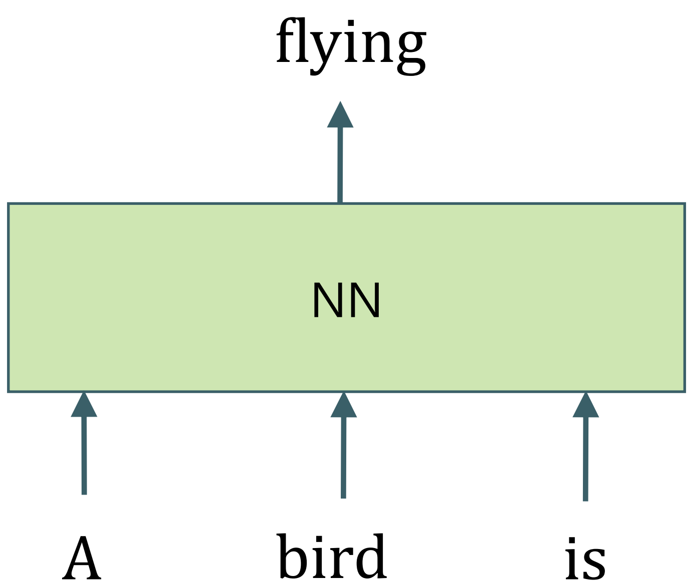
Transformer Model

Word Presentation

- **Next Word Prediction**

A bird is → flying



Training Data

A bird is flying in the sky

(A → bird)

(A bird → is)

(A bird is → flying)

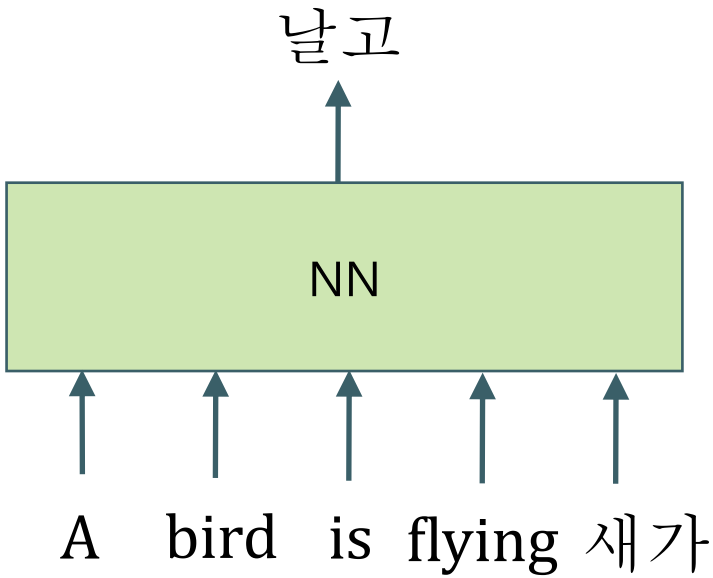
(A bird is flying → in)

…

Word Presentation

➢ **Translator with Next Word Prediction**

A bird is flying → 새가 날고 있다



Training Data

(A bird is flying → 새가)

(A bird is flying 새가 → 날고)

(A bird is flying 새가 날고 → 있다)

…

Word Presentation

➢ How to Handle Texts

(A bird is → flying)

(A dog is → running)

(A fish → swims)

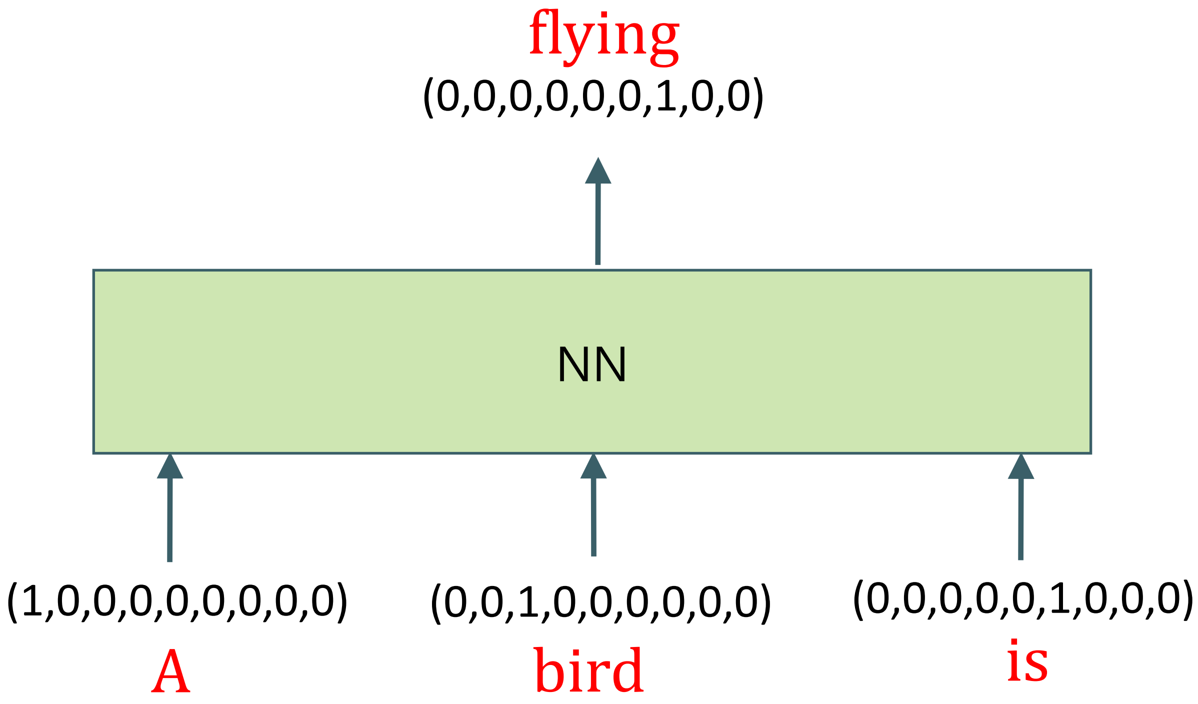
… …

Unique words

	a	an	bird	dong	fish	is	flying	running	swims	…
A	(1 ,	0 ,	0 ,	0 ,	0 ,	0 ,	0 ,	0 ,	0 ,	…)
bird	(0 ,	0 ,	1 ,	0 ,	0 ,	0 ,	0 ,	0 ,	0 ,	…)
is	(0 ,	0 ,	0 ,	0 ,	0 ,	1 ,	0 ,	0 ,	0 ,	…)

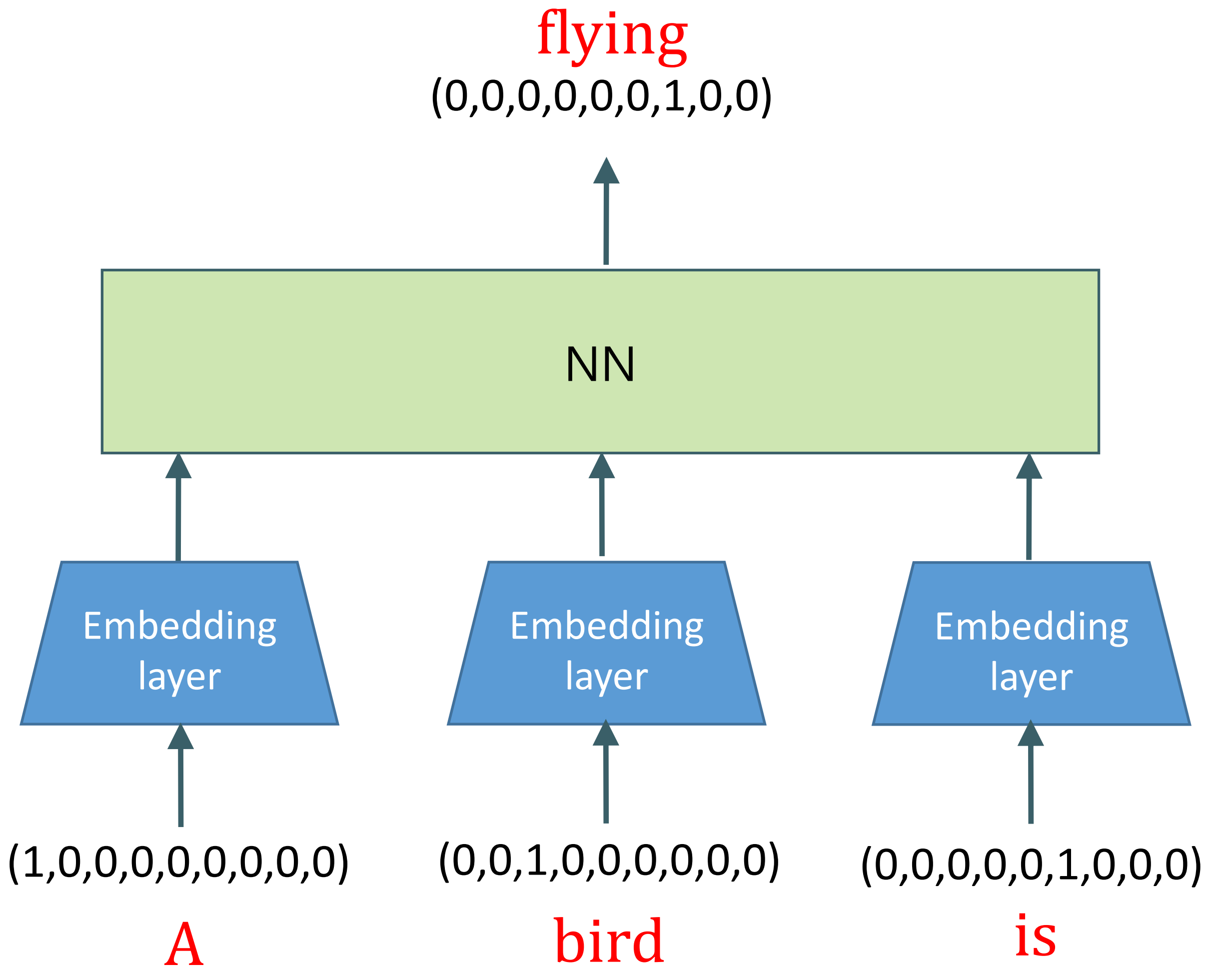
Word Representation

- **How to Handle Texts**



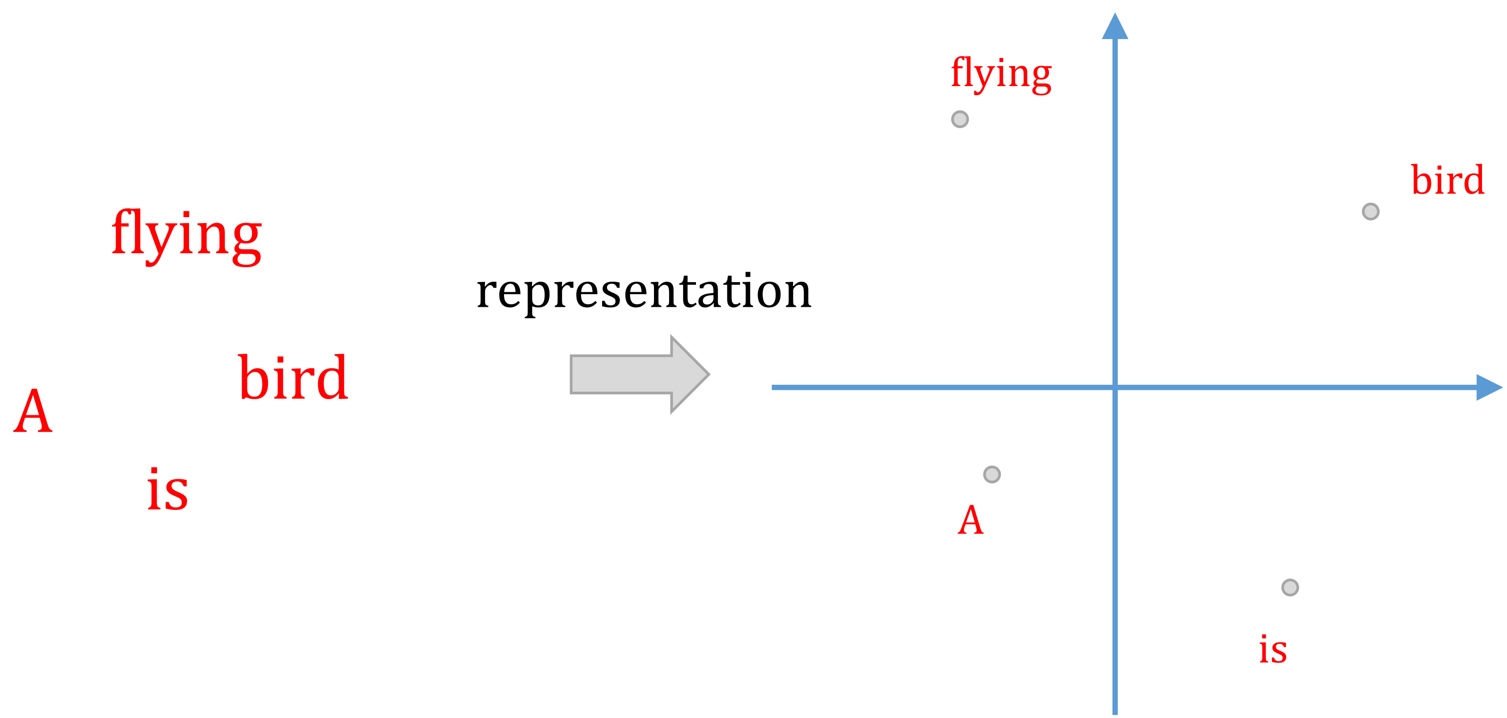
Word Representation

➢ **How to Handle Texts**



Word Representation

➢ Representation, Embedding

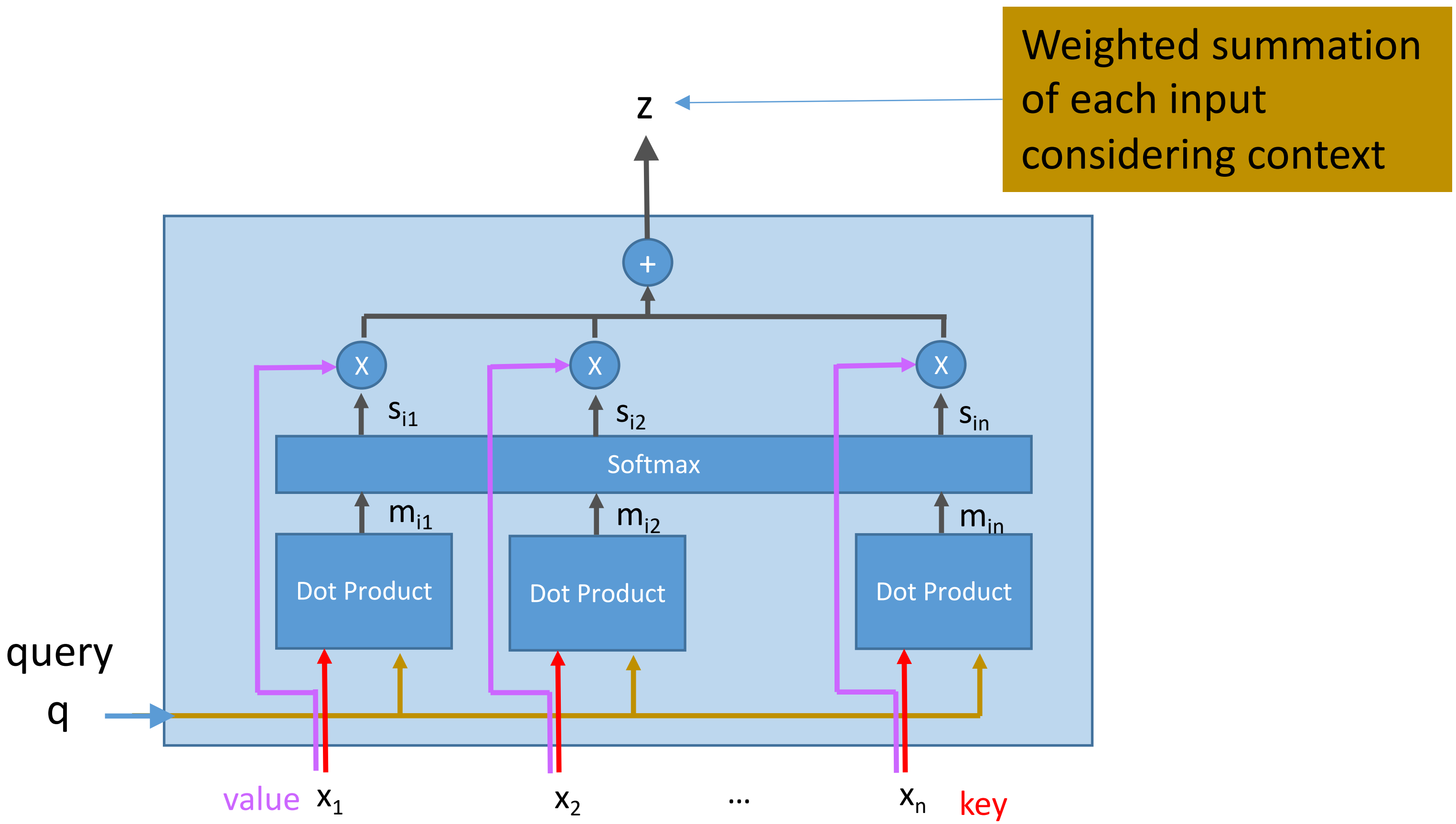


Values in non-Euclidean space

Values in Euclidean space

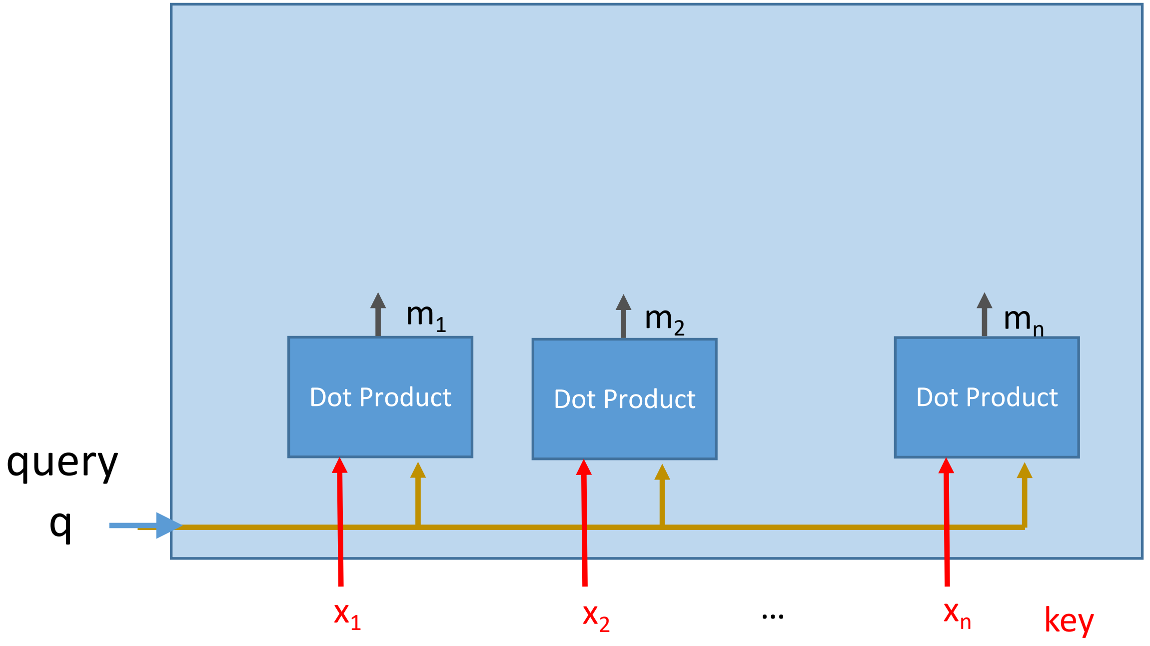
Attention Mechanism

➢ Overview



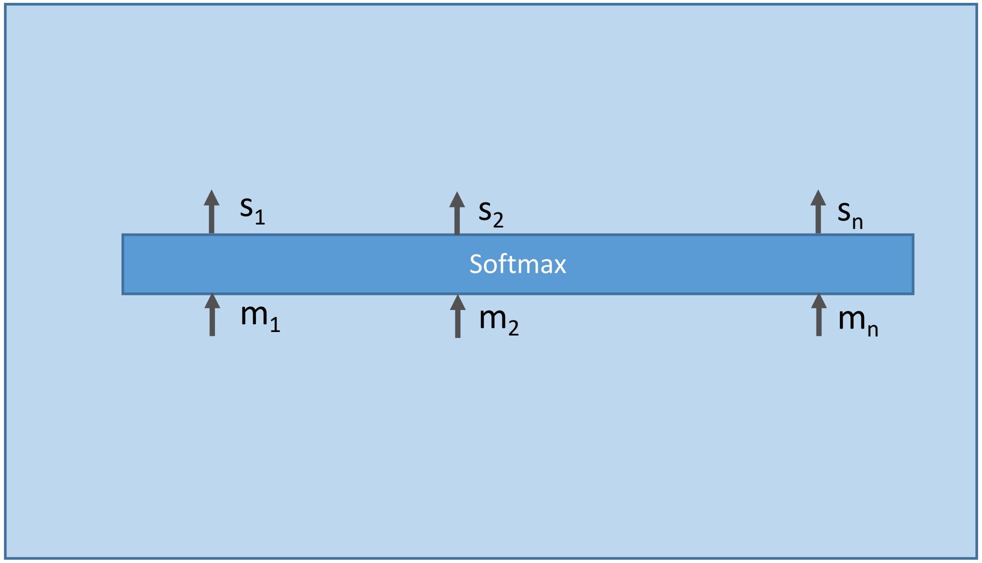
Attention Mechanism

➢ **How it works**



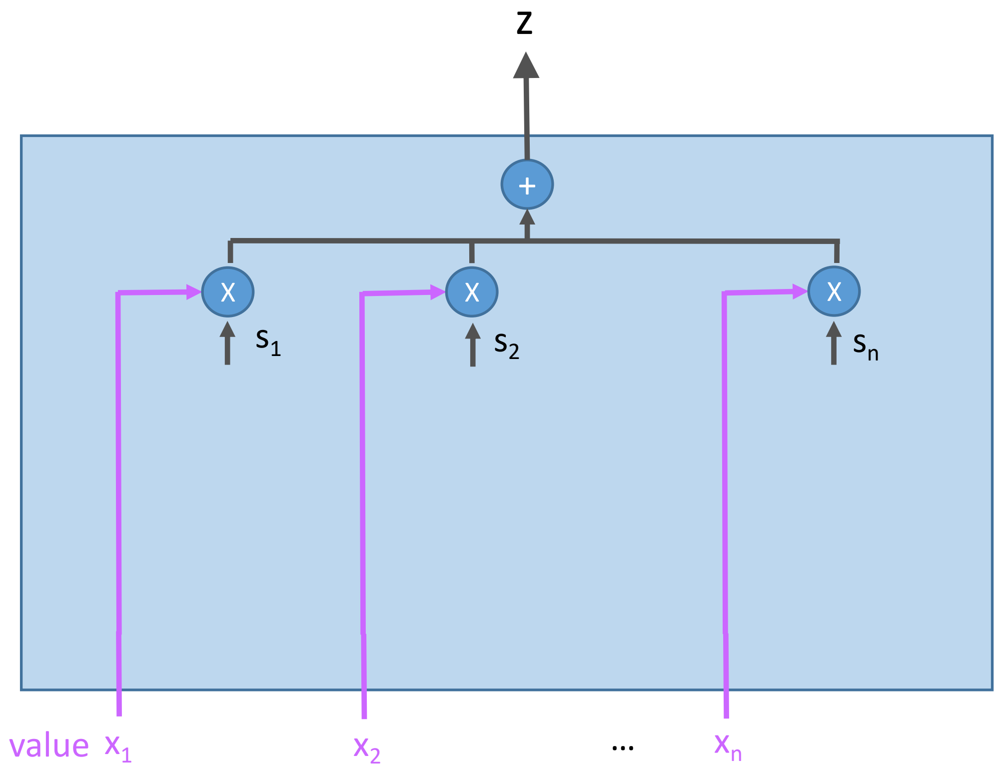
Attention Mechanism

➢ **How it works**



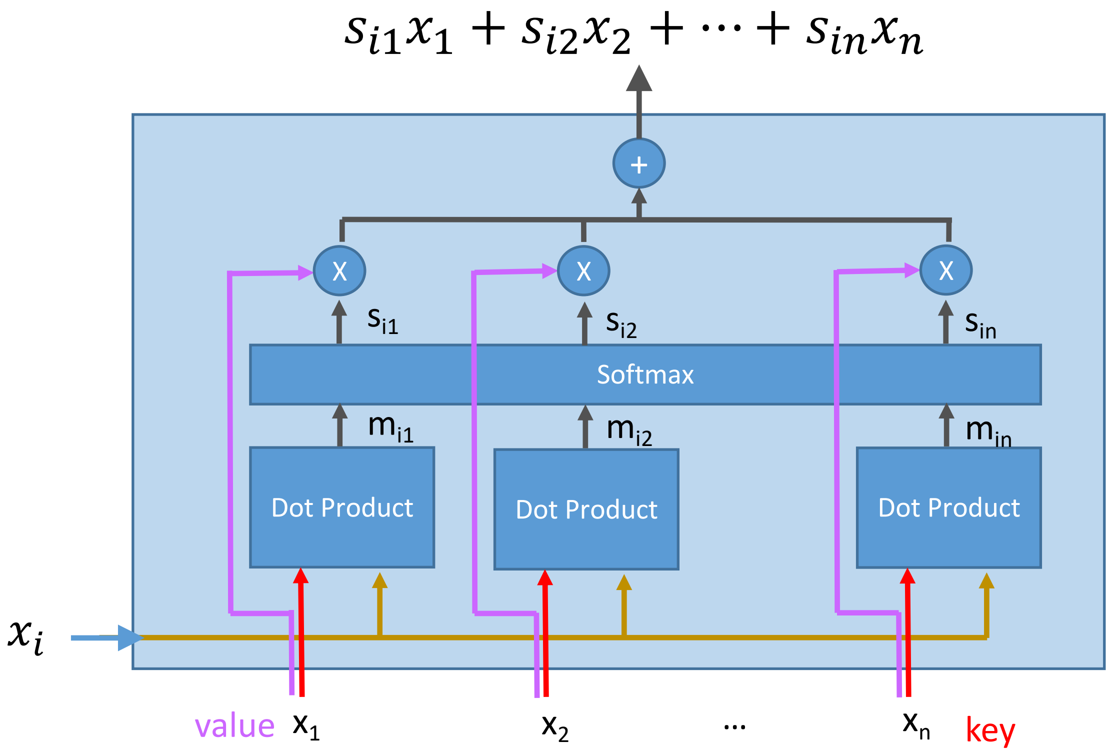
Attention Mechanism

➢ **How it works**



Attention Mechanism

- **How it works:** $A(x_i, X, X)$

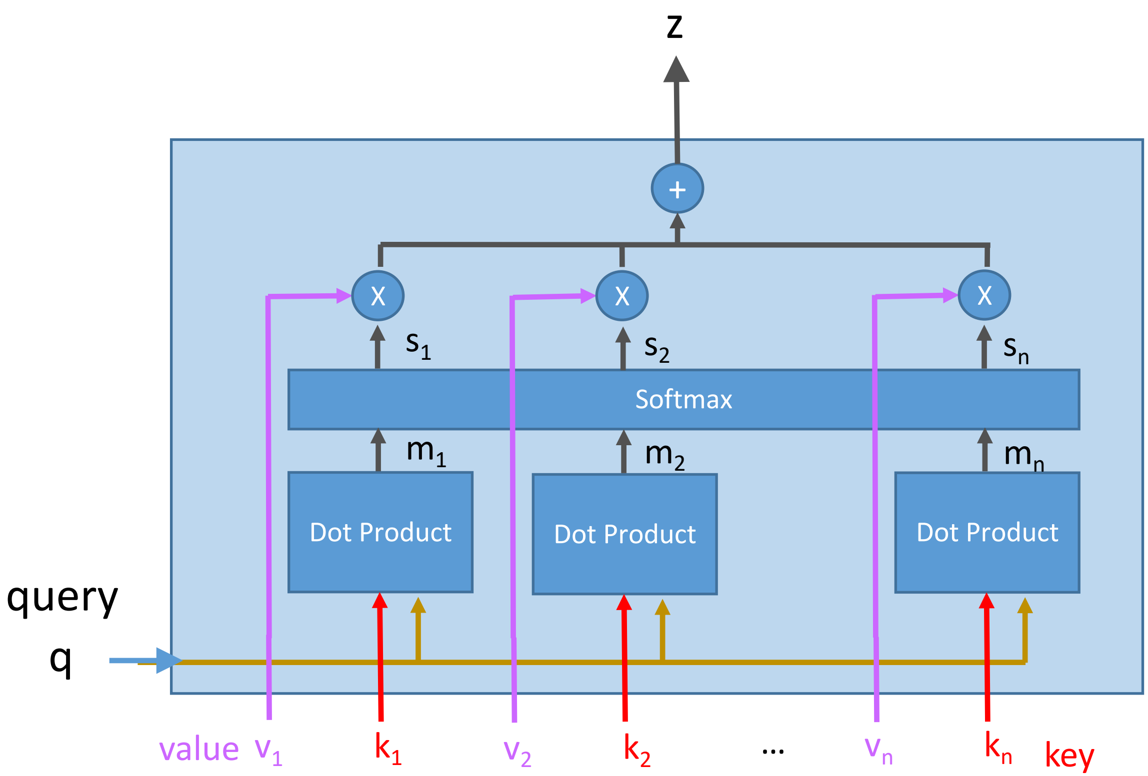


$$A(x_i, X, X) = s_{i1}x_1 + s_{i2}x_2 + \cdots + s_{in}x_n = x_i'$$

Attention Mechanism

➢ General Form

$$z = A(q, K, V)$$



$$q = \boxed{}$$

$$V = \begin{pmatrix} v_1 \\ v_2 \\ \cdots \\ v_n \end{pmatrix}$$

$$K = \begin{pmatrix} k_1 \\ k_2 \\ \cdots \\ k_n \end{pmatrix}$$

Attention Mechanism

➢ Multiple Queries

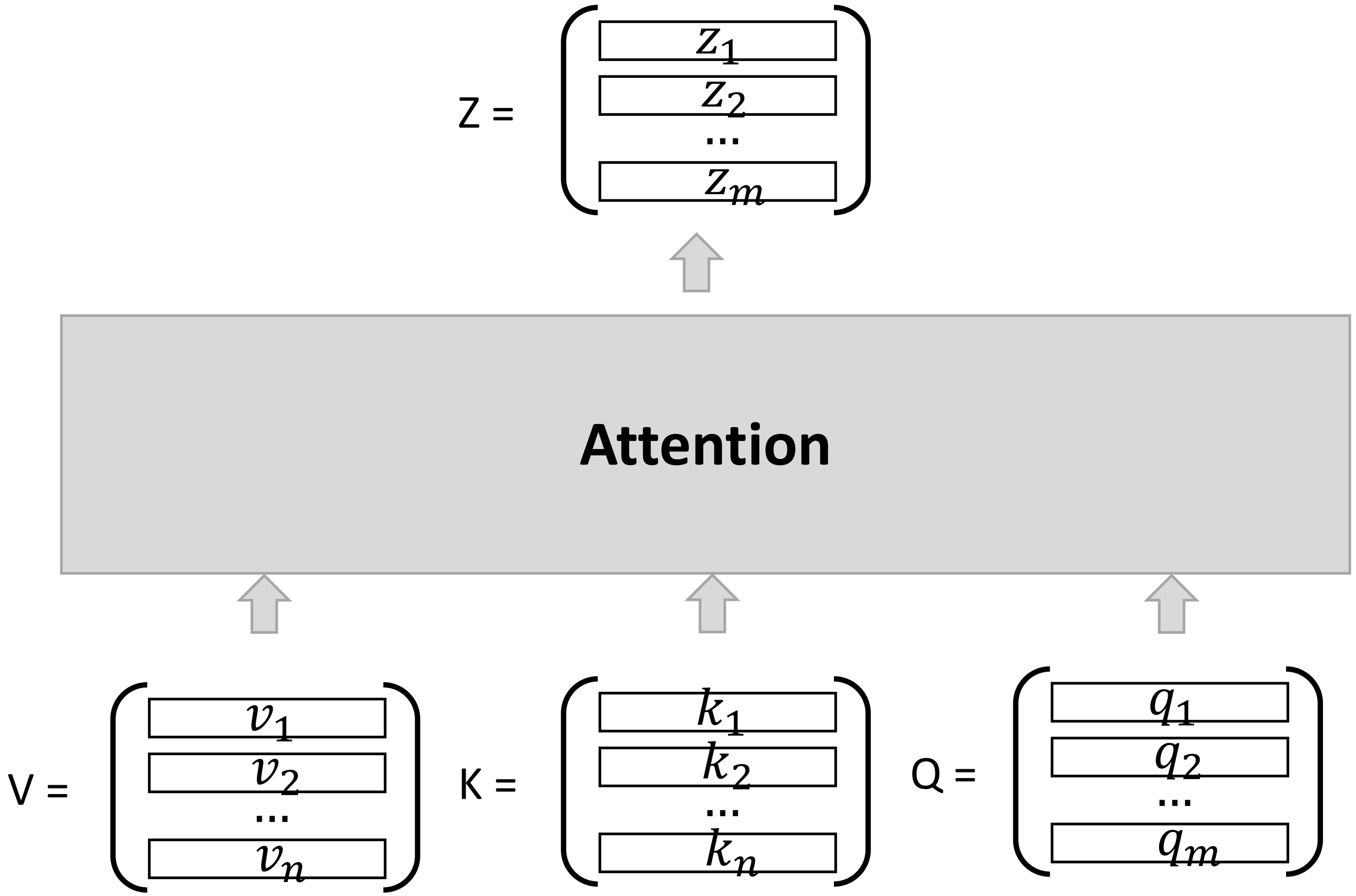
$$\left.\begin{aligned} z_1 &= A(q_1, K, V) \\ z_2 &= A(q_2, K, V) \\ &\dots \\ z_m &= A(q_m, K, V) \end{aligned}\right\} \quad Z = A(Q, K, V)$$

$$Z = \begin{pmatrix} \boxed{\quad z_1 \quad} \\ \boxed{\quad z_2 \quad} \\ \dots \\ \boxed{\quad z_m \quad} \end{pmatrix} \qquad Q = \begin{pmatrix} \boxed{\quad q_1 \quad} \\ \boxed{\quad q_2 \quad} \\ \dots \\ \boxed{\quad q_m \quad} \end{pmatrix}$$

$$K = \begin{pmatrix} \boxed{\quad k_1 \quad} \\ \boxed{\quad k_2 \quad} \\ \dots \\ \boxed{\quad k_n \quad} \end{pmatrix} \qquad V = \begin{pmatrix} \boxed{\quad v_1 \quad} \\ \boxed{\quad v_2 \quad} \\ \dots \\ \boxed{\quad v_n \quad} \end{pmatrix}$$

Attention Mechanism

- $Z = A(Q, K, V)$

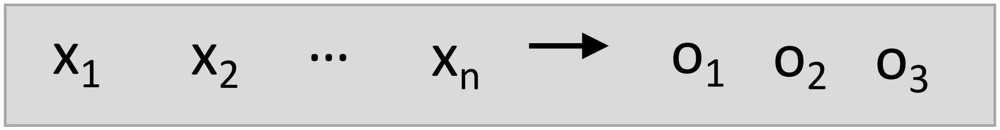


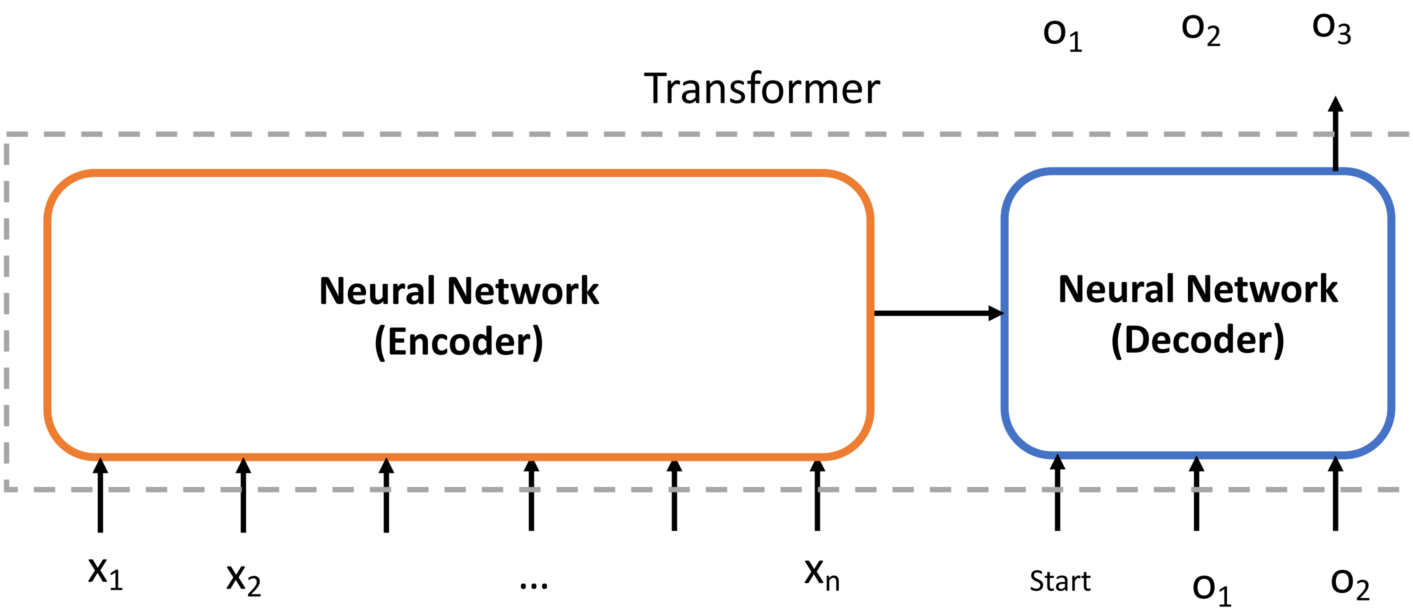
Transformer Overview

- It would take a sentence in source language, and output its translation in another.

Transformer Overview

➢ How it works

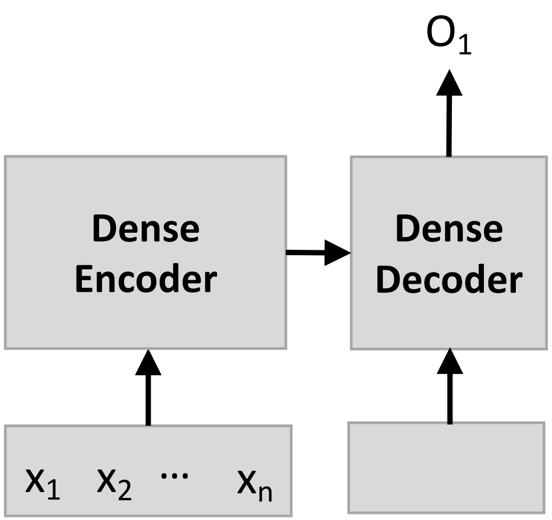




Transformer Overview

➢ **Encoder & Decoder**

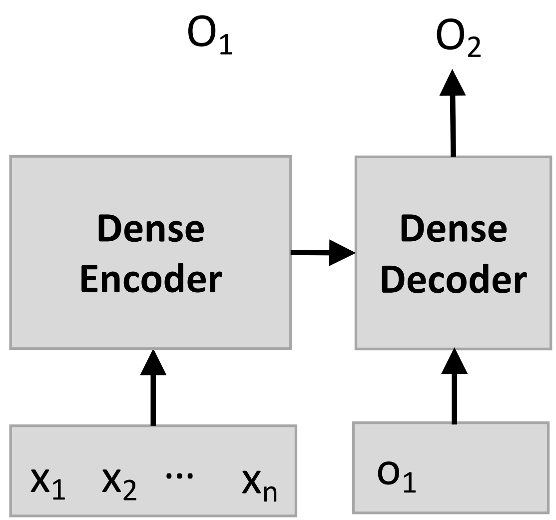
$$x_1 \quad x_2 \quad \cdots \quad x_n \longrightarrow o_1$$



Transformer Overview

➢ Encoder & Decoder

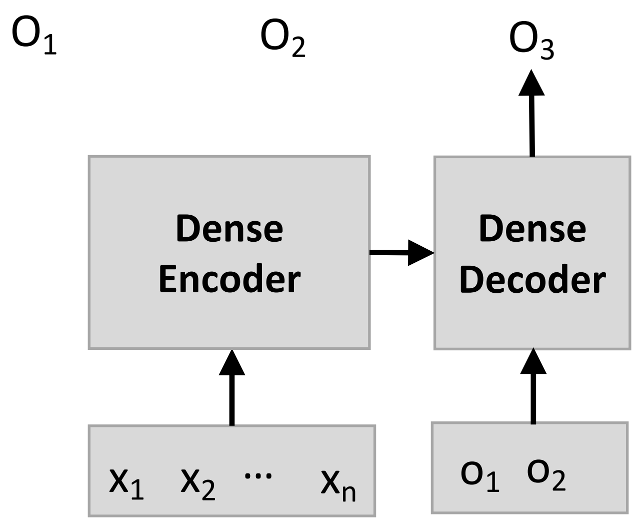
$$x_1 \quad x_2 \quad \cdots \quad x_n \longrightarrow o_1 \quad o_2$$



Transformer Overview

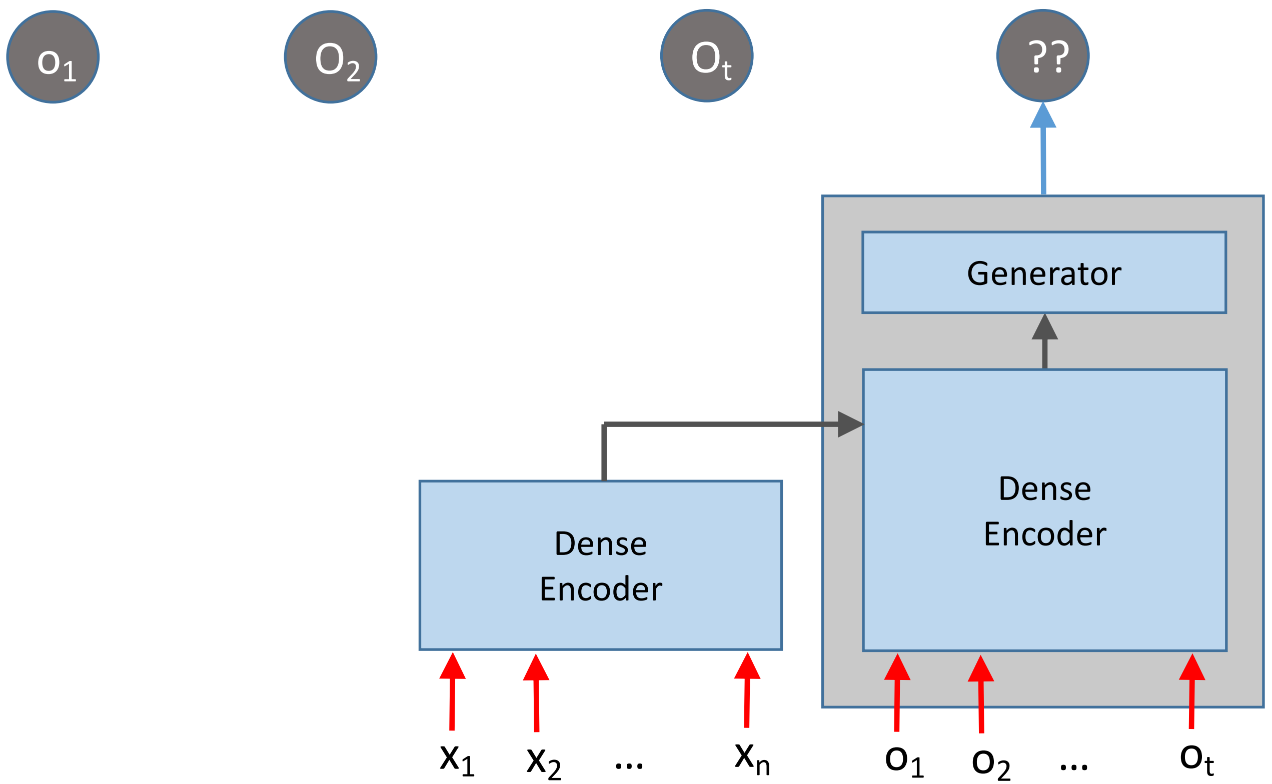
➢ Encoder & Decoder

$x_1 \quad x_2 \quad \cdots \quad x_n \longrightarrow o_1 \quad o_2 \quad o_3$



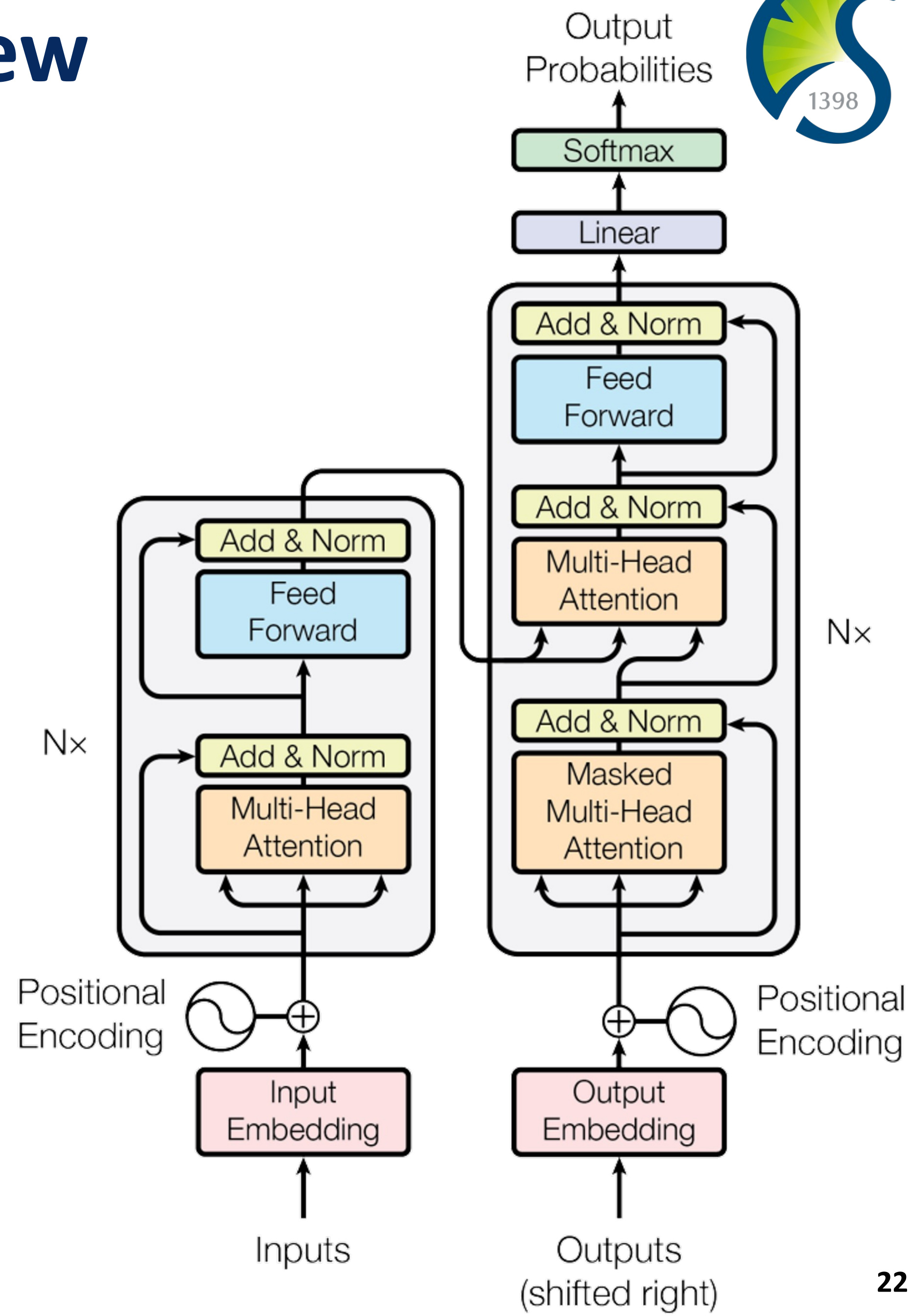
Transformer Overview

➢ **Encoder & Decoder**

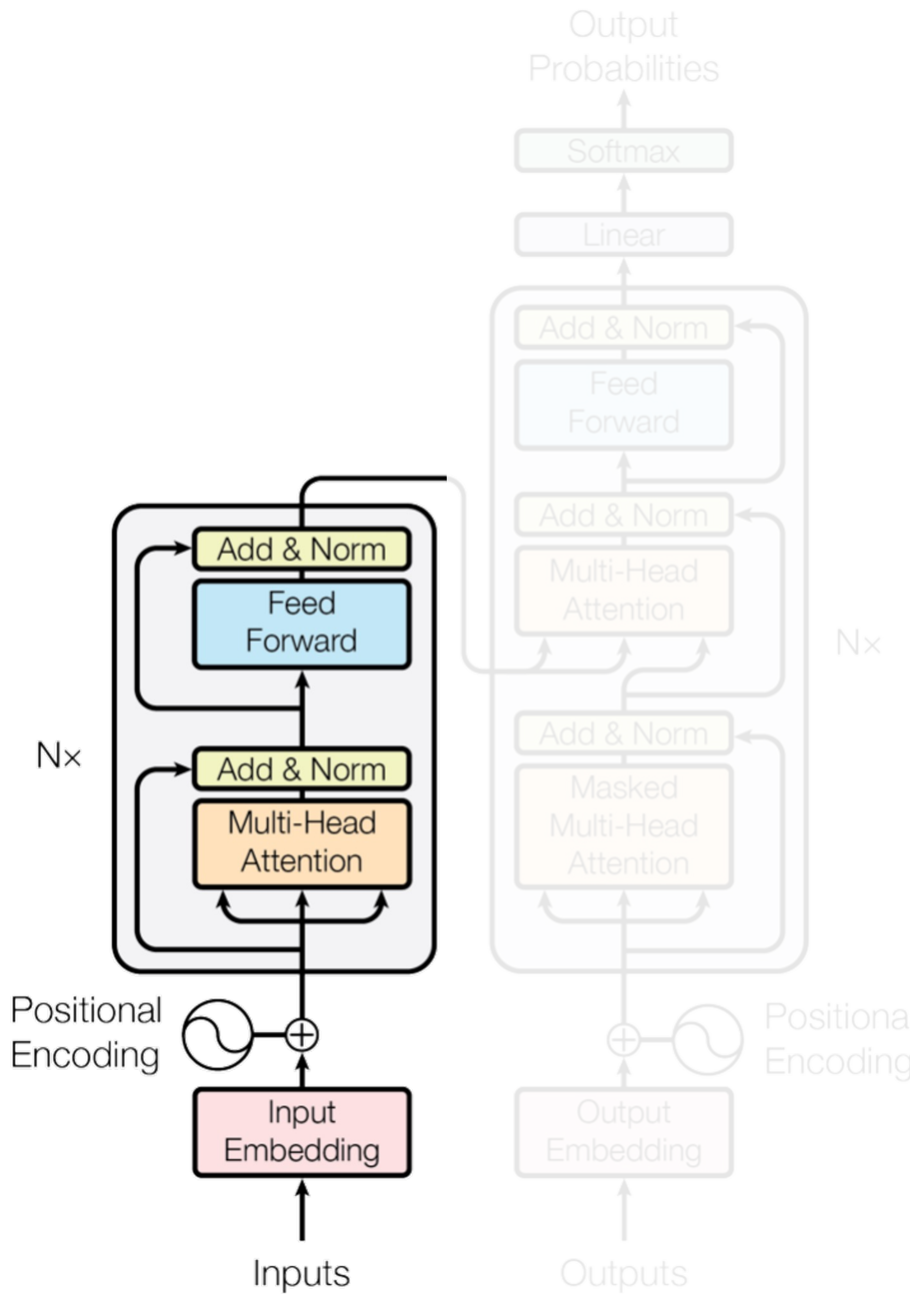


Transformer Overview

- ➢ **Encoder-Decoder approach**
- ➢ **Task: machine translation with parallel corpus**
- ➢ **Predict each translated word.**

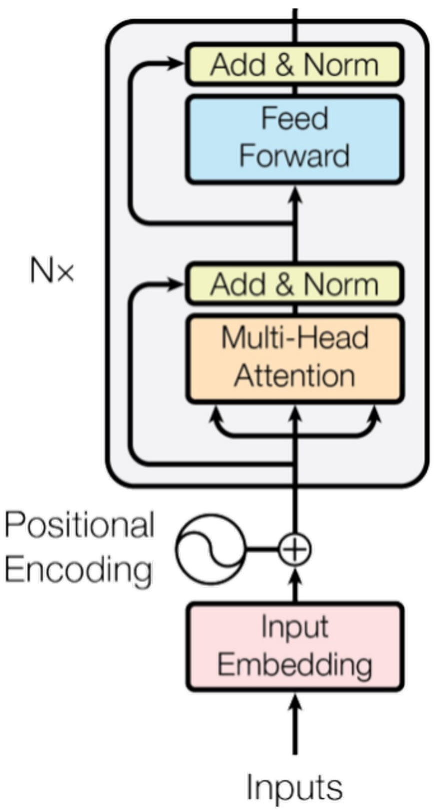


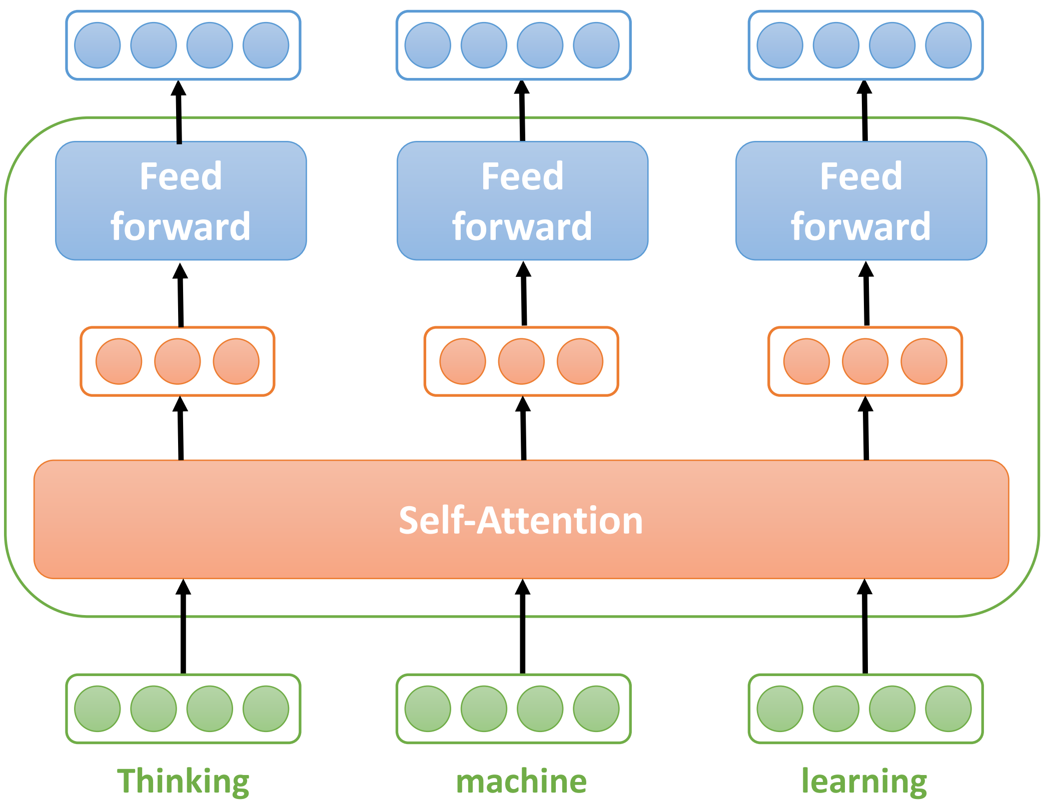
Encoder


Output
Probabilities
Softmax
Linear
Add & Norm
Feed
Forward
Add & Norm
Multi-Head
Attention
Nx
Add & Norm
Masked
Multi-Head
Attention
Add & Norm
Feed
Forward
Add & Norm
Multi-Head
Attention
Nx
Positional
Encoding
Input
Embedding
Inputs
Positional
Encoding
Output
Embedding
Outputs

Encoder Internals

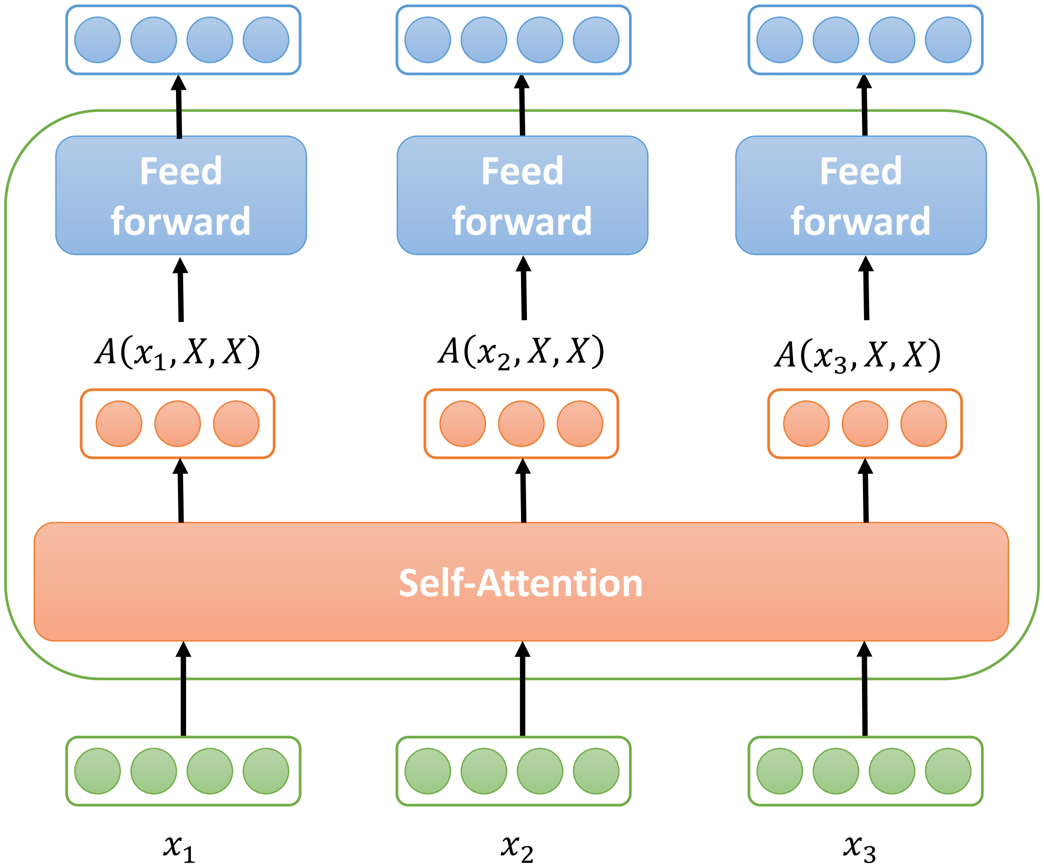
- After embedding the words in the input sentence, each of them flows through the two layers of the encoder.

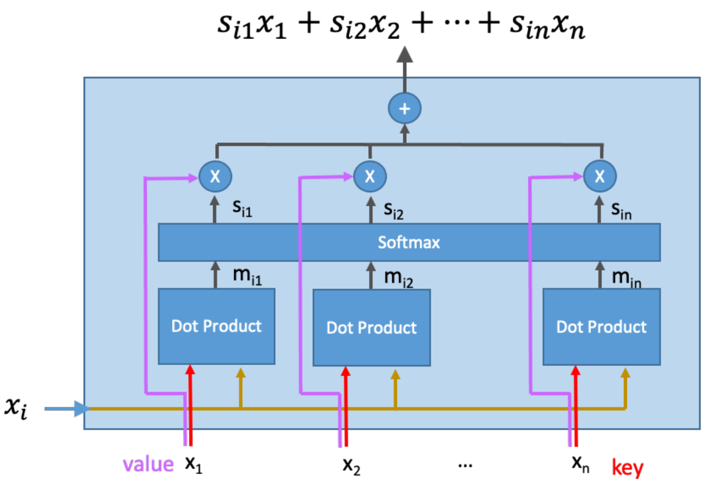




Encoder Internals

- ➢ **Self-Attention Layer in Transformer**

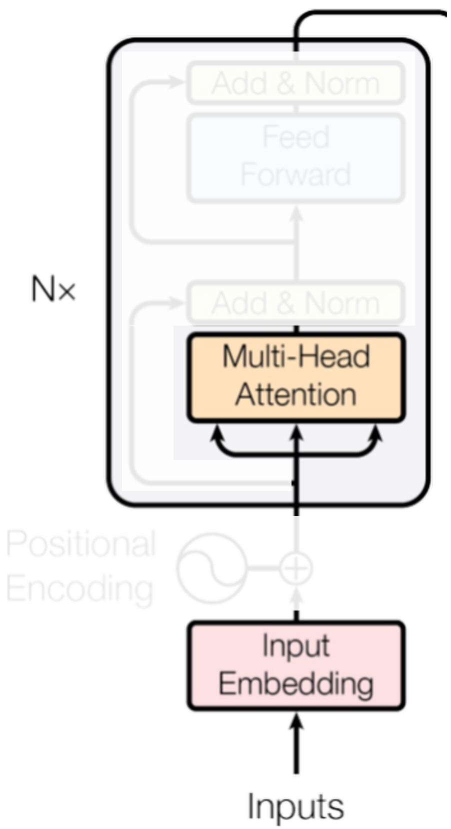




$$X = \begin{bmatrix} x_1 \\ x_2 \\ \cdots \\ x_n \end{bmatrix}$$

Multi-headed Attention

- **Refine the self-attention layer by adding a mechanism called “multi-headed” attention.**
  - It expands the model’s ability to focus on different positions.
  - It gives the attention layer **multiple representation subspaces**.



Multi-headed Attention

- **Attention**

$$A(Q,K,V) = softmax(QK^T)V$$

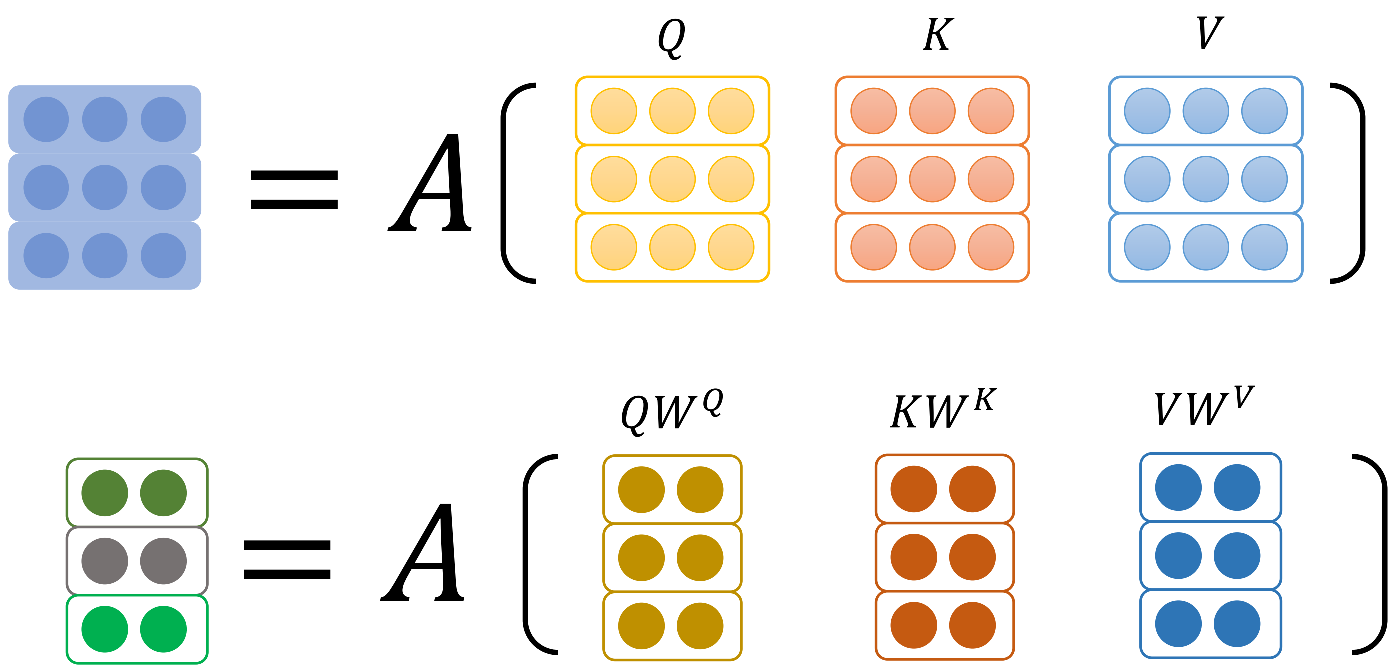
$$Q: |Q| \times k, \qquad K: |K| \times k, \qquad V: |V| \times k$$

- **What is 'headed'?**
  - Linear Transformed: $W^Q, W^K, W^V \ (= k \times m)$

$$A(QW^Q, KW^K, VW^V)$$

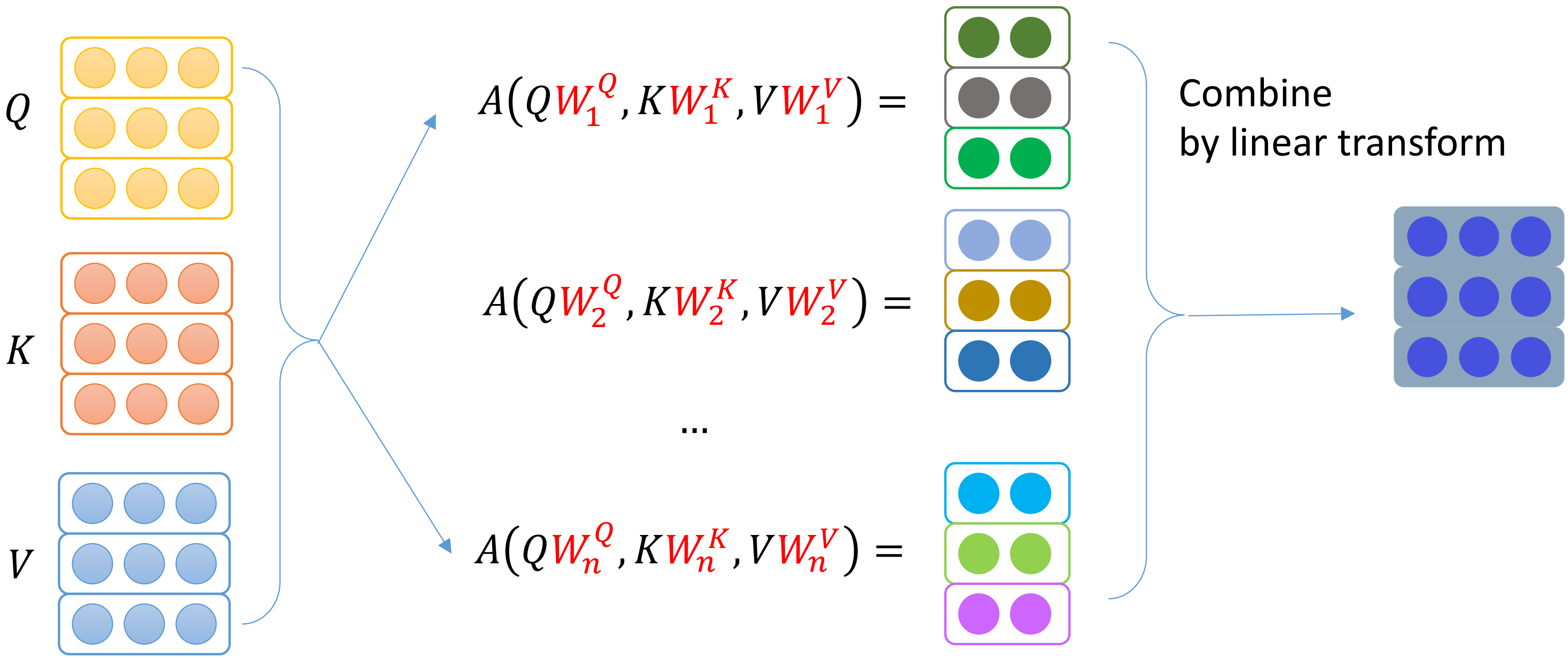
Multi-headed Attention

$$A(QW^Q, KW^K, VW^V)$$



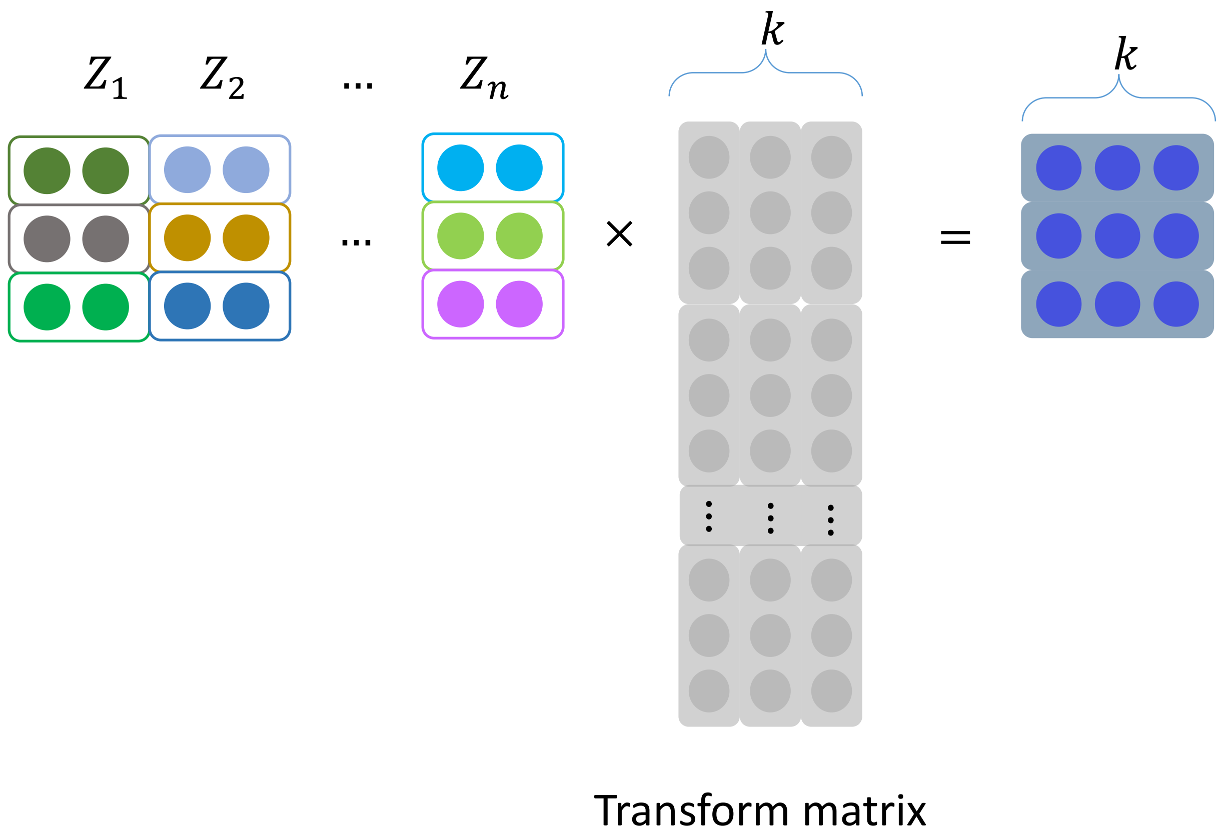
Multi-headed Attention

- Let's use multiple heads to capture various similiarities



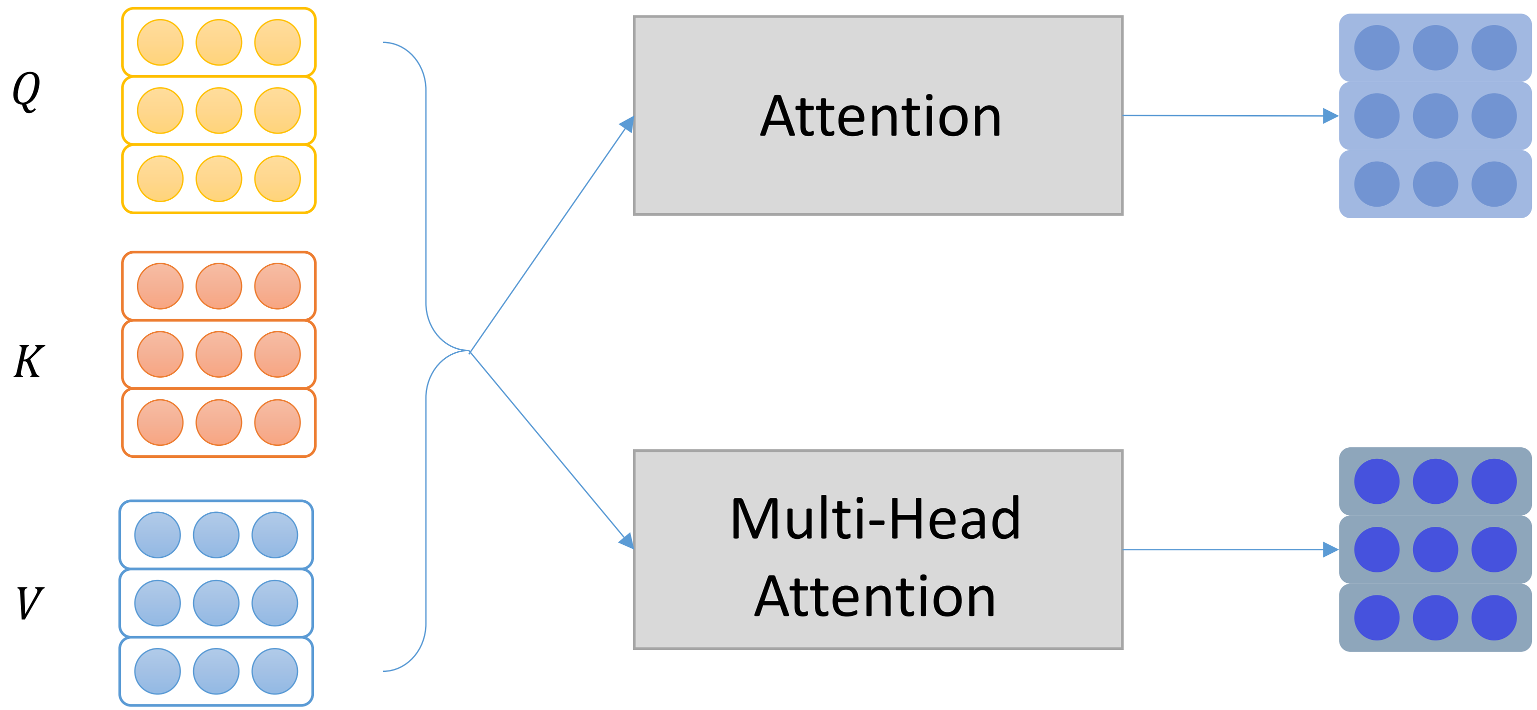
Multi-headed Attention

➢ **Let's use multiple heads to capture various similiarities**



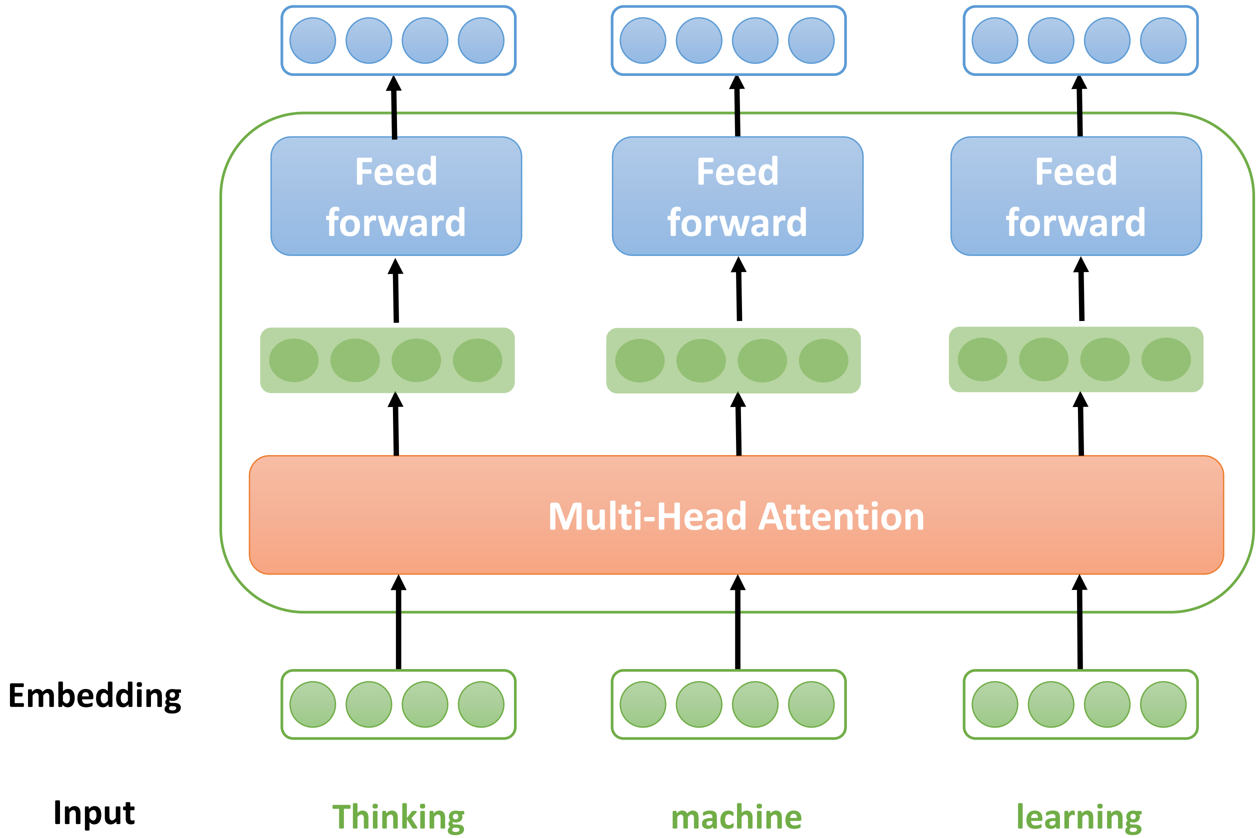
Multi-headed Attention

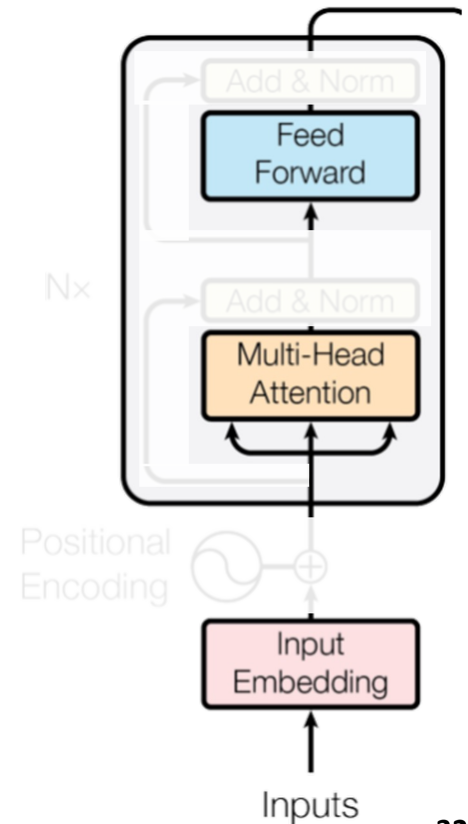
- Let's use multiple heads to capture various similiarities



Feed Forward

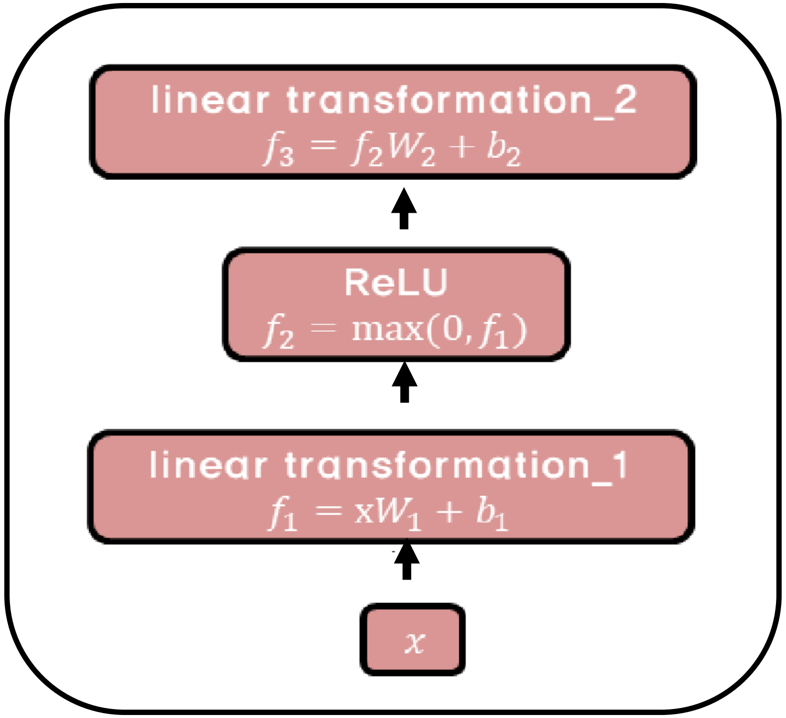
- **Point-wise Feed Forward**

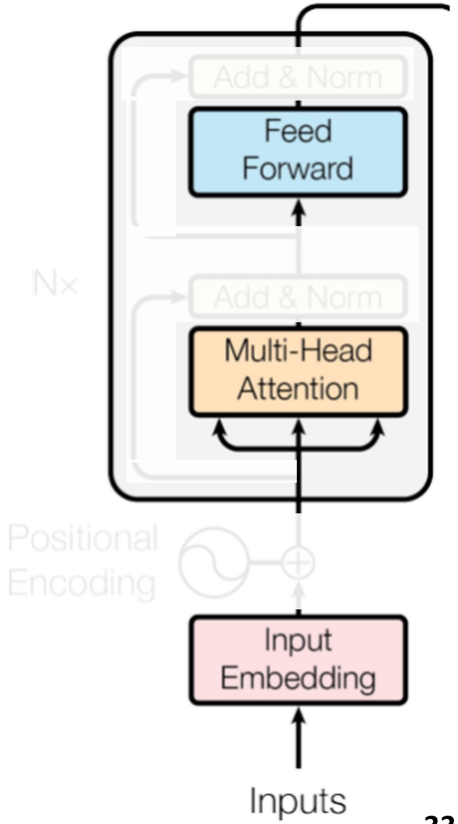




Feed Forward

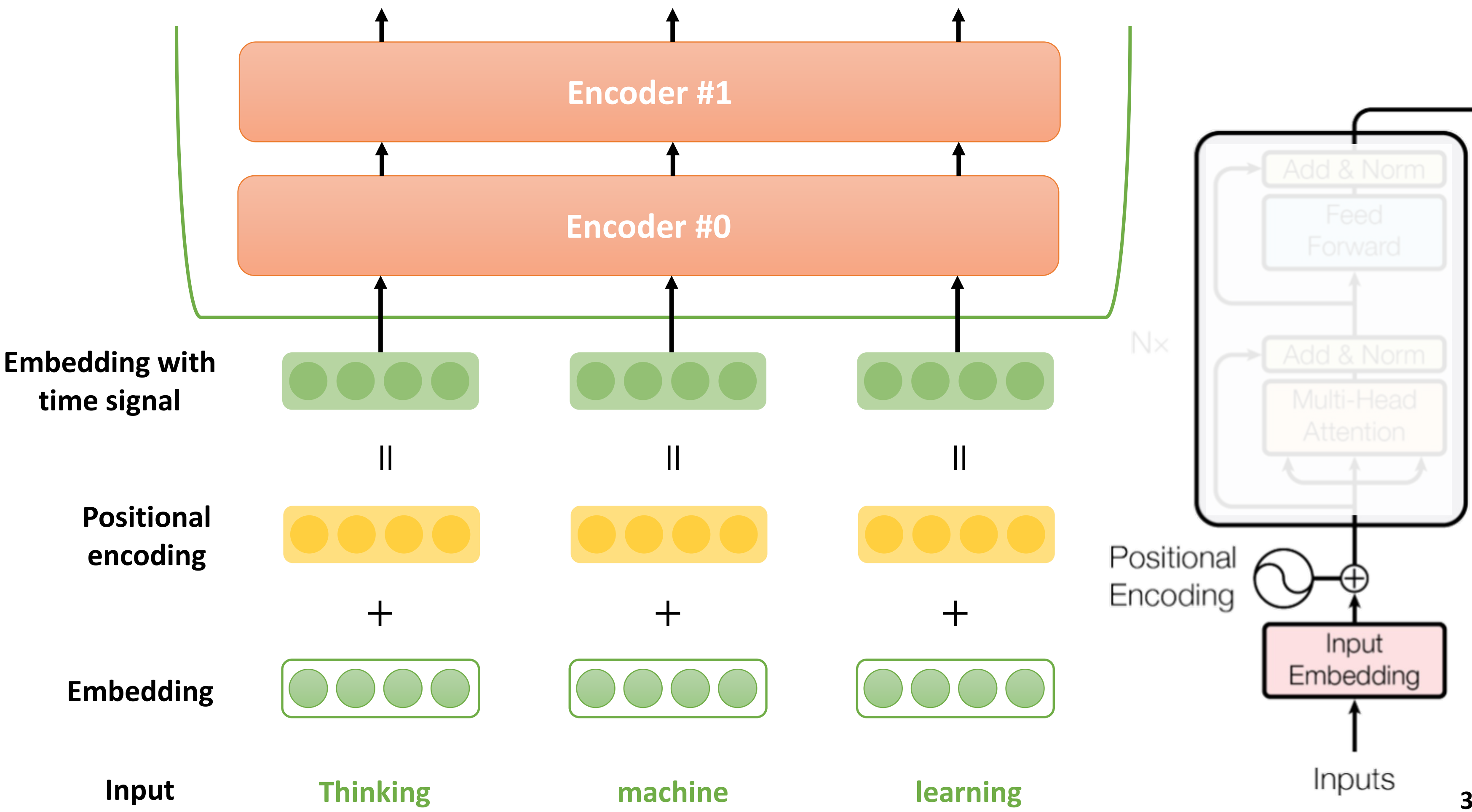
➢ **Point-wise Feed Forward**





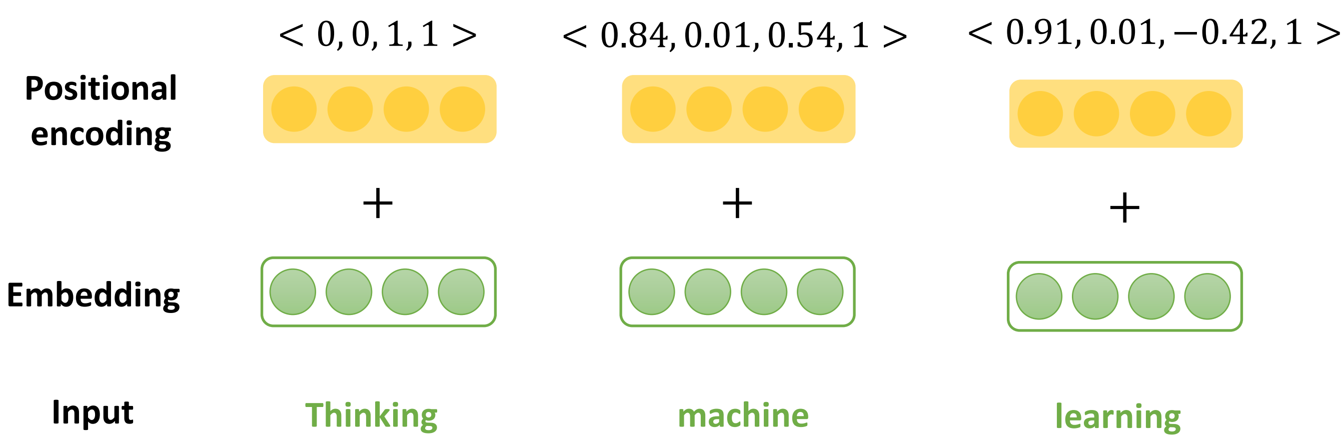
Positional Encoding

➢ **Need to consider the order of the words in the input sentence.**



Positional Encoding

- **Assume that embedding has a dimensionality of 4.**
- **The positional embedding would look like this:**
  - Use cosine and sine curves.



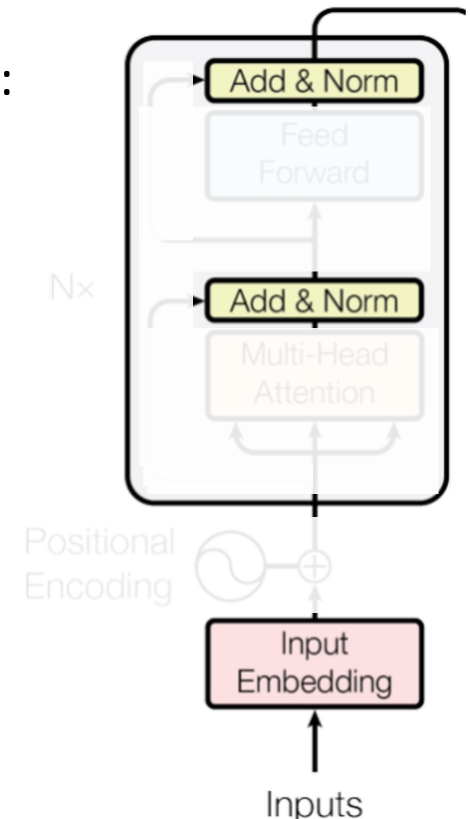
$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{model}})$$

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

Complete Transformer Block

- **Each block has two sublayers.**
  - Multi-head attention
  - 2 layer feed-forward net (with relu)
- **Each of these two steps also has:**
  - **Residual (short-circuit) connection** and **LayerNorm**:

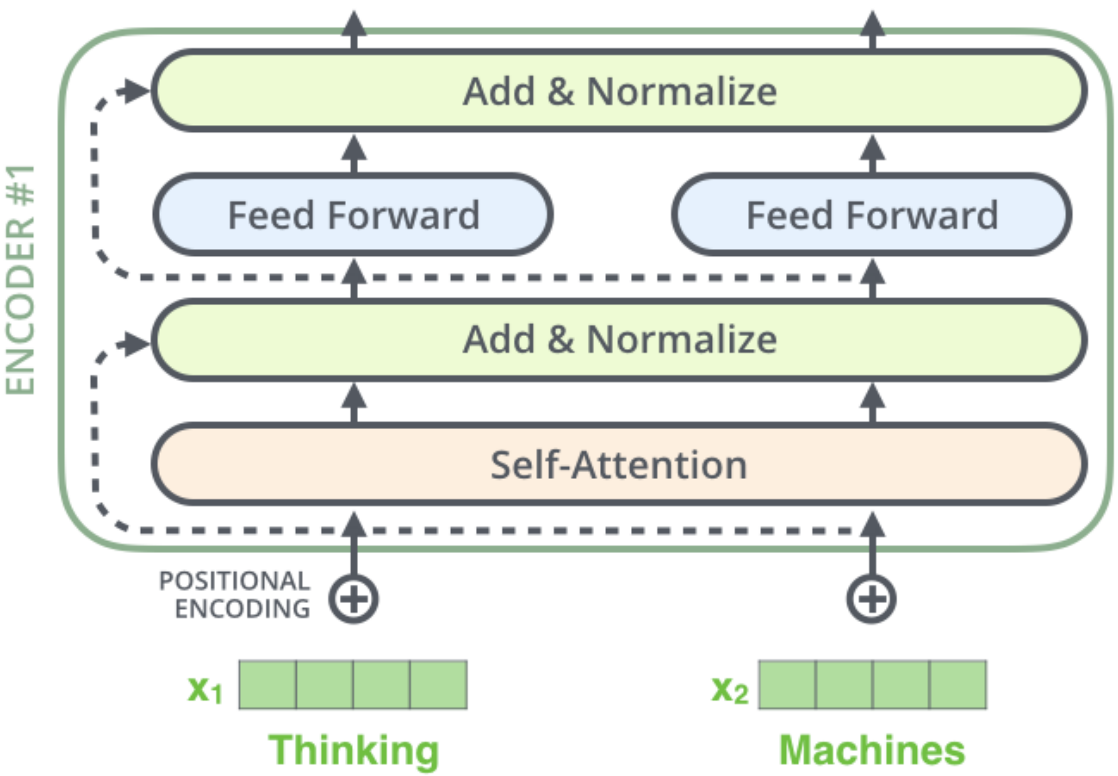
$$LayerNorm(X + Sublayer(X))$$

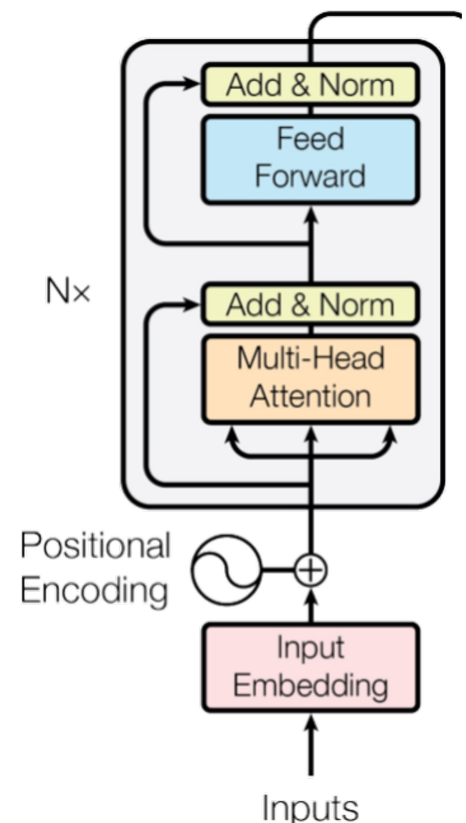


- **Layer normalization**
  - https://arxiv.org/pdf/1607.06450.pdf

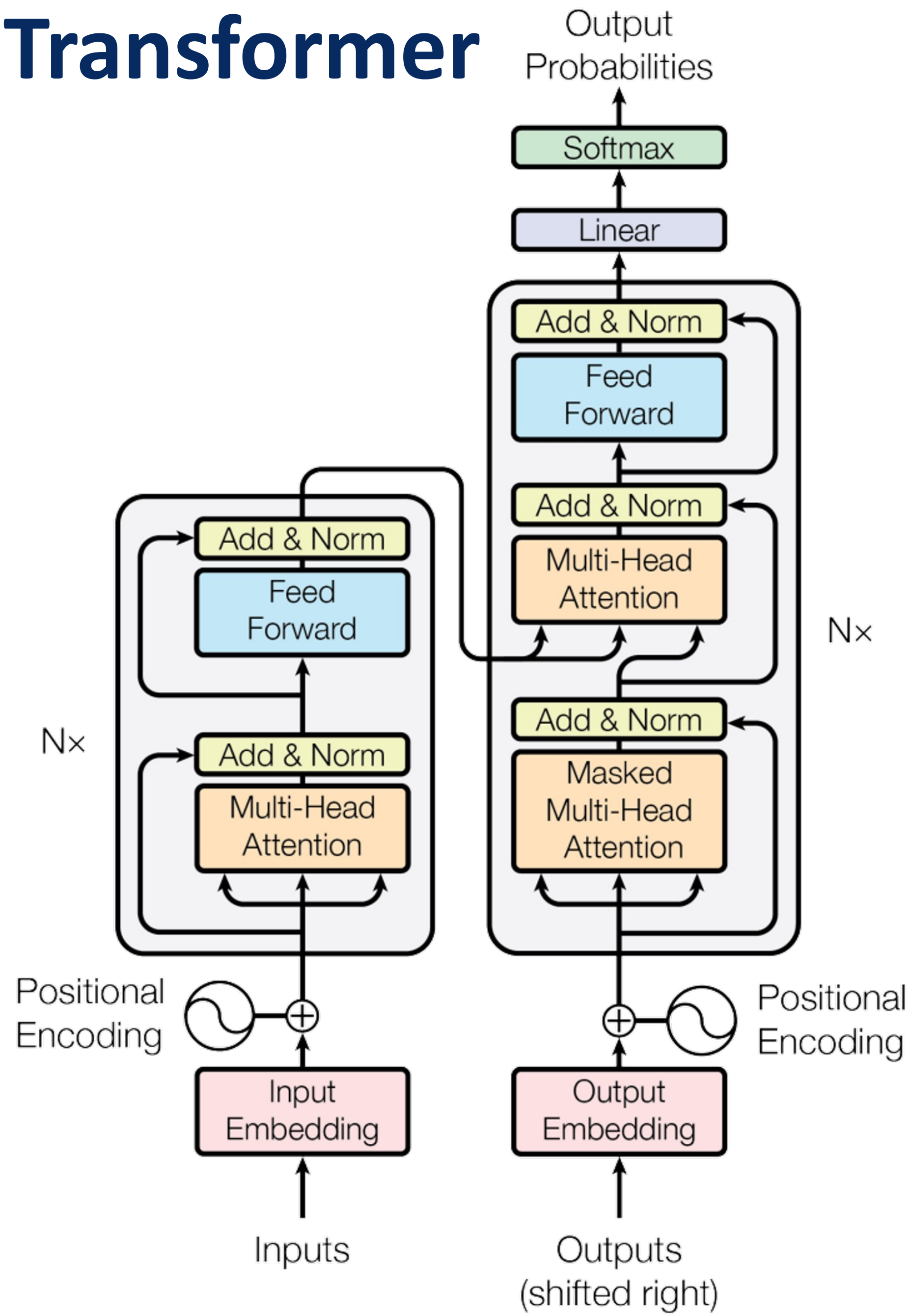
Residual Connection

➢ **Each sublayer in each encoder has a residual connection.**

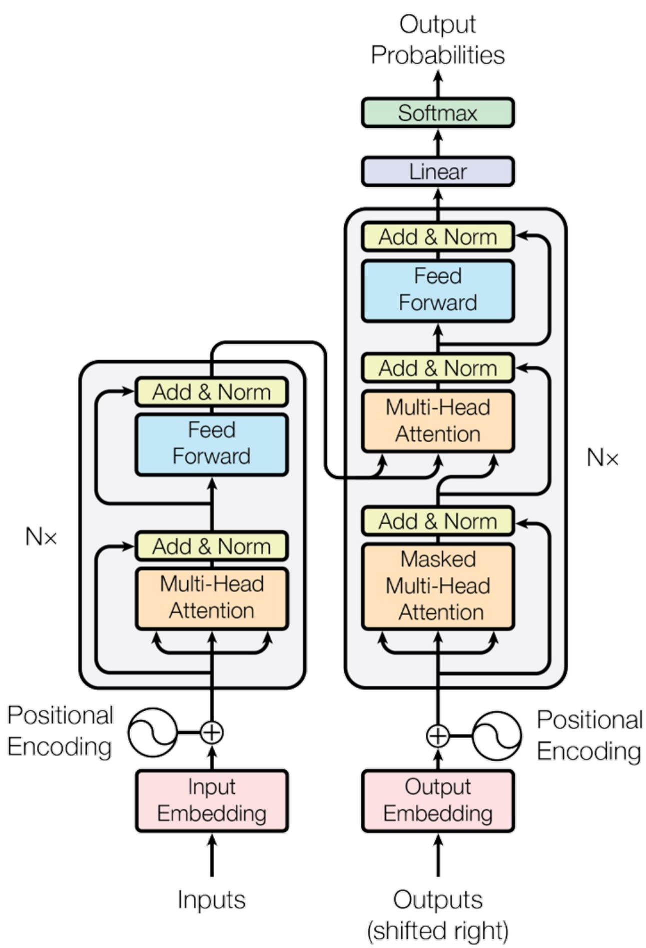


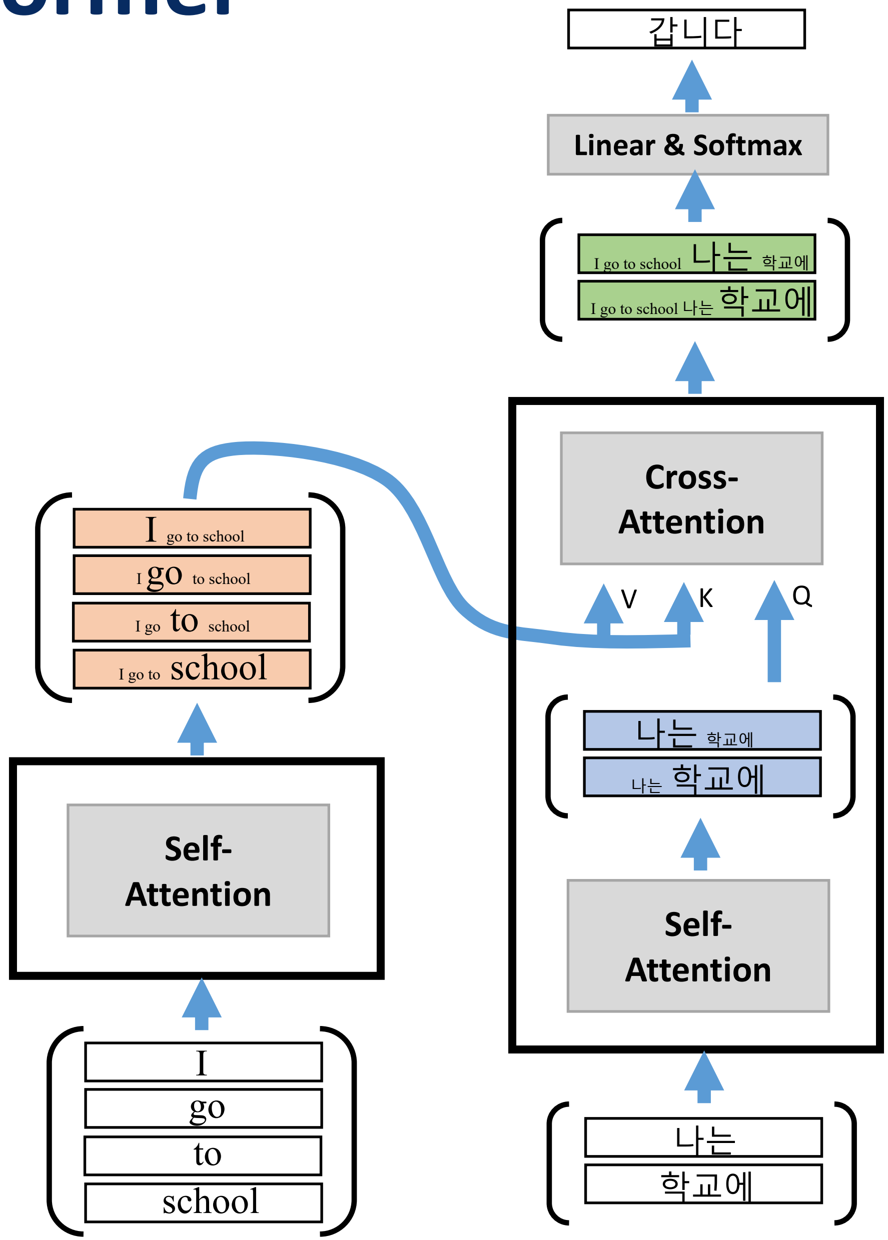


Simplified Transformer



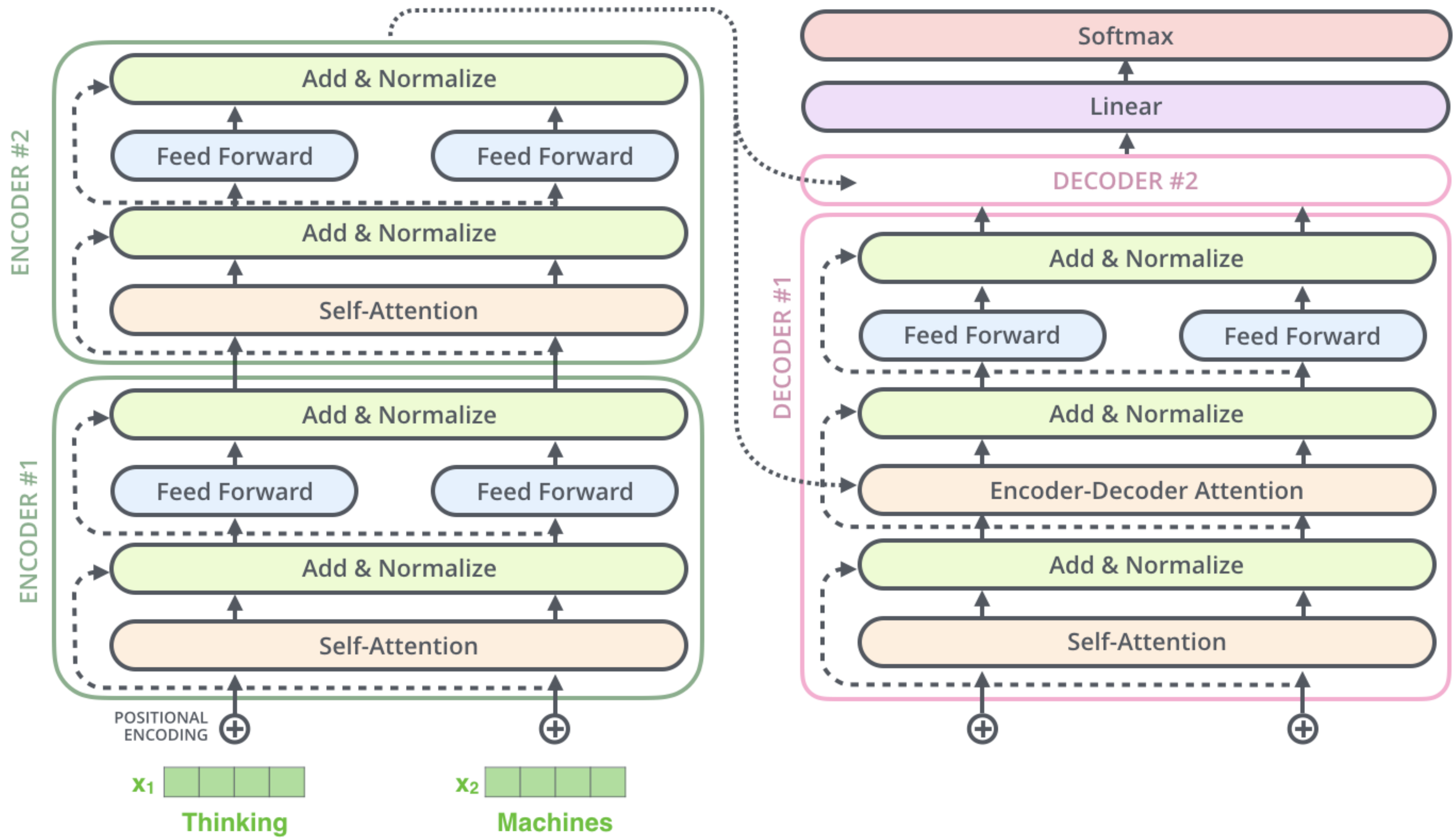
Simplified Transformer





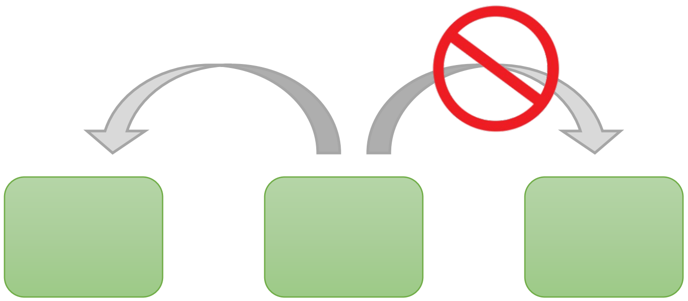
Simplified Transformer

➢ It consists of 2 stacked encoders and decoders.



Decoder: Encoder-Decoder Attention

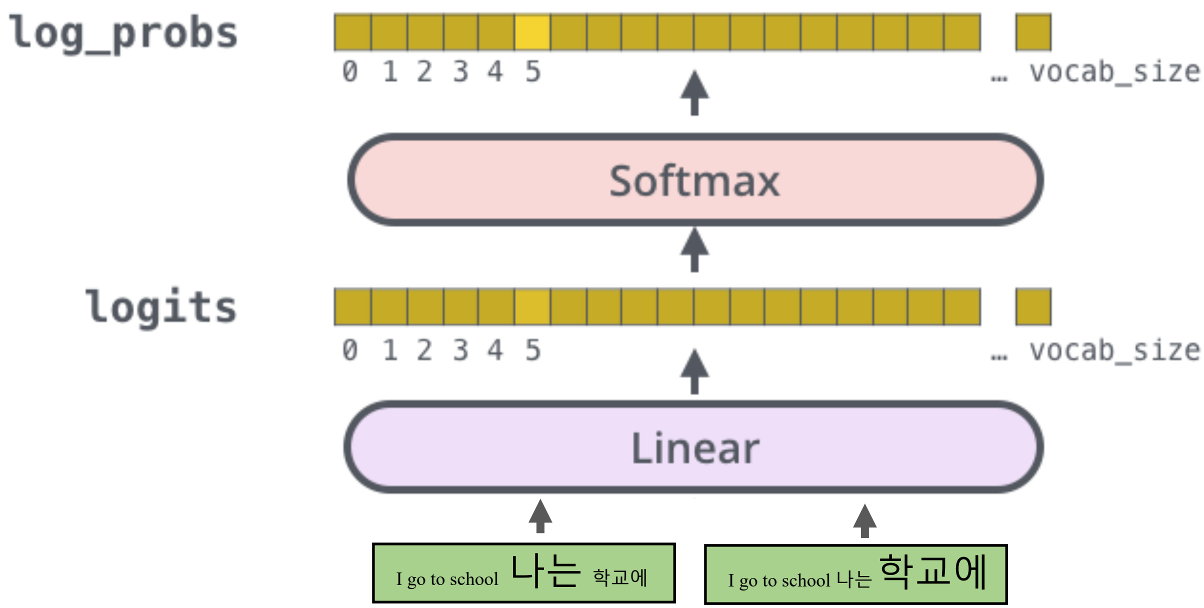
- **2 sublayer changes in decoder.**
- **Masked decoder**
  - Self-attention on previously generated outputs is only used.



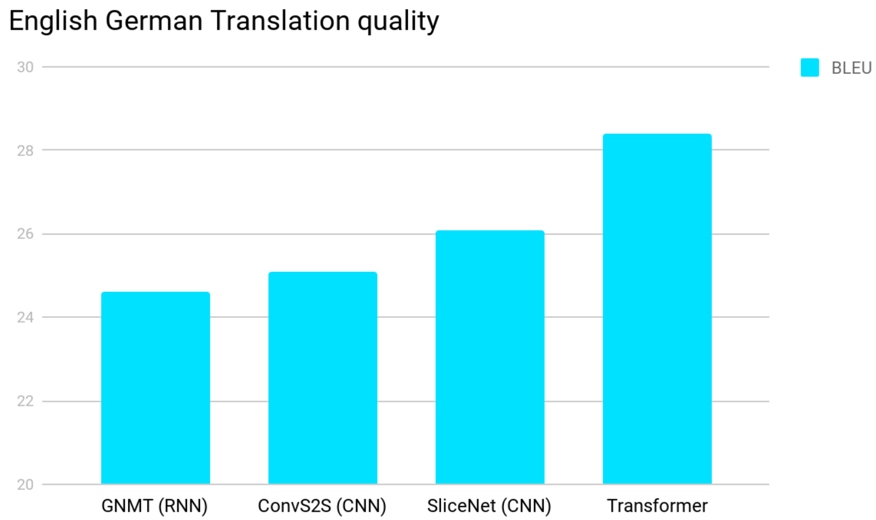
- **Encoder-decoder attention**
  - Queries come from previous decoder layer and keys and values come from output of encoder.

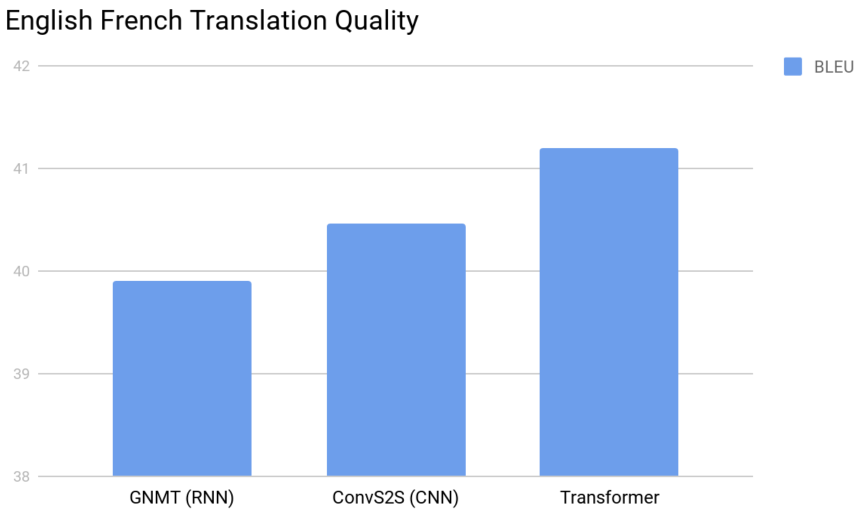
Final Linear and Softmax Layer

- The Linear layer is a simple fully connected neural network that projects the vector produced by the stack of decoders, into a much, much larger vector called a logits vector.
- The softmax layer then turns those scores into probabilities (all positive, all add up to 1.0).



Experimental Results





Experimental Results

- The Transformer achieves better BLEU scores than the previous model, and training cost is much smaller.

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