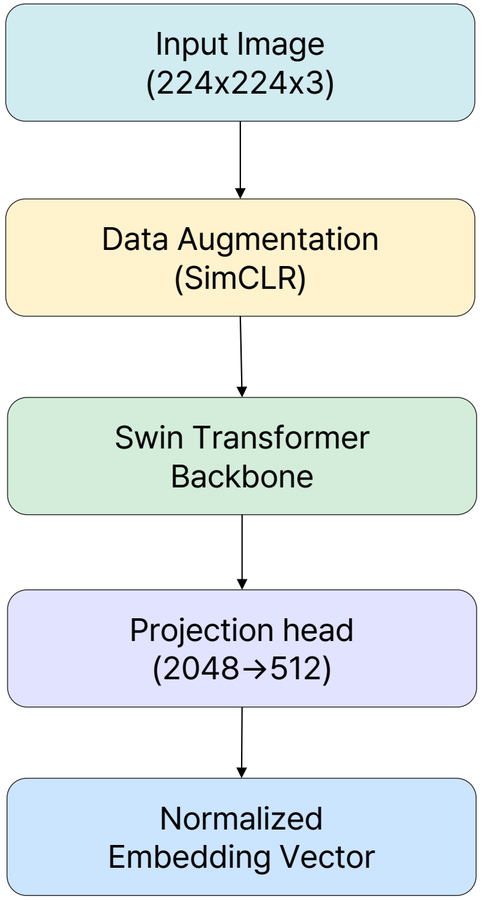
Project

로고 이미지 유사도 계산

SimCLR에 Swin Transformer를 백본으로 결합하여
로고 이미지 유사도 계산

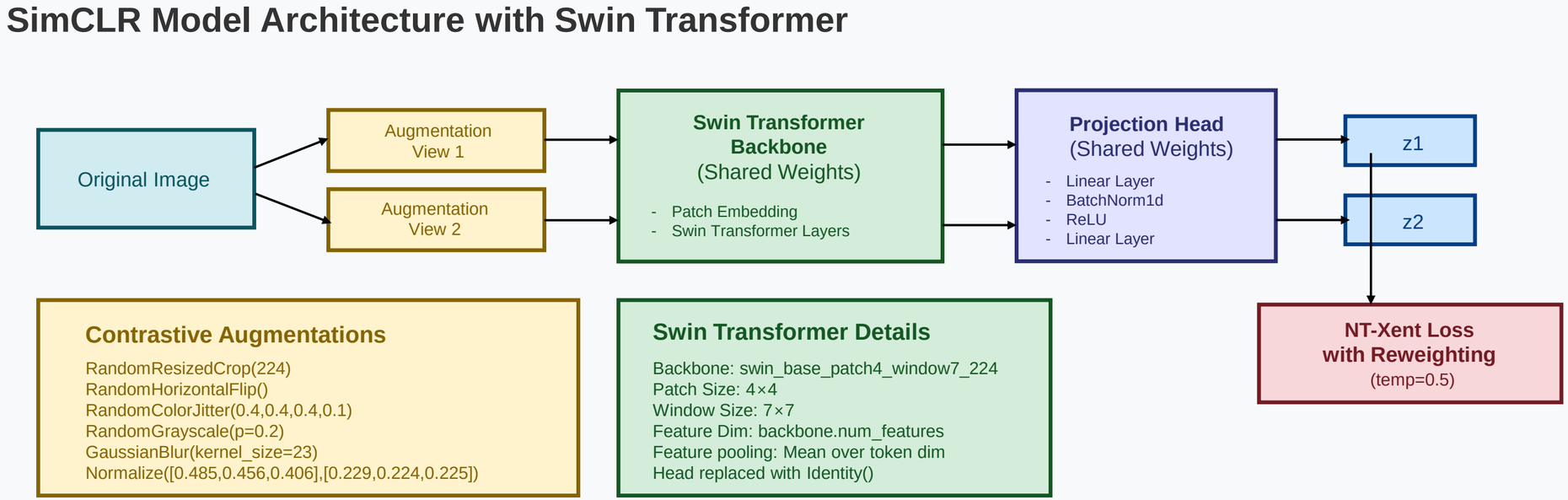
로고 이미지 유사도

프로젝트 기간	2025.03~2025.05
프로젝트 인원수	2 (서혜교, 전민석)
설명	이 프로젝트에서는 SimCLR 프레임워크를 활용하여 로고의 색상, 형태, 크기 변화 등 다양한 시각적 변형에도 견고한 임베딩을 학습했습니다. Swin Transformer 백본은 지역 윈도우 기반의 Self-Attention 구조를 통해 복잡한 로고 패턴과 전역 문맥을 효과적으로 캡처하기 위해 도입되었습니다. 아울러, NT-Xent Loss에 positive와 negative 페어별 가중치를 부여함으로써 False-Positive 발생을 억제하고, 핵심 샘플 학습에 더욱 집중할 수 있도록 설계하였습니다.
담당 역할	• **Contrastive learning 파이프라인 설계 및 구현** • **Swin Transformer 백본과 Projection Head 통합** • **NT-Xent Loss 커스터마이징(Positive/Negative 페어 가중치 적용)** • **학습 안정화 전략 수립: Projection Head 선학습 → Backbone 동시학습 전환** • **Multi-GPU 분산 학습 환경 구성 및 최적화** • 데이터 전처리 및 유사 이미지 필터링(imagehash 기반 구현) • Neptune.ai를 활용한 실험 로깅 및 모니터링
언어 및 환경	Python,Pytorch,SwinTransformer,SimCLR,Neptune.ai
느낀점	• **False-Positive 감소를 위해 Weighted NT-Xent Loss 설계가 결정적 역할 수행** • **Projection Head 선학습 후 Backbone 학습 전환 전략으로 안정적 수렴 달성** • Multi-GPU 도입으로 배치 크기 확장 및 학습 속도 2.5배 향상 • 클러스터링 필터링 메모리 이슈 해결 대안으로 imagehash 근사 필터링 도입 • Contrastive Learning 기법 이해도 및 Transformer 활용 역량 크게 향상 • 향후 하드 네거티브 마이닝 및 경량화 작업 필수

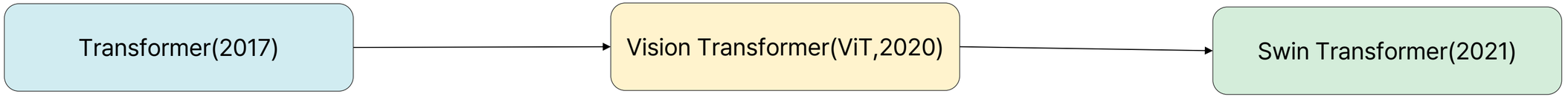


모델 아키텍쳐

프로젝트 아키텍쳐


SimCLR Model Architecture with Swin Transformer
Original Image
Augmentation View 1
Augmentation View 2
Swin Transformer Backbone (Shared Weights)
- Patch Embedding
- Swin Transformer Layers
Projection Head (Shared Weights)
- Linear Layer
- BatchNorm1d
- ReLU
- Linear Layer
z1
z2
NT-Xent Loss with Reweighting
(temp=0.5)
Contrastive Augmentations
RandomResizedCrop(224)
RandomHorizontalFlip()
RandomColorJitter(0.4,0.4,0.4,0.1)
RandomGrayscale(p=0.2)
GaussianBlur(kernel_size=23)
Normalize([0.485,0.456,0.406],[0.229,0.224,0.225])
Swin Transformer Details
Backbone: swin_base_patch4_window7_224
Patch Size: 4×4
Window Size: 7×7
Feature Dim: backbone.num_features
Feature pooling: Mean over token dim
Head replaced with Identity()

Swin Transformer 발전 흐름



- Multi-head Self-Attention
- Position Embedding
- Feed-Forward

- Patch Embedding
- Positional Encoding
- Transformer Encoder

- Pyramid 구조
- Window Self-Attention
- Shifted Window 매커니즘

Attention Is All You Need 논문에서 제안된 **Transformer**는 자연어 처리에서 self-attention 기반 아키텍처의 기초를 닦았습니다.

- **Scaled Dot-Product Attention:** Query, Key, Value 간 매트릭스 연산과 온도 파라미터($\sqrt{d_k}$) 역할
- **Multi-Head Attention:** 병렬화된 여러 Attention 헤드의 장점
- **Position-wise Feed-Forward Network:** 각 토큰별 독립적 비선형 변환
- **Residual Connection & Layer Normalization:** 안정적인 학습을 위한 구조
- **Encoder-Decoder 구조**

An Image is Worth 16×16 Words: Transformers for Image Recognition at Scale 논문에서, 이미지를 고정 크기 패치 시퀀스로 변환한 뒤 순수 Transformer Encoder만으로 학습하여 뛰어난 분류 성능을 보였습니다.

- **Patch Embedding:** 이미지 → 16×16 패치 → 선형 임베딩
- **Global Self-Attention:** 모든 패치 간 전역 상호작용
- **장점/단점:** CNN 없이 순수 Transformer 가능

Hierarchical Vision Transformer using Shifted Windows 논문에서 도입된 **Swin Transformer**는 다음과 같은 특징으로 ViT의 한계를 개선했습니다.

- **Hierarchical Feature Map**: CNN처럼 단계별로 해상도를 줄이며 채널 수를 늘려가며 피라미드 구조 생성
- **Window-based Self-Attention**: 전체 패치가 아닌 작은 윈도우 단위로 self-attention 수행 → 복잡도 선형화
- **Shifted Window**: 층마다 윈도우 배치를 교차(shift) 적용해 윈도우 간 정보 교류 강화
- **효율성**: 입력 해상도에 대해 O(N) 복잡도를 달성, 다양한 비전 태스크에 일반 목적 백본으로 활용

Transformer

Scaled Dot-product Attention

- Query, Key, Value 간 매트릭스 연산과 온도 파라미터($\sqrt{d_k}$) 역할

Multi-Head Attention

- 병렬화된 여러 Attention 헤드의 장점

Position-wise Feed-Forward Network

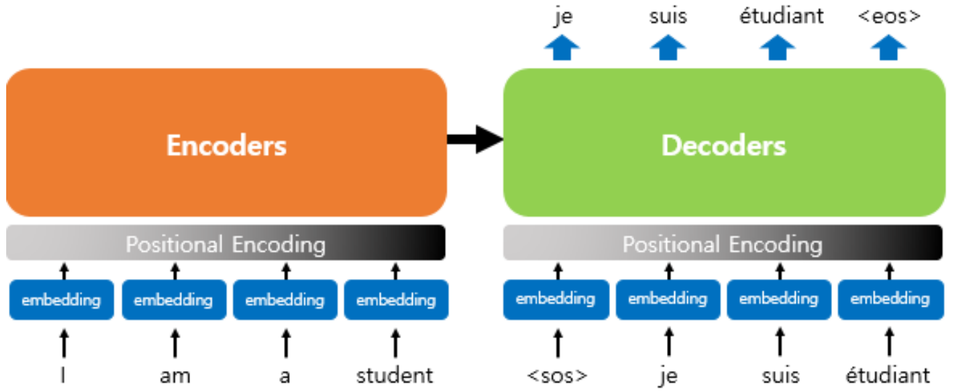
- 각 토큰별 독립적 비선형 변환

Residual Connection & Layer Normalization

- 안정적인 학습을 위한 구조

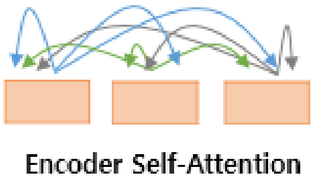
Encoder-Decoder 구조

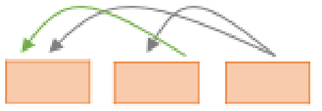
- 인코더 스택과 디코더 스택의 차이점

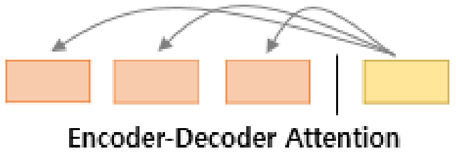


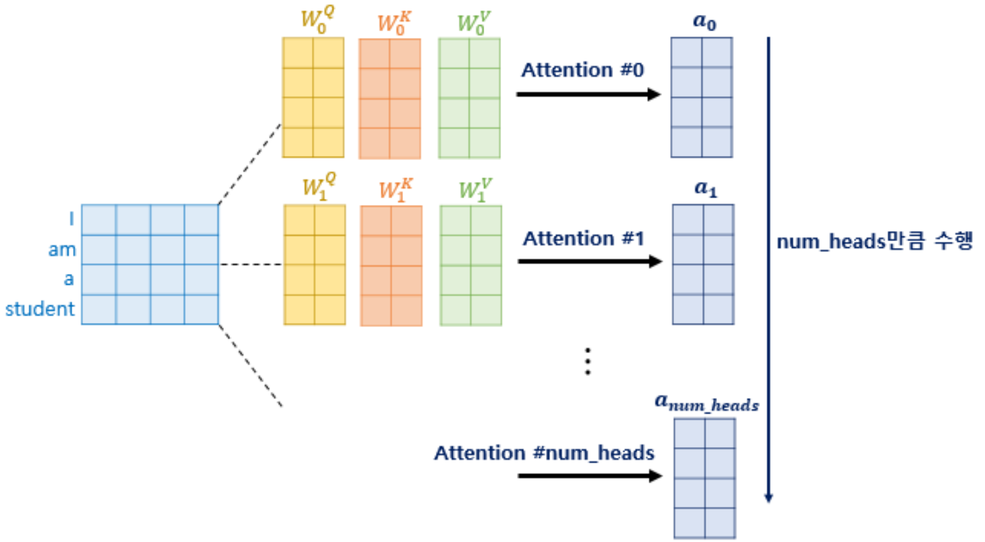
$$PE_{(pos,\,2i)} = sin(pos/10000^{2i/d_{model}})$$

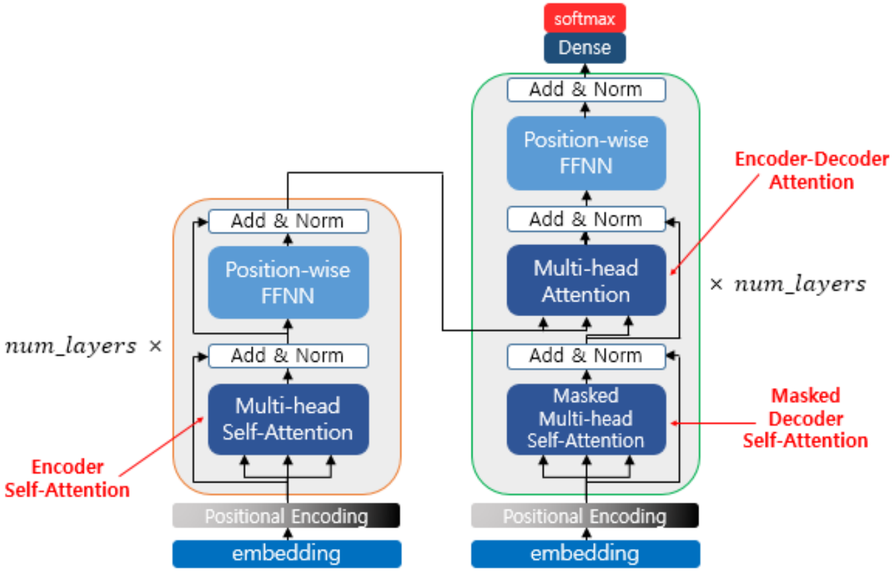
$$PE_{(pos,\,2i+1)} = cos(pos/10000^{2i/d_{model}})$$











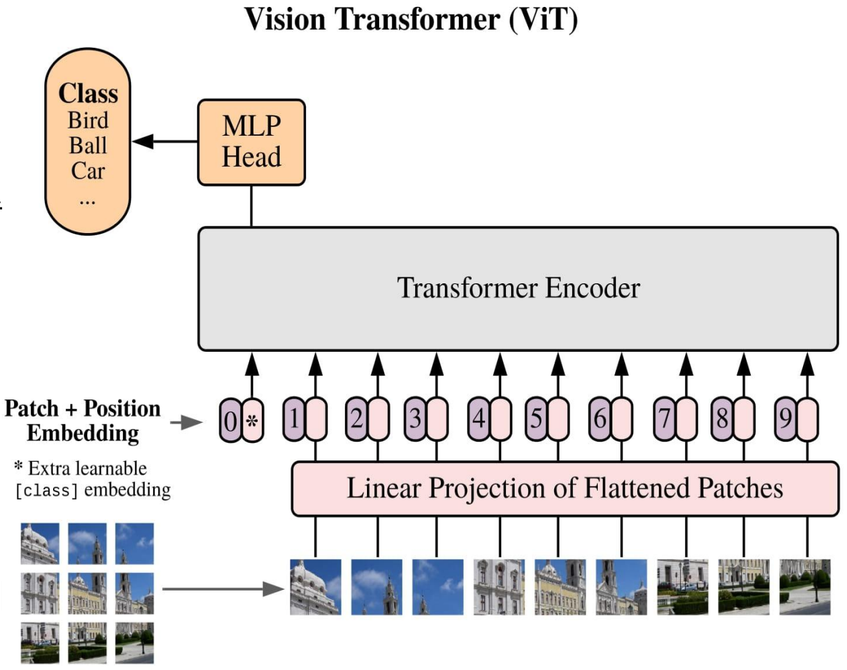
Vision Transformer(ViT)

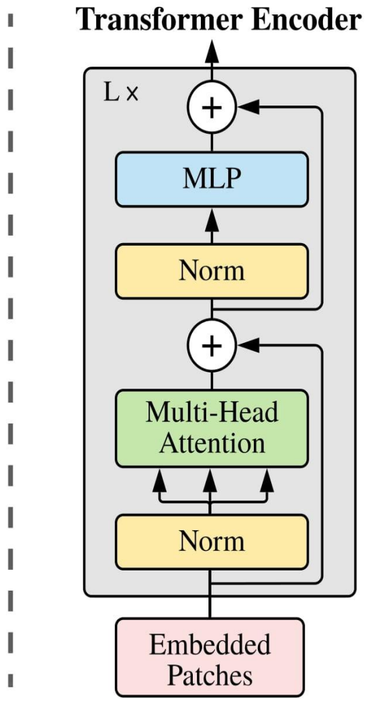
핵심 아이디어

- ImageNet같은 중간 규모 데이터셋으로 학습시킨 ViT는 강력한 정규화 없인, CNN이 가진 translation equivariance 및 locality같은 Inductive Bias를 가지고 있지 않기 때문에 일반화 성능이 떨어집니다.
- CNN은 이미지의 작은 영역(local)정보를 추출하는 convolutional filter를 사용합니다. 이는 이미지의 특정부분이 다른 부분과 독립적으로 의미를 가질 수 있다는 가정에 기반합니다.
- 하지만 대규모 데이터셋으로 학습시킨 ViT는 Inductive Bias를 극복합니다. 즉, ViT는 대규모 데이터셋으로 학습될 경우 Transformer가 컴퓨터비전 분야에서 CNN을 대체할 수 잇는 강력한 모델입니다.

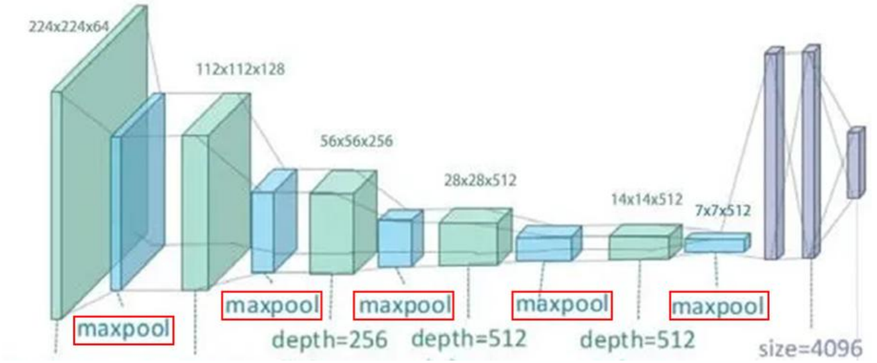
과정

1) image를 고정된 크기의 patch로 분할합니다.
2) 각 patch는 linear projecton을 통해 벡터형태로 변환됩니다.
3) 각 패치의 위치정보를 유지하기 위해 position embedding을 추가합니다.
4) Transformer Encoder 입력: 패치의 선형 임베딩과 위치 임베딩을 결합하여 생성된 벡터 시퀀스를 transformer 인코더에 입력합니다.
5) CLS토큰: 분류 작업을 수행하기 위해 학습가능한 “CLS토큰”을 시퀀스에 추가합니다.
6) Transformer인코더 출력에서 분류 토큰에 해당하는 벡터를 사용하여 이미지를 분류합니다





SWIN Transformer(1)

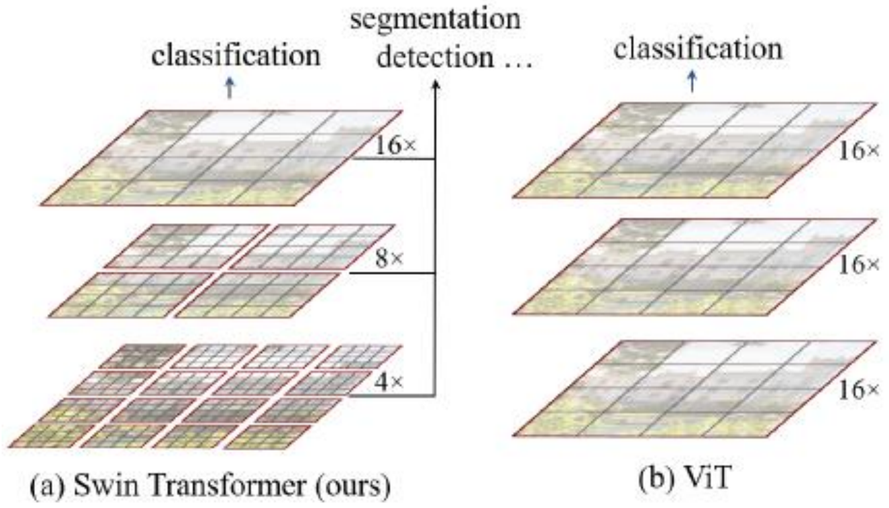


핵심 아이디어

- Swin Transformer는 **계층적 구조**와 **윈도우 기반 셀프어텐션**을 도입해, Vision Transformer(ViT)의 연산 효율과 지역 정보 학습 한계를 극복한 모델입니다.

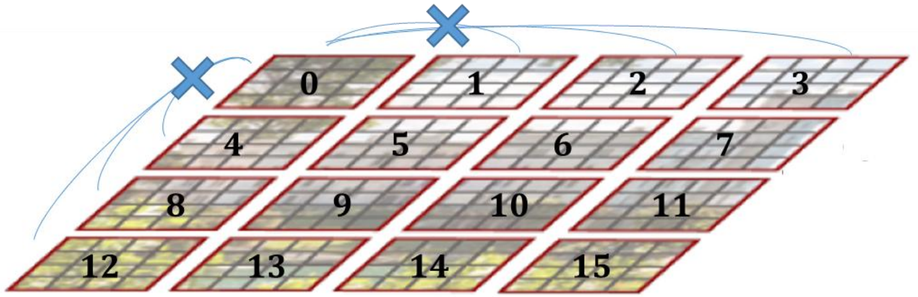
계층적 구조

- CNN의 pooling 구조를 Transformer에도 도입한게 Swin이 제안한 첫번째 기술입니다. 이렇게 함으로써 다양한 Representation을 학습할 수 있을뿐만아니라, 줄어든 해상도에서 Attention연산을 진행하기 때문에 속도에도 이점이 있습니다.



Shift Windows

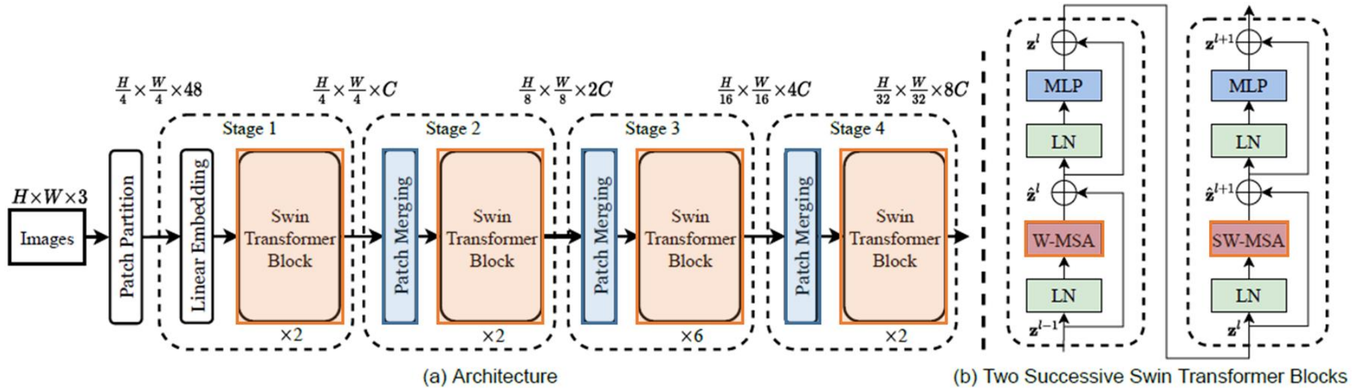
- ViT의 경우 이미지가 (b)처럼 패치로 나눠져 이 패치들이 각각의 Query, Key로 작동하여 기존 Transformer같이 연산이 됩니다. 하지만, 일반적으로 문장에서 단어길이는 많아야 50이지만, 그림의 경우 보통 224*224 해상도인데 이 모든 픽셀이 Query,Key로 작동하면 시간이 엄청 걸립니다. 그래서 (a)가 고안되었습니다.
- 고정된 Window(M*M 크기)를 이미지 패치에 적용하여, Attention의 연산이 Window안에서만 연산되게 합니다. 즉, 빨간 테두리 안에서만 연산 진행을 하는 것 입니다.
- [0-15]의 인덱스를 가지는 Windows가 존재하는데, 빨간 테두리 안에서만 연산하면 다른 Window끼리 연산이 안되기 때문에 Shift Windows 방식으로 Window끼리의 Attention 연산을 진행하는 것 입니다. 모든 Window끼리는 아니더라도 일부 Shift Windows방식을 적용하여 연산 속도와 정확도 둘 다 월등한 성능을 기록합니다.
- 뒷장에서 자세한 설명,,,



SWIN Transformer(2)

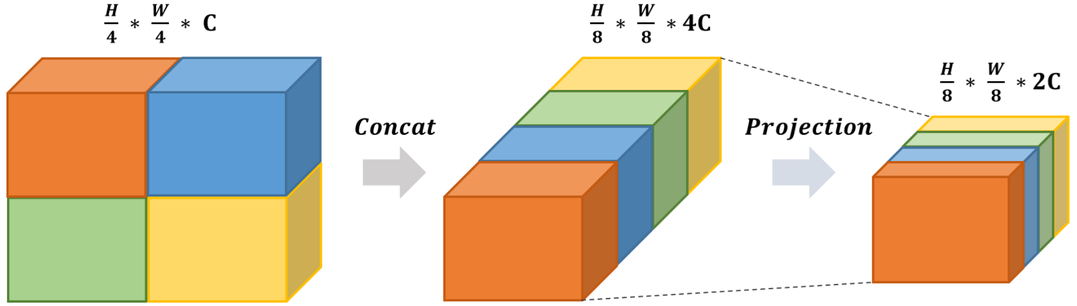
전체 구조

- 파란색 표시 부분은 계층적 구조를 가능케한 Patch Merging입니다.
- 그리고 주황색으로 표시한 부분은 Shift Windows를 기존 Transformer 블록에 적용한 것입니다.
- 나머지는 ViT와 동일합니다



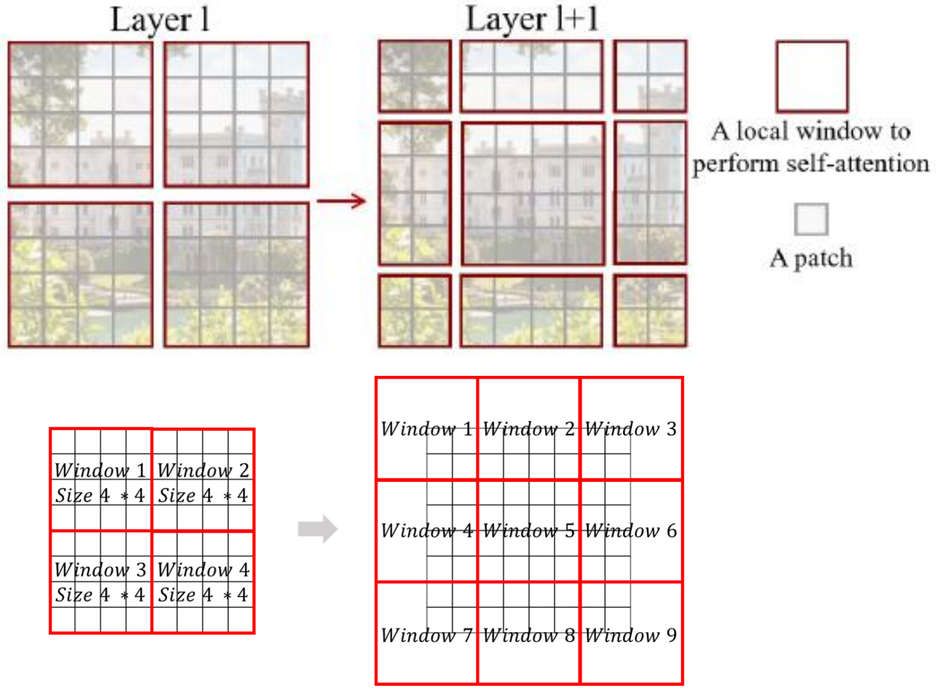
Patch Merging

- Input : H x W x 3의 해상도를 가지는 이미지가 입력으로 들어가고, 이미지가 겹치지 않게 각각 이미지 패치를 나눕니다
- Stage 1: Transformer 학습을 위해 사용자가 정의한 C차원으로 매핑해줍니다. (Linear Embedding , $\frac{H}{4} * \frac{W}{4} *$ C차원). 여기서 2개로 구성된 Swin Transformer Block으로 입력되어 동일한 차원인 $\frac{H}{4} * \frac{W}{4} *$ C가 출력됩니다.
- Stage 2: Patch Merging으로 $\frac{H}{4} * \frac{W}{4} *$ C의 해상도가 $\frac{H}{8} * \frac{W}{8} *$ 2C로 줄어듭니다. 그리고 2개의 Swin Transformer Block을 지나 동일한 차원인 $\frac{H}{8} * \frac{W}{8} *$ 2C를 출력합니다.
- Stage 3, Stage 4: Stage 2와 차원과 Swin Transformer Block의 개수만 다르고 나머지는 동일합니다



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

SWIN Transformer(3)

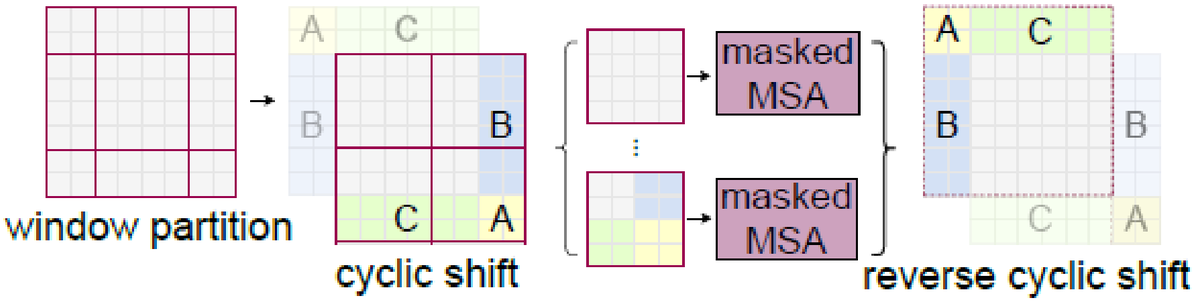


Shift Window

- Window로 쪼개서 연산하는 방식은 연산속도에 큰 이점이 있지만, 앞에서 말한 것과 같이 Window 끼리의 상호작용이 부족합니다. 그렇기 때문에 본 논문에서는 연산 속도에 이점은 가져가 돼, Window 끼리의 상호작용을 충분히 할 수 있는 navie 한 Shifted window 방식은 오른쪽과 같습니다.
- 2 * 2 파티션이 3 * 3개의 파티션으로 분할됩니다.
- 이러한 방식도 2 * 2의 windows가 3 * 3으로 늘어남에 따라 2.25배 정도의 계산량을 요구한다고, 본 논문에서는 이러한 추가적인 계산을 거의 요구하지 않는 cyclic-shifting 방식을 제안하였습니다.

Cyclic-shifting Window

- 기존 window 파티션에서 좌상단 부분을, 우하단으로 cyclic-shifting 시키는 것입니다.
- 이 상태에서 self-attention을 진행하면 각 window에 독립적으로 수행된 self-attention이 다른 window에도 적용될 수 있는 것입니다.
- 아래 그림에서 같은 색으로 칠해진 부분은 기존 2 * 2 windows에서 self-attention이 수행된 것이므로, 중복 attention 연산을 제한하기 위해 masked self-attention을 진행합니다.



SimCLR: Contrastive Learning(1)

핵심 아이디어

- Contrastive Learning은 같은 데이터의 서로 다른 augmentation 쌍(positive pair)은 가깝게, 다른 이미지 쌍(negative pair)은 멀어지도록 표현 학습을 진행하는 방식입니다.
- SimCLR은 별도의 메모리 뱅크나 복잡한 아키텍처 없이 이 과정을 단순화한 프레임워크입니다.

SimCLR의 구조

1. X라는 데이터셋을 각각 2번 augmentation 시켜서 xi, xj를 얻습니다.
2. 각각 xi와 xj에 f(x)를 적용한다(f(x)는 보통 Resnet-50, 본 프로젝트에선 Swin Transformer)
3. 그 다음 Projection head라고 불리는 g(x)를 적용합니다.

$$\ell_{i,j} = -\log \frac{\exp(\text{sim}(\boldsymbol{z}_i, \boldsymbol{z}_j)/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp(\text{sim}(\boldsymbol{z}_i, \boldsymbol{z}_k)/\tau)},$$

InfoNCE

4. Projection head에서 나온 embedding을 값을 InfoNCE(Contrastive loss)를 이용해서 서로의 유사도를 계산해서 loss function을 계산합니다. 여기서 보면 분자는 positive sample에 대한 서로의 유사도이고, 분모는 전체 데이터 셋(positive sample + negative sample)에 대한 유사도의 총합을 이용해서 probability를 계산합니다. 실제 코드에서는 NT-Xent Weighted Loss를 이용합니다.

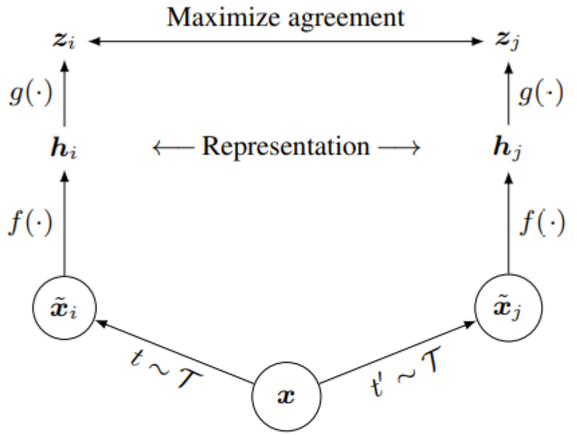
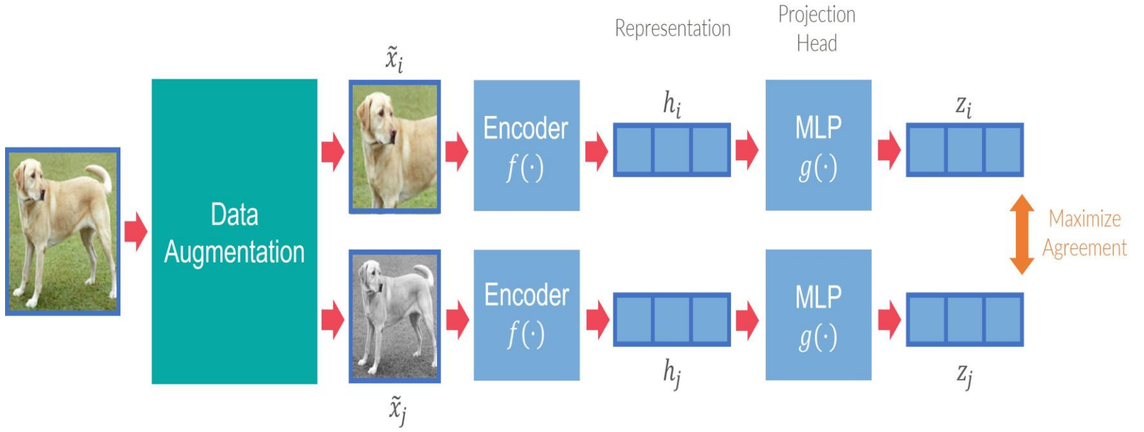


Figure 2. A simple framework for contrastive learning of visual representations. Two separate data augmentation operators are sampled from the same family of augmentations ($t \sim \mathcal{T}$ and $t' \sim \mathcal{T}$) and applied to each data example to obtain two correlated views. A base encoder network $f(\cdot)$ and a projection head $g(\cdot)$ are trained to maximize agreement using a contrastive loss. After training is completed, we throw away the projection head $g(\cdot)$ and use encoder $f(\cdot)$ and representation $\boldsymbol{h}$ for downstream tasks.



SimCLR: Contrastive Learning(2)

사용한 Augmentation

- 단일 Augmentation을 이용했을 때 성능이 가장 좋지 않았습니다.
- Color + Crop 조합을 이용했을 때 성능이 가장 좋다고 논문에서 확인할 수 있습니다.
- Color distortion을 사용해야, Deep learning의 shortcut학습을 방지할 수 있습니다.

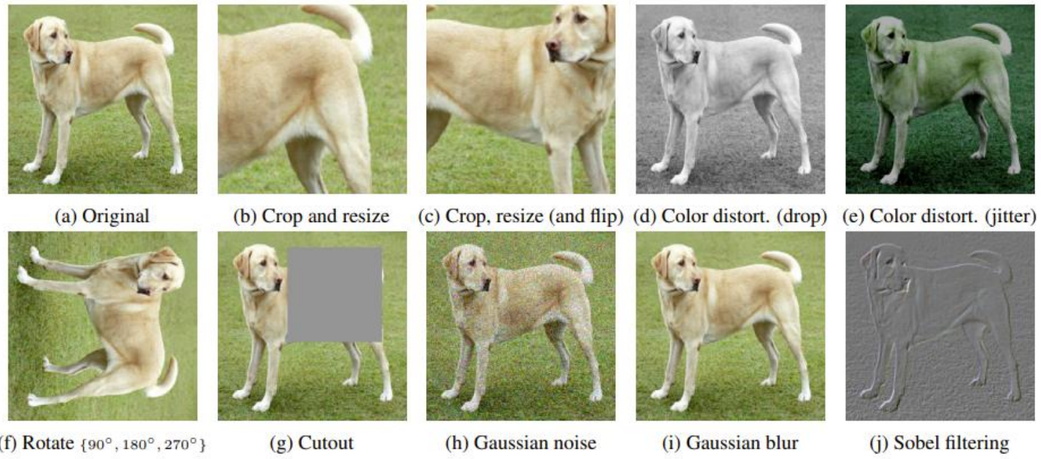


Figure 4. Illustrations of the studied data augmentation operators. Each augmentation can transform data stochastically with some internal parameters (e.g. rotation degree, noise level). Note that we *only* test these operators in ablation, the *augmentation policy used to train our models* only includes *random crop (with flip and resize)*, *color distortion*, and *Gaussian blur*. (Original image cc-by: Von.grzanka)

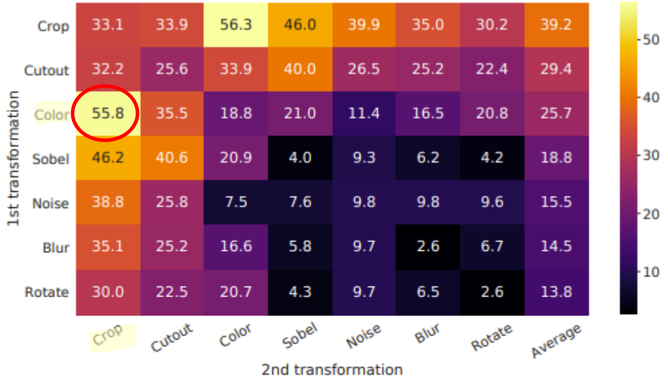


Figure 5. Linear evaluation (ImageNet top-1 accuracy) under individual or composition of data augmentations, applied only to one branch. For all columns but the last, diagonal entries correspond to single transformation, and off-diagonals correspond to composition of two transformations (applied sequentially). The last column reflects the average over the row.

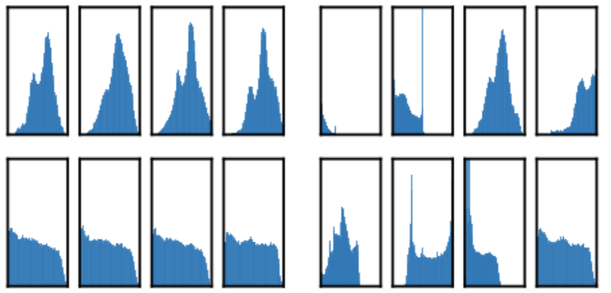


Figure 6. Histograms of pixel intensities (over all channels) for different crops of two different images (i.e. two rows). The image for the first row is from Figure 4. All axes have the same range.

SimCLR의 한계

- Batch size가 커야 성능이 높아진다는 것입니다.
- 이는 Computational cost에 대한 한계가 있고, negative sample에 대한 의존성이 높다는 것을 알 수 있습니다.
- 또한 SSL의 특성상 많은 데이터셋을 이용하기 때문에 보통 Multi-GPU로 학습하는데 이는 computational cost problem이 있다는 것이 명확한 한계점입니다.

NT-Xent Loss with Reweighting

NT-Xent Loss

- 같은 이미지의 두 view(augmentation)로부터 나온 임베딩을 가깝게(positive pair), 다른 이미지 view간 임베딩은 멀어지게(negative pair) 학습.
- 원리 : augmentation된 두 view 는 positive로 지정, 그 외 모든 view 조합을 negative로 간주. Positive 유사도(cosine)을 분자에, negative 유사도 합을 분모에 넣어 softmax crossentropy 형태로 계산합니다.
- 오른쪽 공식에서 sim(u,v)=$u^\top v$ (정규화된 코사인 유사도), τ는 온도 파라미터입니다.

$$\mathbb{l}_{i,j} = -\log \frac{\exp(\text{sim}(\mathbf{z}_i, \mathbf{z}_j)/\tau)}{\sum_{k=1}^{2N} 1_{[k \neq i]} \exp(\text{sim}(\mathbf{z}_i, \mathbf{z}_k)/\tau)}$$

NT-Xent Loss 공식

False-Negative Reweighting의 필요성

- **False Negative**: 같은 클래스(또는 매우 유사한 이미지)임에도 augmentation으로 분리되어 negative로 간주되는 쌍
- **목표**: "실제로는 positive일 수도 있는" negative 샘플들의 손실 기여를 줄여, 모델이 과도하게 멀리 떨어뜨리지 않도록 보정.

NT-Xent Loss with Reweighting

1. 유사도 계산 : 모든 임베딩 쌍 간의 cosine similarity 구해 softmax 입력(logit)으로 사용
2. False-Negative 판별: "유사도가 일정 기준(threshold) 이상인데 positive가 아닌 pair"를 탐지합니다.
3. 가중치 조정: 일반 negative는 weight=1으로 유지. False-negative로 판단된 쌍은 weight<1(예: 0.7)을 부여해, 손실에 미치는 영향을 줄입니다.
4. 최종 손실 계산: reweighting된 유사도를 softmax에 넣어, positiv는 여전히 강하게, false-negativ는 약하게 학습하도록 유도합니다.

NT-Xent Loss with Reweighting 효과

- **진짜 유사한 샘플**끼리는 과도하게 떨어뜨리지 않아, 표현 공간이 더 자연스럽게 클러스터링
- **잡음(Noise)나 클래스 내 변이**를 어느 정도 허용하면서도, 전체적으로 대조 학습의 장점은 유지