

Lecture 05:

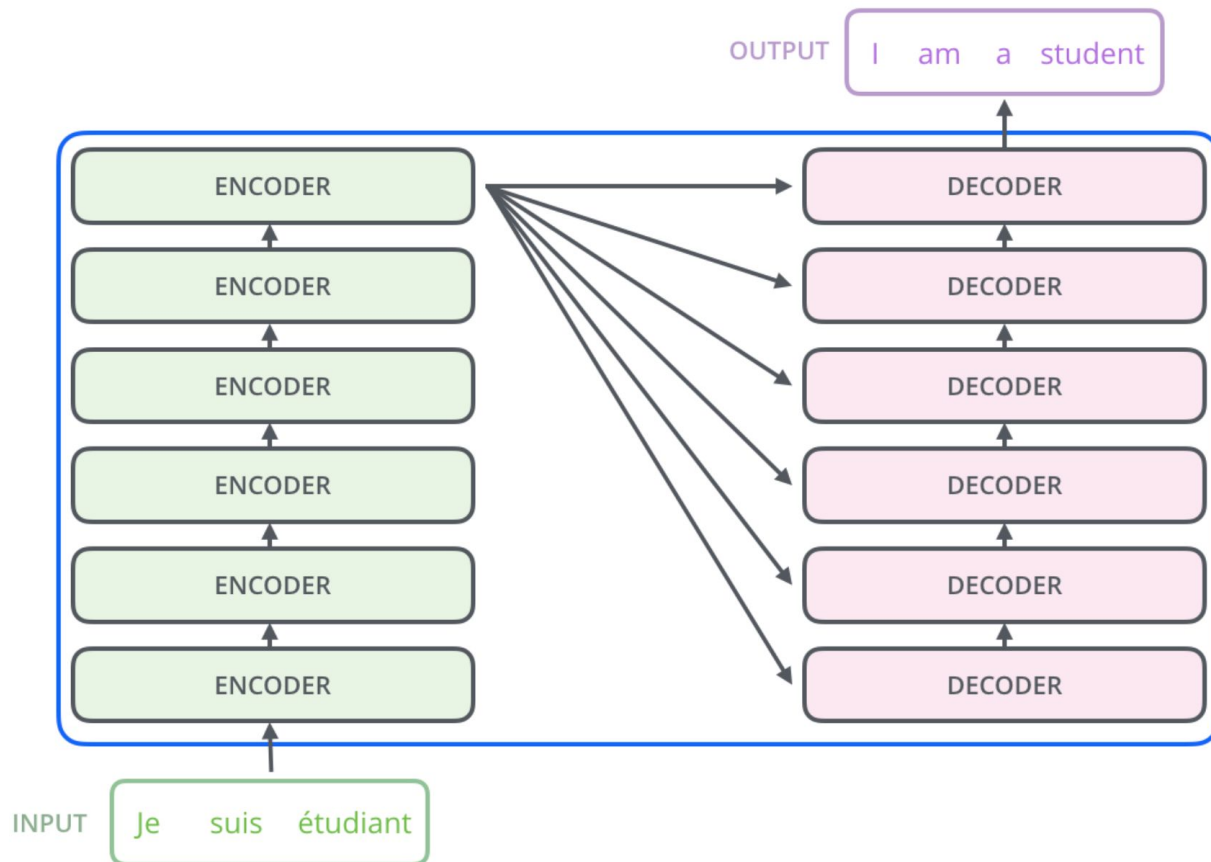
Transformer overview & Positional encoding

Radoslav Neychev

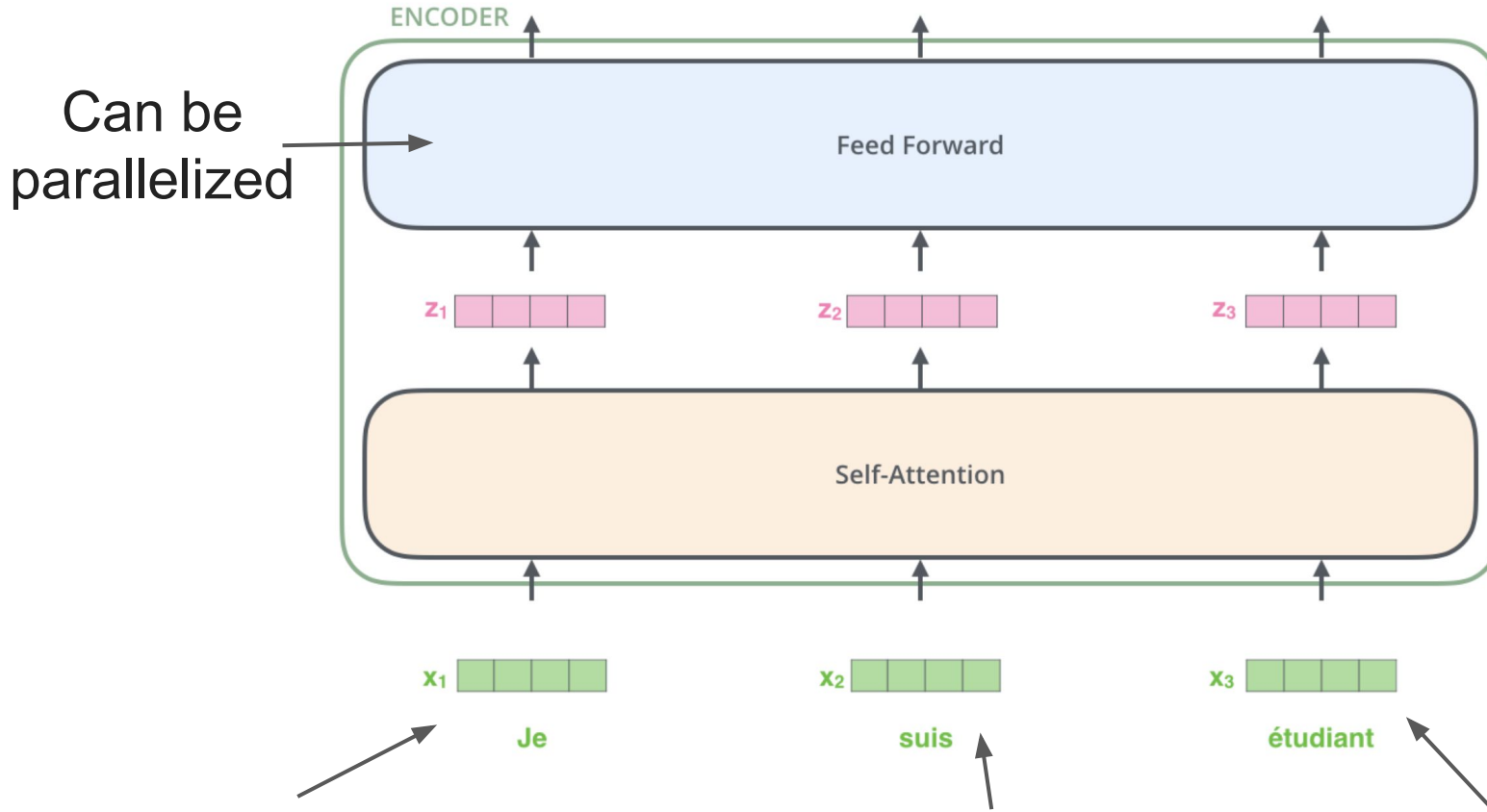
MADE, Moscow

1. recap: Self-attention
2. Positional encoding
3. Layer normalization
4. Decoder in Transformer

The Transformer



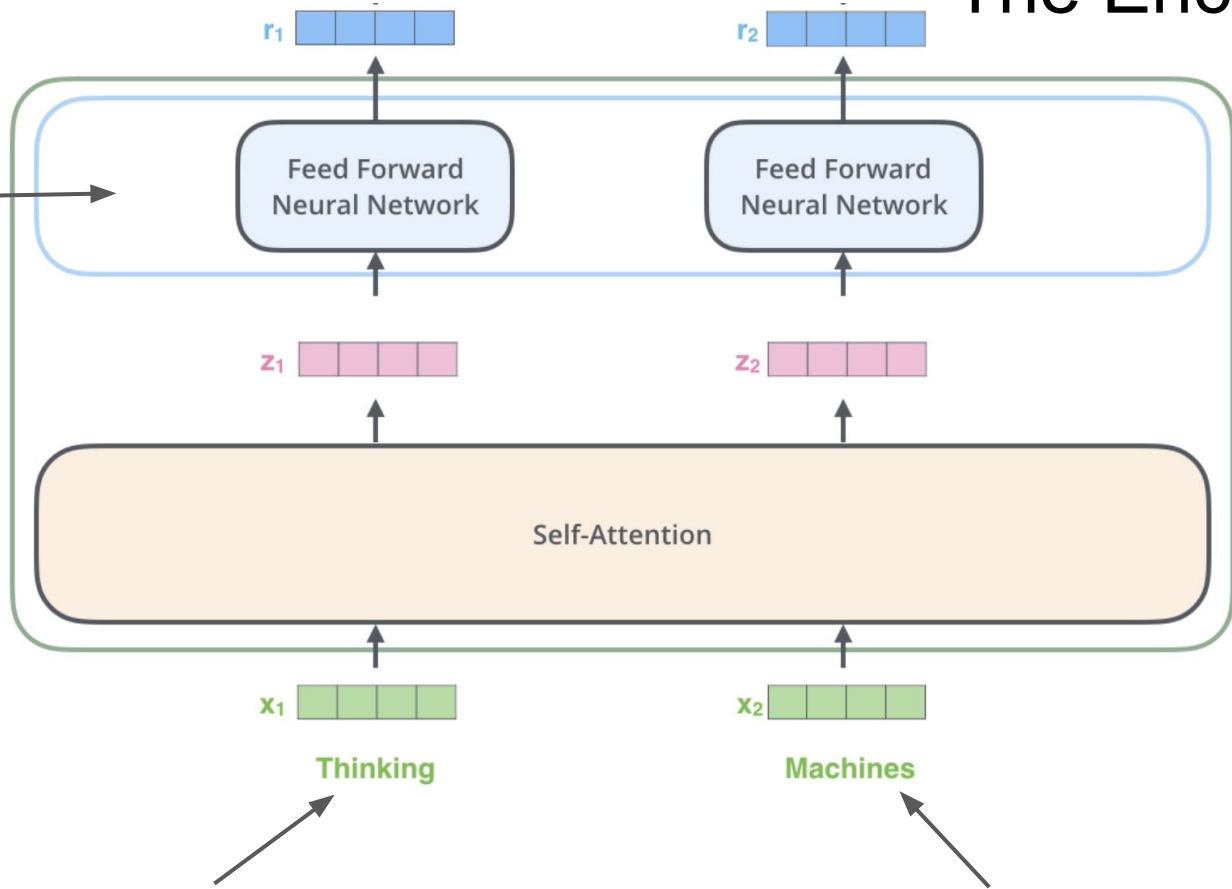
The Encoder Side



the word in each position flows through its own path in the encoder

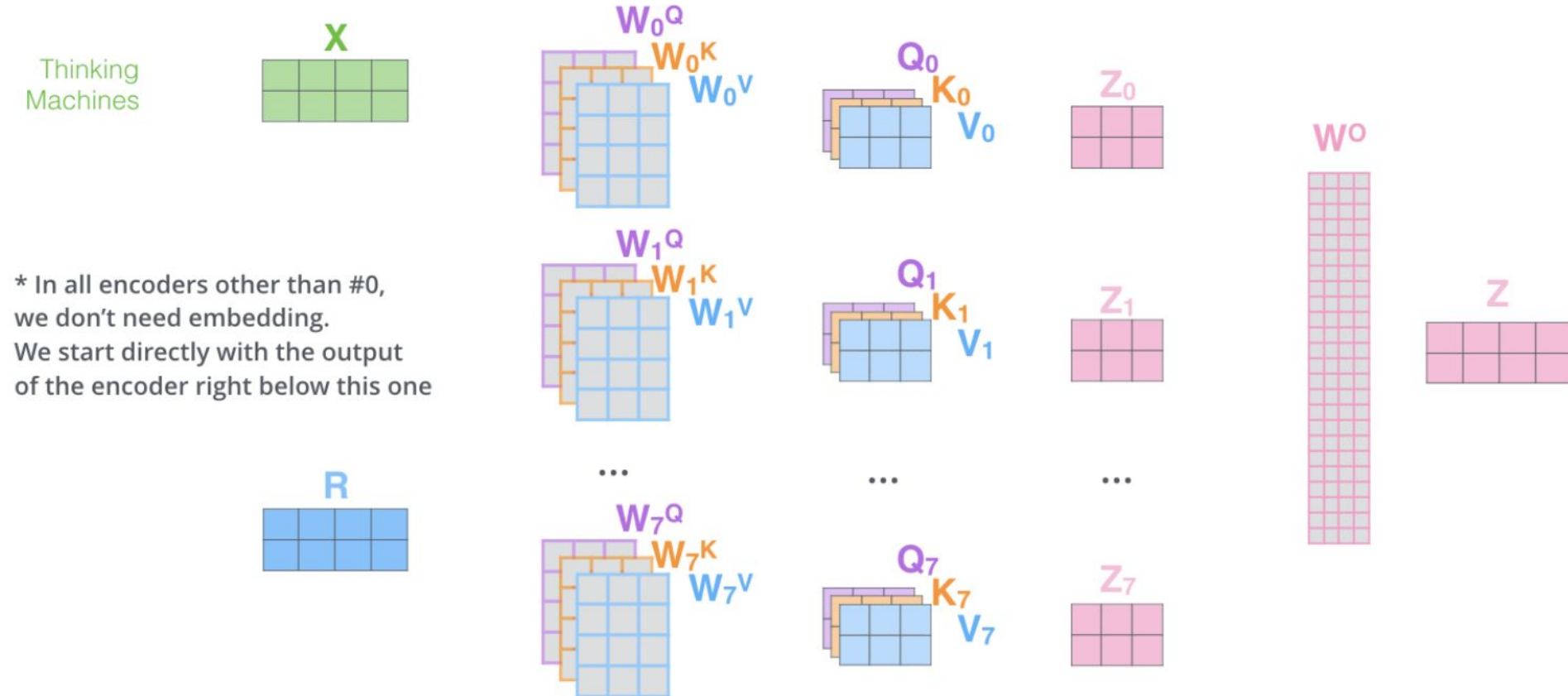
The Encoder Side

Can be
parallelized



the word in each position flows through its own path in the encoder

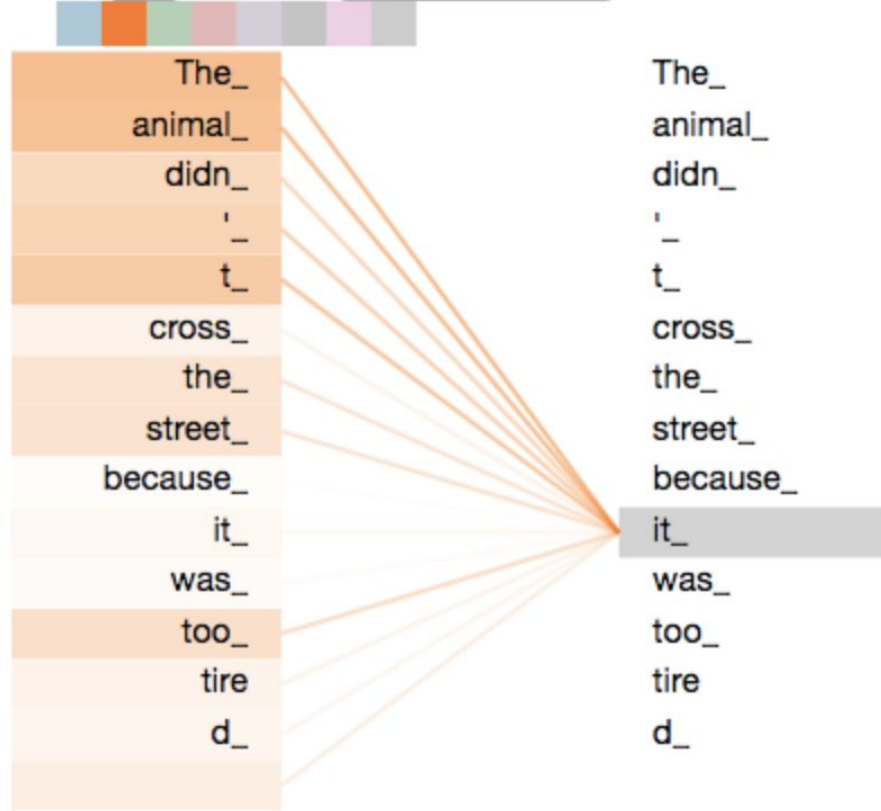
- 1) This is our input sentence*
- 2) We embed each word*
- 3) Split into 8 heads. We multiply X or R with weight matrices
- 4) Calculate attention using the resulting $Q/K/V$ matrices
- 5) Concatenate the resulting Z matrices, then multiply with weight matrix W^O to produce the output of the layer



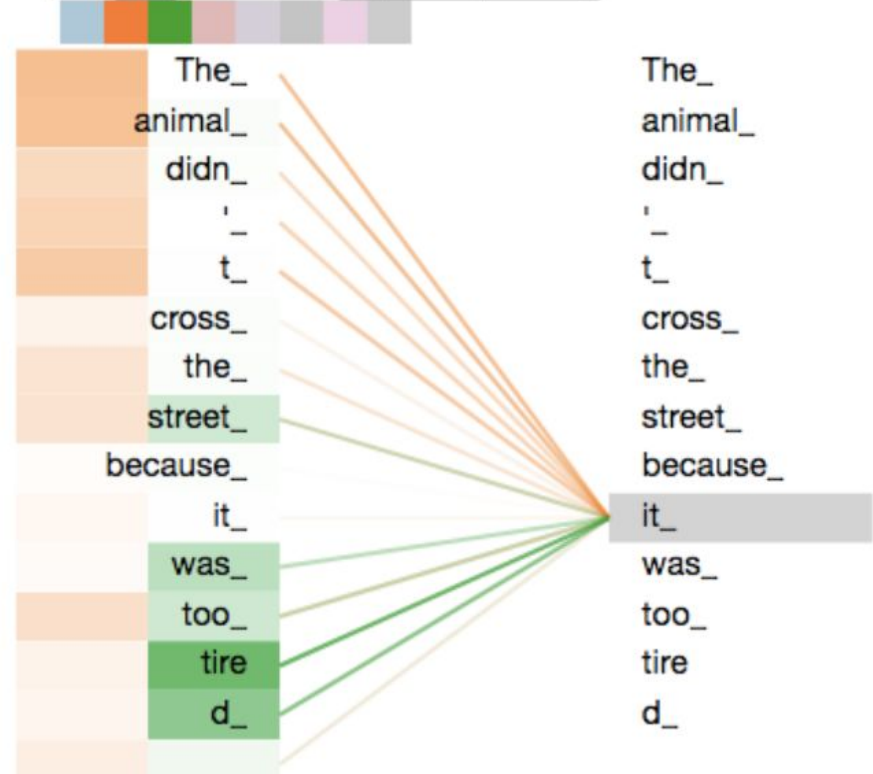
* In all encoders other than #0, we don't need embedding. We start directly with the output of the encoder right below this one

Multi-Head Attention

Layer: 5 Attention: Input - Input

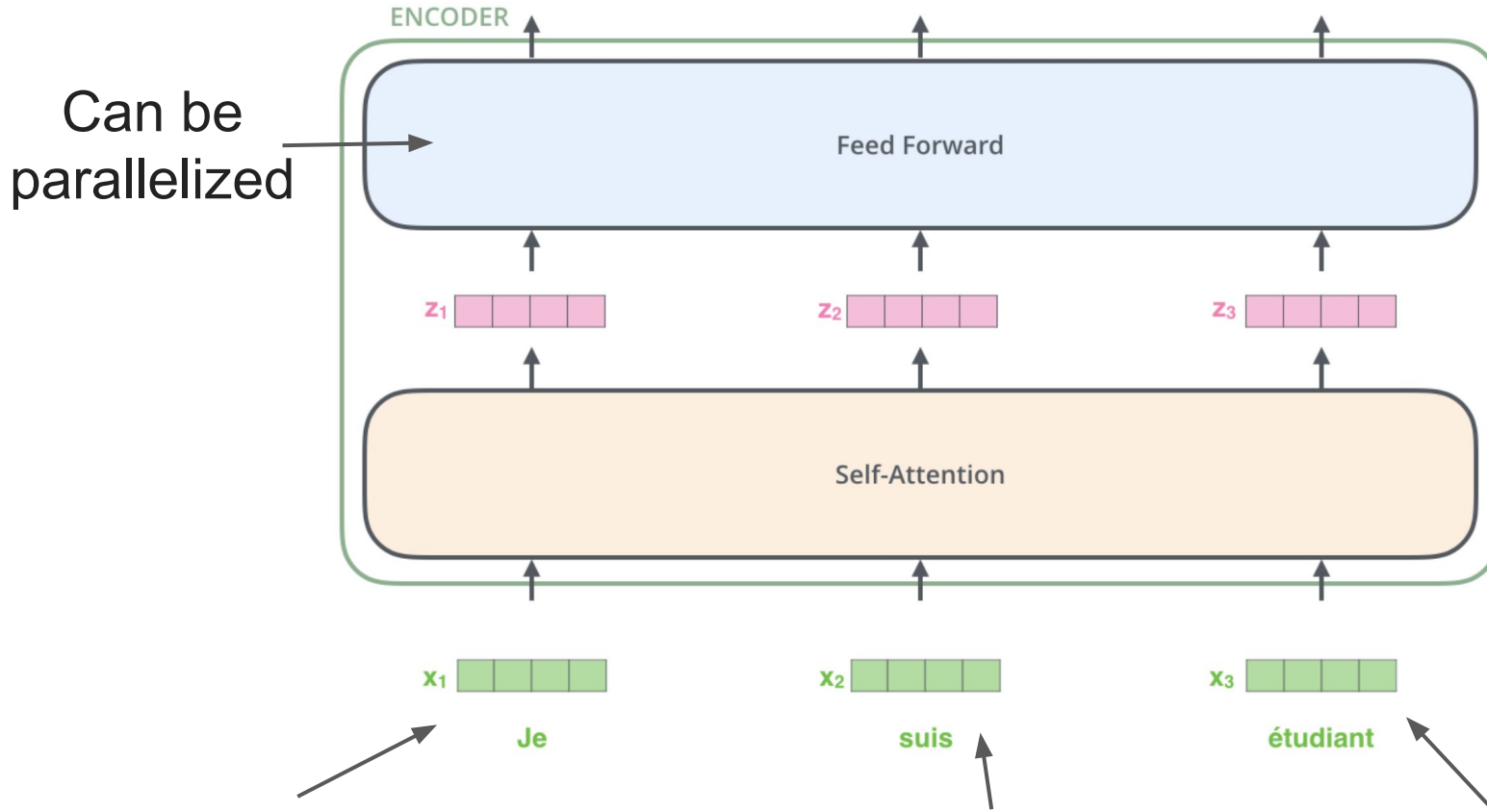


Layer: 5 Attention: Input - Input



Positional Encoding

The Encoder Side

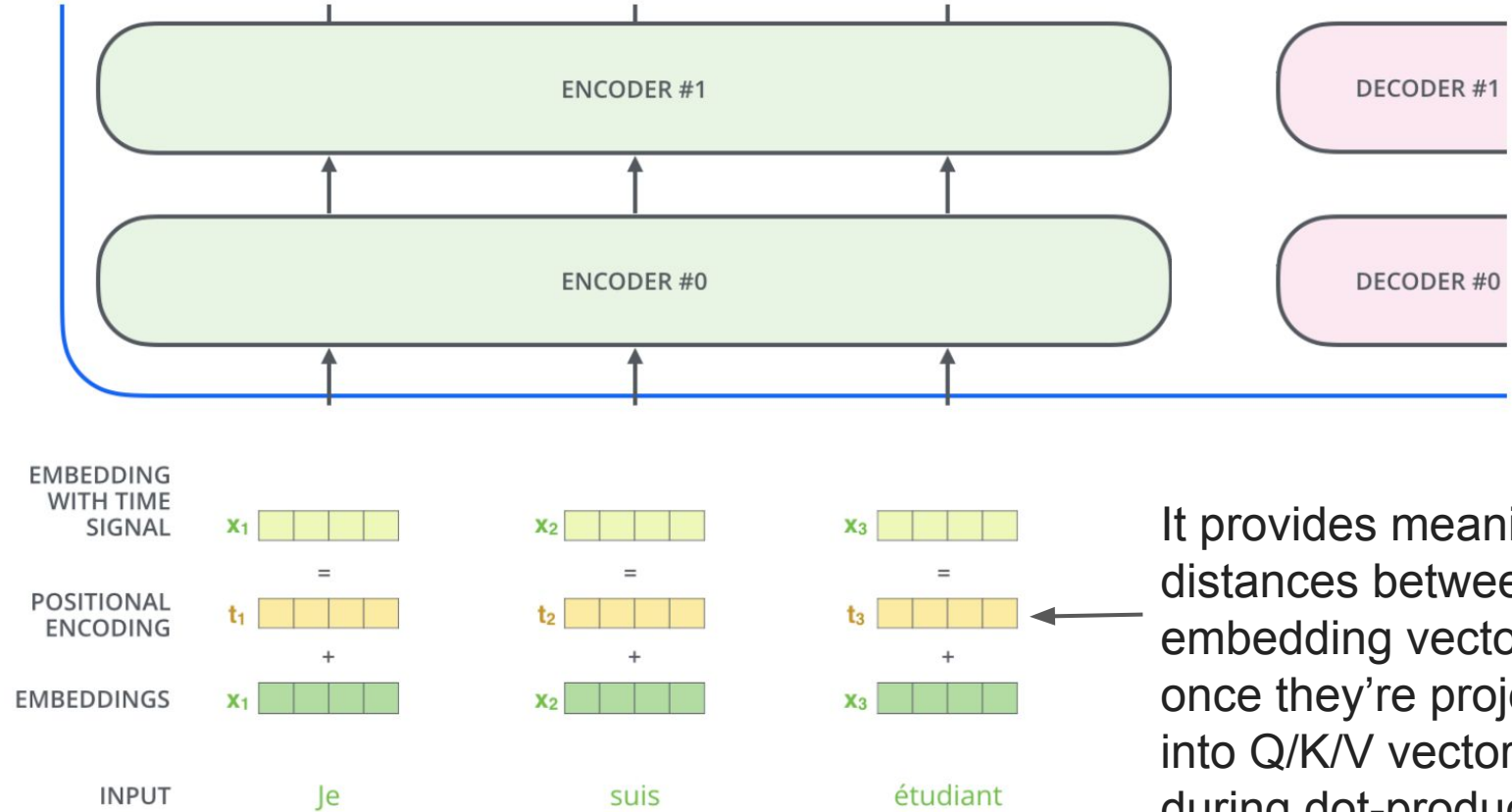


the word in each position flows through its own path in the encoder

Positional encoding requirements

- Positional encoding should be unique for every position in the sequence
- Distance between two same positions should be preserved with sequences of different length
- The positional encoding should be deterministic
- *It would be great if it would work with long sequences (longer than any sequence in the training set)*

Positional Encoding



Positional Encoding: why sin and cos?

$$\vec{p}_t^{(i)} = f(t)^{(i)} = \begin{cases} \sin(\omega_k t), & \text{if } i = 2k \\ \cos(\omega_k t), & \text{if } i = 2k + 1 \end{cases}$$
$$\omega_k = \frac{1}{10000^{2k/d}}$$
$$\vec{p}_t = \begin{bmatrix} \sin(\omega_1.t) \\ \cos(\omega_1.t) \\ \\ \sin(\omega_2.t) \\ \cos(\omega_2.t) \\ \\ \vdots \\ \\ \sin(\omega_{d/2}.t) \\ \cos(\omega_{d/2}.t) \end{bmatrix}_{d \times 1}$$

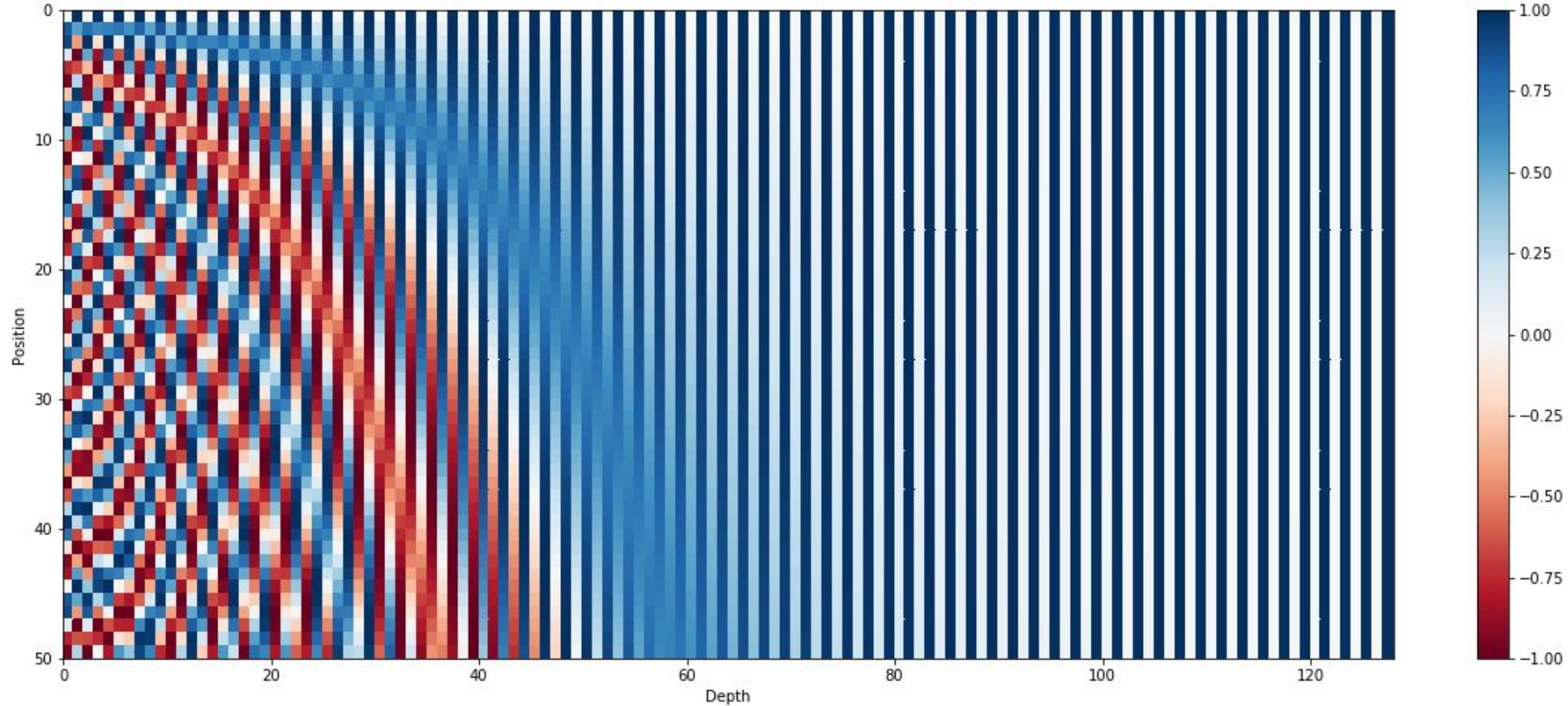
t stays for position in the original sequence

k is the index of the element in the positional vector

Positional Encoding

0 :	0	0	0	0	8 :	1	0	0	0
1 :	0	0	0	1	9 :	1	0	0	1
2 :	0	0	1	0	10 :	1	0	1	0
3 :	0	0	1	1	11 :	1	0	1	1
4 :	0	1	0	0	12 :	1	1	0	0
5 :	0	1	0	1	13 :	1	1	0	1
6 :	0	1	1	0	14 :	1	1	1	0
7 :	0	1	1	1	15 :	1	1	1	1

Positional Encoding



Positional Encoding: why sin and cos?

We chose this function because we hypothesized it would allow the model to easily learn to attend by relative positions, since for any fixed offset k , PE_{pos+k} can be represented as a linear function of PE_{pos} .

$$M \begin{bmatrix} \sin(\omega_k t) \\ \cos(\omega_k t) \end{bmatrix} = \begin{bmatrix} \sin(\omega_k (t + \phi)) \\ \cos(\omega_k (t + \phi)) \end{bmatrix}$$

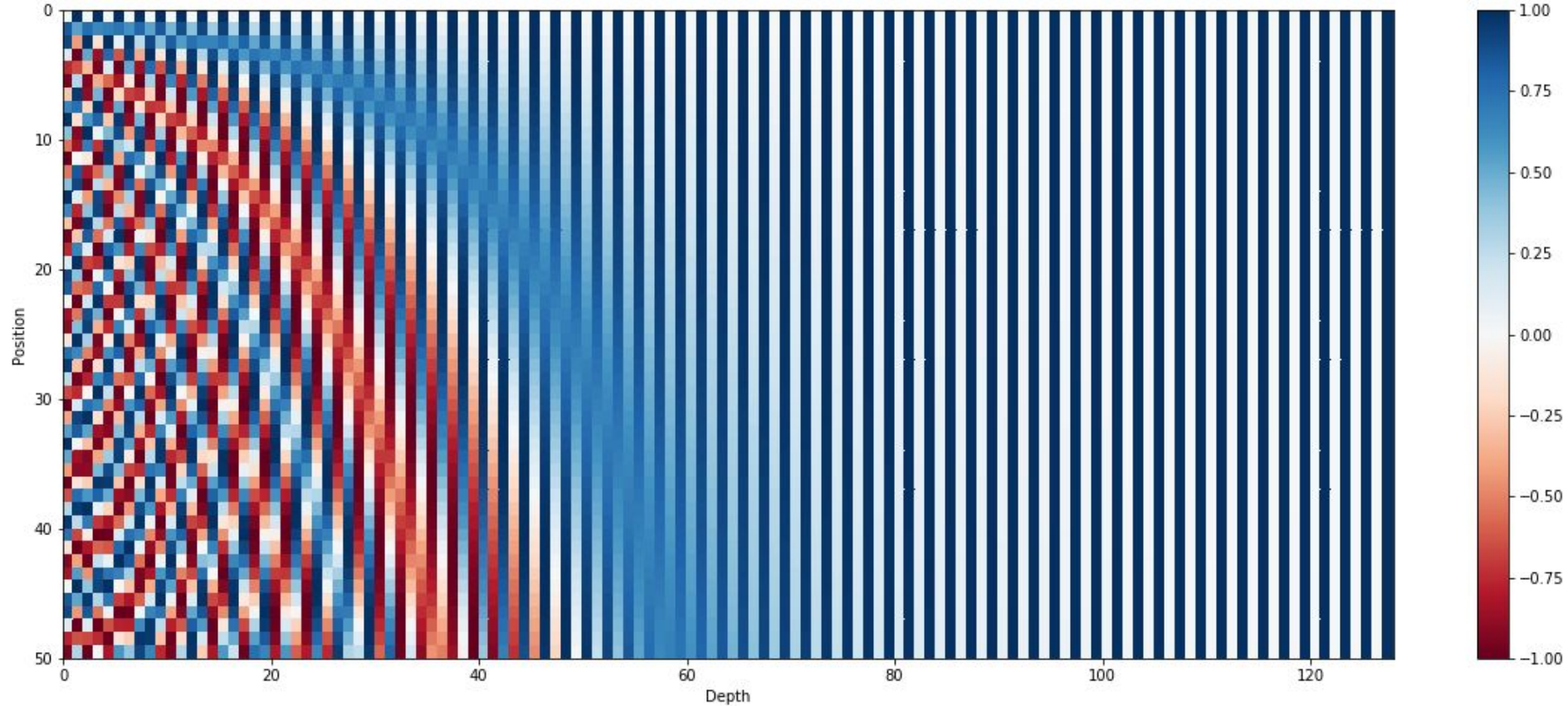
Positional Encoding: why sin and cos?

$$\begin{bmatrix} u_1 & v_1 \\ u_2 & v_2 \end{bmatrix} \begin{bmatrix} \sin(\omega_k t) \\ \cos(\omega_k t) \end{bmatrix} = \begin{bmatrix} \sin(\omega_k(t + \phi)) \\ \cos(\omega_k(t + \phi)) \end{bmatrix}$$

$$\begin{bmatrix} u_1 & v_1 \\ u_2 & v_2 \end{bmatrix} \begin{bmatrix} \sin(\omega_k t) \\ \cos(\omega_k t) \end{bmatrix} = \begin{bmatrix} \sin(\omega_k t) \cos(\omega_k \phi) + \cos(\omega_k t) \sin(\omega_k \phi) \\ \cos(\omega_k t) \cos(\omega_k \phi) - \sin(\omega_k t) \sin(\omega_k \phi) \end{bmatrix}$$

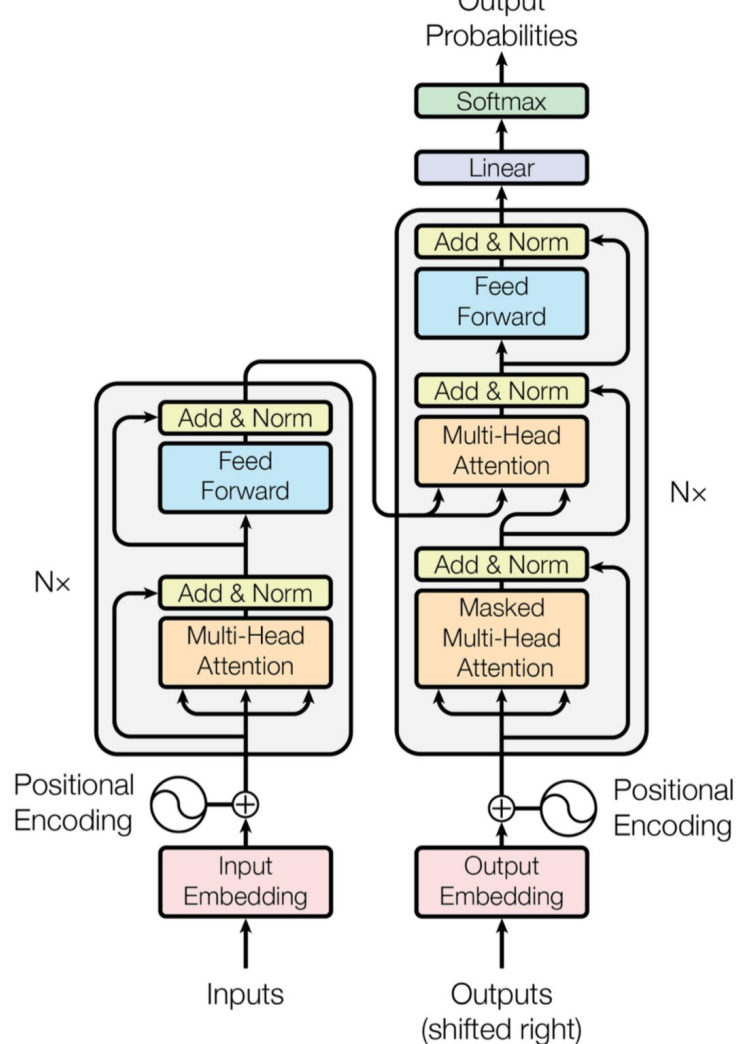
$$M_{\phi, k} = \begin{bmatrix} \cos(\omega_k \phi) & \sin(\omega_k \phi) \\ -\sin(\omega_k \phi) & \cos(\omega_k \phi) \end{bmatrix}$$

Positional Encoding

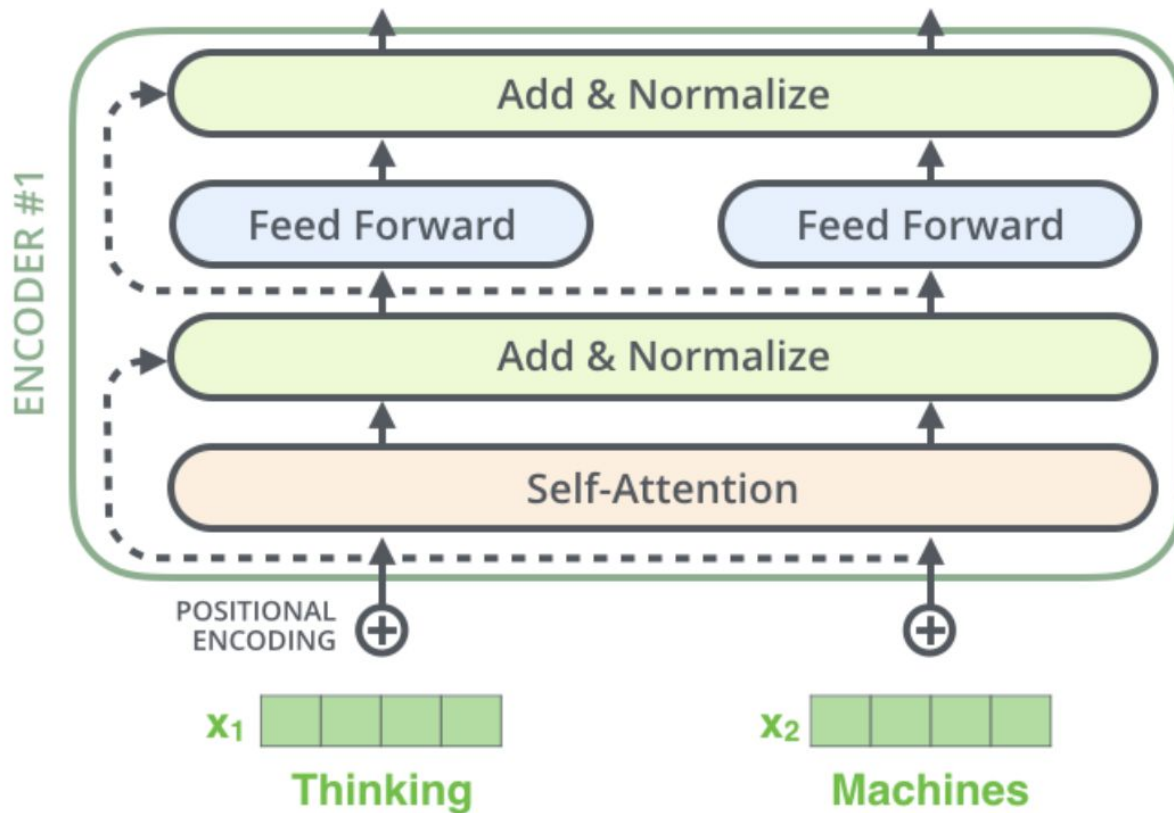


Layer Normalization

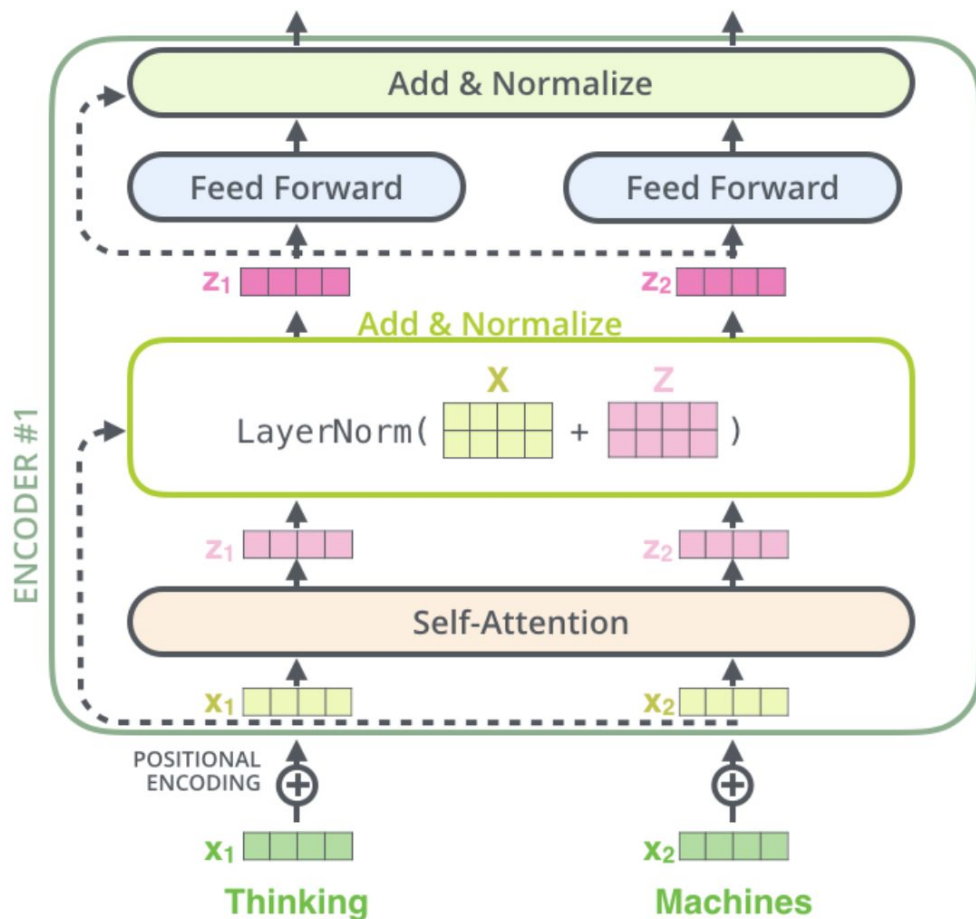
The Transformer: recap



Layer Normalization



Layer Normalization



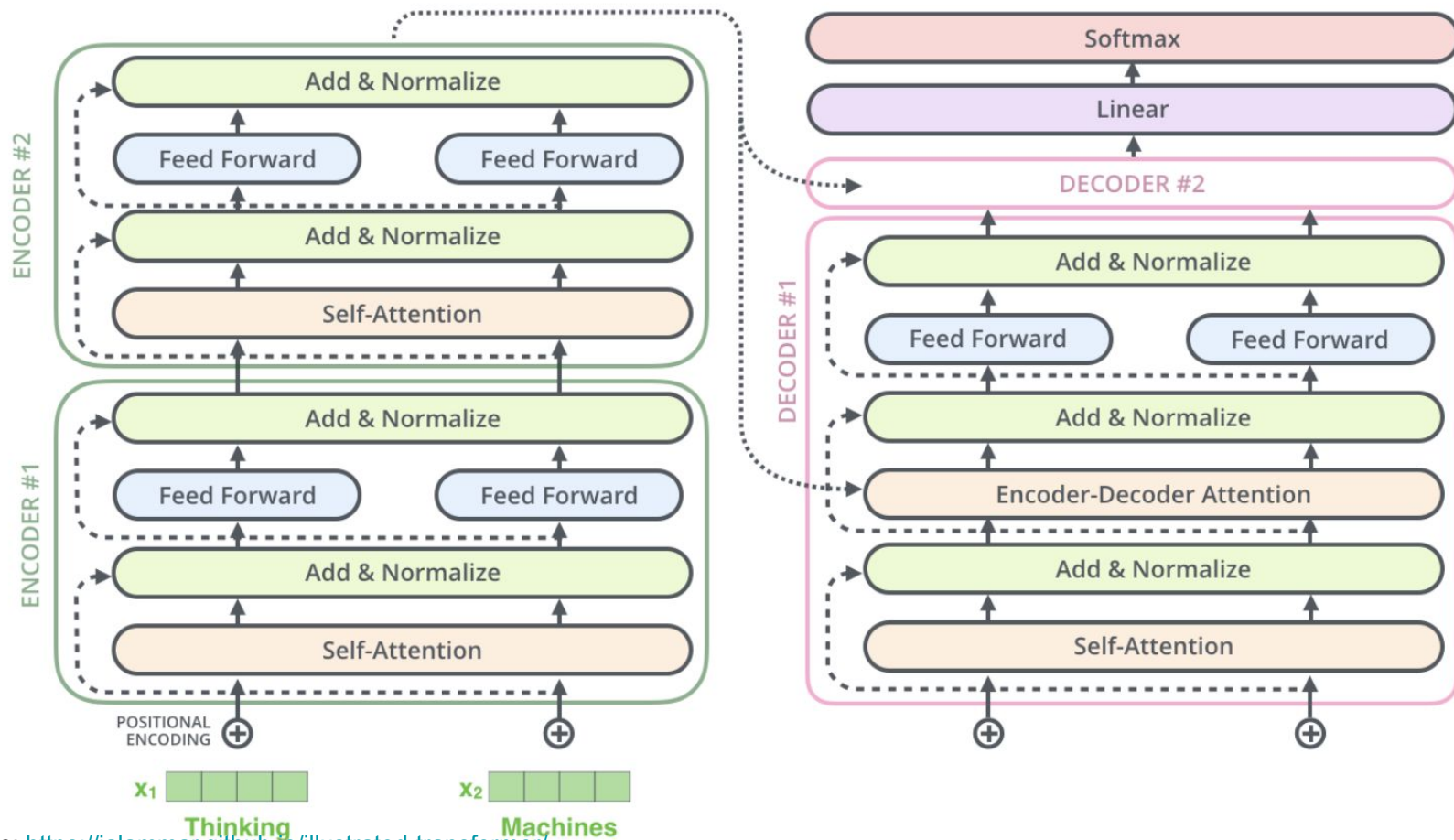
Like BatchNorm

but normalize along
all features
representing latent
vector

More info:

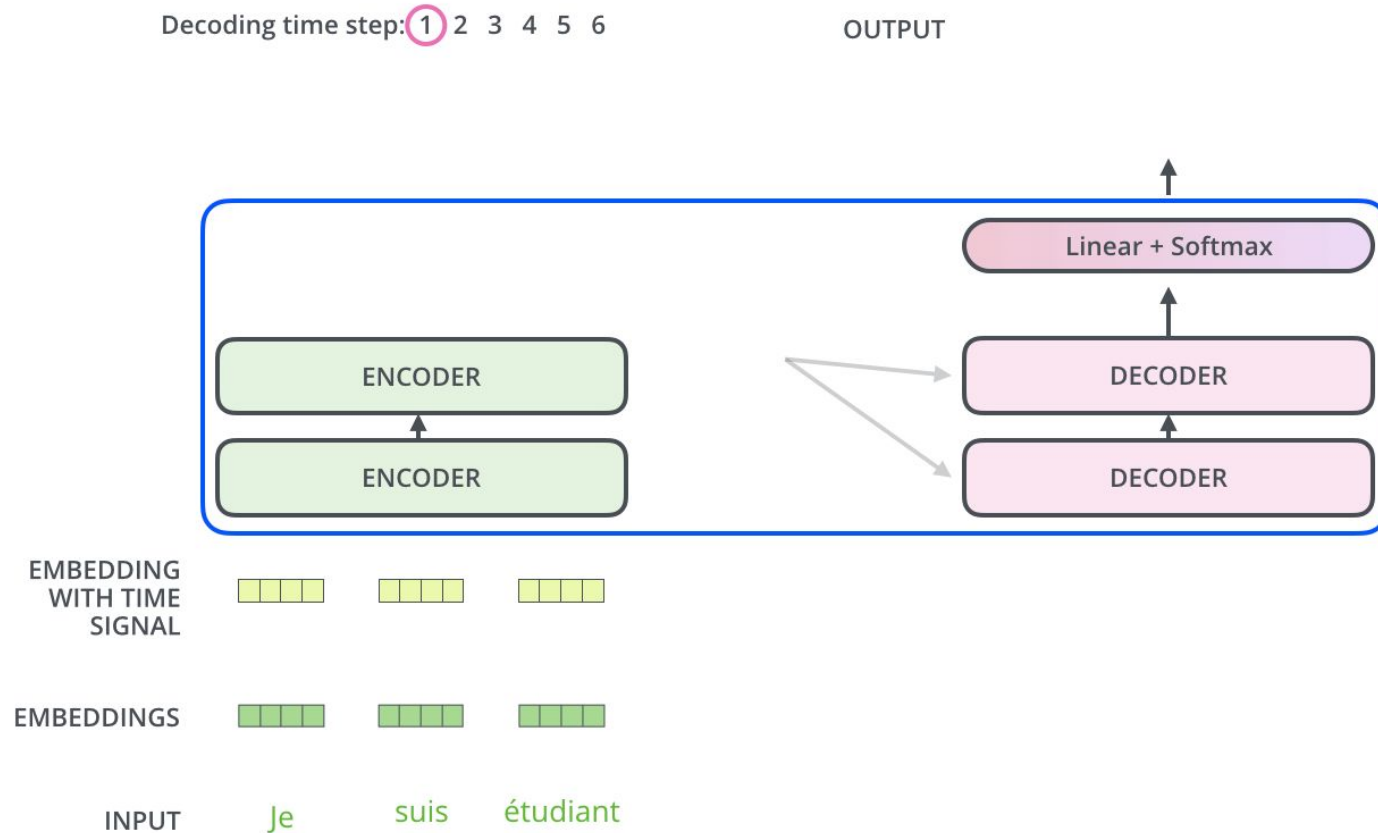
[Layer Normalization](https://jalammar.github.io/illustrated-transformer/)

Layer Normalization

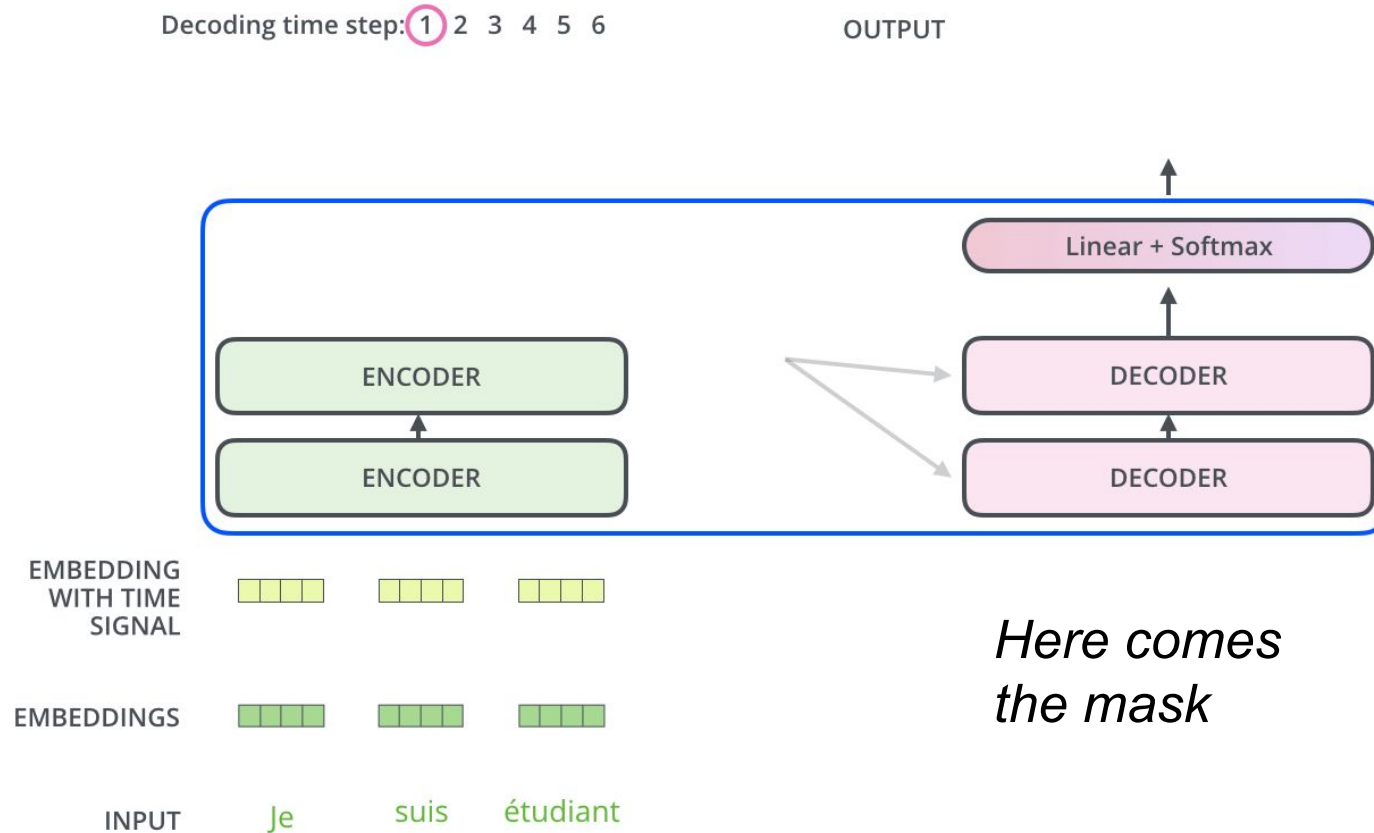


The Decoder

The Decoder Side



The Decoder Side

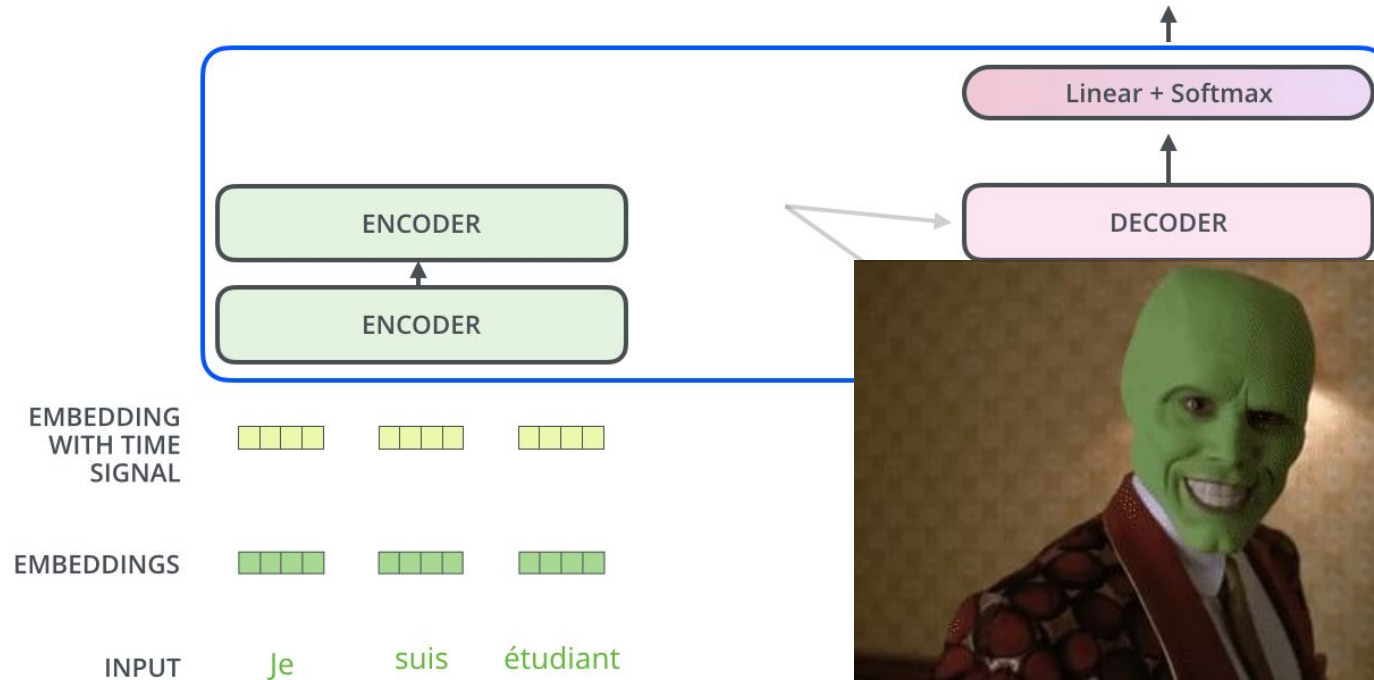


*Here comes
the mask*

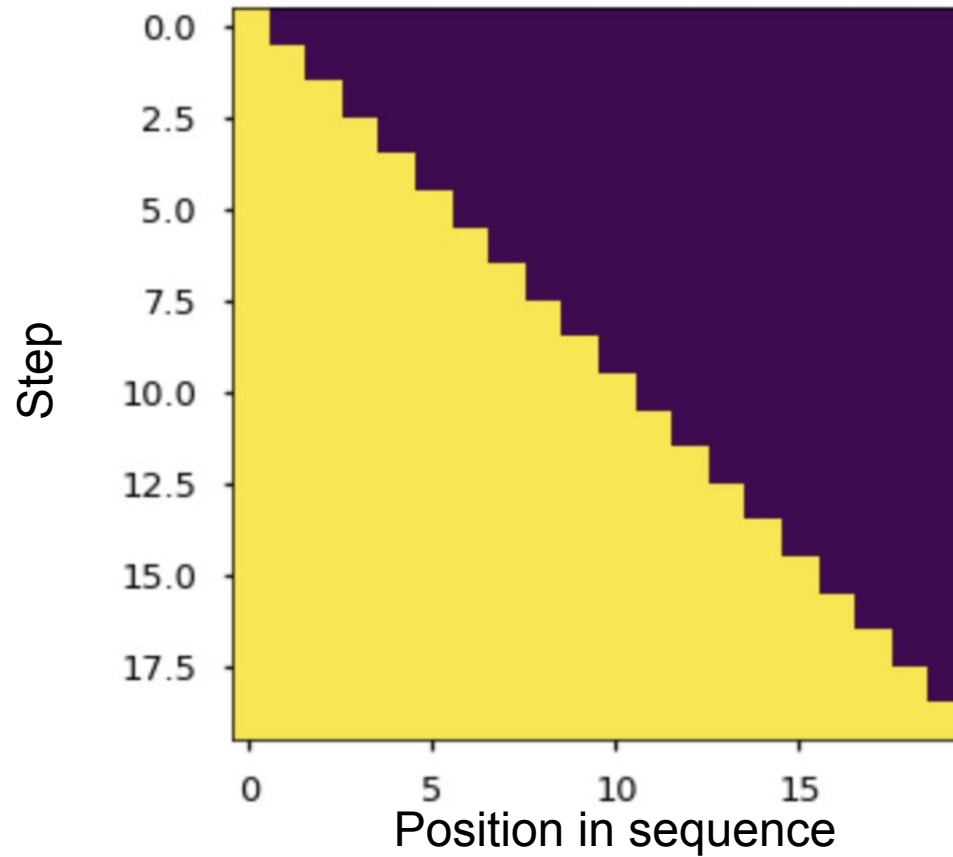
The Decoder Side

Decoding time step: 1 2 3 4 5 6

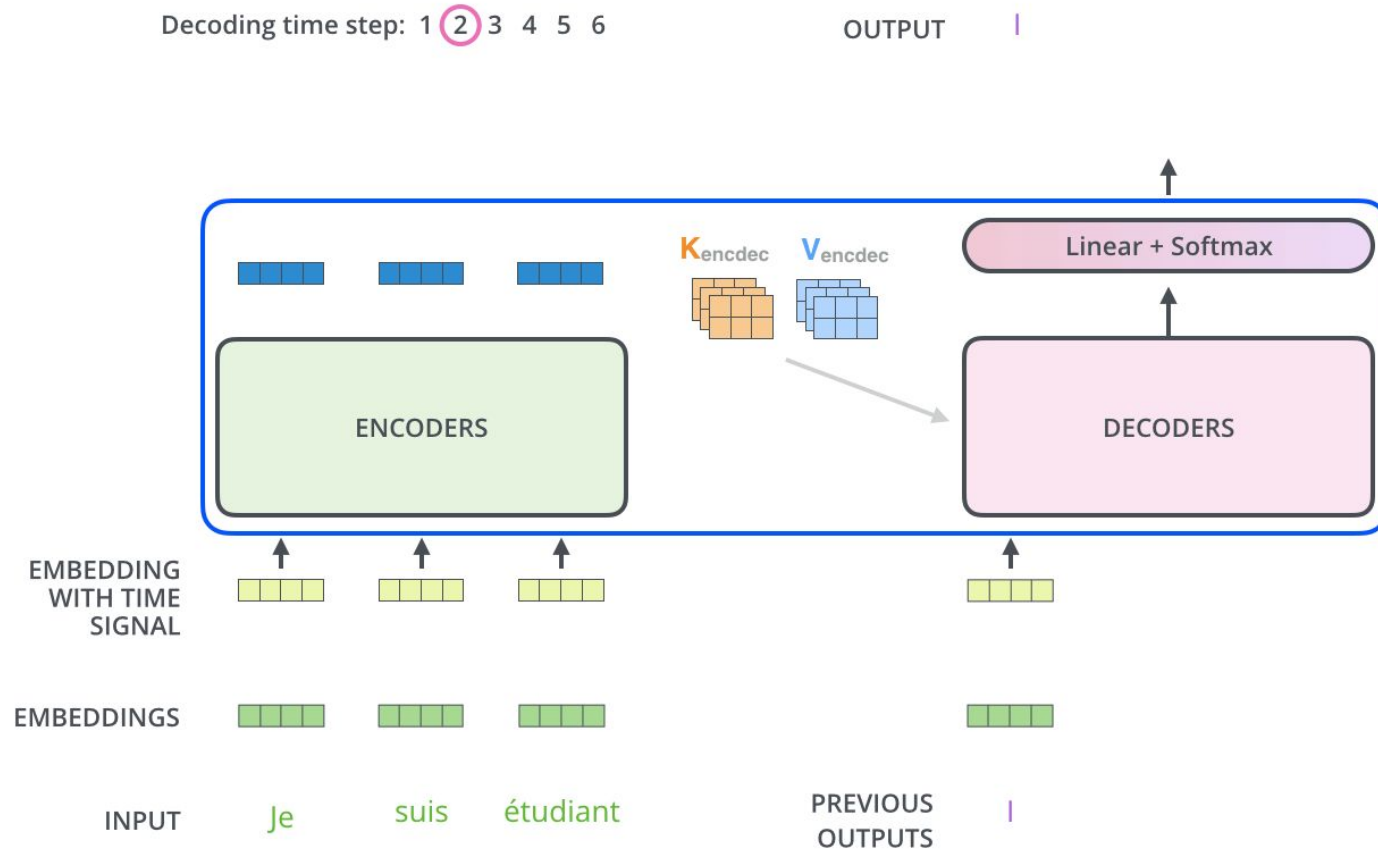
OUTPUT



The masked decoder input



The Decoder Side

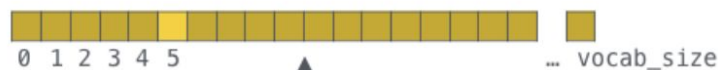


Final Linear and Softmax Layer

Which word in our vocabulary
is associated with this index?

Get the index of the cell
with the highest value
(argmax)

log_probs

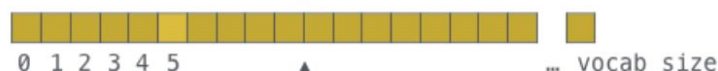


am

5

Softmax

logits

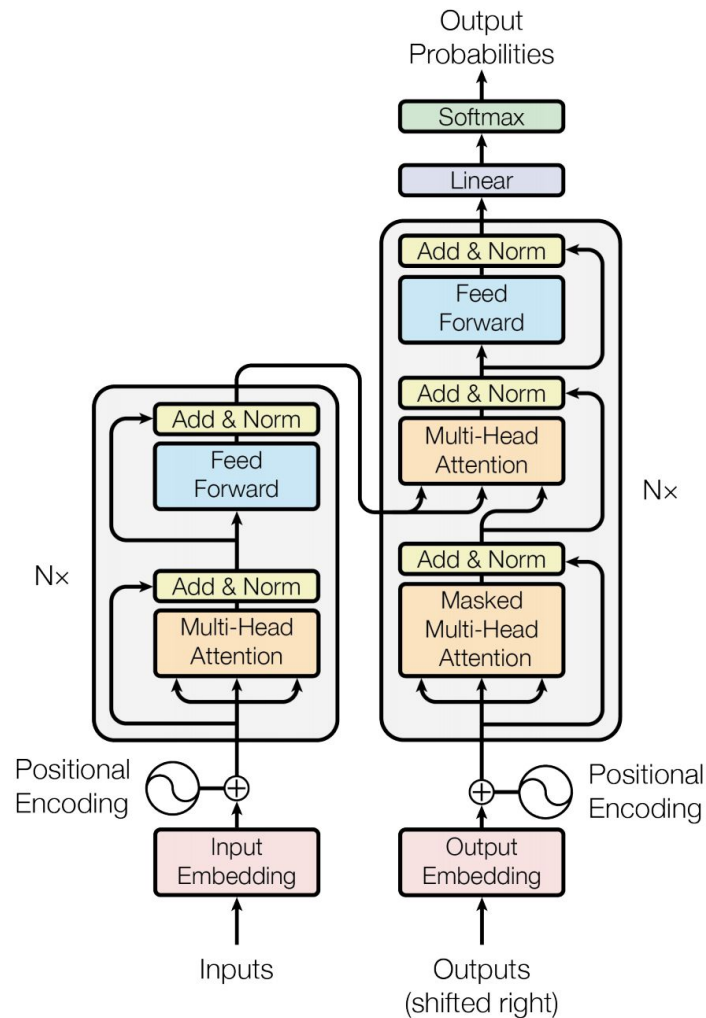


Linear

Decoder stack output



The Transformer



- Transformer is novel and very powerful architecture
 - It is worth it to understand how Self-Attention works
 - Physical analogues can help you
-
- Further readings are available in the repo