Lecture 8: Machine Translation

And the power of Attention

Harvard

AC295/CS287r/CSCI E-115B

Chris Tanner

Florence + the Machine Translation

[SMT] Days are Over (2011)

ANNOUNCEMENTS

- Quizzes 1 and 2 have been graded and are logged on Canvas
- HW1 is being graded. Solutions are posted on Canvas -> Files
- HW2 is due next Tues, Oct 5 @ 11:59pm! Determine your mystery language.
- Research Proposals are due Thursday night, Sept 30 @ 11:59pm.
  - If submitting w/ others, please see the updated Canvas instructions

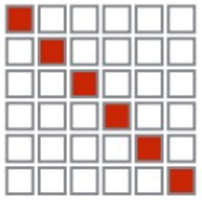
RECAP: L7

Unconditioned Predictions

Independent Prediction

$$P(X) = \prod_{i=1}^{|X|} P(x_i)$$

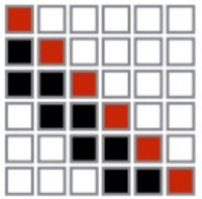
(e.g. unigram model)



Left-to-right Markov Chain (order n-1)

$$P(X) = \prod_{i=1}^{|X|} P(x_i | x_{i-n+1}, \ldots, x_{i-1})$$

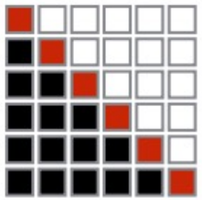
(e.g. n-gram LM, feed-forward LM)



Left-to-right Autoregressive Prediction

$$P(X) = \prod_{i=1}^{|X|} P(x_i | x_1, \ldots, x_{i-1})$$

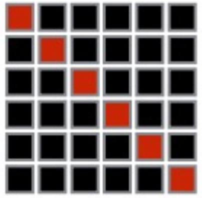
(e.g. RNN LM)



Bidirectional Prediction

$$P(X) \neq \prod_{i=1}^{|X|} P(x_i | x_{\neq i})$$

(e.g. masked language model)

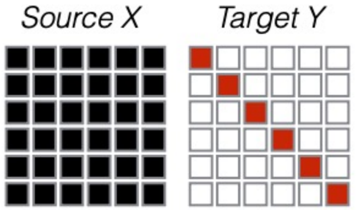


Conditioned Predictions

Non-autoregressive Conditioned Prediction

$$P(Y|X) = \prod_{i=1}^{|Y|} P(y_i | X)$$

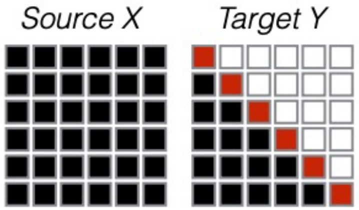
(e.g. sequence labeling, non-autoregressive MT)



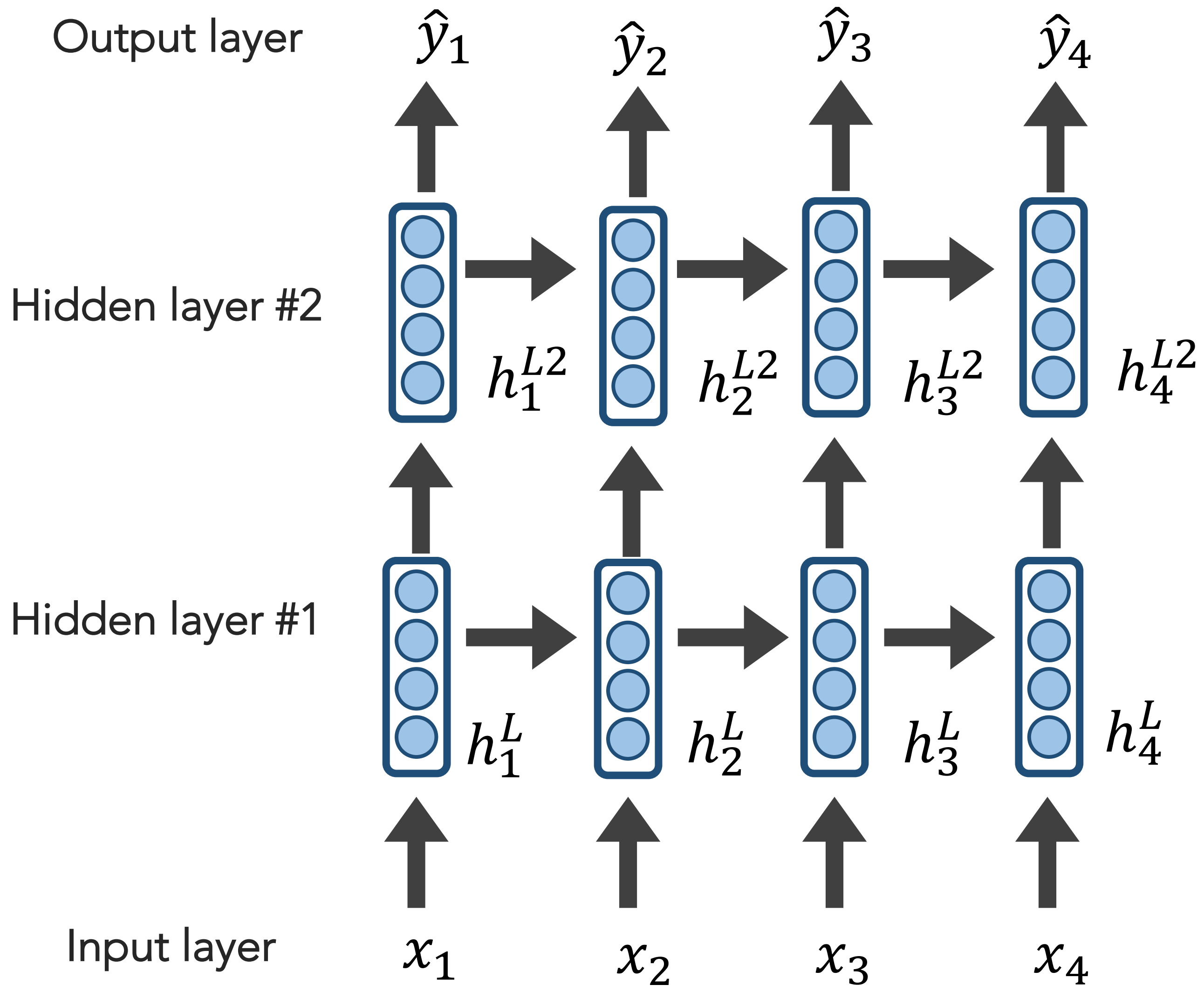
Autoregressive Conditioned Prediction

$$P(Y|X) = \prod_{i=1}^{|Y|} P(y_i | X, y_1, \ldots, y_{i-1})$$

(e.g. seq2seq model)



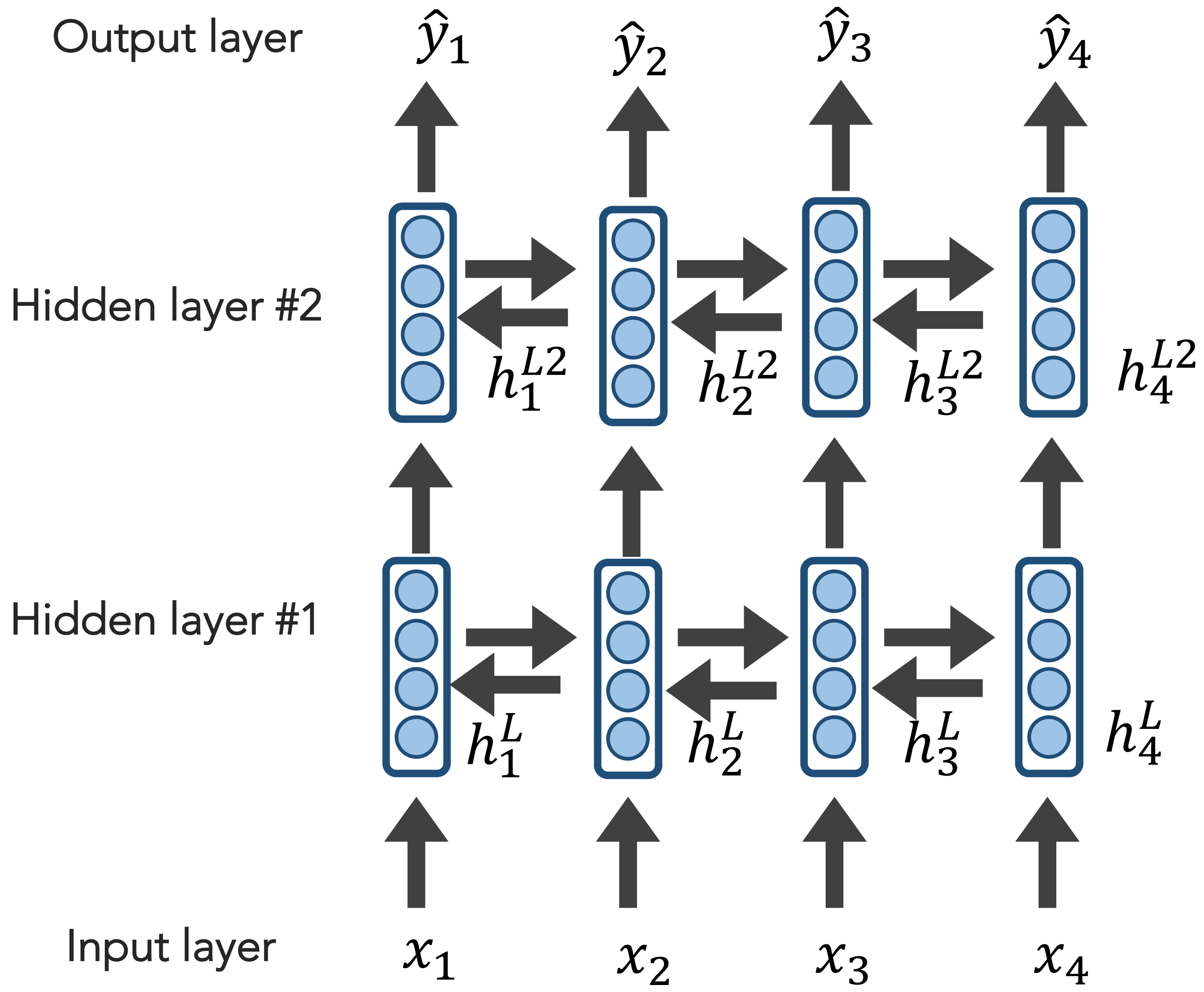
RECAP: L7



Hidden layers provide an abstraction (holds "meaning").

Stacking hidden layers provides increased abstractions.

RECAP: L7



Hidden layers provide an abstraction (holds "meaning").

Stacking hidden layers provides increased abstractions.

Depending on our assumptions, could add **bi-directionality**, too.

Outline

- Machine Translation
- seq2seq
- seq2seq + Attention

Many slides in the MT section were inspired by or adapted from [Abigail See's Stanford CS224N lecture]

Outline

- Machine Translation
- seq2seq
- seq2seq + Attention

Many slides in the MT section were inspired by or adapted from [Abigail See's Stanford CS224N lecture]

Machine Translation (MT) is an NLP task that aims to convert text from one language to another.

Thank you for visiting! → Děkujeme za návštěvu!

$\boldsymbol{x}$ (source language) → $\boldsymbol{y}$ (target language)

Many slides in the MT section were inspired by or adapted from [Abigail See's Stanford CS224N lecture]

Machine Translation (MT) is an NLP task that aims to convert text from one language to another.

9th century: **Al-Kindi** (cryptographer)

17th century: **René Descartes** theorized about a universal, symbolic language

1946: **Warren Weaver** had a seminal publication

1950s: First huge efforts; MIT, IBM, US Government. Motivated by the Cold War.

1990s – 2014: Statistical MT.

2014 – present: Neural MT (Deep Learning)

1990s – 2014: Statistical MT

$$\mathrm{argmax}_{\mathrm{y}} P(y|x)$$

$$= \mathrm{argmax}_{\mathrm{y}} P(x|y) P(y)$$

1990s – 2014: Statistical MT

$$\mathrm{argmax}_{\mathrm{y}} P(y|x)$$

$$= \mathrm{argmax}_{\mathrm{y}} \underbrace{P(x|y)}_{\text{Translation model}} \underbrace{P(y)}_{\text{Language model}}$$

1990s – 2014: Statistical MT

$$\text{argmax}_{\text{y}} P(x|y)P(y)$$

We estimate $P(x|y)$ from a parallel corpus.

1990s – 2014: Statistical MT

$$\mathrm{argmax}_{\mathrm{y}} P(x|y)P(y)$$

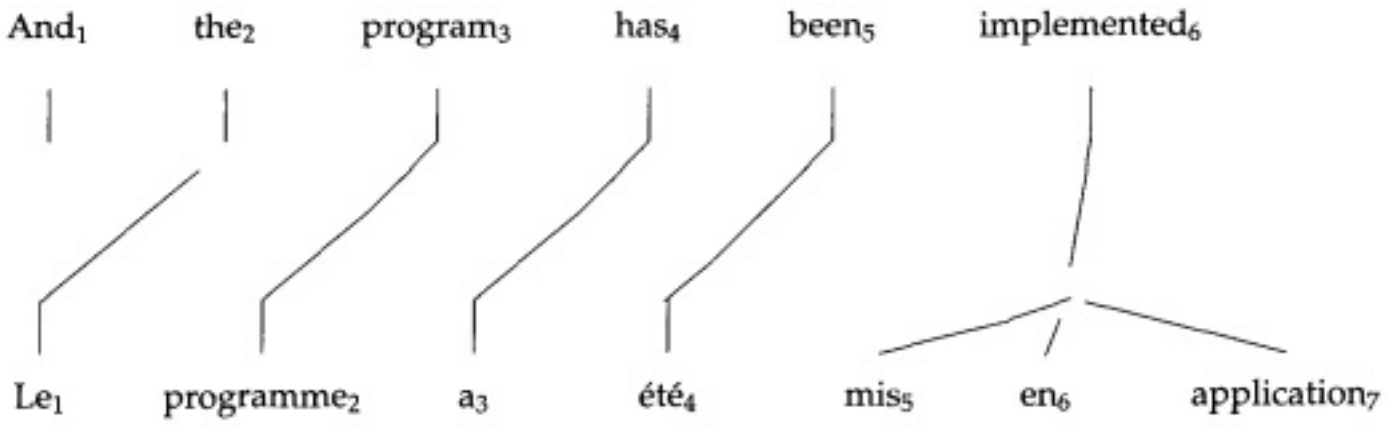
We estimate $P(x|y)$ from a parallel corpus.

Technically, we're actually interested in:

$$\mathrm{argmax}_{\mathrm{y}} P(x, a|y)P(y)$$

Def: alignment is the correspondence between particular words in different languages

1990s – 2014: Statistical MT

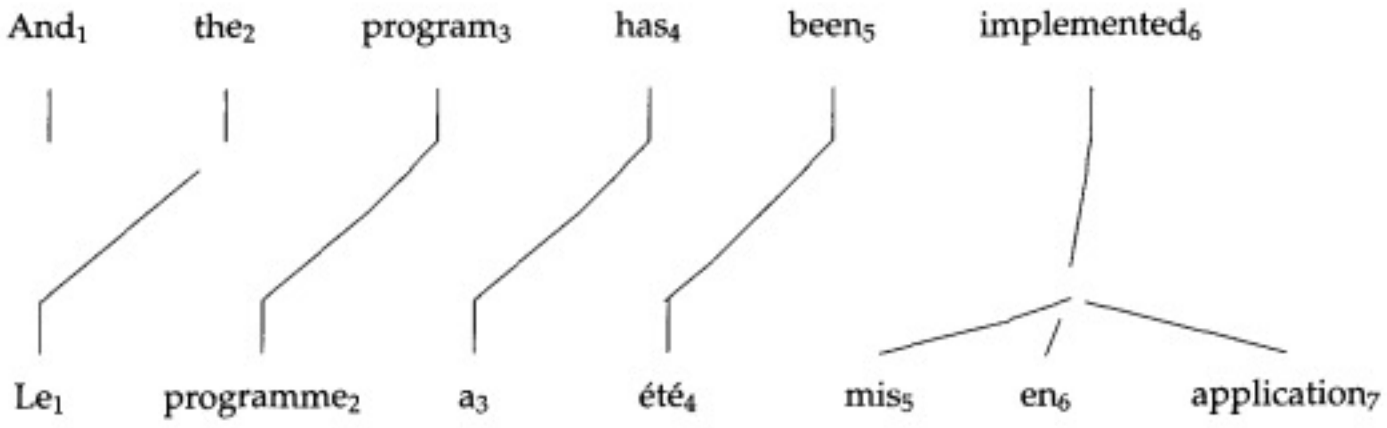


Some 1-to-0 mappings, 1-to-1 mappings, 1-to-many mappings, etc

See chalkboard for a table view.

The Mathematics of Statistical Machine Translation: Parameter Estimation. Brown et al. ACL 1993.

1990s – 2014: Statistical MT



Alignment is complex and can be based on:

- particular words aligning
- sentence positions
- word "fertility" (the # of words a word spans to)
- much more

The Mathematics of Statistical Machine Translation: Parameter Estimation. Brown et al. ACL 1993.

1990s – 2014: Statistical MT

$$\mathrm{argmax}_{\mathrm{y}} P(x|y)P(y)$$

How in the world can we compute this argmax?! Exploding/infinite y's!

1990s – 2014: Statistical MT

$$\text{argmax}_{\mathrm{y}} P(x|y)P(y)$$

How in the world can we compute this argmax?! Exploding/infinite y's!

Neural MT (deep learning approach) needs to address this, too.

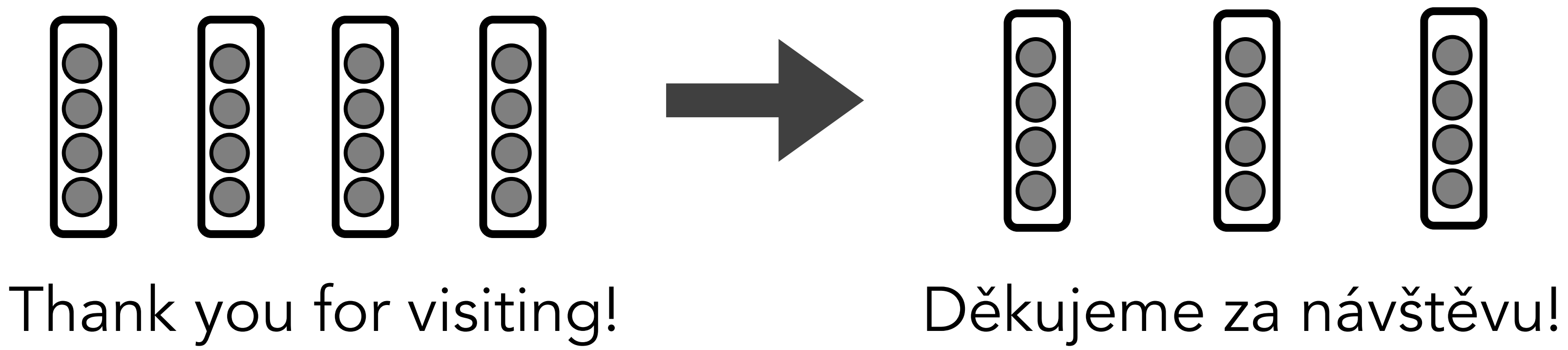
1990s – 2014: Statistical MT

SUMMARY

- SMT was largely successful and useful
- SOTA systems were incredibly complex (e.g., Moses and GIZA++)
- Ridiculous amounts of manual feature engineering
- Relied on external resources like phrase translation tables
- RIP: the years 2011-2012 for me

We want to produce a **variable-length** output

(e.g., n → m predictions)



Outline

- Machine Translation
- seq2seq
- seq2seq + Attention

Many slides in the MT section were inspired by or adapted from [Abigail See's Stanford CS224N lecture]

Outline

- Machine Translation
- seq2seq
- seq2seq + Attention

Sequence-to-Sequence (seq2seq)

- If our input is a sentence in Language A, and we wish to translate it to Language B, it is clearly sub-optimal to translate word by word (like our current models are suited to do).

- Instead, let a ***sequence*** of tokens be the unit that we ultimately wish to work with (a sequence of length **N** may emit a sequences of length **M**)

- seq2seq models are comprised of **2 RNNs**: 1 encoder, 1 decoder

Types of Conditional Prediction

Many-to-1 classification

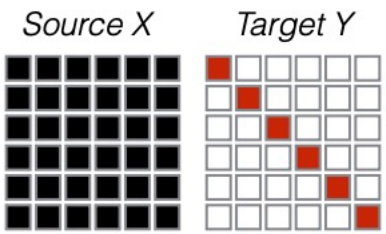
$$P(y|X)$$

Many-to-many classification

Non-autoregressive Conditioned Prediction

$$P(Y|X) = \prod_{i=1}^{|Y|} P(y_i|X)$$

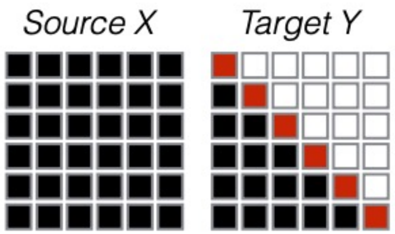
(e.g. sequence labeling, non-autoregressive MT)



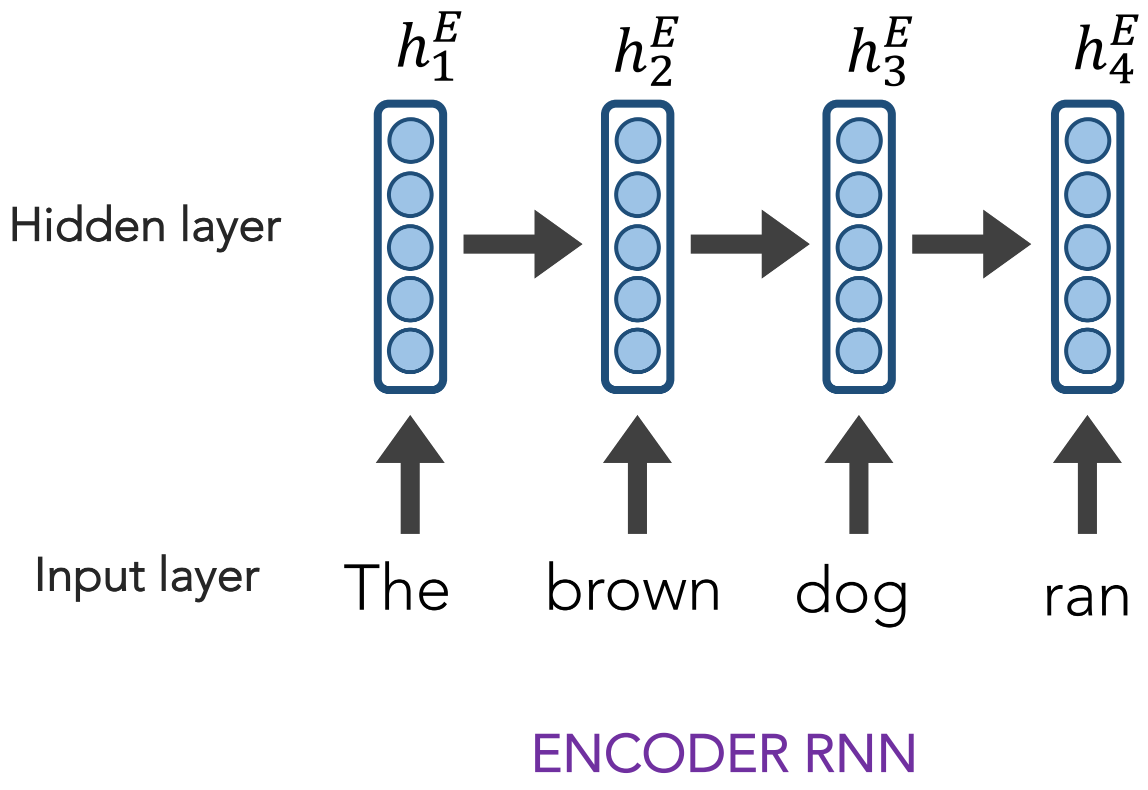
Autoregressive Conditioned Prediction

$$P(Y|X) = \prod_{i=1}^{|Y|} P(y_i|X, y_1, \ldots, y_{i-1})$$

(e.g. seq2seq model)

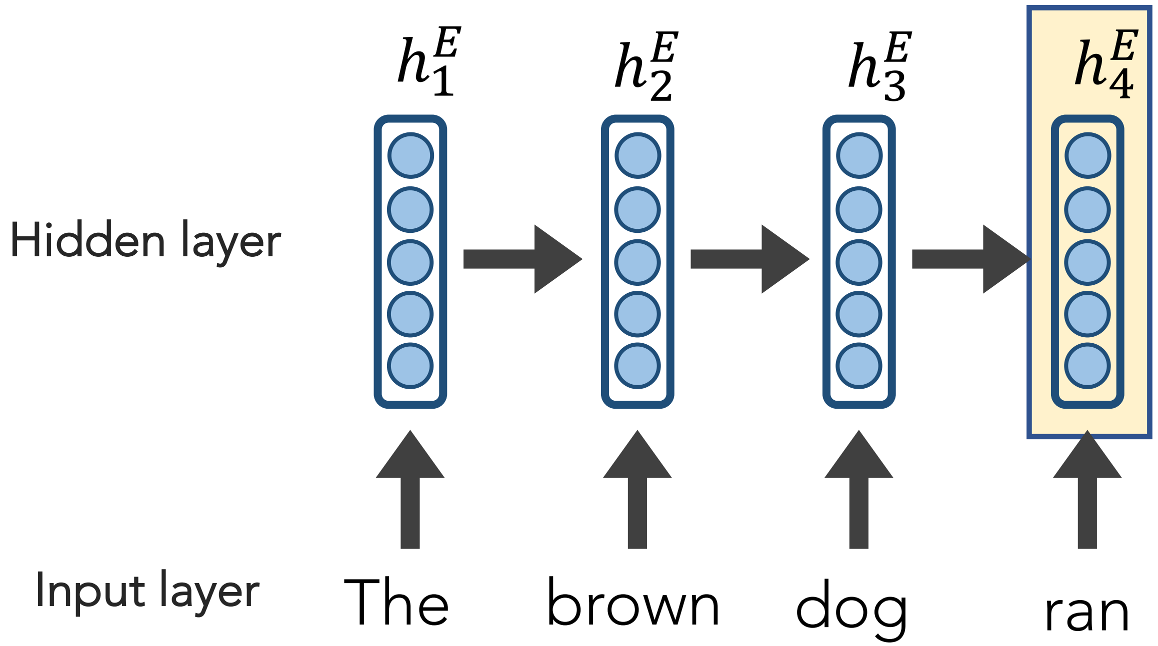


Sequence-to-Sequence (seq2seq)



Sequence-to-Sequence (seq2seq)

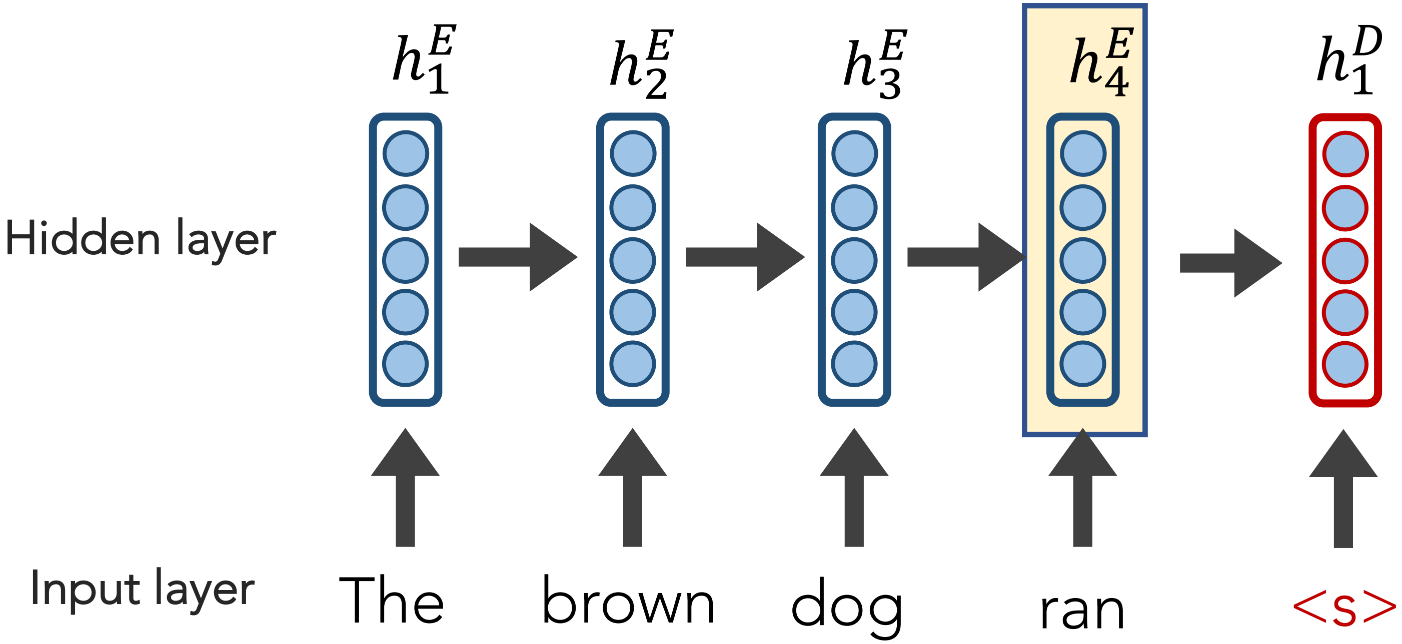
The final hidden state of the encoder RNN
is the initial state of the decoder RNN



ENCODER RNN

Sequence-to-Sequence (seq2seq)

The final hidden state of the encoder RNN
is the initial state of the decoder RNN

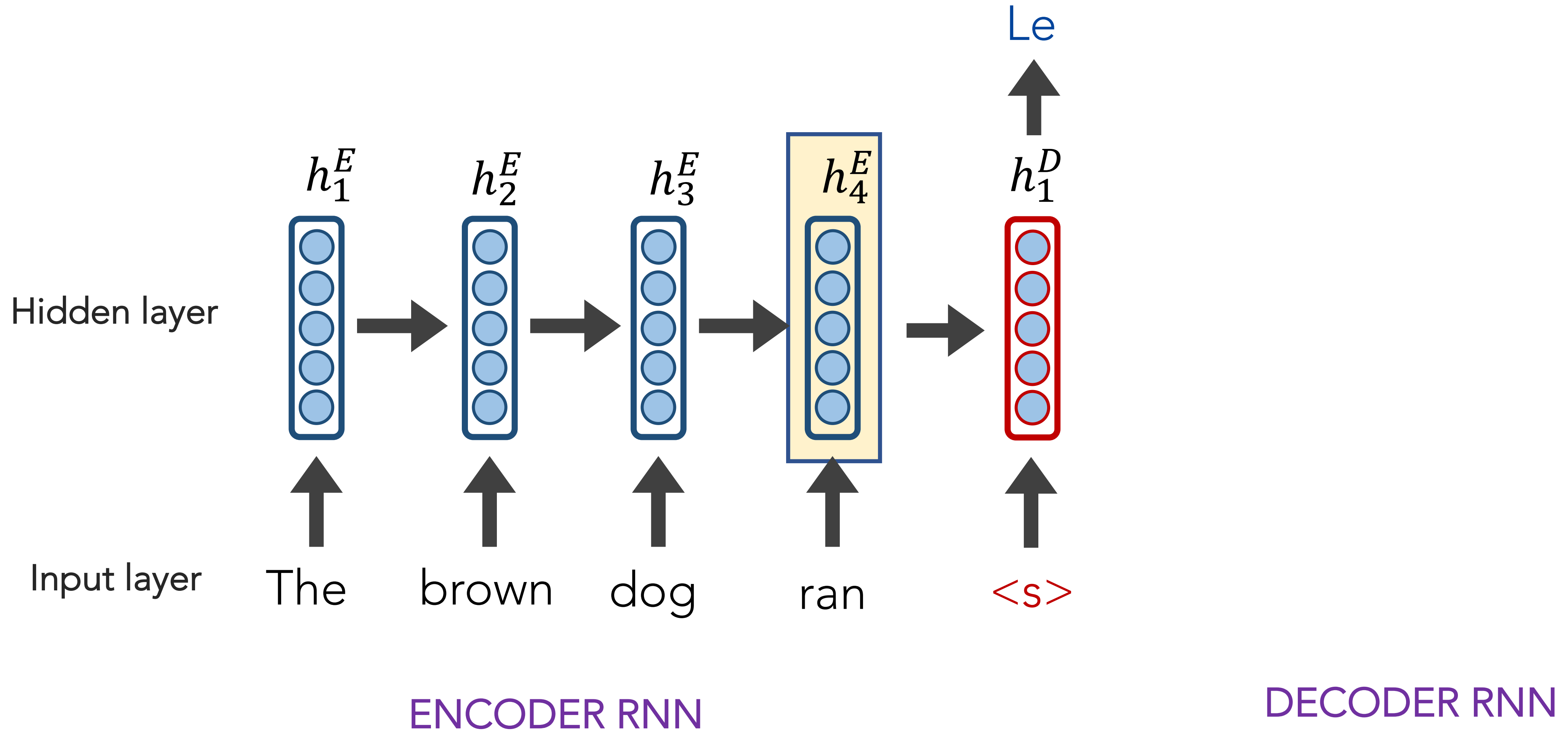


ENCODER RNN

DECODER RNN

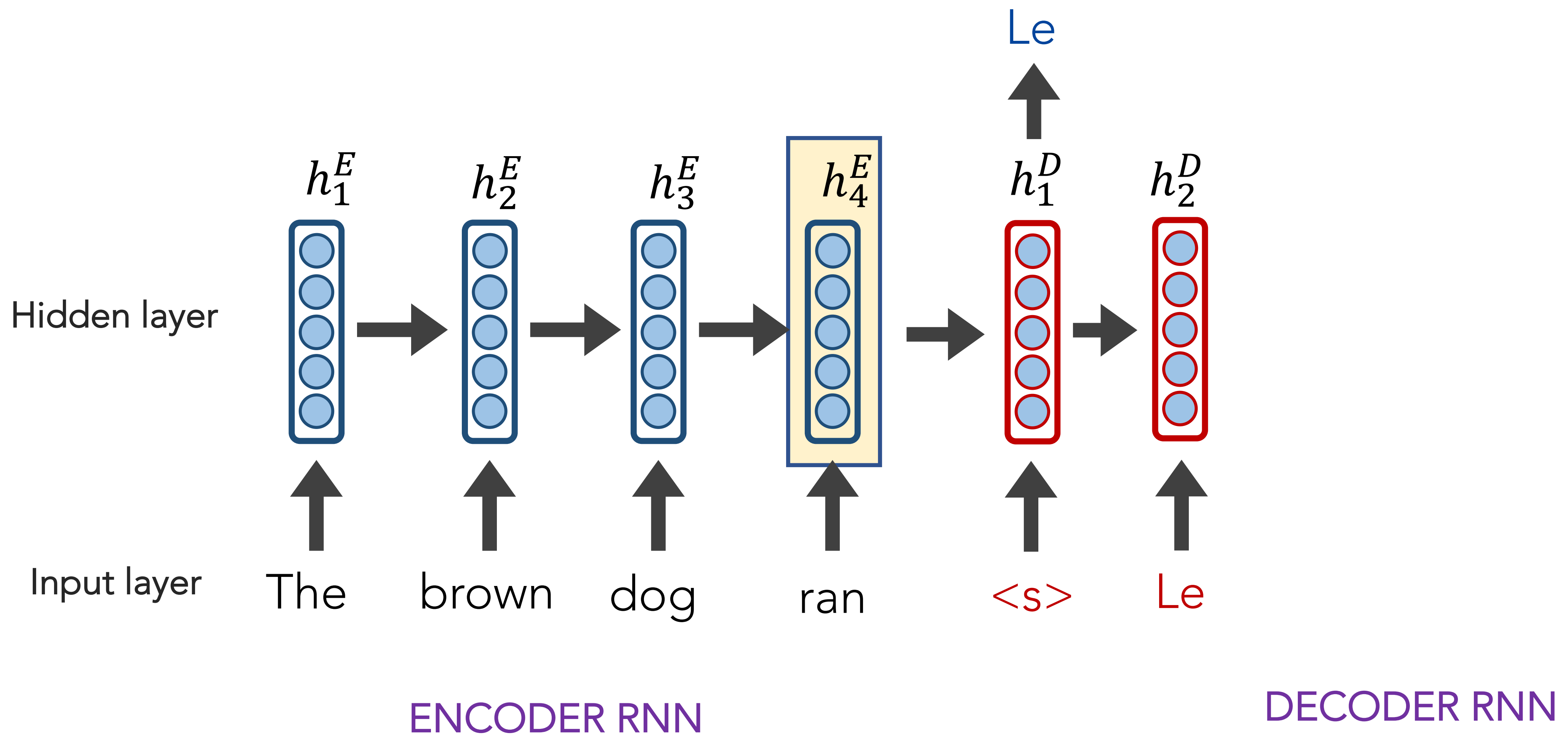
Sequence-to-Sequence (seq2seq)

The final hidden state of the encoder RNN
is the initial state of the decoder RNN

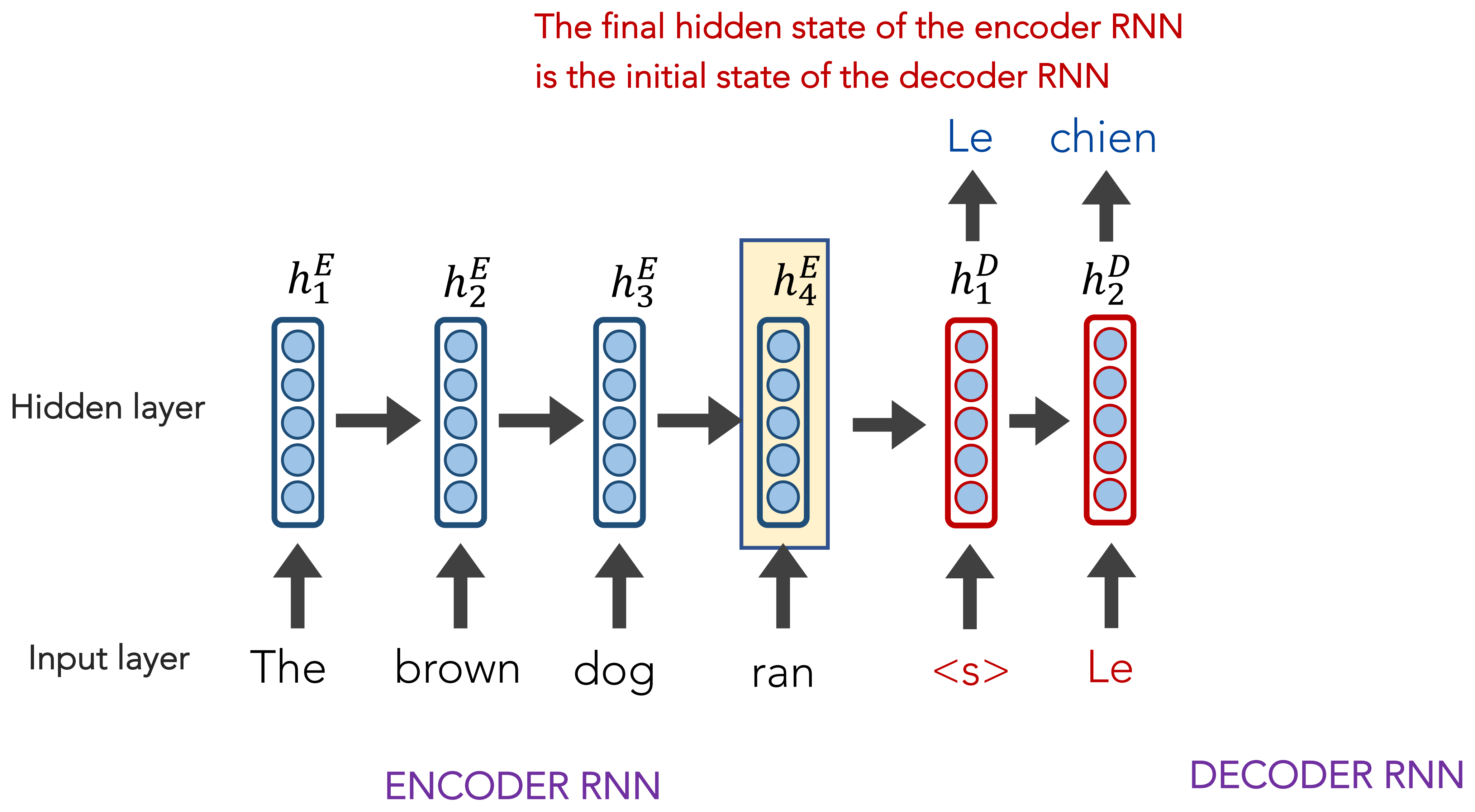


Sequence-to-Sequence (seq2seq)

The final hidden state of the encoder RNN
is the initial state of the decoder RNN

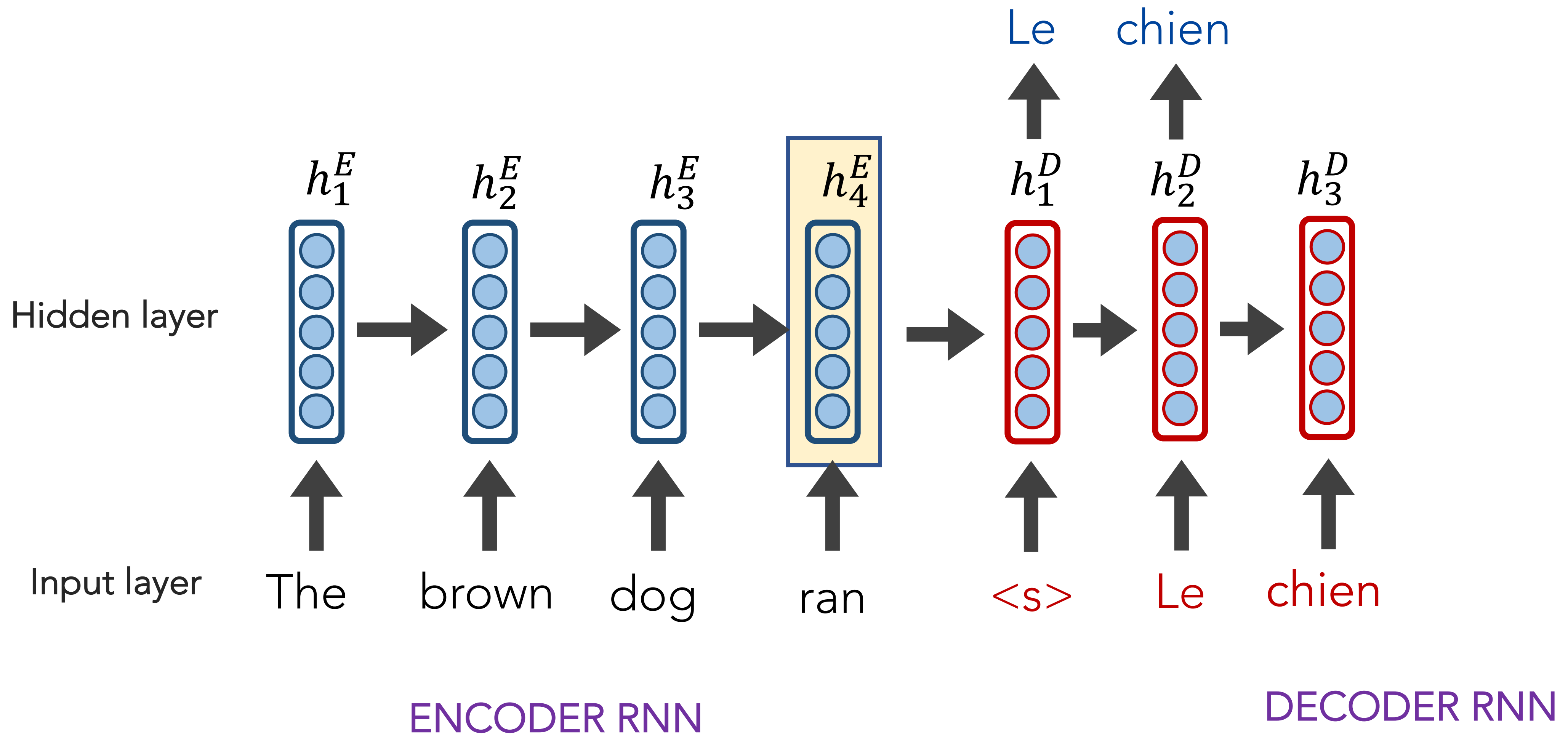


Sequence-to-Sequence (seq2seq)

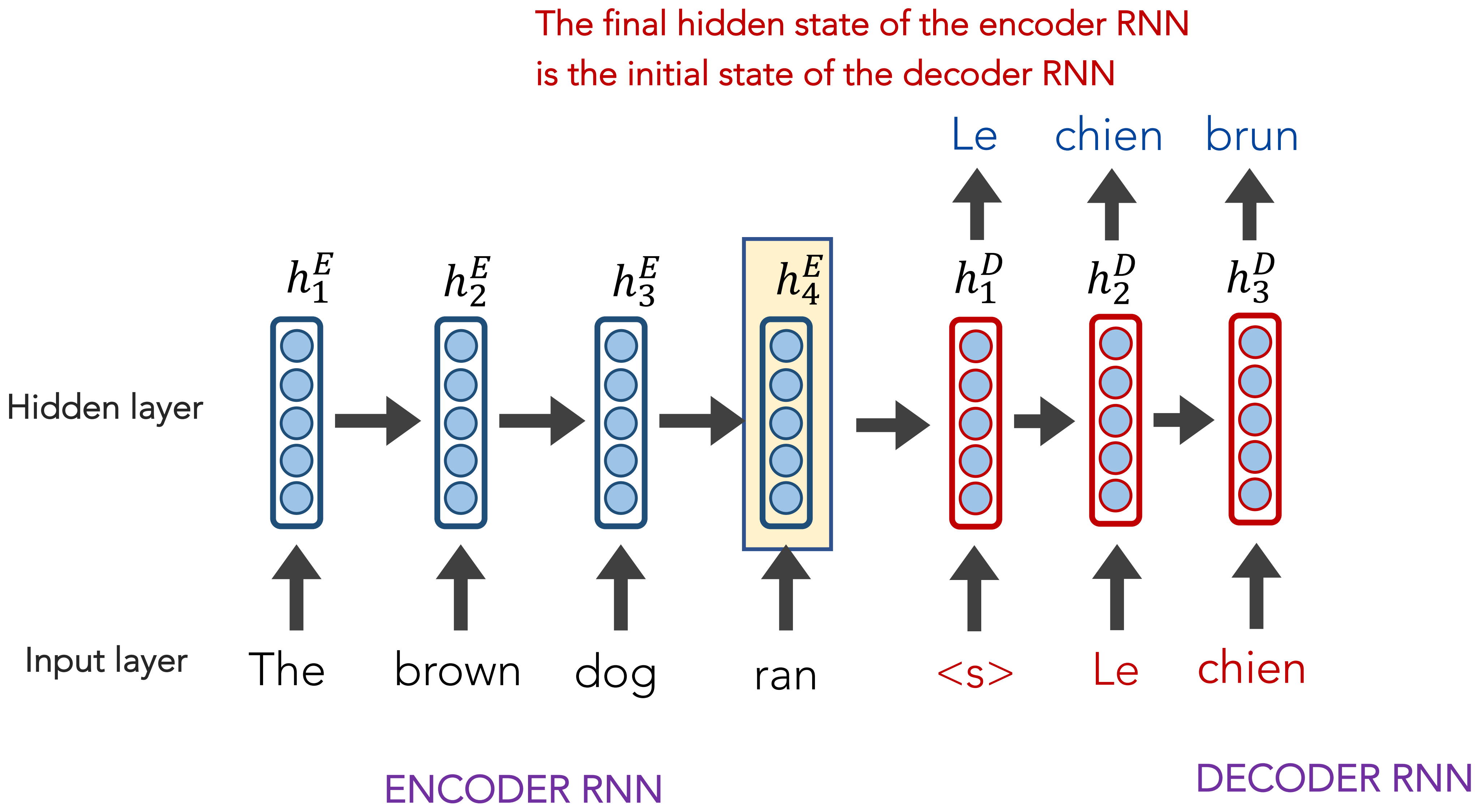


Sequence-to-Sequence (seq2seq)

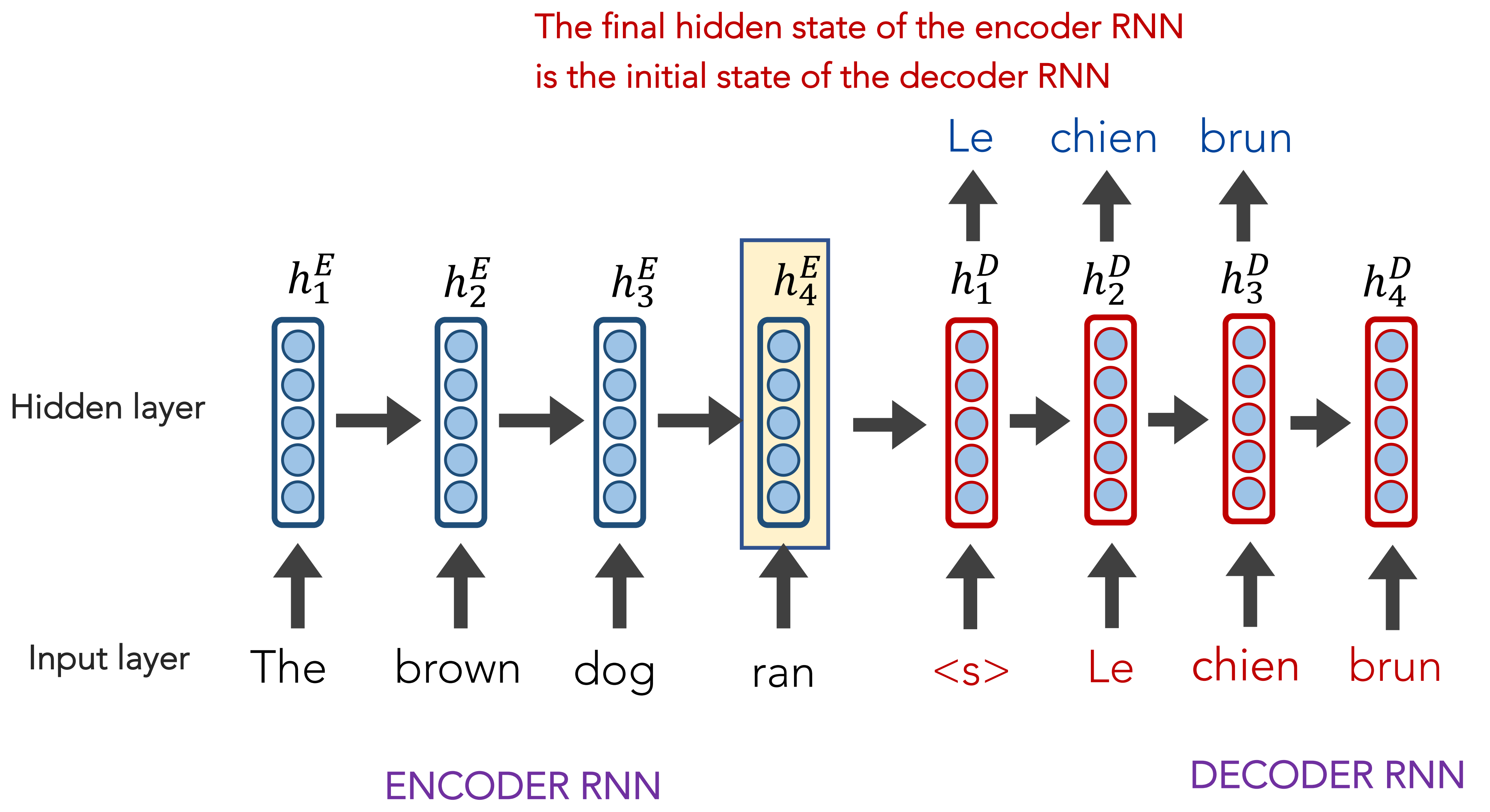
The final hidden state of the encoder RNN
is the initial state of the decoder RNN



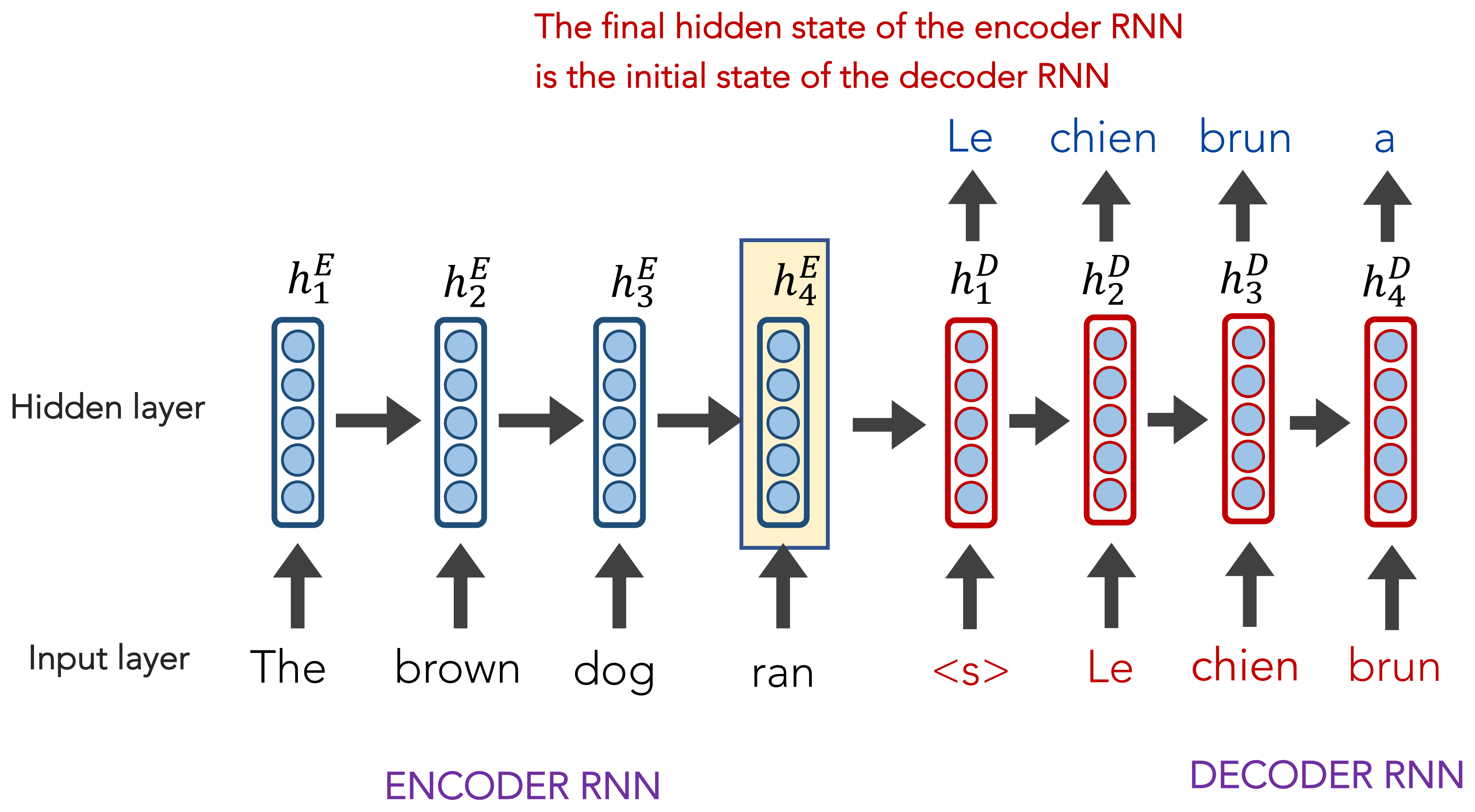
Sequence-to-Sequence (seq2seq)



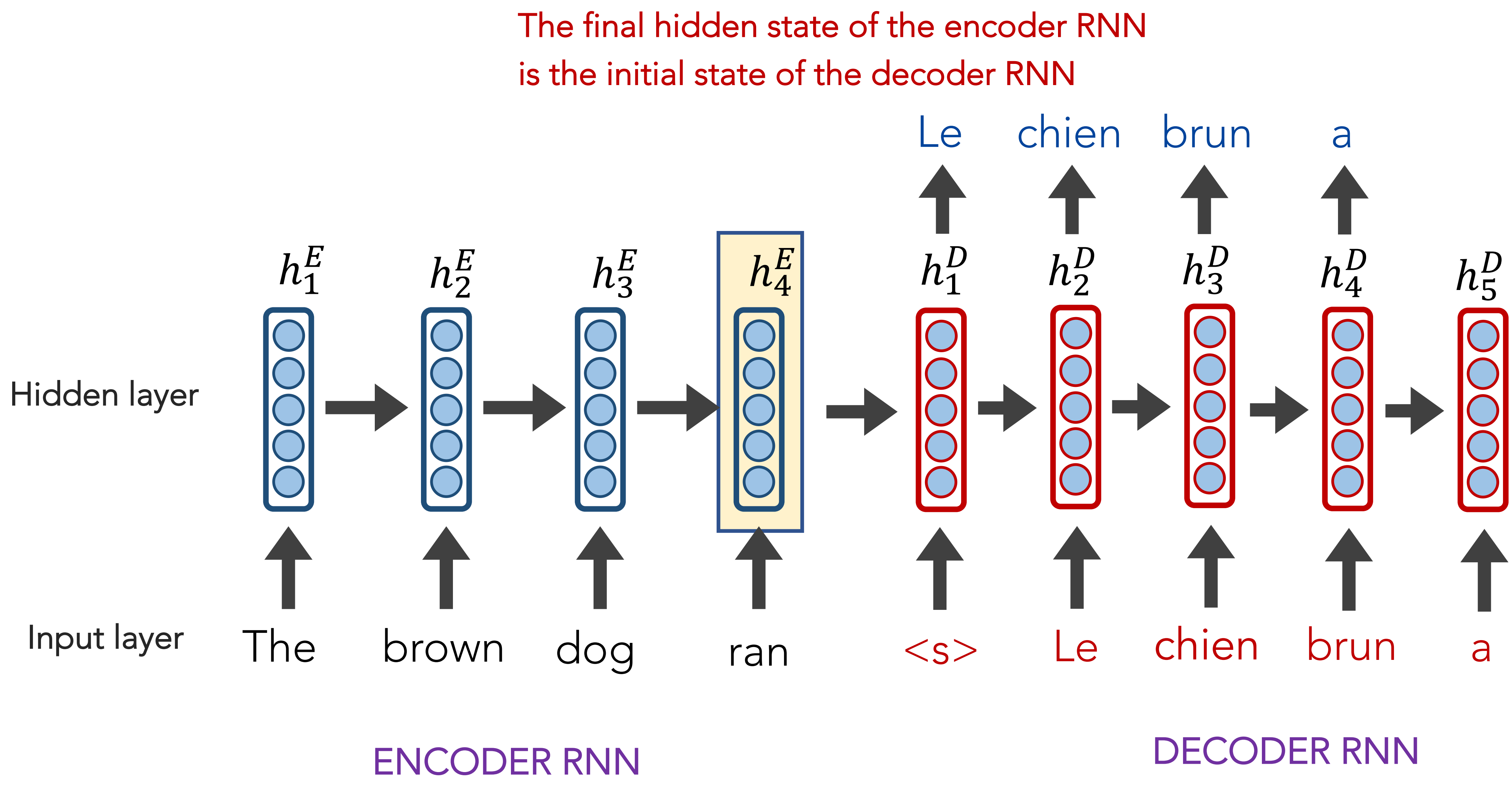
Sequence-to-Sequence (seq2seq)



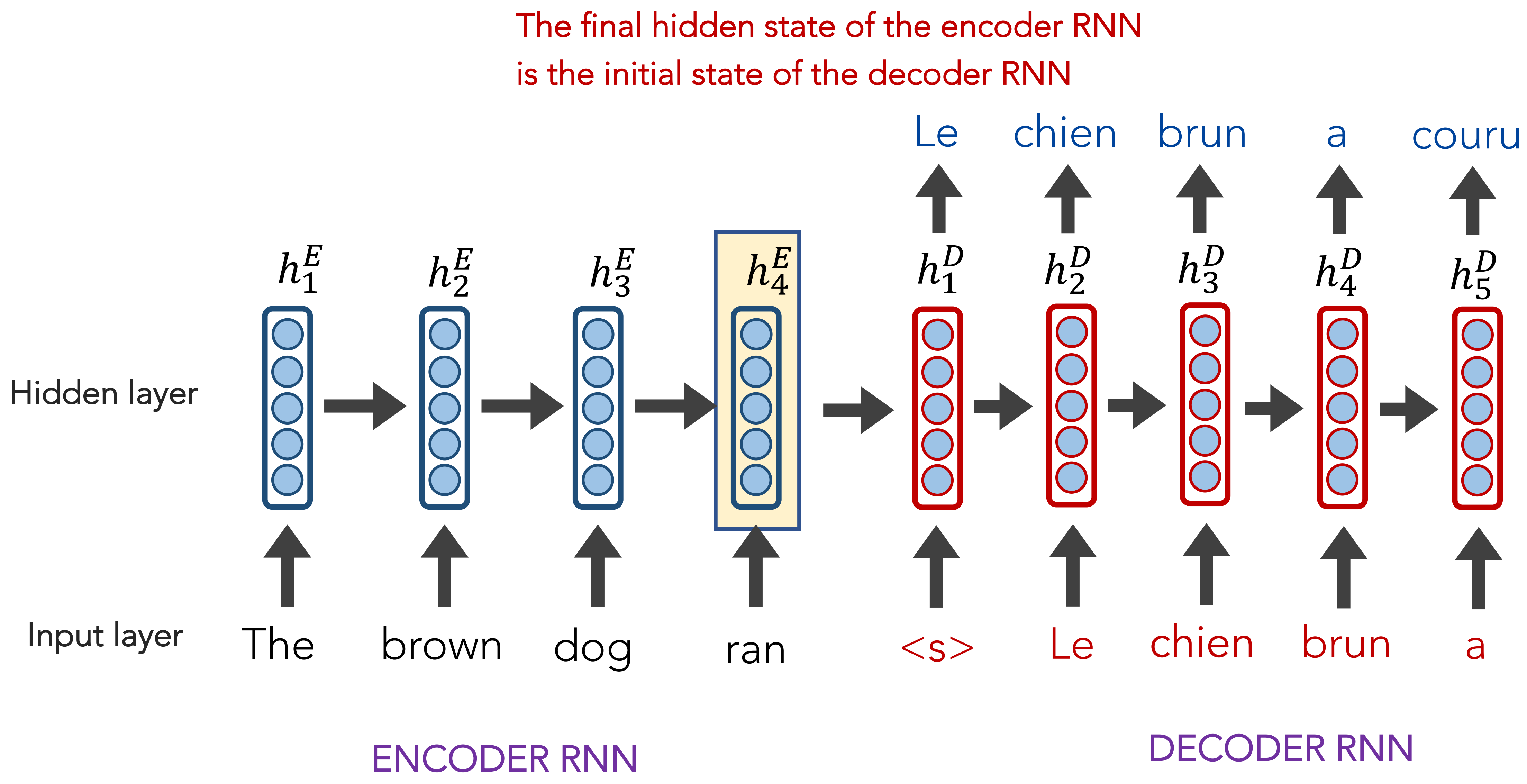
Sequence-to-Sequence (seq2seq)



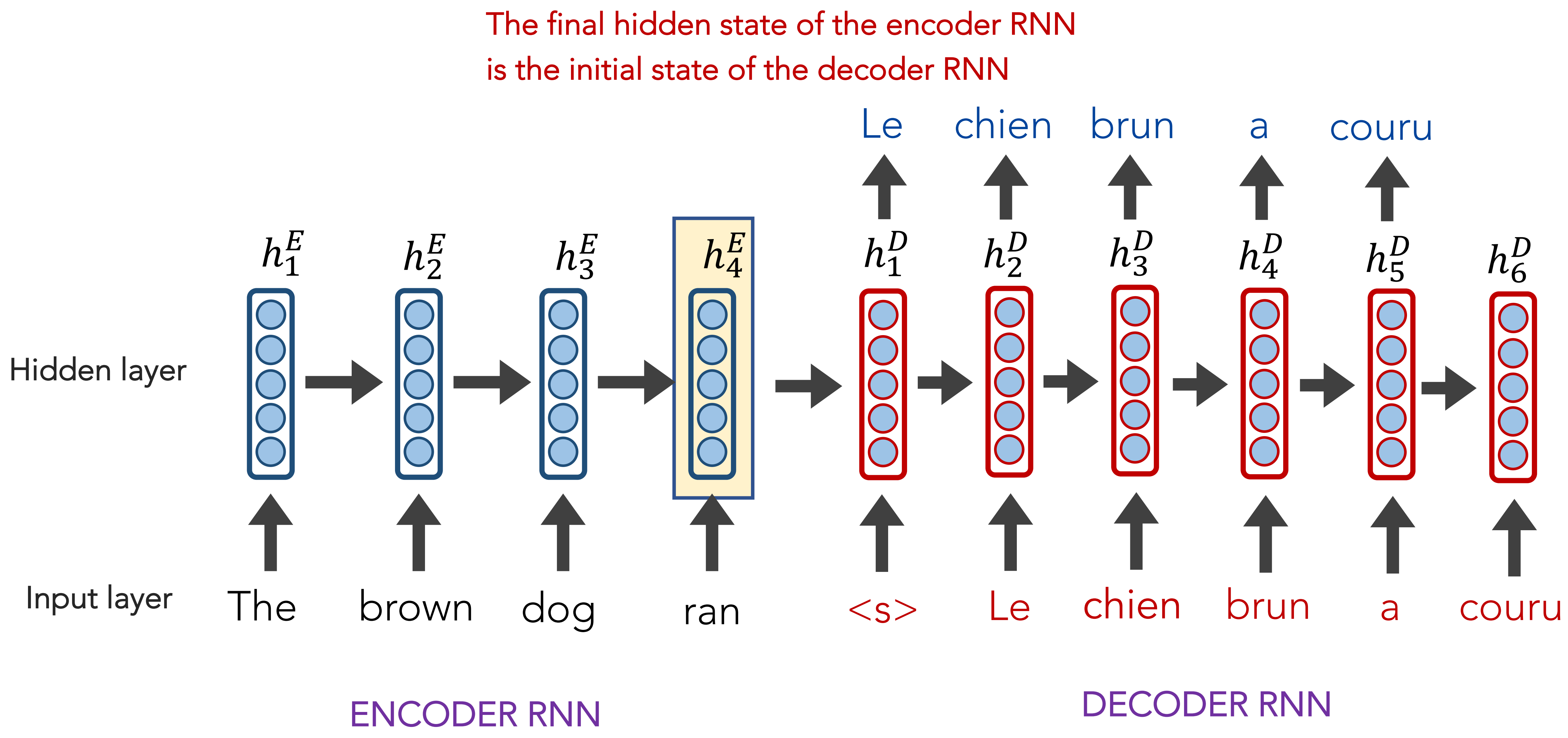
Sequence-to-Sequence (seq2seq)



Sequence-to-Sequence (seq2seq)

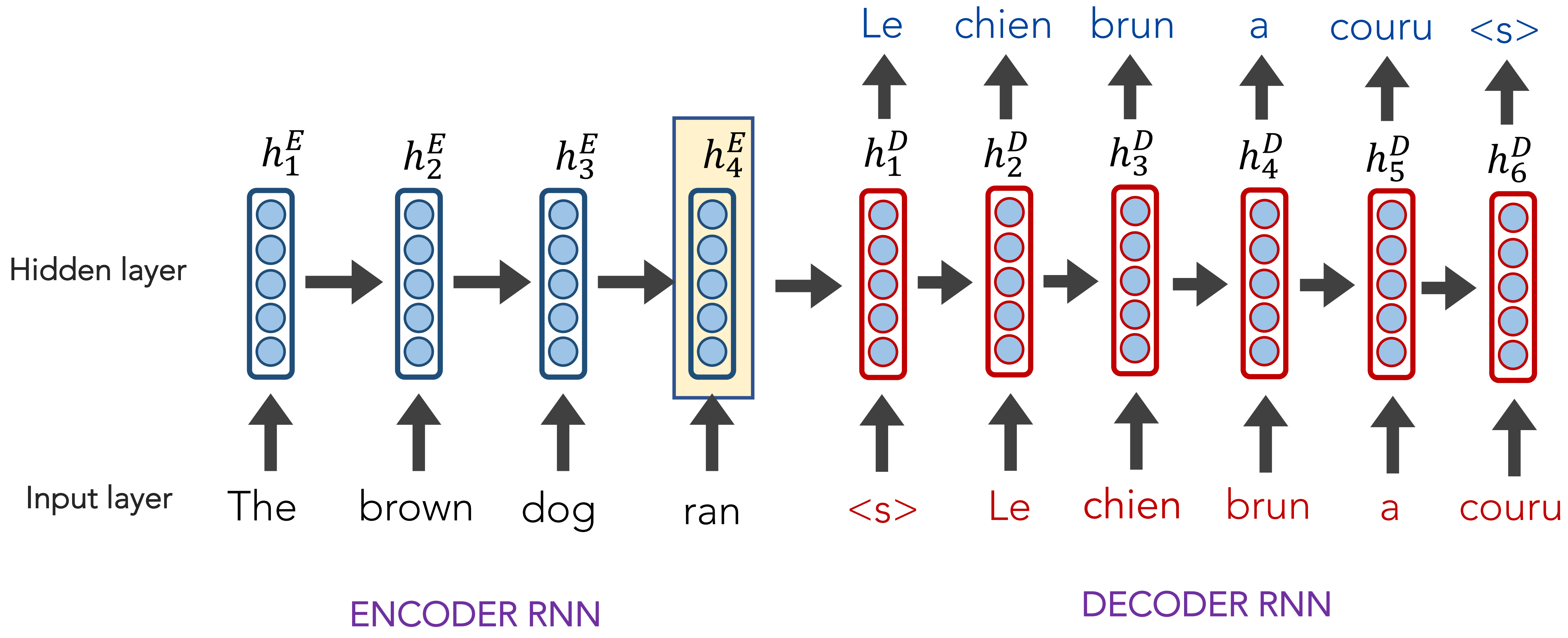


Sequence-to-Sequence (seq2seq)

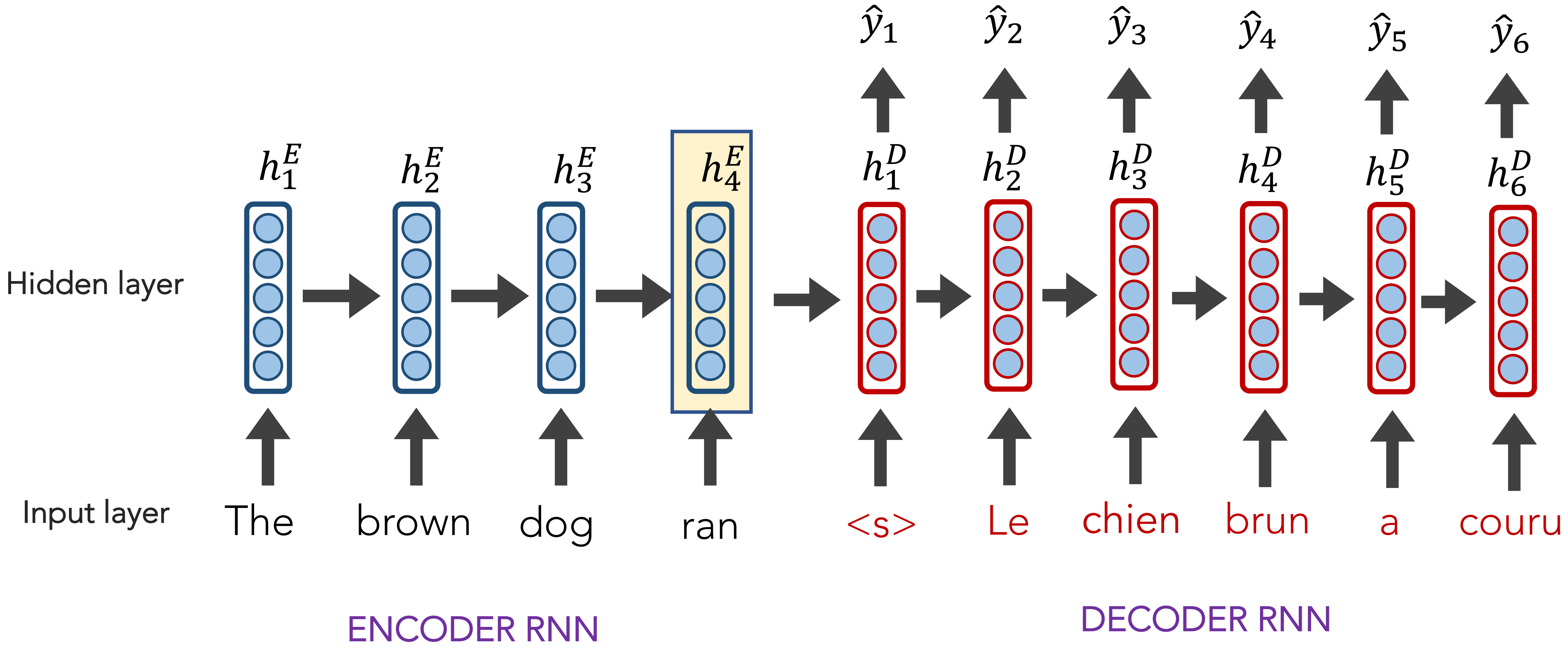


Sequence-to-Sequence (seq2seq)

The final hidden state of the encoder RNN
is the initial state of the decoder RNN

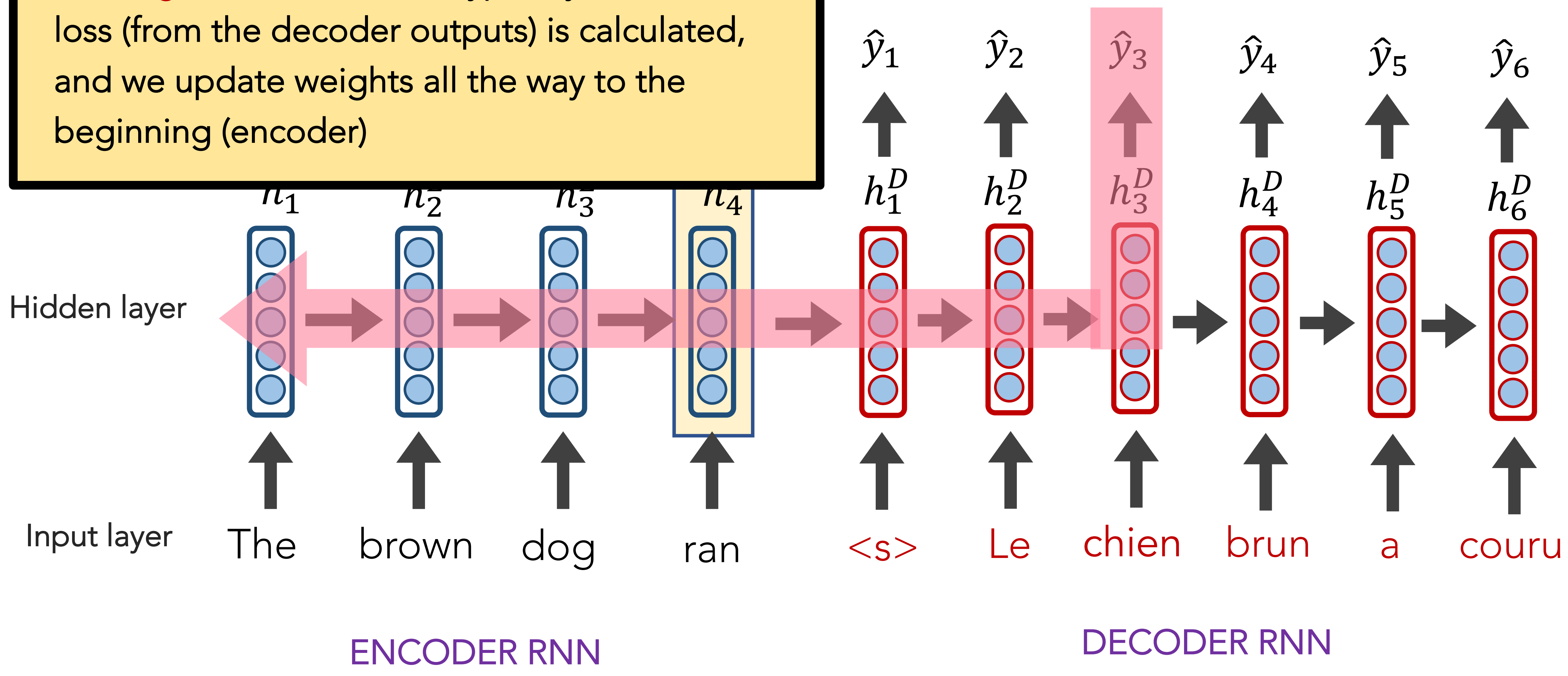


Sequence-to-Sequence (seq2seq)



Sequence-to-Sequence (seq2seq)

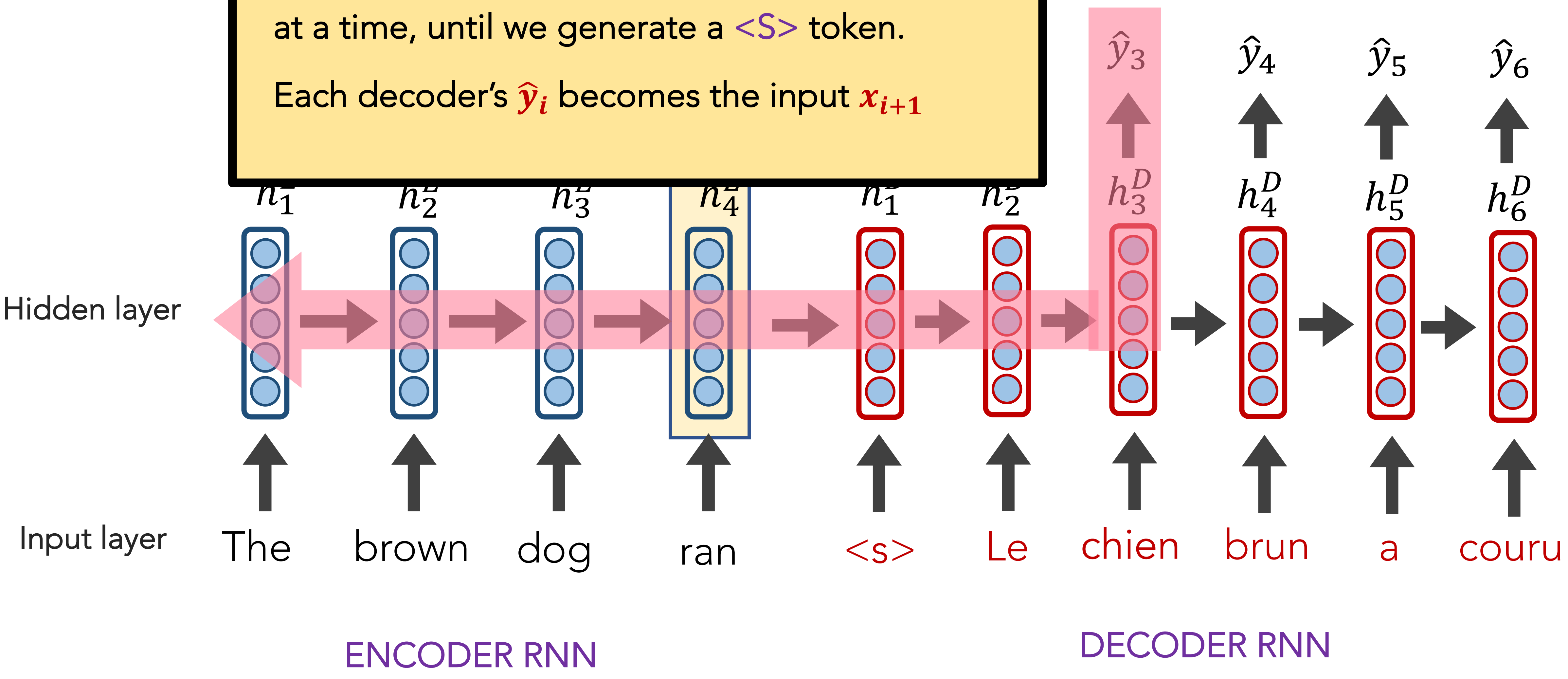
Training occurs like RNNs typically do; the loss (from the decoder outputs) is calculated, and we update weights all the way to the beginning (encoder)



Sequence-to-Sequence (seq2seq)

Testing generates decoder outputs one word at a time, until we generate a <S> token.

Each decoder's $\hat{y}_i$ becomes the input x_{i+1}



Greedy Decoding

What’s an issue w/ greedy decoding?

Greedy Decoding

She _____

She went _____

She went to _____

She went to class _____

$$O(|V|^n)$$

Beam Search Decoding

Sequentially consider the best (highest log likelihoods)
k translations at each step. Prune before expanding.

See chalkboard.

Beam Search Decoding

We can stop generating candidates when sequences are of length N, or when we have M *completed* sequences

Must normalize by lengths!

Neural MT

Pros:

- Better performance
- Uses context more robustly
- Better phrases
- Single model that can be optimized end-to-end (no subcomponents)
- Way less manual, feature engineering

Neural MT

Cons:

- Not too interpretable
- Hard to control/ force any Language-specific aspect
- A vanilla seq2seq approach can have gradient issues

MT Evaluation

BLEU: A similarity metric that compares the generated machine translation to a human-produced translation.

Uses n-gram precision (e.g., n=1,2,3,4,5)

Computer Generated: the dog

Target: the dog ran fast

MT Evaluation

BLEU: A similarity metric that compares the generated machine translation to a human-produced translation.

Uses n-gram precision (e.g., n=1,2,3,4,5)

Computer Generated: the dog

Target: the dog ran fast

Adds a penalty for translations that are too short (akin to recall)

2014 - present: NMT

SUMMARY

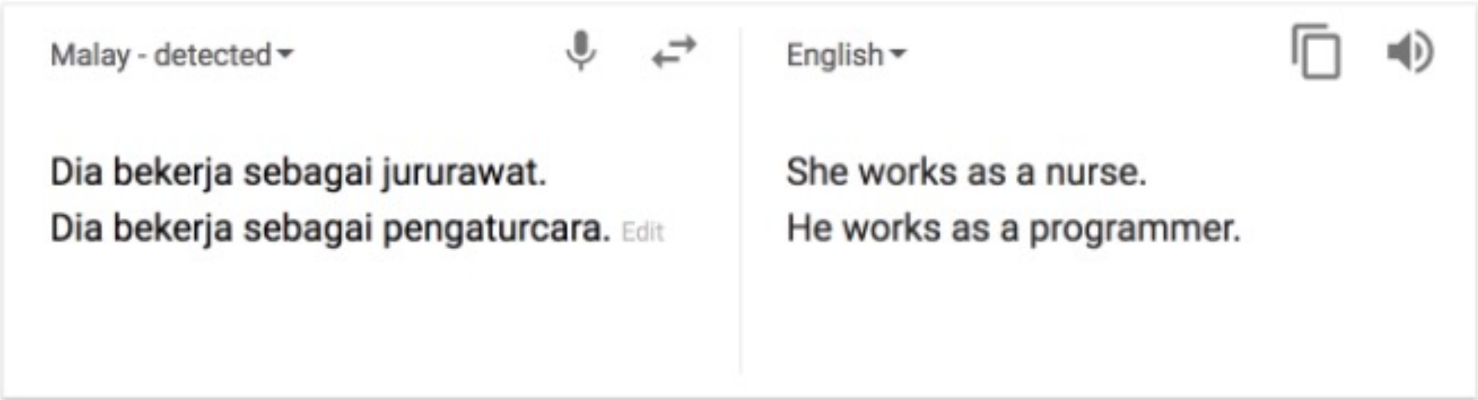
- Became SOTA in just 2 years
- OOV issues still need to be handled
- Susceptible to training data, as always (domain mismatch issues)
- Long-context is always difficult
- Low-resource languages still remains a challenge
- Biases from training data

MT Biases

Level 1: No context provided.

Both Google Translate and Microsoft Translator return

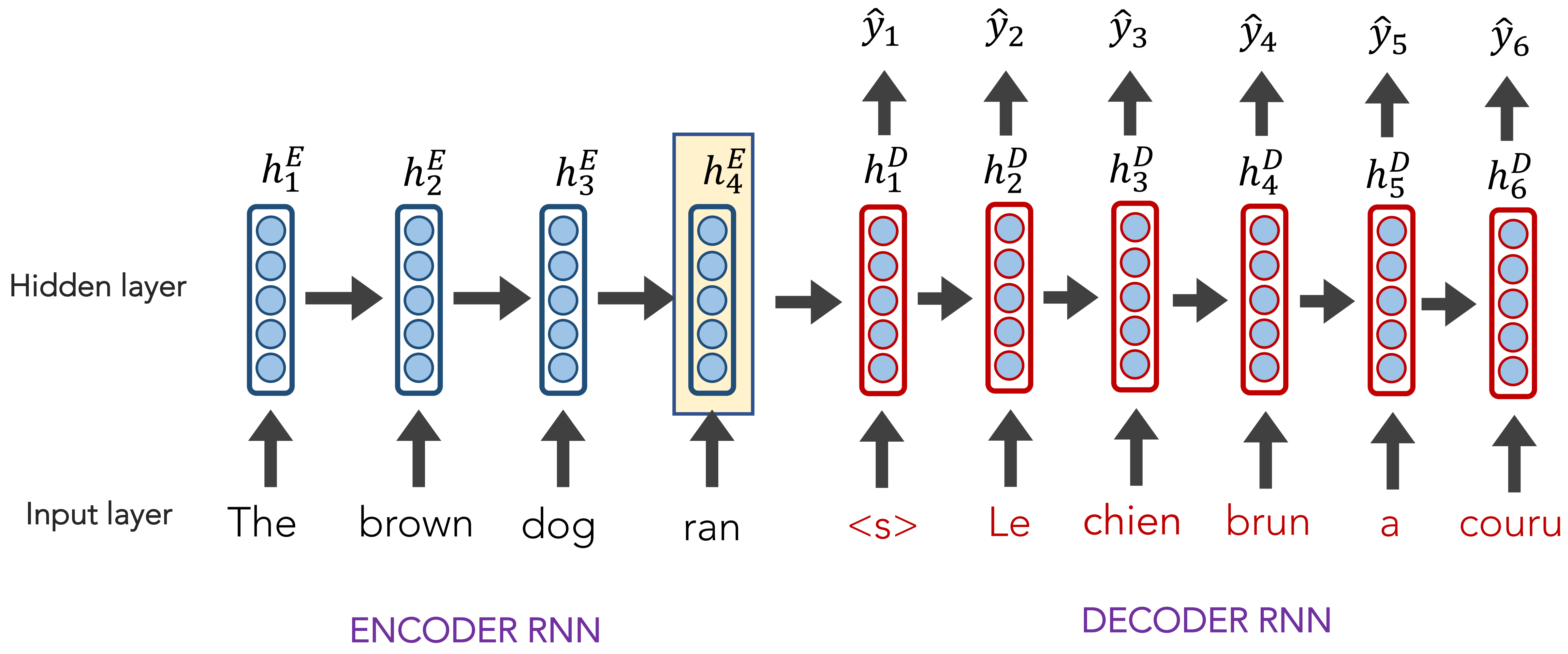
- **she** for **nurse**
- **he** for **programmer**



Both Google and Microsoft return same result.

https://hackernoon.com/bias-sexist-or-this-is-the-way-it-should-be-ce1f7c8c683c

Sequence-to-Sequence (seq2seq)

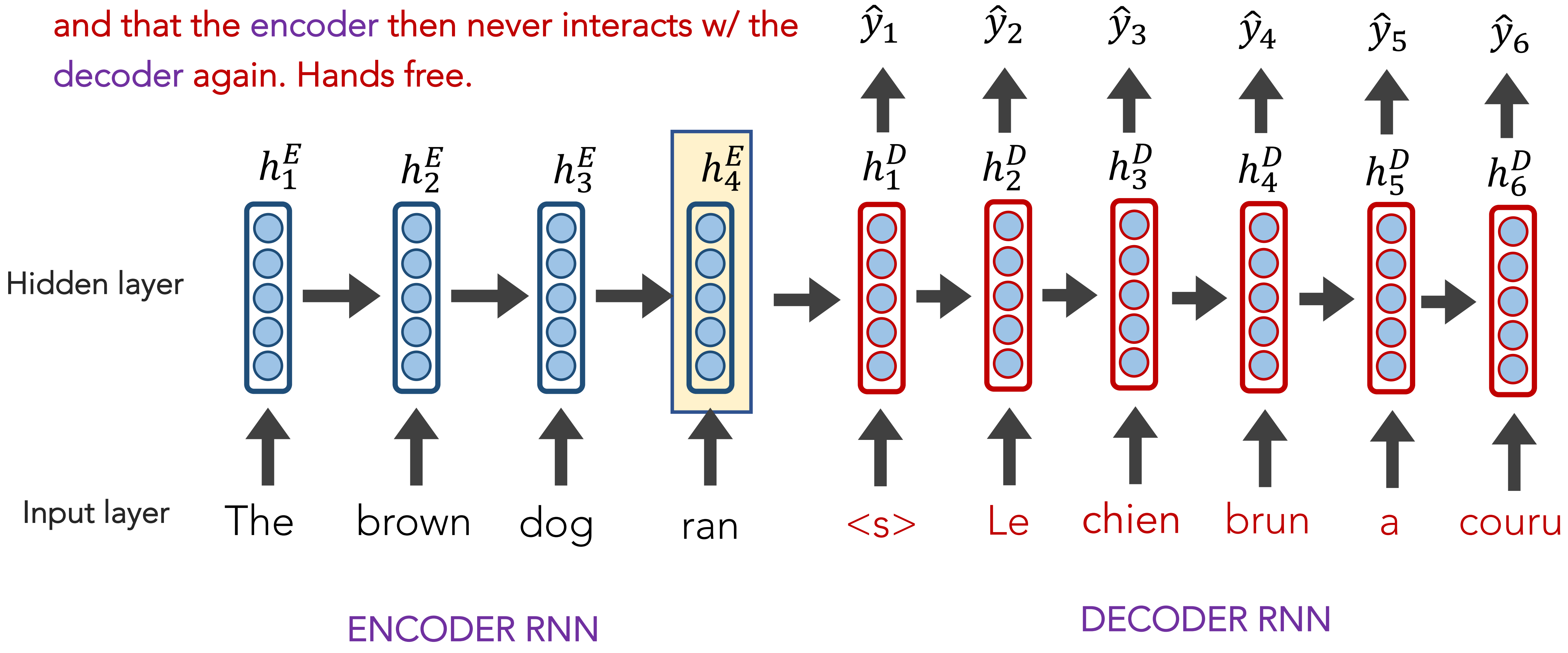


Sequence-to-Sequence (seq2seq)

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.



Outline

- Machine Translation
- seq2seq
- seq2seq + Attention

Outline

- Machine Translation
- seq2seq
- seq2seq + Attention

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?

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?

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.

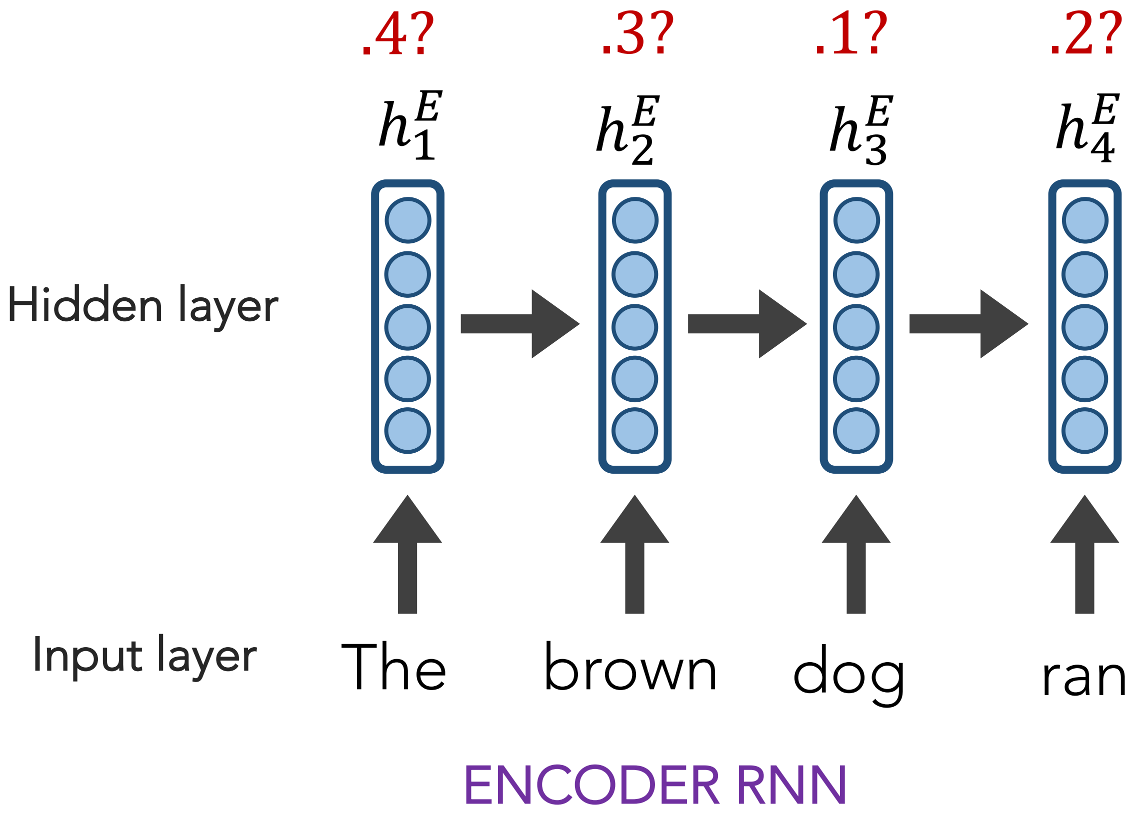
Attention

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

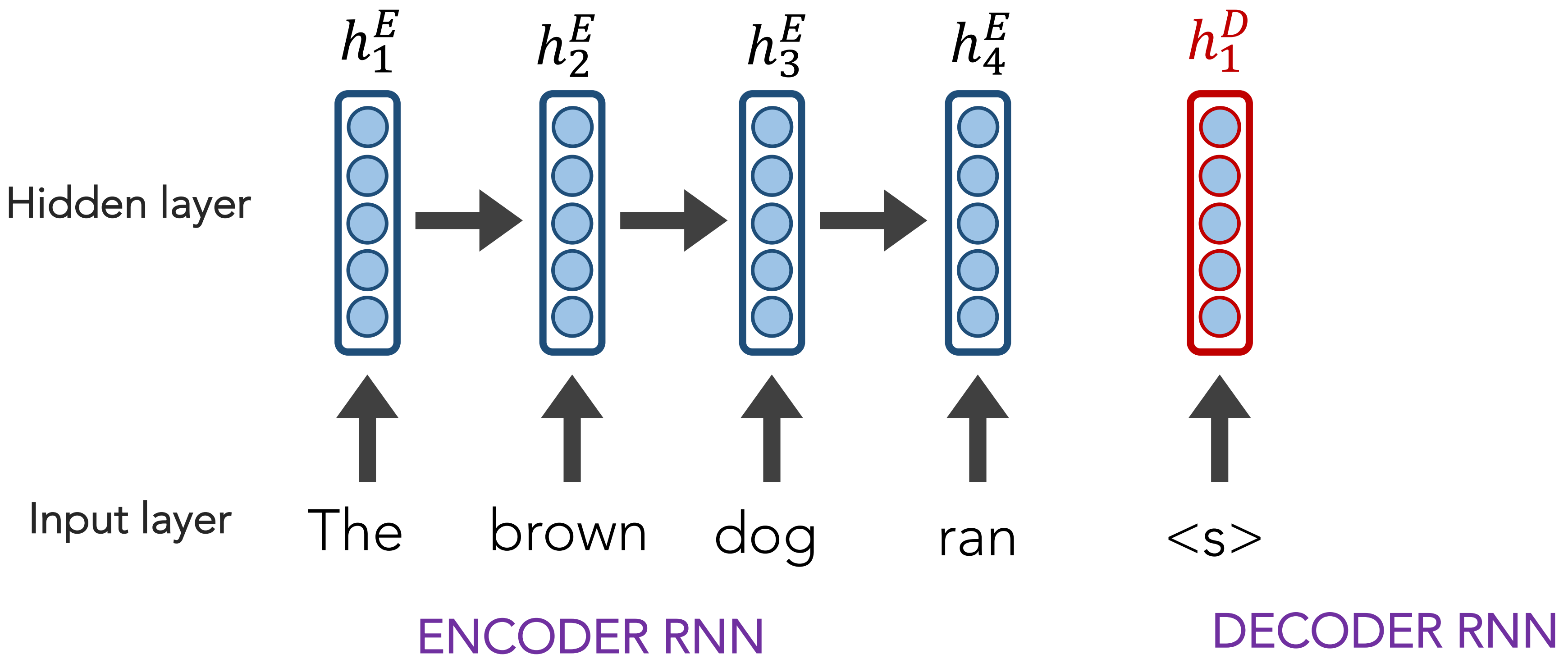
Q: How do we determine how much to pay attention to each of the encoder's hidden layers?



seq2seq + Attention

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

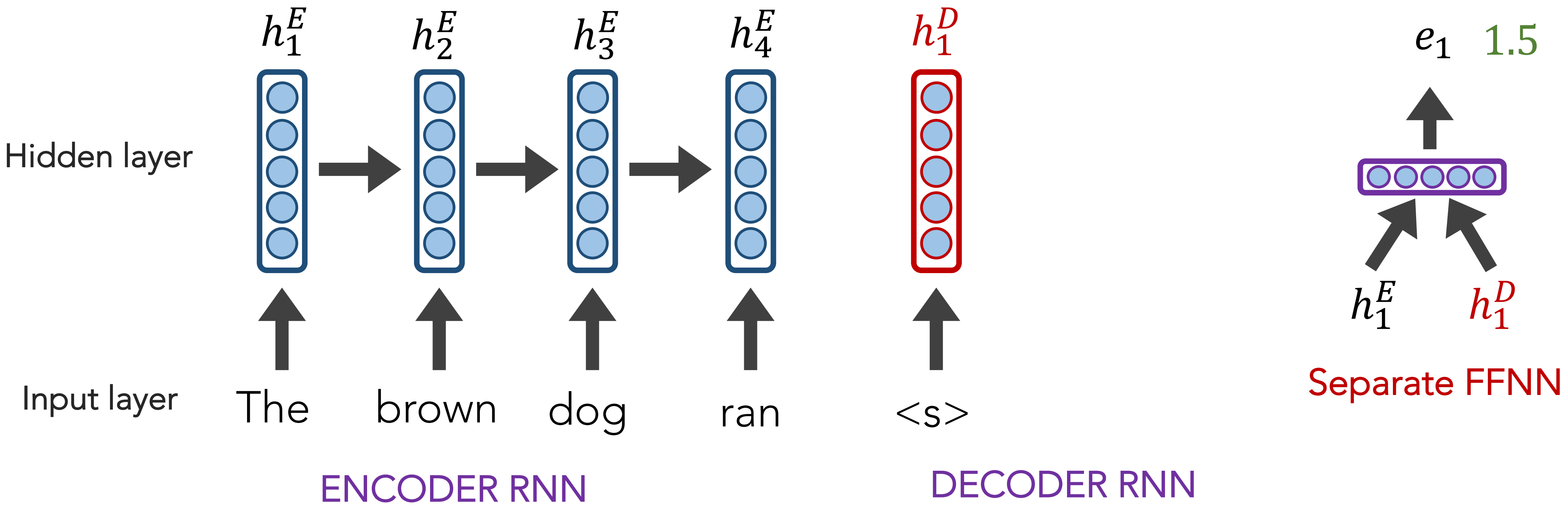
A: Let's base it on our decoder's current hidden state (our current representation of meaning) and all of the encoder's hidden layers!



seq2seq + Attention

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

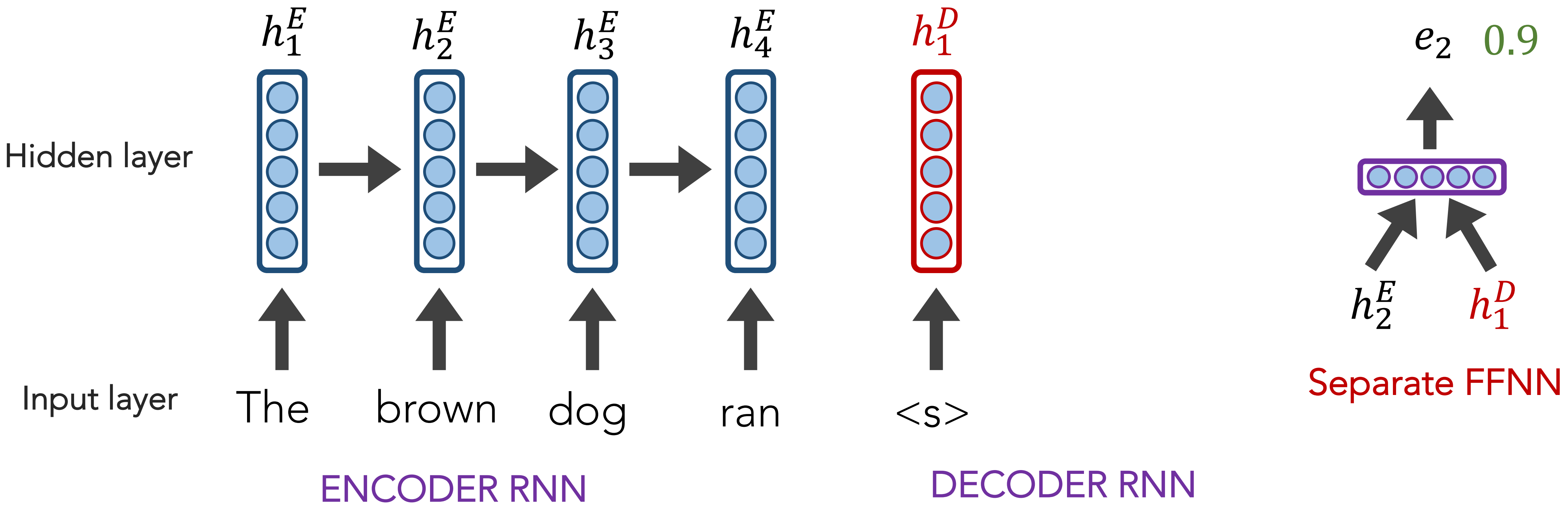
A: Let's base it on our decoder's current hidden state (our current representation of meaning) and all of the encoder's hidden layers!



seq2seq + Attention

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

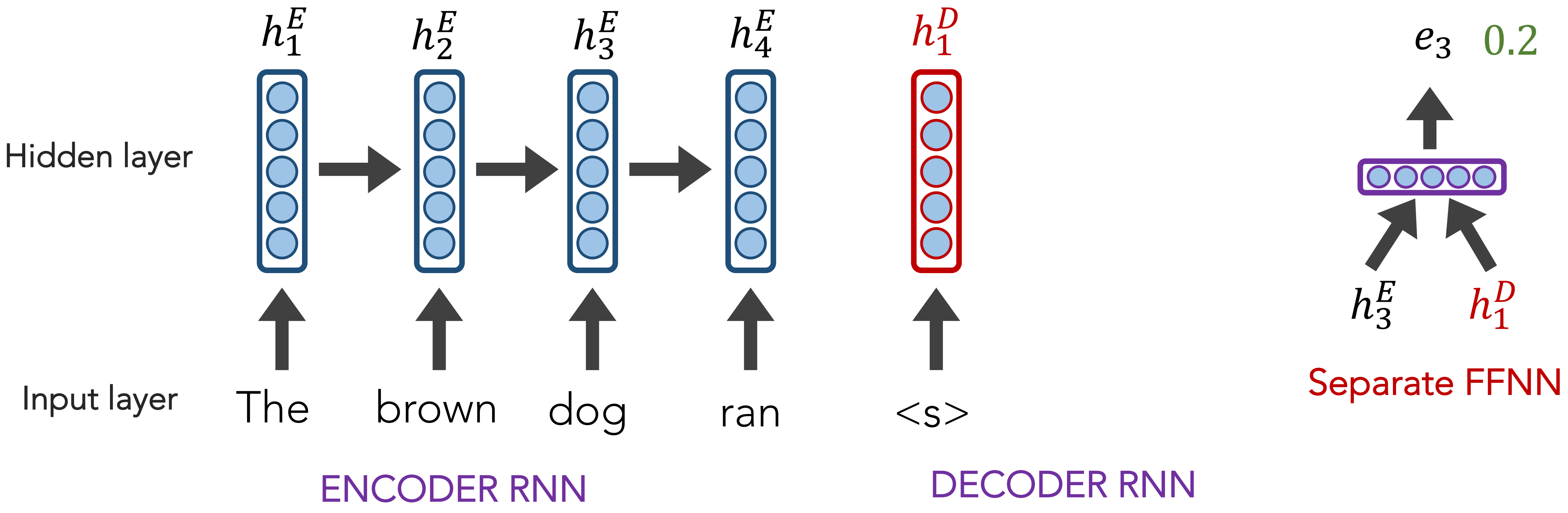
A: Let's base it on our decoder's current hidden state (our current representation of meaning) and all of the encoder's hidden layers!



seq2seq + Attention

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

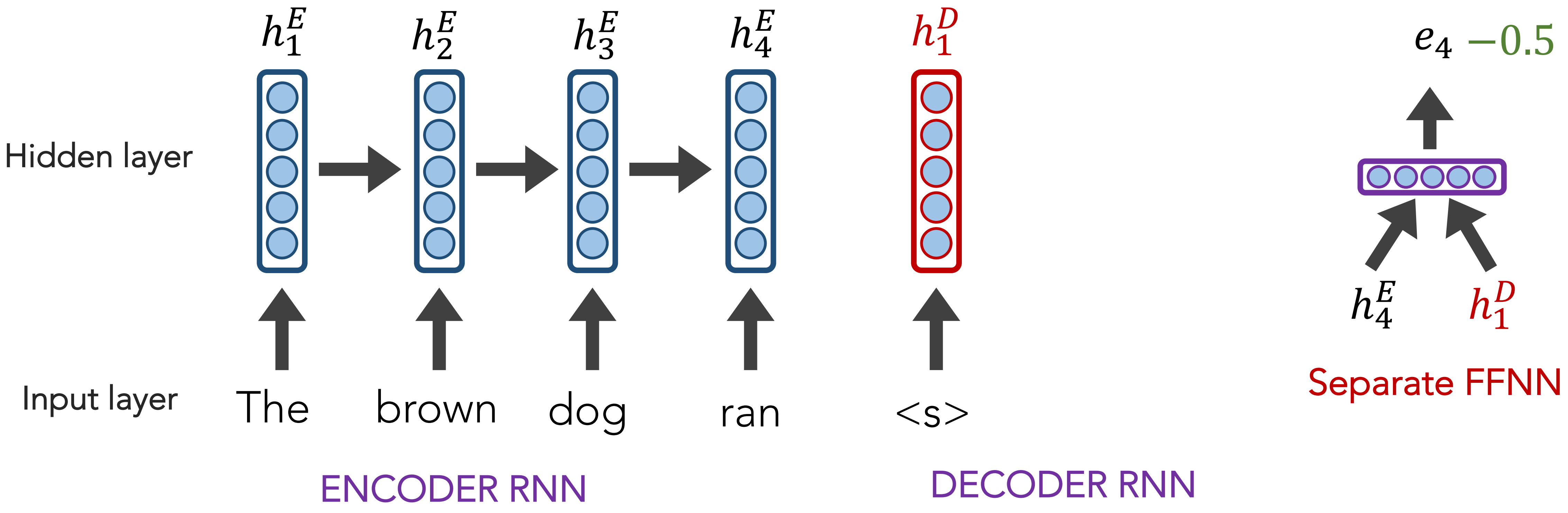
A: Let's base it on our decoder's current hidden state (our current representation of meaning) and all of the encoder's hidden layers!



seq2seq + Attention

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

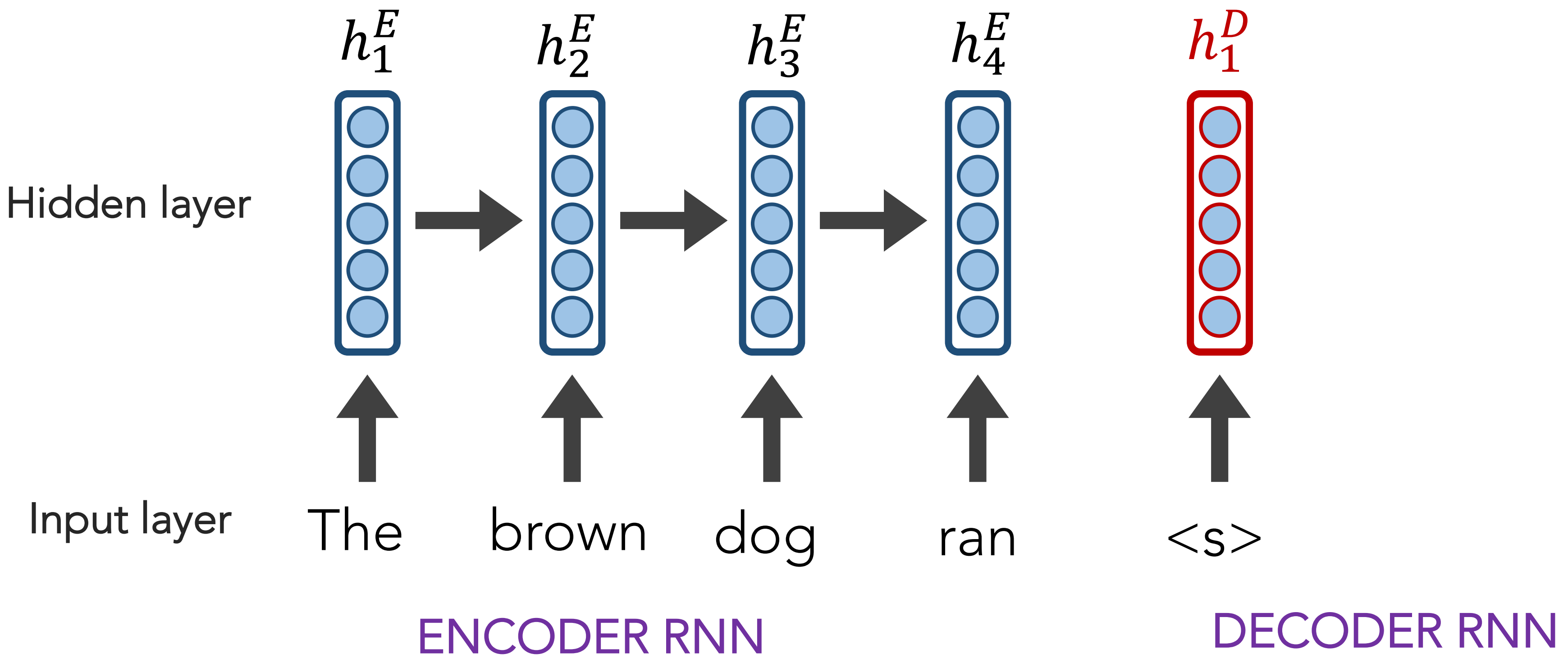
A: Let's base it on our decoder's current hidden state (our current representation of meaning) and all of the encoder's hidden layers!



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!



Attention (raw scores)

e_1 1.5

e_2 0.9

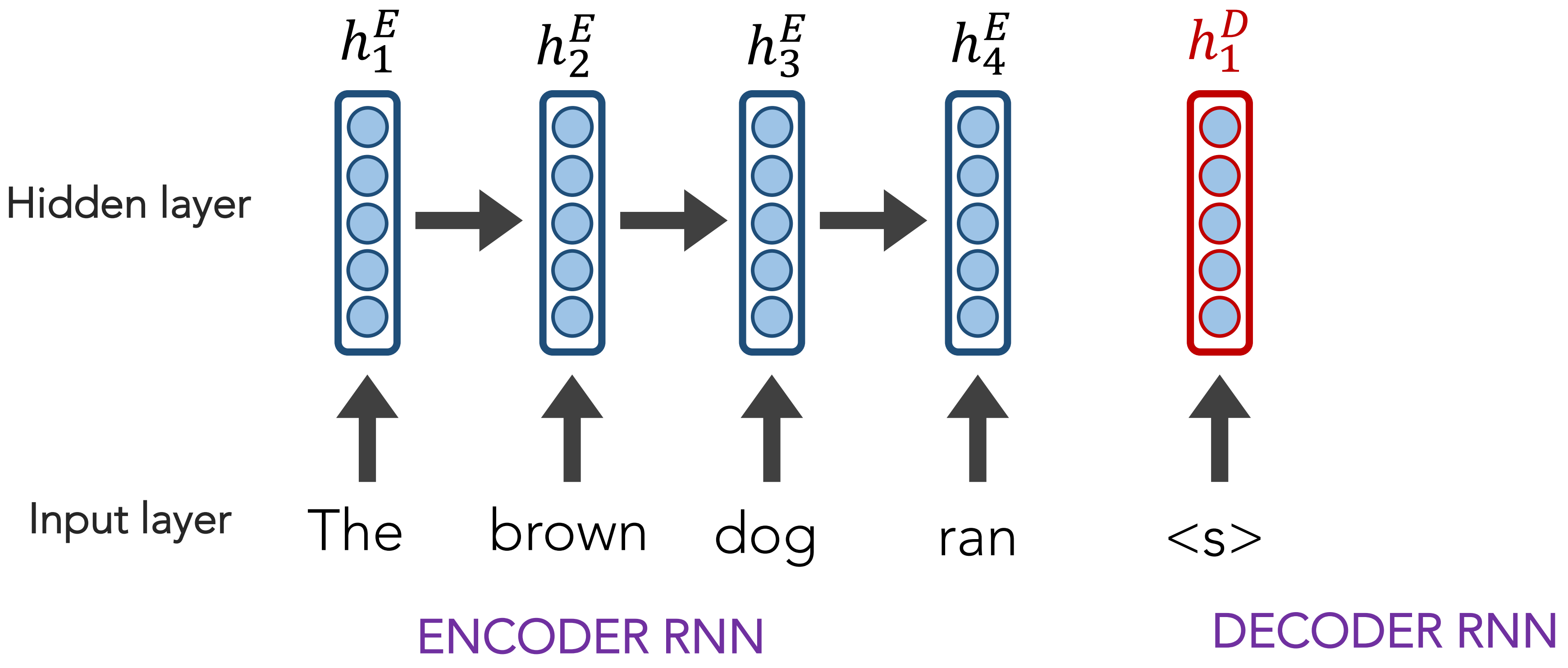
e_3 0.2

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 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

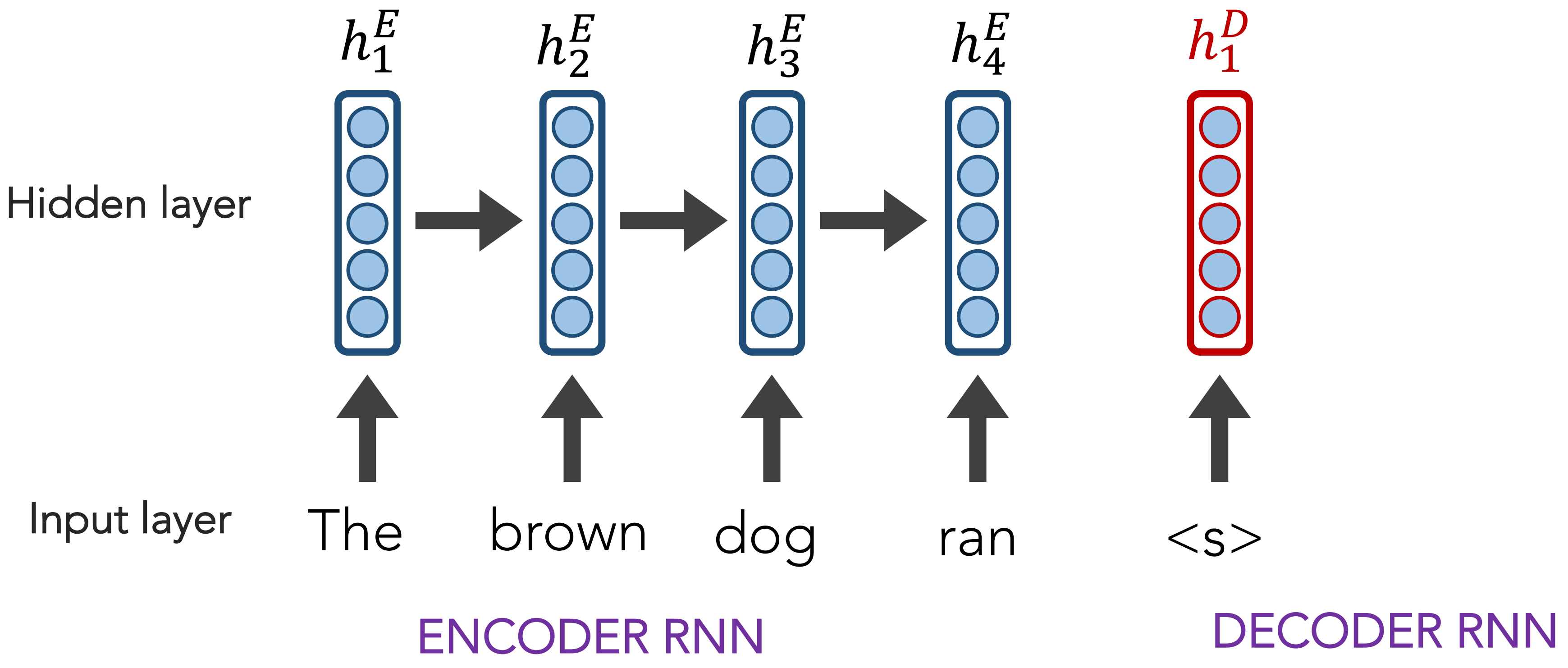
Attention (softmax'd)

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

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)

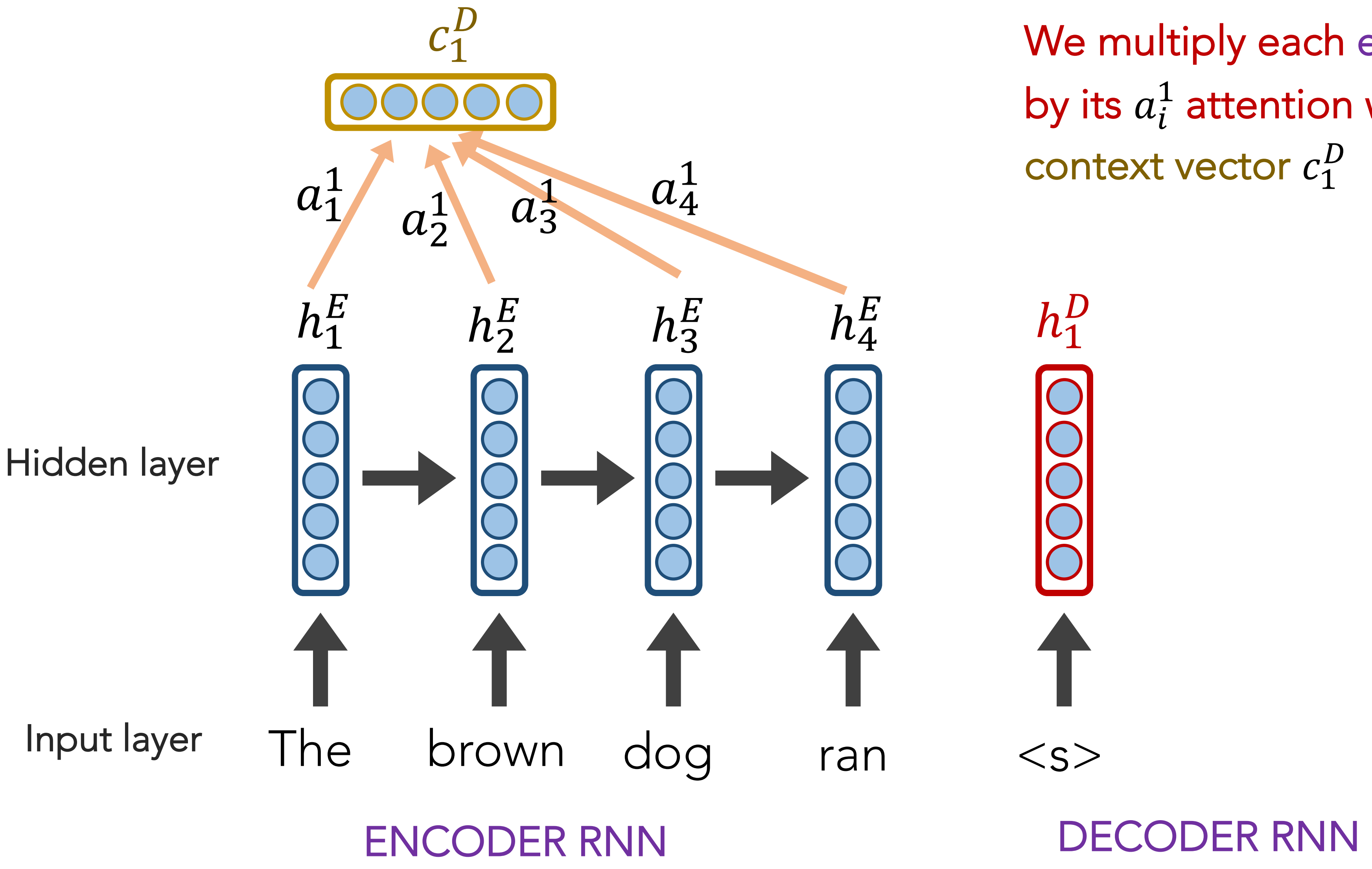
$a_1^1 = 0.51$

$a_2^1 = 0.28$

$a_3^1 = 0.14$

$a_3^1 = 0.07$

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)

$a_1^1 = 0.51$

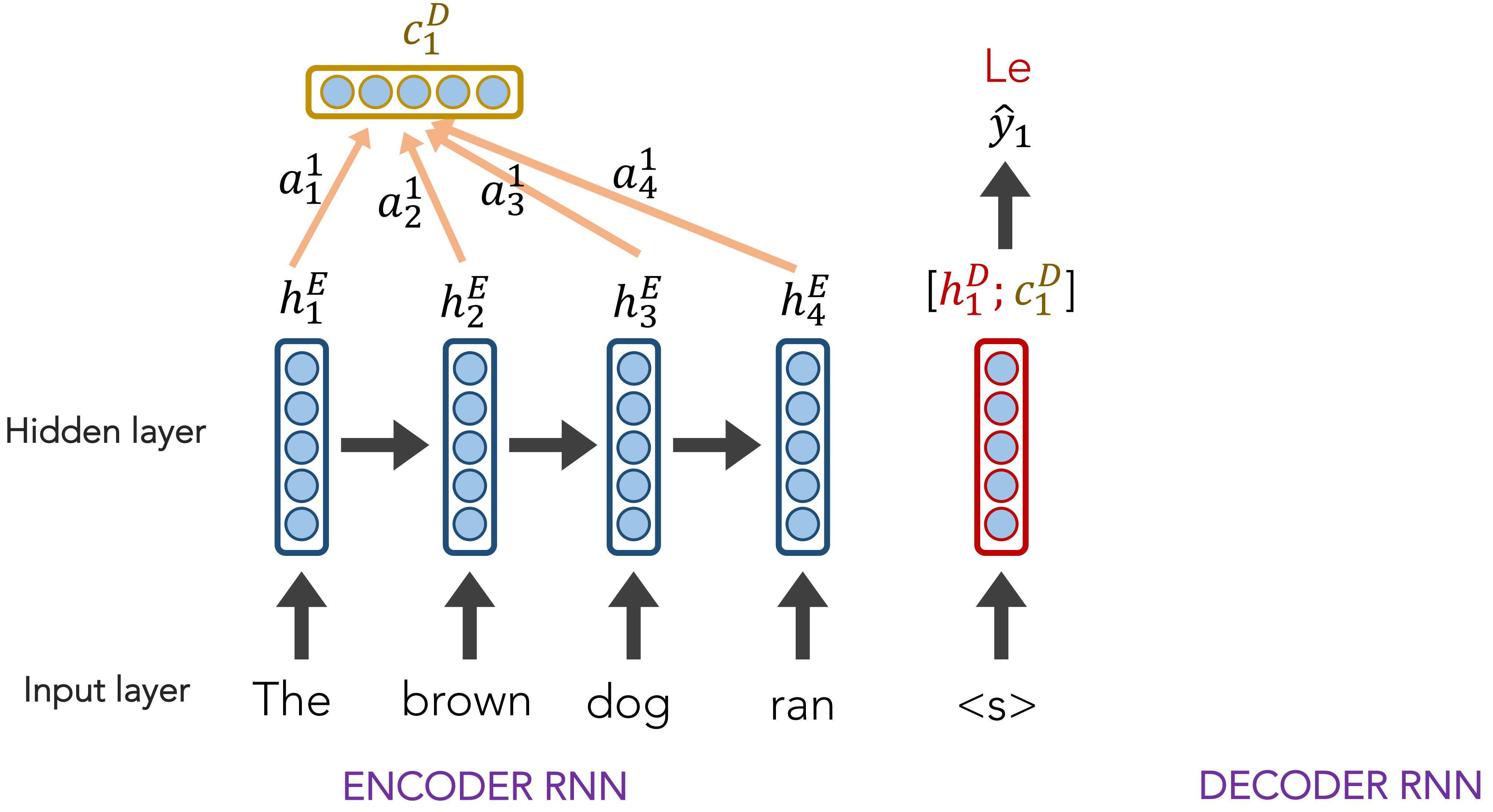
$a_2^1 = 0.28$

$a_3^1 = 0.14$

$a_3^1 = 0.07$

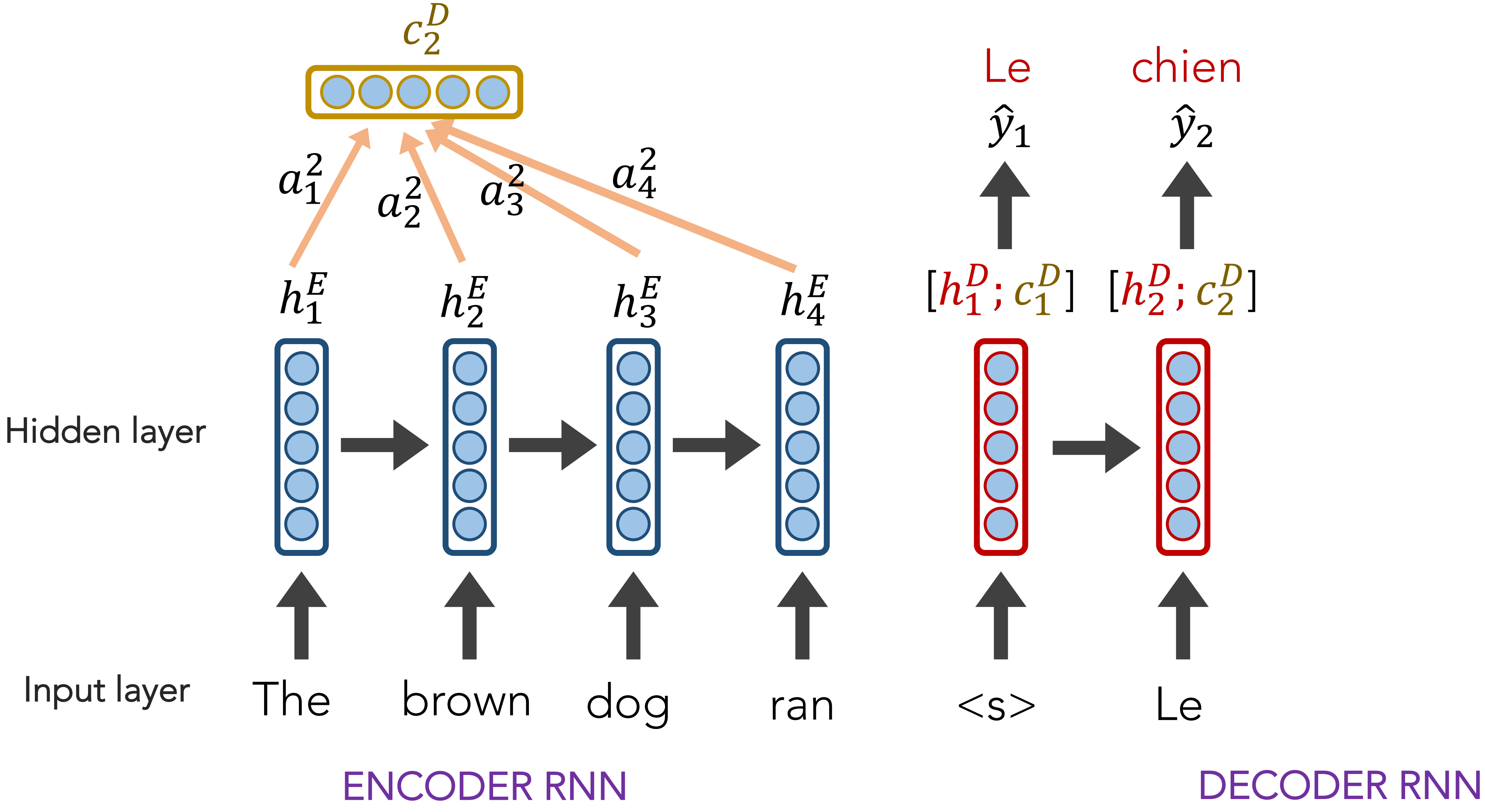
seq2seq + Attention

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



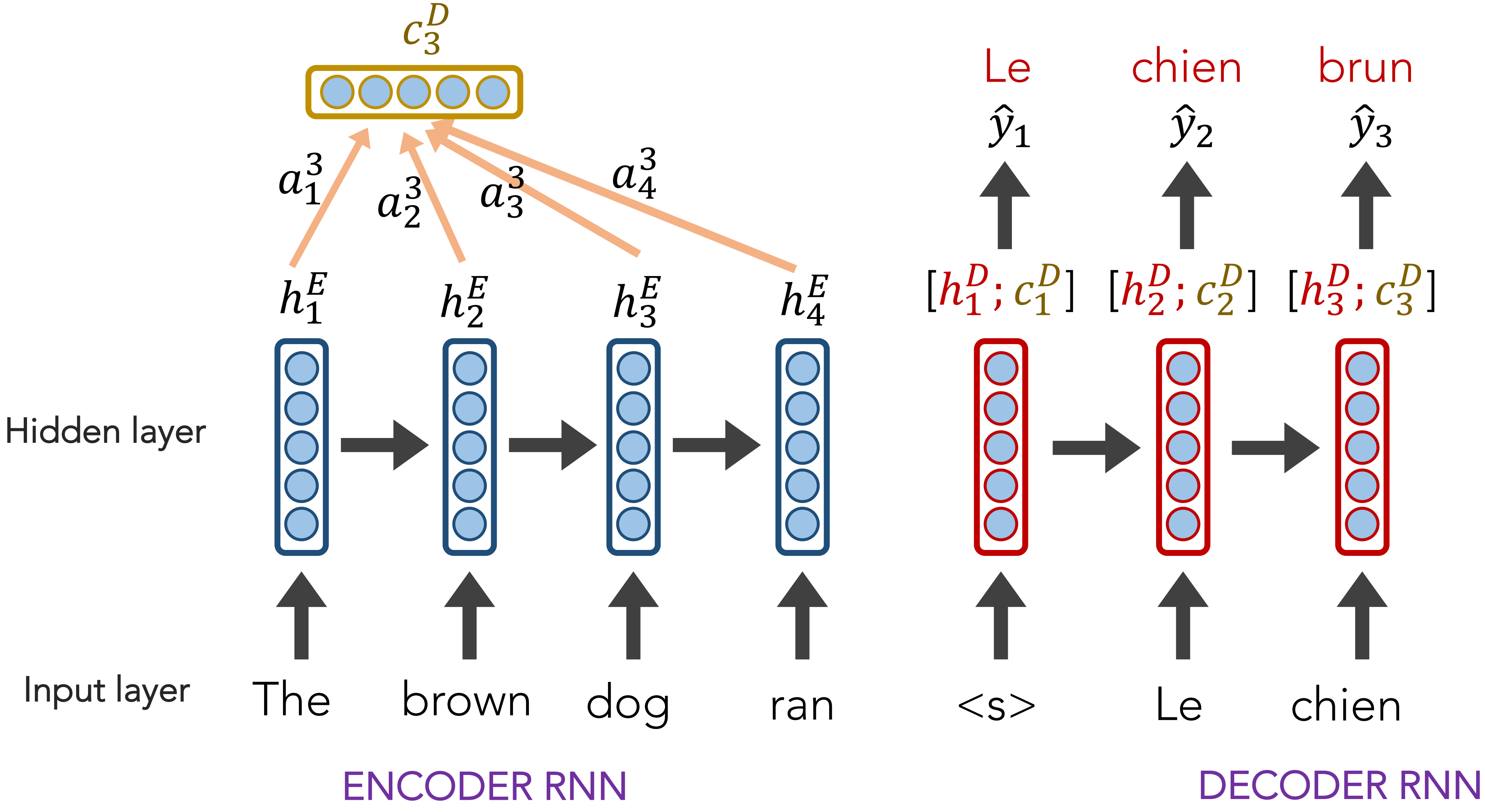
seq2seq + Attention

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



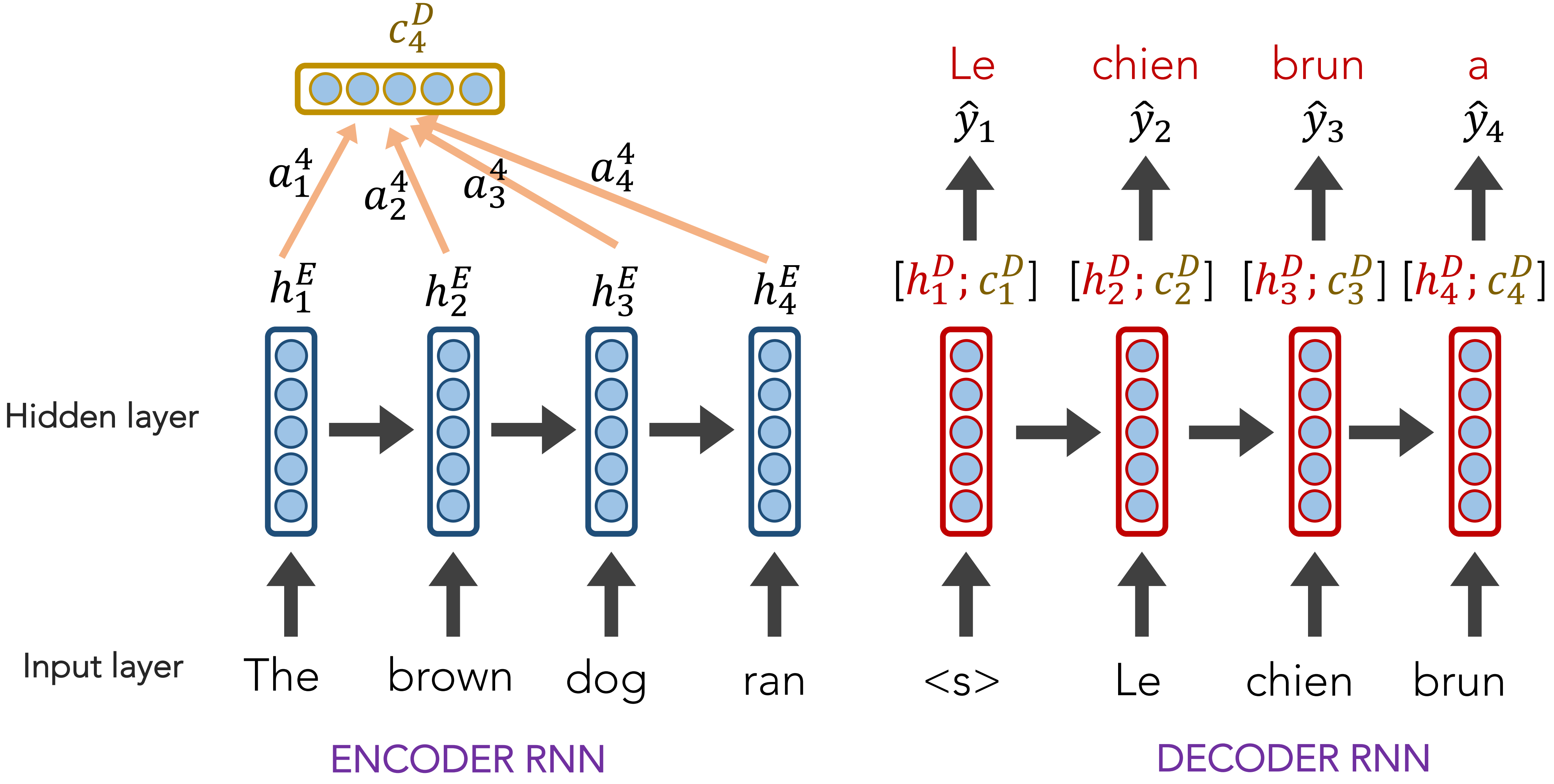
seq2seq + Attention

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



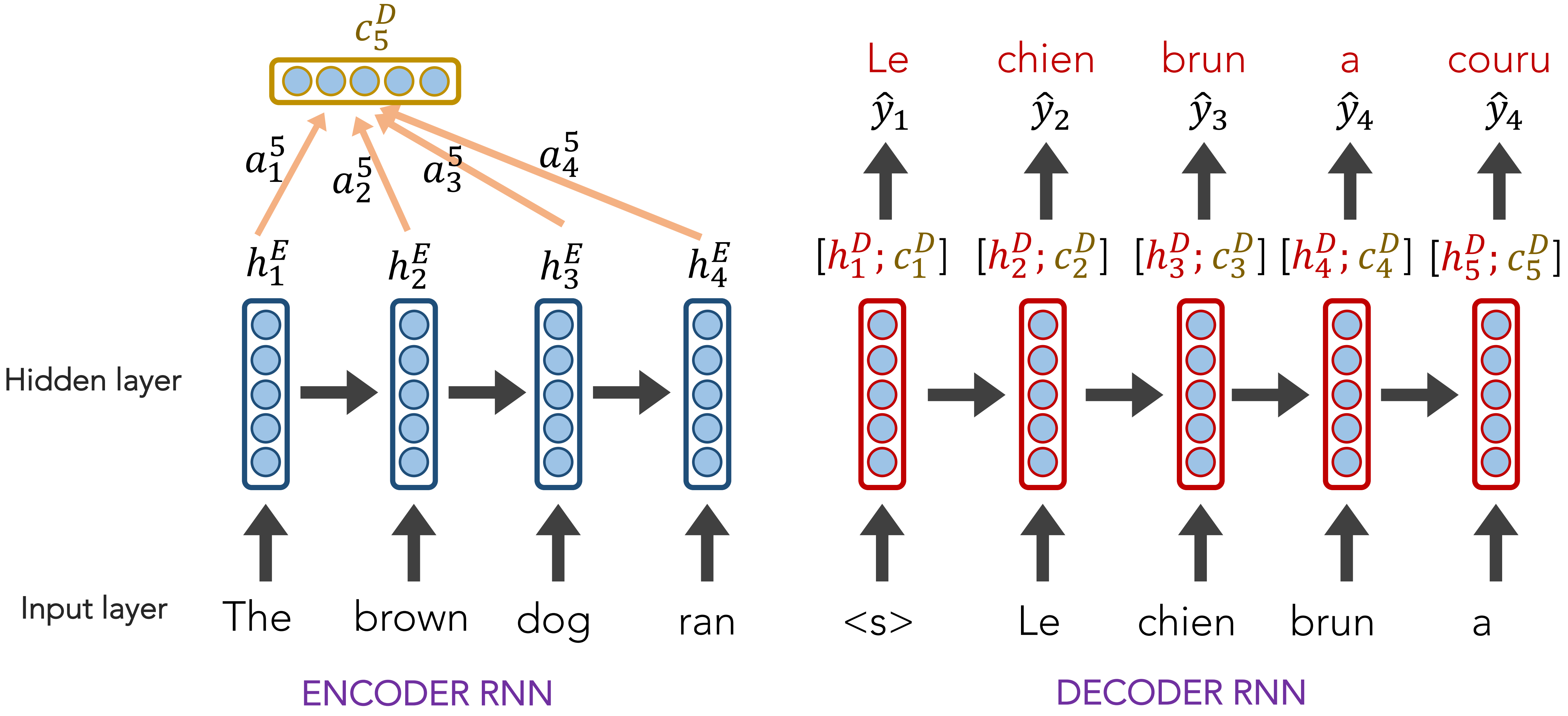
seq2seq + Attention

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

Attention output

$$c^{(t)} = a_1^{(t)} s_1 + a_2^{(t)} s_2 + \cdots + a_m^{(t)} s_m = \sum_{k=1}^{m} a_k^{(t)} s_k$$

"source context for decoder step t"

(weighted sum)

Attention weights

$$a_k^{(t)} = \frac{\exp(\text{score}(h_t, s_k))}{\sum_{i=1}^{m} \exp(\text{score}(h_t, s_i))}, \mathrm{k} = 1..\mathrm{m}$$

"attention weight for source token k at decoder step t"

(softmax)

Attention scores

$$\text{score}(h_t, s_k), \mathrm{k} = 1..\mathrm{m}$$

"How relevant is source token k for target step t?"

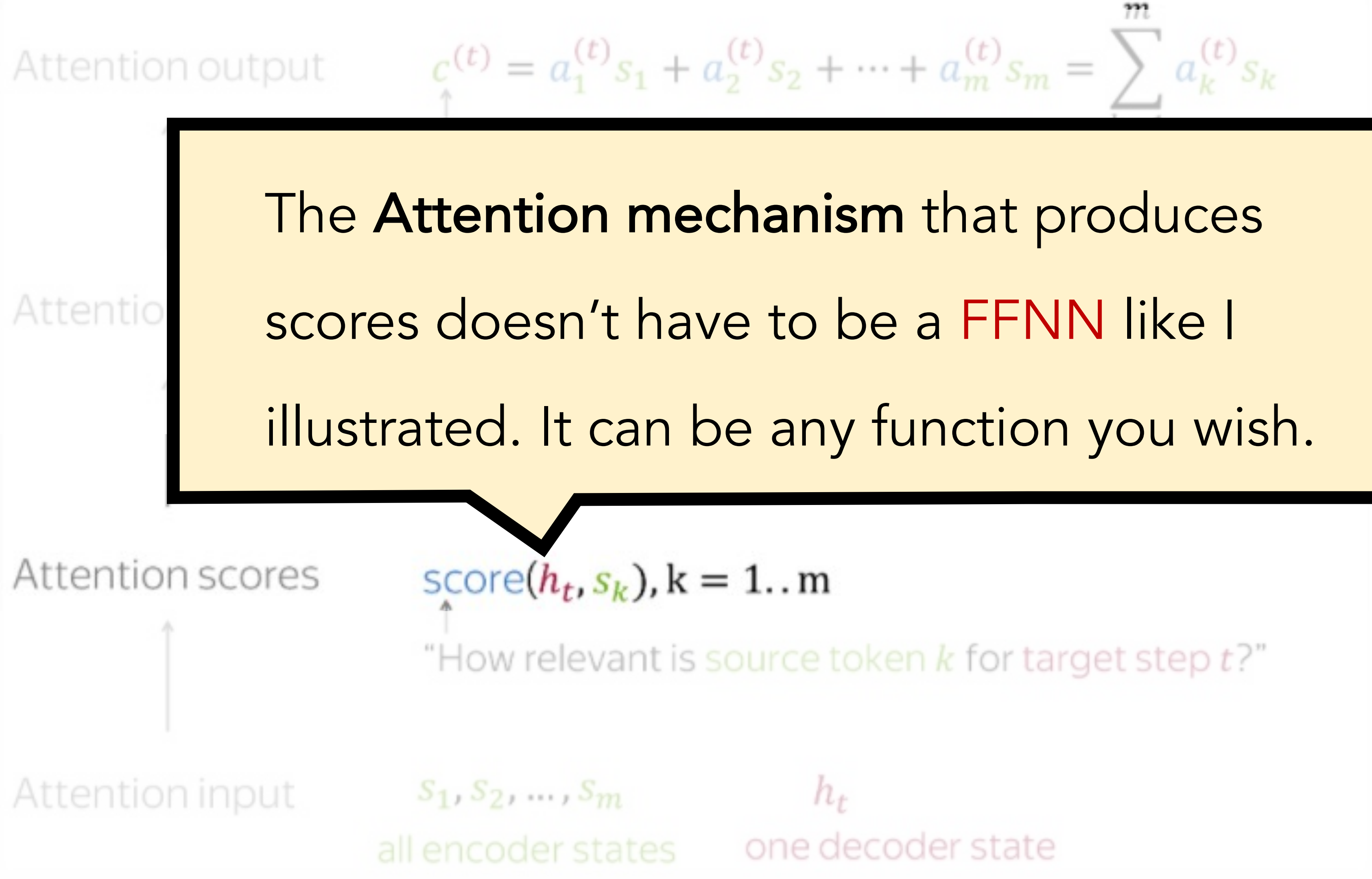
Attention input

$s_1, s_2, \ldots, s_m$ all encoder states

h_t one decoder state

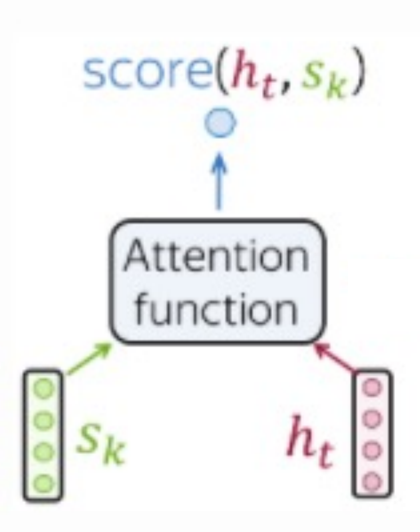
Photo credit: https://lena-voita.github.io/nlp_course/seq2seq_and_attention.html

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



The **Attention mechanism** that produces scores doesn't have to be a FFNN like I illustrated. It can be any function you wish.

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



Popular Attention Scoring functions:

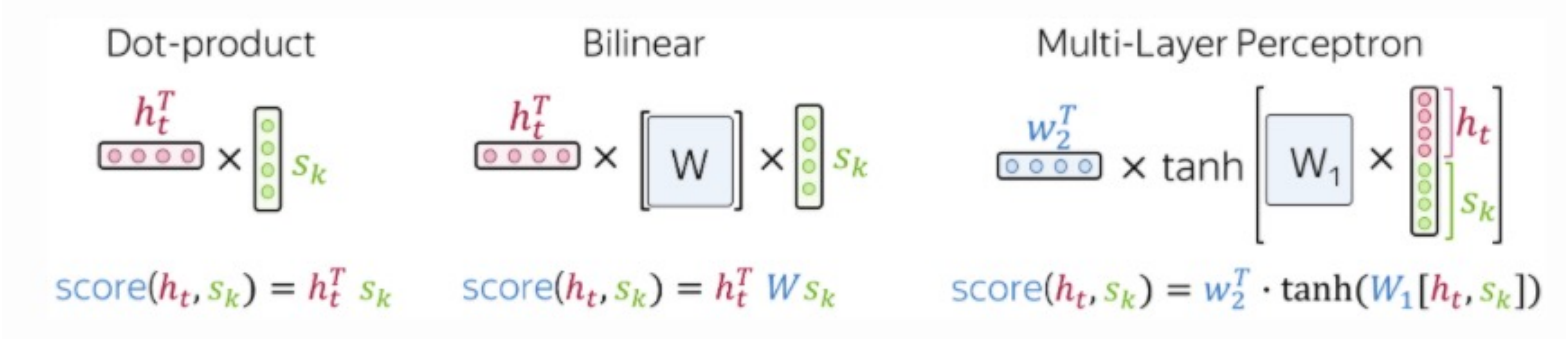


Photo credit: https://lena-voita.github.io/nlp_course/seq2seq_and_attention.html

seq2seq + Attention

Attention:

- greatly improves seq2seq results
- allows us to visualize the contribution each encoding word gave for each decoder's word

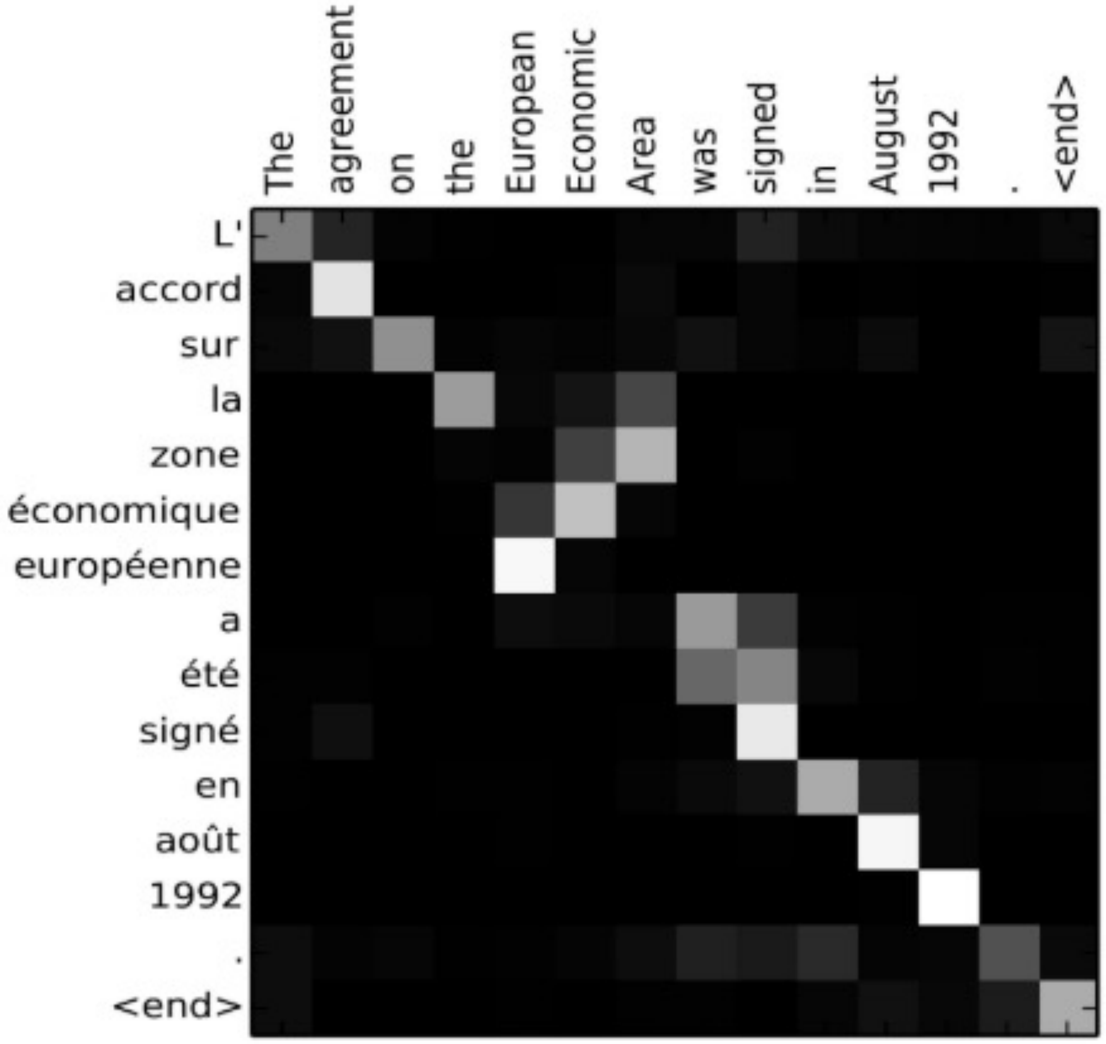


Image source: Fig 3 in [Bahdanau et al., 2015]

Takeaway:

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

Attention is powerful; allows us to conditionally weight our focus

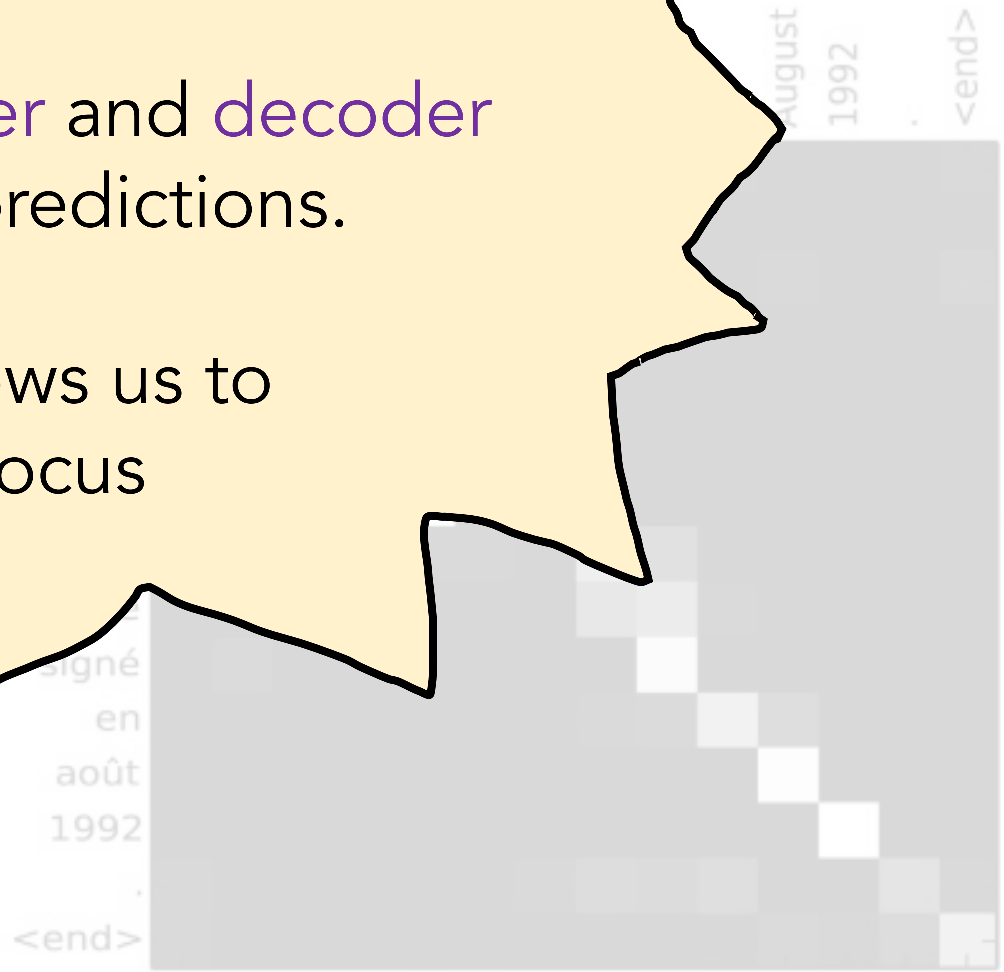
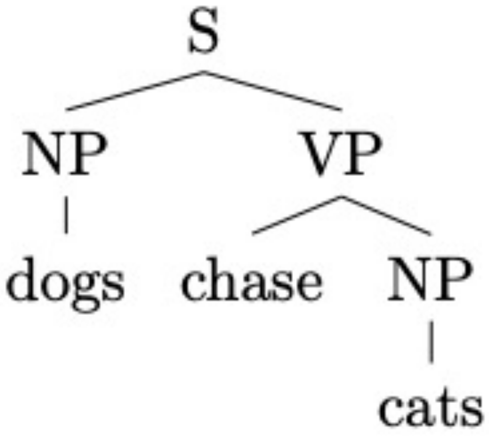


Image source: Fig 3 in [Bahdanau et al., 2015]

Constituency Parsing

Input: dogs chase cats

Output:



or a flattened representation

$$(\text{S}\ (\text{NP dogs}\ )_{\text{NP}}\ (\text{VP chase}\ (\text{NP cats}\ )_{\text{NP}}\ )_{\text{VP}}\ )_{\text{S}}$$

Constituency Parsing

Input: I shot an elephant in my pajamas

Output:

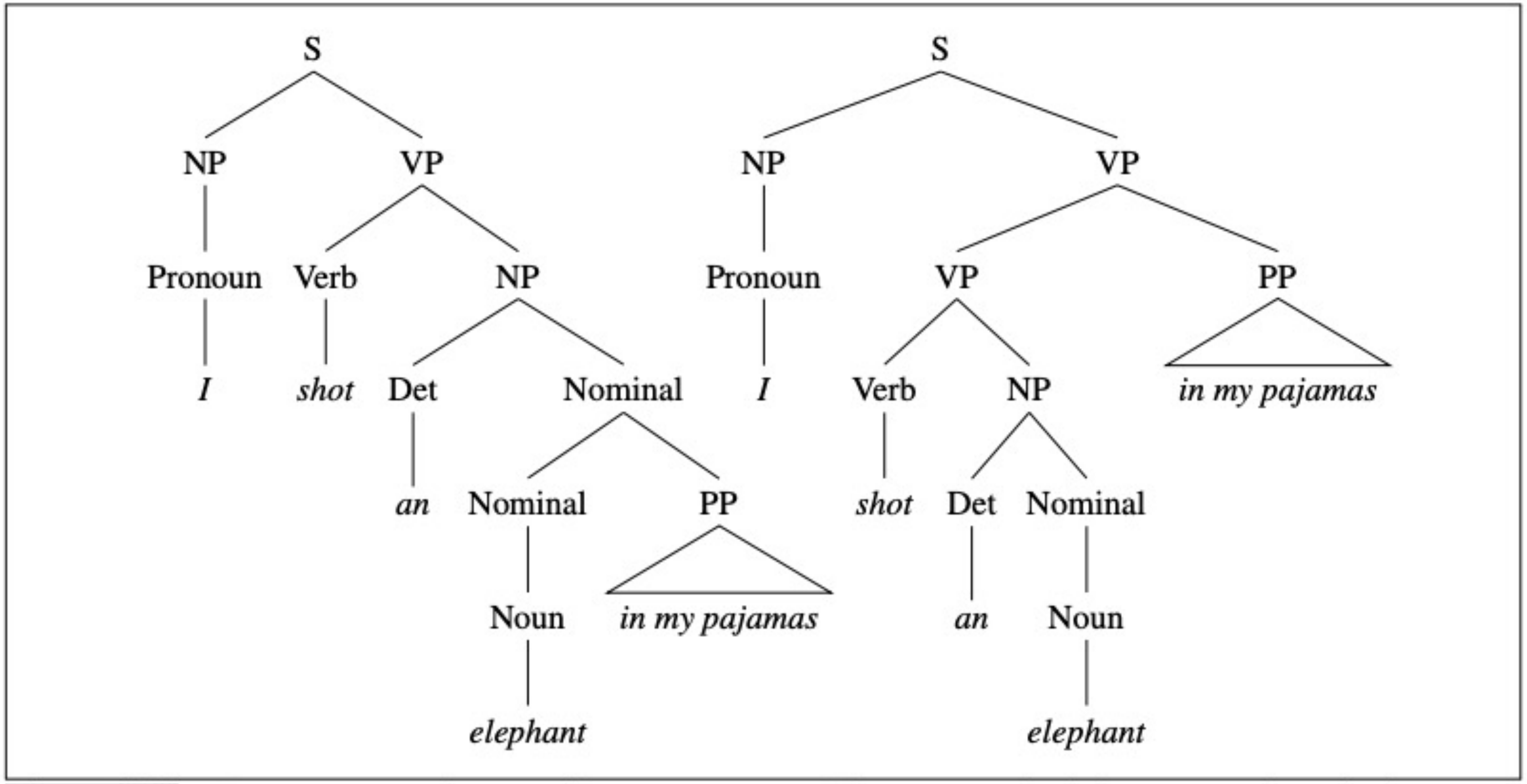


Figure 13.2 Two parse trees for an ambiguous sentence. The parse on the left corresponds to the humorous reading in which the elephant is in the pajamas, the parse on the right corresponds to the reading in which Captain Spaulding did the shooting in his pajamas.

https://web.stanford.edu/~jurafsky/slp3/13.pdf

Results

Model	English			Chinese		
	LR	LP	F1	LR	LP	F1
Shen et al. (2018)	92.0	91.7	91.8	86.6	86.4	86.5
Fried and Klein (2018)	-	-	92.2	-	-	87.0
Teng and Zhang (2018)	92.2	92.5	92.4	86.6	88.0	87.3
Vaswani et al. (2017)	-	-	92.7	-	-	-
Dyer et al. (2016)	-	-	93.3	-	-	84.6
Kuncoro et al. (2017)	-	-	93.6	-	-	-
Charniak et al. (2016)	-	-	93.8	-	-	-
Liu and Zhang (2017b)	91.3	92.1	91.7	85.9	85.2	85.5
Liu and Zhang (2017a)	-	-	94.2	-	-	86.1
Suzuki et al. (2018)	-	-	94.32	-	-	-
Takase et al. (2018)	-	-	94.47	-	-	-
Fried et al. (2017)	-	-	94.66	-	-	-
Kitaev and Klein (2018)	94.85	95.40	95.13	-	-	-
Kitaev et al. (2018)	95.51	96.03	95.77	91.55	91.96	91.75
Zhou and Zhao (2019) (BERT)	95.70	95.98	95.84	**92.03**	92.33	92.18
Zhou and Zhao (2019) (XLNet)	96.21	96.46	96.33	-	-	-
Our work	**96.24**	**96.53**	**96.38**	91.85	**93.45**	**92.64**

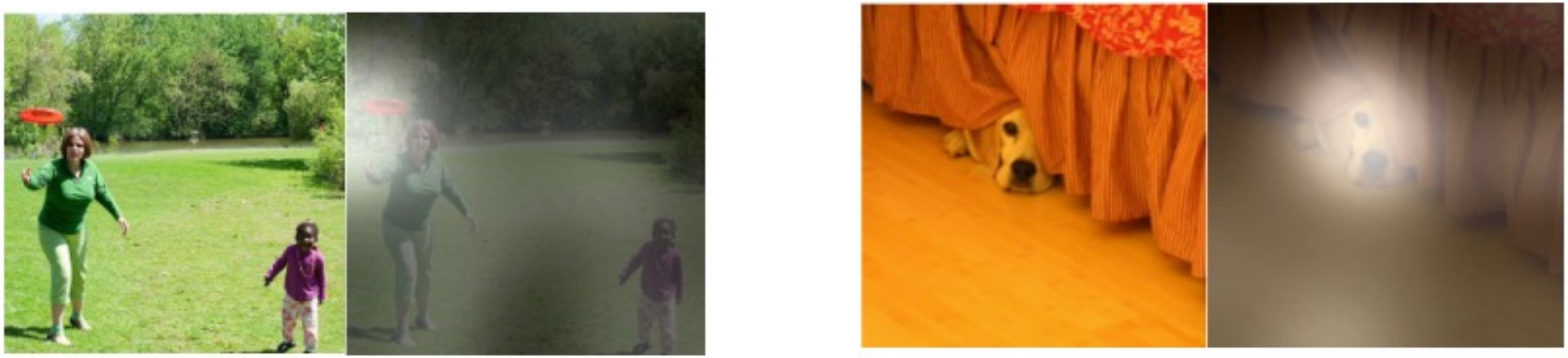
Table 3: Constituency Parsing on PTB & CTB test sets.

https://aclanthology.org/2020.findings-emnlp.65.pdf

Image Captioning

Input: image

Output: generated text



A woman is throwing a frisbee in a park.

A dog is standing on a hardwood floor.

Figure 3. Examples of attending to the correct object (*white* indicates the attended regions, *underlines* indicated the corresponding word)

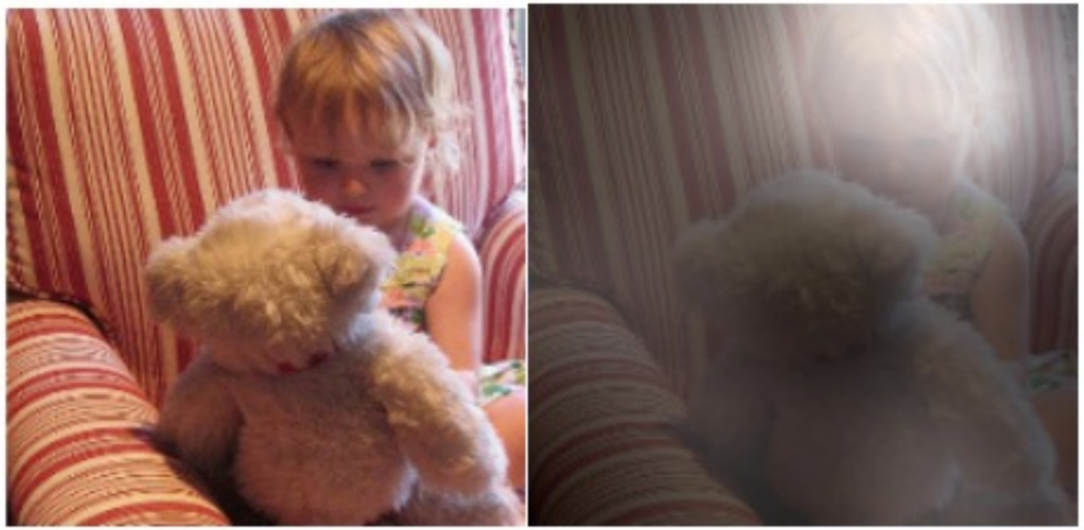
Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. Xu et al. CVPR (2016)

Image Captioning

Input: image

Output: generated text

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



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

Figure 3. Examples of attending to the correct object (*white* indicates the attended regions, *underlines* indicated the corresponding word)

Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. Xu et al. CVPR (2016)

Image Captioning

Figure 5. Examples of mistakes where we can use attention to gain intuition into what the model saw.

Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. Xu et al. CVPR (2016)

Image Captioning

Figure 5. Examples of mistakes where we can use attention to gain intuition into what the model saw.

Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. Xu et al. CVPR (2016)

SUMMARY

- LSTMs yielded state-of-the-art results on most NLP tasks (2014-2018)
- seq2seq+Attention was an even more revolutionary idea (Google Translate used it)
- 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