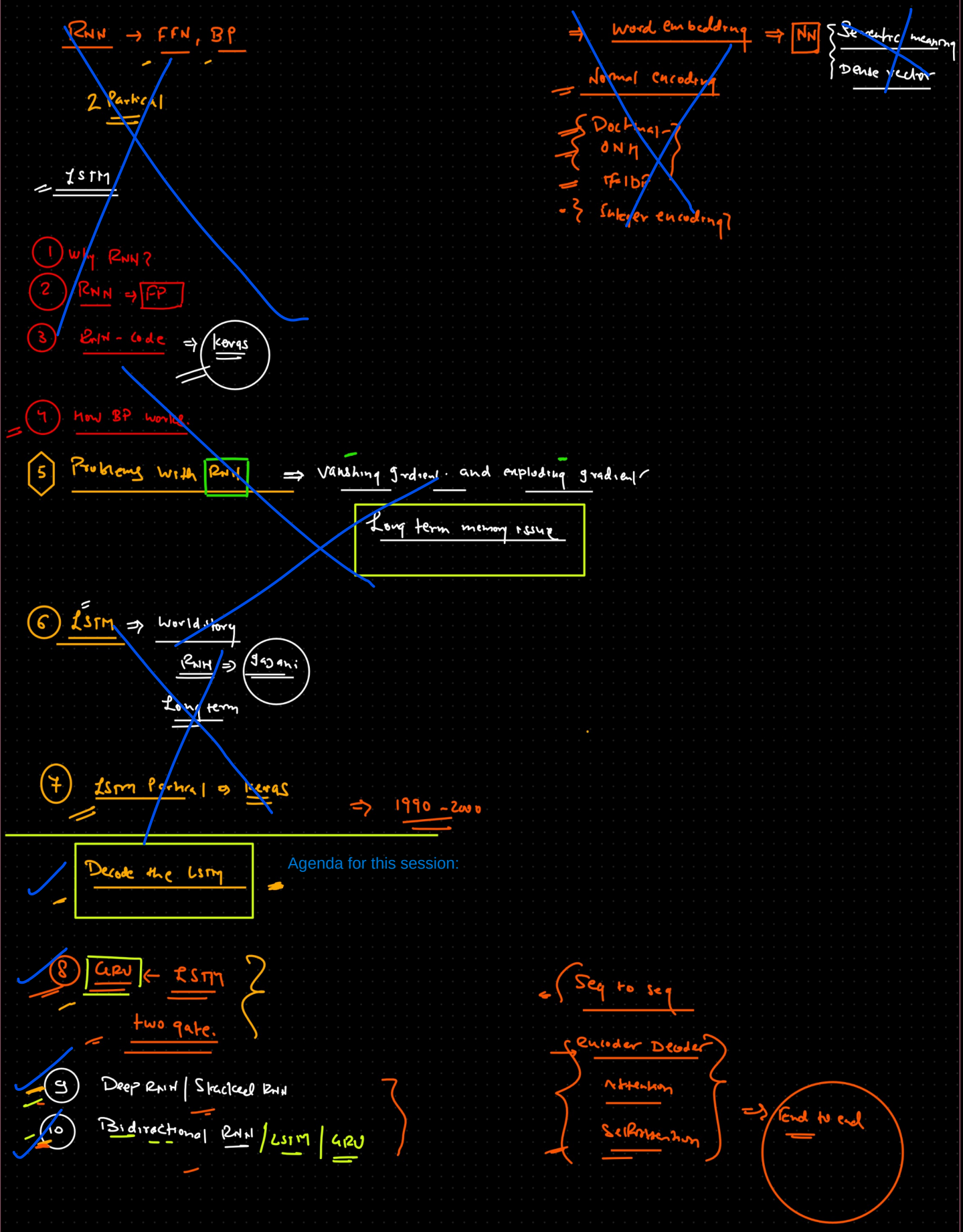
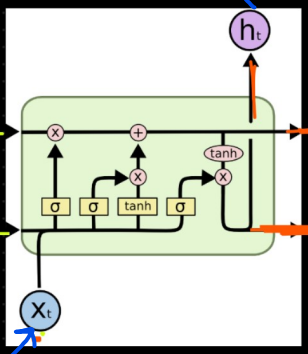


Please note that in the practical we saw RNN and LSTM implementation using keras

⇒ LSTM

Decoding the LSTM architecture in depth:



Output from the hidden state at timestamp t

Forget, update, output

Cell state(memory) from previous timestamp i.e 0

C_{t-0}

Output from hidden state from previous timestamp i.e 0

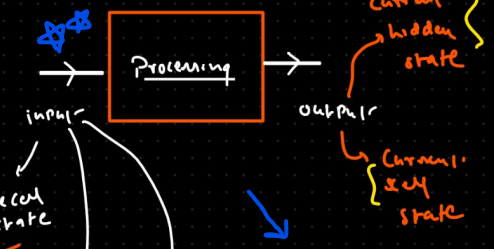
h_{t-0}

C_t

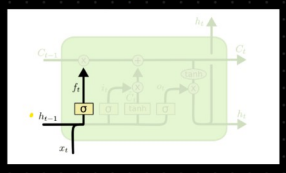
h_t

Input at t timestamp

3 things involve



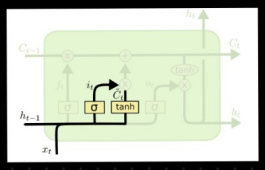
- The LSTM cell does3 processing Forget, Update and Output the cell states by using 3 gates.
- It takes 3 input: Pre cell state, Pre hidden state and input from the current timestamp.
- It generates the 3 output: current hidden state, current cell state and third one is again a hidden state which is passed to the next LSTM cell as an input. Please note this 3rd output is utilized only if there exists another hidden state after the current cell.



Forget gate

to remove something from the C_t

Cell state at t timestamp



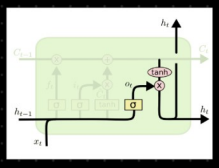
Input gate

↑ add/update in the cell state

To add/update the Ct

C_t

Cell state at t timestamp



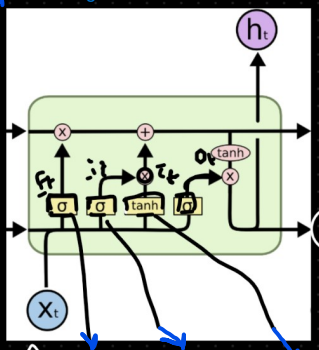
Output gate

Calculate the hidden

$\{C_t / h_t\}$

Cell state and hidden state at t timestamp

Decoding the LSTM cell further:



⊗ ⇒ Point wise multiplication

⊕ ⇒ Point wise addition

tanh ⇒ (act ⇒ cell)

NN (Hidden) (Hidden) Nodes

Hidden tanh act

Hidden neural network with tanh activation fun^ that can have as many neuron as required

- Represents the hidden neural network where sigma represents the sigmoid activation fun^(may be)
- Also, please note that the output from the first hidden NN is represented by Ft (where t represents the current timestamp) and output of second hidden NN is represented by it (Where t represents the current ts)
- Similarly, output of 3rd hidden NN is represented by $\tilde{C}_t$
- and for 4th hidden NN by O_t

Hidden vector

h_t

C_t

Vector

[0.1 0.2 0.3]

C_t Cell

h_t → Hidden

Same

Important note on Ct and ht. Both of these are vector representation. Also, the size of both the vectors will be same.

X_t ⇒ Input on the current TS which is also encoded vector representation of the textual data or any other sequence based data

Vec

Sentiment

OHE [good bad better worst]

good bad better	0
bad bad better	1
worst bad good	0

([1,0,0,0] [0,1,0,0] [0,0,1,0]) ⇒ OHE

X_t → X_{t+1} → X_{t+2}

word by word → Lstm

f_t → forget gate

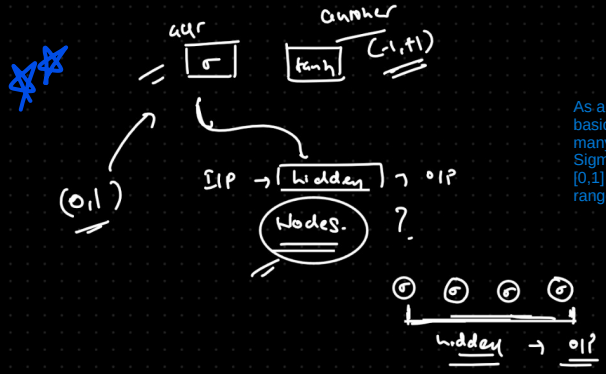
$i_t, \tilde{C}_t$ ⇒ input gate

O_t ⇒ O/P gate

vector

All these are vector. Remember whatever information is flowing inside the architecture are vector representation

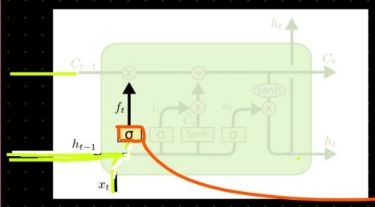
I/P ⇒ architecture → fun^ → vector



As already discussed, each small boxes represents the hidden neural network where basically WEIGHTED SUM is happening. These hidden neural network can have as many neurons as needed as this is the hyperparameter.
Sigma represents the sigmoid activation fun^ which will compress the data in the range [0,1] whereas, tanh is the activation function that is going to compress the data in the range [-1,+1]

Decoding the Forget gate further:

Forget gate



Read this in bottom to top approach

Assumption 2:
Consider that hidden NN specific to forget gate has 3 nodes/neurons

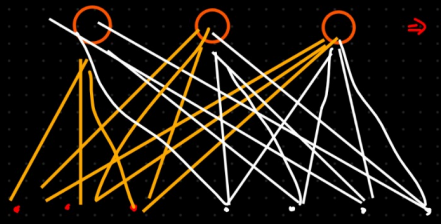
Node = 3

$[F_1\ F_2\ F_3] \Rightarrow F$

Weighted sum of this fully connected NN will give a forget vector represented as F

(3×1)

3 biases $(b_1\ b_2\ b_3)$



⇒ 3 As discussed above since there are 3 nodes in the hidden NN for forget gate we have 3 nodes here

7×3 = 21 weights + 3 bias ⇒ 24

Total number of trainable parameters

Here we have 3 nodes representing ht-1 → h_{t-1}

Here we have 4 nodes representing xt → x_t ⇒ (7×1)

4 dimension

$[x_{i1}\ x_{i2}\ x_{i3}\ x_{i4}]$

Assumption 1:
Here consider that input at current time stamp xt contains 4 words, hence dimension will be 4.
Whereas, Output from hidden state in previous time stamp ht-1 contains 3 words, hence dimension will be 3

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

Equation of forget gate vector. Where Ft represents the forget gate vector at t timestamp Wf represents the weight of forget gate, ht-1 represents the hidden state output at timestamp t-1 , xt represents the input at timestamp t and bf represents the bias for the forget gate. This weighed sum is passed through the activation function represented by sigma

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

Understanding how forget gate at t time stamp (Ft) helps in removing the context being held in the previous state which is represented by cell state at t-1 timestamp (ct-1).
>>> This forget operation is implemented through point wise multiplication b/w Ft and Ct-1

$F_t \otimes C_{t-1}$

cell

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

$\otimes \Rightarrow$

$[\frac{1}{2}\ \frac{1}{2}\ \frac{1}{2}]$ Ft

$[0\ 0\ 0]$ ht

$= [2, 2.5, 3]$

$[0\ 0\ 0]$

100% remove

Case where forget gate vector is a identity vector then in that case there will be no removal of the context being held by cell state in previous timestamp

$[1\ 1\ 1] \Rightarrow [4\ 5\ 6]$

remaining

updating

Observe this vector is pointwise multiplication outcome of Ft=[1/2,1/2,1/2] and Ct-1=[4,5,6] which we can clearly observe the loss in the context

Case where forget gate vector is neither identity nor zero vector then in that case there will be some removal of the context being held by cell state in previous timestamp after applying pointwise multiplication

Case where forget gate vector is a 0 vector then in that case there will be 100% removal of the context being held by cell state in previous timestamp

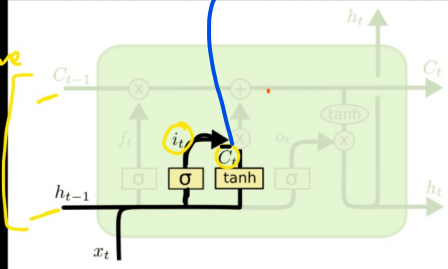
Decoding the Input gate

Input gate

C_t represents the partial cell state for the current timestamp

Please note that both it and Cbar t are vectors of same size. Similarly Ct-1 and ht-1 represents the vector of same size.

Same



In Input gate we have 2 equations:

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

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

Try understanding these equations by observing the figure

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

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

⇒ Pointwise multiplication

Firstly, we perform pointwise multiplication of these 2 equations

$\bar{C}_t$ = 3×1 [· , · , ·] T = tanh

b_1 b_2 b_3

T T T

3×1 4×1 (7×1)

7×3 = 21 + 3 = 24

$x_t = 4$

$h_{t-1} = 3$

$$i_t \otimes \tilde{C}_t = \bar{C}_t$$

Secondly, we will be performing the point wise addition operation b/w forget gate operation and input gate operation, whose outcome represents the cell state in t timestamp

$$C_t = f_t \otimes C_{t-1} \oplus (i_t \otimes \tilde{C}_t) \Rightarrow \text{Current cell state } (t)$$

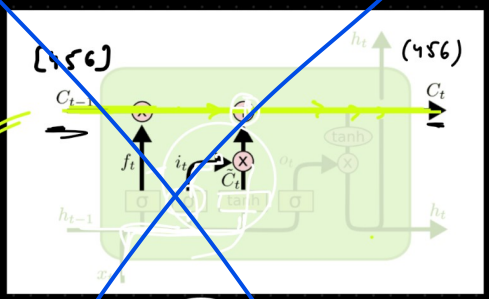
forget gate operation — Input gate operation

To conclude unlike RNN, in LSTM we are able to forget the past context(fully or partially) using forget gate and add any new context to the cell state using input gate. This is how LSTM is able to hold long term context which was missing in the case of RNN.

RNN

[4 5 6]

(4 5 6)



( )

Recall

(1, 2, 3, 4)

C_{t-1} = [4 5 6] ⊗ [4 5 6]

f_t = [1 1 1]

$i_t \otimes \tilde{C}_t$ = [0 0 0]

C_t = (4 5 6) ⊕ [0 0 0]

= [4 5 6]

LSTM

Update

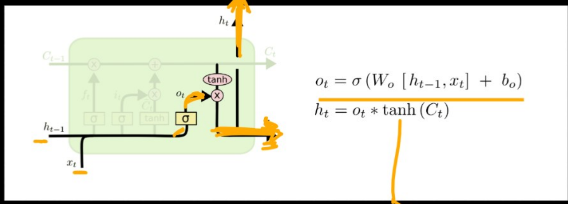
[t-1 t]

Decoding the Output gate:

Output gate

next hidden — Stacked RNN

cell state



Here Ct denotes the cell state at t timestamp that we have updated in Input gate(Check Ct equation in input gate as discussed earlier for the better understanding)

σ ⇒ hidden layer

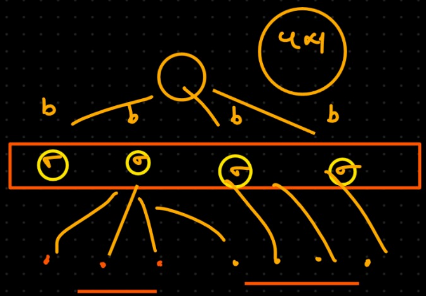
$$h_t = o_t \otimes \tanh(C_t)$$

This is the final equation of this LSTM cell denoted by ht which we are going to pass to the next LSTM cell as an input

x_t , h_{t-1}

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

(3×1) (4×1)



3×1 4×1 7×1

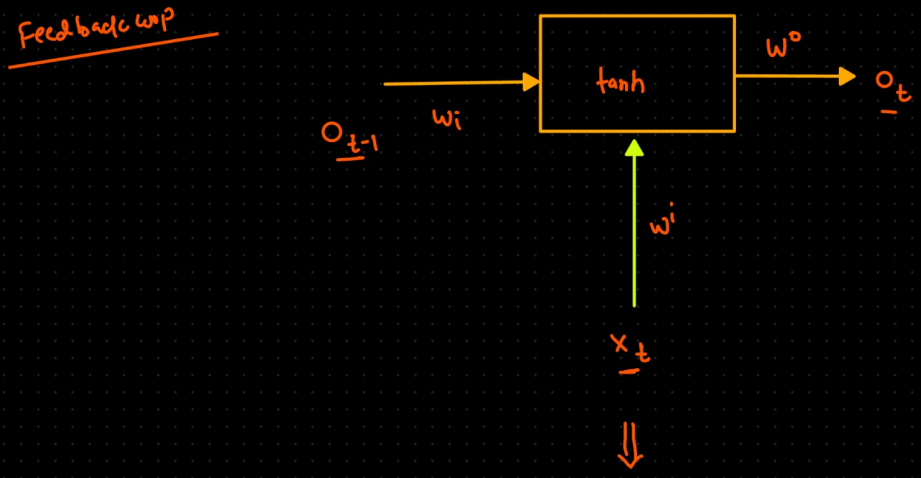
4

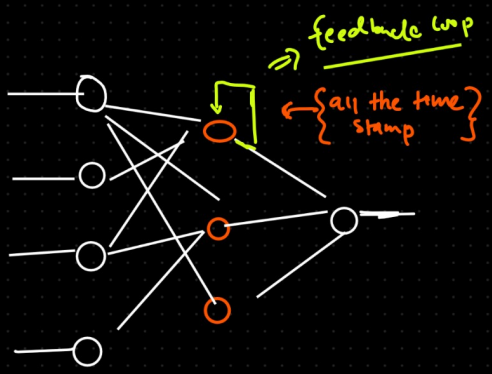
$7 \times 4 = 28 + 4 \Rightarrow 32$

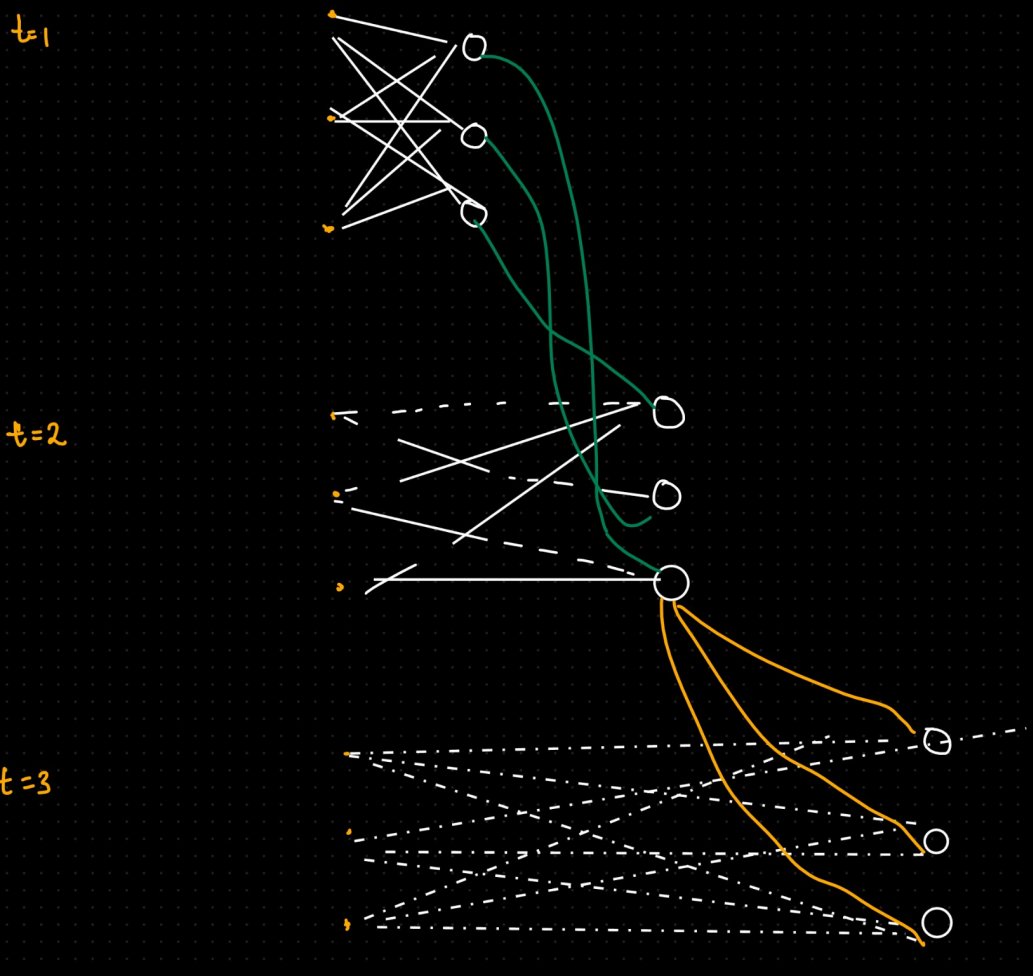
GRU ⇒ 2 gate

Deep RNN / Deep LSTM / Deep GRU

Understanding the basic RNN architecture:
Already discussed in the earlier session!



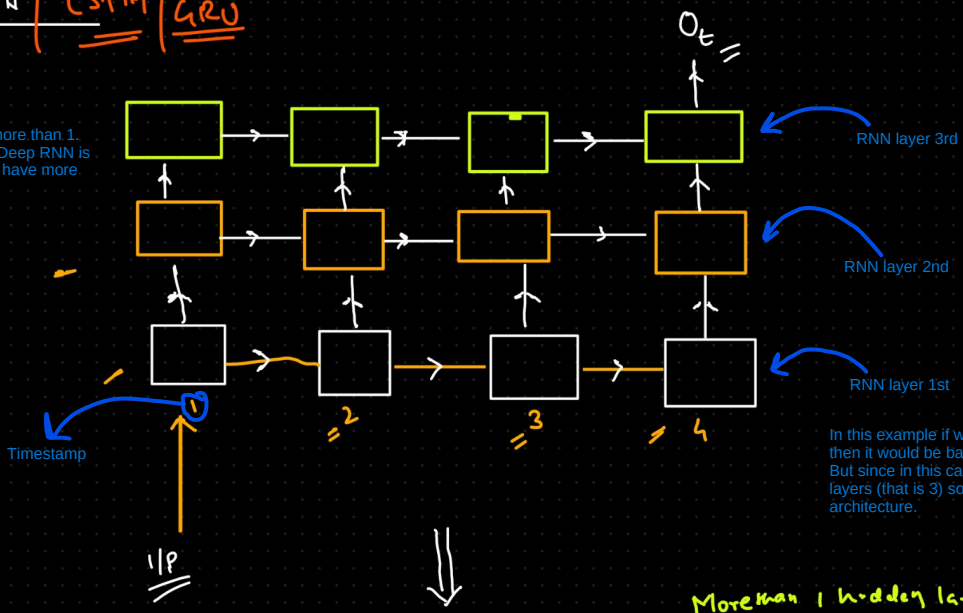




Deep RNN | LSTM | GRU

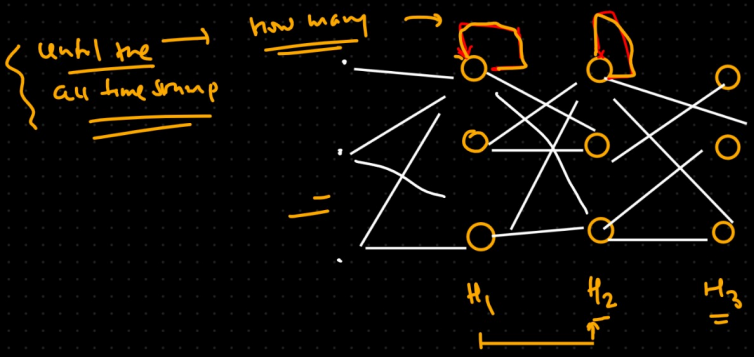
Hidden.

Deep means hidden layer more than 1. Therefore, we can say that Deep RNN is the RNN network where we have more than 1 hidden RNN layers.



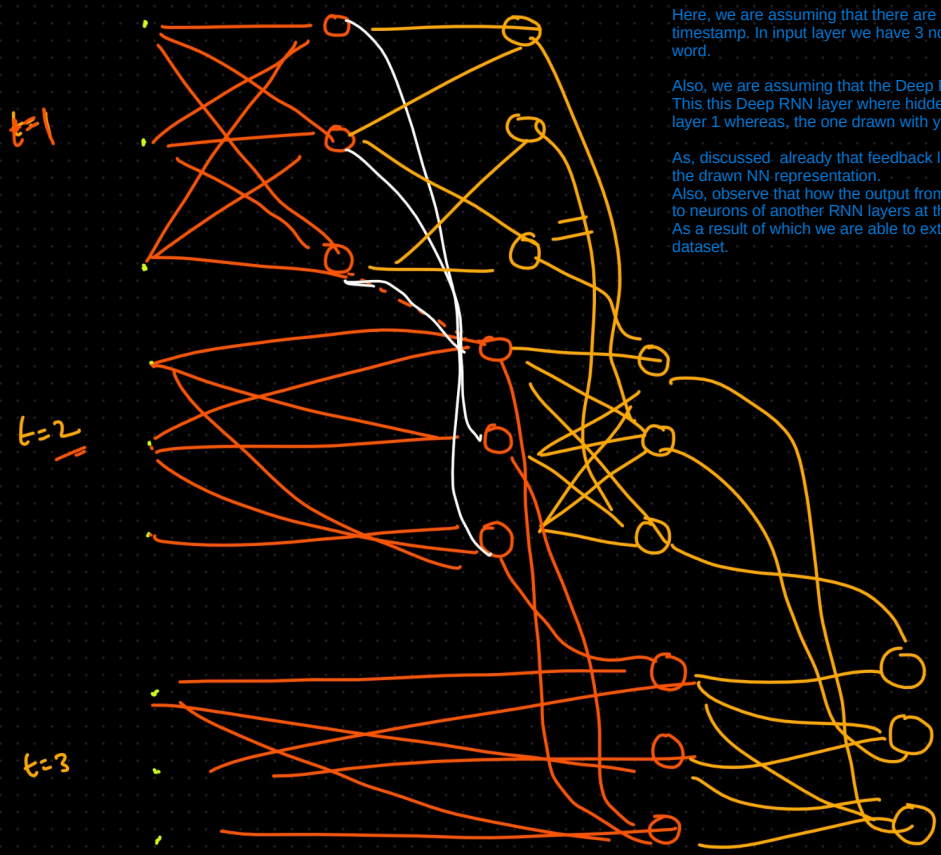
In this example if we were having only 1 RNN layer then it would be basic RNN architecture.
But since in this case we are having multiple RNN layers (that is 3) so this is considered as Deep RNN architecture.

Moreman 1 hidden layer



Please note that here feedback loop will be passed to its own hidden layer. Meaning for layer 1 a feedback loop through a neuron will be passed to all the other neurons in the same hidden RNN layer 1 and not to neurons residing in the layer 2 or layer 3. Similar segregation happens in layer 2 and layer 3 as well.

Neural Network representation of Deep RNN:



Here, we are assuming that there are only 3 words in the sentence hence, we are provided with 3 timestamp. In input layer we have 3 nodes that will be capturing the vector representing of each word.

Also, we are assuming that the Deep RNN architecture consists of 2 RNN layer.
This this Deep RNN layer where hidden nodes/neurons drawn using orange represents the RNN layer 1 whereas, the one drawn with yellow represents RNN Layer 2.

As, discussed already that feedback loop is passed to its own hidden layer. Observe the same in the drawn NN representation.
Also, observe that how the output from nodes of hidden RNN layers at each timestamp is passed to neurons of another RNN layers at the same timestamp. This chain goes on for each timestamp. As a result of which we are able to extract more deeper context from the passed input sequential dataset.

Please note that here we are using RNN cell. If we have used LSTM cell then it would be called Deep LSTM. Similarly, if we have used GRU cell then it would be called Deep GRU.

Why Deep RNN/Deep LSTM/Deep GRU:

Why RNN / LSTM / GRU ?

The main reason to basically use mentioned deep architectures is to increase the number of hidden layers. As we increase the number of hidden layers we are able to capture or retrieve much more complex patterns from the Input feed.

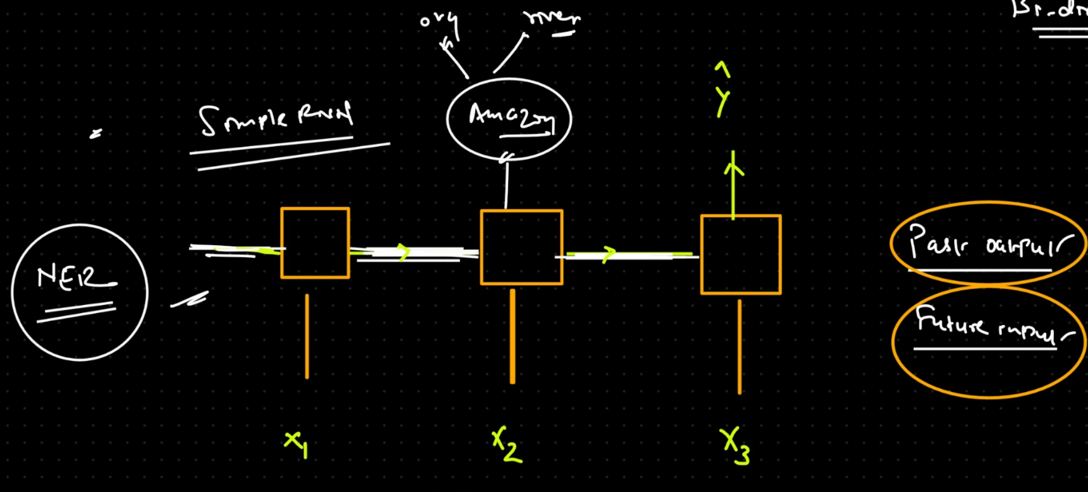
One such use case in when we are working with sequential data having some hierarchy like hierarchical text classification which focuses on classifying one text into multiple labels, which are organized as a hierarchical taxonomy.

Another such use case is the Speech recognition.

Stacked RNN/LSTM/GRU are same where also our main motive is to increase the hidden layers to capture complex patterns

Bidirectional RNN / Bidirectional LSTM / Bidirection GRU

Bi-dire



Single direction

Bi directional RNN/LSTM/GRU is applicable in the case where Past Output is affected by Future Input or vice versa. Or in other words we need to traverse in both forward and backward directions in order to correctly determine the context of the word/words in the text or any sequence based data.

Note that RNN/LSTM/GRU structures that we have seen until now are simple or stacked RNN which are uni-directional where the information is going to flow in only one direction. Whereas, in Bi-directional architecture information is going to flow in both forward and backward direction.

Let's Understand this Bidirectional concept using NER use case which stands for the named entity recognition.

NER:

Named-entity recognition is a subtask of information extraction that seeks to locate and classify named entities mentioned in unstructured text into pre-defined categories such as person names, organizations, locations, medical codes, time expressions, quantities, monetary values, percentages, etc.

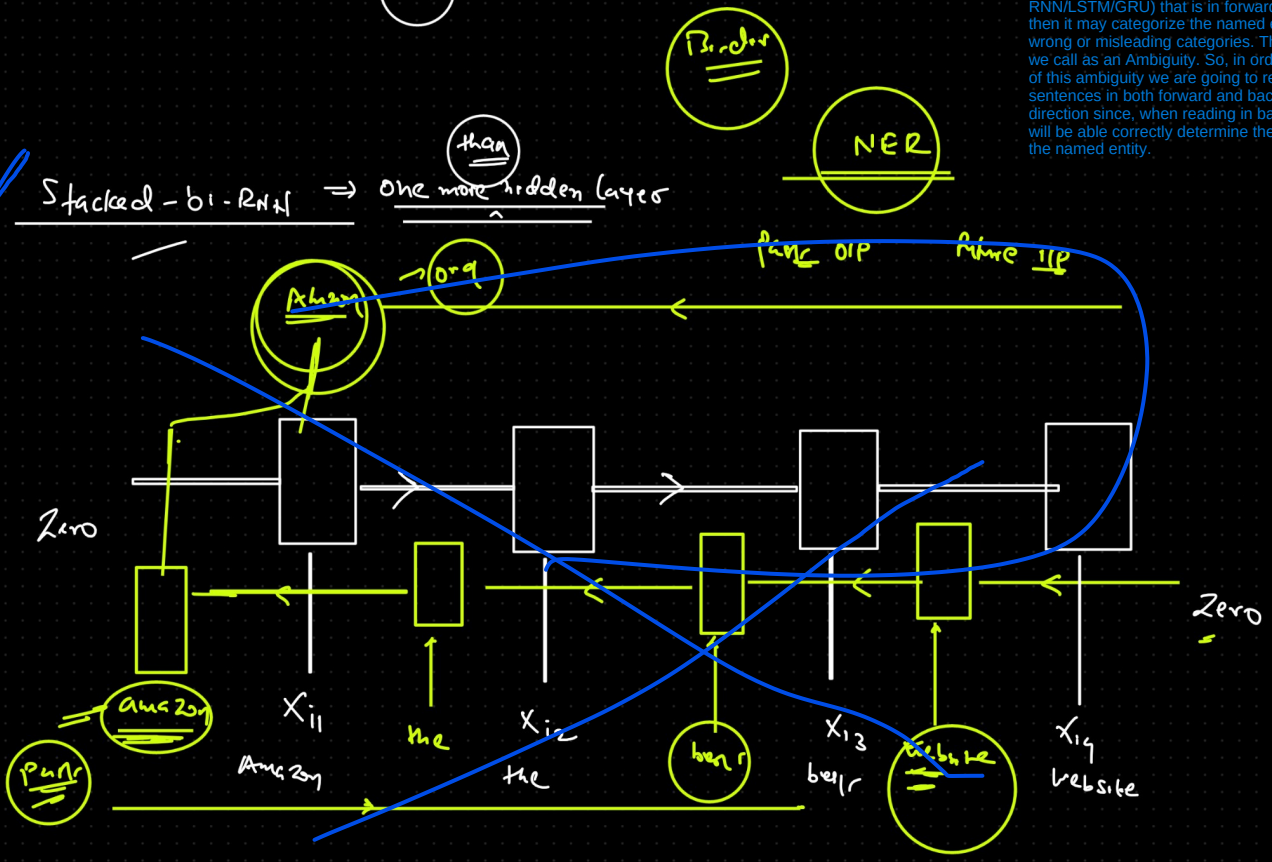
NER ⇒ Named entity Recognition

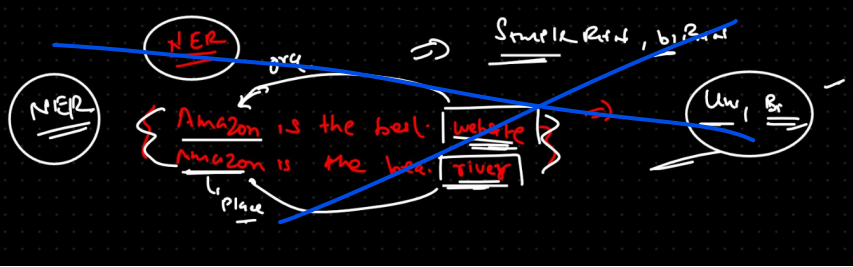
Named entity → Recognized category

Sunny → Person

Bengaluru → Place

In these 2 sentences the context of named entity "amazon" (Past Input) is dependent upon the info or context of the named entity mentioned at the end of the sentence (Future Output). In First sentence "amazon" is recognized as Organization whereas in the Second sentence the same word is recognized as River or Place. If we are going to process these sentence only in one direction/uni-direction (Simple RNN/LSTM/GRU) that is in forward direction then it may categorize the named entity into wrong or misleading categories. This is what we call as an Ambiguity. So, in order to get rid of this ambiguity we are going to read these sentences in both forward and backward direction since, when reading in backward we will be able correctly determine the context of the named entity.

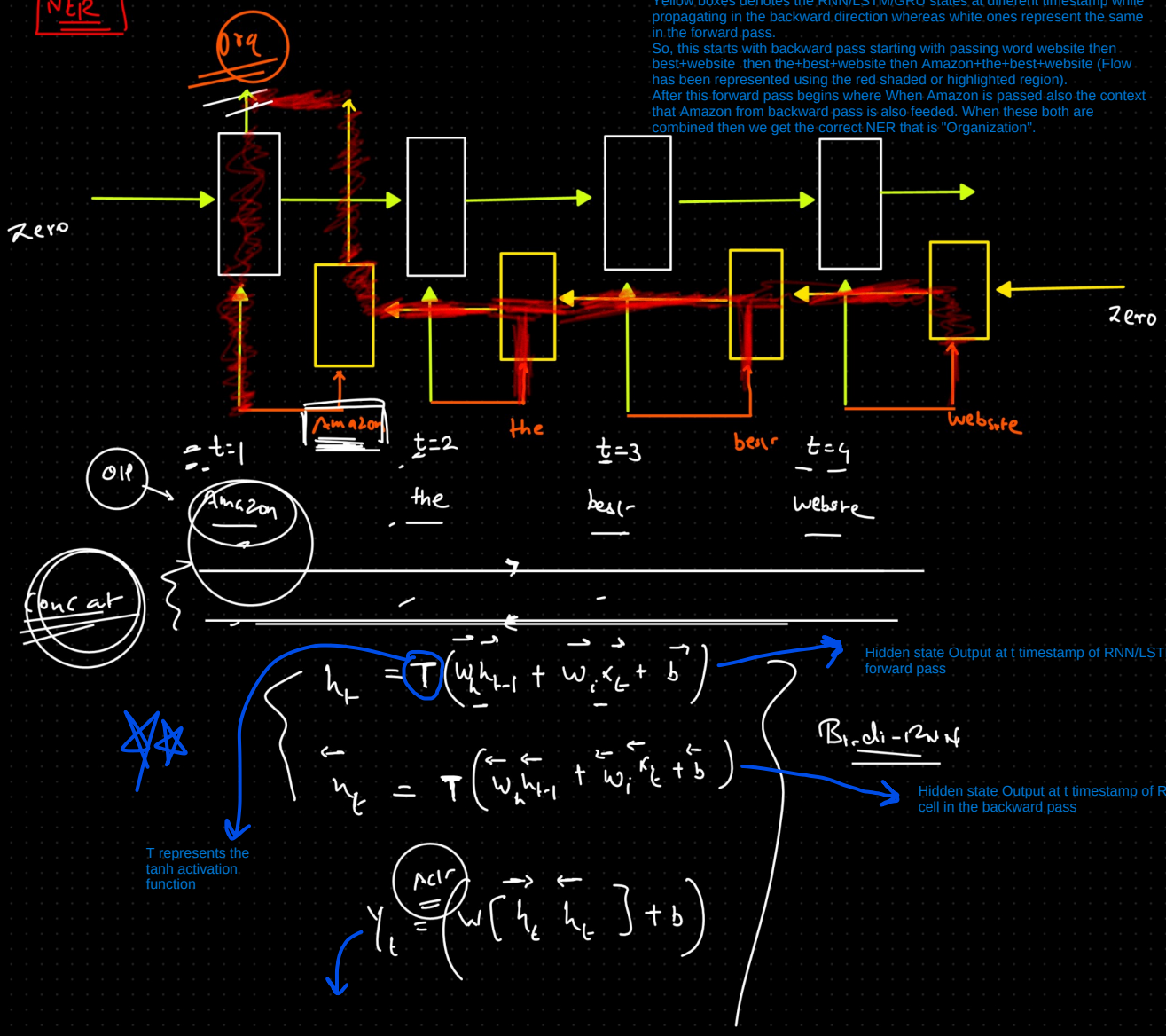




Understanding the Bi-directional architecture using an example:
Sentence: Amazon is the best website.
Problem statement is to correctly classify the category of named entity "Amazon"

NER

Yellow boxes denotes the RNN/LSTM/GRU states at different timestamp while propagating in the backward direction whereas white ones represent the same in the forward pass.
So, this starts with backward pass starting with passing word website then best+website then the+best+website then Amazon+the+best+website (Flow has been represented using the red shaded or highlighted region).
After this forward pass begins where When Amazon is passed also the context that Amazon from backward pass is also feeded. When these both are combined then we get the correct NER that is "Organization".



$$h_t = T\left(W_h h_{t-1} + W_i x_t + b\right)$$

Hidden state Output at t timestamp of RNN/LSTM/GRU cell in the forward pass

$$\overleftarrow{h_t} = T\left(\overleftarrow{W_h}\overleftarrow{h_{t-1}} + \overleftarrow{W_i}\overleftarrow{x_t} + \overleftarrow{b}\right)$$

Bi-directional RNN

Hidden state Output at t timestamp of RNN/LSTM/GRU cell in the backward pass

T represents the tanh activation function

$$Y_t = \text{Activation}\left(W\left[\overrightarrow{h_t}\ \overleftarrow{h_t}\right] + b\right)$$

Finally, we are going to concatenate both the forward and backward hidden states equation at timestamp t to generate the Output at the timestamp t