732A78 Deep Learning (2020)

Recurrent neural networks

Marco Kuhlmann
Department of Computer and Information Science

LINKÖPING UNIVERSITY

Prelude: Word embeddings

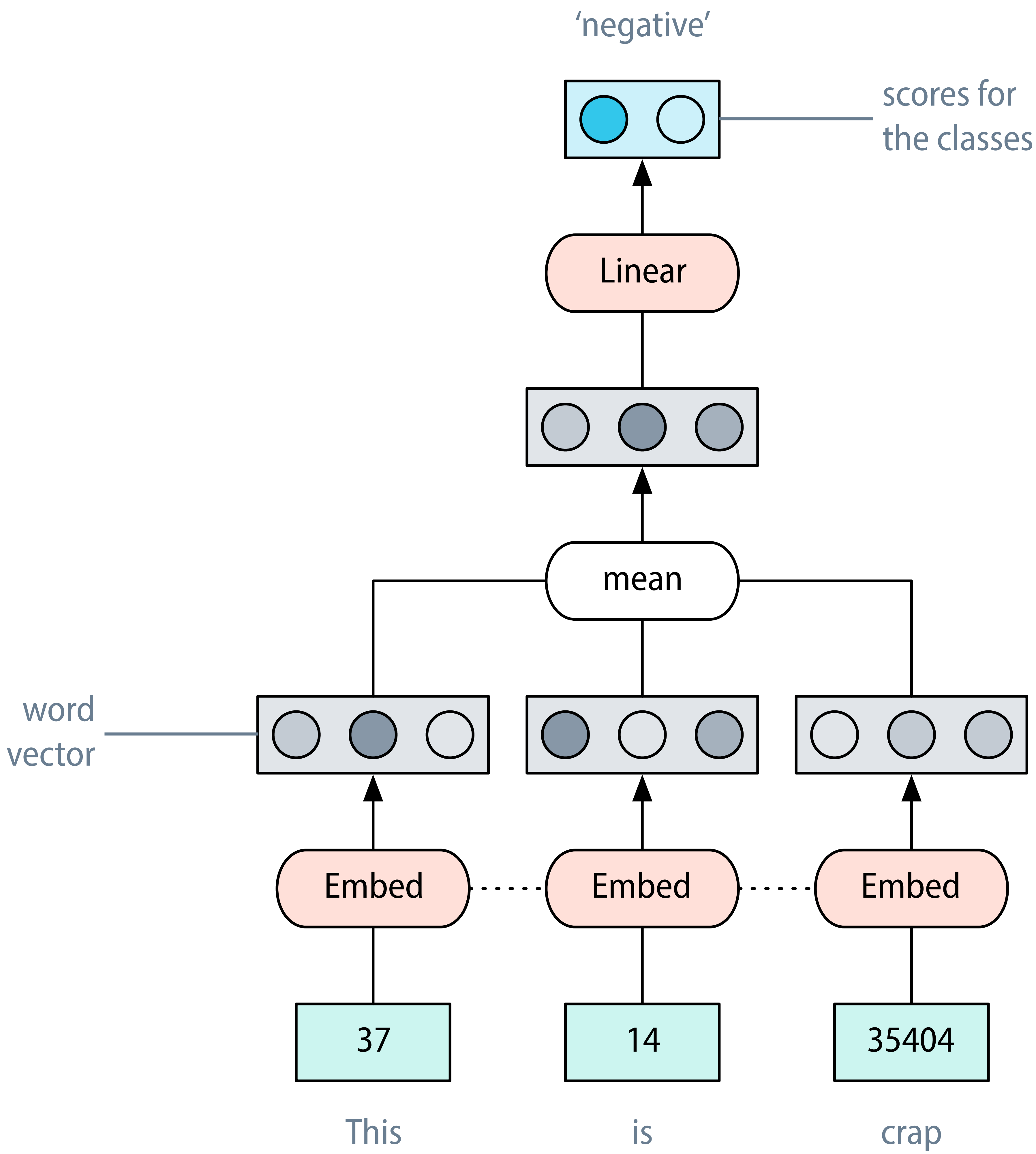
Sentiment analysis

The gorgeously elaborate continuation of "The Lord of the Rings" trilogy is so huge that a column of words cannot adequately describe co-writer/director Peter Jackson's expanded vision of J.R.R. Tolkien's Middle-earth.

positive

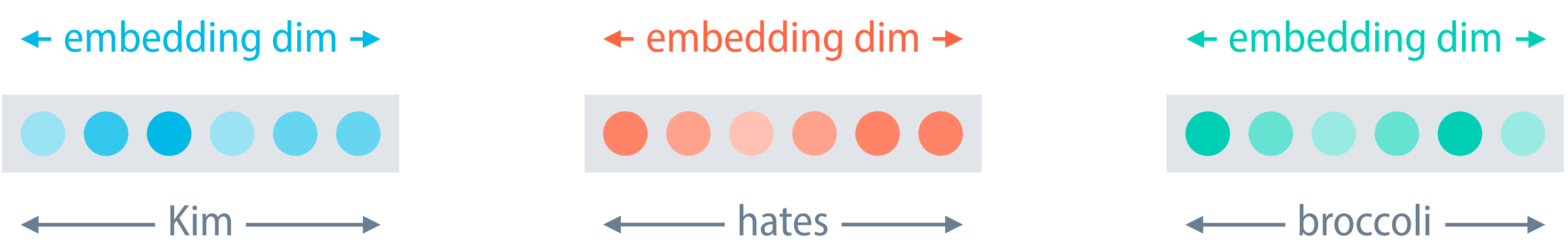
… is a sour little movie at its core; an exploration of the emptiness that underlay the relentless gaiety of the 1920's, as if to stop would hasten the economic and global political turmoil that was to come.

negative


'negative'
scores for
the classes
Linear
mean
word
vector
Embed
Embed
Embed
37
14
35404
This
is
crap

Word embeddings

- A **word embedding** is a mapping from a finite set of words V to a d-dimensional vector space, where $d \ll |V|$.

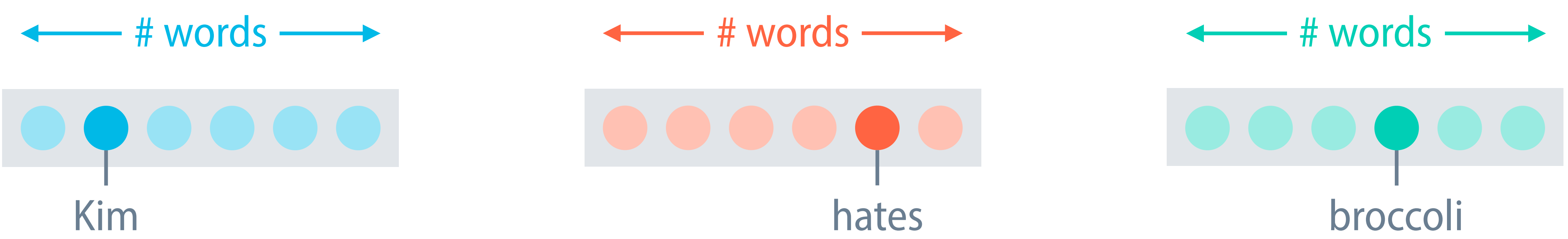


- Embedding vectors can be initialised with random numbers and trained alongside the weights of a network.

- Training tunes the embedding vectors to the task at hand.

Embedding layers are linear transformations

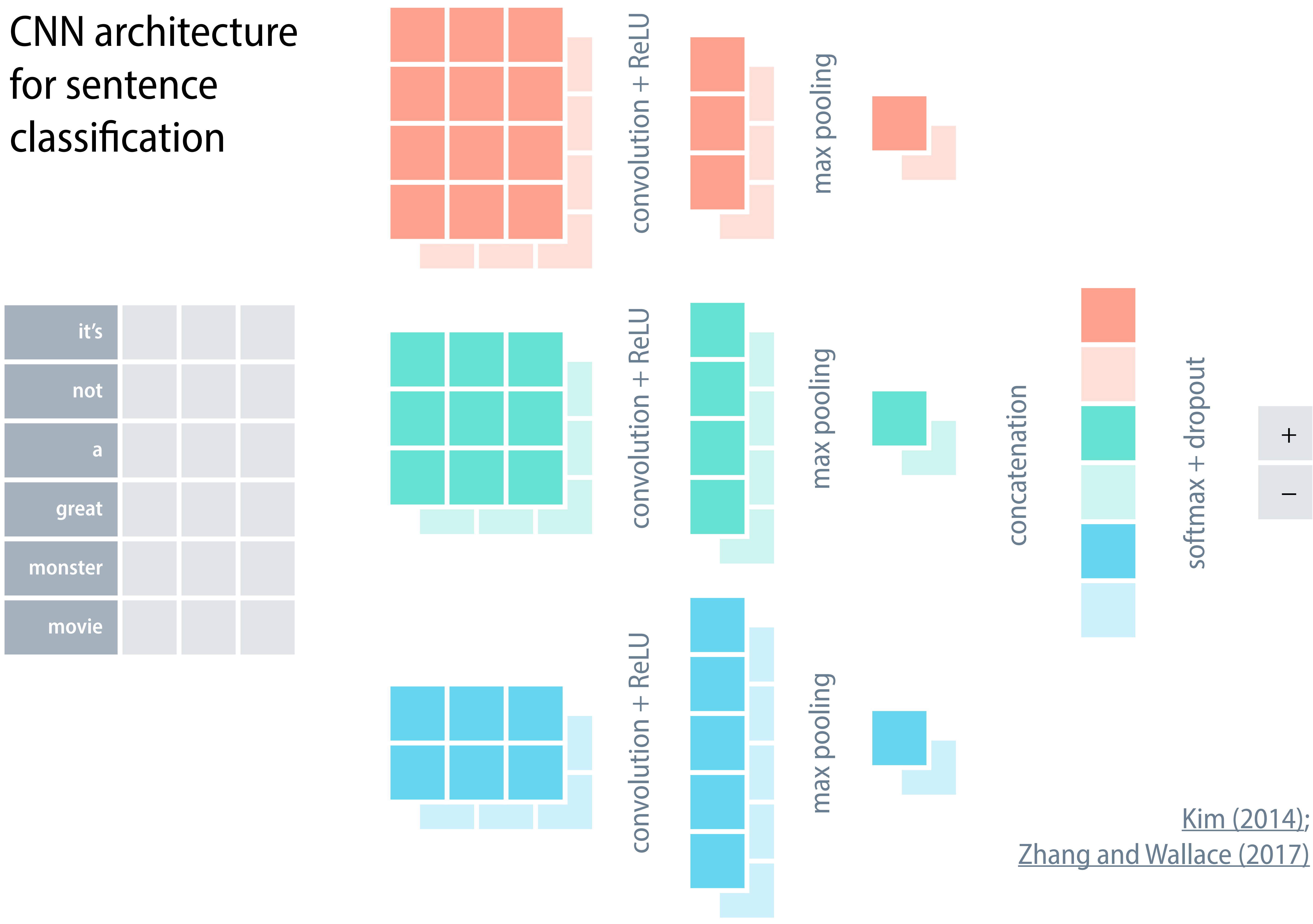
- A **one-hot vector** is a vector where one component has value 1, and all other components have value 0.



- A word embedding can be viewed as a linear transformation from one-hot vectors into the d-dimensional embedding space.

values of the embedding vector = weights for the non-zero component

CNN architecture for sentence classification



Kim (2014); Zhang and Wallace (2017)

This lecture

- Prelude: Word embeddings
- Introduction to recurrent neural networks
- The LSTM architecture
- Use case 1: Part-of-speech tagging
- Use case 2: Contextualised word embeddings

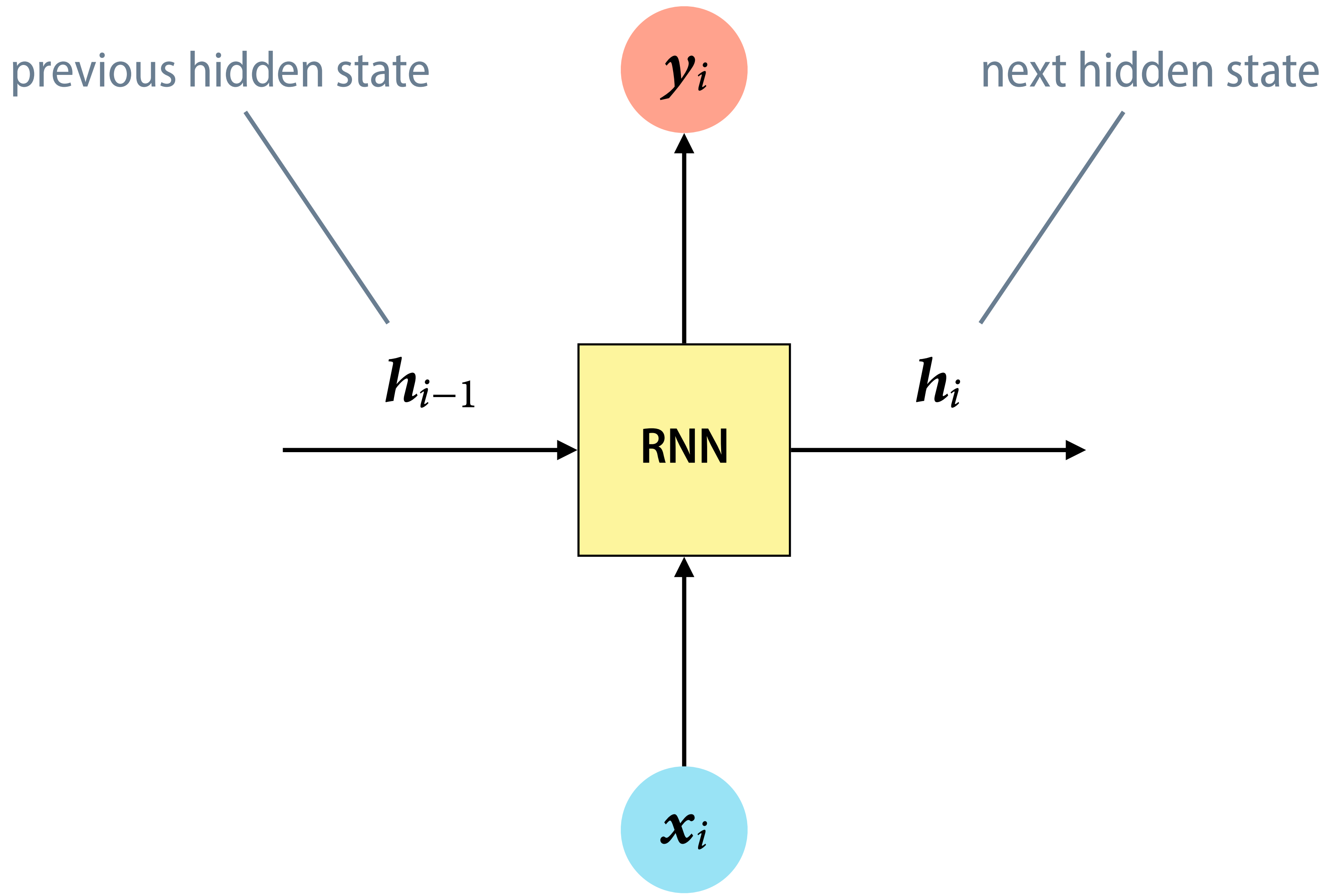
Introduction to recurrent neural networks

Recurrent neural networks

- **Recurrent neural networks (RNNs)** can process variable length sequences of inputs, such as sequences of letters or words.
- For any input sequence, a recurrent neural network is 'unrolled' into a deep feedforward network.

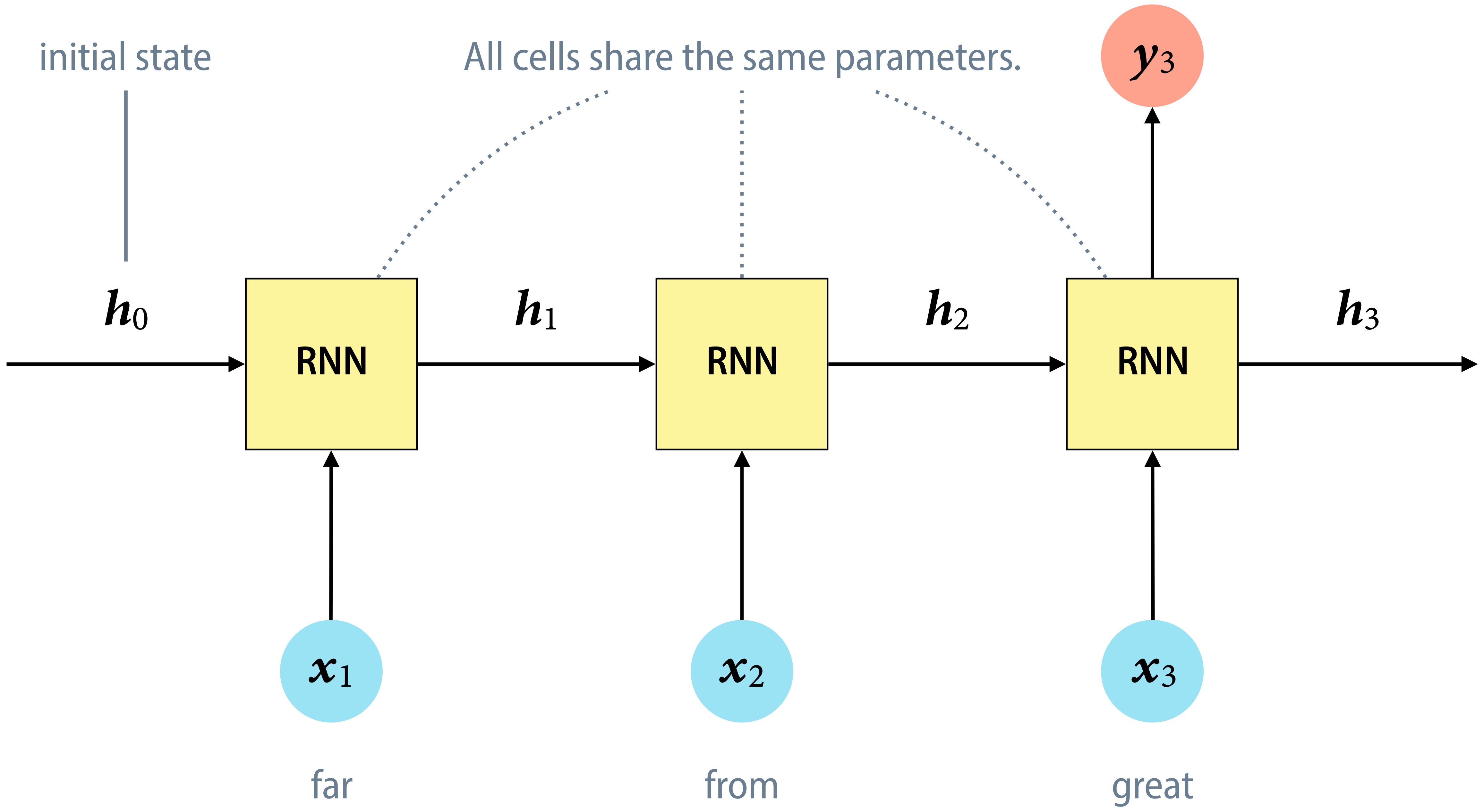
  Depth is proportional to the length of the sequence.
- In contrast to the situation with deep feedforward networks, all parameters are shared across all positions of the sequence.

RNN, recursive view



$$\boldsymbol{h}_i = H(\boldsymbol{h}_{i-1}, \boldsymbol{x}_i) \qquad \boldsymbol{y}_i = O(\boldsymbol{h}_{i-1}, \boldsymbol{x}_i)$$

RNN, unrolled view



Properties of recurrent neural networks

- The parameters of the model are shared across all positions.

  The number of parameters does not grow with the sequence length.

- The output can be influenced by the entire input seen so far.

  Contrast this with the locality constraint of CNNs.

- The hidden state can be a 'lossy summary' of the input sequence.

  Hopefully, it will encode useful information for the task at hand.

Training recurrent neural networks

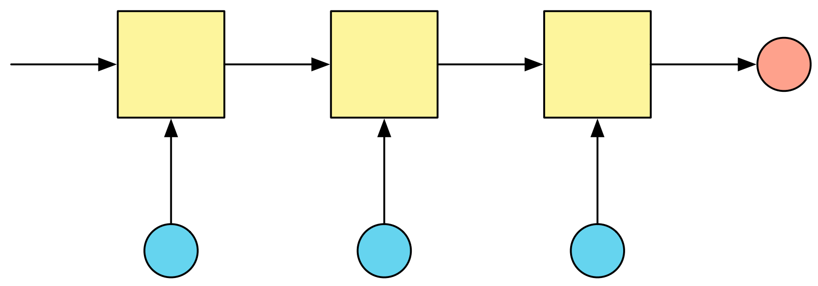
- Unrolled recurrent neural networks are just feedforward networks, and can therefore be trained using backpropagation.

  No specialised algorithm necessary!

- This way of training recurrent neural networks is called **backpropagation through time**.

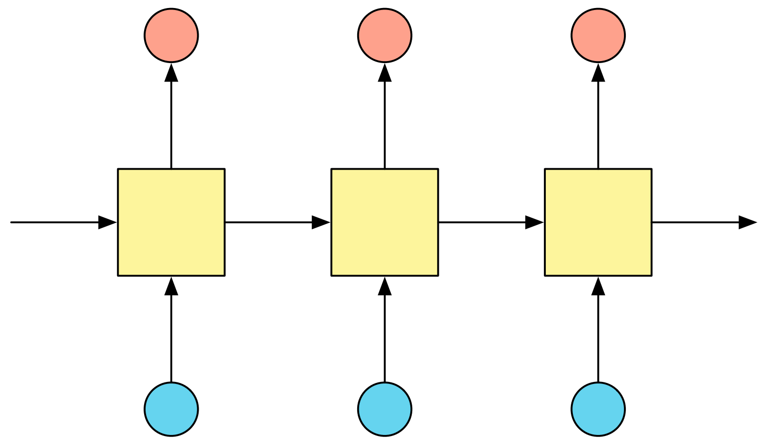
- Shared weights are updated by summing over the gradients computed for each position.

Common usage patterns for RNNs



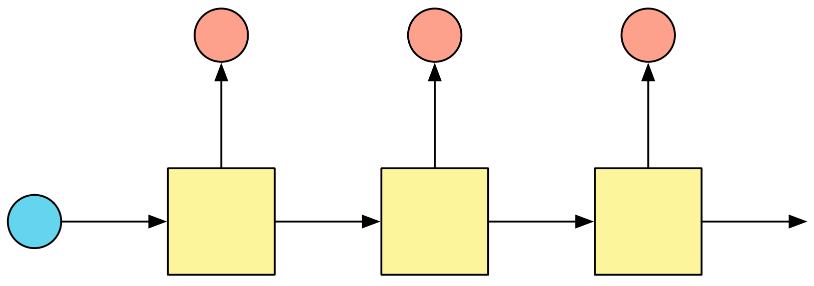
encoder

example: text classification



transducer

example: part-of-speech tagging



decoder

example: text generation

Extensions of the basic RNN architecture

- **Stacked RNNs** are RNNs with several layers, where the outputs of one layer become the inputs of the next.
- **Bidirectional RNNs** combine one RNN that moves forward through the input with another RNN that moves backward.

  outputs at each position are concatenated

This lecture

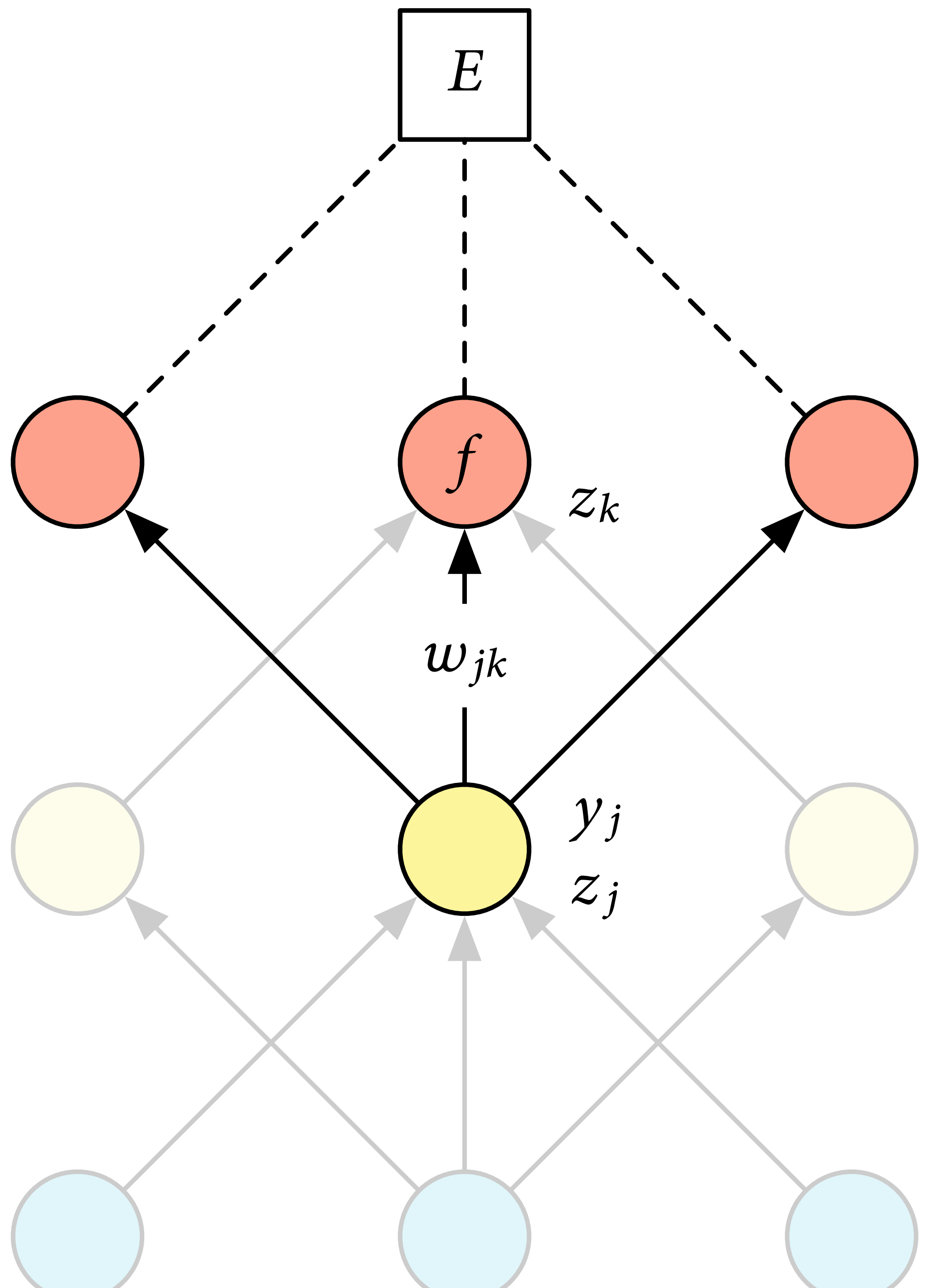
- Prelude: Word embeddings
- Introduction to recurrent neural networks
- The LSTM architecture
- Use case 1: Part-of-speech tagging
- Use case 2: Contextualised word embeddings

The LSTM architecture

Challenges with recurrent neural networks

- In principle, recurrent neural networks are capable of learning long-distance dependencies in input sequences.
- In practice, training recurrent neural networks is challenging due to the large depth of the unrolled networks.

Vanishing and exploding gradients



$$\delta_k = \frac{\partial E}{\partial z_k} = \frac{\partial E}{\partial y_k}\frac{\partial y_k}{\partial z_k} = \frac{\partial E}{\partial y_k} f'(z_k)$$

$$\delta_j = \frac{\partial E}{\partial z_j} = \frac{\partial y_j}{\partial z_j}\sum_k \frac{\partial E}{\partial z_k}\frac{\partial z_k}{\partial y_j} = f'(z_j)\sum_k \delta_k w_{jk}$$

Vanishing and exploding gradients

- In backpropagation there is a risk of gradients either vanishing or exploding, depending on the magnitude of the weights.
- This problem is exacerbated in recurrent networks, whose unrolled computation graphs can be very deep.
- Research on recurrent networks has proposed various methods to mitigate this problem.

  weight scaling and clipping, specialised architectures

Long Short-Term Memory (LSTM)

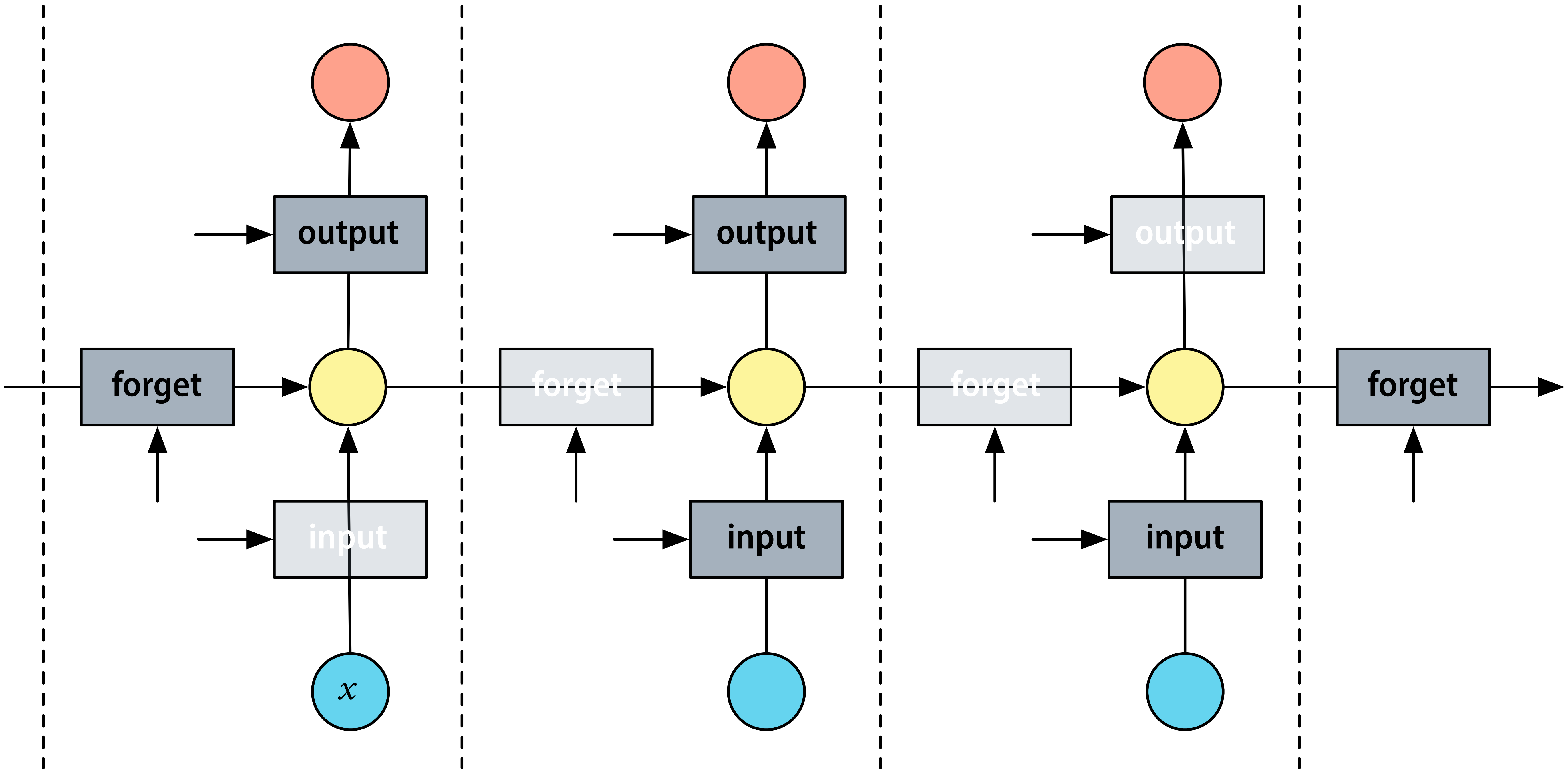
- The **Long Short-Term Memory (LSTM)** architecture was specifically designed to adress the vanishing gradients problem.
- Metaphor: The hidden state of the neural network can be considered as a short-term memory.
- The LSTM architecture tries to make this short-term memory last as long as possible by preventing vanishing gradients.

Memory cell and gating mechanism

The crucial innovation in an LSTM is the design of its memory cell.

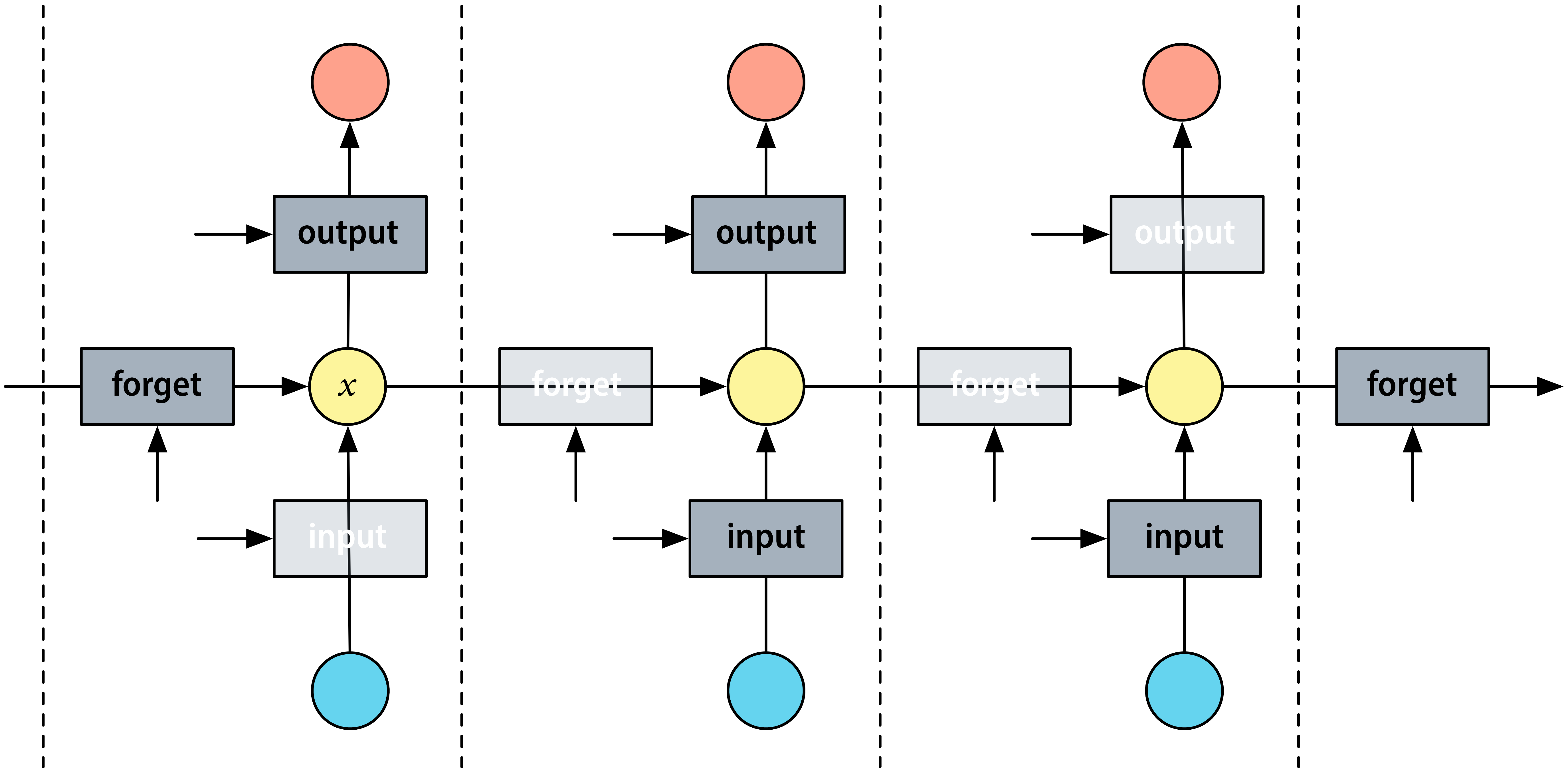
- Information is written into the cell if its INPUT gate is open.
- Information stays in the cell as long as its FORGET gate is closed.
- Information is read from the cell if its READ gate is open.

Information flow in an LSTM



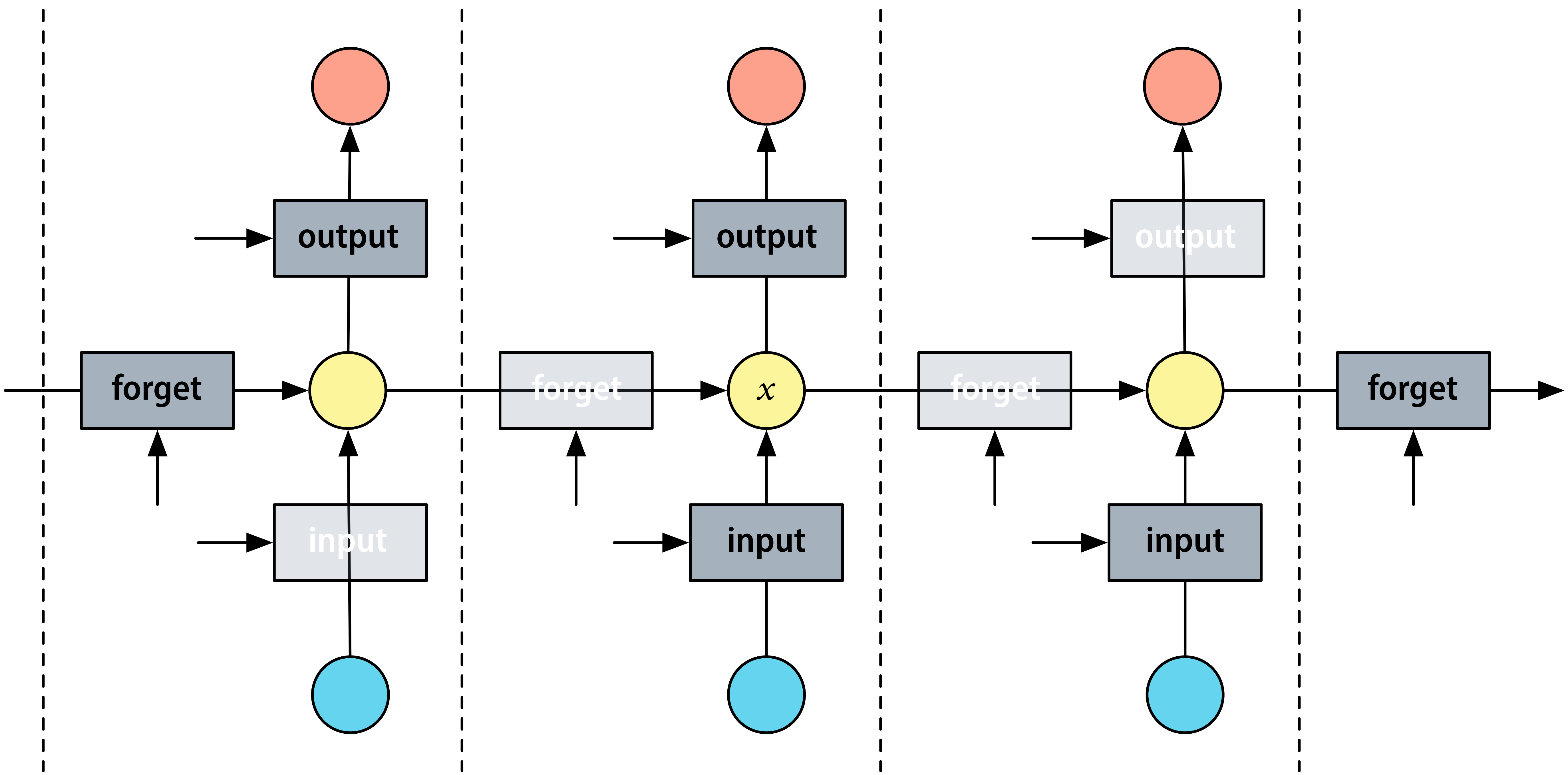
Attribution: Geoffrey Hinton

Information flow in an LSTM



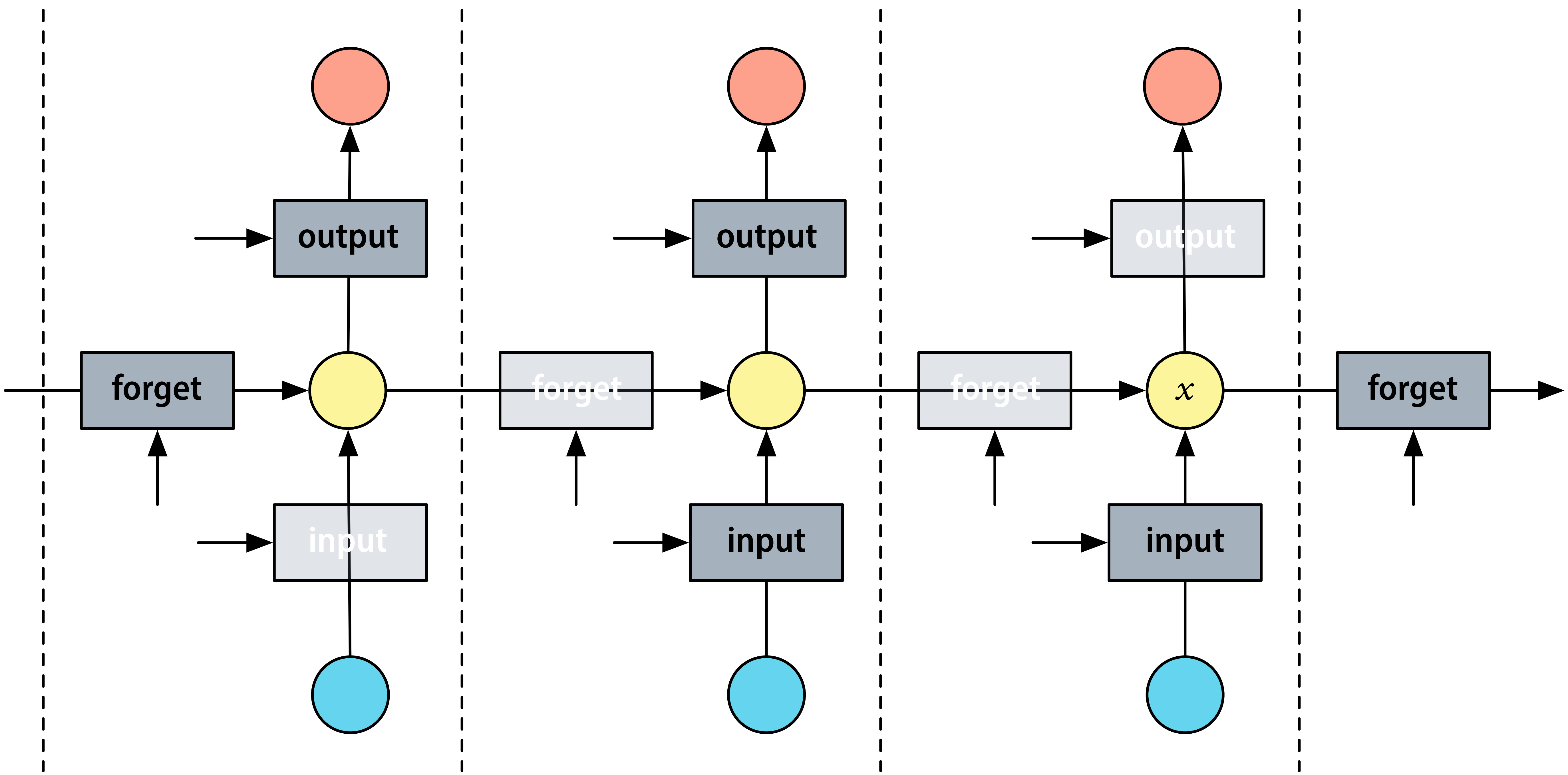
Attribution: Geoffrey Hinton

Information flow in an LSTM



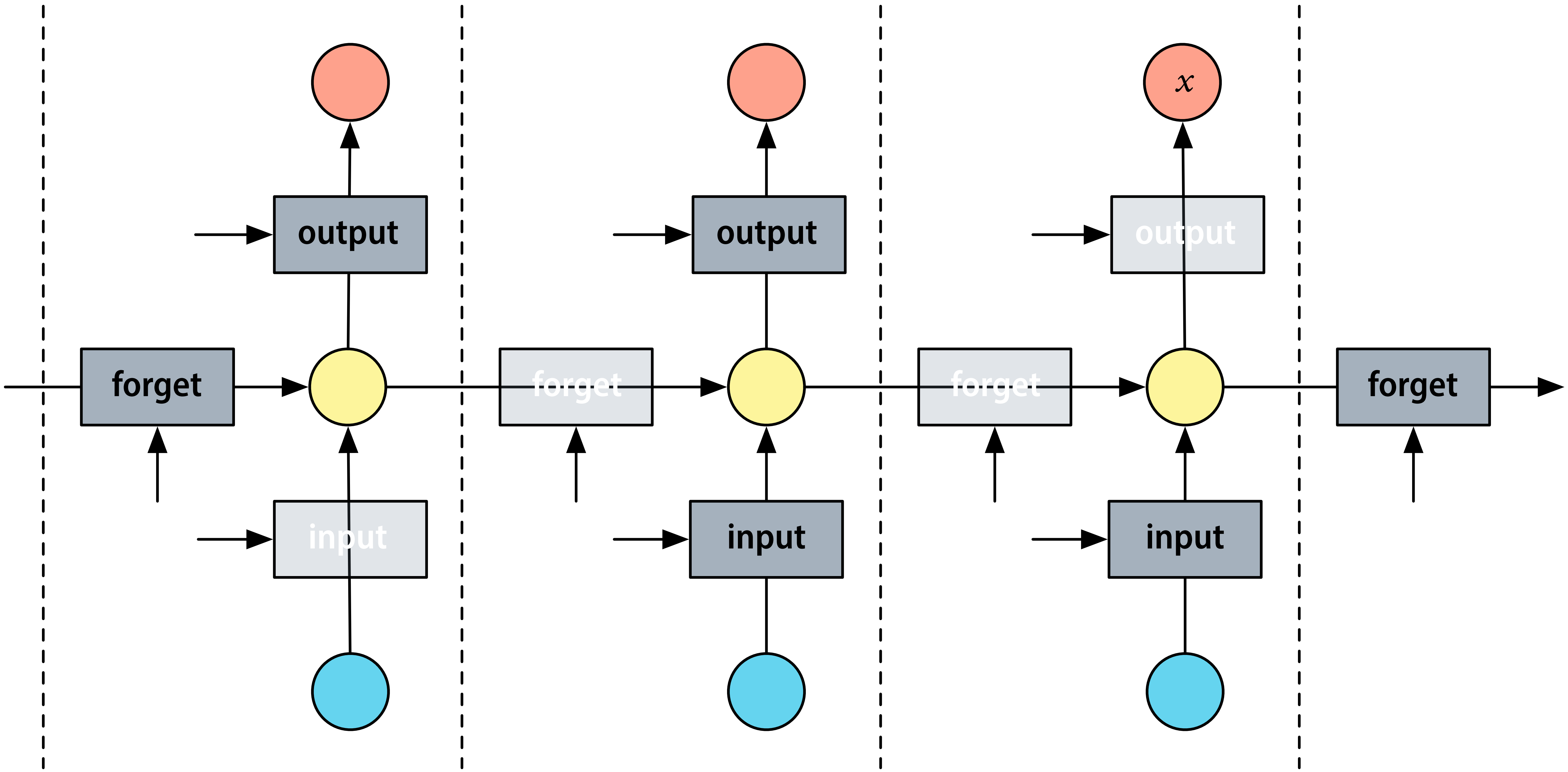
Attribution: Geoffrey Hinton

Information flow in an LSTM



Attribution: Geoffrey Hinton

Information flow in an LSTM



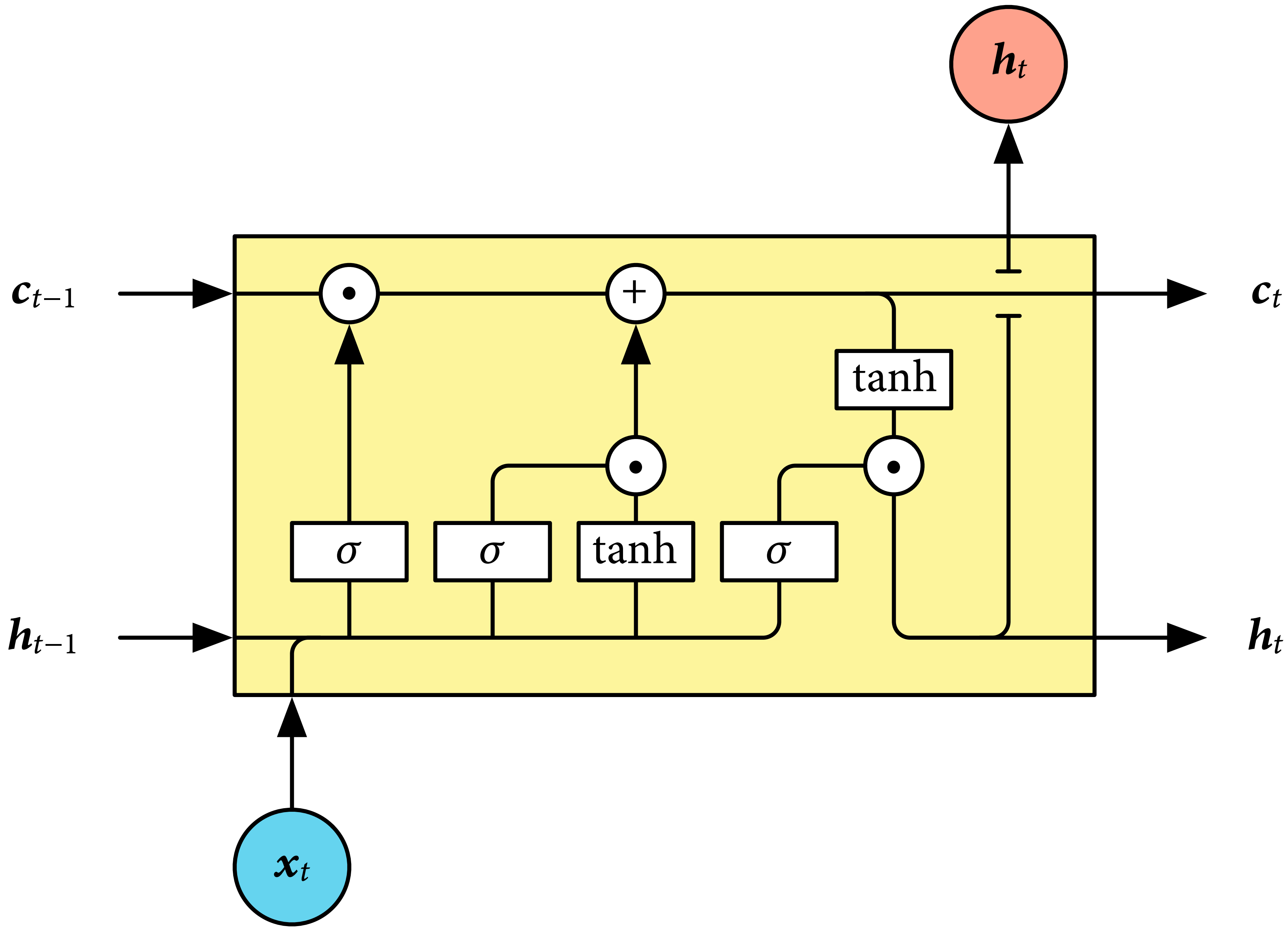
Attribution: Geoffrey Hinton

Gating mechanism

$$\underbrace{\begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix}}_{\boldsymbol{h}_{t-1}} \odot \underbrace{\begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}}_{\boldsymbol{g}} + \underbrace{\begin{bmatrix} 5 \\ 6 \\ 7 \\ 8 \end{bmatrix}}_{\boldsymbol{x}_t} \odot \underbrace{\begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}}_{1-\boldsymbol{g}} = \underbrace{\begin{bmatrix} 5 \\ 2 \\ 3 \\ 8 \end{bmatrix}}_{\boldsymbol{h}_t}$$

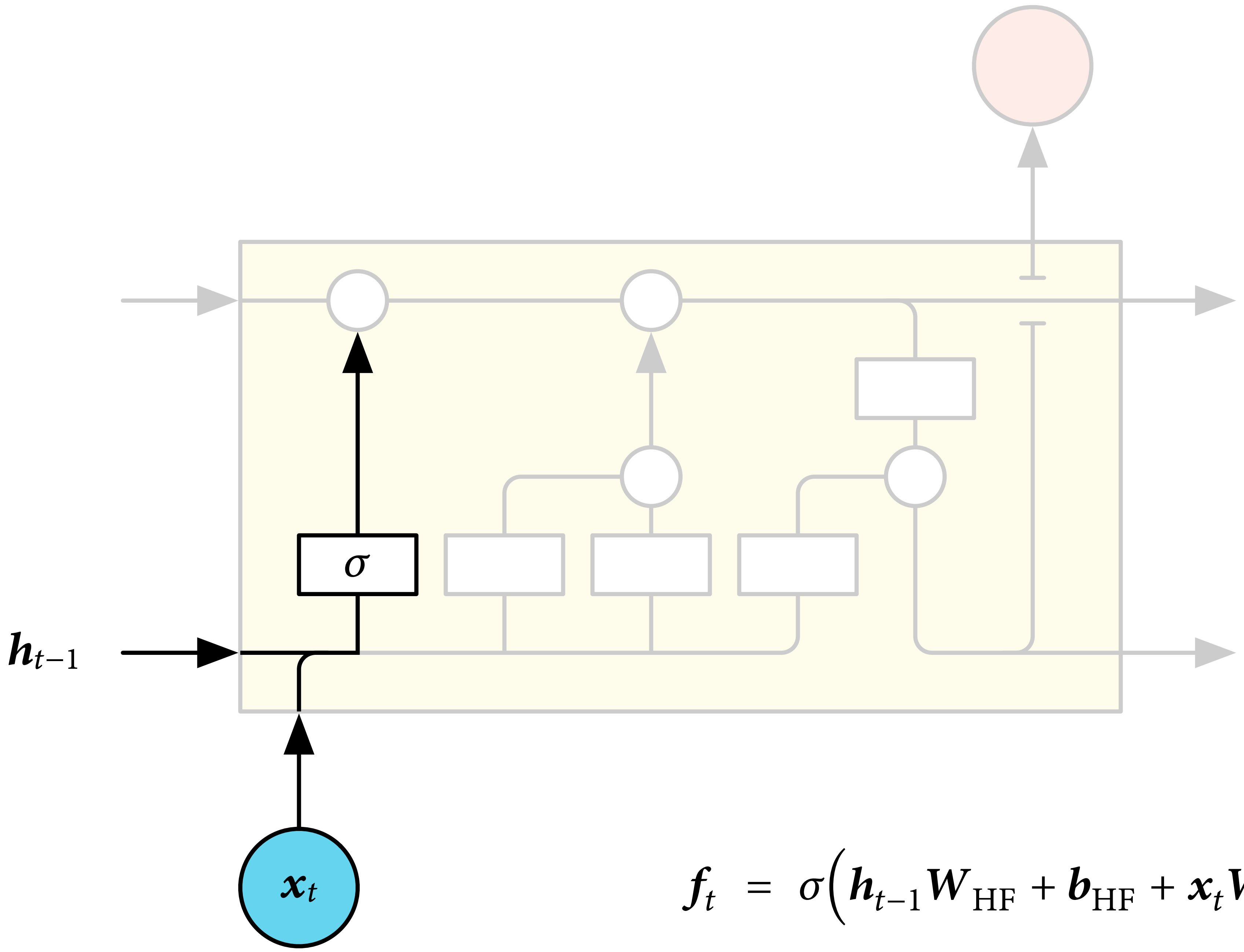
The gating masks $\boldsymbol{g}$ are learned values between 0 and 1.

A look inside an LSTM cell



Attribution: Chris Olah

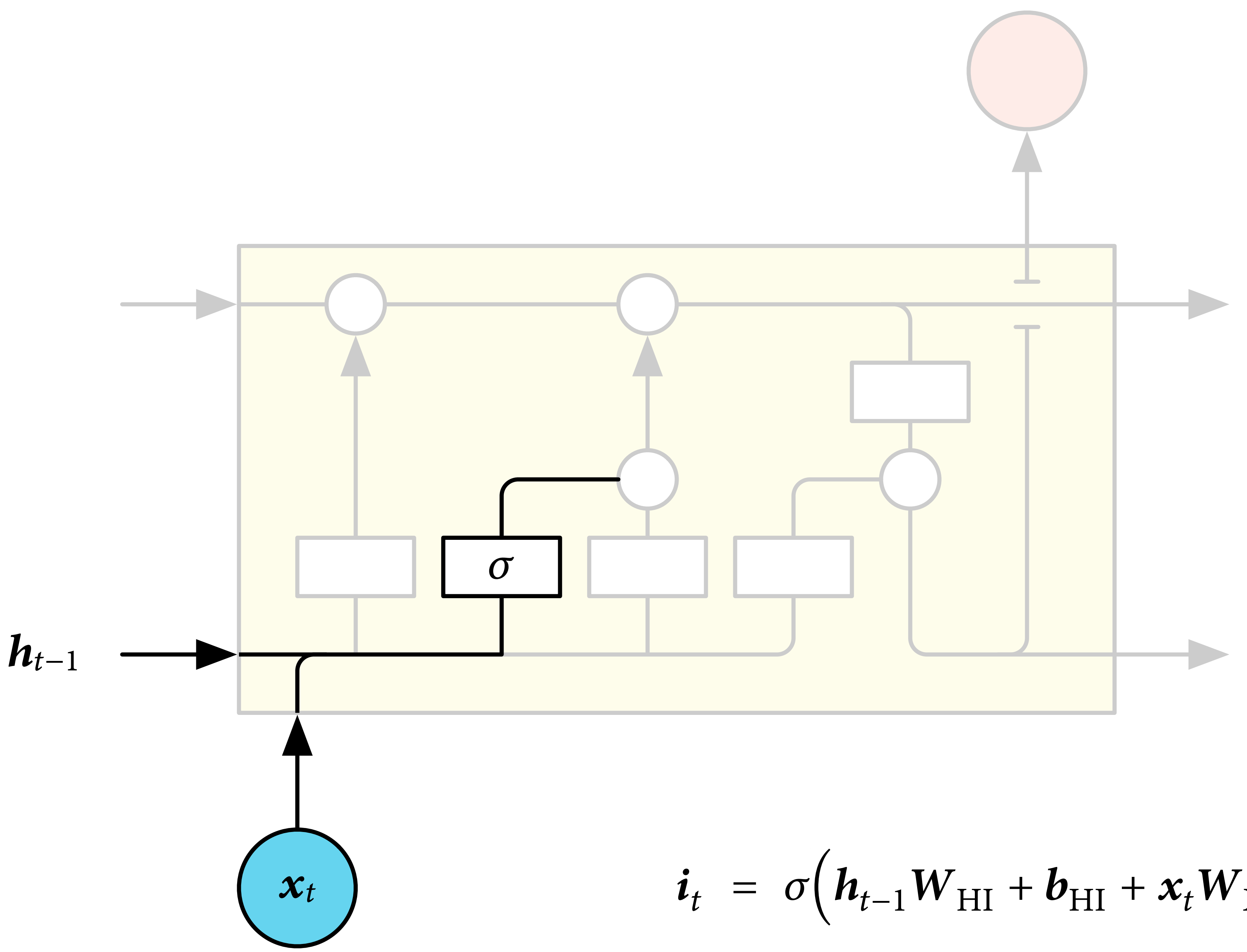
Forget gate



$$\boldsymbol{f}_t = \sigma\left(\boldsymbol{h}_{t-1}\boldsymbol{W}_{\mathrm{HF}} + \boldsymbol{b}_{\mathrm{HF}} + \boldsymbol{x}_t\boldsymbol{W}_{\mathrm{XF}} + \boldsymbol{b}_{\mathrm{XF}}\right)$$

Attribution: Chris Olah

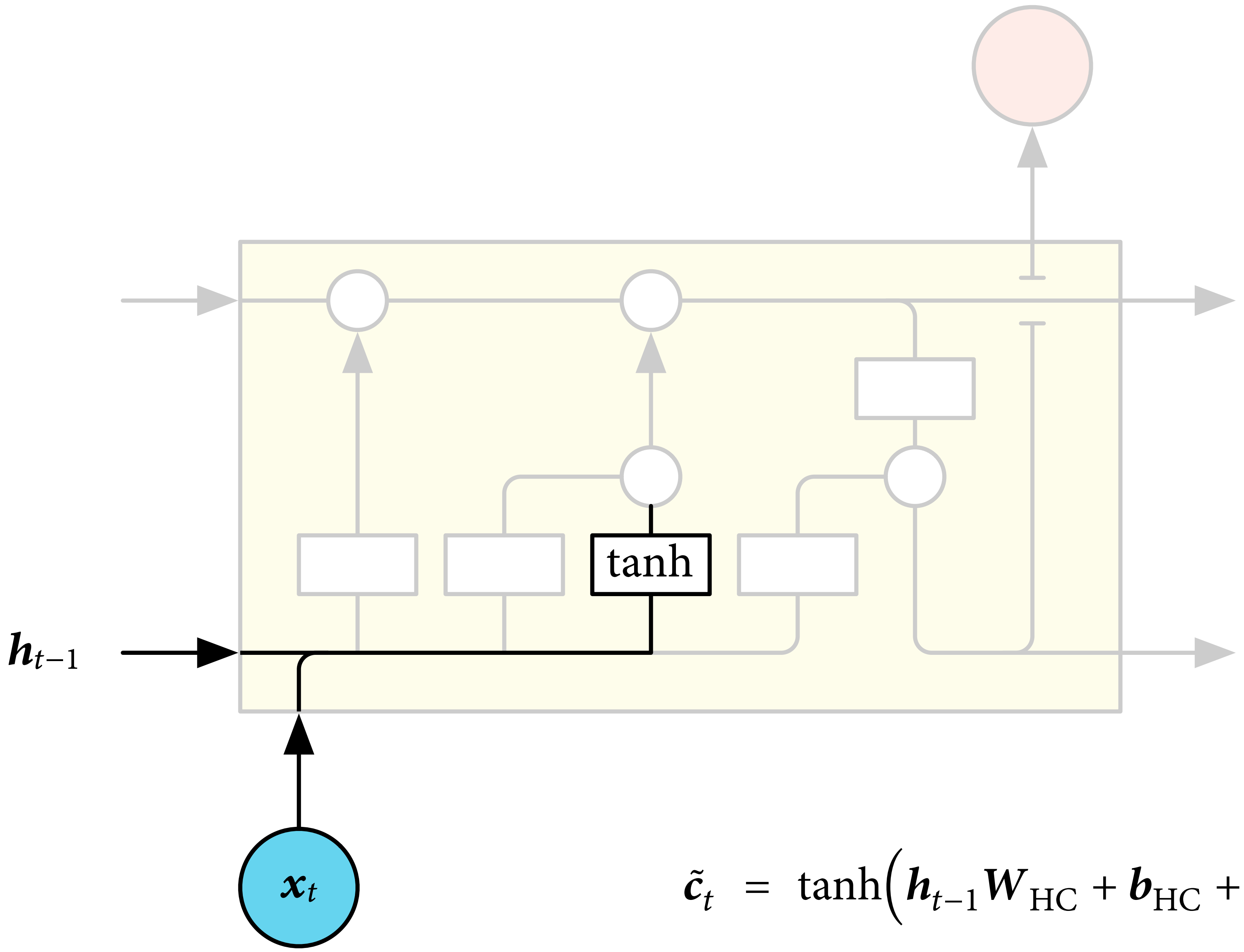
Input gate



$$i_t = \sigma\left(h_{t-1}W_{\mathrm{HI}} + b_{\mathrm{HI}} + x_t W_{\mathrm{XI}} + b_{\mathrm{XI}}\right)$$

Attribution: Chris Olah

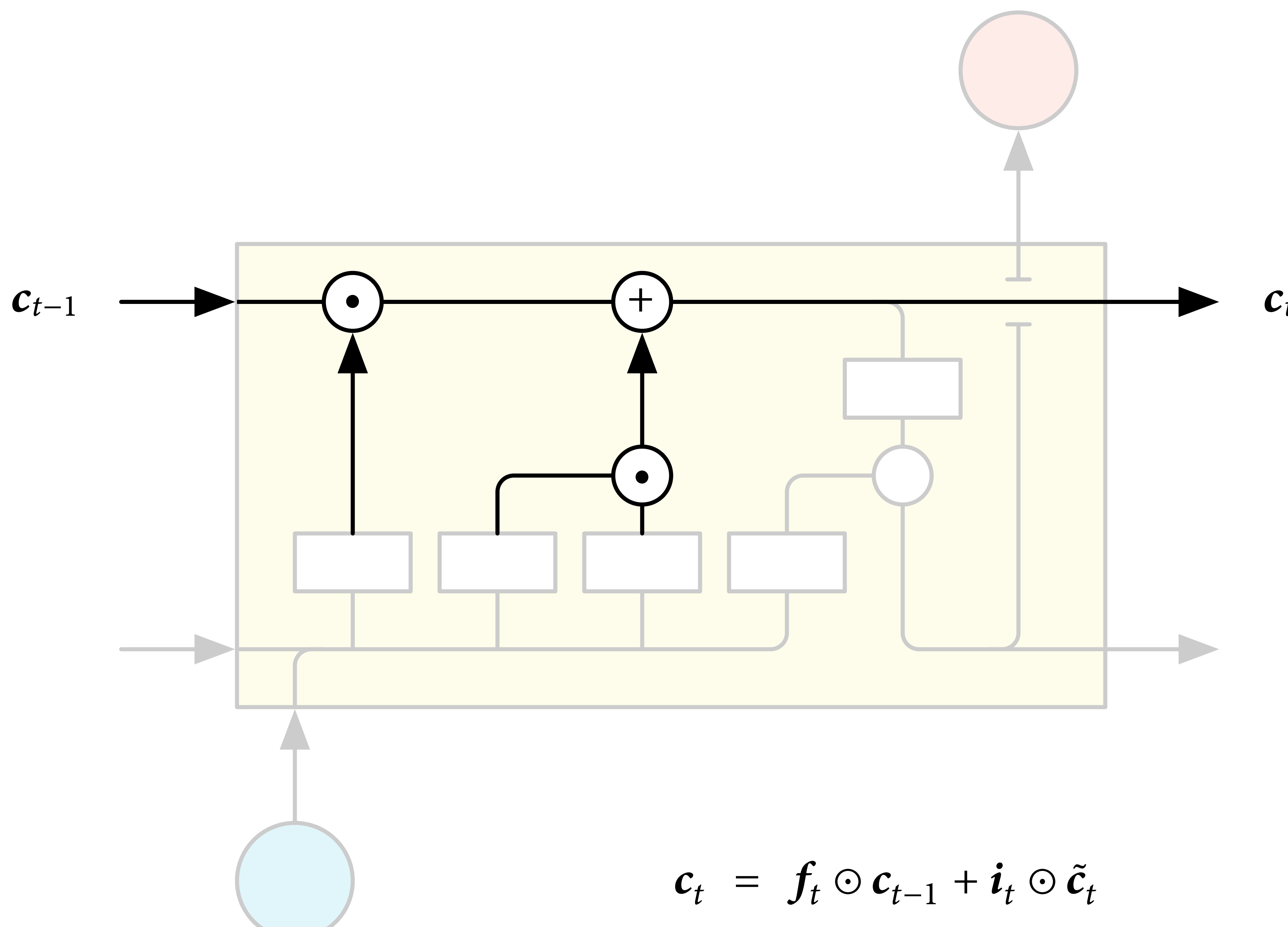
Update candidate



$$\tilde{c}_t = \tanh\left(h_{t-1}W_{\mathrm{HC}} + b_{\mathrm{HC}} + x_t W_{\mathrm{XC}} + b_{\mathrm{XC}}\right)$$

Attribution: Chris Olah

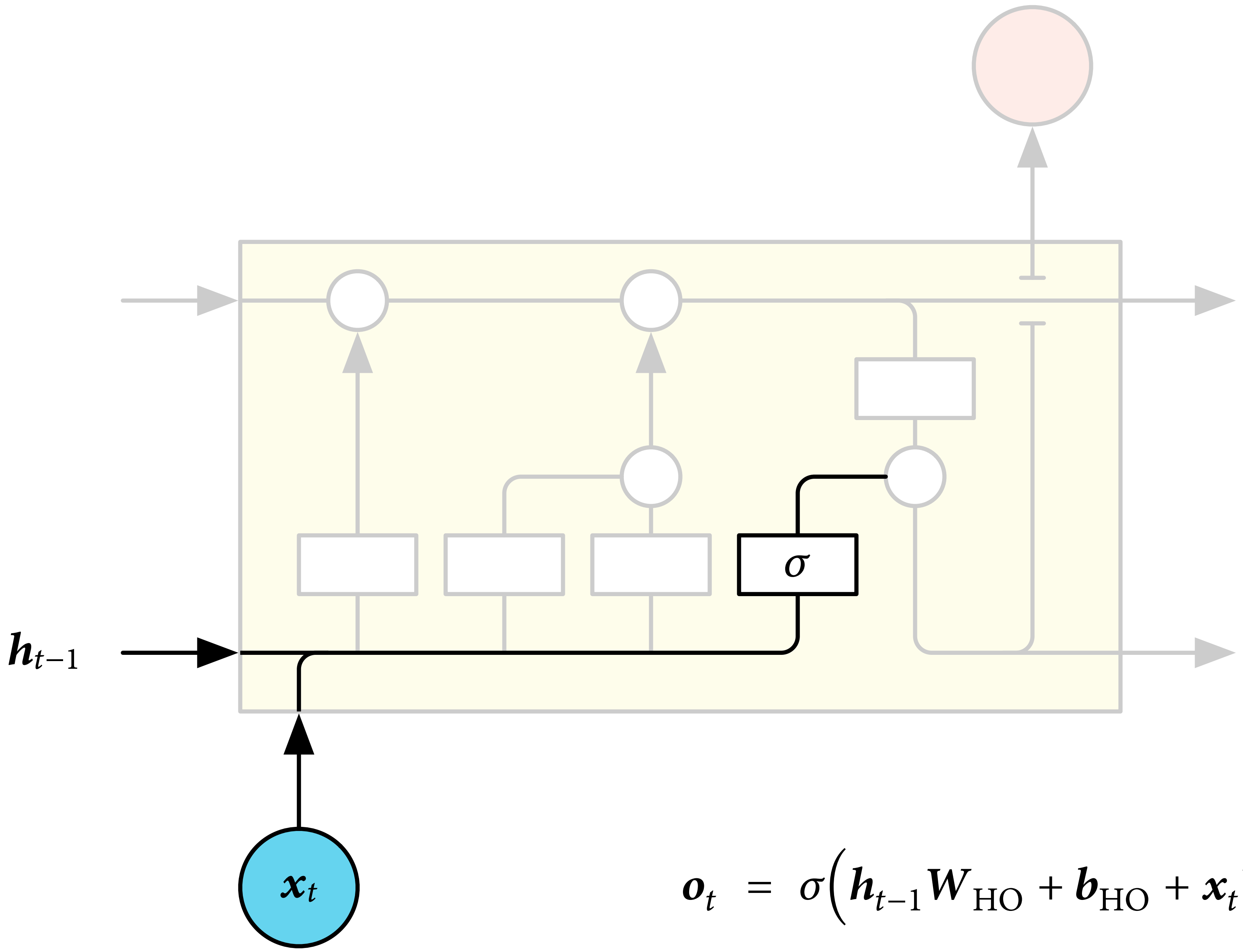
Memory cell update



$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t$$

Attribution: Chris Olah

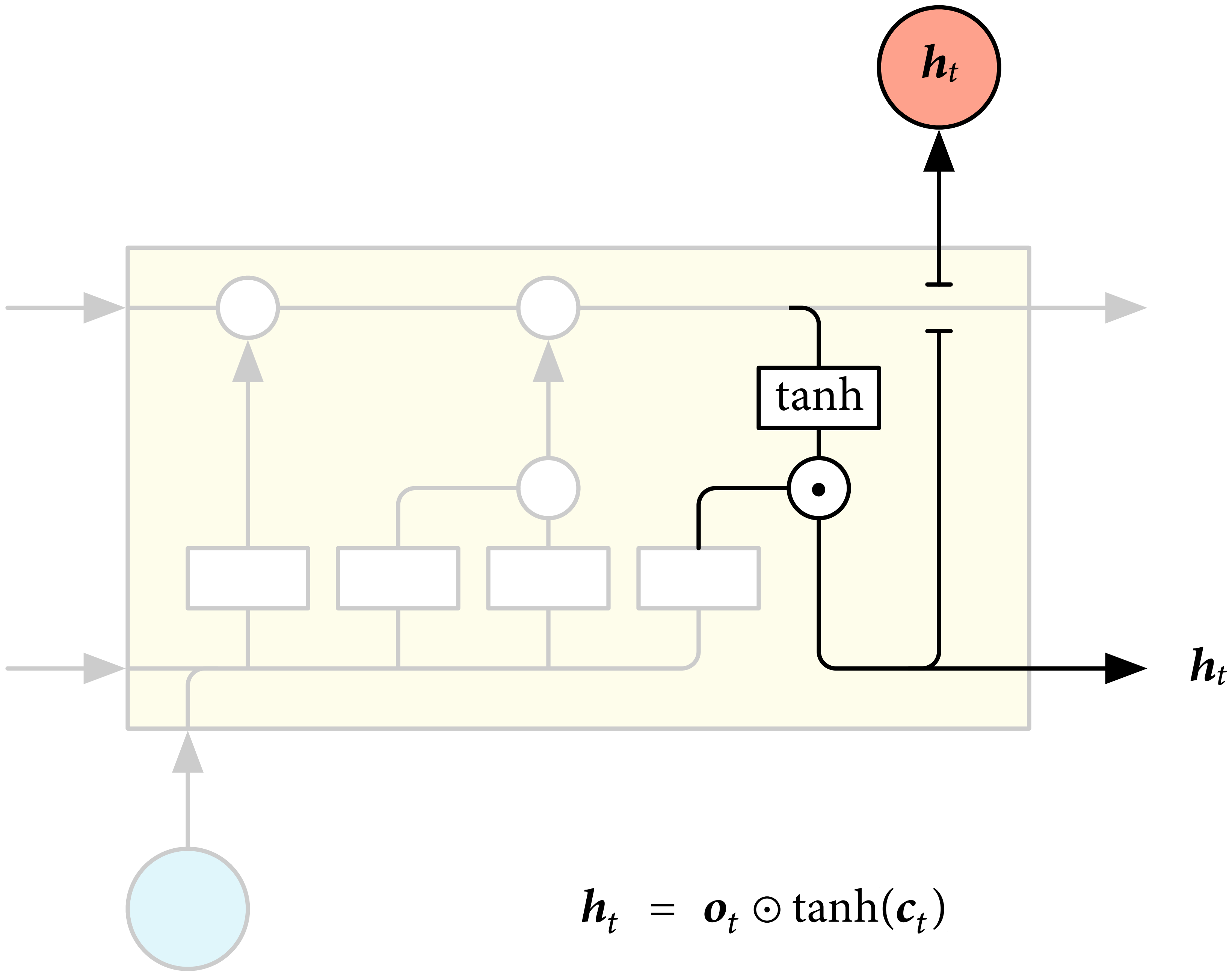
Output gate



$$o_t = \sigma\left(h_{t-1}W_{\mathrm{HO}} + b_{\mathrm{HO}} + x_tW_{\mathrm{XO}} + b_{\mathrm{XO}}\right)$$

Attribution: Chris Olah

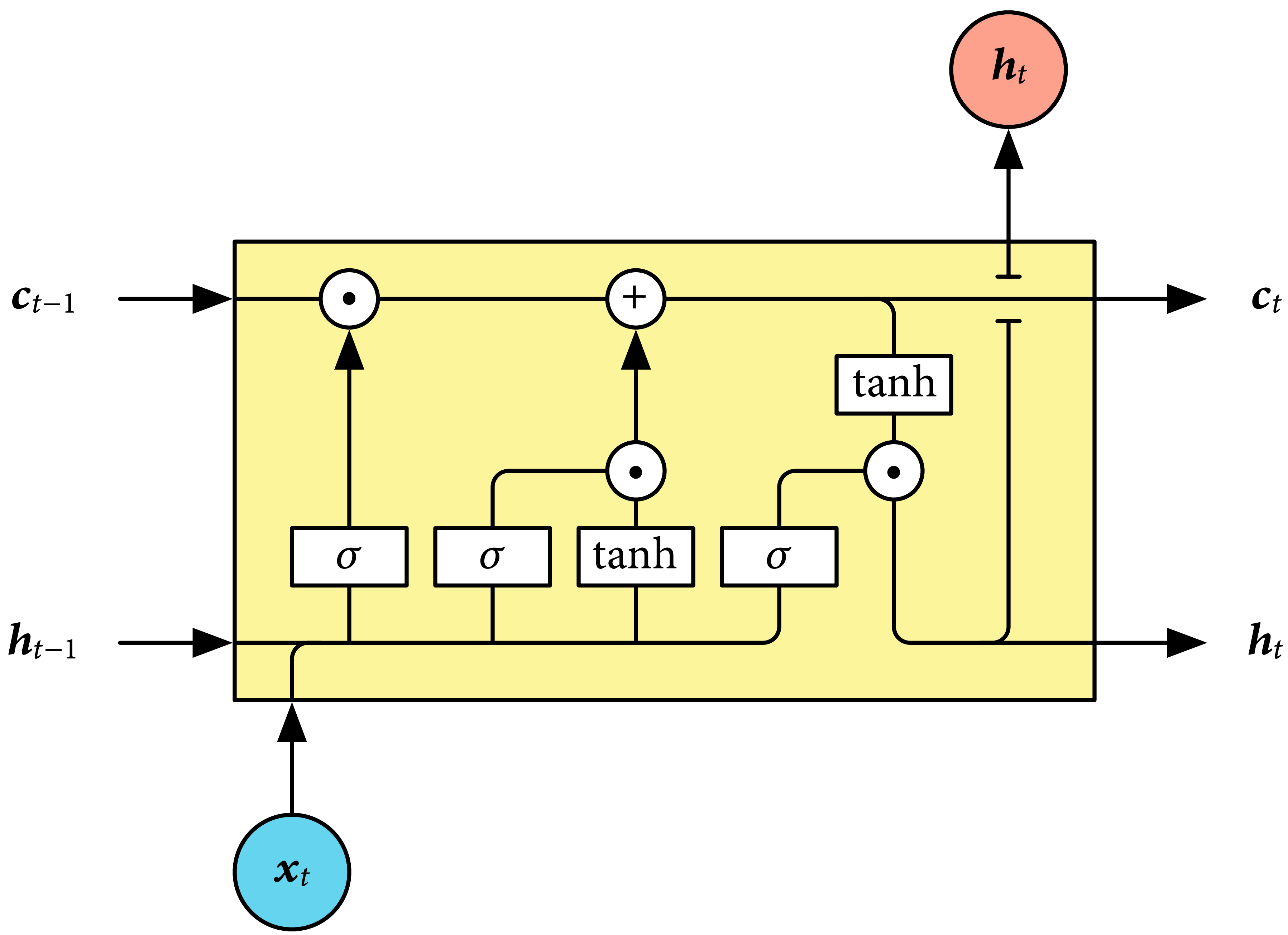
Output



$$h_t = o_t \odot \tanh(c_t)$$

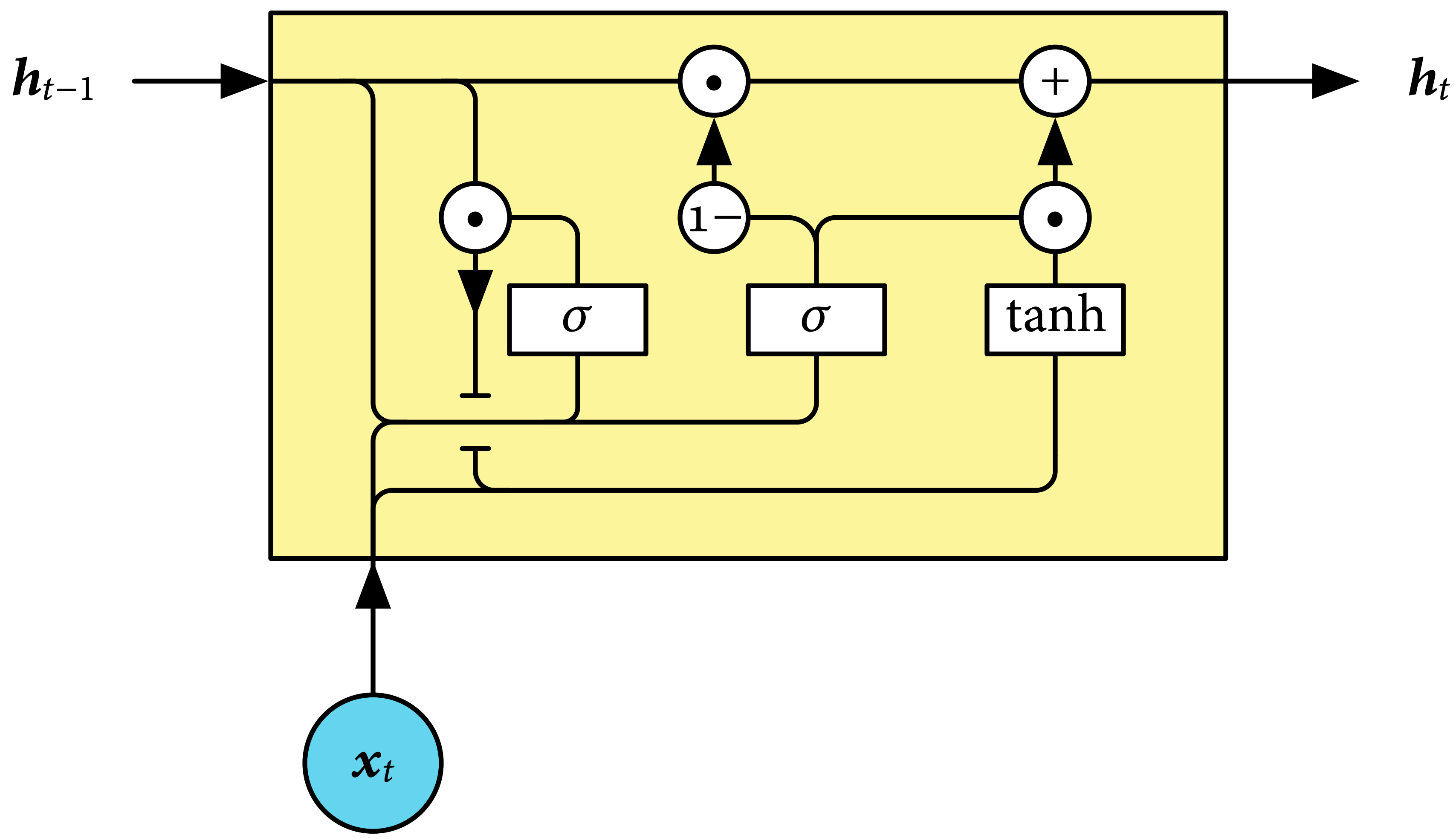
Attribution: Chris Olah

A look inside an LSTM cell



Attribution: Chris Olah

Gated Recurrent Unit (GRU)



Attribution: Chris Olah

This lecture

- Prelude: Word embeddings
- Introduction to recurrent neural networks
- The LSTM architecture
- Use case 1: Part-of-speech tagging
- Use case 2: Contextualised word embeddings

Use case 1: Part-of-speech tagging

Parts of speech

- A **part of speech** is a category of words that play similar roles within the syntactic structure of a sentence.
- Commonly listed English parts of speech include noun, verb, adjective, adverb, pronoun, preposition, and conjunction.
- Determining the parts of speech of a sentence is one of the first steps in the traditional NLP analysis pipeline.

Part-of-speech tagging, example

I	want	to	live	in	peace
PRON	VERB	PART	VERB	ADP	NOUN
pronoun		particle		adposition	

'I only want to live in peace, plant potatoes, and dream!' – Moomin

Universal part-of-speech tags

Tag	Category	Examples
ADJ	adjective	*big, old*
ADV	adverb	*very, well*
INTJ	interjection	*ouch!*
NOUN	noun	*girl, cat, tree*
VERB	verb	*run, eat*
PROPN	proper noun	*Mary, John*

Tag	Category	Examples
ADP	adposition	*in, to, during*
AUX	auxiliary verb	*has, was*
CCONJ	conjunction	*and, or, but*
DET	determiner	*a, my, this*
NUM	cardinal numbers	*0, one*
PRON	pronoun	*I, myself, this*

Missing: PART, SCONJ, PUNCT, SYM, X

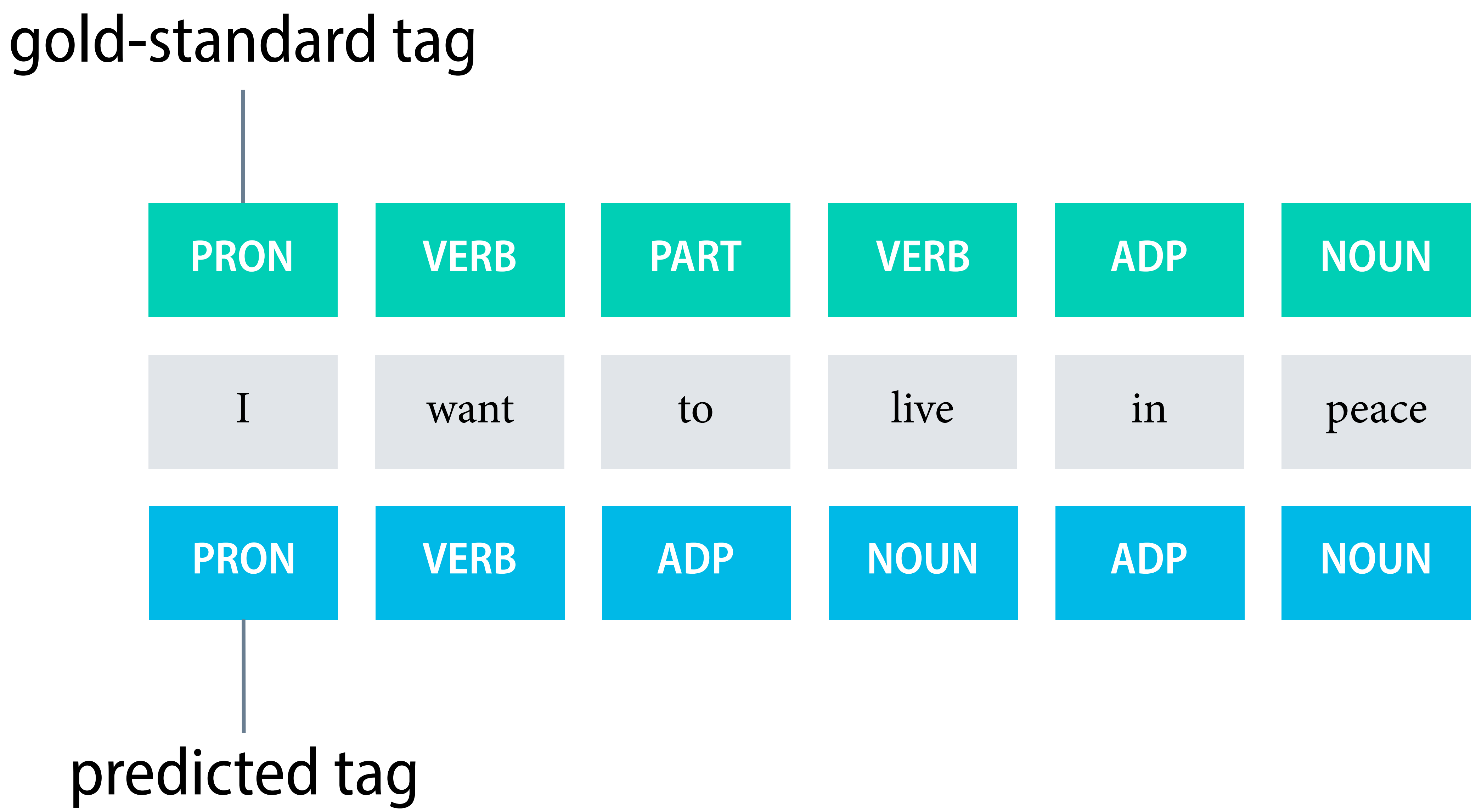
Source: Universal Dependencies Project

Part-of-speech tagging

- A **part-of-speech tagger** is a computer program that tags each word in a sentence with its part of speech.
- Part-of-speech tagging can be approached as a supervised machine learning problem. This requires training data.

  linguistic data sets with gold-standard part-of-speech annotation
- Part-of-speech taggers are commonly evaluated using accuracy, precision, and recall.

Evaluation of part-of-speech taggers



Accuracy

	DET	ADJ	NOUN	ADP	VERB
DET	923	0	0	0	1
ADJ	2	1255	132	1	5
NOUN	0	7	4499	1	18
ADP	0	0	0	2332	1
VERB	0	5	132	2	3436

$$\frac{12445}{12752} = 97.59\%$$

predicted tag

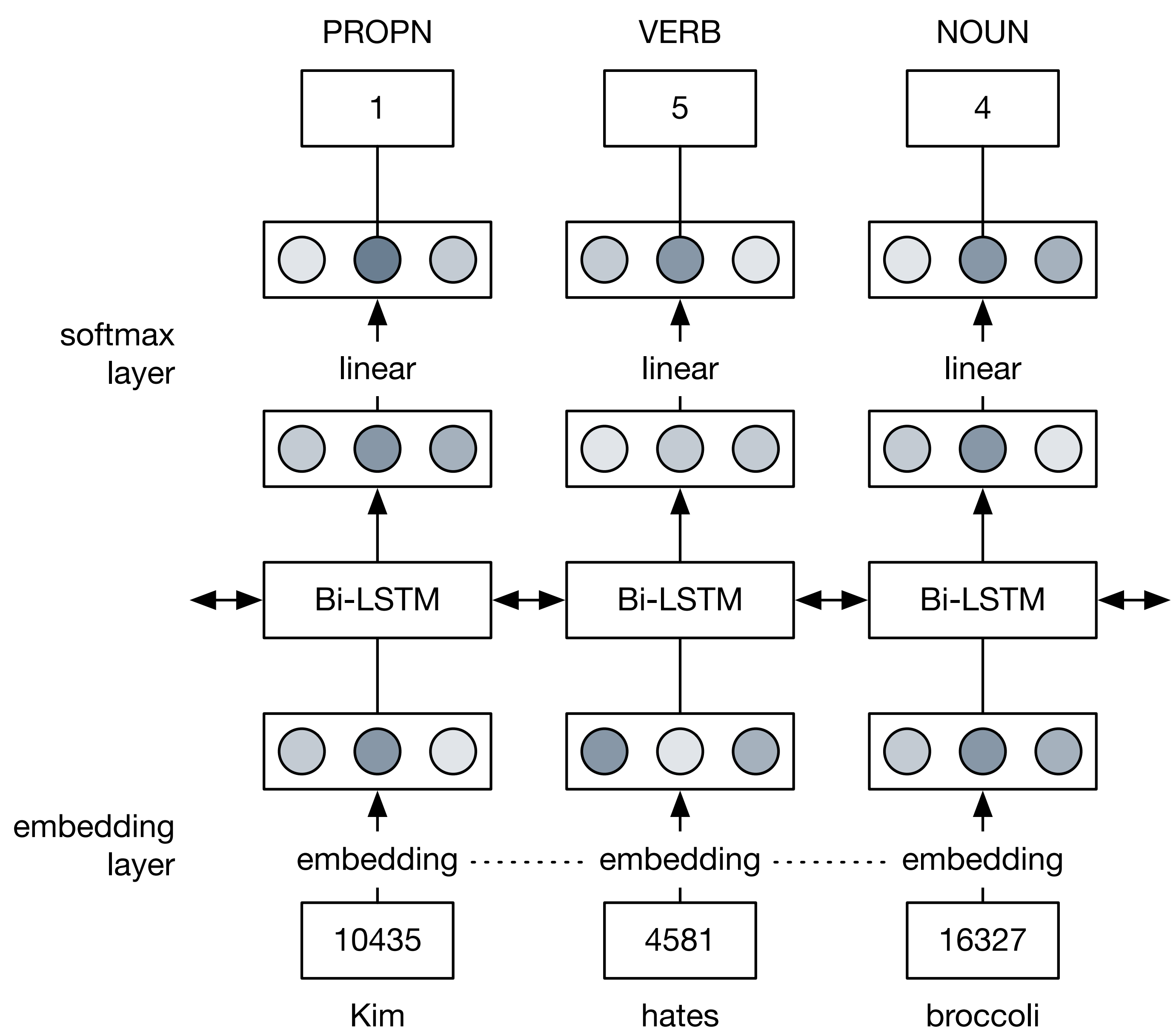
gold-standard tag

Ambiguity

I	want	to	live	in	peace
PRON	VERB	PART	VERB	ADP	NOUN
NOUN	NOUN	ADP	ADJ	ADV	VERB
		ADV	ADV	ADJ	
				NOUN	

‘I only want to live in peace, plant potatoes, and dream!’ – Moomin

Recurrent architecture for tagging



This lecture

- Prelude: Word embeddings
- Introduction to recurrent neural networks
- The LSTM architecture
- Use case 1: Part-of-speech tagging
- Use case 2: Contextualised word embeddings

Use case 2: Contextualised word embeddings

Contextualised embeddings

- In standard word embeddings, each word is assigned a single word vector, independently of its context.

- Such a model cannot account for **polysemy**, the phenomenon that one and the same word may have multiple meanings.

  The children *play* in the park. The *play* premiered yesterday.

- In **contextualised embeddings**, each token is assigned a representation that is a function of its context.

ELMo – Embeddings from Language Models

- A token is represented as a task-specific, weighted sum of representations derived from a bidirectional language model.

  weights are learned for a specific task

- The basic ELMo model is frozen after pre-training and can complement or replace a standard word embedding layer.

- However, it is often beneficial to fine-tune a pre-trained ELMo model on task-specific data.

Peters et al. (2018)

Muppet character image from The Muppet Wiki

Pre-trained word embeddings

- Standard (static) word embeddings can be pre-trained on any text corpus using available tools.

  word2vec, Gensim

- Pre-trained word embeddings for English, Swedish, and various other languages are available for download.

  word2vec, GloVe, Polyglot project, spaCy

Language modelling

- **Language modelling** is the task of predicting which word comes next in a sequence of words.

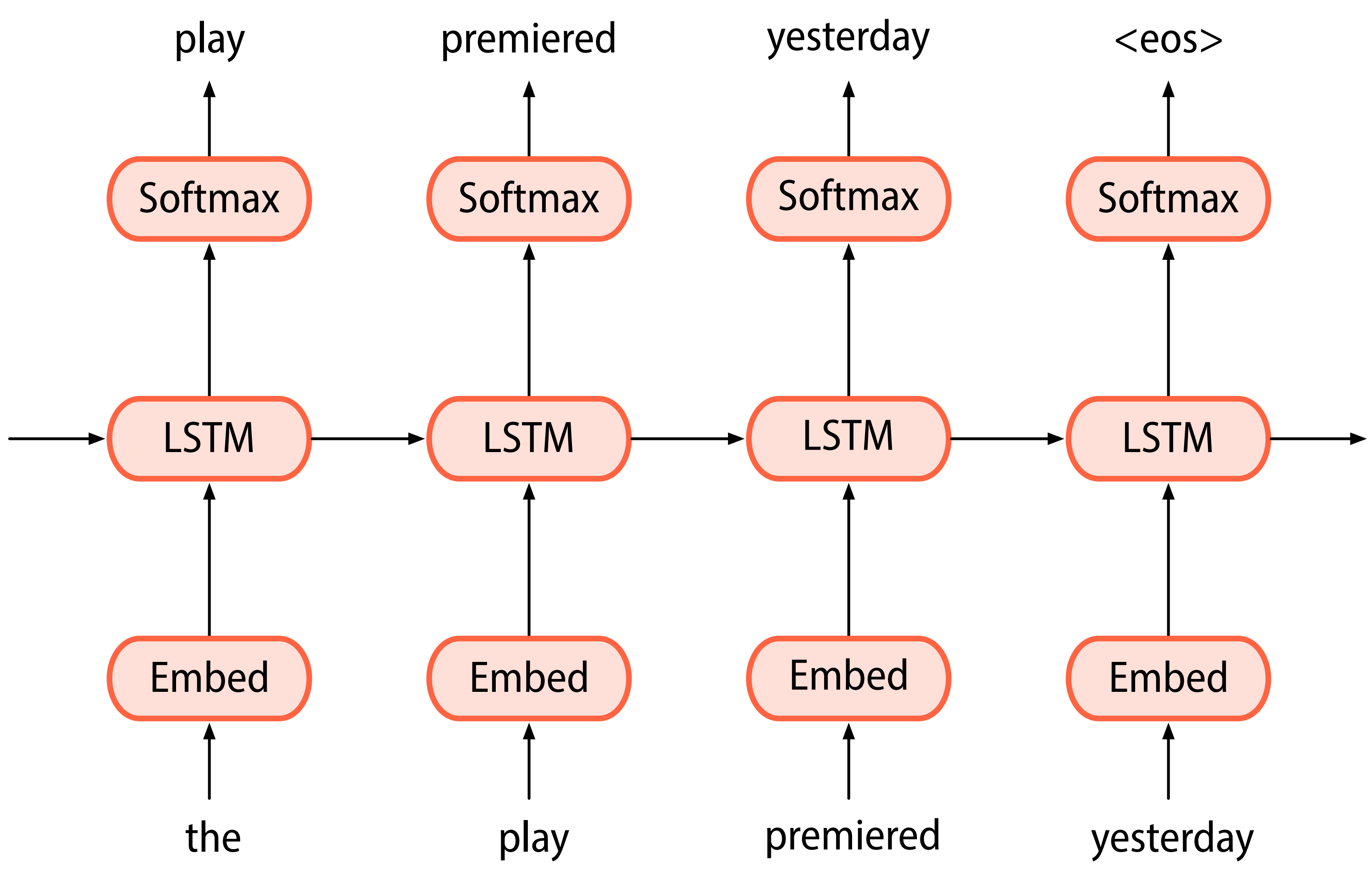
- More formally, given a sequence of words $w_1, \ldots, w_t$, we want to know the probability of the next word, w_{t+1}:

$$P(w_{t+1} \mid w_1, \ldots, w_t)$$

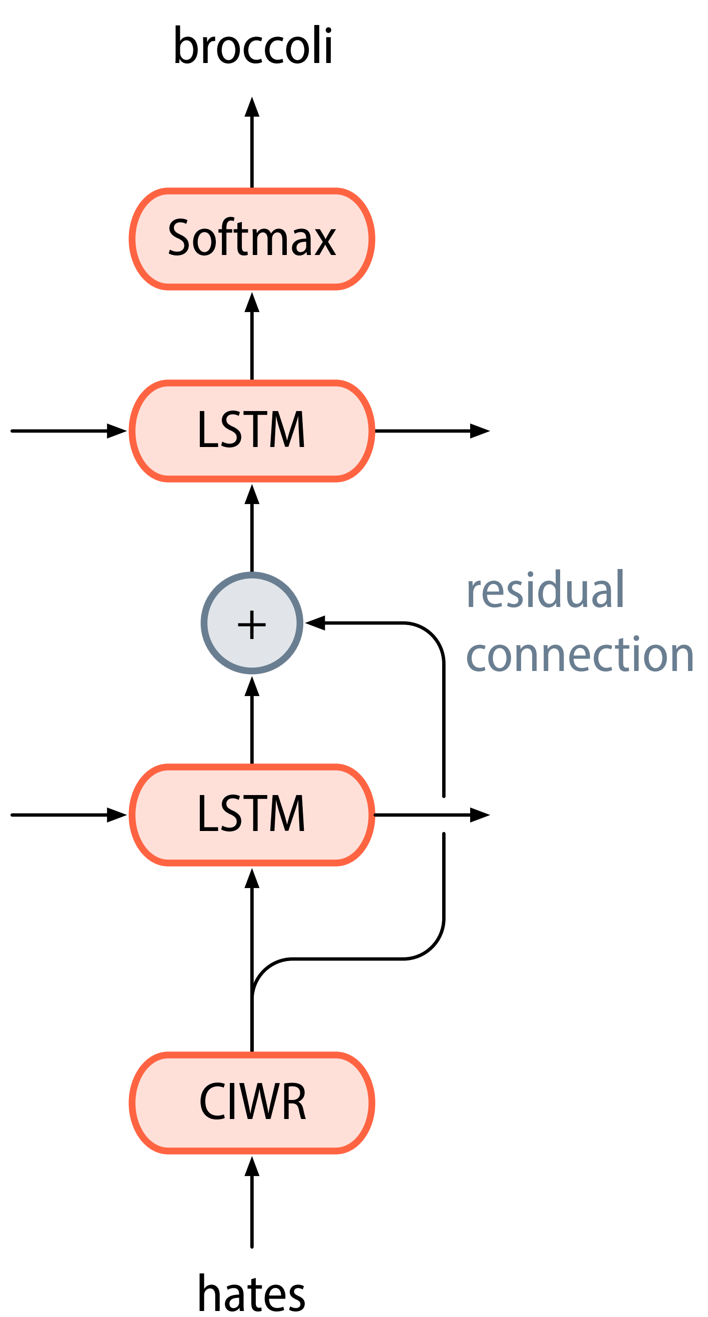
- Here we are assuming that w_{t+1} comes from a fixed vocabulary V.

  This allows language modelling to be treated as a classification task.

LSTM language model



ELMo architecture



- context-insensitive word representation using character convolutions

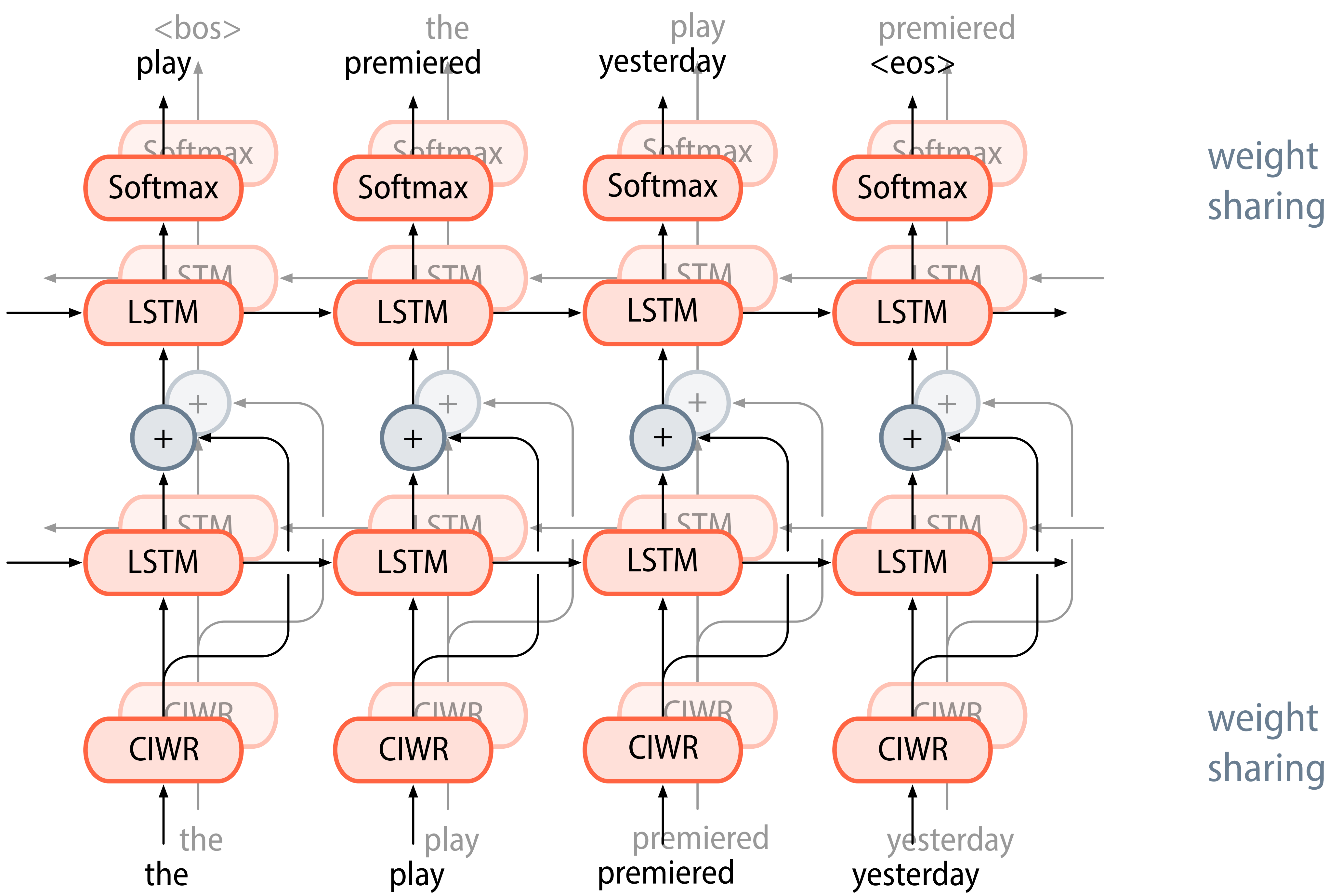
  followed by 2 highway layers, linear projection

- bidirectional LSTM layers with a residual connection between the layers

  4,096 hidden units; projected down to 512 units

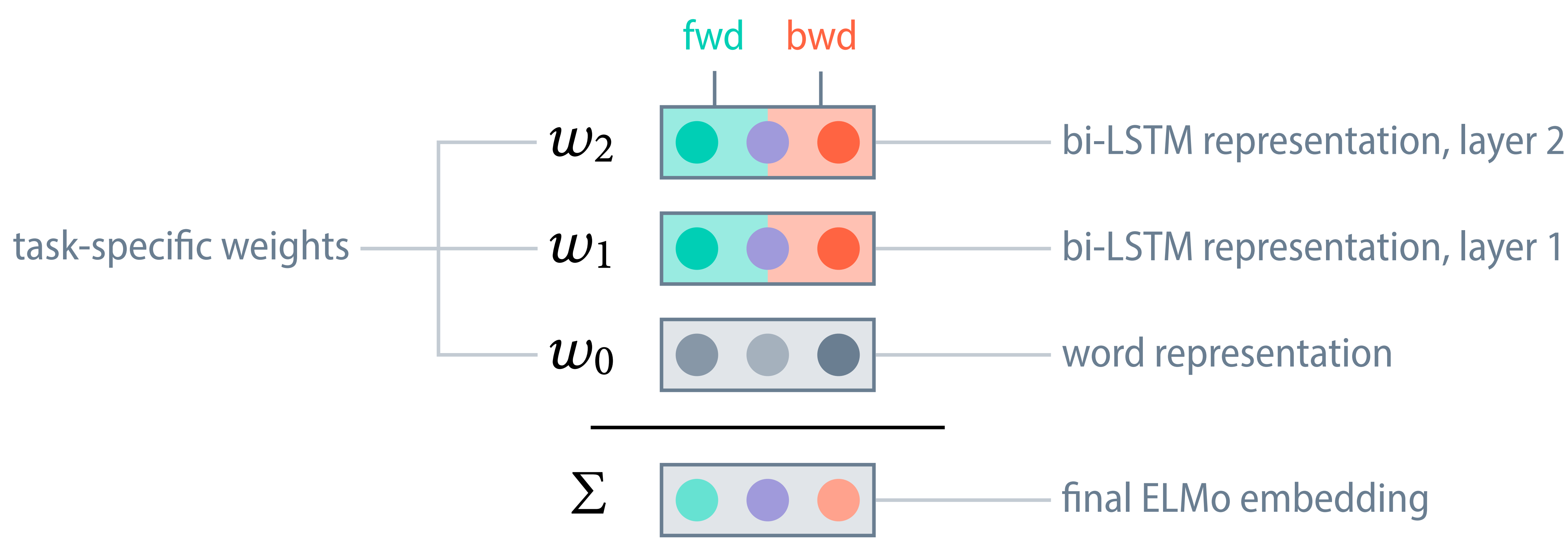
- final softmax layer computes a probability distribution over the next tokens

Bidirectional language model



ELMo – Embeddings from Language Models

ELMo is a task-specific weighted sum of the intermediate representations in the bidirectional language model.



Relative improvements by using ELMo embeddings

Task	Baseline	+ ELMo	Relative increase
Question answering (SQuAD)	81.1	85.8	24.9%
Coreference resolution (Coref)	67.2	70.4	9.8%
Sentiment analysis (SST-5)	51.4	54.7	6.8%
Textual entailment (SNLI)	88.0	88.7	5.8%

This lecture

- Prelude: Word embeddings
- Introduction to recurrent neural networks
- The LSTM architecture
- Use case 1: Part-of-speech tagging
- Use case 2: Contextualised word embeddings