Hedgehog Rock

University of Science and Technology Bannu

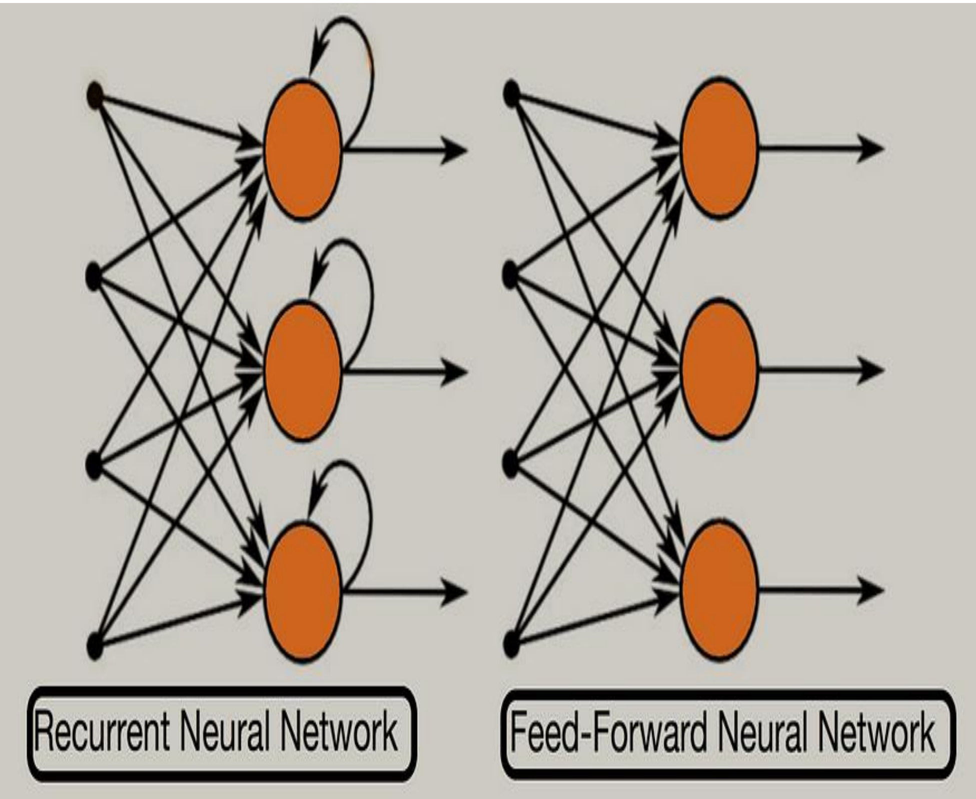
Deep Learning

Lesson 7

September,23 2024

Learning Objectives

- Need for a Recurrent Neural Network
- **Why RNN**
- **RNN Architecture**



RNN

Recurrent neural networks (RNNs) are a type of artificial neural network **specifically designed to handle sequential data.**

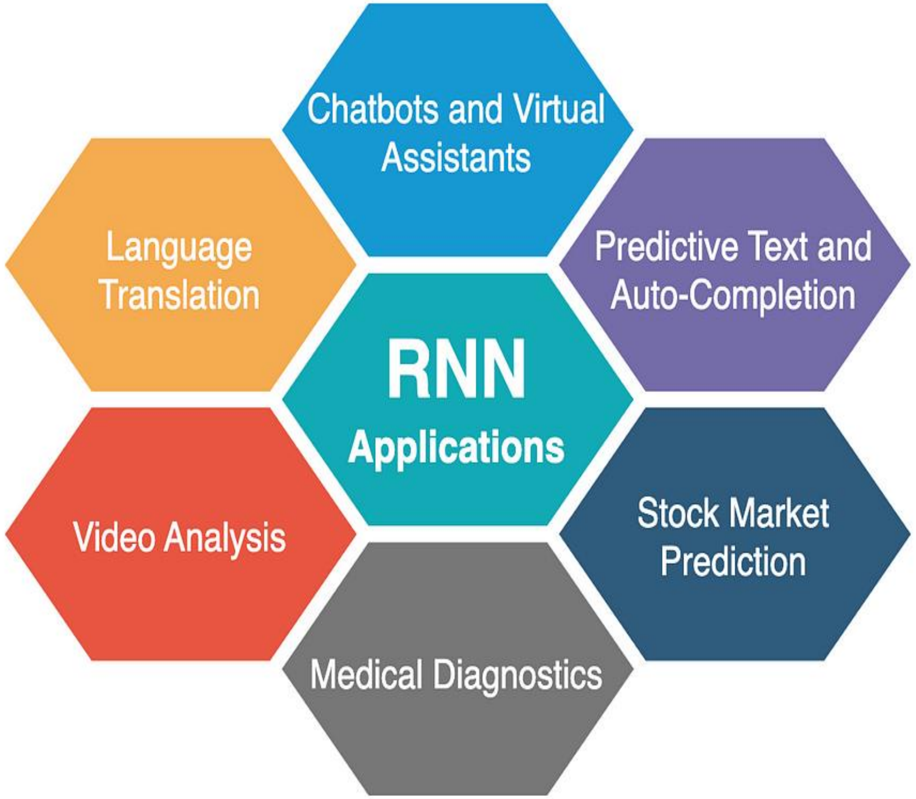
It is designed specifically for tasks where the sequence of input matters

RNNs can be used for mapping inputs to outputs of varying types, lengths and are fairly generalized in their application.

Need of Recurrent Neural Network

A person riding a
motorcycle on a dirt road.

Two dogs play in the grass.



Chatbots and Virtual Assistants

Language Translation

Predictive Text and Auto-Completion

RNN Applications

Video Analysis

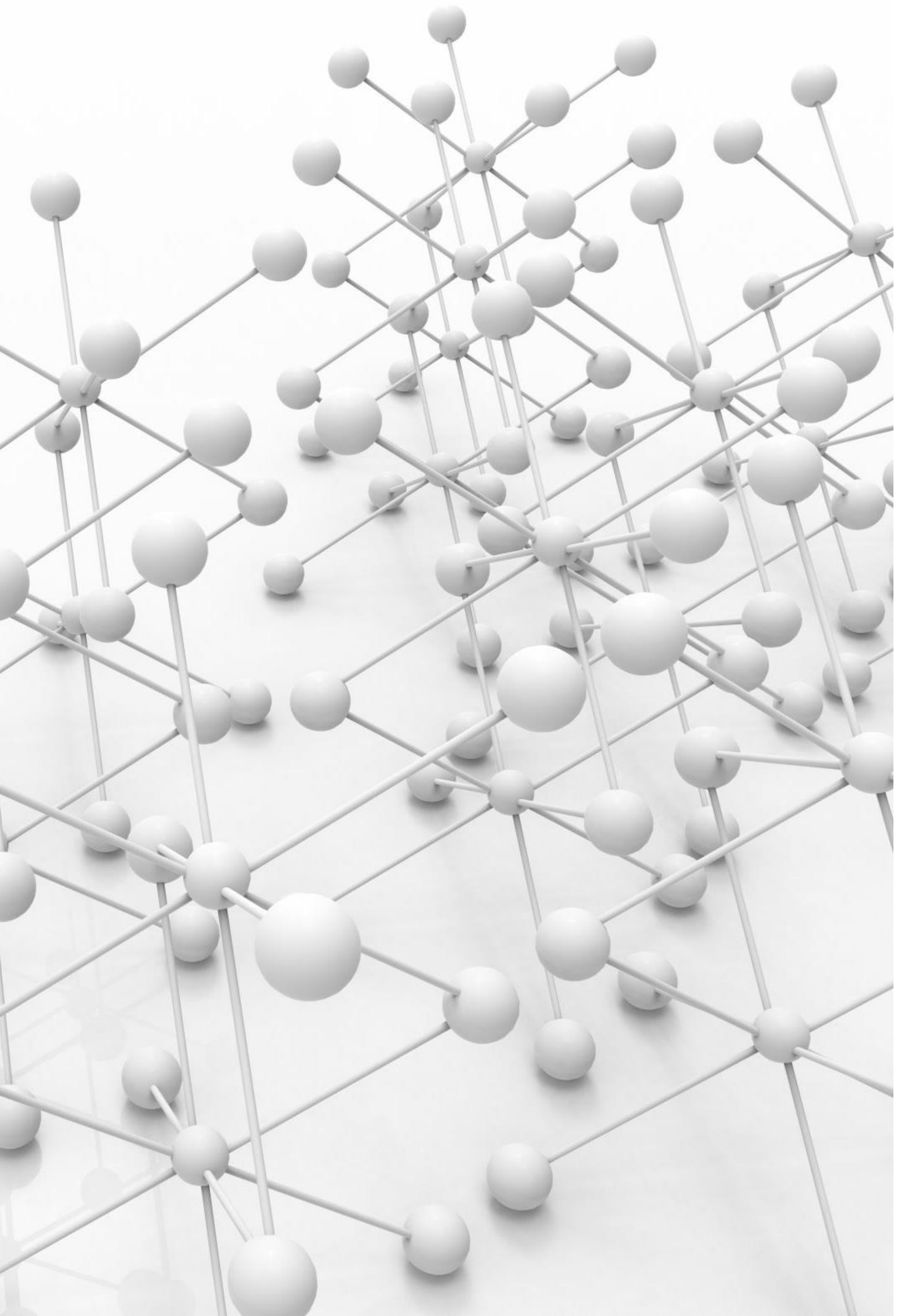
Stock Market Prediction

Medical Diagnostics

The beauty of recurrent neural networks lies in their diversity of application.

When we are dealing with RNNs they have a great ability to deal with various input and output types.

- **Sentiment Classification**
- **Image Captioning**
- **Language Translation**
- **Summarization**
- **Sentiment Analysis**
- **NER**
- **POS**



Why Recurrent Neural Networks?

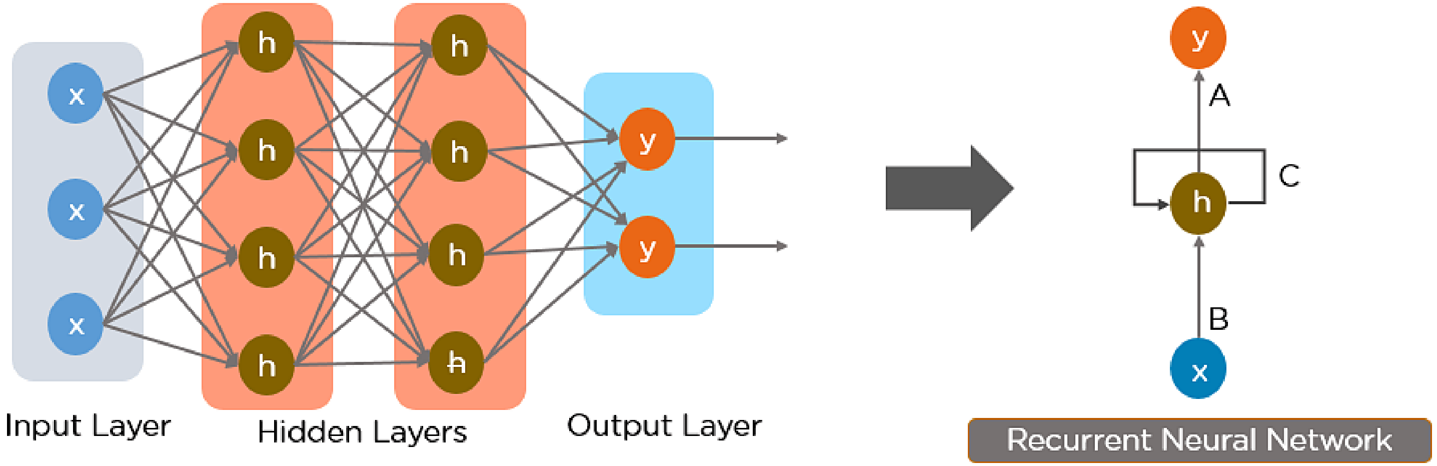
RNN were created because there were a few issues in the feed-forward neural network:

- **Cannot handle sequential data**
- **Considers only the current input**
- **Cannot memorize previous inputs**
- The weights and bias of these hidden layers are different.
- And hence each of these layers behave independently and cannot be combined together.
- To combine these hidden layers together, we shall have the same weights and bias for these hidden layers.

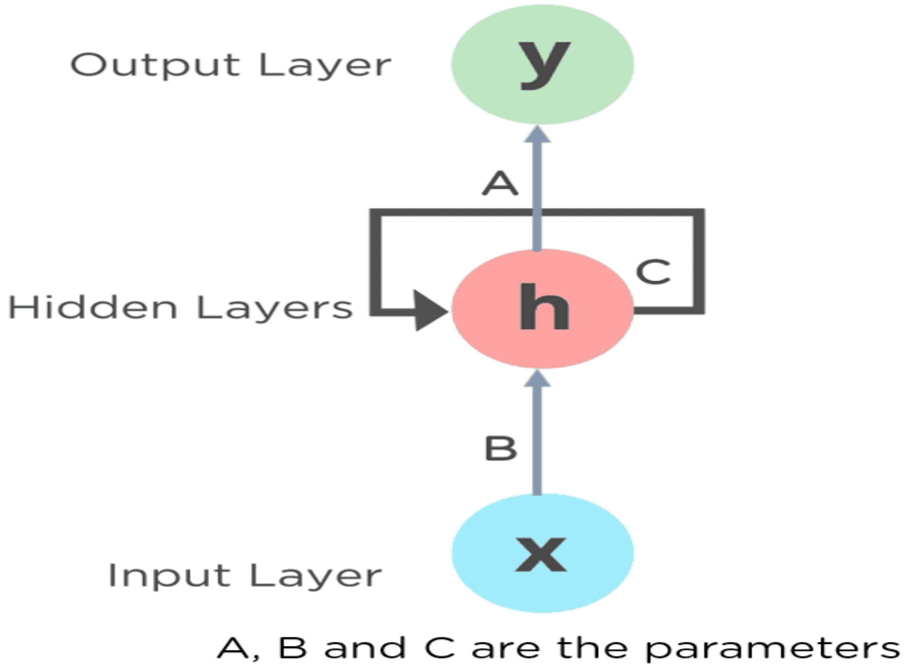
Why Recurrent Neural Networks?

- 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.

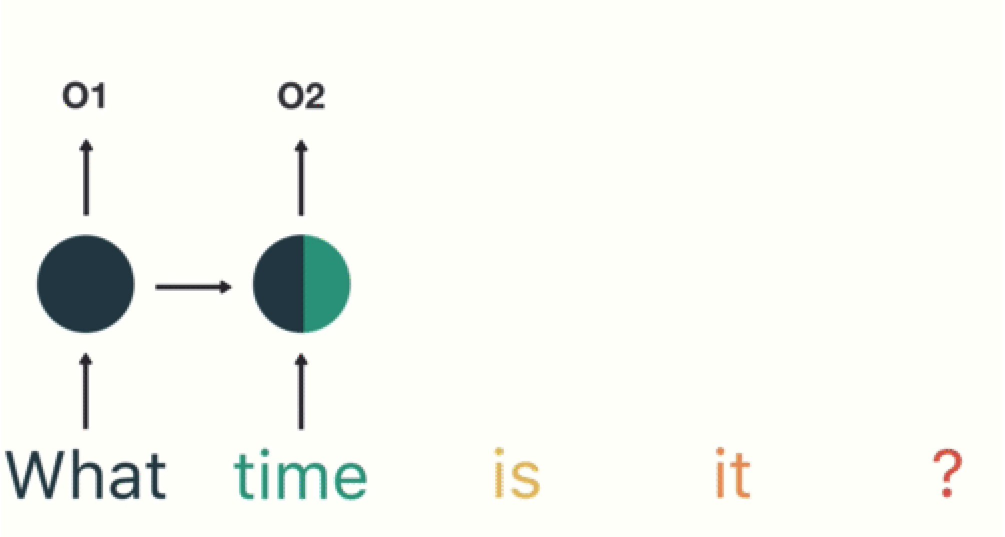
What Is a Recurrent Neural Network (RNN)?



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.

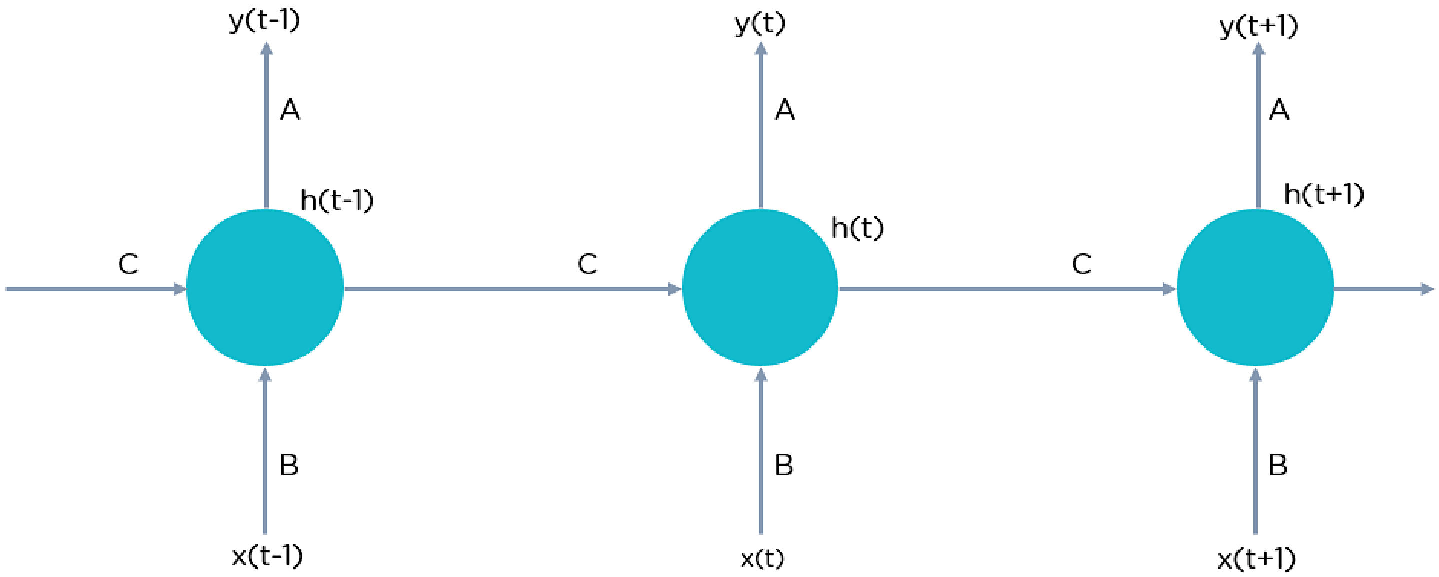


A, B and C are the parameters



RNN

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.



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

$h(t)$ = new state

f_c = function with parameter c

$h(t-1)$ = old state

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

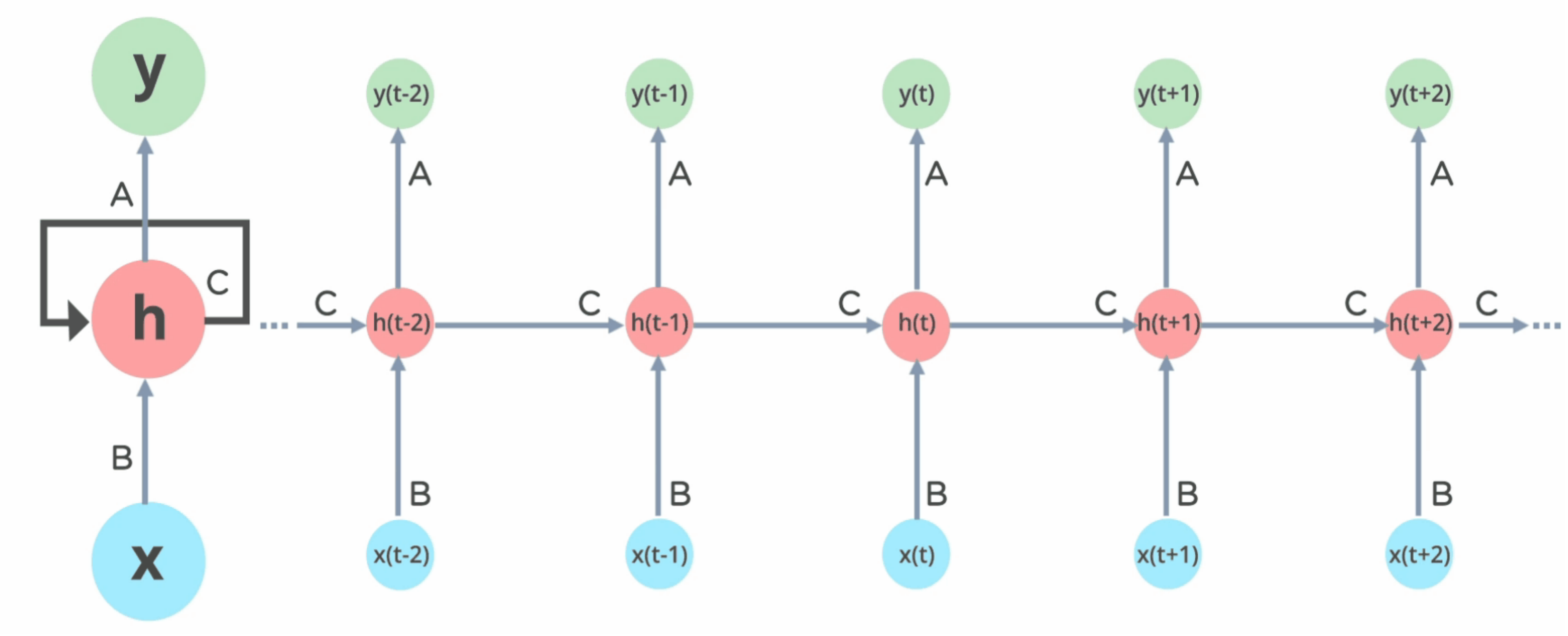
RNN

Here, “x” is the input layer, “h” is the hidden layer, and “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.

How Does Recurrent Neural Networks Work?

In Recurrent Neural networks, the information cycles through a loop to the middle hidden layer.



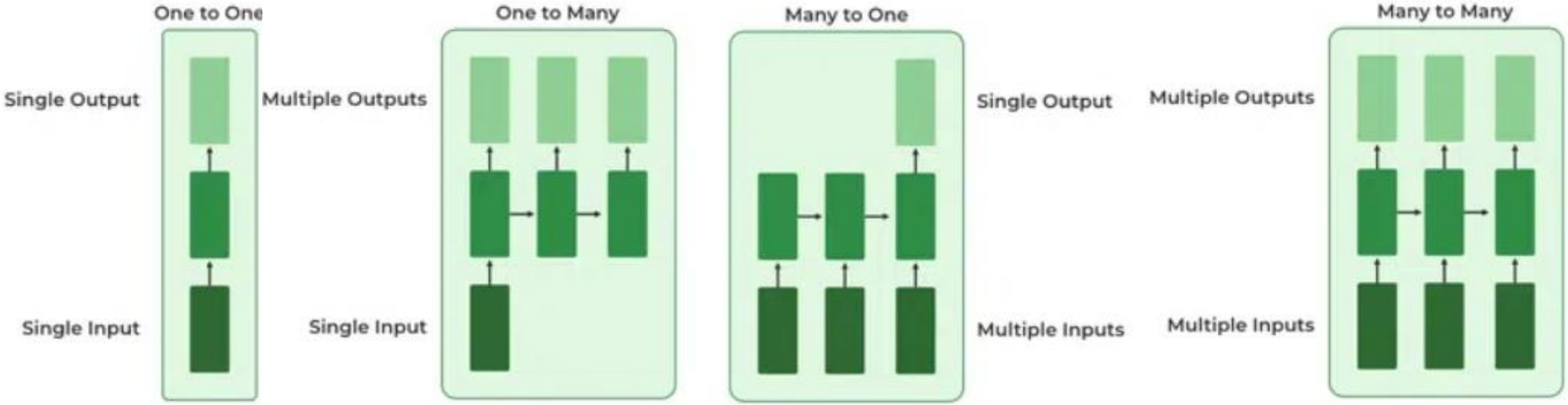
Various Architectures of RNN

1. One to one

2. One to many

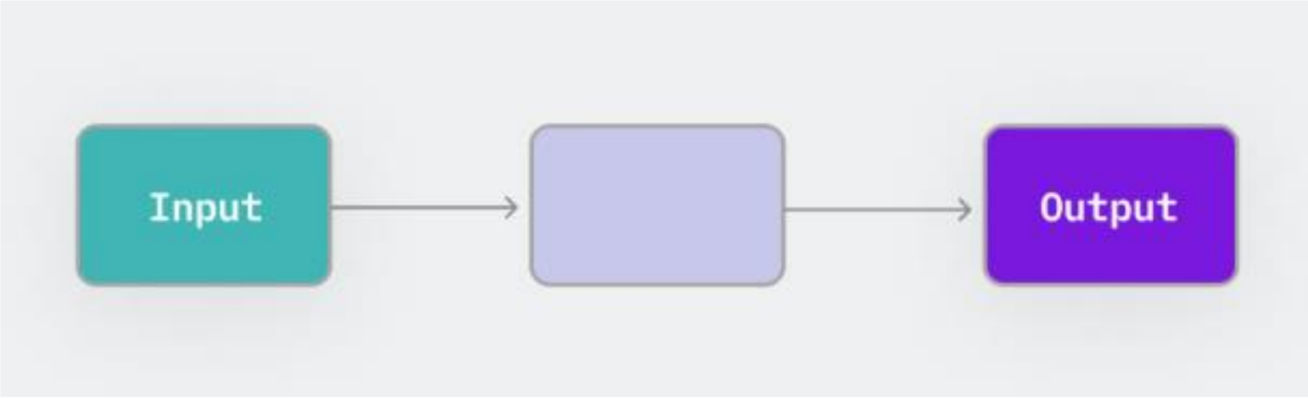
3.Many to one

4. Many to many



One to one

The most straightforward type of RNN is One-to-One, which allows a single input and a single output. It has fixed input and output sizes and acts as a standard neural network. The One-to-One application can be found in *Image Classification, or Next word prediction*.



One to Many

In recurrent neural networks (RNNs), a “one-to-many” architecture represents a scenario where the network receives a **single input** but generates a **sequence of outputs**

- **Single Input:** The RNN takes in a single piece of information as input. This could be an image, a musical note, a short sentence, or any data point that serves as a starting point for the network.
- **Multiple Outputs:** The RNN processes the input and generates a sequence of outputs over time. This sequence can vary in length depending on the specific task.

Common applications of one to many RNNs:

- **Image Captioning:** Based on a single image input, the RNN generates a sentence or paragraph describing the image content (multiple outputs).
- **Music Generation:** The RNN receives a single starting note or short melody as input and produces a sequence of musical notes forming a complete piece (multiple outputs).

Many to One :

In recurrent neural networks (RNNs), a “many-to-one” architecture refers to a specific type of RNN where the network processes a **sequence of inputs** but produces a **single output**.

- **Many Inputs:** The RNN takes in a sequence of data points over time. This sequence could be words in a sentence, sensor readings over a period, or financial data points for multiple days.
- **Single Output:** After processing the entire sequence, the RNN generates a single output value. This output could be a classification (positive/negative sentiment), a prediction (next value in the time series), or a summary of the information in the sequence.

Common applications of many-to-one RNNs:

- **Sentiment Analysis:** Given a sentence or review text (sequence of words), classify its overall sentiment (positive, negative, or neutral) as the single output.
- **Spam Detection:** Analyze an email’s content (sequence of words) to determine if it’s spam (single output).

Many to Many :

In recurrent neural networks (RNNs), a “many-to-many” architecture describes a scenario where the network processes a **sequence of inputs** and generates a corresponding **sequence of outputs**. This means both the input and output have multiple elements processed over time steps.

Here’s a breakdown of the concept:

- **Multiple Inputs:** The RNN takes in a sequence of data points, similar to many-to-one RNNs. This sequence could be words in a sentence, sensor readings, or financial data points.
- **Multiple Outputs:** The RNN generates a new sequence of data points, with a length that may or may not be the same as the input sequence.

Many to Many :

There are two main categories of many-to-many RNNs:

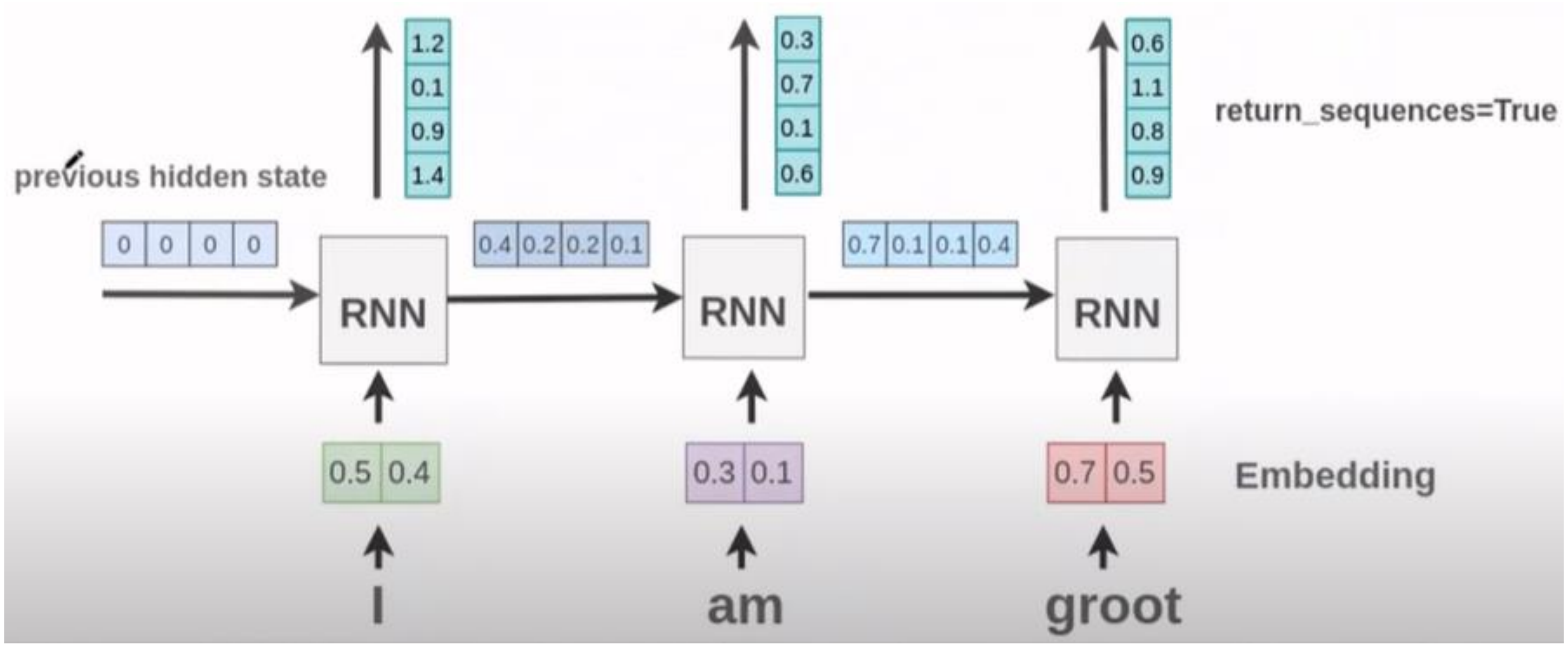
1. **Fixed Length:** In this case, the number of elements in the output sequence is the same as the number of elements in the input sequence. A common application is **named entity recognition (NER)**, where the network identifies and classifies each word in a sentence (input sequence) as a specific entity type (i.e. parts of speech) (output sequence).

2. **Variable Length:** Here, the output sequence can have a different length than the input sequence. This is particularly useful for tasks like **machine translation**, where the source sentence (input sequence) in one language might be translated into a longer or shorter sentence (output sequence) in another language.

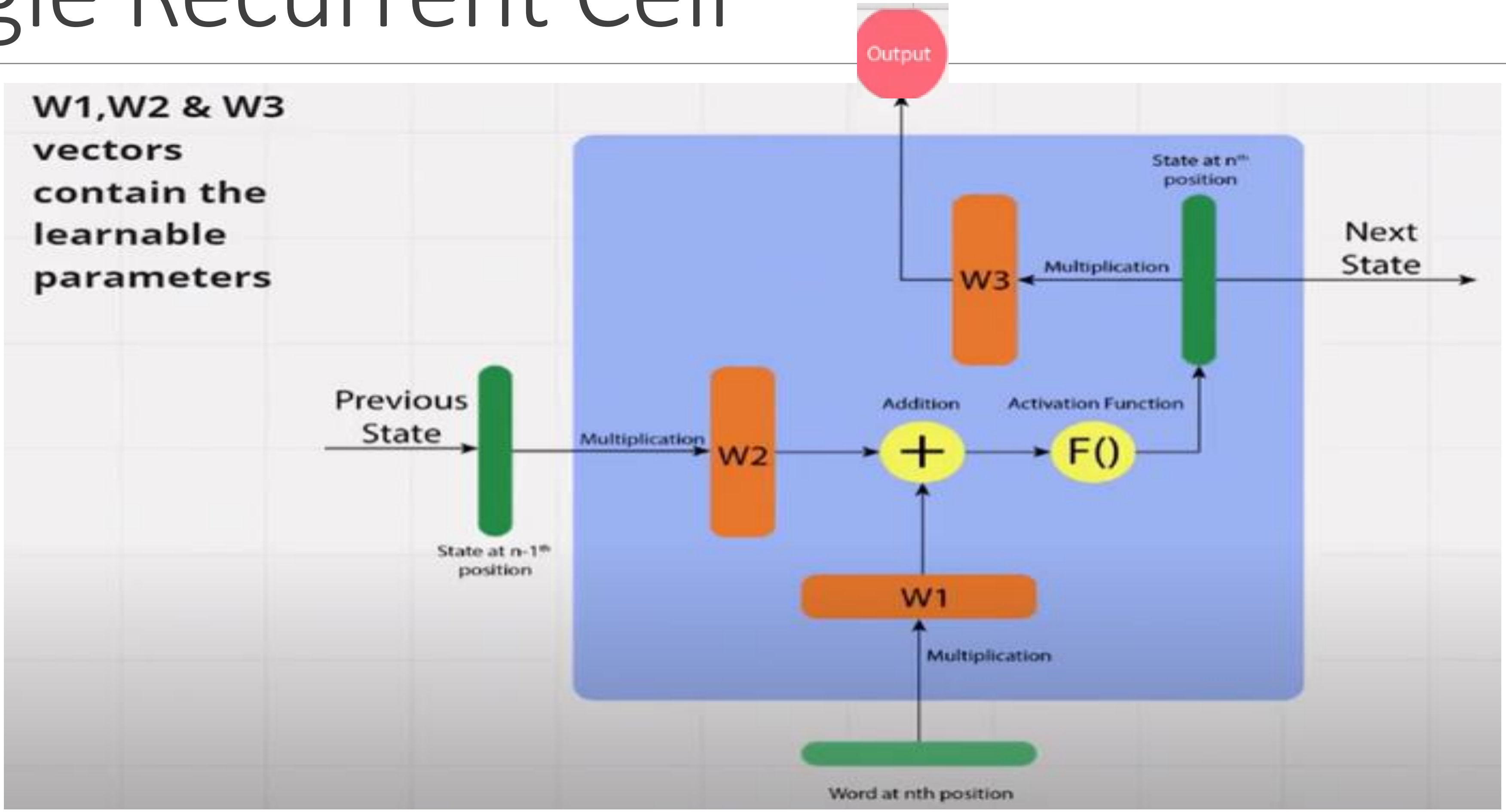
Many to Many :

Common Applications of Many-to-Many RNNs:

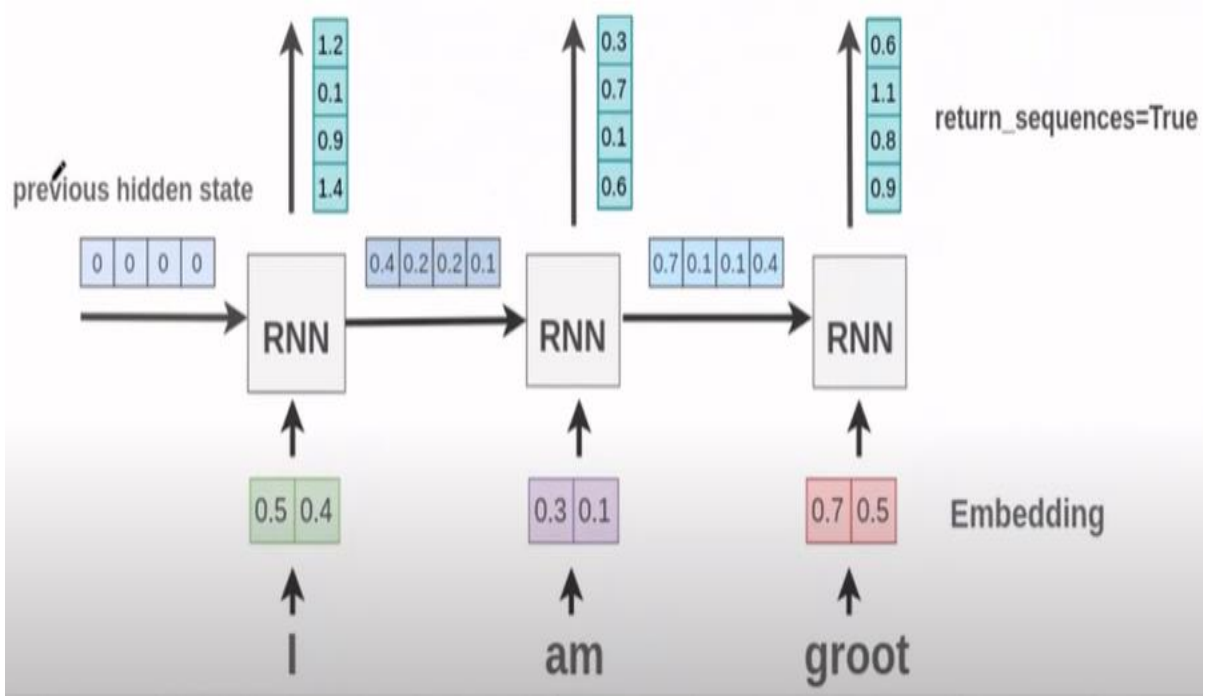
- **Machine Translation:** Translate text from one language to another (e.g., English to French).
- **Video Captioning:** Generate captions describing the content of a video (sequence of video frames as input, sequence of words as output).
- **Text Summarization:** Summarize a long document into a shorter version with key points (sequence of sentences as input, shorter sequence of sentences as output).

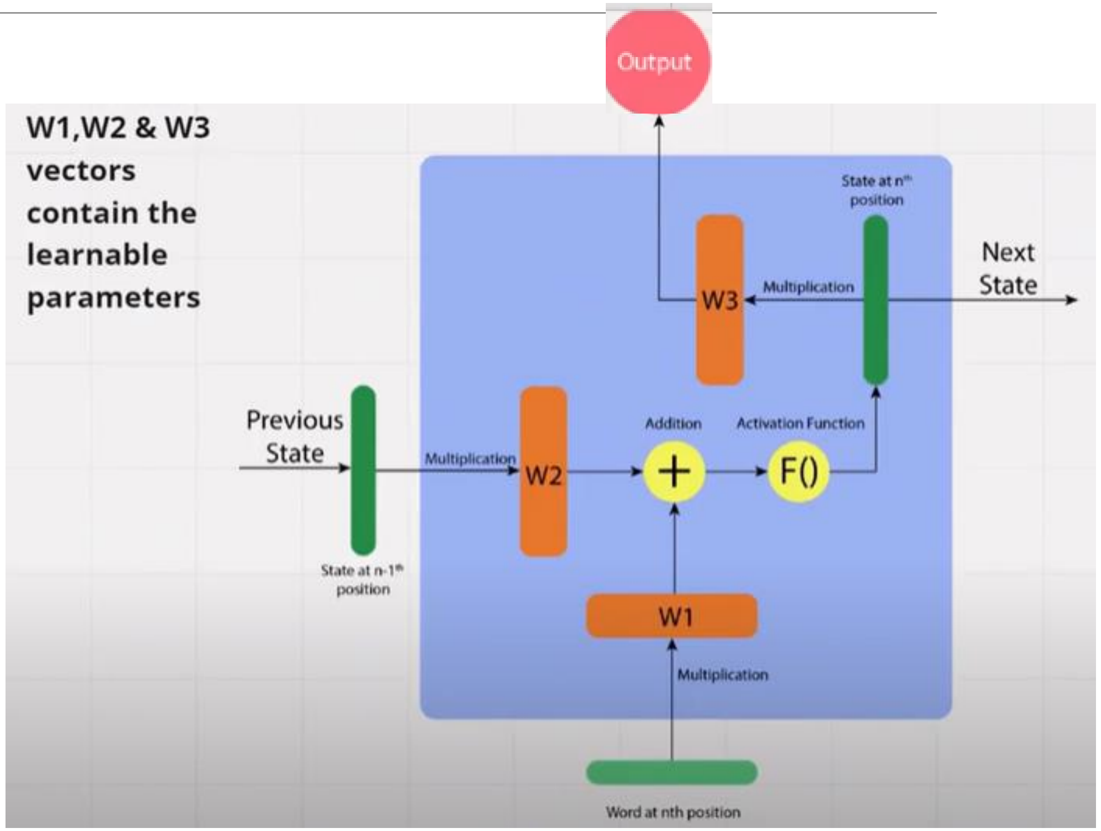

1.2
0.1
0.9
1.4
0.3
0.7
0.1
0.6
0.6
1.1
0.8
0.9
return_sequences=True
previous hidden state
0 0 0 0
0.4 0.2 0.2 0.1
0.7 0.1 0.1 0.4
RNN
RNN
RNN
0.5 0.4
0.3 0.1
0.7 0.5
Embedding
I
am
groot

Single Recurrent Cell



Single Recurrent Cell





W1,W2 & W3 vectors contain the learnable parameters