Recurrent Neural Networks

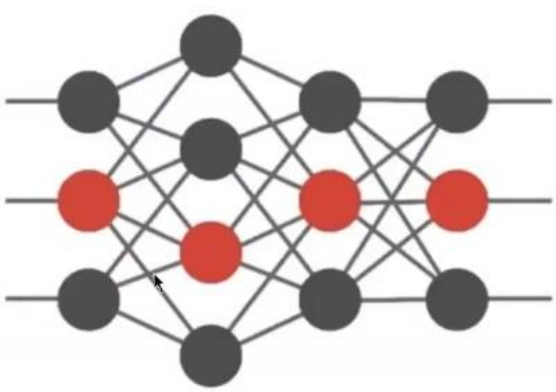
Supervised	Artificial Neural Networks	Used for Regression & Classification
	Convolutional Neural Networks	Used for Computer Vision
	Recurrent Neural Networks	Used for Time Series Analysis

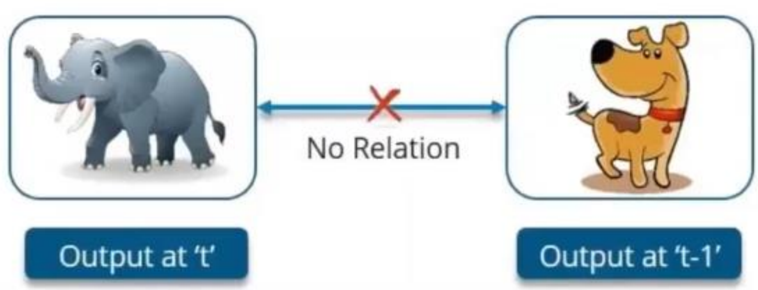
Agenda

- Why Not Feedforward Networks?
- What Is Recurrent Neural Network?
- Issues With Recurrent Neural Networks
- Vanishing And Exploding Gradient
- How To Overcome These Challenges?
- Long Short Term Memory Units
- LSTM Use-Case

Why Not Feedforward Networks?

A trained feedforward network can be exposed to any random collection of photographs, and the first photograph it is exposed to will not necessarily alter how it classifies the second

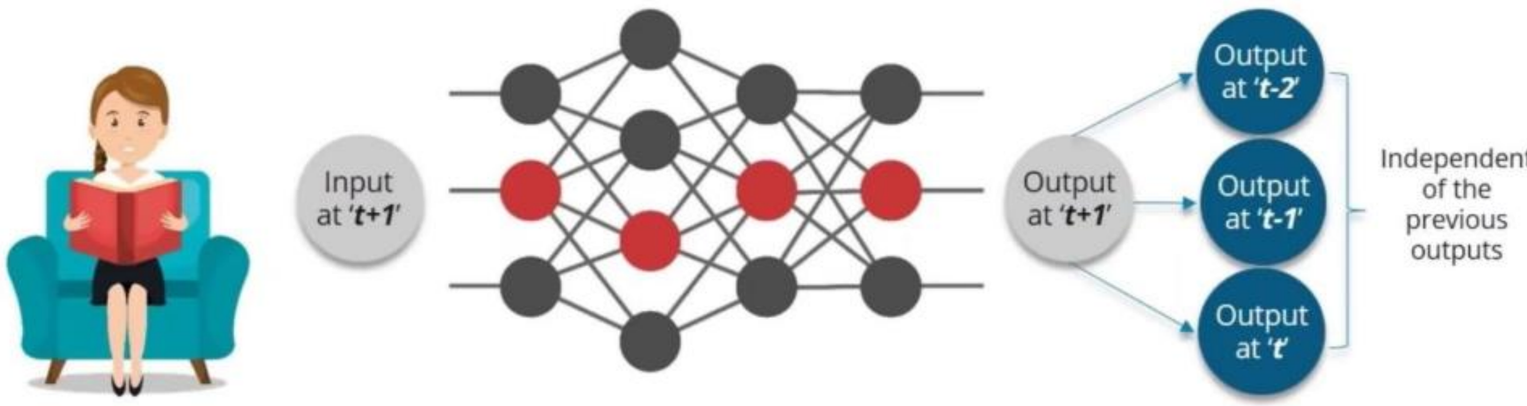




Seeing photograph of a dog will not lead the net to perceive an elephant next

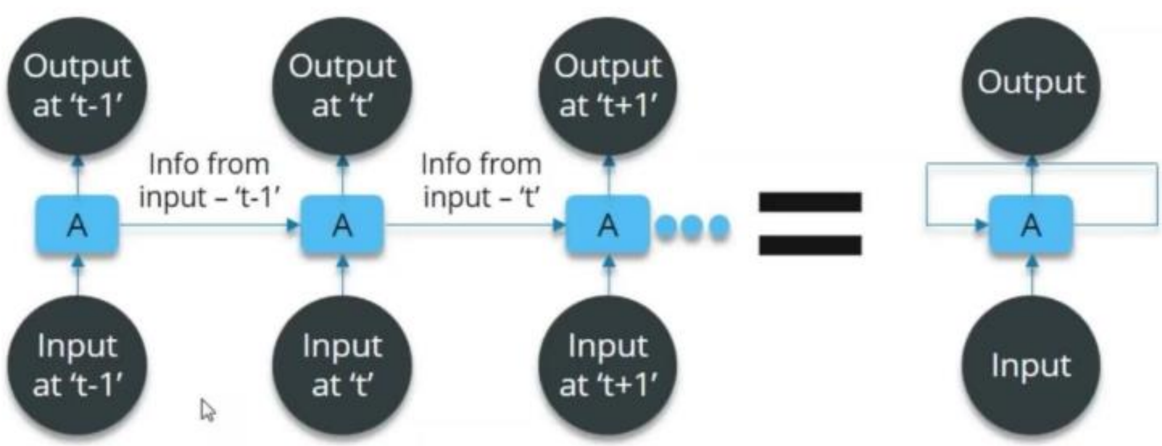
Why Not Feedforward Networks?

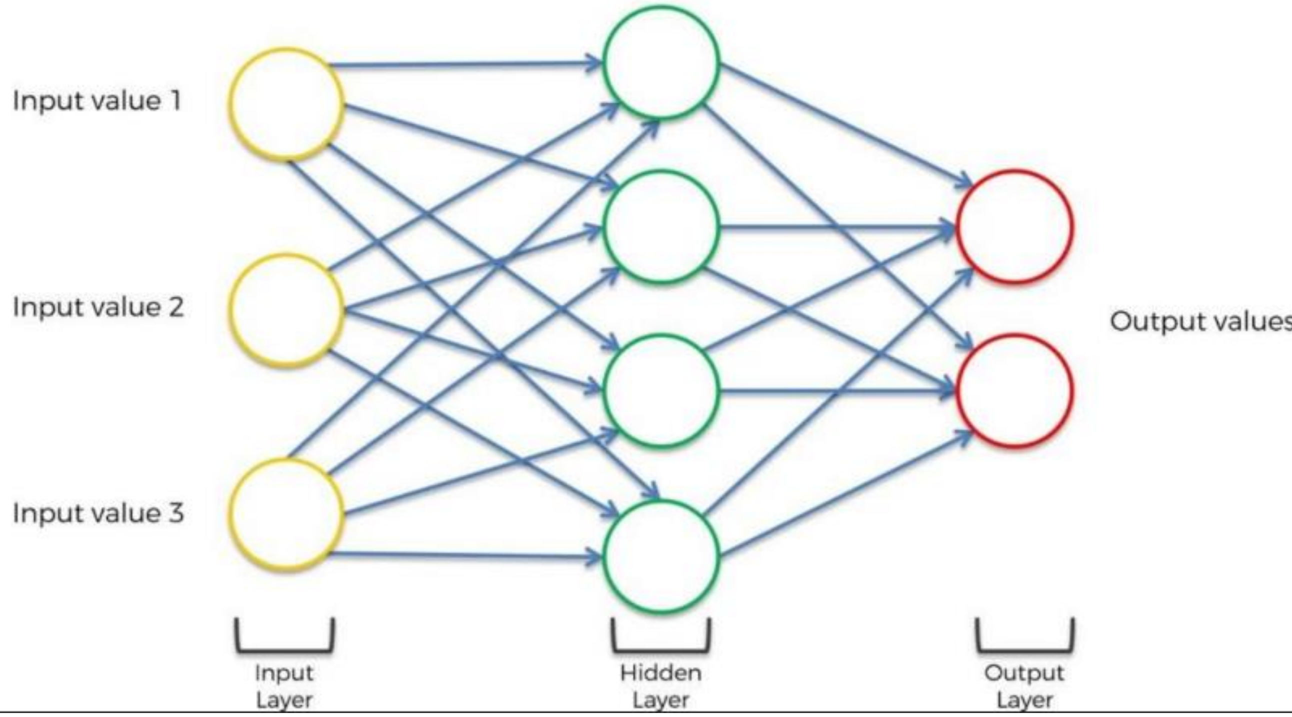
When you read a book, you understand it based on your understanding of previous words

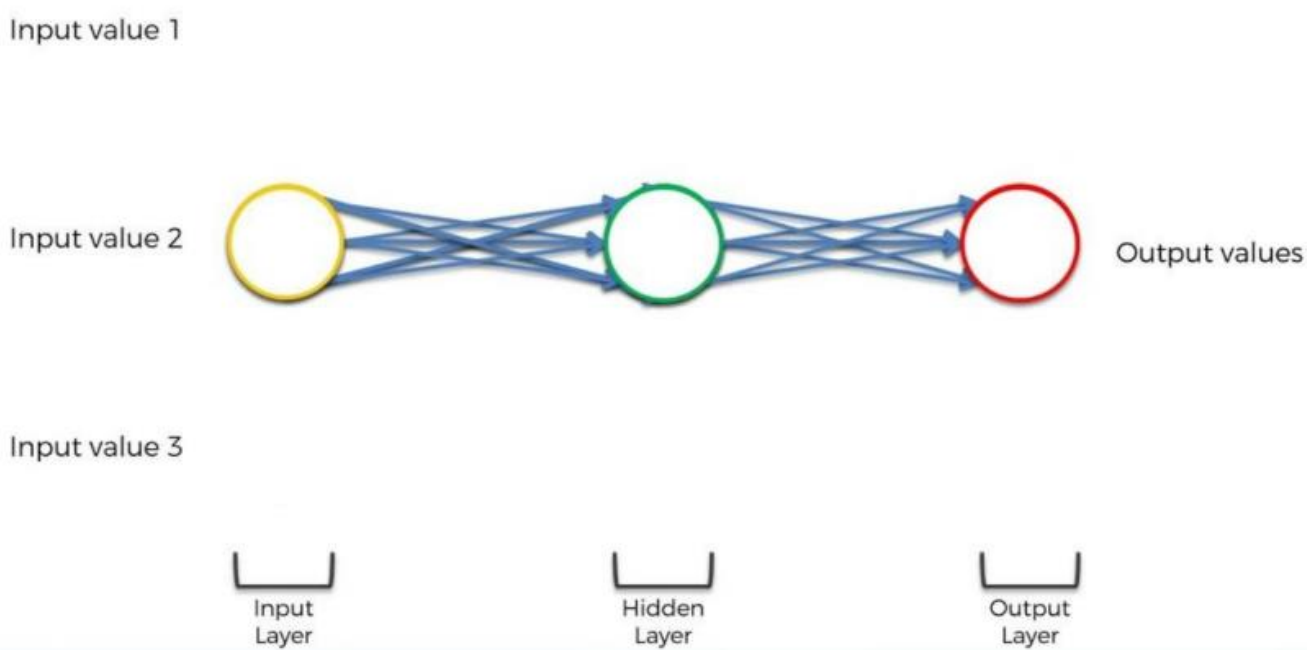


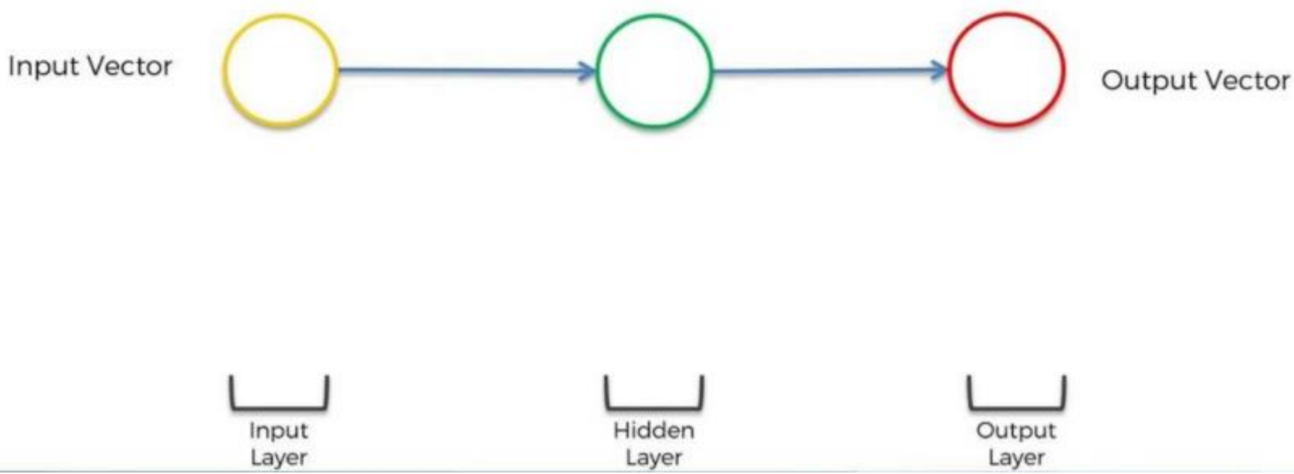
I cannot predict the next word in a sentence if I use feedforward nets

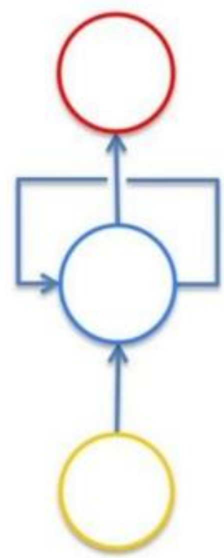
How To Overcome This Challenge?

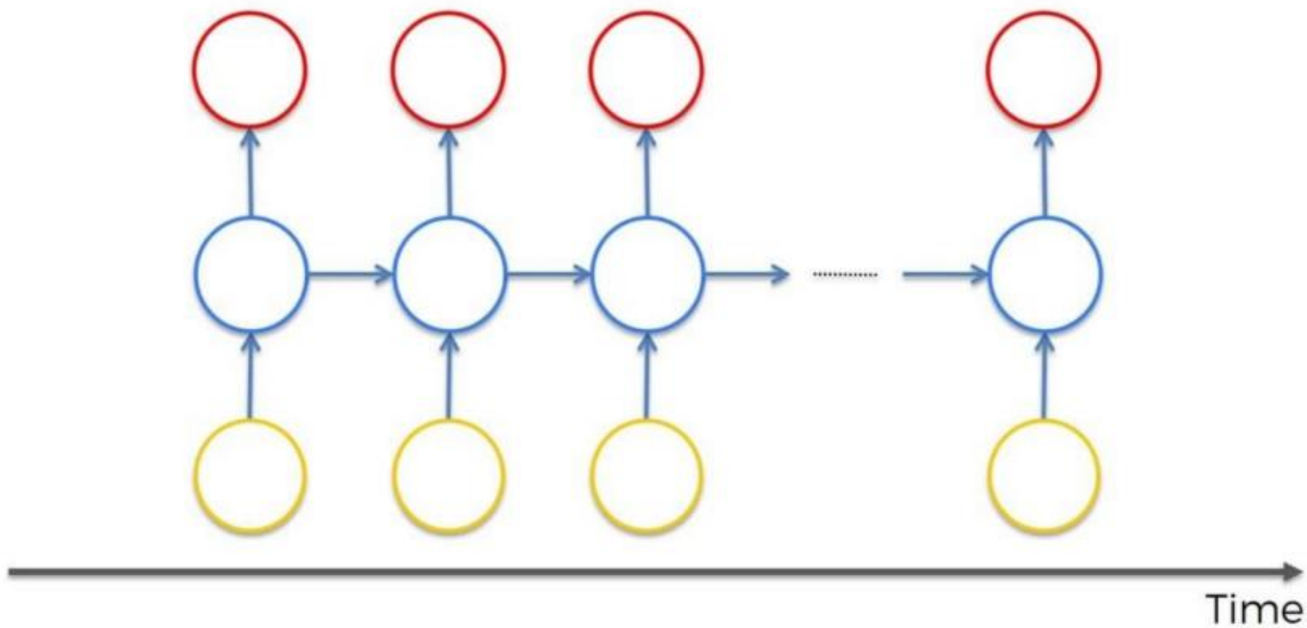


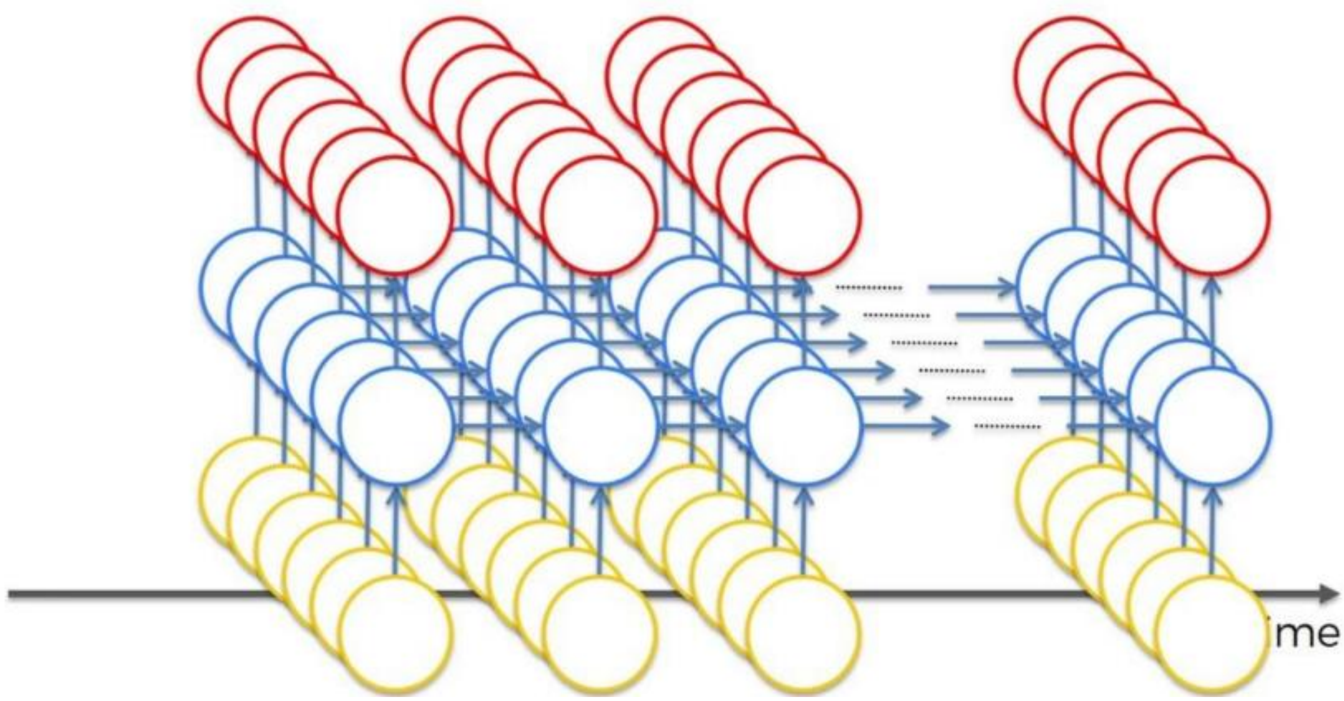

Input value 1
Input value 2
Input value 3
Output values
Input Layer
Hidden Layer
Output Layer


Input value 1
Input value 2
Input value 3
Output values
Input Layer
Hidden Layer
Output Layer


Input Vector
Output Vector
Input
Layer
Hidden
Layer
Output
Layer



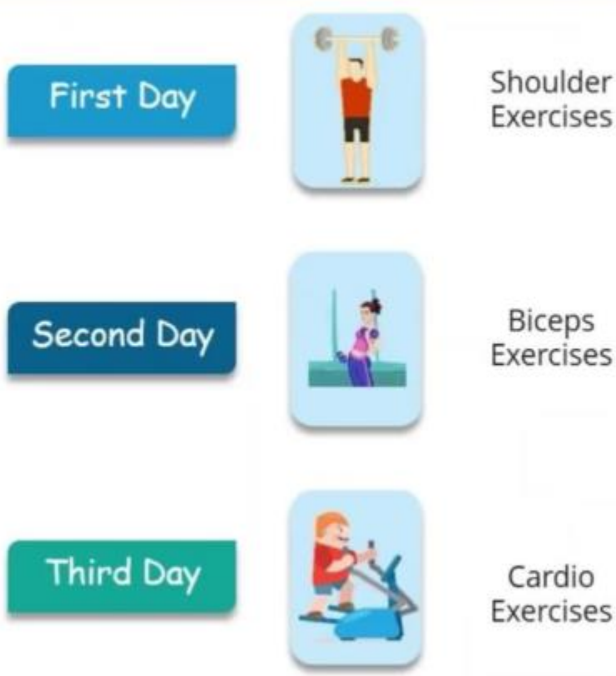

Time



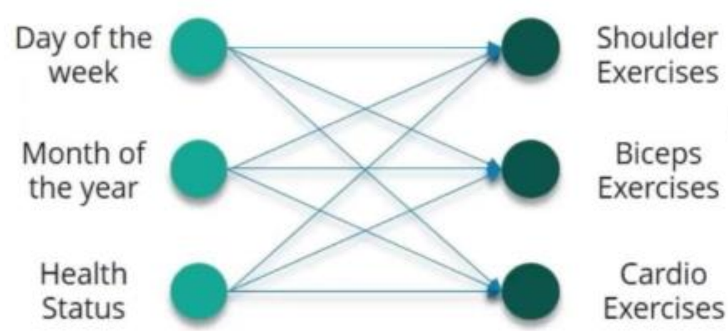
What Is Recurrent Neural Network?

Recurrent Networks are a type of artificial neural network designed to recognize patterns in sequences of data, such as text, genomes, handwriting, the spoken word, or numerical times series data emanating from sensors, stock markets and government agencies.

What Is Recurrent Neural Network?

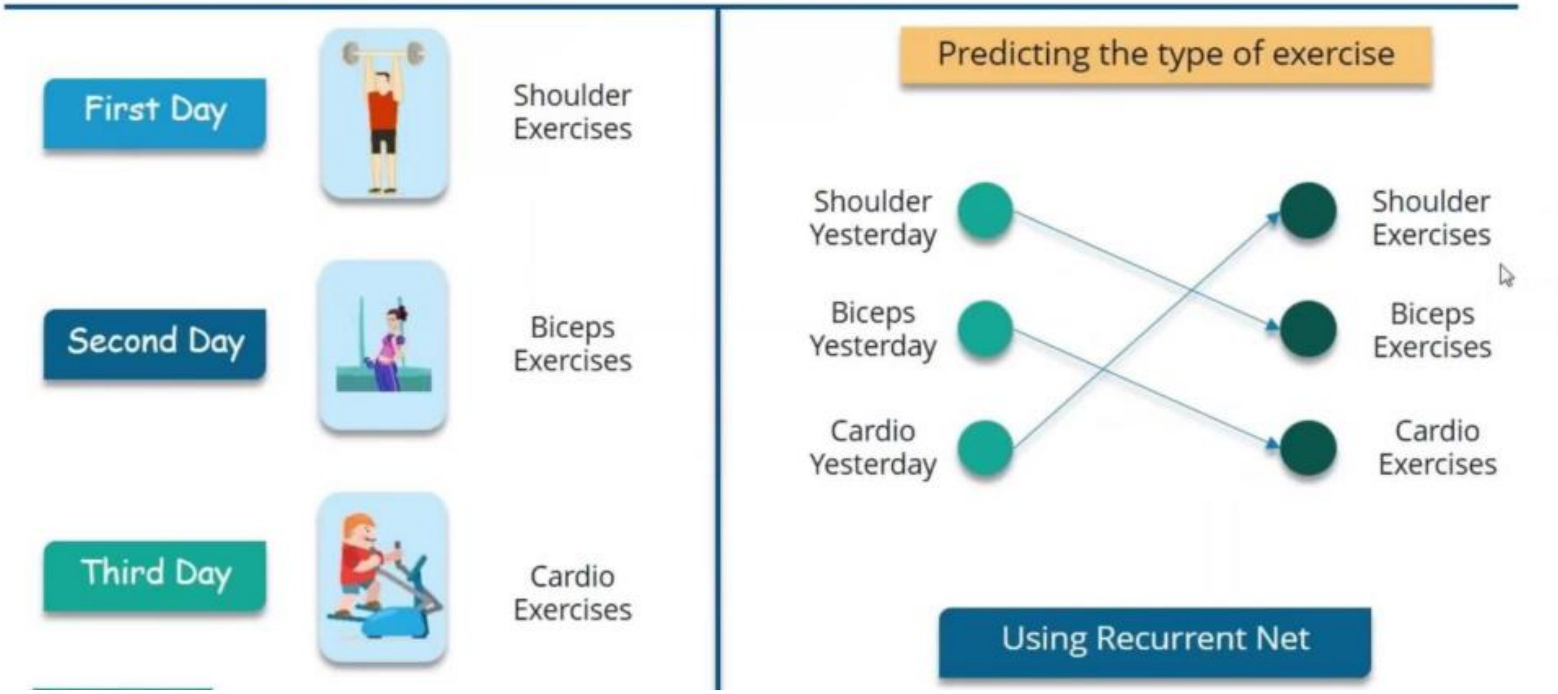


Predicting the type of exercise

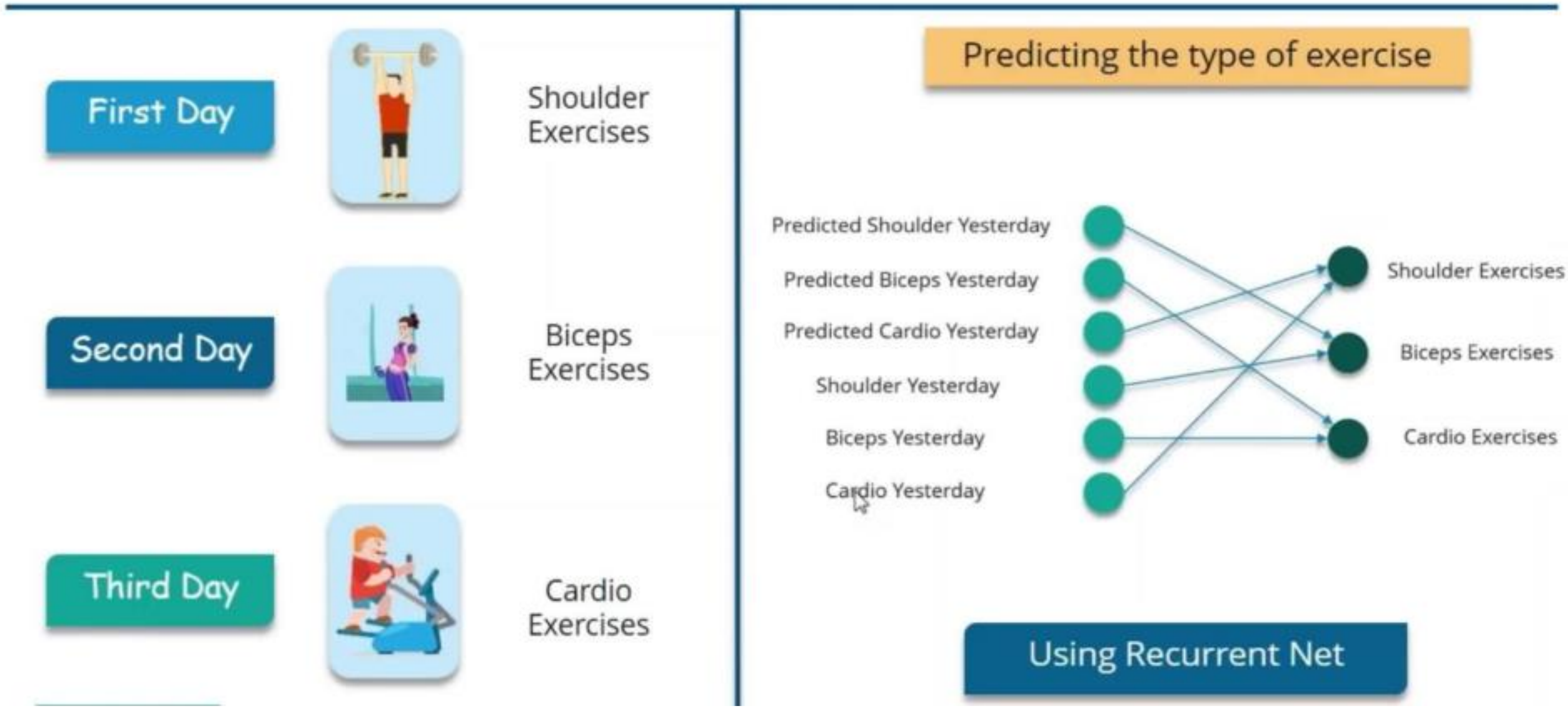


Using Feedforward Net

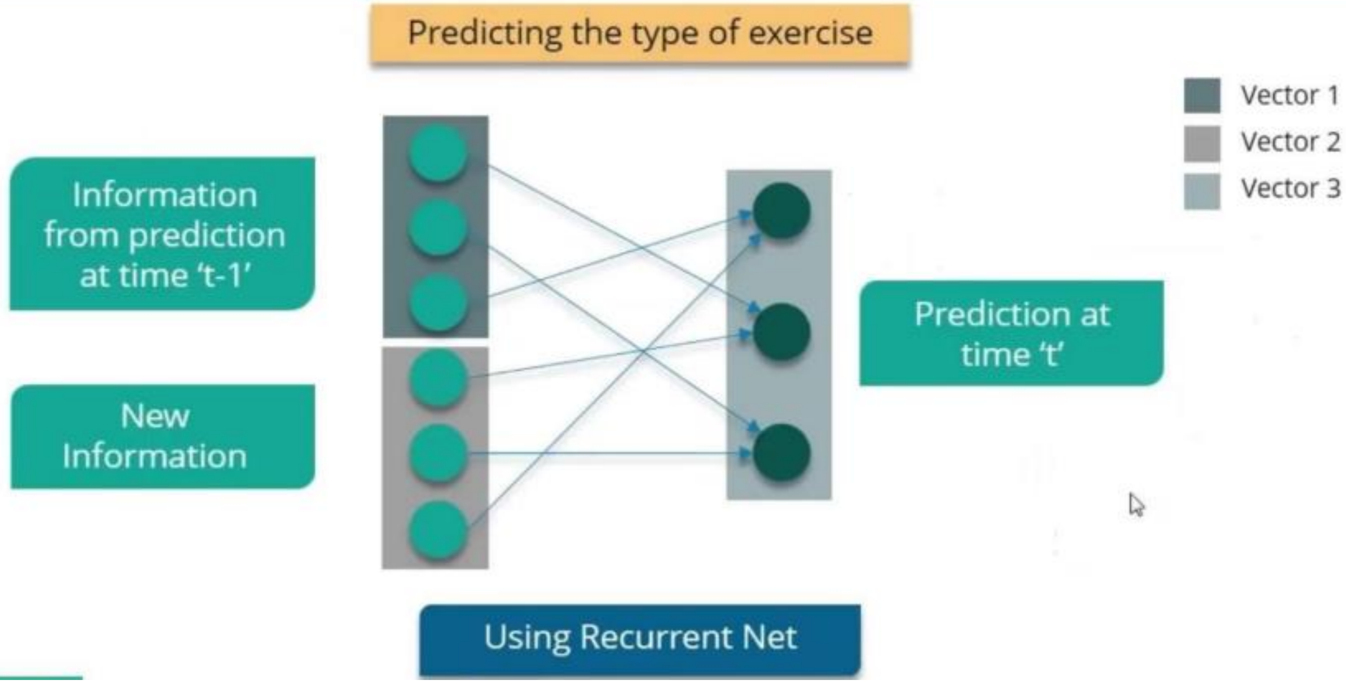
What Is Recurrent Neural Network?



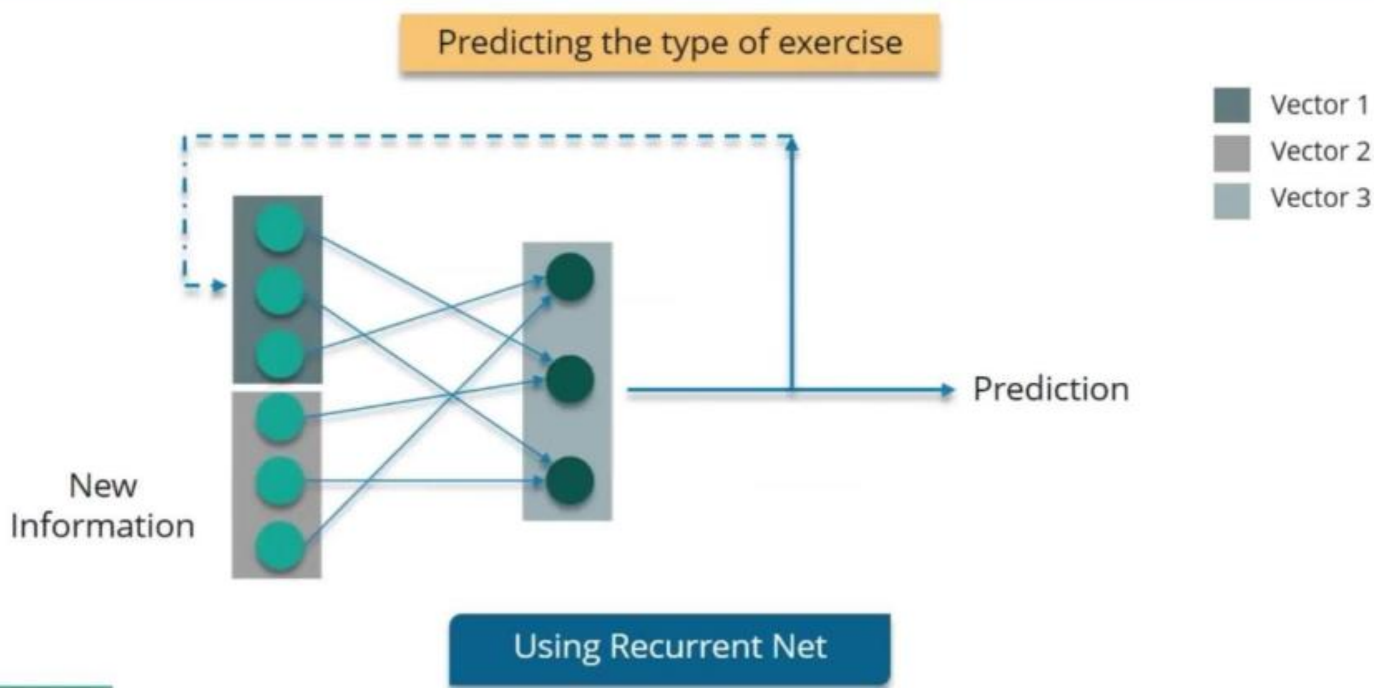
What Is Recurrent Neural Network?



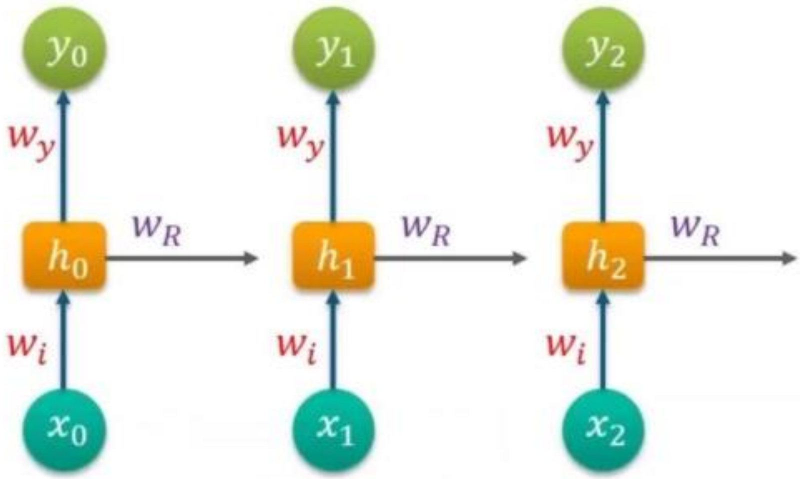
What Is Recurrent Neural Network?



What Is Recurrent Neural Network?



What Is Recurrent Neural Network?



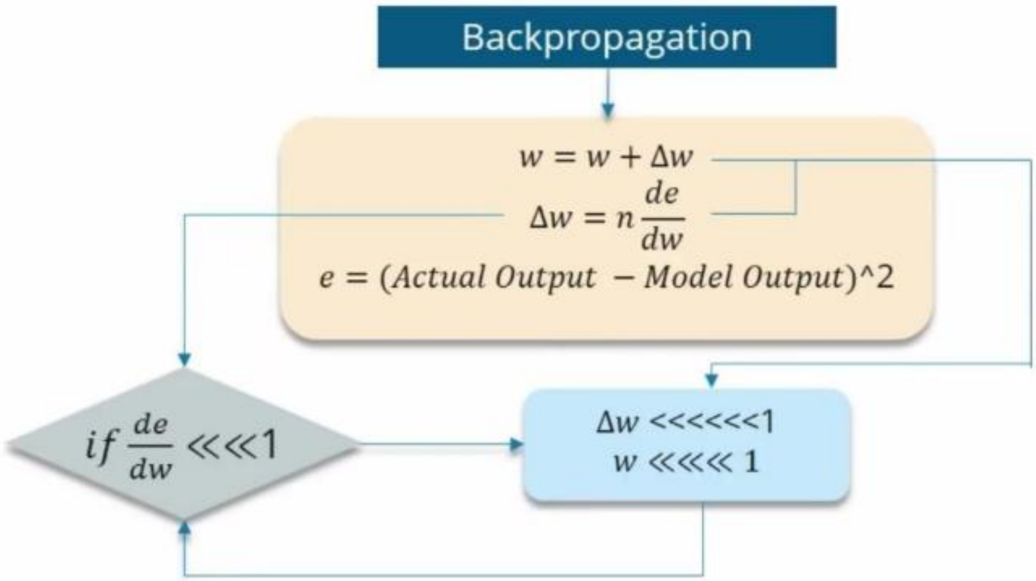
$$h^{(t)} = g_h\ (w_i x^{(t)} + w_R h^{(t-1)} + b_h)$$

$$y^{(t)} = g_y\ (w_y h^{(t)} + b_y)$$

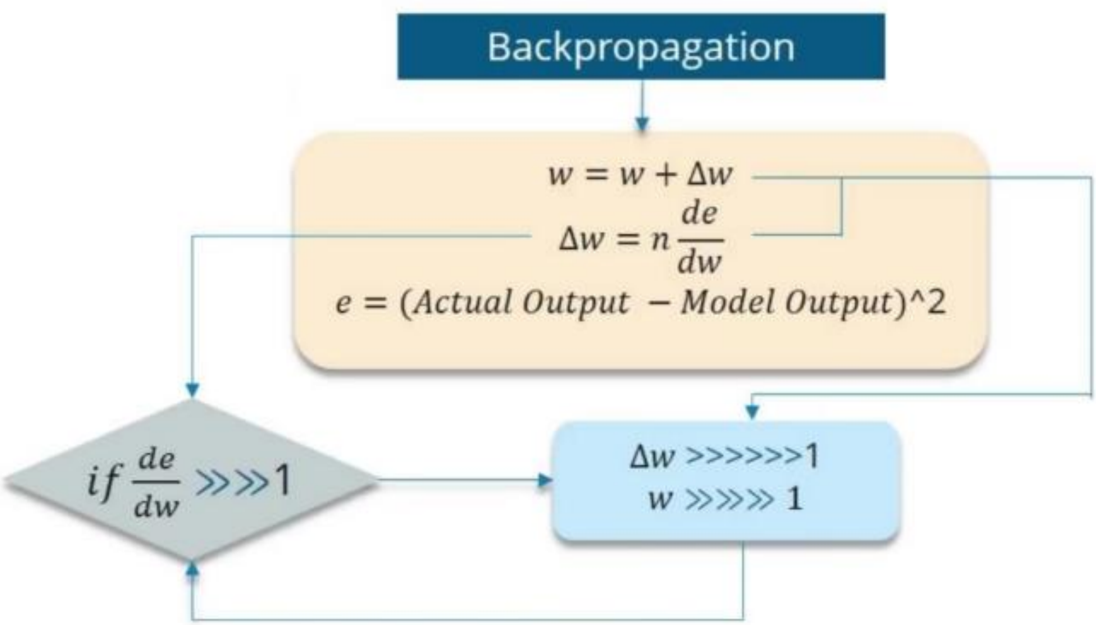
Training A Recurrent Neural Network

Recurrent Neural Nets uses backpropagation algorithm, but it is applied for every time stamp. It is commonly known as Backpropagation Through Time (BTT).

Vanishing Gradient



Exploding Gradient



How To Overcome These Challenges?

Exploding gradients

- *Truncated BTT*

  Instead of starting backpropagation at the last time stamp, we can choose a smaller time stamp like 10 (we will lose the temporal context after 10 time stamps)
- *Clip gradients at threshold*

  Clip the gradient when it goes higher than a threshold
- *RMSprop to adjust learning rate*

Vanishing gradients

- *ReLU activation function*

  We can use activation functions like ReLU, which gives output one while calculating gradient
- *RMSprop*

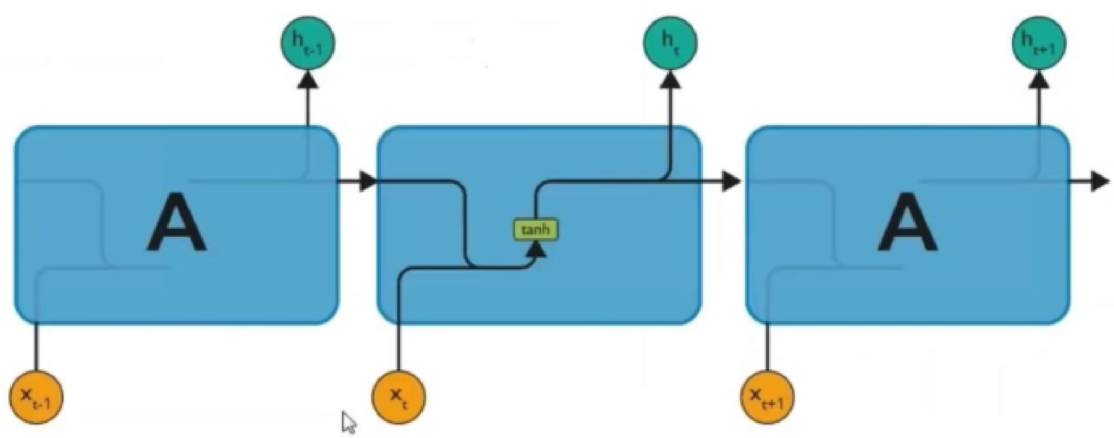
  Clip the gradient when it goes higher than a threshold
- *LSTM, GRUs*

  Different network architectures that has been specially designed can be used to combat this problem

Long Short Term Memory Networks

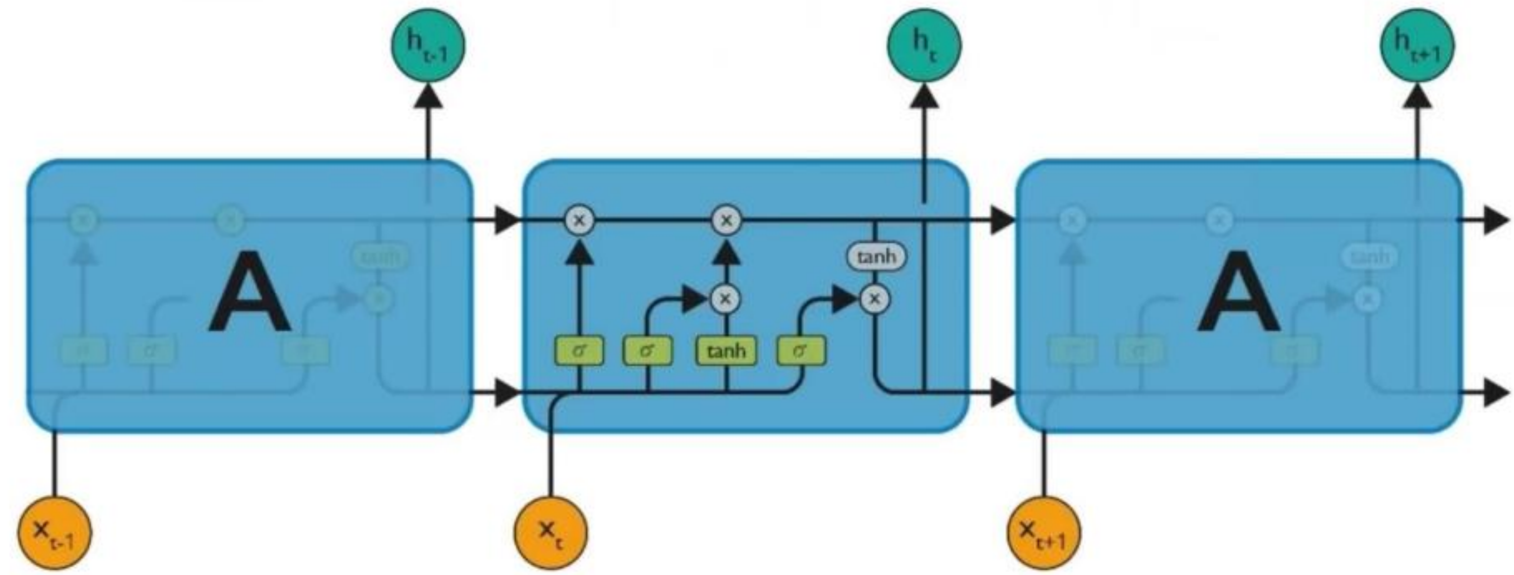
✓ Long Short Term Memory networks – usually just called "LSTMs" – are a special kind of RNN.

✓ They are capable of learning long-term dependencies.



The repeating module in a standard RNN contains a single layer

Long Short Term Memory Networks



Long Short-Term Memory

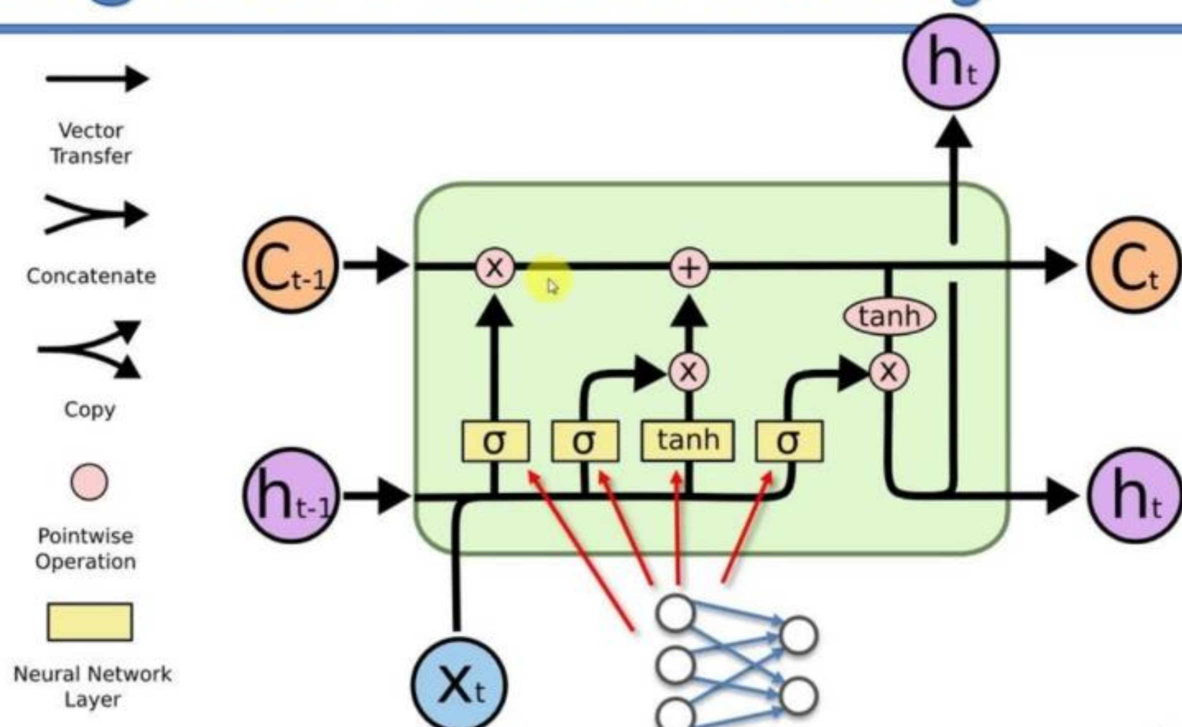


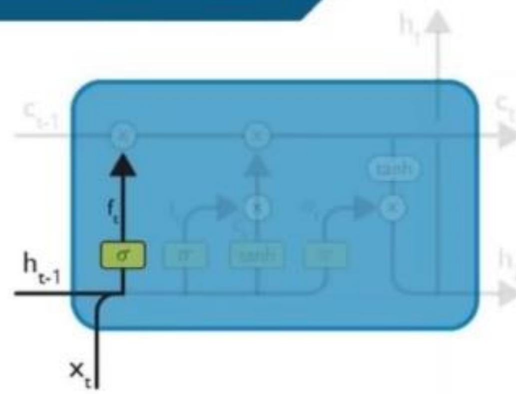
Image Source: colah.github.io

Long Short Term Memory Networks

Step-1

The first step in the **LSTM** is to identify those information that are not required and will be thrown away from the cell state. This decision is made by a sigmoid layer called as forget gate layer.

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

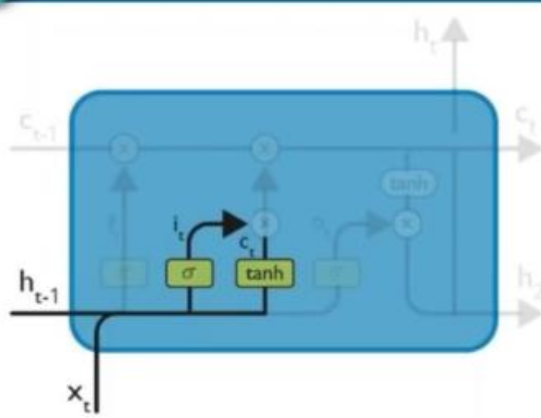


$$f_t = \sigma(w_f[h_{t-1}, x_t] + b_f)$$

Long Short Term Memory Networks

Step-2

The next step is to decide, what new information we're going to store in the cell state. This whole process comprises of following steps. A **sigmoid layer** called the "input gate layer" decides which values will be updated. Next, a **tanh layer** creates a vector of new candidate values, that could be added to the state.



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

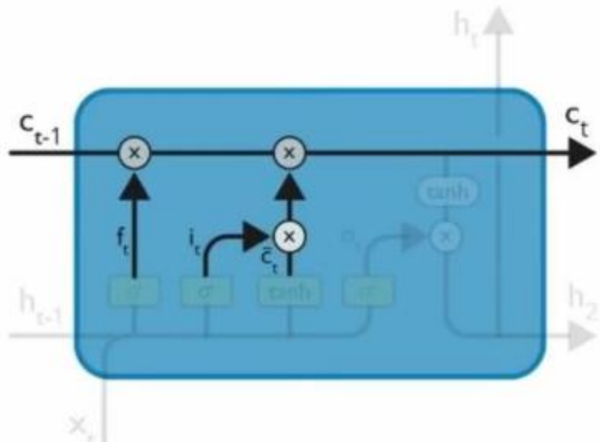
$$c^{\sim}_t = tanh(w_c[h_{t-1}, x_t] + b_c)$$

In the next step, we'll combine these two to update the state.

Long Short Term Memory Networks

Step-3

Now, we will update the old cell state, C_{t-1}, into the new cell state 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 $i_t * \tilde{c}_t$. This is the new candidate values, scaled by how much we decided to update each state value.

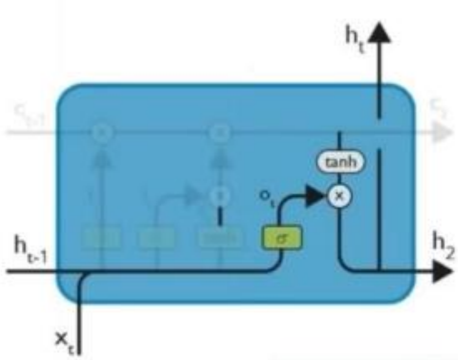


$$c_t = f_t * c_{t-1} + i_t^* \tilde{c}_t$$

Long Short Term Memory Networks

Step-4

We will run a sigmoid layer which decides what parts of the cell state we're 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 decided to.



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

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