CS6421: Deep Neural Networks

Gregory Provan

Spring 2020

Lecture 20: Recurrent Neural Networks

Based on notes from John Canny, Fei-Fei Li, Justin Johnson, Serena Yeung

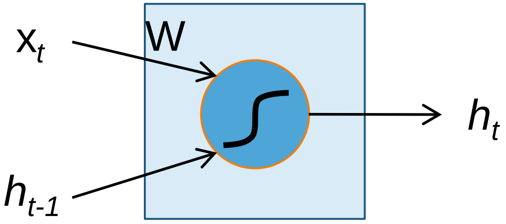
Outline

- Introduction
- Motivation
- RNN architecture
- RNN problems

LSTM - introduction

- LSTM was invented to solve the vanishing gradients problem.
- LSTM maintain a more constant error flow in the backpropogation process.
- LSTM can learn over more than 1000 time steps, and thus can handle large sequences that are linked remotely.

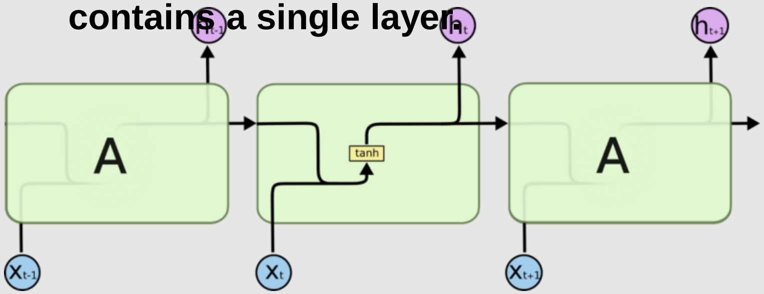
The Vanilla RNN Cell



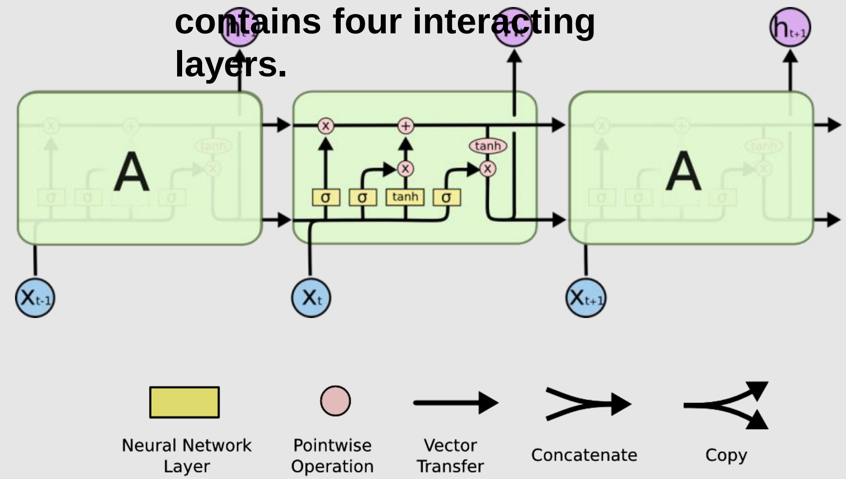
$$h_t = \tanh W \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix}$$

LSTM vs regular RNN

The repeating module in a standard RNN contains a single layer.

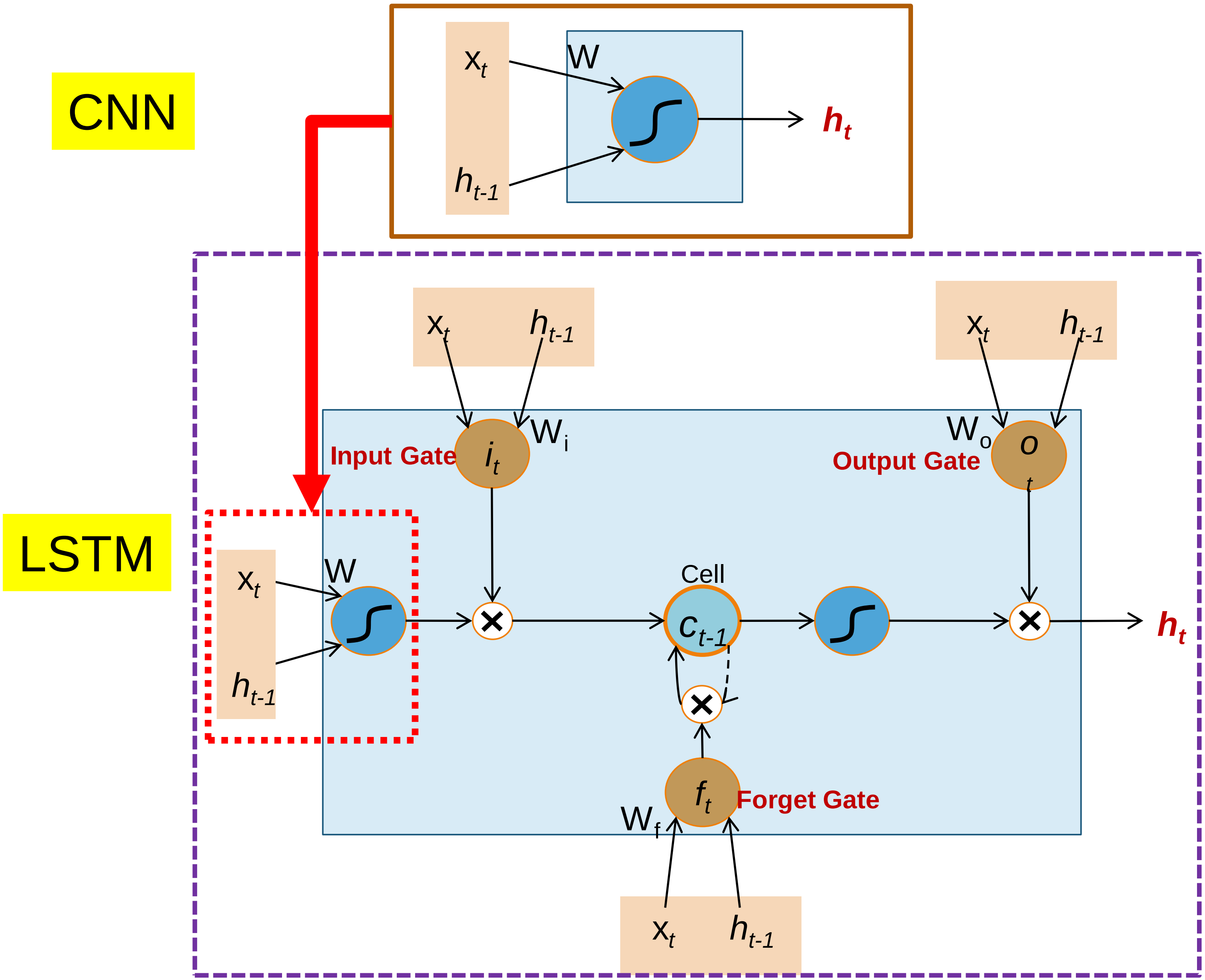


The repeating module in an LSTM contains four interacting layers.



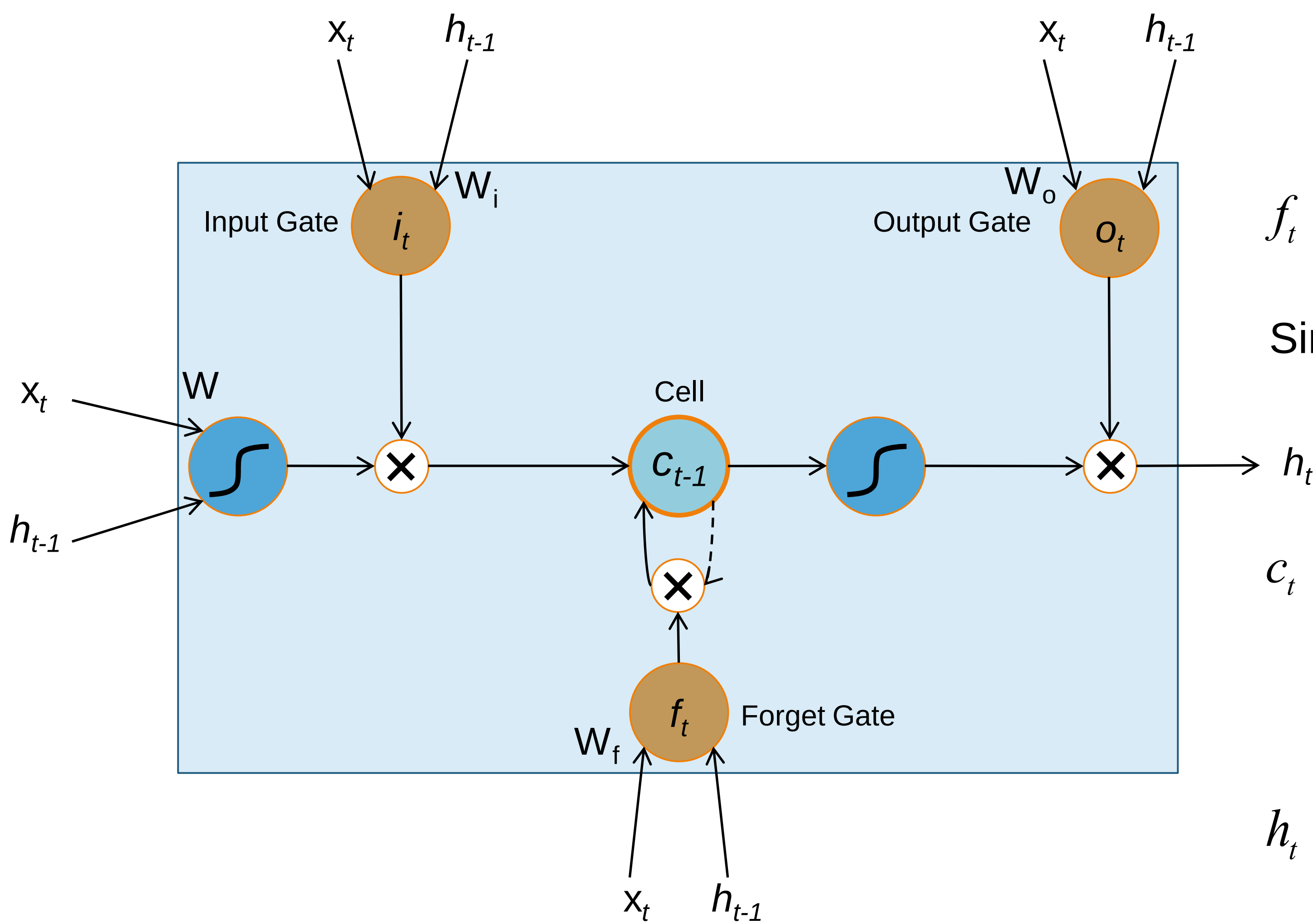
http://colah.github.io/posts/2015-08-Understanding-LSTMs/

CNN vs. LSTM



CNN
W
x_t
h_{t-1}
h_t
LSTM
W
x_t
h_{t-1}
Input Gate
i_t
W_i
Output Gate
W_o
o_t
Cell
c_{t-1}
h_t
Forget Gate
f_t
W_f

Details of LSTM Cell



$$f_t = \sigma\left(W_f \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix} + b_f\right)$$

Similarly for i_t, o_t

$$c_t = f_t \otimes c_{t-1} + i_t \otimes \tanh W \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix}$$

$$h_t = o_t \otimes \tanh c_t$$

* Dashed line indicates time-lag

LSTM - Long Short Term Memory

Vanilla RNN

$$h_t = \tanh\left(W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}\right)$$

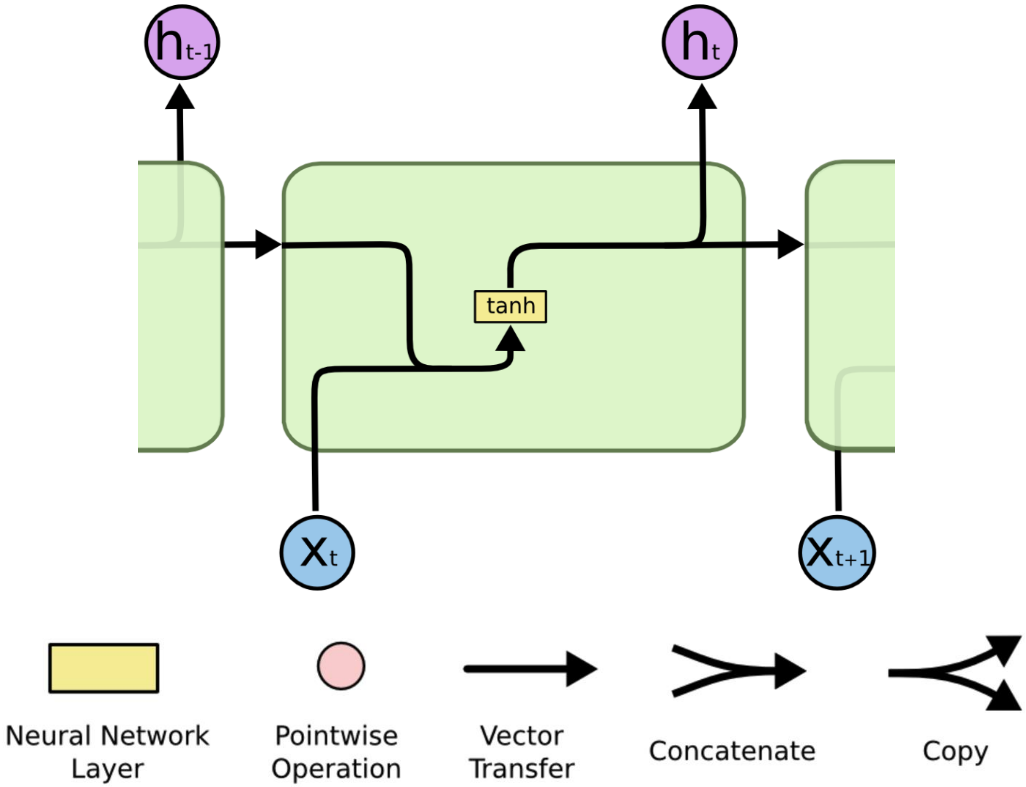
LSTM

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

$$c_t = f \odot c_{t-1} + i \odot g$$

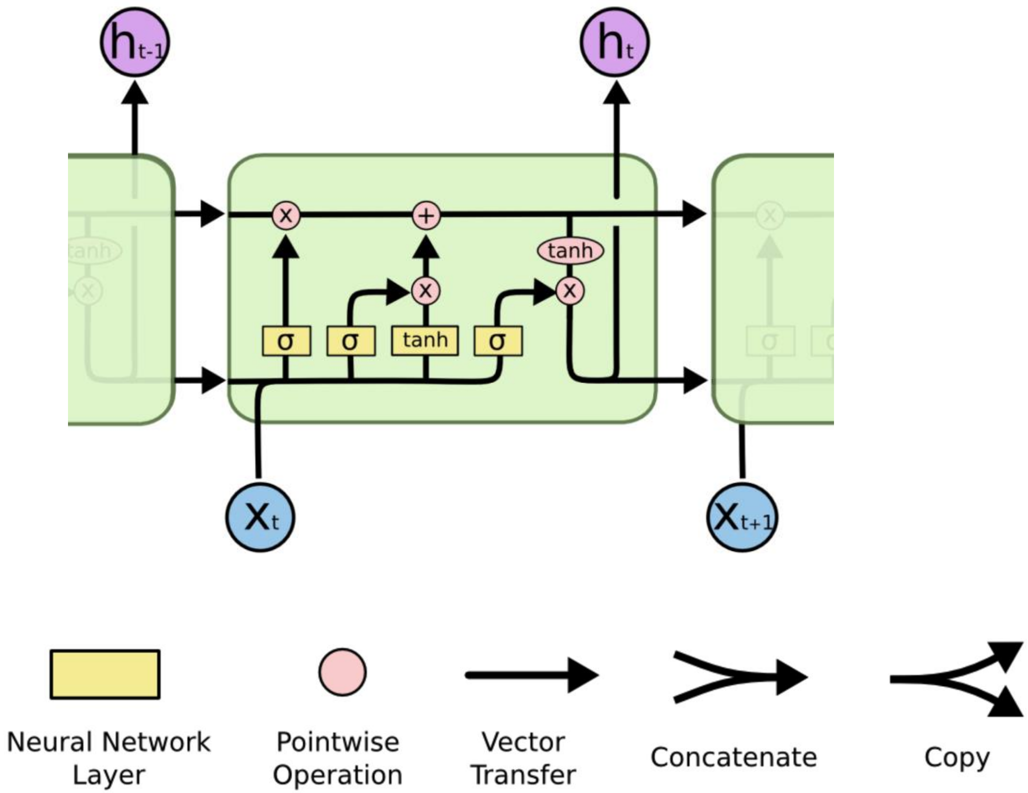
$$h_t = o \odot \tanh(c_t)$$

RNN Cell



Fei-Fei Li & Justin Johnson & Serena Yeung, Stanford 2017

LSTM - Long Short Term Memory



f: Forget gate | Whether to erase cell
i: Input gate | Whether to write to cell
o: Output gate | How much to reveal cell
g: Candidate value for the next cell state

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

$$c_t = f \odot c_{t-1} + i \odot g$$

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

Long Short Term Memory (LSTM)

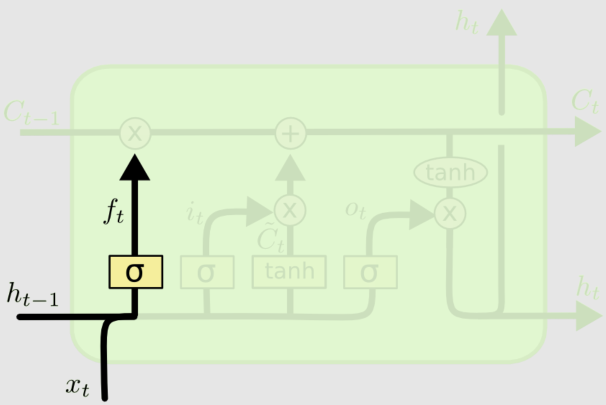
- Long Short Term Memory networks (LSTM) are designed to counter the vanishing gradient problem.
- They introduce the "cell state" (c_t) parameter which allows for almost uninterrupted gradient flow through the network.
- The LSTM module is composed of four gates (layers) that interact with one another:
  - Input gate
  - Forget gate
  - Output gate
  - $tanh$ gate

[Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory]

Forget Gate

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

- The forget gate layer f_t decides what information to discard from the **previous** state c_{t-1} w.r.t. the **current** input (h_{t-1}, x_t).
- It scales input with a sigmoid function.
- When $f_t = 0$ we "forget" the previous state.

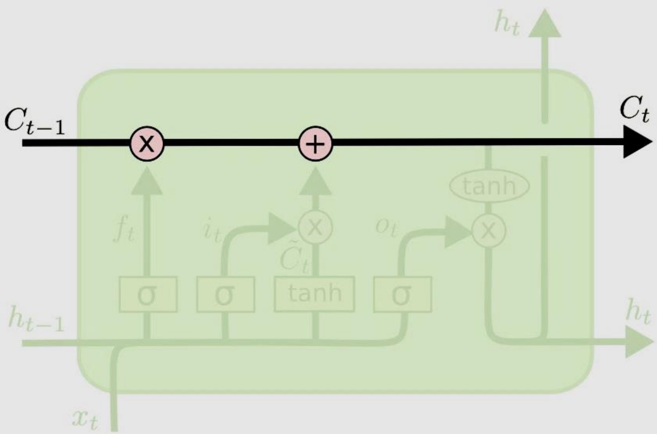


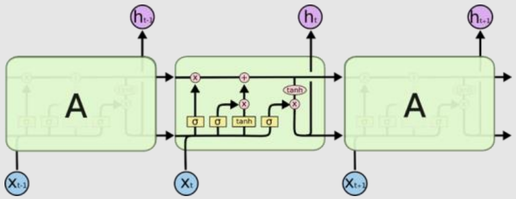
$$f_t = \sigma(w_f[h_{t-1}, x_t] + b)$$

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

LSTM: Cell State

- The Cell state c_t holds information about previous state – *memory* cell.
- $c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t$





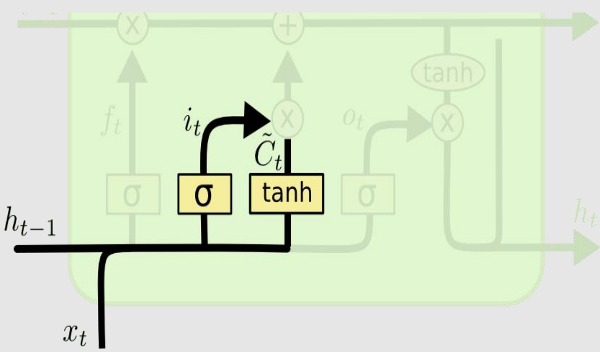
http://colah.github.io/posts/2015-08-Understanding-LSTMs/

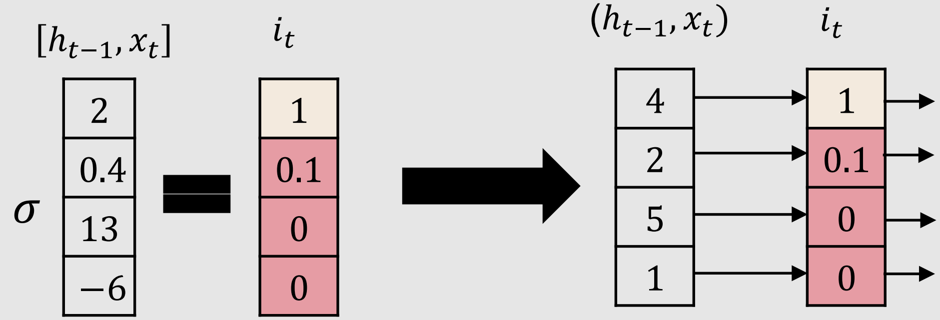
Input gate

$$c_t^* = c_{t-1} + i_t \odot tanh$$

- The input gate layer i_t decides what information should go to the **current** state c_t w.r.t. the **current input** (h_{t-1}, x_t).
- When $i_t = 0$ we ignore the current time step.

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] + b_i\right)$$



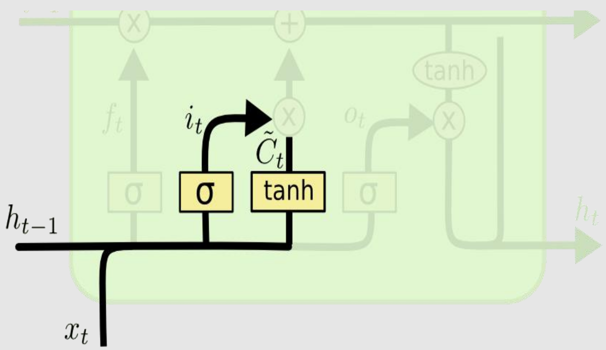


http://colah.github.io/posts/2015-08-Understanding-LSTMs/

(tanh) gate

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

- Creates the current state $\tilde{c}_t$.



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$

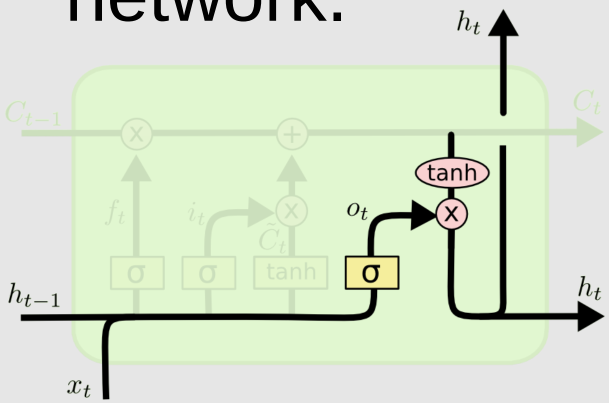
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

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

Output Gate

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t$$
$$h_t = o_t \odot \tanh(c_t)$$

- The output gate layer o_t filters the cell state $\tilde{c}_t$.
- It decides what information goes “out” of the cell and what remains hidden from the rest of the network.



$$o_t = \sigma\left(W_o\ [h_{t-1}, x_t]\ +\ b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

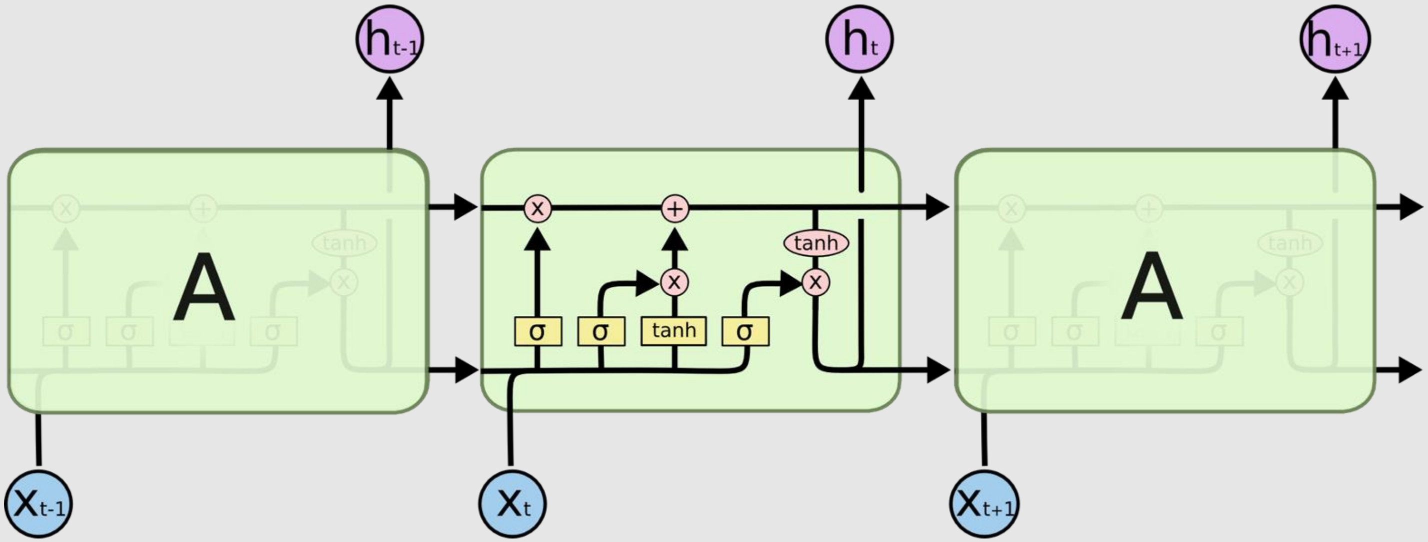
http://colah.github.io/posts/2015-08-Understanding-LSTMs/

LSTM: Summary of Equations

- $f_t = \sigma(w_f[h_{t-1}, x_t] + b_f)$
- $i_t = \sigma(w_i[h_{t-1}, x_t] + b_i)$
- $o_t = \sigma(w_o[h_{t-1}, x_t] + b_o)$
- $\tilde{c}_t = \tanh(Wc[h_{t-1}, x_t] + b_c)$

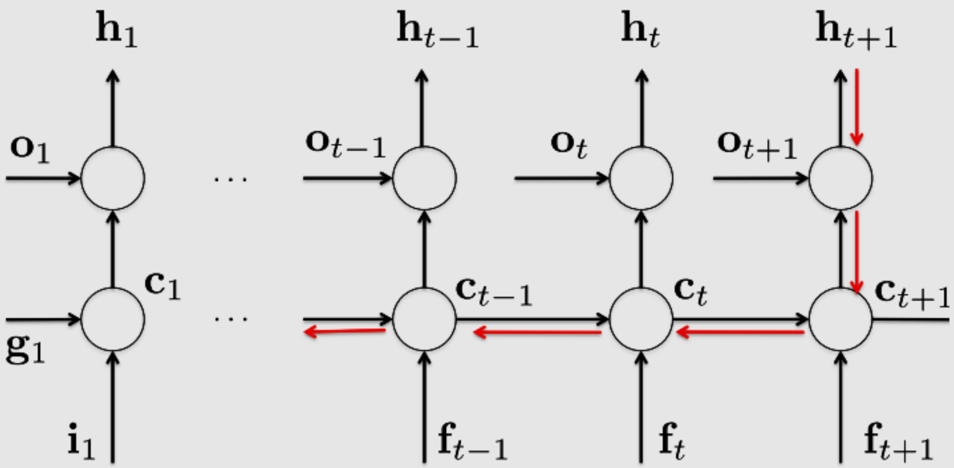
- $c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t$
- $h_t = o_t \odot \tanh(c_t)$

Where h_t is the next hidden state.



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

LSTM – Gradient Flow

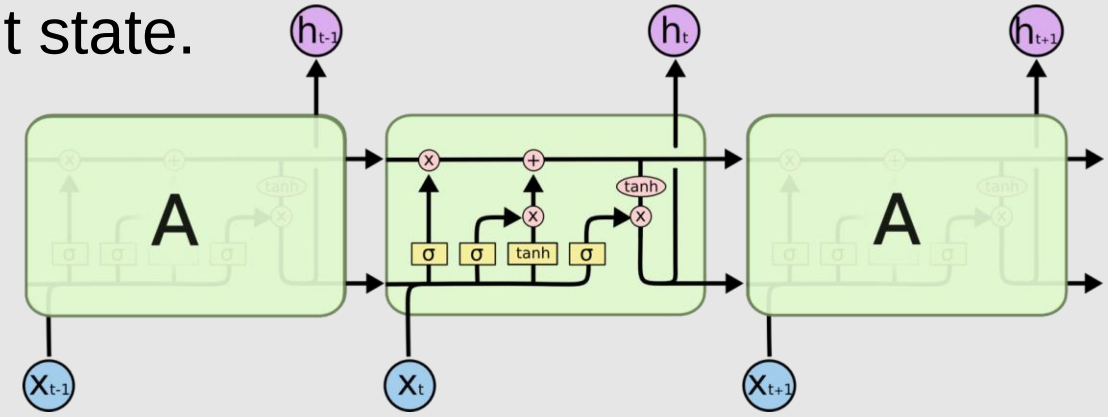


$$\frac{\partial loss_t}{\partial C_t} = \frac{\partial loss_t}{\partial y_t}\frac{\partial y_t}{\partial h_t}\frac{\partial h_t}{\partial c_t}\prod_{j=2}^{t}\frac{\partial c_j}{\partial c_{j-1}}$$

Chen, G. (2016). A Gentle Tutorial of Recurrent Neural Network with Error Backpropagation

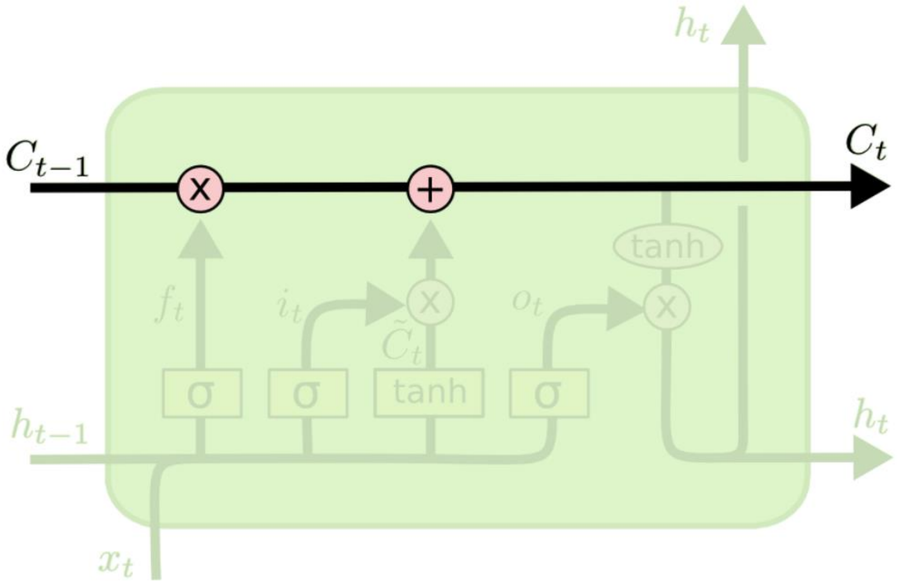
LSTM Summary

- LSTM solves the problem of vanishing gradient by introducing the memory cells - $c_t = f_t * c_{t-1} + i_t * \widetilde{c_t}$ which is mostly defined by addition and element-wise multiplication operators.
- The gates system filters what information we keep from the previous states and what information to add from the current state.



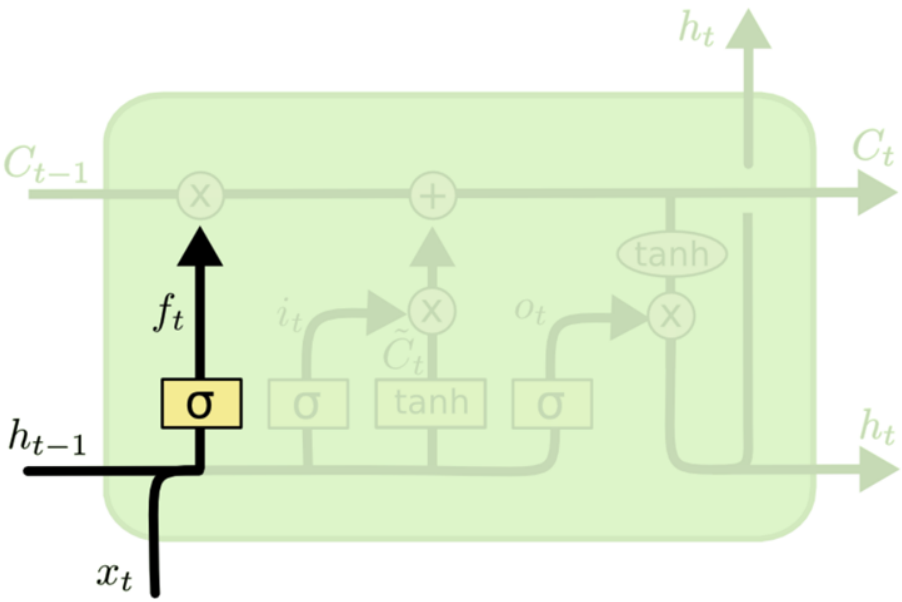
The CELL STATE

It’s very easy for information to just flow along it unchanged



The Forget Gate

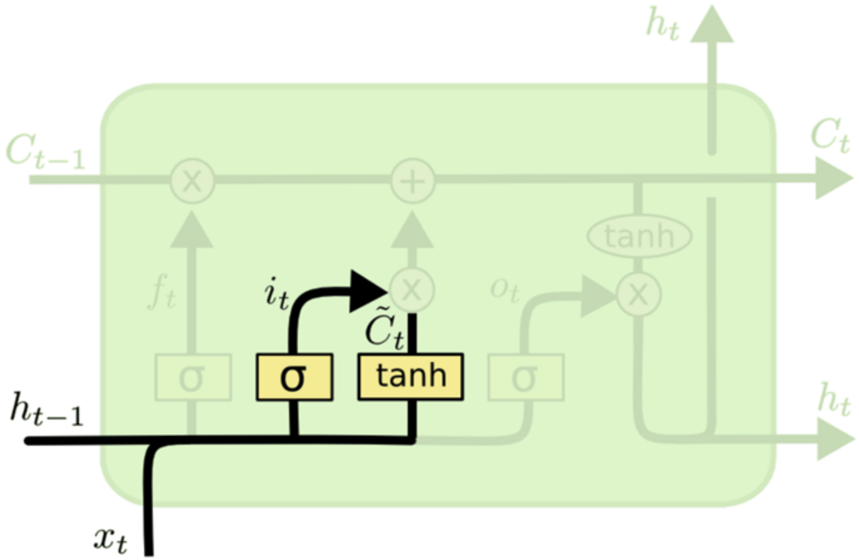
The first step in our LSTM is to decide what information we're going to throw away from the cell state



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \ + \ b_f\right)$$

Input Gate

The next step is to decide what new information we're going to store in the cell state.

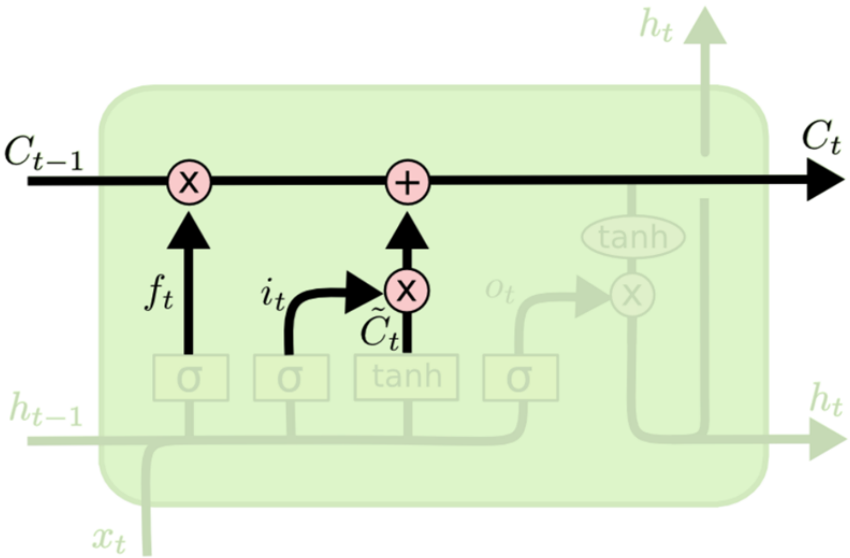


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

Update The Cell State

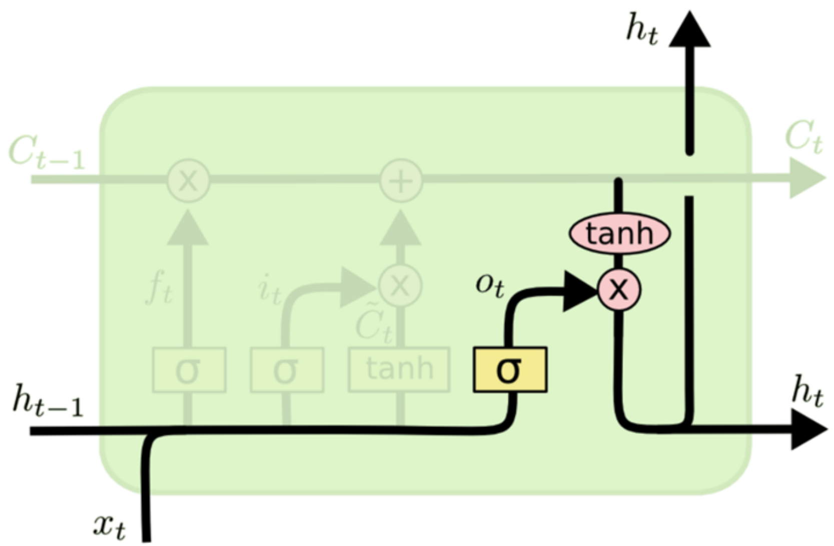
From the **Forget** and **Input** gate, along with the **candidate** for the next cell state, and the **previous** cell state, we compute the next cell state



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

Output Gate

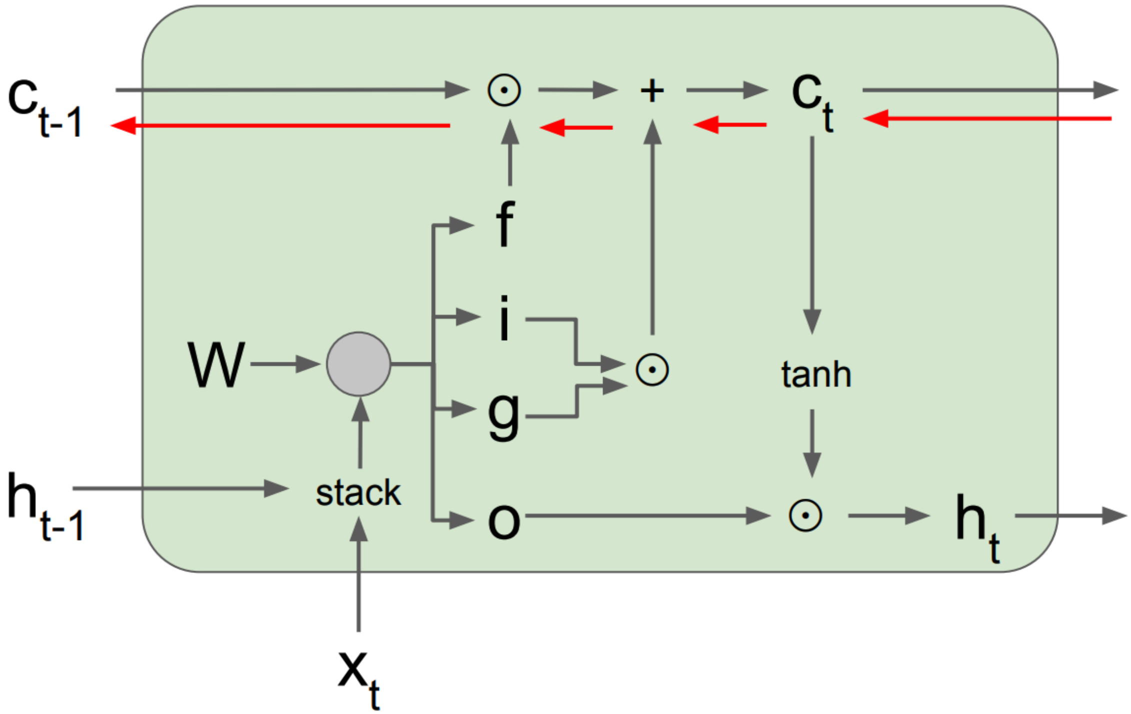
Finally, we compute what we're going to output. This output is a filtered version of our cell state



$$o_t = \sigma\left(W_o\ [h_{t-1}, x_t]\ +\ b_o\right)$$

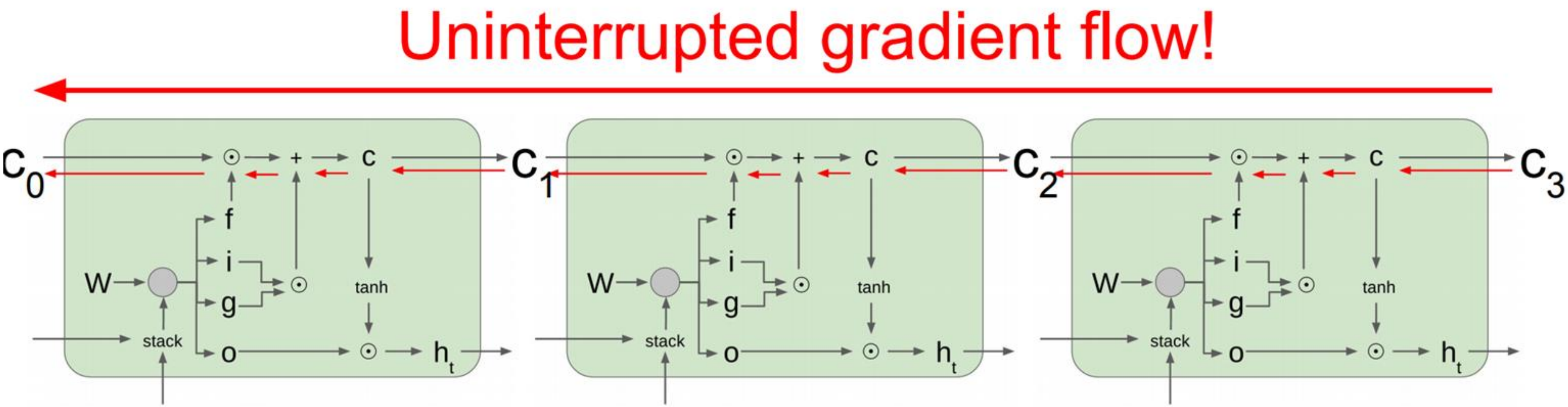
$$h_t = o_t * \tanh\left(C_t\right)$$

LSTM Gradient Flow

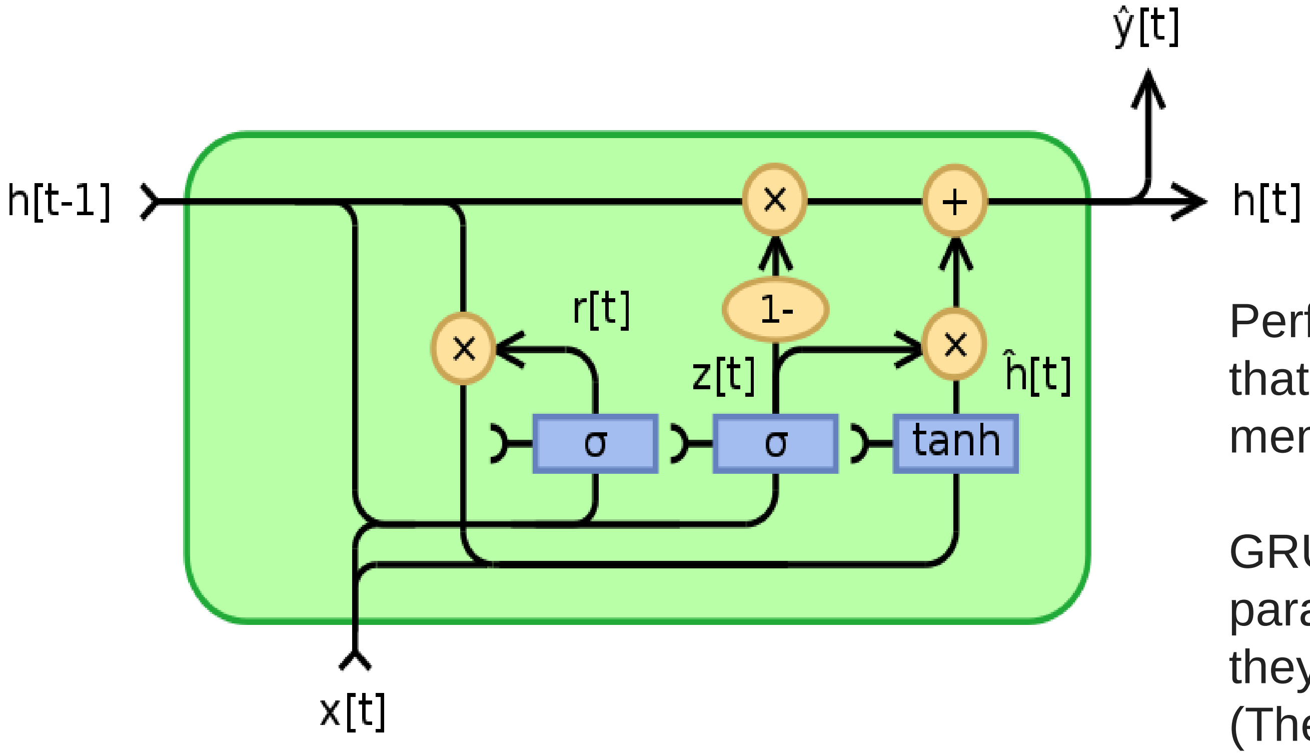


Backpropagation from c_t to c_{t-1} only elementwise multiplication by f, no matrix multiply by W

LSTM Gradient Flow



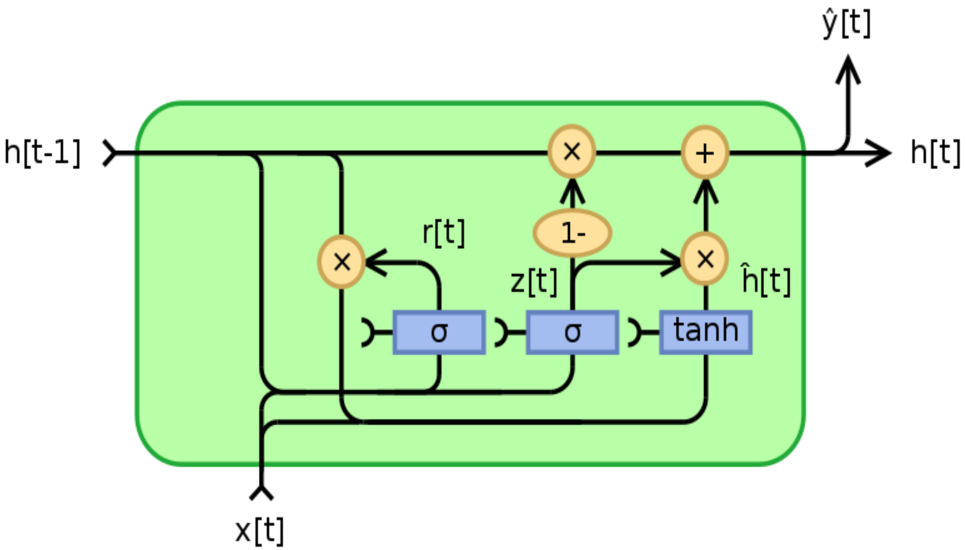
GRU - Gated Recurrent Unit



Performance is similar to that of long short-term memory.

GRUs have fewer parameters than LSTM, as they lack an output gate. (They use 1-z instead)

GRU - Gated Recurrent Unit



x_t: input vector

h_t: output vector

z_t: update gate vector

r_t: reset gate vector

W, U and b: parameter matrices and vector

$$z_t = \sigma_g(W_z x_t + U_z h_{t-1} + b_z)$$
$$r_t = \sigma_g(W_r x_t + U_r h_{t-1} + b_r)$$

$$h_t = (1 - z_t) \circ h_{t-1} + z_t \circ \sigma_h(W_h x_t + U_h(r_t \circ h_{t-1}) + b_h)$$

Advantages of LSTM

- Non-decaying error backpropagation.
- For long time lag problems, LSTM can handle noise and continuous values.
- No parameter fine tuning.
- Memory for long time periods

LSTM conclusions

- RNNs - self connected networks
- Vanishing gradients and long memory problems
- LSTM - solves the vanishing gradient and the long memory limitation problem
- LSTM can learn sequences with more than 1000 time steps.