Transformer Model

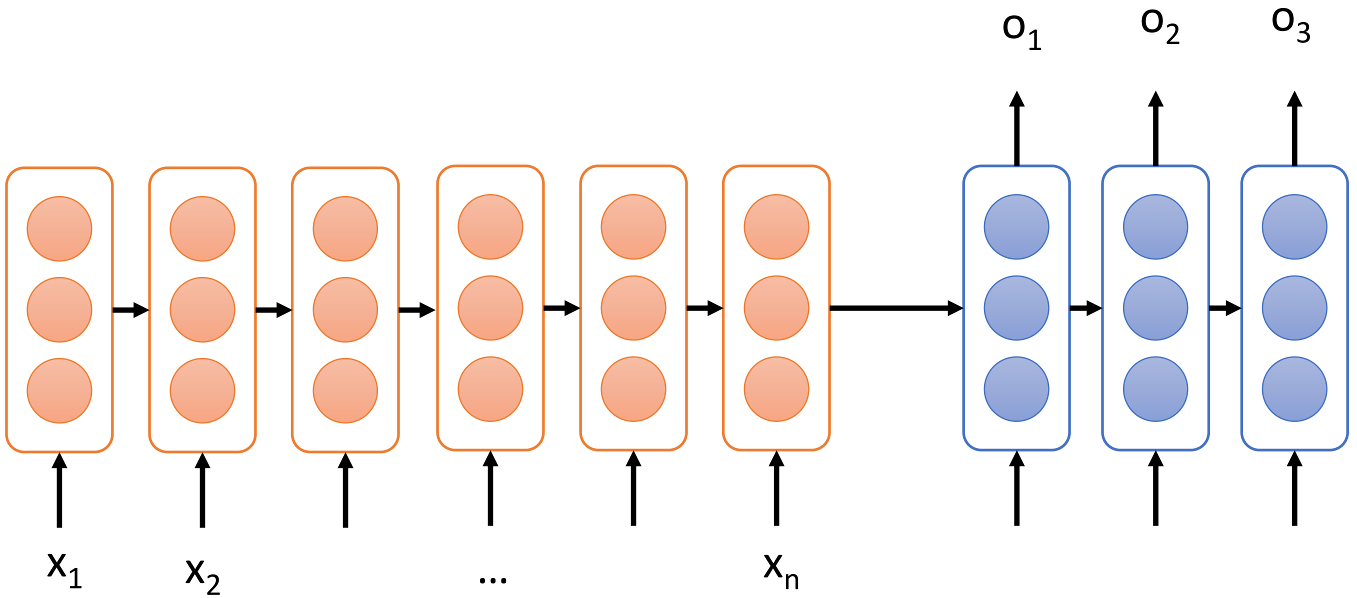
Credit to: Prof. Jongwuk Lee, SKKU

Sequence Generator

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

Problems with RNN

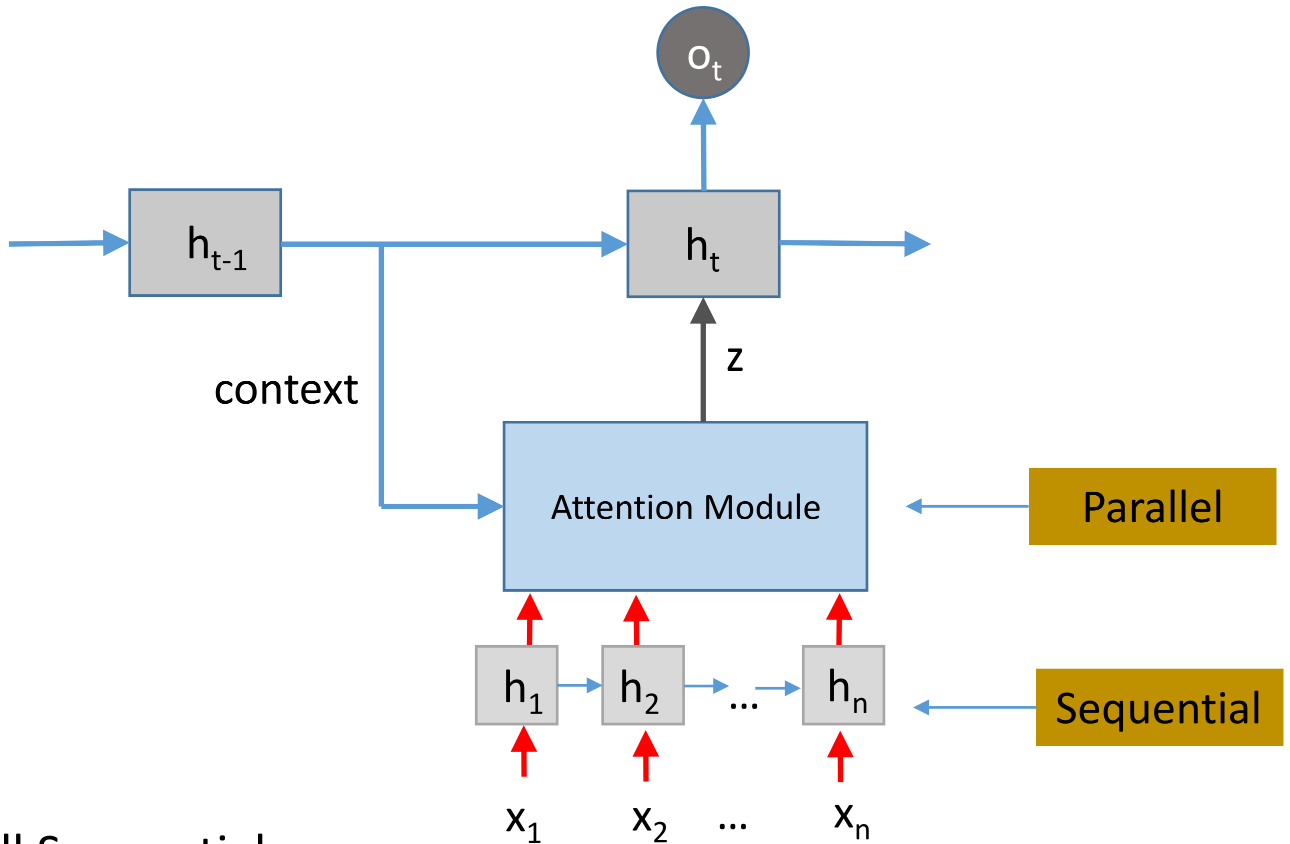
- Sequential computation prevents parallelization.



- Despite GRUs and LSTMs, RNNs still need attention mechanism to deal with long range dependencies.
- Can we parallelize the encoding process?

Problems with RNN

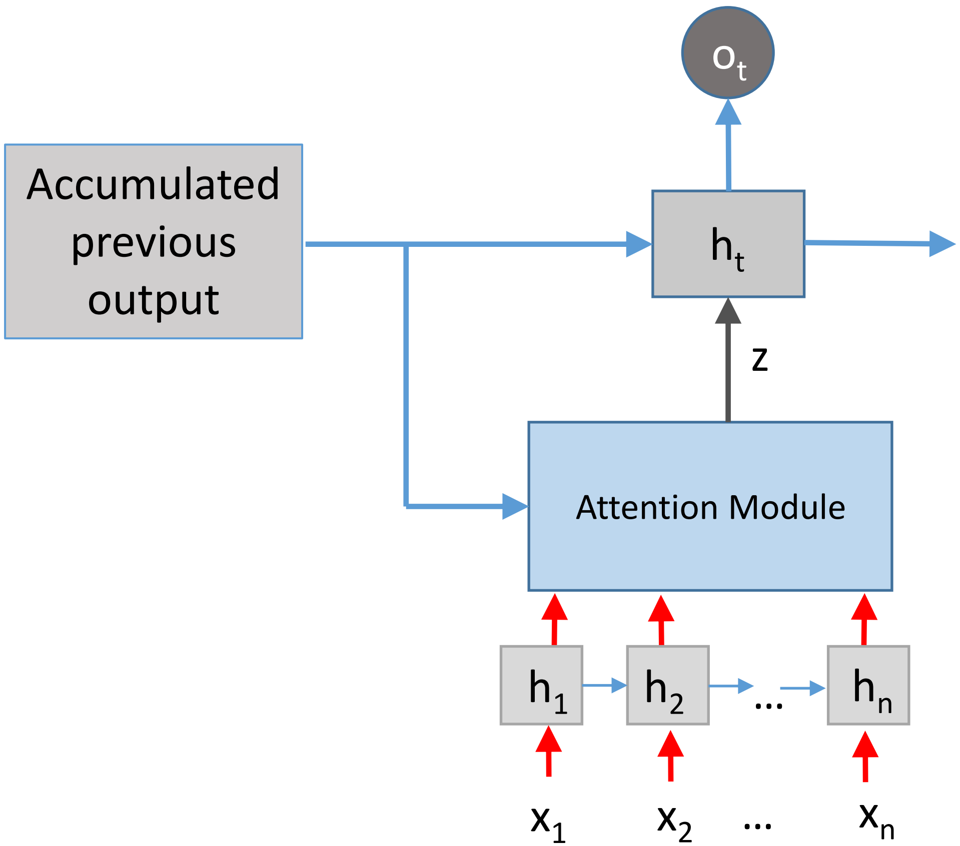
➢ Recap: Attention Module



- Still Sequential

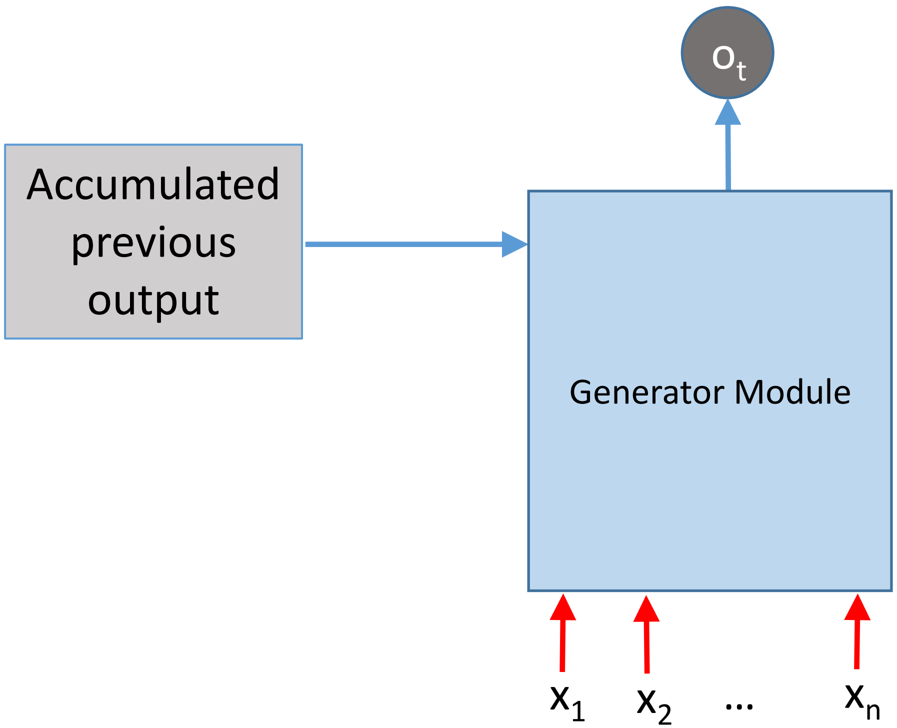
Problems with RNN

➢ **Another Viewpoint of Naïve Attention Module**



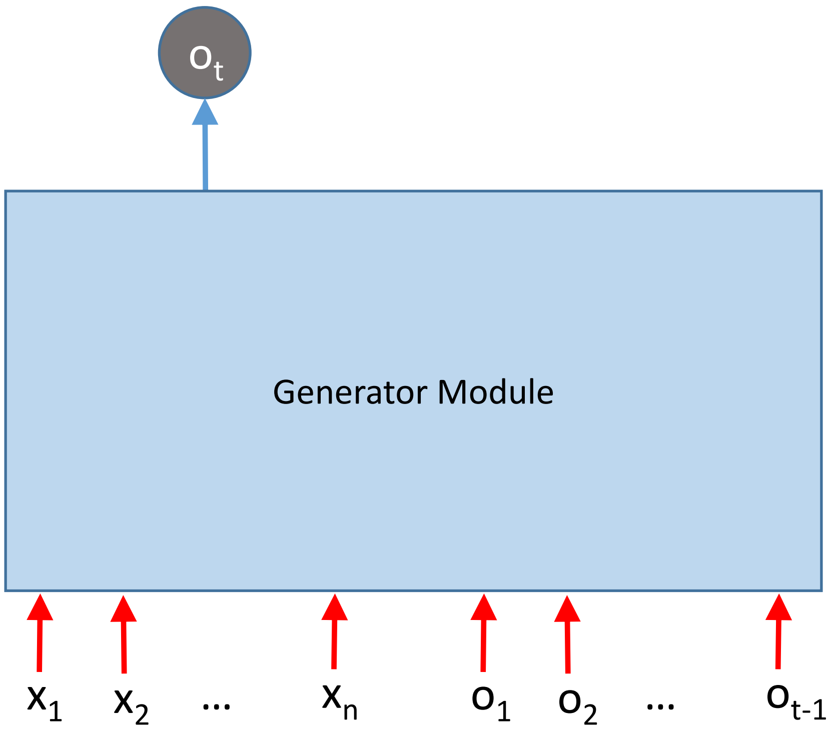
Problems with RNN

➢ **Another Viewpoint of Naïve Attention Module**



Problems with RNN

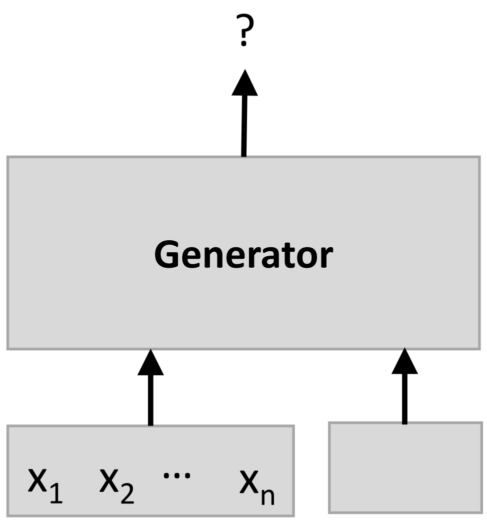
- Another Viewpoint of Naïve Attention Module



Problems with RNN

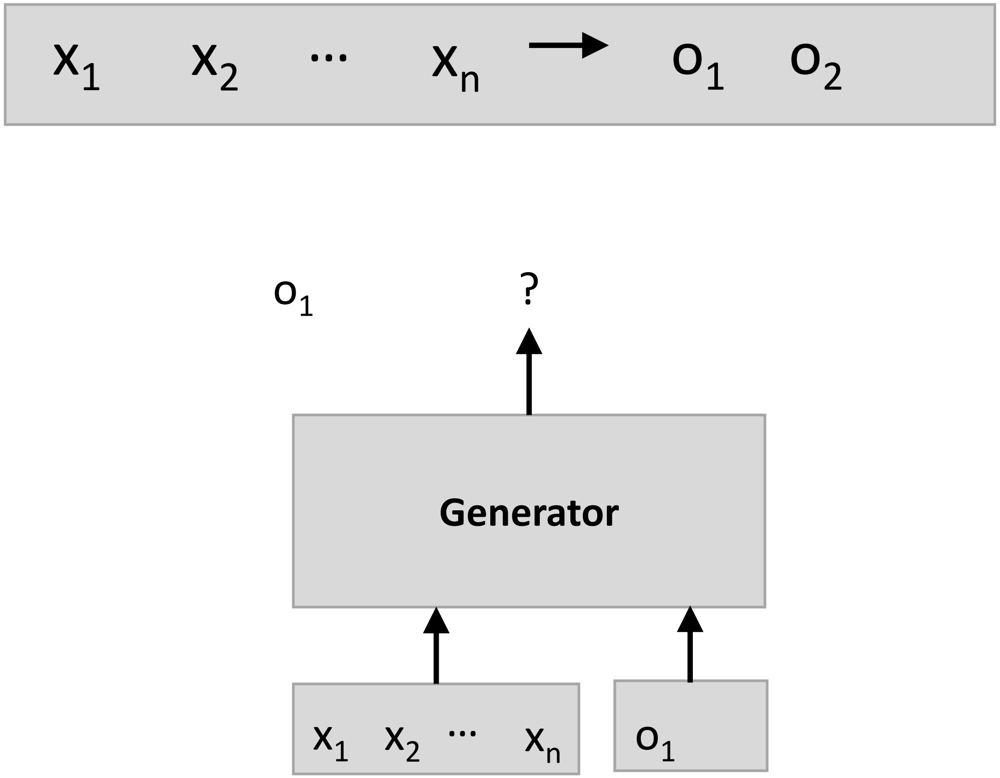
- **Without Sequential Encoder & Decoder**

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



Problems with RNN

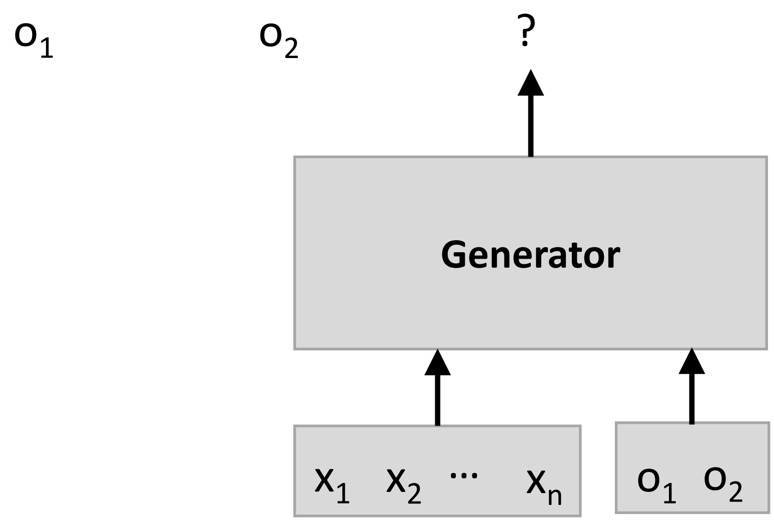
➢ **Without Sequential Encoder & Decoder**



Problems with RNN

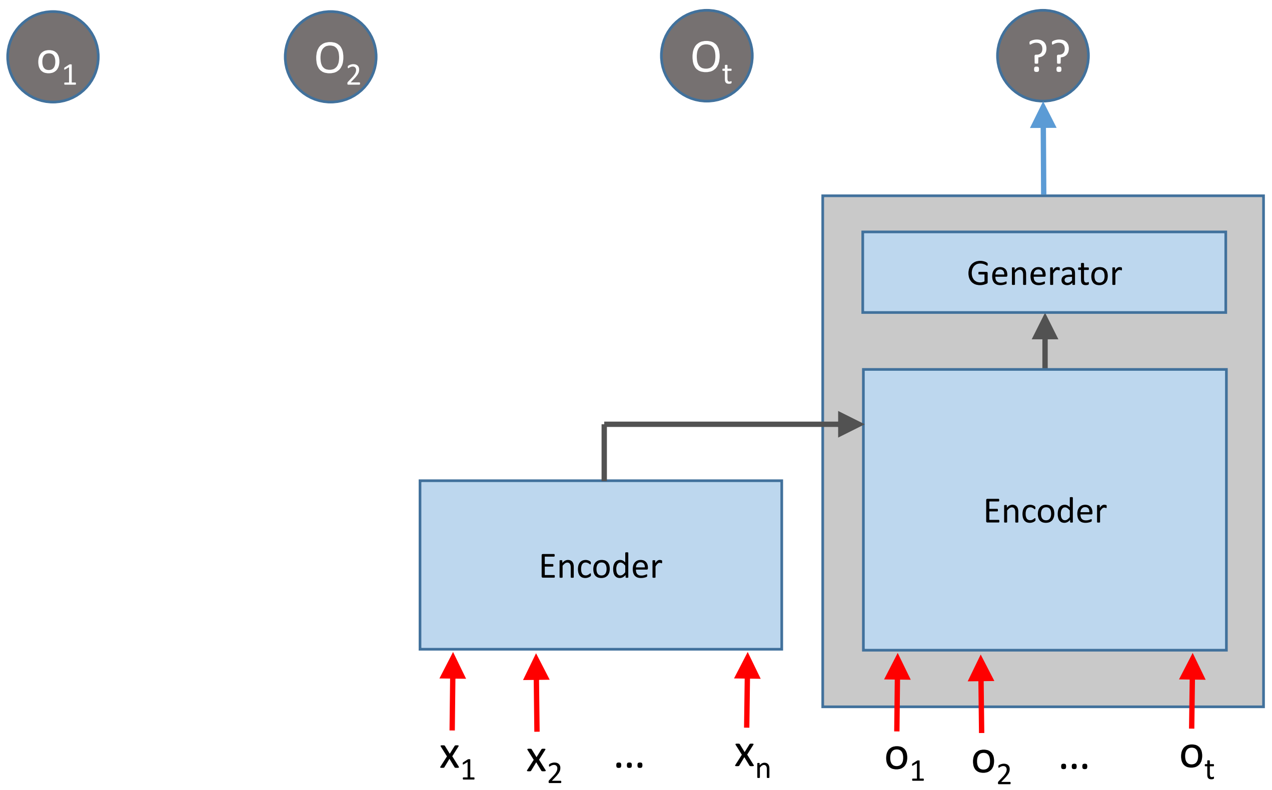
- **Without Sequential Encoder & Decoder**

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



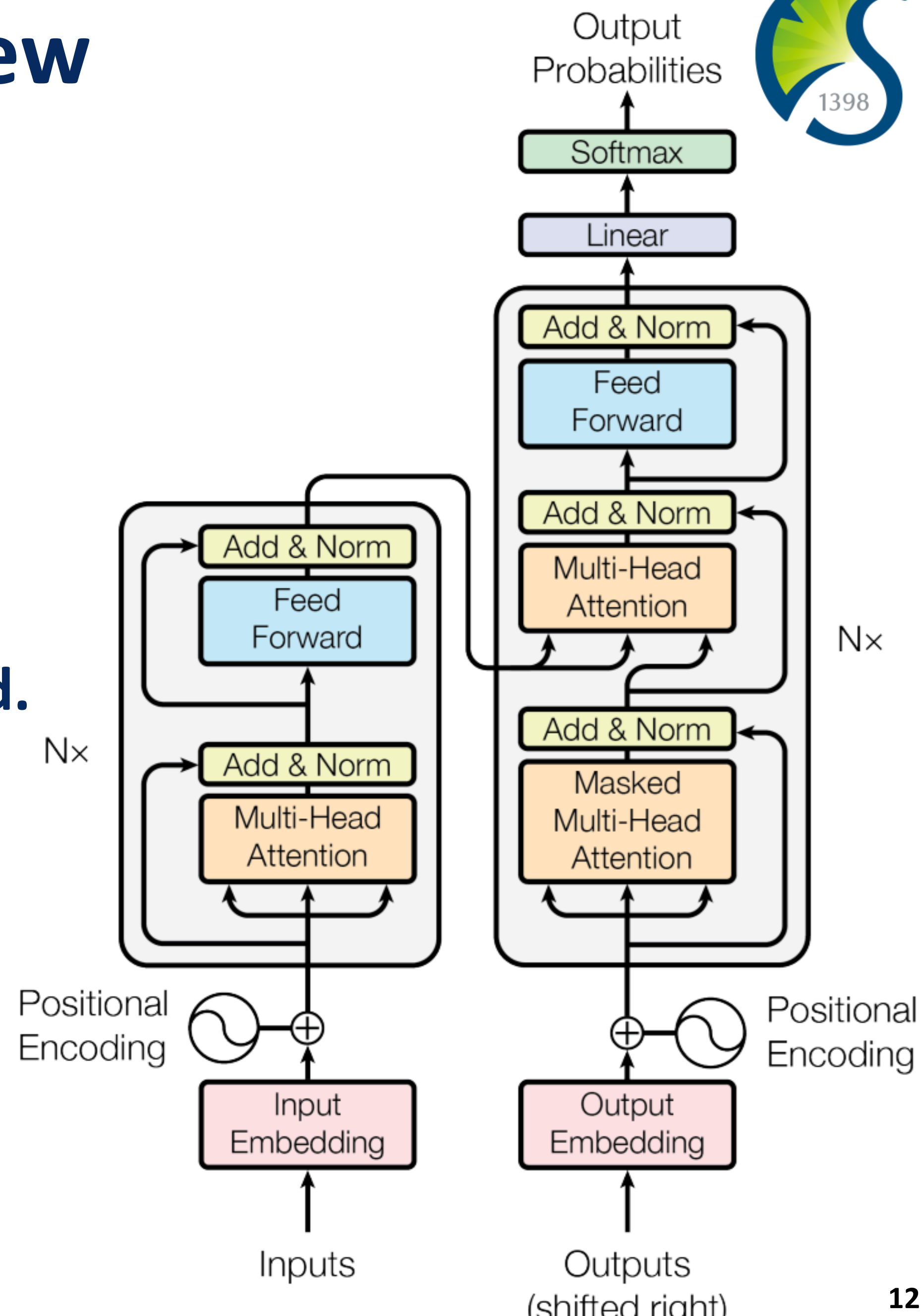
Problems with RNN

➢ **Sequential computation prevents parallelization.**

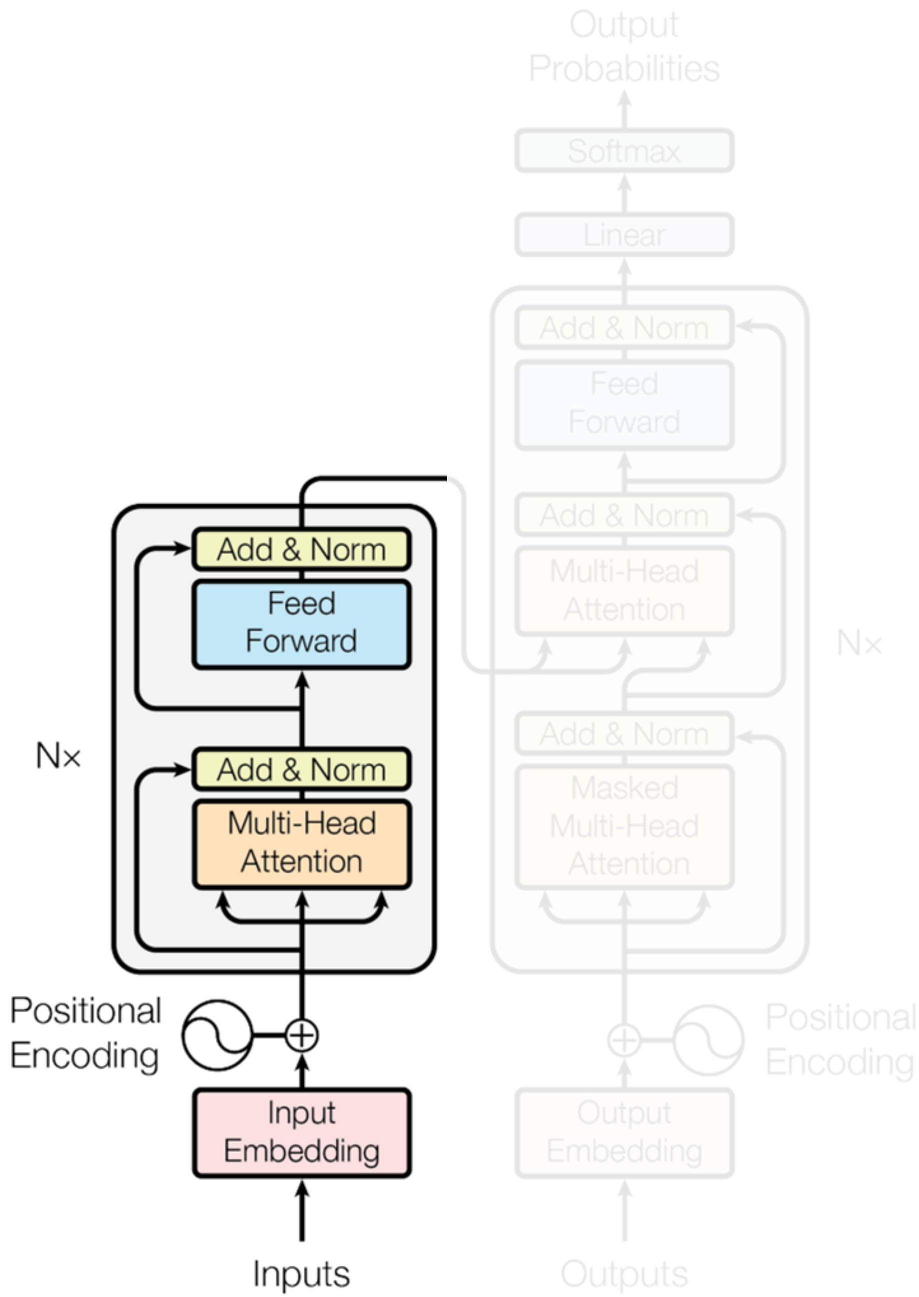


Transformer Overview

- Encoder-Decoder approach
- Task: machine translation with parallel corpus
- Predict each translated word.
- Final cost/error function is standard cross-entropy error on top of a softmax classifier.



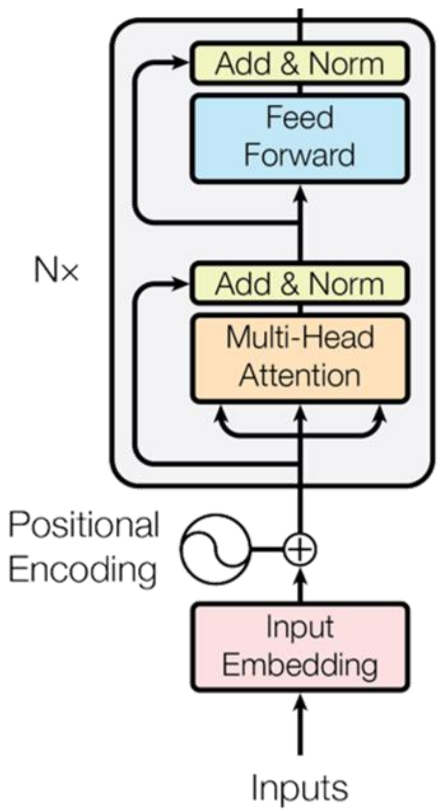
Encoder

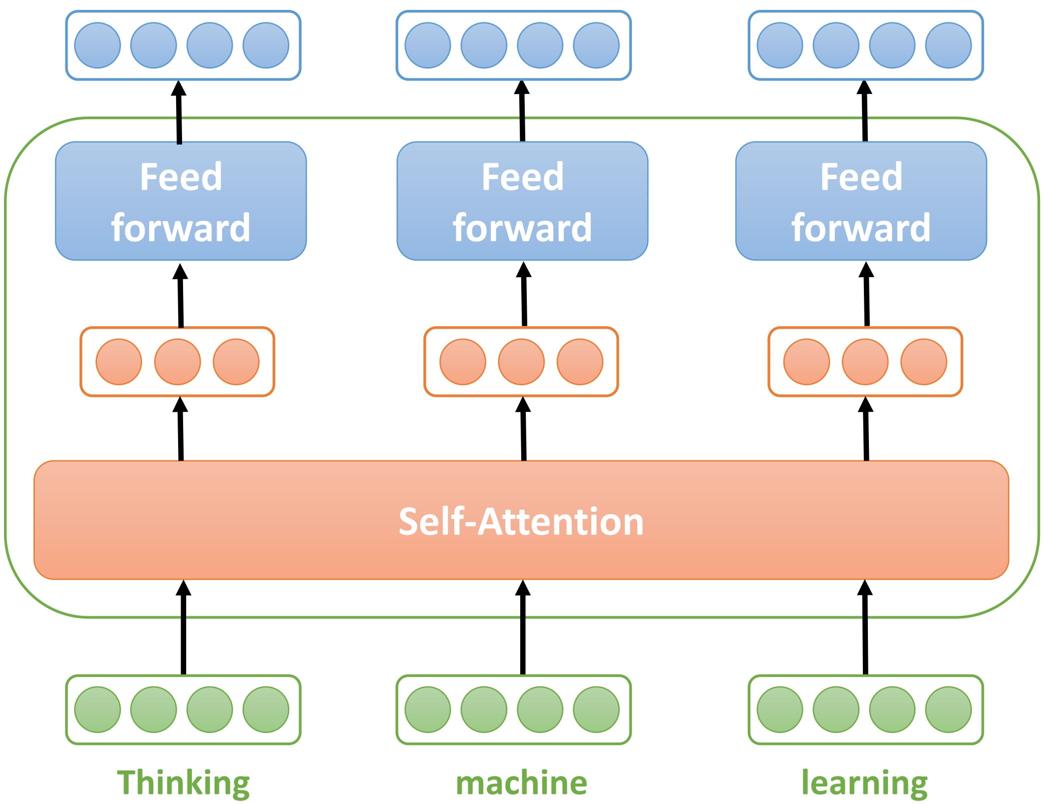


Output
Probabilities
Softmax
Linear
Add & Norm
Feed
Forward
Add & Norm
Multi-Head
Attention
Add & Norm
Masked
Multi-Head
Attention
Nx
Add & Norm
Feed
Forward
Add & Norm
Multi-Head
Attention
Nx
Positional
Encoding
Input
Embedding
Inputs
Positional
Encoding
Output
Embedding
Outputs

Encoder Internals

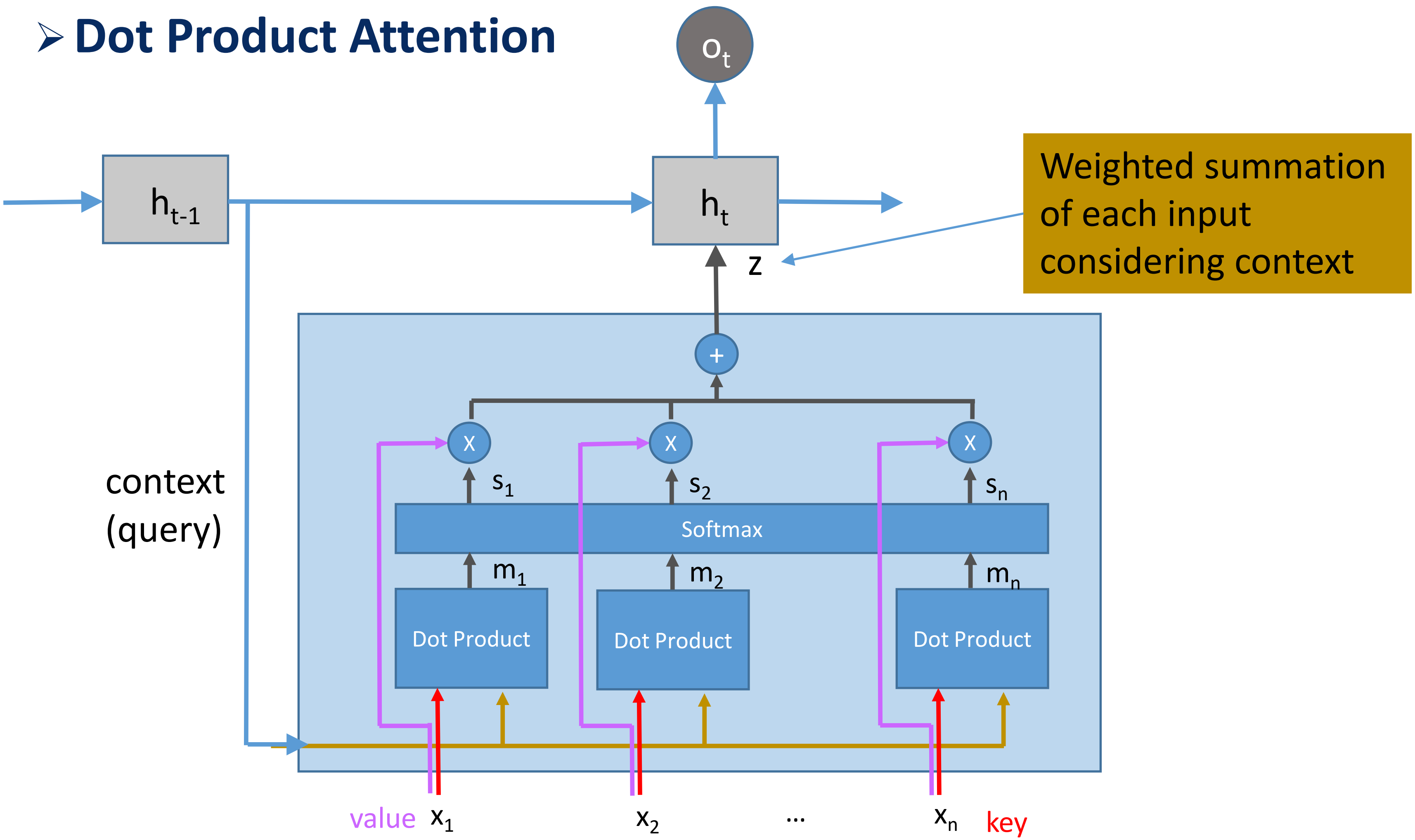
- After embedding the words in the input sentence, each of them flows through the two layers of the encoder.





Recap: Attention Mechanism

- **Dot Product Attention**



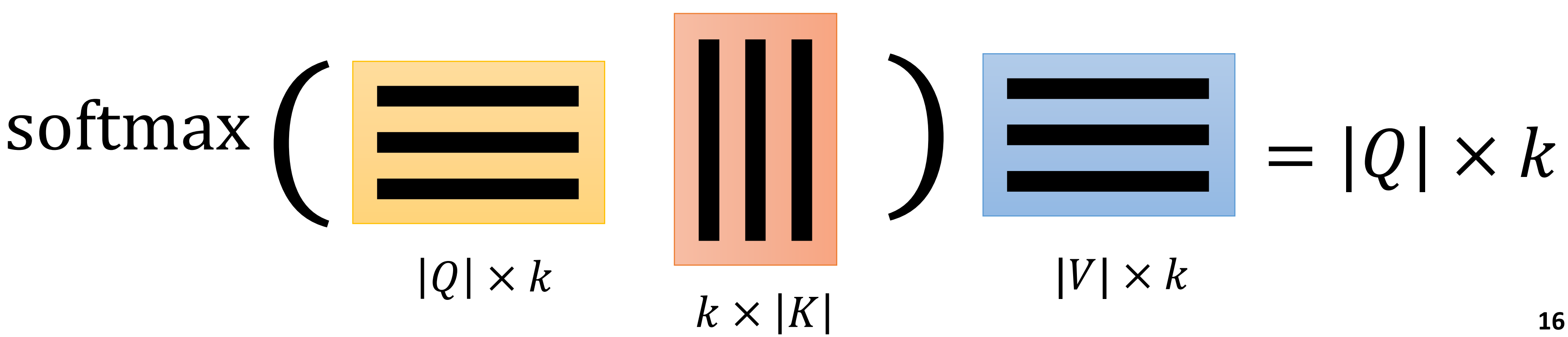
Recap: Attention Mechanism

- Using multiple queries, we stack them into a matrix Q.

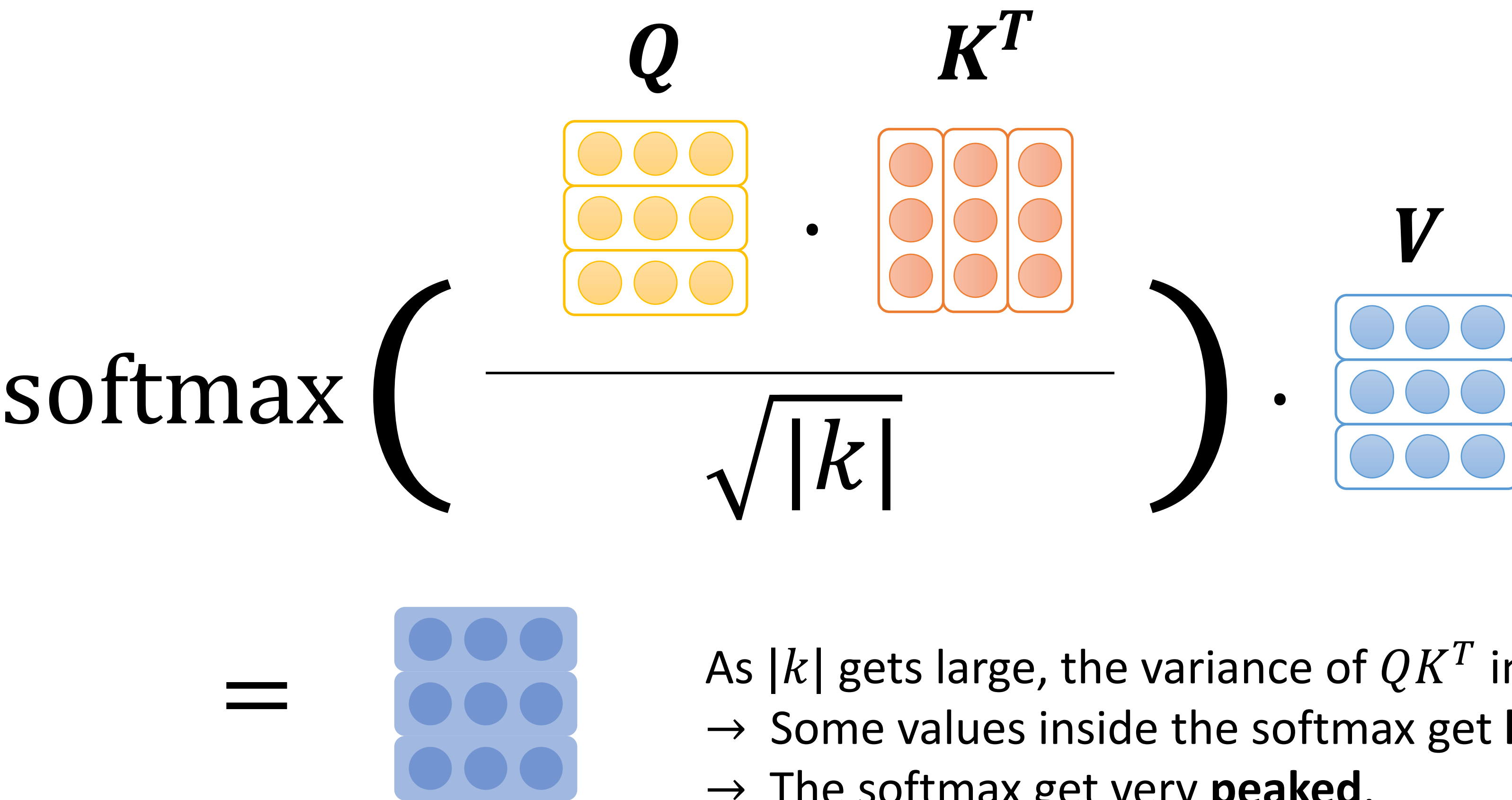
$$A(q, K, V) = \sum_i \frac{e^{q \cdot k_i}}{\sum_j e^{q \cdot k_j}} v_i$$

- It becomes

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



Recap: Attention Mechanism

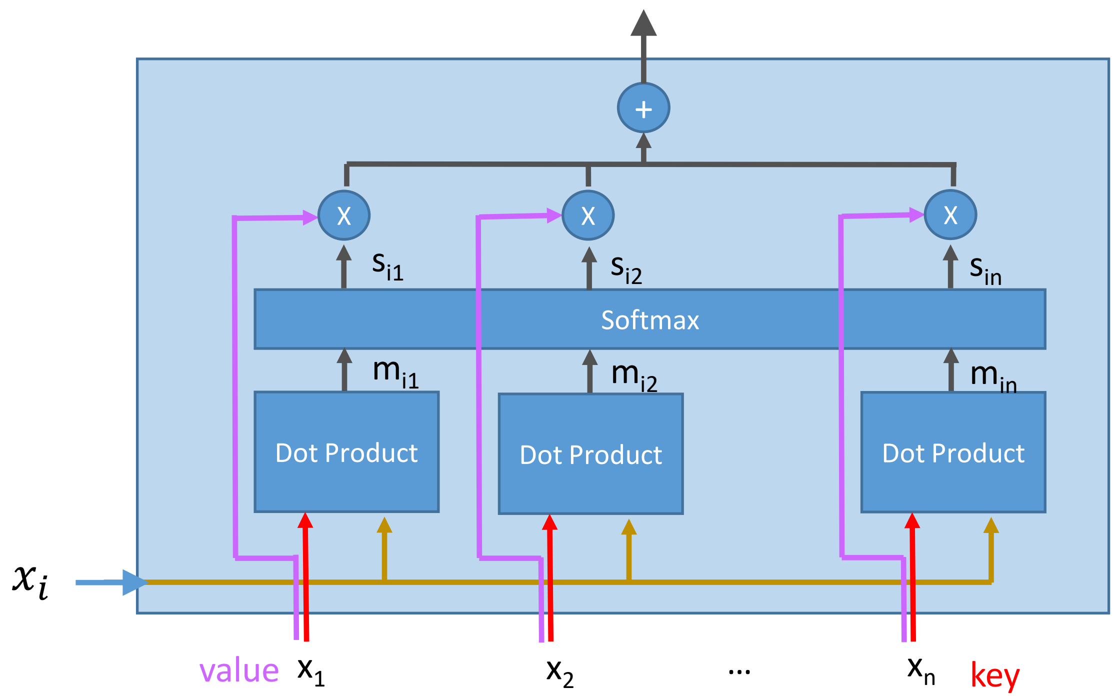


As $|k|$ gets large, the variance of QK^T increases.
→ Some values inside the softmax get **large**.
→ The softmax get very **peaked**.
→ Its gradient gets **smaller**.

Self-Attention

- **Definition:** $A(x_i, X, X)$

$$s_{i1}x_1 + s_{i2}x_2 + \cdots + s_{in}x_n$$



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

Self-Attention

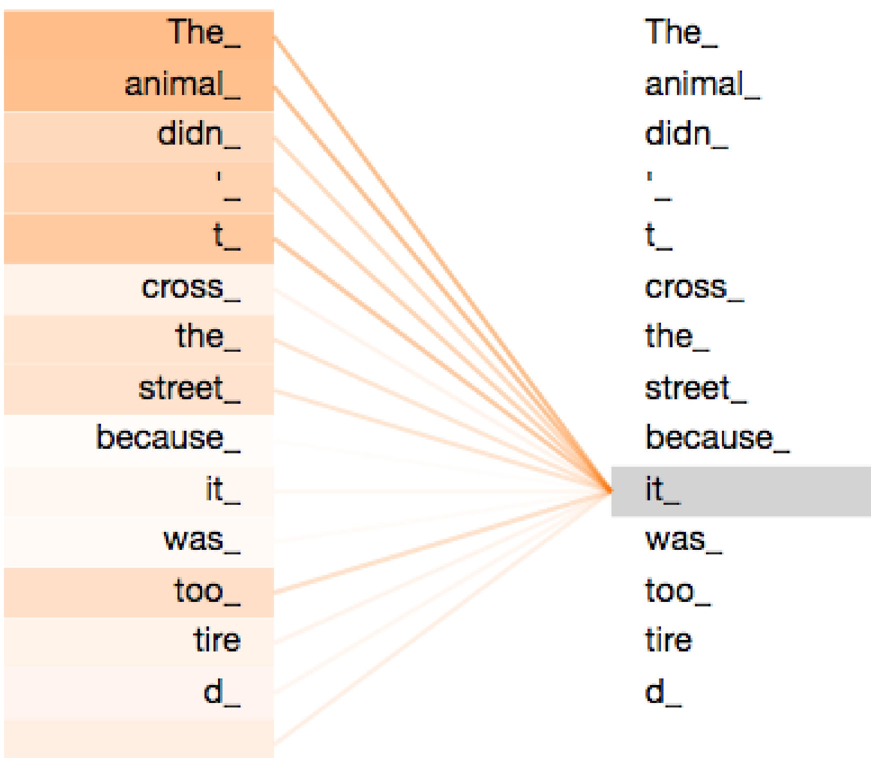
➢ **Definition:**

$$A(X,X,X)=\begin{pmatrix} s_{11}x_1+s_{12}x_2+\cdots+s_{1n}x_n \\ s_{21}x_1+s_{22}x_2+\cdots+s_{2n}x_n \\ \dots \\ s_{n1}x_1+s_{n2}x_2+\cdots+s_{nn}x_n \end{pmatrix}=X'$$

Self-Attention

- ➢ **Learning long-range dependencies in the network**
- ➢ **Effective for parallelization for efficient computation**

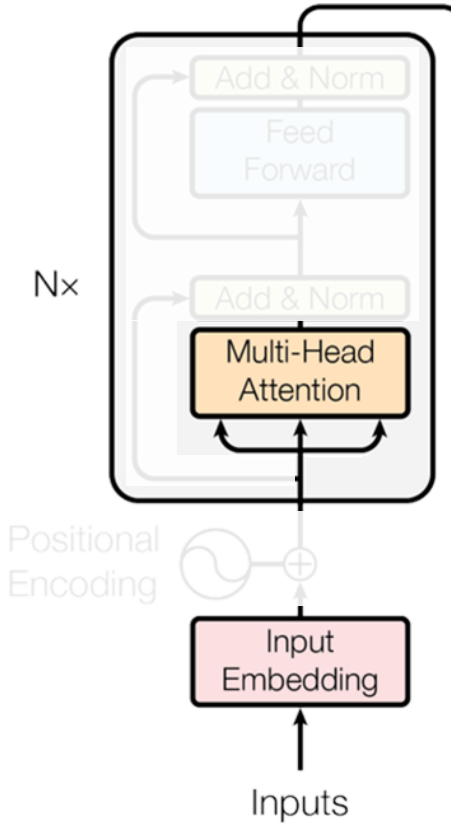
The animal didn't cross the street because ***it*** was too tired



When the model is processing the word "*it*", **self-attention** allows it to associate "*it*" with "*animal*".

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'?

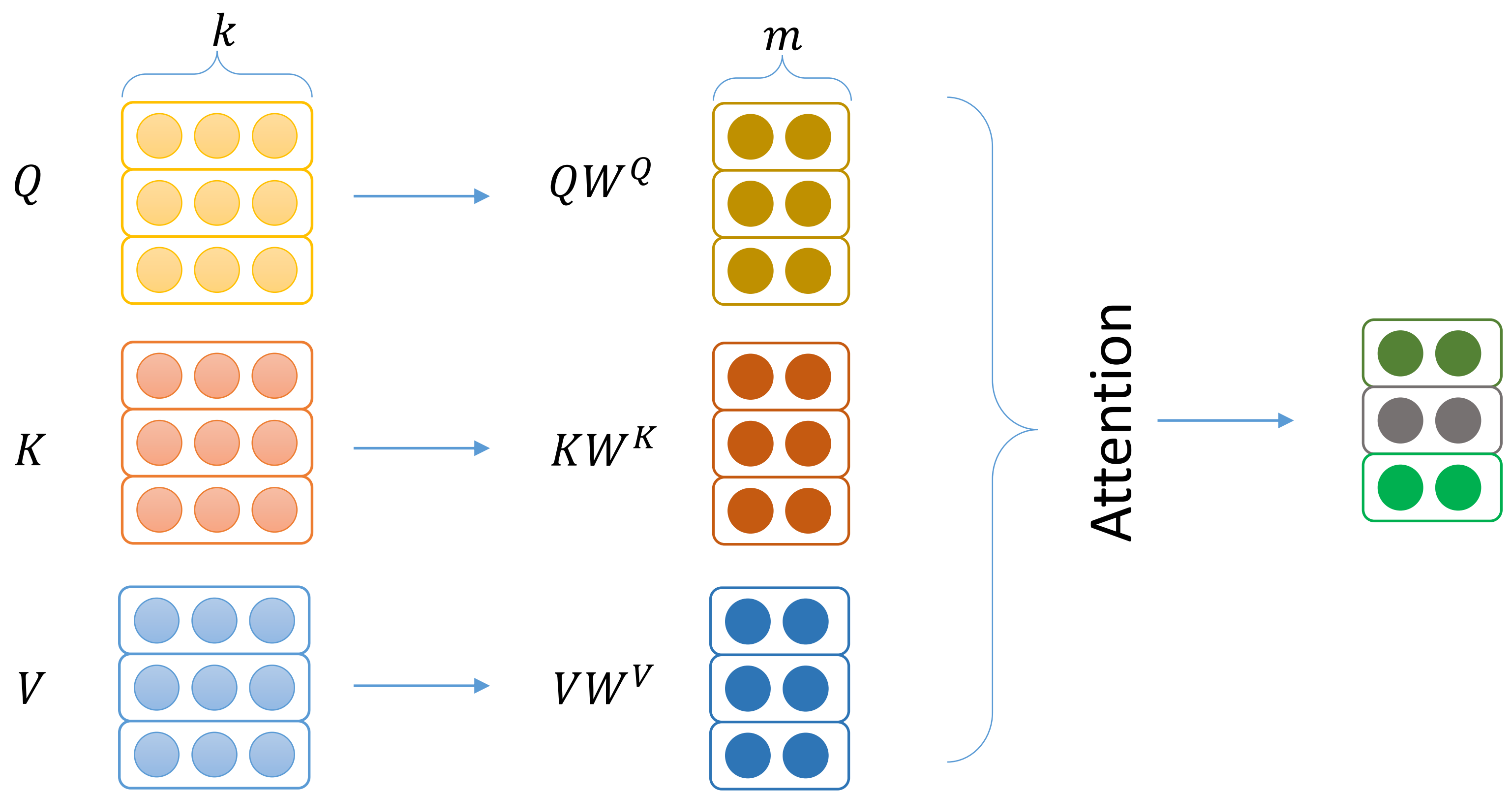
- Linear Transformed: $W^Q, W^K, W^V \ (= k \times m)$

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

Multi-headed Attention

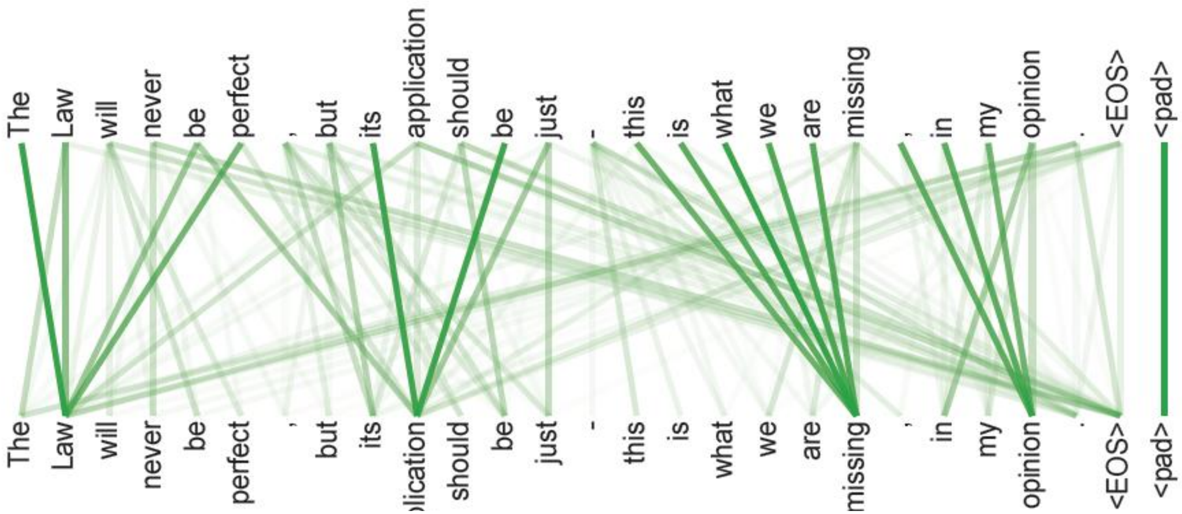
➢ **Head: A viewpoint of similarity**

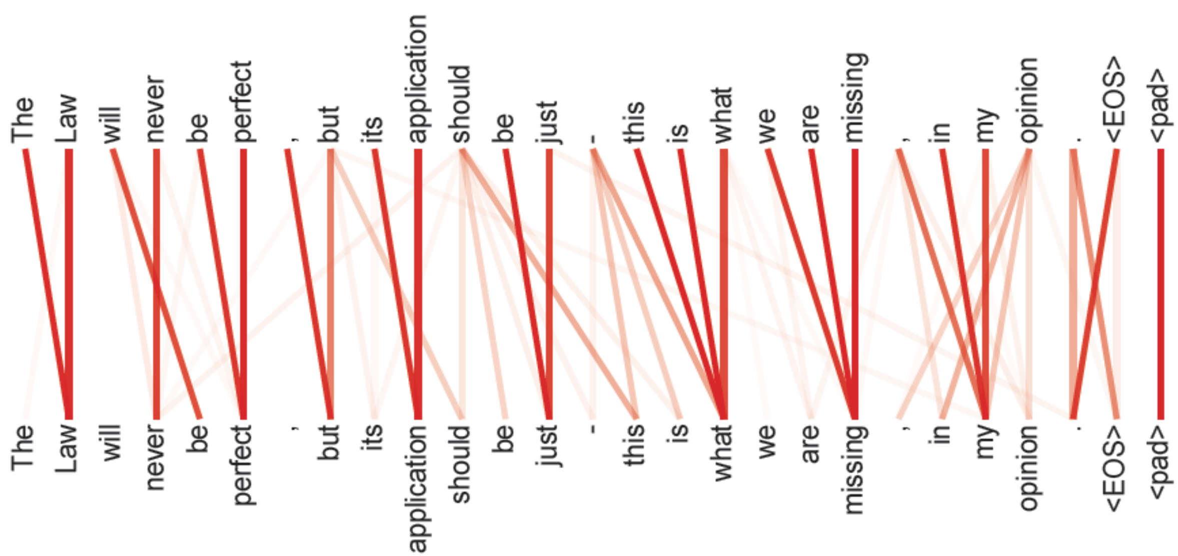
$$A(QW^Q, KW^K, VW^V)$$



Multi-headed Attention

- **Head: A viewpoint of similarity**





The encoder self-attention at layer 5 and 6.

The heads clearly learned to perform different tasks.

Multi-headed Attention

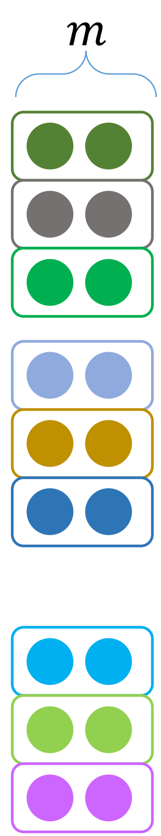
- Let’s use multiple heads to capture various similiarities

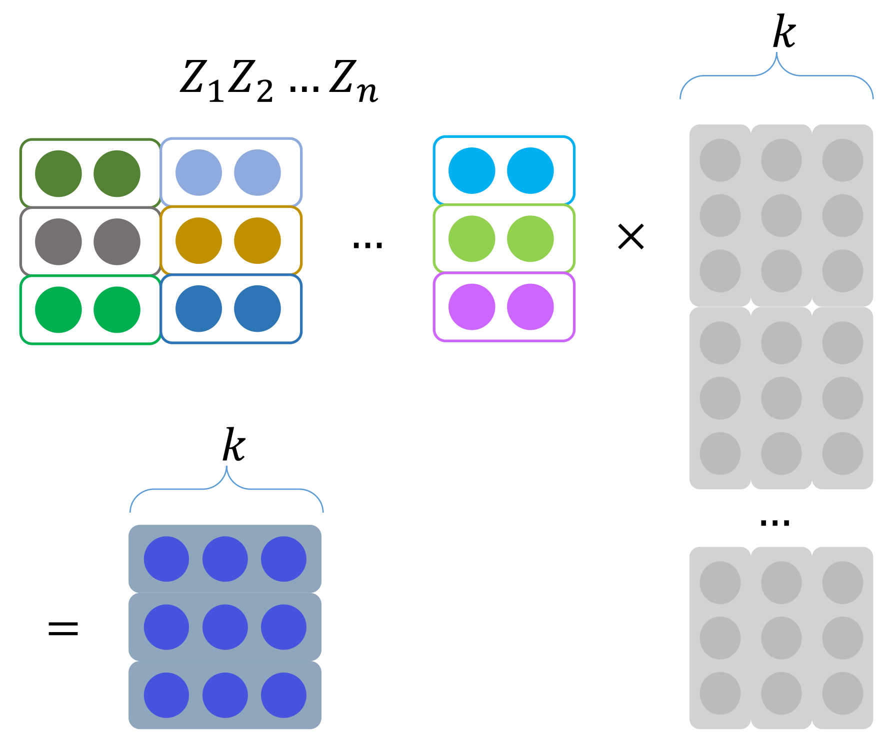
$$A\left(QW_1^Q, KW_1^K, VW_1^V\right) = Z_1$$

$$A\left(QW_2^Q, KW_2^K, VW_2^V\right) = Z_2$$

...

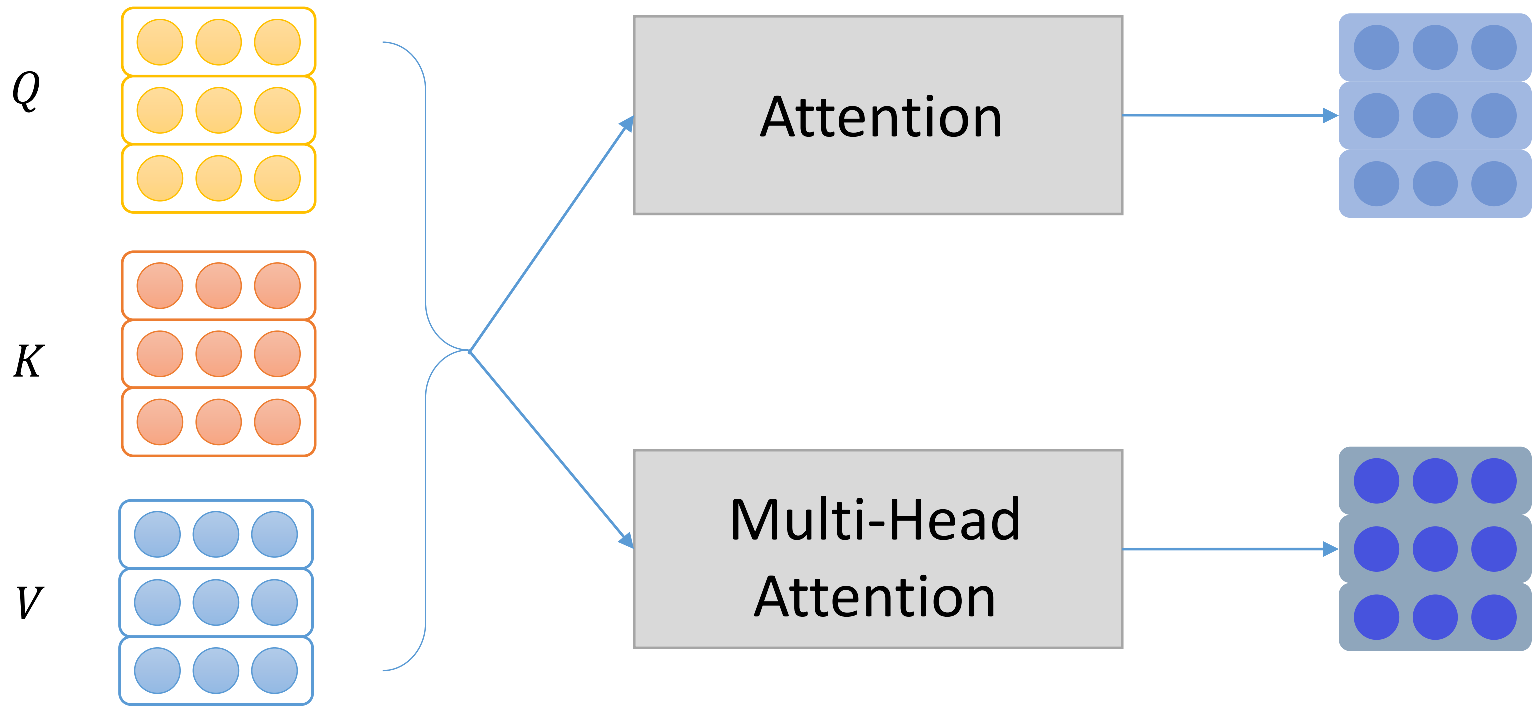
$$A\left(QW_n^Q, KW_n^K, VW_n^V\right) = Z_n$$





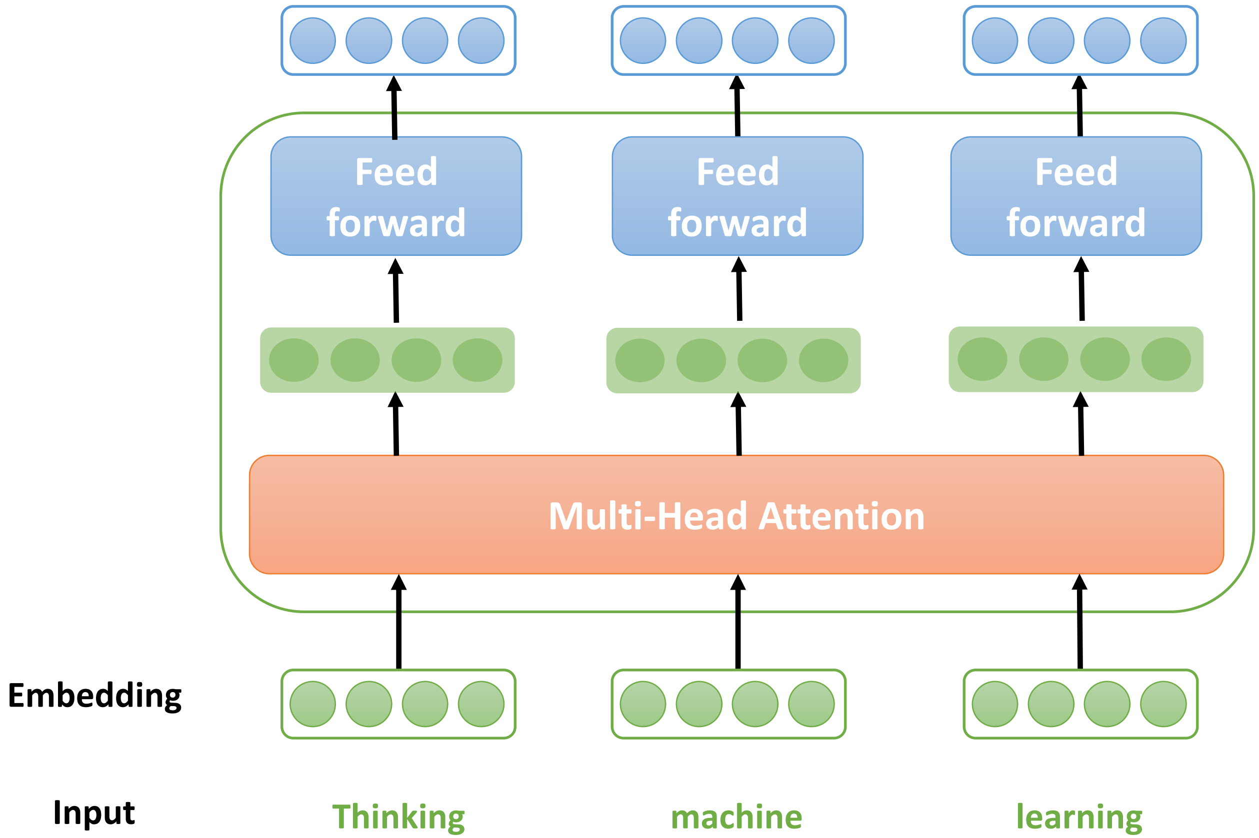
Multi-headed Attention

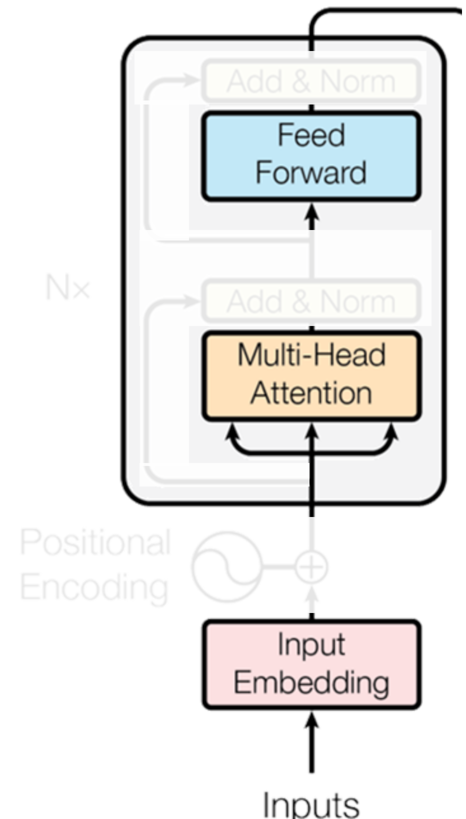
➢ **Let's use multiple heads to capture various similiarities**



Feed Forward

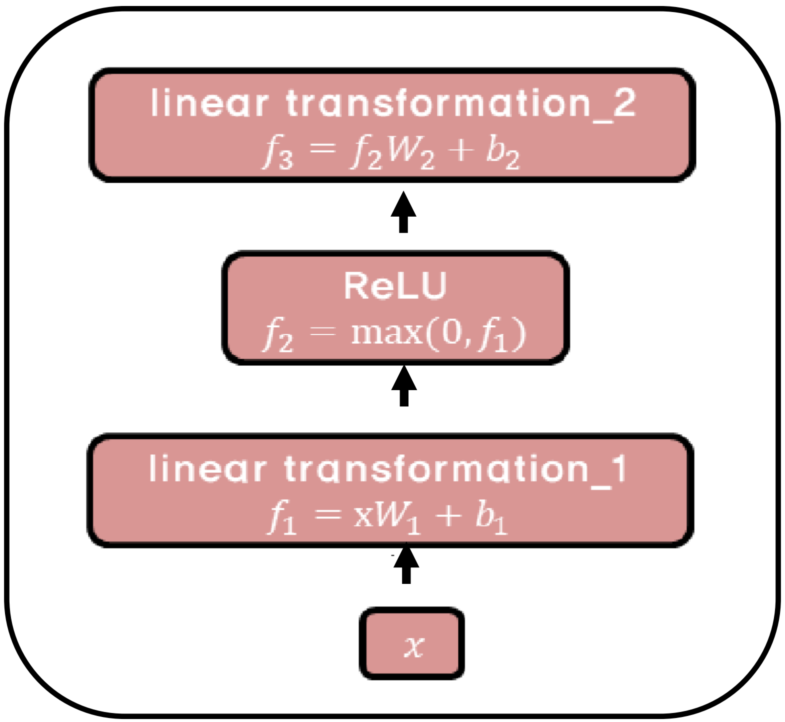
- ➢ **Point-wise Feed Forward**

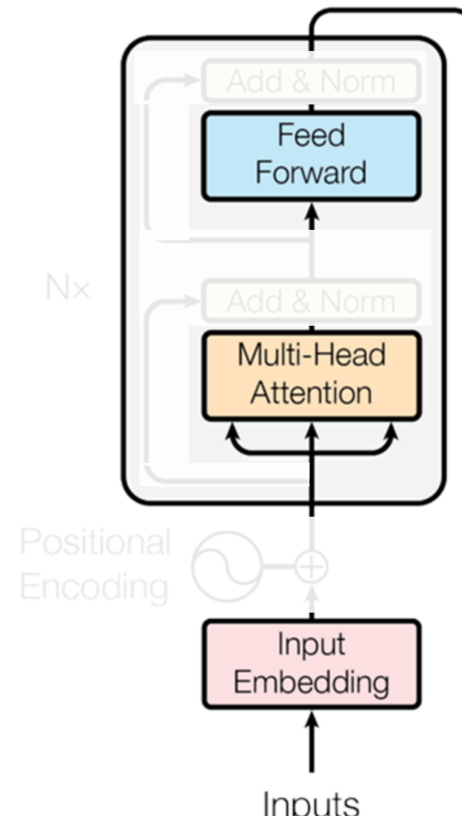




Feed Forward

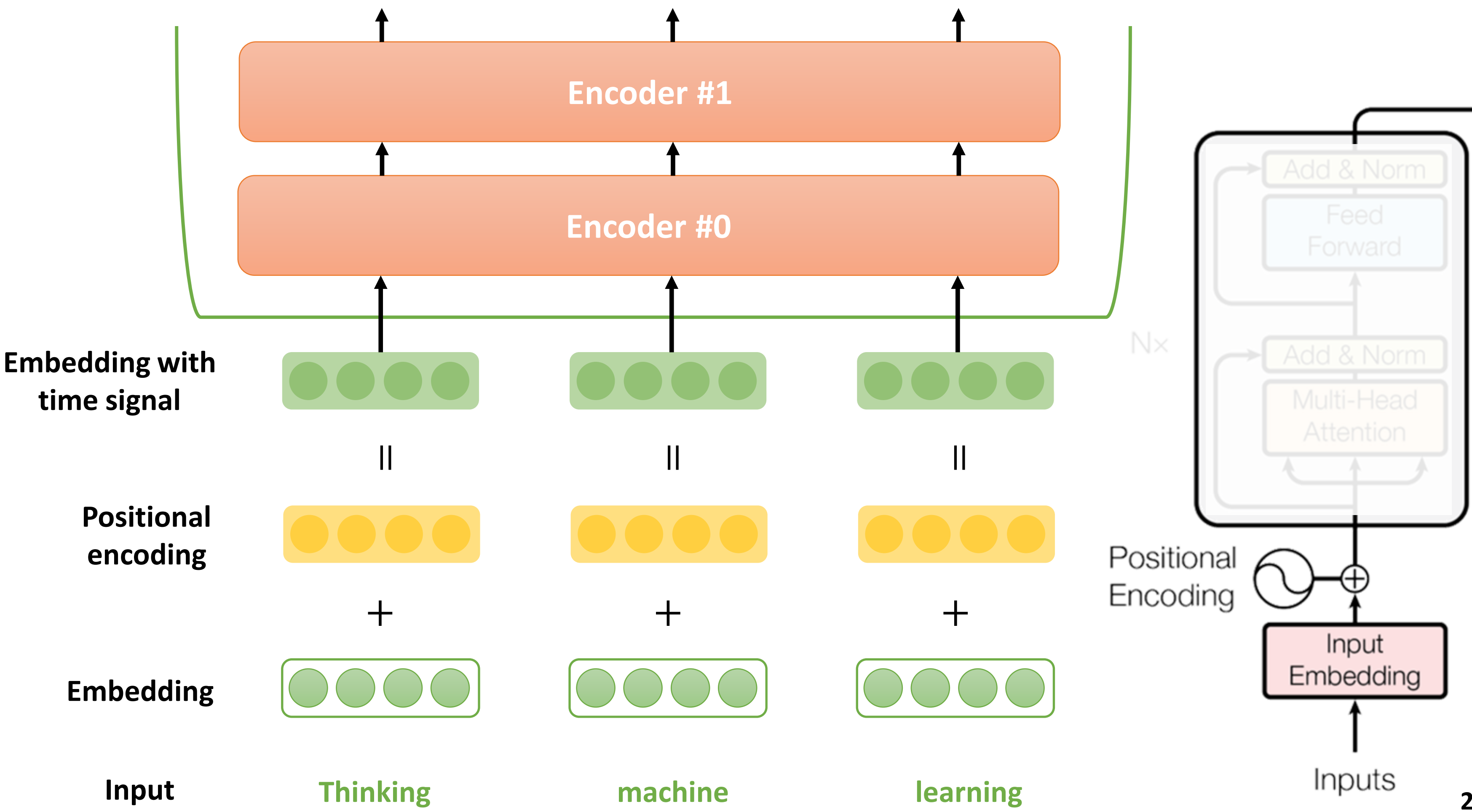
➢ **Point-wise Feed Forward**





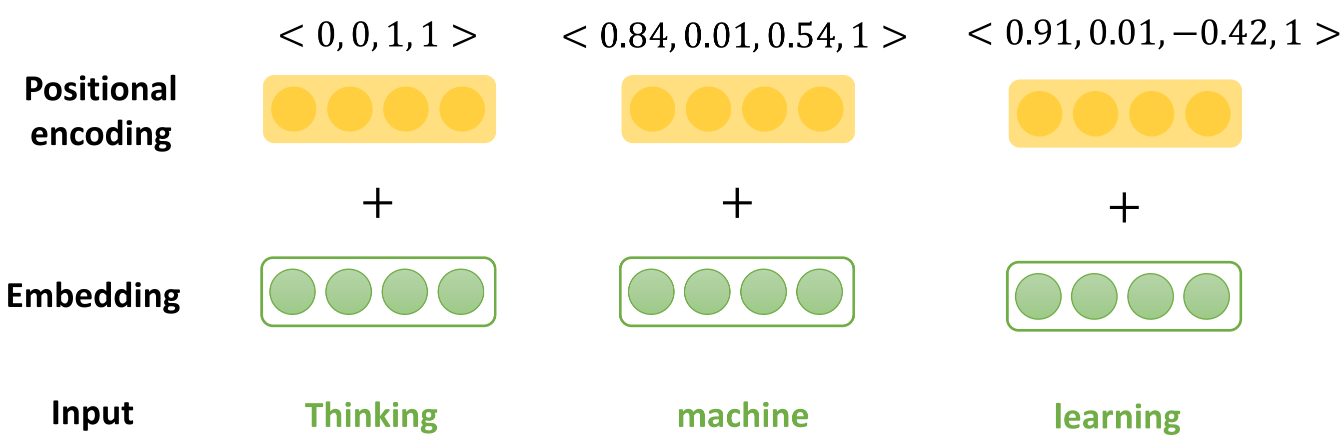
Positional Encoding

- ➢ **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.

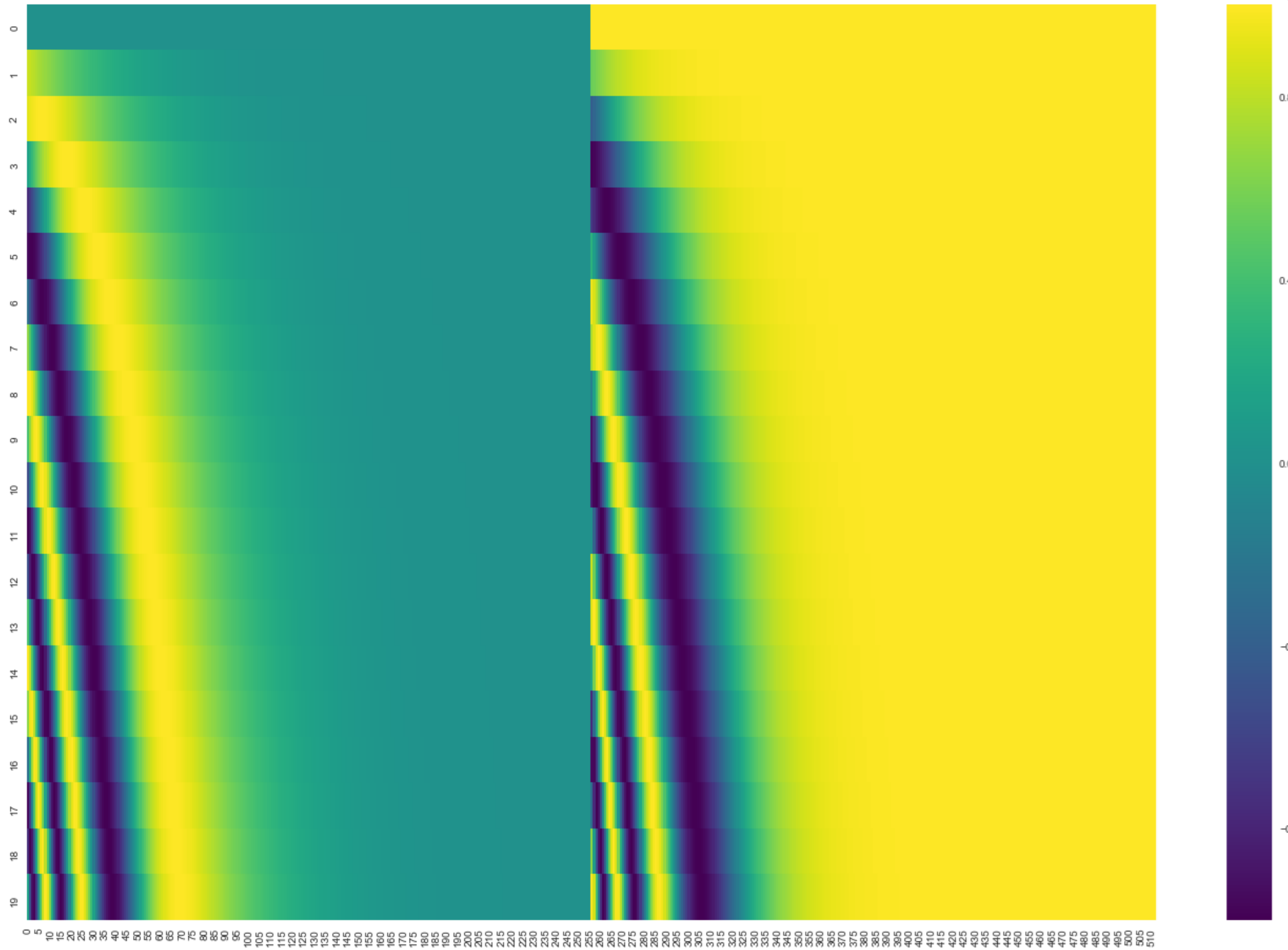


$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{model}})$$

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

Positional Encoding

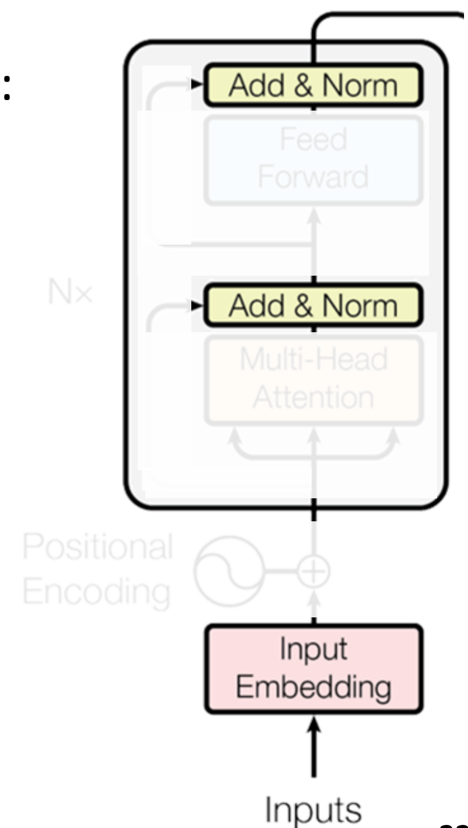
Each row corresponds to the a positional encoding of size 512.



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**:

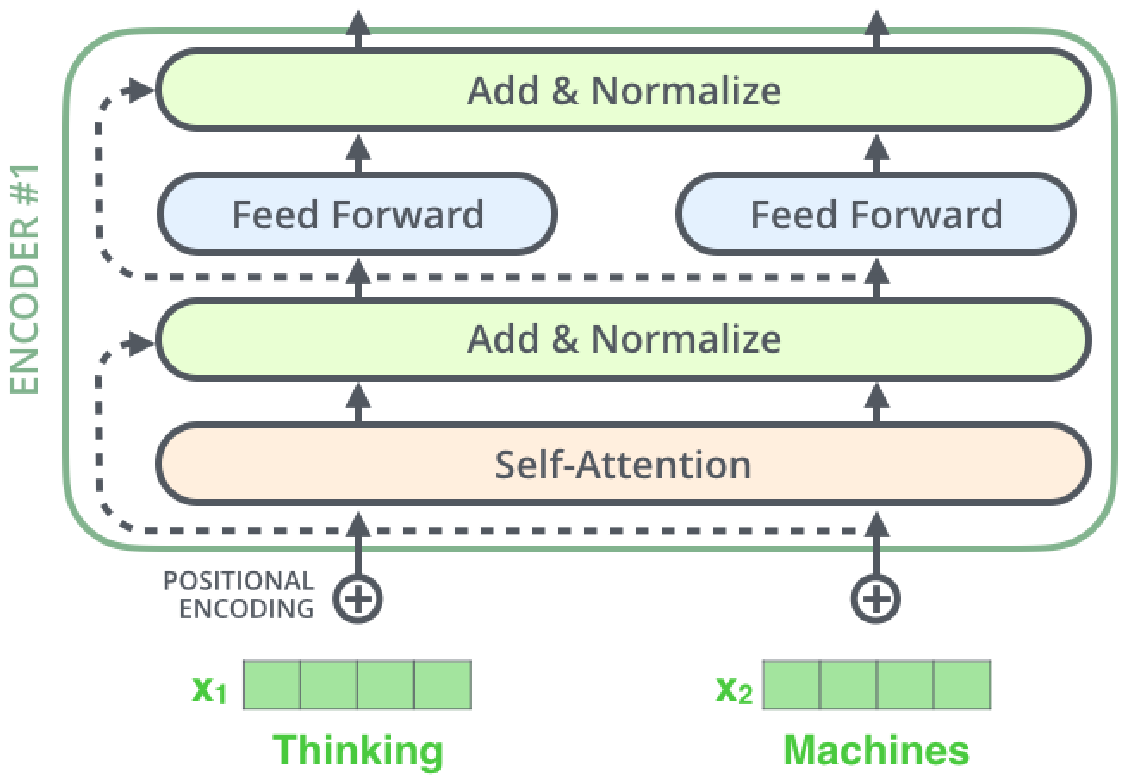
$$LayerNorm(X + Sublayer(X))$$

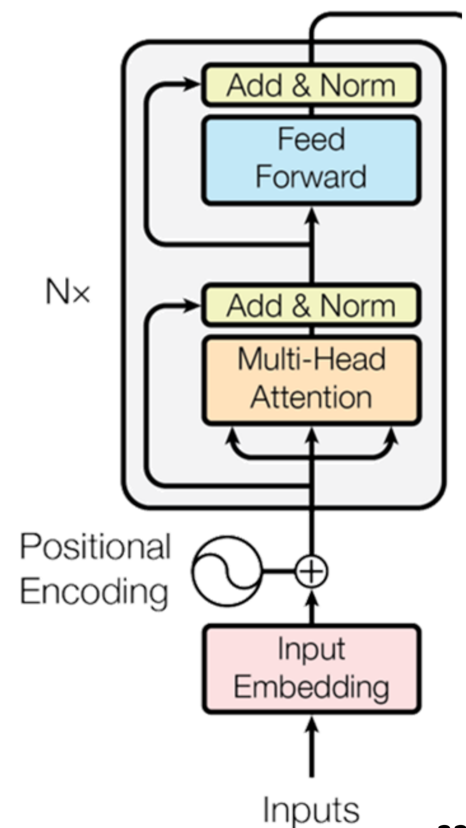


- **Layer normalization**
  - https://arxiv.org/pdf/1607.06450.pdf

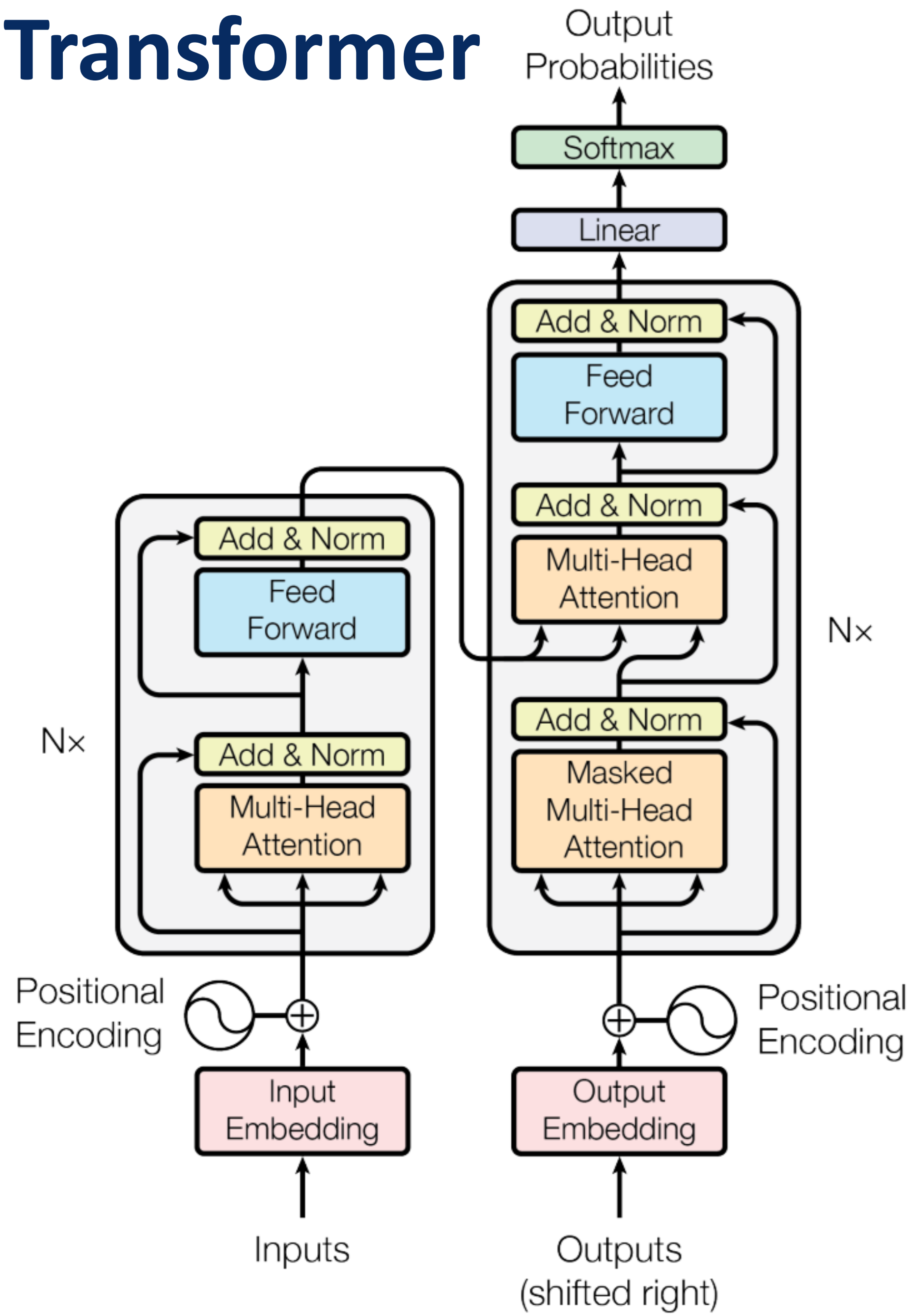
Residual Connection

- Each sublayer in each encoder has a residual connection.



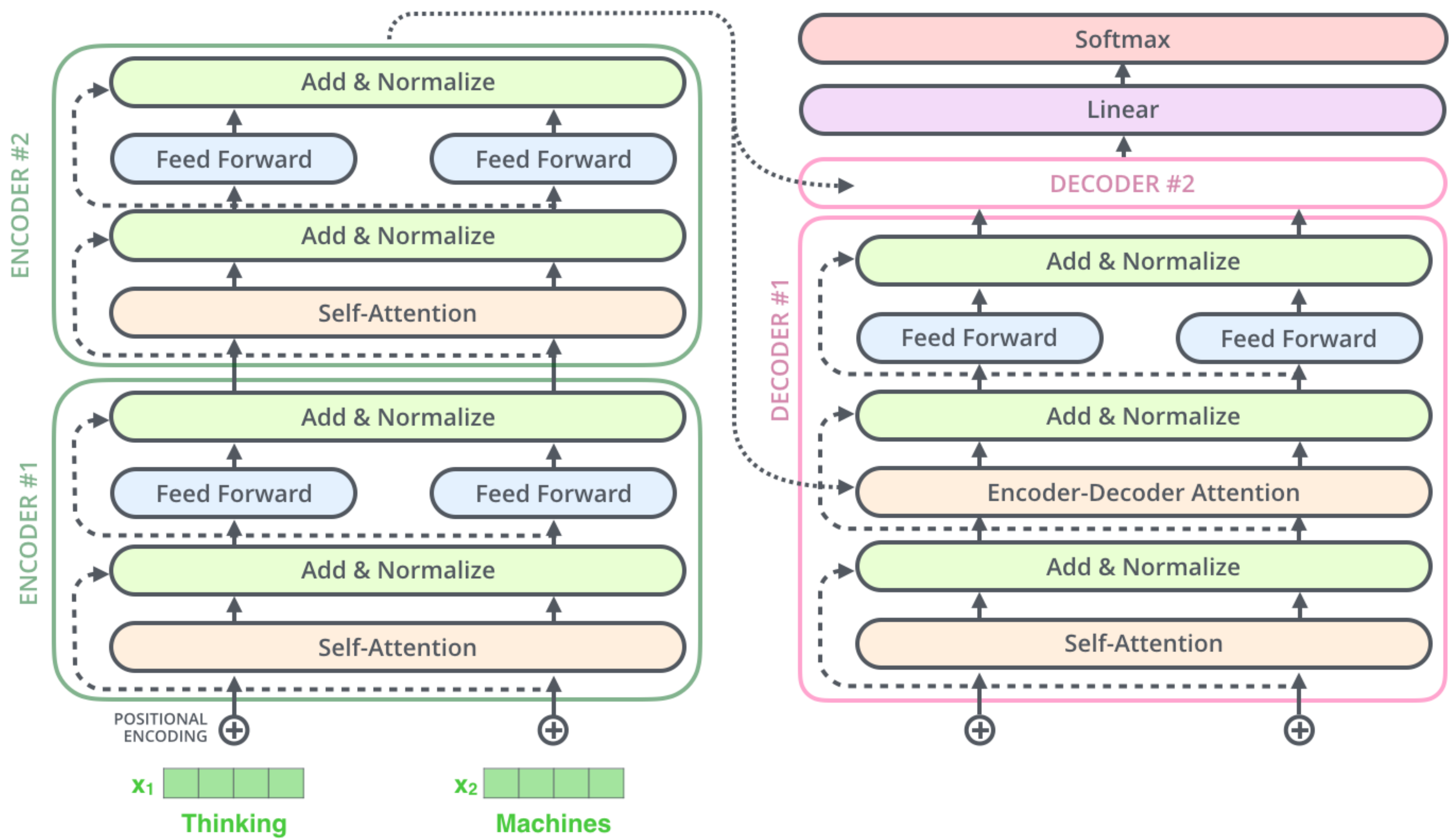


Simplified Transformer



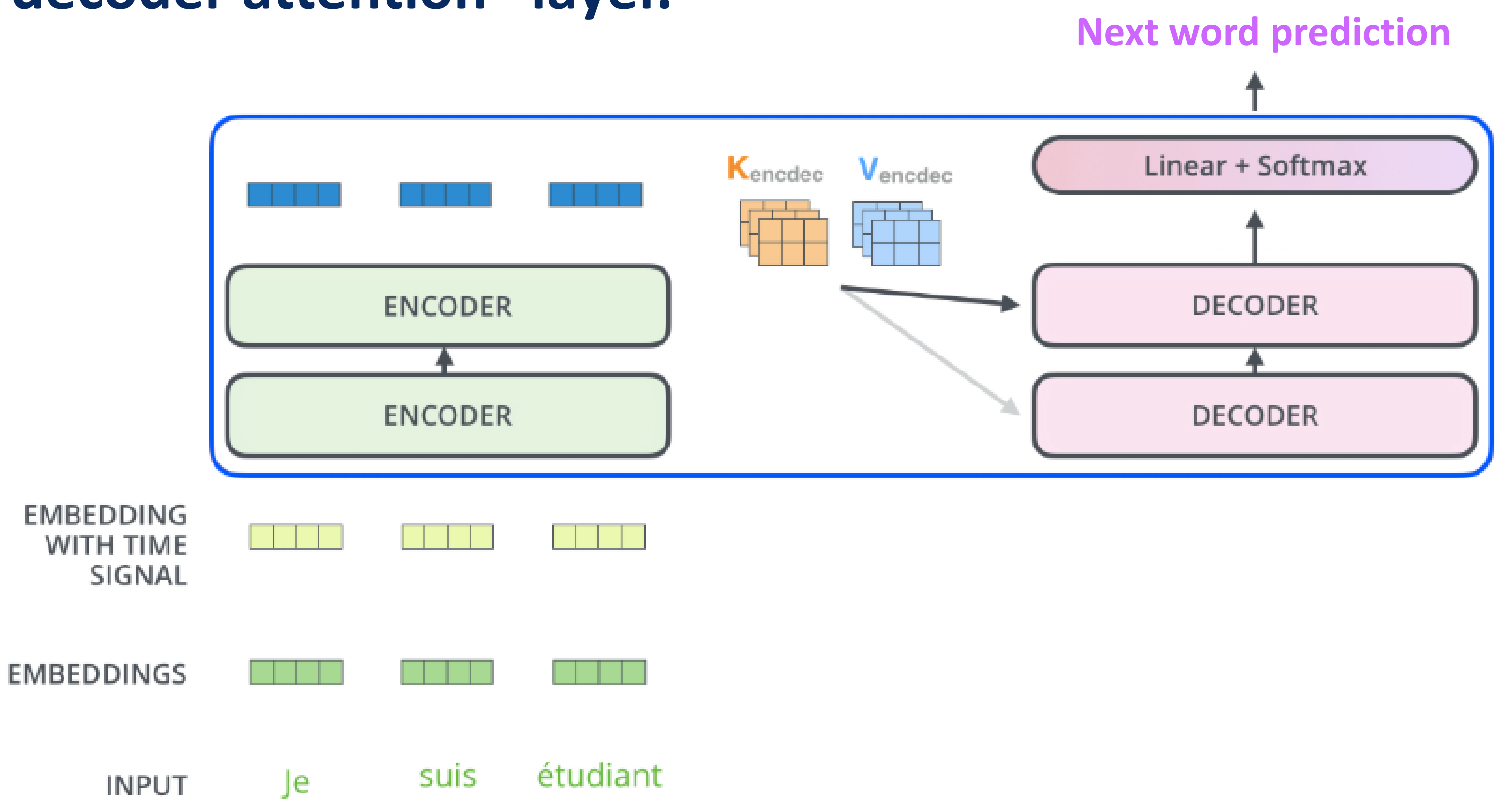
Simplified Transformer

- It consists of 2 stacked encoders and decoders.



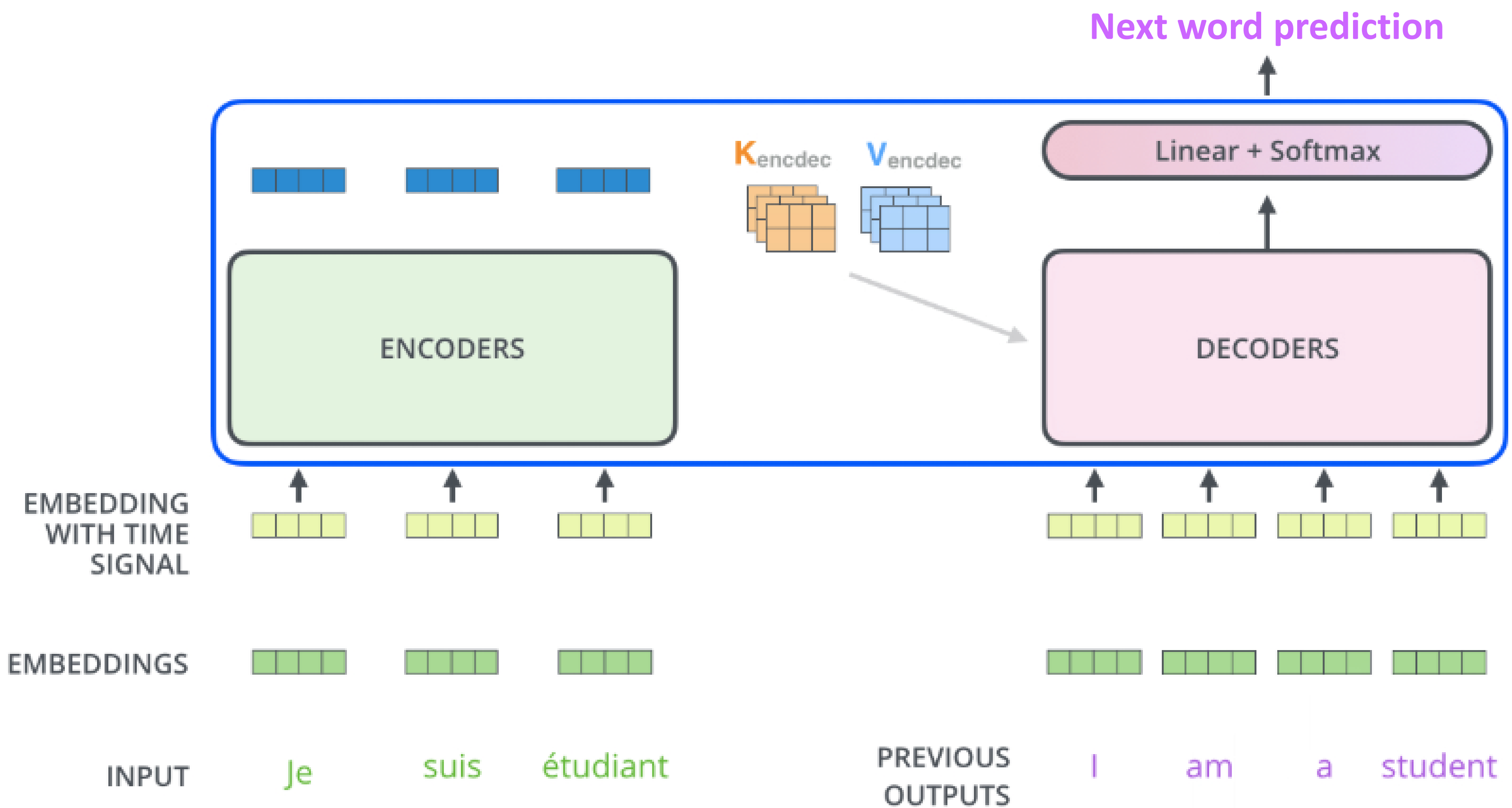
Decoder: Encoder-Decoder Attention

- The output of the top encoder is then transformed into a set of attention vectors K and V.
- These are to be used by each encoder in its "encoder-decoder attention" layer.



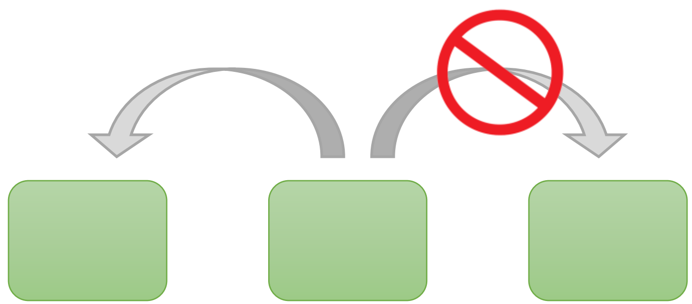
Decoder: Encoder-Decoder Attention

- The self-attention layer in the decoder is only allowed to attend to earlier positions in the output sequence.



Decoder: Encoder-Decoder Attention

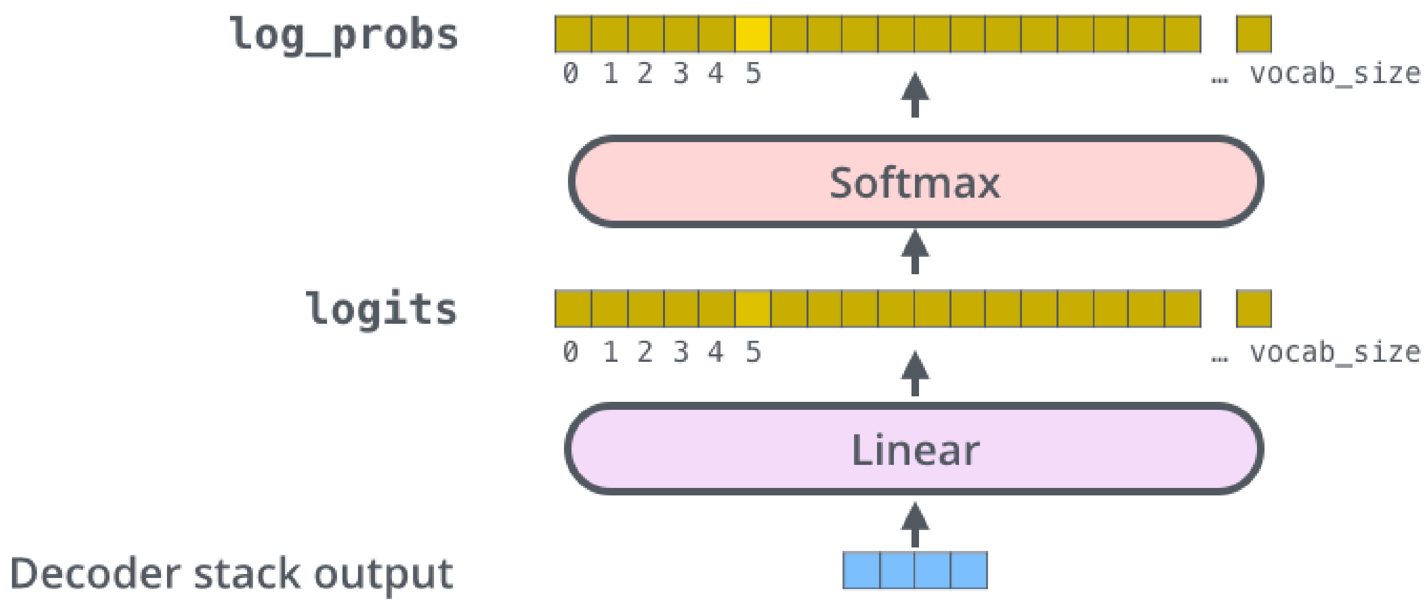
- **2 sublayer changes in decoder.**
- **Masked decoder**
  - Self-attention on previously generated outputs is only used.



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

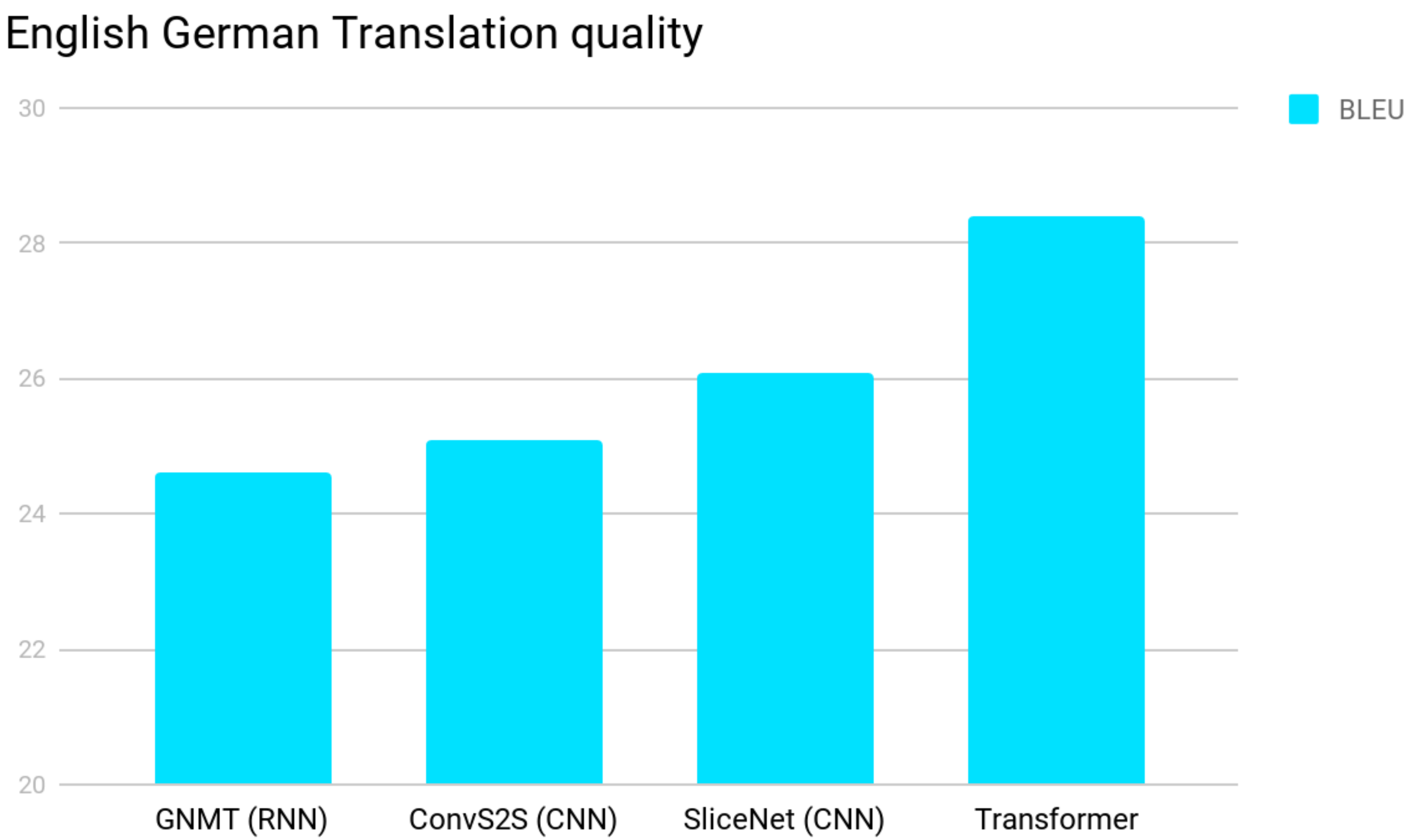
Final Linear and Softmax Layer

- 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).



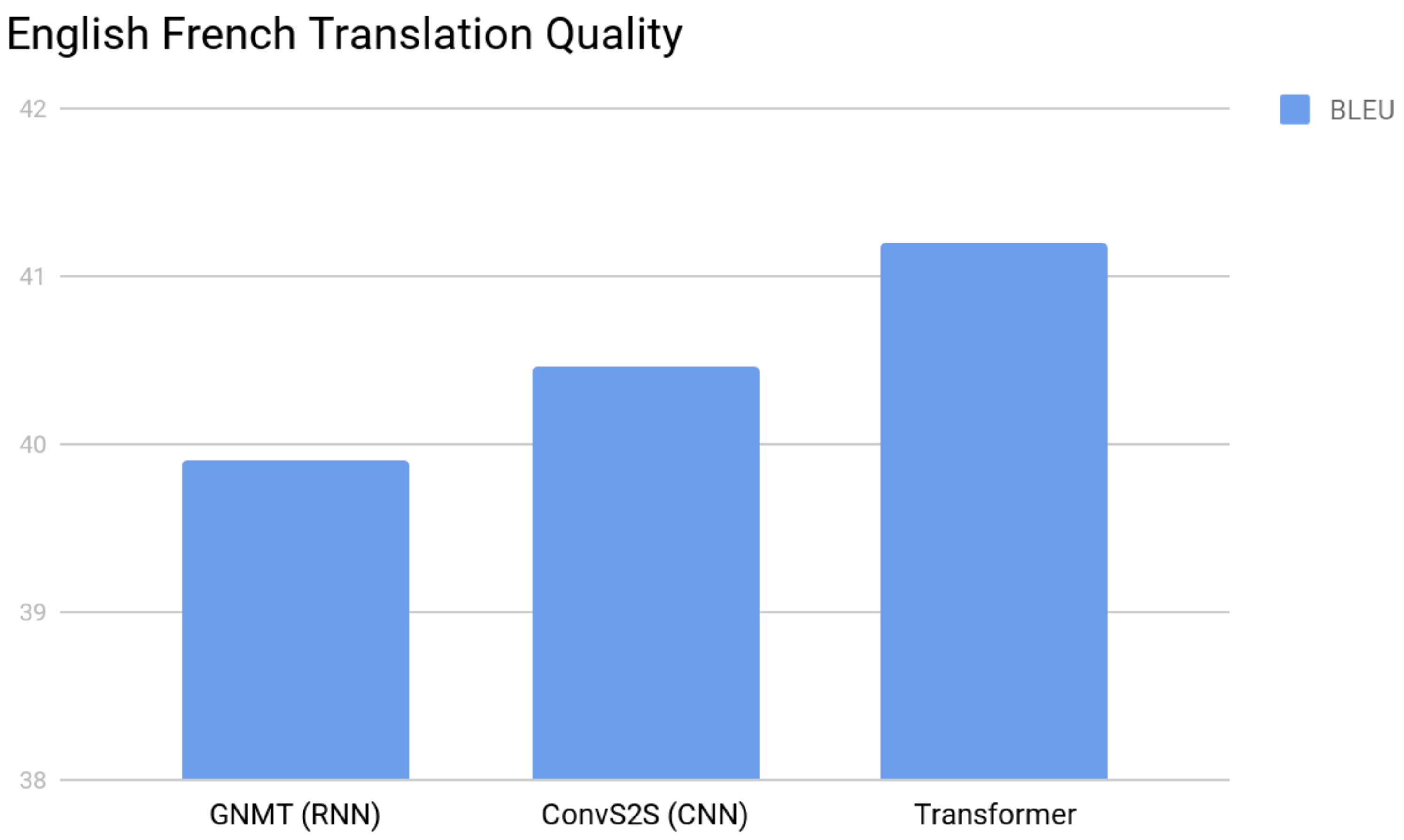
Experimental Results

- https://ai.googleblog.com/2017/08/transformer-novel-neural-network.html



Experimental Results

- https://ai.googleblog.com/2017/08/transformer-novel-neural-network.html



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

References

➢ Online Courses

- Natural Language Processing with Deep Learning (Stanford)
  - Transformer Networks and Convolution Neural Networks
  - http://web.stanford.edu/class/cs224n/lectures/lecture12.pdf
- The Illustrated Transformer
  - http://jalammar.github.io/illustrated-transformer/

➢ Papers

- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, Illia Polosukhin, "Attention Is All You Need," NIPS 2018

Q&A