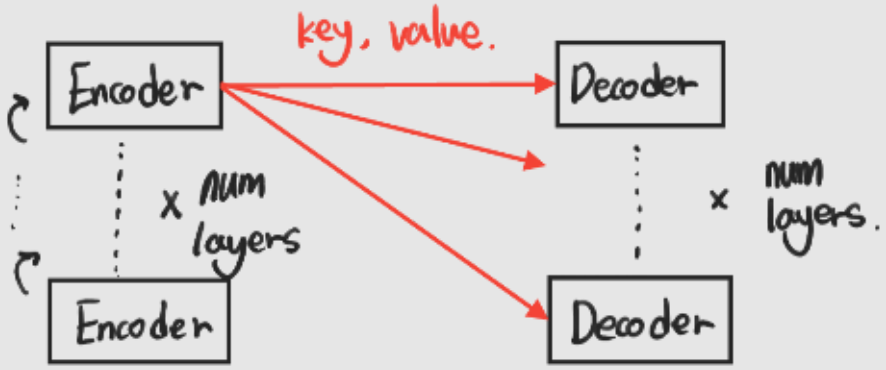
TRANSFORMER 공부

- Transformer의 변수.
  - d_{model} : 인코더, 디코더에서 정해진 입출력 차원수.
    (인코더에서 디코더로 값을 전달할 때도 d_{model} 차원으로 전달)
  - num_layers : 인코더·디코더 층의 개수.
  - num_heads : 인코더·디코더의 Multi-head Attention 에서 헤드
  - d_{ff} : 트랜스포머의 내부 FFNN (피드 포워드 신경망)의 파라미터 수

- Transformer 의 기본 구조.

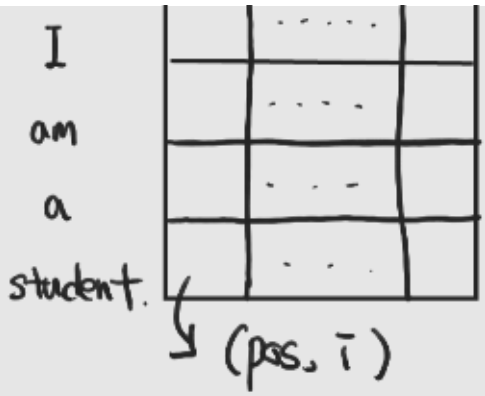


- 포지셔널 Encoding
  - RNN 과 같은 순환 신경망의 경우 학습시 sequence 의 순서대로 입력을 받아 단어의 위치 정보가 자동으로 추가된다.
  - 하지만 Transformer 의 경우 RNN과 입력 방식이 다르기 때문에 단어의 위치 정보를 따로 전달할 필요가 있음.

식

$$\begin{cases} PE(pos, 2i) = \sin(pos / 10000^{2i/d_{model}}) \\ PE(pos, 2i+1) = \cos(pos / 10000^{2i/d_{model}}) \end{cases}$$

ex)



∴ 최종 입력값 : PE + embedding-layer의 값.

- 멀티 헤드 어텐션 (Multi-head Attention)
  - v 기존의 어텐션 vs 셀프 어텐션.
  - 기존 어텐션. → 시점에 따라 "모든"으로 변형 가능
    - Q : t 시점의 디코더 은닉 상태
    - K : 모든 시점의 인코더 은닉상태
    - V : 모든 시점의 인코더 셀의 은닉상태.
  - 셀프 어텐션
    - Q : 입력 문장의 모든 단어 벡터
    - K : "
    - V : "
- Q · K · V 얻기.
  - num_heads에 따라 W^Q, W^K, W^V 각 가중행렬의 shape 정의. $d_{model} \times (d_{model} / num\_heads)$
    - ↳ 연산을 위해 나누어 떨어져야 함.
  - 각 단어는 정의된 가중행렬 W^Q, W^K, W^V에 따라 / $\times (d_{model} / num_heads)$의 Q, K, V 벡터 생성.
- 스케일드 닷-프로덕트 어텐션 (scaled dot-product Attention)

score function $(q, k) = q \cdot k / \sqrt{d_{model}}$

↳ Scaling.

ex) I am a student.

$Q_I \times k_I = I' \quad / \sqrt{d_{model}}$
$\times k_{am} = am' \quad / \sqrt{d_{model}}$
$\times k_a = a' \quad / \sqrt{d_{model}}$
$\times k_{student.} = student' \quad / \sqrt{d_{model}}$

Attention Distribution

$\Rightarrow$ I" $\times V_I$, am" $\times V_{am}$, a" $\times V_a$, student' $\times V_{student}$ $+$ $+$ $+$ $=$

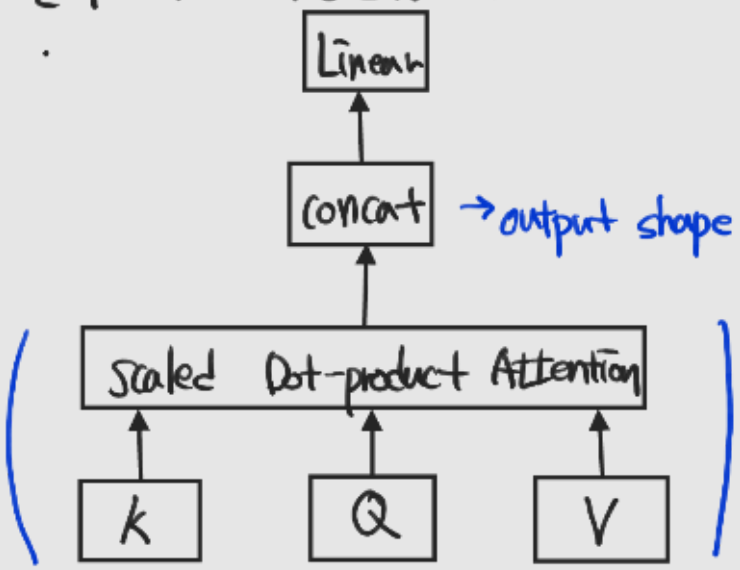
Attention score.

Attention Value.

∴ 결과 나오게 되는 Attention Value 행렬의 shape은

/ × (d_{model} / num_heads)

• 멀티-헤드 어텐션 (Multi-Head Attention)



→ output shape : (seq_len, d_{model})

× num_heads 만큼 존재.

• 포지션 와이즈 피드 포워드 신경망 (Position-wise FFNN)

$FFNN(x) = MAX(0, xW_1 + b_1)W_2 + b_2$

↳ 여기서 x는 Multi-Head Attention을 거친.

seq_len × d_{model} 형태의 행렬.

W_1은 $d_{model} \times d_{ff}$ shape의 가중치.

W_2는 $d_{ff} \times d_{model}$

- 잔차 연결. (Residual connection)

$H(x) = x + f(x)$

↳ FFNN

* 잔차 연결 이란 ??

- 영상처리 분야에서 많이 사용하던 연결 방식

$$x_L = x_l + \sum_{i=l}^{L-1} F(x_i, w_i)$$

→ 잔차 연결이 여러 층으로 되어 있는 경우.

"미분"

$$\frac{\partial \epsilon}{\partial x_l} = \frac{\partial \epsilon}{\partial x_L} \frac{\partial x_L}{\partial x_l} = \frac{\partial \epsilon}{\partial x_L}\left(1 + \frac{\partial}{\partial x_L}\sum_{i=l}^{L-1} F(x_i, w_i)\right)$$

gradient 소실 문제가 적어짐.

- 층 정규화 (Layer Normalization)

✓ Batch Normalization 이란?

→ 공변량 변화 문제 해결

↳ 딥러닝에서 각 Layer를 빠져나온 Layer output은 Layer Input과 서로 다른 분포를 보일 수 있다.

이는 hidden layer의 학습에 부정적 영향을 미칠 수 있다.

∴ 공변량 변화를 줄이기 위해 Normalizing 필요.

⇒ hidden Layer의 중간 층에서 hidden Layer의 Input을 정규화 (0, 1)의 값으로 정규화.

식

$$\mu_B \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \quad \text{특징 값의 평균}$$

$$\sigma_B^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_B)^2 \quad \text{특징 값의 분산.}$$

$$\hat{x}_i \leftarrow \frac{x_i - \mu_B}{\quad} \quad \text{정규화}$$

$\sqrt{\sigma_B^2 + \varepsilon}$ ← 분모가 0이 되는 것 방지.

$y_i \leftarrow \gamma \hat{x}_i + \beta = BN_{\gamma,\beta}(x_i)$

=> 학습을 통해 변환하는 조정 파라미터

↳ 대부분의 경우 $\hat{x}_i$는 0과 매우 근접한 값을 가짐

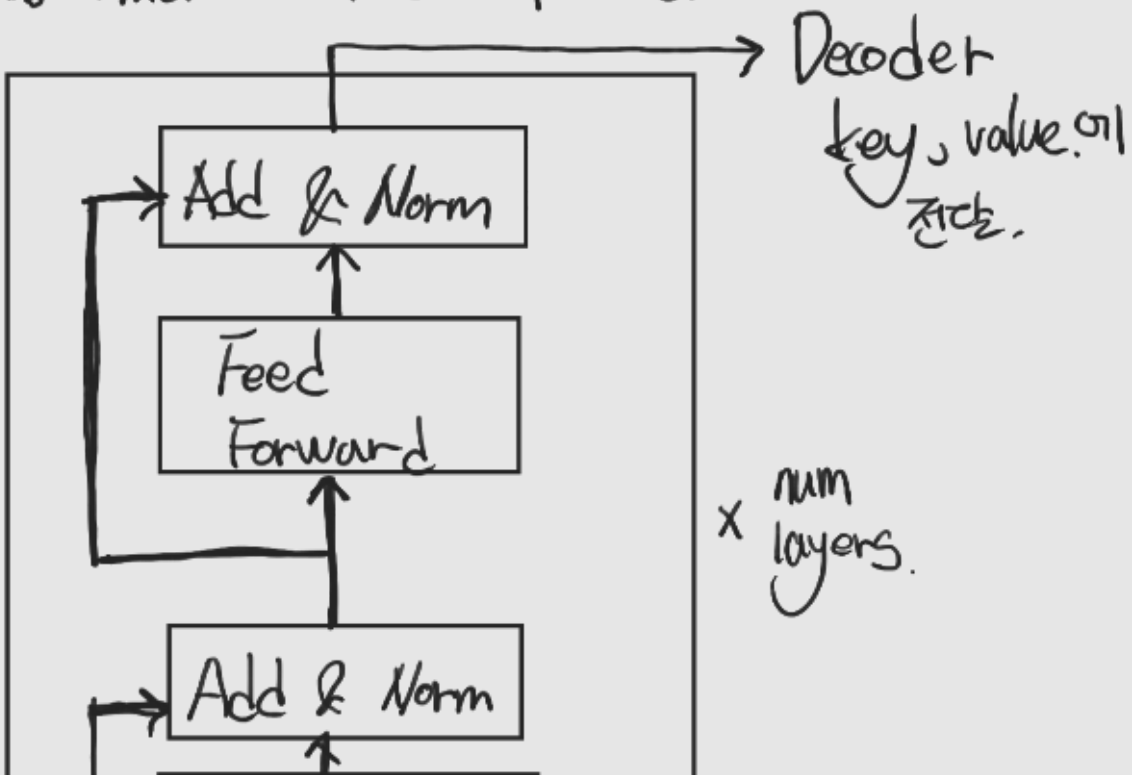
이 경우 Sigmoid 함수에서 선형 구간에 빠져 모델을 단순하게 만드는 경우 발생.

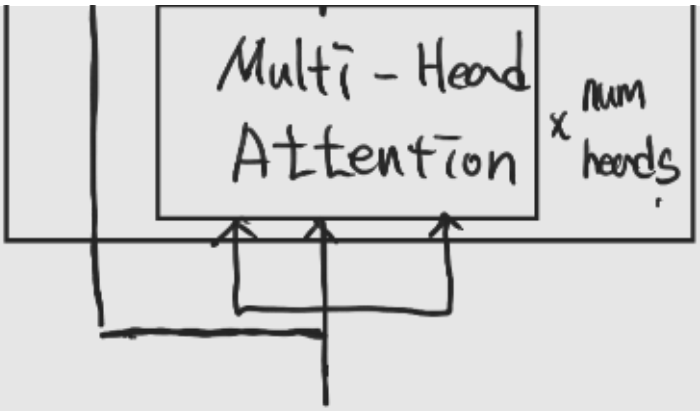
∴ γ, β로 이를 조정함.

- Layer Normalization.
  - Batch 단위의 평균·분산의 경우 실제 모든 데이터의 평균과 분산에 값의 차이가 존재할 수 있으며 이는 Batch 마다 분포가 달라져 학습에 시간이 오래 걸림.

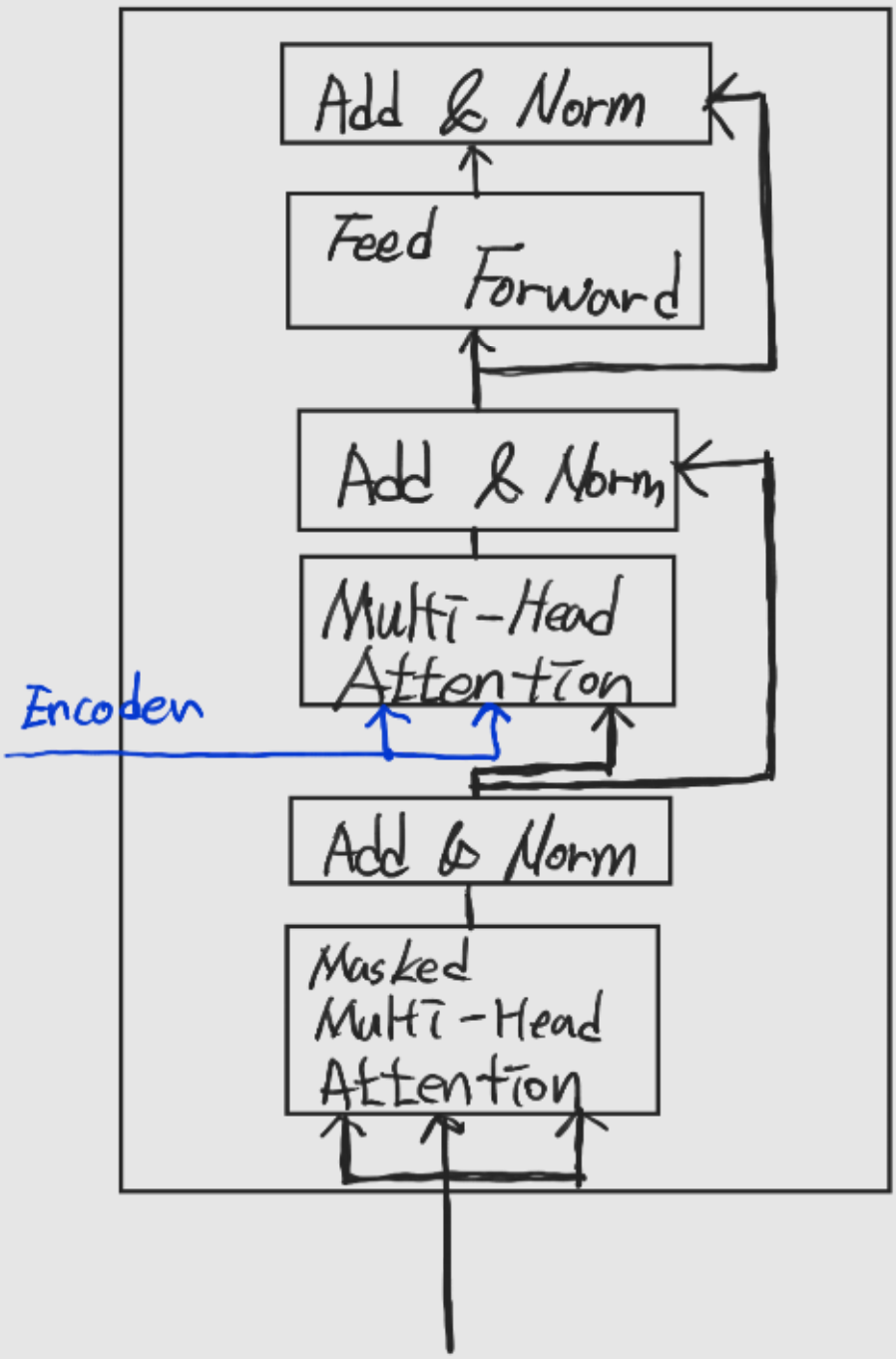
∴ Layer Normalization은 모든 데이터의 각 특징에 대한 평균과 표준편차를 구해 사이 차를 구함.

- Transformer Encoder의 구조.




Multi - Head
Attention
x num heads

- Decoder의 모형


Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Encoder
Add & Norm
Masked Multi-Head Attention