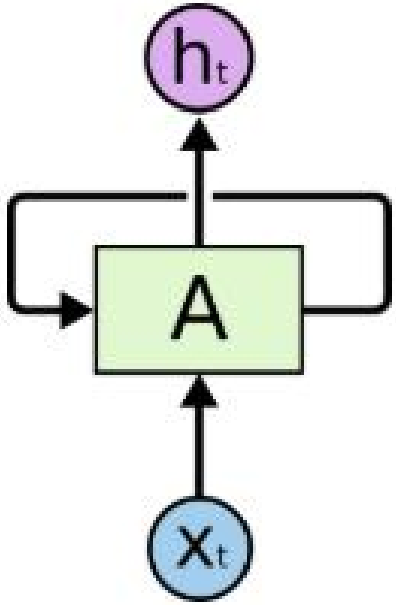
LSTM

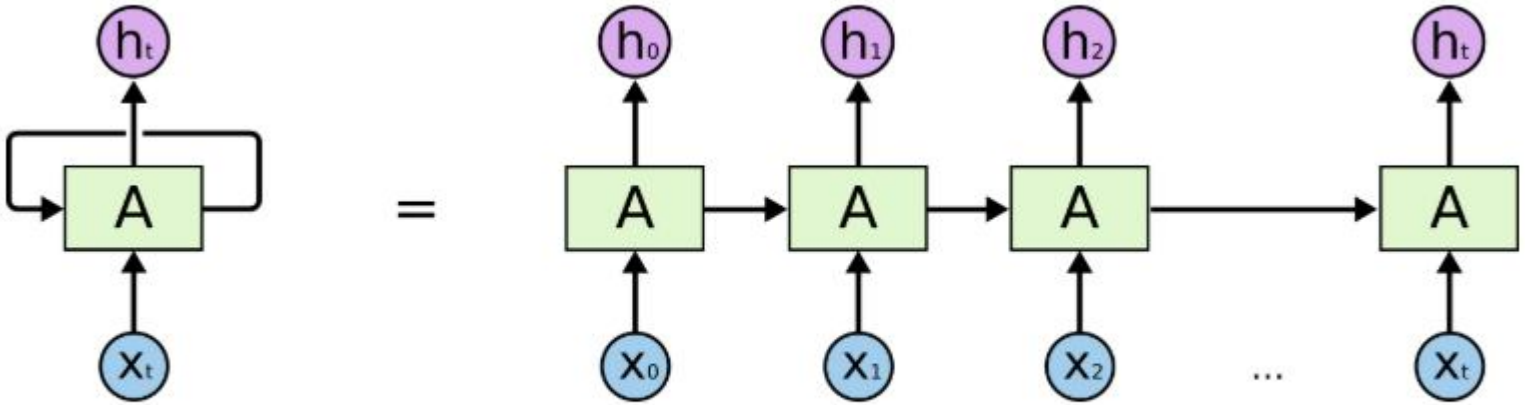
RNN

- Recurrent neural networks ,are networks with loops in them, allowing information to persist.

*a chunk of neural network, **A**,*
*looks at some input **xt** and*
*outputs a value **ht**.*
A loop allows information to be passed from one step of the network to the next.



- A recurrent neural network can be thought of as multiple copies of the same network, each passing a message to a successor.

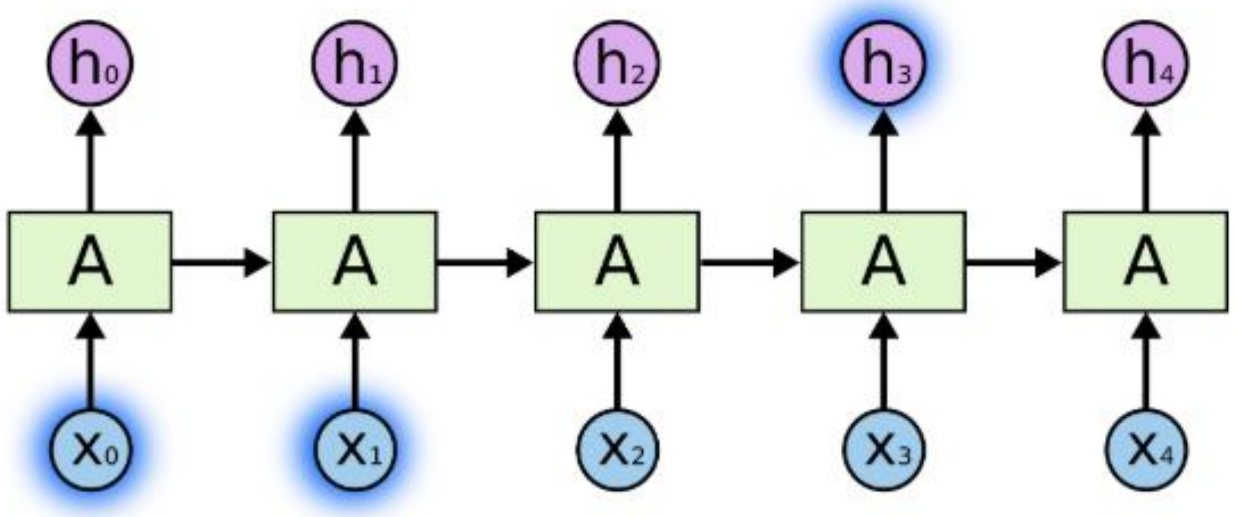


An unrolled recurrent neural network.

The Problem of Long-Term Dependencies

- Sometimes, we only need to look at recent information to perform the present task.
- If we are trying to predict the last word in

  "the clouds are in the *sky*," we don't need any further context – it's pretty obvious the next word is going to be **sky**.

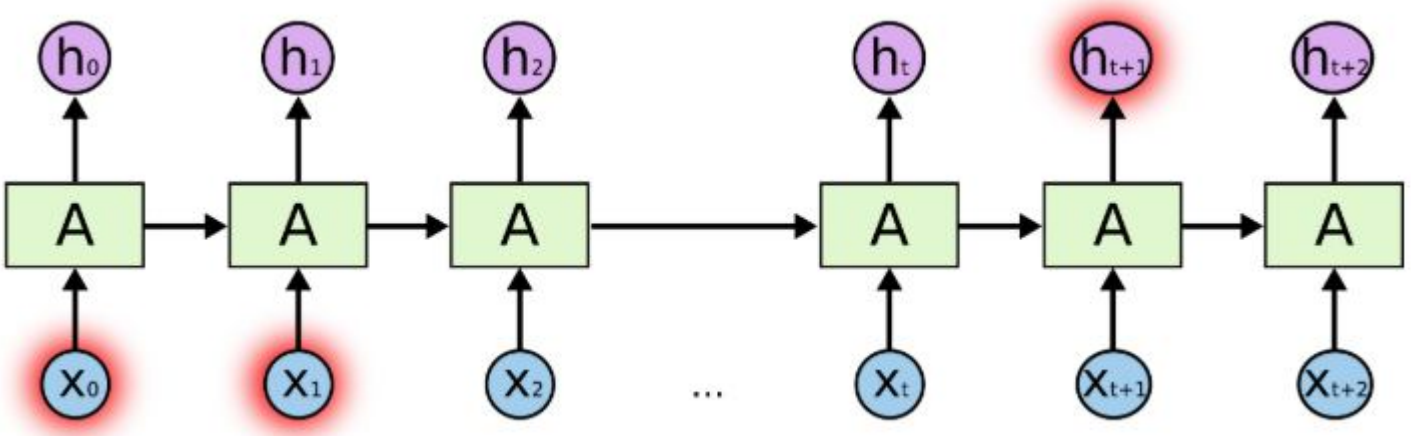


- But there are also cases where we need more context.
- Consider trying to predict the last word in the text

“I grew up in France… I speak fluent *French*.”

Recent information suggests that the next word is probably the name of a language, but if we want to narrow down which language, we need the context of France, from further back.

Unfortunately, as that gap grows, RNNs become unable to learn to connect the information

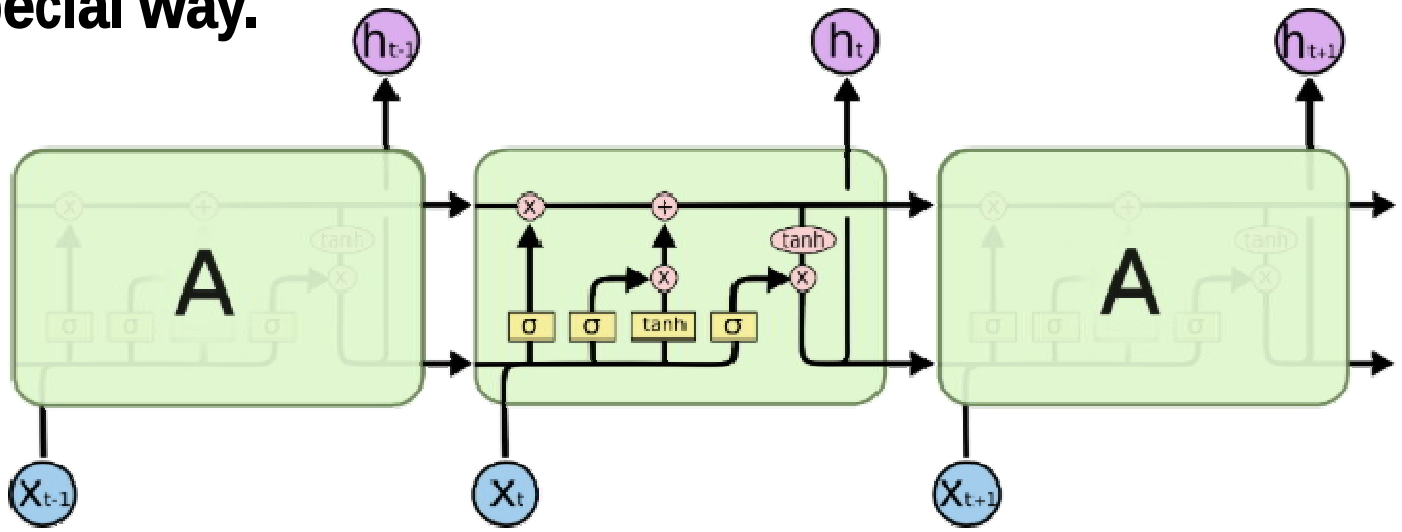


LSTM Networks

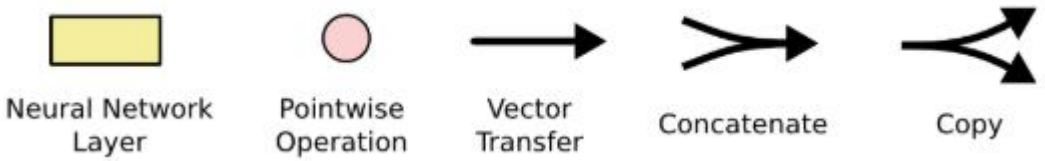
- Long Short Term Memory networks (**LSTM**) are a special kind of RNN, capable of learning long-term dependencies.
- LSTMs are explicitly designed to avoid the long-term dependency problem.
- Remembering information for long periods of time is practically their default behaviour, not something they struggle to learn!

LSTM Architecture

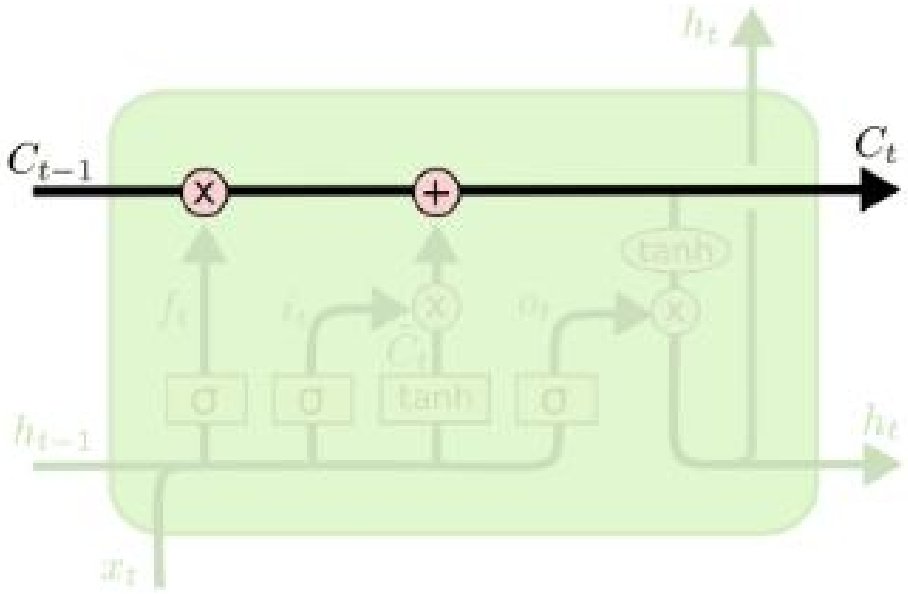
- **LSTMs also have this chain like structure**
- **There are four neural network layer, interacting in a very special way.**



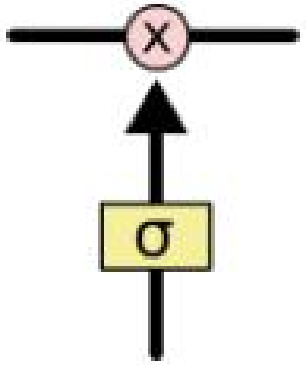
The repeating module in an LSTM contains four interacting layers.



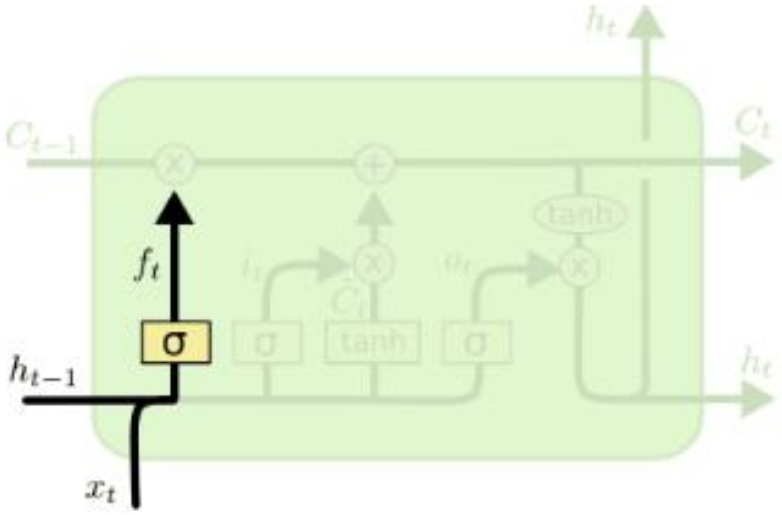
- The key to LSTMs is the cell state, the horizontal line running through the top of the diagram.
- The cell state is kind of like a conveyor belt.



- The LSTM does have the ability to remove or add information to the cell state, carefully regulated by structures called gates.
- Gates are a way to optionally let information through.
- They are composed out of a sigmoid neural net layer and a pointwise multiplication operation.

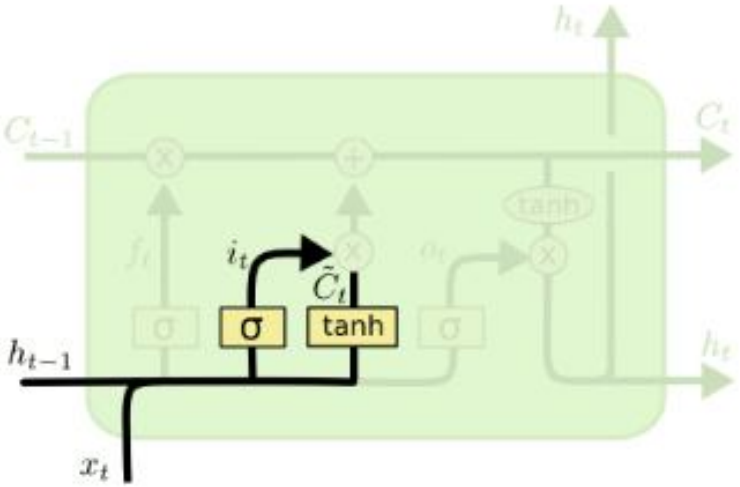


- The first step in our LSTM is to decide what information we're going to throw away from the cell state. This decision is made by a sigmoid layer called the **"forget gate layer."**
- A 1 represents **"completely keep this"** while a 0 represents **"completely get rid of this."**



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

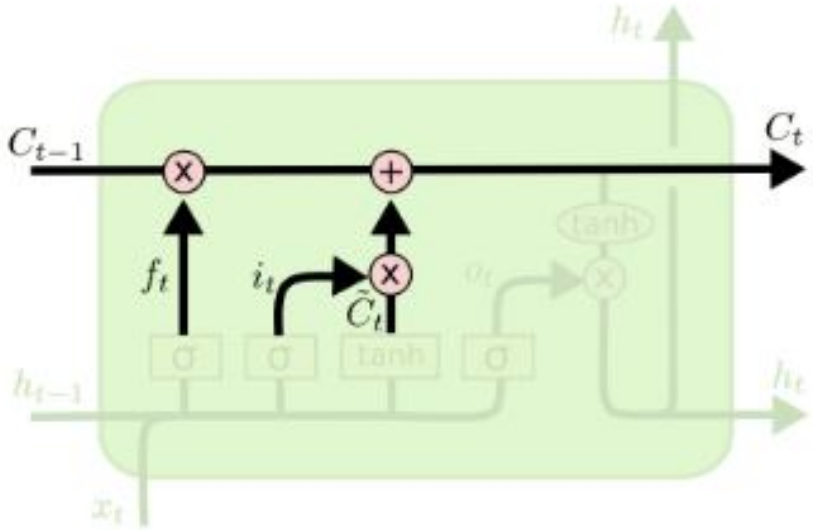
- The next step is to decide what new information we're going to store in the cell state. This has two parts.
- First, a sigmoid layer called the **"input gate layer"** decides which values we'll update.
- Next, a tanh layer creates a vector of new candidate values, C~t, that could be added to the state.



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

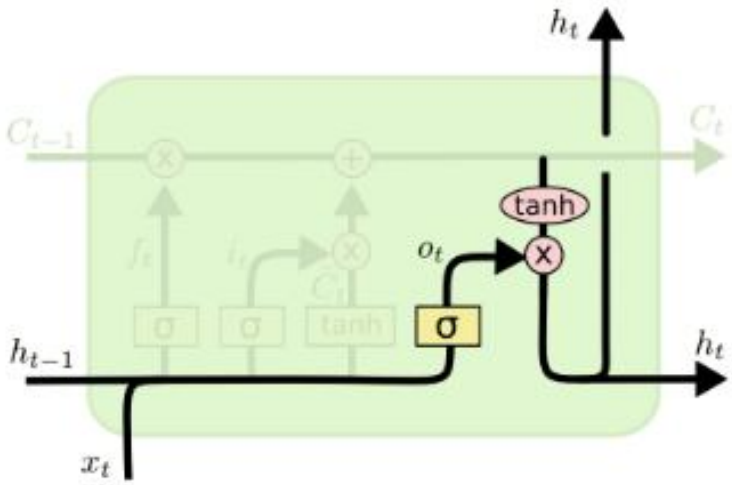
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

- update the old cell state, Ct−1, into the new cell state Ct



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

- we need to decide what we're going to output.
- First, we run a sigmoid layer which decides what parts of the cell state we're going to output.
- Second, we put the cell state through tanh
- multiply 1st & 2nd step so that we only output the parts we decided to.



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

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