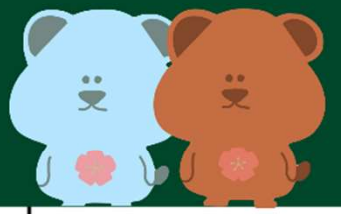
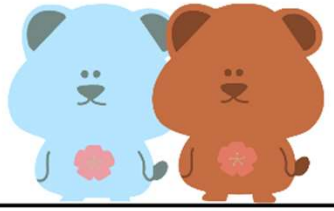


Week17_Transformers and Self-Attention

발표자: 조서영, 임세영

EWHA EURON

목차



EWHA EURON

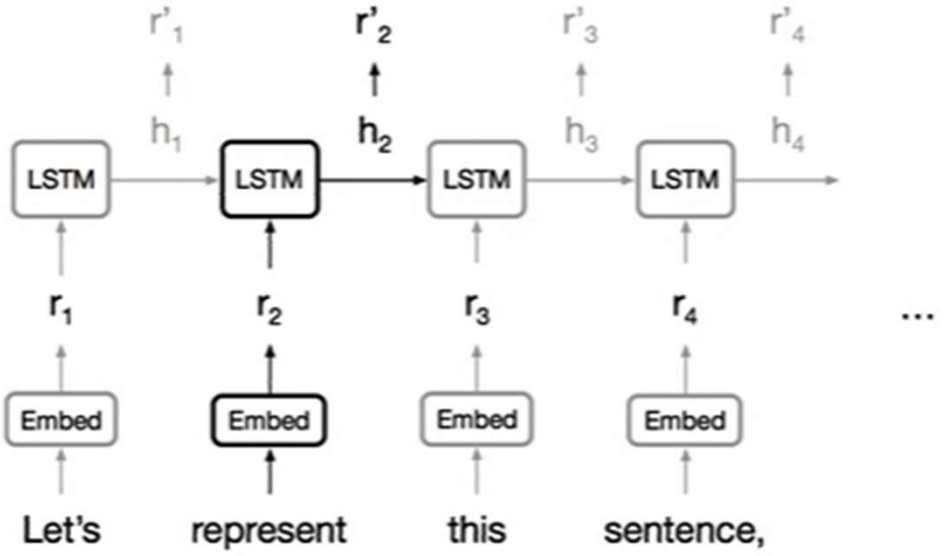
Introduction

EWHA EURON

Introduction

RNN

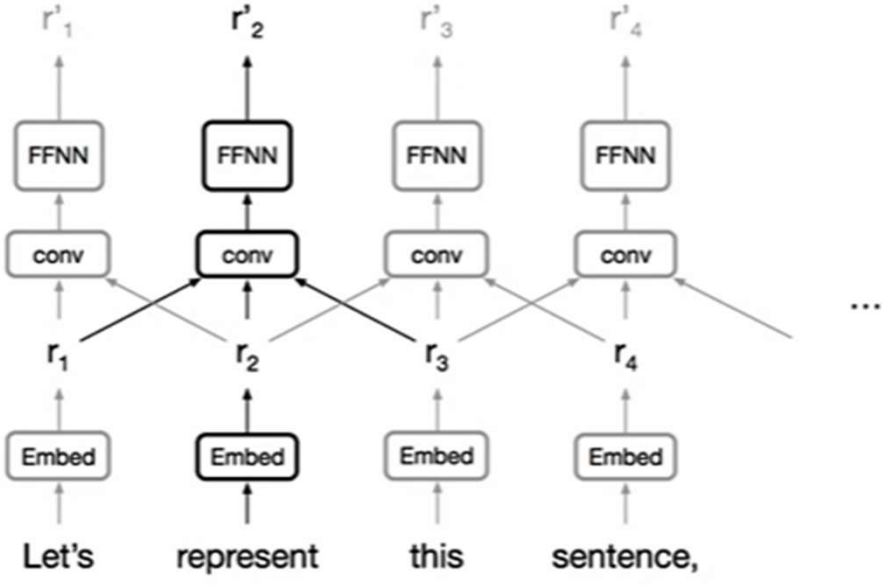
- 이전 단계 계산이 끝나야 다음 단계 계산 가능 -> 병렬화(parallelization) 불가능
- Long term dependency 잘 반영 못 함



Introduction

CNN

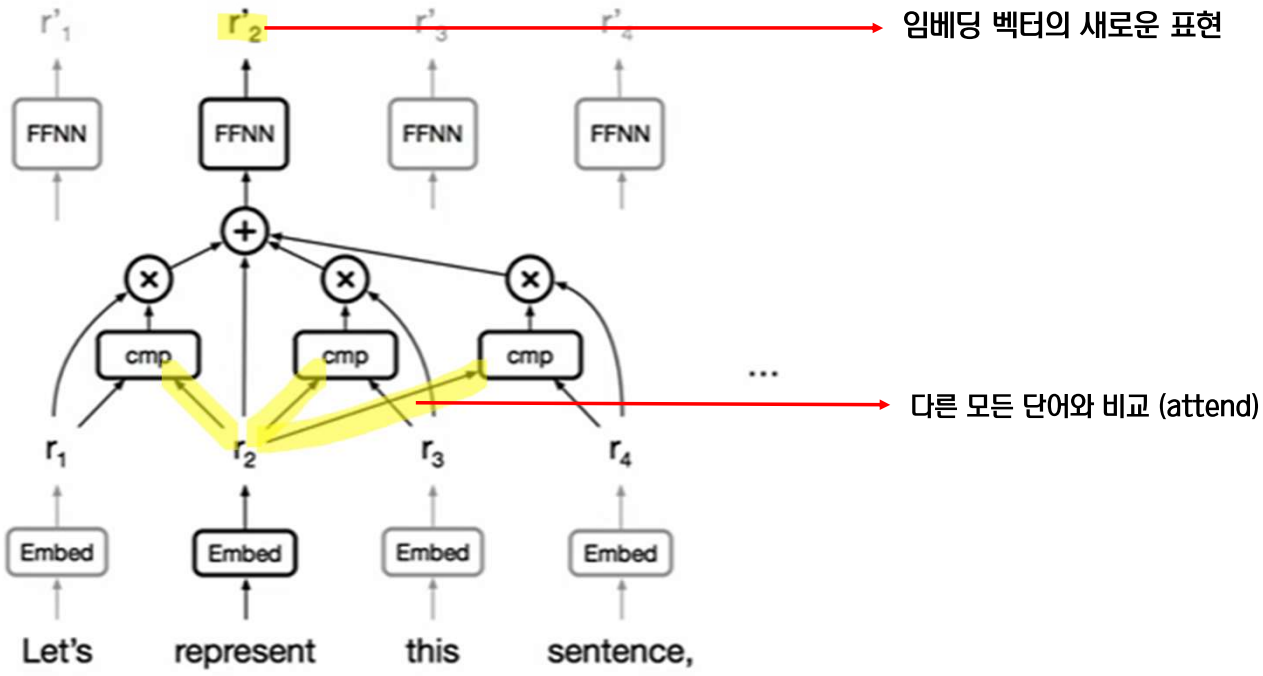
- 병렬화(parallelization)는 가능
- Long term dependency 잘 반영 못 함 (반영하려면 많은 layer가 필요)



Introduction

Self-attention

- 병렬화(parallelization) 가능
- 각 토큰이 최단 거리로 연결 -> Long term dependency 문제 해결



Introduction

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	$O(k \cdot n \cdot d^2)$	$O(1)$	$O(\log_k(n))$

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

병렬화 가능

Long-term dependency 문제 해결

EWHA EURON

Transformer

EWHA EURON

Transformer

- Transformer
  - RNN과 같이 순차적으로 처리해야 하는 부분 모두 제거
  - Attention과 Fully-connected layer만 사용한 encoder와 decoder

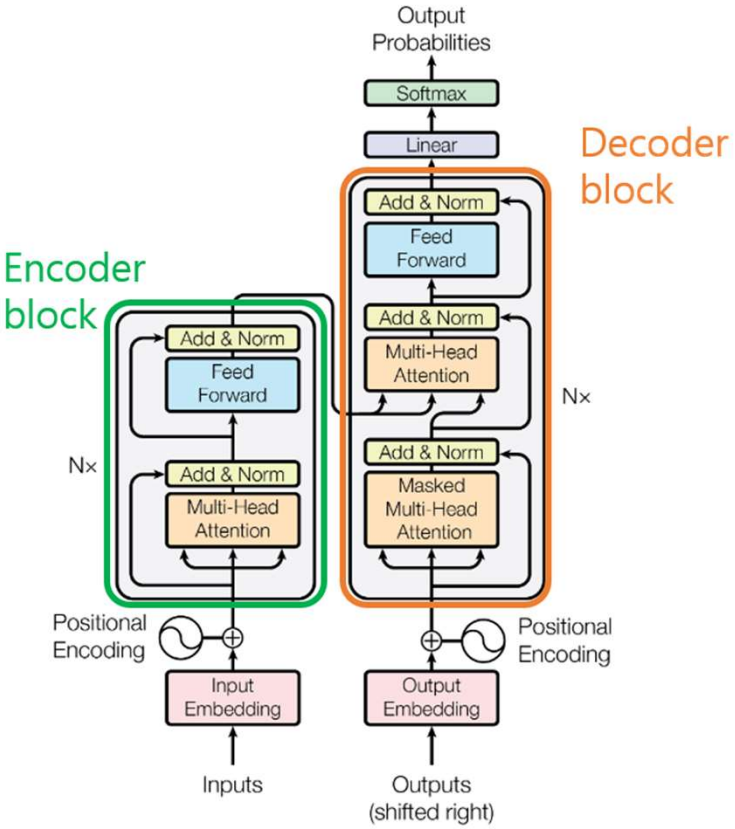
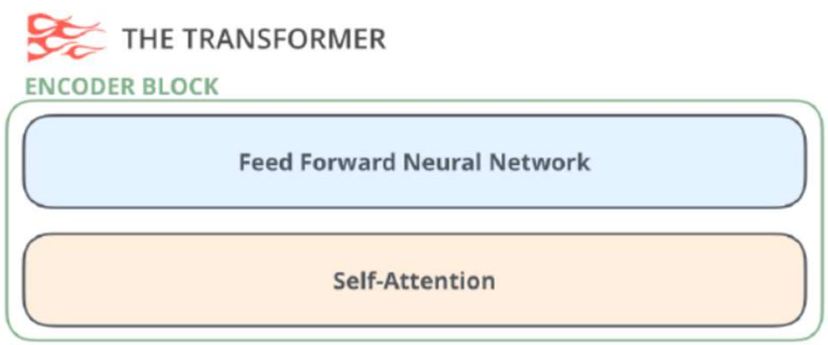
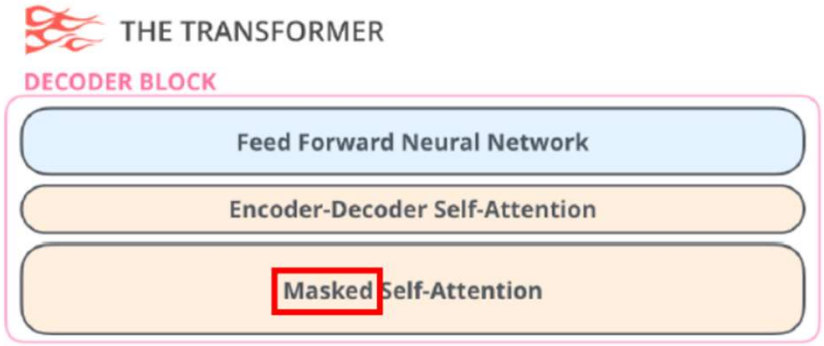


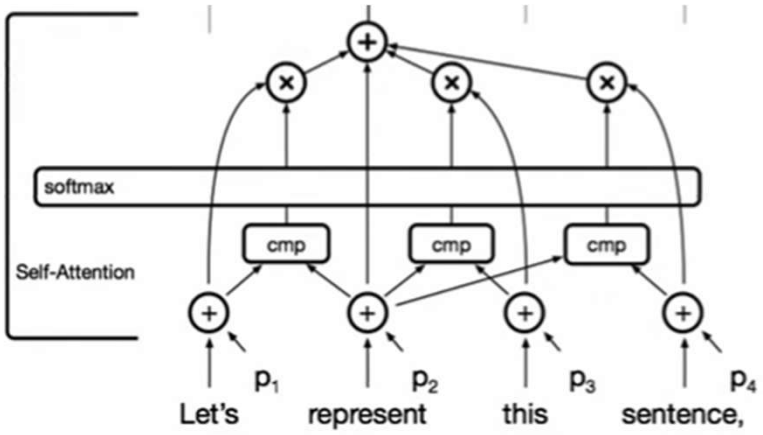
Figure 1: The Transformer - model architecture.

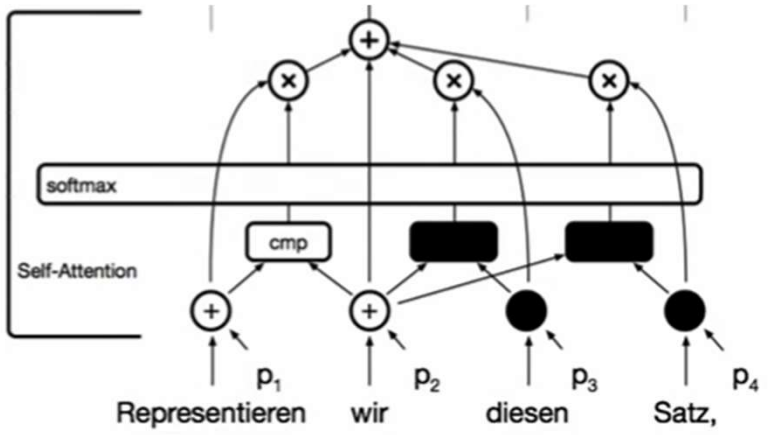
Transformer

- Encoder block vs. Decoder block









Transformer

- Transformer – 1. Input Embedding

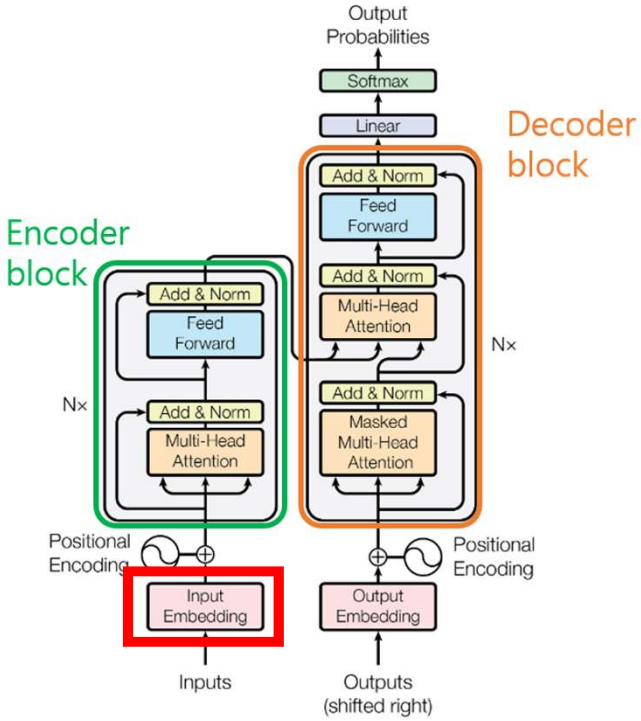


Figure 1: The Transformer - model architecture.

- Word2Vec과 같은 임베딩 알고리즘 사용하여 단어를 벡터로 변환

EWHA EURON

Transformer

- Transformer – 2. Positional Embedding

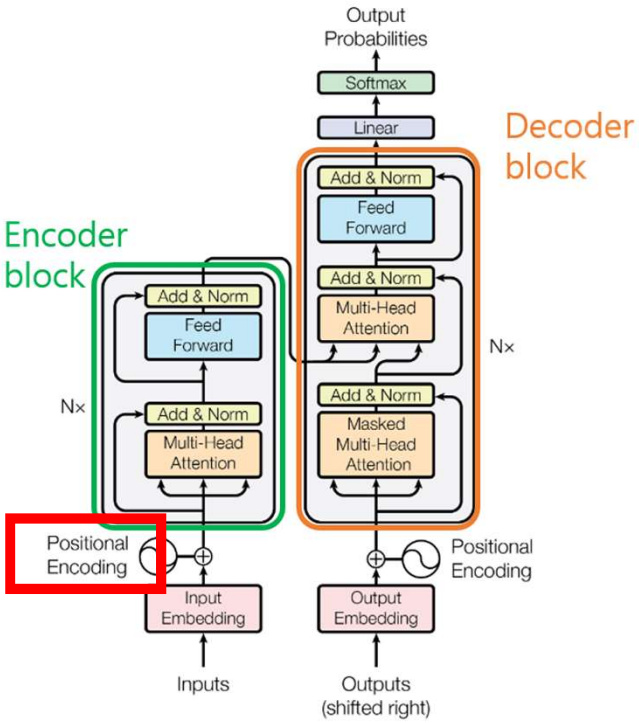
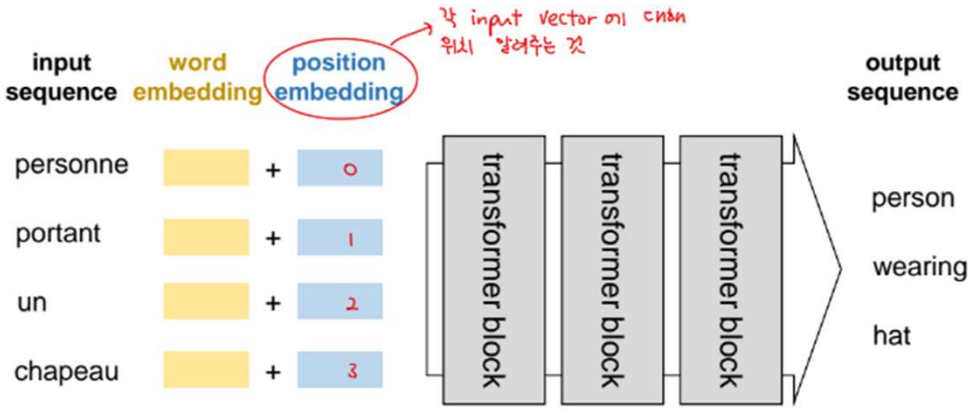


Figure 1: The Transformer - model architecture.

- Transformer 구조에는 순서를 반영할 만한 부분이 없음
  - 따라서 단어의 위치 정보가 손실될 수 있음
- 이를 보완하기 위해 input vector가 위치 정보를 갖도록 하는 부분



Transformer

- Transformer – 3. Multi-head attention,
  Residual connection & Normalization

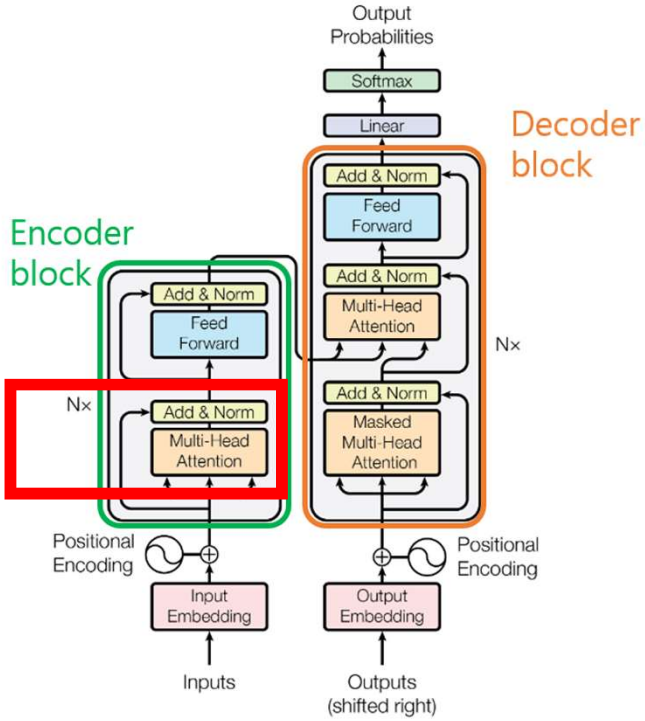


Figure 1: The Transformer - model architecture.

- Multi-head attention은 self attention을 동시에 여러 set에 대해 수행하는 것을 말함

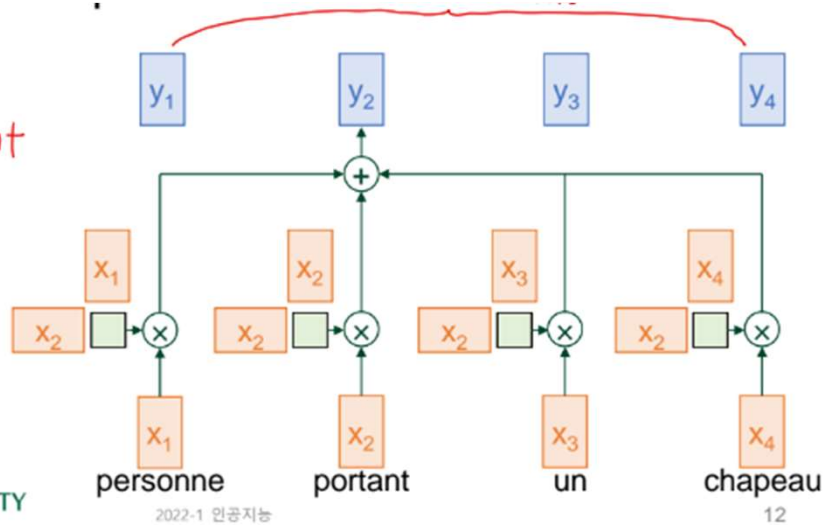
What is self attention? How is it done?

EWHA EURON

Transformer

- Transformer – 3. Multi-head attention → Self attention
  - Basic self attention
    - Input: 각 단어의 embedding이 모인 sequence
    - Output: input의 weighted sum이 모인 sequence

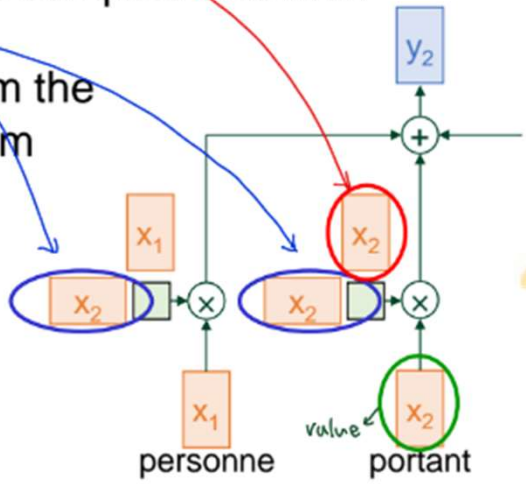
$$y_1, y_2, \dots, y_t$$
$$y_i = \sum_j w_{ij} x_j$$
$$w'_{ij} = x_i^T x_j \Rightarrow \text{weight}$$
$$w_{ij} = \frac{\exp w'_{ij}}{\sum_j \exp w'_{ij}}$$

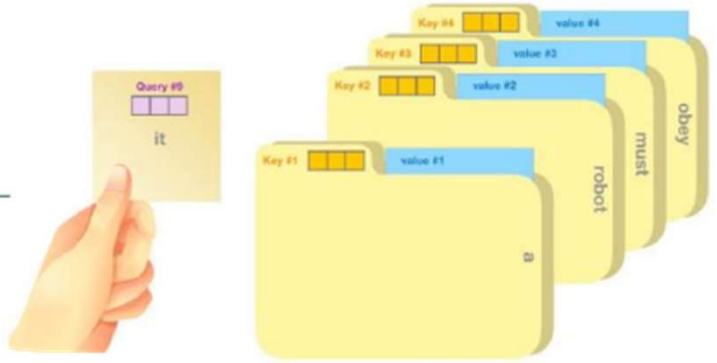


Transformer

- Transformer – 3. Multi-head attention → Self attention

  - Let's make it learnable
  - Every input vector x_i will be used in 3 ways:
    - Compared to every other vector to compute attention weights for its own output y_i (**query**)
    - Compared to every other vector to compute attention weights w_{ij} for output y_i (**key**)
    - Summed with other vectors to form the result of the attention weighted sum (**value**)





Transformer

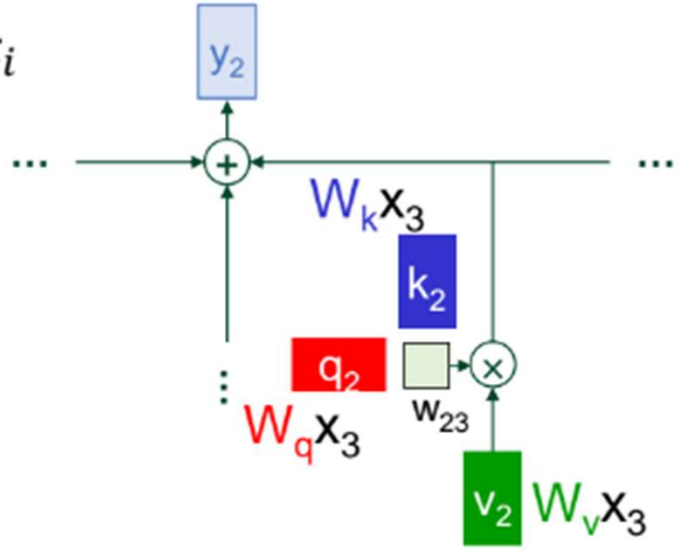
- Transformer – 3. Multi-head attention → Self attention
  - Learning the weight matrices = Learning attention

$$q_i = W_q x_i \quad k_i = W_k x_i \quad v_i = W_v x_i$$

$$w'_{ij} = q_i^T k_j$$

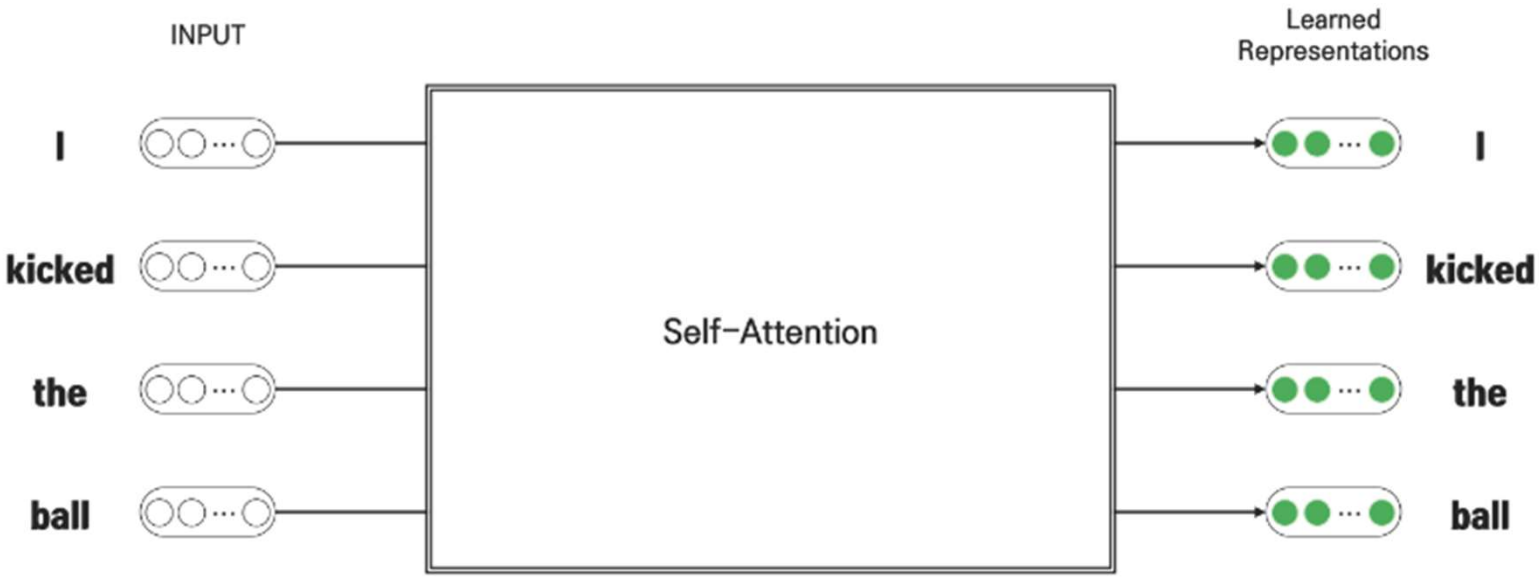
$$w_{ij} = softmax(w'_{ij})$$

$$y_i = \sum_j w_{ij} v_j$$



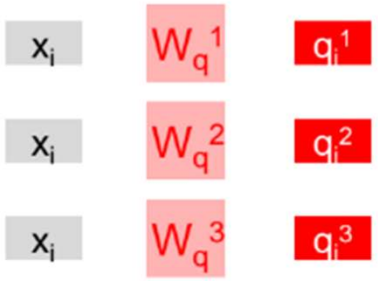
Transformer

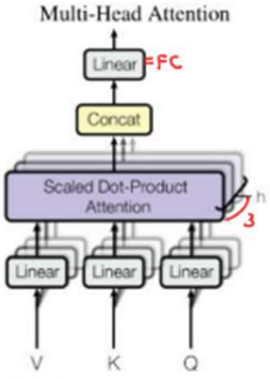
- Transformer – 3. Multi-head attention → Self attention
  - Self attention은 각 token을 sequence 내 모든 token과의 연관성을 기반으로 재표현하는 과정으로 해석 가능

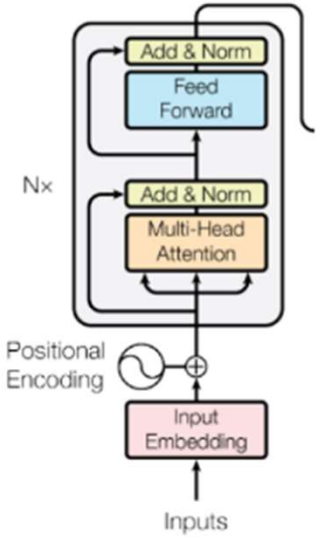


Transformer

- Transformer – 3. Multi-head attention
  - 한 문장 내에 존재하는 다양한 정보를 한 번의 attention만으로 반영하기 어려움
  - Multi-head attention 사용
    - Multiple heads of attention just means learning different set of W_q, W_k, W_v matrices simultaneously
    - Multi-head attentions are concatenated





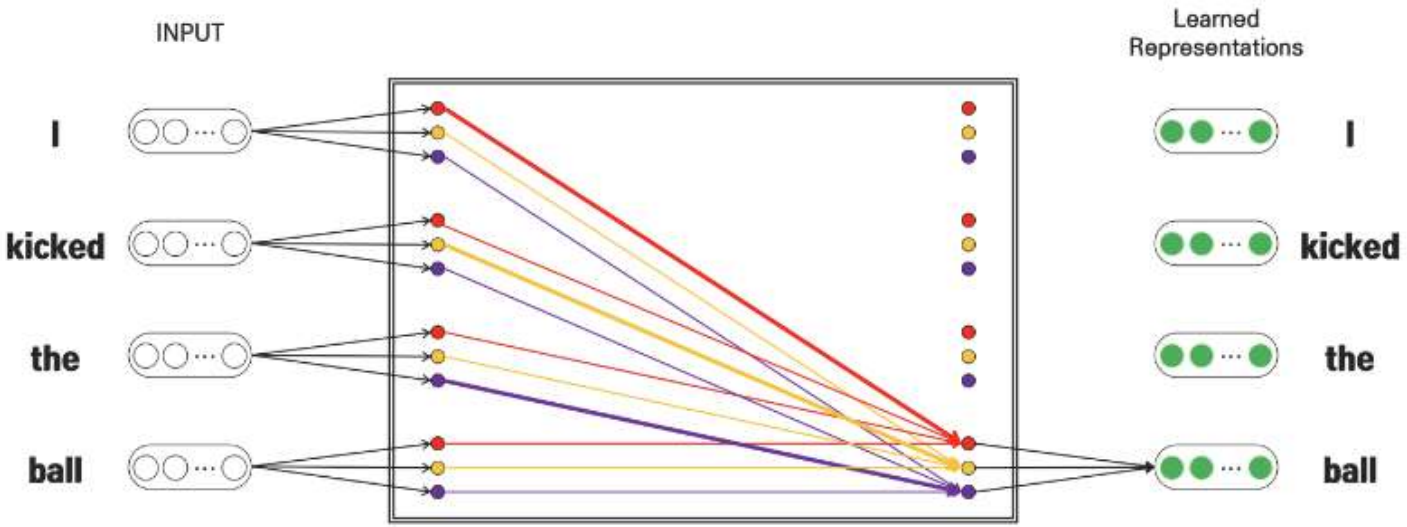


multi head attention을 쓰는 이유; 다양한 관점에서 attention을 수행하기 위함. 오차함을 줄여줌.

Transformer

- Transformer – 3. Multi-head attention
  - 한 문장 내에 존재하는 다양한 정보를 한 번의 attention만으로 반영하기 어려움
  - Multi-head attention 사용

- Red: Who?
- Yellow: Did what?
- Purple: Which?



EWHA EURON

Transformer

- Transformer – 3. Multi-head attention,
  Residual connection & Normalization

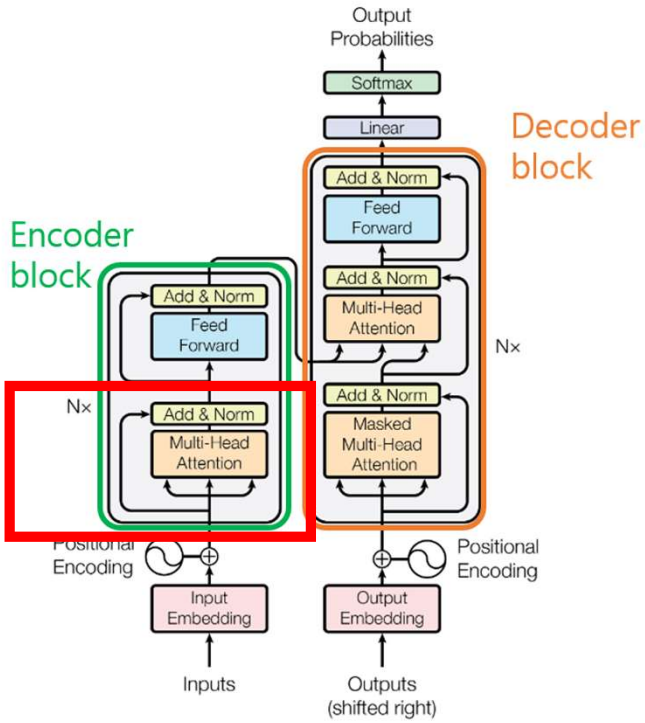
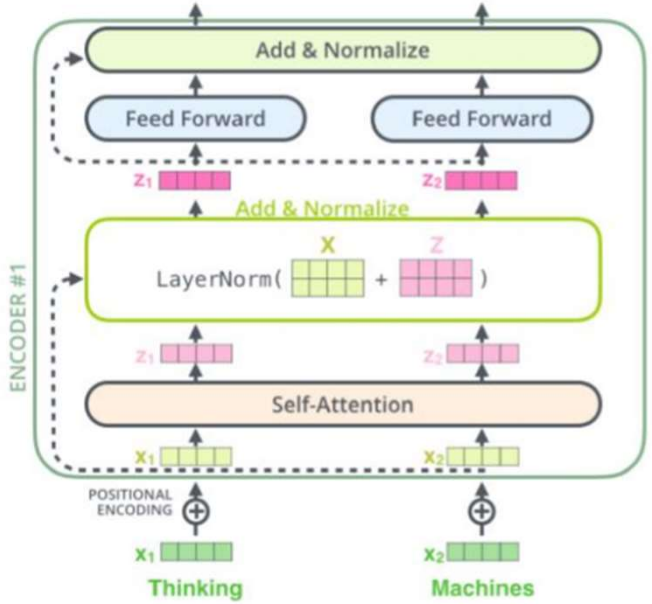


Figure 1: The Transformer - model architecture.

- Residual block: Backprop시 잃어버릴 수 있는 데이터 보존 위함
- Add & LayerNorm



Transformer

- Transformer – 4. Feed forward neural network

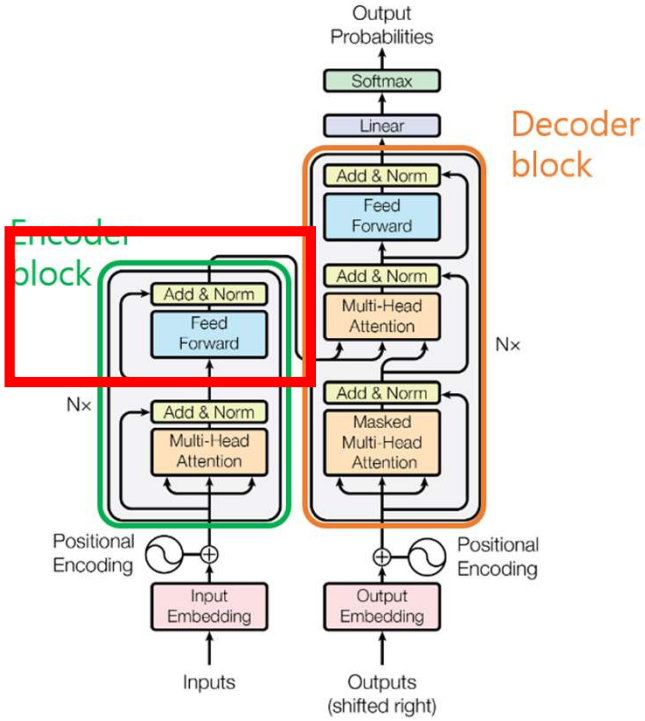


Figure 1: The Transformer - model architecture.

- Fully connected feed forward neural network

EWHA EURON

Transformer

- Transformer – 5. Masked multi-head attention

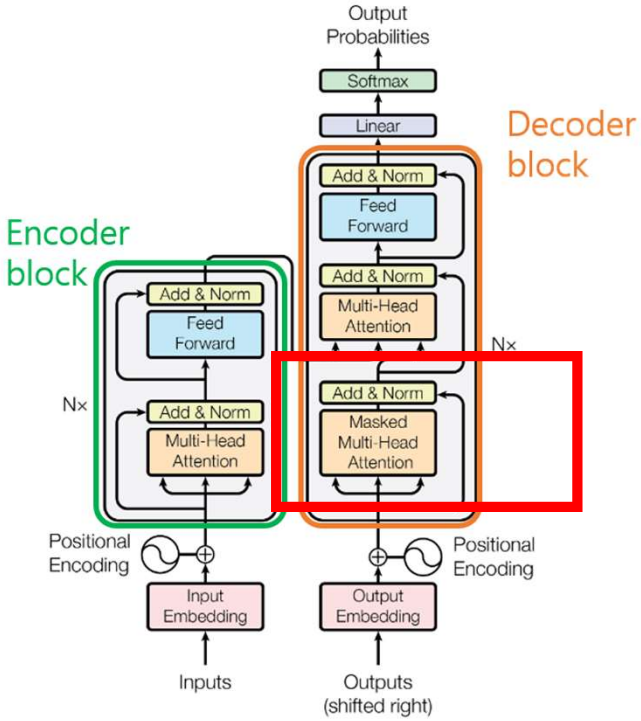
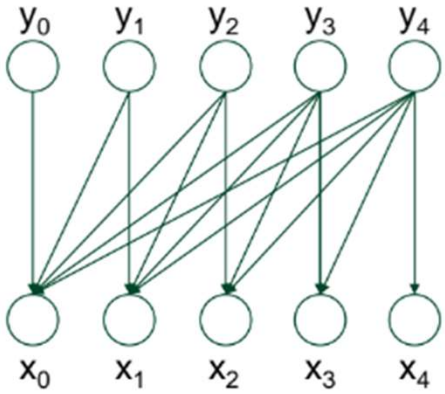


Figure 1: The Transformer - model architecture.

- 현재 attention은 현재와 과거 값만 볼 수 있도록 masking



Transformer

- Transformer – 6. Multi-head attention with encoder outputs

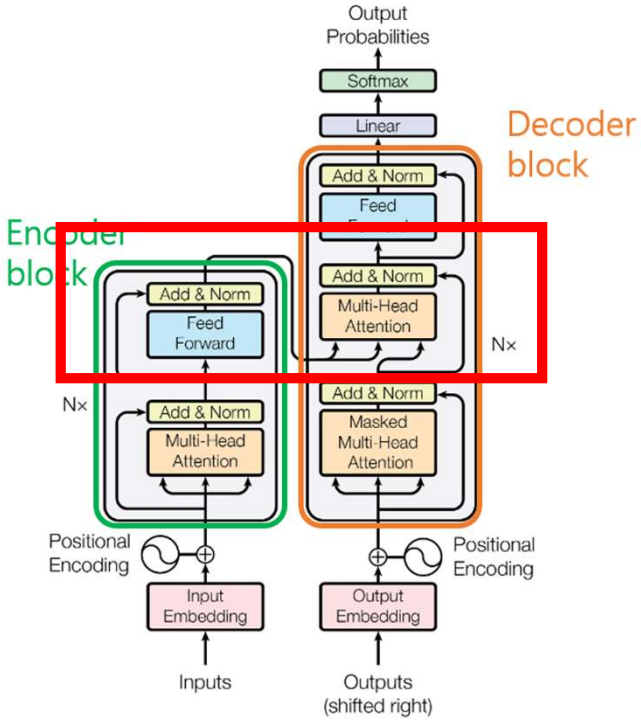
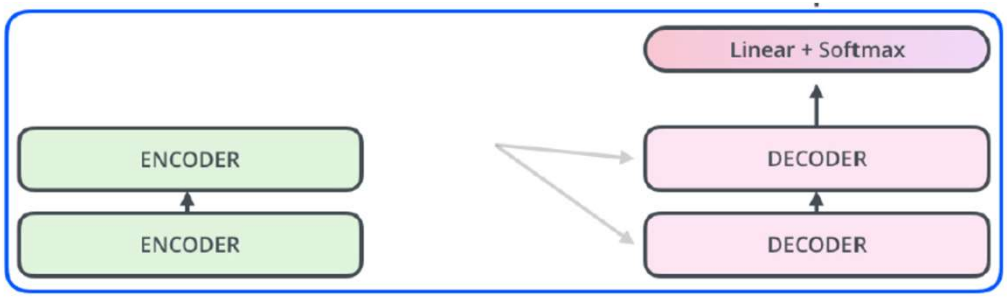


Figure 1: The Transformer - model architecture.

- Encoder output + decoder masked attention output
- Encoder output과 decoder input의 연관성 반영



Transformer

- Transformer – 7. The final linear and softmax layer

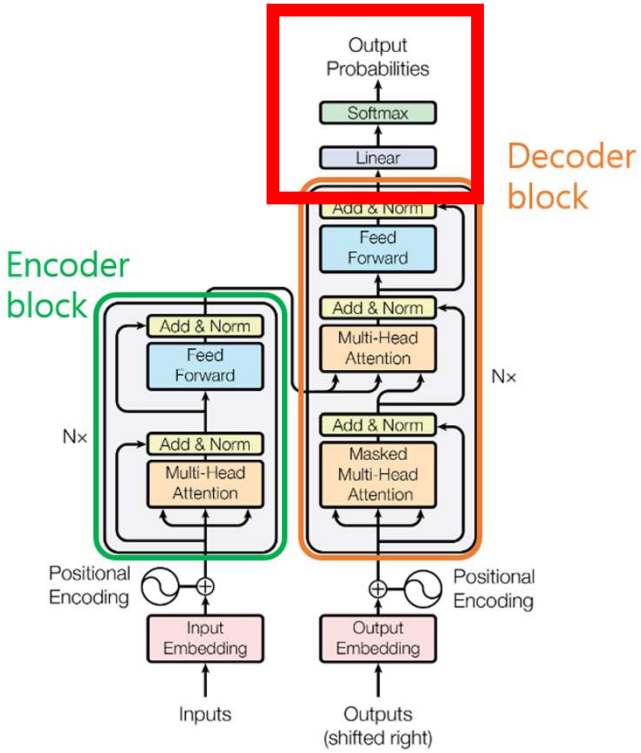


Figure 1: The Transformer - model architecture.

- Linear layer: fully connected neural network
- Softmax layer: scores into probability

EWHA EURON

Image transformer

EWHA EURON

Image transformer

1. Using Self-attention for Image tasks

   Self-Similarity in Image

Starry Night (Van Gogh, June 1889)

EWHA EURON

Image transformer

1. Using Self-attention for Image tasks

Image Transformer Tasks

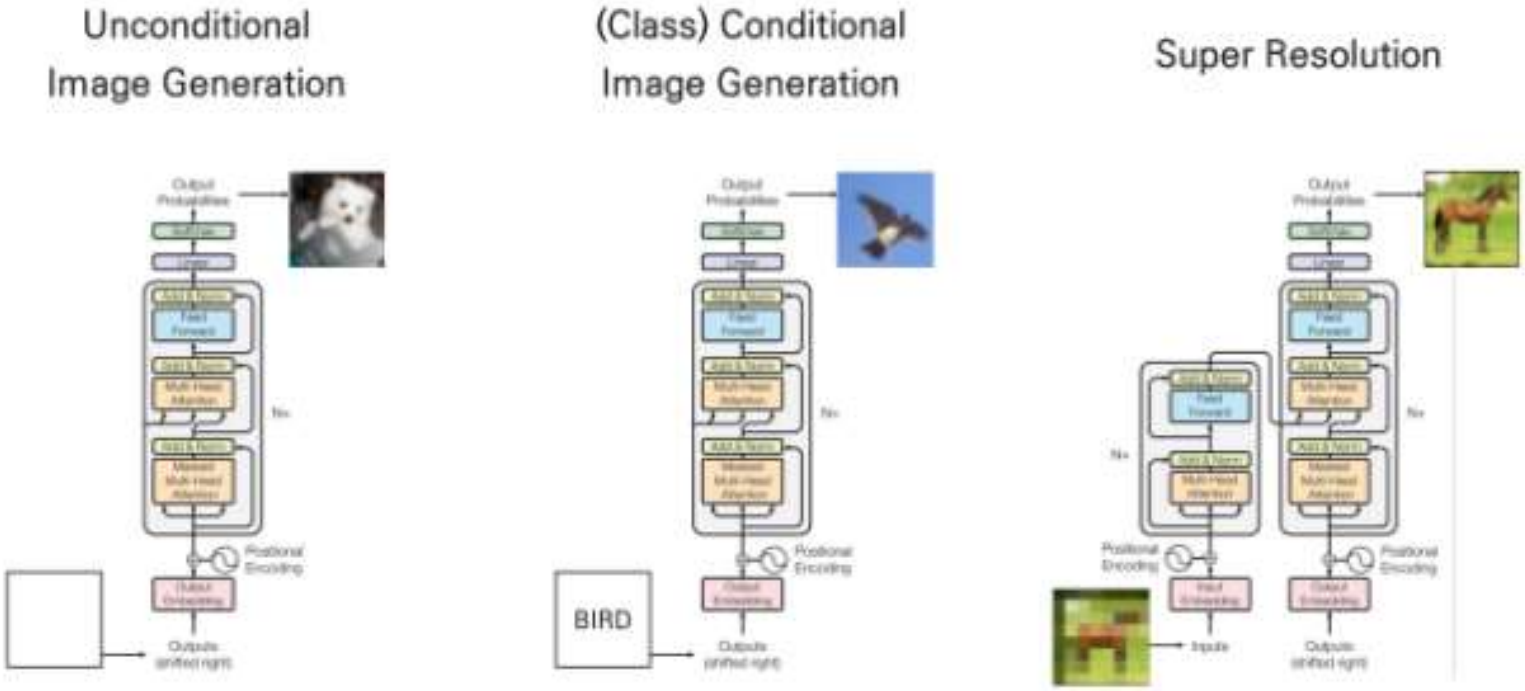


Image transformer

1. Using Self-attention for Image tasks

- Unconditional Image Generation
대규모의 데이터로 특정한 이미지를 제작하는 태스크

- Conditional Image Generation
클래스 각각의 임베딩 벡터를 입력으로 받거나, seed 이미지를 받아 이미지를 제작하는 태스크

- Super Resolution
저화질의 이미지를 입력으로 받아 고화질의 이미지를 출력하는 태스크

EWHA EURON

Image transformer

1. Using Self-attention for Image tasks

complexity

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	$O(k \cdot n \cdot d^2)$	$O(1)$	$O(\log_k(n))$

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

병렬화 가능

Long-term dependency 문제 해결

In Transformer, 입력: 문장 > 사진 / 처리 단위: 토큰 > 픽셀
Self-Attention: model dimension보다 sequence length가 작아 효율적

Sequence length = 픽셀을 나열한 크기 (일반적으로 32x32x3=3072)
-> self-attention을 적용하는 비용이 커짐

EWHA EURON

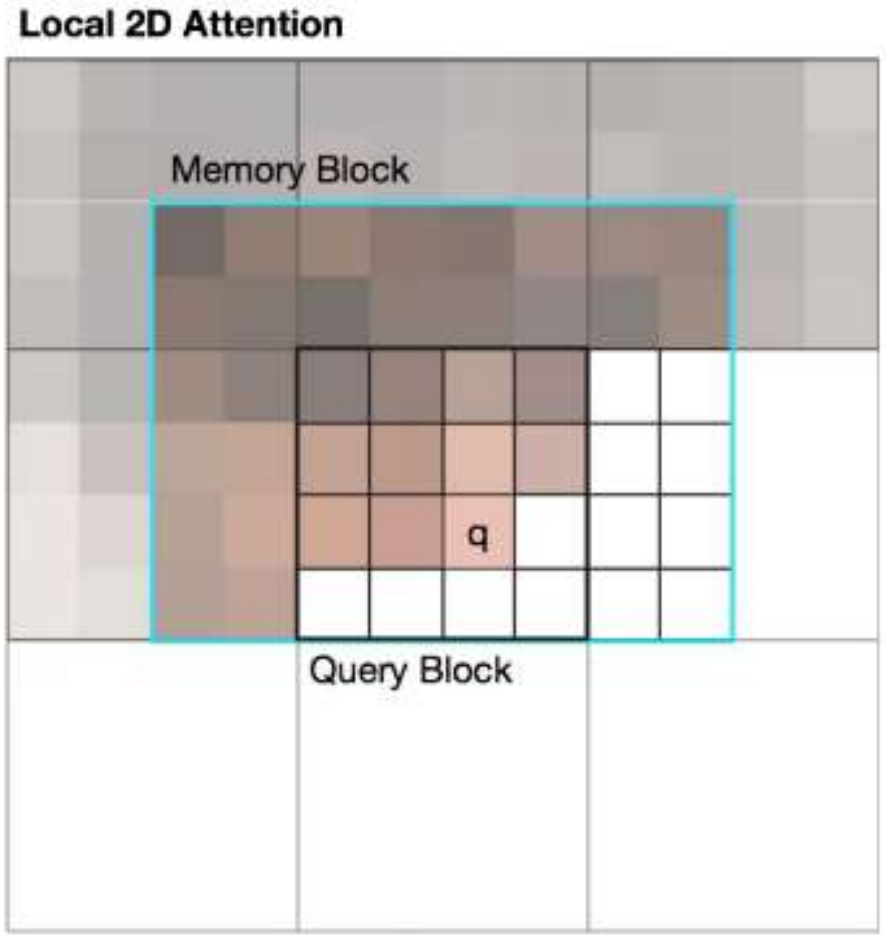
Image transformer

2. Local Self-Attention

- attention window 전체가 아닌 근처의 픽셀들로만 설정해 attention 수행

- Sequence 내 일정 부분: Memory block

- memory block 내에서만 self-attention 적용

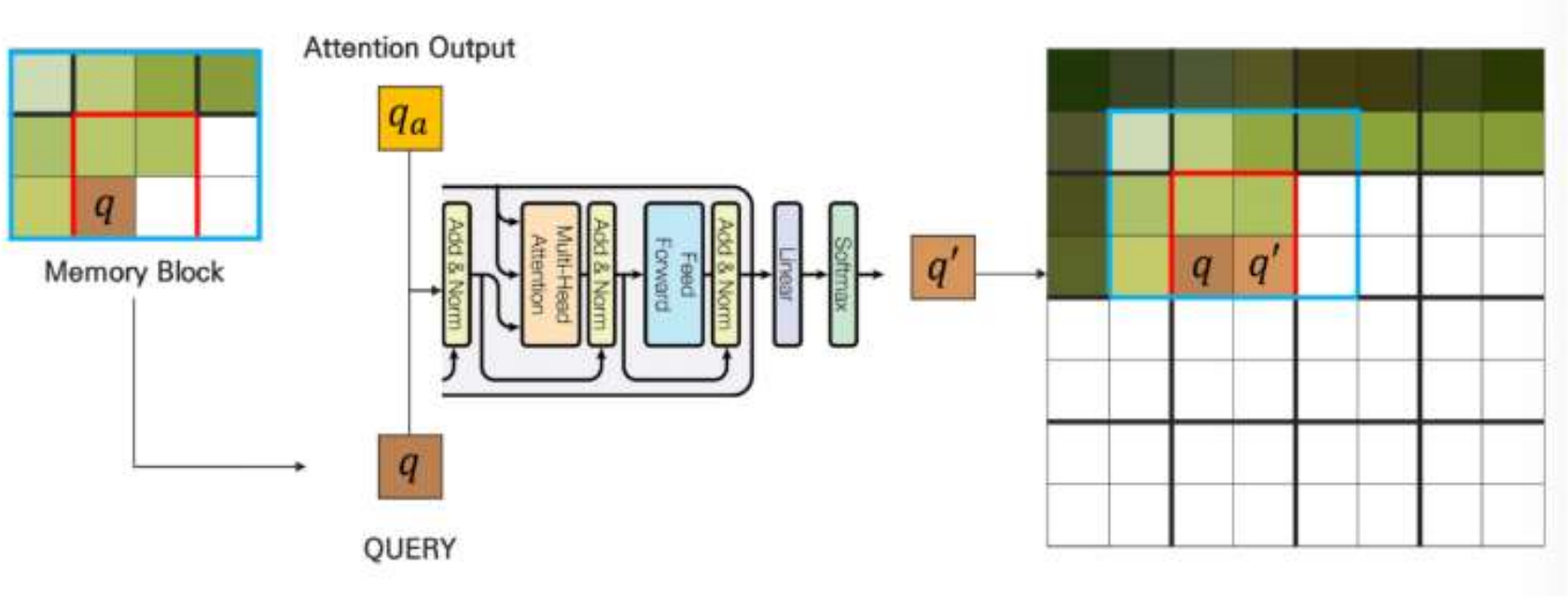


EWHA EURON

Image transformer

2. Local Self-Attention

Super Resolution에서 decoder의 이미지 생성 순서



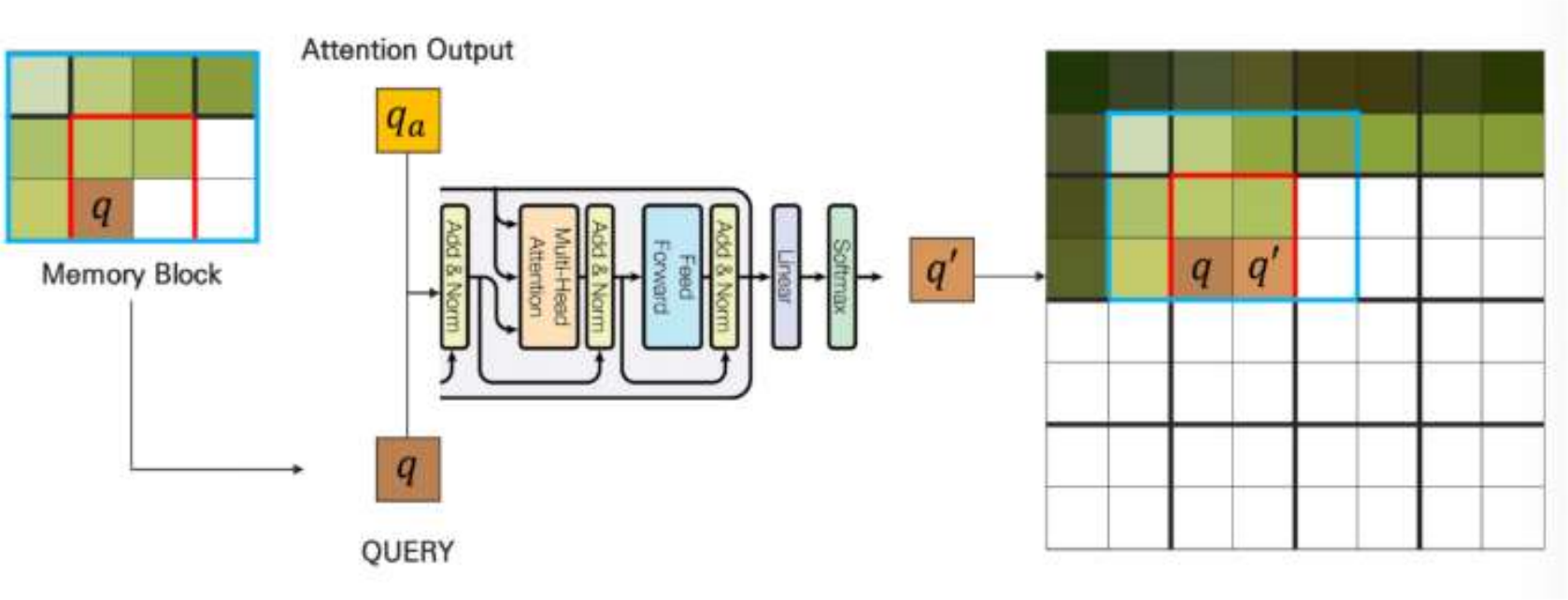
1. Input을 겹치지 않는 Block으로 구분, 마지막으로 생성된 픽셀을 포함하는 block을 Query block이라고 한다. 이때 마지막으로 생성된 픽셀을 Current Query pixel, 그 다음 생성되야 할 픽셀을 Target pixel이라고 한다.

EWHA EURON

Image transformer

2. Local Self-Attention

Super Resolution에서 decoder의 이미지 생성 순서

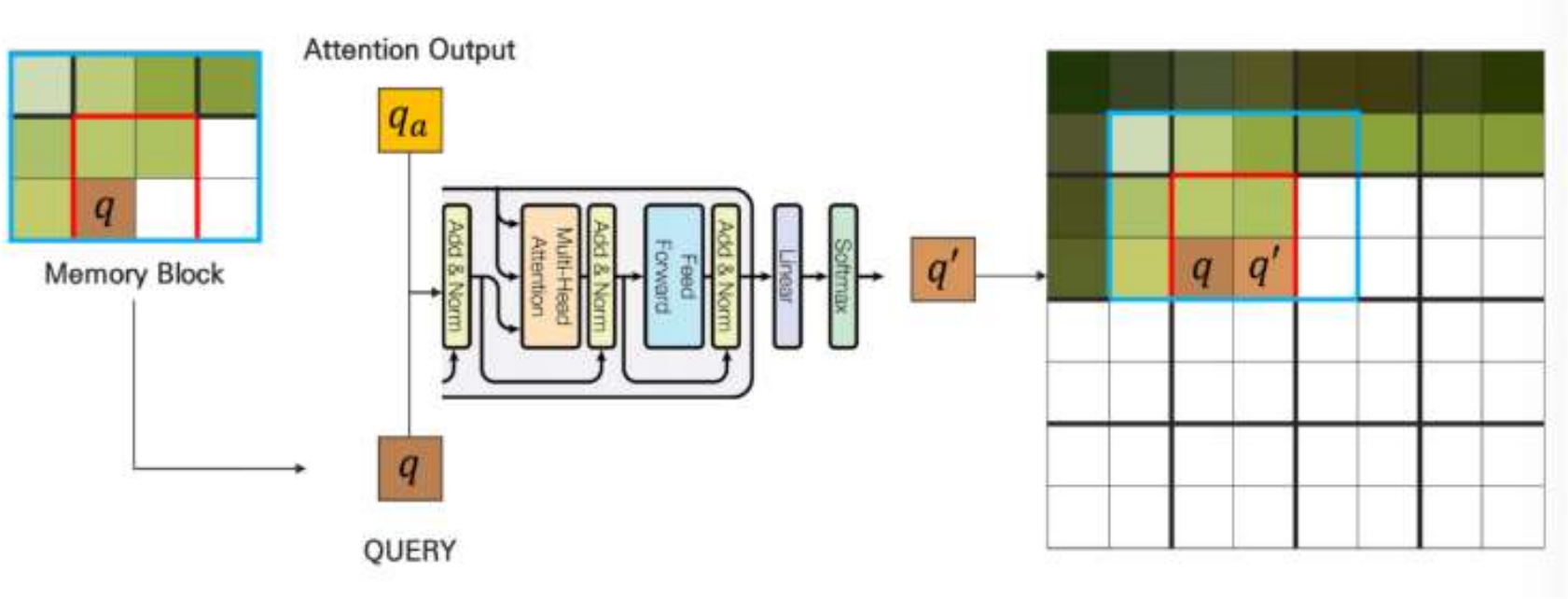


2. 위쪽 방향으로 h_m 픽셀, 양 옆으로 w_m 픽셀만큼 둘러싸는 Memory block을 지정한다.
이는 Key와 Value의 역할을 한다.

Image transformer

2. Local Self-Attention

Super Resolution에서 decoder의 이미지 생성 순서



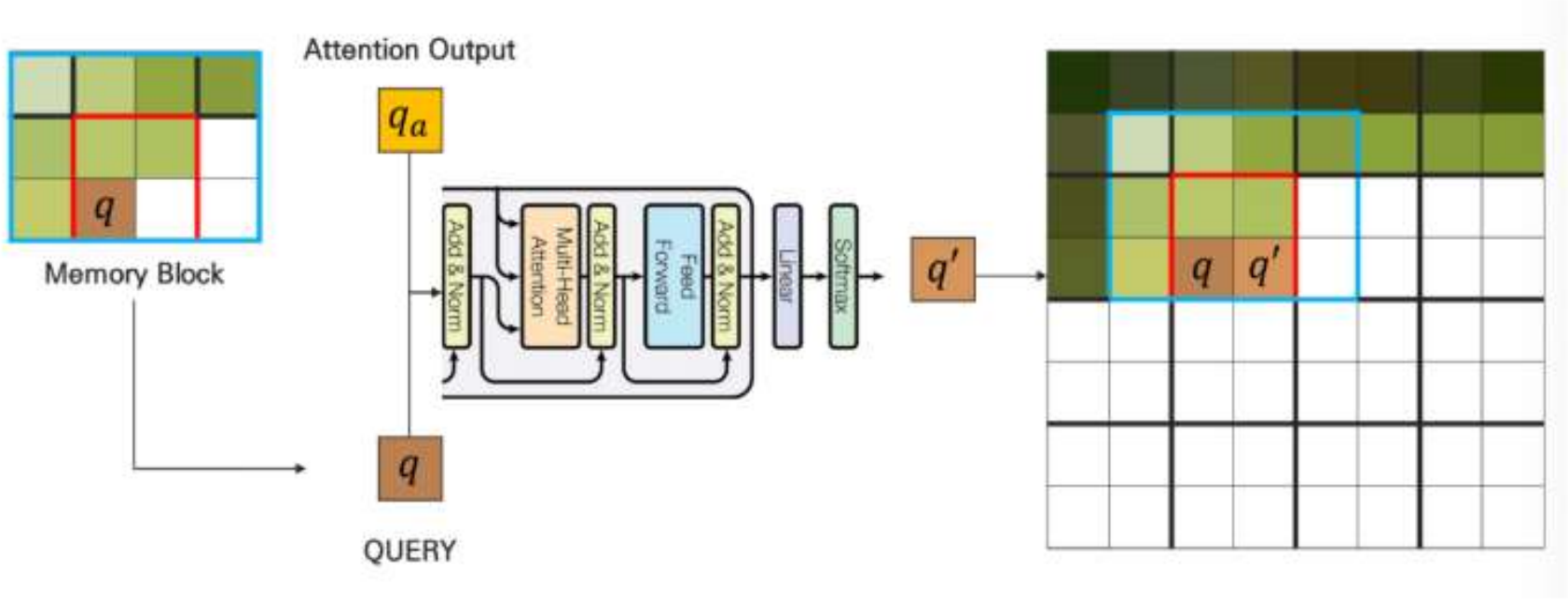
3. Memory block 내 픽셀을 key, value로, Current Query pixel을 query로 하는 self-attention을 수행한다. (Transformer Decoder의 Multihead self attention)

EWHA EURON

Image transformer

2. Local Self-Attention

Super Resolution에서 decoder의 이미지 생성 순서



4. Encoder-Decoder Attention, FFNN을 거쳐 output을 생성한다.

Image transformer

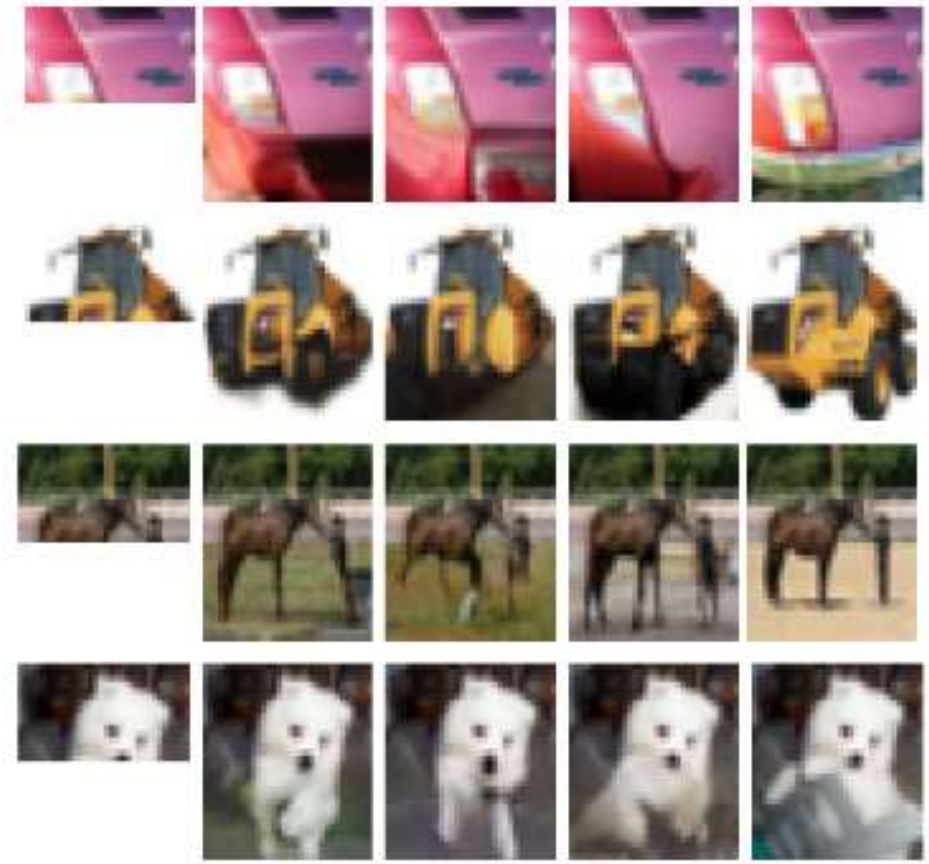
3. Results

Super Resolution

Image transformer

3. Results

Conditional Image Completion



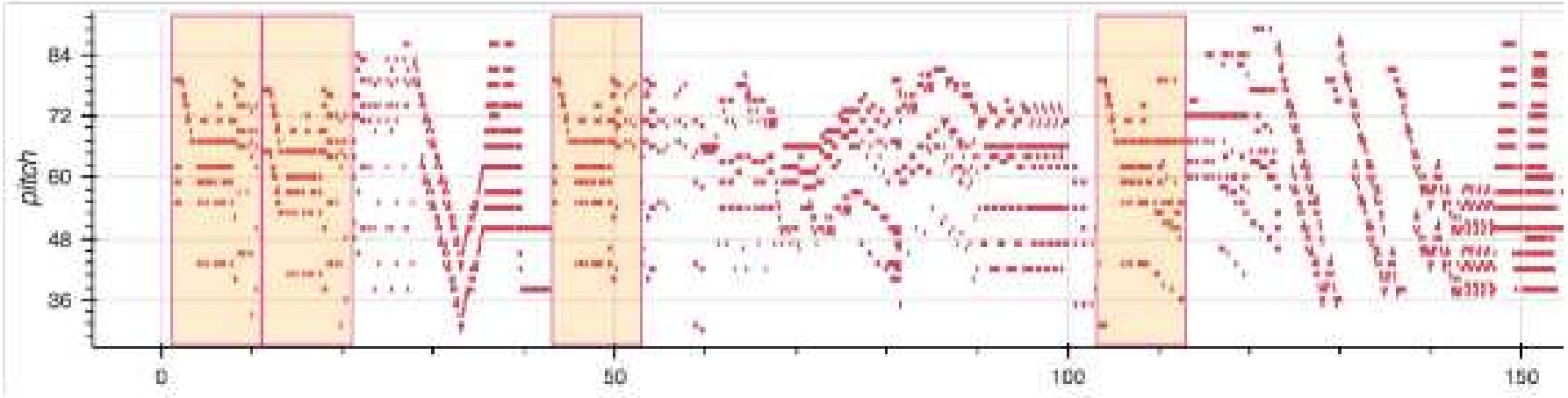
Music transformer

EWHA EURON

Music transformer

1. Using Self-attention for Music generation tasks

   Self-Similarity

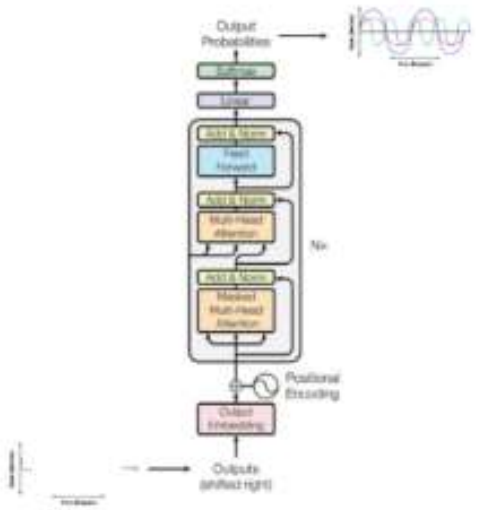


Music transformer

1. Using Self-attention for Music generation tasks

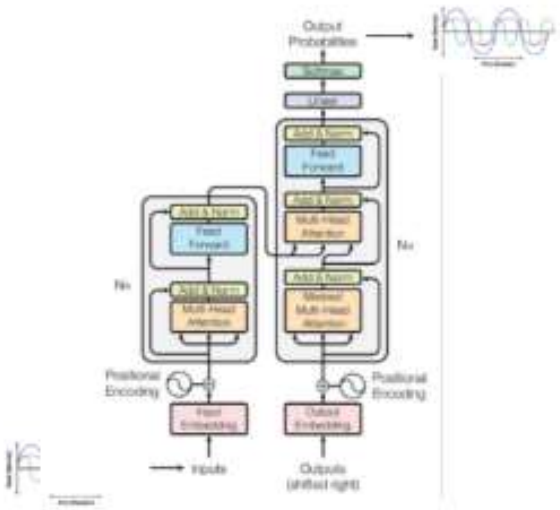
Music Transformer Tasks

Unconditional
Music Generation



대규모의 데이터로 특정한
음악을 제작하는 태스크

Conditional
Music Generation

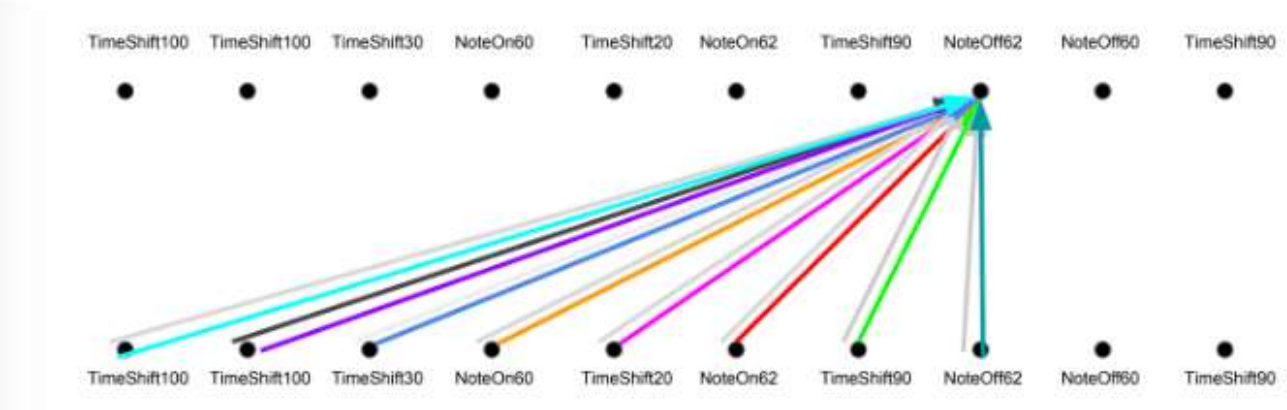


클래스 각각의 임베딩 벡터를
입력으로 받거나, seed 음악을
받아 음악을 제작하는 태스크

EWHA
EURON

Music transformer

2. Relative Positional Self-Attention



self attention

- 지난 정보들에 대한 weighted average를 알 수 있다.
- 어떤 토큰이든 직접적인 접근을 할 수 있다.
- 모든 토큰이 bag of words처럼 여겨져 한 토큰과 다른 토큰의 거리를 알 수 없다.

+

convolution

이동하는 고정된 크기의 필터가 각 토큰 사이 상대적인 거리를 잡아낸다.

EWHA EURON

Music transformer

2. Relative Positional Self-Attention

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

Music Transformer의 Relative Positional Self-Attention은
각 단위의 내용 뿐만 아니라 주기성을 고려한다.

단순히 일반적인 Self-attention에 Relative Positional Vector를 더해
query와 key의 sequence 내 거리를 attention weight에 반영하는 형태다.

EWHA EURON

Music transformer

2. Relative Positional Self-Attention

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

Music Transformer의 Relative Positional Self-Attention은
각 단위의 내용 뿐만 아니라 주기성을 고려한다.

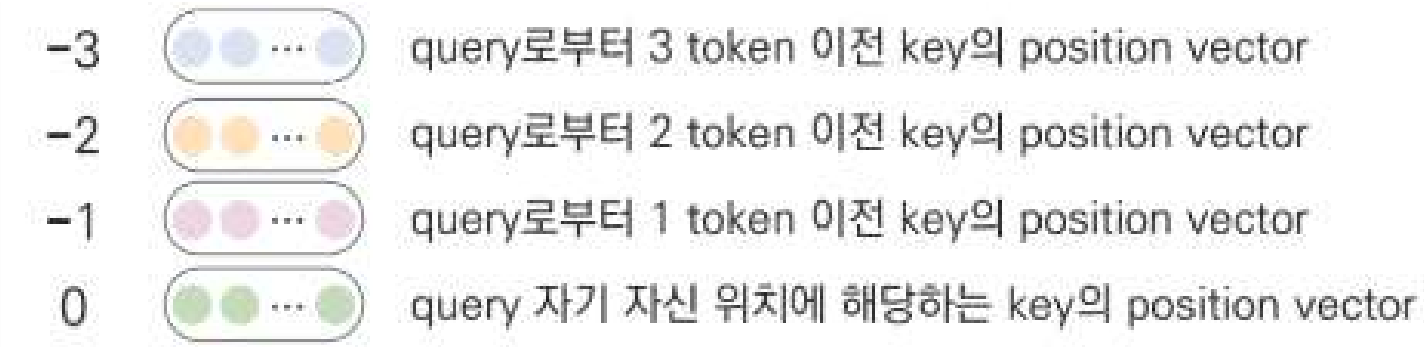
단순히 일반적인 Self-attention에 Relative Positional Vector를 더해
query와 key의 sequence 내 거리를 attention weight에 반영하는 형태이다.

EWHA EURON

Music transformer

2. Relative Positional Self-Attention

S^{rel} 만드는 순서



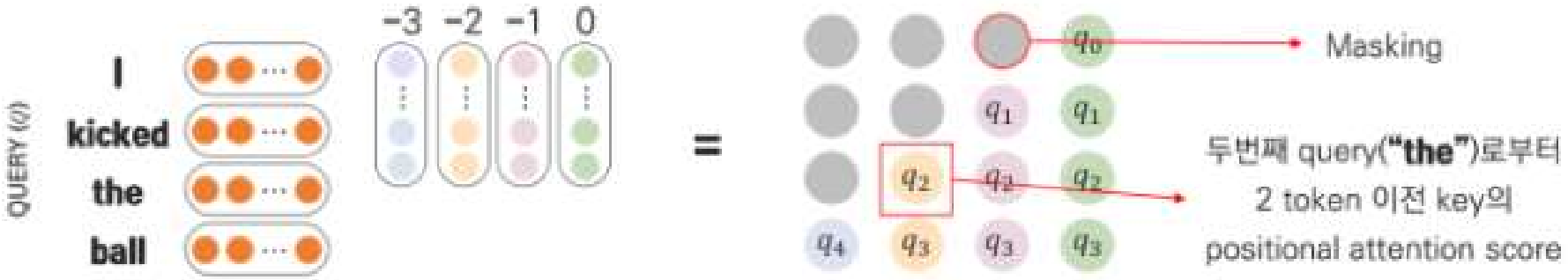
1. Relative Positon Embedding Matrix (E^T) 를 만든다.

EWHA EURON

Music transformer

2. Relative Positional Self-Attention

S^{rel} 만드는 순서

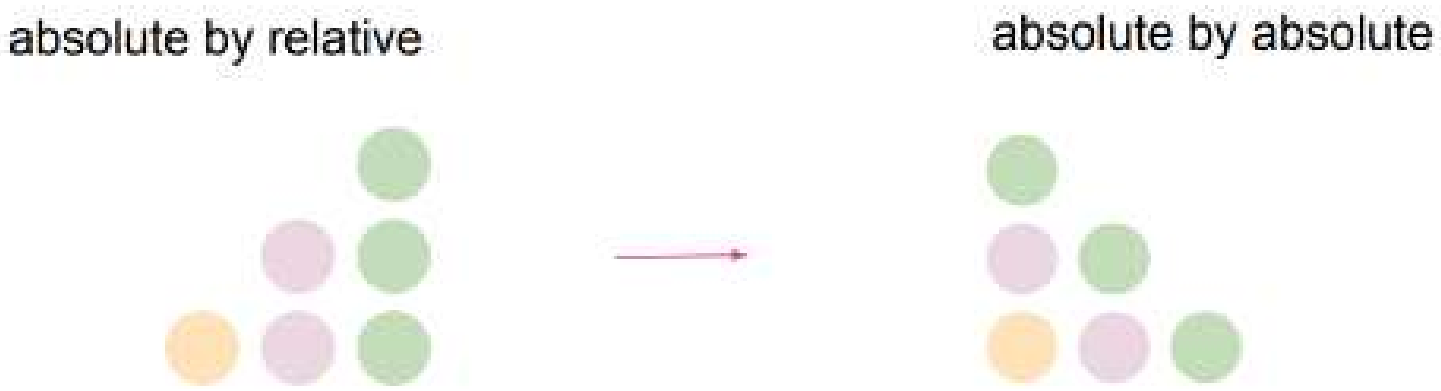


2. Relative Positon Embedding Matrix (E^T) 에 Query Vector를 곱한다.

Music transformer

2. Relative Positional Self-Attention

S^{rel} 만드는 순서



3. 기존 attention과 더할 수 있도록 모양을 변형한다. (Skewing)

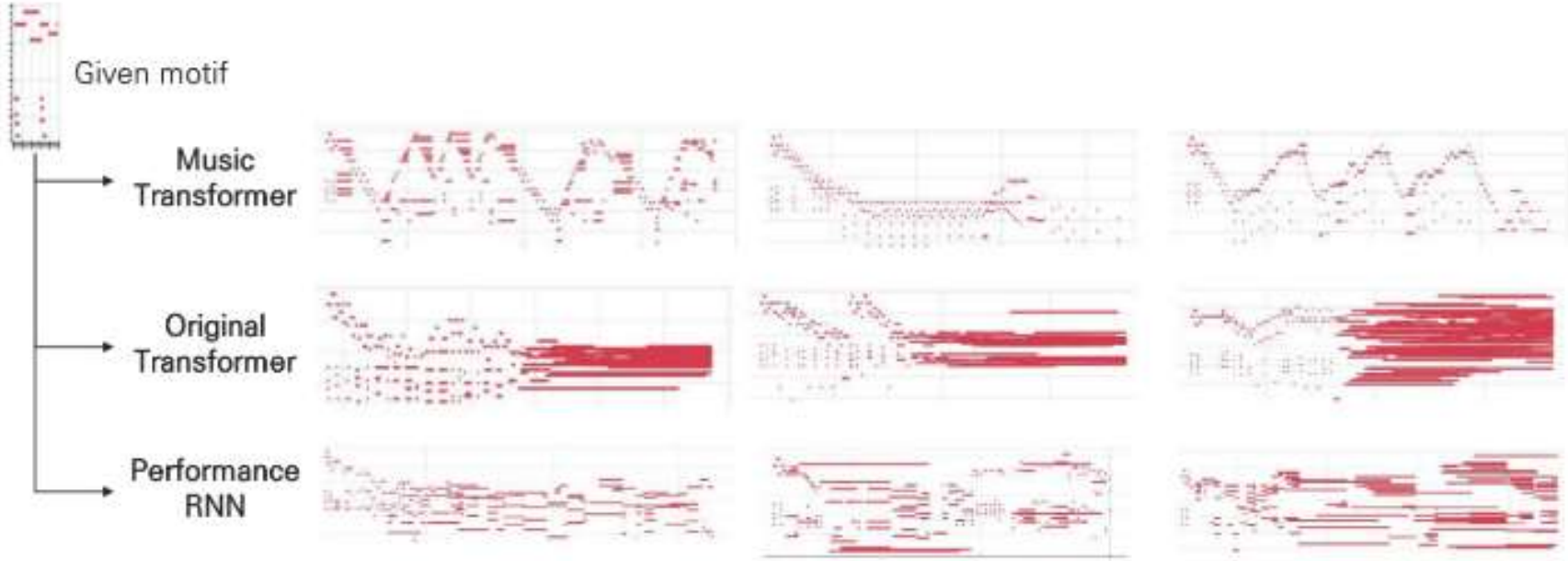
4. 기존 attention score와 relative positional attention score를 더하여 output을 산출한다.

EWHA EURON

Music transformer

2. Relative Positional Self-Attention

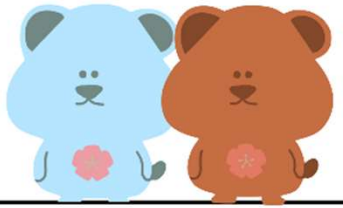
Results



1. 다양하고 반복적인 곡 생성
2. training data보다 2배 긴 sequence에 대한 generation 가능

EWHA EURON

THANK YOU



EWHA EURON