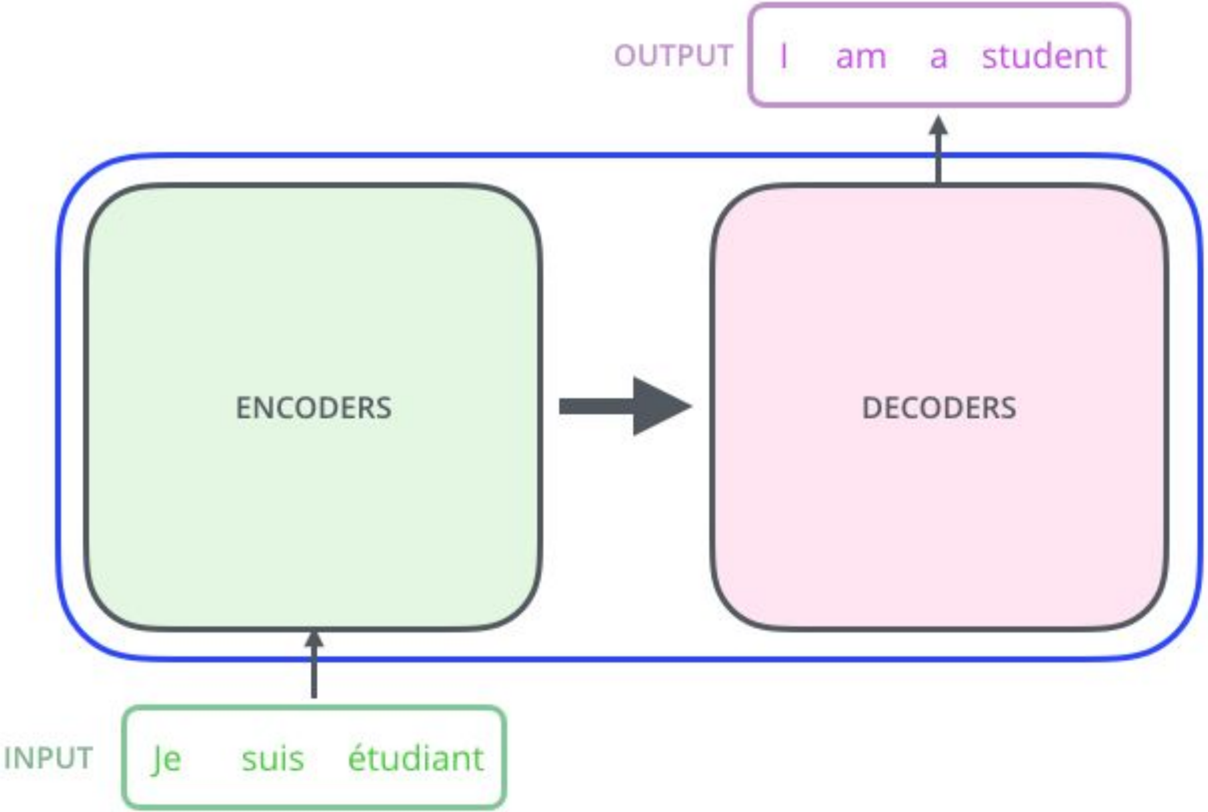
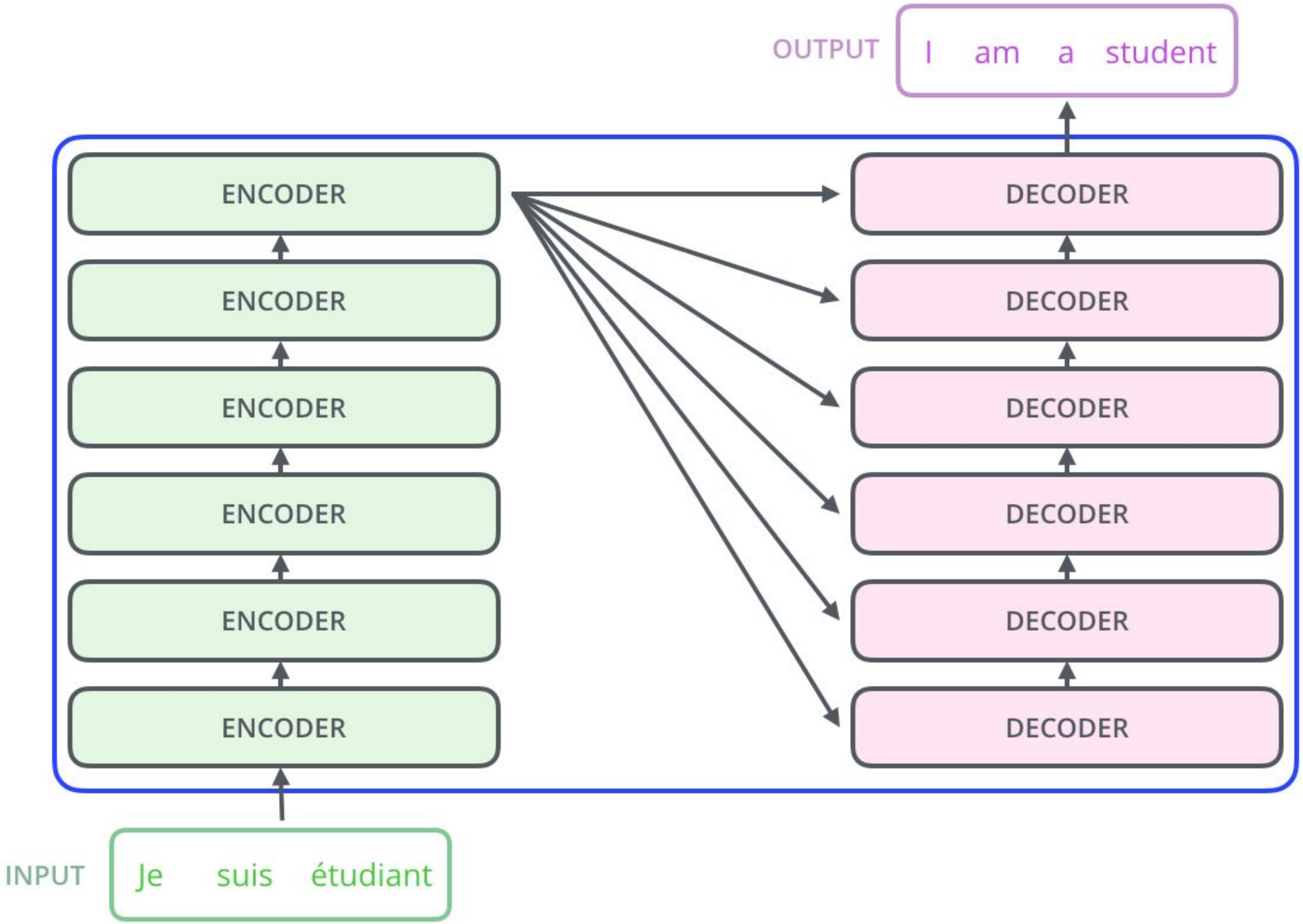
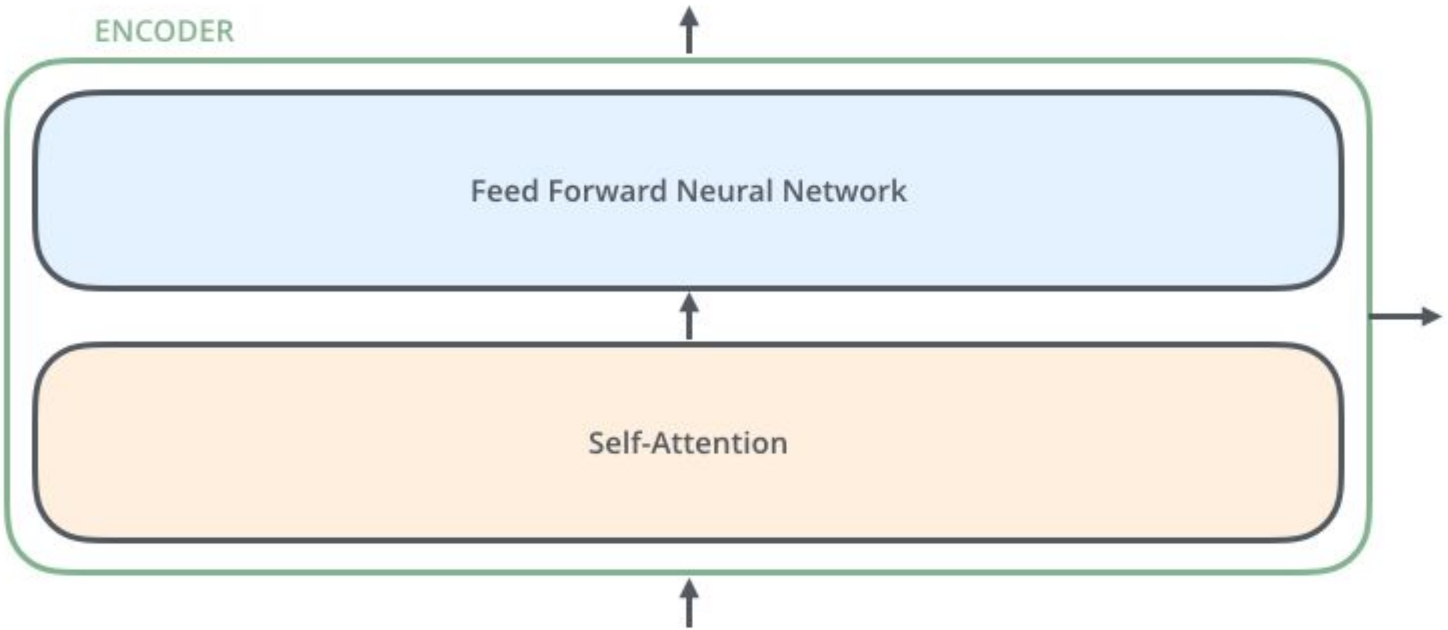
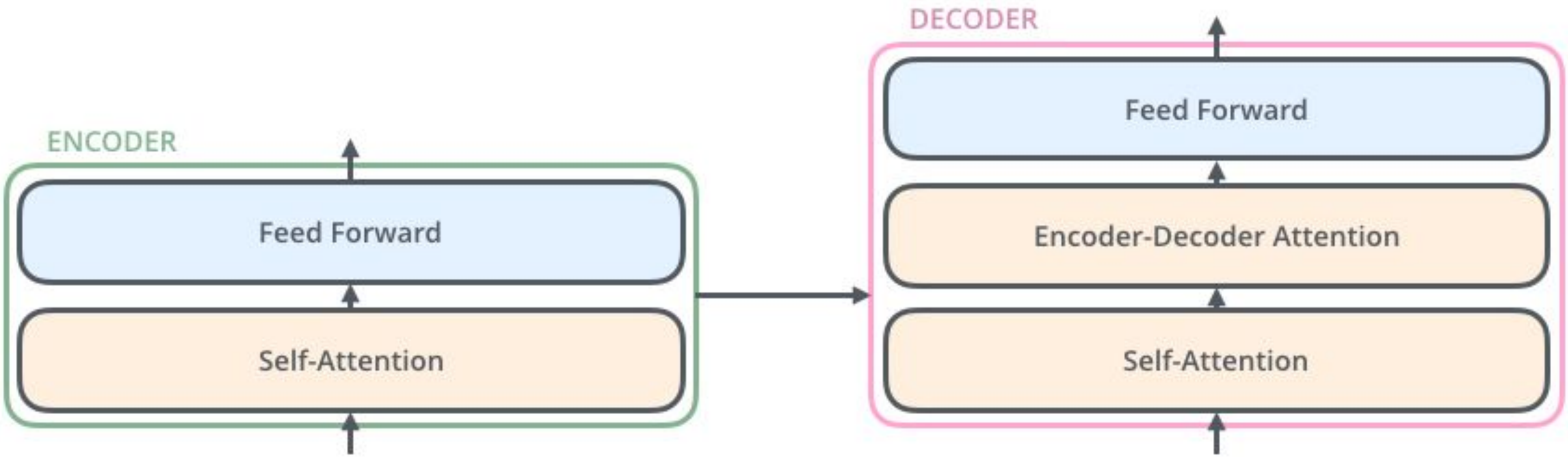
Transformer

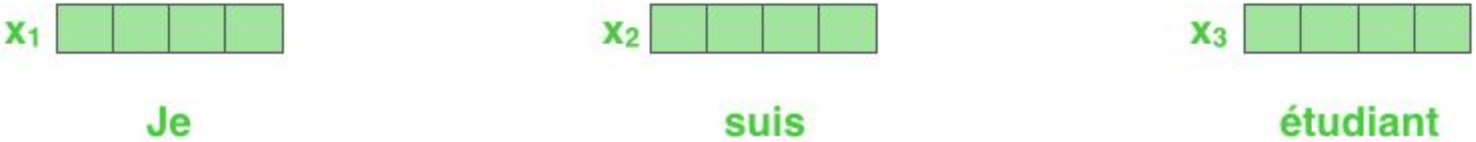

INPUT
Je suis étudiant
THE
TRANSFORMER
OUTPUT
I am a student

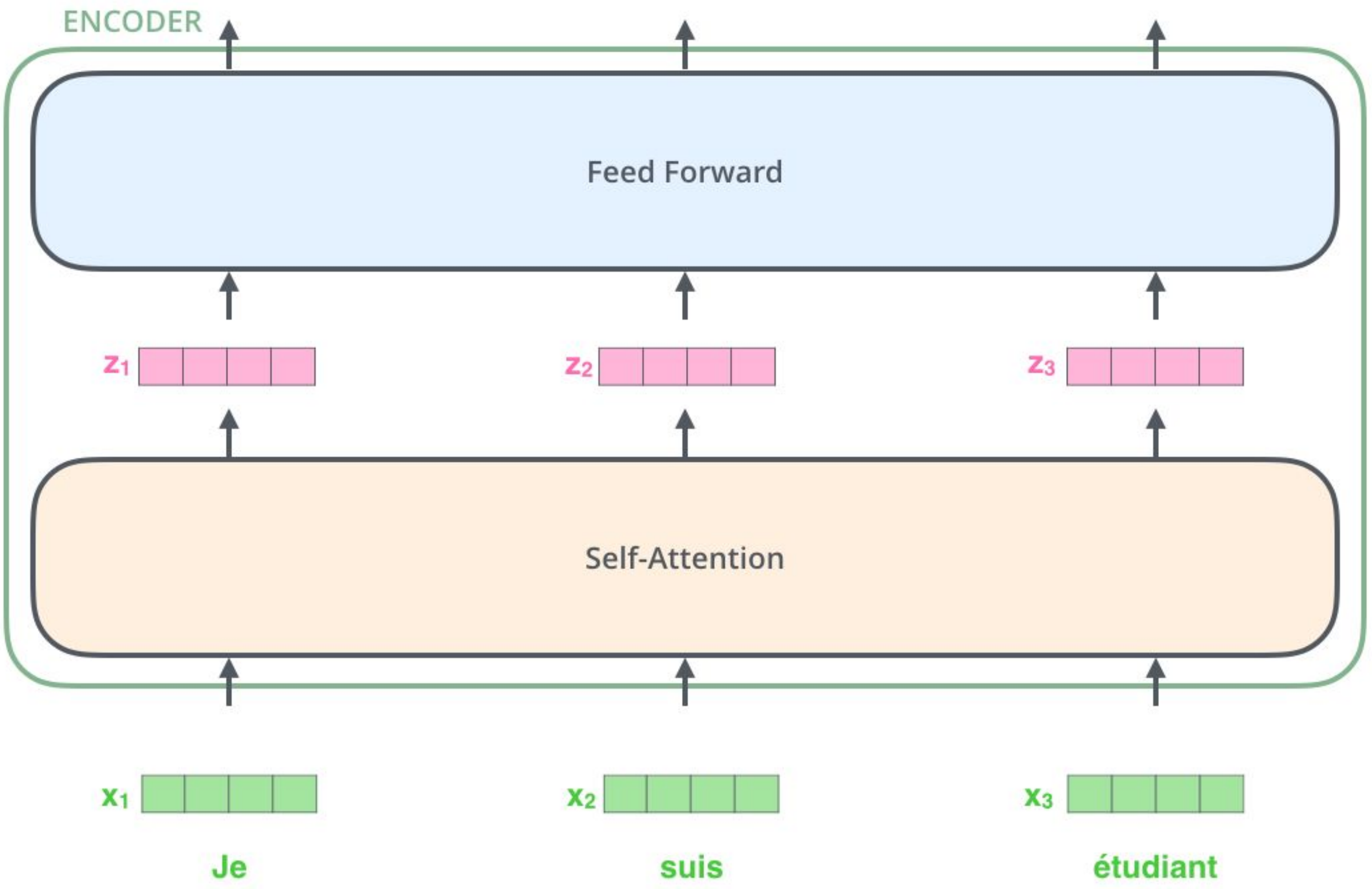

OUTPUT
I am a student
ENCODERS
DECODERS
INPUT
Je suis étudiant

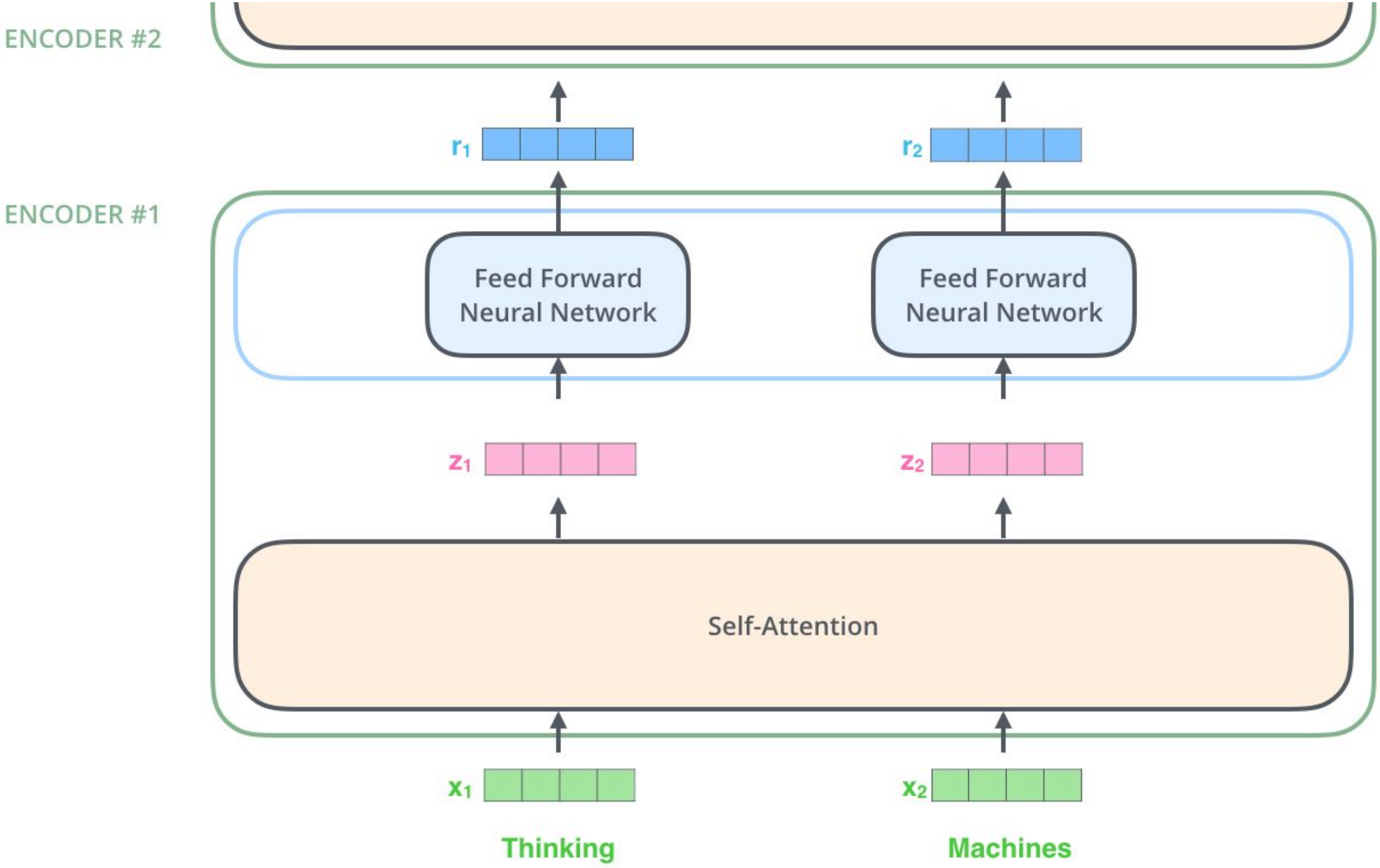

OUTPUT
I am a student
ENCODER
ENCODER
ENCODER
ENCODER
ENCODER
ENCODER
DECODER
DECODER
DECODER
DECODER
DECODER
DECODER
INPUT
Je suis étudiant


ENCODER
Feed Forward Neural Network
Self-Attention


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

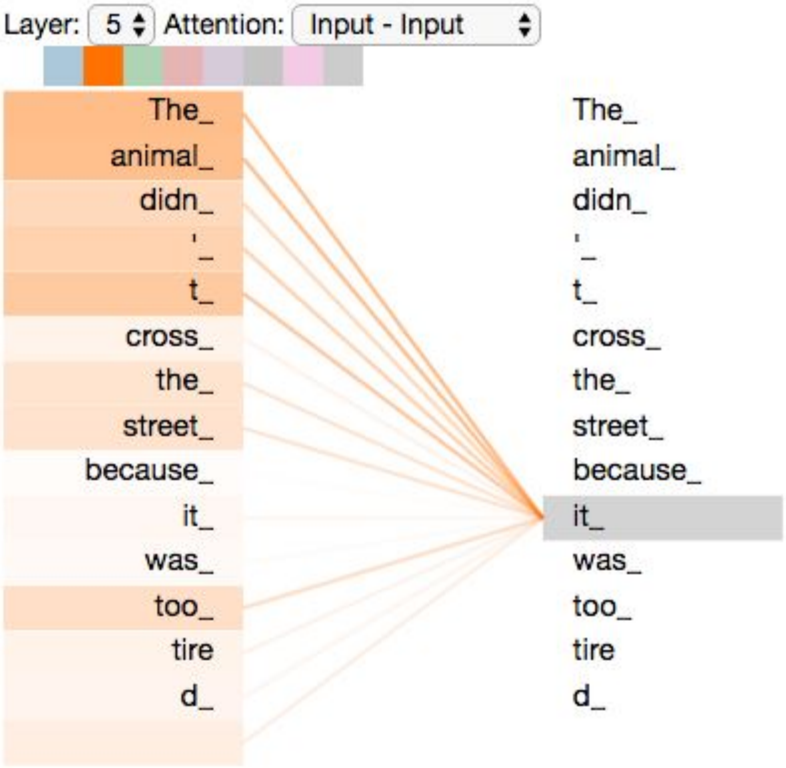

x_1
Je
x_2
suis
x_3
étudiant

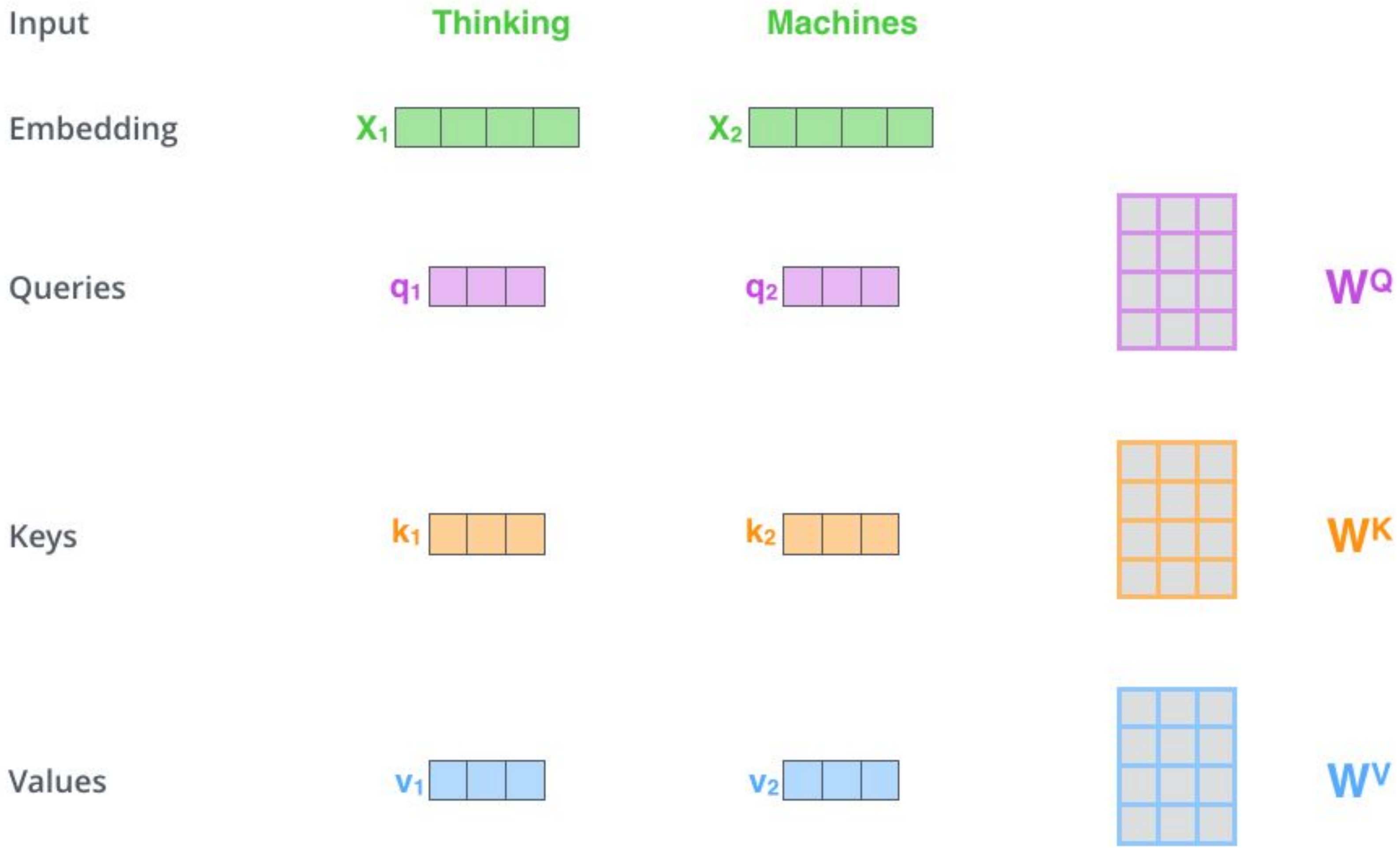

ENCODER
Feed Forward
z1
z2
z3
Self-Attention
x1
x2
x3
Je
suis
étudiant

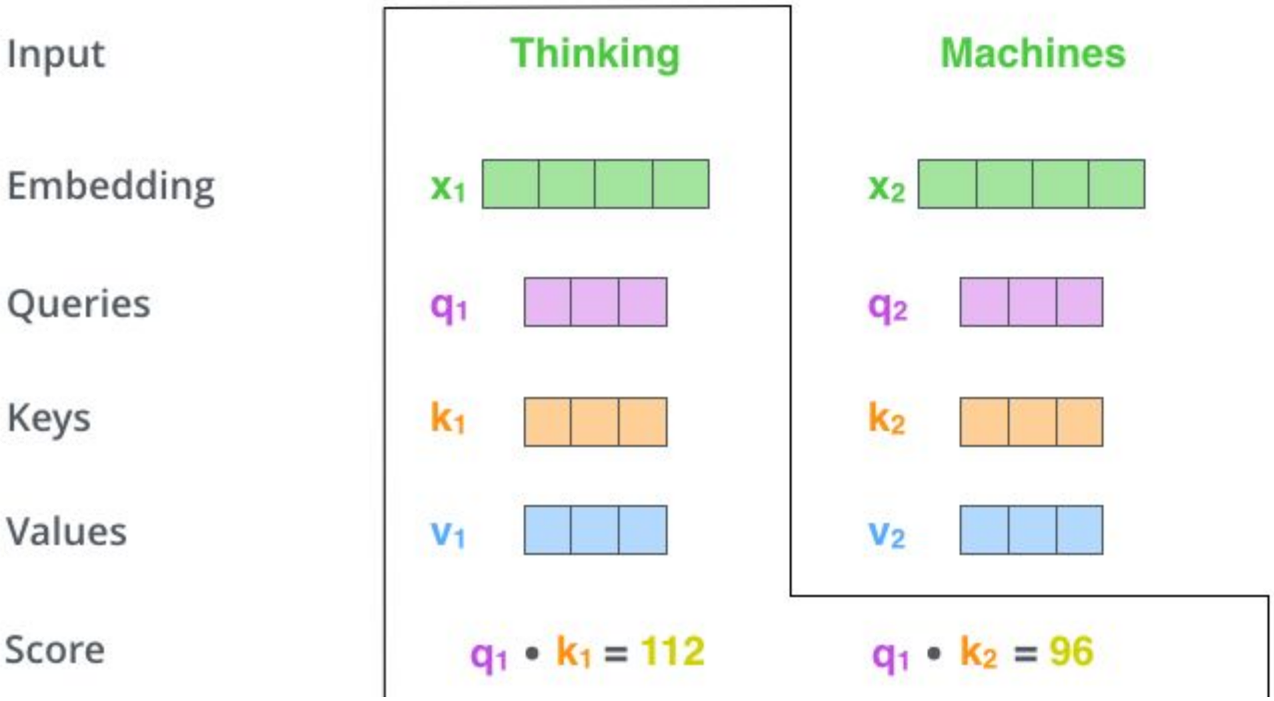

ENCODER #2
r1
r2
ENCODER #1
Feed Forward Neural Network
Feed Forward Neural Network
z1
z2
Self-Attention
x1
x2
Thinking
Machines

Self Attention

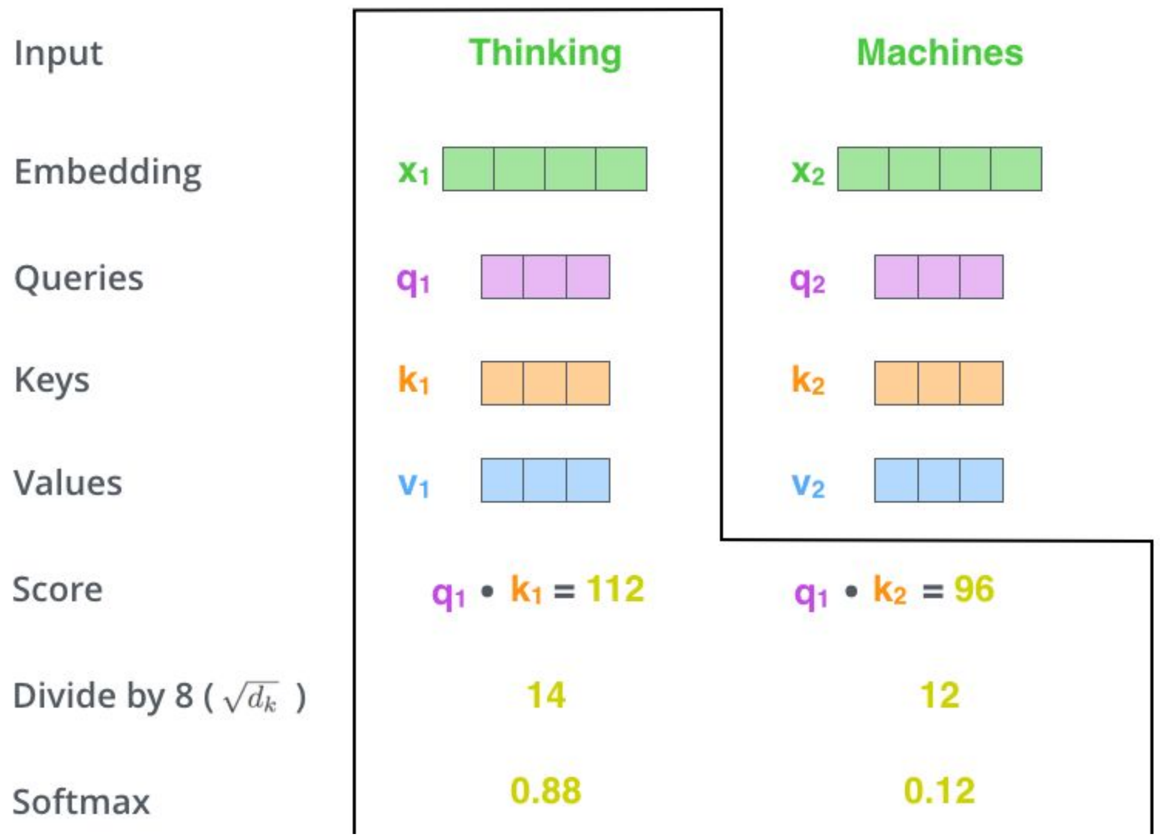
"The animal didn't cross the street because it was too tired"



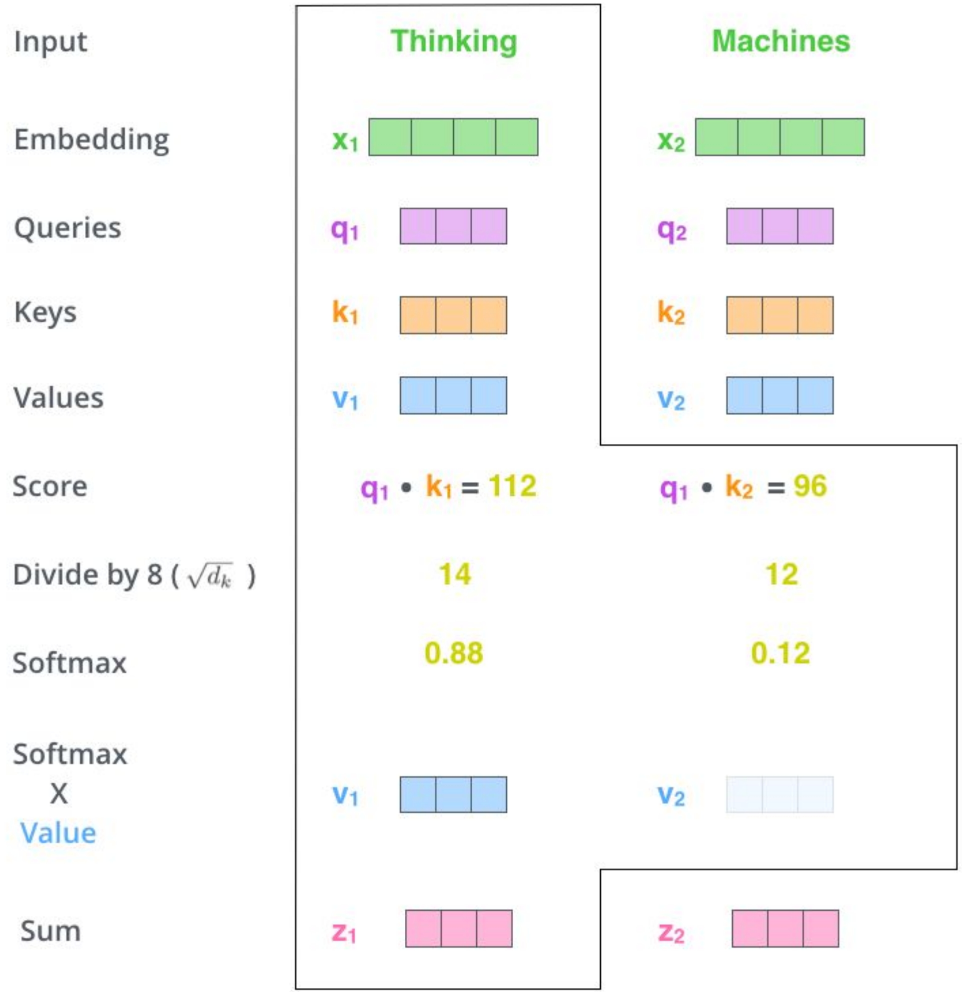

Input
Thinking
Machines
Embedding
X_1
X_2
Queries
q_1
q_2
W^Q
Keys
k_1
k_2
W^K
Values
v_1
v_2
W^V

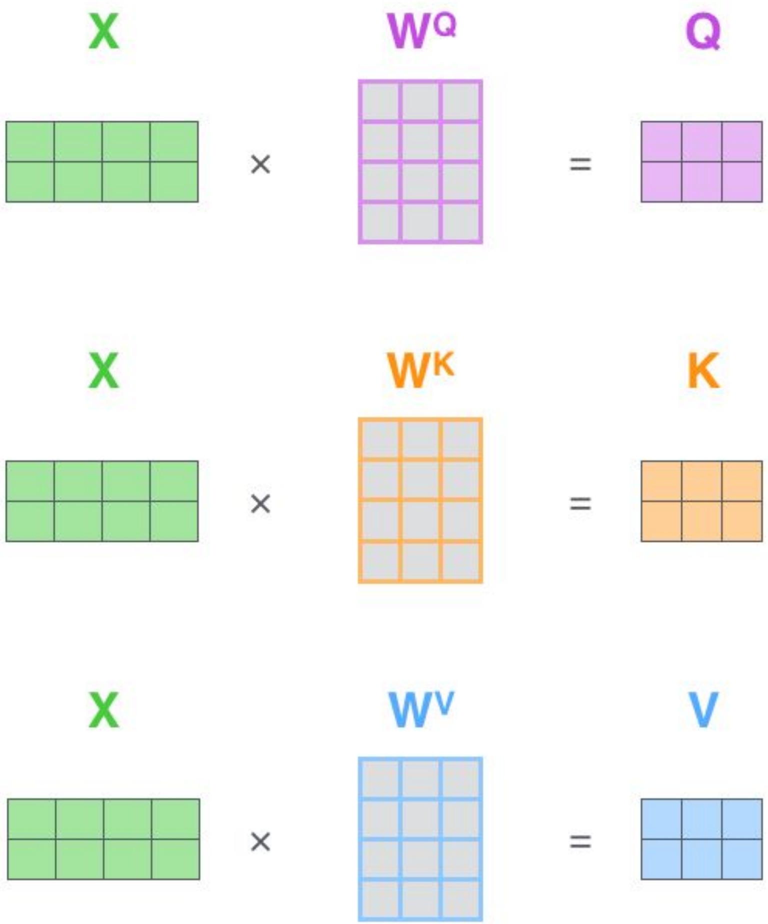


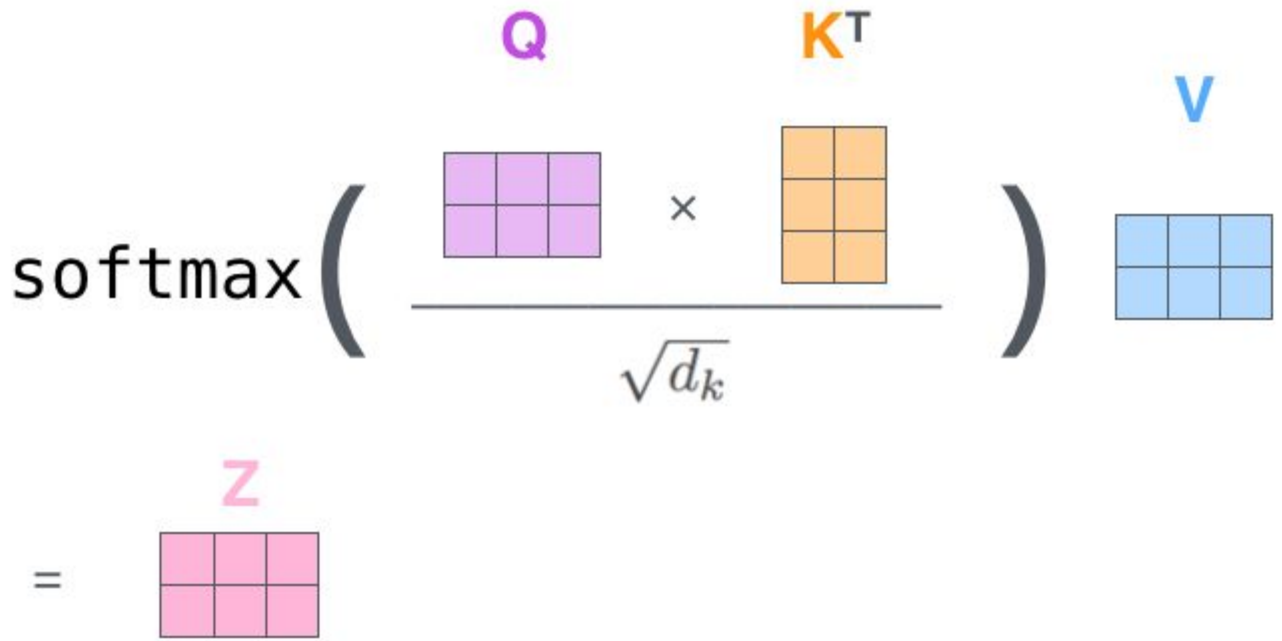
Input
Thinking
Machines
Embedding
x1
x2
Queries
q1
q2
Keys
k1
k2
Values
v1
v2
Score
q1 • k1 = 112
q1 • k2 = 96



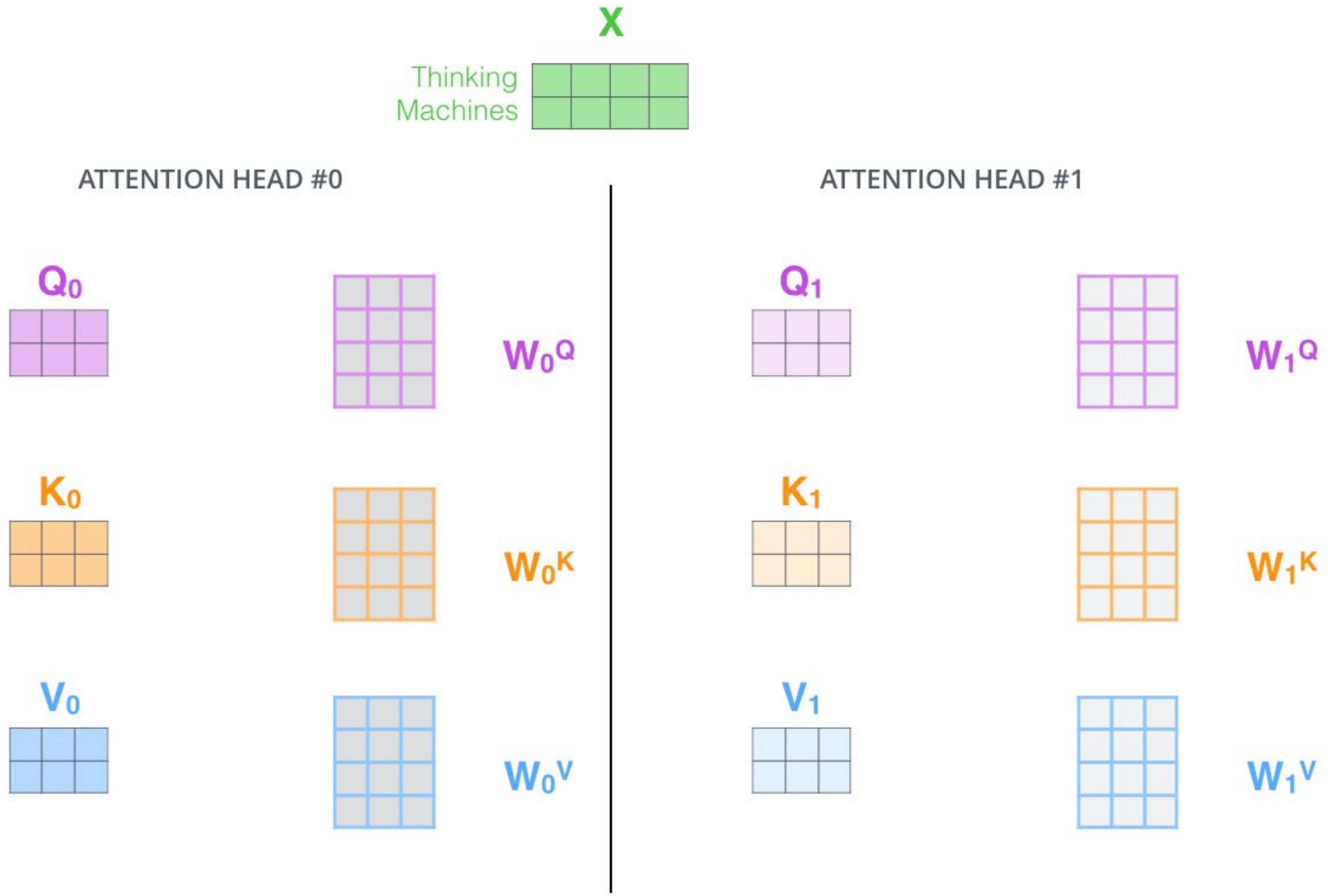
Input
Thinking
Machines
Embedding
x_1
x_2
Queries
q_1
q_2
Keys
k_1
k_2
Values
v_1
v_2
Score
$q_1 \bullet k_1 = 112$
$q_1 \bullet k_2 = 96$
Divide by 8 ($\sqrt{d_k}$)
14
12
Softmax
0.88
0.12

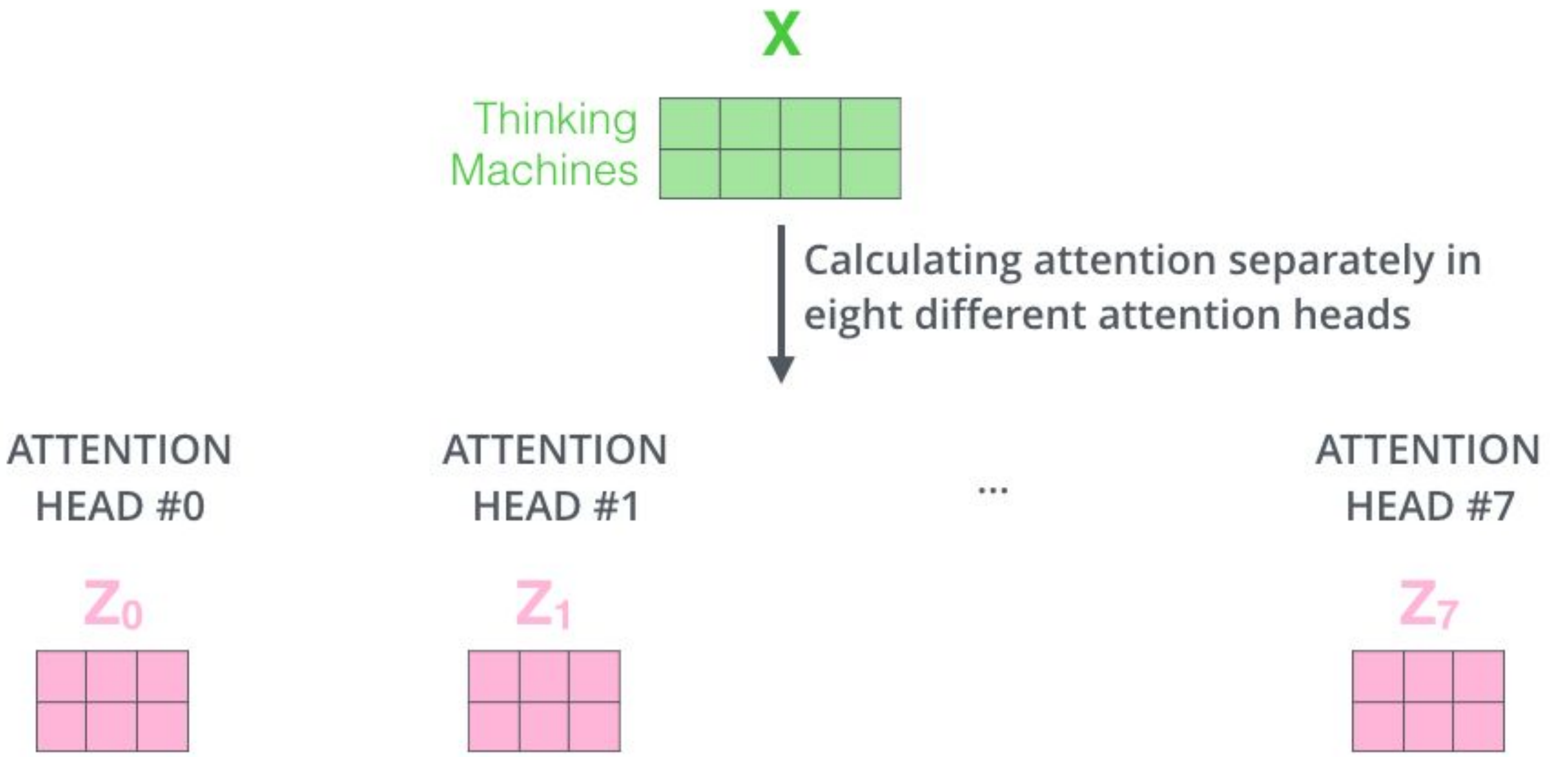

Input
Thinking
Machines
Embedding
x1
x2
Queries
q1
q2
Keys
k1
k2
Values
v1
v2
Score
q1 • k1 = 112
q1 • k2 = 96
Divide by 8 ($\sqrt{d_k}$)
14
12
Softmax
0.88
0.12
Softmax
X
Value
v1
v2
Sum
z1
z2

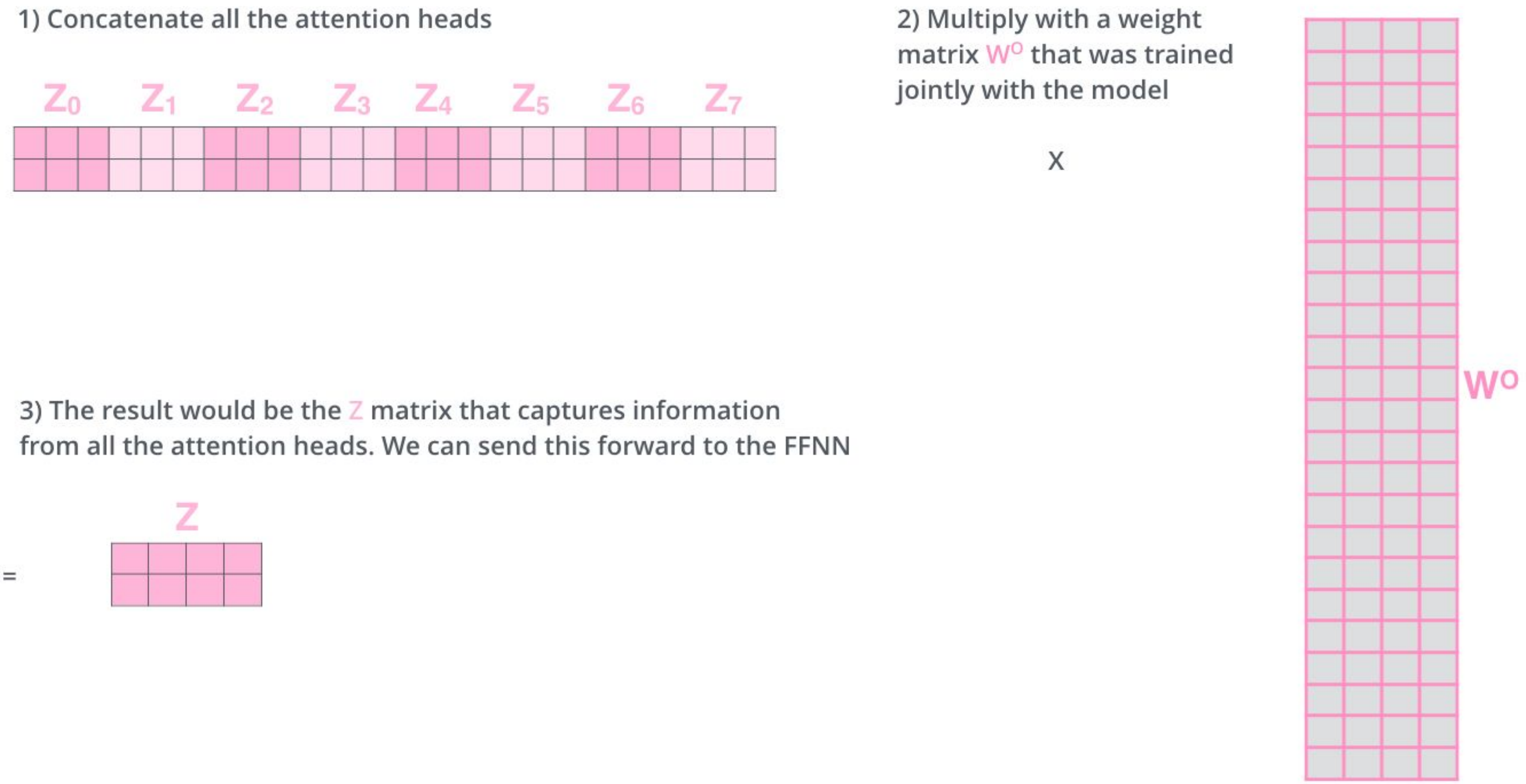

X
W^Q
Q
X
W^K
K
X
W^V
V


softmax
Q
×
K
T
$\sqrt{d_k}$
V
=
Z

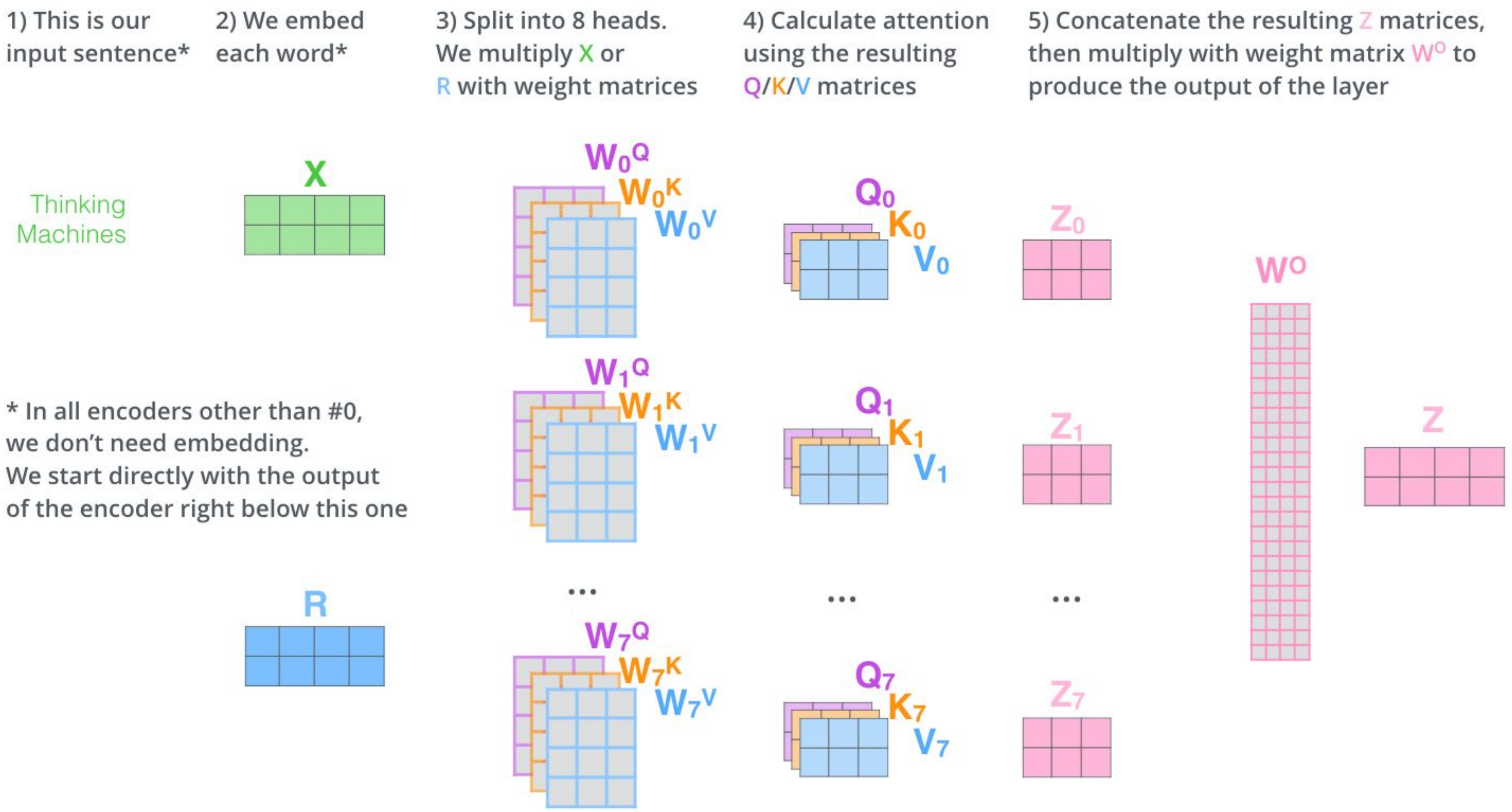
Multi-headed Attention


X
Thinking
Machines
ATTENTION HEAD #0
ATTENTION HEAD #1
Q_0
W_0^Q
K_0
W_0^K
V_0
W_0^V
Q_1
W_1^Q
K_1
W_1^K
V_1
W_1^V

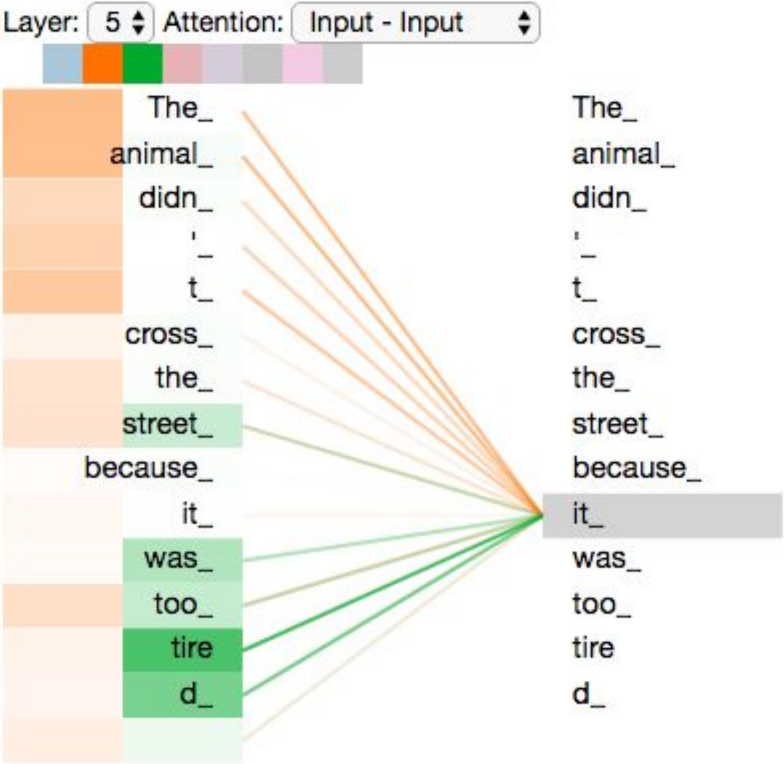

X
Thinking
Machines
Calculating attention separately in eight different attention heads
ATTENTION HEAD #0
ATTENTION HEAD #1
...
ATTENTION HEAD #7
Z0
Z1
Z7

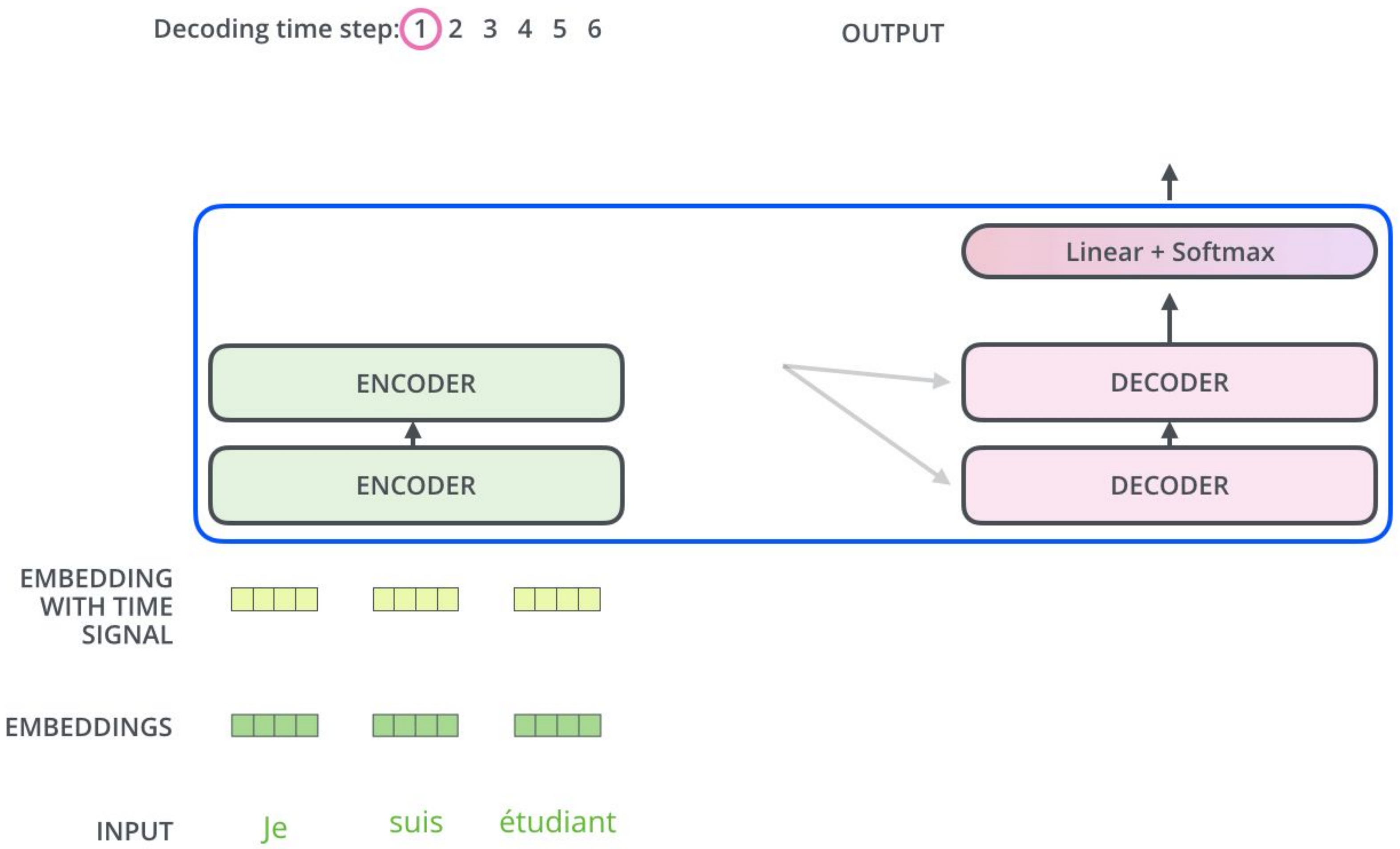

1) Concatenate all the attention heads
Z0 Z1 Z2 Z3 Z4 Z5 Z6 Z7
2) Multiply with a weight matrix W^O that was trained jointly with the model
X
W^O
3) The result would be the Z matrix that captures information from all the attention heads. We can send this forward to the FFNN
=
Z

Multi Headed Self Attention

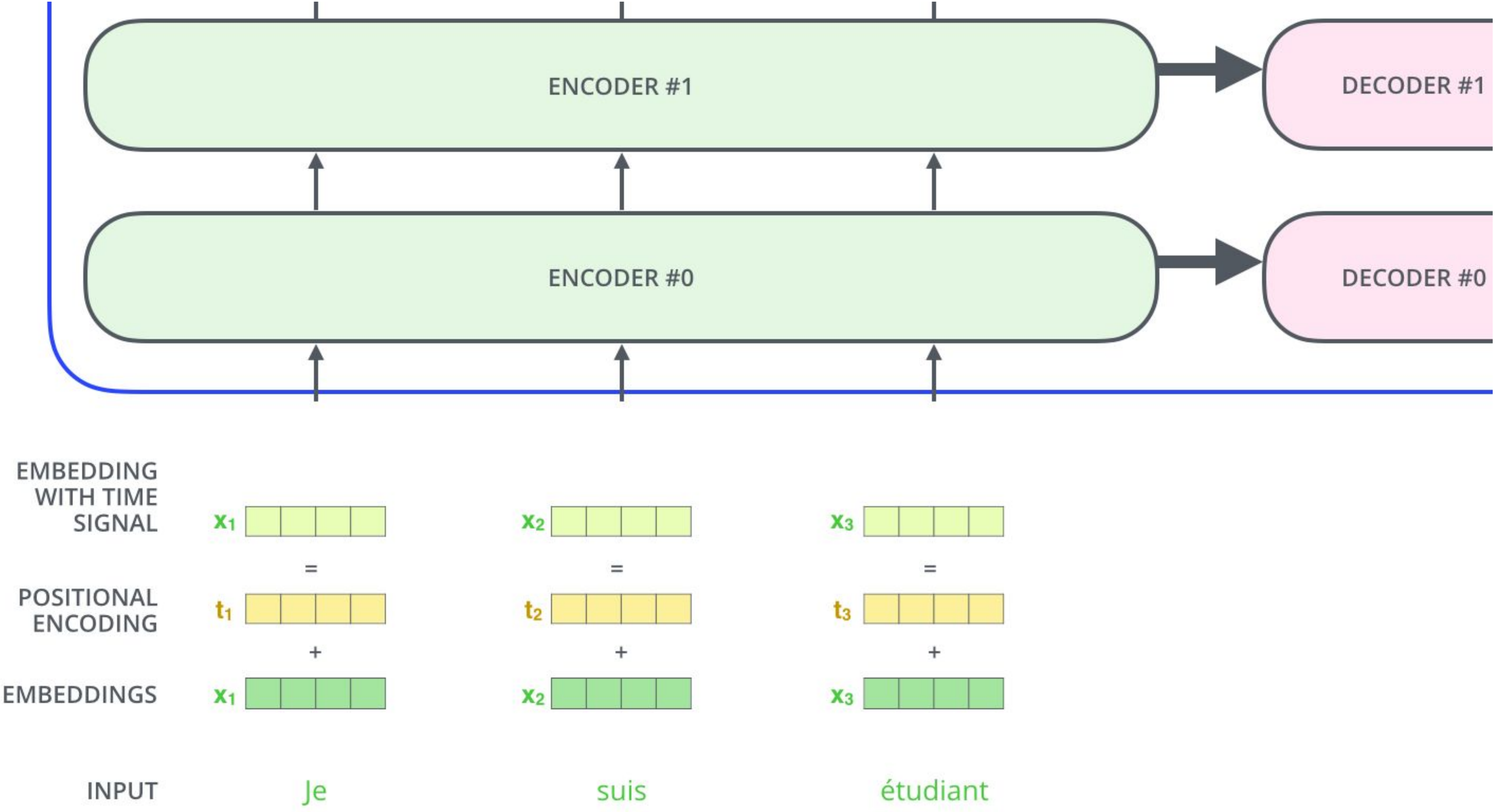


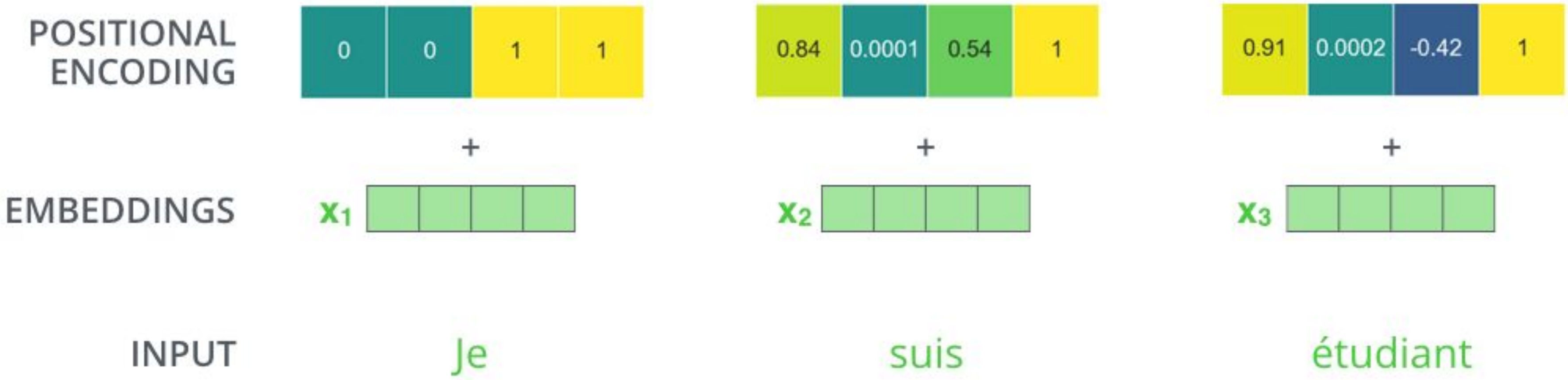
1) This is our input sentence*
2) We embed each word*
3) Split into 8 heads. We multiply X or R with weight matrices
4) Calculate attention using the resulting Q/K/V matrices
5) Concatenate the resulting Z matrices, then multiply with weight matrix W^O to produce the output of the layer
Thinking Machines
X
* In all encoders other than #0, we don't need embedding. We start directly with the output of the encoder right below this one
R
W0Q W0K W0V
W1Q W1K W1V
...
W7Q W7K W7V
Q0 K0 V0
Q1 K1 V1
...
Q7 K7 V7
Z0
Z1
...
Z7
WO
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_

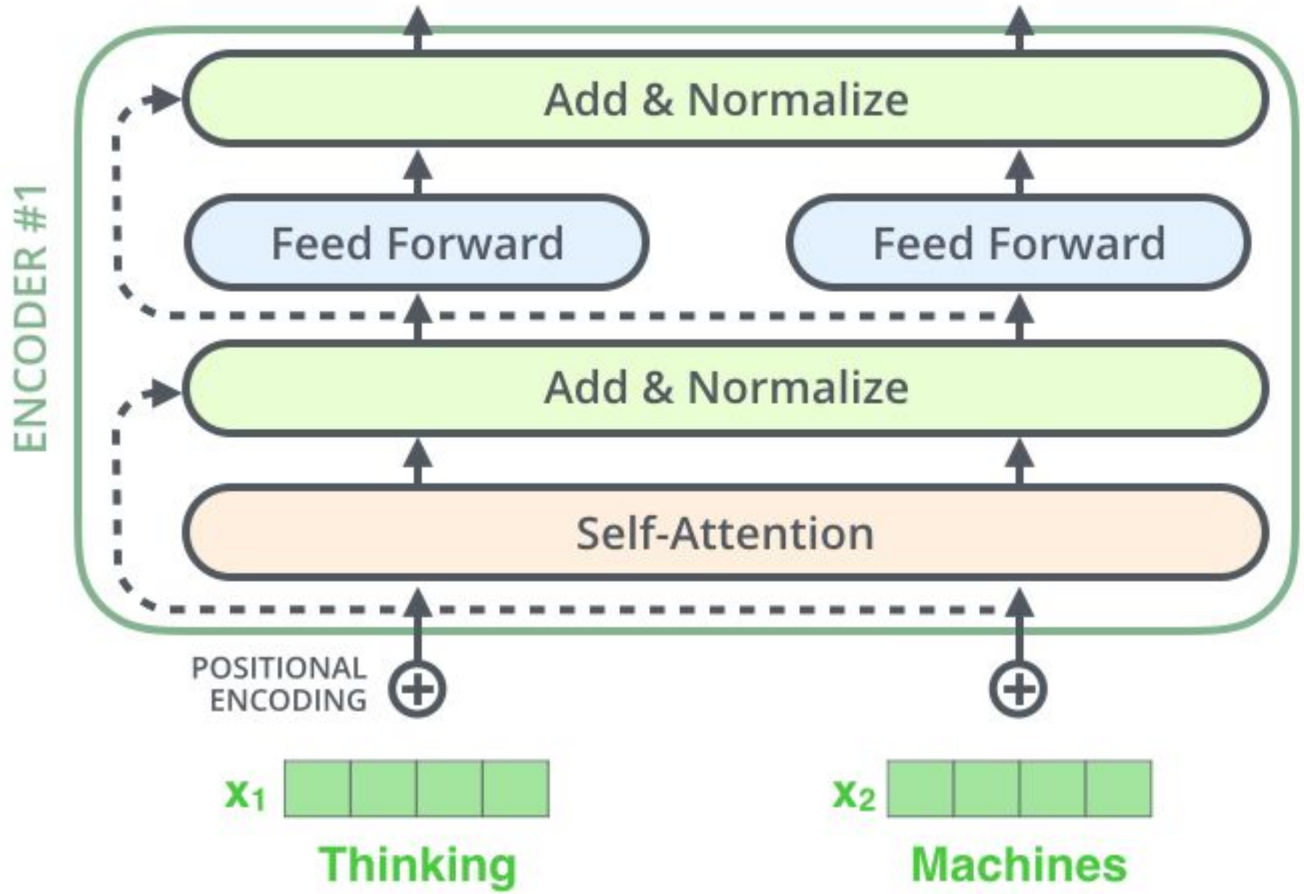

Decoding time step: 1 2 3 4 5 6
OUTPUT
Linear + Softmax
ENCODER
ENCODER
DECODER
DECODER
EMBEDDING WITH TIME SIGNAL
EMBEDDINGS
INPUT
Je
suis
étudiant

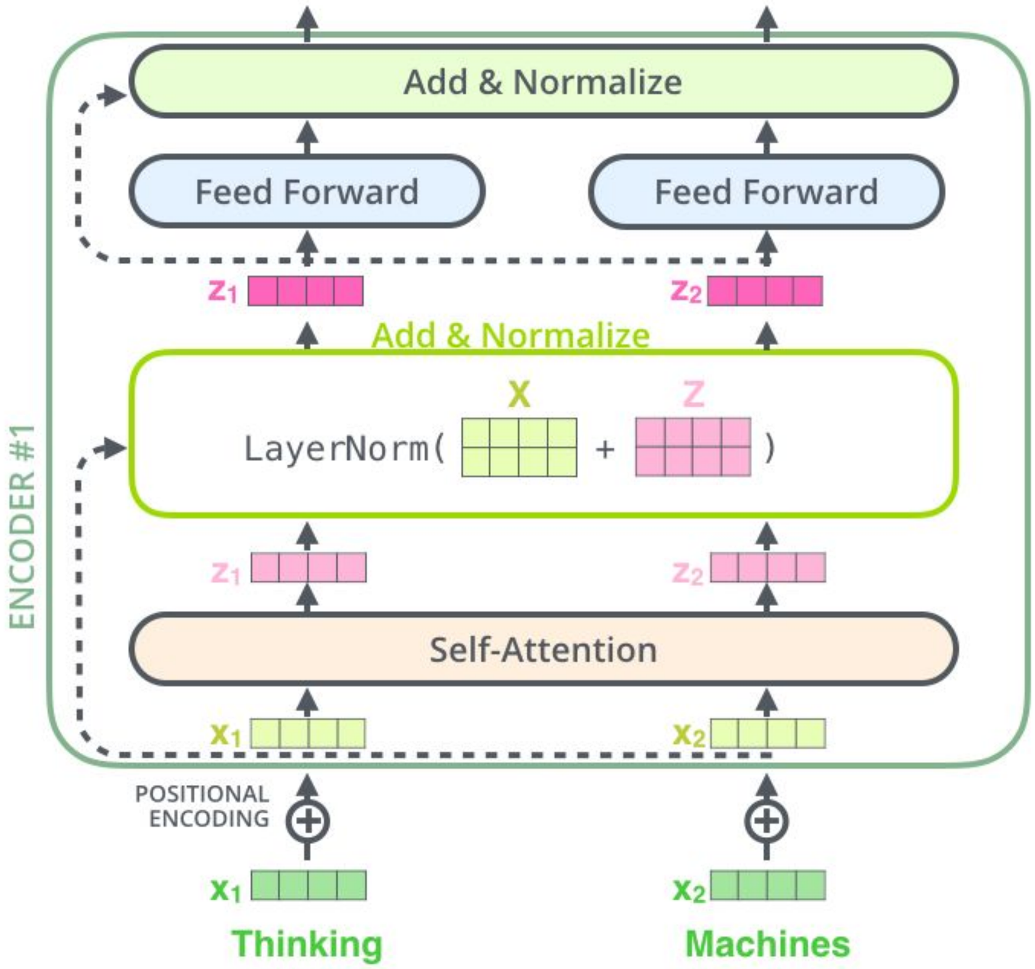
Positional Encoding


ENCODER #1
DECODER #1
ENCODER #0
DECODER #0
EMBEDDING WITH TIME SIGNAL
x_1
x_2
x_3
=
POSITIONAL ENCODING
t_1
t_2
t_3
+
EMBEDDINGS
x_1
x_2
x_3
INPUT
Je
suis
étudiant

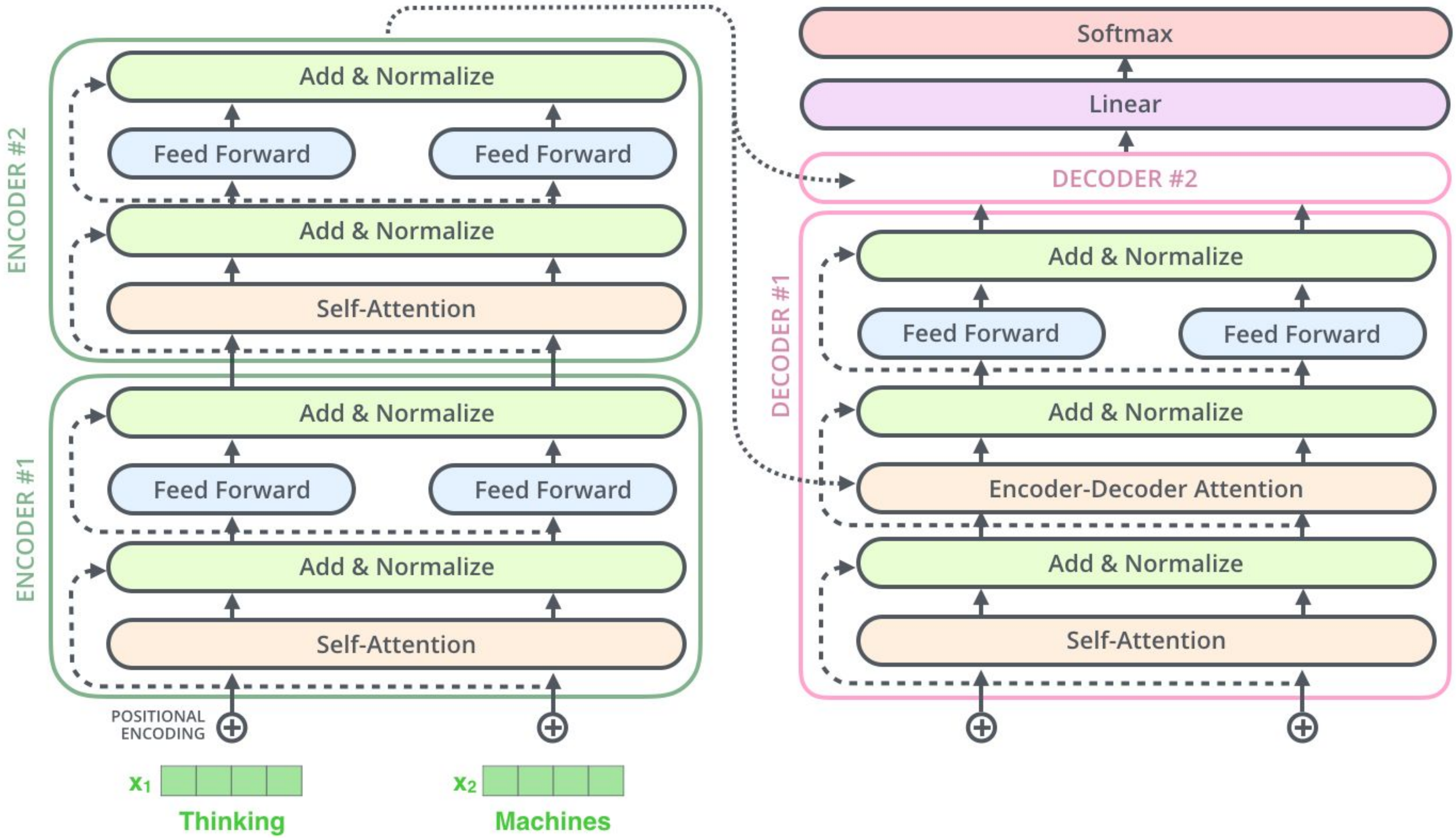

POSITIONAL ENCODING
0
0
1
1
0.84
0.0001
0.54
1
0.91
0.0002
-0.42
1
+
+
+
EMBEDDINGS
x_1
x_2
x_3
INPUT
Je
suis
étudiant

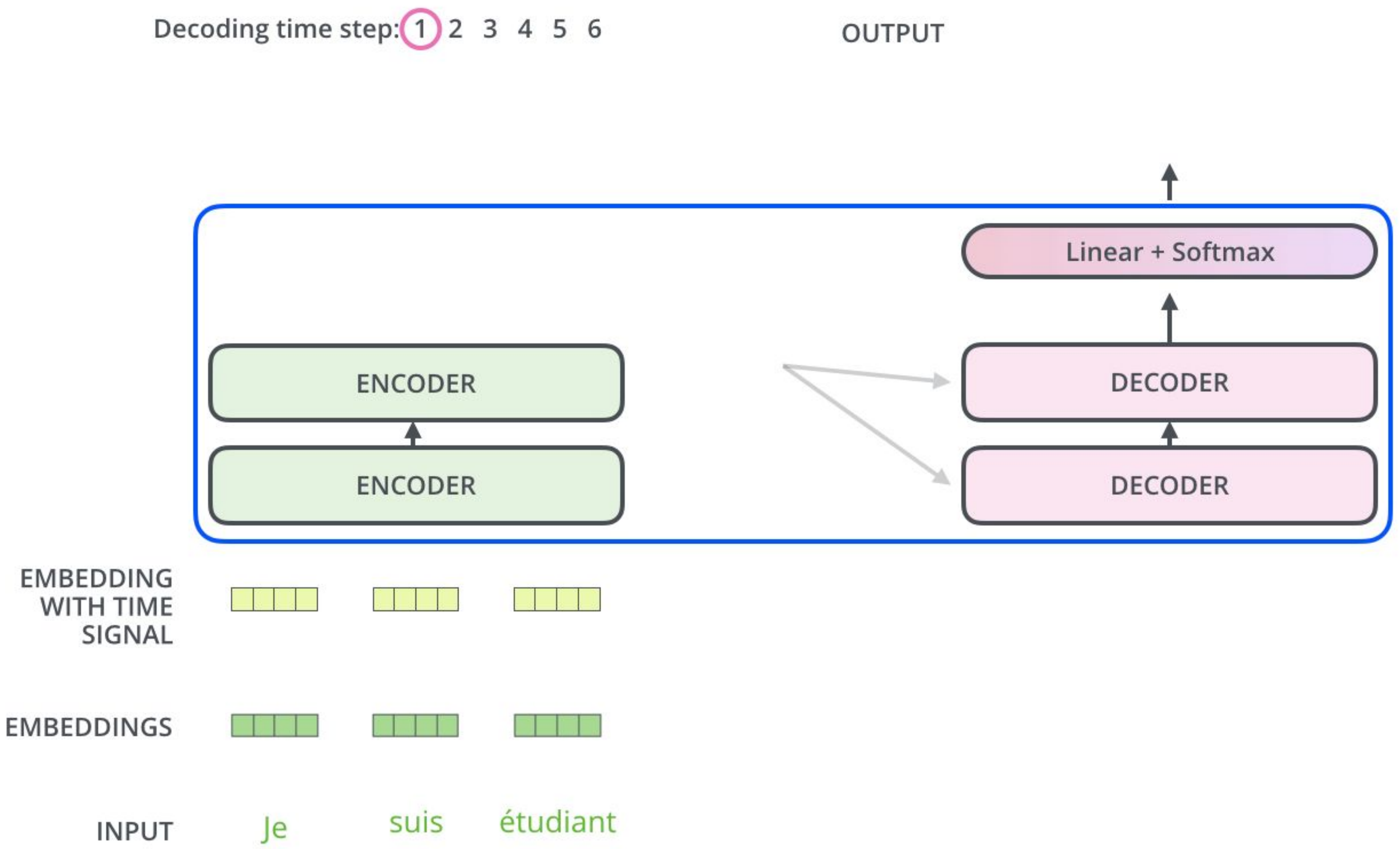
Residuals

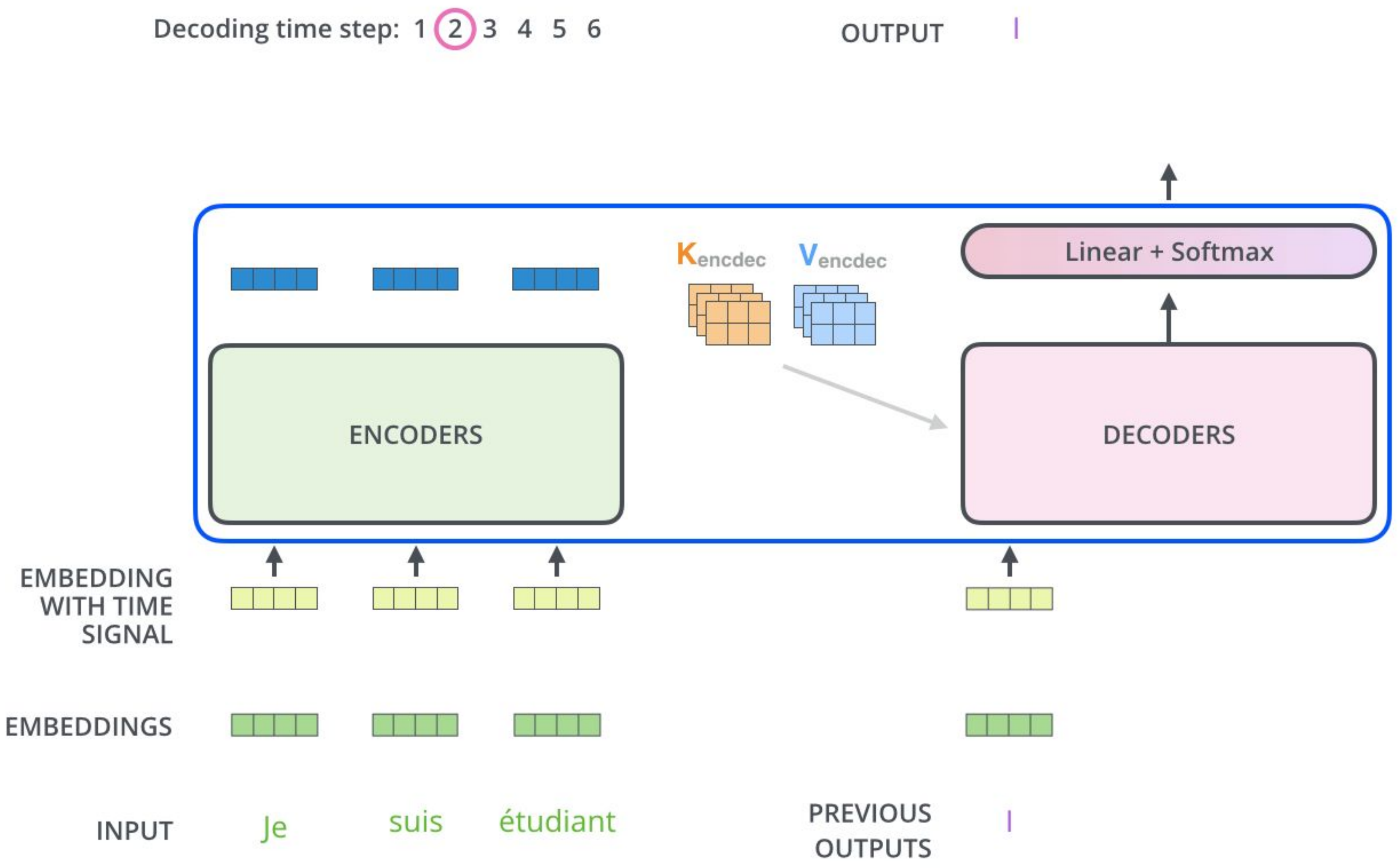

ENCODER #1
Add & Normalize
Feed Forward
Feed Forward
Add & Normalize
Self-Attention
POSITIONAL
ENCODING
x1
x2
Thinking
Machines



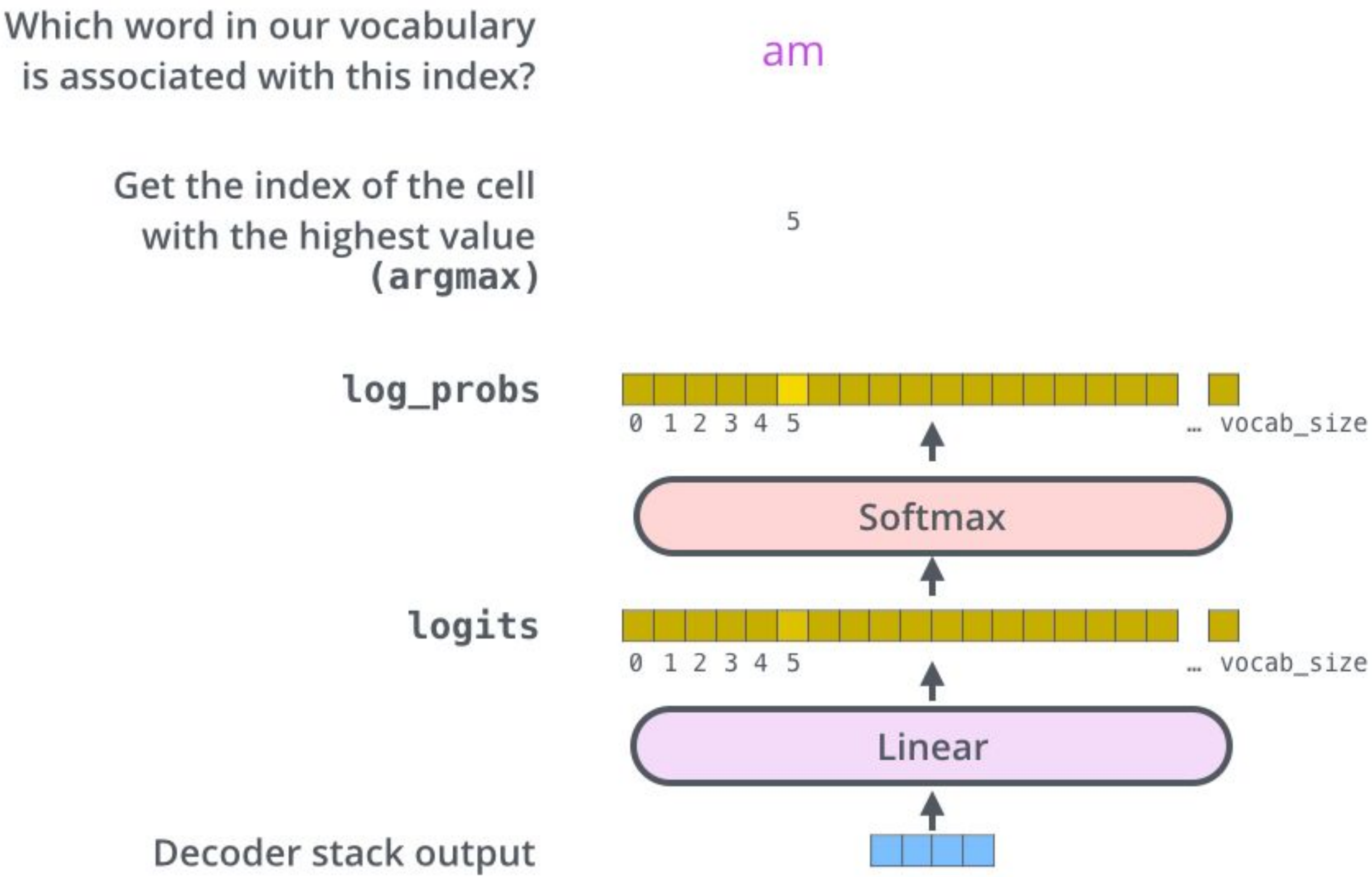
Add & Normalize
Feed Forward
Feed Forward
z1
z2
Add & Normalize
X
Z
LayerNorm(+)
z1
z2
Self-Attention
x1
x2
ENCODER #1
POSITIONAL
ENCODING
x1
x2
Thinking
Machines


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
x_1
Thinking
x_2
Machines


Decoding time step: 1 2 3 4 5 6
OUTPUT
Linear + Softmax
ENCODER
ENCODER
DECODER
DECODER
EMBEDDING WITH TIME SIGNAL
EMBEDDINGS
INPUT
Je
suis
étudiant



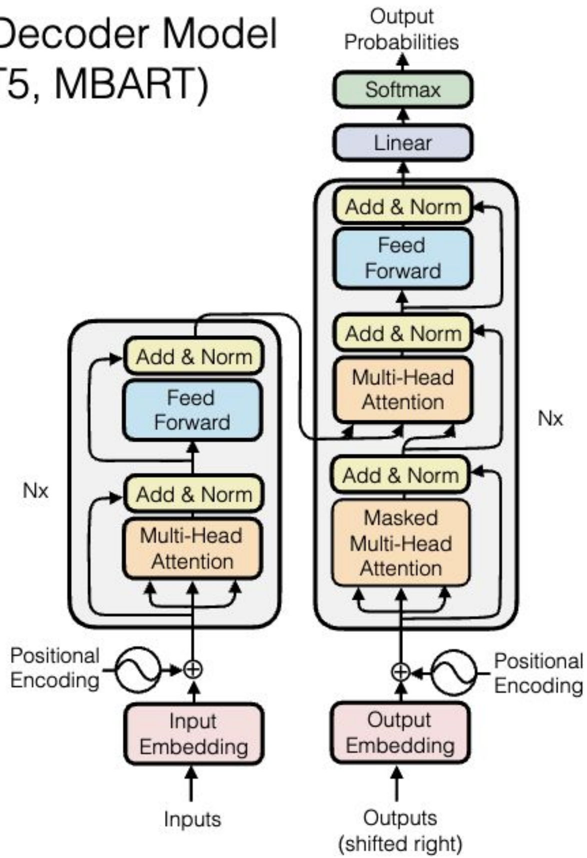
Decoding time step: 1 2 3 4 5 6
OUTPUT
I
Kencdec
Vencdec
Linear + Softmax
ENCODERS
DECODERS
EMBEDDING WITH TIME SIGNAL
EMBEDDINGS
INPUT
Je
suis
étudiant
PREVIOUS OUTPUTS
I


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
0 1 2 3 4 5
… vocab_size
Softmax
logits
0 1 2 3 4 5
… vocab_size
Linear
Decoder stack output

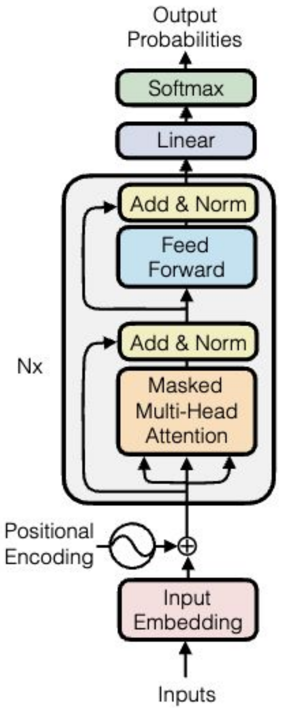
LLMs

Two Types of Transformers

Encoder-Decoder Model (e.g. T5, MBART)

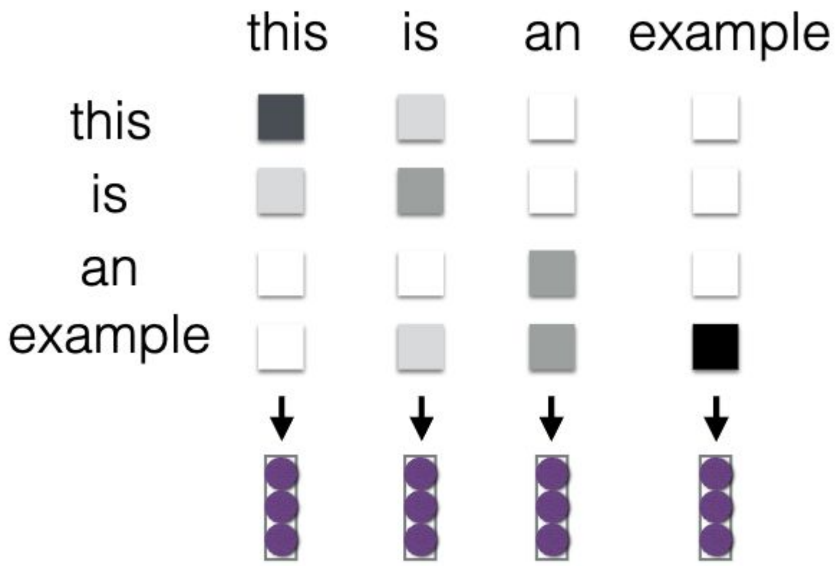


Decoder Only Model (e.g. GPT, LLaMa)



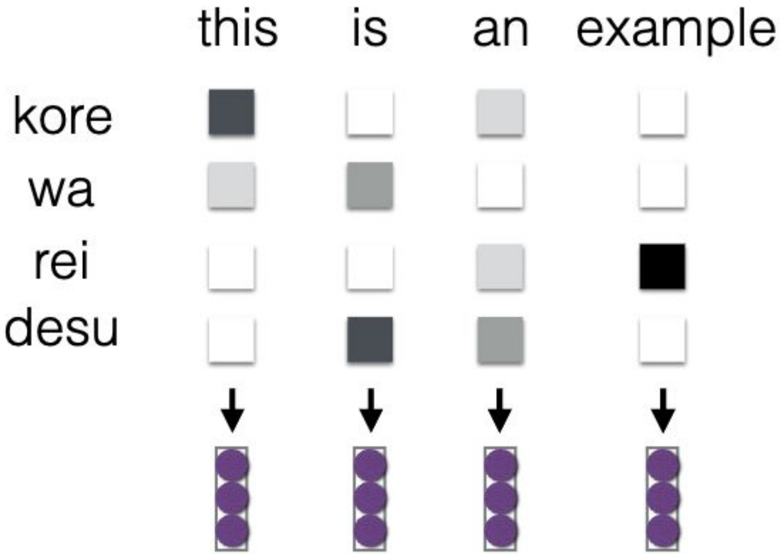
Self Attention
(Cheng et al. 2016, Vaswani et al. 2017)

- Each element in the sequence attends to elements of that sequence



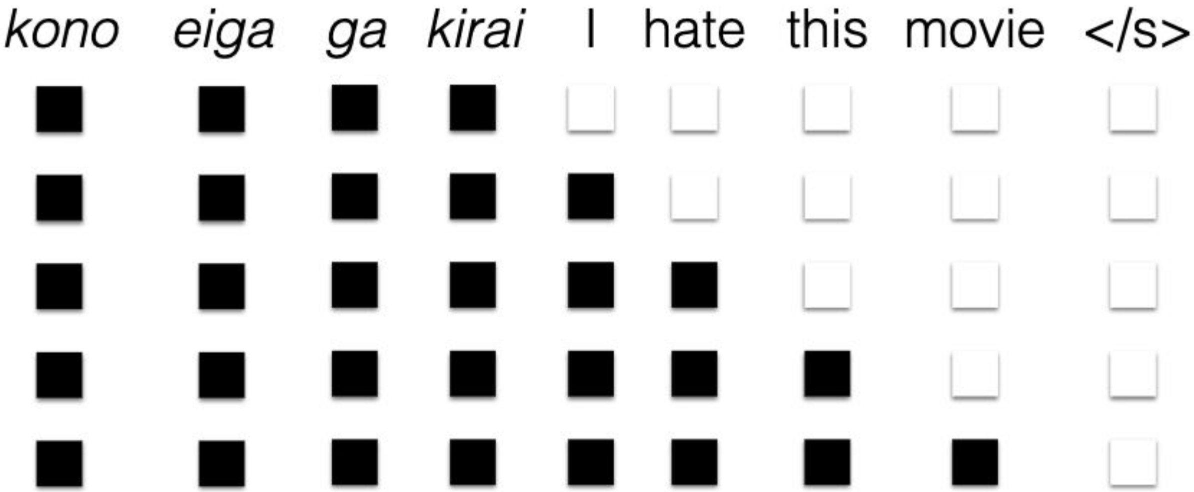
Cross Attention
(Bahdanau et al. 2015)

- Each element in a sequence attends to elements of another sequence

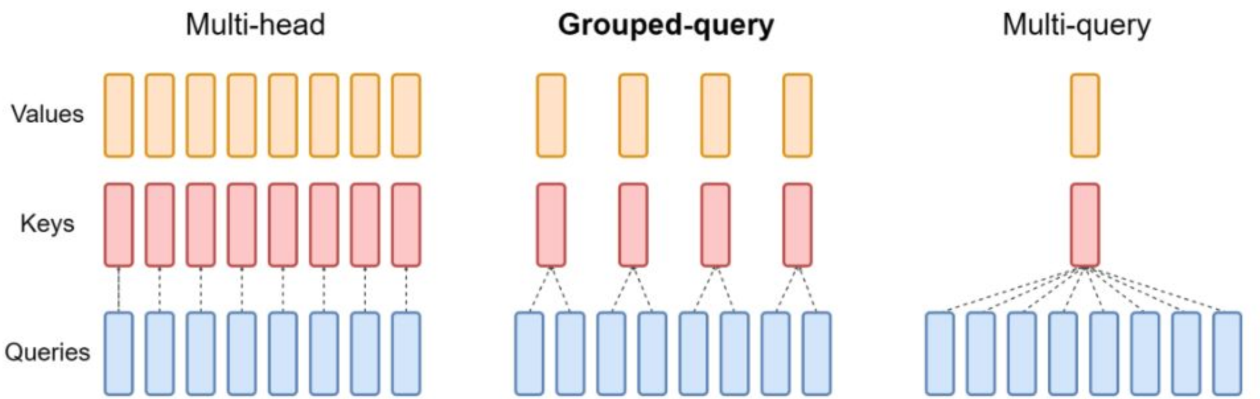


Masking for Language Model Training

- Mask the attention from future timesteps
  - Prevents the model from cheating when predicting the next token



Grouped-query attention



- Shares key and value heads for each *group* of query heads
- Saves on memory, which leads to faster inference

Original Transformer vs. LLama

	Vaswani et al.	LLama	Llama 2
Norm Position	Post	**Pre**	**Pre**
Norm Type	LayerNorm	**RMSNorm**	**RMSNorm**
Non-linearity	ReLU	**SwiGLU**	**SwiGLU**
Positional Encoding	Sinusoidal	**RoPE**	**RoPE**
Attention	Multi-head	Multi-head	**Grouped-query**