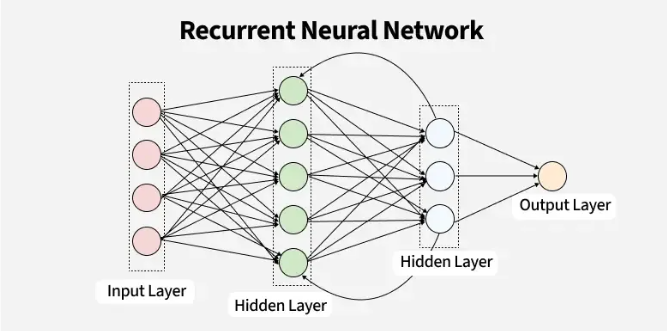
RNN

**Recurrent Neural Networks (RNNs)** differ from regular neural networks in how they process information. While standard neural networks pass information in one direction i.e from input to output, RNNs feed information back into the network at each step.



## **Key Components of RNNs**

There are mainly two components of RNNs that we will discuss.

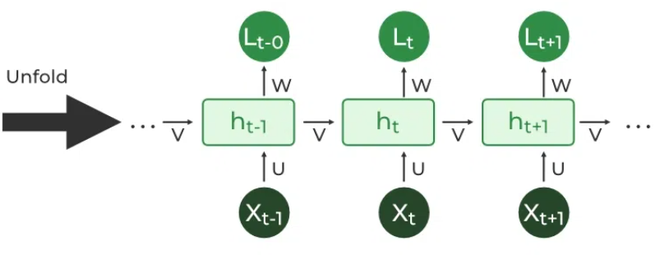
### **1. Recurrent Neurons**recurrent-neuron

The fundamental processing unit in RNN is a **Recurrent Unit.** They hold a hidden state that maintains information about previous inputs in a sequence. Recurrent units can "remember" information from prior steps by feeding back their hidden state, allowing them to capture dependencies across time.

**2. RNN Unfolding**

RNN unfolding or unrolling is the process of expanding the recurrent structure over time steps. During unfolding each step of the sequence is represented as a separate layer in a series illustrating how information flows across each time step.

This unrolling enables [**backpropagation through time (BPTT)**](https://www.geeksforgeeks.org/machine-learning/ml-back-propagation-through-time/) a learning process where errors are propagated across time steps to adjust the network’s weights enhancing the RNN’s ability to learn dependencies within sequential data.



## **Recurrent Neural Network Architecture**

**1. Hidden State Calculation**:

h=σ(U⋅X+W⋅ht−1+B)

* *h* represents the current hidden state.
* *U* and
* *W* are weight matrices.
* *B* is the bias.

**2. Output Calculation**:

Y=O(V⋅h+C)

The output *Y* is calculated by applying *O* an activation function to the weighted hidden state where *V* and *C* represent weights and bias.

**3. Overall Function**:

Y=f(X,h,W,U,V,B,C)

This function defines the entire RNN operation where the state matrix *S* holds each element Si representing the network's state at each time step *i*.

## **Types Of Recurrent Neural Networks**

### **1. One-to-One RNN**

This is the simplest type of neural network architecture where there is a single input and a single output. It is used for straightforward classification tasks such as binary classification where no sequential data is involved.

### **2. One-to-Many RNN**

### In a One-to-Many RNN the network processes a single input to produce multiple outputs over time. This is useful in tasks where one input triggers a sequence of predictions (outputs). For example in image captioning a single image can be used as input to generate a sequence of words as a caption.

### **3. Many-to-One RNN**

The **Many-to-One RNN** receives a sequence of inputs and generates a single output. This type is useful when the overall context of the input sequence is needed to make one prediction. In sentiment analysis the model receives a sequence of words (like a sentence) and produces a single output like positive, negative or neutral.

### **4. Many-to-Many RNN**

The **Many-to-Many RNN** type processes a sequence of inputs and generates a sequence of outputs. In language translation task a sequence of words in one language is given as input and a corresponding sequence in another language is generated as output.

## **Variants of Recurrent Neural Networks (RNNs)**

There are several variations of RNNs, each designed to address specific challenges or optimize for certain tasks:

### **1. Vanilla RNN**

This simplest form of RNN consists of a single hidden layer where weights are shared across time steps. Vanilla RNNs are suitable for learning short-term dependencies but are limited by the vanishing gradient problem, which hampers long-sequence learning.

### **2. Bidirectional RNNs**

[Bidirectional RNNs](https://www.geeksforgeeks.org/bidirectional-recurrent-neural-network/) process inputs in both forward and backward directions, capturing both past and future context for each time step. This architecture is ideal for tasks where the entire sequence is available, such as named entity recognition and question answering.

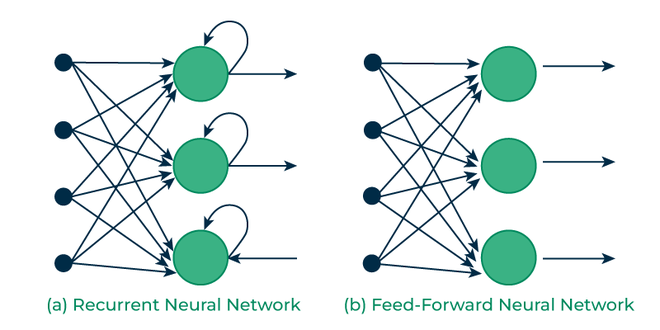
### **3. Long Short-Term Memory Networks (LSTMs)**

[Long Short-Term Memory Networks (LSTMs)](https://www.geeksforgeeks.org/deep-learning-introduction-to-long-short-term-memory/) introduce a memory mechanism to overcome the vanishing gradient problem. Each LSTM cell has three gates:

* **Input Gate**: Controls how much new information should be added to the cell state.
* **Forget Gate**: Decides what past information should be discarded.
* **Output Gate**: Regulates what information should be output at the current step. This selective memory enables LSTMs to handle long-term dependencies, making them ideal for tasks where earlier context is critical.

### **4. Gated Recurrent Units (GRUs)**

[Gated Recurrent Units (GRUs)](https://www.geeksforgeeks.org/machine-learning/gated-recurrent-unit-networks/) simplify LSTMs by combining the input and forget gates into a single update gate and streamlining the output mechanism. This design is computationally efficient, often performing similarly to LSTMs and is useful in tasks where simplicity and faster training are beneficial.



*Recurrent Vs Feedforward networks*

## **Advantages of Recurrent Neural Networks**

* **Sequential Memory**: RNNs retain information from previous inputs making them ideal for time-series predictions where past data is crucial.
* **Enhanced Pixel Neighborhoods**: RNNs can be combined with convolutional layers to capture extended pixel neighborhoods improving performance in image and video data processing.

## **Limitations of Recurrent Neural Networks (RNNs)**

While RNNs excel at handling sequential data they face two main training challenges i.e [vanishing gradient and exploding gradient problem](https://www.geeksforgeeks.org/deep-learning/vanishing-and-exploding-gradients-problems-in-deep-learning/):

1. **Vanishing Gradient**: During backpropagation gradients diminish as they pass through each time step leading to minimal weight updates. This limits the RNN’s ability to learn long-term dependencies which is crucial for tasks like language translation.
2. **Exploding Gradient**: Sometimes gradients grow uncontrollably causing excessively large weight updates that de-stabilize training.

These challenges can hinder the performance of standard RNNs on complex, long-sequence tasks.

## **Applications of Recurrent Neural Networks**

RNNs are used in various applications where data is sequential or time-based:

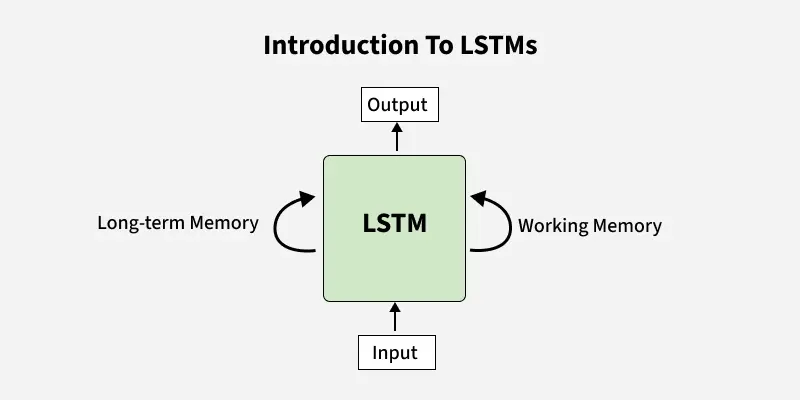
* **Time-Series Prediction**: RNNs excel in forecasting tasks, such as stock market predictions and weather forecasting.
* **Natural Language Processing (NLP)**: RNNs are fundamental in NLP tasks like language modeling, sentiment analysis and machine translation.
* **Speech Recognition**: RNNs capture temporal patterns in speech data, aiding in speech-to-text and other audio-related applications.
* **Image and Video Processing**: When combined with convolutional layers, RNNs help analyze video sequences, facial expressions and gesture recognition.

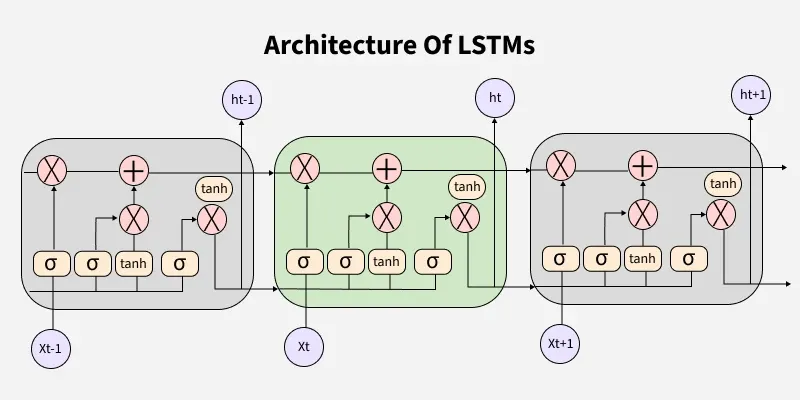
LSTM

**Long Short-Term Memory (LSTM)**

is an enhanced version of the [Recurrent Neural Network (RNN)](https://www.geeksforgeeks.org/introduction-to-recurrent-neural-network/) 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**

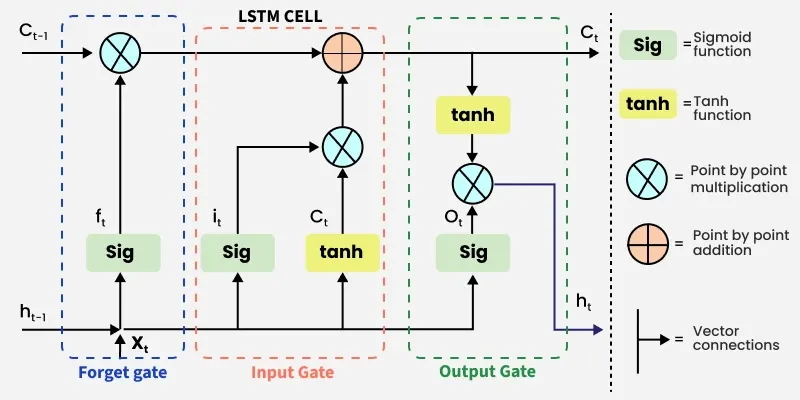
LSTM architectures involves the memory cell which is controlled by three gates:

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*

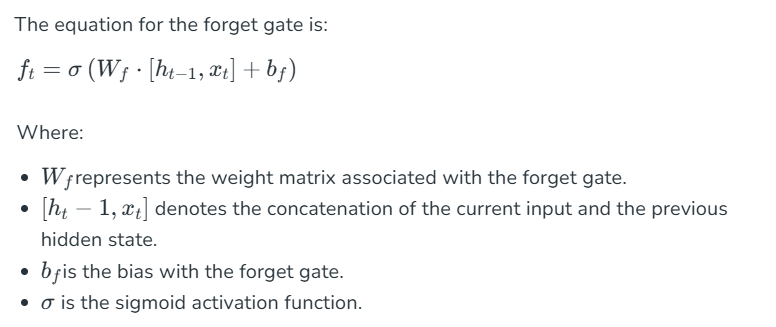
Information is retained by the cells and the memory manipulations are done by thegate

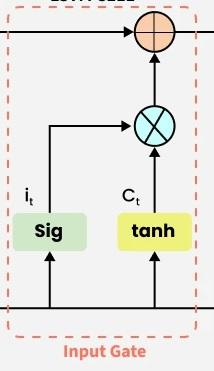
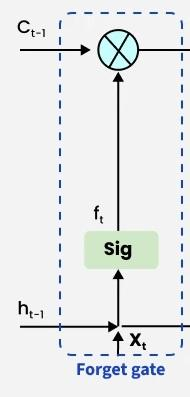
**1. Forget Gate**

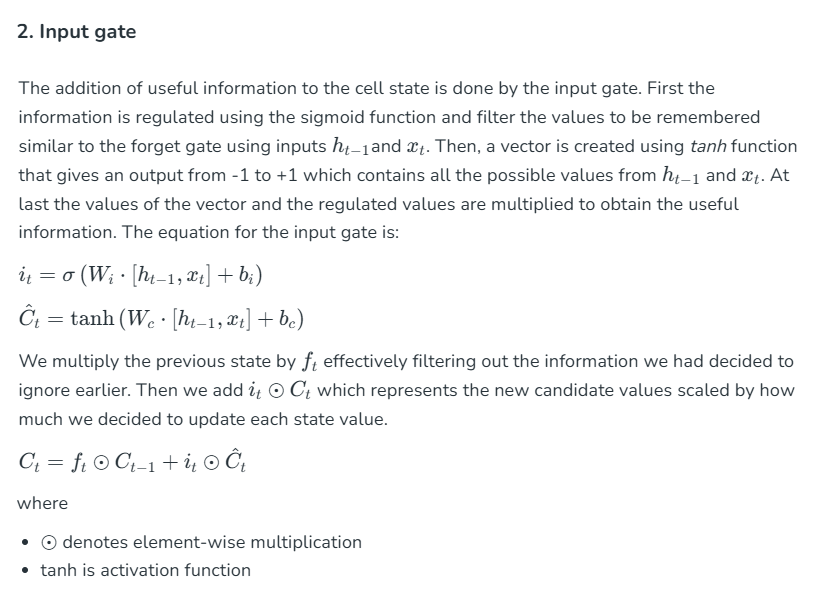
The information that is no longer useful in the cell state is removed with the forget gate. Two inputs Xt(input at the particular time) and *ht*−1(previous cell output) are fed to the gate and multiplied with weight matrices followed by the addition of bias. The resultant is passed through an activation function which gives a binary output. If for a particular cell state the output is 0, the piece of information is forgotten and for output 1, the information is retained for future use.

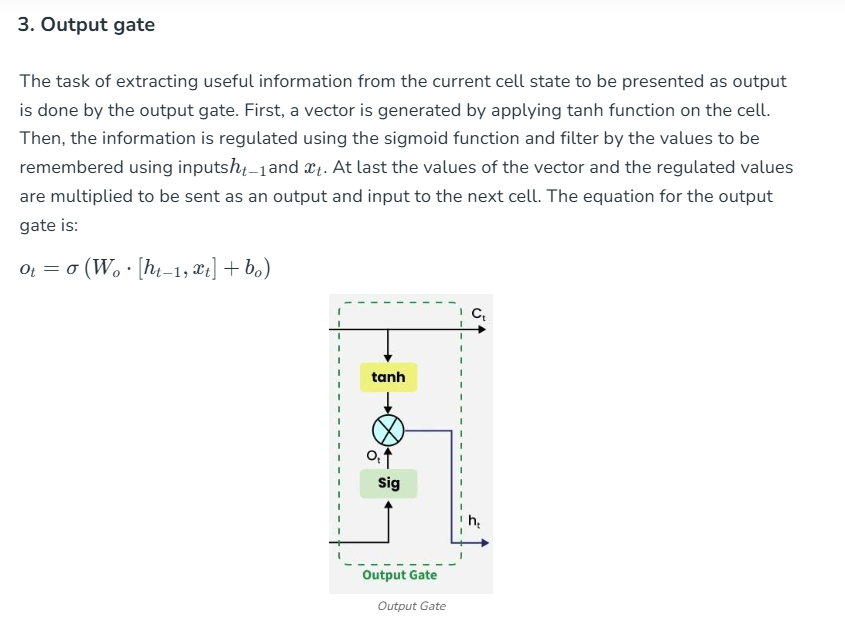
The equation for the forget gate is:

s**.** There are three gates -









## **Applications of LSTM**

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)](https://www.geeksforgeeks.org/introduction-convolution-neural-network/) they help analyze video data and extract useful information.

LSTM ChatGPT

**what is LSTM? Why we use it? Where We use it?? its Components, How it works?**

**LSTM** stands for **Long Short-Term Memory**. It's a special type of **Recurrent Neural Network (RNN)** designed to better learn and remember patterns in **sequential data**—especially when long-term memory is required.

## **🔍 What is LSTM?**

LSTM is a **deep learning architecture** used to process sequences of data, like time series, text, audio, or video. Traditional RNNs struggle with learning long-range dependencies due to the **vanishing gradient problem**, while LSTMs are specifically designed to overcome this issue by using **memory cells and gates** to retain relevant information over longer periods.

## **✅ Why Do We Use LSTM?**

We use LSTM because:

1. **Handles Long-Term Dependencies** – It can remember information over long sequences, which basic RNNs fail to do.
2. **Avoids Vanishing Gradient Problem** – Thanks to its gated structure.
3. **Works Well with Sequential Data** – Useful in tasks where order and context matter (e.g., sentences, stock prices, etc.).

## **📍 Where Is LSTM Used?**

LSTMs are widely used in:

* **Natural Language Processing (NLP)**
  + Language translation
  + Text generation
  + Sentiment analysis
* **Speech Recognition**
* **Time Series Forecasting**
  + Stock price prediction
  + Weather prediction
* **Anomaly Detection**
* **Music and Video Generation**
* **Healthcare** (e.g., predicting disease progression using patient data)

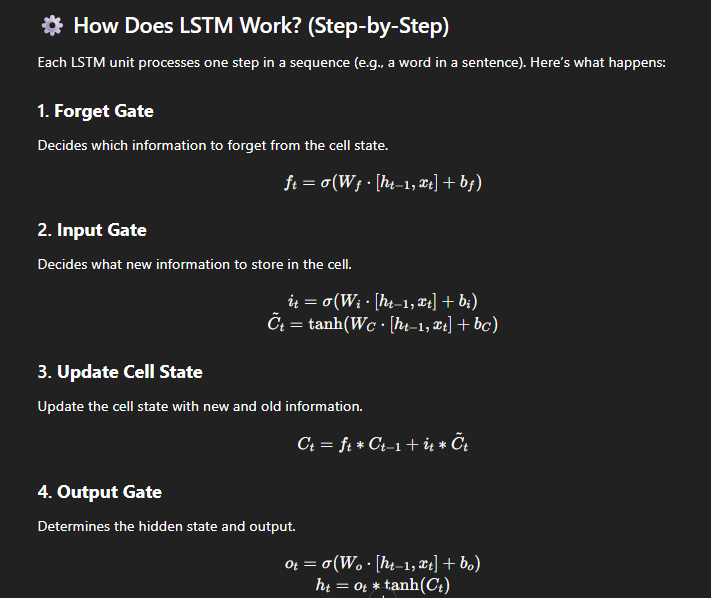
## **🧩 Components of LSTM**

An LSTM unit has the following **key components**:

1. **Cell State (CtC\_t)**: Acts like a conveyor belt carrying information forward.
2. **Hidden State (hth\_t)**: Output at the current time step.
3. **Forget Gate**: Decides what information to discard from the cell state.
4. **Input Gate**: Decides what new information to add to the cell state.
5. **Output Gate**: Decides what to output from the current cell.

## **⚙️ How Does LSTM Work? (Step-by-Step)**

Each LSTM unit processes one step in a sequence (e.g., a word in a sentence). Here’s what happens:



## **🧠 Summary (In Simple Terms)**

* LSTM remembers **important things** from earlier in the sequence.
* It **forgets unimportant things** using gates.
* It’s **great for tasks** where time, order, or context matter.

Would you like a **visual diagram** of an LSTM cell or a **real-world example** (like how it's used in sentiment analysis or time-series prediction)?