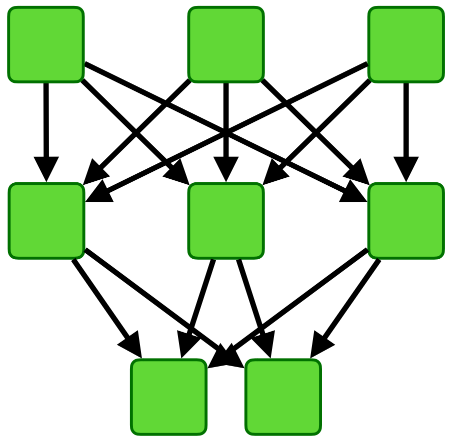
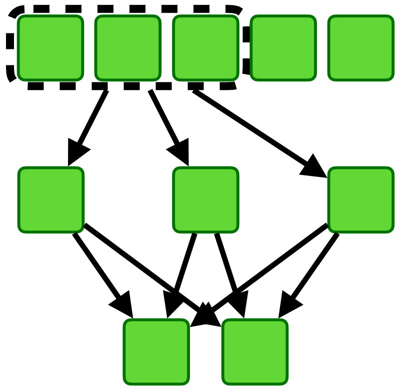
Recurrent Neural Networks:
Long short-term memory (LSTM)

Recap

Multi-layer perceptrons and (basic) convolutional neural networks are **feed-forward** (no cycles):

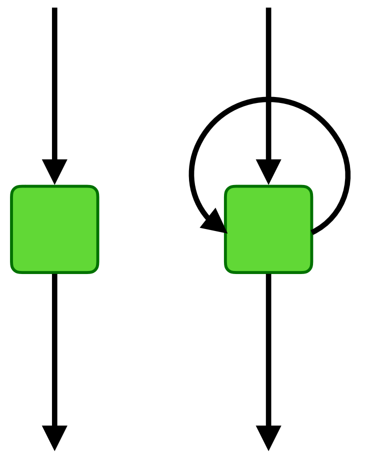


- One-directional information flow, one-to-one input-output mapping, fixed number of computational steps

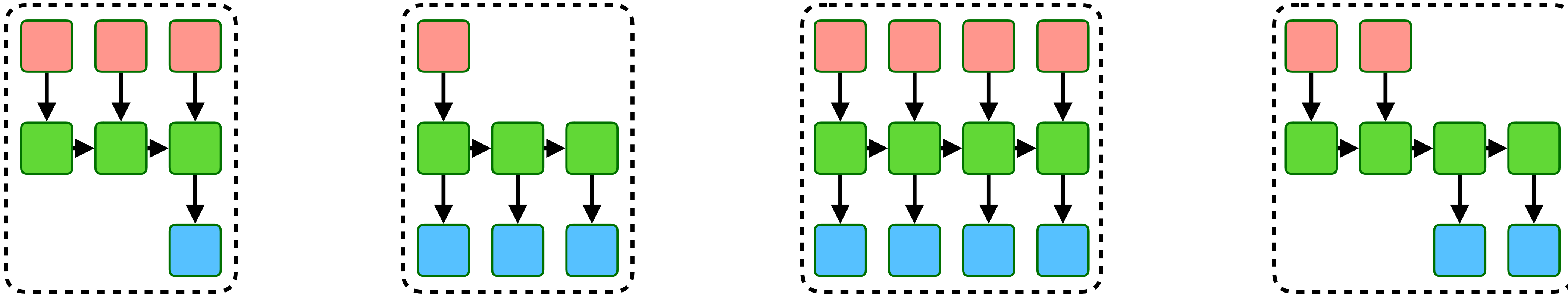


Recurrent neural networks have cycles:

- Information flowing back → state / memory of past inputs
- Variable-length inputs and outputs, number of processing steps tied to sequence length
- one-to-many, many-to-one, and many-to-many input-output mappings



Recap



Natural language data is inherently sequential, and RNNs can be applied to many NLP tasks:

- Many-to-one: text classification, text generation
- One-to-many: image captioning
- Many-to-many (paired inputs and outputs): part-of-speech tagging, named entity recognition
- Many-to-many (variable-length inputs and outputs): machine translation, dialogue systems, chatbots

Recap

Vanilla RNN cell

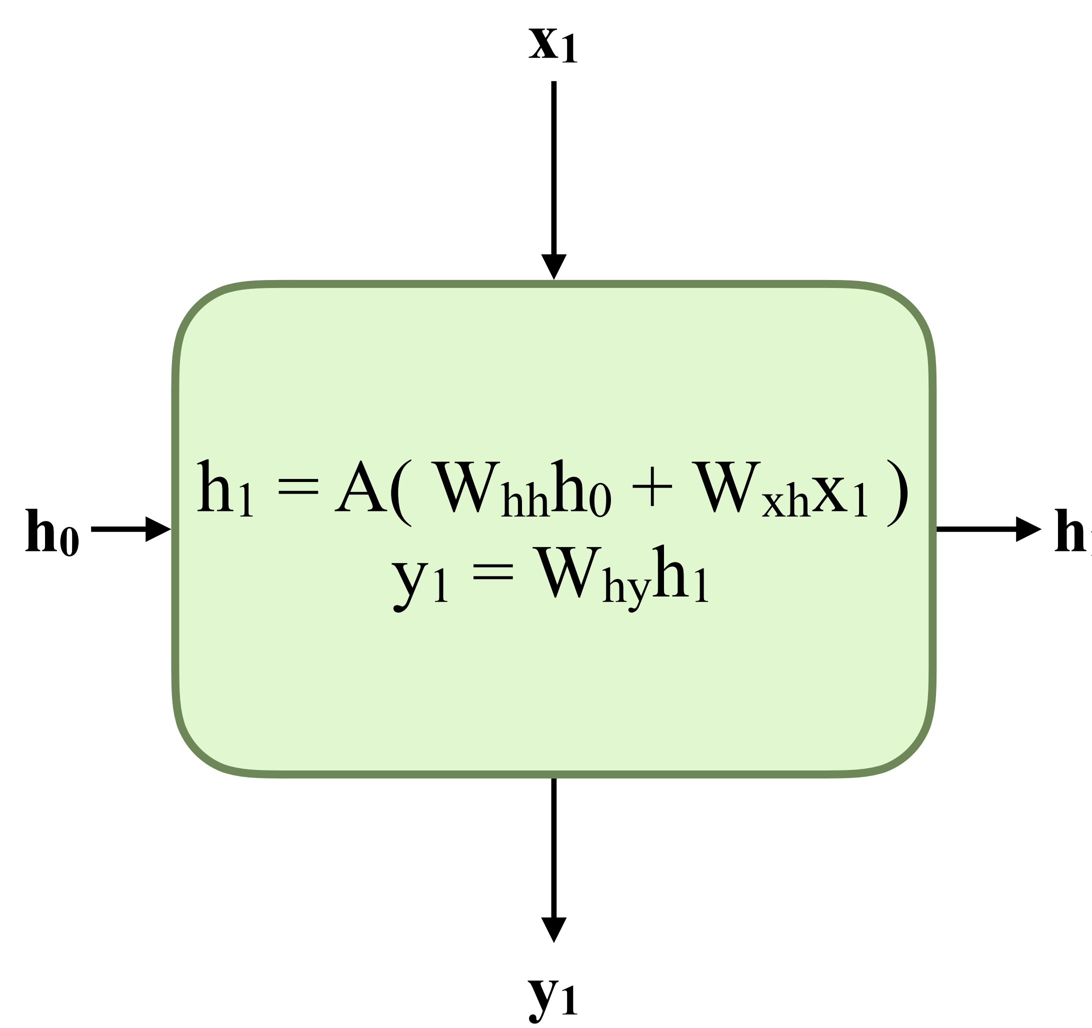
$h0$: hidden state for step 0

$x1$: input for step 1

W_{hh}, W_{xh}, W_{hy}: weights

A : activation function (e.g. tanh)

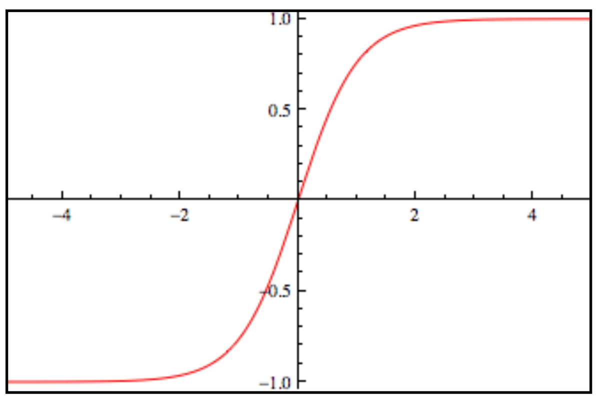
$y1$: output

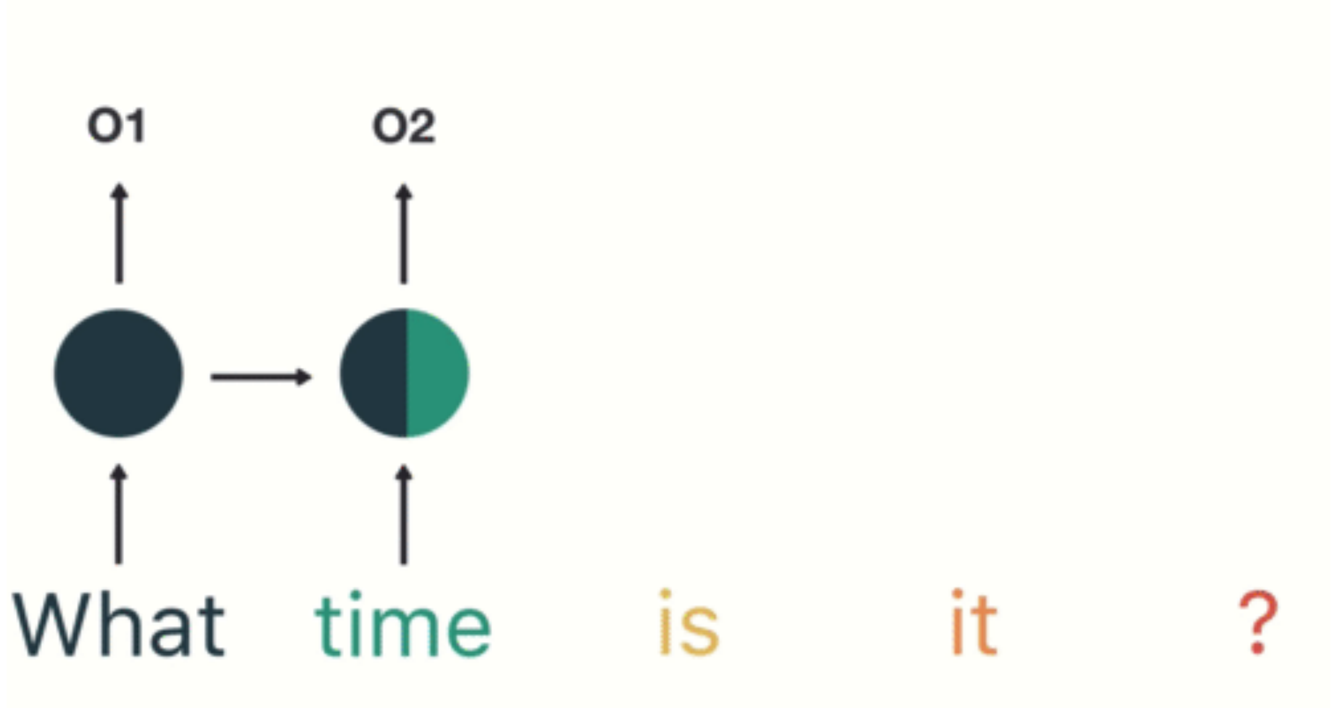


Recap

Vanilla RNNs have poor short-term memory and difficulty with long-term dependencies

State h repeatedly “squished” together with inputs by activation function





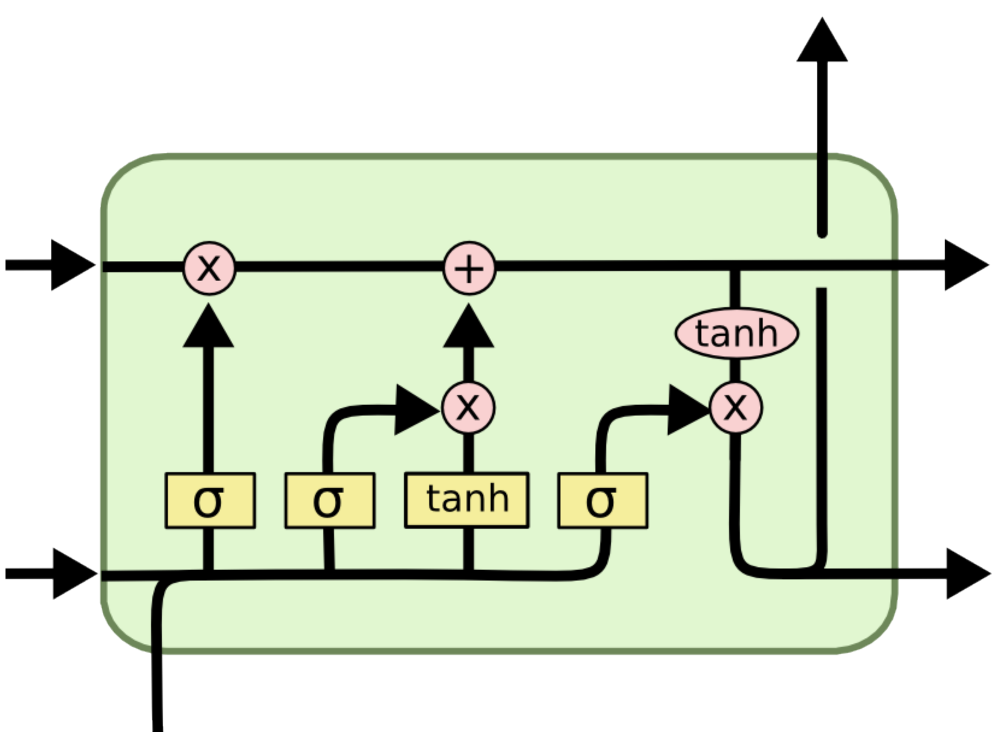
Hidden state at step 5

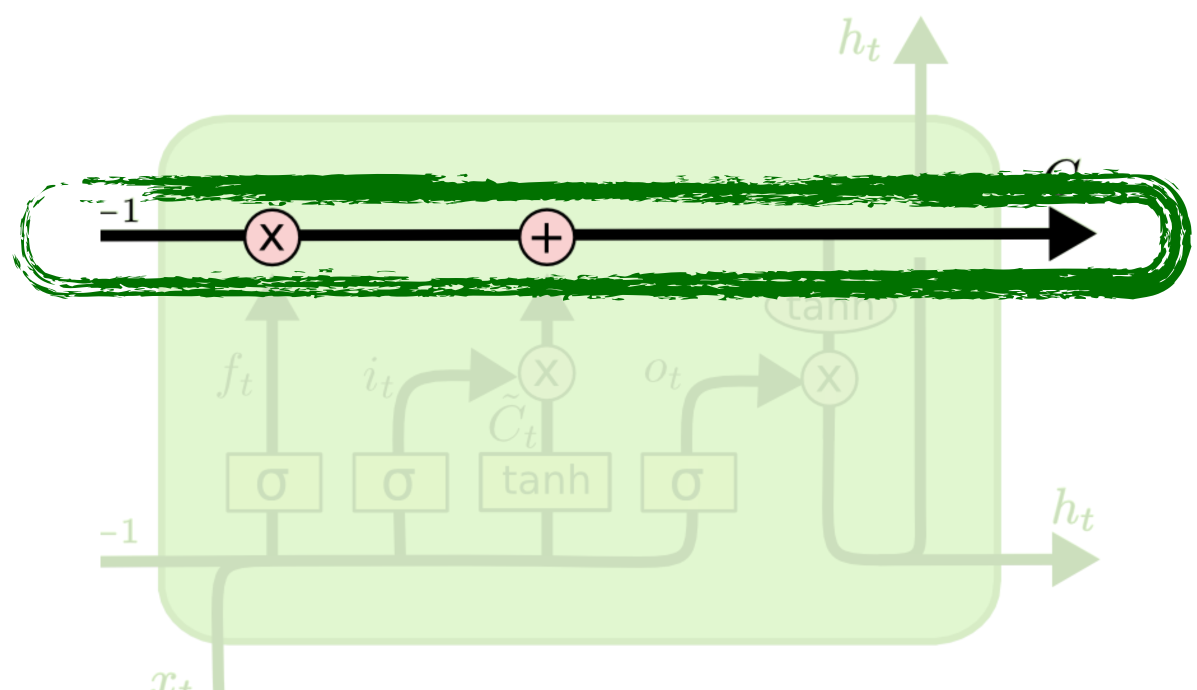
Illustration from Nguyen (2018) *Illustrated Guide to Recurrent Neural Networks*

LSTMs in (more) detail

RNN cell design aiming to alleviate short-term memory issues

Key point: cell state ("memory") passed through unmodified "by default", changes controlled by series of gates

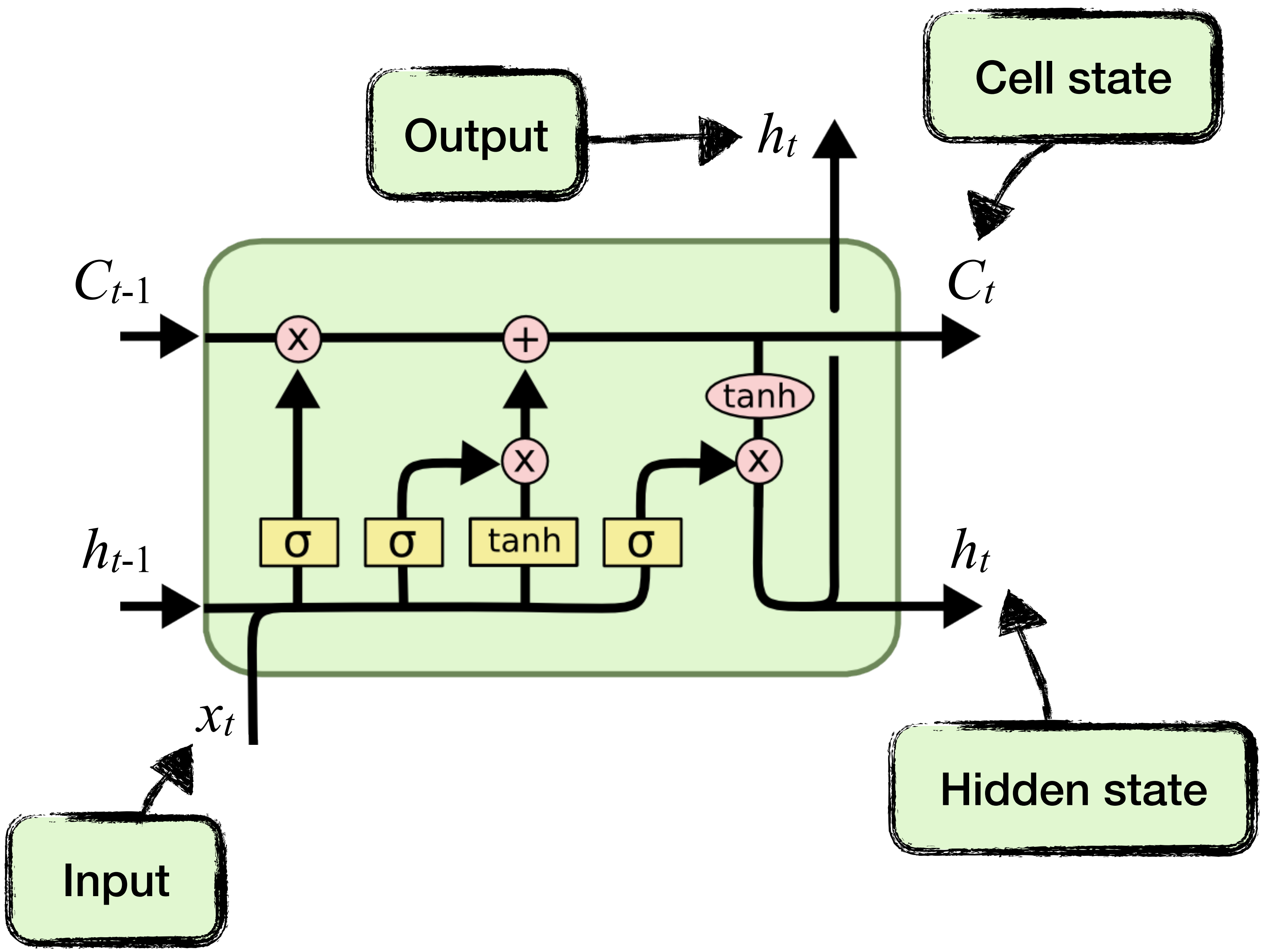




Hochreiter and Schmidhuber (1997) *Long short-term memory*

LSTM illustrations from Olah (2015) *Understanding LSTM Networks*

LSTMs in (more) detail



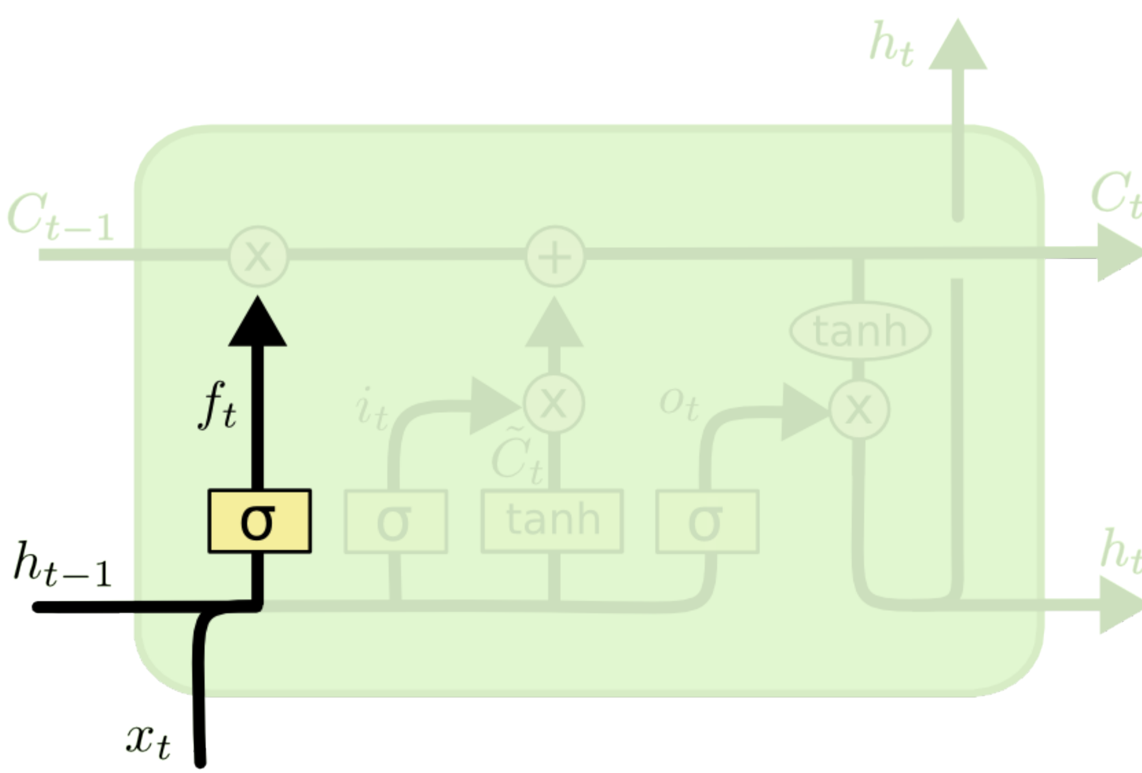
$$
\begin{aligned}
f_t &= \sigma\,(\ W_f x_t + U_f\, h_{t\text{-}1}\ ) \\
i_t &= \sigma\,(\ W_i\, x_t + U_i\, h_{t\text{-}1}\ ) \\
o_t &= \sigma\,(\ W_o\, x_t + U_o\, h_{t\text{-}1}\ ) \\
\tilde{C}_t &= \tanh(\ W_c\, x_t + U_c\, h_{t\text{-}1}\ ) \\
C_t &= f_t * C_{t\text{-}1} + i_t * \tilde{C}_t \\
h_t &= o_t * \tanh(C_t)
\end{aligned}
$$

Hochreiter and Schmidhuber (1997) *Long short-term memory*

LSTM illustrations from Olah (2015) *Understanding LSTM Networks*

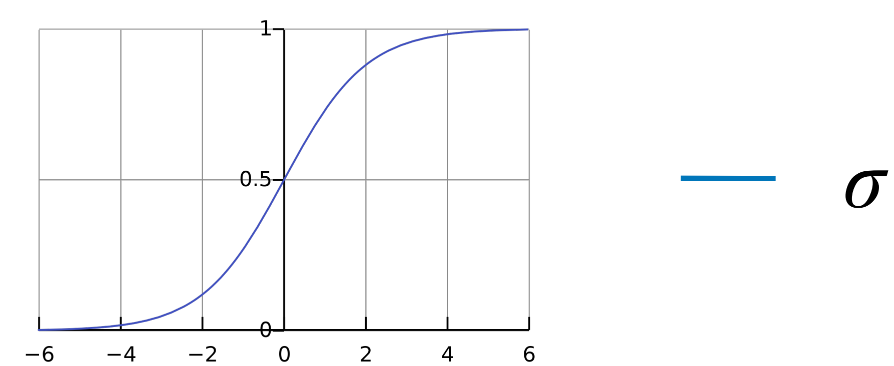
LSTMs in (more) detail

Forget: cell state vector multiplied element-wise by f_t (values in [0,1])



Forget gate activation vector

$$f_t = \sigma\,(\,W_f x_t + U_f\, h_{t\text{-}1}\,)$$

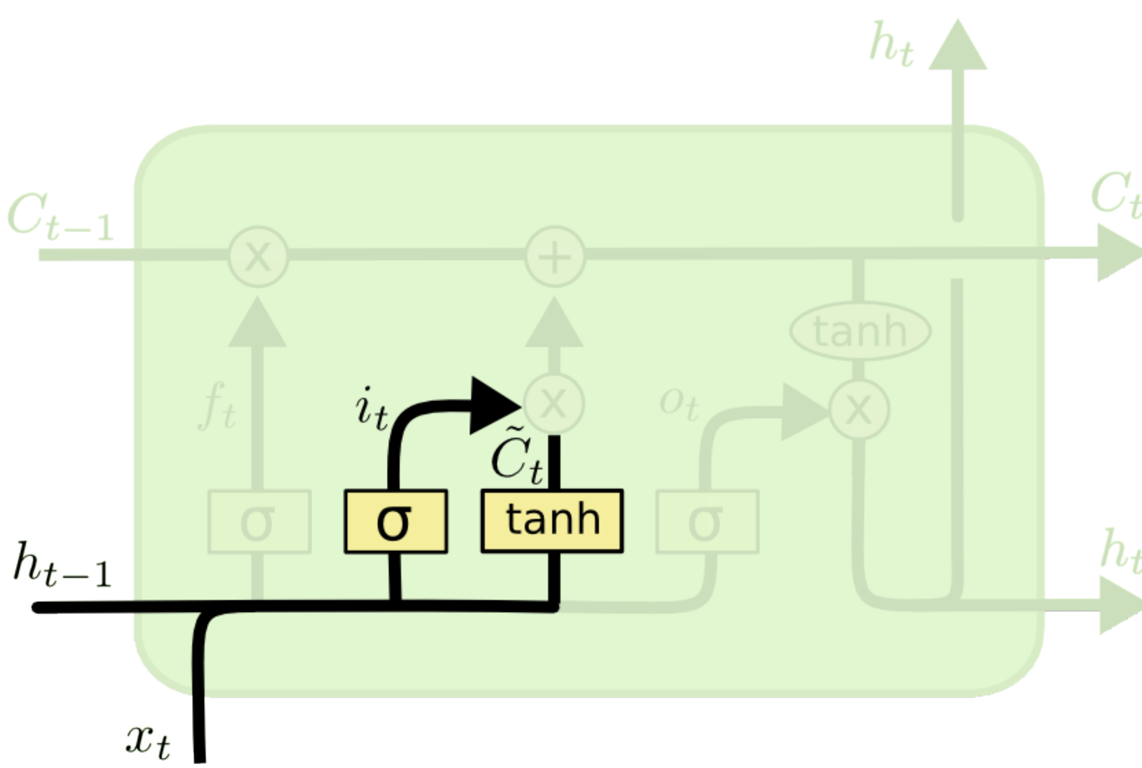


Hochreiter and Schmidhuber (1997) *Long short-term memory*

LSTM illustrations from Olah (2015) *Understanding LSTM Networks*

LSTMs in (more) detail

Input: candidate values $\tilde{C}_t$ scaled by i_t to add to cell state

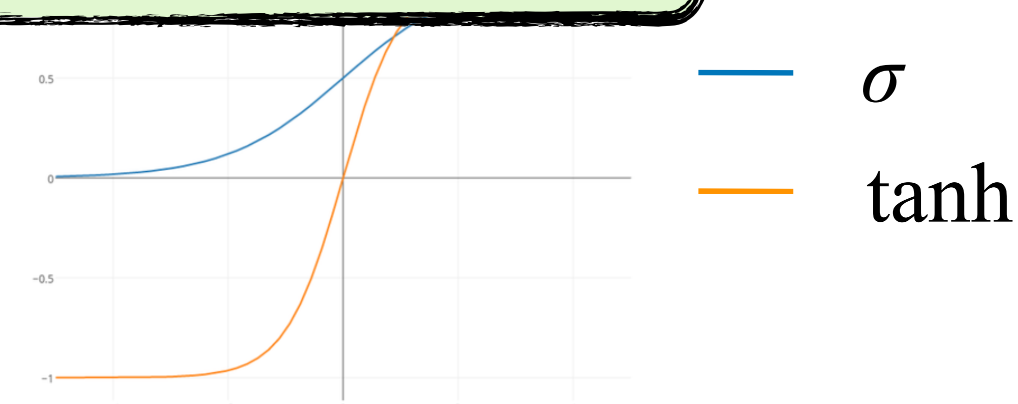


Input gate activation vector

$$i_t = \sigma\,(\,W_i\,x_t + U_i\,h_{t\text{-}1}\,)$$

$$\tilde{C}_t = \tanh(\,W_c\,x_t + U_c\,h_{t\text{-}1}\,)$$

Candidate input to cell state

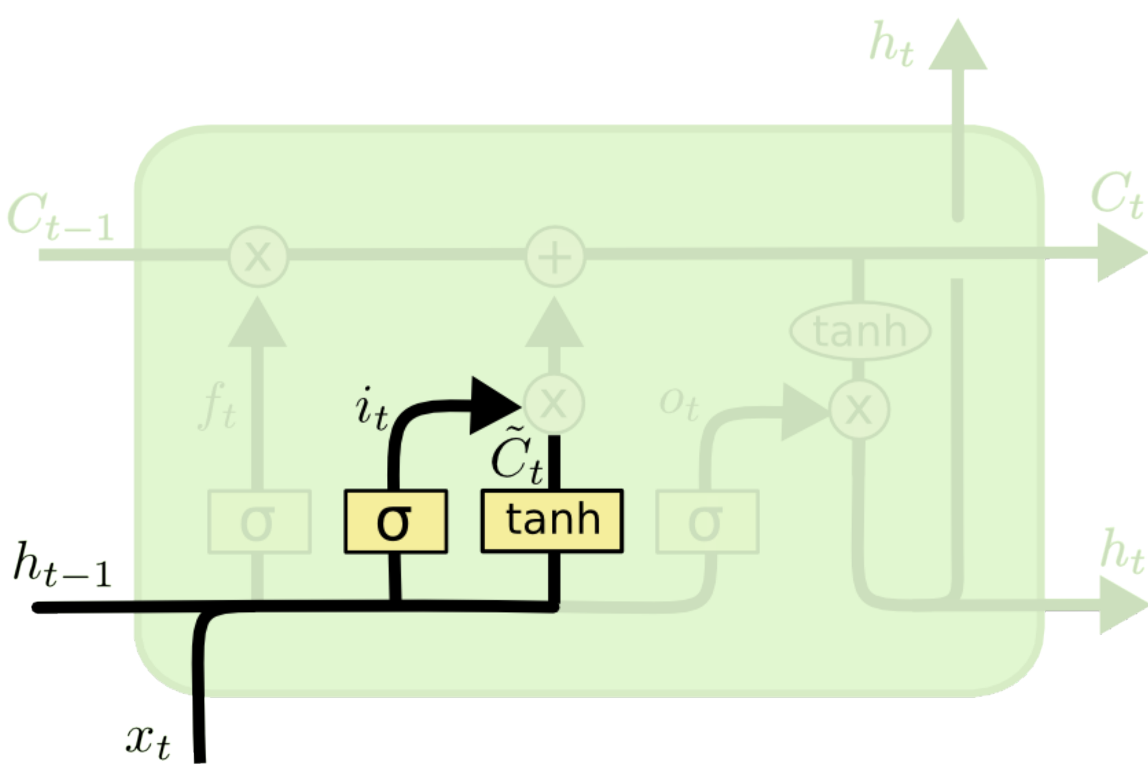


Hochreiter and Schmidhuber (1997) *Long short-term memory*

LSTM illustrations from Olah (2015) *Understanding LSTM Networks*

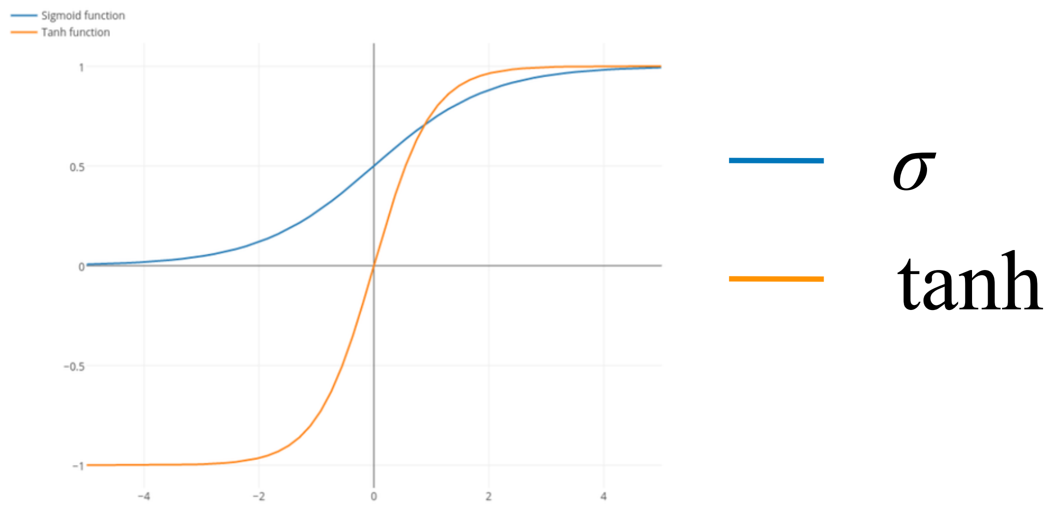
LSTMs in (more) detail

Input: candidate values $\tilde{C}_t$ scaled by i_t to add to cell state



$$i_t = \sigma\,(\ W_i\, x_t + U_i\ h_{t\text{-}1}\ )$$

$$\tilde{C}_t = \tanh(\ W_c\, x_t + U_c\ h_{t\text{-}1}\ )$$

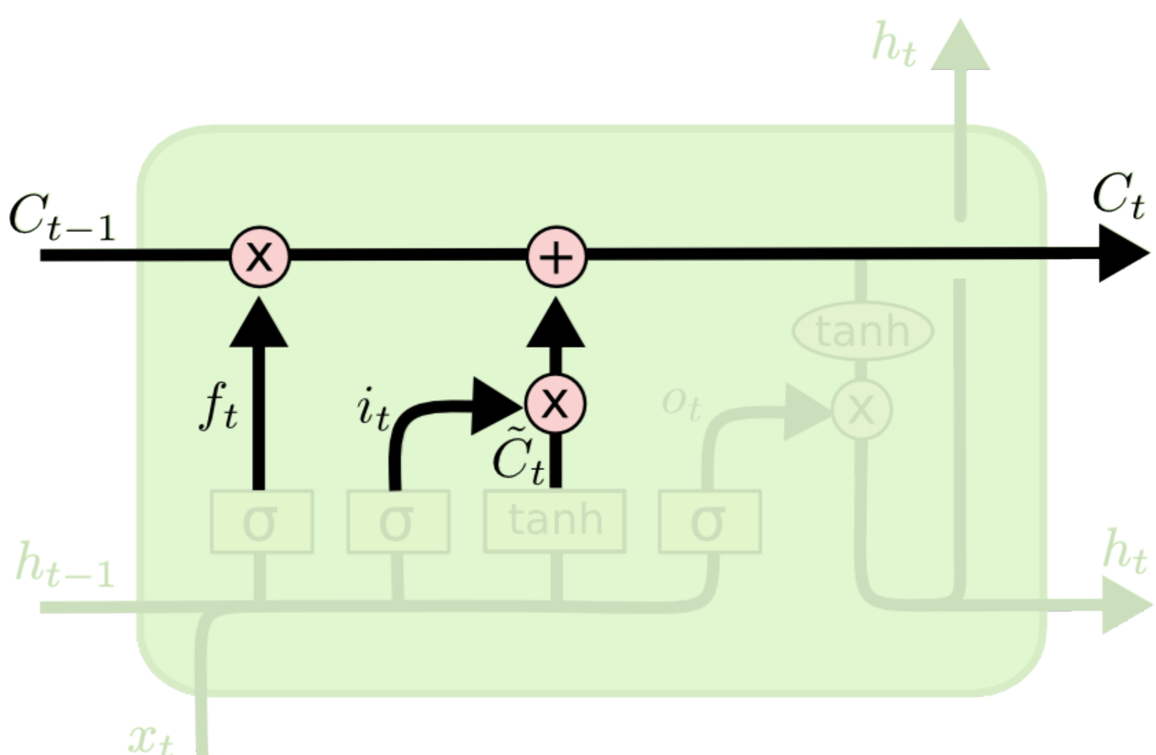


Hochreiter and Schmidhuber (1997) *Long short-term memory*

LSTM illustrations from Olah (2015) *Understanding LSTM Networks*

LSTMs in (more) detail

Update: previous state, forget, candidate and input form new state C_t



Element-wise products

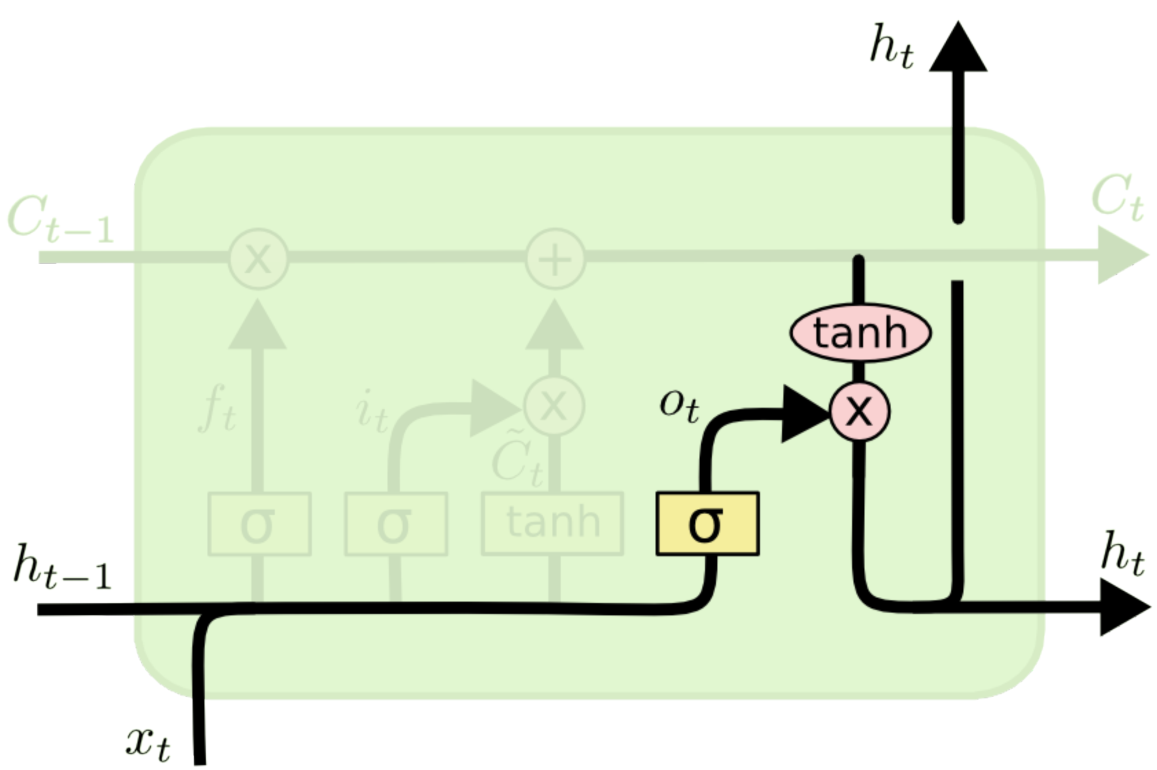
$$C_t = f_t * C_{t\text{-}1} + i_t * \tilde{C}_t$$

Hochreiter and Schmidhuber (1997) *Long short-term memory*

LSTM illustrations from Olah (2015) *Understanding LSTM Networks*

LSTMs in (more) detail

Output: new state C_t “filtered” element-wise by o_t (values in [0,1])



Output gate activation vector

$$o_t = \sigma\,(\,W_o\, x_t + U_o\, h_{t\text{-}1}\,)$$

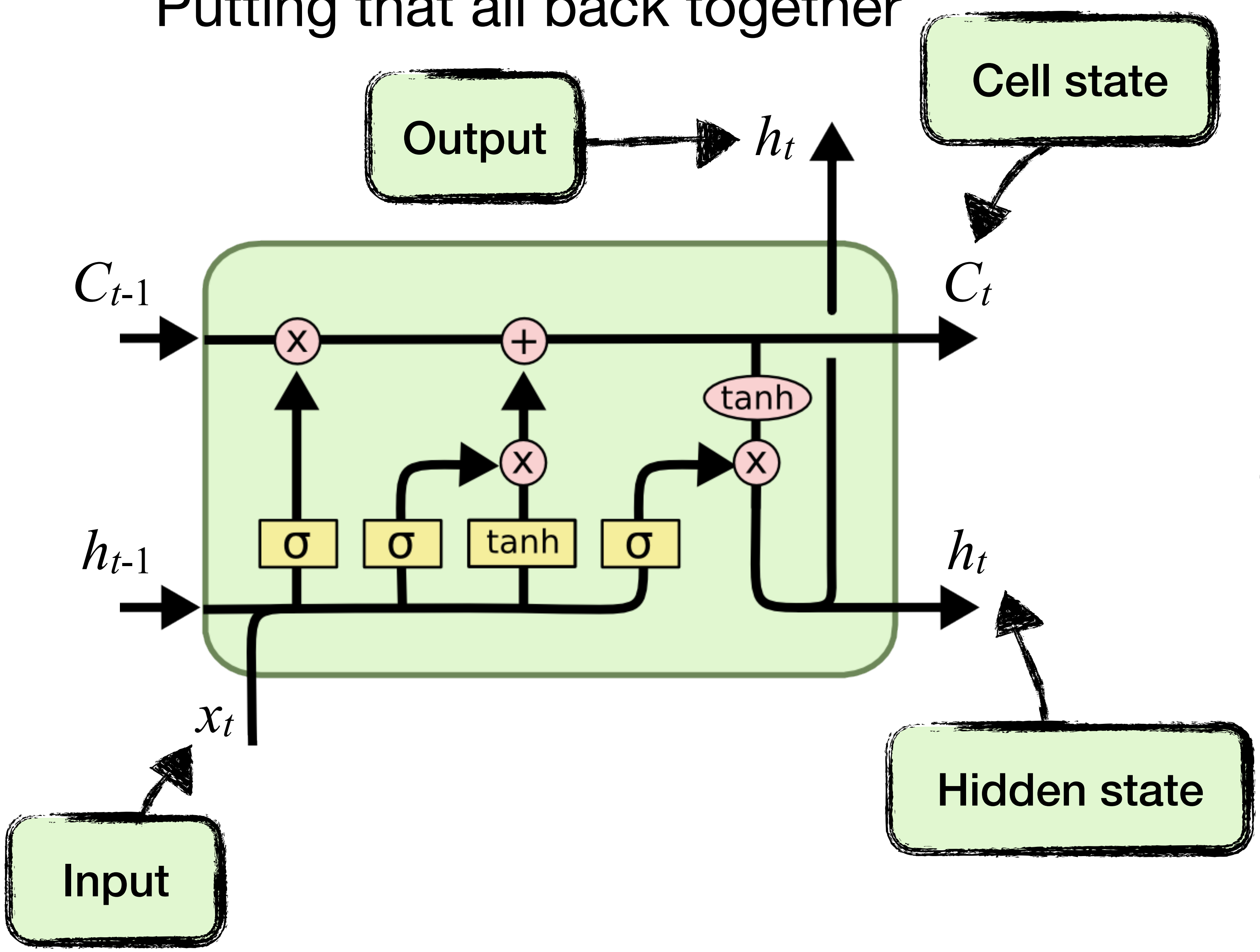
$$h_t = o_t * \tanh(C_t)$$

Hochreiter and Schmidhuber (1997) *Long short-term memory*

LSTM illustrations from Olah (2015) *Understanding LSTM Networks*

LSTMs in (more) detail

Putting that all back together



$$
\begin{aligned}
f_t &= \sigma\,(\ W_f x_t + U_f\, h_{t\text{-}1}\ ) \\
i_t &= \sigma\,(\ W_i\, x_t + U_i\ h_{t\text{-}1}\ ) \\
o_t &= \sigma\,(\ W_o\, x_t + U_o\ h_{t\text{-}1}\ ) \\
\tilde{C}_t &= \tanh(\ W_c\, x_t + U_c\ h_{t\text{-}1}\ ) \\
C_t &= f_t * C_{t\text{-}1} + i_t * \tilde{C}_t \\
h_t &= o_t * \tanh(C_t)
\end{aligned}
$$

Hochreiter and Schmidhuber (1997) *Long short-term memory*

LSTM illustrations from Olah (2015) *Understanding LSTM Networks*

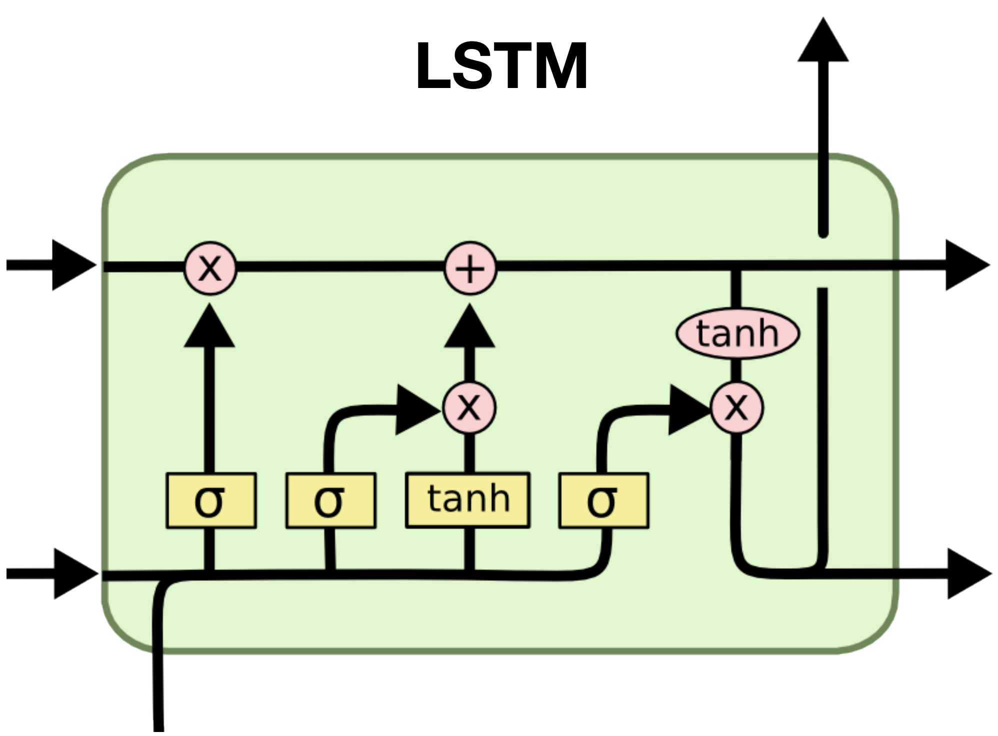
Gated recurrent units (GRUs)

Simpler variant of LSTM: fewer parameters, less computation

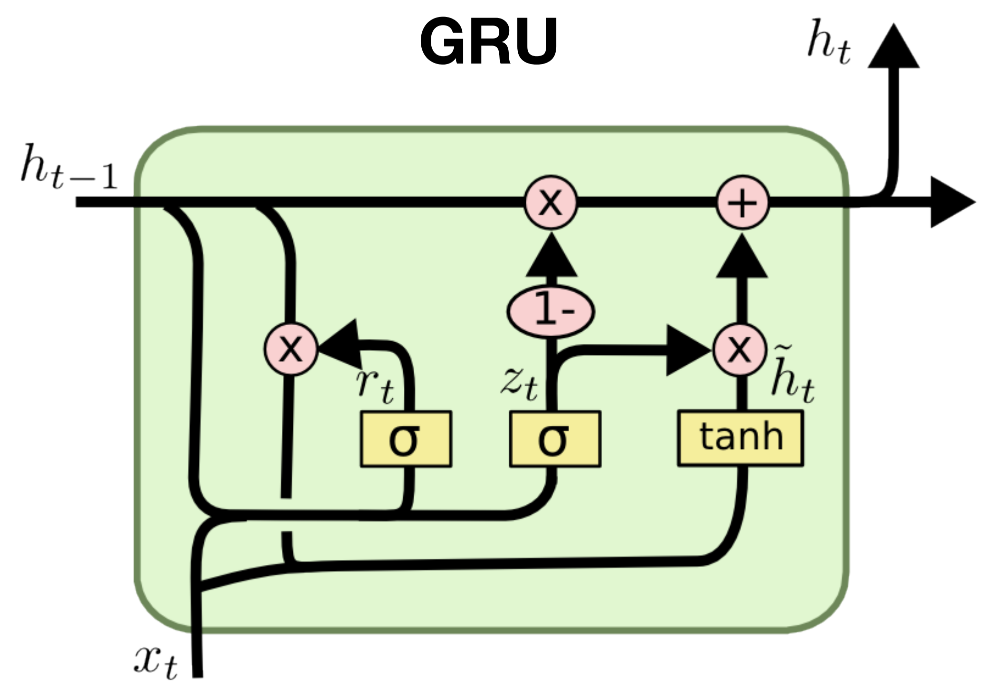
Drop-in replacement for vanilla RNN / LSTM cells

(Will not be covered in detail here)

LSTM



GRU



Cho *et al*. (2014) *Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation*

LSTM illustrations from Olah (2015) *Understanding LSTM Networks*

Resources

Recurrent Neural Networks | Stanford CS 231N
https://www.youtube.com/watch?v=6niqTuYFZLQ

Recurrent Neural Networks | MIT 6.S191
https://www.youtube.com/watch?v=SEnXr6v2ifU

Chris Olah: Understanding LSTM Networks
https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Michael Nguyen: Illustrated Guide to LSTM’s and GRU’s: A step by step explanation
https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21

Andrej Karpathy: The Unreasonable Effectiveness of Recurrent Neural Networks
http://karpathy.github.io/2015/05/21/rnn-effectiveness/