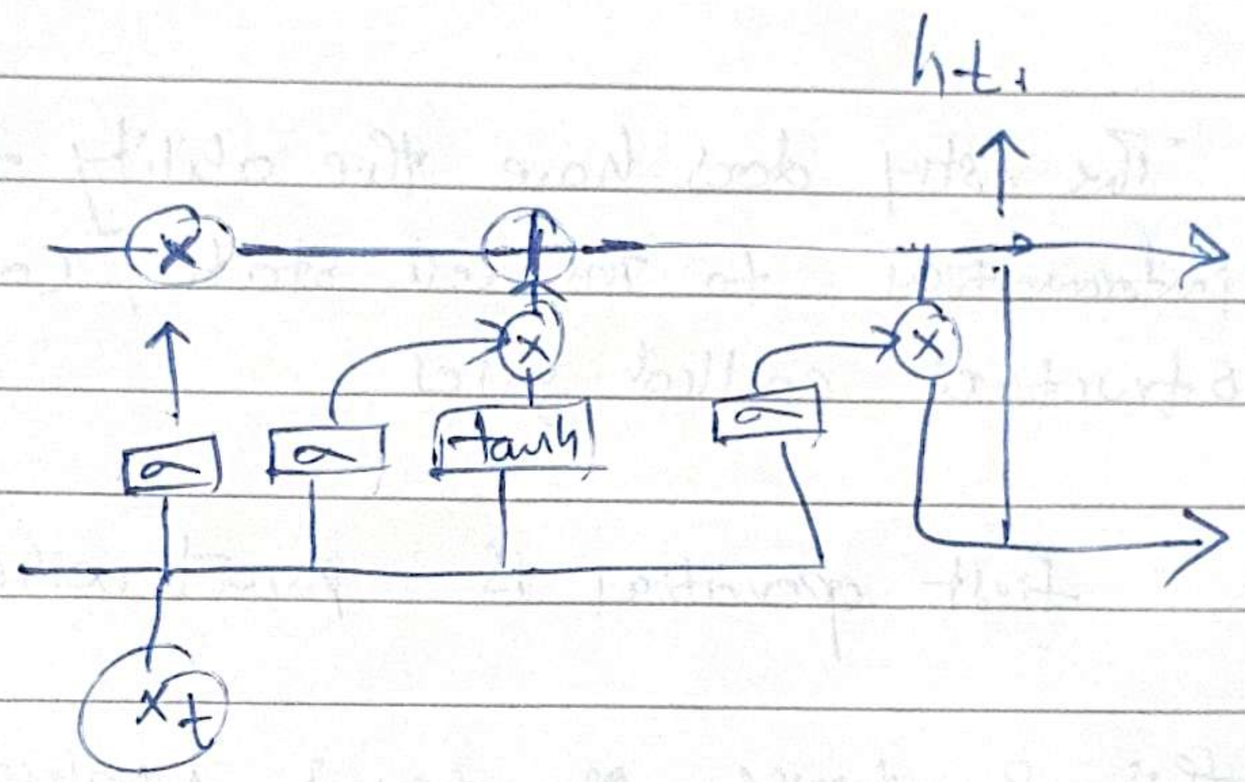
LSTM:

Lstms are explicitly designed to avoid the long term dependency problem Remembering information for long periods of time is practically their default behavior not something they struggle to learn.

In Lstm's have this chain like structure, but repeating module has a different structure instead of having the single neural network layer, there are four interacting layers.



Memory cell

forget gate

Input gate

output gate.

□ ⇒ Neural network layer

⊙ ⇒ Pointwise operation

→ ⇒ Vector transfer

⇉ ⇒ Concatenate

↳ copy

Memory cell is used to Remember or forget the memory based on the content of the input.

ex: text generation the words which are not been used are forgotten.

The key to LSTMs is the cell state, the horizontal line running through the top of the diagram.

The cell state is kind of like a conveyor belt. It runs straight down the entire chain with only some minor interactions. Its very easy for information to just flow along it unchanged

The LSTM does have the ability to remove (or) add information to the cell state, carefully regulated by structures called gates

first operation is point wise operation.

this pointwise operation is a matrix multiplication to the ['1,0] 0 → to forget the data 1 → retain the data

(+) addition operation in last step we are forgetting some information So additional information need to be added

Gates are a way to optionally let information through. they are composed out of a Sigmoid neural net layer and a pointwise multiplication operation.

The Sigmoid layer output numbers between 0 to 1 how much component should be let through A value of zero means let nothing through while a value of one means let everything through.

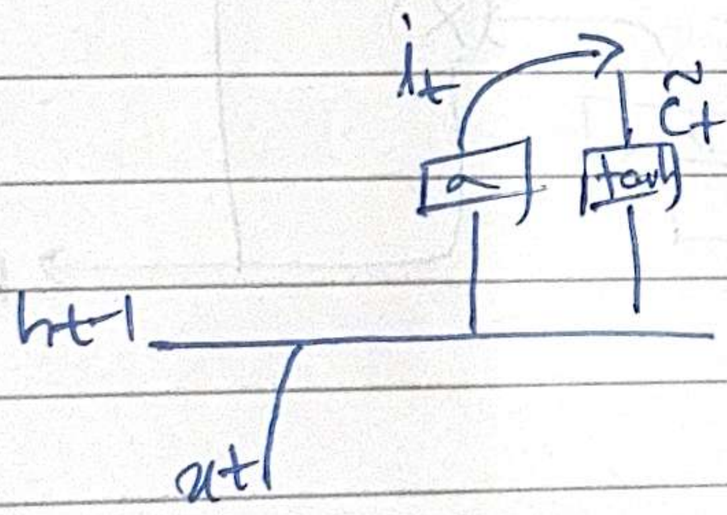
① The first Step is to decide what information we are going to throw away from the cell State. And the decision is made by a Sigmoid layer and the gate is called forgot gate.

=> It looks at h_{t-1} and x_t and outputs a number between $0 \rightarrow 1$ for each number in the cell state C_{t-1}

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

② The next Step is to decide what information we are going to store in the cell state. This has two parts.

first Sigmoid layer called the input gate layer decides which values we will update next a tanh layer creates a vector of new candidate values, $\tilde{C}_t$



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

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

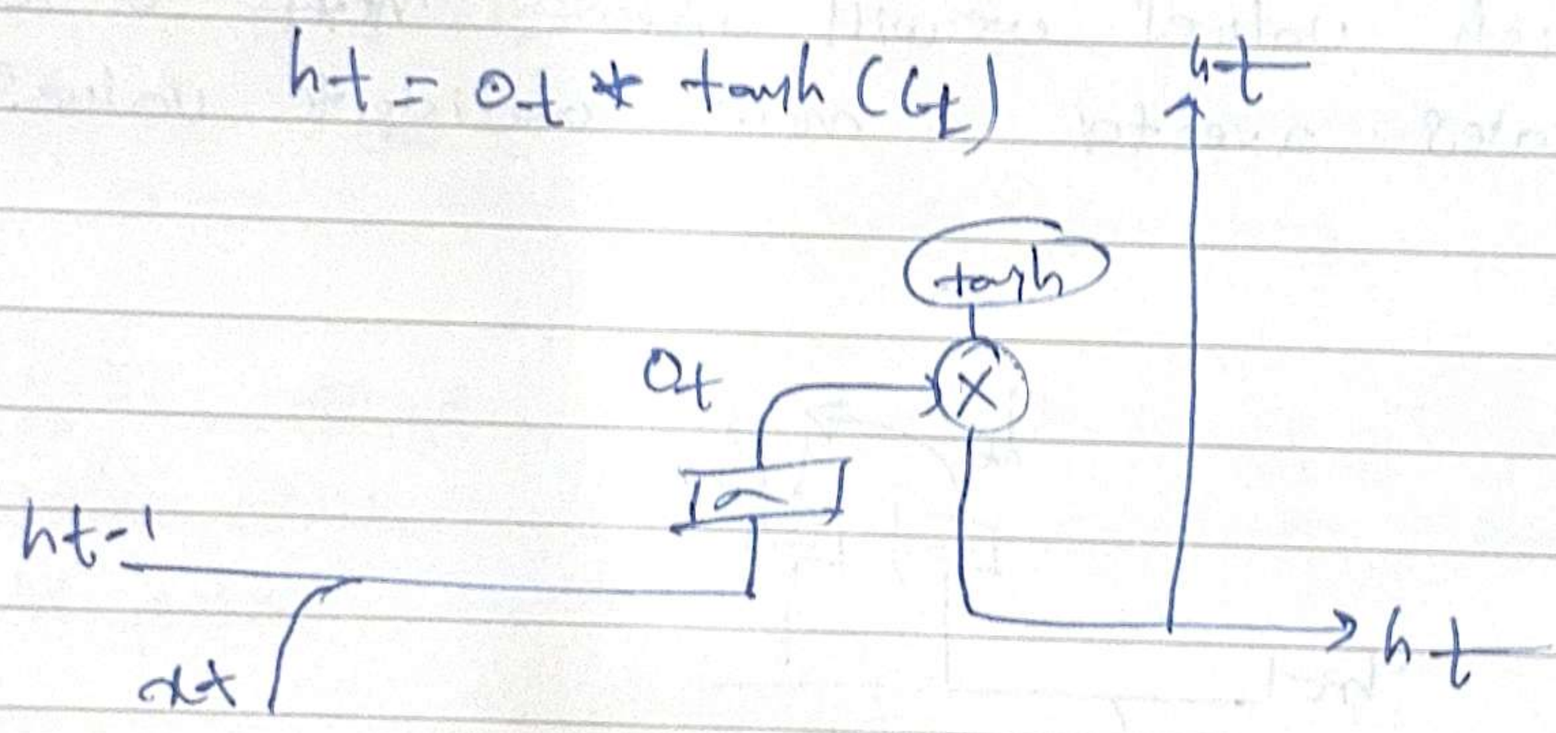
Above we will Upde the old cell state $C_{t-1} \rightarrow C_t$

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

This output will be based on our cell state first we will run the Sigmoid layer which decides what parts of the cell state we are going to output then we put the cell state through tanh and multiply it by the output of the sigmoid gate

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

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



And the hidden state or short memory will update and the goes to next h_{t+1} hidden state.

Gers & Schmidhuber: adding peephole connections

Here we let the gate to look at the cell state.

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

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

$$o_t = \alpha(w_o \cdot [c_t, h_{t-1}, x_t] + b_o)$$

Another variation is to use the coupled forget and input gate. Instead of separately deciding what to forget and what to add we make these two decisions together.

$$c_t = f_t * c_{t-1} + (1 - f_t) * \tilde{c}_t$$

1. Hidden State → h_t

Short term memory
Directly used for predictions
Shape (hidden_unit, .)

2. Cell State → c_t (Long term)

Shape (hidden_units, .)