Understanding LSTM Network

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

Presenter : Hongming Shan
shanh@rpi.edu

Recurrent Neural Networks

Human action recognition

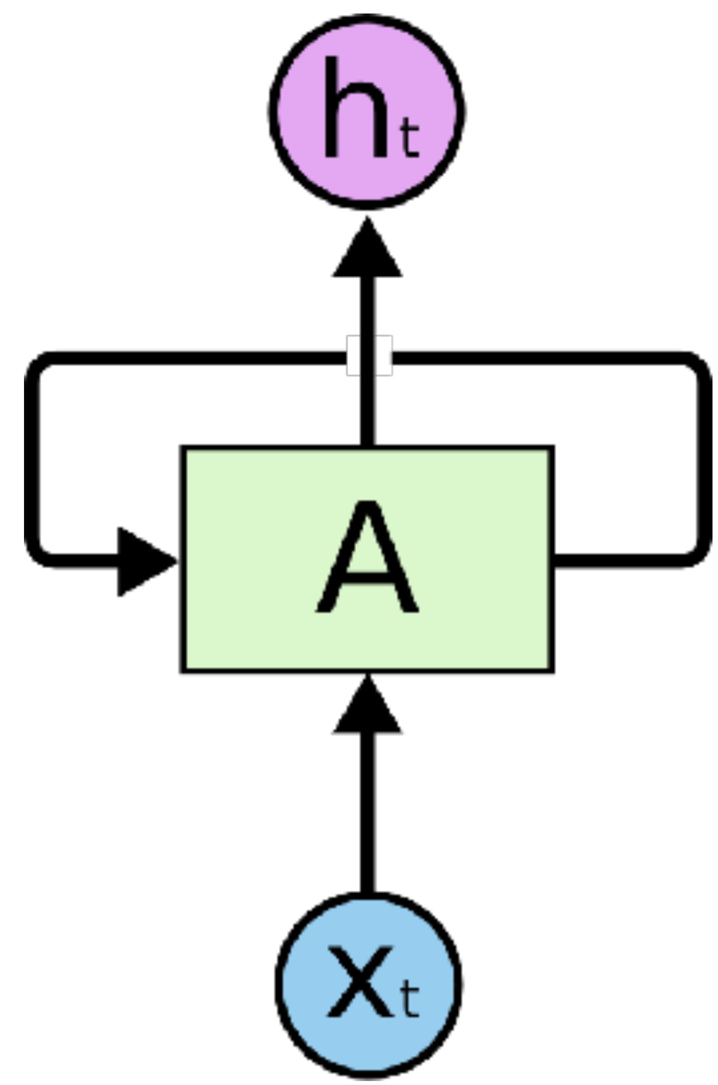
Input: a sequence of body images

Output: Jumping

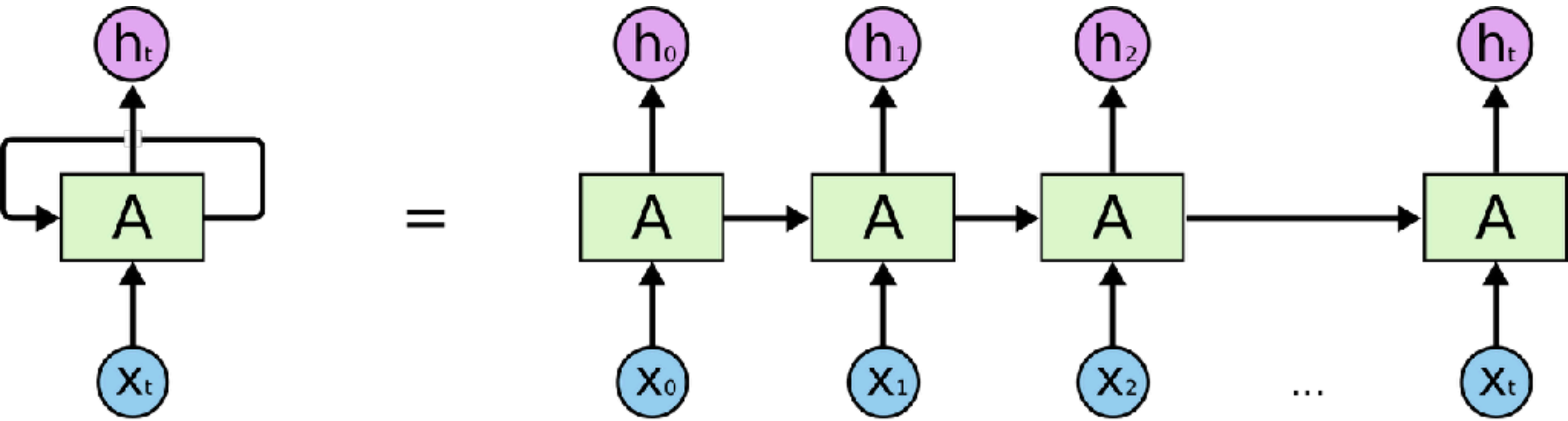
- Traditional neural networks cannot deal with sequential data
- RNN captures and exploits temporal dependences among all input samples to predict output variable

RNN

- **A** looks at some input $\boldsymbol{x_t}$ and outputs a value $\boldsymbol{h_t}$
- A loop allows information to be passed from one step of network to the next



Unrolled RNN



- A RNN can be viewed as multiple copies of the same network, each passing a message to a successor

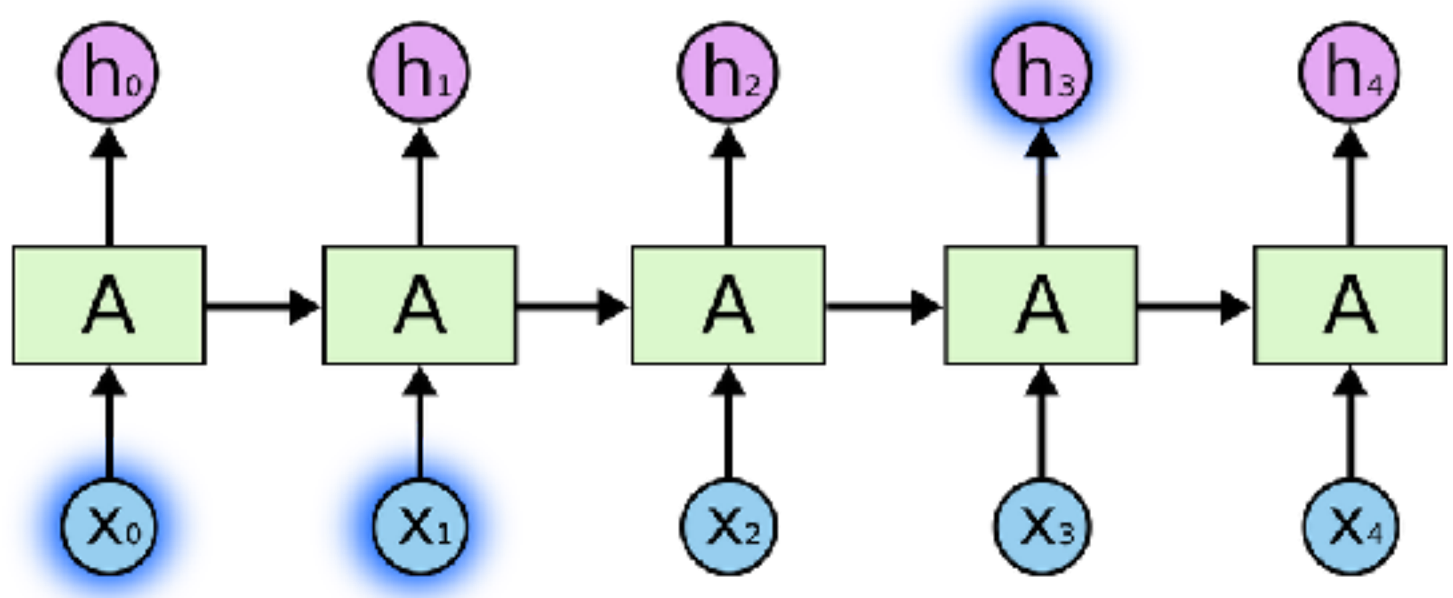
- RNN has memory, **A**

Applications

- Applying RNN to a variety of problems:
  - Speech recognition
  - Language modeling
  - Translation
  - Image caption
  - ……
- Essential to above success is the use of LSTM, a very special kind of RNN.

Short-Term dependencies

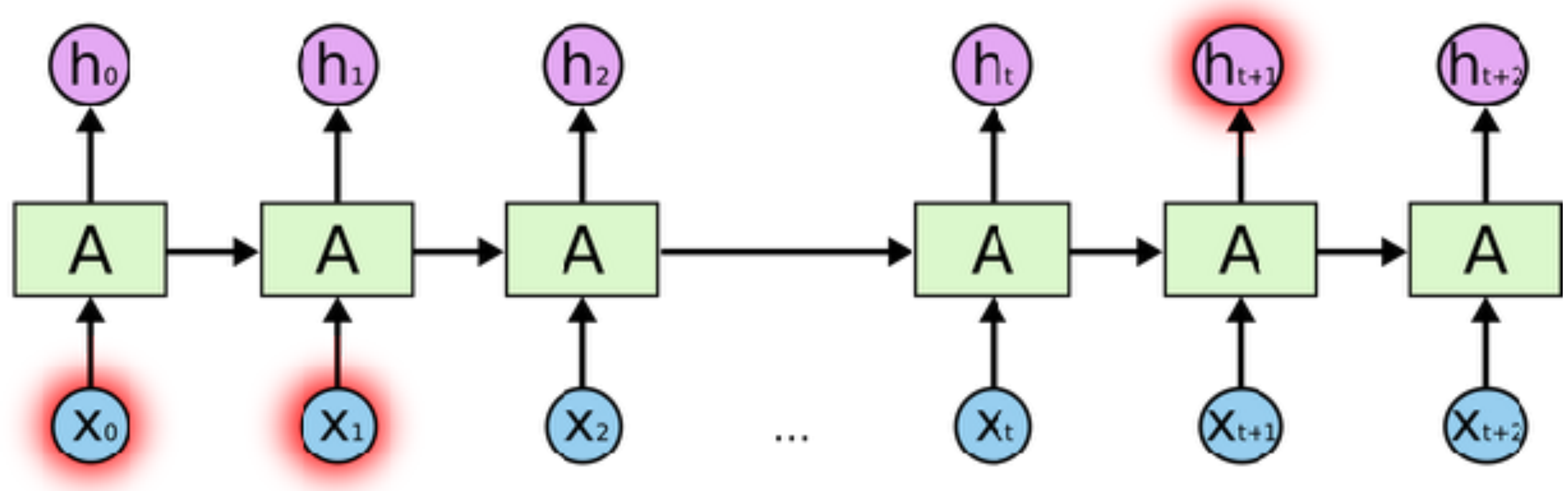
The clouds are in the _____ (sky)



- We don't need any further context; Gap between relevant information the the space that it is needed is small. **Short-term dependencies**.

Long-Term dependencies

I grew up in China….. I speak fluent _____ (Chinese)



- Recent information suggests that the next word is probably the name of a language, but we need the context of China, from further back.
- Gap may be very large in real case.

Long-term dependencies

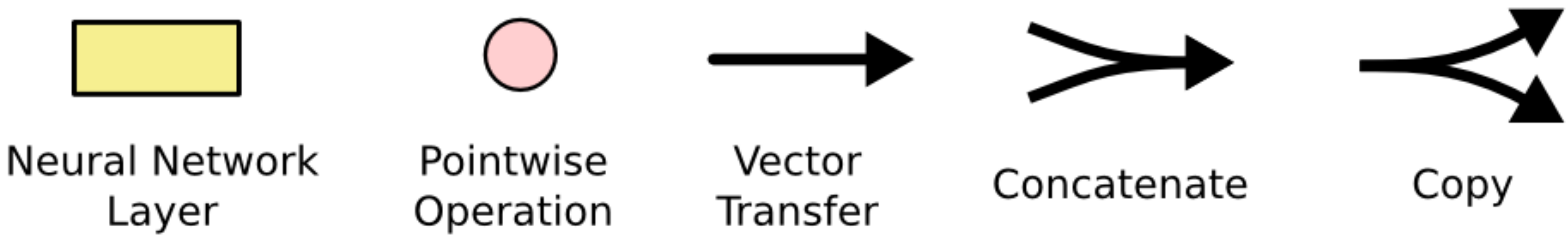
- As that gap grows, RNN becomes unable to learn to connect the information.
- In theory, RNN is absolutely capable of handing such long-term dependencies.
- Sadly, in practice, RNN doesn’t seem to be able to learn them
- Y. Bengio, et al.. "Learning long-term dependencies with gradient descent is difficult." IEEE transactions on neural networks1994

LSTM doesn’t have this problem….

Long Short-Term Memory network (LSTM)

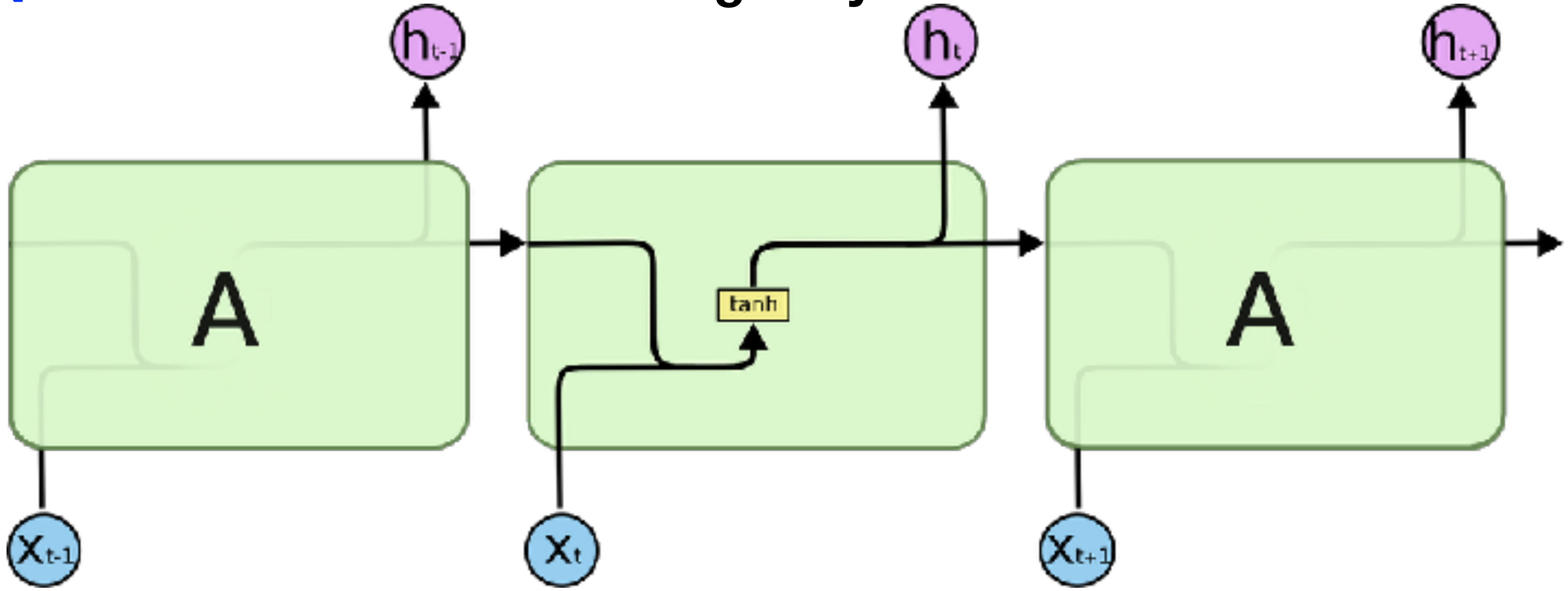
- LSTM is a special kind of RNN, capable of learning long-term dependencies
- LSTM works tremendously well on a large variety of problem, and are not widely used.
- Remembering information for long periods of time is practically their default behavior.

Notation

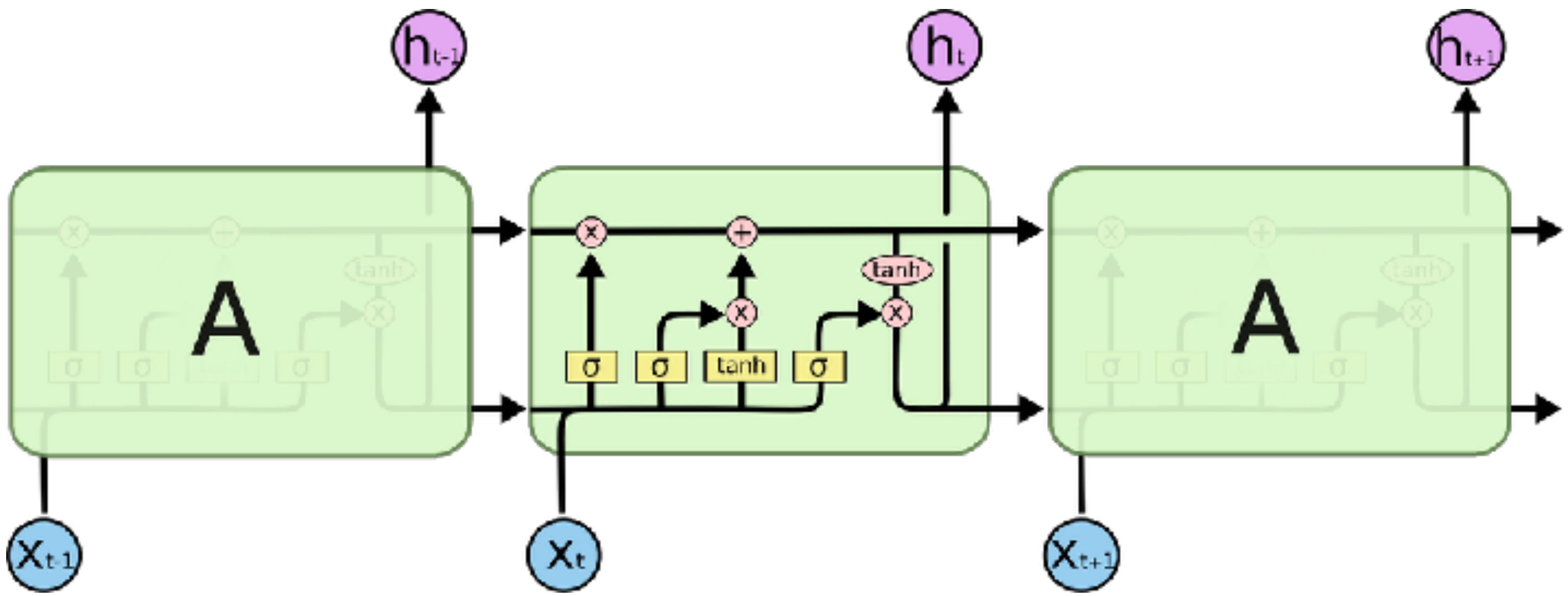

Neural Network Layer
Pointwise Operation
Vector Transfer
Concatenate
Copy

Standard RNN

One single layer



LSTM

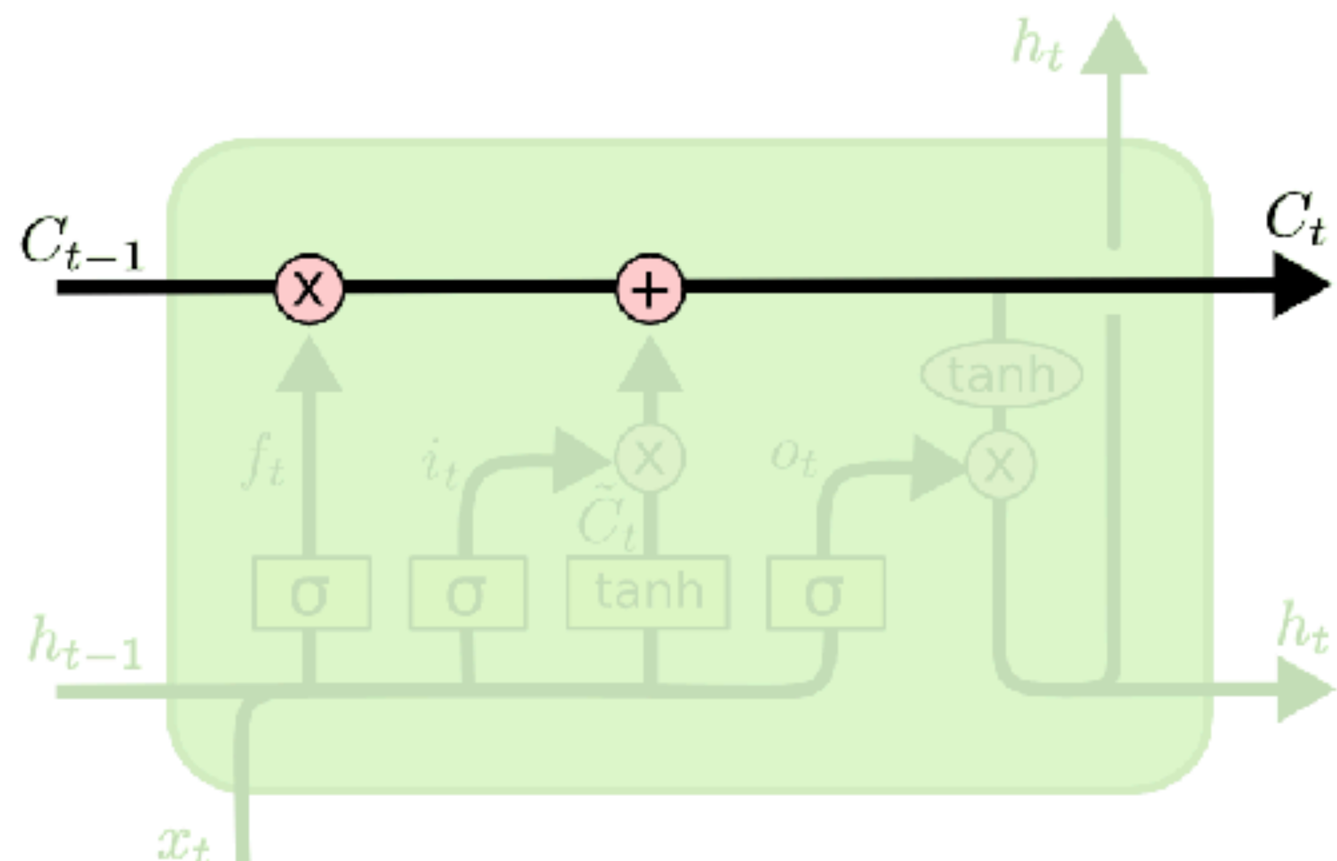


Four interacting layers

Core idea behind LSTM

Core idea: Cell State

- The key to LSTM is the cell state
- Information to flow along it…

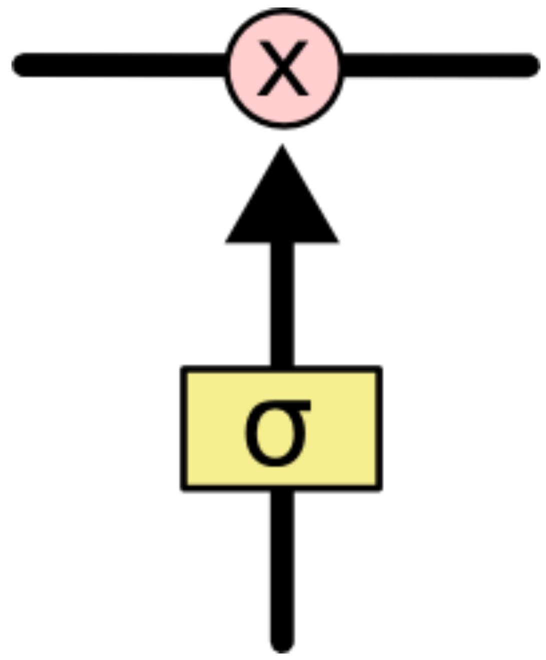


Core idea: Gate

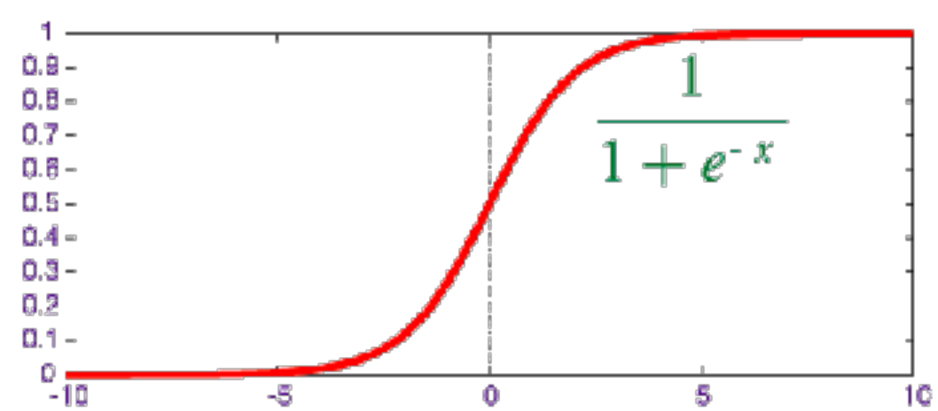
- LSTM has the ability to **add** or **remove** information to the cell state, regulated by structures called gates

- Gates are a way to optionally let information through, composed of a sigmoid NN layer and a point-wise multiplication operation

- Sigmoid outputs 0~1, describing how much of component should be let through



1: Let everything through

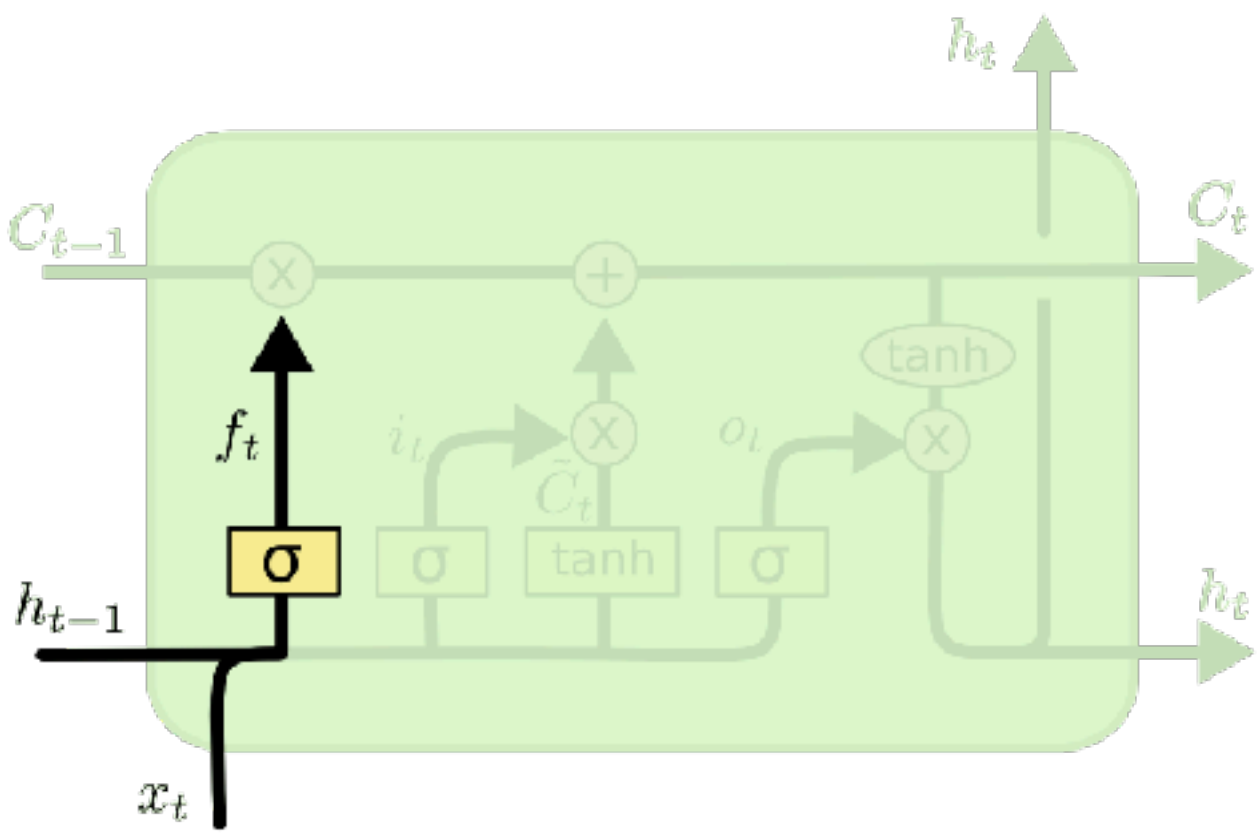


0: Let nothing through

Step-by-Step LSTM Walk Through

Step 1: Forget Gate Layer

- Step1: decide what information we're going to throw away from cell state.
- This decision is made by a sigmoid layer, **forget gate layer**



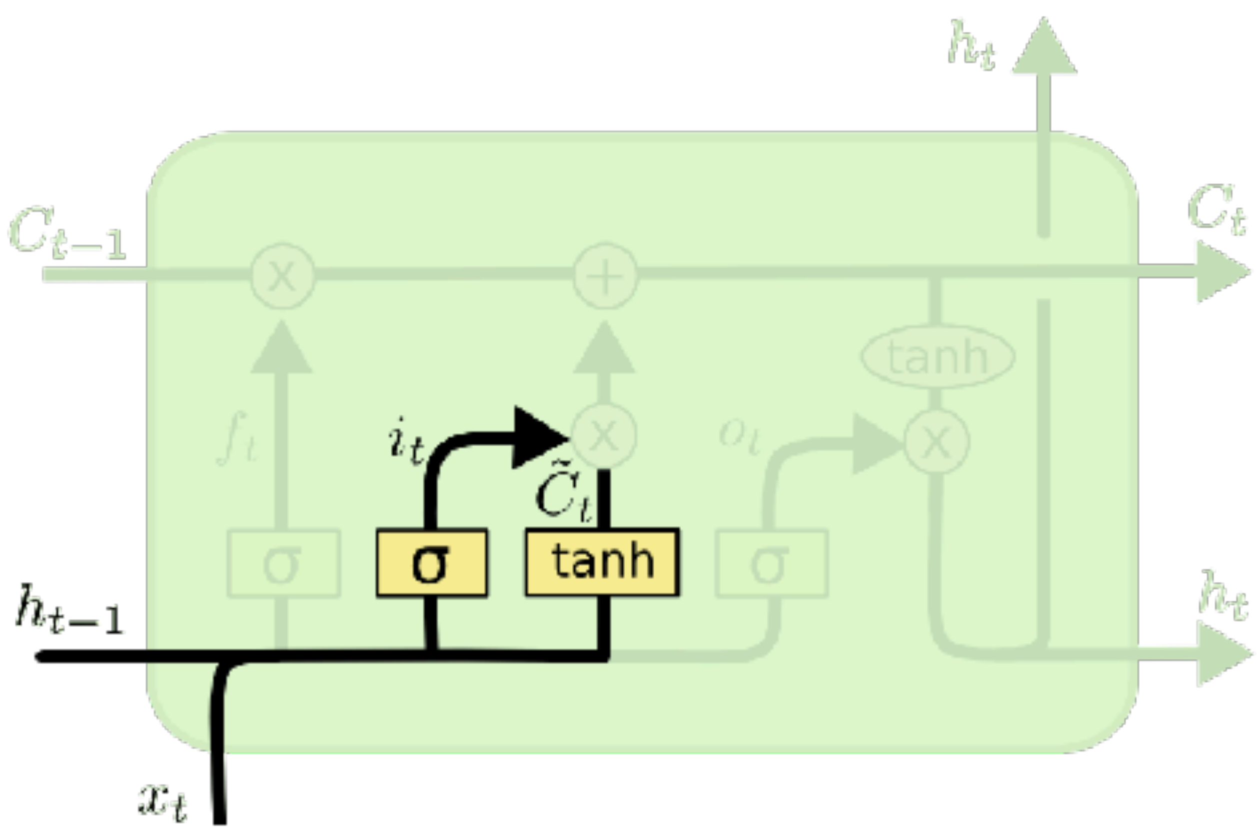
$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \ + \ b_f\right)$$

Example..

- Let’s go back to our example of a language model trying to predict the next word based on all the previous ones.
- In such a problem, the cell state might include the gender of the present subject, so that the correct pronouns can be used.
- When we see a new subject, we want to forget the gender of the old subject.

Step 2: Input Gate Layer

- Step2: decide what new information we're going to store in the cell state.
  - A sigmoid layer called input gate layer: decides which values we'll update
  - A tanh layer creates a vector of new candidates value, $\tilde{C}_t$



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

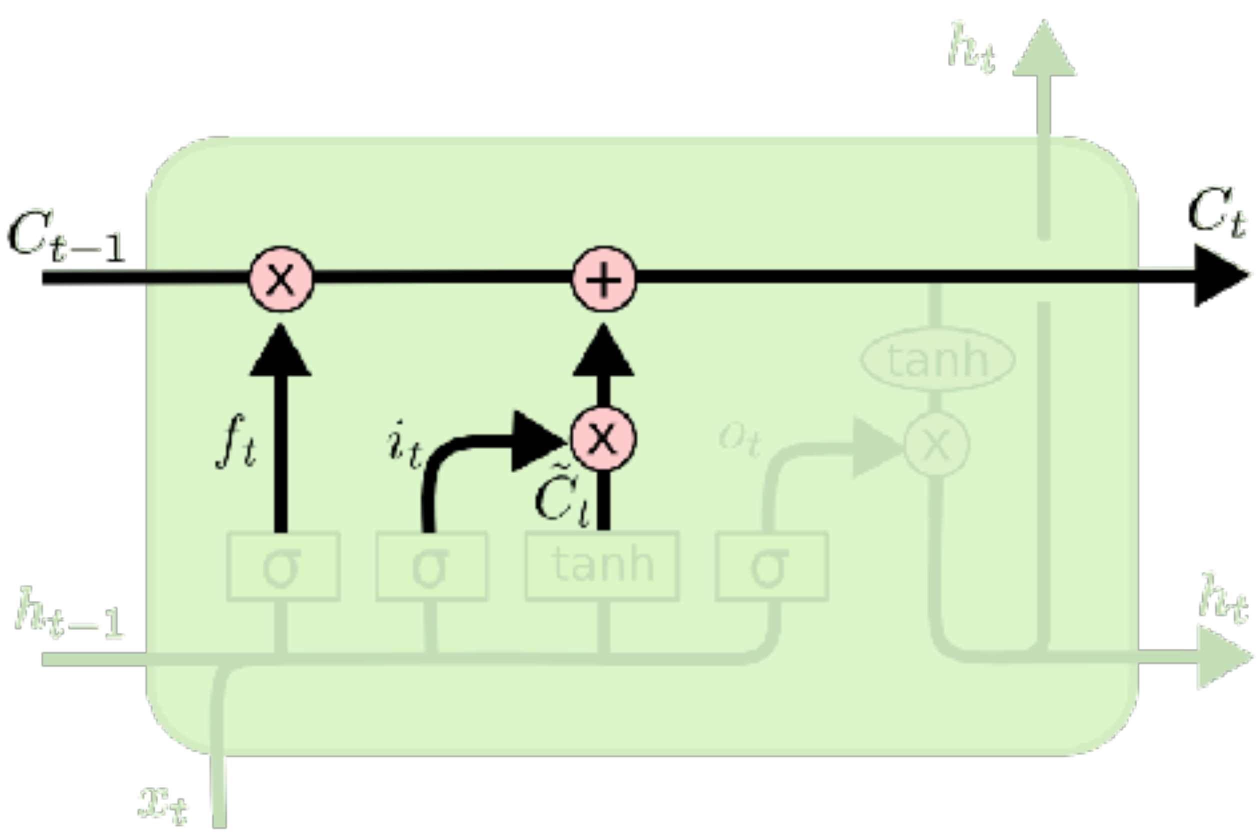
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

Example

- In the example of our language model, we’d want to add the gender of the new subject to the cell state, to replace the old one we’re forgetting.
-

Update Cell State

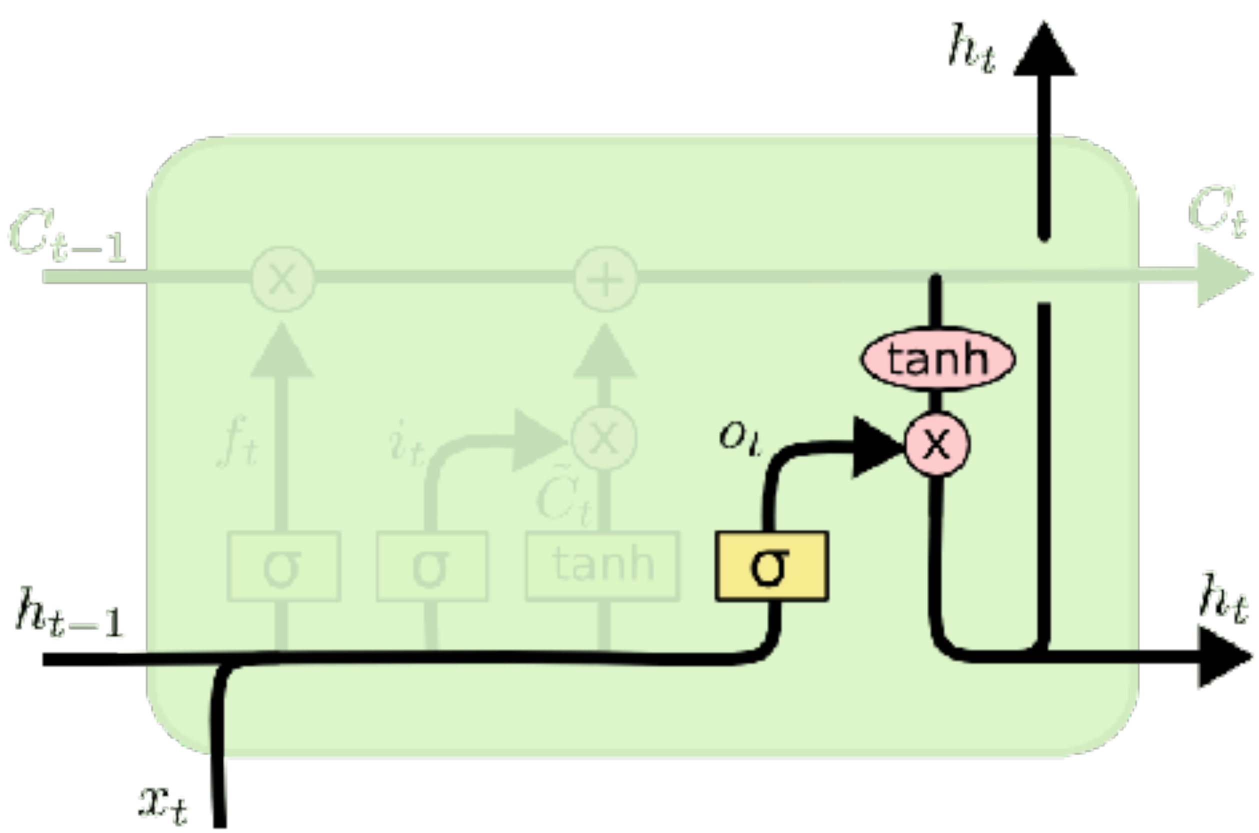
- We first forget something, and then add something new…



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

Step 3: Output Gate Layer

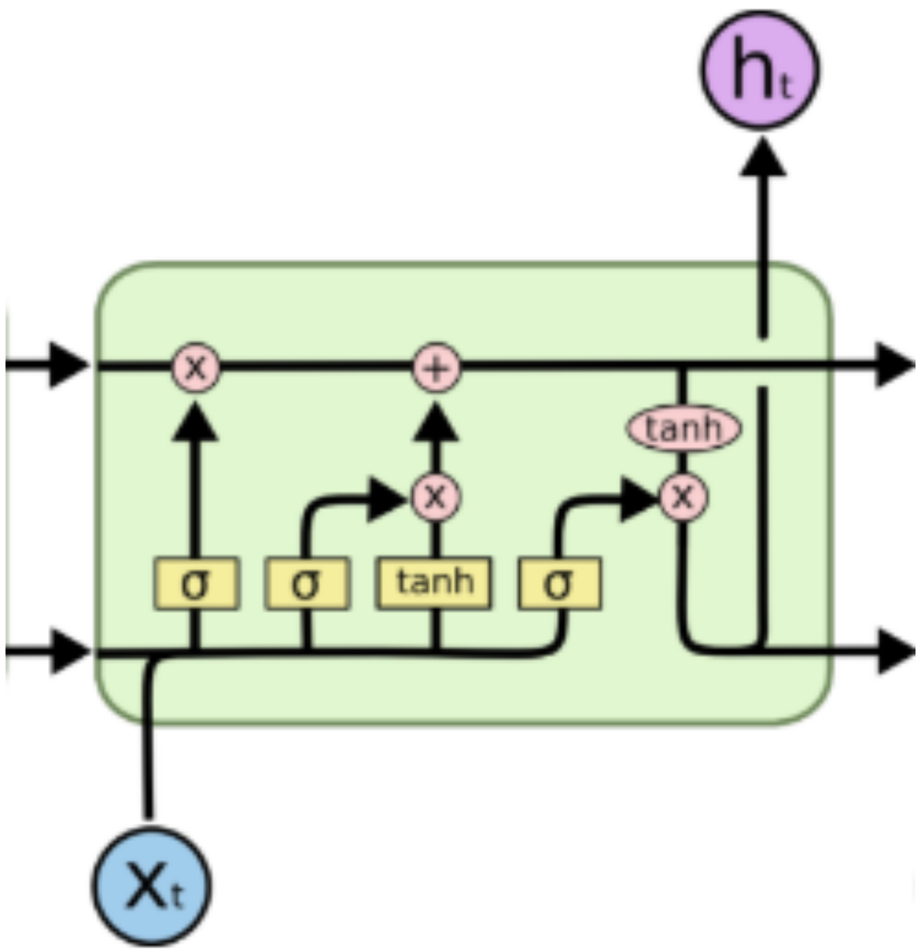
- Step 3: decide what we're going to output…
  - A sigmoid layer decides what parts of cell states to output
  - Put cell state through tank for output



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

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

LSTM



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

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

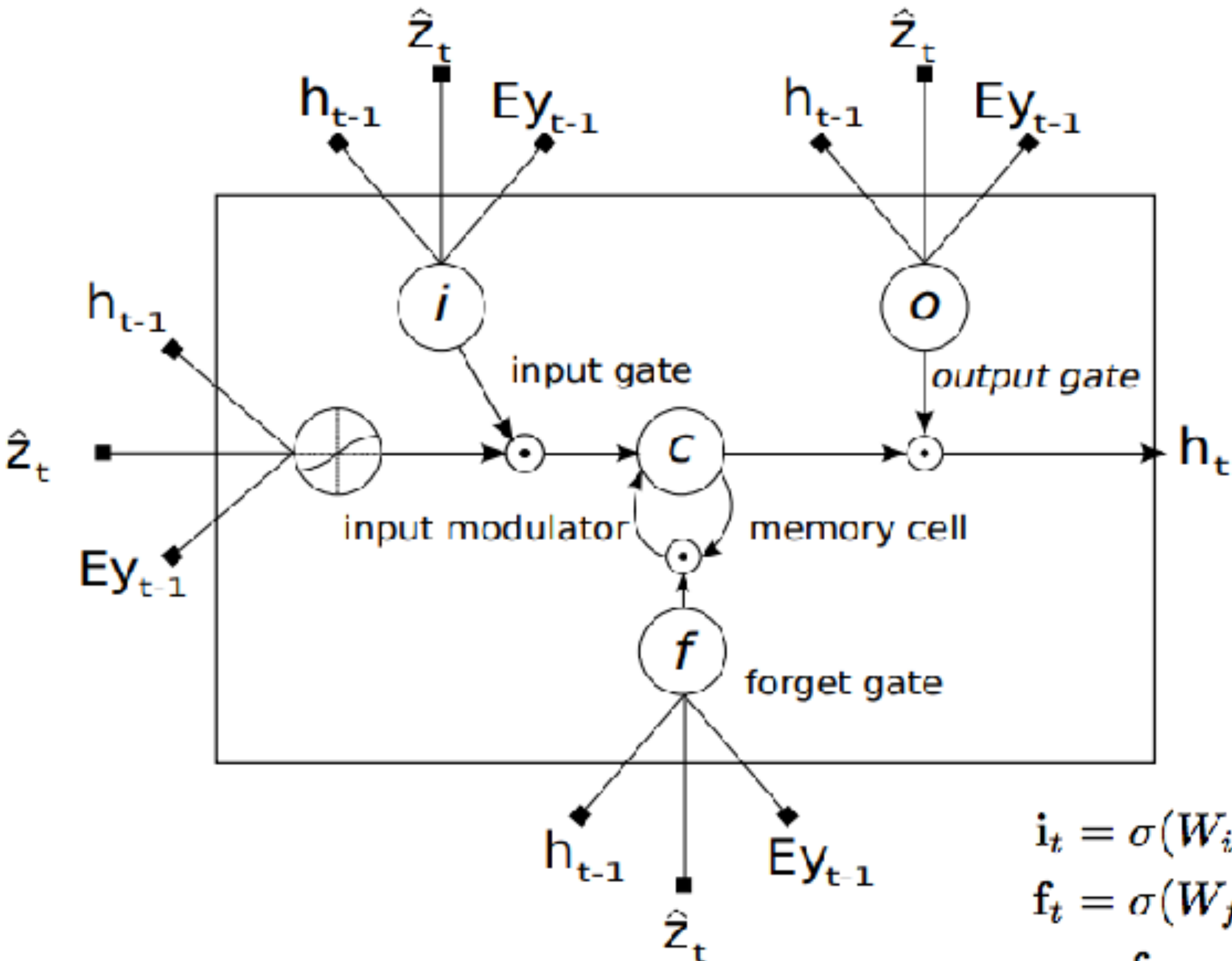
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

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

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

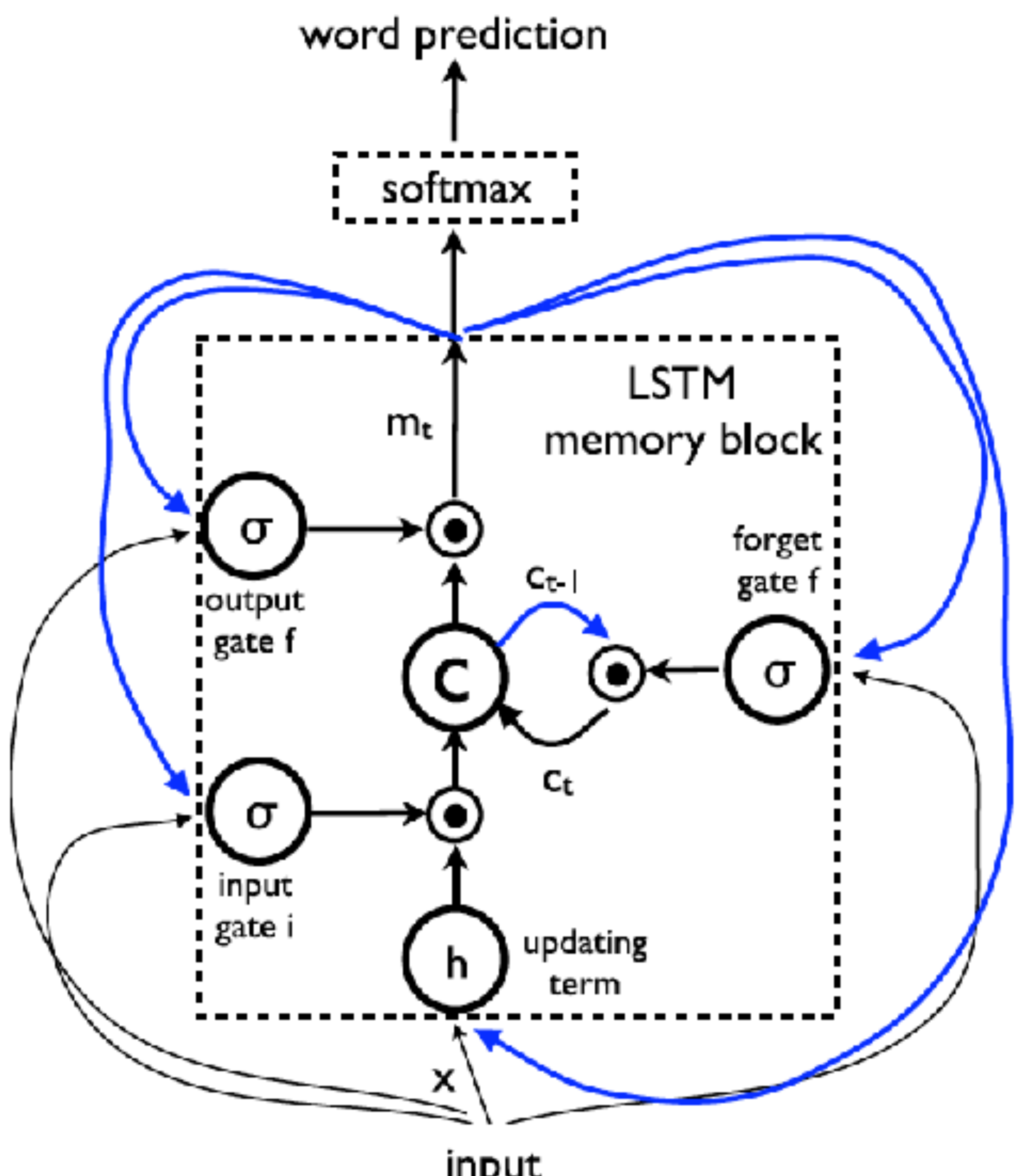
$$h_t = o_t * \tanh\left(C_t\right)$$

LSTM in paper



$$
\begin{aligned}
\mathbf{i}_t &= \sigma(W_i E\mathbf{y}_{t-1} + U_i\mathbf{h}_{t-1} + Z_i\hat{\mathbf{z}}_t + \mathbf{b}_i),\\
\mathbf{f}_t &= \sigma(W_f E\mathbf{y}_{t-1} + U_f\mathbf{h}_{t-1} + Z_f\hat{\mathbf{z}}_t + \mathbf{b}_f),\\
\mathbf{c}_t &= \mathbf{f}_t\mathbf{c}_{t-1} + \mathbf{i}_t \tanh(W_c E\mathbf{y}_{t-1} + U_c\mathbf{h}_{t-1} + Z_c\hat{\mathbf{z}}_t + \mathbf{b}_c),\\
\mathbf{o}_t &= \sigma(W_o E\mathbf{y}_{t-1} + U_o\mathbf{h}_{t-1} + Z_o\hat{\mathbf{z}}_t + \mathbf{b}_o),\\
\mathbf{h}_t &= \mathbf{o}_t \tanh(\mathbf{c}_t).
\end{aligned}
$$

LSTM in paper



$$i_t = \sigma(W_{ix}x_t + W_{im}m_{t-1}) \tag{4}$$

$$f_t = \sigma(W_{fx}x_t + W_{fm}m_{t-1}) \tag{5}$$

$$o_t = \sigma(W_{ox}x_t + W_{om}m_{t-1}) \tag{6}$$

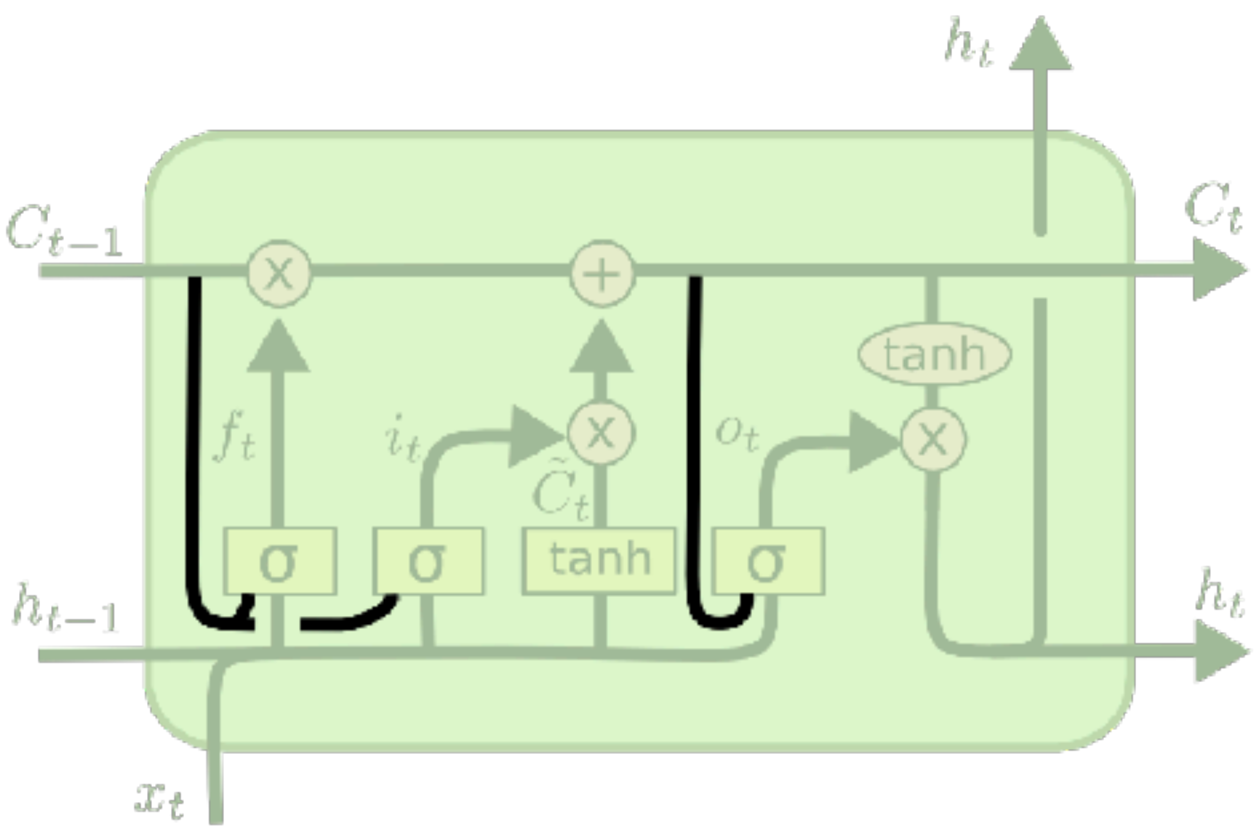
$$c_t = f_t \odot c_{t-1} + i_t \odot h(W_{cx}x_t + W_{cm}m_{t-1}) \tag{7}$$

$$m_t = o_t \odot c_t \tag{8}$$

$$p_{t+1} = \text{Softmax}(m_t), \tag{9}$$

Variants: Peephole Connection

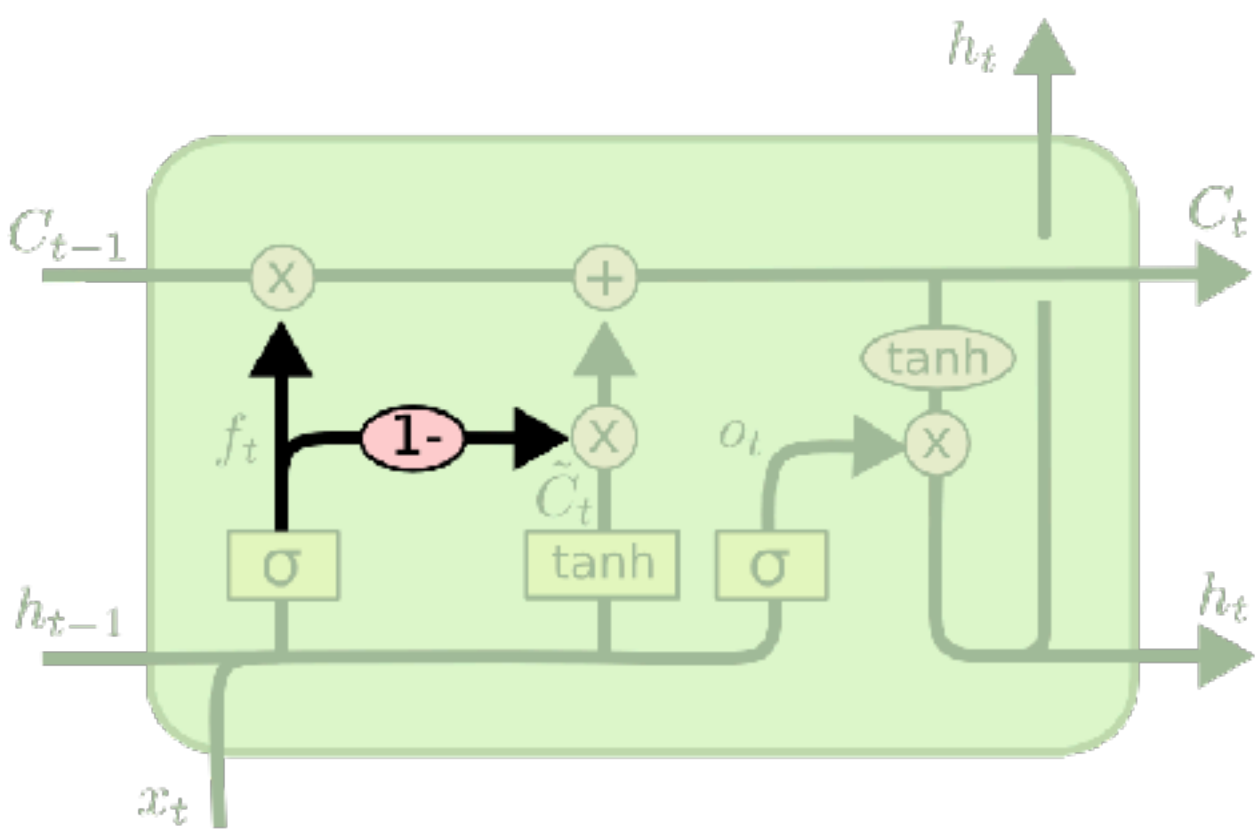
- We let the gate layers look at the cell state



$$f_t = \sigma\left(W_f \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \; + \; b_f\right)$$
$$i_t = \sigma\left(W_i \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \; + \; b_i\right)$$
$$o_t = \sigma\left(W_o \cdot [\boldsymbol{C_t}, h_{t-1}, x_t] \; + \; b_o\right)$$

Variants: Coupled forget & input gates

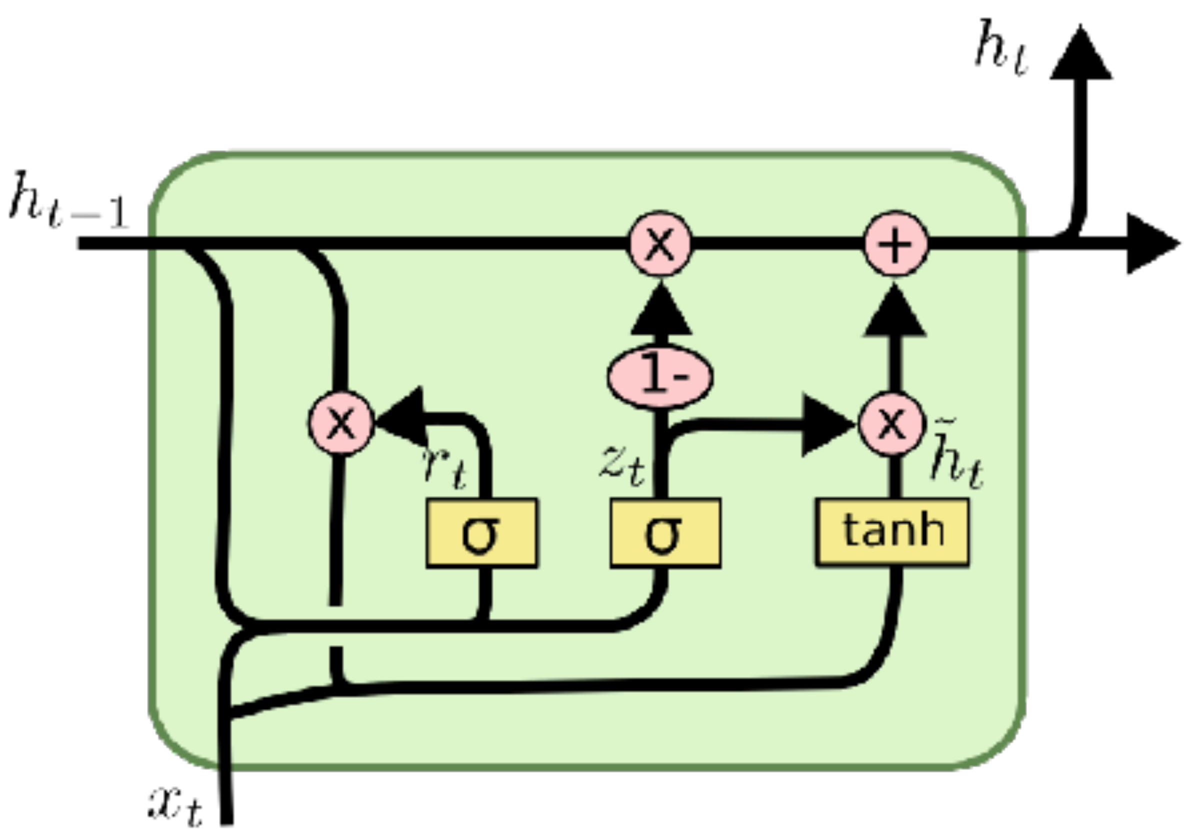
- Instead of separately deciding what to forget and what we should add new information to, we make those decision together.



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

Variants: Gated Recurrent Unit, or GRU

- It combines the forget and input gates into a single "update gate", it also merges the cell state and hidden state, makes some other changes.
- Resulting model is simpler than standard LSTM, and growing increasingly popular.



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Thanks

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

To be continued…

Show and Tell: Lessons Learned from the 2015 MSCOCO Image Captioning Challenge

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan