CHITKARA UNIVERSITY

Long Short Term Memory (LSTM)

Long Short Term Memory

- **L**ong **S**hort-**T**erm **M**emory (LSTM) is one of the most widely used recurrent structures in sequence modeling. It uses gates to control information flow in the recurrent computations.
- LSTM networks are very good at holding long term memories. The memory may or may not be retained by the network depending on the data. Preserving the long term dependencies in the network is done by its Gating mechanisms. The network can store or release memory on the go through the gating mechanism.

Need of LSTM's

- Recurrent Neural Networks suffer from short-term memory. If a sequence is long enough, they have a hard time carrying information from earlier time steps to later ones. So if you are trying to process a paragraph of text to do predictions, RNN's may leave out important information from the beginning.
- During back propagation, recurrent neural networks suffer from below problems:
  - Vanishing Gradients
  - Exploding Gradients

- Vanishing Gradients occurs when the values of a gradient are too small and the model stops learning or takes way too long because of that.
- Exploding Gradients occurs when the algorithm assigns a stupidly high importance to the weights, without much reason.
- This issue can be resolved by applying a slightly tweaked version of RNNs – the Long Short-Term Memory Networks and Gated Recurrent units.

LSTM Gates

The cell state keeps track of several pieces of information, and it is much more complicated than the hidden state. Let us take one LSTM cell. LSTM is built on three major principles.

1. While building the sequential models, we need not carry forward all the information to the next time steps. We can forget some information if it is unnecessary. Of course, that depends on the training data.

2. While building the sequential models, we need not take all the information at each time step into consideration. We can limit the amount of information that is being written into the memory cell. Again, whether to write the full information into the memory cell or not is decided based on historical training data.

3. While building sequential models, we need not output full information from the memory cell at every time step. Depending on the time step, we may want to limit how much amount of data can be read from the memory cell. Once again, how much information can be read from a particular memory cell is decided based on historical training data.

To incorporate these three points, we add a few more nodes in the memory cell. There are already hidden nodes and input nodes in each cell. We incorporate the above three points in each cell. The first point is about forgetting or retaining the information from the previous cell.

LSTM Architecture

- Here is the internal functioning of the LSTM network.

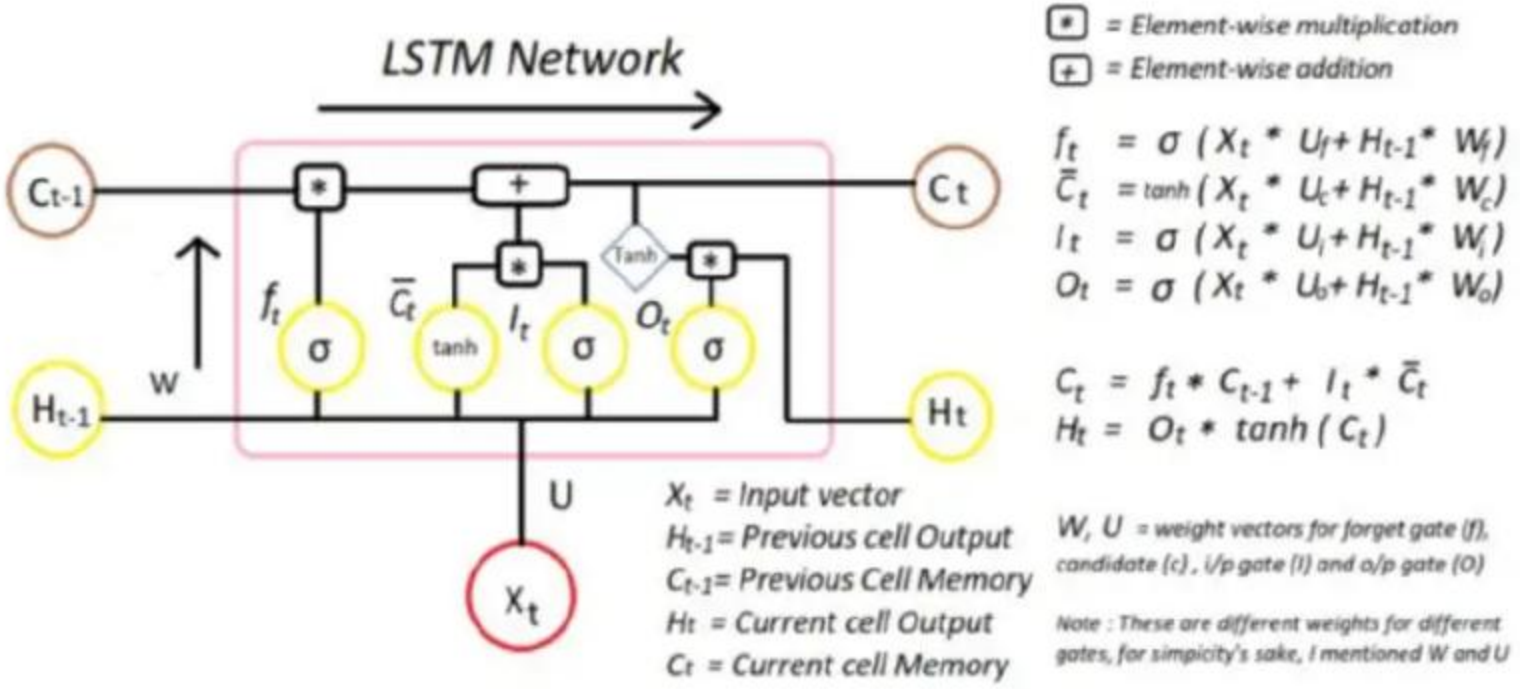


Fig: LSTM cell

Working of LSTM's

- The main three parts of an LSTM cell are known as gates. The first part is called **Forget gate, the** second part is known as **the Input gate** and the last one is **the Output gate**.
- The first gate is called the **forget gate**. If you shut it, no old memory will be kept. If you fully open this gate, all old memory will pass through. It is actually an element-wise multiplication operation. So if you multiply the old memory with a vector that is close to 0, that means you want to forget most of the old memory. If you want to let the old memory go through, forget gate should be equal to 1.

- The second gate is the **input gate**. Exactly how much new input should come in is controlled by the second gate. This gate controls how much the new memory should influence the old memory.
- Next is + **operator**. This operator means piece-wise summation. Current input and the old memory will merge by this operation. This is element-wise summation to merge the old memory and the current input to form **St**.
- We need to generate the output for this LSTM unit. This step has an **output gate** that is controlled by the new memory, the previous output & the current input. This gate controls how much new memory should output to the next LSTM unit.

Long Short Term Memory (LSTM)

- A small example where RNN can work perfectly :
  - Prediction of the last word in the sentence : "The clouds are in the sky"

- RNN can't handle situation where the **gap** between the **relevant information** and the point where it is needed is **very large**.

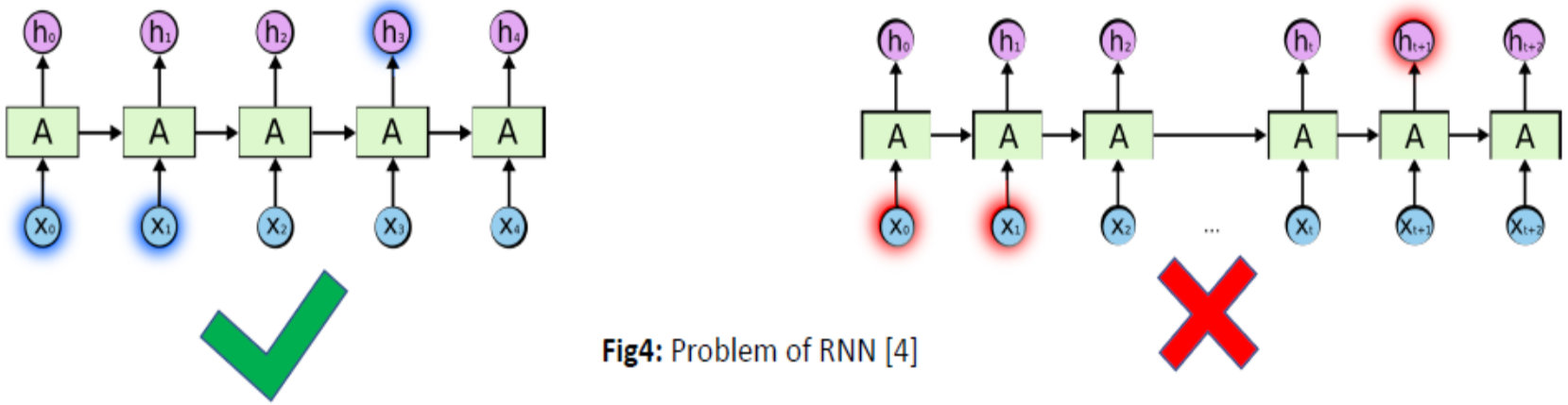


Fig4: Problem of RNN [4]

- LSTM can !

Long Short Term Memory (LSTM)

- **Long Short Term Memory networks** – usually just called "**LSTMs**" – are a special kind of RNN, capable of learning **long-term dependencies**. *Hochreiter & Schmidhuber (1997)*

- All recurrent neural networks have the form of a **chain of repeating modules** of neural network. In standard RNNs, this repeating module will have a very simple structure, such as a single tanh layer.

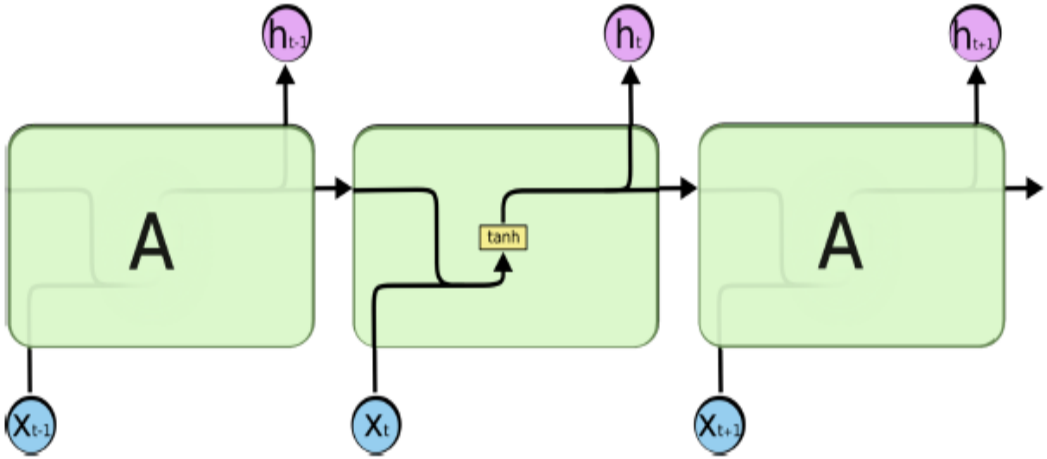


Fig5: The repeating module in a standard RNN contains a single layer [4]

Long Short Term Memory (LSTM)

- **LSTM** have the same chain like structure except for the repeating module.

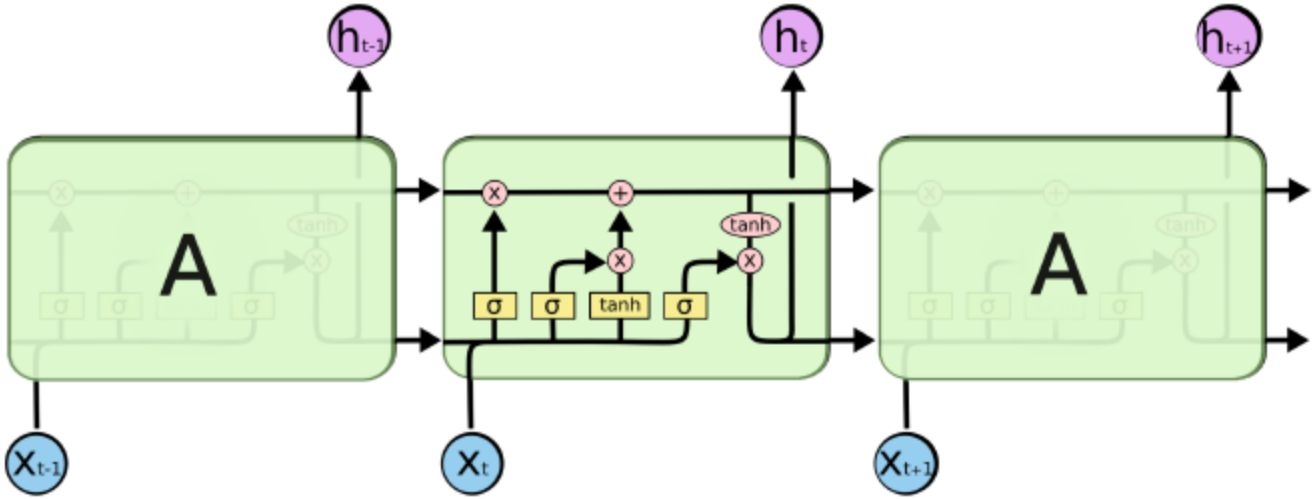
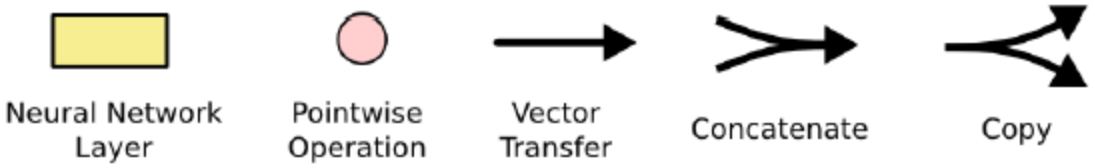


Fig6: The repeating module in a standard RNN contains a single layer [4]

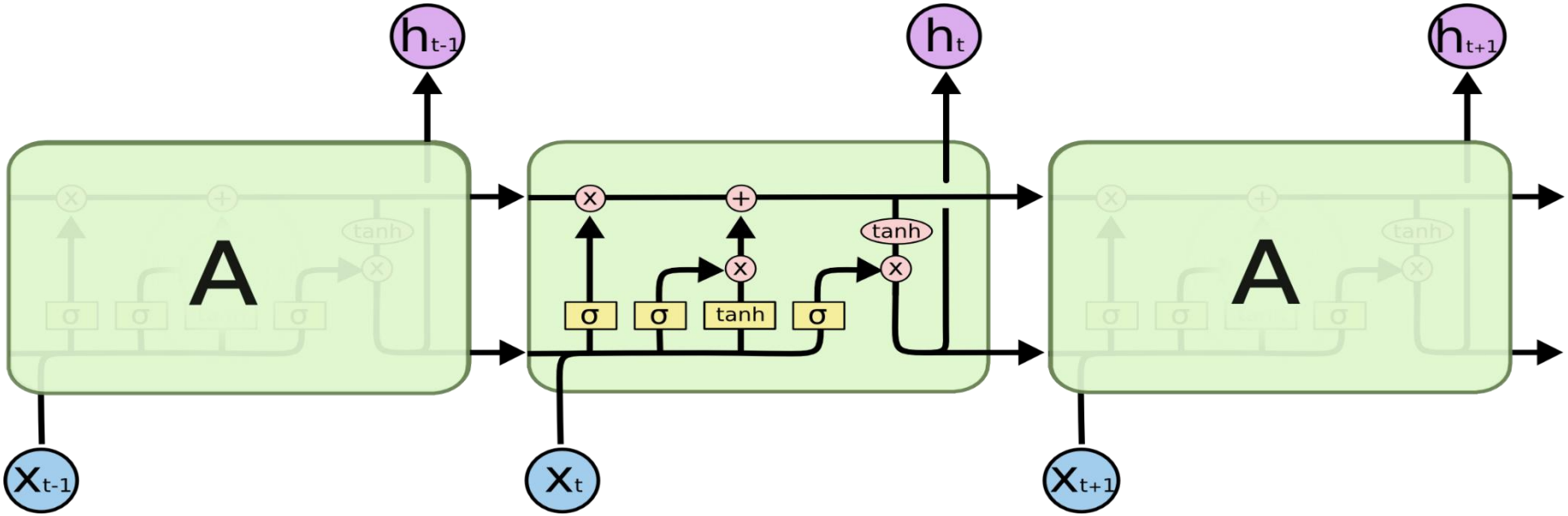


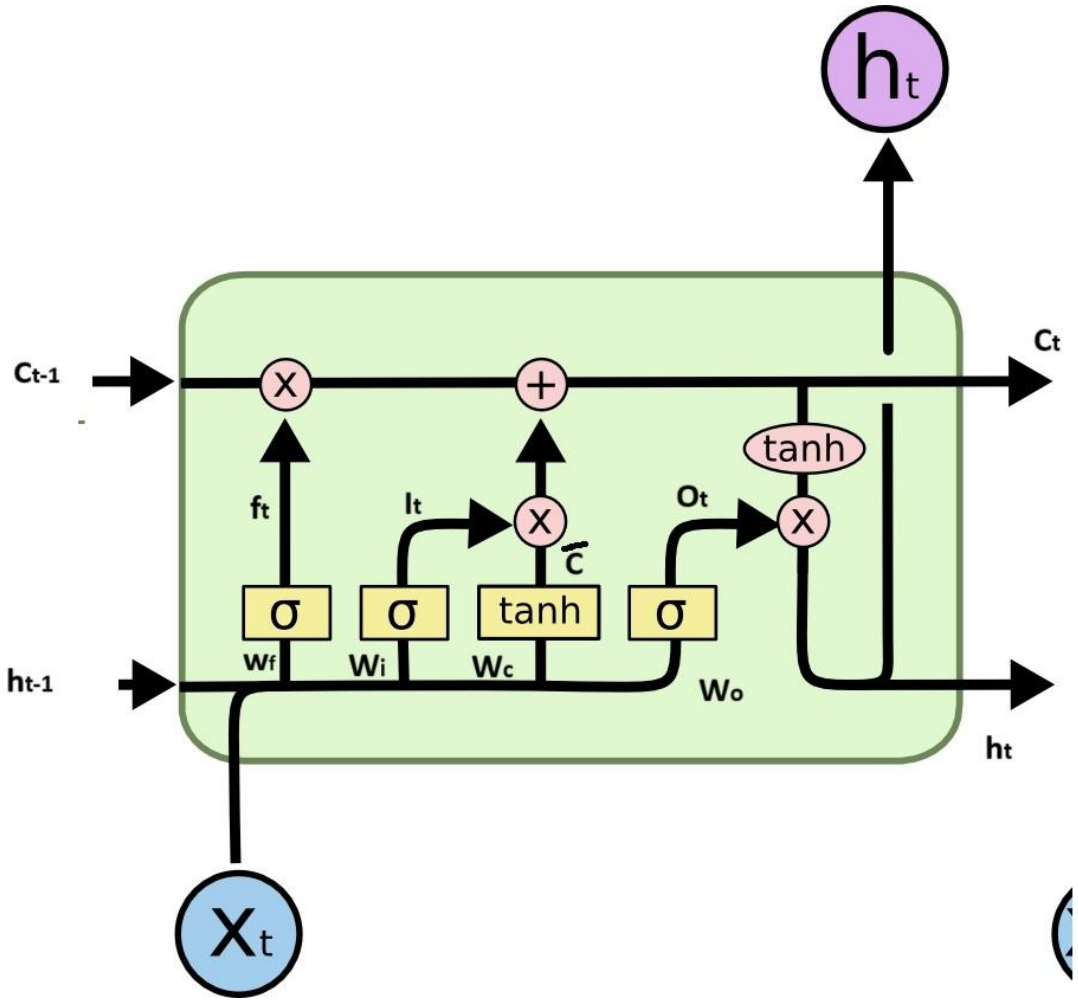
LSTM Networks

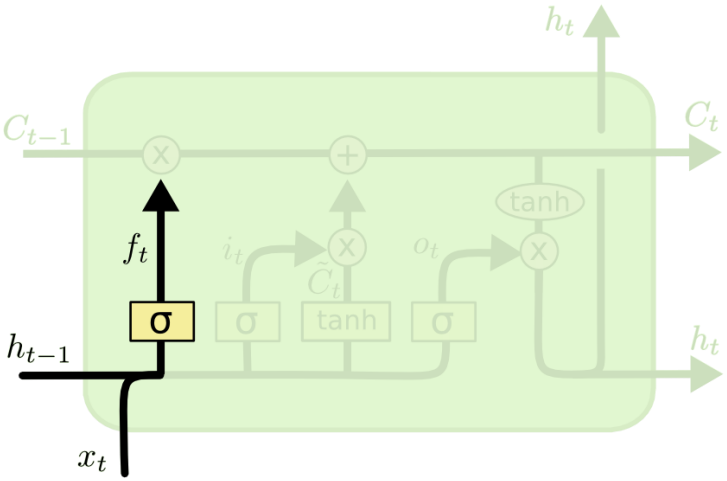
Long Short Term Memory networks – usually just called "LSTMs" – are a special kind of RNN, capable of learning long-term dependencies. They were introduced by Hochreiter & Schmidhuber (1997), and were refined and popularized by many people in following work.[1] They work tremendously well on a large variety of problems, and are now widely used.

LSTMs are explicitly designed to avoid the long-term dependency problem. Remembering information for long periods of time is practically their default behavior, not something they struggle to learn!

All recurrent neural networks have the form of a chain of repeating modules of neural network. In standard RNNs, this repeating module will have a very simple structure, such as a single tanh layer.




ht
Ct-1
Ct
tanh
ft
It
Ot
C̃
σ
σ
tanh
σ
ht-1
Wf
Wi
Wc
Wo
ht
Xt



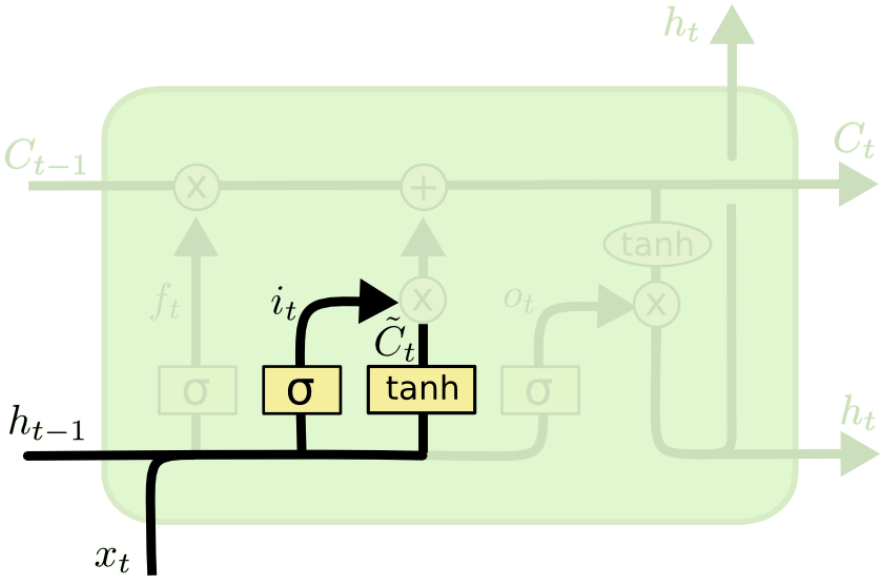
$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

1. Forget Gate

- Purpose: Decides what information from the previous cell state should be discarded.
- Formula:

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

- Activation Function: **Sigmoid (σ)** (values range between 0 and 1)
- If f_t is **0**, the information is completely forgotten; if **1**, it's fully retained.



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

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

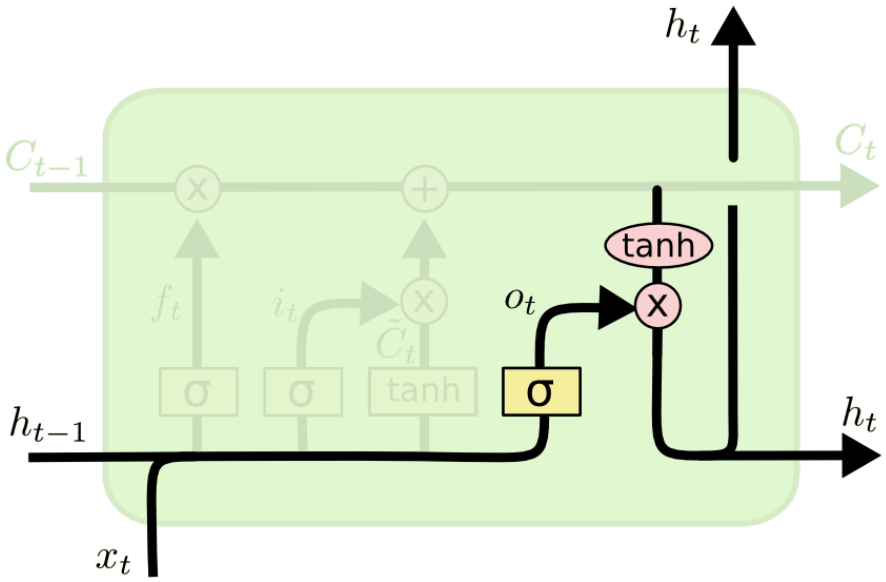
2. Input Gate

- Purpose: Determines what new information should be added to the cell state.
- Formula:

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

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

- The input gate i_t decides how much of the new candidate memory $\tilde{C}_t$ should be added to the cell state.



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

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

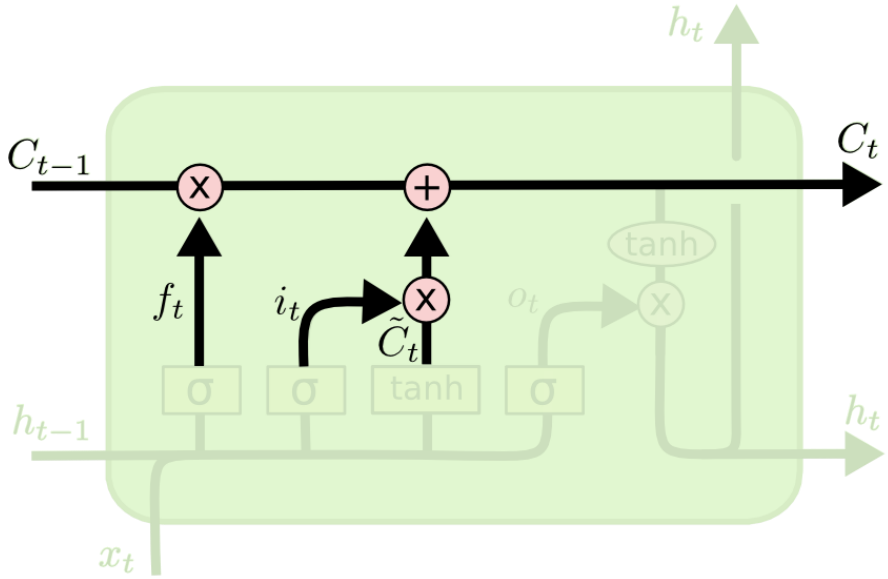
3. Output Gate

- Purpose: Determines what part of the cell state should be output as the hidden state.
- Formula:

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

$$h_t = o_t \cdot \tanh(C_t)$$

- The output gate o_t scales the output based on the current cell state.



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

Cell State Update

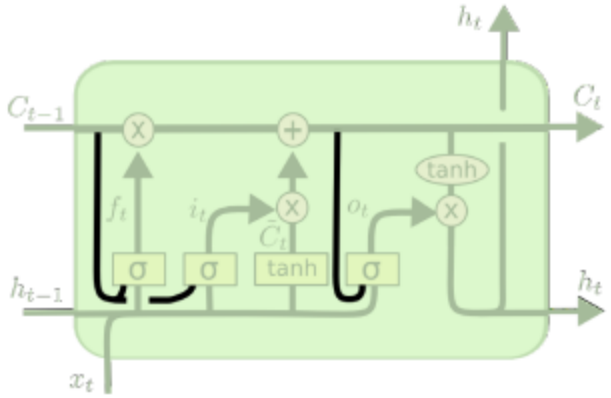
The overall cell state update equation is:

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

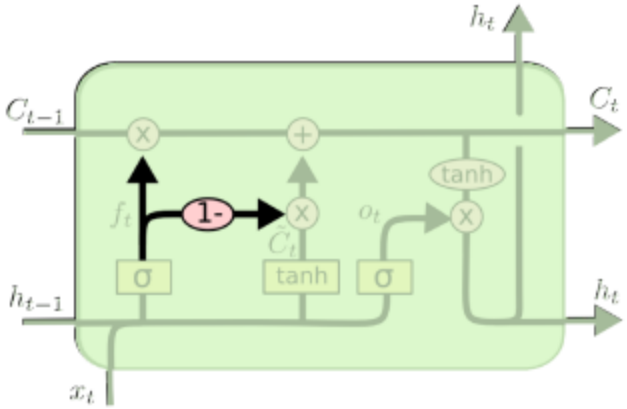
This ensures that relevant long-term information is retained while discarding unnecessary details.

Long Short Term Memory (LSTM)

- Variants of LSTM



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



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

CHITKARA UNIVERSITY

Thank You