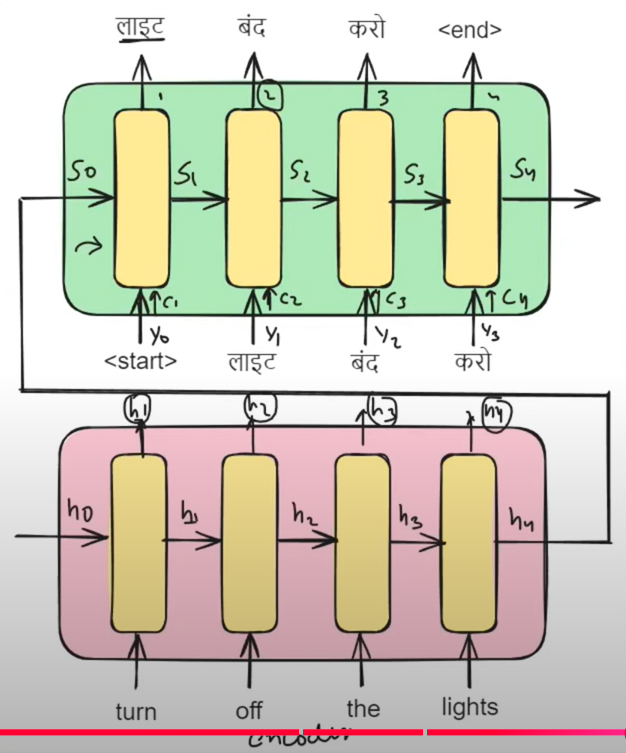
## horizontal line

Attention Mechanism

05.06.2025

# Attention Mechanism

The problem with the encoder decoder mechanism is a long term dependency problem i.e. can’t maintain sequences of more than 25 words.

In that case the attention mechanism solves the issue by seeing that we have to provide the whole sentence at output at each timestamp that is of no use. So we ensure an attention layer that provides only those hidden states of the encoder that are of use to the current decoder hidden state in output prediction.

At the encoder decoder we have to provide [Y1, S1] at i=2.

But in vanilla encoder decoder at i=2 we provide 3 inputs [Yi-1, Si-1, Ci] where “Ci” is attention input.

Ci is a vector with its dimension same as that of the hi.

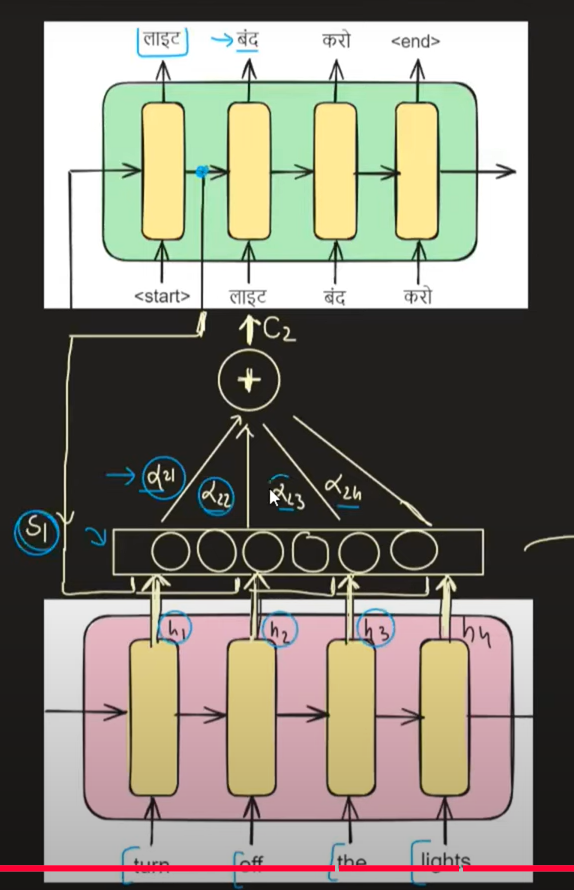
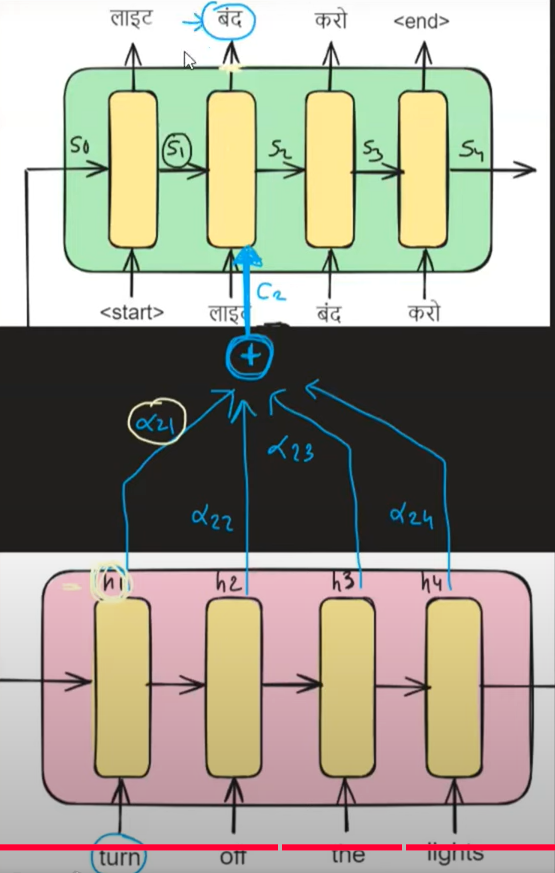
E.g. C1 = + + +

Therefore Ci =

The main question is how we find out this

is an alignment vector / similarity vector which requires h1 , S1 .

S1 is the previous hidden state of the decoder which is used to give context of how the output “Y1” at i=1 is predicted and it is used to predict the next output.

The f is like an ANN which is trained that gives Y2 as output and it is trained along with the decoder.

E.g. At t=2 , C2 = + + +

# 

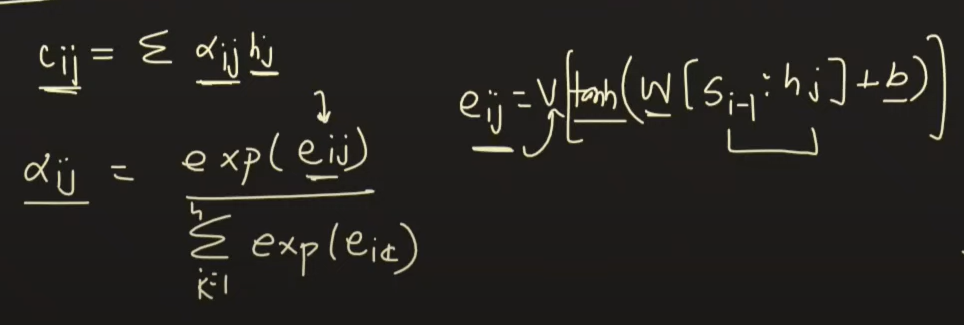
# 

# Bahdanau Vs Luong Attention

## Bahdanau Attention

The basic funda of attention mechanism is same to provide Ci i.e. which encoder hidden states are useful to predict decoder current output.

Ci = Now how we find this

1. We encode input of the sentences as h1 , h2 , h3 , h4 which all are vectors of 4 dimension.
2. Let S0 = [e f g h]
3. Let h0 = [a b c d]
4. Now combinelly [S0 : h0] will look like

[S01 S02 S03 S04 h11 h12 h13 h14]

[S01 S02 S03 S04 h11 h12 h13 h14]

[S01 S02 S03 S04 h11 h12 h13 h14]

[S01 S02 S03 S04 h11 h12 h13 h14]

This is we have 4 rows 8 columns i.e. (4X8)

Weights are (8X3) therefore (4X8) (8X3) = (4X3)

(4X3) (3X1) = (4X1) i.e. the [eij] on which after applying softmax layer gives [].

We send all 4 rows in one batch.

From here i can calculate C1 , onwards using C1 , S0 , Y0 we will calculate Y1.

After that we calculate S1.

Similarly in next t=2 , we calculate C2 using same weights , using C2 , S1 , Y1 we calculate Y2.

When a full sentence is completed the backpropagation starts and updated weights will be used for next sentence.

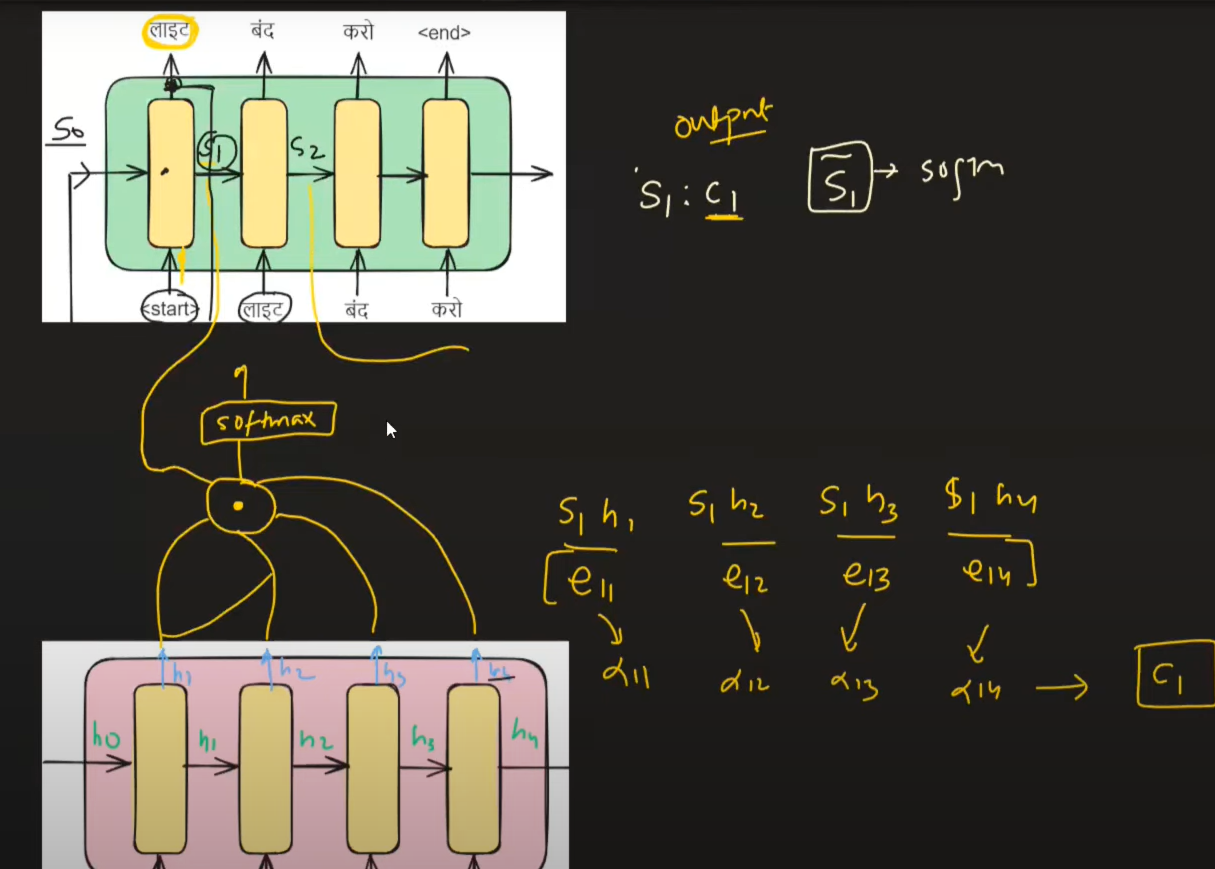
# Luang Attention

What we done in luang is simply dot product instead of ANN .

Let Si = [a b c d]

Hj = [e f g h]

It uses current decoder hidden state to get more updated information.

This dot product also gives the same output and it reduces the parameters.

At t=1

[s1h1 s1h2 s1h3 s1h4]

This gives [e11 e12 e13 e14]

On applying softmax it gives [11 12 13 14]

Now C1 is calculated which is given as output

S1 : C1 = S1\_bar , on which applying softmax it gives Y1.

Similarly at t=2 , we calculate C2 , further S2 : C2 = S2\_bar —---> Y2 .

Main point of difference is we use current updated decoder hidden state and Ci is given in output not in input.