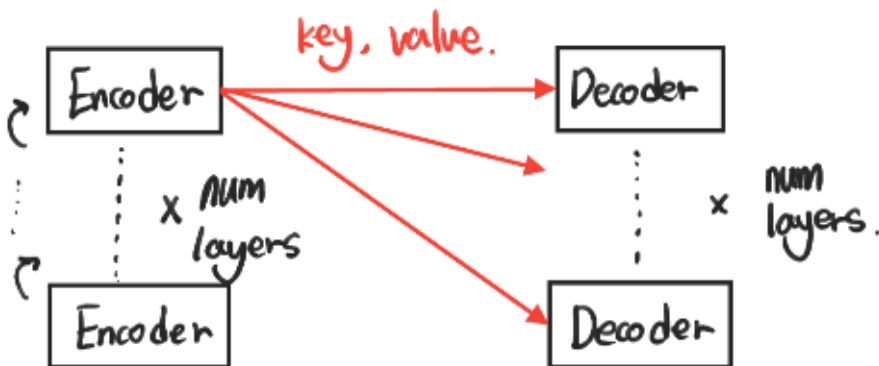


TRANSFORMER 공부

• Transformer의 변수.

- d_{model} : 인코더, 디코더에서 정해진 임베딩 차원수.
(인코더에서 디코더로 값을 전달할 때도 d_{model} 차원으로 전달)
- num_layers : 인코더, 디코더 층의 개수.
- num_heads : 인코더, 디코더의 Multi-head Attention에서
헤드
- d_{ff} : 트랜스포머의 내부 FFNN(퍼포워드 신경망)의 파라미터 수

• Transformer의 기본 구조.



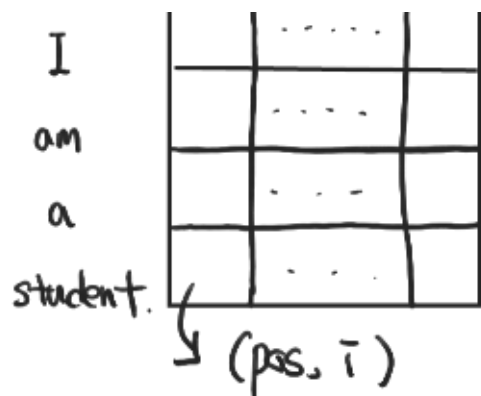
• 포지셔널 Encoding

- ✓ RNN과 같은 순환 신경망의 경우 하층의 sequence의 순서대로 입력을 받아 단어의 위치 정보가 자동으로 추가된다.
- ✓ 하지만 Transformer의 경우 RNN과 입력 방식이 다르기 때문에 단어의 위치 정보를 따로 전달할 필요가 있음.

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

ex)





∴ 최종 입력값 : PE + embedding-layer의 값.

• 멀티 헤드 어텐션 (Multi-head Attention)

v 기준의 어텐션 vs 셀프 어텐션.

• 기준 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 만.

• num_heads에 따라 W^Q, W^K, W^V 각 가중행렬의 shape

정의. $d_{model} \times \frac{d_{model}}{num_heads}$

↳ 연산을 위해 나누기 필요해야 함.

• 각 단어는. 정의된 가중행렬 W^Q, W^K, W^V 에 따라.

$1 \times (d_{model} / num_heads)$ 의 Q, K, V 벡터 생성.

• 스케일드 닷-프로덕트 어텐션 (scaled dot-product Attention)

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

→ Scaling.

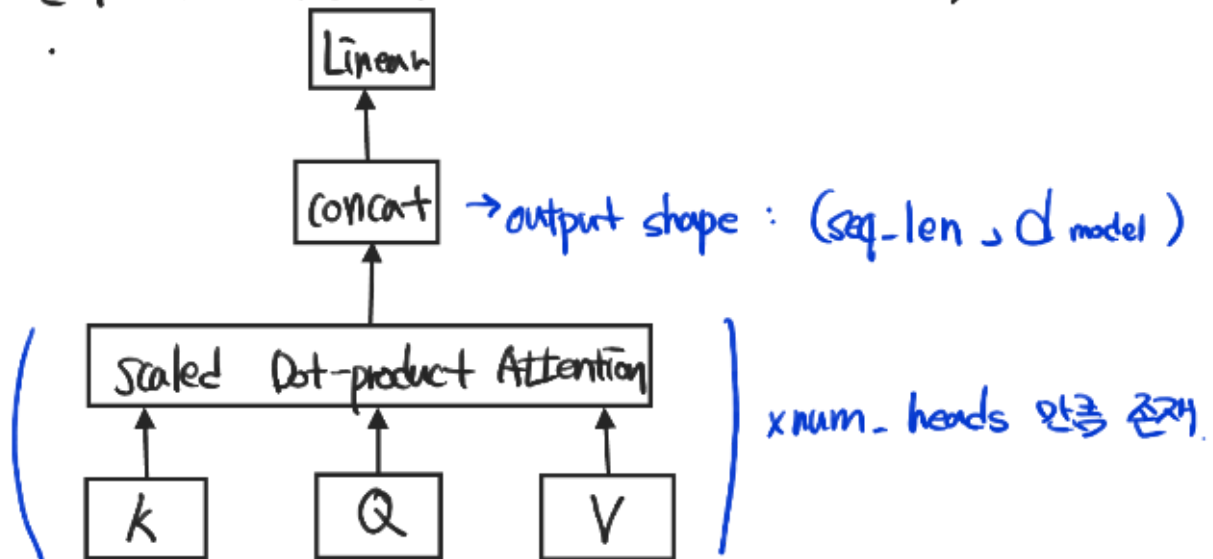
ex) I am a student.

$$\begin{array}{lcl}
 Q_I \times K_I & = & I' \\
 \times K_{am} & = & am' \\
 \times K_a & = & a' \\
 \times K_{student} & = & student'
 \end{array}
 \begin{array}{l}
 / \sqrt{d_{model}} \\
 / \sqrt{d_{model}} \\
 / \sqrt{d_{model}} \\
 / \sqrt{d_{model}}
 \end{array}
 \Rightarrow
 \begin{array}{l}
 I'' \\
 am'' \\
 a'' \\
 student''
 \end{array}
 \begin{array}{l}
 \times V_I \\
 \times V_{am} \\
 \times V_a \\
 \times V_{student}
 \end{array}
 \begin{array}{l}
 + \\
 + \\
 + \\
 +
 \end{array}
 = \text{Attention Value}$$

Attention Distribution Attention Score

∴ 결과로 나오게 되는 Attention Value 행렬의 shape은 $1 \times (d_{model} / num_heads)$

• 멀티-헤드 어텐션 (Multi-Head Attention)



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

$$FFNN(x) = \text{MAX}(0, xW_1 + b_1)W_2 + b_2$$

→ 여기서 x 는 Multi-Head Attention을 거친.

$seq-len \times d_{model}$ 행렬의 행렬.

W_1 은 $d_{model} \times d_{ff}$ shape의 가중치.

W_2 은 $d_{ff} \times d_{model}$

- 잔차 연결. (Residual connection)

$$H(x) = x + \underbrace{f(x)}_{\rightarrow \text{FFNN}}$$

* 잔차 연결이란??

- 영상처리 분야에서 많이 사용하던 연결 방식

$$x_L = x_l + \sum_{i=1}^{L-1} F(x_i, w_i) \rightarrow \text{잔차 연결이 여러층으로 되어있는 경우.}$$

"미분"

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

gradient 소실 문제가 적다.

- 층 정규화 (Layer Normalization)

✓ Batch Normalization. 아란?

→ 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

$$\begin{aligned} \mu_B &\leftarrow \frac{1}{m} \sum_{i=1}^m x_i && \text{특정 값의 평균} \\ \sigma_B^2 &\leftarrow \frac{1}{m} \sum_{i=1}^m (x_i - \mu_B)^2 && \text{특정 값의 분산} \\ \hat{x}_i &\leftarrow \frac{x_i - \mu_B}{\sigma_B} && \text{정규화} \end{aligned}$$

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

$\sqrt{\sigma^2 + \epsilon}$ → 분포가 0이 되는 것 방지.
 \Rightarrow 학습을 통해 변환하는 조정 파라미터.

→ 대부분의 경우, γ 는 0과 매우 근접한 값을 가짐
 이 경우, Sigmoid 함수에서 선형 구간이 빠져
 모델을 단조하게 만드는 경우 발생.
 너무

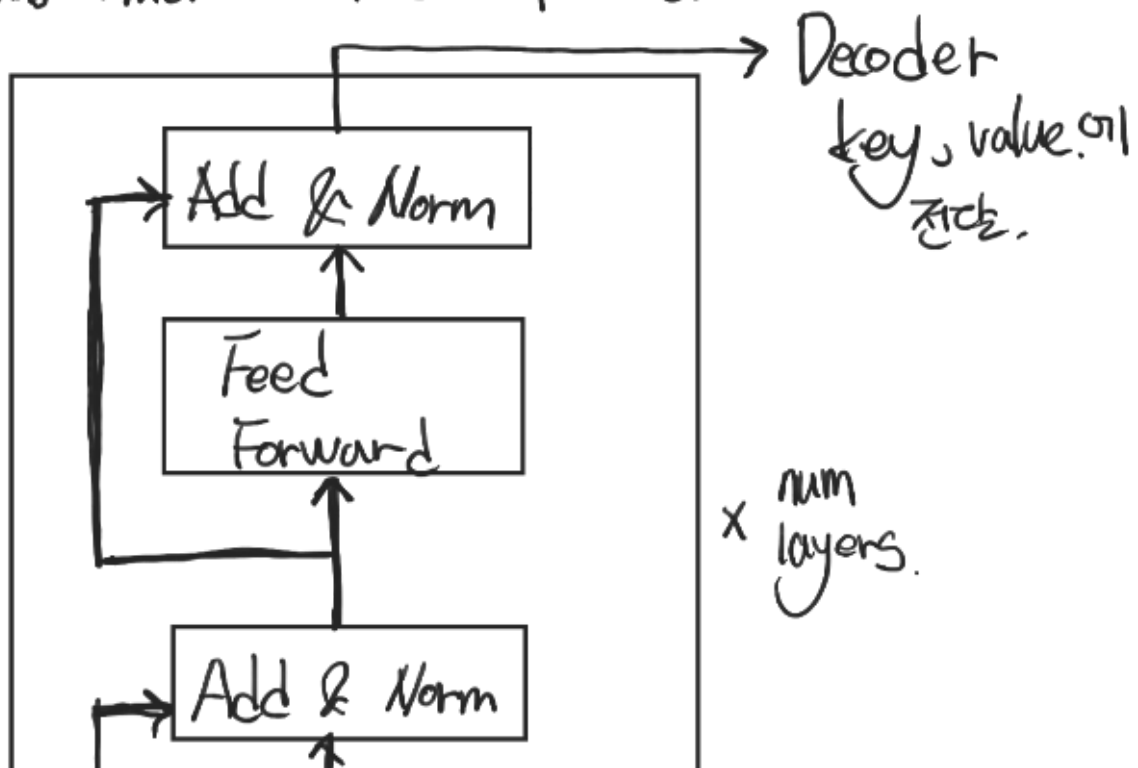
$\therefore \gamma, \beta$ 로 이를 조정함.

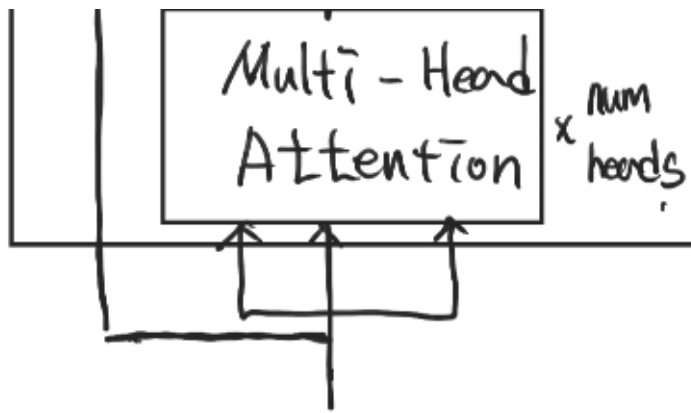
Layer Normalization.

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

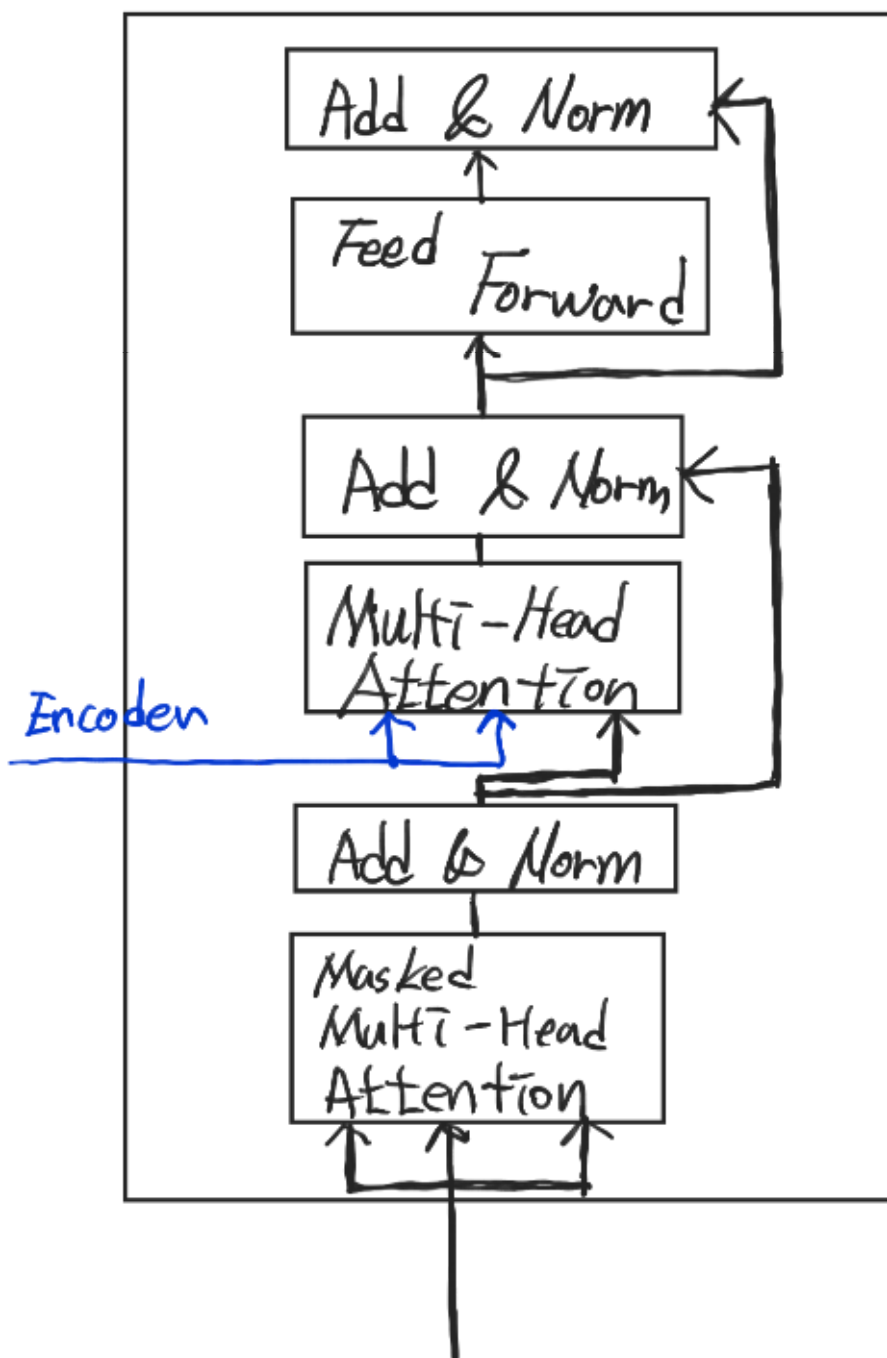
\therefore Layer Normalization은 모든 데이터의 각 특징에 대한 표준편차와 표준 편차를 위해 사제 제곱.

Transformer Encoder의 구조.





- Decoder의 구성



• Masked self Attention.

Decoder의 self Attention이다. n 번째 단계를 예측하기 $n + \alpha$ 번째의 단계를 정보를 사용하여 학습을 수행하는 경우 단계 $n + \alpha$ 번째의 단계를 외치는 경우가 생길 수 있다.

따라서 이런 문제를 해결하기 위해 Attention Block에 Masking 필요.

Masked 된 부분에 가중치를 주지 softmax 연산과 같이 커지도록 변화.