UNIVERSITY OF THE WITWATERSRAND, JOHANNESBURG

COMS 4030A/7047A Adaptive Computation and Machine Learning

Hima Vadapalli

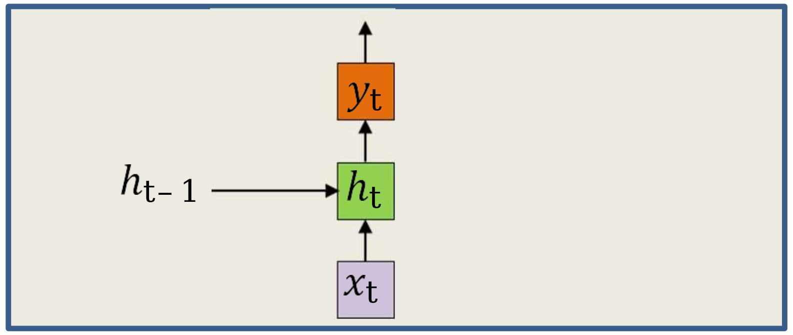
Semester I, 2022

Last lecture:
Recurrent Neural Networks

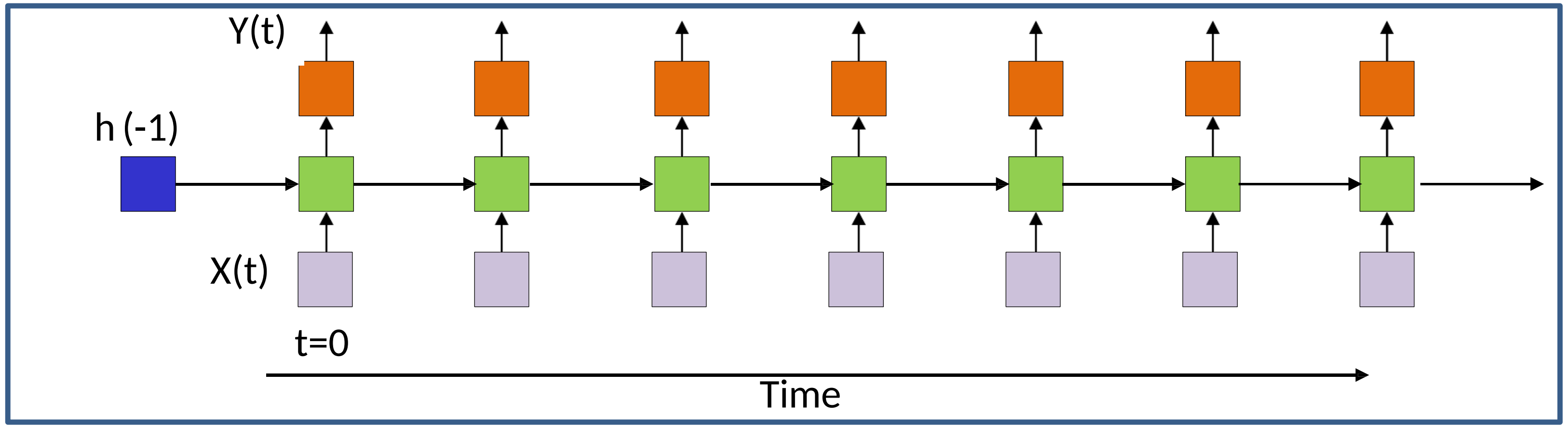
Today:
Long Short-Term Memory (LSTM)

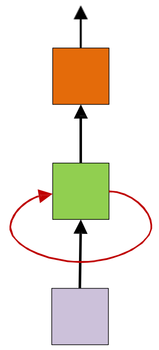
Slides based heavily on course material by Eric
Eton Andrew Moore and Andrew Ng

Recurrent Neural Network: A simple state-space model



$$h_t = f(x_t, h_{t-1})$$
$$y_t = g(h_t)$$

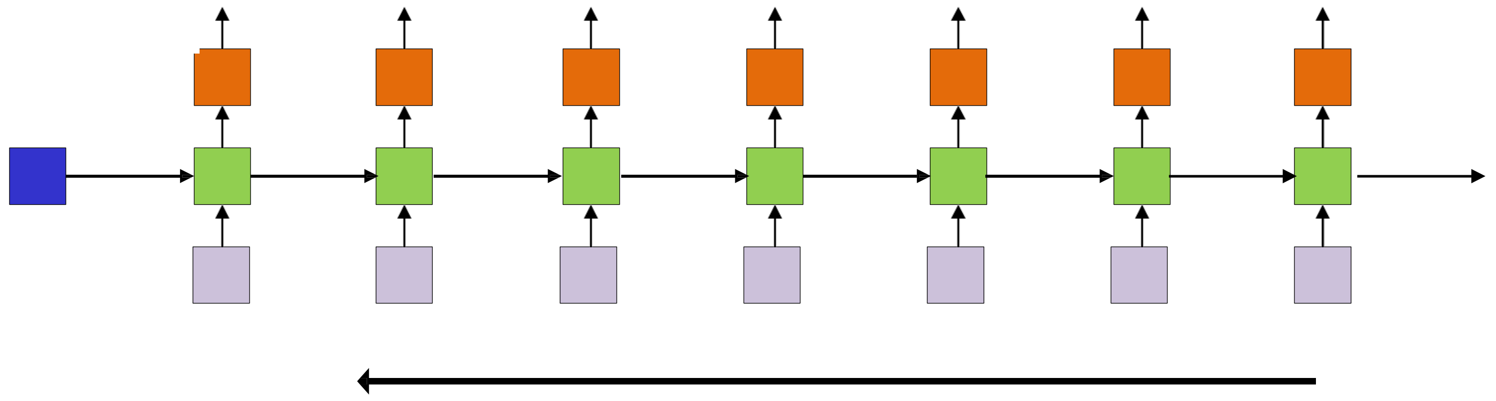




- The state (green) at any time is determined by the input at that time, and the state at the previous time

Problem with RNNs

- Vanishing gradients



- Why is vanishing gradients a problem?
  - Gradient signal from faraway is lost because it’s much smaller than gradient signal from close-by.
  - Model weights are only updated with respect to near effects, not long-term effects.

Effect of vanishing gradient on RNN-Language Modelling

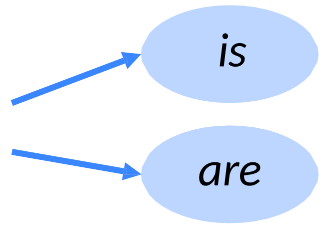
- **LM task:** *When she tried to print her tickets, she found that the printer was out of toner. She went to the stationery store to buy more toner. It was very overpriced. After installing the toner into the printer, she finally printed her* ________

- To learn from this training example, the RNN-LM needs to model the dependency between *"tickets"* on the 7th step and the target word *"tickets"* at the end.

- But if gradient is small, the model can't learn this dependency
  - So the model is unable to predict similar long-distance dependencies at test time

Effect of vanishing gradient on RNN-LM

- **LM task:** *The writer of the books* ____



- Correct answer: *The writer of the books is planning a sequel*

- Syntactic recency: *The writer of the books is* (correct)

- Sequential recency: *The writer of the books are* (incorrect)

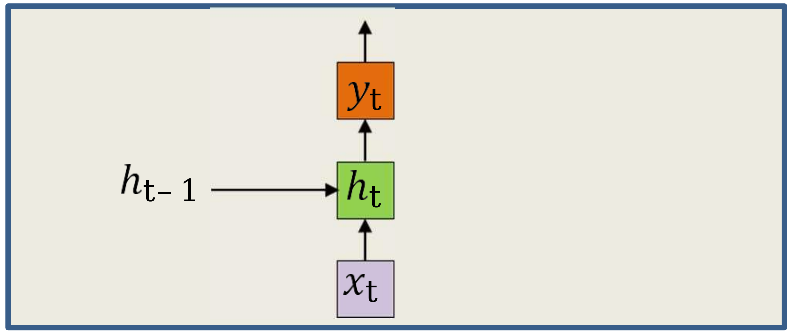
- Due to vanishing gradient, RNN-LMs are better at learning from sequential recency than syntactic recency, so they make this type of error more often

How to fix vanishing gradient problem?

- The main problem is that *it's too difficult for the RNN to learn to preserve information over many timesteps.*

- In a vanilla RNN, the hidden state is constantly being rewritten

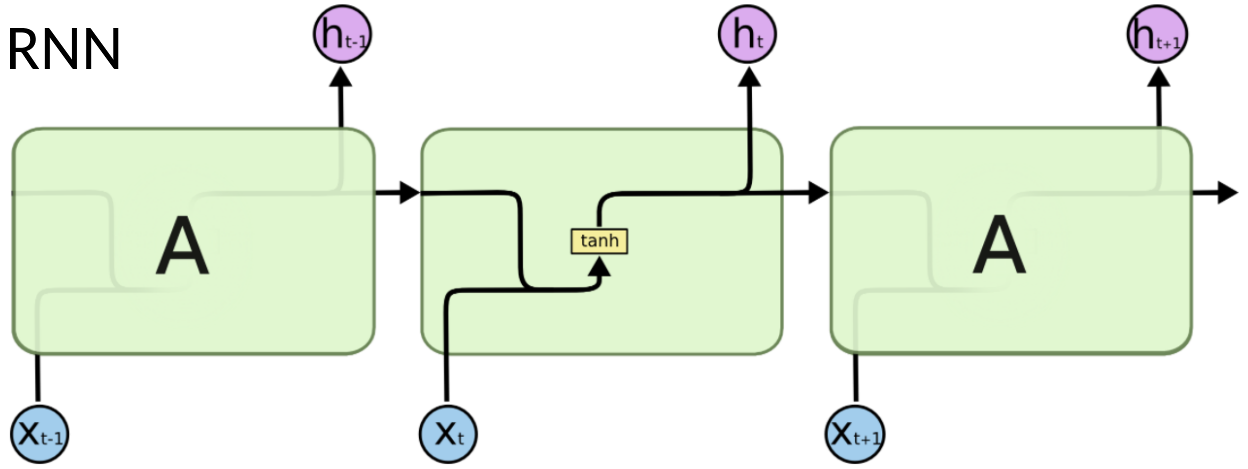
$$\boldsymbol{h}^{(t)} = \sigma\left(\boldsymbol{W}_h \boldsymbol{h}^{(t-1)} + \boldsymbol{W}_x \boldsymbol{x}^{(t)} + \boldsymbol{b}\right)$$



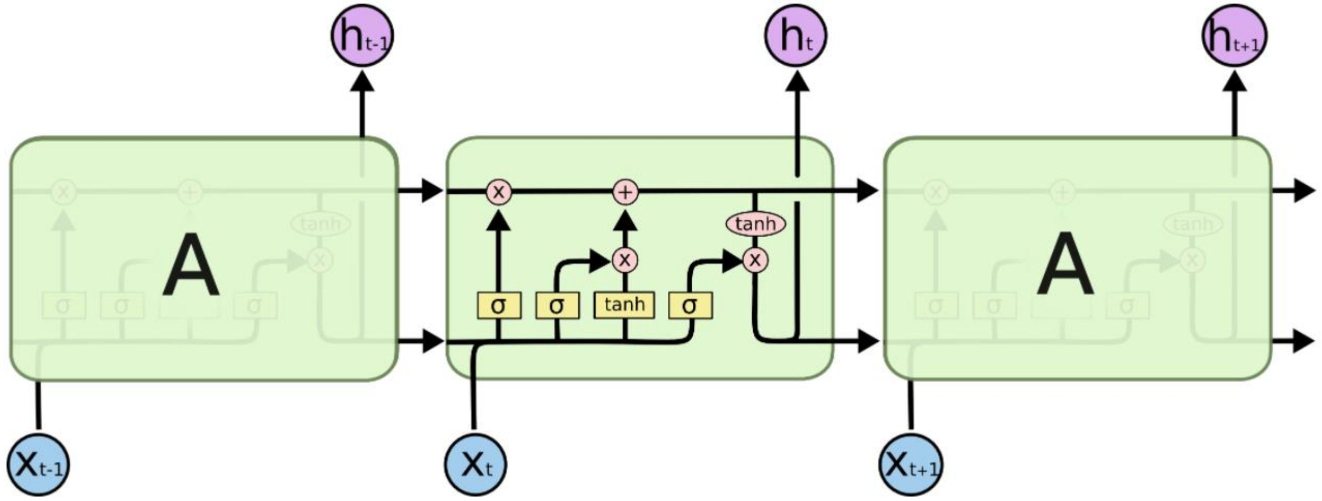
- How about a RNN with separate memory?

Adding memory unit…

- RNN

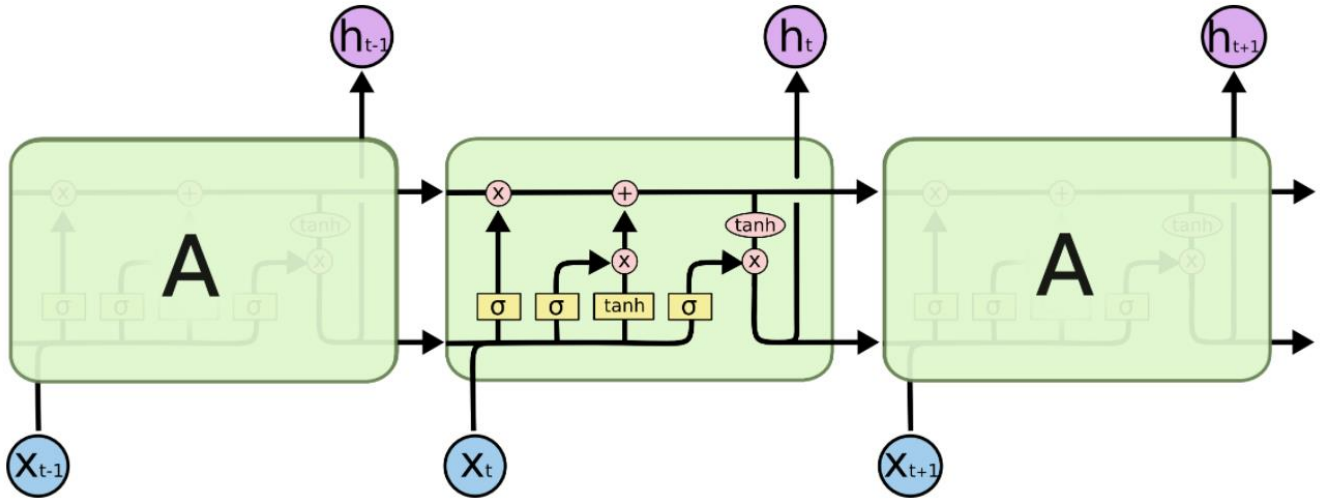


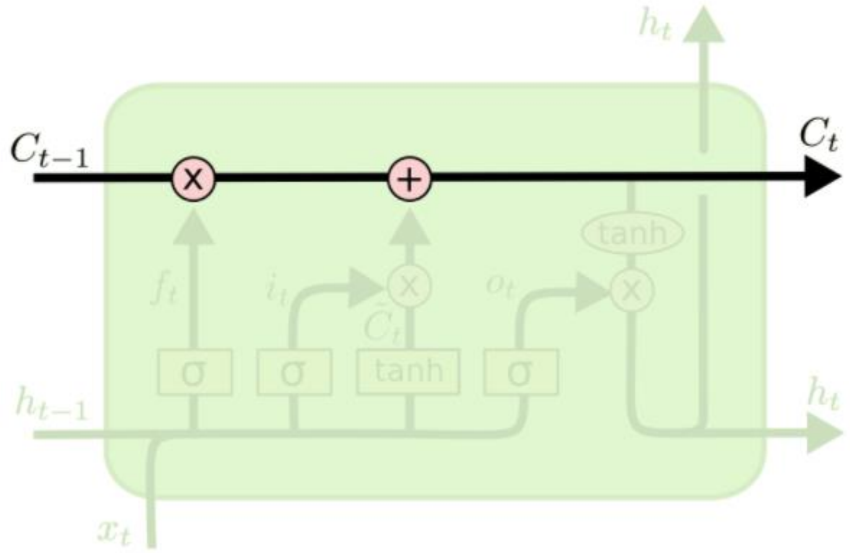
- LSTM



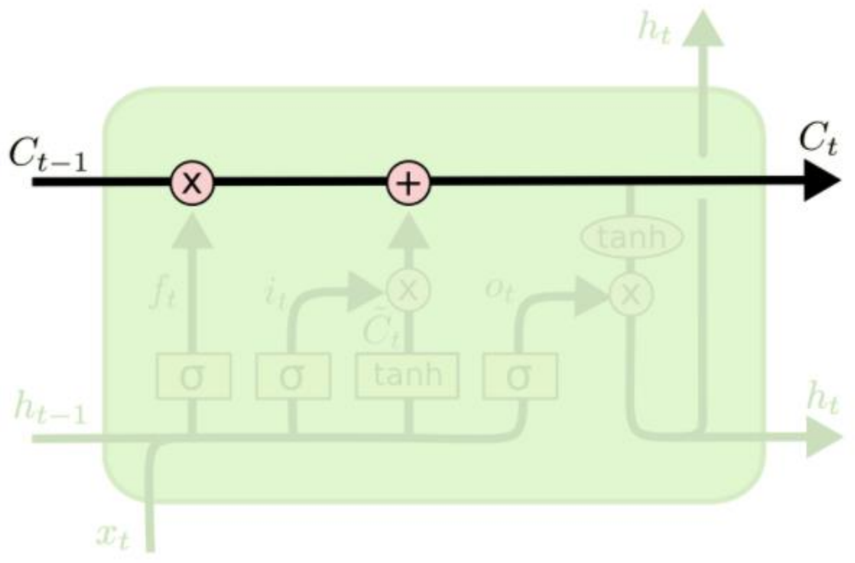
The core idea behind LSTM: Cell State

- LSTM

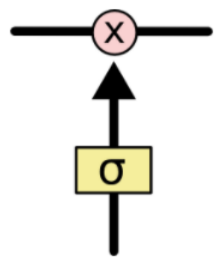




So how do we add/remove information to/from the cell state?

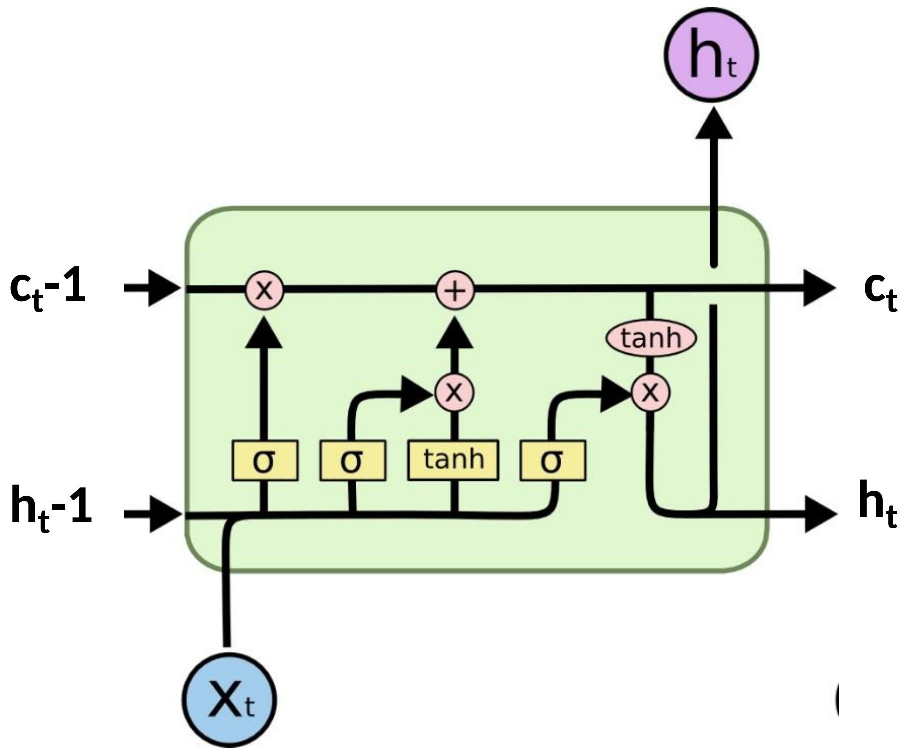


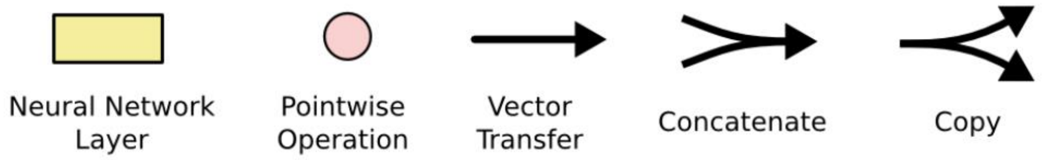
- Gates



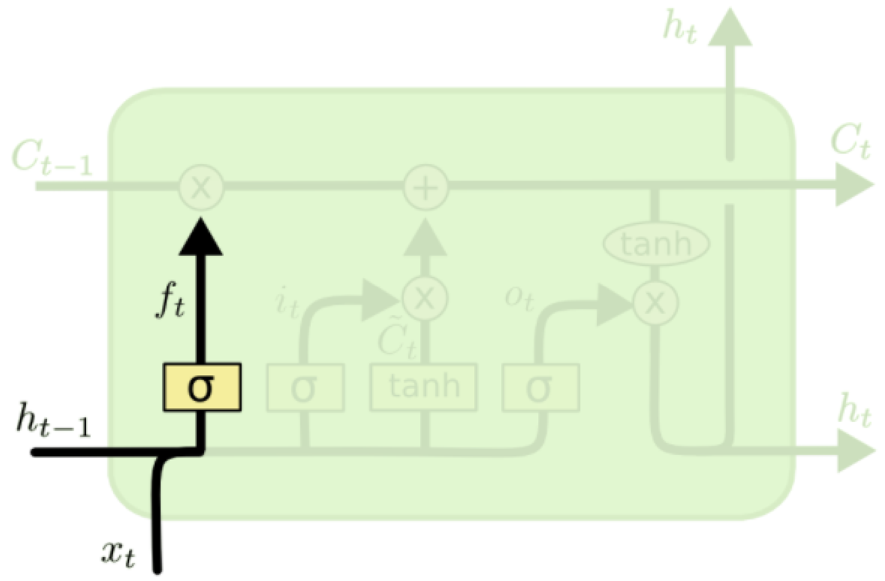
- Forget Gate
- Input Gate
- Output Gate

Let us first understand some notation...



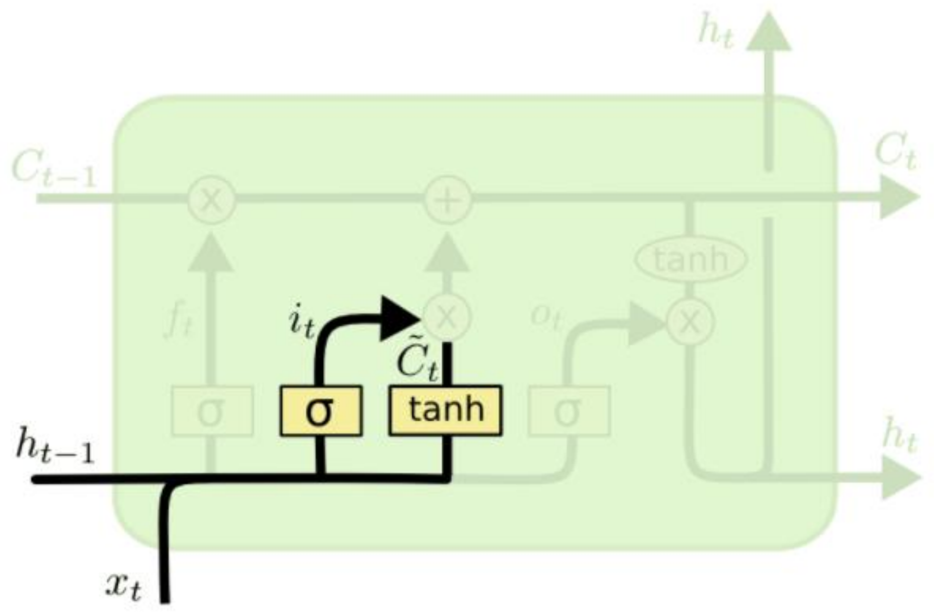


Forget Gate: what can I safely forget?



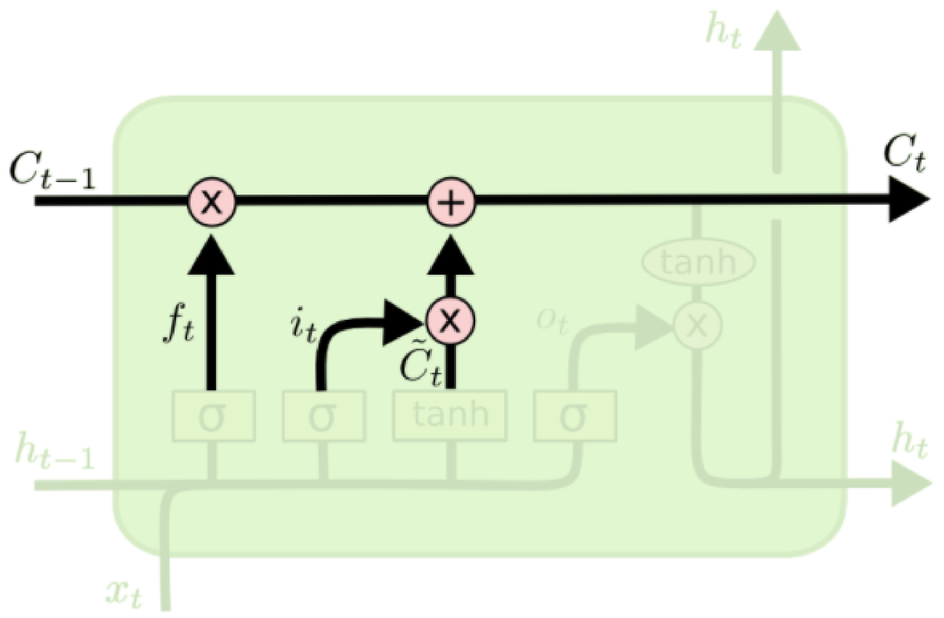
$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \ + \ b_f\right)$$

Input Gate: what new information to use?



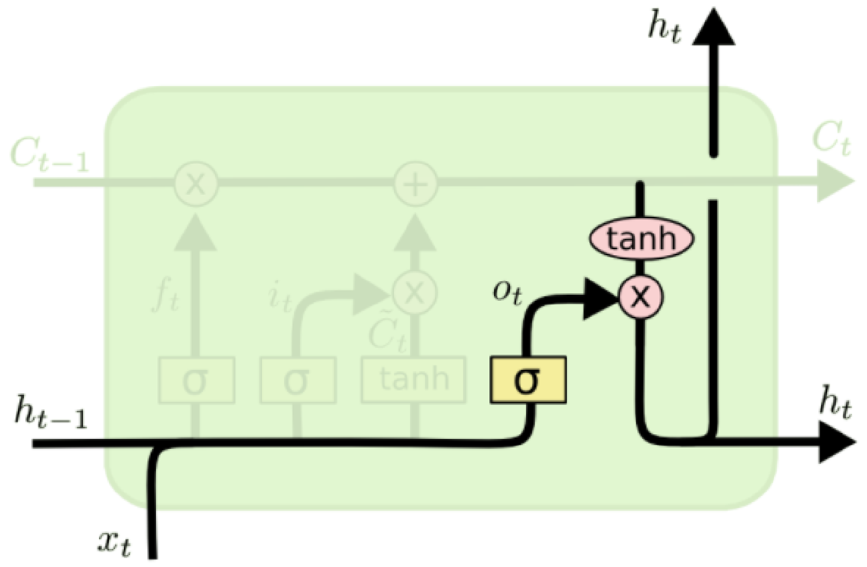
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

So we forget some information and remember some new information



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

Output gate: What to output?



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

Other popular variant of LSTMs

- Gated Recurrent Unit
  - Combines forget and input gate into a single “update” gate
  - Merges the cell state and hiddens state

  Etc…

  GRUs are simpler versions of LSTMs

Other solutions to vanishing gradient

- Gradient clipping
- Skip connections