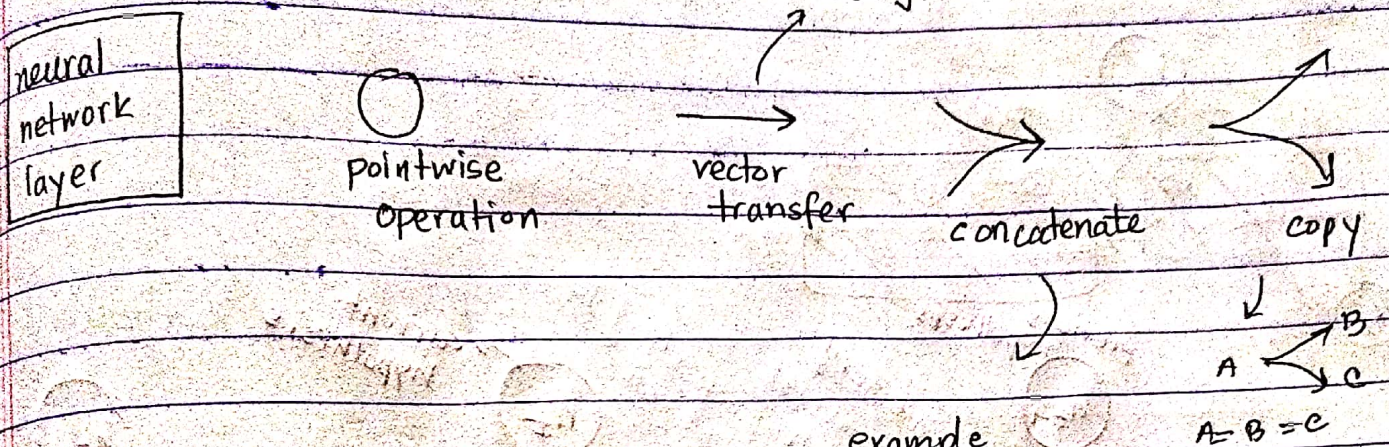
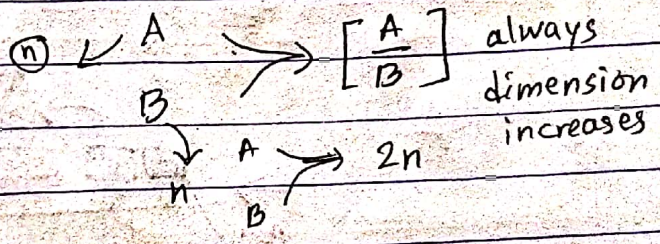
07/03/2022

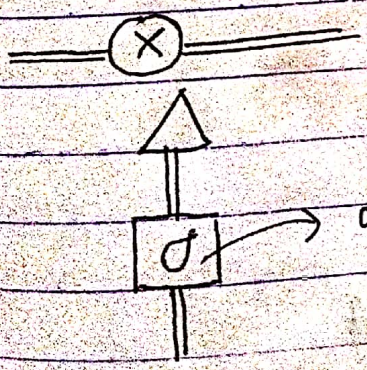
Neural Topic: LSTM



example



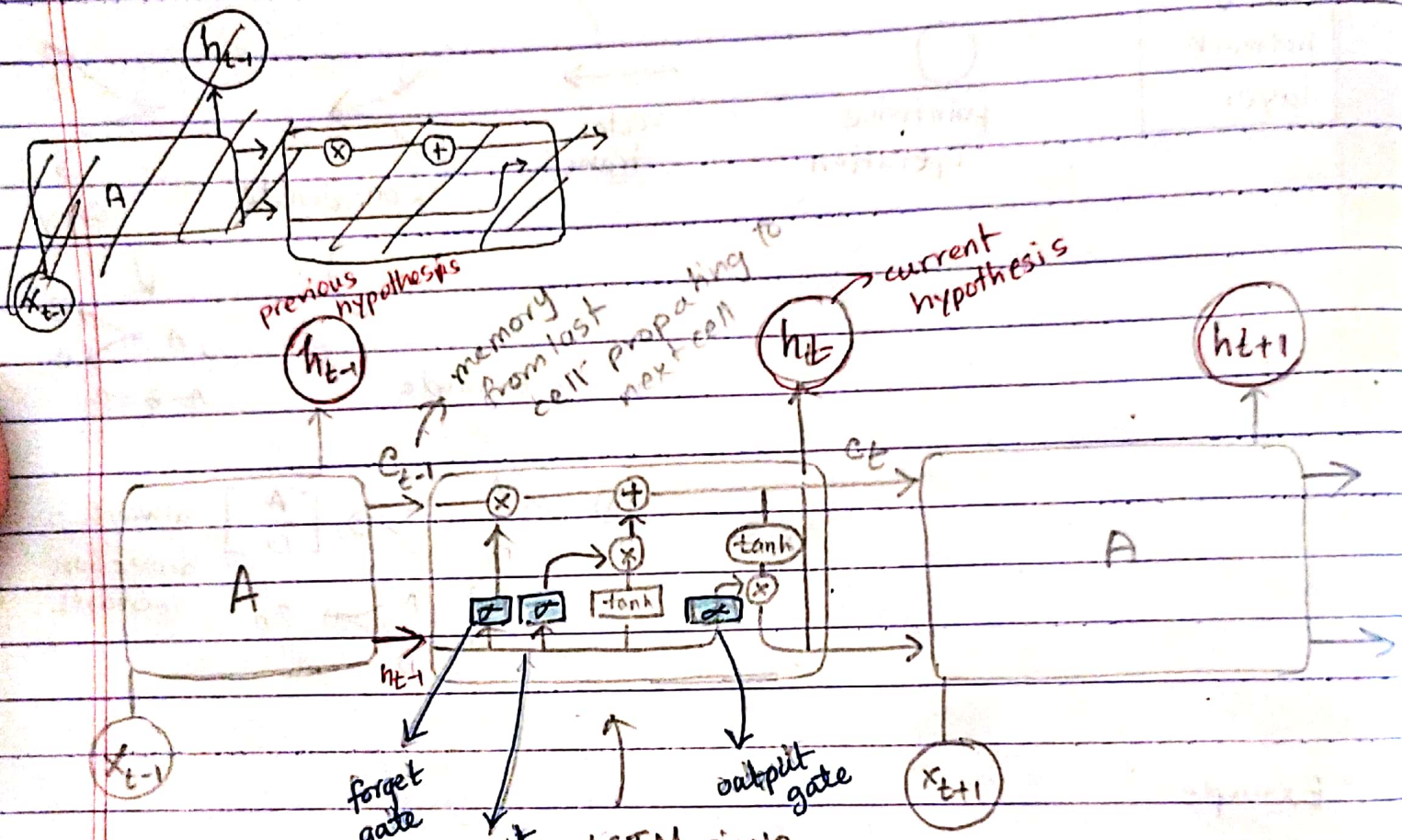
Example



activation function [using sigmoid] example of neural network

sigmoid = (0~1) acts like a gate designed by the activation function of the neural network.

Example:



LSTM single cell

Past state x_{t-1}

present state x_t

Here are 4 neural network gates and 3 ~~of~~ acha of them are σ (sigmoid) we will represent them using 3 names.

(a) forget gate

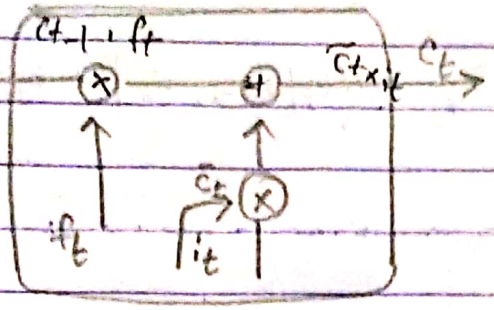
(b) input gate (jei info ashtese tar koto percent info ami current cell e hold korbo)

(c) output gate.

koto tuku output and present cell theke next cell e propagate korbo.

~~tahh~~

tanh → enhances or shrinks current info.



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

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

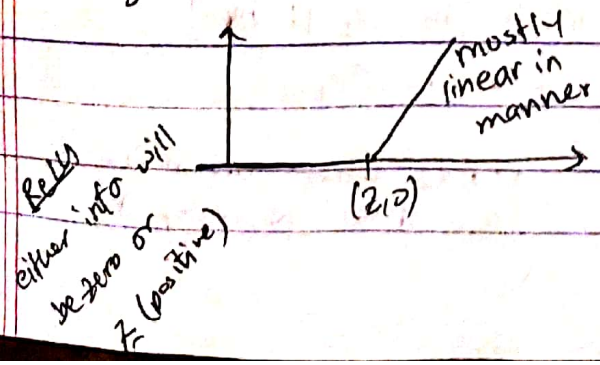
$$i_t = \sigma\left(W_i \left[h_{t-1}, X_t\right] + b_i\right)$$

$$\tilde{C}_t = \tanh\left(W_{infor} \left[h_{t-1}, X_t\right] + b_{info}\right)$$

↓ updated /enhanced information

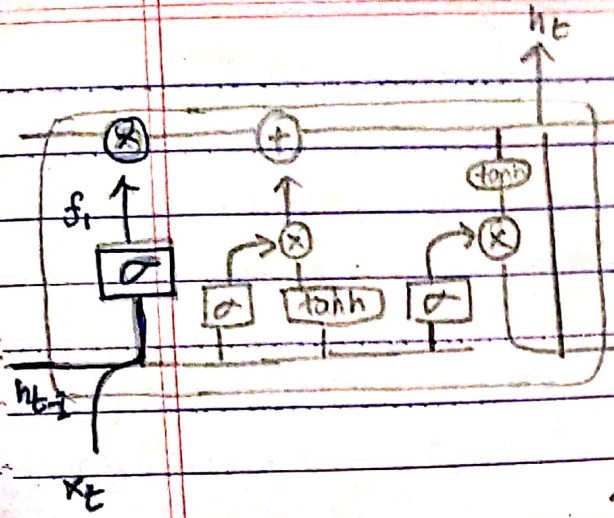
What if we use ~~RE~~ ReLU replaces tanh?

tanh has non linear characteristics as a result it can shrink and enhance. But ReLU has properties such as the graph shown below:

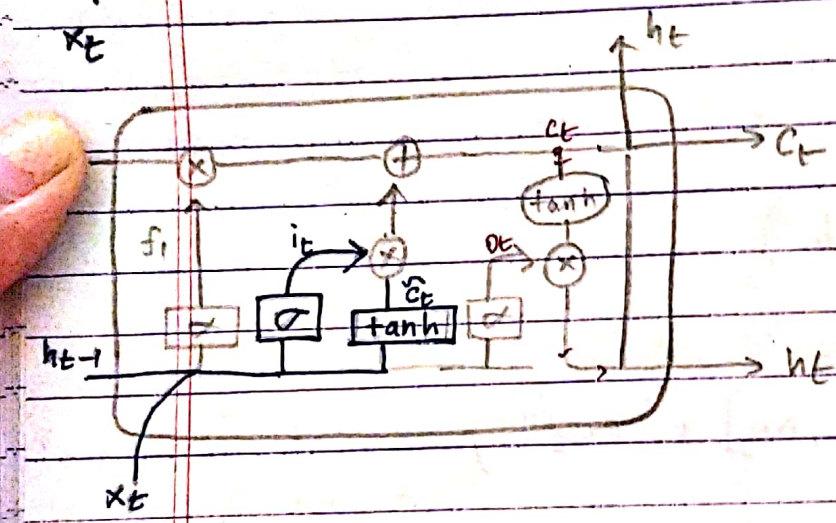


so it cannot ~~inf pprte~~ enhance and shrink info as fast so in LSTM structure we used tanh.

Summary



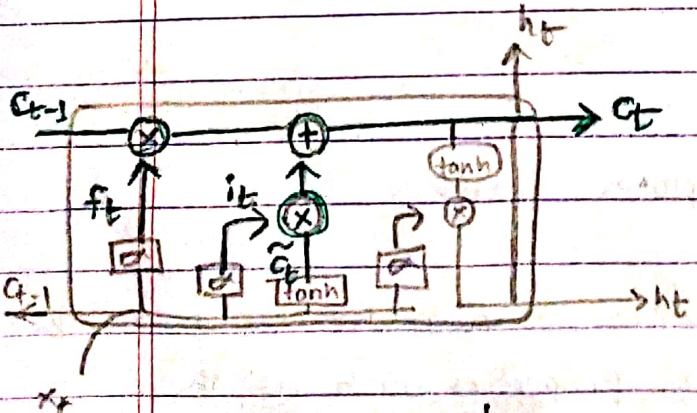
$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f)$$



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

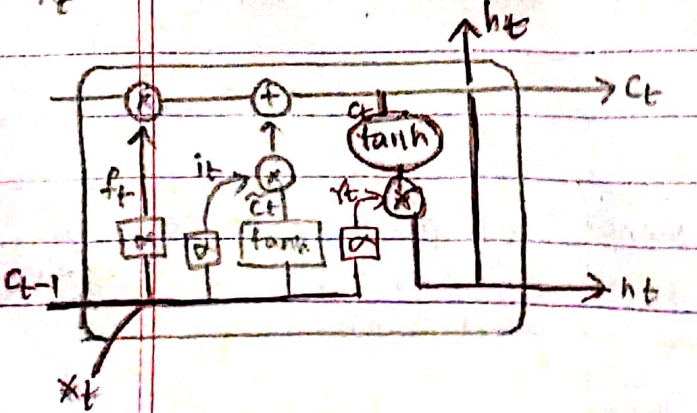
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

i_t decides what component is to be updated. $\tilde{C}_t$ provides change contents.



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

updating the cell state



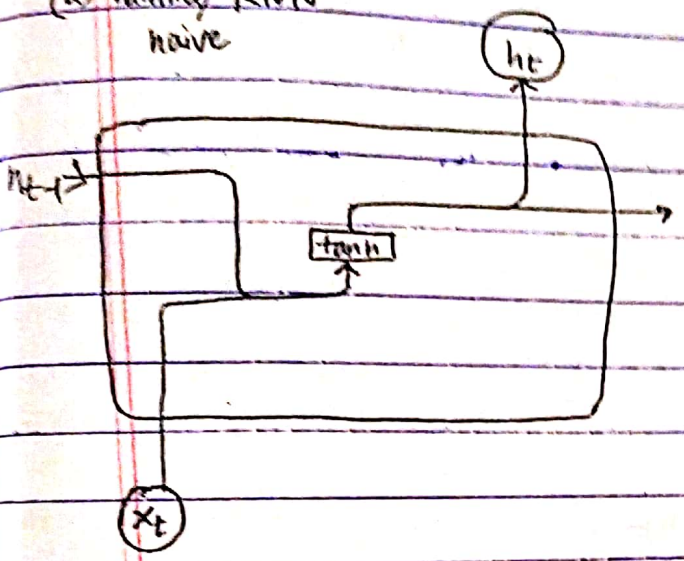
$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o)$$

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

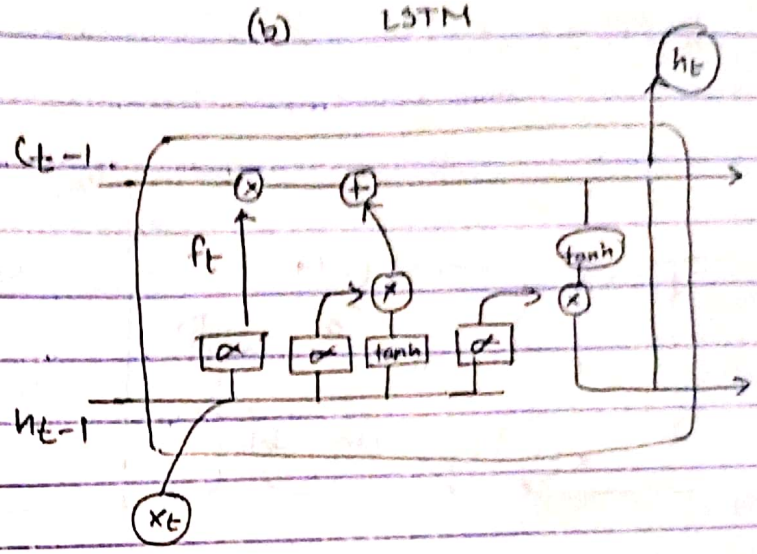
decide what part of the cell state to output

RNN VS LSTM

(a) Vanilla/ RNN
naive



(b) LSTM



$$h_t = \tanh\left(W_R \left[h_{t-1}, x_t\right] + B_R\right)$$

Difference

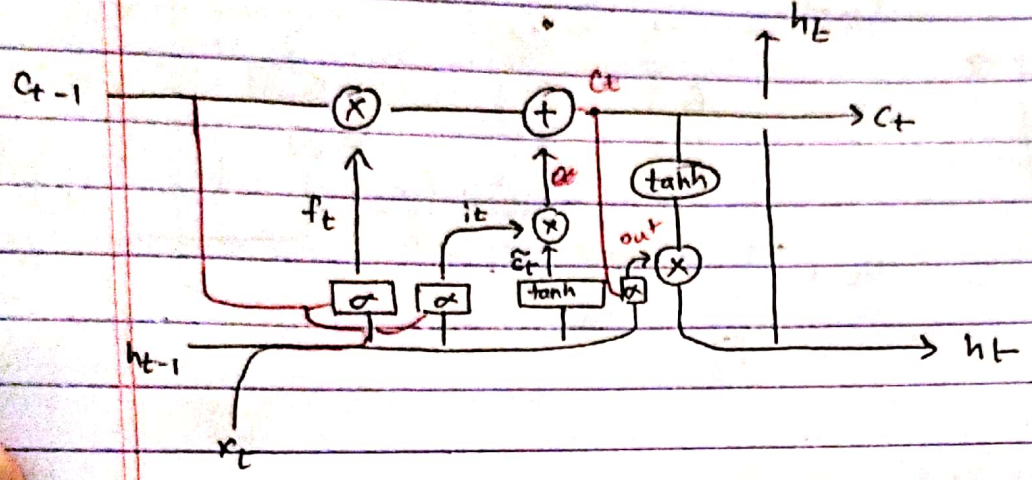
① no cell state in RNN.

② No gates in RNN, no sigmoid locking key.
LSTM has 3 sigmoid keys → i_t, f_t, o_t

③ LSTM expensive, kan hold longer memory.
prone to vanishing gradient problem.

gates regulates better flow of info

Peephole LSTM



normal LSTM e only (h_{t-1} and x_t) mile sigmoid er box e dhukto but peephole LSTM e tara arekta jinish ke concate kortese that is previous cell state (memory & infor)

So (C_{t-1}) copy hole 1st sigmoid then second sigmoid e dhukbe.

∴ basically f_t and i_t er info te, C_{t-1} er info dhuktese, but output gate C_t dhuktese

current cell info.

$$f_t = \sigma(W_f [C_{t-1}, h_{t-1}, x_t] + b_f)$$
$$i_t = \sigma(W_i [C_{t-1}, h_{t-1}, x_t] + b_i)$$
$$o_t = \sigma(W_o [C_t, h_{t-1}, x_t] + b_o)$$
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$
$$h_t = o_t * \tanh(C_t)$$

c changes slowly → c^t is c^{t-1} added by something

jodi cell info onk taratari update hoi taile tar shathe c_{t-1} and c_t then current cell state onk fast update hobe so it can hold longer memory.

h " faster → h_t and h_{t-1} can be different.