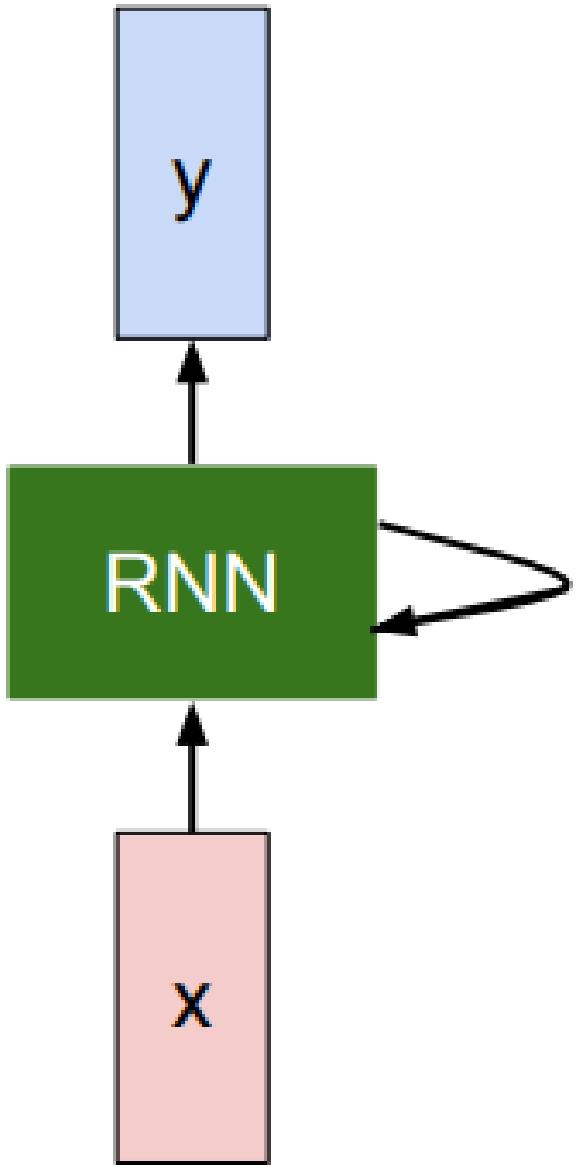
Course Title: Artificial Intelligence

- Course Code: CSC462
- Credit Hours: 4 (3,1)
- Lab Hours/Week: 3
- Pre-Requisites: CSC102-Discrete Structures
- Text and Reference Books:Textbook: 1. Artificial Intelligence: A Modern Approach, Russell, S., and Norvig, P., Pearson, 2020. Reference Book: 1. Artificial Intelligence Basics: A Non-Technical Introduction, Taulli, T., Apress, 2019.

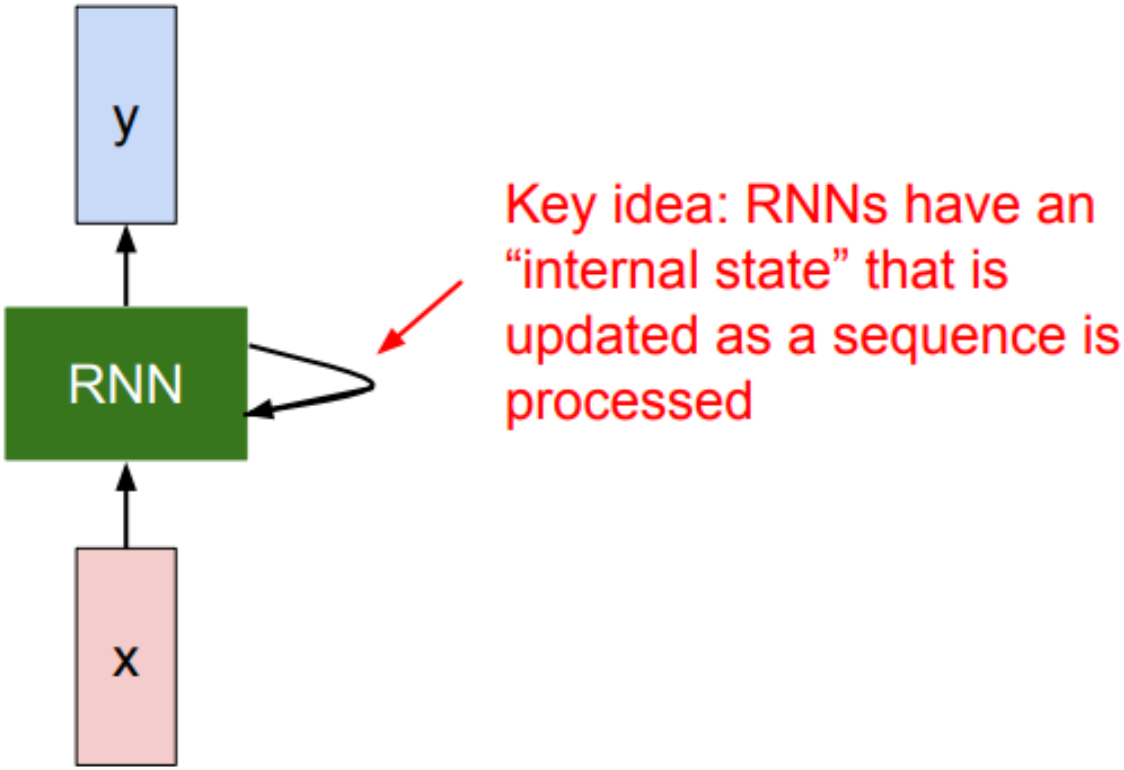
Recurrent Neural Networks (RNN)

- Recurrent Neural Networks (RNNs) are a type of **artificial neural network** designed to process sequences of data. They work especially well for jobs requiring sequences, such as **time series data, voice, natural language,** and other activities.
- 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.

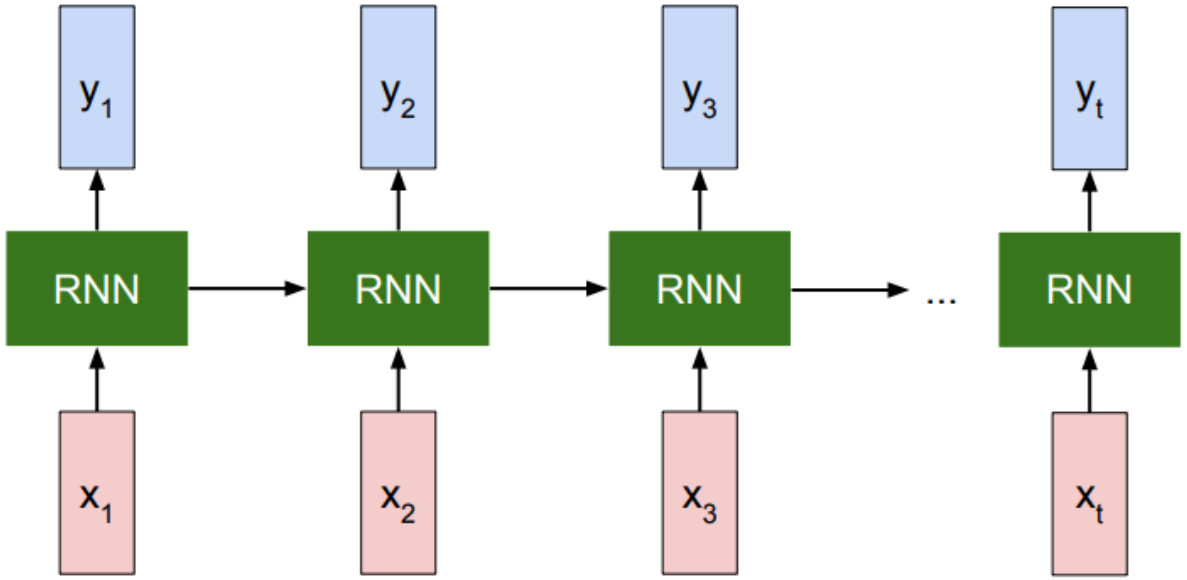
Recurrent Neural Networks (RNN)



Recurrent Neural Networks (RNN)



Recurrent Neural Networks (RNN)



RNN hidden state update

We can process a sequence of vectors x by applying a recurrence formula at every time step:

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

new state

some function with parameters W

old state

input vector at some time step

RNN output generation

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

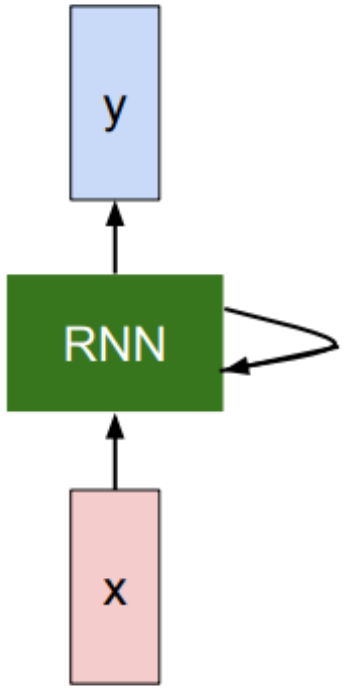
We can process a sequence of vectors **x** by applying a **recurrence formula** at every time step:

$$y_t = f_{W_{hy}}(h_t)$$

output

new state

another function with parameters W_o



Formula for applying Activation function(tanh)

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

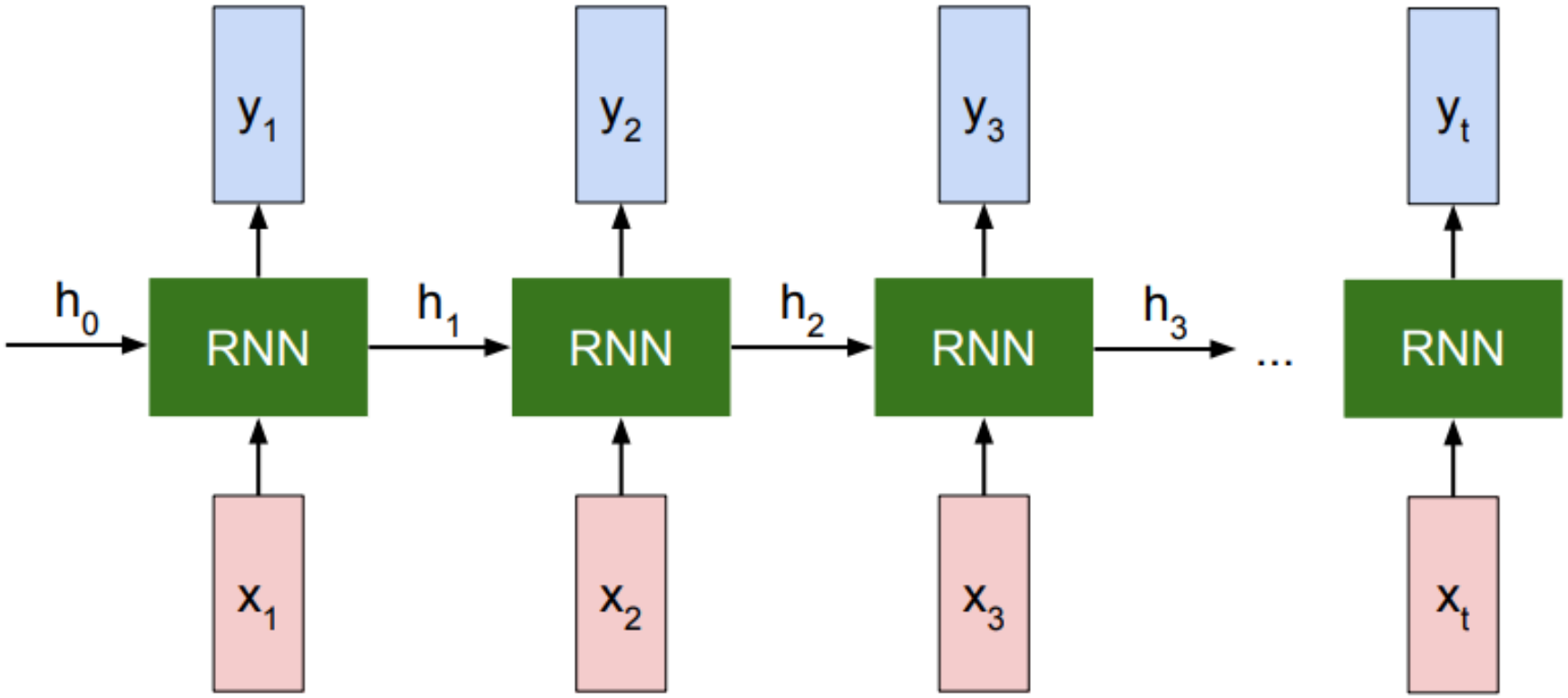
$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$

$$y_t = W_{hy}h_t$$

where,

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

Recurrent Neural Network

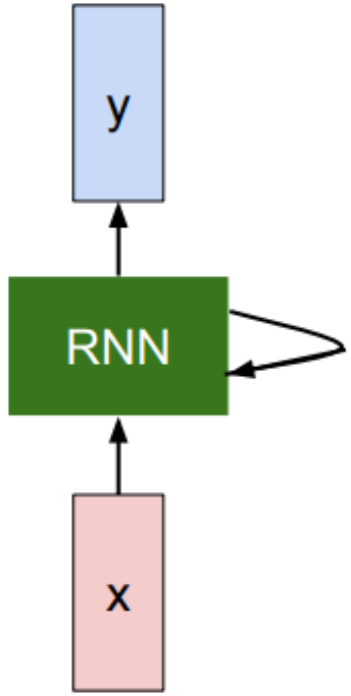


Recurrent Neural Network

We can process a sequence of vectors **x** by applying a **recurrence formula** at every time step:

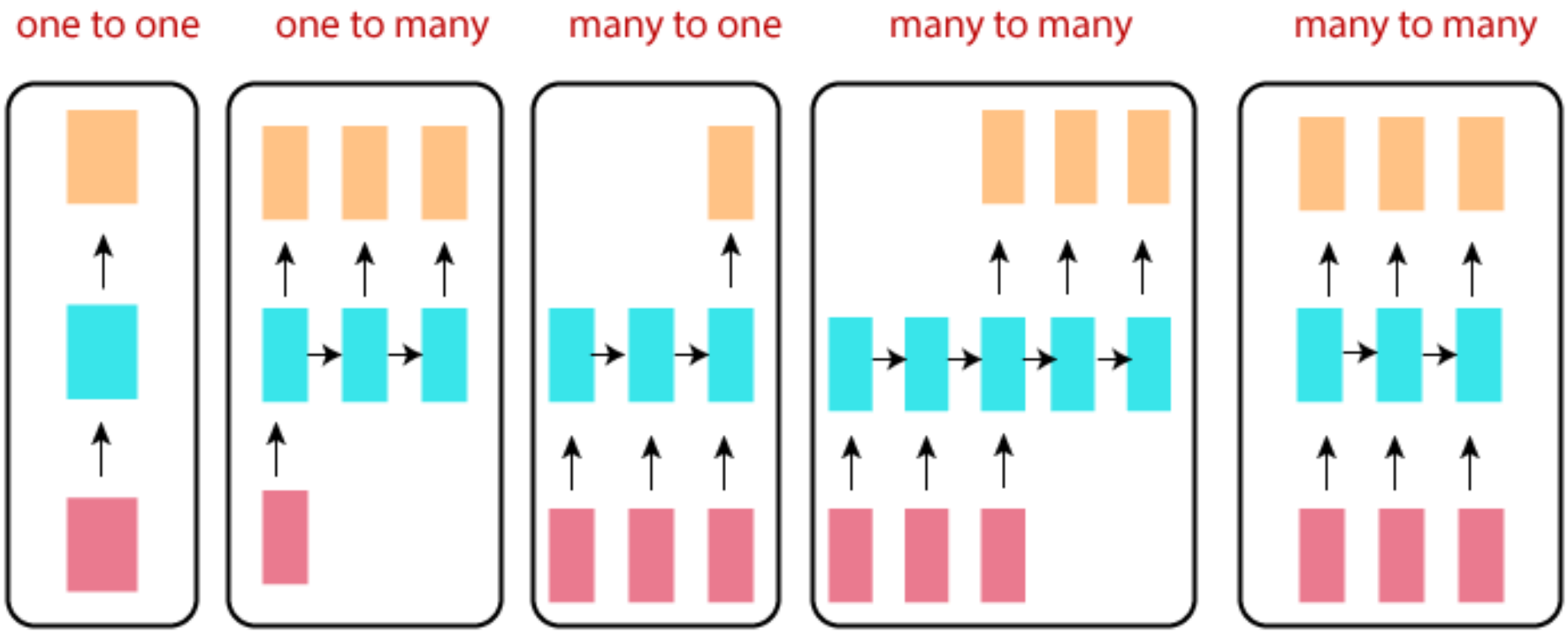
$$h_t = f_W(h_{t-1}, x_t)$$

Notice: the same function and the same set of parameters are used at every time step.



Types of RNN

Each rectangle in the above image represents **vectors**, and arrows represent **functions**. Input vectors are Red, output vectors are blue, and green holds RNN's state.

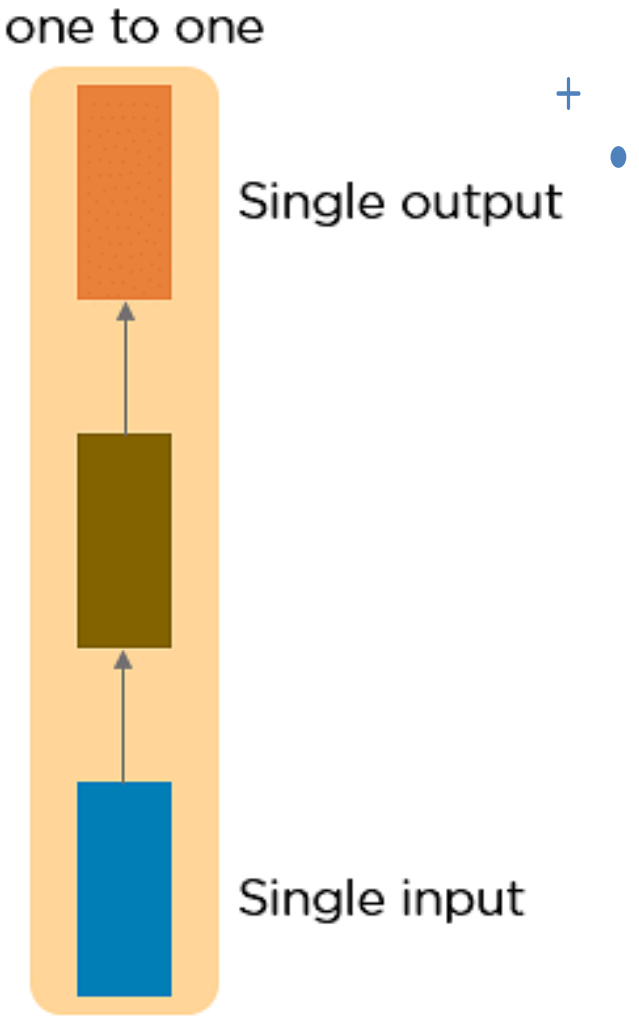


Types of RNN

- **One-to-one**

This is also called **Plain Neural networks**. It deals with a fixed size of the input to the fixed size of output, where they are independent of previous information/output.

Example: Image classification.

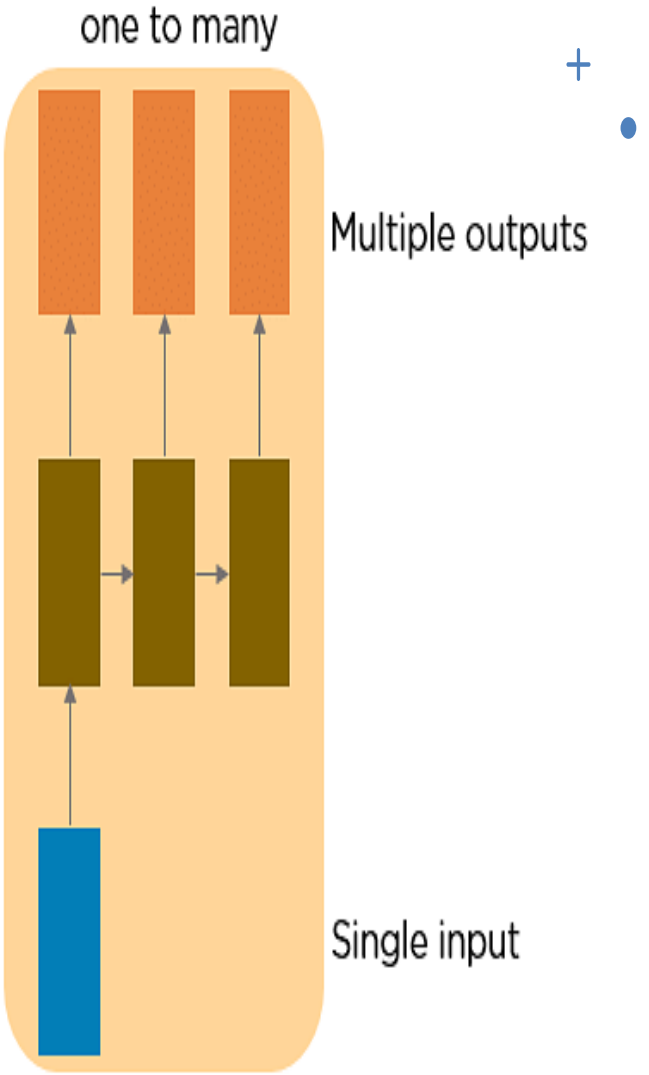


Types of RNN

- **One-to-Many:**

It deals with a fixed size of information as input that gives a sequence of data as output.

Example: Image Captioning takes the image as input and outputs a sentence of words.

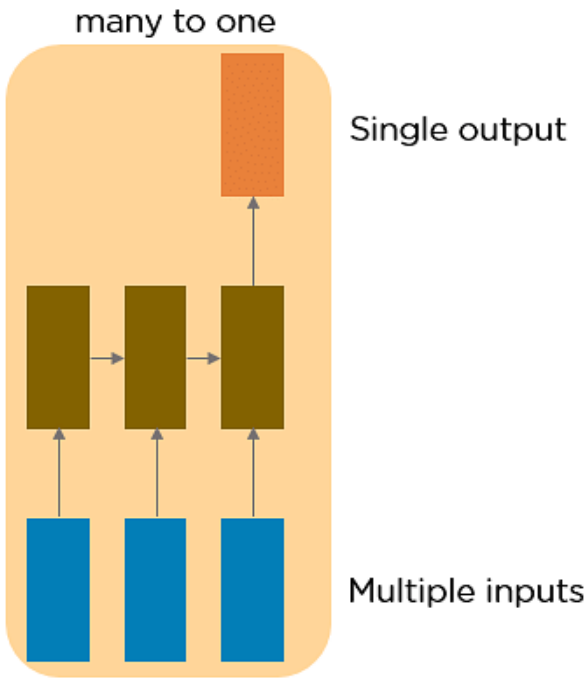


Types of RNN

- **Many-to-One:**

It takes a sequence of information as input and outputs a fixed size of the output.

Example: sentiment analysis where any sentence is classified as expressing the positive or negative sentiment.

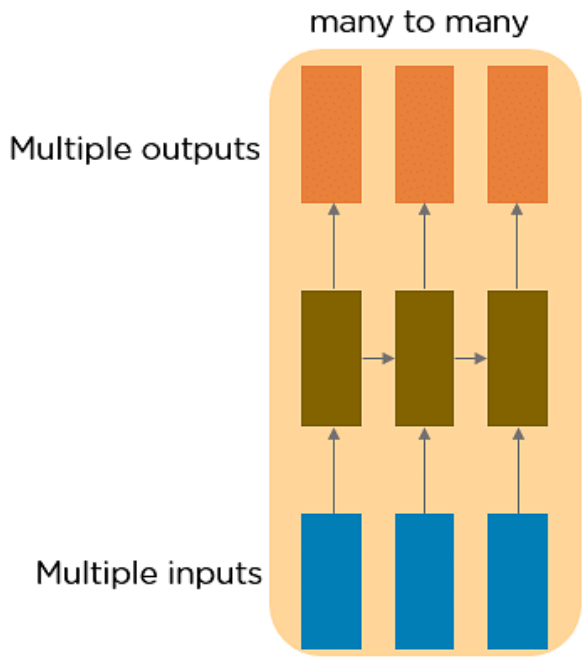


Types of RNN

- **Many-to-Many:**

It takes a Sequence of information as input and processes the recurrently outputs as a Sequence of data.

Example: Machine Translation, where the RNN reads any sentence in English and then outputs the sentence in French.

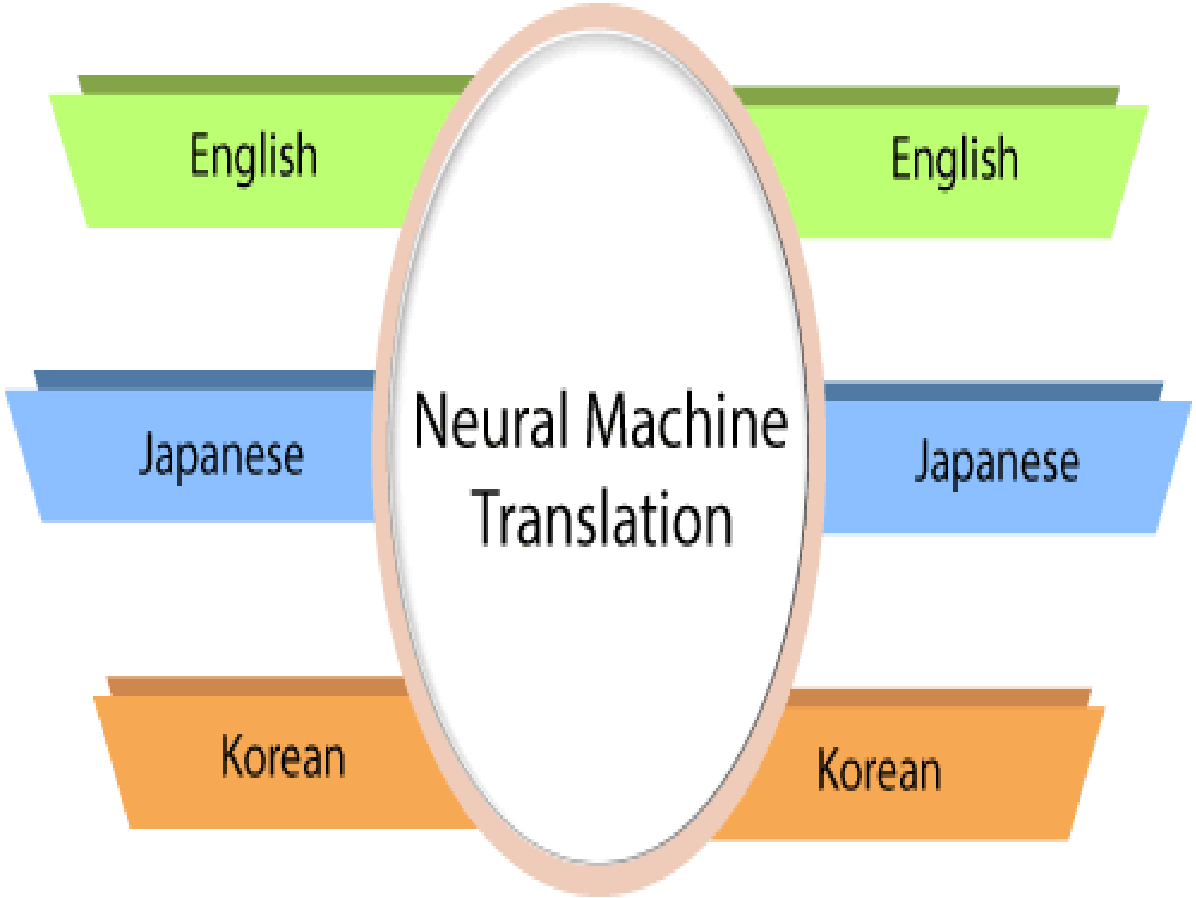


Recurrent Neural Networks (RNN)

Application of RNN

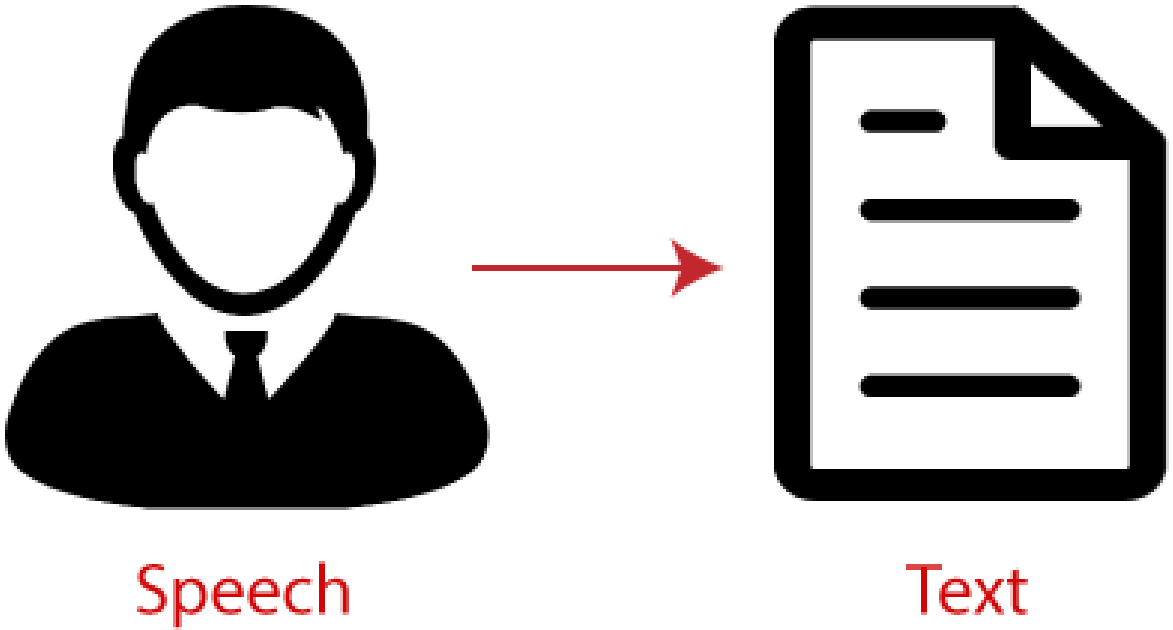
Following are the application of RNN:

1. Machine Translation



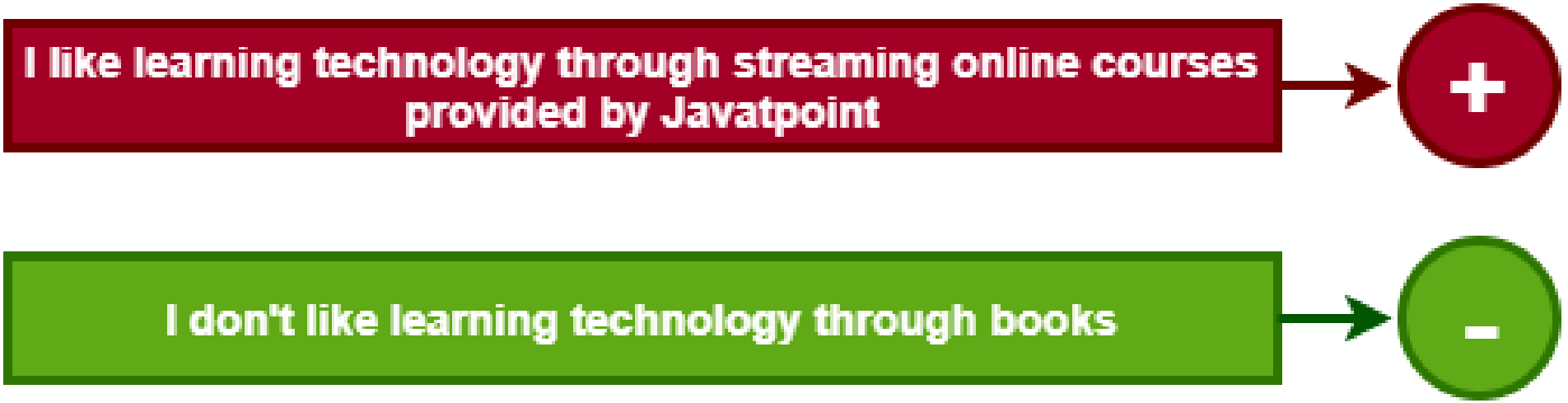
Recurrent Neural Networks (RNN)

2. Speech Recognition



Recurrent Neural Networks (RNN)

3. Sentiment Analysis



Two Issues of Standard RNNs

1

Vanishing Gradient Problem

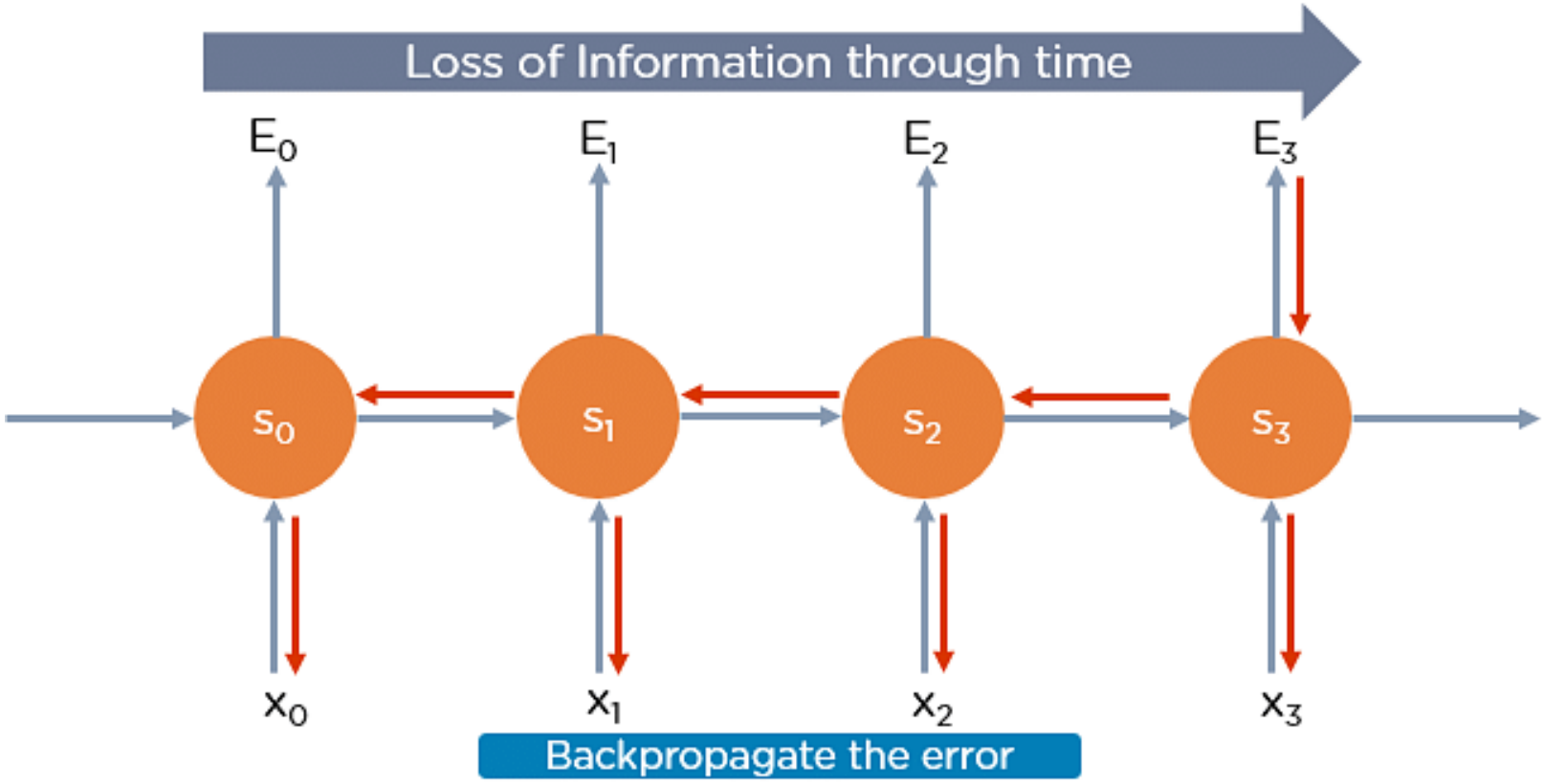
2

Exploding Gradient Problem

Vanishing Gradient Problem

RNNs suffer from the matter of vanishing gradients. The gradients carry information utilized in the RNN, and when the **gradient becomes too small**, the **parameter updates become insignificant**. This makes the training of long data sequences difficult.

Vanishing Gradient Problem



Exploding Gradient Problem

While training a neural network, if the slope tends to grow exponentially rather than decaying, this is often called an Exploding Gradient. This problem arises when **large error gradients accumulate**, leading to very large updates to the neural network model weights during the training process.