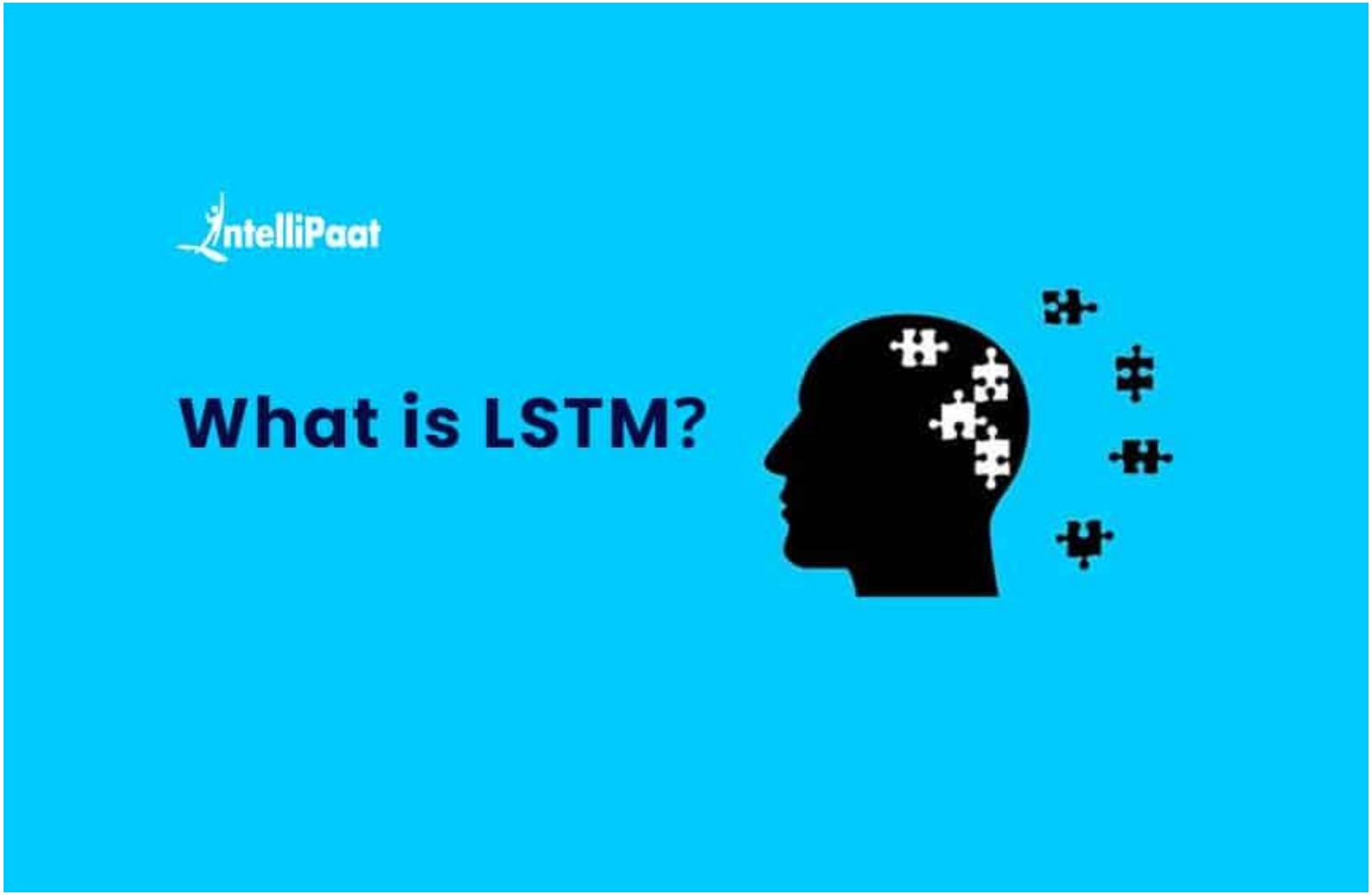
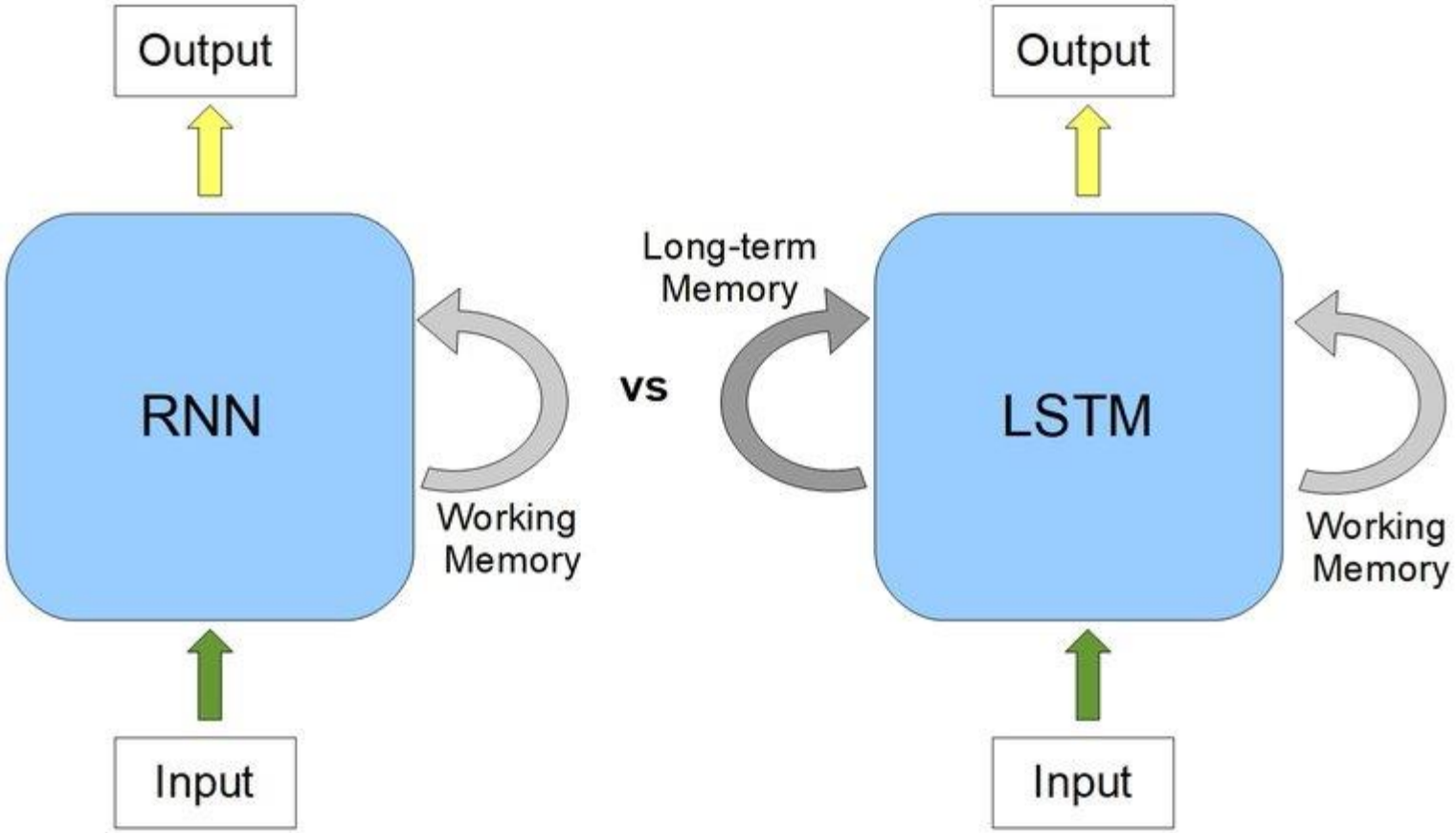


Dr. Amit Sinhal

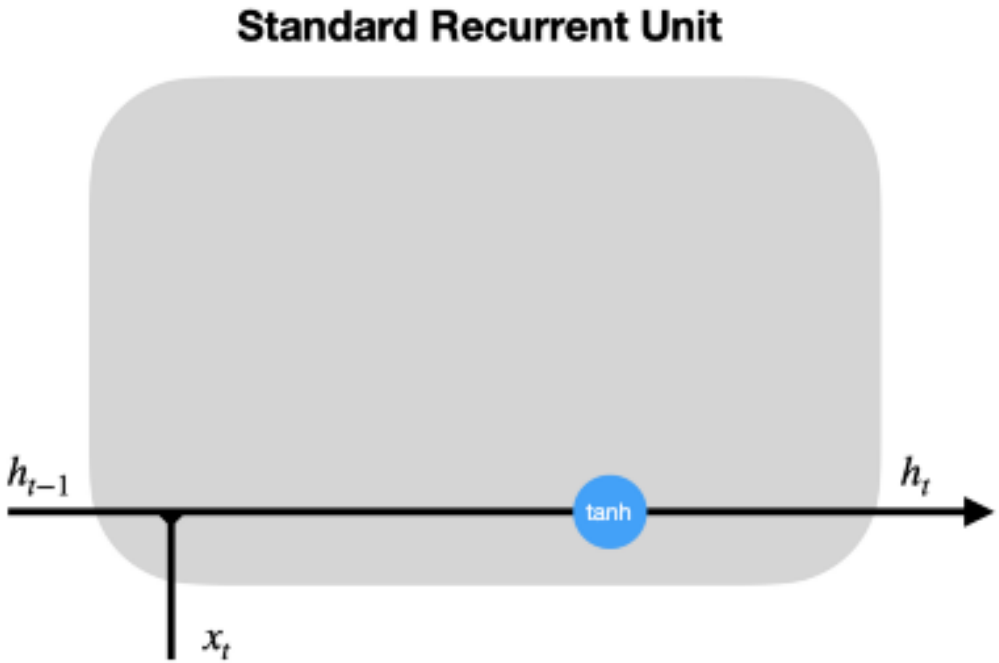
Professor

Department of Computer Science & Engineering

RNN Vs LSTM



RNN Vs LSTM



h_{t-1} - hidden state at previous timestep t-1 (memory)
x_t - input vector at current timestep t
h_t - hidden state at current timestep t
tanh - tanh activation function
- concatenation of vectors

After the hidden state is calculated at timestep t, it is **passed back to the recurrent unit** and combined with the input at timestep t+1 to calculate the new hidden state at timestep t+1. This process repeats for t+2, t+3, …, t+n until the predefined number (n) of timesteps is reached.

Meanwhile, LSTM employs various gates to decide what information to keep or discard. Also, it adds a **cell state**, which is like a long-term memory of LSTM

Consider a Scenario

In order to add a new information, RNN transforms the existing information completely by applying a function. Because of this, the entire information is modified, on the whole, i. e. there is no consideration for *'important'* information and *'not so important'* information.

LSTMs on the other hand, make small modifications to the information by multiplications and additions. With LSTMs, the information flows through a mechanism known as cell states. This way, LSTMs can selectively remember or forget things. The information at a particular cell state has three different dependencies.

Let's take the example of predicting stock prices for a particular stock. The stock price of today will depend upon:

1.The trend that the stock has been following in the previous days, maybe a downtrend or an uptrend.

2.The price of the stock on the previous day, because many traders compare the stock's previous day price before buying it.

3.The factors that can affect the price of the stock for today. This can be a new company policy that is being criticized widely, or a drop in the company's profit, or maybe an unexpected change in the senior leadership of the company.

Consider another Scenario

Another important feature of LSTM is its analogy with conveyor belts!

Industries use them to move products around for different processes. LSTMs use this mechanism to move information around.

We may have some addition, modification or removal of information as it flows through the different layers, just like a product may be molded, painted or packed while it is on a conveyor belt.

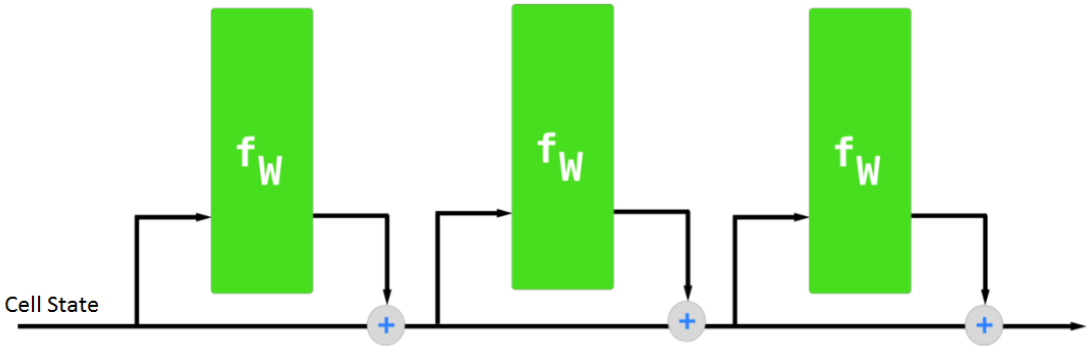
Just because of this property of LSTMs, where they do not manipulate the entire information but rather modify them slightly, they are able to *forget* and *remember* things selectively.

These dependencies can be generalized to any problem as:

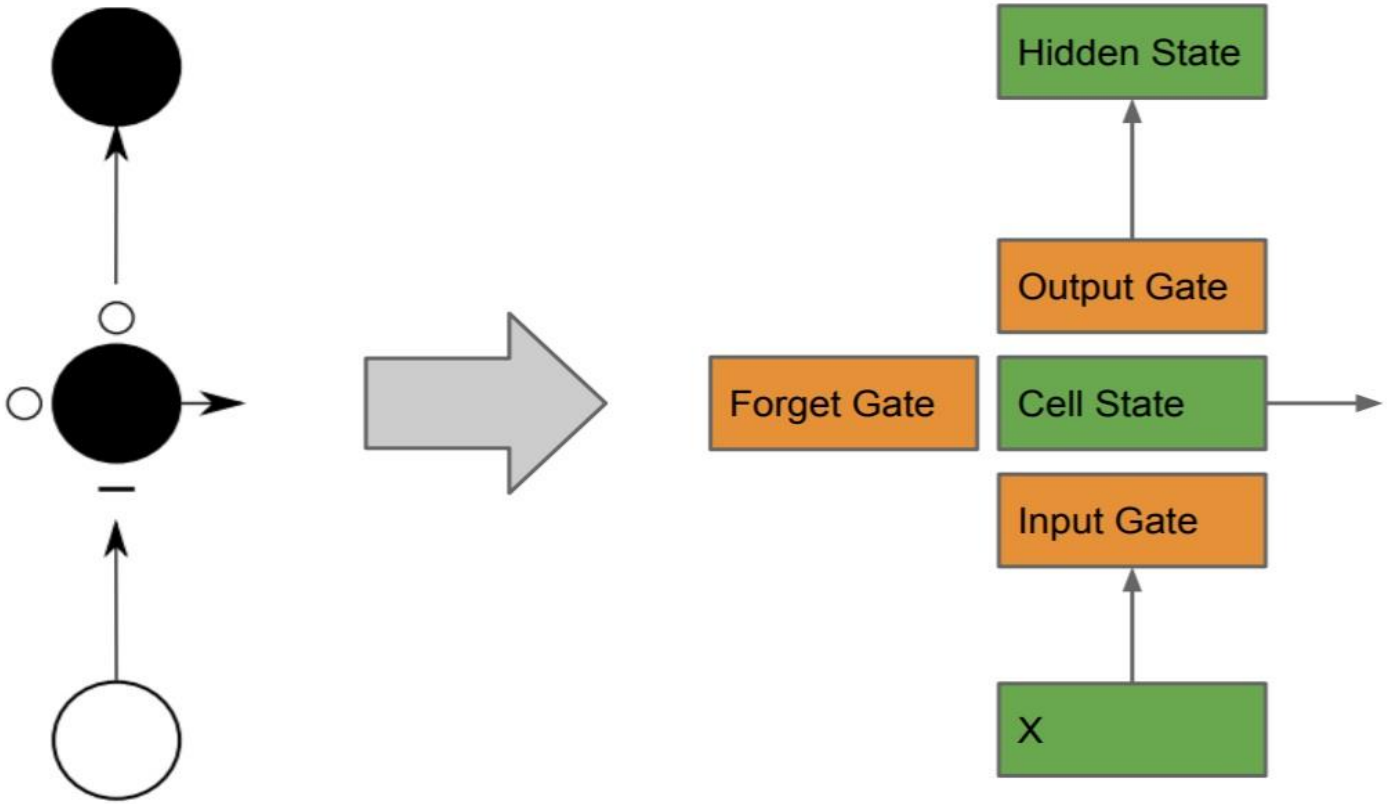
1.The **previous cell state** *(i.e. the information that was present in the memory after the previous time step)*

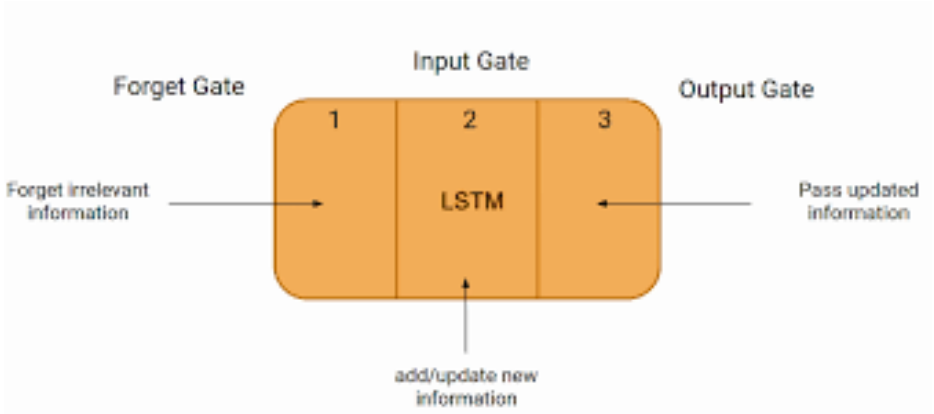
2.The **previous hidden state** *(i.e. this is the same as the output of the previous cell)*

3.The **input at the current time step** *(i.e. the new information that is being fed in at that moment)*

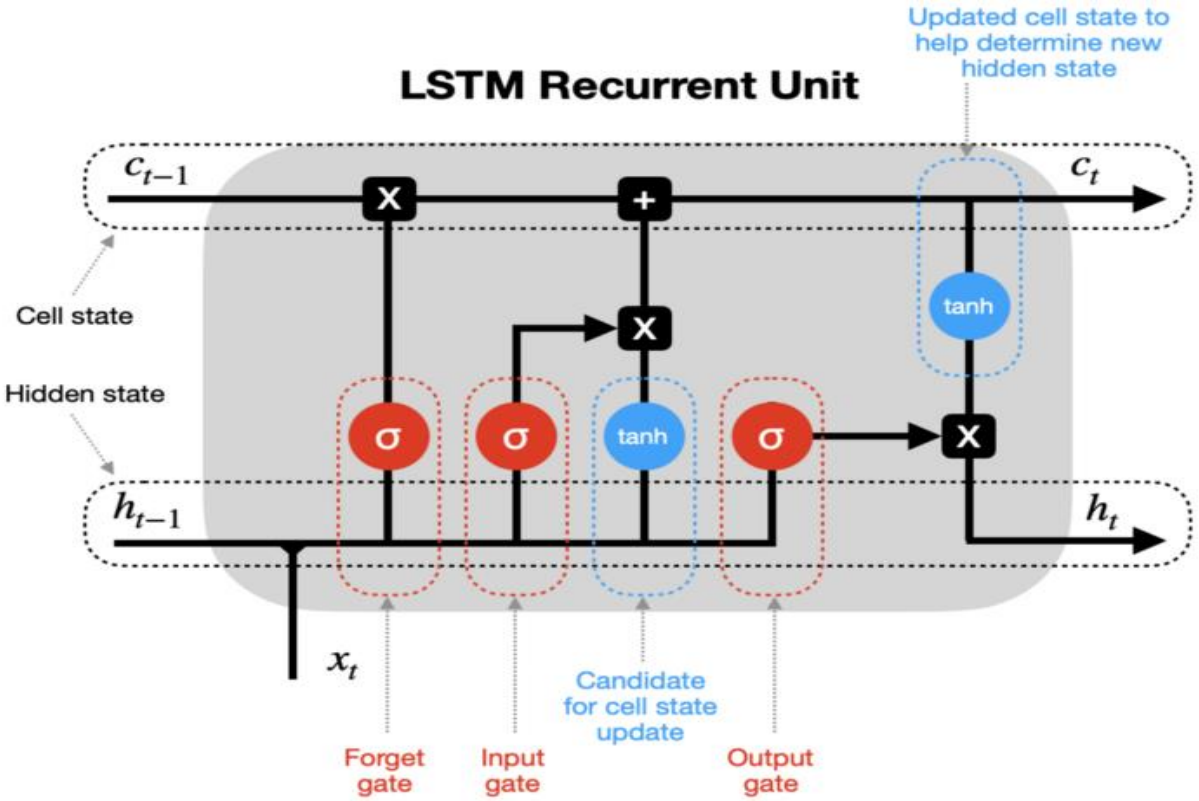


Anatomy of an LSTM node





LSTM recurrent unit is much more complex than that of RNN, which improves learning but requires more computational resources.



h_{t-1} - hidden state at previous timestep t-1 (short-term memory)
c_{t-1} - cell state at previous timestep t-1 (long-term memory)
x_t - input vector at current timestep t
h_t - hidden state at current timestep t
c_t - cell state at current timestep t

X - vector pointwise multiplication
\+ - vector pointwise addition
tanh - tanh activation function
σ - sigmoid activation function
⊤ - concatenation of vectors
- states
- gates
- updates

Hidden state & new inputs — hidden state from a previous timestep (h_t-1) and the input at a current timestep (x_t) are combined before passing copies of it through various gates.

Forget gate — this gate controls what information should be forgotten. Since the sigmoid function ranges between 0 and 1, it sets which values in the cell state should be discarded (multiplied by 0), remembered (multiplied by 1), or partially remembered (multiplied by some value between 0 and 1).

Input gate helps to identify important elements that need to be added to the cell state. Note that the results of the input gate get multiplied by the cell state candidate, with only the information deemed important by the input gate being added to the cell state.

Update cell state —first, the previous cell state (c_t-1) gets multiplied by the results of the forget gate. Then we add new information from [input gate × cell state candidate] to get the latest cell state (c_t).

Update hidden state — the last part is to update the hidden state. The latest cell state (c_t) is passed through the tanh activation function and multiplied by the results of the output gate.

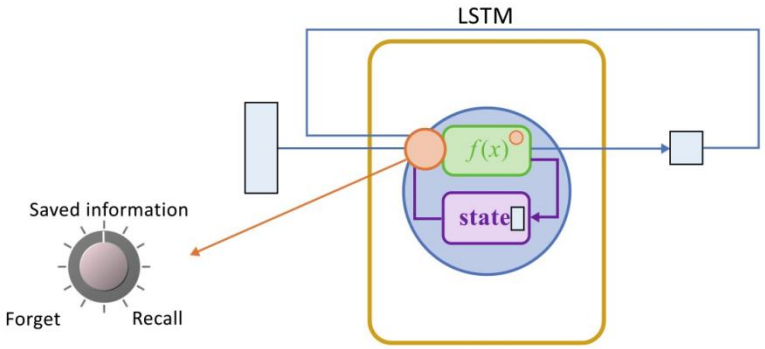
Finally, the latest cell state (c_t) and the hidden state (h_t) go back into the recurrent unit, and the **process repeats at timestep t+1**. The loop continues until we reach the end of the sequence.

LSTM Cycle

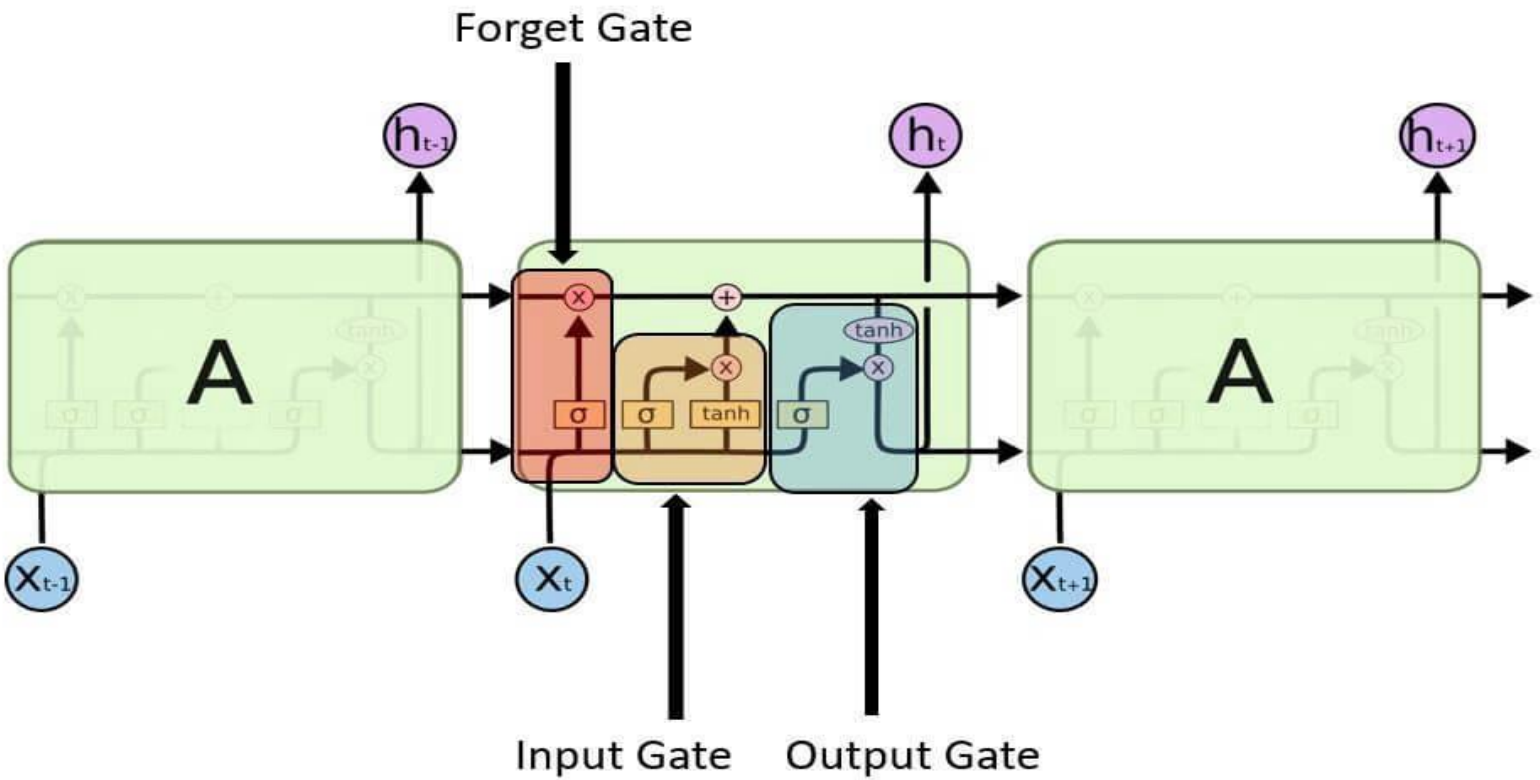
The LSTM cycle is divided into four steps:

- Using the forget gate, information to be forgotten is identified from a prior time step.
- Using input gate and tanh, new information is sought for updating cell state.
- The information from the two gates above is used to update the cell state.
- The output gate and the squashing operation provide useful information.

A dense layer receives the output of an LSTM cell. After the dense layer, the output stage is given the softmax activation function.

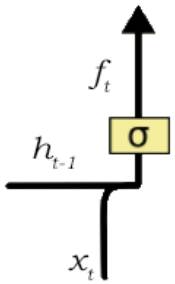


A typical LSTM network is comprised of different memory blocks called **cells** (the rectangles that we see in the image)**.** There are two states that are being transferred to the next cell; the **cell state** and the **hidden state**. The memory blocks are responsible for remembering things and manipulations to this memory is done through three major mechanisms, called **gates.**



Forget Gate

A forget gate is responsible for removing information from the cell state. The information that is no longer required for the LSTM to understand things or the information that is of less importance is removed via multiplication of a filter. This is required for optimizing the performance of the LSTM network.



This gate takes in two inputs; h_t-1 and x_t.

h_t-1 is the hidden state from the previous cell or the output of the previous cell and x_t is the input at that particular time step. The given inputs are multiplied by the weight matrices and a bias is added.

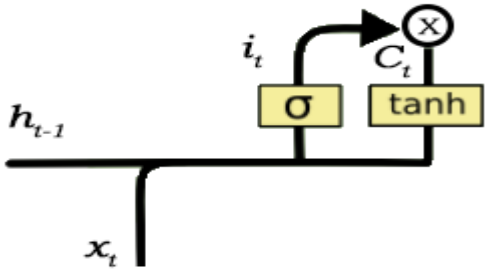
Following this, the sigmoid function is applied to this value. The sigmoid function outputs a vector, with values ranging from 0 to 1, corresponding to each number in the cell state.

Basically, the sigmoid function is responsible for deciding which values to keep and which to discard. If a '0' is output for a particular value in the cell state, it means that the forget gate wants the cell state to forget that piece of information completely.

Similarly, a '1' means that the forget gate wants to remember that entire piece of information. This vector output from the sigmoid function is multiplied to the cell state.

Input Gate

The input gate is responsible for the addition of information to the cell state. This addition of information is basically three-step process as seen from the diagram above.

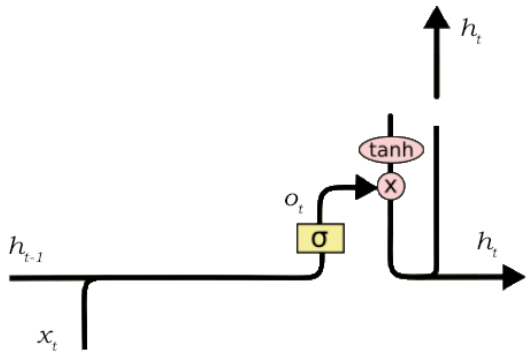


- Regulating what values need to be added to the cell state by involving a sigmoid function. This is basically very similar to the forget gate and acts as a filter for all the information from h_t-1 and x_t.
- Creating a vector containing all possible values that can be added (as perceived from h_t-1 and x_t) to the cell state. This is done using the **tanh** function, which outputs values from -1 to +1.
- Multiplying the value of the regulatory filter (the sigmoid gate) to the created vector (the tanh function) and then adding this useful information to the cell state via addition operation.

Once this three-step process is done with, we ensure that only that information is added to the cell state that is *important* and is not *redundant.*

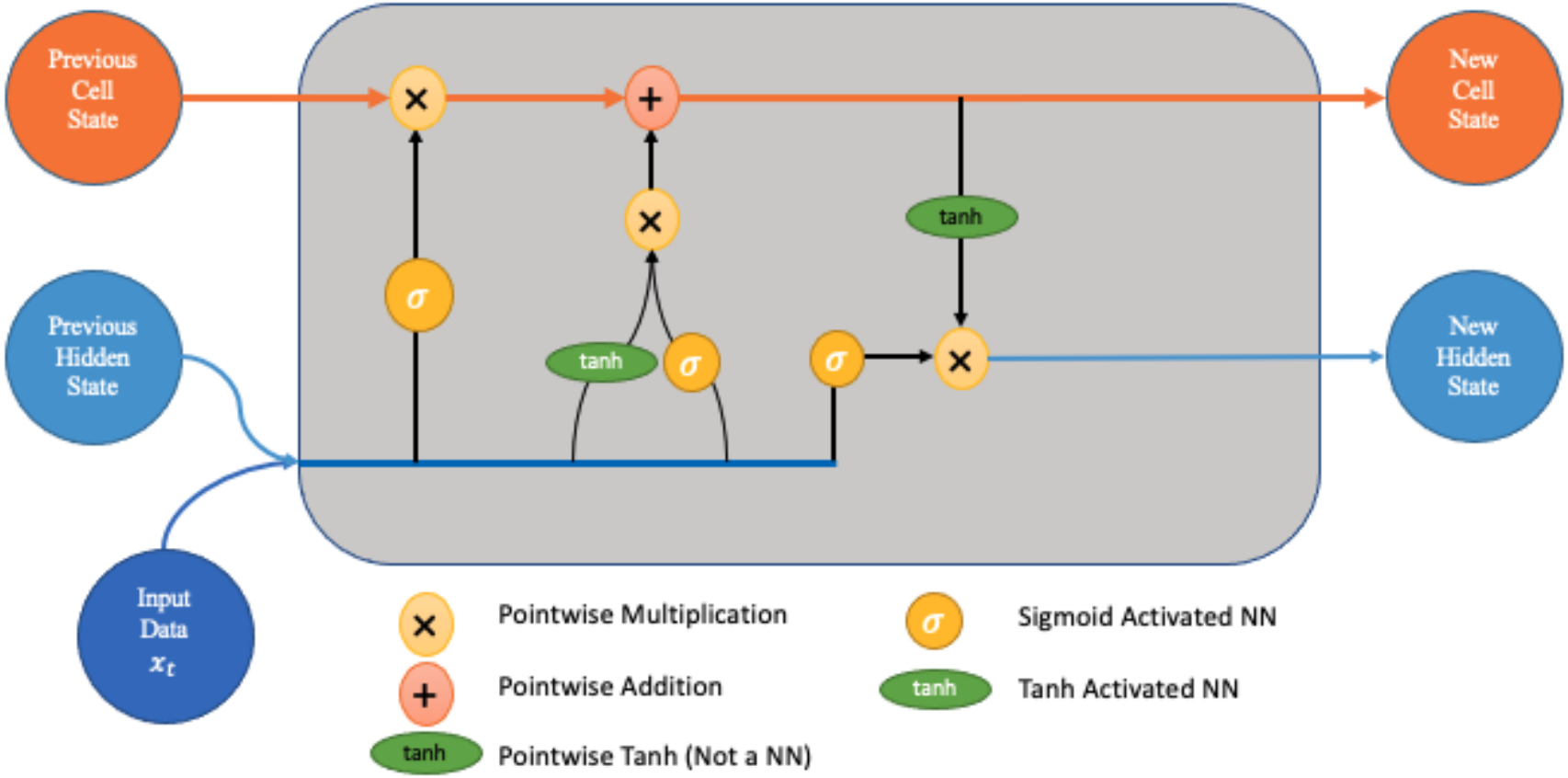
Output Gate

The job of selecting useful information from the current cell state and showing it out as an output is done via the output gate. Here is its structure:



The functioning of an output gate can again be broken down to three steps:

- Creating a vector after applying **tanh** function to the cell state, thereby scaling the values to the range -1 to +1.
- Making a filter using the values of h_t-1 and x_t, such that it can regulate the values that need to be output from the vector created above. This filter again employs a sigmoid function.
- Multiplying the value of this regulatory filter to the vector created in step 1, and sending it out as a output and also to the hidden state of the next cell.



- The state of the cell resembles that of a conveyor belt in certain ways. There are only a few tiny linear interactions as it travels down the entire chain. It's quite easy for data to simply travel down it without being altered.
- The LSTM can delete or add information to the cell state, which is carefully controlled by structures called gates.

- Gates are a mechanism to selectively allow information to pass through. A sigmoid neural net layer plus a pointwise multiplication operation make them up.
- The sigmoid layer produces integers ranging from zero to one, indicating how much of each component should be allowed to pass. A value of zero indicates that “nothing” should be allowed through, whereas a value of one indicates that “everything” should be allowed through.
- Three of these gates are present in an LSTM to protect and govern the cell state.

Applications

LSTM has a number of well-known applications, including:

- Image captioning
- Machine translation
- Language modelling
- Handwriting generation
- Question answering chatbots