NeurIPS 2017

Attention Is All You Need

- Authors: *Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, Illia Polosukhin*
- Main Affiliation: Google Brain

2025.07.22

서정호

DGIST DATA SCIENCE LAB

대구경북과학기술원
Daegu Gyeongbuk Institute of Science & Technology

Contents

1. Proposal

NLP → Sequence transduction model.

- Sequence → Sequence 변환
- 기존 (1): RNN or CNN 사용. using encoder & decoder
- 기존 (2): Best model: Attention mechanism 사용해서 encoder - decoder 연결

Transformer architecture 제안.

- RNN[LSTM, GRU], CNN 사용 X
- 오직 attention mechanisms [specifically, self-attention]만 사용.
- 기존: encoder-decoder architectures를 사용한 recurrent layers
  → transformer: multi-headed self-attention
- Experiments 결과 → transformer: 병렬화 up & 학습시간 down.
- 평가척도: BLEU score, 2014 WMT [Workshop on Machine Translation]에서 성과 얻음.
  - English to German, English to French
- GPU도 적게 사용!!.

2. Model Architecture

▼ Sequence transduction models

encoder-decoder structure

- Encoder: [input] $\mathbf{x} \rightarrow \mathbf{z}$ [output]
    - $\mathbf{x} = (x_1, \ldots, x_n)$: Symbol sequence
    - $\mathbf{z} = (z_1, \ldots, z_n)$: Continuous sequence
- Decoder: [input] $\mathbf{z} \rightarrow \mathbf{y}$ [output]
    - $\mathbf{z} = (z_1, \ldots, z_n)$: Continuous sequence
    - $\mathbf{y} = (y_1, \ldots, y_m)$: Symbol sequence

2. Model Architecture

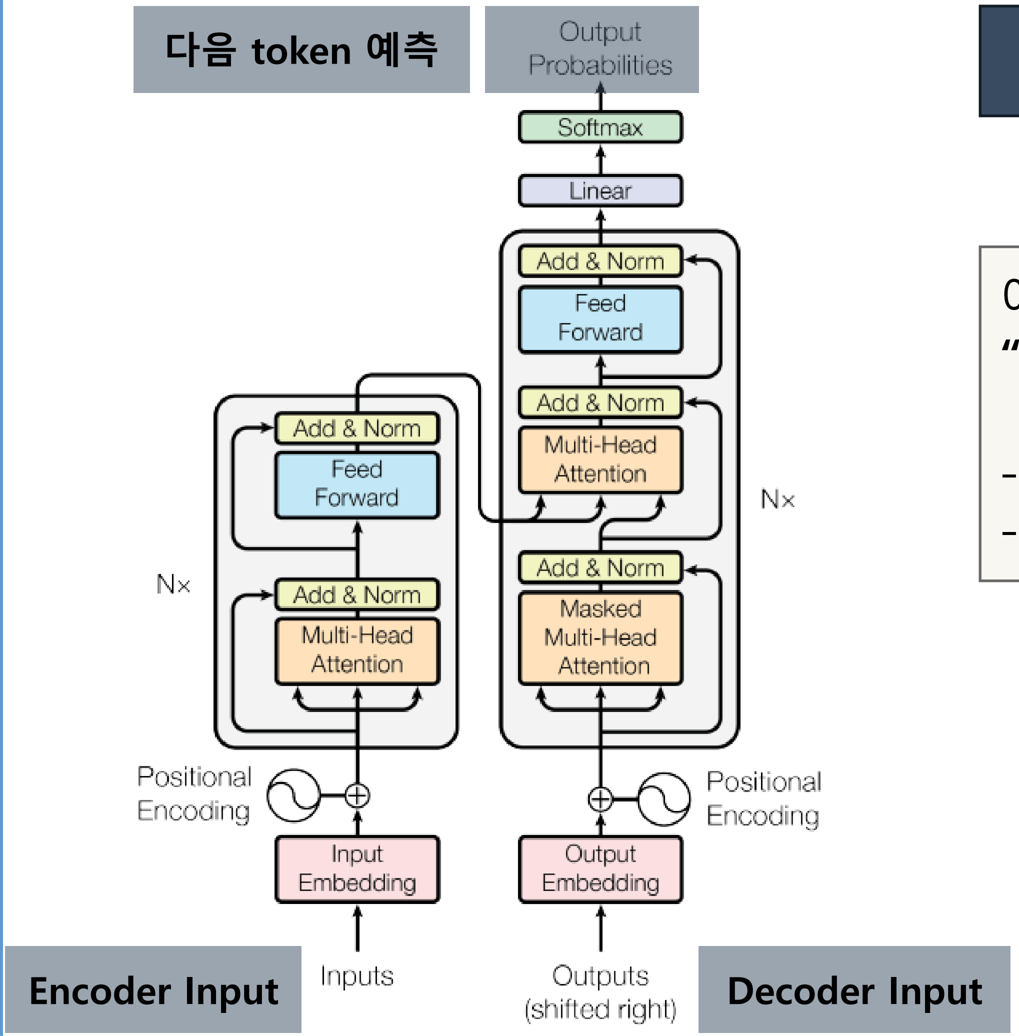


Figure 1: The Transformer - model architecture.

Total Architecture

예. Eng. -> French.
"I am a student" -> "Je suis étudiant »

- Input token: **[I, am, a, student]**
- Target token: **[Je, suis, étudiant]**

2. Model Architecture

Learning

예. Eng. -> French.
"I am a student" -> "Je suis étudiant »

- Input token: **[I, am, a, student]**
- Target token: **[Je, suis, étudiant]**

입력

- **Encoder input**: `[I, am, a, student]`
- **Decoder input** (shifted right): `[<s>, Je, suis]`
- **Target** (예측 정답): `[Je, suis, étudiant]`

학습 목표

각 시점 t에서 다음을 예측하도록 학습:

maximized

Loss ftn.

Step t		Decoder Input	Target Token	모델이 학습할 조건부 확률
1	시작 token	`[<s>]`	`Je`	$P(\text{Je} \mid <s>, x)$
2		`[<s>, Je]`	`suis`	$P(\text{suis} \mid <s>, Je, x)$
3		`[<s>, Je, suis]`	`étudiant`	$P(\text{étudiant} \mid <s>, Je, suis, x)$

DGIST DATA SCIENCE LAB

2. Model Architecture

Inference

예. Eng. -> French.
"I am a student" -> "Je suis étudiant »

- Input token: **[I, am, a, student]**
- Target token: **[Je, suis, étudiant]**

입력

- **Encoder input**: `[I, am, a, student]`
- **Decoder 시작**: `[<s>]` → 이후 토큰은 모델이 생성

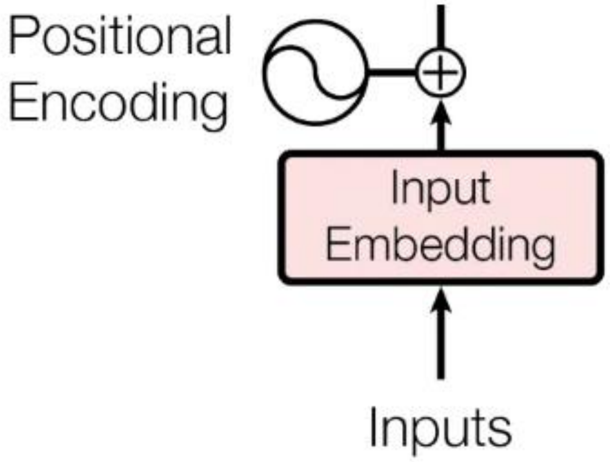
디코딩 과정 (Greedy 예시)

Step t	Decoder Input	Softmax Output (예시)	선택된 토큰
1	`[<s>]`	`{"Je": 0.8, "Tu": 0.1, ...}`	`Je`
2	`[<s>, Je]`	`{"suis": 0.9, "vais": 0.05, ...}`	`suis`
3	`[<s>, Je, suis]`	`{"étudiant": 0.95, ...}`	`étudiant`
4	`[<s>, Je, suis, étudiant]`	`{"</s>": 0.99, ...}`	`</s>` → 종료

종료 token

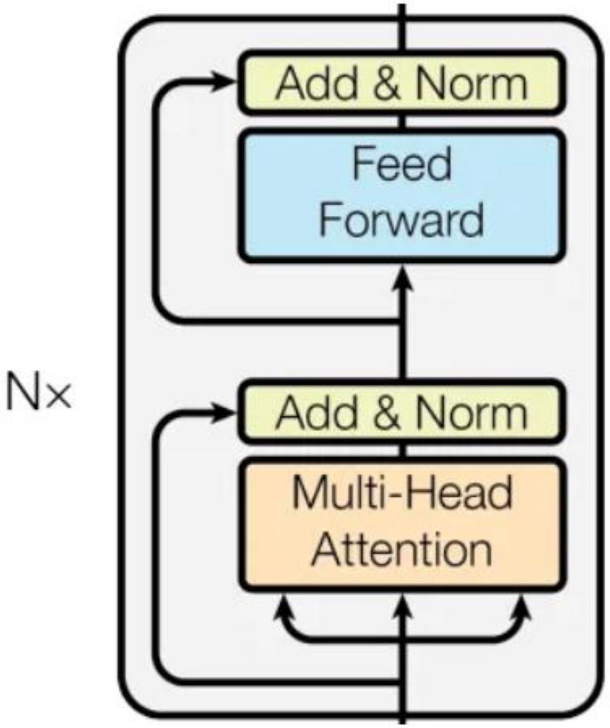
2. Model Architecture

▾ 1st.



- Inputs == input tokens
- tokens → embedding vector $\in \mathbb{R}^{d_{\text{model}}}$
- embedding vector + positional encoding: Input.

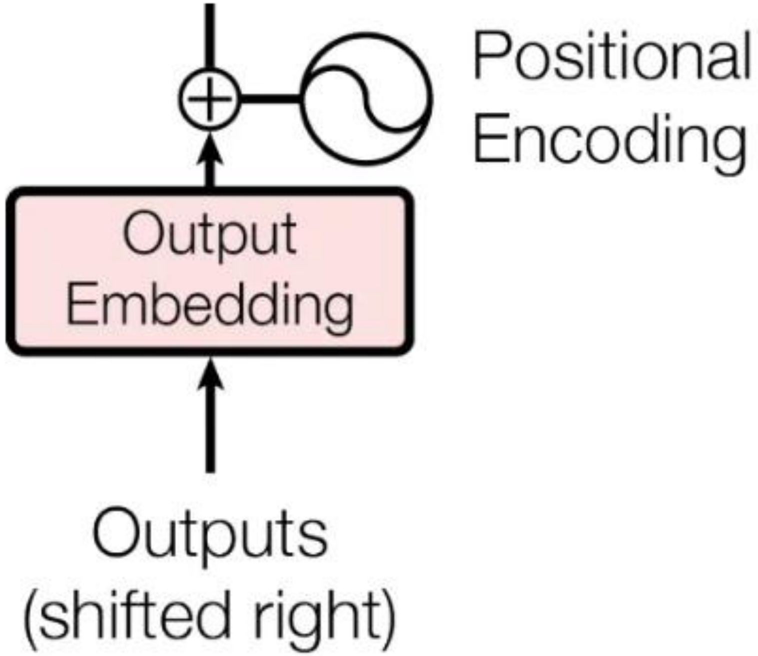
▾ 2nd. == Encoder



- N = 6개의 identical layers, input → 1st layer → 2nd layer → ... → N-th layer → output
- each layer == two sub layers
  1. [Multi-Head Attention] Multi-Head self-Attention mechanism
  2. [Feed Forward] position-wise fully connected Feed-Forward Network [FFN]
  - [Add & Norm] $\mathrm{LayerNorm}\,()$ function
- Sub-layers' outputs
  - $\mathrm{LayerNorm}\,(\mathbf{x} + \mathrm{Sublayer}\,(\mathbf{x}))$, where $\mathbf{x}$: input of the each sub-layer

2. Model Architecture

▾ 3rd.

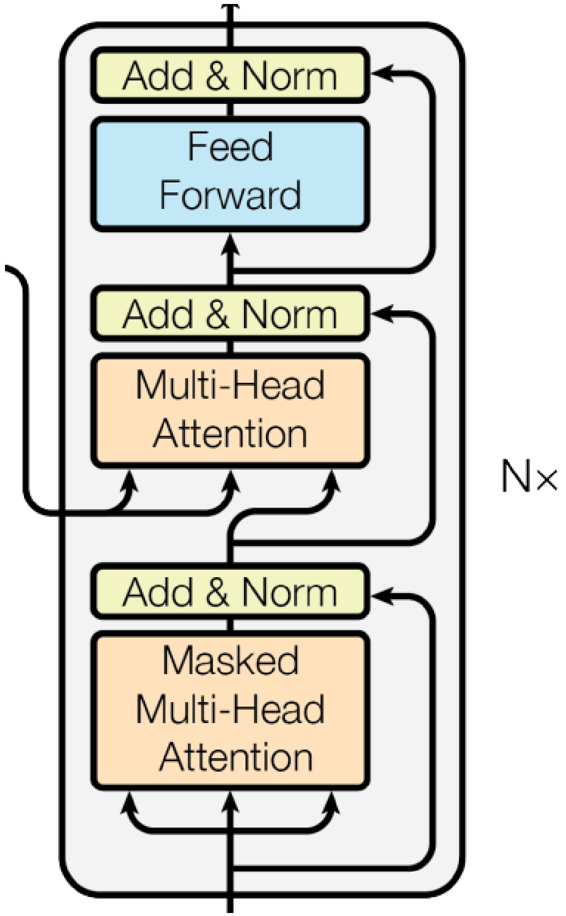


- Outputs == output tokens
- tokens → embedding vector $\in \mathbb{R}^{d_{\text{model}}}$
- embedding vector + positional encoding: Output.

2. Model Architecture

▼ 4th. == Decoder

- N = 6개의 identical layers, input → 1st layer → 2nd layer → … → N-th layer → output
- each layer == three sub layers
  1. **[Masked Multi-Head Attention]**
     - **Masking**: i의 position이 i보다 작은 positions의 outputs에만 의존함을 보장.
     - 1, 2, 3, …, i-1 → i
  2. [Multi-Head Attention] Multi-Head self-Attention mechanism
     - 주의: Encoder output에 multi-head attention 적용.
  3. [Feed Forward] position-wise fully connected Feed-Forward Network [FFN]
  - [Add & Norm] $\mathrm{LayerNorm}\,()$ function
- Sub-layers' outputs
  - $\mathrm{LayerNorm}\,(\mathbf{x} + \mathrm{Sublayer}\,(\mathbf{x}))$, where $\mathbf{x}$: input of the each sub-layer

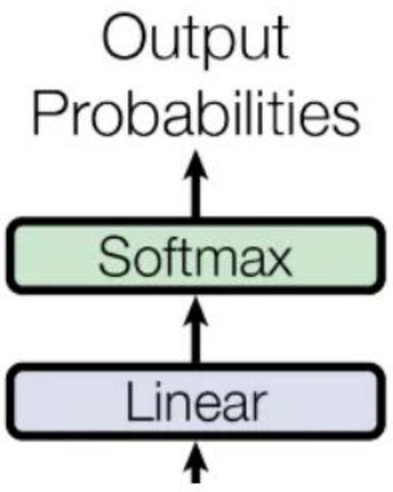


DGIST DATA SCIENCE LAB

DGIST 대구경북과학기술원 Daegu Gyeongbuk Institute of Science & Technology

2. Model Architecture

▾ 5th. == 3.4.



Total Architecture

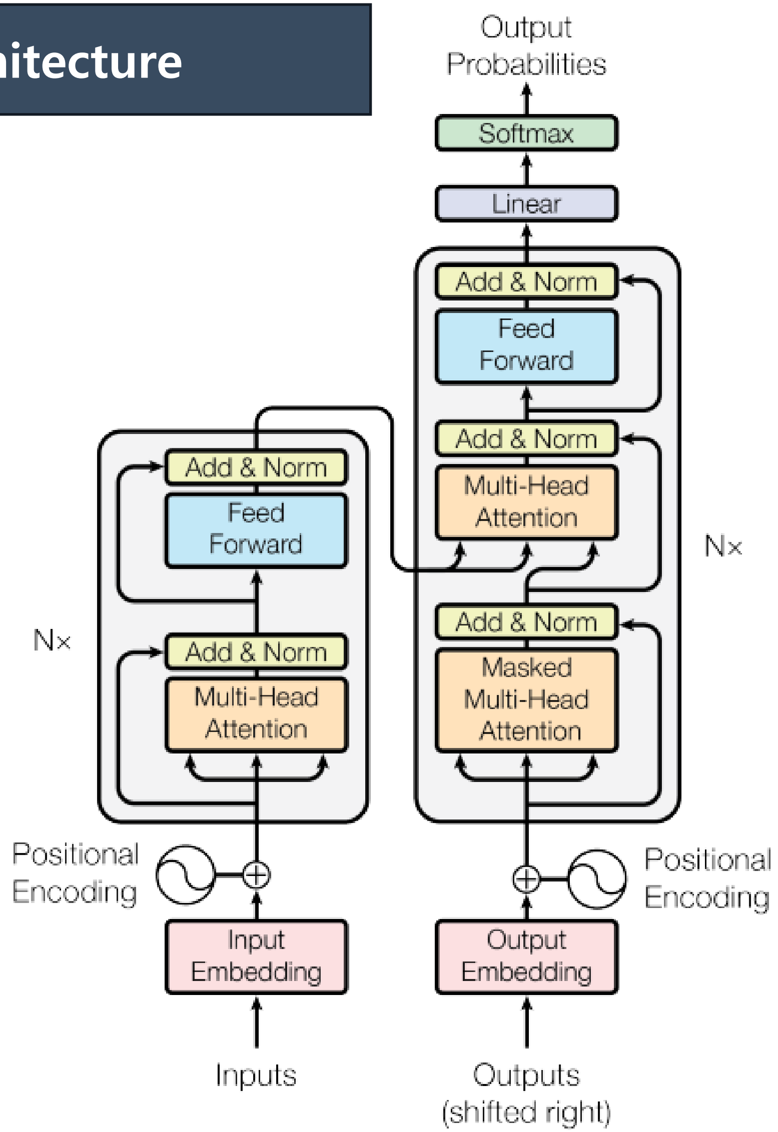


Figure 1: The Transformer - model architecture.

2. Model Architecture

3.2. Attention

- Inputs
  - query vector: $\mathbf{q}$ → dimension $d_q = d_k$
  - keys vector: $\mathbf{k}$ → dimension d_k
  - values vector: $\mathbf{v}$ → dimension d_v
- Output vector = values의 가중합, weight ← (query, key)

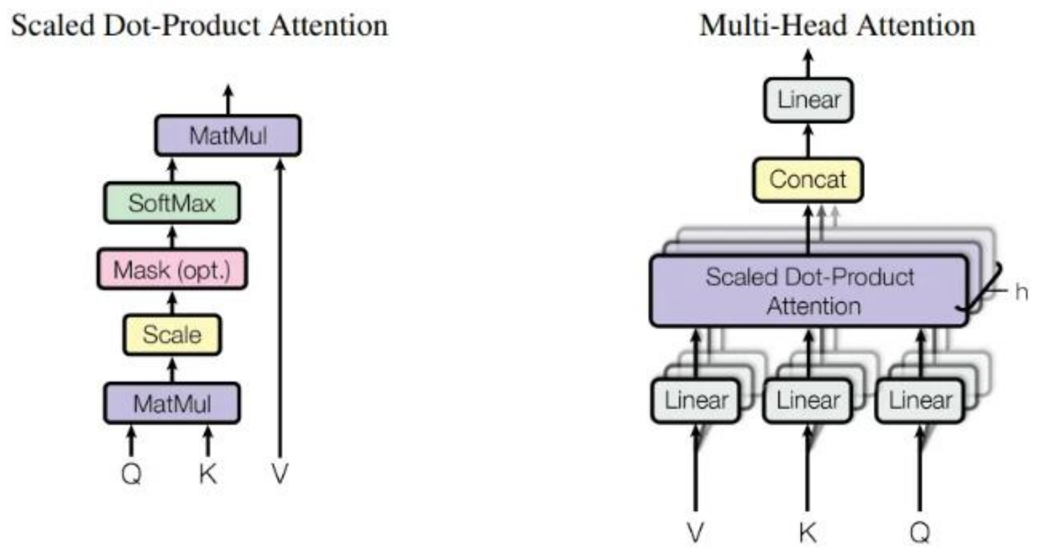
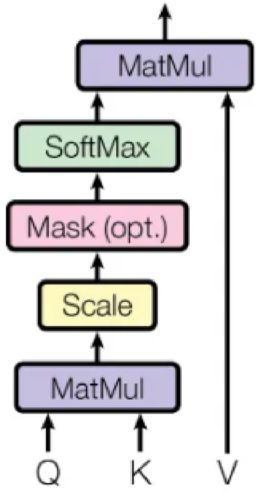


Figure 2: (left) Scaled Dot-Product Attention. (right) Multi-Head Attention consists of several attention layers running in parallel.

2. Model Architecture

- 3.2.1. Scaled Dot-Product Attention [single attention head]

Scaled Dot-Product Attention



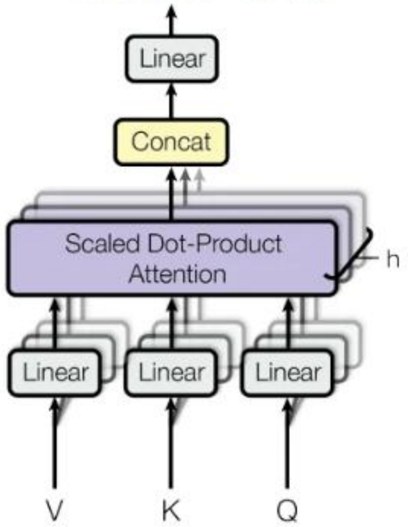
- $\mathbf{q}, \mathbf{k}, \mathbf{v} \rightarrow$ matrix Q, K, V

$$\mathrm{Attention}(Q, K, V) = \mathrm{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (1)$$

  - dot product → scaling → softmax

- 3.2.2. Multi-Head Attention

Multi-Head Attention



- $h = 8$개의 multi-head 사용. with W : parameter matrix

$$\begin{aligned} \mathrm{MultiHead}(Q, K, V) &= \mathrm{Concat}(\mathrm{head}_1, \ldots, \mathrm{head}_h)W^O \\ \text{where} \quad \mathrm{head}_i &= \mathrm{Attention}\left(QW_i^Q, KW_i^K, VW_i^V\right) \end{aligned}$$

$$W_i^Q \in \mathbb{R}^{d_{\text{model}} \times d_k}, \; W_i^K \in \mathbb{R}^{d_{\text{model}} \times d_k}, \; W_i^V \in \mathbb{R}^{d_{\text{model}} \times d_v}$$

$$d_k = d_v = \frac{d_{\text{model}}}{h} = 64 \rightarrow d_{\text{model}} = 512$$

DGIST DATA SCIENCE LAB

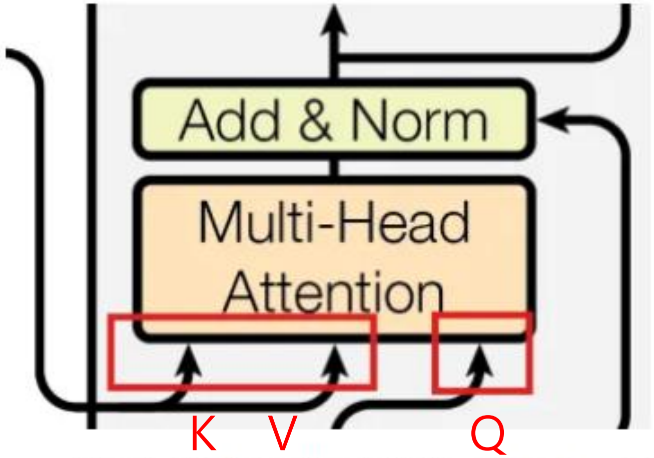
대구경북과학기술원
Daegu Gyeongbuk Institute of Science & Technology

2. Model Architecture

▼ 3.2.3. Applications of Attention in our Model

Transformer uses multi-head attention in three different ways

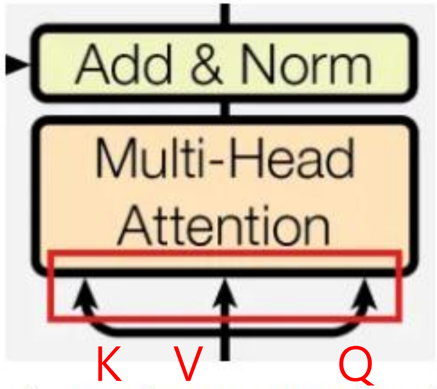
1. Encoder-Decoder Attention layers



   a. previous decoder layer's output → queries

   b. encoder layer's output → memory keys and values

2. Encoder's self-attention layers [Q, K, V가 모두 같은 곳에서 나옴.]



   1. previous encoder layer's output → keys, values and queries

3. Decoder's self-attention layers

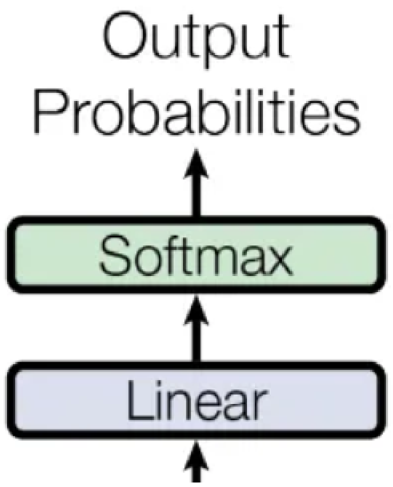
2. Model Architecture

3.3. Position-wise Feed-Forward Networks

- [Feed Forward]: Fully connected feed-forward network

$$\mathrm{FFN}(\mathbf{x}) = \mathrm{RELU}(\mathbf{x}W_1 + \mathbf{b}_1)W_2 + \mathbf{b}_2 \quad (2)$$
$$\text{where } \mathrm{RELU}(x) = \max(0, x)$$

3.4. Embeddings and Softmax



- input/output tokens → embedding vectors
- Learned linear transformation & softmax function: decoder output → predicted next-token probabilities.

2. Model Architecture

Total Architecture

▾ 3.5. Positional Encoding

- RNN, CNN 사용 X → sequence의 ordering 사용 불가.
- → Positional eocodings를 encoder and decoder stacks의 bottoms에 삽입.

> 💡 Sequence의 ordering property 사용 위함.

- 다양한 positional encodings 사용 가능. → 논문에서는 아래와 같이 사용.

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{\frac{2i}{d_{\text{model}}}}}\right)$$

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

  - pos: position
  - i: dimension

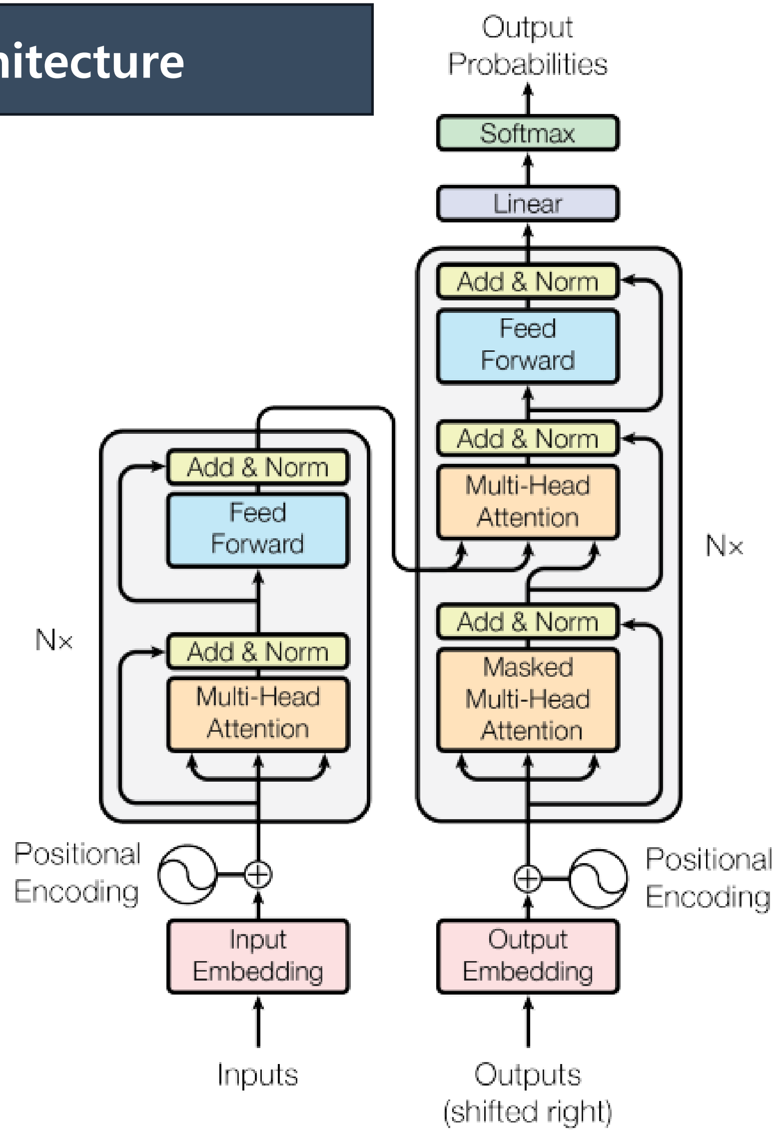


Figure 1: The Transformer - model architecture.

DGIST 대구경북과학기술원 Daegu Gyeongbuk Institute of Science & Technology

DGIST DATA SCIENCE LAB

3. Why Self-Attention

Table 1: Maximum path lengths, per-layer complexity and minimum number of sequential operations for different layer types. n is the sequence length, d is the representation dimension, k is the kernel size of convolutions and r the size of the neighborhood in restricted 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	$O(k \cdot n \cdot d^2)$	$O(1)$	$O(log_k(n))$
Self-Attention (restricted)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$