Attentional Seq2seq model

Recurrent neurone

- **Most significant change: new set of weights, U**
    - connect the hidden layer from the previous time step to the current hidden layer.
    - determine how the network should make use of past context in calculating the output for the current input.

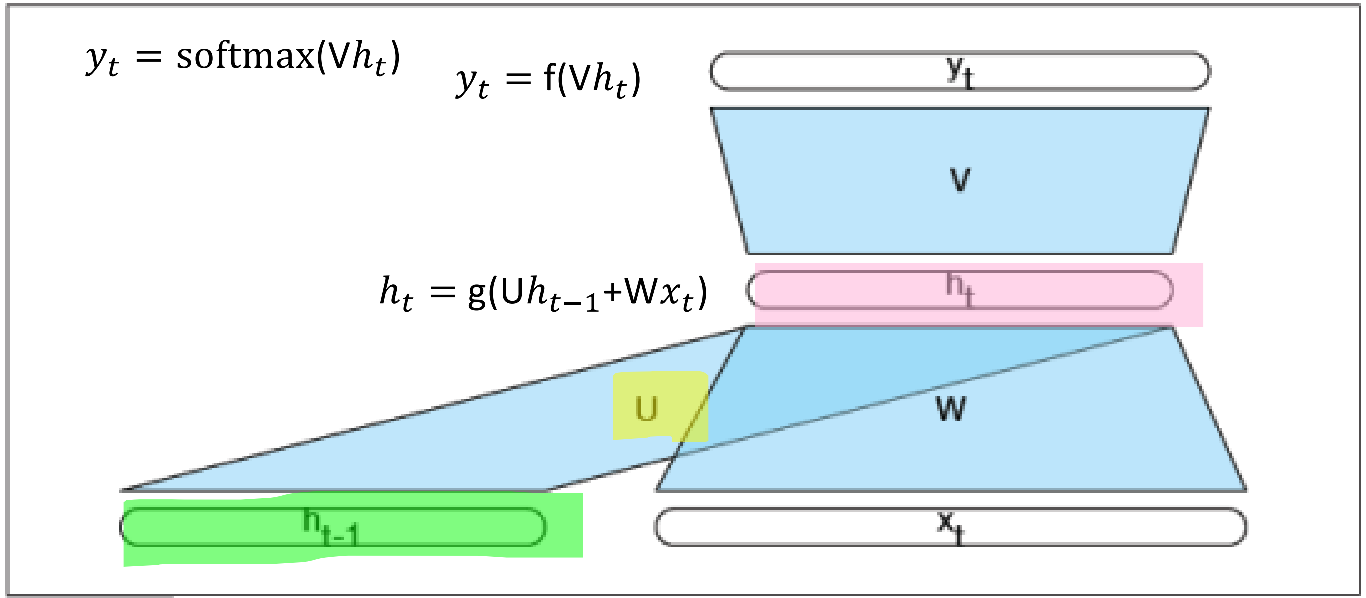
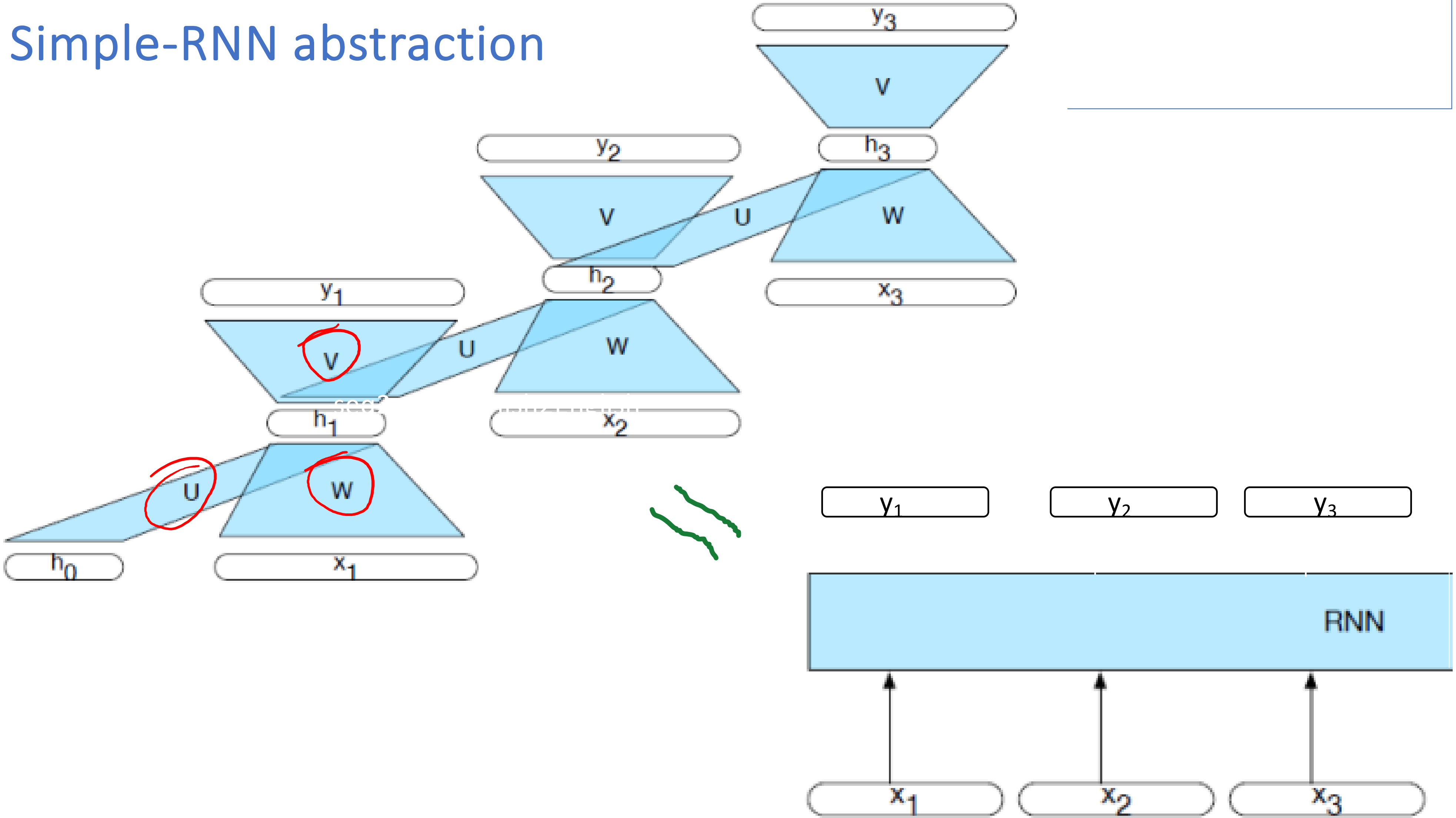


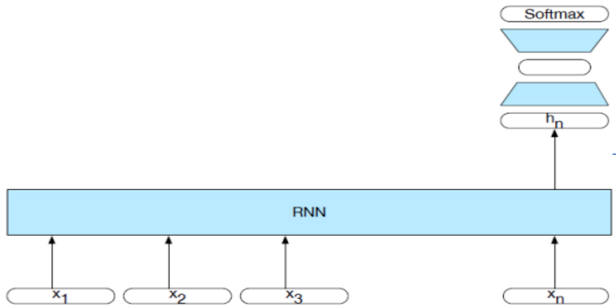
Figure 9.3 Simple recurrent neural network illustrated as a feed-forward network.

Simple-RNN abstraction

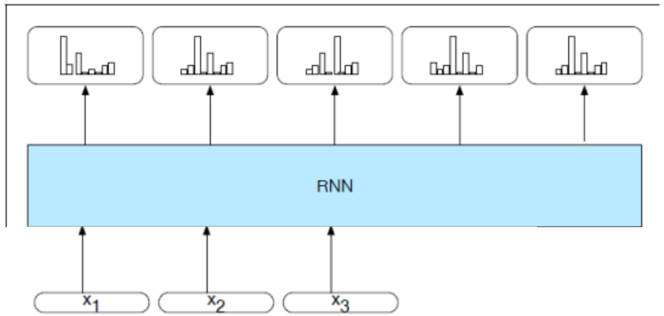


RNN Applications

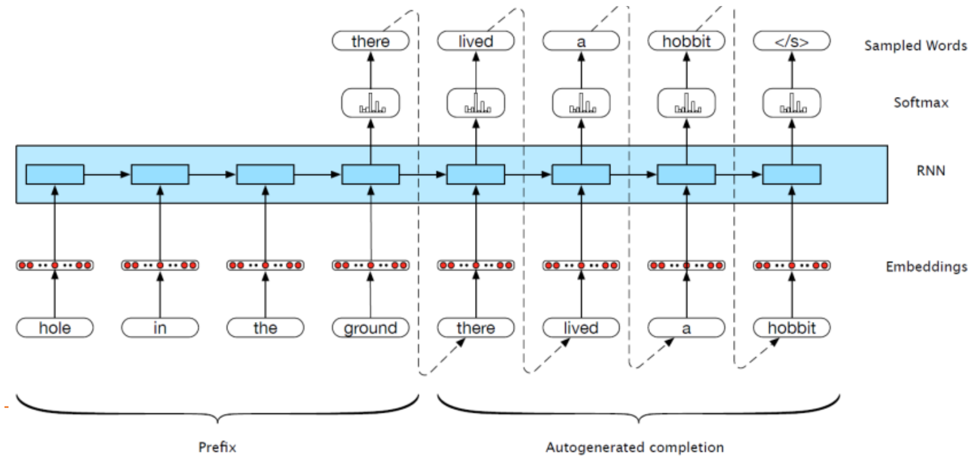
- Sequence Classification (Sentiment, Topic)



- POS, NER, Language Modeling



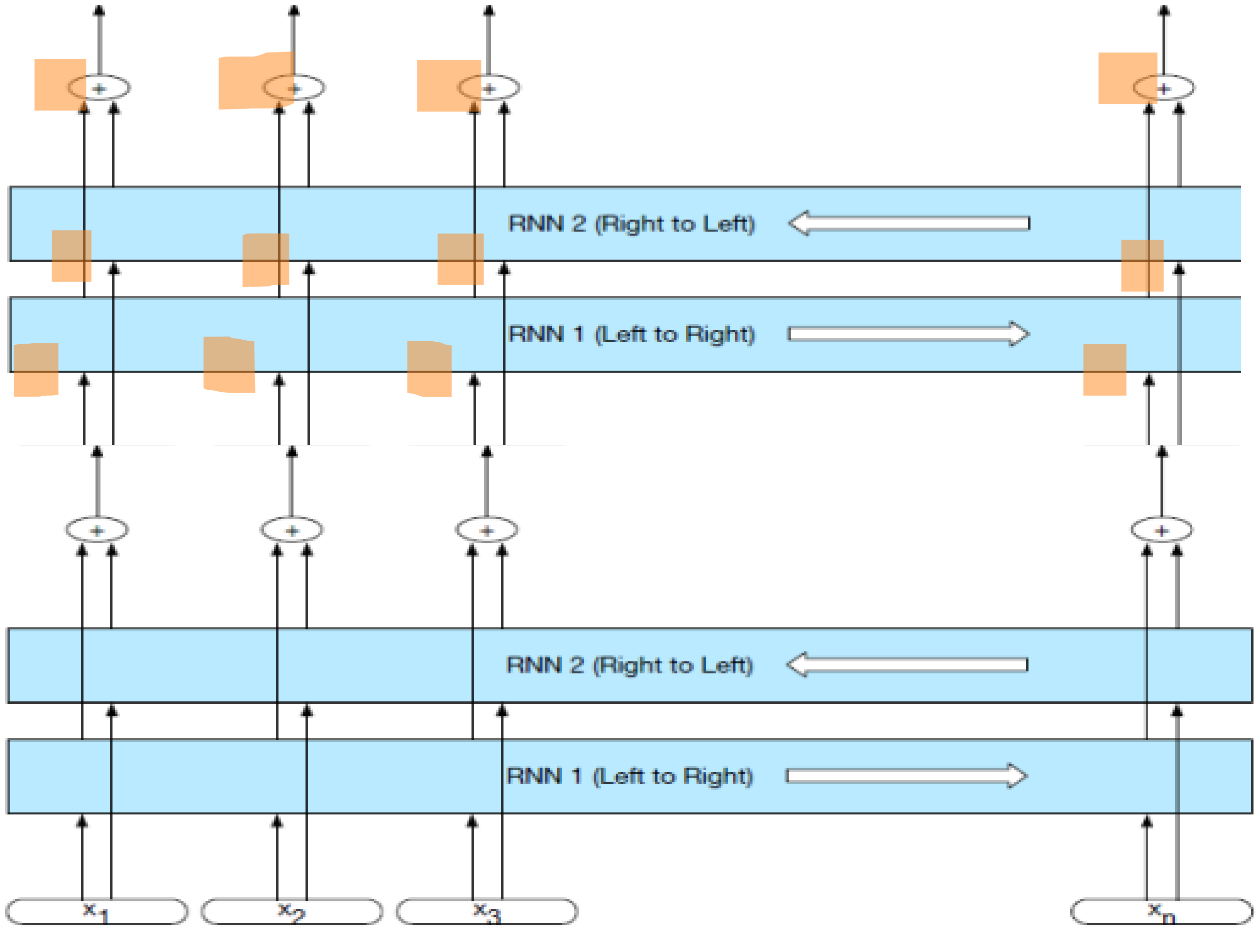
- Sequence to Sequence
  - Machine translation
  - Question answering



Popular architectural choices: Encoder

Widely used encoder design: **stacked Bi-LSTMs**

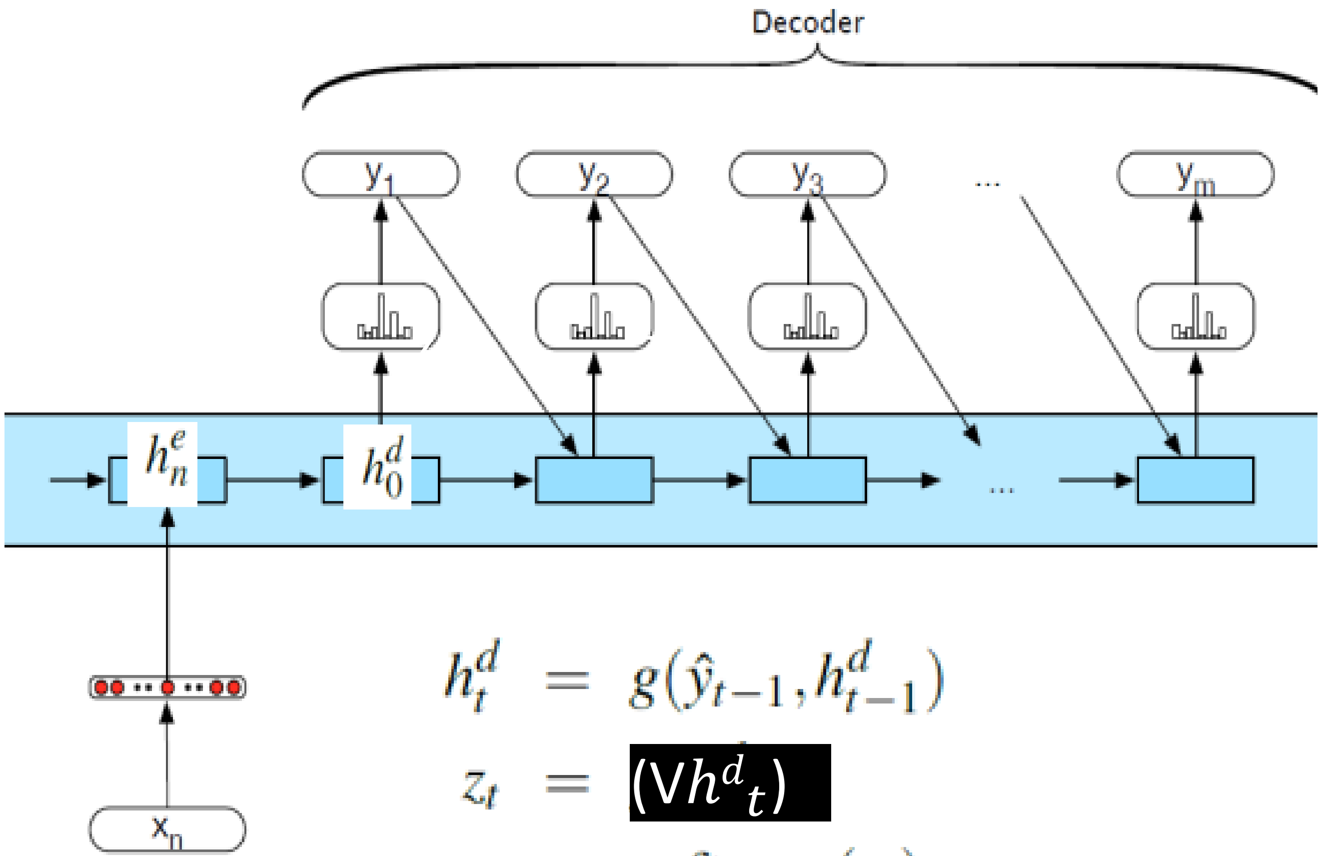
- Contextualized representations for each time step: **hidden states from top layers** from the forward and backward passes



Decoder Basic Design

- produce an output sequence an element at a time

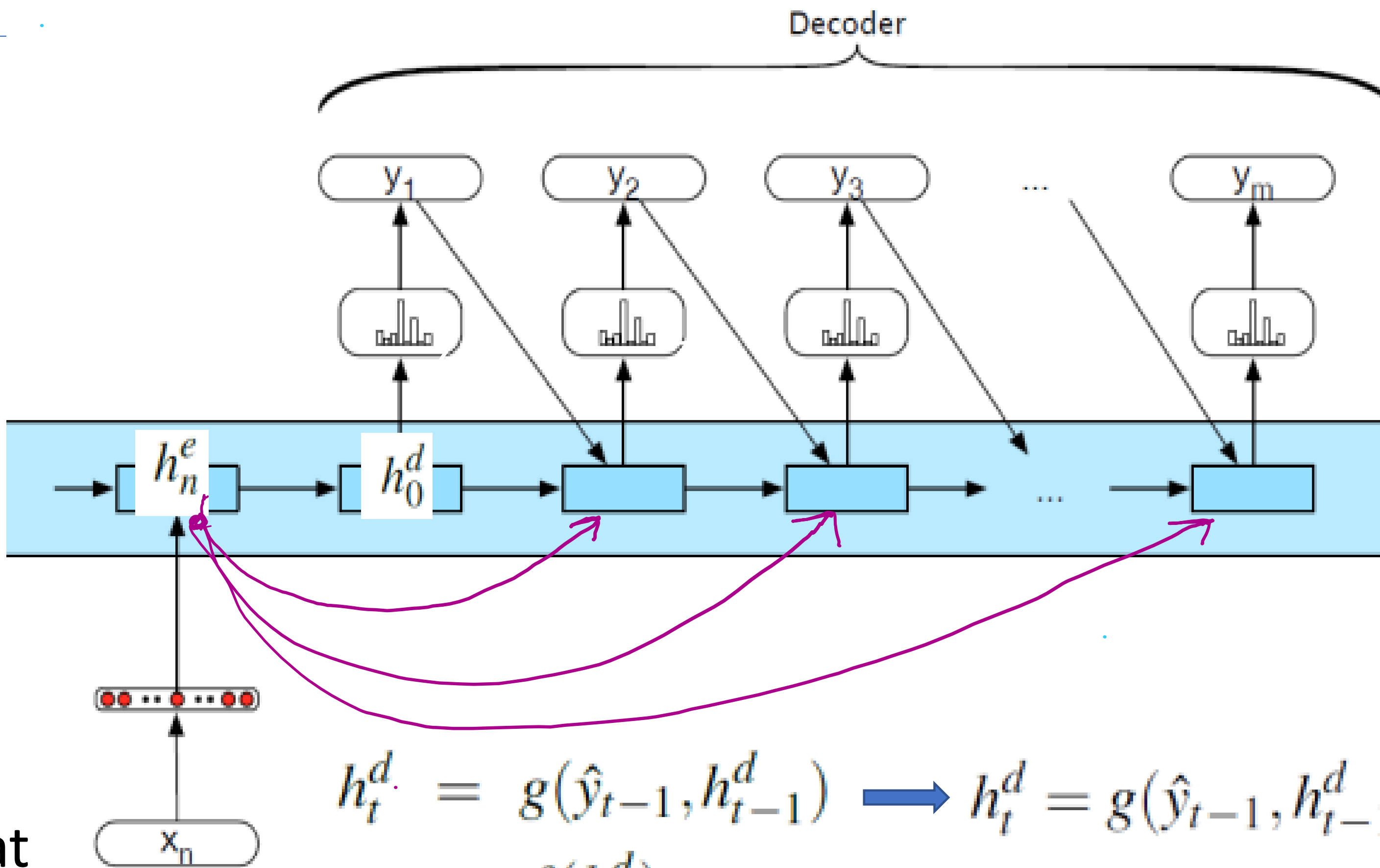
$$c = h_n^e$$
$$h_0^d = c$$



$$h_t^d = g(\hat{y}_{t-1}, h_{t-1}^d)$$
$$z_t = (Vh^d{}_t)$$
$$y_t = \text{softmax}(z_t)$$

Decoder Design Enhancement

$$c = h_n^e$$
$$h_0^d = c$$



Context available at each step of decoding

$$h_t^d = g(\hat{y}_{t-1}, h_{t-1}^d) \Rightarrow h_t^d = g(\hat{y}_{t-1}, h_{t-1}^d, c)$$
$$z_t = f(h_t^d)$$
$$y_t = \text{softmax}(z_t)$$

From Sequence-to-Sequence (seq2seq) to Attention

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

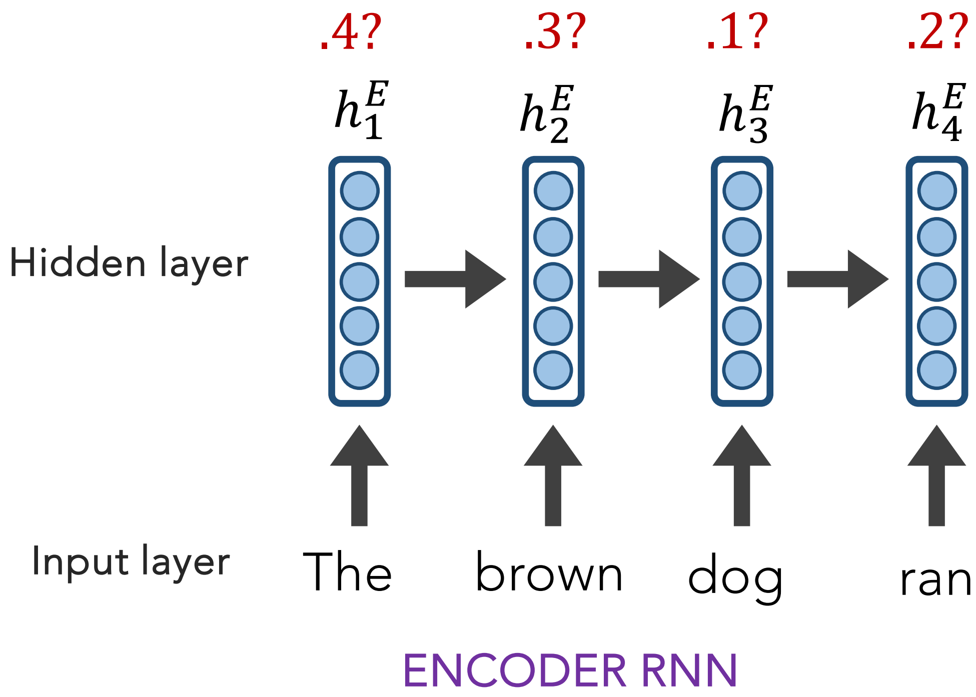
- **Intuition:** when we (humans) translate a sentence, we don't just consume the original sentence, reflect on the meaning of the last word, then regurgitate in a new language; we continuously think back at the original sentence while focusing on different parts.

- The concept of attention within cognitive neuroscience and psychology dates back to the 1800s. [William James, 1890].

- Nadaray-Watson kernel regression proposed in 1964. It *locally weighted* its predictions.

seq2seq + Attention

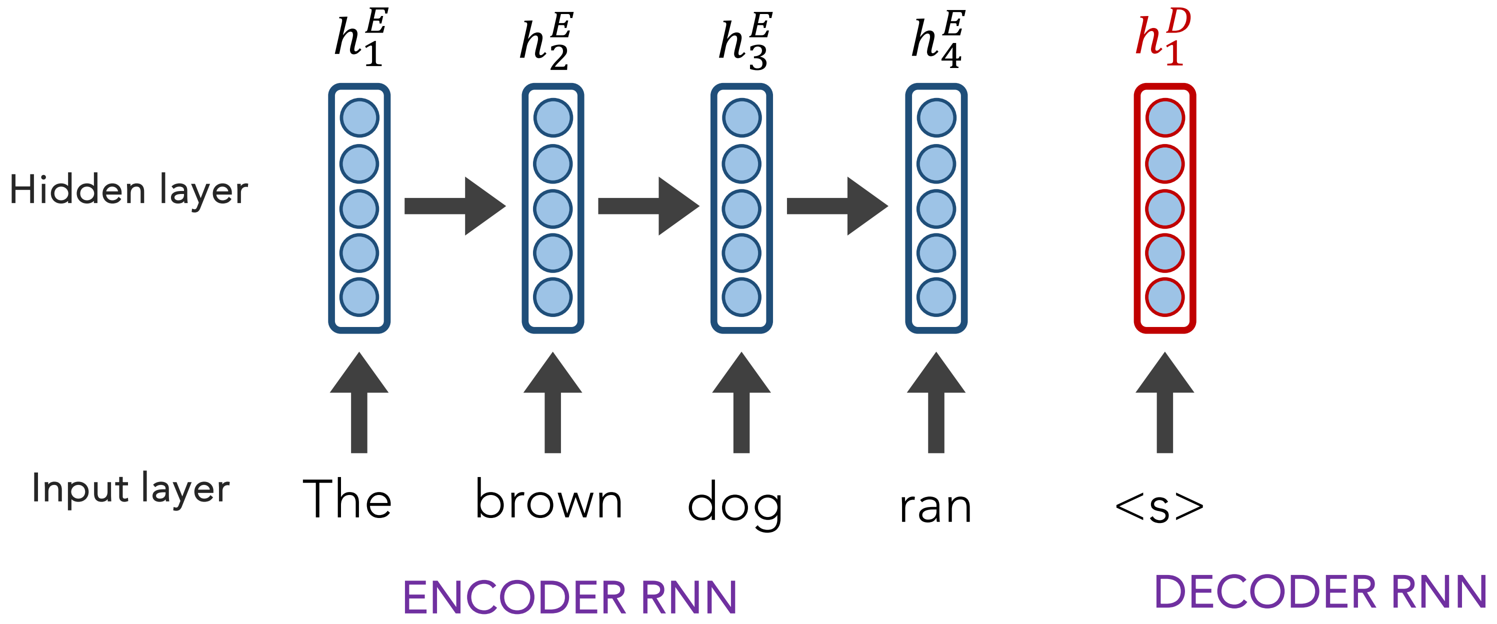
Q: How do we determine how much to pay attention to each of the encoder's hidden layers?



seq2seq + Attention

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

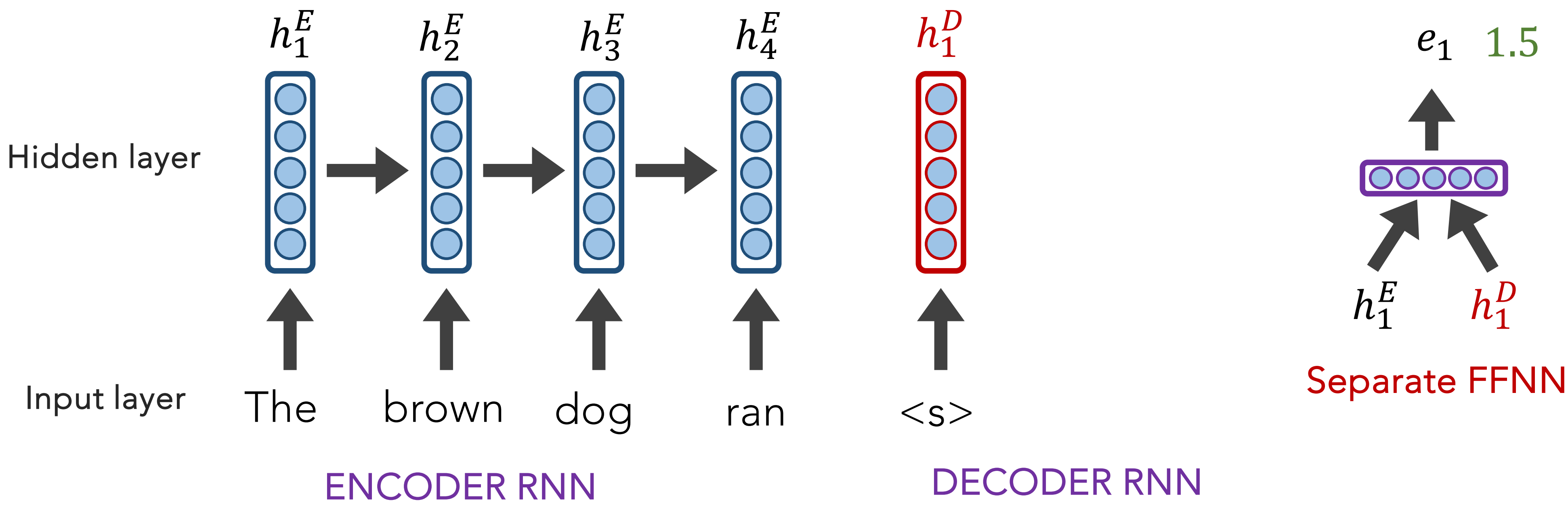
A: Let's base it on our decoder's current hidden state (our current representation of meaning) and all of the encoder's hidden layers!



seq2seq + Attention

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

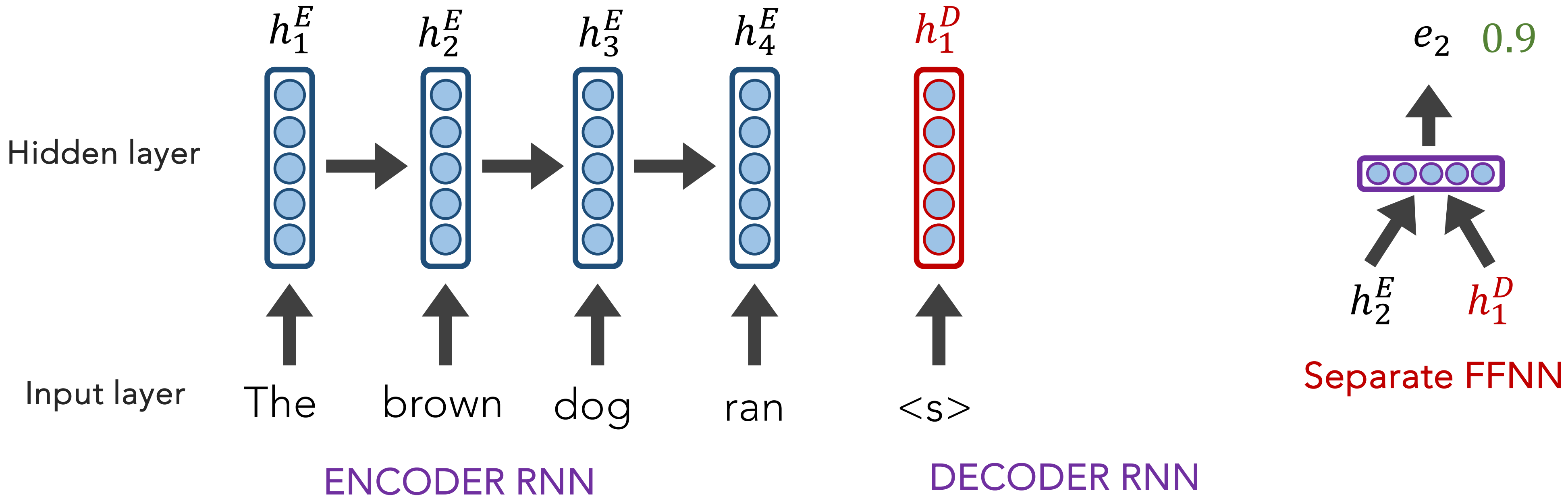
A: Let's base it on our decoder's current hidden state (our current representation of meaning) and all of the encoder's hidden layers!



seq2seq + Attention

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

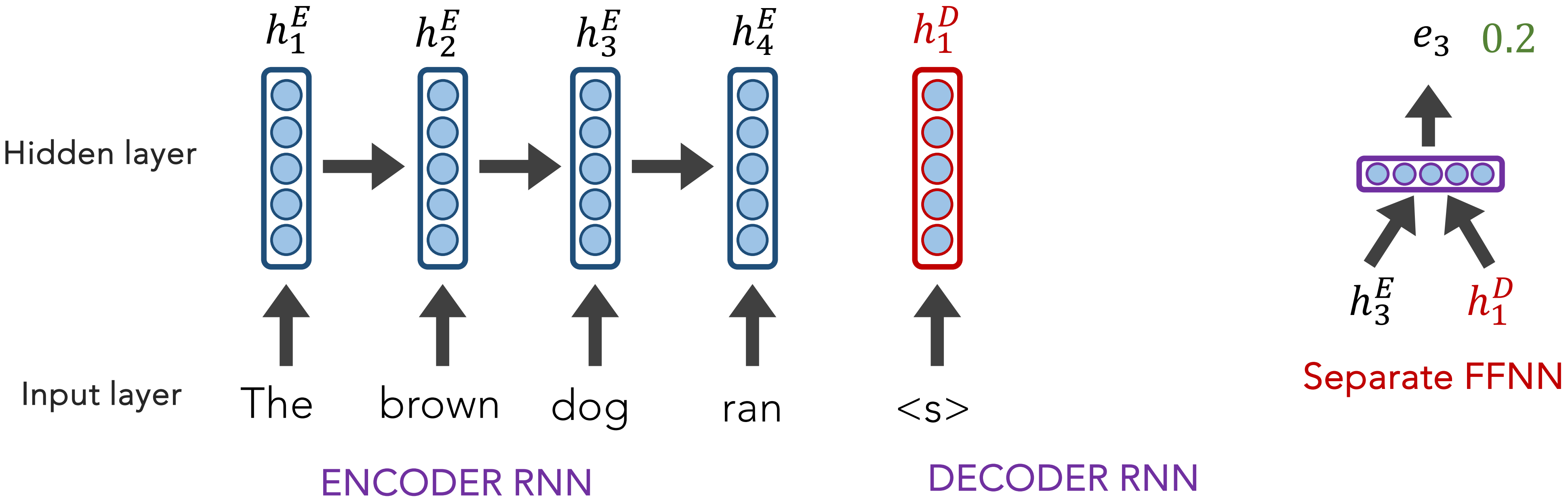
A: Let's base it on our decoder's current hidden state (our current representation of meaning) and all of the encoder's hidden layers!



seq2seq + Attention

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

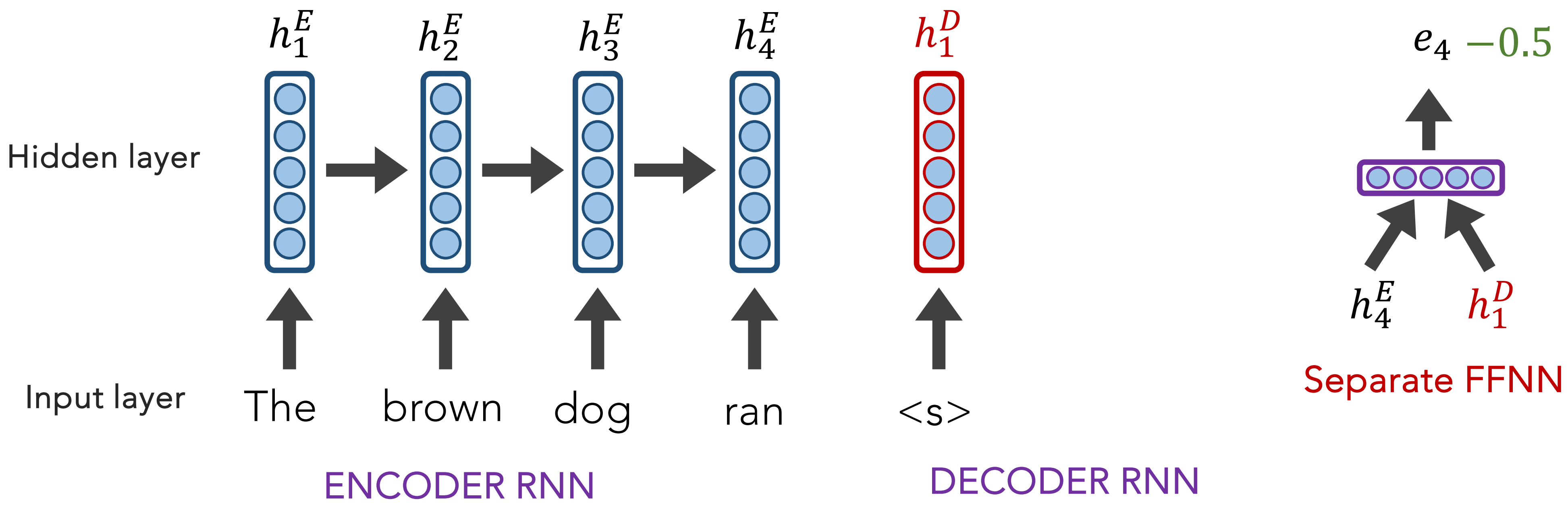
A: Let's base it on our decoder's current hidden state (our current representation of meaning) and all of the encoder's hidden layers!



seq2seq + Attention

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

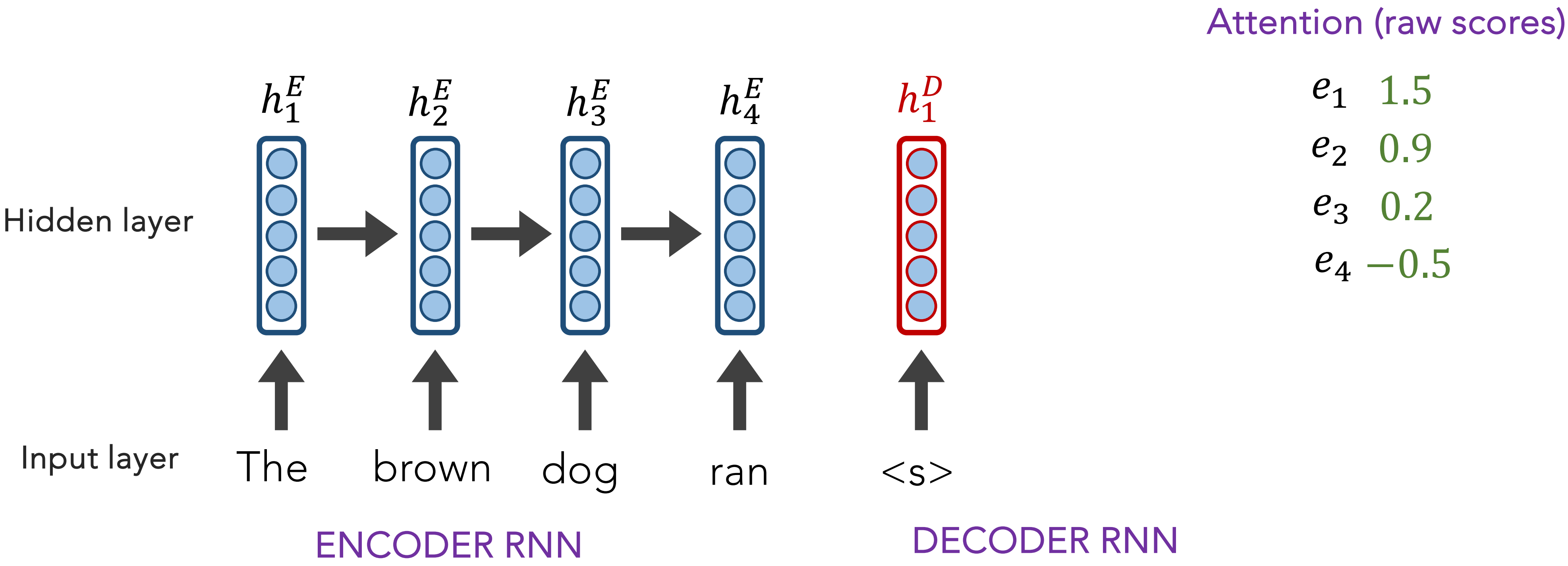
A: Let's base it on our decoder's current hidden state (our current representation of meaning) and all of the encoder's hidden layers!



seq2seq + Attention

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

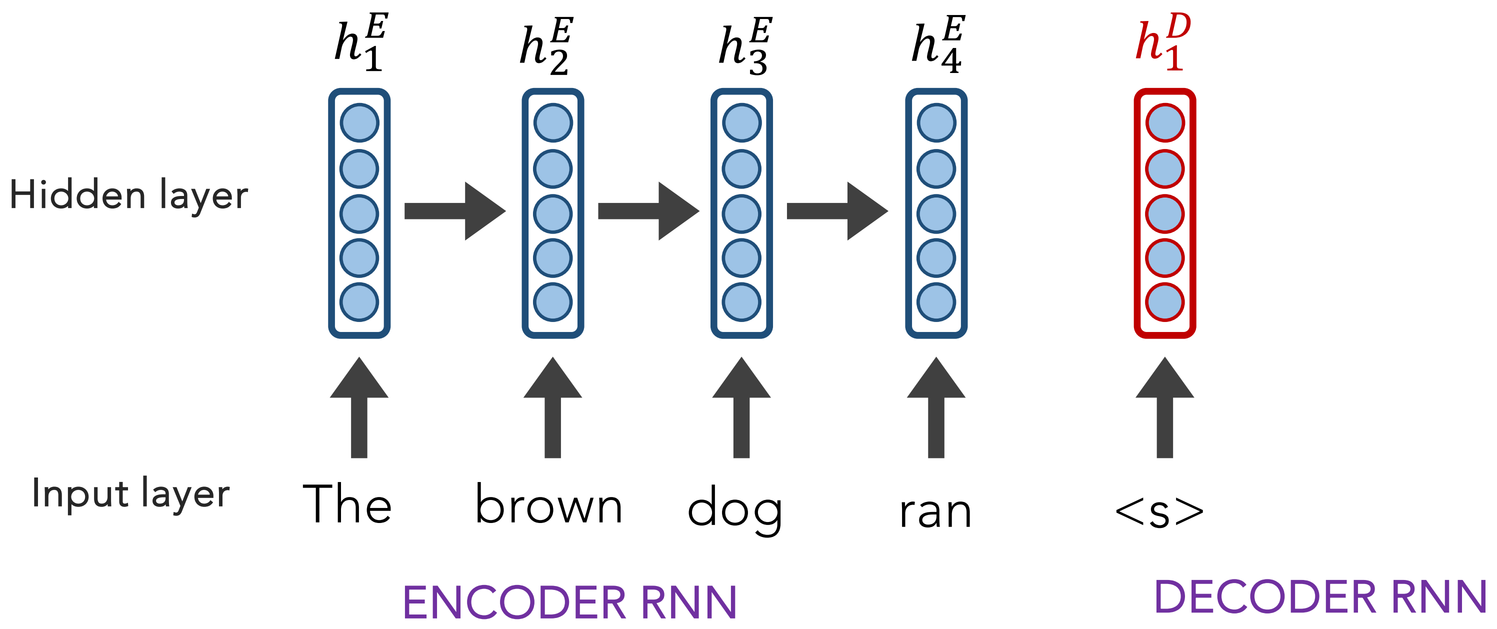
A: Let's base it on our decoder's current hidden state (our current representation of meaning) and all of the encoder's hidden layers!



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 current hidden state (our current 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

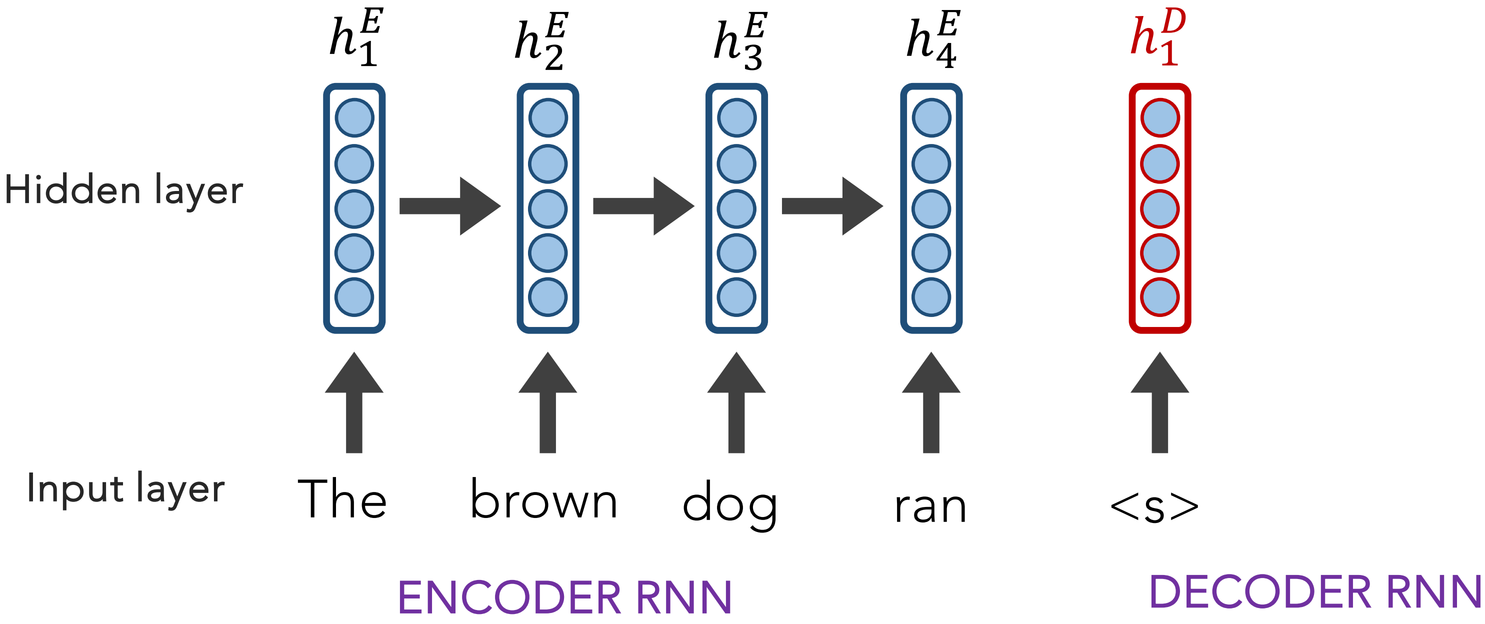
Attention (softmax'd)

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

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 current hidden state (our current 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)

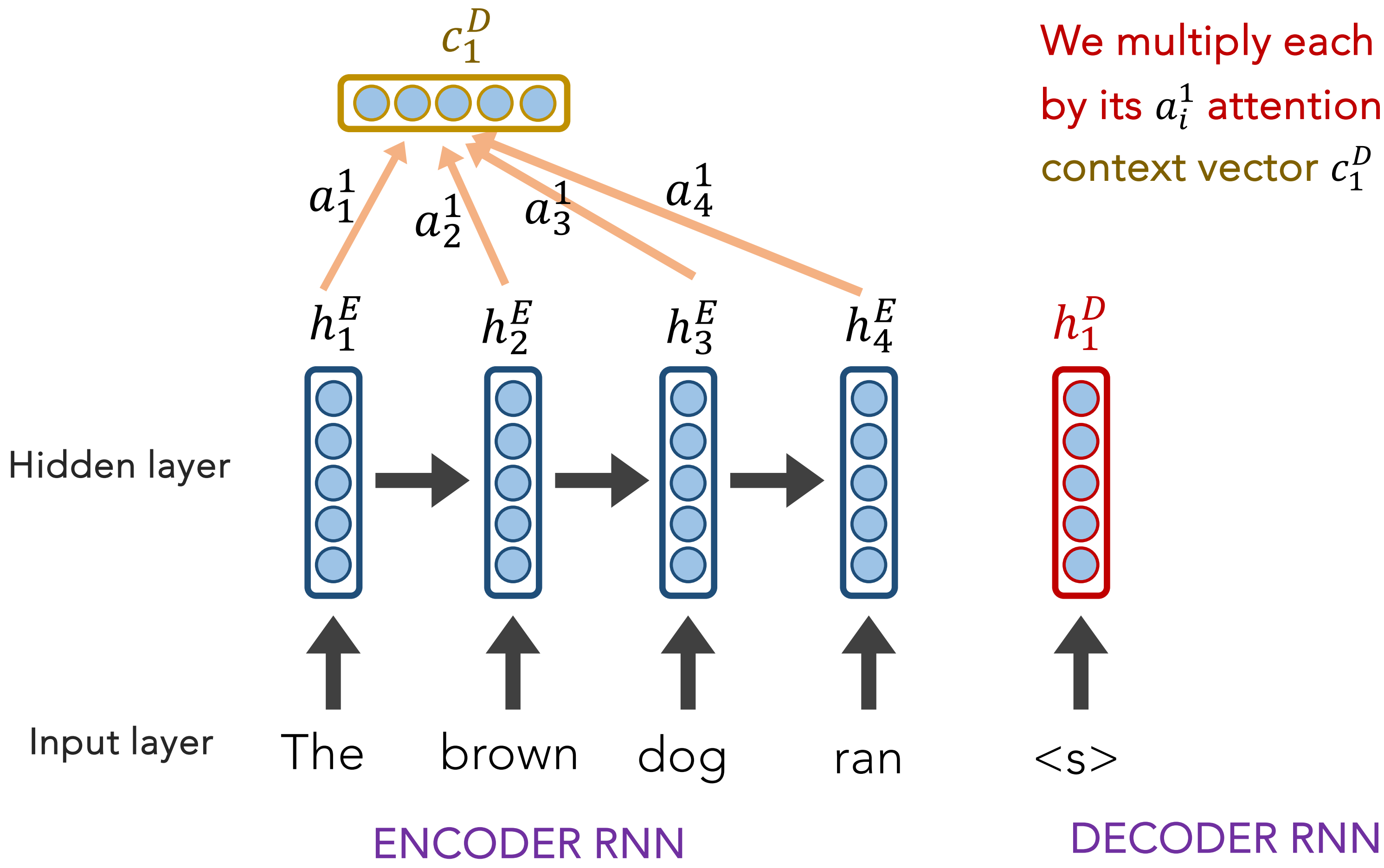
$a_1^1 = 0.51$

$a_2^1 = 0.28$

$a_3^1 = 0.14$

$a_3^1 = 0.07$

seq2seq + Attention



We multiply each encoder's hidden layer by its a_i^1 attention weights to create a context vector c_1^D

Attention (softmax'd)

$a_1^1 = 0.51$

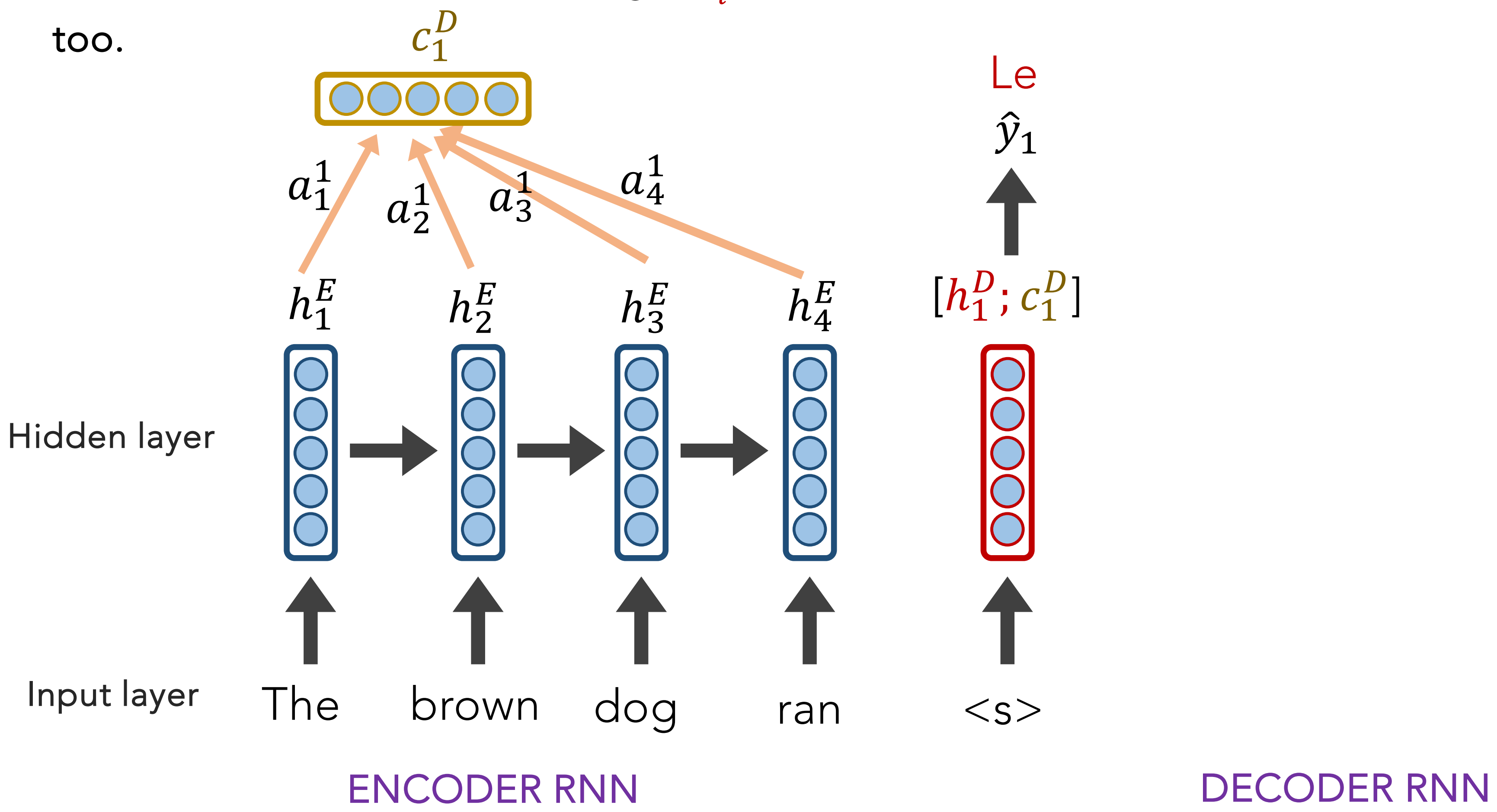
$a_2^1 = 0.28$

$a_3^1 = 0.14$

$a_3^1 = 0.07$

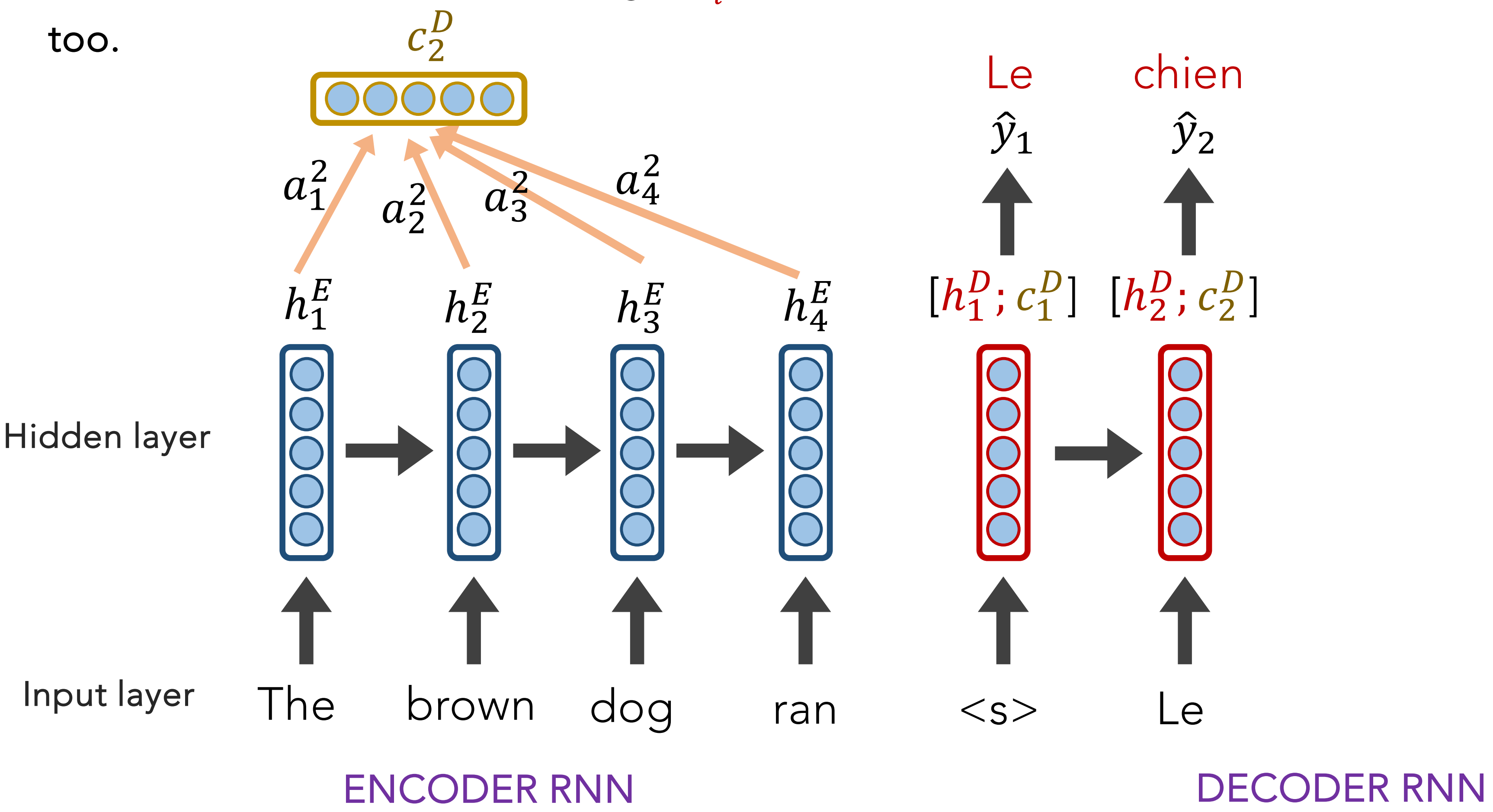
seq2seq + Attention

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



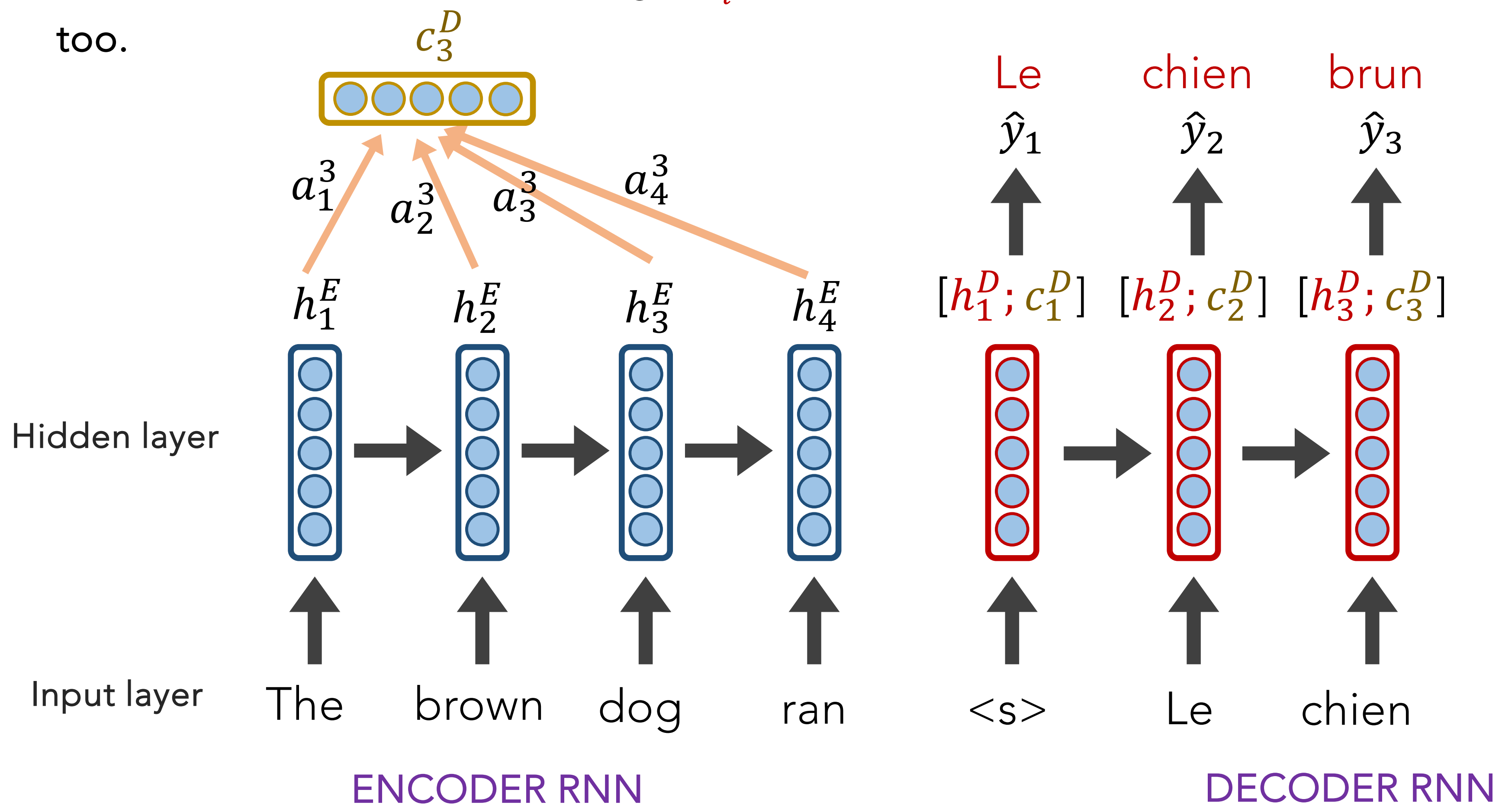
seq2seq + Attention

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



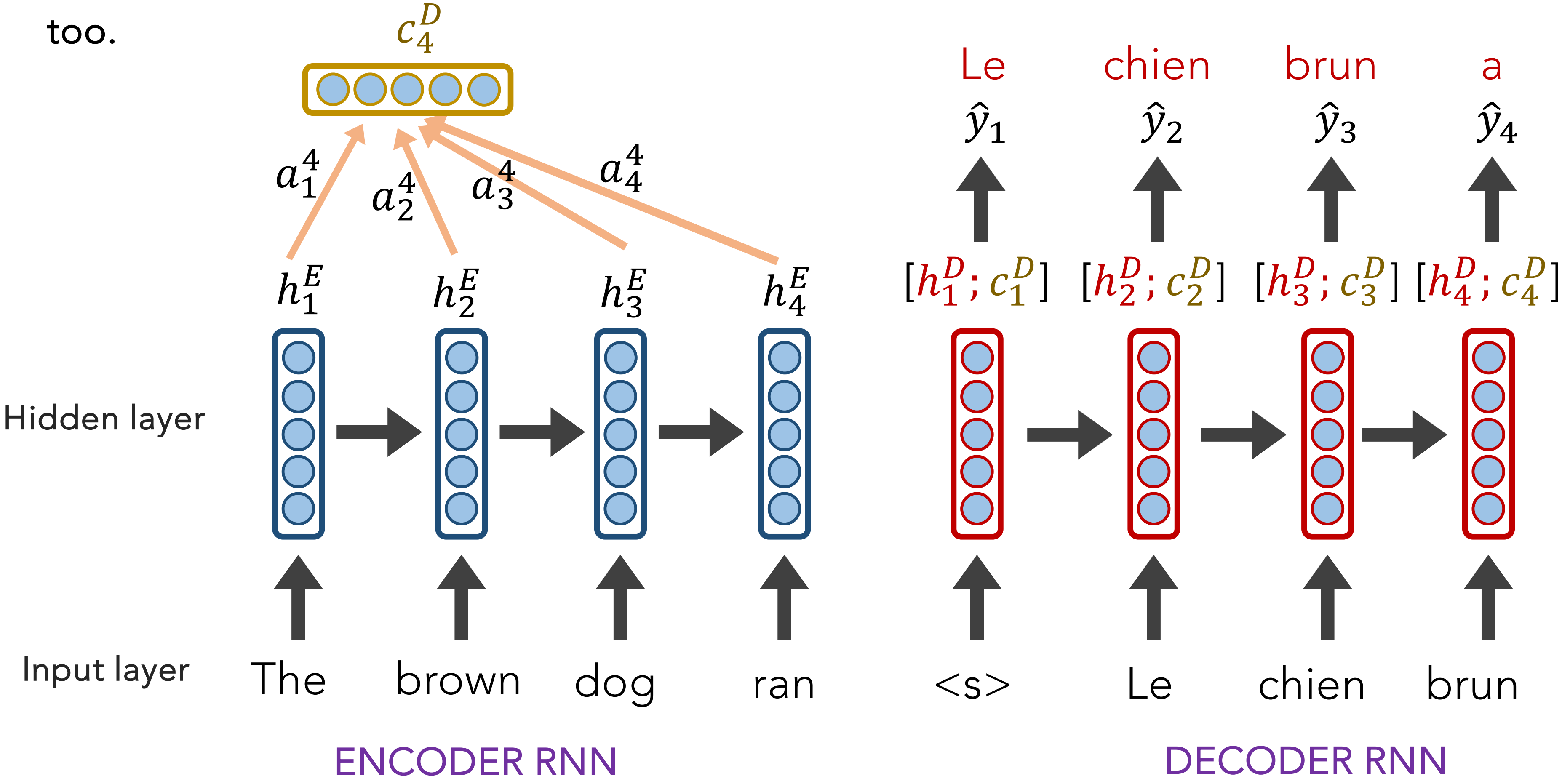
seq2seq + Attention

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



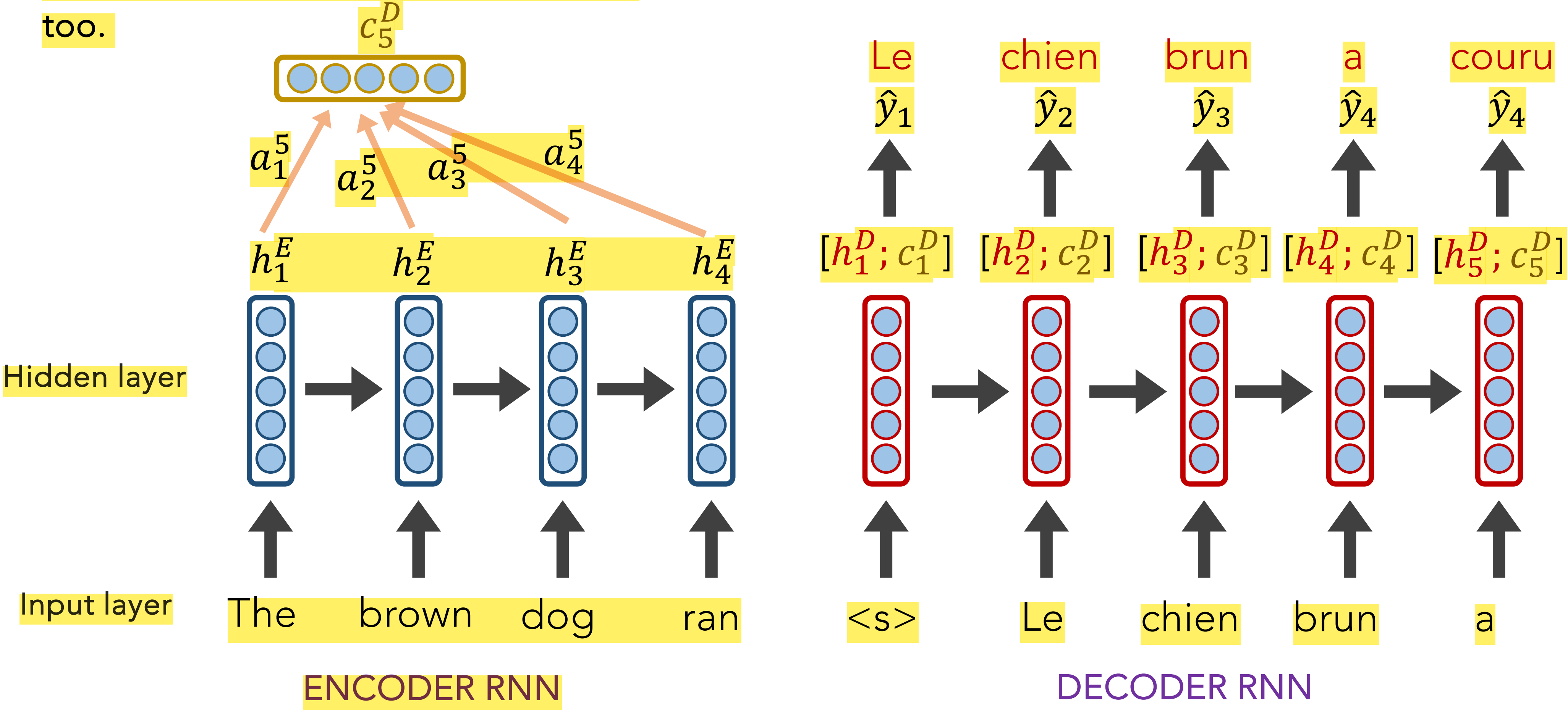
seq2seq + Attention

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

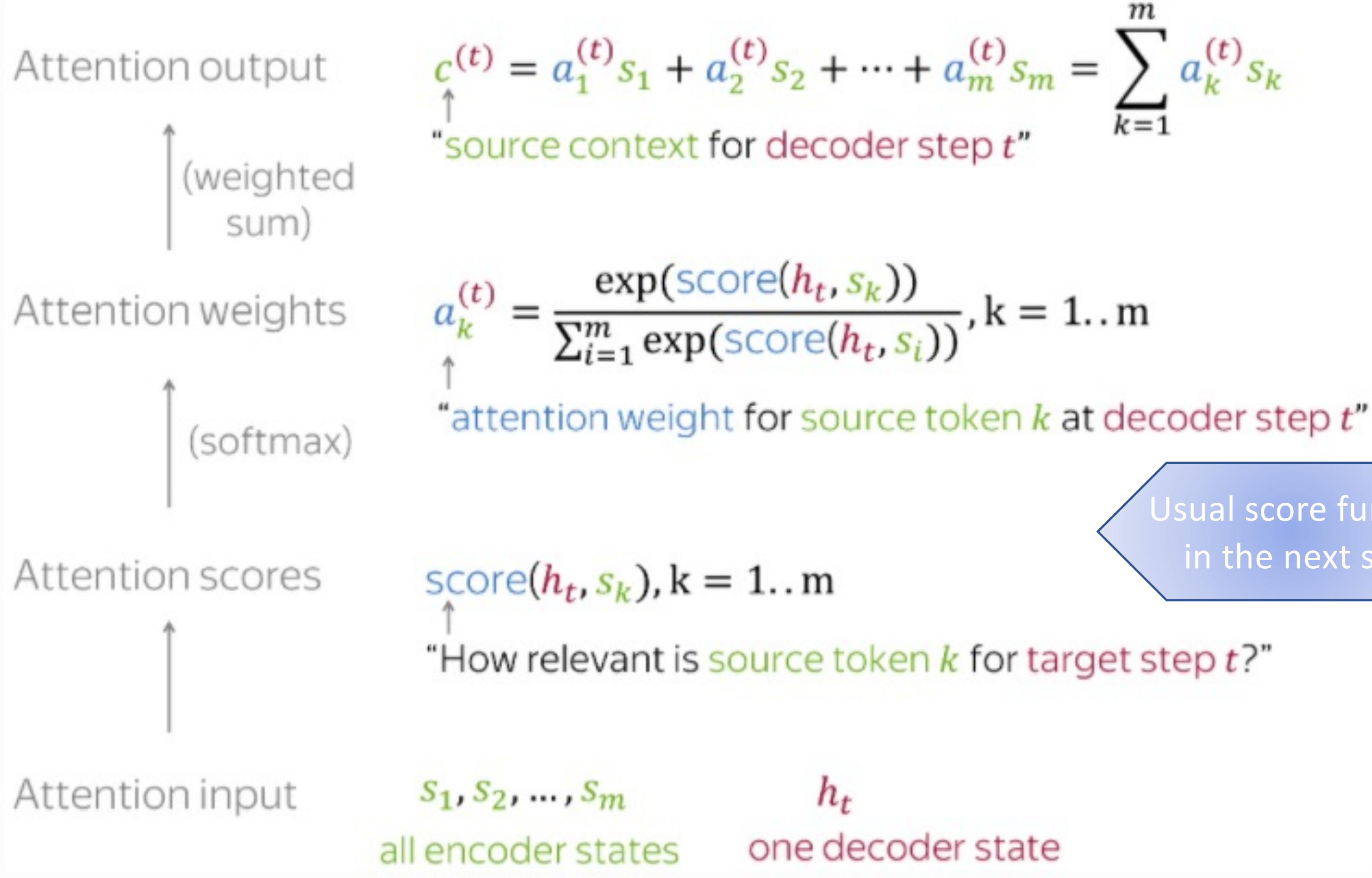


seq2seq + Attention

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

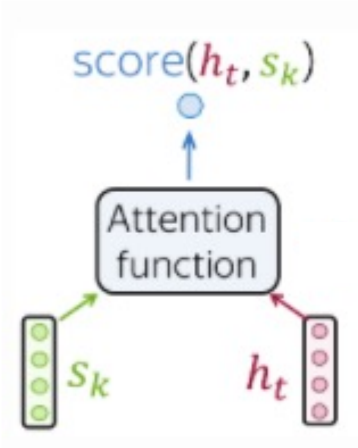


For convenience, here's the Attention calculation summarized on 1 slide



Usual score function in the next slide

Photo credit: https://lena-voita.github.io/nlp_course/seq2seq_and_attention.html



Popular Attention Scoring functions:

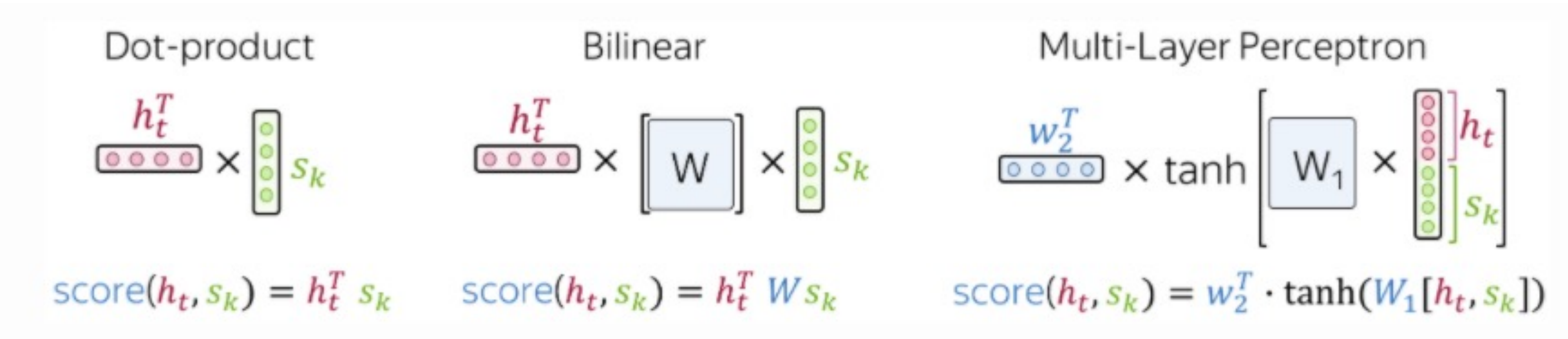


Photo credit: https://lena-voita.github.io/nlp_course/seq2seq_and_attention.html

seq2seq + Attention

Attention:

- Improves seq2seq results
- Allows us to visualize the contribution each encoding word gave for each decoder's word

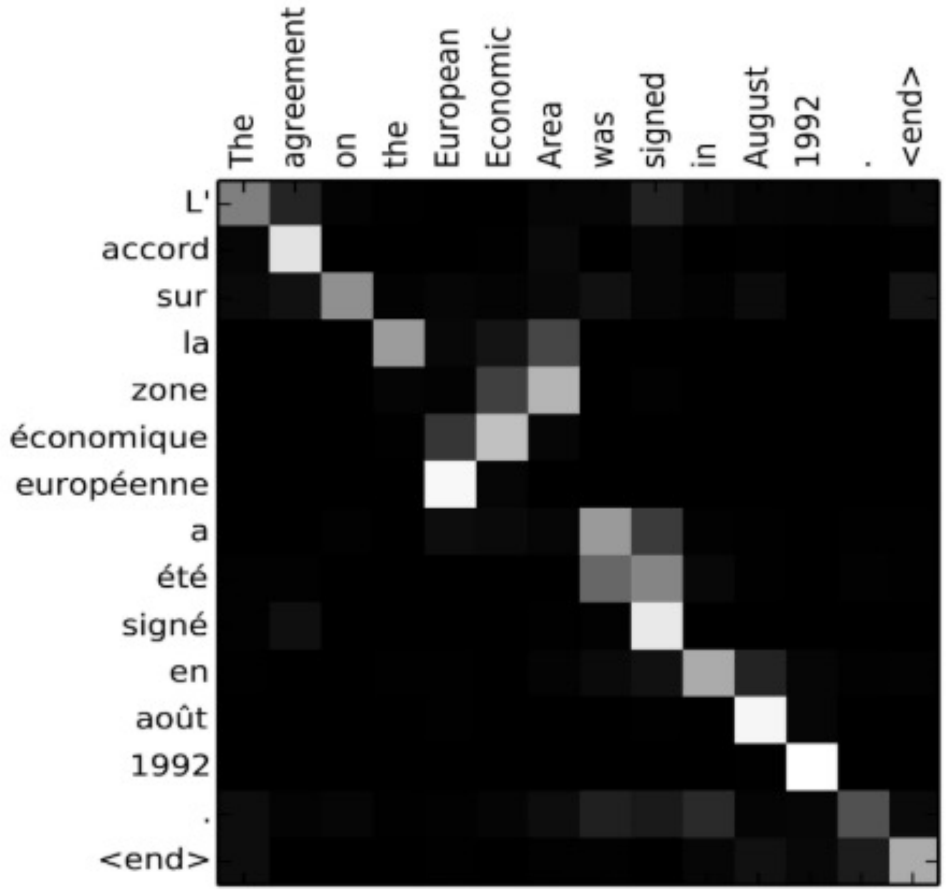


Image source: Bahdanau et al., 2015

Takeaway:

Having a separate encoder and decoder allows for n → m length predictions.

Attention is powerful; allows us to conditionally weight our focus

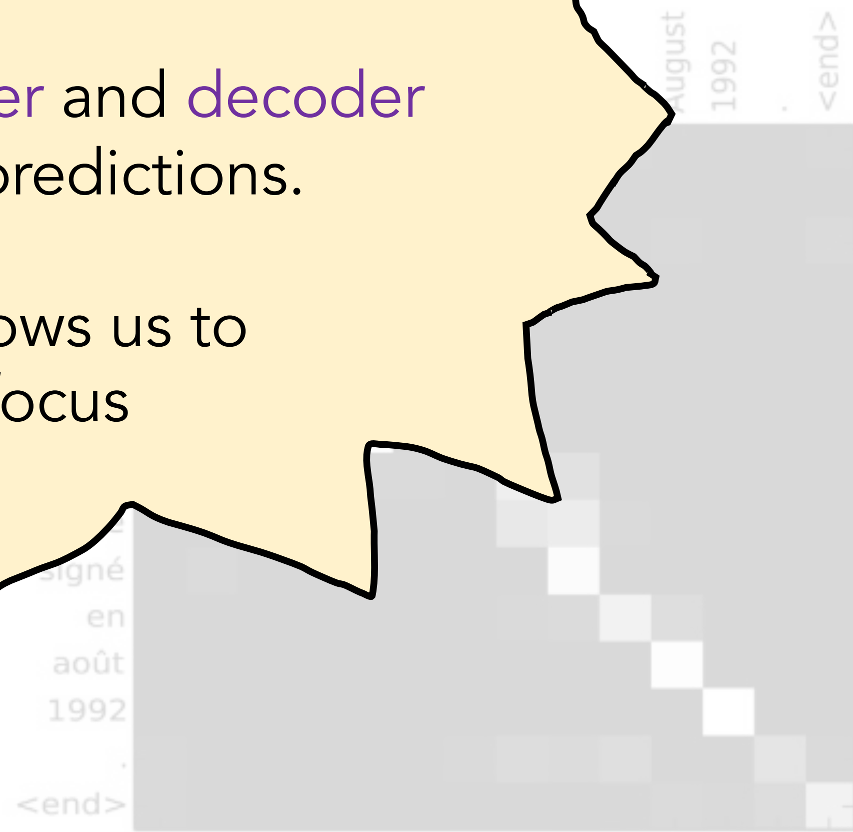


Image source: Fig 3 in [Bahdanau et al., 2015]

SUMMARY

- **LSTMs** yielded state-of-the-art results on most NLP tasks (2014-2018)
- **seq2seq+Attention** was an even more revolutionary idea
- **Attention** allows us to place appropriate weight to the encoder's hidden states
- But, **LSTMs** require us to iteratively scan each word and wait until we're at the end before we can do anything
  - It is not possible to parallelize the task

  → Transformers will correct this

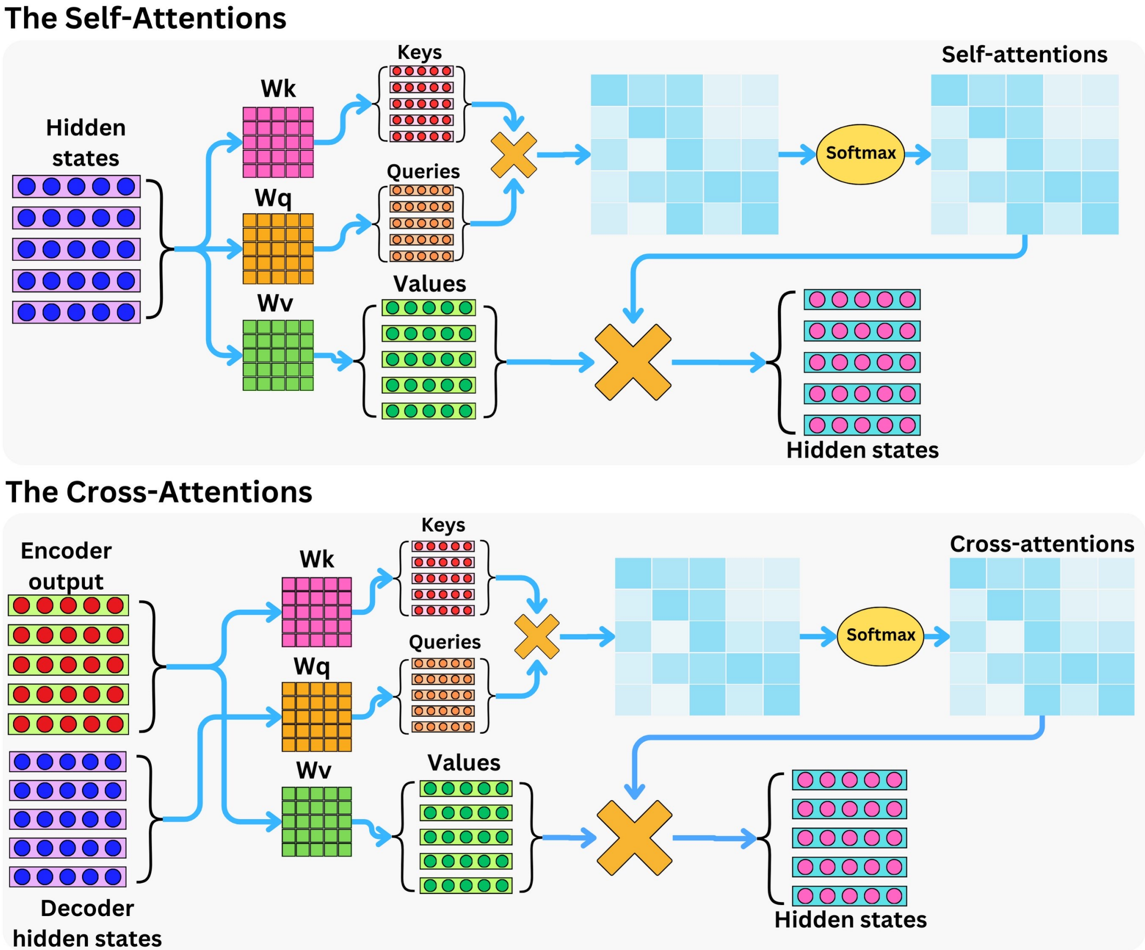
Be carefull when you read papers: 3 types of attention

- *global attention (or cross attention):* uses all the encoder hidden
  - Presented in this lecture
- *local attention or self :* uses only a subset of the encoder hidden states
- *Self attention:* uses with transformer architecture
  - Could be parallized

Be carefull when you read papers: 2 kind of attention

Self or local attention

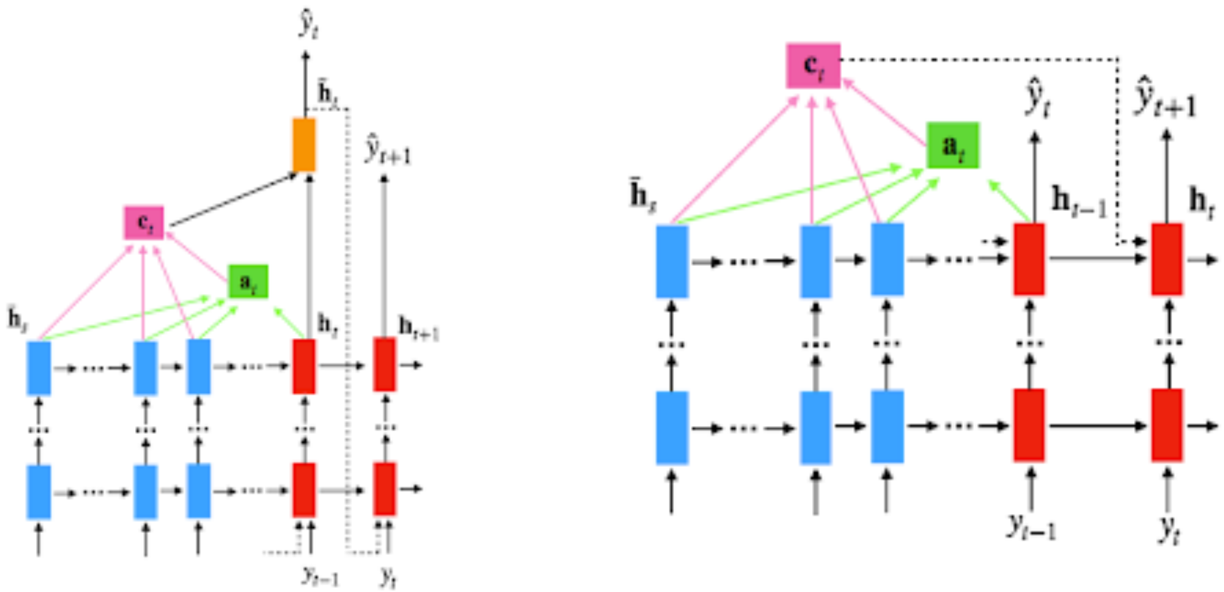
Cross or global attention



TheAiEdge.io

Be carefull when you read papers: 2 types of architecture

Luong Attention Mechanism	Bahdanau Attention Mechanism
$\mathbf{h}_t \rightarrow \mathbf{a}_t \rightarrow \mathbf{c}_t \rightarrow \tilde{\mathbf{h}}_t$	$\mathbf{h}_{t-1} \rightarrow \mathbf{a}_t \rightarrow \mathbf{c}_t \rightarrow \mathbf{h}_t$



$\mathbf{a}_t(s) = \text{align}(\mathbf{h}_t, \bar{\mathbf{h}}_s)$	$\mathbf{a}_t(s) = \text{align}(\mathbf{h}_{t-1}, \bar{\mathbf{h}}_s)$
$\mathbf{c}_t = \sum a_t \mathbf{h}_s$	$\mathbf{c}_t = \sum a_t \mathbf{h}_s$
$\tilde{\mathbf{h}}_t = \tanh(W_c[\mathbf{c}_t; \mathbf{h}_t])$	$\mathbf{h}_t = \text{RNN}(\mathbf{h}_{t-1}^{l-1}, [\mathbf{c}_t; \mathbf{h}_{t-1}])$

Tensorflow Attentional layer

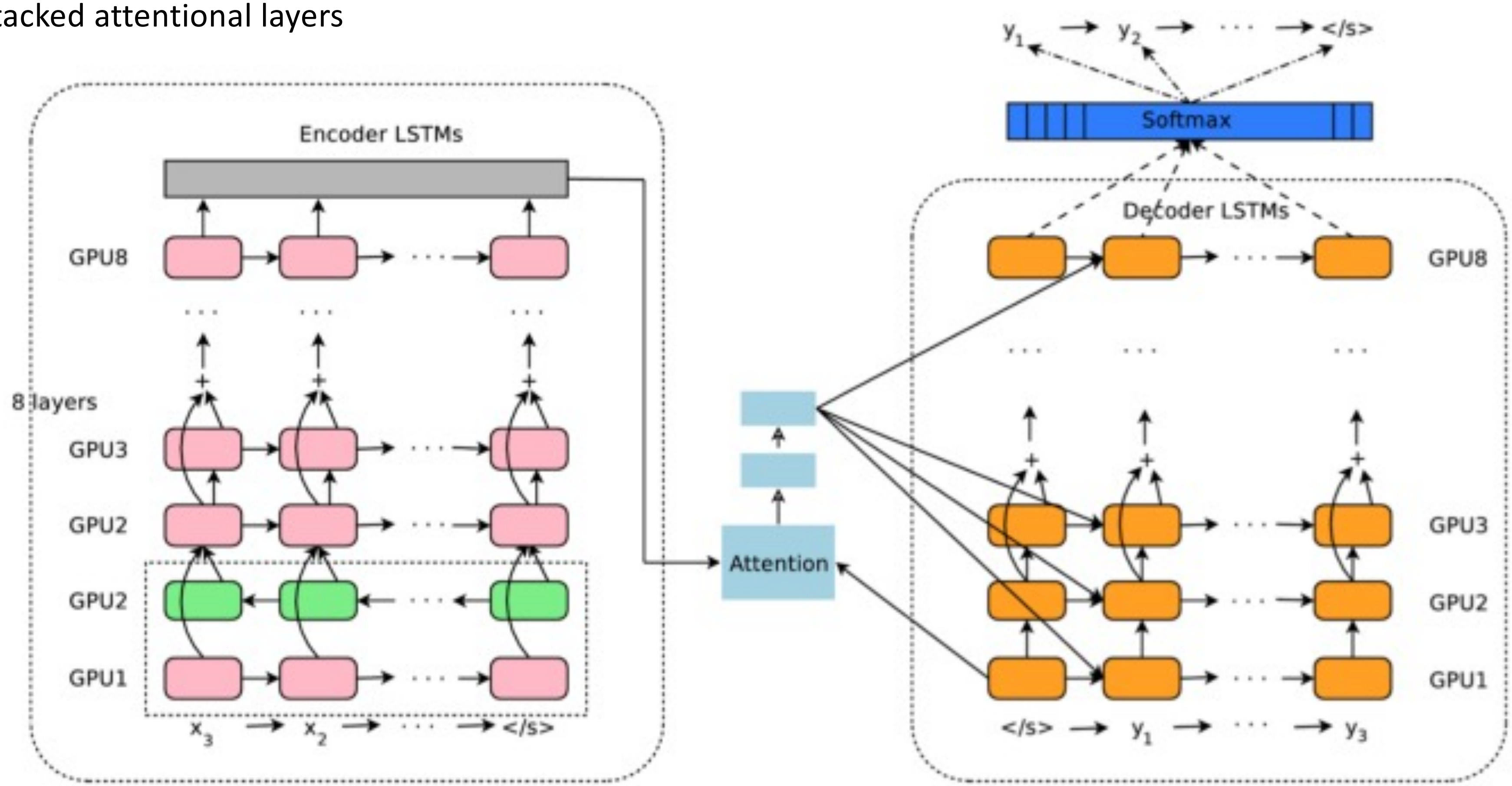
```
tf.keras.layers.Attention
```

Query-value attention of shape [batch_size, Tq, filters].

```
query_value_attention_seq =
        tf.keras.layers.Attention()([
             query_seq_encoding,
             value_seq_encoding
          ])
```

The Google Neural Machine Translation — GNMT architecture

Stacked attentional layers



Yonghui Wu et al. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation (2016) https://arxiv.org/pdf/1609.08144.pdf

Some popularization paper

- Sequence to Sequence (seq2seq) and Attention; https://lena-voita.github.io/nlp_course/seq2seq_and_attention.html
- Craft your own Attention layer in 6 lines — Story of how the code evolved: https://towardsdatascience.com/create-your-own-custom-attention-layer-understand-all-flavours-2201b5e8be9e
- DeepAR: Mastering Time-Series Forecasting with Deep Learning: https://towardsdatascience.com/deepar-mastering-time-series-forecasting-with-deep-learning-bc717771ce85

MSc. DSAI Lab

- Add a new step on your notebook
  - Preprocessing
  - Seq2Seq architecture
  - Improved seq2seq architecture (optional)
    - Stacked bi-LSTM
    - Context available to each decoder step

  → Attentional Seq2Seq architecture
  - Transformer architecture
  - Transformer Transfer learning with https://huggingface.co/ (optional)