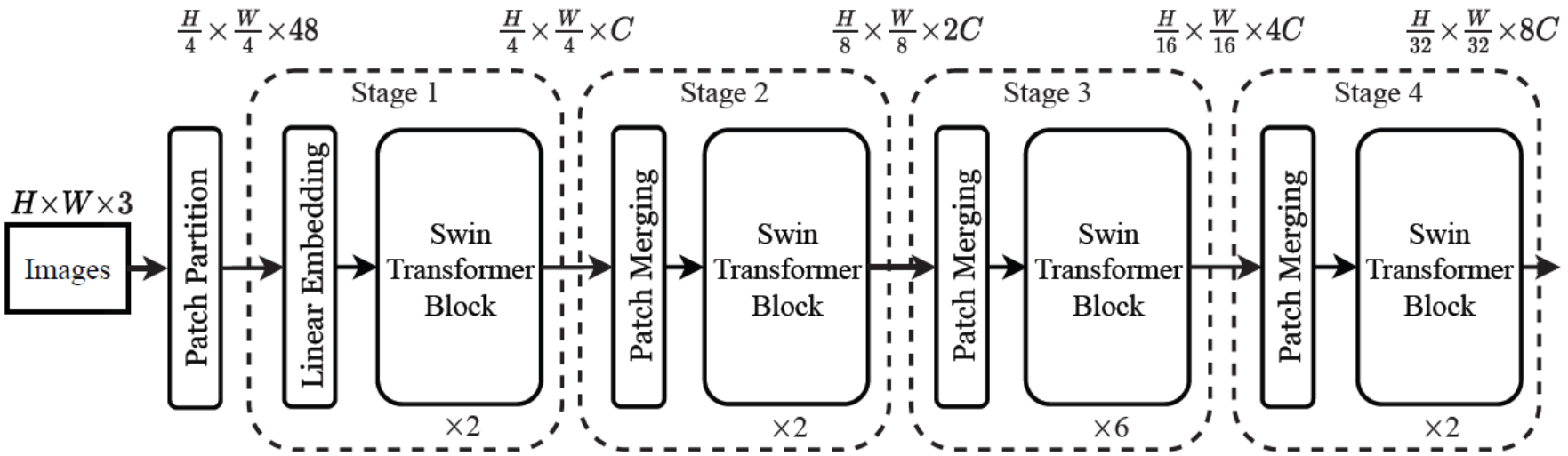
Swin Transformer

Sang Yup Lee

Swin Transformer

- 모형의 구조



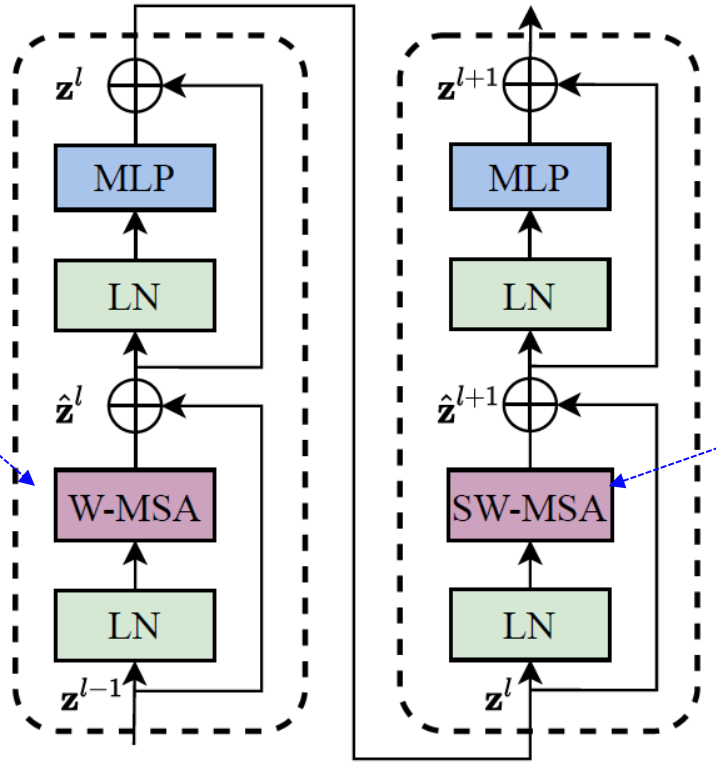
Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., ... & Guo, B. (2021). Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (pp. 10012-10022).

Swin Transformer

- 작동 원리
  - Stage 1
    - 원본 이미지를 여러 개의 겹치지 않는 패치들로 분할
      - 패치 크기는 4x4
      - 하나의 패치를 나타내는 벡터의 차원은 4x4x3 = 48
      - 토큰 수 = (H/4)x(W/4)
    - 여기에 linear embedding layer를 적용해서 C 차원의 임베딩 벡터를 생성
      - 가중치 행렬의 크기는 48xC
    - 여기에 Swin Transformer Blocks (다음 슬라이드 참고)을 적용

Swin Transformer

- 두 개의 연속된 Swin Transformer Blocks



Window-based MSA

MSA: Multi-head self-attention

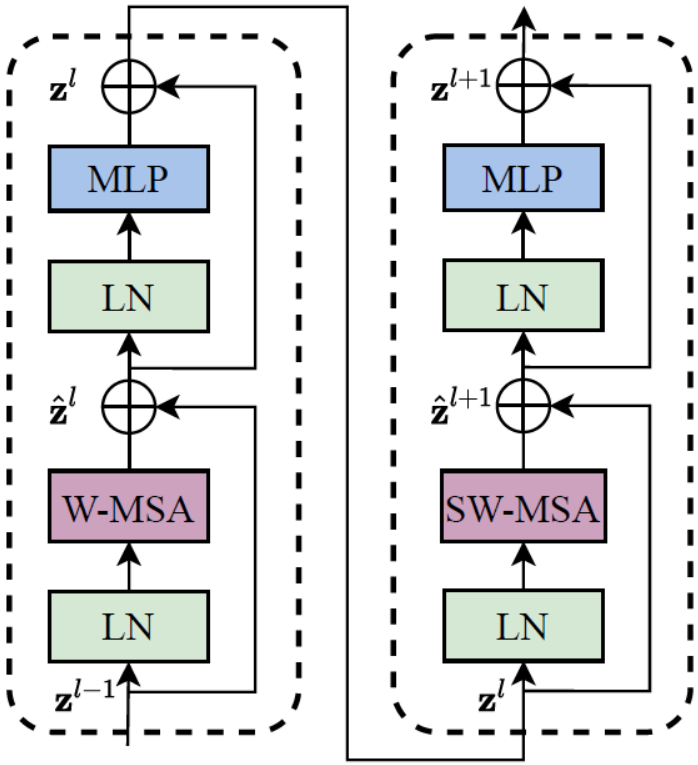
Shifted Window MSA

Swin Transformer

- 작동 원리 (cont'd)
  - Stage 2
    - 위계적 표상(hierarchical representation)을 위해서 patch merging layers를 이용해서 토큰의 수를 줄임
    - 2x2의 이웃하는 패치들의 벡터를 concat을 해서 하나의 벡터 생성
    - 토큰 수가 ¼ 감소 => (H/8)x(W/8)
    - 하나의 토큰의 벡터 차원이 C => 4C 벡터 생성
    - 다시 linear projection 적용 => 2C로 변경
    - 그 결과에 다시 Swin Transformer Blocks를 적용
  - 이 과정을 Stage 3과 4에서 반복
    - 토큰의 수 감소: Stage 3 => H/16xW/16, Stage 4 => H/32xW/32
    - 전통적 CNN (예, ResNet 등)과 같이 위계적 표상 가능

Swin Transformer

- Swin Transformer Blocks
  - Shifted window multi-head self attention 모듈 사용
  - 2-layer MLP 사용 (이는 Transformer에서 FFN)
    - GeLU 활성화 함수
  - Layer Norm의 순서가 원래와 다름
  - Skip connection 적용

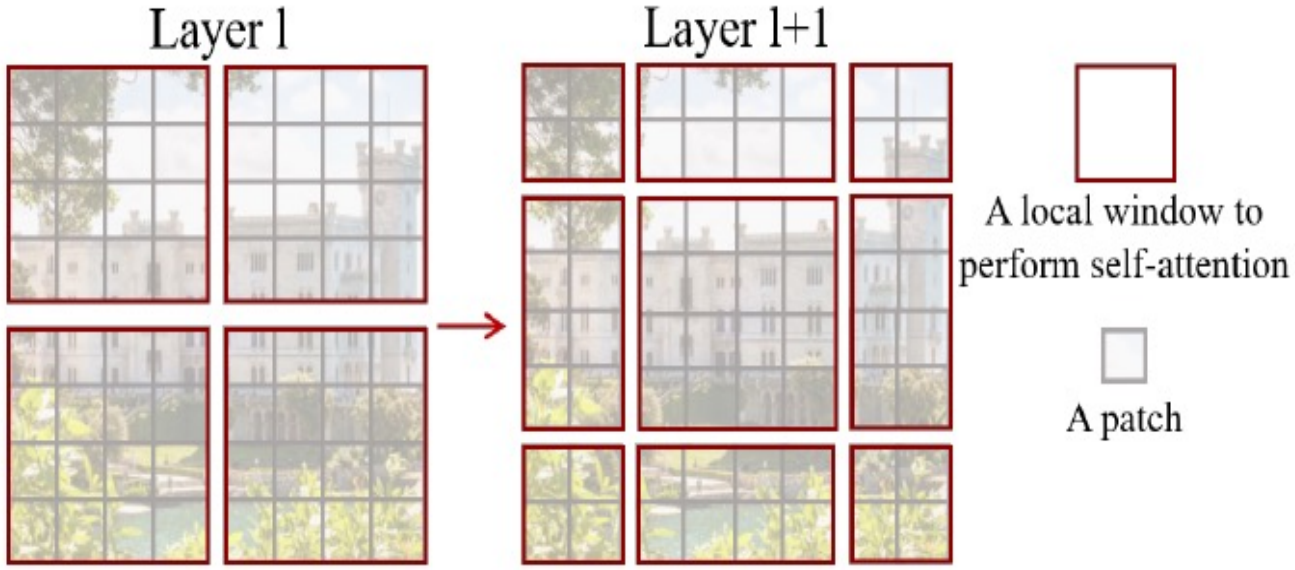


Swin Transformer

- (Shifted) Window-based Self-Attention
  - 기본 Transformer나 ViT에서는 self-attention이 globally 적용
    - 이러한 방법은 해상도가 높은 이미지 분석에 부적절 => 연산량 증가
  - Self-attention in non-overlapped windows
    - 보다 효율적인 학습을 위해서 local window 안에서의 self-attention 방법을 제안
      - 이 윈도우들은 이미지에 고르게 분포
  - Shifted window partitioning in successive blocks
    - 윈도우 기반의 self-attention 방법은 윈도우간의 연결이 부족 => 성능 저하의 원인
    - 이를 해결하기 위해서 shifted window partitioning 방법 제안 (다음 슬라이드 참고)

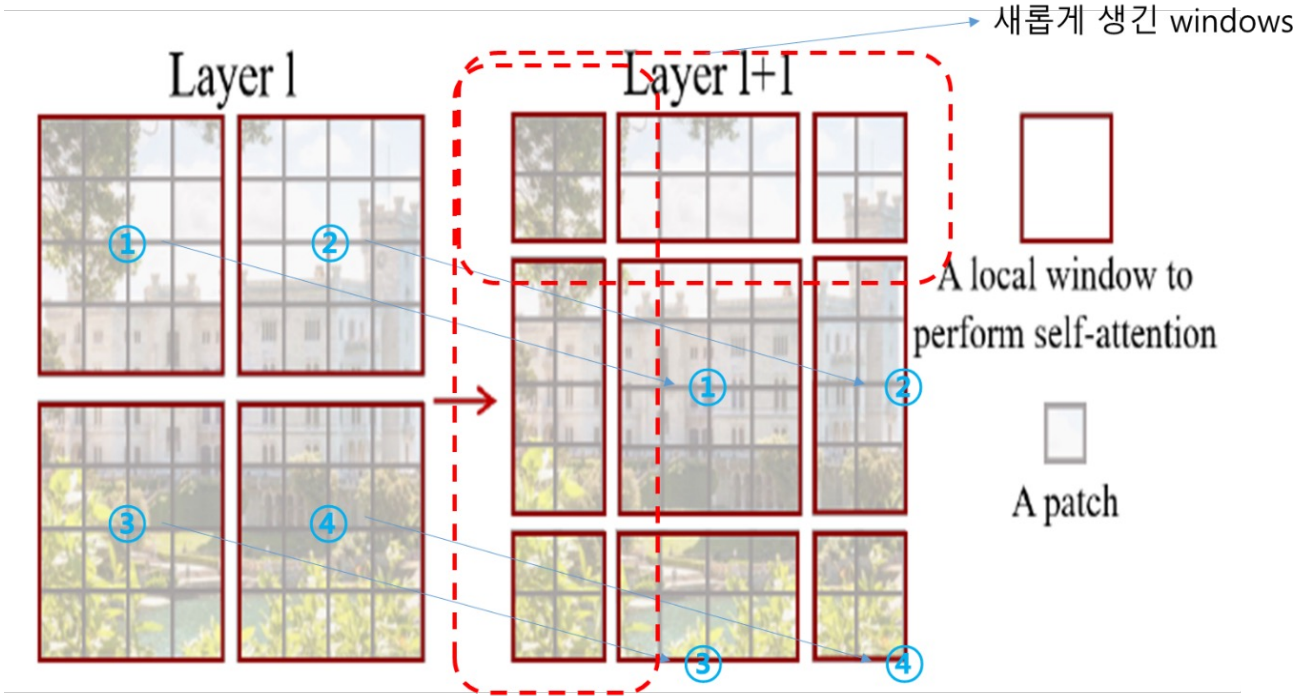
Swin Transformer

- Shifted window partitioning
  - 첫 번째 모듈은 일반적인 window partitioning 방법을 사용
    - 8x8 feature map이 4x4 (M=4) 크기의 4개의 윈도우로 구분
  - 그 다음은 shifted window 방법 적용 (M/2, M/2) 만큼 이동
  - 이렇게 하면 이전 단계에서 겹치지 않는 하지만 이웃한 윈도우 간의 연결을 만들 수 있어서 모형의 성능 향상



Swin Transformer

- Shifted window partitioning

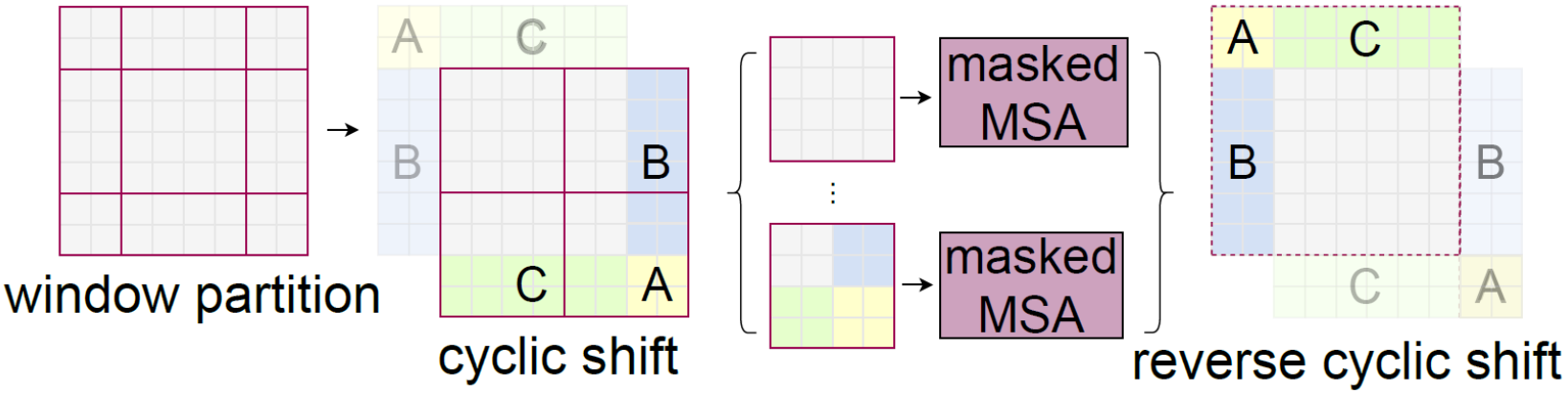


Swin Transformer

- Efficient batch computation for shifted configuration
  - Shifted window partitioning 방법의 한 가지 문제 => 윈도우 수 증가
    - (H/M)x(W/M) => (H/M+1)x(W/M+1)
    - 이전 슬라이드의 경우: 4 => 9
    - MxM 보다 작아지는 윈도우 발생
  - 단순 해결 방법
    - 패딩 방법을 사용해서 MxM 보다 작은 윈도우의 크기를 MxM으로 만든 후 padded 된 부분은 masking 해 버리는 것
    - 하지만 이렇게 하면 계산해야 하는 윈도우 수 증가

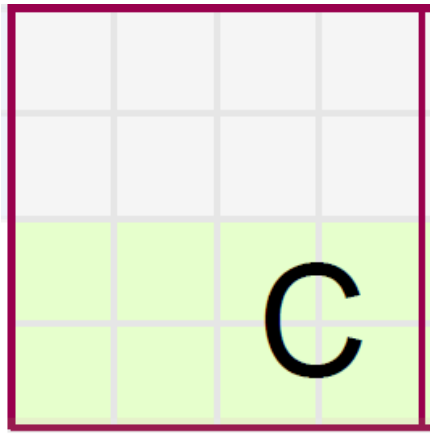
Swin Transformer

- Efficient batch computation for shifted configuration
  - 논문에서 사용한 방법 => cyclic-shifting 방법
  - 이렇게 하면 계산해야 하는 윈도우의 수는 동일
  - 하지만, 동일한 윈도우안에 원래의 feature map 상에서 이웃하지 않는 하위 윈도우들이 이웃하게 되는 경우가 발생



Swin Transformer

- Efficient batch computation for shifted configuration
  - 회색 부분에 존재하는 셀들과 초록색 부분에 존재하는 셀들 간의 self-attention 따로 진행
  - 이를 위해 마스킹 기법 사용

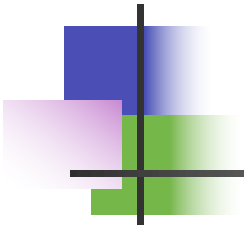


Swin Transformer

- Self-attention 계산
  - $\mathrm{Attention}(\mathrm{Q},\mathrm{K},\mathrm{V}) = \mathrm{SoftMax}\left(\frac{\mathrm{QK}^{\mathrm{T}}}{\sqrt{d_k}} + B\right)V$
  - B는 relative position bias를 의미
    - 본 논문에서는 위치 기반 임베딩 정보를 사용하지 않고, relative position bias를 사용했음
    - 이미지에 존재하는 각 패치 간의 상대적 위치를 파악하는 방법
    - https://www.youtube.com/watch?v=2lZvuU_IIMA&t=1884s 참고

Swin Transformer

- Hugging Face
  - https://huggingface.co/docs/transformers/model_doc/swin#transformers.SwinModel



Q & A