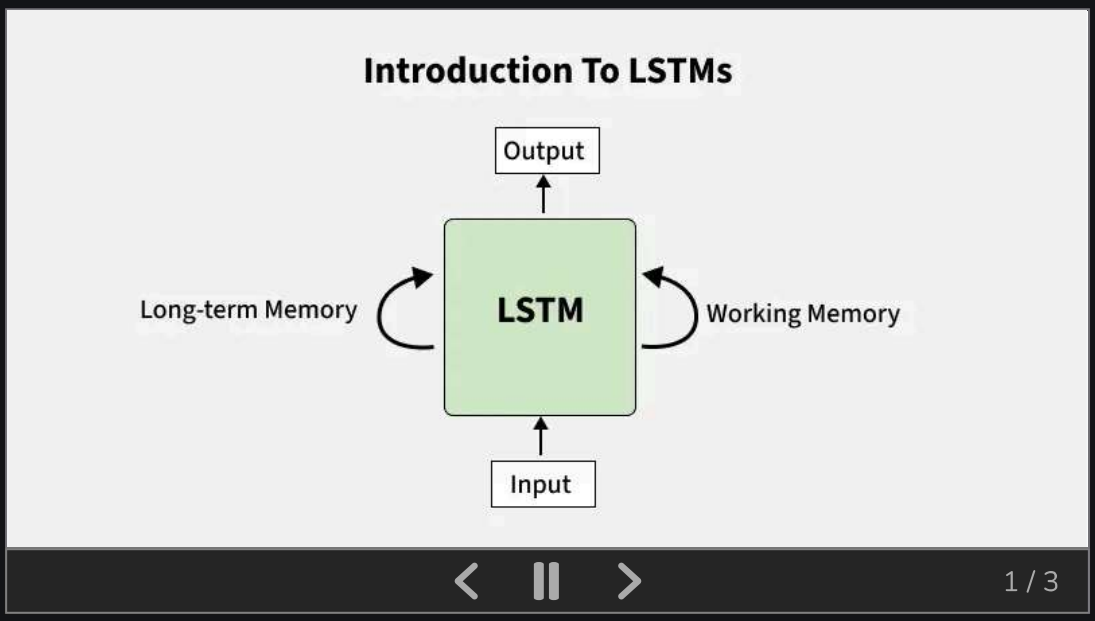
What is LSTM - Long Short Term Memory?

Last Updated : 07 Oct, 2025

Long Short-Term Memory (LSTM) is an enhanced version of the Recurrent Neural Network (RNN) designed by Hochreiter and Schmidhuber. LSTMs can capture long-term dependencies in sequential data making them ideal for tasks like language translation, speech recognition and time series forecasting. Unlike traditional RNNs which use a single hidden state passed through time LSTMs introduce a memory cell that holds information over extended periods addressing the challenge of learning long-term dependencies.



Problem with Long-Term Dependencies in RNN

Recurrent Neural Networks (RNNs) are designed to handle sequential data by maintaining a hidden state that captures information from previous time steps. However they often face challenges in learning long-term dependencies where information from distant time steps becomes crucial for making accurate predictions for current state. This

problem is known as the vanishing gradient or exploding gradient problem.

- **Vanishing Gradient**: When training a model over time, the gradients which help the model learn can shrink as they pass through many steps. This makes it hard for the model to learn long-term patterns since earlier information becomes almost irrelevant.
- **Exploding Gradient**: Sometimes gradients can grow too large causing instability. This makes it difficult for the model to learn properly as the updates to the model become erratic and unpredictable.

Both of these issues make it challenging for standard RNNs to effectively capture long-term dependencies in sequential data.

LSTM Architecture

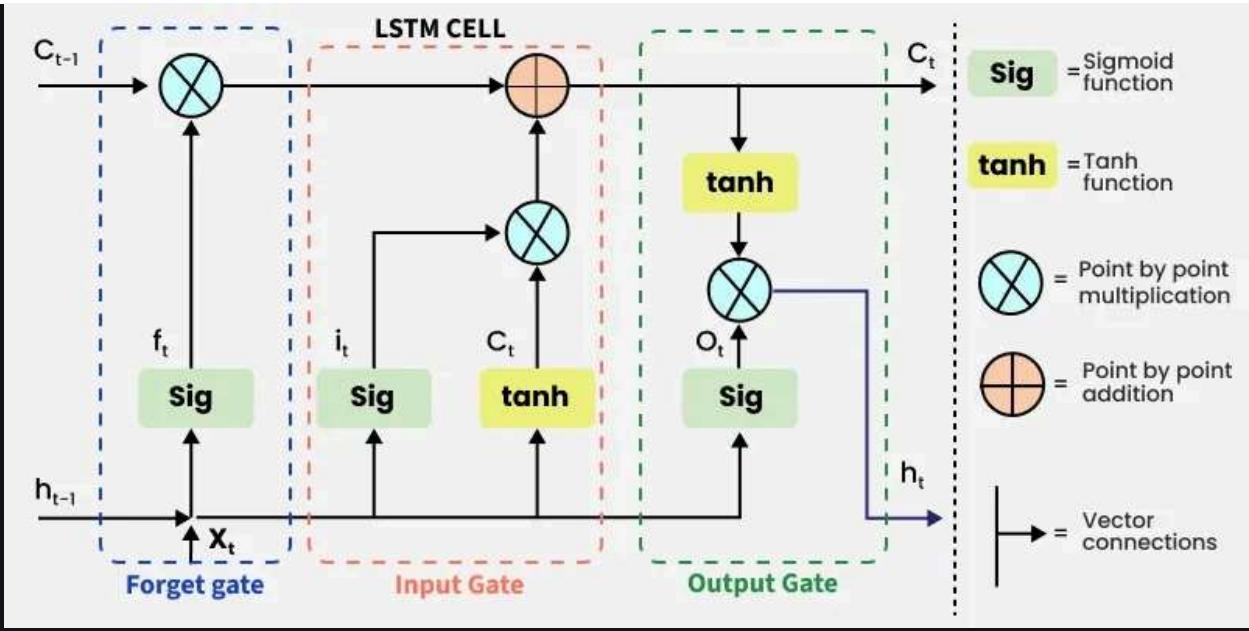
LSTM architectures involves the memory cell which is controlled by three gates:

1. **Input gate**: Controls what information is added to the memory cell.
2. **Forget gate**: Determines what information is removed from the memory cell.
3. **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. The network has a hidden state which is like its short-term memory. This memory is updated using the current input, the previous hidden state and the current state of the memory cell.

Working of LSTM

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



LSTM Model

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 is 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 sigmoid activation function which gives output in range of [0,1]. If for a particular cell state the output is 0 or near to 0, the piece of information is forgotten and for output of 1 or near to 1, the information is retained for future use.

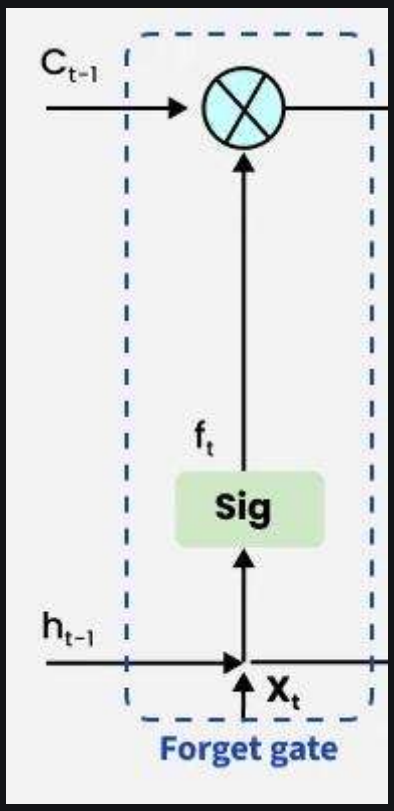
The equation for the forget gate is:

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

Where:

- W_f represents the weight matrix associated with the forget gate.
- $[h_t - 1, x_t]$ denotes the concatenation of the current input and the previous hidden state.

- b_f is the bias with the forget gate.
- σ is the sigmoid activation function.



Forget Gate

2. Input gate

The addition of useful information to the cell state is done by the 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 an output from -1 to +1 which contains all the possible values from h_{t-1} and x_t. At last the values of the vector and the regulated values are multiplied to obtain the useful information. The equation for the input gate is:

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

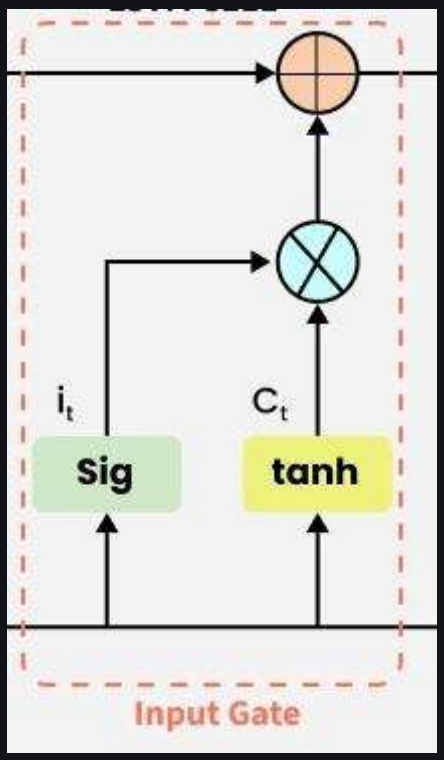
$$\hat{C}_t = \tanh\left(W_c \cdot [h_{t-1}, x_t] + b_c\right)$$

We multiply the previous state by f_t effectively filtering out the information we had decided to ignore earlier. Then we add $i_t \odot C_t$ which represents the new candidate values scaled by how much we decided to update each state value.

$$C_t = f_t \odot C_{t-1} + i_t \odot \hat{C}_t$$

where

- $\odot$ denotes element-wise multiplication
- tanh is activation function



Input Gate

3. Output gate

The output gate is responsible for deciding what part of the current cell state should be sent as the hidden state (output) for this time step.First, the gate uses a sigmoid function to determine which information from the current cell state will be output. This is done using the previous hidden state h_{t-1} and the current input x_t:

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

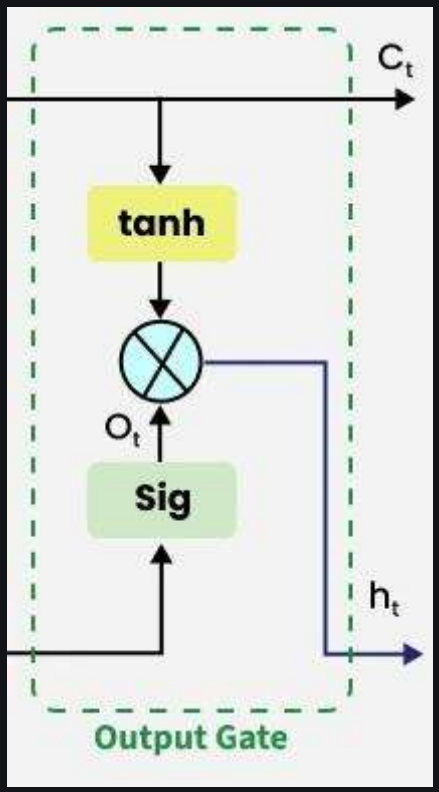
Next, the current cell state C_t is passed through a tanh activation to scale its values between -1 and $+1$. Finally, this transformed cell state is multiplied element-wise with o_t to produce the hidden state h_t:

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

Here:

- o_t is the output gate activation.
- C_t is the current cell state.
- $\odot$ represents element-wise multiplication.
- σ is the sigmoid activation function.

This hidden state h_t is then passed to the next time step and can also be used for generating the output of the network.



Output Gate

Applications

Some of the famous applications of LSTM includes:

- **Language Modeling**: Used in tasks like language modeling, machine translation and text summarization. These networks learn the dependencies between words in a sentence to generate coherent and grammatically correct sentences.
- **Speech Recognition**: Used in transcribing speech to text and recognizing spoken commands. By learning speech patterns they can match spoken words to corresponding text.
- **Time Series Forecasting**: Used for predicting stock prices, weather and energy consumption. They learn patterns in time series data to predict future events.
- **Anomaly Detection**: Used for detecting fraud or network intrusions. These networks can identify patterns in data that deviate drastically and flag them as potential anomalies.
- **Recommender Systems**: In recommendation tasks like suggesting movies, music and books. They learn user behavior patterns to provide personalized suggestions.
- **Video Analysis**: Applied in tasks such as object detection, activity recognition and action classification. When combined with Convolutional Neural Networks (CNNs) they help analyze video data and extract useful information.

Comment
aakars... + Follow
42

Article Tags : Deep Learning

GeeksforGeeks
Sanchhaya Education Private Limited
Corporate & Communications Address: