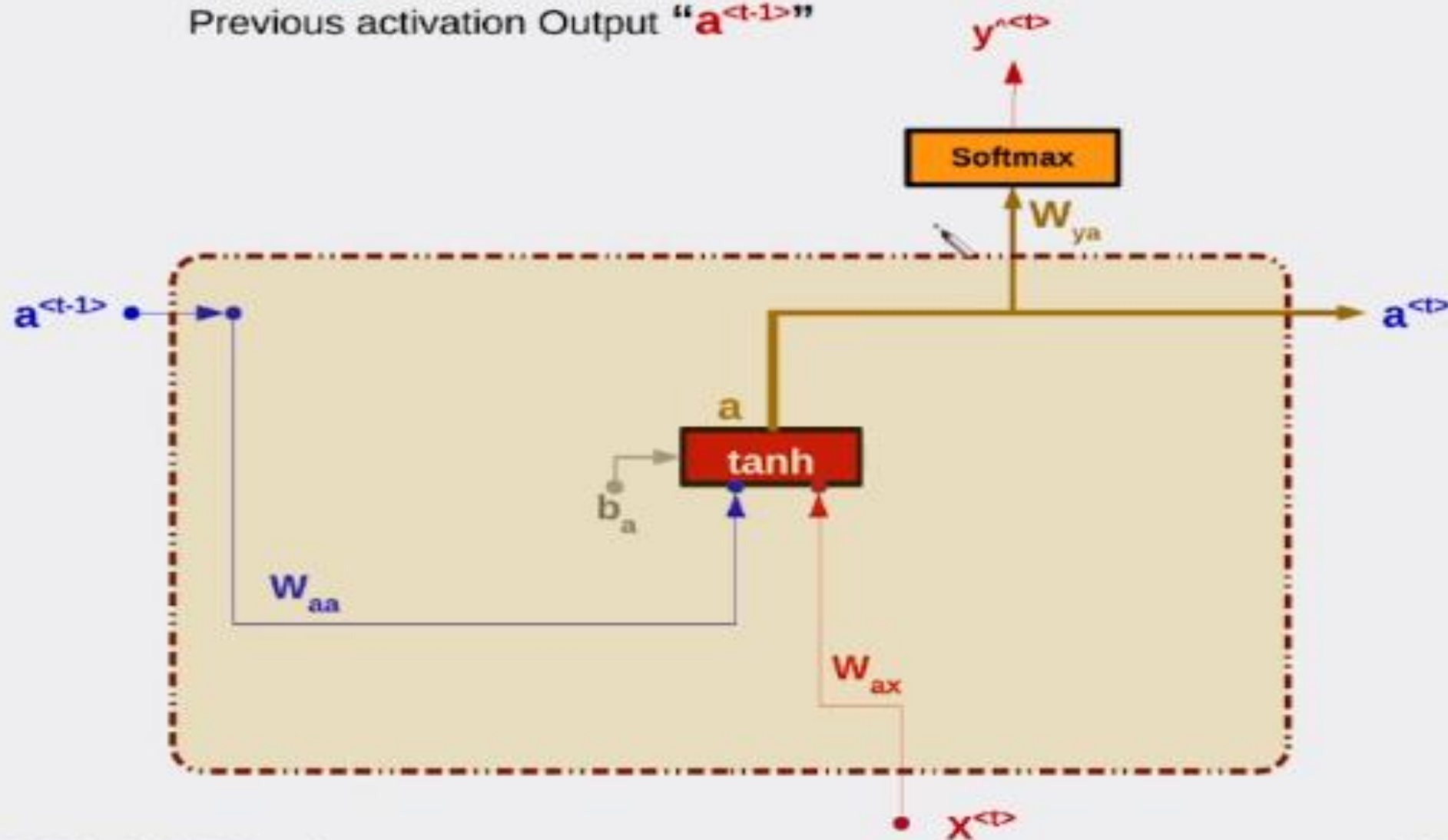


+

[1] Recurrent Neural Network (RNN)

Recurrent Neural Network (RNN)

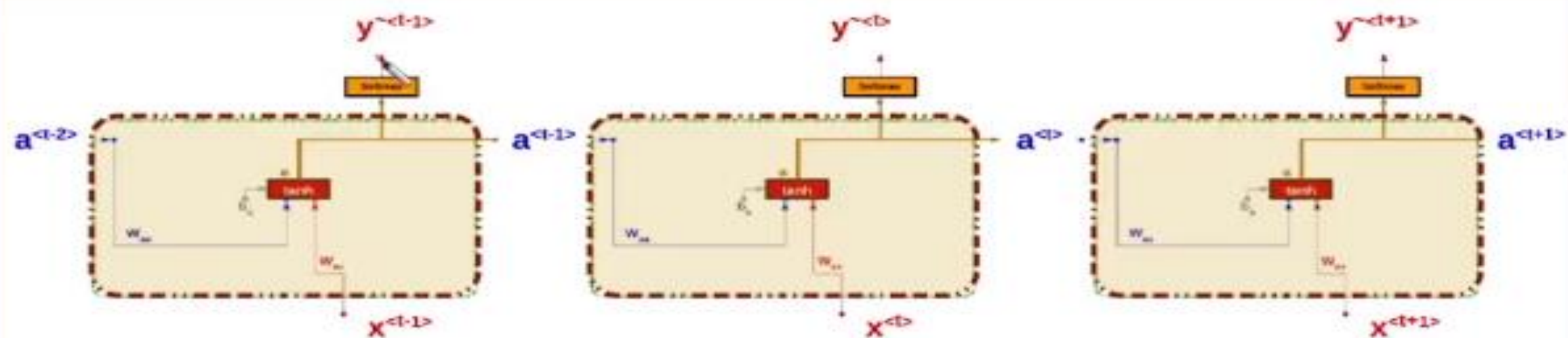
Output of Activation Function at time t ; " $a^{(t)}$ " depends on BOTH:-
Input " $x^{(t)}$ " and
Previous activation Output " $a^{(t-1)}$ "



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 $t-1, t, t+1$



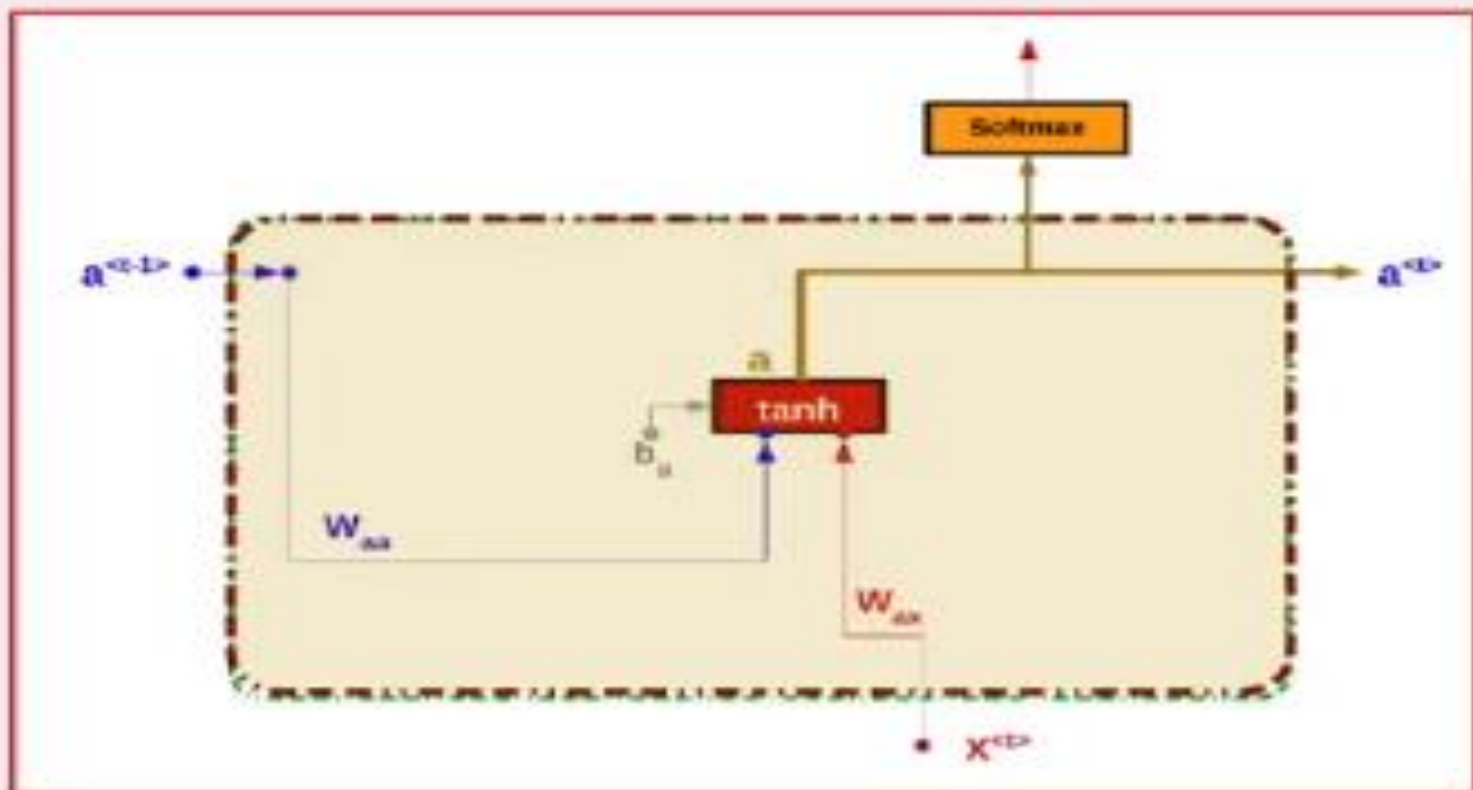
Recurrent Neural Network (RNN)

From Prev. Current I/P

$$a^{<1>} = F^n (W_{aa} a^{<0>} + W_{ax} x^{<1>} + b_a)$$

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

Non Linearity



Recurrent Neural Network (RNN)

From Prev. Current I/P

$$(W_{aa} a^{<0>} + W_{ax} x^{<1>} + b_a)$$

$$(W_{ya} a^{<1>} + b_y)$$

earity

$$\tanh(W_{aa} a^{<0>} + W_{ax} x^{<1>} + b_a)$$

May be tanh, ReLU, ...

$$\text{softmax}(W_{ya} a^{<1>} + b_y)$$

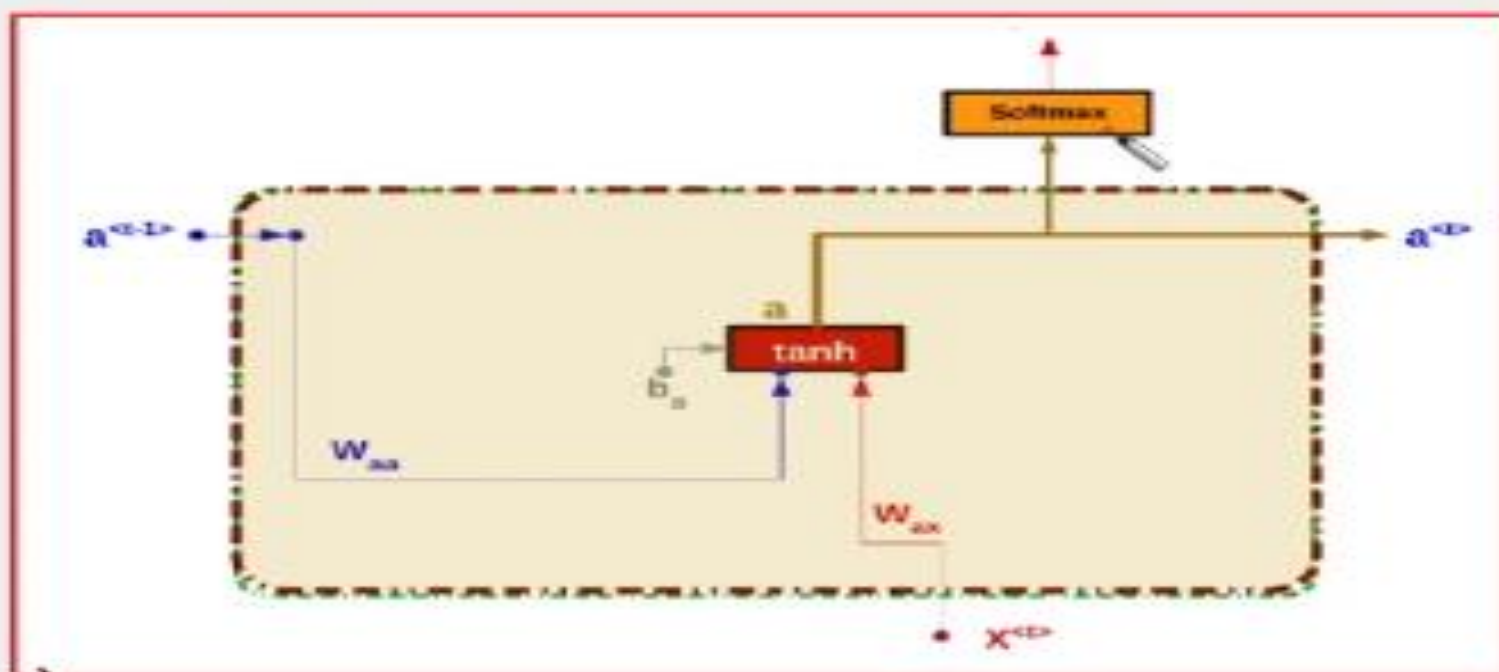
Segmoid for Binary O/P, Softmax for Multi-Class O/P

$$\tanh(W_{aa} a^{<t-1>} + W_{ax} x^{<t>} + b_a)$$

May be tanh, ReLU, ...

$$\text{softmax}(W_{ya} a^{<t>} + b_y)$$

Segmoid for Binary O/P, Softmax for Multi-Class O/P

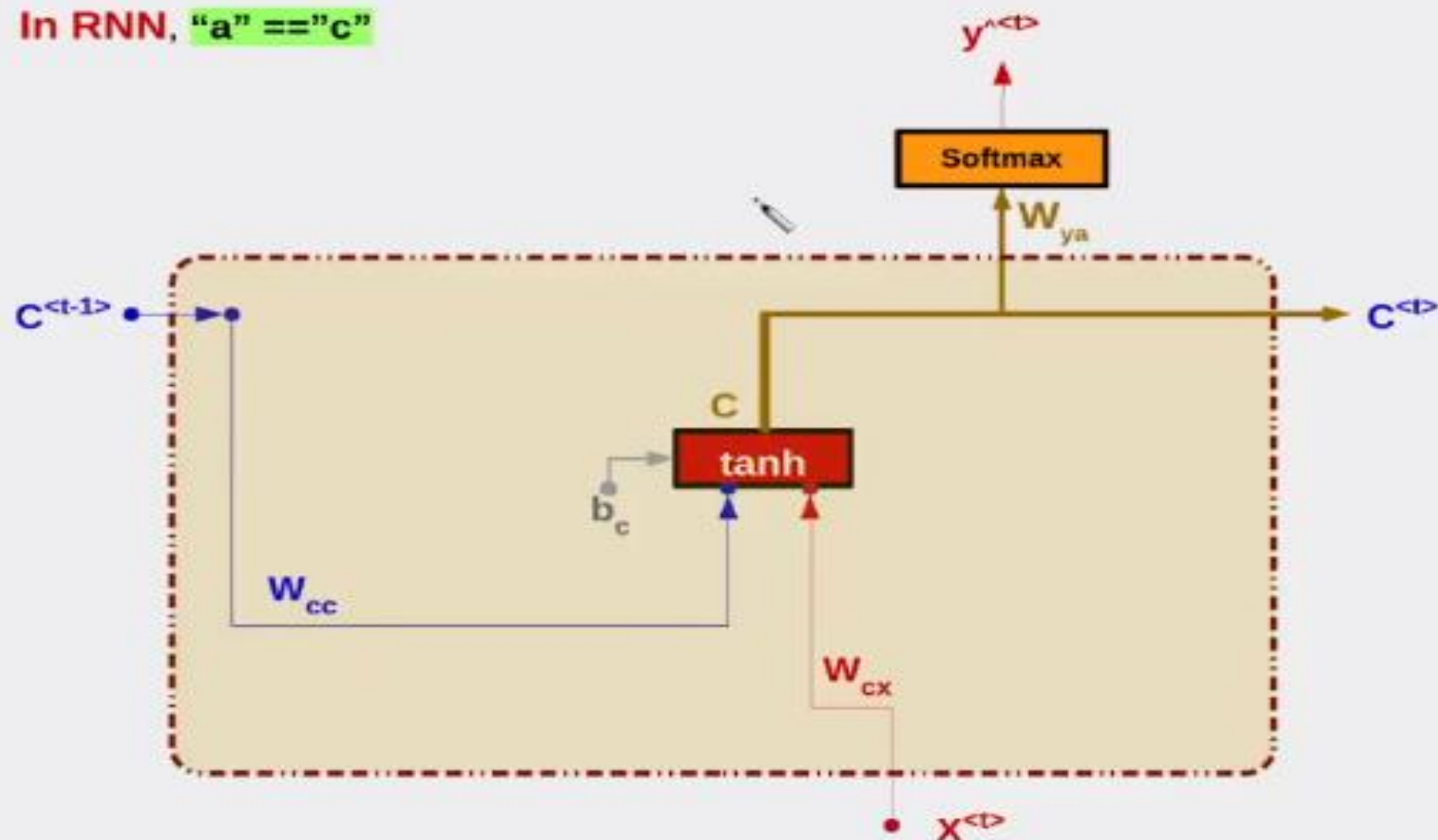


[2] Gated Recurrent Unit (GRU)

From RNN to GRU

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

In RNN, "a" == "c"



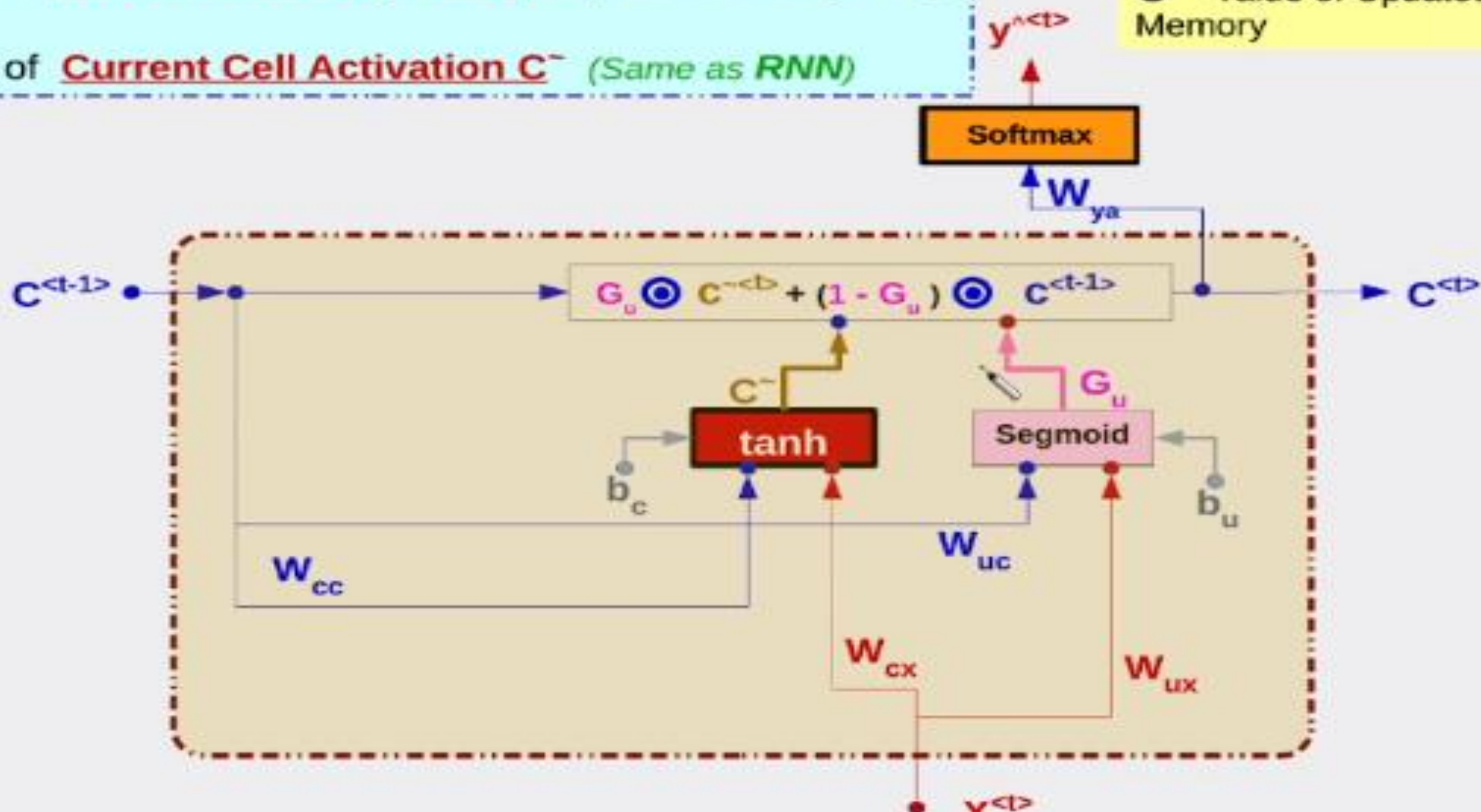
From RNN to GRU

[1] Adding Update Gate G_u

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

C^- 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:

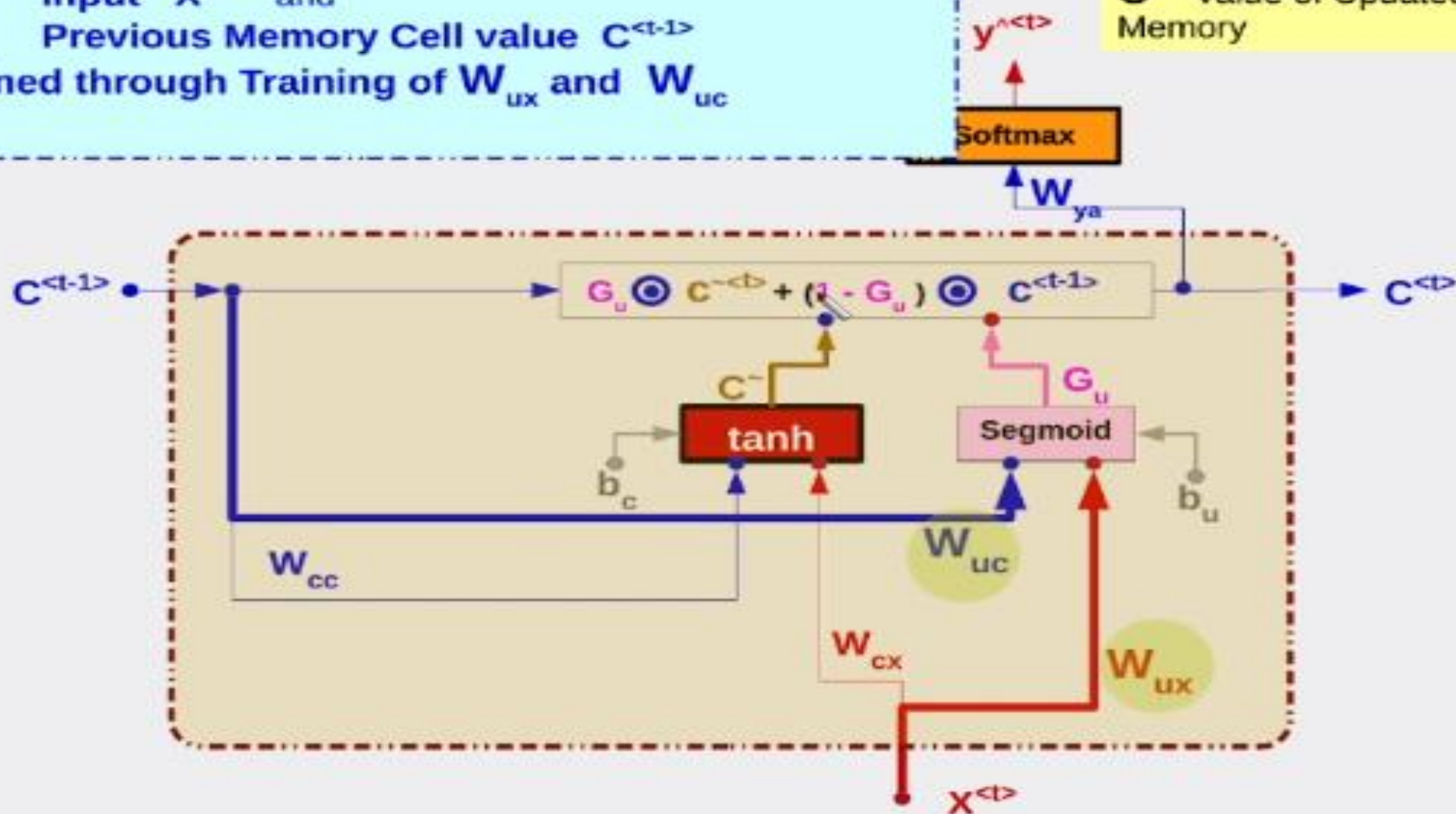
Input $X^{<t>}$ and

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

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

C^- Candidate Value of Updated Memory

C Value of Updated Memory



From RNN to GRU

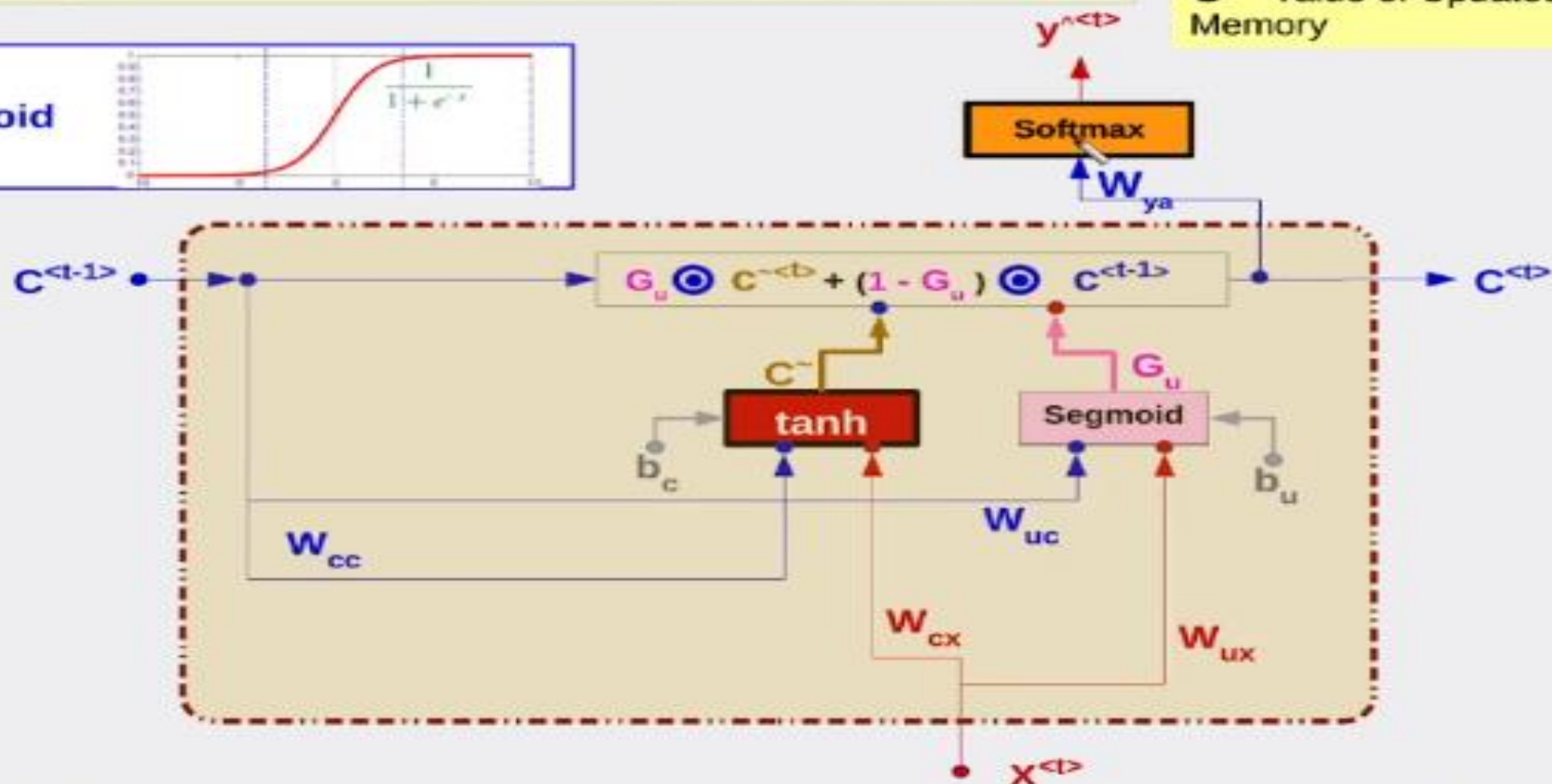
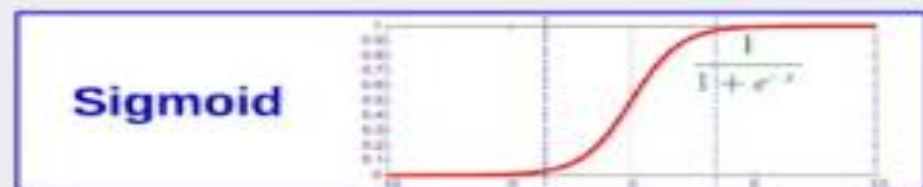
[1] Adding Update Gate G_u

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

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

C^- Candidate Value of Updated Memory

C Value of Updated Memory



From RNN to GRU

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

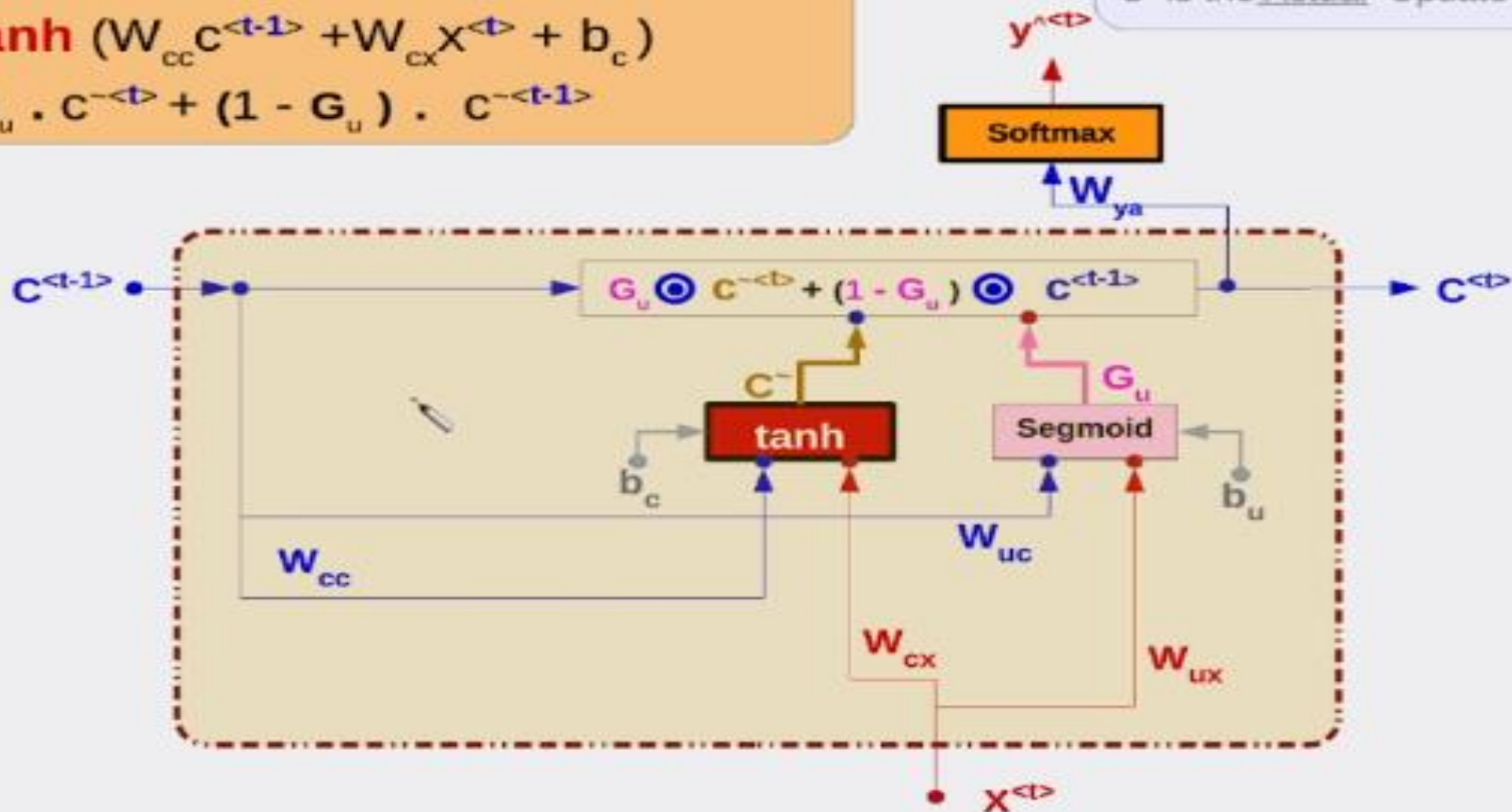
$$c^{-<t>} = \tanh(W_{cc}c^{<t-1>} + W_{cx}x^{<t>} + b_c)$$

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

c^{-} is the Candidate Update

G_u is the Update Gate

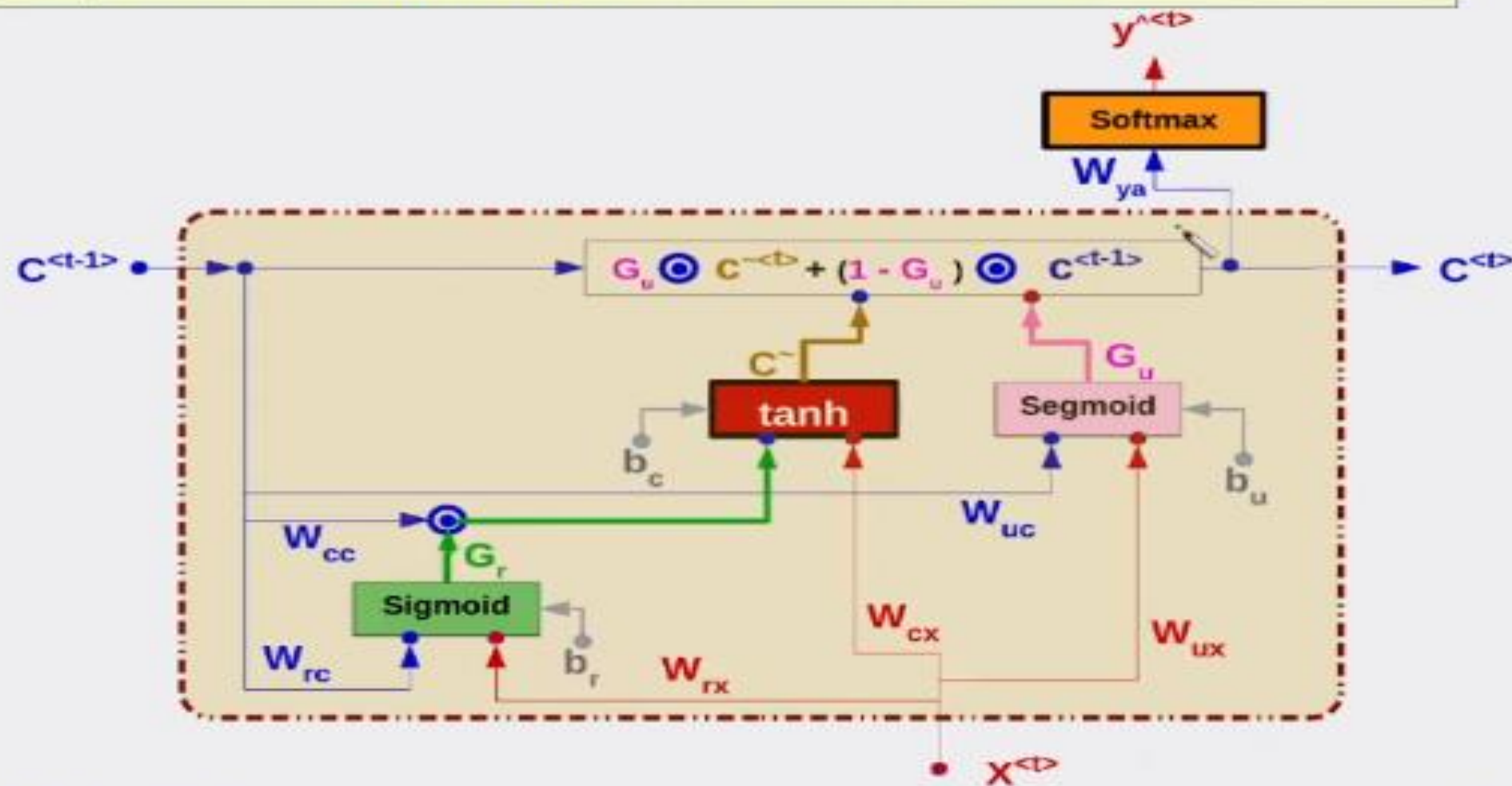
C is the Actual Update



From RNN to GRU

[2] Adding Relevance 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

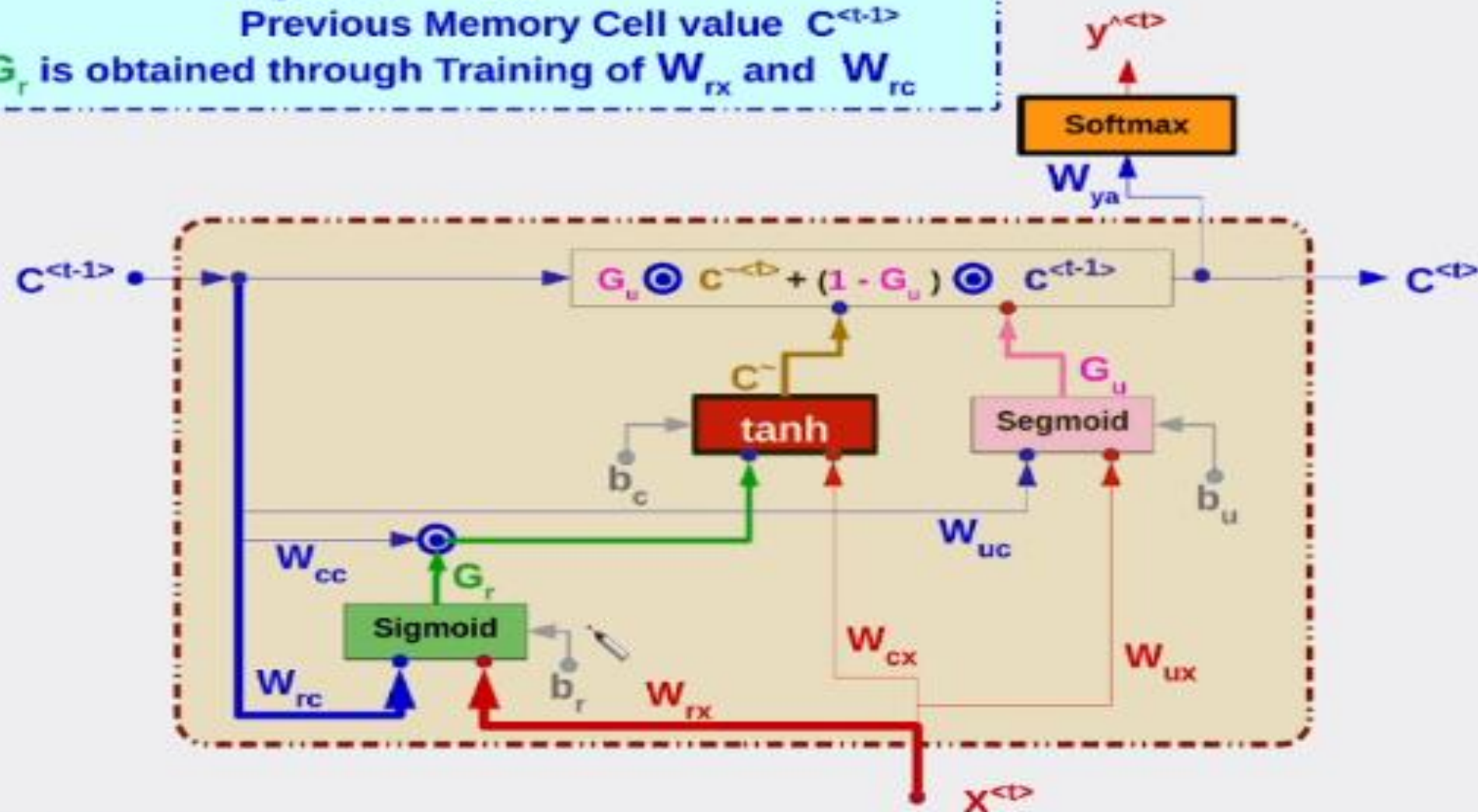
[2] Adding Relevance Gate G_r

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}



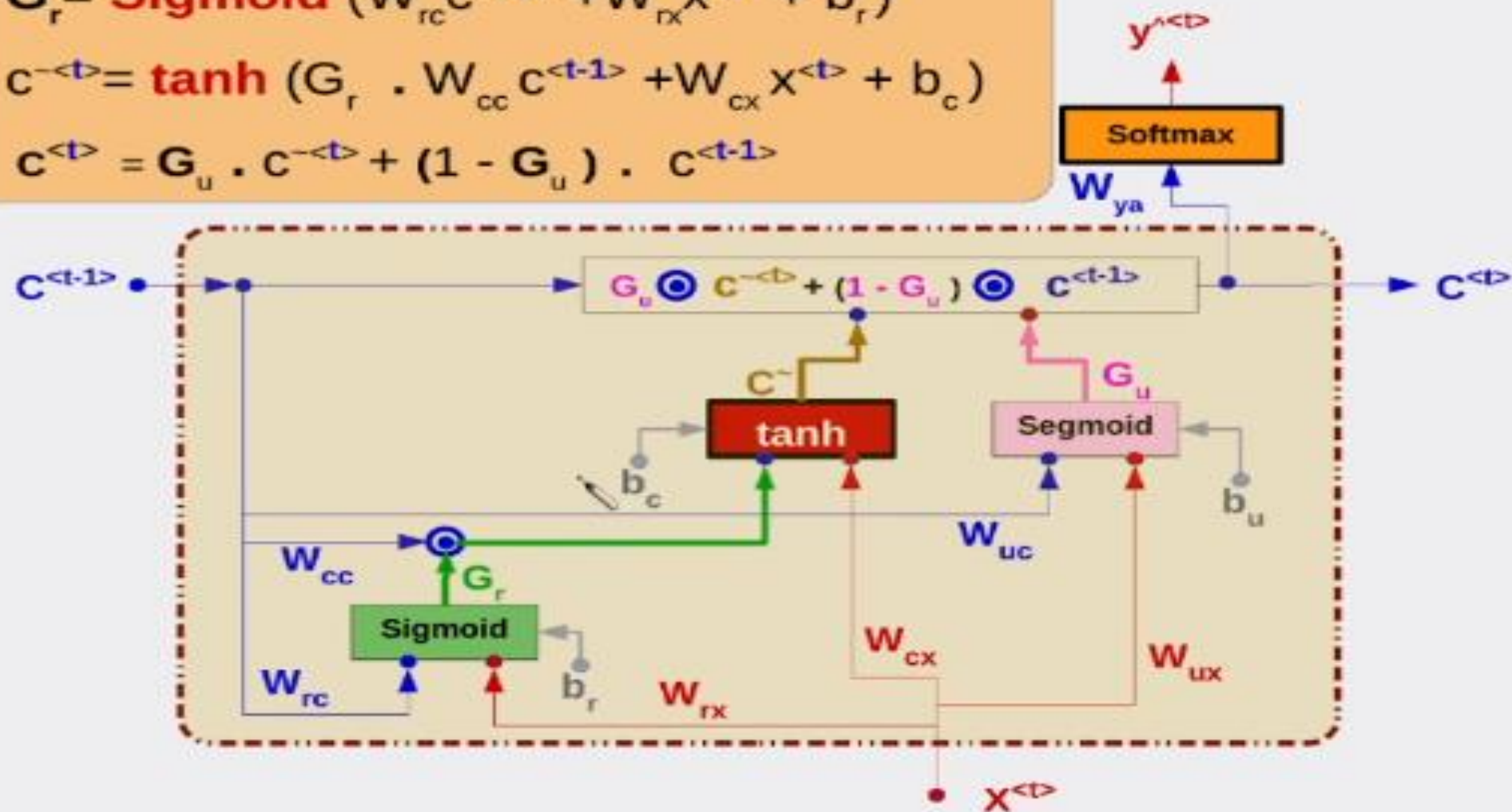
From RNN to GRU

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

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

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

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



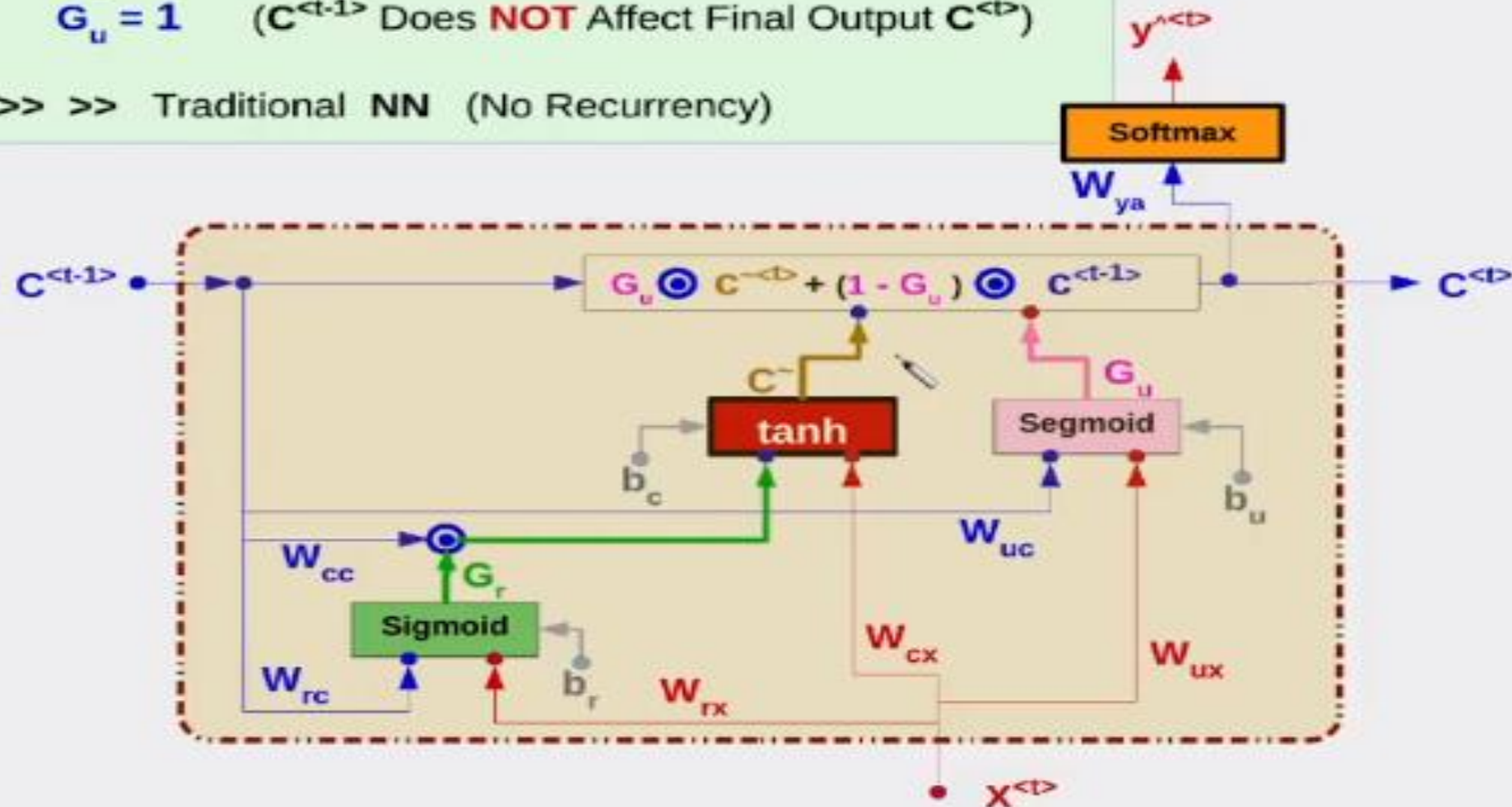
From RNN to GRU

IF

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

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

>> >> Traditional NN (No Recurrency)



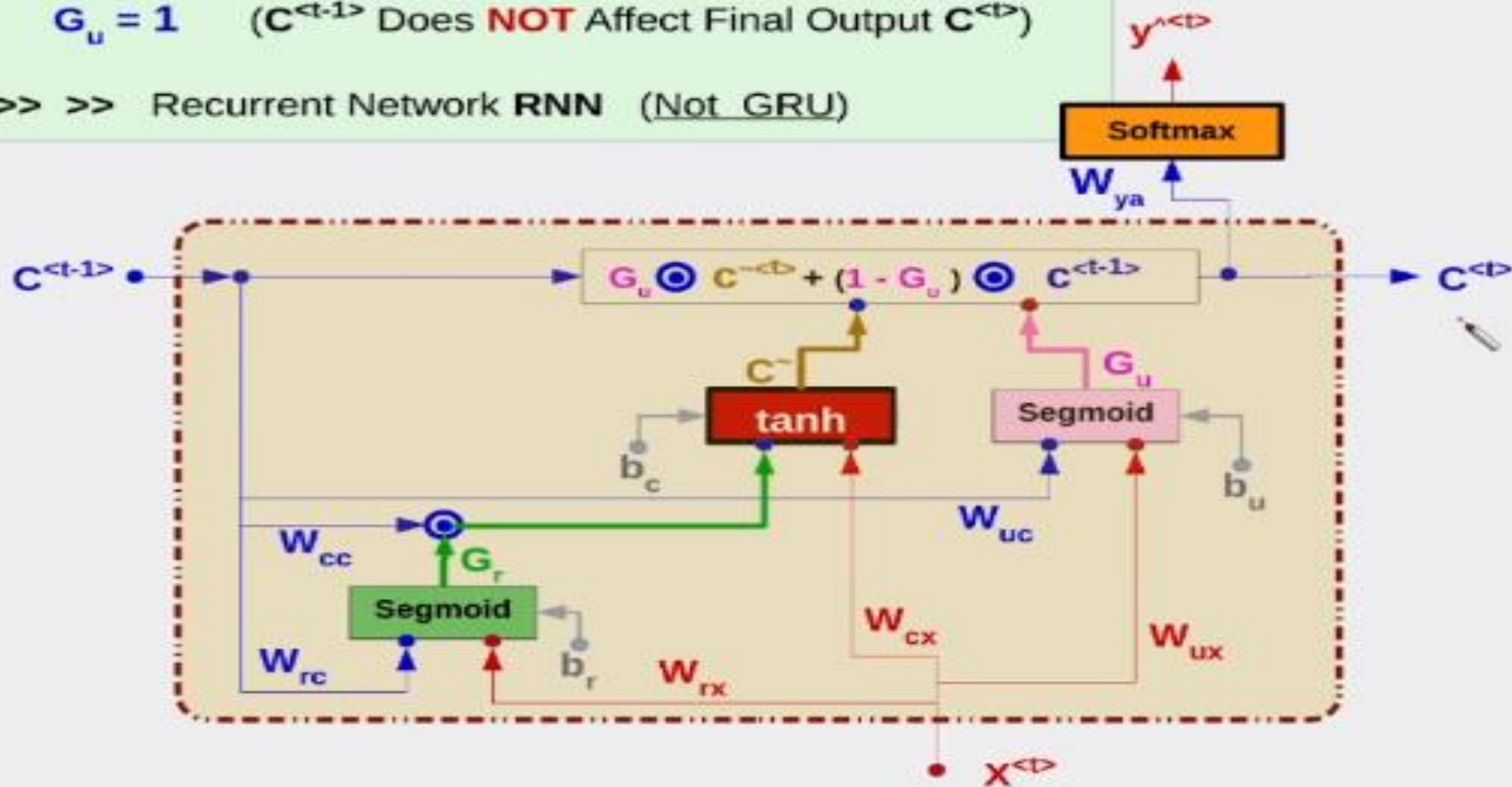
From RNN to GRU

IF

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

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

>> >> Recurrent Network **RNN** (Not GRU)



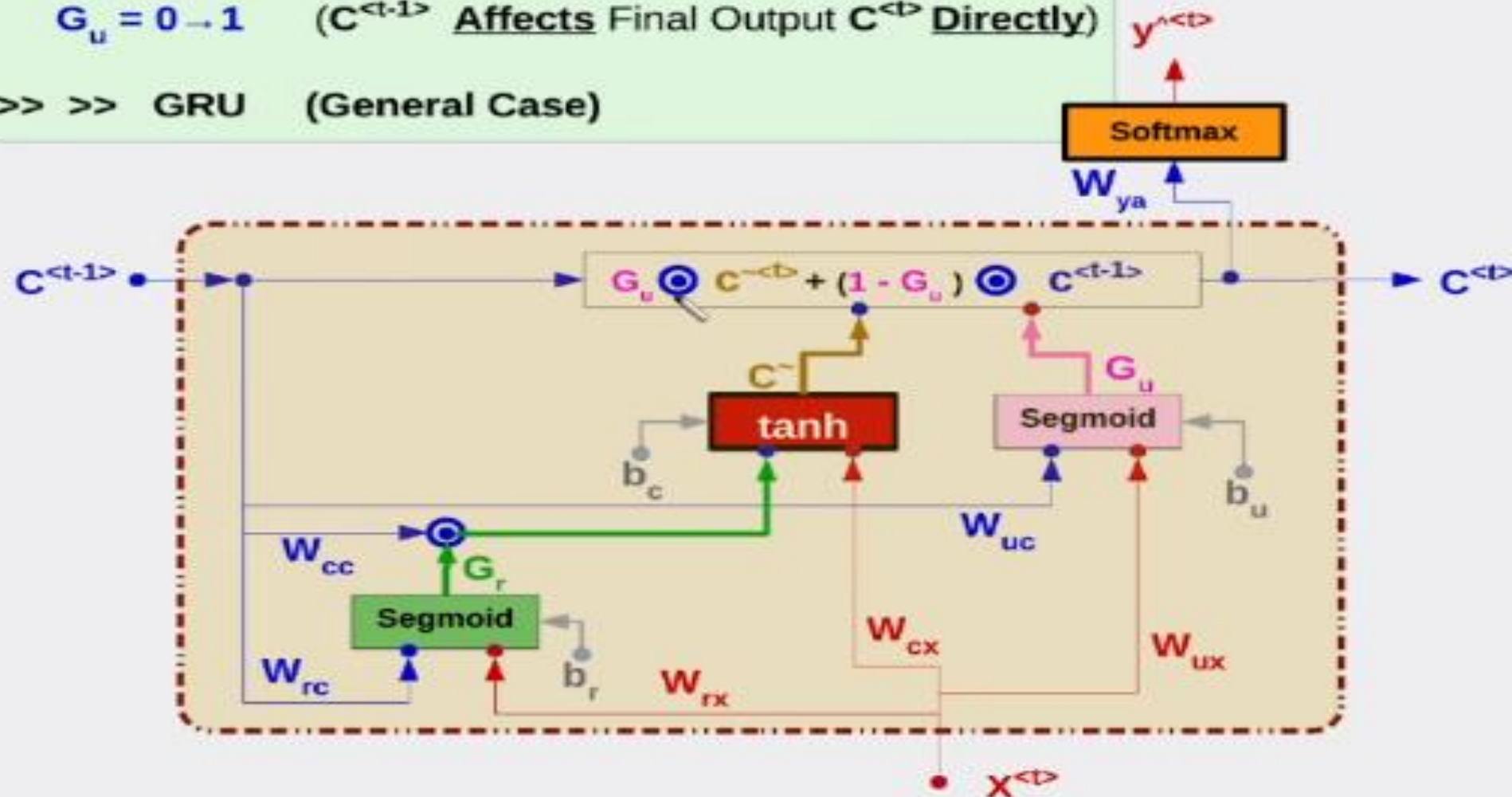
From RNN to GRU

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)
