How Transformer Model Works

주재걸 교수
KAIST 김재철AI대학원

DAVIAN
Data and Visual Analytics Lab

Review: Seq2Seq with Attention ← RNN 기반

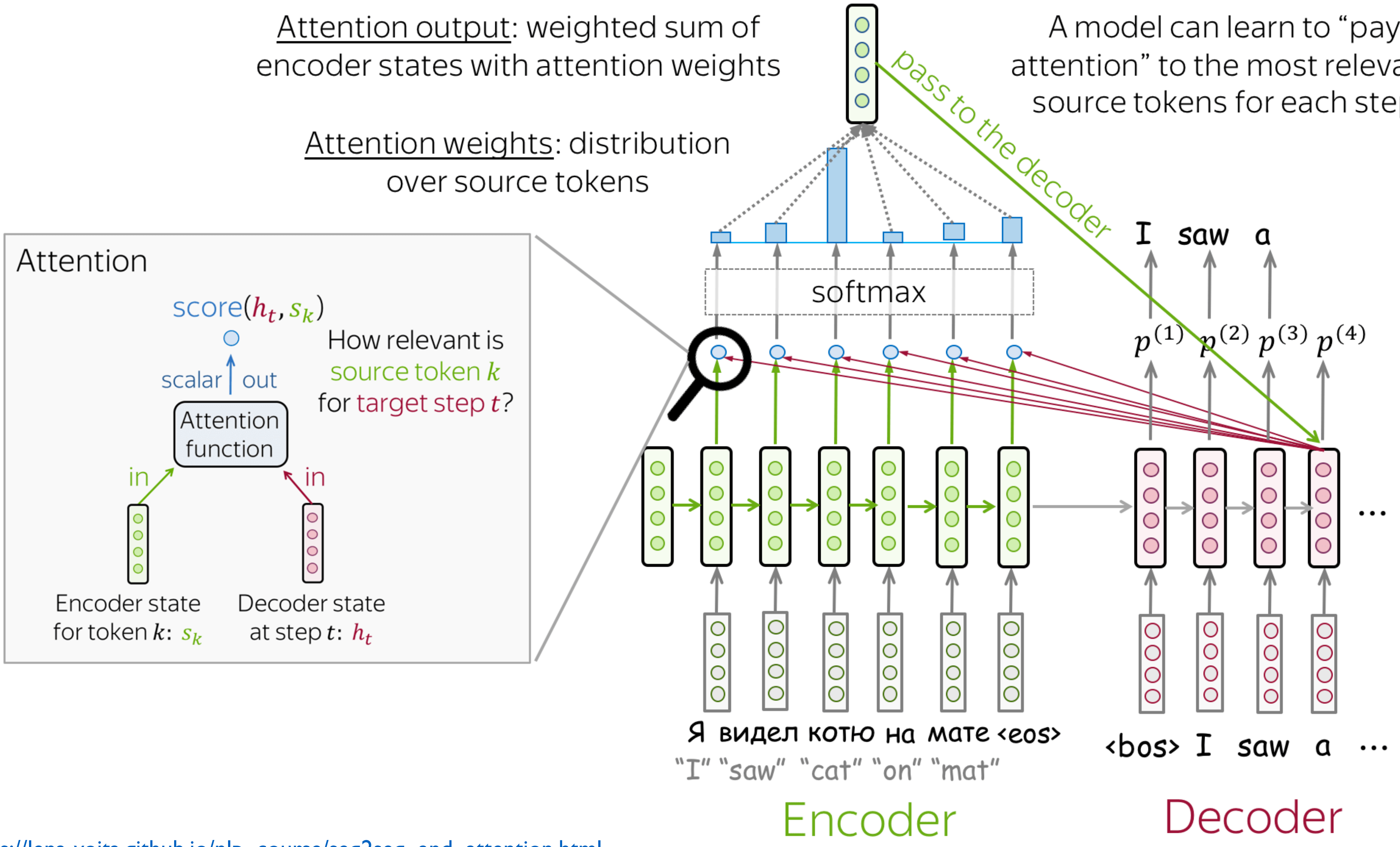
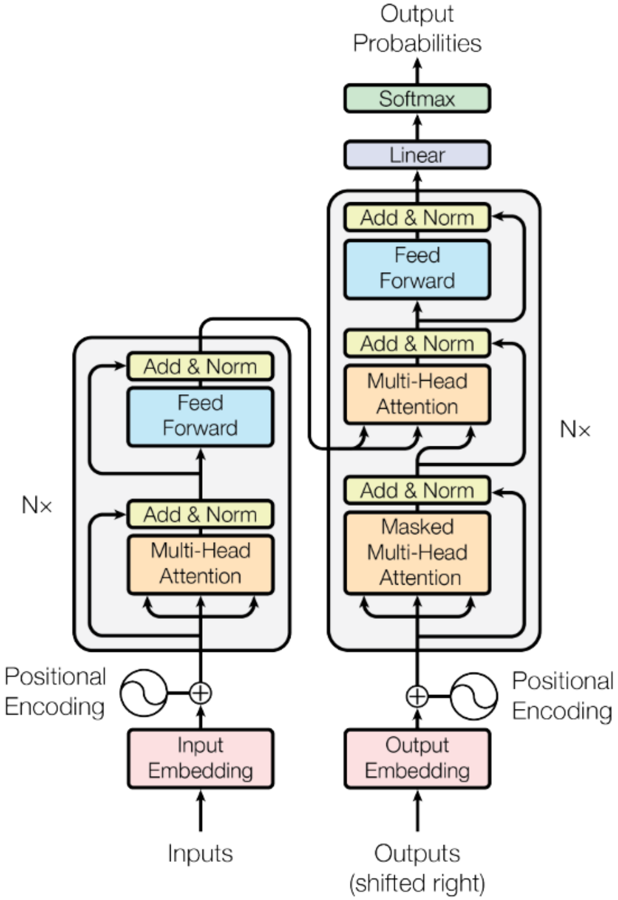


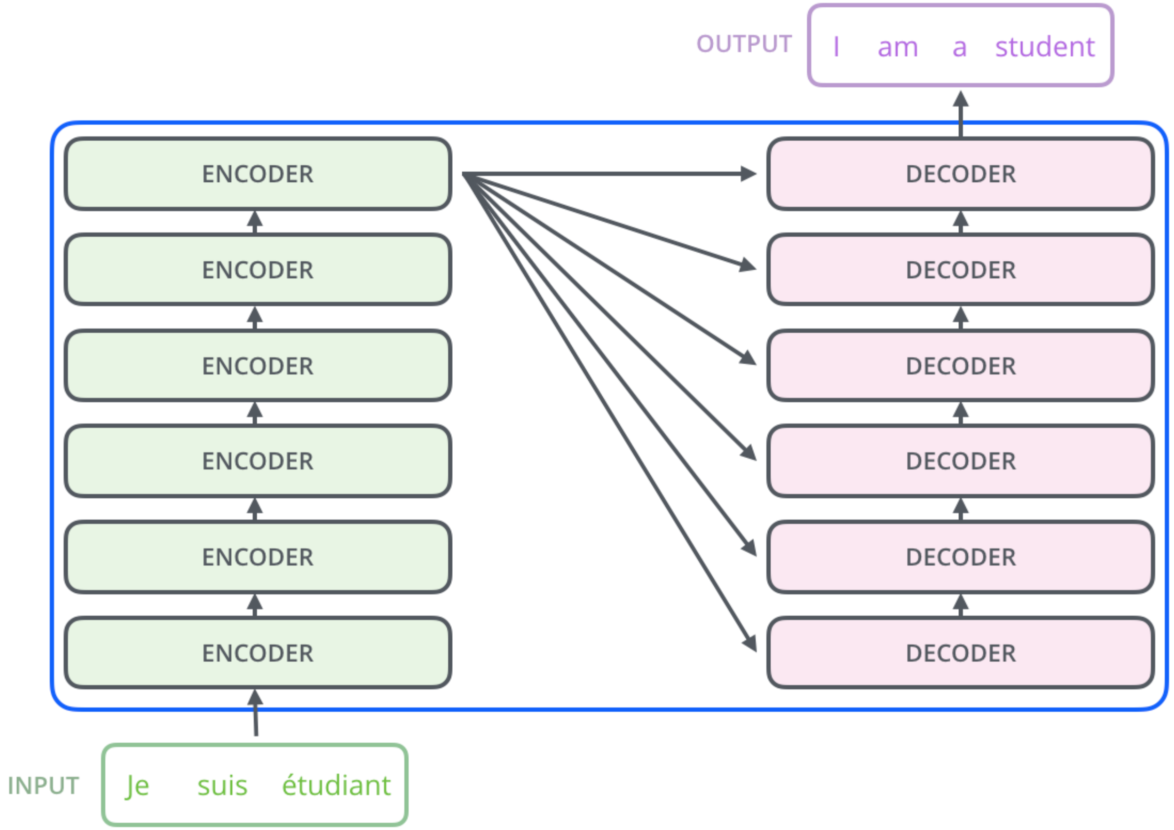
Image courtesy: https://lena-voita.github.io/nlp_course/seq2seq_and_attention.html

Seq2seq attention 개선

Transformer: High-level View

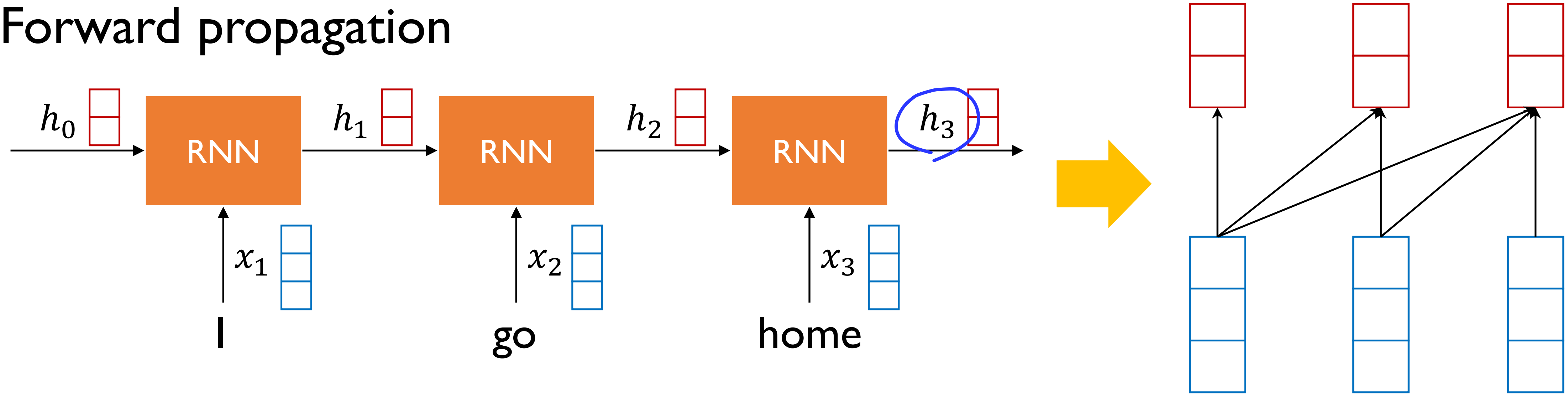
- Attention module can work as both a sequence encoder and a decoder in seq2seq with attention.
- In other words, RNNs or CNNs are no longer necessary, but all we need is attention modules.



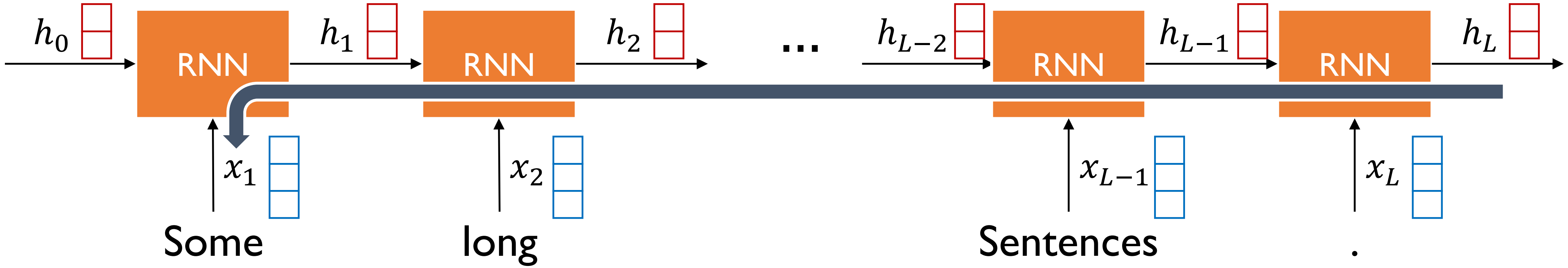


Long-term Dependency Issue of RNN Models

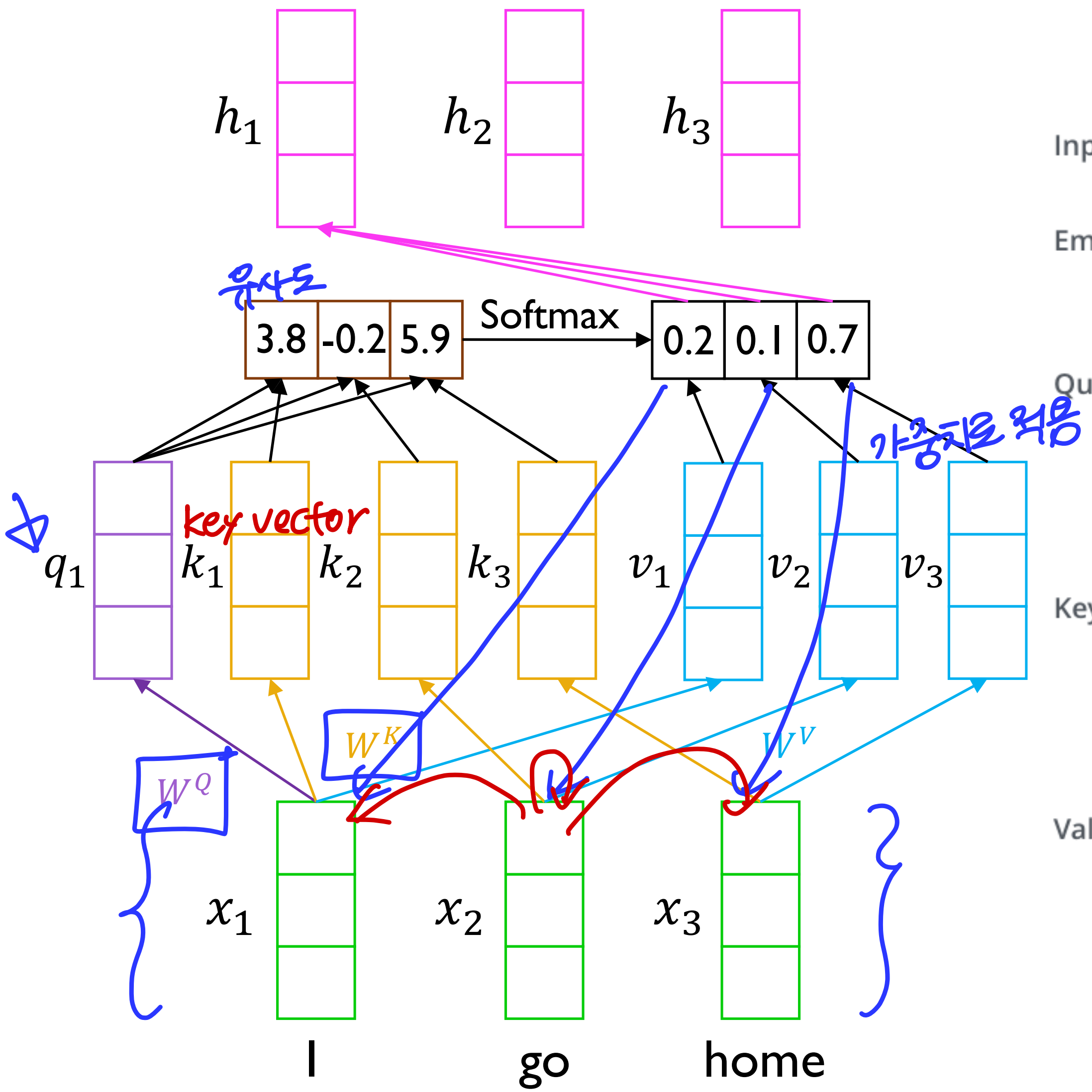
Forward propagation



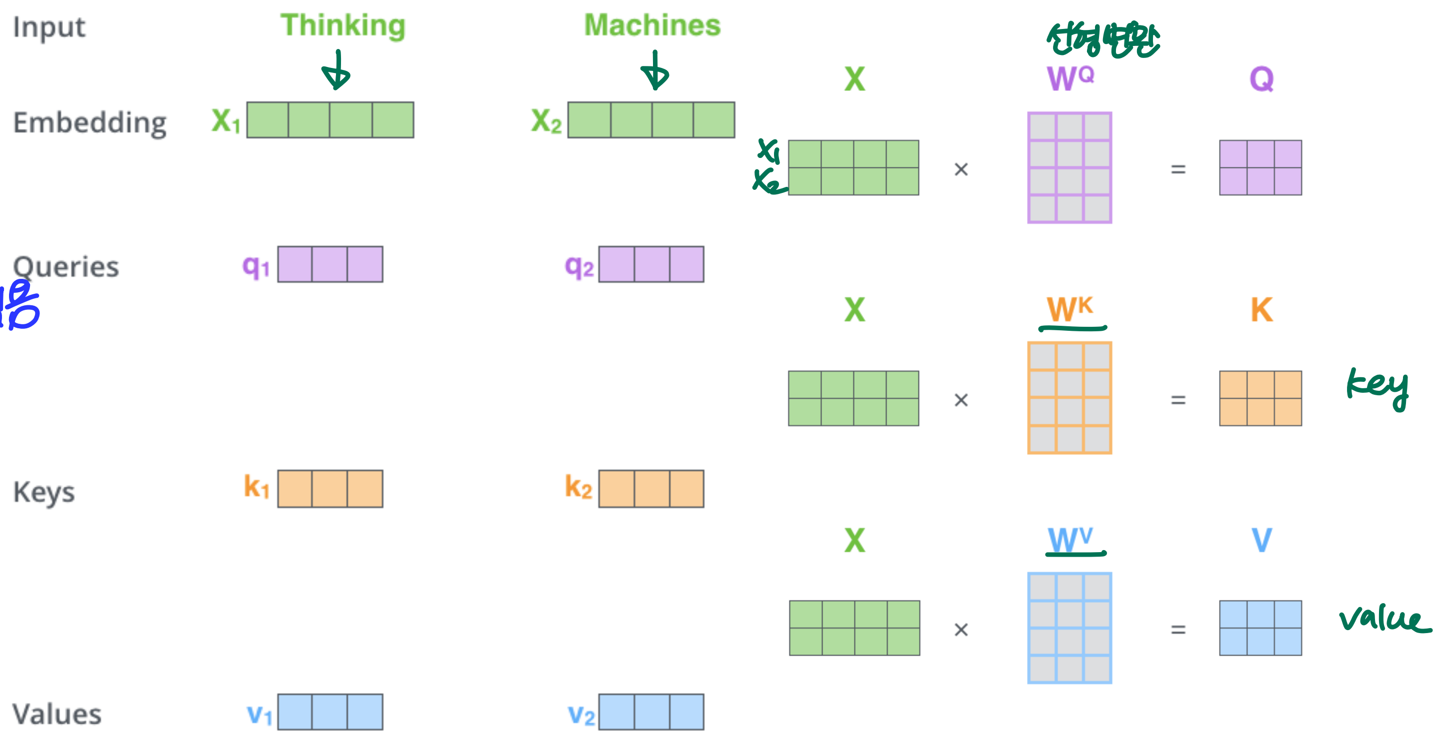
Backpropagation



Transformer: Solving Long-term Dependency Problem

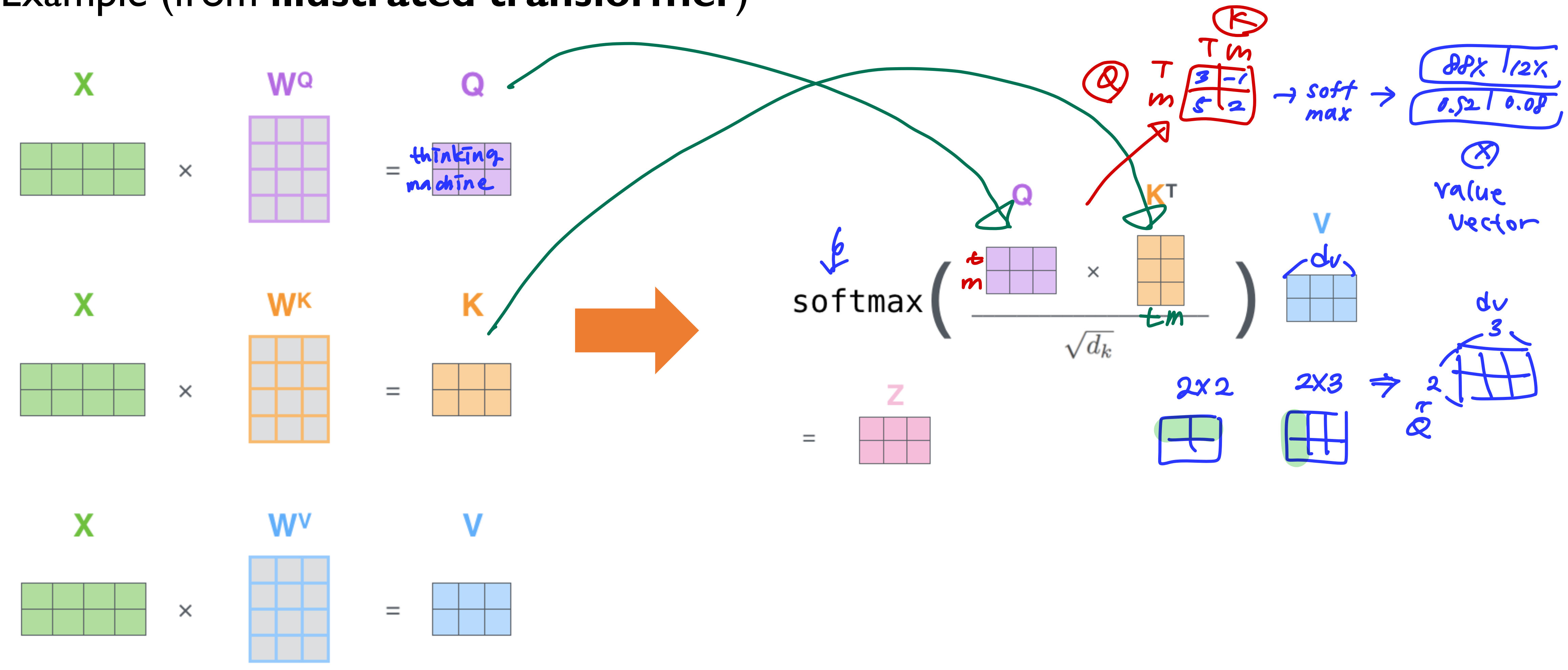


★ Self-attention



Transformer: Scaled Dot-product Attention

- Example (from **illustrated transformer**)



Transformer: Scaled Dot-product Attention

평균 분산
10 → (0, 5)
1000 → (0, 200)

- Problem

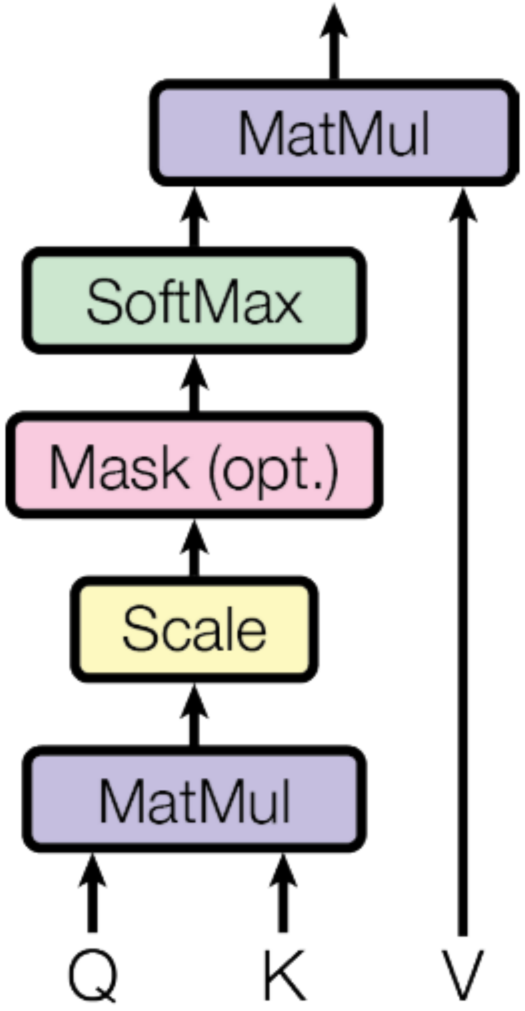
dimension이 클수록 더 많은 숫자 합 → 분산 큼 → softmax 특성상 한 값에 100% 몰아줌.
↓
gradient 학습 X

  - As d_k gets large, the variance of $q^T k$ increases.
  - Some values inside the softmax get large.
  - The softmax gets very peaked.
  - Hence its gradient gets smaller.

- Solution
  - Scaled by the length of query / key vectors:

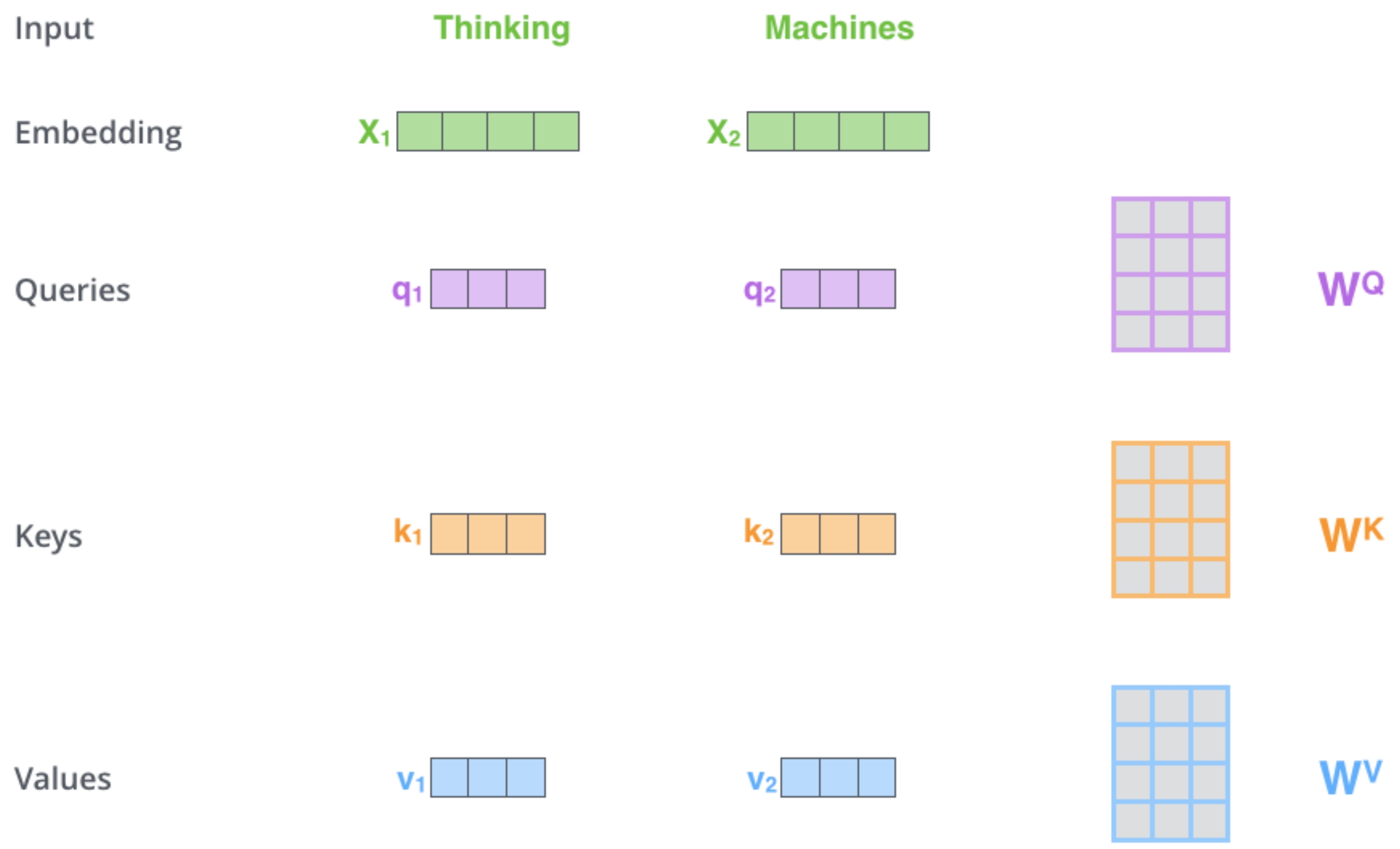
일정한 분산값 갖도록.

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



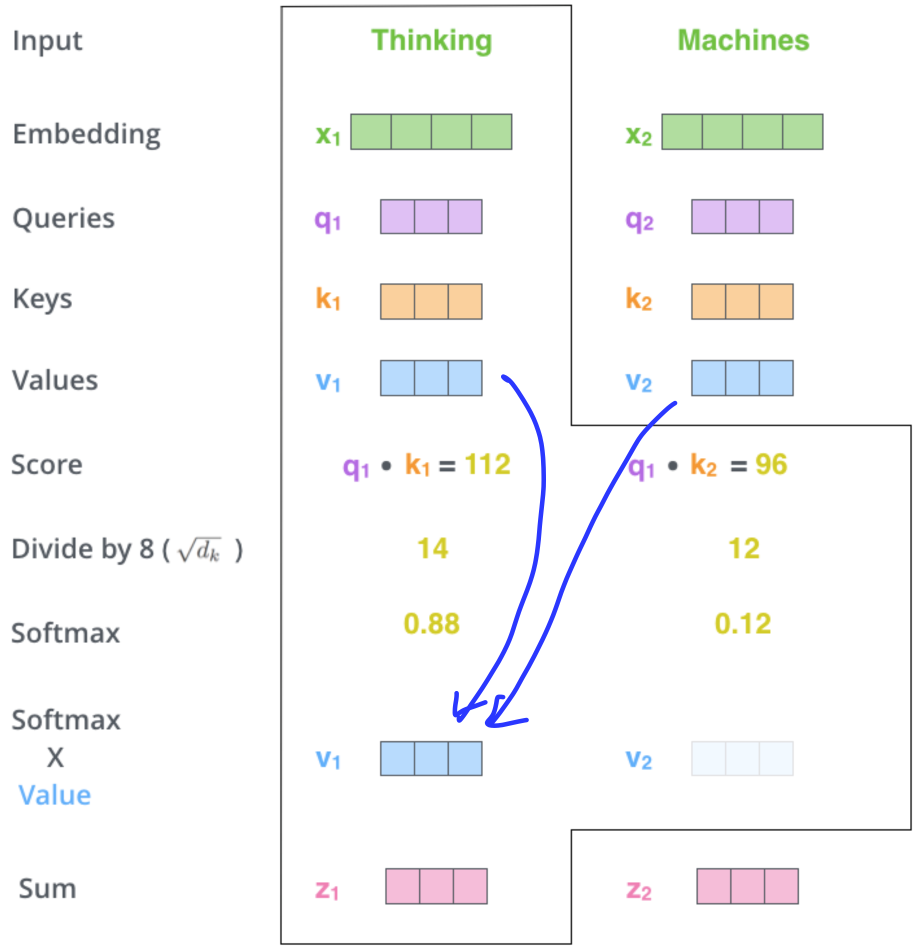
Transformer: Scaled Dot-product Attention

- Example (from **illustrated transformer**)



Transformer: Scaled Dot-product Attention

- Example (from **illustrated transformer**)

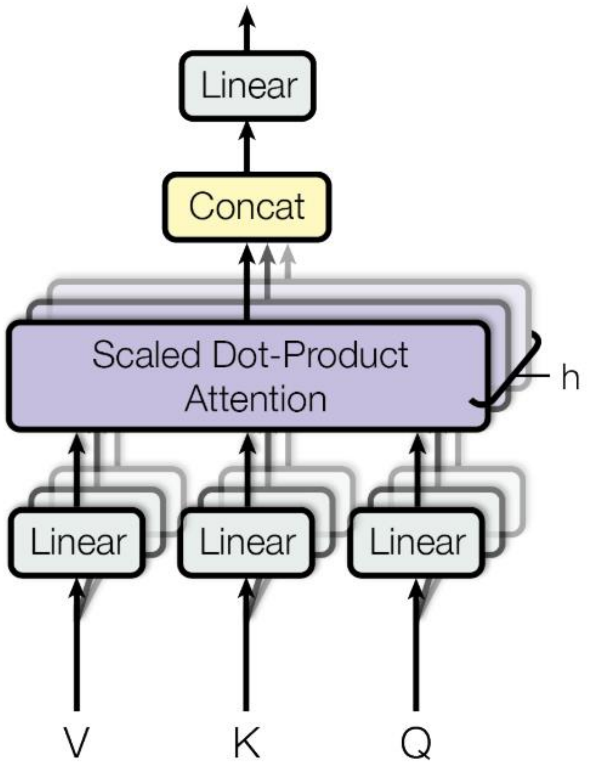


http://jalammar.github.io/illustrated-transformer/

Transformer: Multi-head Attention

- The input word vectors can be the queries, keys and values.
- In other words, the word vectors themselves select one another.
- **Problem:** only one way for words to interact with one another.

- **Solution:** multi-head attention maps Q, K, and V into the h number of lower-dimensional spaces via W matrices.
- Afterwards, apply attention, then concatenate outputs and pipe through linear layer.



$W^Q \; W^K \; W^V$

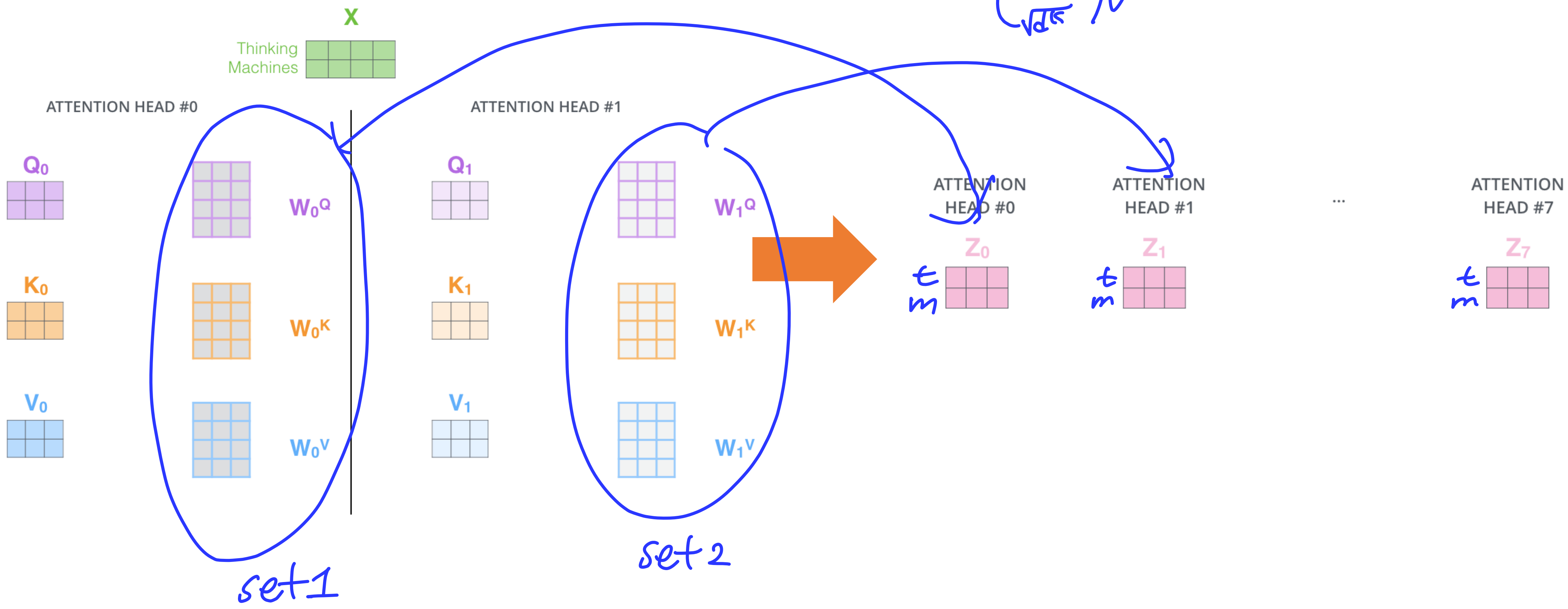
↳ 3개의 선형변환 × 여러개

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

Transformer: Multi-head Attention

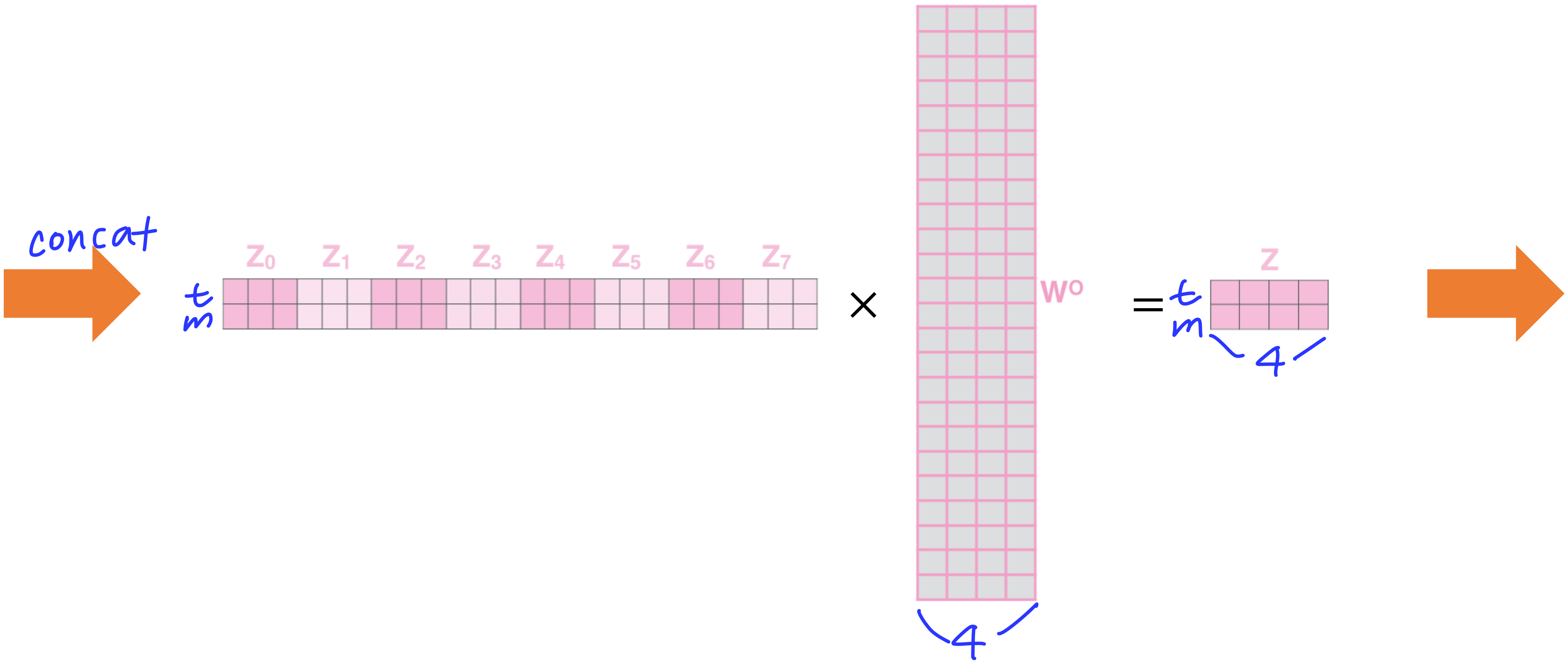
- Example (from **illustrated transformer**)

$$\text{Softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$



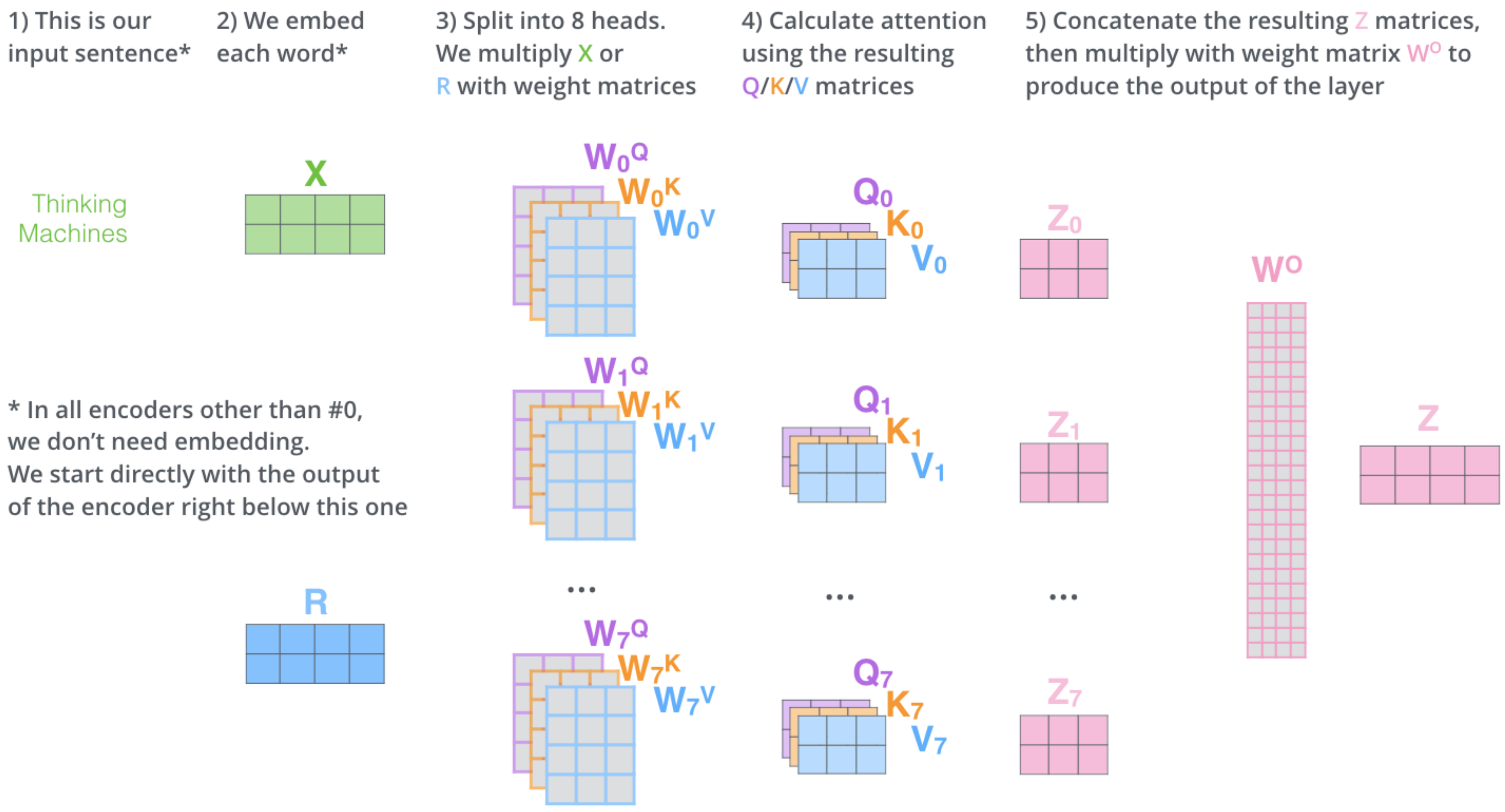
Transformer: Multi-head Attention

- Example (from **illustrated transformer**)



Transformer: Multi-head Attention

- Example (from **illustrated transformer**)



Transformer: Quadratic Memory Complexity

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	$\boxed{O(n^2 \cdot d)}$	$O(1)$	$O(1)$
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)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

메모리 사이즈 넘 커짐.

Transformer: Block-Based Model

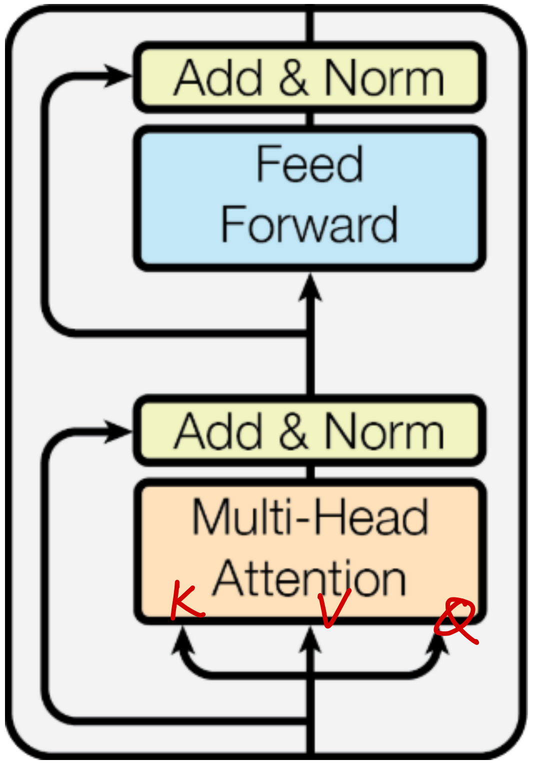
Each block has two **sub-layers**

- Multi-head attention
- Two-layer feed-forward NN (with ReLU)

Each of these two steps also has

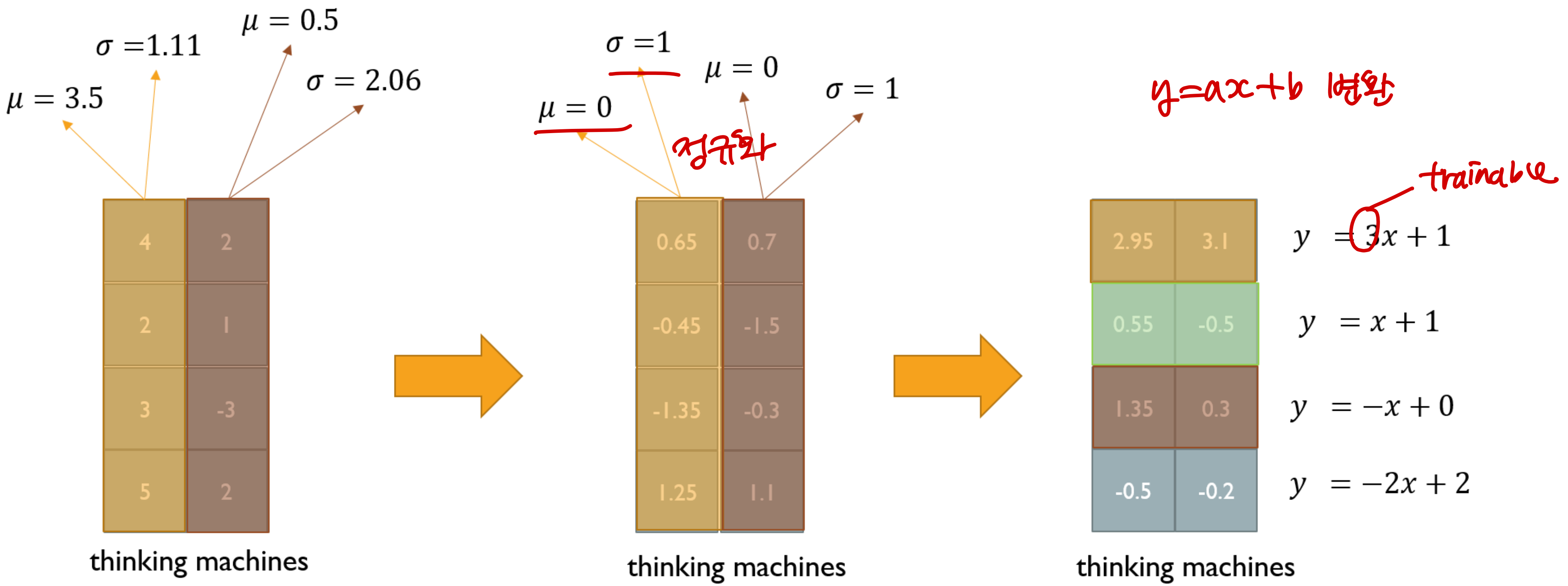
- Residual connection and layer normalization:

  $\mathrm{LayerNorm}(x + \mathrm{sublayer}(x))$



Layer Normalization

- Layer normalization consists of two steps:
  - Normalization of each word vectors to have zero mean of zero and variance of one.
  - Affine transformation of each sequence vector with learnable parameters.



Transformer: Positional Encoding

- Use sinusoidal functions of different frequencies:

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

자체적으로 학습해서 사용하는 이거

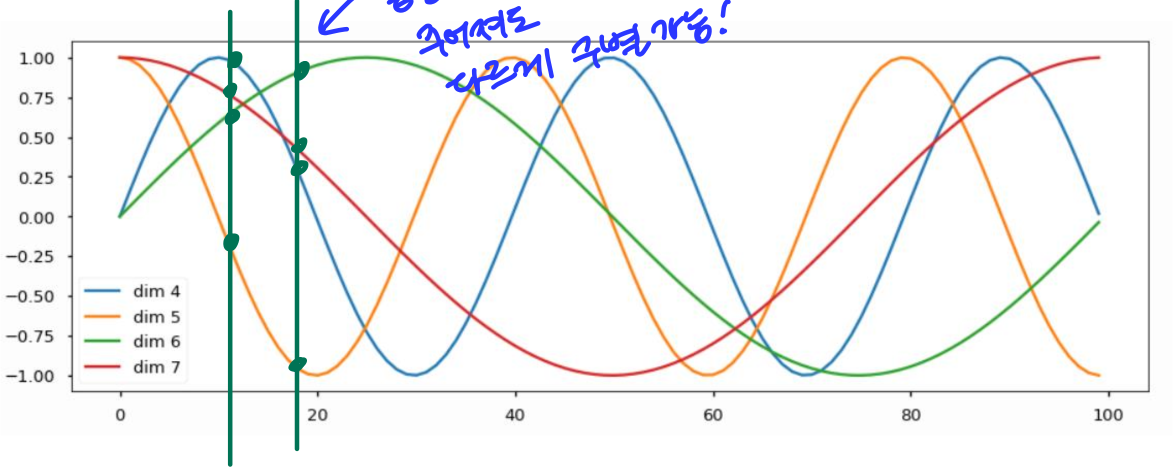
$$PE_{(\text{pos},2i+1)} = \cos(\text{pos}/10000^{2i/d_{\text{model}}})$$

home

- Easily learn to attend by relative position, since for any fixed offset k, $PE_{(\text{pos}+k)}$ can be represented as linear function of $PE_{(\text{pos})}$

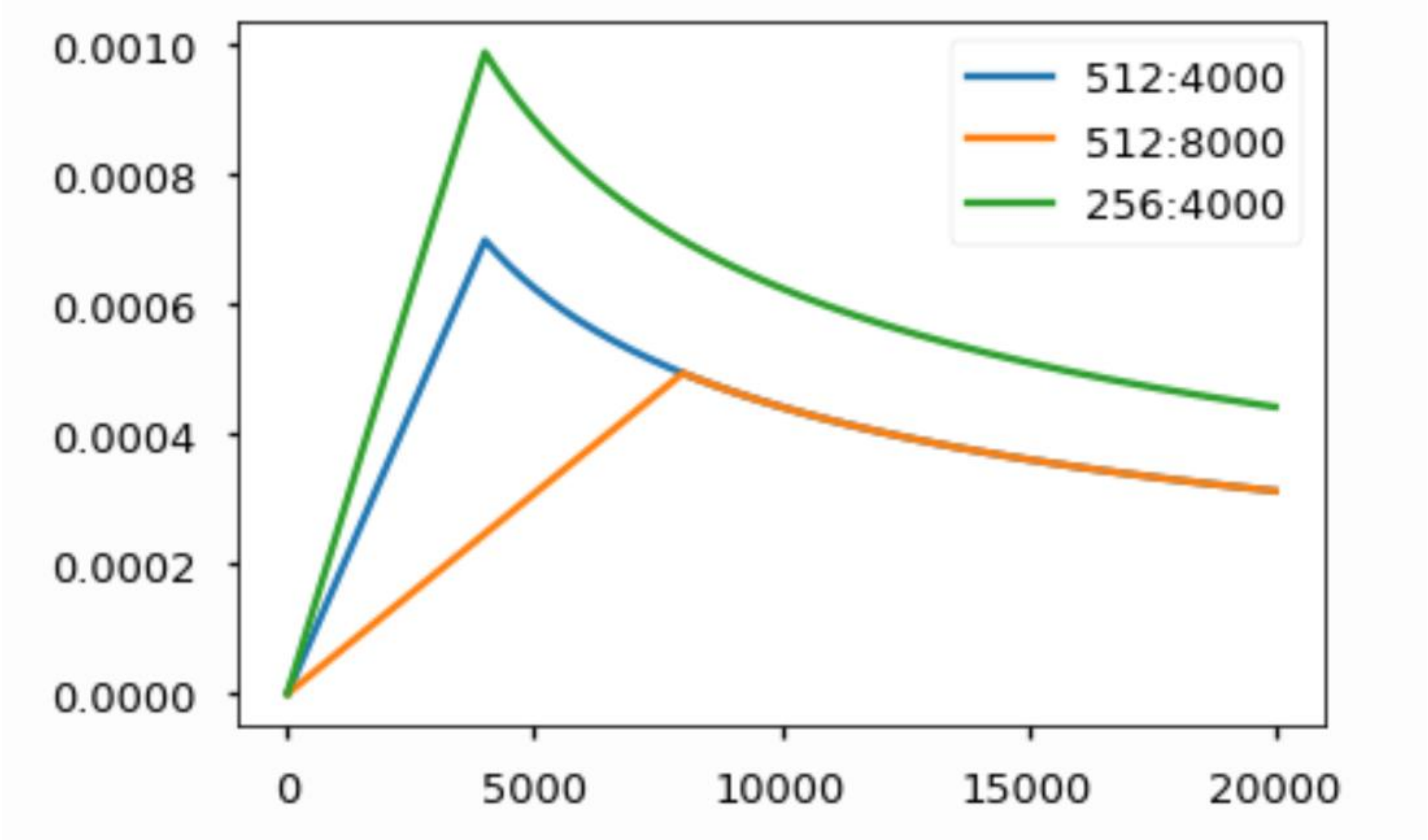
- Another positional encoding can also be used (e.g., positional encoding in ConvS2S).

같은 단어 주어져도 다르게 구별 가능!



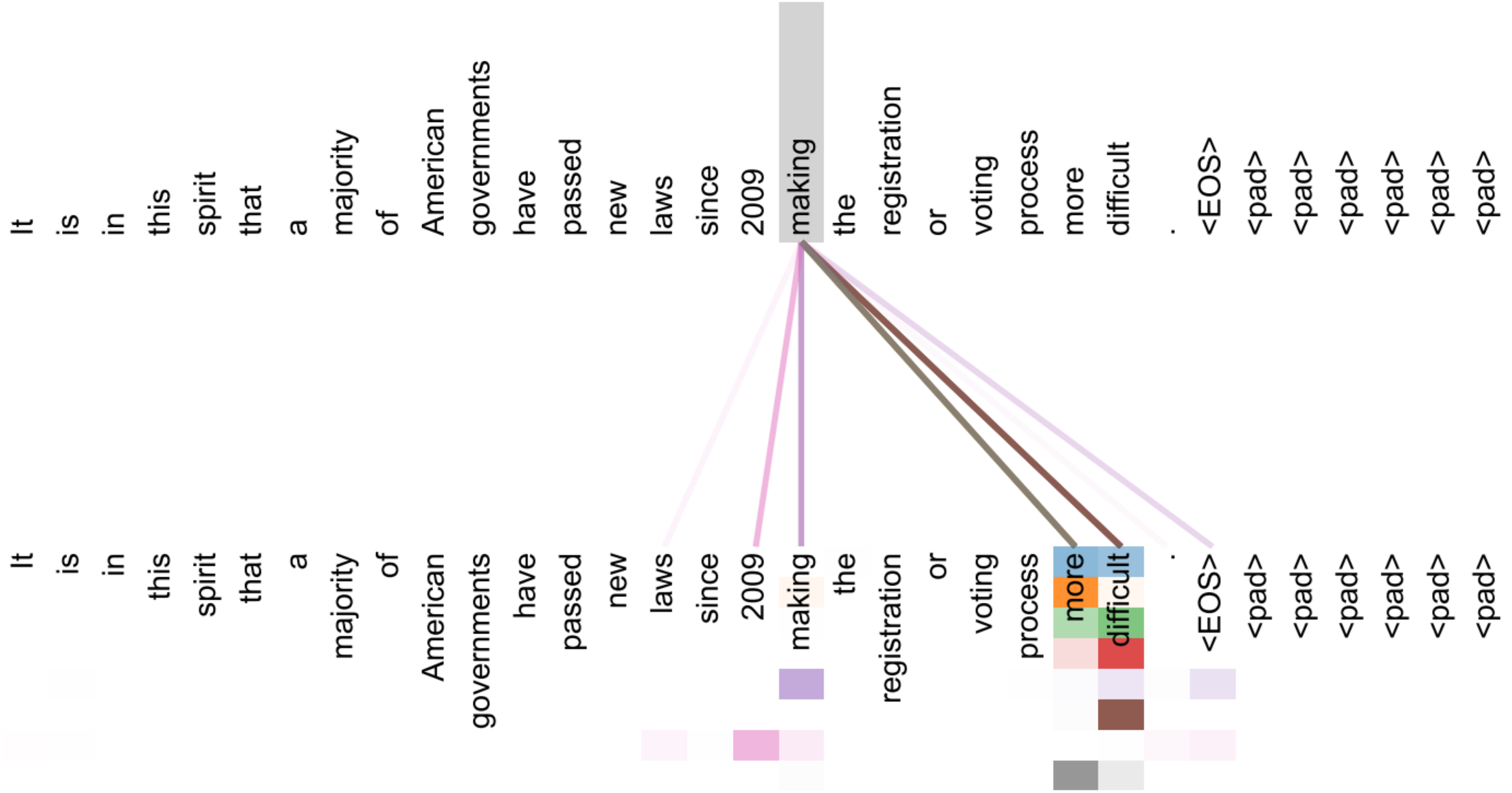
Transformer: Warm-up Learning Rate Scheduler

- learning rate $= d_{\text{model}}^{-0.5} \cdot \min(\#\text{step}^{-0.5}, \#\text{step} \cdot \text{warmup\_step}^{-1.5})$



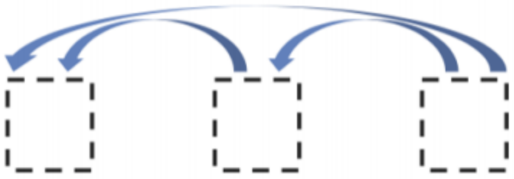
Transformer: Encoder Self-attention Visualization

- Words start to pay attention to other words in sensible ways.

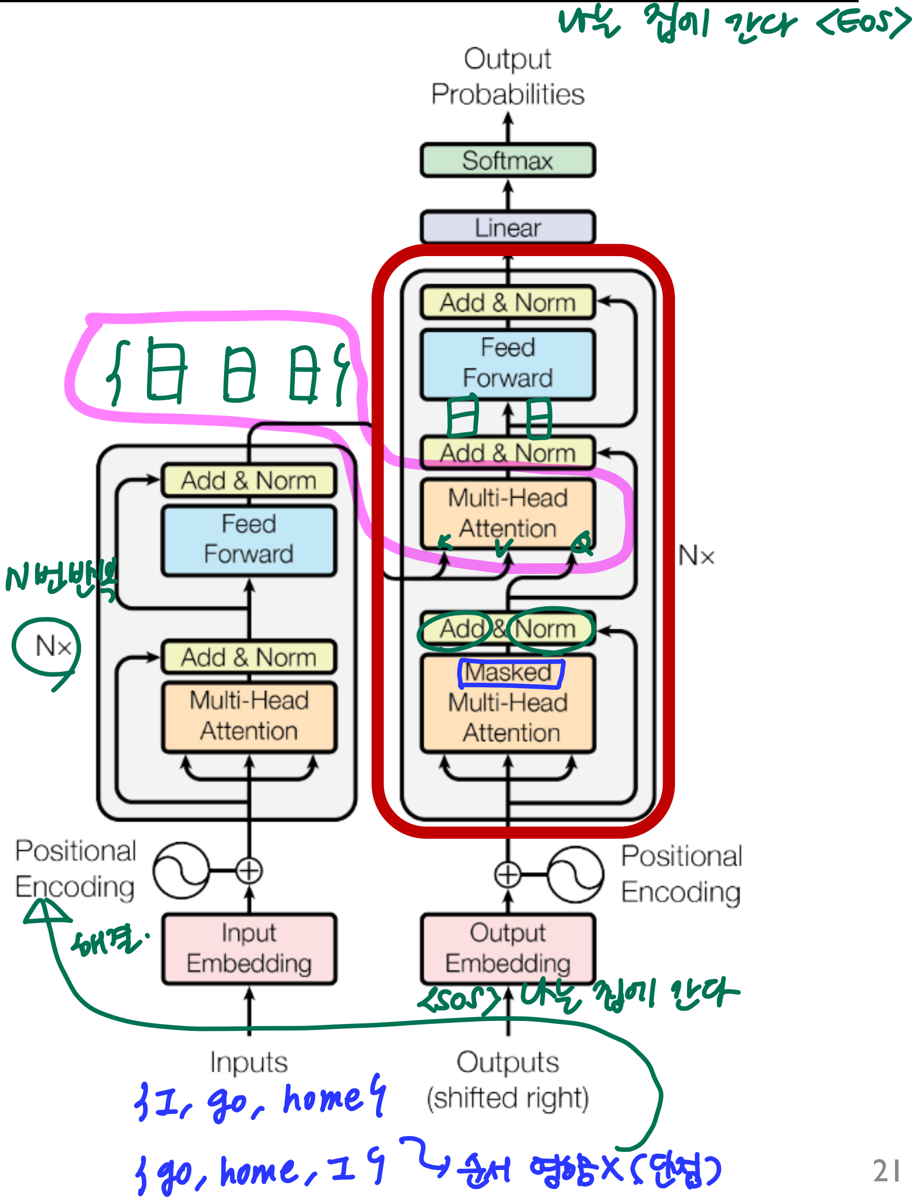


Transformer: Decoder

- Two sub-layer changes in decoder
- Masked decoder self-attention on previously generated outputs:

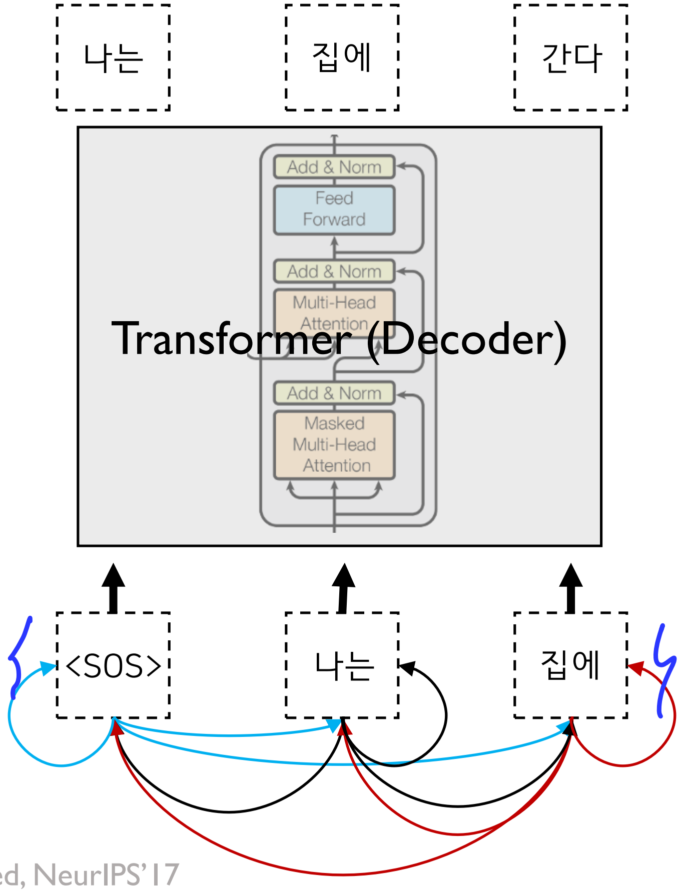


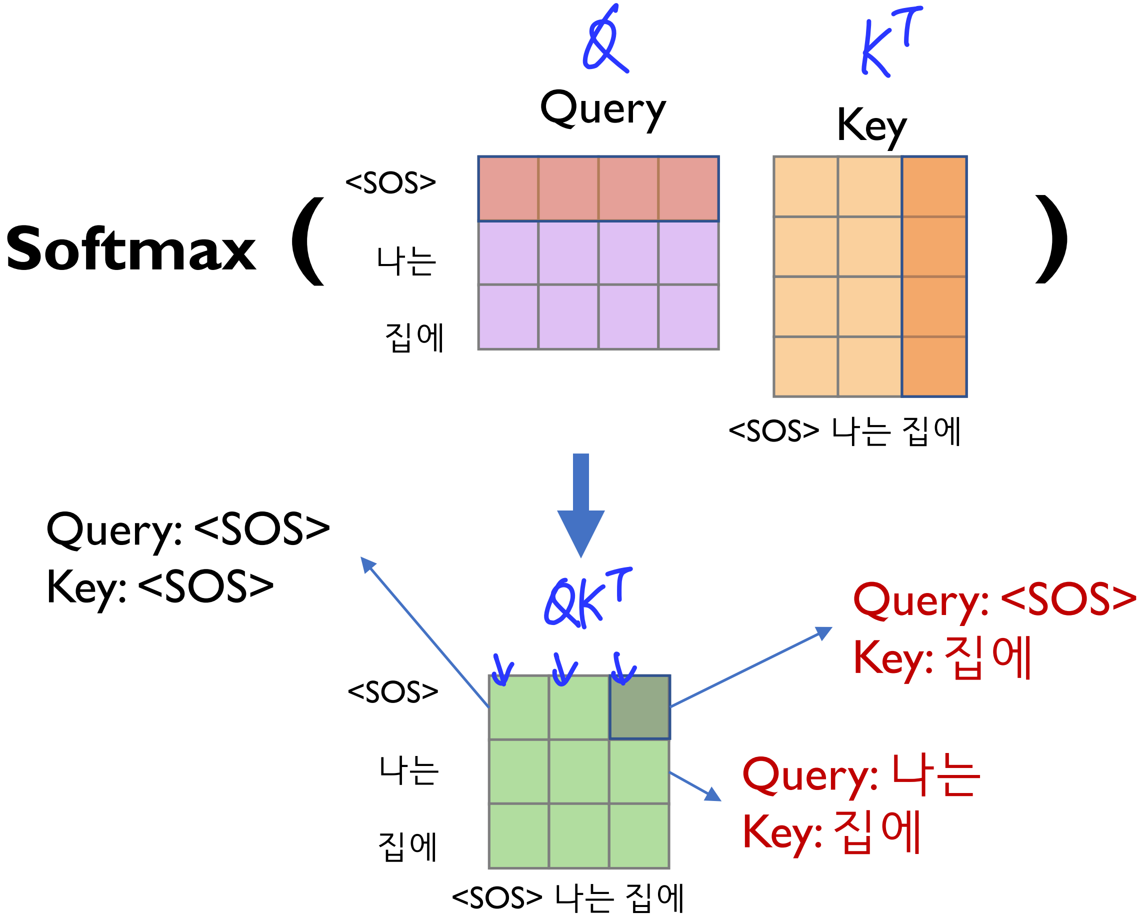
- Encoder-Decoder attention,
  where queries come from previous decoder layer and keys and values come from output of encoder



Transformer: Masked Self-attention

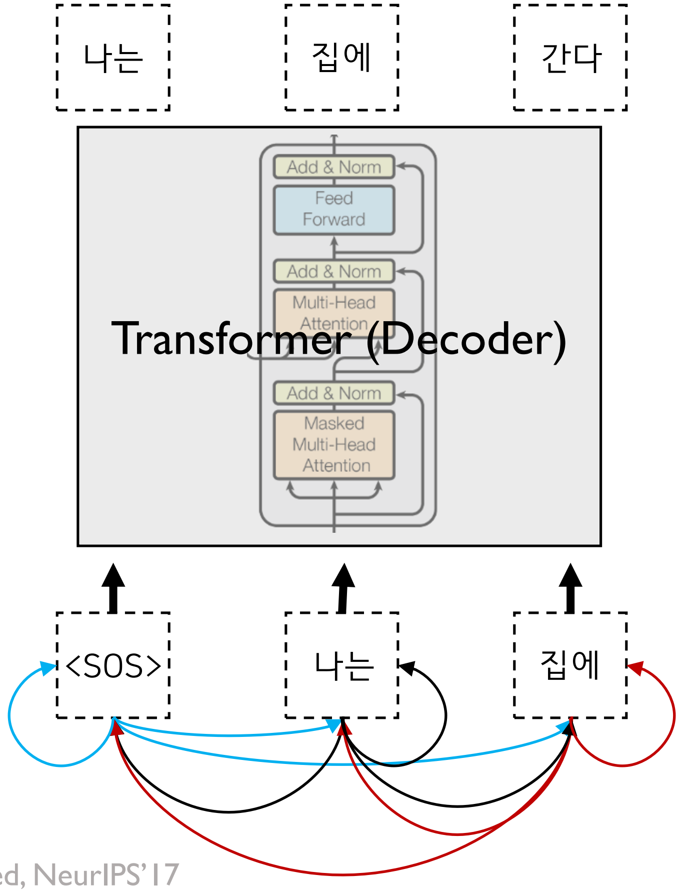
- Those words not yet generated cannot be accessed during the inference time.
- Renormalization of softmax output prevents the model from accessing not yet generated words.





Transformer: Masked Self-attention

- Those words not yet generated cannot be accessed during the inference time.
- Renormalization of softmax output prevents the model from accessing not yet generated words.



	<SOS>	나는	집에
<SOS>	2.01	-0.94	-1.18
나는	0.84	0.96	-0.49
집에	0.29	0.48	0.84

$-\infty$ $-\infty$

SoftMax

	<SOS>	나는	집에
<SOS>	0.91	0.05	0.04
나는	0.42	0.47	0.11
집에	0.25	0.31	0.44

Query: <SOS>
Key: <SOS>

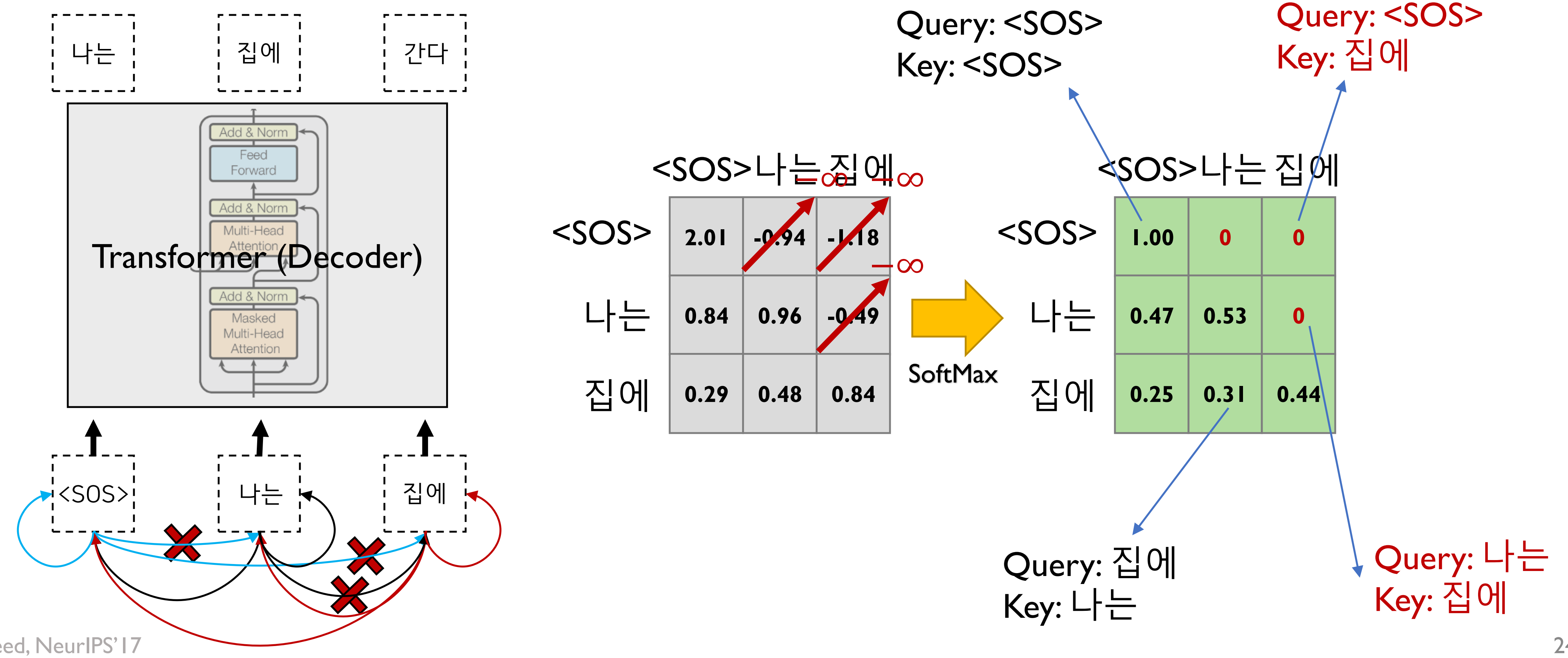
Query: <SOS>
Key: 집에

Query: 집에
Key: 나는

Query: 나는
Key: 집에

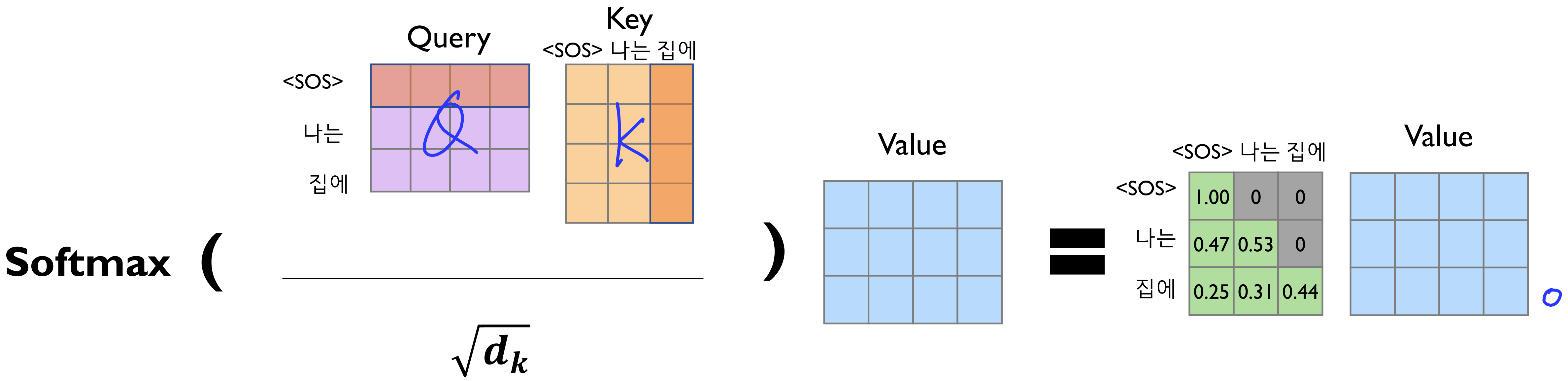
Transformer: Masked Self-attention

- Those words not yet generated cannot be accessed during the inference time.
- Renormalization of softmax output prevents the model from accessing not yet generated words.



Transformer: Masked Self-attention

- Those words not yet generated cannot be accessed during the inference time.
- Renormalization of softmax output prevents the model from accessing not yet generated words.



Transformer: Experimental Results

- Results on English-German/French translation (newstest2014)

Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [18]	23.75			
Deep-Att + PosUnk [39]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [38]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [9]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [32]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [39]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [38]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [9]	26.36	**41.29**	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	$\mathbf{3.3 \cdot 10^{18}}$	
Transformer (big)	**28.4**	**41.8**	$2.3 \cdot 10^{19}$	

Recent Trends

- Transformer model and its self-attention block has become a general-purpose sequence (or set) encoder in recent NLP applications as well as in other areas.
- Training deeply stacked Transformer models via a self-supervised learning framework has significantly advanced various NLP tasks via transfer learning, e.g., BERT, GPT-2, GPT-3, XLNet, ALBERT, RoBERTa, Reformer, T5, …
- Other applications are fast adopting the self-attention architecture and self-supervised learning settings, e.g., computer vision, recommender systems, drug discovery, and so on
- As for natural language generation, self-attention models still require a greedy decoding of words one at a time.

References

- Harvard NLP - The Annotated Transformer
- Stanford University CS224n – Deep learning for Natural Language Processing
- Fully-parallel text generation for neural machine translation
- Convolution Sequence to Sequence
- The Illustrated Transformer (Eng)
- The Illustrated Transformer (Kor)