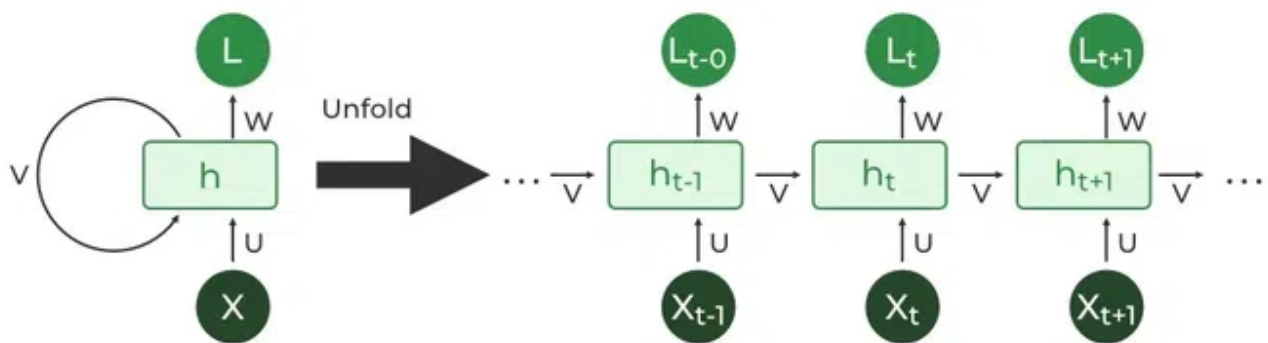


Introduction to Recurrent Neural Network

In this article, we will introduce a new variation of neural network which is the **Recurrent Neural Network** also known as (**RNN**) that works better than a simple neural network when data is sequential like Time-Series data and text data.

What is Recurrent Neural Network (RNN)?

Recurrent Neural Network(RNN) is a type of Neural Network where the output from the previous step is fed as input to the current step. In traditional neural networks, all the inputs and outputs are independent of each other. Still, in cases when it is required to predict the next word of a sentence, the previous words are required and hence there is a need to remember the previous words. Thus RNN came into existence, which solved this issue with the help of a Hidden Layer. The main and most important feature of RNN is its **Hidden state**, which remembers some information about a sequence. The state is also referred to as *Memory State* since it remembers the previous input to the network. It uses the same parameters for each input as it performs the same task on all the inputs or hidden layers to produce the output. This reduces the complexity of parameters, unlike other neural networks.

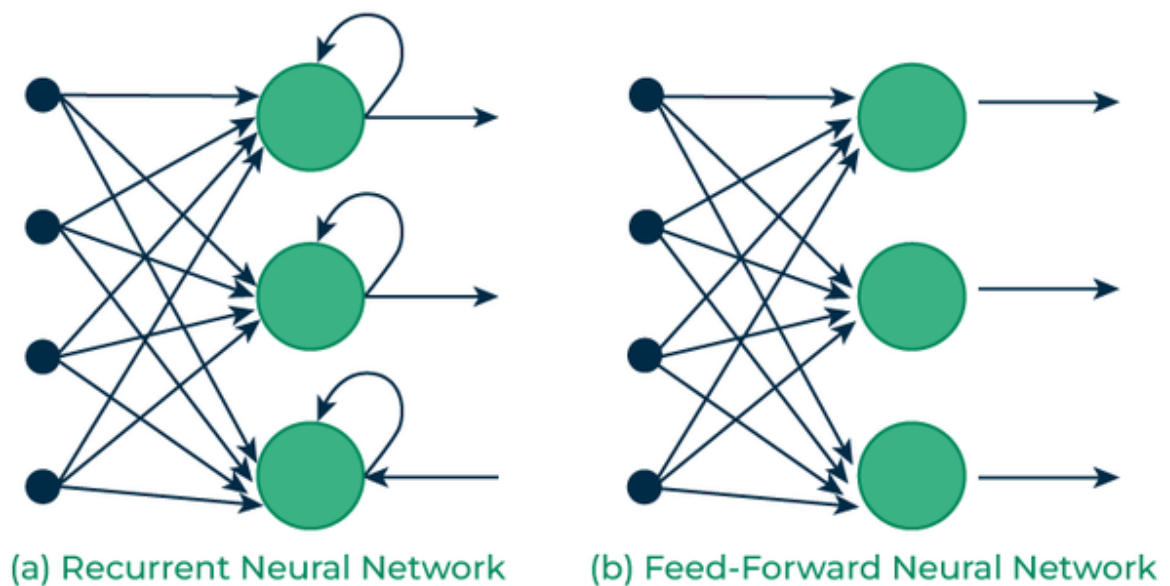


Recurrent Neural Network

How RNN differs from Feedforward Neural Network?

Artificial neural networks that do not have looping nodes are called feed forward neural networks. Because all information is only passed forward, this kind of neural network is also referred to as a multi-layer neural network.

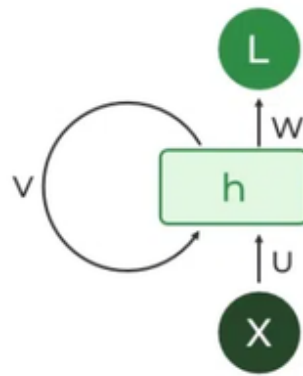
Information moves from the input layer to the output layer – if any hidden layers are present – unidirectionally in a feedforward neural network. These networks are appropriate for image classification tasks, for example, where input and output are independent. Nevertheless, their inability to retain previous inputs automatically renders them less useful for sequential data analysis.



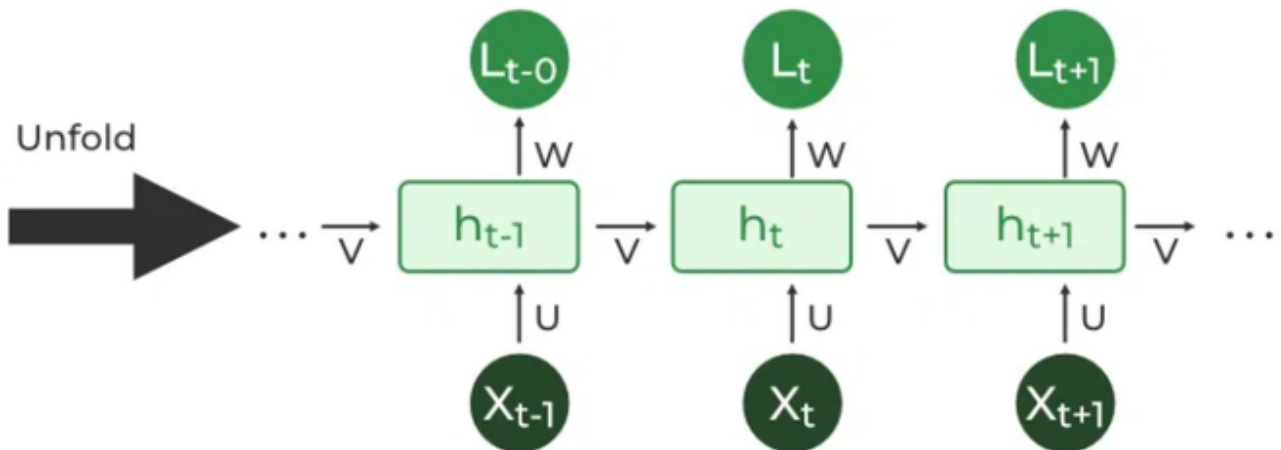
Recurrent Vs Feedforward networks

Recurrent Neuron and RNN Unfolding

The fundamental processing unit in a Recurrent Neural Network (RNN) is a Recurrent Unit, which is not explicitly called a “Recurrent Neuron.” This unit has the unique ability to maintain a hidden state, allowing the network to capture sequential dependencies by remembering previous inputs while processing. Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) versions improve the RNN’s ability to handle long-term dependencies.



Recurrent Neuron



RNN Unfolding

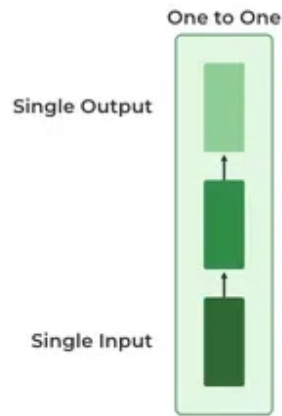
Types Of RNN

There are four types of RNNs based on the number of inputs and outputs in the network.

1. One to One
2. One to Many
3. Many to One
4. Many to Many

One to One

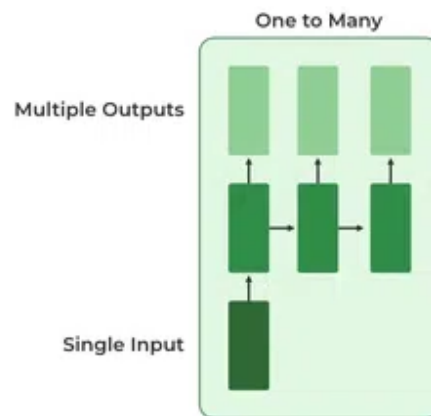
This type of RNN behaves the same as any simple Neural network it is also known as Vanilla Neural Network. In this Neural network, there is only one input and one output.



One to One RNN

One To Many

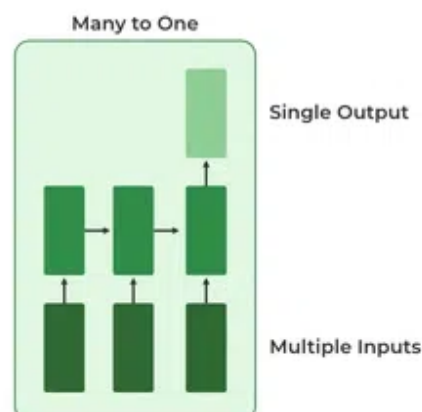
In this type of RNN, there is one input and many outputs associated with it. One of the most used examples of this network is Image captioning where given an image we predict a sentence having Multiple words.



One to Many RNN

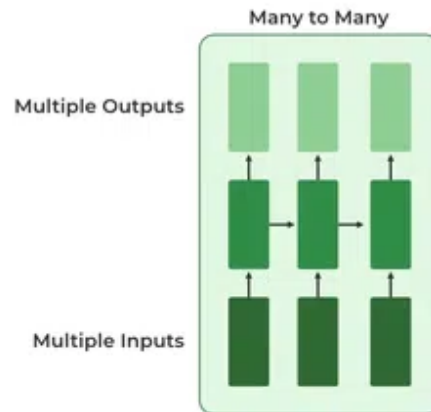
Many to One

In this type of network, Many inputs are fed to the network at several states of the network generating only one output. This type of network is used in the problems like sentimental analysis. Where we give multiple words as input and predict only the sentiment of the sentence as output.



Many to Many

In this type of neural network, there are multiple inputs and multiple outputs corresponding to a problem. One Example of this Problem will be language translation. In language translation, we provide multiple words from one language as input and predict multiple words from the second language as output.



Many to Many RNN

Recurrent Neural Network Architecture

RNNs have the same input and output architecture as any other deep neural architecture. However, differences arise in the way information flows from input to output. Unlike Deep neural networks where we have different weight matrices for each Dense network in RNN, the weight across the network remains the same. It calculates state hidden state H_i for every input X_i . **By using the following formulas:**

$$h = \sigma(UX + Wh_{-1} + B)$$

$$Y = O(Vh + C)$$

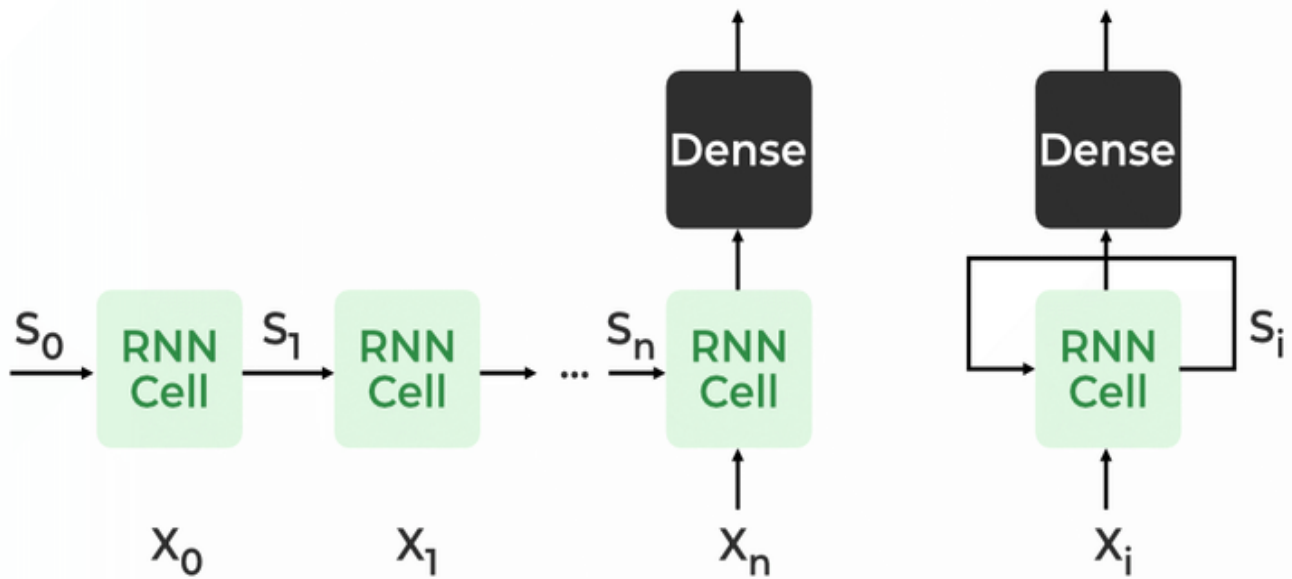
Hence

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

Here S is the State matrix which has element s_i as the state of the network at timestep i

The parameters in the network are W, U, V, c, b which are shared across timestep

RECURRENT NEURAL NETWORKS



Recurrent Neural Architecture

How does RNN work?

The Recurrent Neural Network consists of multiple fixed activation function units, one for each time step. Each unit has an internal state which is called the hidden state of the unit. This hidden state signifies the past knowledge that the network currently holds at a given time step. This hidden state is updated at every time step to signify the change in the knowledge of the network about the past. The hidden state is updated using the following recurrence relation:-

The formula for calculating the current state:

$$h_t = f(h_{t-1}, x_t)$$

where,

- h_t -> current state
- h_{t-1} -> previous state
- x_t -> input state

Formula for applying Activation function(tanh)

$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$

where,

- w_{hh} -> weight at recurrent neuron
- w_{xh} -> weight at input neuron

The formula for calculating output:

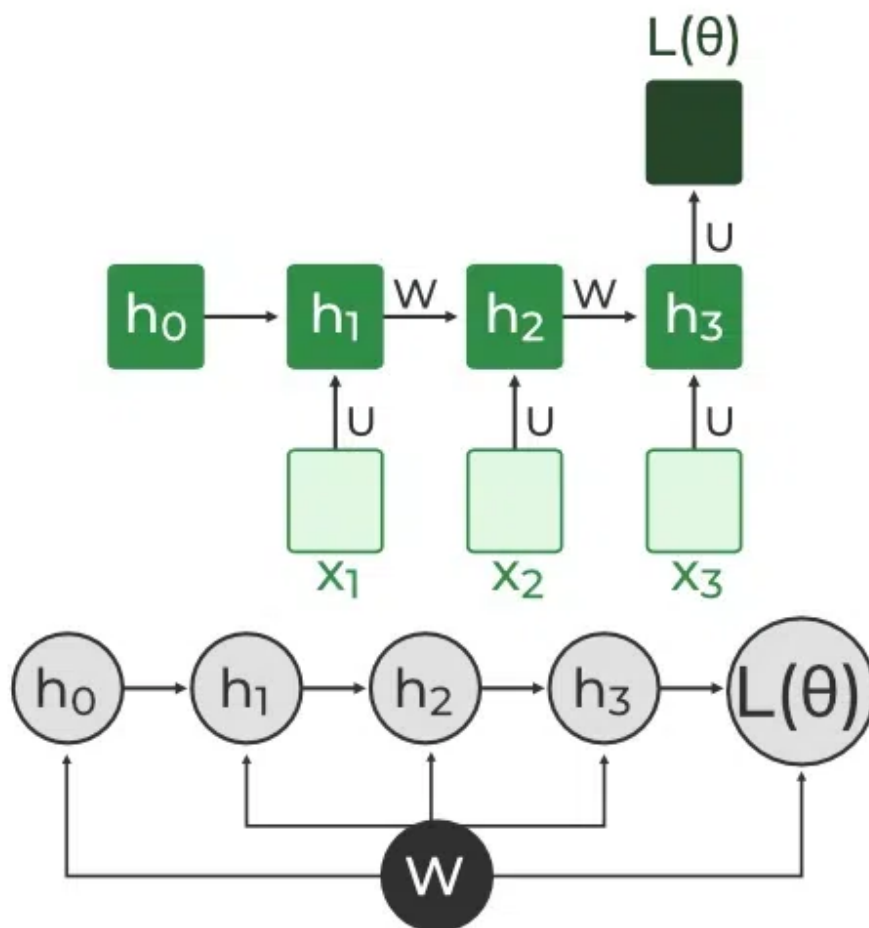
$$y_t = W_{hy}h_t$$

- Y_t -> output
- W_{hy} -> weight at output layer

These parameters are updated using Backpropagation. However, since RNN works on sequential data here we use an updated backpropagation which is known as Backpropagation through time.

Backpropagation Through Time (BPTT)

In RNN the neural network is in an ordered fashion and since in the ordered network each variable is computed one at a time in a specified order like first h_1 then h_2 then h_3 so on. Hence we will apply backpropagation throughout all these hidden time states sequentially.



Backpropagation Through Time (BPTT) In RNN

- $L(\theta)$ (loss function) depends on h_3
- h_3 in turn depends on h_2 and W
- h_2 in turn depends on h_1 and W
- h_1 in turn depends on h_0 and W
- where h_0 is a constant starting state.

$$\frac{\partial \mathbf{L}(\theta)}{\partial W} = \sum_{t=1}^T \frac{\partial \mathbf{L}(\theta)}{\partial W}$$

For simplicity of this equation, we will apply backpropagation on only one row

$$\frac{\partial L(\theta)}{\partial W} = \frac{\partial L(\theta)}{\partial h_3} \frac{\partial h_3}{\partial W}$$

We already know how to compute this one as it is the same as any simple deep neural network backpropagation.

$$\frac{\partial L(\theta)}{\partial h_3}$$

$$\frac{\partial h_3}{\partial W}$$

.However, we will see how to apply backpropagation to this term

As we know $h_3 = \sigma(W h_2 + b)$

$$\frac{\partial h_3}{\partial W}$$

And In such an ordered network, we can't compute $\frac{\partial h_3}{\partial W}$ by simply treating h_3 as a constant because as

$$\frac{\partial h_3}{\partial W}$$

it also depends on W . the total derivative has two parts:

1. **Explicit:**

$$\frac{\partial h_3}{\partial W}$$

treating all other inputs as constant

2. **Implicit:** Summing over all indirect paths from h_3 to W

Let us see how to do this

$$\begin{aligned} \frac{\partial h_3}{\partial W} &= \frac{\partial h_3^+}{\partial W} + \frac{\partial h_3}{\partial h_2} \frac{\partial h_2}{\partial W} \\ &= \frac{\partial h_3^+}{\partial W} + \frac{\partial h_3}{\partial h_2} \left[\frac{\partial h_2^+}{\partial W} + \frac{\partial h_2}{\partial h_1} \frac{\partial h_1}{\partial W} \right] \\ &= \frac{\partial h_3^+}{\partial W} + \frac{\partial h_3}{\partial h_2} \frac{\partial h_2^+}{\partial W} + \frac{\partial h_3}{\partial h_2} \frac{\partial h_2}{\partial h_1} \left[\frac{\partial h_1^+}{\partial W} \right] \end{aligned}$$

For simplicity, we will short-circuit some of the paths

$$\frac{\partial h_3}{\partial W} = \frac{\partial h_3^+}{\partial W} + \frac{\partial h_3}{\partial h_2} \frac{\partial h_2^+}{\partial W} + \frac{\partial h_3}{\partial h_1} \frac{\partial h_1^+}{\partial W}$$

Finally, we have

$$\frac{\partial L(\theta)}{\partial W} = \frac{\partial L(\theta)}{\partial h_3} \cdot \frac{\partial h_3}{\partial W}$$

Where

$$\frac{\partial h_3}{\partial W} = \sum_{k=1}^3 \frac{\partial h_3}{\partial h_k} \cdot \frac{\partial h_k}{\partial W}$$

Hence,

$$\frac{\partial L(\theta)}{\partial W} = \frac{\partial L(\theta)}{\partial h_3} \sum_{k=1}^3 \frac{\partial h_3}{\partial h_k} \cdot \frac{\partial h_k}{\partial W}$$

This algorithm is called backpropagation through time (BPTT) as we backpropagate over all previous time steps

Issues of Standard RNNs

1. **Vanishing Gradient:** Text generation, machine translation, and stock market prediction are just a few examples of the time-dependent and sequential data problems that can be modelled with recurrent neural networks. You will discover, though, that the gradient problem makes training RNN difficult.
2. **Exploding Gradient:** An Exploding Gradient occurs when a neural network is being trained and the slope tends to grow exponentially rather than decay. Large error gradients that build up during training lead to very large updates to the neural network model weights, which is the source of this issue.

Training through RNN

1. A single-time step of the input is provided to the network.
2. Then calculate its current state using a set of current input and the previous state.
3. The current h_t becomes h_{t-1} for the next time step.
4. One can go as many time steps according to the problem and join the information from all the previous states.
5. Once all the time steps are completed the final current state is used to calculate the output.
6. The output is then compared to the actual output i.e the target output and the error is generated.
7. The error is then back-propagated to the network to update the weights and hence the network (RNN) is trained using Backpropagation through time.

Advantages and Disadvantages of Recurrent Neural Network

Advantages

1. An RNN remembers each and every piece of information through time. It is useful in time series prediction only because of the feature to remember previous inputs as well. This is called Long Short Term Memory.
2. Recurrent neural networks are even used with convolutional layers to extend the effective pixel neighborhood.

Disadvantages

1. Gradient vanishing and exploding problems.
2. Training an RNN is a very difficult task.
3. It cannot process very long sequences if using tanh or relu as an activation function.

Applications of Recurrent Neural Network

1. Language Modelling and Generating Text
2. Speech Recognition
3. Machine Translation
4. Image Recognition, Face detection
5. Time series Forecasting

Variation Of Recurrent Neural Network (RNN)

To overcome the problems like vanishing gradient and exploding gradient descent several new advanced versions of RNNs are formed some of these are as;

1. Bidirectional Neural Network (BiNN)
2. Long Short-Term Memory (LSTM)

Bidirectional Neural Network (BiNN)

A BiNN is a variation of a Recurrent Neural Network in which the input information flows in both direction and then the output of both direction are combined to produce the input. BiNN is useful in situations when the context of the input is more important such as Nlp tasks and Time-series analysis problems.

Long Short-Term Memory (LSTM)

Long Short-Term Memory works on the read-write-and-forget principle where given the input information network reads and writes the most useful information from the data and it forgets about the information which is not important in predicting the output. For doing this three new gates are introduced in the RNN. In this way, only the selected information is passed through the network.

Difference between RNN and Simple Neural Network

RNN is considered to be the better version of deep neural when the data is sequential. There are significant differences between the RNN and deep neural networks they are listed as:

Recurrent Neural Network	Deep Neural Network
Weights are same across all the layers number of a Recurrent Neural Network	Weights are different for each layer of the network
Recurrent Neural Networks are used when the data is sequential and the number of inputs is	A Simple Deep Neural network does not have any special method for sequential data also

not predefined.	here the the number of inputs is fixed
The Numbers of parameter in the RNN are higher than in simple DNN	The Numbers of Parameter are lower than RNN
Exploding and vanishing gradients is the the major drawback of RNN	These problems also occur in DNN but these are not the major problem with DNN