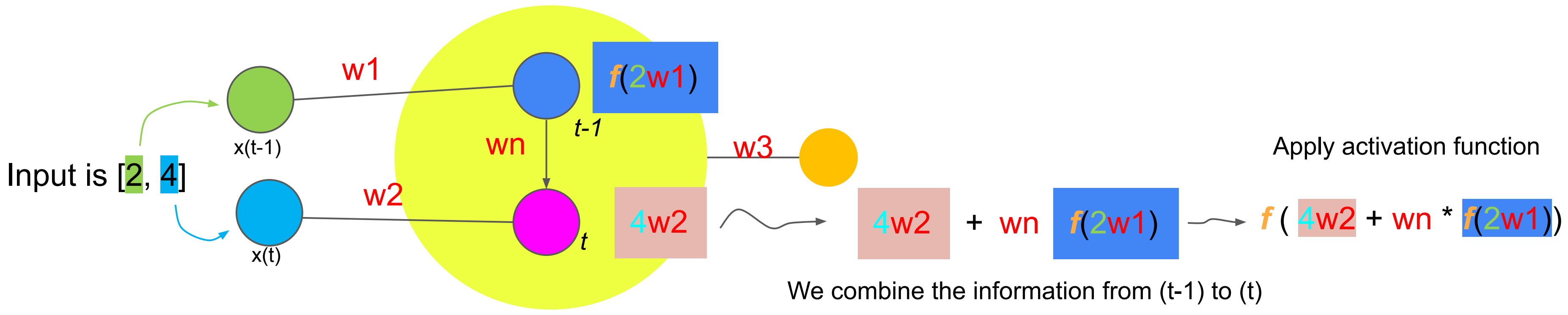
LSTM (Long short-term memory)

Difference between RNN and LSTM

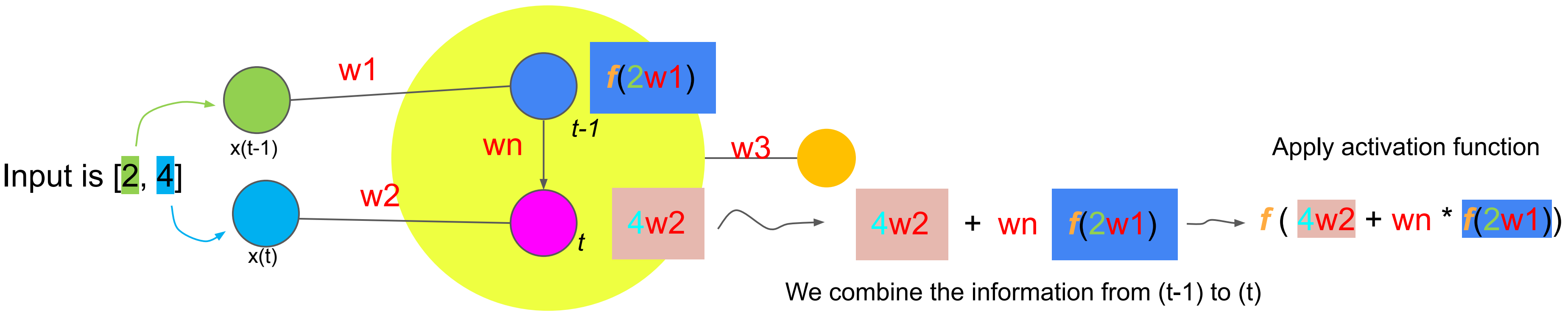
The difference between Simple RNN and LSTM

For RNN, within one neuron, all the intermediate neuron values are updated time step by time step (e.g., from (t-1) to (t)):

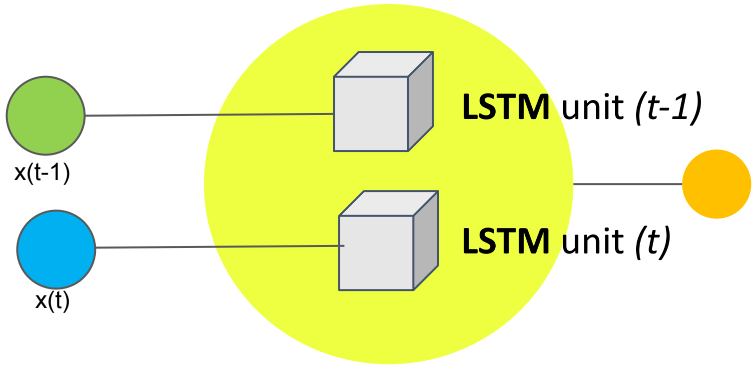


The difference between Simple RNN and LSTM

For RNN, within one neuron, all the intermediate neuron values are updated time step by time step (e.g., from (t-1) to (t)):



For LSTM, instead of an intermediate neuron, we have LSTM unit to go from one time step to the next, and we therefore avoid the use of wn, which is causing the gradient vanishing/explosion issue in RNN



How LSTM works: concept

Long term memory (t-1): zl(t-1)

Each LSTM unit must have three inputs:

- input (e.g., output from last time step)
- short term memory (updated from last step)
- long term memory (updated from last step)

Short term memory (t-1) ~ zs(t-1)

Input ~ x1 (t)

Long term memory (t-1): zl(t-1)

Each LSTM unit must have three inputs:

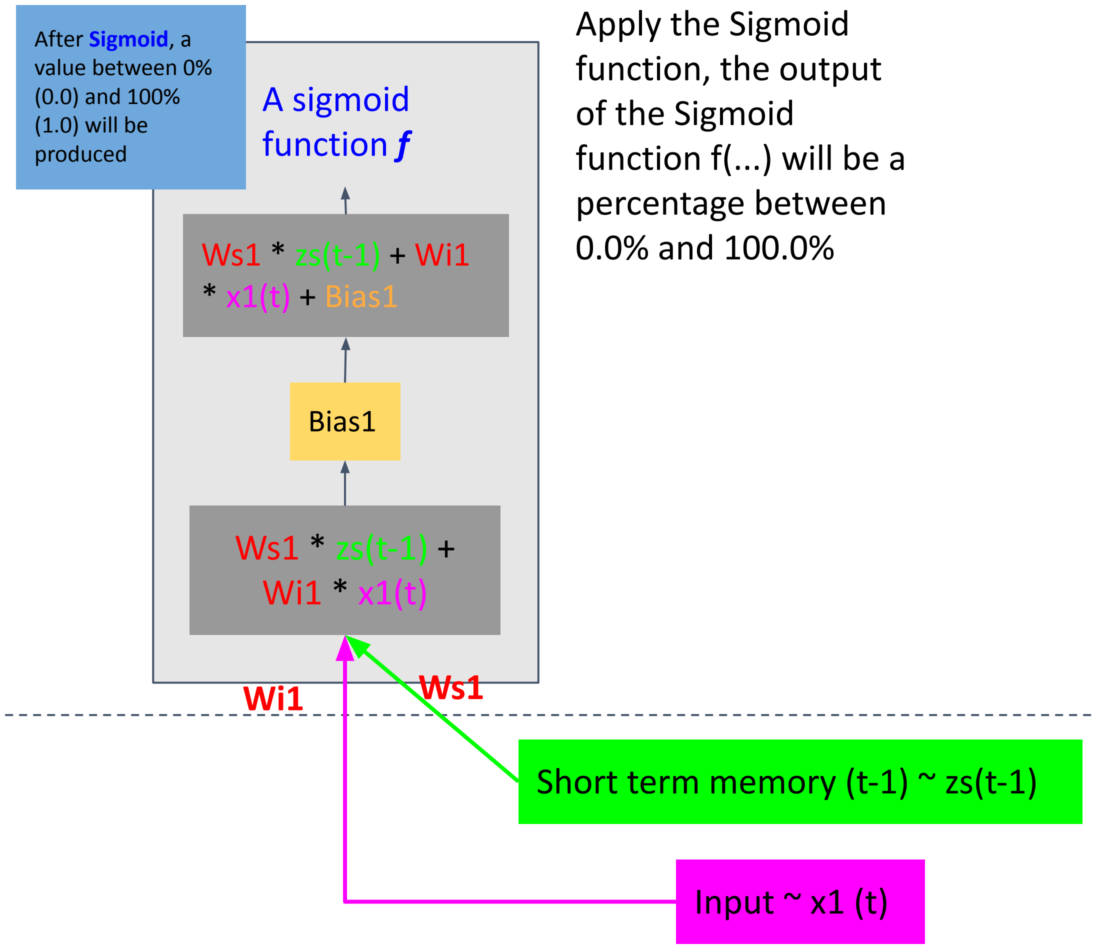
- input (e.g., output from last time step)
- short term memory (updated from last step)
- long term memory (updated from last step)


Long term memory (t-1): zl(t-1)
Each LSTM unit must have three inputs:
- input (e.g., output from last time step)
- short term memory (updated from last step)
- long term memory (updated from last step)
Ws1 * zs(t-1) + Wi1 * x1(t) + Bias1
Add bias to the combined value
Bias1
Ws1 * zs(t-1) + Wi1 * x1(t)
Wi1
Ws1
Short term memory (t-1) ~ zs(t-1)
Input ~ x1 (t)

Long term memory (t-1): zl(t-1)

Each LSTM unit must have three inputs:

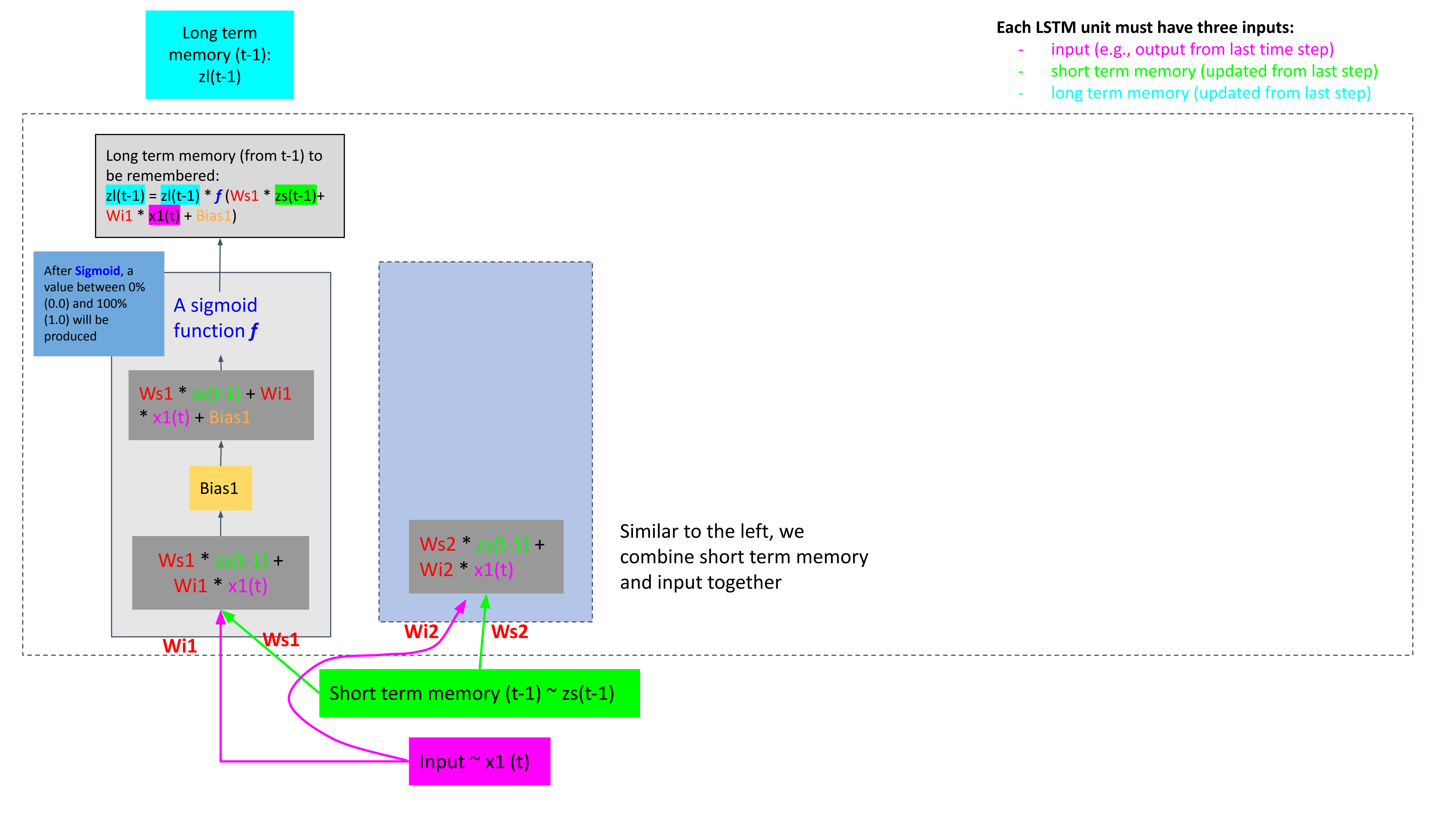
- input (e.g., output from last time step)
- short term memory (updated from last step)
- long term memory (updated from last step)

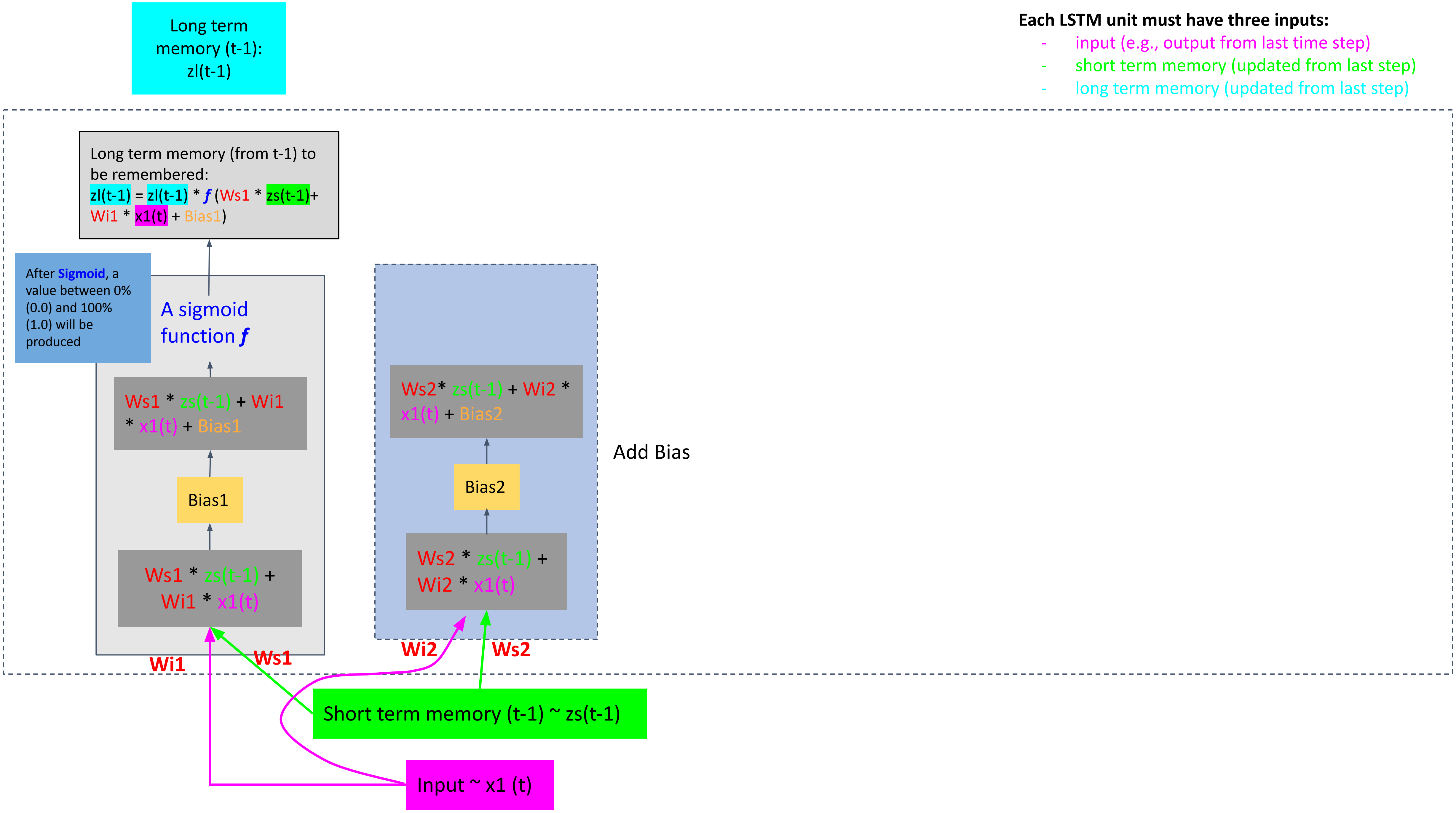


Apply the Sigmoid function, the output of the Sigmoid function f(...) will be a percentage between 0.0% and 100.0%

Each LSTM unit must have three inputs:

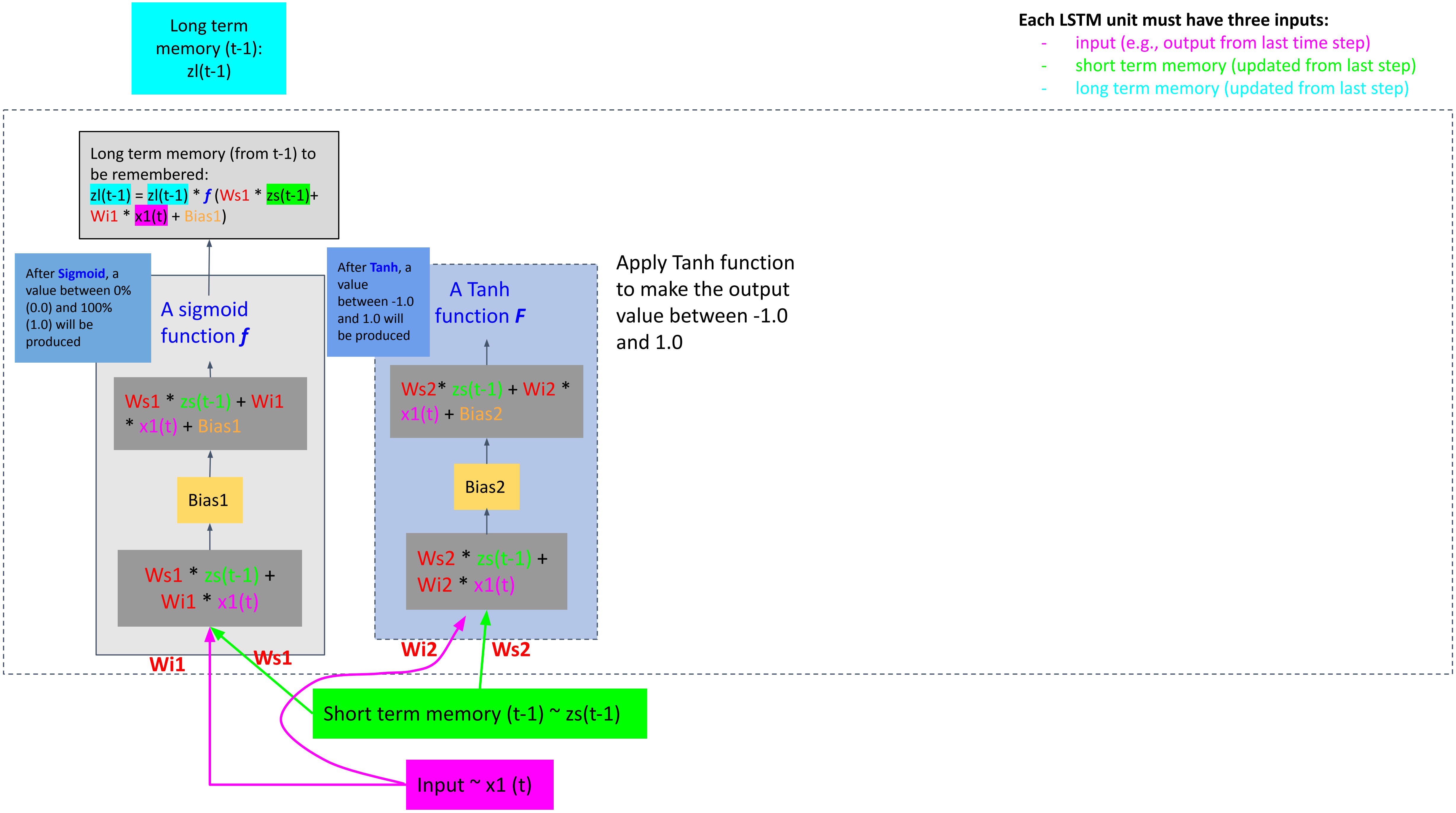
- input (e.g., output from last time step)
- short term memory (updated from last step)
- long term memory (updated from last step)

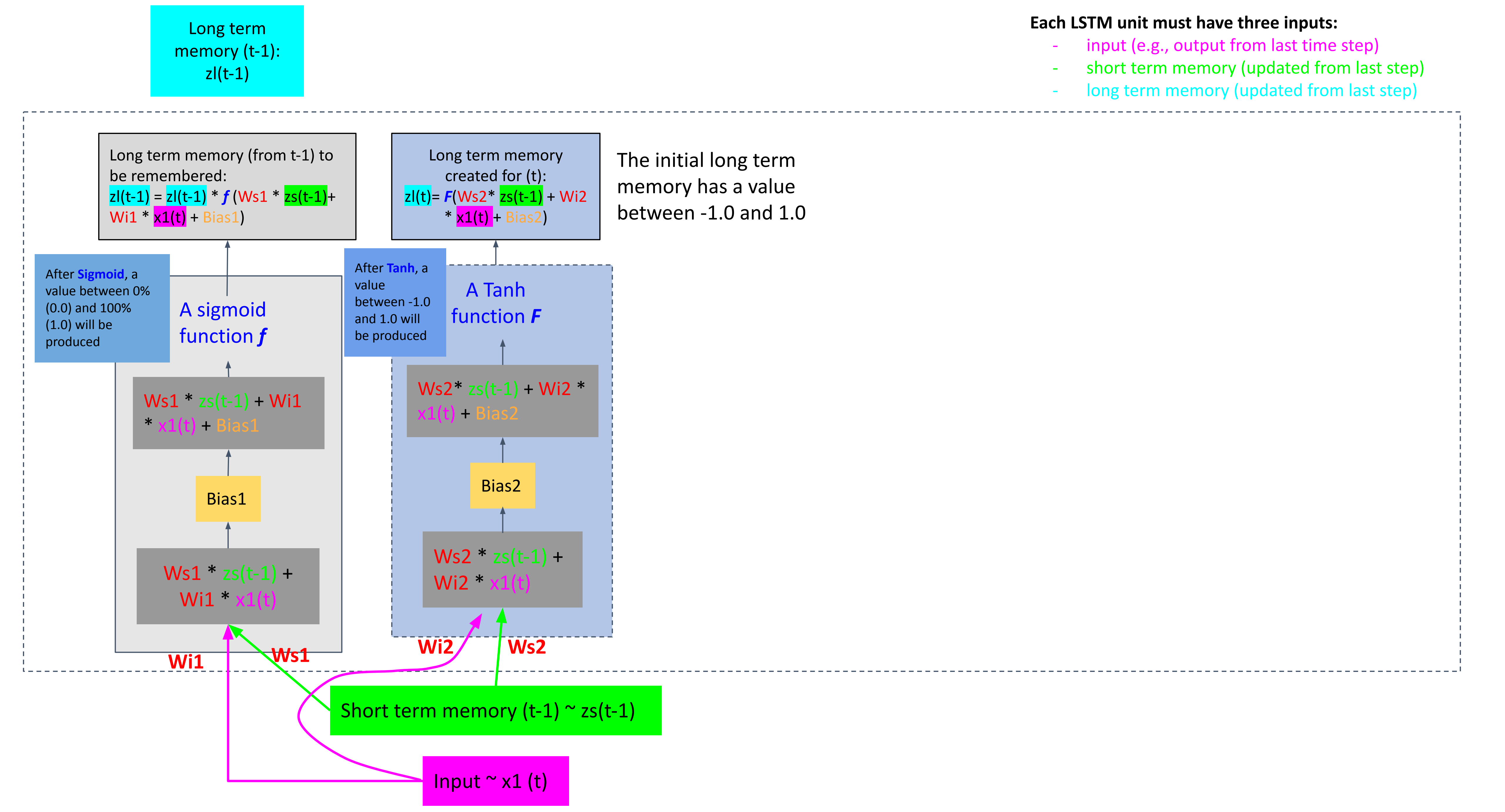

Long term memory (t-1): zl(t-1)
Each LSTM unit must have three inputs:
- input (e.g., output from last time step)
- short term memory (updated from last step)
- long term memory (updated from last step)
Long term memory (from t-1) to be remembered:
zl(t-1) = zl(t-1) * f (Ws1 * zs(t-1)+ Wi1 * x1(t) + Bias1)
After Sigmoid, a value between 0% (0.0) and 100% (1.0) will be produced
A sigmoid function f
Ws1 * zs(t-1) + Wi1 * x1(t) + Bias1
Bias1
Ws1 * zs(t-1) + Wi1 * x1(t)
Ws2 * zs(t-1) + Wi2 * x1(t)
Similar to the left, we combine short term memory and input together
Wi1
Ws1
Wi2
Ws2
Short term memory (t-1) ~ zs(t-1)
Input ~ x1 (t)


Long term memory (t-1): zl(t-1)
Each LSTM unit must have three inputs:
- input (e.g., output from last time step)
- short term memory (updated from last step)
- long term memory (updated from last step)
Long term memory (from t-1) to be remembered:
zl(t-1) = zl(t-1) * f (Ws1 * zs(t-1)+ Wi1 * x1(t) + Bias1)
After Sigmoid, a value between 0% (0.0) and 100% (1.0) will be produced
A sigmoid function f
Ws1 * zs(t-1) + Wi1 * x1(t) + Bias1
Bias1
Ws1 * zs(t-1) + Wi1 * x1(t)
Ws2* zs(t-1) + Wi2 * x1(t) + Bias2
Bias2
Ws2 * zs(t-1) + Wi2 * x1(t)
Add Bias
Wi1
Ws1
Wi2
Ws2
Short term memory (t-1) ~ zs(t-1)
Input ~ x1 (t)

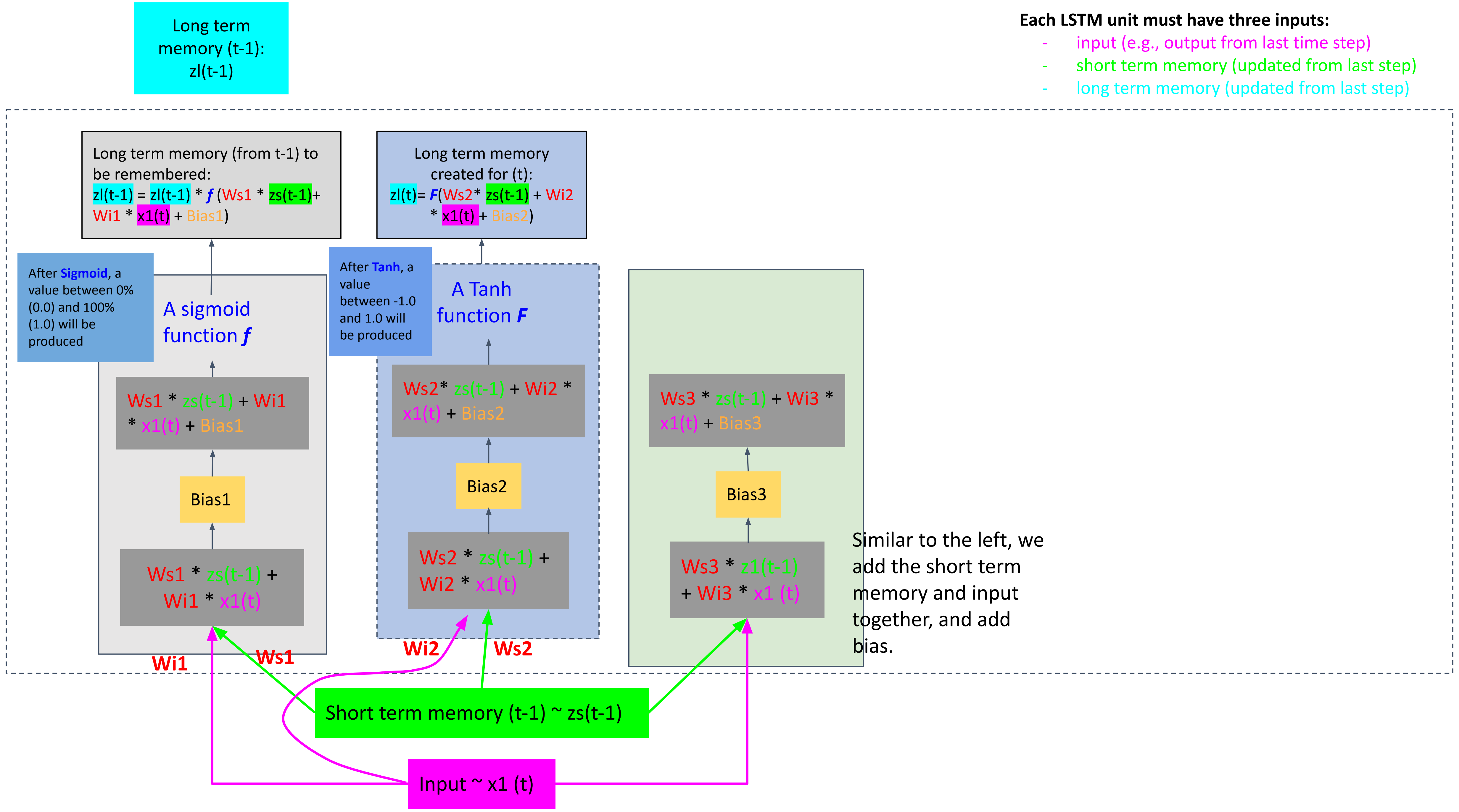
Each LSTM unit must have three inputs:

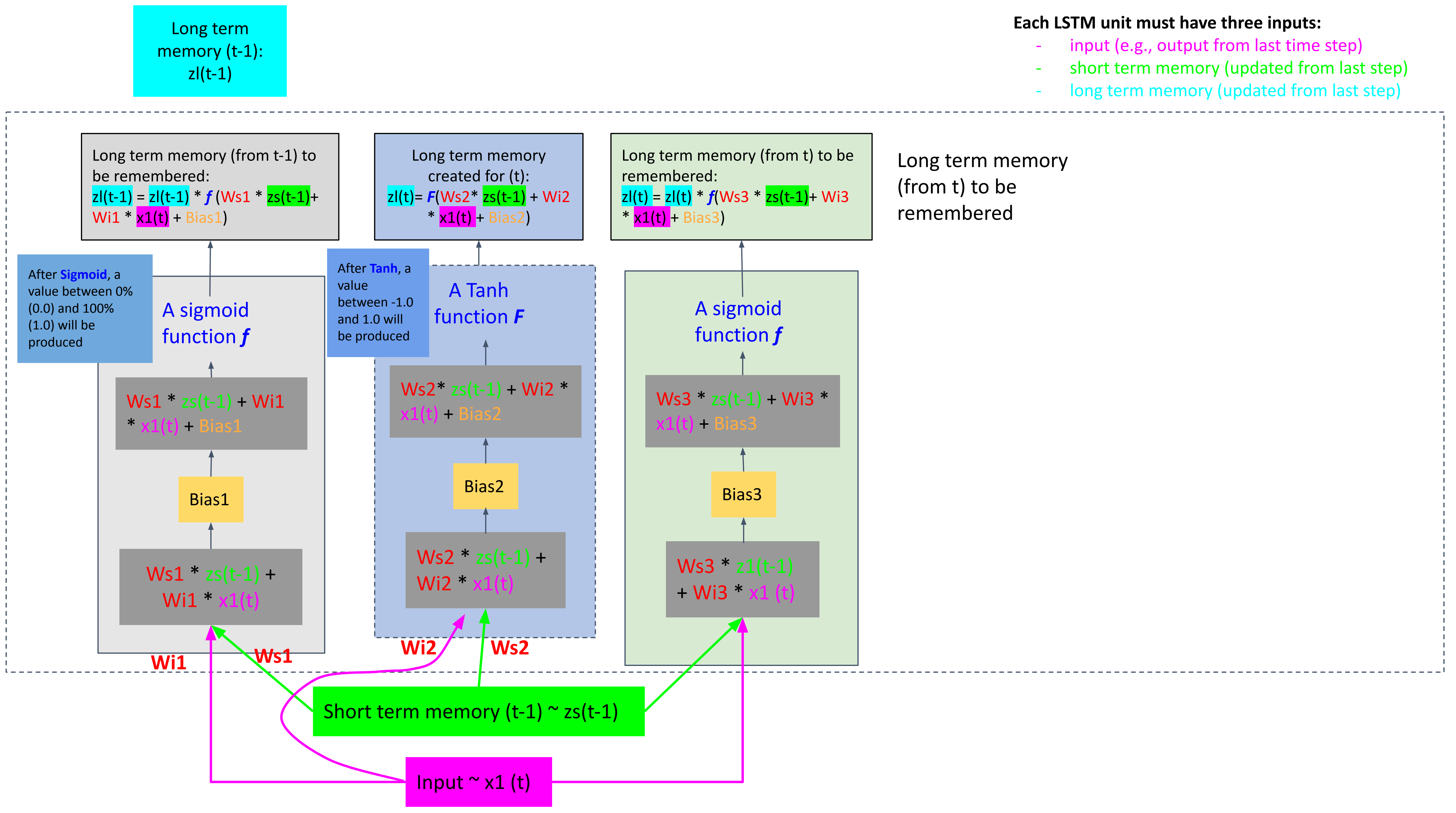
- input (e.g., output from last time step)
- short term memory (updated from last step)
- long term memory (updated from last step)

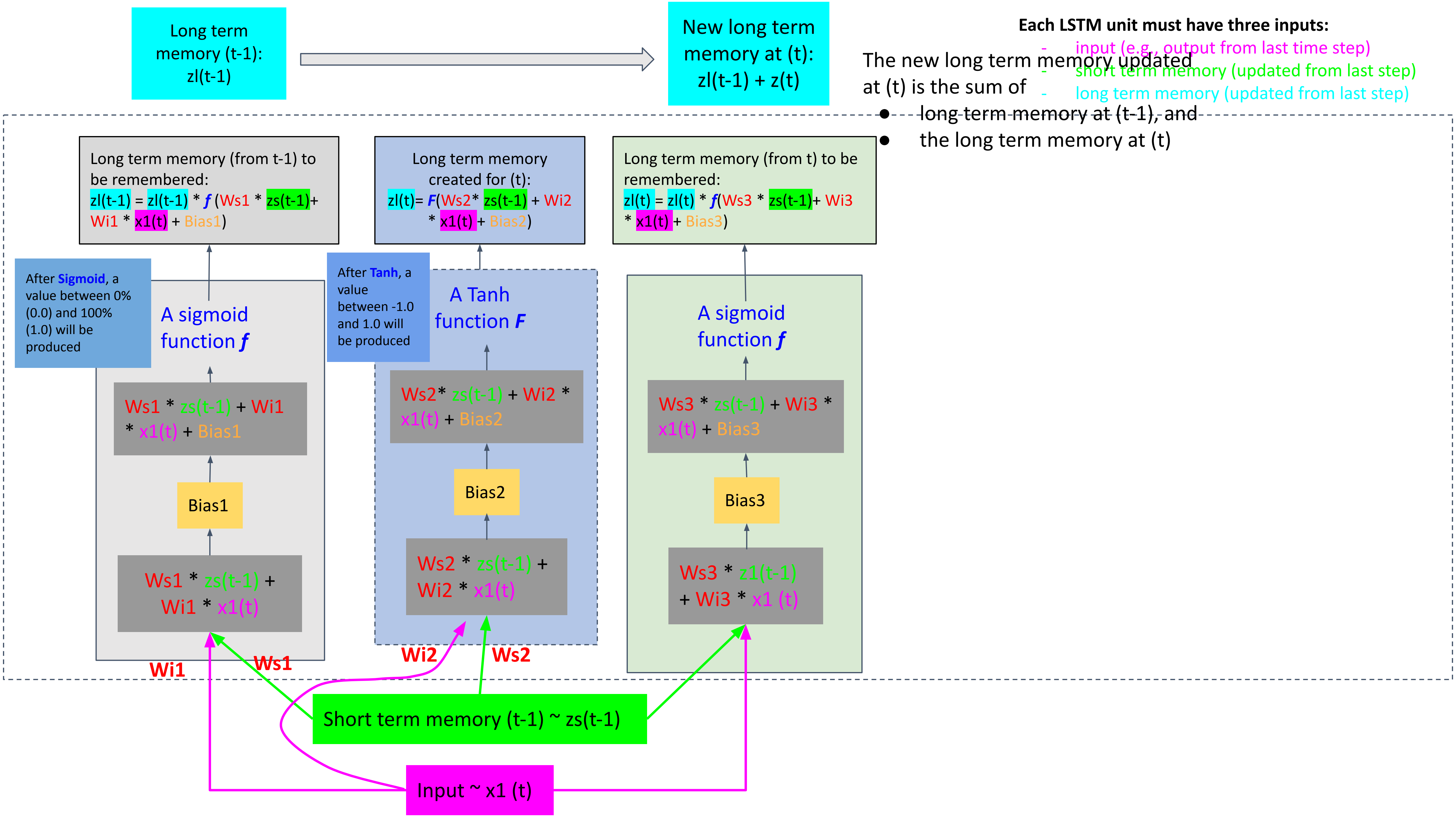


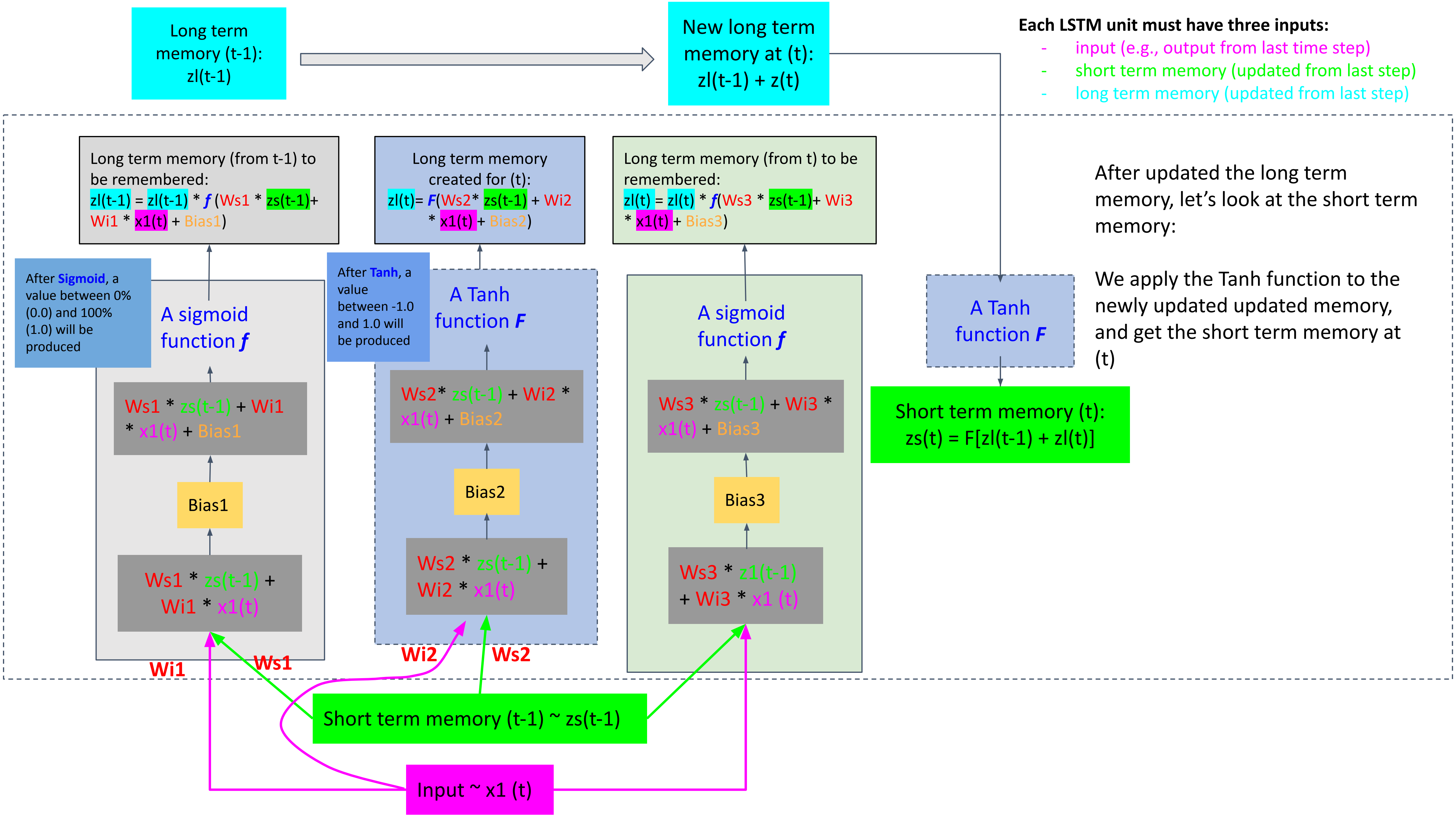


Long term memory (t-1): zl(t-1)
Each LSTM unit must have three inputs:
- input (e.g., output from last time step)
- short term memory (updated from last step)
- long term memory (updated from last step)
Long term memory (from t-1) to be remembered: zl(t-1) = zl(t-1) * f (Ws1 * zs(t-1)+ Wi1 * x1(t) + Bias1)
Long term memory created for (t): zl(t)= F(Ws2* zs(t-1) + Wi2 * x1(t) + Bias2)
The initial long term memory has a value between -1.0 and 1.0
After Sigmoid, a value between 0% (0.0) and 100% (1.0) will be produced
A sigmoid function f
After Tanh, a value between -1.0 and 1.0 will be produced
A Tanh function F
Ws1 * zs(t-1) + Wi1 * x1(t) + Bias1
Ws2* zs(t-1) + Wi2 * x1(t) + Bias2
Bias1
Bias2
Ws1 * zs(t-1) + Wi1 * x1(t)
Ws2 * zs(t-1) + Wi2 * x1(t)
Wi1
Ws1
Wi2
Ws2
Short term memory (t-1) ~ zs(t-1)
Input ~ x1 (t)

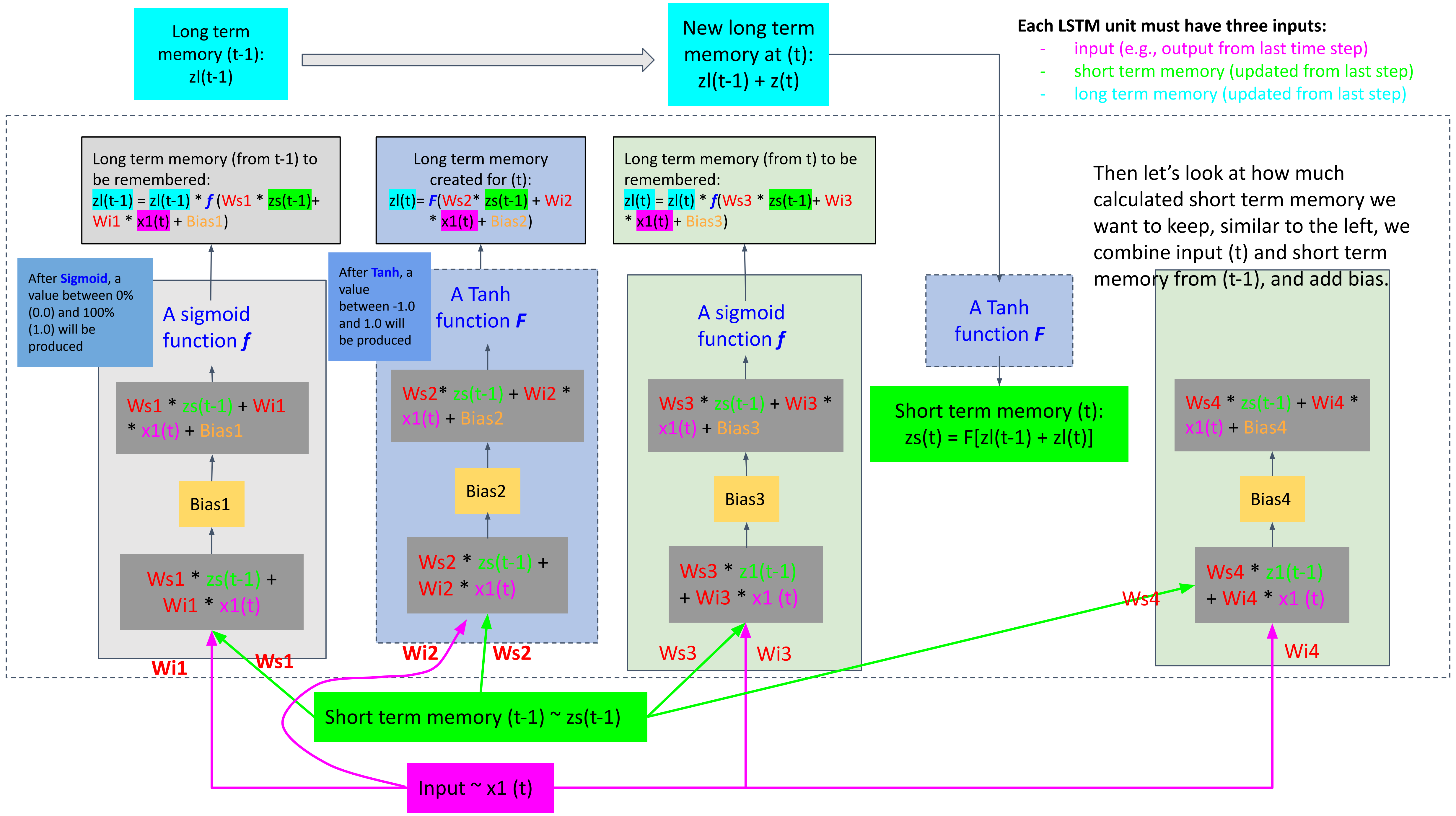

Each LSTM unit must have three inputs:
- input (e.g., output from last time step)
- short term memory (updated from last step)
- long term memory (updated from last step)
Long term memory (t-1): zl(t-1)
Long term memory (from t-1) to be remembered: zl(t-1) = zl(t-1) * f (Ws1 * zs(t-1)+ Wi1 * x1(t) + Bias1)
Long term memory created for (t): zl(t)= F(Ws2* zs(t-1) + Wi2 * x1(t) + Bias2)
After Sigmoid, a value between 0% (0.0) and 100% (1.0) will be produced
A sigmoid function f
Ws1 * zs(t-1) + Wi1 * x1(t) + Bias1
Bias1
Ws1 * zs(t-1) + Wi1 * x1(t)
After Tanh, a value between -1.0 and 1.0 will be produced
A Tanh function F
Ws2* zs(t-1) + Wi2 * x1(t) + Bias2
Bias2
Ws2 * zs(t-1) + Wi2 * x1(t)
Ws3 * zs(t-1) + Wi3 * x1(t) + Bias3
Bias3
Ws3 * z1(t-1) + Wi3 * x1 (t)
Similar to the left, we add the short term memory and input together, and add bias.
Wi1
Ws1
Wi2
Ws2
Short term memory (t-1) ~ zs(t-1)
Input ~ x1 (t)

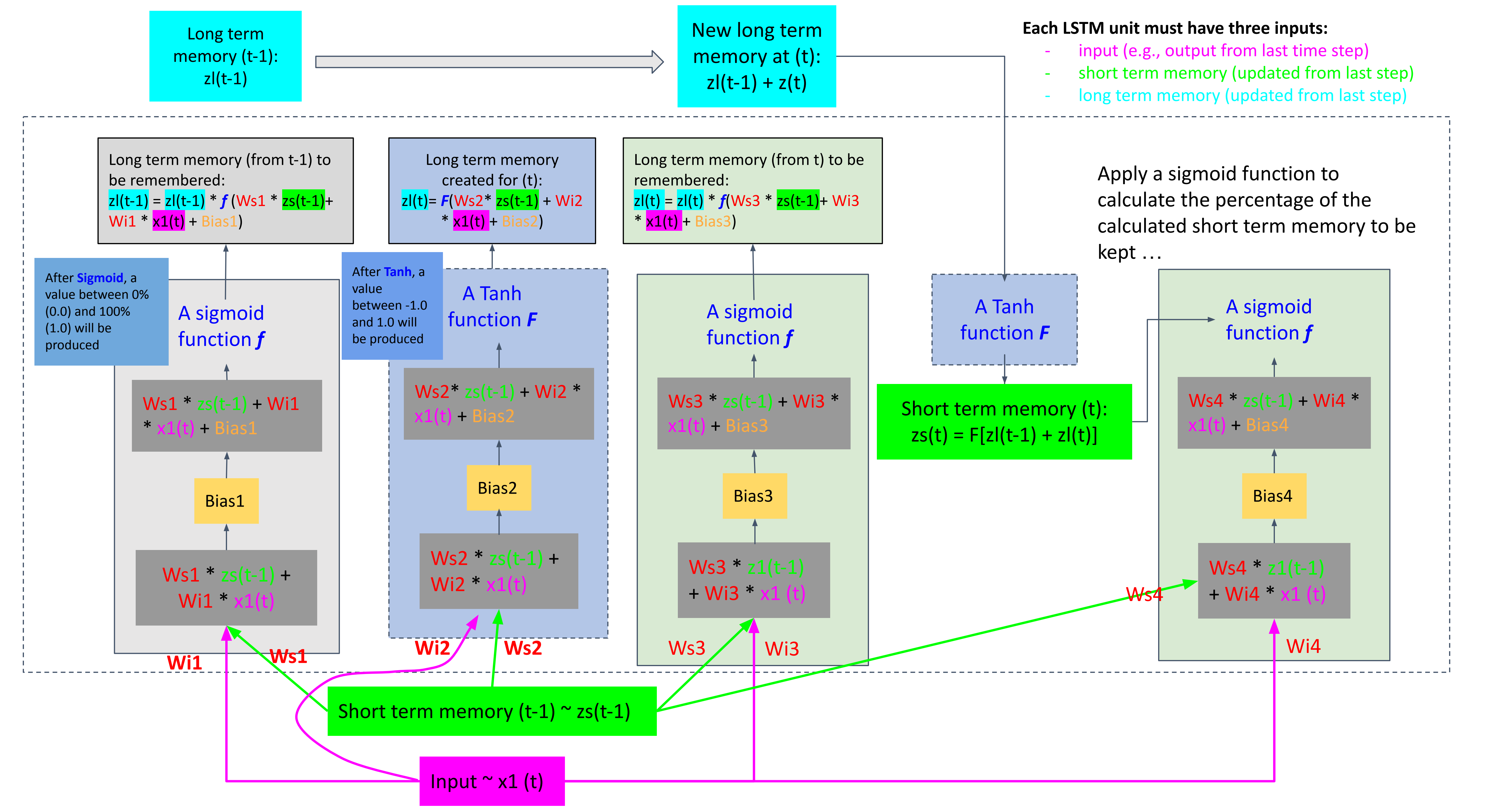

Long term memory (t-1): zl(t-1)
Each LSTM unit must have three inputs:
- input (e.g., output from last time step)
- short term memory (updated from last step)
- long term memory (updated from last step)
Long term memory (from t-1) to be remembered: zl(t-1) = zl(t-1) * f (Ws1 * zs(t-1)+ Wi1 * x1(t) + Bias1)
Long term memory created for (t): zl(t)= F(Ws2* zs(t-1) + Wi2 * x1(t) + Bias2)
Long term memory (from t) to be remembered: zl(t) = zl(t) * f(Ws3 * zs(t-1)+ Wi3 * x1(t) + Bias3)
Long term memory (from t) to be remembered
After Sigmoid, a value between 0% (0.0) and 100% (1.0) will be produced
A sigmoid function f
Ws1 * zs(t-1) + Wi1 * x1(t) + Bias1
Bias1
Ws1 * zs(t-1) + Wi1 * x1(t)
After Tanh, a value between -1.0 and 1.0 will be produced
A Tanh function F
Ws2* zs(t-1) + Wi2 * x1(t) + Bias2
Bias2
Ws2 * zs(t-1) + Wi2 * x1(t)
A sigmoid function f
Ws3 * zs(t-1) + Wi3 * x1(t) + Bias3
Bias3
Ws3 * z1(t-1) + Wi3 * x1 (t)
Wi1
Ws1
Wi2
Ws2
Short term memory (t-1) ~ zs(t-1)
Input ~ x1 (t)

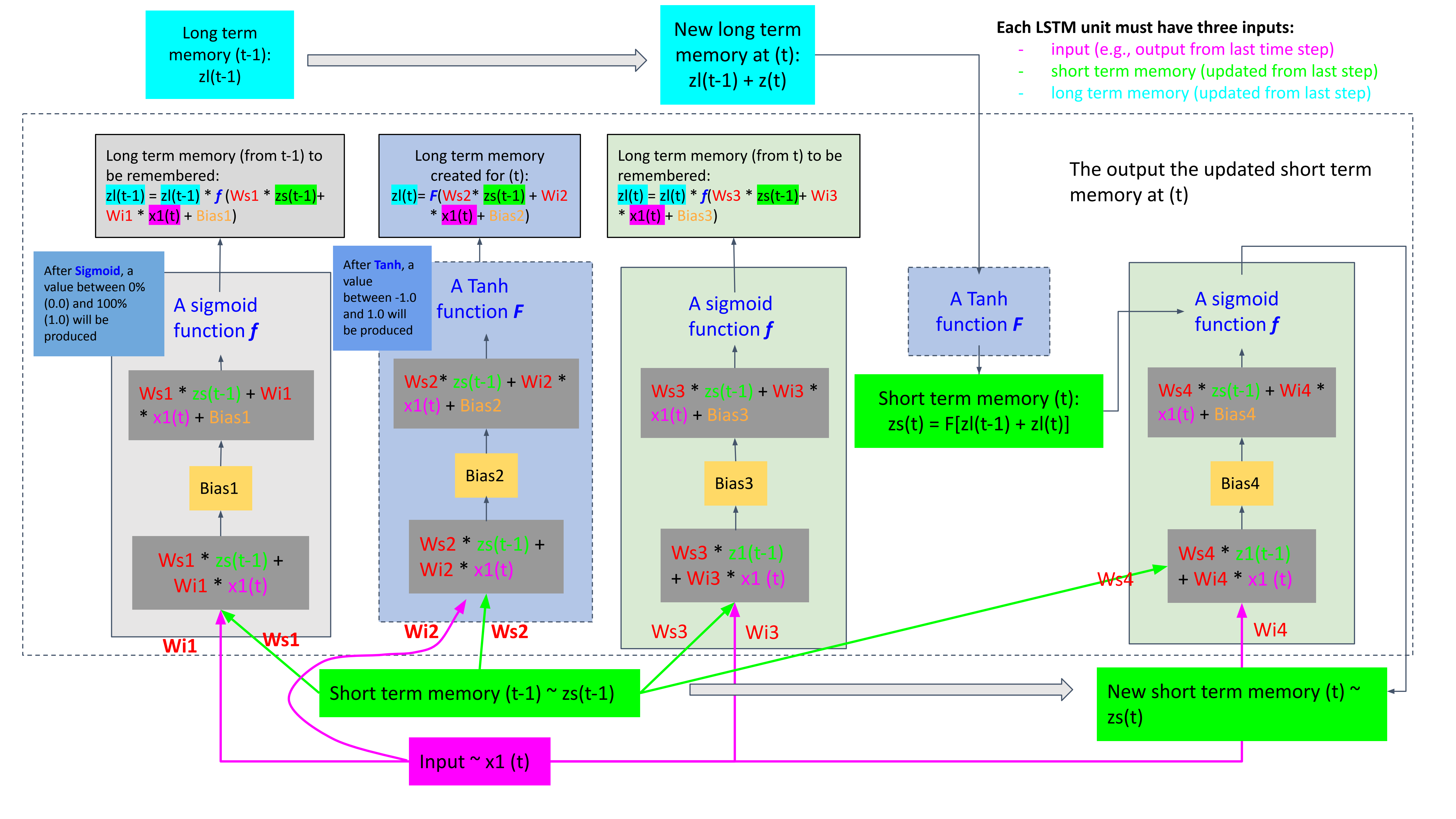

Long term memory (t-1): zl(t-1)
New long term memory at (t): zl(t-1) + z(t)
Each LSTM unit must have three inputs:
- input (e.g., output from last time step)
- short term memory (updated from last step)
- long term memory (updated from last step)
The new long term memory updated at (t) is the sum of
● long term memory at (t-1), and
● the long term memory at (t)
Long term memory (from t-1) to be remembered:
zl(t-1) = zl(t-1) * f (Ws1 * zs(t-1)+ Wi1 * x1(t) + Bias1)
Long term memory created for (t):
zl(t)= F(Ws2* zs(t-1) + Wi2 * x1(t) + Bias2)
Long term memory (from t) to be remembered:
zl(t) = zl(t) * f(Ws3 * zs(t-1)+ Wi3 * x1(t) + Bias3)
After Sigmoid, a value between 0% (0.0) and 100% (1.0) will be produced
A sigmoid function f
Ws1 * zs(t-1) + Wi1 * x1(t) + Bias1
Bias1
Ws1 * zs(t-1) + Wi1 * x1(t)
After Tanh, a value between -1.0 and 1.0 will be produced
A Tanh function F
Ws2* zs(t-1) + Wi2 * x1(t) + Bias2
Bias2
Ws2 * zs(t-1) + Wi2 * x1(t)
A sigmoid function f
Ws3 * zs(t-1) + Wi3 * x1(t) + Bias3
Bias3
Ws3 * z1(t-1) + Wi3 * x1 (t)
Wi1
Ws1
Wi2
Ws2
Short term memory (t-1) ~ zs(t-1)
Input ~ x1 (t)



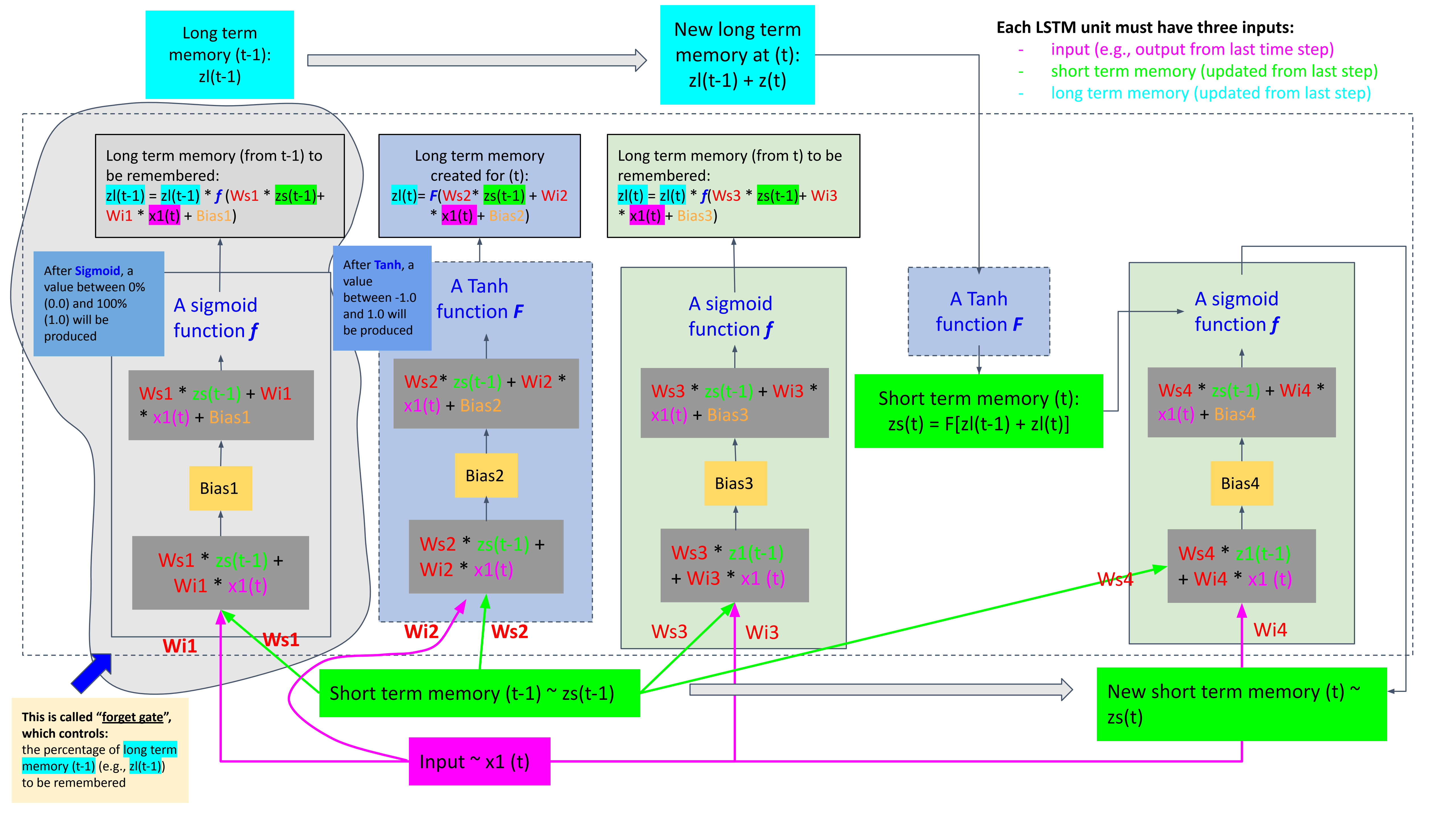
Long term memory (t-1): zl(t-1)
New long term memory at (t): zl(t-1) + z(t)
Each LSTM unit must have three inputs:
- input (e.g., output from last time step)
- short term memory (updated from last step)
- long term memory (updated from last step)
Long term memory (from t-1) to be remembered: zl(t-1) = zl(t-1) * f (Ws1 * zs(t-1)+ Wi1 * x1(t) + Bias1)
Long term memory created for (t): zl(t)= F(Ws2* zs(t-1) + Wi2 * x1(t) + Bias2)
Long term memory (from t) to be remembered: zl(t) = zl(t) * f(Ws3 * zs(t-1)+ Wi3 * x1(t) + Bias3)
After updated the long term memory, let's look at the short term memory:
We apply the Tanh function to the newly updated updated memory, and get the short term memory at (t)
After Sigmoid, a value between 0% (0.0) and 100% (1.0) will be produced
A sigmoid function f
After Tanh, a value between -1.0 and 1.0 will be produced
A Tanh function F
A sigmoid function f
A Tanh function F
Short term memory (t): zs(t) = F[zl(t-1) + zl(t)]
Ws1 * zs(t-1) + Wi1 * x1(t) + Bias1
Ws2* zs(t-1) + Wi2 * x1(t) + Bias2
Ws3 * zs(t-1) + Wi3 * x1(t) + Bias3
Bias1
Bias2
Bias3
Ws1 * zs(t-1) + Wi1 * x1(t)
Ws2 * zs(t-1) + Wi2 * x1(t)
Ws3 * z1(t-1) + Wi3 * x1 (t)
Wi1
Ws1
Wi2
Ws2
Short term memory (t-1) ~ zs(t-1)
Input ~ x1 (t)


Long term memory (t-1): zl(t-1)
New long term memory at (t): zl(t-1) + z(t)
Each LSTM unit must have three inputs:
- input (e.g., output from last time step)
- short term memory (updated from last step)
- long term memory (updated from last step)
Long term memory (from t-1) to be remembered: zl(t-1) = zl(t-1) * f (Ws1 * zs(t-1)+ Wi1 * x1(t) + Bias1)
Long term memory created for (t): zl(t)= F(Ws2* zs(t-1) + Wi2 * x1(t) + Bias2)
Long term memory (from t) to be remembered: zl(t) = zl(t) * f(Ws3 * zs(t-1)+ Wi3 * x1(t) + Bias3)
Then let's look at how much calculated short term memory we want to keep, similar to the left, we combine input (t) and short term memory from (t-1), and add bias.
After Sigmoid, a value between 0% (0.0) and 100% (1.0) will be produced
A sigmoid function f
After Tanh, a value between -1.0 and 1.0 will be produced
A Tanh function F
A sigmoid function f
A Tanh function F
Ws1 * zs(t-1) + Wi1 * x1(t) + Bias1
Ws2* zs(t-1) + Wi2 * x1(t) + Bias2
Ws3 * zs(t-1) + Wi3 * x1(t) + Bias3
Short term memory (t): zs(t) = F[zl(t-1) + zl(t)]
Ws4 * zs(t-1) + Wi4 * x1(t) + Bias4
Bias1
Bias2
Bias3
Bias4
Ws1 * zs(t-1) + Wi1 * x1(t)
Ws2 * zs(t-1) + Wi2 * x1(t)
Ws3 * z1(t-1) + Wi3 * x1 (t)
Ws4 * z1(t-1) + Wi4 * x1 (t)
Wi1
Ws1
Wi2
Ws2
Ws3
Wi3
Ws4
Wi4
Short term memory (t-1) ~ zs(t-1)
Input ~ x1 (t)

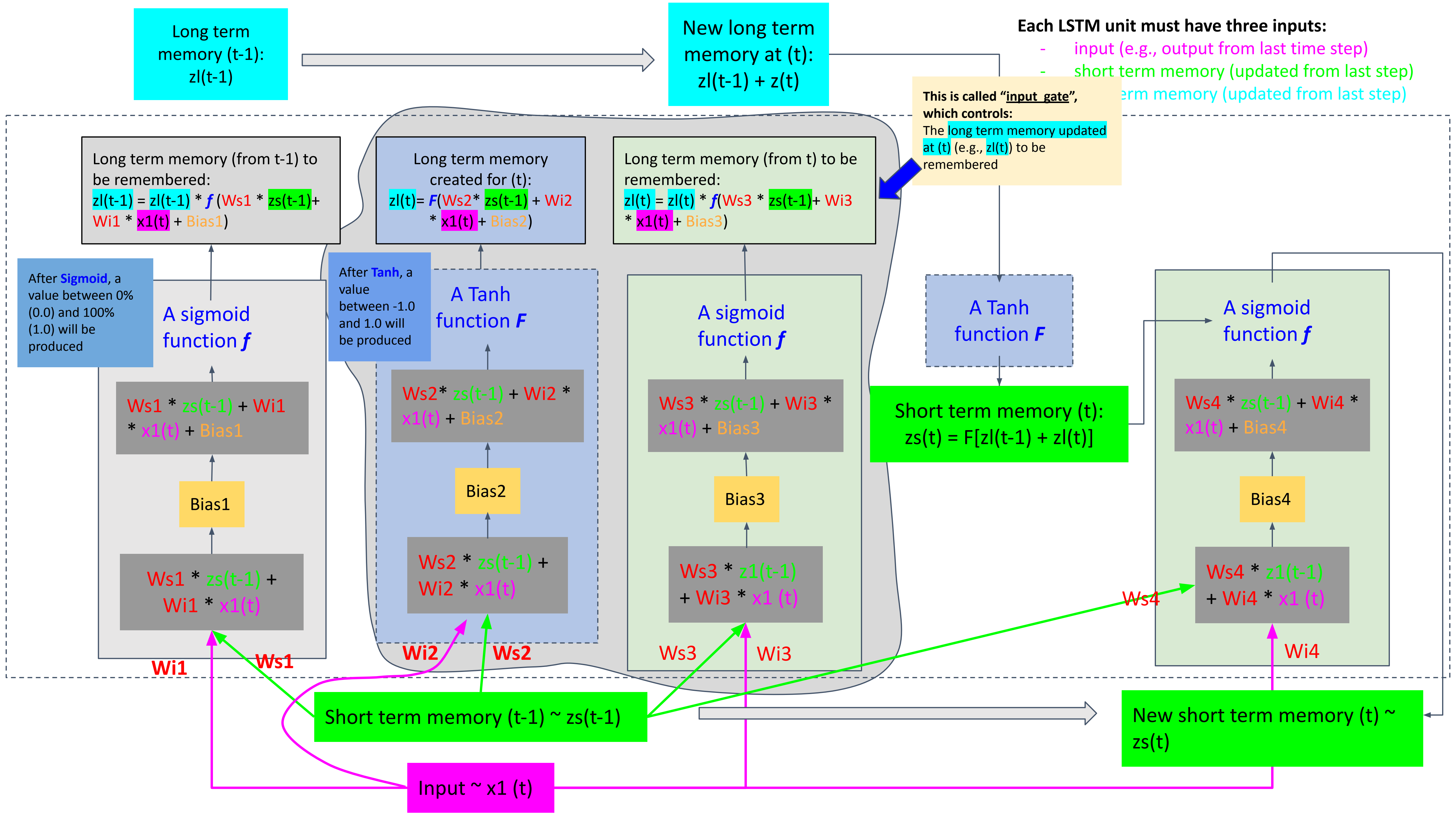

Long term memory (t-1): zl(t-1)
New long term memory at (t): zl(t-1) + z(t)
Each LSTM unit must have three inputs:
- input (e.g., output from last time step)
- short term memory (updated from last step)
- long term memory (updated from last step)
Long term memory (from t-1) to be remembered: zl(t-1) = zl(t-1) * f (Ws1 * zs(t-1)+ Wi1 * x1(t) + Bias1)
Long term memory created for (t): zl(t)= F(Ws2* zs(t-1) + Wi2 * x1(t) + Bias2)
Long term memory (from t) to be remembered: zl(t) = zl(t) * f(Ws3 * zs(t-1)+ Wi3 * x1(t) + Bias3)
Apply a sigmoid function to calculate the percentage of the calculated short term memory to be kept …
After Sigmoid, a value between 0% (0.0) and 100% (1.0) will be produced
A sigmoid function f
Ws1 * zs(t-1) + Wi1 * x1(t) + Bias1
Bias1
Ws1 * zs(t-1) + Wi1 * x1(t)
After Tanh, a value between -1.0 and 1.0 will be produced
A Tanh function F
Ws2* zs(t-1) + Wi2 * x1(t) + Bias2
Bias2
Ws2 * zs(t-1) + Wi2 * x1(t)
A sigmoid function f
Ws3 * zs(t-1) + Wi3 * x1(t) + Bias3
Bias3
Ws3 * z1(t-1) + Wi3 * x1 (t)
A Tanh function F
Short term memory (t): zs(t) = F[zl(t-1) + zl(t)]
A sigmoid function f
Ws4 * zs(t-1) + Wi4 * x1(t) + Bias4
Bias4
Ws4 * z1(t-1) + Wi4 * x1 (t)
Wi1
Ws1
Wi2
Ws2
Ws3
Wi3
Ws4
Wi4
Short term memory (t-1) ~ zs(t-1)
Input ~ x1 (t)

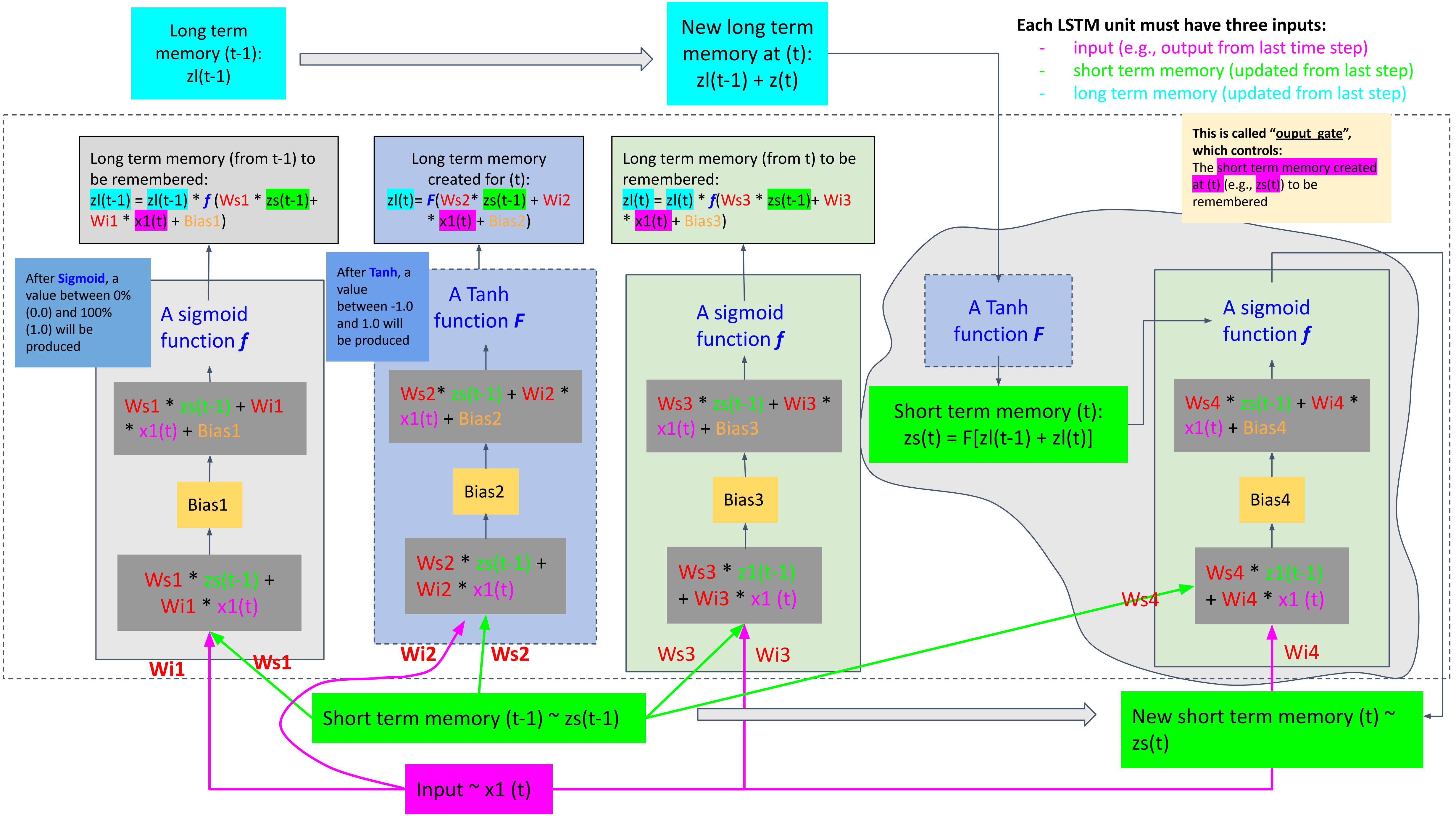


Long term memory (t-1): zl(t-1)
New long term memory at (t): zl(t-1) + z(t)
Each LSTM unit must have three inputs:
- input (e.g., output from last time step)
- short term memory (updated from last step)
- long term memory (updated from last step)
Long term memory (from t-1) to be remembered: zl(t-1) = zl(t-1) * f (Ws1 * zs(t-1)+ Wi1 * x1(t) + Bias1)
Long term memory created for (t): zl(t)= F(Ws2* zs(t-1) + Wi2 * x1(t) + Bias2)
Long term memory (from t) to be remembered: zl(t) = zl(t) * f(Ws3 * zs(t-1)+ Wi3 * x1(t) + Bias3)
The output the updated short term memory at (t)
After Sigmoid, a value between 0% (0.0) and 100% (1.0) will be produced
A sigmoid function f
Ws1 * zs(t-1) + Wi1 * x1(t) + Bias1
Bias1
Ws1 * zs(t-1) + Wi1 * x1(t)
After Tanh, a value between -1.0 and 1.0 will be produced
A Tanh function F
Ws2* zs(t-1) + Wi2 * x1(t) + Bias2
Bias2
Ws2 * zs(t-1) + Wi2 * x1(t)
A sigmoid function f
Ws3 * zs(t-1) + Wi3 * x1(t) + Bias3
Bias3
Ws3 * z1(t-1) + Wi3 * x1 (t)
A Tanh function F
Short term memory (t): zs(t) = F[zl(t-1) + zl(t)]
A sigmoid function f
Ws4 * zs(t-1) + Wi4 * x1(t) + Bias4
Bias4
Ws4 * z1(t-1) + Wi4 * x1 (t)
Wi1
Ws1
Wi2
Ws2
Ws3
Wi3
Ws4
Wi4
Short term memory (t-1) ~ zs(t-1)
New short term memory (t) ~ zs(t)
Input ~ x1 (t)



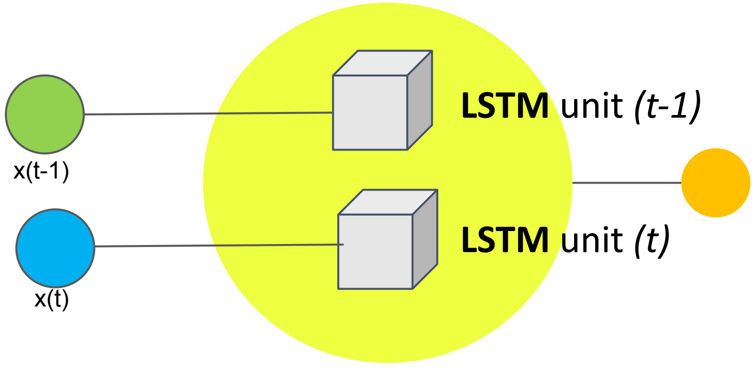
Long term memory (t-1): zl(t-1)
New long term memory at (t): zl(t-1) + z(t)
Each LSTM unit must have three inputs:
- input (e.g., output from last time step)
- short term memory (updated from last step)
- long term memory (updated from last step)
Long term memory (from t-1) to be remembered: zl(t-1) = zl(t-1) * f (Ws1 * zs(t-1)+ Wi1 * x1(t) + Bias1)
Long term memory created for (t): zl(t)= F(Ws2* zs(t-1) + Wi2 * x1(t) + Bias2)
Long term memory (from t) to be remembered: zl(t) = zl(t) * f(Ws3 * zs(t-1)+ Wi3 * x1(t) + Bias3)
After Sigmoid, a value between 0% (0.0) and 100% (1.0) will be produced
A sigmoid function f
Ws1 * zs(t-1) + Wi1 * x1(t) + Bias1
Bias1
Ws1 * zs(t-1) + Wi1 * x1(t)
Wi1
Ws1
After Tanh, a value between -1.0 and 1.0 will be produced
A Tanh function F
Ws2* zs(t-1) + Wi2 * x1(t) + Bias2
Bias2
Ws2 * zs(t-1) + Wi2 * x1(t)
Wi2
Ws2
A sigmoid function f
Ws3 * zs(t-1) + Wi3 * x1(t) + Bias3
Bias3
Ws3 * z1(t-1) + Wi3 * x1 (t)
Ws3
Wi3
A Tanh function F
Short term memory (t): zs(t) = F[zl(t-1) + zl(t)]
A sigmoid function f
Ws4 * zs(t-1) + Wi4 * x1(t) + Bias4
Bias4
Ws4 * z1(t-1) + Wi4 * x1 (t)
Ws4
Wi4
Short term memory (t-1) ~ zs(t-1)
New short term memory (t) ~ zs(t)
Input ~ x1 (t)
This is called "forget gate", which controls: the percentage of long term memory (t-1) (e.g., zl(t-1)) to be remembered


Long term memory (t-1): zl(t-1)
New long term memory at (t): zl(t-1) + z(t)
Each LSTM unit must have three inputs:
- input (e.g., output from last time step)
- short term memory (updated from last step)
erm memory (updated from last step)
This is called "input_gate", which controls:
The long term memory updated at (t) (e.g., zl(t)) to be remembered
Long term memory (from t-1) to be remembered:
zl(t-1) = zl(t-1) * f (Ws1 * zs(t-1)+ Wi1 * x1(t) + Bias1)
Long term memory created for (t):
zl(t)= F(Ws2* zs(t-1) + Wi2 * x1(t) + Bias2)
Long term memory (from t) to be remembered:
zl(t) = zl(t) * f(Ws3 * zs(t-1)+ Wi3 * x1(t) + Bias3)
After Sigmoid, a value between 0% (0.0) and 100% (1.0) will be produced
After Tanh, a value between -1.0 and 1.0 will be produced
A sigmoid function f
A Tanh function F
A sigmoid function f
A Tanh function F
A sigmoid function f
Ws1 * zs(t-1) + Wi1 * x1(t) + Bias1
Ws2* zs(t-1) + Wi2 * x1(t) + Bias2
Ws3 * zs(t-1) + Wi3 * x1(t) + Bias3
Ws4 * zs(t-1) + Wi4 * x1(t) + Bias4
Short term memory (t): zs(t) = F[zl(t-1) + zl(t)]
Bias1
Bias2
Bias3
Bias4
Ws1 * zs(t-1) + Wi1 * x1(t)
Ws2 * zs(t-1) + Wi2 * x1(t)
Ws3 * z1(t-1) + Wi3 * x1 (t)
Ws4 * z1(t-1) + Wi4 * x1 (t)
Wi1
Ws1
Wi2
Ws2
Ws3
Wi3
Ws4
Wi4
Short term memory (t-1) ~ zs(t-1)
New short term memory (t) ~ zs(t)
Input ~ x1 (t)



Long term memory (t-1): zl(t-1)
New long term memory at (t): zl(t-1) + z(t)
Each LSTM unit must have three inputs:
- input (e.g., output from last time step)
- short term memory (updated from last step)
- long term memory (updated from last step)
Long term memory (from t-1) to be remembered: zl(t-1) = zl(t-1) * f (Ws1 * zs(t-1)+ Wi1 * x1(t) + Bias1)
Long term memory created for (t): zl(t)= F(Ws2* zs(t-1) + Wi2 * x1(t) + Bias2)
Long term memory (from t) to be remembered: zl(t) = zl(t) * f(Ws3 * zs(t-1)+ Wi3 * x1(t) + Bias3)
This is called "ouput_gate", which controls: The short term memory created at (t) (e.g., zs(t)) to be remembered
After Sigmoid, a value between 0% (0.0) and 100% (1.0) will be produced
A sigmoid function f
Ws1 * zs(t-1) + Wi1 * x1(t) + Bias1
Bias1
Ws1 * zs(t-1) + Wi1 * x1(t)
After Tanh, a value between -1.0 and 1.0 will be produced
A Tanh function F
Ws2* zs(t-1) + Wi2 * x1(t) + Bias2
Bias2
Ws2 * zs(t-1) + Wi2 * x1(t)
A sigmoid function f
Ws3 * zs(t-1) + Wi3 * x1(t) + Bias3
Bias3
Ws3 * z1(t-1) + Wi3 * x1 (t)
A Tanh function F
Short term memory (t): zs(t) = F[zl(t-1) + zl(t)]
A sigmoid function f
Ws4 * zs(t-1) + Wi4 * x1(t) + Bias4
Bias4
Ws4 * z1(t-1) + Wi4 * x1 (t)
Wi1
Ws1
Wi2
Ws2
Ws3
Wi3
Ws4
Wi4
Short term memory (t-1) ~ zs(t-1)
New short term memory (t) ~ zs(t)
Input ~ x1 (t)

From the above we understand that the workflow in the LSTM unit:



By "**Forget gate**":

Step 1: Calculating the Long term memory from (t-1) to be kept

Step 2: Creating the Long term memory at (t)

By "**Input gate**":

Step 3: Calculating the Long term memory from (t) to be kept

Step 4: Updating the Long term memory at (t)

By "**Output gate**":

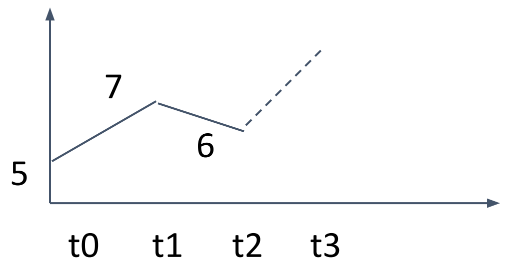
Step 5: Creating the Short term memory at (t)

Step 6: Calculating the Short term memory from (t) to be kept

Step 7: Updating the Short term memory at (t)

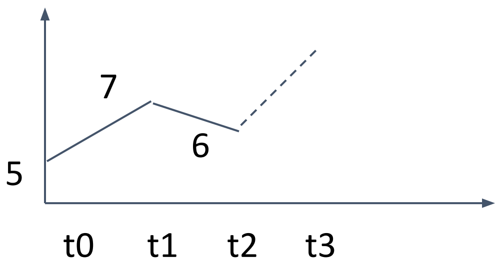
How LSTM works: a real case

How it works in a real data time series

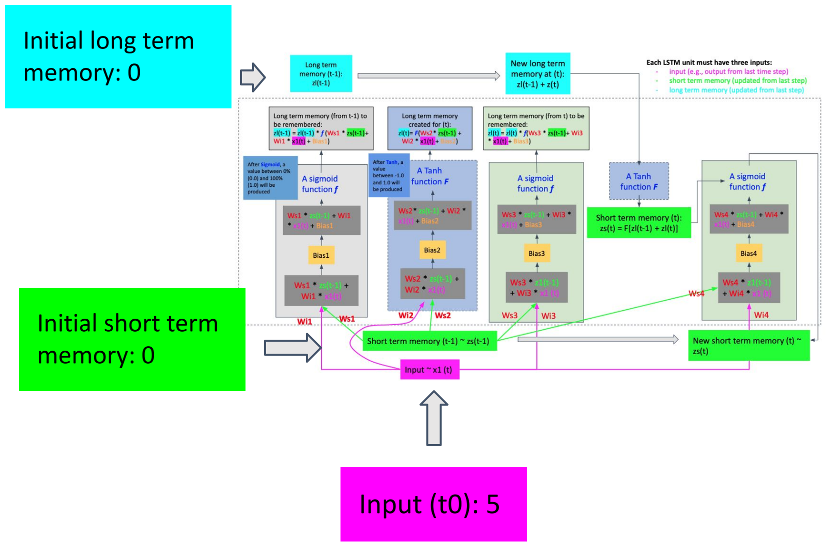


Assuming we have a time series from t0 to t2, and we would like to predict the value at t3

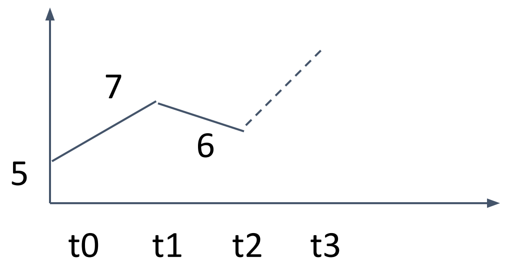
How it works in a real data time series



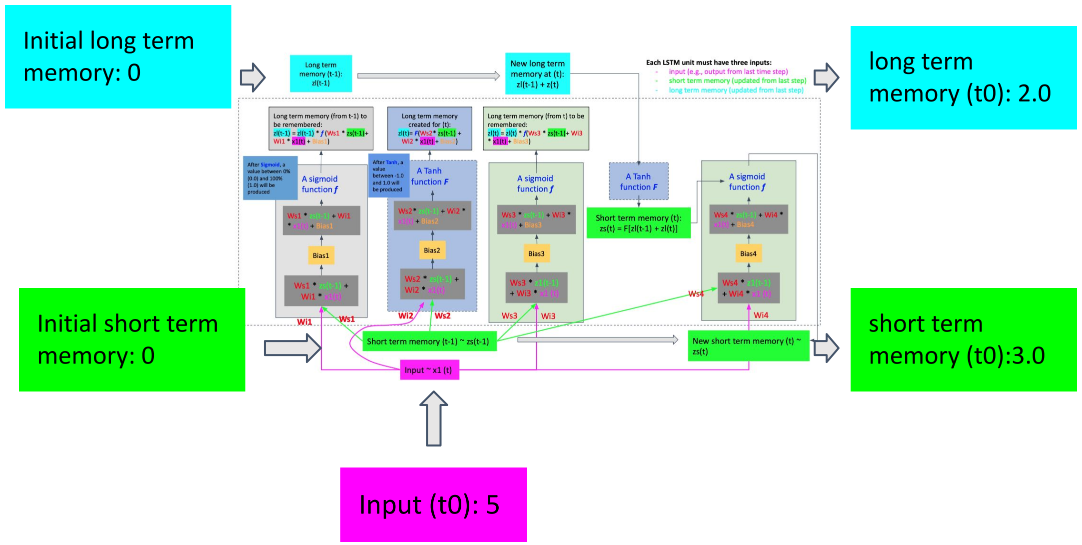
Assuming we have a time series from t0 to t2, and we would like to predict the value at t3



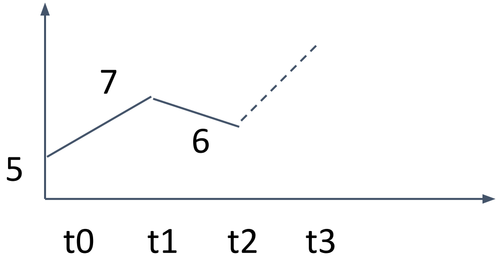
How it works in a real data time series



Assuming we have a time series from t0 to t2, and we would like to predict the value at t3



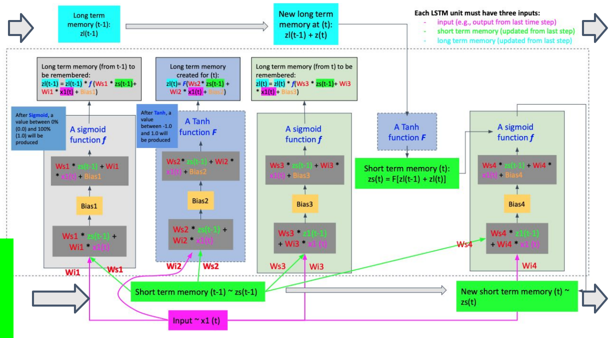
How it works in a real data time series



Assuming we have a time series from t0 to t2, and we would like to predict the value at t3

Initial long term memory: 0

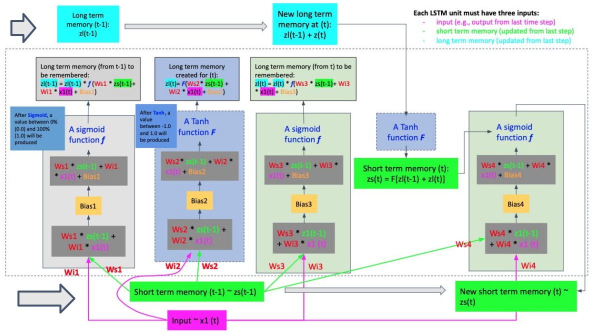
Initial short term memory: 0



long term memory (t0): 2.0

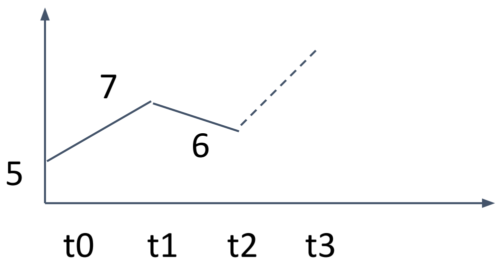
short term memory (t0):3.0

Input (t0): 5

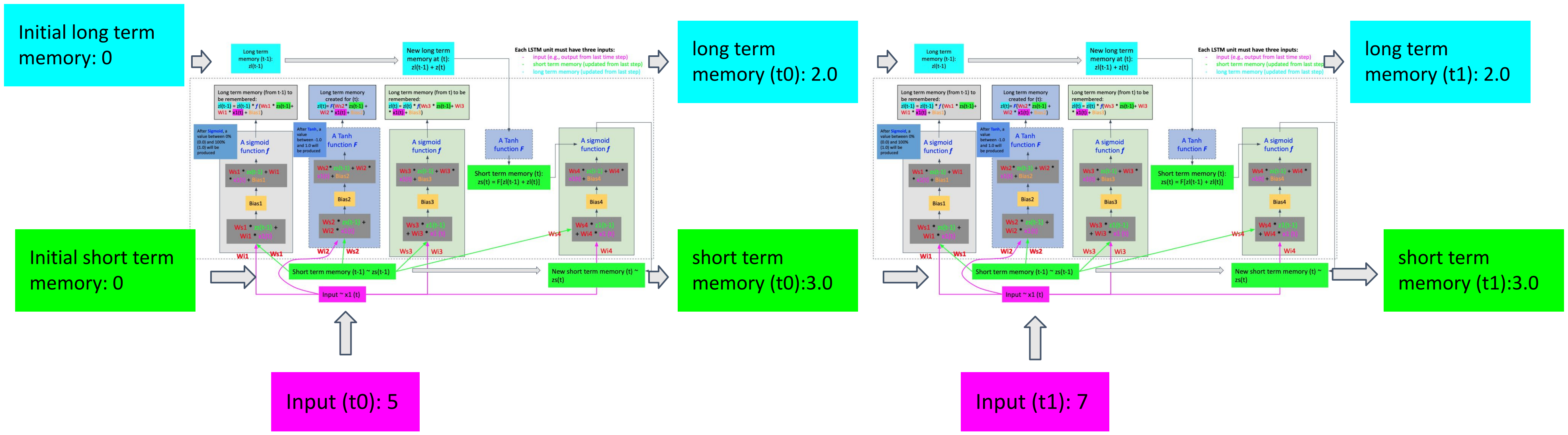


Input (t1): 7

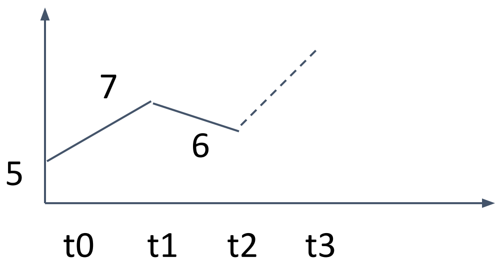
How it works in a real data time series



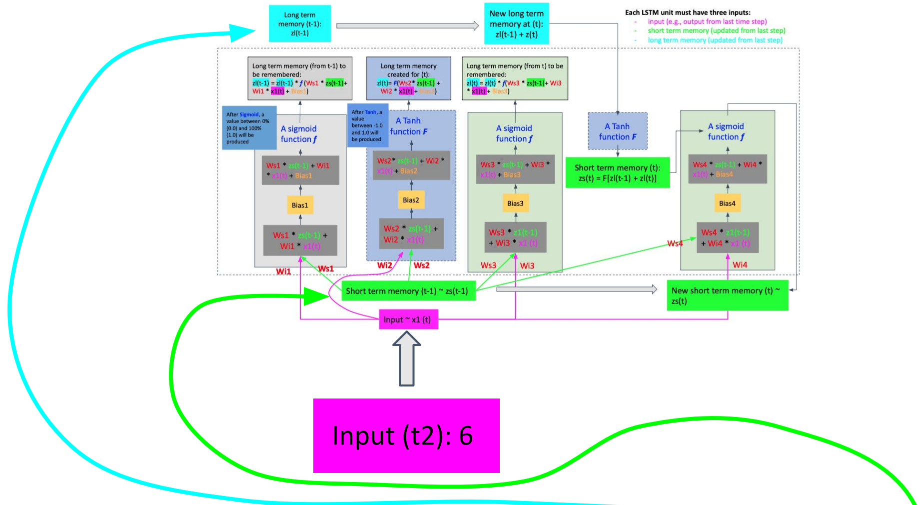
Assuming we have a time series from t0 to t2, and we would like to predict the value at t3

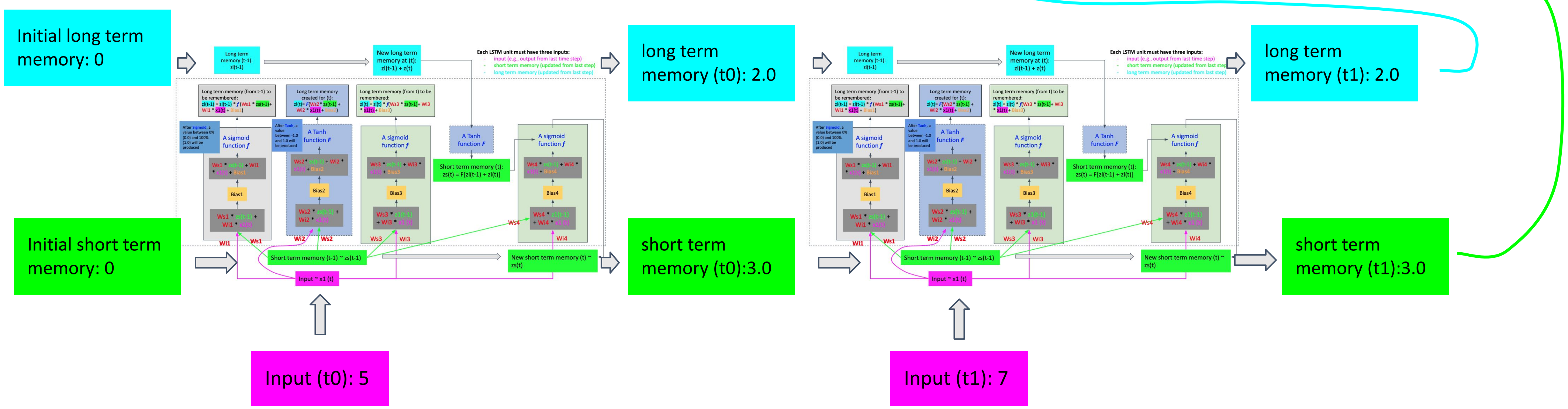


How it works in a real data time series

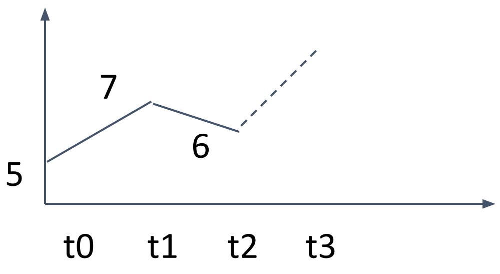


Assuming we have a time series from t0 to t2, and we would like to predict the value at t3

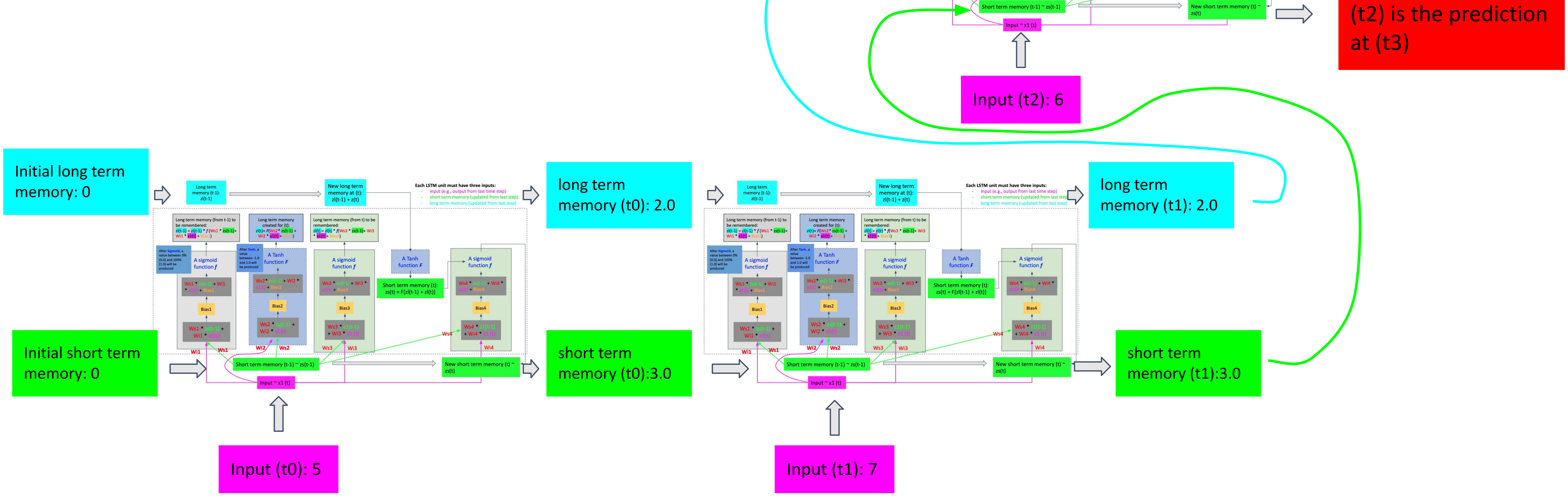




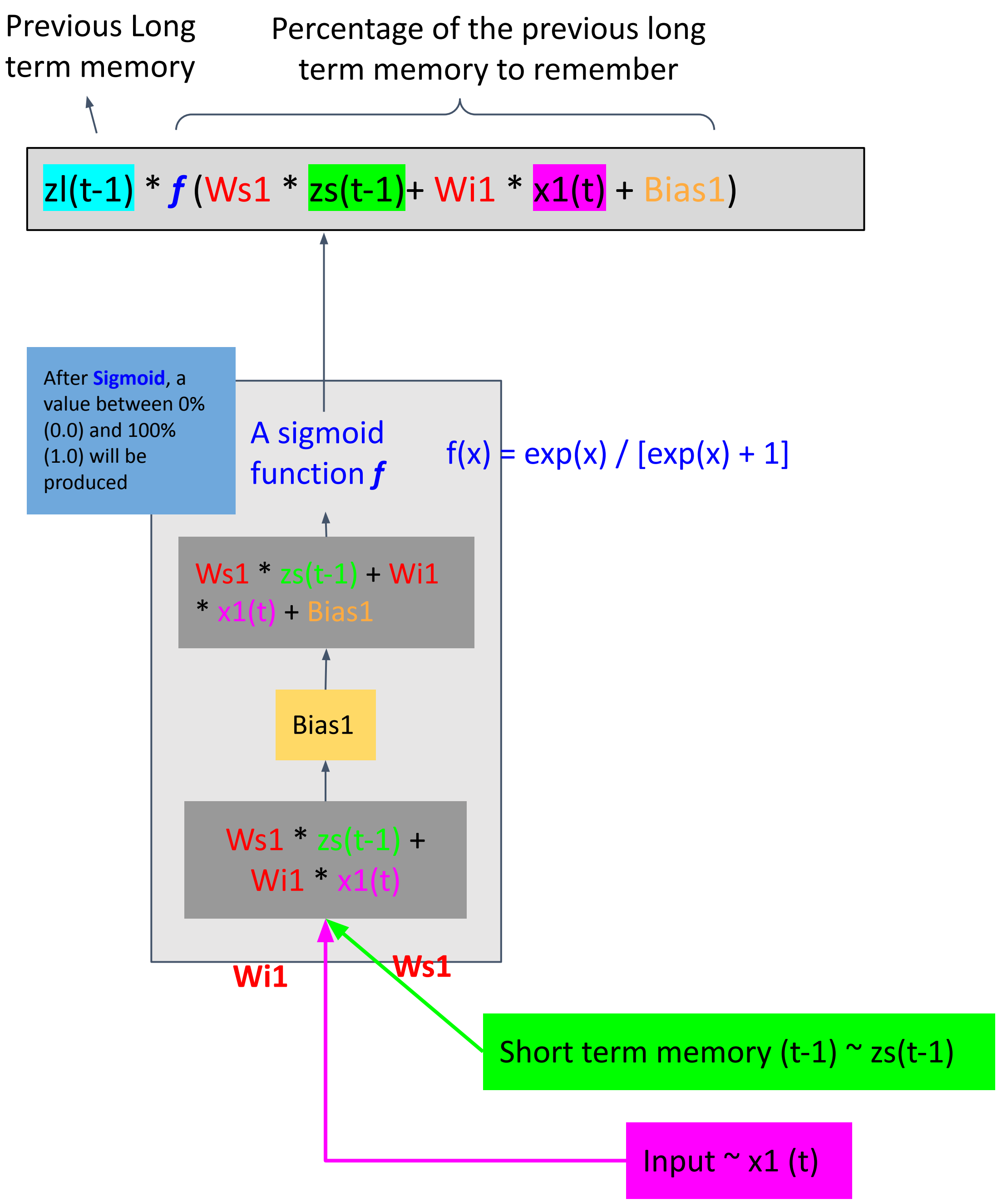
How it works in a real data time series



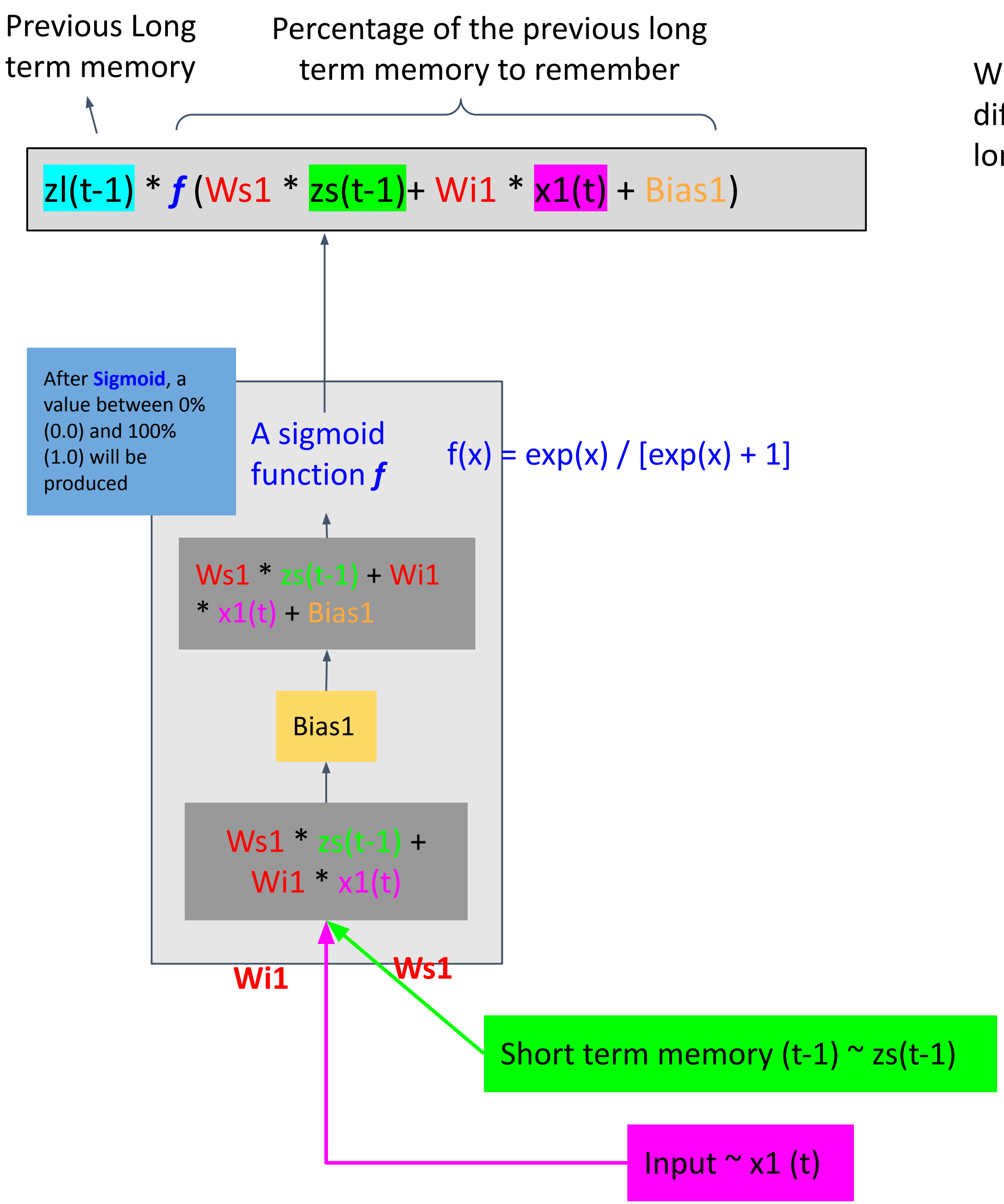
Assuming we have a time series from t0 to t2, and we would like to predict the value at t3



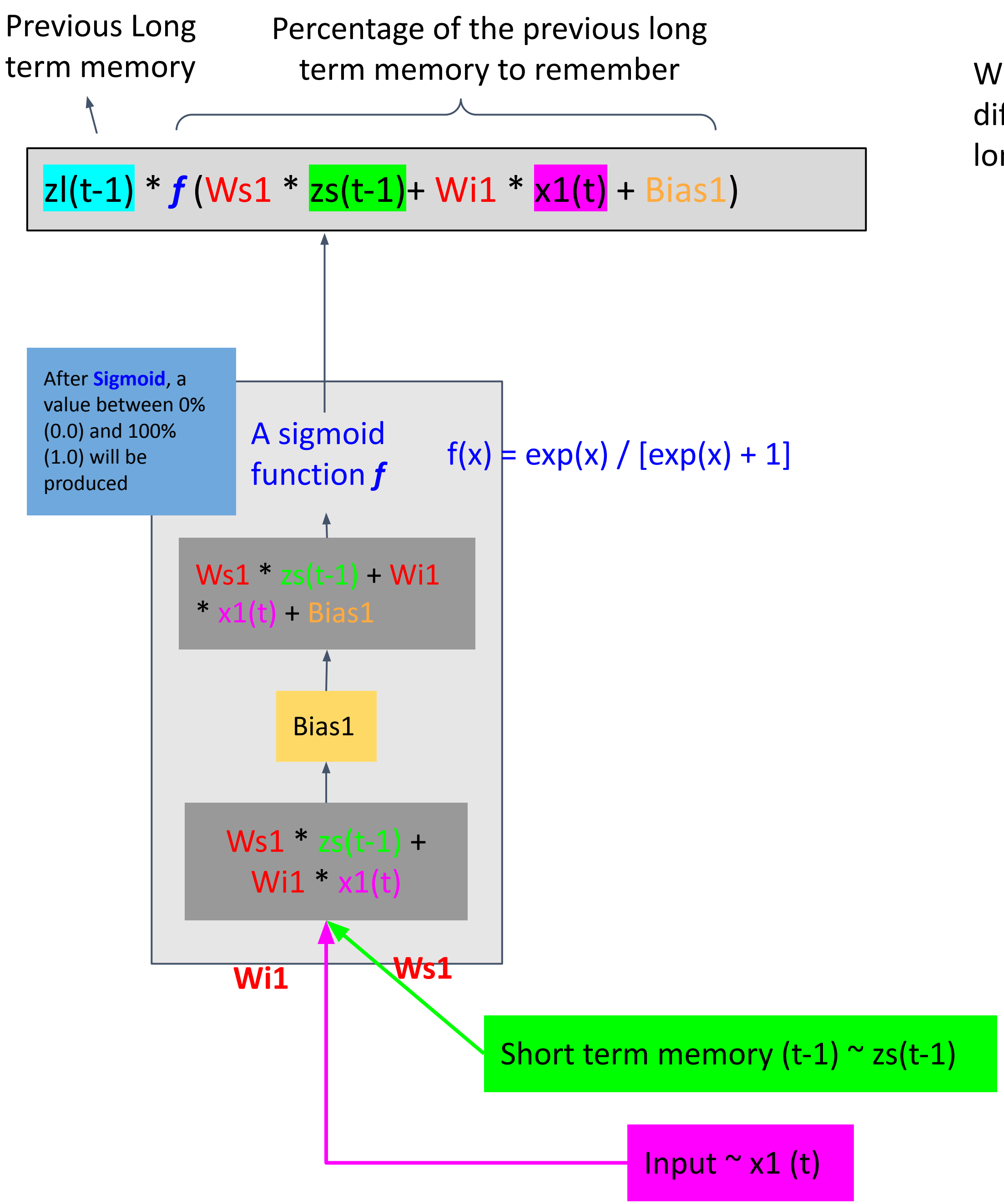
How “forget gate” remember/forget things



Previous Long term memory
Percentage of the previous long term memory to remember
zl(t-1) * f (Ws1 * zs(t-1)+ Wi1 * x1(t) + Bias1)
After Sigmoid, a value between 0% (0.0) and 100% (1.0) will be produced
A sigmoid function f
f(x) = exp(x) / [exp(x) + 1]
Ws1 * zs(t-1) + Wi1 * x1(t) + Bias1
Bias1
Ws1 * zs(t-1) + Wi1 * x1(t)
Wi1
Ws1
Short term memory (t-1) ~ zs(t-1)
Input ~ x1 (t)



When the previous time step output (represented by the short term memory (t-1)) is very different to the input (represented by x1(t)), the forget gate tends to forget the previous long term memory **more**



When the previous time step output (represented by the short term memory (t-1)) is very different to the input (represented by x1(t)), the forget gate tends to forget the previous long term memory **more**

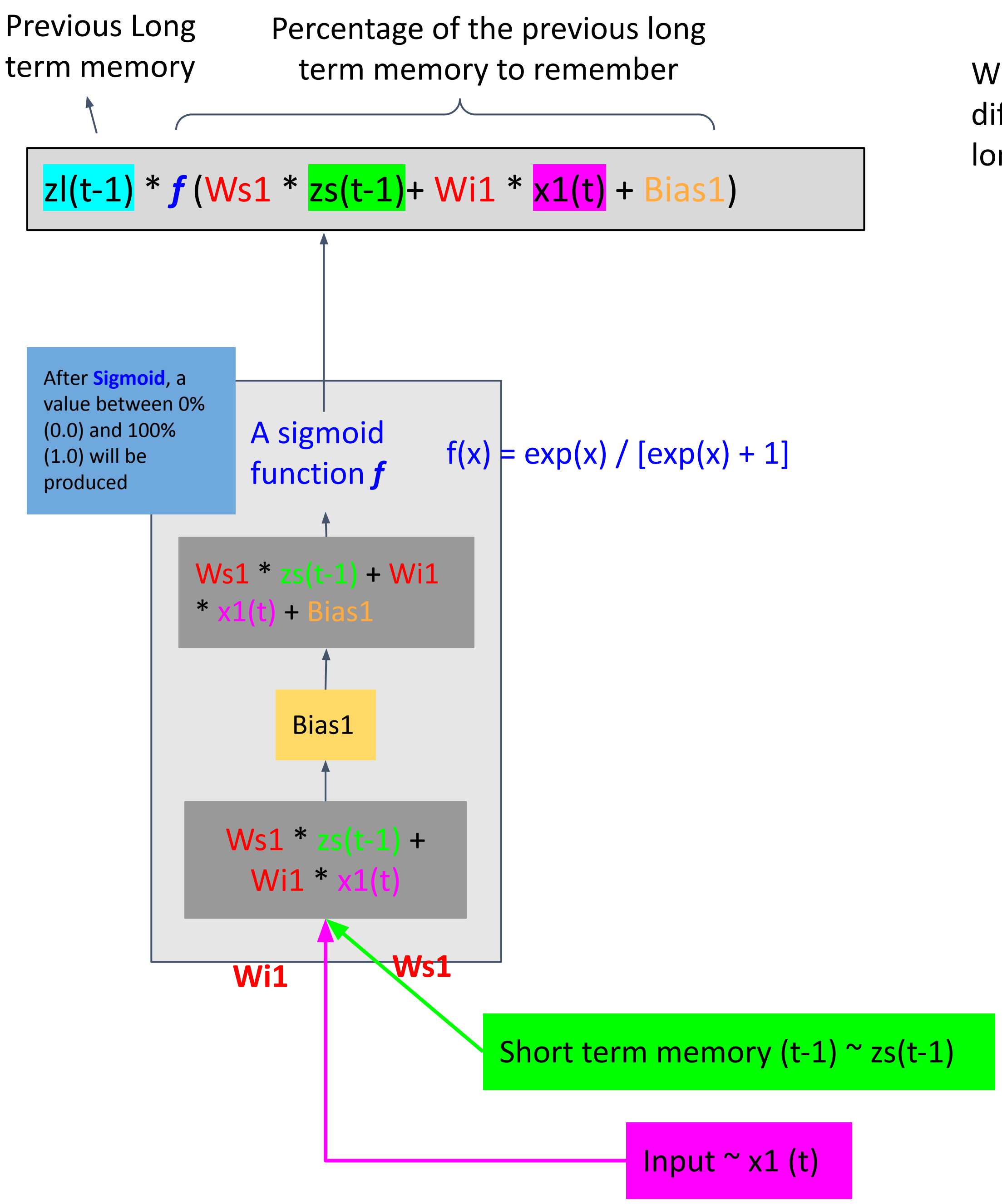
Input and short term memory are **positively** related

Assuming that:

- The short term memory (t-1) = 1.0
- Input (t) = 1.0
- wi1 = 1.0, ws1 = 2.0, and bias = 0.0

Before the Sigmoid function, we can have the combined value as:
1.0 * 1.0 + 2.0 * 1.0 + 0.0 = 3.0

After the Sigmoid function, we have the percentage as:
f (21.0) = 0.95



When the previous time step output (represented by the short term memory (t-1)) is very different to the input (represented by x1(t)), the forget gate tends to forget the previous long term memory **more**

Input and short term memory are **positively** related

Assuming that:

- The short term memory (t-1) = 1.0
- Input (t) = 1.0
- wi1 = 1.0, ws1 = 2.0, and bias = 0.0

Before the Sigmoid function, we can have the combined value as:
1.0 * 1.0 + 2.0 * 1.0 + 0.0 = 3.0

After the Sigmoid function, we have the percentage as:
f (21.0) = 0.95

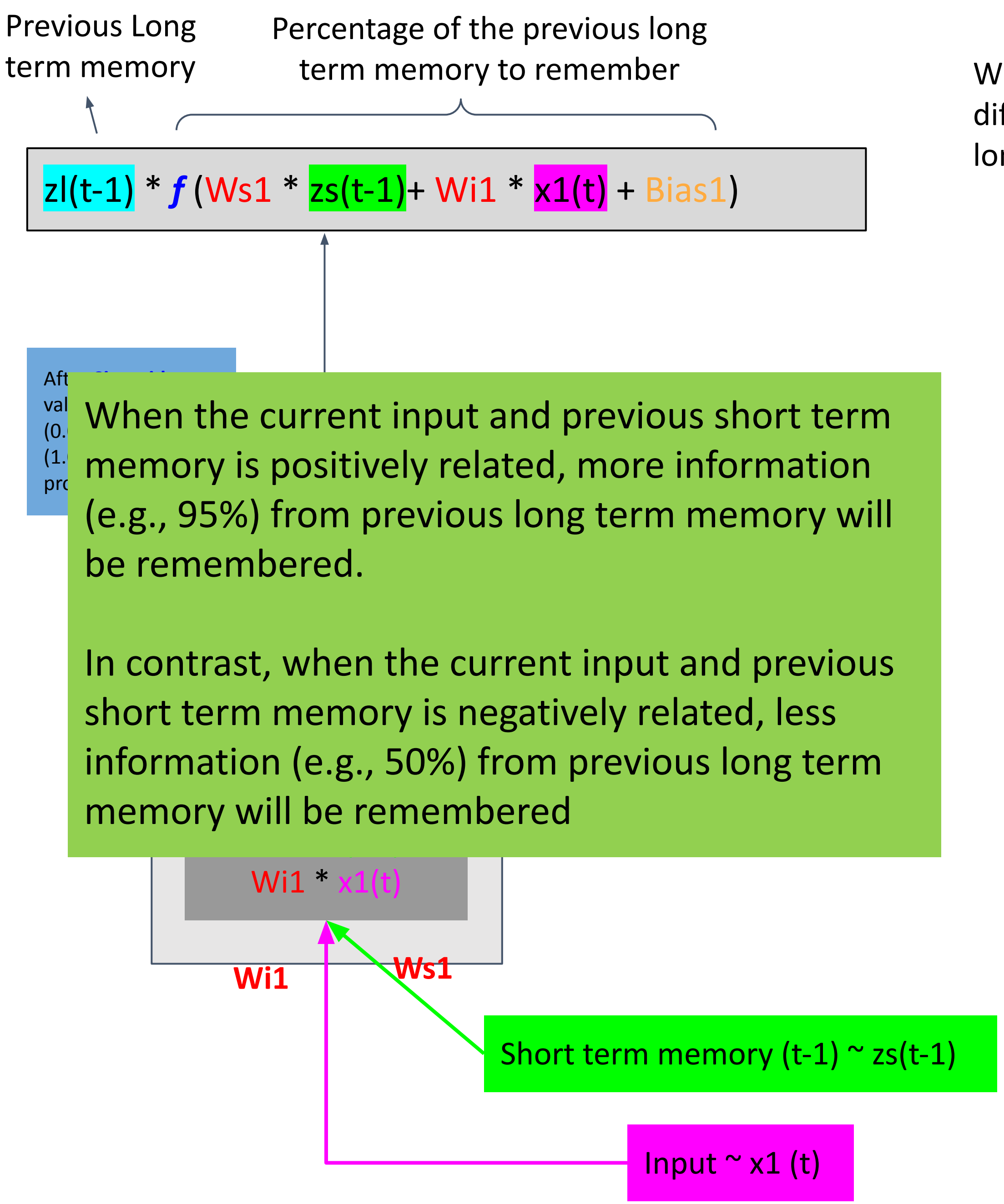
Input and short term memory are **negatively** related

Assuming that:

- The short term memory (t-1) = 1.0
- Input (t) = -1.0
- wi1 = 1.0, ws1 = 2.0, and bias = 0.0

Before the Sigmoid function, we can have the combined value as:
1.0 * 1.0 + 2.0 * -1.0 + 0.0 = 0.0

After the Sigmoid function, we have the percentage as:
f (0.0) = 0.5



When the current input and previous short term memory is positively related, more information (e.g., 95%) from previous long term memory will be remembered.

In contrast, when the current input and previous short term memory is negatively related, less information (e.g., 50%) from previous long term memory will be remembered

When the previous time step output (represented by the short term memory (t-1)) is very different to the input (represented by x1(t)), the forget gate tends to forget the previous long term memory ***more***

Input and short term memory are ***positively*** related

Assuming that:

- The short term memory (t-1) = 1.0
- Input (t) = 1.0
- wi1 = 1.0, ws1 = 2.0, and bias = 0.0

Before the Sigmoid function, we can have the combined value as:
1.0 * 1.0 + 2.0 * 1.0 + 0.0 = 3.0

After the Sigmoid function, we have the percentage as:
f (21.0) = 0.95

Input and short term memory are ***negatively*** related

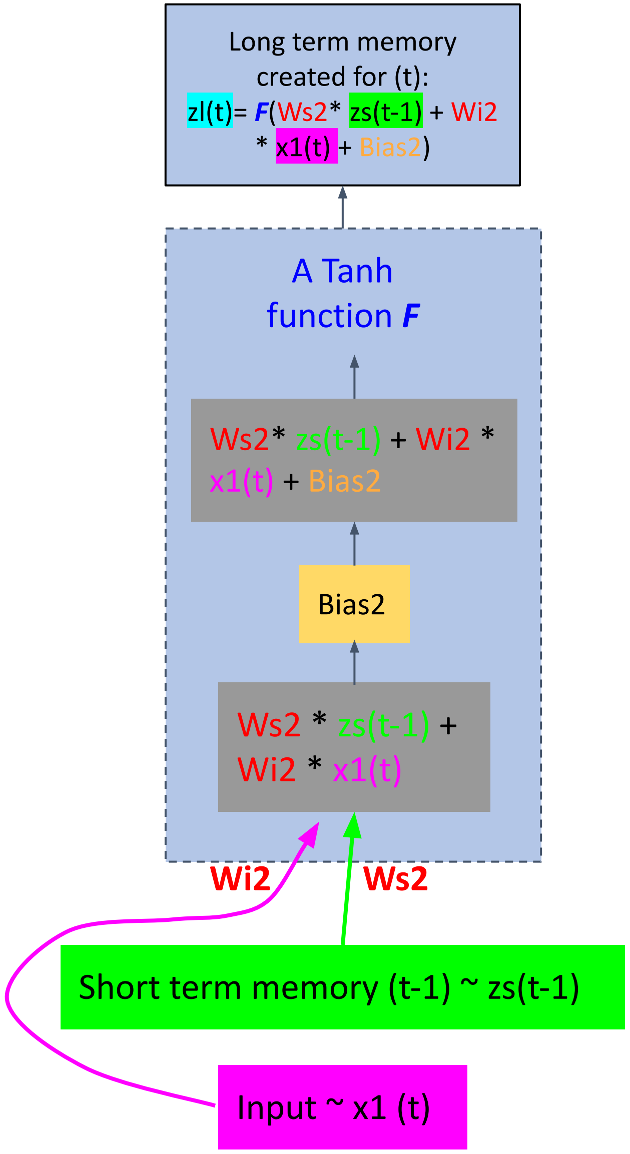
Assuming that:

- The short term memory (t-1) = 1.0
- Input (t) = -1.0
- wi1 = 1.0, ws1 = 2.0, and bias = 0.0

Before the Sigmoid function, we can have the combined value as:
1.0 * 1.0 + 2.0 * -1.0 + 0.0 = 0.0

After the Sigmoid function, we have the percentage as:
f (0.0) = 0.5

How “input gate” create long term memory

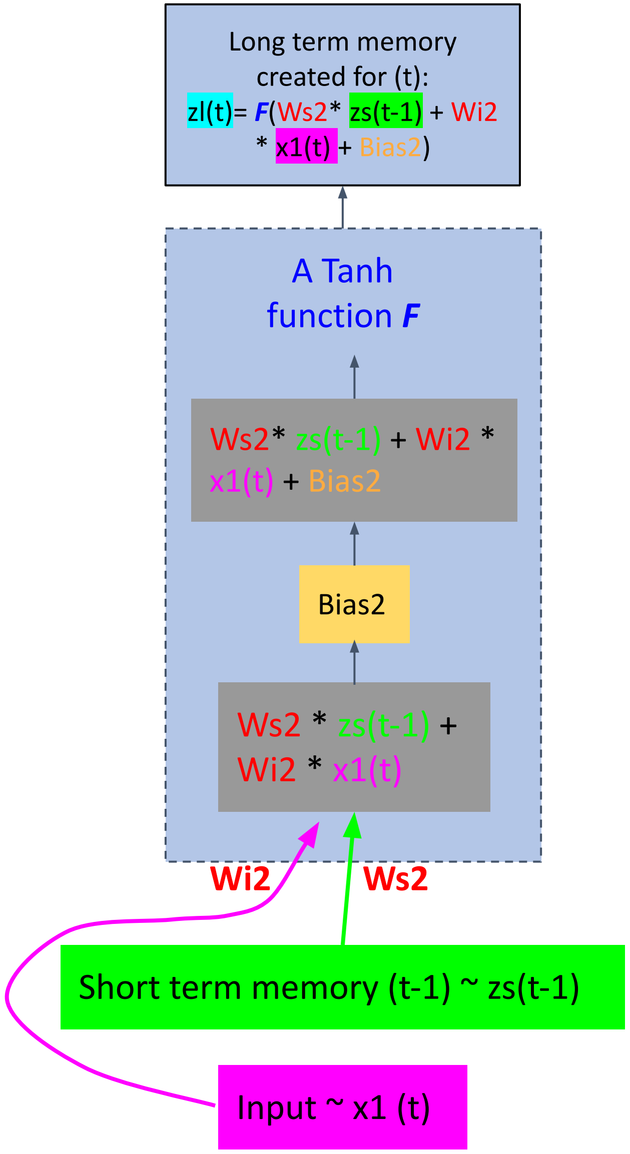


When both the previous time step output (represented by the short term memory (t-1)) and input (represented by x1(t)) are positive, the output from Tanh will be a positive value

Assuming that:

- The short term memory (t-1) = 1.0
- Input (t) = 1.0
- wi1 = 1.0, ws1 = 2.0, and bias = 0.0

zl(t) = F (1.0 * 1.0 + 2.0 * 1.0 + 0.0) = 0.995



When both the previous time step output (represented by the short term memory (t-1)) and input (represented by x1(t)) are positive, the output from Tanh will be a positive value

Assuming that:

- The short term memory (t-1) = 1.0
- Input (t) = 1.0
- wi1 = 1.0, ws1 = 2.0, and bias = 0.0

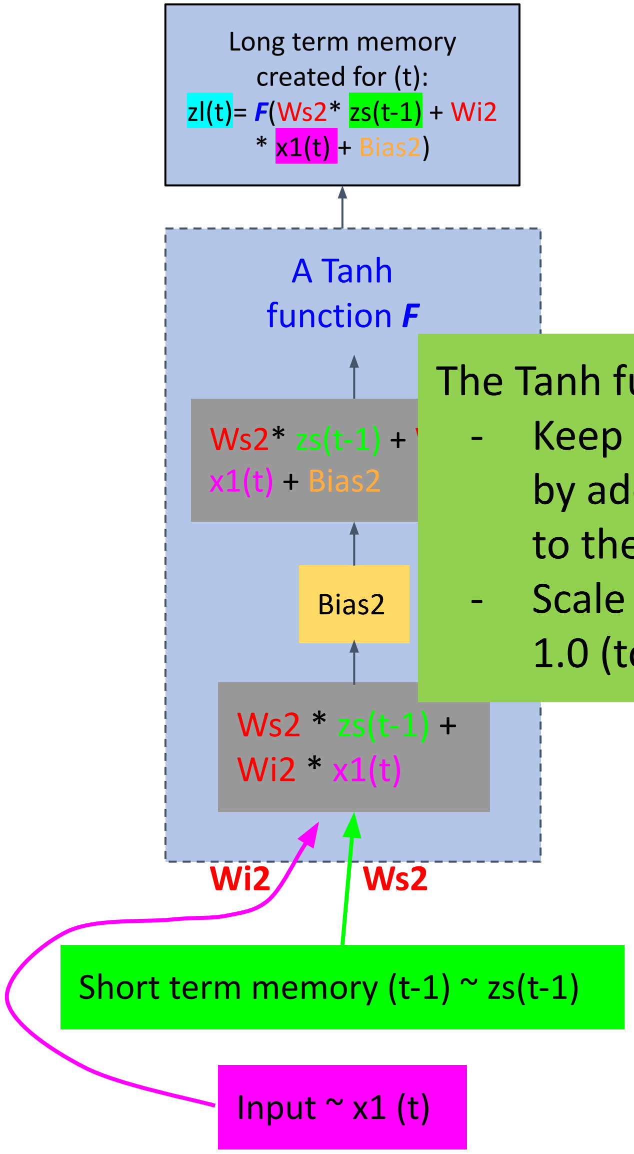
zl(t) = F (1.0 * 1.0 + 2.0 * 1.0 + 0.0) = 0.995

When both the previous time step output (represented by the short term memory (t-1)) and input (represented by x1(t)) are negative, the output from Tanh will be a negative value

Assuming that:

- The short term memory (t-1) = -1.0
- Input (t) = -1.0
- wi1 = 1.0, ws1 = 2.0, and bias = 0.0

zl(t) = F (1.0 * -1.0+ 2.0 * -1.0 + 0.0) = -0.995



When both the previous time step output (represented by the short term memory (t-1)) and input (represented by x1(t)) are positive, the output from Tanh will be a positive value

Assuming that:

- The short term memory (t-1) = 1.0
- Input (t) = 1.0
- wi1 = 1.0, ws1 = 2.0, and bias = 0.0

zl(t) = F (1.0 * 1.0 + 2.0 * 1.0 + 0.0) = 0.995

When both the previous time step output

The Tanh function will:

- Keep the sign (positive/negative) obtained by adding the previous short-term memory to the current input
- Scale the added up value between -1.0 and 1.0 (to make the computing more efficient)

Assuming that:

- The short term memory (t-1) = -1.0
- Input (t) = -1.0
- wi1 = 1.0, ws1 = 2.0, and bias = 0.0

zl(t) = F (1.0 * -1.0+ 2.0 * -1.0 + 0.0) = -0.995

(t-1)) and input (represented by x1(t)) have opposite sign, the output from Tanh will be sth in between, and may be around zero (in this case, input and previous long term memory cancel each other).

Assuming that:

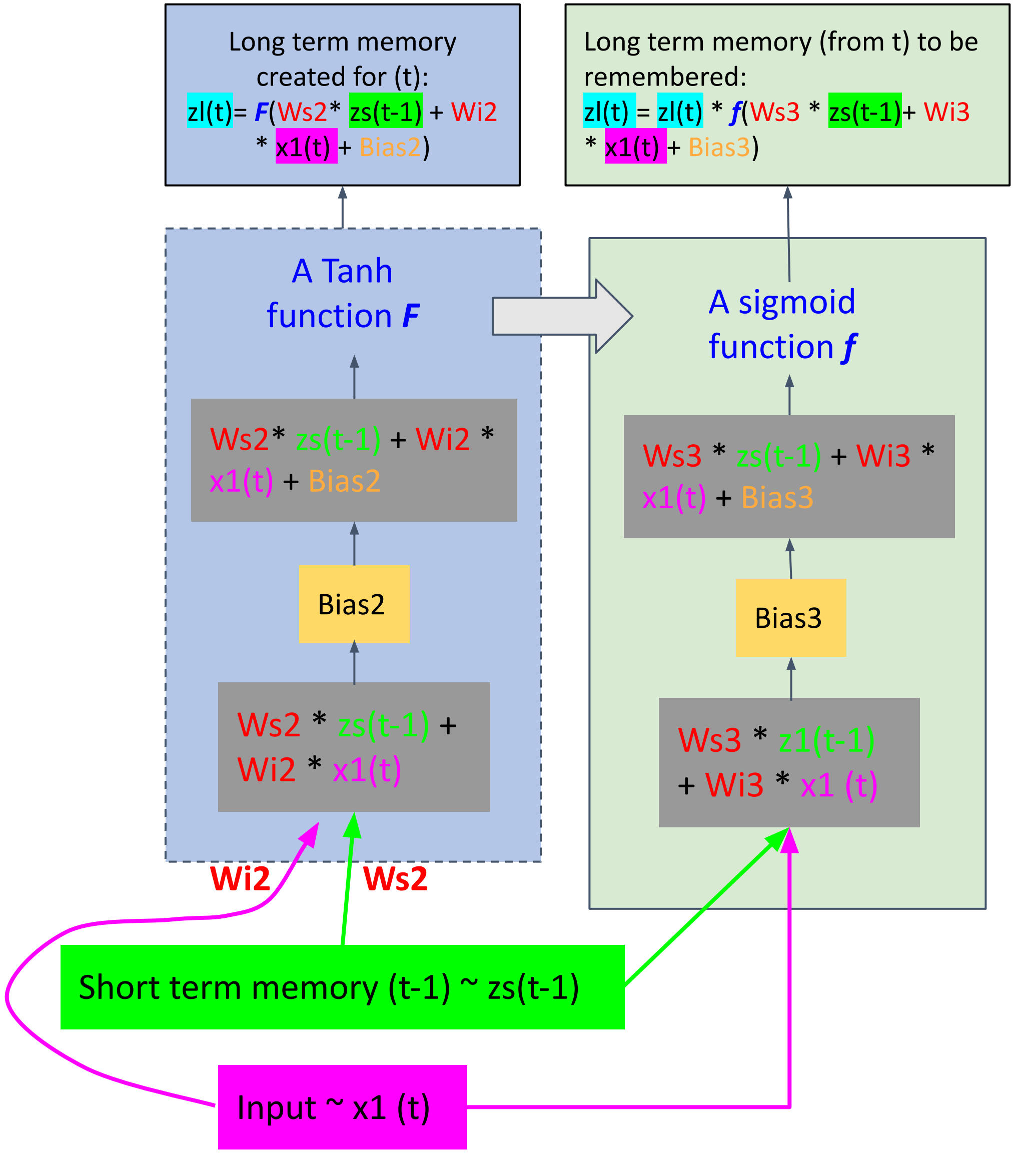
- The short term memory (t-1) = 1.0
- Input (t) = -1.0
- wi1 = 1.0, ws1 = 2.0, and bias = 0.0

zl(t) = F (1.0 * 1.0+ 2.0 * -1.0 + 0.0) = -0.76

Assuming that:

- The short term memory (t-1) = 1.0
- Input (t) = -1.0
- wi1 = 1.0, ws1 = 1.0, and bias = 0.0

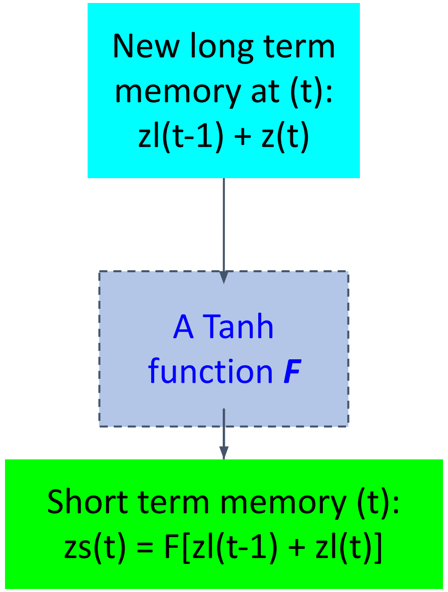
zl(t) = F (1.0 * 1.0+ 10 * -1.0 + 0.0) = 0.0



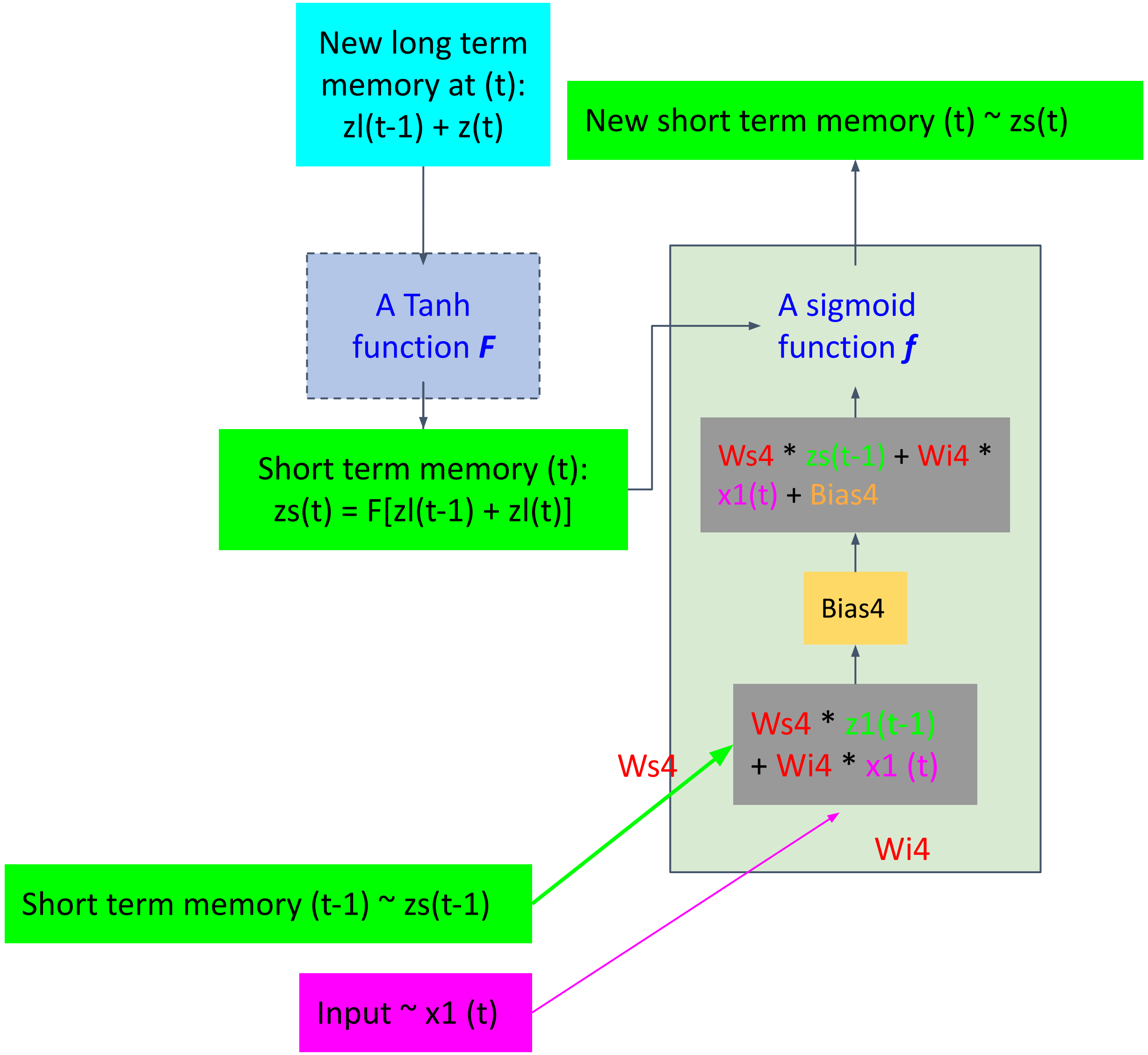
After we created the long term memory for the current time step (with the function Tanh), we will decide how much we want to keep it depending on the difference between the current input and the previous short term memory. For example:

The more differences between the current input and previous short term memory, the less information will be kept (see the section “how forget gate” remember/forget things)

How “output gate” create short term memory



After we get the updated long term memory from both previous time step and the current step, we will use it to as the short term memory for the current time step (e.g., after the Tanh scaling like in the "input gate")



After we get the updated long term memory from both previous time step and the current step, we will use it to as the short term memory for the current time step (e.g., after the Tanh scaling like in the "input gate")

After we created the short term memory for the current time step (with the function Tanh), we will decide how much we want to keep it depending on the difference between the current input and the previous short term memory. For example:

The more differences between the current input and previous short term memory, the less information will be kept (just like how the "forget gate" and "input gate" work)