Jan. 17, 2020

CS 886 Deep Learning and NLP

Ming Li

CONTENT

- 01. Word2Vec
- 02. Attention / Transformers
- 03. GPT / BERT
- 04. Simplicity, ALBERT, Single headed attention RNN
- 05. Student presentations Starting Feb. 3
- 06. Student presentations ending March 30
- 07. Student short presentations of research projects

02

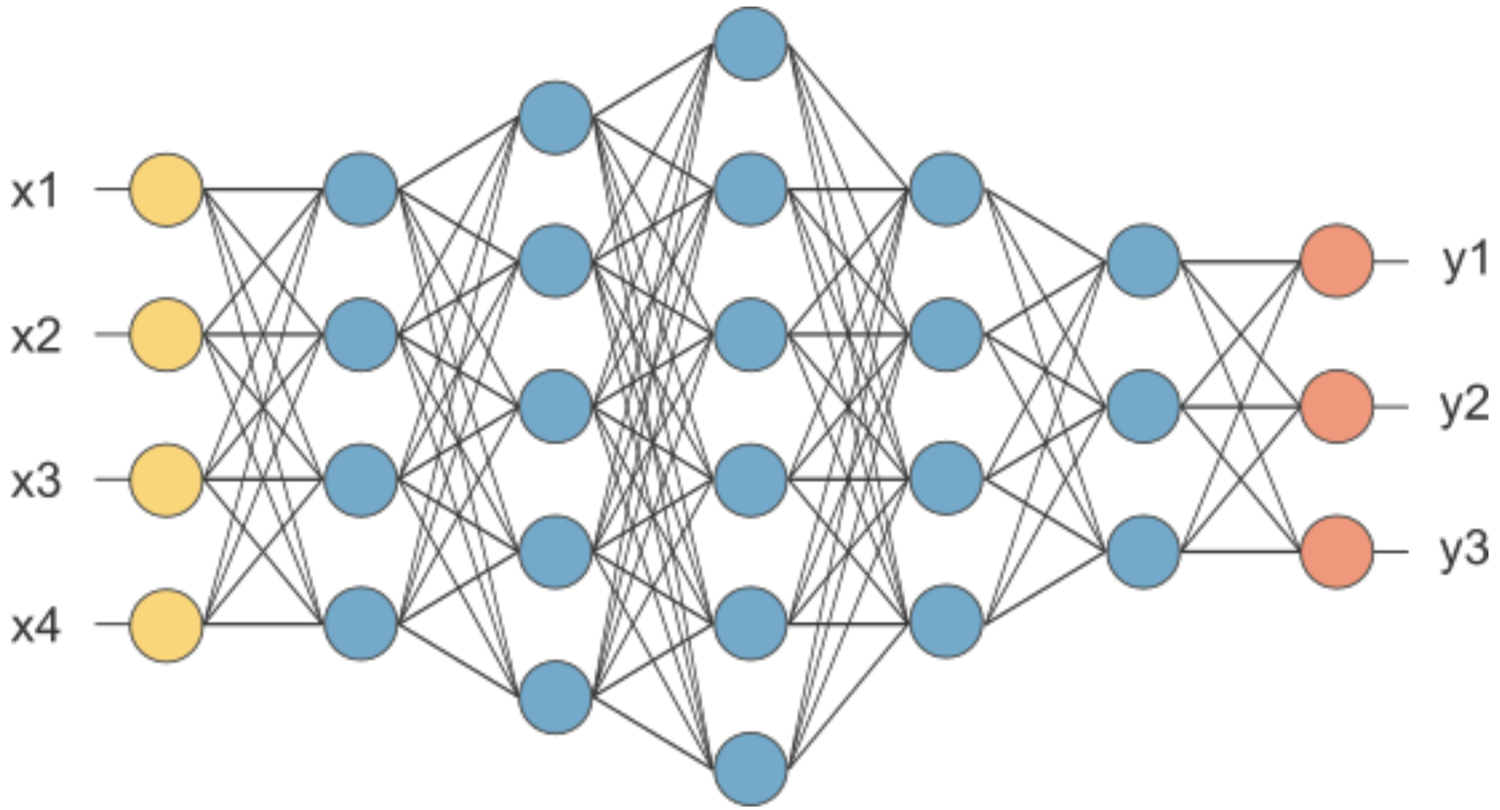
Attention and Transformers

LECTURE TWO

Plan

1. Basic models, related to transduction models and attention.
2. Encoder-Decoder model, using recurrent networks such as LSTM.
3. Attention and Transformers

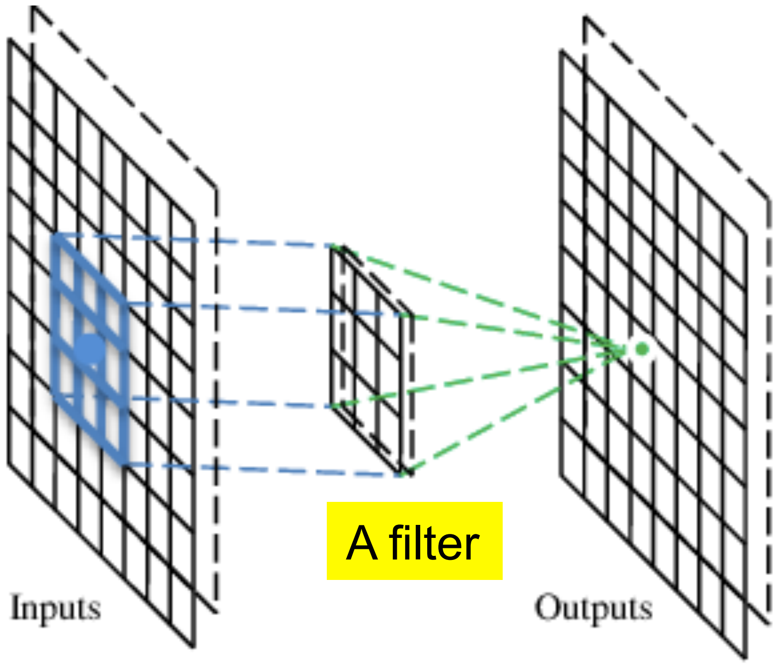
1. Fully connected network, feedforward network



To learn the weights on the edges

2. CNN

A CNN is a neural network with some convolutional layers (and some other layers). A convolutional layer has a number of filters that do convolutional operation.



Convolutional layer

These are the network parameters to be learned.

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

Input

Filter 1

1	-1	-1
-1	1	-1
-1	-1	1

Filter 2

-1	1	-1
-1	1	-1
-1	1	-1

⋮ ⋮

Each filter detects a small pattern (3 x 3).

Convolution Operation

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

stride=1

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

Input

Dot product

3

-1

Convolution

stride=1

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

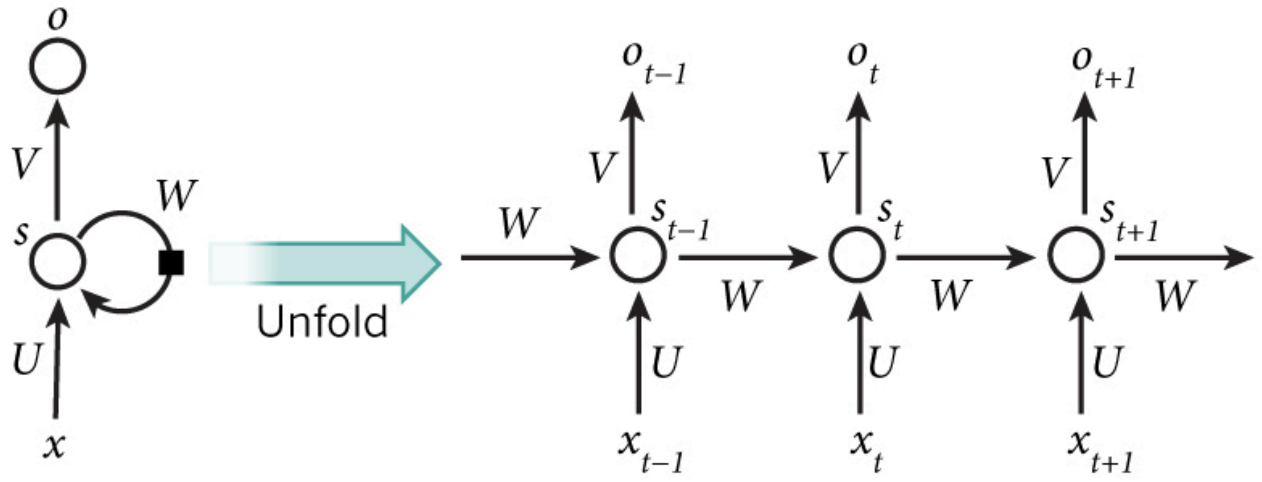
Input

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

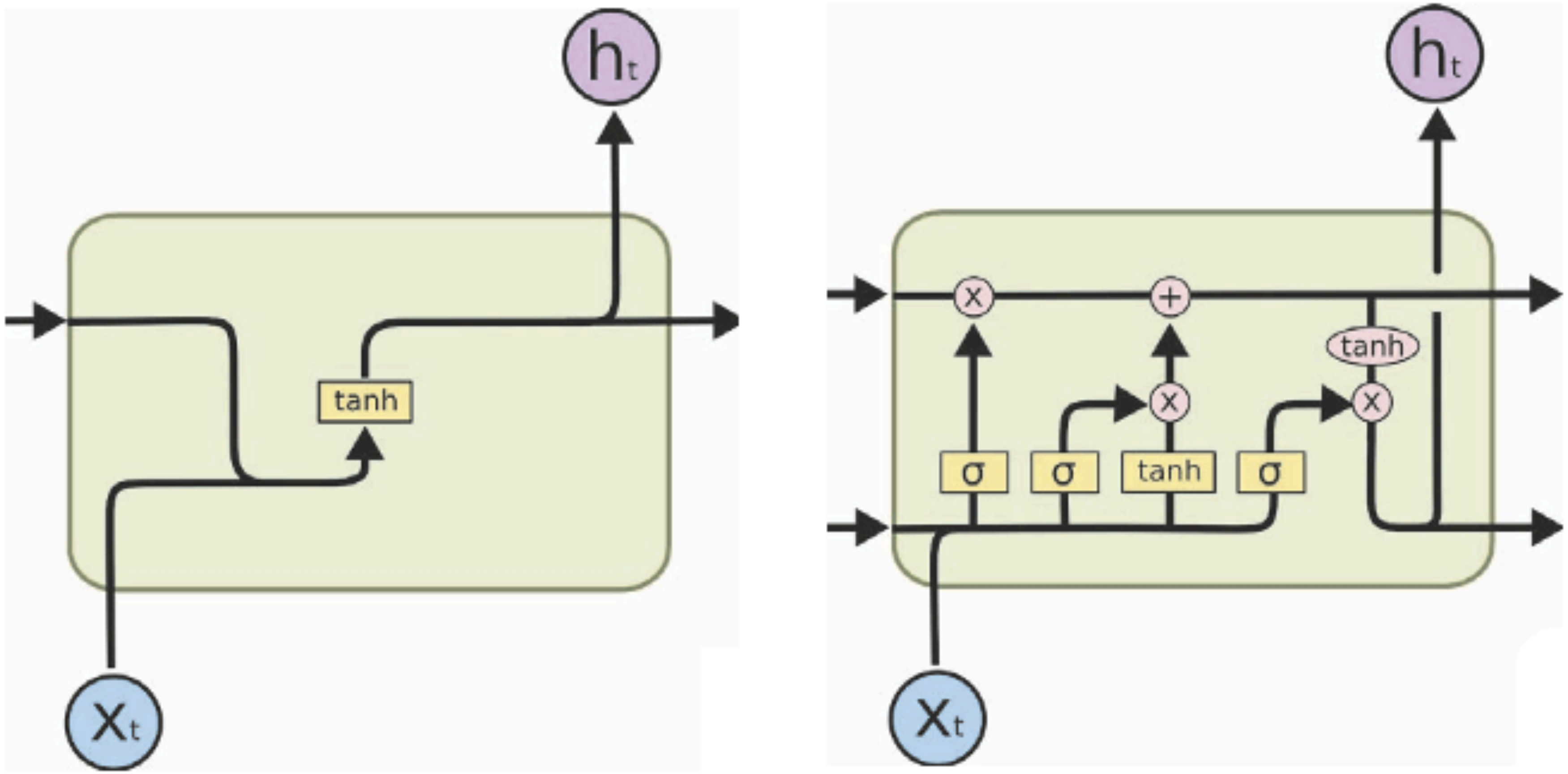
3	-1	-3	-1
-3	1	0	-3
-3	-3	0	1
3	-2	-2	-1

3. RNN



Parameters to be learned:
U, V, W

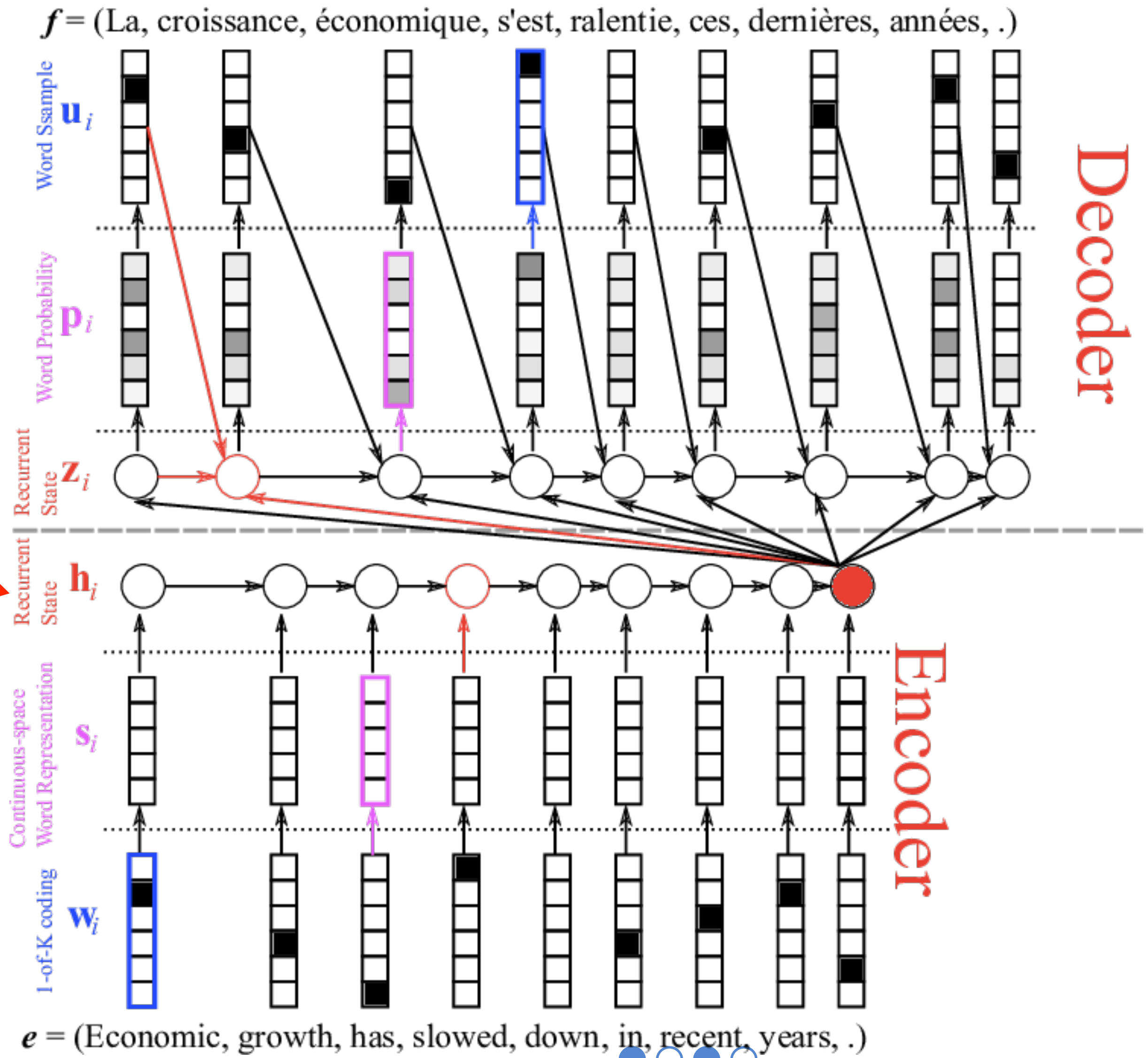
Simple RNN vs LSTM



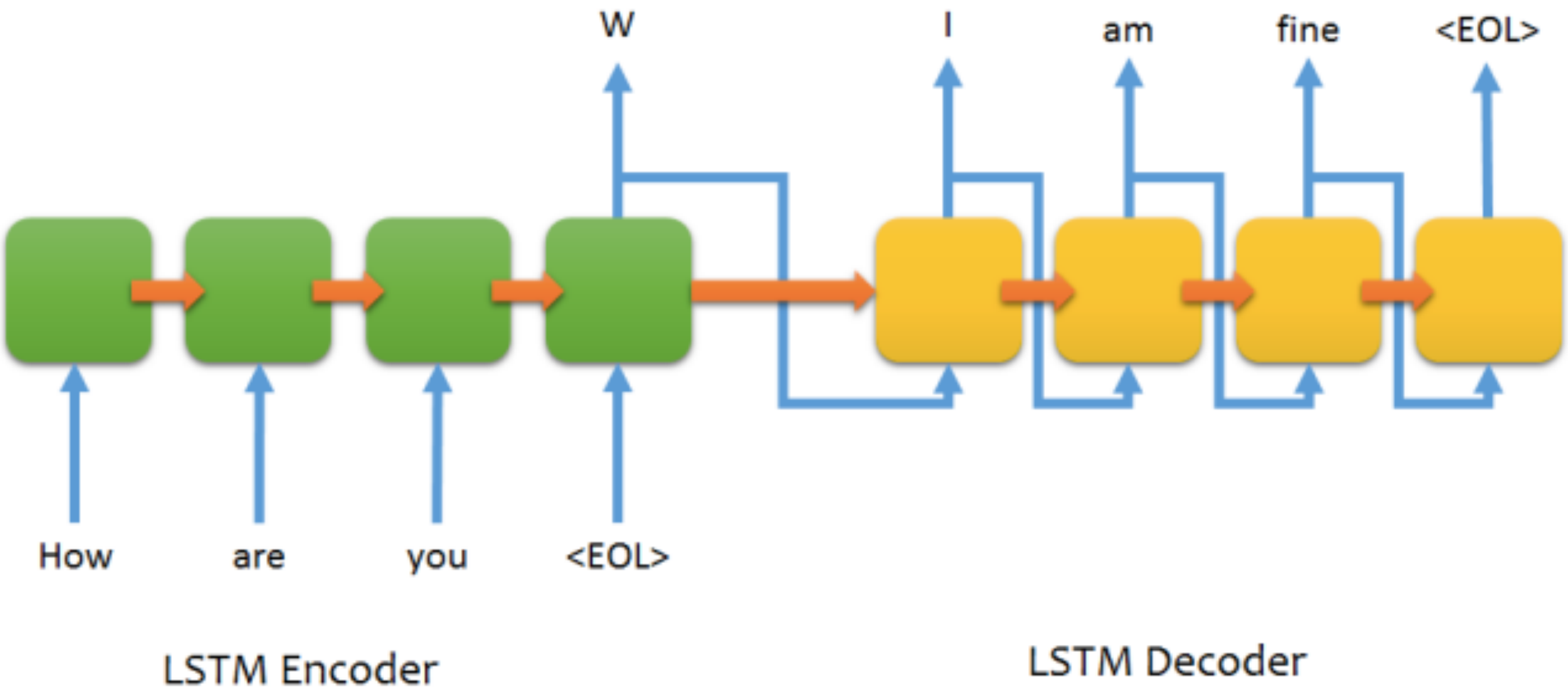
(a) RNN (b) LSTM

Encoder-Decoder machine translation

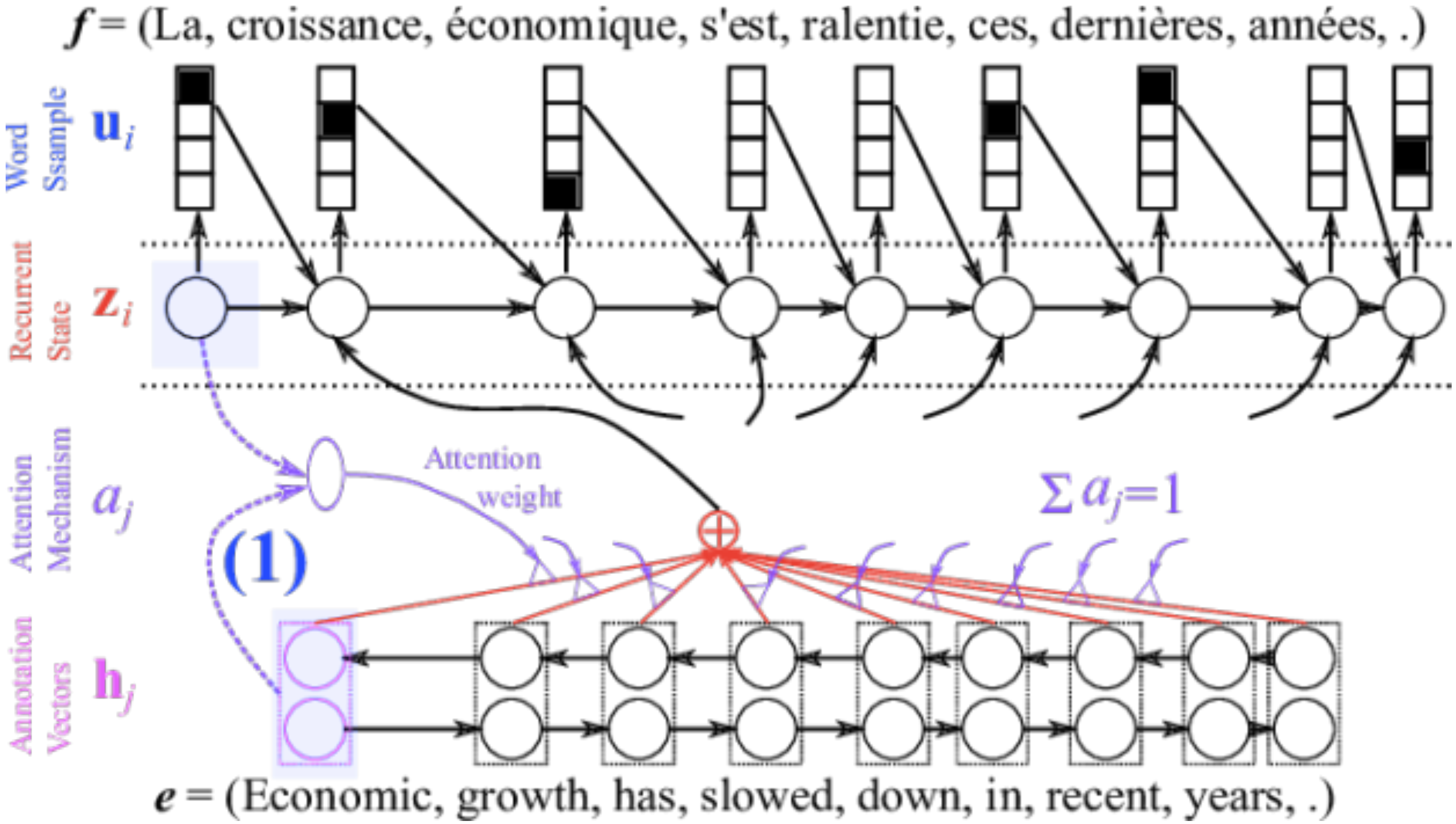
LSTM



Encoder-Decoder LSTM structure for chatting (for non-intelligent beings)

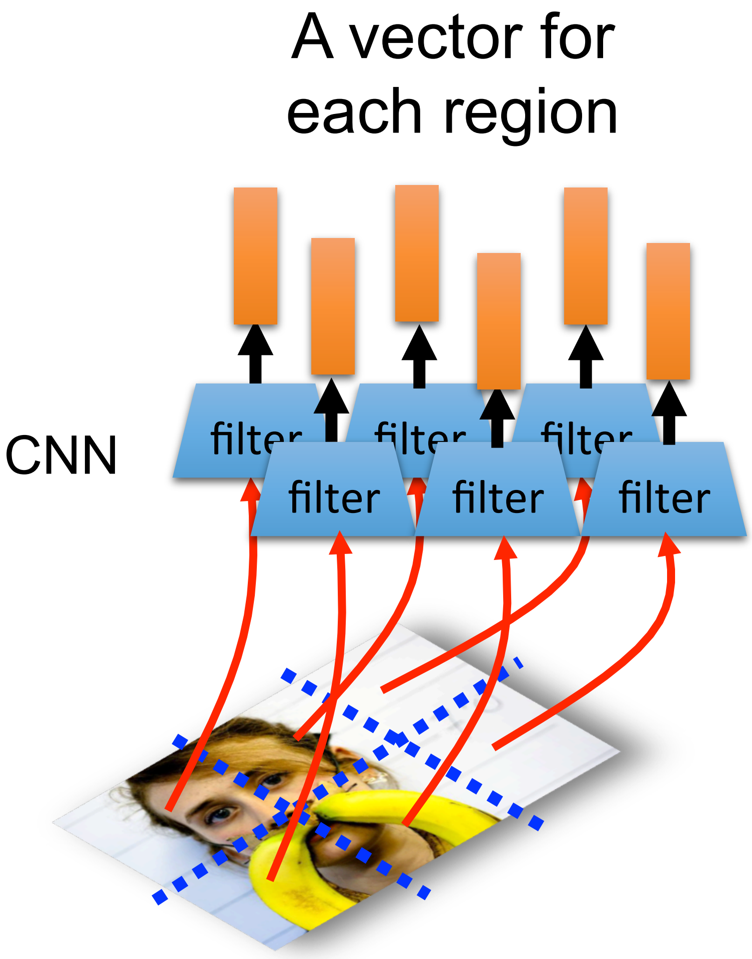


Attention



f = (La, croissance, économique, s'est, ralentie, ces, dernières, années, .)
Word Ssample
u_i
Recurrent State
z_i
Attention Mechanism
a_j
Attention weight
Σ a_j=1
(1)
Annotation Vectors
h_j
e = (Economic, growth, has, slowed, down, in, recent, years, .)

Image caption generation using attention



z^0 is initial parameter, it is also learned

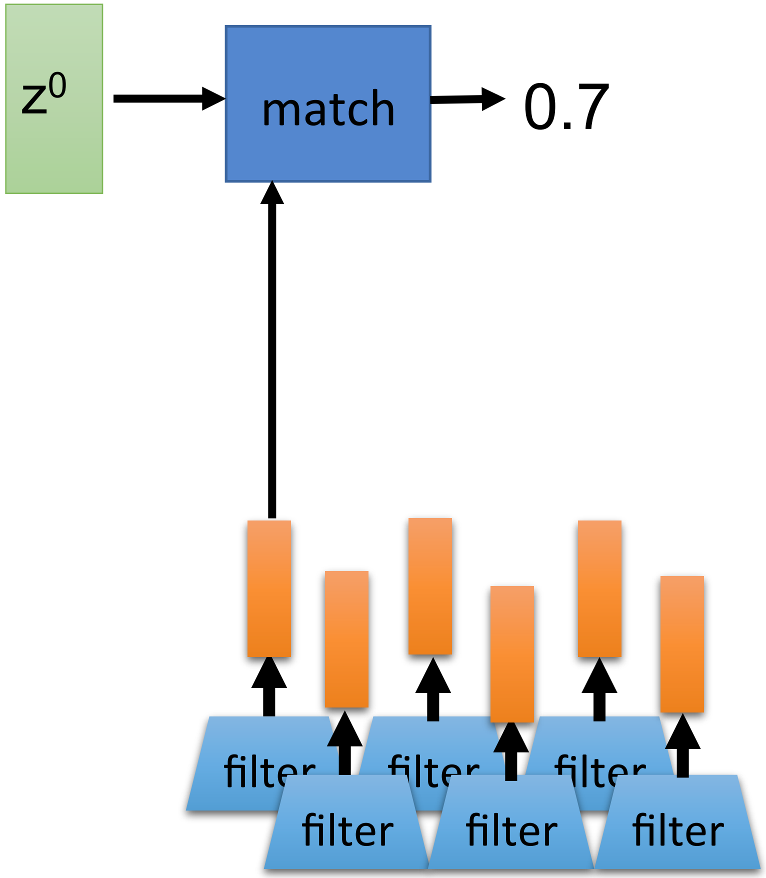


Image caption generation using attention

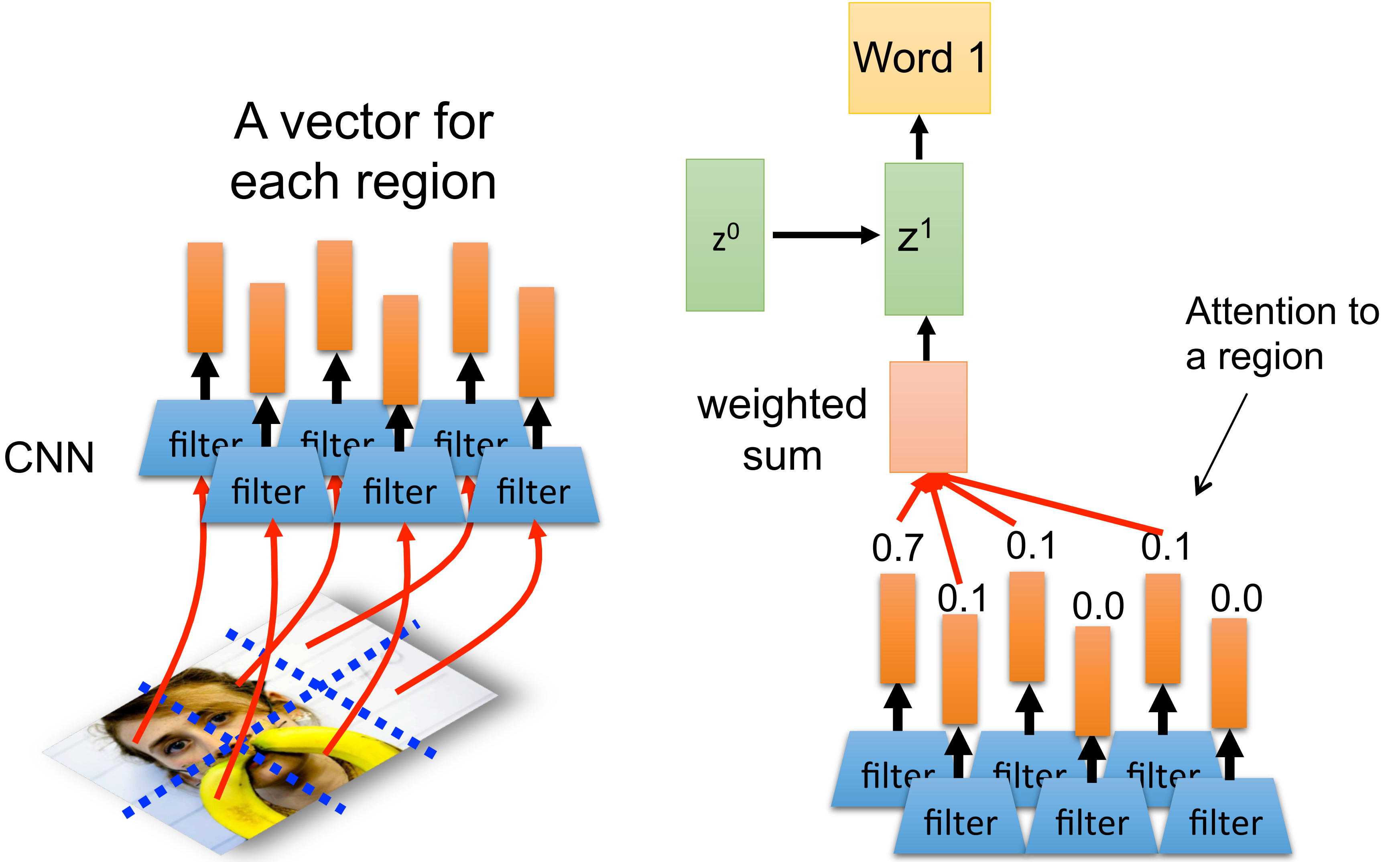
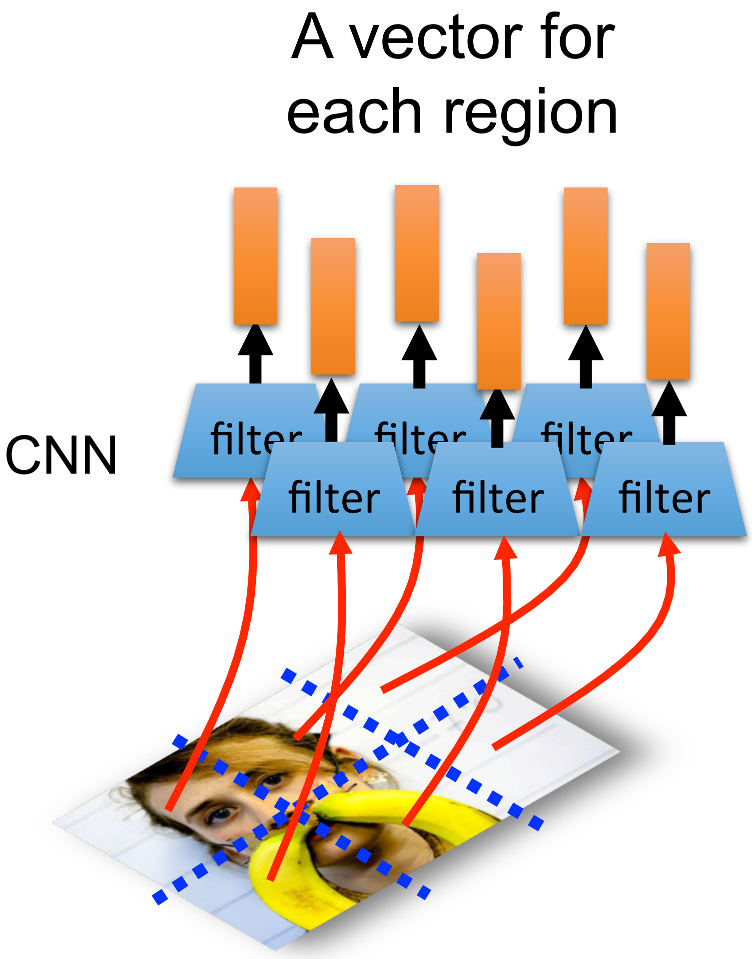


Image caption generation using attention



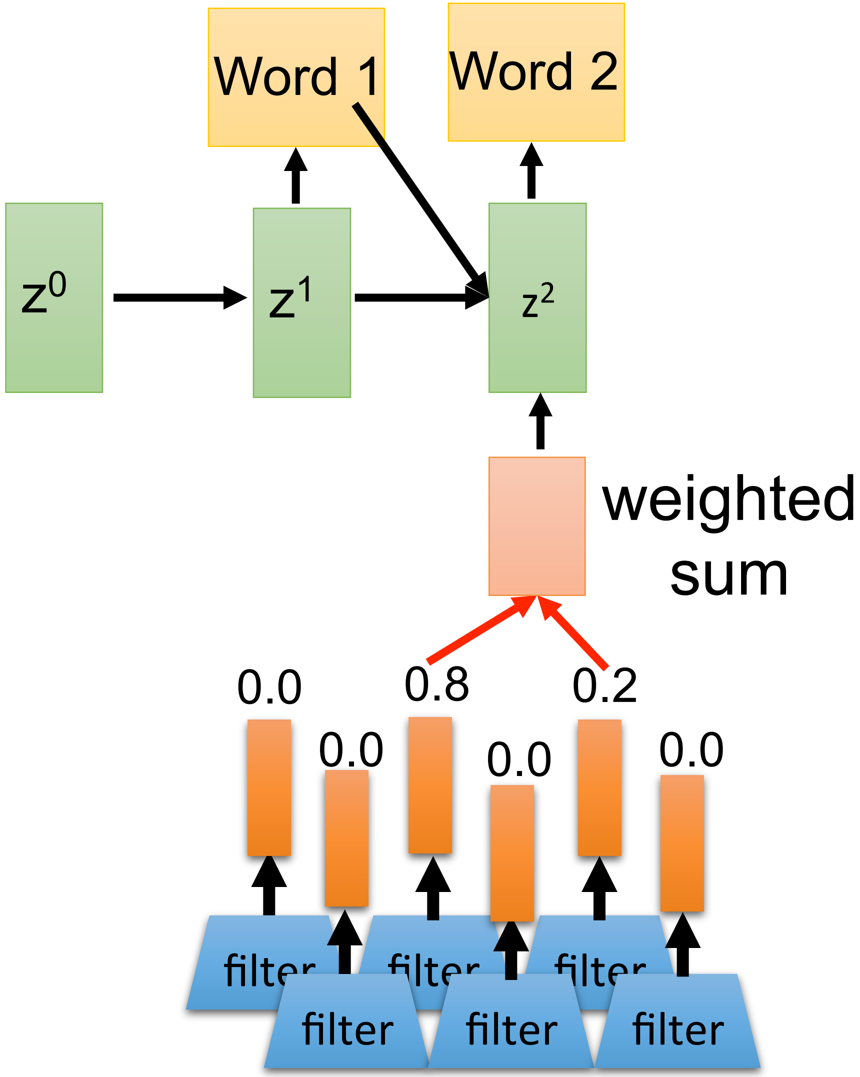
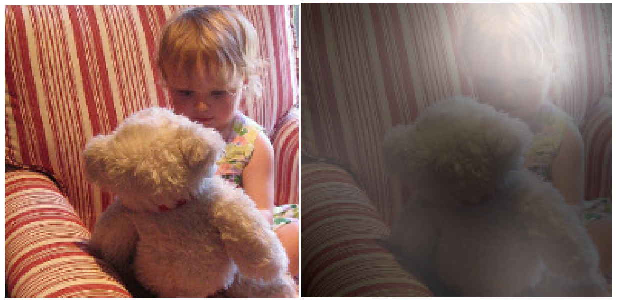


Image caption generation using attention

A woman is throwing a frisbee in a park.

A dog is standing on a hardwood floor.

A stop sign is on a road with a mountain in the background.



A little girl sitting on a bed with a teddy bear.

A group of people sitting on a boat in the water.

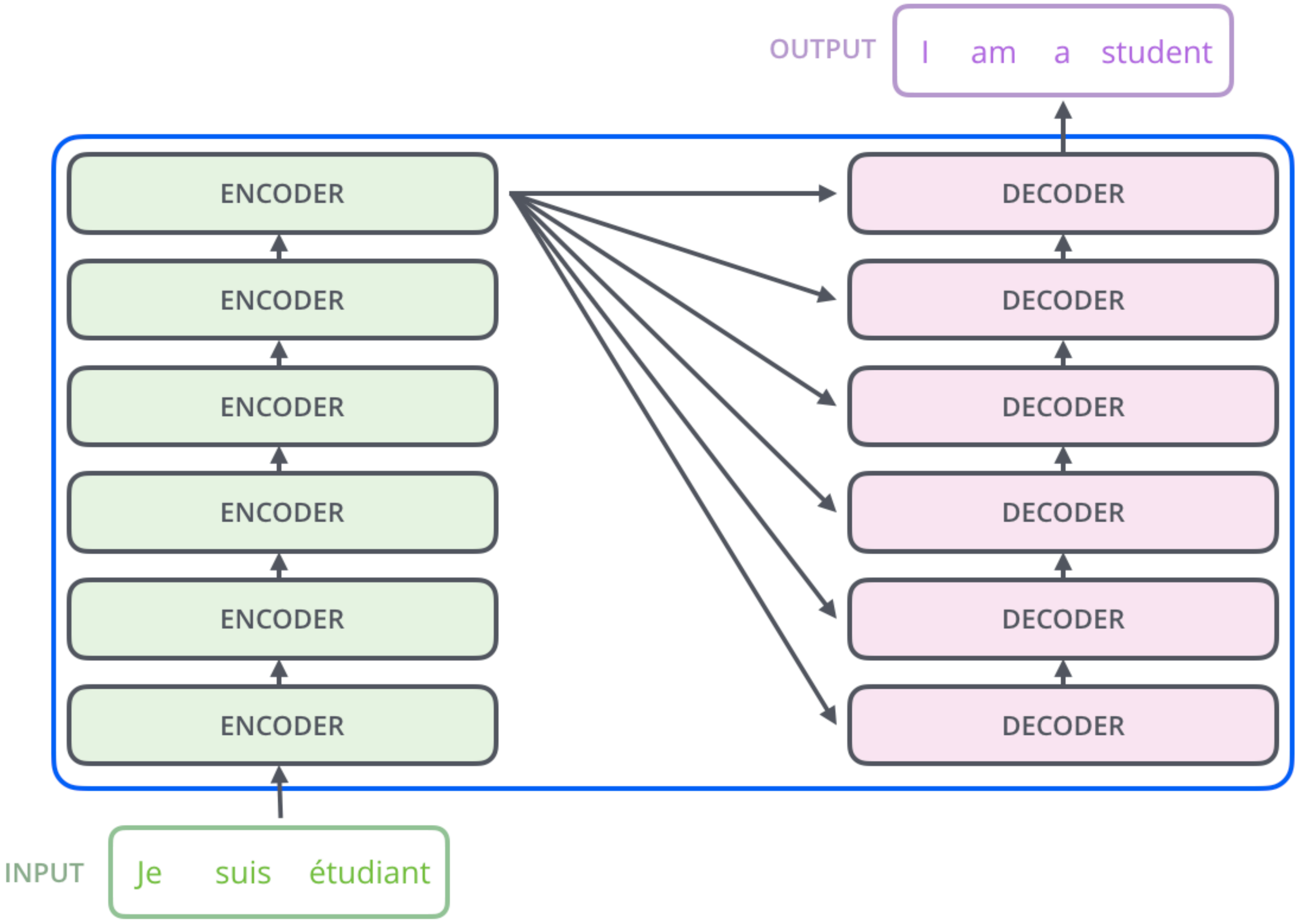
A giraffe standing in a forest with trees in the background.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard Zemel, Yoshua Bengio, "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention", ICML, 2015

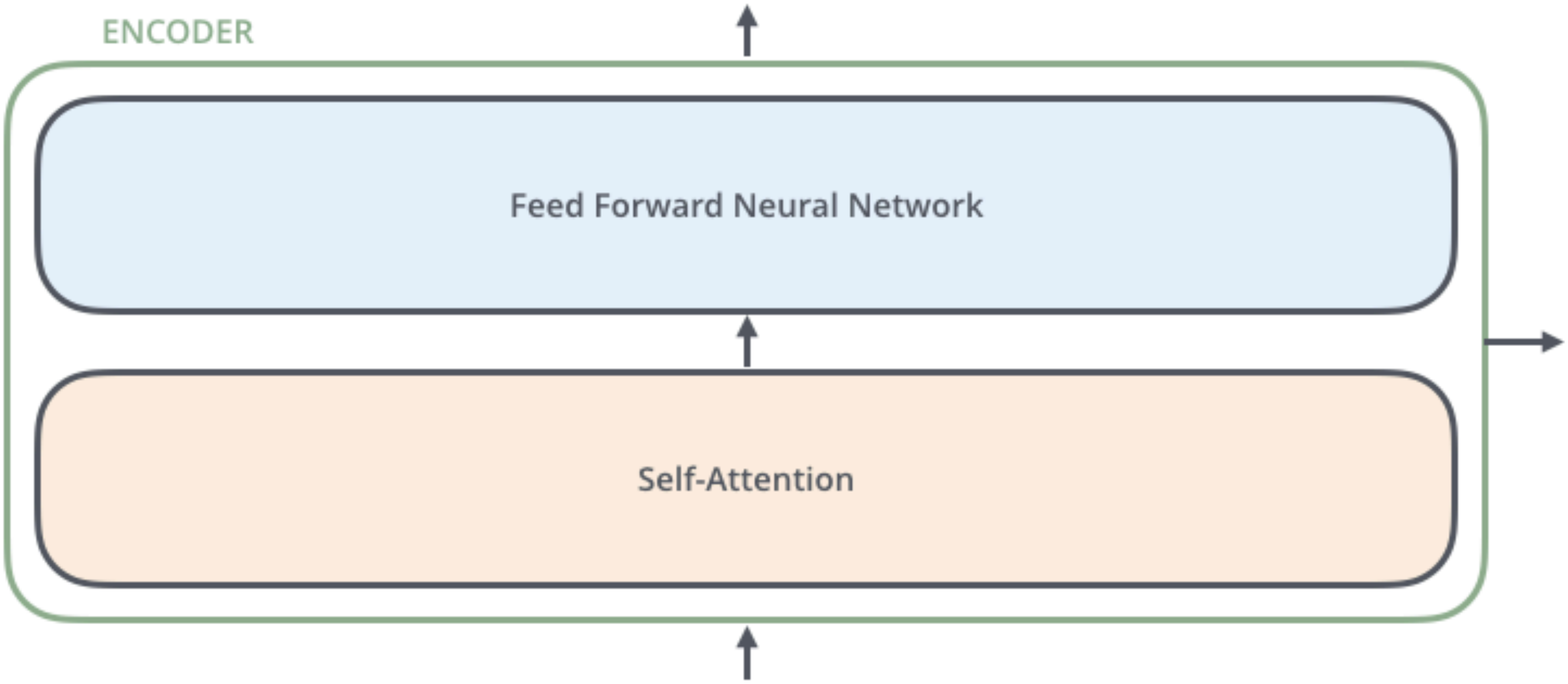
Many new ideas

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.

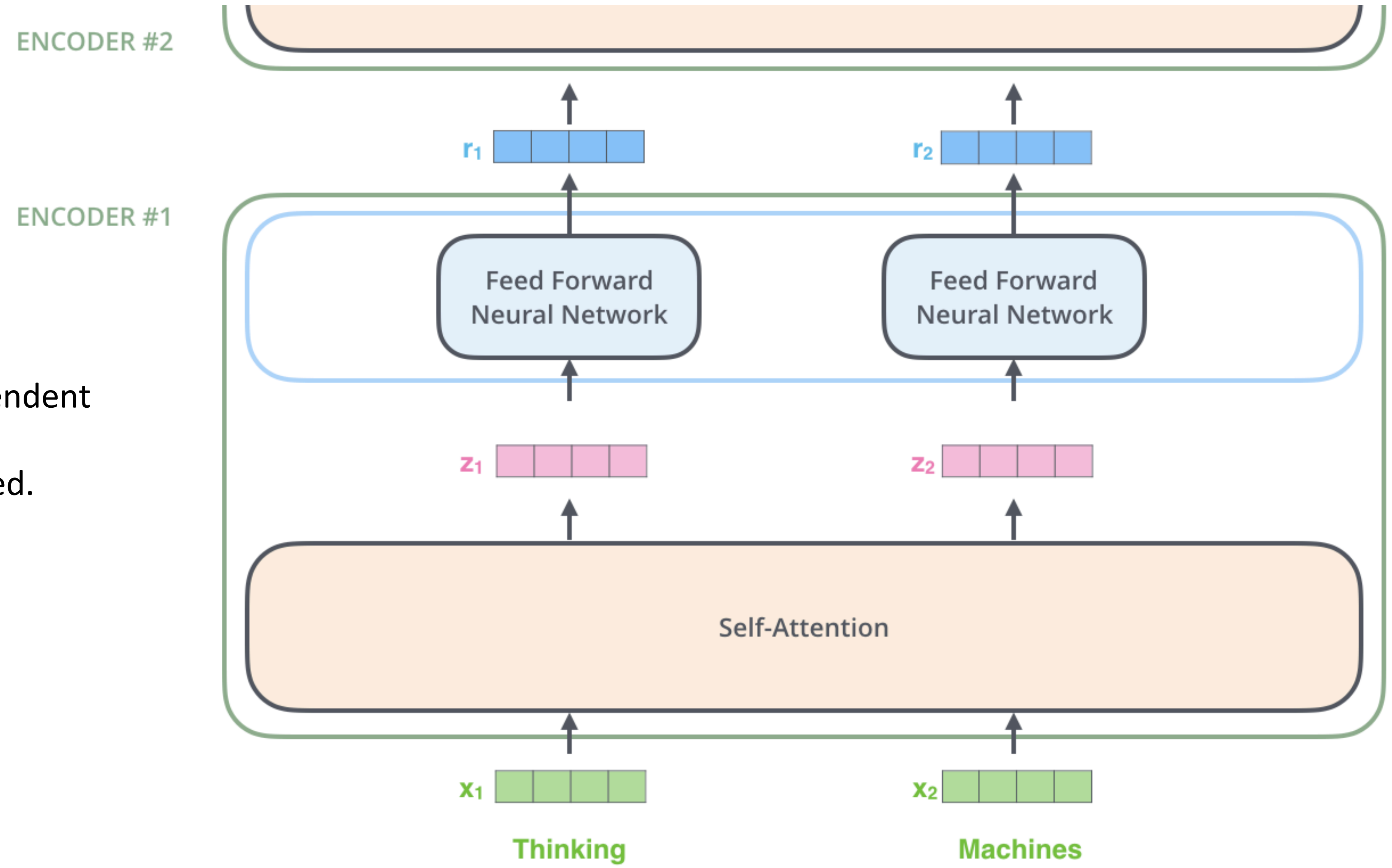
Transformer



An Encoder Block: same structure, different parameters

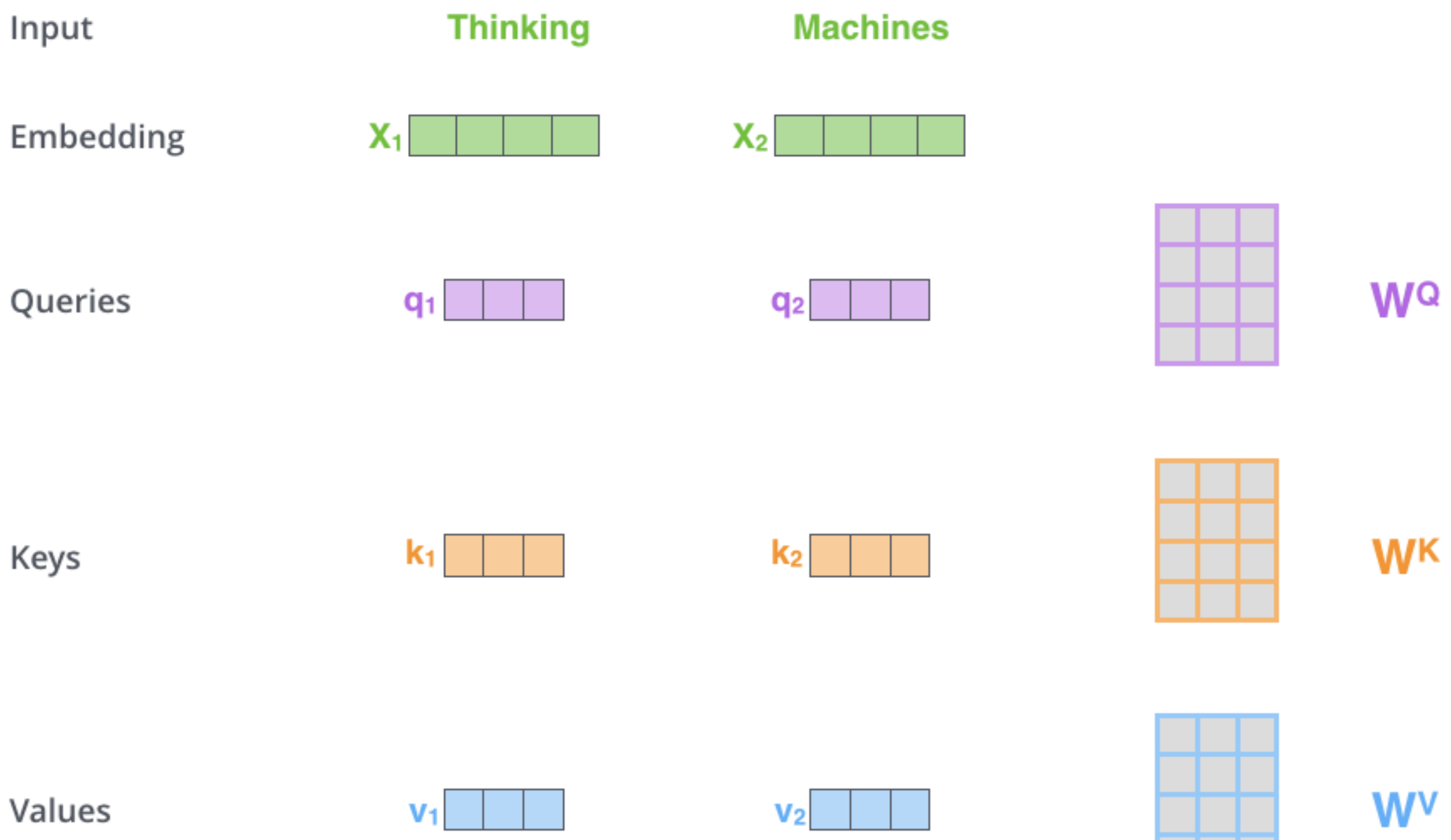


Encoder



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

Self Attention

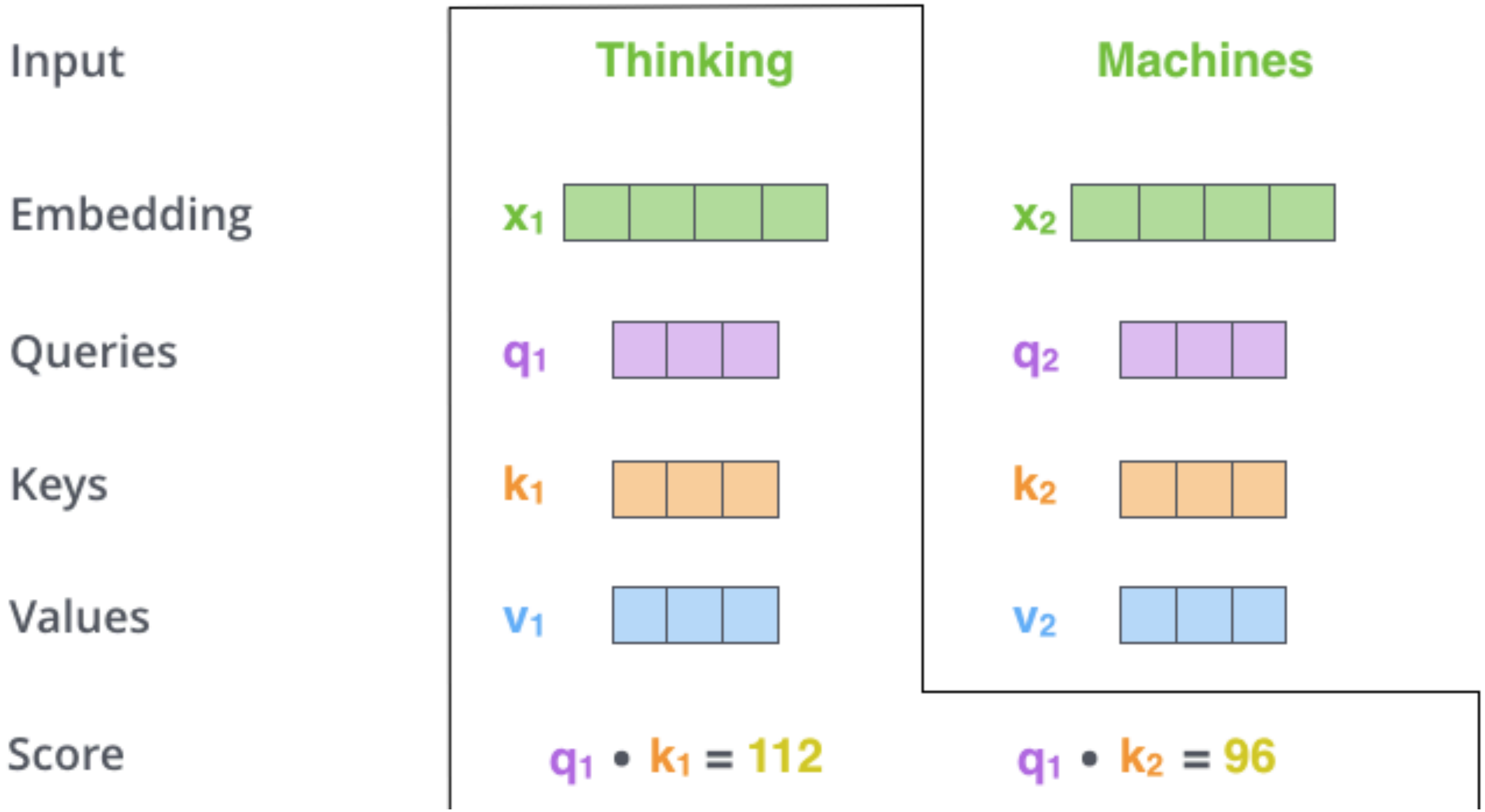


First we create three vectors by multiplying input embedding (1x512) x_i with three matrices (64x512):

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

Self Attention

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

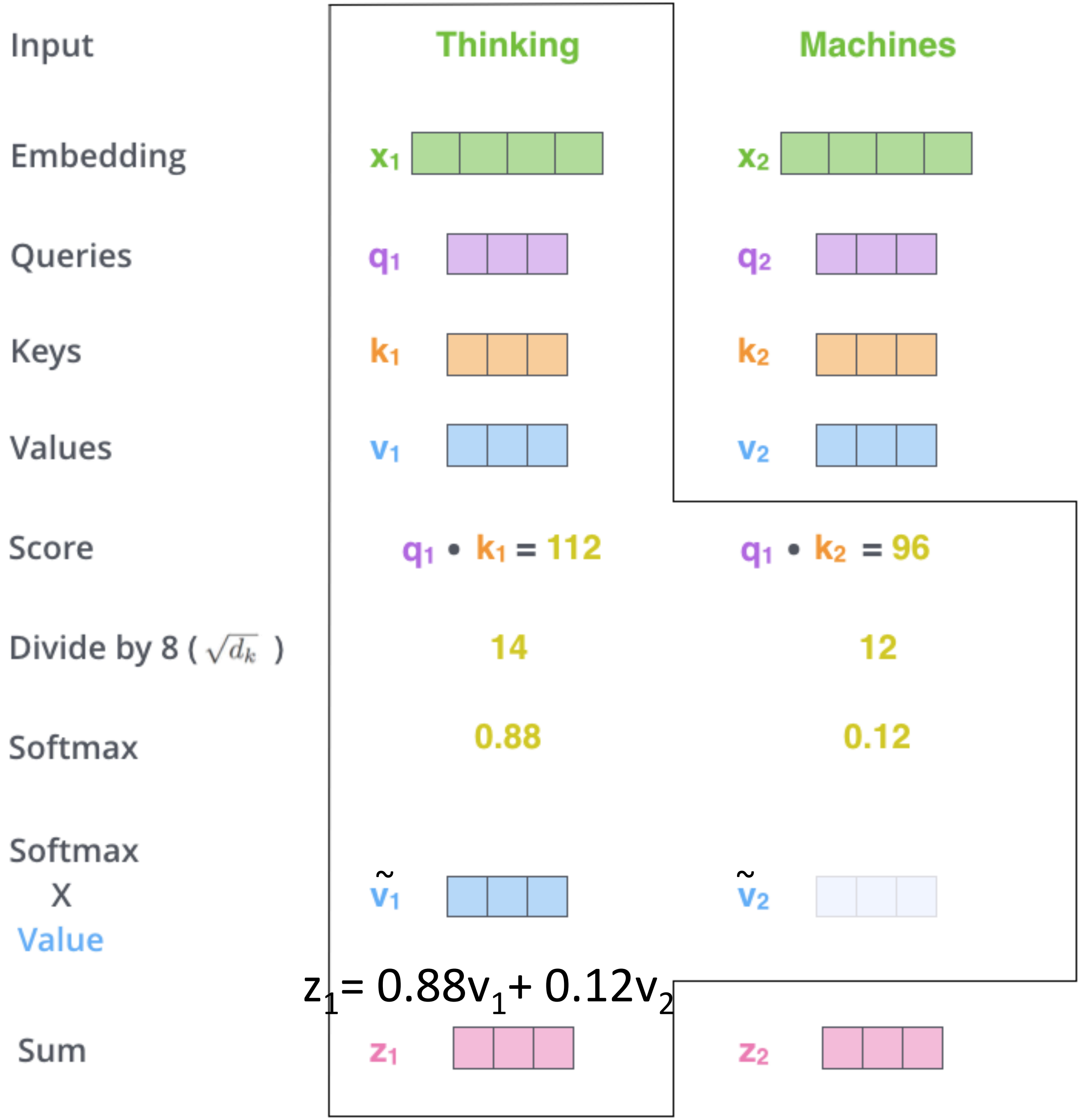


Self Attention

Formula

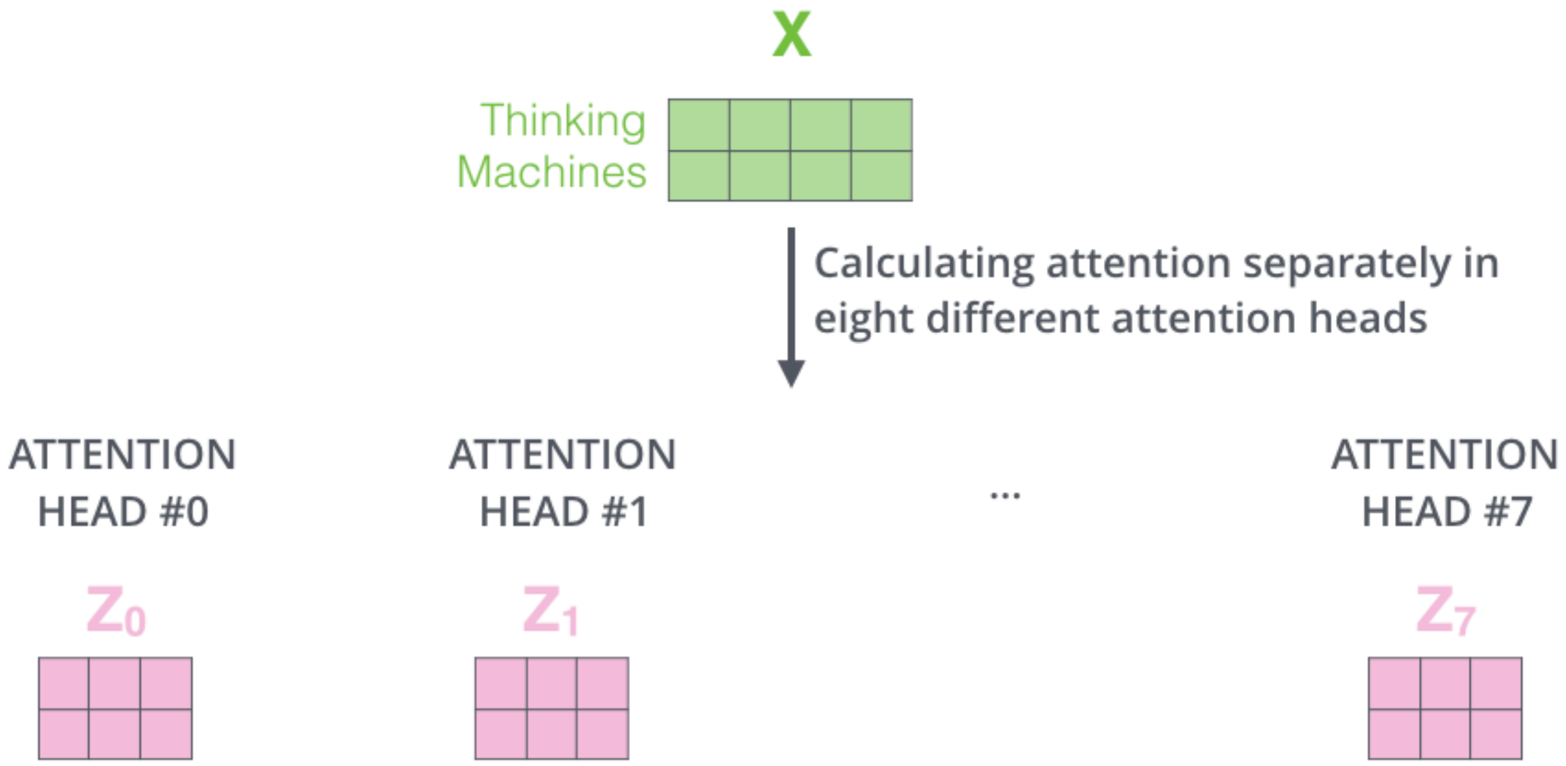
$$\text{softmax}\left(\frac{Q \times K^T}{\sqrt{d_k}}\right) V = Z$$

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"

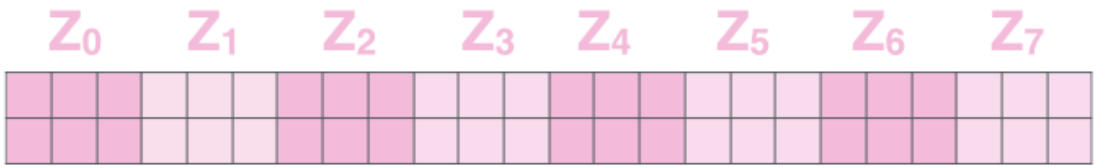


02

Attention and Transformers

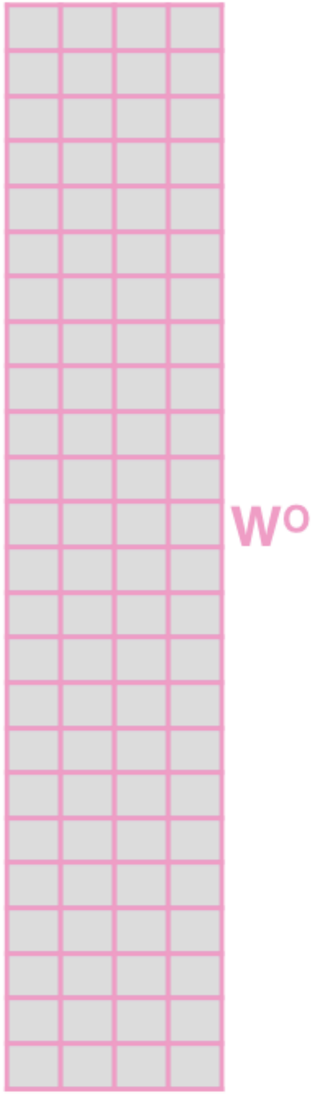
The output is expecting only a 2x4 (|input|x64) matrix, hence,

1) Concatenate all the attention heads



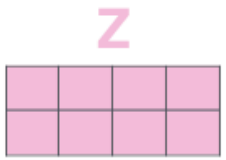
2) Multiply with a weight matrix W^O that was trained jointly with the model

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

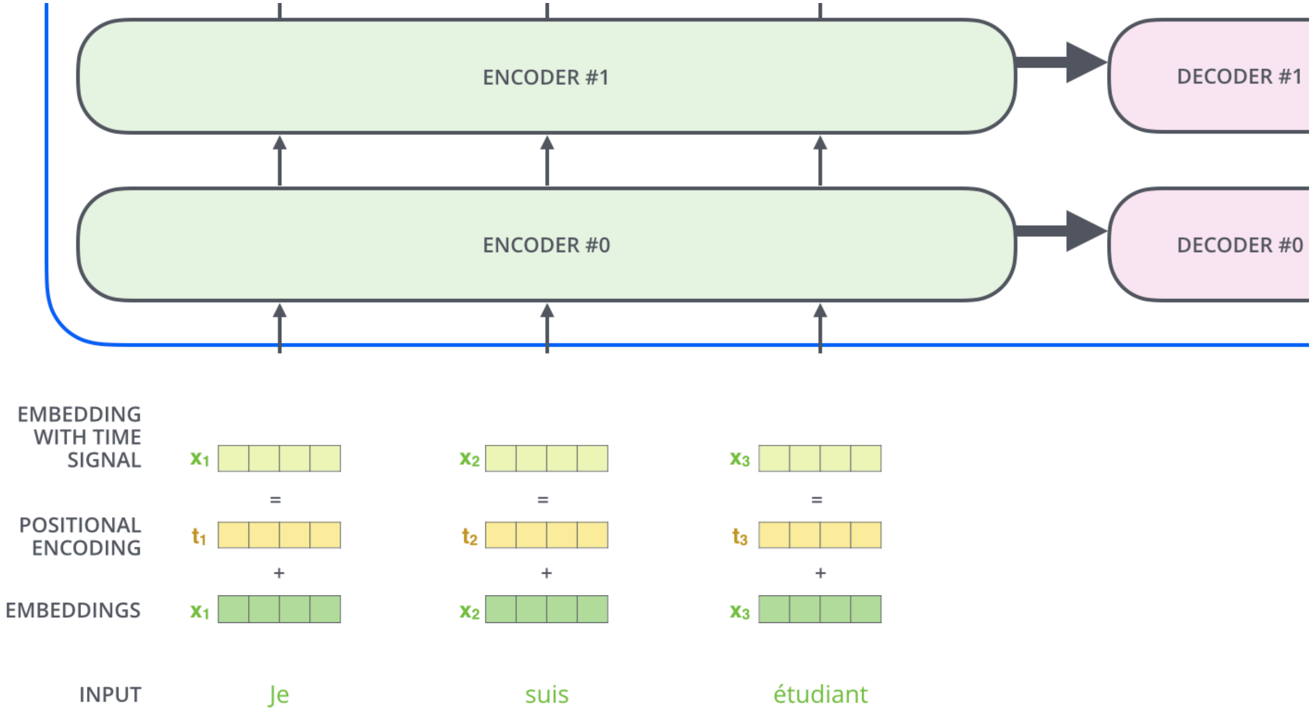
=



Representing the input order (positional encoding)

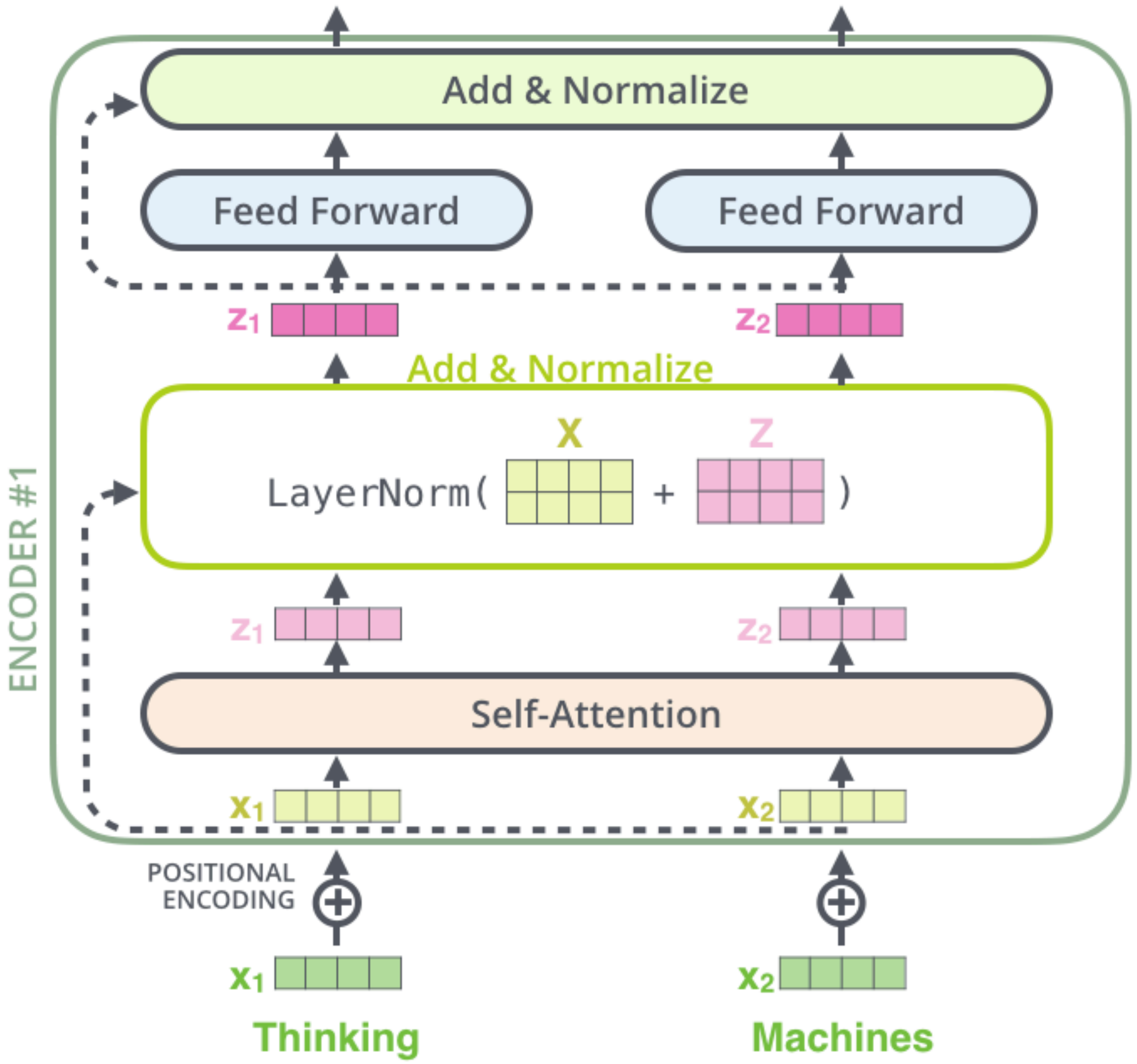
The transformer adds a vector to each input embedding. These vectors follow a specific pattern that the model learns, which helps it determine the position of each word, or the distance between different words in the sequence. The intuition here is that adding these values to the embeddings provides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

Can somebody present positional encoding following https://kazemnejad.com/blog/transformer_architecture_positional_encoding/



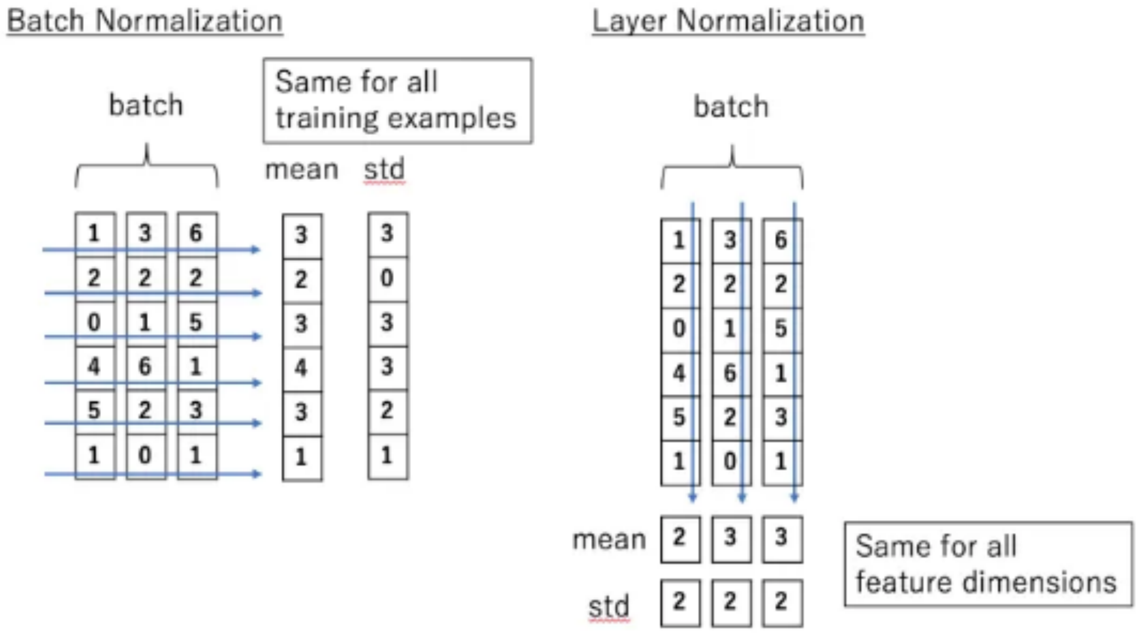
Add and Normalize

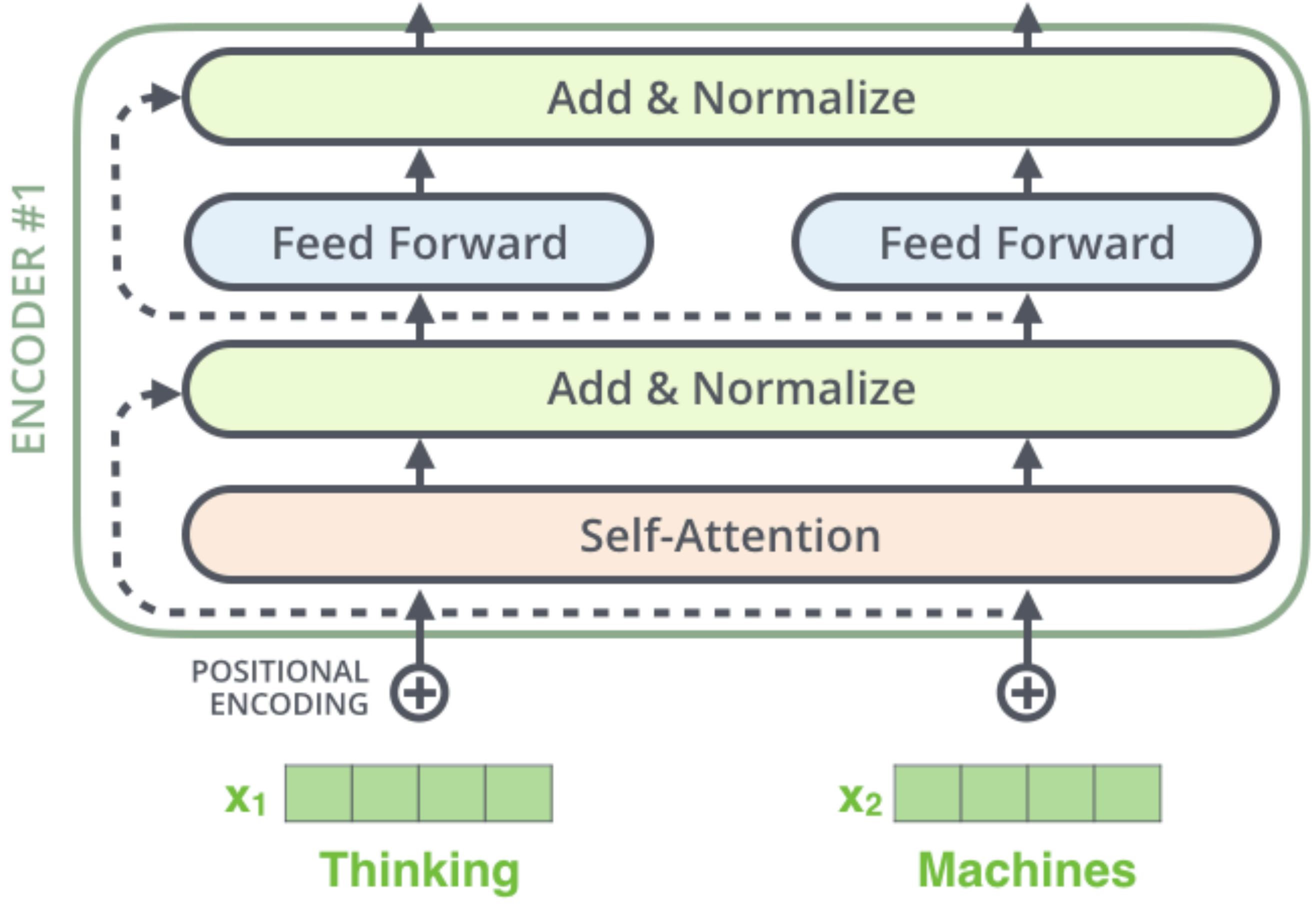
In order to regulate the computation, this is a normalization layer so that each feature (column) have the same average and deviation.



Layer Normalization (Hinton)

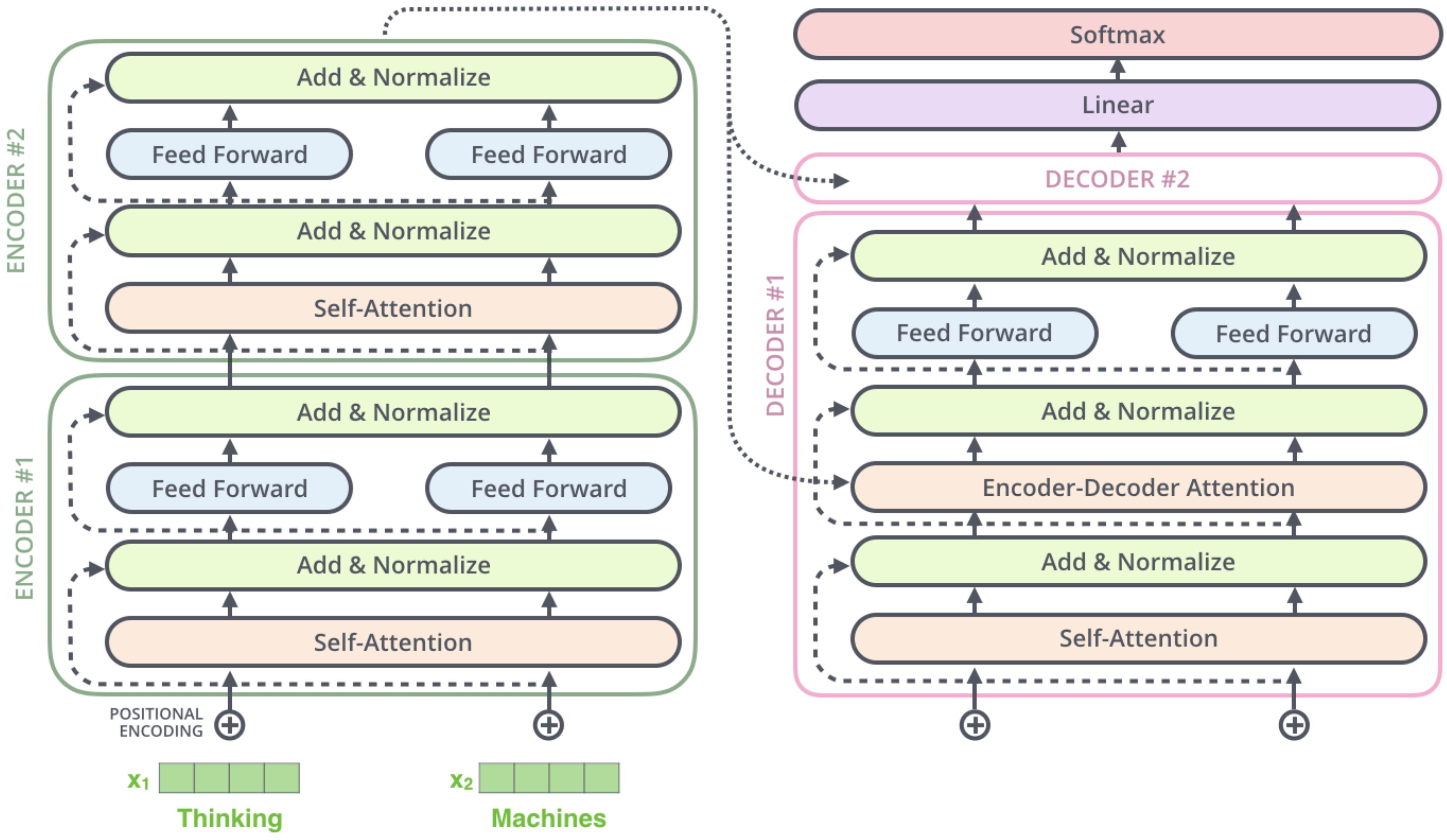
Layer normalization normalizes the inputs across the features.



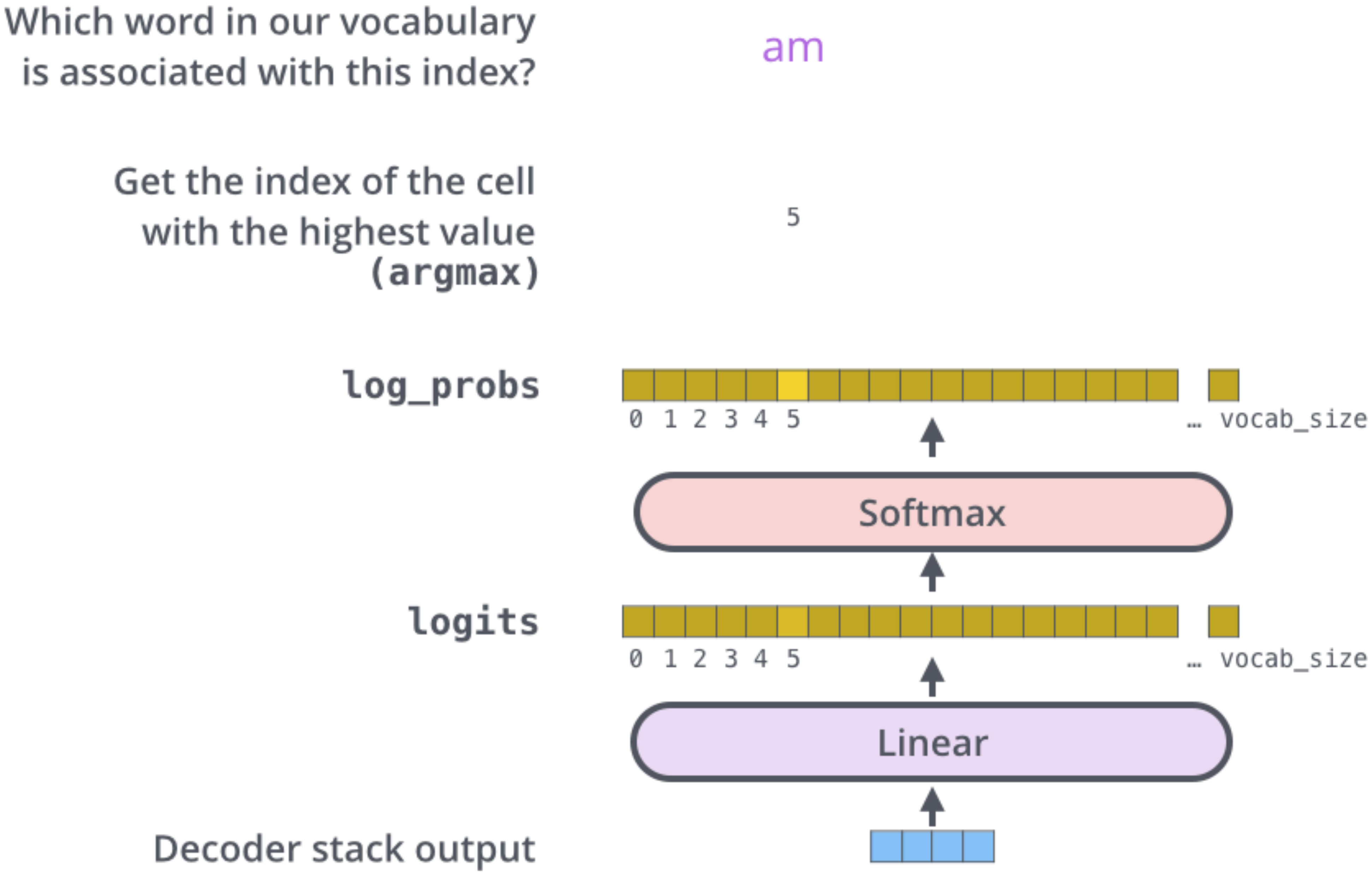


The complete transformer

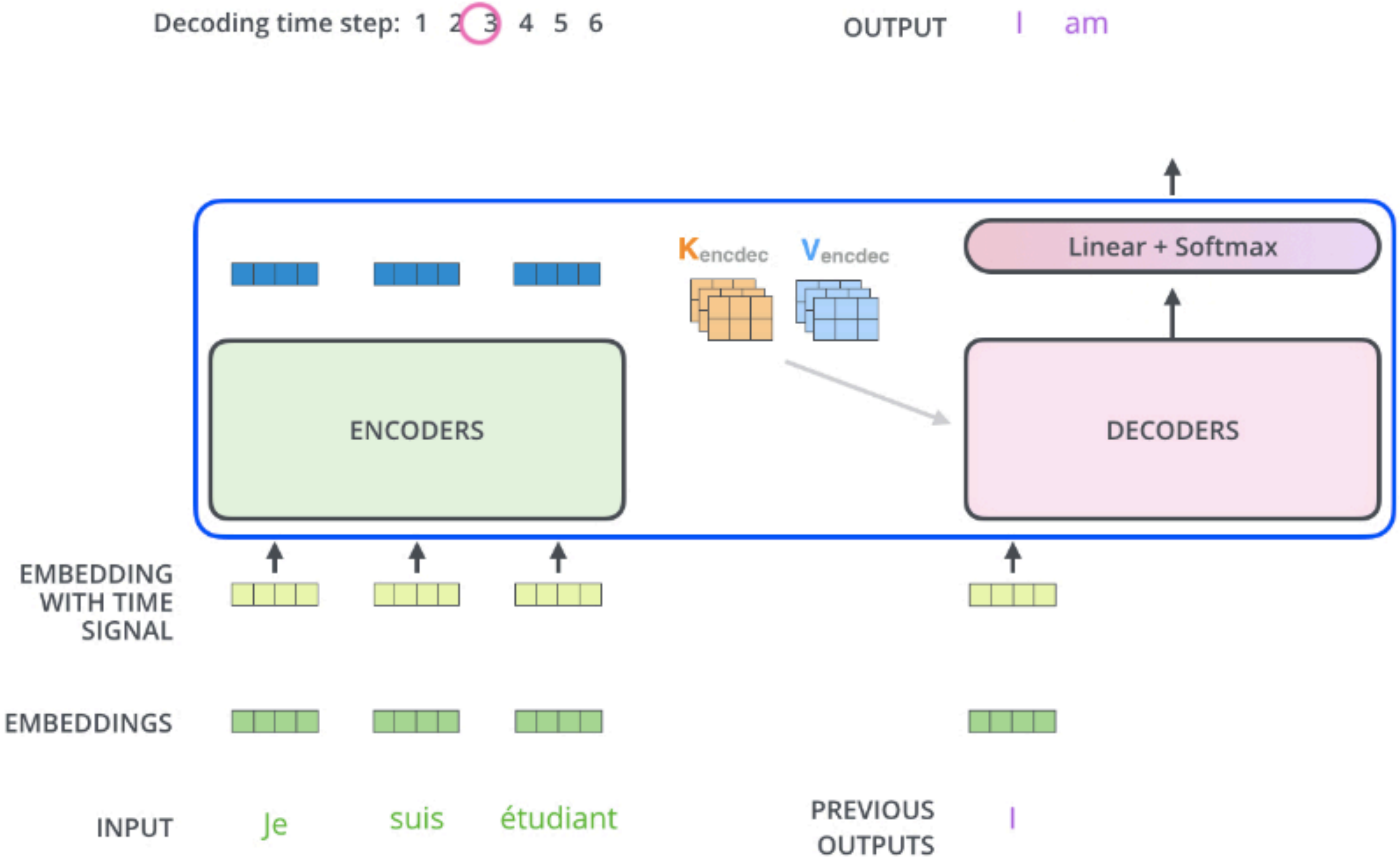
The encoder-decoder attention is just like self attention, except it uses K, V from the top of encoder output, and its own Q



Decoder's Output Linear Layer



How it works



But what about Self-attention?

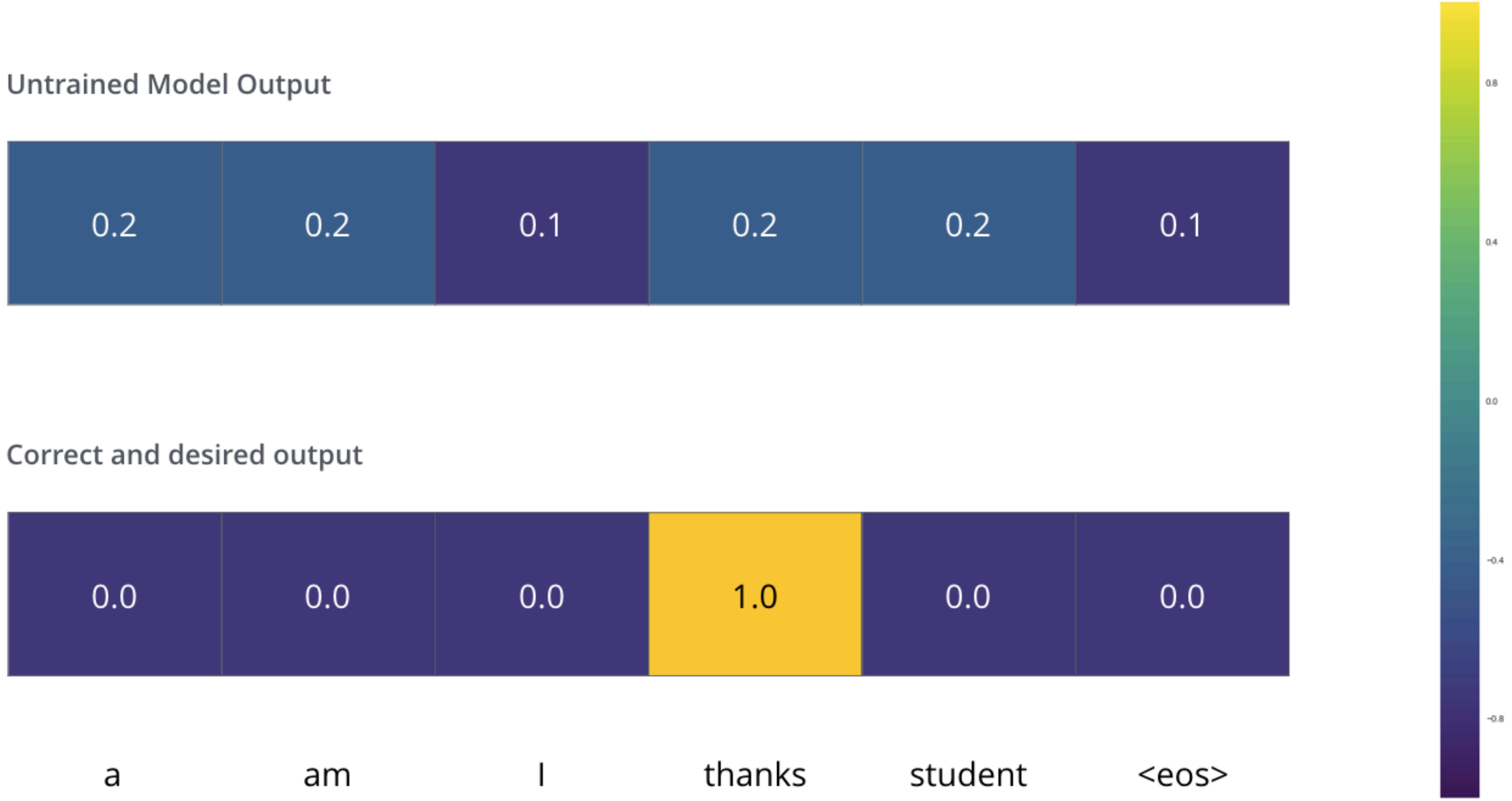
Attention and Transformers

But what about self-attention when the input is “incomplete”?

The solution is to set future unknown values with “-inf”.

The same for Encoder-Decoder Attention.

Training and the Loss Function



We can use cross Entropy.

We can also optimize two words at a time: using BEAM search: keep a few alternatives for the first word.

Cross Entropy and KL (Kullback-Leibler) divergence

- Entropy: $E(P) = -\Sigma_i P(i)\log P(i)$ - expected code length (also optimal)
- Cross Entropy: $C(P) = -\Sigma_i P(i) \log Q(i)$ – expected coding length using optimal code for Q
- KL divergence:

  $D_{KL}(P \,||\, Q) = \Sigma_i P(i)\log[P(i)/Q(i)] = \Sigma_i P(i)[\log P(i) - \log Q(i)]$, extra bits

- JSD$(P||Q) = ½ D_{KL}(P||M) + ½ D_{KL}(Q||M)$, $M = ½ (P+Q)$, symmetric KL

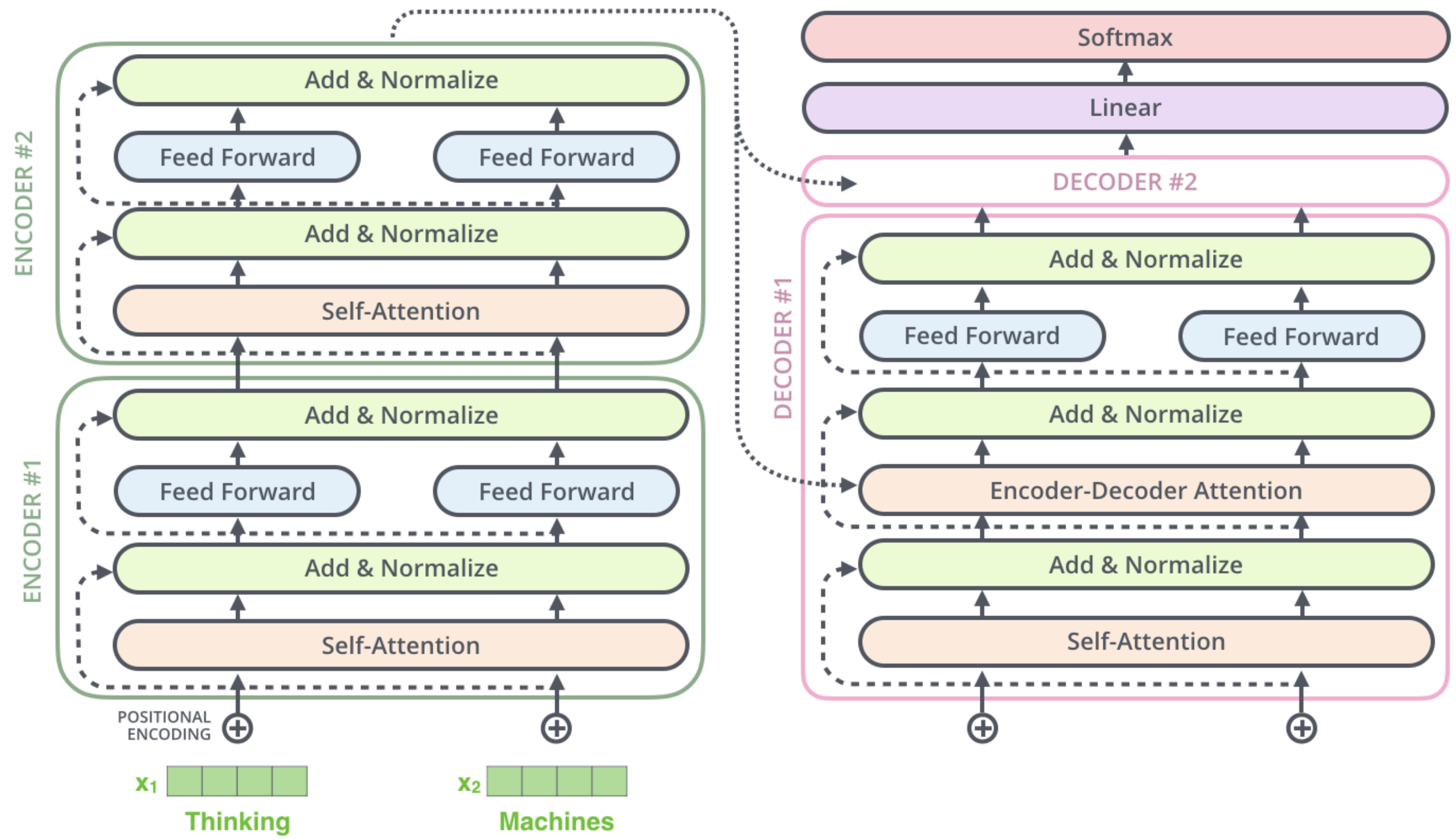
* JSD = Jensen-Shannon Divergency

Transformer Results

Table 2: The Transformer achieves better BLEU scores than previous state-of-the-art models on the English-to-German and English-to-French newstest2014 tests at a fraction of the training cost.

Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [18]	23.75			
Deep-Att + PosUnk [39]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [38]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [9]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [32]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [39]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [38]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [9]	26.36	**41.29**	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	$\mathbf{3.3 \cdot 10^{18}}$	
Transformer (big)	**28.4**	**41.8**	$2.3 \cdot 10^{19}$	

Next Lecture



BERT

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 (from which I borrowed many pictures): http://jalammar.github.io/illustrated-transformer/

Kate Logninova: Attention in NLP, summarizes all sorts of attentions. Can somebody present this and related literature? https://medium.com/@joealato/attention-in-nlp-734c6fa9d983