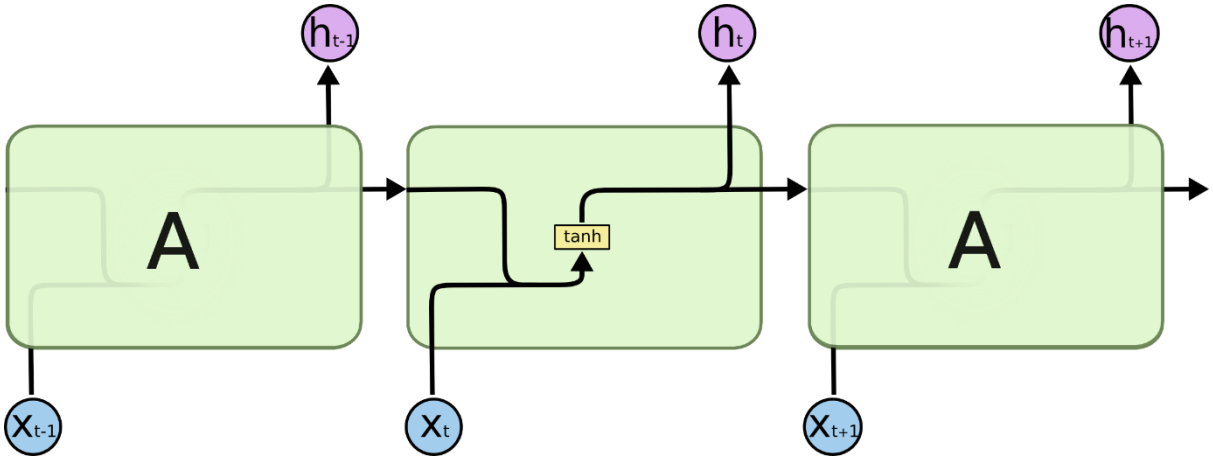
LSTM (Long-Short Term Memory) RNNs

Recurrent Neural Network Cell



Recurrent Neural Network(RNN) are a type of Neural Network where the output from previous step are fed as input to the current step. The main and most important feature of RNN is Hidden state, which remembers some information about a sequence. RNN have a "memory" which remembers all information about what has been calculated. It uses the same parameters for each input as it performs the same task on all the inputs or hidden layers to produce the output. This reduces the complexity of parameters, unlike other neural networks.

Advantages of Recurrent Neural Network

1. An RNN remembers each and every information through time. It is useful in time series prediction only because of the feature to remember previous inputs as well. This is called Long Short-Term Memory.
2. Recurrent neural network are even used with convolutional layers to extend the effective pixel neighbourhood.

Submitted By: Ritwik Sharma [15MI427]
Gajendra Babu [15MI431]

Major Disadvantage of Recurrent Neural Network

The RNN cannot predict the word stored in the long-term memory but can give more accurate predictions from the recent information. As the gap length increases RNN does not give efficient performance. Basically, RNN have poor memory.

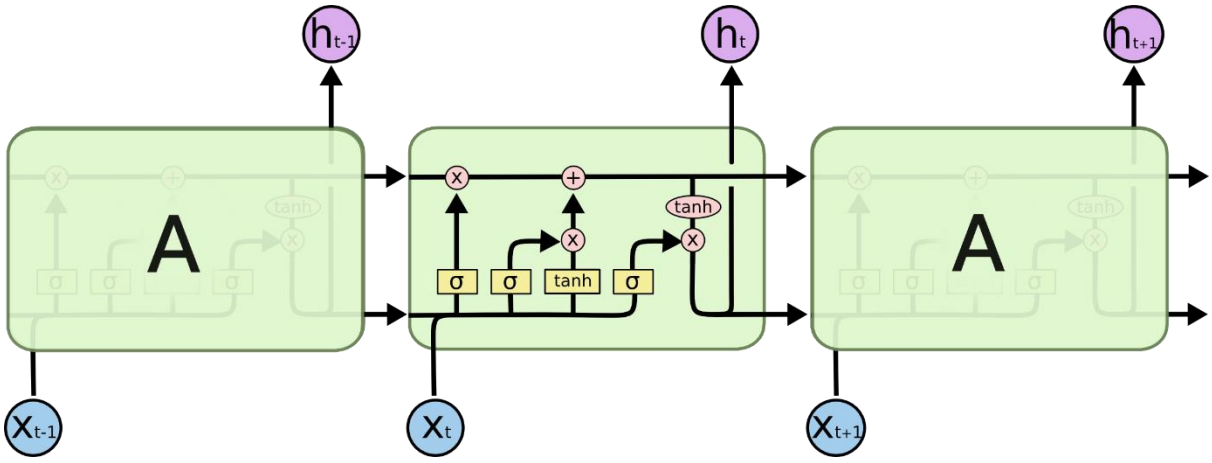
To overcome this poor memory problem the concept of LSTM (Long-Short Term Memory) RNNs was introduced.

LSTM RNNs

LSTM can by default retain the information for long period of time. It is used for processing, predicting and classifying on the basis of time series data.

Structure Of LSTM

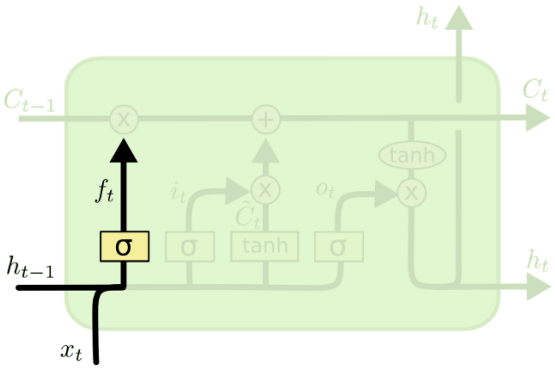
LSTM has a chain structure that contains four neural networks and different memory blocks called cells.



Information is retained by the cells and the memory manipulations are done by the gates. There are three gates:

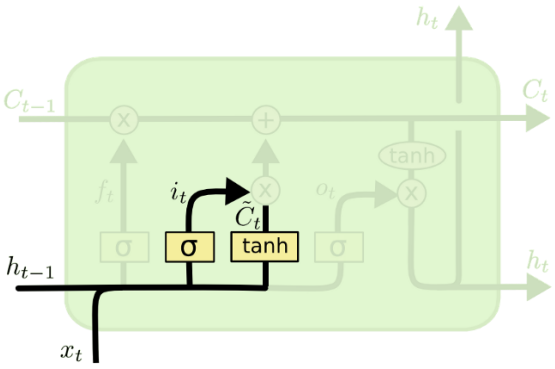
1. **Forget Gate**: The information that no longer useful in the cell state is removed with the forget gate. Two inputs *x_t* (input at the

particular time) and *h_t-1* (previous cell output) are fed to the gate and multiplied with weight matrices followed by the addition of bias. The resultant is passed through an activation function which gives a binary output. If for a particular cell state the output is 0, the piece of information is forgotten and for the output 1, the information is retained for the future use.



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

2. **Input Gate**: Addition of useful information to the cell state is done by input gate. First, the information is regulated using the sigmoid function and filter the values to be remembered similar to the forget gate using inputs *h_t-1* and *x_t*. Then, a vector is created using *tanh* function that gives output from -1 to +1, which contains all the possible values from h_t-1 and *x_t*. Atlast, the values of the vector and the regulated values are multiplied to obtain the useful information.

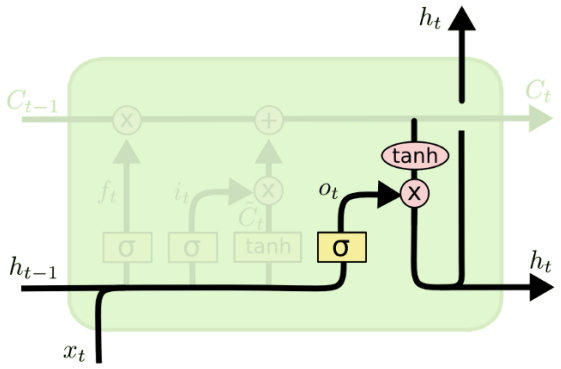


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

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

3. **Output Gate**: The task of extracting useful information from the current cell state to be presented as an output is done by output gate. First, a vector is generated by applying tanh function on the cell.

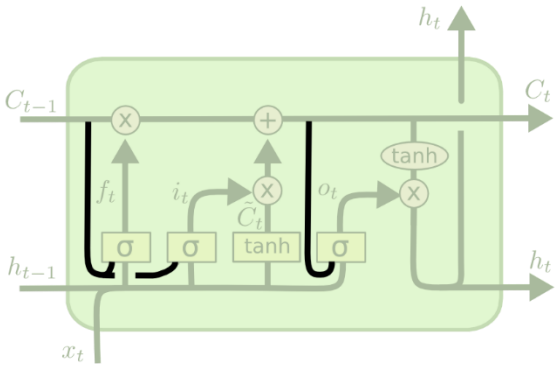
Then, the information is regulated using the sigmoid function and filter the values to be remembered using inputs *h_t-1* and *x_t*. Atlast, the values of the vector and the regulated values are multiplied to be sent as an output and input to the next cell.



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

Variants on Long Short-Term Memory

1. **Peephole Connections**: Introduced by Gers & Schmidhuber (2000), we add peepholes to let the gate layers look at the cell state.

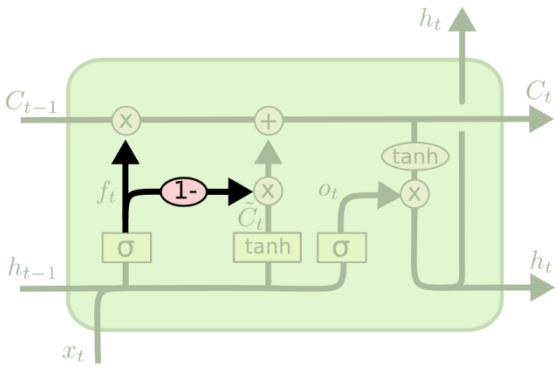


$$f_t = \sigma\left(W_f \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] + b_f\right)$$
$$i_t = \sigma\left(W_i \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] + b_i\right)$$
$$o_t = \sigma\left(W_o \cdot [\boldsymbol{C_t}, h_{t-1}, x_t] + b_o\right)$$

The above diagram adds peepholes to all the gates, but many papers will give some peepholes and not others.

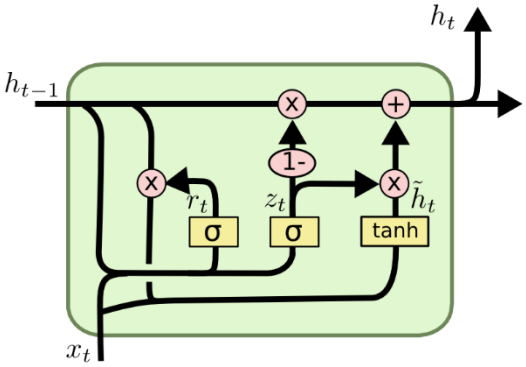
2. **Coupled Gates**: Another variation is to use coupled forget and input gates. Instead of separately deciding what to forget and what we should add new information to, we make those decisions together. We only forget when we're going to input something in its place. We only input new values to the state when we forget something older.

Submitted By: Ritwik Sharma [15MI427]
Gajendra Babu [15MI431]



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

3. **Gated Recurrent Unit**: Introduced by Cho, et al. (2014). It combines the forget and input gates into a single "update gate." It also merges the cell state and hidden state, and makes some other changes. The resulting model is simpler than standard LSTM models, and has been growing increasingly popular.



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

These are only a few of the most notable LSTM variants. There are lots of others, like Depth Gated RNNs by Yao, et al. (2015). There's also some completely different approach to tackling long-term dependencies, like Clockwork RNNs by Koutnik, et al. (2014).

Submitted By: Ritwik Sharma [15MI427]
Gajendra Babu [15MI431]