Transformer

Attention is all you need

서울대학교 IDEA 연구실

박사과정 신윤섭

Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com

Noam Shazeer* Google Brain noam@google.com

Niki Parmar* Google Research nikip@google.com

Jakob Uszkoreit* Google Research usz@google.com

Llion Jones* Google Research llion@google.com

Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu

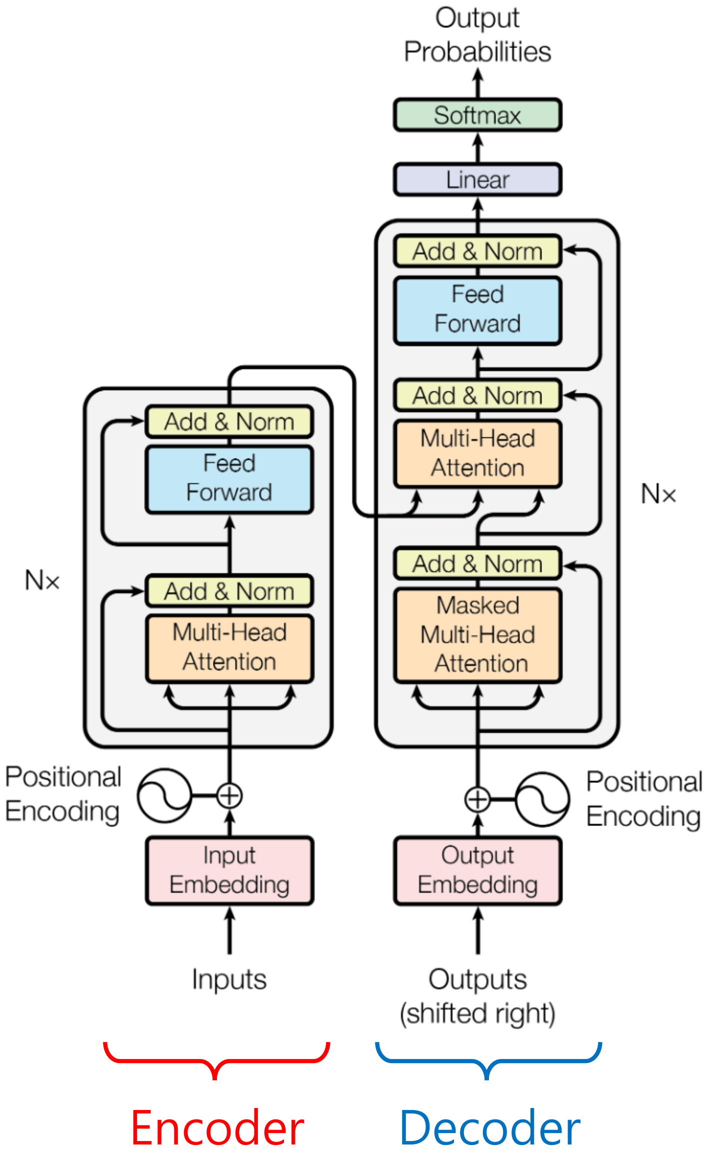
Łukasz Kaiser* Google Brain lukaszkaiser@google.com

Illia Polosukhin* ‡ illia.polosukhin@gmail.com

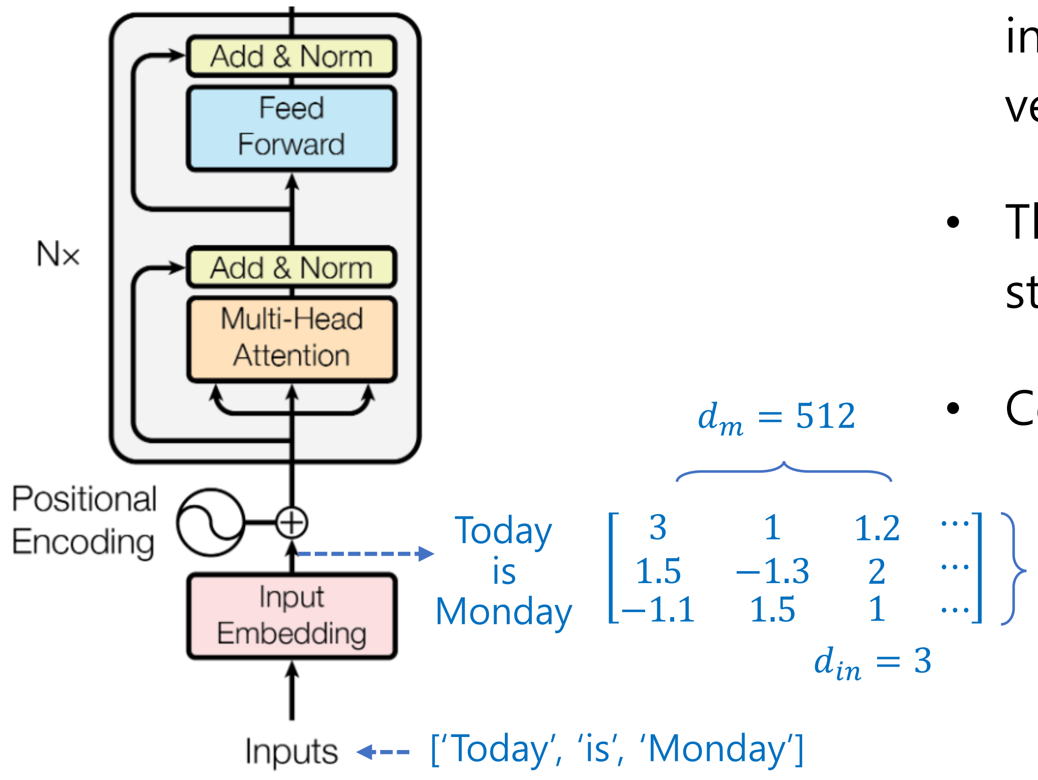
OUTLINE

01. Transformer

- The Transformer algorithm was first proposed in a paper published in 2017 by researchers from Google Brain.
- To overcome the limitations of RNN used in traditional sequence-to-sequence models, the Transformer completely avoids RNN structures and instead utilizes Attention mechanisms.
  (This is why the paper is titled "Attention is all you need.")
- The Transformer consists of an Encoder and a Decoder, each made up of multiple sub-layers.
- In this review, we will focus on the overall structure of the Transformer.

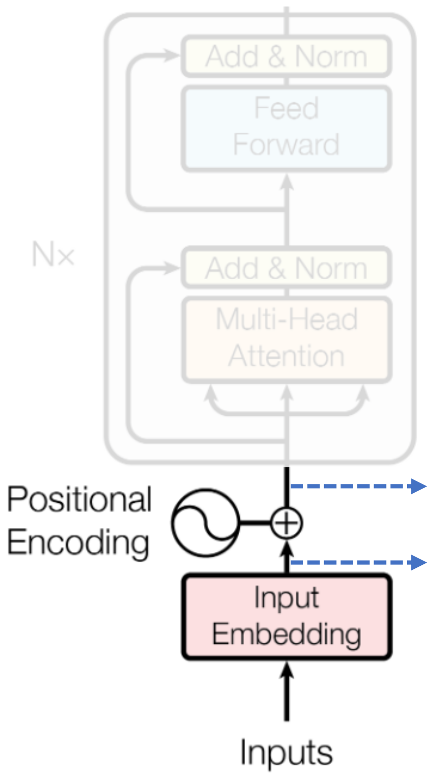


02. Encoder



- The encoder of the Transformer model serves to convert an input sequence (e.g., a sentence) into a high-dimensional vector.
- The Encoder part uses multiple identical Encoder blocks stacked together (the paper uses N=6).
- Components:
  - Positional Encoding
  - Multi-Head Attention
  - Feed Forward
  - Add & Norm

02. Encoder - Positional Encoding



$\boldsymbol{I_E} : d_{in} \times d_m$

$$\begin{bmatrix} 3 & 1 & 1.2 & \cdots \\ 1.5 & -1.3 & 2 & \cdots \\ -1.1 & 1.5 & 1 & \cdots \end{bmatrix}$$

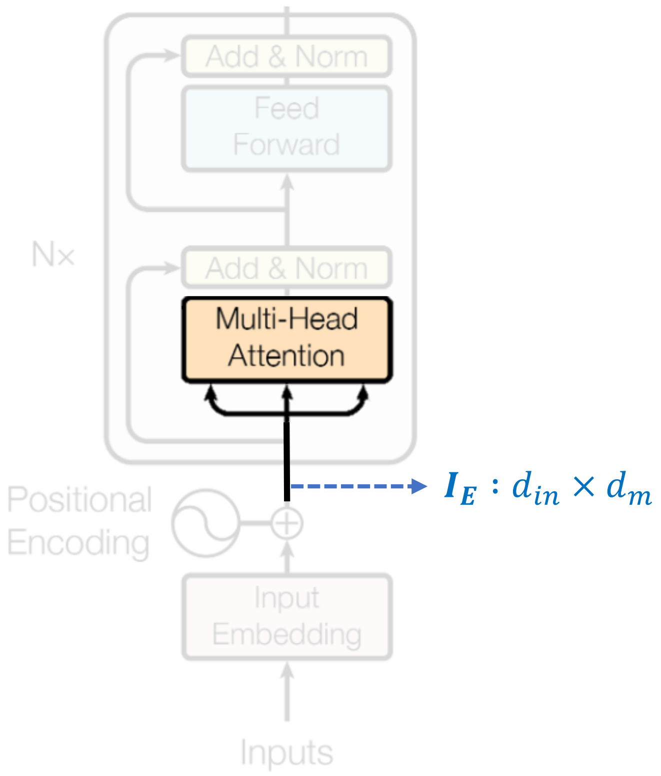
- Positional Encoding adds positional information to input matrix that lack order information.
- In this work, they use sine and cosine functions of different frequencies:

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

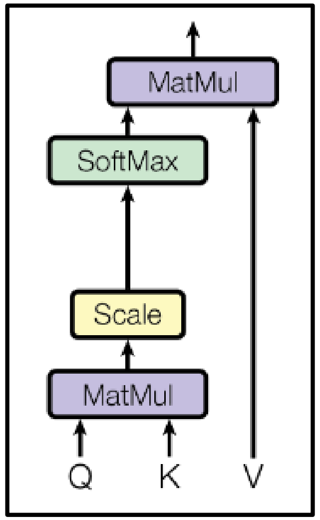
- Ex. Adding Positional Encoding to the sequence ['Today', 'is', 'Monday']

$$I_E = \begin{bmatrix} 3 & 1 & 1.2 & \cdots \\ 1.5 & -1.3 & 2 & \cdots \\ -1.1 & 1.5 & 1 & \cdots \end{bmatrix} + \begin{bmatrix} cos\left(\frac{1}{10^{4*(0/d_m)}}\right) & \sin\left(\frac{1}{10^{4*(2*1/d_m)}}\right) & cos\left(\frac{1}{10^{4*(2*2/d_m)}}\right) & \cdots \\ cos\left(\frac{2}{10^{4*(0/d_m)}}\right) & \sin\left(\frac{2}{10^{4*(2*1/d_m)}}\right) & cos\left(\frac{2}{10^{4*(2*2/d_m)}}\right) & \cdots \\ cos\left(\frac{3}{10^{4*(0/d_m)}}\right) & \sin\left(\frac{3}{10^{4*(2*1/d_m)}}\right) & cos\left(\frac{3}{10^{4*(2*2/d_m)}}\right) & \cdots \end{bmatrix}$$

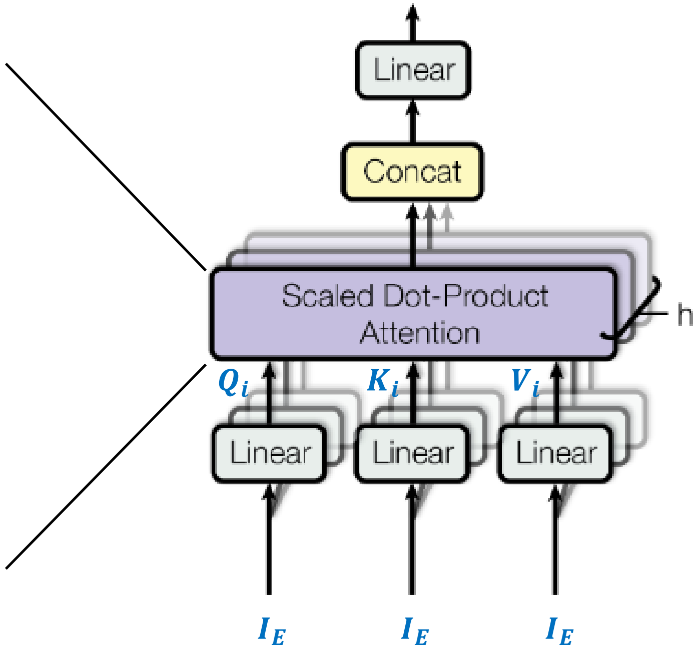
02. Encoder - Multi-head Attention (self-attention)



- Multi-Head Attention in the Encoder consists of h (the paper uses $h = 8$) self-Attention mechanisms.
- It calculates the relationships between words in the input sequence through these mechanisms.

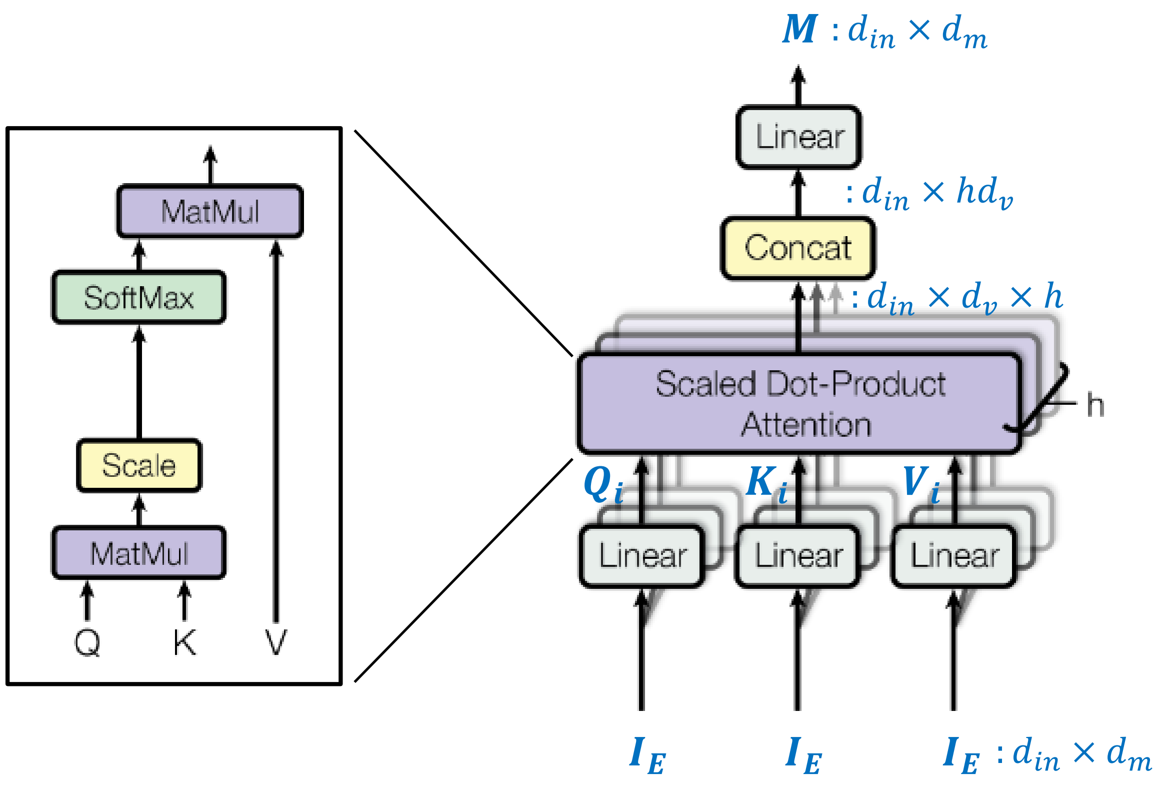


〈Scaled Dot-Product Attention〉



〈Multi-Head Attention〉

02. Encoder - Multi-head Attention (self-attention)



〈Scaled Dot-Product Attention〉 〈Multi-Head Attention〉

- Query, Key, Value are the product of the embedded sentence matrix and each weight matrix.
- Scaled Dot-Product Attention is equivalent to the Dot-Product attention multiplied by a scaling factor of $\frac{1}{\sqrt{d_k}}$ (where $d_k = 64$).
- Multi-Head Attention can be expressed with the following formula.

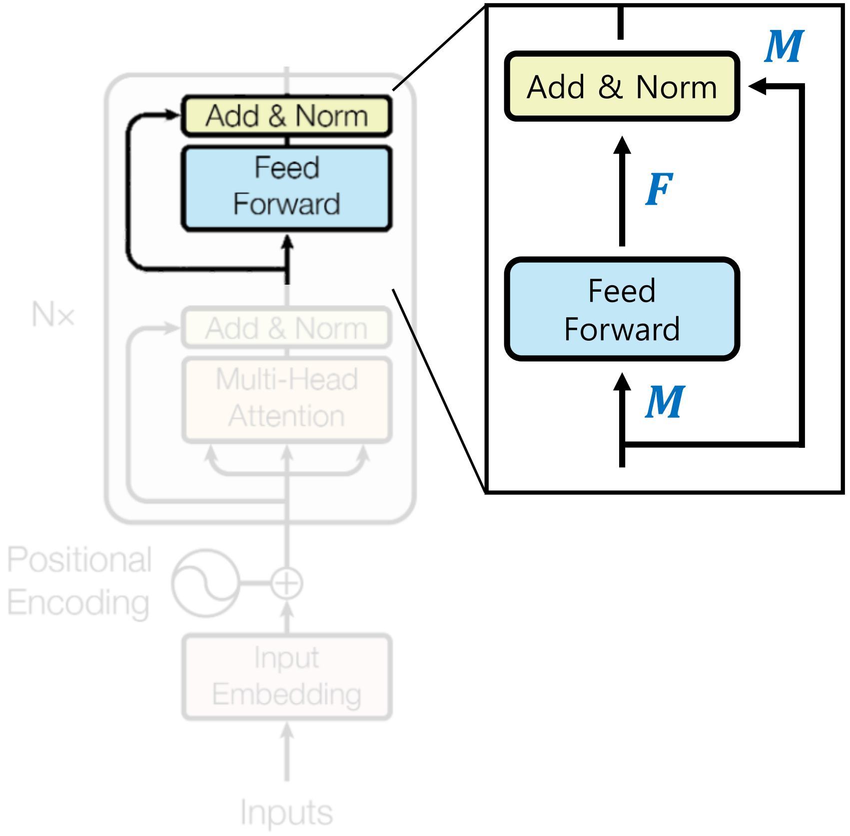
$$M = MultiHead(I_E, I_E, I_E) = Concat(head_1, \cdots, head_h)W^o$$

$$\text{where } head_i = Attention(I_E W_i^Q, I_E W_i^K, I_E W_i^V)$$

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

- Where the projection parameter matrix $W_i^Q, W_i^K \in \mathbb{R}^{d_m \times d_k}, W_i^V \in \mathbb{R}^{d_m \times d_v}, W^o \in \mathbb{R}^{hd_v \times d_m}$

02. Encoder - Feed Forward and Add & Norm layer



- **Feed Forward Neural Network** Feed Forward
  - The position-wise Feed-Forward Networks used in the Transformer are applied independently to each position.
  - This consists of two linear transformation with a ReLu activation in between :

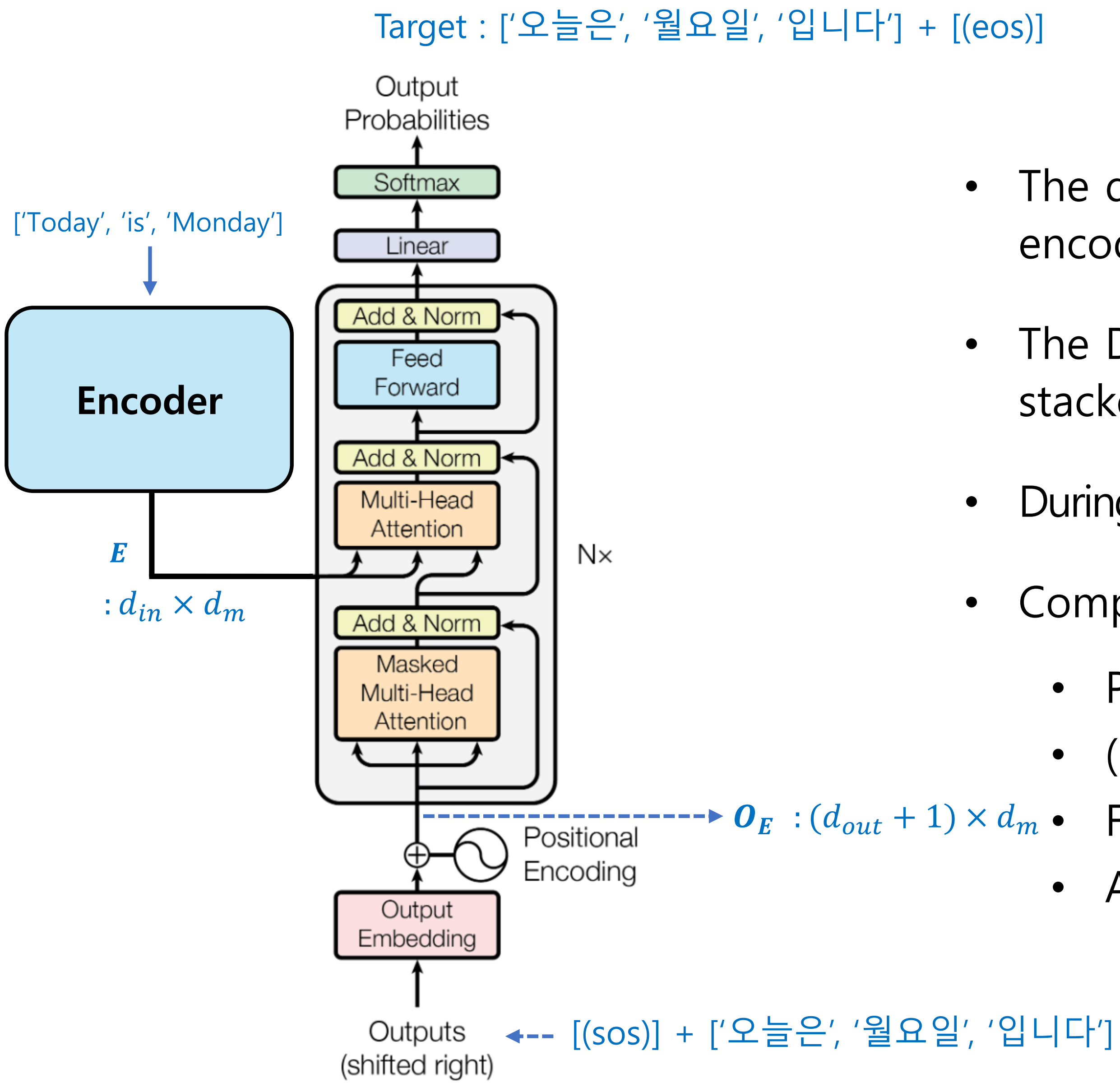
    $$FNN(x) = W_2 ReLu(W_1 x + b_1) + b_2$$

    (Input and output dim : d_m, inner-layer dim : $d_{ff} = 2048$)

- **Add & Norm layer** Add & Norm
  - In the Transformer, each sub-layer's output is not used directly.
  - Instead, the input and output of the sub-layer are added together and then layer normalization is applied.

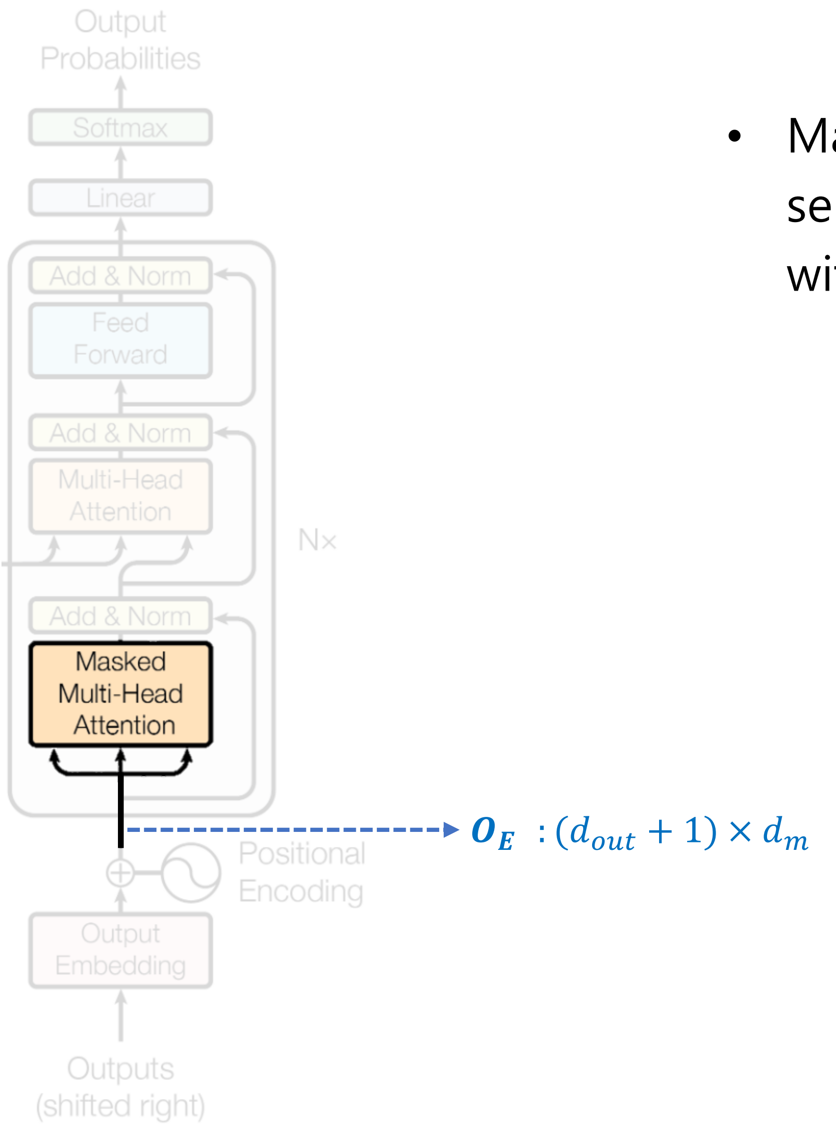
$$Add\&Norm(M, S) = LayerNorm(M + S)$$

03. Decoder

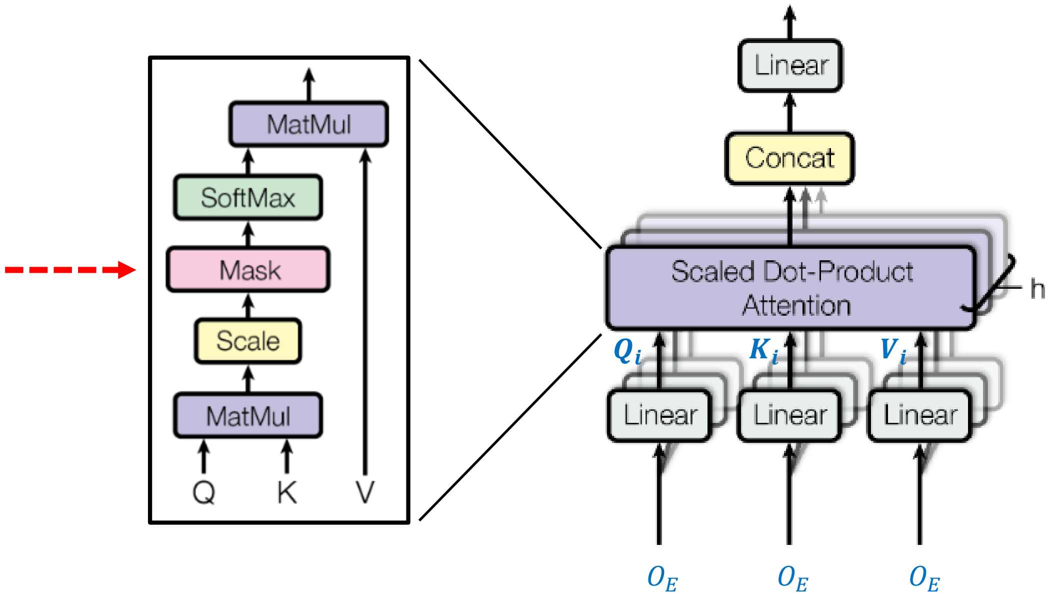


- The decoder in the Transformer receives the output from the encoder and generates the target sequence.
- The Decoder part also uses multiple identical Decoder blocks stacked together (the paper uses N=6).
- During training in the Decoder, the teacher forcing method is used.
- Components:
    - Positional Encoding
    - (Masked) Multi-Head Attention
    - Feed Forward
    - Add & Norm

03. Decoder - Masked Multi-head Attention (self-attention)

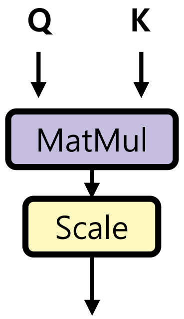


- Masked Multi-Head Attention used in the Decoder employs a self-Attention mechanism similar to that in the Encoder, but with some differences.



〈Scaled Dot-Product Attention〉 〈Masked Multi-Head Attention〉

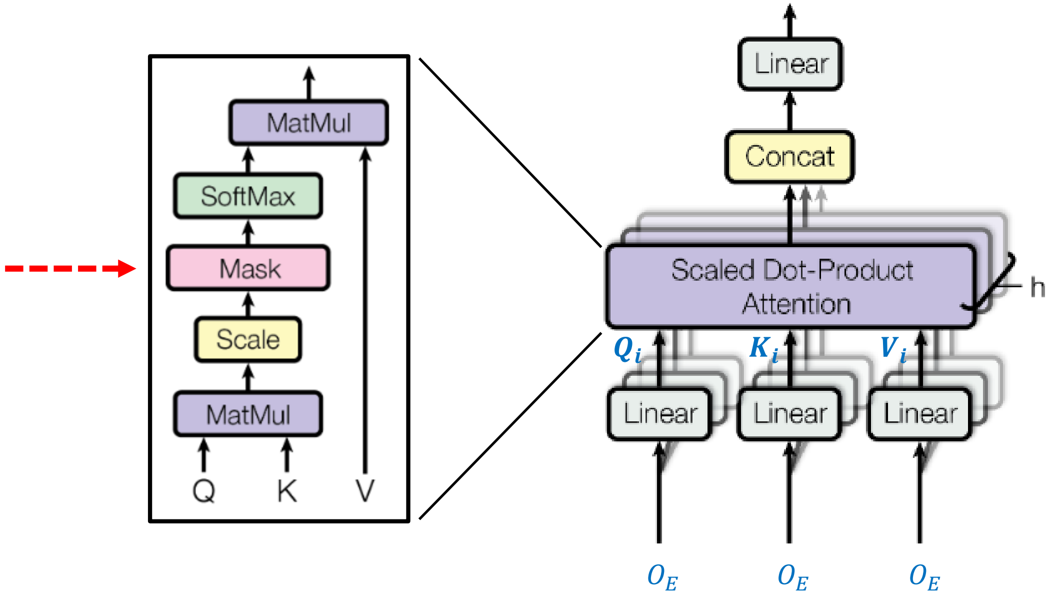
03. Decoder - Masked Multi-head Attention (self-attention)



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

	(sos)	오늘은	월요일	입니다
(sos)	10	2	1	1
오늘은	1	8	1	4
월요일	1	2	9	1
입니다	1	2	1	7

- Since the Decoder operates sequentially, it cannot use information about words that appear after the current word in the attention scores.
- Therefore, the attention scores for words that appear after the reference word are replaced with $-\infty$.



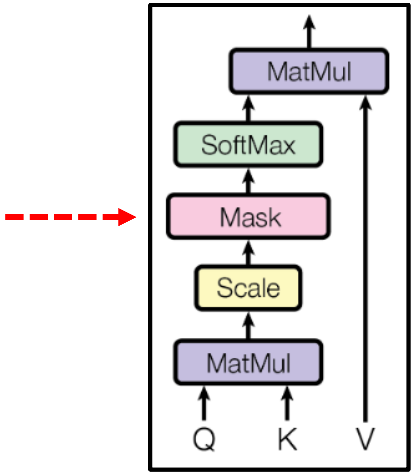
〈Scaled Dot-Product Attention〉 〈Masked Multi-Head Attention〉

03. Decoder - Masked Multi-head Attention (self-attention)

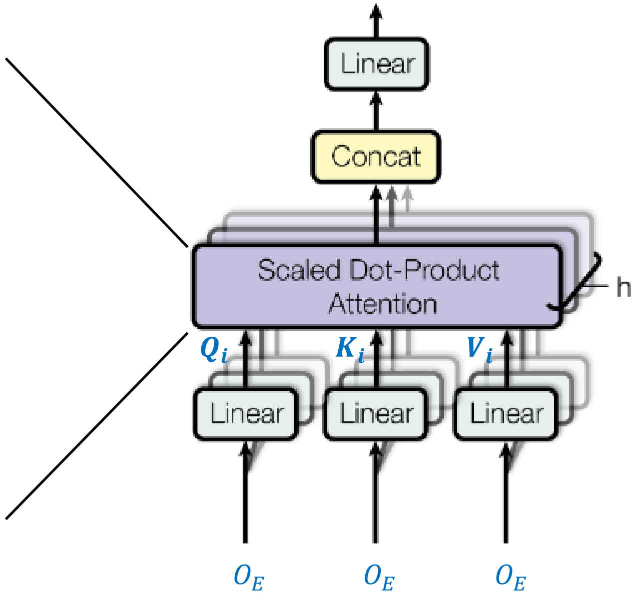
- Since the Decoder operates sequentially, it cannot use information about words that appear after the current word in the attention scores.

- Therefore, the attention scores for words that appear after the reference word are replaced with $-\infty$.

$Mask\left(\frac{QK^T}{\sqrt{d_k}}\right)$	(sos)	오늘은	월요일	입니다
(sos)	10	$-\infty$	$-\infty$	$-\infty$
오늘은	1	8	$-\infty$	$-\infty$
월요일	1	2	9	$-\infty$
입니다	1	2	1	7

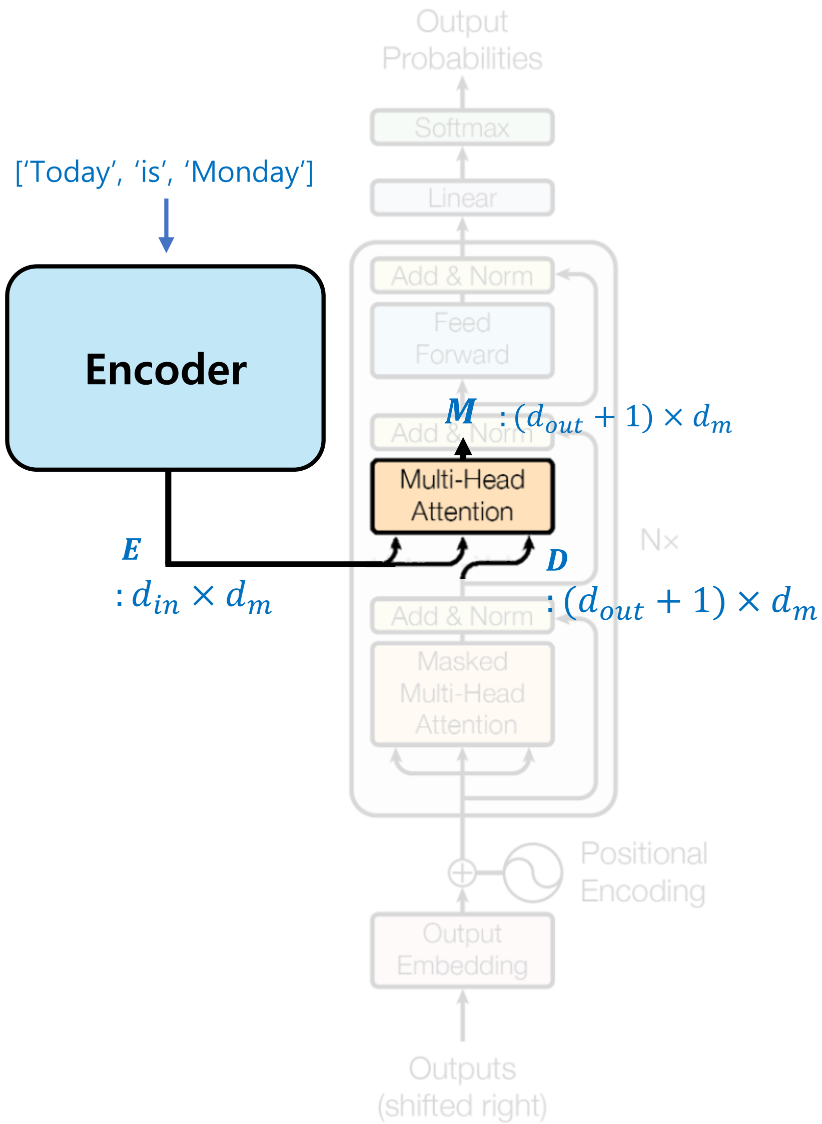


〈Scaled Dot-Product Attention〉

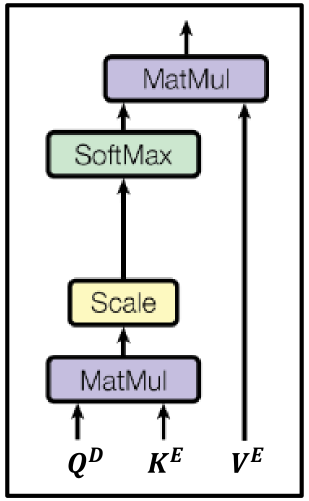


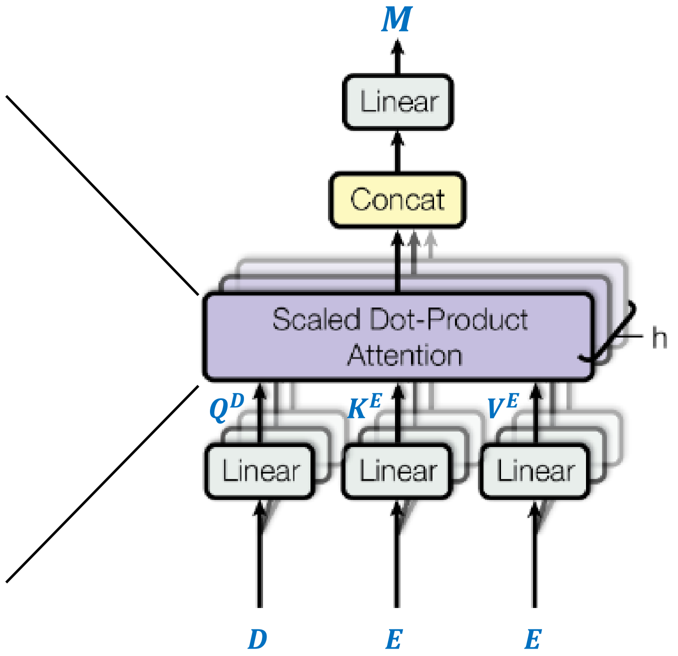
〈Masked Multi-Head Attention〉

03. Decoder - Multi-head Attention(encoder-decoder attention)



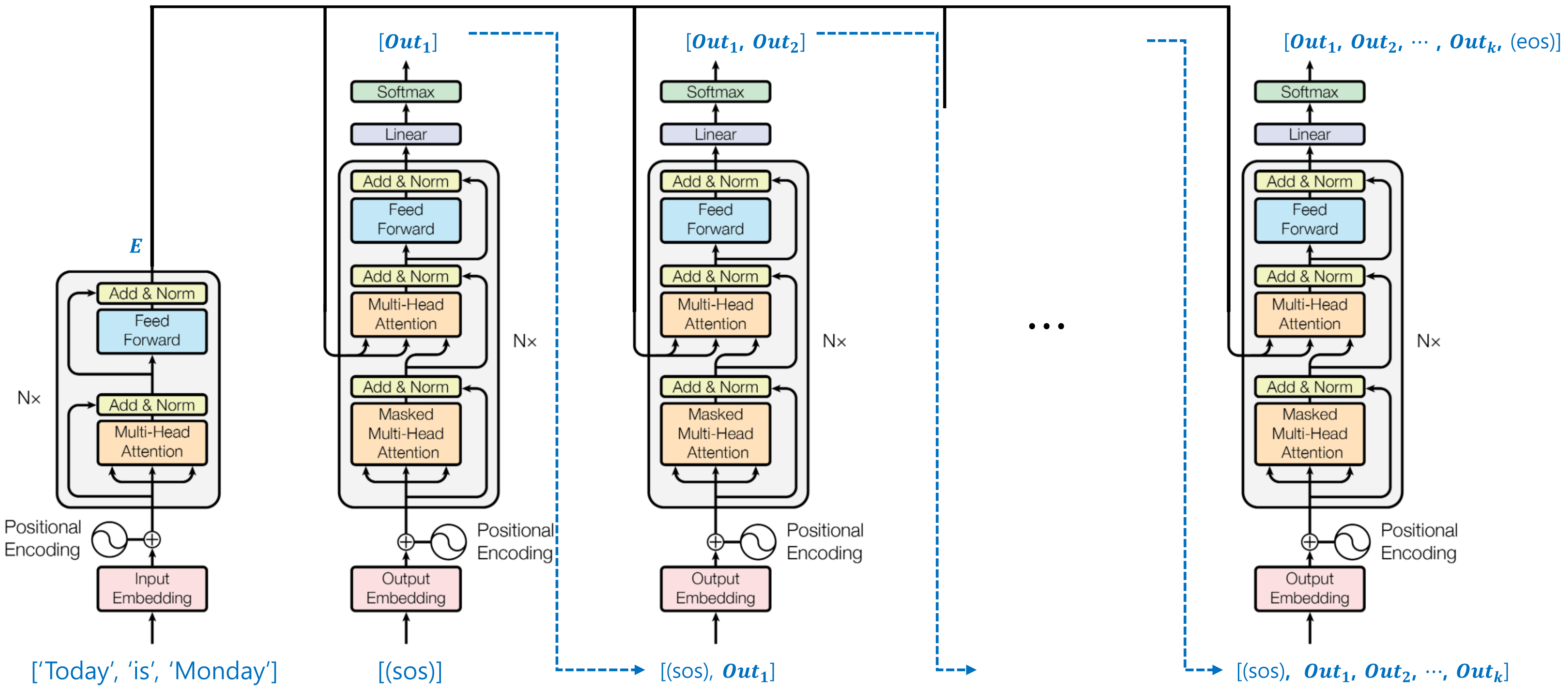
- In the Decoder block, the second Multi-Head Attention is the encoder-decoder attention.
- It uses the output from the encoder to compute the key and value matrices, and computes the query matrix using the output computed within the Decoder block.





〈Scaled Dot-Product Attention〉 〈Encoder-Decoder Multi-Head Attention〉

04. Inference example



05. Experiment

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

End