Attention I

AC295

Pavlos Protopapas
Institute for Applied Computational Science, Harvard

Announcements

- Vote!
- Submit your reading questions by Wed 10/28 noon on Ed.
- Exercise was due 10:15 am, next coming up today.
- Project Milestone 1 - due 10/29 - 10:15 AM

Outline

Seq2seq with attention

Transformers

Bert

Outline

Seq2seq with attention

Transformers

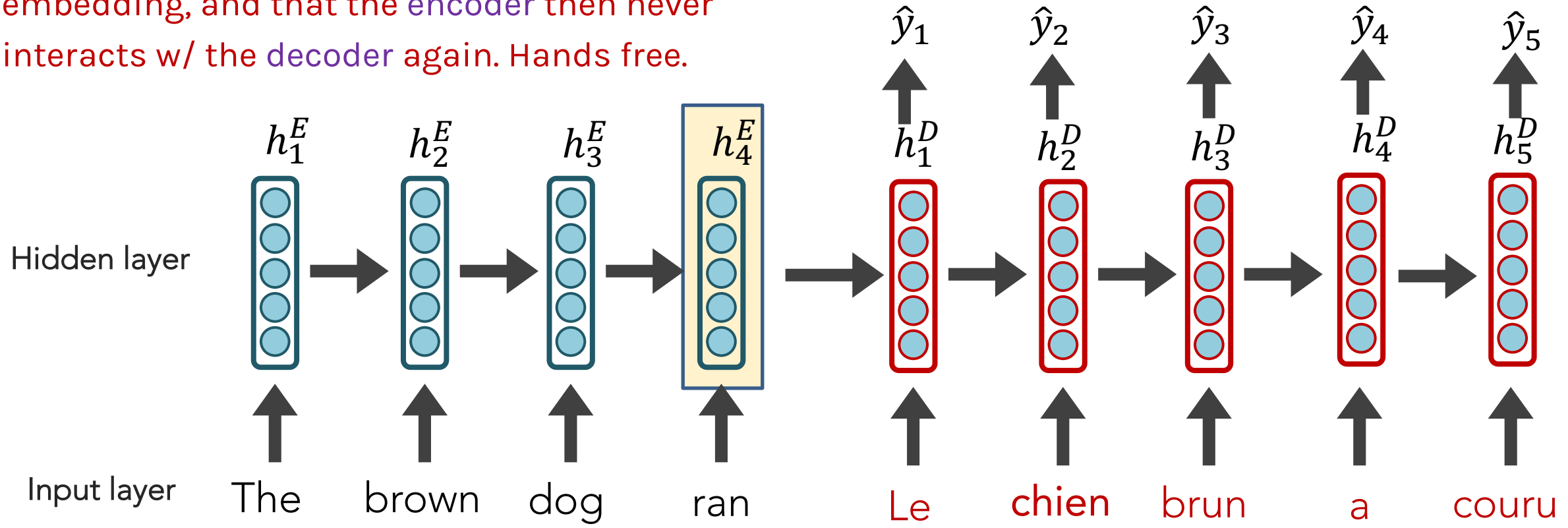
Bert

Sequence-to-Sequence (seq2seq)

See any issues with this traditional **seq2seq** paradigm?

Sequence-to-Sequence (seq2seq)

It's crazy that the entire "meaning" of the 1st sequence is expected to be packed into this one embedding, and that the encoder then never interacts w/ the decoder again. Hands free.

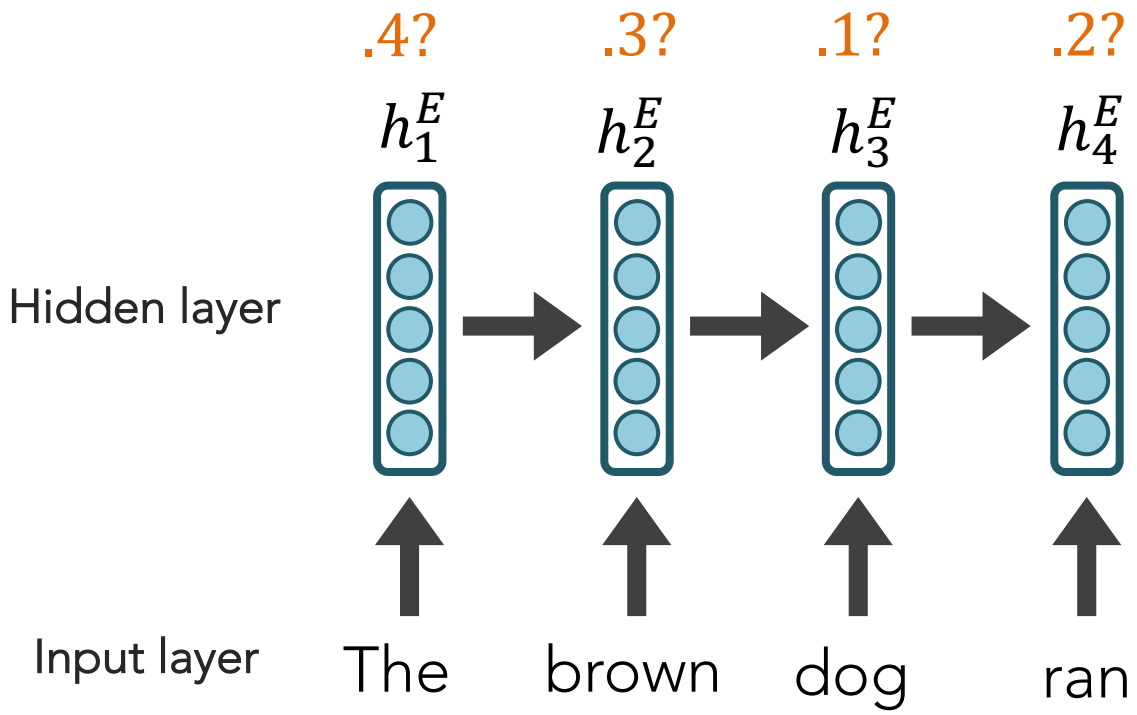


Sequence-to-Sequence (seq2seq)

Instead, what if the decoder, at each step, pays attention to a *distribution* of all of the encoder's hidden states?

seq2seq + Attention

Q: How do we determine how much to pay attention to each of the encoder's hidden layers?

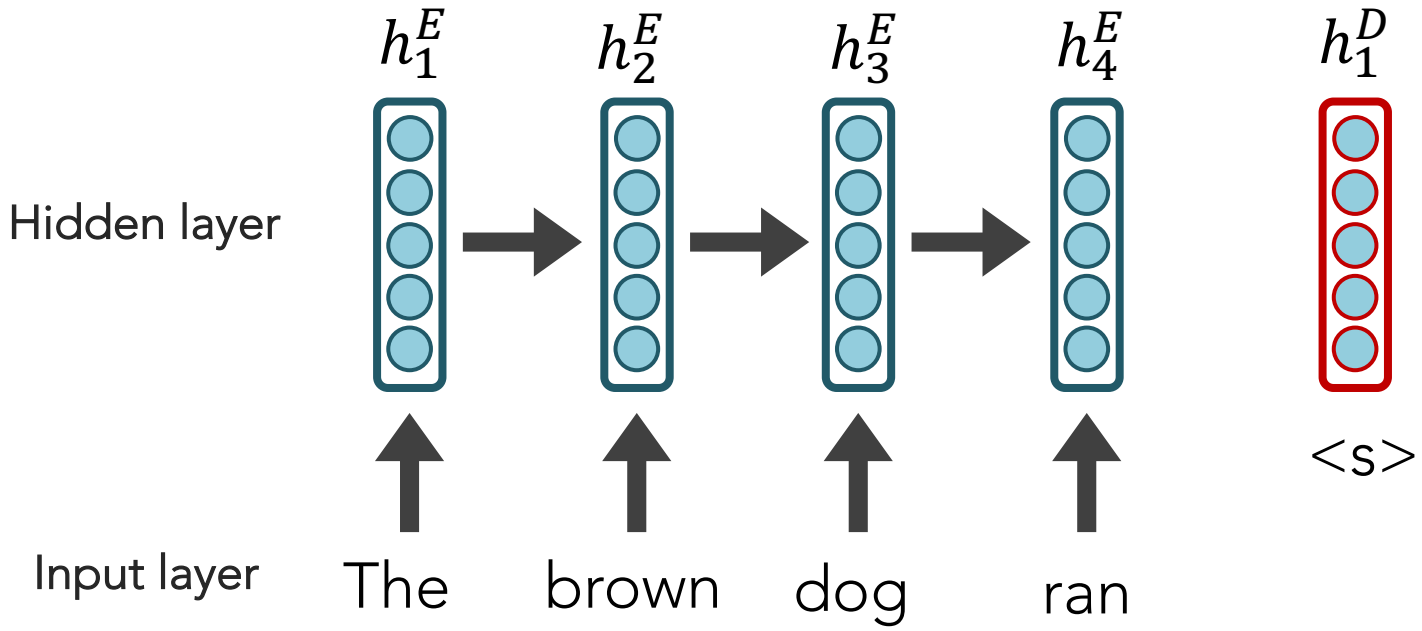


ENCODER RNN

seq2seq + Attention

Q: How do we determine how much to pay attention to each of the encoder's hidden layers?

A: Let's base it on our decoder's previous hidden state (our latest representation of meaning) and all of the encoder's hidden layers!

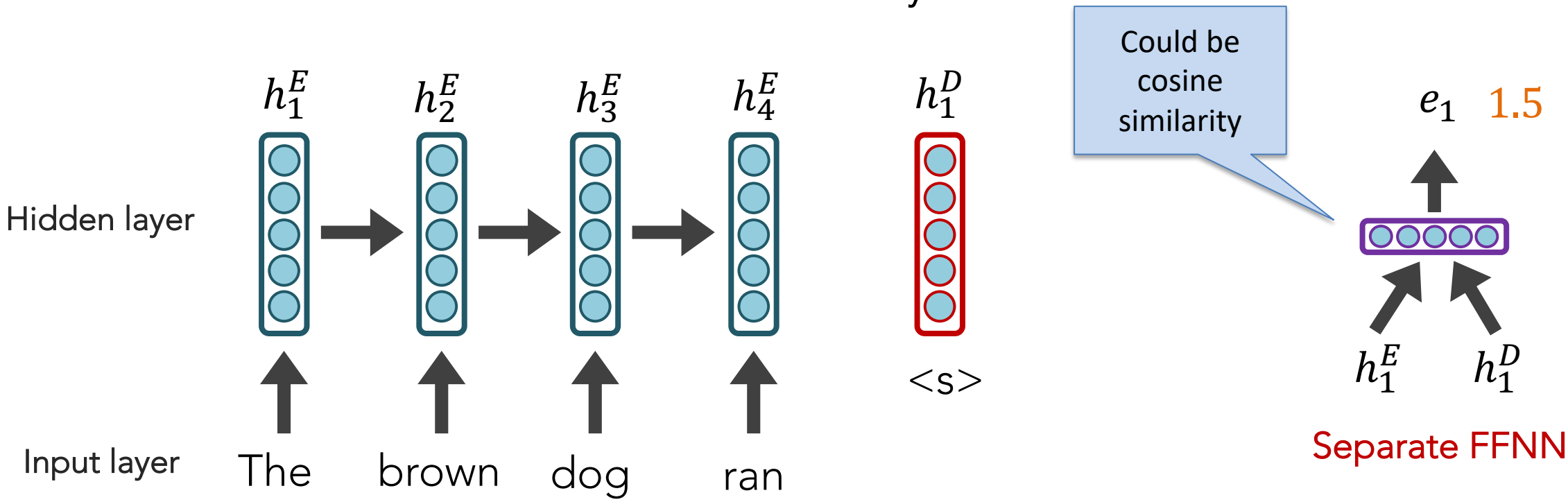


ENCODER RNN

seq2seq + Attention

Q: How do we determine how much to pay attention to each of the encoder's hidden layers?

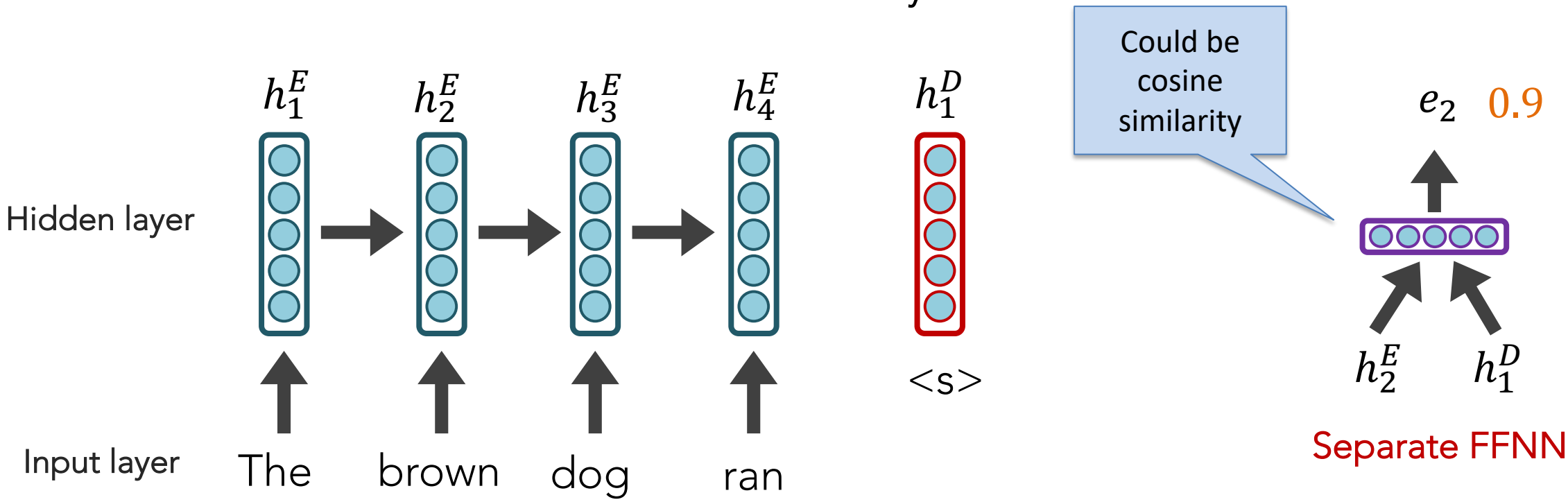
A: Let's base it on our decoder's previous hidden state (our latest representation of meaning) and all of the encoder's hidden layers! We want to measure similarity between decoder hidden state and encoder hidden stateS in some ways.



seq2seq + Attention

Q: How do we determine how much to pay attention to each of the encoder's hidden layers?

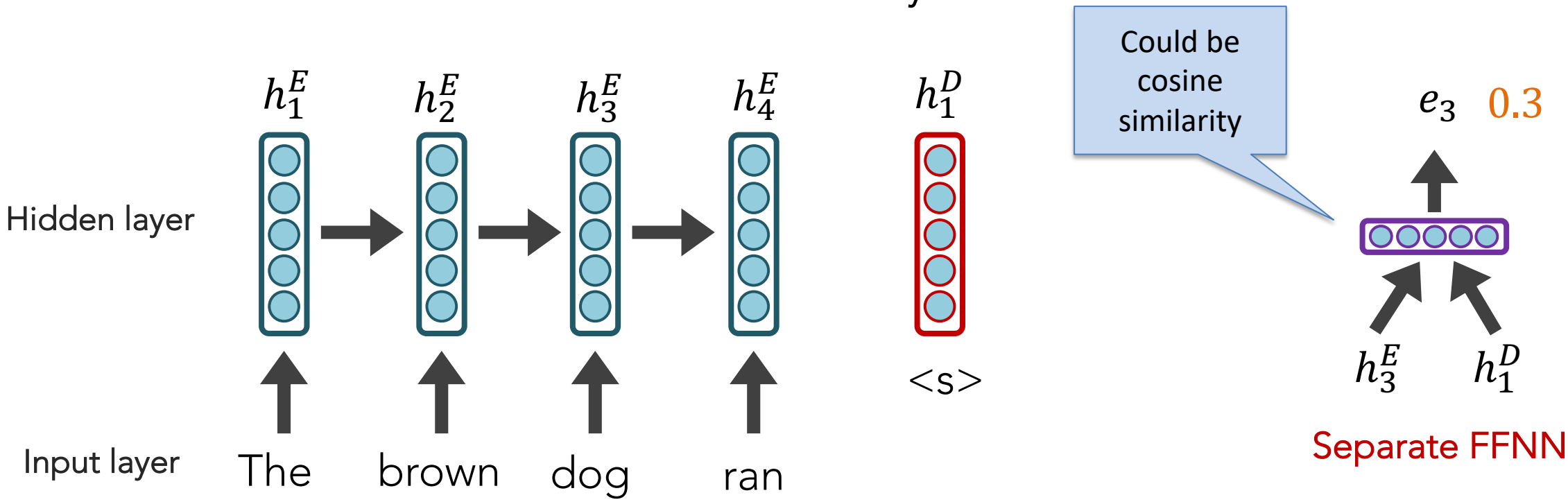
A: Let's base it on our decoder's previous hidden state (our latest representation of meaning) and all of the encoder's hidden layers! We want to measure similarity between decoder hidden state and encoder hidden stateS in some ways.



seq2seq + Attention

Q: How do we determine how much to pay attention to each of the encoder's hidden layers?

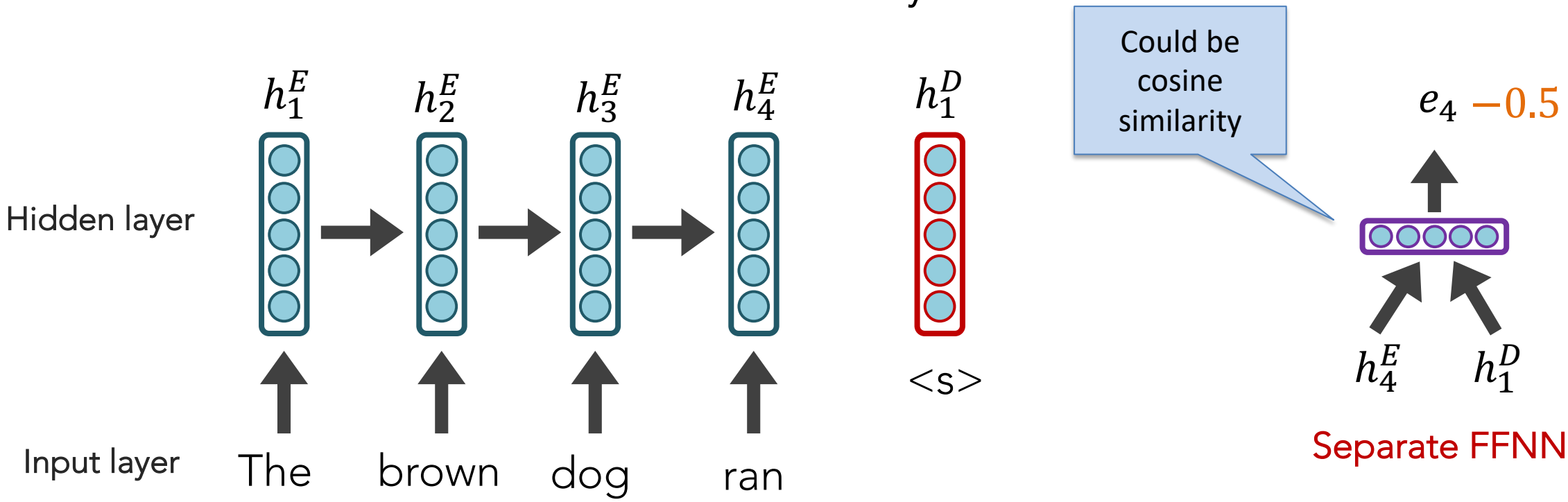
A: Let's base it on our decoder's previous hidden state (our latest representation of meaning) and all of the encoder's hidden layers! We want to measure similarity between decoder hidden state and encoder hidden stateS in some ways.



seq2seq + Attention

Q: How do we determine how much to pay attention to each of the encoder's hidden layers?

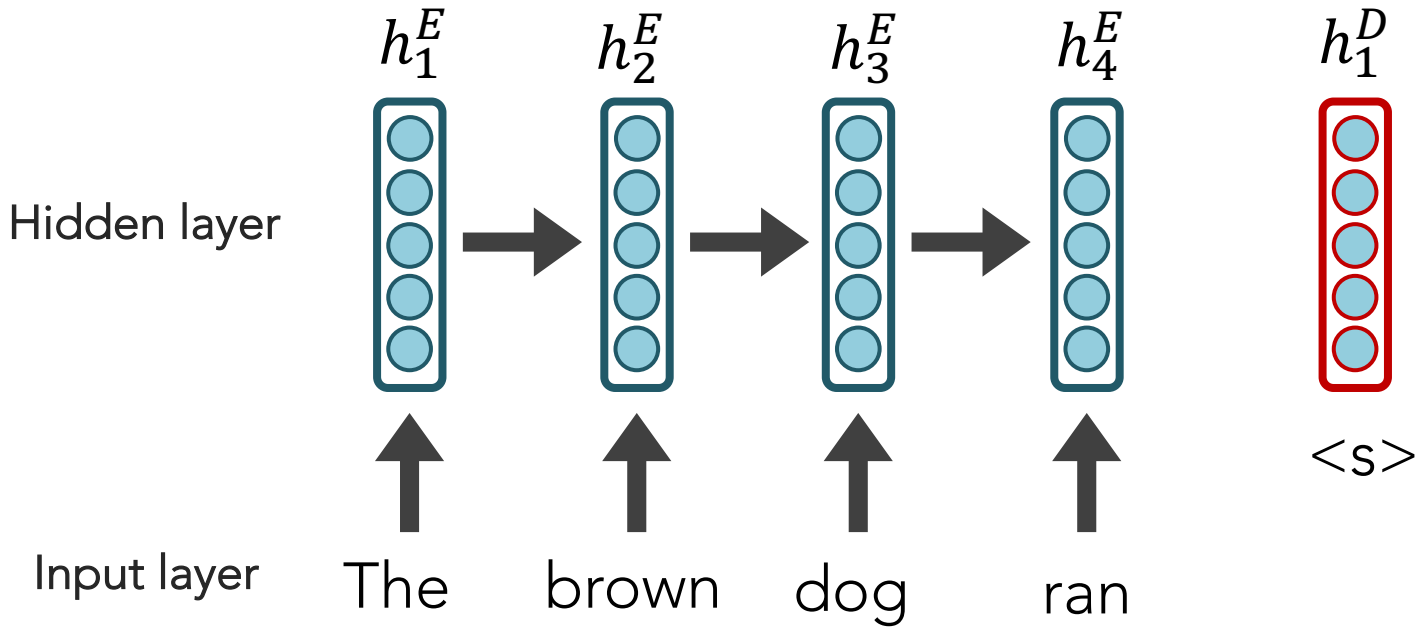
A: Let's base it on our decoder's previous hidden state (our latest representation of meaning) and all of the encoder's hidden layers! We want to measure similarity between decoder hidden state and encoder hidden stateS in some ways.



seq2seq + Attention

Q: How do we determine how much to pay attention to each of the encoder's hidden layers?

A: Let's base it on our decoder's previous hidden state (our latest representation of meaning) and all of the encoder's hidden layers!



Attention (raw scores)

e_1 1.5

e_2 0.9

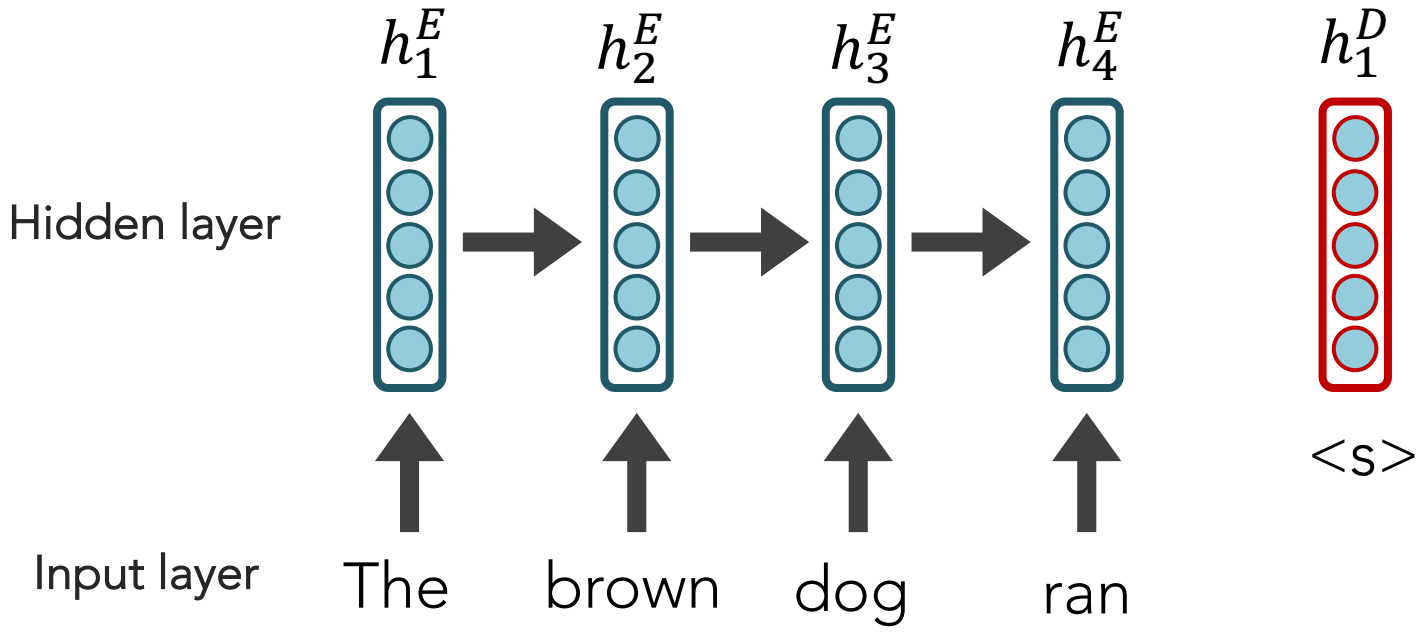
e_3 0.2

e_4 −0.5

seq2seq + Attention

Q: How do we determine how much to pay attention to each of the encoder's hidden layers?

A: Let's base it on our decoder's previous hidden state (our latest representation of meaning) and all of the encoder's hidden layers!



Attention (raw scores)

e_1 1.5

e_2 0.9

e_3 0.2

e_4 −0.5

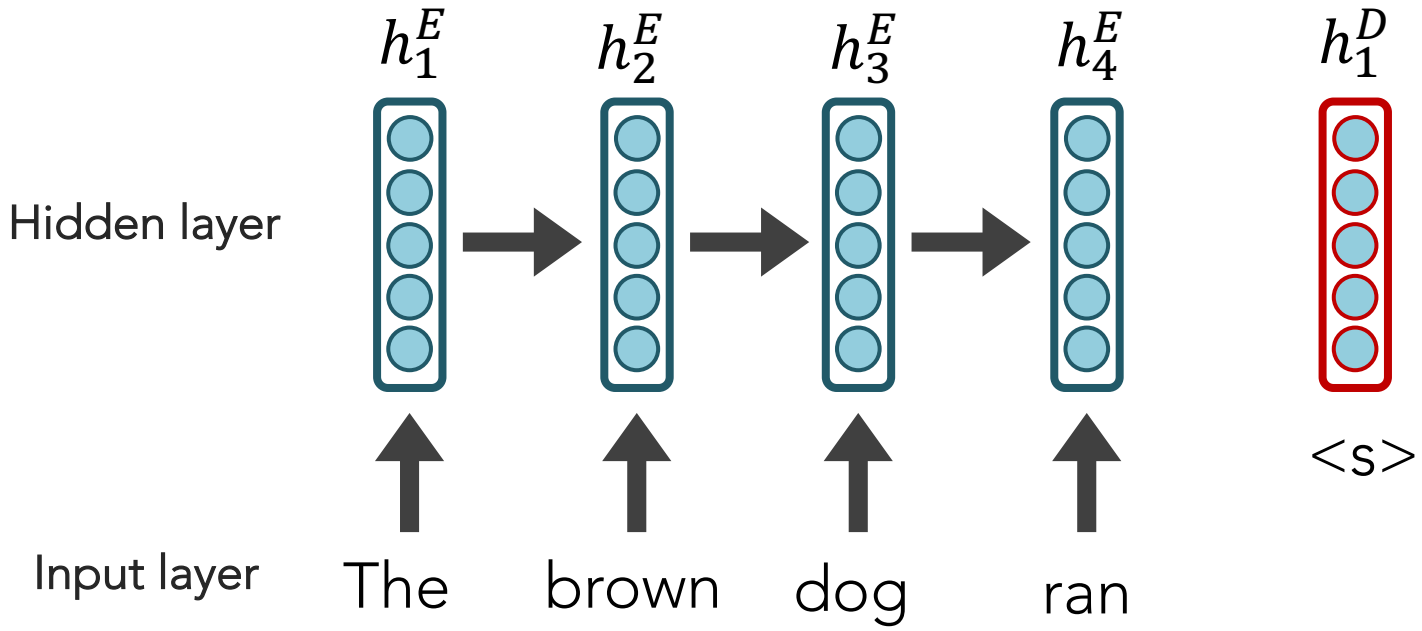
Attention (softmax'd)

$$a_i^1 = \frac{\exp(e_i)}{\sum_i^N \exp(e_1)}$$

seq2seq + Attention

Q: How do we determine how much to pay attention to each of the encoder's hidden layers?

A: Let's base it on our decoder's previous hidden state (our latest representation of meaning) and all of the encoder's hidden layers!



ENCODER RNN

Attention (raw scores)

e_1 1.5

e_2 0.9

e_3 0.2

e_4 −0.5

Attention (softmax'd)

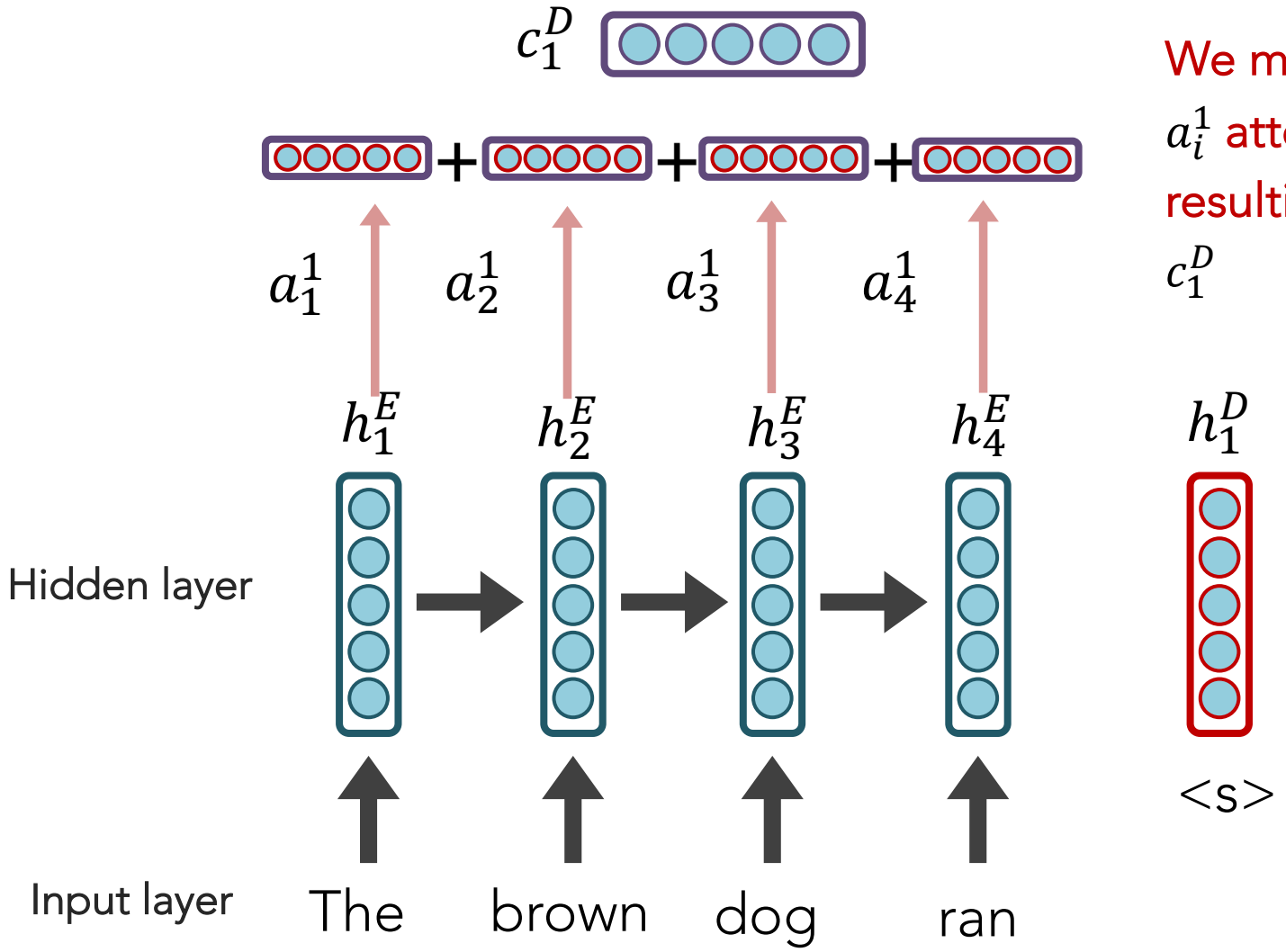
$a_1^1 = 0.51$

$a_2^1 = 0.28$

$a_3^1 = 0.14$

$a_3^1 = 0.07$

seq2seq + Attention



We multiply each hidden layer by its a_i^1 attention weights and then add the resulting vectors to create a context vector c_1^D

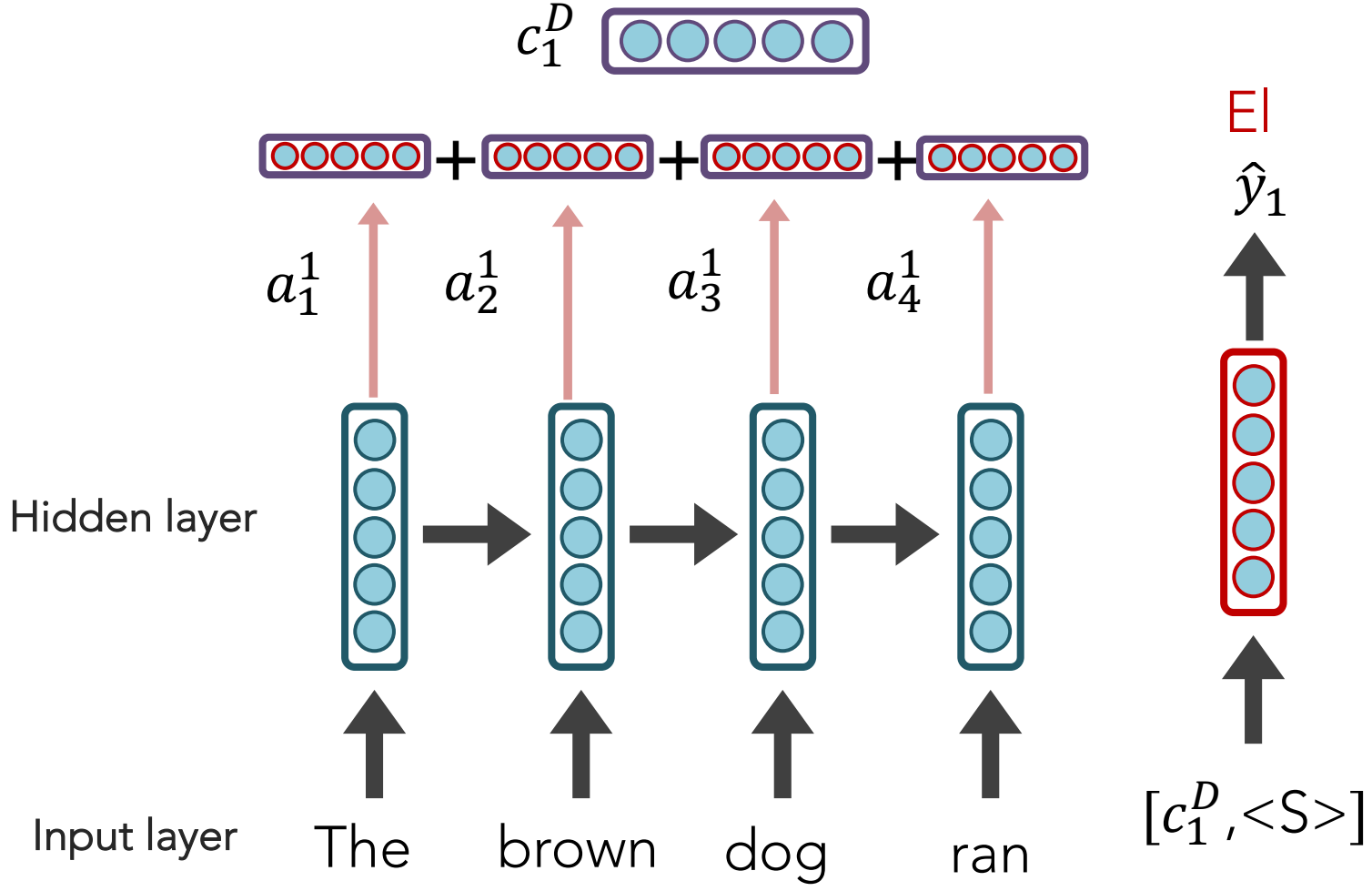
Attention (softmax'd)

$$a_1^1 = 0.51$$
$$a_2^1 = 0.28$$
$$a_3^1 = 0.14$$
$$a_4^1 = 0.07$$

ENCODER RNN

seq2seq + Attention

NOTE: each attention weight a_i^j is based on the decoder's current hidden state, too.

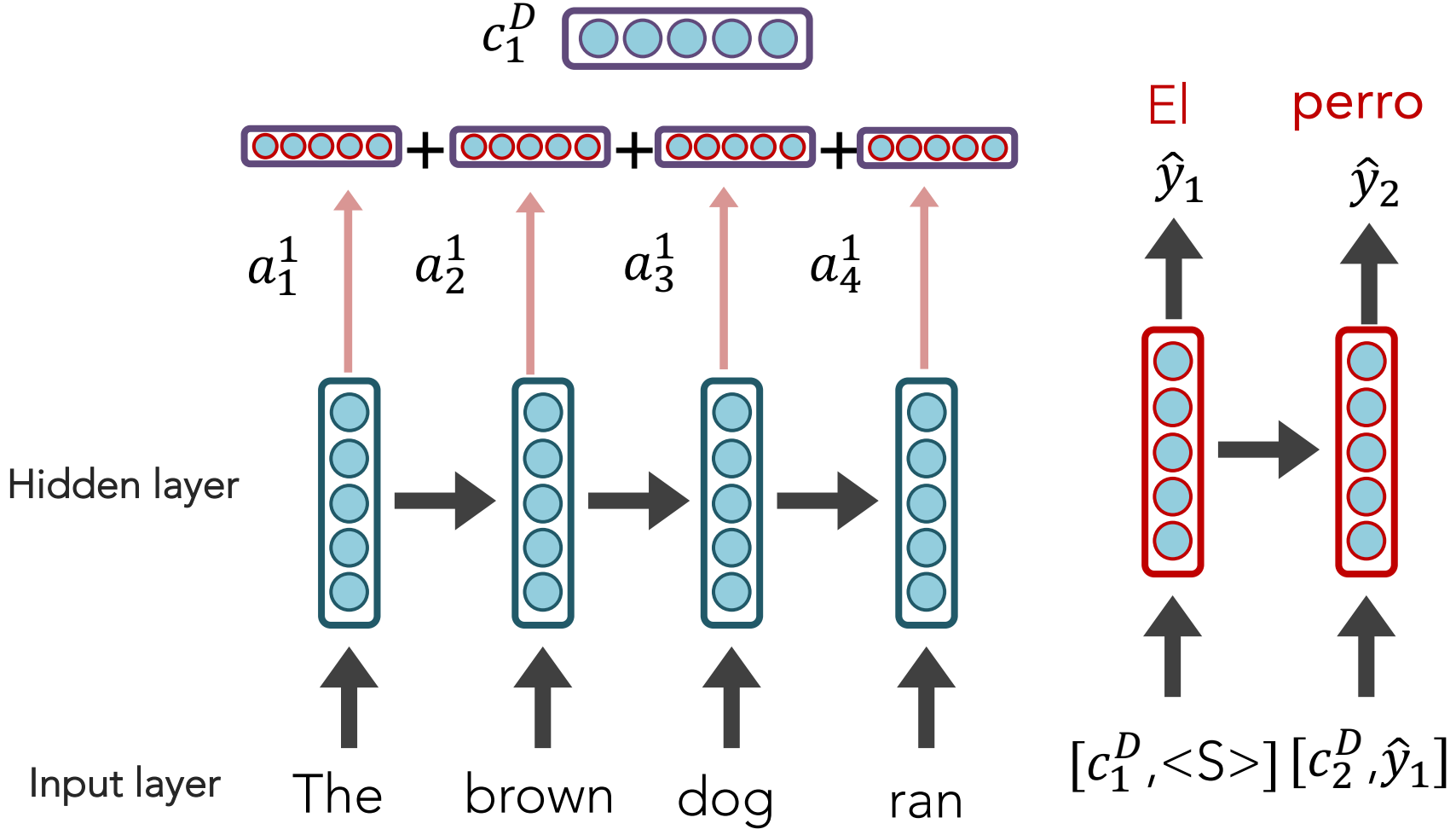


ENCODER RNN

DECODER RNN

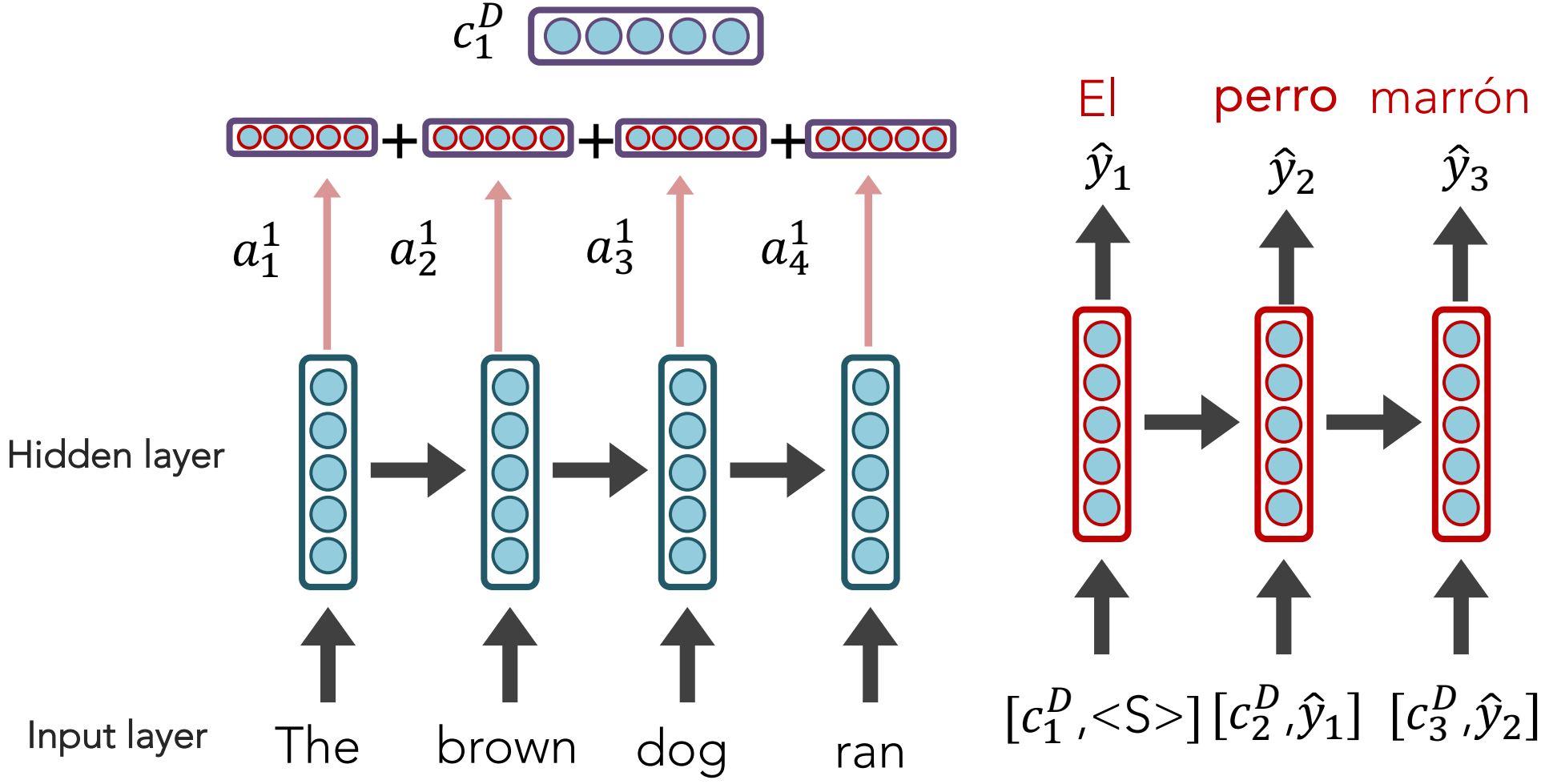
seq2seq + Attention

NOTE: each attention weight a_i^j is based on the decoder's current hidden state, too.



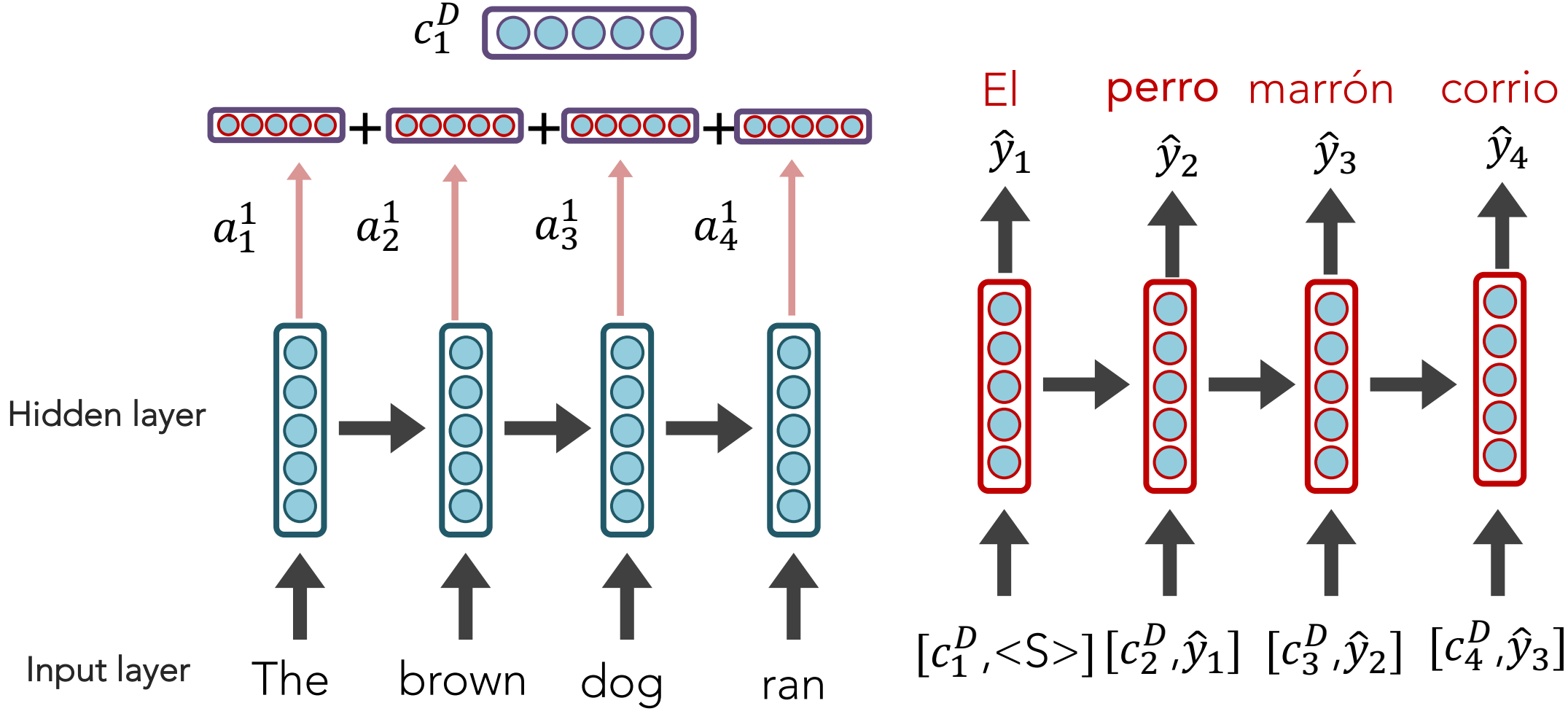
seq2seq + Attention

NOTE: each attention weight a_i^j is based on the decoder's current hidden state, too.



seq2seq + Attention

NOTE: each attention weight a_i^j is based on the decoder's current hidden state, too.



seq2seq + Attention

Attention:

- greatly improves seq2seq results
- allows us to visualize the contribution each word gave during each step of the decoder

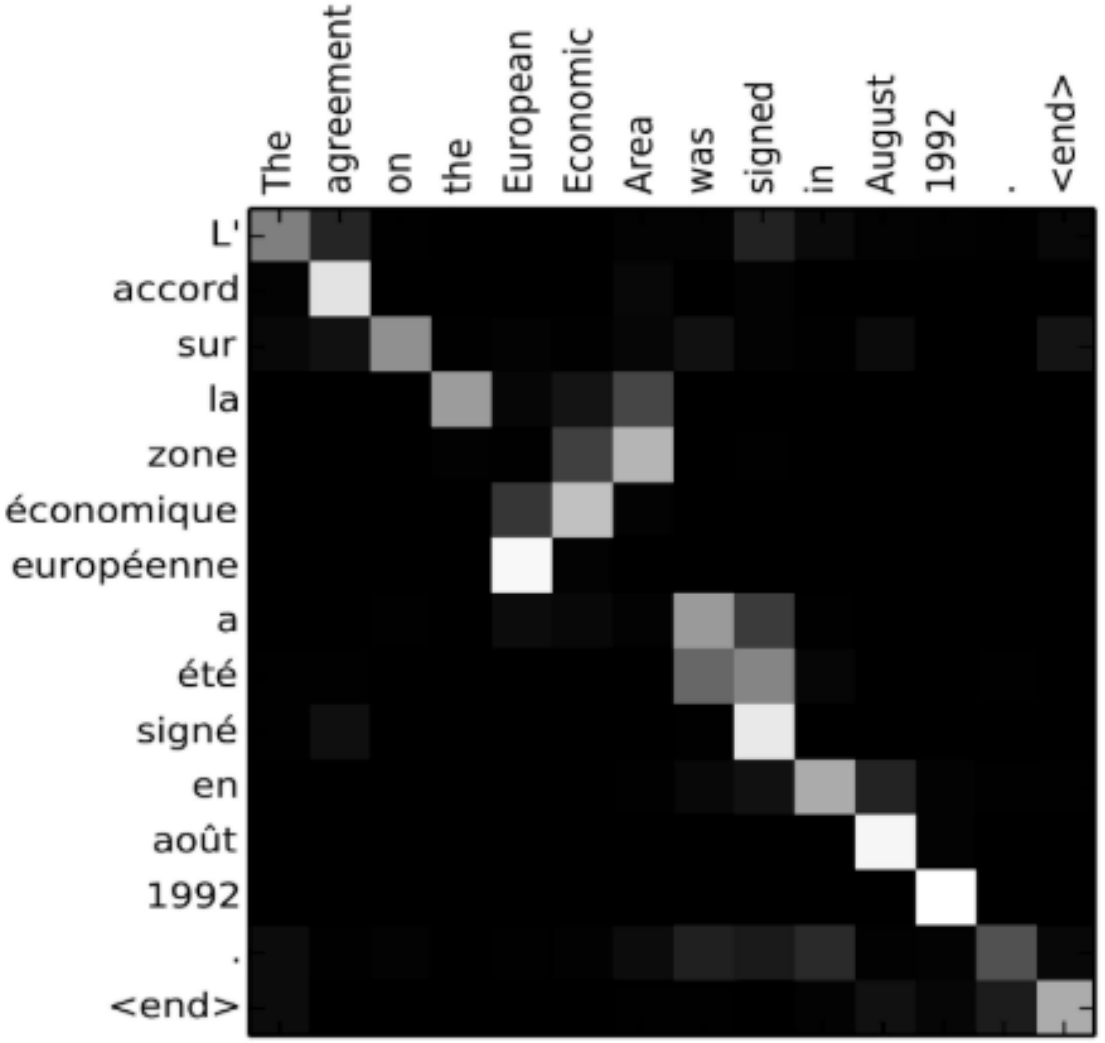


Image source: Fig 3 in Bahdanau et al., 2015

More reading

- Kalchbrenner and Blunsom (2013), Sutskever *et. al* (2014) and Cho. *et. al* (2014b)

- Bahdanau et. al (2015) https://arxiv.org/abs/1409.0473
  seq2seq with bidirectional GRU encoder + attention, score is FFNN.

- Luong et. al (2015) https://arxiv.org/abs/1508.04025
  Two-stacked LSTMs for the encoder and decoder, score experimented with cosine and FFNN. Context vector and output goes through another FFNN.

- Google's Neural Machine Translation (GNMT)
  https://arxiv.org/pdf/1609.08144.pdf
  8 LSTMs, where the first is bidirectional (whose outputs are concatenated), and a residual connection exists between outputs from consecutive layers (starting from the 3rd layer). The decoder is a *separate* stack of 8 unidirectional LSTMs.

Outline

Seq2seq with attention

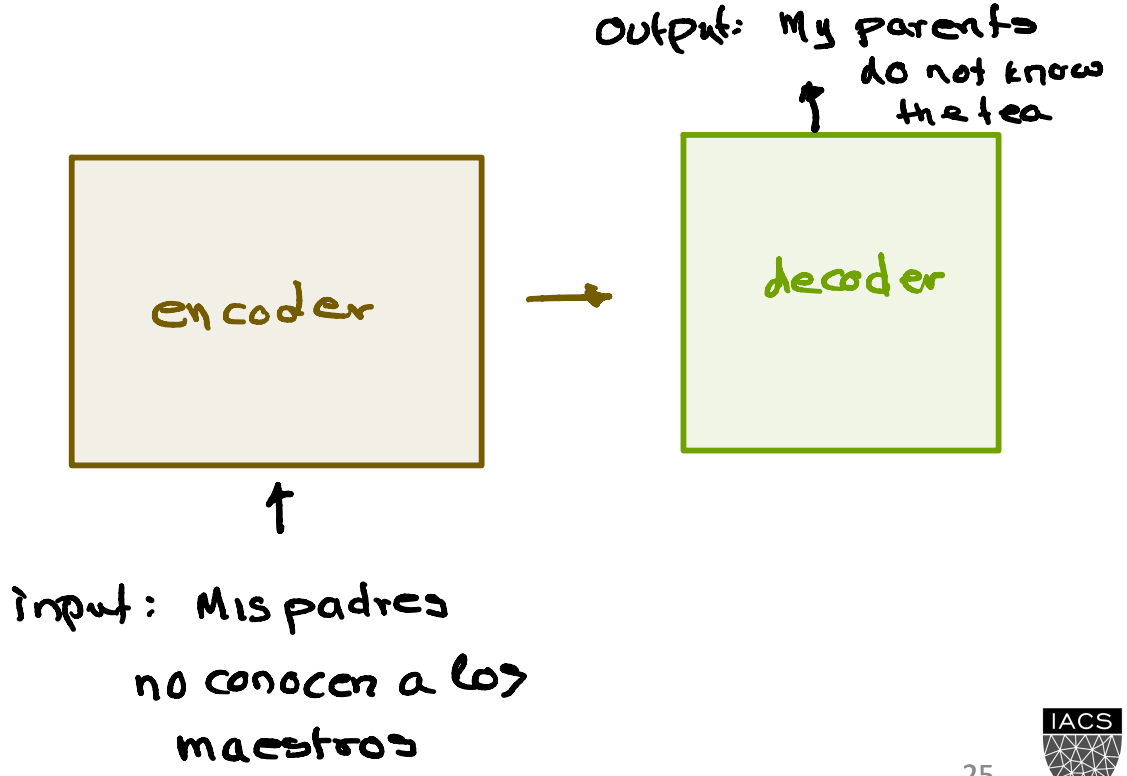
Transformers

Bert

Transformers

The Transformer is a model that uses attention to boost the speed with which seq2seq with attention models can be trained. The biggest benefit, however, comes from how The Transformer lends itself to parallelization. We will break it apart and look at how it functions.

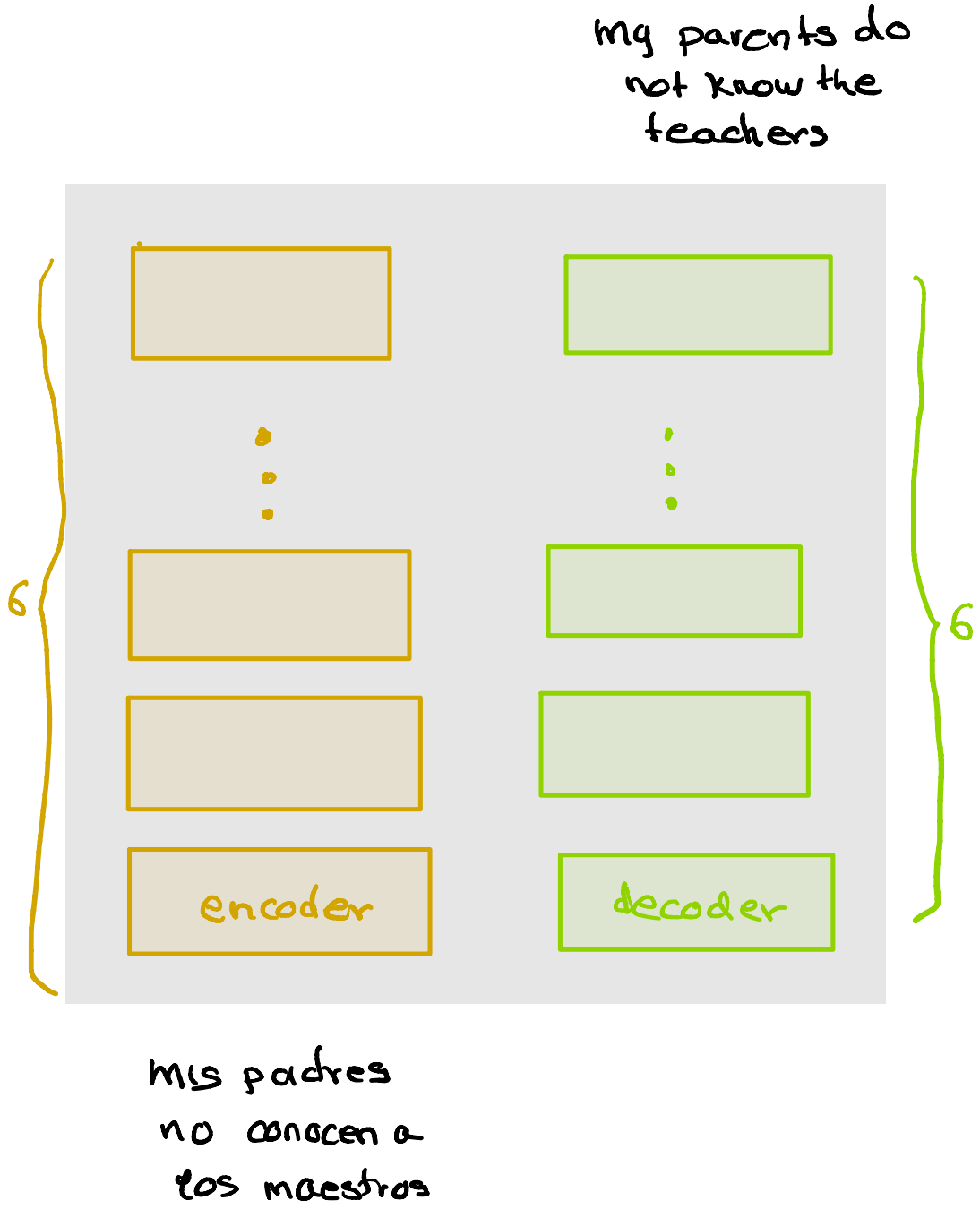
In its heart it contains an encoding component, a decoding component, and connections between them.



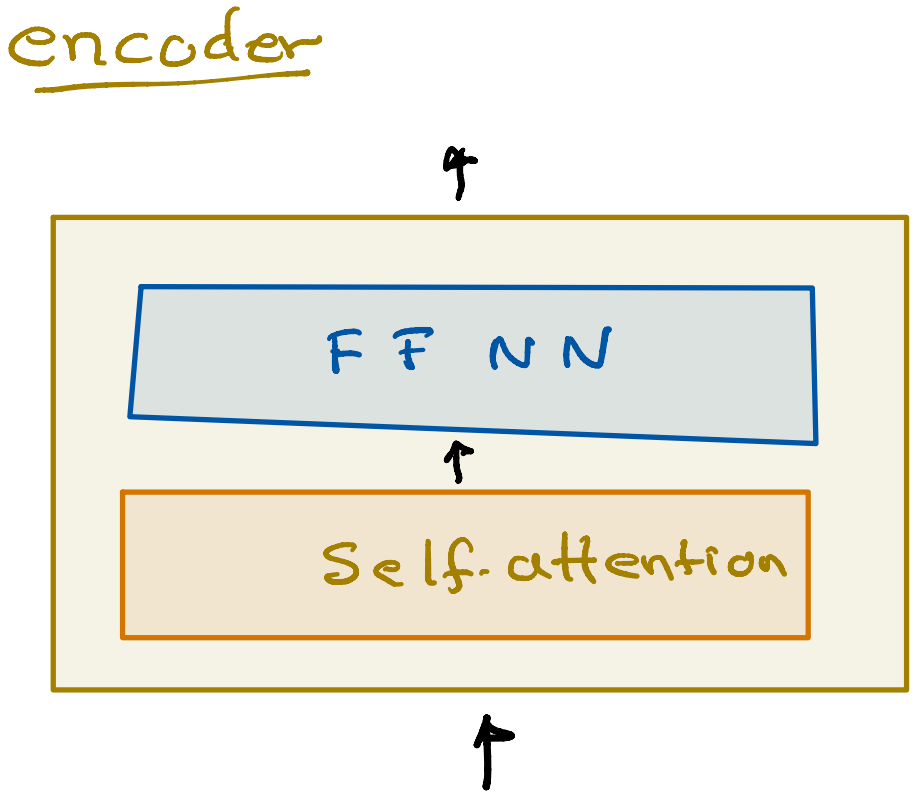
The encoding is a stack of encoders.

The original paper stacks six of them on top of each other – there's nothing magical about the number six, one can definitely experiment with other arrangements).

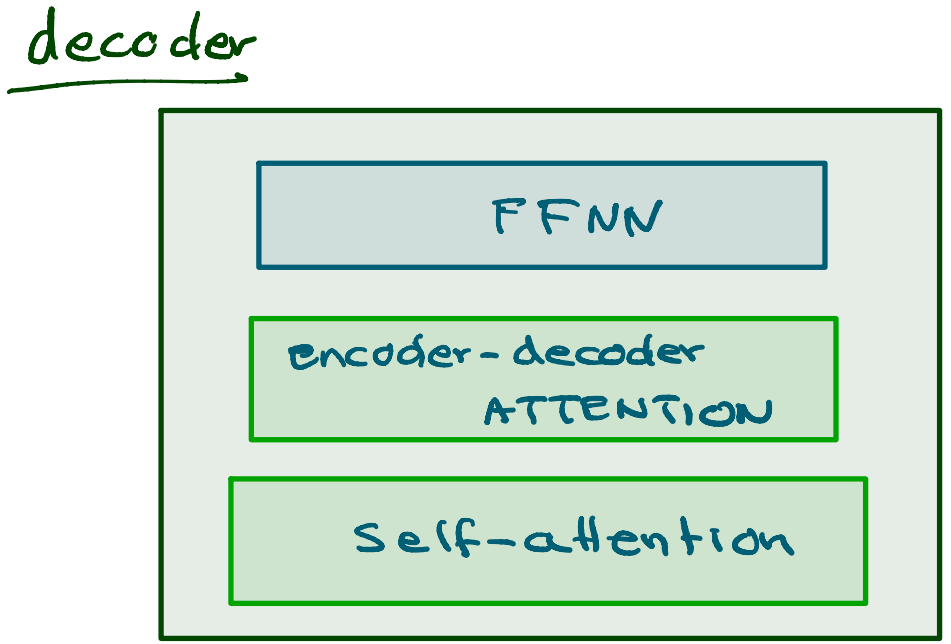
The decoding is a stack of decoders of the same number.

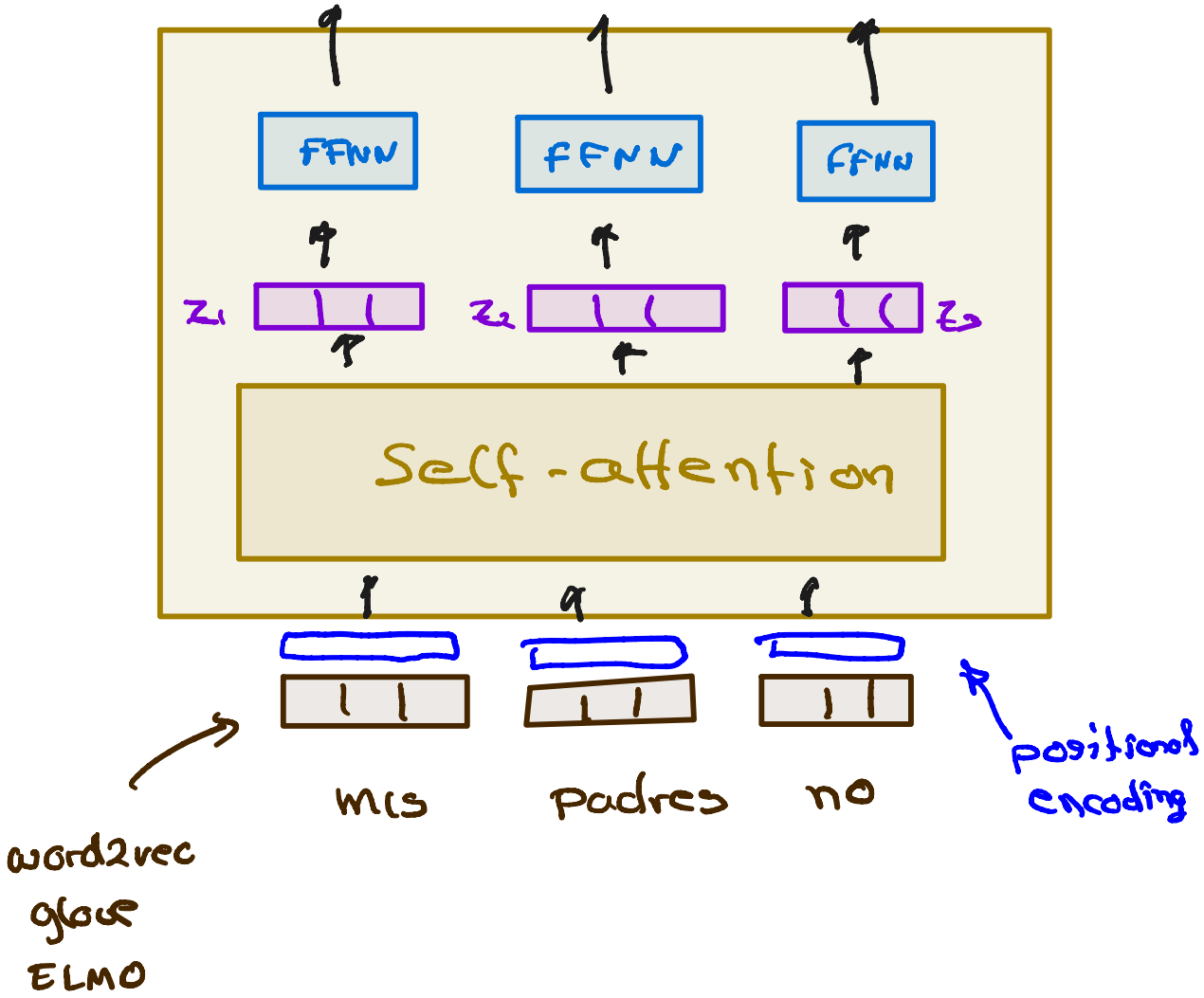


The encoder's inputs goes through a self-attention layer – a layer that helps the encoder look at other words in the input sentence as it encodes a specific word.



The decoder has both those layers, but between them is an attention layer that helps the decoder focus on relevant parts of the input sentence (similar what attention does in seq2seq models).



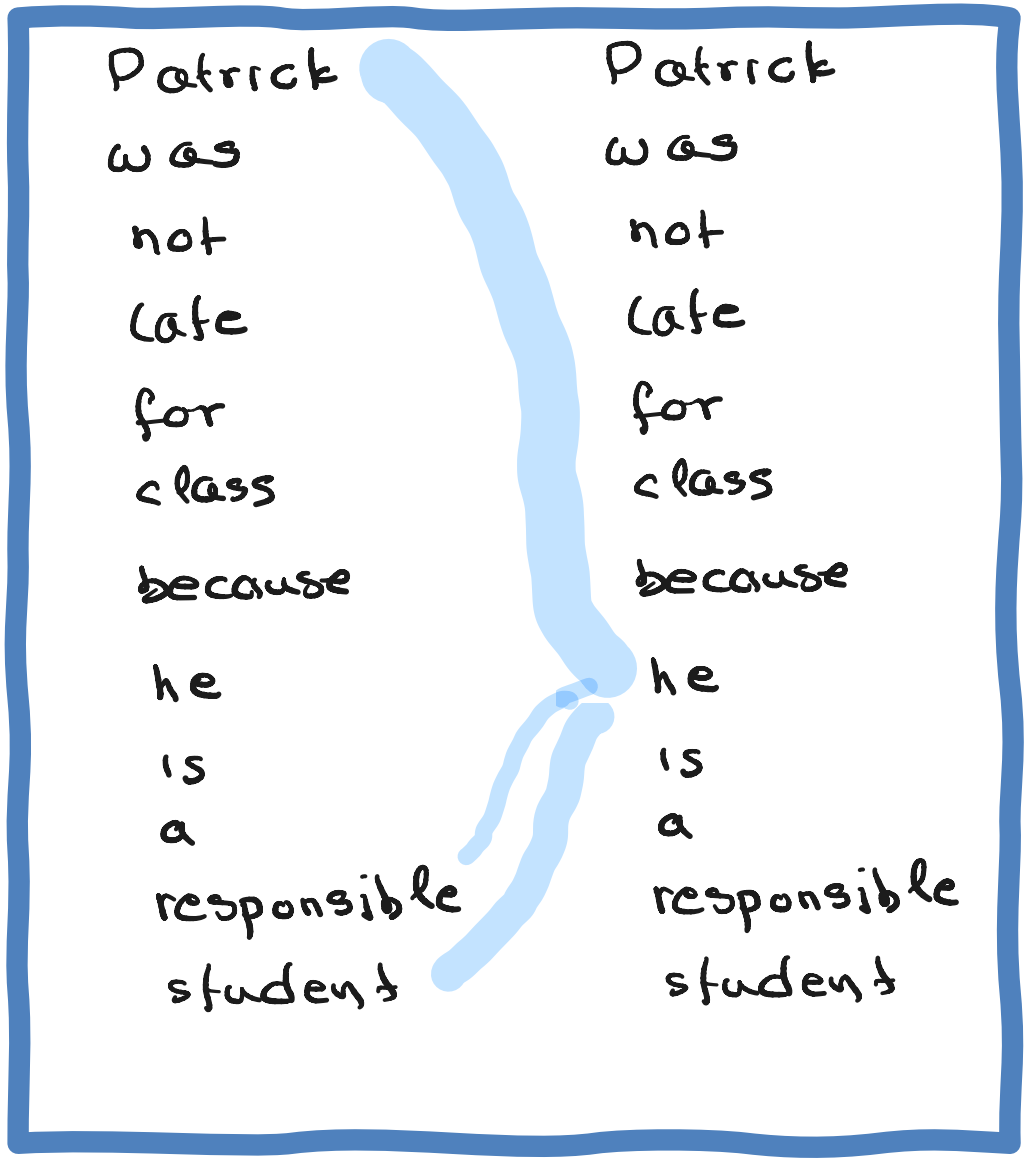


A key property of the Transformer, which is that the word in each position flows through its own path in the encoder. There are dependencies between these paths in the self-attention layer.

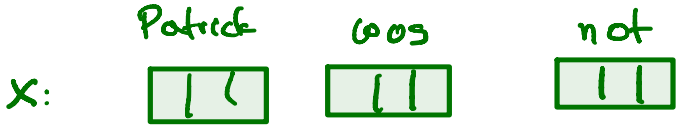
The feed-forward layer does not have those dependencies, Therefore, the various paths can be executed in parallel while flowing through the feed-forward layer.

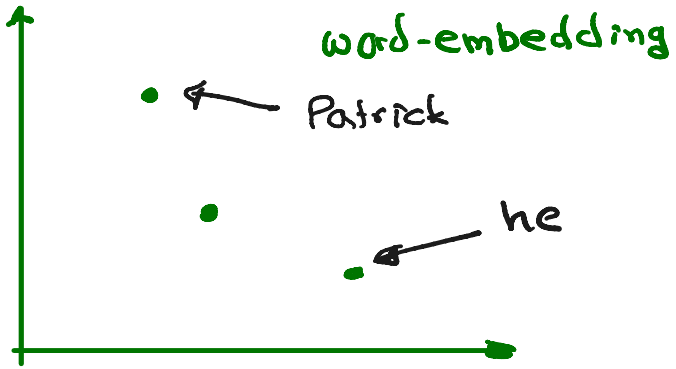
Patrick was not late for class because he is a responsible student

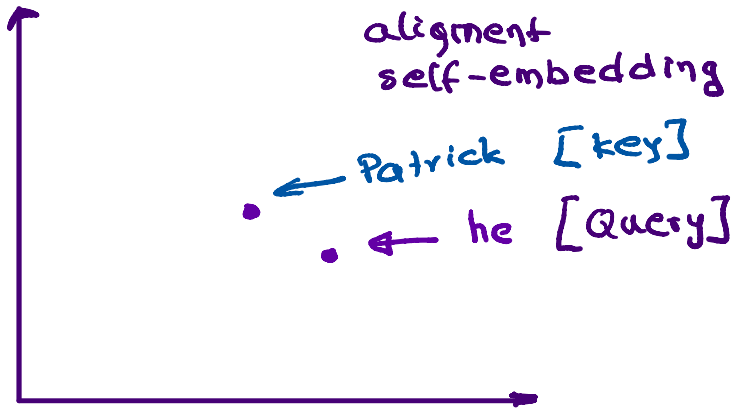
What does "he" in this sentence refer to? Is it referring to the class or to the student? It's a simple question to a human, but not as simple to an algorithm.
When the model is processing the word "he", self-attention allows it to associate "he" with "Patrick".



self-attention







Query:

$$Q = X W^{Q}$$

transformation

$n_w \times n_a$ $\quad$ $n_w \times n_e$ $\quad$ $n_e \times n_a$

key:

$$K = X W^{K}$$

Value:

$$V = X W^{V}$$

Similarity:

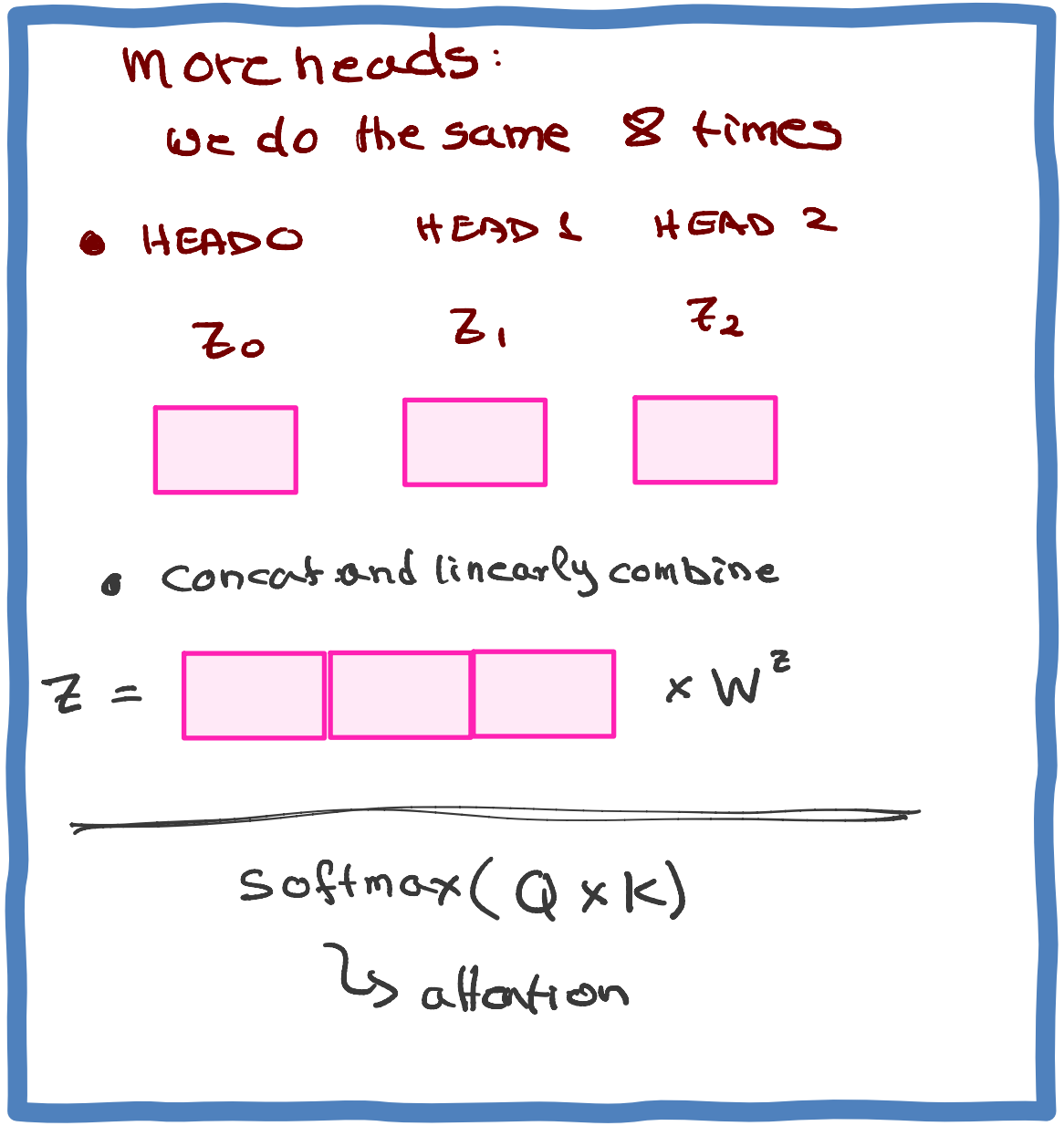
$$Z^{T} = \text{softmax}\left(\frac{Q^{T} \times K}{\sqrt{d}}\right) \cdot V^{T}$$

$n_a \times n_e$ $\quad$ $n_e \times n_a$ $\quad$ $n_w \times n_a$

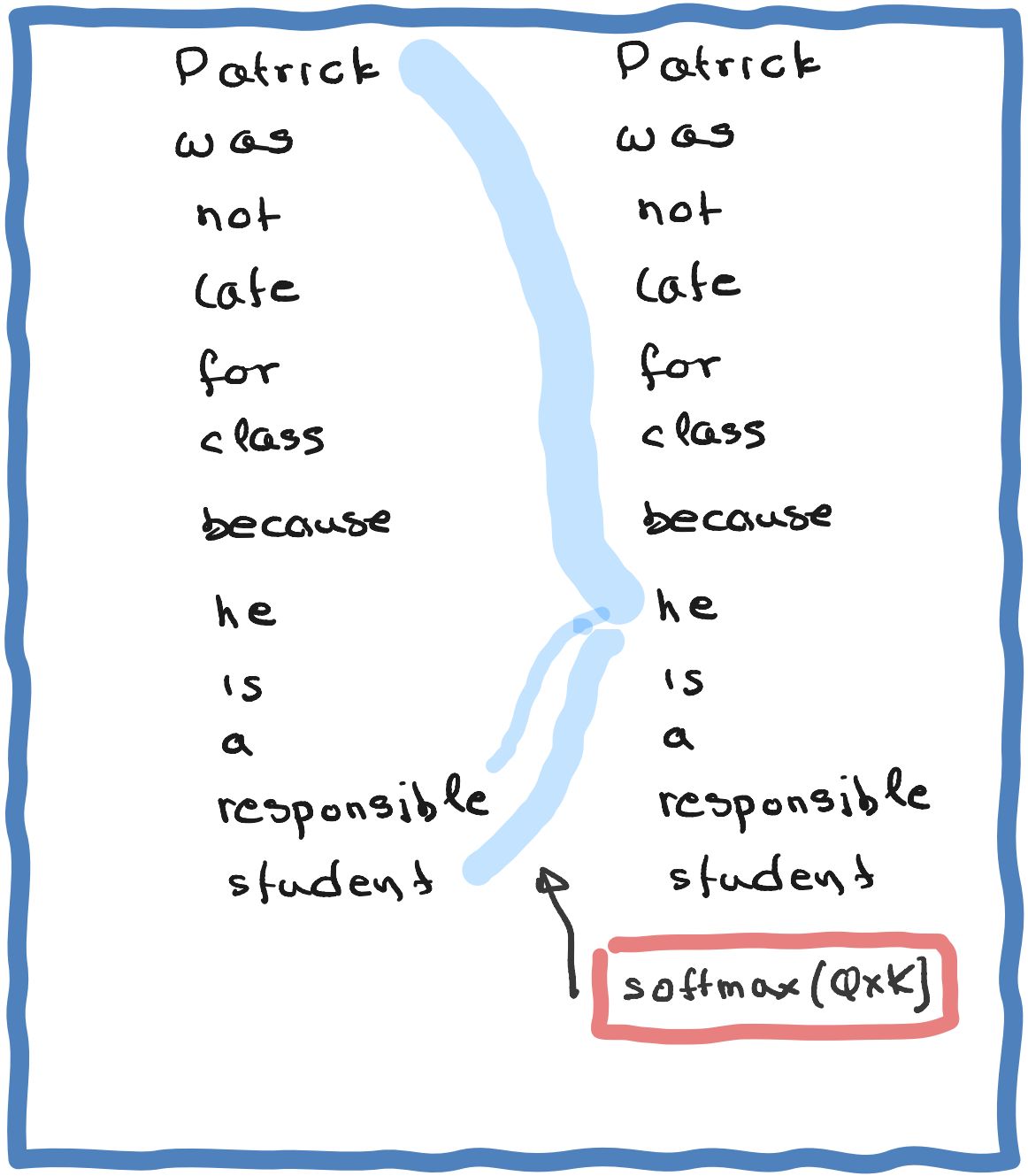
$n_w \times n_a$

In the same fashion as CNN that we need more than one filter, transformers add a mechanism called “multi-headed” attention. This improves the performance of the attention layer in two ways:

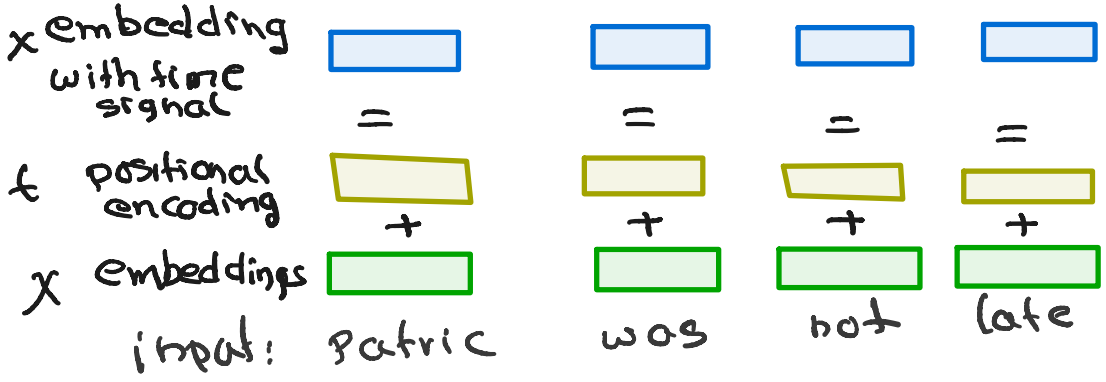
- It expands the model’s ability to focus on different positions.
- It gives the attention layer multiple “representation subspaces”



As we encode the word ”he", one attention head is focusing most on ”Patrick", while another is focusing on ”student" -- in a sense, the model's representation of the word ”he" combines the representations of both ”Patrick" and ”stduent".



Positional encoding:



$q = x \cdot W^q \qquad k = x W^k$

$q = (x+t)W^q \qquad k = (x+t)W^k$

$q \cdot k = (x_1+t_1)W^q \cdot (x_2+t_2)W^k$

$= (x_1 W^q) \cdot (x_2 W^k)$ ← attention without positional encoding

$+ \; x_1 W^q t_2 W^k$

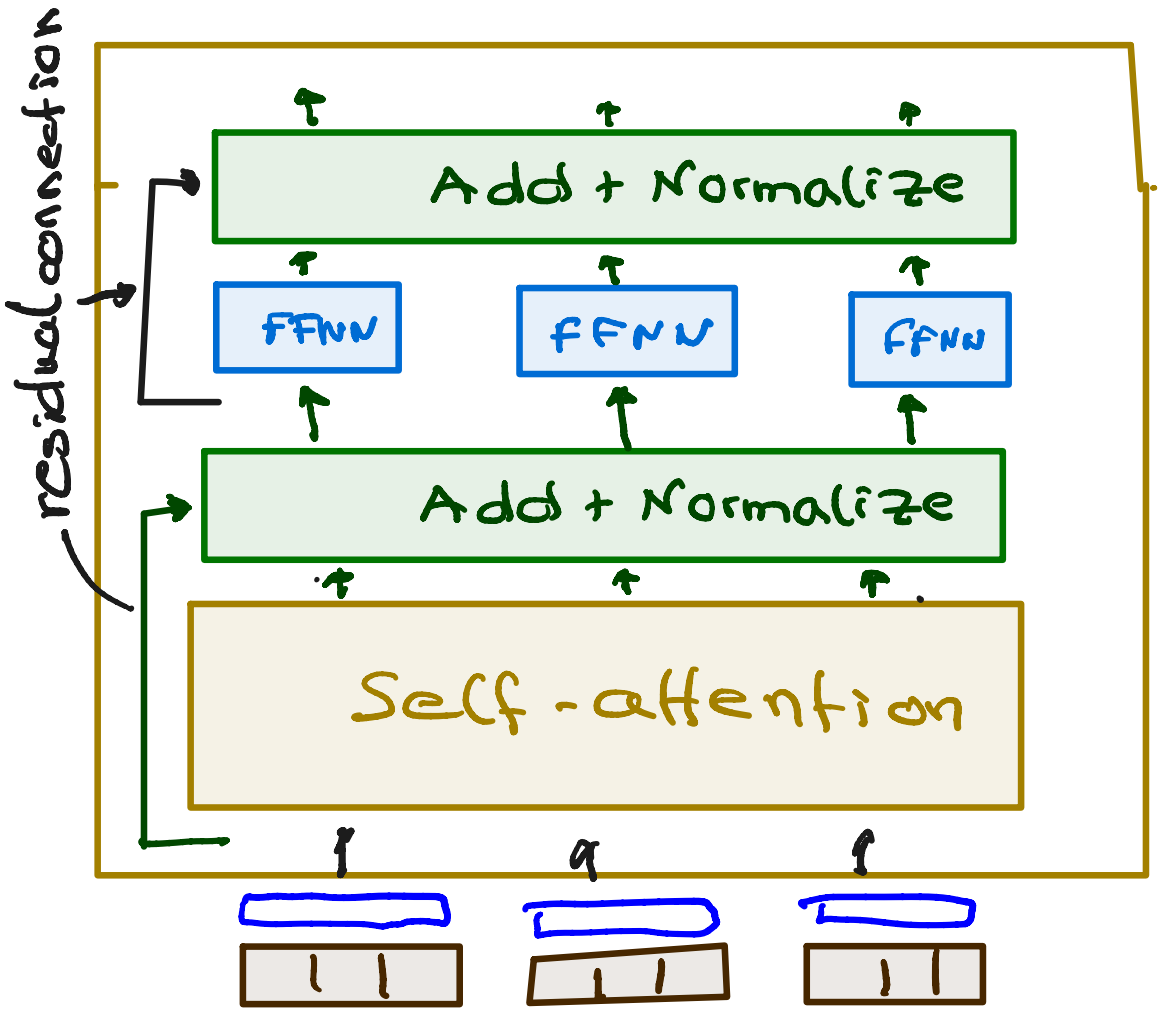
$+ \; t_1 W^q x_2 W^k$ } ?

$+ \; t_1 W^q t_2 W^k$ ← positional relationship

Details in the architecture of the encoder:

Each sub-layer in each encoder has a residual connection around it

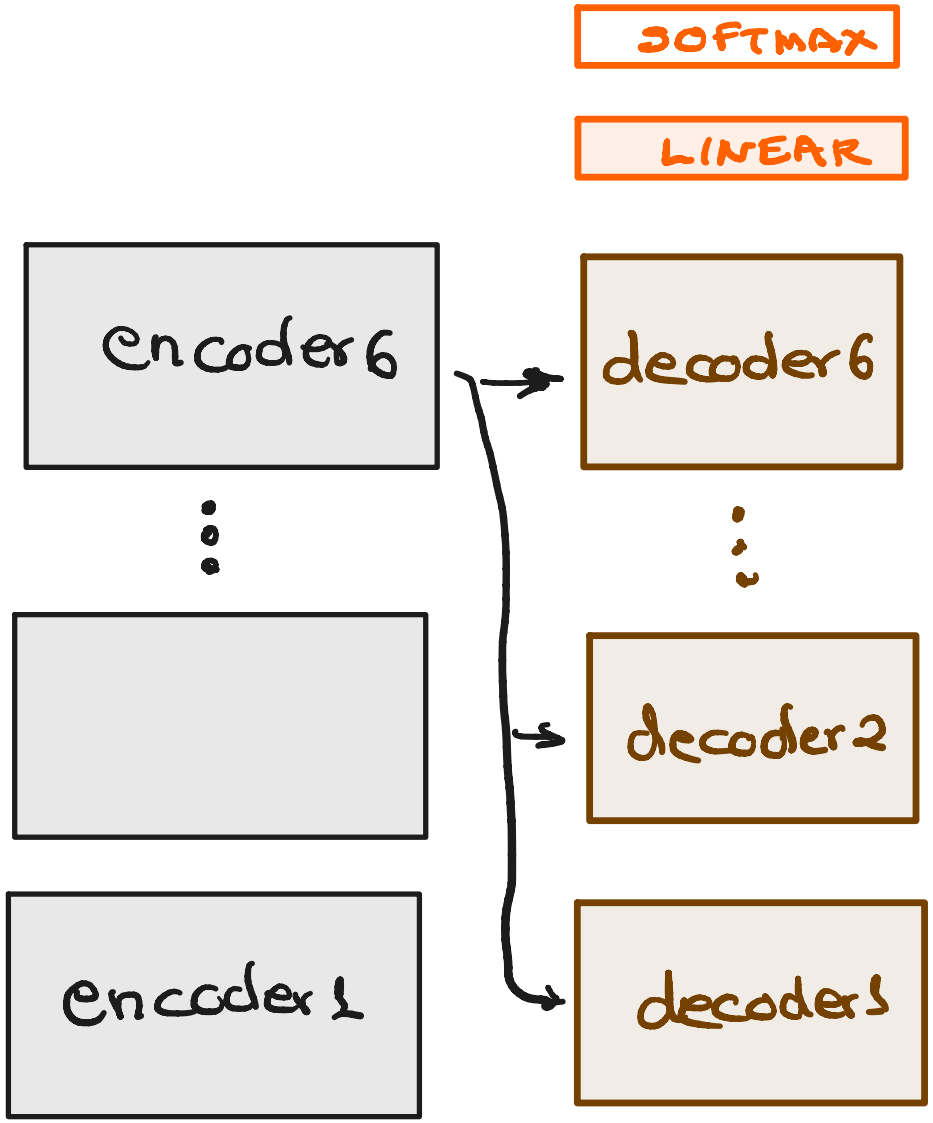
And a layer-normalization step.



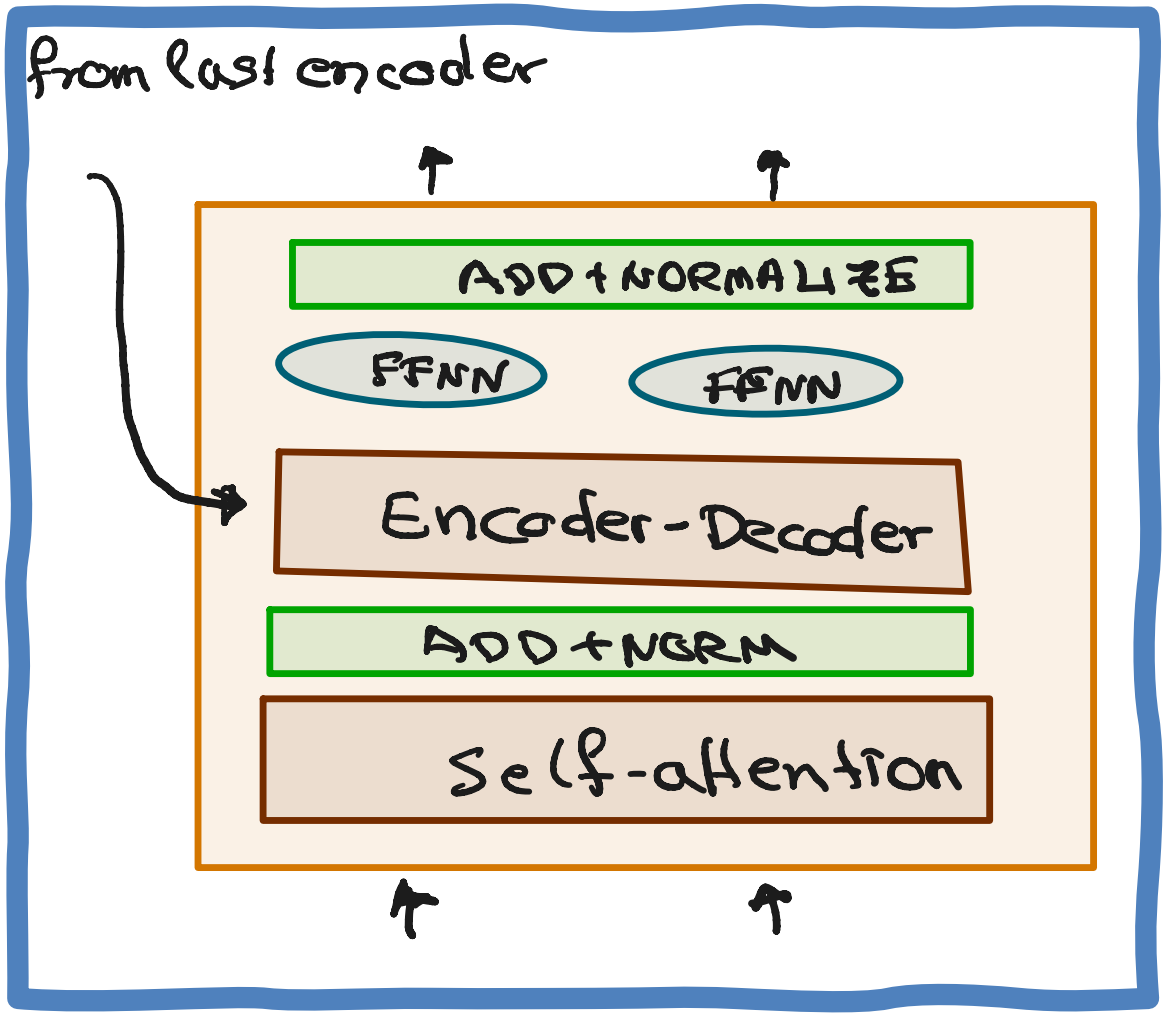
Transformer is stacked encoders and decoders.

So far:

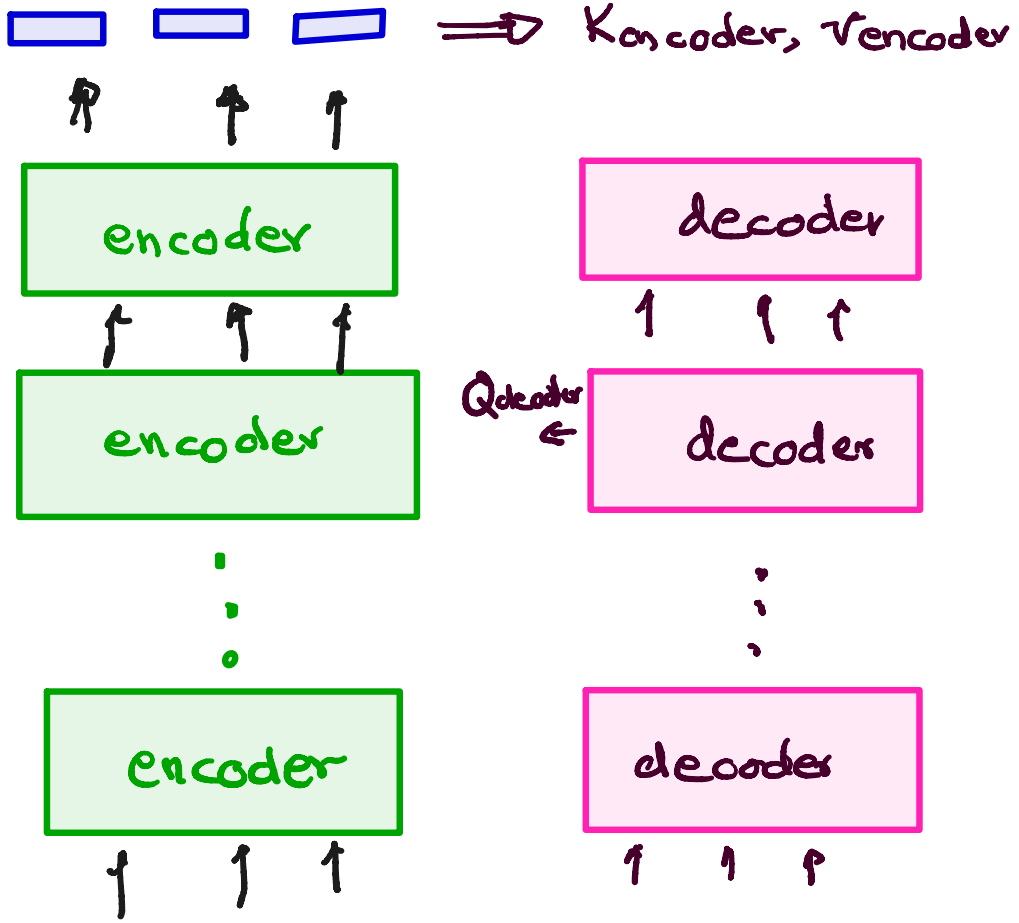
Positional encoding ☑
Encoder ☑
Self-attention ☑



decoders



encoder-decoder attention

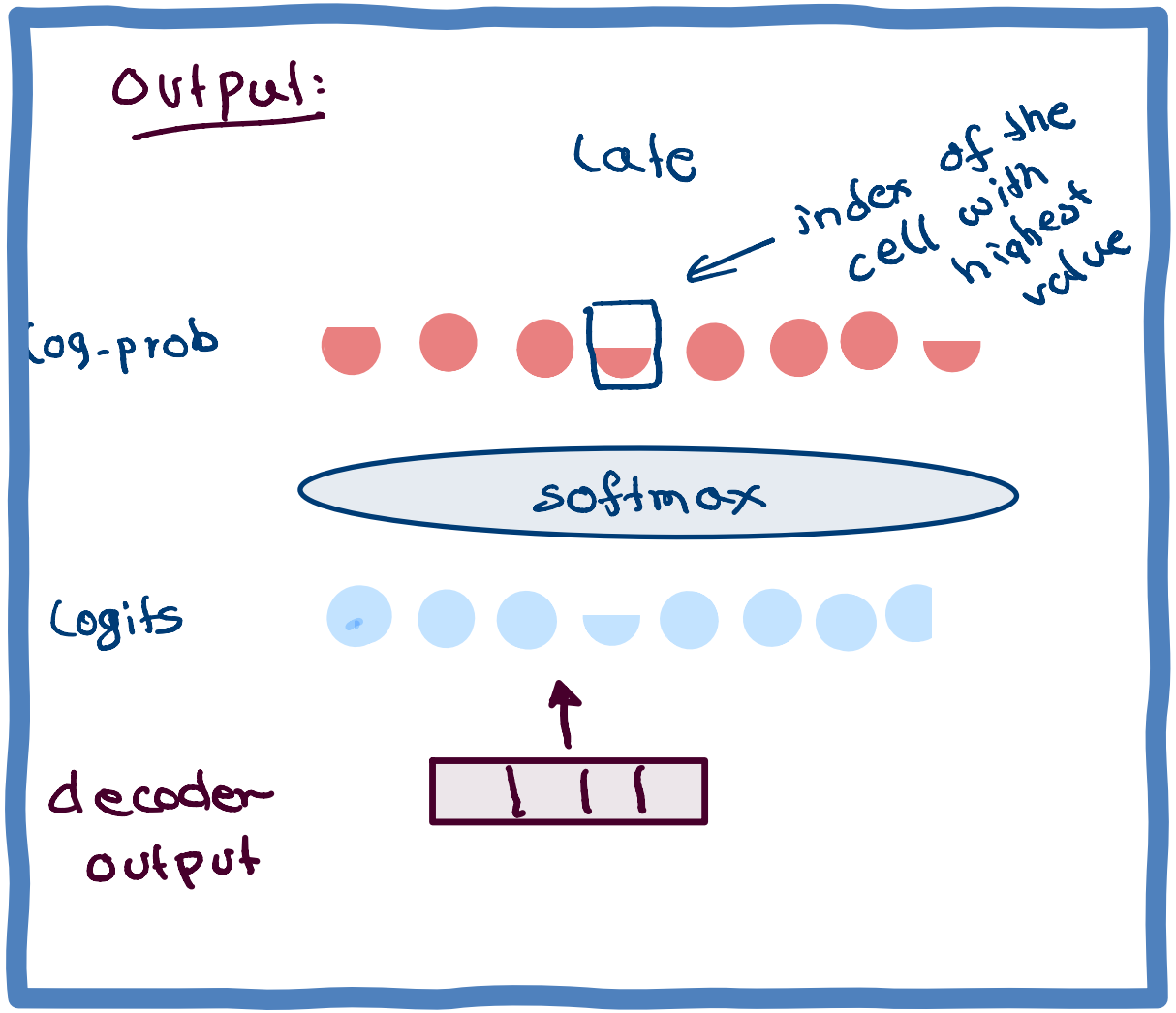


encoder-decoder attention:

$$\text{softmax}\left(Q_{decoder} \times K_{encoder}\right) V_{encoder}$$

$$Z_{decoder} \times W^{Q}_{decoder}$$

The Final Linear and Softmax Layer



training: usual CE

↳ greedy decoding

use max(·log_prob) to select $\hat{y}$ which we use for next step.

alternatively:

keep two top and try both

.... beam search

Outline

Seq2seq with attention

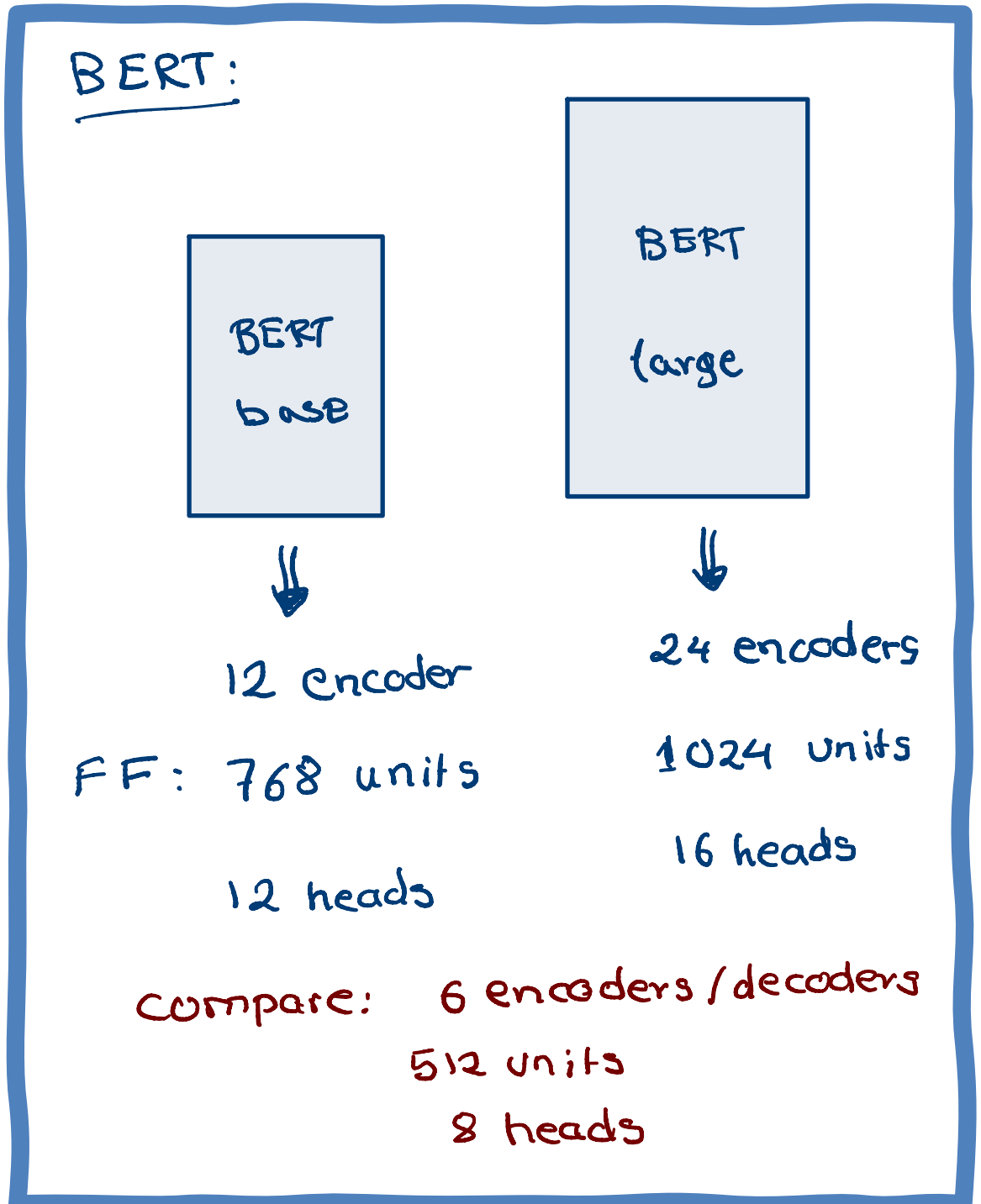
Transformers

Bert

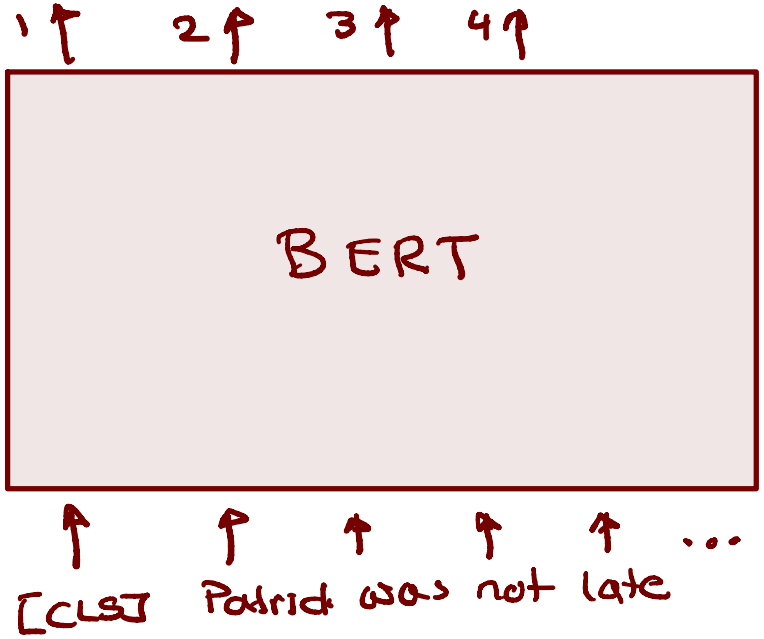
BERT

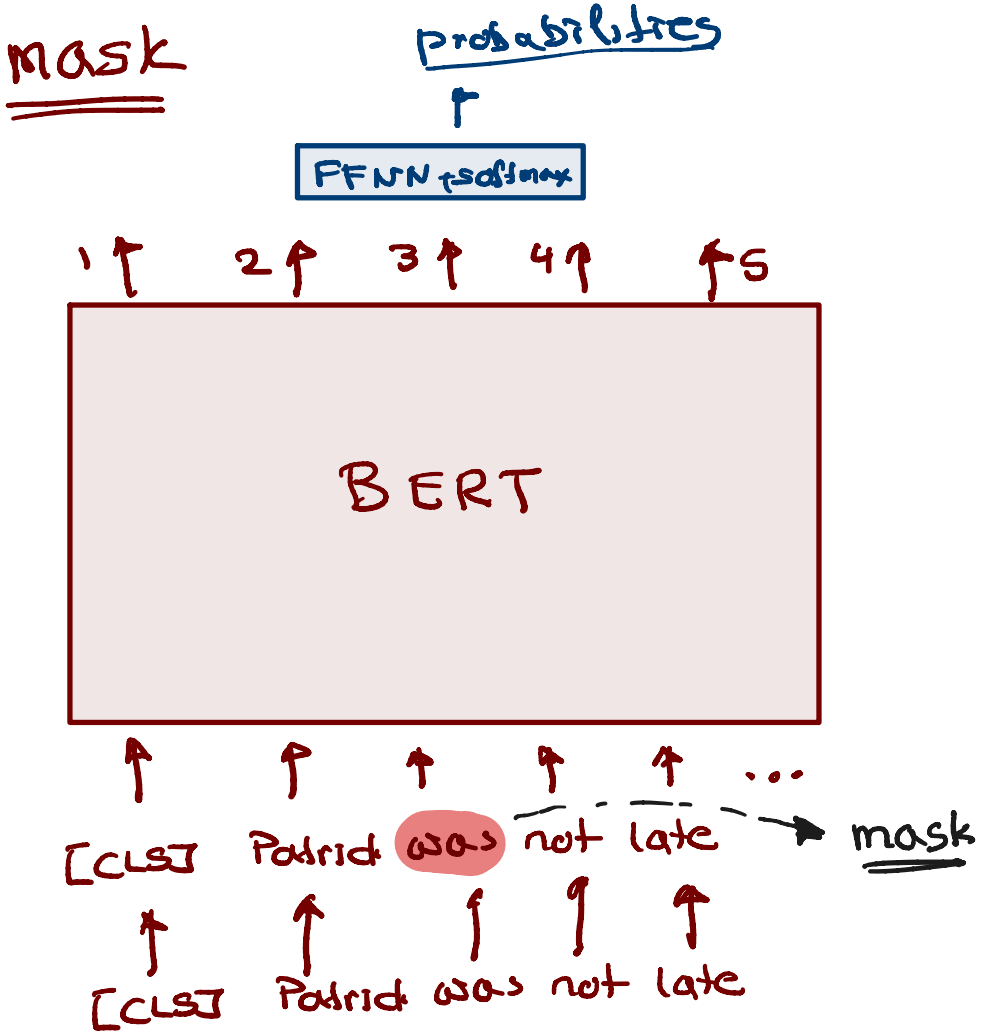
Bert stands for Bidirectional Encoder Representations from Transformers. It's google's new techniques for **NLP** pre-training language representation. Which means now machine learning communities can use Bert models that have been training already on a large number of words, for NLP models to do a wide variety of tasks such as Question Answering tasks, Named Entity Recognition (**NER**), and Classification like sentiment analysis.

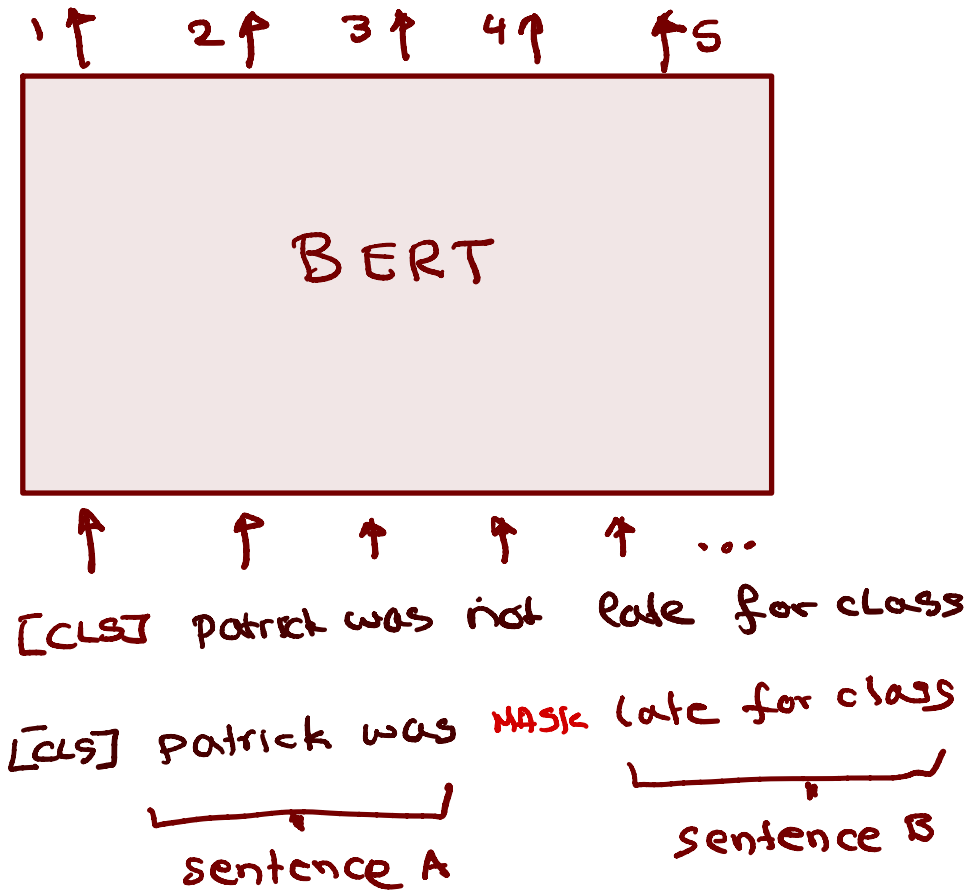
In Bert paper, they present two types of Bert models one is the **Bert Base** and the other is **Bert Large.**



What is the input of Bert?
The input of Bert is a special input start with [CLS] token stand for classification. As in the Transformers, Bert will take a sequence of words (vector) as an input that keeps feed up from the first encoder layer up to the last layer in the stack. Each layer in the stack will apply the self-attention method to the sequence after that it will pass to the feed-forward network to deliver the next encoder layer.






1
2
3
4
5
BERT
[CLS] patrick was not late for class
[CLS] patrick was
MASK
late for class
sentence A
sentence B