Машинное обучение

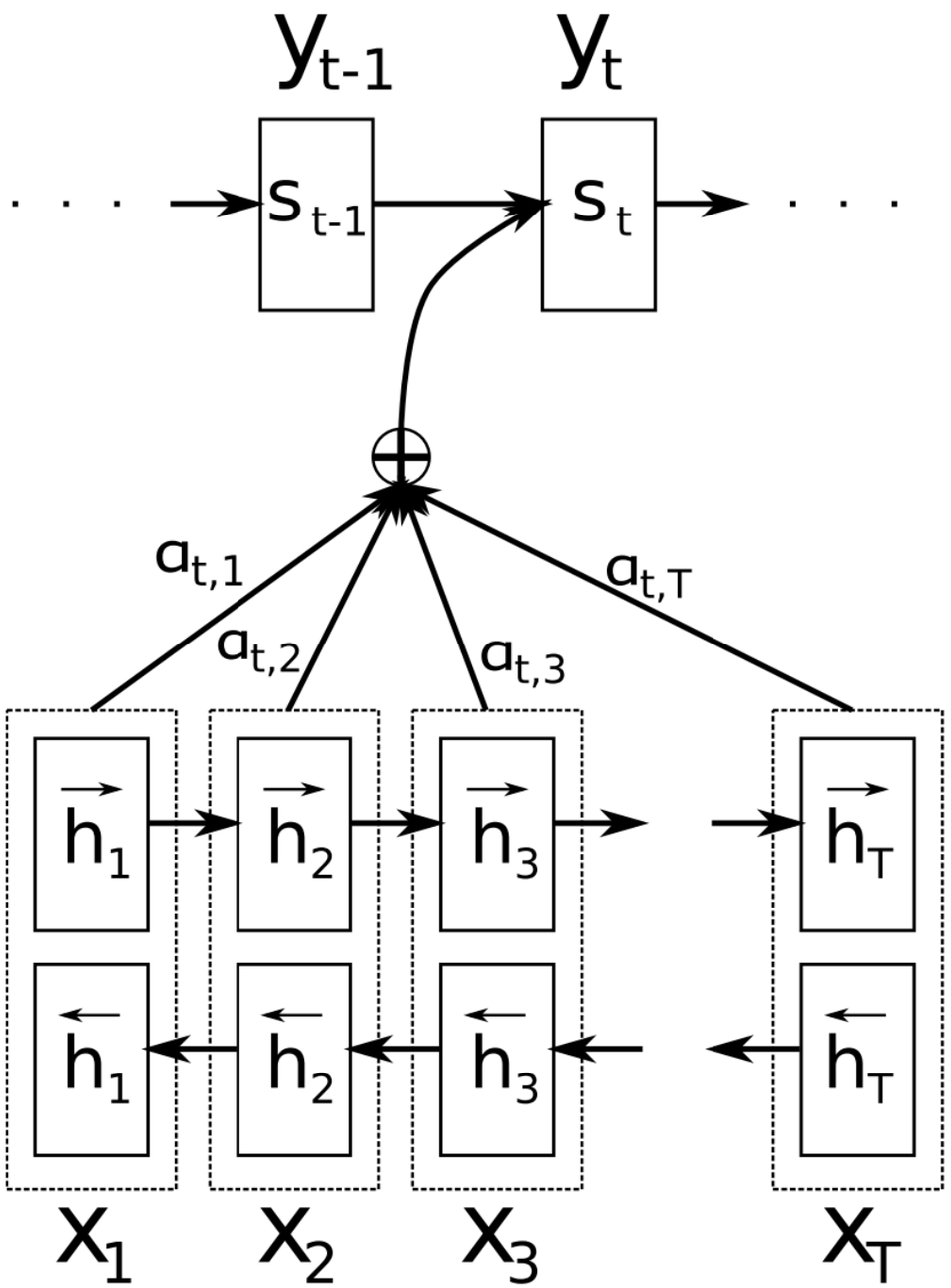
Лекция 10

Seq2seq learning, Attention

Власов Кирилл Вячеславович

2019

Recap: Attention



Скрытый слой RNN

$$s_i = f(s_{i-1}, y_{i-1}, c_i).$$

Взвешенная сумма

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j.$$

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})}$$

$$e_{ij} = a(s_{i-1}, h_j)$$

Neural Machine Translation by Jointly Learning to Align and Translate

А что если без RNN?

А что если без RNN?

Но ведь нам нужно обрабатывать последовательности…

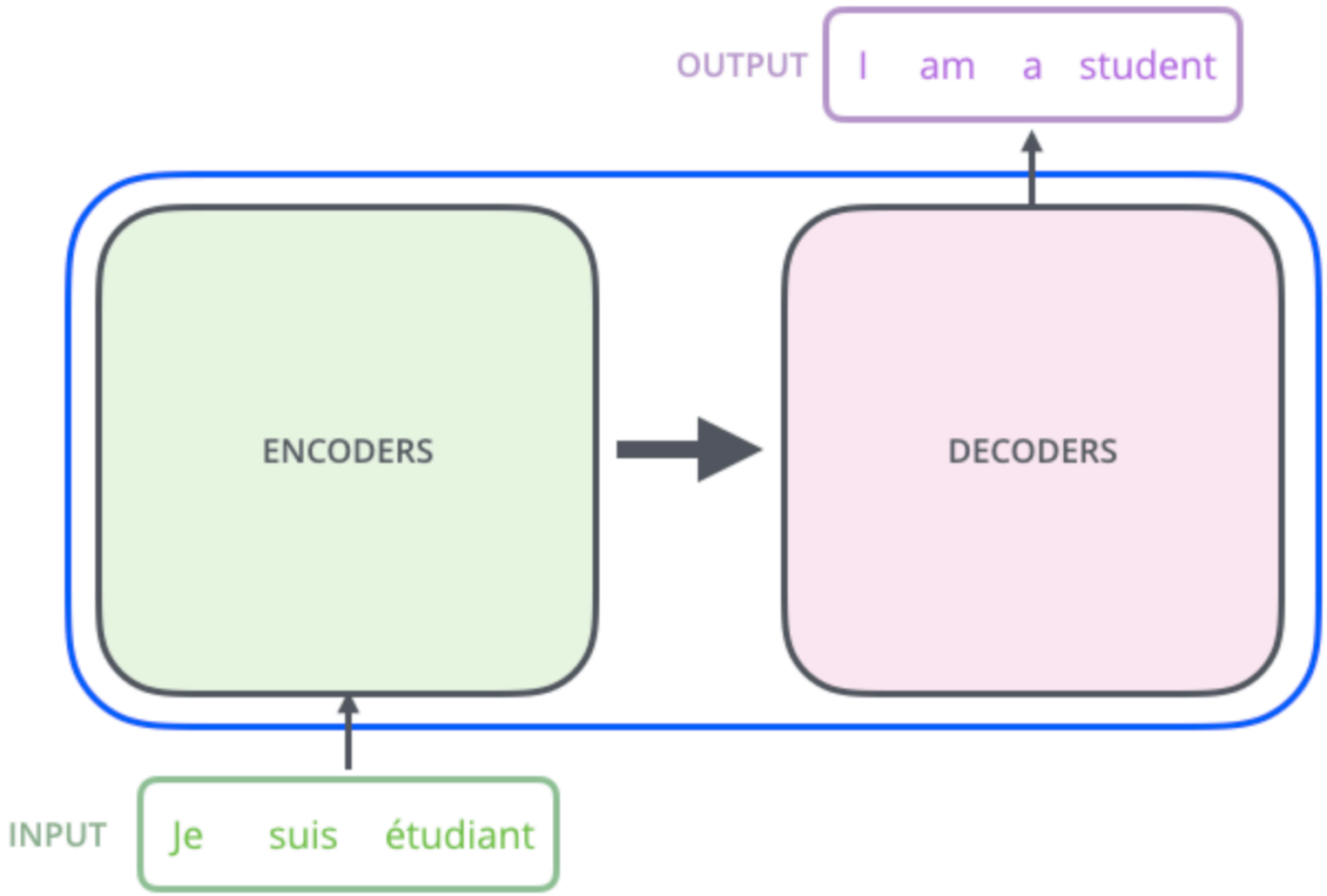
Self-Attention

The Illustrated Transformer

Self-Attention

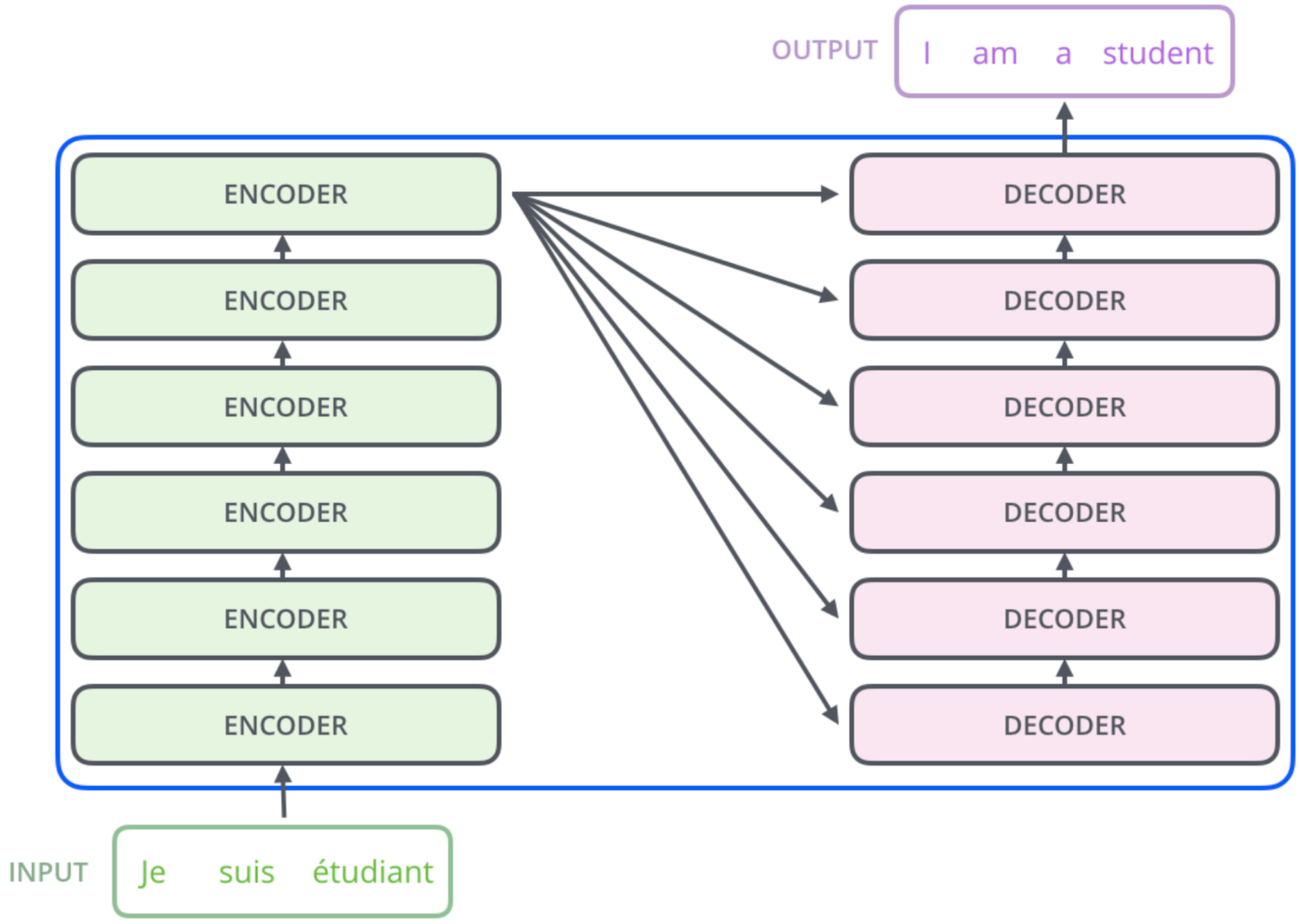
The Illustrated Transformer

Self-Attention



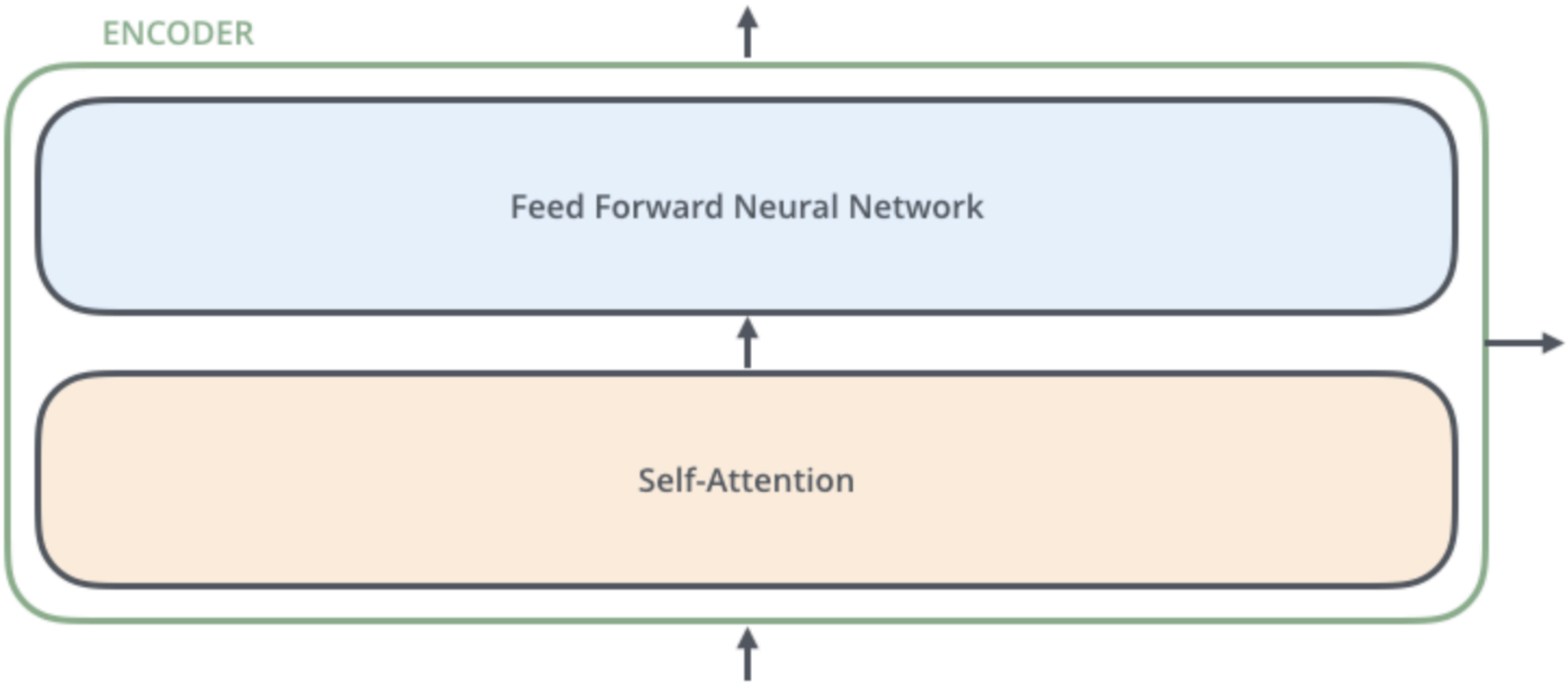
The Illustrated Transformer

Self-Attention



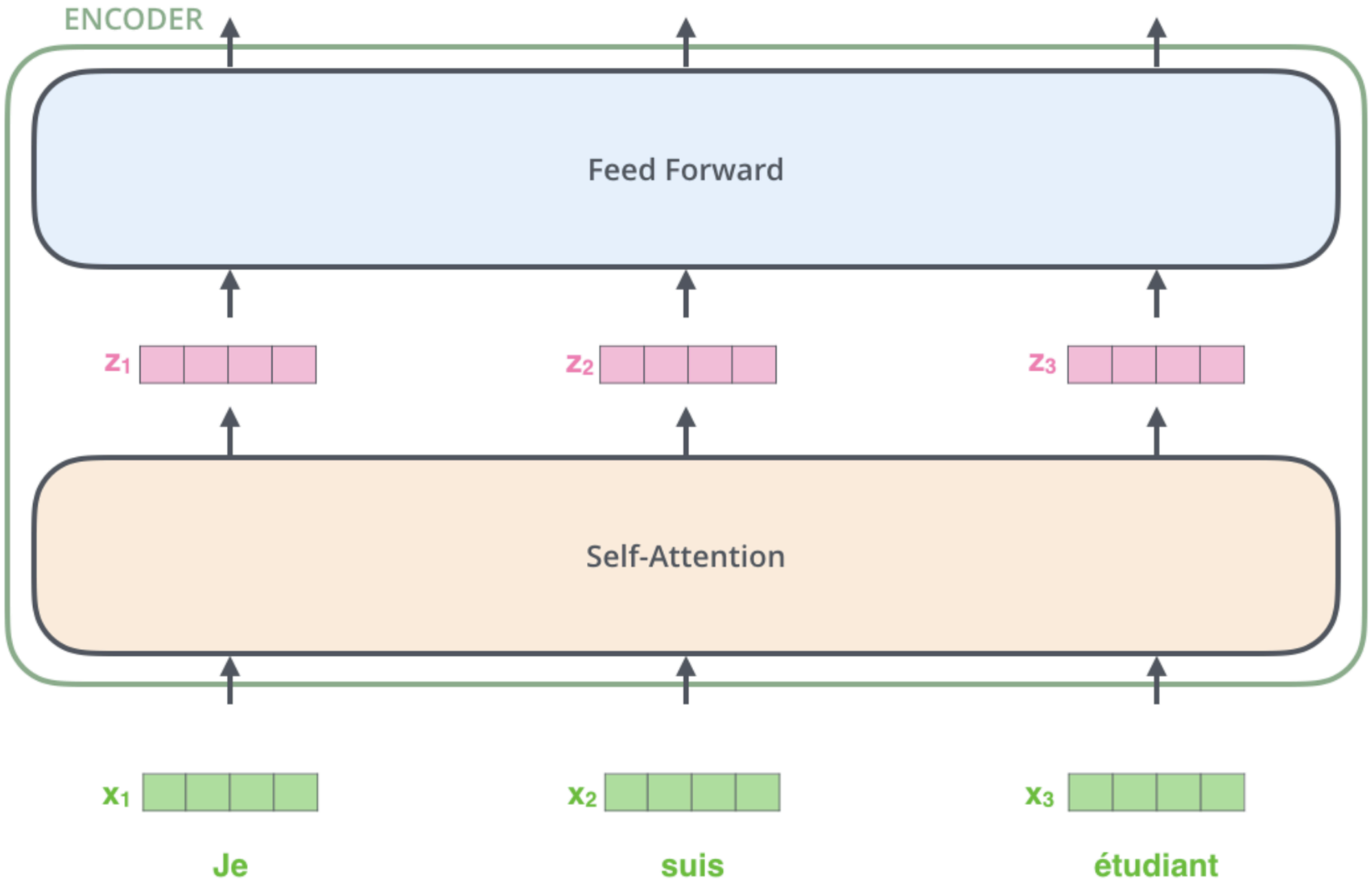
The Illustrated Transformer

Self-Attention



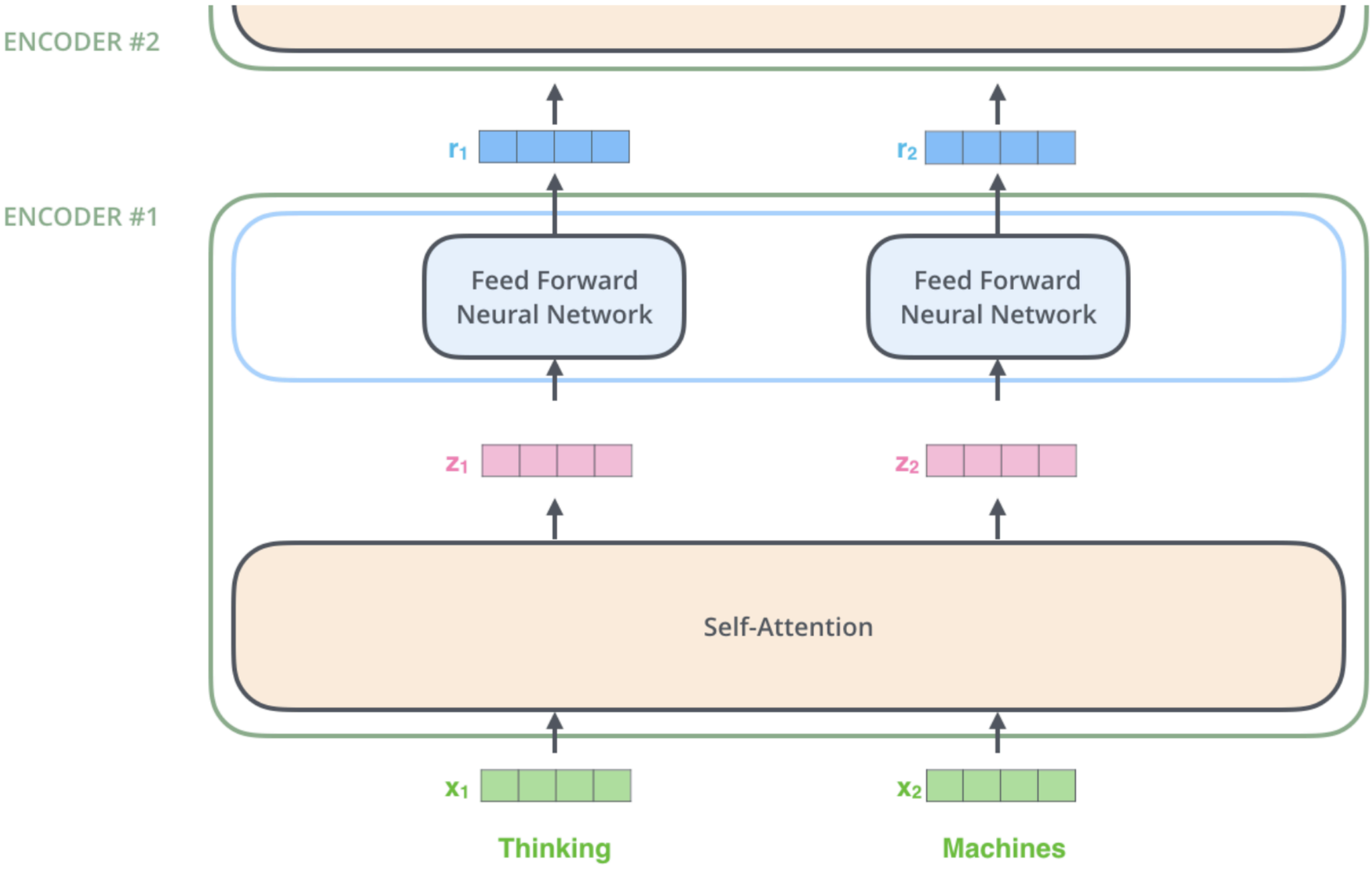
The Illustrated Transformer

Self-Attention



The Illustrated Transformer

Self-Attention



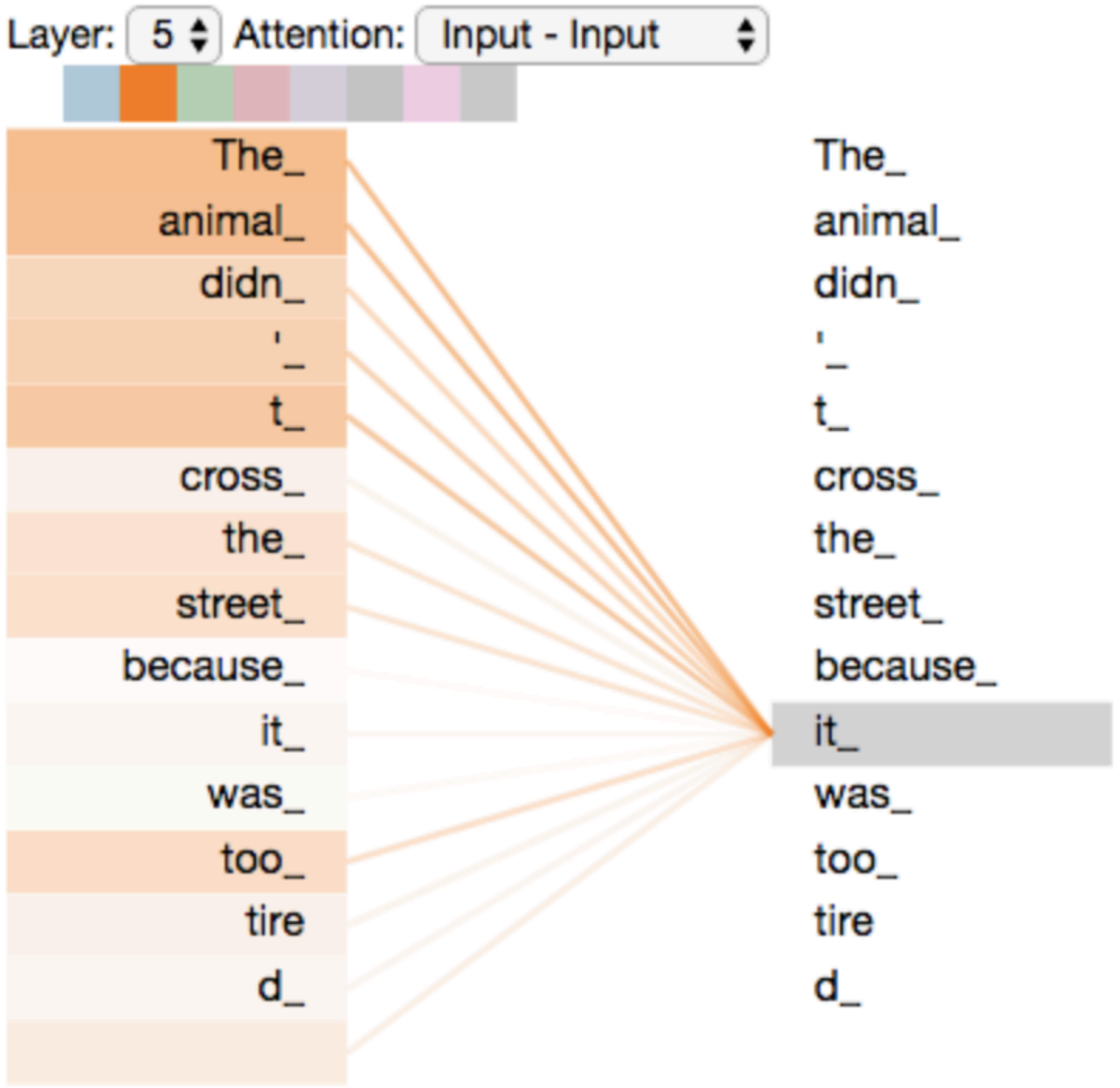
The Illustrated Transformer

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

?

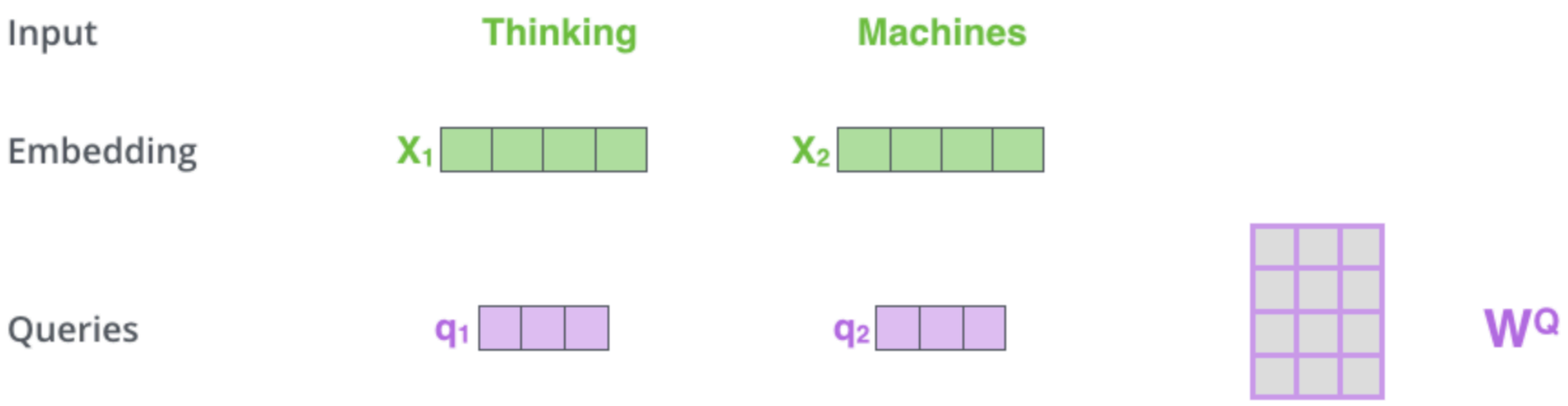
The animal didn't cross the street because it was too tired

Self-Attention



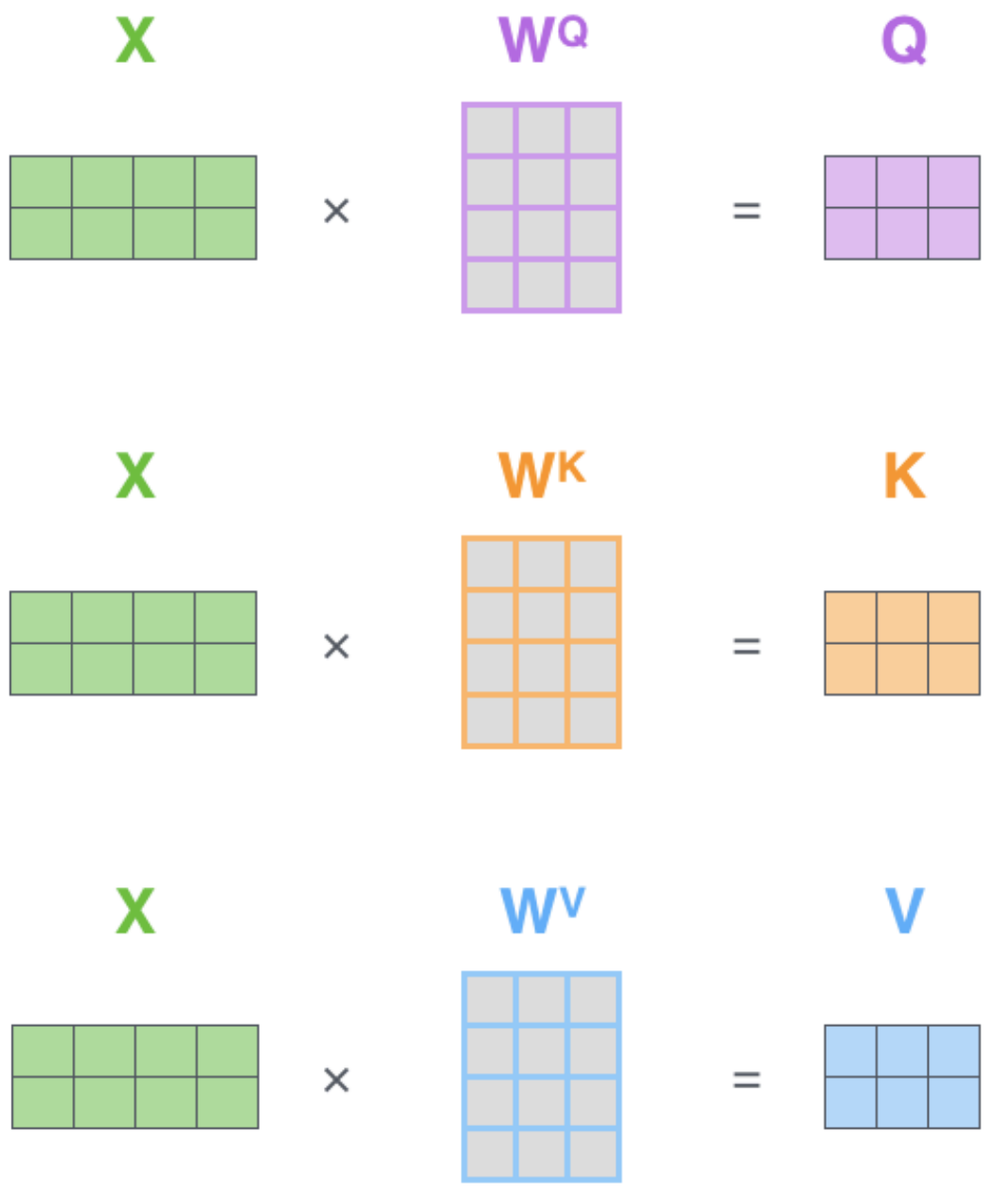
The Illustrated Transformer

Шаг 1 - Векторы Queries, Keys, Values



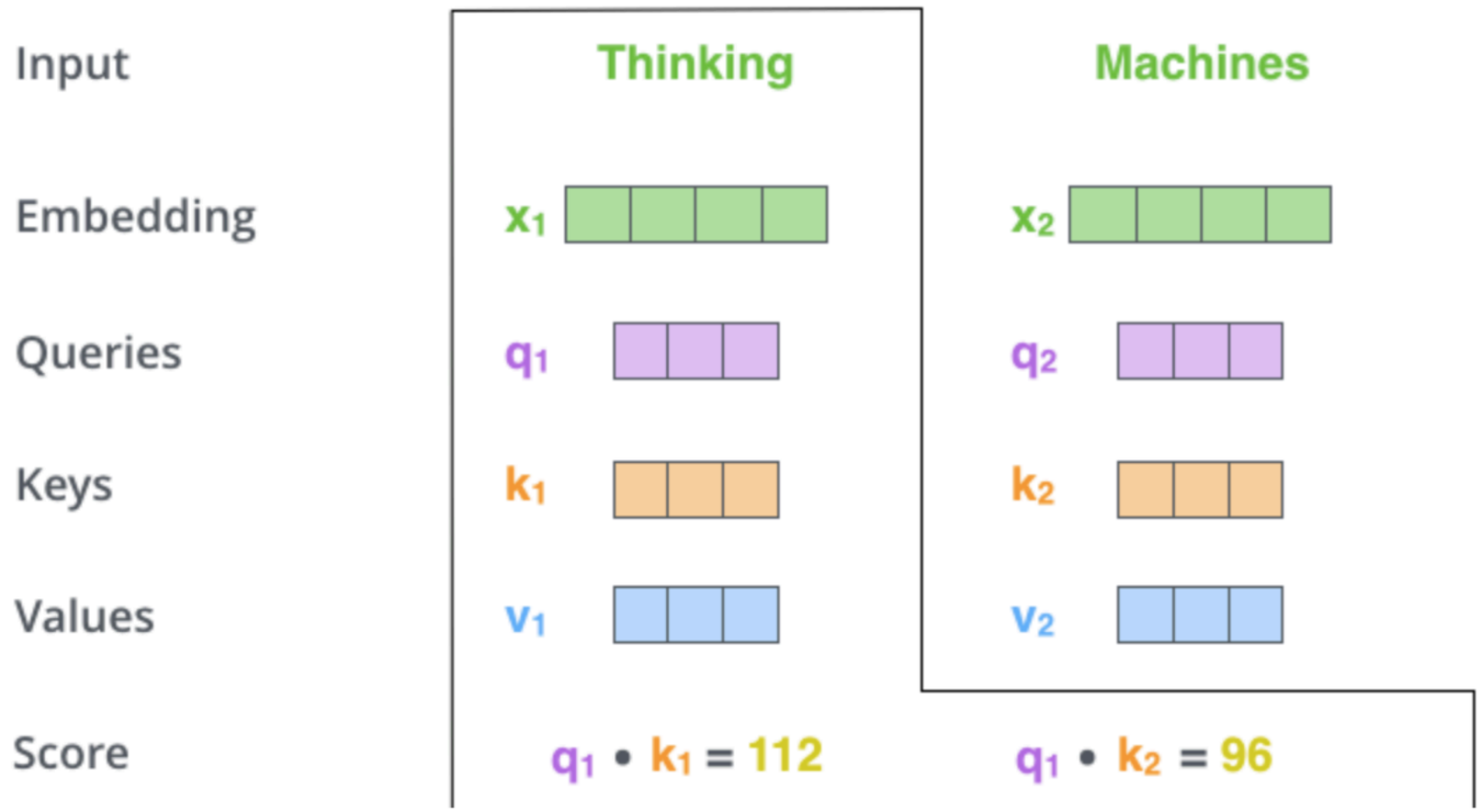
The Illustrated Transformer

Шаг 1 - Векторы Queries, Keys, Values



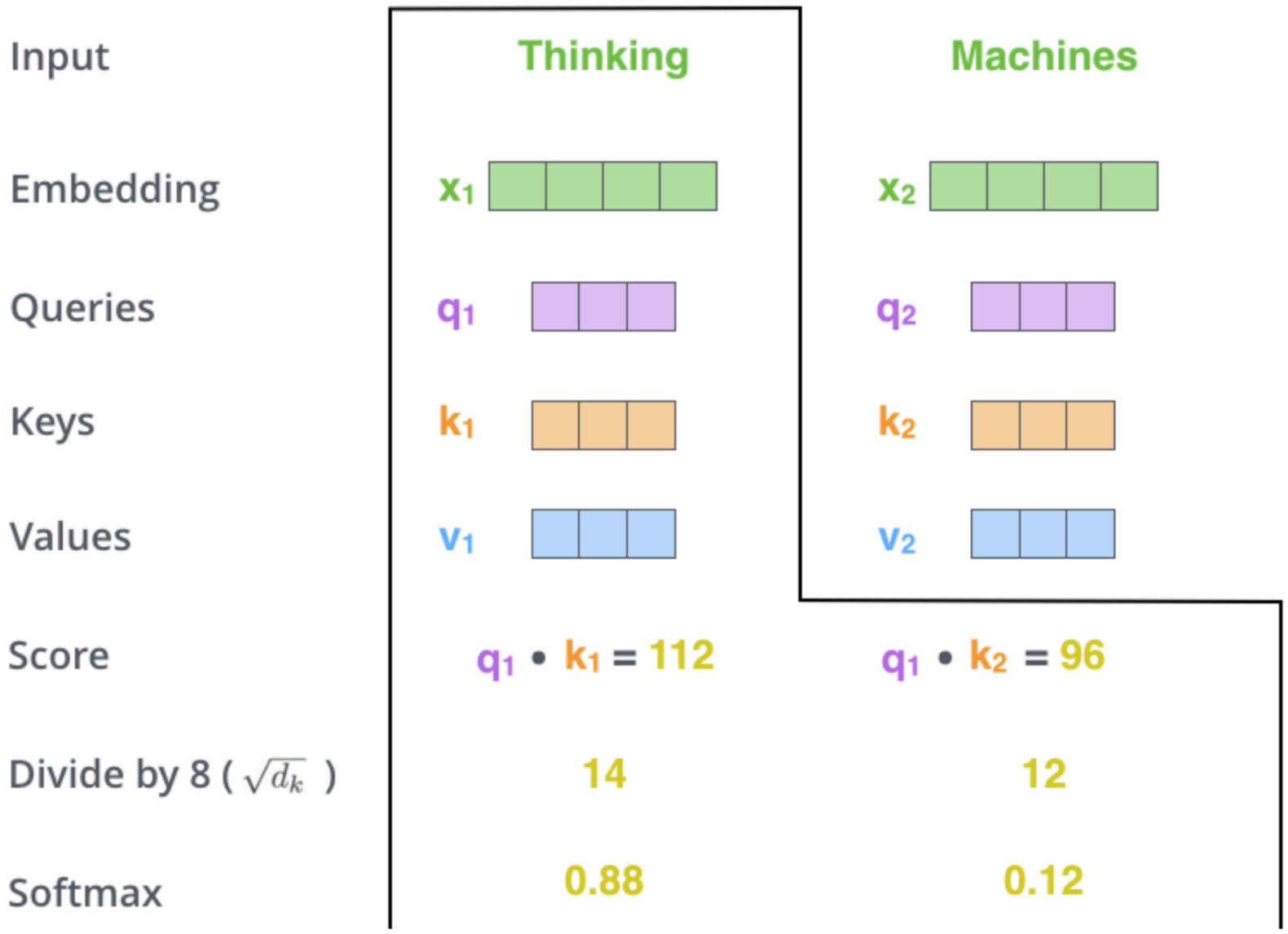
The Illustrated Transformer

Шаг 2 - Вычисление Score



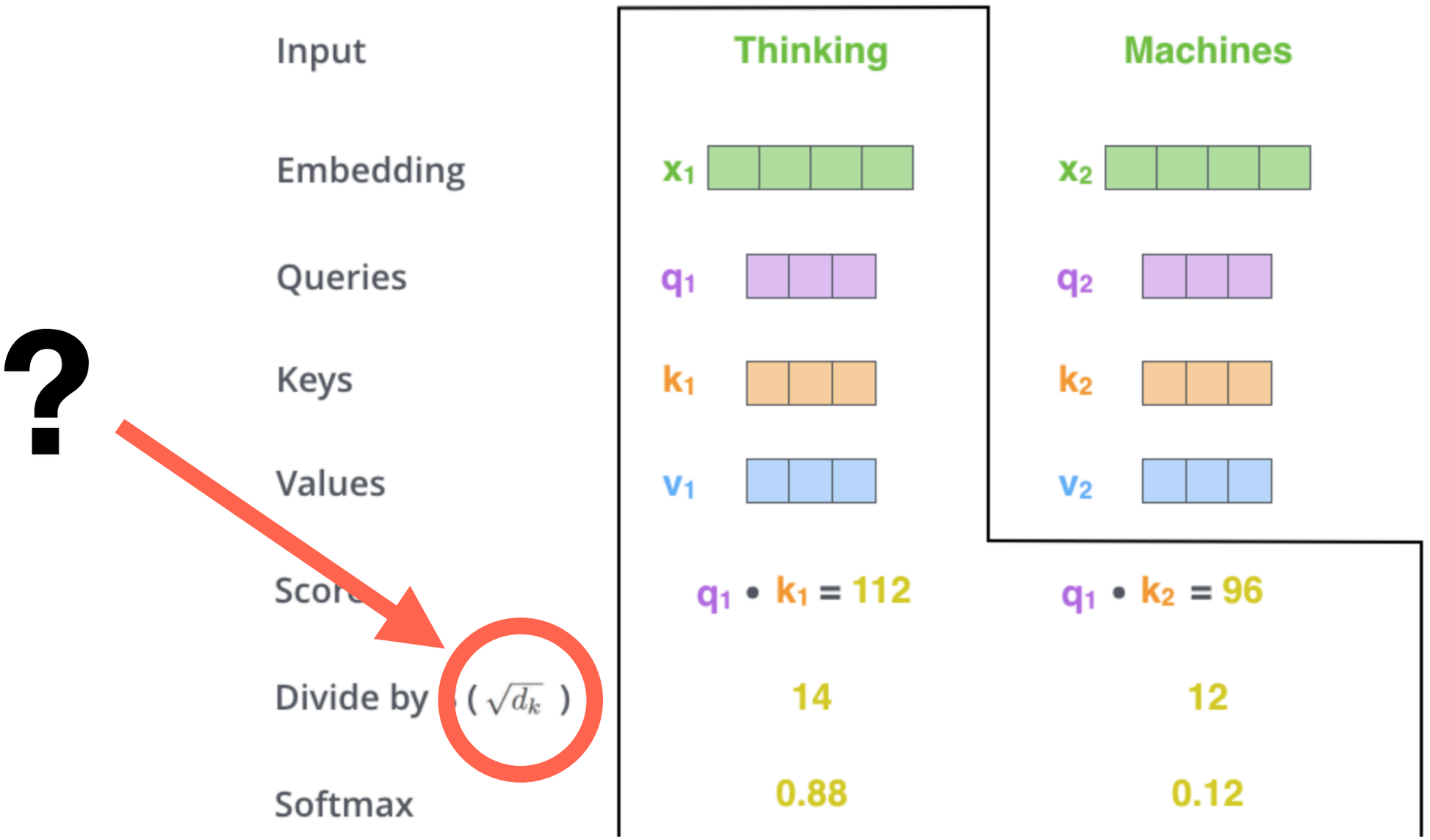
The Illustrated Transformer

Шаг 3: Делим Скор на sqrt(d)
Шаг 4: Softmax



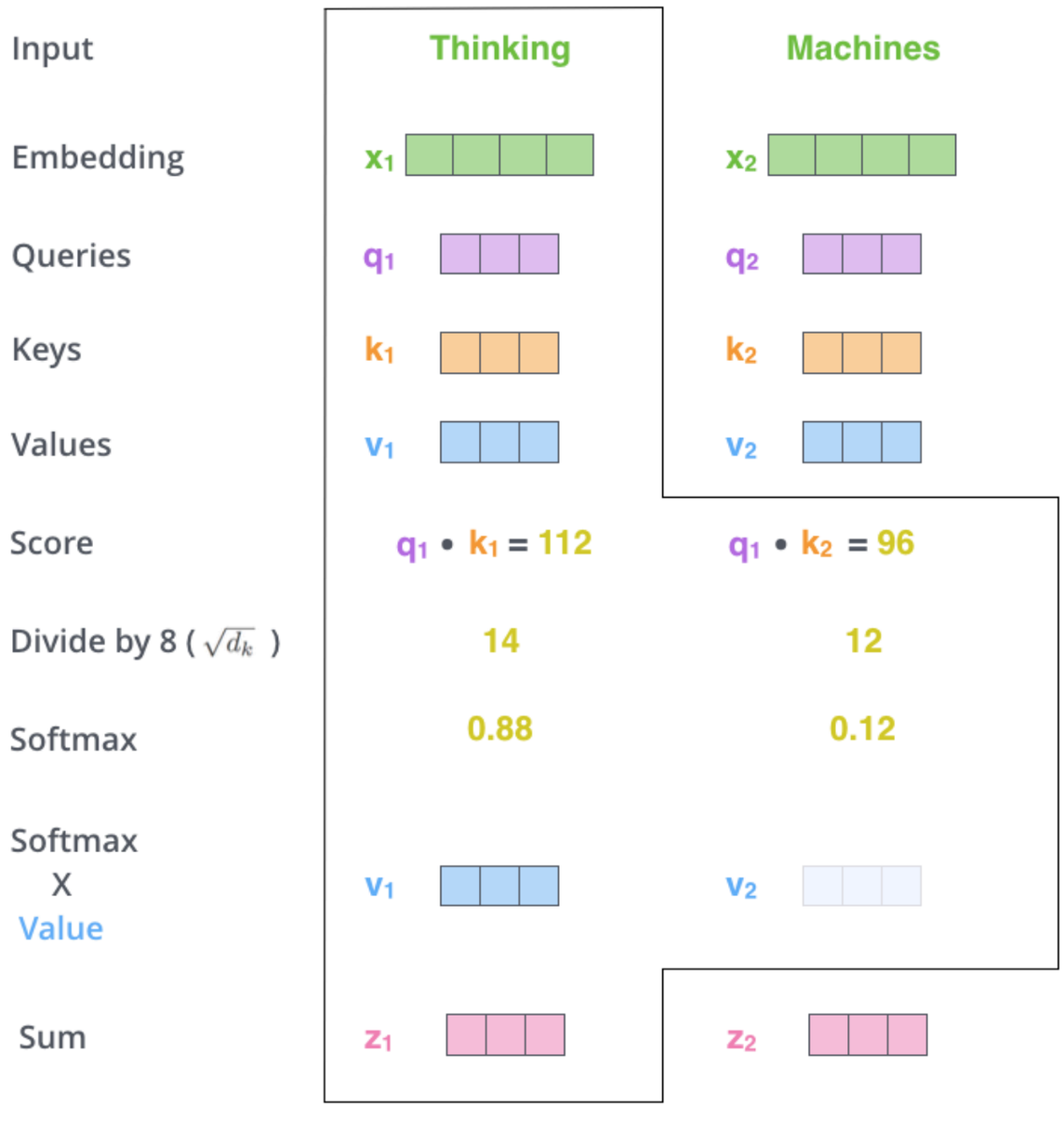
The Illustrated Transformer

Шаг 3: Делим Скор на sqrt(d)
Шаг 4: Softmax



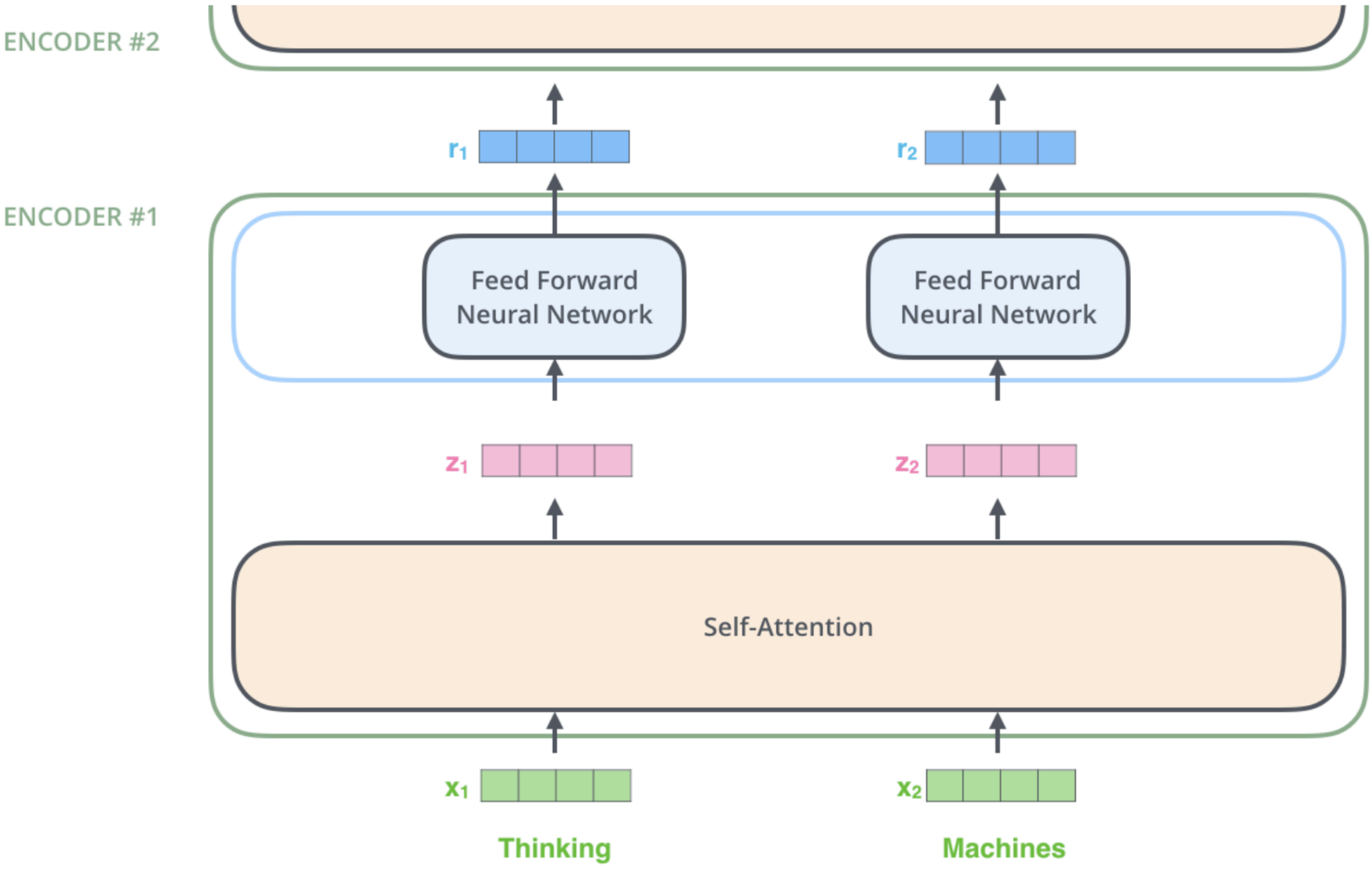
The Illustrated Transformer

Шаг 5: Умножение каждого Values вектора на значение softmax
Шаг 6: Суммирование векторов



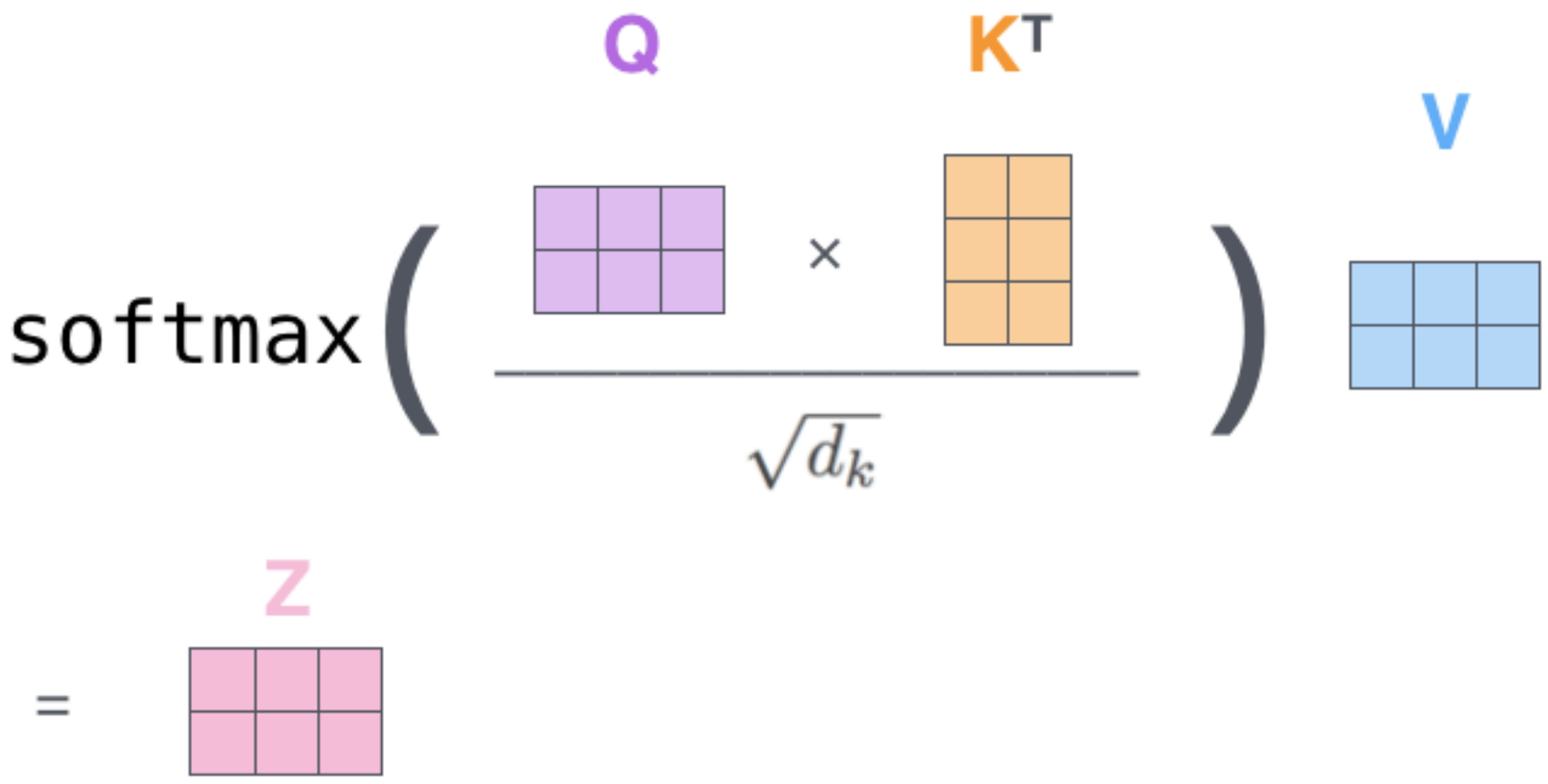
The Illustrated Transformer

Self-Attention

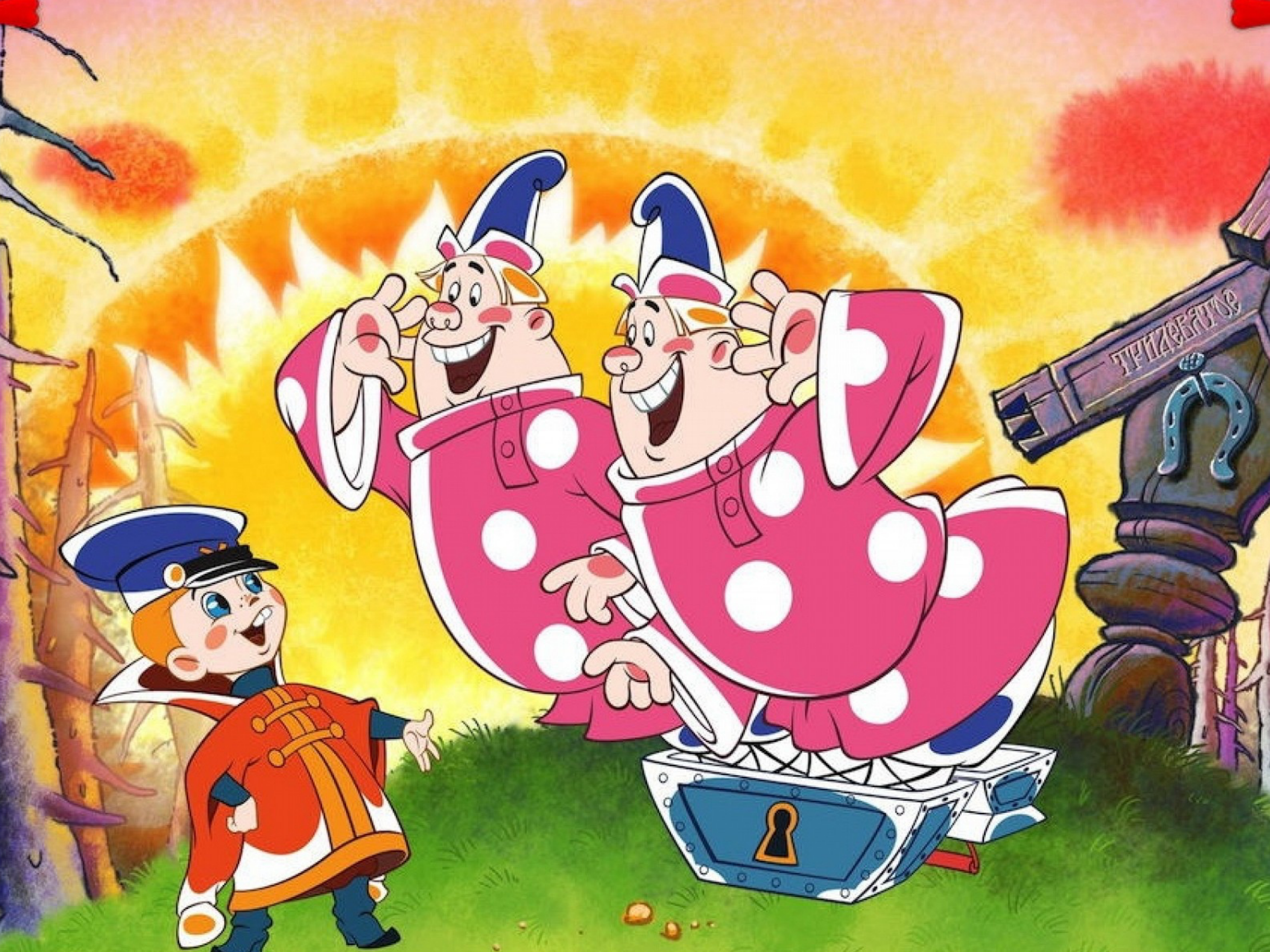


The Illustrated Transformer

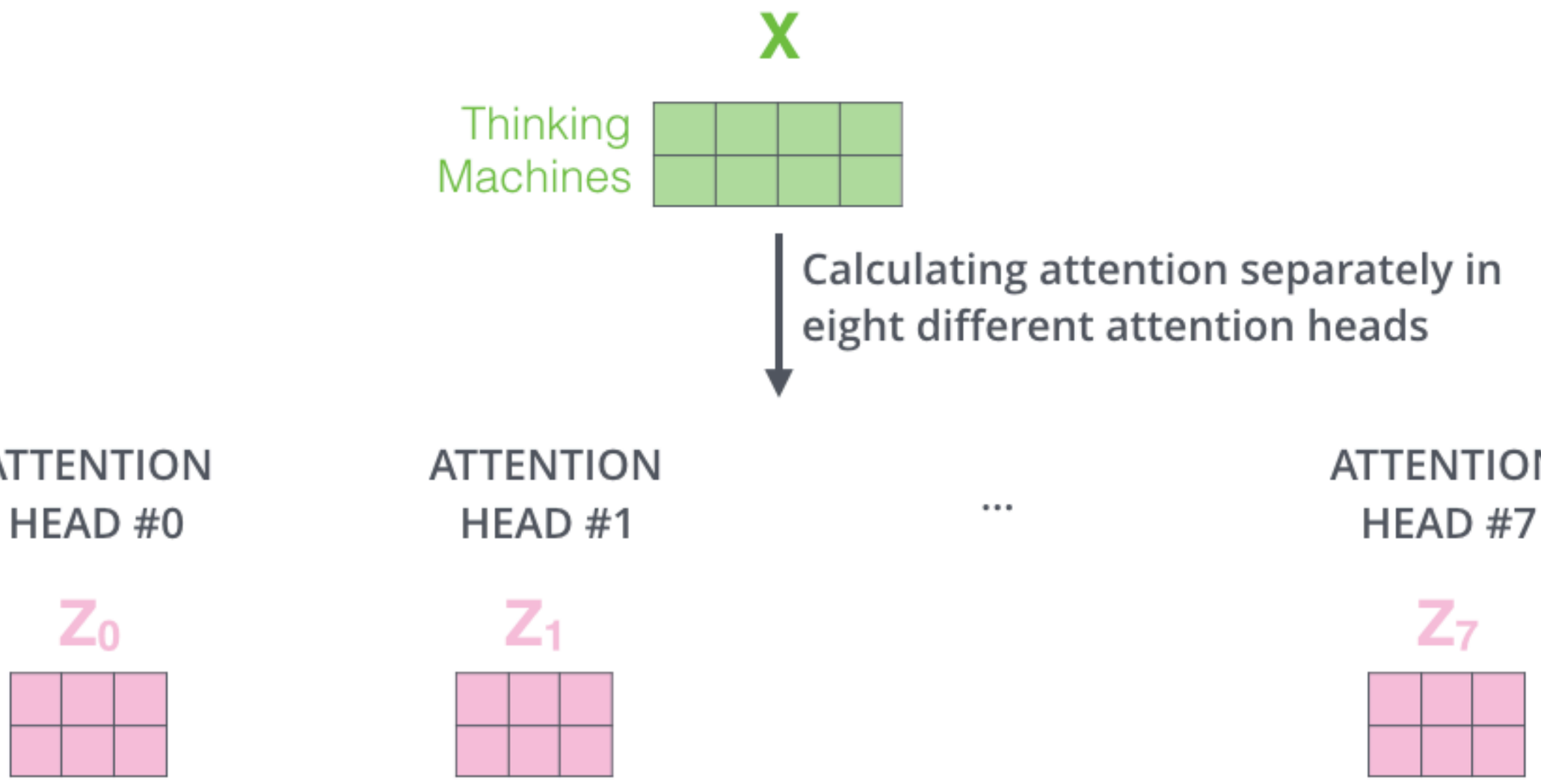
Self-Attention



The Illustrated Transformer


ТРИЗВЕЗДОВ

Multihead Self-Attention



The Illustrated Transformer

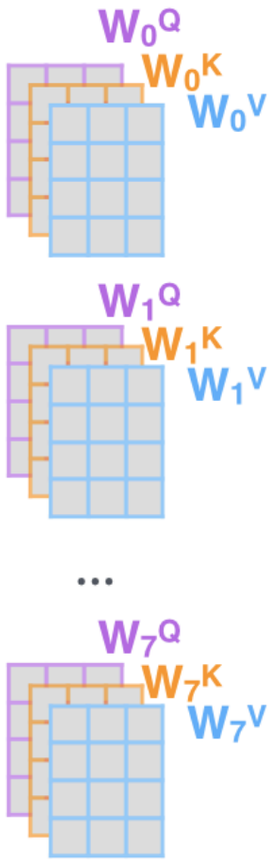
Multihead Self-Attention

1) This is our input sentence*

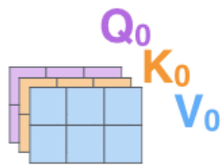
Thinking
Machines

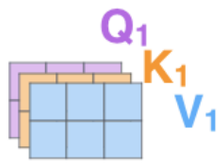
2) We embed each word*

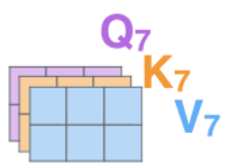
3) Split into 8 heads. We multiply X or R with weight matrices



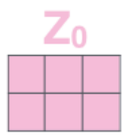
4) Calculate attention using the resulting Q/K/V matrices

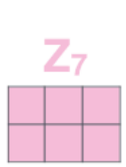


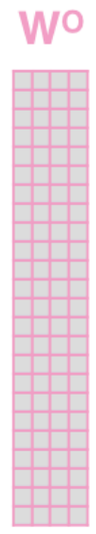


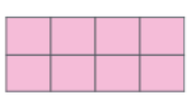


5) Concatenate the resulting Z matrices, then multiply with weight matrix W^O to produce the output of the layer









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

...

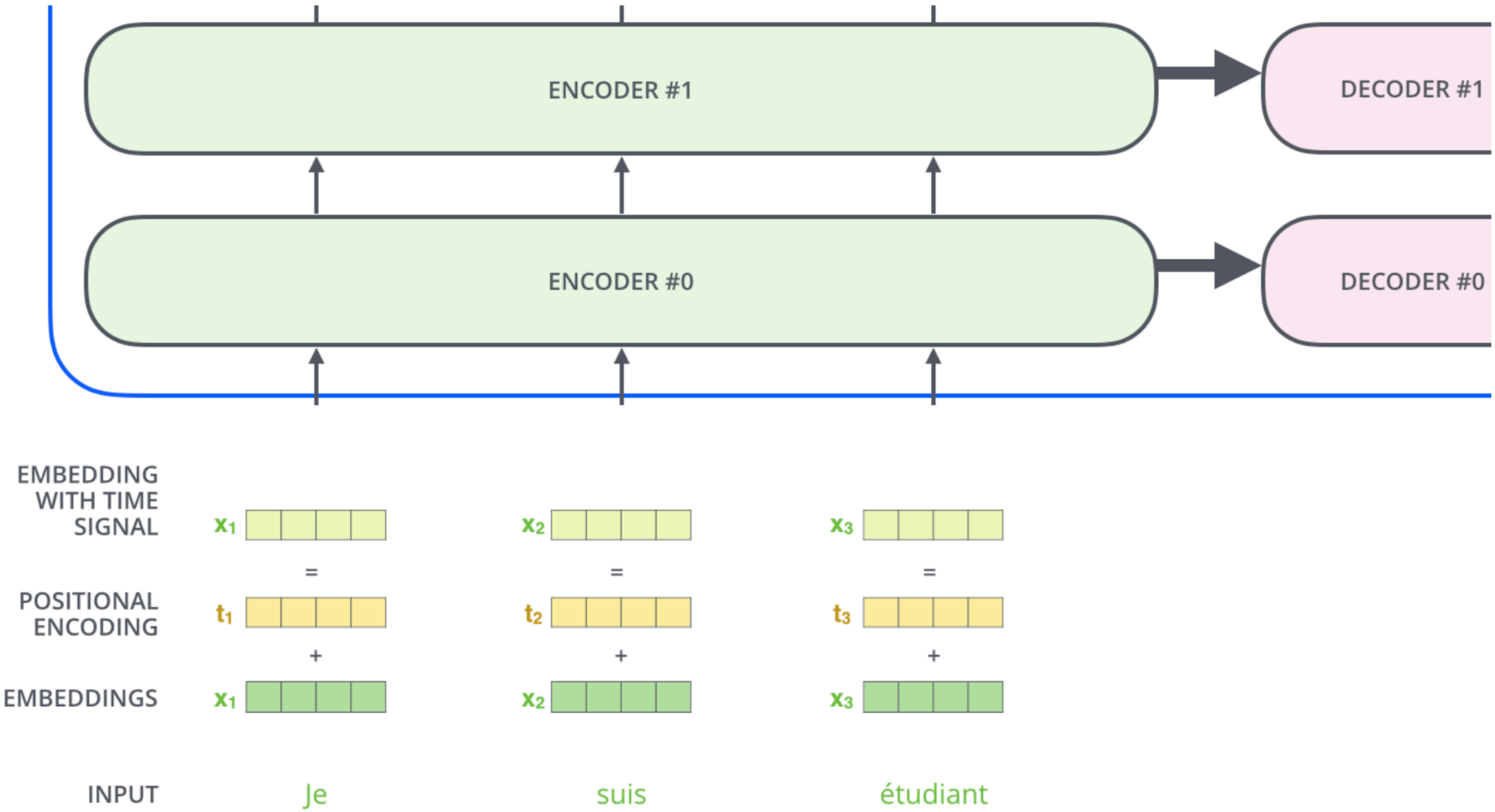
...

...

The Illustrated Transformer

Но ведь нам нужно обрабатывать последовательности…

Positional Encoding



The Illustrated Transformer

Positional Encoding

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

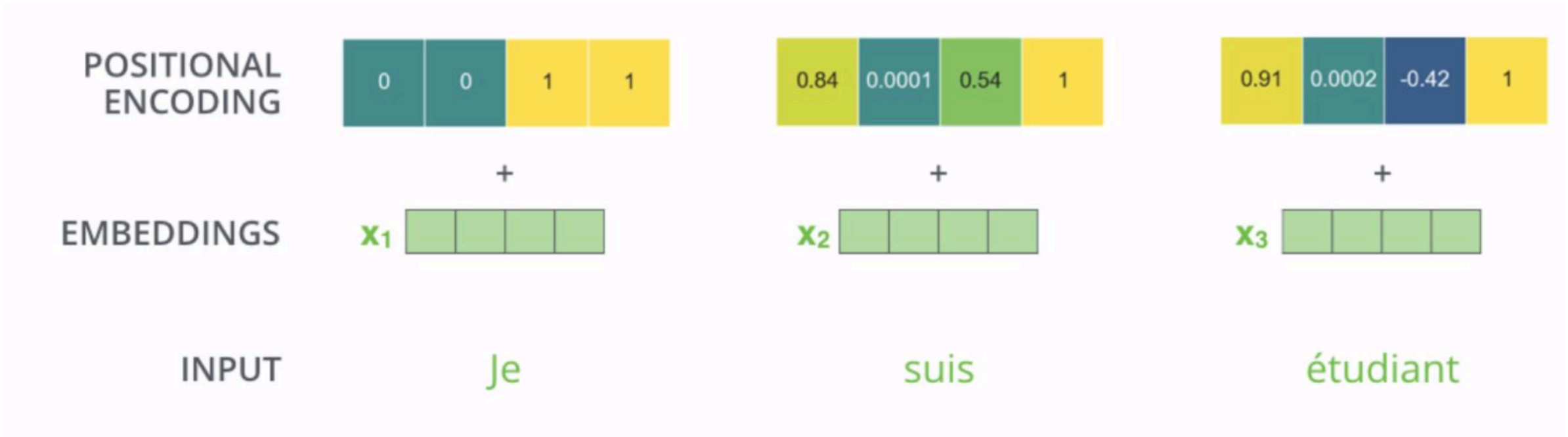
$$PE_{(pos,\ 2i+1)} = cos(pos\ /\ 10000^{2i\ /\ d_{\text{model}}})$$

- *pos* is the position
- *i* is the dimension.

Each dimension of the positional encoding corresponds to a sinusoid.
The wavelengths form a geometric progression from 2pi to 2pi * 10 000

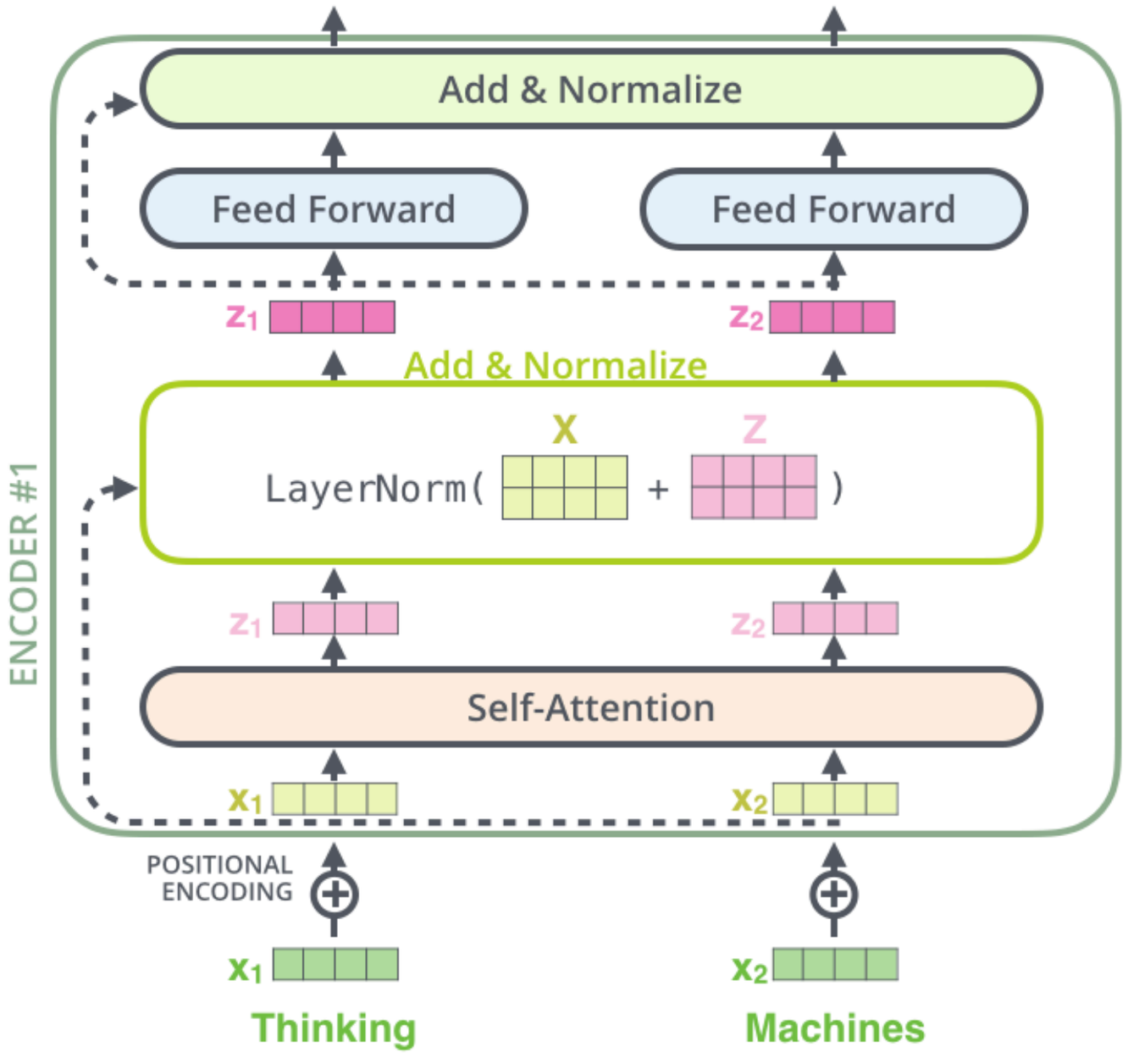
https://github.com/ml-mipt/ml-mipt/blob/advanced/week04_Transformer/week04_Transformer.pdf

Positional Encoding

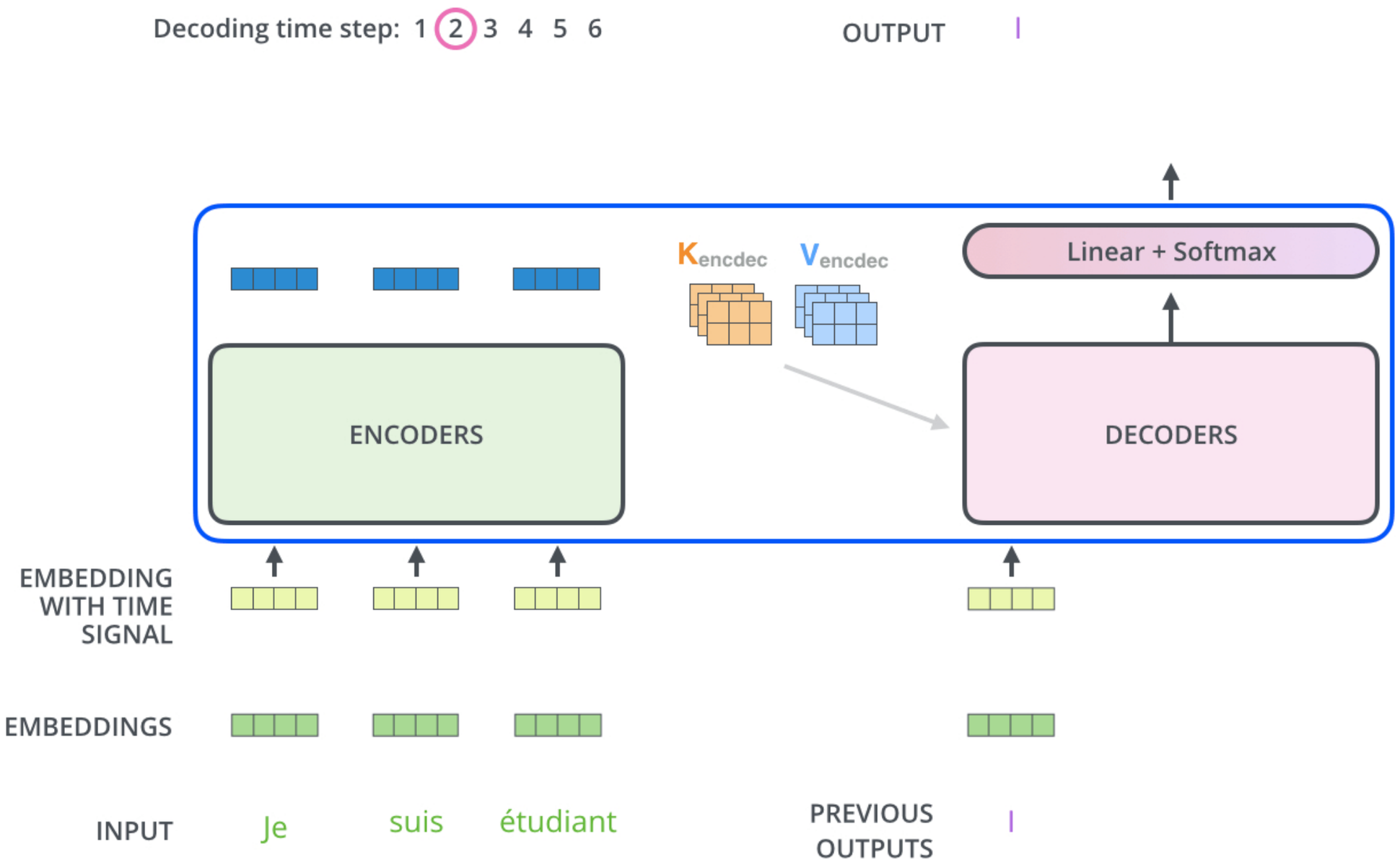


https://github.com/ml-mipt/ml-mipt/blob/advanced/week04_Transformer/week04_positional_encoding_carriers.ipynb

The Residuals & layer-normalization



Self-Attention



The Illustrated Transformer

Output Probabilities

Softmax

Linear

Stage 5

Add & Norm

Feed Forward

Stage 4

Add & Norm

Multi-Head Attention

Add & Norm

Masked Multi-Head Attention

Stage 3

Nx

Add & Norm

Feed Forward

Add & Norm

Multi-Head Attention

ge 2 Nx

sitional coding

Input Embedding

Inputs

Positional Encoding

Output Embedding

Outputs (shifted right)