Attention is All You Need

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones,
Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin (NIPS 2017)

단국대학교 모바일시스템공학과 양윤성

Contents

1. Introduction
2. The Model
3. Experiments
4. Conclusion

1-(1). Background

- Attention : Solving the problem of seq2seq
- But still use RNN or LSTM
- Unable to parallelize
- Inefficient of memory and speed

1-(2). Main Idea

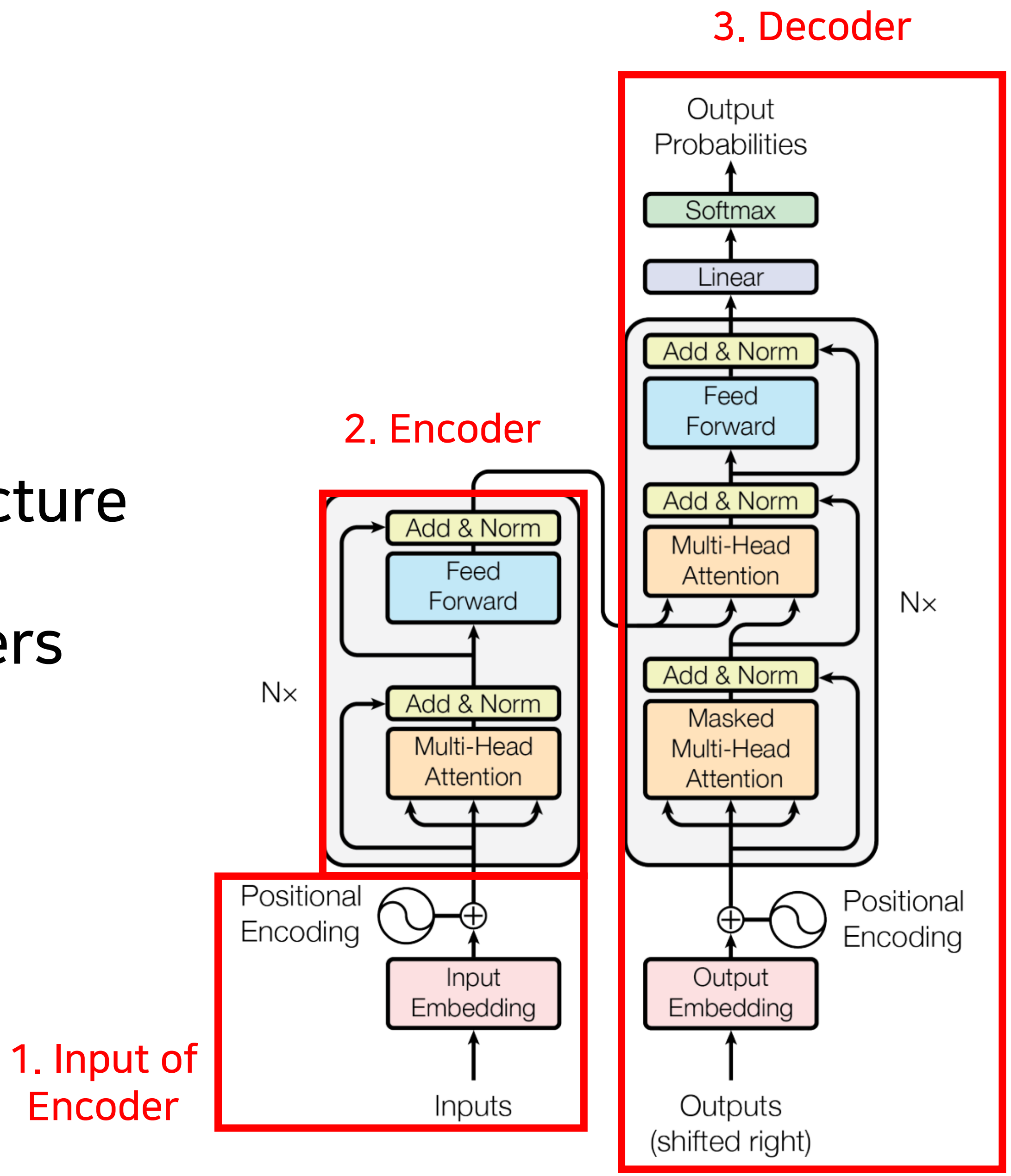
- Remove recursive properties
- Only use attention mechanism
- Parallel processing at once
  - ✓ Much better performance (Max. BLEU : 41.8)
  - ✓ Can also be used for sequence analysis

Contents

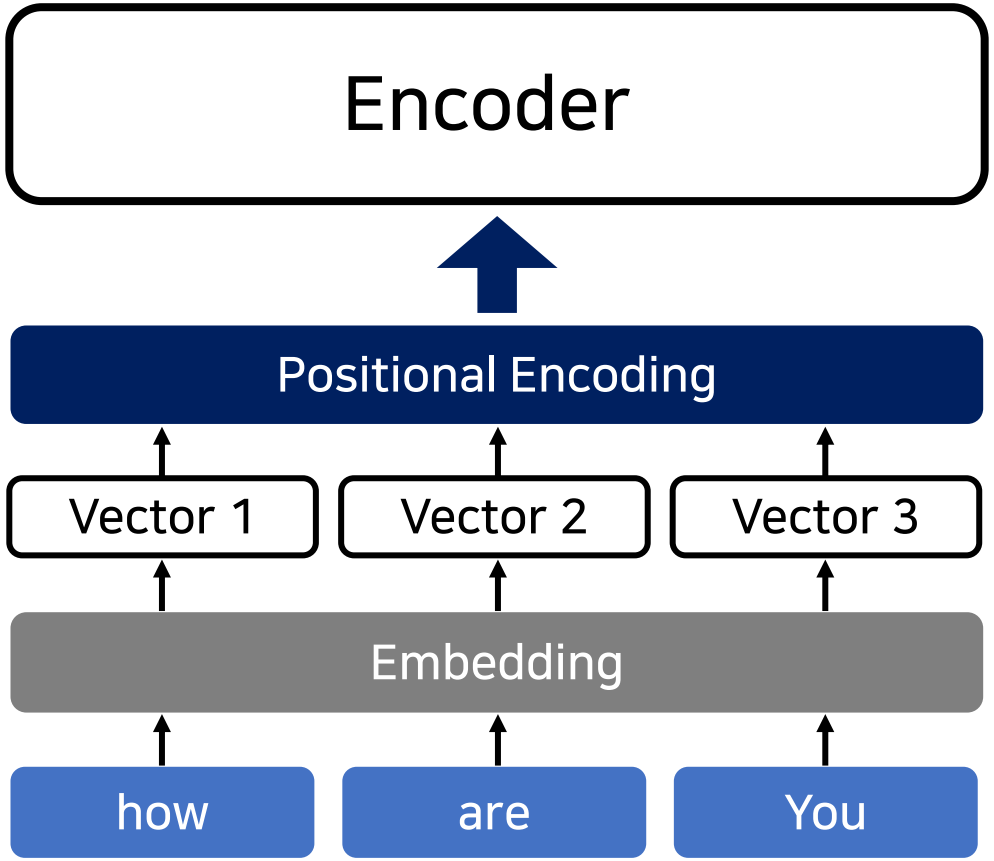
1. Introduction
2. The Model
3. Experiments
4. Conclusion

2-(1). Transformer

- No RNN and LSTM
- Still using encoder-decoder structure
- Repeat attention on multiple layers
- Base technology of BERT, GPT



2-(2). Input of Encoder

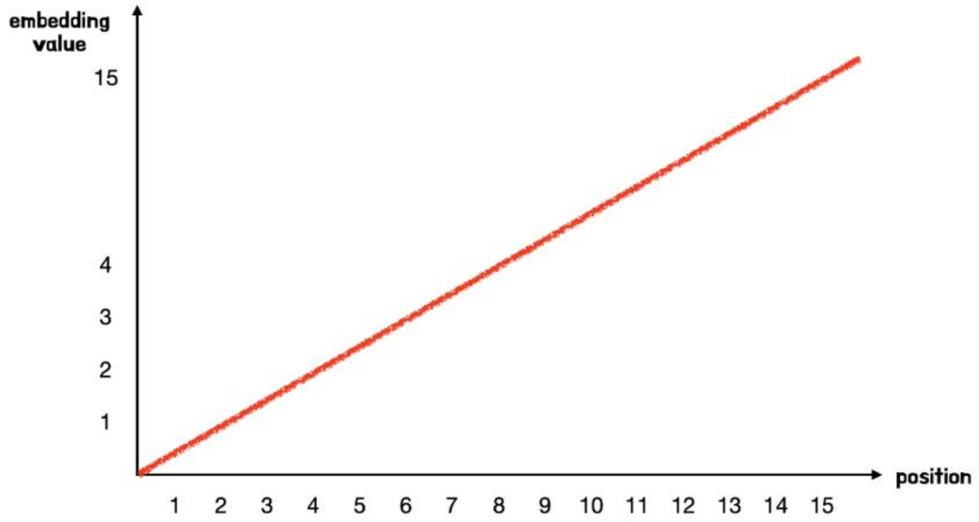


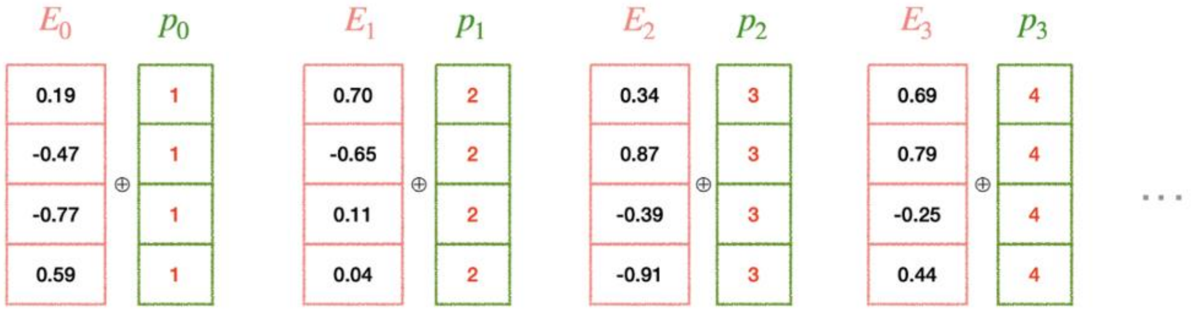
- Embedding by each word first
- No order → No location info
- Using positional encoding

2-(3). Positional Encoding

- To include location information
- How can we get a position vector?

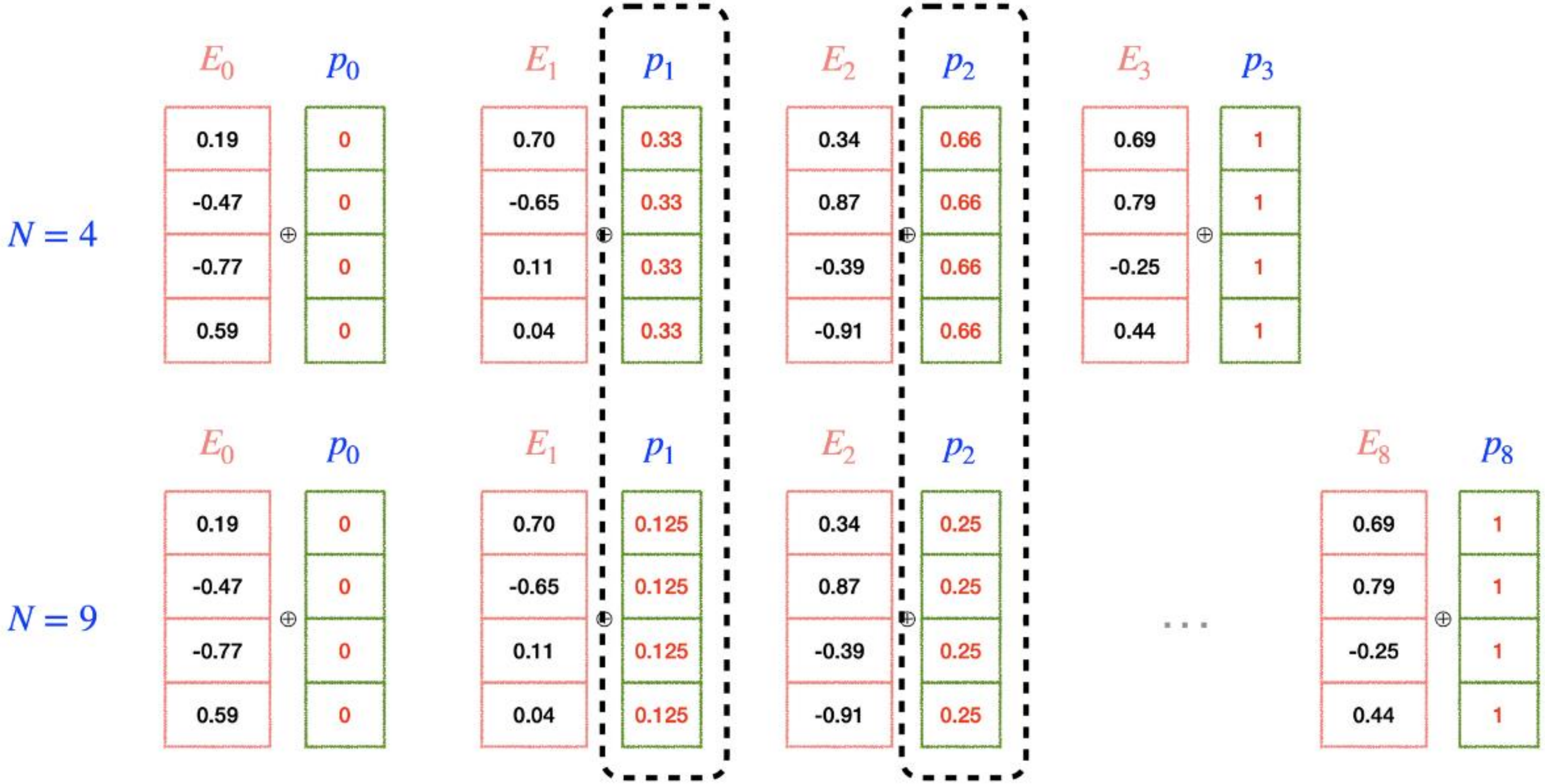
1. Impose an integer values in ascending order





2-(3). Positional Encoding

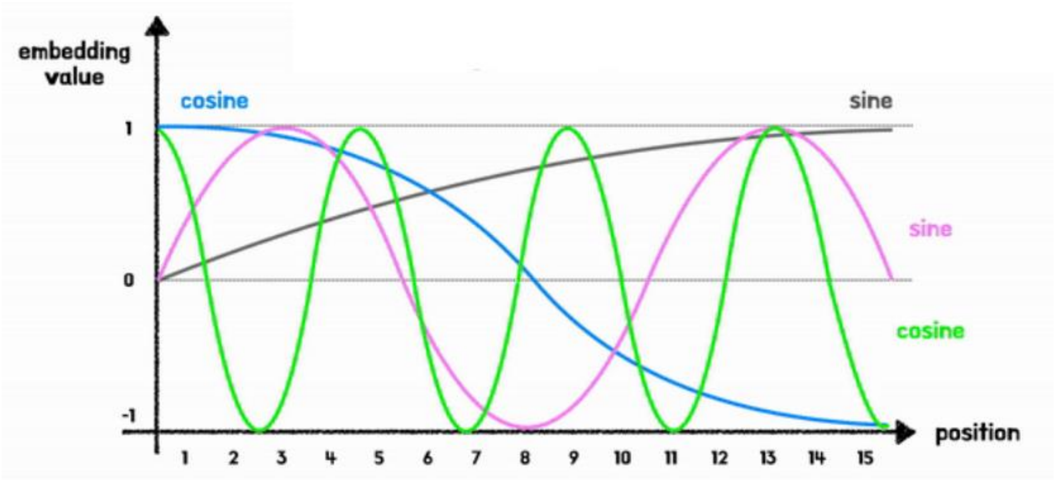
2. Using normalization



2-(3). Positional Encoding

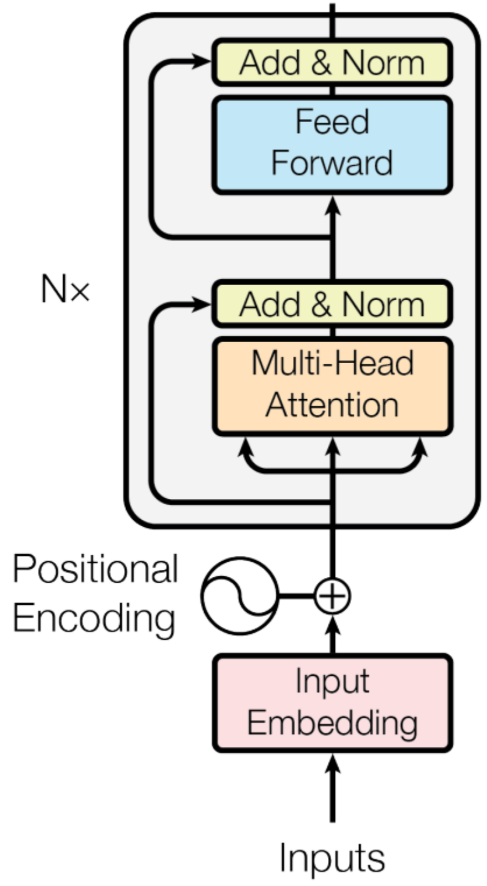
- Best method : Using periodic function (sin, cos)
  - pos : Index of each word
  - i : Order(Index) in the embedding vector
  - d_{model} : Embedding dimension

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



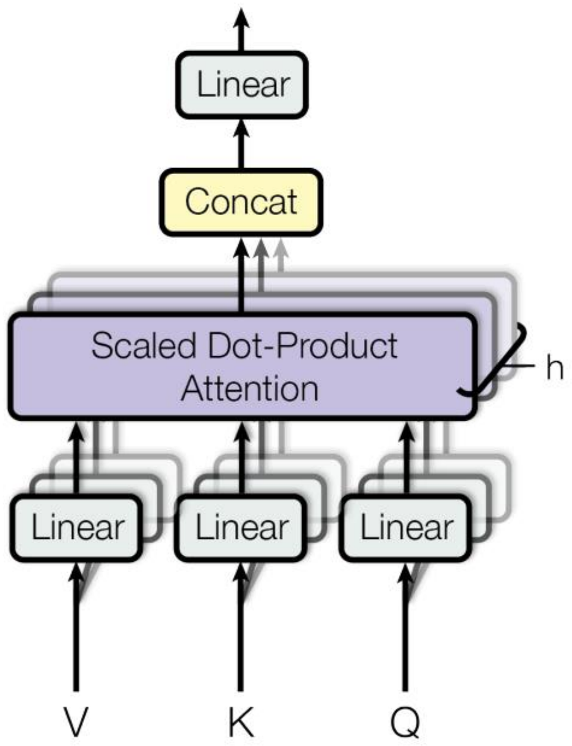
2-(4). Encoder

- Multiple multi-head attention
- Use residual learning and feed forward
- Repeat N times (In the paper, N = 6)
- Use encoder's last output in decoder

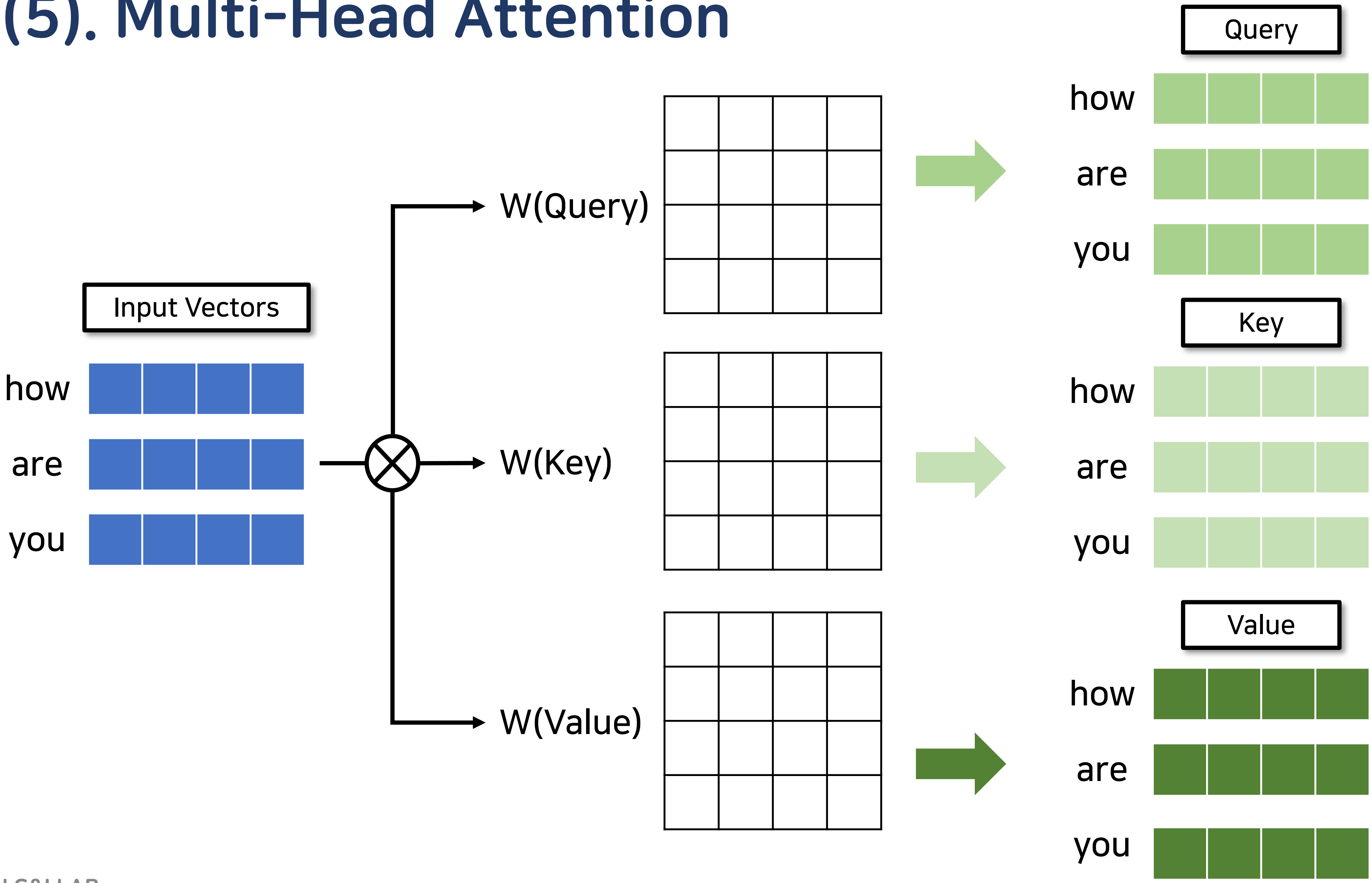


2-(5). Multi-Head Attention

- Use Query, Key, Value
- In encoder, self-attention is used
  - Q, K, V are same
  - Sentence performs attention on its and learns representation
  - Identify each word’s connection to each other



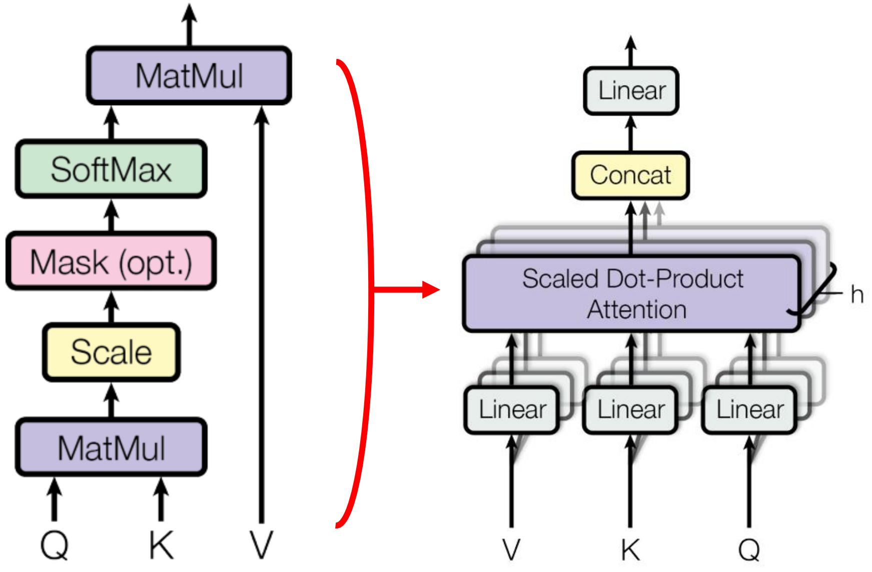
2-(5). Multi-Head Attention



2-(5). Multi-Head Attention

- There are h heads, different V, K, Q for each head
- For linear transformation, multiply weight matrix corresponding V, K, Q
- Concat results and convert it linearly → Final output

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



2-(6). Scaled Dot-Product Attention

1. **Multiply query and key's inverse matrix**
2. **Divide scale factor (scaling)**

→ **Attention energies**

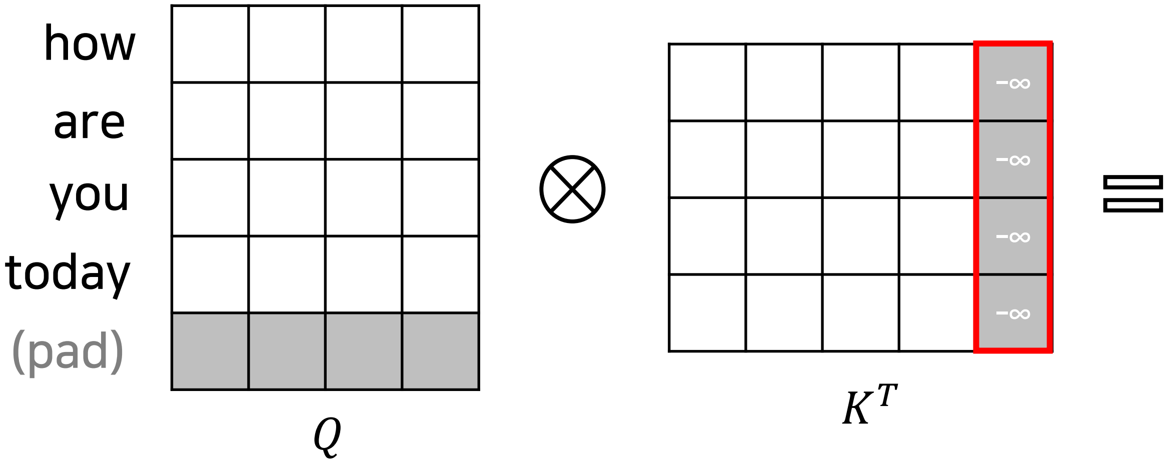
$$\frac{QK^T}{\sqrt{d_k}}$$

- Obtain energy of the key for each query
- Express energy as probability → Determine which key has high weight
  - d_k : Key's dimension

2-(6). Scaled Dot-Product Attention

3. Masking (optional)

- Less tokens than embedding dimension → padding
- Padding part should not be calculated as attention
- Set padding value to converge to 0 for calculate attention score (like -∞)



	how	are	you	today	(pad)
how	0.88	0.15	0.03	0.01	0
are	0.13	0.91	0.08	0.05	0
you	0.05	0.07	0.85	0.11	0
today	0.02	0.06	0.12	0.92	0
(pad)	0	0	0	0	0

Attention Energies

2-(6). Scaled Dot-Product Attention

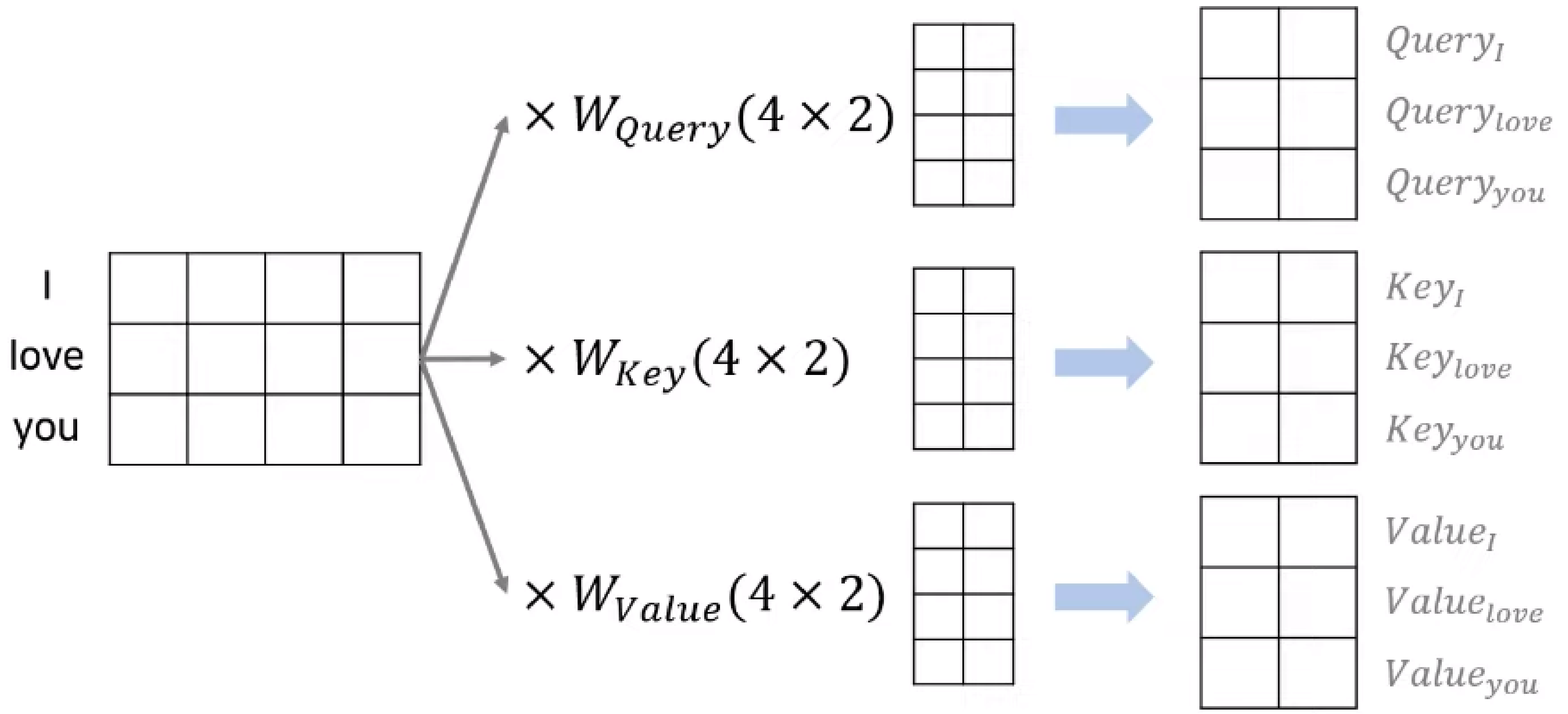
4. Calculate attention score (probability)

5. Multiply attention score and value

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

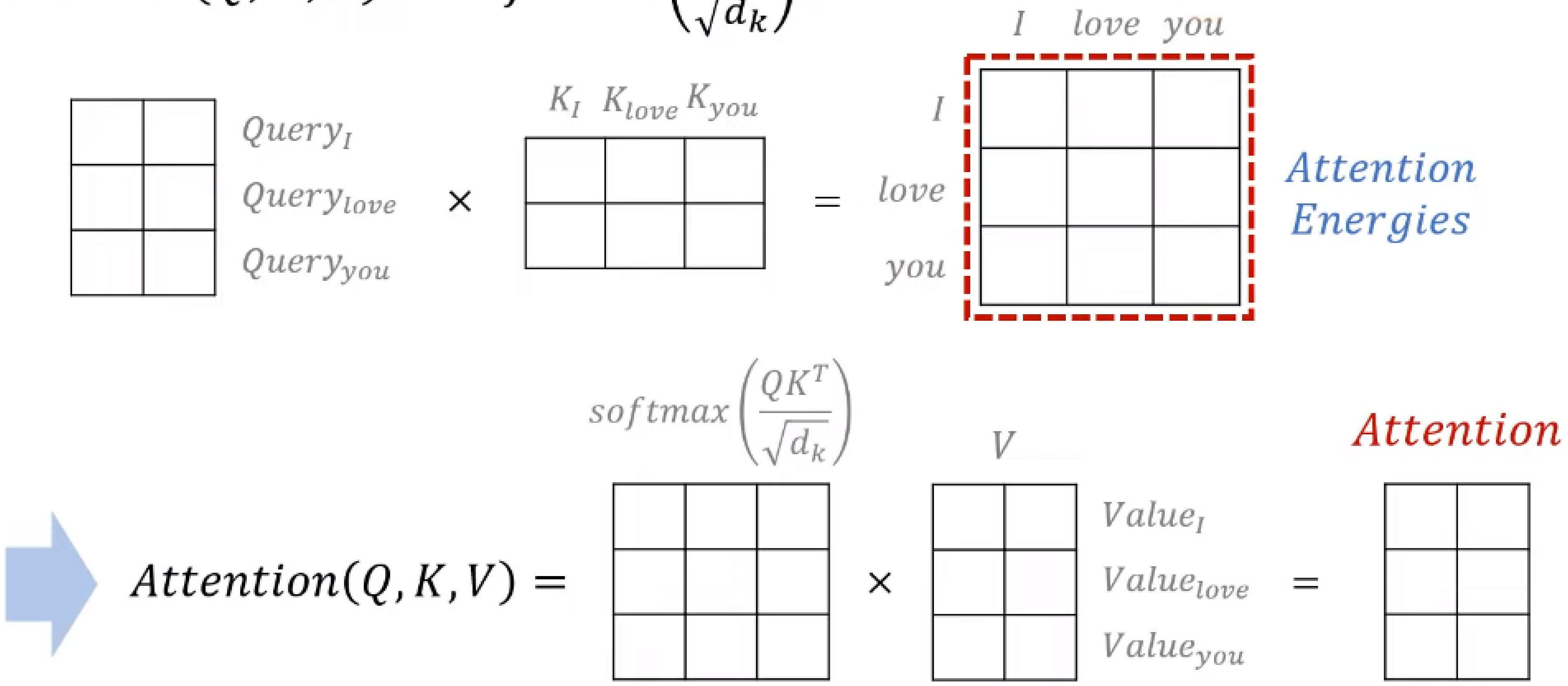
- Example
  - Embbeding dimension : 4
  - Head(h) : 2

2-(6). Scaled Dot-Product Attention

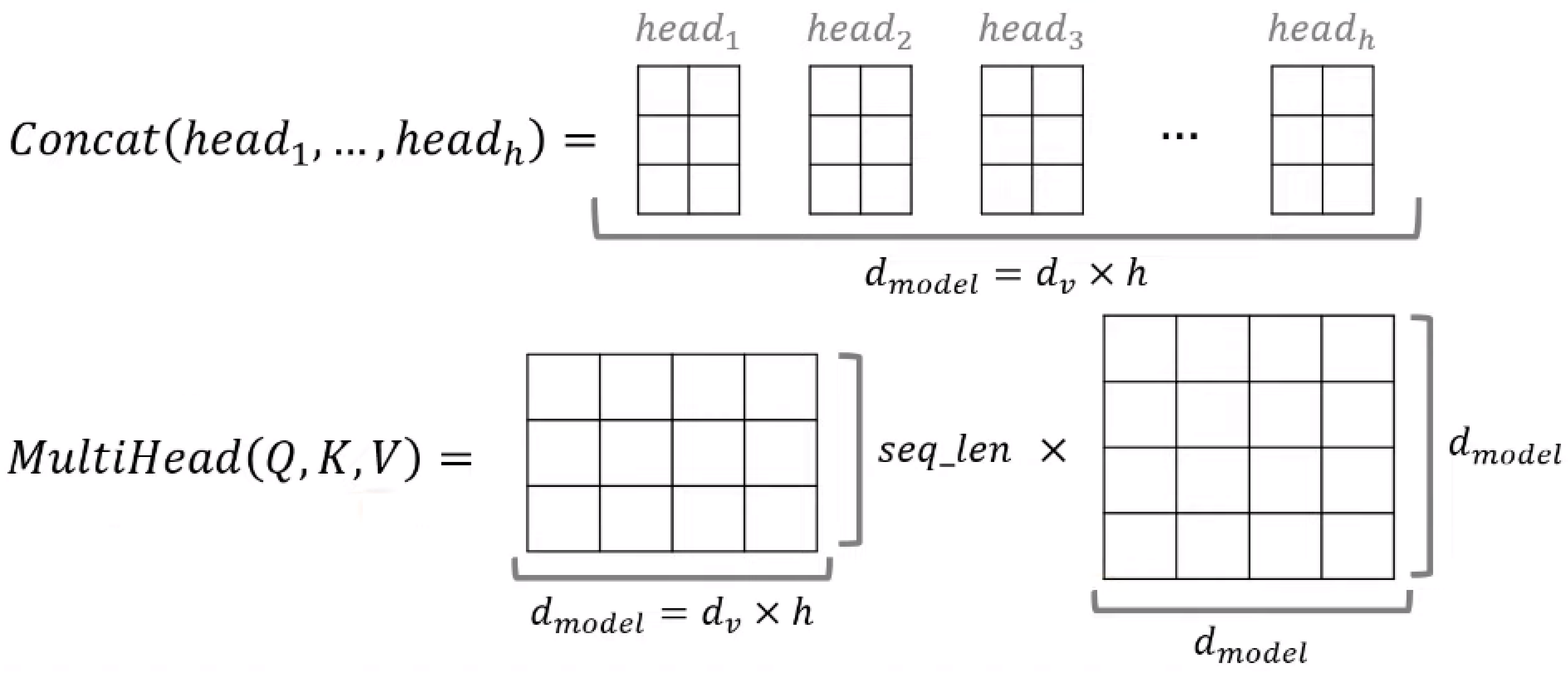


2-(6). Scaled Dot-Product Attention

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

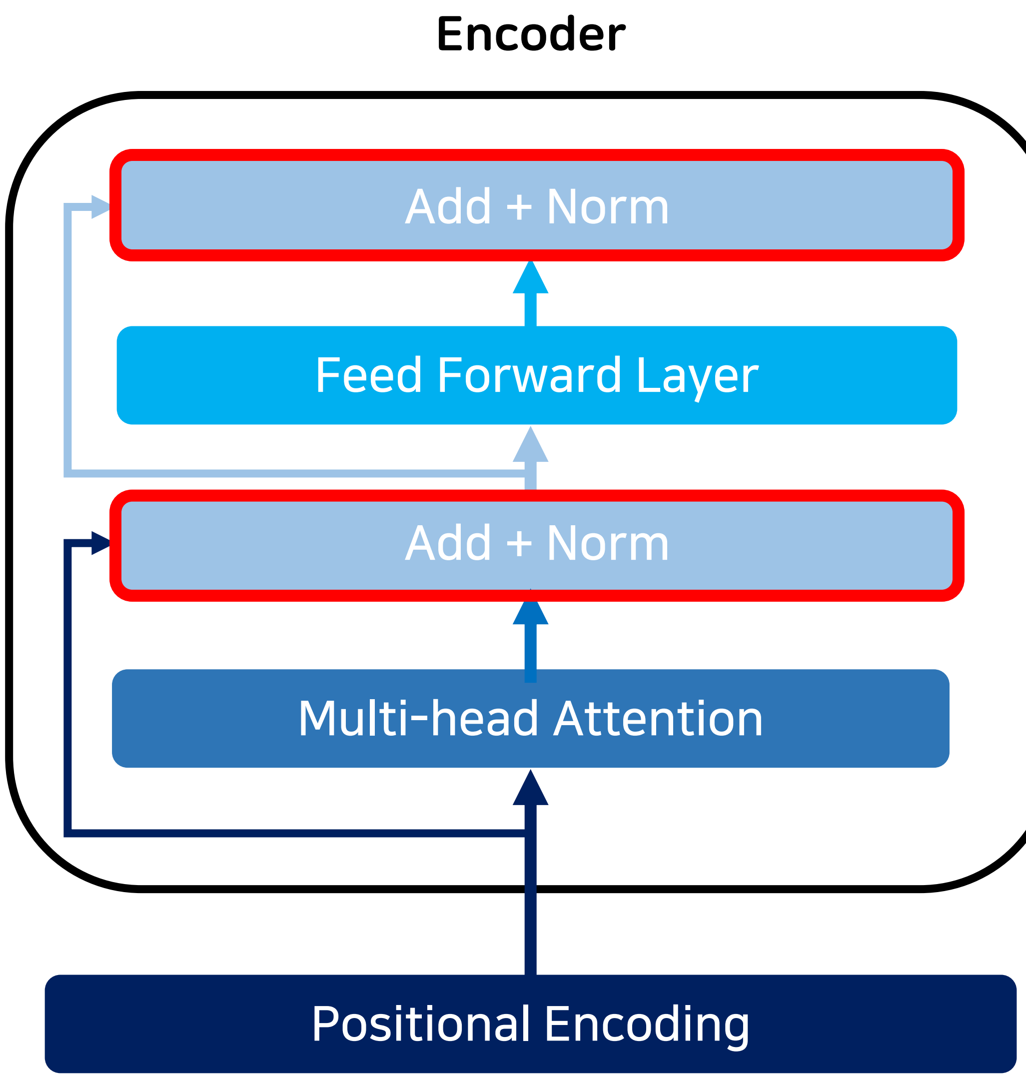


2-(6). Scaled Dot-Product Attention



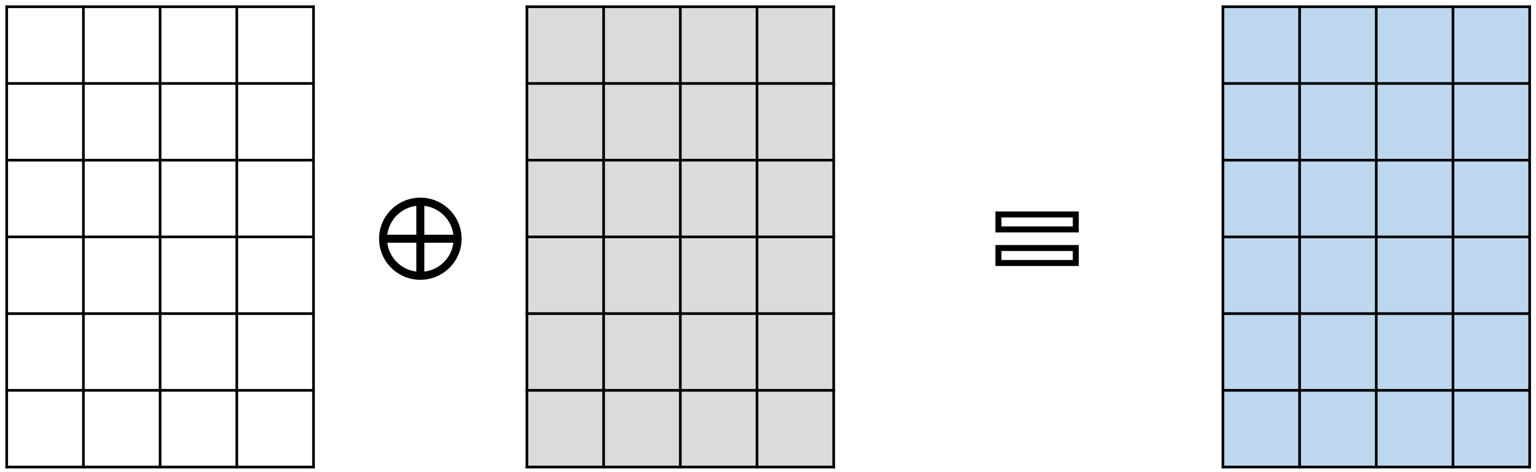
2-(7). Residual Connection

- Add the original input 1 more time
- Give original information explicitly
- To find better optimal solution



2-(7). Residual Connection

- Maintain base properties + only learn the remaining parts
  - ➢ Less learning difficulty



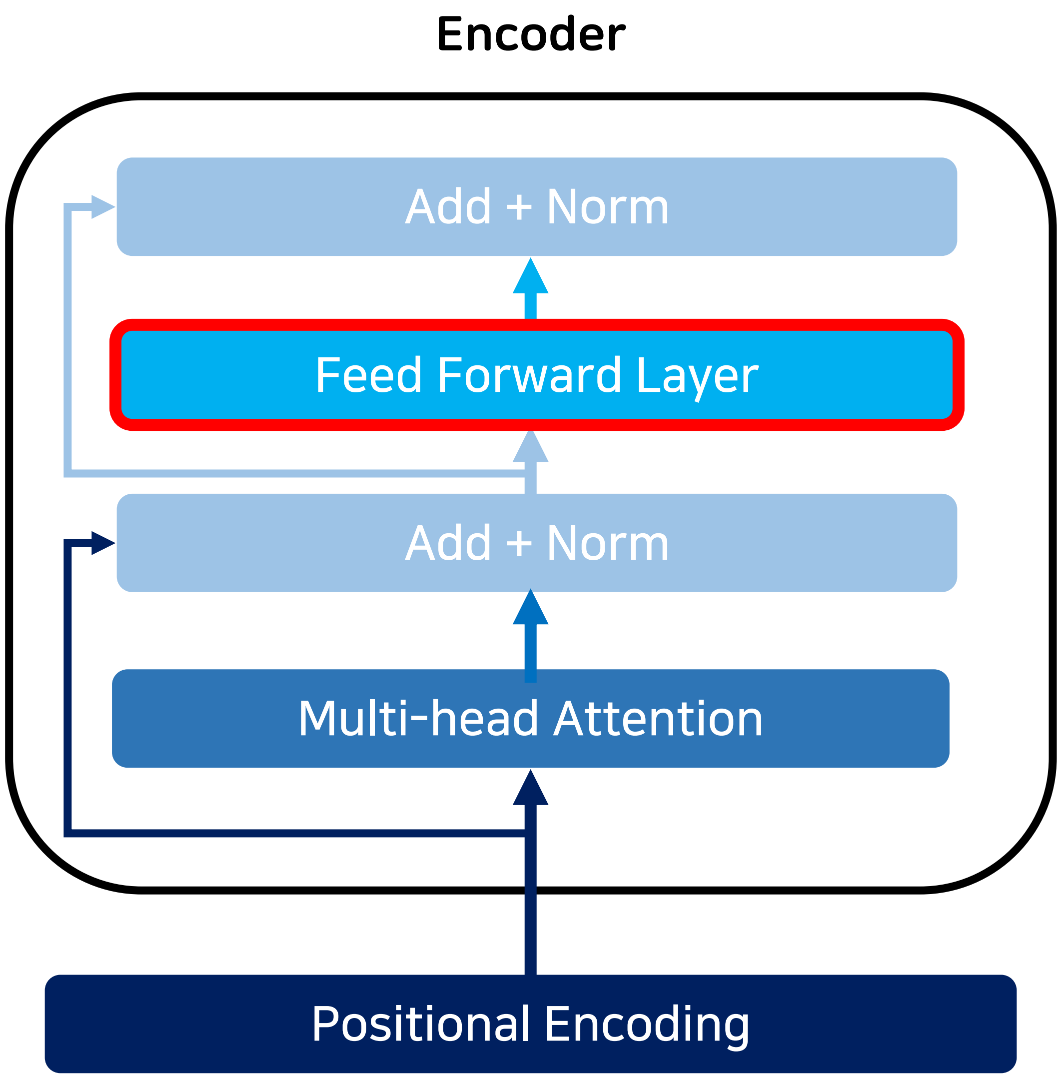
Origin input

After multi-head attention

Final output

2-(8). Feed Forward Network

- Adding non-linearity
- More effective learning
  - ✓ Complex grammatical patterns
  - ✓ Expression of various words
  - ✓ Semantic relationship



2-(8). Feed Forward Network

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

1. Multiply weight matrix(W_1) and add bias(b_1)
2. Use ReLU function
   - Remove negative num → Non-linearity
3. Multiply weight matrix(W_2) and add bias(b_2) → FFN value

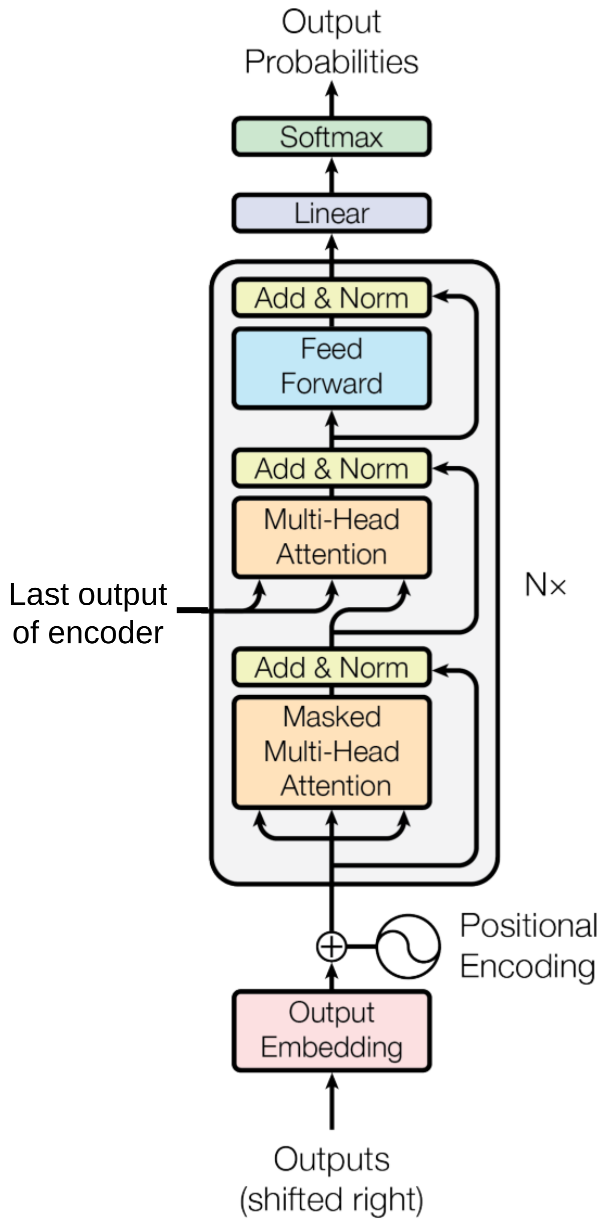
- Linearity → Non-linearity → Linearity

2-(9). Decoder

- Self-attention only first time
- Use encoder's results as key and value afterwards
- Unlike encoder, masking is necessary

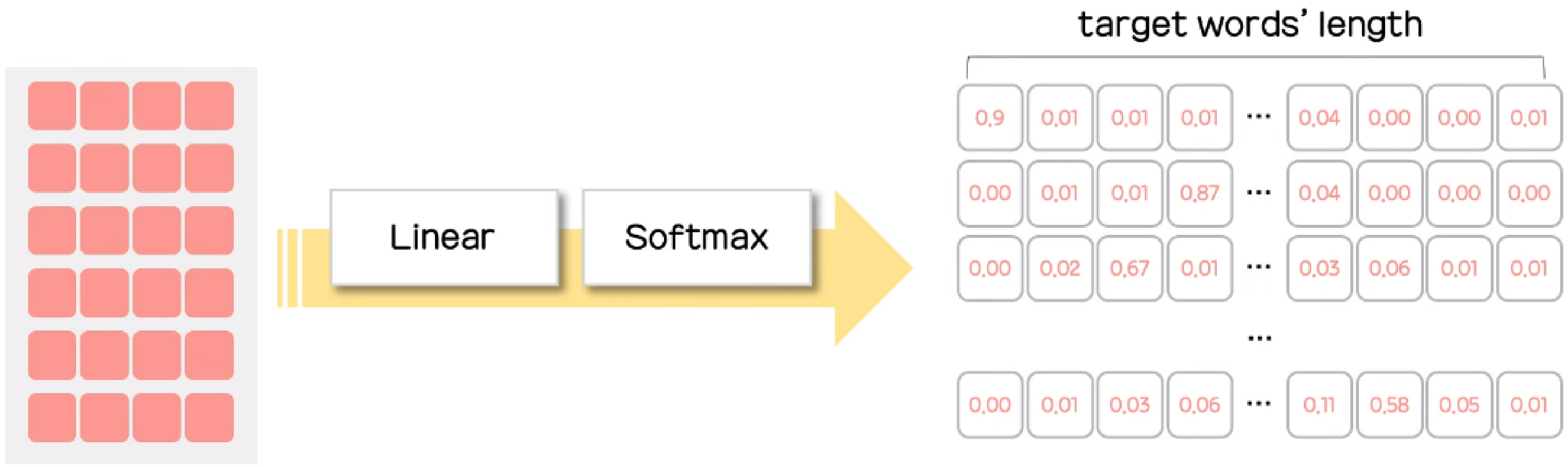
Avoid referencing behind tokens

	오늘	너는	좀	어때	(pad)
오늘		0	0	0	0
너는			0	0	0
좀				0	0
어때					0
(pad)	0	0	0	0	0



2-(9). Decoder

- Similar to encoder
- Last output of decoder → Dense, Softmax (same as seq2seq)
- Select the highest probability each vector and complete translation



Contents

1. Introduction
2. The Model
3. Experiments
4. Conclusion

3-(1). BLEU and Training Cost

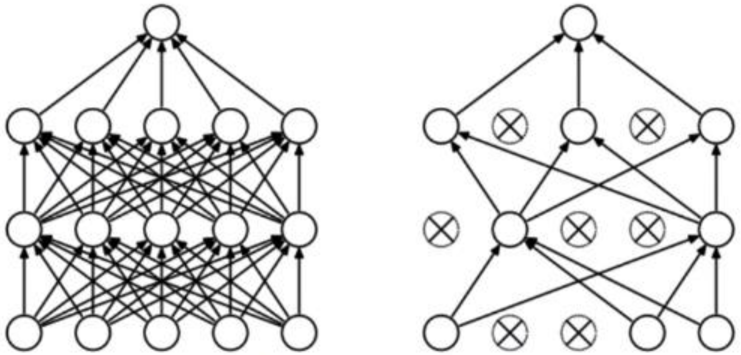
Table 2: The Transformer achieves better BLEU scores than previous state-of-the-art models on the English-to-German and English-to-French newstest2014 tests at a fraction of the training cost.

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**	$2.3 \cdot 10^{19}$	

- Low learning cost, high BLEU score → Much more efficient performance

3-(2). Changing Model Variations

- **Drop-out** : Randomly eliminated nodes → Prevent overfitting



(a) Standard Neural Net (b) After applying dropout.

- **Label-Smoothing** : Softly label of probability distribution → Increase the generalization of the model

$$([1,0,0,0]\times(1-\epsilon)) + (1-[1,0,0,0])\times(\frac{\epsilon}{len(class)-1})$$

$$= [0.9, 0.03, 0.03, 0.03]$$

3-(2). Changing Model Variations

$d_k(d_v) = d_{model}/h$

(A) : Reduce head

(B) : Reduce d_k

(C) : Increase model size

(D) : Drop out, Label Smoothing

(E) : Other positional method

	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

3-(3). Parsing Ability

Parser	Training	WSJ 23 F1
Vinyals & Kaiser el al. (2014) [37]	WSJ only, discriminative	88.3
Petrov et al. (2006) [29]	WSJ only, discriminative	90.4
Zhu et al. (2013) [40]	WSJ only, discriminative	90.4
Dyer et al. (2016) [8]	WSJ only, discriminative	91.7
Transformer (4 layers)	WSJ only, discriminative	91.3
Zhu et al. (2013) [40]	semi-supervised	91.3
Huang & Harper (2009) [14]	semi-supervised	91.3
McClosky et al. (2006) [26]	semi-supervised	92.1
Vinyals & Kaiser el al. (2014) [37]	semi-supervised	92.1
Transformer (4 layers)	semi-supervised	92.7
Luong et al. (2015) [23]	multi-task	93.0
Dyer et al. (2016) [8]	generative	93.3

- Test English constituency parsing, using WSJ dataset
- Performance as good as existing parser

Contents

1. Introduction
2. The Model
3. Experiments
4. Conclusion

4. Conclusion

- Use attention only, remove recursive property
  - Parallel processing
  - Significantly faster than existing architecture
- Transformer can be applied to many task
- Goal : Apply to audio, video, and image as well as text

Thank You!

Attention is All You Need