투빅스 텍스트 세미나

ToBig's 13기 정주원

CS224n Lecture 14

Transformer and Self-Attention

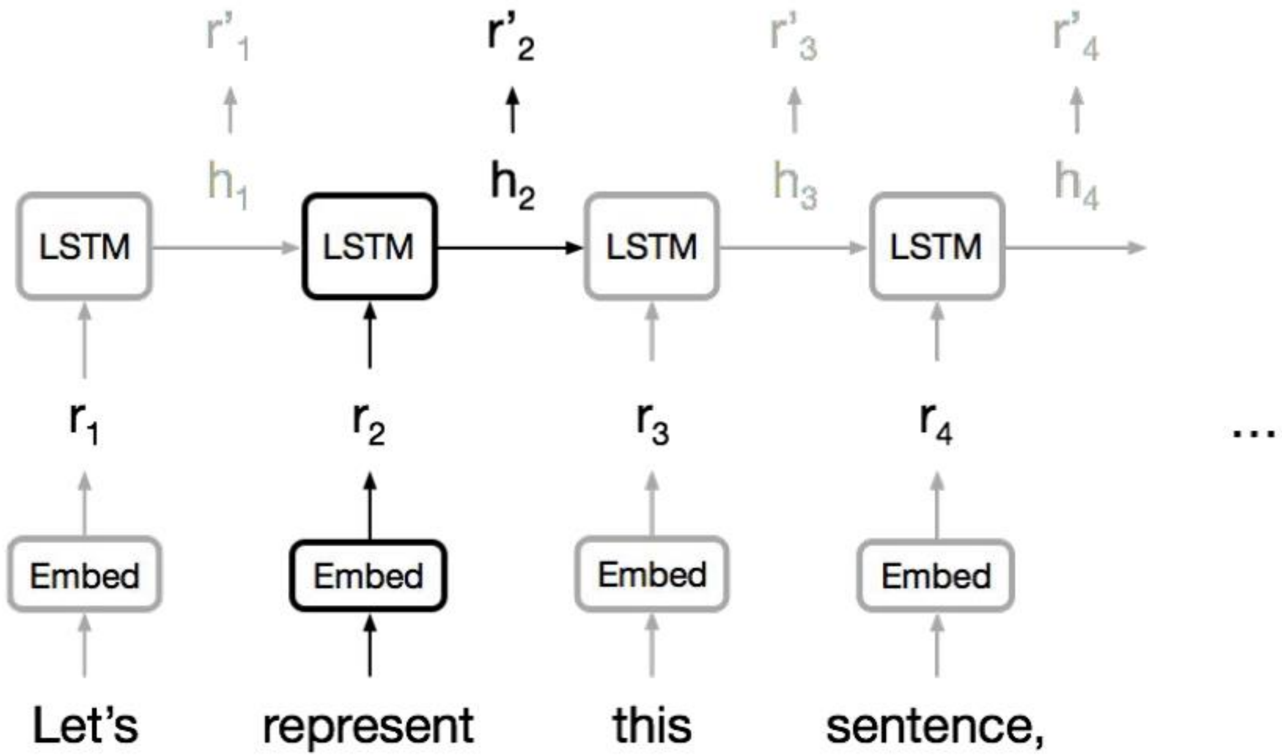
Contents

Unit 01 | RNN, CNN, and Self-Attention

Sequence Modeling에서

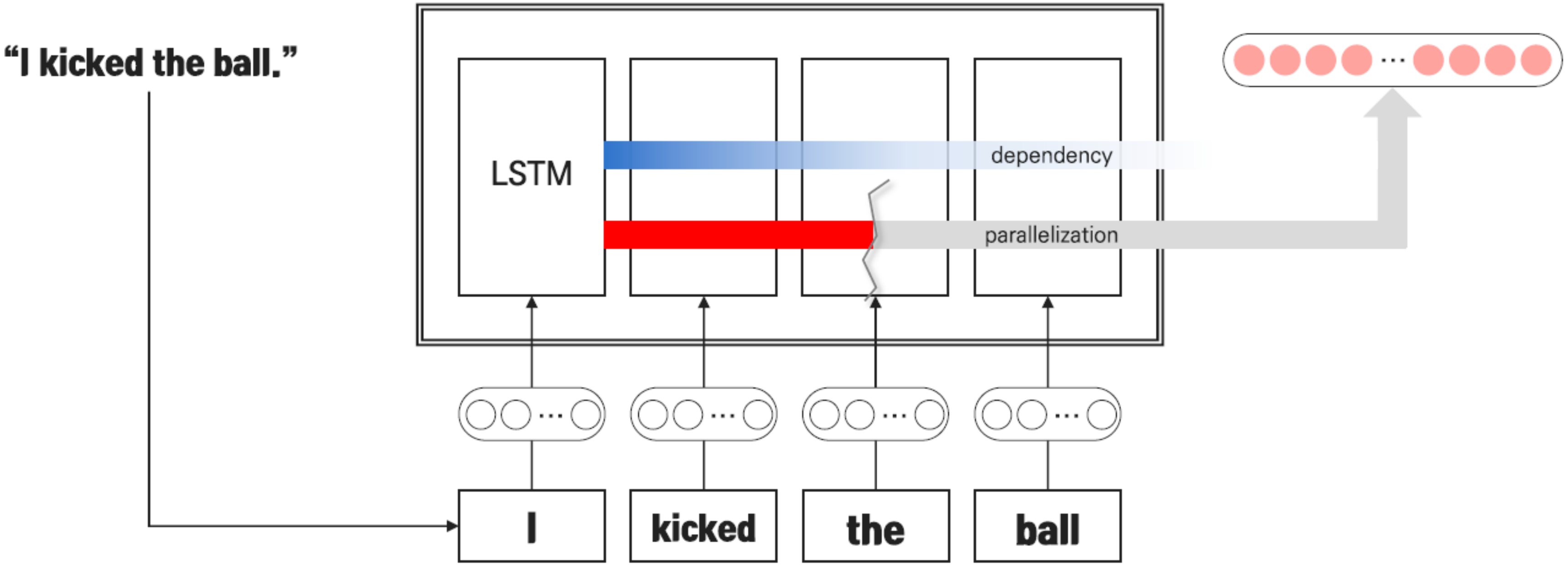
가변 길이의 데이터를 **고정 크기**의 데이터로 표현하는 것은 필수적

Unit 01 | RNN, CNN, and Self-Attention



왜 Self-Attention인가?

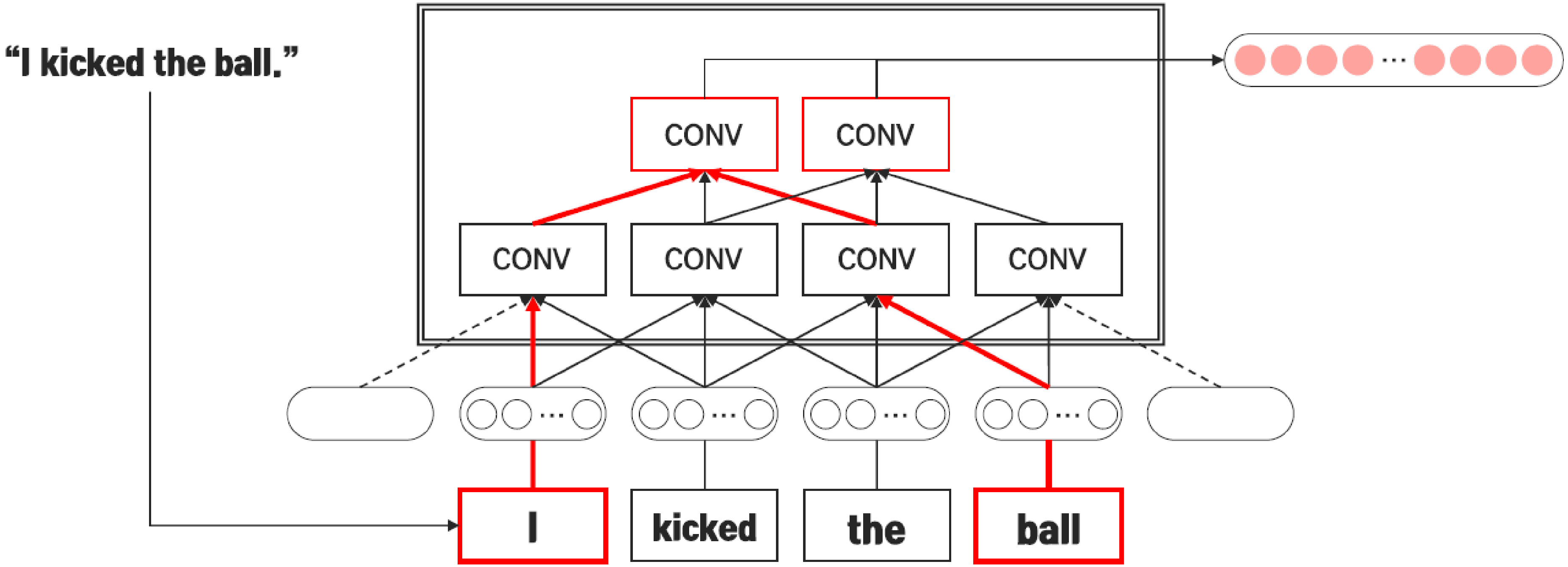
Unit 01 | RNN, CNN, and Self-Attention



RNN 계열의 모델은

parallelization이 불가능하고 long-term dependency를 잘 반영하지 못함

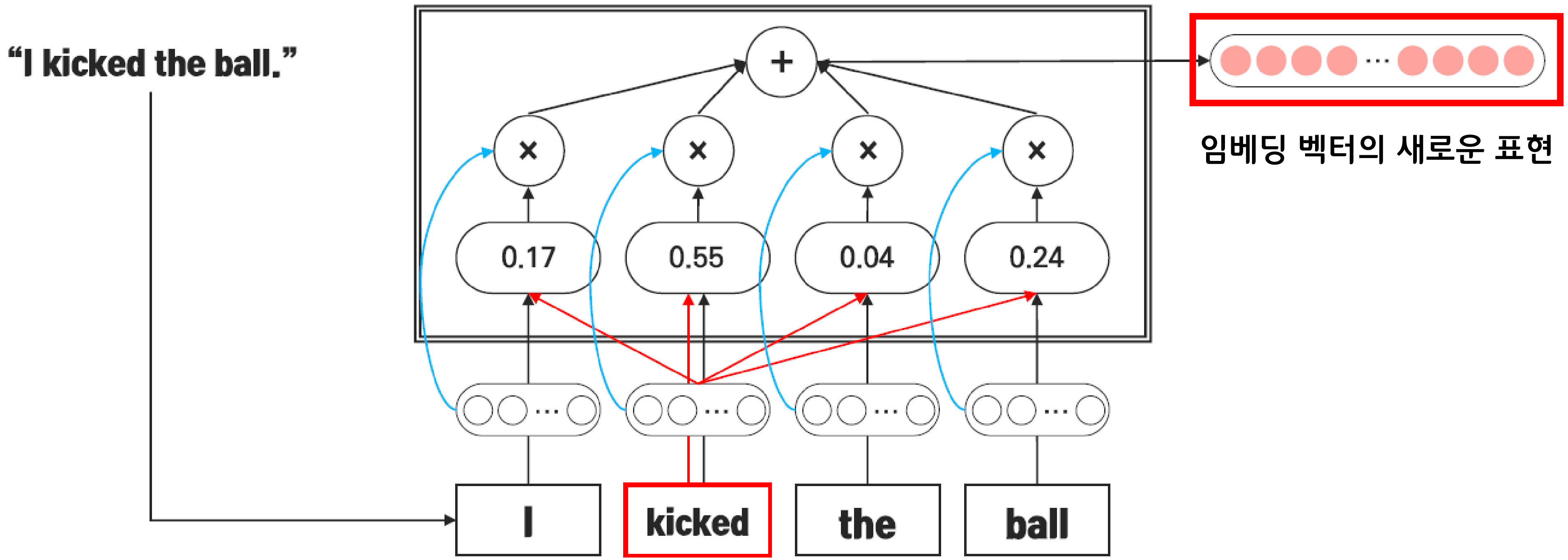
Unit 01 | RNN, CNN, and Self-Attention



CNN 계열의 모델은

parallelization은 가능하지만 long-term dependency를 잘 반영하지 못함

Unit 01 | RNN, CNN, and Self-Attention



Self-attention은

병렬화가 가능함과 동시에 long-term dependency 문제를 해결함

Unit 01 | RNN, CNN, and Self-Attention

Layer Type	Complexity per Layer	Sequential Operations	Maximum Path Length
Self-Attention	$O(n^2 \cdot d)$	$O(1)$	$O(1)$
Recurrent	$O(n \cdot d^2)$	$O(n)$	$O(n)$
Convolutional	$O(k \cdot n \cdot d^2)$	$O(1)$	$O(\log_k(n))$

Complexity per Layer → Sequence length(n)가 model dimension(d)보다 작은 일반적인 경우 연산량이 가장 적음

Sequential Operations → 병렬화 가능

Maximum Path Length → Long-term dependency 문제 해결

Unit 02 | Transformer

Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com

Noam Shazeer* Google Brain noam@google.com

Niki Parmar* Google Research nikip@google.com

Jakob Uszkoreit* Google Research usz@google.com

Llion Jones* Google Research llion@google.com

Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu

Łukasz Kaiser* Google Brain lukaszkaiser@google.com

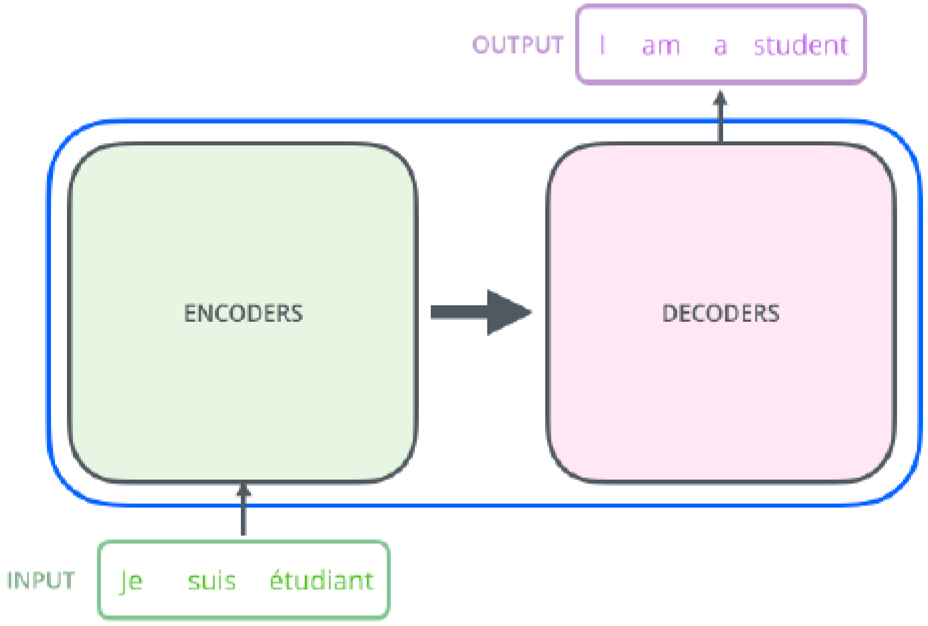
Illia Polosukhin* ‡ illia.polosukhin@gmail.com

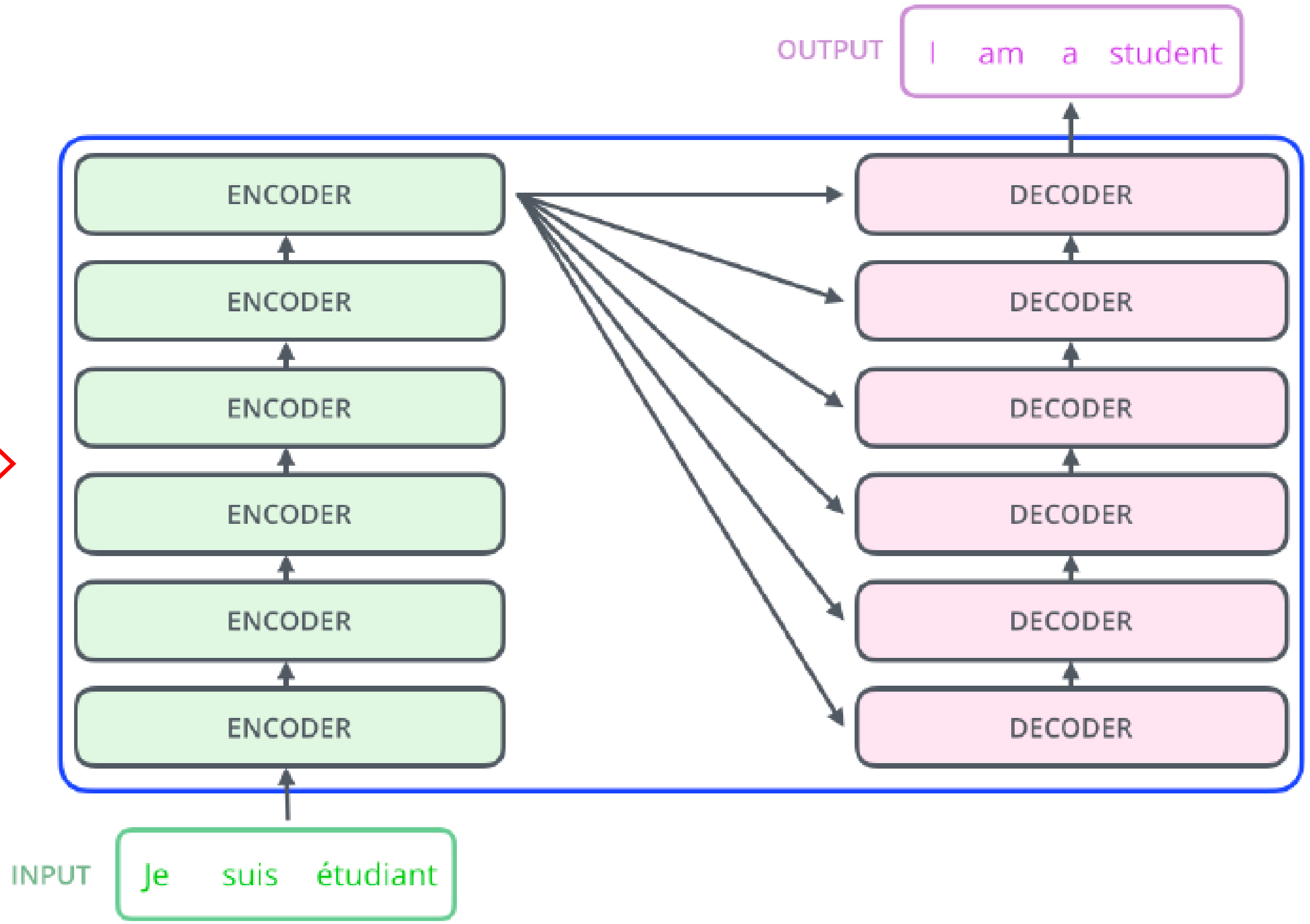
Transformer

sequential token을 sequential하게 처리하지 않음

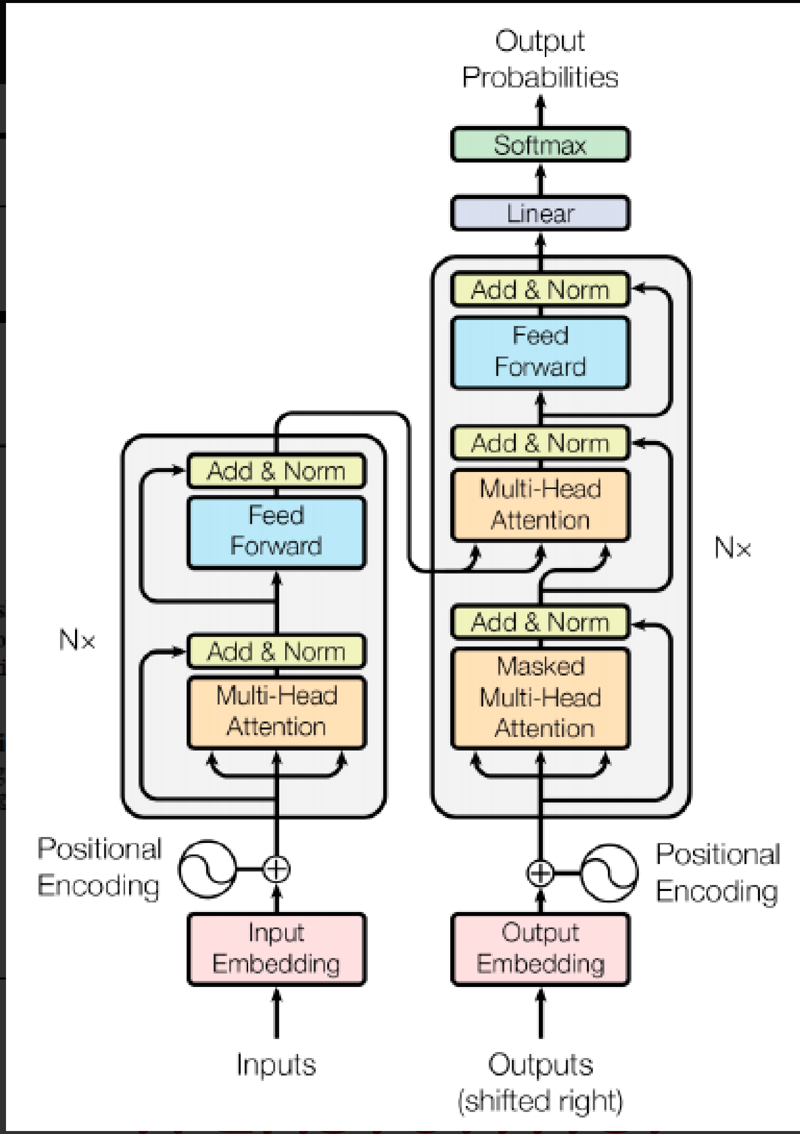
parallelization with self-attention

Unit 02 | Transformer





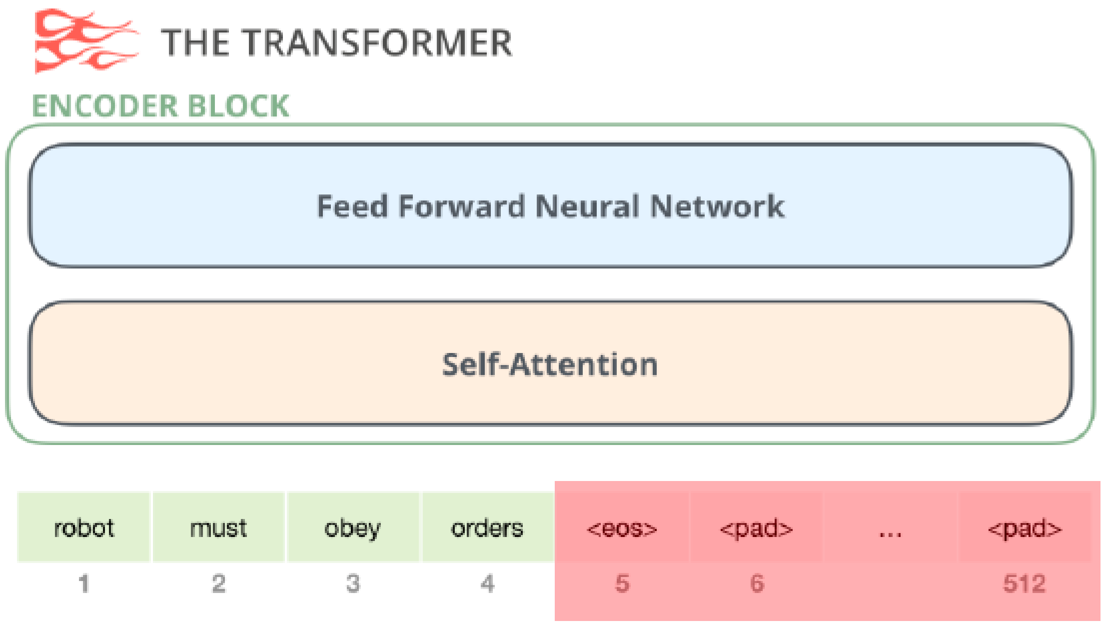
Unit 02 |

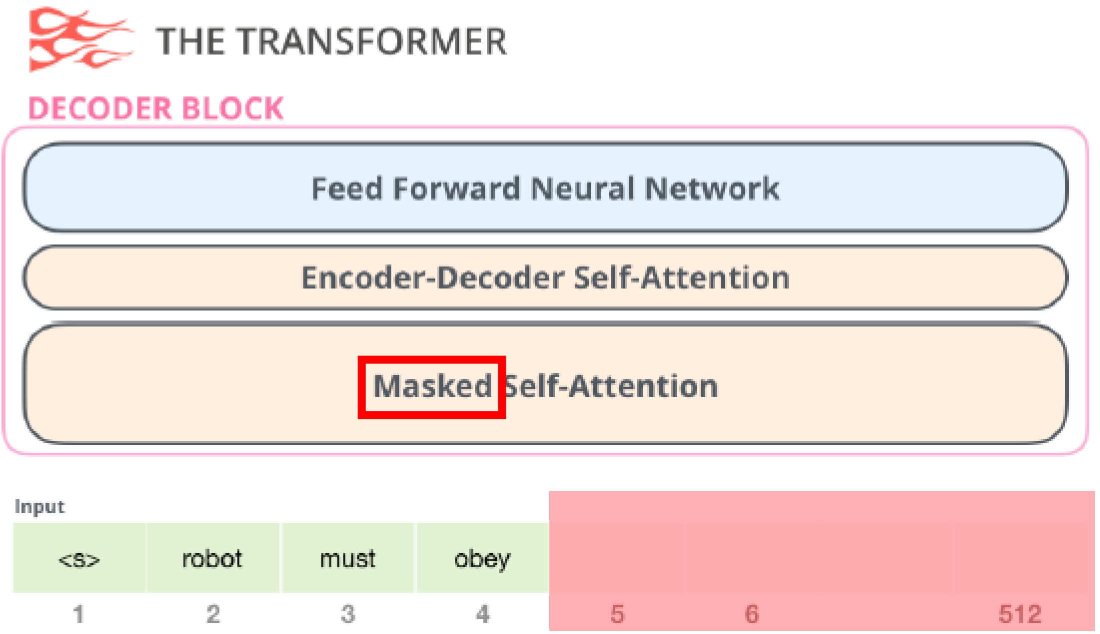


Ashis
Goo
avaswani
Uszkoreit*
Research
ogle.com
Lli
Goog
llion@
le.com

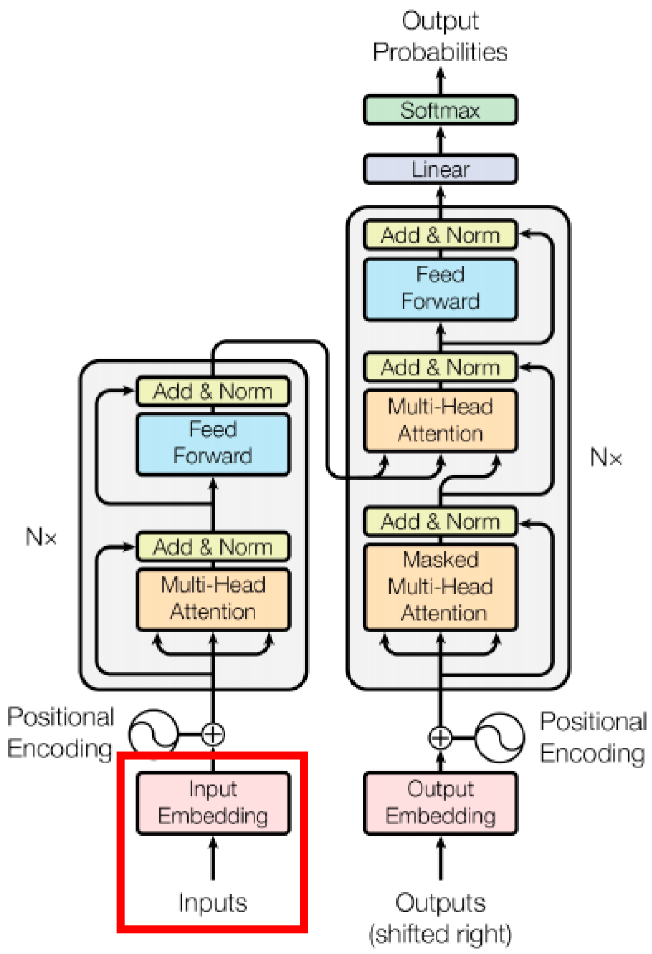
] 6 Dec 2017

Encoding block vs. Decoding block





Unit 02 | Transformer



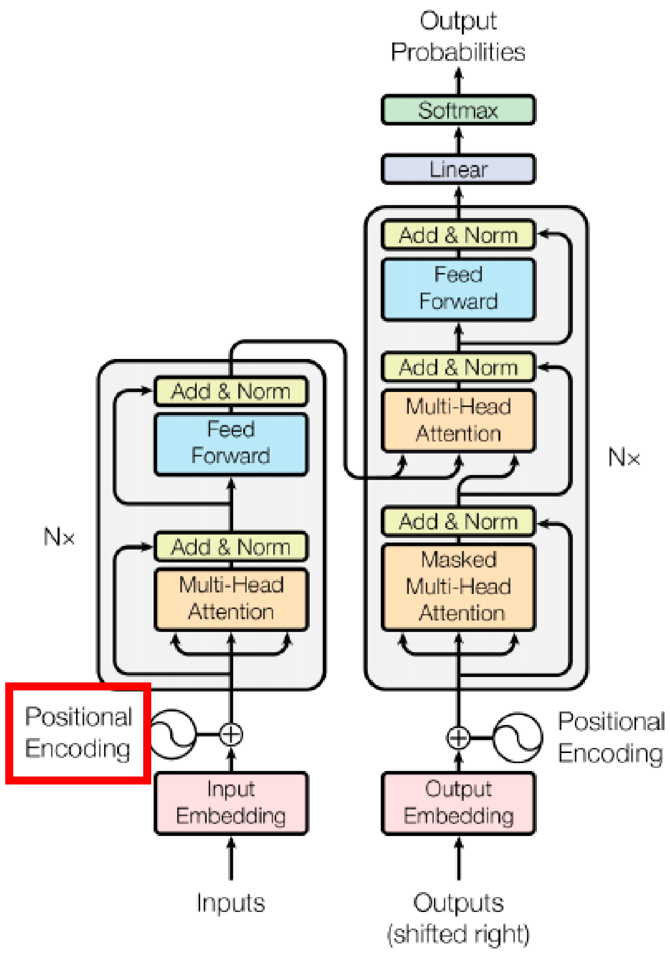
1. Input Embedding

512 dimensions

- 임베딩 알고리즘 (W2V, Glove, FastText) : 단어를 $\boxed{\text{벡터}}$로 변환
- 첫 번째 인코더의 입력으로만 사용 → 이후 인코더의 입력은?
- $\boxed{\text{sequence}}$의 길이: 가장 긴 문장의 단어 수 or 상위 95%에 해당하는 토큰의 개수

attention에서 한 번에 볼 길이

Unit 02 | Transformer



2. Positional Encoding

- 한 번에 모든 sequence를 적용해 생기는 단점 보완

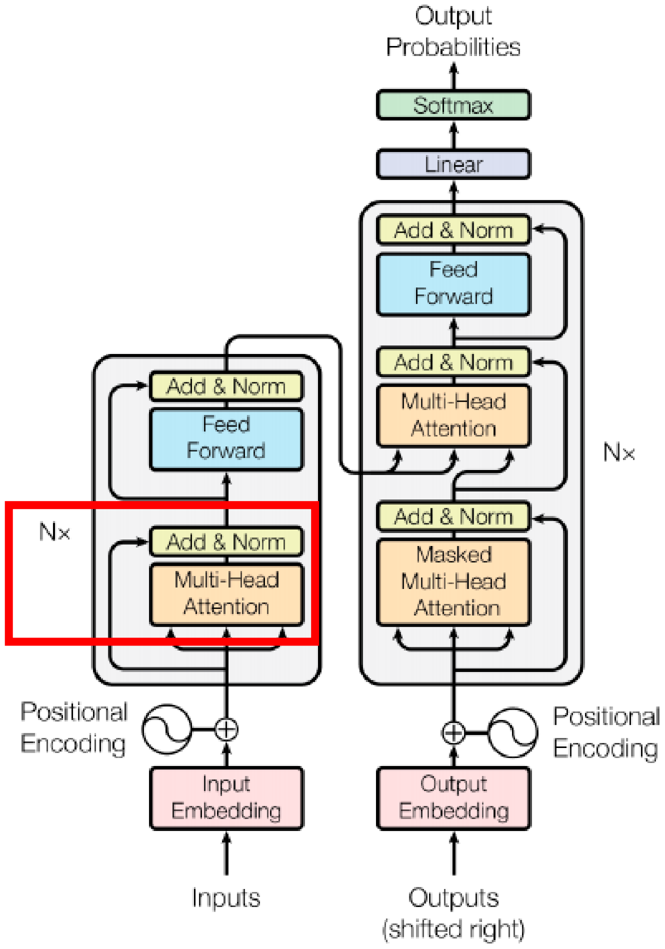
단어의 위치 정보가 손실됨

- Input Embedding과 Positional Encoding 벡터를 더한add 값이 첫 번째 인코딩 블록의 입력으로(만) 쓰임

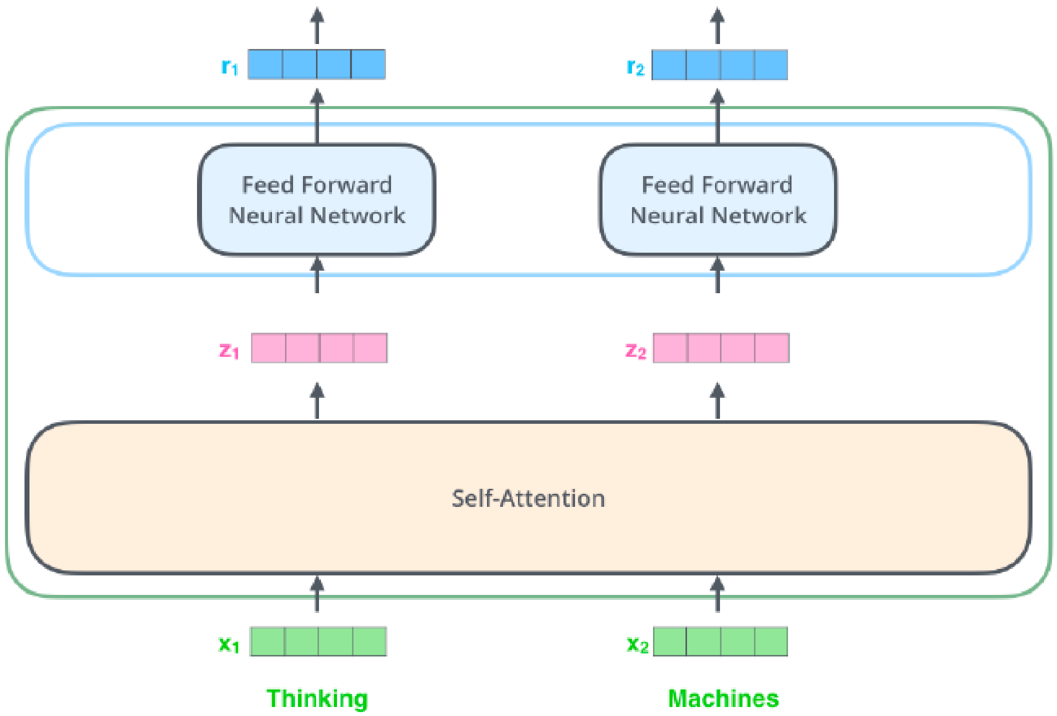
$$PE_{(pos,2i)} = \sin(pos/10000^{2i/d_{model}})$$

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

Unit 02 | Transformer



3. Multi-Head Attention, Residual connection & Normalization

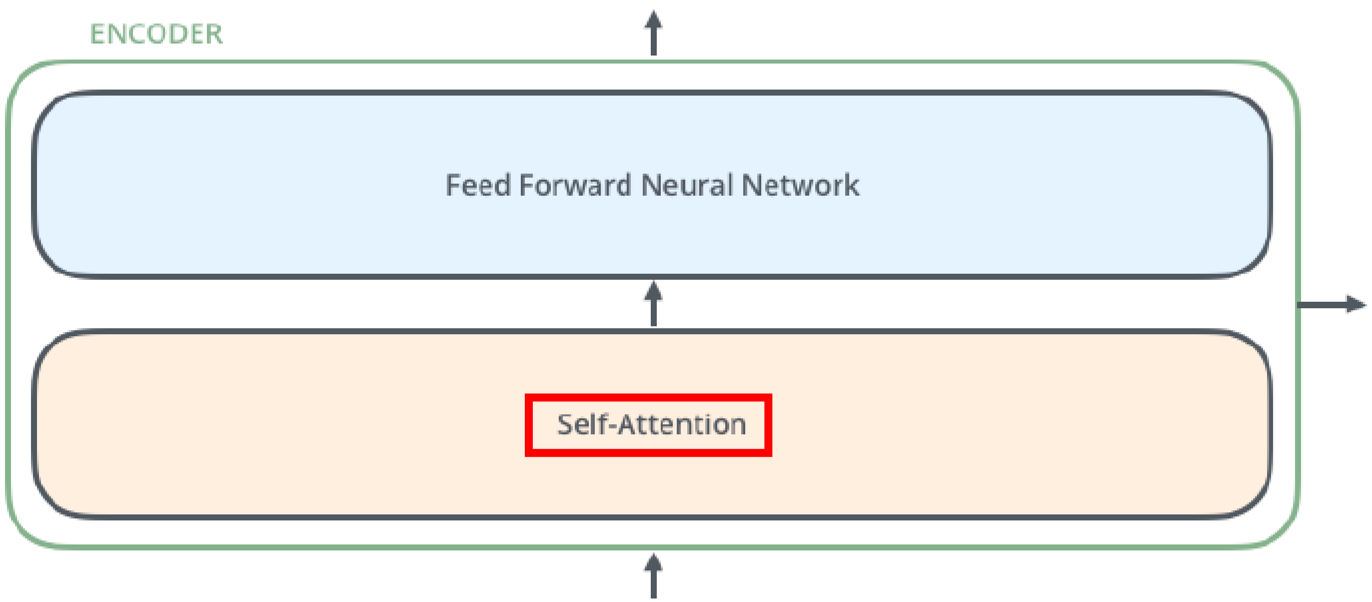


dependency X

dependency O

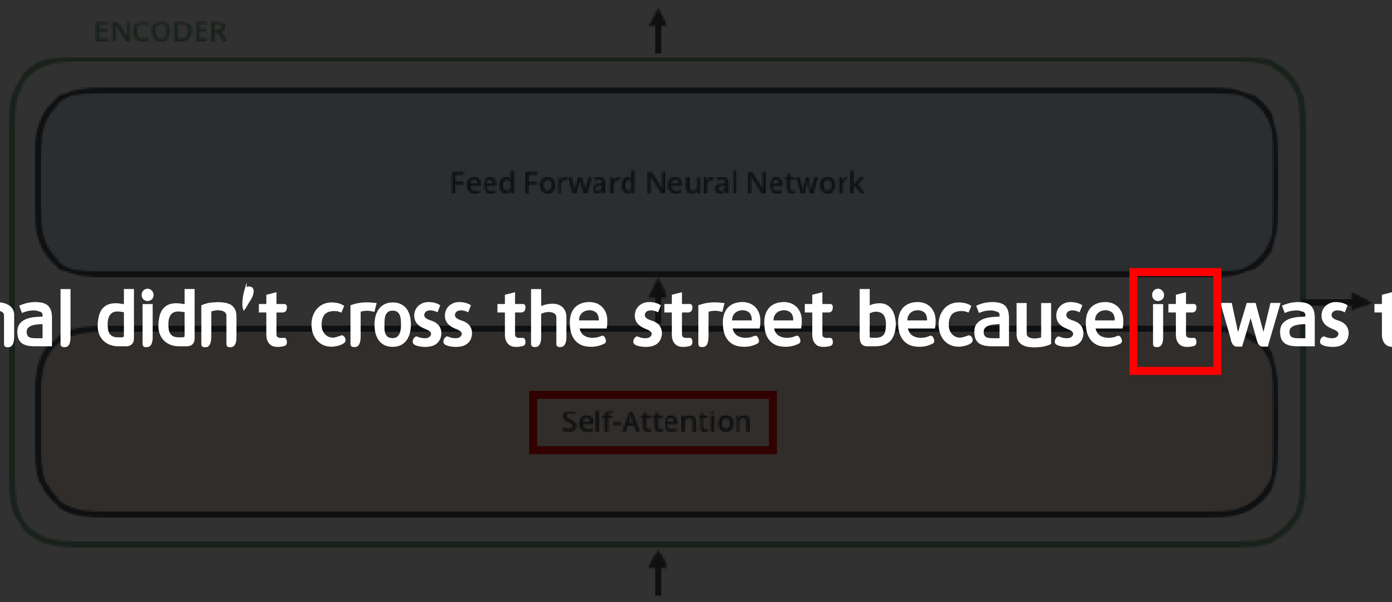
벡터들이 서로 영향을 미침

Unit 02 | Transformer



한 token을 처리할 때

다른 token을 얼마나 중요하게 볼 것인가에 대한 layer



The animal didn't cross the street because it was too tired

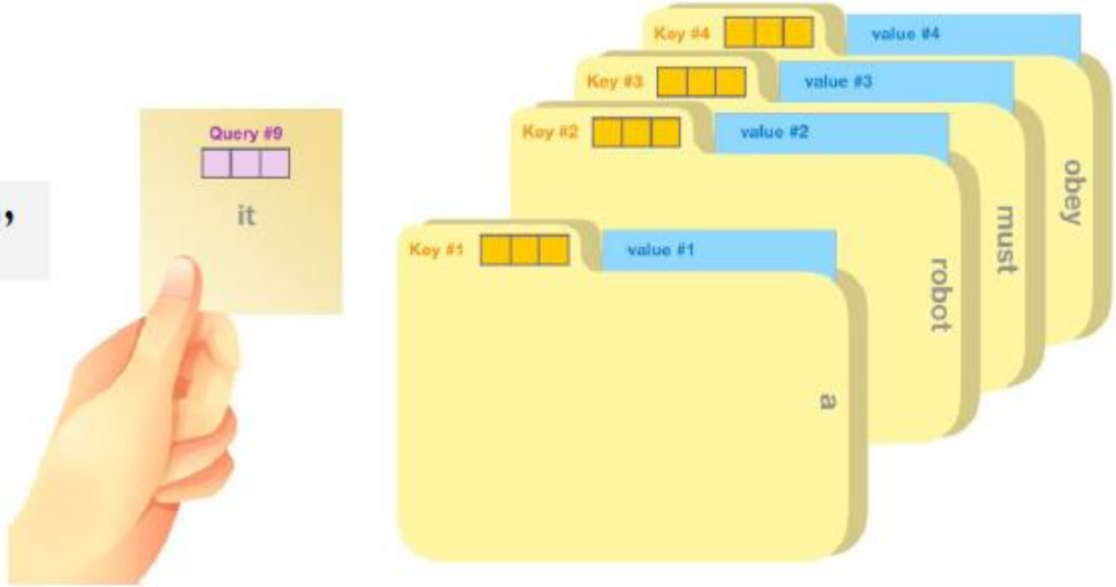
한 token을 처리할 때

다른 token을 얼마나 중요하게 볼 것인가에 대한 layer

Step 1

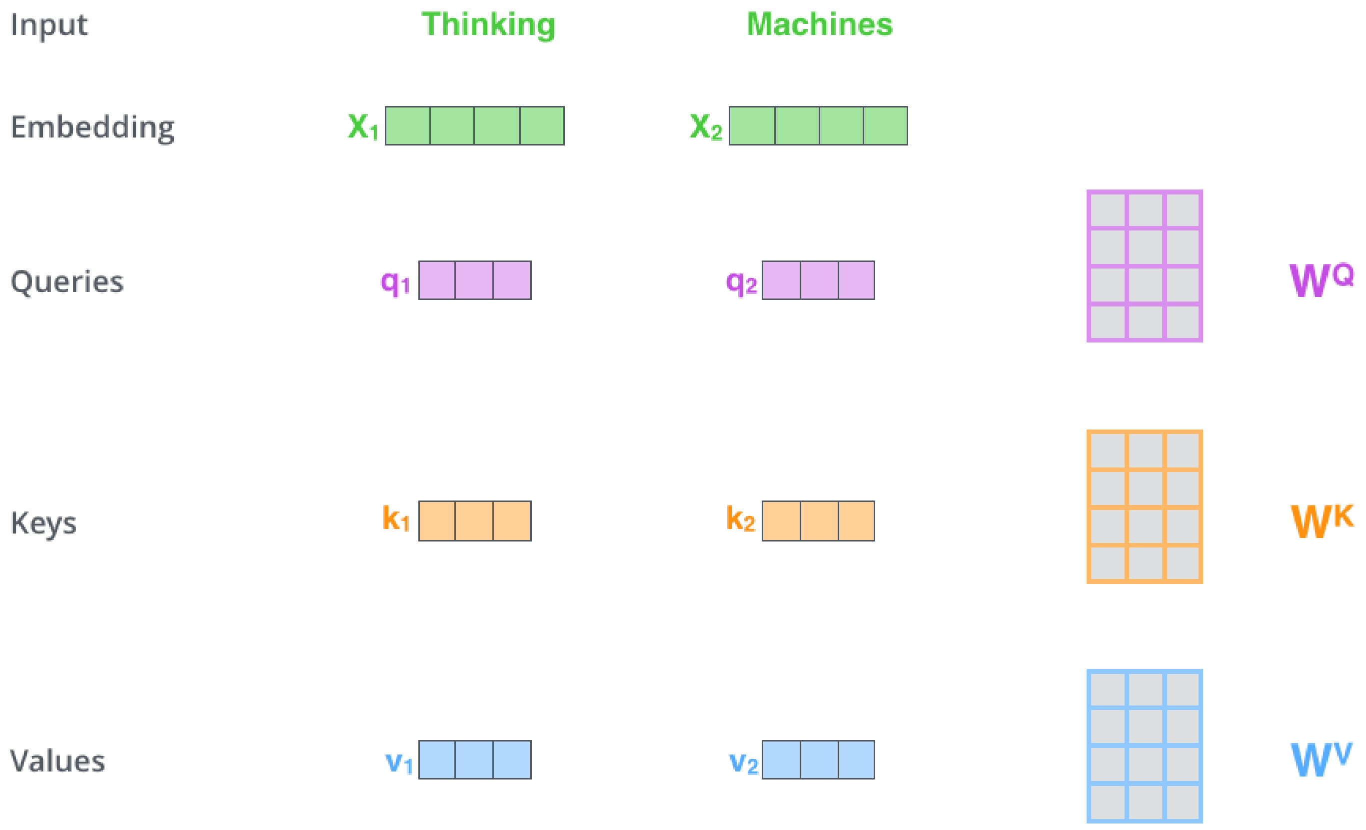
- Query: representation of the current word
- Key: like labels
- Value: actual word representations

"A robot must obey the orders given it"

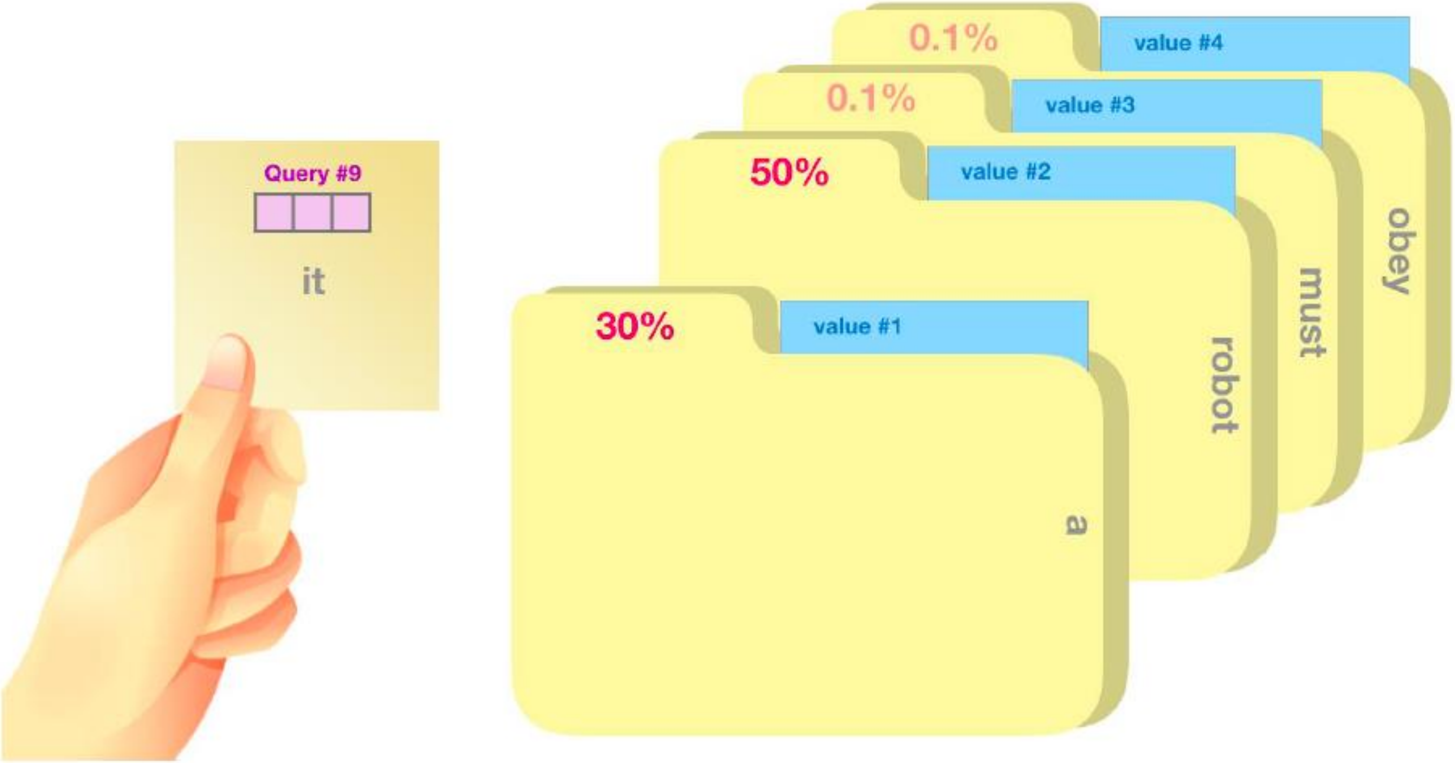


query와 key로 적절한 value를 찾아 연산

Step 1

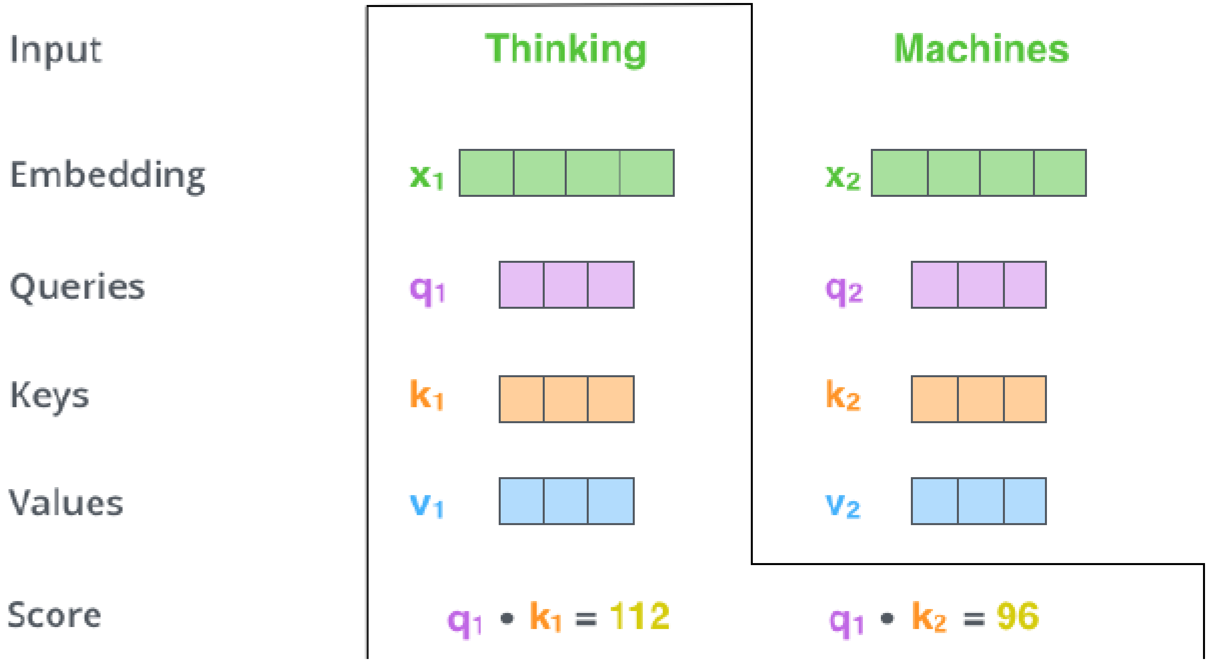

Input
Thinking
Machines
Embedding
X_1
X_2
Queries
q_1
q_2
W^Q
Keys
k_1
k_2
W^K
Values
v_1
v_2
W^V

Step 2



query vector와 key vector를 곱해 각각의 "폴더"에 대해 score 계산

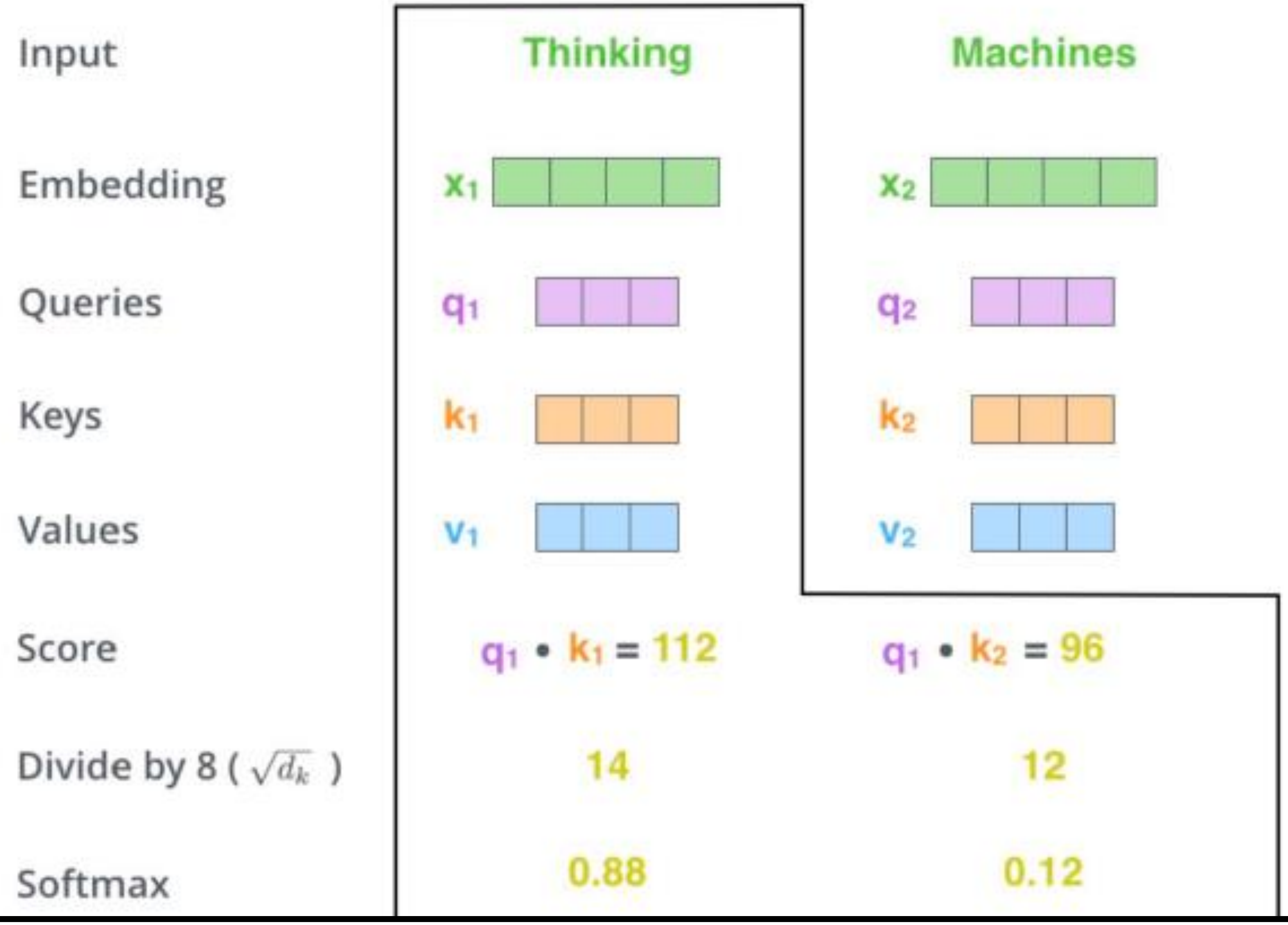
Step 2



query "Thinking" vector와 모든 key vector를 곱해 토큰에 대해 score 계산

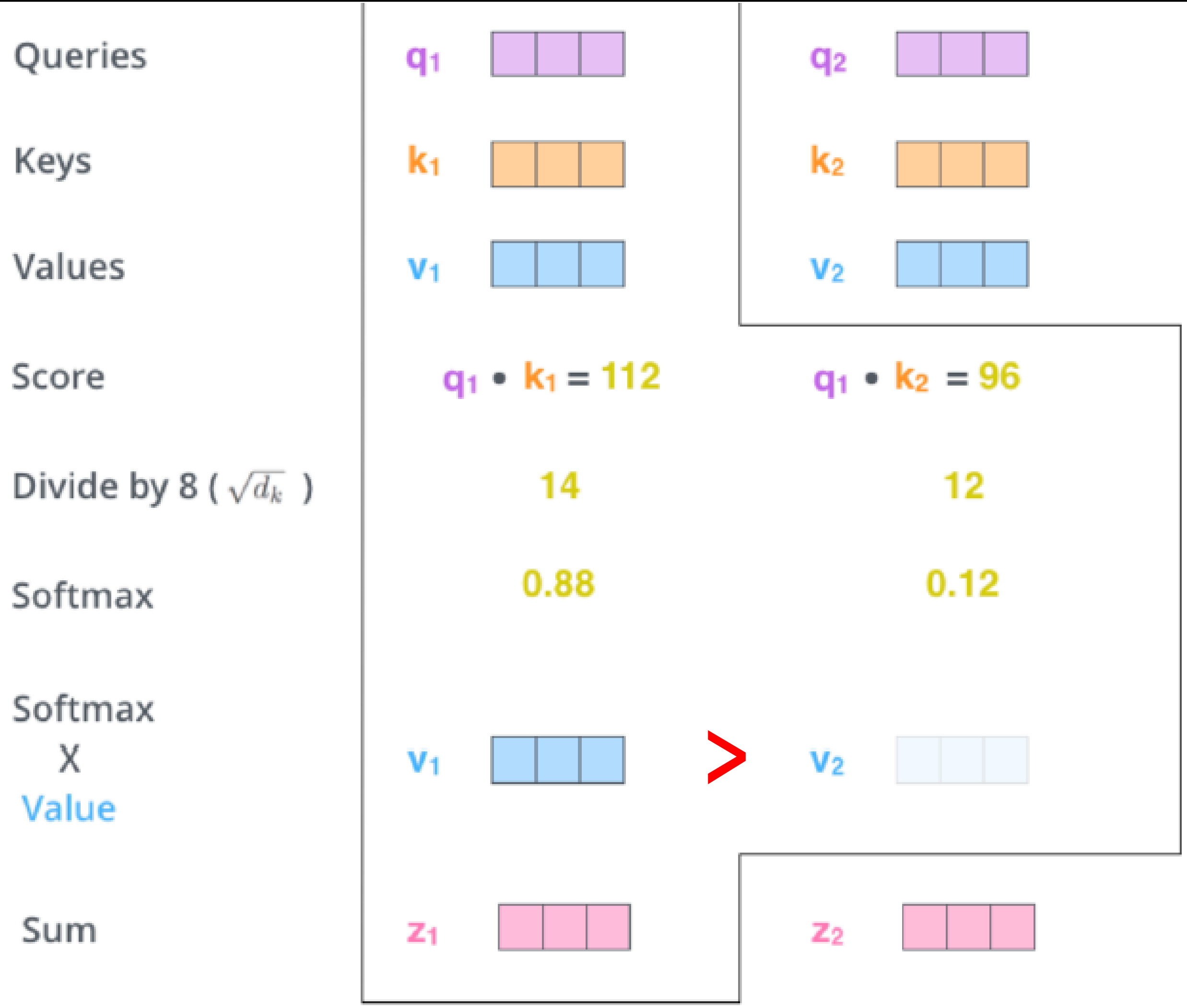
Step 3 and 4

- Step 3: $\sqrt{dimension}$ 으로 나누기
- Step 4: softmax operation

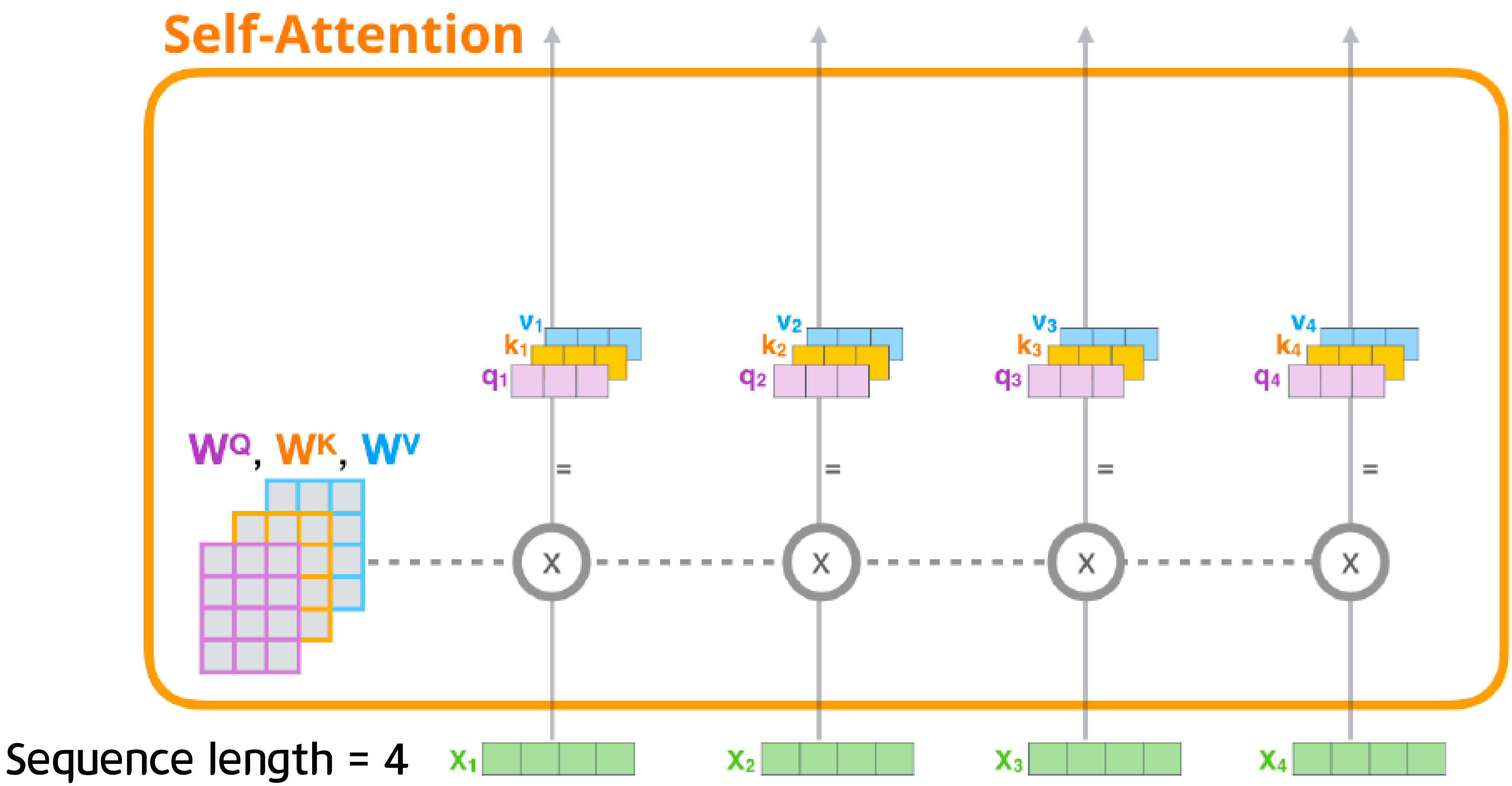


Step 5 and 6

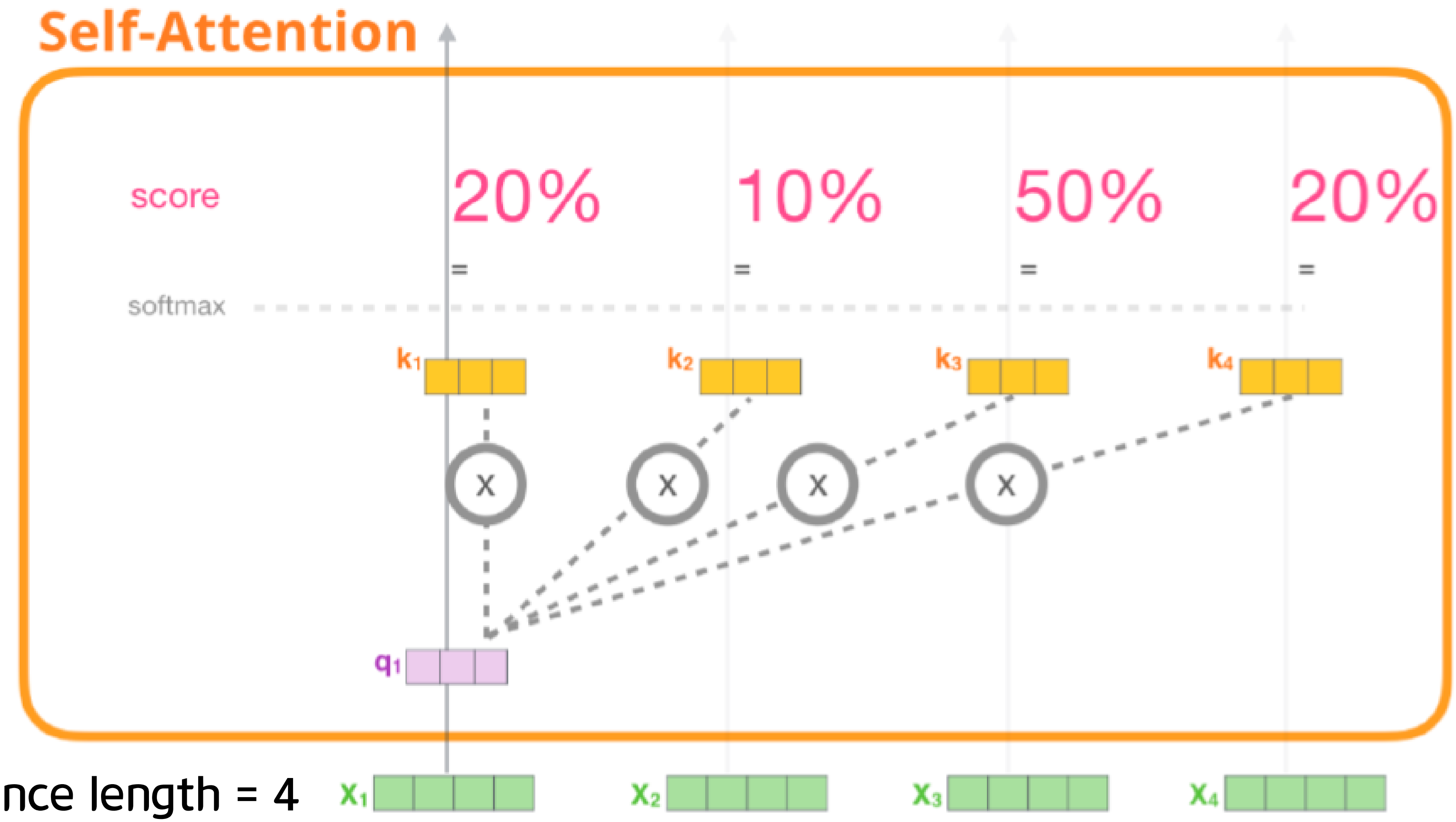
- Step 5: softmax * value
- Step 6: weighted vector 더하기



Unit 02 | Transformer



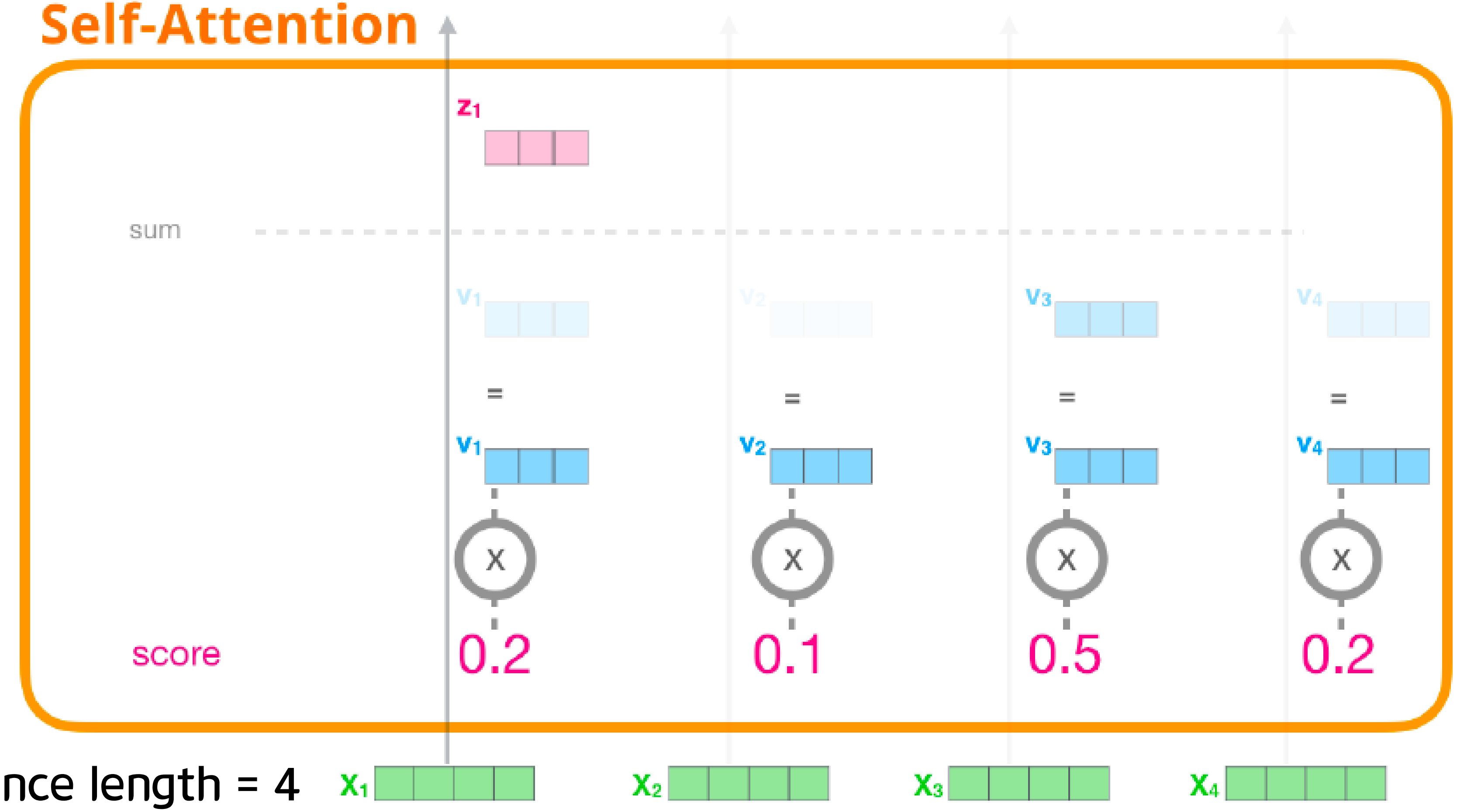
Unit 02 | Transformer



Sequence length = 4

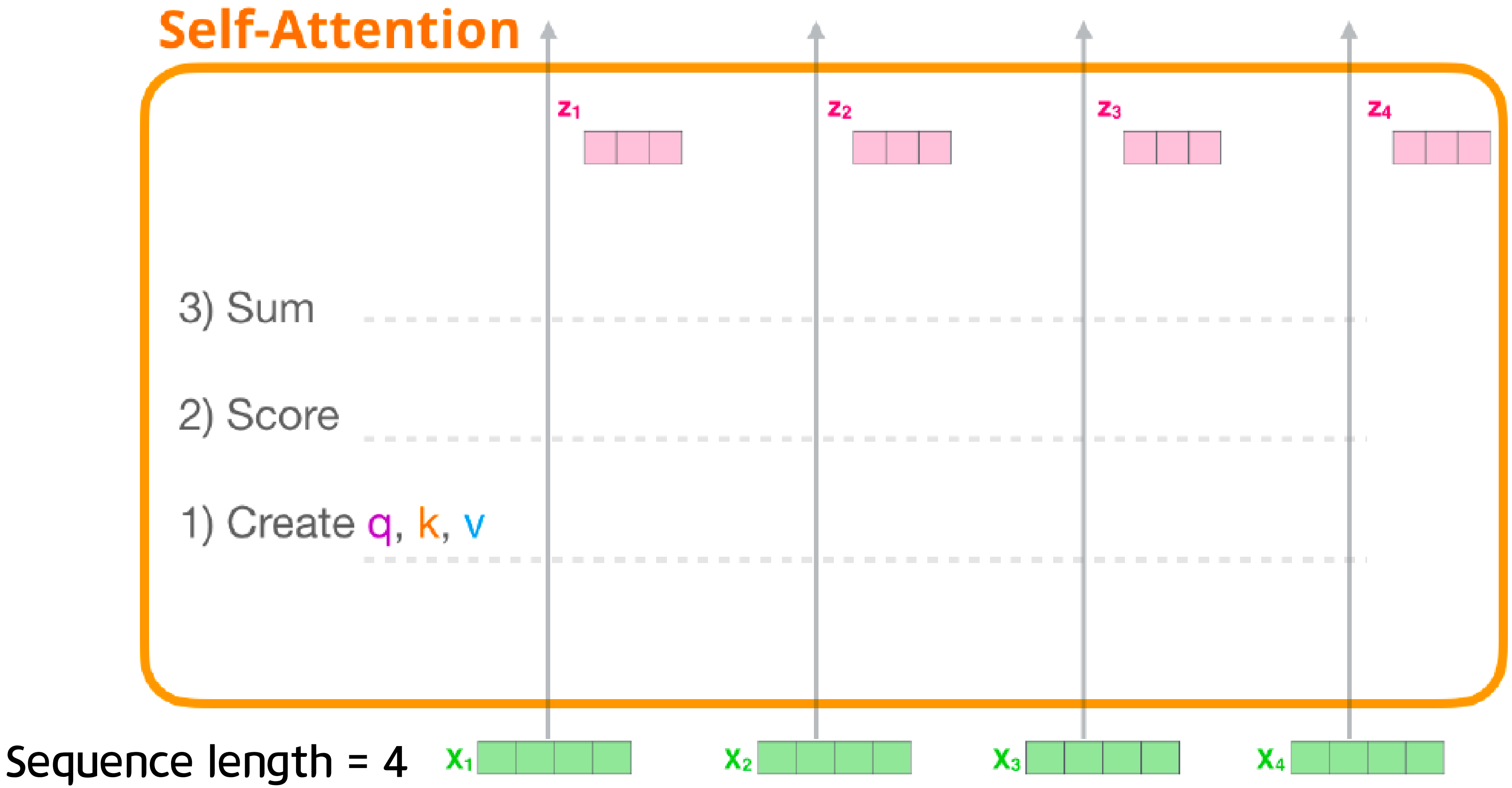
Unit 02 | Transformer

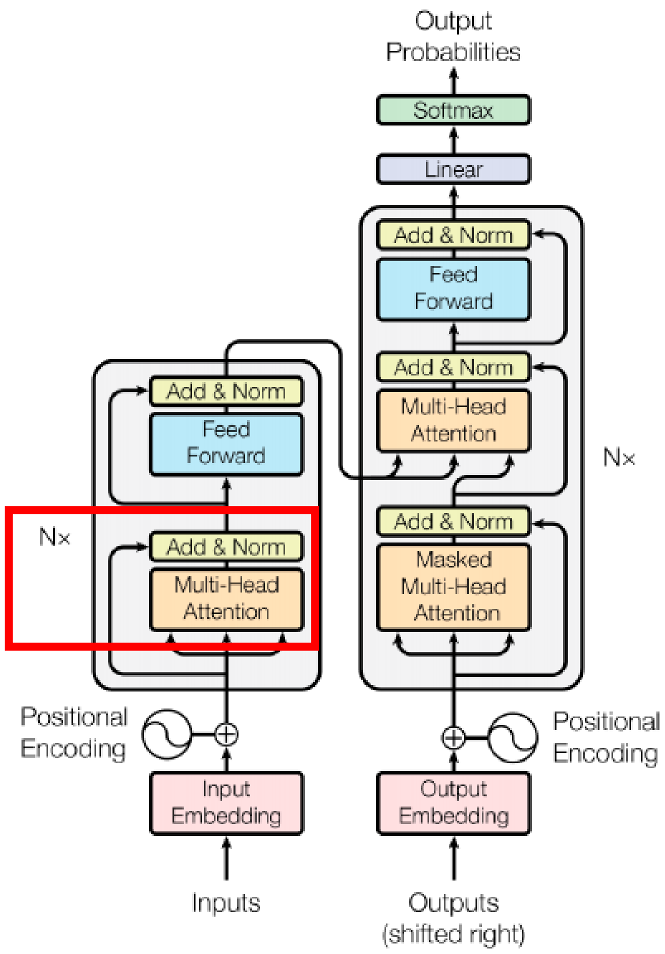
Self-Attention



Sequence length = 4

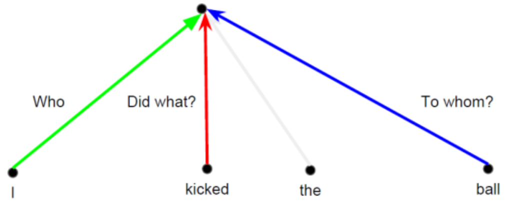
Unit 02 | Transformer

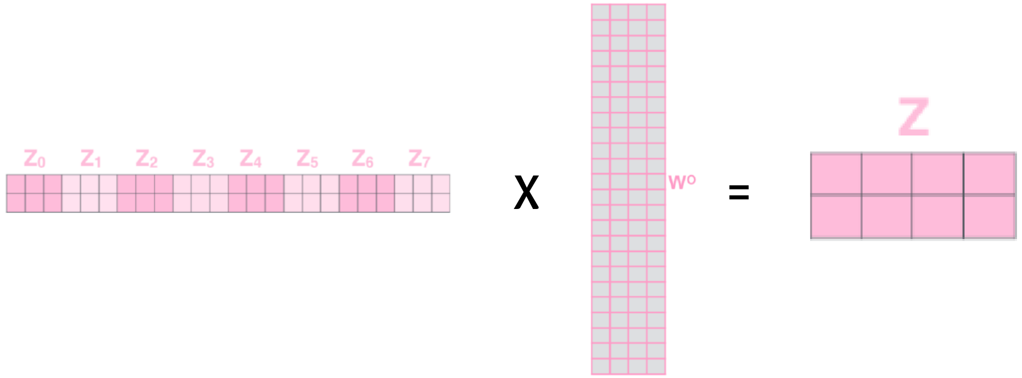




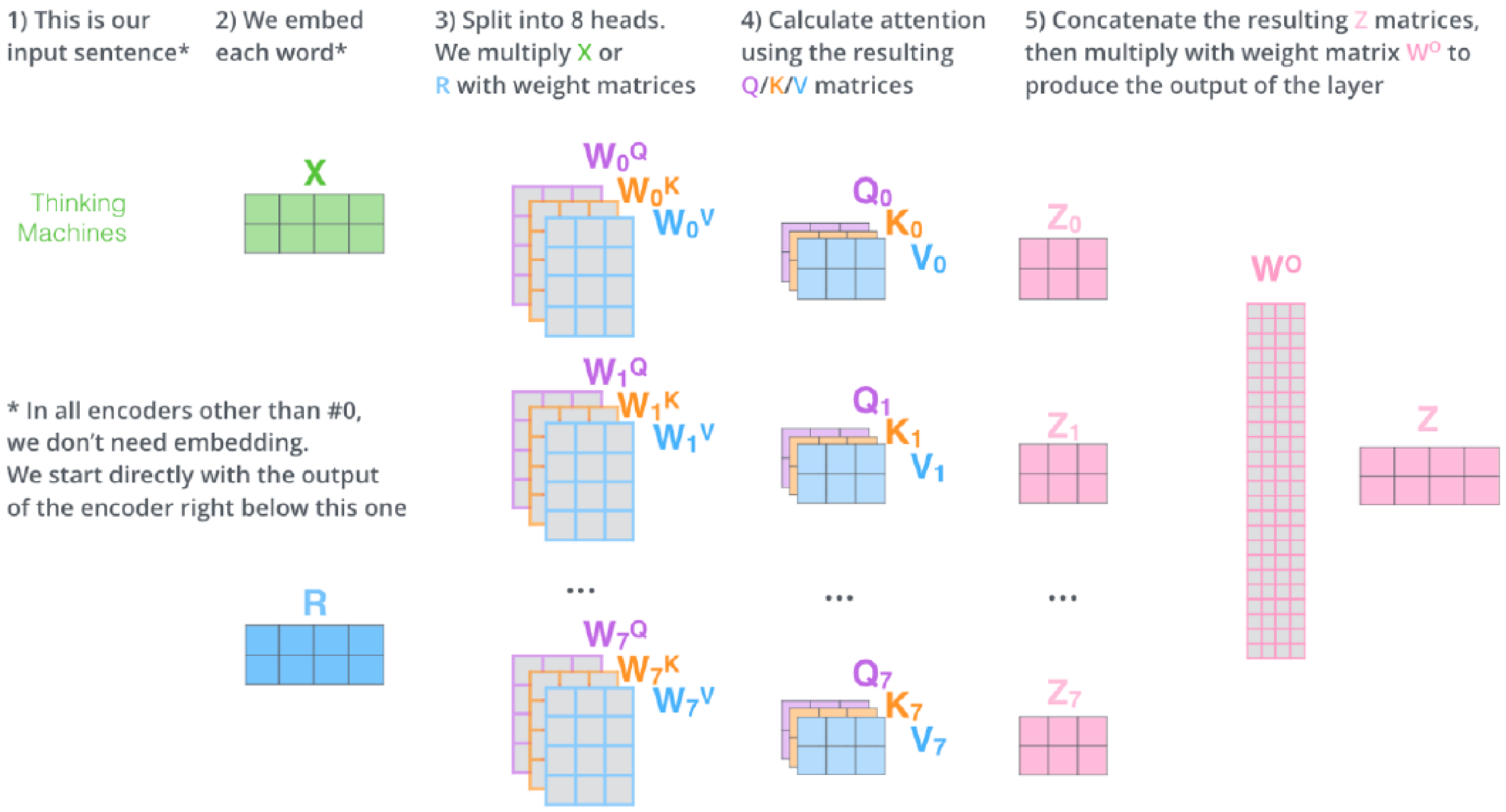
3. Multi-Head Attention, Residual connection & Normalization

- 한 문장 내에서 존재하는 다양한 정보를 **한 번**의 attention 으로 적절히 반영하기 어려움
- 8 Attention Heads (64d * 8 = 512d)

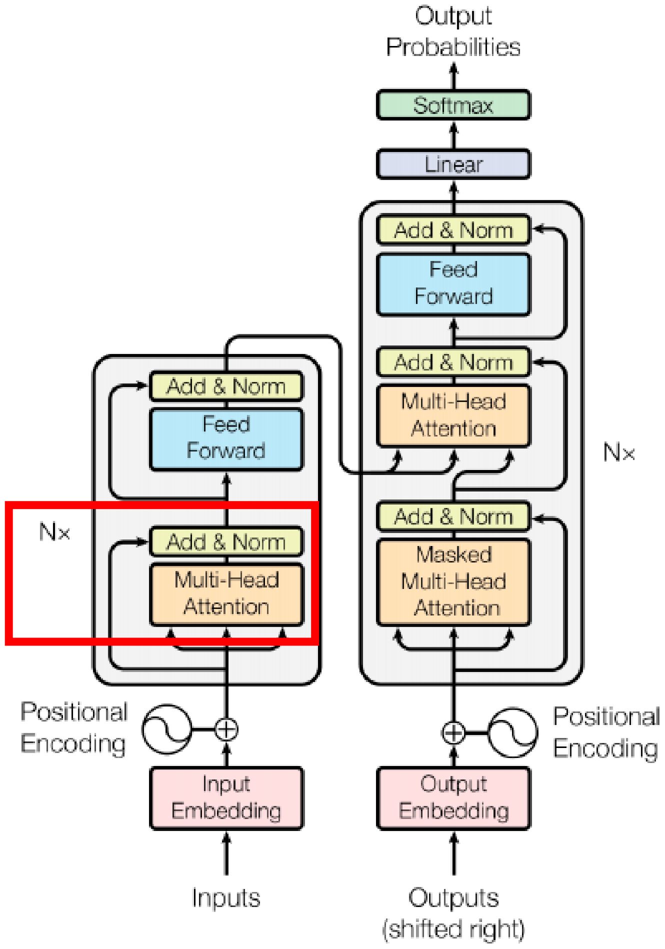




Unit 02 | Transformer

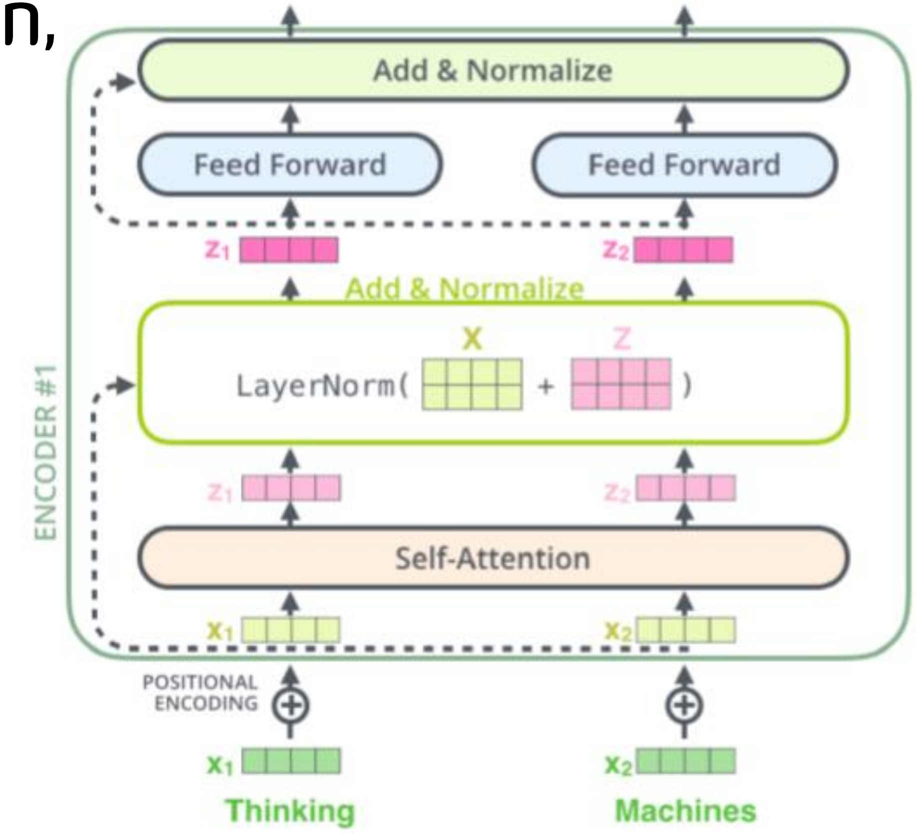


Unit 02 | Transformer

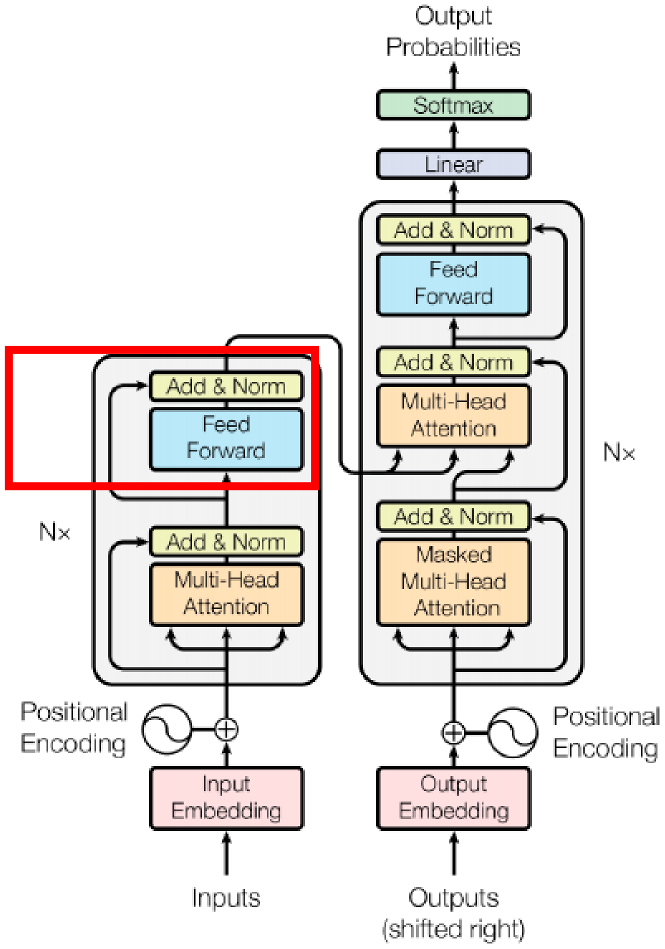


3. Multi-Head Attention, Residual connection & Normalization

- Add & LayerNorm



Unit 02 | Transformer

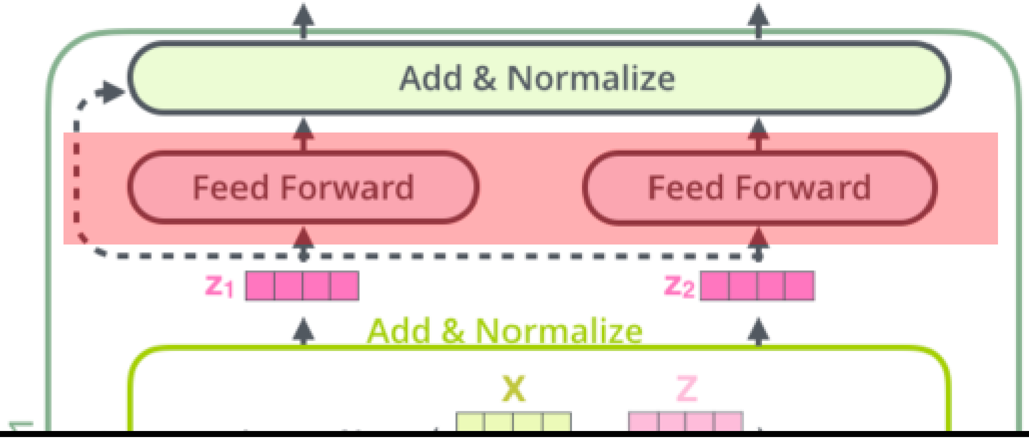


4. Position-wise Feed-Forward Networks

- Fully connected feed-forward network

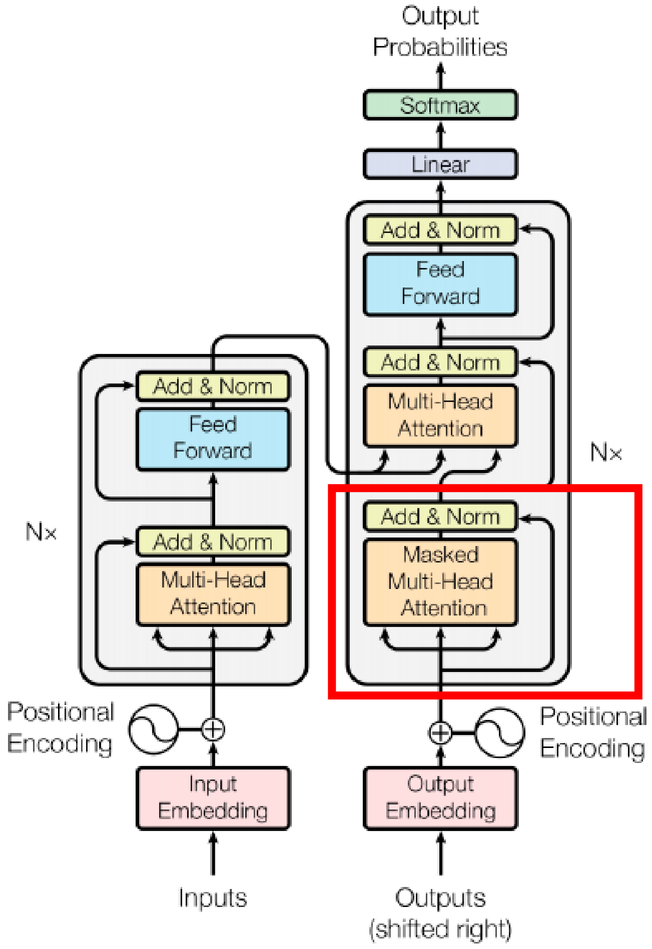
- Different parameters from layer to layer

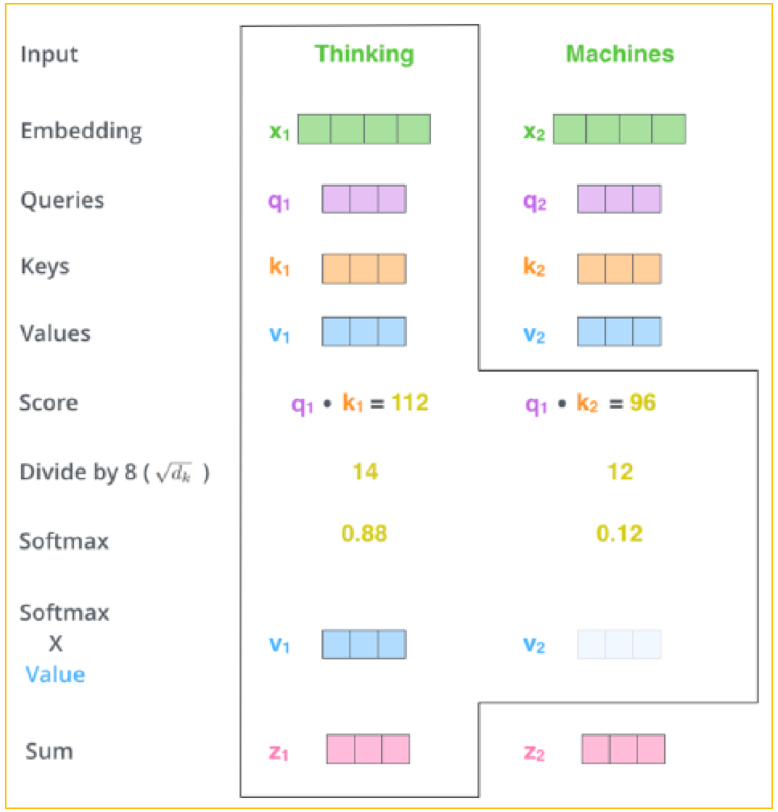
  같은 layer의 FFNN은 같은 가중치를 가짐

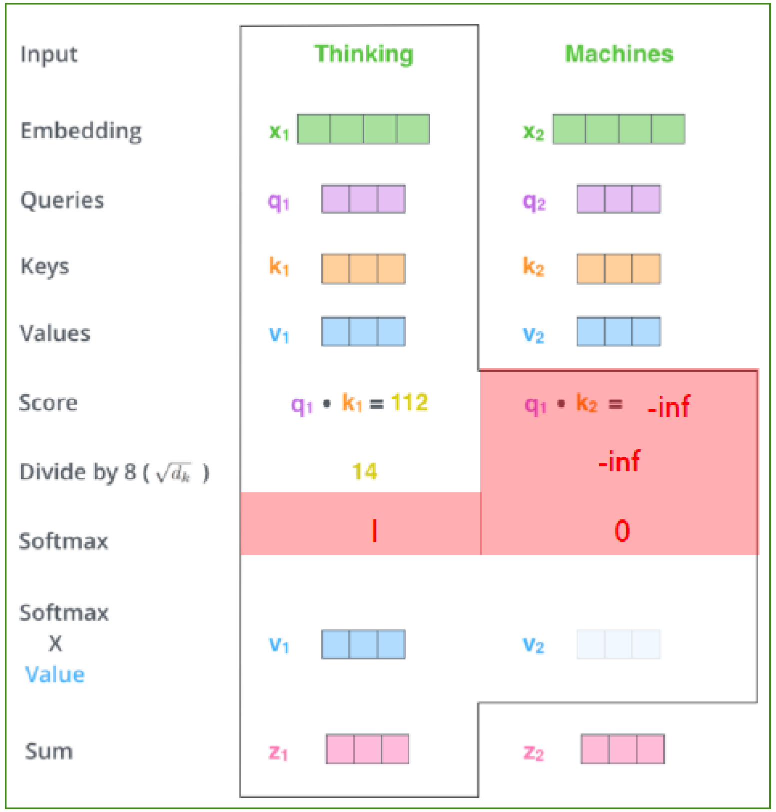


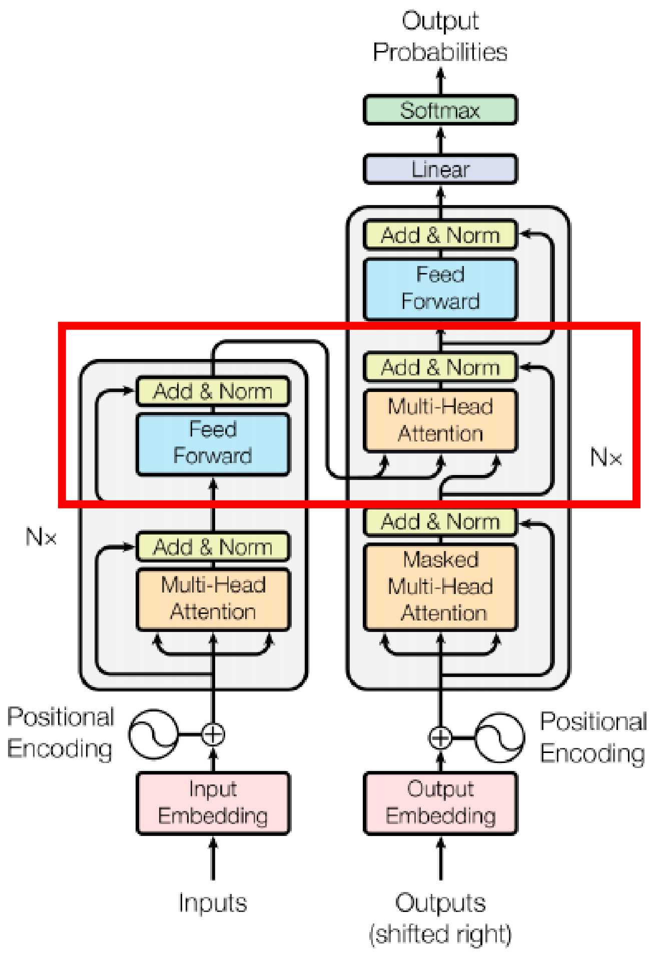
Unit 02 | Transformer

5. Masked Multi-Head Attention

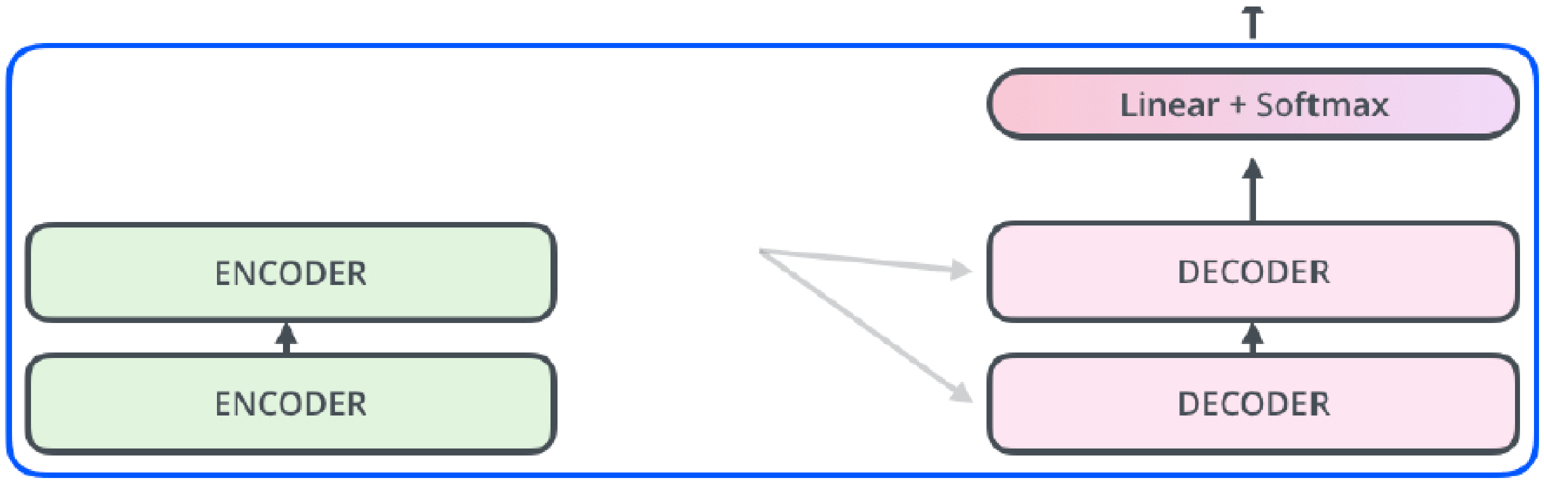


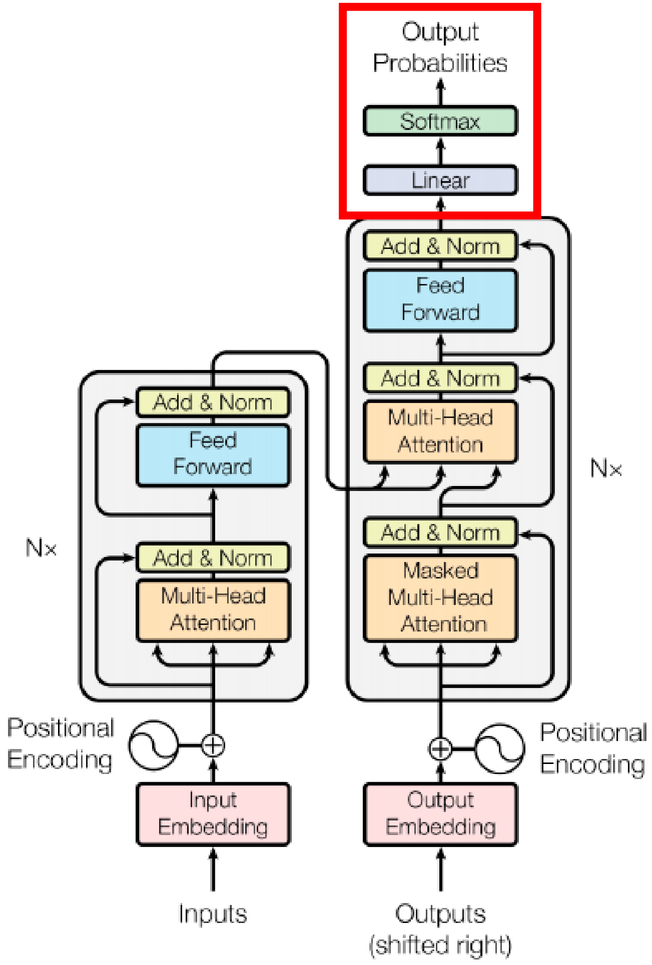






6. Multi-Head Attention with Encoder Outputs





7. The Final Linear and Softmax Layer

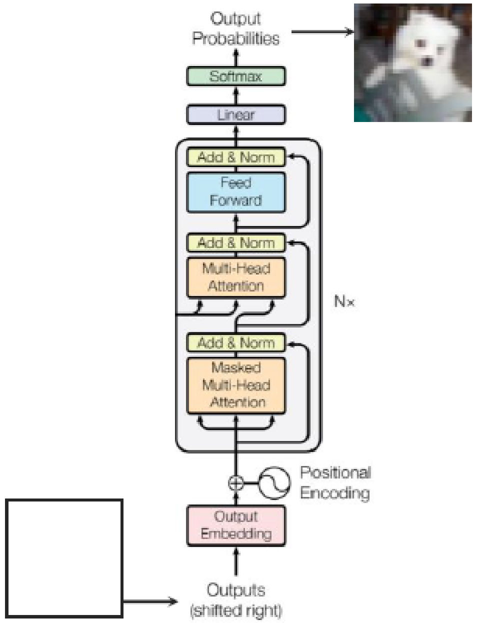
- Linear layer: fully connected neural network

- Softmax layer: scores into probability

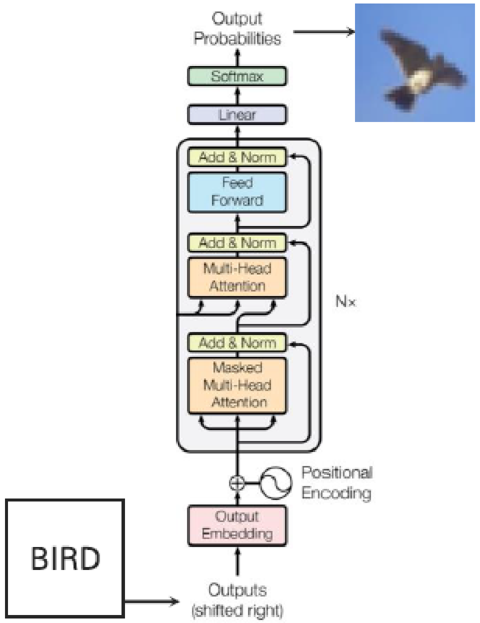
  어떤 단어가 현재 인덱스의 단어와 연관되는지?

Unit 03 | Image Transformer

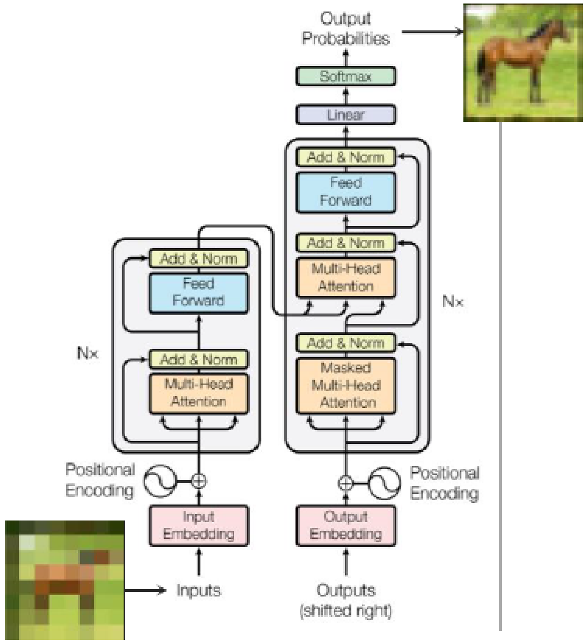
Unconditional
Image Generation



(Class) Conditional
Image Generation



Super Resolution



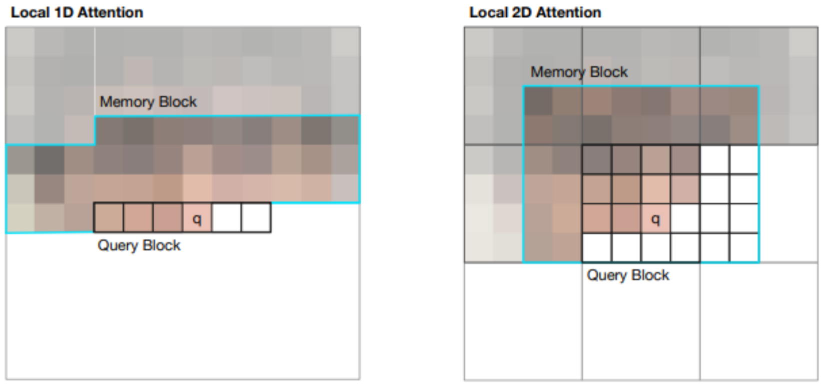
Unit 03 | Image Transformer

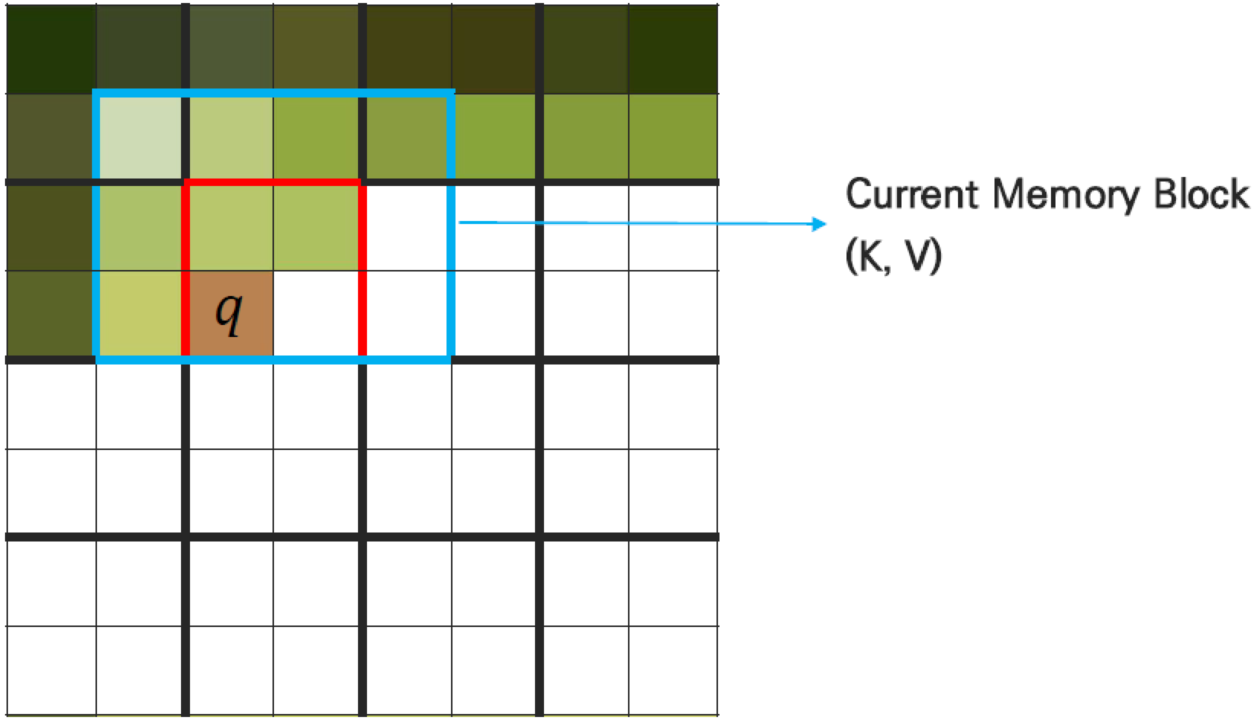
Layer Type	Complexity per Layer	Sequential Operations	Maximum Path Length
Self-Attention	$\boxed{O(n^2 \cdot d)}$	$O(1)$	$O(1)$

Input	$\boldsymbol{n}$	$\boldsymbol{O(n^2 \cdot d)}$	
8x8 image	$8 * 8 * 3 = 192$	$192 * 192 * d$	Feasible
32x32 image	$32 * 32 * 3 = 3072$	$3072 * 3072 * d$	Infeasible

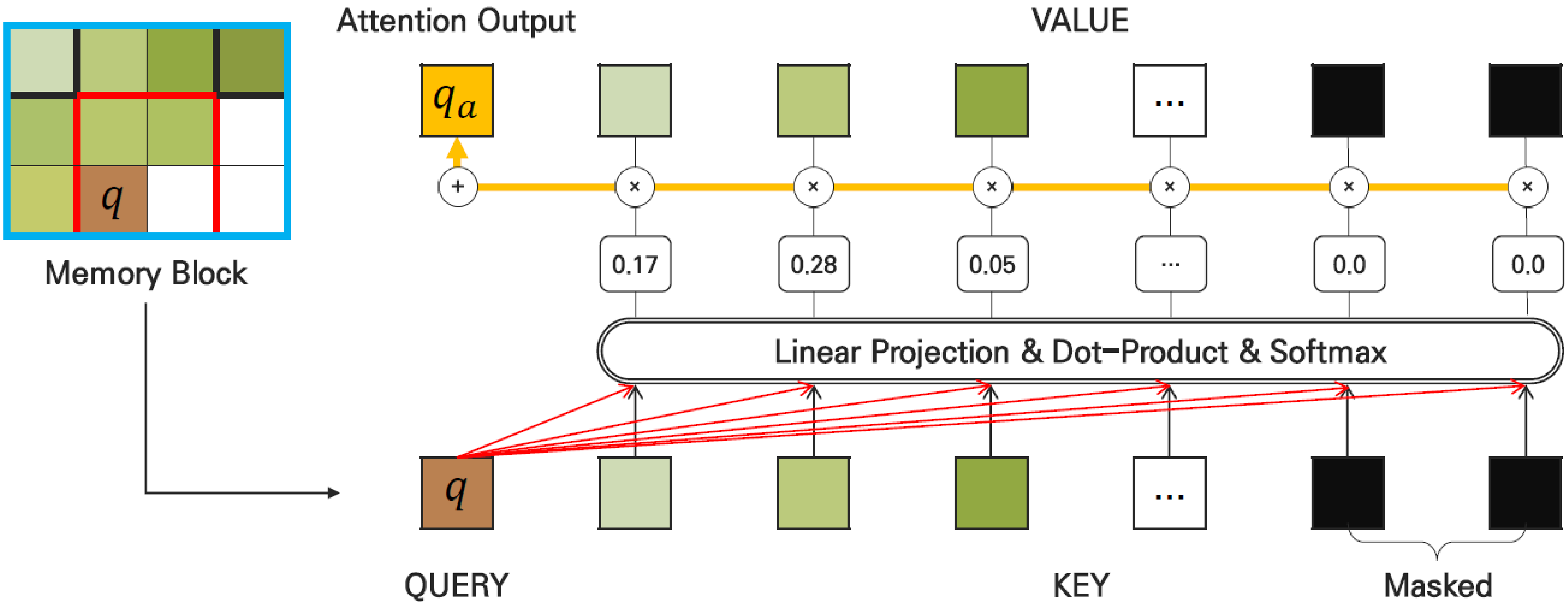
그래서 Image Transformer는 Local Self-Attention을 사용합니다.

- Sequence 내 일정 부분 안에서만 self-attention을 적용 (예: Super Resolution)



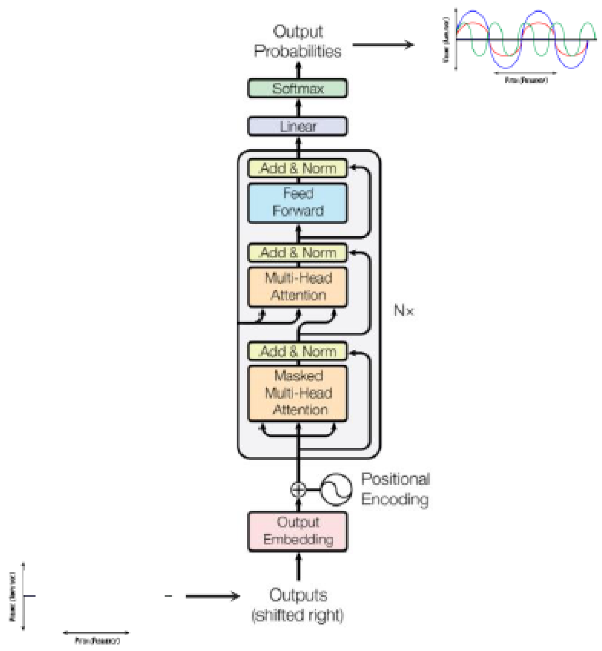


Unit 03 | Image Transformer

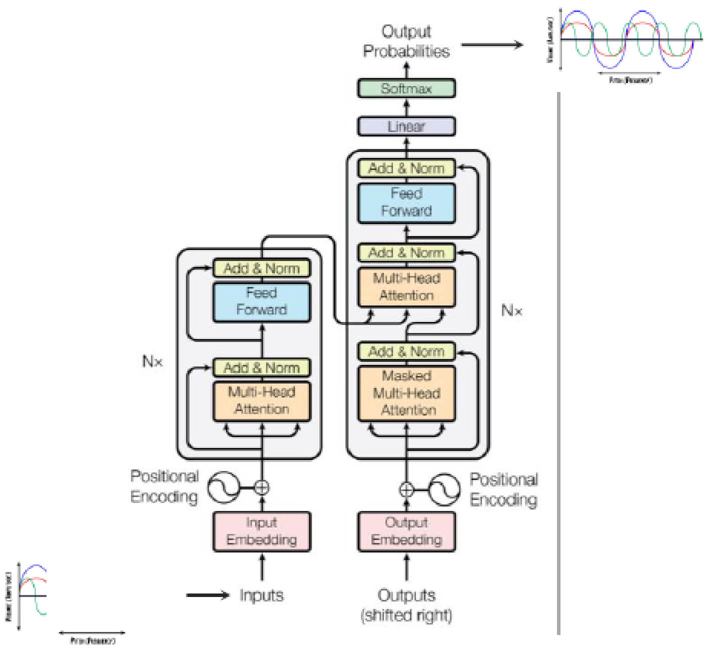


Unit 04 | Music Transformer

Unconditional Music Generation



Conditional Music Generation



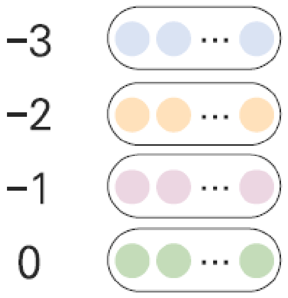
Music Transformer는 Relative Positional Self-Attention을 사용합니다.

- Query와 Key의 sequence 내 거리를 attention weight에 반영

$$Relative\ Attention = Softmax\left(\frac{QK^T + \boxed{S^{rel}}}{\sqrt{D_h}}\right)V$$

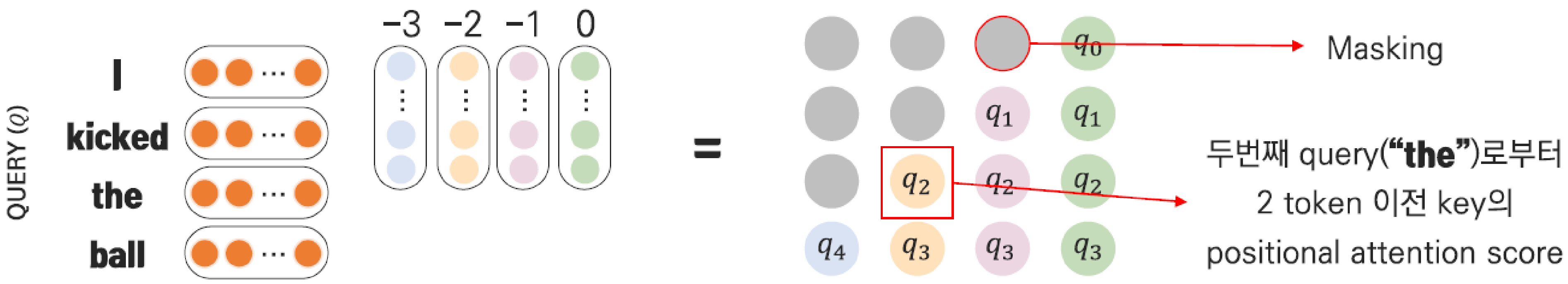
Unit 04 | Music Transformer

Step 1. Relative Position Embedding Matrix (E^r)



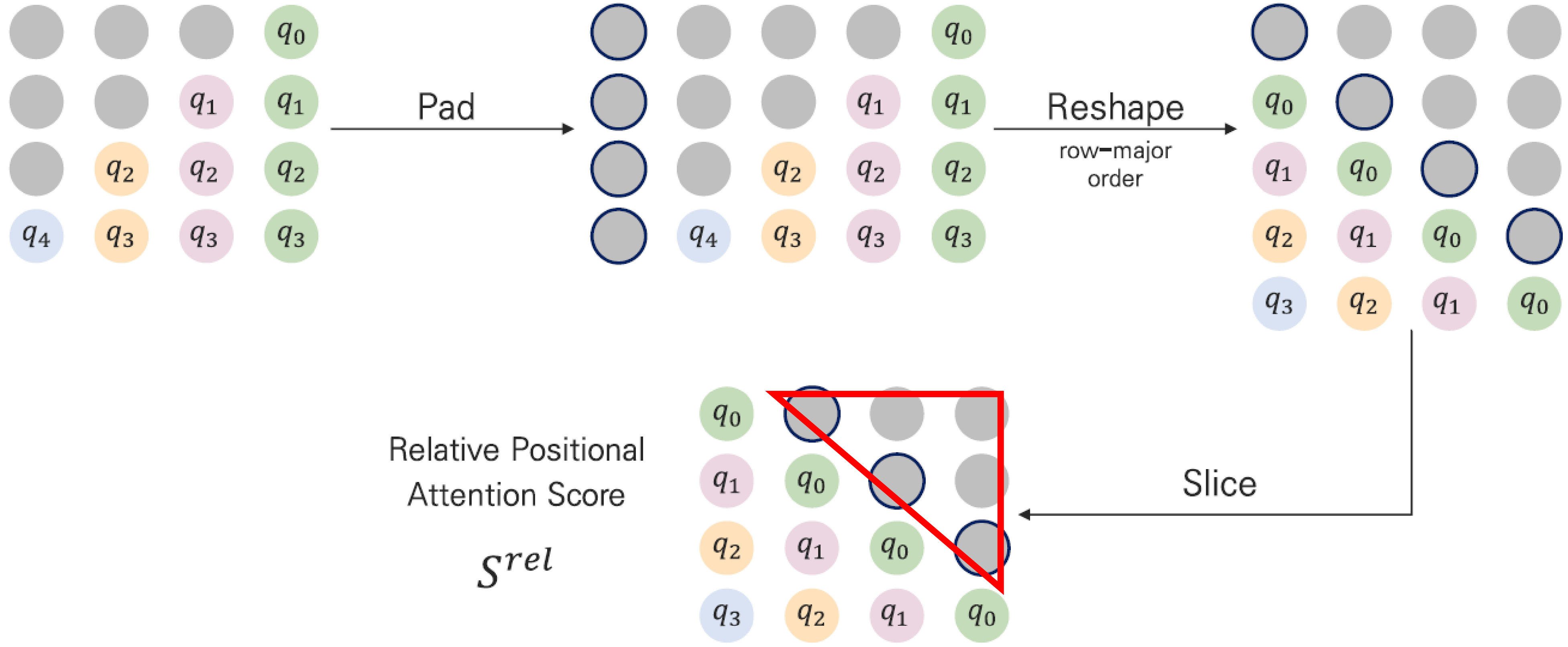
query로부터 3 token 이전 key의 position vector
query로부터 2 token 이전 key의 position vector
query로부터 1 token 이전 key의 position vector
query 자기 자신 위치에 해당하는 key의 position vector

Step 2. Relative Position Embedding Matrix와 Query를 곱함 (QE^{r^T})



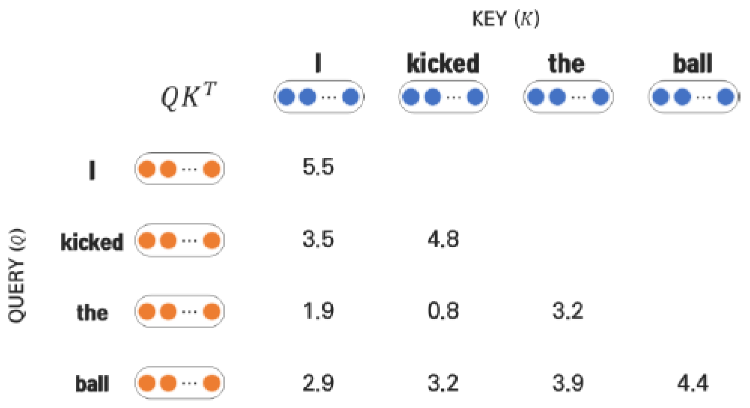
Unit 04 | Music Transformer

Step 3. Skewing (기존 attention score와 더할 수 있도록 각 score를 알맞은 자리로 reshaping)



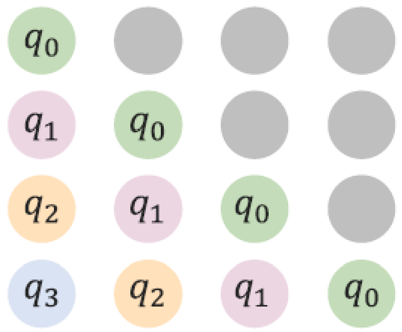
Unit 04 | Music Transformer

Step 4. 기존 attention score와 relative positional attention score를 더하여 output 산출



기존 Attention Score

$\oplus$

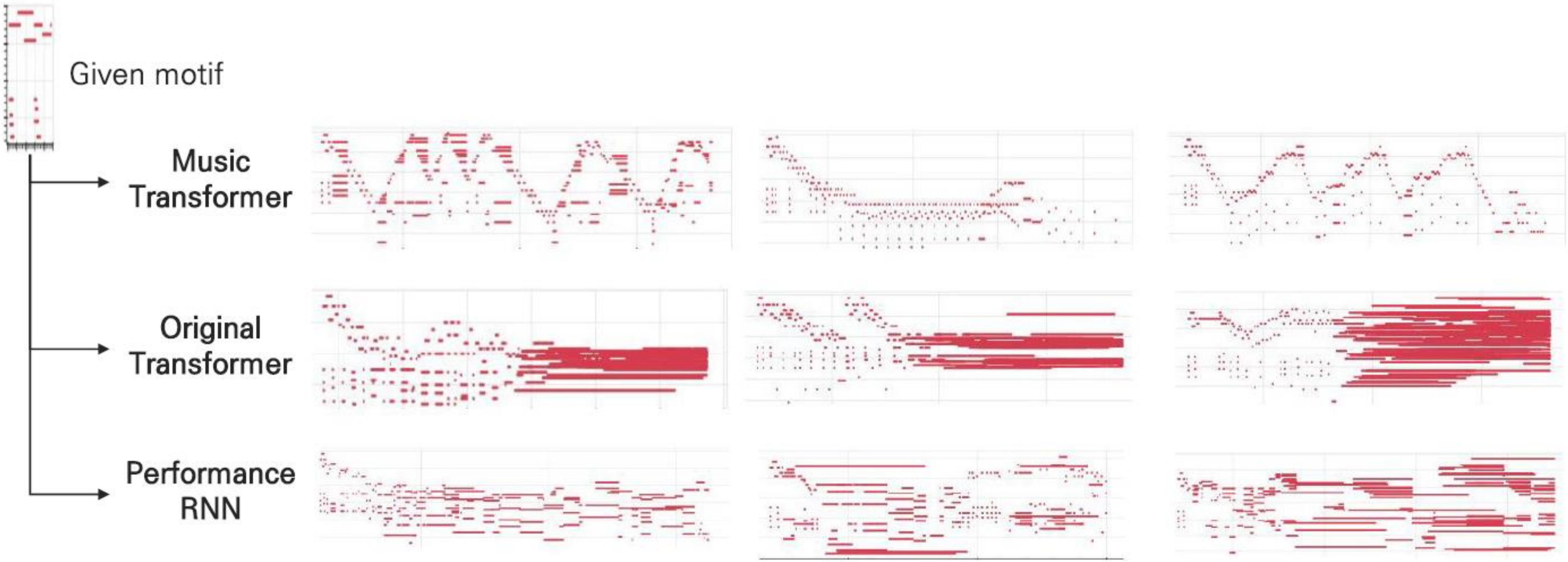


Relative Positional Attention Score

$= \quad QK^T + S^{rel}$

$$Relative\ Attention = Softmax\left(\frac{\boxed{QK^T + S^{rel}}}{\sqrt{D_h}}\right)V$$

Unit 04 | Music Transformer



Unit 05 | References

0.

CS224n: Natural Language Processing with Deep Learning in Stanford / Winter 2019 중

Transformers and Self-Attention For Generative Models *(guest lecture by Ashish Vaswani and Anna Huang)*

1.

고려대학교 산업경영공학과 DSBA 연구실 CS224n Winter 2019 세미나 중

14. Transformers and Self-Attention For Generative Models 강의자료와 강의 영상 (노영빈님)

2.

고려대학교 산업경영공학과 강필성교수님 2020-1학기 '비정형데이터분석' 수업 (Graduate) 중

08-2_Transformer 강의자료와 강의 영상

Q & A

들어주셔서 감사합니다.