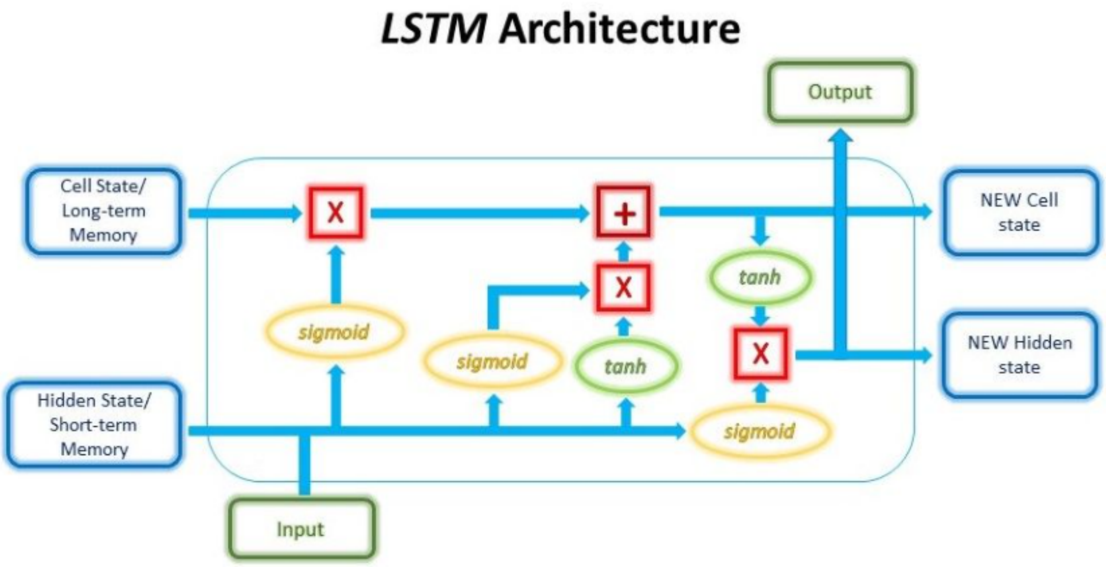
Understanding LSTMs

LSTM vs RNN

The key difference is in the information these cells are able to store. RNNs only take two pieces of information: a hidden state and an input. LSTMs however are able to take in three different pieces of information: current input, short-term memory from the previous cell (hidden states in RNNs), and long-term memory.

In terms of terminology, short-term memory is referred to as the "hidden state", and long-term memory is referred to as the "cell state". Each LSTM cell uses gates to regulate information and determine if something should be kept or discarded. Ideally, the role of these gates is to selectively remove irrelevant information. Think of how water filters prevent impurities from passing through. The difficulty is training these gates to accurately filter the useful and the irrelevant.

There are three gates: **Input**, **Forget**, and **Output**.



The Input Gate

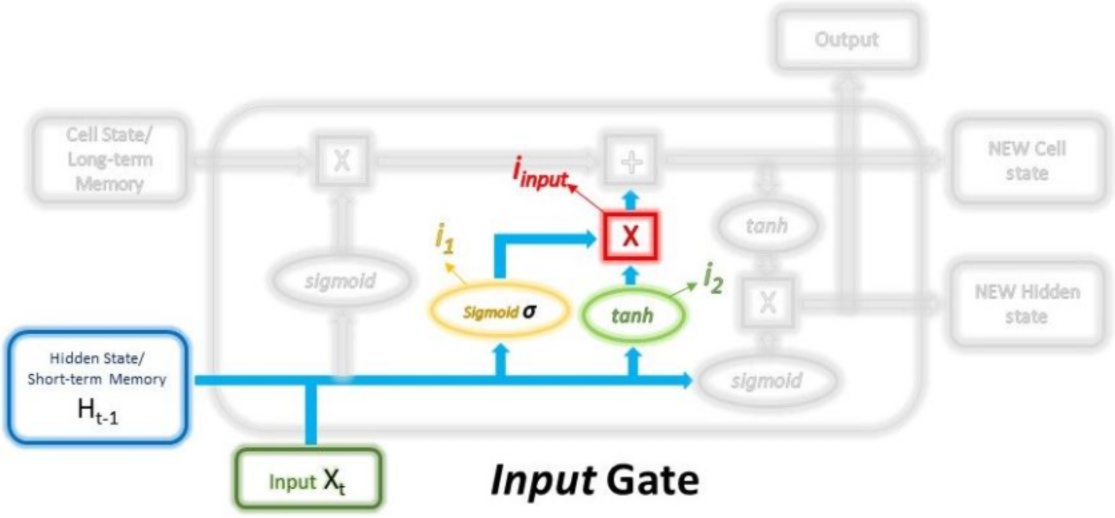
Input gate decides what new information will be stored in the long-term memory. It only works with the current input and the short term memory from the previous step. There are two separate layers at play, i_1 and i_2. In i_1, information is selected from the hidden state and the input to be added into the long term memory, done through a sigmoid activation function. During back propagation, the weights in the sigmoid function are updated such that it learns to only let the useful pass while discarding less critical features.

$$i_1 = \sigma(W_{i_1} \cdot (Ht-1, x_t) + bias_{i_1})$$

The second layer takes the hidden state and the input and passes it through an activation function, typically $tanh$, to regulate the network.

$$i_2 = tanh(Wi_2 \cdot (H_{t-1}, x_t) + biasi_2)$$

The results from these two inputs are multiplied together with it's final outcome representing the information to be kept in the cell state (long-term memory).

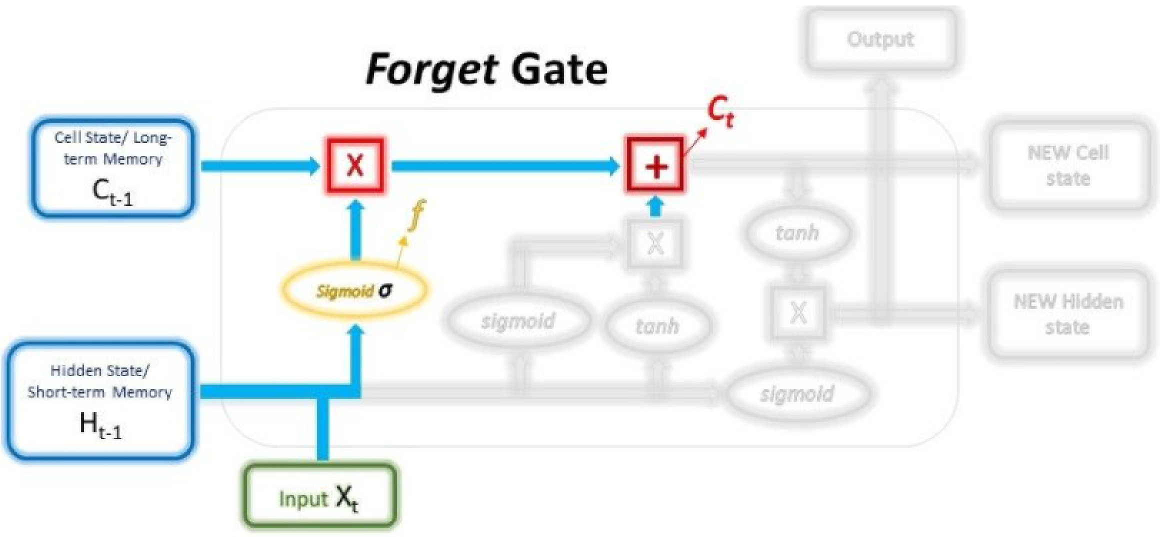


Forget Gate

Forget gate decides the information which will be discarded from the cell state. This is done by multiplying the incoming cell state by something called a **forget vector**, generated by the current input and incoming hidden layer. The **forget vector** is also a selective filter layer. This will be done with a sigmoid function with different weights than those in the input gate filters. The result will be a vector of 0s and 1s and will be multiplied with long-term memory to choose which parts of the long-term memory is to be retained.

$$f = \sigma(W_{forget} \cdot (H_{t-1}, x_t) + bias_{forget})$$

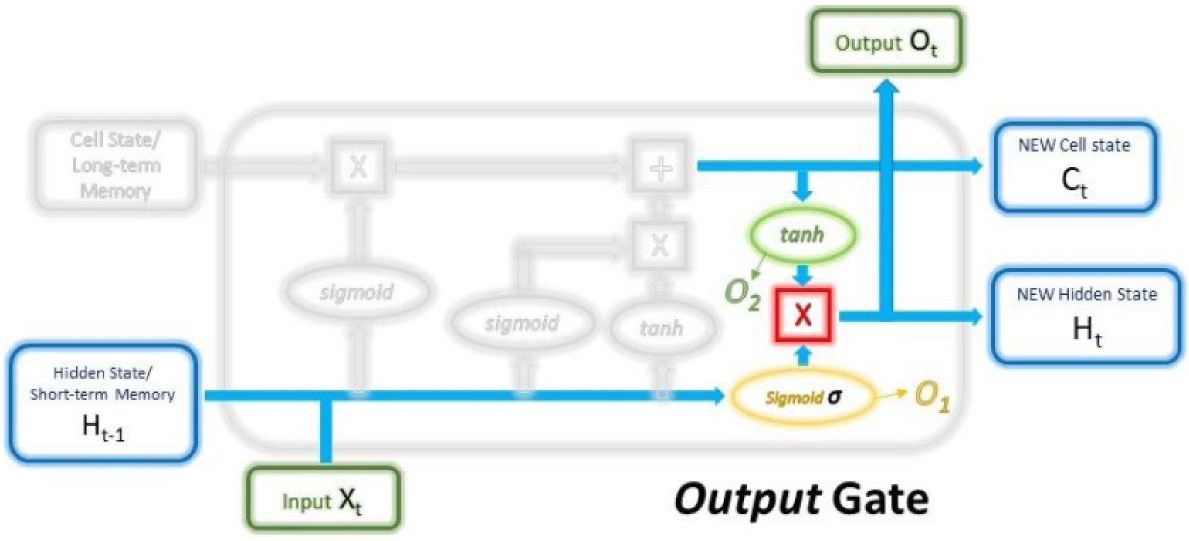
The outputs from the **Input gate*** and the **Forget gate** will be summed through pointwise addition to determine an updated cell state (long-term memory) to be passed on to the next cell. This value will also be used in the **Output gate**.



Output gate

Output gate understandably determines the output. It does so using the current input, previous hidden state, and the newly computed cell state. The results from the output gate are the output as well as the new hidden state. The process is split into two different sections.

O_1 takes in the new cell state and passes it through a $tanh$ activation function while O_2 takes the hidden state and input and passes it through another sigmoid filter. The results are multiplied together to produce the new hidden state. The output of the cell is the same value as the new hidden state?



Classification Types

There are generally two types of classification when using LSTMs, or RNNs in general.

Many to One

In this instance, we only require the final output from the last cell and generate some answer based on this value. Think along the lines of sentiment analysis, or subjectivity analysis. The last output is then fed into some classifier and a final label is received.

Many to Many

In this instance we require an output at every time step and extract information to feed into a fully connected layer. Think text generation or music transcription.