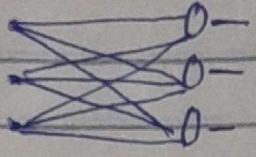
Long Short-Term Memory (LSTM):

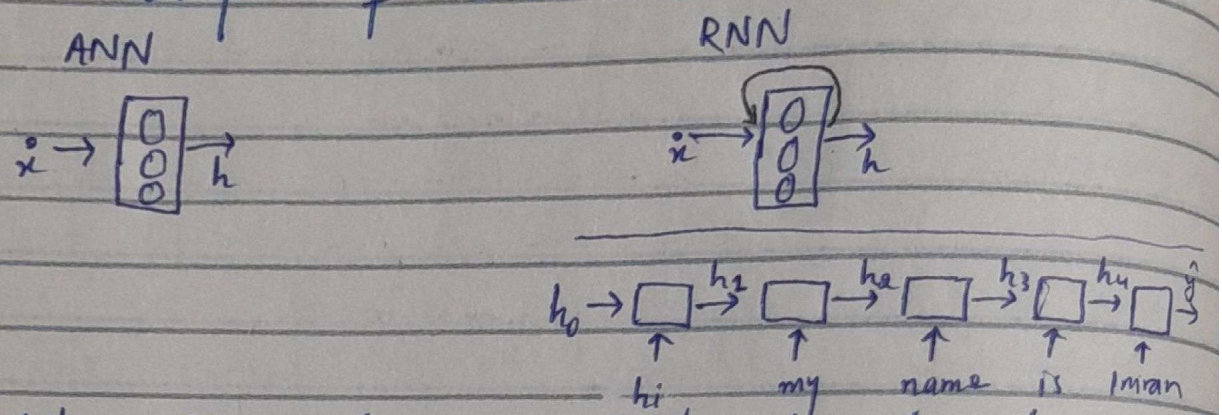
ANN:



Problem: Not working with sequential data.

RNN:

Use similar architecture to ANN but introduced a concept of state.



* When the chain in RNN gets too long, the prediction at the end forgetten initial inputs and that is why initial inputs have not much effect on far away output.

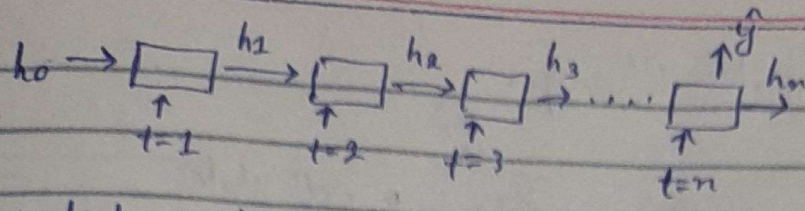
LSTM was developed because RNN was not able to handle long sequences because of vanishing and exploding gradients.

LSTM Core Idea:

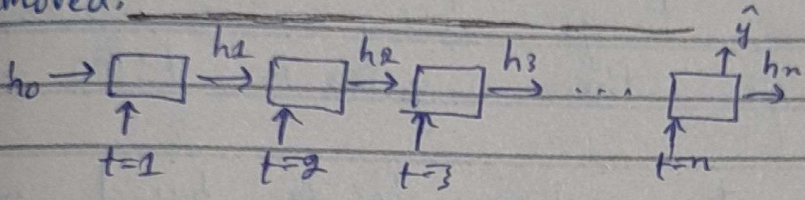
→ In RNN, there is only one path, which is used to retain information about past data.

→ This single line have the responsibility of maintaining both long term and short term context.

→ Mathematically, not possible to put both contexts on this single line/variable. (short-term context dominate)



→ Then scientists decide what if we maintain another extra path.
→ One for keeping short-term memory.
→ Another for keeping long-term memory.
→ Now if something is consider important at the initial steps, it will reach to end until not removed.

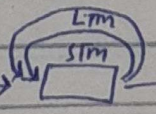


→ In LSTM, a separate path is introduced for long term memory other than short term path.

RNN: input → output

LSTM: It is much similar to RNN. There are two major differences.

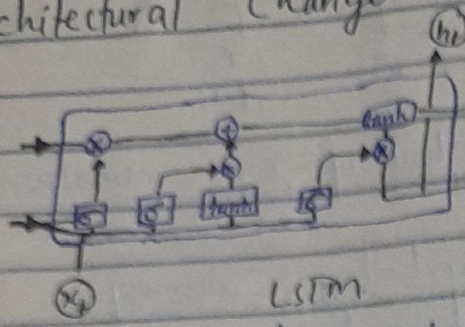
1. Taking two states:



* In LSTM context it is called cell state.

2. Architecture:
   → RNN cell have a simple architecture.
   → LSTM cell have a complex architecture.
      ↳ Because here we keep both short-term and long-term memory.
      ↳ Also we have to do communication between them.

Architectural Changes:



→ The complex design in LSTM box is because of, to enable communication between long-term and short-term memory.
→ The inner things are called gates in LSTM.
→ There are three gates in LSTM

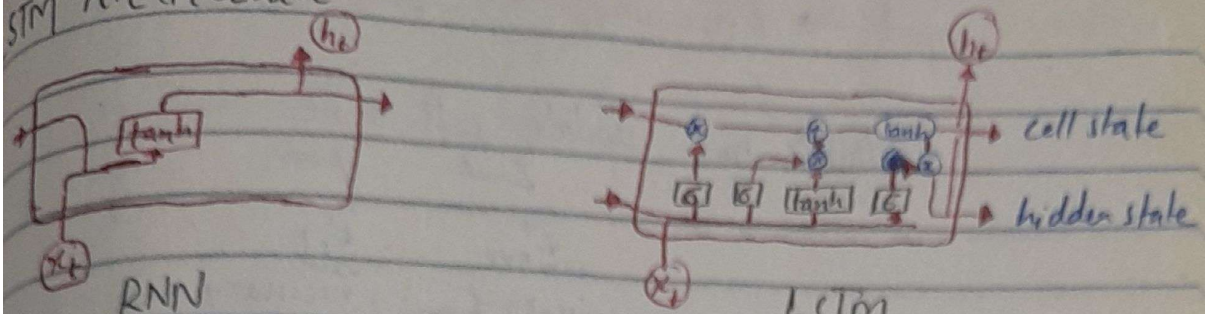
1) Forget Gate:
The gate which decide based on the current input and short term context, what should be remain and what should be remored from long-term memory.

2) Input Gate:
The gate which decides based on the current input what should be added to long term memory.

3) Output Gate:
The gate which decides based on the current input what should be extract from long-term memory to show as output.

Output gate: → Give output at the end.
→ At every given stage, create short term memory.

LSTM Architecture:



RNN LSTM

LSTM maintains two types of context.
1) LTM 2) STM
In a technical language LTM called cell state and STM called hidden state.

input → [processing] → output

Input: (takes three things)	Output:
1) Previous Cell state	1) Current hidden state
2) Previous hidden state	2) Current cell state
3) Input for current timestep (x_t)	

→ Other work than keeping LTM & STM, LSTM cell also do communication between both.

Processing
Two works done in processing:
1) Update cell state [$c_{t_0} \longrightarrow c_{t_1}$]
2) Calculate h_t [hidden state/STM]

$c_{t_0} \rightarrow c_{t_1}$
To reach c_{t_1} from c_{t_0}, two works done in it.
i) Based on the current input (x_t), decide what to remove from LTM (remove unnecessary info).
ii) Then again on the basis of x_t, decide what to add new. (Add necessary)

input → Processing → output

calculate h_t

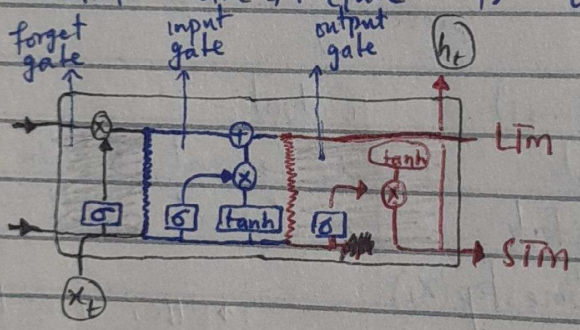
update cell state.

based x_t → remove unnecessary from LTM.

based x_t → add necessary to LTM.

Gates in LSTM:

- To update cell state and calculate hidden state h_t, LSTM uses gates concept.
- The LSTM architecture is divided into three gates

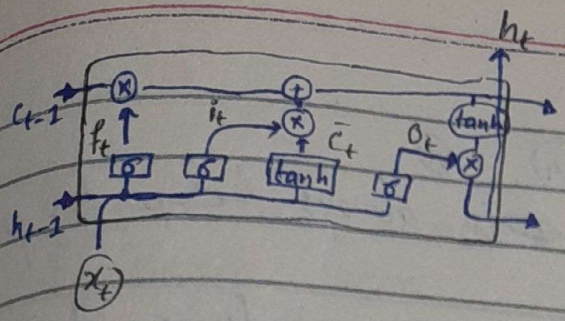


Forget gate
- to remove something from cellstate c_t

Input gate
- to add something to the cellstate c_t

Output gate
- to calculate hidden state h_t.

Update Cell state (c_t)



What are c_t and h_t ?

c_t, h_t both are vectors.

[0.1 0.3 0.5]

$\therefore c_t$ and h_t dimensions will be same.

∴ mathematically, how things look inside LSTM?

x_t : It is also a vector

→ we can calculate x_t.

→ So cat mat rat would be like [100] [010] [001].

→ x_t have no restriction of dimension.

cat mat rat	0
cat rat rat	0
mat mat cat	1

→ Let's use one-hot encoding

→ Vocabulary: cat, mat, rat

→ So represent

1 0 0 → cat

0 1 0 → mat

0 0 1 → rat

f_t : inside forget gate

i_t : input gate

$\bar{c}_t$: input gate (also called candidate cell state)

o_t : output gate.

Mathematically, all of them are vectors, interestingly f_t, i_t, $\bar{c}_t$, o_t dimensions are exactly same to c_t and h_t.

Pointwise Operations:

⊗ ⊕ (tanh) ⊗

→ Pointwise operations is always between two points

⊗ Pointwise Multiplication:

$C_{t-1} \otimes f_t$

$C_{t-1} = [4\ 5\ 6]$

$f_t = [1\ 2\ 3]$

$= [4\ 10\ 18]$

⊕ Pointwise addition:

$C_{t-1} \oplus f_t$

$C_{t-1} = [4\ 5\ 6]$

$f_t = [1\ 2\ 3]$

$= [5\ 7\ 9]$

(tanh) Pointwise tanh:

$C_{t-1} = [4\ 5\ 6]$

find tanh(4), tanh(5) and then tanh(6)

→ Pointwise operation will always be between two vectors, their dimensions must be same. Point wise operation output will always be a vector.

Neural Network Layers:

[σ]

→ A neural network layer.

→ There are multiple nodes.

→ Every node will have a sigmoid activation function

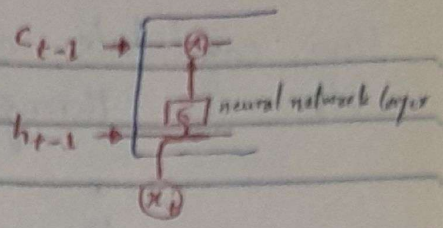
[tanh]

→ A neural network layer.

→ Every neuron will have tanh activation function

* In each particular [σ], how much number of units or nodes will be? is flexible. This is like hyperparameter, you have to decide. It will be same in all (once you decide)

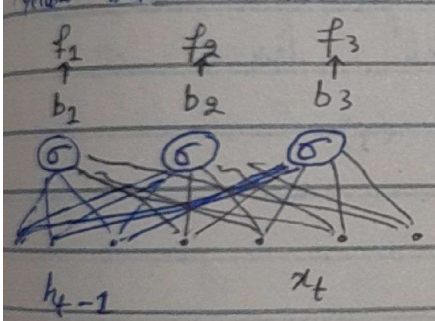
Forget Gate:



let say we have 3 units in a neural network layer.
- num of units : 3
- x_t : 4 dim vector $[x_{i1}\ x_{i2}\ x_{i3}\ x_{i4}]$
- h_{t-1} and c_{t-1} are also vectors. Their dimension is (equal to number of nodes in your neural network layer)
- In forget gate, work will done in two stages.

→ Forget gate take three things in input: x_t, h_{t-1}, c_{t-1}
→ Output: remove from cell state.

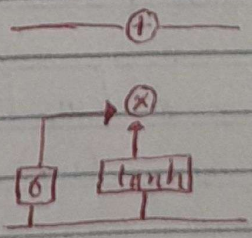
1) Calculate f_t (yellow box looks like)



$(3\times1)+(3\times1) = (3+1)$
3×7 (21 weights)
7×1 (4 Biases)

2) $c_{t-1} \otimes f_t$ (Pointwise multiplication)

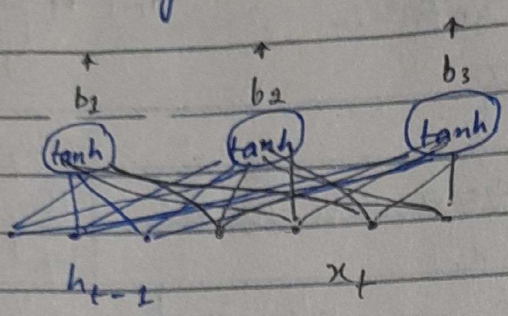
Input Gate:



The reason to use input gate is to add some new important information to the cell state.
Input gate work in three stages:
1) Calculate $\bar{c}_t$ (candidate cell state)
2) i_t (calculate i_t)
3) c_t (calculate c_t)
c_t is our current cell state.

→ it decides candidate cell state which will be add in cell state.

How candidate cell state calculated?
→ It is through neural network.



$$\bar{c}_t = \tanh\left(w_c\,[h_{t-1}, x_t] + b_c\right)$$

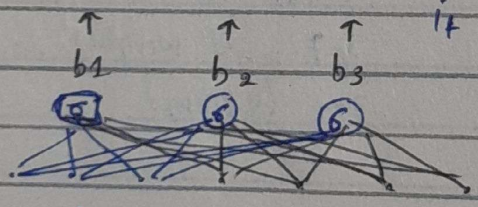
(3×7) (7×1) (3×1)

(3×1)

(3×1)

→ Candidate cell state is potential important information.

i_t is calculated with other path, here input is same but neural network have σ



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

(3×7) (7×1)

3×1 (3×1)

$\sigma(3\times1)$ → (3×1)

i_t is like a filter, it decides that what info from candidate cell state will go in cell state.

i_t filter $\bar{c}_t \rightarrow c_t$

Now we have i_t and c_t

$i_t \otimes \bar{c}_t \rightarrow \bar{c}_t'$ (Filtered candidate) Cell state

(3×1) (3×1)

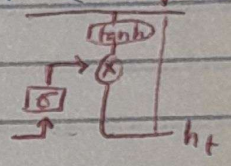
[0.5 0.5 0.3] [4 5 6]

$c_t = f_t \otimes c_{t-1} \oplus i_t \otimes \bar{c}_t$

Output Gate:

For current timestep decide output (hidden state)

$c_t \rightarrow h_t$

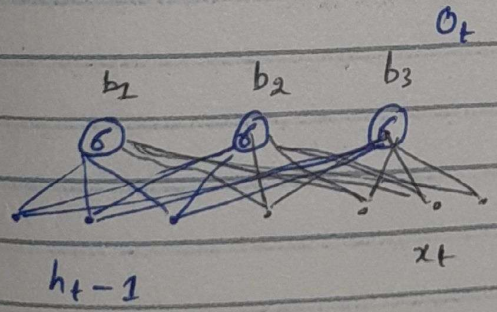


Two steps:

1) tanh(c_t)
2) You do filteration on it. (need O_t for it)

$h_t = [O_t \otimes \tanh(c_t)]$

O_t calculate through neural network layer



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

(3×7) (7×1) (3×1) (3×1) (3×1)