LSTM AND GRU

$(0,1)$ $\sigma(W_f h_{t-1} + U_f x_t + b_f)$

* Forget Gate -

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

The forget gate decides how much of previous hidden states to forget & how much of new information to retain.

$\hookrightarrow U_f x_t$

* Output gate -

[regulating flow of information in & out of cell.]

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

The output state determines which information from cell state should be output at current time step

* New Candidate values for Cell state -

$(W h_{t-1} + U x_t + b)$

$$g_t = \tanh[W h_{t-1} + U x_t + b]$$

Input modulation gate [current input to LSTM cell & previous hidden state of cell]

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

* Memory of network at time t

$$s_t = \underbrace{f_t}_{\text{forget gate}} \odot \underbrace{s_{t-1}}_{\text{previous cell state}} + \underbrace{i_t}_{\text{Input gate}} \odot \underbrace{g_t}_{\text{New candidate cell state}}$$

$0.7 s_{t-1} + 0.6 g_t$

Two internal states involved - Cell State (C) & Hidden state (H)

Cell state at time $t = \underbrace{s_t}$ [responsible for carrying information through time.]

↳ updation done on basis of
input gate (i_t)
forget gate (f_t)

[forget gate decides how much of previous hidden states to forget & how much of new input to retain.]

[Input state decides how much of previous hidden state to keep & how much of new input to incorporate.]

The hidden state (h_t) is filtered version of cell state & is computed based on the current input & the cell state using output gate.

> <u>Hidden state (layman terms)</u> → <u>Compressed</u> representation of relevant information from the input sequence that is useful for current prediction task.

<u>* Input gate at time t</u>

$$i_t = \sigma(\underbrace{W_i h_{t-1}} + U_i x_t + \underbrace{b_i}) \rightarrow \text{Bias for input gate}$$

Input state decides how much of previous hidden state to keep.
& how much of new input to incorporate → $U_i x_t$

* Hidden state -

output of network at time t

$$h_t = o_t \odot \tanh(s_t)$$

$$a = \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} \quad b = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix}$$

$$\begin{pmatrix} a_1 * b_1 \\ a_2 * b_2 \\ a_3 * b_3 \end{pmatrix}$$

* Output - $\hat{y}_t = g(\underbrace{Vs_t}_{} + c)$

$$\boxed{\begin{aligned} i_t &= \sigma(W_i h_{t-1} + U_i x_t + b_i) \\ f_t &= \sigma(W_f h_{t-1} + U_f x_t + b_f) \\ o_t &= \sigma(W_o h_{t-1} + U_o x_t + b_o) \\ g_t &= \tanh(W h_{t-1} + U x_t + b) \\ s_t &= f_t \odot s_{t-1} + i_t \odot g_t \\ h_t &= o_t \odot \tanh(s_t) \\ \hat{y}_t &= g(Vs_t + c) \end{aligned}}$$

Question: Consider An LSTM Network With One Hidden Layer Of 20 Nodes Used For Predicting The Next Word In A Text Corpus. No Bias Is...

Consider an LSTM network with one hidden layer of 20 nodes used for predicting the next word in a text corpus. No bias is used in any of the nodes. The corpus is of length 1000 words and there are 100 unique words. Assume a 10 dimensional word embedding module outside of the LSTM etwork, whose output is fed to the word predictor LSTM network. What will be the total number of trainable weights in the LSTM network?

We have x_t of shape $\underbrace{10 \times 1}$

x_t is 10 dimensional word embedding.

$$x_t = \begin{pmatrix} 0.5 \\ 0.1 \\ 0.7 \\ 0.11 \\ \vdots \\ 0.9 \end{pmatrix}_{10 \times 1}$$

Following the LSTM architecture, we first need to evaluate parameter sizes for i_t, o_t, f_t, g_t.

Equation	Parameter size
$i_t = \sigma(W_i h_{t-1} + U_i x_t)$ i_t: 20×1; W_i: 20×20; h_{t-1}: 20×1; x_t: 10×1; U_i: 20×10	U_i is of size 20×10. [1 hidden layer with 20 nodes] W_i is of size 20×20 [$\because$ h_{t-1} will be of size 20×1]
$o_t = \sigma(W_o h_{t-1} + U_o x_t)$	U_o is of size 20×10. W_o is of size 20×20.
$f_t = \sigma(W_f h_{t-1} + U_f x_t)$	W_f is of size 20×20 U_f is of size 20×10
$g_t = \sigma(W h_{t-1} + U x_t)$	U is of size 20×10 W is of size 20×20.
Total parameters	$20 \times 10 \times 4 + 20 \times 20 \times 4$ $= 2400.$

We will eventually have

$$\hat{y}_t = g(Vs_t + c)$$

$$\Rightarrow \boxed{\hat{y}_t = g(Vs_t)}$$

$V_{100 \times 20} \; s_{t \, 20 \times 1}$

Here V will be a matrix of order 100×20

[$\because$ s_t is a vector of order 20×1 because
$s_t = f_t \odot s_{t-1} + i_t \odot g_t$]

Total trainable parameters $= \underline{2400} + \underbrace{100 \times 20}$
$= \underline{\underline{4400}}$

GRU

Two gates for GRU → reset gate & update gate.

* Reset Gate

$$r_t = \sigma(W_r \cdot h_{t-1} + U_r x_t + b_r)$$

Reset gate determines how much of past knowledge to forget

* Update Gate

$$u_t = \sigma(W_u h_{t-1} + U_u x_t + b_u)$$

update gate determines how much of past knowledge to be passed on to future.

* New Candidate value for All state

$$g_t = \tanh(W_c (r_t * h_{t-1}) + U_c x_t + b_c)$$

$$\Rightarrow \boxed{h_t = u_t * g_t + (1 - u_t)\, h_{t-1}} \longrightarrow \text{Memory unit at time } t.$$

$$h_t = 0.8 g_t + 0.2 h_{t-1}$$

$$s_t = h_t.$$

$$\boxed{\hat{y}_t = g(V s_t + c)} \longrightarrow \text{[output layer]}$$