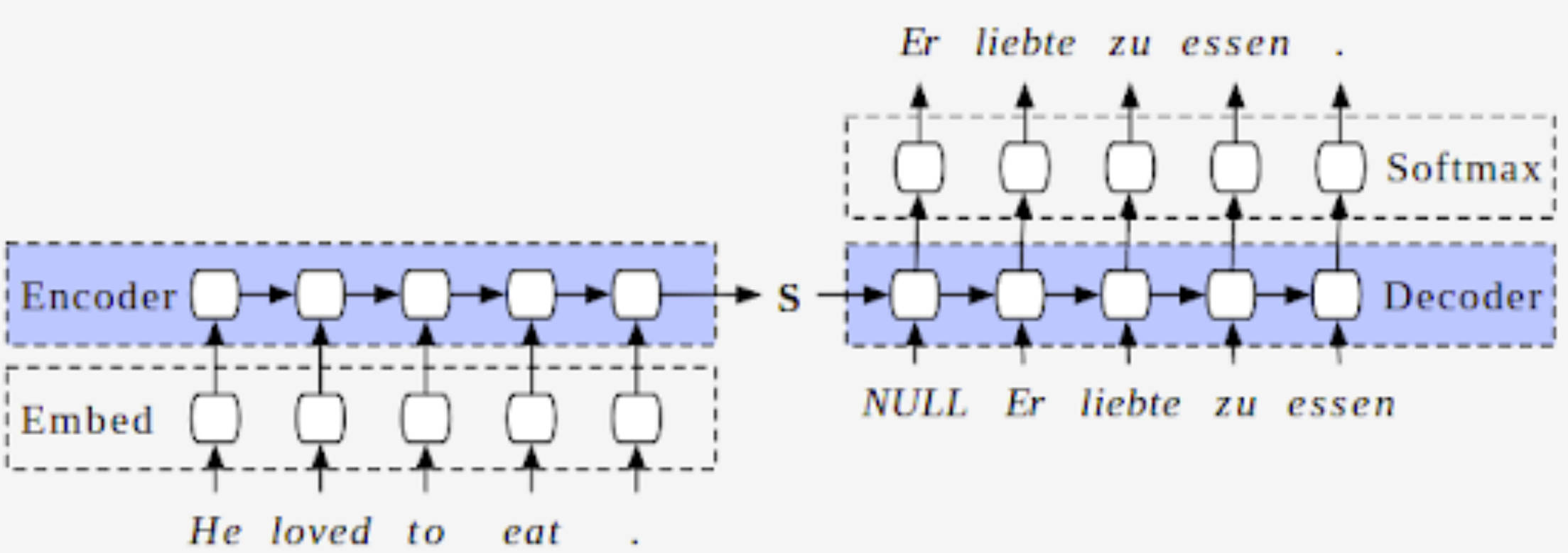
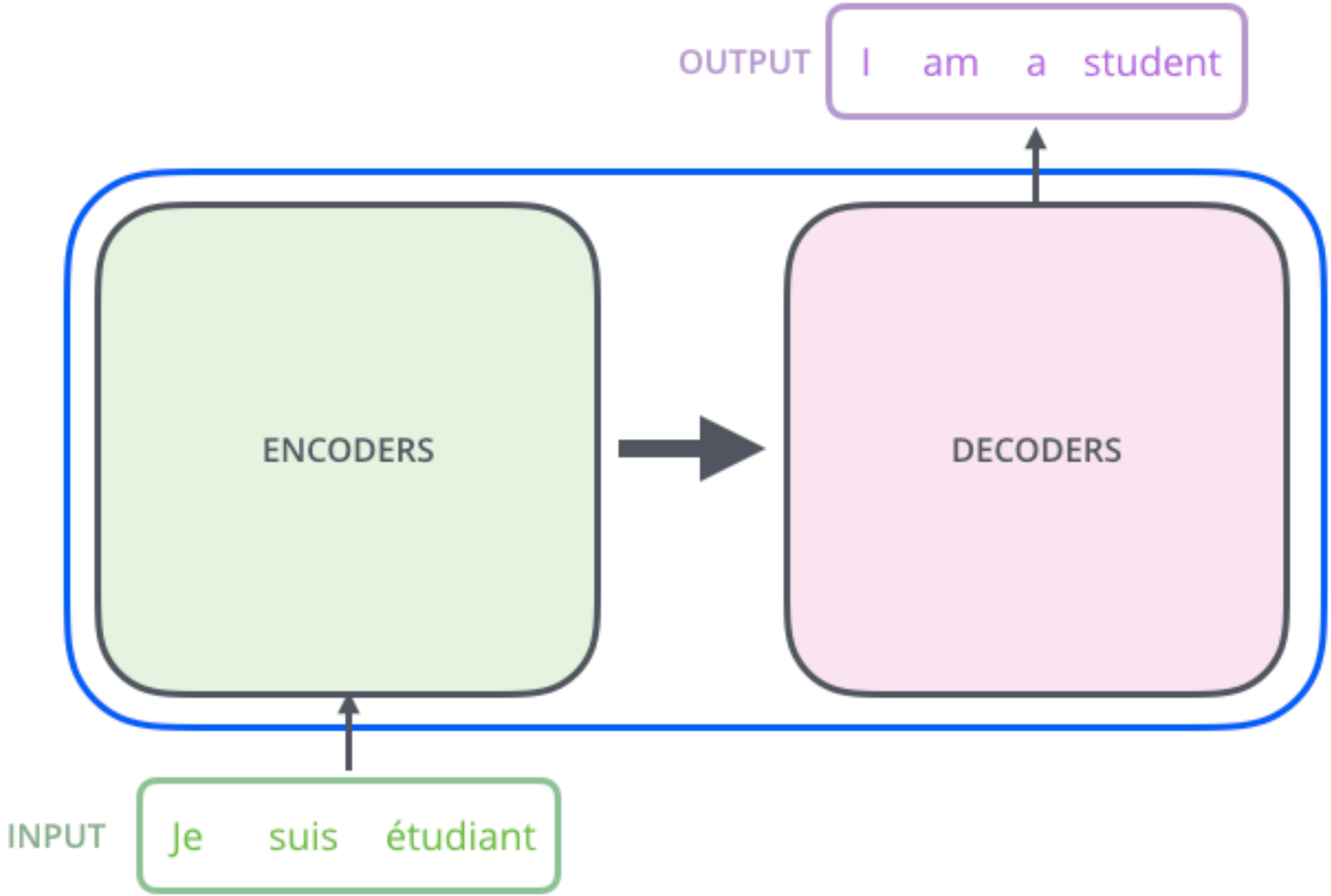
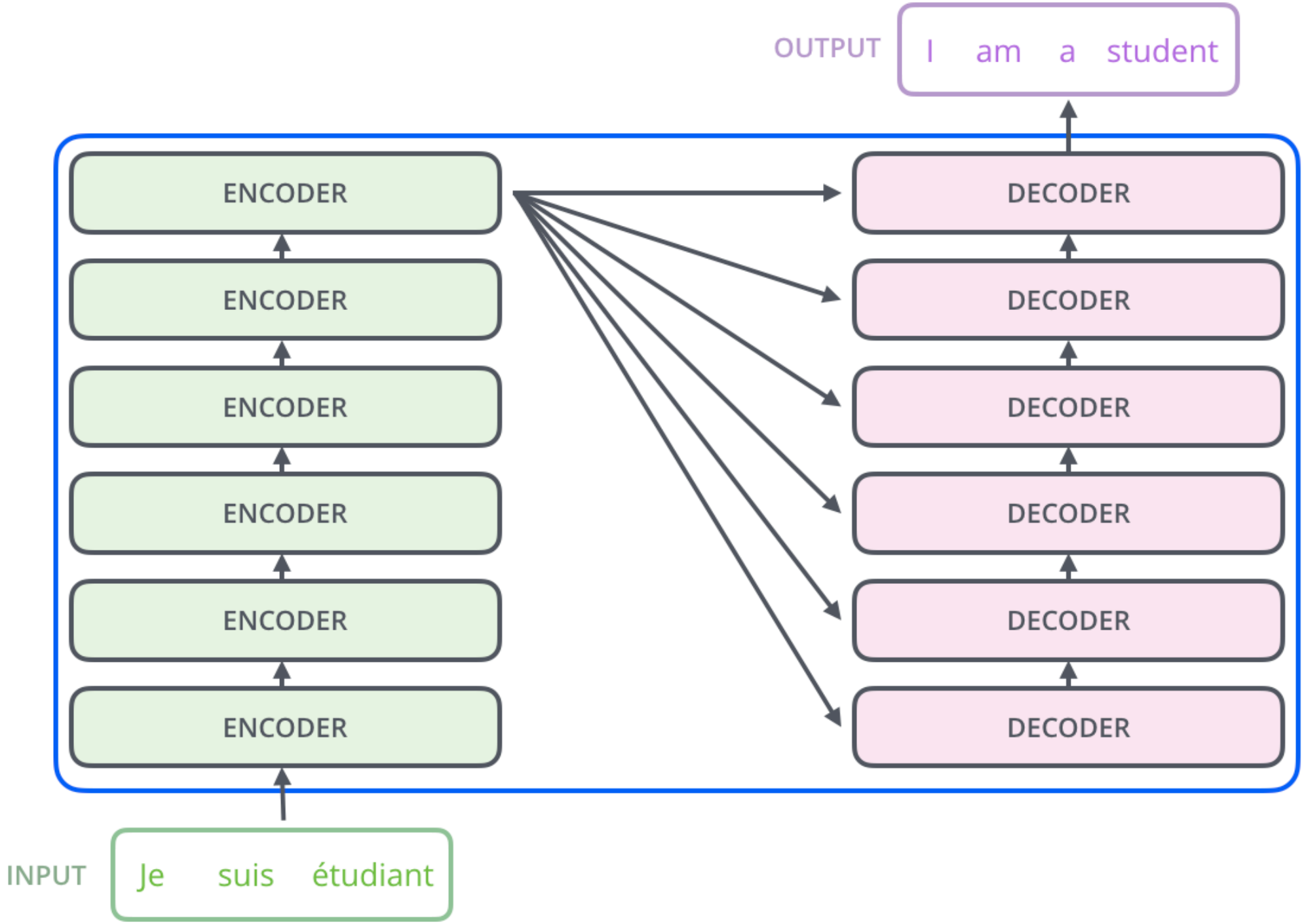
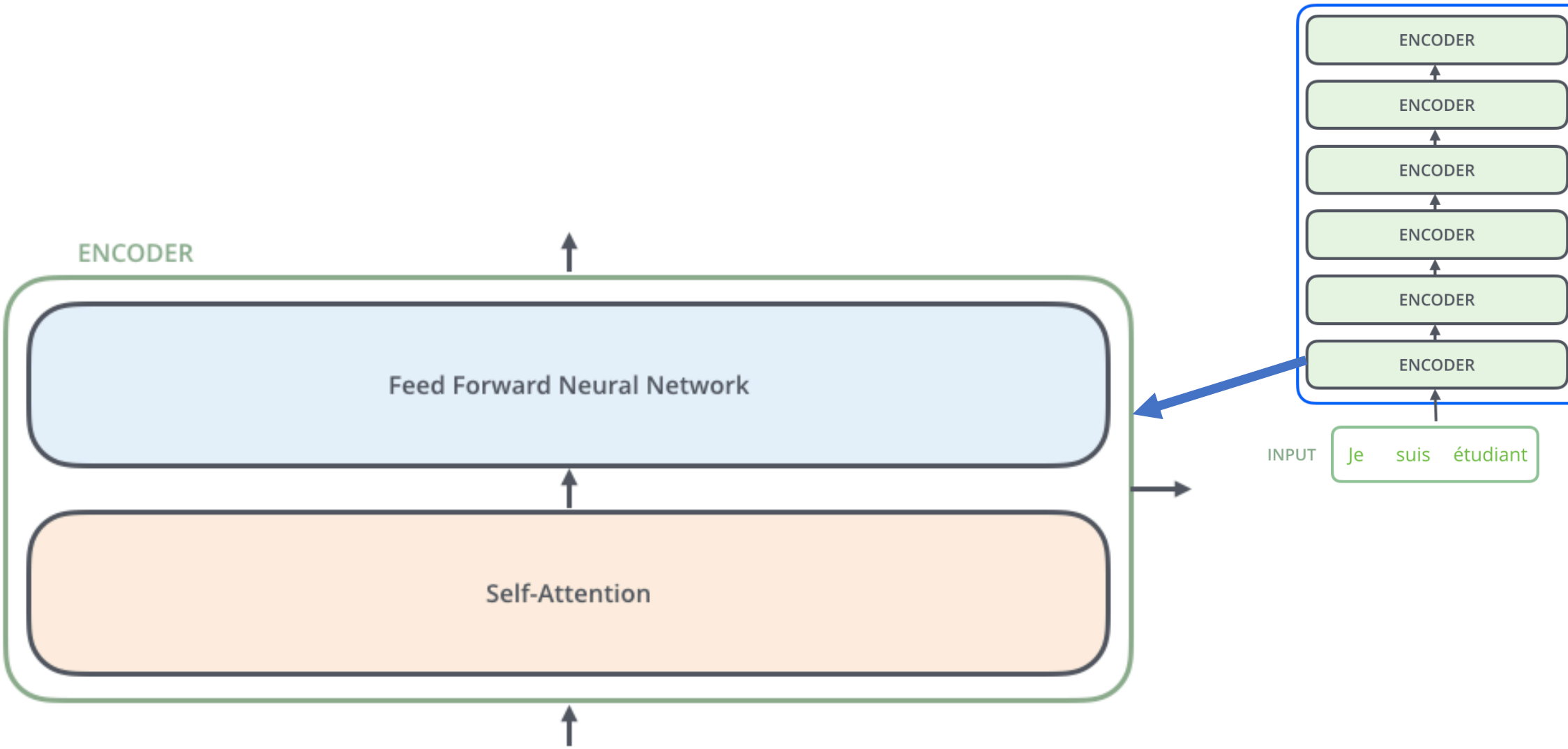

Transformer

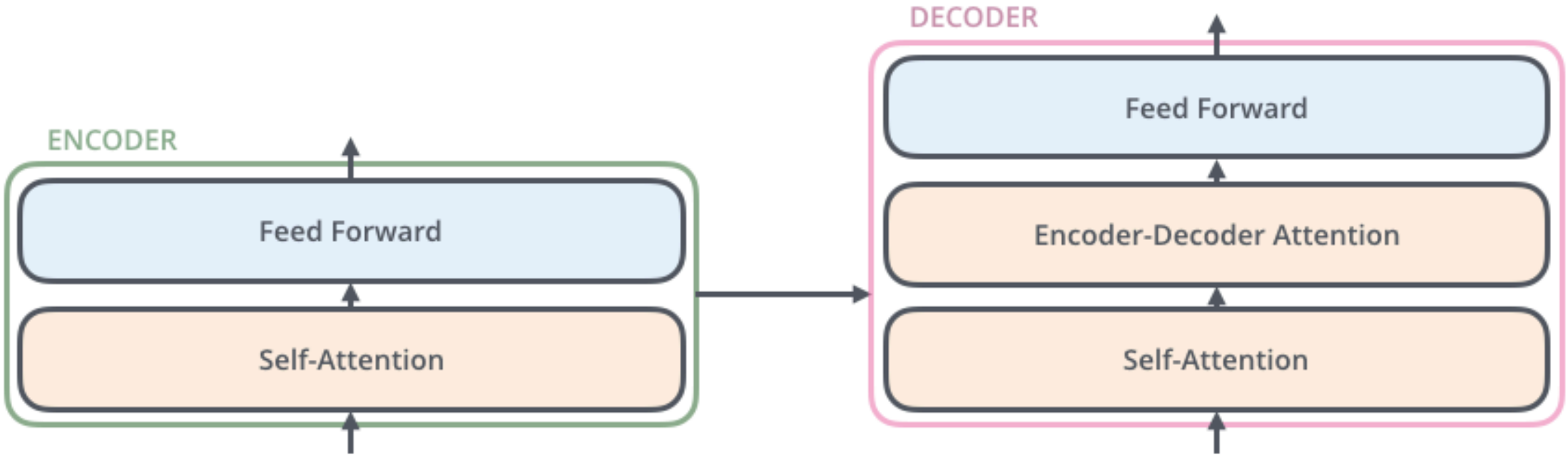
Seq2Seq Model

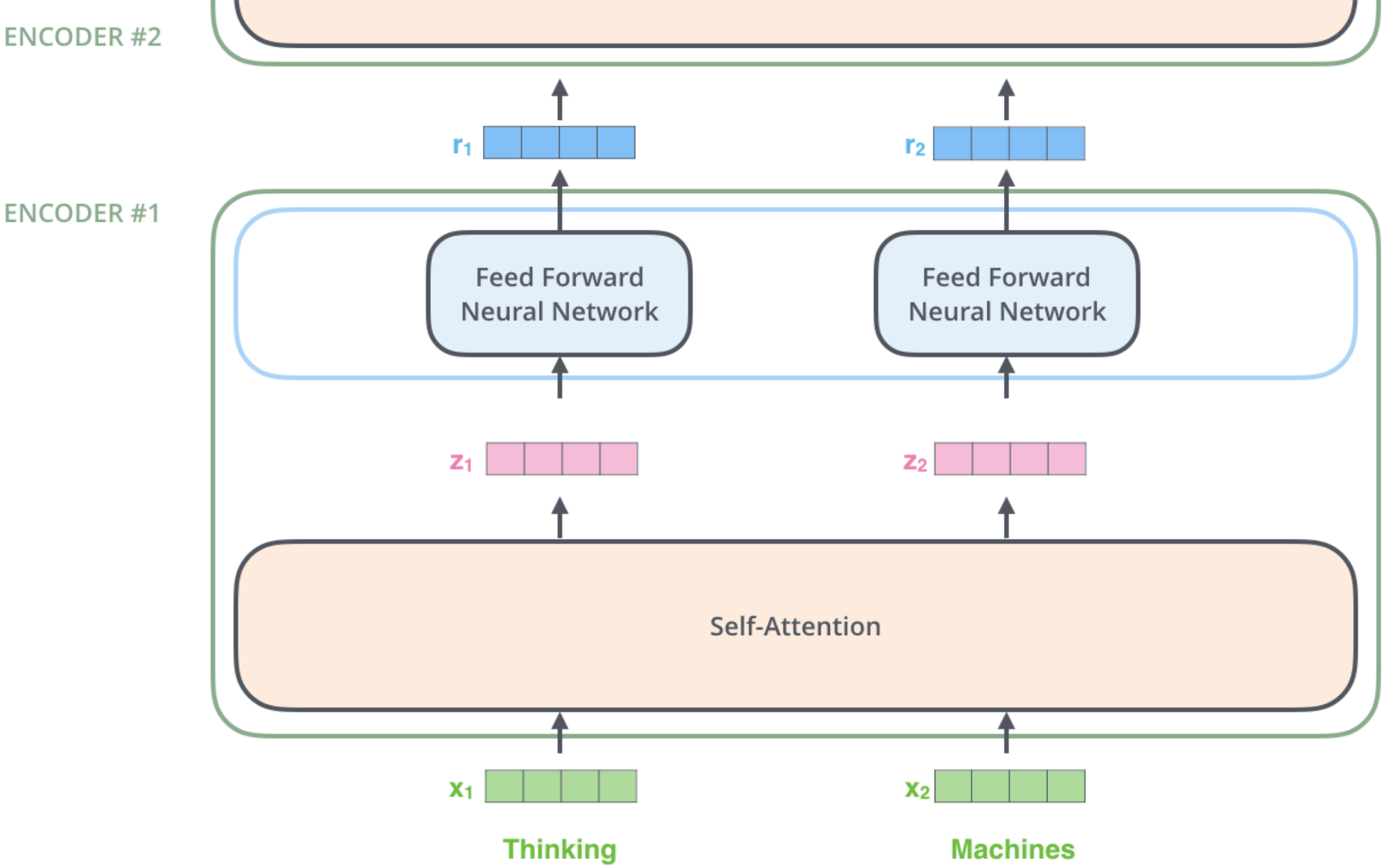


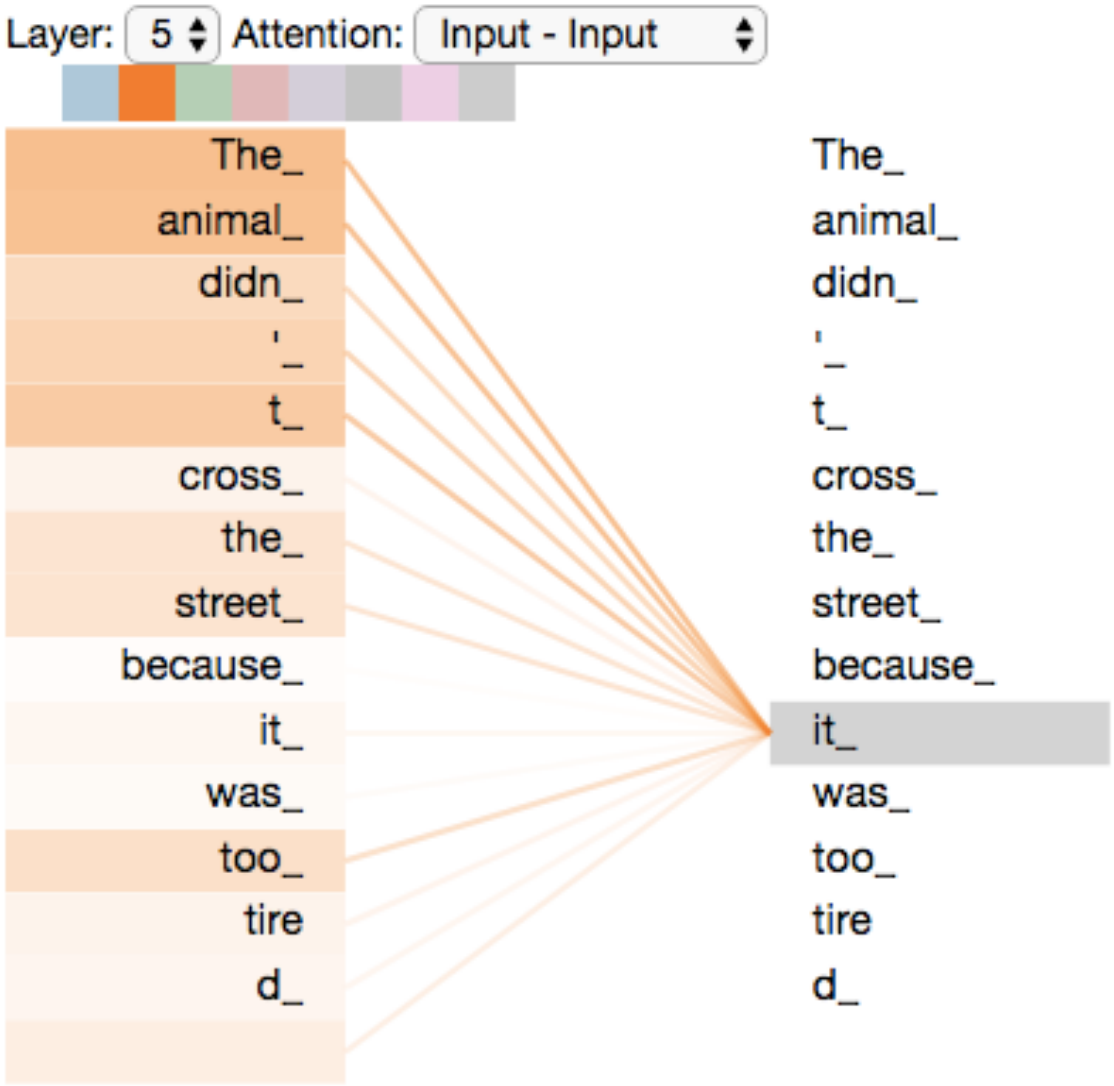

OUTPUT
I am a student
ENCODERS
DECODERS
INPUT
Je suis étudiant

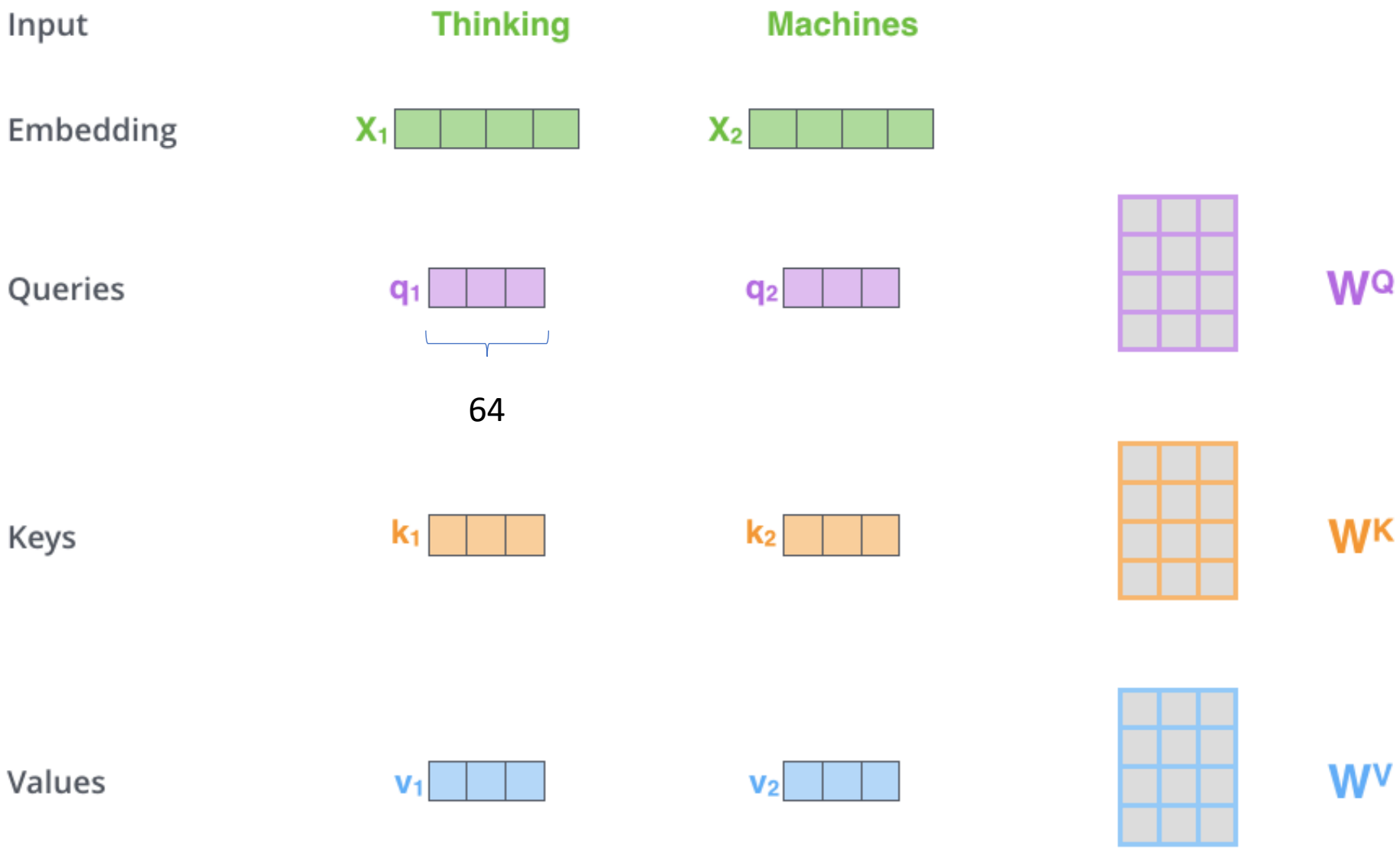

OUTPUT
I am a student
ENCODER
ENCODER
ENCODER
ENCODER
ENCODER
ENCODER
DECODER
DECODER
DECODER
DECODER
DECODER
DECODER
INPUT
Je suis étudiant

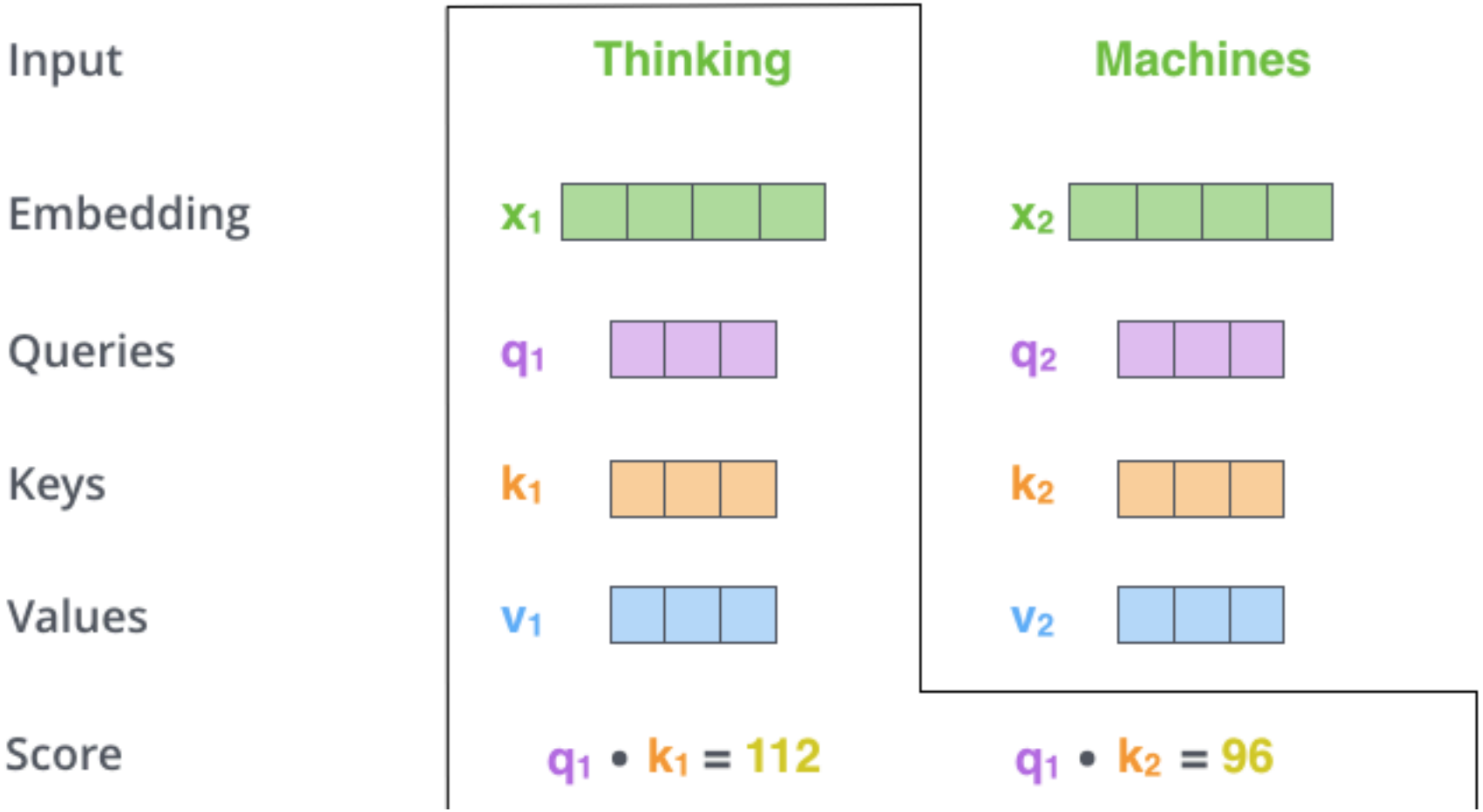

ENCODER
Feed Forward Neural Network
Self-Attention
ENCODER
ENCODER
ENCODER
ENCODER
ENCODER
ENCODER
INPUT
Je
suis
étudiant

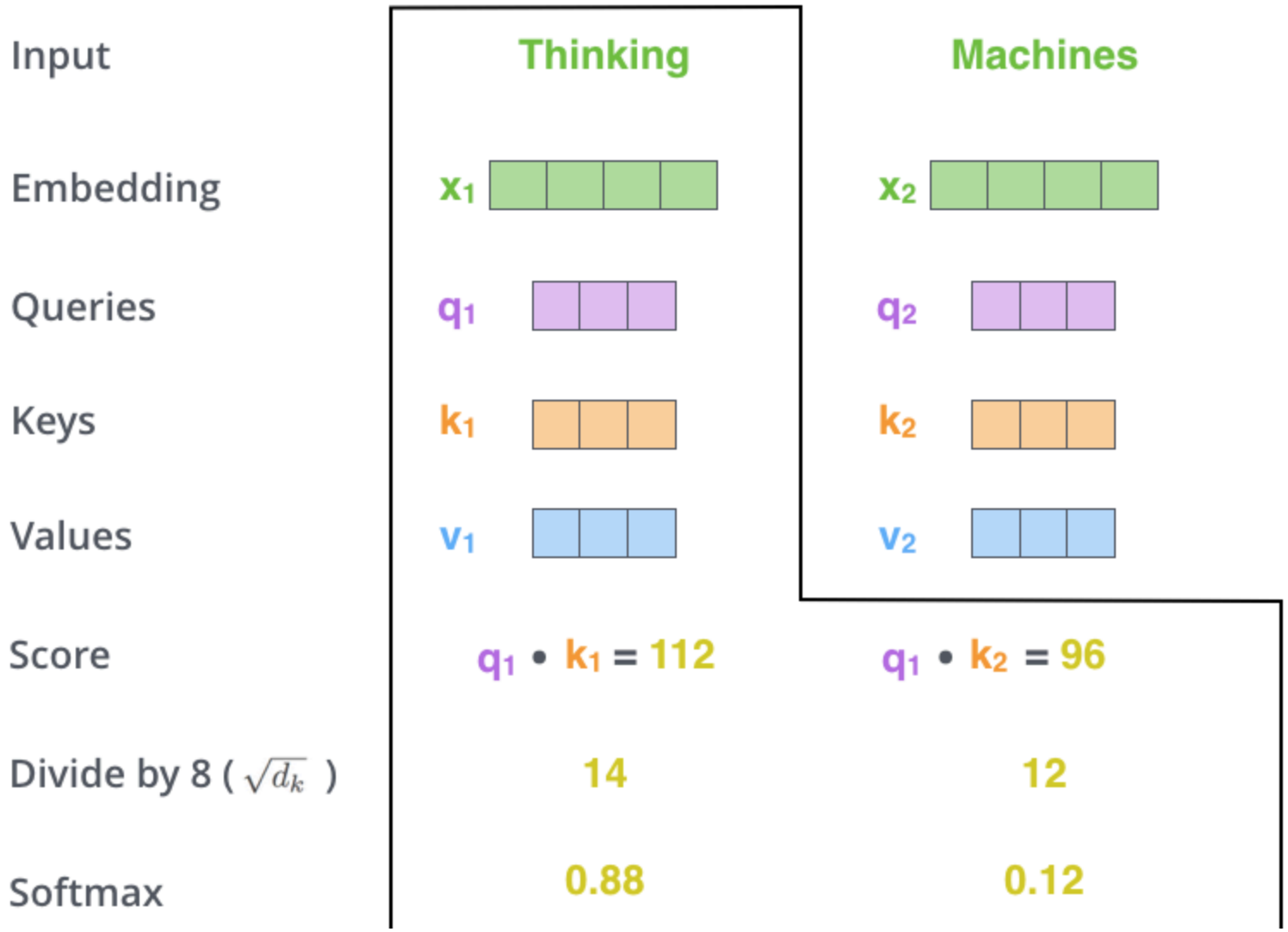

ENCODER
Feed Forward
Self-Attention
DECODER
Feed Forward
Encoder-Decoder Attention
Self-Attention

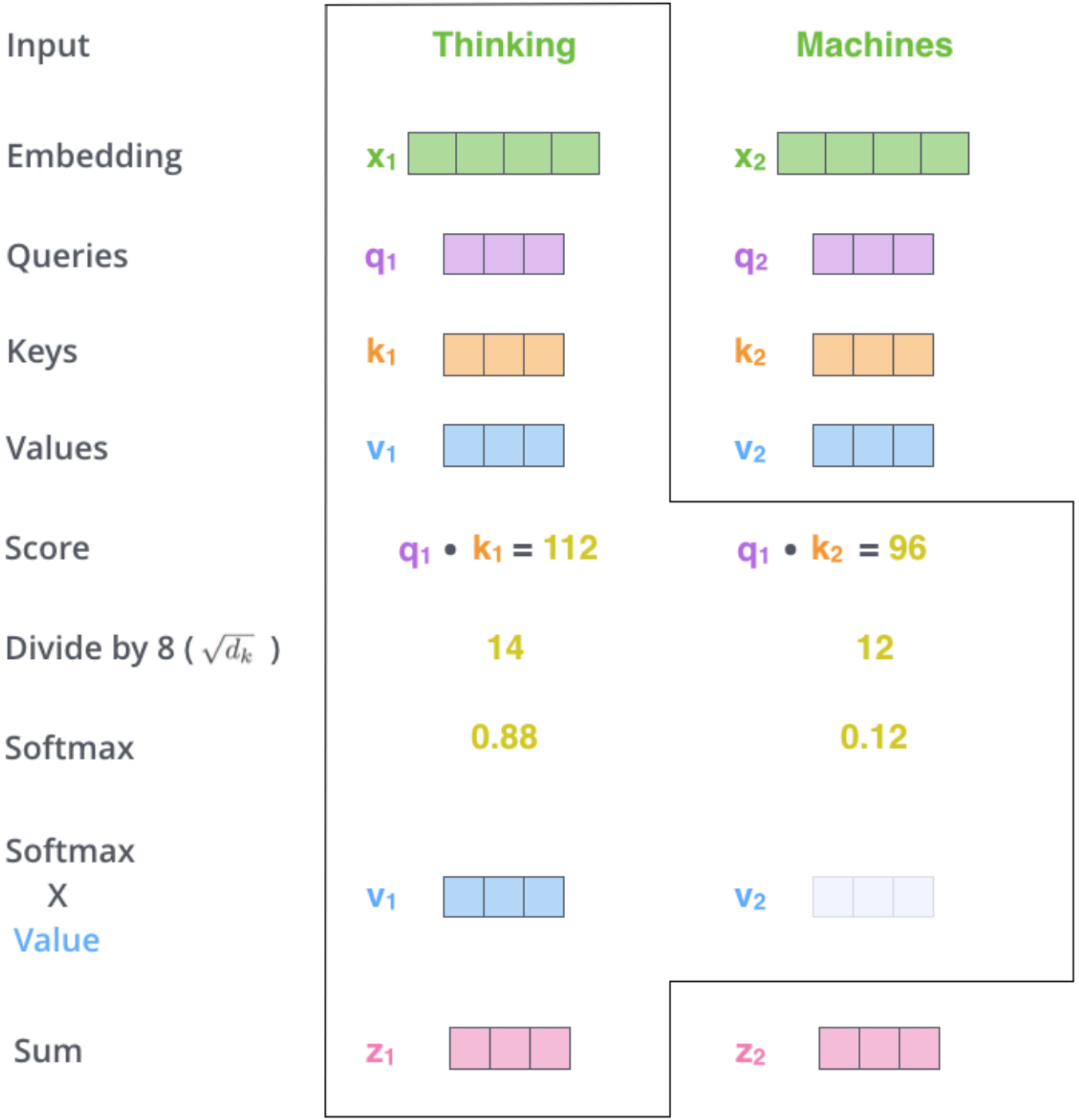

ENCODER #2
r1
r2
ENCODER #1
Feed Forward Neural Network
Feed Forward Neural Network
z1
z2
Self-Attention
x1
x2
Thinking
Machines

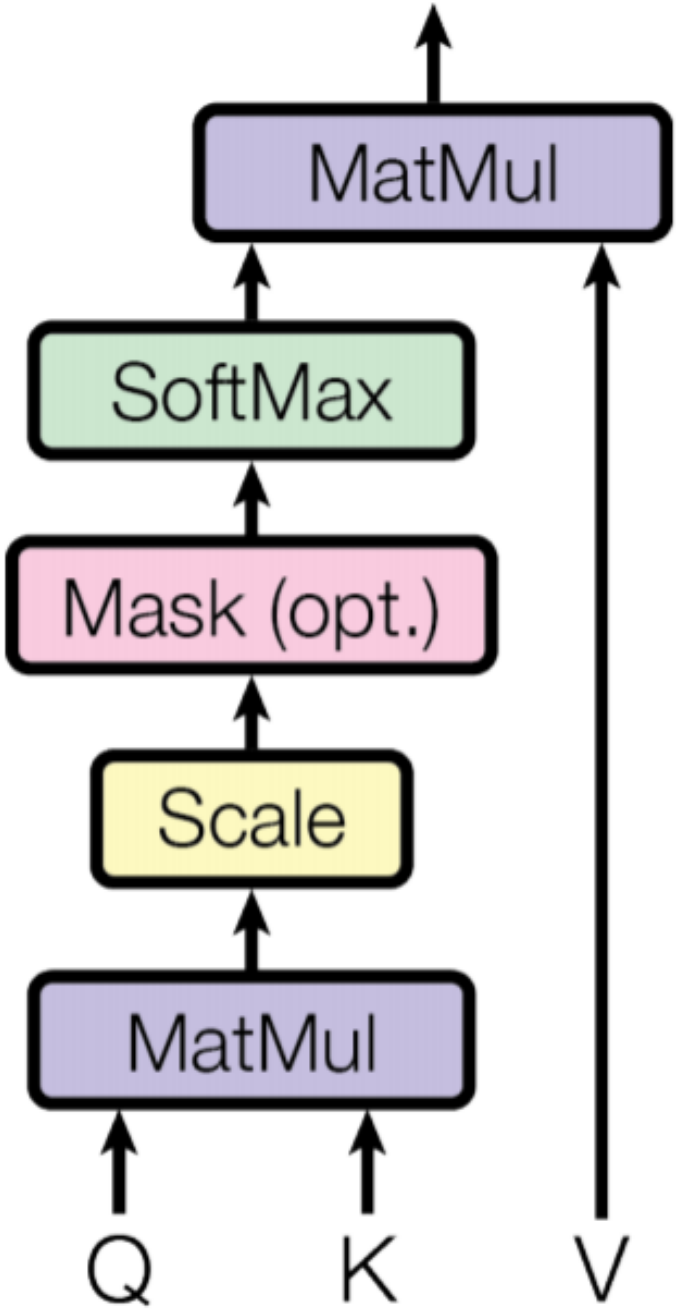

Layer: 5
Attention: Input - Input
The_
animal_
didn_
'_
t_
cross_
the_
street_
because_
it_
was_
too_
tire
d_
The_
animal_
didn_
'_
t_
cross_
the_
street_
because_
it_
was_
too_
tire
d_


Input
Thinking
Machines
Embedding
X1
X2
Queries
q1
q2
64
WQ
Keys
k1
k2
WK
Values
v1
v2
WV


Input
Thinking
Machines
Embedding
x_1
x_2
Queries
q_1
q_2
Keys
k_1
k_2
Values
v_1
v_2
Score
$q_1 \bullet k_1 = 112$
$q_1 \bullet k_2 = 96$

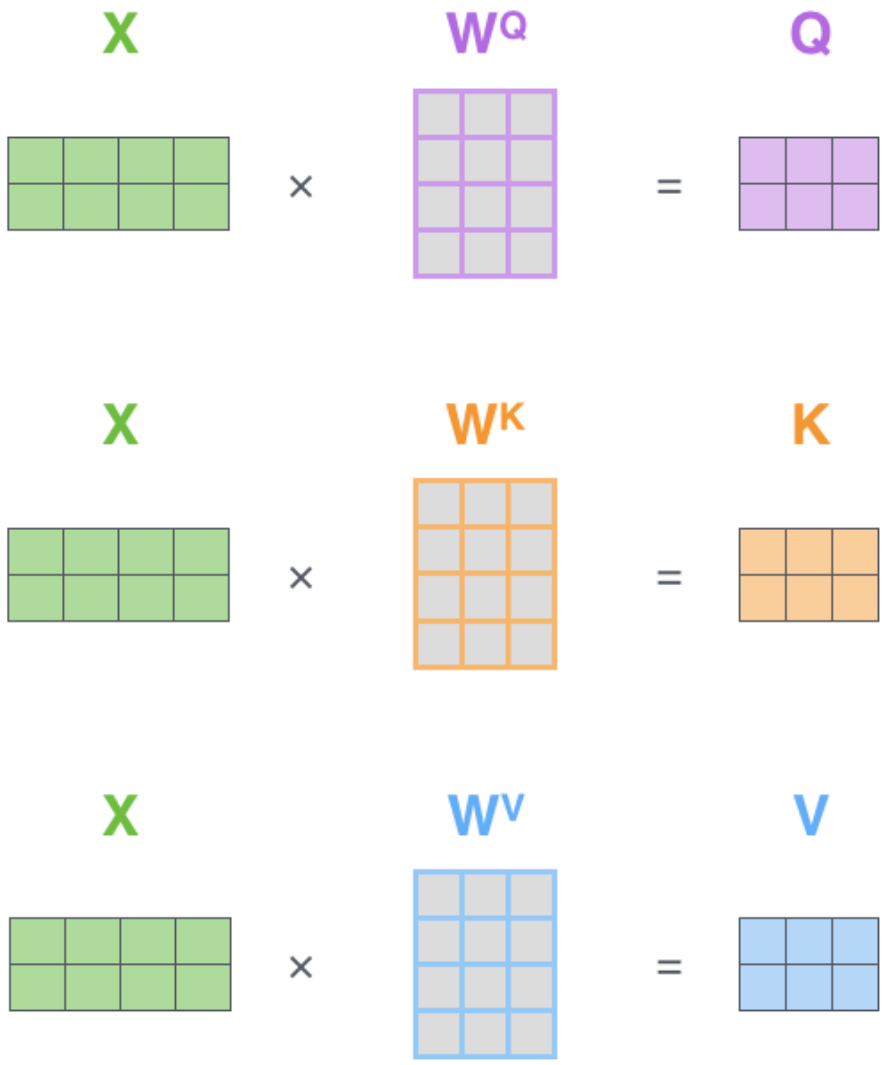

Input
Thinking
Machines
Embedding
x_1
x_2
Queries
q_1
q_2
Keys
k_1
k_2
Values
v_1
v_2
Score
$q_1 \bullet k_1 = 112$
$q_1 \bullet k_2 = 96$
Divide by 8 ($\sqrt{d_k}$)
14
12
Softmax
0.88
0.12

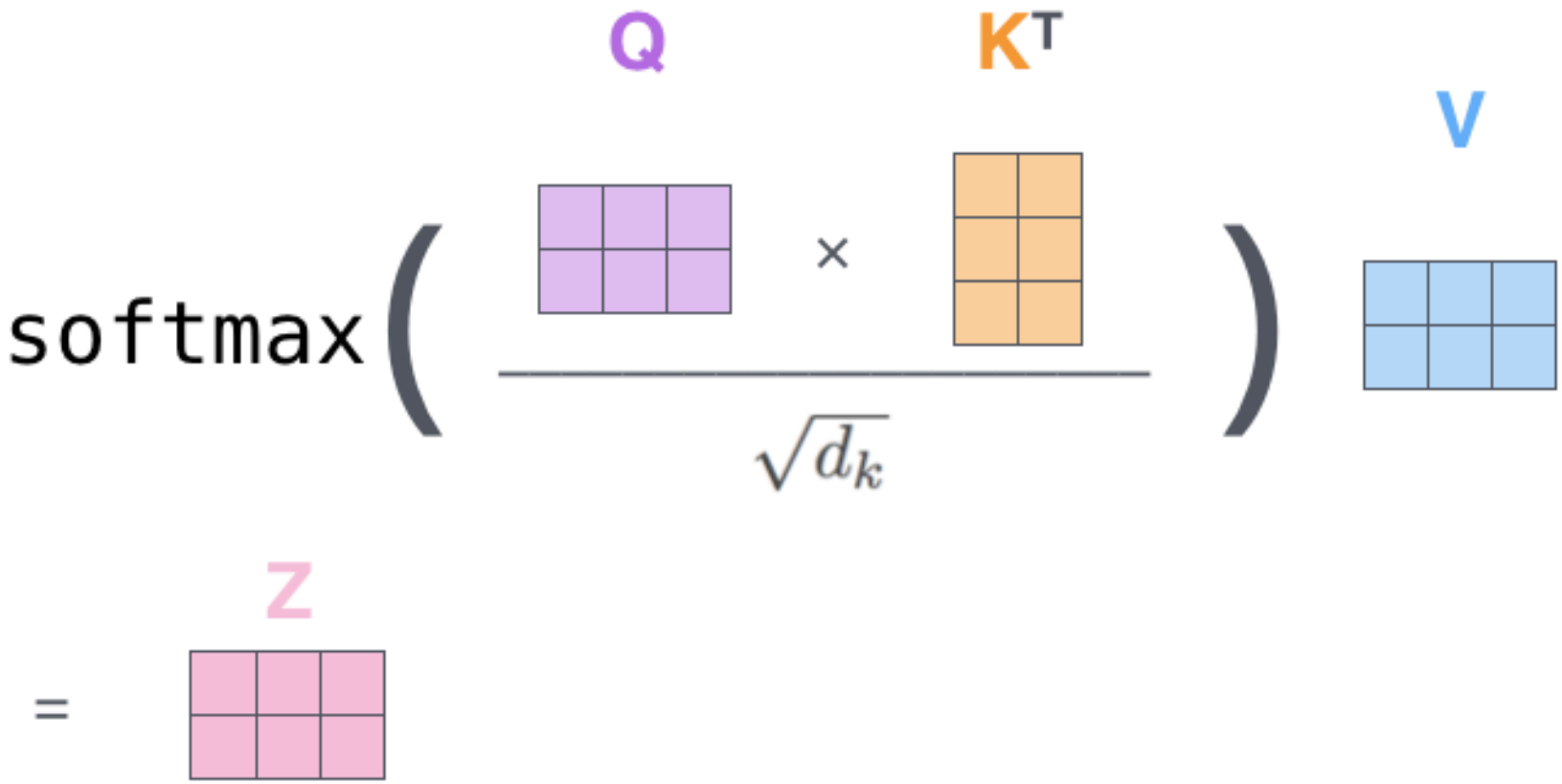

Input
Thinking
Machines
Embedding
x1
x2
Queries
q1
q2
Keys
k1
k2
Values
v1
v2
Score
q1 • k1 = 112
q1 • k2 = 96
Divide by 8 (√dk)
14
12
Softmax
0.88
0.12
Softmax X Value
v1
v2
Sum
z1
z2

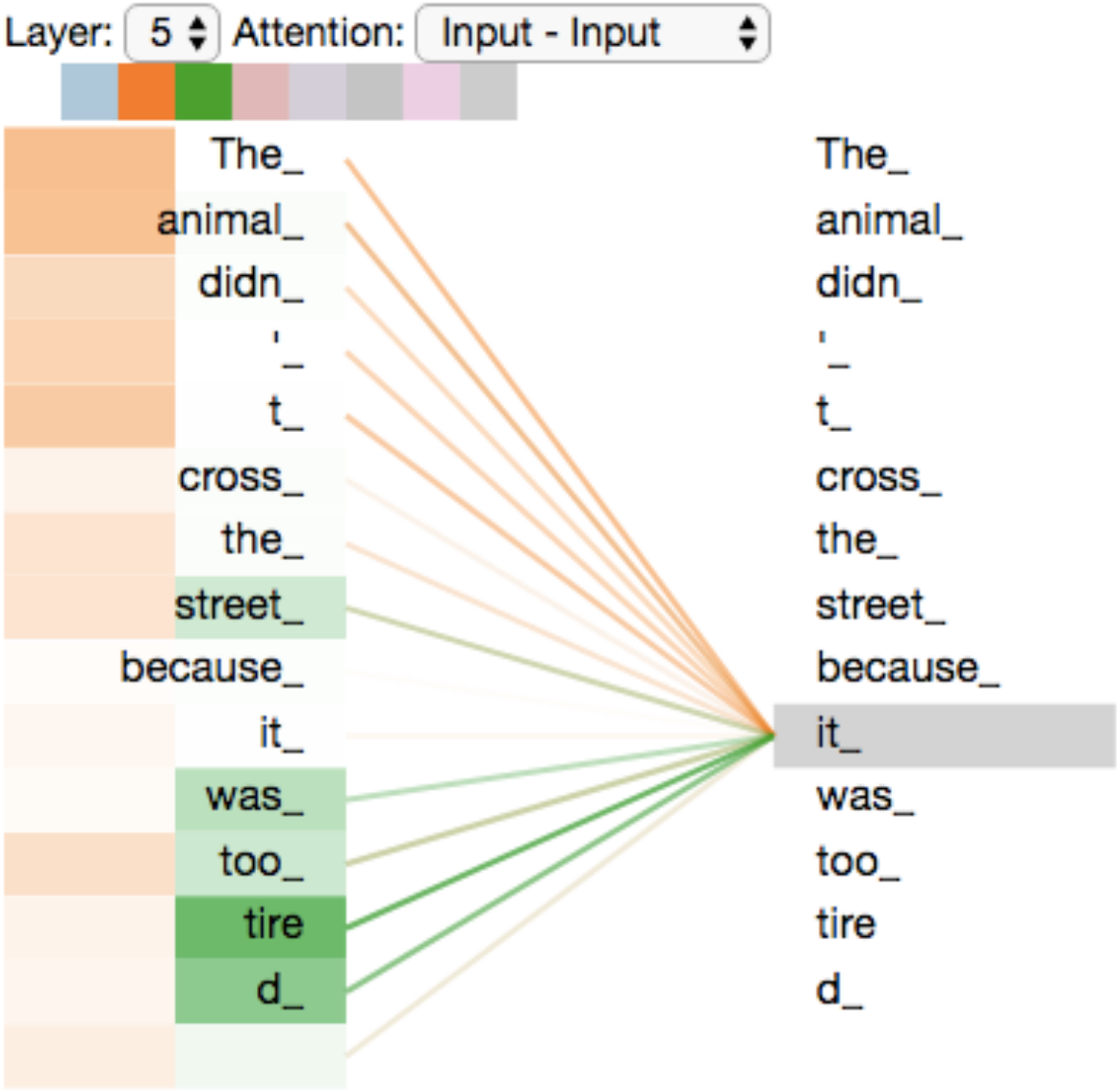


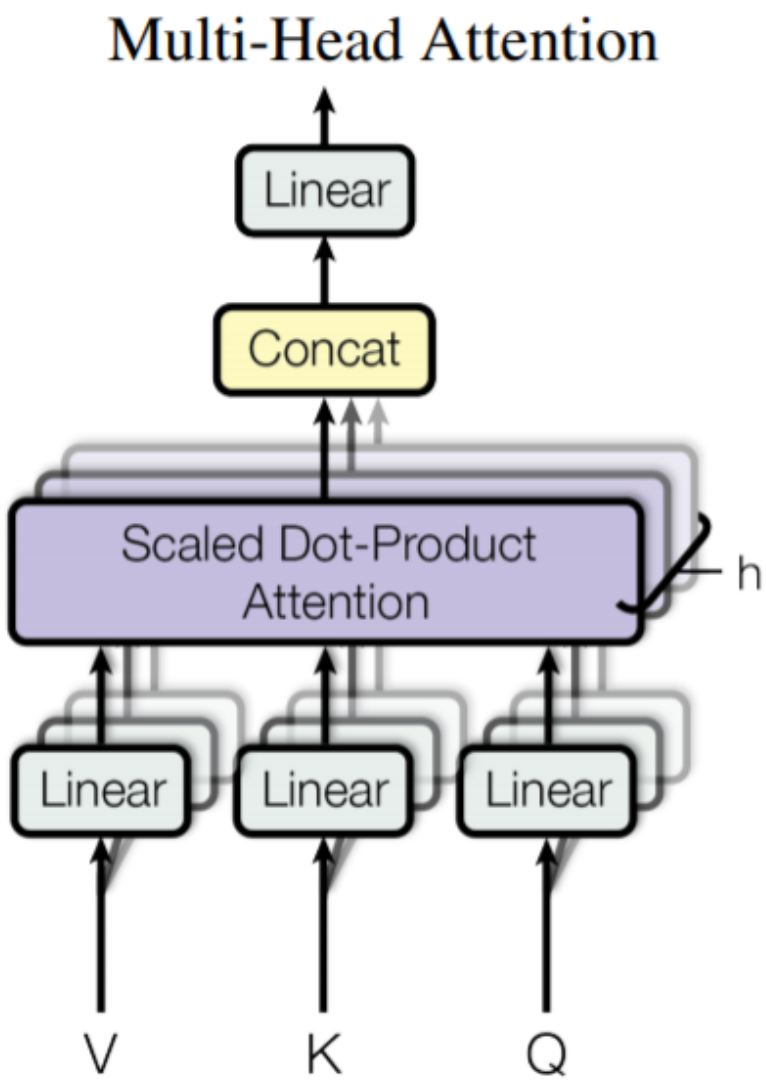
Scaled Dot-Product Attention

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


X
W^Q
Q
×
=
X
W^K
K
×
=
X
W^V
V
×
=


Q
K^T
V
softmax
×
$\sqrt{d_k}$
Z
=


Layer: 5
Attention: Input - Input
The_
animal_
didn_
'_
t_
cross_
the_
street_
because_
it_
was_
too_
tire
d_
The_
animal_
didn_
'_
t_
cross_
the_
street_
because_
it_
was_
too_
tire
d_


Multi-Head Attention
Linear
Concat
Scaled Dot-Product
Attention
h
Linear
Linear
Linear
V
K
Q

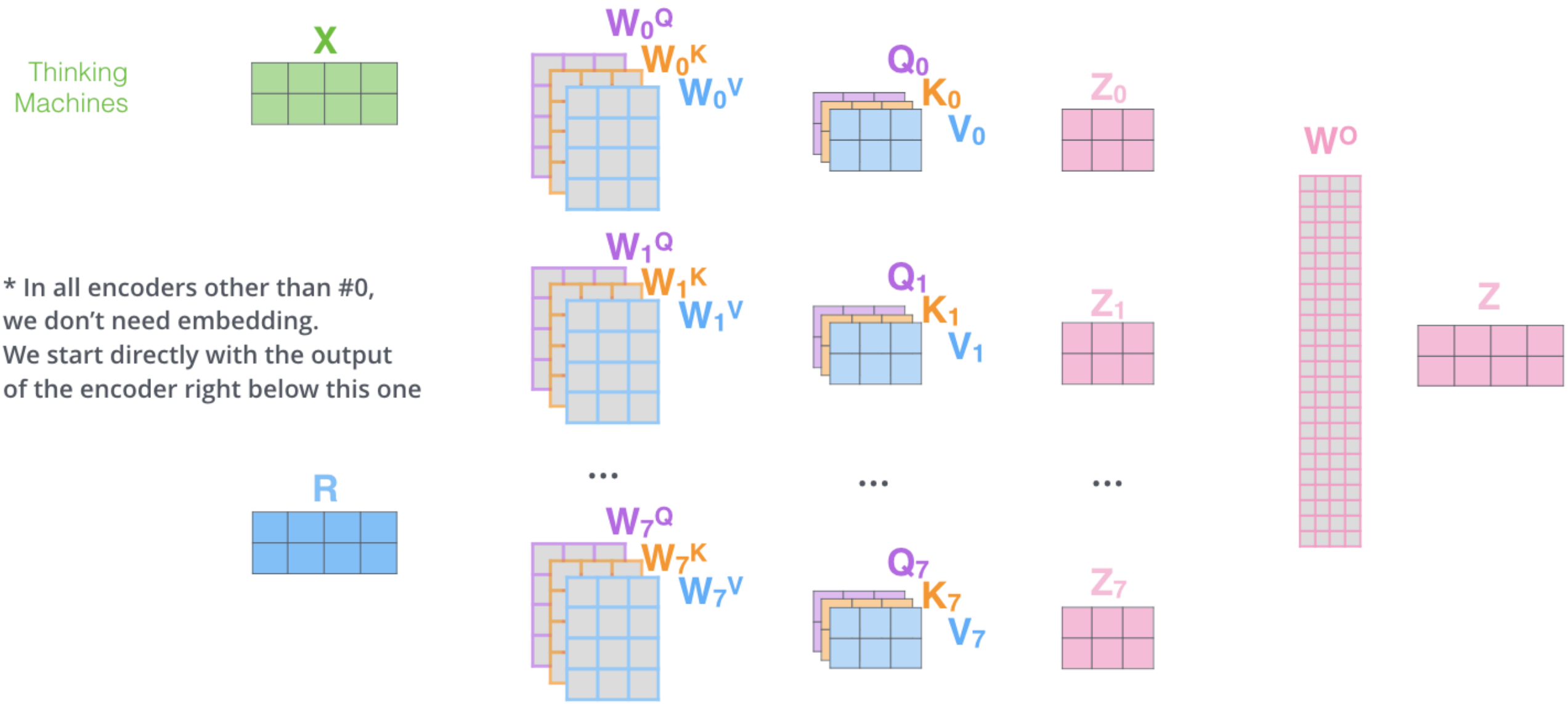
1) This is our input sentence*

2) We embed each word*

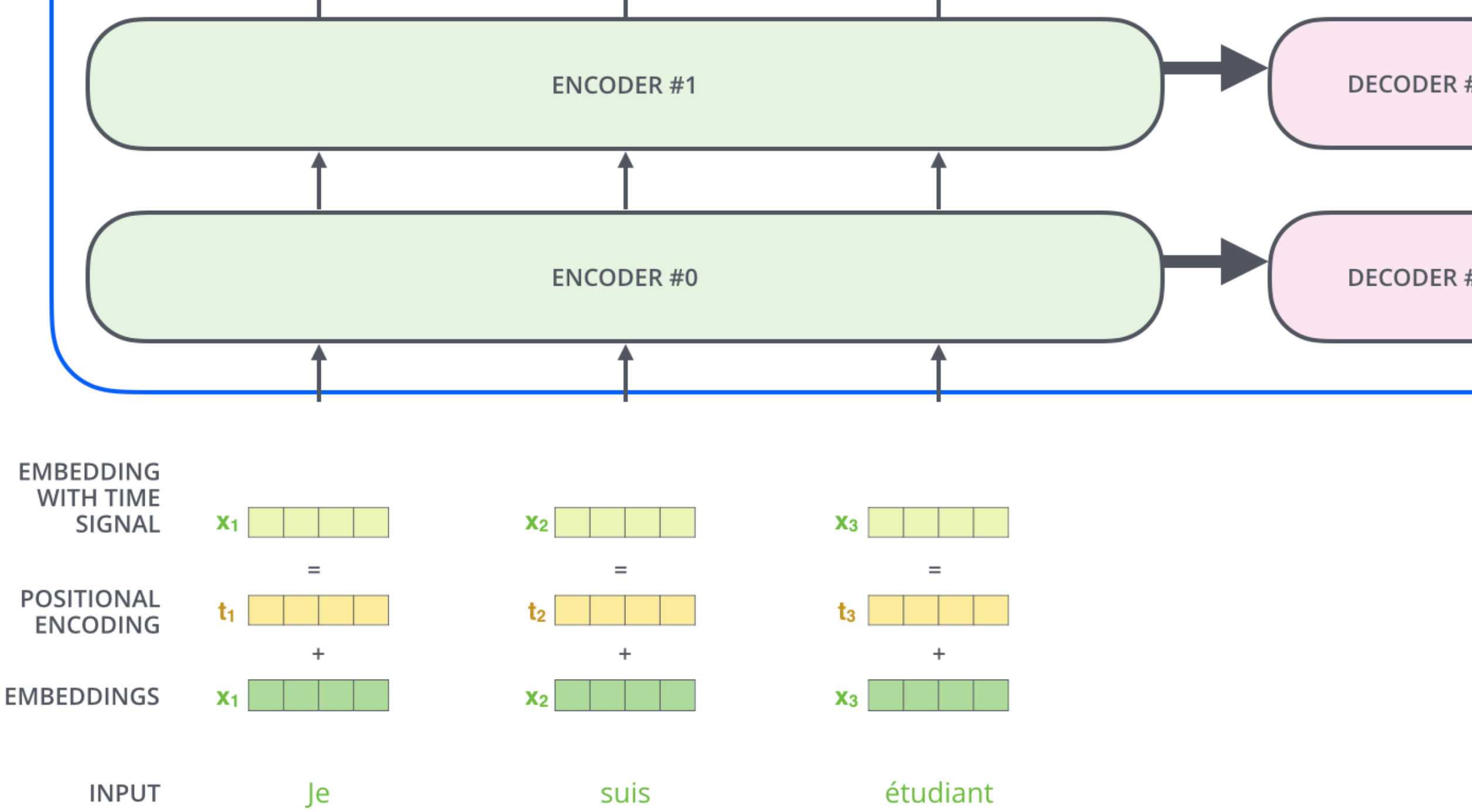
3) Split into 8 heads. We multiply X or R with weight matrices

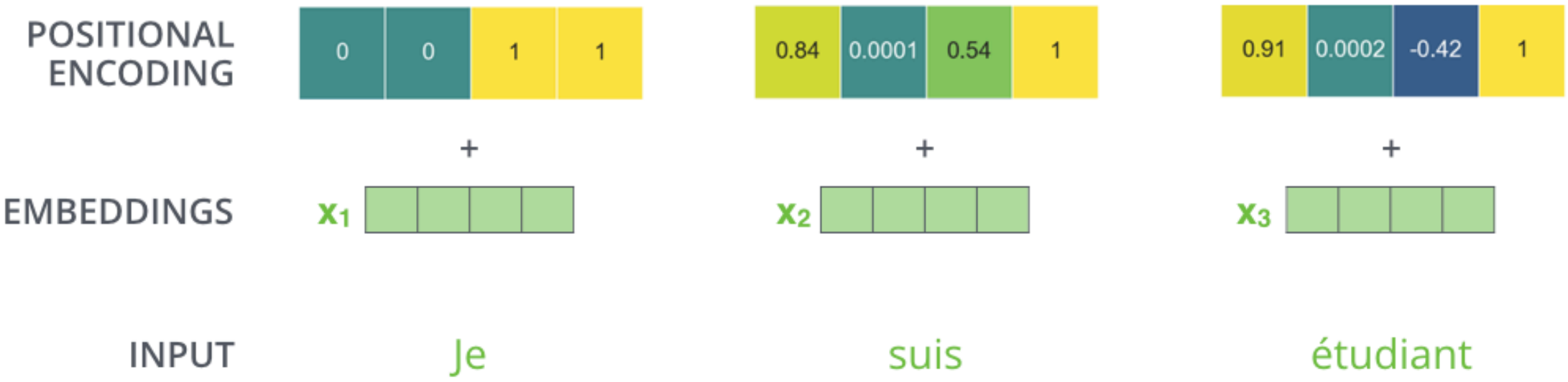
4) Calculate attention using the resulting Q/K/V matrices

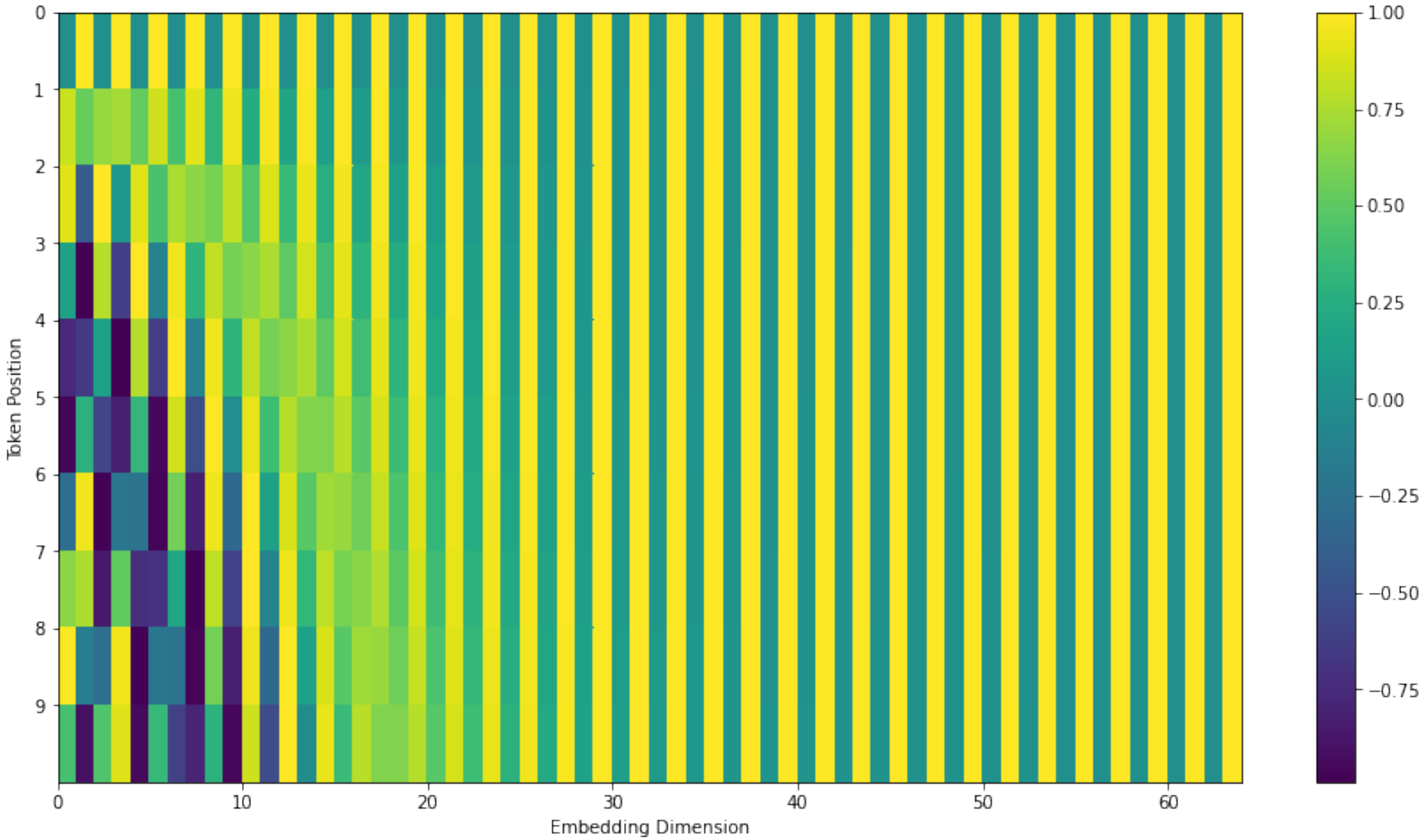
5) Concatenate the resulting Z matrices, then multiply with weight matrix W^O to produce the output of the layer

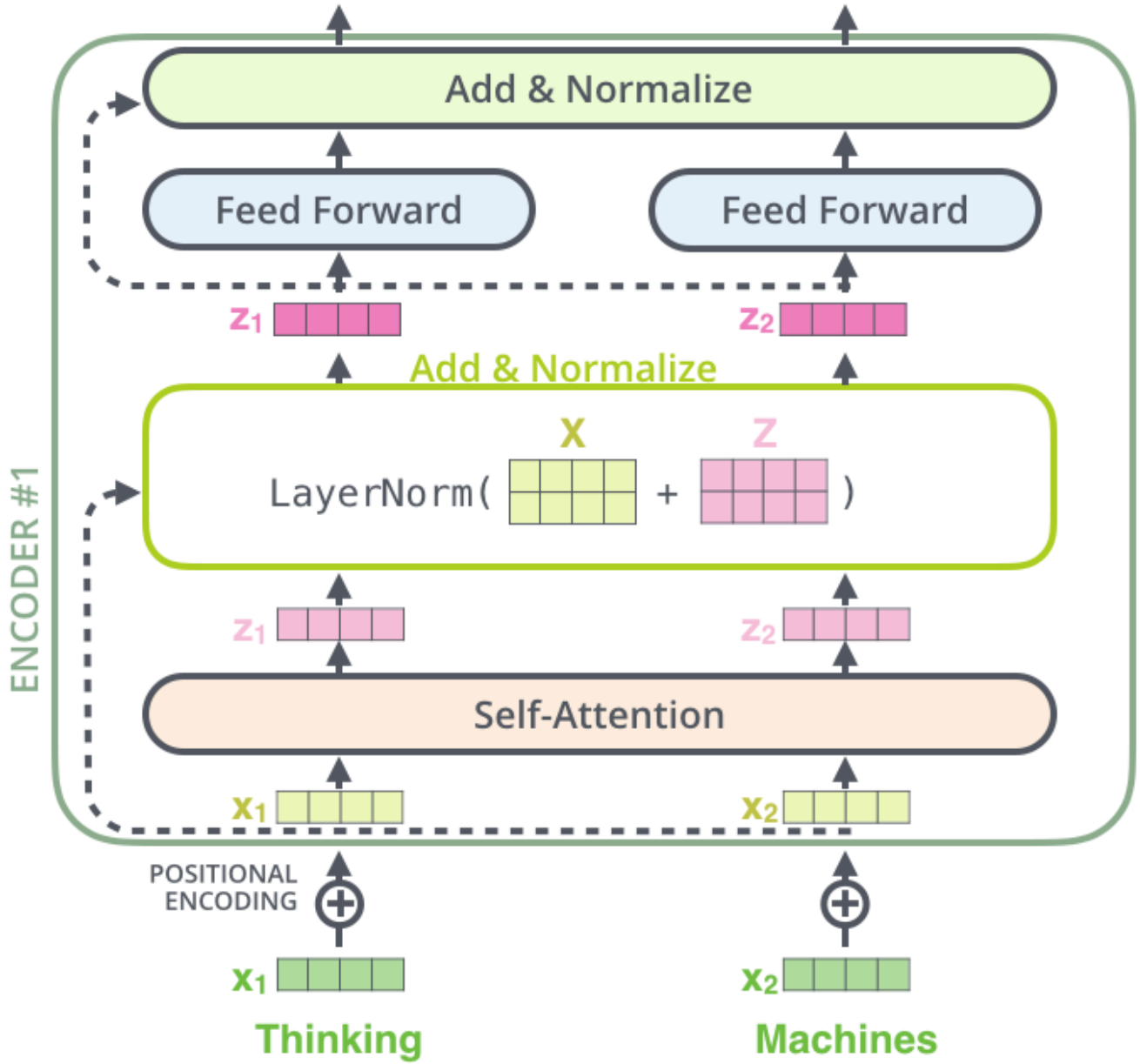


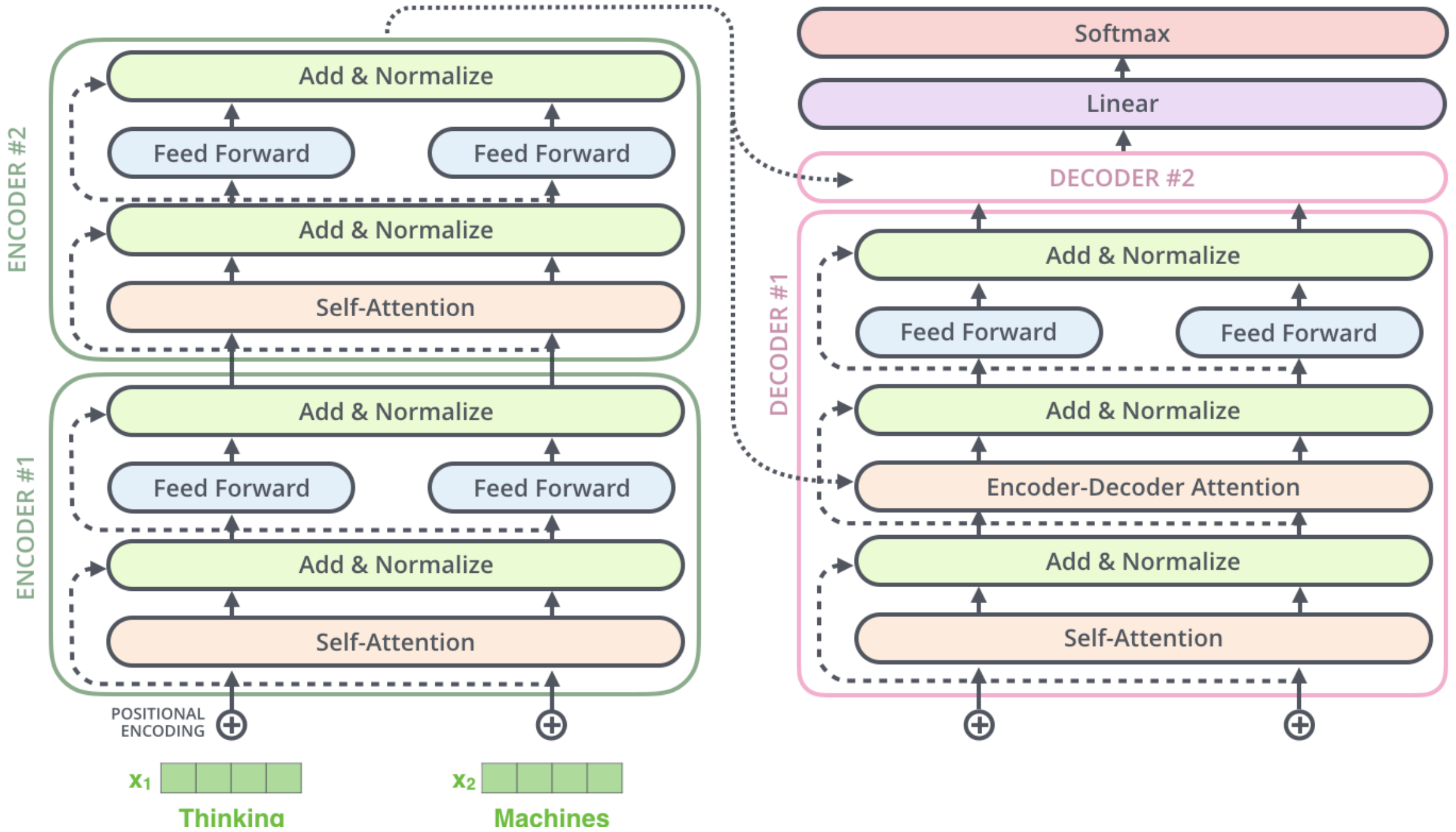
* In all encoders other than #0, we don't need embedding. We start directly with the output of the encoder right below this one

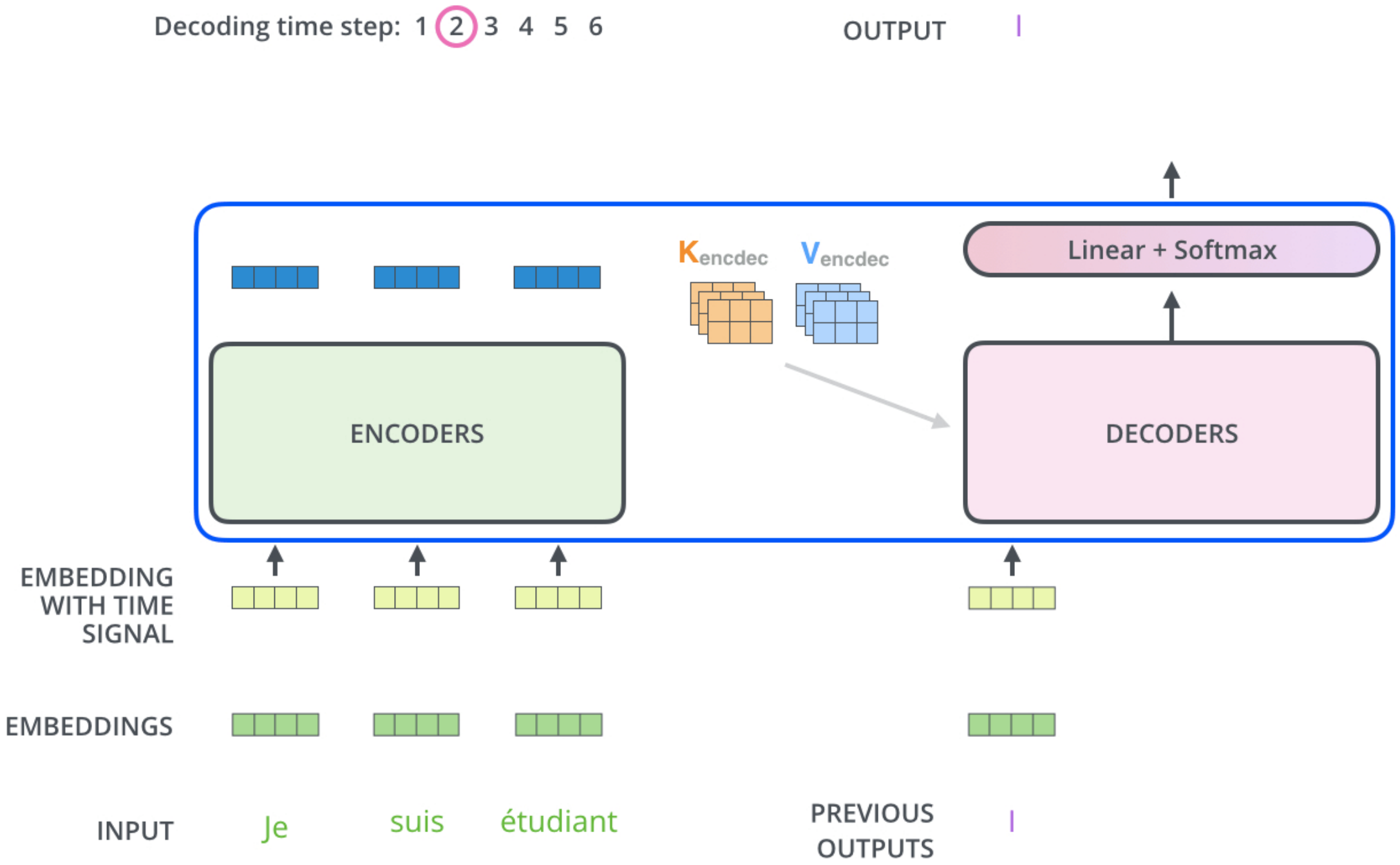

ENCODER #1
DECODER #
ENCODER #0
DECODER #
EMBEDDING WITH TIME SIGNAL
POSITIONAL ENCODING
EMBEDDINGS
INPUT
x1
x2
x3
=
t1
t2
t3
+
Je
suis
étudiant


POSITIONAL ENCODING
0
0
1
1
0.84
0.0001
0.54
1
0.91
0.0002
-0.42
1
+
+
+
EMBEDDINGS
x1
x2
x3
INPUT
Je
suis
étudiant


Token Position
0
1
2
3
4
5
6
7
8
9
Embedding Dimension
0
10
20
30
40
50
60
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75


Add & Normalize
Feed Forward
Feed Forward
z1
z2
Add & Normalize
X
Z
LayerNorm(+)
ENCODER #1
z1
z2
Self-Attention
x1
x2
POSITIONAL ENCODING
x1
x2
Thinking
Machines


Softmax
Linear
DECODER #2
DECODER #1
Add & Normalize
Feed Forward
Feed Forward
Add & Normalize
Encoder-Decoder Attention
Add & Normalize
Self-Attention
ENCODER #2
Add & Normalize
Feed Forward
Feed Forward
Add & Normalize
Self-Attention
ENCODER #1
Add & Normalize
Feed Forward
Feed Forward
Add & Normalize
Self-Attention
POSITIONAL ENCODING
x1
x2
Thinking
Machines


Decoding time step: 1 2 3 4 5 6
OUTPUT
I
Linear + Softmax
Kencdec
Vencdec
ENCODERS
DECODERS
EMBEDDING WITH TIME SIGNAL
EMBEDDINGS
INPUT
Je
suis
étudiant
PREVIOUS OUTPUTS
I

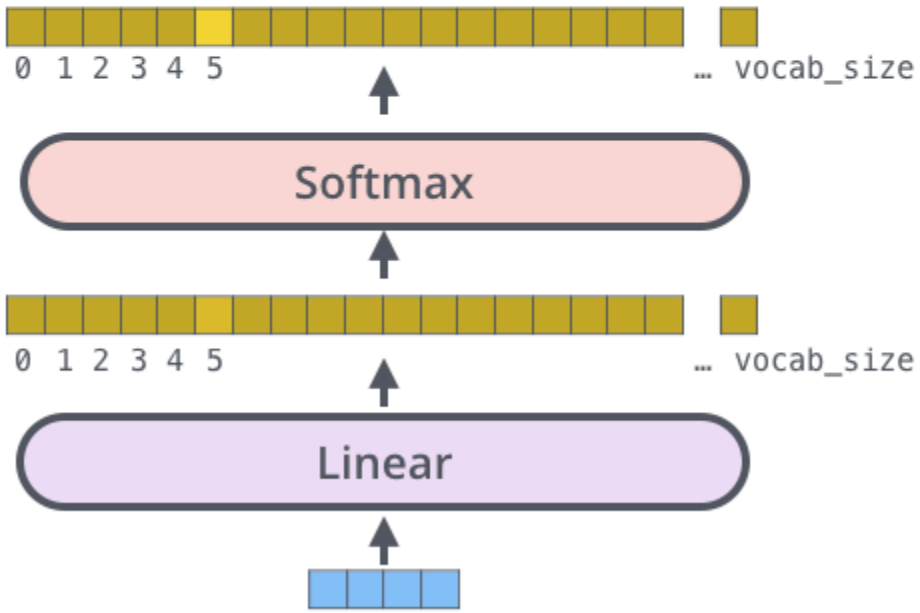
Which word in our vocabulary is associated with this index?

am

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

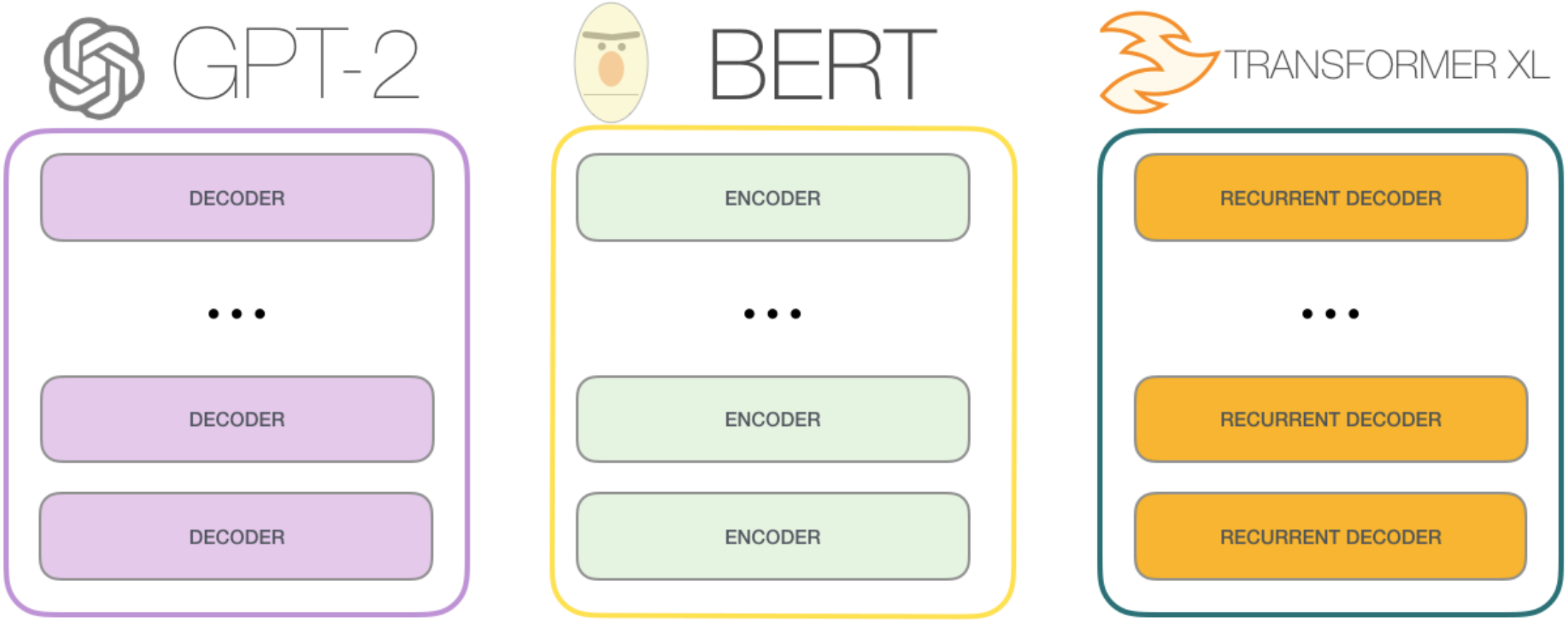
5

`log_probs`



`logits`

Decoder stack output


GPT-2
DECODER
...
DECODER
DECODER
BERT
ENCODER
...
ENCODER
ENCODER
TRANSFORMER XL
RECURRENT DECODER
...
RECURRENT DECODER
RECURRENT DECODER

Reference

- “Attention Is All You Need” paper. https://arxiv.org/pdf/1706.03762.pdf
- Visualization: http://jalammar.github.io/illustrated-transformer/