Transformers & BERT

↓

Bidirectional encoder representation from Transformers.

Assumption : Studied encoder-decoder models

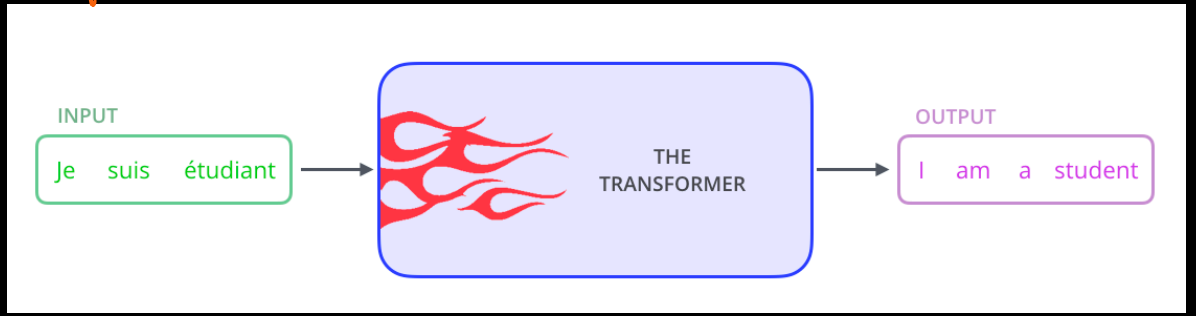
Attention

LSTMs

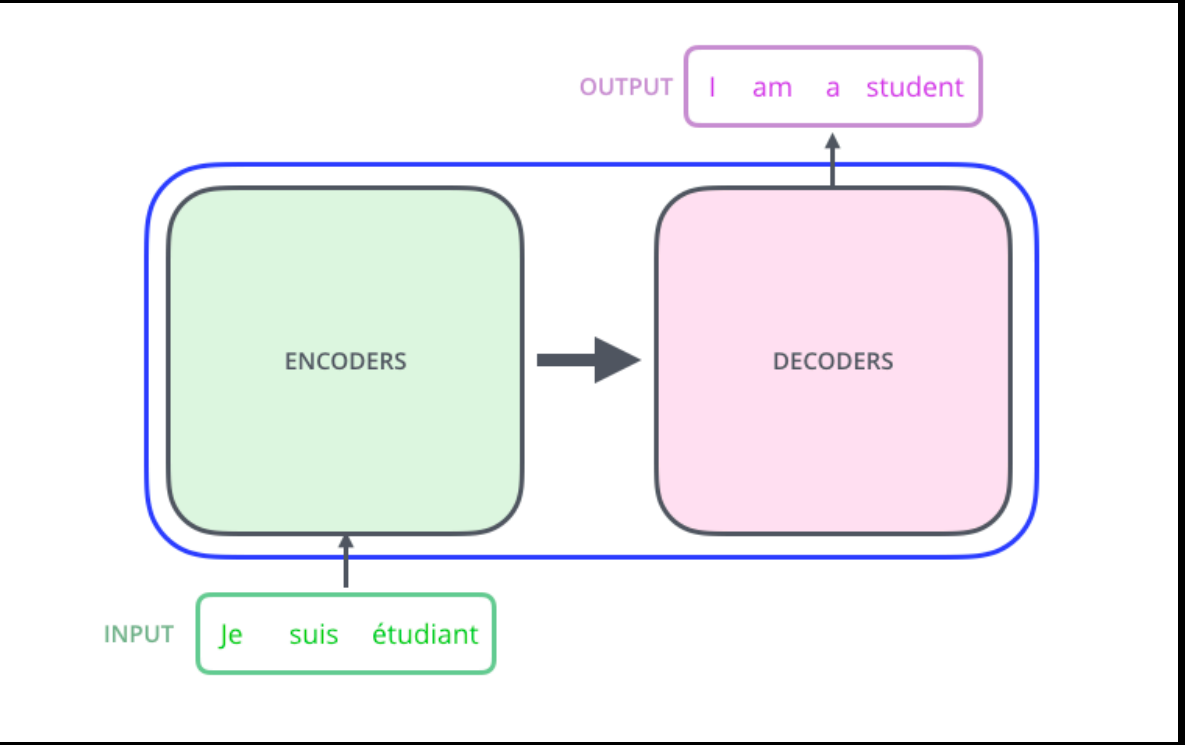
CNNs

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

Transformers: top-down approach

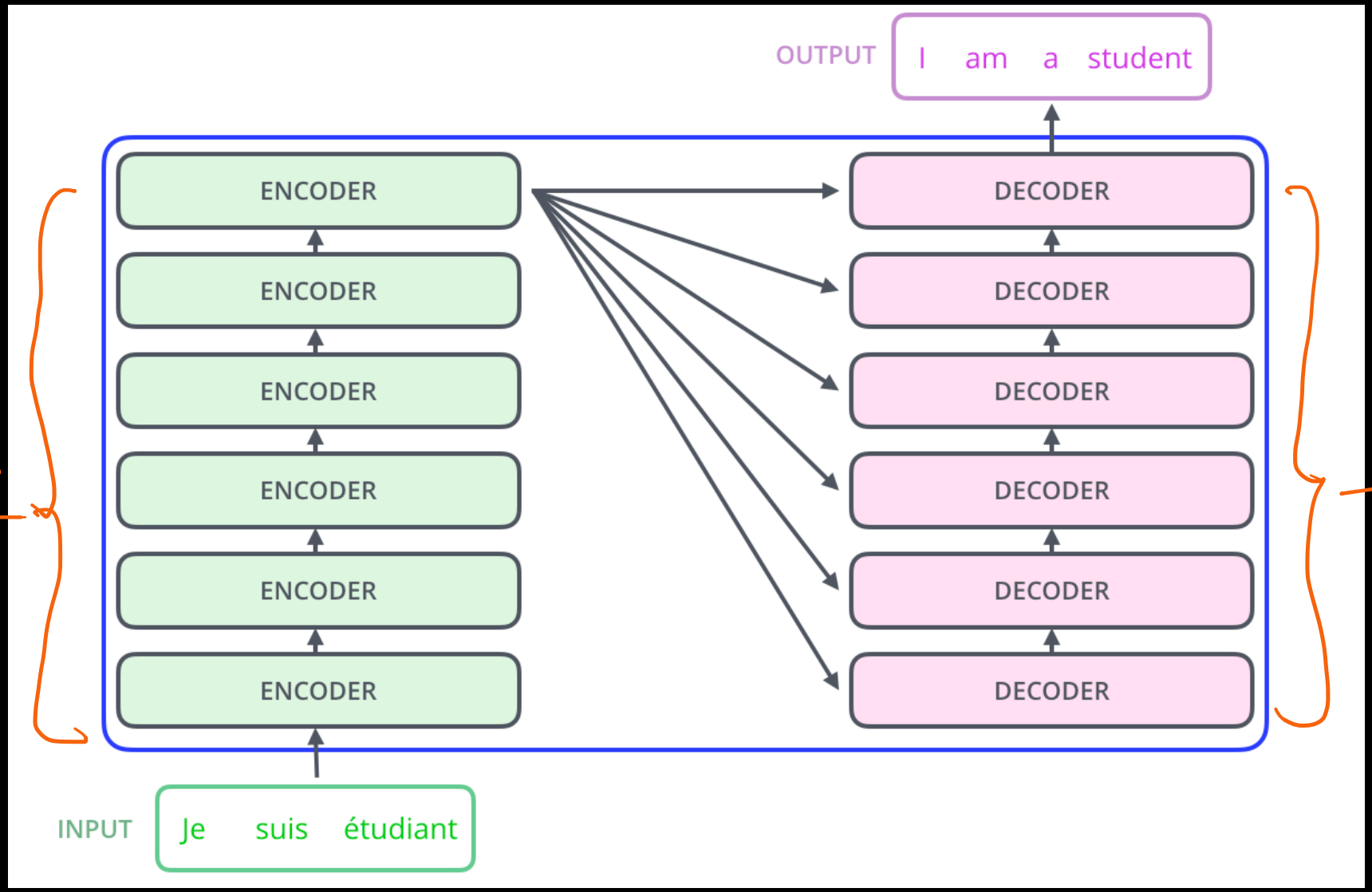


Seq-Seq model



encoder-decoder

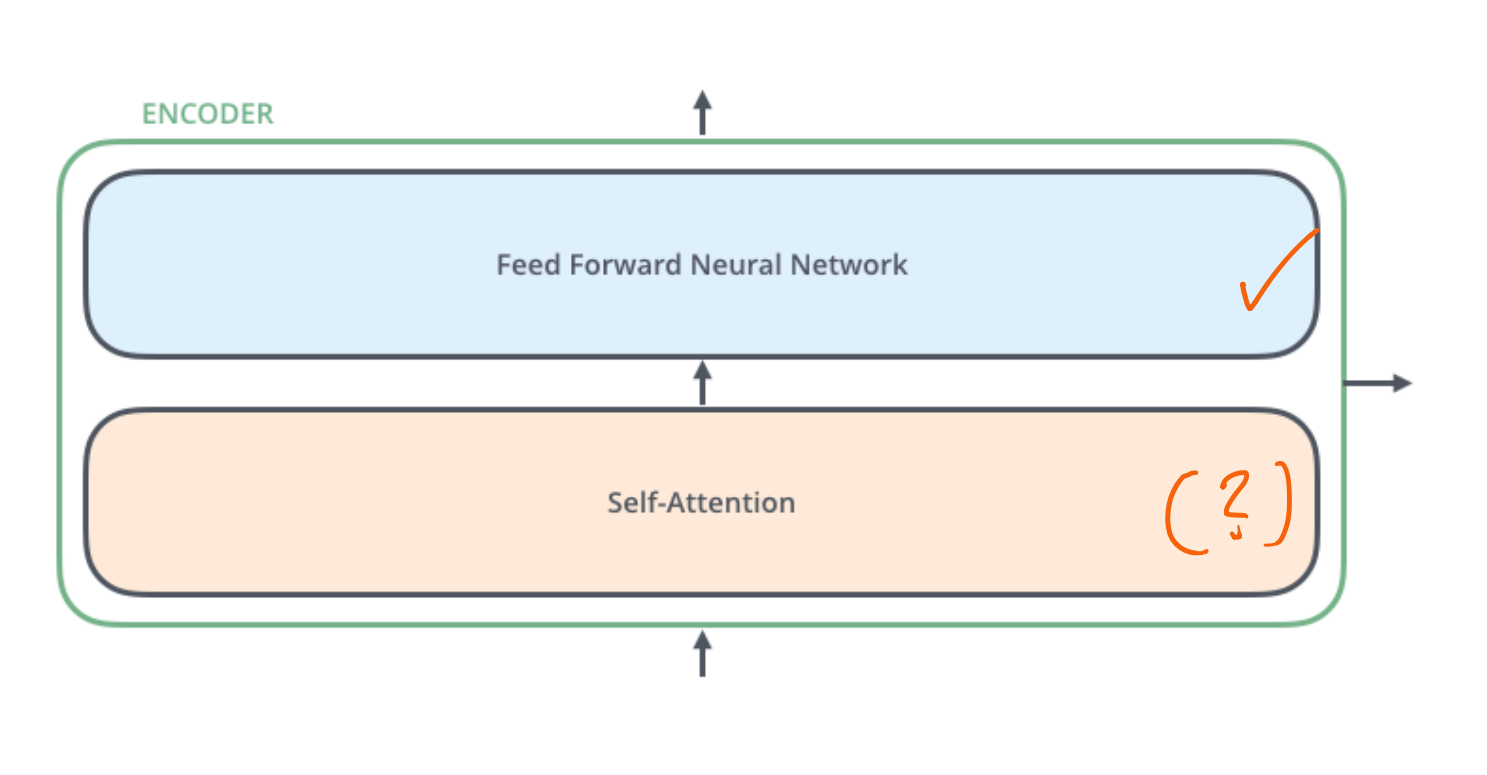
stack of 6 encoders & decoders



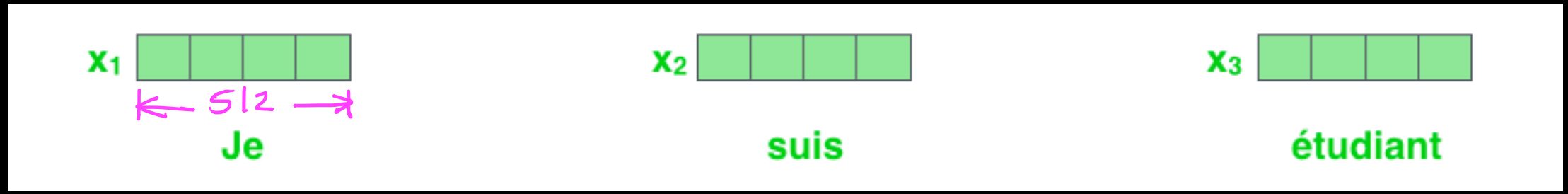
identical

identical

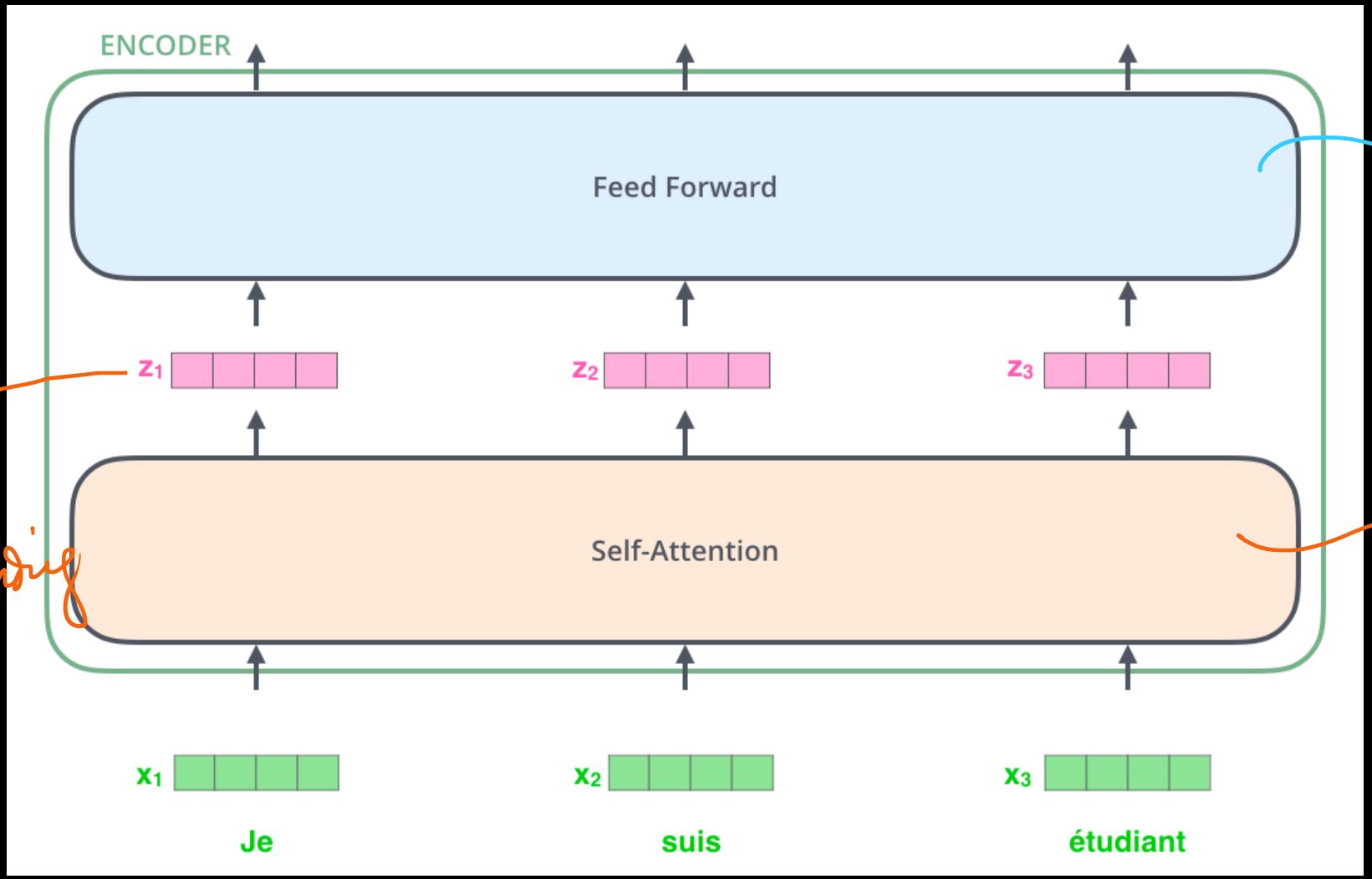
Structure of an encoder:



input : word-vectors/tensors



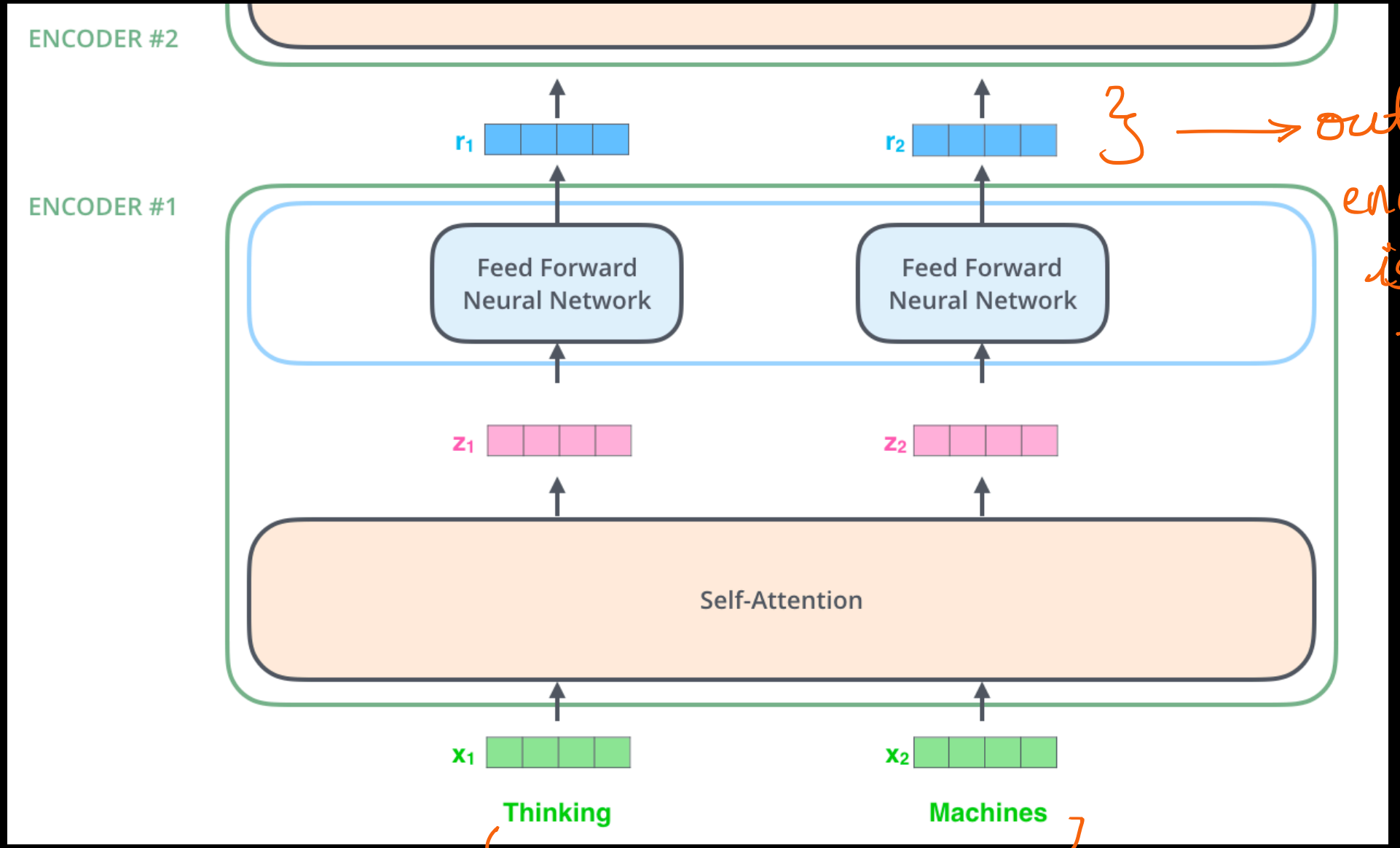
word2vec



512 units

output corresponding to x_1

we will see later how this works



z → output from encoder 1 is input to encoder 2.

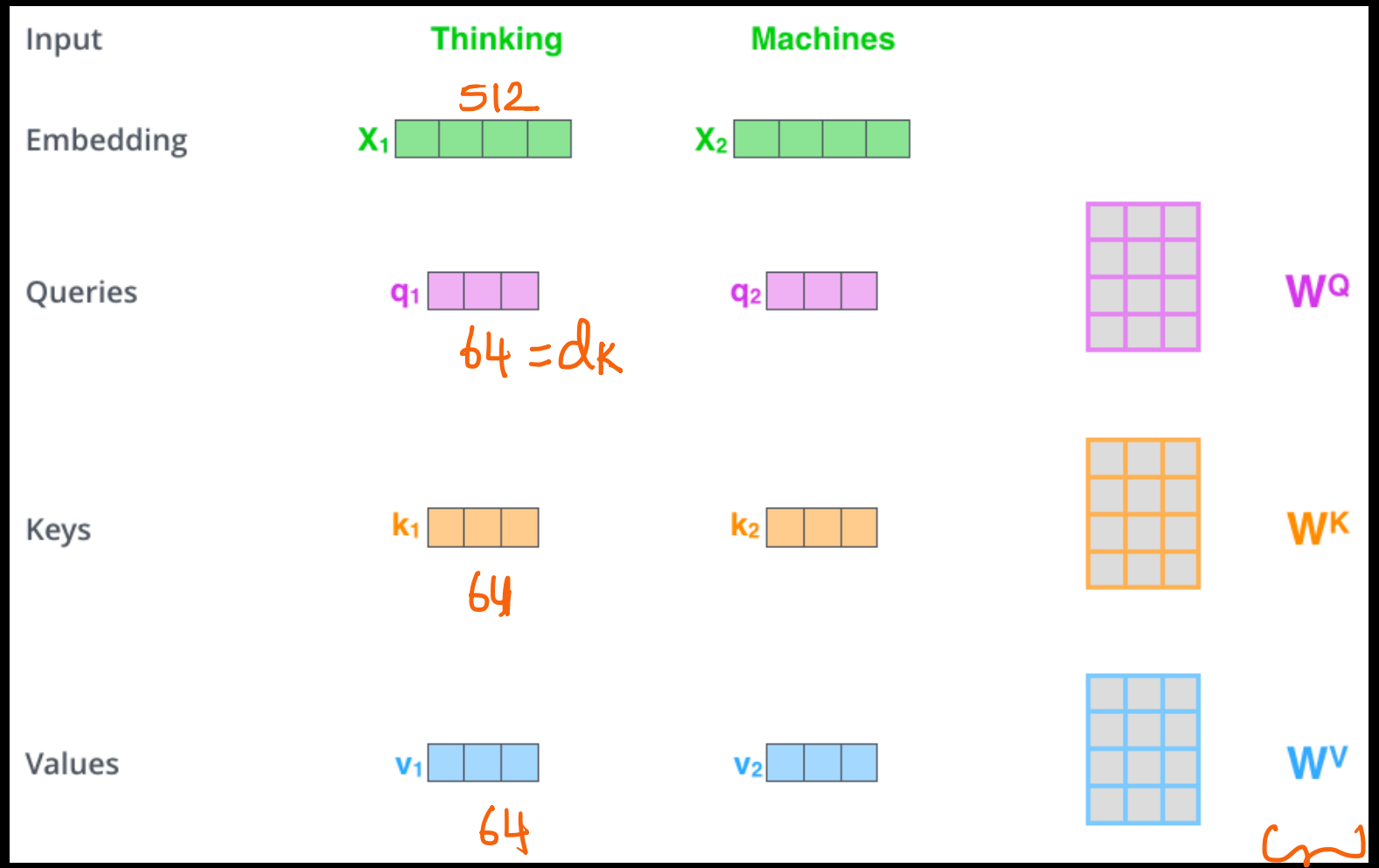
smaller input as an example.

Self-attention:

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

↳ animal or street ?

1



$q_i = X_i W^Q$

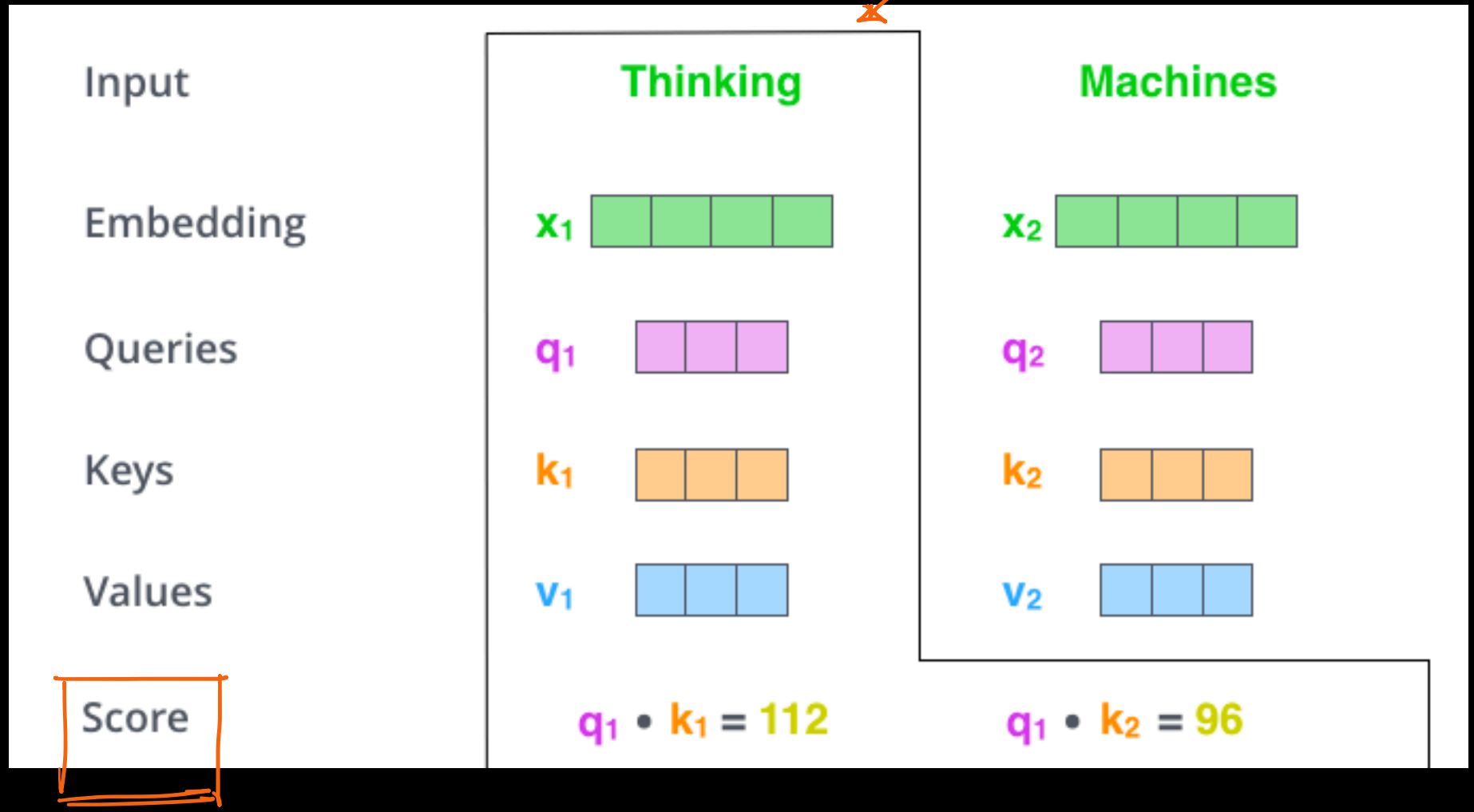
$k_i = X_i W^K$

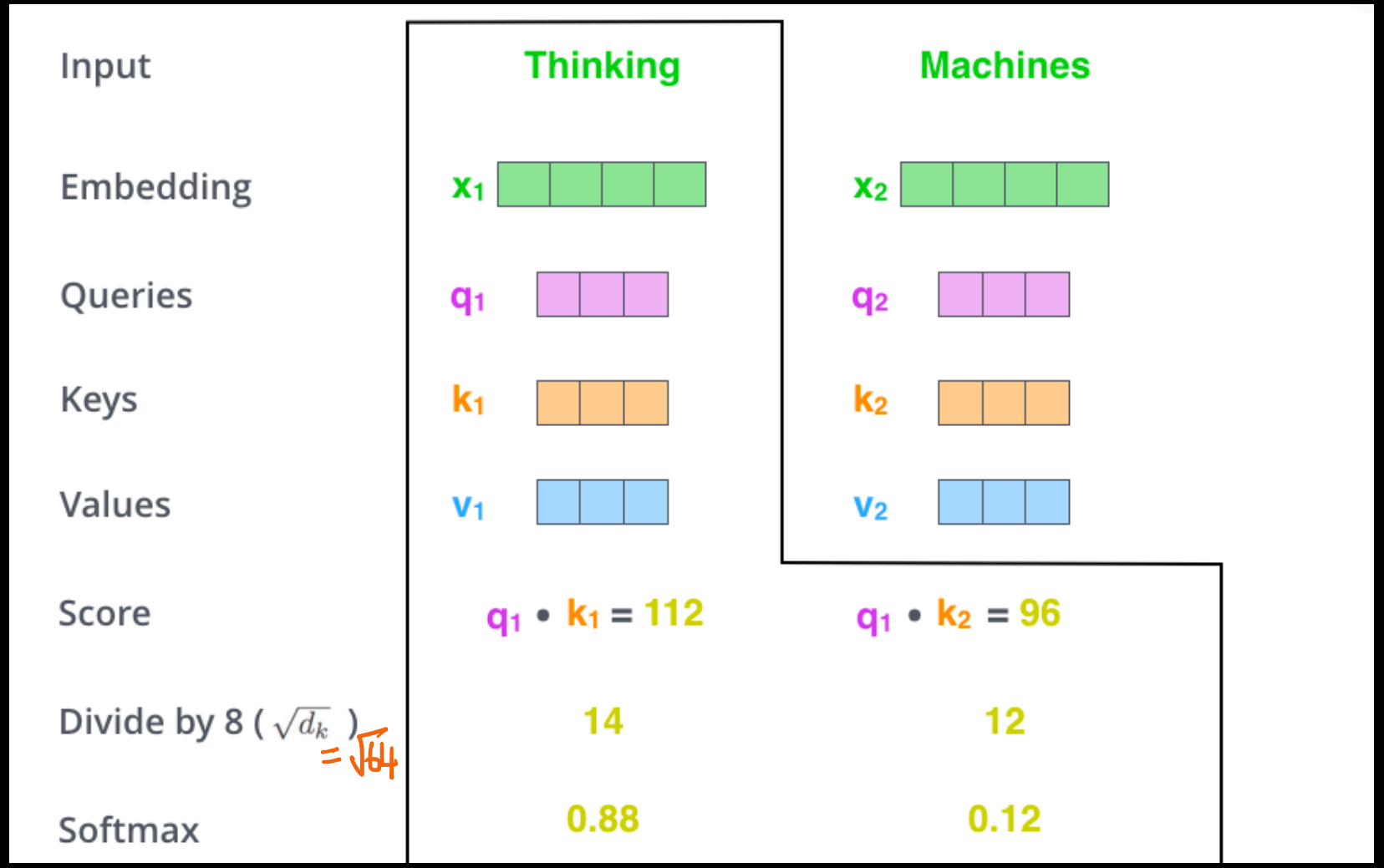
$v_i = X_i W^V$

Trainable matrices.

APPLIED COURSE

2 computing self-attention for 1st word (z_1) to generate (z_1)





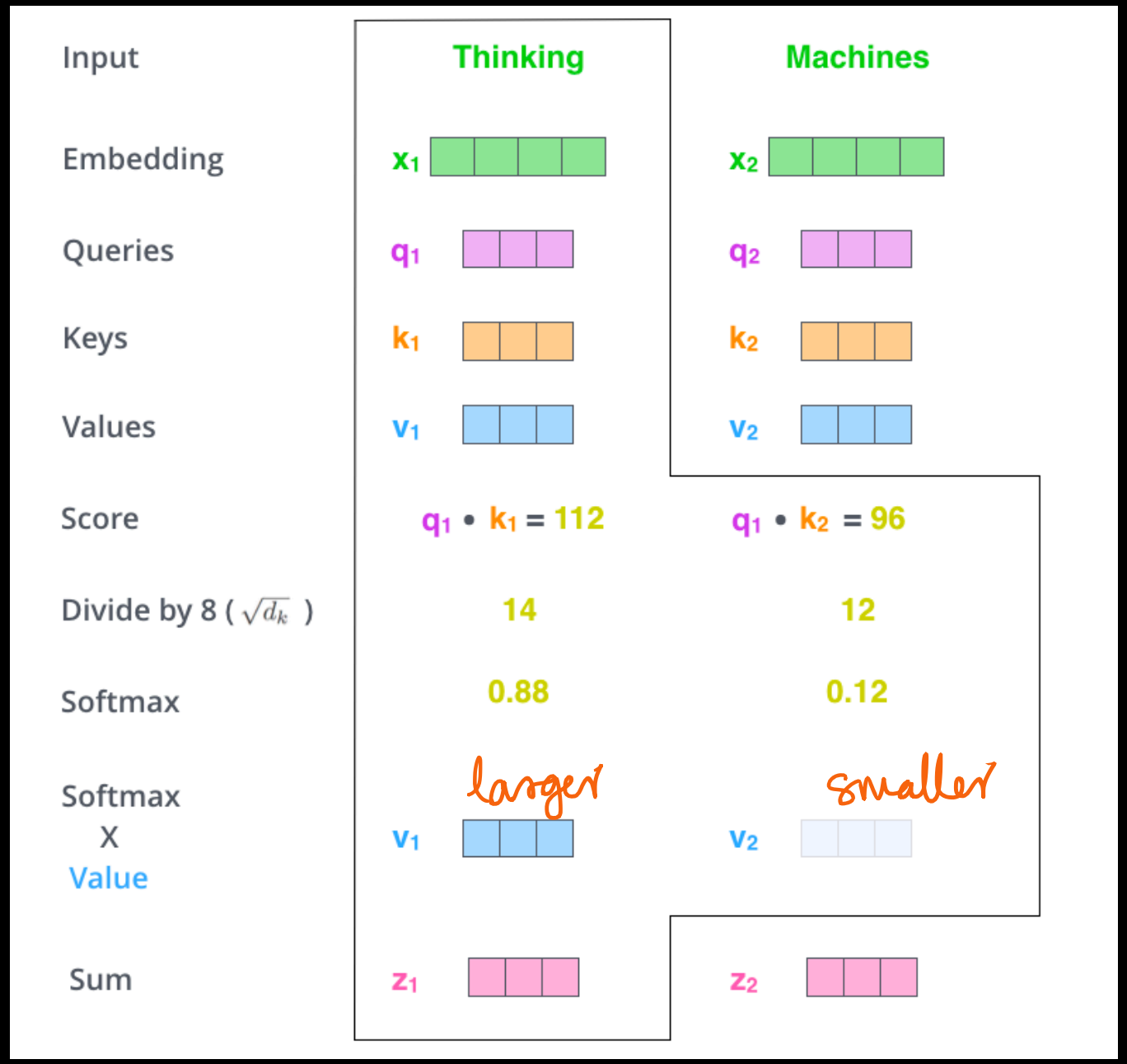
more stable gradients

3

4

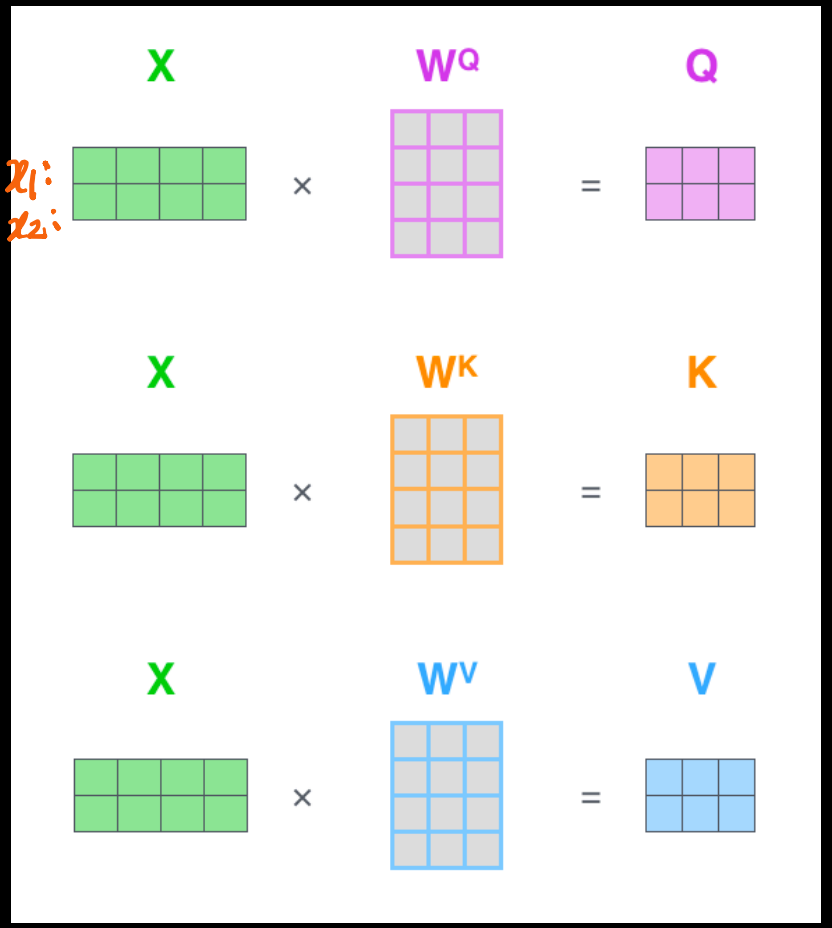
Sum = 1

APPLIED COURSE

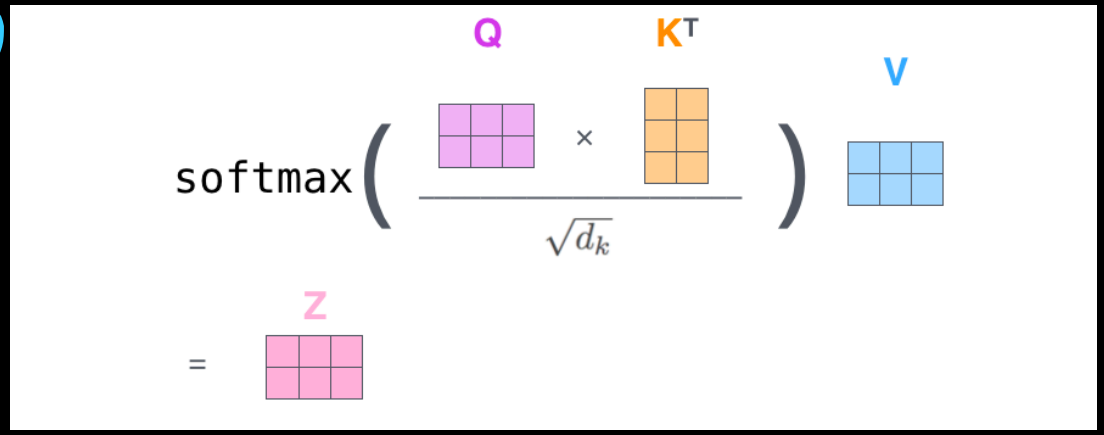


Matrix - computation :

1

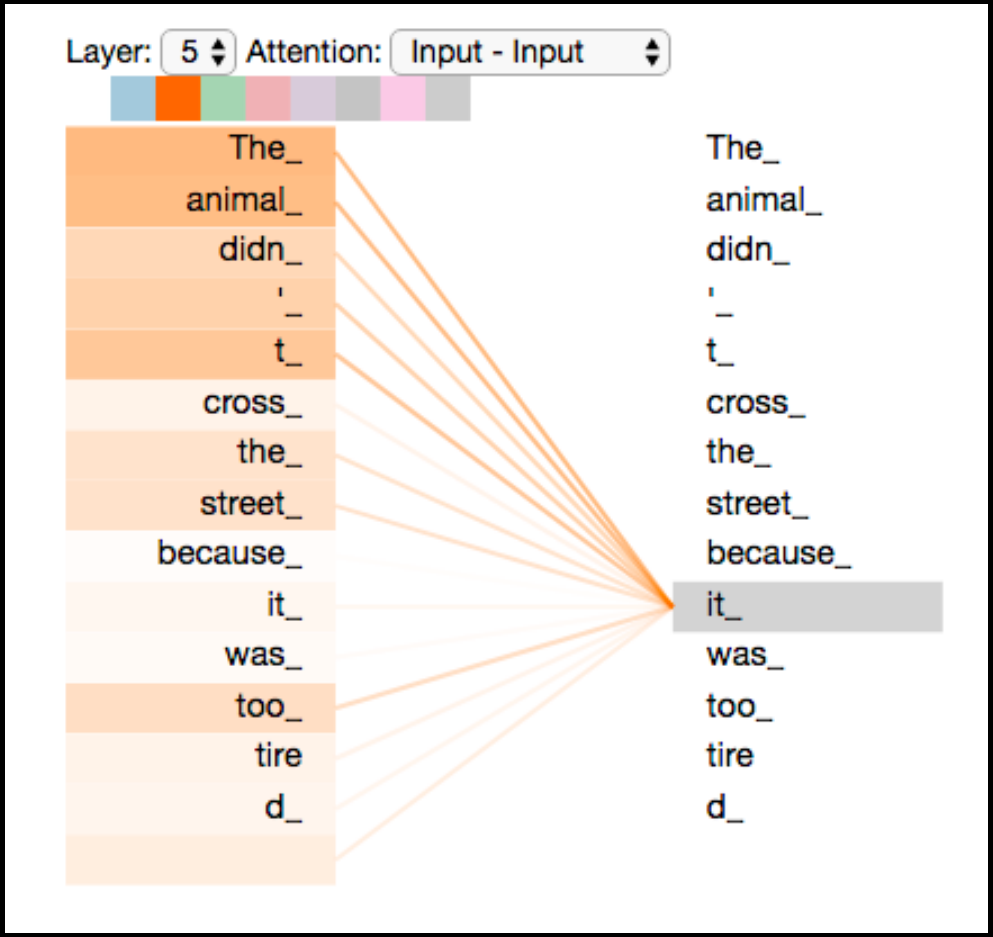


2

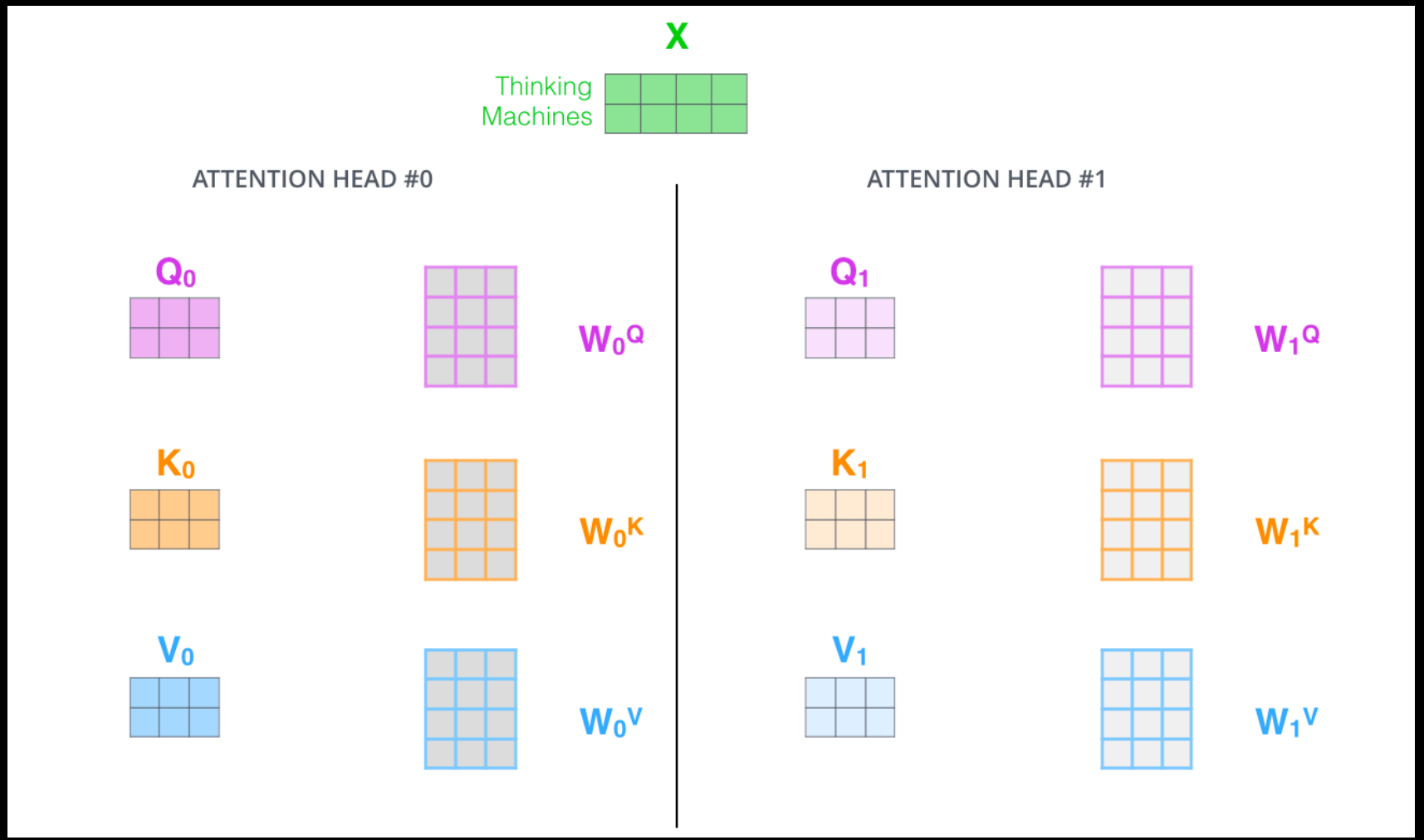


APPLIED COURSE

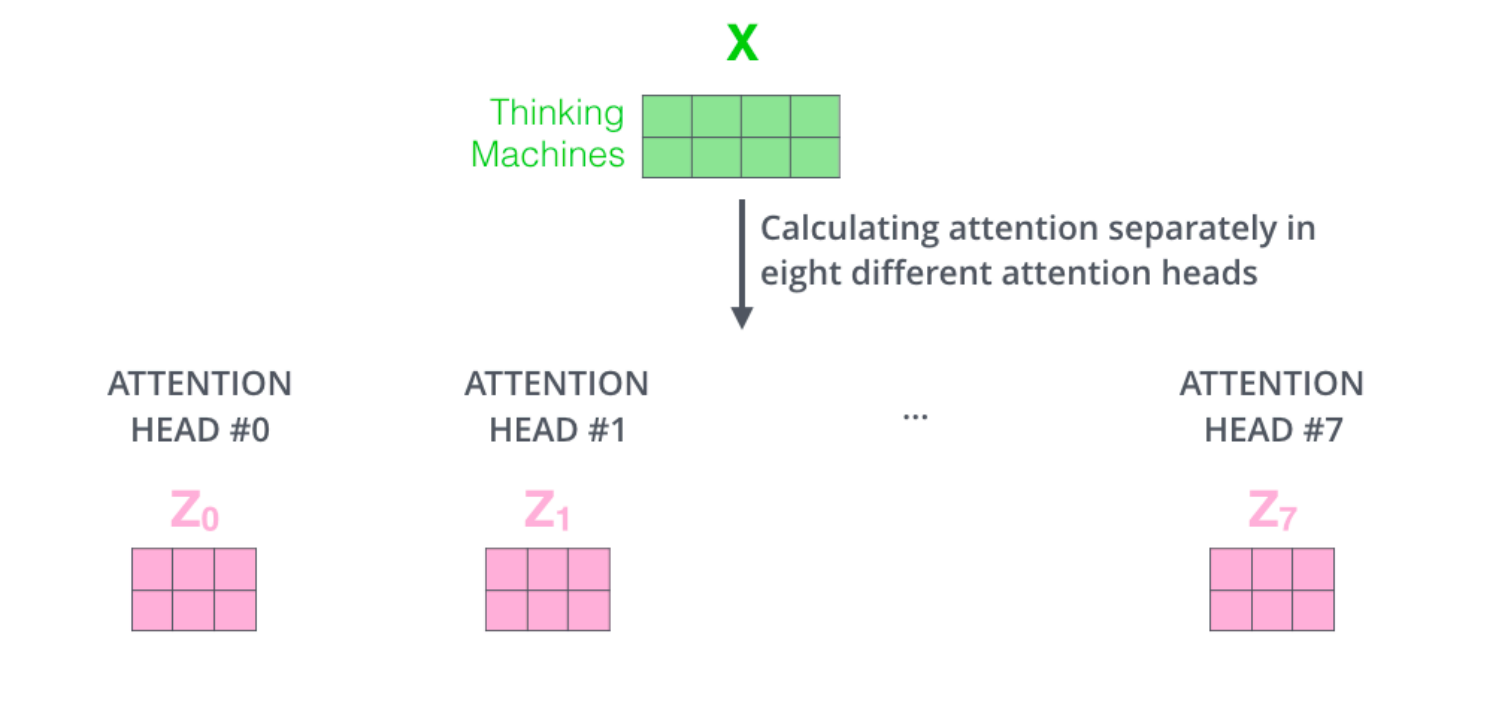
Attention helps us to focus on some of the words in the vicinity which matter



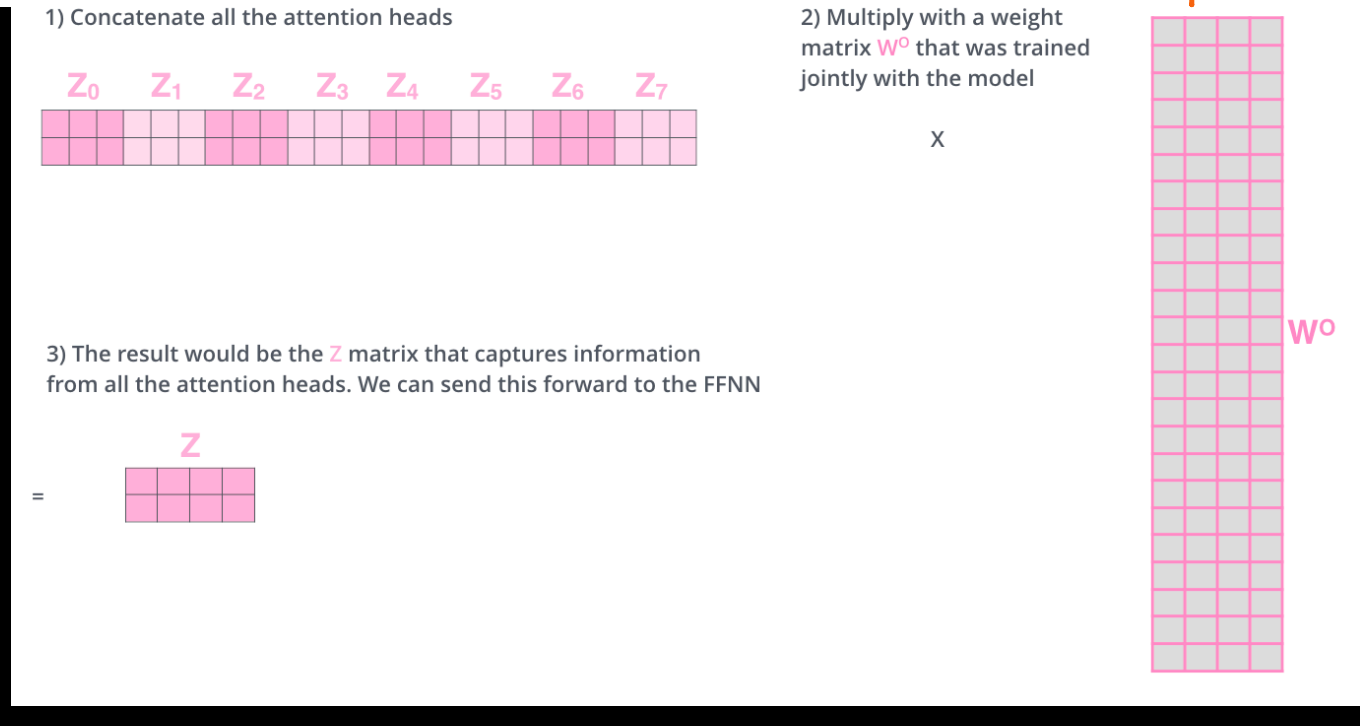
Multi-headed Attention: 8-headed (increases representational power)



single-headed z_1 could focus on 1st word itself more



another weight matrix



APPLIED COURSE

Summary:

1) This is our input sentence*

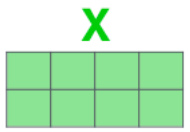
2) We embed each word*

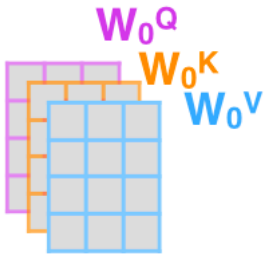
3) Split into 8 heads. We multiply X or R with weight matrices

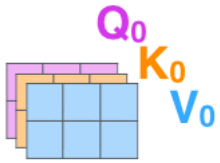
4) Calculate attention using the resulting Q/K/V matrices

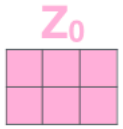
5) Concatenate the resulting Z matrices, then multiply with weight matrix W^O to produce the output of the layer

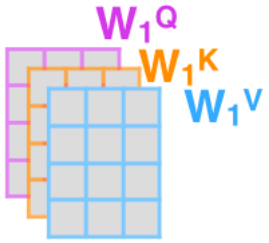
Thinking Machines

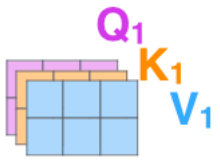


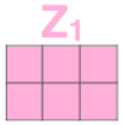




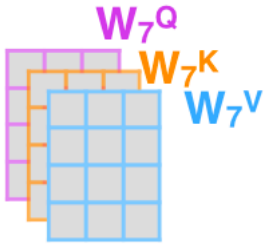


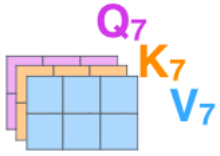


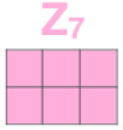


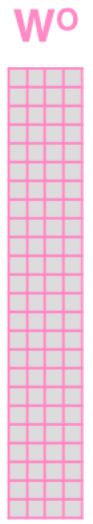


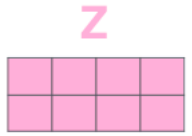
* In all encoders other than #0, we don't need embedding. We start directly with the output of the encoder right below this one







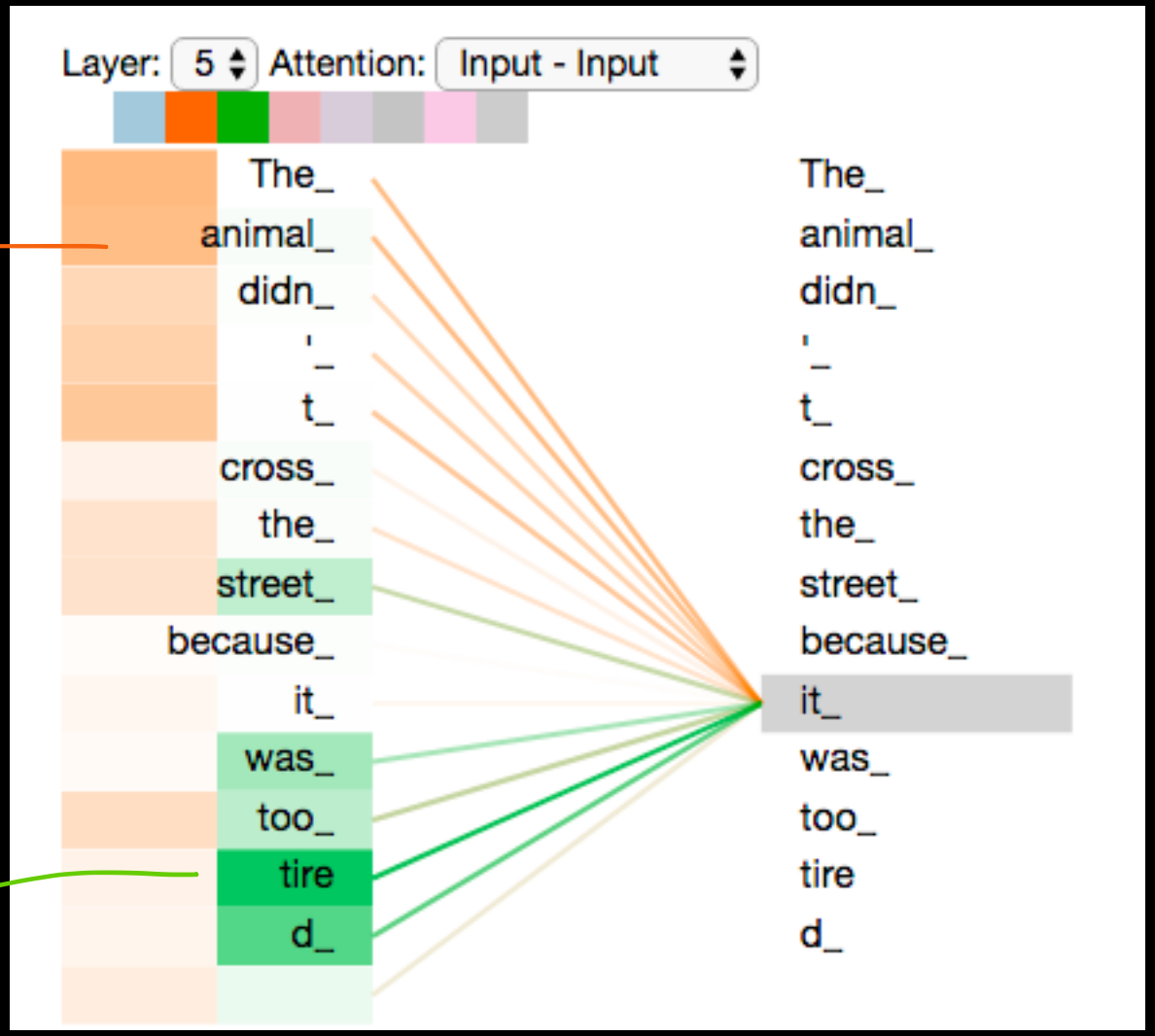




trainable

Trainable

APPLIED COURSE



1st head focus on 'animal'

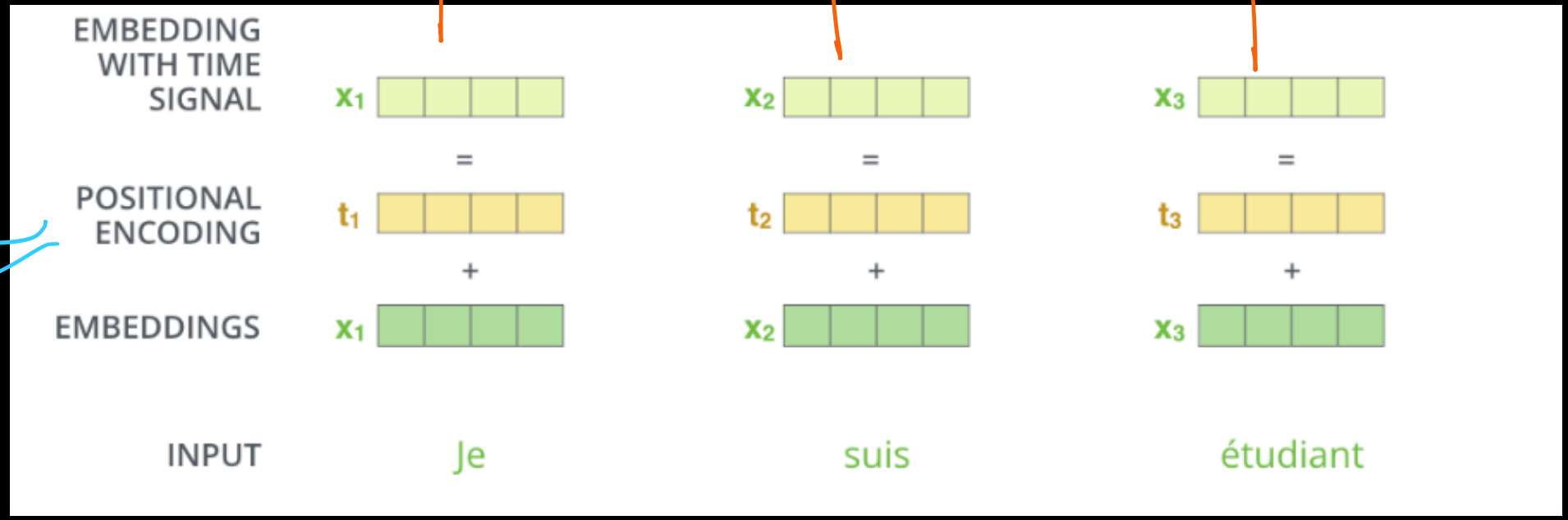
2nd head focus on 'tire'

Order of words:

ENCODER

position

distance b/w words



POSITIONAL ENCODING

0	0	1	1

+

EMBEDDINGS x_1

INPUT Je

0.84	0.0001	0.54	1

+

EMBEDDINGS x_2

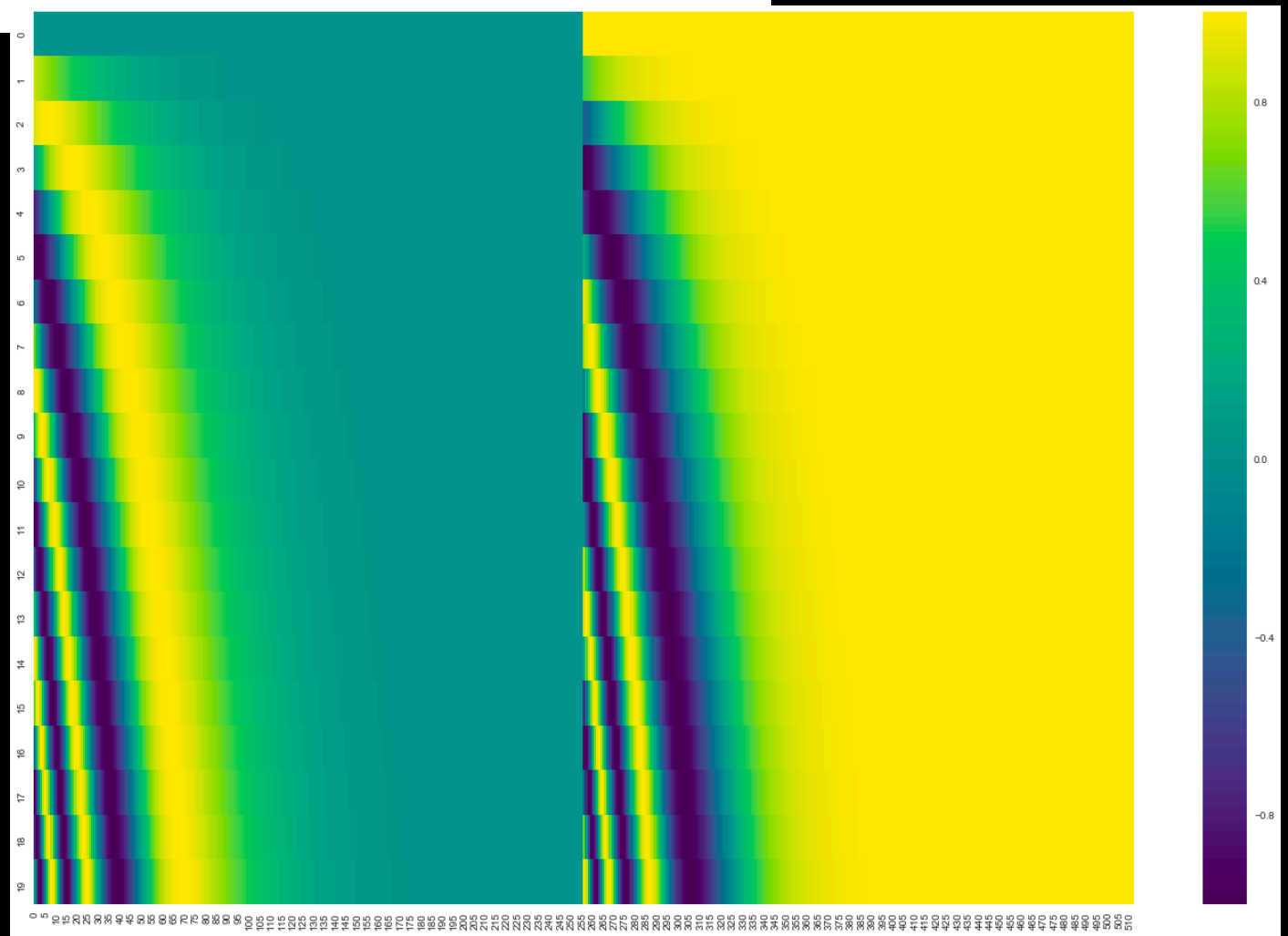
INPUT suis

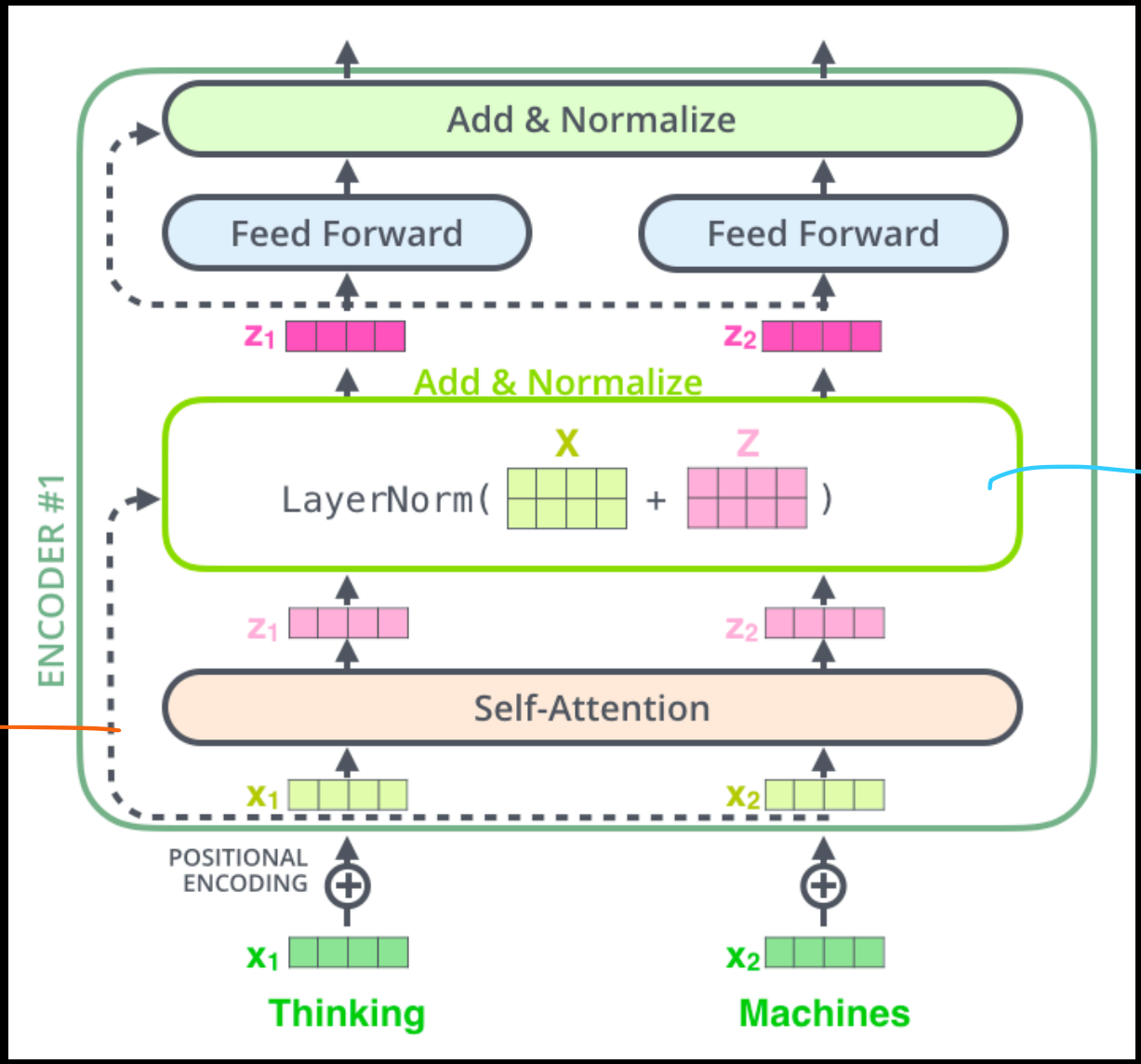
0.91	0.0002	-0.42	1

+

EMBEDDINGS x_3

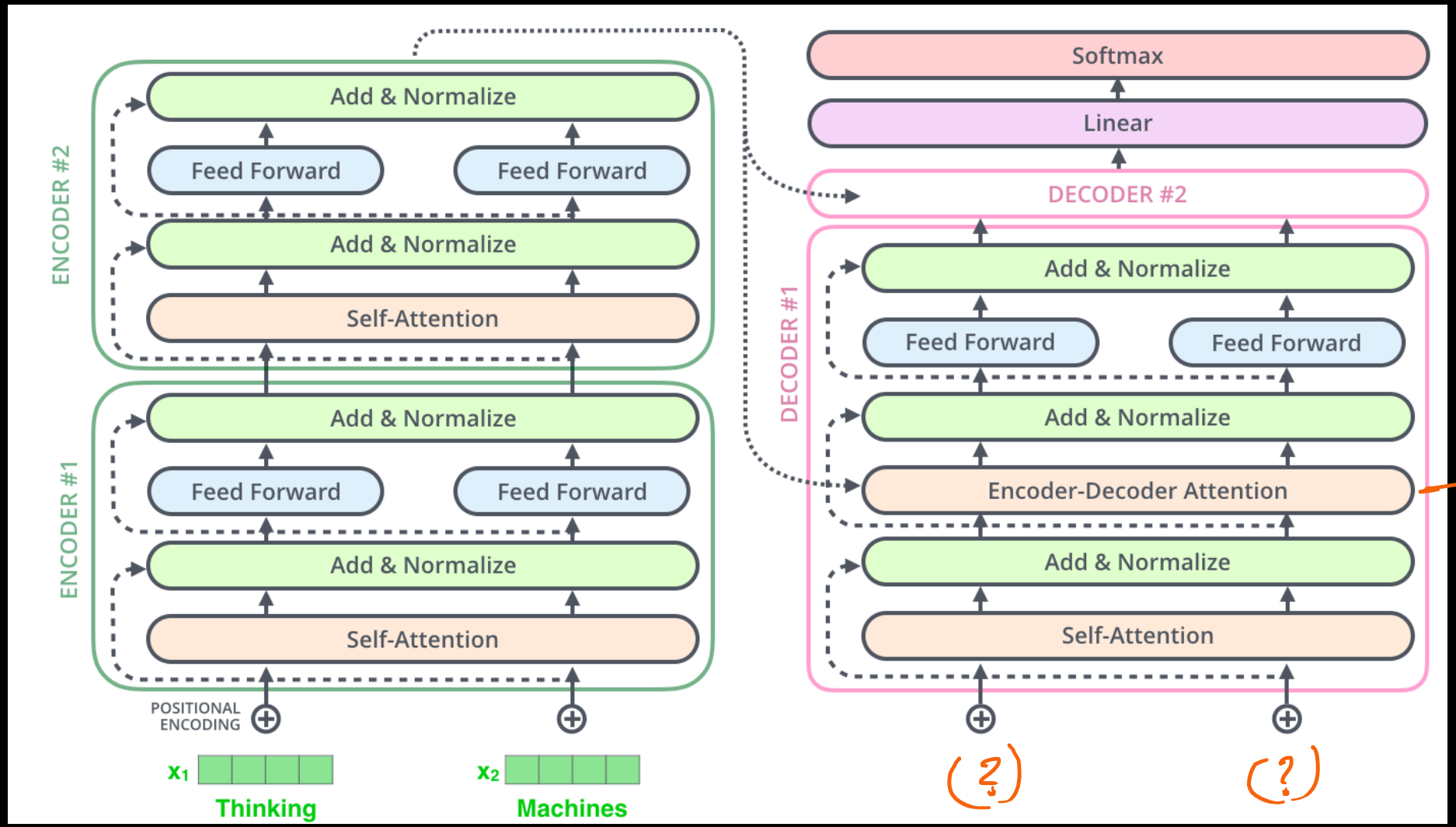
INPUT étudiant





Residual connection (like in ResNets)

Add & Normalize



Decoding time-steps:

1. http://jalammar.github.io/images/t/transformer_decoding_1.gif

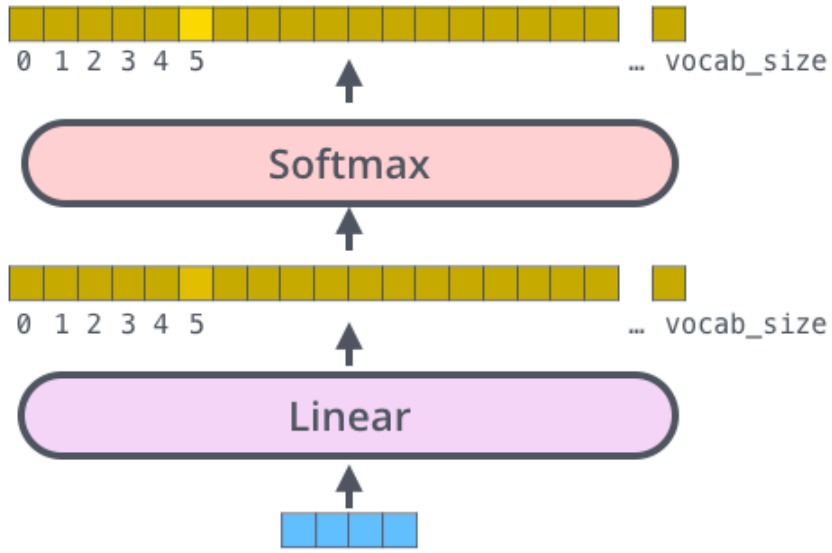
2. http://jalammar.github.io/images/t/transformer_decoding_2.gif

Which word in our vocabulary is associated with this index?

am

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

5



$\mathcal{L}$: Cross-entropy

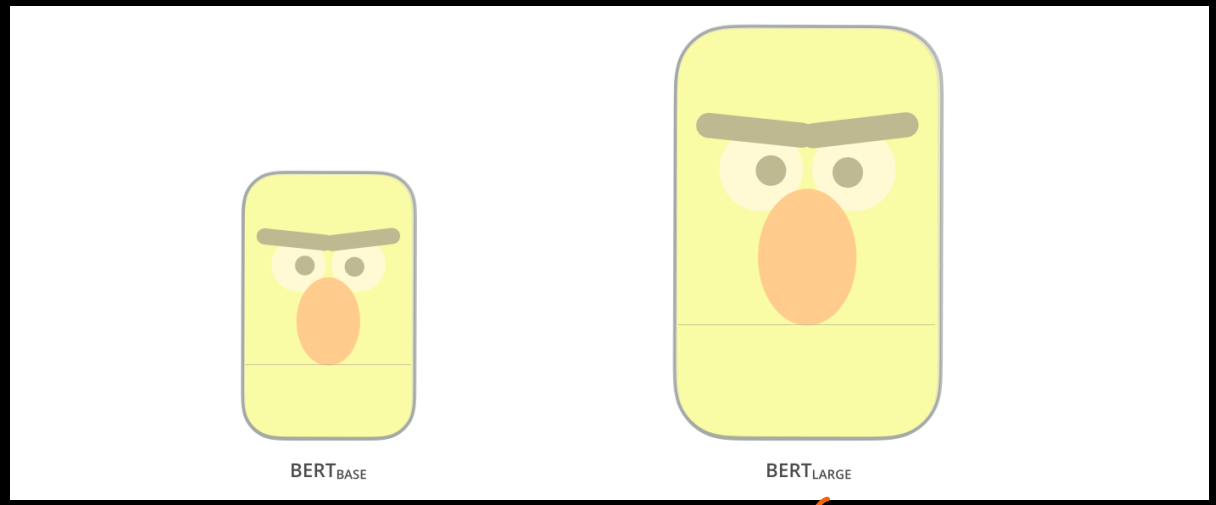
Key - take aways:

→ faster than RNN-based models as all input is ingested once

→ inspired from neighborhood - like concept in CNNs

→ state of the art.

BERT: bi-directional encoder representation from Transformers.



SQuAD1.1 Leaderboard

Rank	Model	EM	F1
	Human Performance *Stanford University* (Rajpurkar et al. '16)	82.304	91.221
1 Oct 05, 2018	BERT (ensemble) *Google AI Language* https://arxiv.org/abs/1810.04805	**87.433**	**93.160**
2 Sep 09, 2018	nlnet (ensemble) *Microsoft Research Asia*	85.356	91.202
3 Jul 11, 2018	QANet (ensemble) *Google Brain & CMU*	84.454	90.490

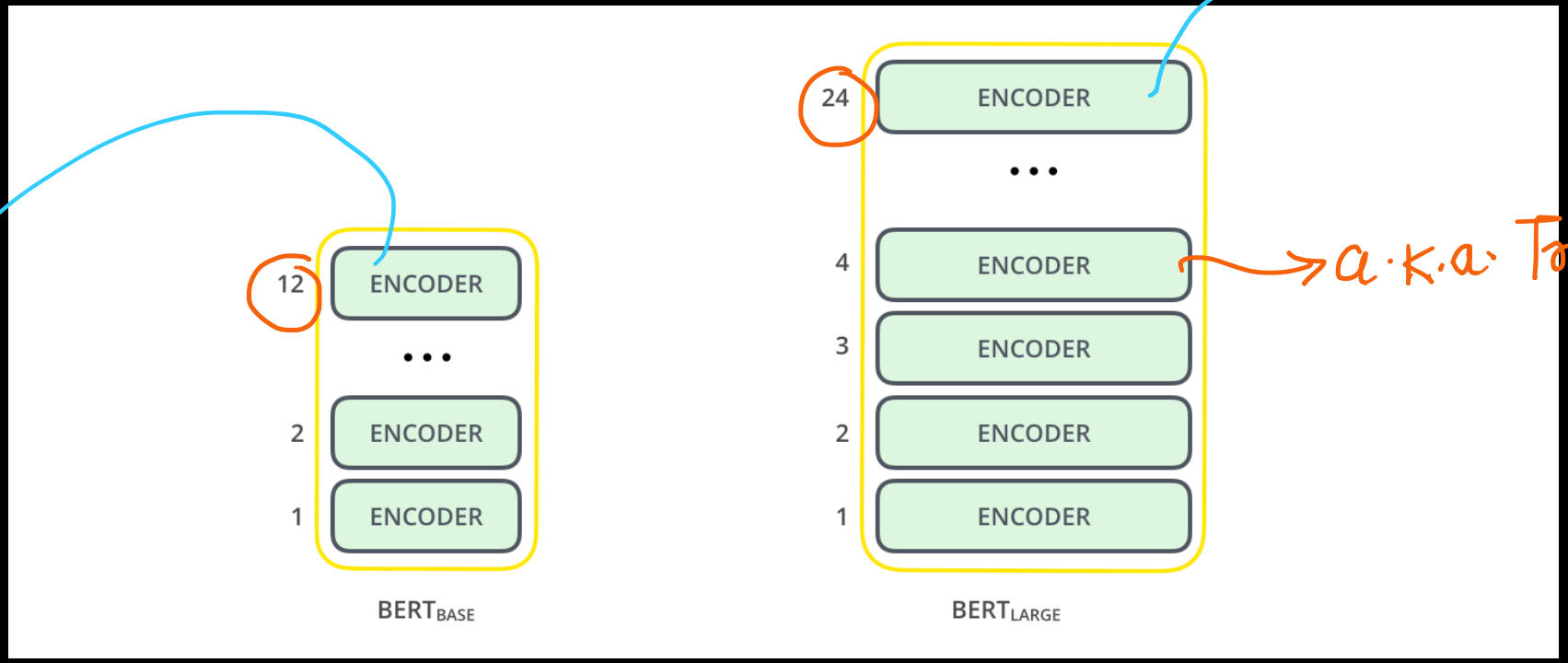
→ Transfer Learning in NLP.

→ Comparable to humans on multiple NLP tasks.

Source: http://jalammar.github.io/illustrated-bert/

APPLIED COURSE

FFN: 1024, 16 AH,



FFN: 768

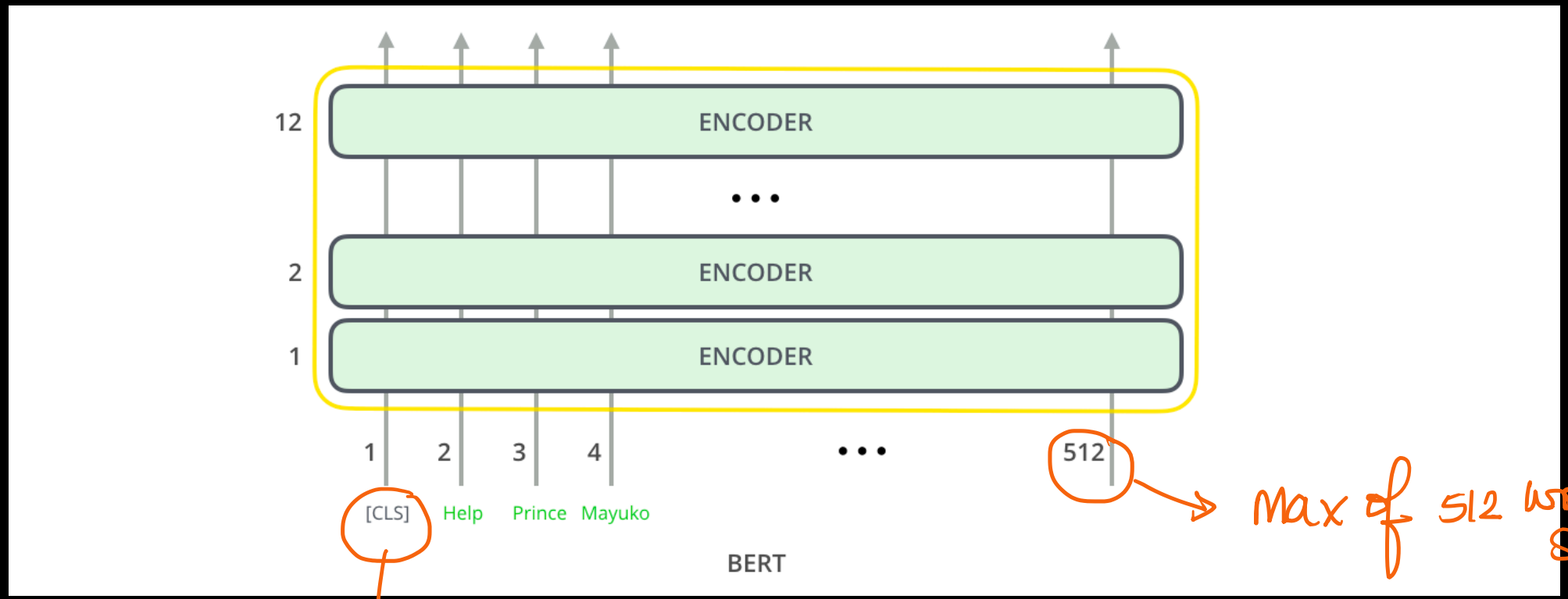
12 Attention heads

a.k.a. Transformer Blocks

Encoder-stack

NO-DECODERS

Spam or NOT (Transfer Learning)

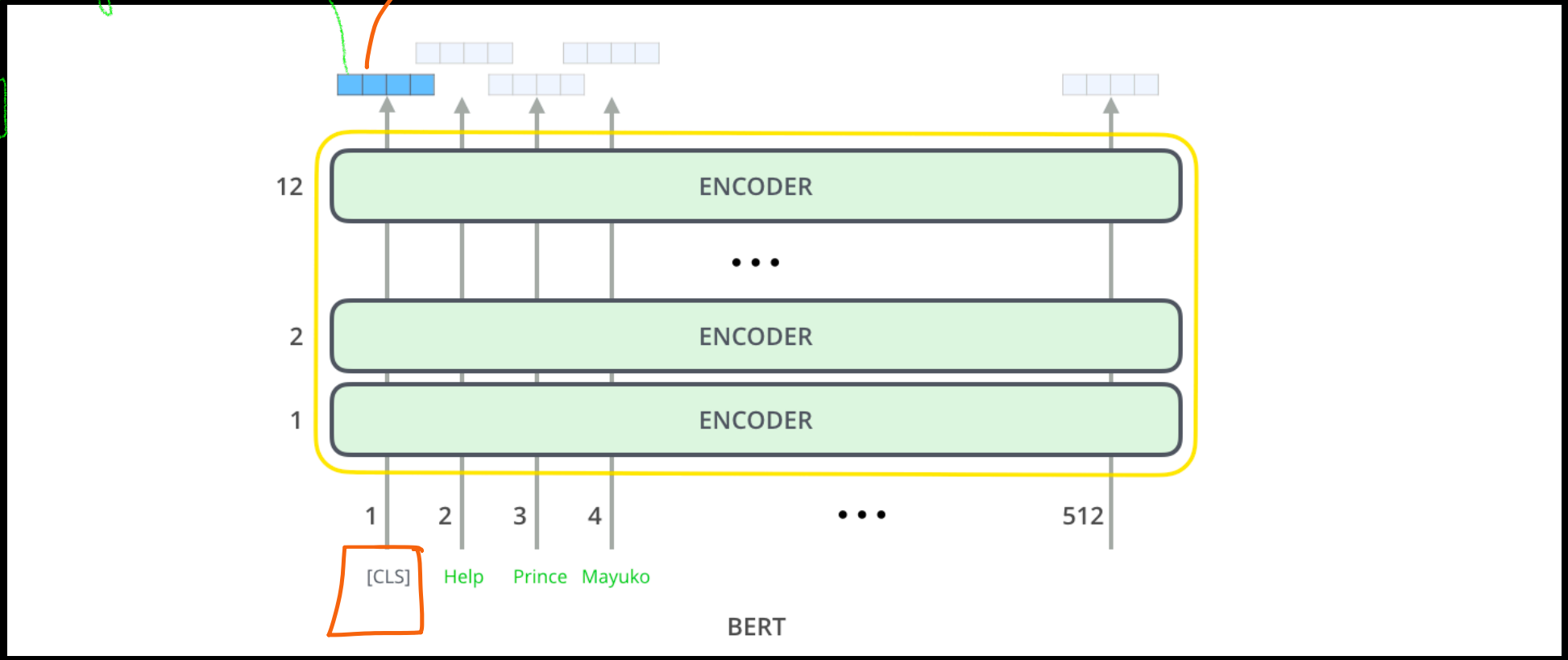


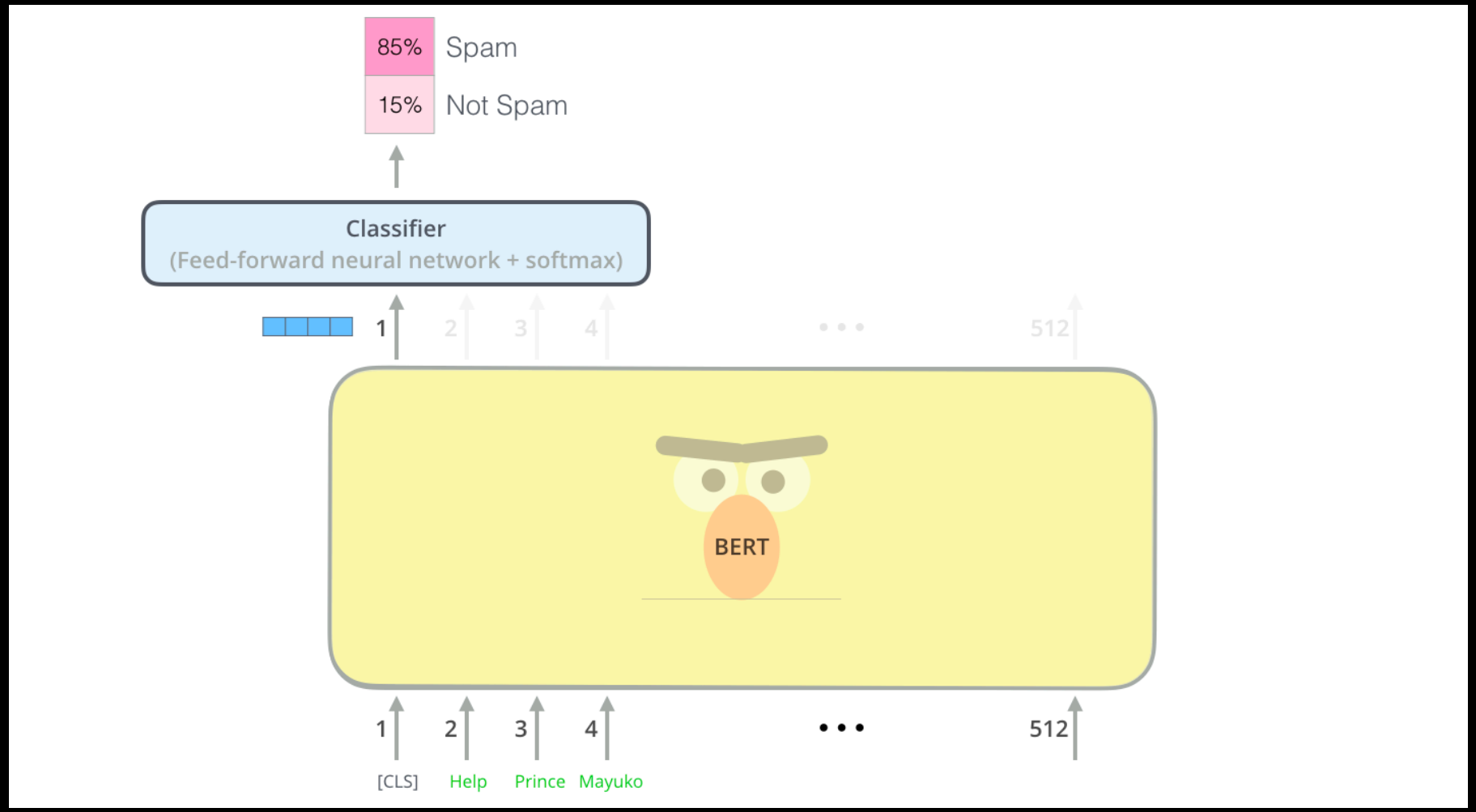
Max of 512 worded Sentences.

Classification

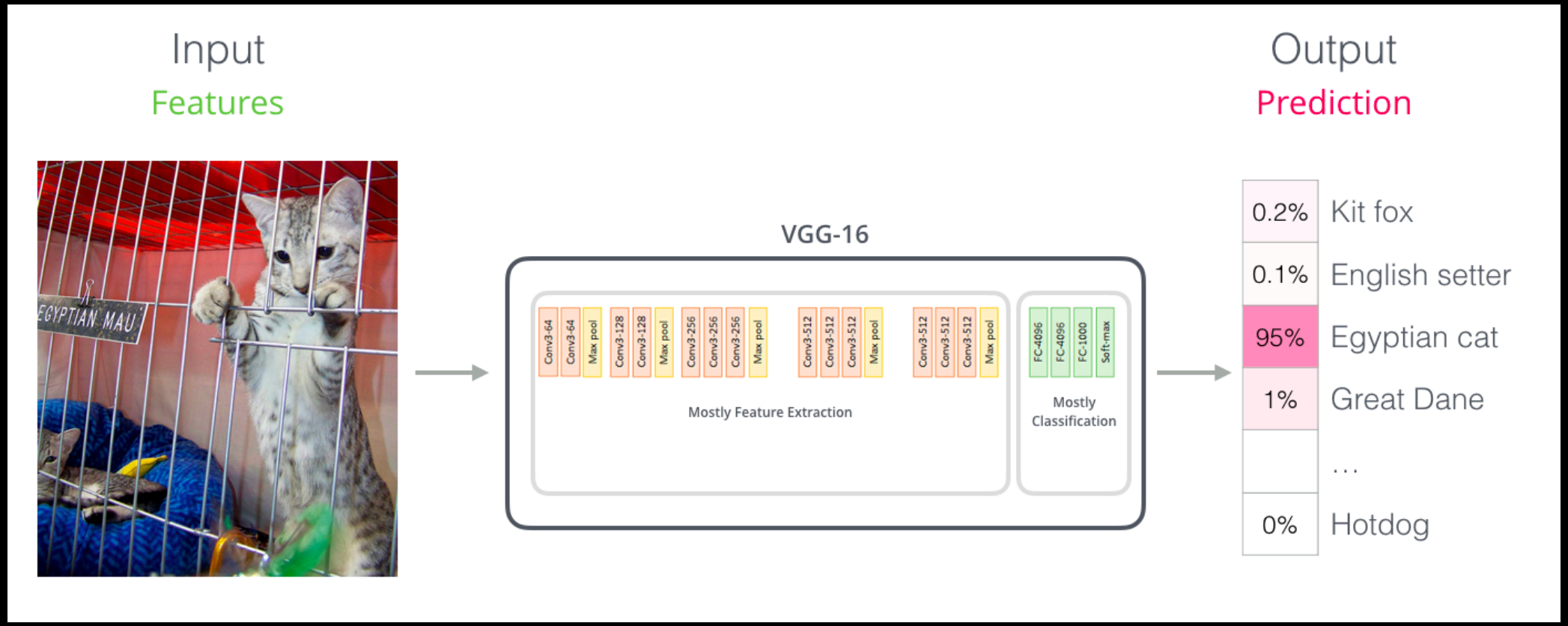
output corresponding to [CLS]

768-dim vector in BERT_base





Very- similar to popular CNN-Models:



BERT training:

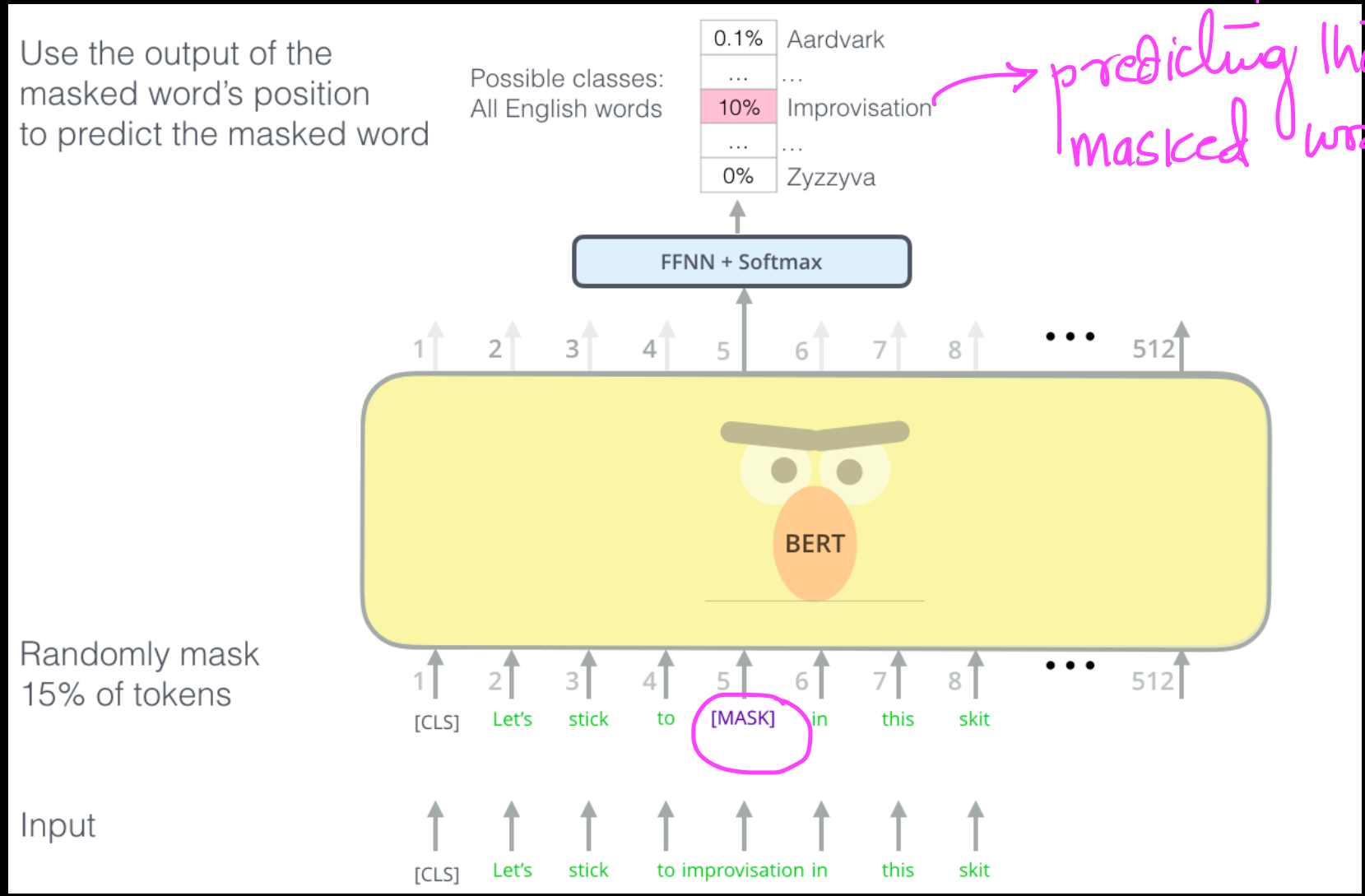
→ Bi-directional models are better than forward-only models.

⇓

looking @ words before & after matters

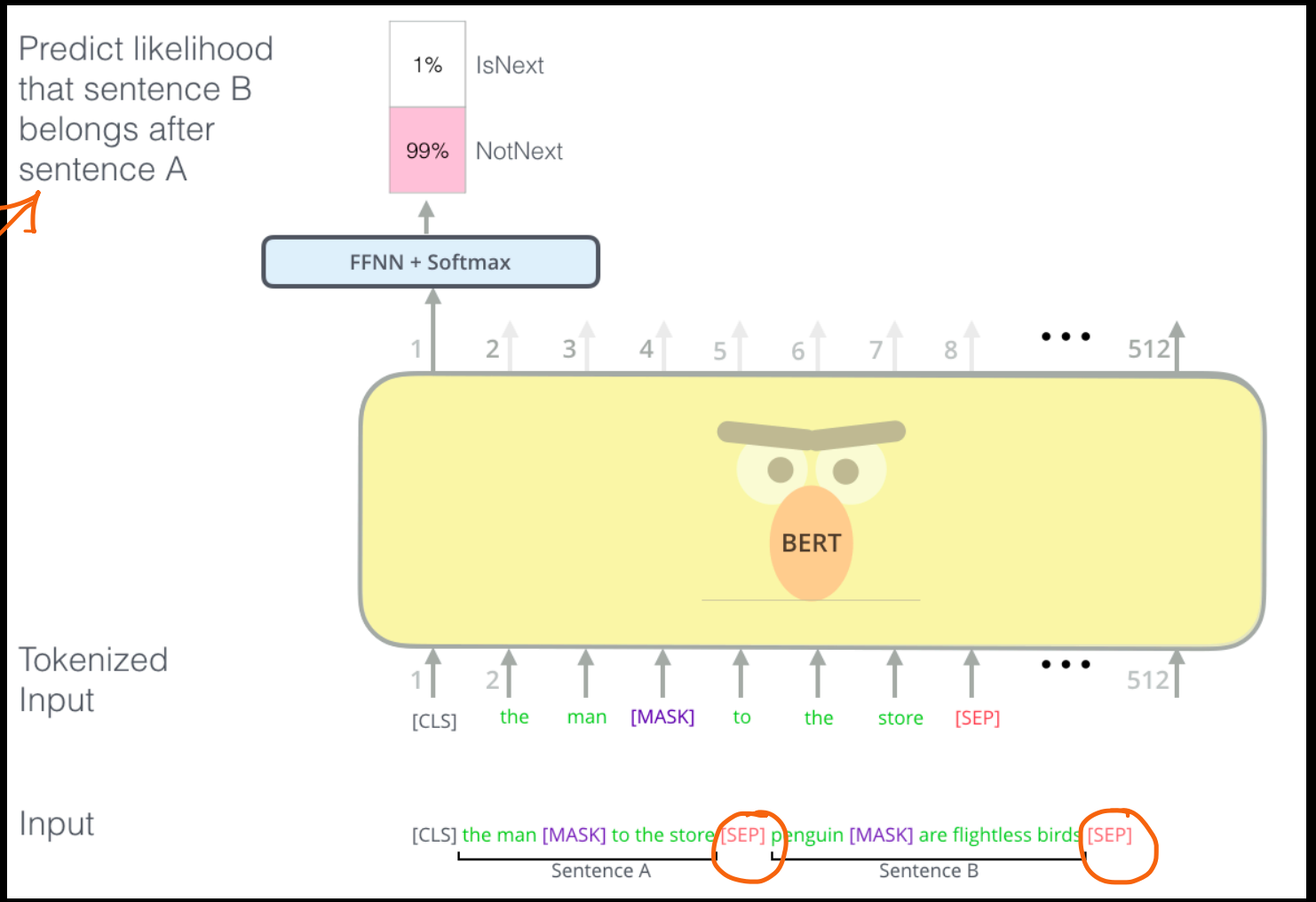
Masked Language model

Similar in idea to word2Vec

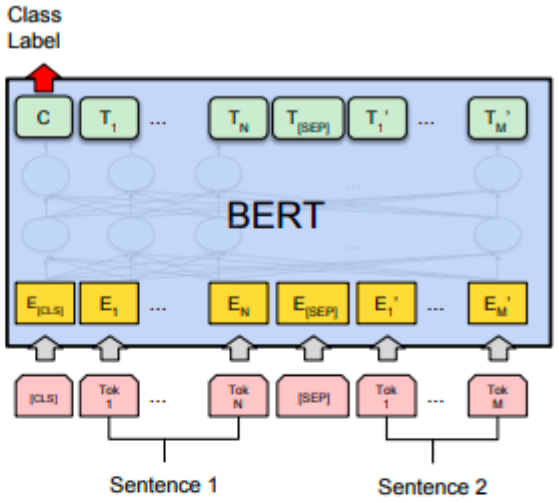


2-Sentence-tasks:

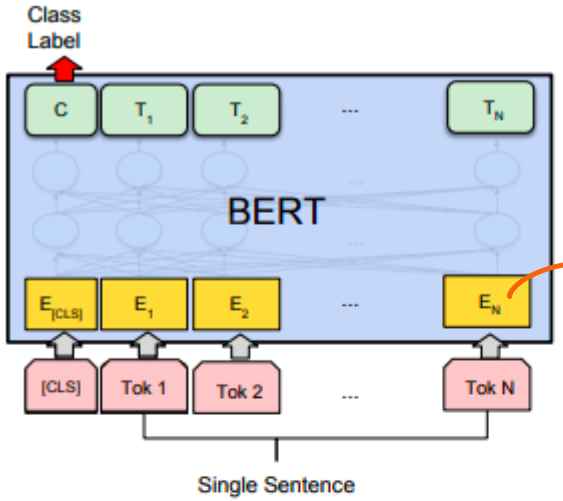
Task:



Task-specific models:

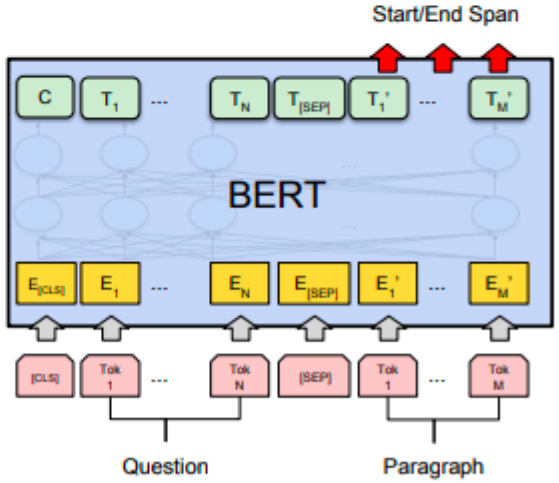


(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG

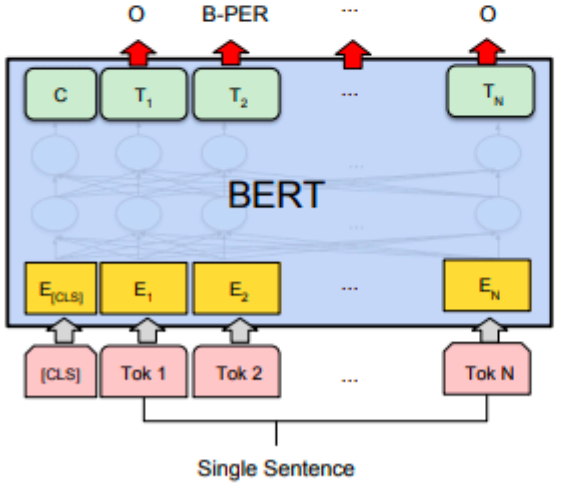


(b) Single Sentence Classification Tasks: SST-2, CoLA

word/token embeddings

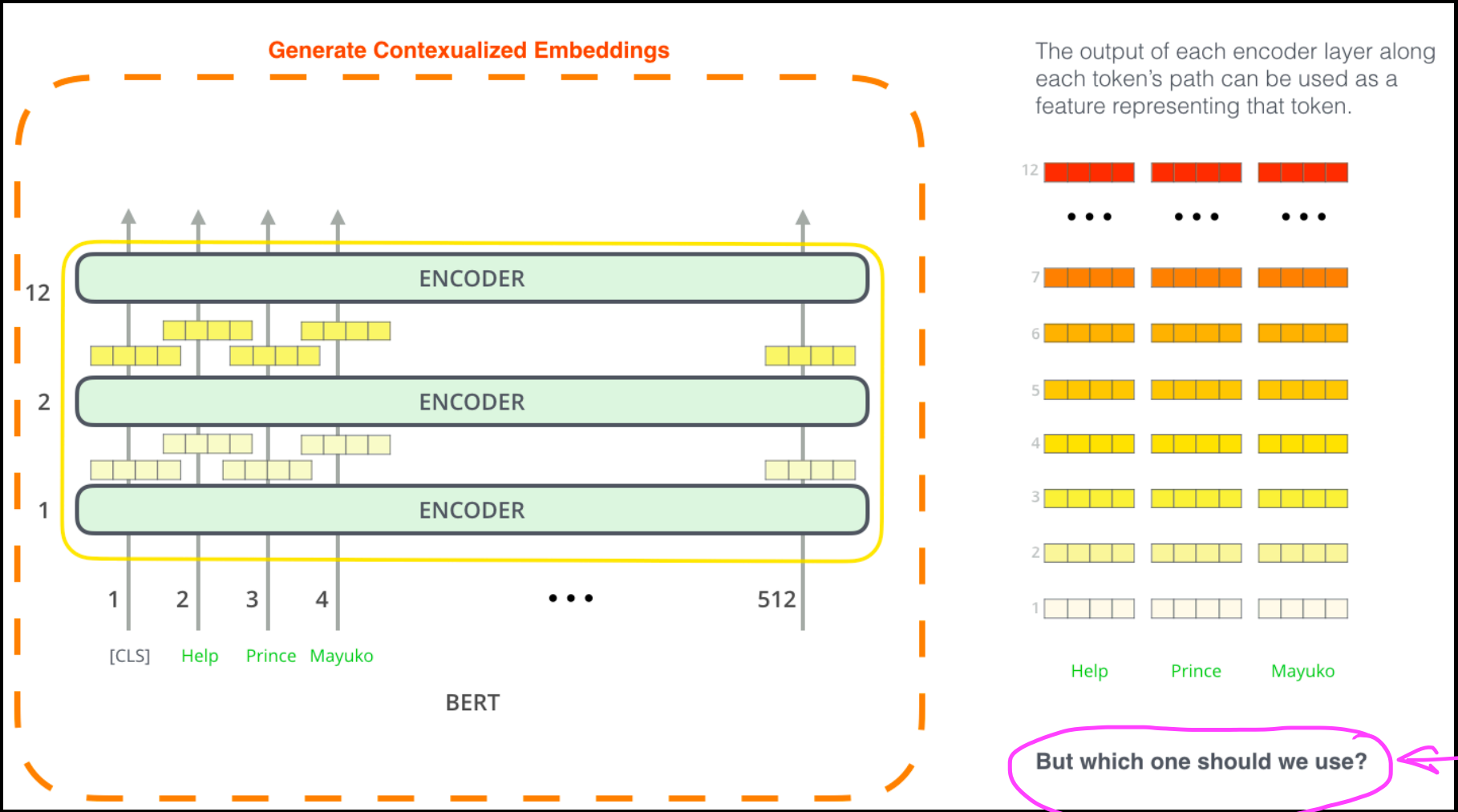


(c) Question Answering Tasks: SQuAD v1.1



(d) Single Sentence Tagging Tasks: CoNLL-2003 NER

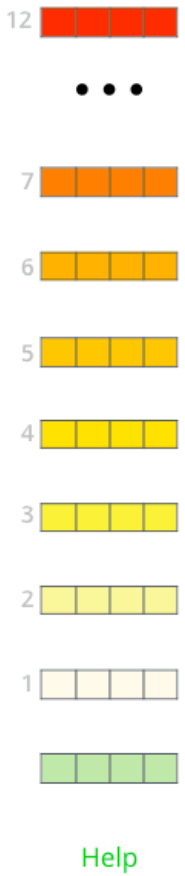
BERT - feature extraction:

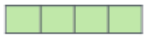
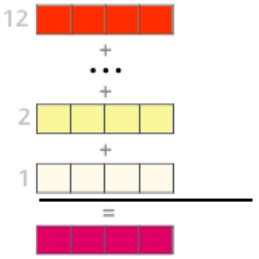
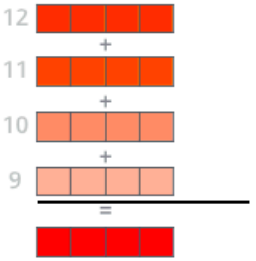
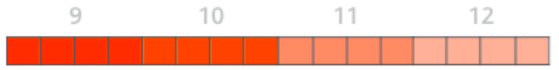


Ques

What is the best contextualized embedding for "Help" in that context?

For named-entity recognition task CoNLL-2003 NER



		Dev F1 Score
First Layer		91.0
Last Hidden Layer		94.9
Sum All 12 Layers		95.5
Second-to-Last Hidden Layer		95.6
Sum Last Four Hidden		95.9
Concat Last Four Hidden		96.1

Example to use BERT pre-trained model for sentence classification:
http://jalammar.github.io/a-visual-guide-to-using-bert-for-the-first-time/