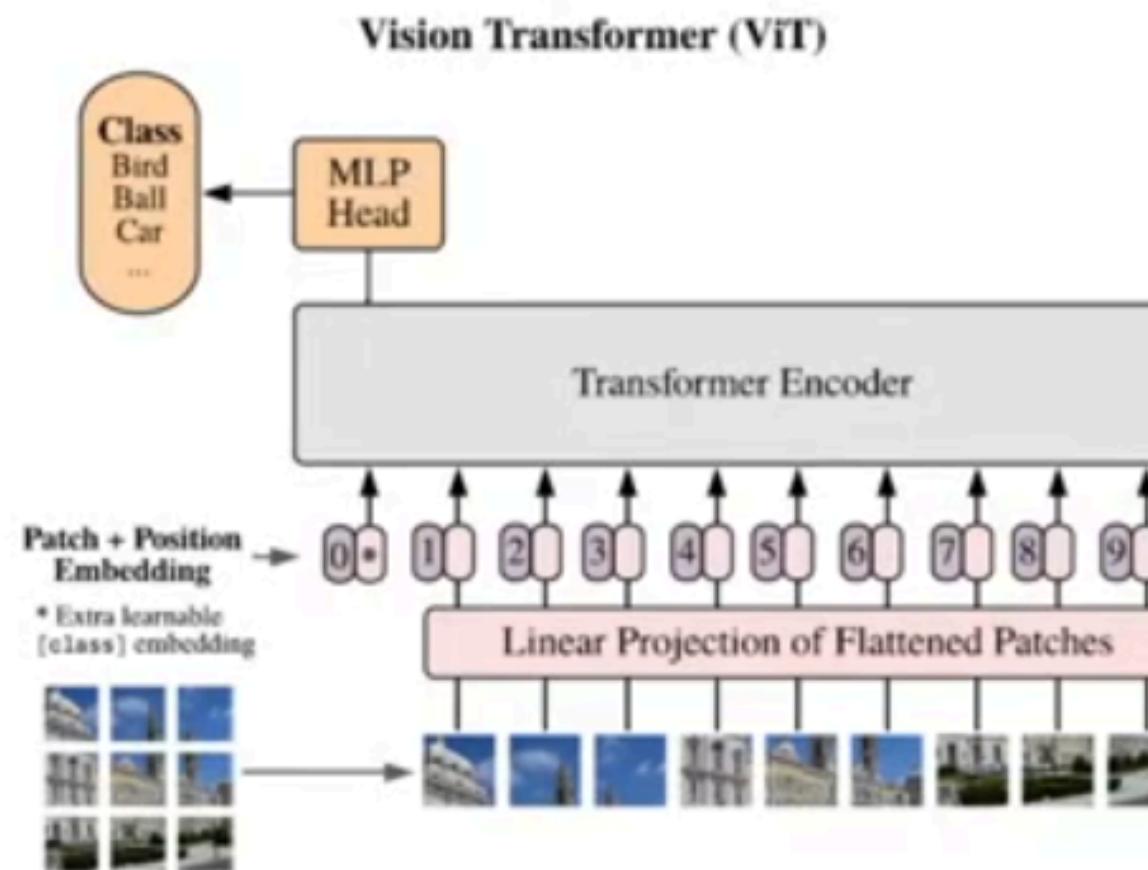
Position Embedding Similarity



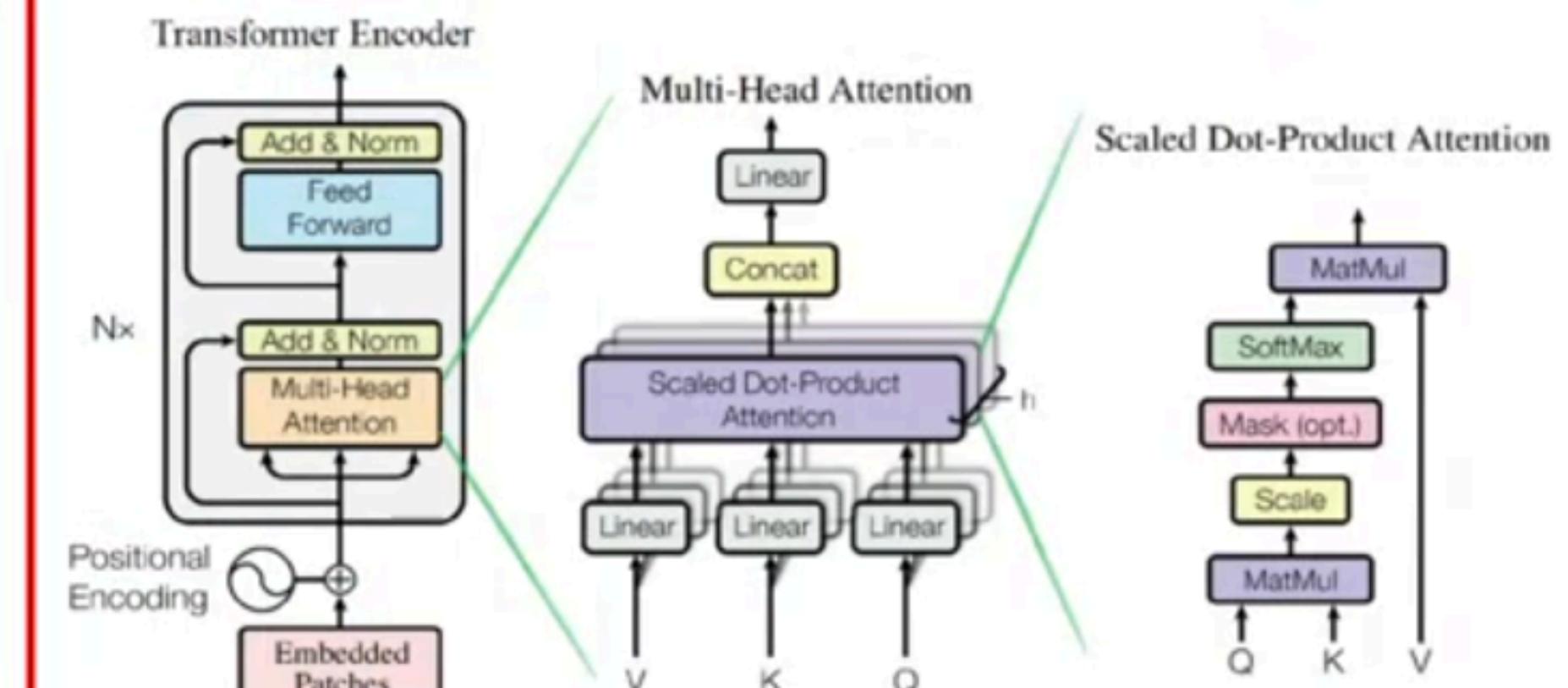
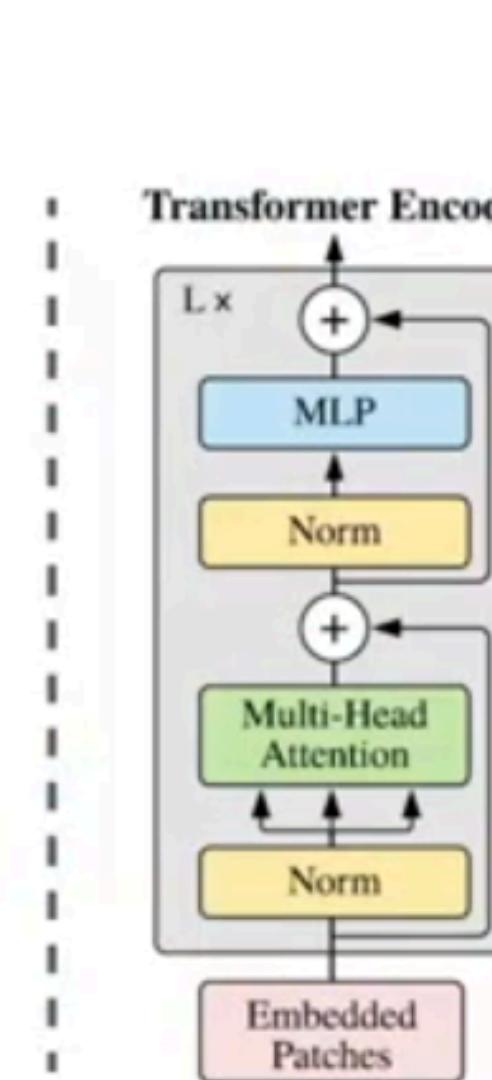
- › CNN과 유사하게 색상 필터를 학습
- › CNN과 유사하게 공간적 정보를 학습
- › 헤드별 다양성 : 국소적 → 전역적 계층형성과 동시에 얇은 레이어에서도 전역적정보 학습

02 Vision Transformer, ViT (4/9)

Network

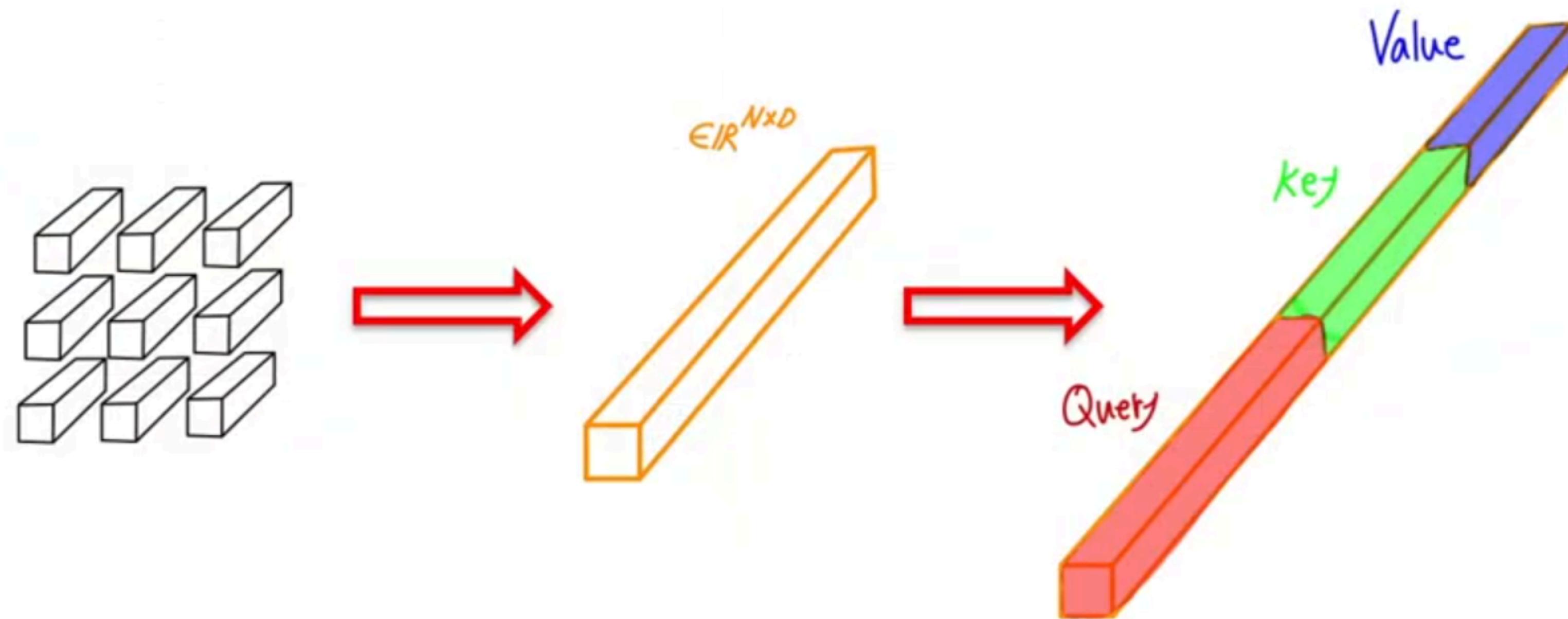


Vision Transformer

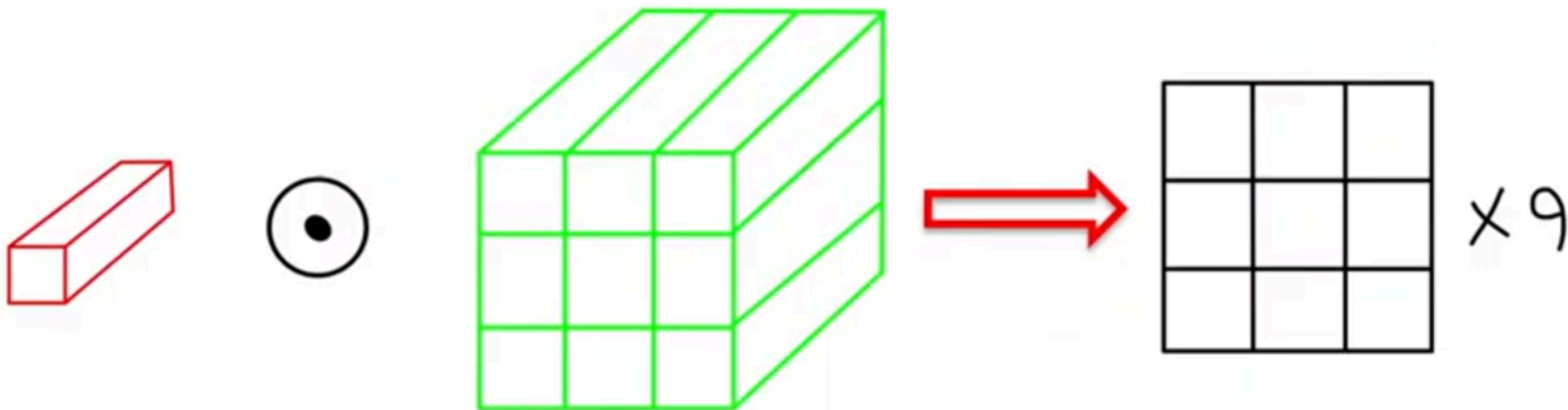


NLP Transformer

02 Vision Transformer, ViT (5/9)



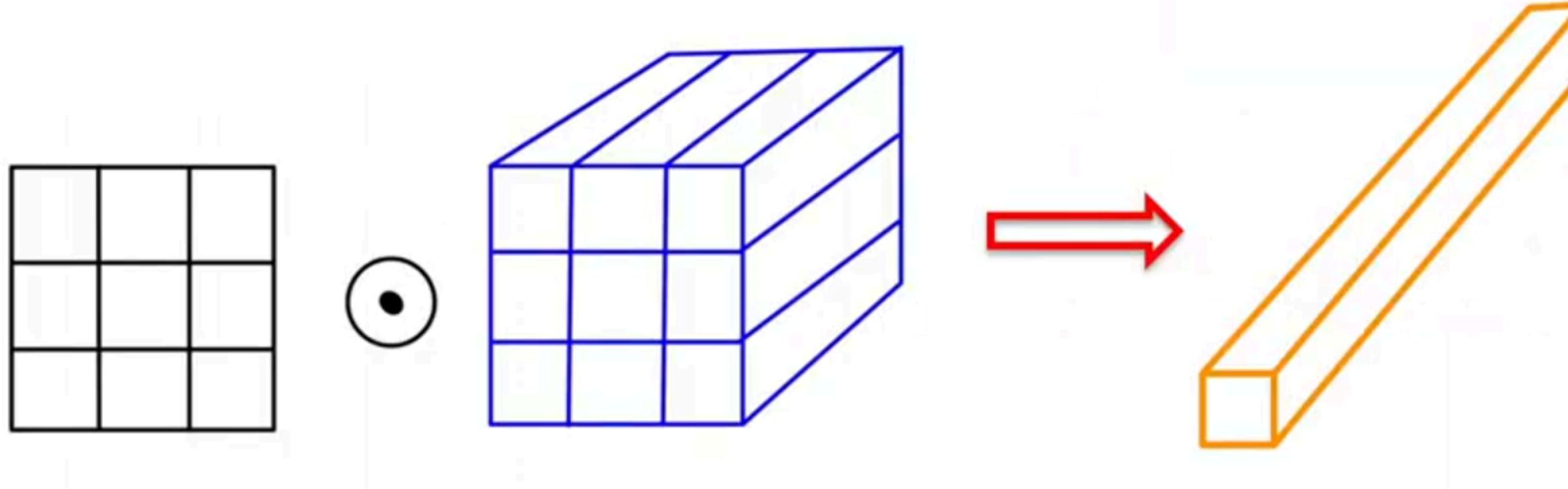
02 Vision Transformer, ViT (6/9)



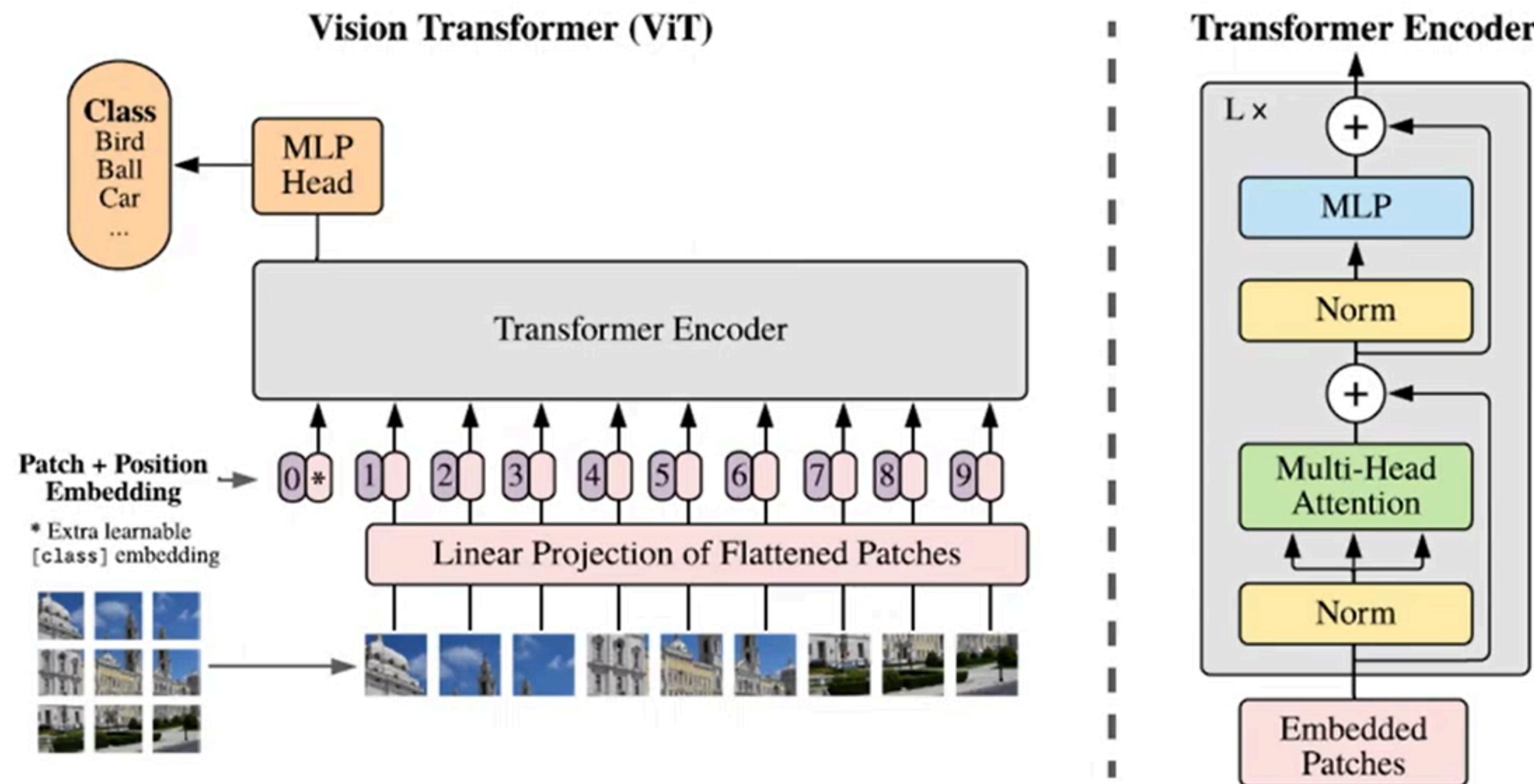
02 Vision Transformer, ViT (7/9)

Attention

$$\text{SoftMax} \left(\begin{array}{|c|c|c|} \hline & & \\ \hline & & \\ \hline & & \\ \hline \end{array} \times 9 \right) = \text{attention Score}$$

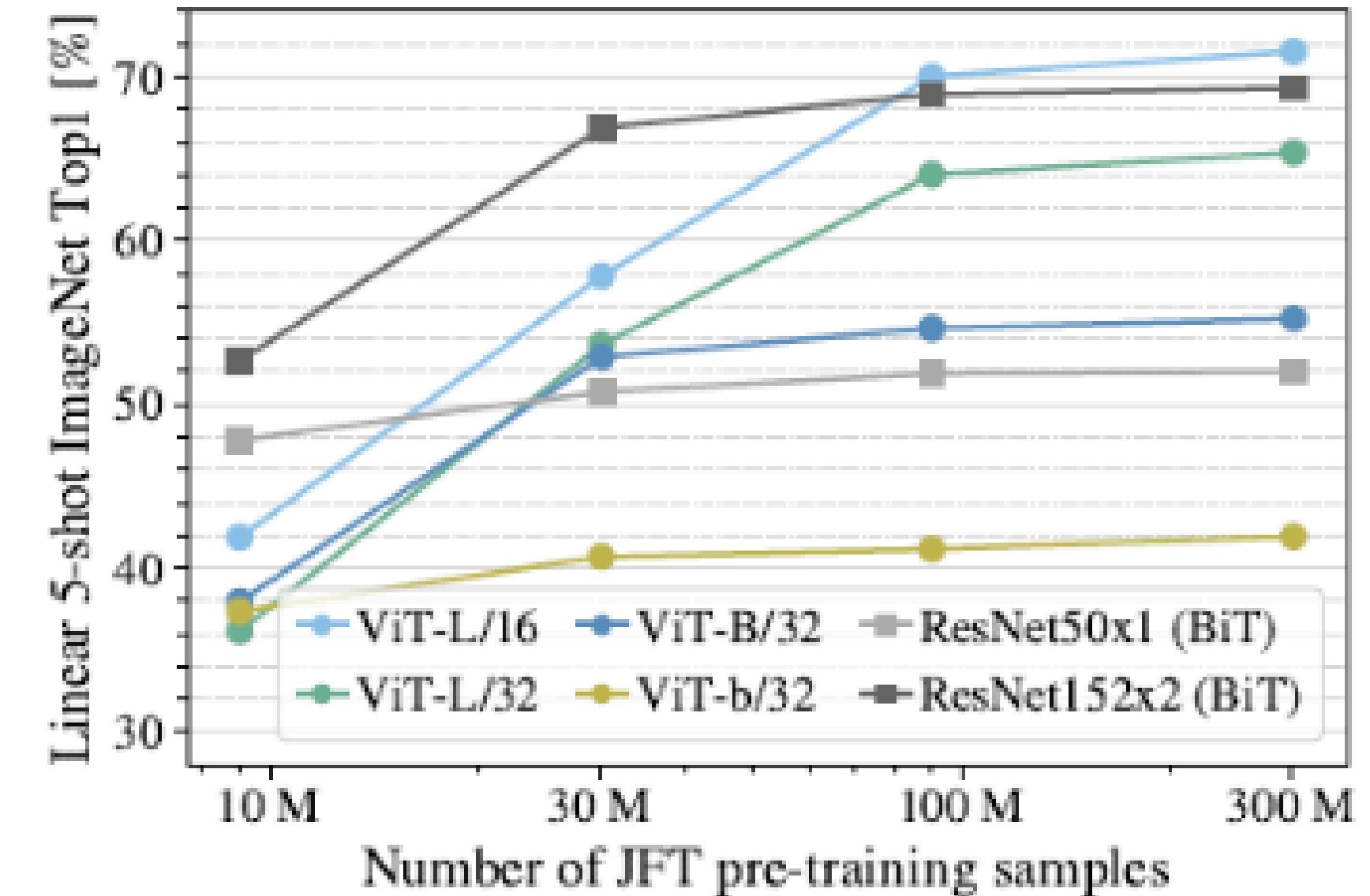
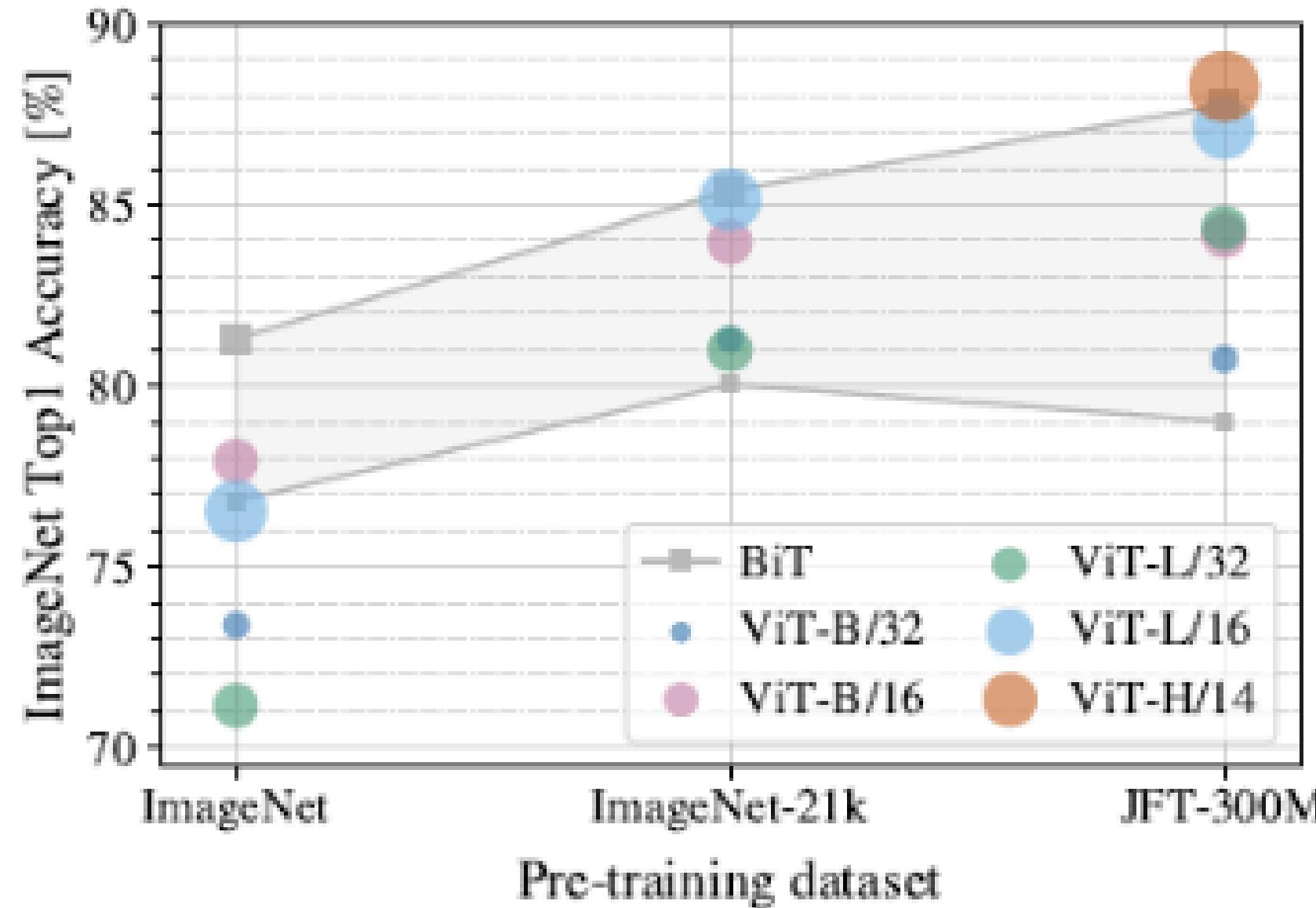


02 Vision Transformer, ViT (8/9)



02 Vision Transformer, ViT (9/9)

Pre-training data



03 Swin Transformer (1/7)

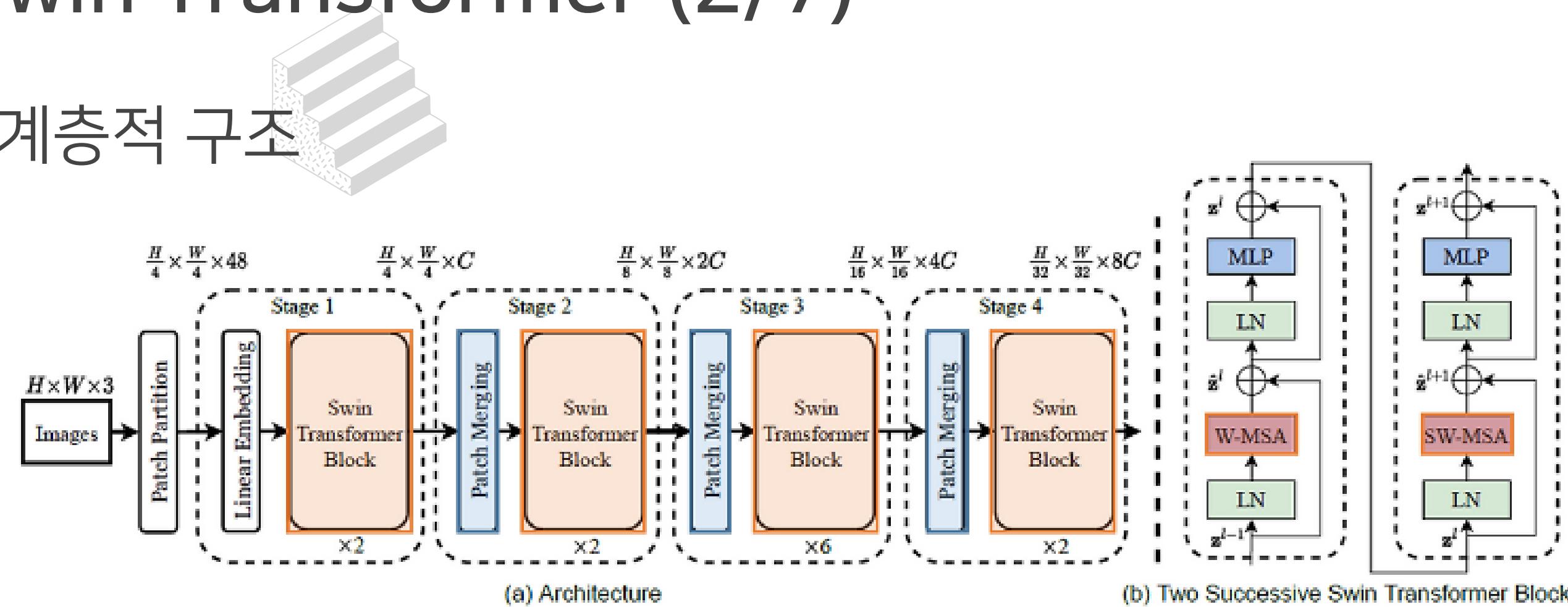
Hierarchical Vision Transformer using Shifted Windows

- › ViT의 단점인 연산량폭발($O(n^2)$ = 패치개수)을 개선하고자 함
- › CNN 모델의 특징인 공간적 계층성: “부분 특징 → 더 큰 의미적 특징”을 도입하고자 함
- › 계층적(Hierarchical): 해상도(Resolution)는 절반, 채널(Channel)은 두 배
- › Stage를 거치며 feature map 크기가 줄어드니, Attention의 계산량이 감소

Stage	Feature map 크기	Channel 수	의미
Stage 1	56×56	96	세밀한 지역 정보
Stage 2	28×28	192	중간 규모 패턴
Stage 3	14×14	384	객체 단위 표현
Stage 4	7×7	768	전역적 의미 표현

03 Swin Transformer (2/7)

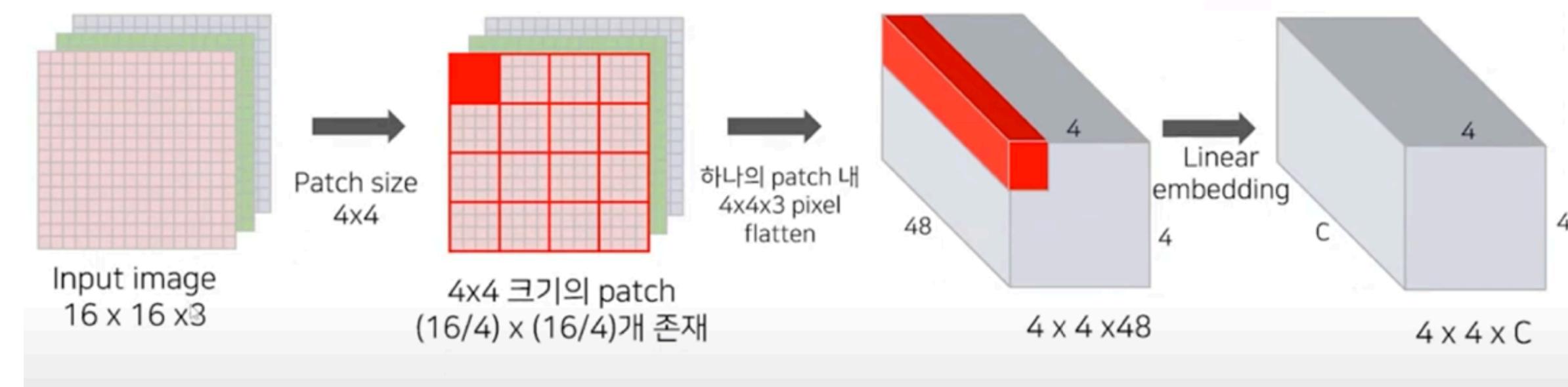
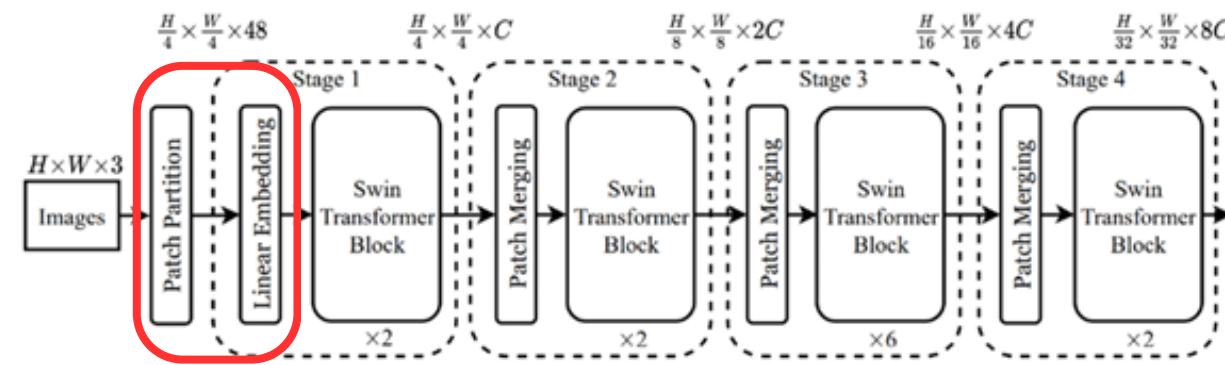
아키텍처: 계층적 구조



출처: Swin Transformer

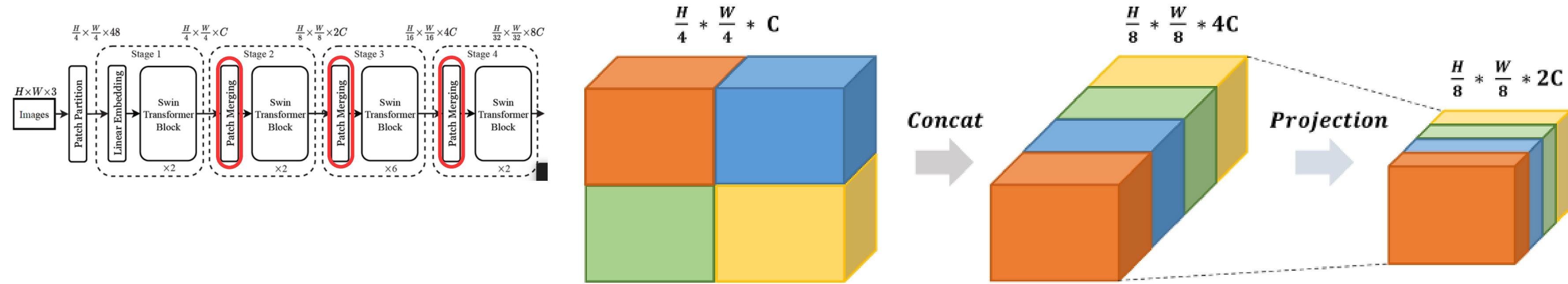
- Patch Partition , Linear Embedding, Swin Transformer Block, Patch Merging 4가지 모듈로 구성
- Stage 1 ~ 4 를 거치며 해상도가 작아지고, 채널 수는 늘어나면서 Hierarchical 한 구조 생성
- VGG 같이 일반적인 CNN backbone과 동일한 feature map resolution 가짐

03 Swin Transformer (3/7)



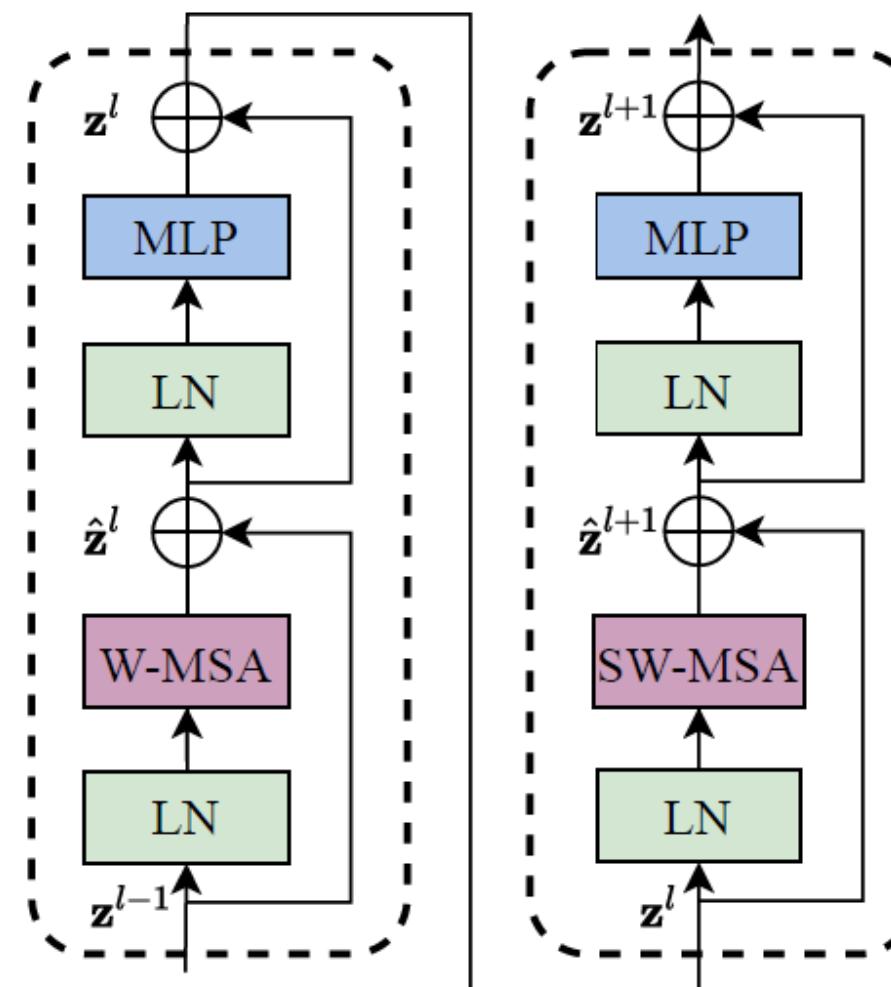
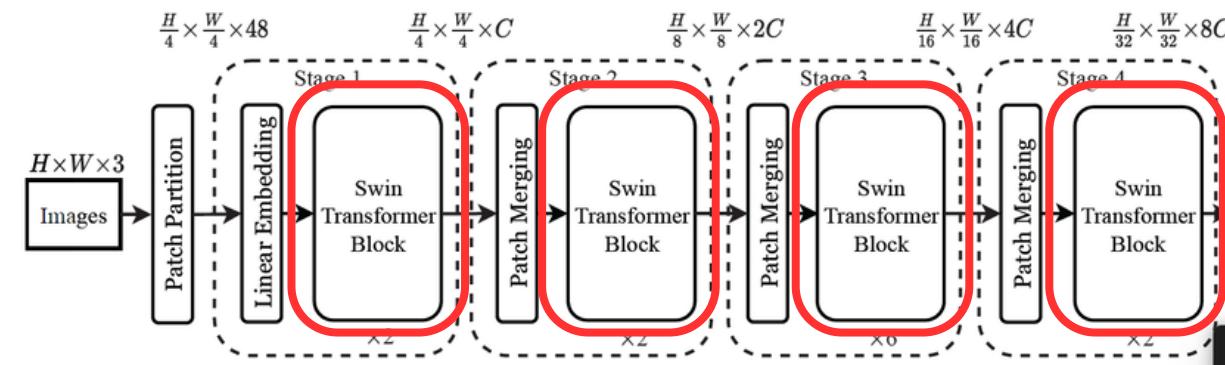
- ViT와 동일하게 patch 단위로 분할 후 flatten
- ViT와 달리 CLS(클래스 토큰)이 없음

03 Swin Transformer (4/7)



- Stage 1의 출력인 $\frac{H}{4} * \frac{W}{4} * C$ 의 차원을 $2 * 2$ 그룹들로 나눕니다.
- 나눠진 하나의 그룹은 $\frac{H}{8} * \frac{W}{8} * C$ 의 차원을 가지고, 4개의 그룹들을 채널을 기준으로 병합합니다(Concat).
- 병합된 $\frac{H}{8} * \frac{W}{8} * 4C$ 의 차원 축소를 위해 절반인 $2C$ 의 차원으로 축소합니다.
- 위 과정들은 모든 Stage에서 동일하게 작용합니다.

03 Swin Transformer (5/7)



- ViT와 동일하게 트랜스포머 인코더 사용
- ViT와 동일하게 Layer Norm이 MSA, MLP 앞에 위치함
- ViT와 달리 W-MSA 구역 \rightarrow SW-MSA 구역 2블록이 세트로 구성 (x2)
 - W-MSA : Window-based Multihead Self Attention
 - SW-MSA : Shifted-window Multihead Self Attention

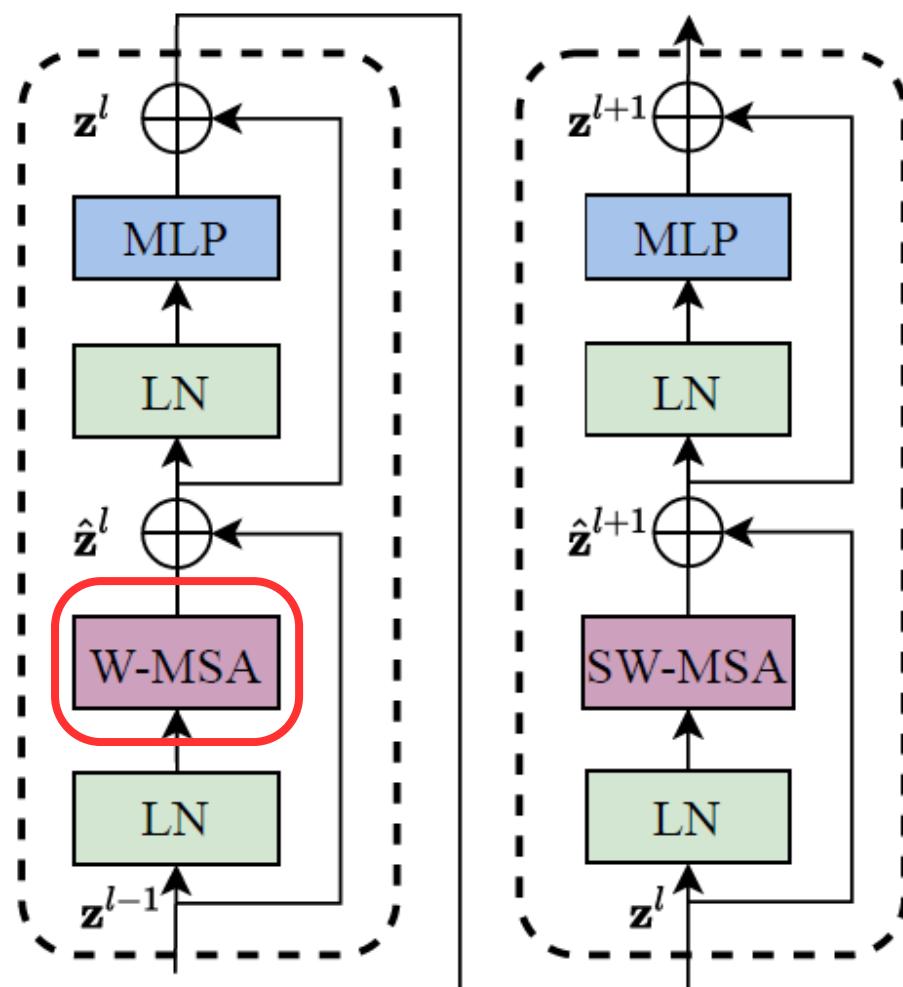
$$\hat{\mathbf{z}}^l = \text{W-MSA}(\text{LN}(\mathbf{z}^{l-1})) + \mathbf{z}^{l-1},$$

$$\mathbf{z}^l = \text{MLP}(\text{LN}(\hat{\mathbf{z}}^l)) + \hat{\mathbf{z}}^l,$$

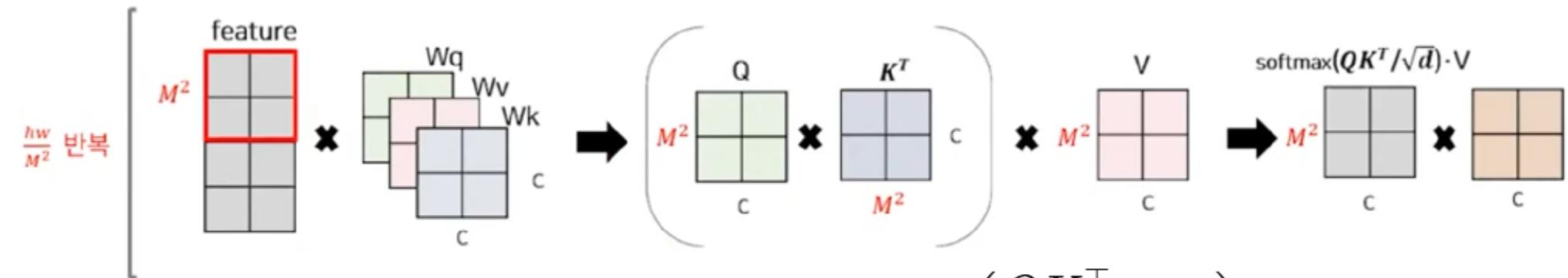
$$\hat{\mathbf{z}}^{l+1} = \text{SW-MSA}(\text{LN}(\mathbf{z}^l)) + \mathbf{z}^l,$$

$$\mathbf{z}^{l+1} = \text{MLP}(\text{LN}(\hat{\mathbf{z}}^{l+1})) + \hat{\mathbf{z}}^{l+1},$$

03 Swin Transformer (6/7)



➤ W-MSA



$$\text{W-MSA}(Q, K, V) = \text{Softmax} \left(\frac{QK^T}{\sqrt{d}} + B \right) V$$

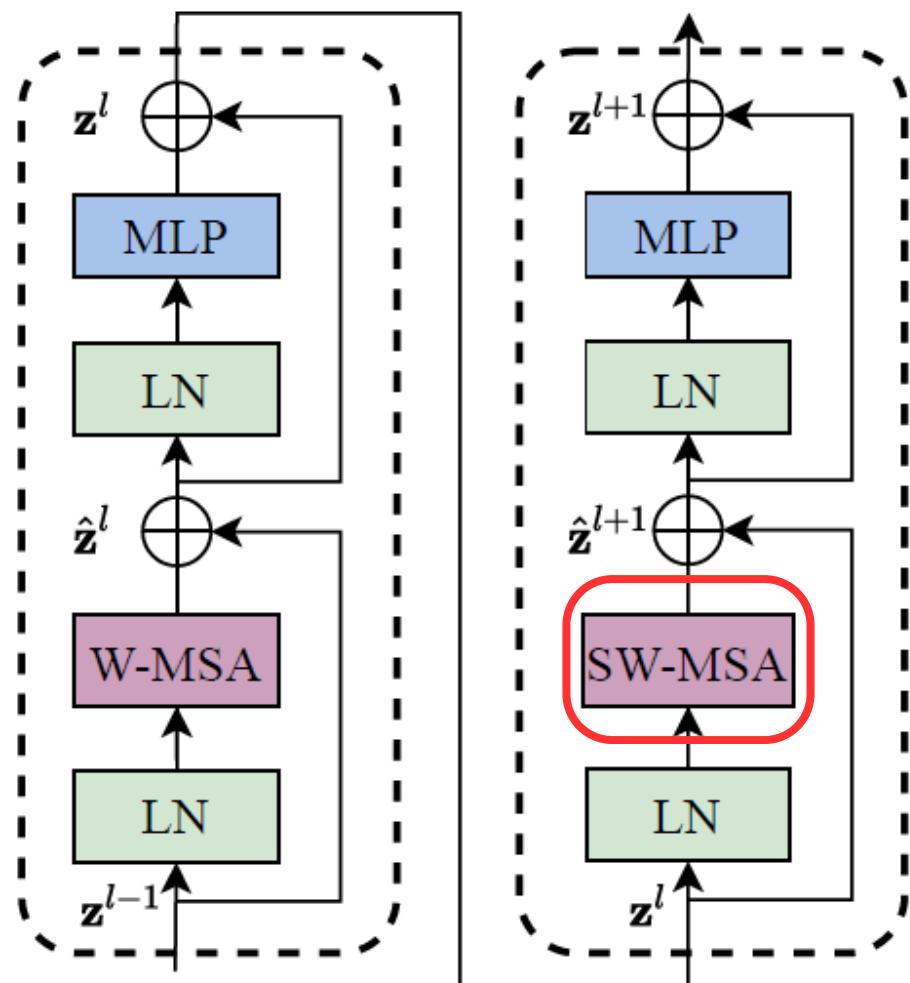
B : relative position bias (상대적 위치 인코딩)
 → Inductive Bias를 부여하기 위해 사용

➤ W-MSA가 attention 연산량이 적다

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

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

03 Swin Transformer (7/7)

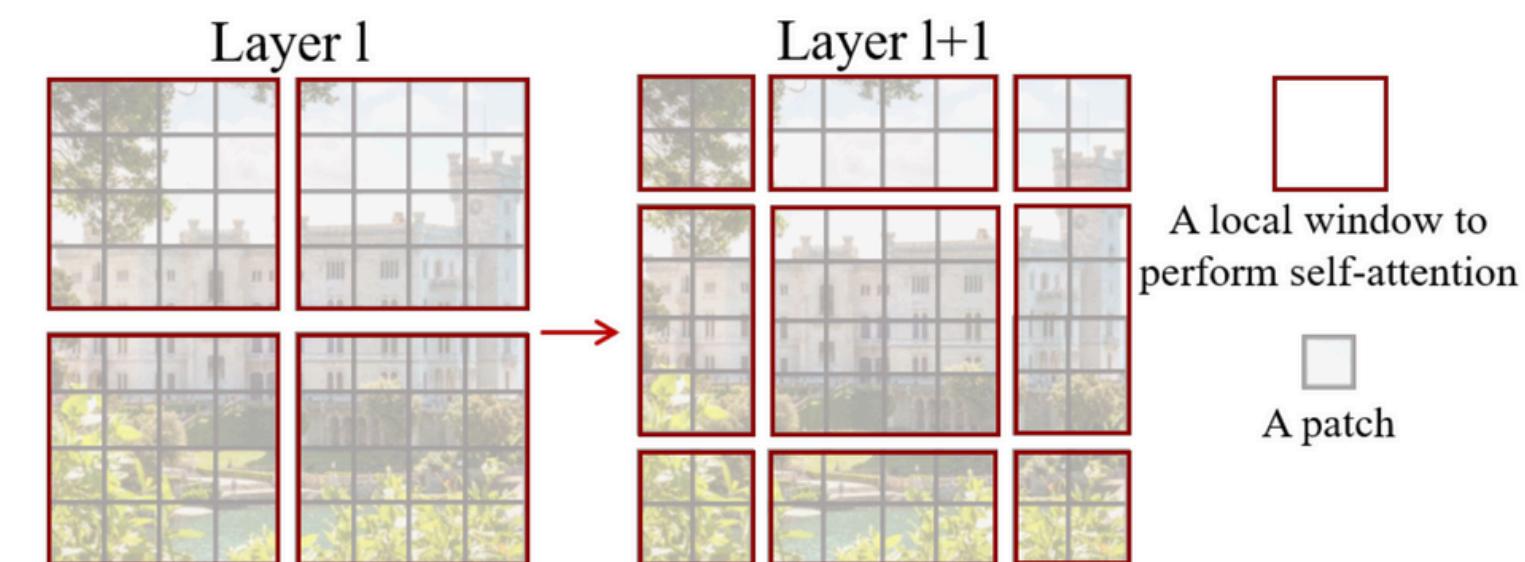
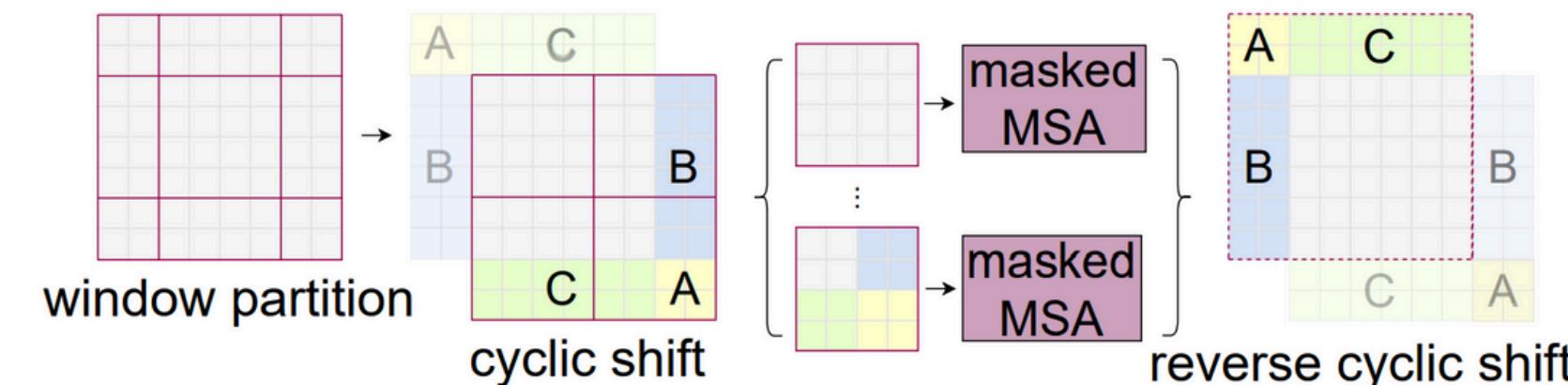


$$\text{SW-MSA}(Q, K, V) = \text{Softmax} \left(\frac{QK^\top}{\sqrt{d}} + B + M \right) V$$

B: relative position bias (상대적 위치 인코딩)

M: mask

➤ SW-MSA



04 Swin Transformer V2 (1/5)

Scaling Up Capacity and Resolution

Swin에 사용된 Transformer는 모델의 용량이 커지고, 해상도가 올라감에 따라 성능 향상에 한계가 있다고 지적

3가지 문제점

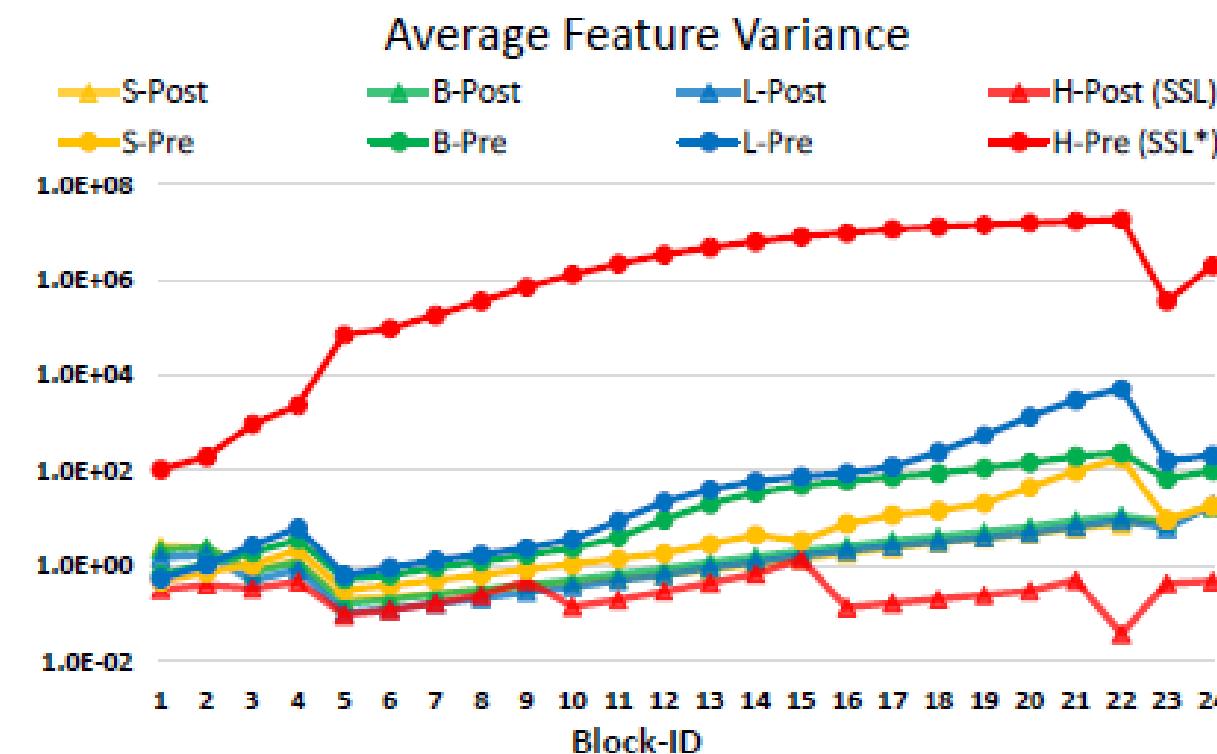
학습의 불안정성

Pre-training과 Fine-tuning의 해상도 차이

Labeled data 부족

04 Swin Transformer V2 (2/5)

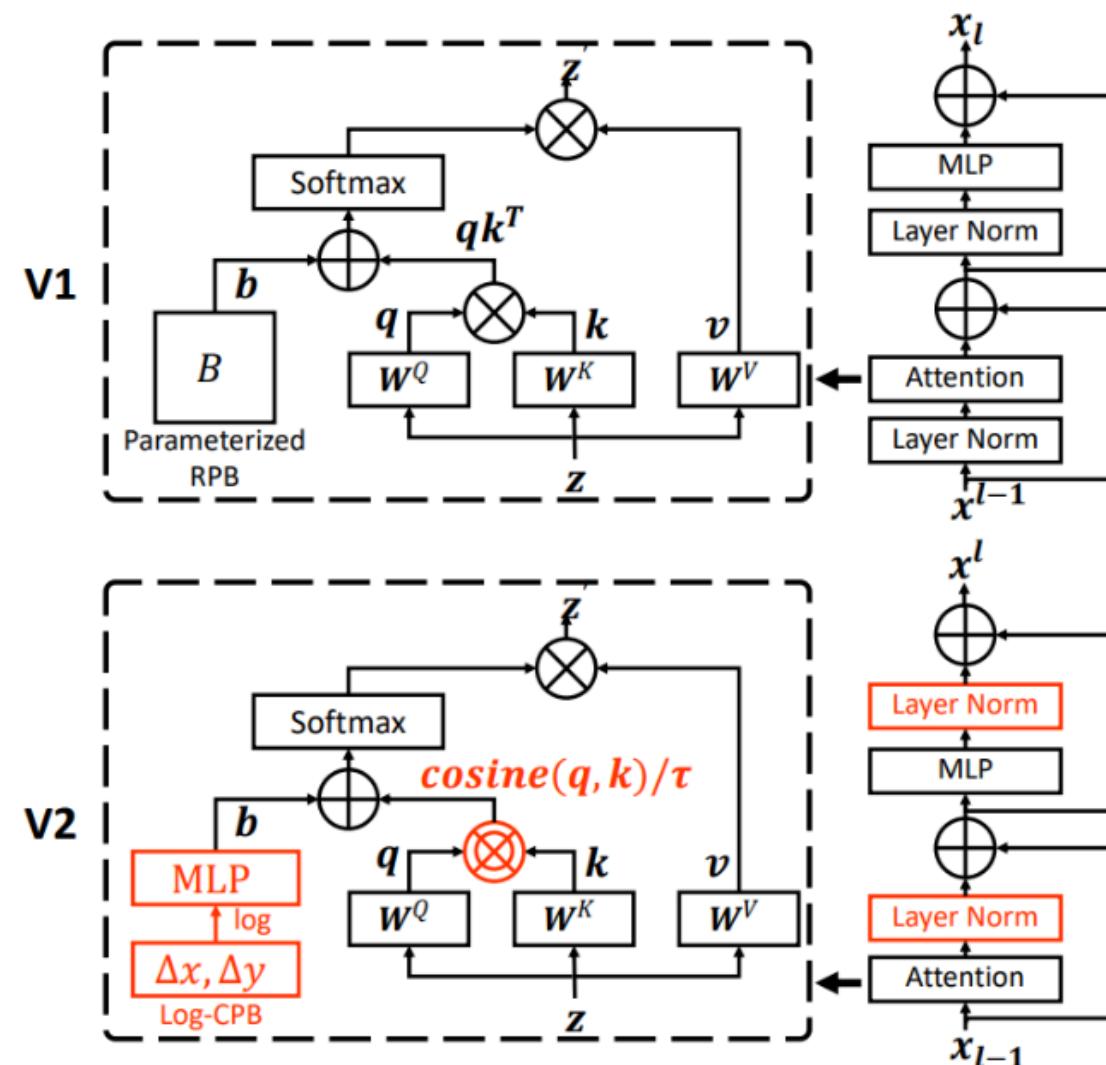
학습의 불안정성



- ▶ 레이어가 깊어질수록 값이 점진적으로 커지는 것을 볼 수 있고,
이는 해상도가 더 커질수록 심해짐

04 Swin Transformer V2 (3/5)

학습의 불안정성



➤ LayerNorm(LN)이 위치해 있는 Pre-Norm → Post-Norm으로 변경
Attention 및 MLP 연산 후 LN이 이루어지기 때문에 Activation Value가 점진적으로 커지는 것을 막음

$$\text{Sim}(\mathbf{q}_i, \mathbf{k}_j) = \cos(\mathbf{q}_i, \mathbf{k}_j) / \tau + B_{ij},$$

B_{ij} : Positional Encoding

➤ Scaled Cos Attention
Self-Attention은 Query랑 Key 벡터를 내적 Dot-Product 유사도를 코사인 유사도로 바꿈

04 Swin Transformer V2 (4/5)

사전학습과 미세조정의 해상도 차이

$$B(\Delta x, \Delta y) = G(\Delta x, \Delta y),$$

- 로그 공간 연속 위치 편향 → Log CPB
log-spaced Continuous Position Bias
로그 간격의 연속적 위치 인코딩을 어텐션 바이어스로 더하는 방식
- 상대좌표 = “두 패치의 관계”를 배우게 해 주므로
 - 이동·해상도 변화에 강함
 - 파라미터를 아낄 수 있음
 - 윈도우/시프트 구조와 잘 맞음
 - 로그 변환(로그-CPB)과 결합하면 대해상도 전이까지 부드러움

04 Swin Transformer V2 (5/5)

Labeled data 부족

	method	ImageNet*	ImageNet [†]				COCO		ADE20k		
		W8, I256 top-1 acc	W12, I384 top-1 acc	W16, I512 top-1 acc	W20, I640 top-1 acc	W24, I768 top-1 acc	W16 AP ^{box}	W32 AP ^{box}	W16 mIoU	W20 mIoU	W32 mIoU
Swin V1	Parameterized position bias [46]	81.7	79.4/82.7	77.2/83.0	73.2/83.2	68.7/83.2	50.8	50.9	45.5	45.8	44.5
Swin V2	Linear-Spaced CPB	81.7 (+0.0)	82.0/82.9 (+2.6/+0.2)	81.2/83.3 (+4.0/+0.3)	79.8/83.6 (+6.6/+0.4)	77.6/83.7 (+8.9/+0.5)	50.9 (+0.1)	51.7 (+0.8)	47.0 (+1.5)	47.4 (+1.6)	47.2 (+2.7)
Swin V2	Log-Spaced CPB	81.8 (+0.1)	82.4/83.2 (+3.0/+0.5)	81.7/83.8 (+4.5/+0.8)	80.4/84.0 (+7.2/+0.8)	79.1/84.2 (+10.4/+1.0)	51.1 (+0.3)	51.8 (+0.9)	47.0 (+1.5)	47.7 (+1.9)	47.8 (+3.3)

➤ SimMIM 기법 = "A Simple Framework for Masked Image Modeling"

간단한 마스킹 이미지 모델링 프레임워크

- 이미지 일부를 가린다 → 패치를 무작위(주로 블록 단위)로 많이 가림(보통 높은 마스크 비율 ~60–80%).
- 가려진 부분을 맞힌다 → 백본(예: Swin/ViT) 위에 **아주 얇은 선형 디코더(Linear head)**만 붙여 가려진 패치의 원래 픽셀(또는 정규화된 픽셀)을 복원하도록 학습.
- 손실은 가려진 곳만 계산 → L1(혹은 L2) 재구성 손실을 마스킹된 패치에만 적용 → 단순·안정.

➤ 정답 label 없이 자기지도학습, 이미지 자체만 가지고 학습

기존 labeled data의 1/40 분량만을 사용하여 SOTA 성능 기록

05 요약 (1/4)

CLS Token - ViT

Why?

가변 길이를 고정 길이로

- 입력 해상도/패치 수 N이 달라도, [CLS] 한 토큰의 최종 은닉벡터만 사용 → 항상 1개 벡터 출력.
- 결과: 분류 헤드 설계가 항상 동일(입력 크기와 무관) → 전이학습·배포 용이.

규칙 기반 요약이 아니라 '학습 가능한 요약'

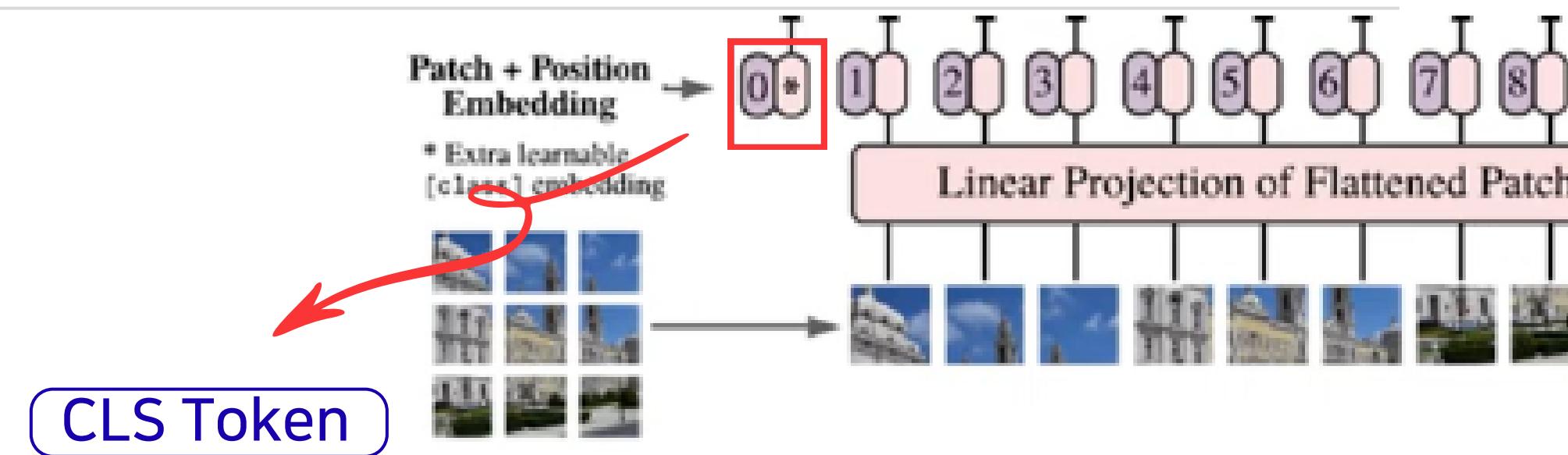
- 평균풀링: 모든 패치에 동일 가중 더함 → 중요도 반영 불가.
- [CLS]는 어떤 패치가 중요한지 가중치를 학습해 작은 물체/희귀 패턴/복잡한 배경에서 유리.

그라디언트 허브(학습 안정성)

- 분류 손실이 직접 은닉벡터에 걸림 → 그라디언트가 [CLS]를 통해 전 레이어로 명확하게 역전파.
- 깊은 모델에서도 신호 소실/분산을 줄여 수렴 안정성에 도움.

엔지니어링 단순성

- 출력은 항상 [CLS] 1개 → 헤드 교체(Linear/MLP)만으로 다중 태스크 전환이 쉬움.
- BERT와 동일한 인터페이스 → 코드/가중치·레시피 재사용이 용이.



입력 시퀀스 맨 앞에 붙이는 학습 가능한 1개의 벡터

self-attention 동안 중요 패치 정보만 모아 최종 전역 요약 벡터로 사용

05 요약 (2/4) LN과 BN의 핵심 차이

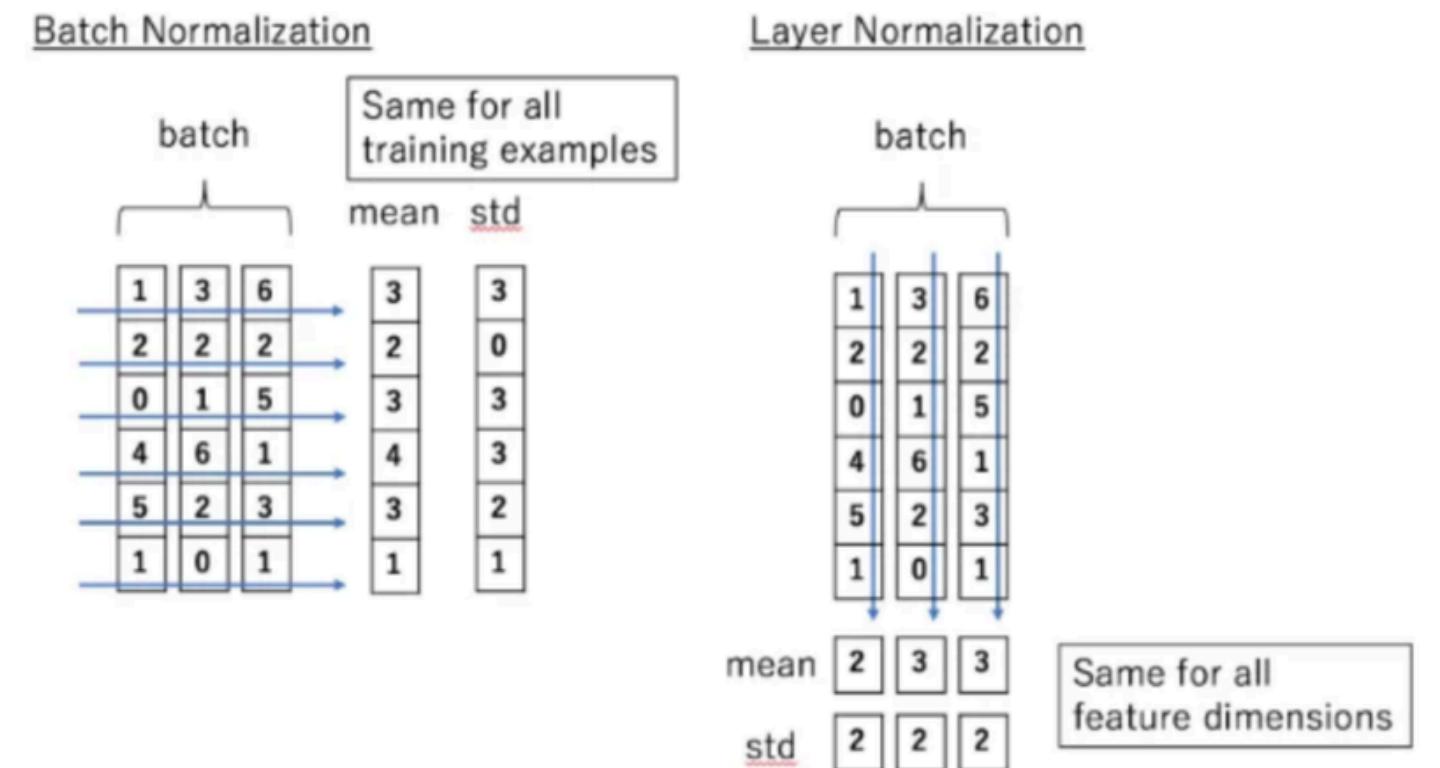
모두 신경망 내부 활성화 값의 분포를 안정화하기 위해 등장한 정규화 개념이지만 정규화의 차원과 적용 대상이 서로 다름

Layer Normalization (LN)

- 한 샘플(토큰) 내부의 피처 축에서 평균·분산을 계산. 배치와 무관, 학습·추론 모두 같은 방식.

장점: 작은 배치/시퀀스/토큰 모델에 안정, 분산 학습에서도 일관.

단점: CNN에서 BN만큼의 규제효과/속도 이득은 덜함.



Batch Normalization (BN)

- 채널마다 배치 차원(CNN이면 공간 H×W까지)을 모아서 평균·분산을 잡고 정규화.

학습 시 배치 통계를 쓰고, 추론 시엔 러닝 평균/분산(고정 통계)을 사용. 배치 크기/분포에 민감.

장점: CNN에서 수렴 가속·일반화↑, 추론 시 빠름.

단점: 작은/가변 배치, 분포 변화, 시퀀스(RNN) 모델에 취약.

구분	LayerNorm (LN)	BatchNorm (BN)
정규화 기준 축	샘플 내부의 모든 feature	배치(+공간) 축의 채널별 통계
배치 크기 의존성	없음 (소배치·가변 배치에 안정)	큽 (소배치·분산학습에서 불안정)
러닝 평균/분산	없음 (학습=추론 동일)	있음 (추론 시 러닝 통계 사용)
가변 길이/마스크	강함 (시퀀스 길이 자유)	주의 필요
주 사용처	Transformer/ViT/RNN	CNN(큰 배치)
장단 요약	일관성·안정성↑, 구현 단순	수렴 가속·규제 효과↑ (하지만 배치 의존)

05 요약 (3/4)

ViT, Swin, Swin v2에서 LN을 사용하는 이유

Transformer 계열 모델은 입력의 길이와 배치 크기가 가변적이고, 시퀀스 기반의 Self-Attention 구조를 가짐

1. Batch 크기 불균형과 통계 불안정성

- Transformer는 다양한 길이의 토큰 시퀀스를 처리하므로, mini-batch의 크기나 분포가 일정하지 않음.
- BN은 batch 전체의 평균과 분산을 계산하기 때문에, 작은 batch나 불균일한 시퀀스에서는 통계값이 불안정해짐.
- 반면 LN은 각 Sample에 대해 독립적으로 작동하므로 이러한 영향을 받지 않음.

2. 분산 학습 시 통계 동기화 문제

- BN은 여러 GPU에서 batch 통계를 동기화(synchronize)해야 하므로 통신 비용이 큼.
- LN은 이런 연산이 필요하지 않아 분산학습에 효율적.

3. Transformer 구조의 안정성

- Feed-Forward와 Self-Attention 블록의 출력을 바로 정규화해야 하는데, BN은 각 시퀀스별 길이 변화나 attention mask 때문에 오작동하기 쉬움.
- LN은 feature 단위로만 정규화하므로 길이에 관계없이 일관된 안정성을 제공.

05 요약 (4/4)

항목	ViT	Swin Transformer v1	Swin Transformer v2
시기/목표	2020, 트랜스포머를 그대로 이미지에	2021, 고해상도 효율 확보	2022, 대규모·초고해상도 안정화/스케일링
토큰/구조	고정 패치(P16 등), 평평한 토큰열	계층형 피라미드(Patch Merging)	v1 계층 구조 유지
어텐션 범위	글로벌 MHSA($O(N^2)$)	윈도우 MHSA + Shift (국소 + 연결)	Scaled Cosine Attn(수치 안정)
위치 정보	절대 1D PosEmb	상대 위치 바이어스(RPB)	Log-CPB(해상도 일반화)
정규화	LayerNorm(Pre-LN)	LayerNorm	Res-Post-Norm 등 안정화 트릭
분류 방식	[CLS] 토큰	보통 풀링(CLSSless)	풀링(동일)
장점 키워드	단순/전역 문맥 강함	효율·성능 균형, 범용 백본	대규모/고해상도 안정·전이 우수
약점 키워드	고해상도 비효율	해상도 변경시 RPB 재적합 이슈	설계/튜닝 복잡도 ↑

ViT

“이미지를 문장처럼” —
전역 어텐션으로 단순하지만 고해상도에 비효율

Swin v1

“윈도우로 쪼개고 시프트로 잇는다” —
효율과 성능의 실용적 균형

Swin v2

“더 크고 깊게 안정적으로” —
스케일드 코사인 어텐션·Log-CPB·Post-Norm로
대규모 학습 안정화

06 코드구현(1/2)

학습 내용

▶ 224 x 224

▶ 클래스 당 200장 증강

▶ 다중 분류

▶ 30 Epochs

```
# =====
# 메인 실행 블록: 스크립트가 직접 실행될 때만 이 안의 코드가 동작합니다.
#
if __name__ == '__main__':
    print(f"사용하는 장치: {device}")
    print(f"총 {NUM_CLASSES}개의 클래스를 분류합니다.")
    print(f"클래스 목록: {class_names}")

    # --- 3. 모델 정의 ---
    # 비교할 모델들을 딕셔너리 형태로 정의
    models_to_train = {
        "Swin_Transformer": timm.create_model('swin_tiny_patch4_window7_224', pretrained=True, num_classes=NUM_CLASSES),
        "ViT": timm.create_model('vit_tiny_patch16_224', pretrained=True, num_classes=NUM_CLASSES)
    }

    # 각 모델의 학습된 가중치를 저장할 딕셔너리
    trained_models = {}

    # --- 4. 모델 학습 (반복) ---
    for model_name, model in models_to_train.items():
        print(f"\n{'='*20}")
        print(f"모델 학습 시작: {model_name}")
        print(f"{'='*20}")

        model = model.to(device)
        criterion = nn.CrossEntropyLoss()
        optimizer = optim.Adam(model.parameters(), lr=LEARNING_RATE)

        model.train()
        for epoch in range(NUM_EPOCHS):
            running_loss = 0.0
            running_corrects = 0

            for inputs, labels in dataloaders['train']:
                inputs, labels = inputs.to(device), labels.to(device)
                optimizer.zero_grad()

                outputs = model(inputs)
                loss = criterion(outputs, labels)
                _, preds = torch.max(outputs, 1)

                loss.backward()
                optimizer.step()

                running_loss += loss.item() * inputs.size(0)
                running_corrects += torch.sum(preds == labels.data)

            epoch_loss = running_loss / len(image_datasets['train'])
            epoch_acc = running_corrects.double() / len(image_datasets['train'])
            print(f'Epoch {epoch+1}/{NUM_EPOCHS} | Train Loss: {epoch_loss:.4f} Acc: {epoch_acc:.4f}'')
```

06 코드구현(2/2)

Train



Test



SWIN_V1: EUNSU
SWIN_V2: EUNSU
VIT: EUNSU

SWIN_V1: SUNJOON
SWIN_V2: HEEJOON
VIT: HEEJOON

SWIN_V1: JAEWOO
SWIN_V2: JAEWOO
VIT: JAEWOO

SWIN_V1: SUNJOON
SWIN_V2: SUNJOON
VIT: HEEJOON

SWIN_V1: WOOJIN
SWIN_V2: WOOJIN
VIT: WOOJIN

07 References

- [1] Icyking, “논문리뷰 — ViT (Vision Transformer) 이해,” 티스토리 블로그, <https://icyking.tistory.com/entry/%EB%85%BC%EB%AC%B8%EB%A6%AC%EB%B7%BO-ViTVision-Transformer%EC%9D%98-%EC%9D%B4%ED%95%B4>
- [2] Icyking, “논문리뷰 — Swin Transformer: Hierarchical Vision Transformer using Shifted Windows,” 티스토리 블로그, <https://icyking.tistory.com/entry/%EB%85%BC%EB%AC%B8%EB%A6%AC%EB%B7%BO-Swin-Transformer-Hierarchical-Vision-Transformer-using-Shifted-Windows>
- [3] 엔자이너TV, “[Vision Transformer] An Image is Worth 16 x 16 Words: Transformer for Image Recognition at Scale,” YouTube, 2022-07-17. <https://www.youtube.com/watch?v=91Qipj5NMnk>
- [4] A. Dosovitskiy, et al., “An Image is Worth 16×16 Words: Transformers for Image Recognition at Scale,” ICLR, 2021. (Vision Transformer)
- [5] Z. Liu, et al., “Swin Transformer: Hierarchical Vision Transformer using Shifted Windows,” ICCV, 2021.
- [6] Z. Liu, et al., “Swin Transformer V2: Scaling Up Capacity and Resolution,” CVPR, 2022.

감사합니다
THANK YOU

