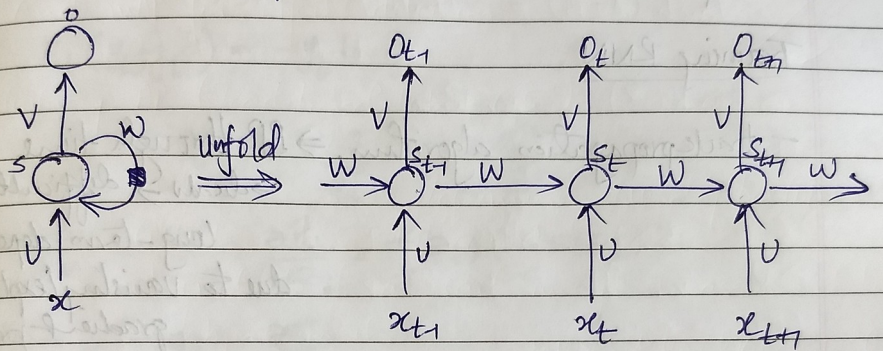
RNN, LSTM, GRU, ConvLSTM

By, Joe Johnson, D17024

Recurrent Neural Networks (RNN)

- make use of sequential information.

recurrent → perform the same task for every element of a sequence.



Unrolled or unfolded into → full n/w
↳ write out n/w for the complete sequence.
- one layer for each input (each word)

formulas:
- x_t is the i/p at time step t. (x_t → one-hot encoding)
- S_t is the hidden state at time step t. (memory)

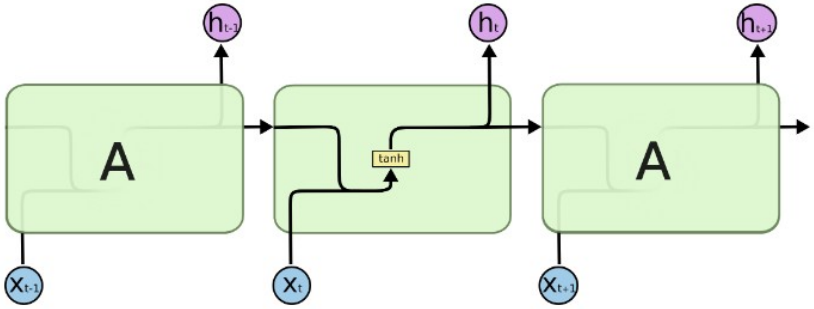
$$S_t = f(U \cdot x_t + W S_{t-1})$$, f → AF (tanh or ReLU)

S_{-1} → first hidden state (initialized to all zeros)

- O_t - o/p at time step t. → vector of prob across o/p ($O_t = \text{Softmax}(V \cdot S_t)$)

S_t → memory of n/w → captures info for all the previous time steps

O_t → solely based on the memory at time 't', other previous by S_t



The repeating module in a standard RNN contains a single layer.

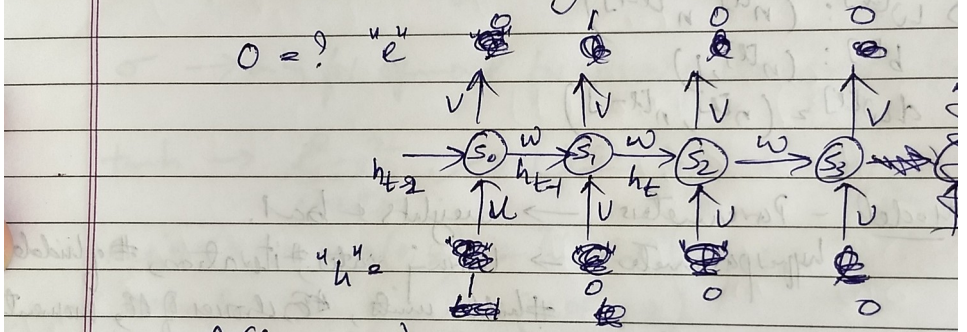
Understanding a Recurrent neuron in details.

"Hello" → task → h-e-l-l → o.

vocabularies → 4 letters {h, e, l, o}

FP in recurrent Neuron.

let $h = [1000]^T$
$e = [0100]^T$
$l = [0010]^T$
} one-hot-encoding



$h_t = f(h_{t-1}, x_t)$

$h_t = \tanh(W_{hh} h_{t-1} + W_{xh} x_t)$

$y_t = W_{hy} h_t$

$W_{hh} = W$
$W_{xh} = U$
$W_{hy} = V$

$$X = \begin{matrix} \text{"h"} & \text{"e"} & \text{"l"} & \text{"l"} \\ \end{matrix}$$
$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$
x_{t-1} x_t

Step 1

$$U = W_{xh} = \begin{bmatrix} 0.287 & 0.846 & 0.572 & 0.486 \\ 0.902 & 0.871 & 0.691 & 0.189 \\ 0.535 & 0.092 & 0.558 & 0.891 \end{bmatrix}$$
← randomly

$$U.x_t = W_{xh} \cdot x_t = \begin{bmatrix} 0.287 & - & - & - \\ 0.902 & - & - & - \\ 0.537 & - & - & - \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0.287 \\ 0.902 \\ 0.537 \end{bmatrix}$$

Step 2

$W = W_{hh} = 0.427$ } randomly selected.
bias, $b_{hh} = 0.567$ } , assign $h_{t-1} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$

$$W.h_{t-1} + b_{hh} = \begin{bmatrix} 0.567 \\ 0.567 \\ 0.567 \end{bmatrix}$$

Step 3

$$z_t = W.h_{t-1} + U.x_t \Rightarrow \begin{bmatrix} 0.287 \\ \vdots \\ \vdots \end{bmatrix} + \begin{bmatrix} 0.567 \\ \vdots \\ \vdots \end{bmatrix} = \begin{bmatrix} 0.854 \\ 1.469 \\ 1.104 \end{bmatrix}$$

$$h_t = \tanh(z_t) \Rightarrow \begin{bmatrix} 0.693 \\ 0.899 \\ 0.802 \end{bmatrix}$$

Step 4

Repeat from Step 1 to 3 for $x = "e" = [0\ 1\ 0\ 0]^T$
$"l" = [0\ 0\ 1\ 0]^T$

$$h_t = \begin{bmatrix} 0.936 \\ 0.949 \\ 0.762 \end{bmatrix}$$

Step 5

$$y_t = W_{hy} \cdot h_t = V.h_t$$

$$V = W_{hy} = \begin{bmatrix} 0.371 & 0.974 & 0.830 \\ 0.341 & 0.282 & 0.659 \\ 0.649 & 0.098 & 0.334 \\ 0.912 & 0.325 & 0.144 \end{bmatrix} \leftarrow \text{randomly selected}$$

Step 6

Probability for a particular letter → Softmax fn.

Softmax (y_t)

- Unlike traditional DNN
  → RNN share the same parameter (U, V, W) across all steps.
  → Reduce Total # of parameter required to learn

- hidden state capture some information about a sequence.

Training RNN

- backpropagation algorithm. → BP through time (BPTT)
  ↳ disadv → difficulty learning long-term dependencies due to vanishing/exploding gradient problem.
  Soln → LSTM.

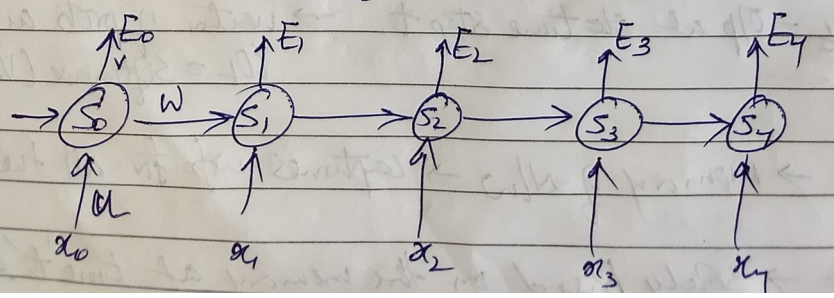
$$S_t = f(U x_t + W S_{t-1}) = \tanh(U x_t + W S_{t-1})$$

$$\hat{y}_t = f(V S_t) = \text{softmax}(V S_t)$$

loss ⇒ Cross entropy loss.

$$E_t(y_t, \hat{y}_t) = -y_t \log \hat{y}_t$$

$$E(y_t, \hat{y}_t) = -\sum_t y_t \log \hat{y}_t$$



BPTT

$$\frac{\partial E_t}{\partial V} = \frac{\partial E_t}{\partial \hat{y}_t} \cdot \frac{\partial \hat{y}_t}{\partial V} = \frac{\partial E_t}{\partial \hat{y}_t} \cdot \frac{\partial \hat{y}_t}{\partial z_t} \cdot \frac{\partial z_t}{\partial V}$$

$$E_t(y_t, \hat{y}_t) = -y_t \log \hat{y}_t$$

$$\hat{y}_t = \text{softmax}(\underbrace{V \cdot s_t}_{z_t}) = \frac{e^{z_t}}{\sum e^{z_t}}$$

$\sigma(z_t) \rightarrow$ softmax

$\sigma'(z_t) = \sigma(z_t)(1 - \sigma(z_t))$

$$\Rightarrow \frac{\partial z_t}{\partial V} = s_t, \quad \frac{\partial \hat{y}_t}{\partial z_t} = \sigma(z_t)(1 - \sigma(z_t))$$

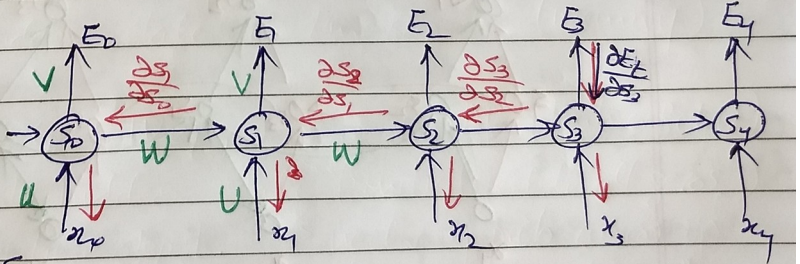
~~$\frac{\partial E_t}{\partial \hat{y}_t} = -y_t$~~

$$\frac{\partial E_t}{\partial z_t} = \frac{\partial(-y_t \log \sigma(z_t))}{\partial z_t} = -y_t \frac{1}{\underbrace{\hat{y}_t}_{\sigma(z_t)}} \cdot \frac{\partial \hat{y}_t}{\partial z_t}$$

$$= \hat{y}_t - y_t$$

$$\Rightarrow \frac{\partial E_t}{\partial V} = (\hat{y}_t - y_t)(s_t)^T$$

$\frac{\partial E_t}{\partial W}$



$$\frac{\partial E_t}{\partial W} = \frac{\partial E_t}{\partial \hat{y}_t} \cdot \frac{\partial \hat{y}_t}{\partial s_t} \cdot \frac{\partial s_t}{\partial W}$$

$s_t = \tanh(U x_t + W s_{t-1}) \Rightarrow s_{t-1}$ depend on s_{t-2} etc.

$$\Rightarrow \frac{\partial E_t}{\partial W} = \sum_{k=0}^{t} \frac{\partial E_t}{\partial \hat{y}_t} \cdot \frac{\partial \hat{y}_t}{\partial s_t} \cdot \frac{\partial s_t}{\partial s_k} \cdot \frac{\partial s_k}{\partial W}$$

$$\frac{\partial E_t}{\partial U} = \frac{\partial E_t}{\partial \hat{y}_t} \cdot \frac{\partial \hat{y}_t}{\partial s_t} \cdot \frac{\partial s_t}{\partial U}$$

$$\& \; s_t = \tanh(U x_t + W s_{t-1})$$

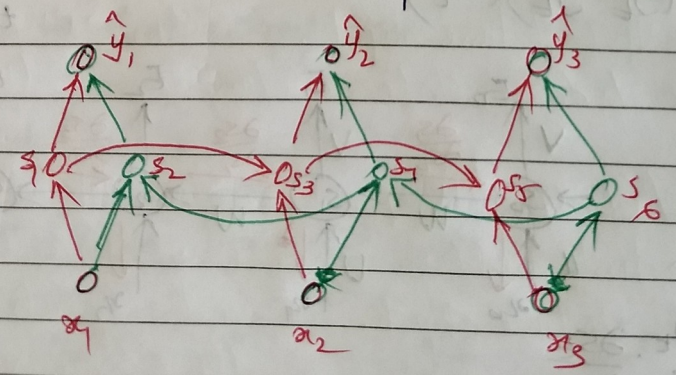
$$\tanh(A \pm B) = \frac{\tanh(A) \pm \tanh(B)}{1 \pm \tanh(A)\tanh(B)}$$

$$\frac{\partial \tanh(x)}{\partial x} = \text{sech}^2 x$$

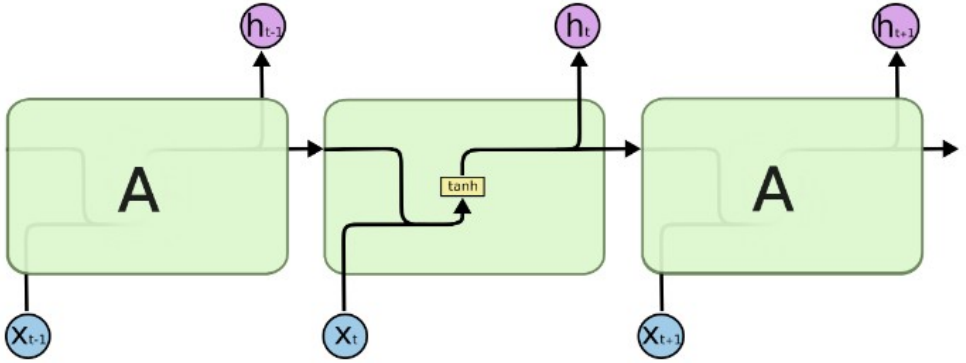
$$\frac{\partial E_t}{\partial U} = \sum_{k=0}^{t} \frac{\partial E_t}{\partial \hat{y}_t} \cdot \frac{\partial \hat{y}_t}{\partial s_t} \cdot \frac{\partial s_t}{\partial s_k} \cdot \frac{\partial s_k}{\partial U}$$

Bidirectional RNN

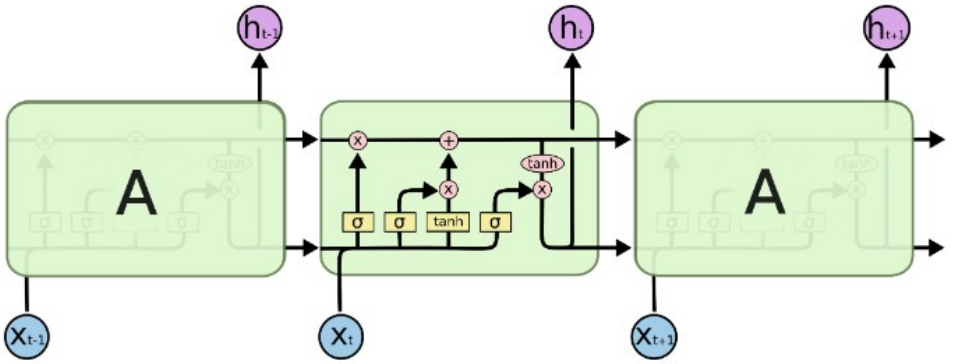
- Op at time 't' depend on both previous and future elements in the sequence.



LSTM



The repeating module in a standard RNN contains a single layer.



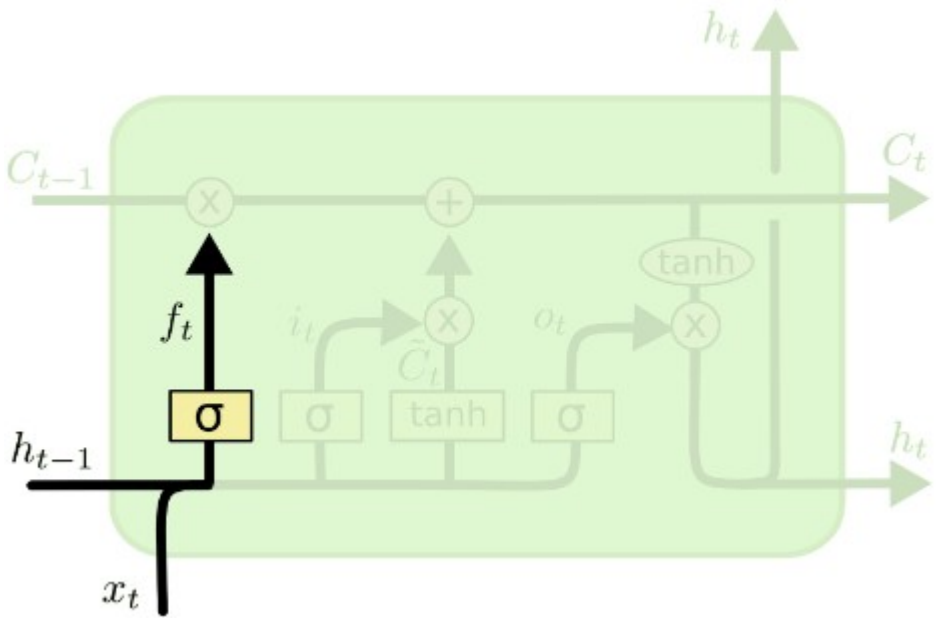
The repeating module in an LSTM contains four interacting layers.

$$
\begin{aligned}
i_t &= \sigma(W_i x_t + b_{ii} + U_i h_{t-1} + b_{hi}) \\
f_t &= \sigma(W_f x_t + b_{if} + U_f h_{t-1} + b_{hf}) \\
g_t &= tanh(W_g x_t + b_{ig} + U_g h_{t-1} + b_{hg}) \\
o_t &= \sigma(W_o x_t + b_{io} + U_o h_{t-1} + b_{ho}) \\
c_t &= f_t \circ c_{t-1} + i_t \circ g_t \\
h_t &= o_t \circ tanh(c_t)
\end{aligned}
$$

$\circ$ denotes the Hadamard or entry-wise product

Forget gate layer

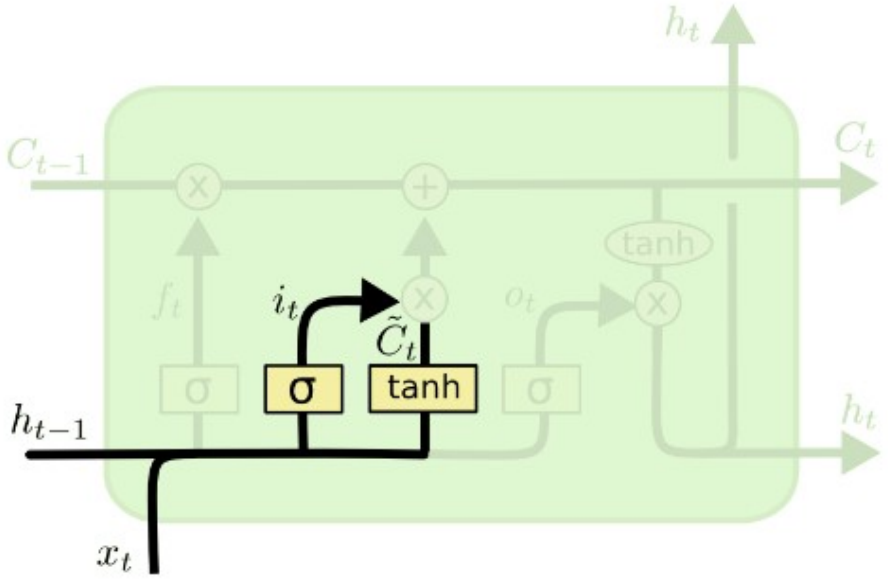
- Decides what information we're going to throw away from the cell state
- This decision is made by a sigmoid layer called the "forget gate layer"
- It looks at h_{t-1} and x_t, and outputs a number between 0 and 1 for each number in the cell state C_{t-1}
- 1 represents "completely keep this" while a 0 represents "completely get rid of this"



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

Input gate layer

- Decide what new information we're going to store in the cell state
- Sigmoid layer called the "input gate layer" decides which values we'll update
- tanh layer creates a vector of new candidate values, C_hat$_t$, that could be added to the state
- Combine these two to create an update to the state

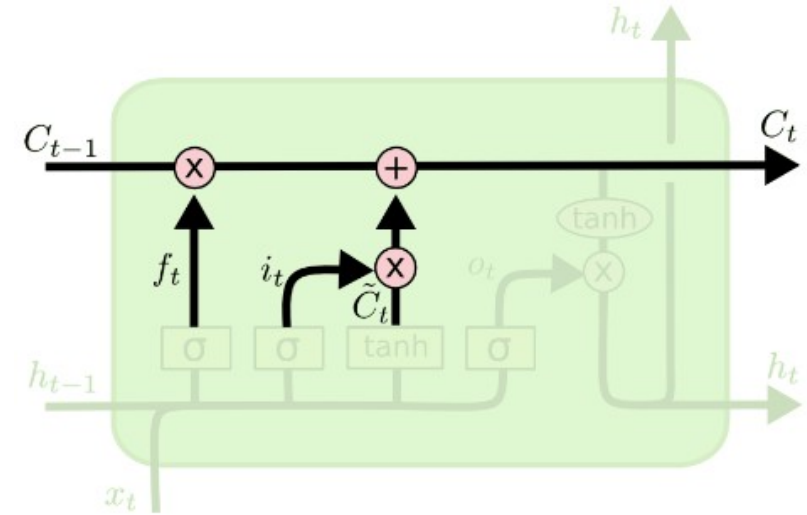


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

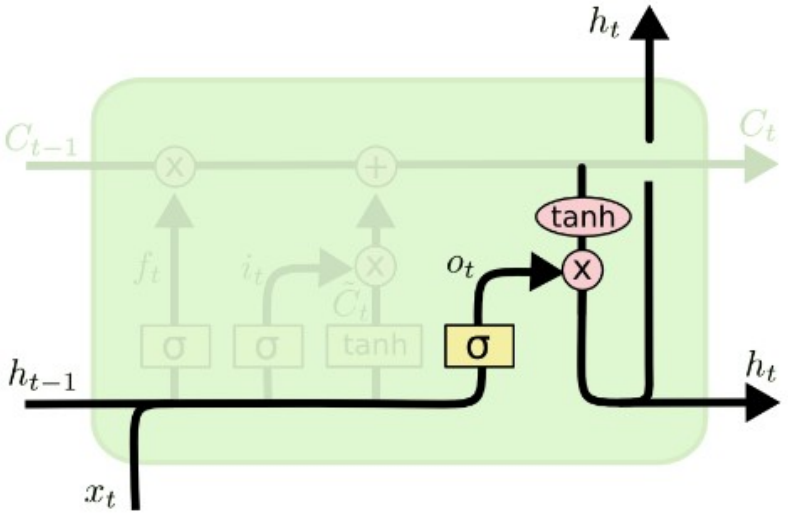
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

Update cell state, Hidden state, Output

- Update the old cell state, C_{t-1}, into the new cell state C_t



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



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

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

GRU - Gated Recurrent Units → learn long term dependencies.
2014

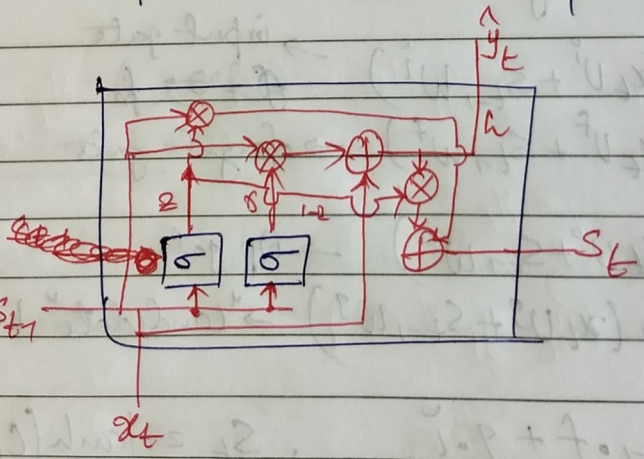
$$z = \sigma(x_t U^z + s_{t-1} W^z) \rightarrow \text{update gate}$$
$$r = \sigma(x_t U^r + s_{t-1} W^z) \rightarrow \text{reset gate}$$
$$h = \tanh(x_t U^h + (s_{t-1} \cdot r) W^h)$$
$$s_t = (1-z) \cdot h + z \cdot s_{t-1}$$

reset gate determines - how to combine the new i/p with the previous memory.

update gate → how much of the previous memory to keep around.

If r=1 reset all '1' & update all '0' → plain RNN.

GRU → don't possess internal memory 'c_t'.



adv
- few parameters → train faster → less data to generalize

Conv-LSTM

FC-LSTM

$$
\begin{aligned}
i_t &= \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci} \circ c_{t-1} + b_i) \\
f_t &= \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf} \circ c_{t-1} + b_f) \\
c_t &= f_t \circ c_{t-1} + i_t \circ \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \\
o_t &= \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co} \circ c_t + b_o) \\
h_t &= o_t \circ \tanh(c_t)
\end{aligned}
$$

Input & state at a timestamp are 1D vectors. Dimensions of the state can be permuted without affecting the overall structure.

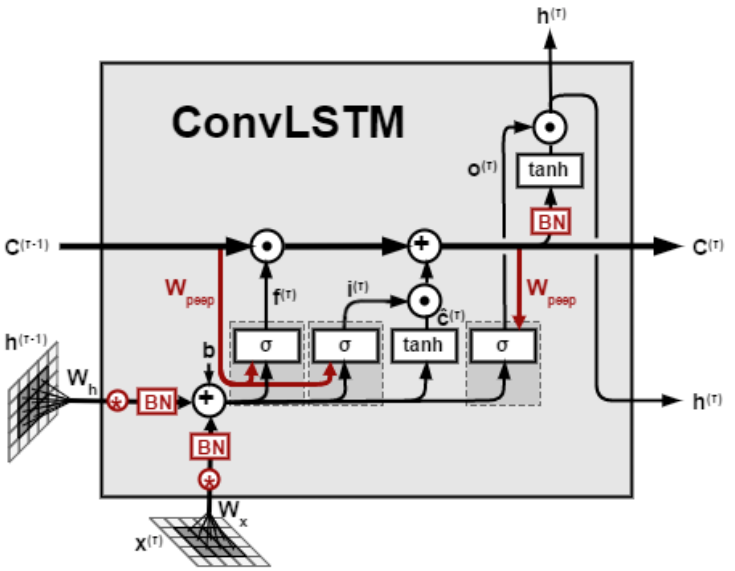
ConvLSTM

$$
\begin{aligned}
i_t &= \sigma(W_{xi} * \mathcal{X}_t + W_{hi} * \mathcal{H}_{t-1} + W_{ci} \circ \mathcal{C}_{t-1} + b_i) \\
f_t &= \sigma(W_{xf} * \mathcal{X}_t + W_{hf} * \mathcal{H}_{t-1} + W_{cf} \circ \mathcal{C}_{t-1} + b_f) \\
\mathcal{C}_t &= f_t \circ \mathcal{C}_{t-1} + i_t \circ \tanh(W_{xc} * \mathcal{X}_t + W_{hc} * \mathcal{H}_{t-1} + b_c) \\
o_t &= \sigma(W_{xo} * \mathcal{X}_t + W_{ho} * \mathcal{H}_{t-1} + W_{co} \circ \mathcal{C}_t + b_o) \\
\mathcal{H}_t &= o_t \circ \tanh(\mathcal{C}_t)
\end{aligned}
$$

Input & state at a timestamp are 3D tensors. Convolution is used for both input-to-state and state-to-state connection.

LSTM

$$
\begin{aligned}
i_t &= \sigma(W_i x_t + b_{ii} + U_i h_{t-1} + b_{hi}) \\
f_t &= \sigma(W_f x_t + b_{if} + U_f h_{t-1} + b_{hf}) \\
g_t &= tanh(W_g x_t + b_{ig} + U_g h_{t-1} + b_{hg}) \\
o_t &= \sigma(W_o x_t + b_{io} + U_o h_{t-1} + b_{ho}) \\
c_t &= f_t \circ c_{t-1} + i_t \circ g_t \\
h_t &= o_t \circ tanh(c_t)
\end{aligned}
$$



Thank you