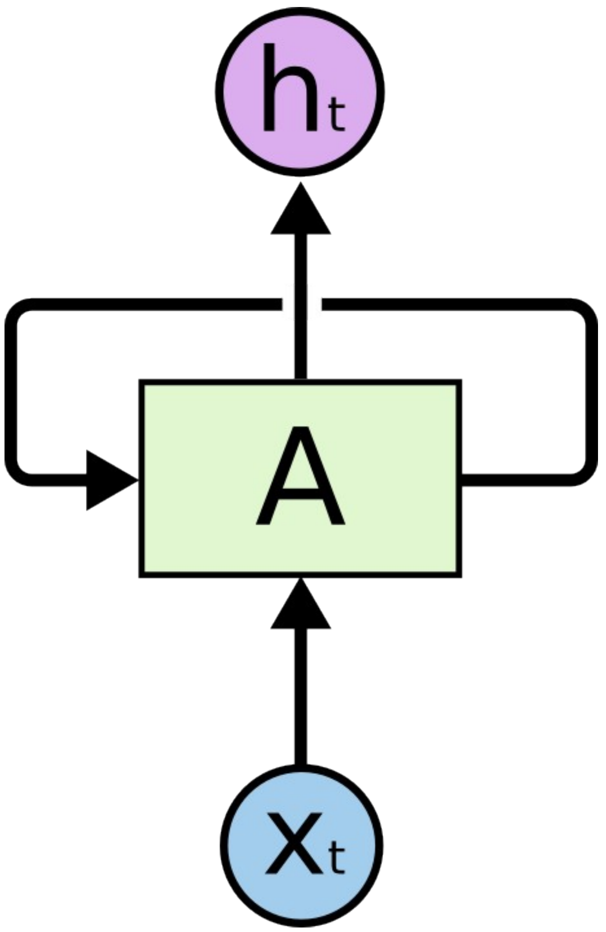
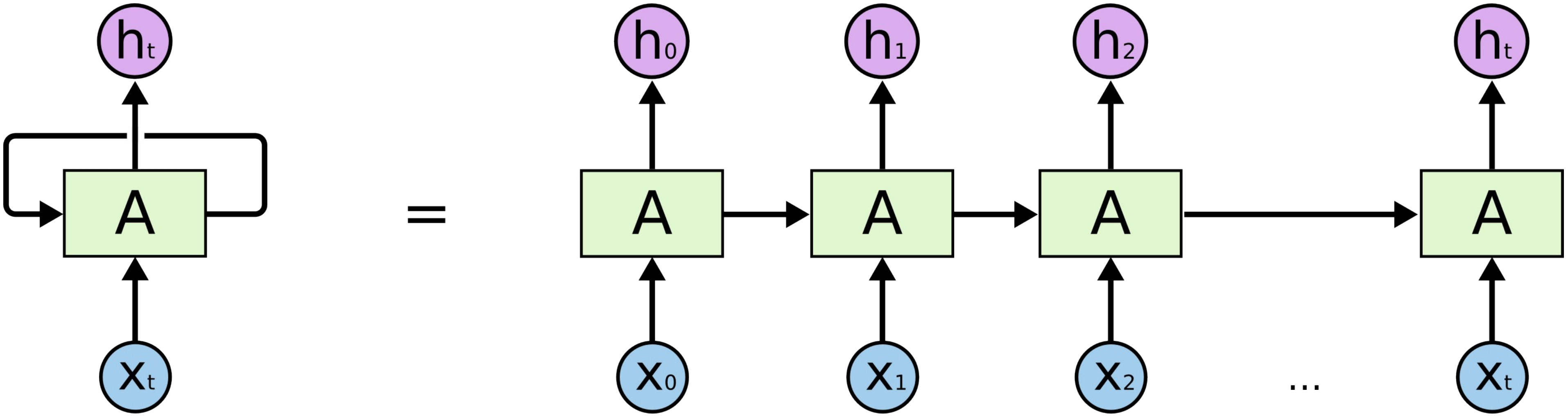
Understanding LSTM Networks

Recurrent Neural Networks



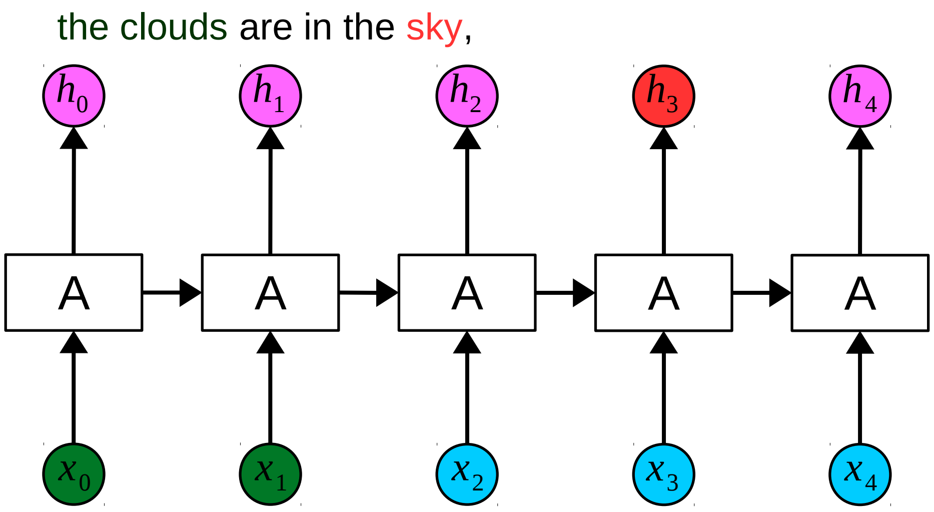
An unrolled recurrent neural network



The Problem of Long-Term Dependencies

RNN short-term dependencies

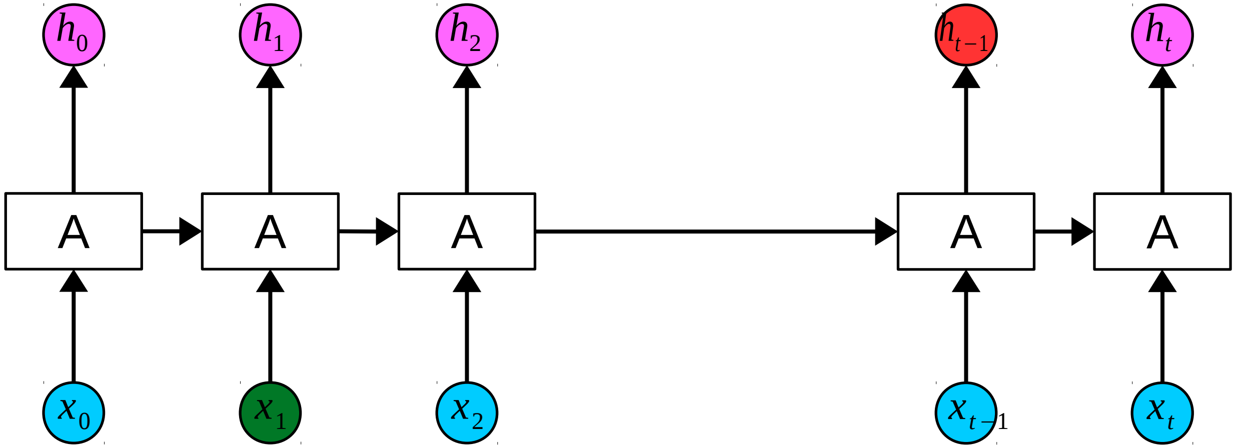
Language model trying to predict the next word based on the previous ones



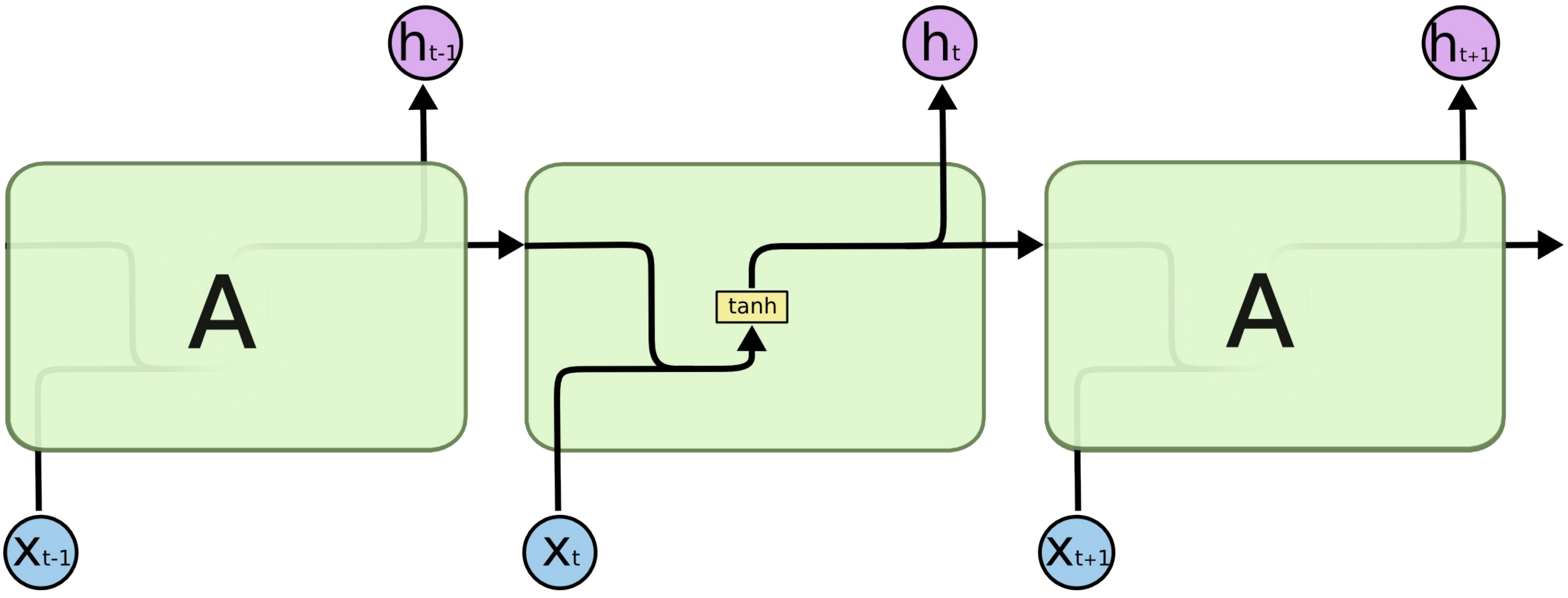
RNN long-term dependencies

Language model trying to predict the next word based on the previous ones

I grew up in India… I speak fluent Hindi.



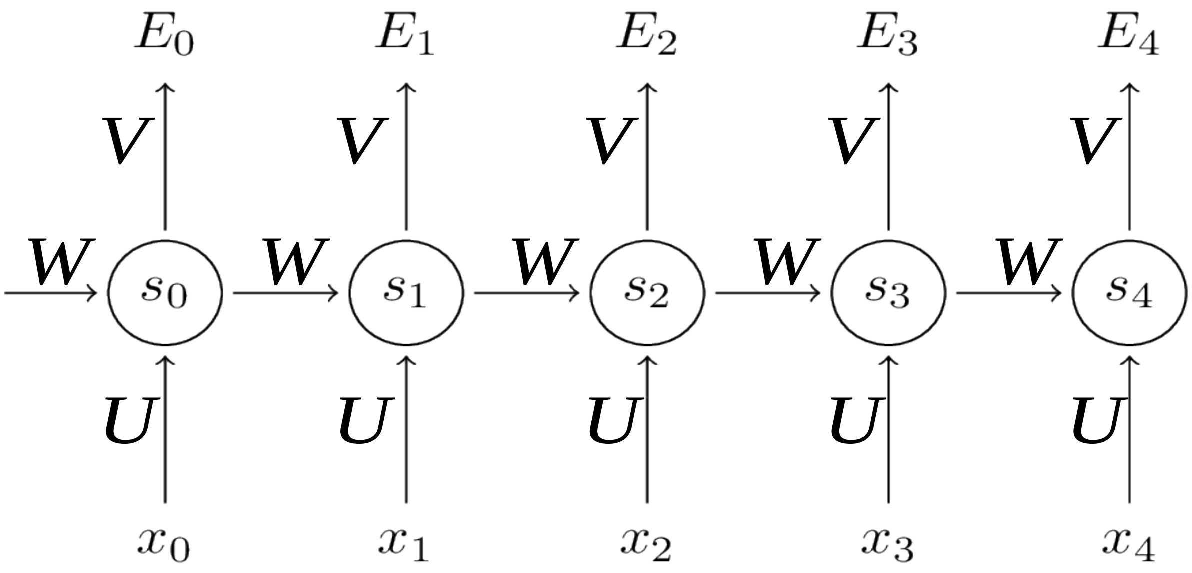
Standard RNN



Backpropagation Through Time (BPTT)

RNN forward pass

$$s_t = \tanh(Ux_t + Ws_{t-1})$$
$$\hat{y}_t = softmax(Vs_t)$$



$$E(y, \hat{y}) = -\sum_t E_t(y_t, \hat{y}_t)$$

Backpropagation Through Time

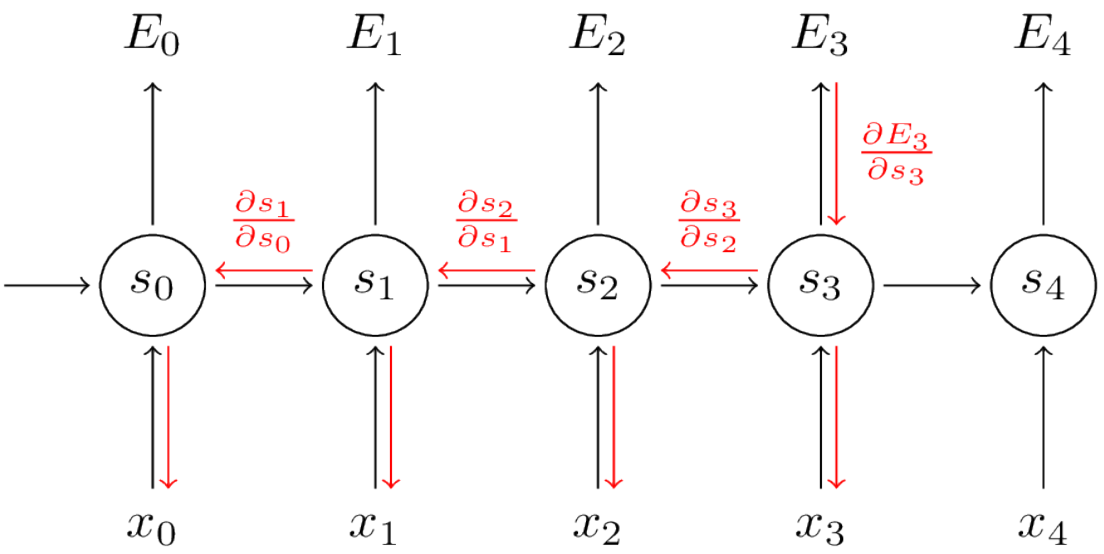
$$\frac{\partial E}{\partial \mathbf{W}} = \sum_t \frac{\partial E_t}{\partial \mathbf{W}}$$

$$\frac{\partial E_3}{\partial \mathbf{W}} = \frac{\partial E_3}{\partial \hat{y}_3} \frac{\partial \hat{y}_3}{\partial s_3} \frac{\partial s_3}{\partial \mathbf{W}}$$

But $s_3 = \tanh(Ux_t + Ws_2)$

S_3 depends on s_2, which depends on W and s_1, and so on.

$$\frac{\partial E_3}{\partial \mathbf{W}} = \sum_{k=0}^{3} \frac{\partial E_3}{\partial \hat{y}_3} \frac{\partial \hat{y}_3}{\partial s_3} \frac{\partial s_3}{\partial s_k} \frac{\partial s_k}{\partial \mathbf{W}}$$



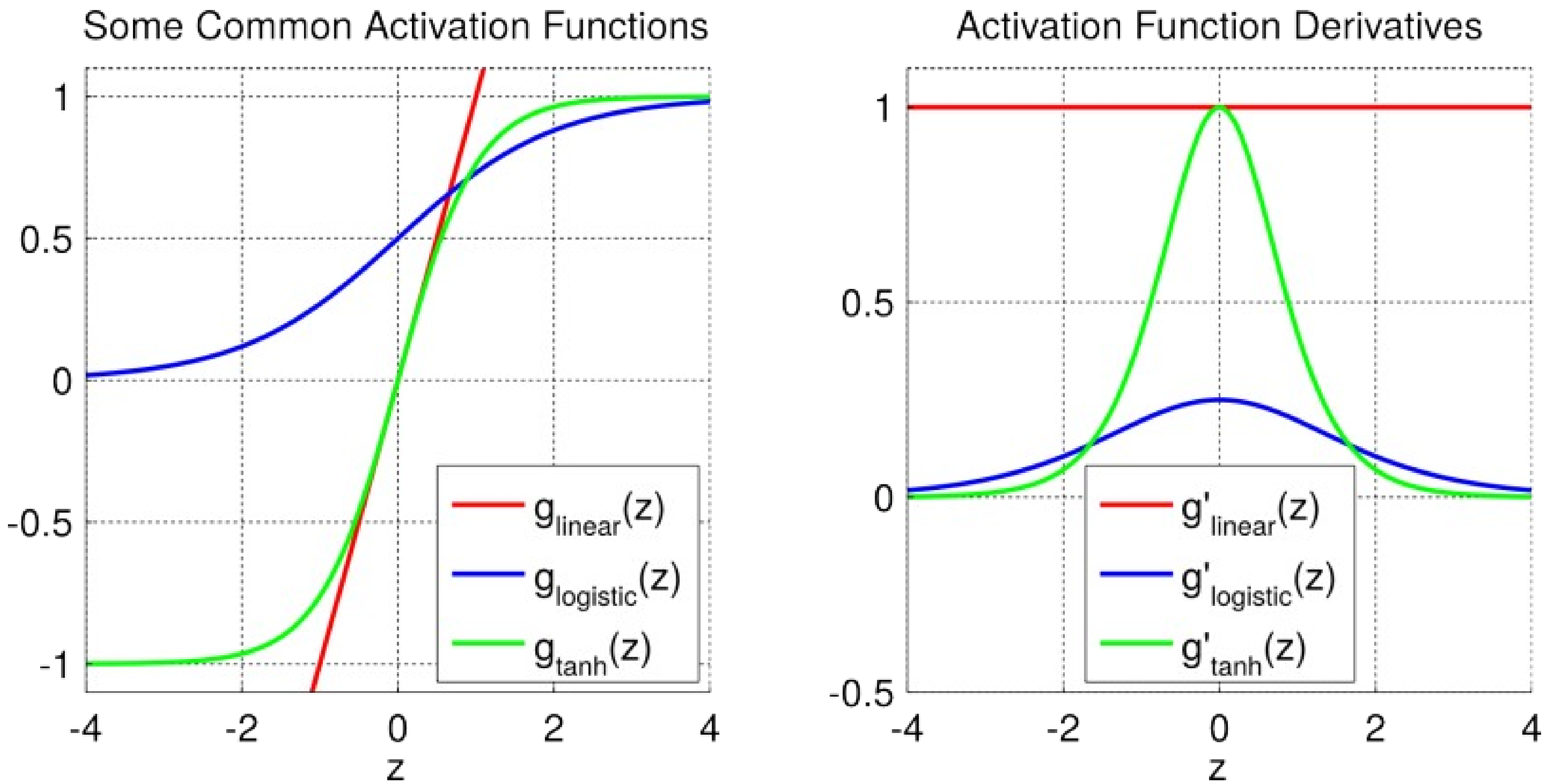
The Vanishing Gradient Problem

$$\frac{\partial E_3}{\partial W} = \sum_{k=0}^{3} \frac{\partial E_3}{\partial \hat{y}_3} \frac{\partial \hat{y}_3}{\partial s_3} \frac{\partial s_3}{\partial s_k} \frac{\partial s_k}{\partial W}$$

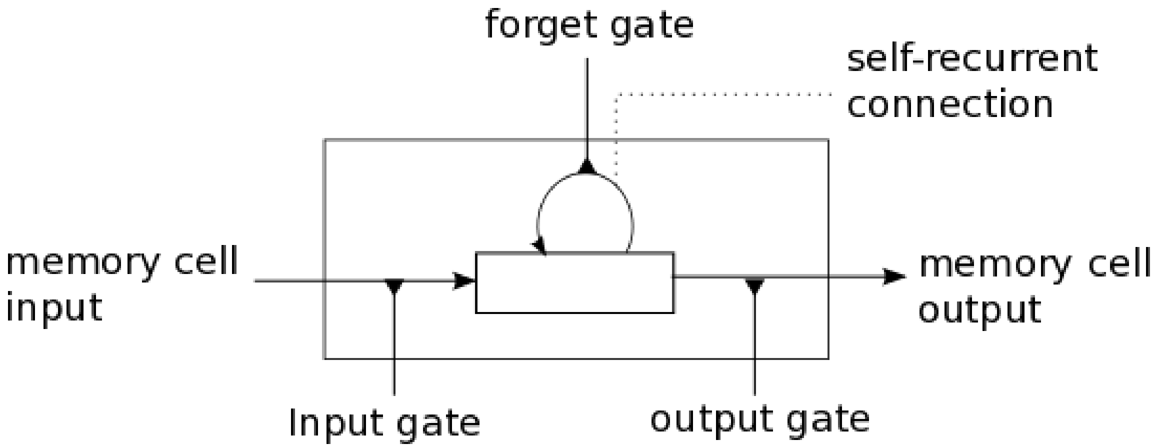
$$\frac{\partial E_3}{\partial W} = \sum_{k=0}^{3} \frac{\partial E_3}{\partial \hat{y}_3} \frac{\partial \hat{y}_3}{\partial s_3} \left(\prod_{j=k+1}^{3} \frac{\partial s_j}{\partial s_{j-1}} \right) \frac{\partial s_k}{\partial W}$$

- Derivative of a vector w.r.t a vector is a matrix called jacobian
- 2-norm of the above Jacobian matrix has an upper bound of 1
- **tanh** maps all values into a range between -1 and 1, and the **derivative is bounded by 1**
- With multiple matrix multiplications, gradient values **shrink exponentially**
- Gradient contributions from “far away” steps become zero
- Depending on activation functions and network parameters, gradients could **explode** instead of vanishing

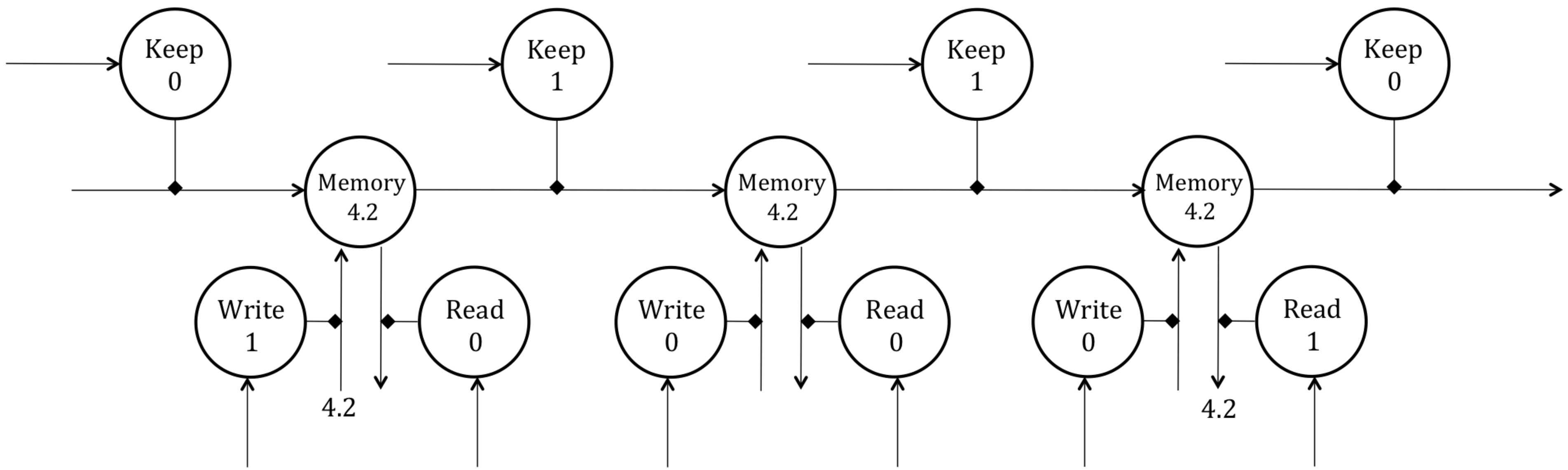
Activation function



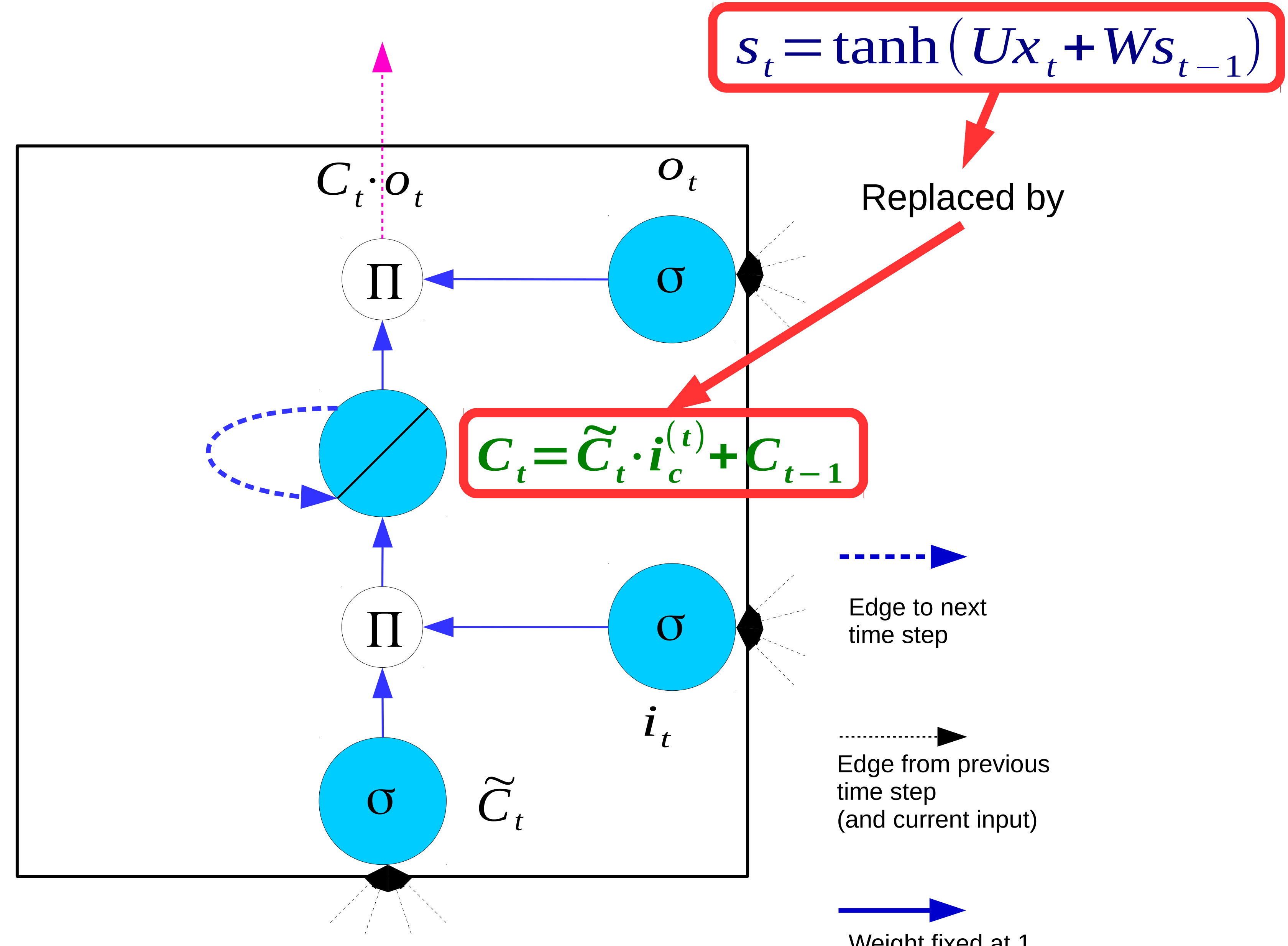
Basic LSTM


forget gate
self-recurrent connection
memory cell input
memory cell output
Input gate
output gate

Unrolling the LSTM through time

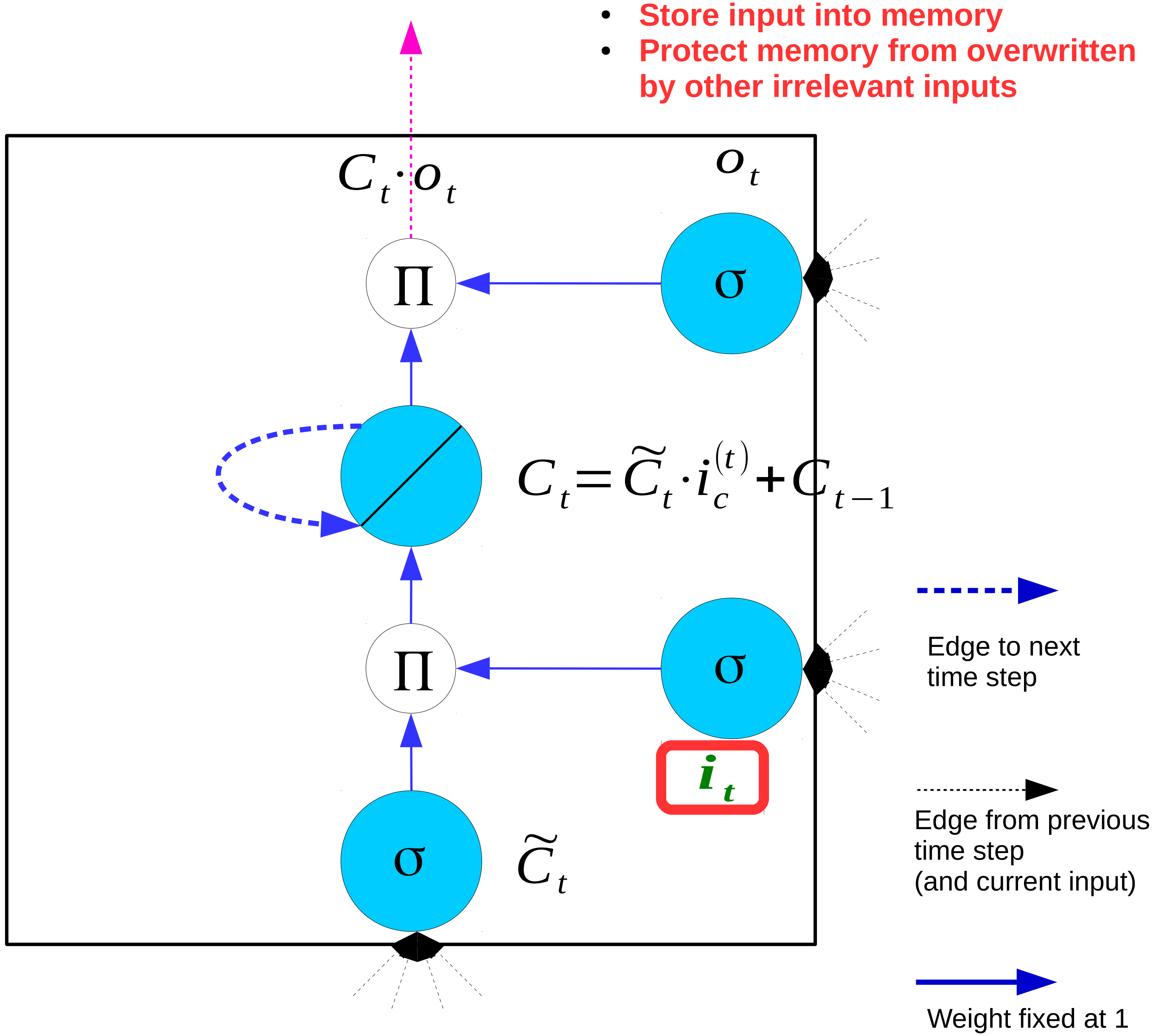


Constant error carousel



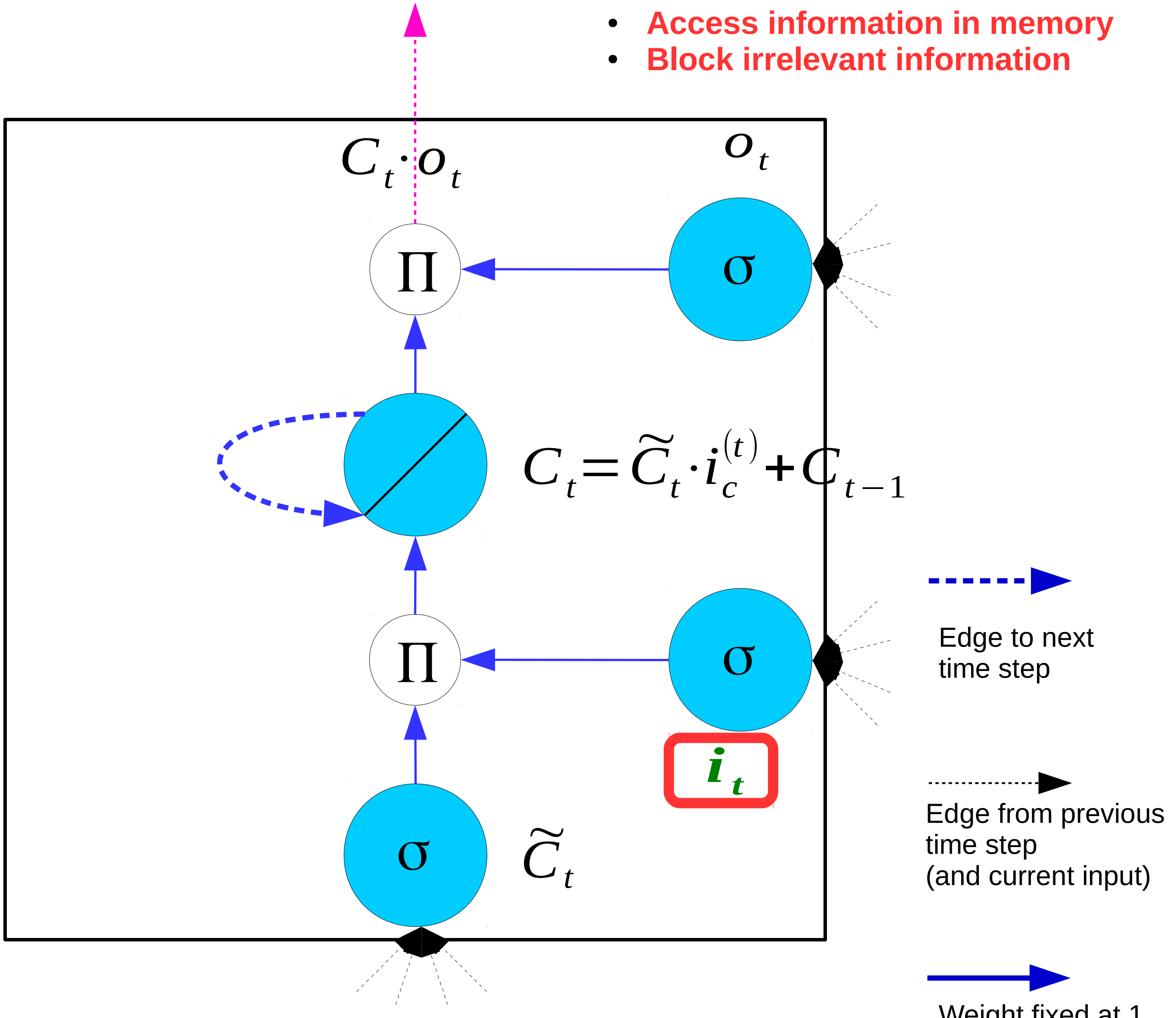
Input gate

- **Use contextual information to decide**
  - **Store input into memory**
  - **Protect memory from overwritten by other irrelevant inputs**

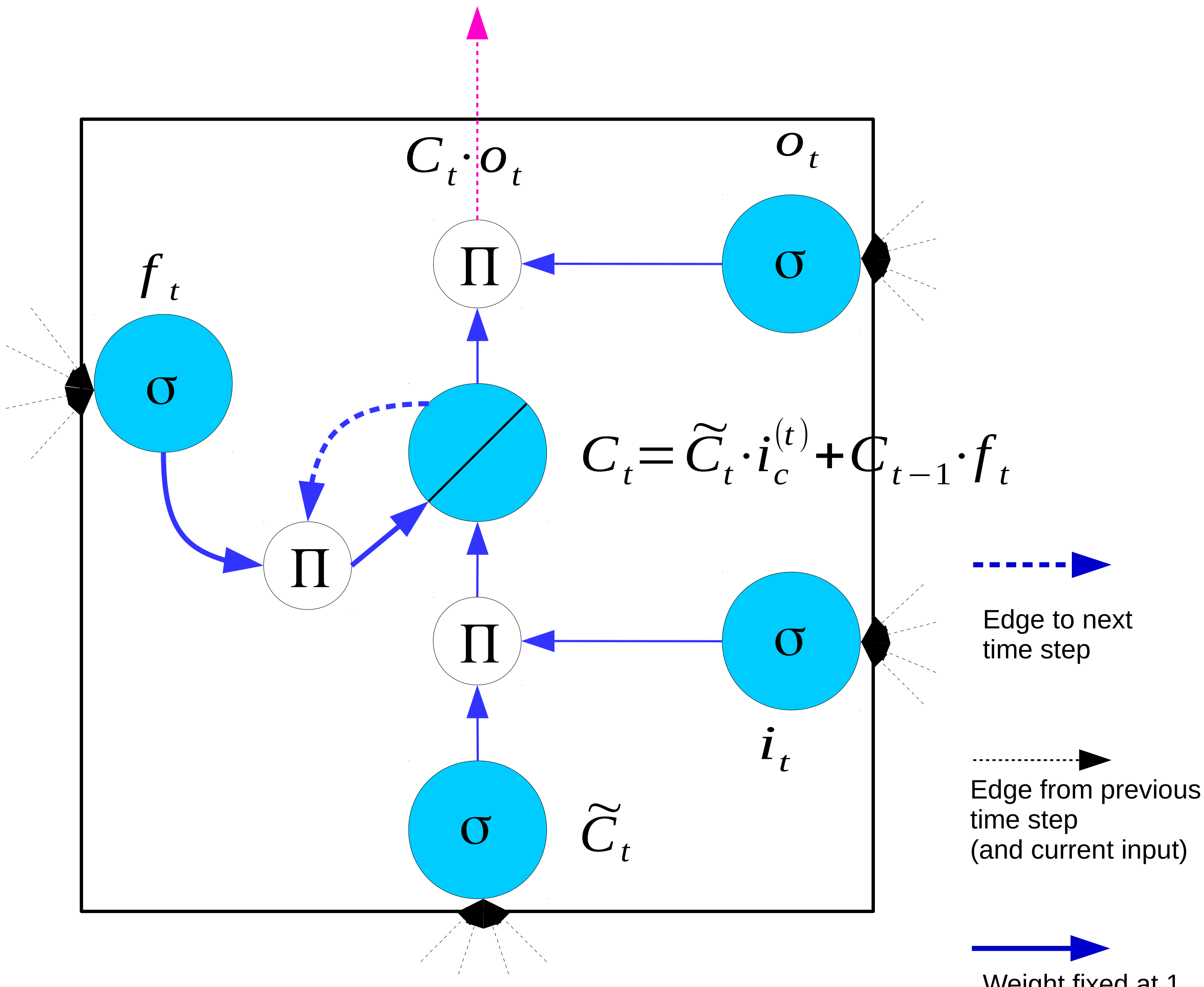


Output gate

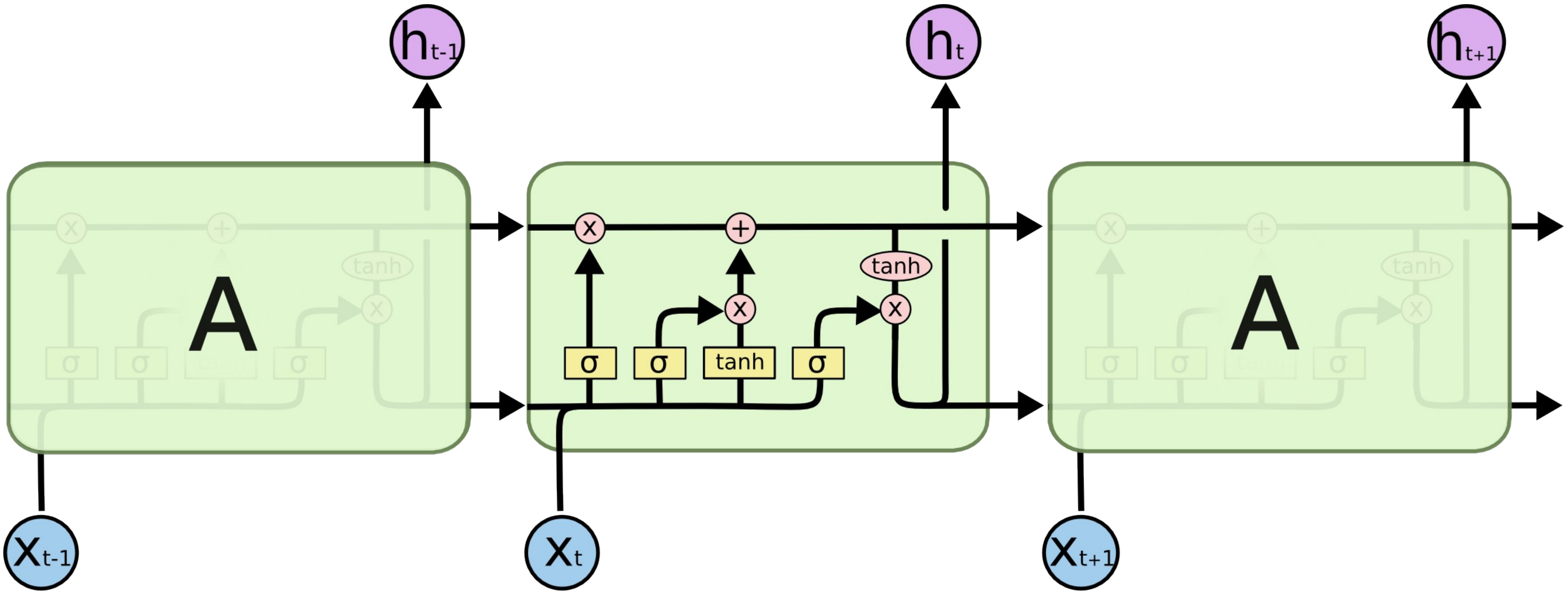
- **Use contextual information to decide**
  - **Access information in memory**
  - **Block irrelevant information**



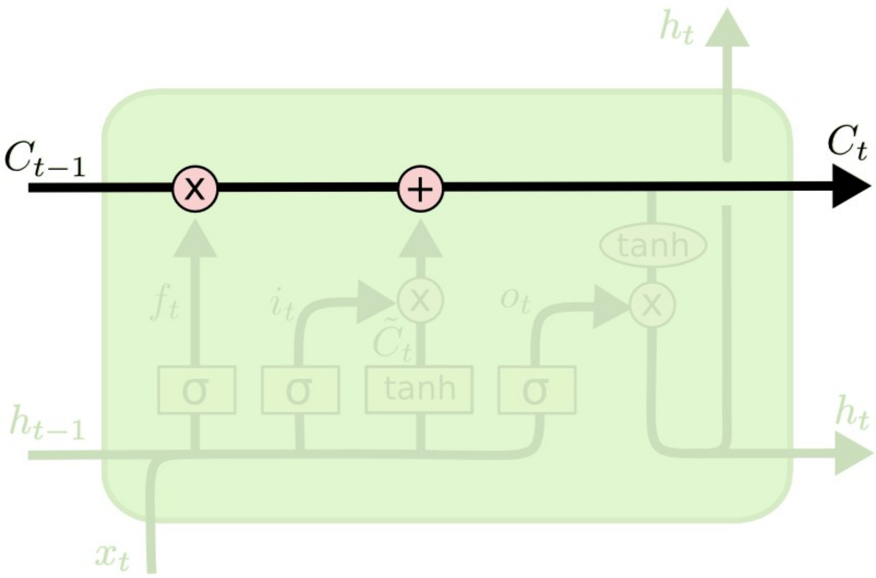
Forget or reset gate



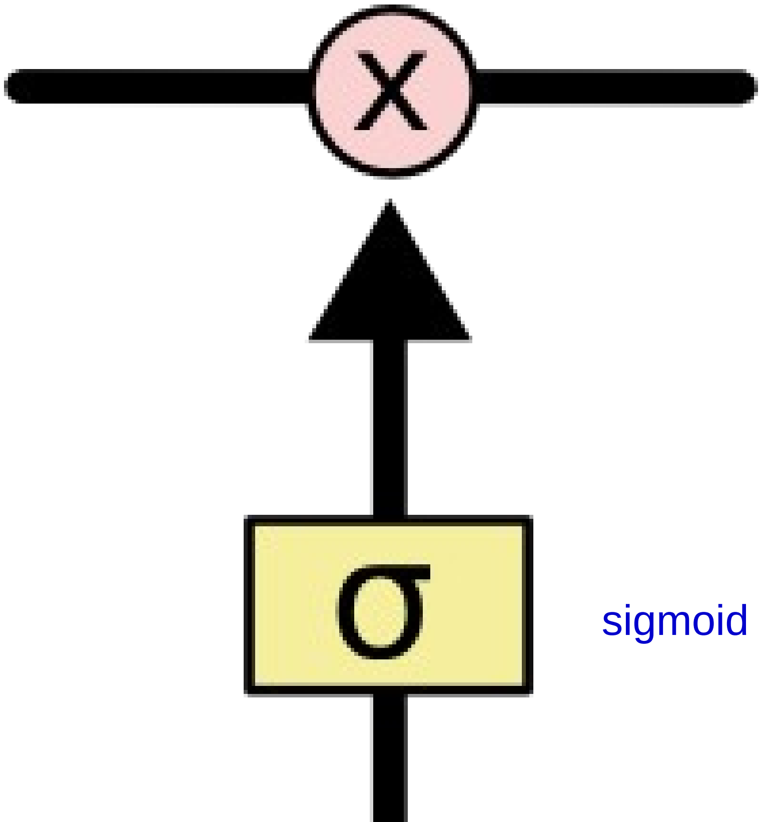
LSTM with four interacting layers



The cell state



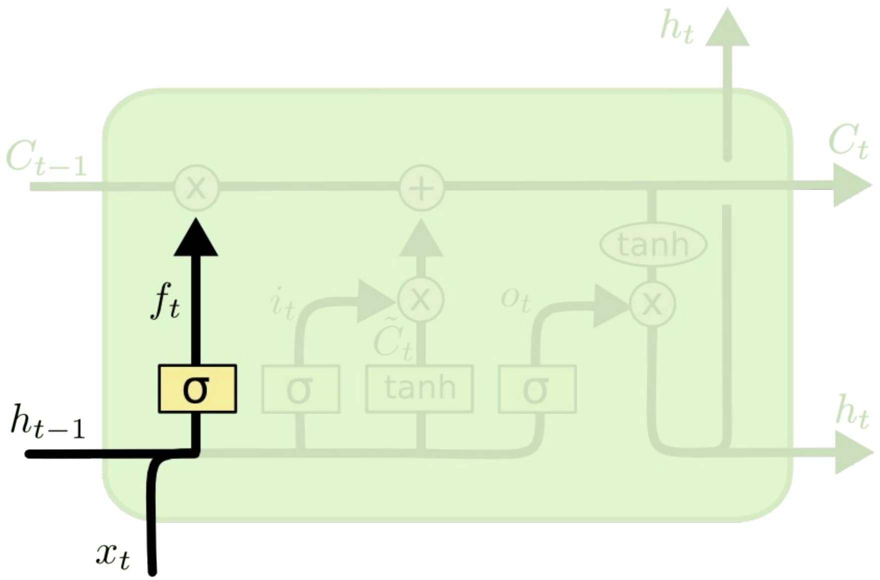
Gates



sigmoid layer

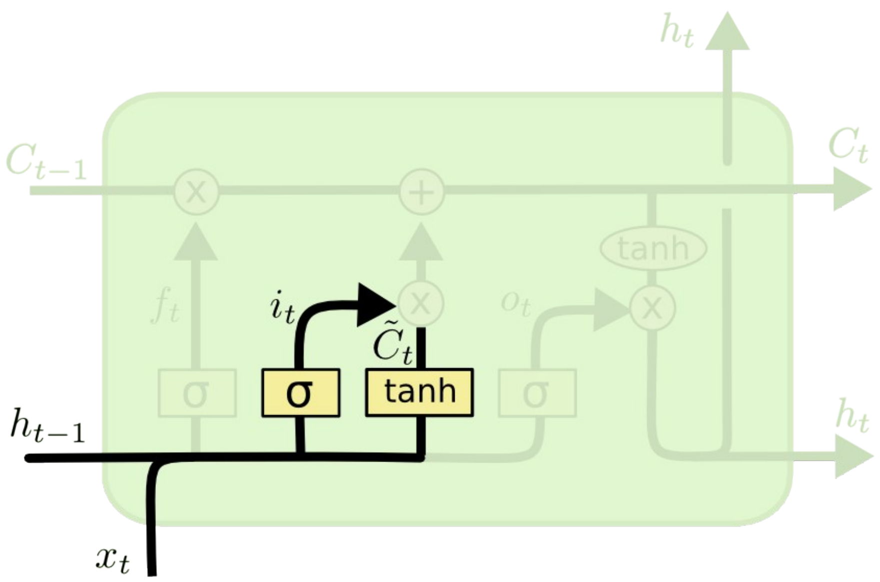
Step-by-Step LSTM Walk Through

Forget gate layer



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

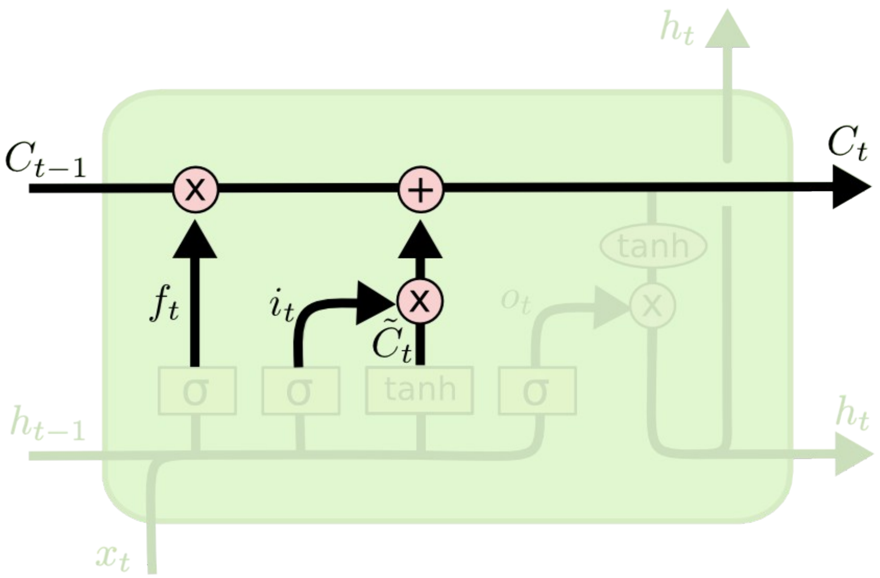
Input gate layer



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

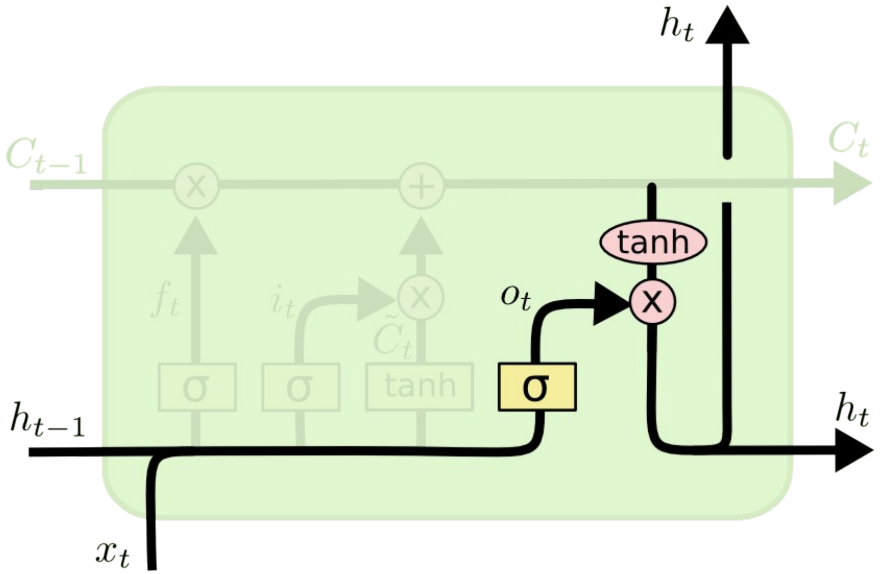
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

The current state



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

Output layer



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

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

Refrence

- http://colah.github.io/posts/2015-08-Understanding-LSTMs/
- http://www.wildml.com/
- http://nikhilbuduma.com/2015/01/11/a-deep-dive-into-recurrent-neural-networks/
- http://deeplearning.net/tutorial/lstm.html
- https://theclevermachine.files.wordpress.com/2014/09/act-funs.png
- http://blog.terminal.com/demistifying-long-short-term-memory-lstm-recurrent-neural-networks/
- A Critical Review of Recurrent Neural Networks for Sequence Learning, Zachary C. Lipton, John Berkowitz
- Long Short-Term Memory, Hochreiter, Sepp and Schmidhuber, Jurgen, 1997
- Gers, F. A.; Schmidhuber, J. & Cummins, F. A. (2000), 'Learning to Forget: Continual Prediction with LSTM.', Neural Computation 12 (10) , 2451-2471 .