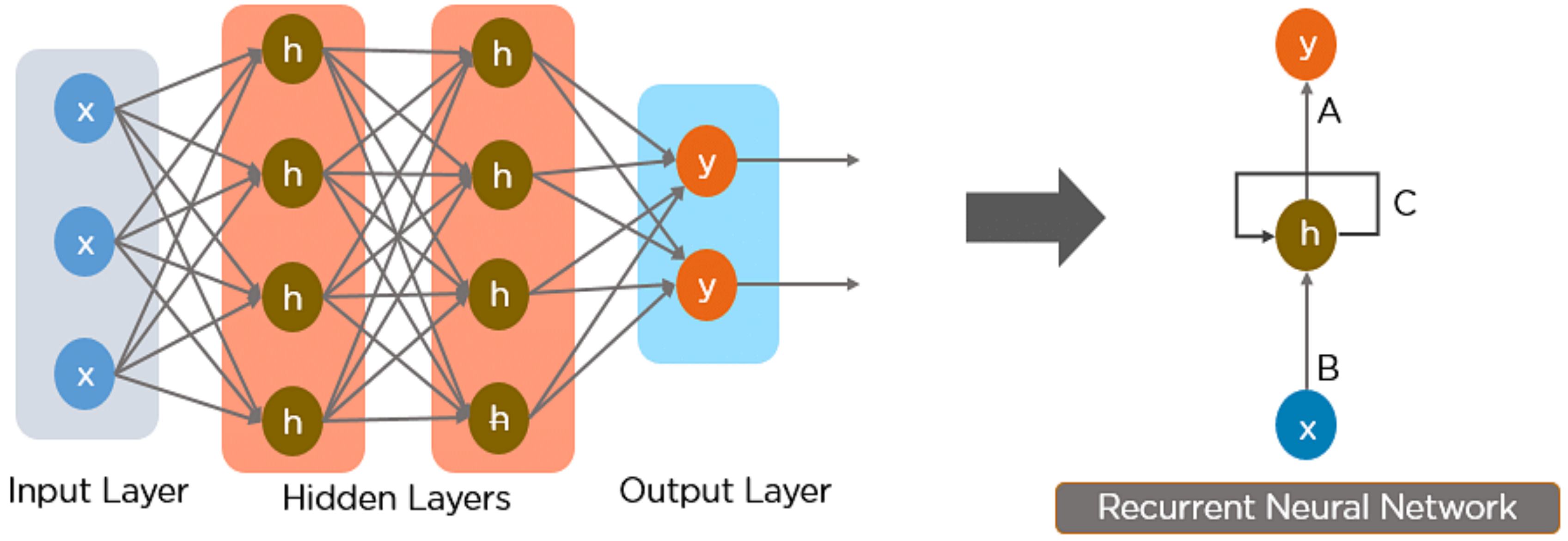
Recurrent Neural Network (RNN)

Motivation

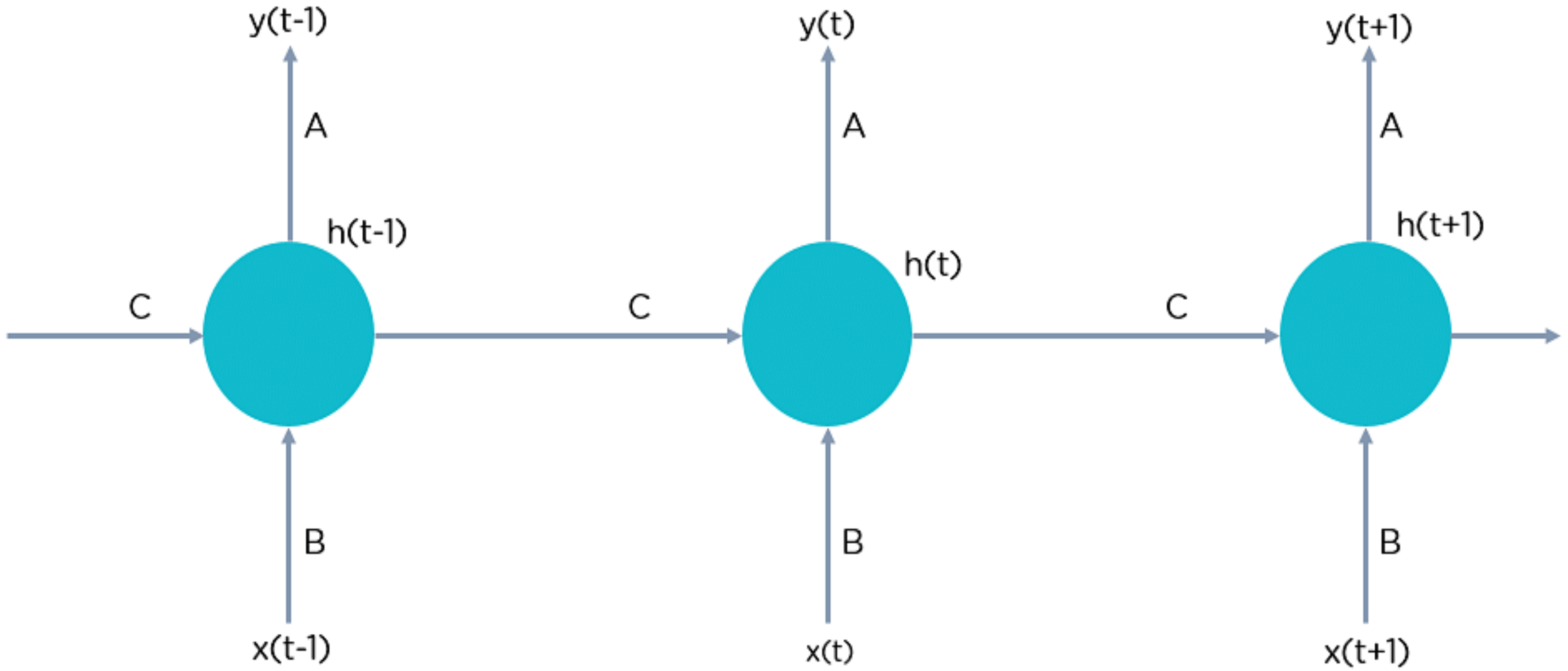
- Feed-forward neural network failed to
  - Cannot handle sequential data
  - Considers only the current input
  - Cannot memorize previous inputs
- The solution to these issues is the RNN. An RNN can handle sequential data, accepting the current input data, and previously received inputs. RNNs can memorize previous inputs due to their internal memory.

- RNN works on the principle of saving the output of a particular layer and feeding this back to the input in order to predict the output of the layer.

- The nodes in different layers of the neural network are compressed to form a single layer of recurrent neural networks. A, B, and C are the parameters of the network.



- “x” is the input layer
- “h” is the hidden layer
- “y” is the output layer.
- A, B, and C are the network parameters used to improve the output of the model.
- At any given time t, the current input is a combination of input at x(t) and x(t-1).
- The output at any given time is fetched back to the network to improve on the output.
-



$$h(t) = f_c\,(h(t-1), x(t))$$

$h(t)$ = new state

f_c = function with parameter c

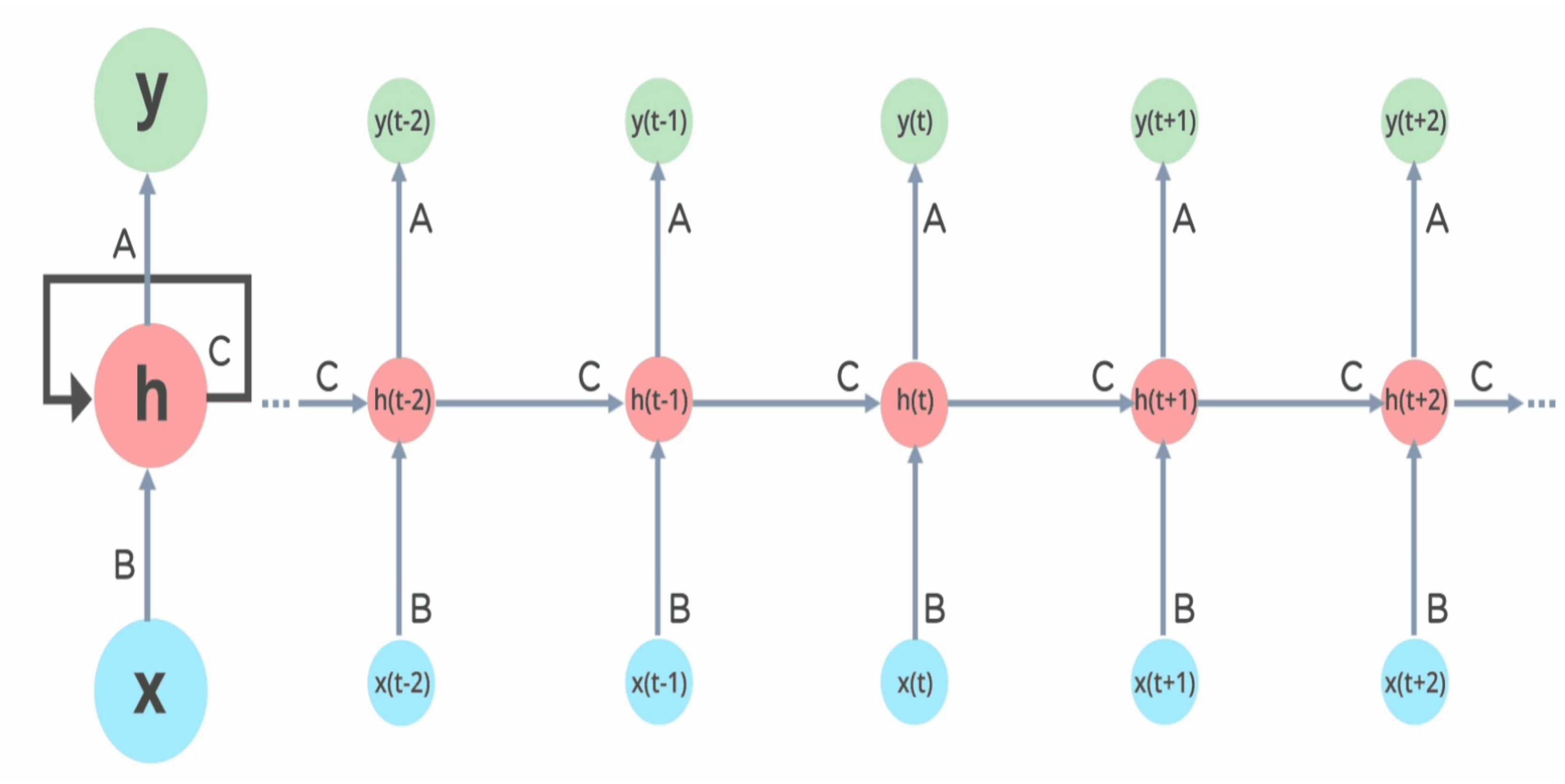
$h(t-1)$ = old state

$x(t)$ = input vector at time step t

How Does Recurrent Neural Networks Work?

- In Recurrent Neural networks, the information cycles through a loop to the middle hidden layer.
- The input layer ‘x’ takes in the input to the neural network and processes it and passes it onto the middle layer.
- The middle layer ‘h’ can consist of multiple hidden layers, each with its own activation functions and weights and biases.

- The Recurrent Neural Network will standardize the different activation functions and weights and biases so that each hidden layer has the same parameters.
- Then, instead of creating multiple hidden layers, it will create one and loop over it as many times as required.


y
A
h
C
B
x
y(t-2)
y(t-1)
y(t)
y(t+1)
y(t+2)
h(t-2)
h(t-1)
h(t)
h(t+1)
h(t+2)
x(t-2)
x(t-1)
x(t)
x(t+1)
x(t+2)

Feed-Forward Neural Networks vs Recurrent Neural Networks

- A feed-forward neural network allows information to flow only in the forward direction, from the input nodes, through the hidden layers, and to the output nodes.
- There are no cycles or loops in the network.
- In a feed-forward neural network, the decisions are based on the current input.
- It doesn’t memorize the past data, and there’s no future scope.
- Feed-forward neural networks are used in general regression and classification problems.

Applications of RNN

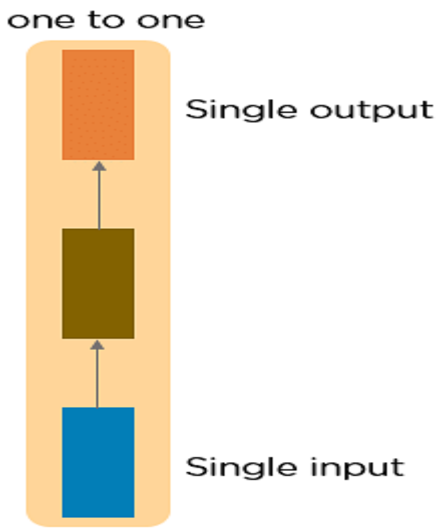
- Image Captioning
- Time Series Prediction
- Natural Language Processing
- Machine Translation
-

Types

- One to One
- One to Many
- Many to One
- Many to Many

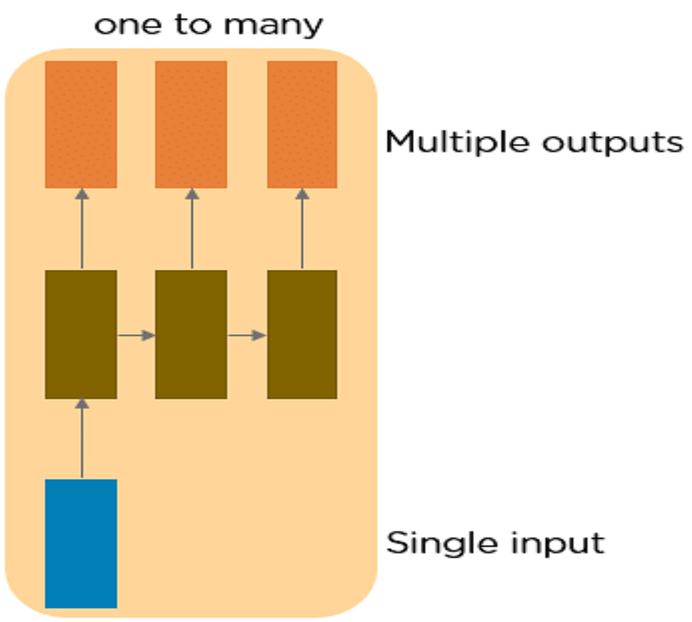
One to One RNN

- This type of neural network is known as the Vanilla Neural Network.
- It's used for general machine learning problems, which has a single input and a single output.



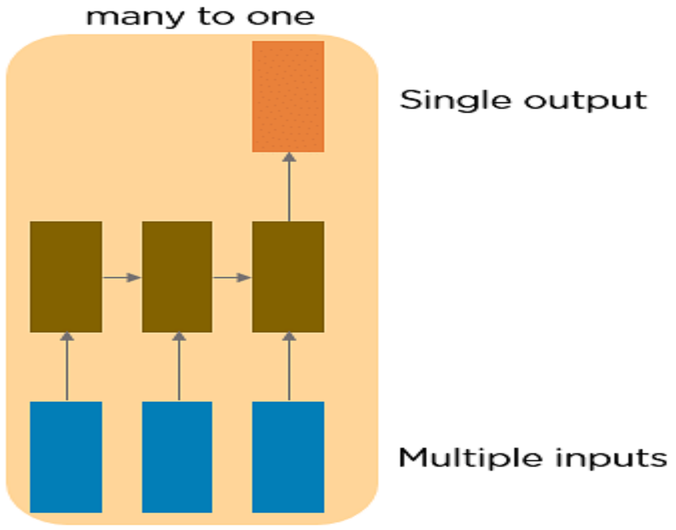
One to Many RNN

- This type of neural network has a single input and multiple outputs.
- An example of this is the image caption.



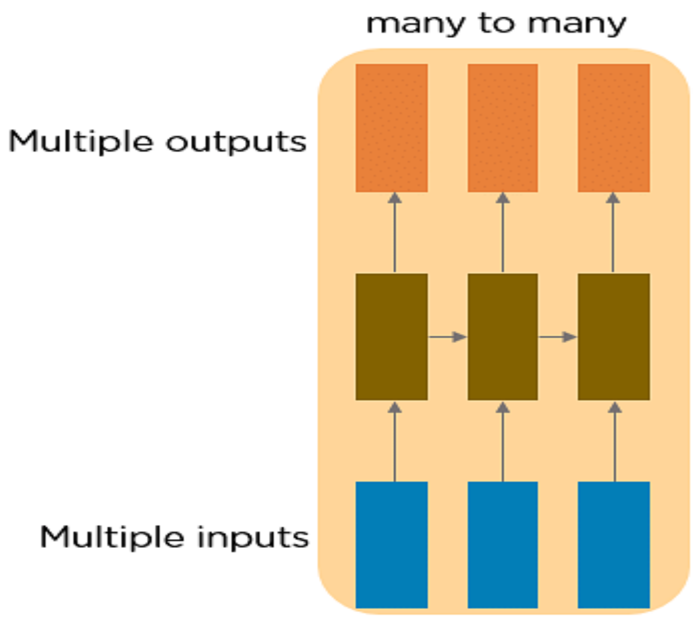
Many to One RNN

- This RNN takes a sequence of inputs and generates a single output.
- Sentiment analysis is a good example of this kind of network where a given sentence can be classified as expressing positive or negative sentiments.



Many to Many RNN

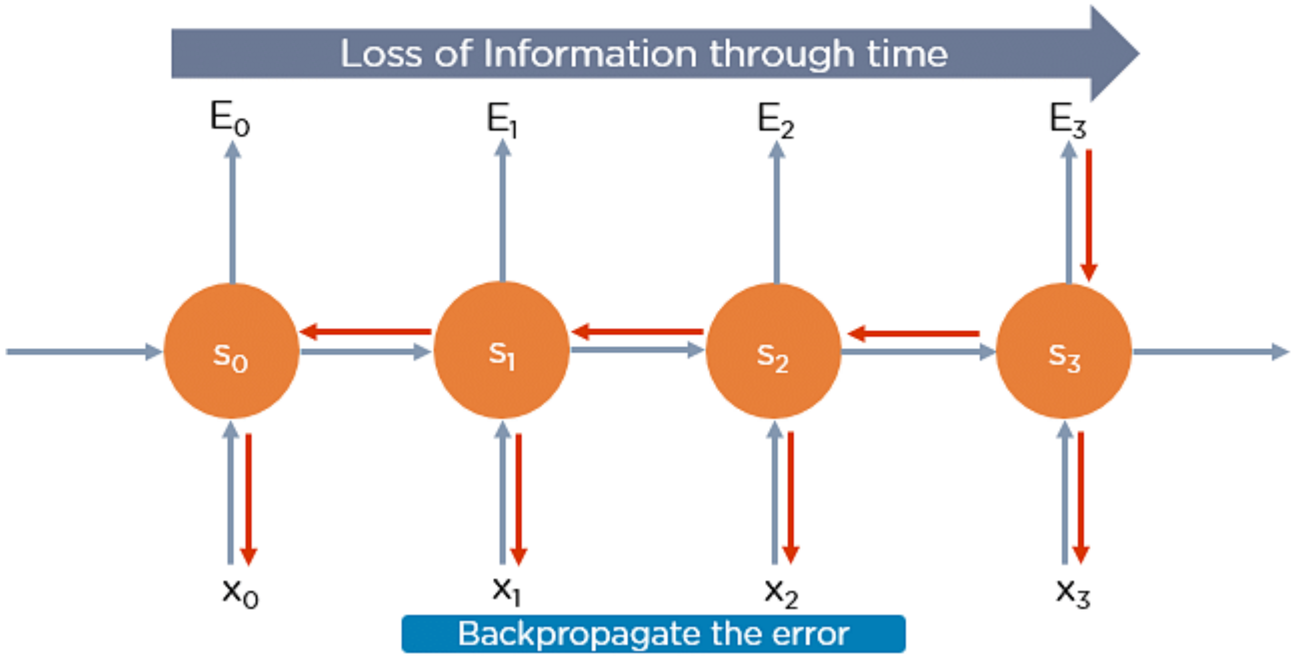
- This RNN takes a sequence of inputs and generates a sequence of outputs.
- Machine translation is one of the examples.



Issues

- Two Issues of Standard RNNs

1. Vanishing Gradient Problem
   - Recurrent Neural Networks enable you to model time-dependent and sequential data problems, such as stock market prediction, machine translation, and text generation. You will find, however, RNN is hard to train because of the gradient problem.
   - RNNs suffer from the problem of vanishing gradients. The gradients carry information used in the RNN, and when the gradient becomes too small, the parameter updates become insignificant. This makes the learning of long data sequences difficult.


Loss of Information through time
E_0
E_1
E_2
E_3
s_0
s_1
s_2
s_3
x_0
x_1
x_2
x_3
Backpropagate the error

Issues

- Exploding Gradient Problem

- While training a neural network, if the slope tends to grow exponentially instead of decaying, this is called an Exploding Gradient.
- This problem arises when large error gradients accumulate, resulting in very large updates to the neural network model weights during the training process.
- Long training time, poor performance, and bad accuracy are the major issues in gradient problems.