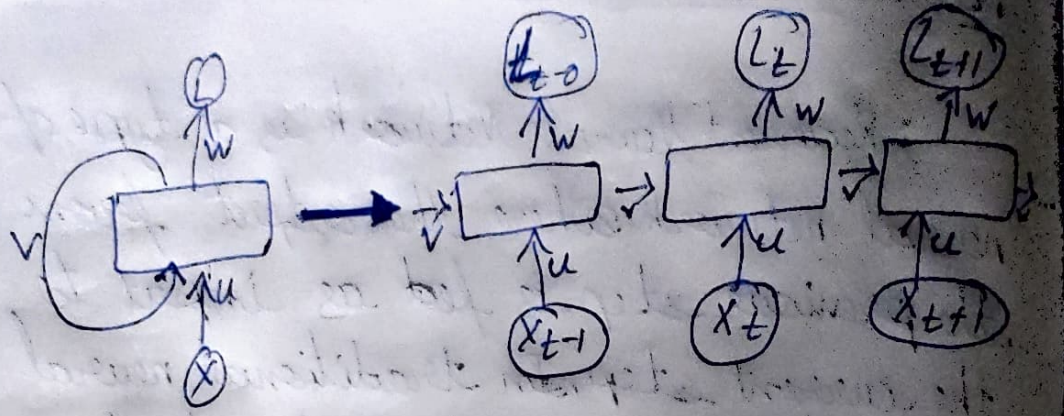
Recurrent Neural Network

→ Recurrent Neural Network is a type of neural network where the output from the previous step is fed as input to the current step. In traditional neural network, all the inputs and outputs are independent of each other.

→ Still, in case 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.

→ 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.



→ RNNs have loops that allow information to persist, enabling the network to maintain a "memory" of past inputs.

→ The main components of an RNN is the recurrent cell, which takes both the current input and the output of the previous cell to produce the current output.

Working

→ The basic idea is to have loops within the network, which allows it to use information from previous steps.

→ Given a sequence of input $x_1, x_2, \ldots x_T$, the RNN produces a sequence of hidden states $h_1, h_2, \ldots h_T$, where each hidden state h_t is influenced by the current input x_t and the previous hidden state h_{t-1}.

Formula:

i) 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

ii) The hidden state calculation using Activation function (tanh).

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

where,

→ W_{hh} → weight at current neuron.

→ W_{xh} → weight at input neuron.

→ h_t → hidden state at time step t,

→ x_t → input at time t,

iii) The formula for calculating Output:-

$$y_t = W_{hy} h_t$$

→ y_t → Output

→ W_{hy} → weight at output layer.

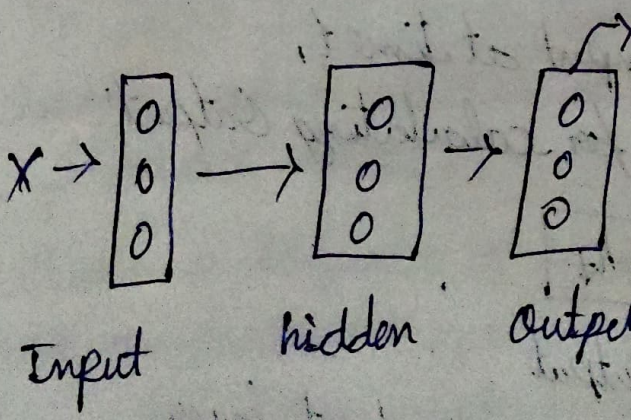
Types of RNN

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

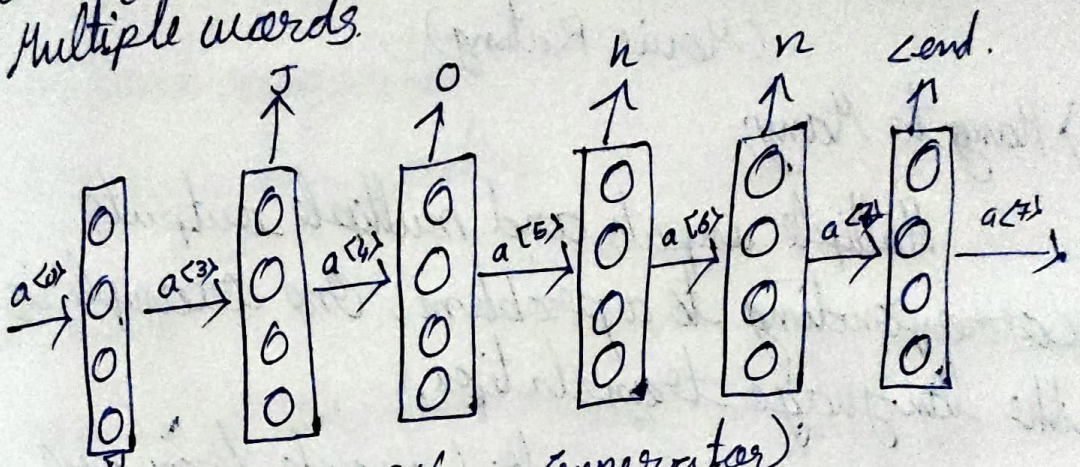
1.) One to one:-

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 & one output.



2) 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.

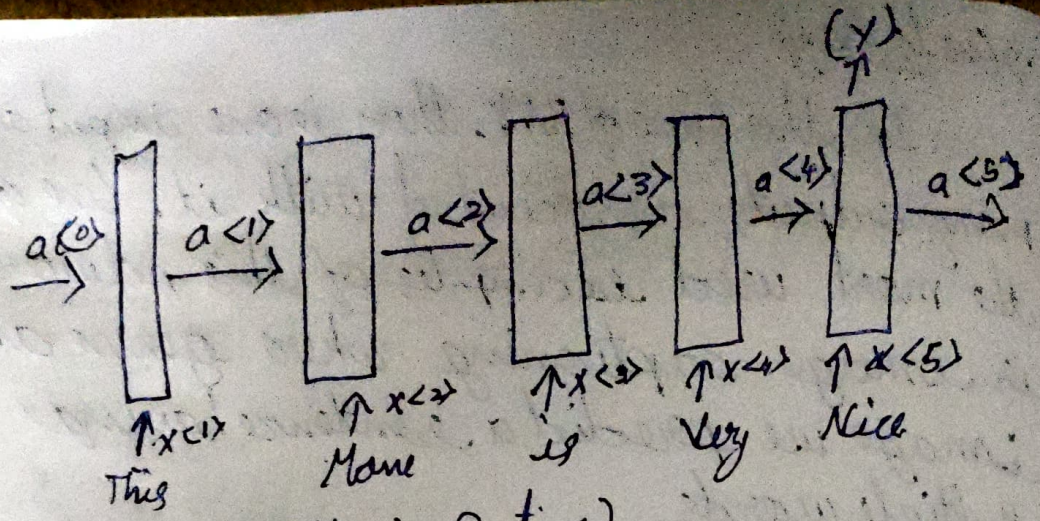


(Name Generator)

3) Many to one:

Many inputs are fed to the network at several states of the network generating only one output. This is commonly used in sentimental analysis.

Where we give multiple words as input and predict only the sentiment of the sentence as output.

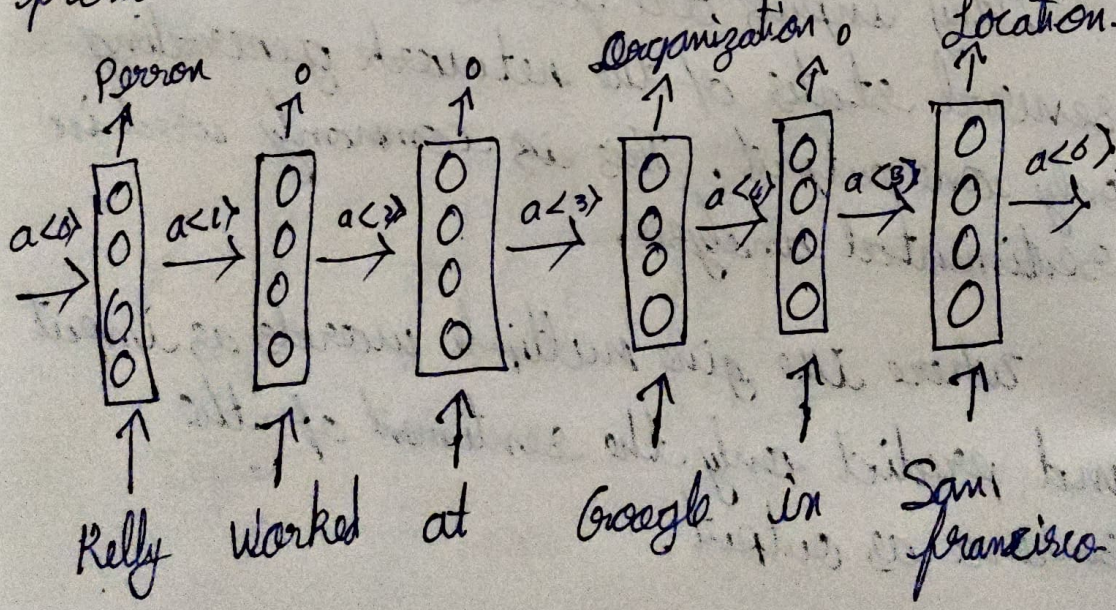


(Movie Rating)

iv) Many to Many:-

Multiple inputs and multiple outputs corresponding to a problem. One example is the language translation.

→ We provide Multiple words from one language as input & predict multiple words from the second language as output.



(Name, company & Location finder)

Applications of RN

→ Time-series p

→ Natural Langua

→ Speech recogn

→ Video classif

Challenges

→ Vanishing gra
backpropag
exponentially
through tim
learn very rl

→ Exploding g
also grow
learning a

Improvements

→ Long short
RNN designe
gradient pr
(input gate, fo

Applications of RNN

→ Time-series prediction (eg. Weather forecasting, stock prices).

→ Natural Language processing (eg. Text generation, sentiment analysis, machine translation)

→ Speech recognition.

→ Video classification

Challenges

→ Vanishing gradient problem: During backpropagation, gradients can shrink exponentially as they propagate back through time, causing earlier layers to learn very slowly.

→ Exploding gradient problem: Gradients can also grow exponentially, making learning unstable.

Improvements in RNNs

→ Long short-term Memory (LSTM): A type of RNN designed to combat the vanishing gradient problem by introducing gates (input gate, forget gate, output gate) that regulate

the flow of information.

→ Gated Recurrent Units (GRU): Similar to LSTM but with fewer gates and a simpler structure, making it computationally more efficient.

Training RNNs

→ RNNs are typically trained using Backpropagation through time (BPTT), which is a variant of backpropagation that works by unrolling the network through time and updating weights based on errors at each time step.

Use of Activation functions:

→ Commonly used activation functions in RNNs include "tanh" ~~include~~ and ReLU, but "sigmoid" is also used within gated mechanisms like LSTM cells.