NLP course

Lecture 06: Transformer overview & Positional encoding

Radoslav Neychev

MADE, Moscow

14.04.2021

Outline

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

Based on: http://web.stanford.edu/class/cs224n/slides/cs224n-2019-lecture08-nmt.pdf
https://jalammar.github.io/illustrated-transformer/

The Transformer

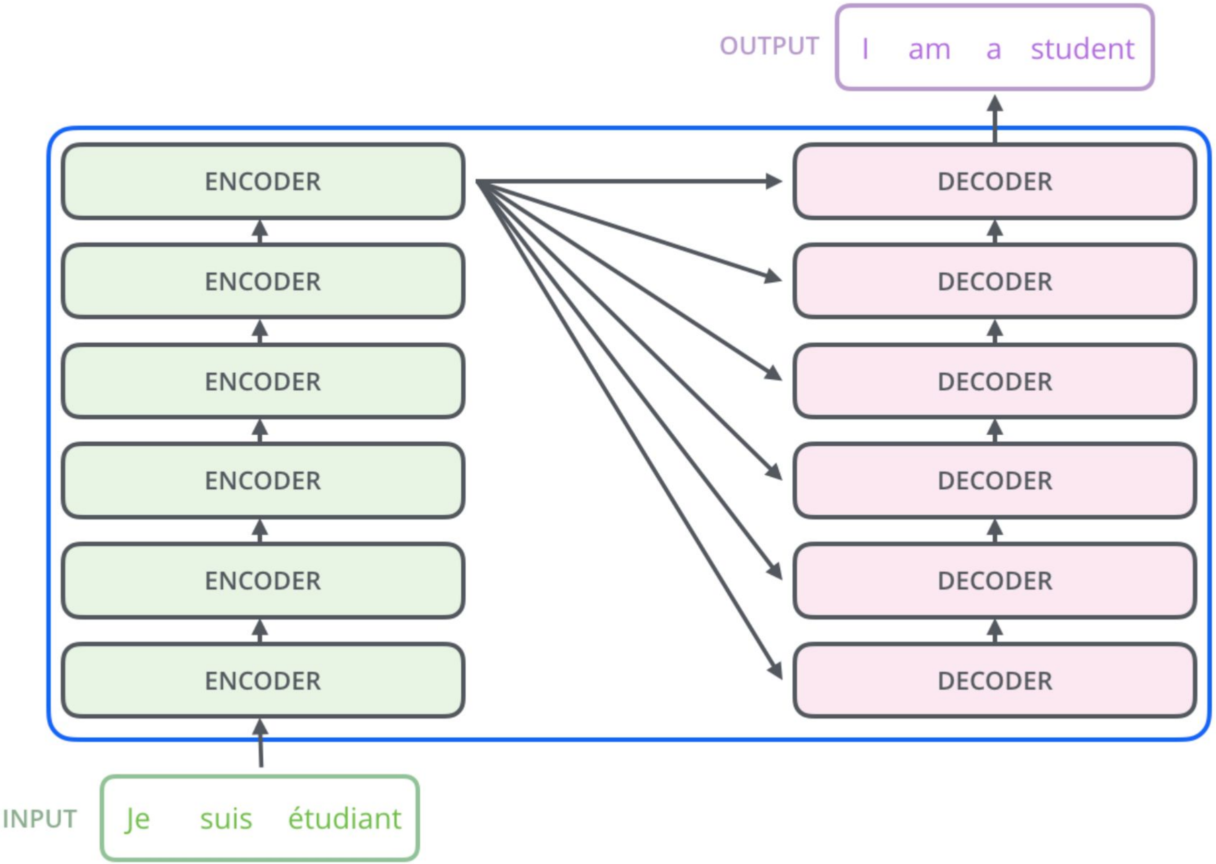
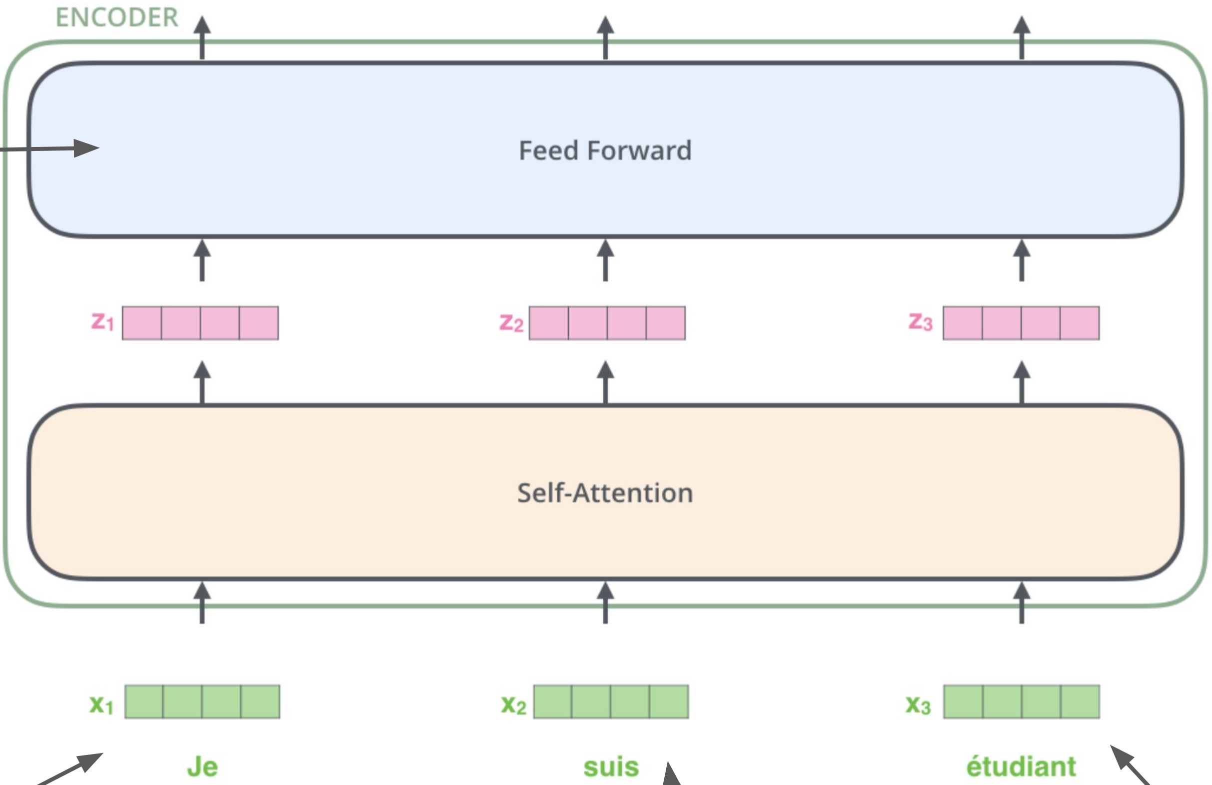


Image source: https://jalammar.github.io/illustrated-transformer/

The Encoder Side

Can be parallelized

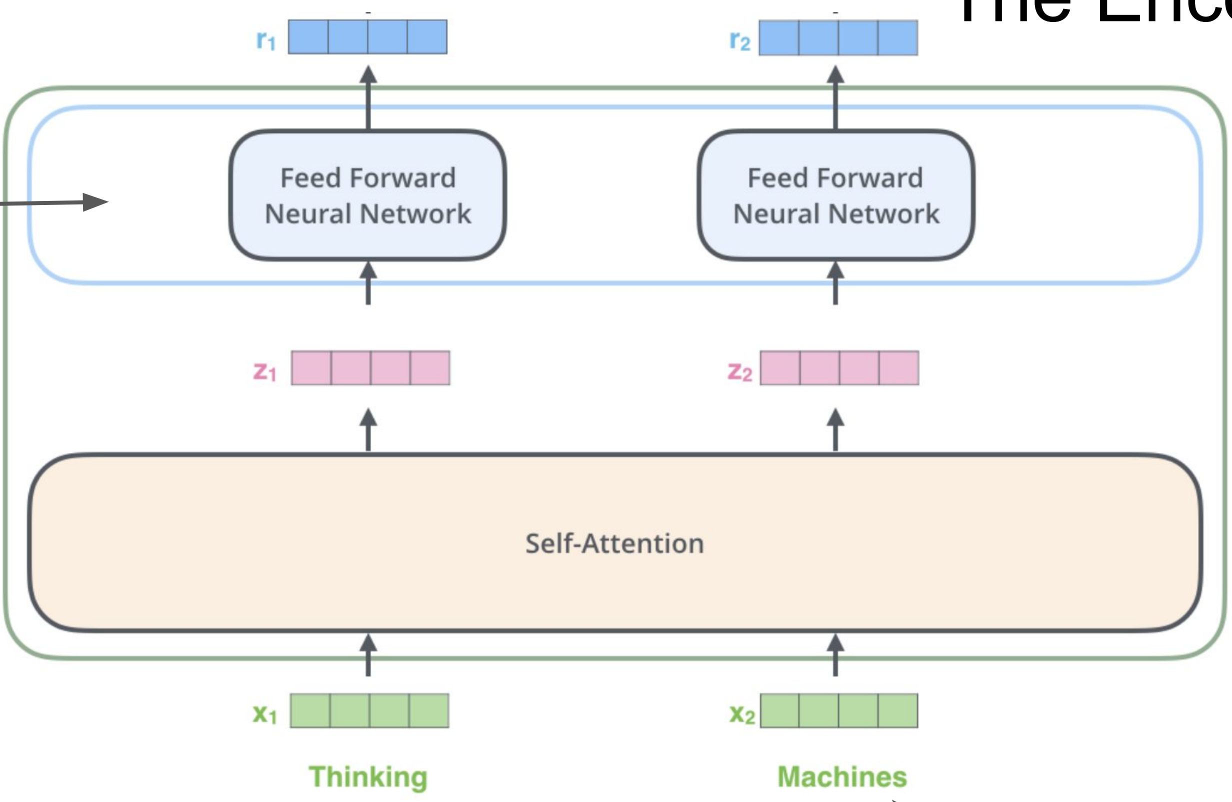


the word in each position flows through its own path in the encoder

Image source: https://jalammar.github.io/illustrated-transformer/

The Encoder Side

Can be parallelized



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

Image source: https://jalammar.github.io/illustrated-transformer/

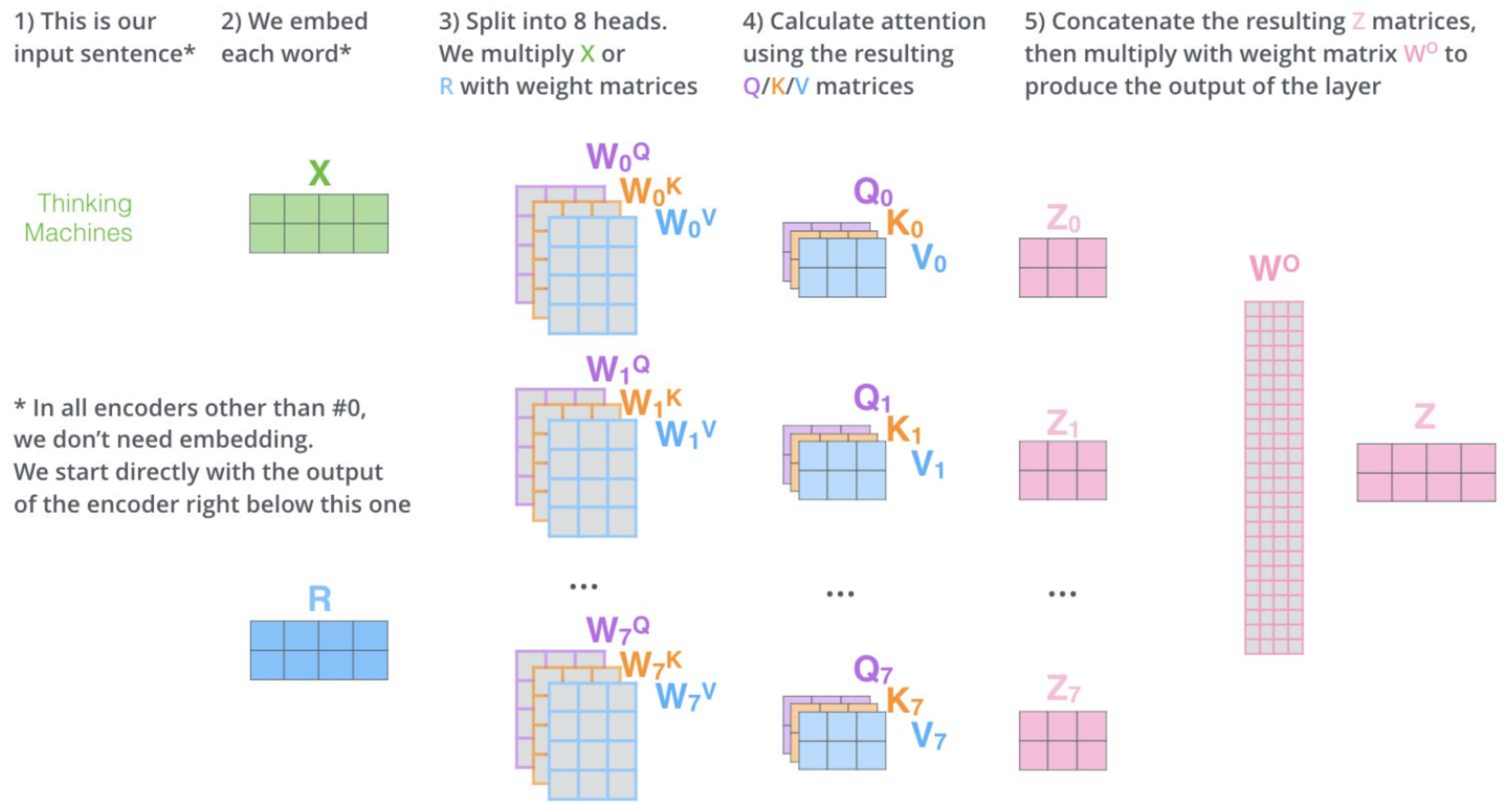
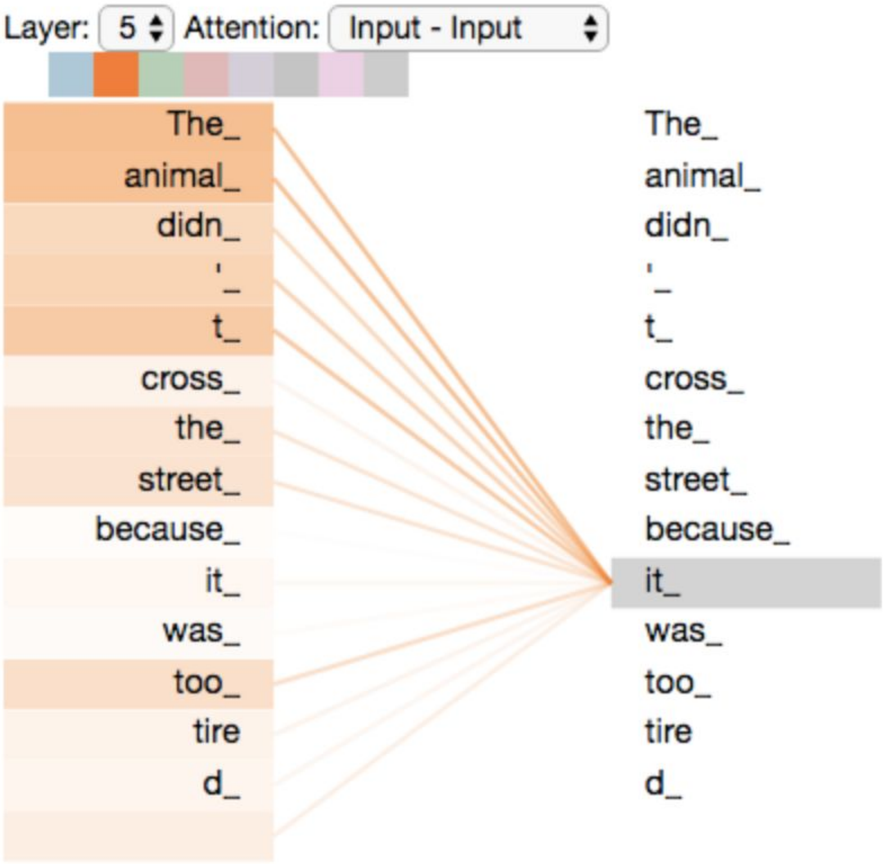
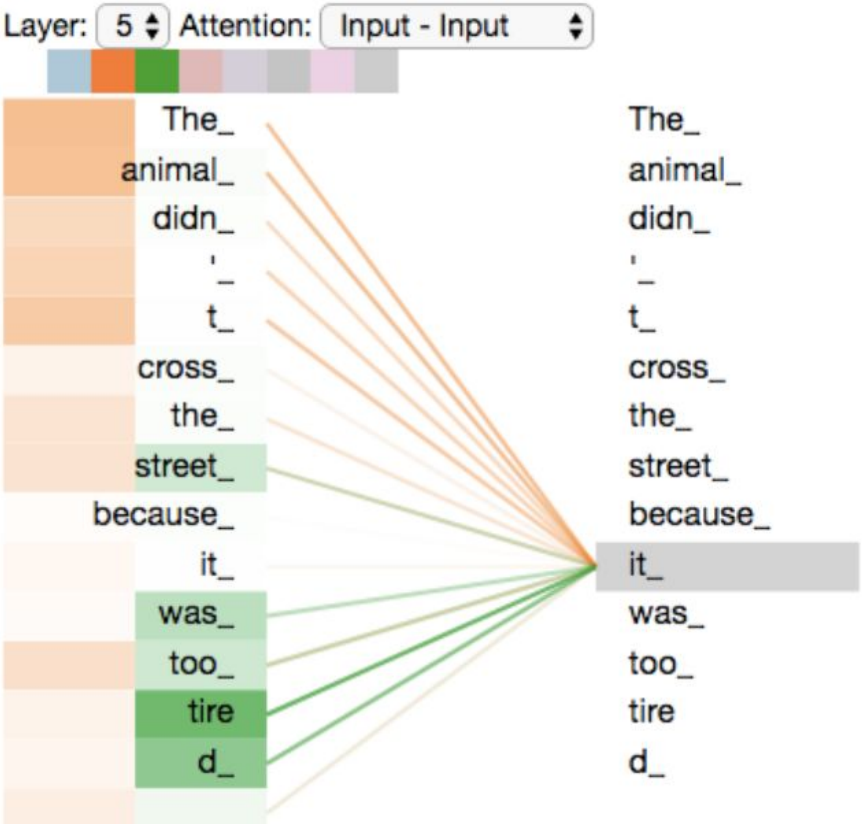


Image source: https://jalammar.github.io/illustrated-transformer/

Multi-Head Attention

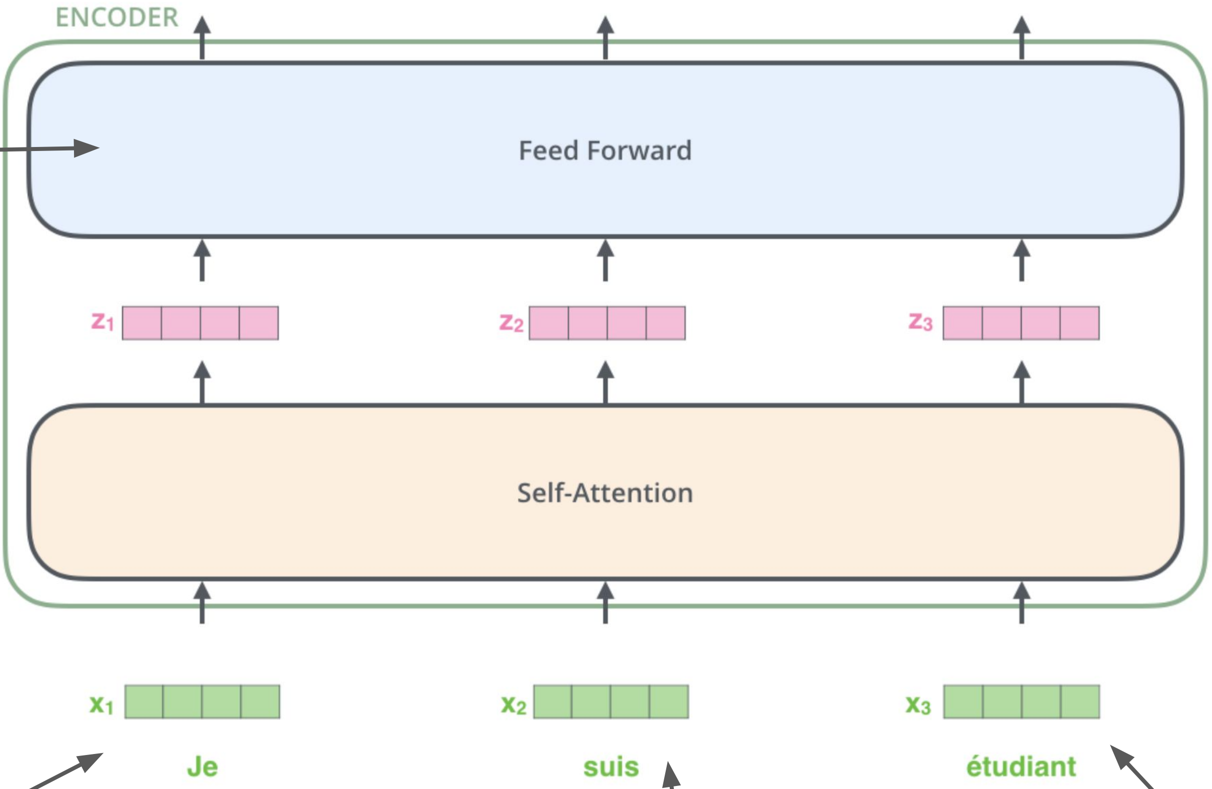




Positional Encoding

The Encoder Side

Can be parallelized



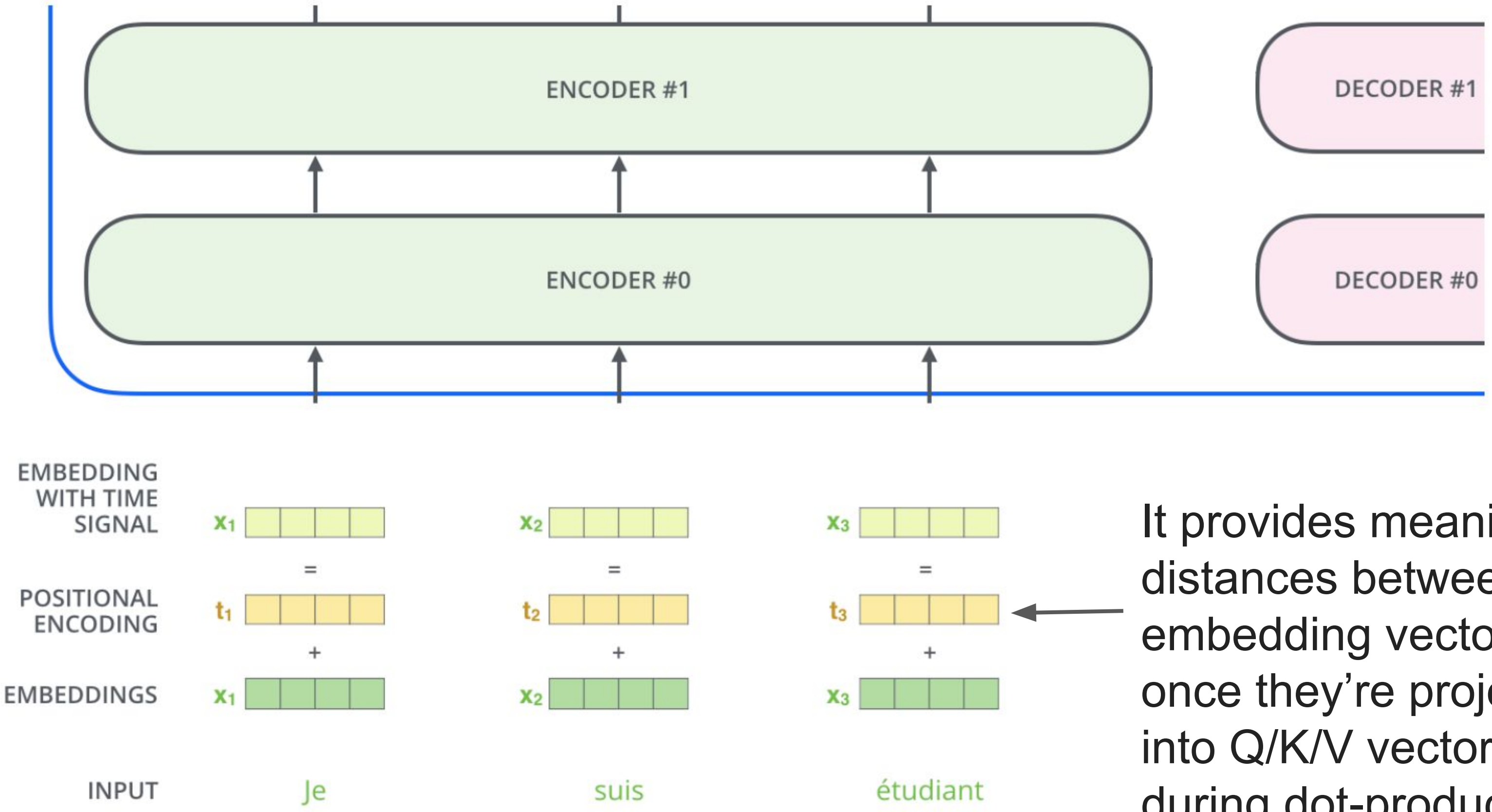
the word in each position flows through its own path in the encoder

Image source: https://jalammar.github.io/illustrated-transformer/

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



It provides meaningful distances between the embedding vectors once they're projected into Q/K/V vectors and during dot-product attention

Image source: https://jalammar.github.io/illustrated-transformer/

Positional Encoding: why sin and cos?

$$\vec{p_t}^{(i)} = f(t)^{(i)} \quad = \begin{cases} \sin(\omega_k t), & \text{if } i = 2k \\ \cos(\omega_k t), & \text{if } i = 2k+1 \end{cases}$$

$$\omega_k = \frac{1}{10000^{2k/d}}$$

$$\vec{p_t} = \begin{bmatrix} \sin(\omega_1 . t) \\ \cos(\omega_1 . t) \\ \\ \sin(\omega_2 . t) \\ \cos(\omega_2 . t) \\ \\ \vdots \\ \\ \sin(\omega_{d/2} . t) \\ \cos(\omega_{d/2} . t) \end{bmatrix}_{d \times 1}$$

t stays for position in the original sequence
k is the index of the element in the positional vector

Image source: https://kazemnejad.com/blog/transformer_architecture_positional_encoding/

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

Image source: https://kazemnejad.com/blog/transformer_architecture_positional_encoding/

Positional Encoding

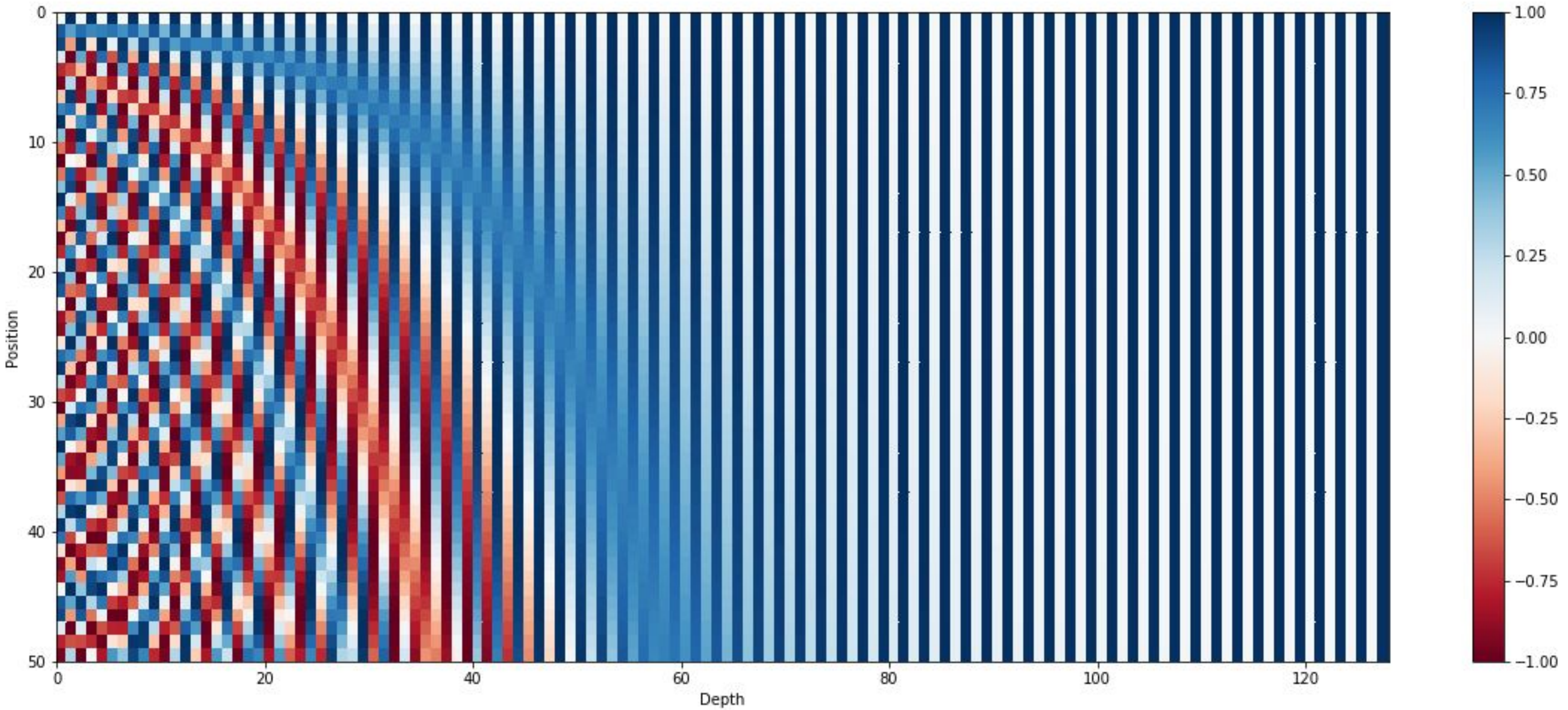


Image source: https://kazemnejad.com/blog/transformer_architecture_positional_encoding/

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, PEpos+k can be represented as a linear function of PEpos.

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

Image source: https://kazemnejad.com/blog/transformer_architecture_positional_encoding/

Positional Encoding: why sin and cos?

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

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

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

Image source: https://kazemnejad.com/blog/transformer_architecture_positional_encoding/

Positional Encoding

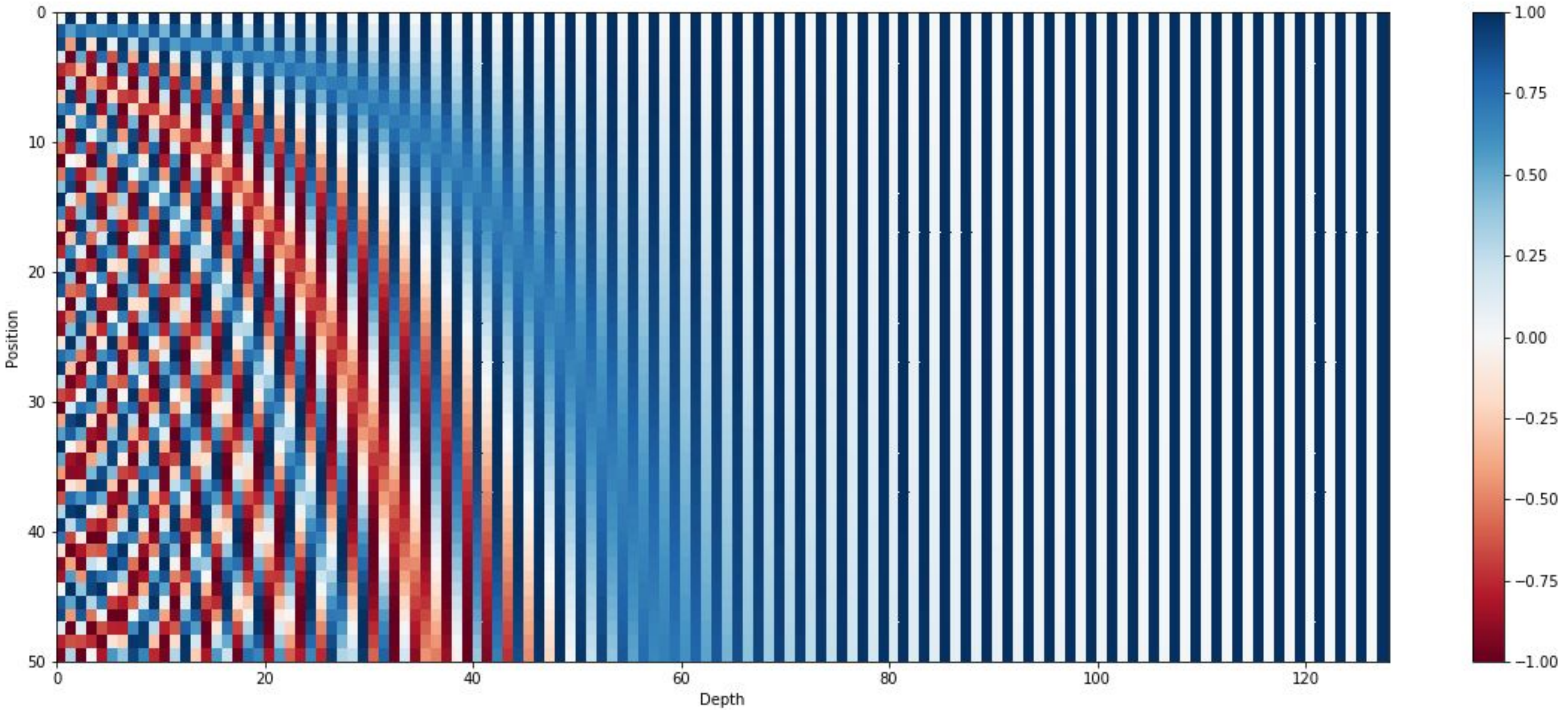


Image source: https://kazemnejad.com/blog/transformer_architecture_positional_encoding/

Layer Normalization

The Transformer: recap

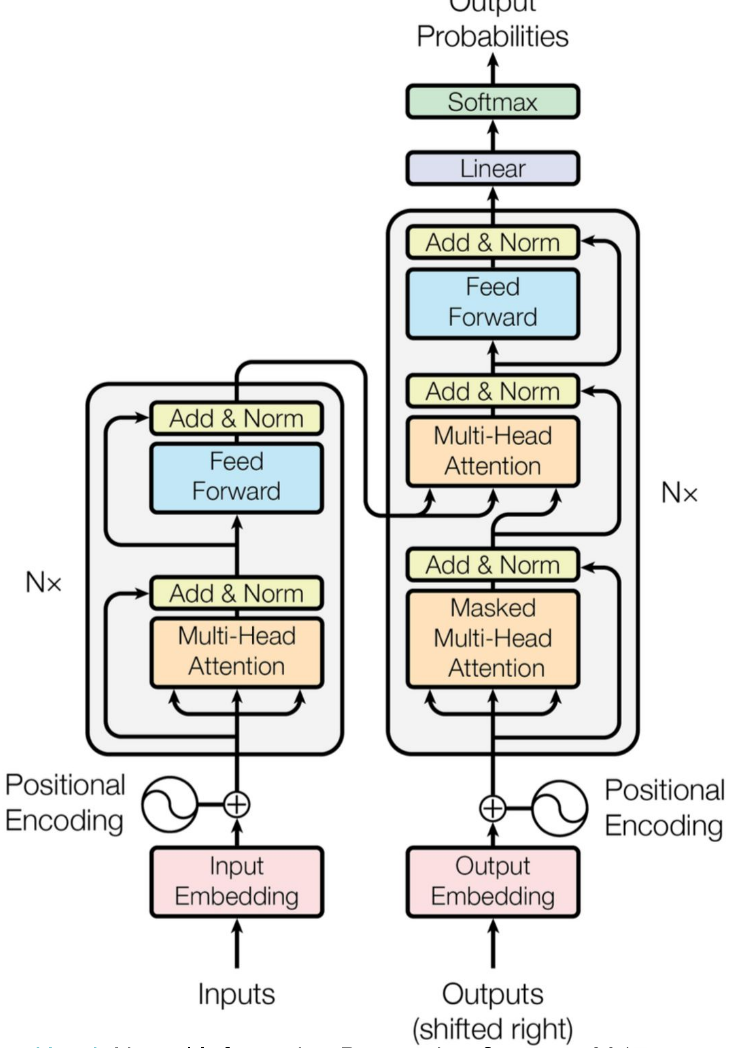


Image source: [Attention Is All You Need](), Neural Information Processing Systems 2017

Layer Normalization

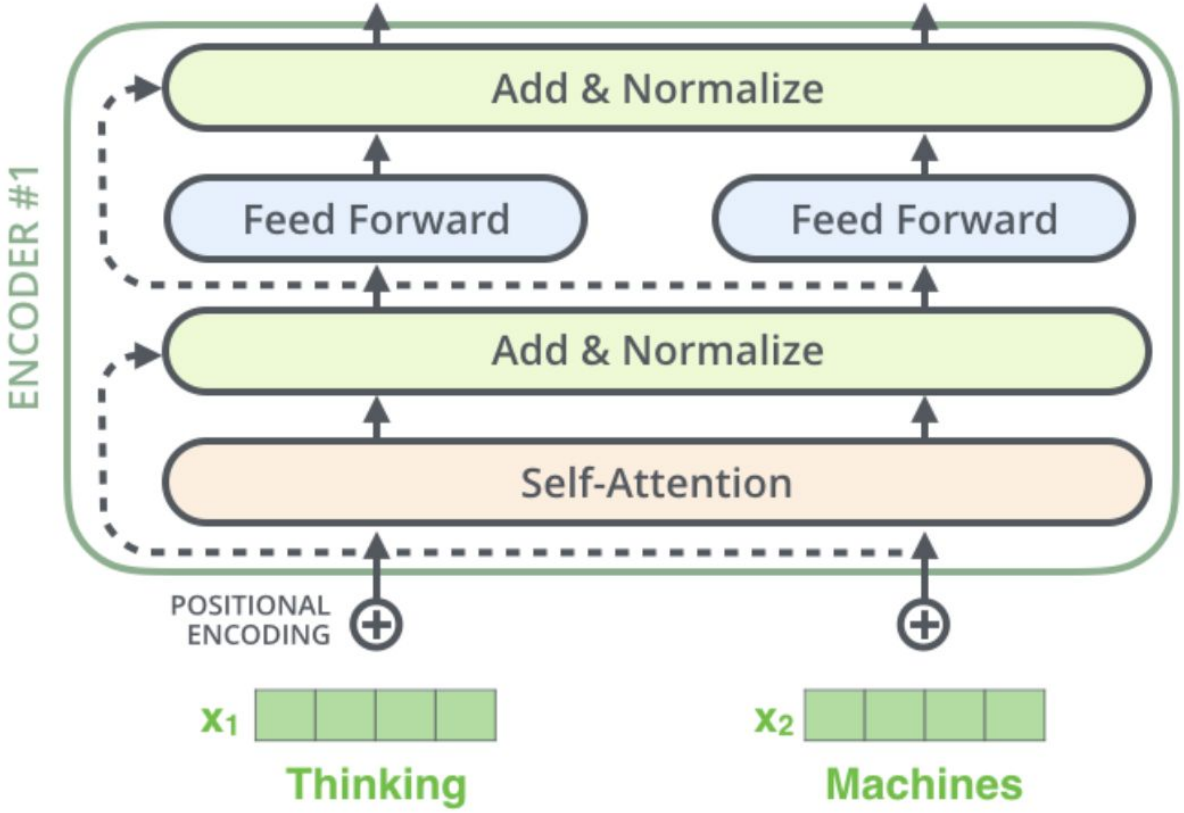
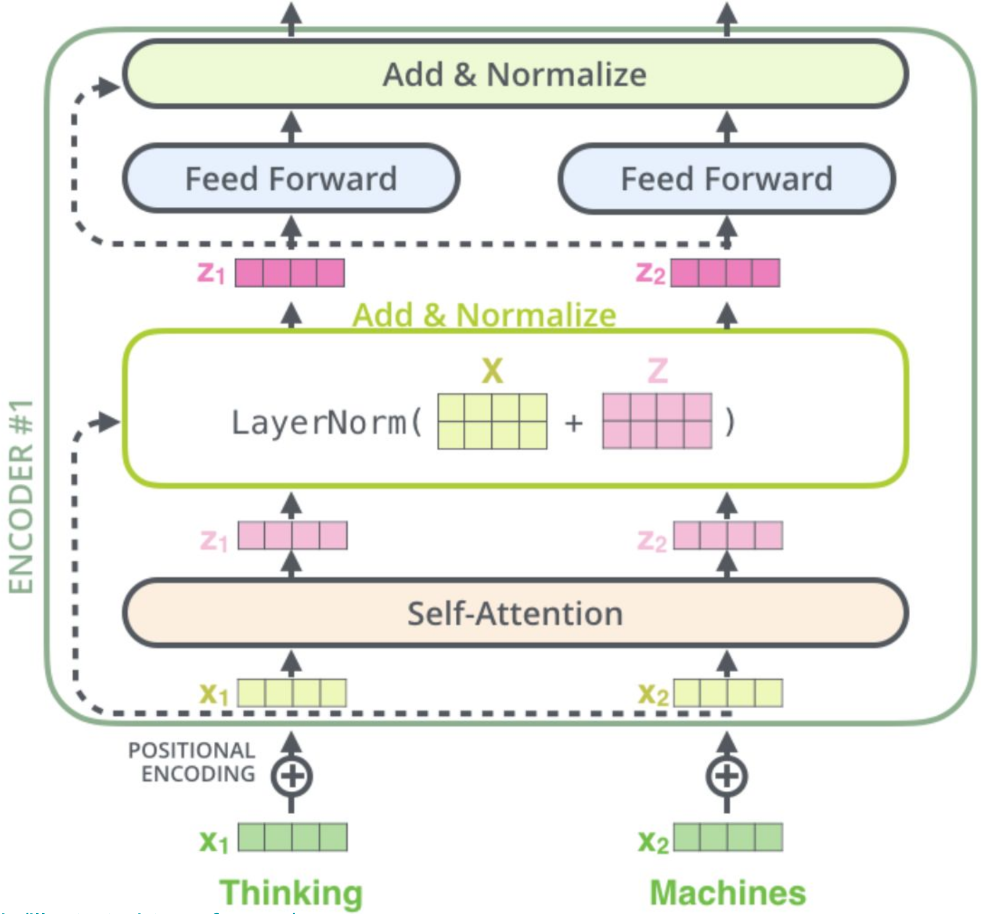


Image source: https://jalammar.github.io/illustrated-transformer/

Layer Normalization

Like BatchNorm

but normalize along all features representing latent vector



More info:
[Layer Normalization]()

Image source: https://jalammar.github.io/illustrated-transformer/

Layer Normalization

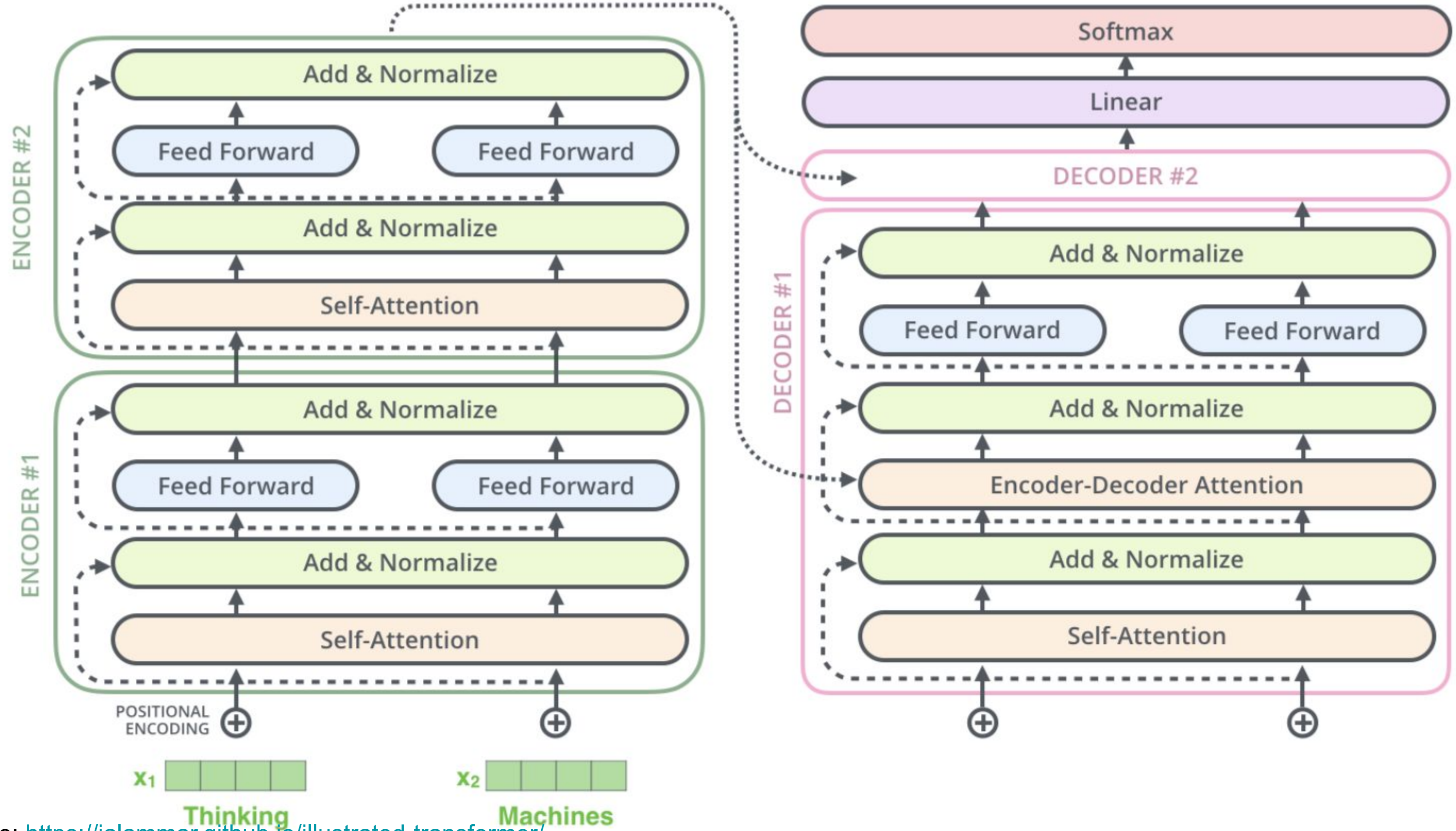


Image source: https://jalammar.github.io/illustrated-transformer/

The Decoder

The Decoder Side

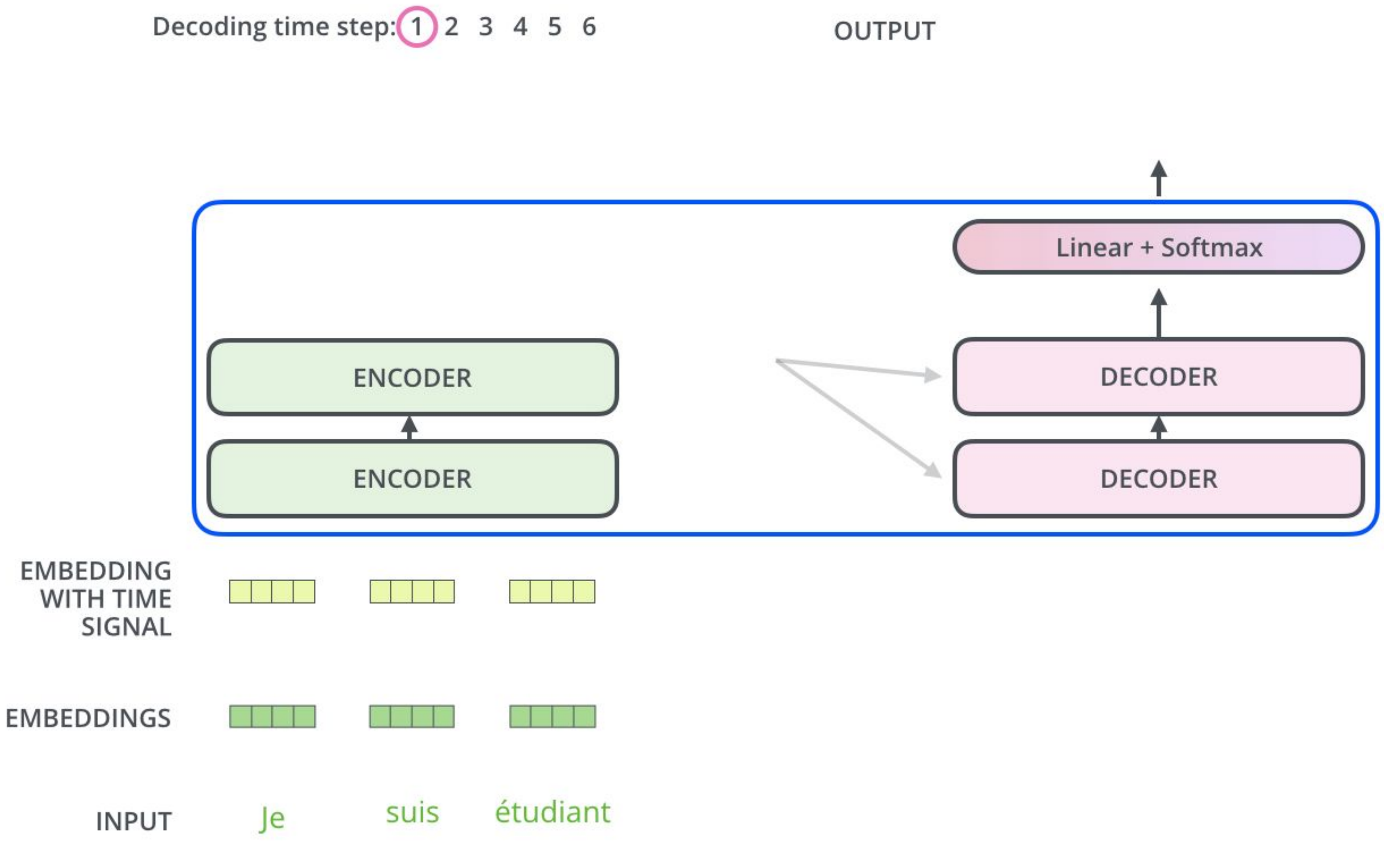
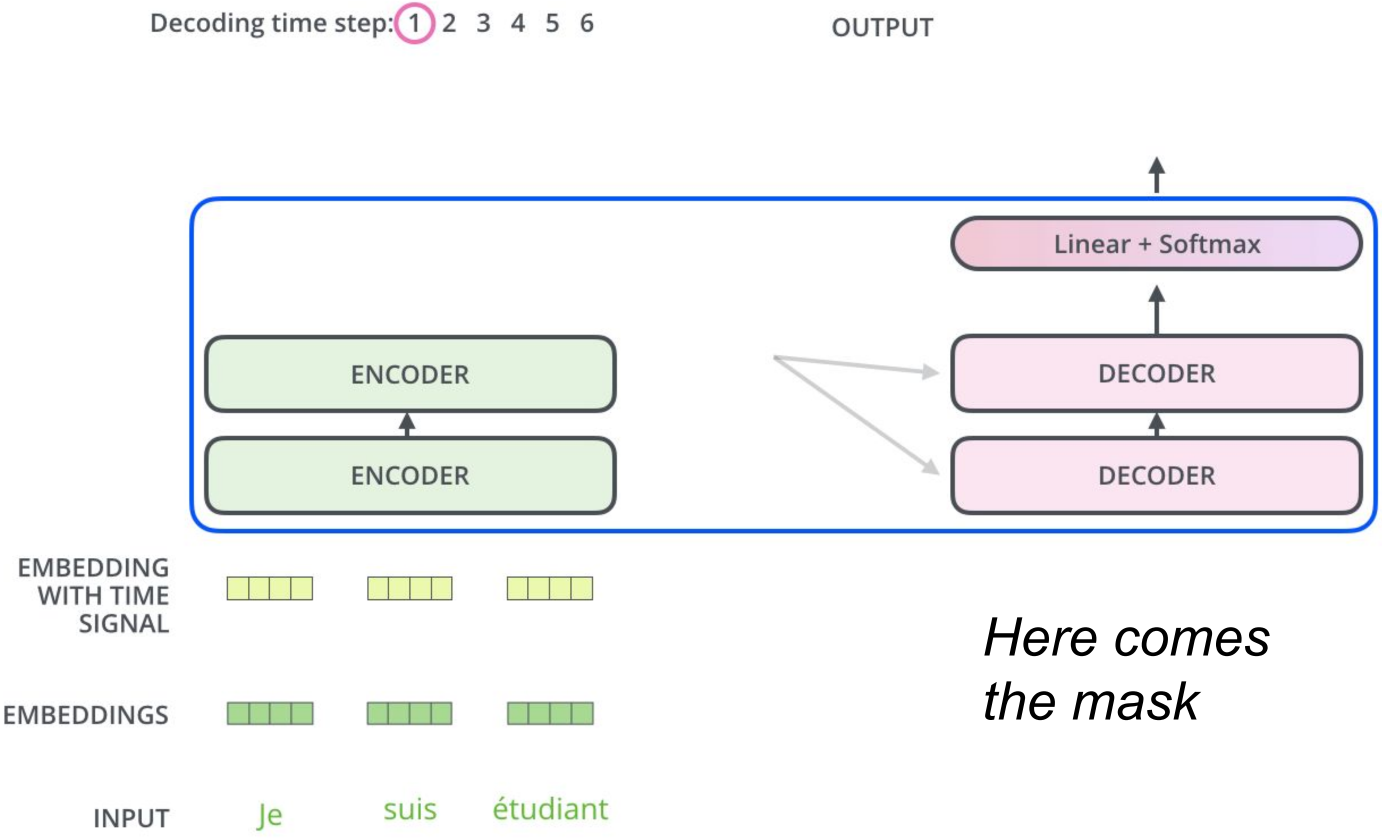


Image source: https://jalammar.github.io/illustrated-transformer/

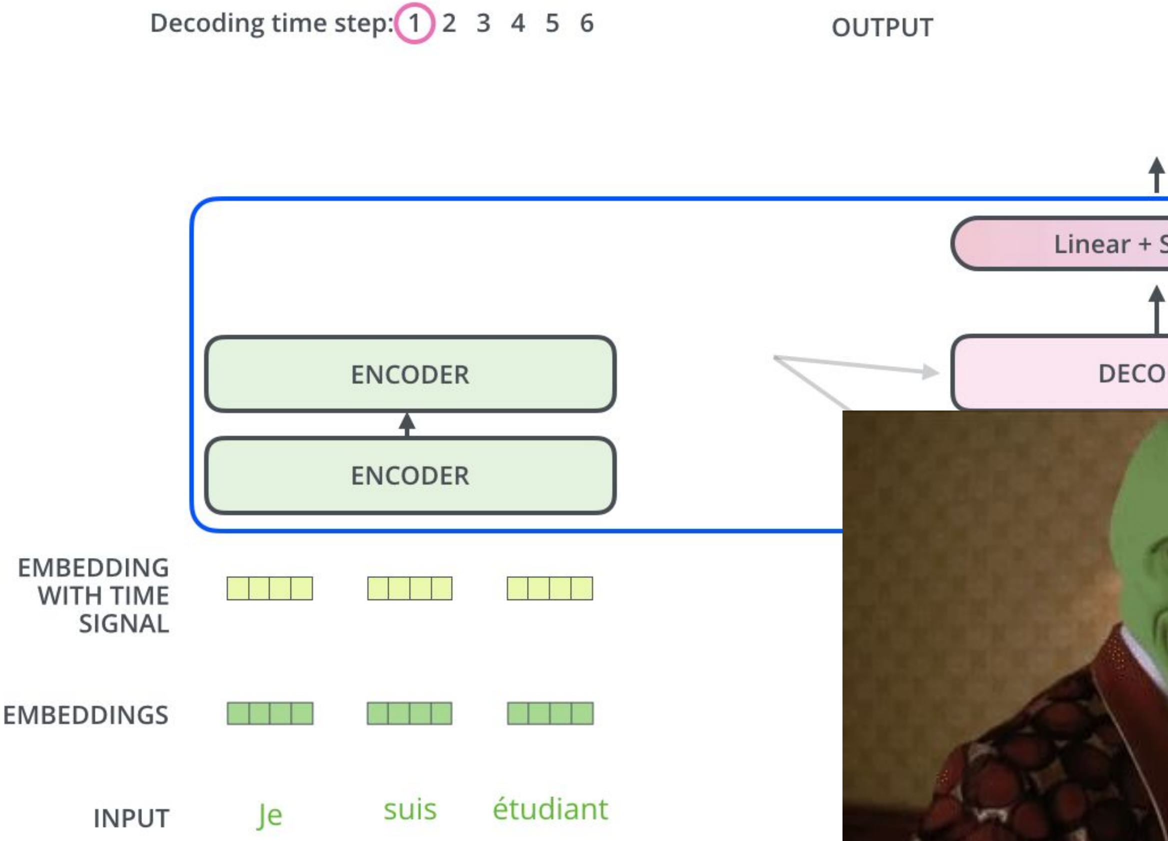
The Decoder Side



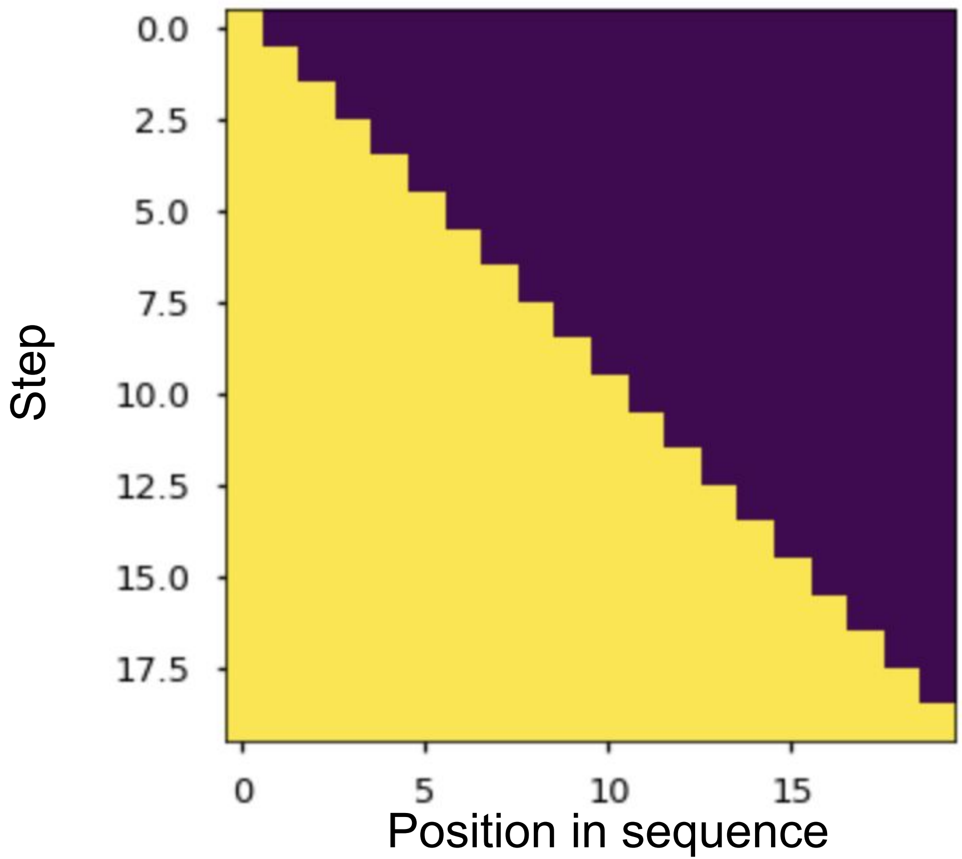
Here comes the mask

Image source: https://jalammar.github.io/illustrated-transformer/

The Decoder Side



The masked decoder input



The Decoder Side

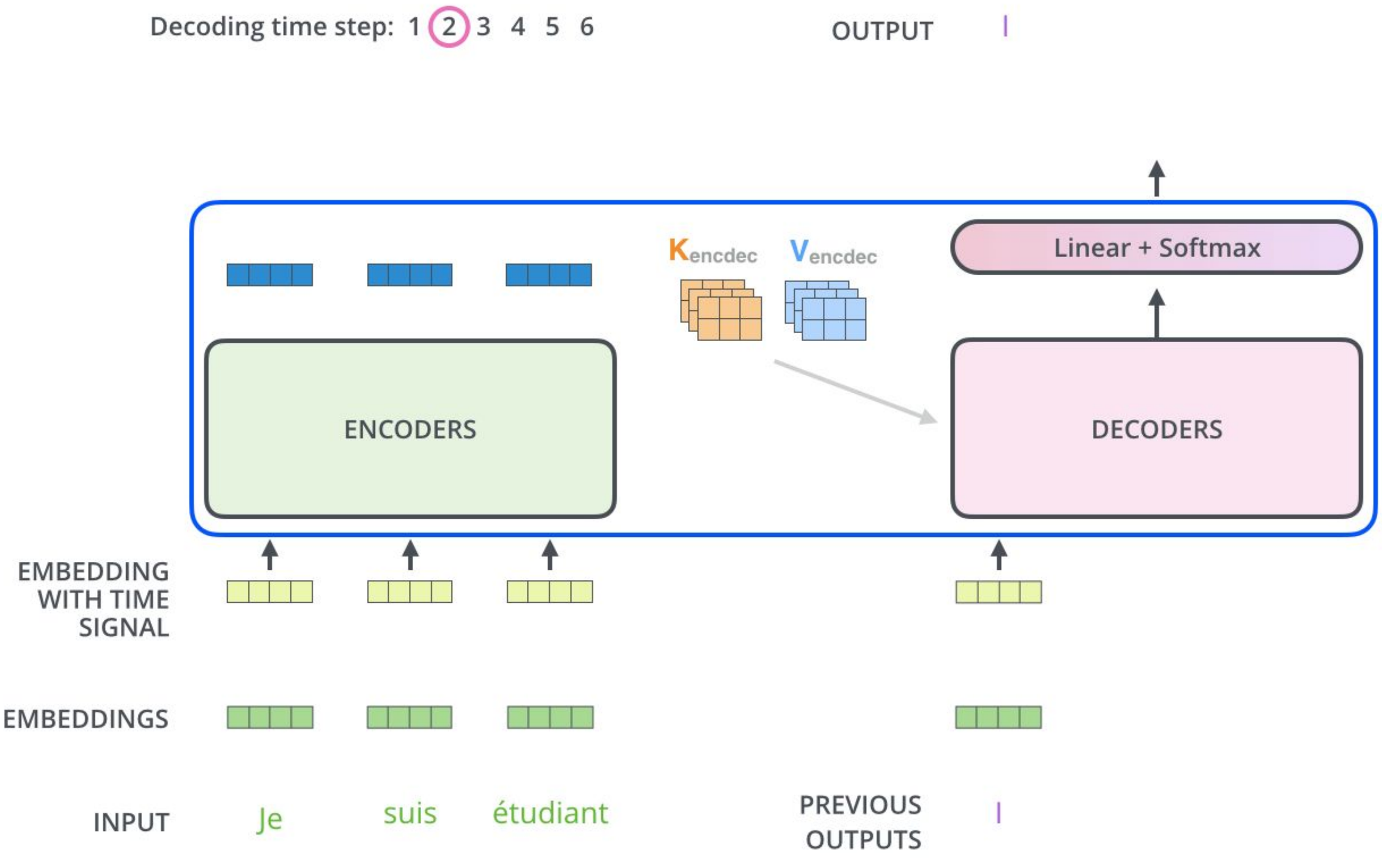


Image source: https://jalammar.github.io/illustrated-transformer/

Final Linear and Softmax Layer

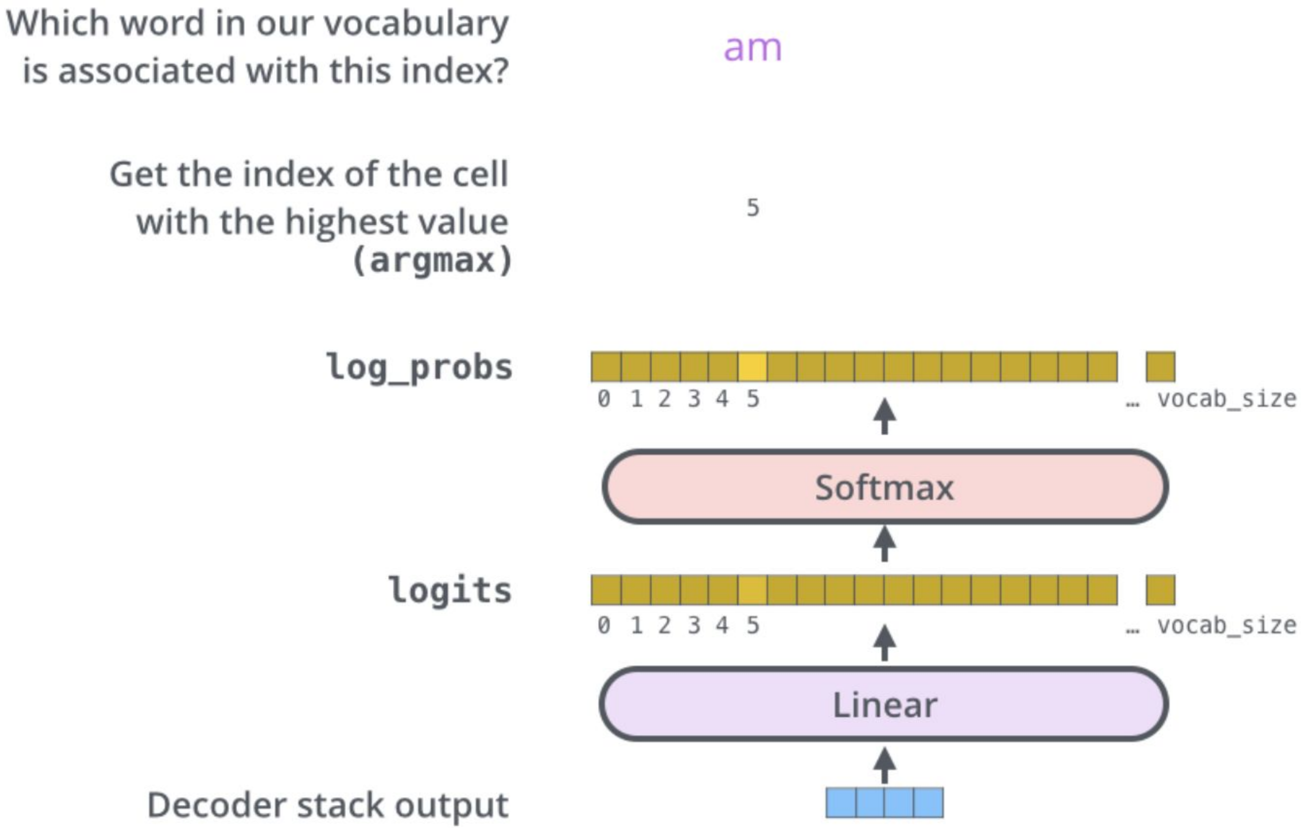


Image source: https://jalammar.github.io/illustrated-transformer/

The Transformer

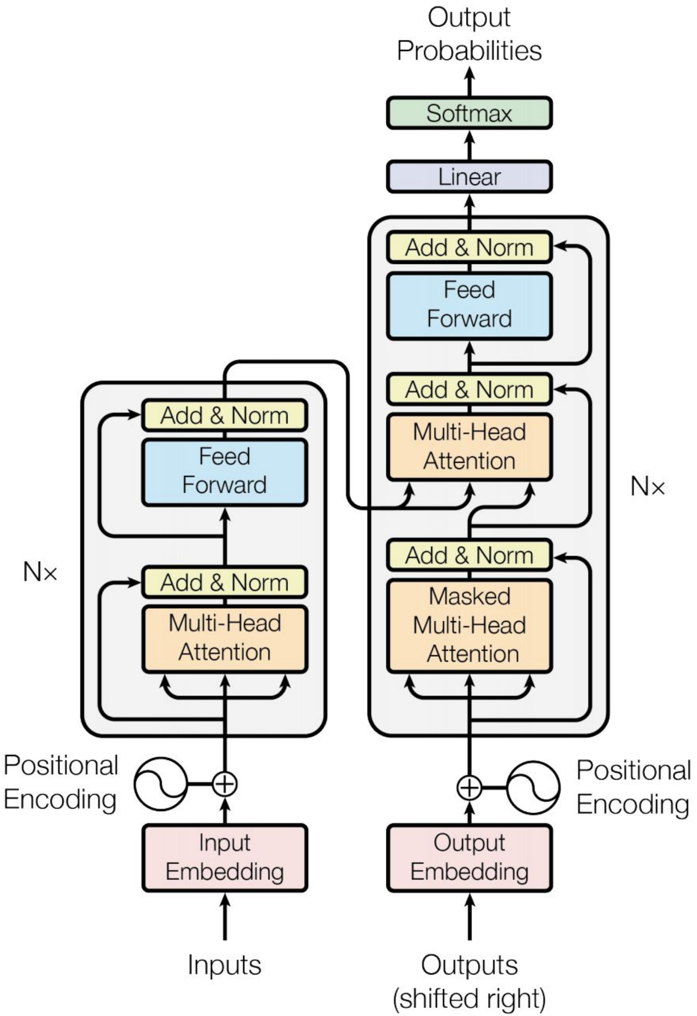


Image source: [Attention Is All You Need](), Neural Information Processing Systems 2017

Outro and Q&A

- 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