Part 3
Base model

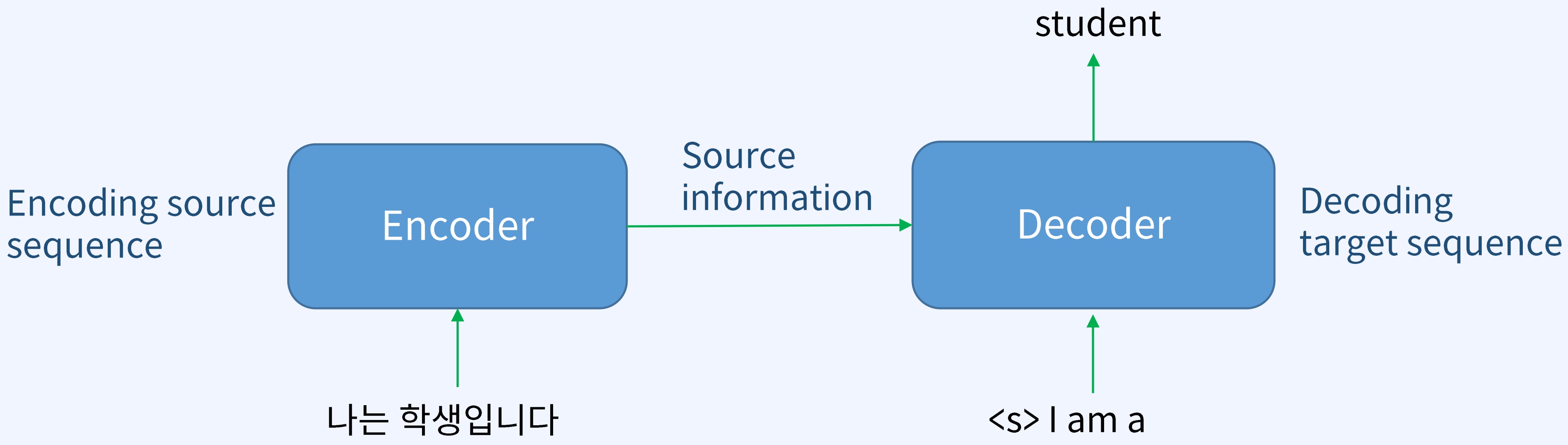
Ch1. Transformer

Attention Is All You Need

Transformer

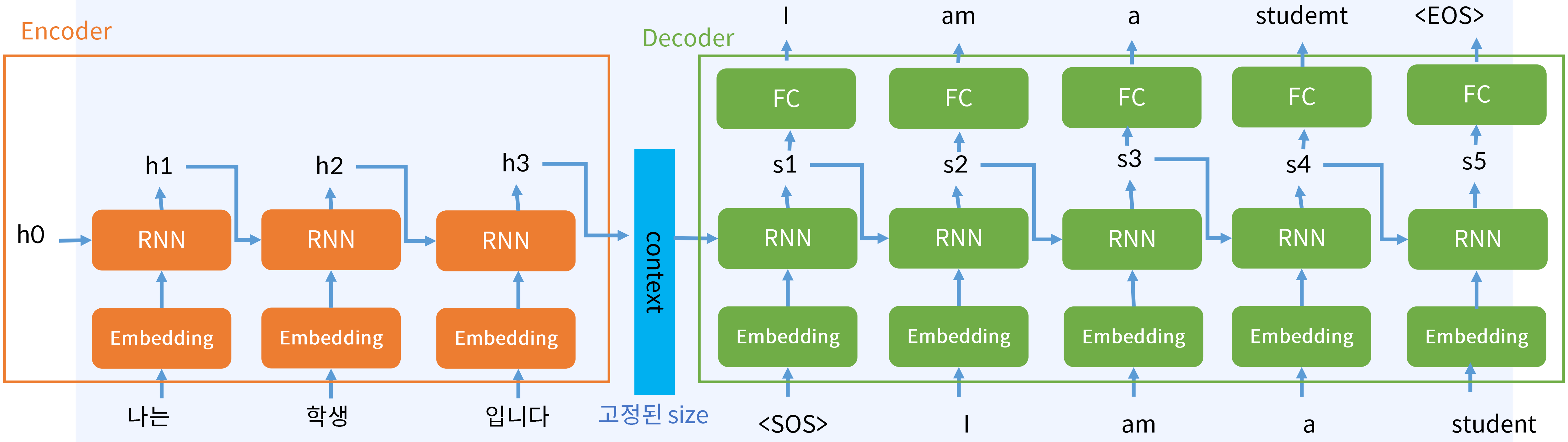
Transformer

- 구글이 제안한 Sequence to Sequence 모델(2017)
- Sequence to Sequence에 특화된 모델



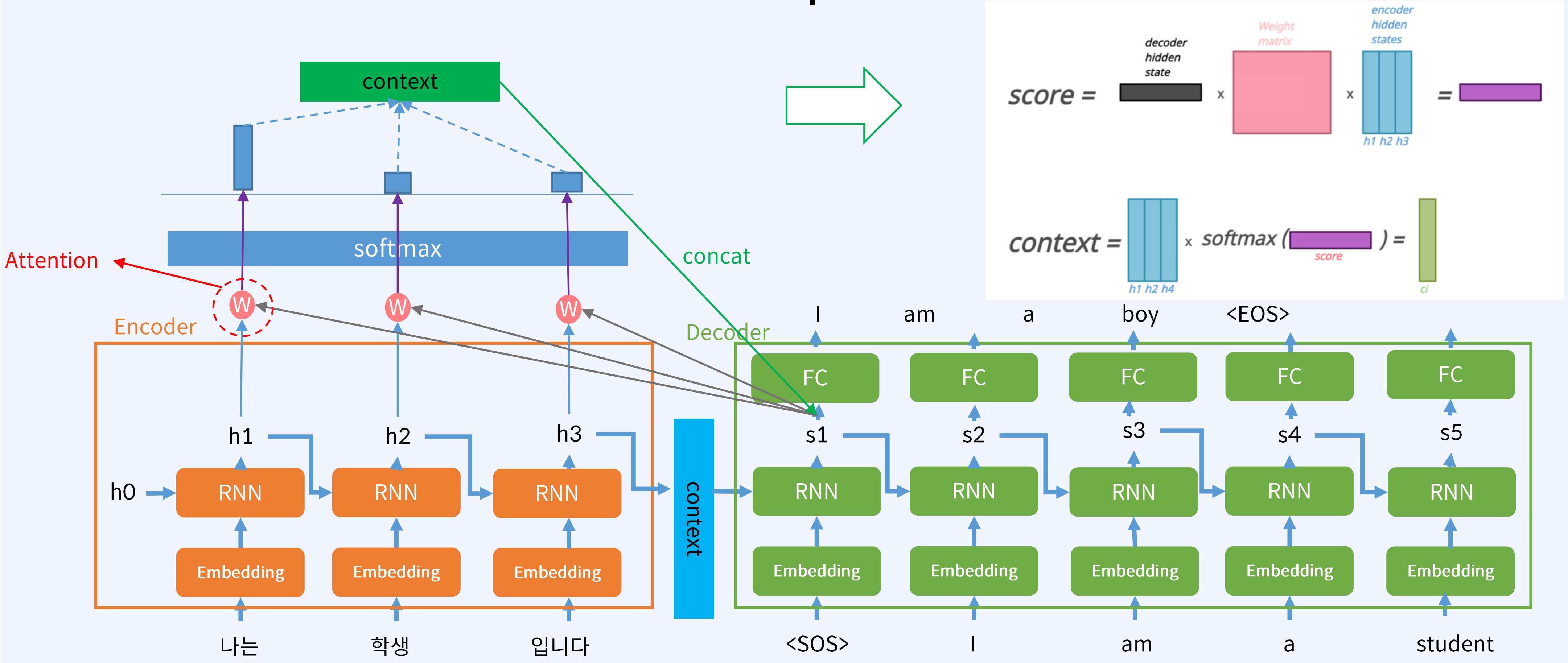
이전 Seq2Seq 모델의 단점

- Context vector에 Sequence 정보를 encoding
- 고정된 context vector의 bottleneck에 의한 성능 저하



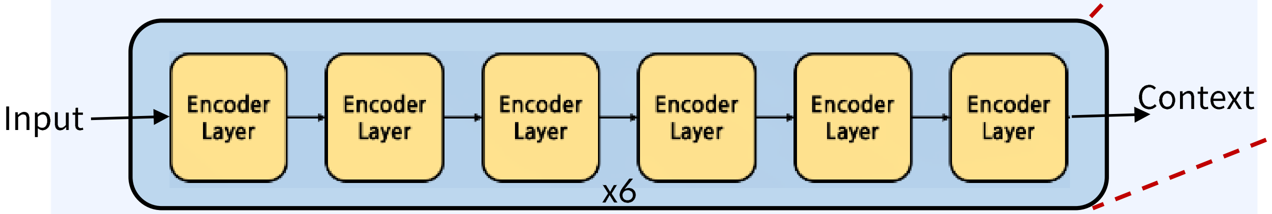
Seq2Seq with Attention

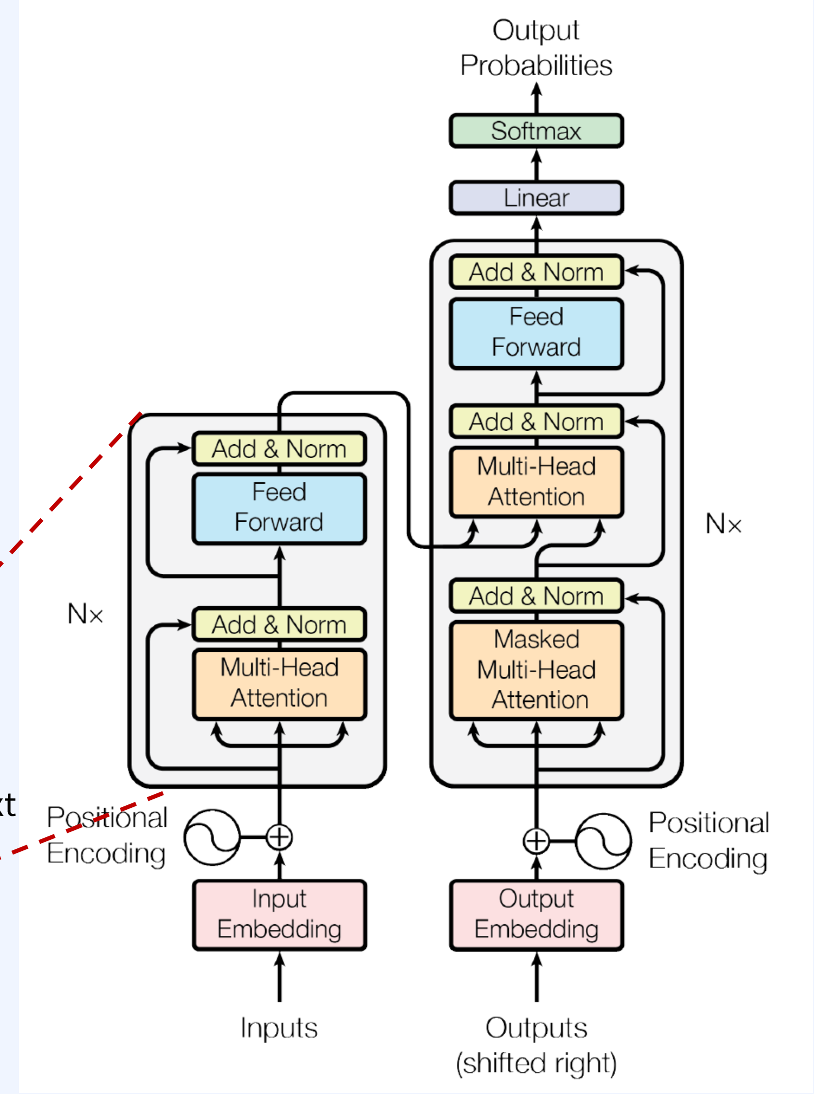
- Seq2seq에 attention을 적용
- Decoder는 Encoder의 모든 output을 참조



Transformer architecture

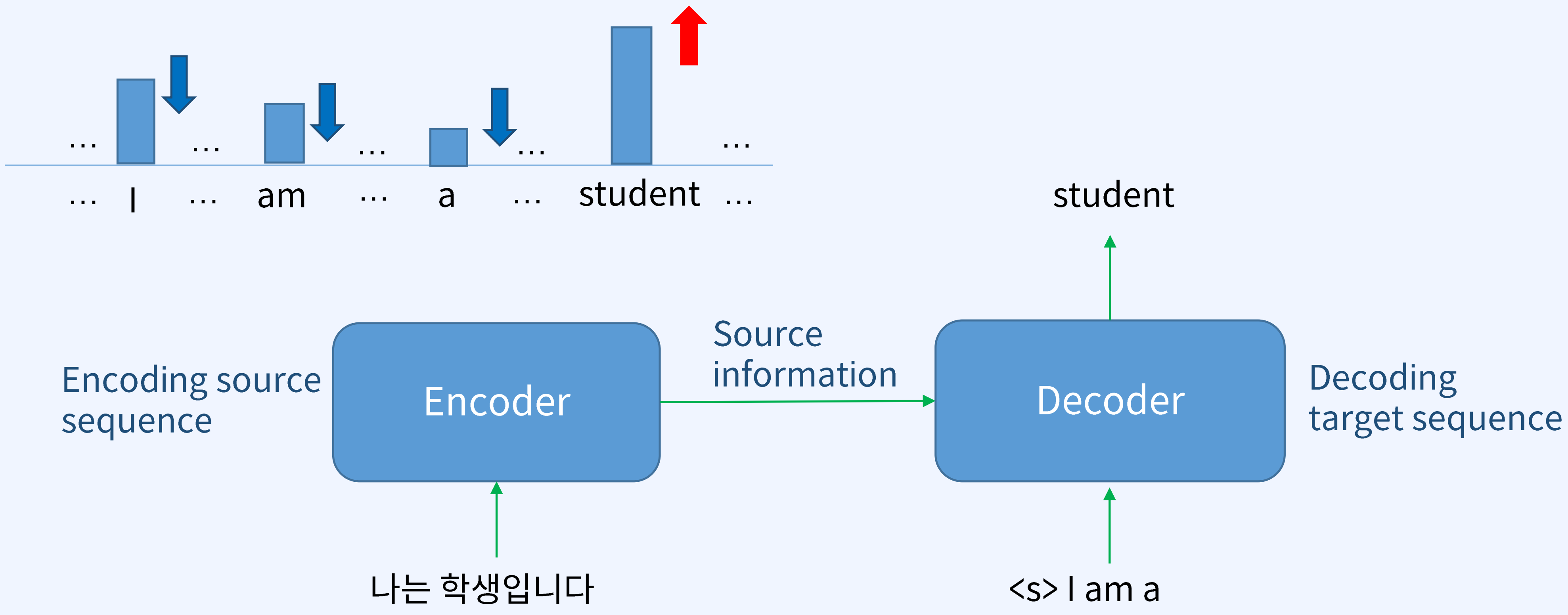
- RNN이나 CNN을 사용 X
- Positional Encoding 사용
- Encoder와 Decoder로 구성
- Attention을 여러 layer에서 반복





Transformer training

• Encode와 Decoder 입력이 주어졌을 때, 정답에 해당하는 단어의 확률값을 높이는 방식으로 수행

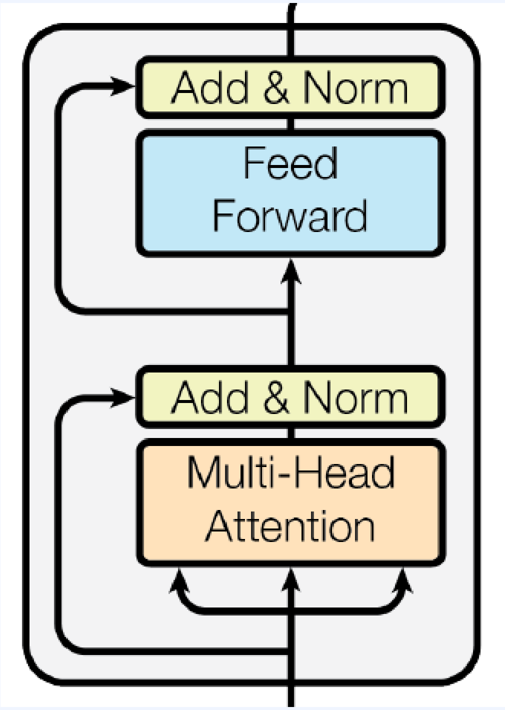


Encoder & Decoder

Transformer block(Layer)

- Encoder Block 구성
  - Multi-head self-attention
  - position-wise feed-forward network
  - Residual connection
  - Layer Normalization

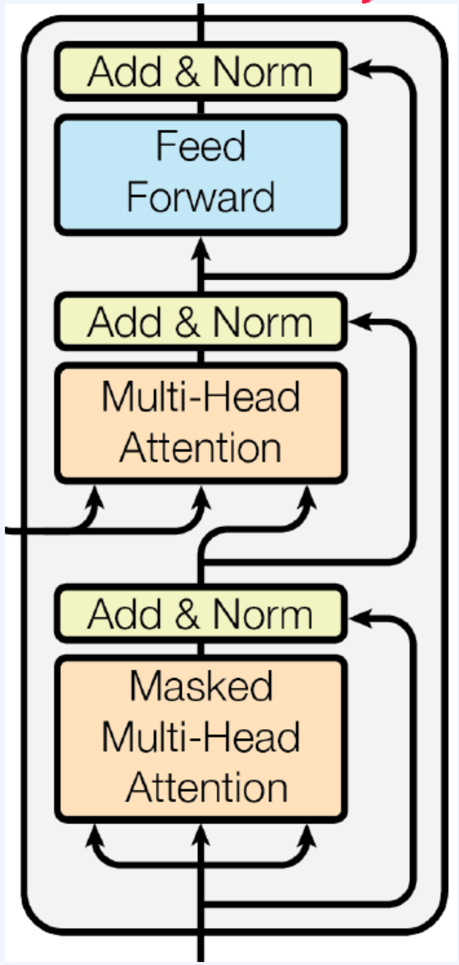
- Each sub-layer output :
  - LayerNorm(x + Sublayer(x))



<Encoder block>

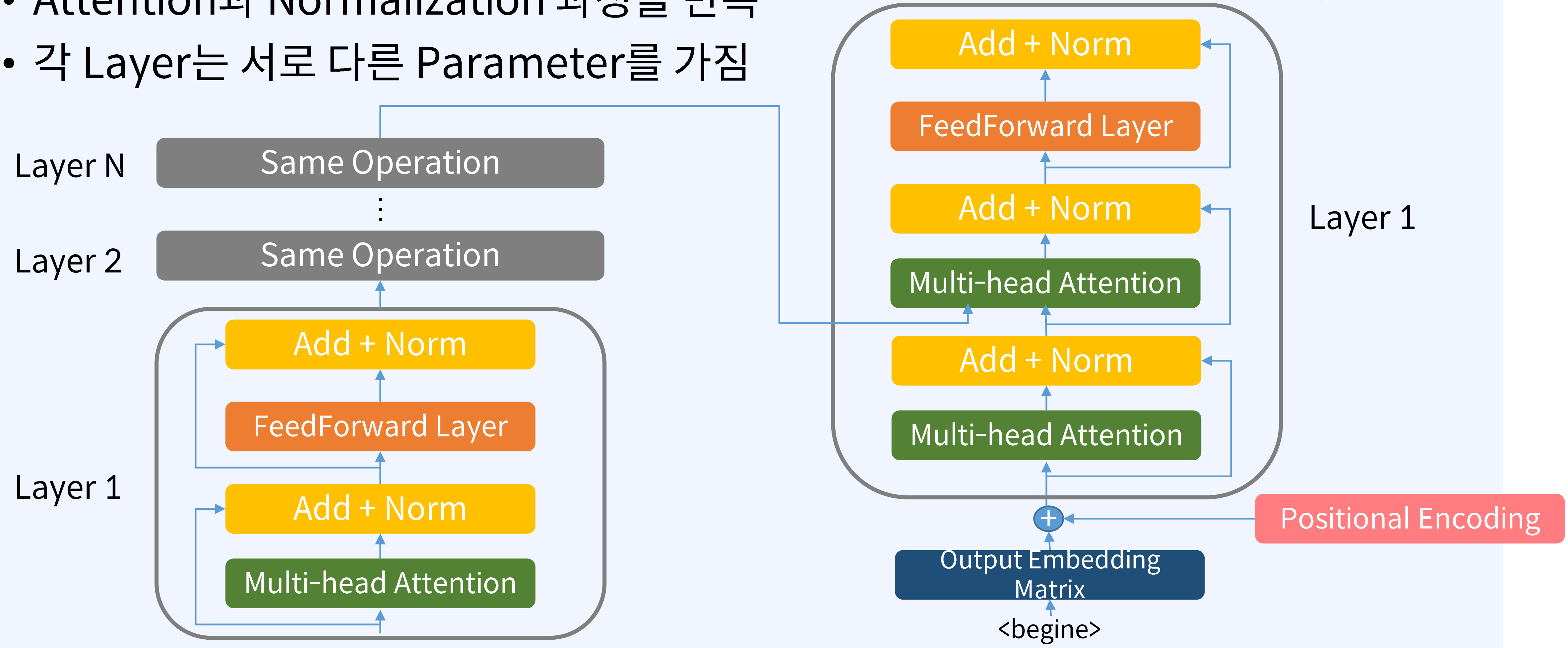
Transformer block

- Decoder Block 구성
  - 기본적인 Encoder Block 구성과 동일
  - Cross-attention 추가
  - Decoding 시에 미래 시점의 단어 정보를 사용하는 것을 방지하기 위해 masked self-attention을 사용



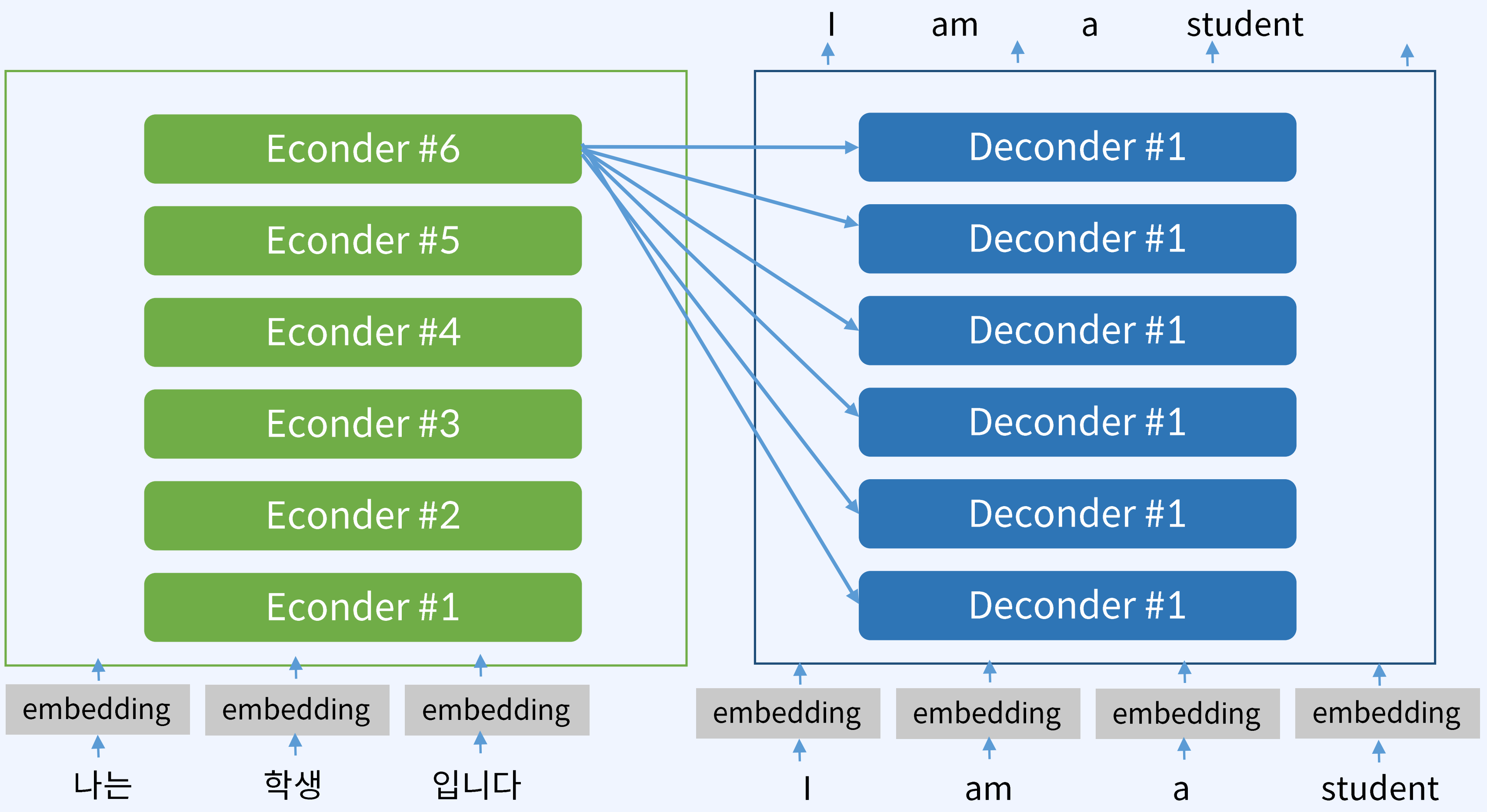
Encoder Decoder Stack과 동작

- Encoder와 Decoder는 N개의 동일한 Block이 Stack된 형태
- Attention과 Normalization 과정을 반복
- 각 Layer는 서로 다른 Parameter를 가짐



Encoder Decoder Stack과 동작

- 마지막 Encoder의 output이 모든 Decoder layer에 input으로 사용
- Encoder Decoder 다수를 사용

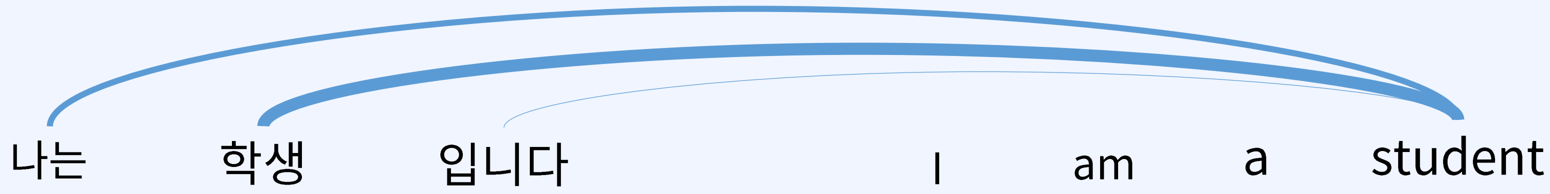


Attention

Attention과 Self-Attention

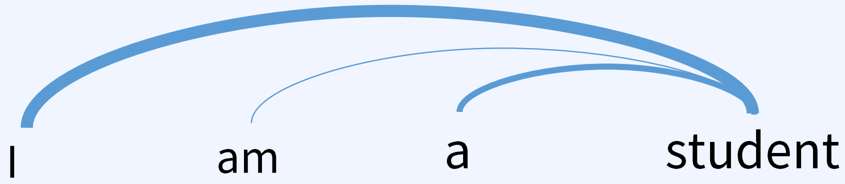
- Attention

< ‘student’의 attention >

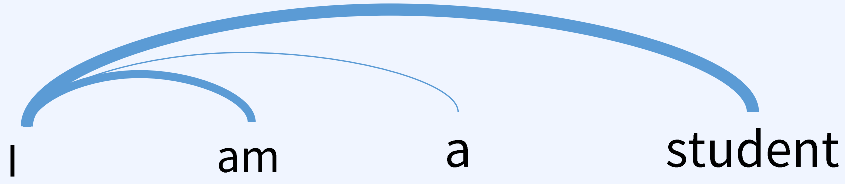


- Self-Attention

< ‘student’의 Self-Attention >

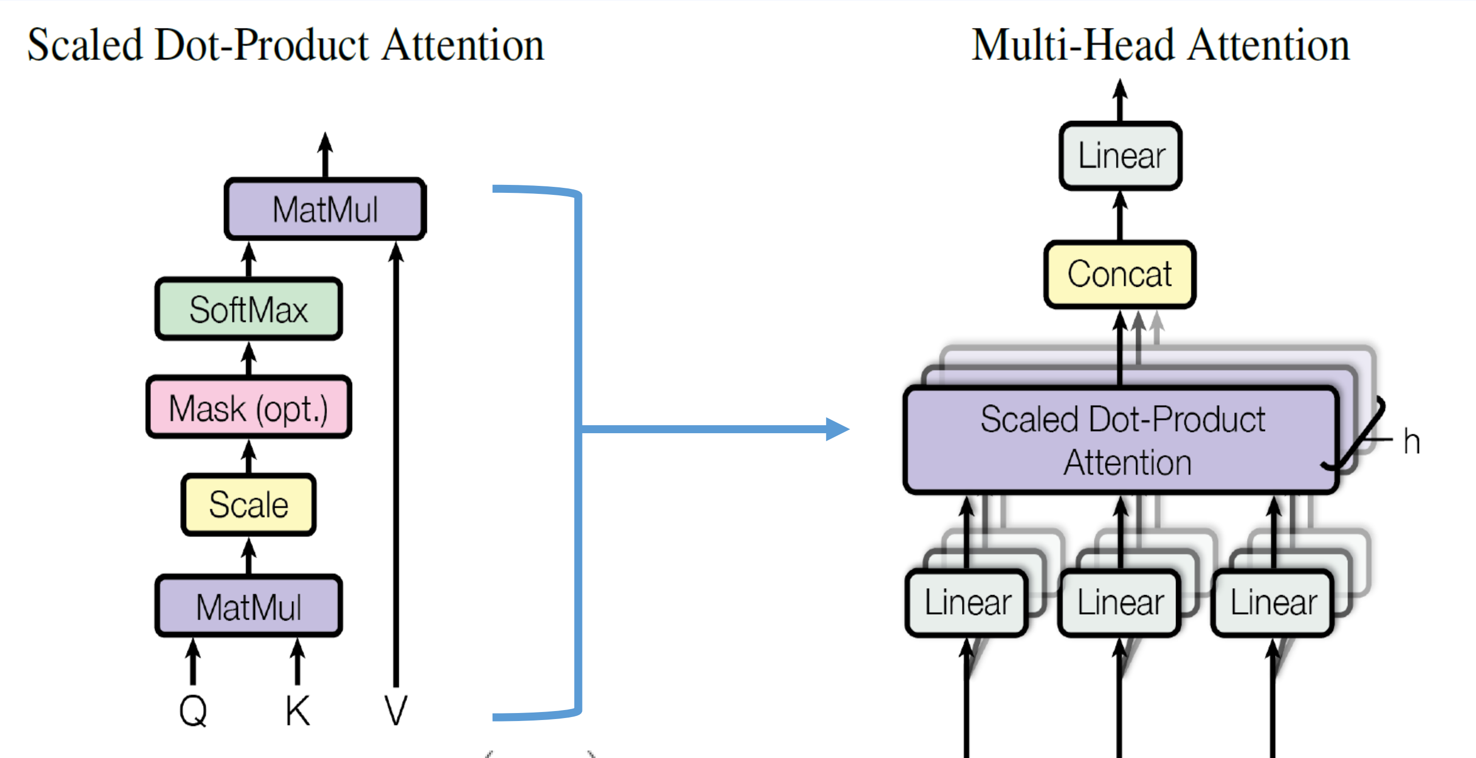


< ‘I’의 Self-Attention >



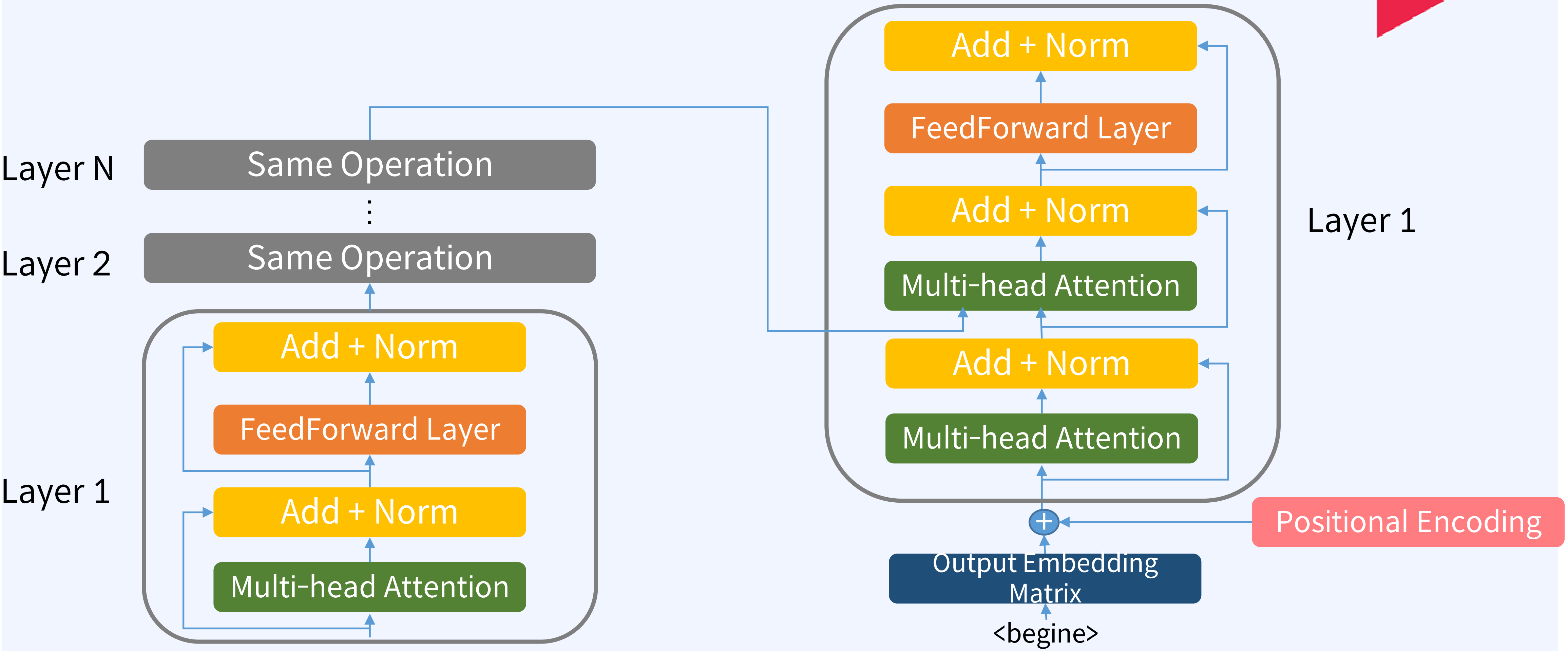
Attention 동작

- Attention에서 사용하는 3가지 요소
  - Query
  - Key
  - Value



$$\mathrm{Attention}(Q,K,V) = \mathrm{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Encoder Decoder 동작

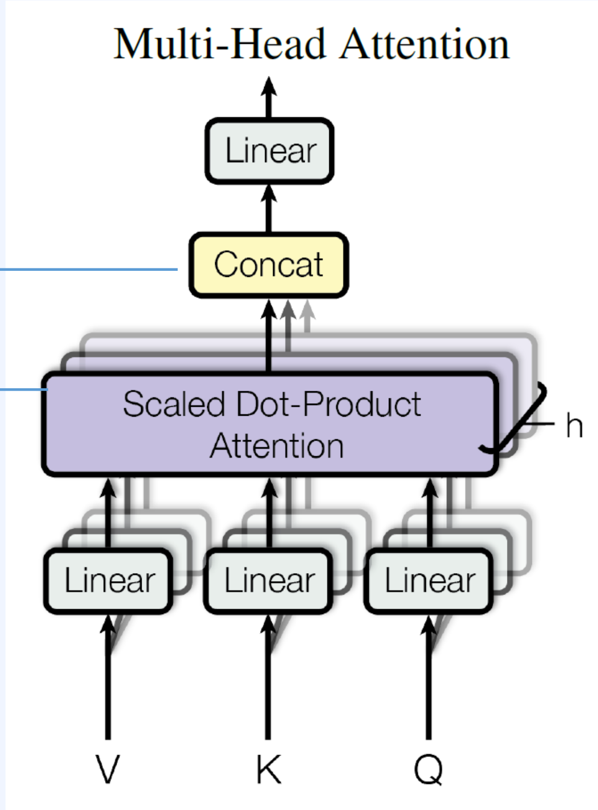


Multi-Head Attention 계산

- Attention function

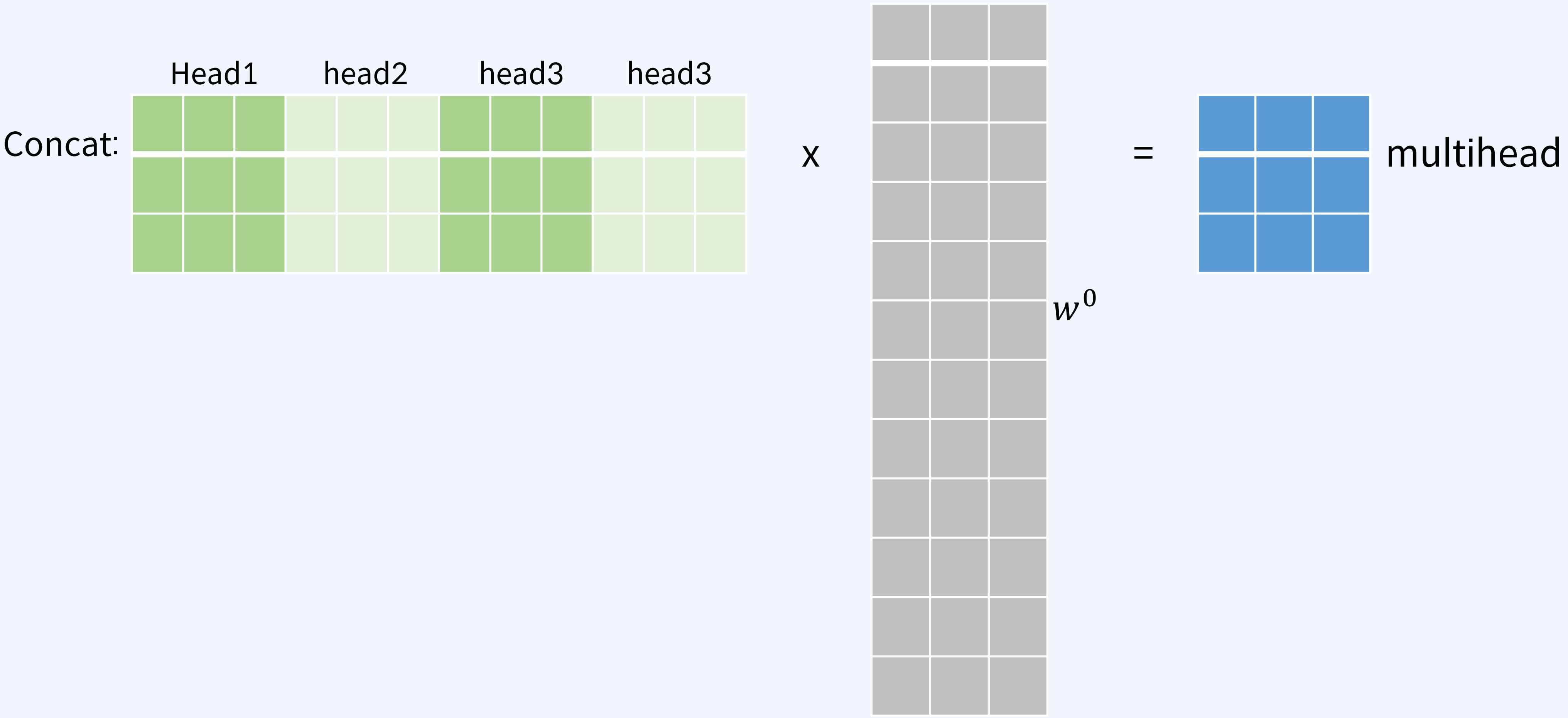
$$\mathrm{Attention}(Q,K,V) = \mathrm{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

$$\mathrm{MultiHead}(Q,K,V) = \mathrm{Concat}(\mathrm{head_1}, \ldots, \mathrm{head_h})W^O$$
$$\text{where } \mathrm{head_i} = \mathrm{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$



Multi-Head

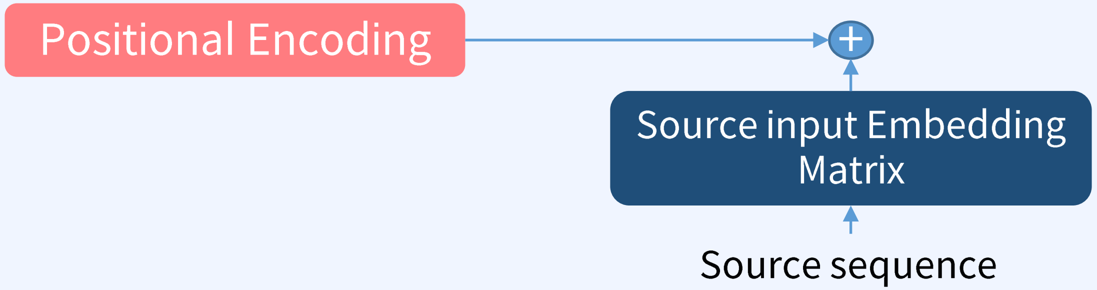
$$\mathrm{MultiHead}(Q, K, V) = \mathrm{Concat}(\mathrm{head_1}, ..., \mathrm{head_h})W^O$$



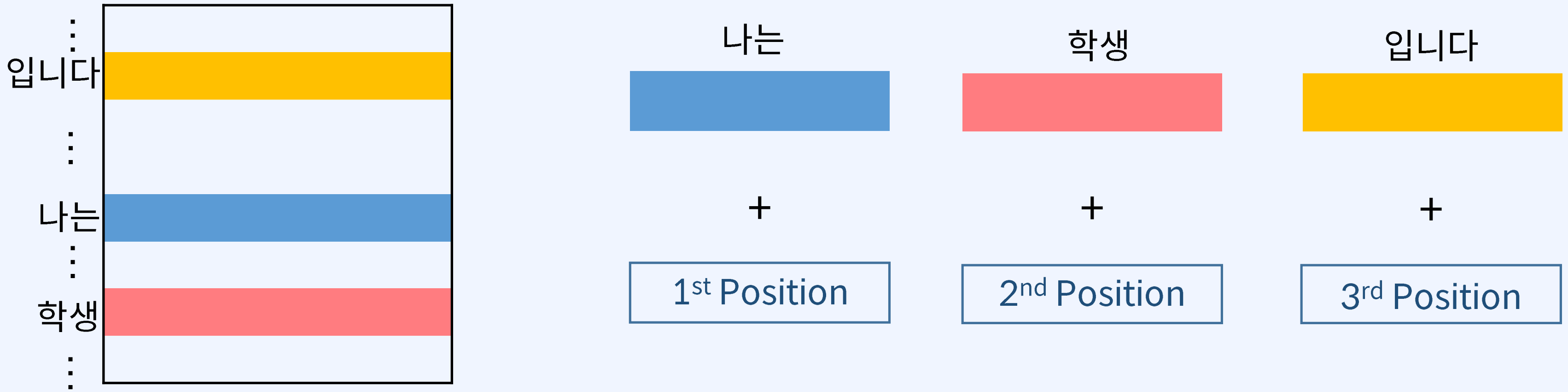
Embedding

Embedding

- Encoder의 input layer



- Encoder의 input 예시



Embedding

- Sequence : 나는 학생 입니다

Embedding

One hot encoding

1	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	0	1

X

Lookup table

0.1.	0.3	3.1	3.1
2.4	3.1	0.1	2.4
0.1	3.1	0.3	2.4
2.2	1.5	1.5	3.1
2.4	0.1	2.4	0.3
3.1	2.6	2.4	0.1
0.3	3.1	3.1	3.1
3.1	2.2	2.4	2.2
2.2	0.3	0.1	2.2
2.2	2.2	0.9	2.4

trainable parameters

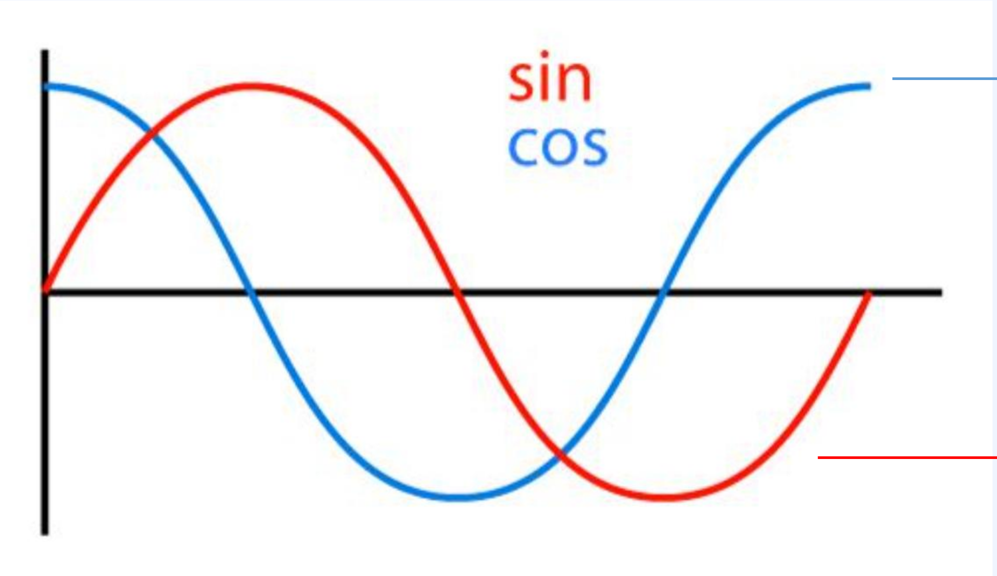
Dense vector

[0.3, 3.1, 3.1, 3.1]

=>학생

Positional embedding

- 주기 함수를 활용
- 각 단어의 상대적 위치를 입력



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

Positional embedding

- 주기함수

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

(dmodel=4, pos=1 , i = 0,1,2,3)

- 예시 (Sequence : 나는 학생 입니다)

d_{model}

나는				
학생	0.3	3.1	3.1	3.1
입니다				

embedding

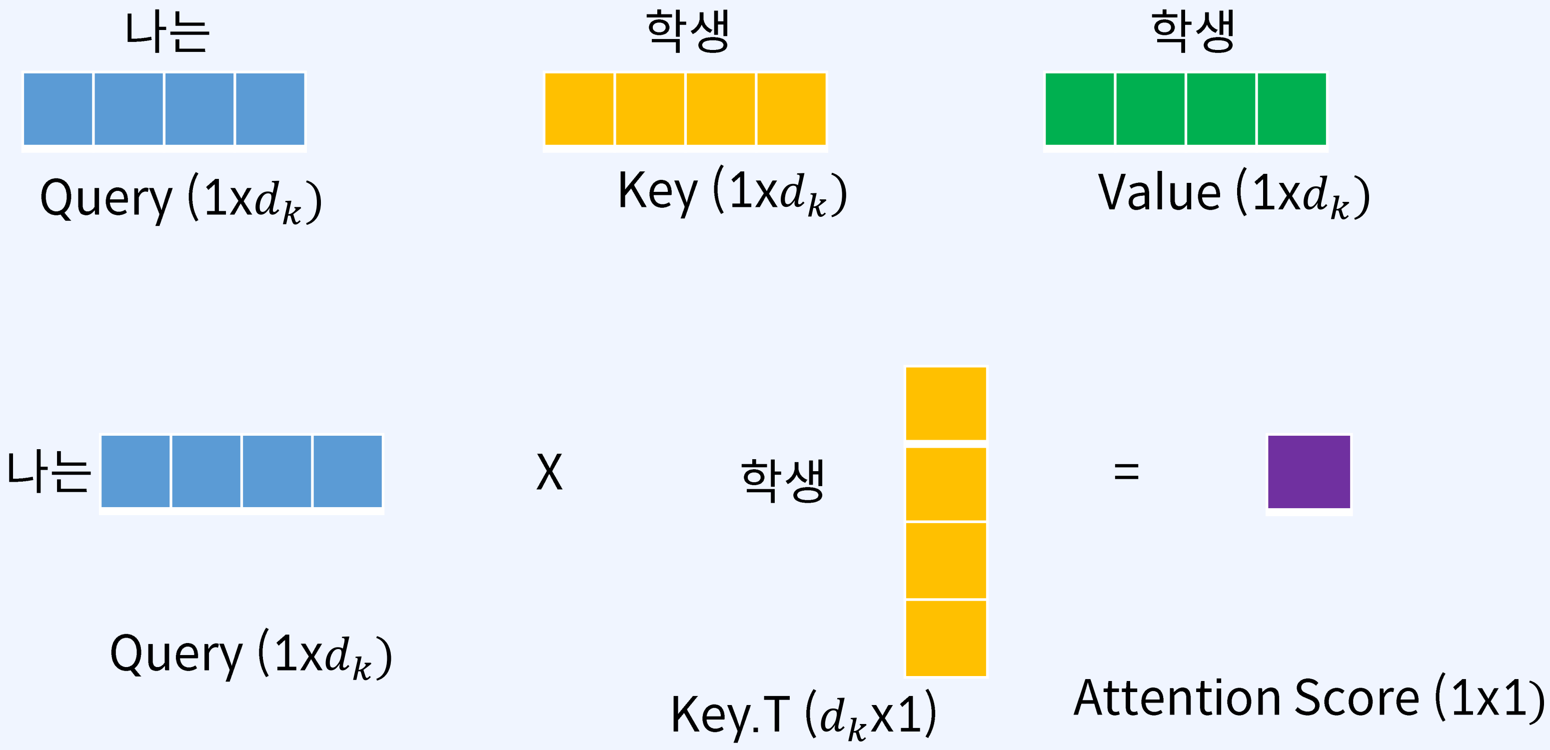
+

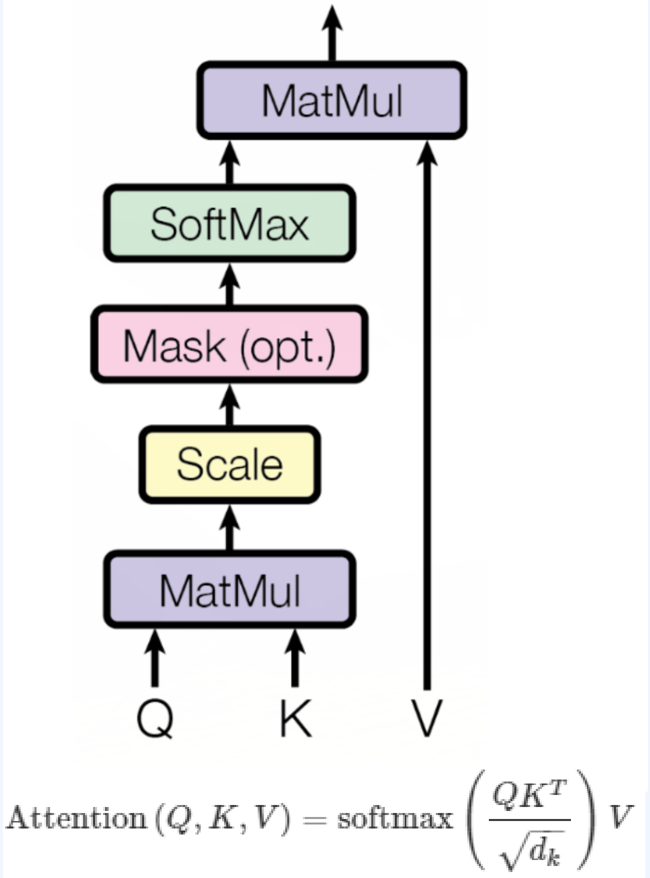
0.25	0.99	0.01	0.99

Positional embedding

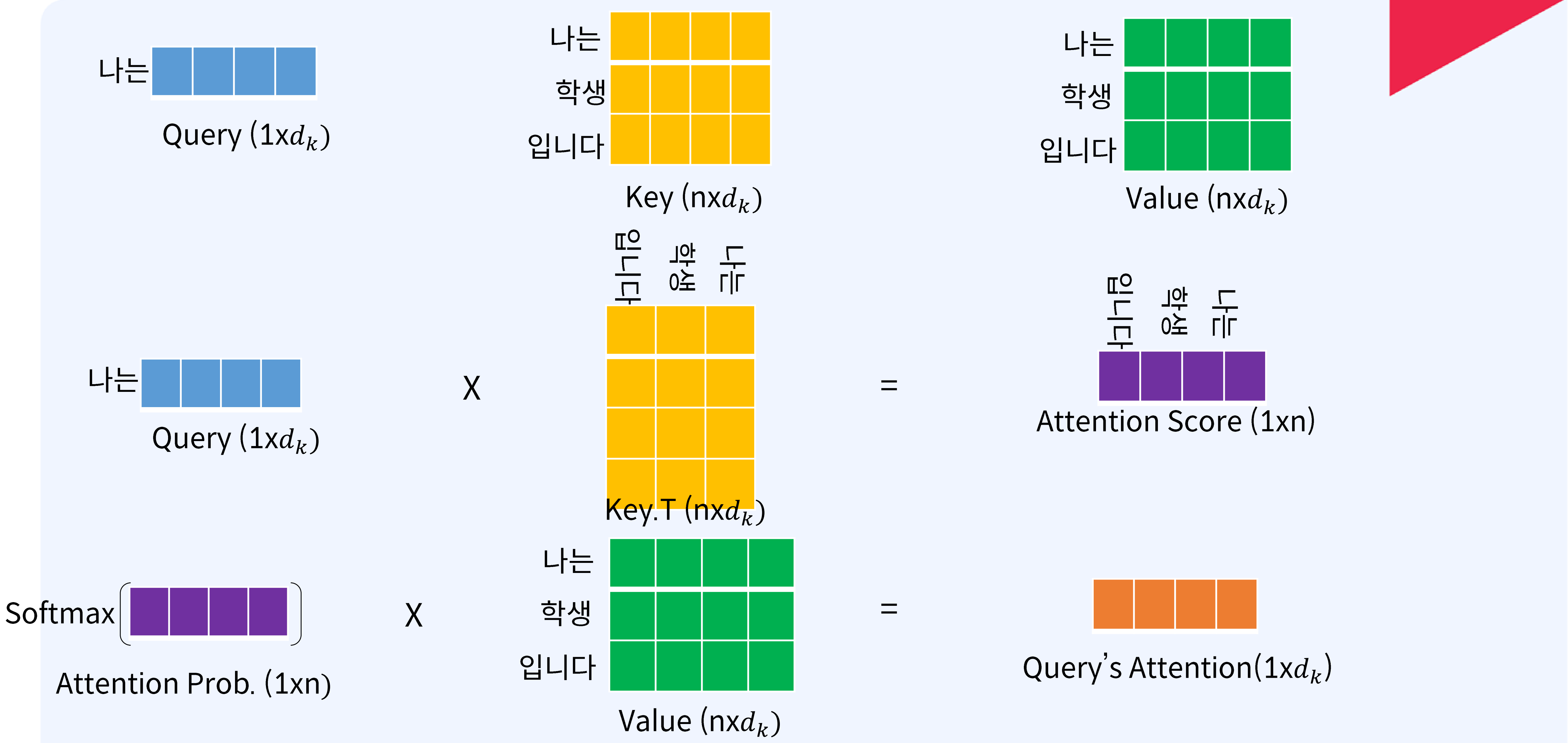
Key,Query,Value
FFN

Query, Key, Value의 Attention 계산

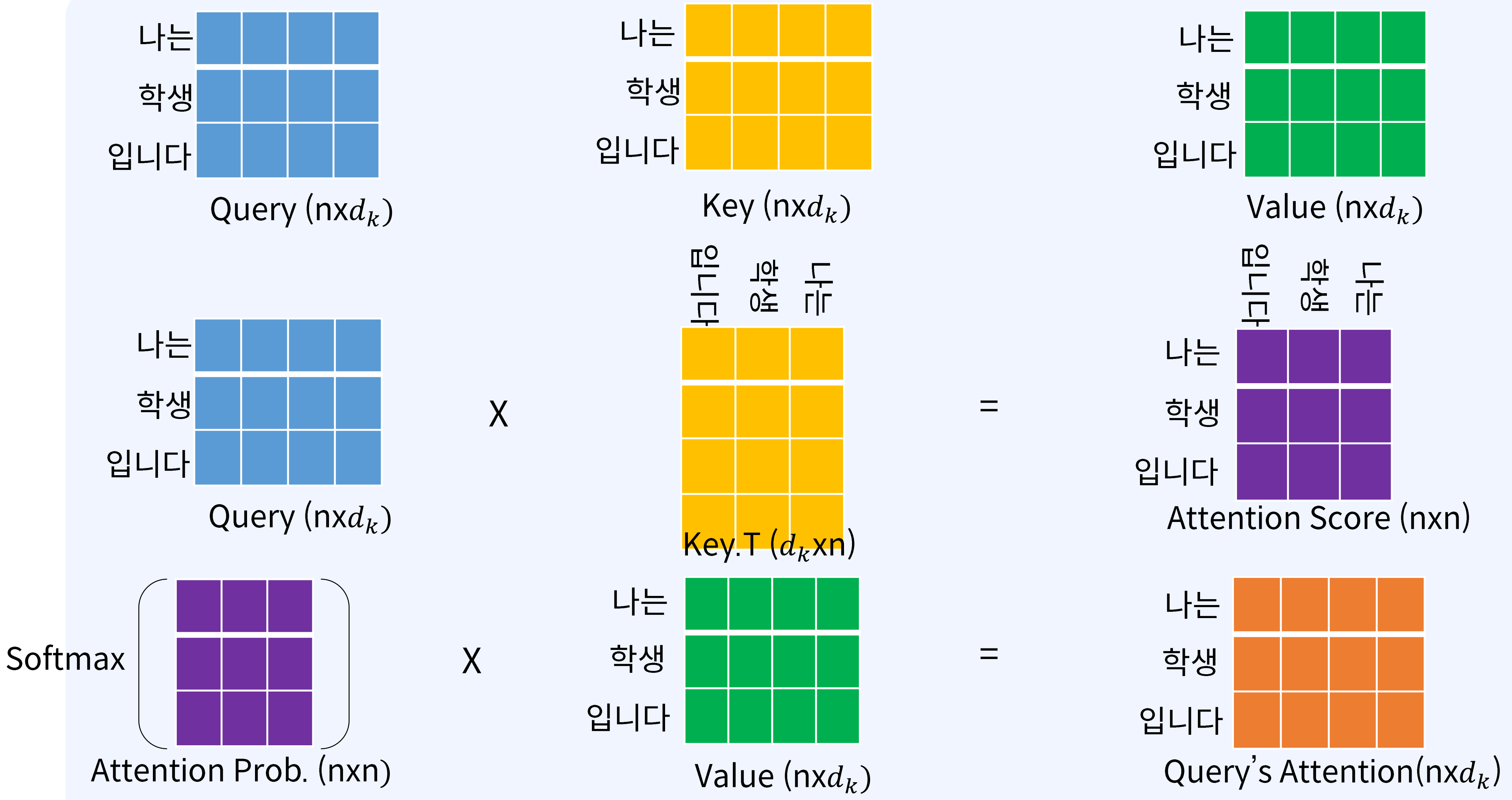




Query, Key, Value의 Attention 계산



Query, Key, Value의 Attention 계산



Query, Key, Value

- 단어 임베딩 차원수 (d) = 4
- 인코딩에 입력된 단어 개수 = 3

$$X = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 2 & 0 & 2 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

- Query, Key, Value 생성
  - $Q = X \times W_Q$
  - $K = X \times W_k$
  - $V = X \times W_v$

< Query 생성 예시 >

$$\overset{X}{\begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 2 & 0 & 2 \\ 1 & 1 & 1 & 1 \end{bmatrix}} \times \overset{W_Q}{\begin{bmatrix} 1 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}} = \begin{array}{cc} Q & \\ \begin{bmatrix} 1 & 0 & 2 \\ 2 & 2 & 2 \\ 2 & 1 & 3 \end{bmatrix} & \begin{array}{c} Q_1 \\ Q_2 \\ Q_3 \end{array} \end{array}$$

< Key 생성 예시 >

$$\overset{X}{\begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 2 & 0 & 2 \\ 1 & 1 & 1 & 1 \end{bmatrix}} \times \overset{W_K}{\begin{bmatrix} 0 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 0 \end{bmatrix}} = \begin{array}{cc} K & \\ \begin{bmatrix} 0 & 1 & 1 \\ 4 & 4 & 0 \\ 1 & 3 & 1 \end{bmatrix} & \begin{array}{c} K_1 \\ K_2 \\ K_3 \end{array} \end{array}$$

< Value 생성 예시 >

$$\overset{X}{\begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 2 & 0 & 2 \\ 1 & 1 & 1 & 1 \end{bmatrix}} \times \overset{W_V}{\begin{bmatrix} 0 & 2 & 0 \\ 0 & 3 & 0 \\ 1 & 0 & 3 \\ 1 & 1 & 0 \end{bmatrix}} = \begin{array}{cc} V & \\ \begin{bmatrix} 1 & 2 & 3 \\ 2 & 8 & 0 \\ 2 & 6 & 3 \end{bmatrix} & \begin{array}{c} V_1 \\ V_2 \\ V_3 \end{array} \end{array}$$

Masking (scaled dot-product Attention)

- Mask Matrix을 이용해 특정 단어를 무시
- Mask 값으로 음수의 무한 값을 넣어 softmax 함수의 출력이 0%에 가까워지게 함

$QK^T =$

	I	am	a	student
I				
am				
a				
student				

Attention Score

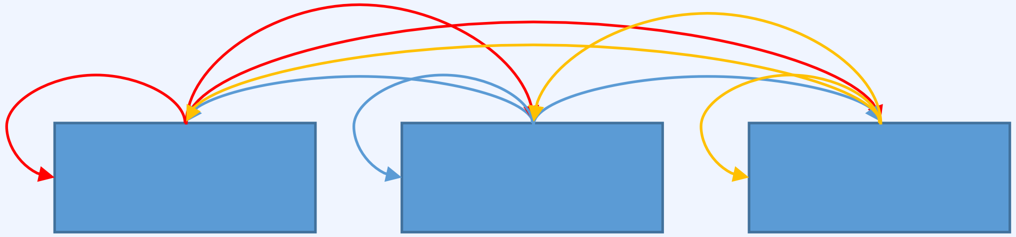
$\circledast$

	I	am	a	student
I				
am				
a				
student				

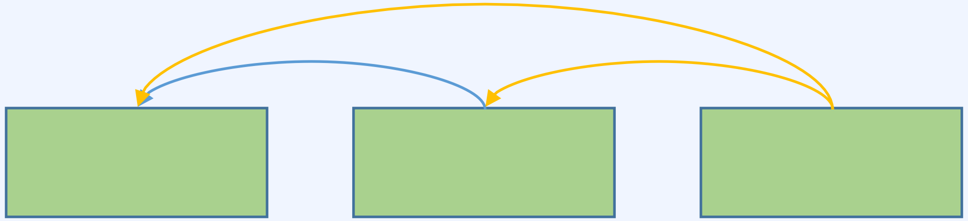
Mask Matrix

3가지 Attention layer

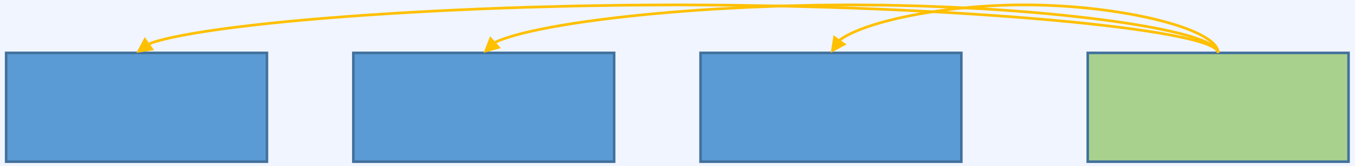
- Encoder Self-Attention



- Masked Decoder Self-Attention

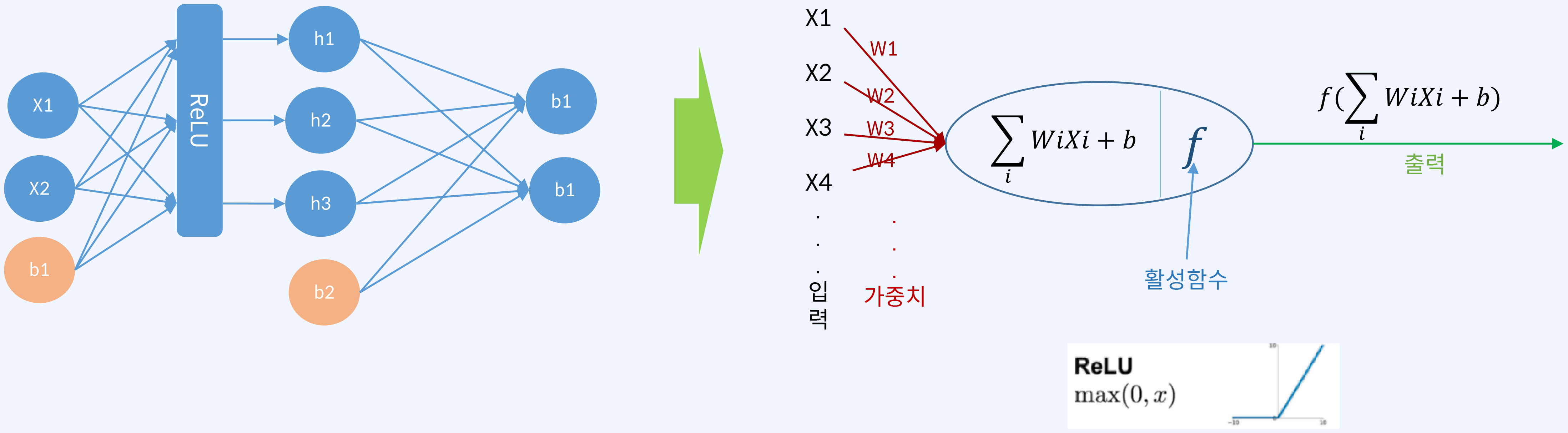


- Encoder-Decoder Attention



Position-wise Feed-forward network

$$\mathrm{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2$$



Block의 기타 구성 요소

- Residual connection

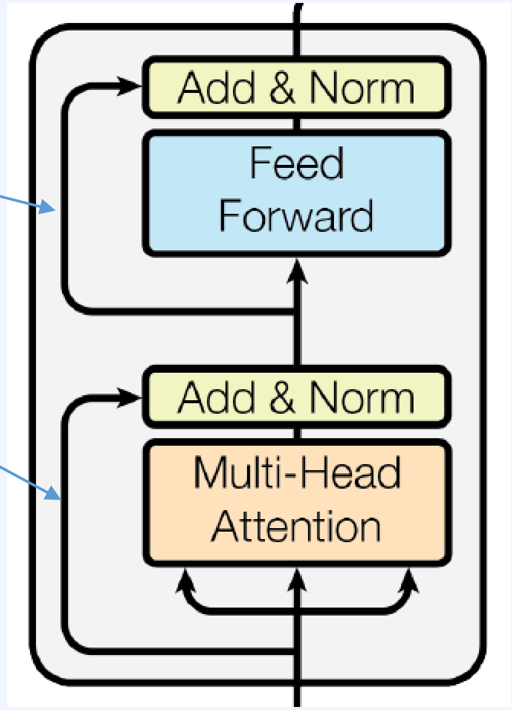
- Layer normalization

$$y = \frac{x - E[x]}{\sqrt{\forall[x] + \epsilon}} \gamma + \beta$$

미니 배치

1	2	3
1	2	3

평균	표준편차
2	0.8164
1	0



Summary

- Encoder/Decoder
- Multi-Head Attention
- Embedding
- FFN