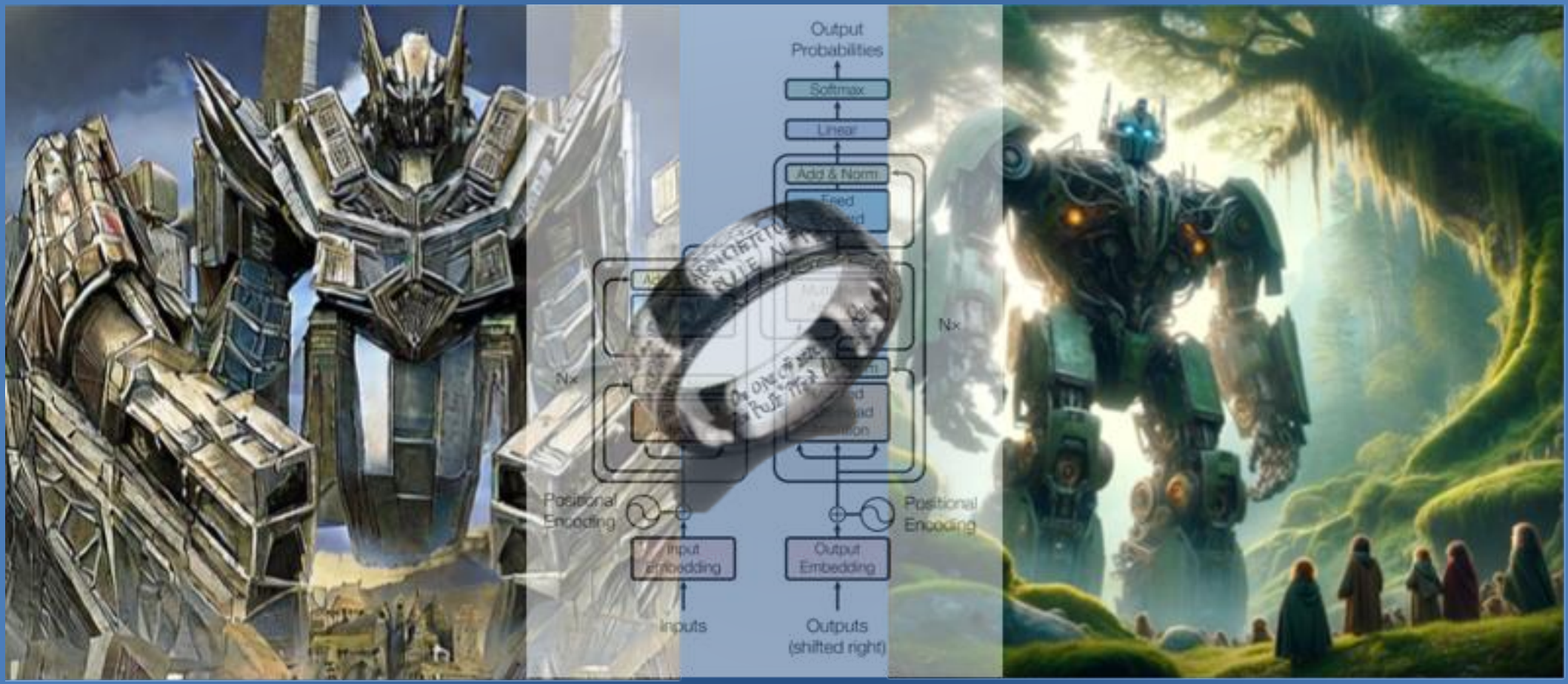
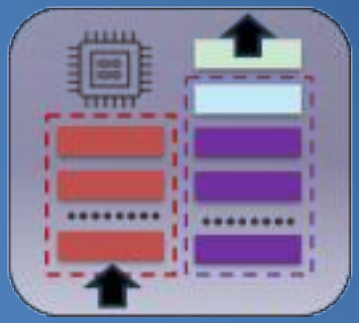


Image: Transformers: ' ...and one 'architecture' to rule them all' – Juxtaposition by Raeid Saqur (2024).



Transformers

CSC401/2511 – Natural Language Computing – Spring 2026

Ken Shi and Gerald Penn

Lecture 7

University of Toronto

Transformer networks

- Breakout paper: Vaswani et al. (2017) Attention is all you need.
- **Core idea**: replace recurrent connections with attention

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.

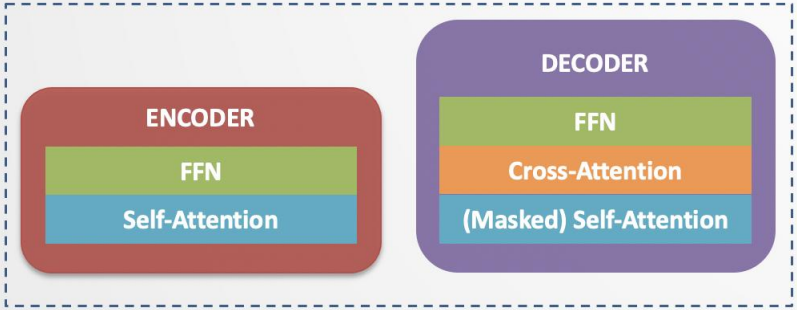
Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [15]	23.75			
Deep-Att + PosUnk [32]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [31]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [8]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [26]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
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	**$3.3 \cdot 10^{18}$**	
Transformer (big)	**28.4**	**41.0**	$2.3 \cdot 10^{19}$	

[1] Vaswani, Ashish, et al. "Attention is all you need." *NeuIPS* (2017).

Transformer networks (abstract)

Core Idea

Now two kinds of attention



- Encoder uses self-attention

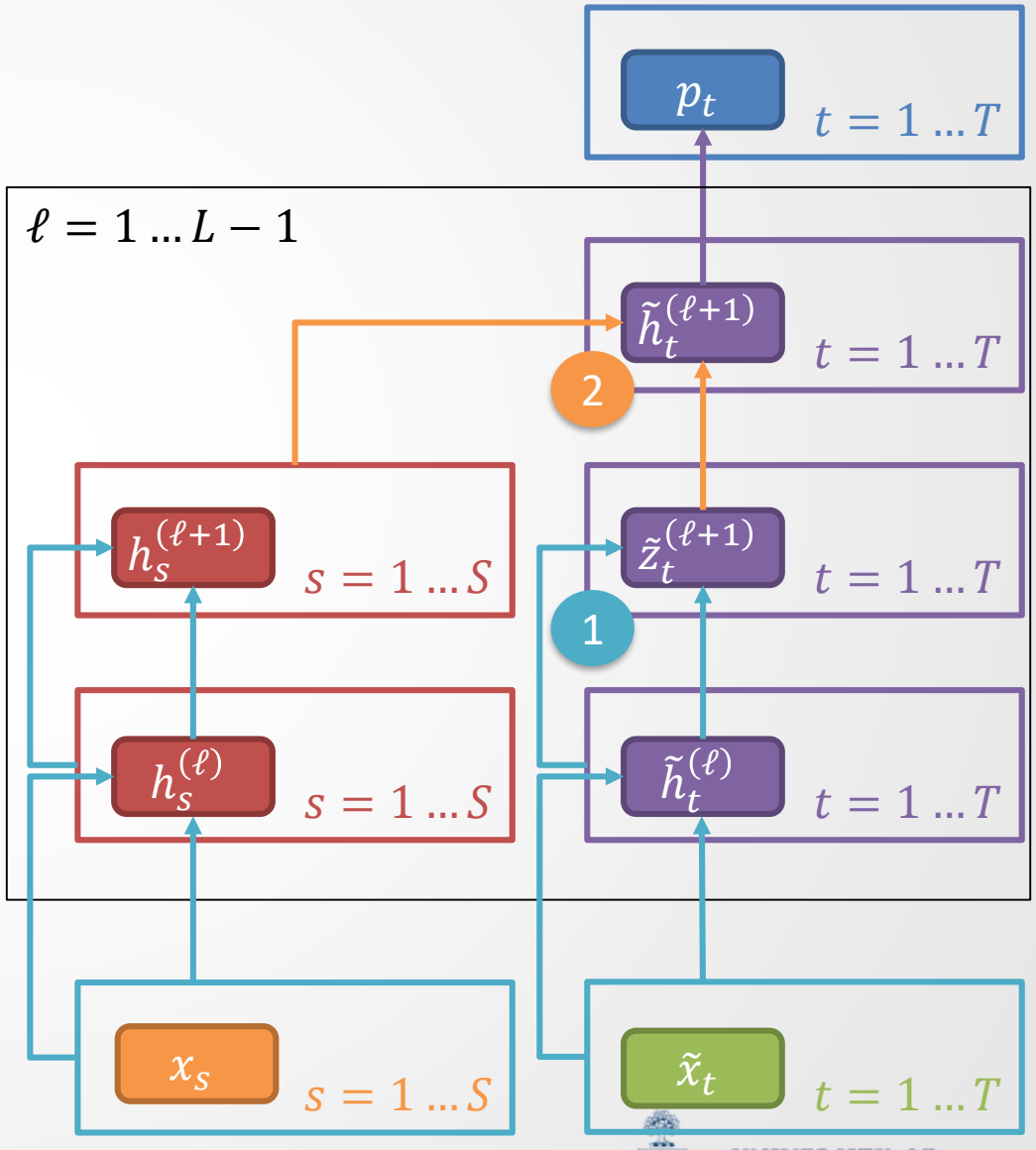
$$h_s^{(\ell+1)} \leftarrow Att_{Enc}\left(h_s^{(\ell)}, h_{1:S}^{(\ell)}\right)$$

Decoder uses 1. self-attention*

$$\tilde{z}_t^{(\ell+1)} \leftarrow Att_{Dec1}\left(\tilde{h}_t^{(\ell)}, \tilde{h}_{1:t}^{(\ell)}\right)$$

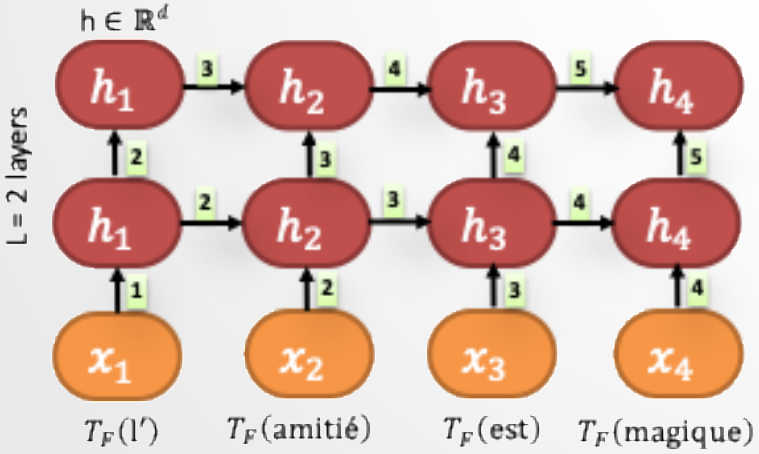
then 2. attention with encoder

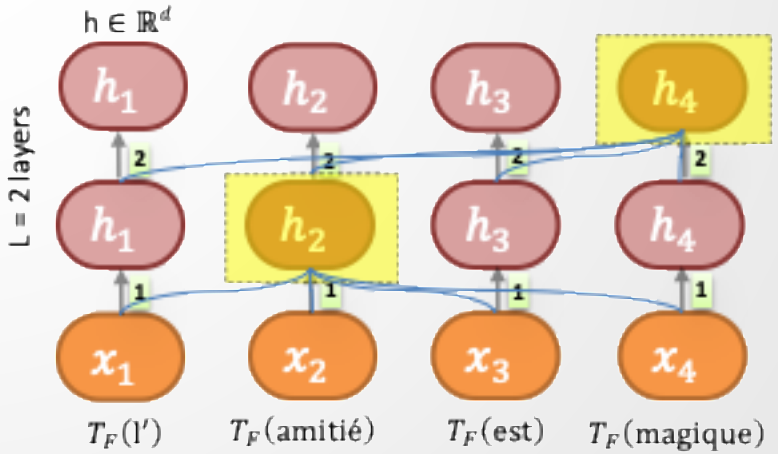
$$\tilde{h}_t^{(\ell+1)} \leftarrow Att_{Dec2}\left(\tilde{z}_t^{(\ell+1)}, h_{1:S}^{(\ell+1)}\right)$$



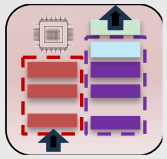
Transformer motivations

- Limitations of recurrent connections: long-term dependencies, lack of parallelizability, interaction distance (steps to distant tokens).
- Attention allows access to entire sequence
- Lots of computation can be shared, parallelized across sequence indices. Identical layers: [self, cross]-attention, feed-forward w/ tricks
- There are more *avant-garde* applications to other domains



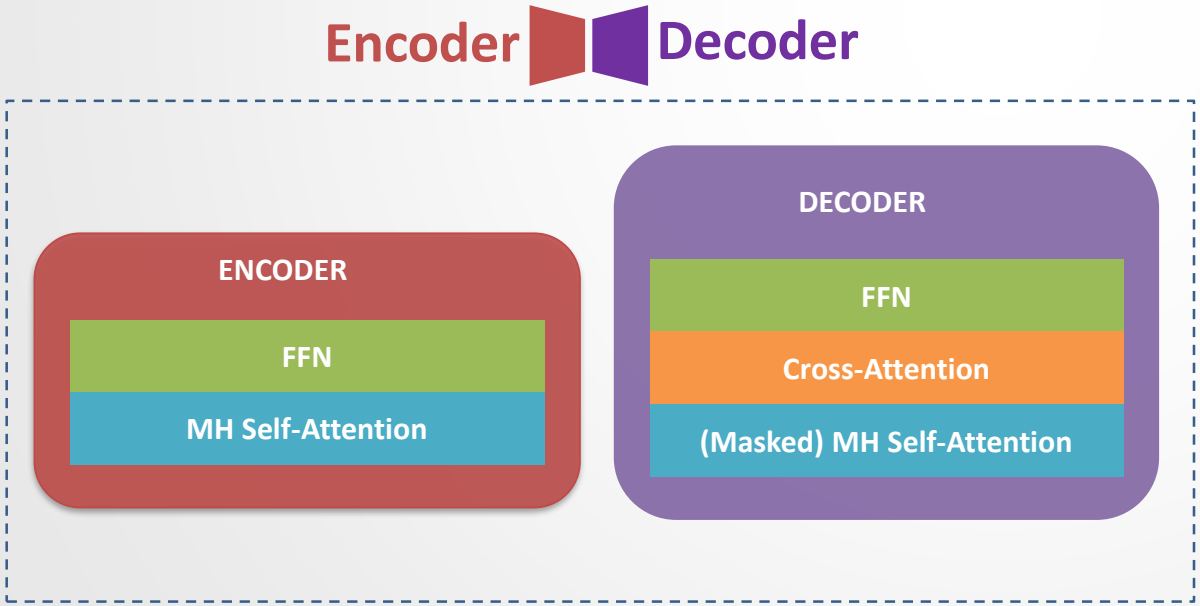


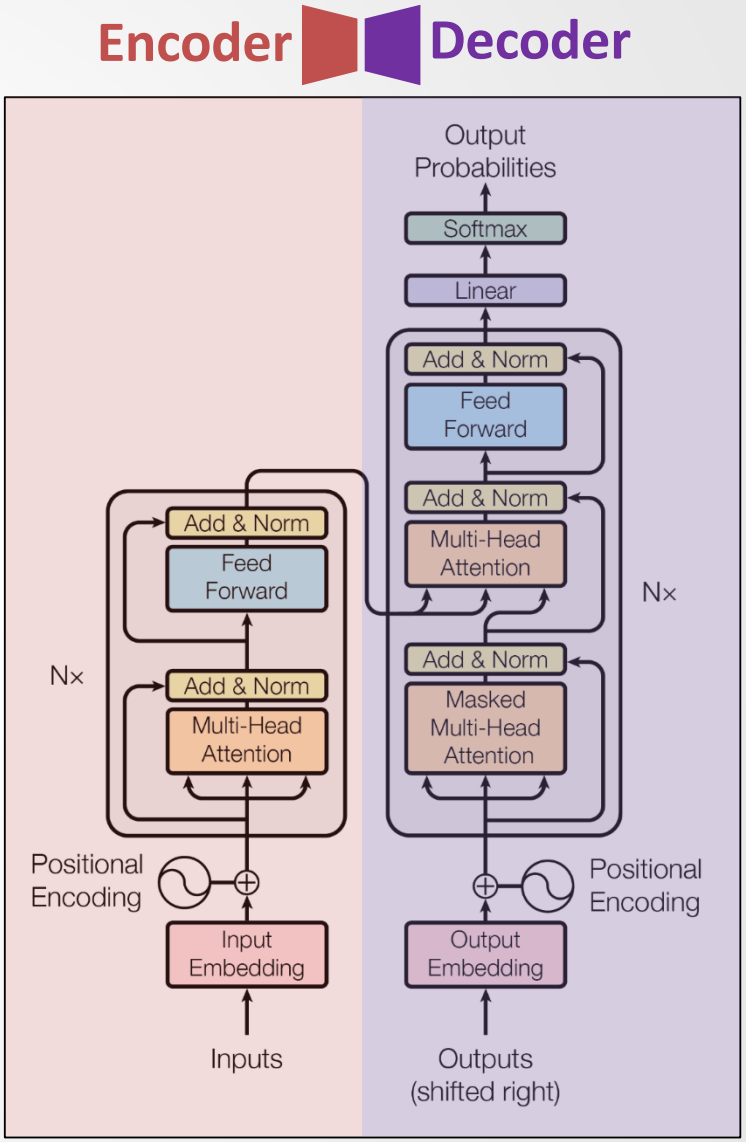
Source sentence (French): *L' amitié est magique*
Target sentence (English): *Friendship is magic*



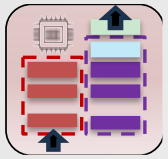
Transformer Architecture

- Architecture diagram from Vaswani[1]
- Building blocks:
  1. Encoder
  2. Decoder



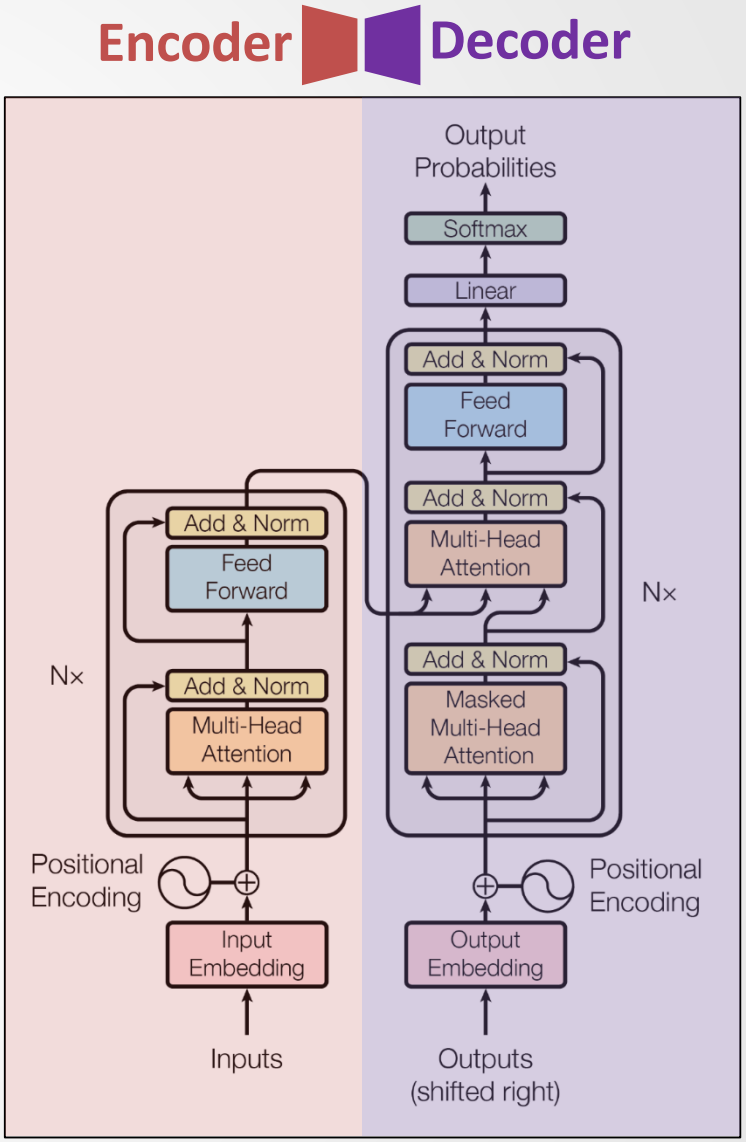


[1] Vaswani, Ashish, et al. "Attention is all you need." *NeurIPS* (2017).



Transformer Architecture

- Architecture diagram from Vaswani[1]
- Building blocks:
  1. Encoder
  2. Decoder
- Main components within building blocks:
  - Attention mechanisms:
    - single and multi-head attention
    - self, cross, and masked attention
  - Feed-forward MLPs (FFN)
  - Layer normalization (LN)
  - Positional encodings (PE)
  - Residual connections

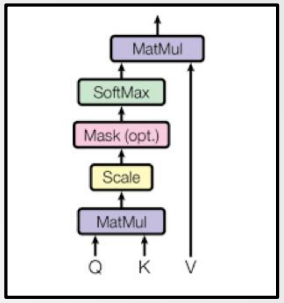


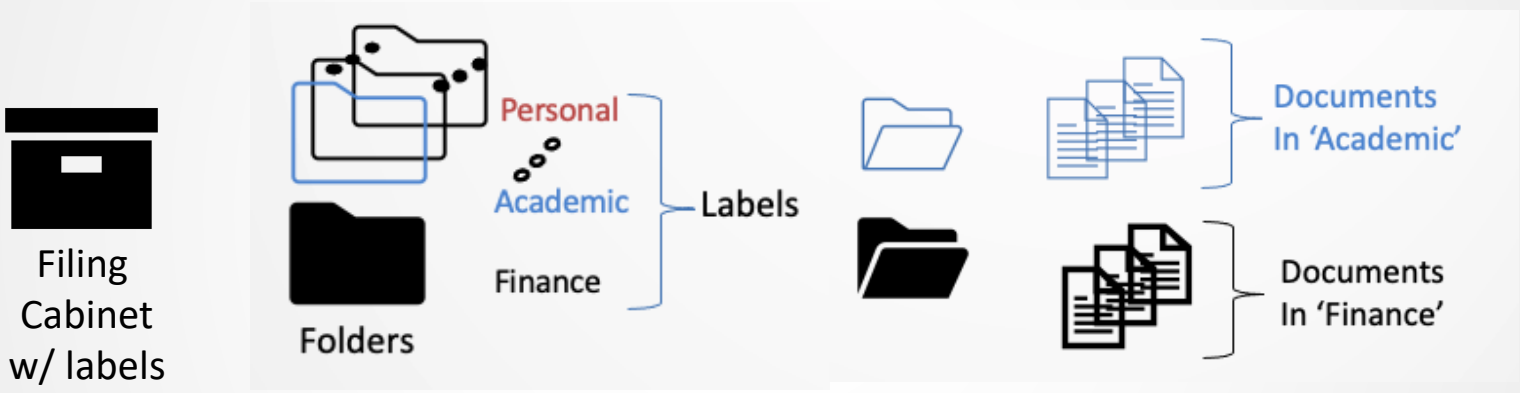
[1] Vaswani, Ashish, et al. "Attention is all you need." *NeuIPS* (2017).

Transformer Attention(Q,K,V) : Intuition

In the classical roboquity era (2100-2250 AD), humans in designated zones/zoos are only allowed *filing cabinets* and *paper documents* to store information.

ACORN has been terminated, and UofT students' info (financial, academic, personal) retrieval works as follows:





K Keys

V Values

Similarity Measure

Dot Product Attention

Q Queries

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

q1 What is the student's current academic standing?

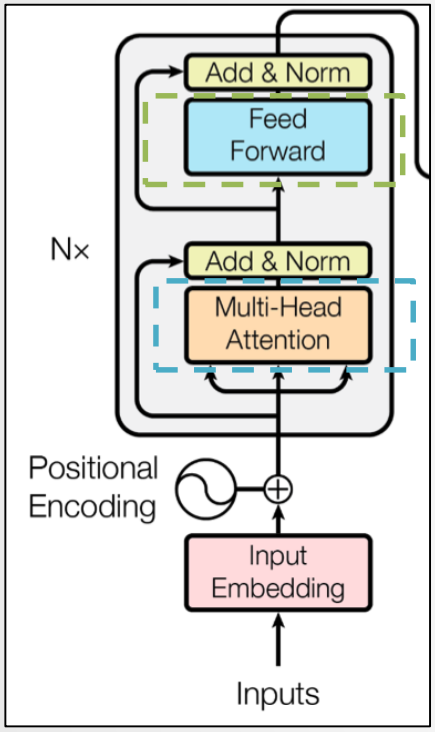
q2 What is the student's current financial status?

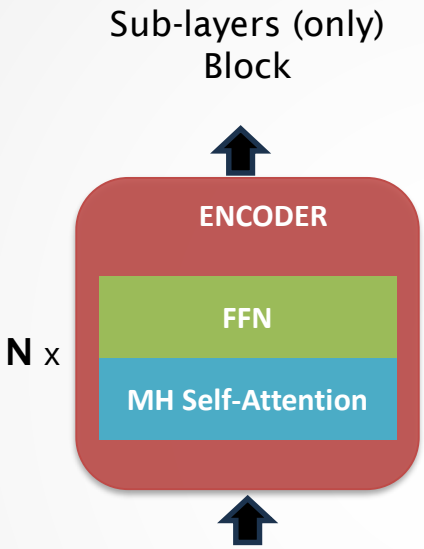
q3 What is the student's residency status in Canada?

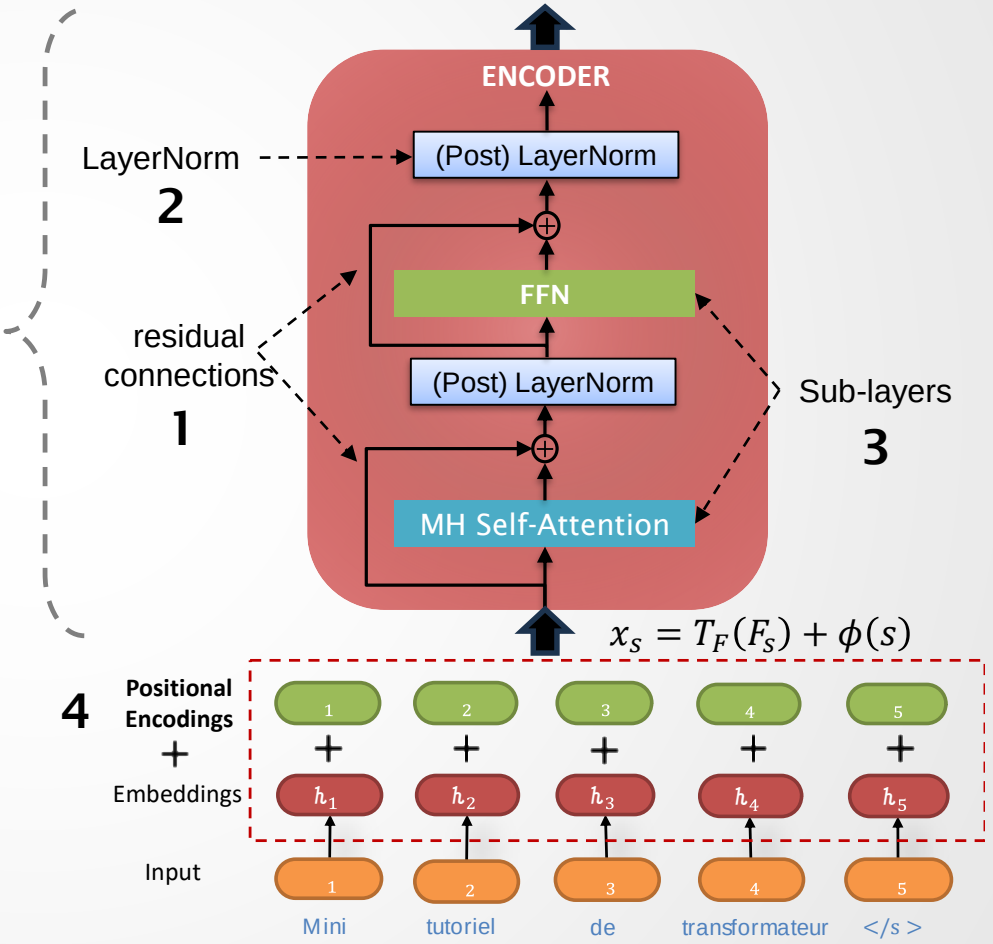
Transformer Encoder

F_S: Mini tutoriel de transformateur

Encoder Decoder E_T:Tiny transformer tutorial

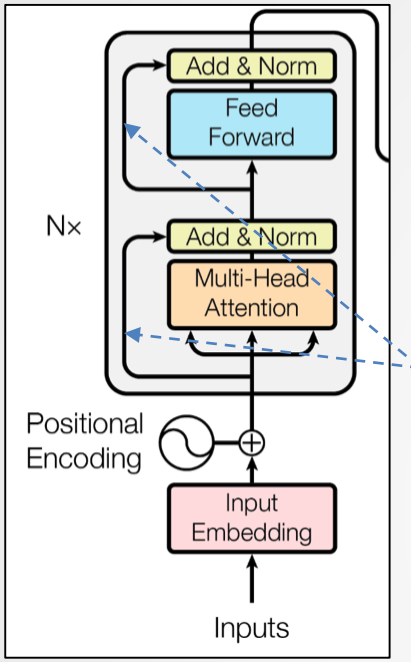


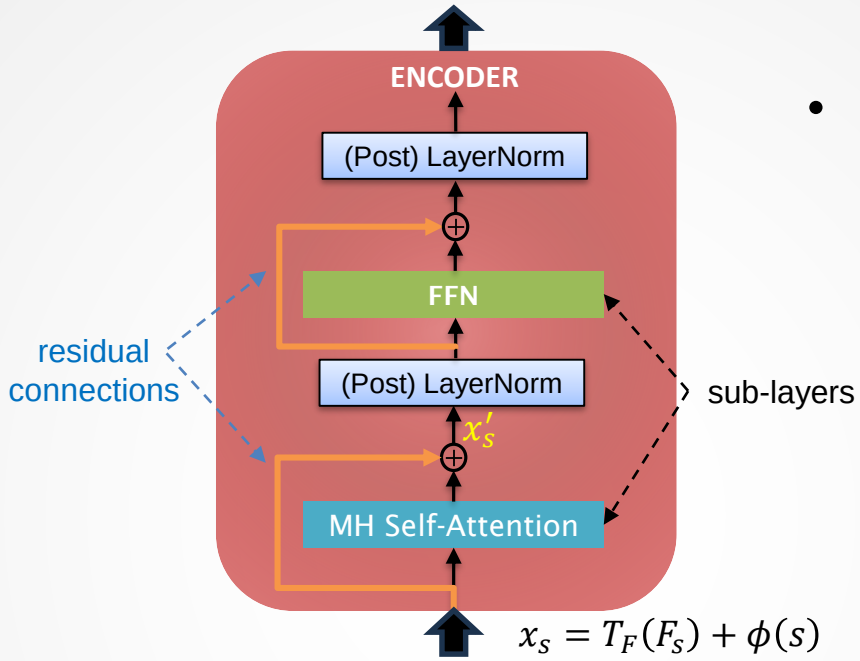




- **Plan**: discuss building blocks:
  1. Residual connections
  2. LayerNorm
  3. Attention and FFN sub-layers
  4. Positional encodings

Residual Connections





- Old idea used in speech:

$$x_{l+1} = \mathcal{F}(x_l) + x_l$$

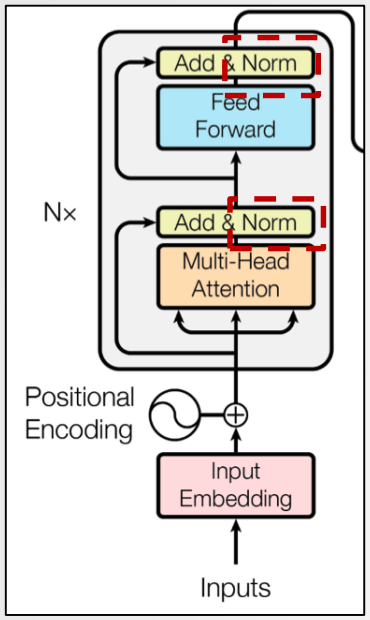
+ Helps smoothen loss curvature allowing better backprop.

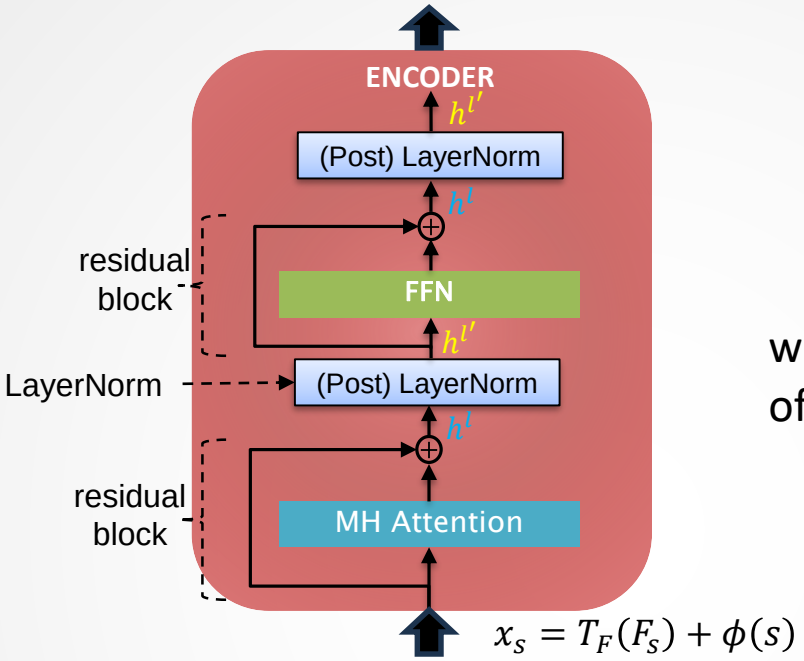
- Problem: NNs can't recover some important information from their input
- Solution: Add back the input embeddings to the sub-layer's output moving up

$$x'_s = Sublayer(x_s) + x_s$$

- *Analogy*: think of broadcasting through a noisy-channel. Big help to pass information through a side-channel without distortion.

Layer Normalization: default (Post-) LN





$$h^{l'} = LayerNorm(h^l) = \gamma\left(\frac{h^l - \mu^l}{\sigma^l}\right) + \beta$$

where μ, σ are mean and std. dev. of features in h^l. γ, β are scale, bias params.

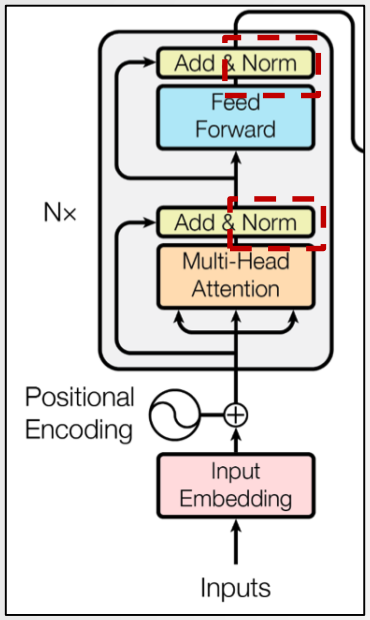
$$\mu = \frac{1}{d}\sum_{k=1}^{d} h_k^l$$

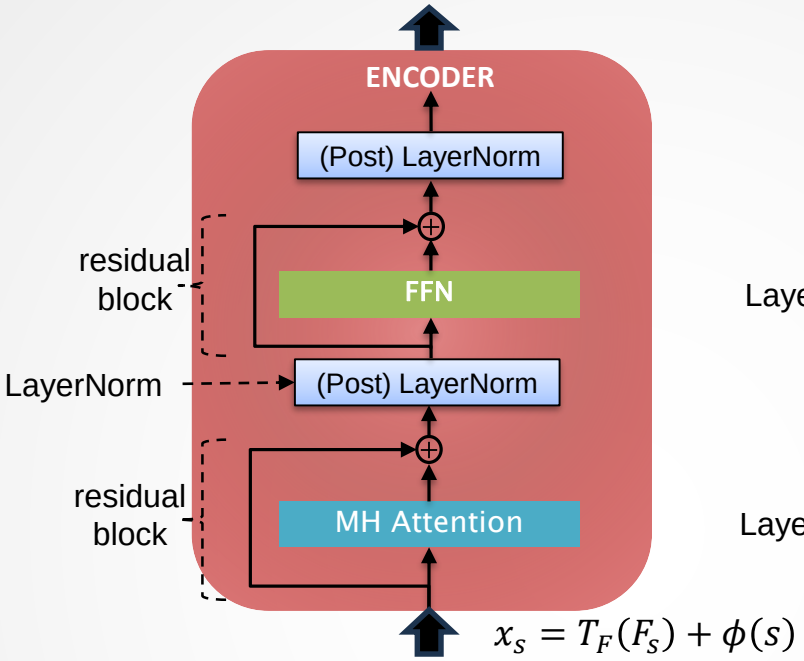
$$\sigma^2 = \frac{1}{d}\sum_{k=1}^{d} \left(h_k^l - \mu\right)^2$$

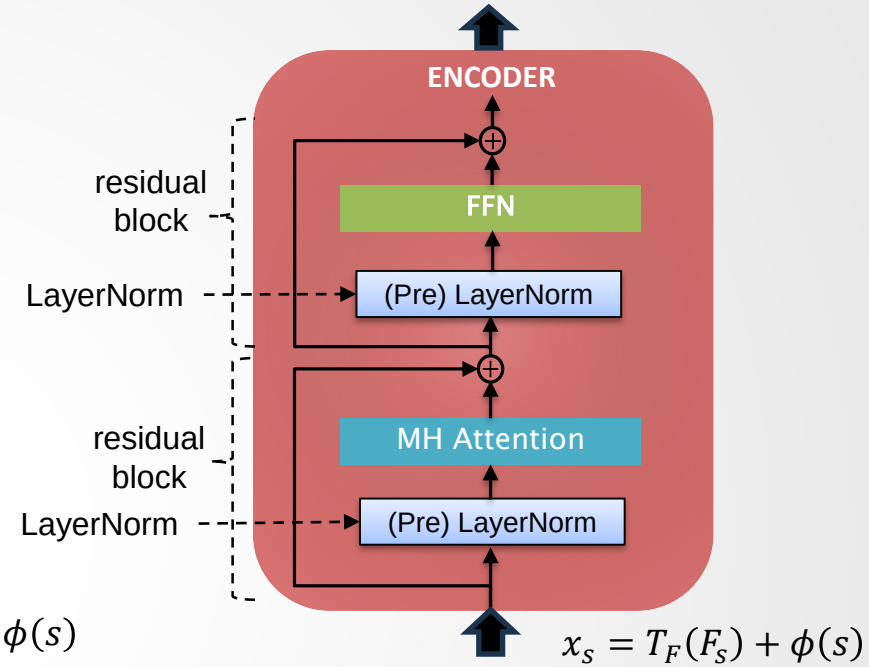
- Layer Normalization[1]:
  - **Normalize** input layer's distribution to 0 mean and 1 standard deviation.
  - Removes uninformative variation in layer's features

[1] Ba, Jimmy Lei, Jamie Ryan Kiros, and Geoffrey E. Hinton. "Layer normalization. 2016" [link]

Layer Normalization Variant: Pre-LN





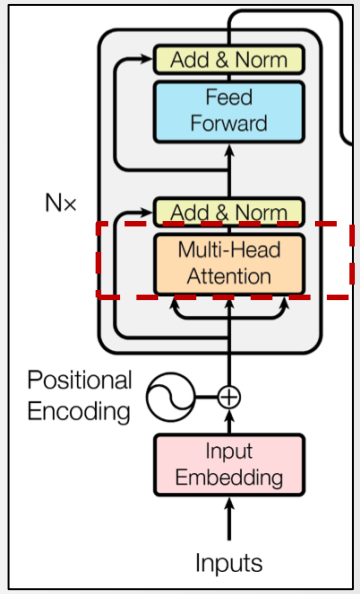


- Layer Normalization: two popular variants
  - *Post layer normalization* (Post-LN): original Transformer model: requires learning rate warm-up due to initial instability of large output gradients.
  - *Pre layer normalization* (**Pre-LN**): puts layer-norm within the residual block. Allows removing warm-up stage. More stable training initialization.[2]
- Secret sauce: Fisher information matrix (based on KL-divergence!)

[2] Xiong, Ruibin, et al. "On layer normalization in the transformer architecture." *ICML*. PMLR, 2020

Transformer Encoder - Self Attention

- Recall our discussion of attention in transformer LMs
- Steps:
  1. Calculate the query, key, and value for each token
     - Attention of each query (q_i) against all the keys ($k_{1:j}$)
  2. Calculate the **attention score** between query and keys
  3. **Normalize** the attention scores by applying softmax
  4. Calculate values by taking a **weighted sum**



$$q_i = W^Q x_i$$
$$k_i = W^K x_i$$
$$v_i = W^V x_i$$

$$a_{i,j} = score(q_i, k_j)$$
$$a_{i,j} = q_i . k_j$$
$$a_{i,j} = \frac{q_i . k_j}{\sqrt{d_k}}$$

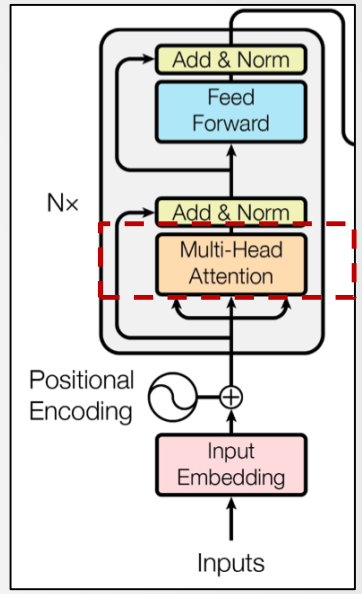
$$\alpha_{i,j} = softmax(a_{i,1:K})$$
$$\alpha_{i,j} = \frac{exp^{(a_{i,j})}}{\sum_{k=1}^{K} exp^{(a_{i,k})}}$$

$$c_i = \sum_j \alpha_{i,j} v_j$$

[1] Vaswani, Ashish, et al. "Attention is all you need." *NeurIPS* (2017).

Transformer Encoder - Self Attention

- Recall our discussion of attention in transformer LMs
- Steps:
  1. Calculate the query, key, and value for each token
     - Attention of each query (q_i) against all the keys ($k_{1:j}$)
  2. Calculate the **attention score** between query and keys
  3. **Normalize** the attention scores by applying softmax
  4. Calculate values by taking a **weighted sum**



Vectorized notation:

$$Q = XW^Q$$
$$K = XW^K$$
$$V = XW^V$$

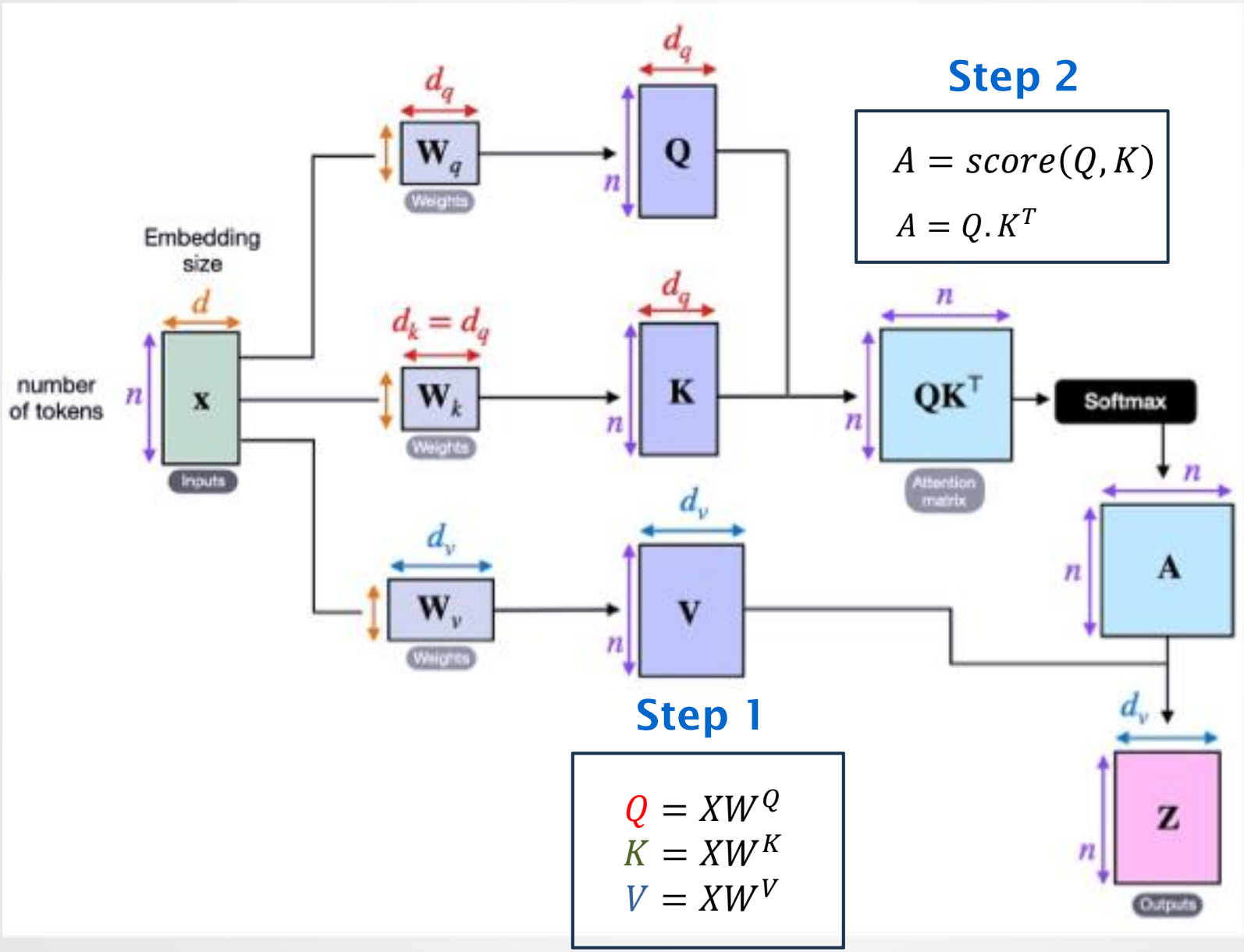
$$A = score(Q, K)$$
$$A = Q.K^T$$
$$A = \frac{Q.K^T}{\sqrt{d_k}}$$

$$\mathrm{A} = softmax(A)$$
$$\mathrm{A} = softmax(\frac{QK^T}{\sqrt{d_k}})$$

$$Z = \mathrm{A}.V$$

[1] Vaswani, Ashish, et al. "Attention is all you need." *NeuIPS* (2017).

Transformer Encoder - Self Attention



Step 1

$$Q = XW^Q$$
$$K = XW^K$$
$$V = XW^V$$

Step 2

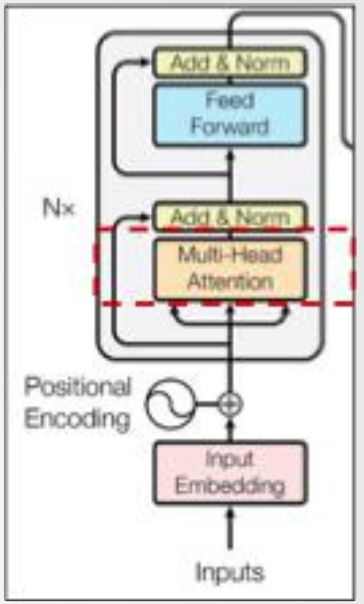
$$A = score(Q, K)$$
$$A = Q.K^T$$

Step 3

$$A = softmax(A)$$

Step 4

$$Z = A.V$$



[1] Vaswani, Ashish, et al. "Attention is all you need." *NeuIPS* (2017).

Multi-head Self Attention (MHA)

- Multi-head attention (MHA) jointly attends to information from different representation subspaces – hopefully the extra heads will get something right that the earlier heads missed.

$$MHA(Q,K,V) = Concat(head_1, \dots, head_h)W^O$$

$$where\ head_i = Attention\left(QW_i^Q, KW_i^K, VW_i^V\right)$$

And projections are parameter matrices:

$$W_i^Q \in \mathbb{R}^{d_{model} \times d_k}$$

$$W_i^K \in \mathbb{R}^{d_{model} \times d_k}$$

$$W_i^V \in \mathbb{R}^{d_{model} \times d_v}$$

$$W^O \in \mathbb{R}^{hd_v \times d_{model}}$$

$$d_k = d_v = \frac{d_{model}}{h}$$

[1] Vaswani, Ashish, et al. "Attention is all you need." *NeurIPS* (2017).

Feed-forward (FFN) layers

- Attention only **re-weighs** the **value** **vectors**
- We still need more degrees of freedom (and *non-linearity)* to learn
- The feed-forward layer(s) (**FFN**) provide these non-linearities to attention layer outputs
- Specifically, each output x undergoes two (layer) linear transformations with a ReLU activation in between. Pointwise:

$$FFN(x_i) = \max(0, xiW_1 + b_1)\, W_2 + b_2$$

$$z_i = \sum_j \alpha_{i,j} v_j$$

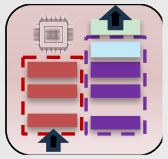
- FFN sub-layer is applied to each token pos. separately and identically
- Given x, a sequence of tokens $(x_1, \ldots, x_S)$:

$$FFN(x) = ReLU(xW_1 + b_1)\, W_2 + b_2$$

$$Z = \mathrm{A}.V$$

where W_1, W_2, b_1 *and* b_2 are parameters

[1] Vaswani, Ashish, et al. "Attention is all you need." *NeurIPS* (2017).



Recap: Transformer Architecture

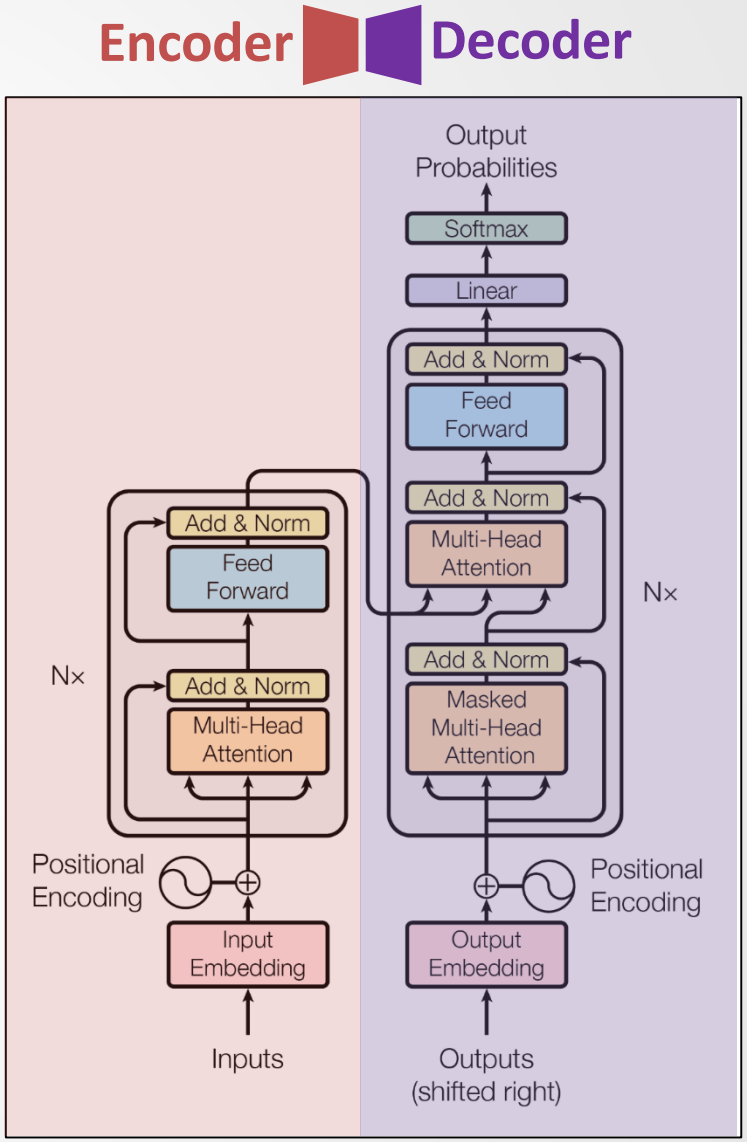
- Architecture diagram from Vaswani[1]
- Building blocks:

☑ 1. Encoder

2. **Decoder**

- Main components within building blocks:
  - Attention mechanisms:
    - ☑ single and multi-head attention
      - self, **cross, and masked attention**

☑
- Feed-forward MLPs (FFN)
- Layer normalization (LN)
- Positional encodings (PE)
- Residual connections



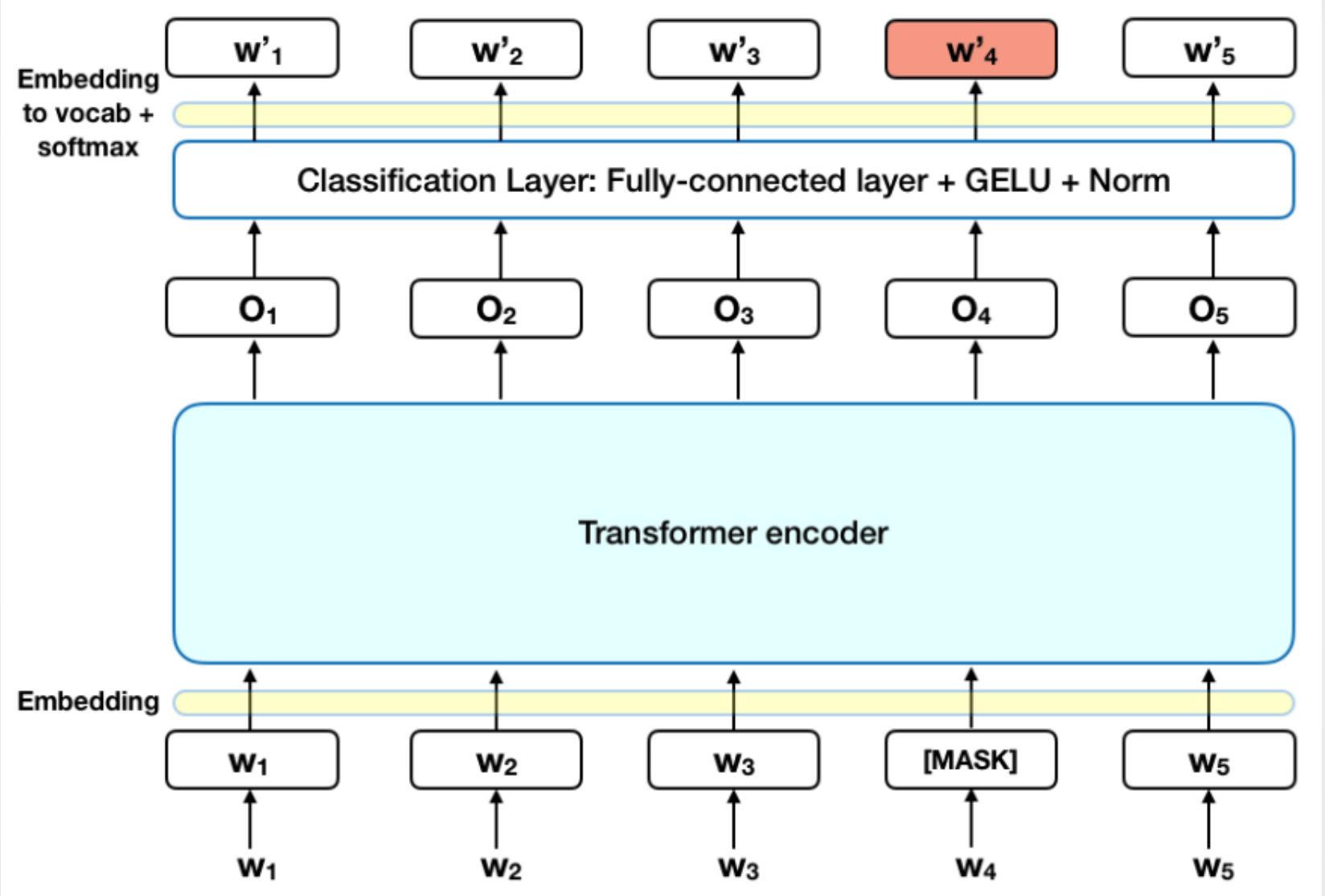
Aside: Contextualized embeddings

- What does the word *play* mean?

AllenNLP

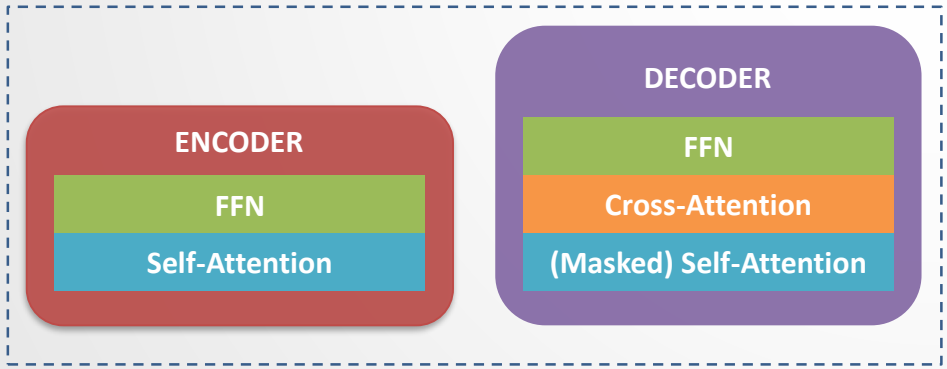
Peters ME, Neumann M, Iyyer M, *et al.* (2018) Deep contextualized word representations. Published Online First: 2018. doi:10.18653/v1/N18-1202; http://arxiv.org/abs/1802.05365

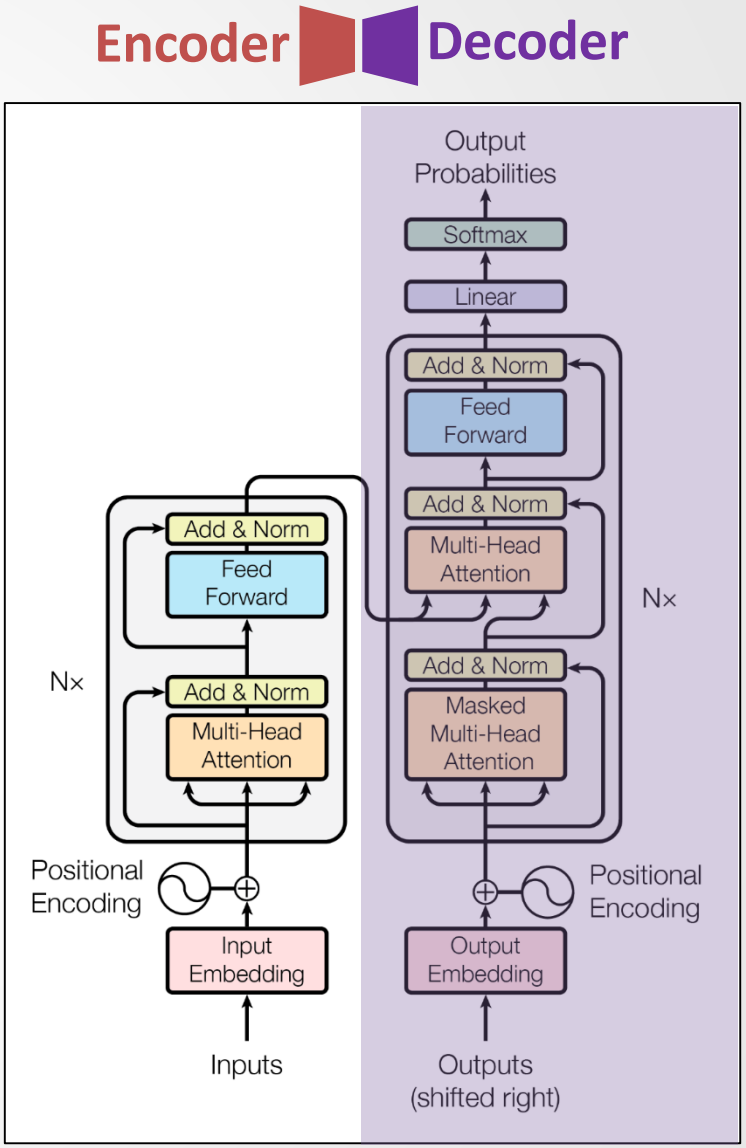
BERT: Masked Language Modelling



Next block: Transformer Decoder

- Layer normalization, residual connections, FFNs are identical to the encoder block
- Thus, we focus on remaining:
  - Masked/Causal self-attention sub-layer
  - Cross-attention

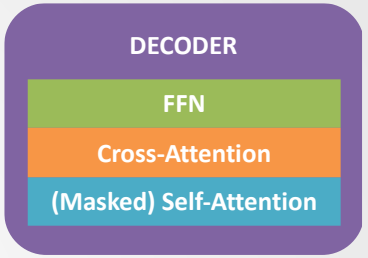




[1] Vaswani, Ashish, et al. "Attention is all you need." *NeurIPS* (2017).

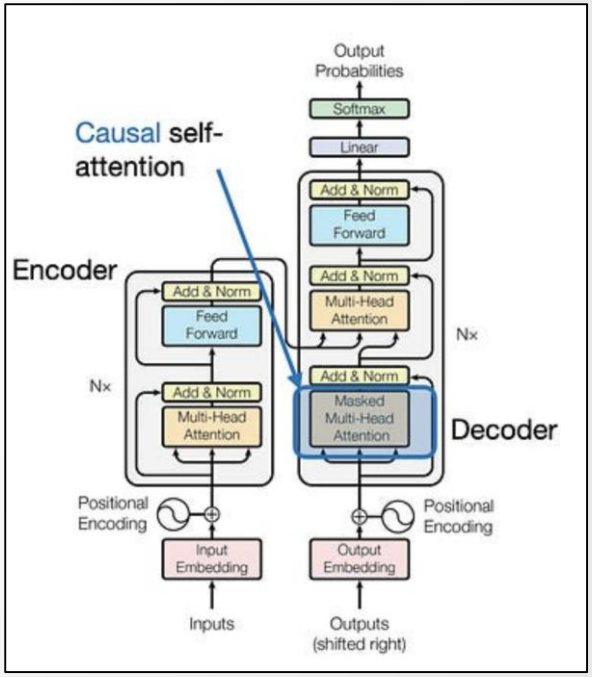
Decoder – Masked Self-Attention

- Masked (Multi-head) self-attention:
  - Enforce auto-regressive language modeling objective. The decoder cannot peek and pay attention to the (unknown) future words
  - **Solution**: use a *look-ahead mask M,* by setting attention scores of future tokens to –inf.



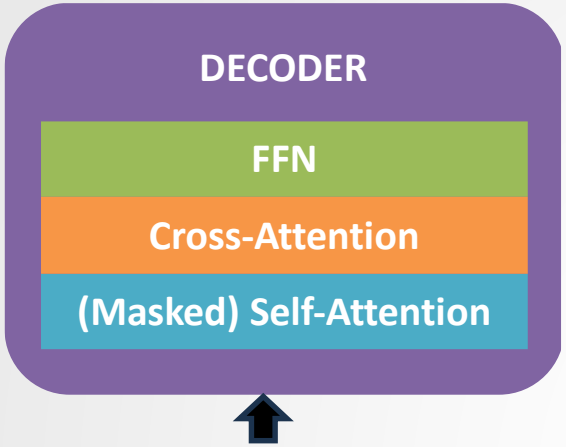
$$\begin{array}{c} \\ a^Q \\ b^Q \\ c^Q \\ D^Q \end{array} \begin{array}{c} \begin{array}{cccc} a^K & b^K & c^K & D^K \end{array} \\ \begin{bmatrix} 0 & -\infty & -\infty & -\infty \\ 0 & 0 & -\infty & -\infty \\ 0 & 0 & 0 & -\infty \\ 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

$$a_{ij} = \begin{cases} q_i^T . k_j, j < i \\ -\infty, j \geq i \end{cases}$$

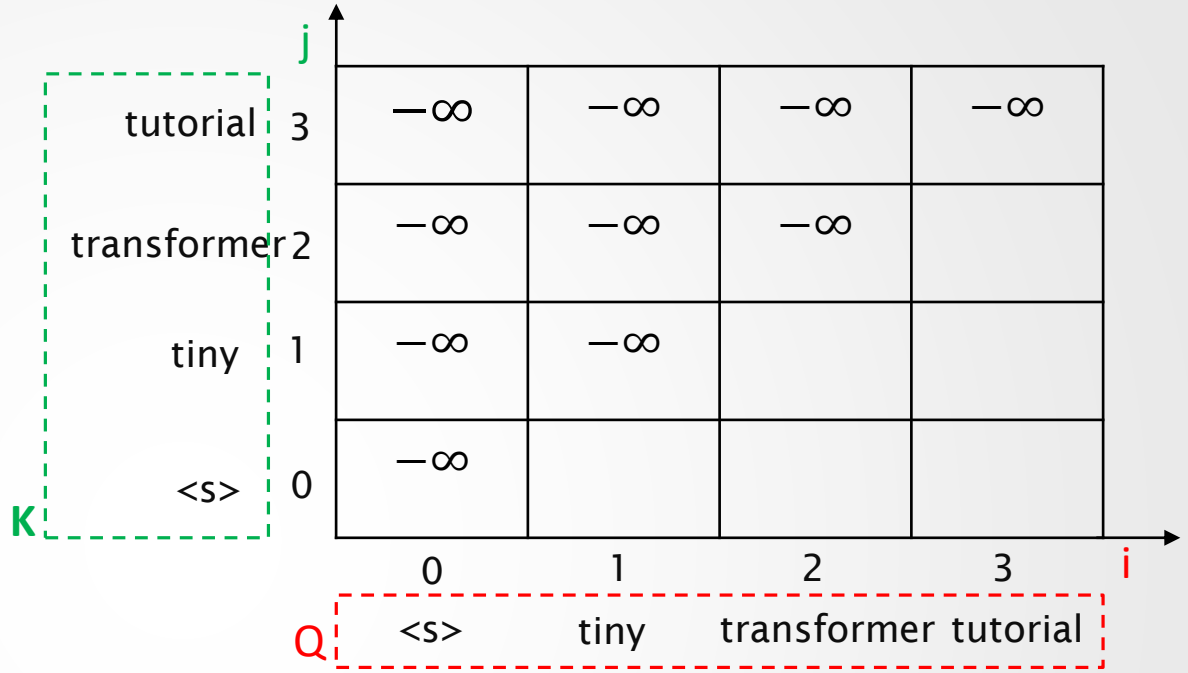


[1] Vaswani, Ashish, et al. "Attention is all you need." *NeurIPS* (2017).

Decoder – Masked Self-Attention



E_T: <s> tiny transformer tutorial



Recall

$$a_{i,j} = score(q_i, k_j)$$

or,

$$A = score(Q, K)$$

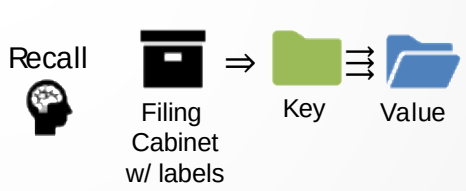
[1] Vaswani, Ashish, et al. "Attention is all you need." *NeurIPS* (2017).

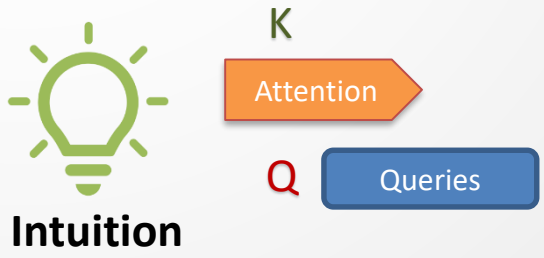
Encoder-Decoder (Cross) Attention

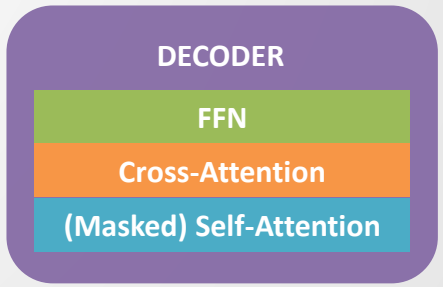
- In self-attention: Q, K and V have same source (tokens)
- **Cross attention** is encoder <> decoder attention between output vectors
- Using our running example and notation:
  - Let $h_1, \ldots, h_S$ be encoder output vectors, where $h_i \in \mathbb{R}^{d_k}$
  - Let $\tilde{h}_1, \ldots, \tilde{h}_T$ be decoder output vectors, where $\tilde{h}_i \in \mathbb{R}^{d_q}$

- Then, keys and values: K and V comes from encoder (or, *memory*):
  - $k_i = K h_i \,,\; v_i = V h_i$

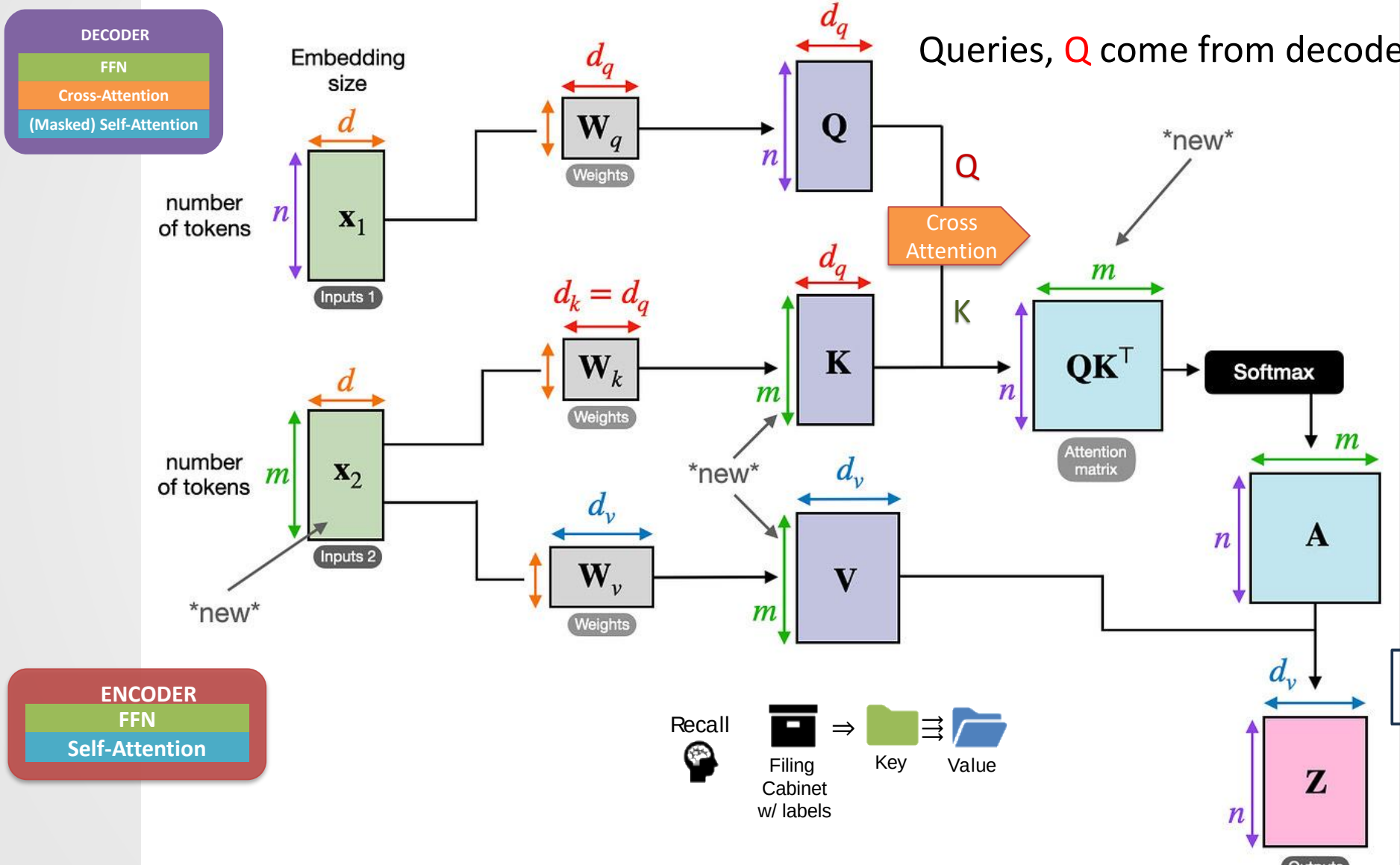
- Queries, Q comes from decoder:
  - $q_i = Q \tilde{h}_i$



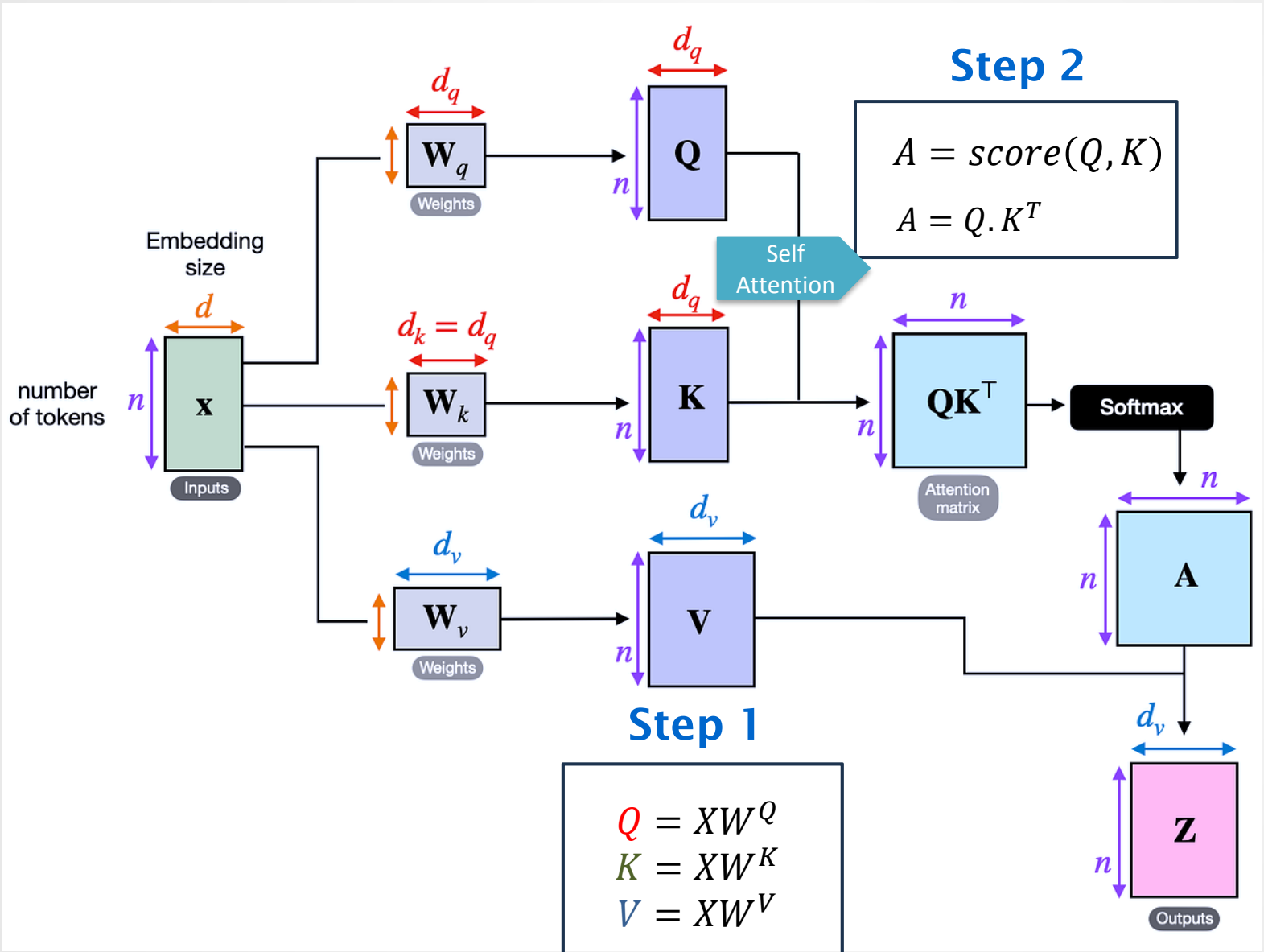


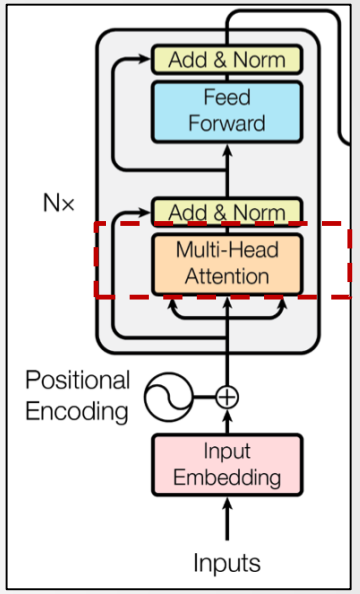


Encoder-Decoder (Cross) Attention



(Compare) Encoder - Self Attention





[1] Vaswani, Ashish, et al. "Attention is all you need." *NeuIPS* (2017).

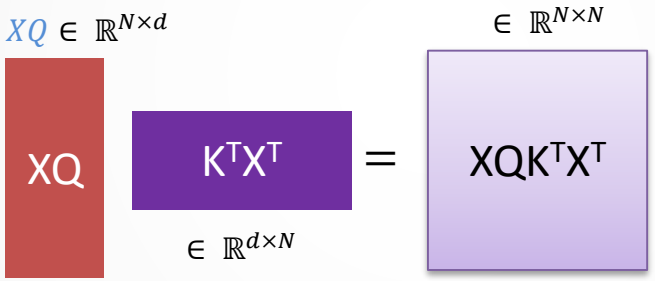
Recap

- We have now covered all the primitives you need for building a transformer!
- All of these equations look abstract, but not to worry …
- Assignment 2 was designed for you to implement all these concepts into a working MT model of your own.

Tasks	Section	Class	criterion	Max mark	Sub-Total	File
1	Building Blocks	LayerNorm	:forward	2		
2		MultiHeadAttention	:attention	4		
3			:forward	5		
4		FeedForwardLayer	:forward	1	12	
5	Architecture	TransformerEncoderLayer	:pre_layer_norm_forward	2		
6			:post_layer_norm_forward	1		
7		TransformerDecoderLayer	:__init__	4		
8			:pre_layer_norm_forward	2		
9			:post_layer_norm_forward	2		
10		TransformerDecoder	:forward	3		*a2_transformer_model.py*
11		TransformerEncoderDecoder	:create_pad_mask	1		
12			:create_causal_mask	2		
13			:forward	3	20	
15			:greedy_decode	5		

Transformers - Drawbacks

- Attention's **quadratic** computation **cost**
  - Function of sequence length N, and token dimension d
  - Computing all token pairs mean the function grows quadratically with N, $O(N^2 d)$ unlike RNNs: $O(Nd)$



- Can you see why this could be the biggest hurdle for increasing a transformer LM's **context length** (i.e., the size of input it can process)?

Transformers - Drawbacks

- Context (input) size limitation:
  - Dimension d in modern LLMs are ~>3K
  - If length of one sentence is ~10-30 word tokens, then computation scales with 10^2-30^2 times d
  - Thus, many encoders set a bound on N (usually 500-700 tokens)
  - But, some people want **N** to be much larger, e.g., processing a document (N > 10K) at one go (instead of chunking by N for every call)

- Active research area: improving the quadratic cost of attention, like self-attention with linear complexity[1]
  - + Roformer, flash attention, sliding-window etc.

[1] Wang, Sinong, et al. "Linformer: Self-attention with linear complexity." (2020).

Transformers - Drawbacks

- Other drawbacks / improvement areas:
  - **Positional encoding representations**:
    - Do we need absolute indices to represent position?

$$f_{t:t\in\{q,k,v\}}(\boldsymbol{x}_i, i) := \boldsymbol{W}_{t:t\in\{q,k,v\}}(\boldsymbol{x}_i + \boldsymbol{p}_i)$$

  - Slew of variants proposed to the (sinusoidal, absolute) positional encoding we saw

- General trend:
  - Move towards **relative position encoding**
  - E.g. Relative linear position attention [Shaw et al. 2018]

$$f_q(\boldsymbol{x}_m) := \boldsymbol{W}_q\boldsymbol{x}_m$$
$$f_k(\boldsymbol{x}_n, n) := \boldsymbol{W}_k(\boldsymbol{x}_n + \tilde{\boldsymbol{p}}_r^k)$$
$$f_v(\boldsymbol{x}_n, n) := \boldsymbol{W}_v(\boldsymbol{x}_n + \tilde{\boldsymbol{p}}_r^v)$$

where $\tilde{\boldsymbol{p}}_r^k, \tilde{\boldsymbol{p}}_r^v \in \mathbb{R}^d$ are trainable relative position embeddings

Relative distance between pos. m and n

$$r = clip(m - n, r_{min}, r_{max})$$

Position (in)dependence

- To a great extent, attention is agnostic to order
  - For permutation vector v on $(1,2,\ldots,V)$:

$$Att(a, b_v) = Att(a, b_{1:V})$$

- But *word order* **matters** in language
- **Solution**: encode position in input:

$$x_s = T_F(F_s) + \phi(s)$$

- More on this soon…

Aside: Rotary Position Embeddings

- RoPE – perhaps the most common in modern LLMs
  - Encodes absolute position with a rotation matrix
  - RoPE + Transformer = **RoFormer**

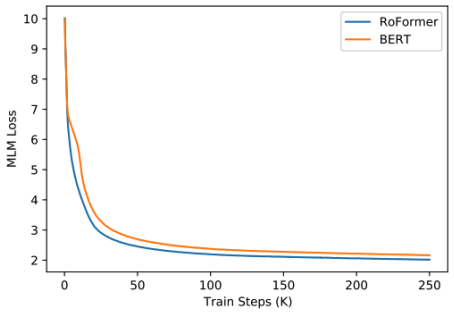
$$f_{\{q,k\}}(\boldsymbol{x}_m, m) = \boldsymbol{R}^d_{\Theta,m}\boldsymbol{W}_{\{q,k\}}\boldsymbol{x}_m$$

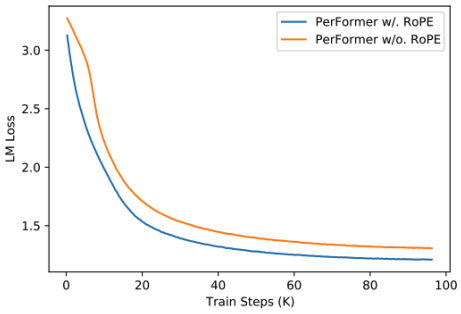
$$f_{\{q,k\}}(\boldsymbol{x}_m, m) = \begin{pmatrix} \cos m\theta & -\sin m\theta \\ \sin m\theta & \cos m\theta \end{pmatrix} \begin{pmatrix} W^{(11)}_{\{q,k\}} & W^{(12)}_{\{q,k\}} \\ W^{(21)}_{\{q,k\}} & W^{(22)}_{\{q,k\}} \end{pmatrix} \begin{pmatrix} x^{(1)}_m \\ x^{(2)}_m \end{pmatrix}$$

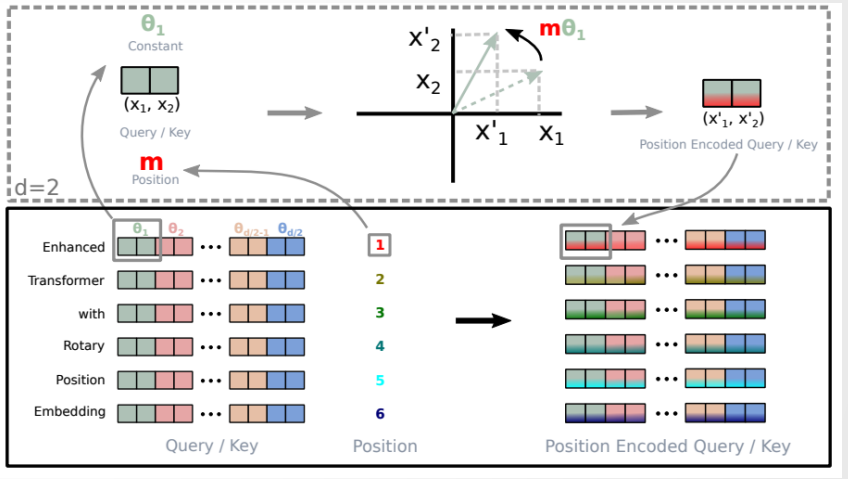
Model	BLEU
Transformer-baseVaswani et al. [2017]	27.3
RoFormer	**27.5**

Table 2: Comparing RoFormer and BERT by fine tuning on downstream GLEU tasks.

Model	MRPC	SST-2	QNLI	STS-B	QQP	MNLI(m/mm)
BERTDevlin et al. [2019]	88.9	93.5	90.5	85.8	71.2	84.6/83.4
RoFormer	**89.5**	90.7	88.0	**87.0**	**86.4**	80.2/79.8







[1] **RoPE**: Su, Jianlin, et al. "Roformer: Enhanced transformer with rotary position embedding." (2021)