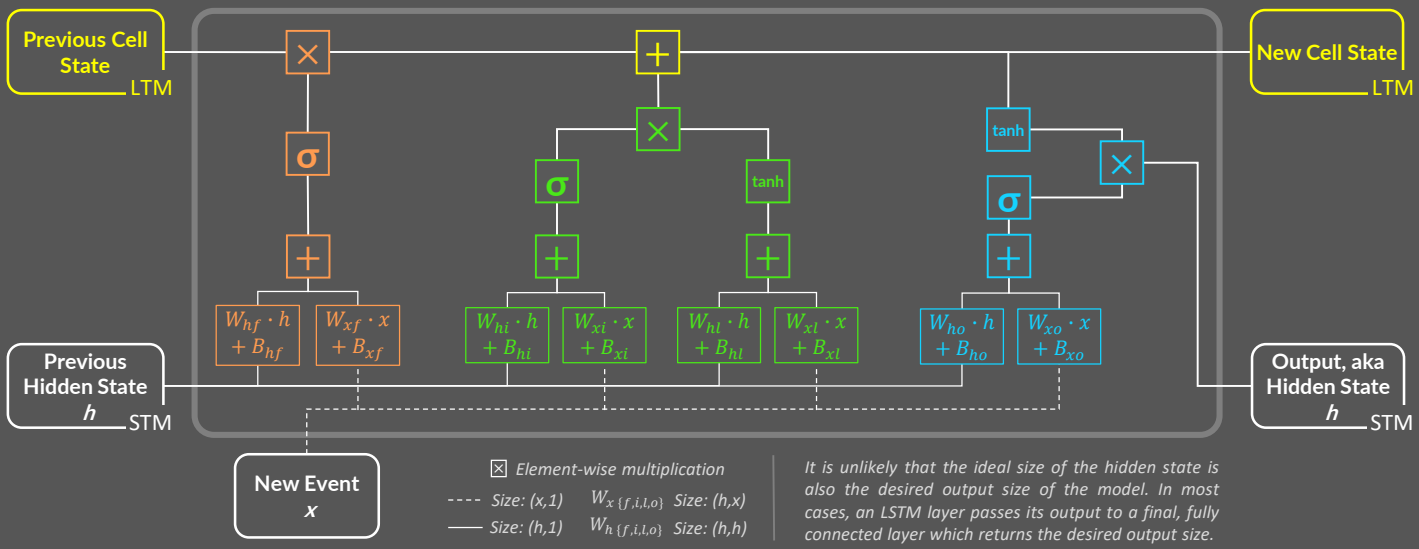
The LSTM Reference Card



It is unlikely that the ideal size of the hidden state is also the desired output size of the model. In most cases, an LSTM layer passes its output to a final, fully connected layer which returns the desired output size.

Forget Gate

The Forget Gate is a way to selectively forget some of what the Cell State (LTM) has in memory. The New Event and the previous period's Hidden State are summed (element-wise) and then transformed with a sigmoid function. This output is therefore a vector with entries between 0 and 1. When the Previous Cell State is multiplied (element-wise) with this vector, the effect is that some proportion (between 0 and 1) of each value in the Previous Cell State makes it "through the gate" and is retained; the rest is forgotten.

```
import numpy as np
from scipy.special import expit as sigmoid

def forget_gate(x, h, Weights_hf, Bias_hf, Weights_xf, Bias_xf, prev_cell_state):
    forget_hidden  = np.dot(Weights_hf, h) + Bias_hf
    forget_eventx  = np.dot(Weights_xf, x) + Bias_xf
    return np.multiply( sigmoid(forget_hidden + forget_eventx), prev_cell_state )
```

Input Gate (aka Learn Gate)

The Input Gate has 2 components: a way to "Ignore" new information, and a way to "Learn" new information. In each case, the New Event and previous period's Hidden State are summed and transformed. The Ignore component is transformed using logic similar to the Forget Gate: a sigmoid function creates a vector of proportions (values between 0 and 1). The Learn component uses a hyperbolic tangent function, which returns a vector with values between -1 and 1; this helps the model learn both positive and negative relationships in the data. When the Learn component is multiplied (element-wise) by the Ignore component, the effect is that some proportion of each value from the Learn component makes it "through the gate" and is retained; the rest is ignored.

```
def input_gate(x, h, Weights_hi, Bias_hi, Weights_xi, Bias_xi, Weights_hl, Bias_hl, Weights_xl, Bias_xl):
    ignore_hidden  = np.dot(Weights_hi, h) + Bias_hi
    ignore_eventx  = np.dot(Weights_xi, x) + Bias_xi
    learn_hidden   = np.dot(Weights_hl, h) + Bias_hl
    learn_eventx   = np.dot(Weights_xl, x) + Bias_xl
    return np.multiply( sigmoid(ignore_eventx + ignore_hidden), np.tanh(learn_eventx + learn_hidden) )
```

Cell State (aka Long Term Memory)

The Cell State at each time is calculated by adding two things together: the vector from the Forget Gate, and the vector from the Input Gate. The Cell State is used in the Output Gate (below) to determine the model's current output; it's also carried forward to be used for the next Event's forward pass.

```
def cell_state(forget_gate_output, input_gate_output):
    return forget_gate_output + input_gate_output
```

Output Gate (aka Use Gate)

The Output Gate returns a vector that is both the model's output for that Event, and the new hidden state *h* (STM), which is carried forward to the next Event's forward pass. The Cell State, previous Hidden State, and New Event all contribute to this vector: the New Event and previous Hidden State are combined and multiplied (element-wise) by the transformed Cell State.

```
def output_gate(x, h, Weights_ho, Bias_ho, Weights_xo, Bias_xo, cell_state):
    out_hidden = np.dot(Weights_ho, h) + Bias_ho
    out_eventx = np.dot(Weights_xo, x) + Bias_xo
    return np.multiply( sigmoid(out_eventx + out_hidden), np.tanh(cell_state) )
```