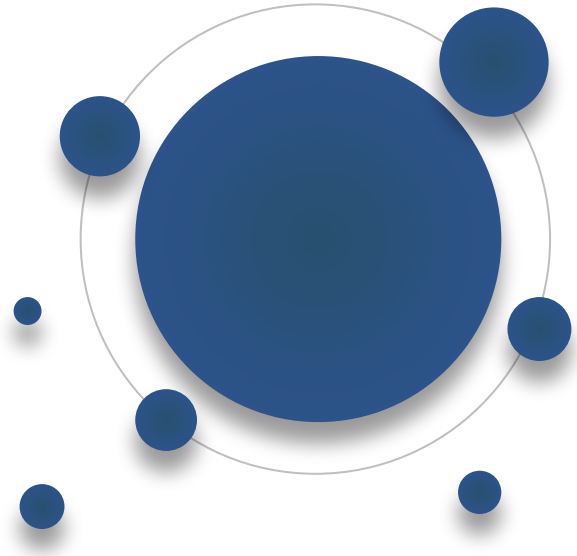


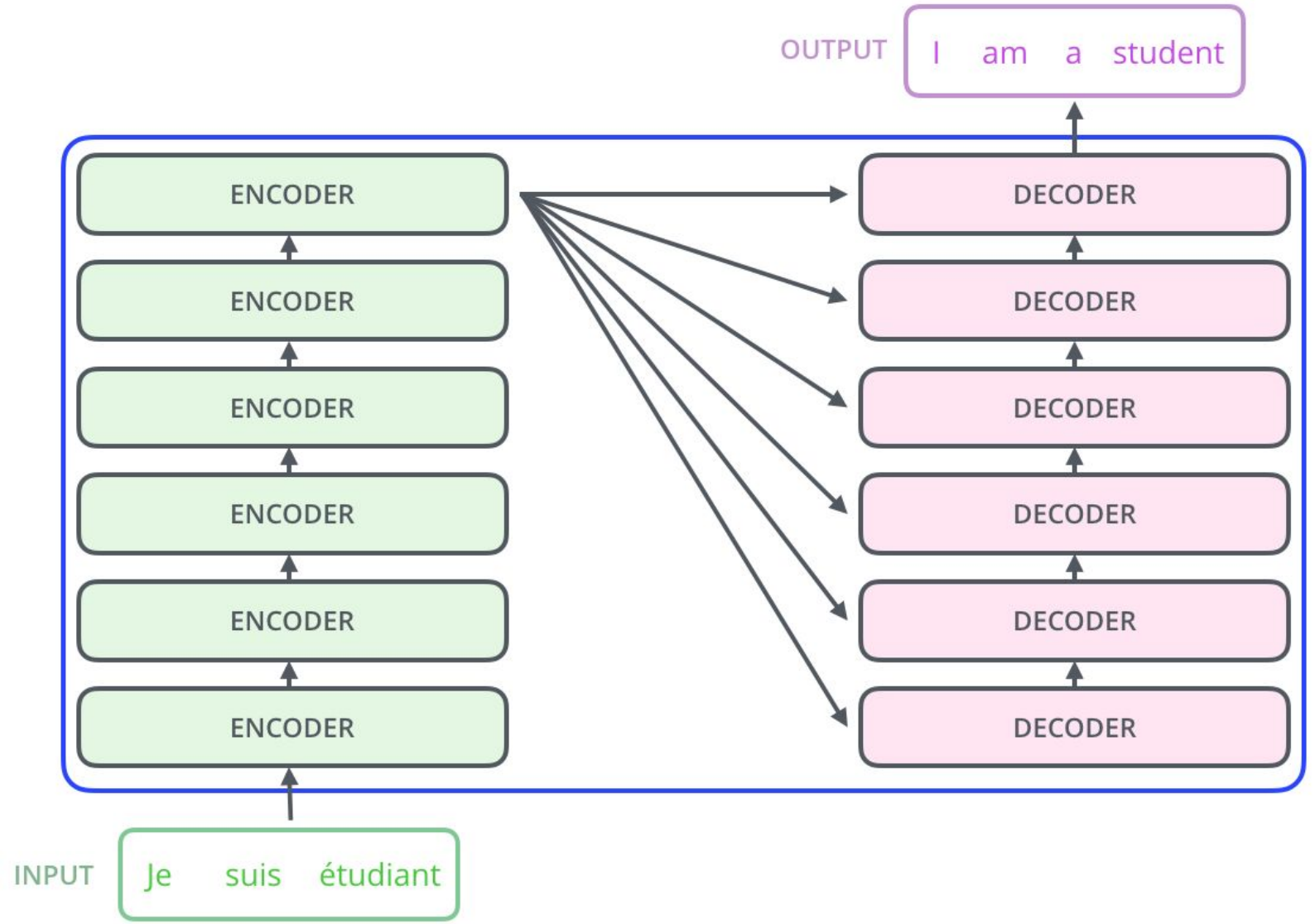
Attention and Transformers

November 3, 2021

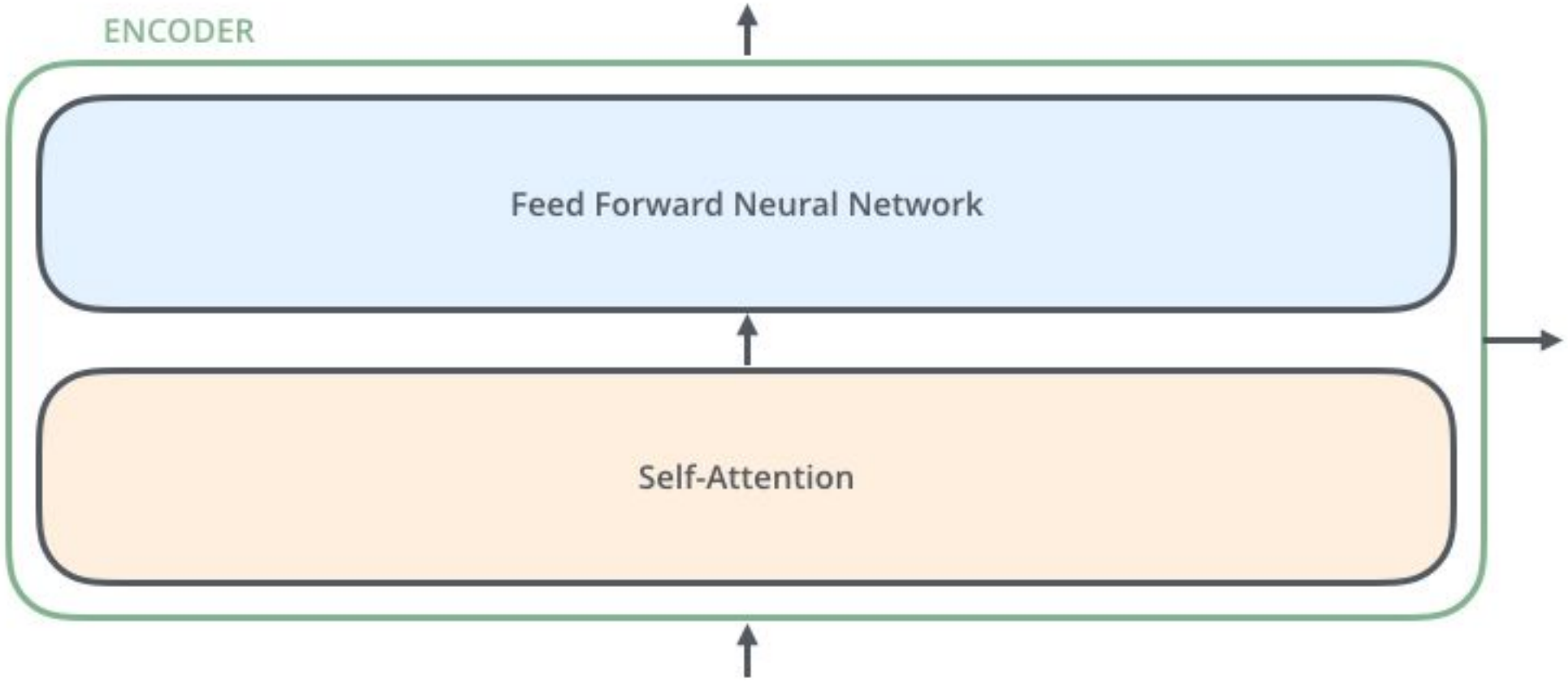
Many ideas for Deep Learning in NLP

1. ULM-FiT, pre-training, transfer learning in NLP
2. Recurrent models require linear sequential computation, hard to parallelize. ELMo, bidirectional LSTM.
3. In order to reduce such sequential computation, several models based on CNN are introduced, such as ConvS2S and ByteNet. Dependency for ConvS2S needs linear depth, and ByteNet logarithmic.
4. The transformer is the first transduction model relying entirely on self-attention to compute the representations of its input and output without using RNN or CNN.

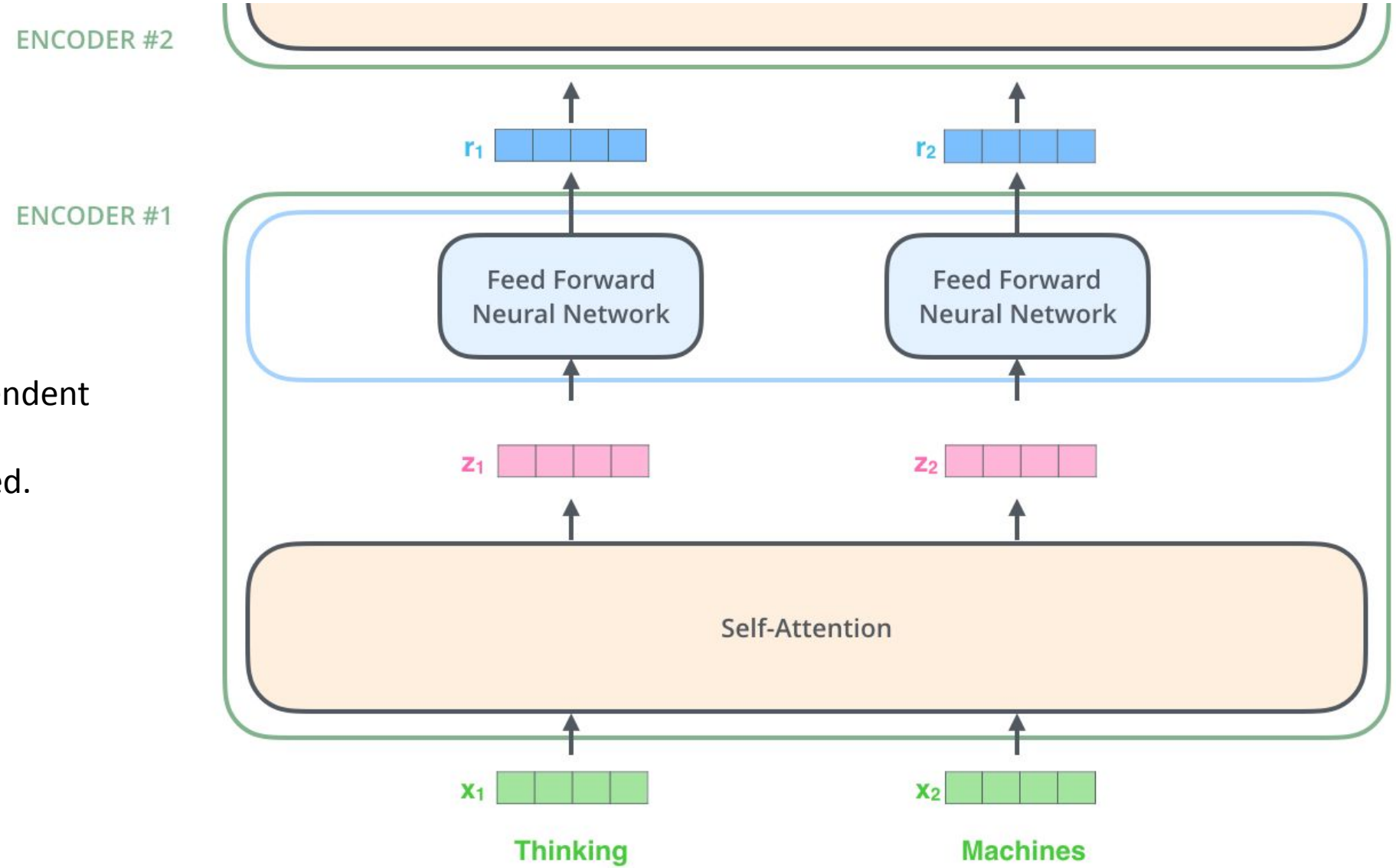
Transformer



An Encoder Block: same structure, different parameters



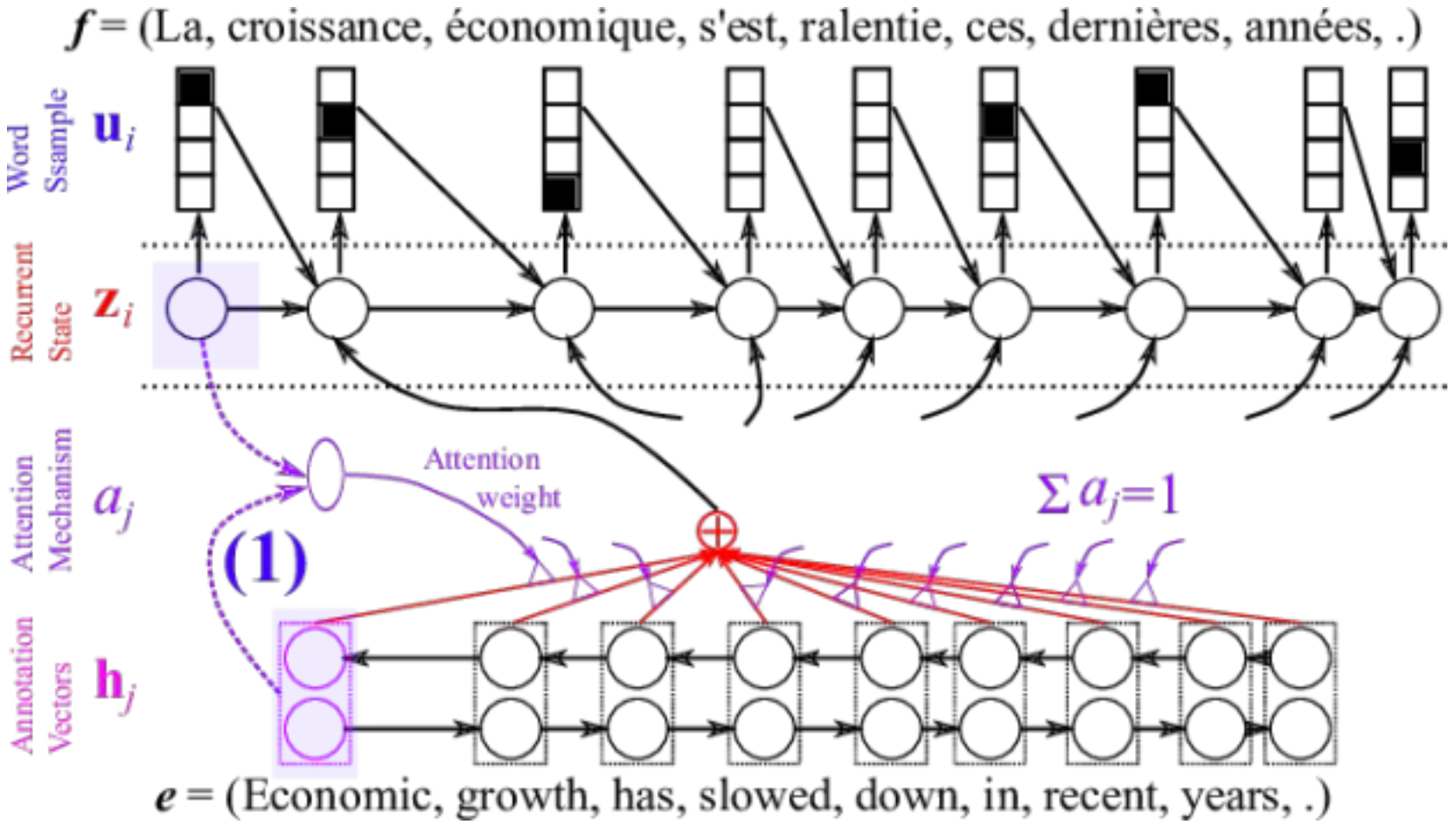
Encoder



Note: The ffnn is independent for each word.
Hence can be parallelized.

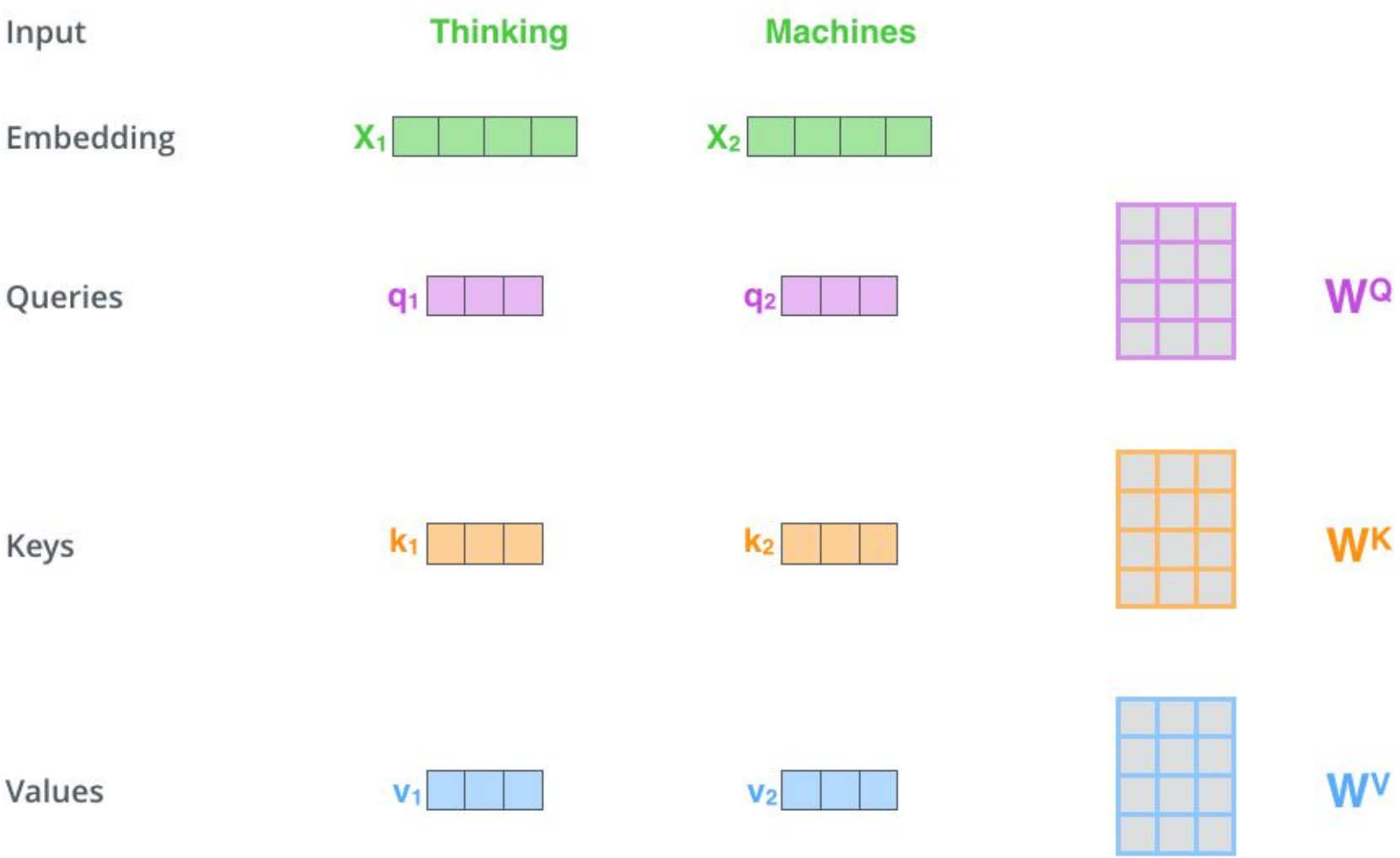
Attention

Given a set of vector values and a vector query, the **attention** is a technique to compute a weighted sum of the values, dependent on the queries.



Self Attention

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

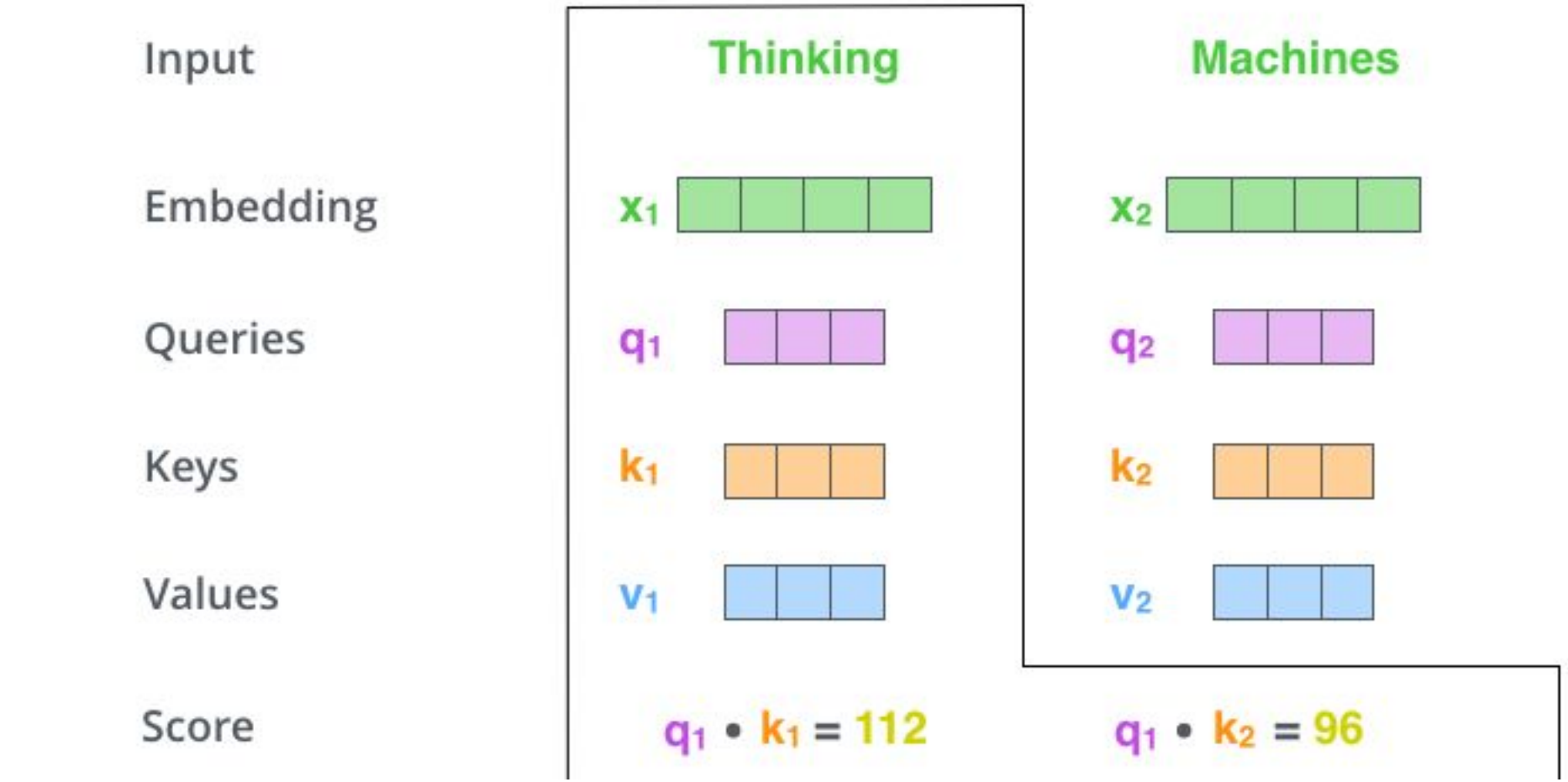


First we create three vectors by multiplying input embedding (1x512)
x_i with three matrices (64x512):

$q_i = x_i W^Q$
$K_i = x_i W^K$
$V_i = x_i W^V$

Self Attention

Now we need to calculate a score to determine how much focus to place on other parts of the input.

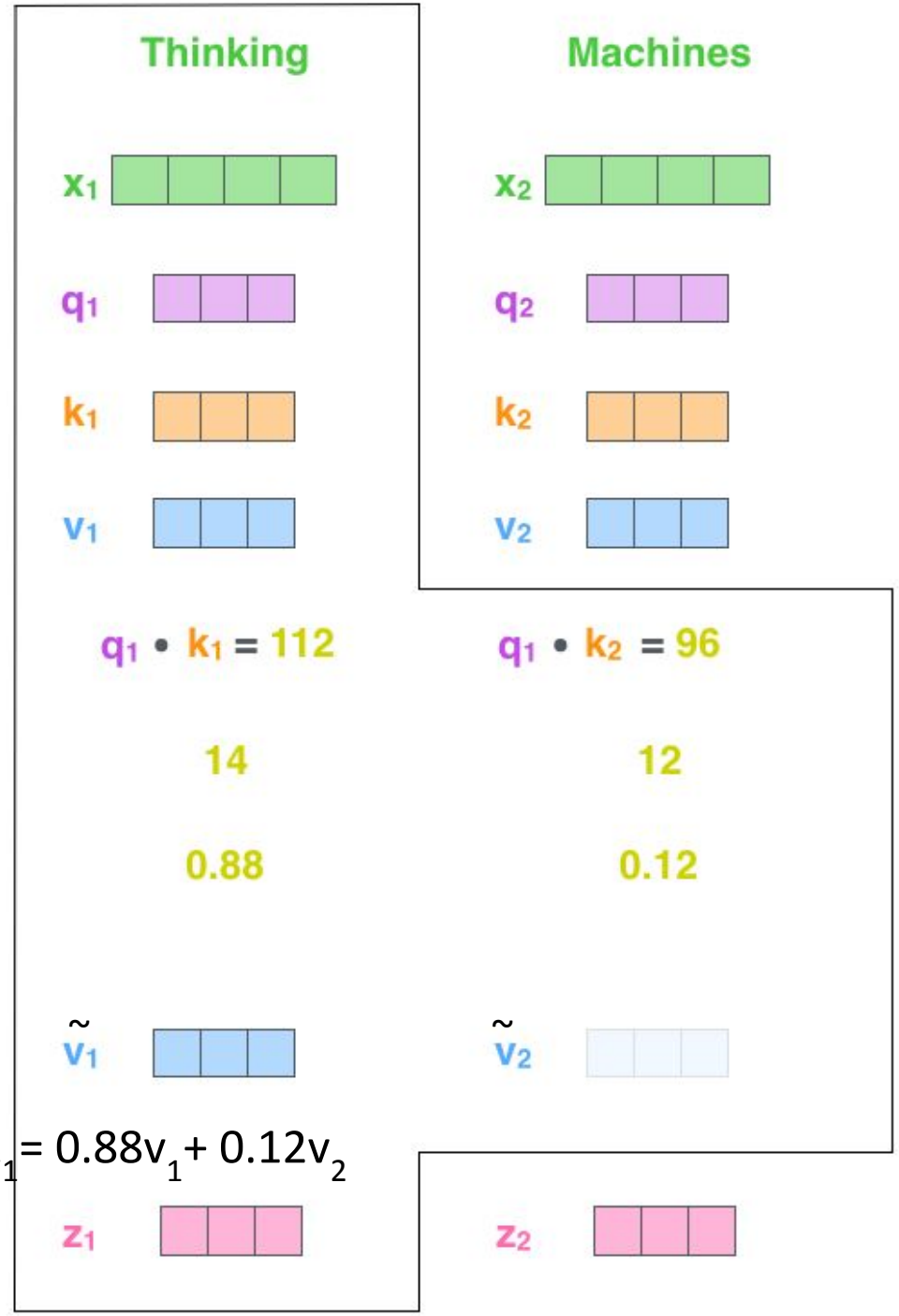


Self Attention

Formula

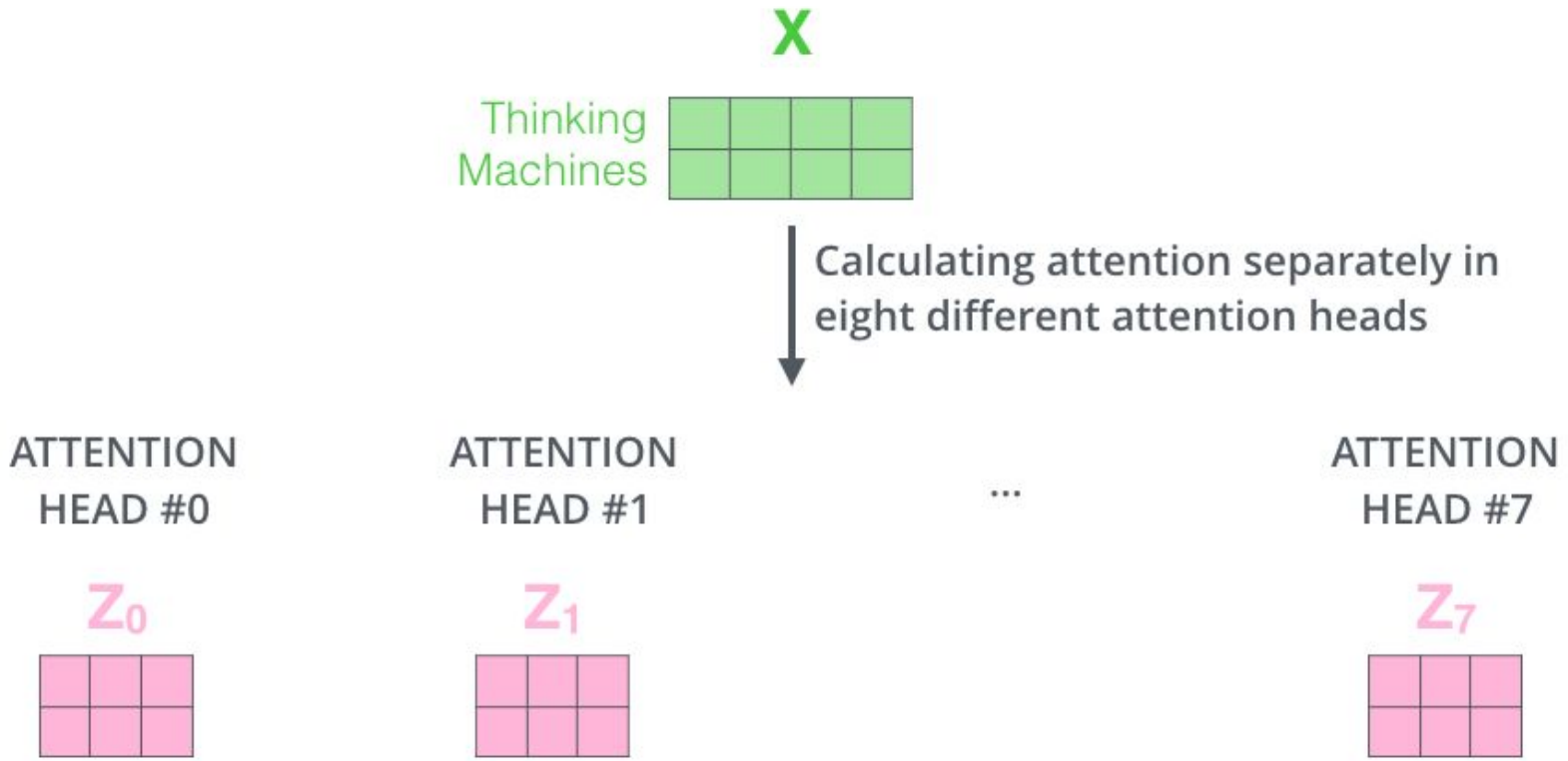
$$\text{softmax}\left(\frac{Q \times K^T}{\sqrt{d_k}}\right) V = Z$$

d_k=64 is dimension of key vector



Multiple heads

1. It expands the model's ability to focus on different positions.
2. It gives the attention layer multiple "representation subspaces"



The output is expecting only a 2x4 (|input|x64) matrix, hence,

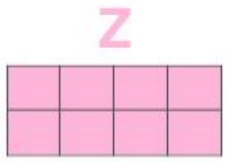
1) Concatenate all the attention heads

2) Multiply with a weight matrix W^O that was trained jointly with the model

X

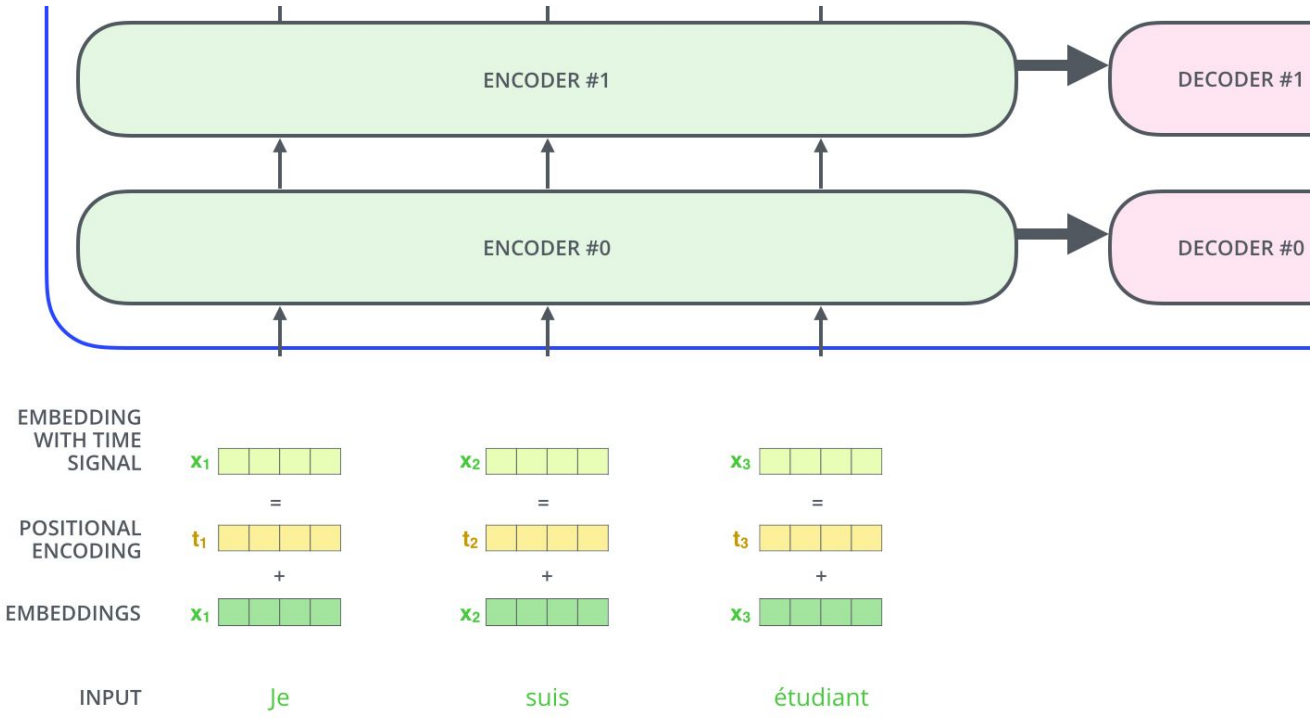
3) The result would be the Z matrix that captures information from all the attention heads. We can send this forward to the FFNN

=



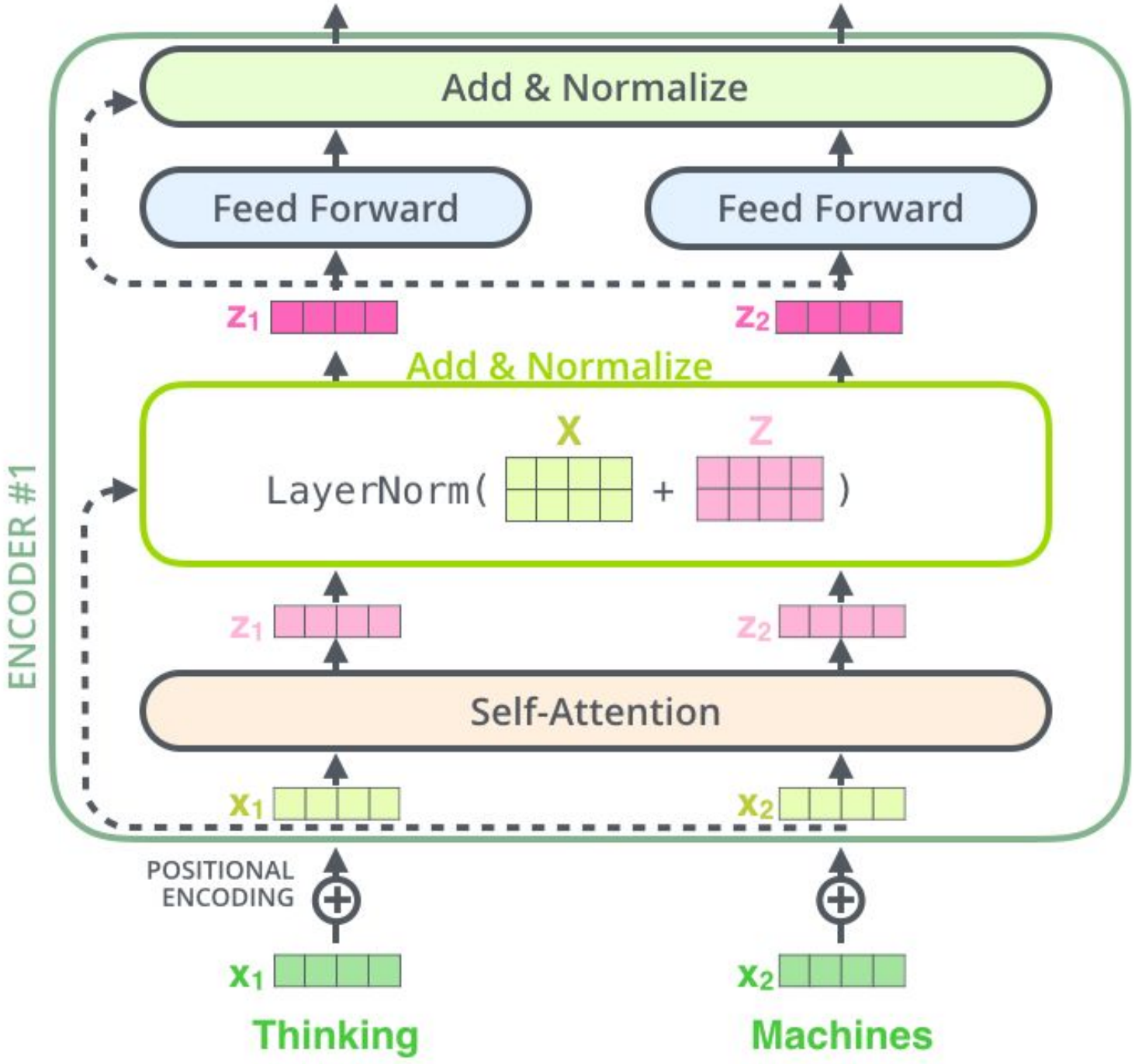
Representing the input order (positional encoding)

The transformer adds a vector to each input embedding. These vectors follow a specific pattern that the model learns, which helps it determine the position of each word, or the distance between different words in the sequence. The intuition here is that adding these values to the embeddings provides meaningful distances between the embedding vectors once they're projected into Q/K/V vectors and during dot-product attention.



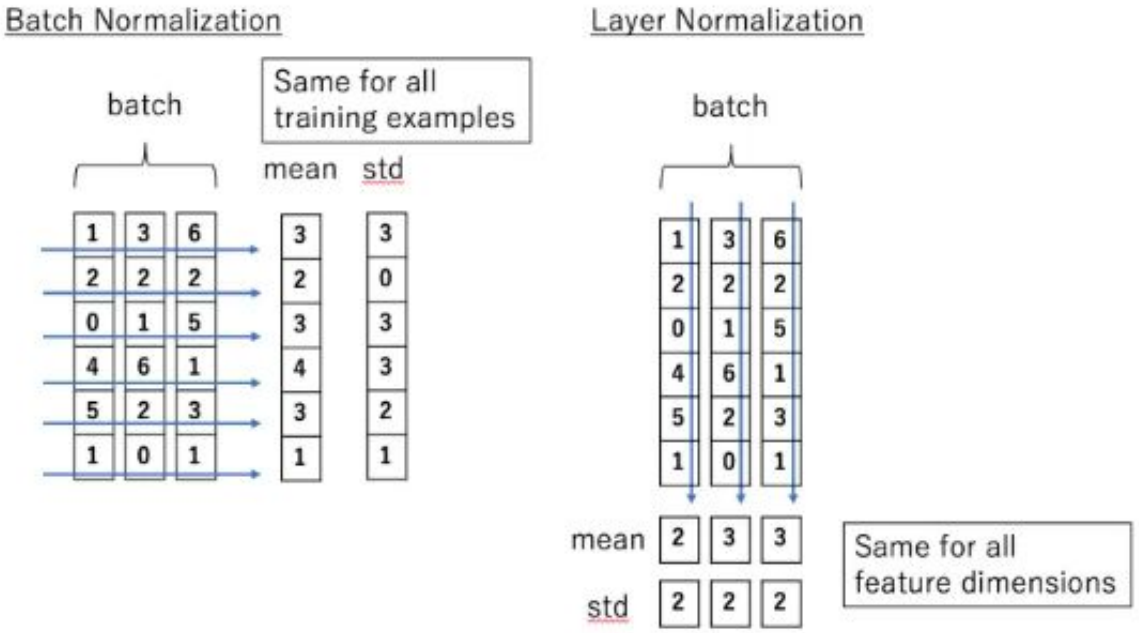
Add and Normalize

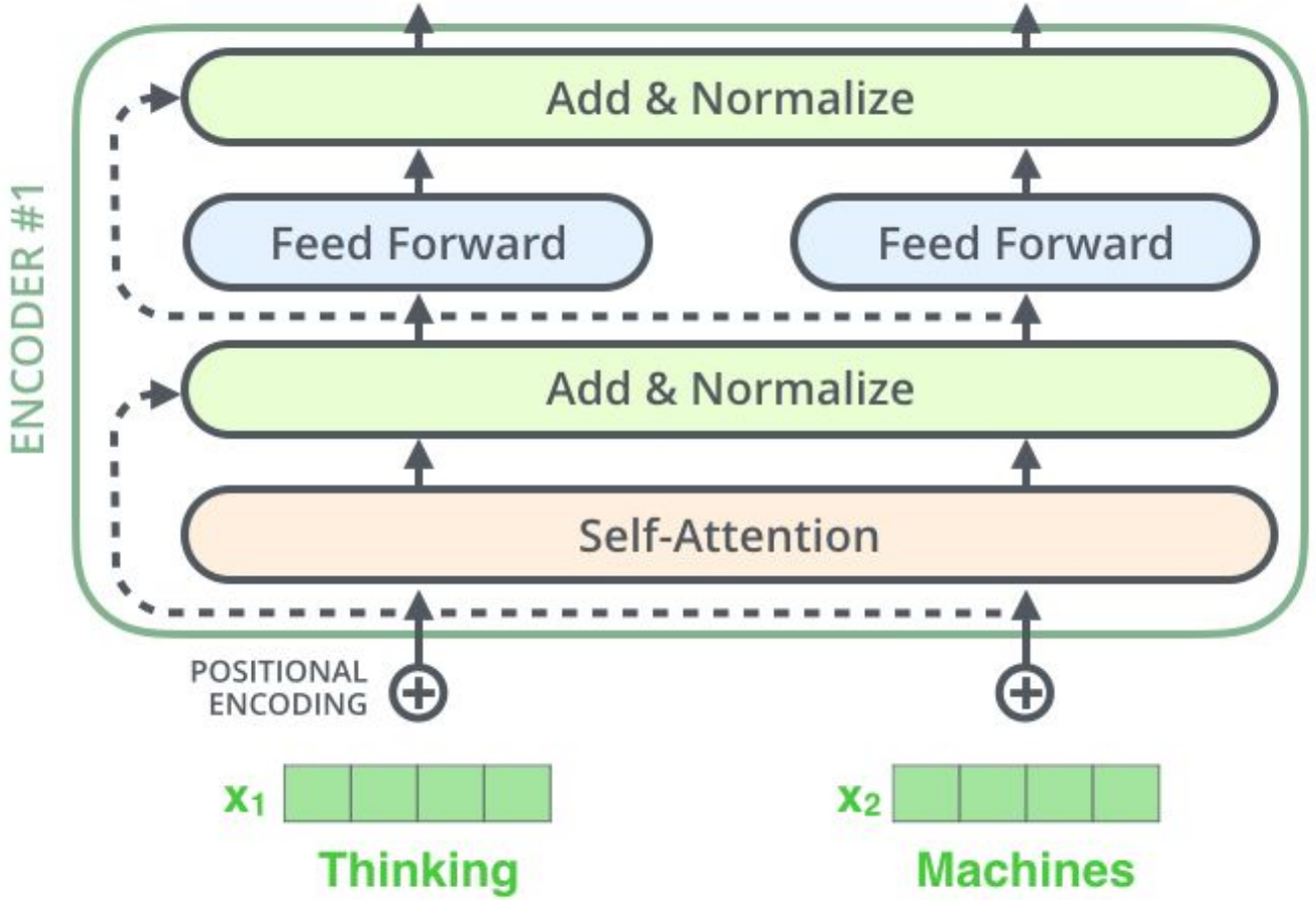
In order to regulate the computation, this is a normalization layer so that each feature (column) have the same average and deviation.



Layer Normalization (Hinton)

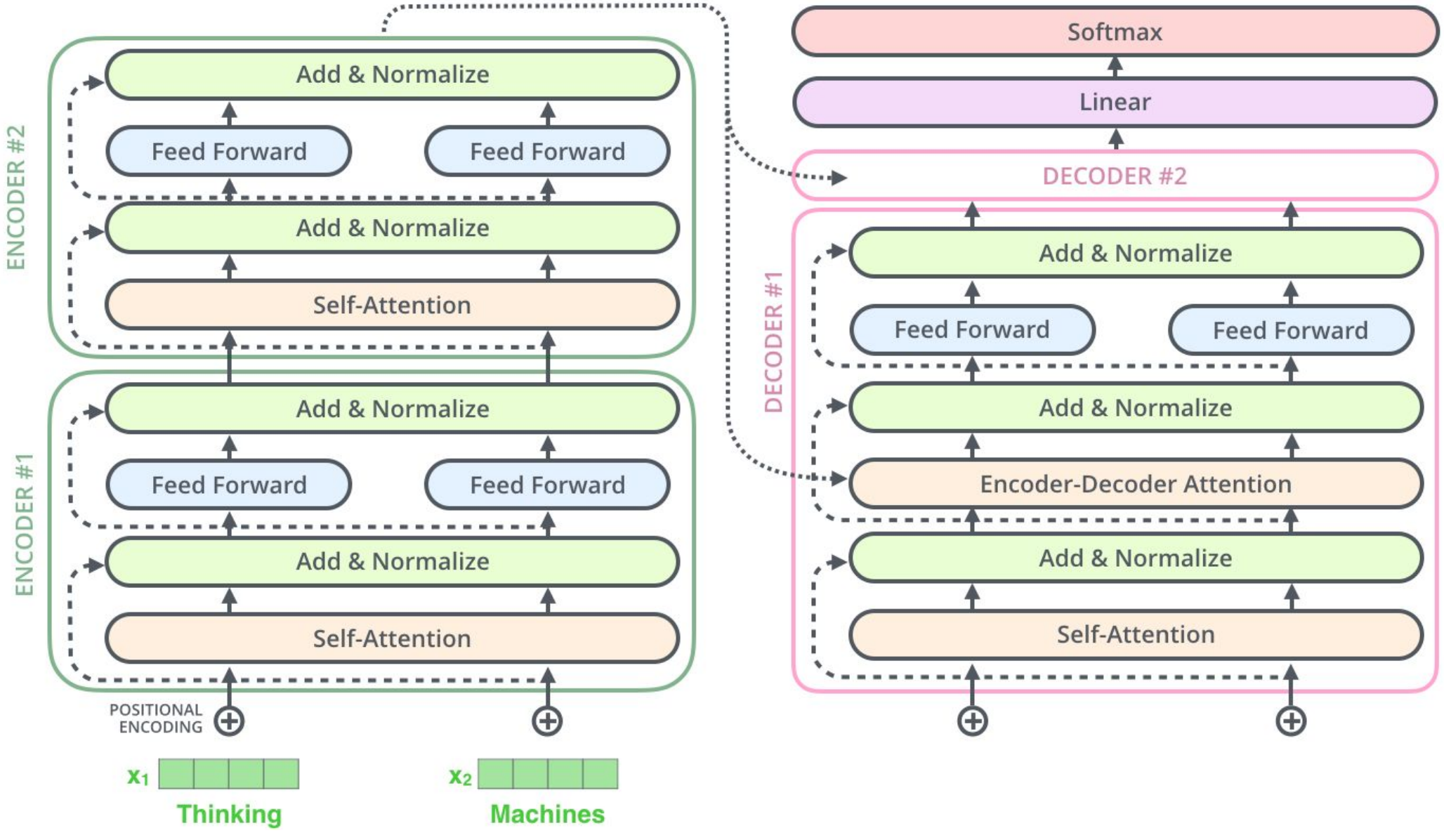
Layer normalization normalizes the inputs across the features.

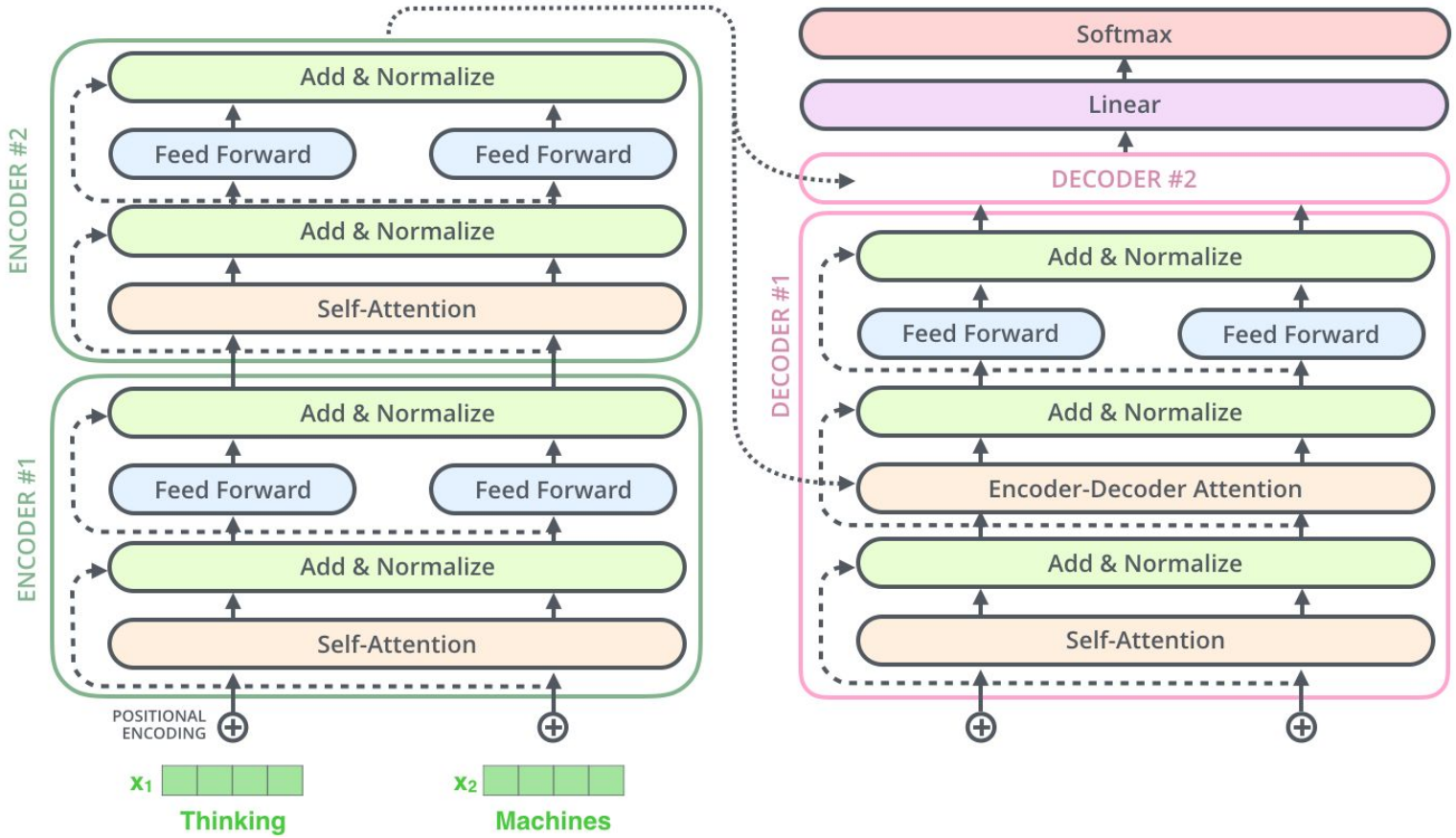




The complete transformer

The encoder-decoder attention is just like self attention, except it uses K, V from the top of encoder output, and its own Q





BERT

https://www.kaggle.com/residentmario/notes-on-gpt-2-and-bert-models

GPT

Literature & Resources for Transformers

Vaswani et al. “Attention is all you need.” 2017.

Resources:

https://nlp.seas.harvard.edu/2018/04/03/attention.html (Excellent explanation of transformer model with codes.)

Jay Alammar, The illustrated transformer (where many pictures in these slides came from):

http://jalammar.github.io/illustrated-transformer/

Kate Logninova: Attention in NLP, summarizes all sorts of attentions.

https://medium.com/@joealato/attention-in-nlp-734c6fa9d983