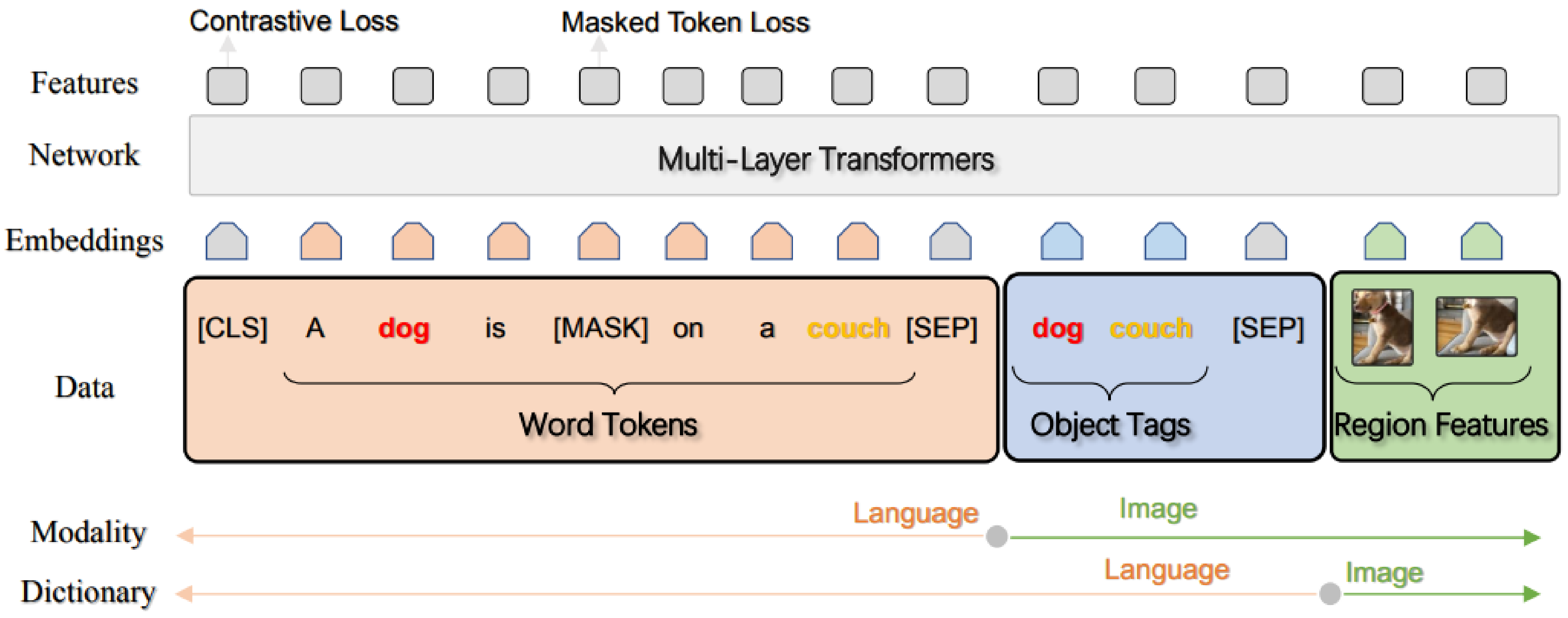
Transformer

Machine Learning Study
JinHo Kim

Contents

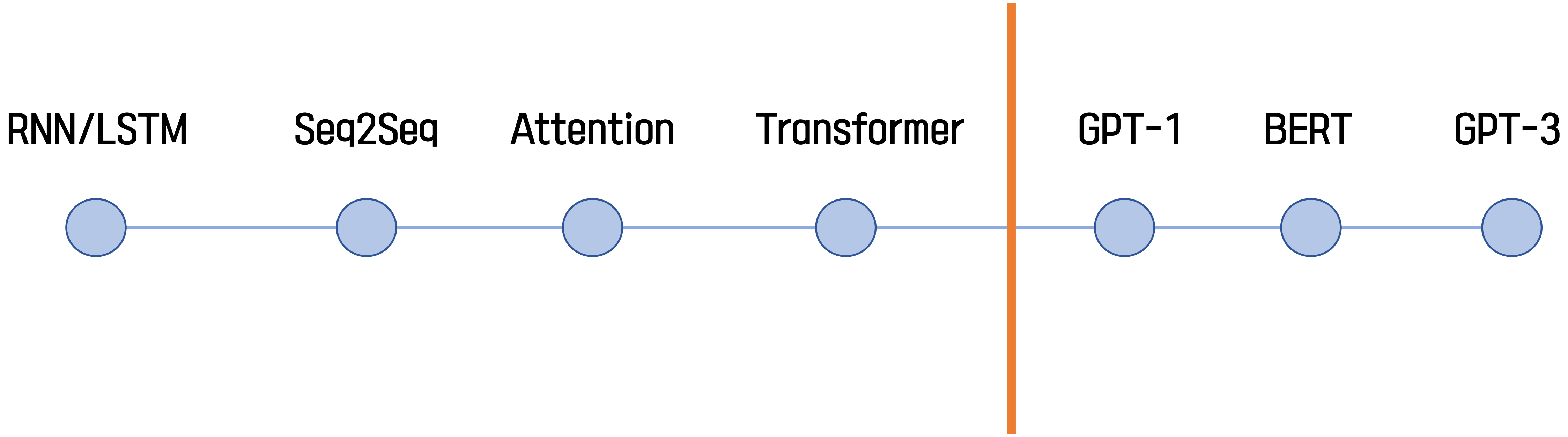
1. Introduction

OSCAR...?



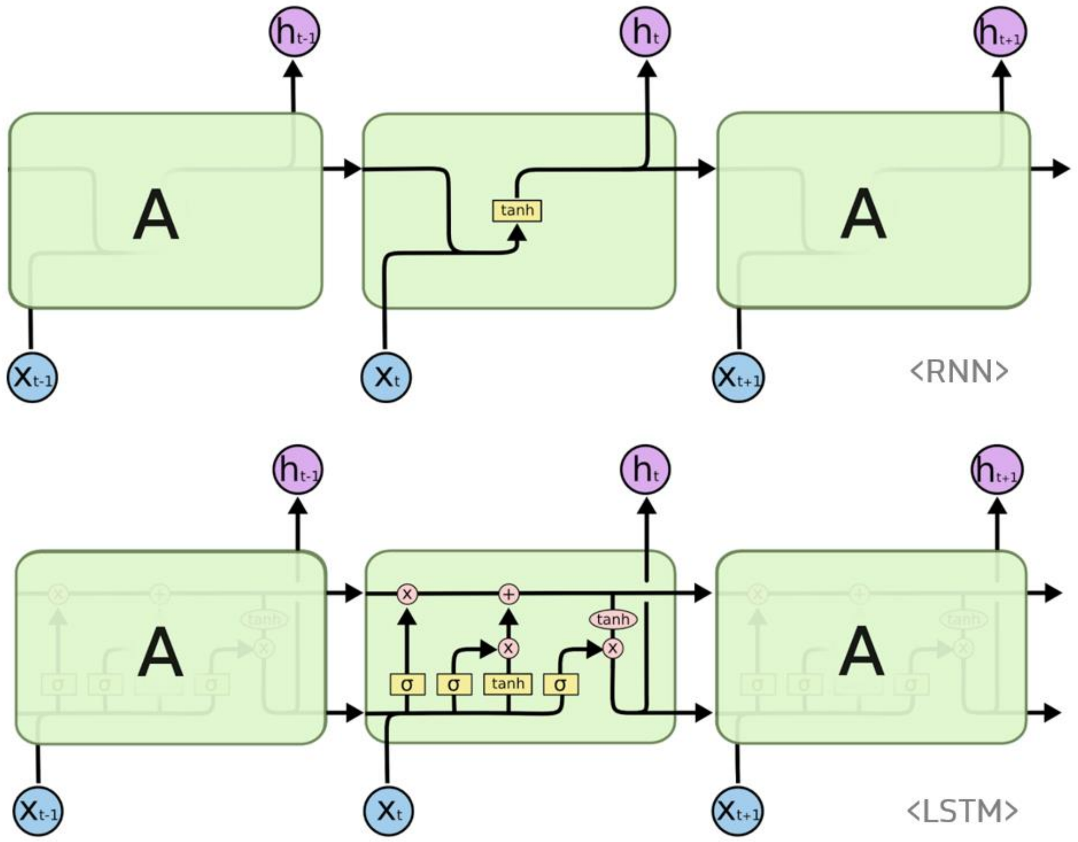
Contrastive Loss
Masked Token Loss
Features
Network
Multi-Layer Transformers
Embeddings
Data
[CLS] A dog is [MASK] on a couch [SEP]
Word Tokens
dog couch [SEP]
Object Tags
Region Features
Modality
Language
Image
Dictionary
Language
Image

NLP Model History



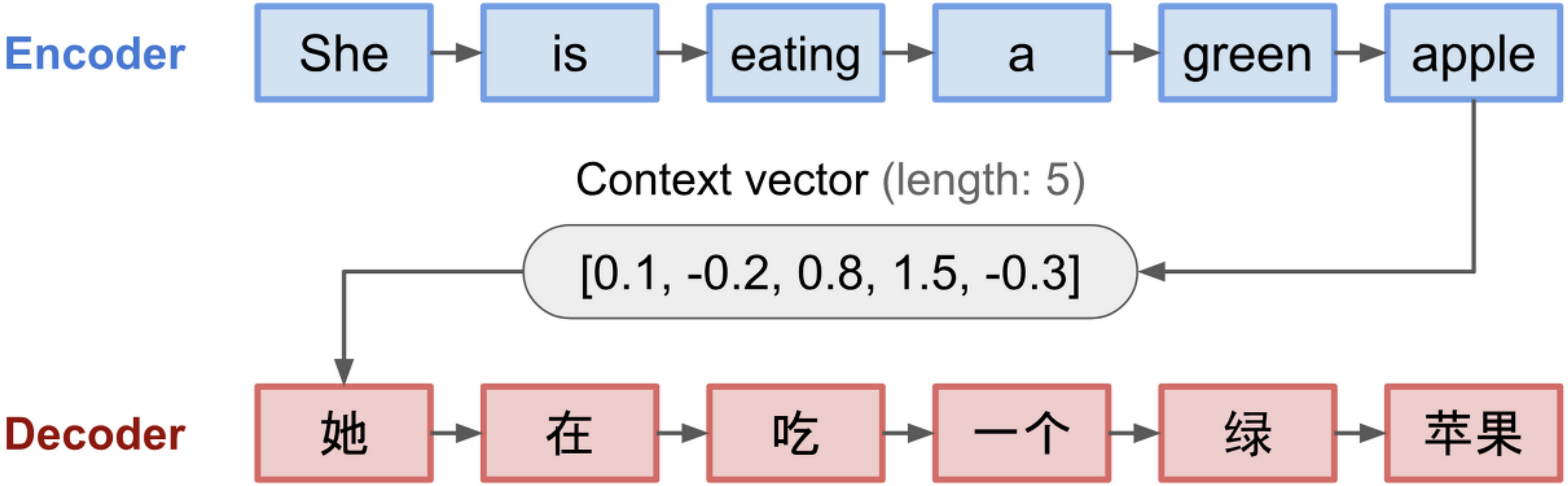
https://www.youtube.com/watch?v=AA621UofTUA (나동빈 유튜브)

RNN / LSTM



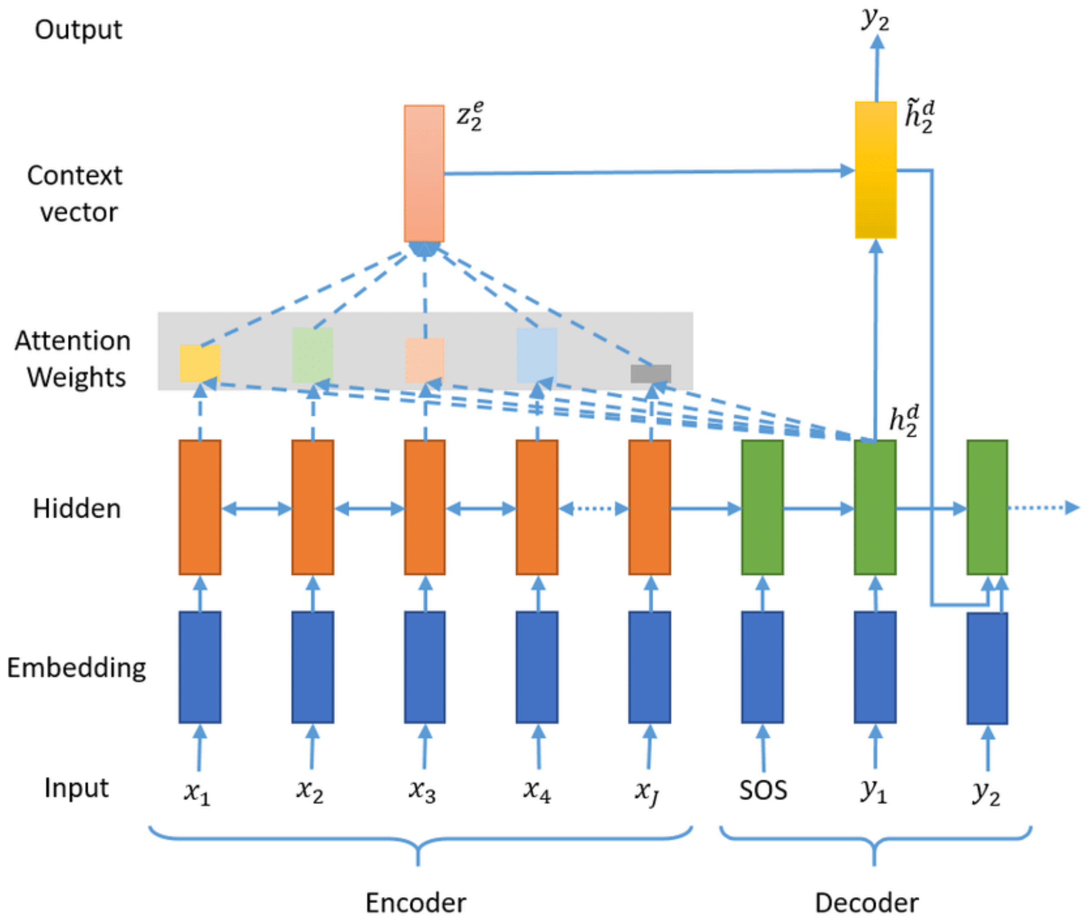
https://ratsgo.github.io/natural%20language%20processing/2017/03/09/rnnlstm/

Seq2Seq



https://exupery-1.tistory.com/209

Seq2Seq with Attention



https://velog.io/@jaehyeong/Basic-NLP1-Sequence-to-Sequence-with-Attention

2. Paper Review

‘Attention is All You Need’

‘Attention is All You Need’

the Transformer,
base solely on attention mechanism,
dispense with recurrence and convolutions entirely

→ establishes a new model **state-of-the-art**

‘Attention is All You Need’

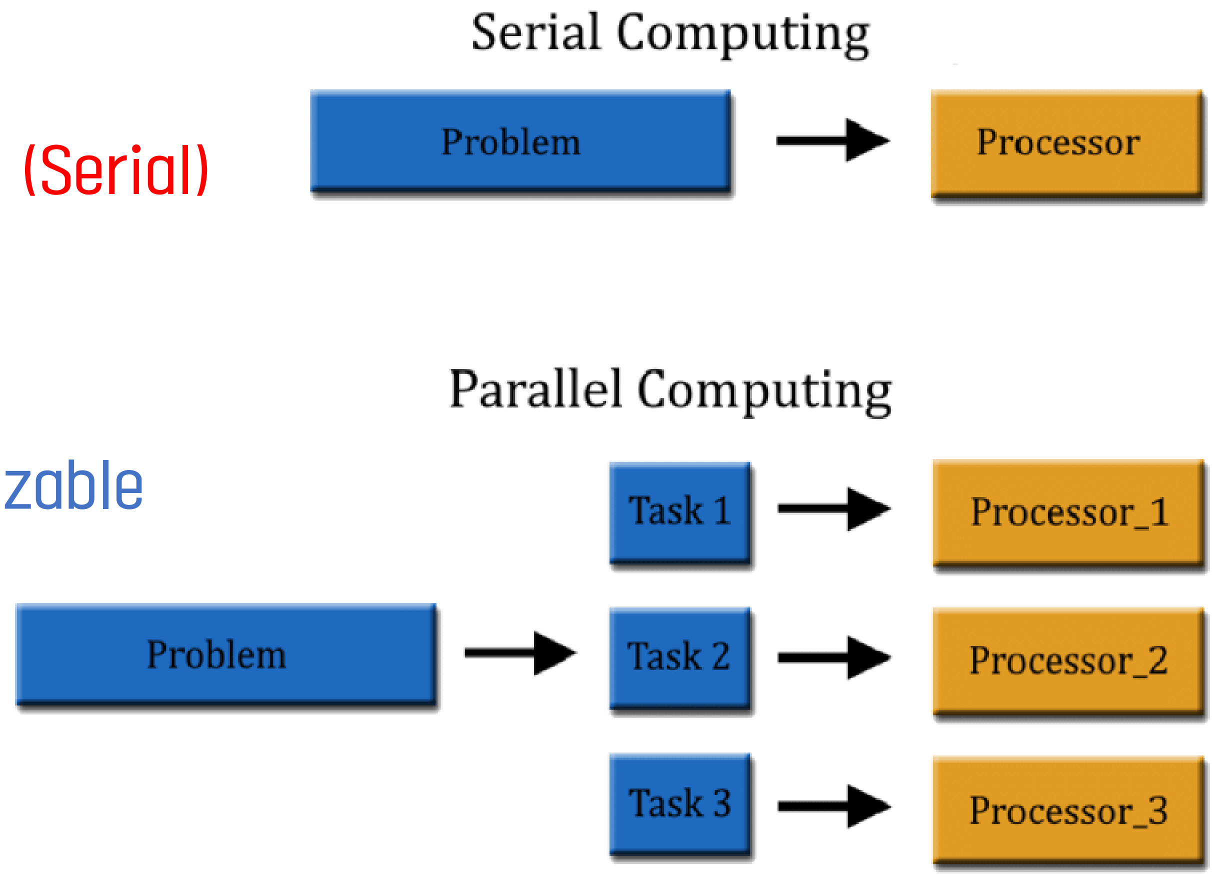
Parallelizable ? ⟷ Sequential ?

https://www.computerhope.com/jargon/p/parallelization.htm

‘Attention is All You Need’

Sequential (Serial)

Parallelizable



https://www.computerhope.com/jargon/p/parallelization.htm

Transformer architecture

Encoder

Decoder

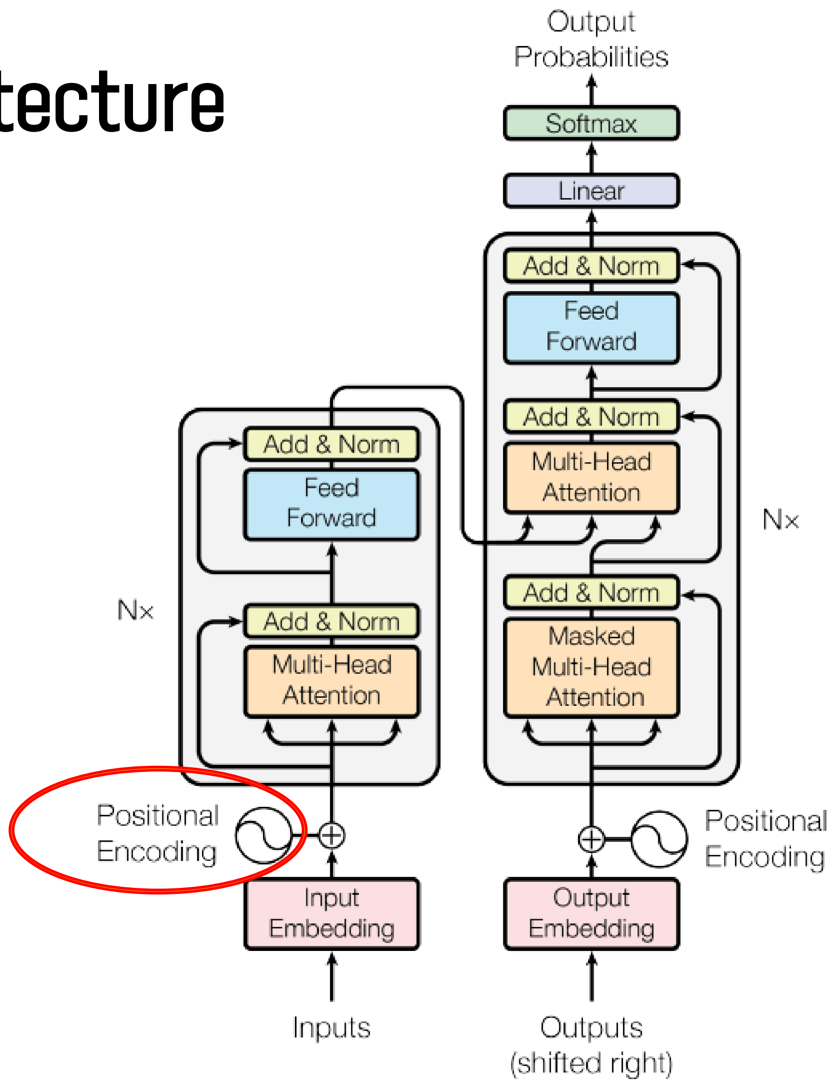
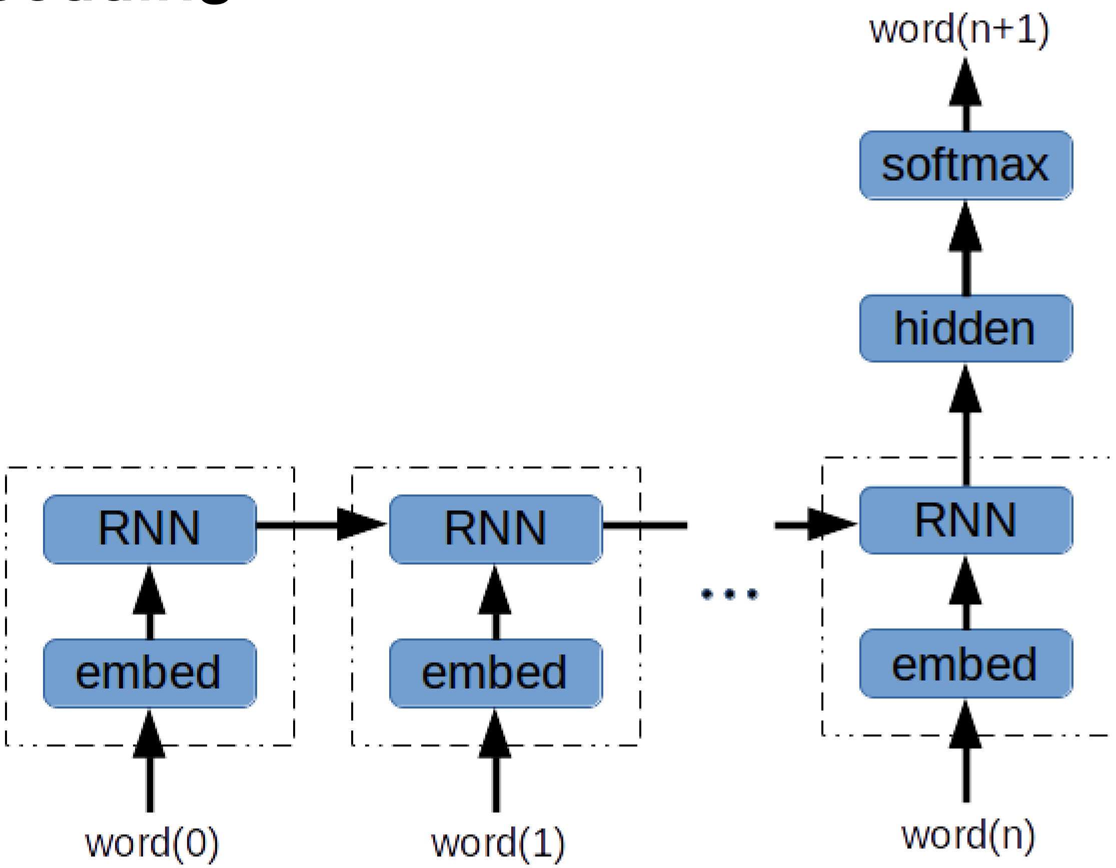


Figure 1: The Transformer - model architecture.

https://arxiv.org/abs/1706.03762

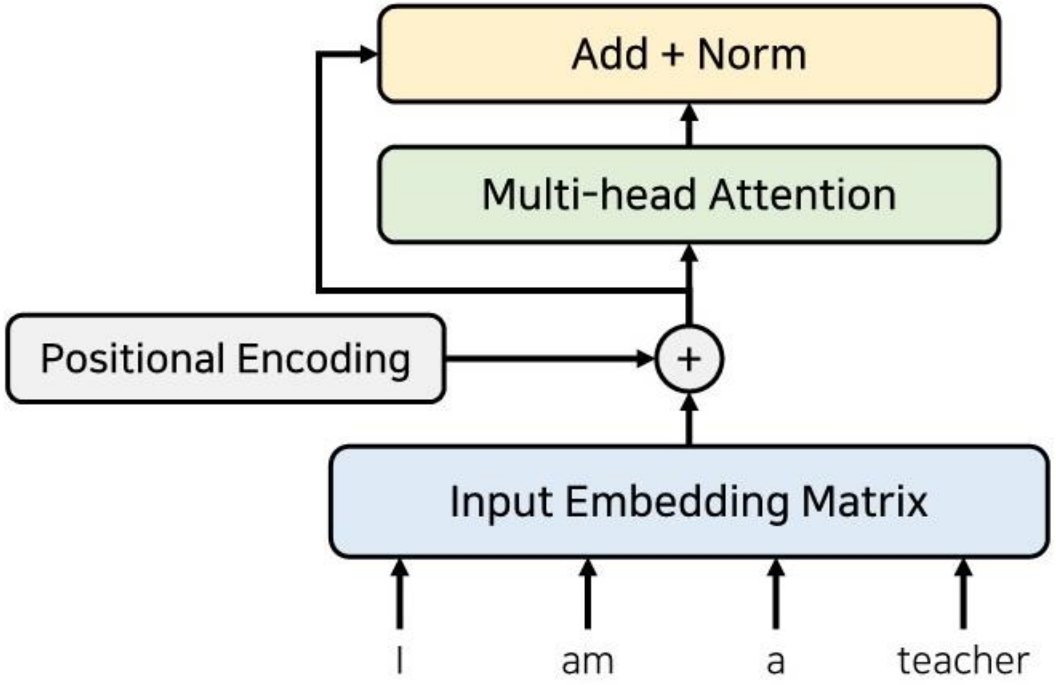
RNN - Word Embedding



word order!

https://pat-coady.github.io/rnn/

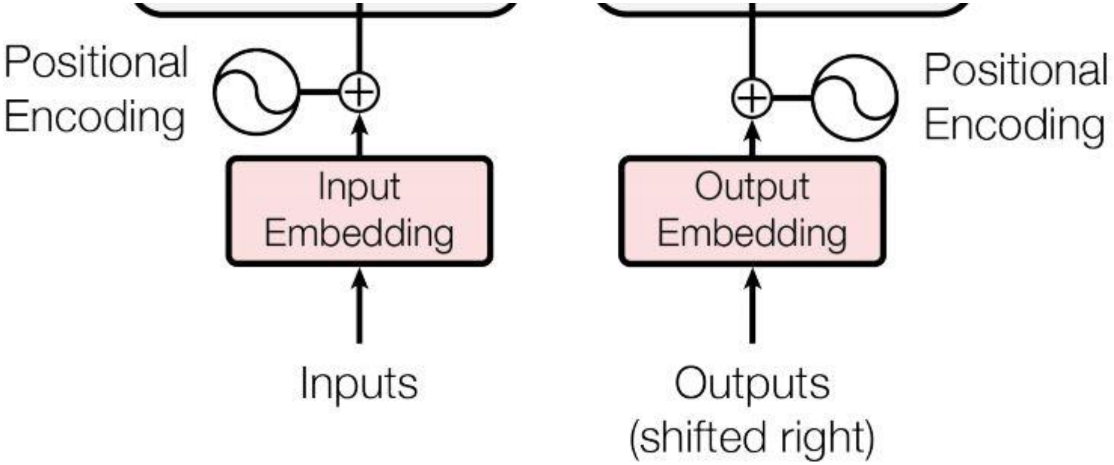
Postional Encoding



← 문장 자체를 한꺼번에 입력!

https://www.youtube.com/watch?v=AA621UofTUA (나동빈 유튜브)

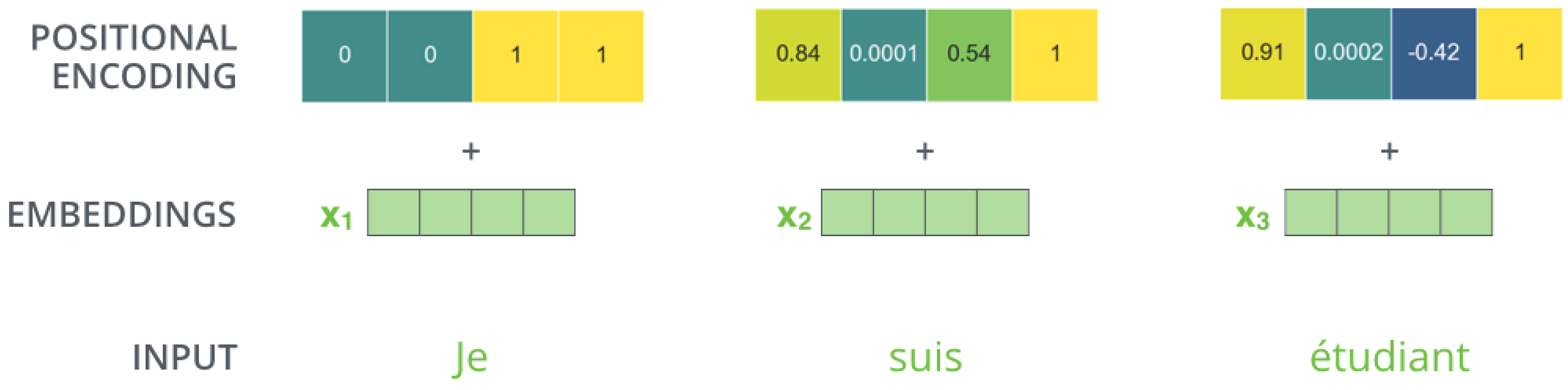
Postional Encoding



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

https://arxiv.org/abs/1706.03762

Postional Encoding



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

Transformer architecture

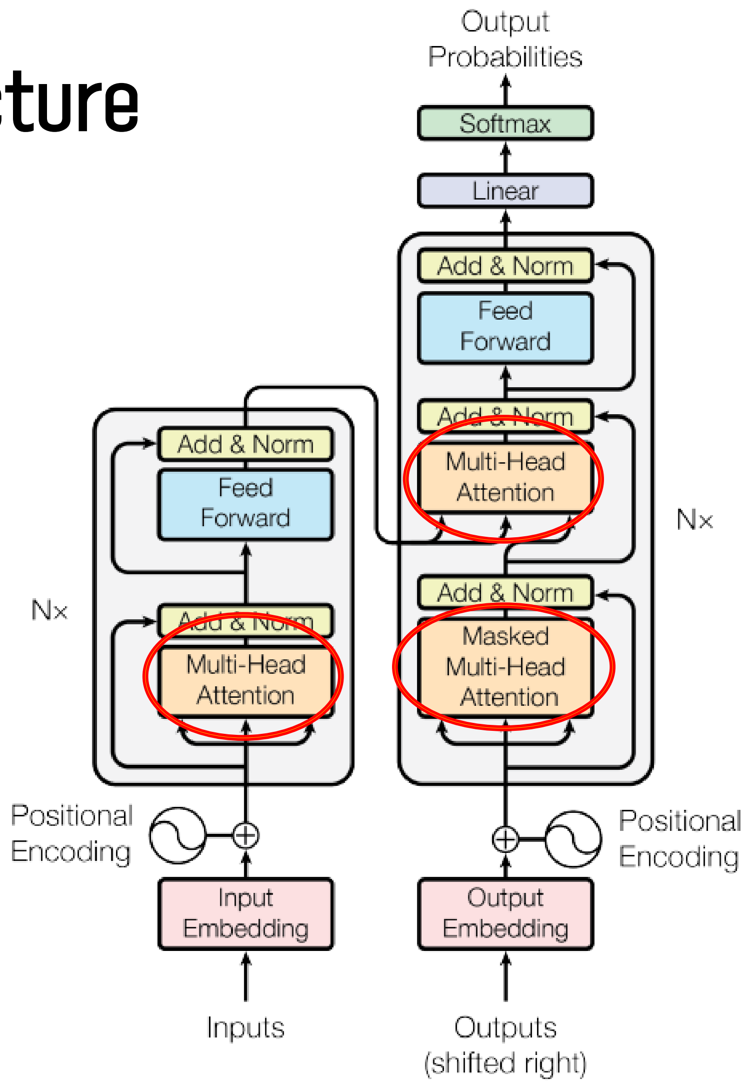
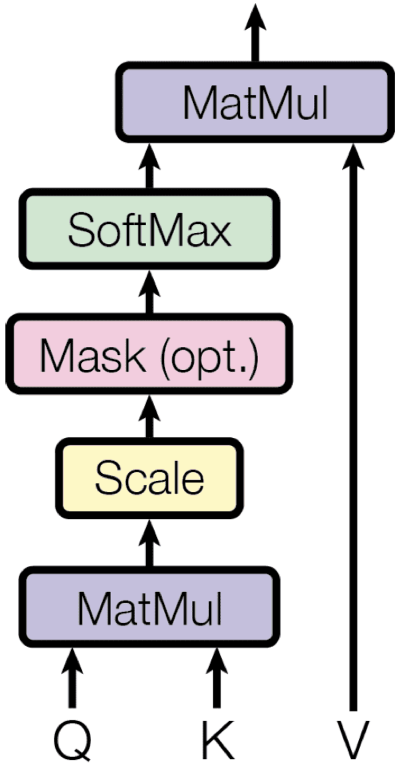


Figure 1: The Transformer - model architecture.

https://arxiv.org/abs/1706.03762

Scaled Dot-Product Attention



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

https://arxiv.org/abs/1706.03762

Dot-Product Attention

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

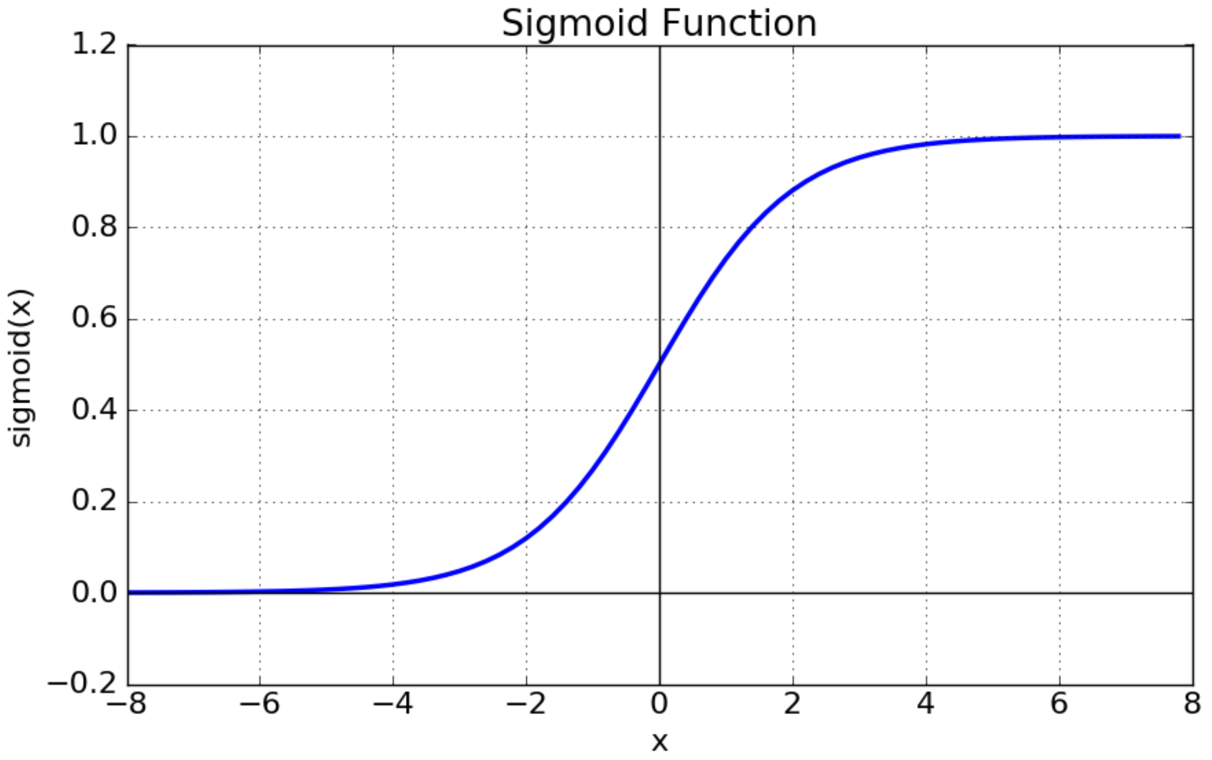
$$\cancel{\frac{1}{\sqrt{d_k}}}$$

'Scale'

extremely small gradients

https://arxiv.org/abs/1706.03762

Sigmoid function



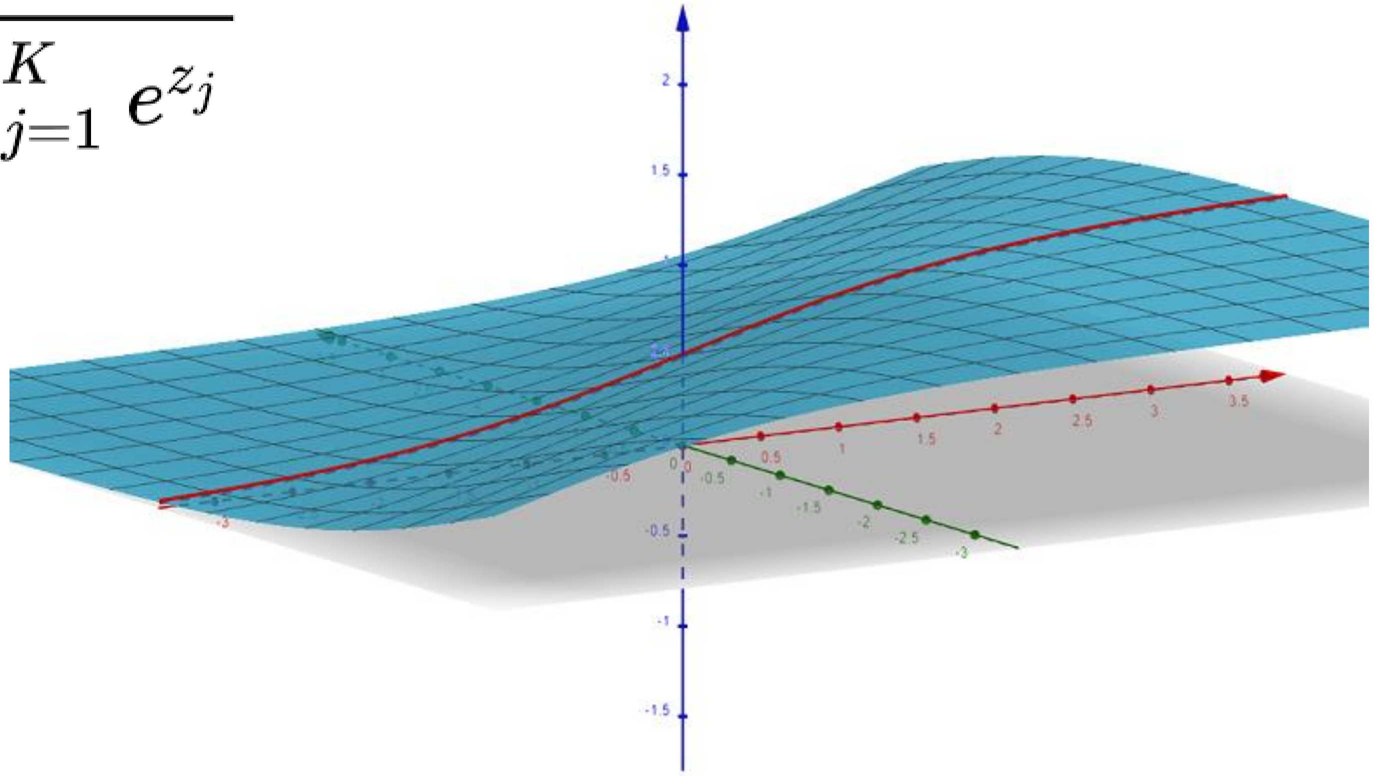
$$S(x) = \frac{1}{1+e^{-x}} = \frac{e^x}{e^x+1}$$

https://themaverickmeerkat.com/2019-10-23-Softmax/

Softmax function

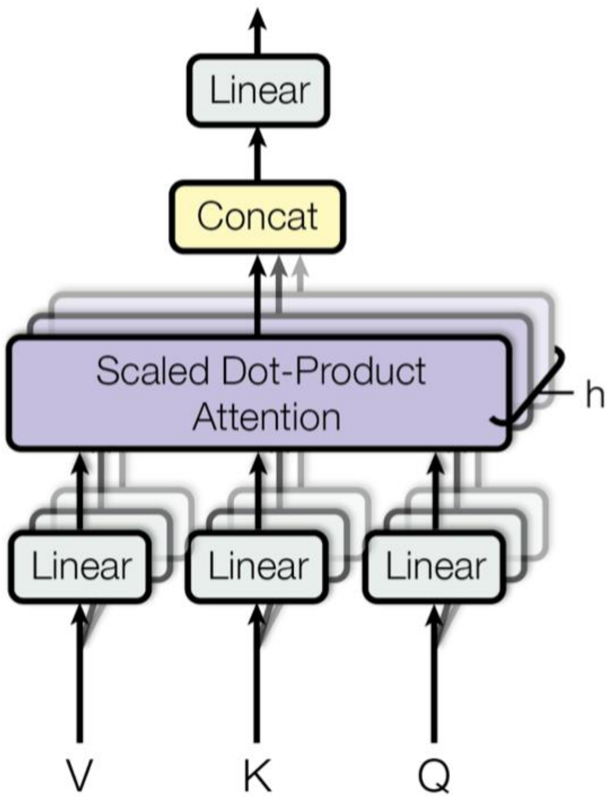
$$\mathrm{Attention}(Q, K, V) = \mathrm{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$



https://themaverickmeerkat.com/2019-10-23-Softmax/

Multi-Head Attention



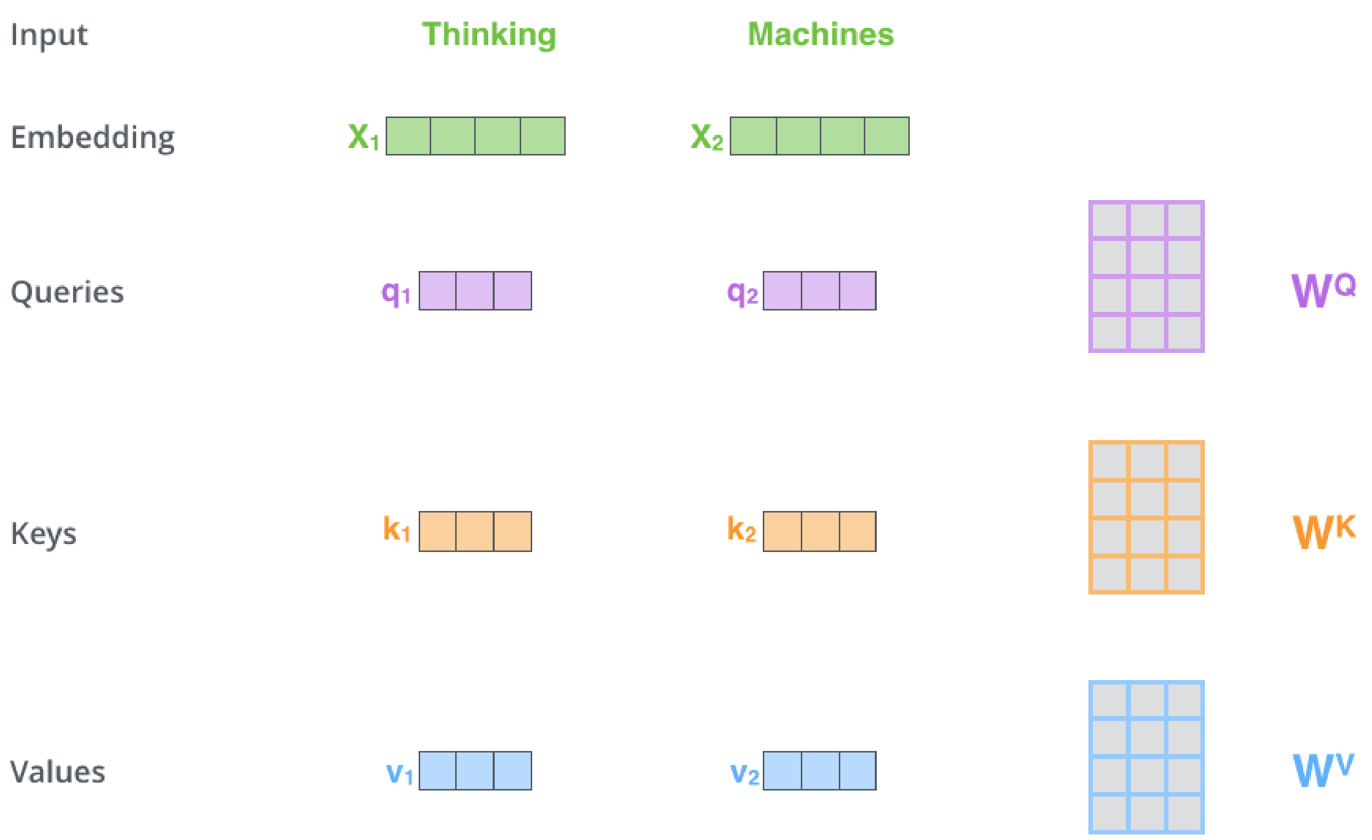
$$\mathrm{MultiHead}(Q, K, V) = \mathrm{Concat}(\mathrm{head_1}, ..., \mathrm{head_h})W^O$$
$$\text{where } \mathrm{head_i} = \mathrm{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

head : h * scaled dot-product attention

model parameter → h = 8 (base)

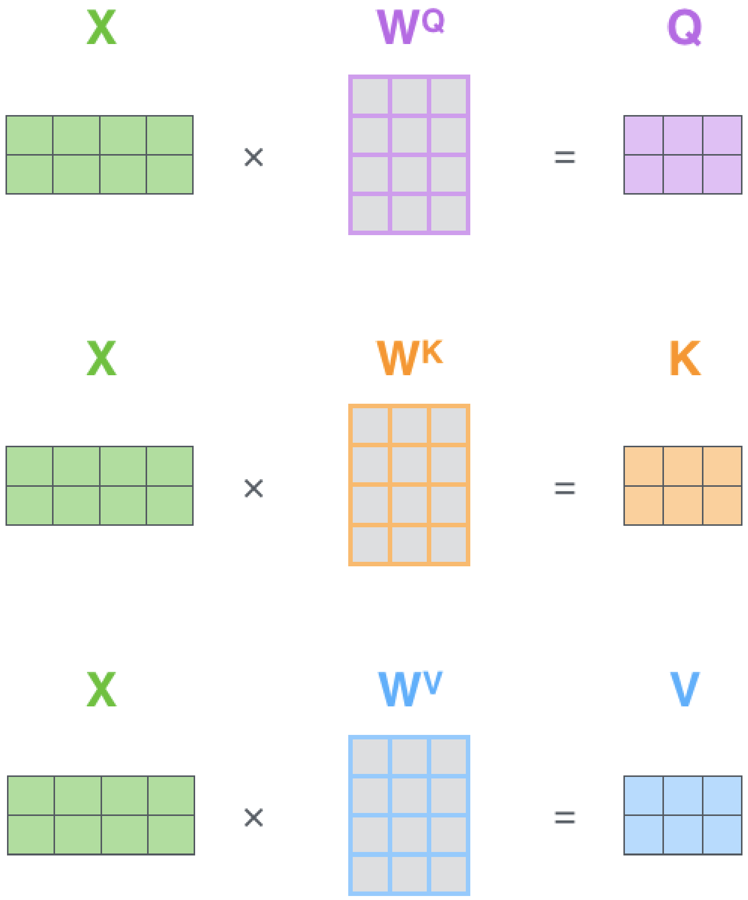
https://arxiv.org/abs/1706.03762

Scaled Dot-Product Attention (1)



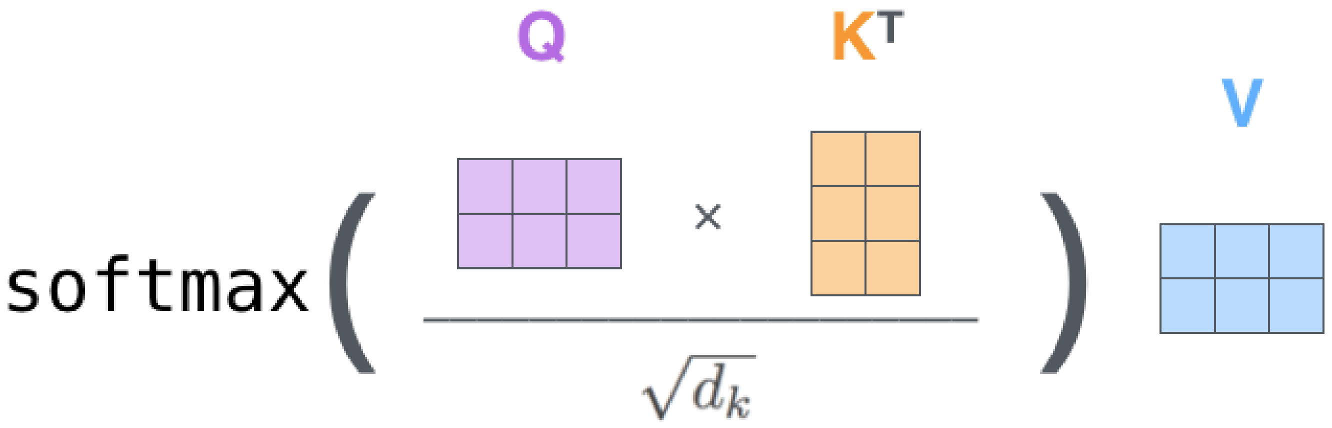
https://jalammar.github.io/illustrated-transformer/

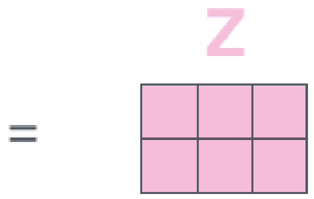
Scaled Dot-Product Attention (2)



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

Scaled Dot-Product Attention (3)



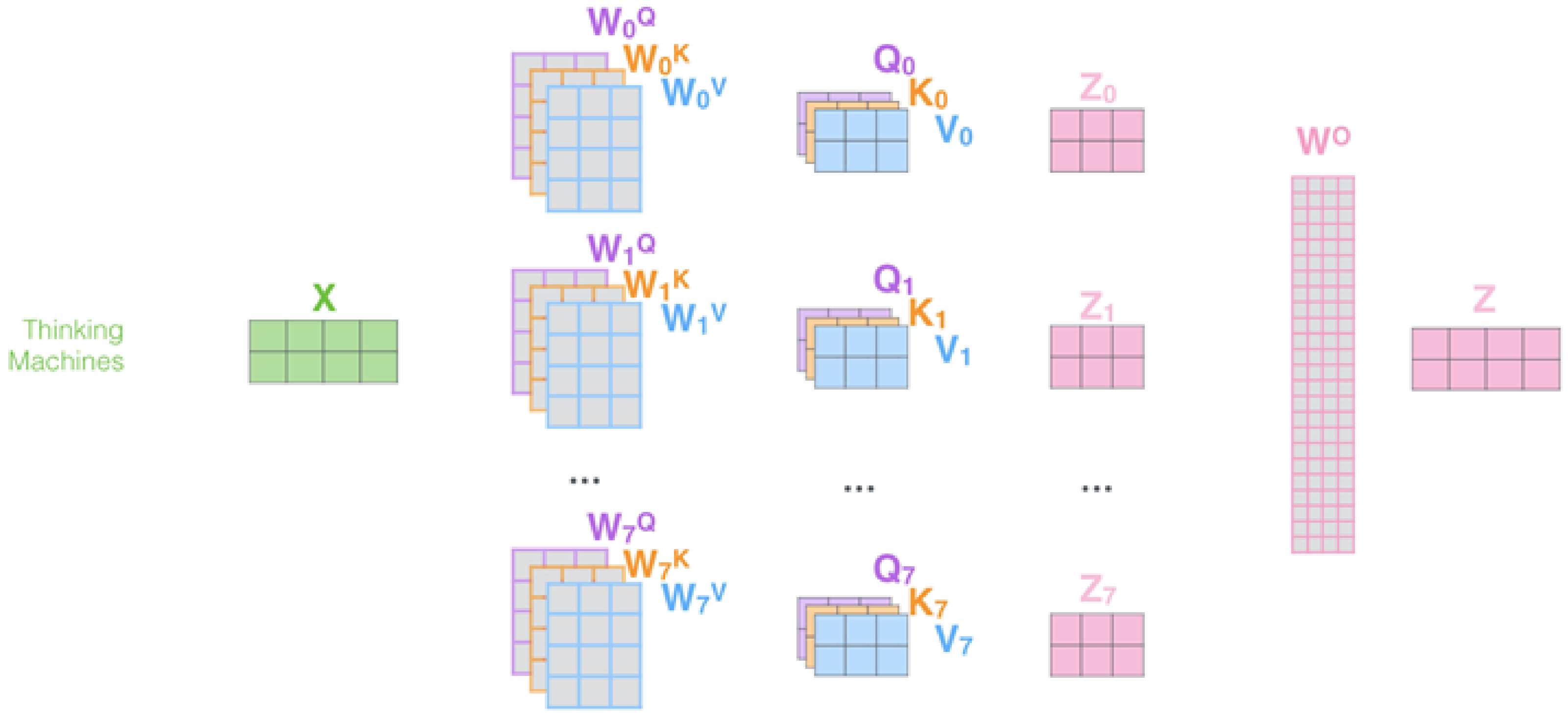


$$\mathrm{Attention}(Q, K, V) = \mathrm{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

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

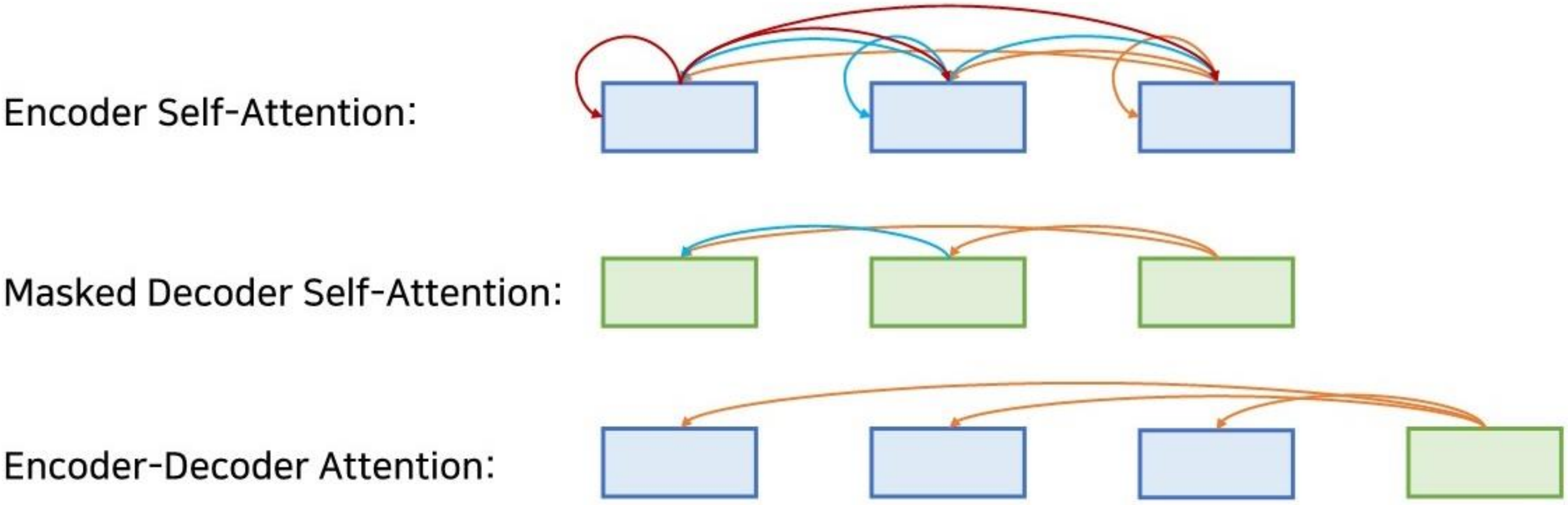
Multi-Head Attention

$$\mathrm{MultiHead}(Q,K,V) = \mathrm{Concat}(\mathrm{head_1}, \dots, \mathrm{head_h})W^O$$
$$\text{where head}_{\mathrm{i}} = \mathrm{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$



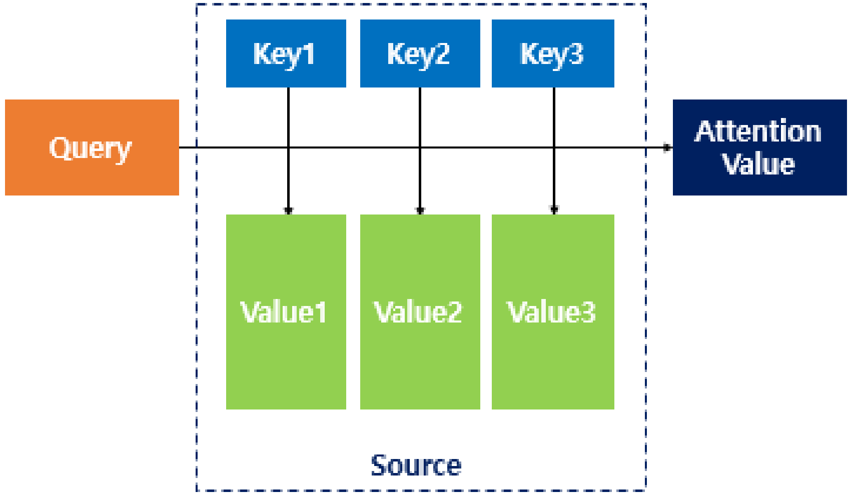
https://jalammar.github.io/illustrated-transformer/

multi-head attention in 3 different ways :



https://www.youtube.com/watch?v=AA621UofTUA (나동빈 유튜브)

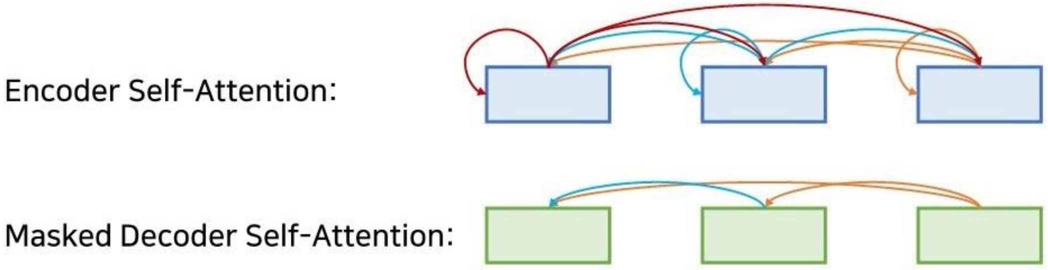
Self-Attention



Attention Mechanism

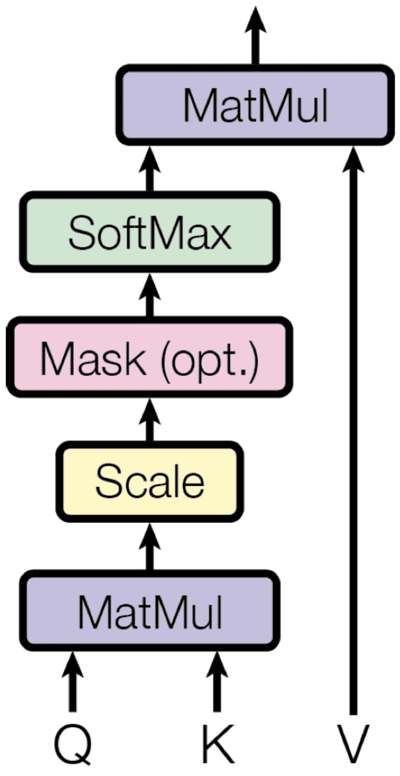
In Self-Attention,

Query = Key = Vector

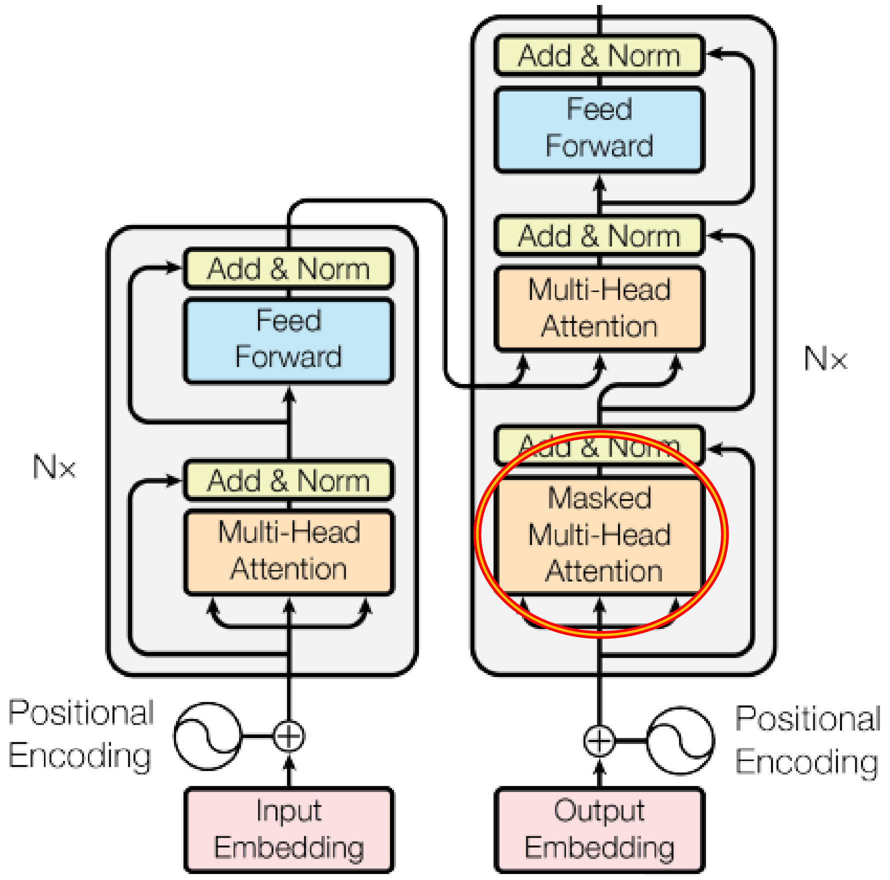


https://jalammar.github.io/illustrated-transformer/

'Masked' Self-Attention

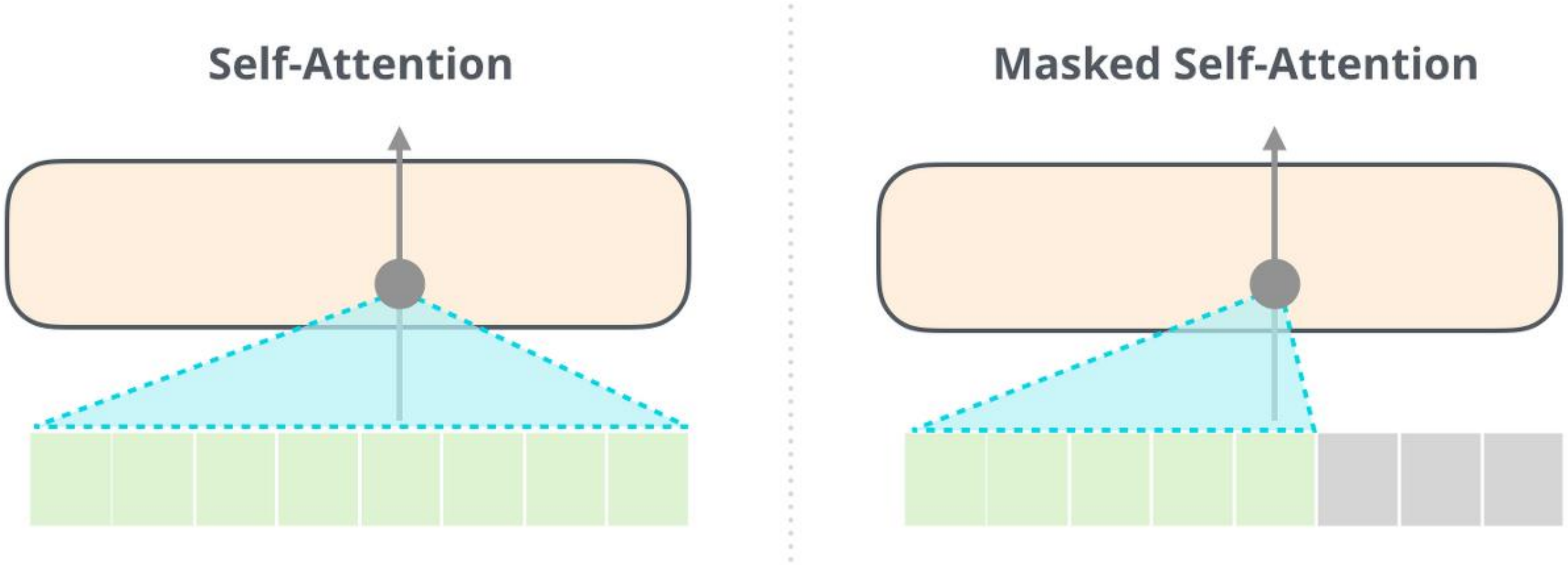


Scaled Dot-Product Attention



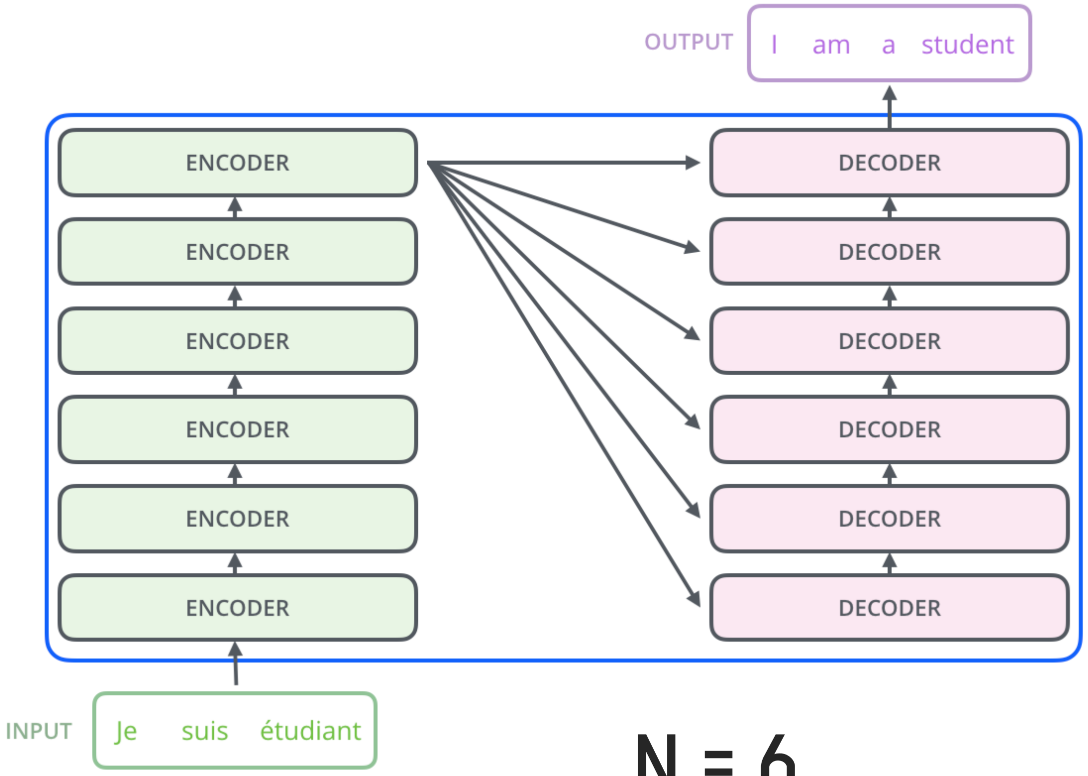
https://jalammar.github.io/illustrated-gpt2/

‘Masked’ Self-Attention

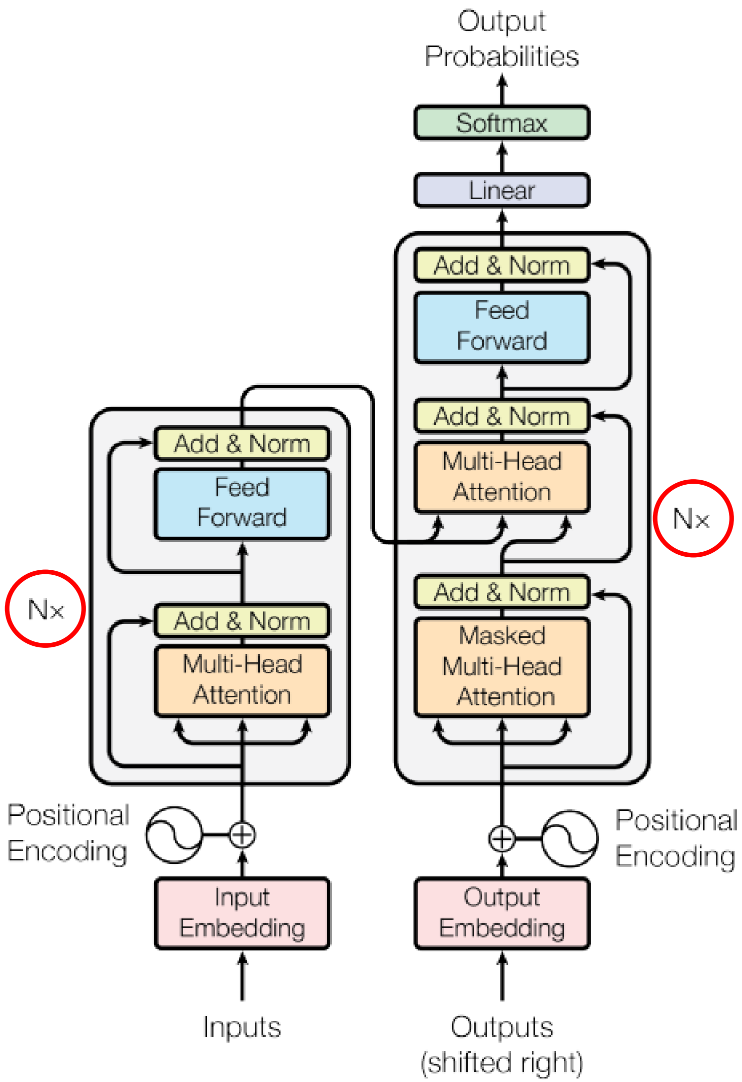


https://jalammar.github.io/illustrated-gpt2/

Encoder Decoder Attention



N = 6



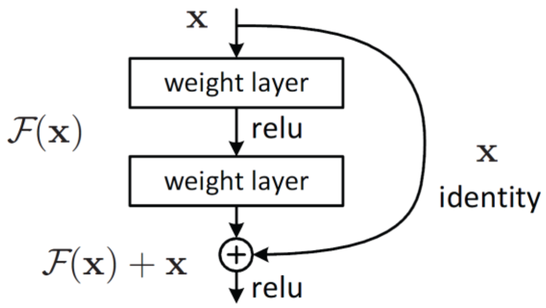
https://jalammar.github.io/illustrated-transformer/

Feed Forward network

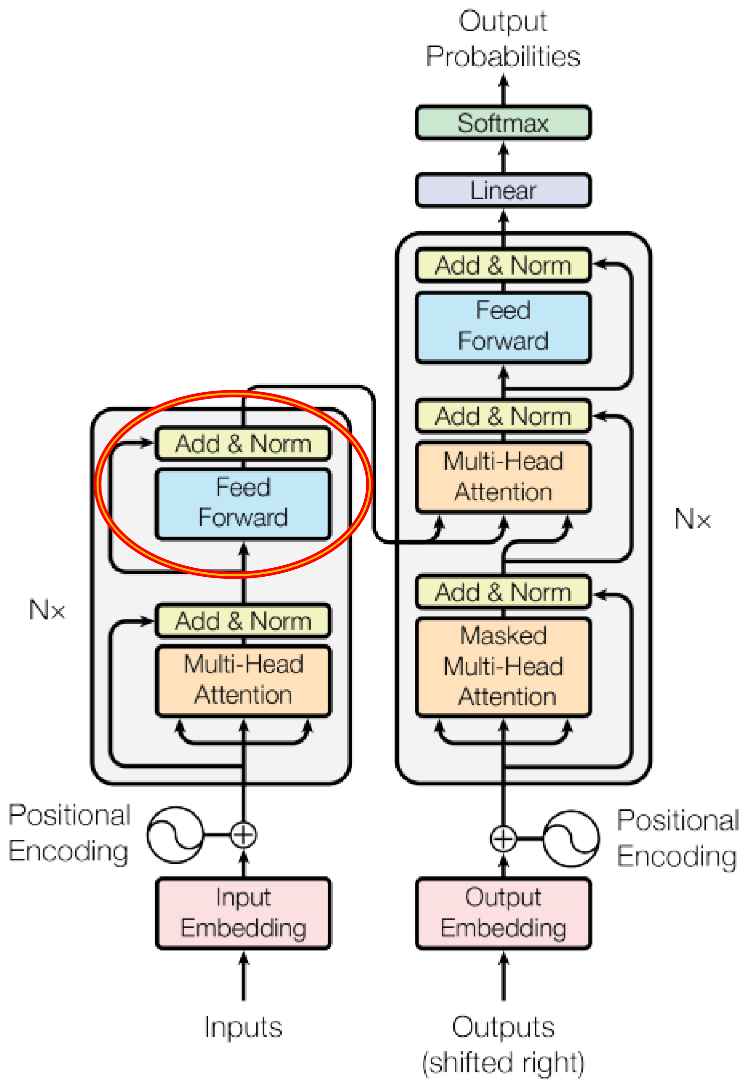
$$\mathrm{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2$$

ReLU Function → ELU, GeLU, etc.,

Residual connection



In ResNet



https://arxiv.org/abs/1706.03762

3. Code Practice

QnA?

Thank You!