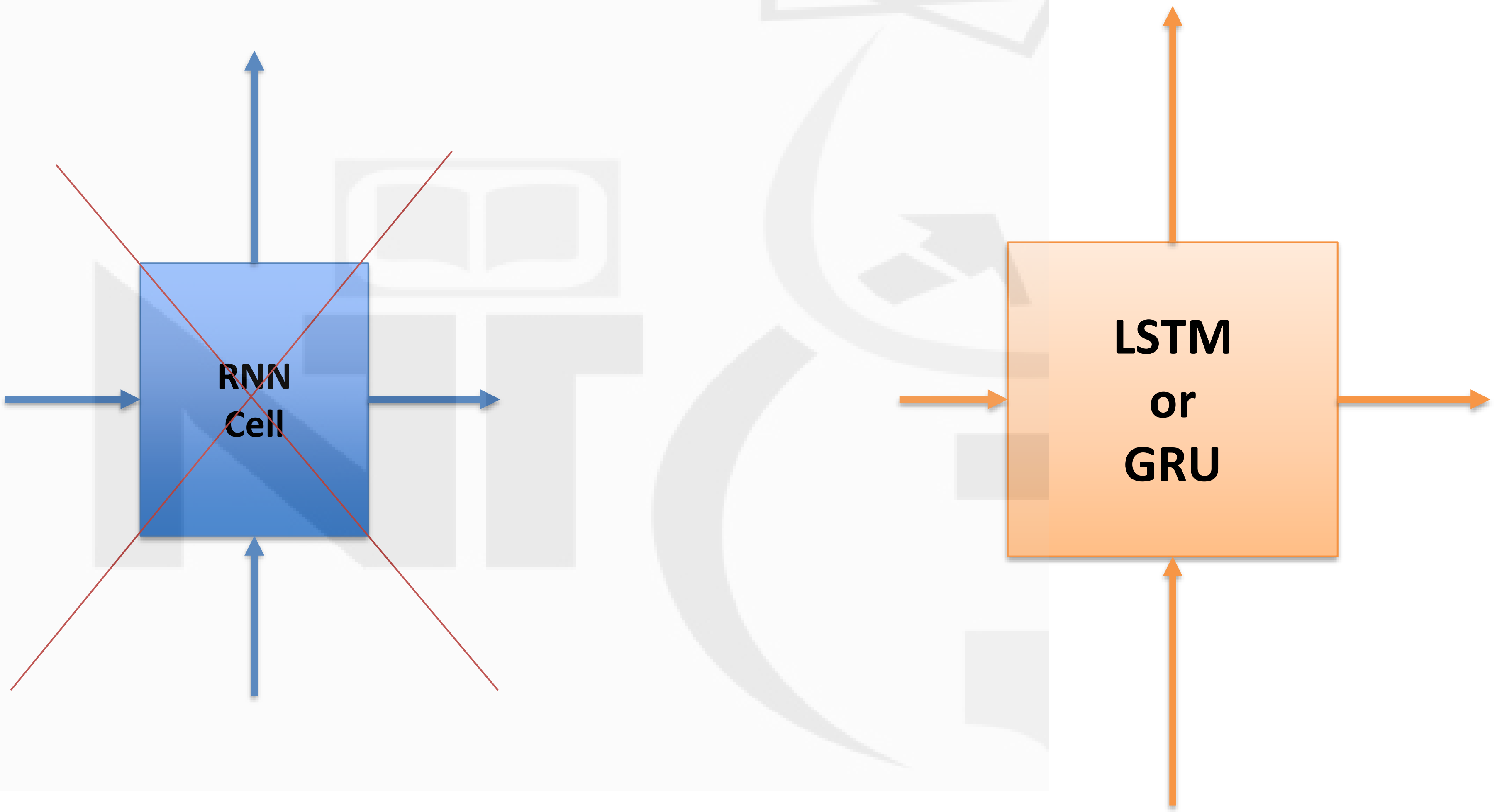
Solving Problems with RNN

MUKESH KUMAR

In Practice, LSTM or GRU is used instead of simple RNN



UNDERSTANDING LONG SHORT TERM MEMORY (LSTM)

Both Simple RNN and LSTM

- look at one data element at a time
- build memory over the sequence

but ...

- differ in how they build memory

Let's understand how humans build their memory

We humans keep building our memory throughout our life, we build memory on different things ...e.g

- Traveling from home to workplace
- Performing office work
- Knowledge of Neural networks
- So many things ....

For humans, the operations needed to build memory are:

1. **Create**
2. **Read**
3. **Update**
4. **Delete**

Based on today's session what should be changed in our memory?

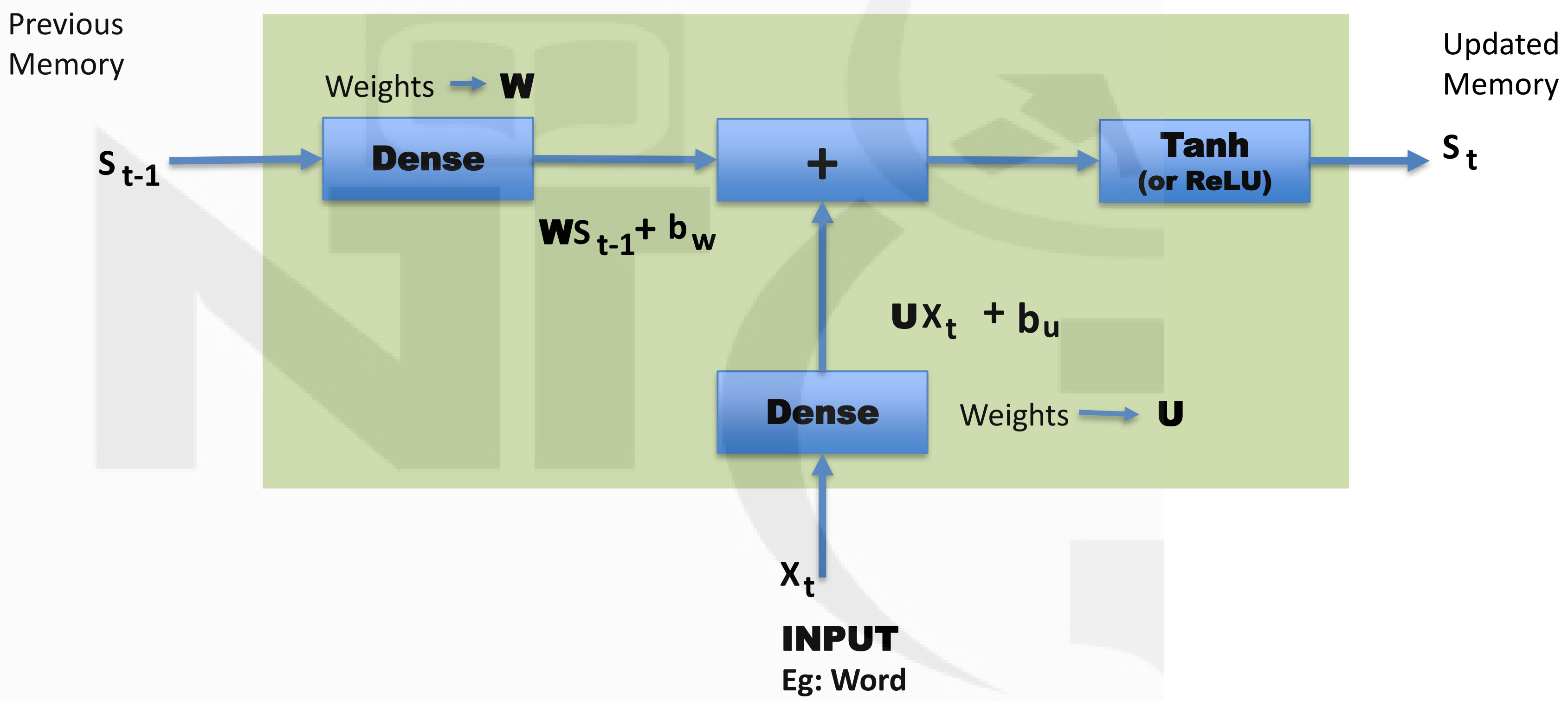
Things to **forget** and to **remember**	Things to **add/update** in the memory

What are the inputs to build this lists

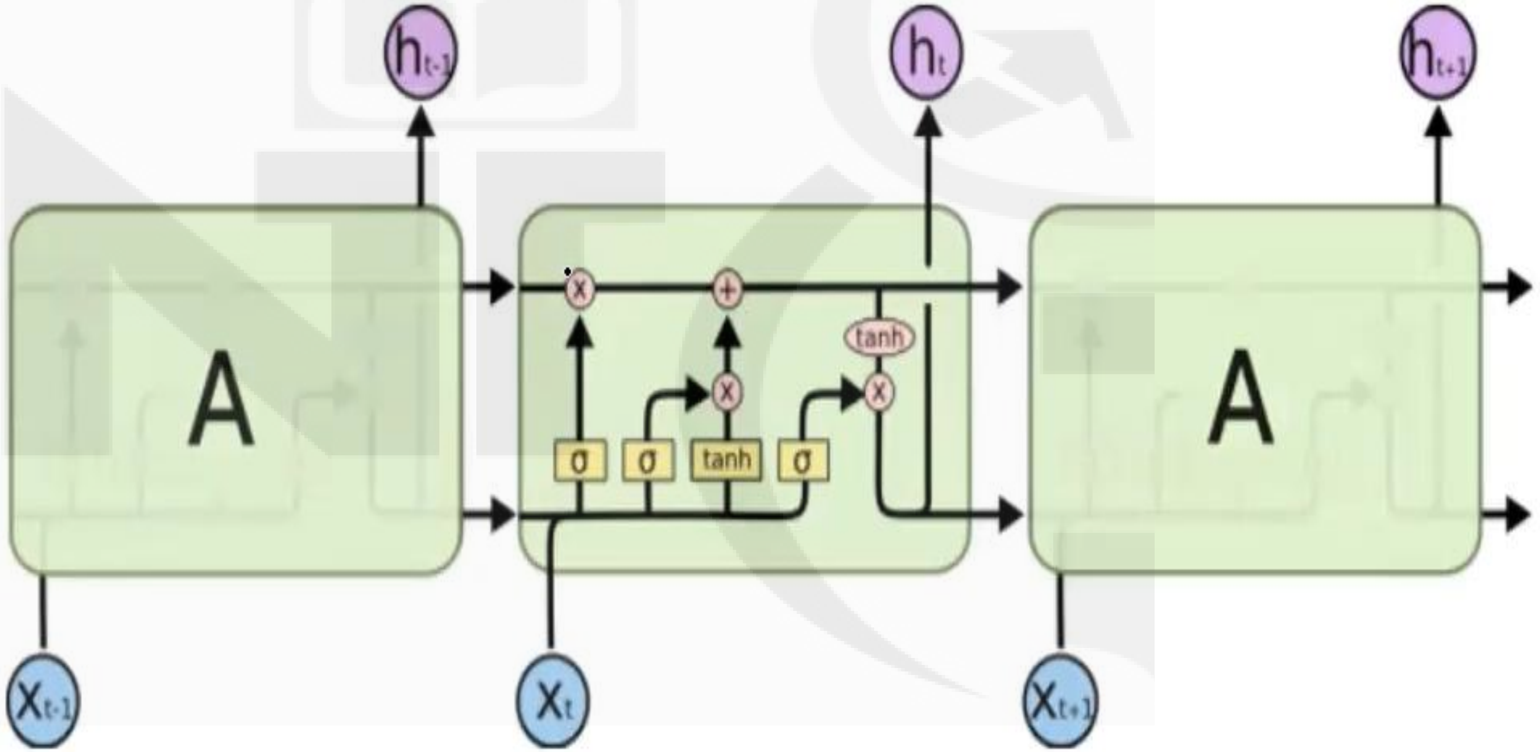
1. Our previous memory of NN
2. Current input (todays learning)

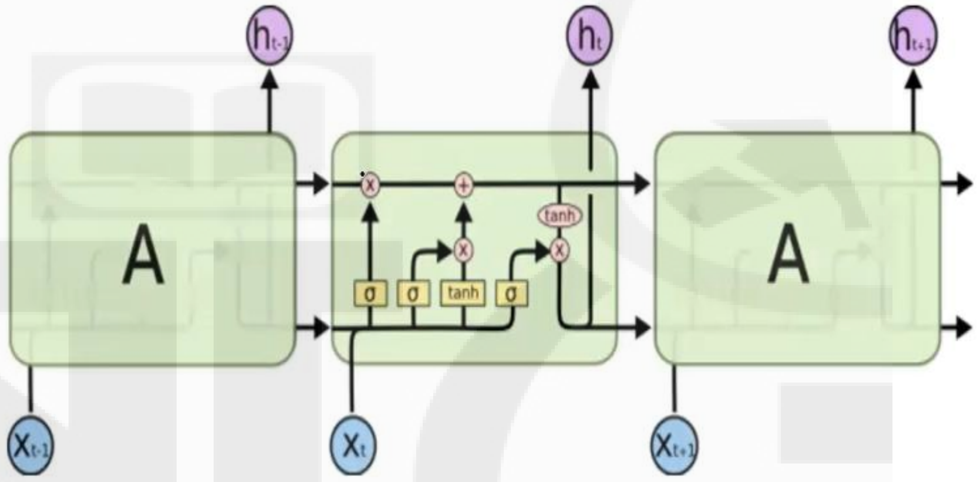
RNN VS LSTM

RNN CELL


Previous
Memory
S_{t-1}
Weights → W
Dense
$WS_{t-1} + b_w$
+
Tanh
(or ReLU)
Updated
Memory
S_t
$UX_t + b_u$
Dense
Weights → U
X_t
INPUT
Eg: Word

LSTM Cell





LSTM uses different neural networks to perform different memory operation

Long Term Memory
i.e. Cell State

Hidden State

LSTM has **two** type of memories (compared to one in simple RNN)

- Hidden state is a derived from cell state

- Cell state is all the knowledge about everything

- Both the memory will be of same size(hyperparameter)

Long Term Memory
i.e. Cell State

Cell State keeps getting updated as LSTM gets to look a new point

To start with, both Cell state and hidden state will be randomly initialized.

LSTM has **two** type of memories (compared to one in simple RNN)

Hidden State

Filtered version of Cell state based on current input

Long Term Memory
i.e. Cell State

Cell State keep s getting updated as LSTM gets to look a new point

Let's say LSTM memory size is 5. We initialize both cell and hidden state with some random values

LSTM has **two** type of memories (compared to one in simple RNN)

Hidden State

Filtered version of Cell state based on current input

Long Term Memory
i.e. Cell State

Cell State keep s getting updated as LSTM gets to look a new point

[3,1,0,4,-2]

Let's say LSTM memory size is 5. We initialize both cell and hidden state with some random values

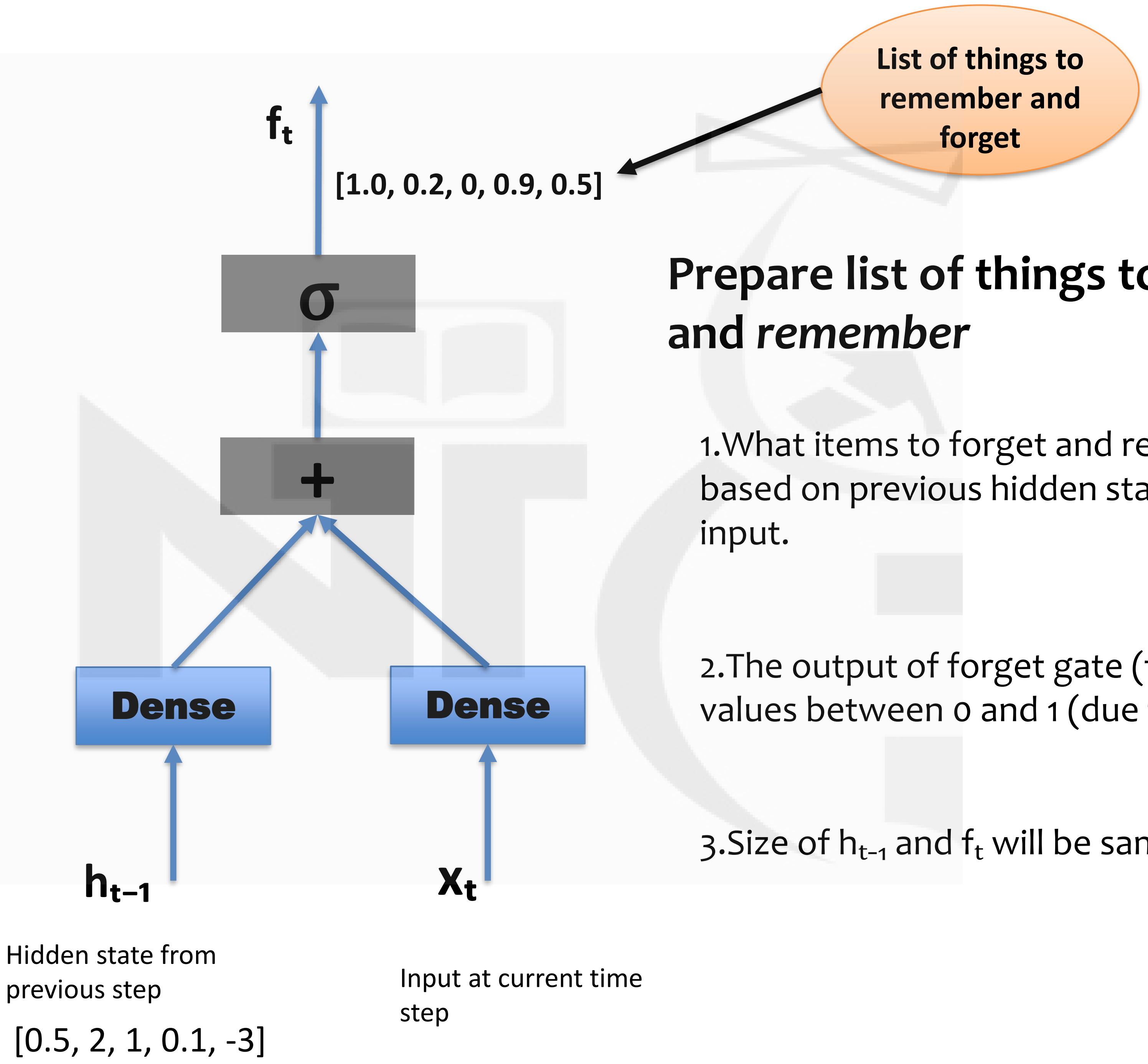
[0.5, 2, 1, 0.1, -3]

LSTM has **two** type of memories (compared to one in simple RNN)

Hidden State

Filtered version of Cell state based on current input

How to delete (or forget) things from LSTM memory?

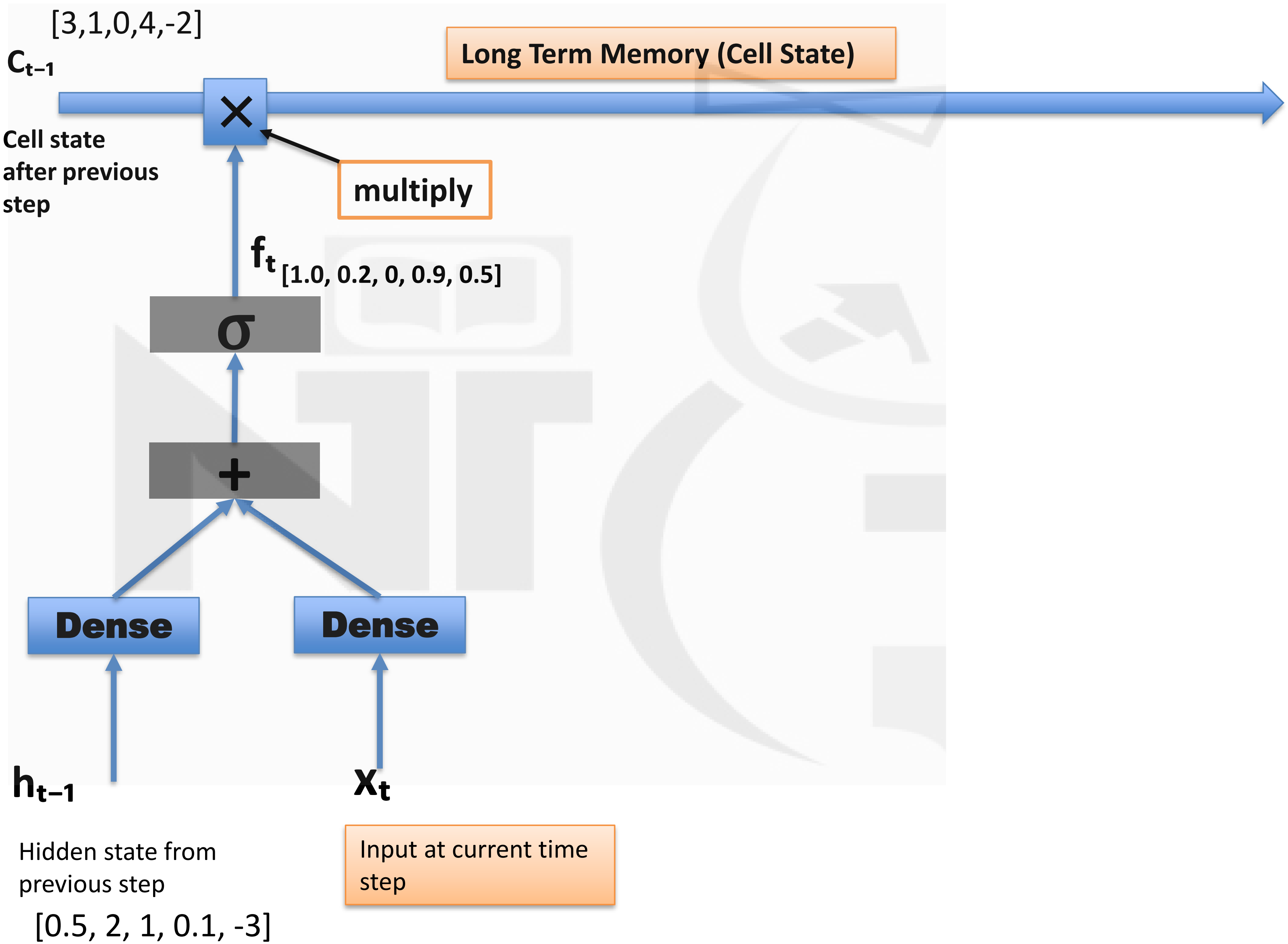


Prepare list of things to *forget* and *remember*

1. What items to forget and remember are based on previous hidden state and current input.

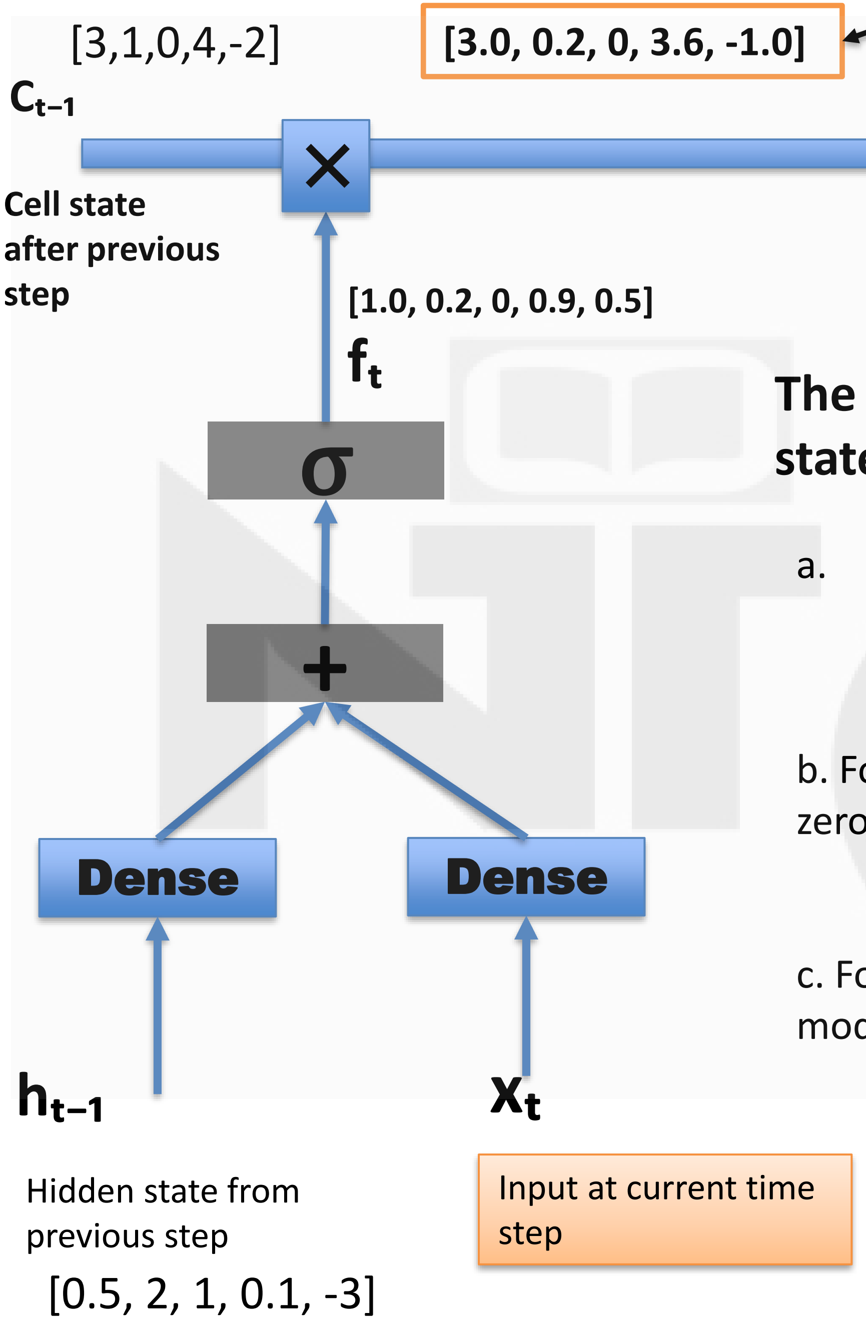
2. The output of forget gate (f_t) will have values between 0 and 1 (due to sigmoid).

3. Size of h_{t-1} and f_t will be same



[3,1,0,4,-2]
C_{t-1}
Long Term Memory (Cell State)
Cell state
after previous
step
multiply
f_t [1.0, 0.2, 0, 0.9, 0.5]
σ
+
Dense
Dense
h_{t-1}
x_t
Hidden state from
previous step
[0.5, 2, 1, 0.1, -3]
Input at current time
step

This is the updated memory after removing what we wanted to forget

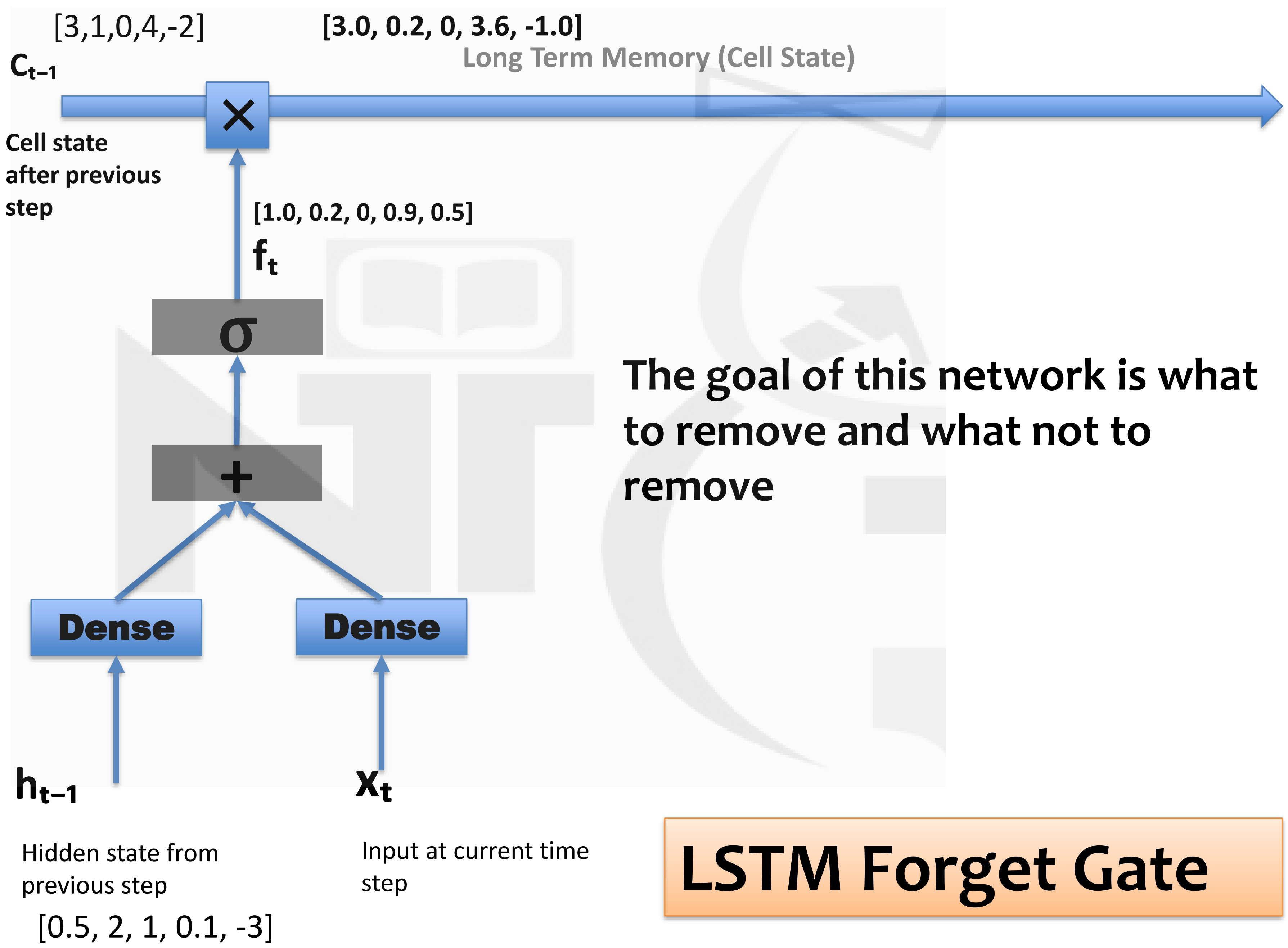


The output (ft) is then multiplied with cell state:

a. For values equal to 1 in ft, cell state values will be kept as is (**remember** things)

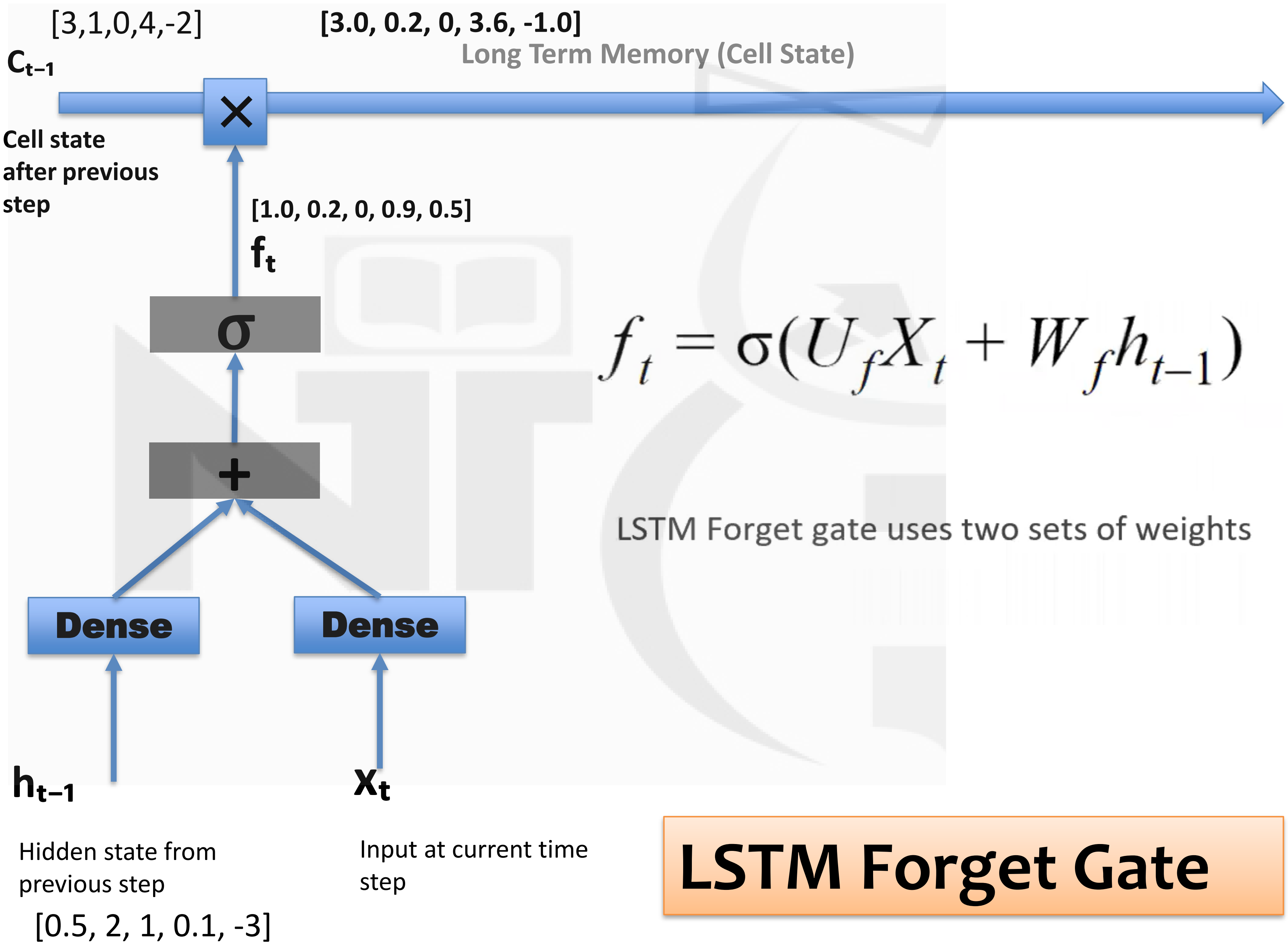
b. For values equal to 0 in ft, cell state values will be set to zeros (**forget** things)

c. For values between 0 and 1 in ft, cell state values will be modified (something to remember, something to forget)



The goal of this network is what to remove and what not to remove

LSTM Forget Gate



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

LSTM Forget gate uses two sets of weights

LSTM Forget Gate

LSTM has another specialized team/network for adding/updating

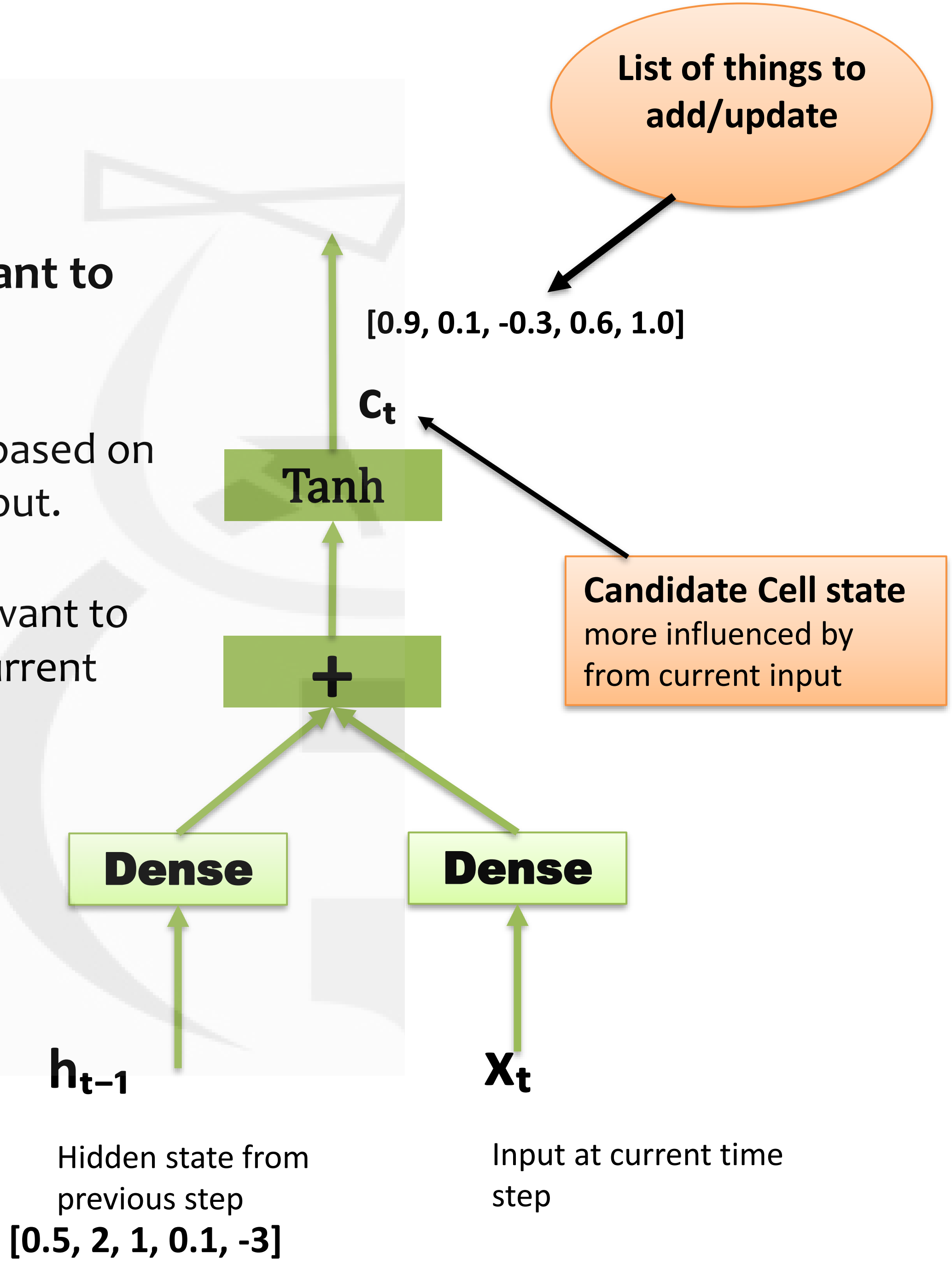
HOW TO UPDATE THINGS FROM LSTM MEMORY

Prepare list of things which you want to add and update

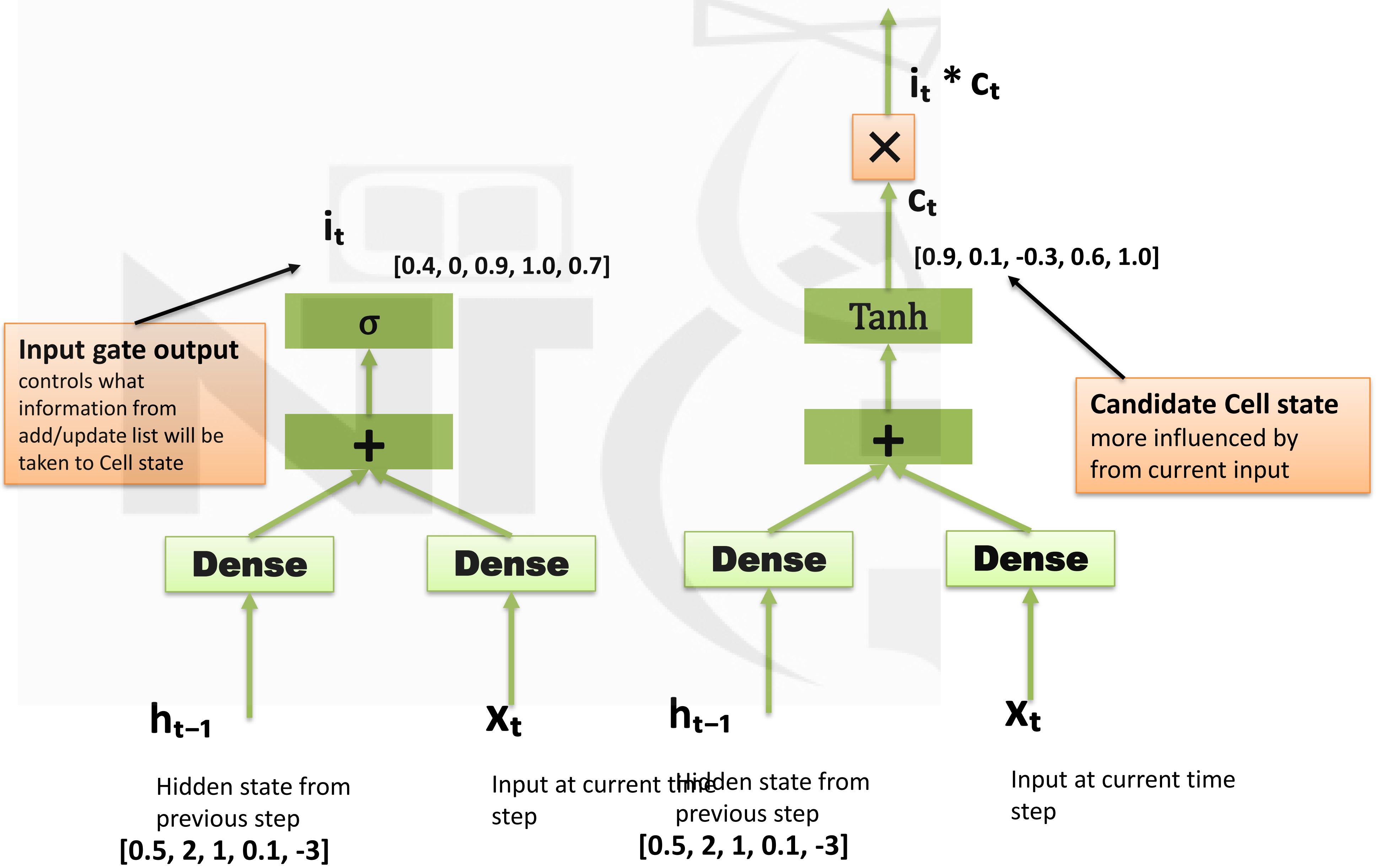
1.What items to add and update are based on previous hidden state and current input.

2.The output (c_t) indicates what we want to add/update to long term based on current input.

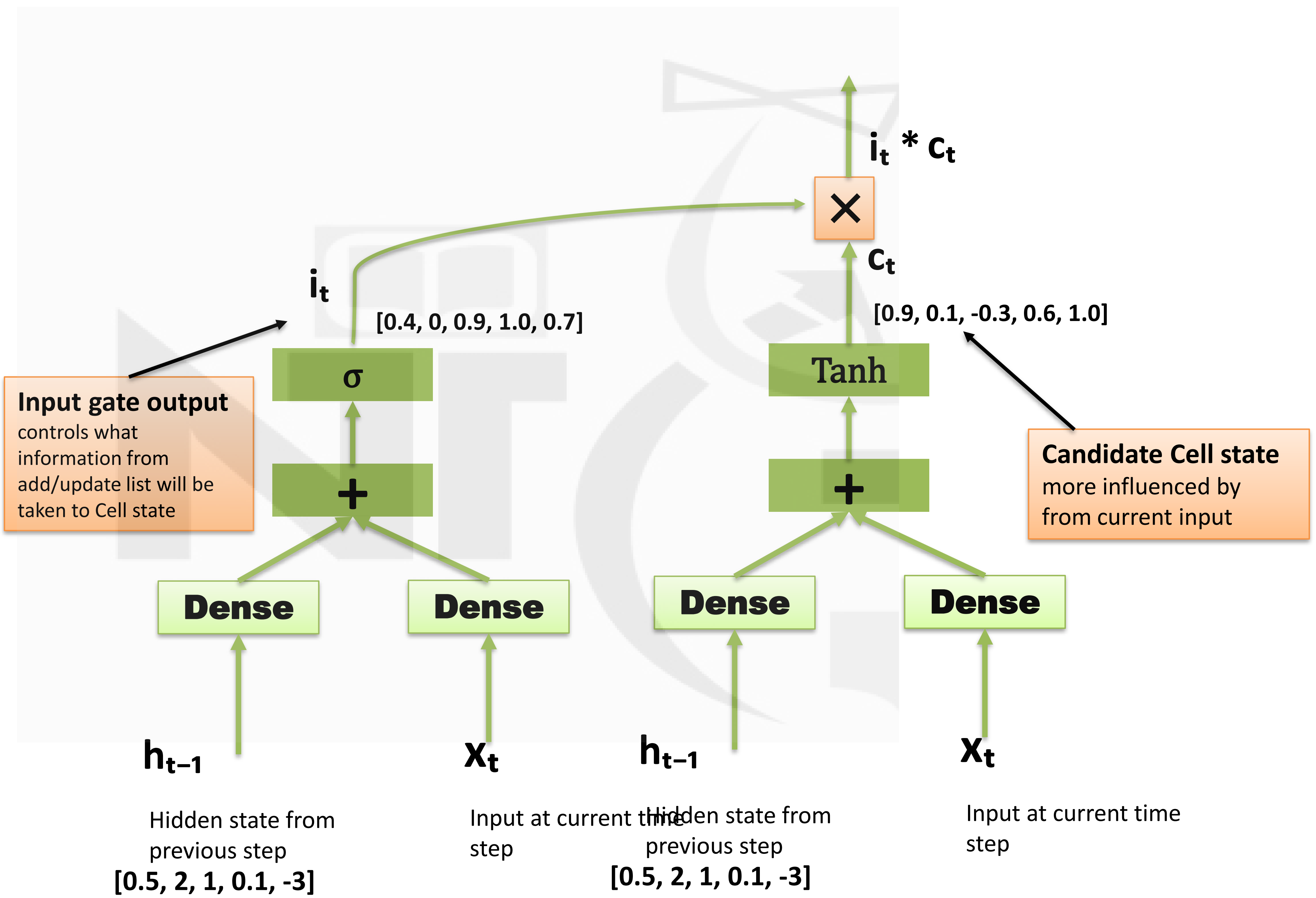
3.We can use 'Relu' in place of 'tanh'



- Not the entire info is retained as is in the final memory
- There is another network that works parallelly, which will look at each of these values and decide how much of these values will go to final memory



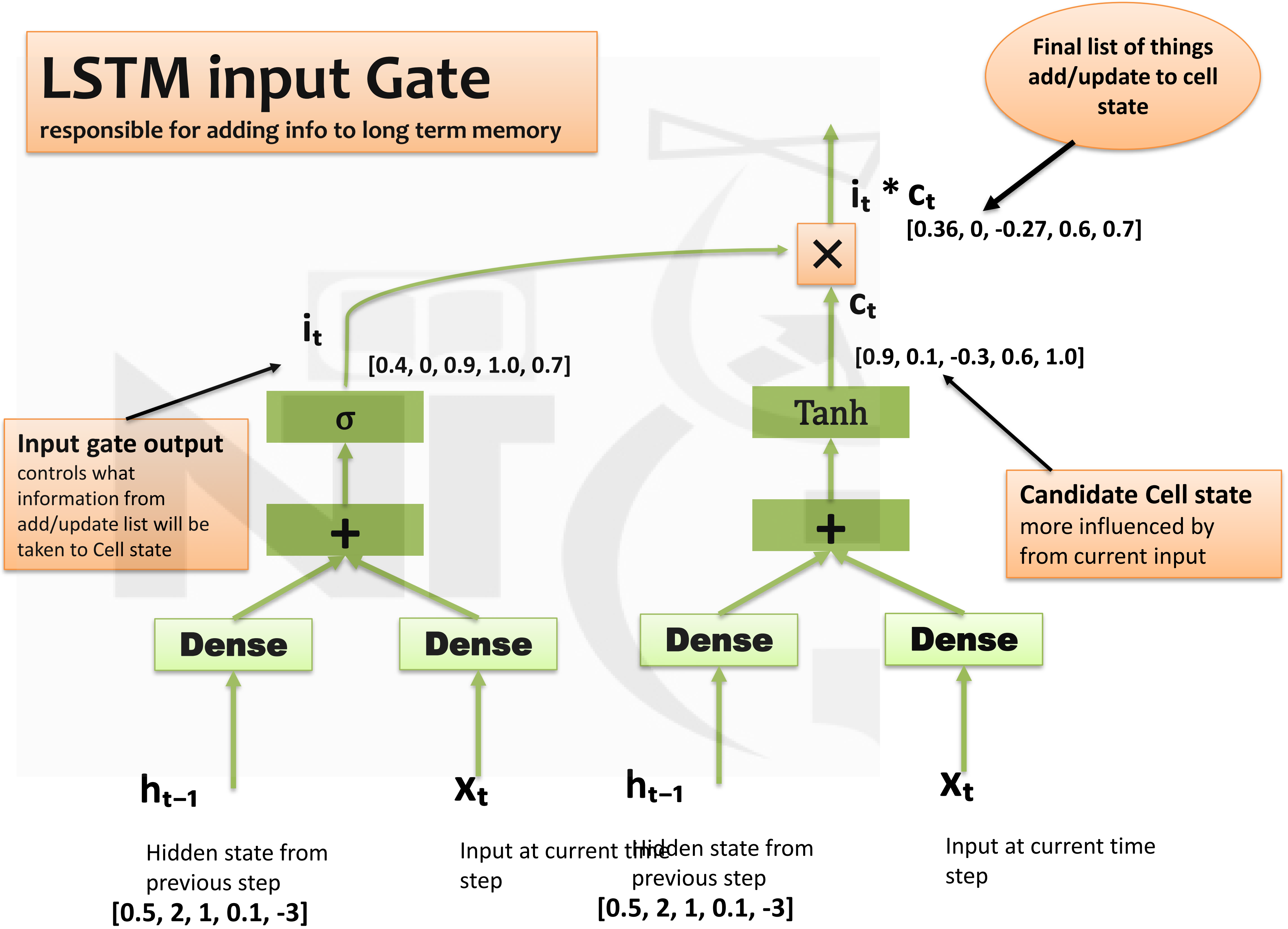
i_t * C_t
C_t
[0.9, 0.1, -0.3, 0.6, 1.0]
Tanh
i_t
[0.4, 0, 0.9, 1.0, 0.7]
σ
Input gate output
controls what information from add/update list will be taken to Cell state
Candidate Cell state
more influenced by from current input
Dense
Dense
Dense
Dense
h_t−1
x_t
h_t−1
x_t
Hidden state from previous step
[0.5, 2, 1, 0.1, -3]
Input at current time step
Hidden state from previous step
[0.5, 2, 1, 0.1, -3]
Input at current time step

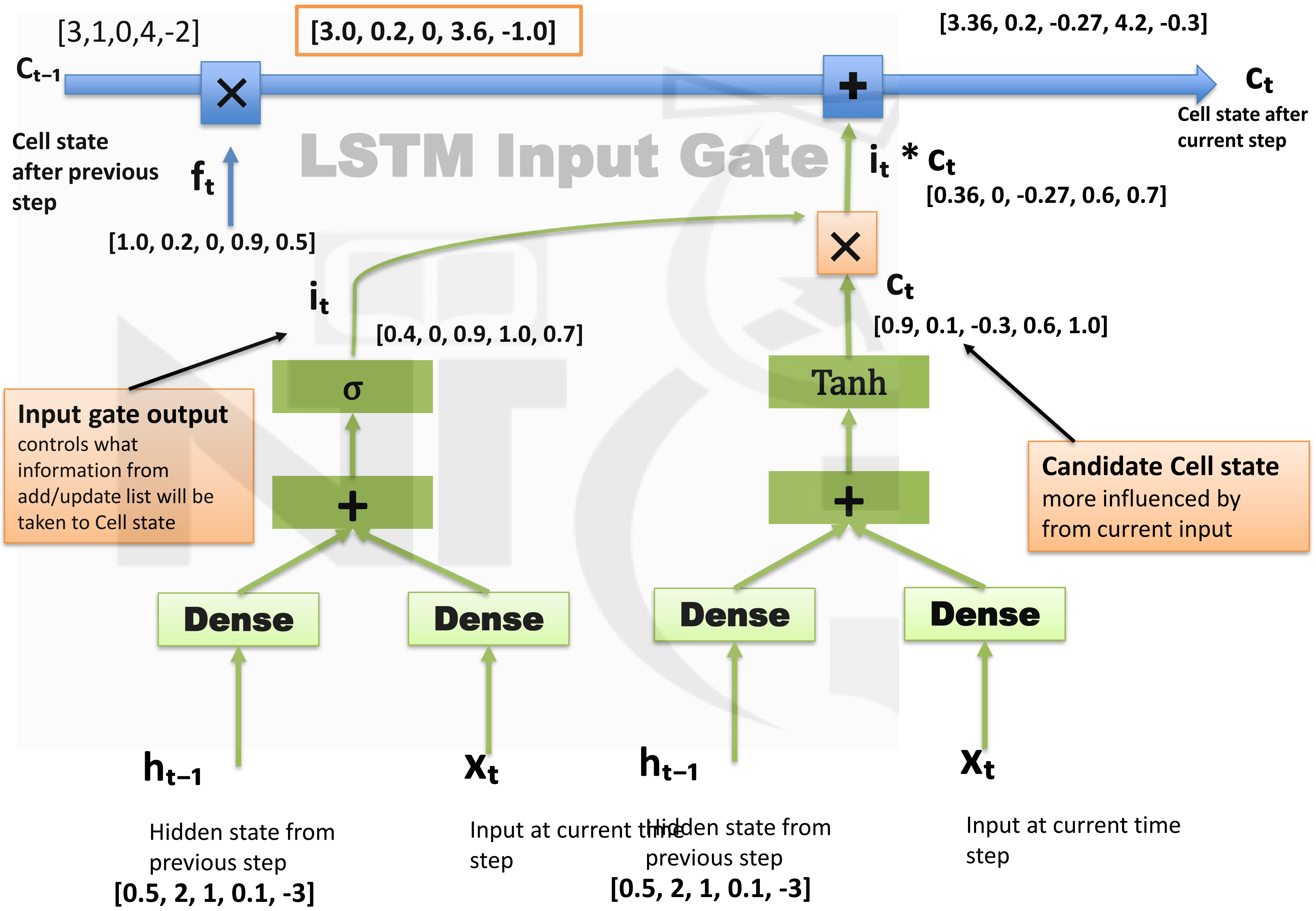


i_t * C_t
×
C_t
[0.9, 0.1, -0.3, 0.6, 1.0]
i_t
[0.4, 0, 0.9, 1.0, 0.7]
σ
Tanh
Input gate output
controls what information from add/update list will be taken to Cell state
Candidate Cell state
more influenced by from current input
+
+
Dense
Dense
Dense
Dense
h_t−1
x_t
h_t−1
x_t
Hidden state from previous step
[0.5, 2, 1, 0.1, -3]
Input at current time step
Hidden state from previous step
[0.5, 2, 1, 0.1, -3]
Input at current time step

LSTM input Gate

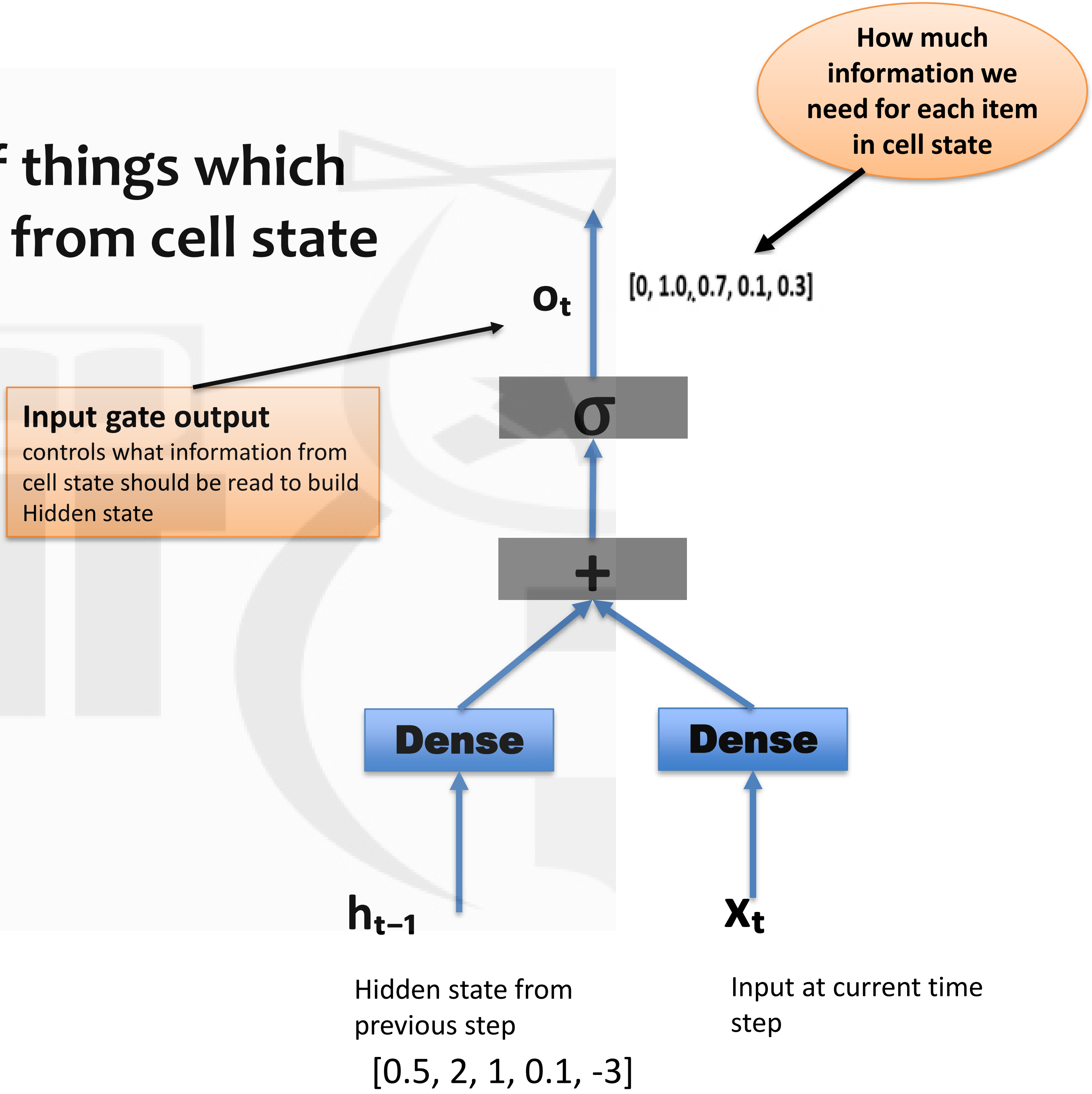
responsible for adding info to long term memory

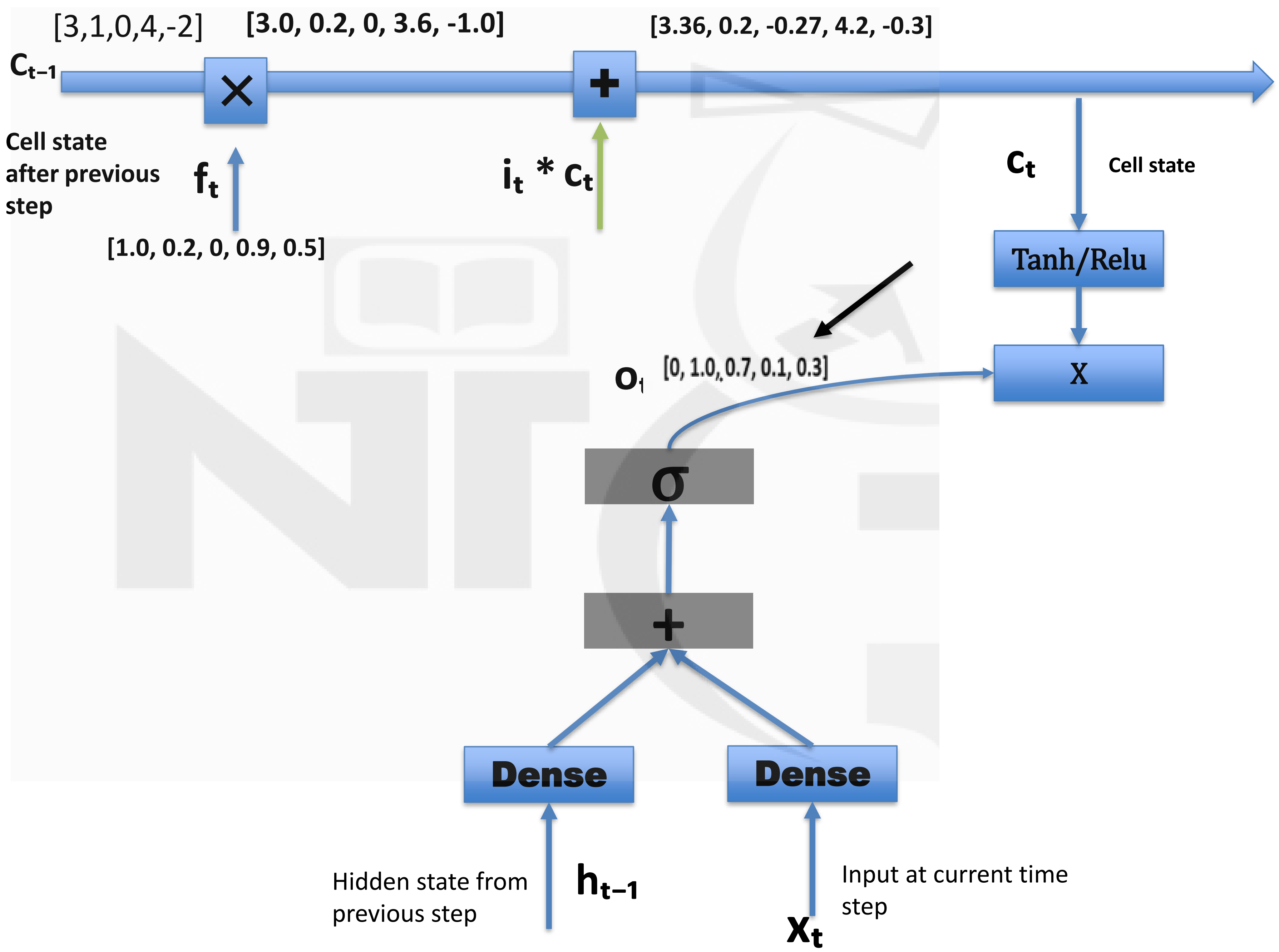



LSTM Input Gate
[3,1,0,4,-2]
[3.0, 0.2, 0, 3.6, -1.0]
[3.36, 0.2, -0.27, 4.2, -0.3]
Ct-1
Ct
Cell state after current step
Cell state after previous step
ft
it * Ct
[0.36, 0, -0.27, 0.6, 0.7]
[1.0, 0.2, 0, 0.9, 0.5]
it
[0.4, 0, 0.9, 1.0, 0.7]
Ct
[0.9, 0.1, -0.3, 0.6, 1.0]
σ
Tanh
Input gate output
controls what information from add/update list will be taken to Cell state
Candidate Cell state
more influenced by from current input
Dense
Dense
Dense
Dense
ht-1
Xt
ht-1
Xt
Hidden state from previous step
[0.5, 2, 1, 0.1, -3]
Input at current time step
Hidden state from previous step
[0.5, 2, 1, 0.1, -3]
Input at current time step

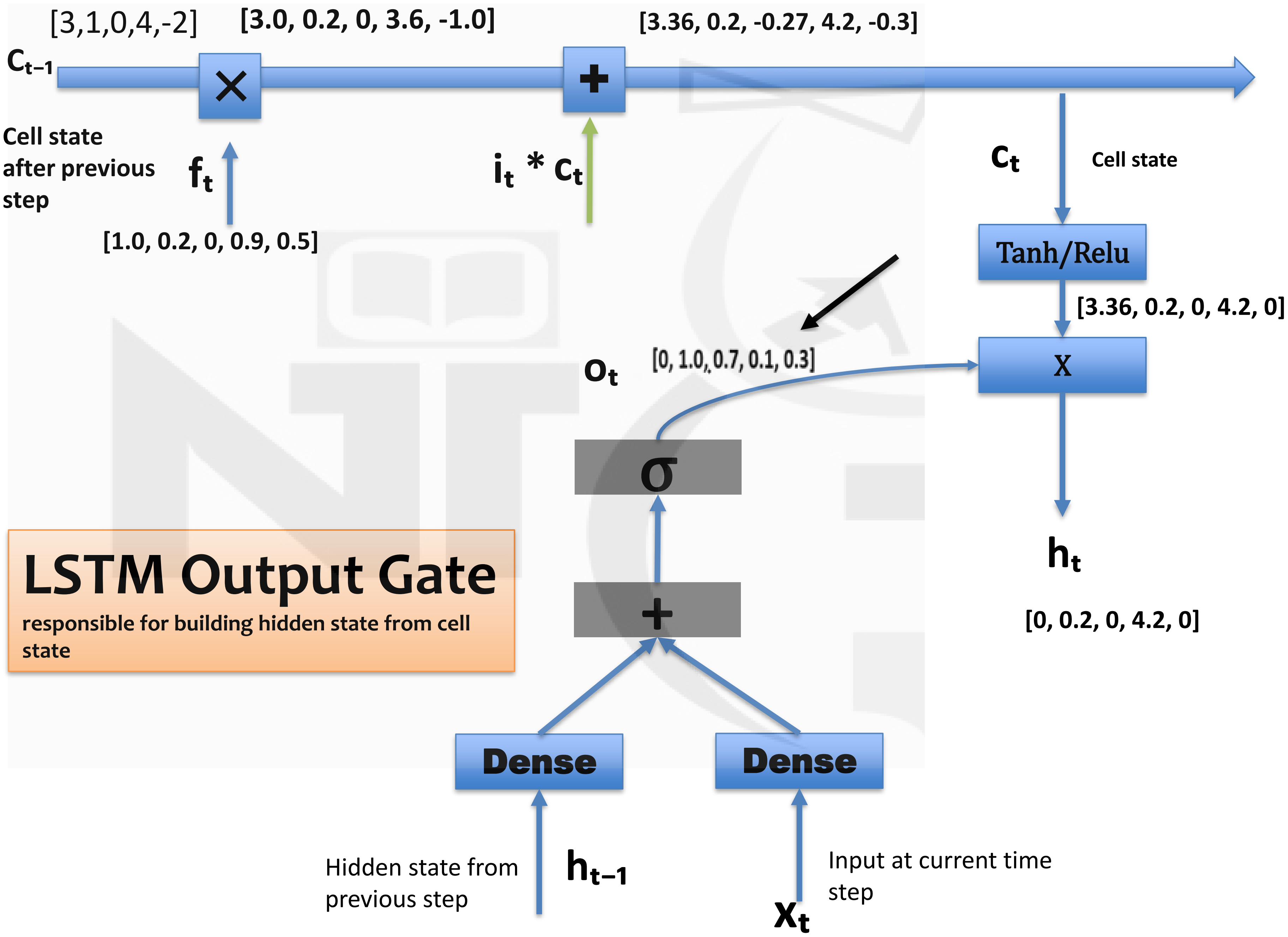
Reading Cell State to build Hidden State

Prepare list of things which we want read from cell state





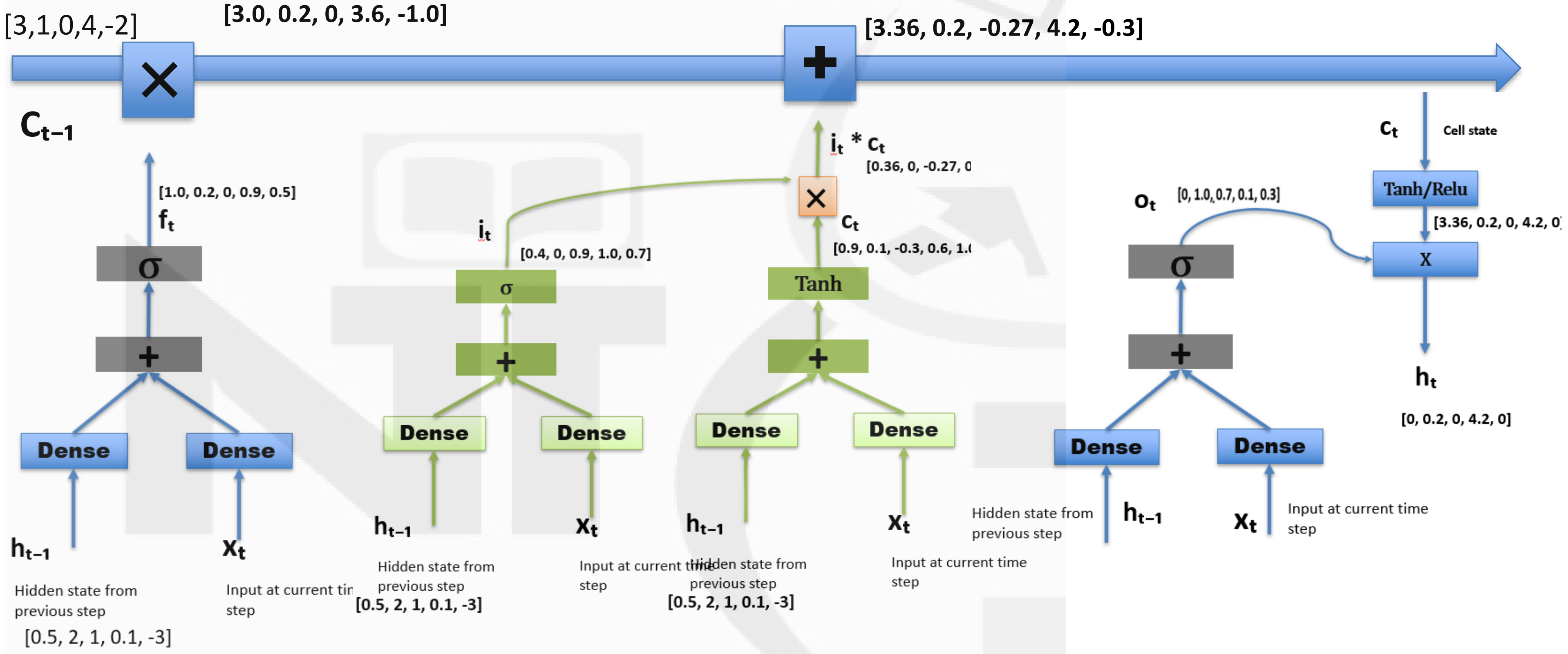
[3,1,0,4,-2]
[3.0, 0.2, 0, 3.6, -1.0]
[3.36, 0.2, -0.27, 4.2, -0.3]
C_{t-1}
Cell state after previous step
f_t
[1.0, 0.2, 0, 0.9, 0.5]
$i_t * C_t$
C_t
Cell state
Tanh/Relu
X
o_t
[0, 1.0, 0.7, 0.1, 0.3]
σ
+
Dense
Dense
Hidden state from previous step
h_{t-1}
Input at current time step
X_t



LSTM Output Gate

responsible for building hidden state from cell state

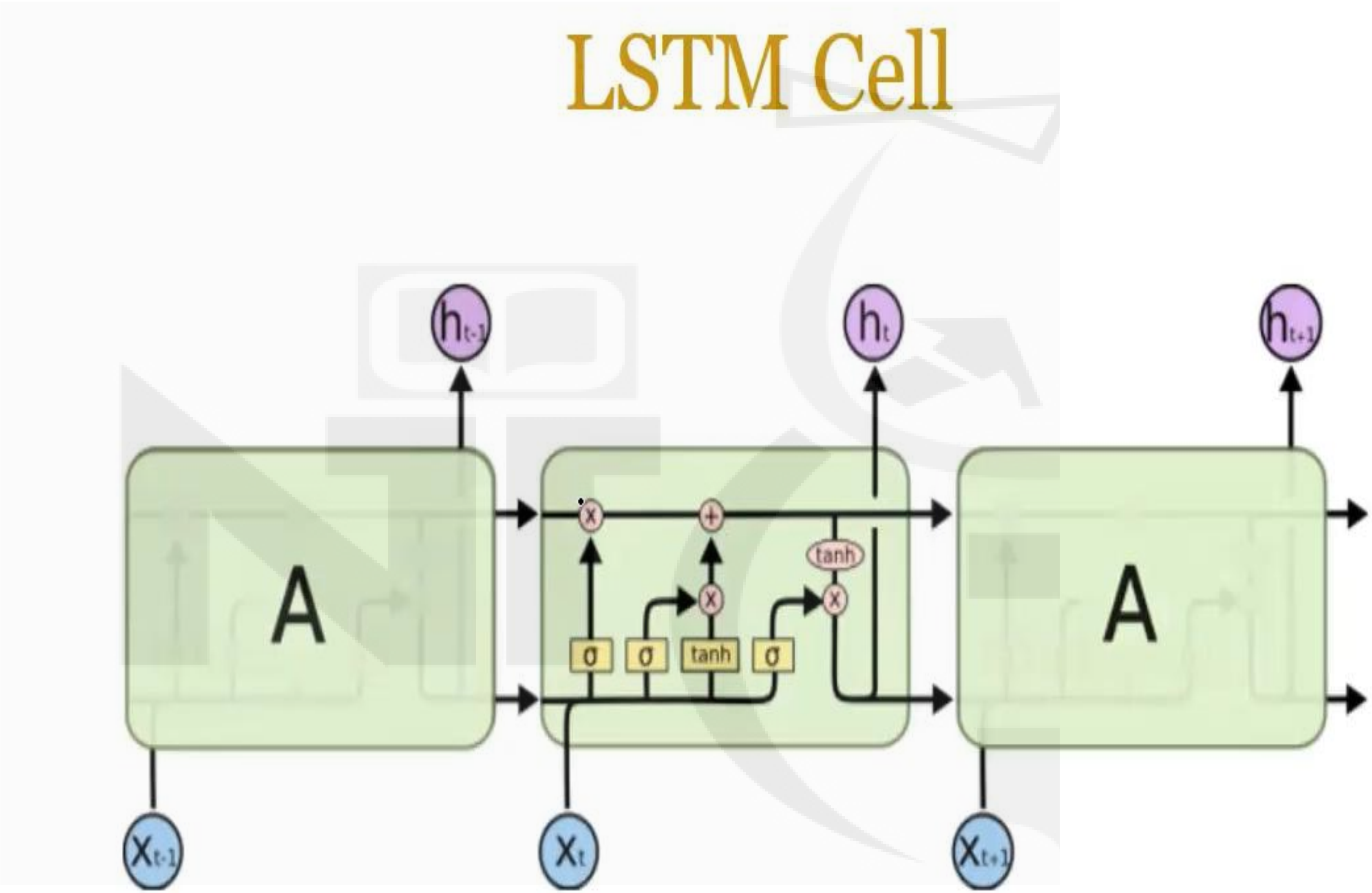
LSTM CELL



Forget Gate
Controls things to remember & forget from memory

Input Gate
Controls what to add/update to memory

Output Gate
Controls what to read from memory



Another View of the same Cell

LSTM Summary

1. **Use two states or memory**
   - ✓ **Cell State:** used to store Long term memory.
   - ✓ **Hidden State:** A filtered version of Cell state, used to build final output
2. **Three (3) Gates**
   a) What is a 'Gate'?
      - A Gate controls flow of information to/from long term memory (Cell state)

   b) What are three gates in LSTM?
   - ✓ **Forget Gate:** Controls what to forget and what to keep in long term memory (Cell state)
   - ✓ **Input Gate:** Controls what to add to long term memory
   - ✓ **Output Gate:** What information from cell state will be sent to hidden state

What is the output of LSTM??

Hidden state

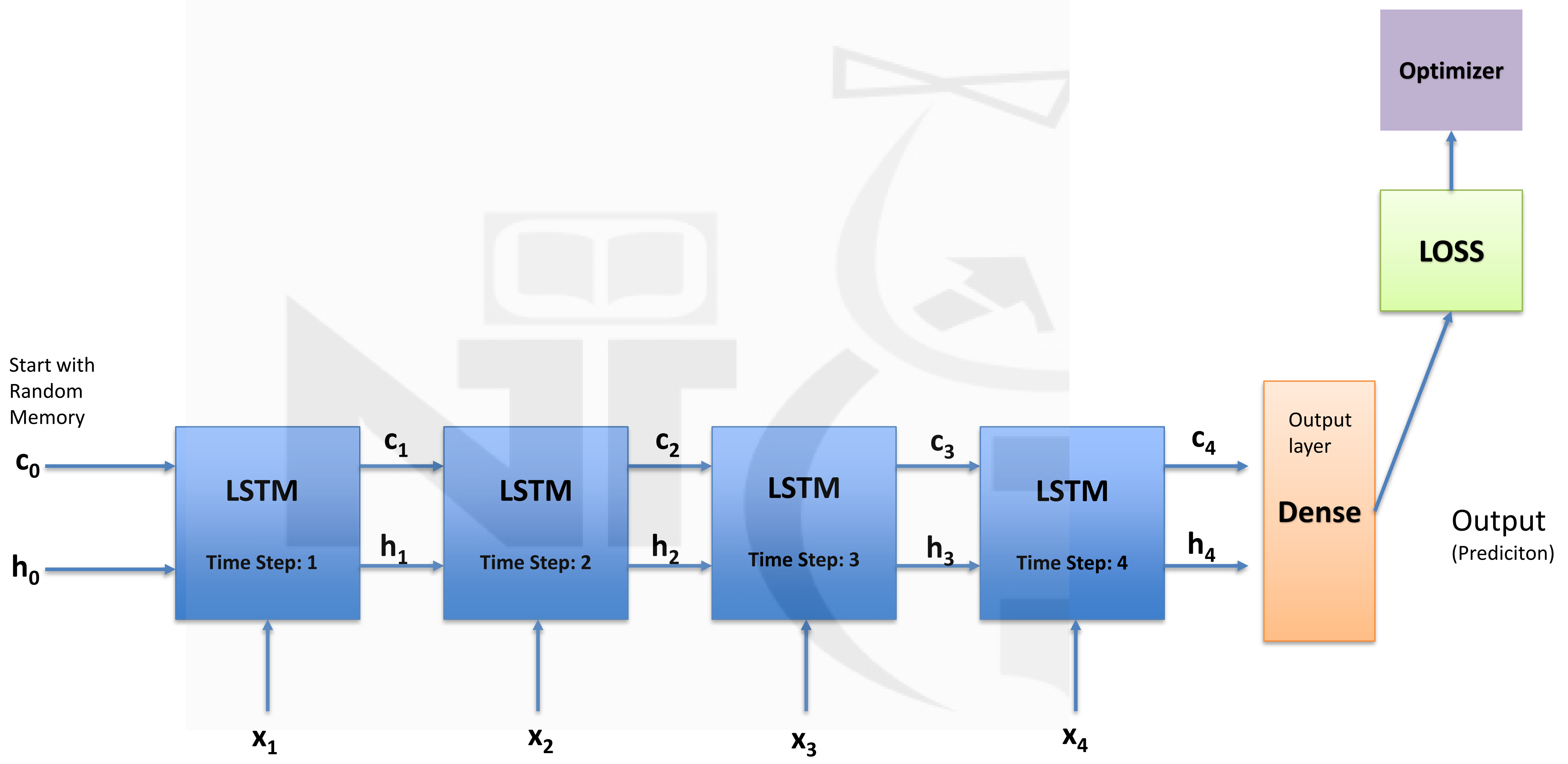
and/or

Cell State

How many numbers will we get in the LSTM output?

- Memory Size
- It's a hyperparameter
- If memory=100, hidden State=100 and Cell State =100

BACK PROPAGATION IN LSTM



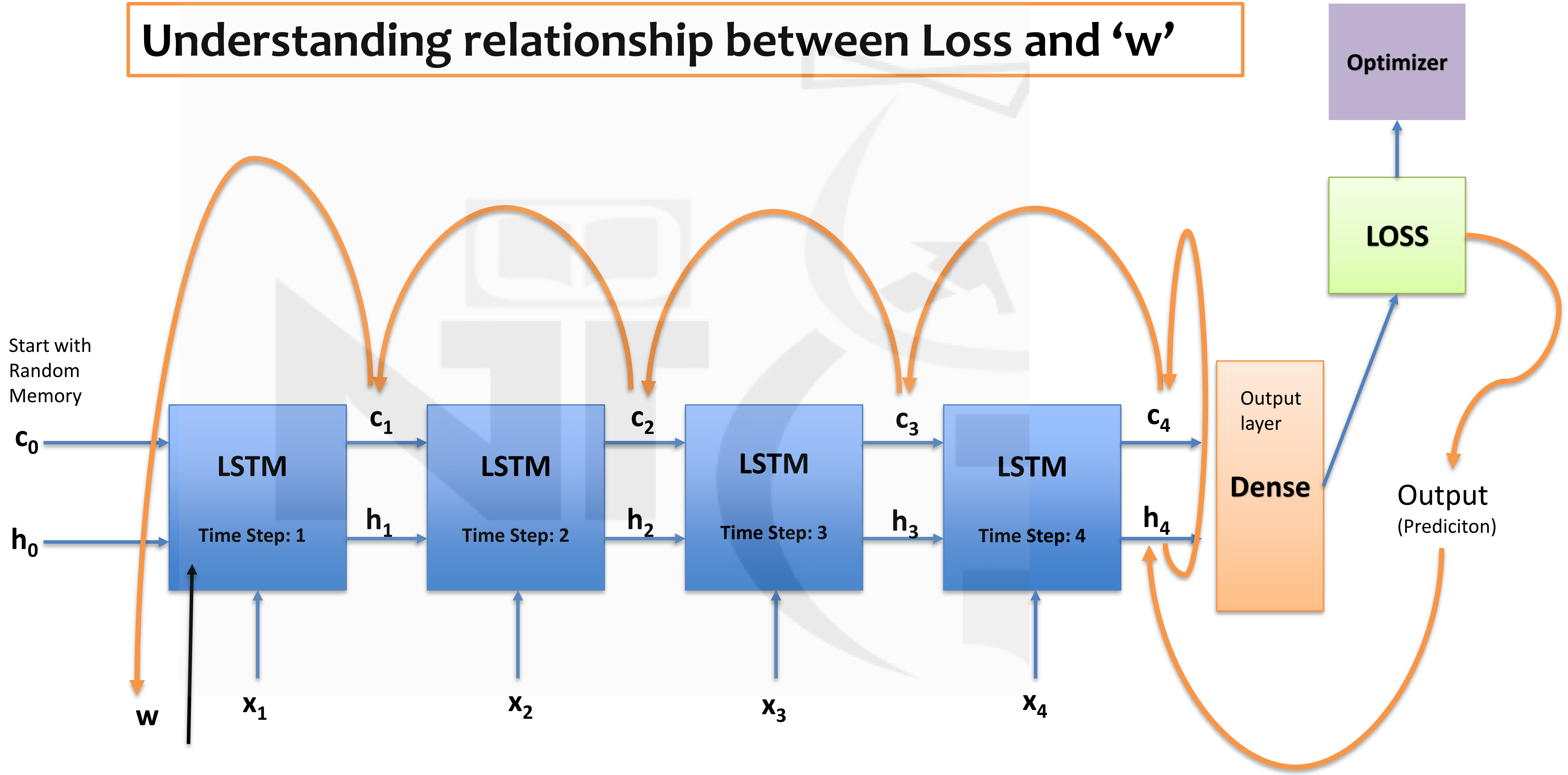
FORWARD PROPAGATION IN LSTM

Let's Calculate

$$\frac{dLoss}{dw}$$

Where 'w' is any of the weights in LSTM

Understanding relationship between Loss and ‘w’



FORWARD PROPAGATION IN LSTM

$$\frac{dLoss}{dw} = \frac{dLoss}{dO} \cdot \frac{dO}{dh_4} \cdot \frac{dh_4}{dc_4} \cdot \frac{dc_4}{dc_3} \cdot \frac{dc_3}{dc_2} \cdot \frac{dc_2}{dc_1} \cdot \frac{dc_1}{dw}$$

Calculating updated Cell State from the previous Cell State

$$C_t = C_{t-1} \cdot f_t + i_t \cdot \hat{c}_t$$

Calculating updated Cell State from the previous Cell State

$$C_t = C_{t-1} \bullet f_t + i_t \bullet \widehat{c}_t$$

Let's calculate gradient of c_t with c_{t-1}

$$\frac{dc_t}{dc_{t-1}} = f_t + \frac{d(i_t \cdot \widehat{c_t})}{dc_{t-1}}$$

$$\frac{dc_t}{dc_{t-1}} = f_t + \frac{d(i_t \cdot \widehat{c_t})}{dc_{t-1}}$$

Forget gate output...between 0 and 1

Additional gradient...can be any value... different at different time step . . .

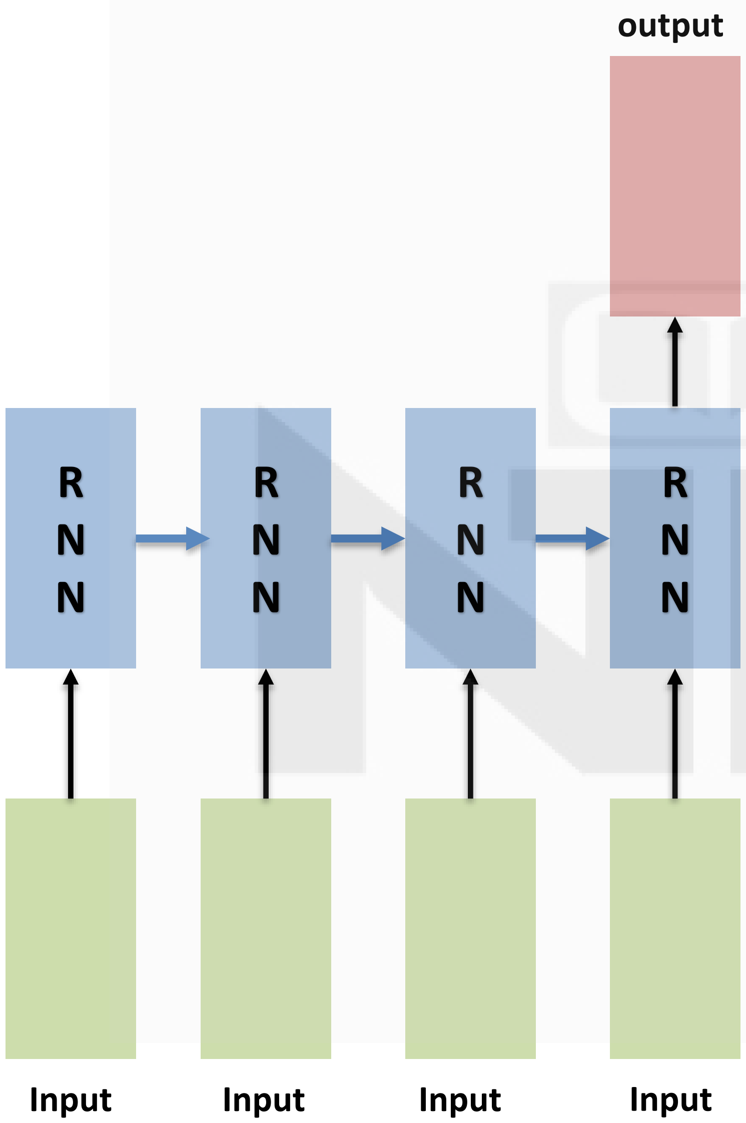
Together , These gradient terms allow LSTM to reduce vanishing gradient problem

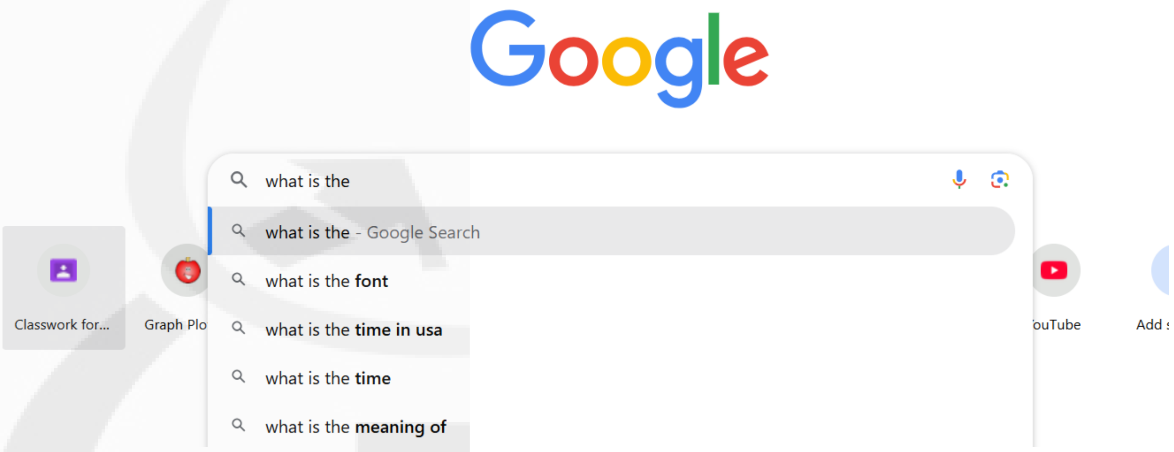
Gradient in LSTM

$$\frac{dLoss}{dw} = \frac{dLoss}{dO} \cdot \frac{dO}{dh_4} \cdot \frac{dh_4}{dc_4} \cdot \frac{dc_4}{dc_3} \cdot \frac{dc_3}{dc_2} \cdot \frac{dc_2}{dc_1} \cdot \frac{dc_1}{dw}$$

Each of these values can be less than 1 or more than 1 … reducing chance of vanishing gradient

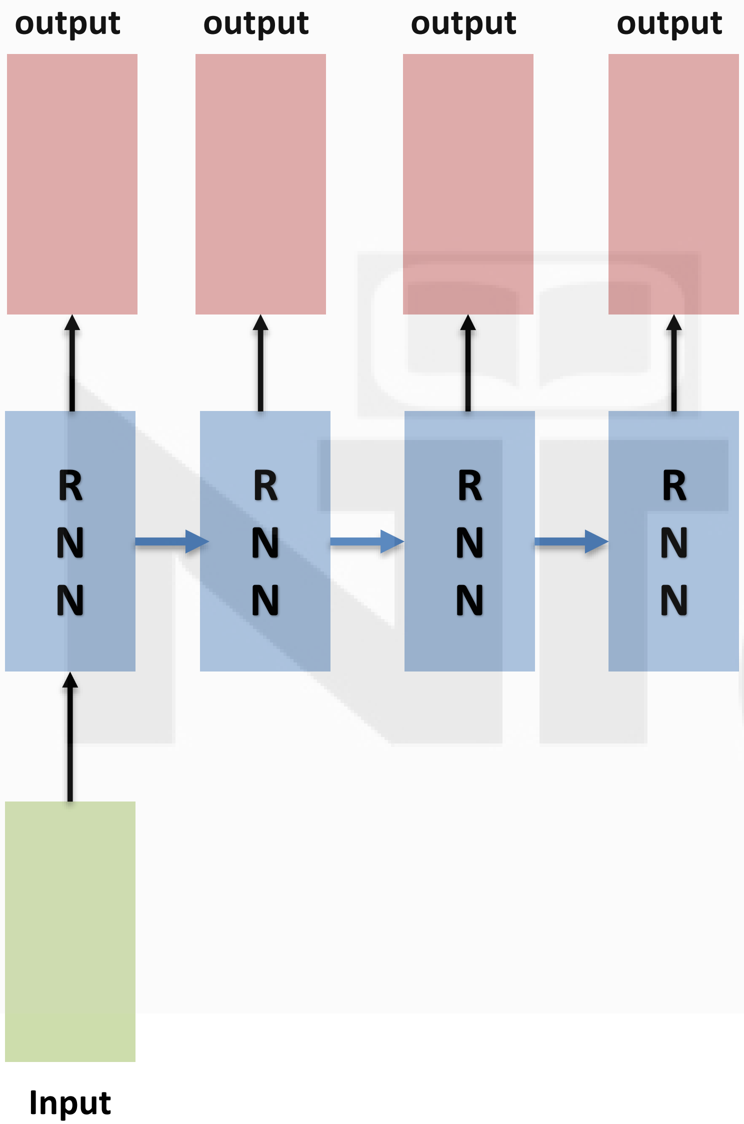
TYPES OF RNN





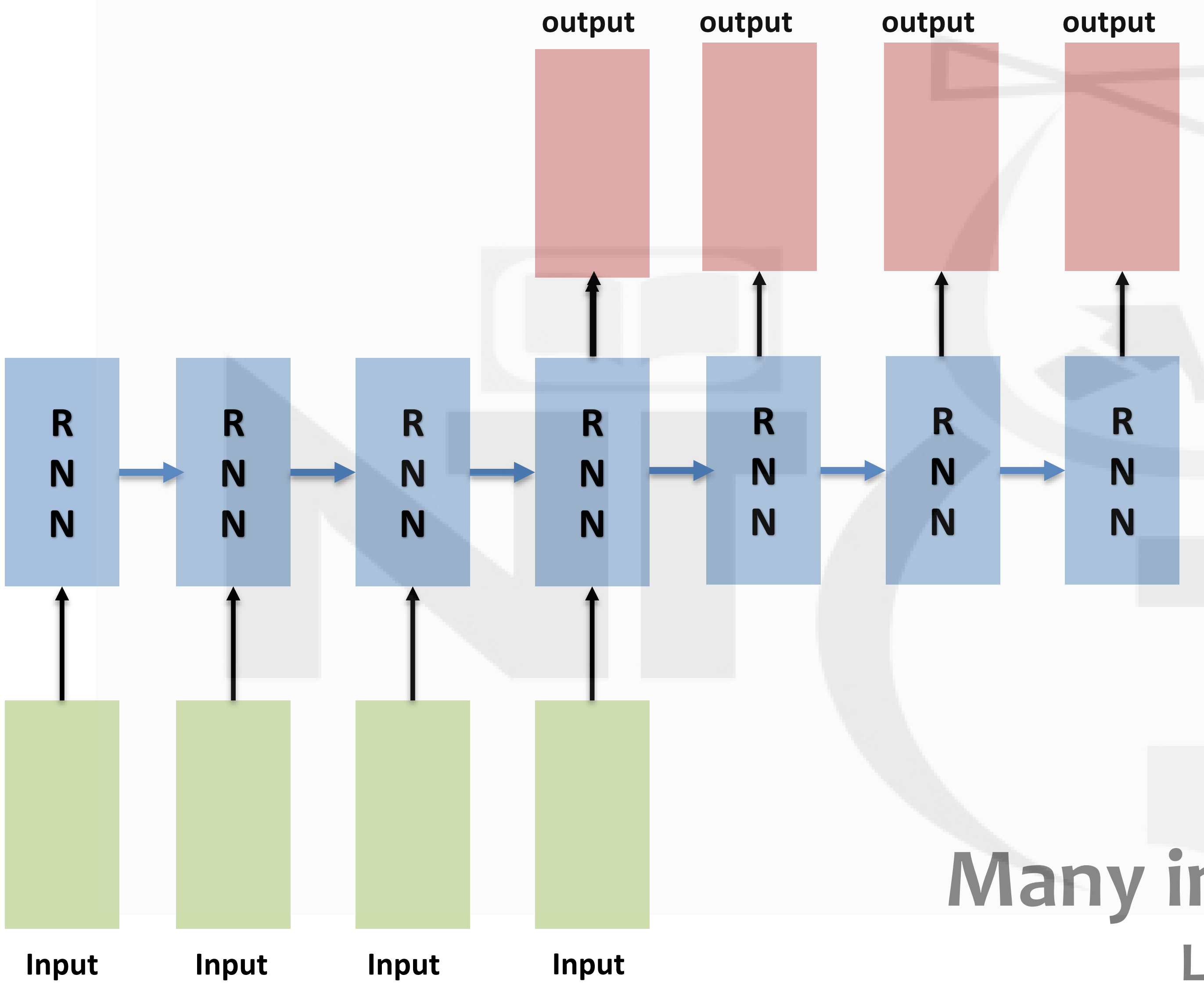
Many Inputs , One Output

Sentiment Analysis



One Input , Multiple Outputs

Image captioning



Many inputs, Many outputs

Language Translation