Recurrent Neural Networks

What have we covered so far?

- Feed Forward Neural Networks
- Auto Encoders
- Convolutional Neural Networks

What are all of these missing?

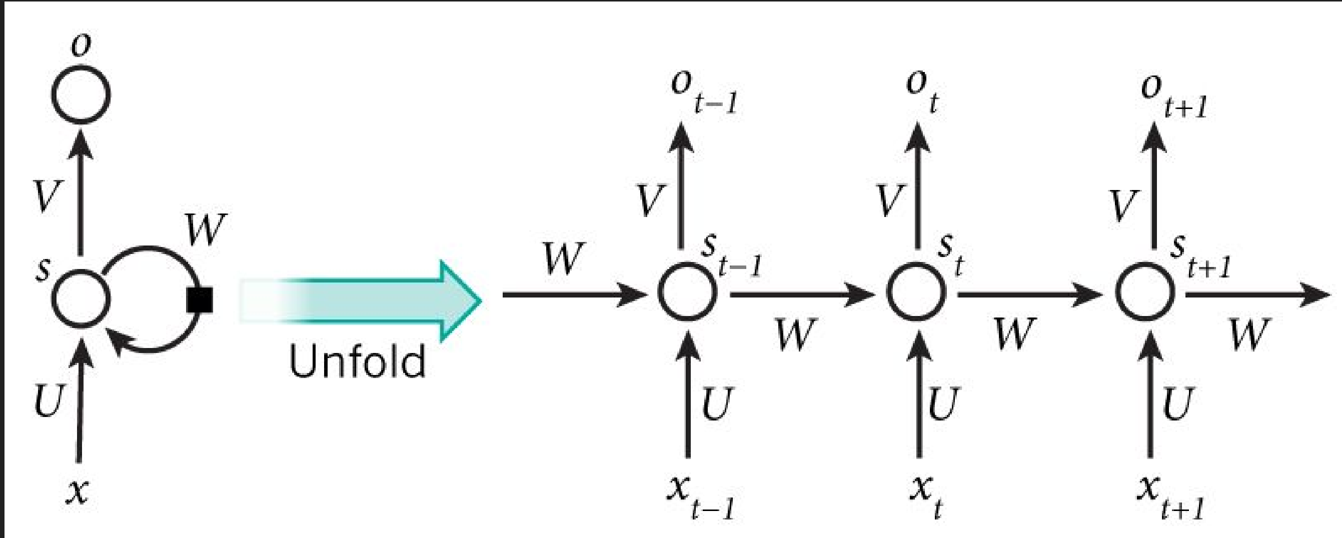
What was in the reading?

Memory

- Each iteration of the networks we’ve discussed deals *only* with the input immediately present
- How do we make them remember?

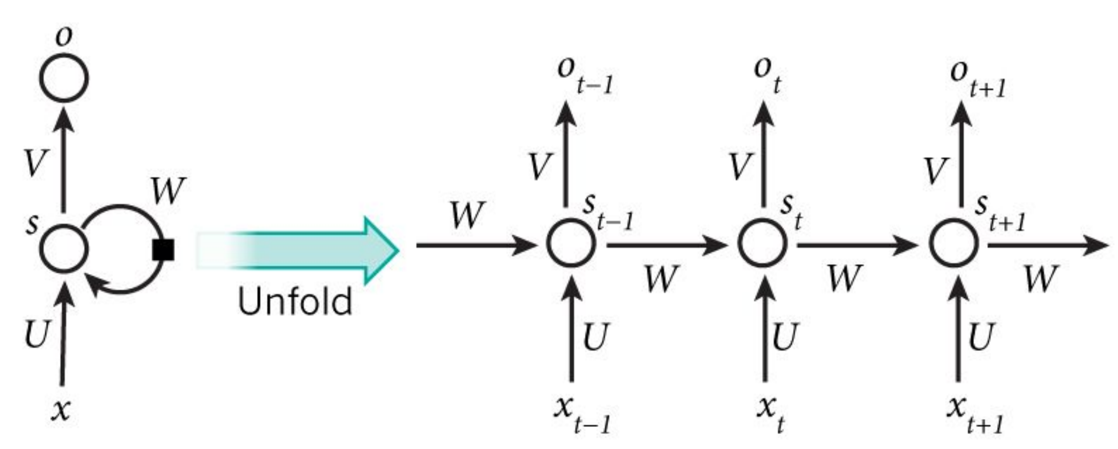
Recurrent Neural Networks

- The network takes *two* inputs and produces *two* outputs
- **Inputs**: data input; previous iteration's hidden state output
- **Outputs**: regular output; hidden state for next iteration



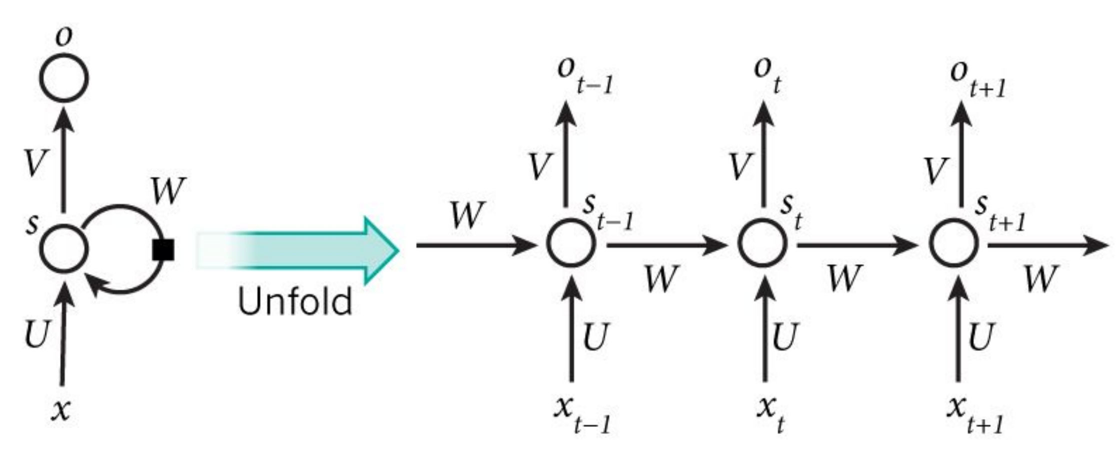
The Hidden State

- Hidden state contains network's 'memory'
- How does backpropagation work?
- How do we handle having two inputs?
- How do we handle having two outputs?



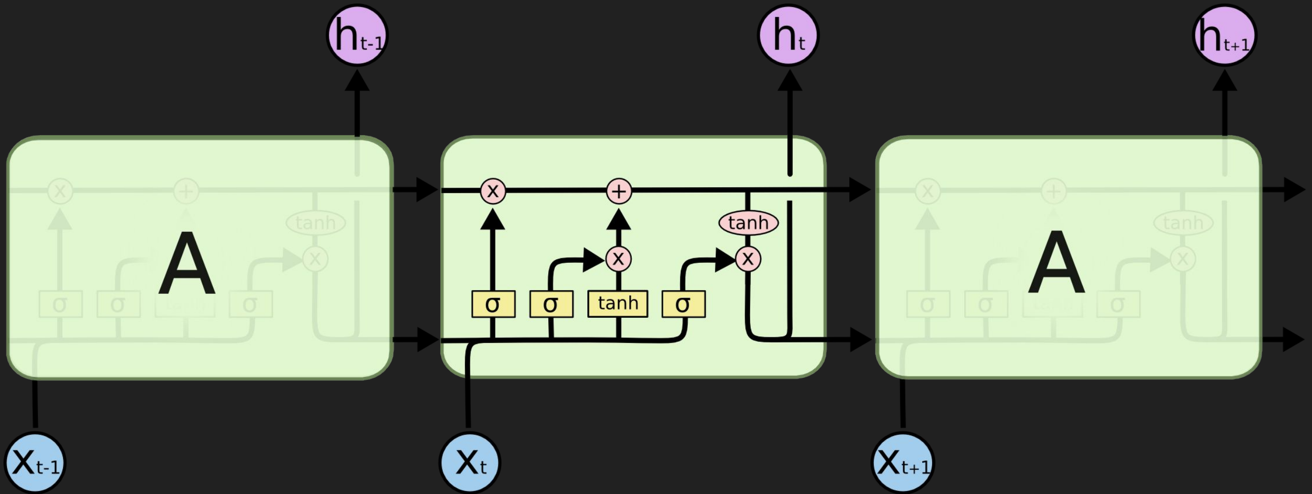
But we have a problem

- What is the relationship between o_{50} and x_1?
- Chain rule on chain rule on chain rule
- Gradients vanishing and exploding, can't actually learn long-term dependencies



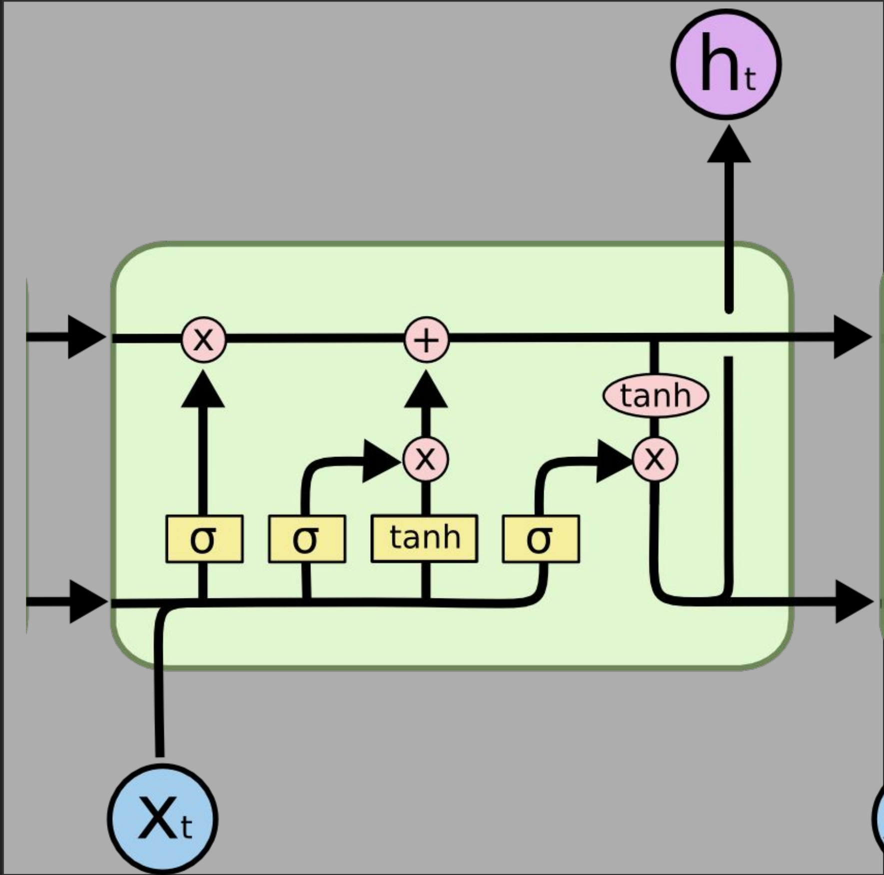
LSTMs fix this issue

- [Long Short-Term Memory networks]
- LSTMs have several gates that maintain their hidden state
  - This overcomes the *vanishing* gradient problem
- These gates are the forget gate, input gate and output gate



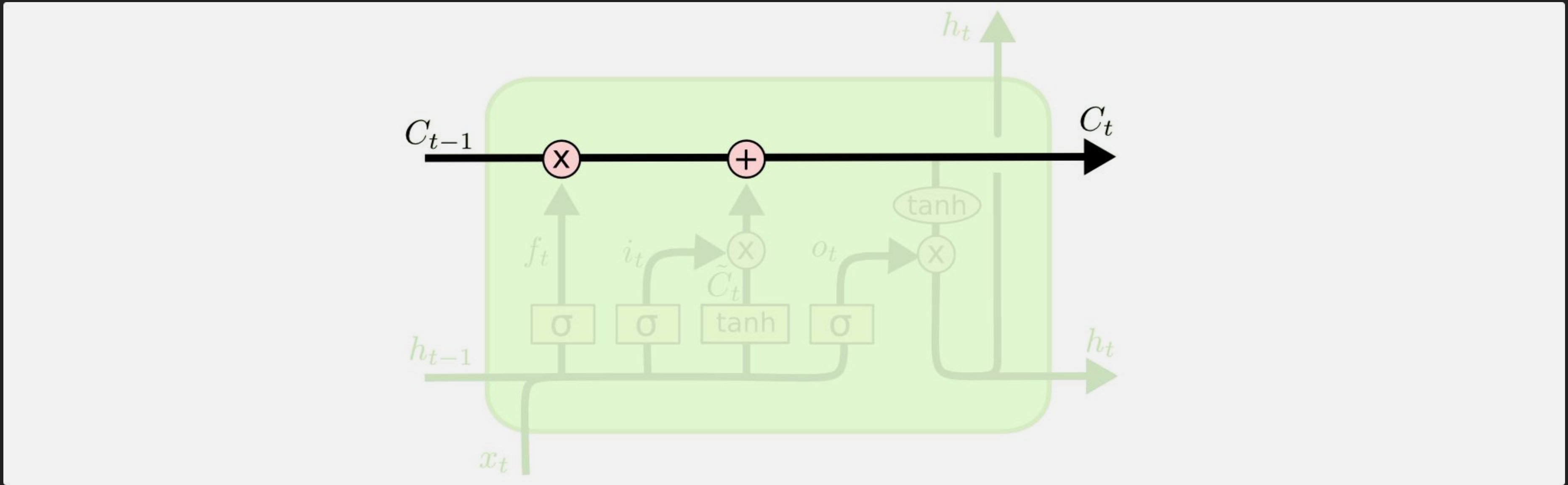
We'll break this down step by step

- The x and + are element-wise operations
- The σ and tanh are activation functions
- The x_t and h_t are the input and output respectively
- *The top line* is the ***cell state***
    - This is the 'memory'
- The LSTM takes in *input, previous output, cell state*



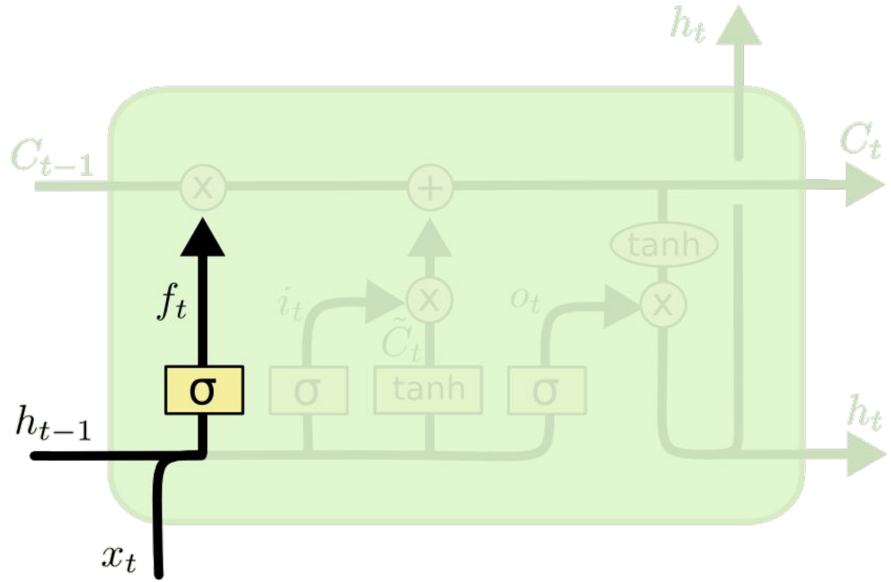
The cell state

- The cell state is the memory within our LSTM
- It only has minor linear interactions, and no activations on it
  - That's what makes the LSTM avoid vanishing gradients



The forget gate

- A gate is a way to optionally let information through
- The forget gate concatenates the previous output and current input, and performs sigmoid activation on it (range of 0-1)

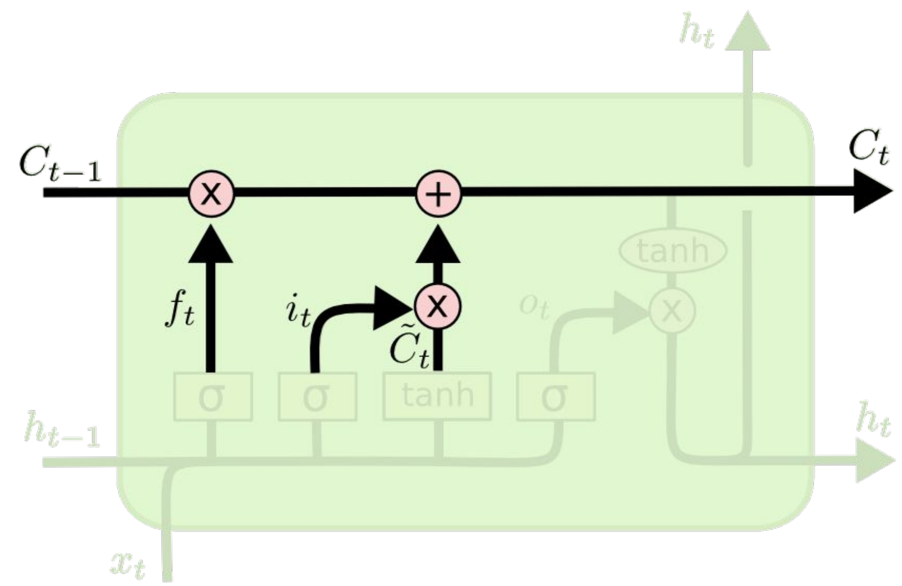


$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \ + \ b_f\right)$$

F_t = forget range ~ (0, 1)

We *multiply* C_{t-1} element-wise by f_t

- Since f_t is between 0 and 1, this means we either
  - Forget old information if f_t is 0
  - Pass old information through completely if f_t is 1

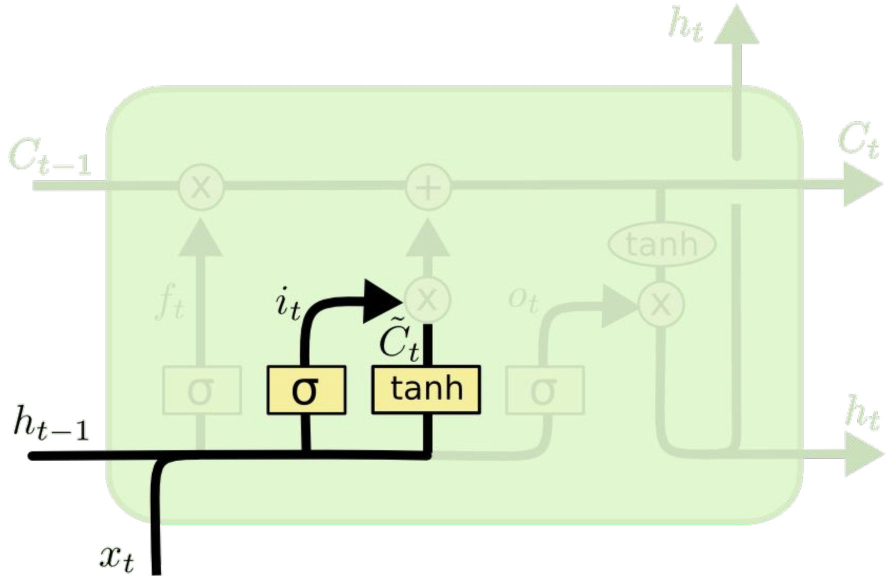


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

So dC_t / dC_{t-1} is just f_t

The input gate

- Now we want to update *new* information into the cell state
- i_t decides which values we will update
- $\hat{C}_t$ is how much we want to update them by

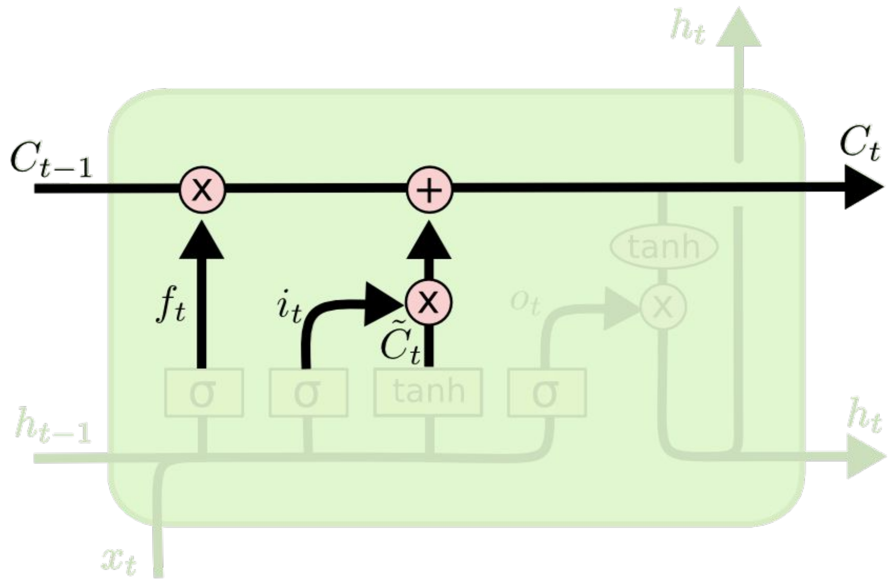


$$i_t = \sigma\left(W_i\cdot[h_{t-1}, x_t] \ + \ b_i\right)$$

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

We *add* $i_t \times \hat{C}_t$ to our cell state

- This is the new information we want the LSTM to remember

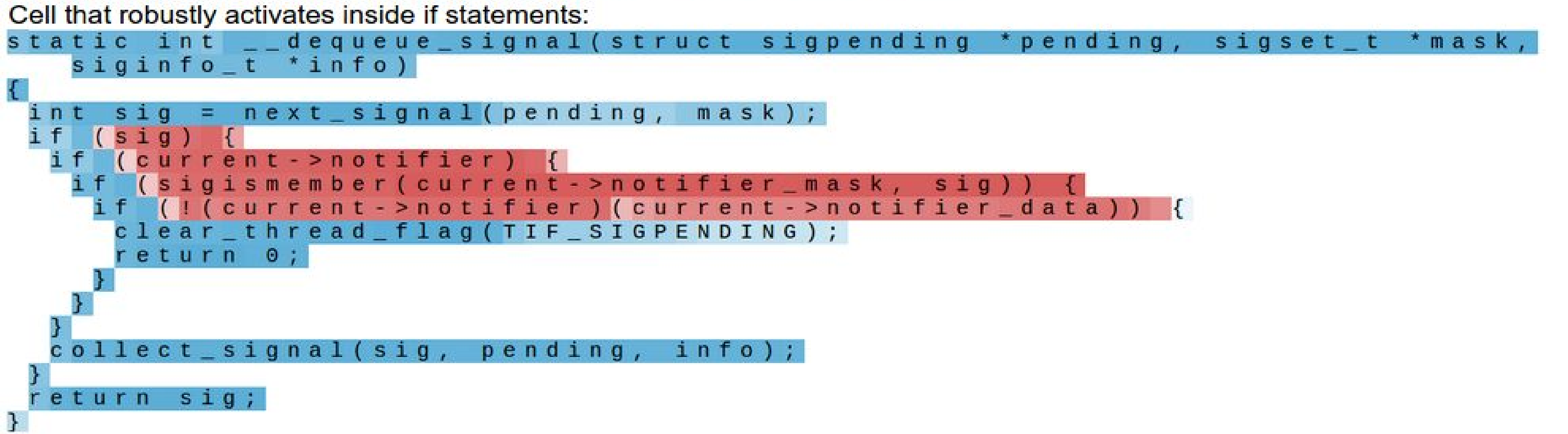


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

So dC_t / dC_{t-1} is just f_t

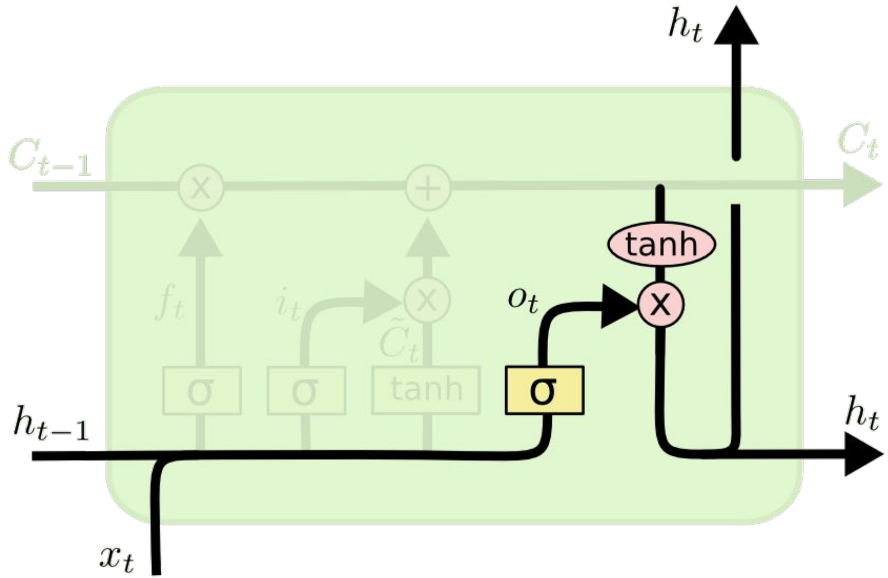
Forgetting and Adding New Information Example

- If we’re writing C code, we would need a neuron in the cell state that is ‘on’ inside if statements
- When the if statement ends, *forget* old cell state and *add* new information



The output gate

- We tanh our cell state so its range = (-1, 1)...gives consistent output
- Sigmoid layer times this chooses what we do and don't output
  - Similar to forget gate, we only send some things to output

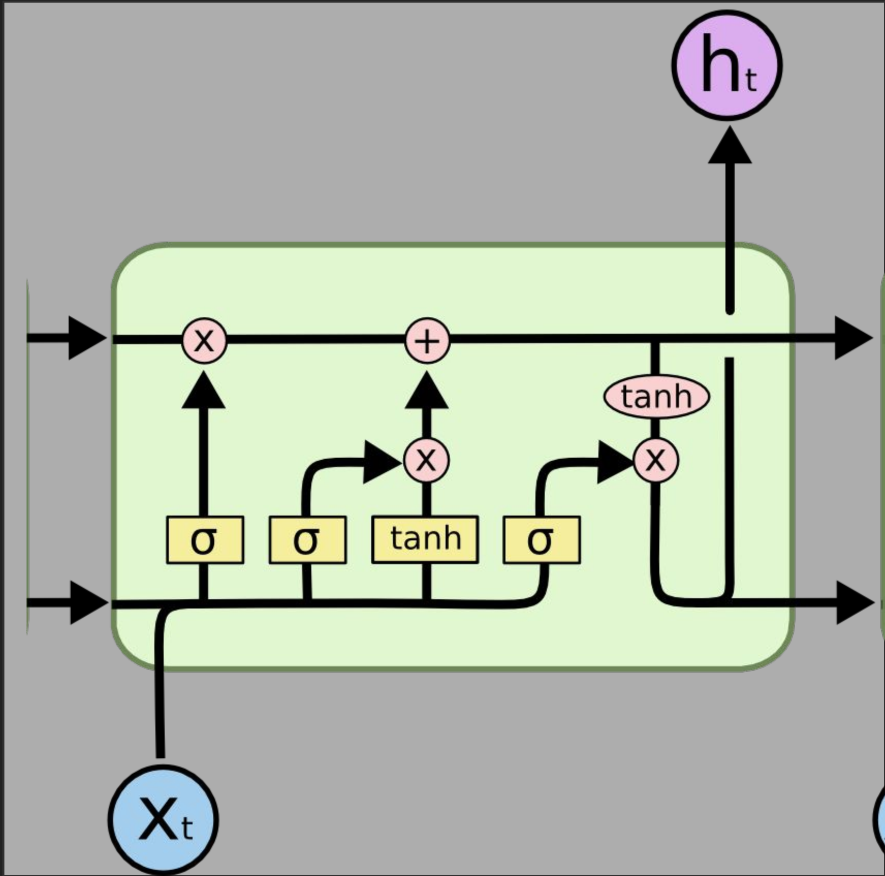


$$o_t = \sigma\left(W_o\ [h_{t-1}, x_t]\ +\ b_o\right)$$

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

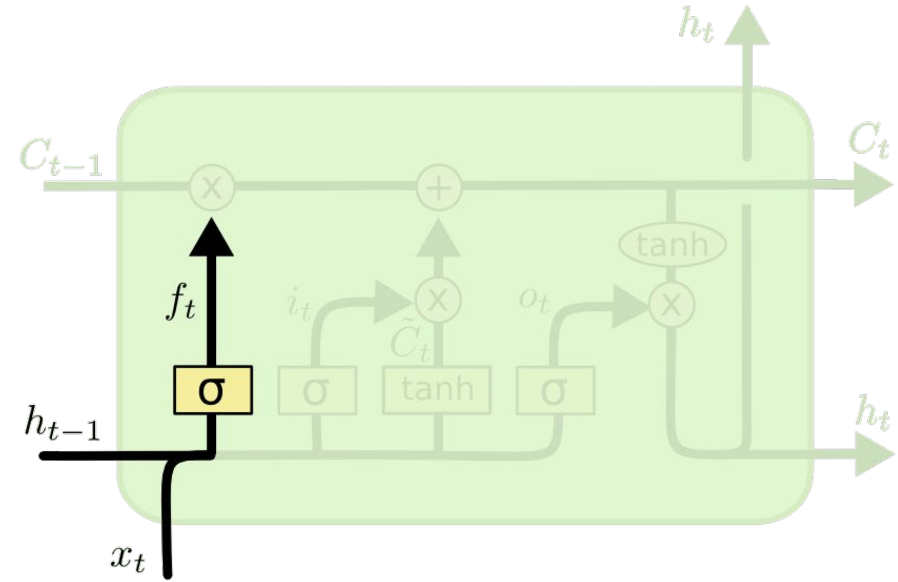
Putting everything together

- LSTM takes in the *input, previous output* and *cell state*
- It forgets parts of the cell state at the forget gate
- Adds new info to the cell state at the input gate
- Then chooses what from the cell state to pass as output at the output gate



Making the dimensionality work out

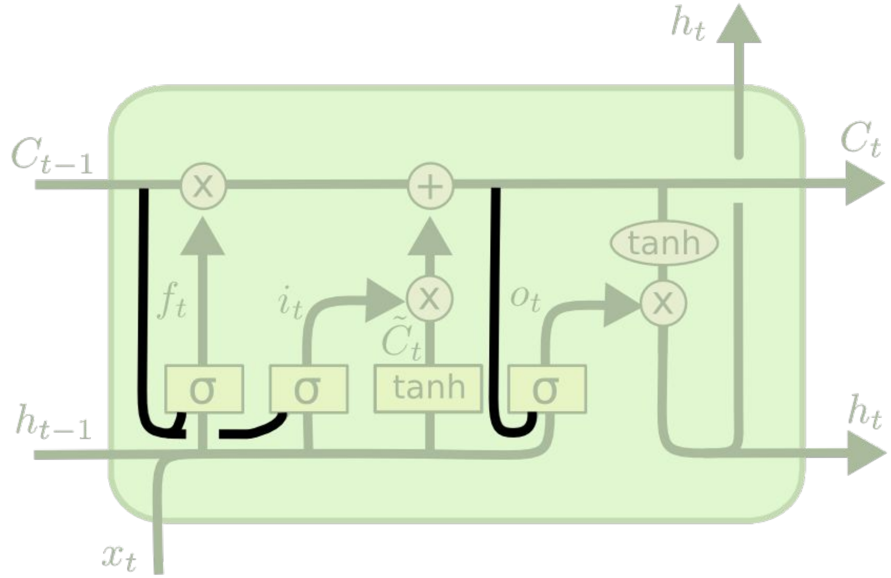
- What size are the output, cell state and input vectors?
- How do we concatenate and activate them so they add up?



$$f_t = g(W_{xf}x_t + W_{hf}h_{t-1} + b_f)$$

So we add peepholes

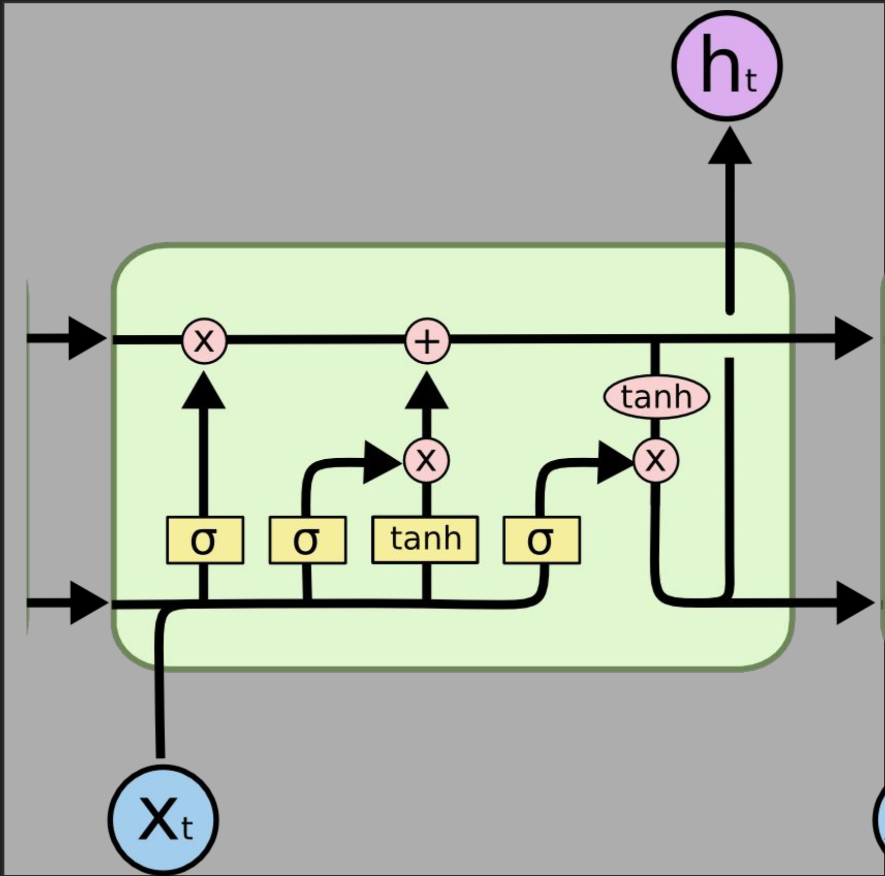
- Peepholes give the sigmoid state insight as to what it's actually forgetting



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

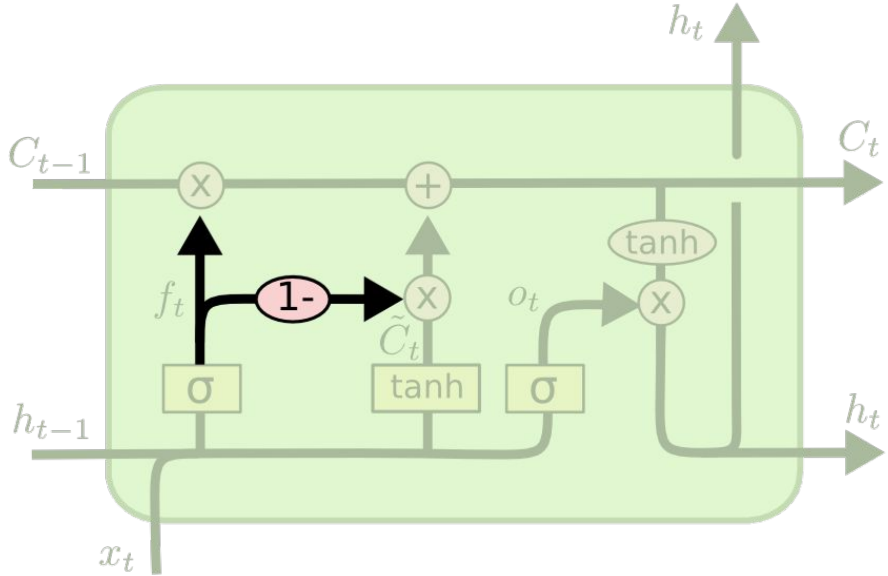
But the sigmoid gates can’t see the cell state?...

- Sigmoid gates choose what to forget without seeing what the cell state actually knows
- How do we fix this?



But what if we forget but don’t input in its place?

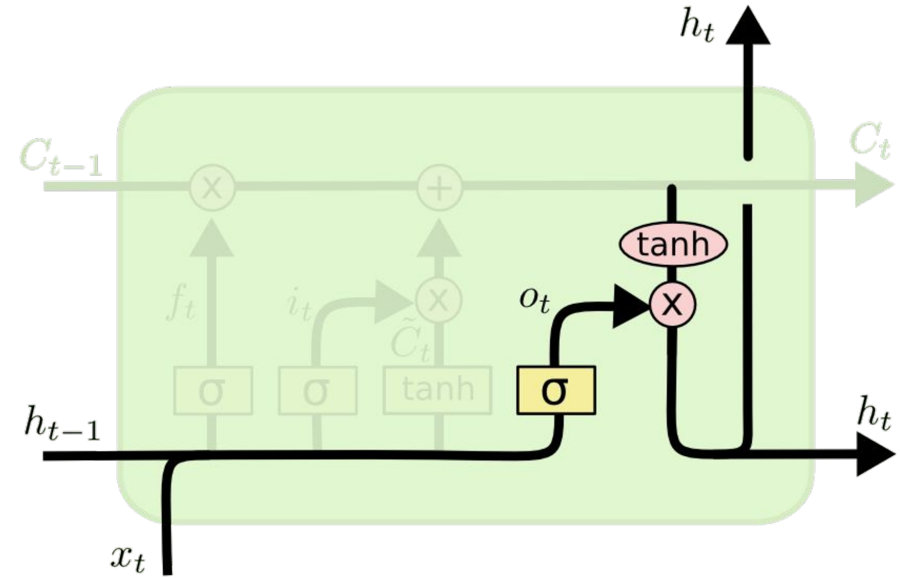
- Couple the forget and input gates so that they act in tandem



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

What do we do with the LSTM output?

- We do a few layers of a regular feed forward network to interpret the output of the LSTM

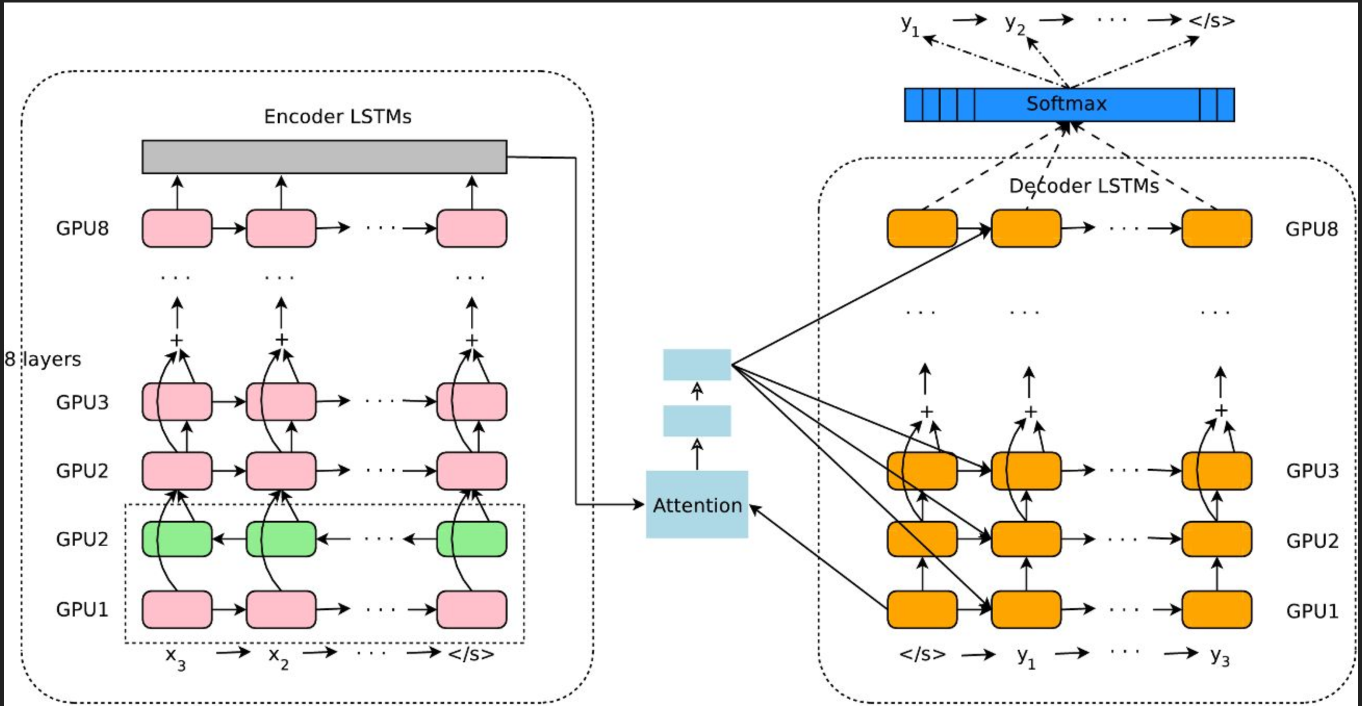


$$o_t = \sigma\left(W_o \left[h_{t-1}, x_t\right] + b_o\right)$$

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

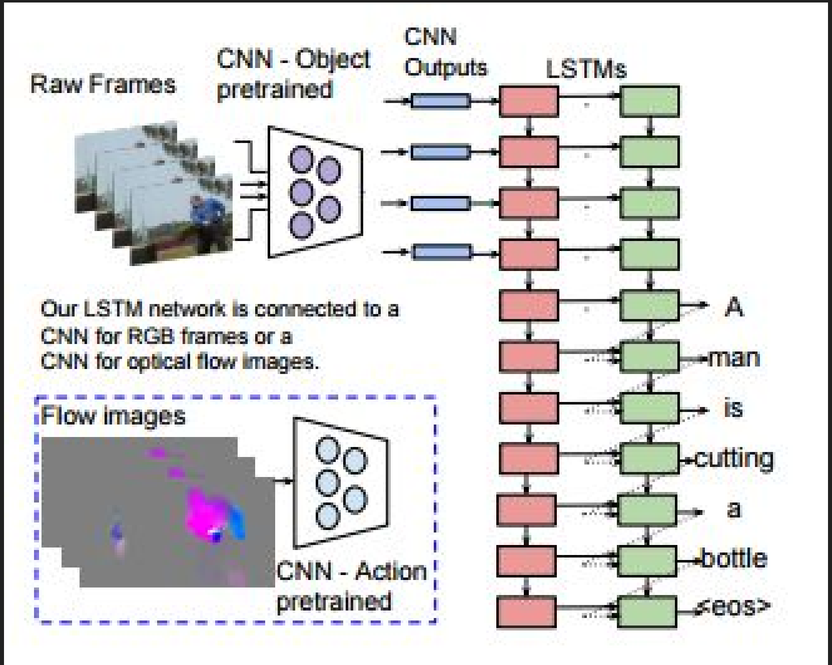
Chaining LSTM Cells

- What passes between LSTM cells and what is unique to each?



Example Application

- LSTMs are useful anytime you need memory of past states
- This paper describes how to train a computer to describe a video in words.
- Convolutions to understand images, LSTMs to see the connections between frames



The rest of the semester

- We've talked about the necessary neural network architectures
- These are the four papers which we will read in this order:
    - ImageNet Classification with Deep Neural Networks
    - Dynamic Pooling and Unfolding Recursive Autoencoders for Paraphrase Detection
    - Pixel Recursive Super Resolution
    - Mastering the Game of Go with Deep Neural Networks and Tree Search

Your Job

- Choose one of the papers, and turn in a two-page summary of the paper on the day it is being discussed
- Come ready to help answer any questions if it’s your paper’s week
- You must read every paper! Only write a summary for one
- I will structure these lectures as me asking questions about why the researchers made the decisions they did