UNIT- IV

Recurrent Neural Networks

Applications of Various Neural Networks

* Feed - Forward Neural Network :

Used for general Regression and classification problems.

* Convolutional Neural Network :

Used for object detection and image classification.

* Deep Belief Network :

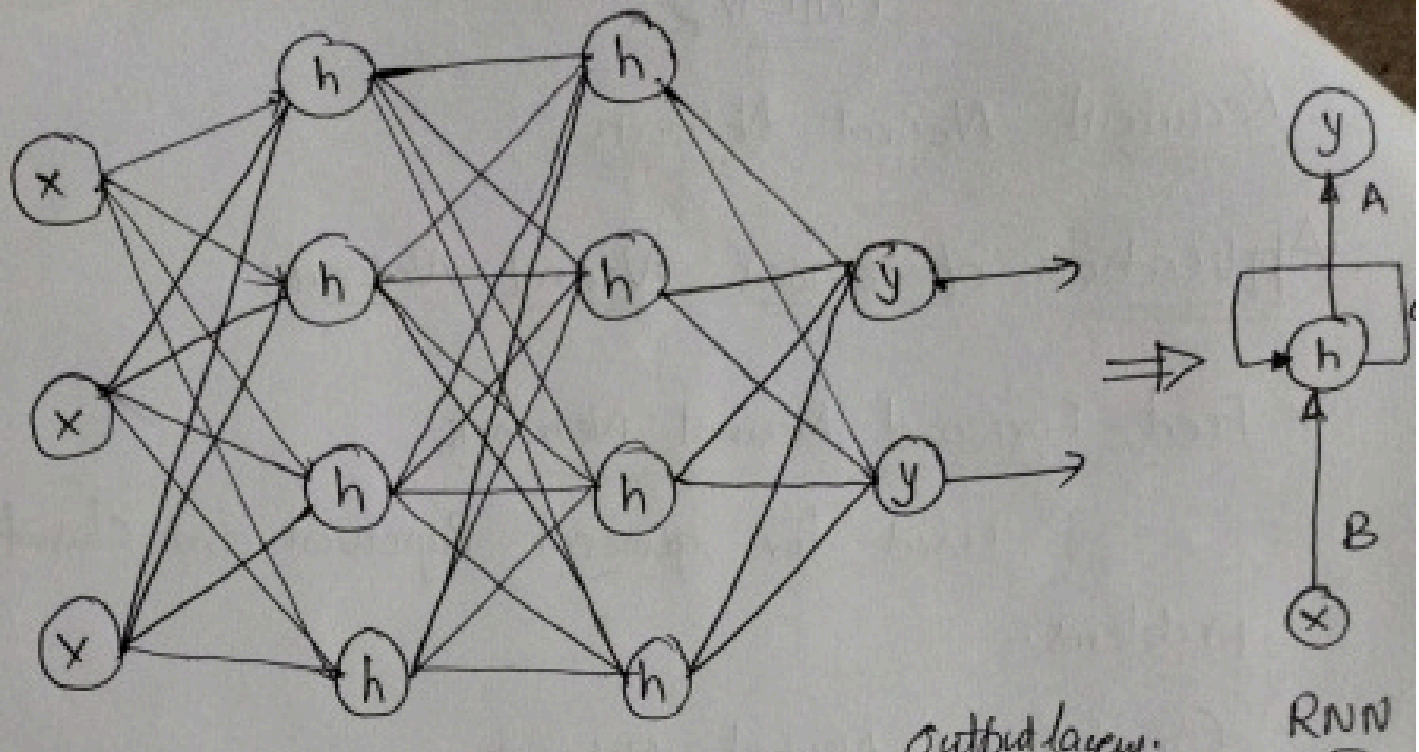
Used in healthcare sectors for cancer detection.

* RNN : Used for speech recognition, voice recognition, time series prediction, and natural language processing.

What is 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.

⇒ 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.

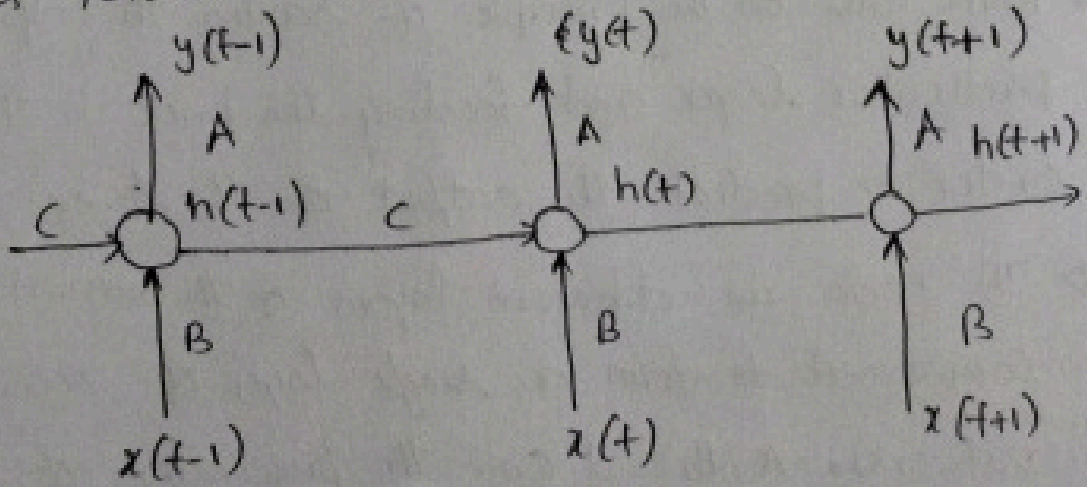


Input layer Hidden Layers Output layers.

What is 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 I/P 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.

Why RNN

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

  => Can't handle sequential data

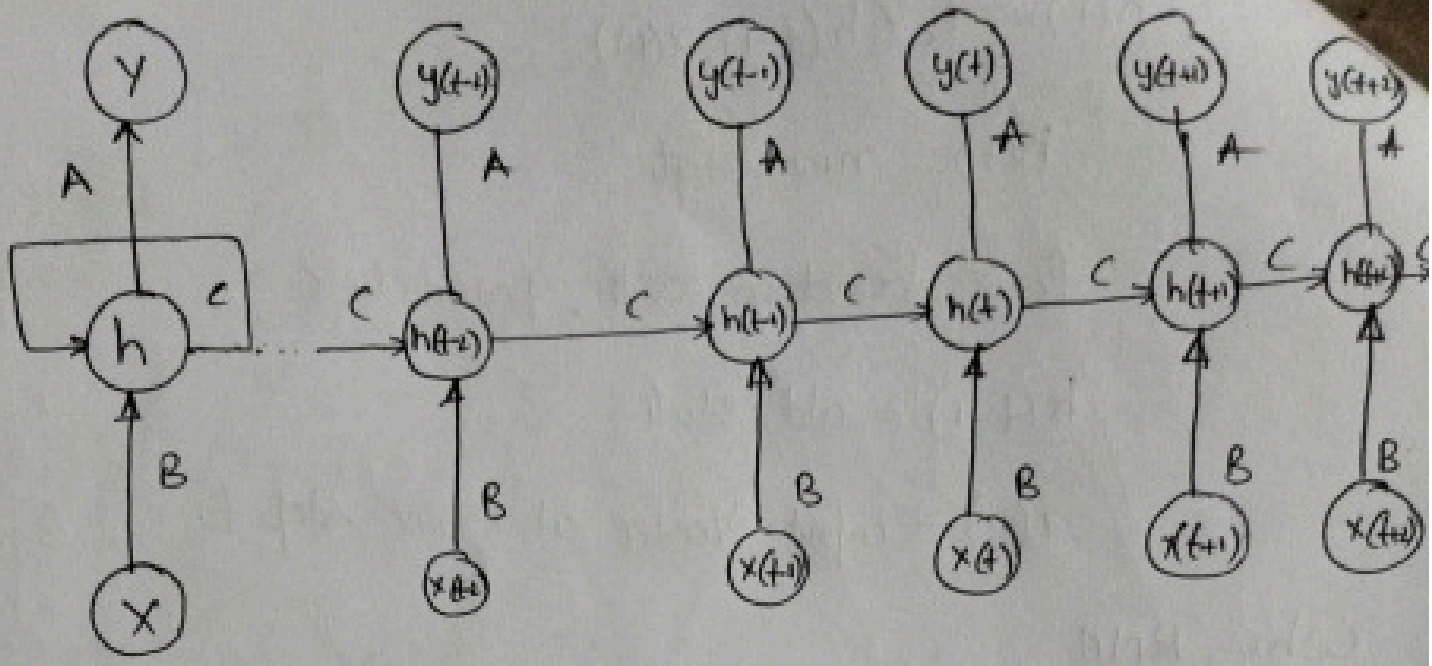
  => Considers only the current input

  => Cannot memorize previous inputs.

So These issues can handle and sequential data can handle in RNN. Accepting the current input data, and previously received inputs. RNN can memorize previous inputs due to their internal memory.

How RNN Works :

* In RNN, the information cycles through a loop to the middle hidden layer.



⇒ The input layer 'x' taken in the input to the neural nw 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.

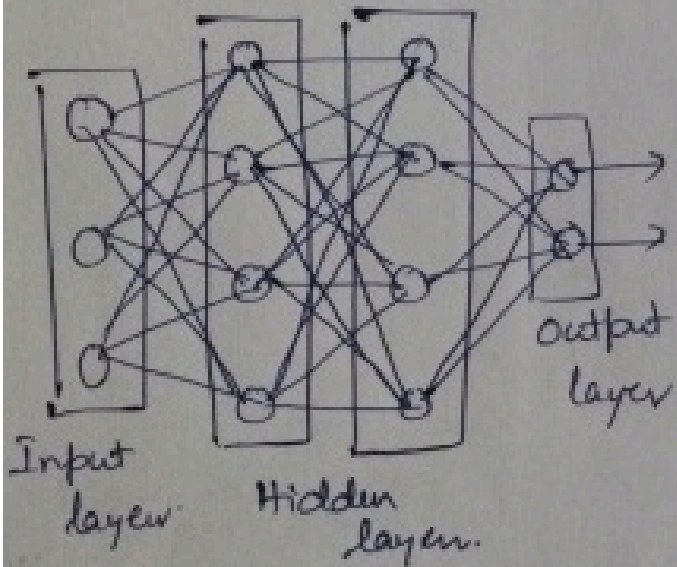
⇒ If you have a neural network where the various parameters of different hidden layers are not affected by the previous layer, i.e the neural network does not have memory, then you can use a recurrent neural network.

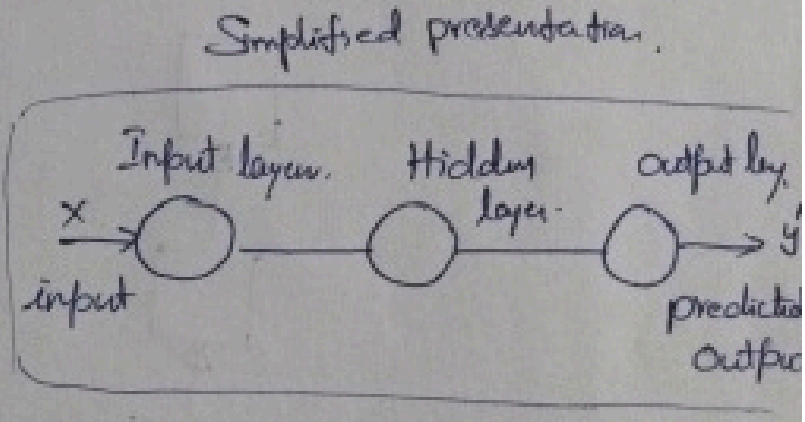
⇒ The Recurrent Neural Network will standardize the different activation functions and weights and biases so that each hidden layer has the same parameters. Then instead of creating multiple hidden layers, it will create one and loop over it as many times as required.

Feed forward Networks Vs RNN

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 networks, the decisions are based on the current input. It doesn't memorize the past data, and there is no future scope. Feed forward neural networks are used in general regression and classification problems.





Applications of RNN

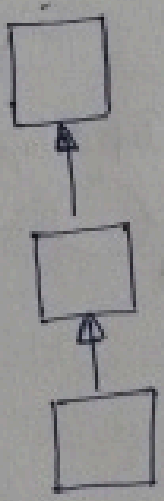
* RNNs are used to caption an image by analyzing the activities present
* Natural language processing.
* Machine Translation

Types of RNN (Recurrent Neural Network)

* One to one
* One to many
* Many to one
* Many to Many

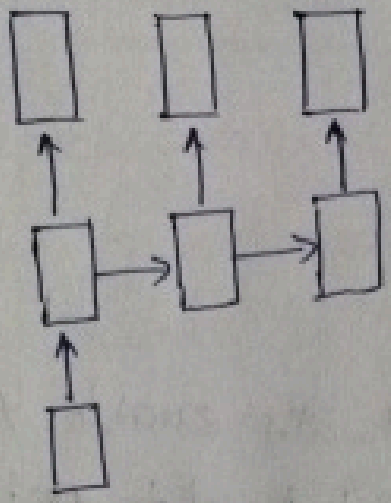
One to One

The Simplest 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 traditional neural network. The one-to-one application can be found in ima Classification.



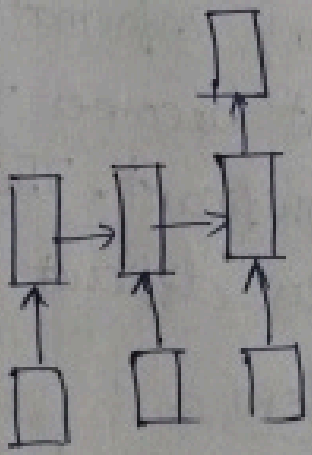
One-to-Many

One-to-Many is a type of RNN that gives multip outputs when given a single input. It takes a fixed inpu size and gives a sequence of data outputs. Its applications can be found in Music Generation and Image Captioning



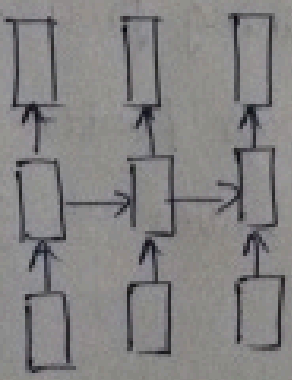
Many-to-One

Many to one is used when a single output is requi
from multiple input units or a sequence of them. It tak
a sequence of inputs to displays a fixed output. Senti
Analysis is a common example of this type of RNN.



Many-to-Many

Many-to-many is used to generate a sequence of
output data from a sequence of input units.



Issues of Standard RNNs

1. Vanishing Gradient Problem
2. Exploding Gradient Problem

Vanishing Gradient Problem:

=> Recurrent Neural Networks enable you to mod
time-dependent and sequential data problems, such
stock market prediction, machine translation, and te
generation. RNN is hard to train because of the gra
problem.

=> RNNs suffer from the problem of vanishing grad
-ents. The gradients carry information used in the RN
and when the gradient becomes too small, the paran
updates become insignificant. This makes the learni
of long data sequences difficult.

Exploding Gradient Problem:

=> While training a neural network, if the slope tend
to grow exponentially instead of decaying, This is
called an Exploding gradient. This problem arises mod
weights during the training proces.

=> long training time, poor performance, and bad
accuracy are the major issues in gradient problems.

What is Backpropagation :-

* Backpropagation is a training algorithm that is use
for training neural networks.
* When training a neural network, we are actually
tuning the weights of the network to minimize the erri
with respect to the already available true values
(labels) by using the Backpropagation algorithm.
* It is a Supervised learning algorithm as we find
errors with respect to already given labels.

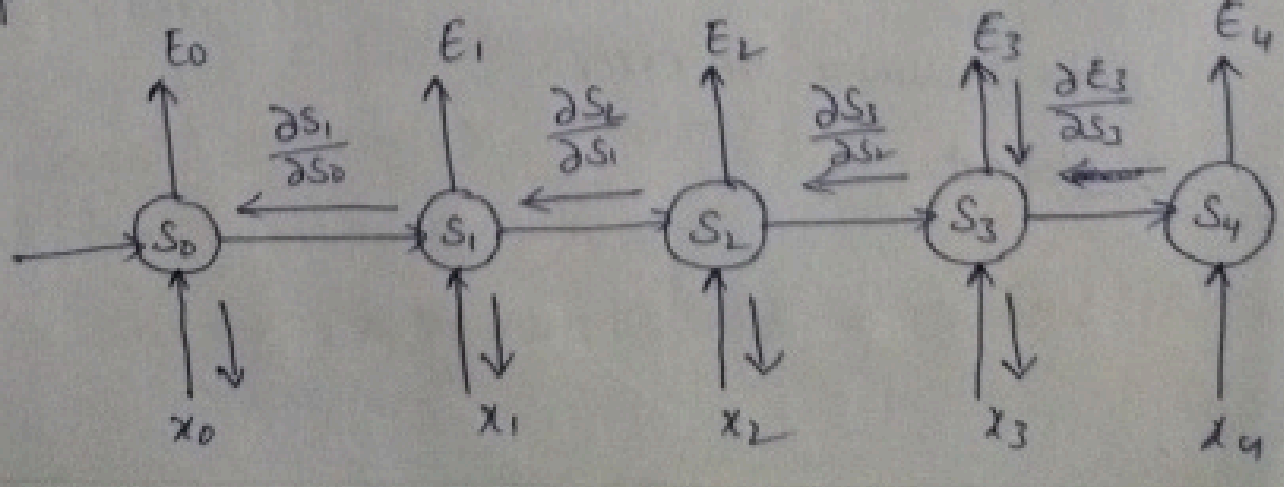
Algorithm :-

1. Present a training input pattern and propagate it
through the network to get an output.
2. Compare the predicted outputs to the expected output
and calculate the error.
3. Calculate the derivates of the error with respect to
the network weights.
4. Use these calculated derivatives to adjust the weigh
to minimize the error.
5. Repeat.

Backpropagation Through Time (BPTT)

* RNN uses a technique called Backpropagation Through tim
to backpropagate through the network to adjust their
weights so that we can reduce the error in the netwo
* It got its name "Through time" as in RNN we deal
with sequential data and every time we go back it
like going back in time towards the past.
* In the BPTT step, we calculate the partial derivative
at each weight in the network. So if we are in time
$t=3$, Then we consider the derivation of $E3$ with respect
to that of $S3$. Now, $X3$ is also connected to $S3$, its
derivative is also considered.
* Now if we see $S3$ is connected to $S2$ so $S3$ is
depending on the value from $S2$ and here derivative
of $S3$ with respect to $S2$ is also considered. This acts
as a chain rule and we accumulate all the dependency
with their derivatives and use it for error calculation.

BPTT



Advantages of RNN

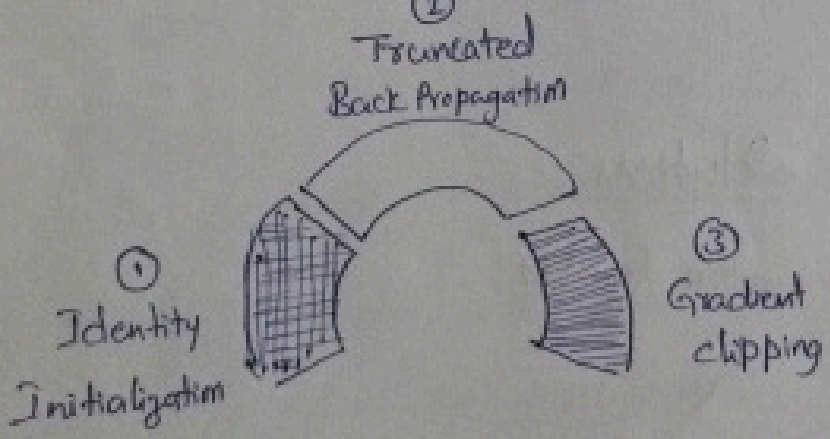
* Possibility of processing input of any length
* Model size not increasing with size of input
* Computation takes into account historical information
* Weights are shared across time.

Disadvantages of RNN :

* Computation being slow
* Difficulty of accessing information from a long time a
* Cannot consider any future input for the current sta

Solutions for Vanishing Gradient and Exploding Gradient

Exploding Gradient Solutions



1 Identity Initialization

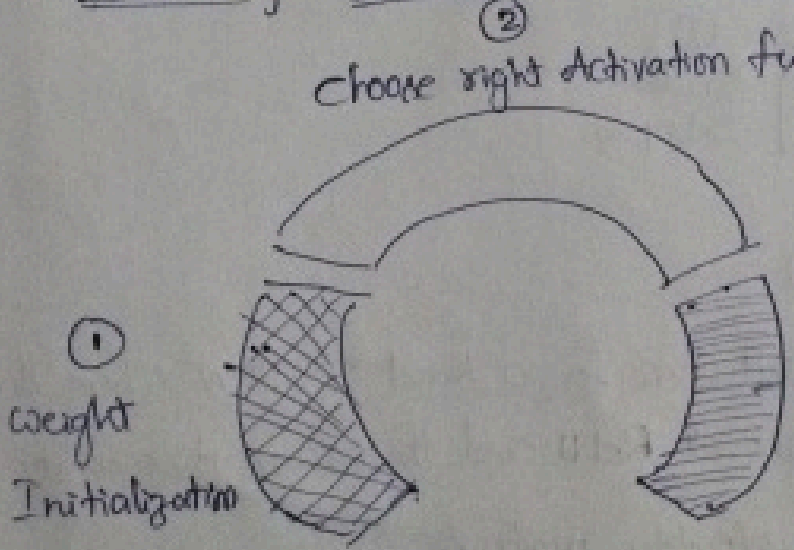
* The identity RNN architecture is a kind of RNN where the activation functions are all ReLU and the recurrent weights are initialized to the identity matrix.

② Truncate Propagation :-

The input is considered as fixed-length Subsequen
in truncated backpropagation Through time (TBPTT). The
hidden state of the previous subsequence is passed on a
input to the following subsequence in the forward pass
on the other hand, the computed gradient values are drop
at the end of each subsequence as we move back in
gradient computation.

③ Gradient Clipping :- It is a technique used to cope wi
the exploding gradient problem. Sometimes encountered
when performing backpropagation. By capping the maximu
value for the gradient, this phenomenon is controlled in
practice.

* Vanishing Gradient Solutions



Variants of RNN's

* LSTM
* Bidirectional RNN
* GRU
* Encoder & Decoder RNN. (or) Sequence-to-Sequence RNN

LSTM (Long Short Term Memory Network)

It is mainly used for resolved the problem in RNN i.e Vanishing gradient Decent.

* In the given Example statement

Venky stays in India. Venky has a friend, who name is Dev. Dev is a AI faculty. Venky learns AI from .....

In Simple RNN, we don't know which part of the sentence is important for the given question.

* In the above statement which sentence is Underline by single line i.e irrelevant
* which sentence is underline by two lines i.e relevant.
* which sentence is not in underline i.e also relevant

following Concepts in LSTM

* long term memory
* Short term memory
* forget gate
* Input gate
* Output gate.

Long term memory

It is like a convey belt (long from starting to end)

Short term memory

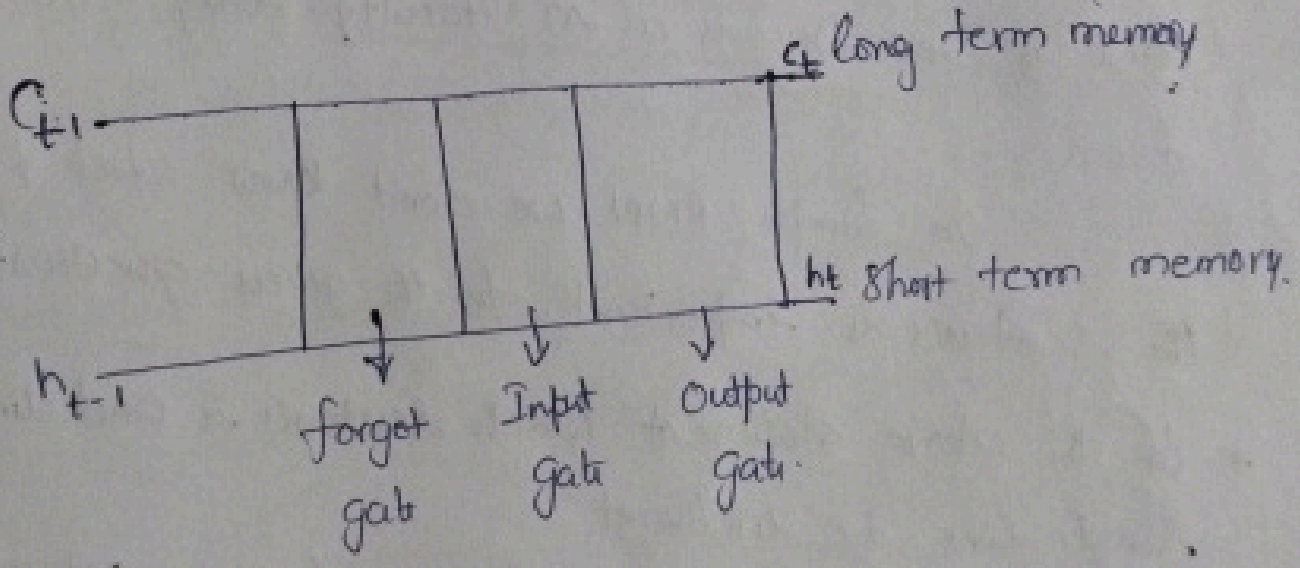
Intrested in the part of sentence.

Forget gate : It is helps you to remove the irrelevant informa

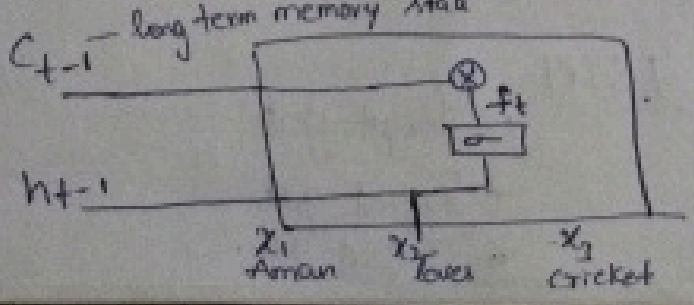
Input gate : It is helps you to keep the relevant information

output gate : It gives output with the help of forget gate & Inpu gate.

LSTM cell / Unit / model



(*) Forget gate and process :



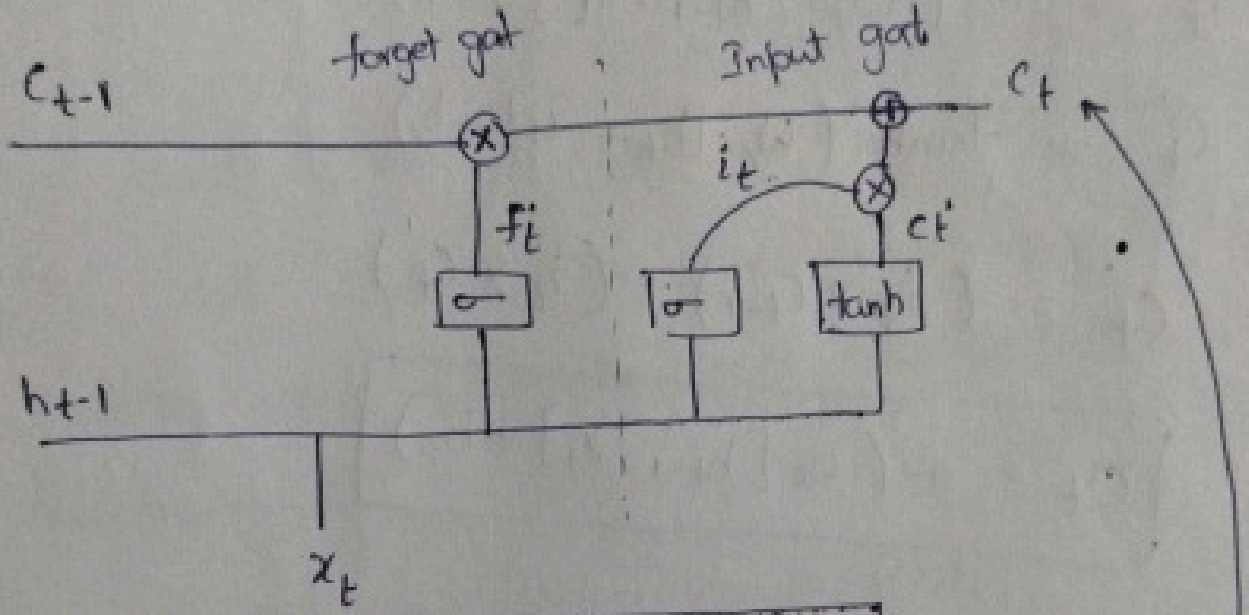
$$f_t = \sigma(w_f \cdot [h_{t-1}, x_t] + b_f)$$

w_f → forget gate weight

b_f → forget gate bias

forget gate main importance in LSTM is gave the less importance to the data to next state.

Input gate process



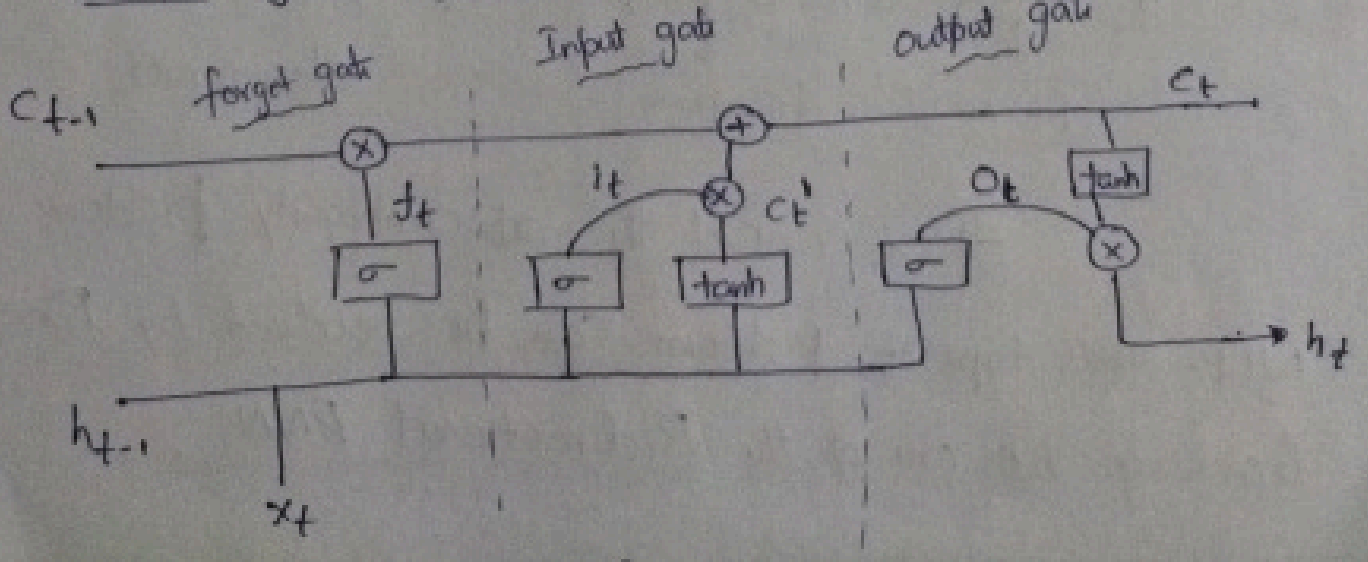
$$i_t = \sigma\left(w_i \cdot [h_{t-1}, x_t] + b_i\right)$$

$$C_t' = \tanh\left(w_c \cdot [h_{t-1}, x_t] + b_c\right)$$

$$C_t = f_t \times C_{t-1} + i_t \times C_t'$$

i - input ga
c - candida

output gate process.



Mathematical Calculation of different gates

$$f_t = \sigma(w_f \cdot [h_{t-1}, x_t] + b_f)$$

$$i_t = \sigma(w_i \cdot [h_{t-1}, x_t] + b_i)$$

$$c_t' = \tanh(w_c \cdot [h_{t-1}, x_t] + b_c)$$

$$c_t = (f_t' \times c_{t-1}) + (i_t' \times c_t')$$

$$\boxed{o_t = \sigma(w_o [h_{t-1}, x_t] + b_o)}$$

Bidirectional RNN

We are facing one problem in NLP, Through NER technique. NER - means Named Entity Recog-nition.

Example

Person → Venky loves apple, it keeps him healthy

Fruit

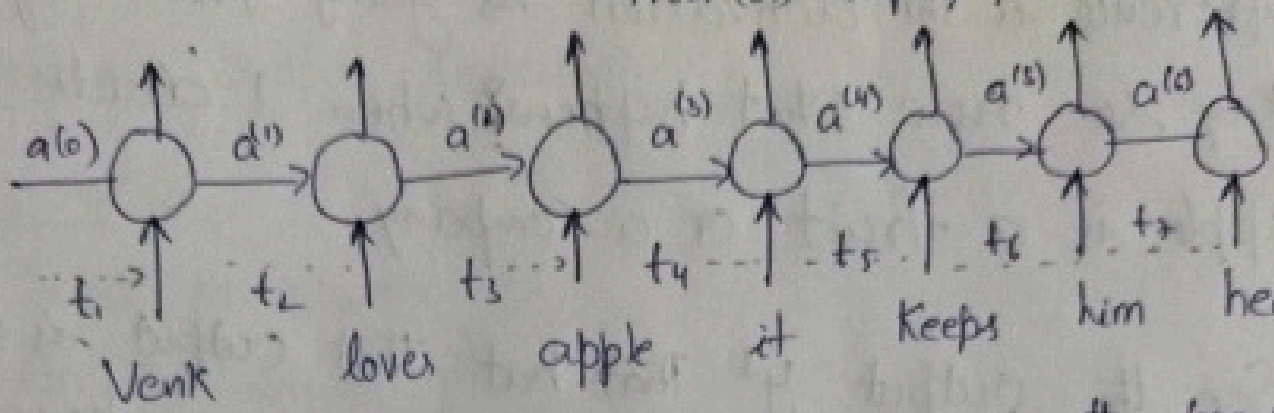
* Venky loves apple, the company produces best electronics

Person → Venky

Company → apple

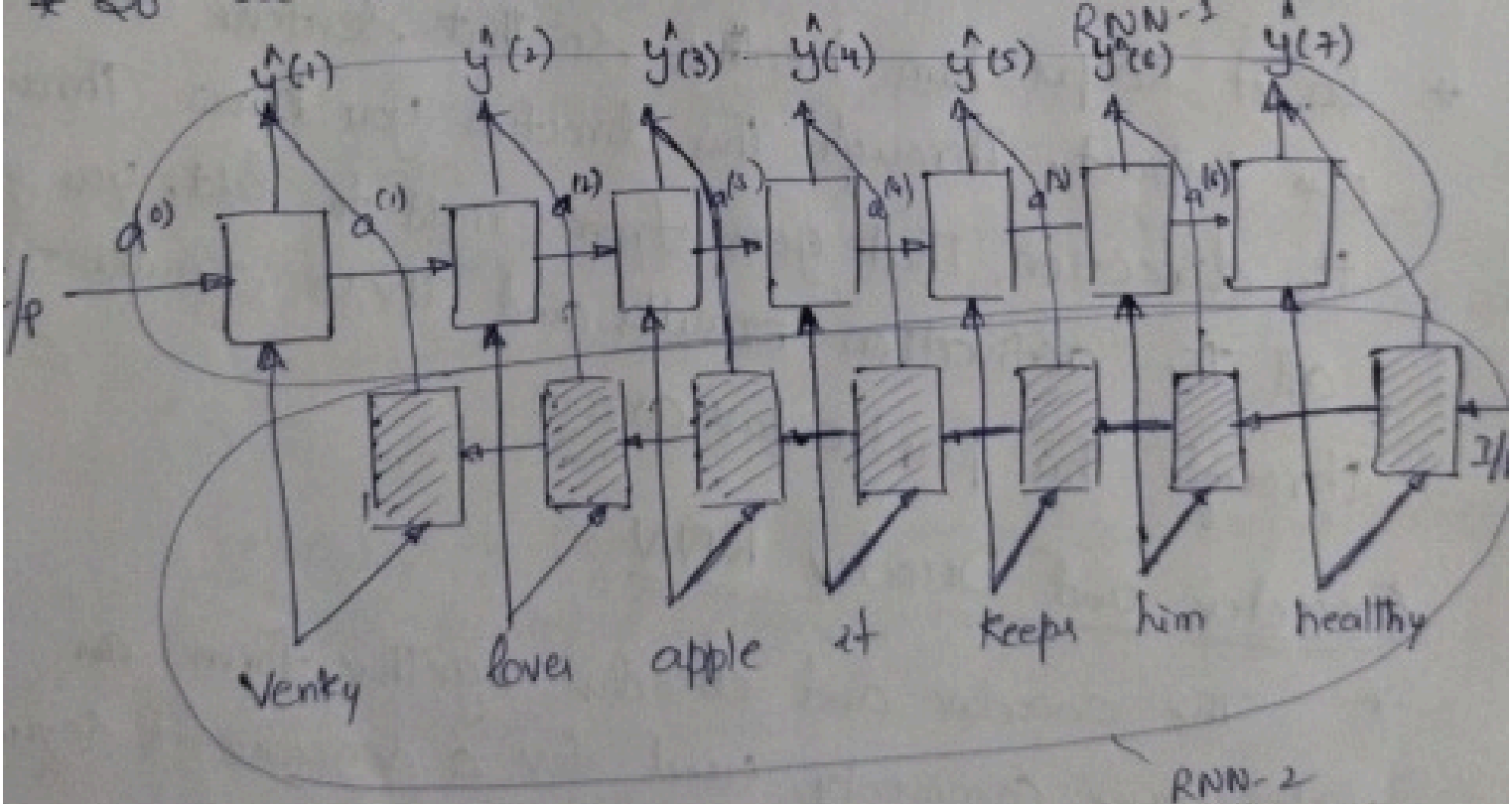
It is one of the ambiguity problem in NLP. This type of problem can be resolved by Deep learning i.e one of the Bidirectional RNN.

General RNN, It is a single direction from left to right



* The word apple will have influence only the previous words which is Venky lover now people have a Comm Confusion.

* So we will use Bidirectional RNN



* In the above network, the layer you know the forward basically single direction left to right a layer.
* Here add a new layer which

* So it's a same kind of RNN cells but the only difference is the activation is going from right to left okay so now what happens? when I decide whether apple is a fruit or a company.

* See the output y^3 ~~that~~ hat that output is a function of $a^{(1)}$ and $a^{(2)}$ okay so basically you have the influence of Venky loves the previous words on your y^3 hat. Now you have influence of your future words as well.

* So it keeps him healthy, so that sentence will have an impact, through this direction you know through the direction that goes from right to left, you get all the activation results and these results feed through this particular arrow

Encoder and Decoder RNN :

* The encoder and decoder together form an architecture commonly used for a sequence-to-sequence tasks.
* This Architecture is widely used in Natural Language Processing tasks like machine translation, Summarization and more.

The encoder processes the input and creates a context vector, and the decoder uses this context vector to generate output sequence.

Encoder :-

* Input processing
* Hidden States
* Context Vector.

Input Processing :-

=> The encoder is responsible for processing the input sequence step by step.

=> At each time step, it takes an input and processes it, producing a hidden state.

Hidden States :-

=> The hidden state at each time step captures information about the input sequence up to that point

=> The sequential nature of the processing allows the encoder to consider the context and dependencies within the input sequence.

Context Vector :-

=> The final hidden state or a summary of these hidden states is often referred to as the "context vector"

=> The context vector is a compressed representation of the entire input sequence.

Decoder -

* Initial state
* Output Generation
* Training and inference.

Initial state :

=> The Decoder takes the context Vector from the encoder as its initial hidden state.

=> It also takes an initial input, typically a special token representing the begining of the sequence.

Output Generation :

=> Similar to the encoder, the decoder processes inputs step by step.

=> At each time step, it generates an output token based on its current hidden state and the input token.

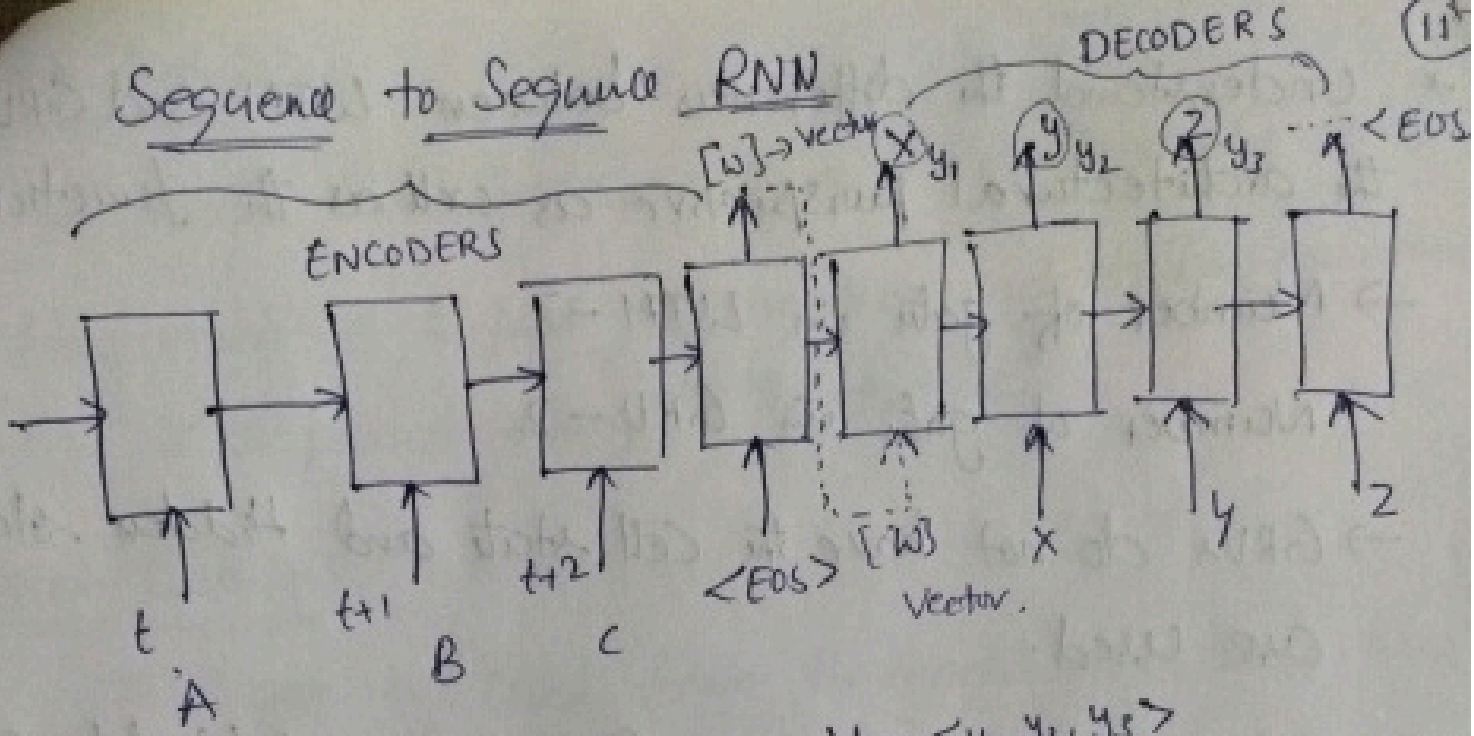
=> The output at each step becomes the input for the next step, creating an autoregressive process.

Training and inference :

=> During training, the decoder is provided with the correct target sequence as input at each step. the model is optimized to minimize the difference between the predicted & actual sequences.

=> During inference, the decoder operates in a self-feeding.

Sequence to Sequence RNN



$x = \langle x_1, x_2, x_3 \rangle$ $y = \langle y_1, y_2, y_3 \rangle$

Vectors → Embedding layer.
one hot Representation

GRU (Gated Recurrent Unit)

* This is one of the recent innovations, It was invented by 2014 by K. Cho.
* This is having the same gene as LSTM, They are very closely related.
* When we say, they are related, they share many good features with newer generation getting better over the previous one.
* GRU's can effectively address the Vanishing gradient problem just like LSTM.

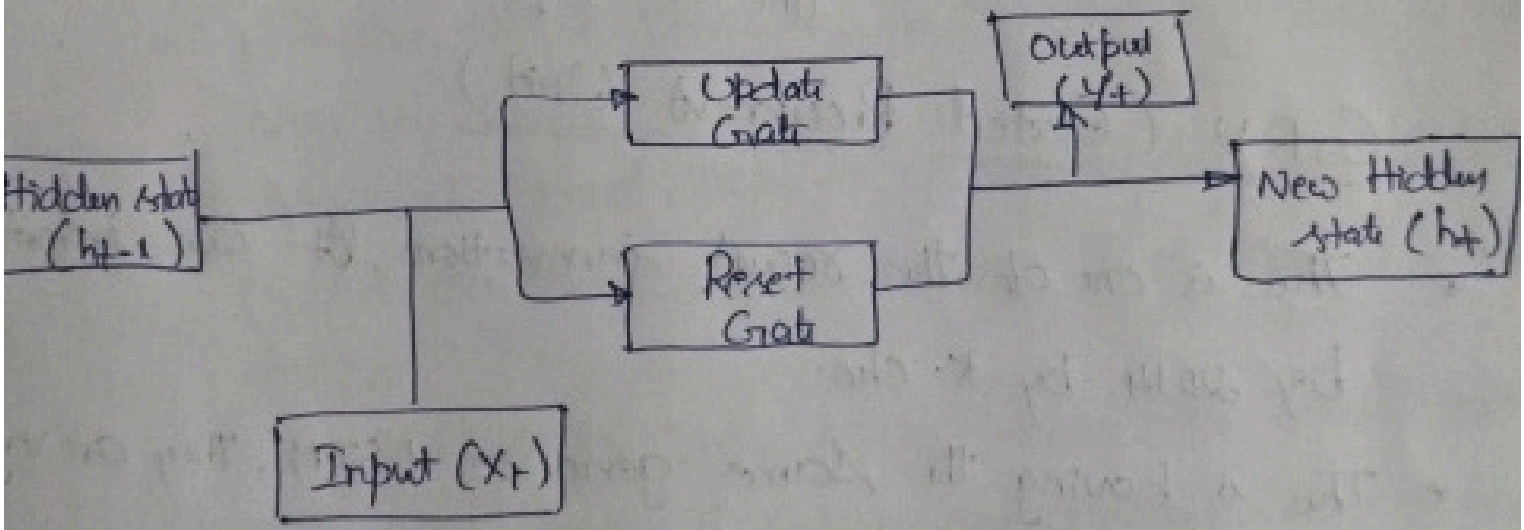
* Understand the differences between LSTM and GRU in the architectural perspective as well as the functioning

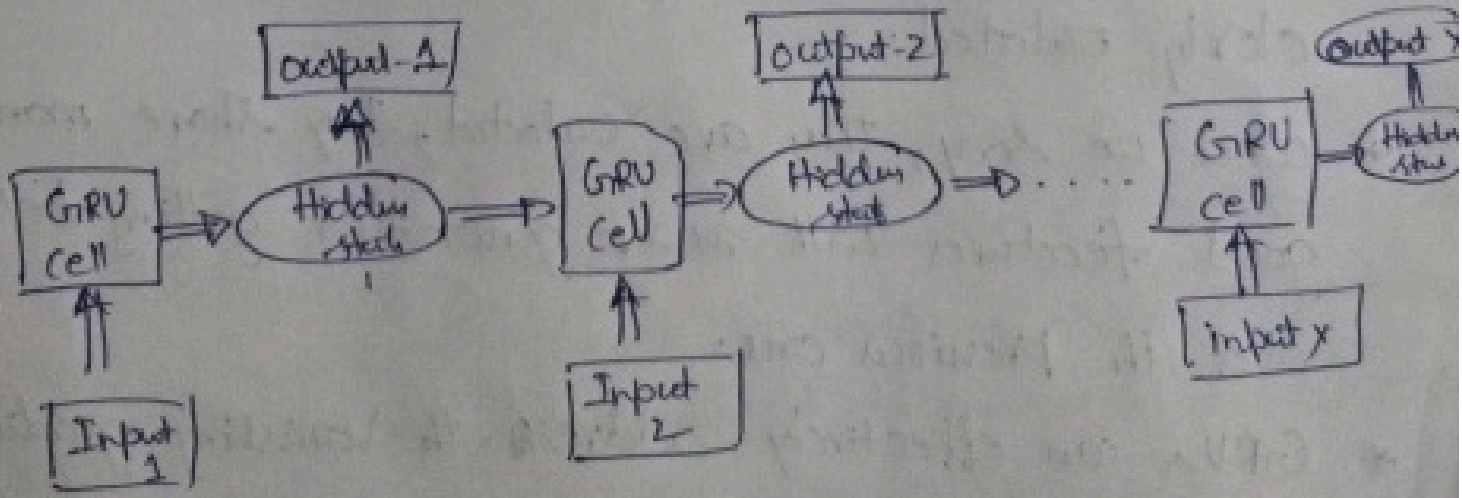
→ Number of gates in LSTM - 3

Number of gates in GRU - 2

→ GRUs do not have the cell state and Hidden state are used.

* GRU gates are names Reset Gate and Update Gate





Update Gate (z)

The major task of this gate is to tell the model about "How much of the past information needs to be maintained i.e to be passed along to the future."

Reset Gate (r)

This gate is used from the model to decide how much of the past information to forget.

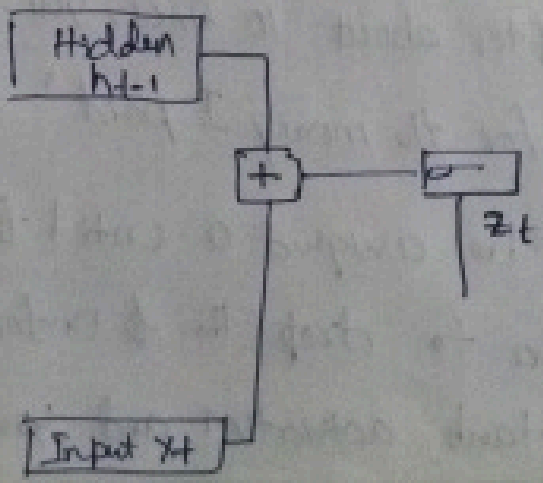
* As usual there are weights associated with every state! remember it!

Update gate

Input represented by x_t, Previous h_{t-1} state information multiplied with respective weight as the parameter

Sigmoid activation is used to derive Z_t

$$Z_t = \sigma(W_{(z)} x_t + h_{t-1})$$

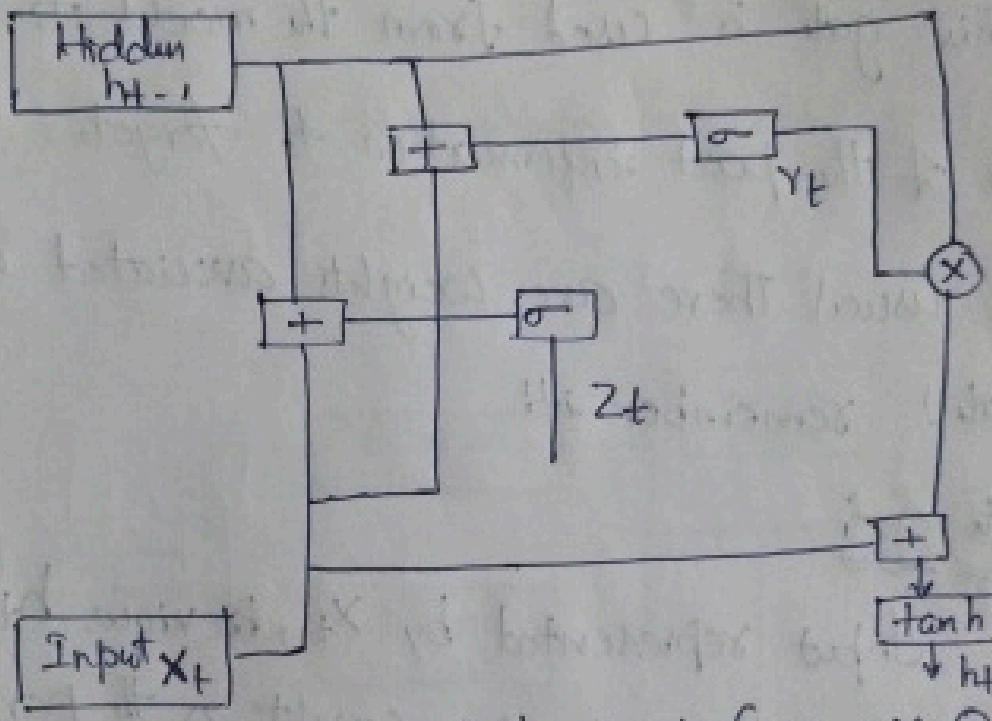


Reset gate (r)

Input represented by x_t, previous h_{t-1} state inf multiplied with respective weights as a parameters.

Sigmoid activation is used to derive r_t

$$r_t = \sigma(W_r \cdot x_t + h_{t-1})$$



$$h_t' = \tanh(W \cdot x_t + r_t \odot h_{t-1})$$

for Example "The movie is directed by X, it has featured Y, Music by Z, etc... After about 10 lines, you say the movie, I think a bad one for the money I paid"

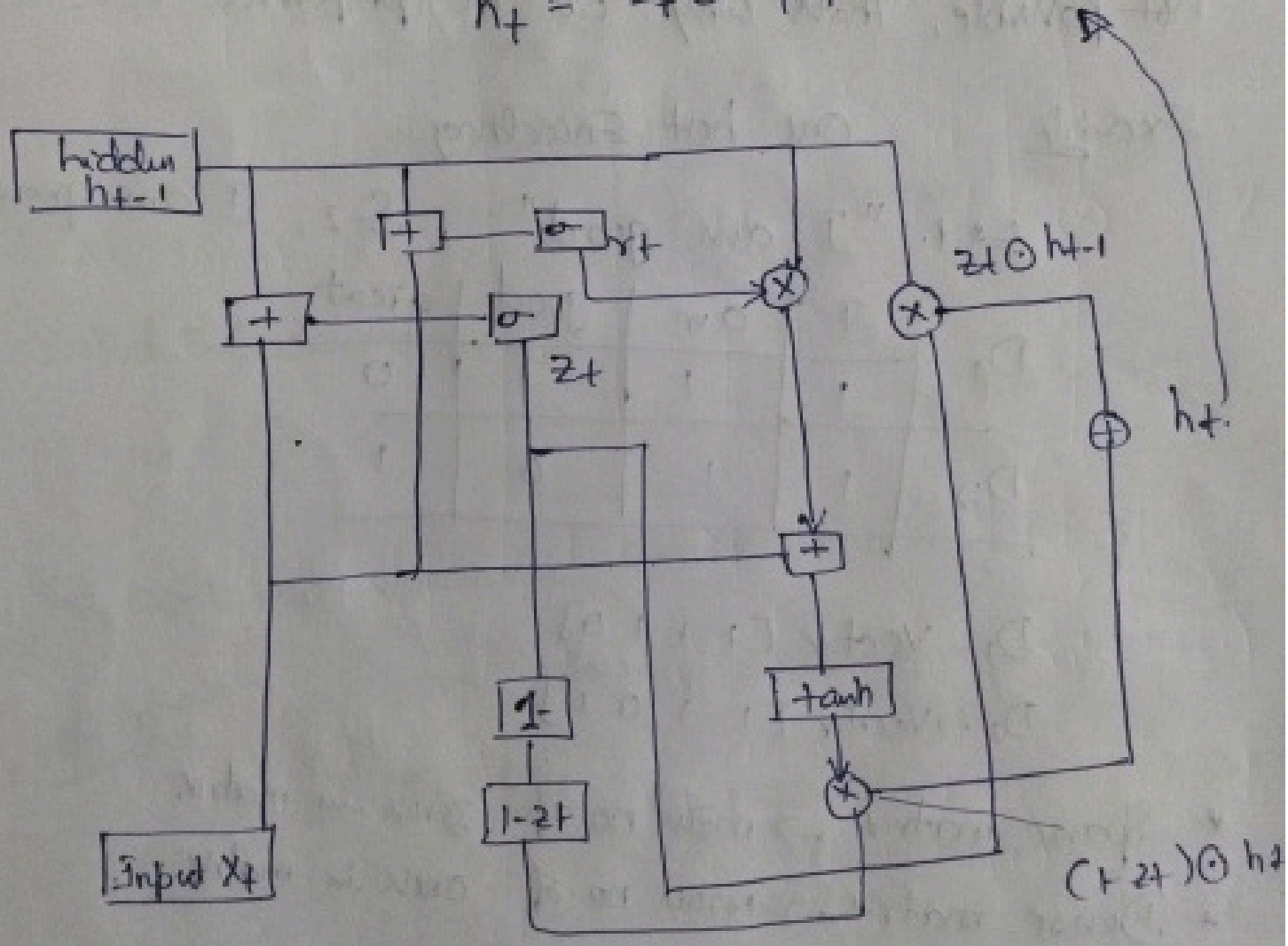
Here r_t will be assigned 0 until the final phrase is in analyze so as to drop the unfocused items.

Then comes the tanh activation and you get h_t'

* As the last step, the network needs to calculate h_{t-1} vect
which holds information for the current unit and passe
it down to the network.

* In order to do that the update gate is needed.

$$h_t = z_t \odot h_{t-1} + (1 - z_t) \odot h_t'$$



Word Embedding using Word2Vec

Word Embedding is nothing but Converting word into number.

Why : for Example : "I am good"

→ processing of this sentence in ML, it is not possible, That's why we are preferred "one hot encoding"

Example One hot Encoding

S-1. "I am good" , S-2. I am great

	I	am	good	great
D_1	1	1	1	0
D_2	1	1	0	1

D_1 - Vector [1 1 1 0]

D_2 → Vector [1 1 0 1]

* Sparse matrix → more no. of zeros in matrix.
* Dense matrix → more no. of one's in matrix.

Word2Vec

Word2Vec is a technique in Natural language processing (NLP) for obtaining vector representations of word.

Example	"fruit"	"animal"	is edible
Apple	0.9	0.01	0.9
Mango	0.85	0.02	0.92
Elephant	0.1	0.9	0.01

Based on feature in the above table which words are close to each other

Word Embedding
- using Pretained model
- Train own embedding.

* word2vec internally Neural Network model.

one hot Representation

I am doing good

I [1 0 0 0]
am [0 1 0 0]
doing [0 0 1 0]
good [0 0 0 1]

(*) word2vec models / Architectures. Types:
- CBOW ✓
- Skip gram ✓

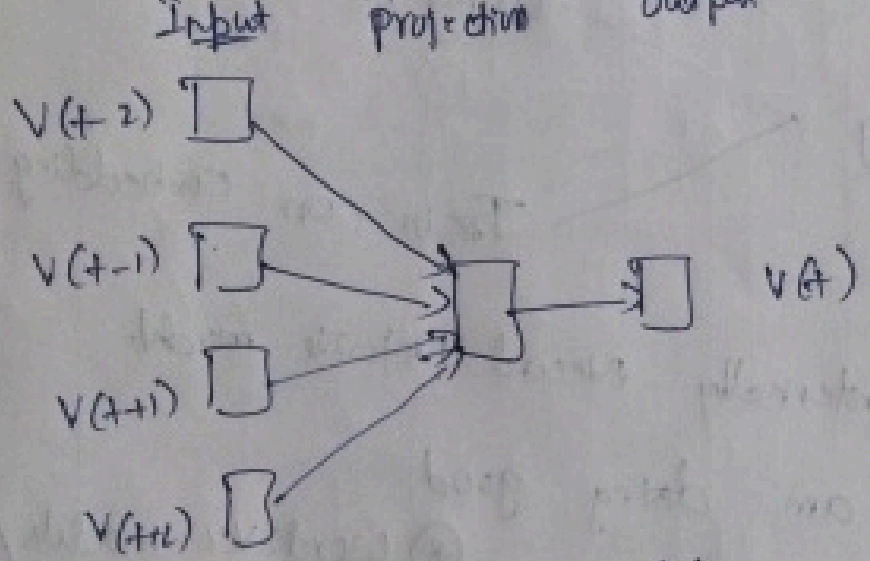
CBOW (Continuous Bag of words)

The CBOW model predict the target word from its surrounding Context words.

for Example

In the Sentence "The cat sat on the mat", if we use "cat" as our target word, the CBOW model will take "The", "sat", "on", "The", "mat" as context and predict the word "cat"

* The model is beneficial when we have a small dataset and it is faster than the Skip-Gram model.



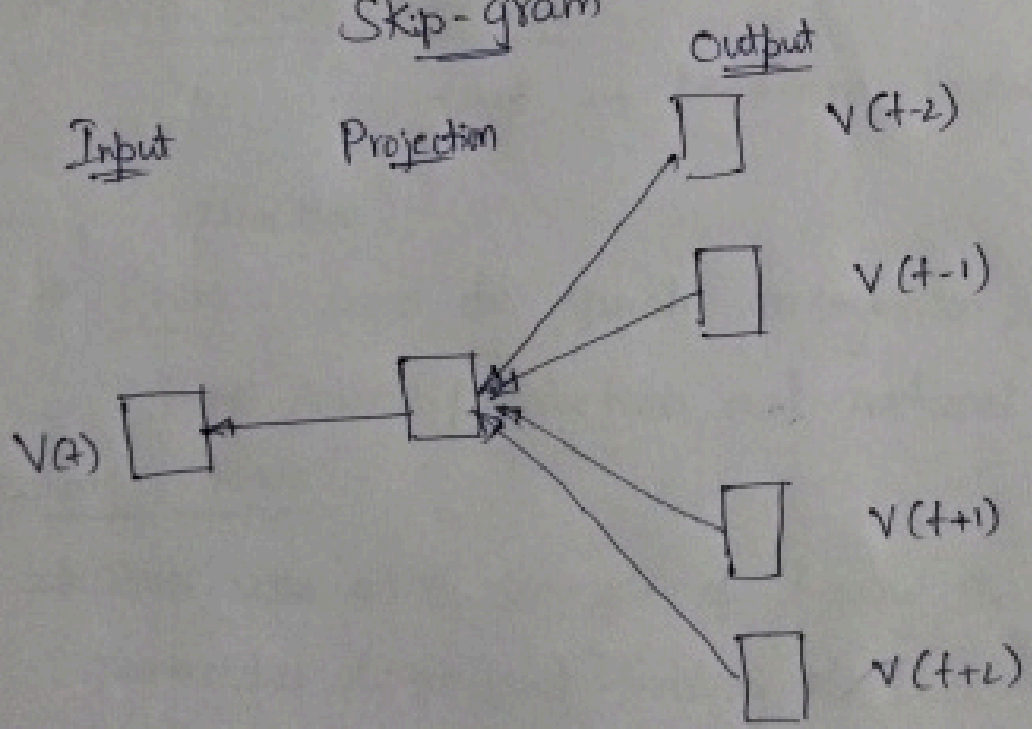
Continuous Skip gram model

The Skip-gram model predicts the surrounding context words. In other words, it uses a single word to predict its surrounding context.

For example, if we again take the sentence "The cat sat on the mat." The Skip-gram model

will take "cat" as the input and predict "The", "Sat", "on", "the", "mat".

This model works well with a large dataset and with rare words. It's computationally more expensive than the CBOW model due to its task of predicting multiple context words. However, it provides several advantages, including an improved ability to capture semantic relationships, handle rare words, and be flexible to linguistic context.



network. A, B, and C are the parameters of the network.