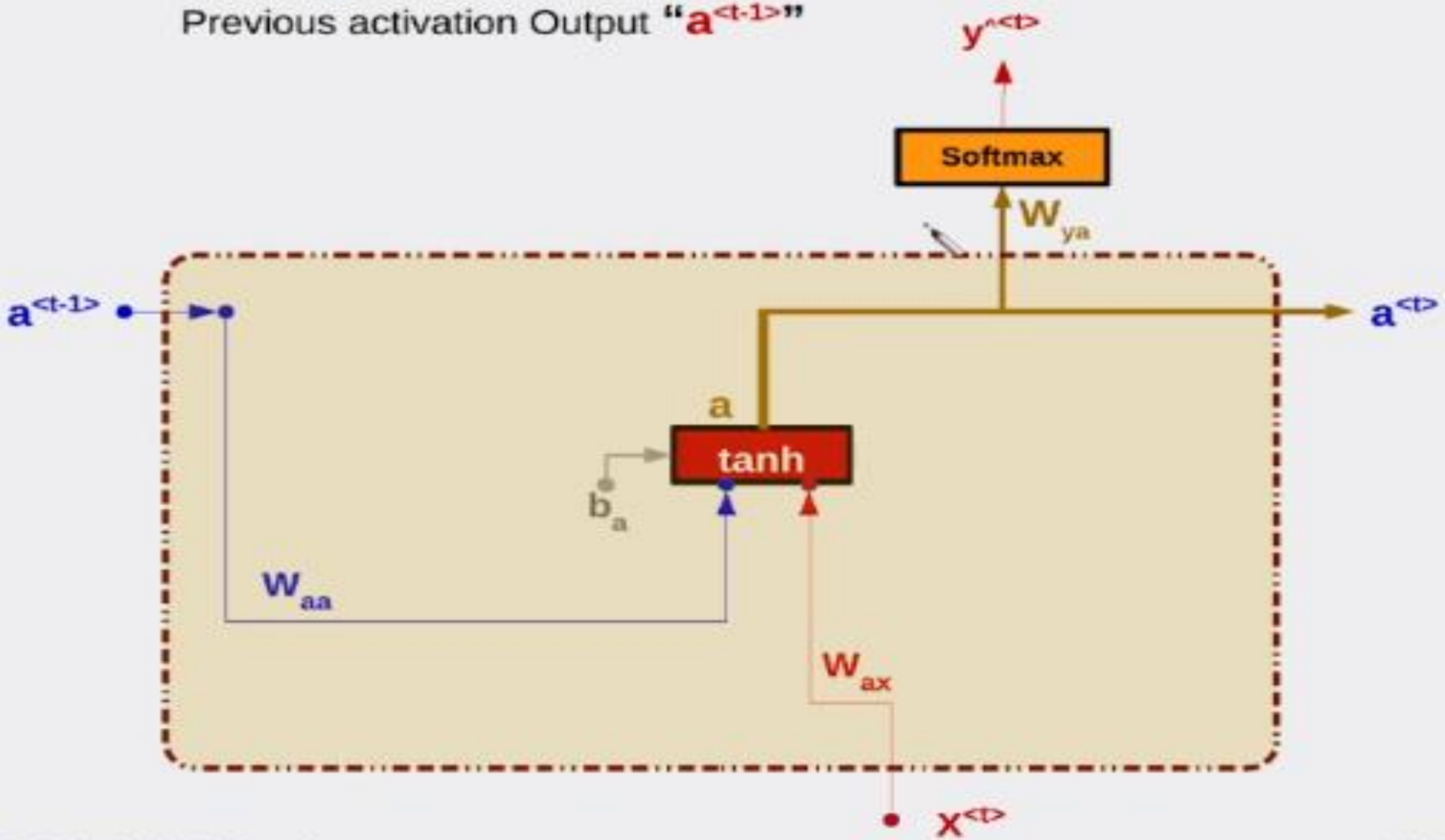
[1] Recurrent Neural Network (RNN)

Recurrent Neural Network (RNN)

Output of Activation Function at time t; "$a^{<t>}$" depends on BOTH:-

Input "$X^{<t>}$" and

Previous activation Output "$a^{<t-1>}$"



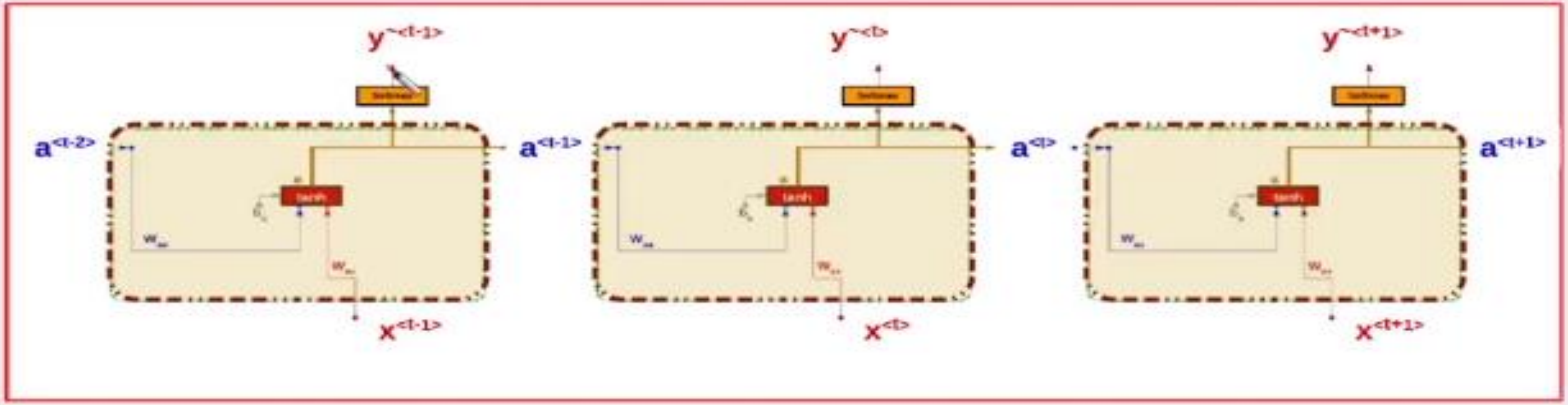
Recurrent Neural Network (RNN)

Output of Activation Function at time t; **"$a^{<t>}$"** depends on BOTH:-

Input **"$X^{<t>}$"** and

Previous activation Output **"$a^{<t-1>}$"**

Inputs at time t-1 , t , t+1

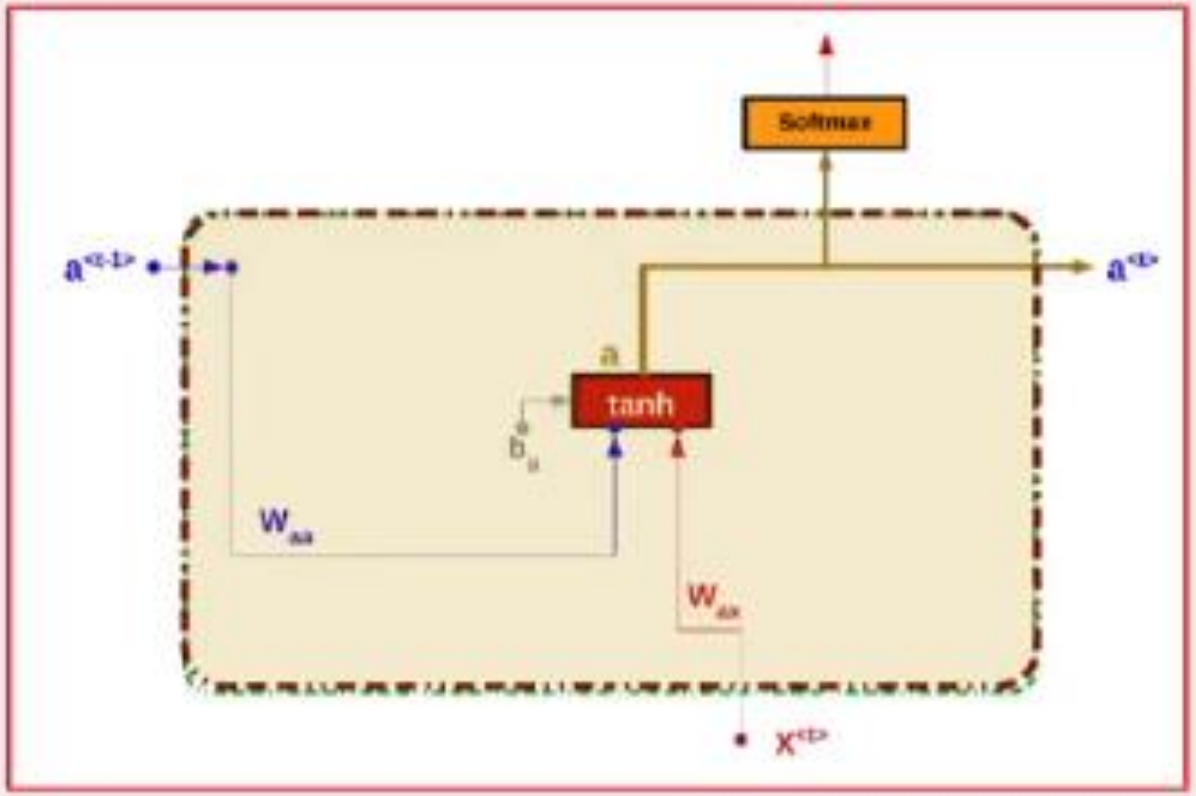


Recurrent Neural Network (RNN)

$$a^{<1>} = F^n(\underbrace{W_{aa}a^{<0>}}_{\text{From Prev.}} + \underbrace{W_{ax}x^{<1>}}_{\text{Current I/P}} + b_a)$$

$$\hat{y}^{<1>} = F^n(W_{ya}\, a^{<1>} + b_y)$$

Non Linearity

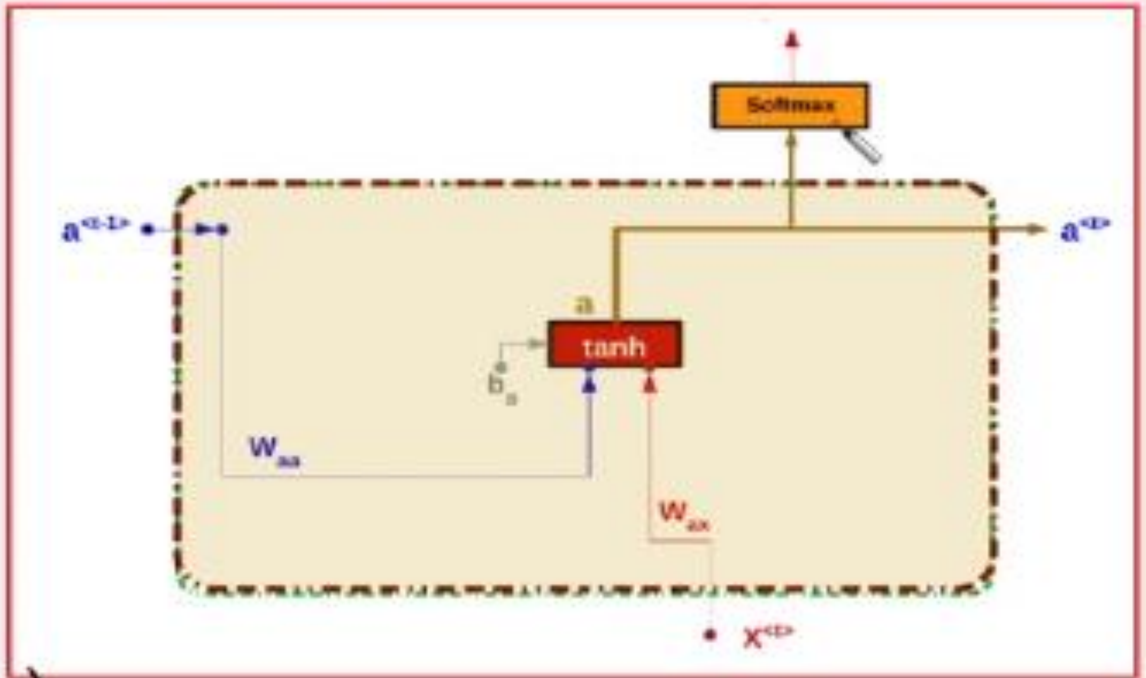


Recurrent Neural Network (RNN)

$$a^{<1>} = F^n\,(\overbrace{W_{aa}a^{<0>}}^{\text{From Prev.}} + \overbrace{W_{ax}x^{<1>}}^{\text{Current I/P}} + b_a)$$

$$\hat{y}^{<1>} = F^n\,(W_{ya}\,a^{<1>} + b_y)$$

Non Linearity



$$a^{<1>} = \textbf{tanh}\,(W_{aa}a^{<0>} + W_{ax}x^{<1>} + b_a)$$ May be tanh, ReLU, ...

$$\hat{y}^{<1>} = \textbf{Softmax}\,(W_{ya}\,a^{<1>} + b_y)$$ Segmoid for Binary O/P, Softmax for Multi-Class O/P

$$a^{<t>} = \textbf{tanh}\,(W_{aa}a^{<t-1>} + W_{ax}x^{<t>} + b_a)$$ May be tanh, ReLU, ...

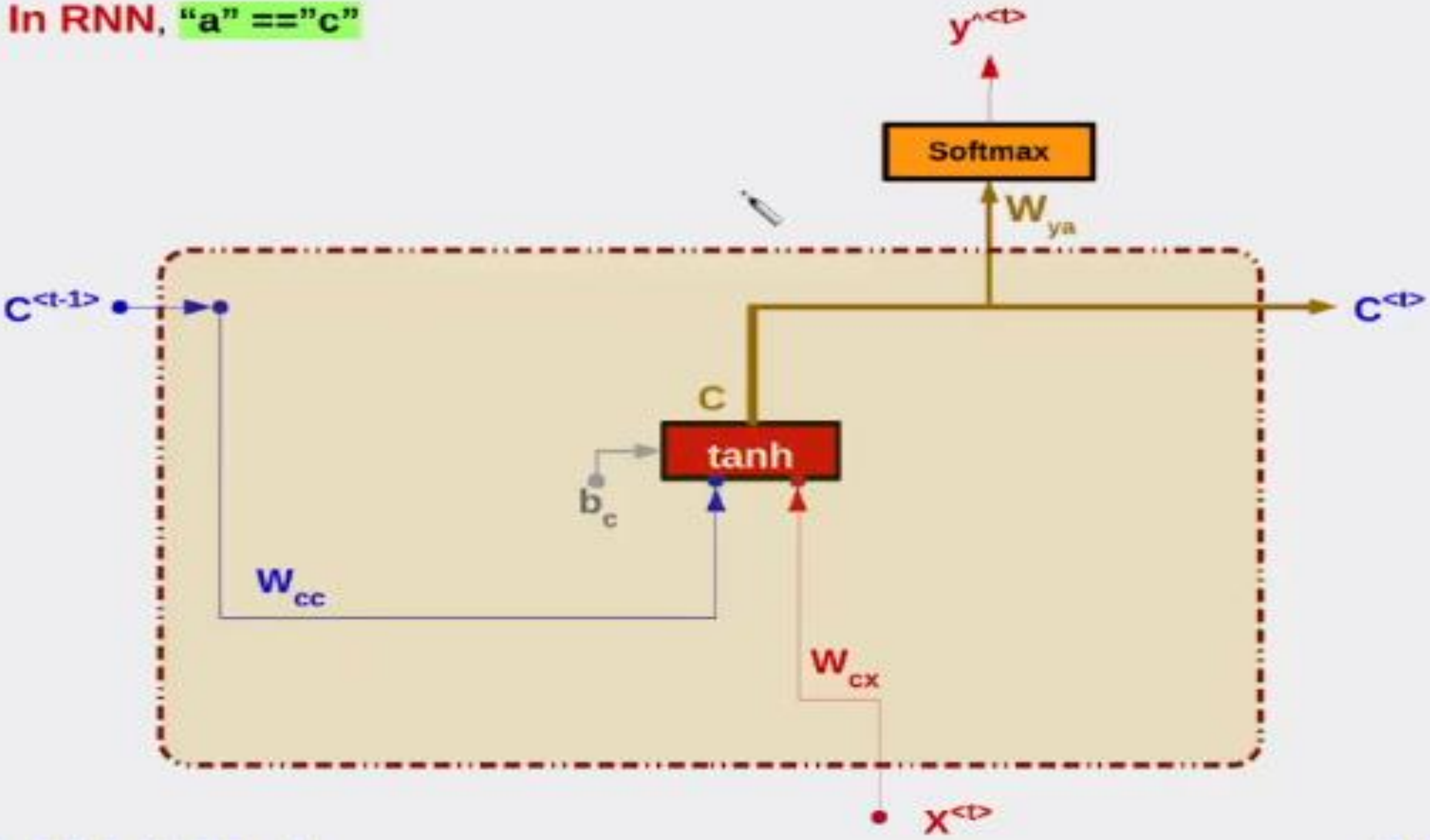
$$\hat{y}^{<t>} = \textbf{Softmax}\,(W_{ya}\,a^{<t>} + b_y)$$ Segmoid for Binary O/P, Softmax for Multi-Class O/P

[2] Gated Recurrent Unit (GRU)

From RNN to GRU

Define **Memory** Cell "**C**" in addition to Output of **Activation** function "**a**".

In RNN, "a" =="c"



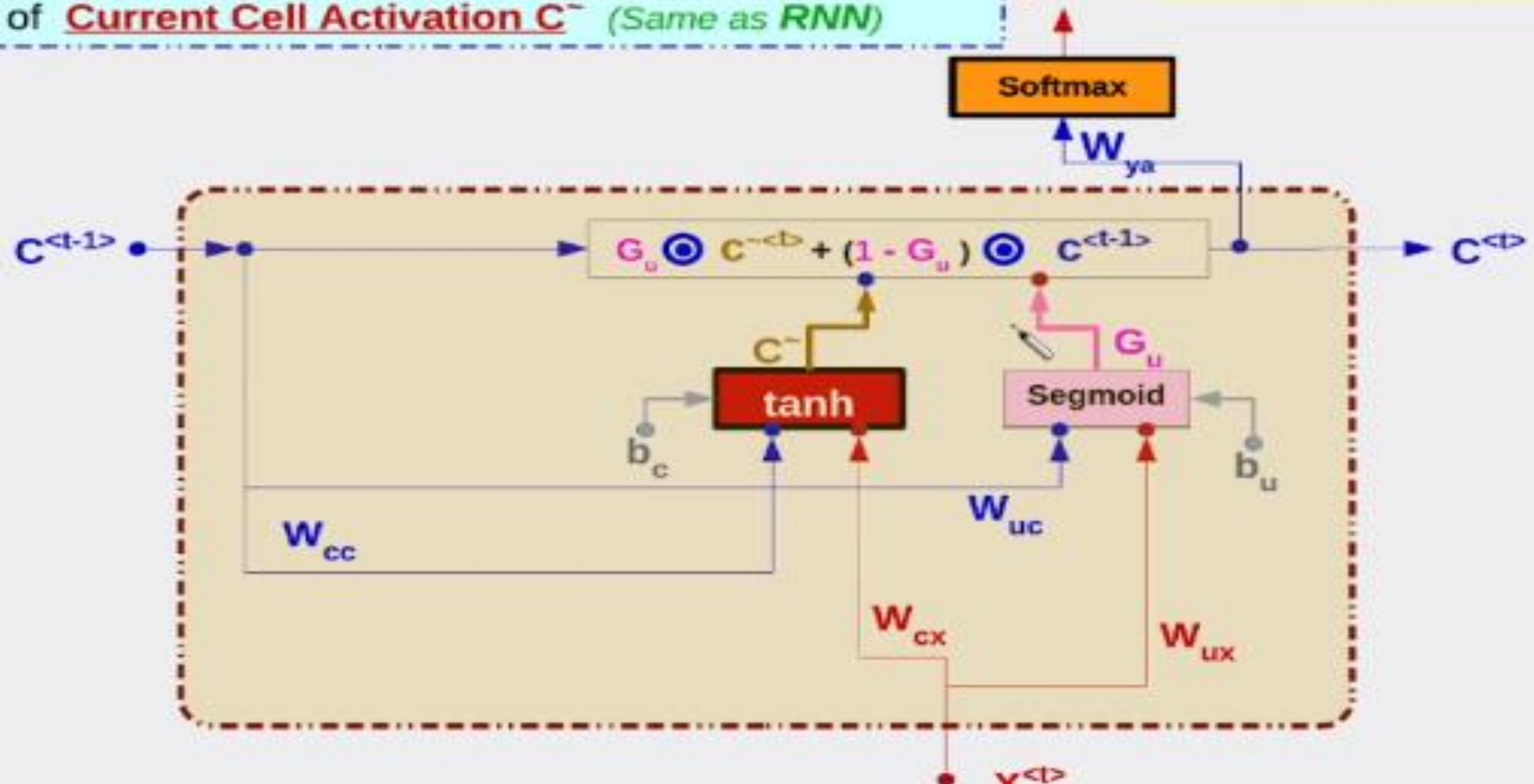
From RNN to GRU

[1] Adding <u>Update Gate</u> G_u

Value of CURRENT Memory Cell $C^{<t>}$=
Percentage of <u>**Previous Memory Cell $C^{<t-1>}$**</u> *(Not Affected by $X^{<t>}$)*
+
Percentage of <u>**Current Cell Activation $C^{\sim}$**</u> *(Same as **RNN**)*

$C^{\sim}$ Candidate Value of Updated Memory

C Value of Updated Memory



From RNN to GRU

[1] Adding Update Gate G_u

Value of G_u is based on:

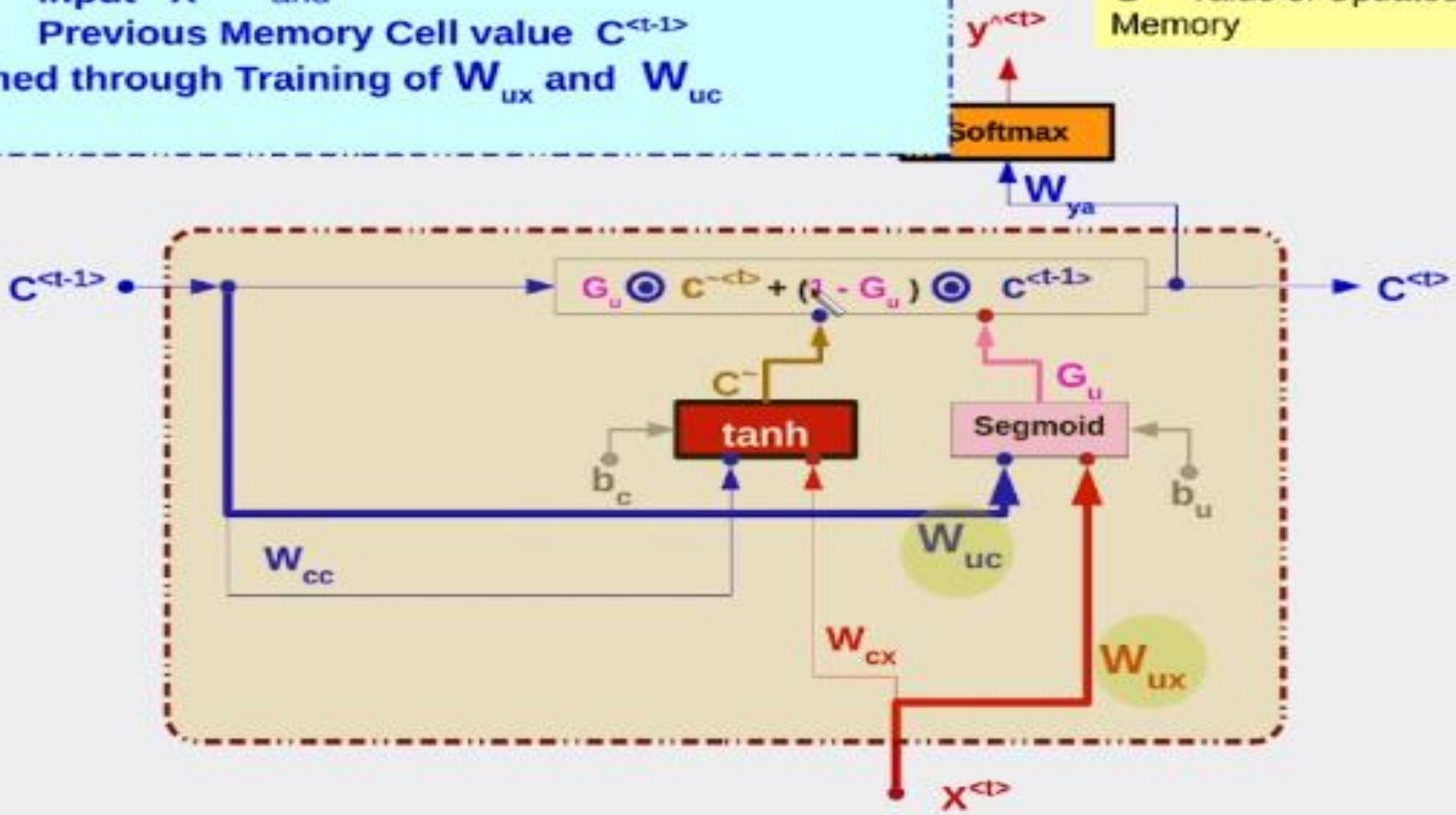
Input $X^{<t>}$ and

Previous Memory Cell value $C^{<t-1>}$

G_u is obtained through Training of W_{ux} and W_{uc}

$C^{\sim}$ Candidate Value of Updated Memory

C Value of Updated Memory



From RNN to GRU

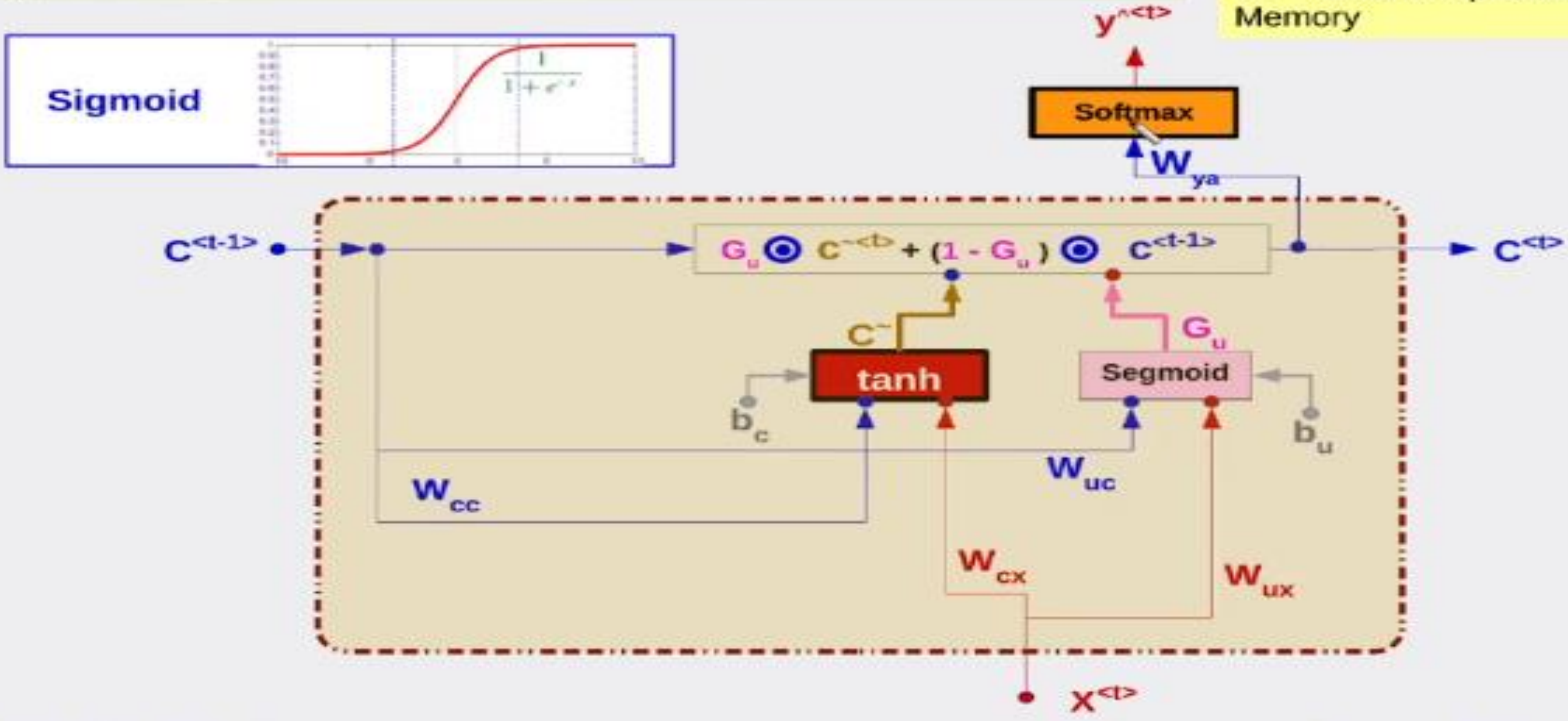
[1] Adding Update Gate G_u

If $G_u = 0$, **Keep** Memory Value "$C^{<t>}$" **Same as** Previous Value "$C^{<t-1>}$"

If $G_u = 1$, **Forget** Previous Memory Value "$C^{<t-1>}$"

$\tilde{C}$ Candidate Value of Updated Memory

C Value of Updated Memory



From RNN to GRU

$$G_u = \textbf{Sigmoid}\,(W_{uc}c^{<t-1>} + W_{ux}x^{<t>} + b_u)$$

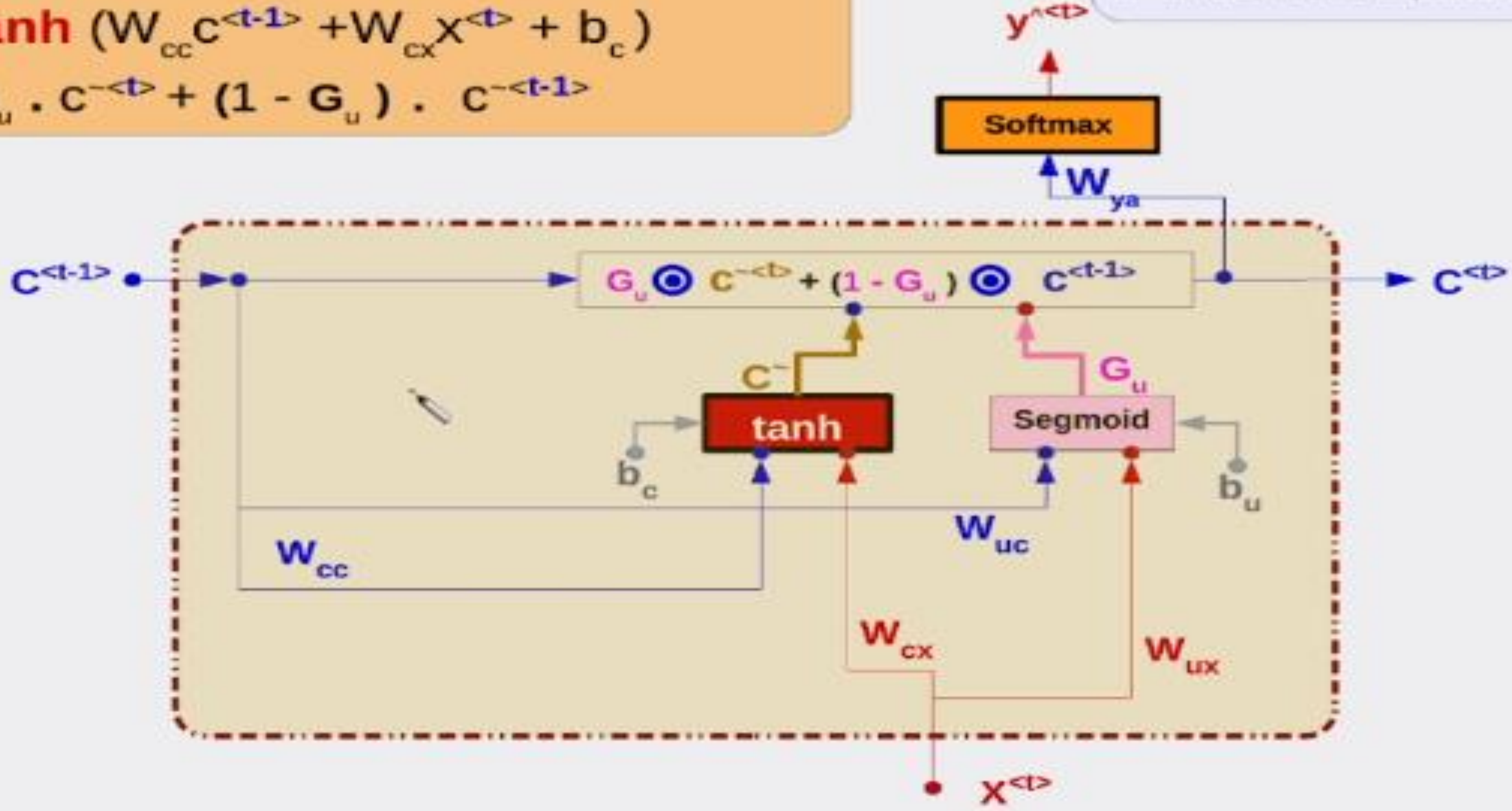
$$c^{\sim<t>} = \textbf{tanh}\,(W_{cc}c^{<t-1>} + W_{cx}x^{<t>} + b_c)$$

$$c^{<t>} = G_u \,.\, c^{\sim<t>} + (1 - G_u)\,.\; c^{\sim<t-1>}$$

$C^{\sim}$ is the Candidate Update

G_u is the Update Gate

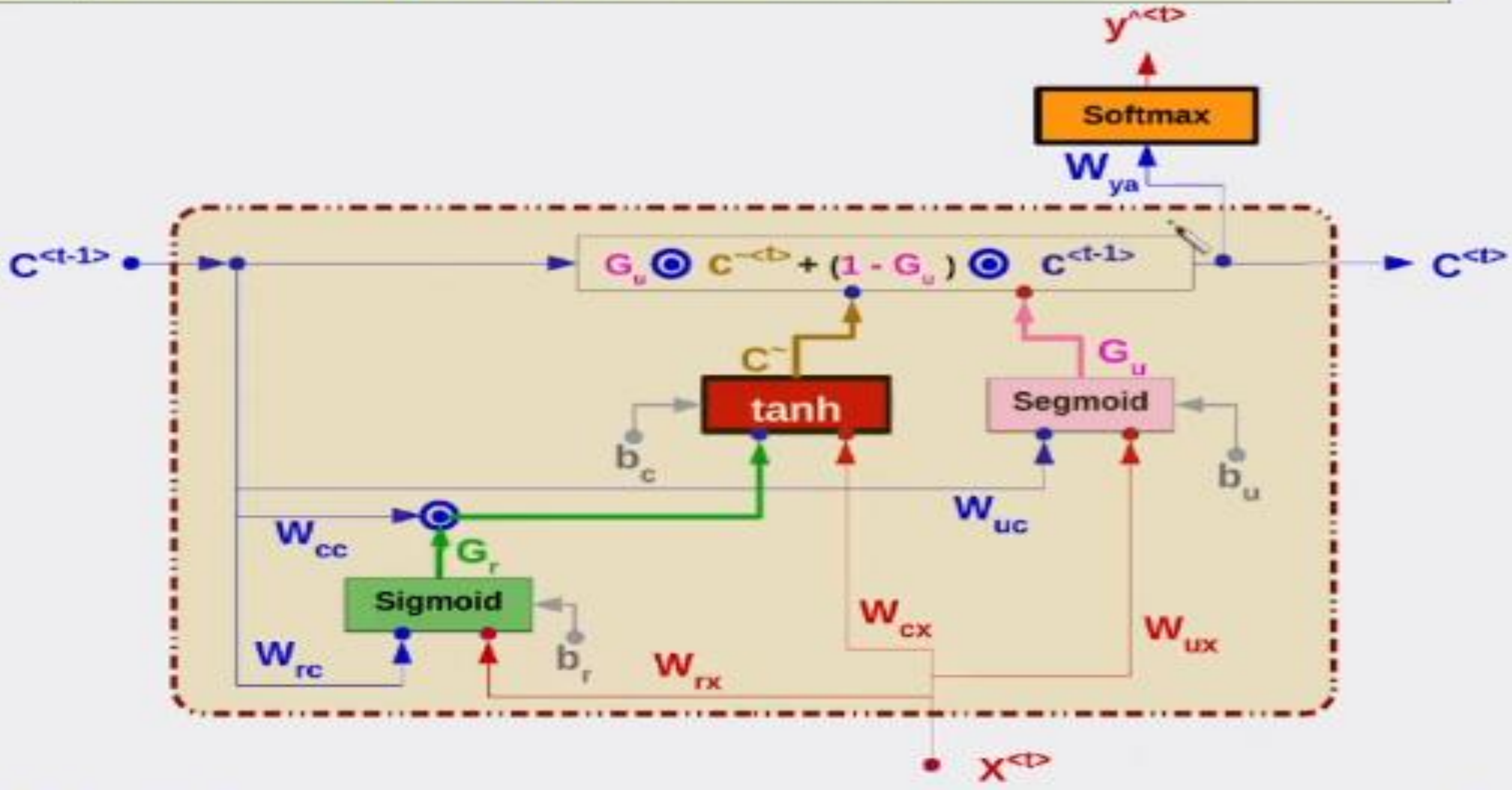
C is the Actual Update



From RNN to GRU

[2] Adding <u>Relevance</u> Gate G_r

If $G_r = 1$, $C^{<t-1>}$ is **Relevant** to update Candidate Memory cell value "$C^{\sim}$"
If $G_r = 0$, $C^{<t-1>}$ is **IrRelevant** to update Candidate Memory cell value "$C^{\sim}$"



From RNN to GRU

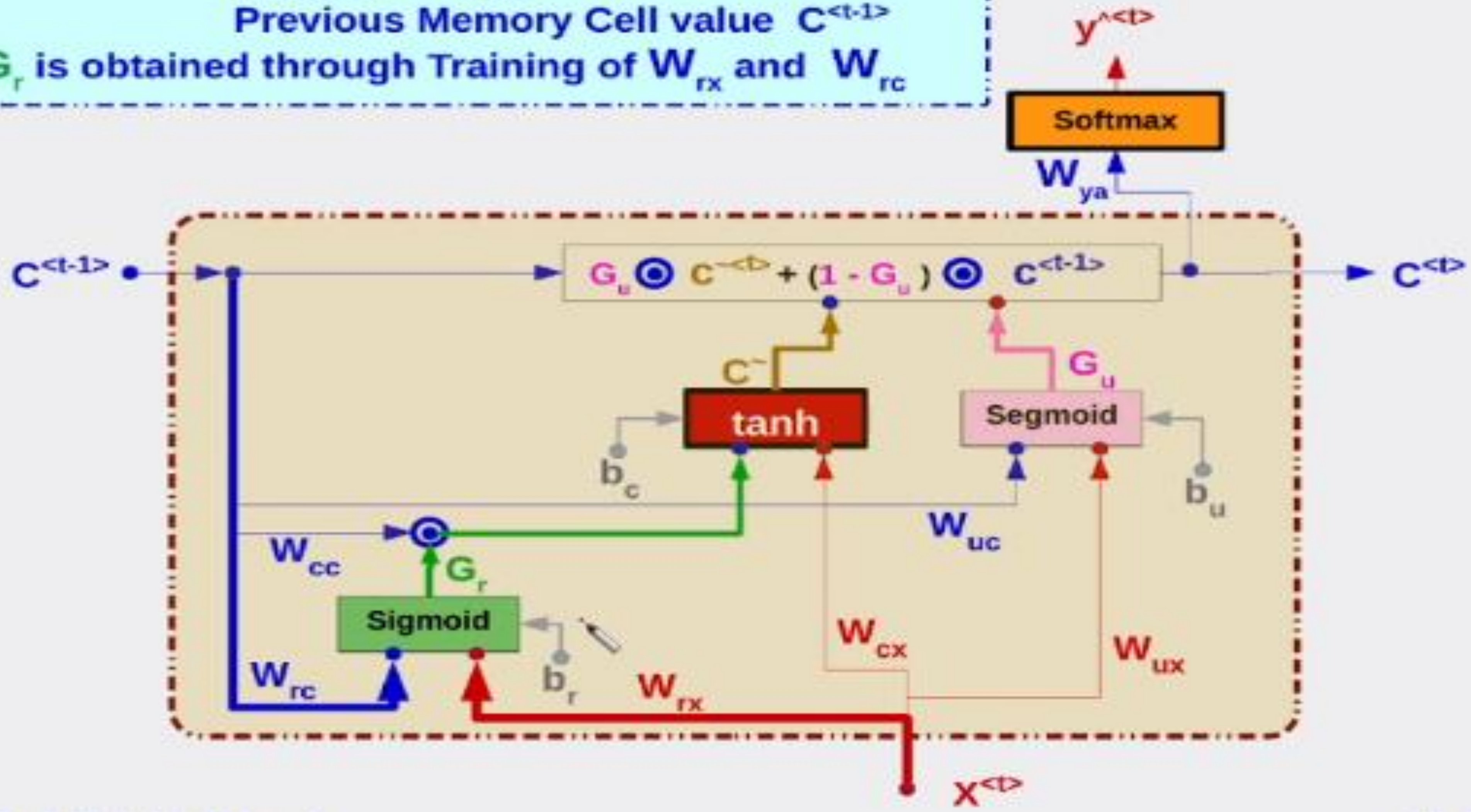
[2] Adding Relevance Gate G_r

Value of G_r is based on:

Input $X^{<t>}$ and

Previous Memory Cell value $C^{<t-1>}$

G_r is obtained through Training of W_{rx} and W_{rc}



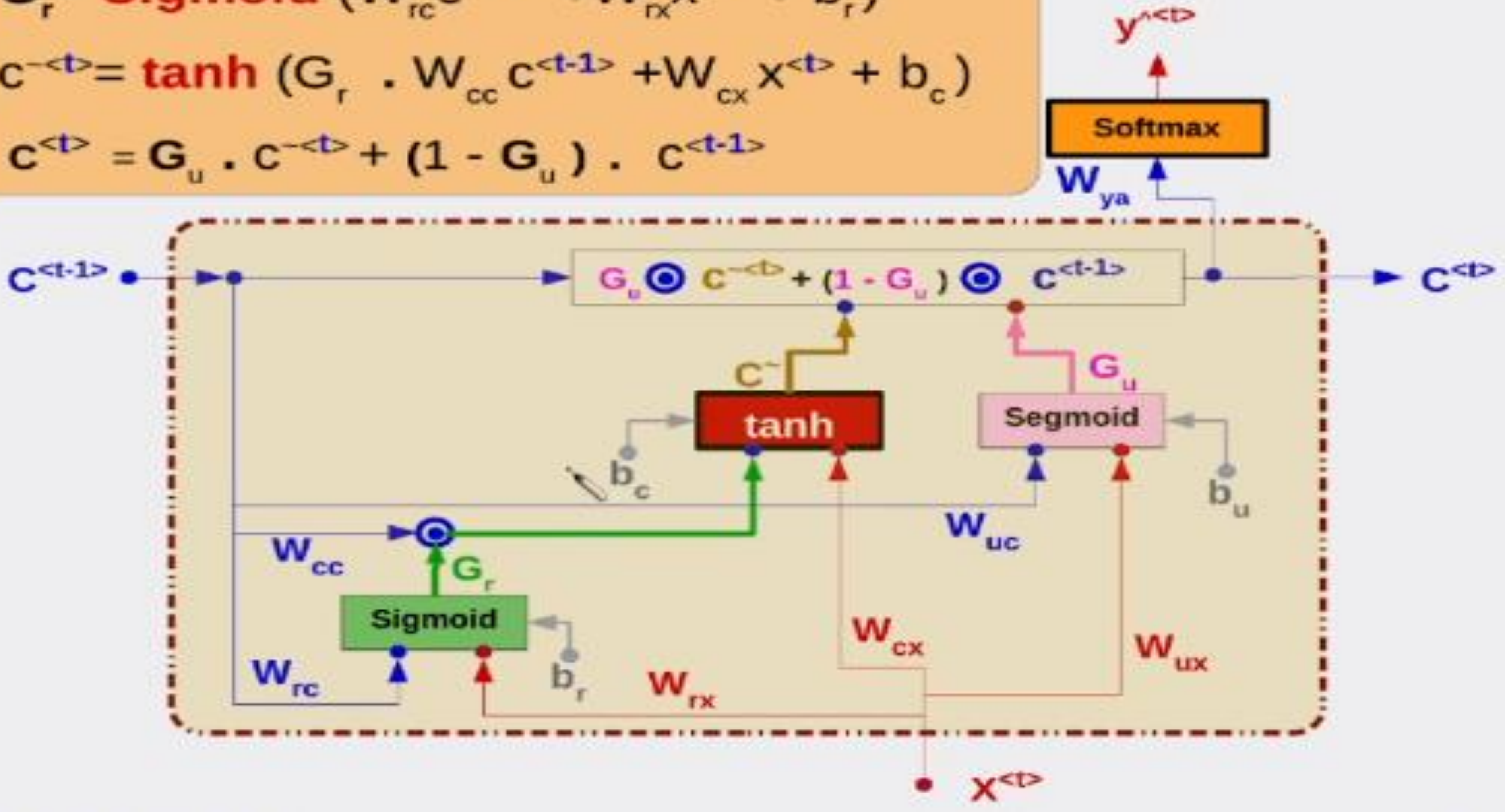
From RNN to GRU

$G_u = \textbf{Sigmoid}\ (W_{uc}c^{<t-1>} + W_{ux}x^{<t>} + b_u)$

$G_r = \textbf{Sigmoid}\ (W_{rc}c^{<t-1>} + W_{rx}x^{<t>} + b_r)$

$c^{\sim<t>} = \textbf{tanh}\ (G_r\ .\ W_{cc}\,c^{<t-1>} + W_{cx}\,x^{<t>} + b_c)$

$c^{<t>} = G_u\ .\ c^{\sim<t>} + (1 - G_u)\ .\ c^{<t-1>}$



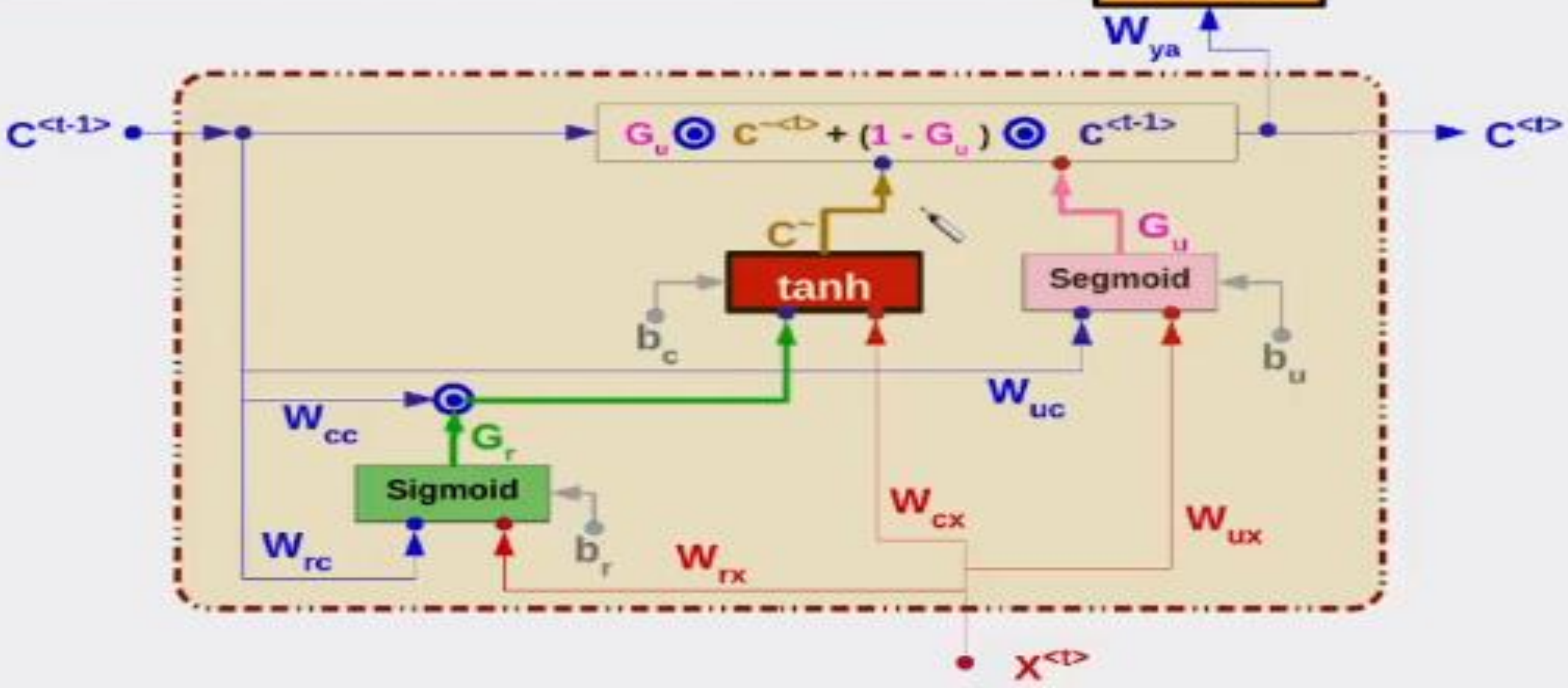
From RNN to GRU

IF

$G_r = 0$ ($C^{<t-1>}$ Does **NOT** Affect tanh Output $\tilde{C}$)

$G_u = 1$ ($C^{<t-1>}$ Does **NOT** Affect Final Output $C^{<t>}$)

>> >> Traditional **NN** (No Recurrency)



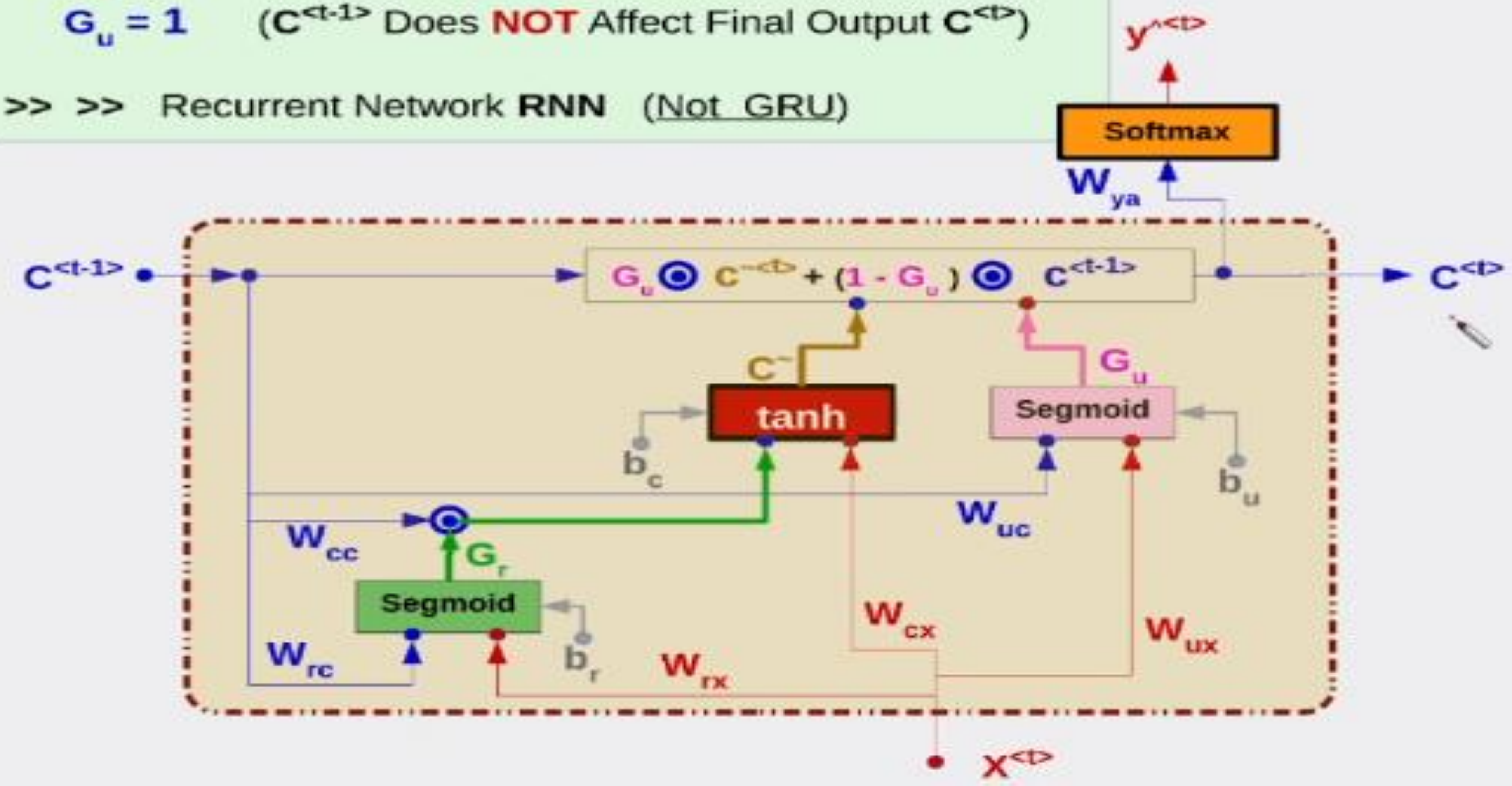
From RNN to GRU

IF

$G_r = 1$ ($C^{<t-1>}$ **<u>Affects</u>** tanh Output $\tilde{C}$)

$G_u = 1$ ($C^{<t-1>}$ Does **NOT** Affect Final Output $C^{<t>}$)

>> >> Recurrent Network **RNN** (<u>Not GRU</u>)



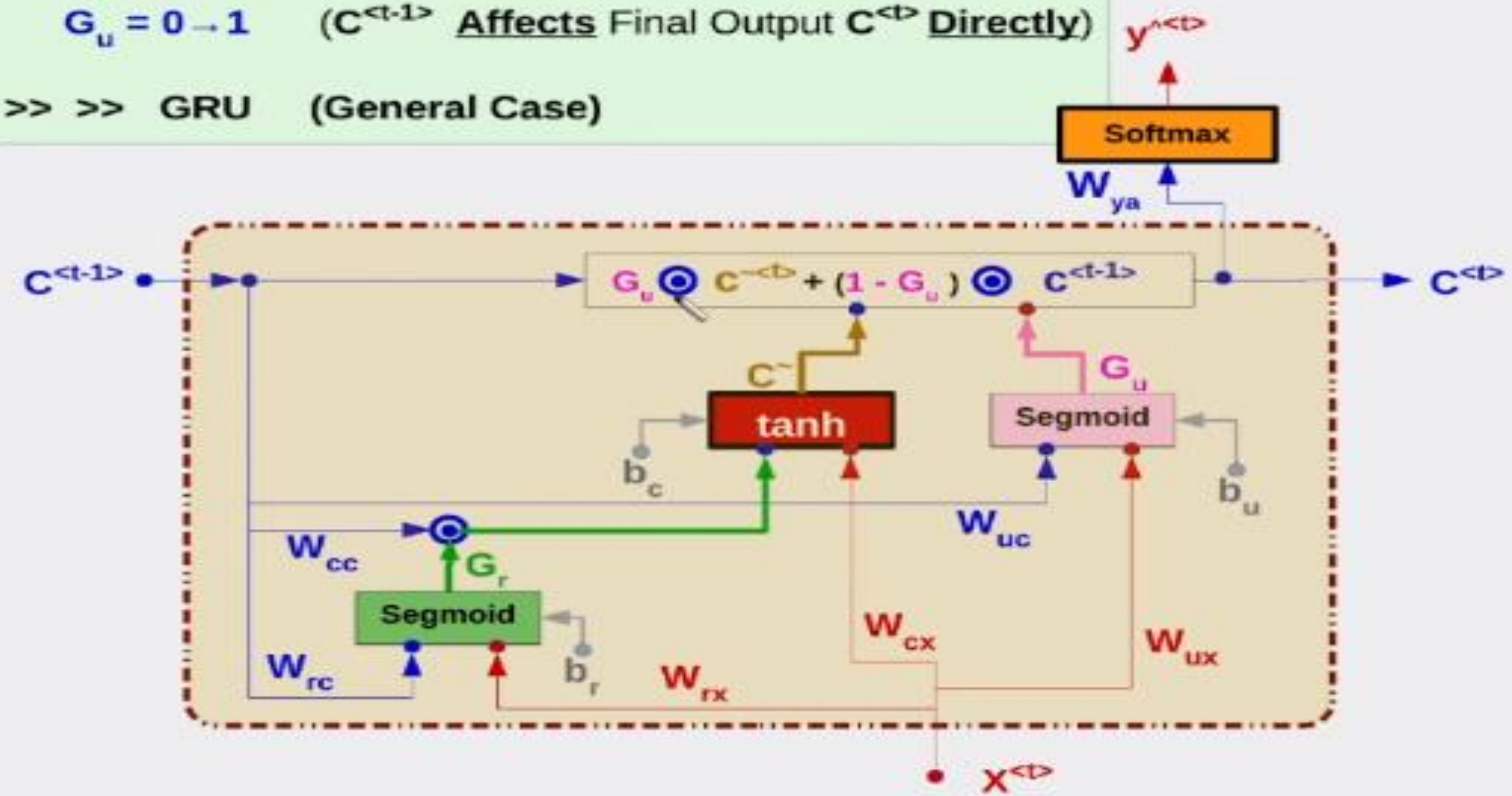
From RNN to GRU

IF

$G_r = 0 \rightarrow 1$ ($C^{<t-1>}$ **Affects** tanh Output $C^{\sim}$)

$G_u = 0 \rightarrow 1$ ($C^{<t-1>}$ **Affects** Final Output $C^{<t>}$ **Directly**)

>> >> GRU (General Case)

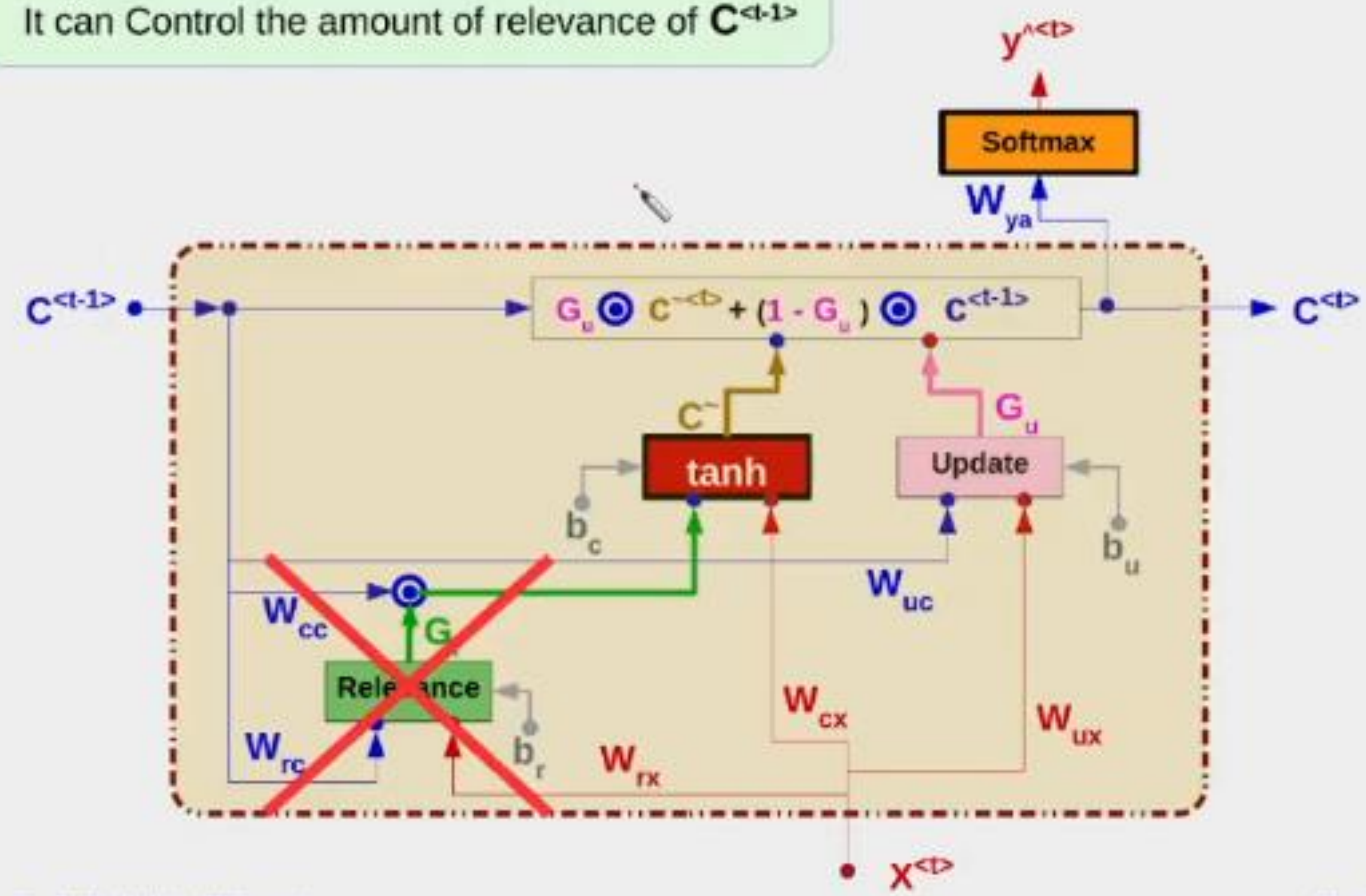


[3] Long Short Term Memory (LSTM)

From GRU to LSTM

[1] Removing **Relevance Gate** G_r

W_{uc} can Do the same functionality
It can Control the amount of relevance of $C^{<t-1>}$



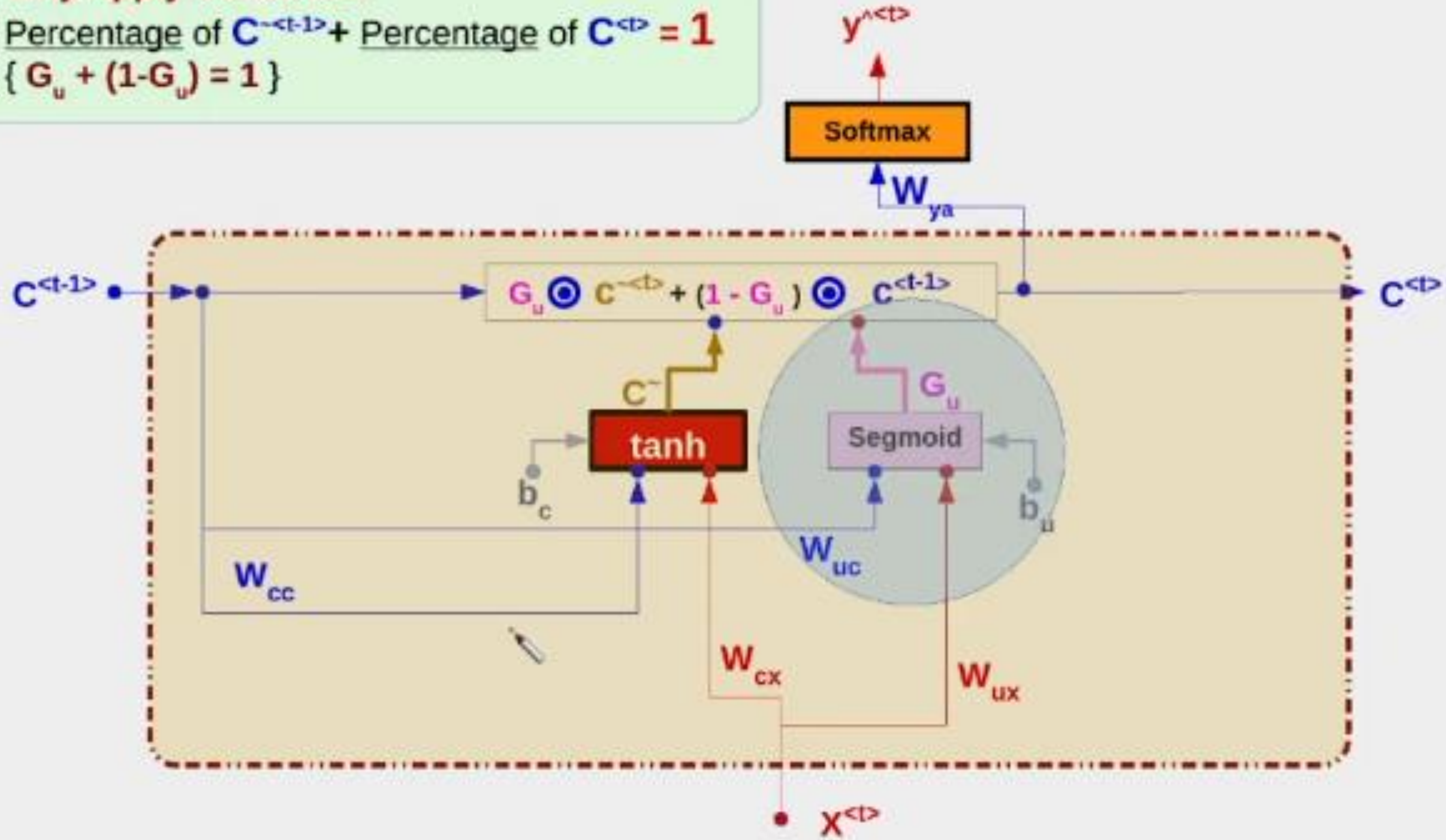
From GRU to LSTM

[2] ??????

Why Apply Constrain:

Percentage of $C^{\sim<t-1>}$ + Percentage of $C^{<t>} = 1$

$\{ G_u + (1-G_u) = 1 \}$



From GRU to LSTM

[2] ??????

Why Apply Constrain:

Percentage of $C^{\sim<t-1>}$ + Percentage of $C^{<t>}$ = 1

{ $G_u + (1-G_u) = 1$ }

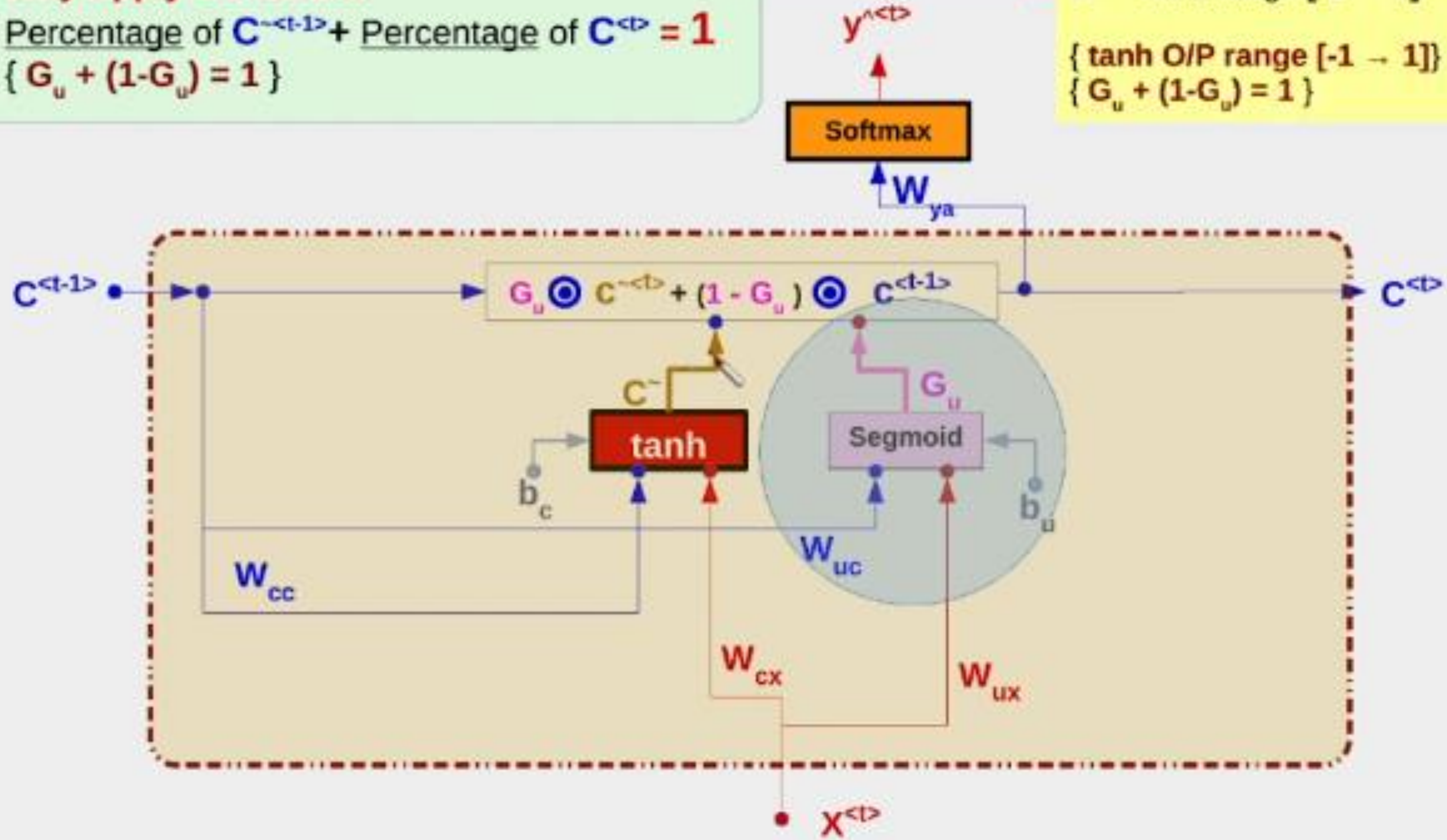
Answer

$C^{\sim}$ value range [-1 → 1]

C value range [-1 → 1]

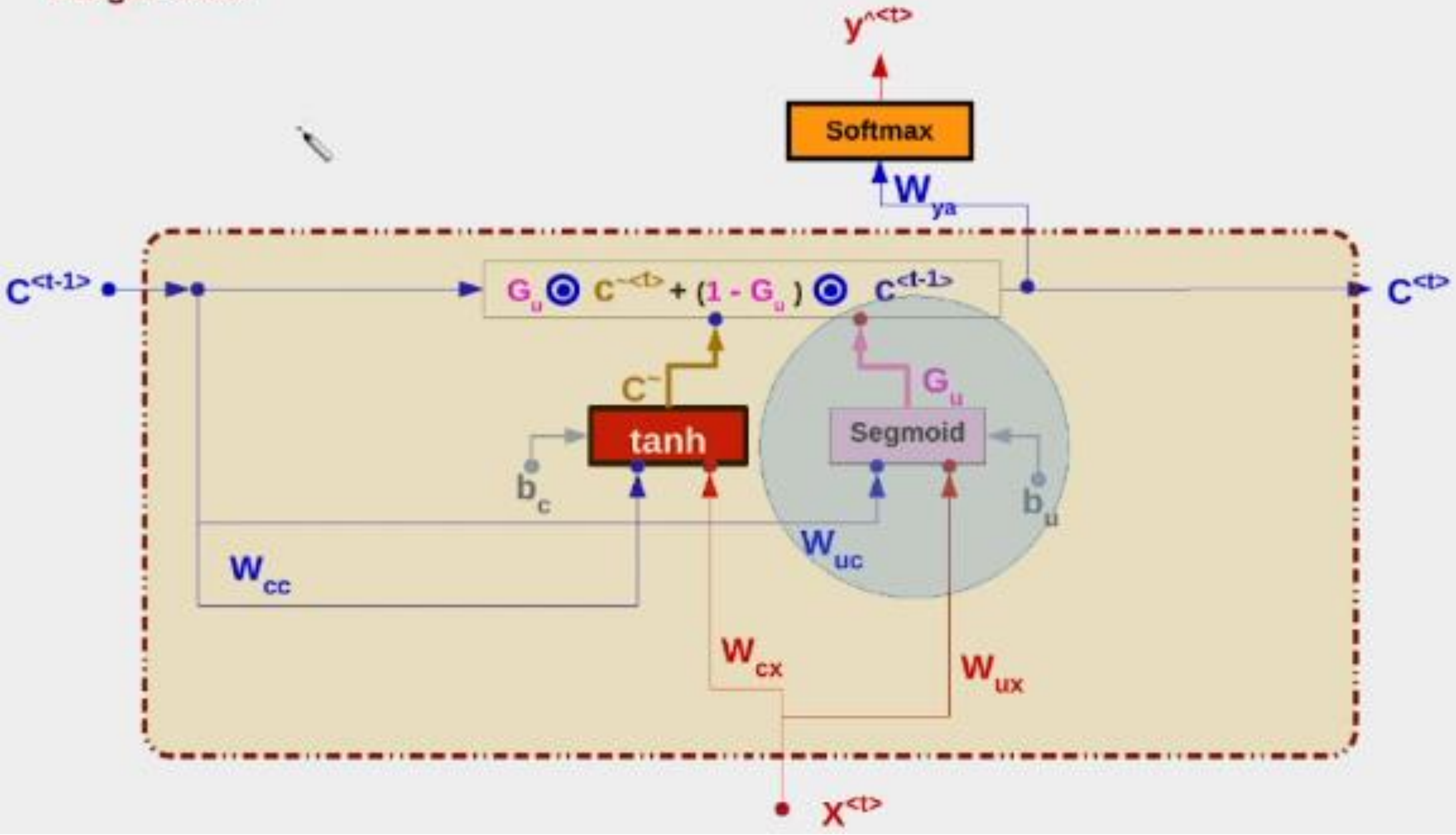
{ tanh O/P range [-1 → 1]}

{ $G_u + (1-G_u) = 1$ }



From GRU to LSTM

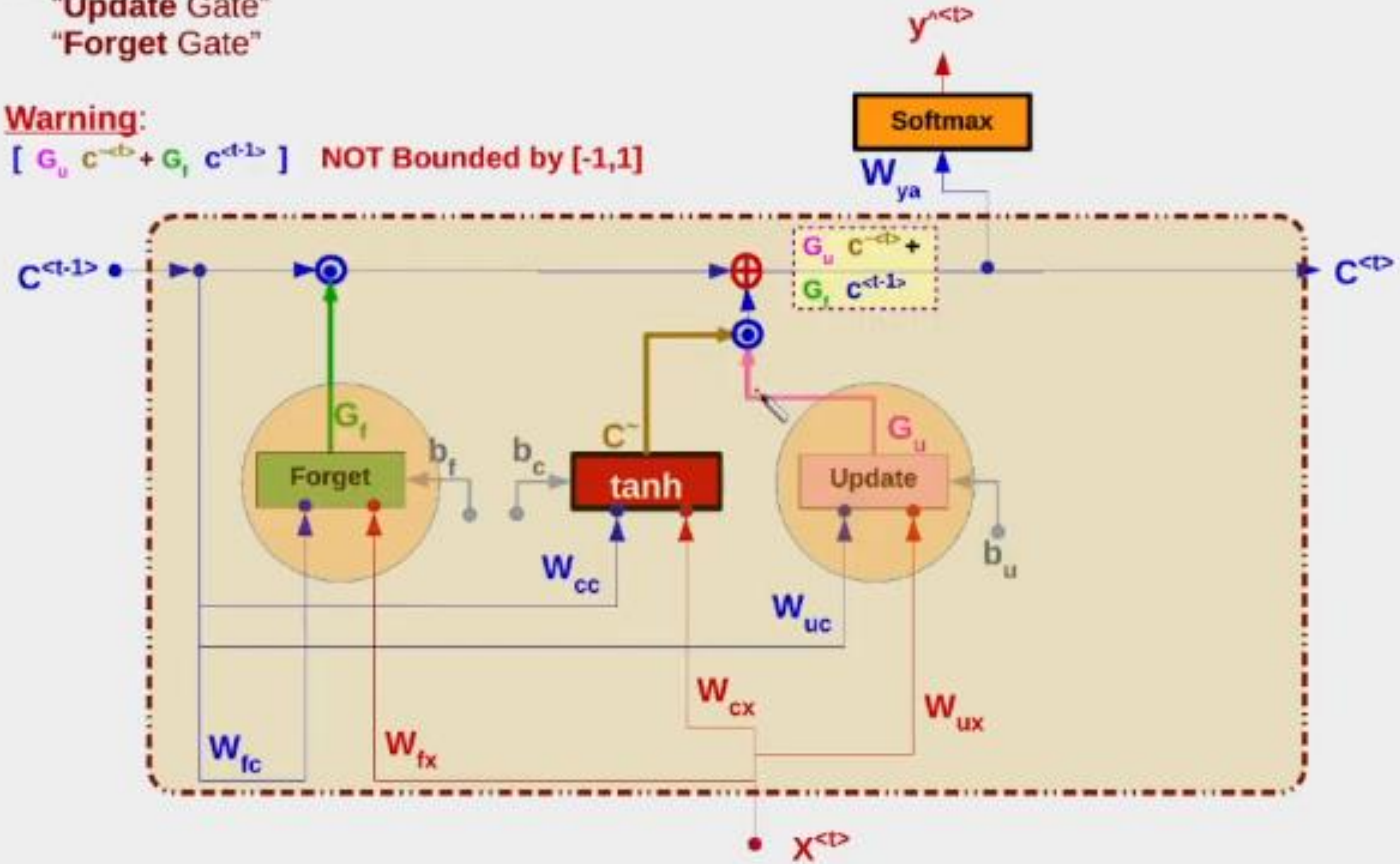
[2] Split "**Update** Gate" into two gates:
"**Update** Gate"
"**Forget** Gate"



From GRU to LSTM

[2] Split "**Update** Gate" into two gates:
"**Update** Gate"
"**Forget** Gate"

Warning:
[G_u $c^{\sim<t>}$ + G_f $c^{<t-1>}$] **NOT Bounded by [-1,1]**

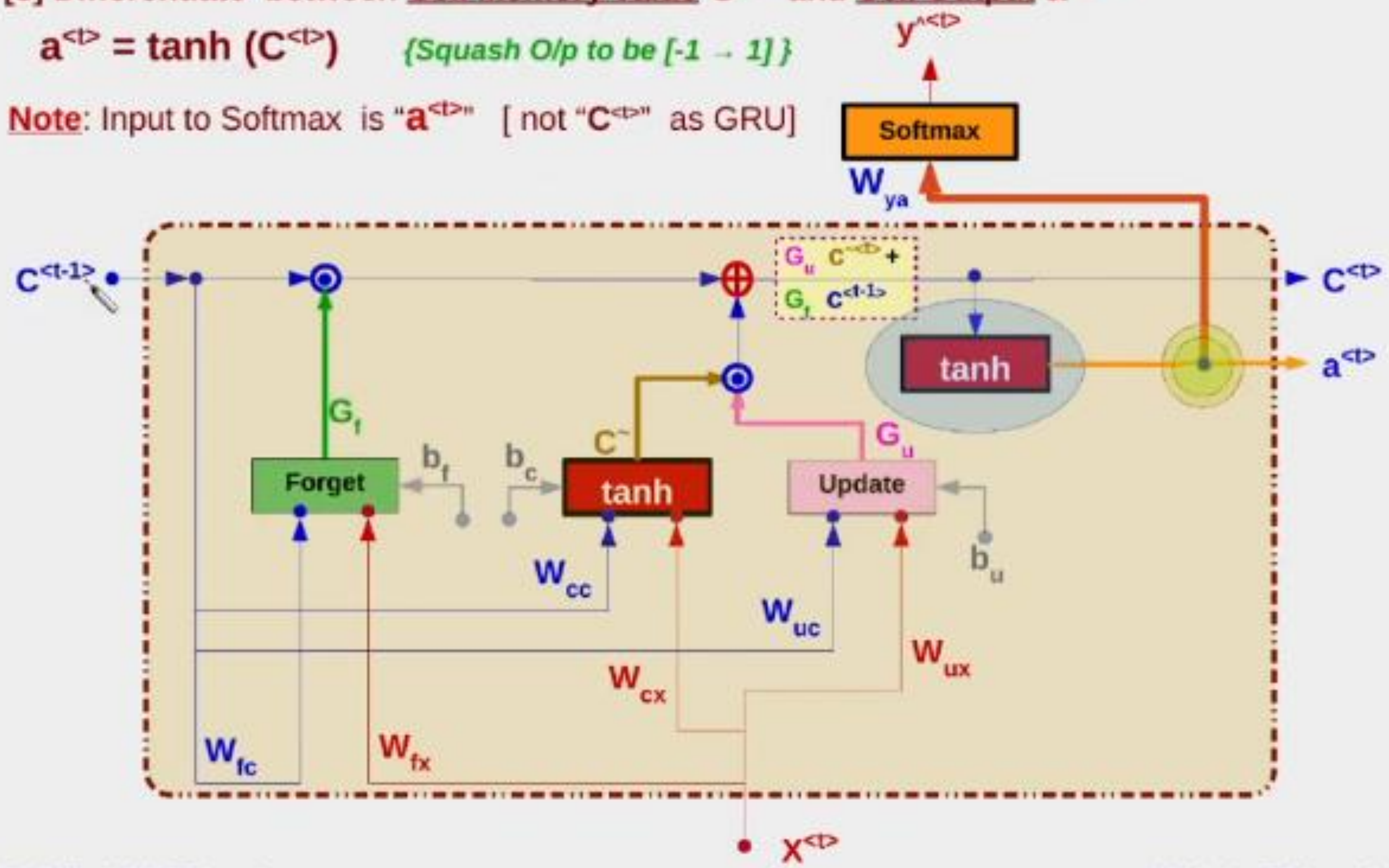


From GRU to LSTM

[3] Differentiate between <u>Cell Memory value</u> $C^{<t>}$ and <u>Cell Output</u> $a^{<t>}$

$a^{<t>} = \tanh(C^{<t>})$ *{Squash O/p to be [-1 → 1] }*

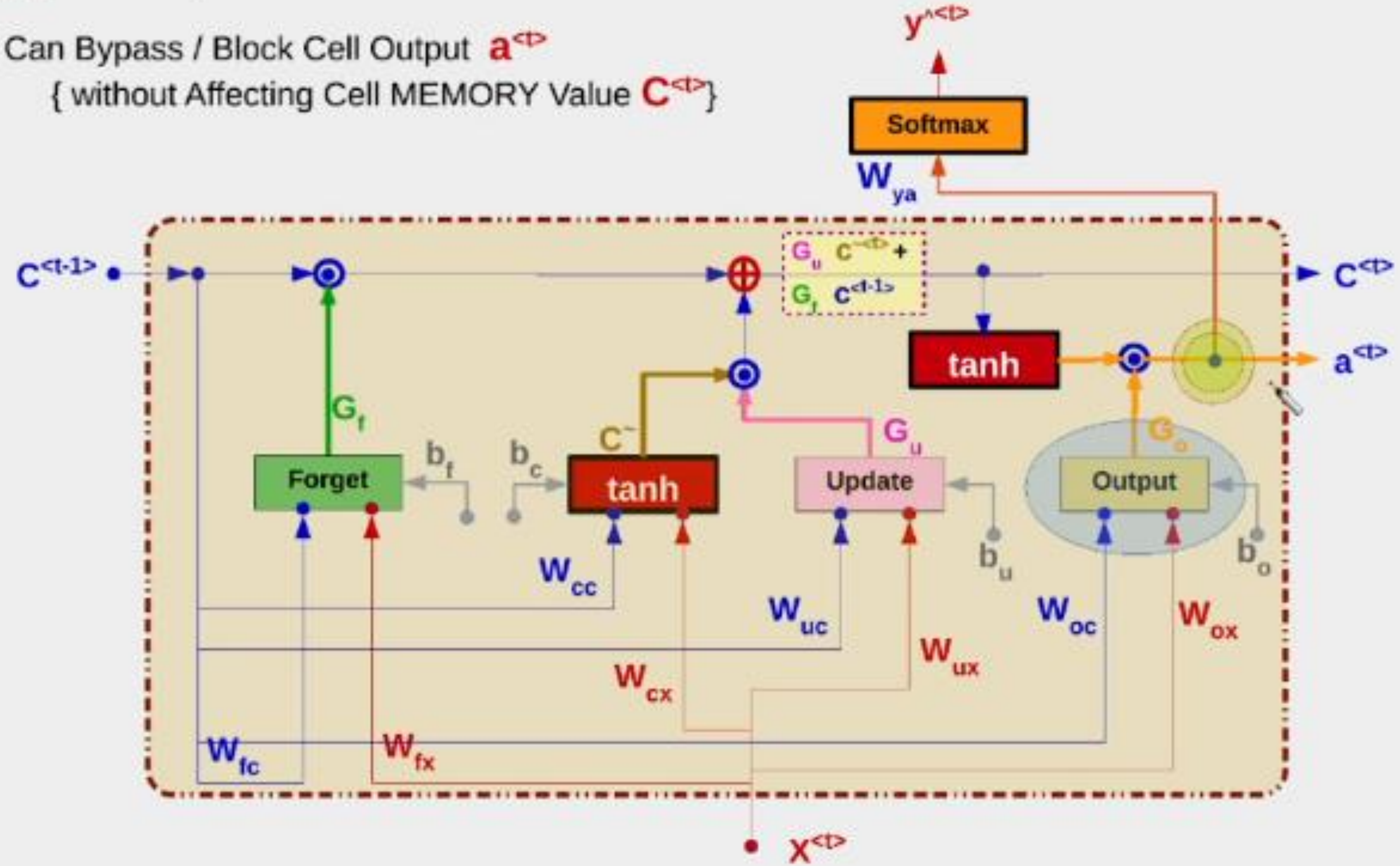
<u>Note</u>: Input to Softmax is "$a^{<t>}$" [not "$C^{<t>}$" as GRU]



From GRU to LSTM

[4] Add "**Output** Control Gate"

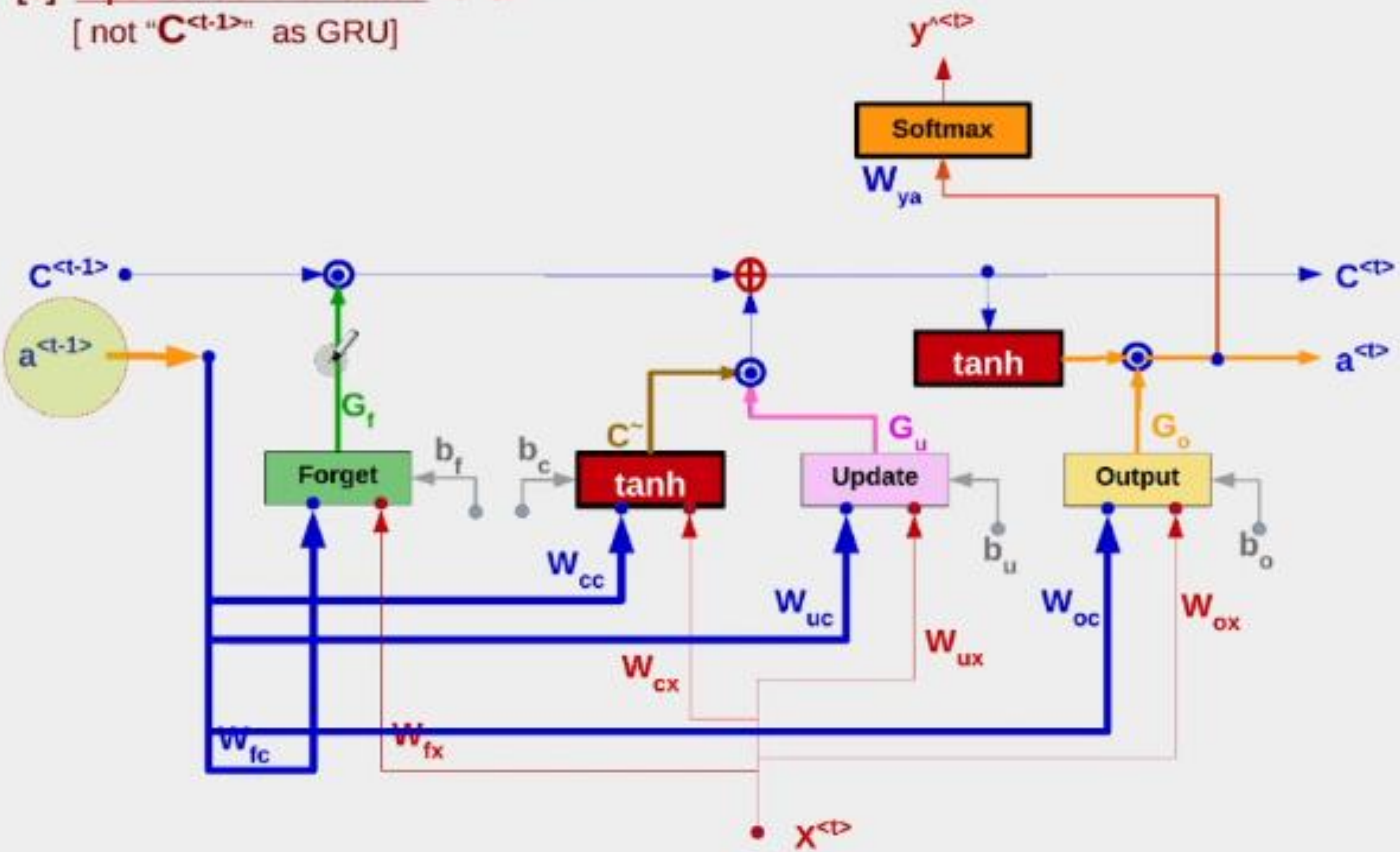
Can Bypass / Block Cell Output $a^{<t>}$
{ without Affecting Cell MEMORY Value $C^{<t>}$}



From GRU to LSTM

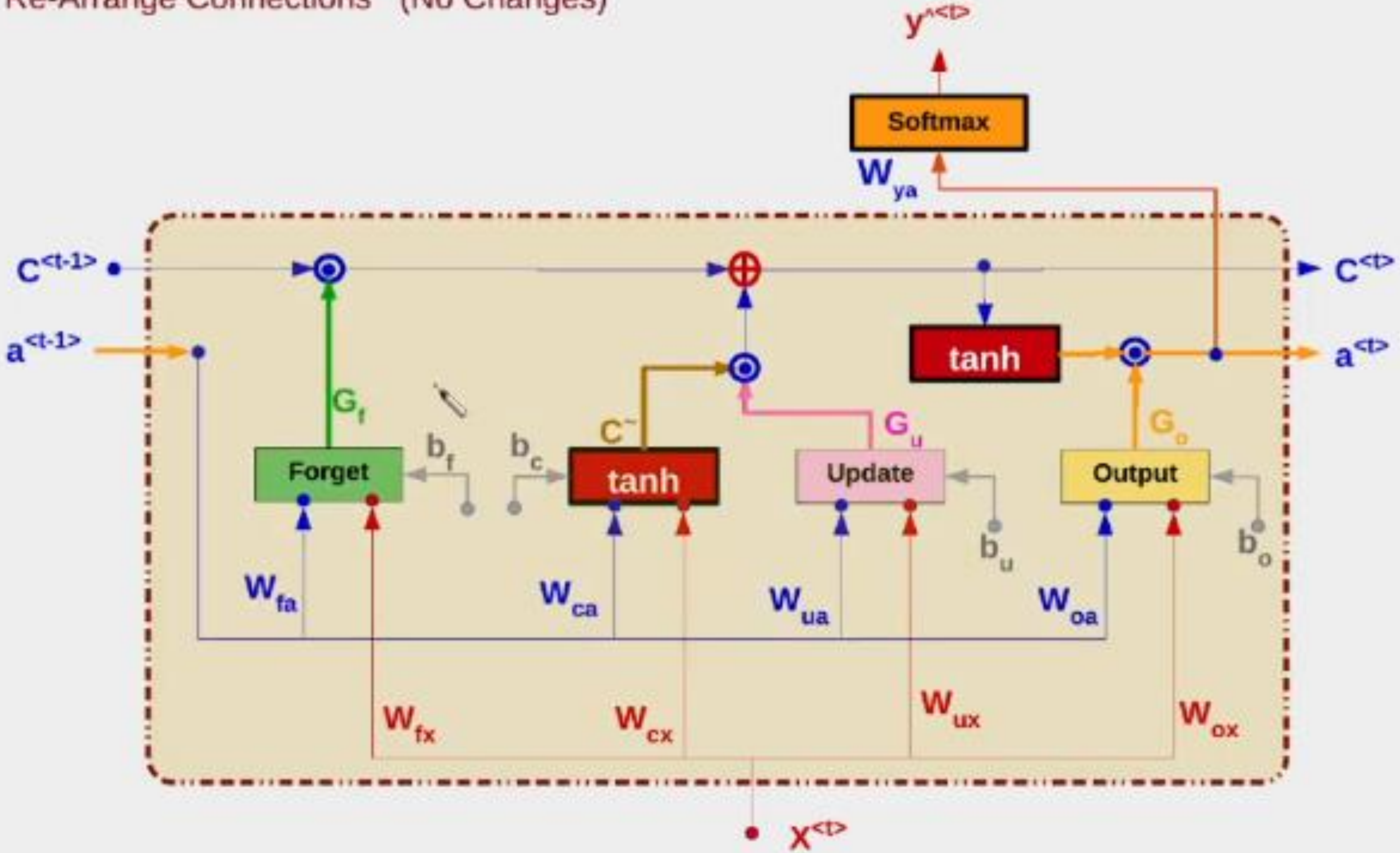
[5] **Input to Control Gates** is "$a^{<t-1>}$"

[not "$C^{<t-1>}$" as GRU]



LSTM Unit

Re-Arrange Connections (No Changes)



LSTM Unit

Gates

G_u = Sigmoid $(W_{ua} a^{<t-1>} + W_{ux} x^{<t>} + b_u)$

G_f = Sigmoid $(W_{fa} a^{<t-1>} + W_{fx} x^{<t>} + b_f)$

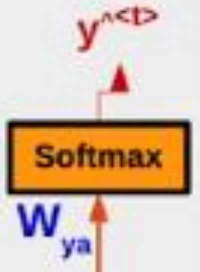
G_o = Sigmoid $(W_{oa} a^{<t-1>} + W_{ox} x^{<t>} + b_o)$

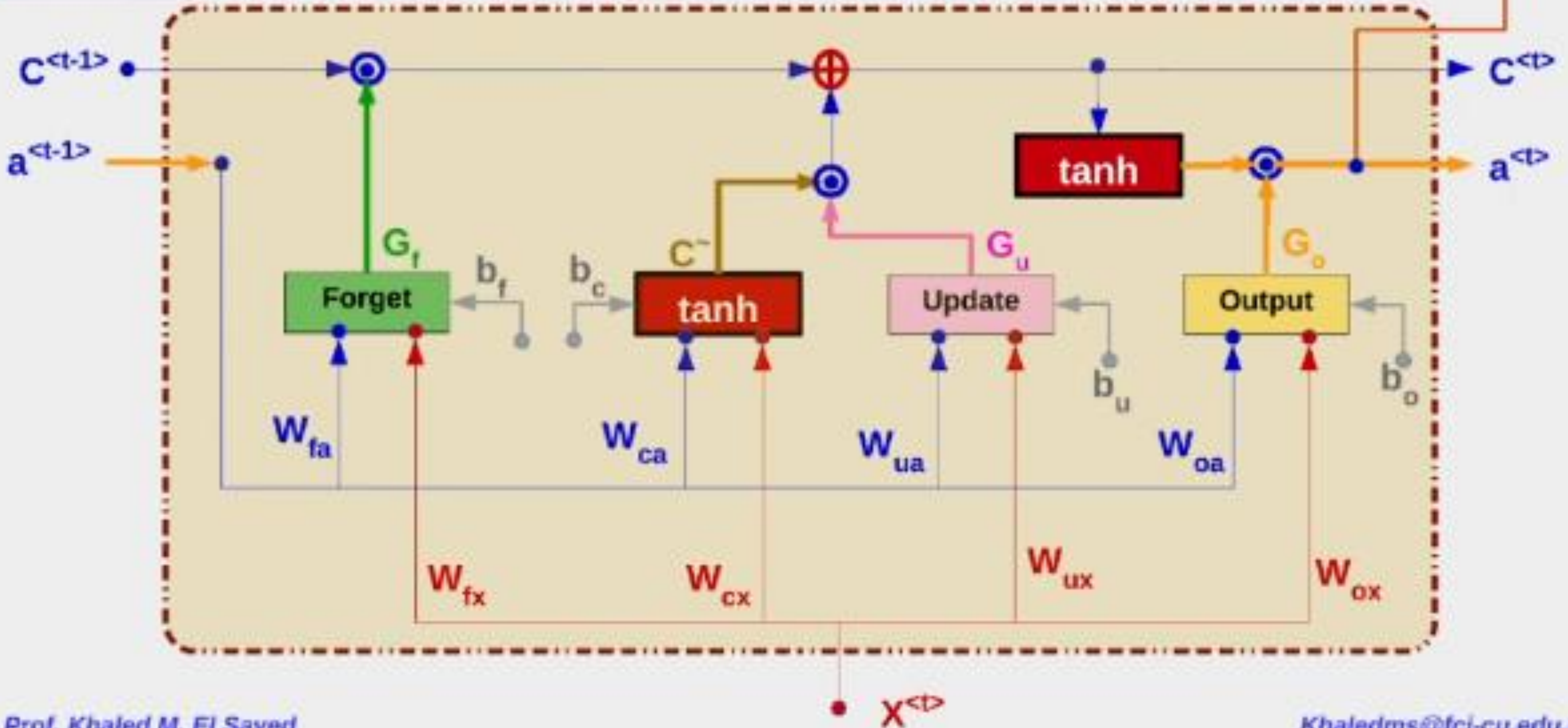
Outputs

$c^{\sim<t>}$ = tanh $(W_{ca} a^{<t-1>} + W_{cx} x^{<t>} + b_c)$

$c^{<t>} = G_u \,.\, c^{\sim<t>} + G_f \,.\, c^{<t-1>}$

$a^{<t>} = G_o \,.\,$ tanh $(c^{<t>})$





By Prof. Khaled M. El Sayed
Khaledms@fci-cu.edu.eg

Input Sequence to LSTM