Module 5

Long Short-Term Memory Networks (LSTM)

LSTM excels in sequence prediction tasks, capturing long-term dependencies. Ideal for time series, machine translation, and speech recognition due to order dependence. The article provides an in-depth introduction to LSTM, covering the LSTM model, architecture, working principles, and the critical role they play in various applications.

What is LSTM?

Long Short-Term Memory is an improved version of recurrent neural network designed by Hochreiter & Schmidhuber.

A traditional RNN has a single hidden state that is passed through time, which can make it difficult for the network to learn long-term dependencies. LSTMs model address this problem by introducing a memory cell, which is a container that can hold information for an extended period.

LSTM architectures are capable of learning long-term dependencies in sequential data, which makes them well-suited for tasks such as language translation, speech recognition, and time series forecasting.

LSTMs can also be used in combination with other neural network architectures, such as Convolutional Neural Networks (CNNs) for image and video analysis.

LSTM Architecture

The LSTM architectures involves the memory cell which is controlled by three gates: the input gate, the forget gate, and the output gate. These gates decide what information to add to, remove from, and output from the memory cell.

- The input gate controls what information is added to the memory cell.
- The forget gate controls what information is removed from the memory cell.
- The output gate controls what information is output from the memory cell.

This allows LSTM networks to selectively retain or discard information as it flows through the network, which allows them to learn long-term dependencies.

Module 5

The LSTM maintains a hidden state, which acts as the short-term memory of the network. The hidden state is updated based on the input, the previous hidden state, and the memory cell's current state.

What are long-term dependencies?

Many times only **recent data** is needed in a model to **perform operations.** But there might be a **requirement** from **data** that was **obtained** in the **past.**

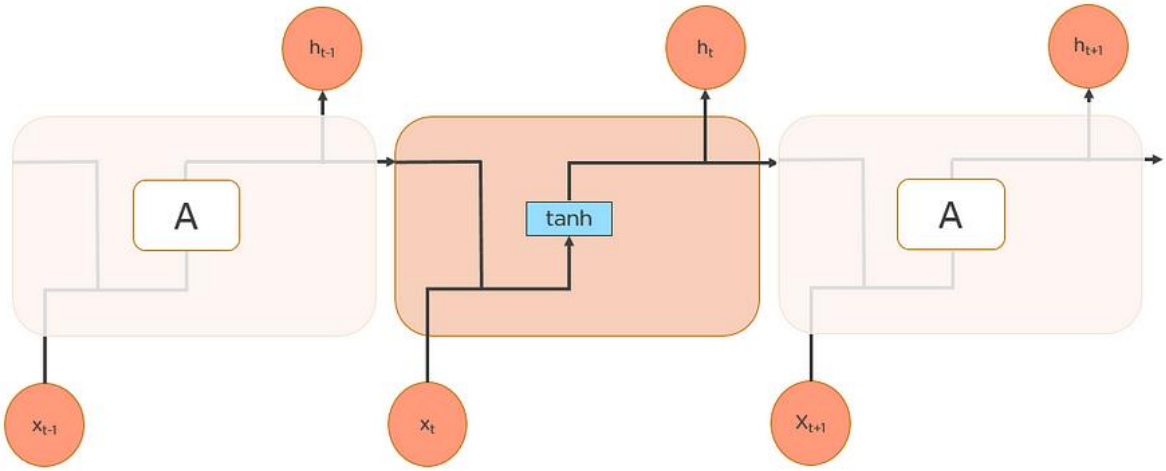
Let's look at the following example:

If we are trying to predict the last word in "The clouds are in the **sky**" we don't need any further context - it's pretty obvious the next word is going to be the **sky**.

In such cases, where the gap between the relevant information and the place that is needed is small, RNNs can learn to use the past information.

LSTMs are a special kind of Recurrent Neural Network — capable of learning long-term dependencies by remembering information for long periods is the default behavior.

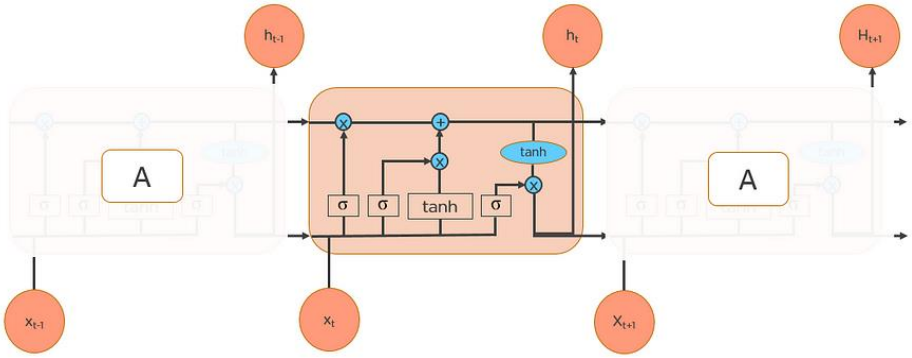
All recurrent neural networks are in the form of a chain of repeating modules of a neural network. In standard RNNs, this repeating module will have a very simple structure, such as a single tanh layer.



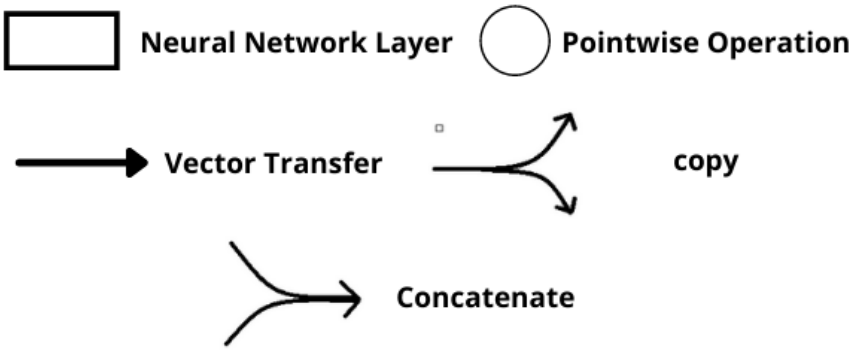
Long Short Term Memory Networks

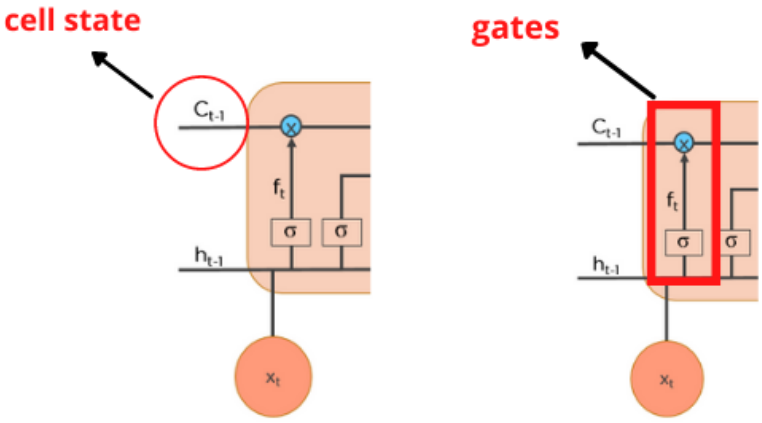
Module 5

LSTMs also have a chain-like structure, but the repeating module is a bit different structure. Instead of having a single neural network layer, four interacting layers are communicating extraordinarily.



Let's know about some notations and statements:





Module 5

Cell State — The key to LSTMs is the cell state, the horizontal line running through the top of the diagram.

The cell state is kind of like a conveyor belt. It runs straight down the entire chain, with only some minor linear interactions. It's very easy for information to just flow along with it unchanged.

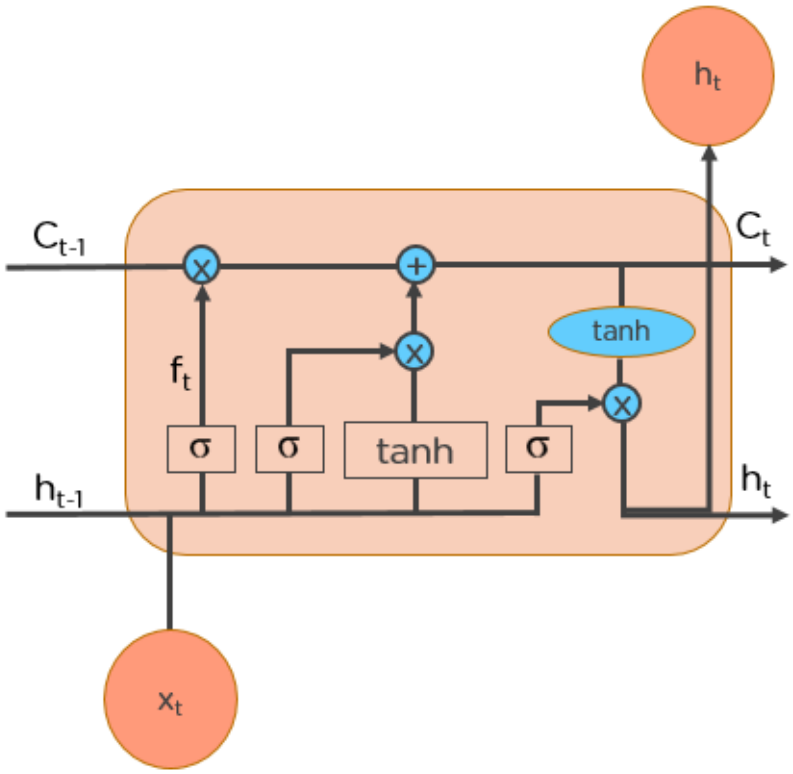
The LSTM does have the ability to remove or add information to the cell state, carefully regulated by a structure called **gates**

Gates — Gates are the way to optionally let information through. They are composed out of a sigmoid neural net layer and a pointwise multiplication operation.

The sigmoid layer outputs numbers between zero and one, describing how much of each component should be let through. A value of **zero** means "**let nothing through**" while a value of **one** means "**let everything through**"

An LSTM has three of these gates, to protect and control the cell state.

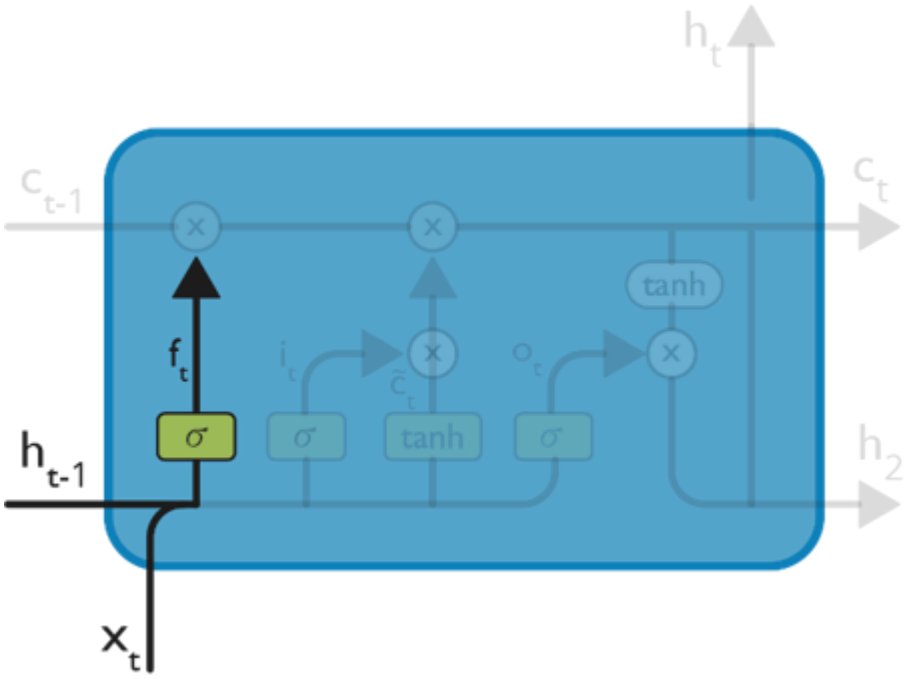
Workings of LSTMs



Module 5

LSTMs work in a 4-step process:

STEP 1:



— The first step in our LSTM is to decide which information should be omitted from the cell in that particular timestamp.

— This decision is made by a Sigmoid Layer called the "**Forget Gate Layer**".

— It looks at **h(t-1)** and **x(t)** and outputs a number between 0 and 1 for each number in the cell state **(c t-1).**

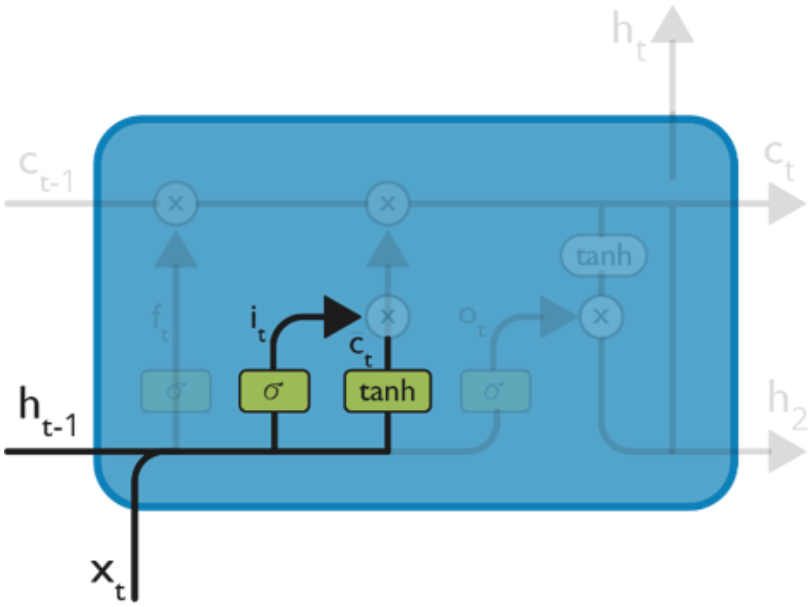
— **1** represents "**completely keep this**" while a **0** represents "**completely get rid of this**".

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

$w_f = Weight$
$h_{t-1} = Output\ from\ previous\ timestamp$
$x_t = New\ input$
$b_f = Bias$

Module 5

STEP 2:



— In this second layer, there are two parts. One is the sigmoid function and the other is the tanh function.

— In the **sigmoid function**, it decides which value to let through (0 or 1).

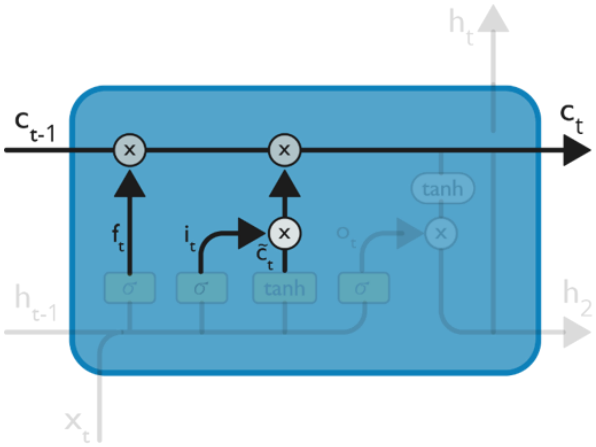
— In the **tanh function**, it gives weightage to the value which is passed, deciding their level of importance (-1 to 1).

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

$$\tilde{c}_t = tanh(w_c[h_{t-1}, x_t] + b_c)$$

STEP 3:

Module 5

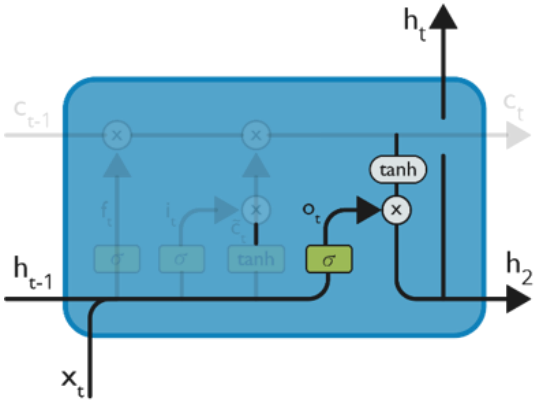


— Now, we will update the old cell state **c(t-1)**, into the new cell **c(t)**. First, we multiply the old state **c(t-1)** by **f(t)**, forgetting the things we decided to forget earlier.

— Then we add **i(t) * c(t)**. These are the new candidate values, scaled by how much we decided to update each state value.

$$c_t = f_t * c_{t-1} + i_t^* \tilde{c}_t$$

STEP 4:



— Finally, we need to decide what we are going to output. This output will be based on our cell state but will be a filtered version.

— We will run a sigmoid layer which decides what parts of the cell state we are going to output. Then, we put the cell state through tanh (push the values to be between −1 and 1)

and multiply it by the output of the sigmoid gate, so that we only output the parts we decide to.

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

$$h_t = o_t * \tanh(c_t)$$

Advantages of LSTM:

- Non-decaying error backpropagation
- For long-time lag problems, LSTM can handle noise and continuous values.
- No parameter fine-tuning
- Memory for long time periods.