Foundations of Machine Learning (CS 725)

FALL 2024

Lecture 18:
- Recurrent Neural Networks
- Encoder/Decoder Model with Attention

Instructor: Preethi Jyothi

Why Recurrent Neural Networks (RNNs)?

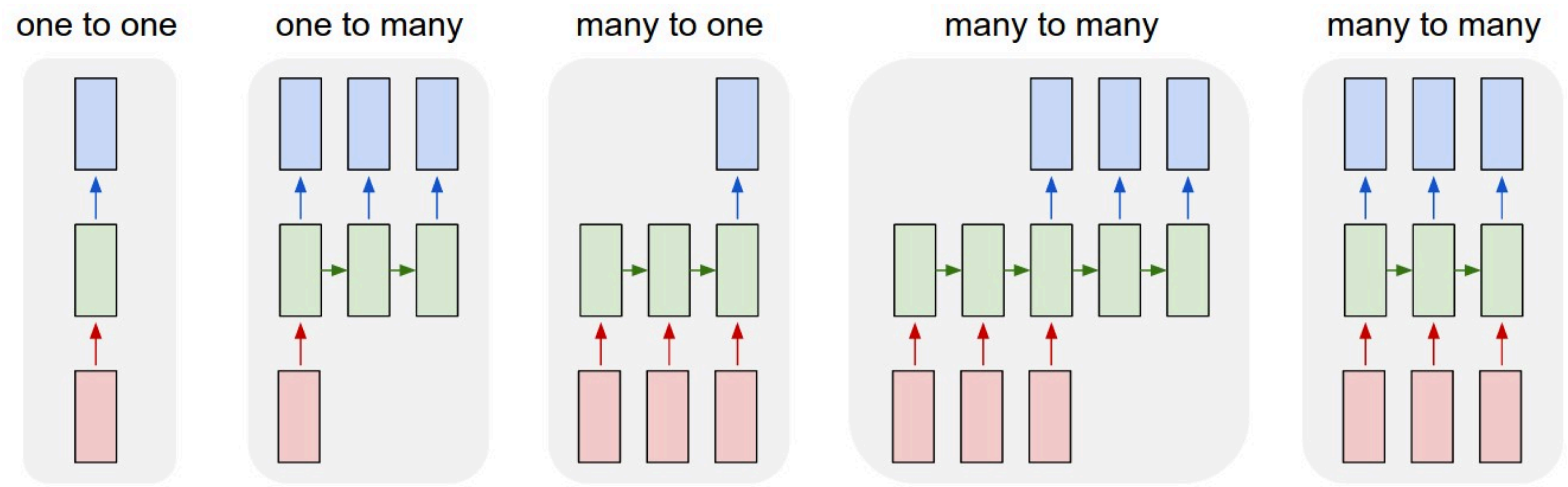
- Fixed-length inputs and outputs may not always be a reasonable choice. E.g. speech recognition, machine translation, sentiment classification, etc.

- Context information is crucial for many sequence prediction problems.

  - Hard to choose a fixed context window.

- RNNs are a family of neural networks that are better suited to handle sequential data

RNNs appear in different forms



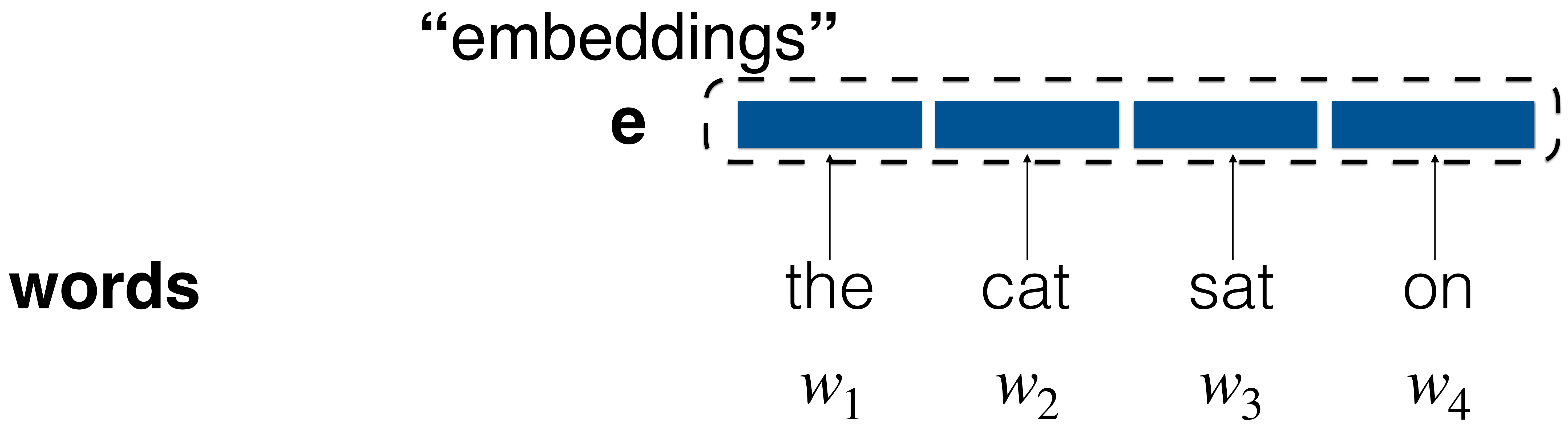
Example:
Language Modeling!

Image from:. http://karpathy.github.io/2015/05/21/rnn-effectiveness/

What is language modeling?

- Given a sequence of words/characters, $w_1, \ldots, w_{t-1}$, what is the most likely word/character at the next timestep w_t?
- $P(w_1, \ldots, w_T) = \prod_t P(w_t \mid w_{t-1}, \ldots, w_1)$
- Why is this an interesting problem?
  - Useful for a wide range of problems involving natural language. Examples include speech recognition, machine translation, spelling correction, summarization, etc.

Fixed-window NN language model



Word representations in Ngram models

- In standard Ngram models, words are represented in the discrete space involving the vocabulary
- Limits the possibility of truly interpolating probabilities of unseen Ngrams
- Can we build a representation for words in the continuous space?

Word representations

- 1-hot representation:
    - Each word is given an index in {1, … , V}. The 1-hot vector $f_i \in R^V$ contains zeros everywhere except for the i^{th} dimension being 1
- 1-hot form, however, doesn't encode information about word similarity
- Distributed (or continuous) representation: Each word is associated with a dense vector. Based on the "distributional hypothesis".
  E.g. dog ⟶ {-0.02, -0.37, 0.26, 0.25, -0.11, 0.34}

Word embeddings

- These distributed representations in a continuous space are also referred to as "word embeddings"
  - Low dimensional
  - Similar words will have similar vectors
- Word embeddings capture semantic properties (such as *man* is to *woman* as *boy* is to *girl*, etc.) and morphological properties (*glad* is similar to *gladly*, etc.)
- The word embeddings could be learned via the first layer of a neural network [B03].

[B03]: Bengio et al., "A neural probabilistic LM", JMLR, 03

Fixed-window NN language model

output probability:

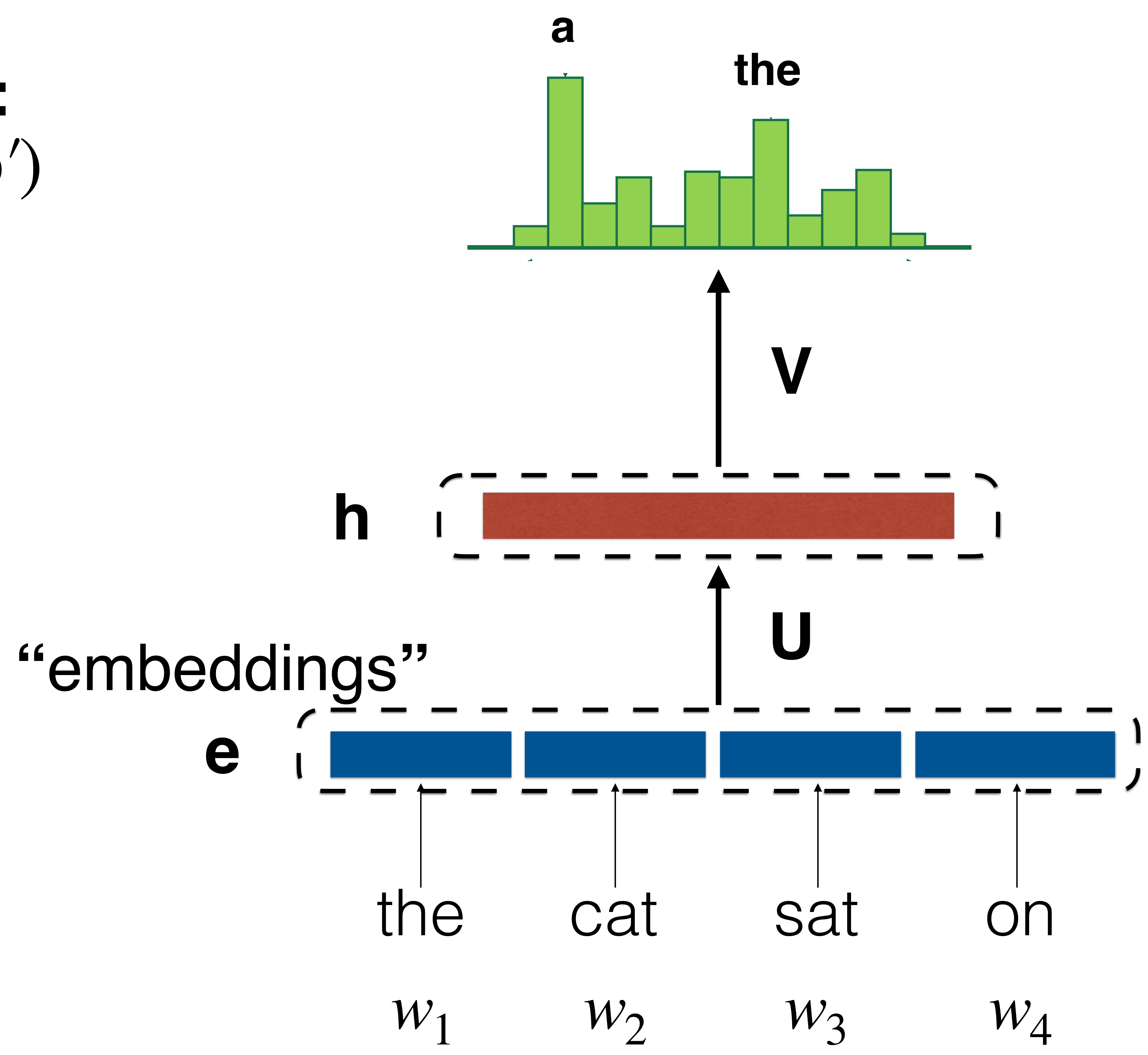
$$\hat{\mathbf{y}} = \mathrm{softmax}(\mathbf{Vh} + \mathbf{b}')$$

$$\mathrm{softmax}(\mathbf{z}) = \frac{\exp(\mathbf{z})}{\sum^{k} \exp(z_i)}$$

hidden layer:

$$\mathbf{h} = \tanh(\mathbf{Ue} + \mathbf{b})$$

words



Longer word context?

- What have we seen so far: A feedforward NN used to compute an Ngram probability $P(w_j = i|h_j)$ (where h_j encodes the Ngram history)

- We know Ngrams are limiting:
  Alice who had attempted *the assignment asked* the lecturer

- How can we predict the next word based on the entire sequence of preceding words? Use recurrent neural networks (RNNs)

Recurrent Neural Networks (RNNs)

A **hidden state** is associated with each time-step

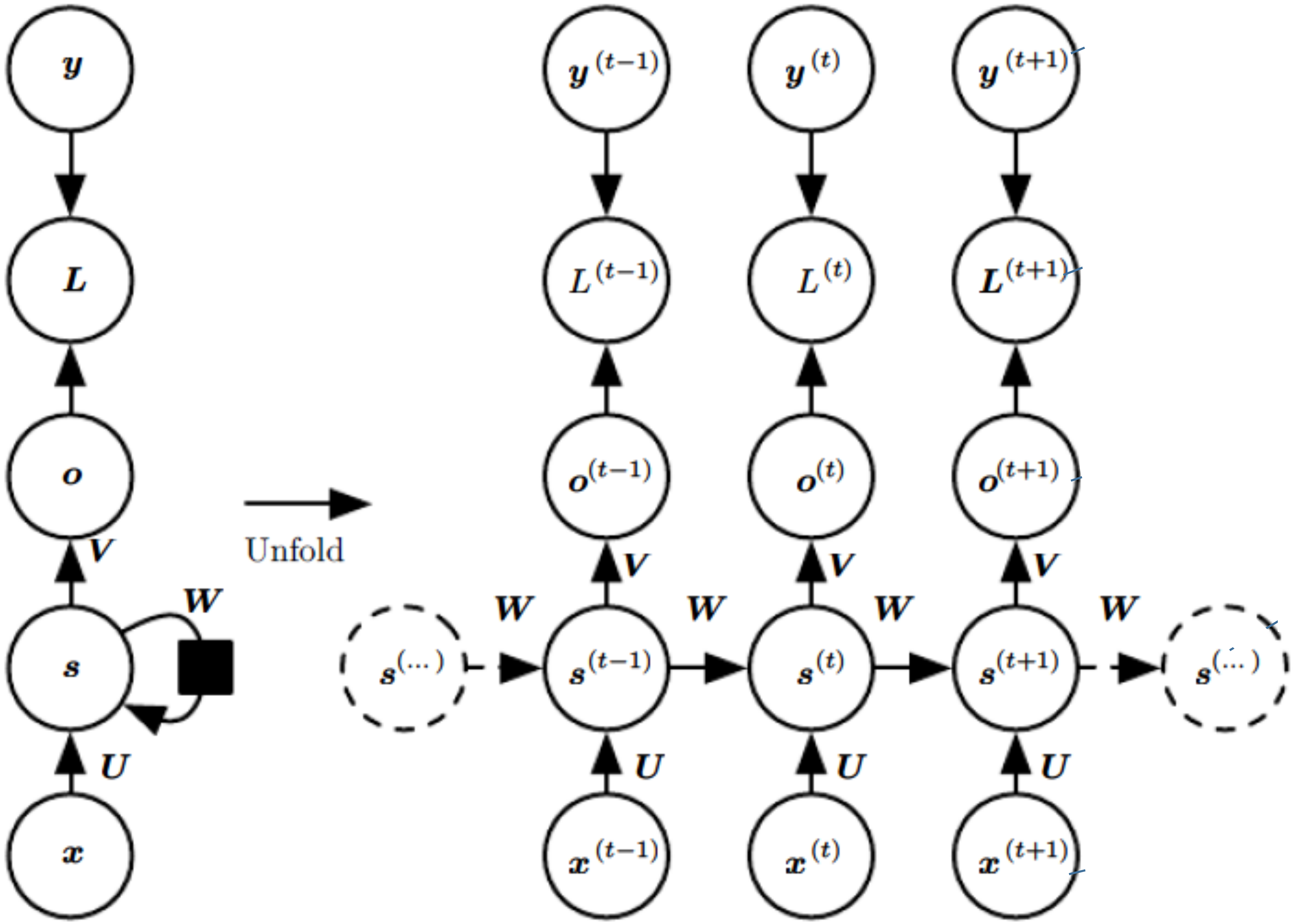
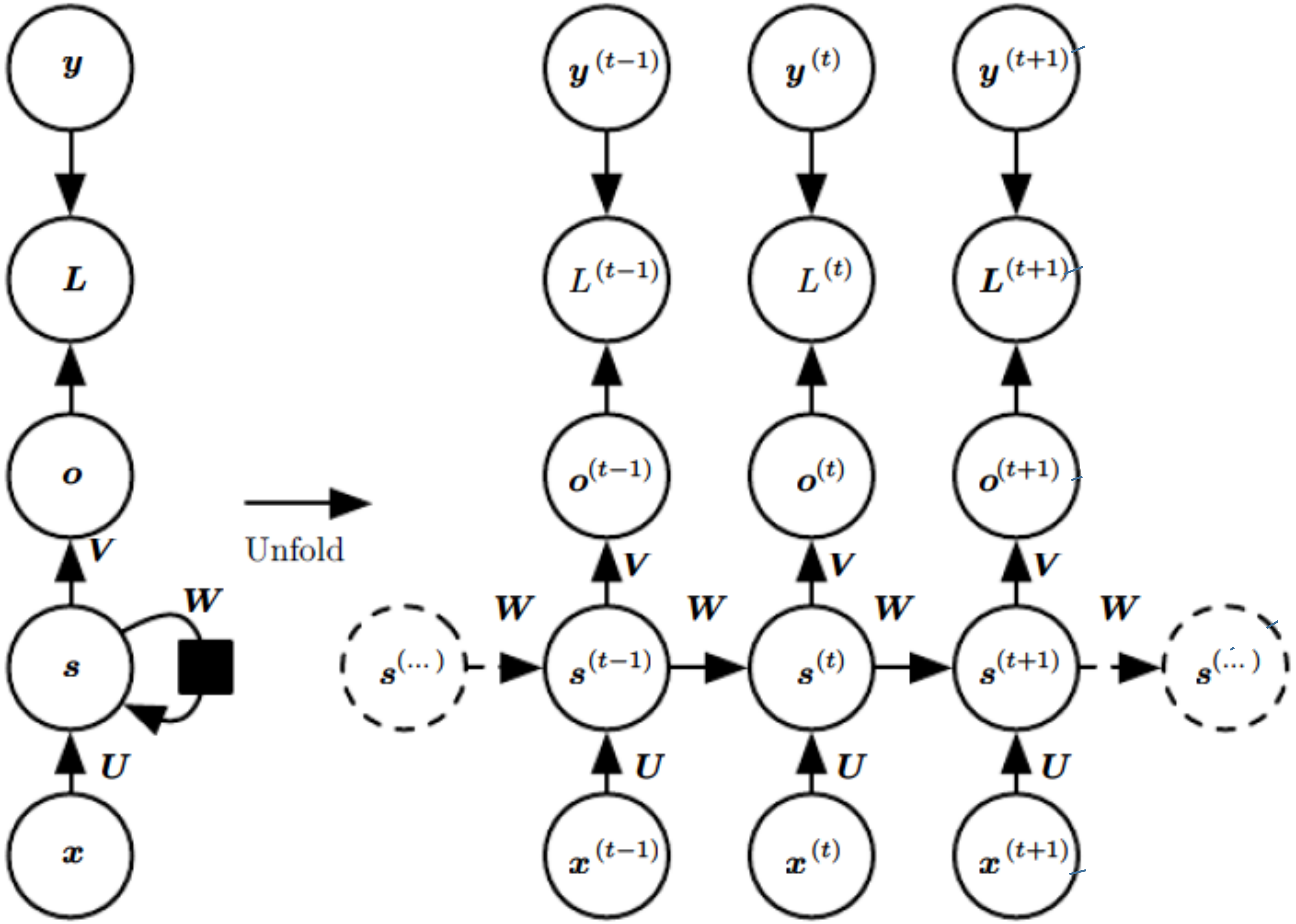


Image from: Deep Learning, https://www.deeplearningbook.org/.

Recurrent Neural Networks (RNNs)

At each time step: Use the input and the **previous hidden state** to compute the output



$$\mathbf{s}_t = \tanh(\mathbf{W}\mathbf{s}_{t-1} + \mathbf{U}\mathbf{x}_t + \mathbf{b})$$
$$\mathbf{o}_t = \mathbf{V}\mathbf{s}_t + \mathbf{b}'$$
$$\hat{\mathbf{y}}_t = \mathrm{softmax}(\mathbf{o}_t)$$

Image from: Deep Learning, https://www.deeplearningbook.org/.

RNN language model

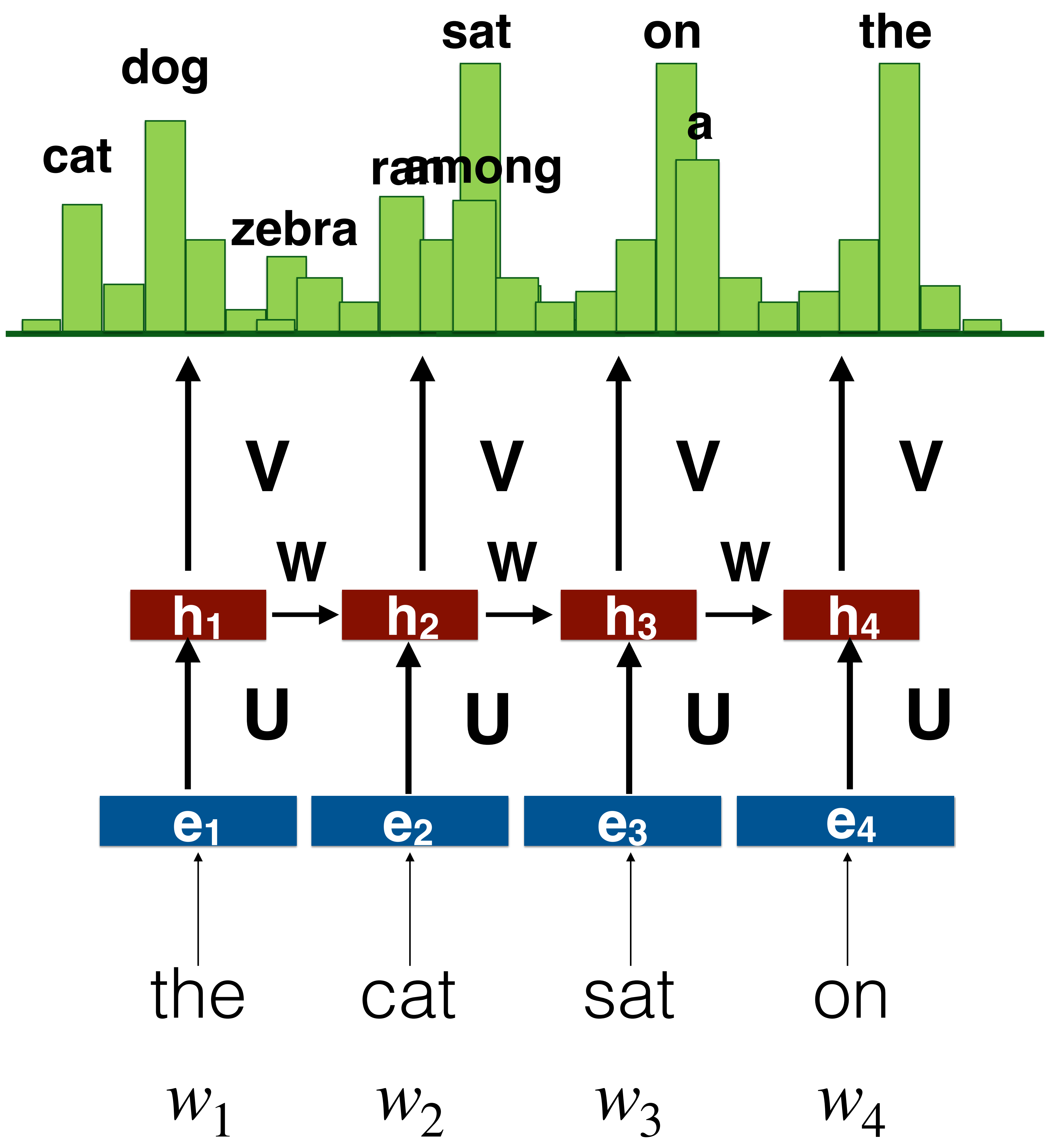
output probability:

$$\hat{\mathbf{y}}_t = \mathrm{softmax}(\mathbf{V}\mathbf{h}_t + \mathbf{b}')$$

hidden layer:

$$\mathbf{h}_t = \tanh(\mathbf{U}\mathbf{e}_t + \mathbf{W}\mathbf{h}_{t-1} + \mathbf{b})$$

words



Character-based RNNLMs

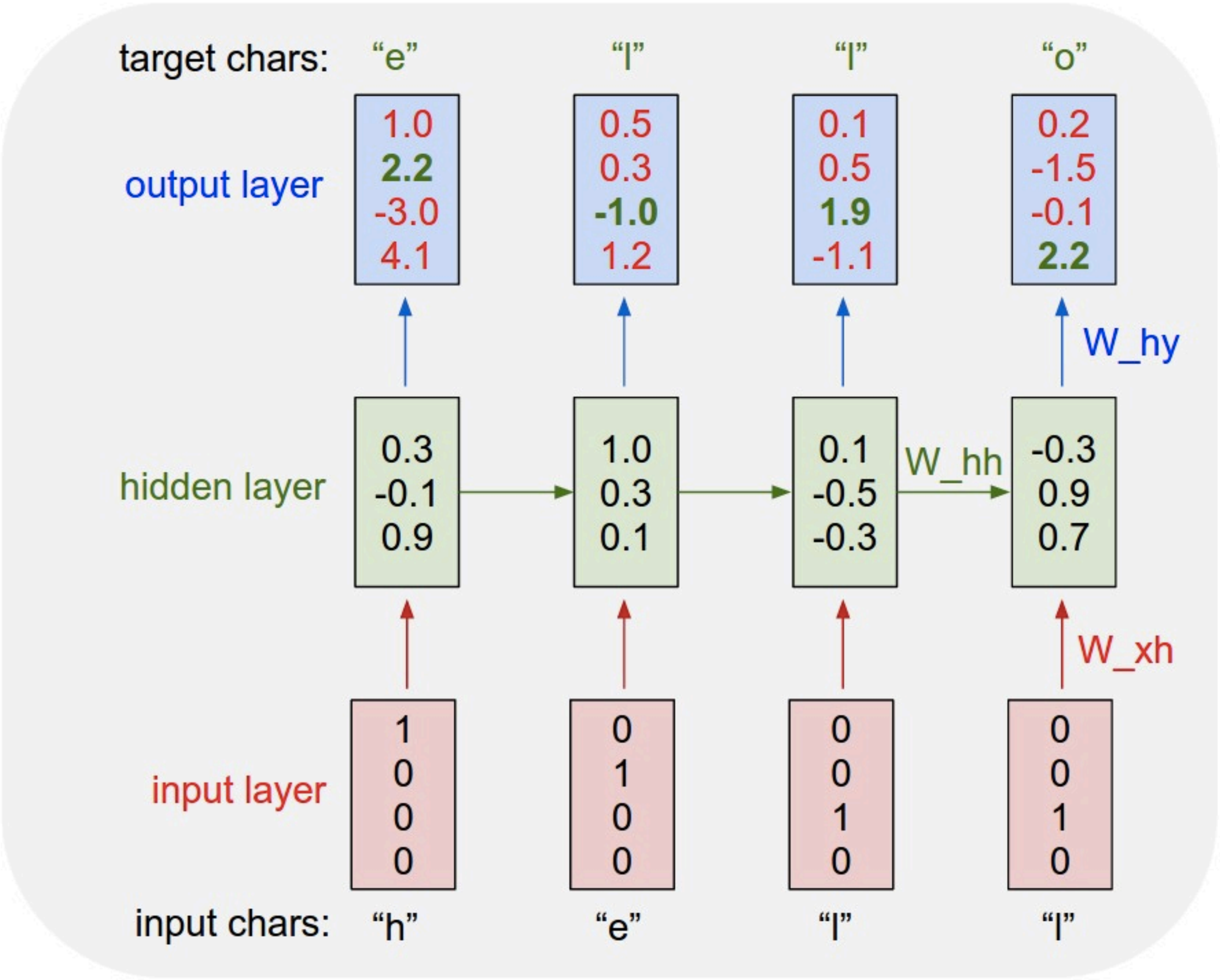
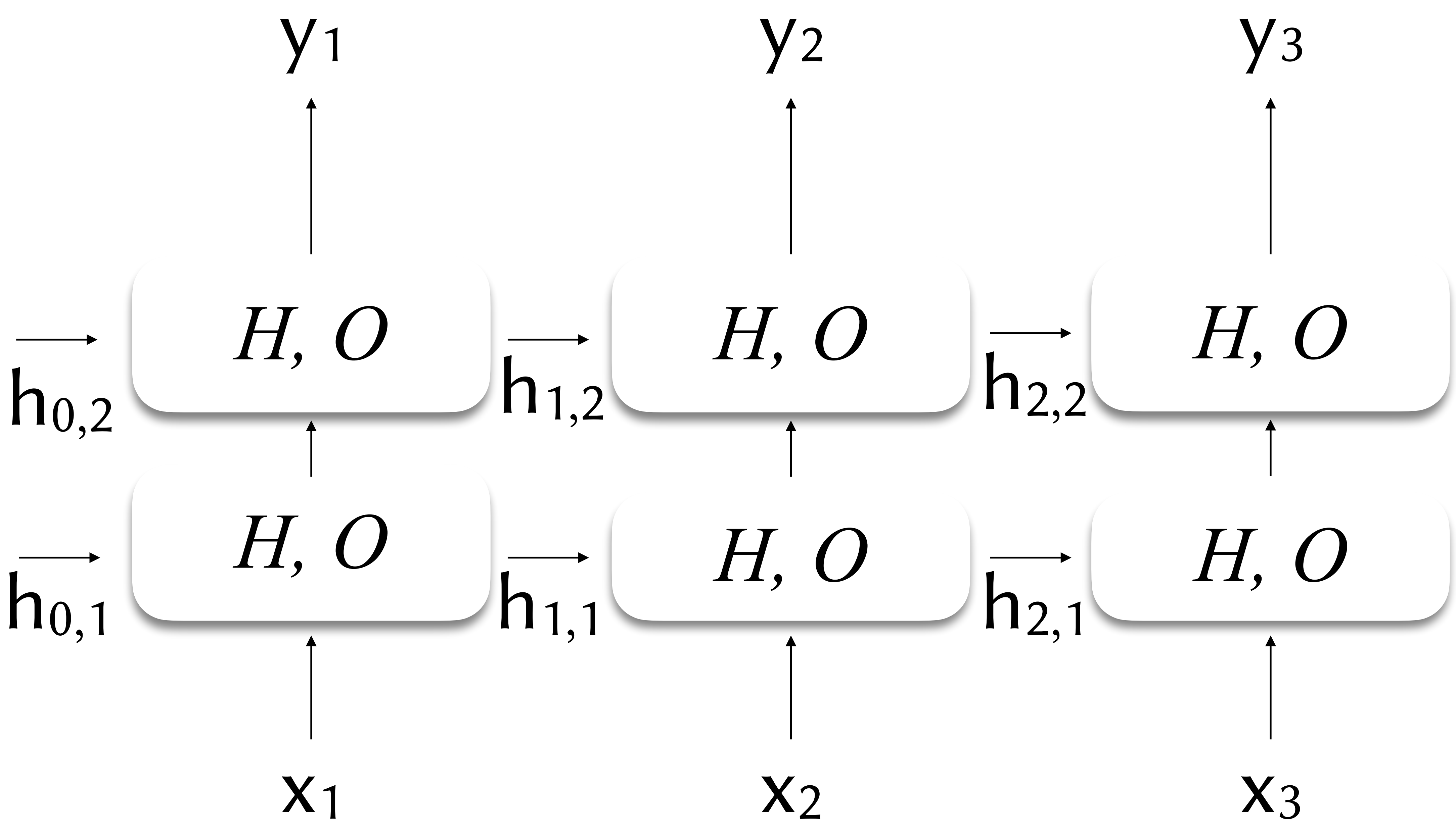


Image from: http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Generate text using a trained character-based LM

VIOLA:
Why, Salisbury must find his flesh and thought
That which I am not aps, not a man and in fire,
To show the reining of the raven and the wars
To grace my hand reproach within, and not a fair are hand,
That Caesar and my goodly father's world;
When I was heaven of presence and our fleets,
We spare with hours, but cut thy council I am great,
Murdered and by thy master's ready there
My power to give thee but so much as hell:
Some service in the noble bondman here,
Would show him to her wine.

Deep RNNs



- RNNs can be stacked in layers to form deep RNNs

Vanilla RNN Model

$$h_t = H(Vx_t + Wh_{t-1} + b^{(h)})$$

$$y_t = O(Uh_t + b^{(y)})$$

H : element wise application of the sigmoid or tanh function

O : the softmax function

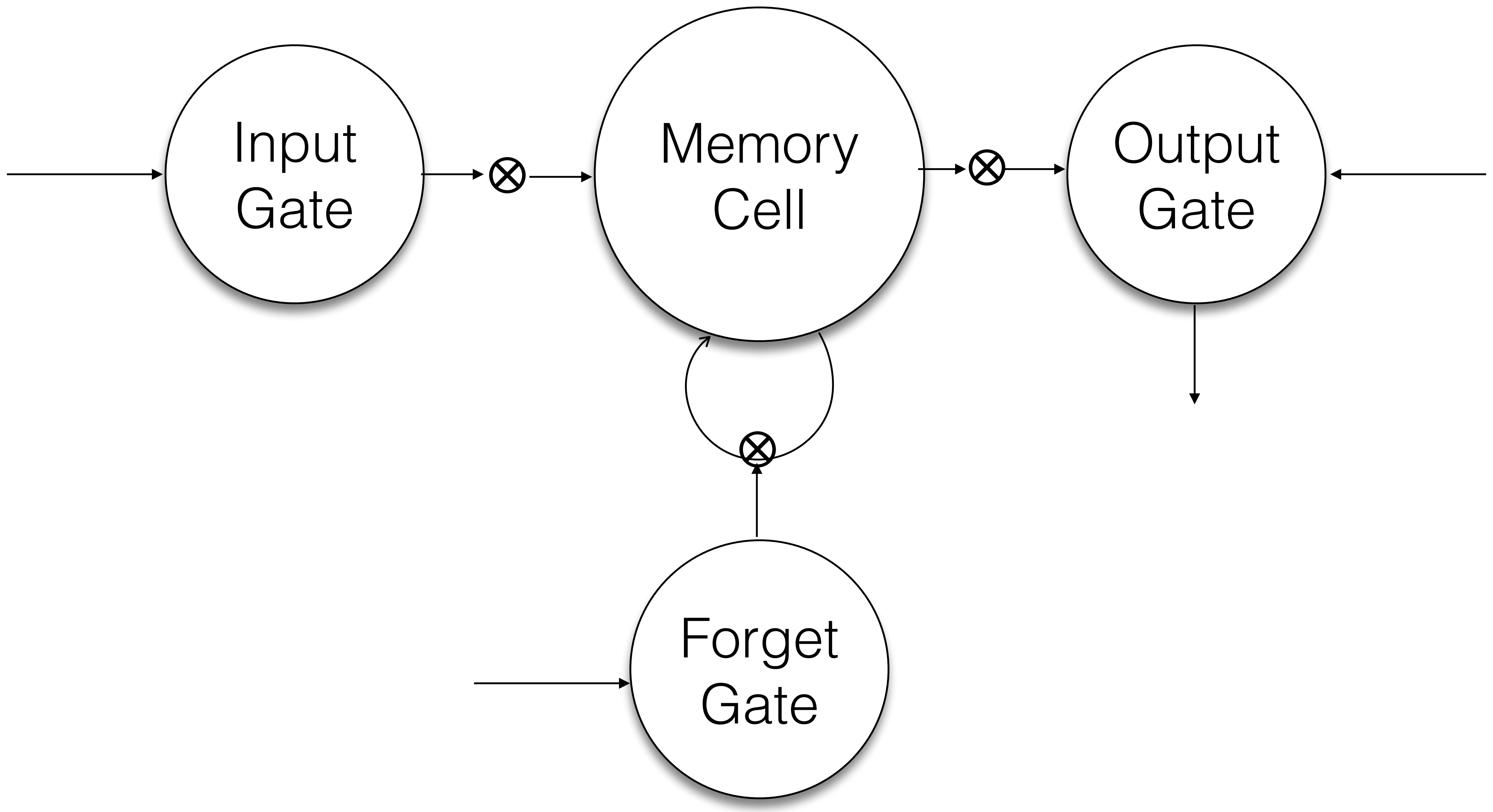
Run into problems of exploding and vanishing gradients.

Exploding/Vanishing Gradients

- In deep networks, gradients in early layers are computed as the product of terms from all the later layers
- This leads to unstable gradients:
  - If the terms in later layers are large enough, gradients in early layers (which is the product of these terms) can grow exponentially large: *Exploding gradients*
  - If the terms in later layers are small, gradients in early layers will tend to exponentially decrease: *Vanishing gradients*
- To address this problem in RNNs, Long Short Term Memory (LSTM) units were proposed [HS97]

[HS97] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," Neural Computation, 1997.

Long Short Term Memory (LSTM) Cells

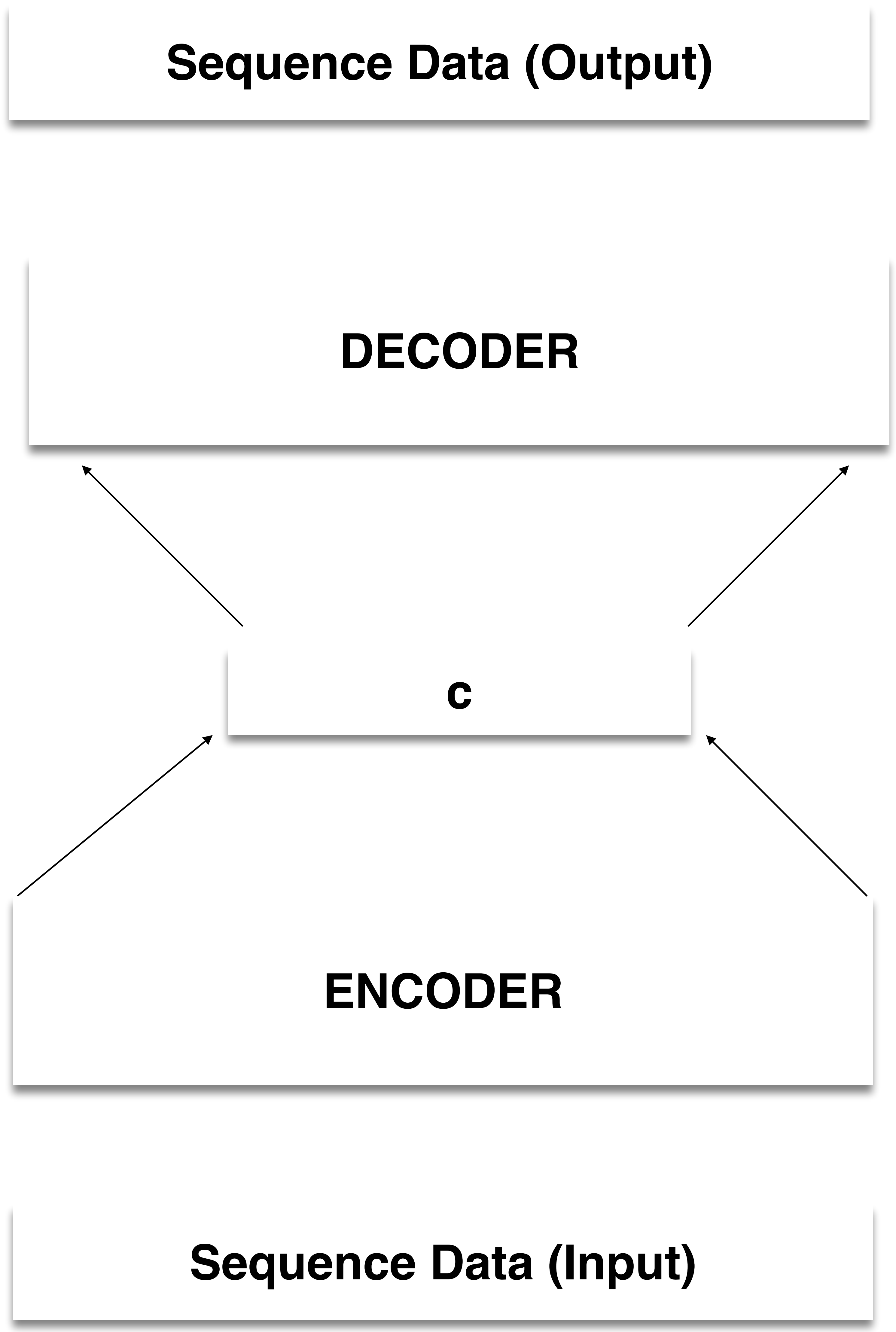


- Memory cell: Neuron that stores information over long time periods
- Forget gate: When on, memory cell retains previous contents. Otherwise, memory cell forgets contents.
- When input gate is on, write into memory cell
- When output gate is on, read from the memory cell

https://colah.github.io/posts/2015-08-Understanding-LSTMs/

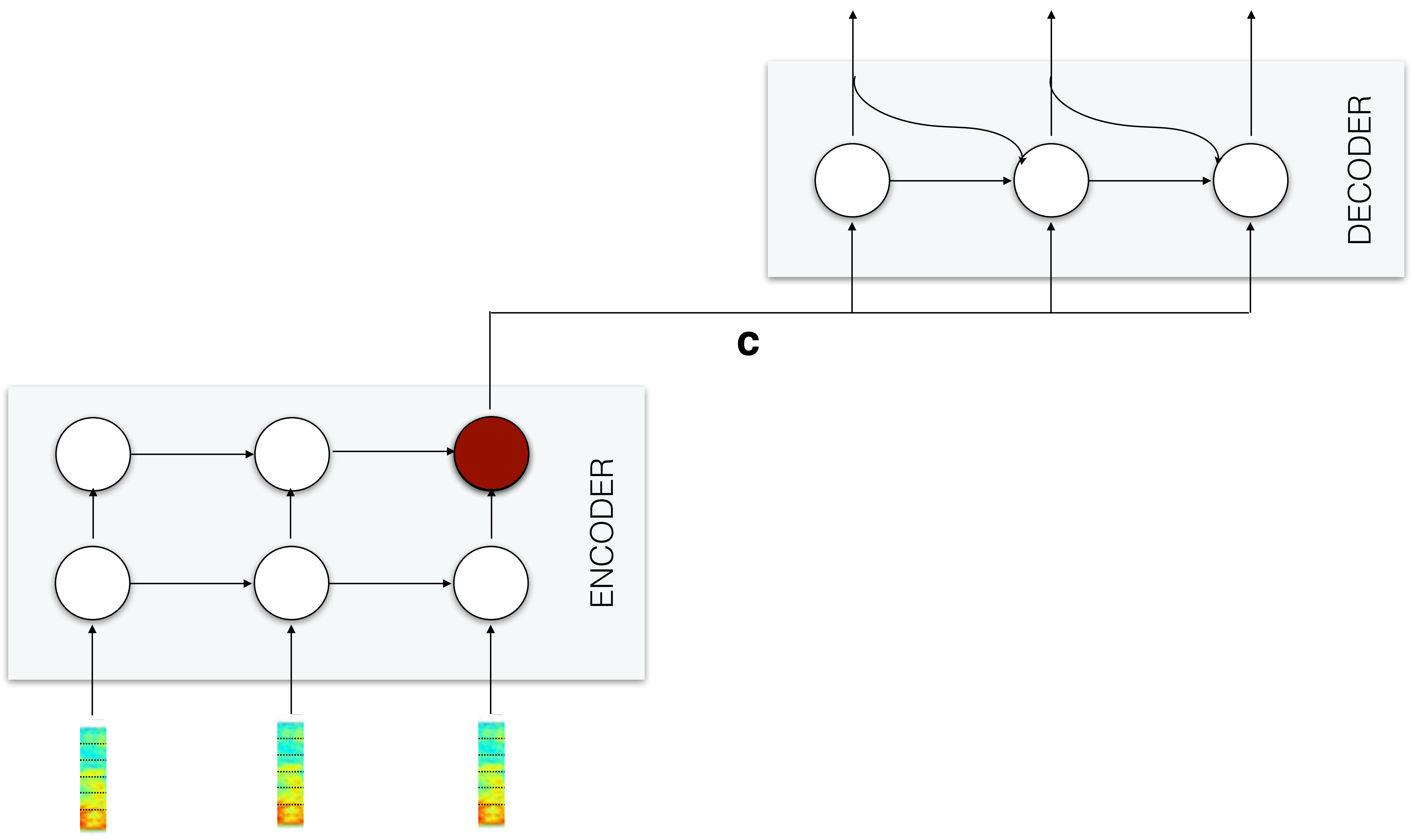
Sequence to sequence models

Encoder-decoder architecture

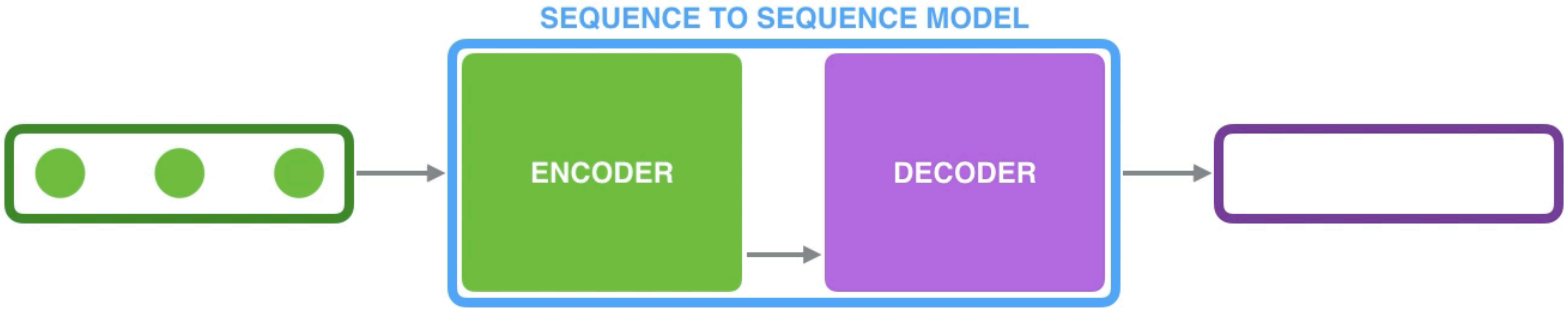


Sequence to sequence models

Encoder-decoder architecture



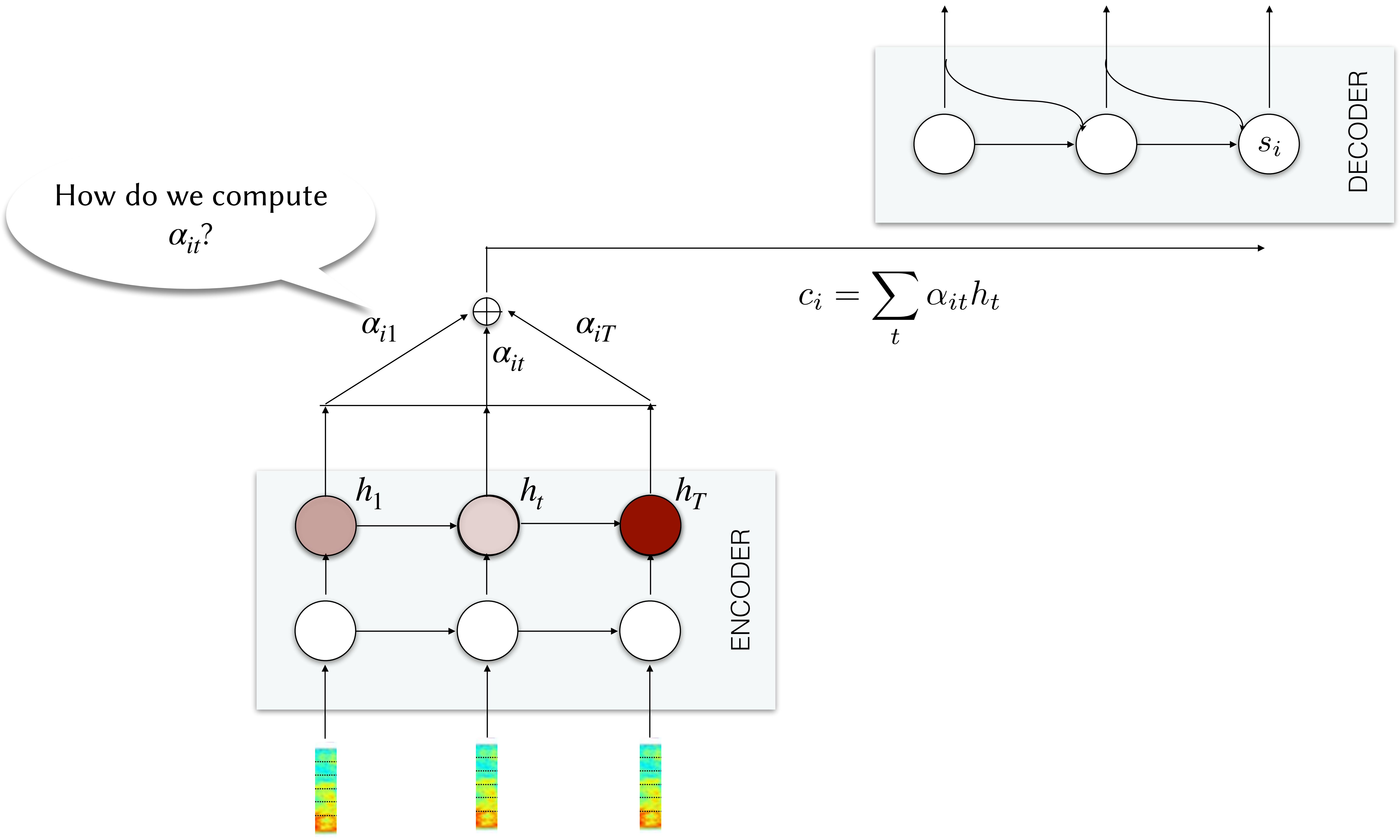
Sequence to sequence model



Jay Alammar, Visualizing A Neural Machine Translation Model (Mechanics of Seq2seq Models With Attention), 2018. http://jalammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention/

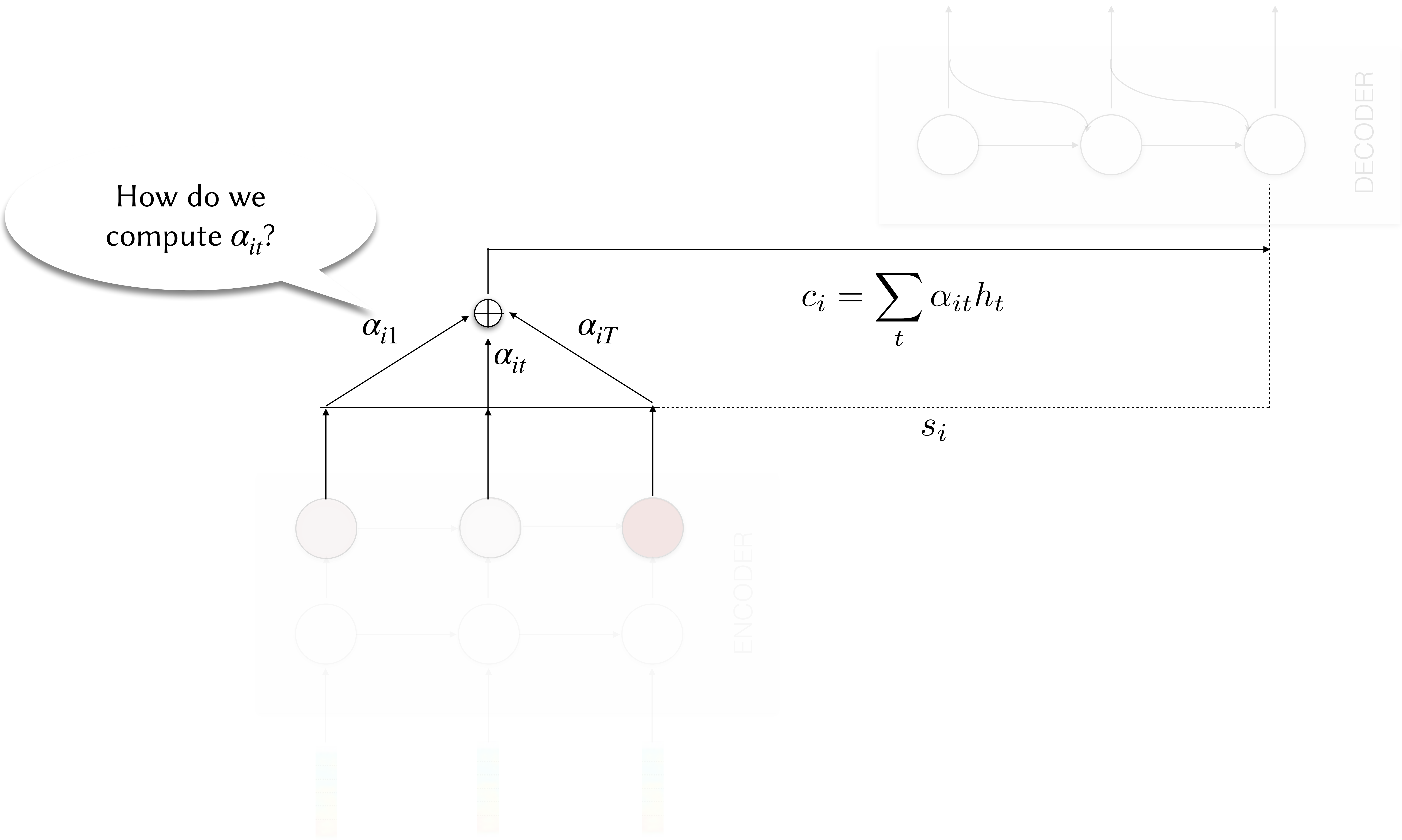
Sequence to sequence models

Encoder-decoder architecture with attention



Sequence to sequence models

Encoder-decoder architecture with attention

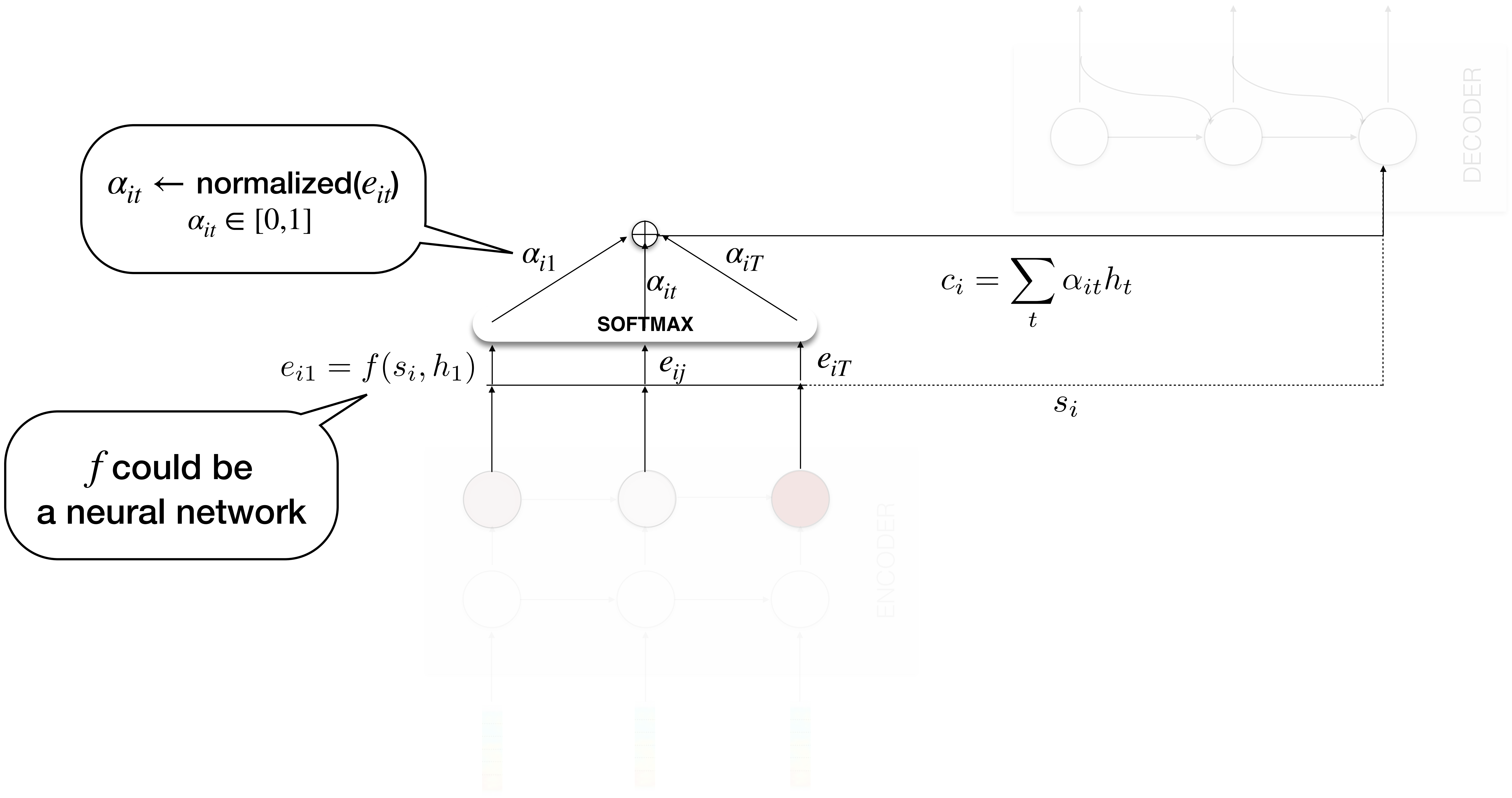


Sequence to sequence model with Attention

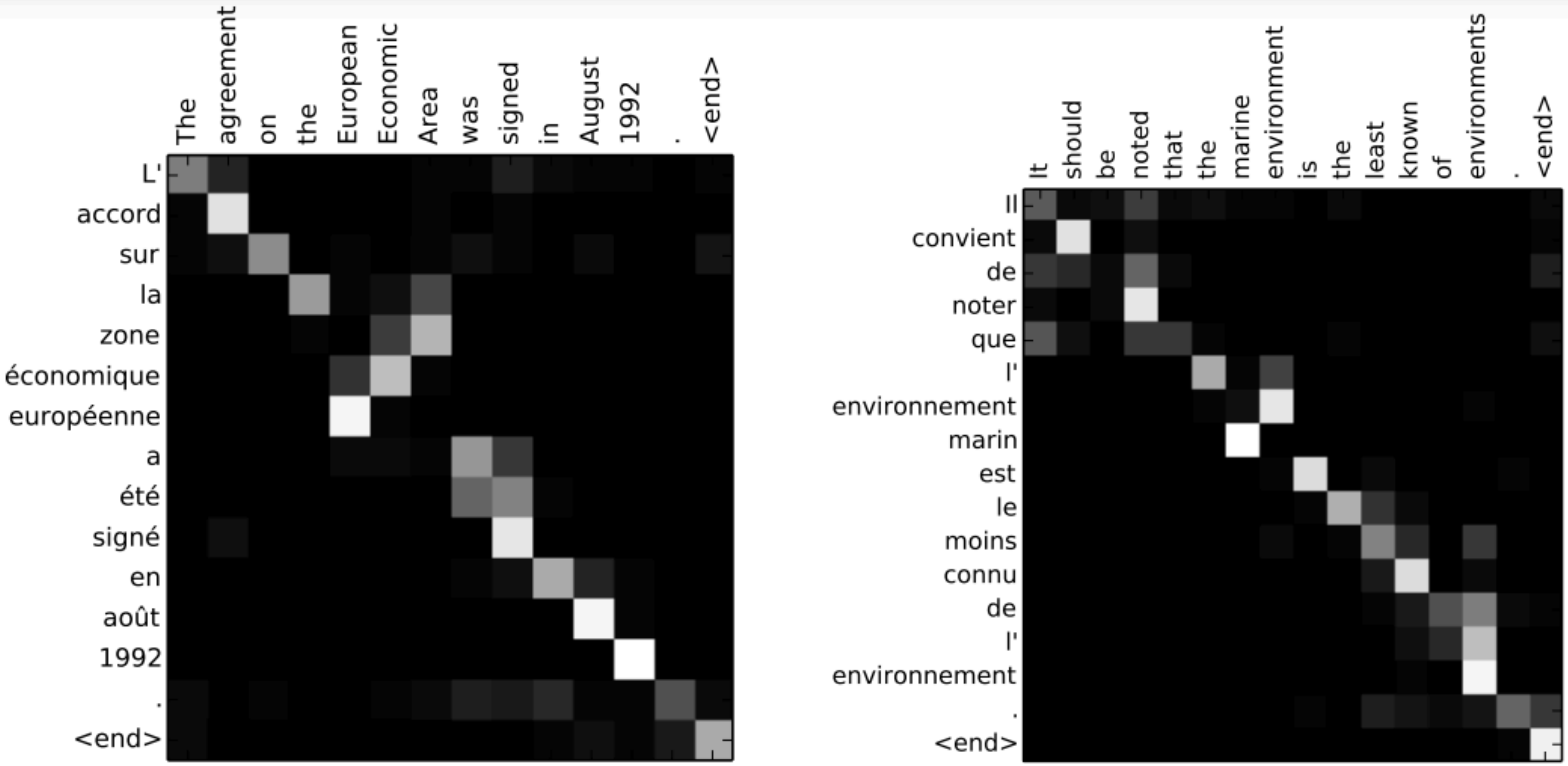
Jay Alammar, Visualizing A Neural Machine Translation Model (Mechanics of Seq2seq Models With Attention), 2018.
http://jalammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention/

Sequence to sequence models

Encoder-decoder architecture with attention



Attention Learns Alignment



(a)

(b)

Image reproduced from: Neural Machine Translation by Jointly Learning to Align and Translate, Dzimitry Bahdanau, KyungHyun Cho, Yoshua Bengio, ICLR 2015