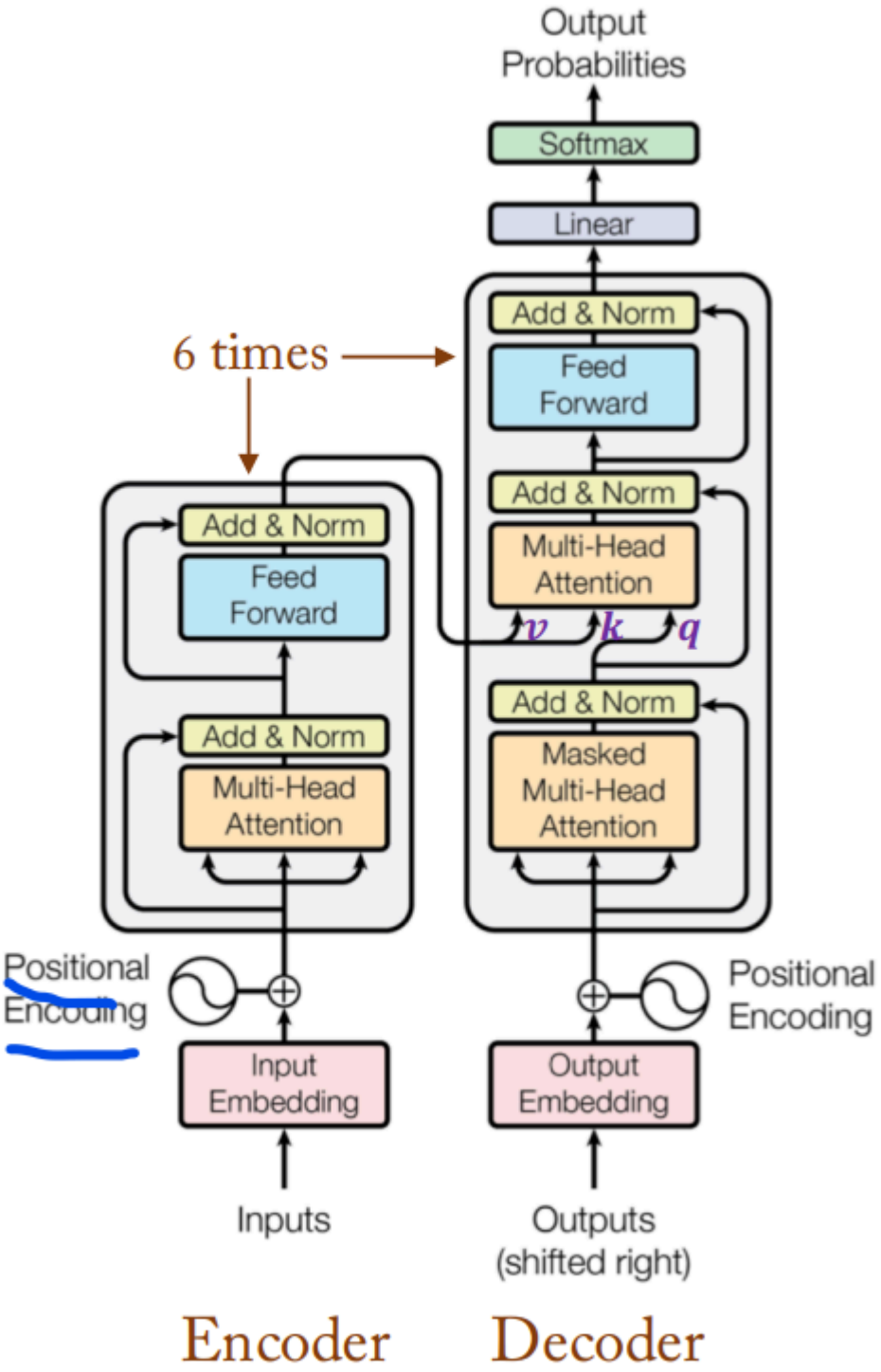
1) Transformer



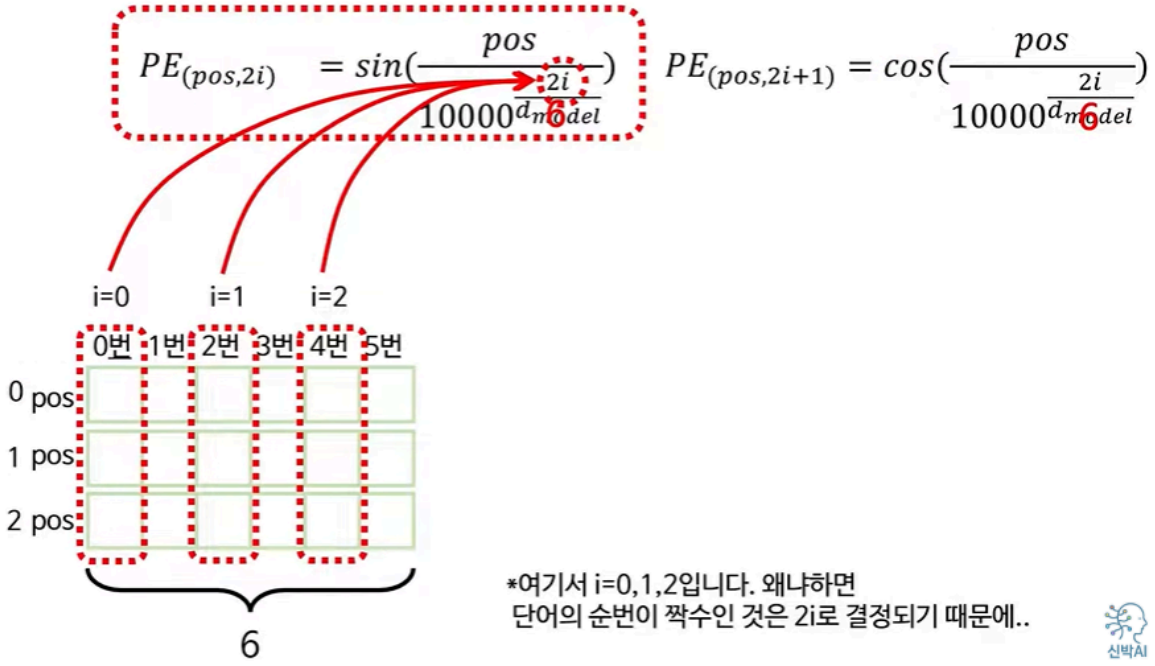
decoder 에서는 value 와 key가 encder의 final output으로 한다
Add & Norm 에서는 일종의 residual connection 넣어서 기존정보 누락을 방지

Input에 시퀀스가 아니라 병렬처리된 문장 자체를 넣어줌. (input 사이즈가 다를 경우엔 어떻게 하지?→padding 사용)

Positional embedding

각 단어 token별 position을 embedding해줌.

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



Self -Attention

Self-Attention : hidden state 들간의 inner product를 계산해 새로운 hidden state를 토큰별로 부과하는 방식

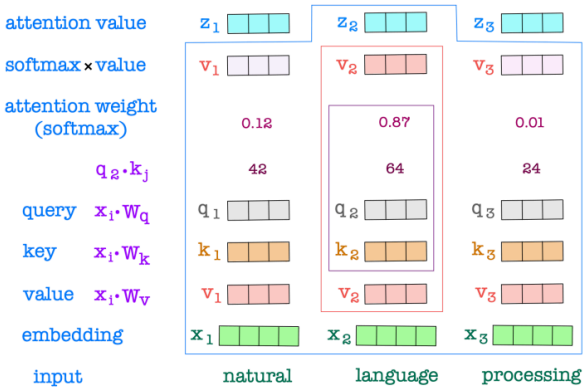
Self-Attention helps the encoder looking at other words in input sentence as it encodes a word.

딥러닝과수학 DLmath Korea University Seungsang Oh

- 3 vectors from the input vector $\boldsymbol{x}_i$ $(d_m = 512)$ which is the output of the previous layer.
  - query vector $\boldsymbol{q}_i = \boldsymbol{x}_i W_q$ $(d_k = 64)$
  - key vector $\boldsymbol{k}_i = \boldsymbol{x}_i W_k$ $(d_k = 64)$
  - value vector $\boldsymbol{v}_i = \boldsymbol{x}_i W_v$ $(d_v = 64)$
    with 3 learnable W_q, W_k, W_v
    without activations (i.e., linear)
- Scaled dot-product attention

  $$z_i = \sum_{j=1}^{N} \text{softmax}\left(\frac{\boldsymbol{q}_i \cdot \boldsymbol{k}_j}{\sqrt{d_k}}\right) \boldsymbol{v}_j \quad (d_v = 64)$$

  - Each position pays attention to all positions in the previous layer.
  - $\boldsymbol{q}_i \cdot \boldsymbol{k}_j$ is the relativity of $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$.
  - scaled by $\frac{1}{\sqrt{d_k}}$ to lead stable gradients (softmax)



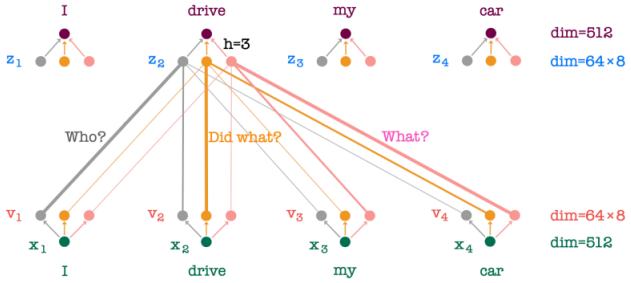
- In encoder-decoder LSTM, query vector is usually the hidden state of the decoder, whereas key and value vectors are the hidden state of the encoder.

228 / 279

Multi-Head Attention

딥러닝과수학 DLmath Korea University Seungsang Oh

- Instead of a single attention with queries, keys and values $(d_m = 512)$, we use $h = 8$ attentions (heads) with smaller queries, keys and values $(d_v = \frac{d_m}{h} = 64)$ using weights W_q, W_k, W_v (different for each head, different from layer to layer, but the same across all word positions).
- These attentions independently learn in parallel, yielding h output values $z_i^1, \ldots, z_i^h$ with $d_v = 64$, which are concatenated and once again a FC layer without activations, resulting in final values with $d_m = 512$ for each word position i (position-wise from the input values to the final values).



$$[z_i^1; \ldots; z_i^h] W_o$$

* It pays attention jointly to information in different representations (heads) that average differently attention-weighted positions.

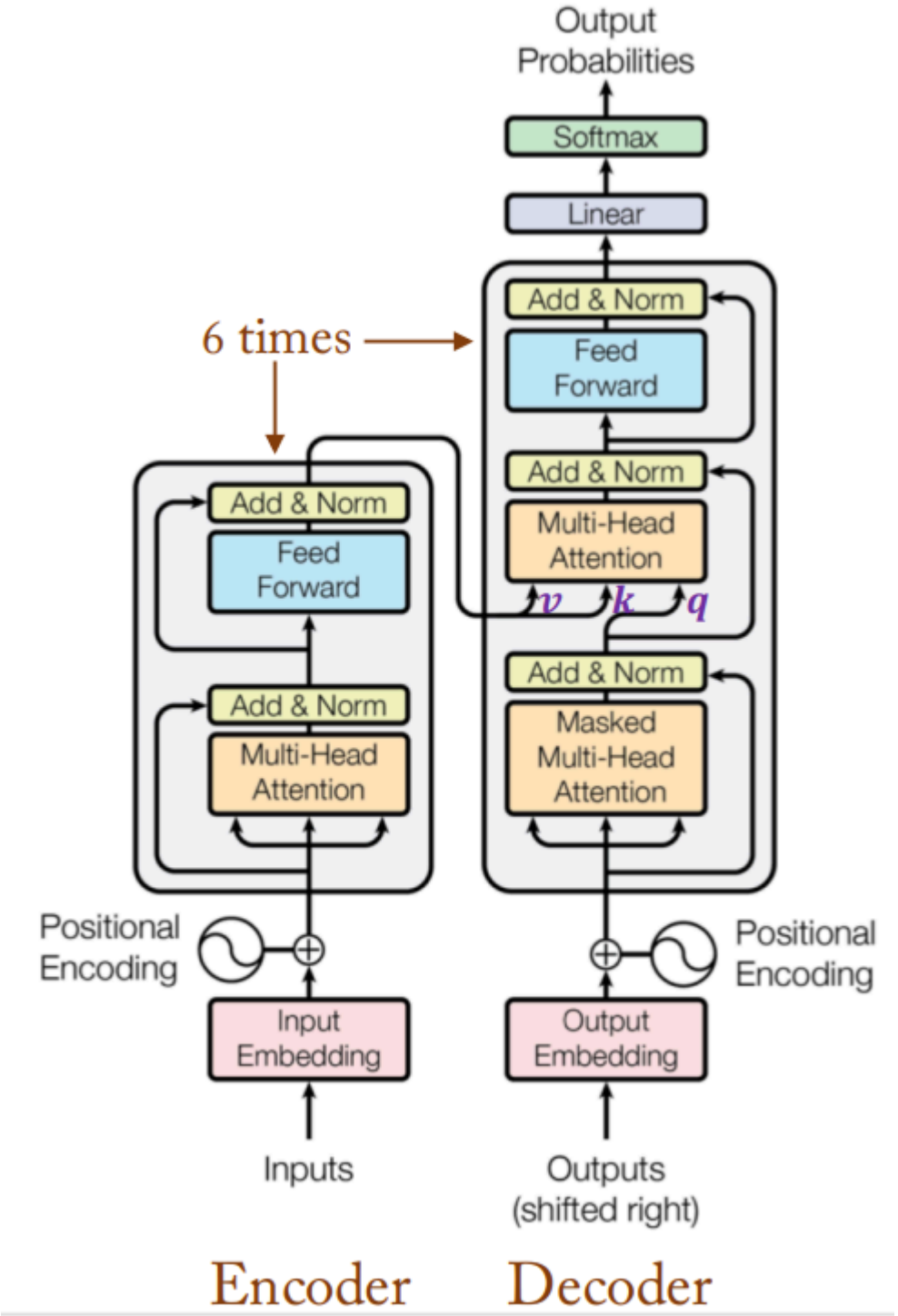
각 attention 별로 다른 관계를 포착한다 (문법, 의미 등) 이후 모은 정보들 concat(64 dimension 씩 8개 head)

Masked Multi-head Attention

For decoder, input is the whole sequence of sentence. And at first layer of multi-head attention, we are generating word that fits sentences based on past sequence. Thus whole inner product of sequence is not appropriate (since it

has future data.) because of this, we are masking past values. Then the attention matrix becomes lower triangular.

At second part of multi-head attention, we are testing if the generated sentences fit meaning of the past sentences encoded in the encoder. Thus key and value vectors are extracted from the encoder. We extracted data which represents meaning. Then residual connection of first and second layers is text that is appropriate in context(meaning) and order.



Position-wise Feedforward neural network

At this part, we are removing words that is too simple and into more complicated form. Since attention is almost linear model. And such FFN is

activated independently per postion.

This sub-layer consists of 2 FCs ($512 \rightarrow 2048 \rightarrow 512$) with ReLU in between:

$$\mathrm{FFNN}(z) = \max(0, zW_1 + b_1)W_2 + b_2$$

https://www.youtube.com/watch?v=8E6-emm_QVg&t=287s

https://www.youtube.com/watch?v=p216tTVxu

오승상. *Deep Learning*. 고려대학교 수리과학과 및 데이터과학과 강의자료, 2024.