LSTM (Hochreiter & Schmidhuber)

[Long Short Term Memory]

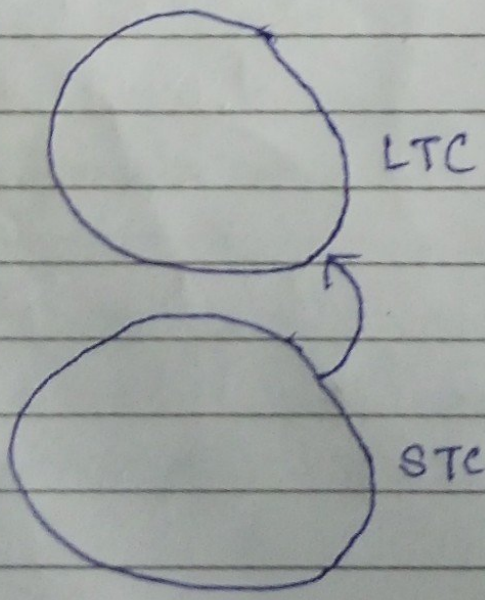
ANN → Disadvantages ← RNN

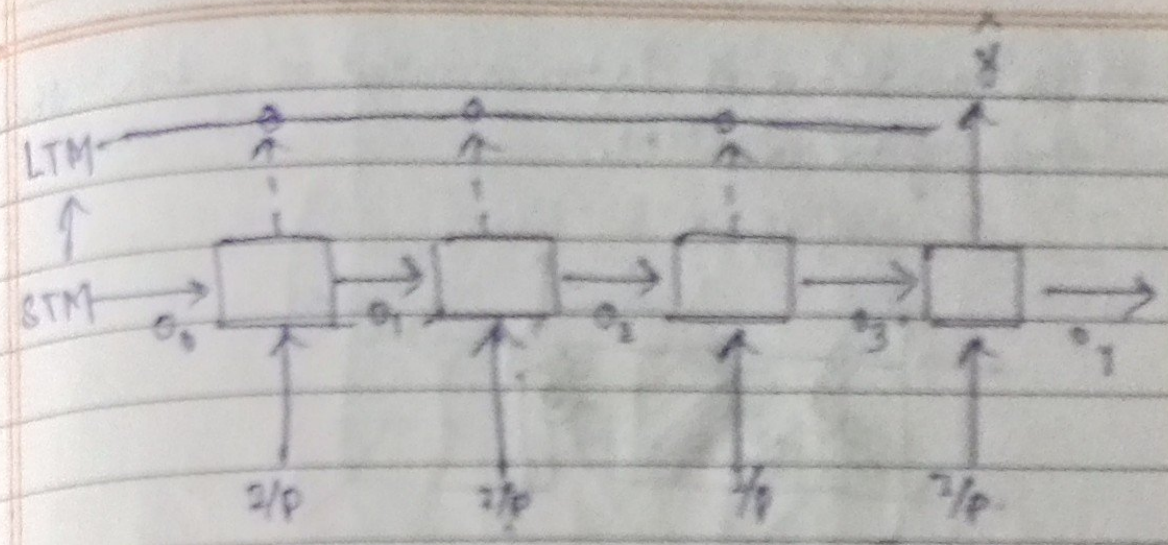
(i) It's not useful for sequential data.

(ii) Hidden unit used to return feedback that propagate with other nodes. where time stamp plays a vital role. When the chain become long, then the machine (can) forgot the old data.

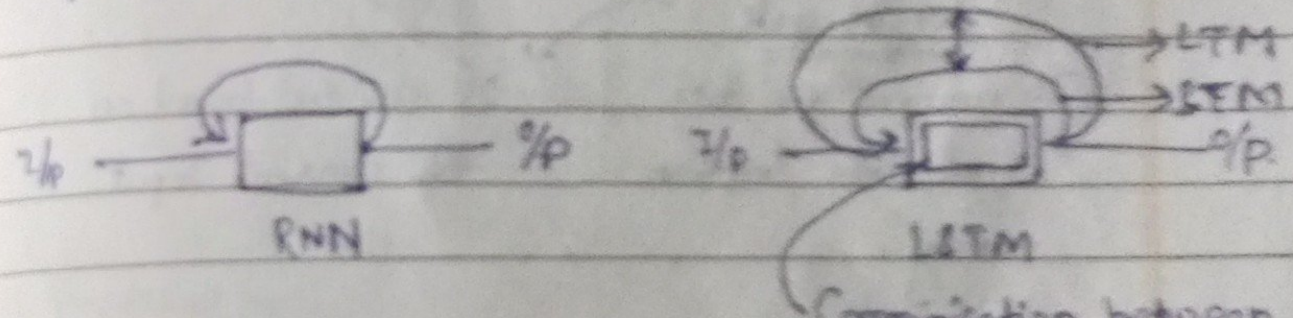
Vanishing Gradient →

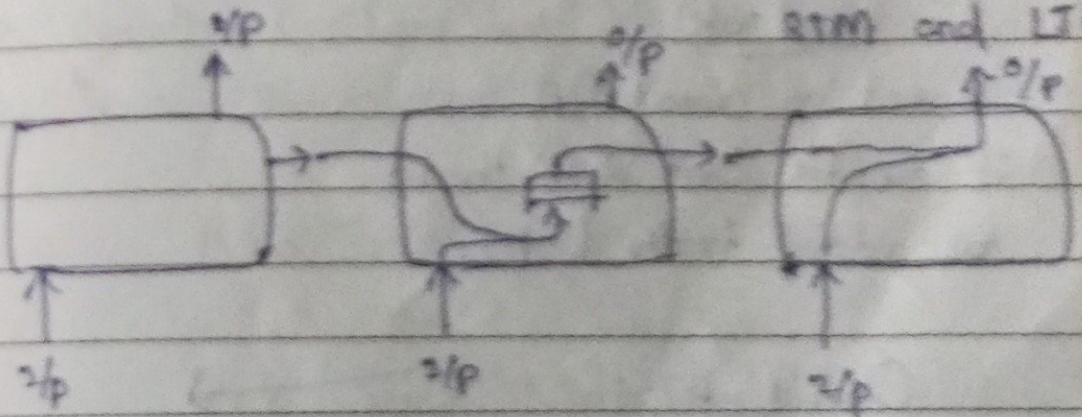
When we deriving a story perceptrons processes synd by word. that maintain two context.





If you find any datablock or perception that is important in long term then we move that data from STM to LTM.



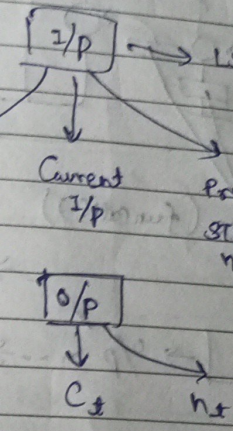


(RNN) architecture

Where we can find a Short Term memory is there which is there to

How they are working →

During input we provide previous LTM of previous time step i.e. (C_{t-1}) and STM as (h_{t-1})



Processing :-

→ Updation of Long Term memory.

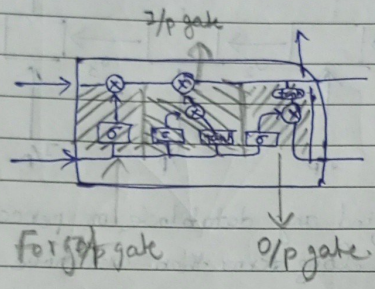
→ Create Short term memory.

Now I Get it →

↓

What is LSTM.

The architecture of lstm is called as gates. there are three gates in LSTM.



forgot gate → If decides based on the current Input.

It removes the data from the long term memory.

Input gate

Based on current I/P it decides what data should be added to the LTM.

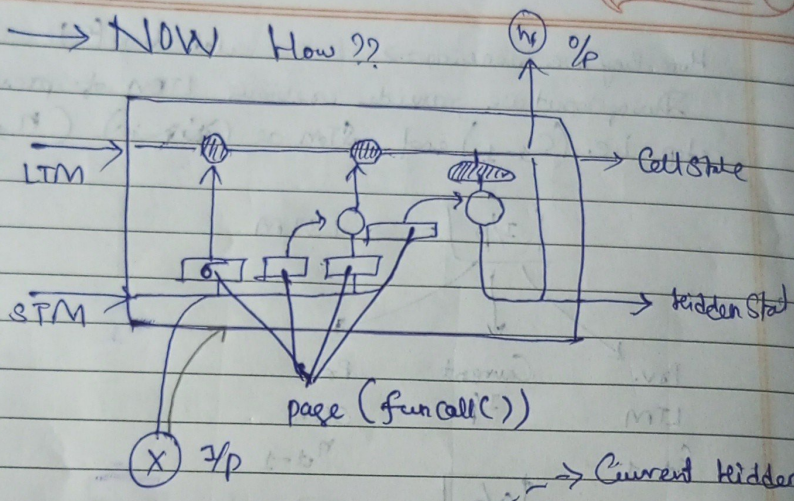
Output gate :- →

Based on current input it returns the data based on long term memory

It not only returns output but also create a short term memory.

fun call( )

→ NOW How ??



→ Current hidden state

I/p → PU → o/p → Current Cell State

I/p

Present hidden State

Present Cell State

I/p for the Current time step (x)

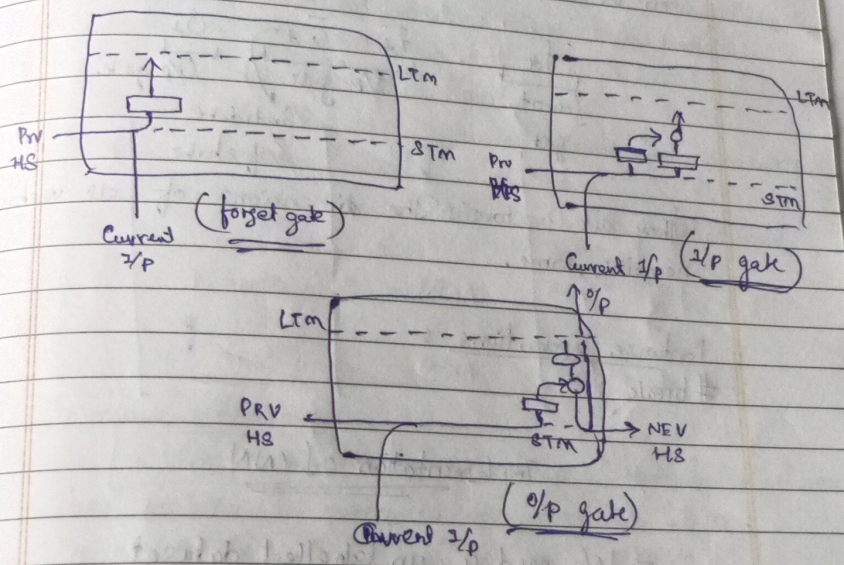
Here the PU, What it does
→ Update the LTM
→ Recalculate the hidden state / STM

While going from C_t to $\longrightarrow C_{t-1}$.

(i) Based on the current input X_t, we are going to decide which data is to remove..

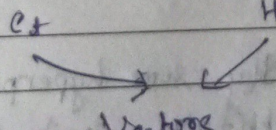
(ii) Based on the current input X_t, we are going to add the Necessary data.

The Gates of LSTM architecture is based on three gates in below there :-



forget gate used to remove something / datas from cell state. I/p gate is there to add somethink to the Cell State. O/p is used to calculate the Hidden state for the next time step.

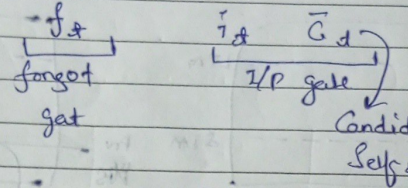
(Cell state) , (Hidden State)



Note :-

The dimensions of Ct and Ht will be same.

X_t is also vector / Set of number. It's dimensions depending on the data. Let's use more 4 gates



They are vectors. The dimensions of all vectors remains same.

Pointwise operations :-

#breaks ;

Implementation of CNN

→ We need a unlabelled dataset

→ We make a dataloader and fit the data to form a dataframe

→ Split the data, train ~~and~~ test

→ Convert train, test → traingenerator and testgenerator. These are the method which generates batch of data.

↑

Here we modify some hyperparameter ie image-size, batch size, specify the column

→ Creation of Model.
Model is created using Conv2D, maxpolling2D, Dropout, Flatten, Dense, activation, BatchNormalisation.
→ Training of the model.
→ using the train-generator and test-generator we train & validate our data.
→ Visualize the accuracy by training loss and accuracy plot.
→ testing the model.

Pointwise operations →

The pointwise operations are created in between two vectors. Its pointwise operators can be a

→ ⊕
⊗ ←
→ tanh

⊗: let a vector $C_{1\times3} = [4\ 5\ 6]$ 1×3
$f_{1\times3} = [1\ 2\ 3]$ 1×3

o/p

⊗ → [4 10 18]
⊕ → [5 7 9]

⊕ :
tan :

$[\tanh(X)\ \tanh(Y)\ \tanh(Z)]$

(Go to) funCall()

In that pase

σ → Represents a neural network layer and that layer containing number of Neurons / nodes and each of them containing with an activation function (Sigmoid activation function).

	TYPE	Activation
3, 4, 5 [σ] →	ANN	Sigmoid
3, 4, 5 [σ] →	ANN	Sigmoid
3, 4, 5 [tanh] →	ANN	tanh

↓

Contains Same No of Neurons } Let's welcomes a [Hyperparameter]

↓

i.e No. of Neuron.

→ user will provide it using UDA →

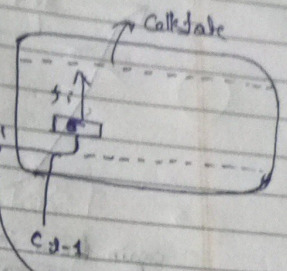
↓

user Defined Arguement

And the Hyperparameter will be universal one if there is a change occured in the 1st layer then it will effect all activation Neural Networks

The forgot gate :-

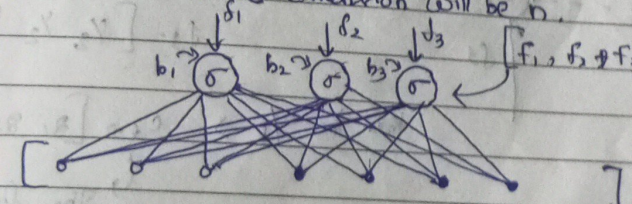
forgot gate is a container / get three i/p as arguements i.e → c_{t-1}, h_{t-1}, x_t



It give a o/p i.e will remove information from cell state.

Here [n dimension] of h_{t-1} and (c_{t-1}), where the no. of neurons will be n. i.e

the dim/product of h_{t-1}, c_{t-1} dependent on the number of neurons. if there are n number of neurons. Then the dimension will be n.



So there are 3 weights and each node, contains one biases, i.e 3 biases are there.

Here dim, $W_f = (7 \times 3)$

$h_{t-1} = (3 \times 7)$

$b_1 = (3 \times 3)$

$\delta_t = (3 \times 1)$

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

3×4 3×1 3×3

3×1

3×1

again there is a pointwise operator

C_{t-1} ⊗ f_t

$f_t \otimes C_{t-1}$

How this sign is really removing data from long term memory.

C_{t-1} ⊗ f_t

If we take $C_{t-1} = [4, 5, 6]$

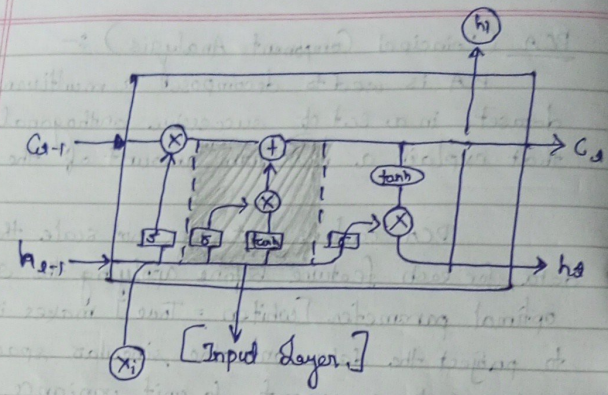
$f_t = [1/2, 1/2, 1/2]$

then the o/p → $C_t \Rightarrow [2, 2.5, 3]$

↳ The long term memory is decreased.

When We use [0 0 0] then the long term memory will be empty.

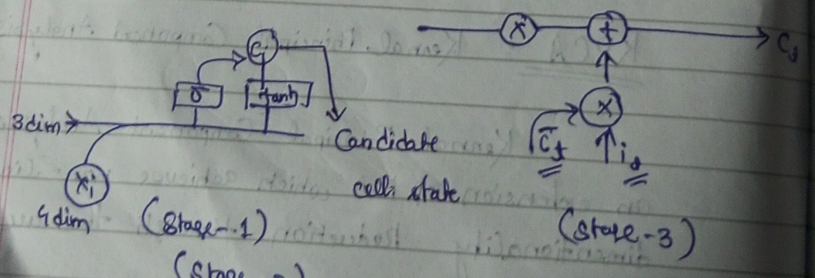
Contd..... (LSTM)

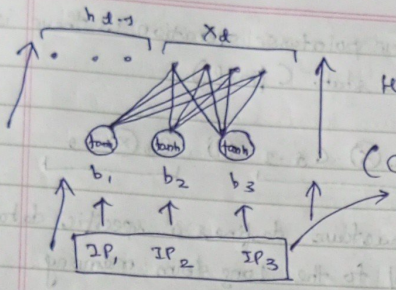


When we input data into the layer it check weather it's important or not. If it is then it will move to the cell state.

It's working under 3 stages :-

1) Candidate cell state ($\tilde{C}_t$)
2) (i_t) decides candidate cells which cell is / cell state is going to add to the Long term memory.
3) (C_t) find C_t to get / update the Long term context.

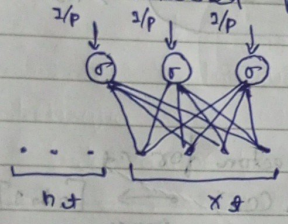




Here the weight matrix W_C (3×7) (Candidate cell state)

$$\bar{C}_t = \tanh\left(W_C \left[h_{t-1} . x_t\right] + b_C\right)$$

Now we find out the o/p then we will add the o/p. which is returned from this solution



$\rightarrow w[i]$

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

(3×1)

(3×7) (7×1)

(3×1) (3×1)

(3×1)

There is a pointwise operation in between $\bar{C}_t$ and i_t, see to the graph.

pointwise,

$i_t \otimes \bar{C}_t \rightarrow \bar{C}_t$ (filtered candidate cell state)

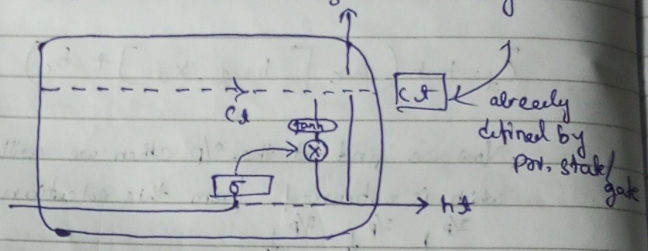
(3×1) (3×1) (3×1)

(3×1)

There is a another pointwise operation is there is to define the cell state (final)

$$C_t = \underbrace{f_t \otimes C_{t-1}} \oplus \underbrace{\bar{C}_t \otimes I_t}$$

Gating structure defines a specific data which need to be added to the Long term memory.



basic funda →

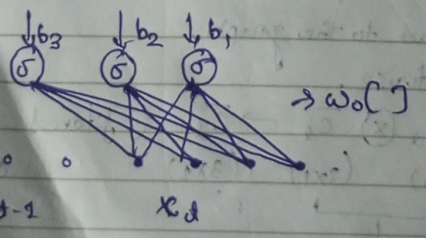
(i) Calculate O_t using

(ii) Regularise tanh vector for C_t

$\tanh(C_t)$ ⟶ range $[-1, 1]$

(iii) Pointwise operation between them that provides h_t

$\tanh(C_t)$



$$O_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_1\right)$$

$o_t = \sigma\left(w_o\,[h_{t-1}, x_t] + b_t\right)$

(3x1) (3x7) (7x1) (3x1) (3x1)

$\sigma(3\times1) = 3\times1$

to find h_t → we need $[o_t \otimes \tanh(c_t)]$

(3x1) (3x1) → (3x1)

then h_t containing (3x1) / h_t value is found by us and we find that h_t containing (3x1) matrix.

(LSTM ARCHITECTURE) ↩

THE OVERALL THEORY BEHIND THIS

NOW LET'S MOVE TO TRANSFORMERS

↓

$$o_t = \sigma\left(w_o\,[h_{t-1}, x_t] + b_t\right)$$

(3×1)

$3\times7 \quad 7\times1$

$3\times1 \qquad 3\times1$

$\sigma(3\times1) = 3\times1$

to find h_t → we need $[o_t \otimes \tanh(c_t)]$

$(3\times1) \quad (3\times1)$

(3×1)

then h_t containing (3×1) / h_t value is found by us and we find that h_t containing (3×1) matrix.

(LSTM ARCHITECTURE)

THE OVERALL THEORY BEHIND THIS