Attention Is All You Need

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin
(Google Brain, Google Research)

NIPS '17

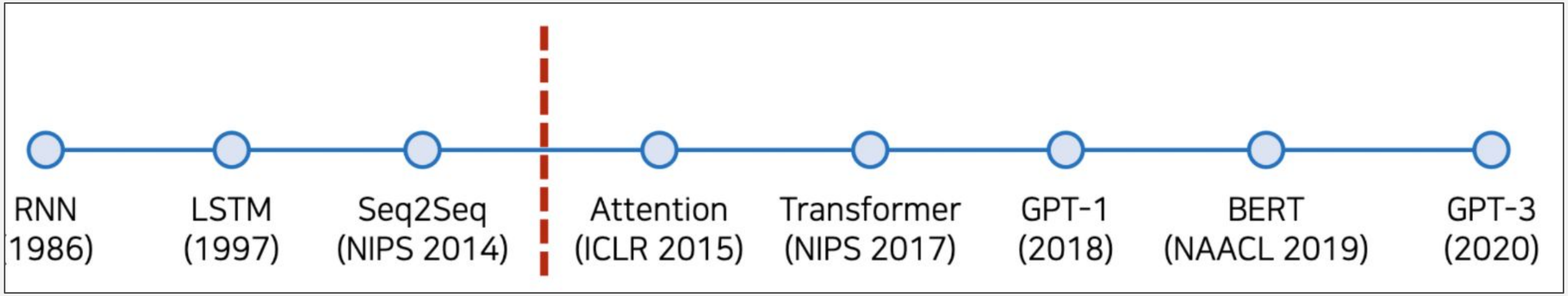
Date 09 June 2025

Wonseok

Table of Contents

Background

- Machine Translation
  - RNN and LSTM were firmly known as SOTA approaches
    - Seq2Seq model (encoder-decoder architecture)
  - Limitations of RNN
    - Sequential nature of RNN models
    - Vanishing gradients on long sequences
    - Difficult in accessing information from long time ago

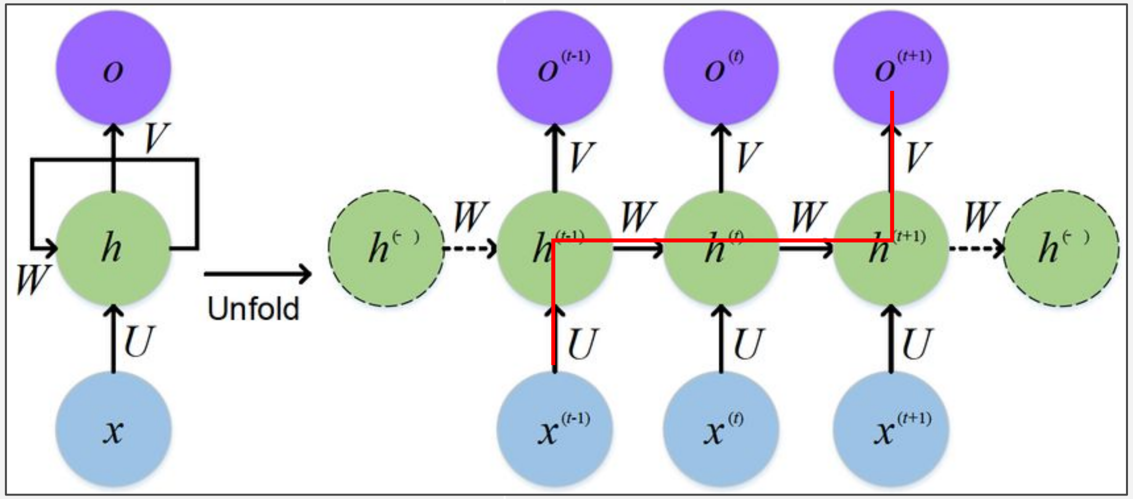


*SOTA: State Of The Art

RNN and LSTM

- RNN (Recurrent Neural Network)

$$h^{(t)} = \tanh(Ux^{(t)} + Wh^{(t-1)})$$
$$o^{(t)} = \text{softmax}(Vh^{(t)})$$



  - $Wh^{(t-1)}$: What to remember from the past
  - $Ux^{(t)}$: How to process the new input
  - *ex) **input**: "I ate pizza" -> **output**: "난 피자를 먹었다"*

  ⇢ Sequential nature precludes parallelization

- LSTM (Long-Short Term Memory)
  - Gates mechanism to mitigate vanishing gradients problem

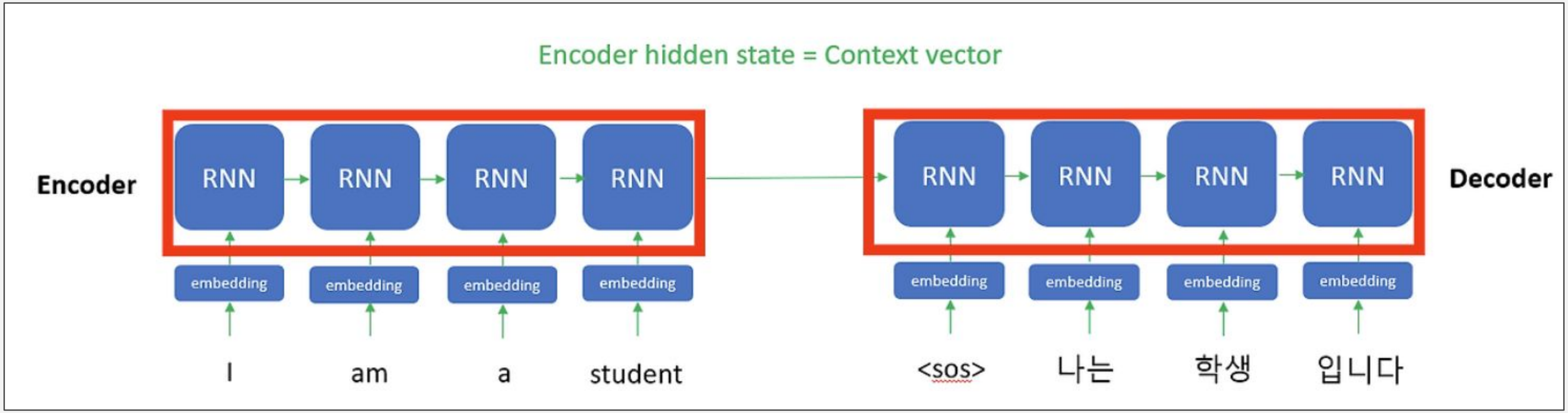
- Limitation
  - Input and output must have the same length

    ⇢ **<u>Seq2Seq</u>**

*SOTA: State Of The Art

Seq2Seq

- Encoder And Decoder
  - LSTM stacks for different purpose each
    - Encoder: Compresses all the inputs into a final **context vector**
    - Decoder: Generate the output sequence with context vector as initial hidden state
  - Context vector alone can't hold enough of long sentences
    - Refer to all input sequences at every decoder output ⇢ 'Seq2Seq with **Attention**'

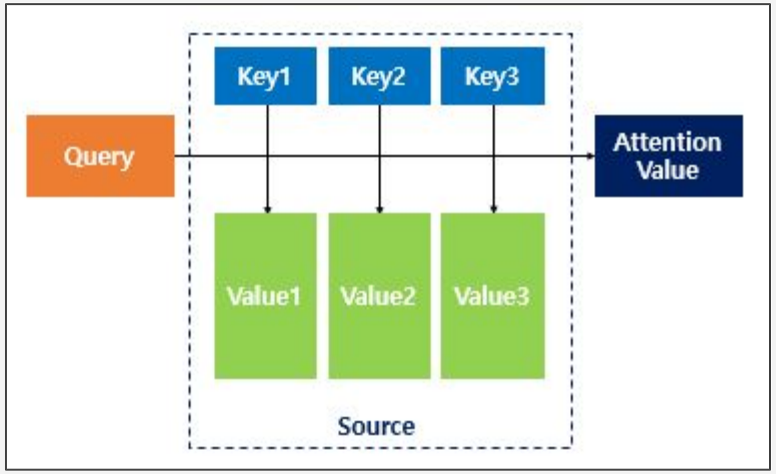


*SOTA: State Of The Art

Attention

- Idea

$$\boxed{\mathrm{Attention}(Q,K,V) = \mathrm{Softmax}(QK)V^T}$$



s h

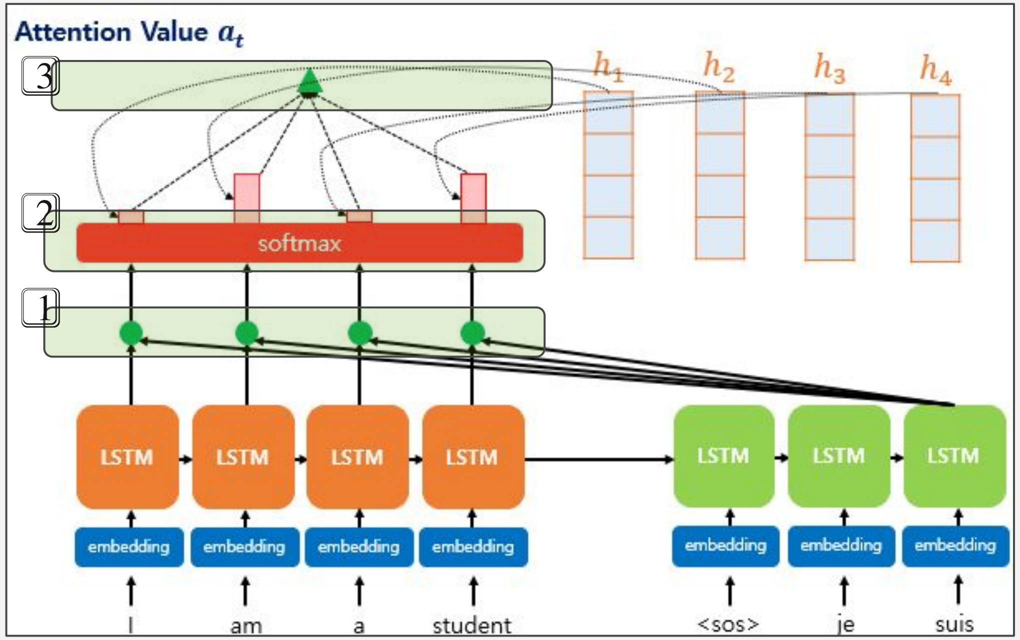
  - 1. [Attention Score] Similarity between Query and Key
  - 2. [Attention Weight] Normalized similarity between Q and K

$$\boldsymbol{a}_t = \mathrm{softmax}\left(\left[s_{t-1}^{\top}h_1,\ s_{t-1}^{\top}h_2,\ \ldots\right]\right)$$

  - 3. [Attention Value] Weighted sum of Value (vector)

$$c_t = \sum_{i=1}^{4} \alpha_{ti}\cdot h_i = \underbrace{\alpha_{t1}\cdot h_1}_{\in\mathbb{R}^d} + \underbrace{\alpha_{t2}\cdot h_2}_{\in\mathbb{R}^d} + \underbrace{\alpha_{t3}\cdot h_3}_{\in\mathbb{R}^d} + \underbrace{\alpha_{t4}\cdot h_4}_{\in\mathbb{R}^d}$$

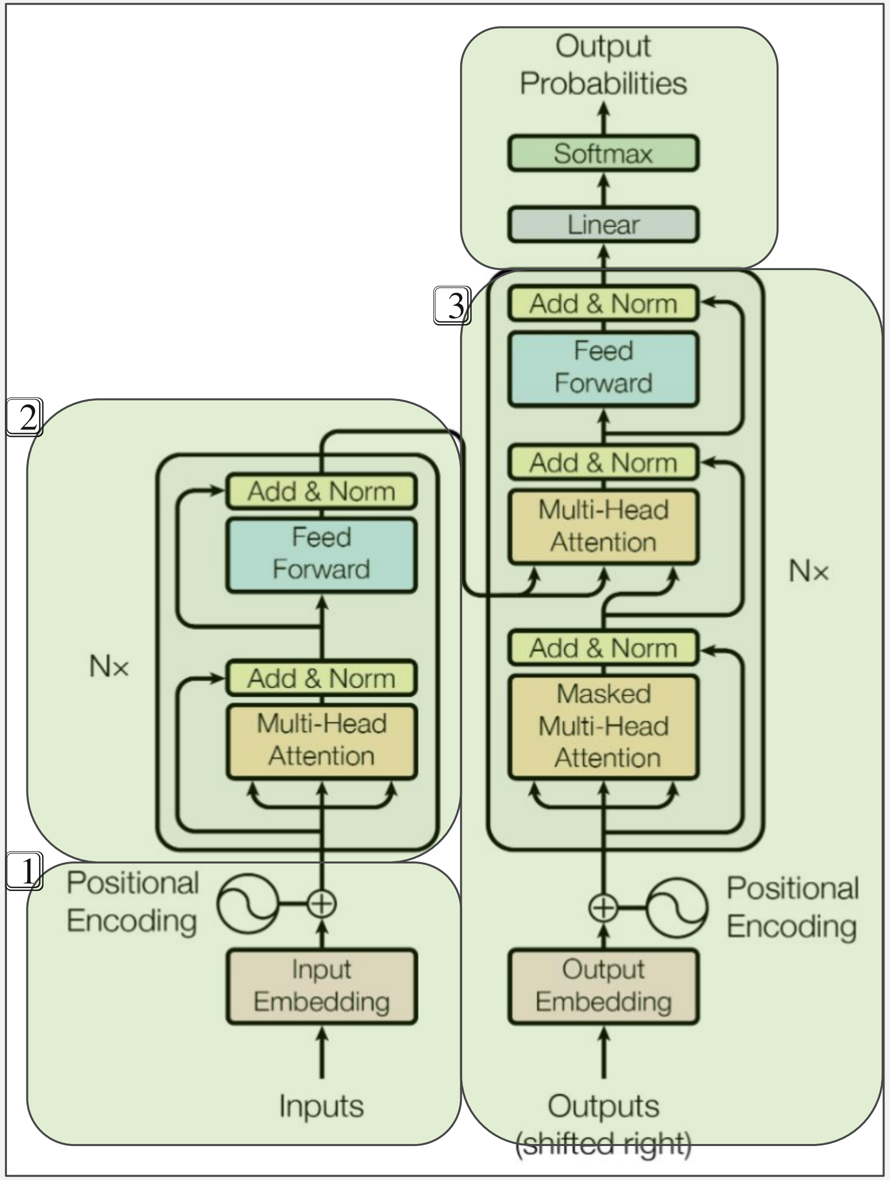
⇢ All input sequences are considered,
weighted by similarity between input and current output



Transformer

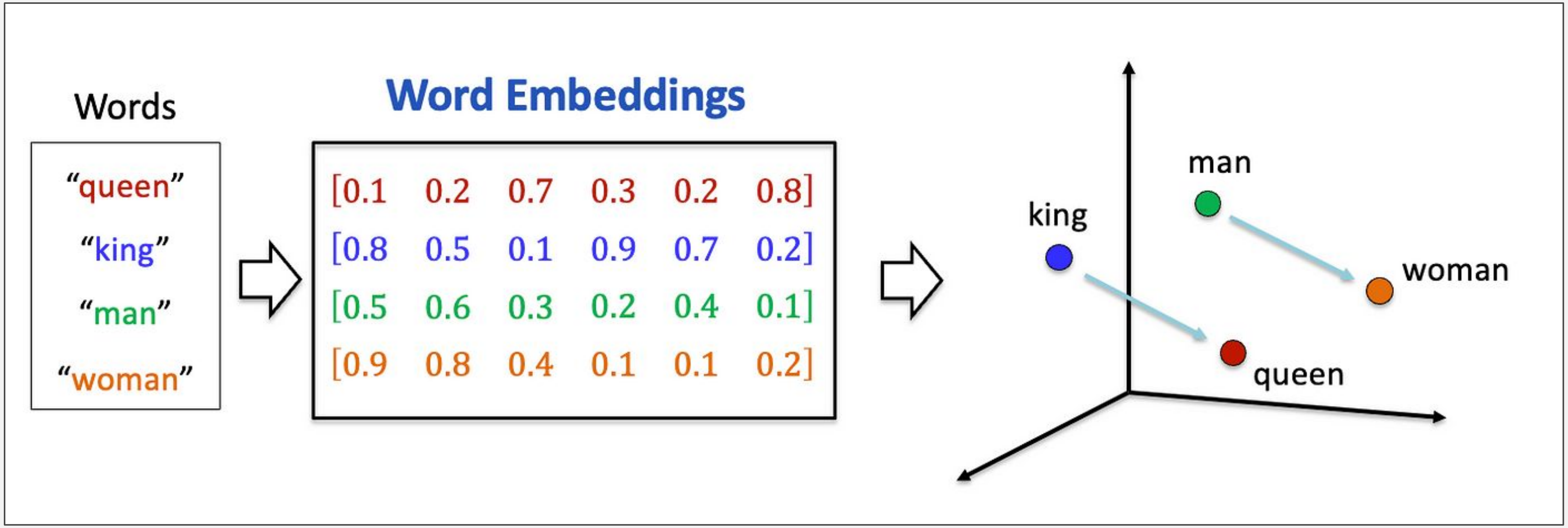
Fully dependent on 'attention' without RNN/LSTM

- 1. Input
  - Word Embedding
  - Positional Encoding
- 2. Encoder
  - Multi-head Attention
  - Feed Forward Networks
- 3. Decoder
  - Masked Attention
- 4. Output Layer



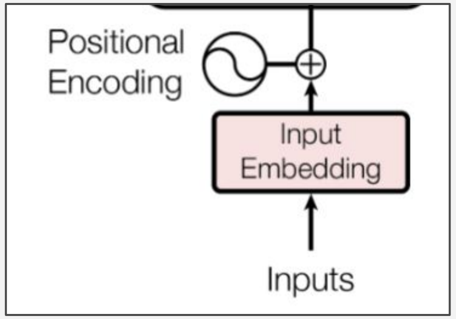
Word Embedding

- Word Embedding
  - Learned embeddings to convert tokens to a vector (d_{model} = 512 in this paper)
  - The more similar words, the closer the embeddings are



Positional Encoding

- Positional Encoding
  - No RNN or CNN → no order of the sequence
    - Need to insert positional info in some way
  - The relative/absolute position of the tokens
    - Each token position must have a unique value
    - Differences between tokens should have consistent meaning
  - Sine and cosine functions in this paper
    - To be able to apply on arbitrary length of sequence and word-embedding



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

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

pos

[“I”, “am”, “a”, “student”] → pos from the sequence

i

“I” = [-0.3, 0.2, -0.5, 0, …] → i from the word embedding

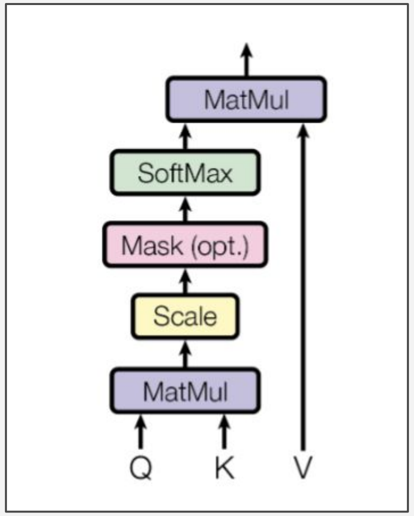
$$\text{Input} = \text{word} + PE_{(pos,i)}$$

Attention

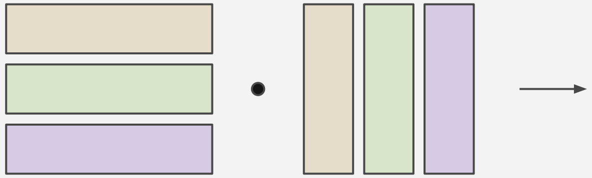
- Scaled Dot-Product

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

  - QT (Attention Score) grows large with dimensionality d_k
    - $d_k = d_q = d_v$ in this paper
    - Softmax outputs are pushed toward 1

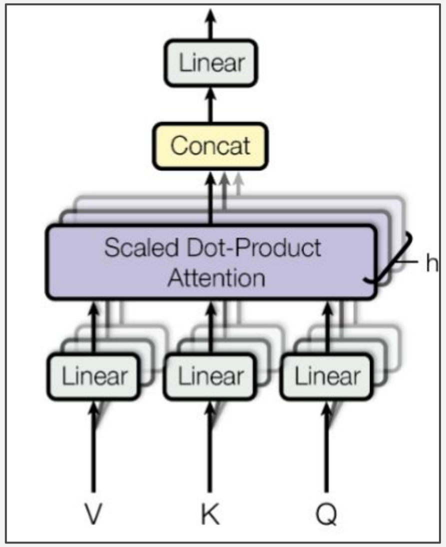


- Multi-Head Attention
  - Attention is not sequential
    - Matrix chunks + Parallel computing + Concatenation



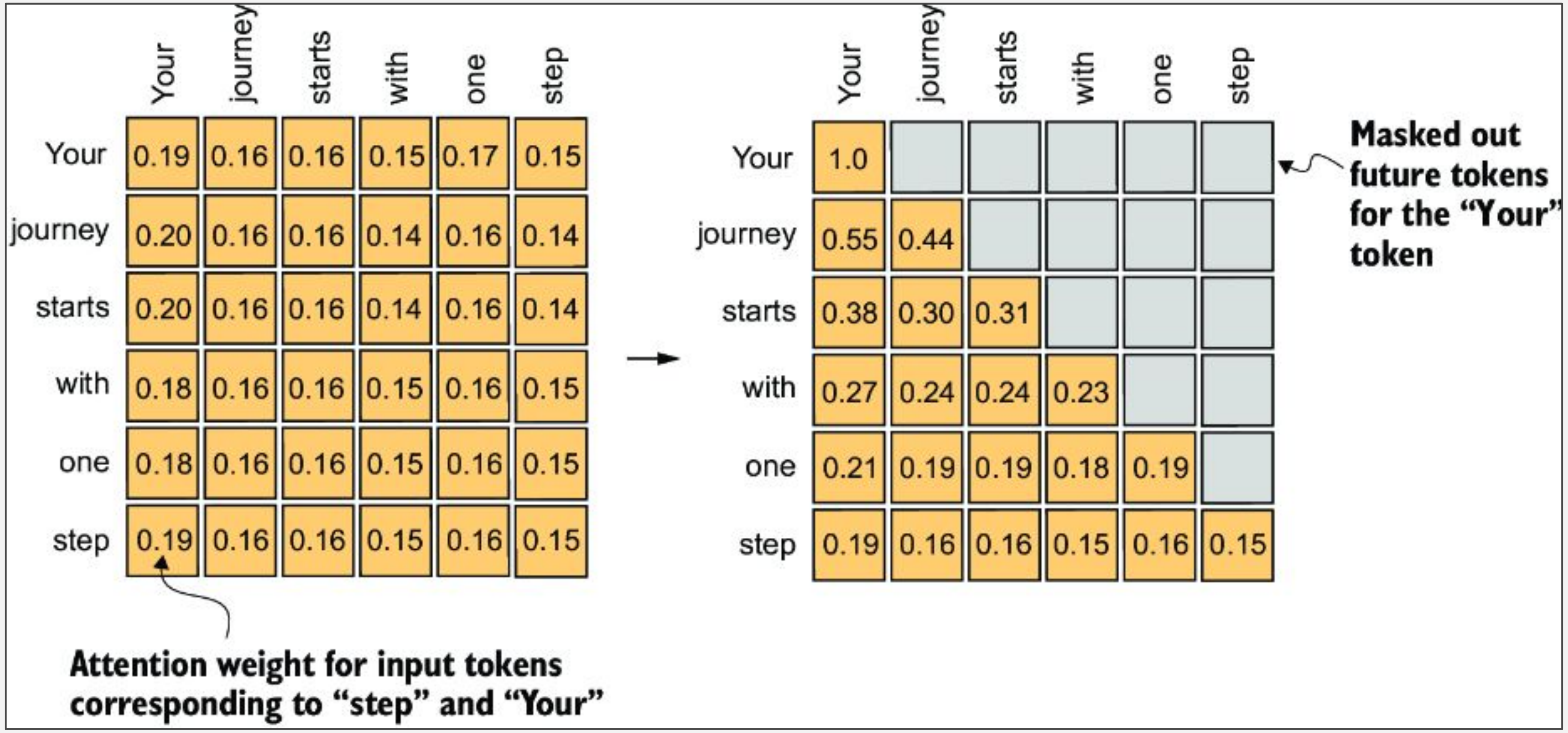
Head = 3

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



Masking

- Infinite Attention Score for later tokens
  - Prevent decoder from looking ahead at future tokens
  - Attention value after apply softmax becomes **0**

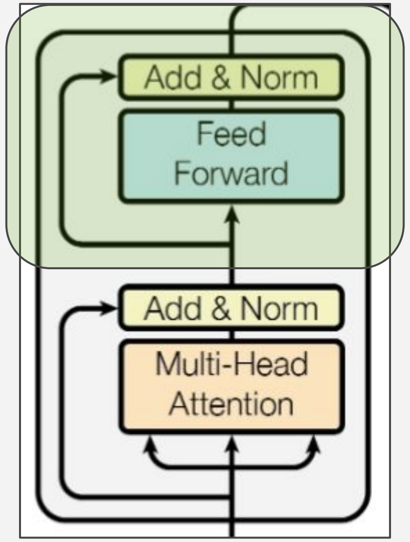


Feed Forward Networks

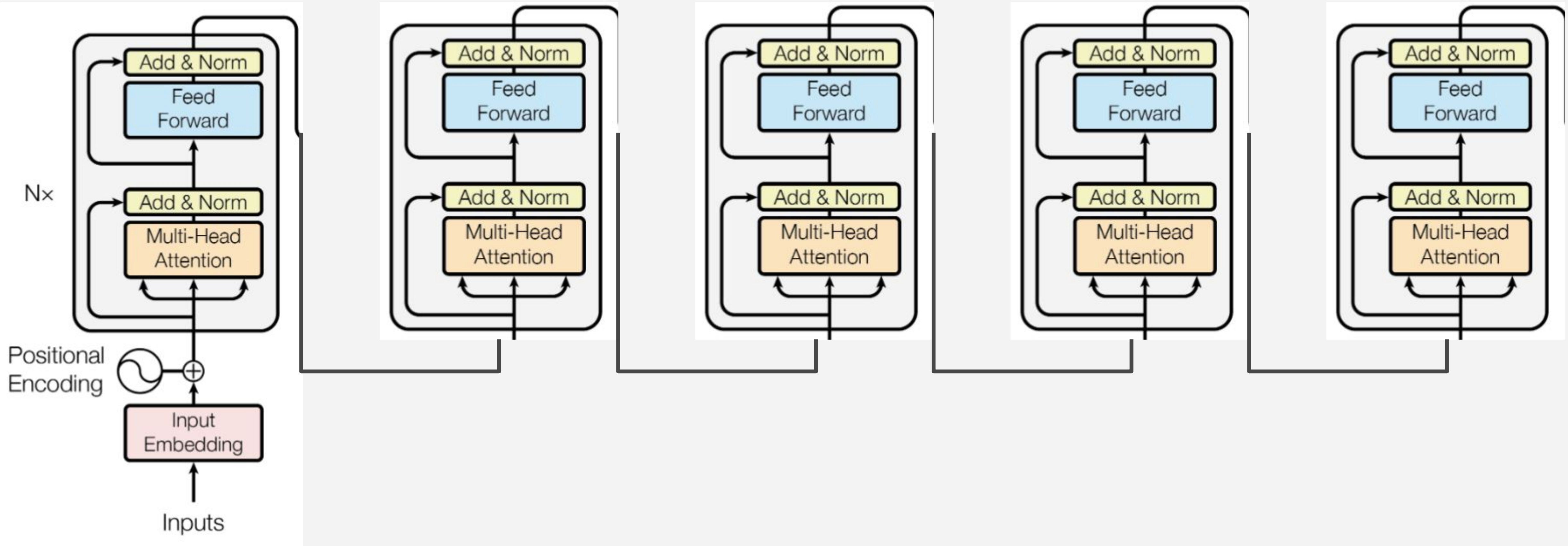
- Two-Layer DNN

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

  - Input x: attention-applied input matrix
    - *ex) "I am a student" + Attention*
  - ReLU activation
    - Adds non-linearity at the end of encoder / decoder
  - Fully connected layers
    - *ex) 512 -> 2024 -> 512*



Encoder/Decoder Stack



Experiment

Result

Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [15]	23.75			
Deep-Att + PosUnk [32]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [31]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [8]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [26]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [32]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [31]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [8]	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**	**41.0**	$2.3 \cdot 10^{19}$	

Experiment

Result

	N	d_{model}	d_{ff}	h	d_k	d_v	P_{drop}	ϵ_{ls}	train steps	PPL (dev)	BLEU (dev)	params $\times 10^6$
base	6	512	2048	8	64	64	0.1	0.1	100K	4.92	25.8	65
(A)				1	512	512				5.29	24.9	
				4	128	128				5.00	25.5	
				16	32	32				4.91	25.8	
				32	16	16				5.01	25.4	
(B)					16					5.16	25.1	58
					32					5.01	25.4	60
(C)	2									6.11	23.7	36
	4									5.19	25.3	50
	8									4.88	25.5	80
		256			32	32				5.75	24.5	28
		1024			128	128				4.66	26.0	168
			1024							5.12	25.4	53
			4096							4.75	26.2	90
(D)							0.0			5.77	24.6	
							0.2			4.95	25.5	
								0.0		4.67	25.3	
								0.2		5.47	25.7	
(E)	positional embedding instead of sinusoids									4.92	25.7	
big	6	1024	4096	16			0.3		300K	**4.33**	**26.4**	213