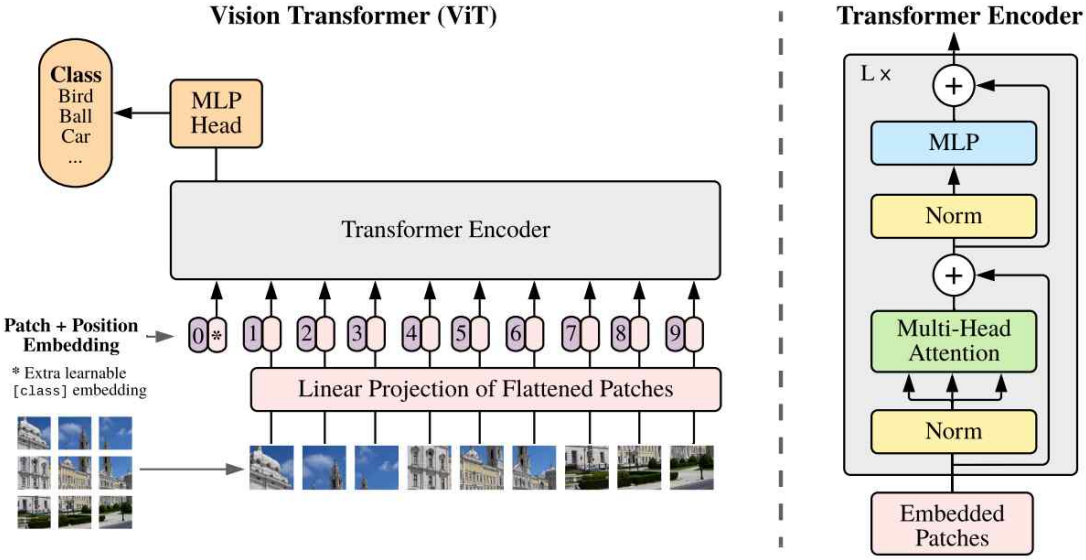
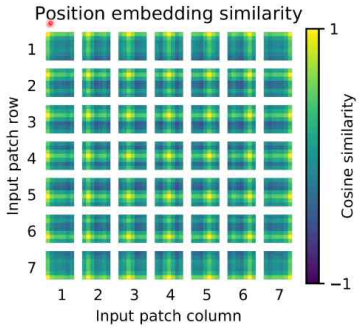
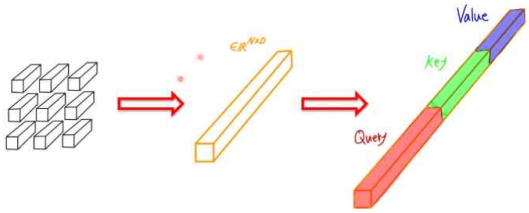
An Image is Worth 16 x 16 Words : Transformer for Image Recognition at Scale 하고자 하는 것: Image Classification (input 이미지가 무엇인지 구별)



- input 이미지를 9개의 patch로 나누어줌
- 그 순서대로 각각의 패치들을 임베딩 해(1차원 벡터로 펼침) 선형변환 하여 고정된 차원의 벡터로 변환 - 모든 패치가 같은 차원이 되고, 모델이 처리할 수 있는 형태로 정렬
- 각각의 패치는 이미지 내에서의 순서가 중요하기 때문에, 각각의 패치의 가장 앞부분에 position embedding을 더해줌으로써 위치 값을 보존함 (1)
- Position Embedding이 포함된 임베딩 벡터들을 Transformer 인코더에 입력함. 각 패치는 Query, Key, Value로 변환되어 self- attention 메커니즘을 통해 상호작용하며, 이미지 내에서 의미 있는 특징을 학습함





- 각 임베딩된 패치 벡터를 Normalization한 후(2. LN), Concatenation으로 모두 합쳐줌
- self attention 구조에 들어가기 위해 합쳐진 vector에 각각 다른 행렬(weight)을 곱해서 Query, Key, Value로 나눔.
- Multi-Head Attention 구조로 Self-Attention을 반복하여 다양한 관점에서 패치 간의 상호작용을 학습(2. MSA)
- skip connection 기법을 통해 multi head attention을 통과한 출력값과 통과하지 않은 원래 입력값을 더해 기존의 값을 보존해줌. (2. $+z_{l-1}$)
- normalization 해줌 (3. LN)
- multi layer perceptron(MLP)을 통과한 값과 기존의 값을 한 번 더 더해주는 skip connection을 통해 최종 output 출력 (3)
- Transformer Encoder의 최종 output은 MLP를 통해 최종적으로 이미지가 어떤 클래스에 속하는지 Classification (4)

● 디코더 없이 인코더만으로 이미지의 의미 있는 특징을 추출하고, 이를 기반으로 MLP에서 분류 작업을 수행하는 구조임

$$\mathbf{z}_0 = [\mathbf{x}_{\text{class}}; \mathbf{x}_p^1\mathbf{E}; \mathbf{x}_p^2\mathbf{E}; \cdots; \mathbf{x}_p^N\mathbf{E}] + \mathbf{E}_{pos}, \qquad \mathbf{E} \in \mathbb{R}^{(P^2 \cdot C) \times D},\ \mathbf{E}_{pos} \in \mathbb{R}^{(N+1) \times D} \tag{1}$$

$$\mathbf{z}'_\ell = \text{MSA}(\text{LN}(\mathbf{z}_{\ell-1})) + \mathbf{z}_{\ell-1}, \qquad \ell = 1 \ldots L \tag{2}$$

$$\mathbf{z}_\ell = \text{MLP}(\text{LN}(\mathbf{z}'_\ell)) + \mathbf{z}'_\ell, \qquad \ell = 1 \ldots L \tag{3}$$

$$\mathbf{y} = \text{LN}(\mathbf{z}_L^0) \tag{4}$$

-> 출력된 y vector를 MLP를 통해서 이미지 classification 수행

(1): 초기 입력 임베딩: 이 식은 Transformer에 입력될 초기 임베딩 벡터를 구성하는 방식입니다.

- x_{class}: 이미지 전체를 대표하는 벡터. 이미지의 전반적인 정보를 종합하여 최종적으로 이미지가 어떤 클래스에 속하는지 예측하는 데 중요한 역할을 함- 이 class 토큰과 각 패치 임베딩 벡터들을 함께 모델에 입력해 이미지 전체와 각 패치 정보가 동시에 학습되도록 함
- x_p^i: 각 패치 i 를 나타내며, 이미지에서 잘라낸 패치 조각들이 여기에 해당됩니다.
- E: 패치를 고정된 차원의 벡터로 변환하기 위한 학습 가능한 선형 변환 행렬입니다. 각 패치 x_p^i에 E를 곱하여 고정된 차원의 임베딩 벡터로 만듭니다.
- E_{pos}: 위치 임베딩으로, 각 패치의 위치 정보를 저장합니다. 이를 추가함으로써 모델이 패치의 위치를 인식할 수 있게 합니다.

결과적으로, 초기 임베딩 벡터 z_0는 각 패치의 벡터와 위치 정보를 합한 형태로 Transformer에 입력됩니다.