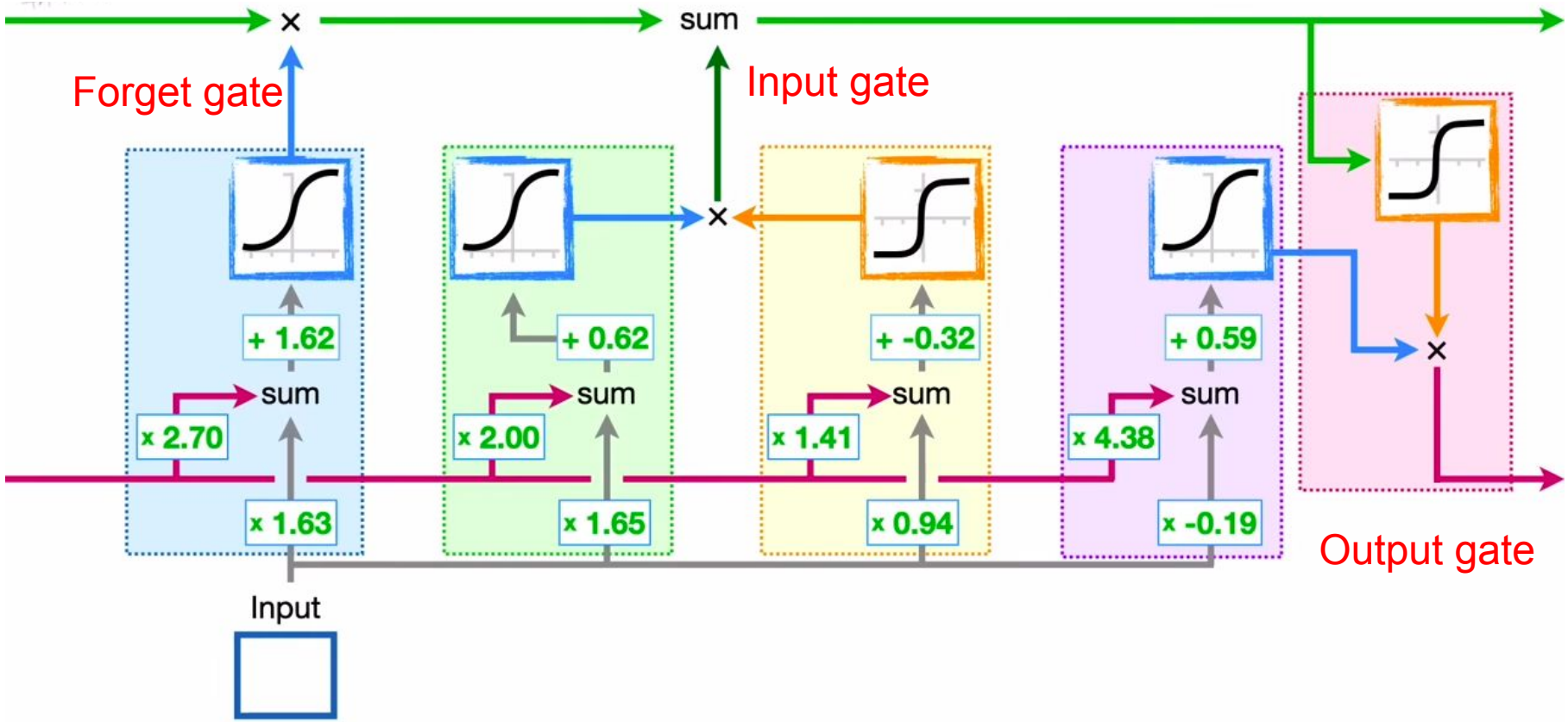
LSTM

Usman Nazir

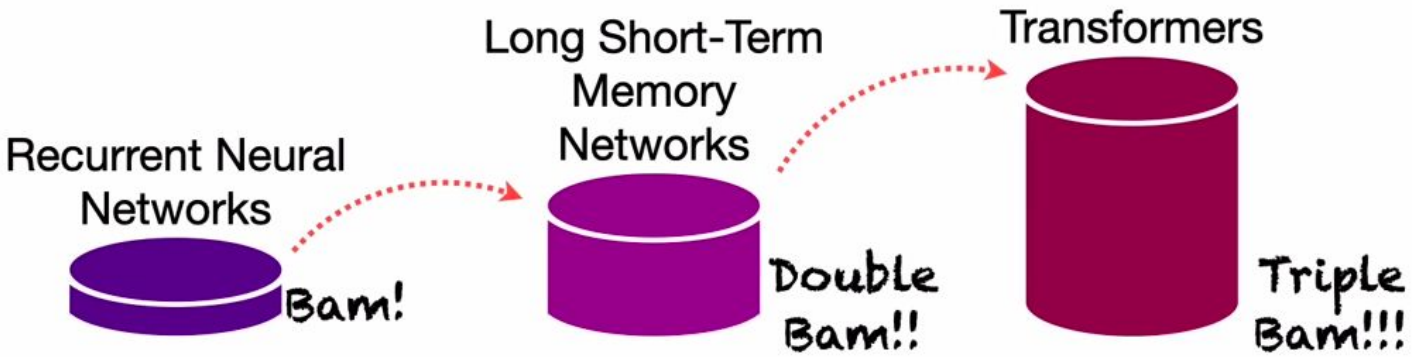

×
sum
Forget gate
Input gate
+ 1.62
+ 0.62
+ -0.32
+ 0.59
sum
sum
sum
sum
× 2.70
× 2.00
× 1.41
× 4.38
× 1.63
× 1.65
× 0.94
× -0.19
Output gate
Input

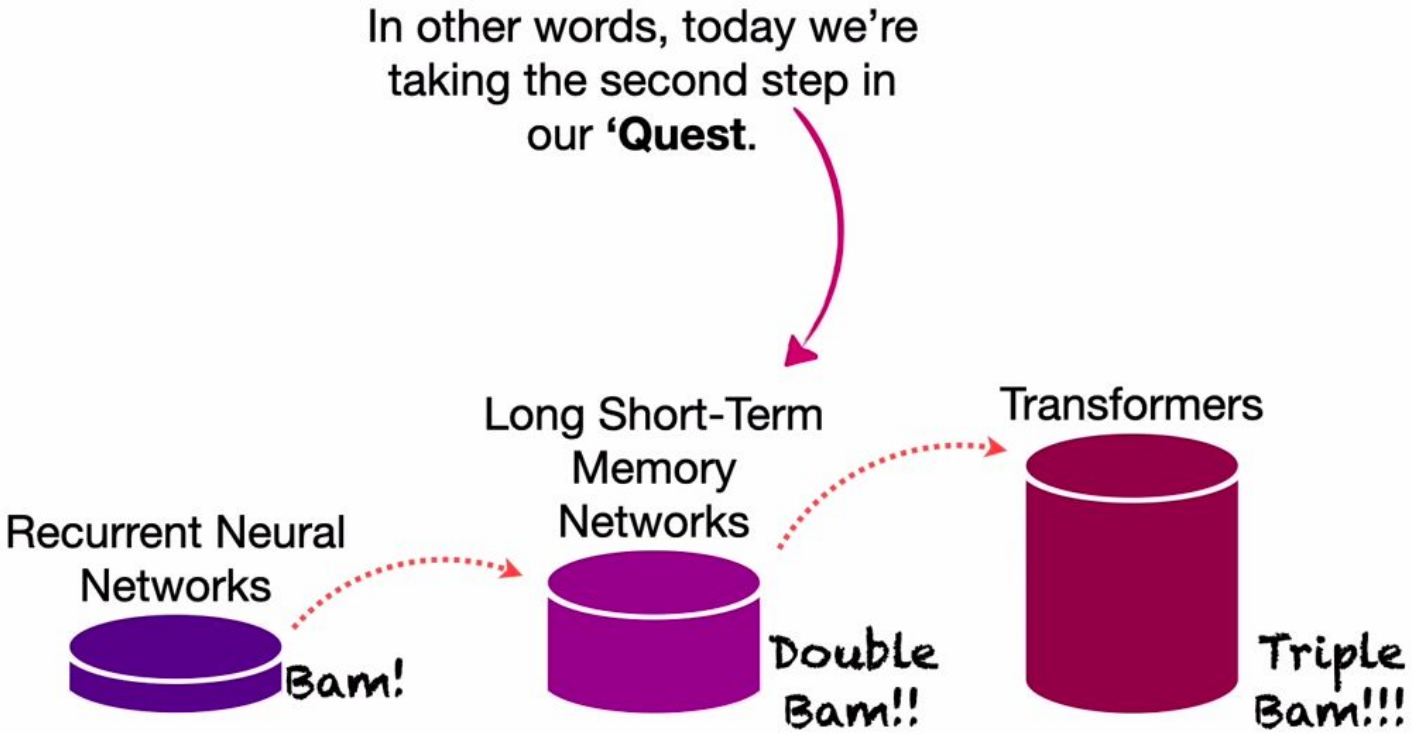
Always

Be

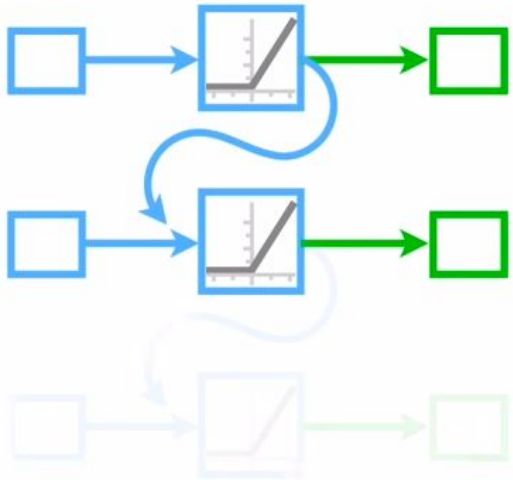
Curious

ALSO NOTE: Although **Long Short-Term Memory** (**LSTM**) is totally awesome, it is also a stepping stone to learning about **Transformers**, which we will talk about in future **StatQuests**.

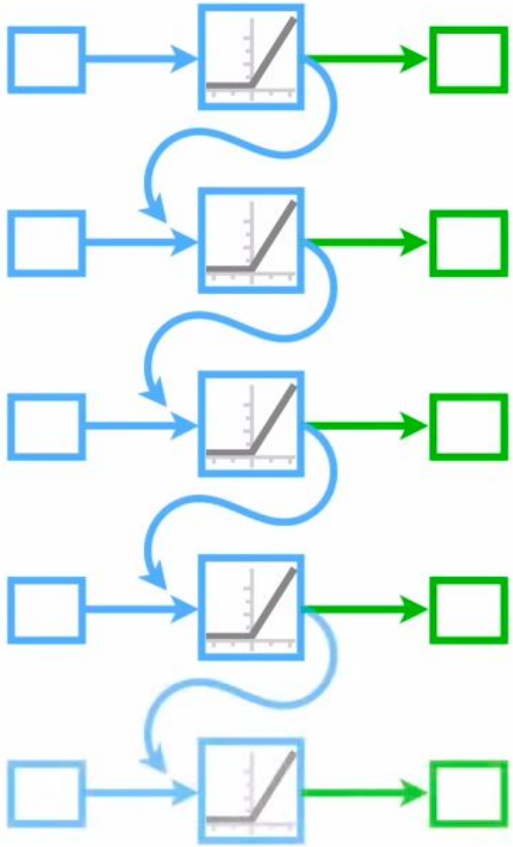



In other words, today we're taking the second step in our 'Quest.
Recurrent Neural Networks
Bam!
Long Short-Term Memory Networks
Double Bam!!
Transformers
Triple Bam!!!

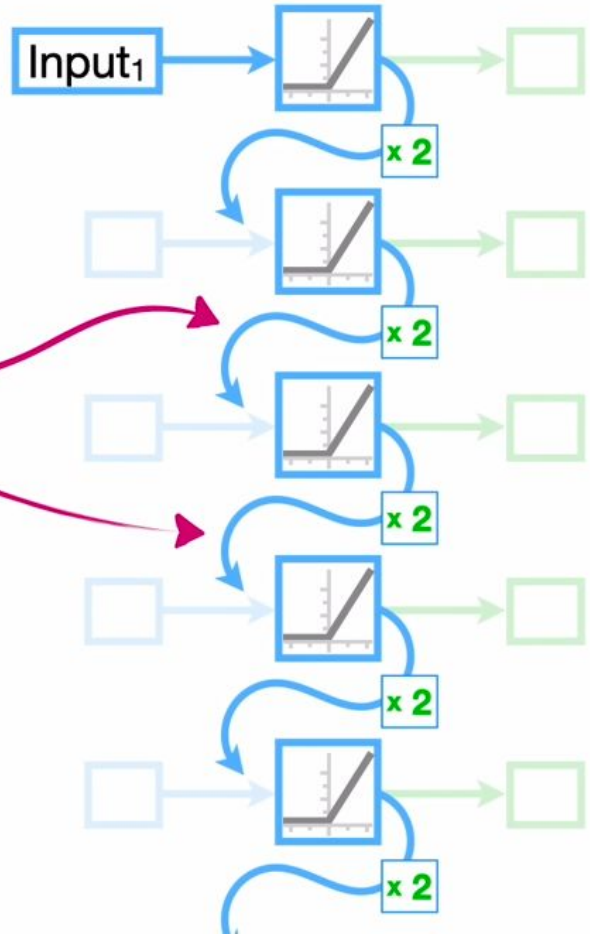
…to ***unroll*** a network that works well with *different amounts* of **sequential data**.



…to ***unroll*** a network that works well with *different amounts* of **sequential data**.

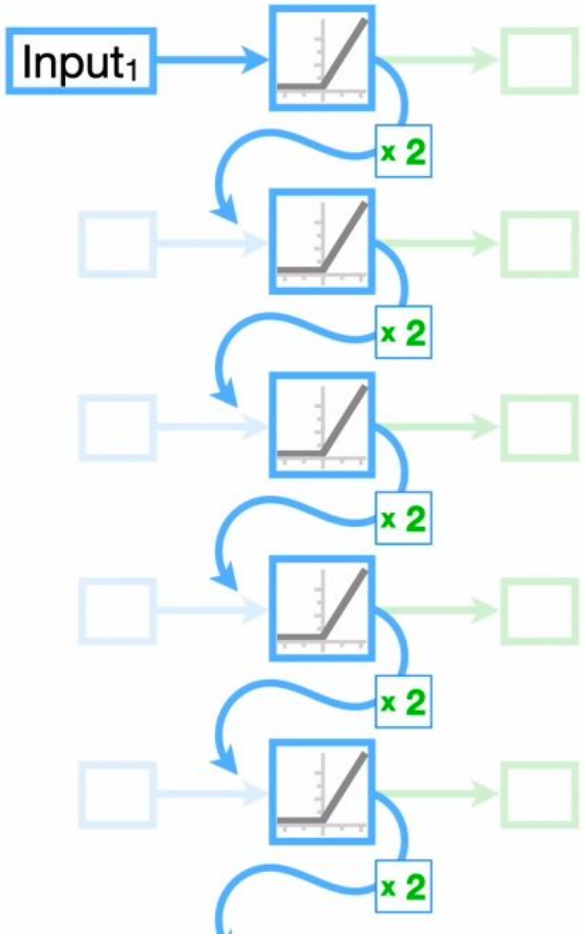


…when the **Weight** on the feedback loop is greater than **1**, and in this example the **Weight** is **2**…



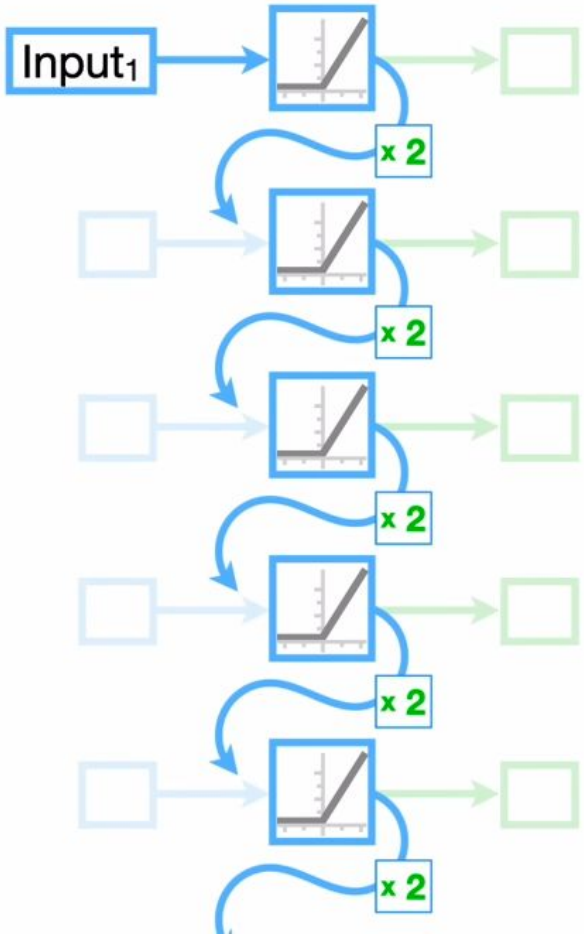
…then, when we do the math, we end up multiplying the **Input** by the **Weight**, which in this case is **2**, raised to the number of times we unrolled the network.

$= \textbf{Input}_1 \times 2^{\text{Num. Unroll}}$



And thus, if we had **50** sequential data points, like **50** days of stock market data, which isn't really that much…

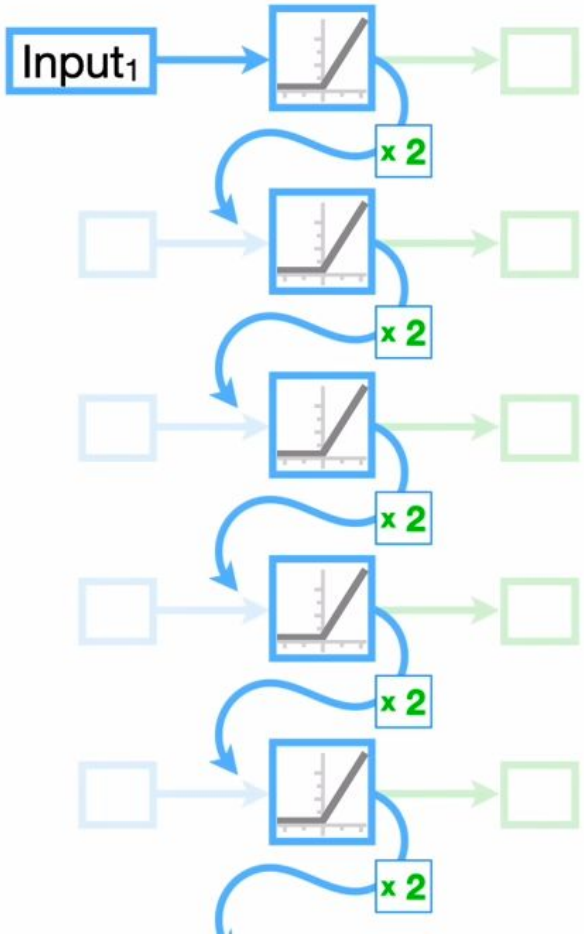
= $\mathbf{Input_1} \times \mathbf{2^{Num.\ Unroll}}$



And thus, if we had **50** sequential data points, like **50** days of stock market data, which isn't really that much…

…then we would raise **2** by **50**…

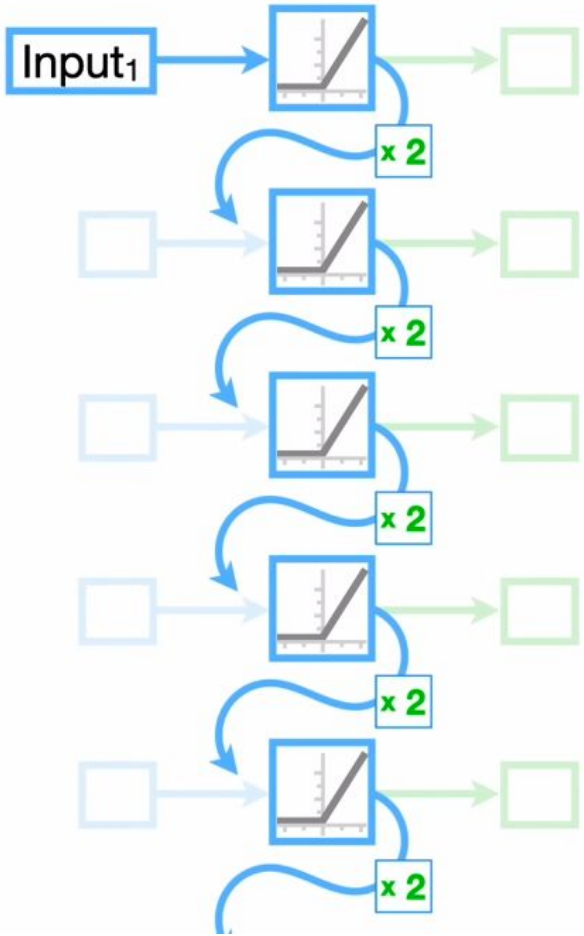
= $\mathbf{Input_1} \times \mathbf{2^{50}}$



…and this **HUGE NUMBER** would cause the gradient, which we need for **Gradient Descent**…

= **Input$_1$** × **2^{50}**

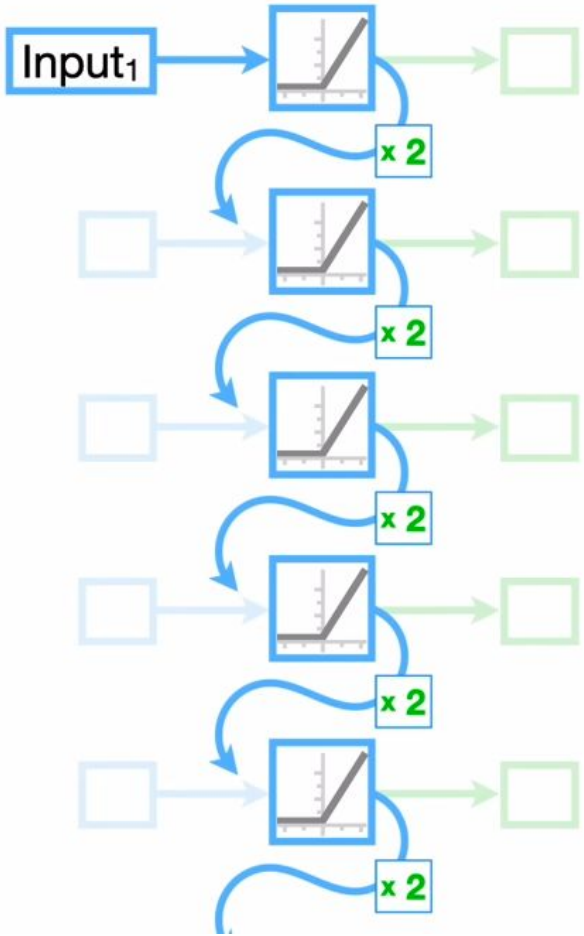
= **Input$_1$** × A HUGE NUMBER



…to **EXPLODE!!!**

= **Input$_1$** × **2^{50}**

= **Input$_1$** × A HUGE NUMBER



Alternatively, we saw that if the **Weight** on the feedback loop was less than **1**, and now we have it set to **0.5**…

= **Input$_1$** × **0.5^{50}**

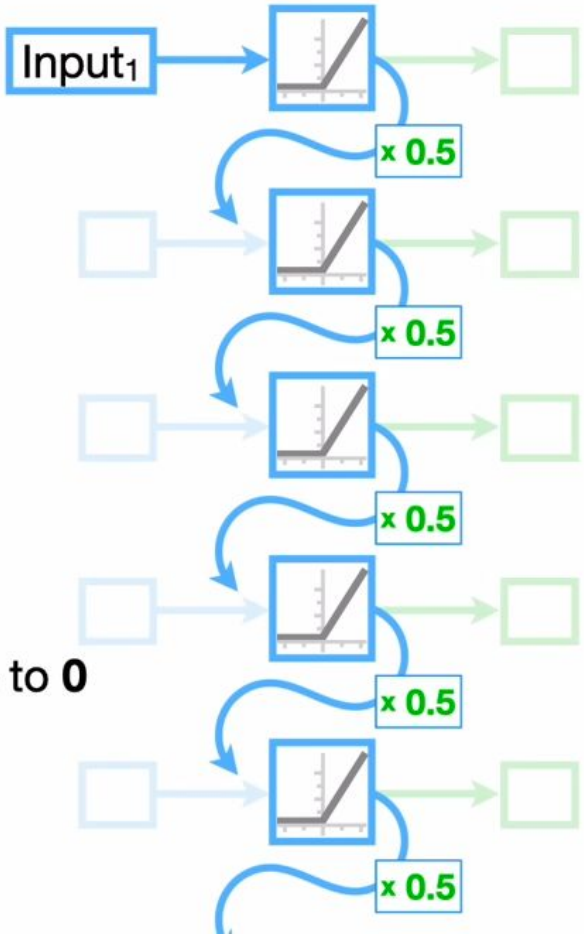
…then we'll end up multiplying the **Input** value by **0.5** raised to the **50th** power…

$= \textbf{Input}_1 \times \textbf{0.5}^{50}$

…and **0.5** raised to the **50th** power is a number super close to **0**…

= **Input**$_1$ × **0.5**^{50}

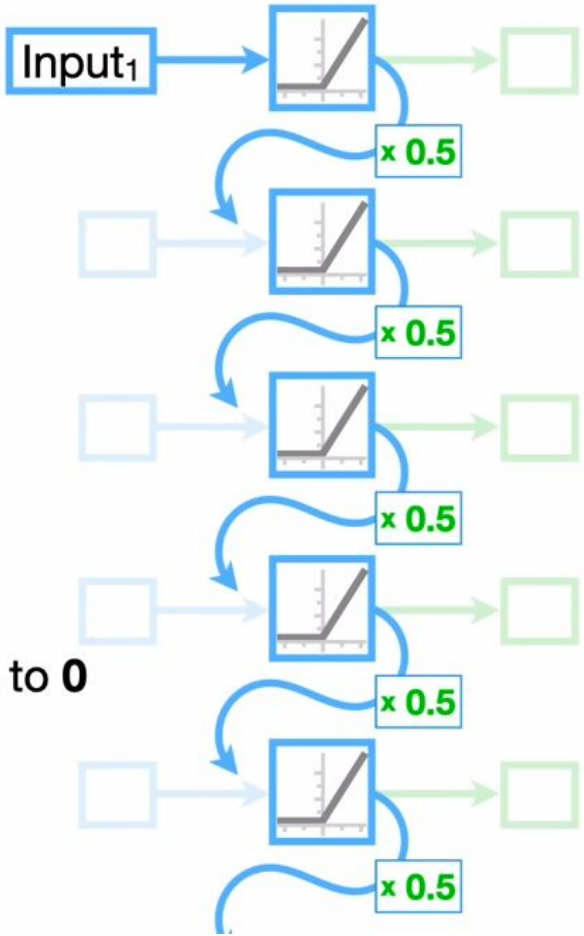
= **Input**$_1$ × a number super close to **0**



…and this number super close to **0** would cause the gradient, which we need for **Gradient Descent**…

= **Input$_1$** × **0.5^{50}**

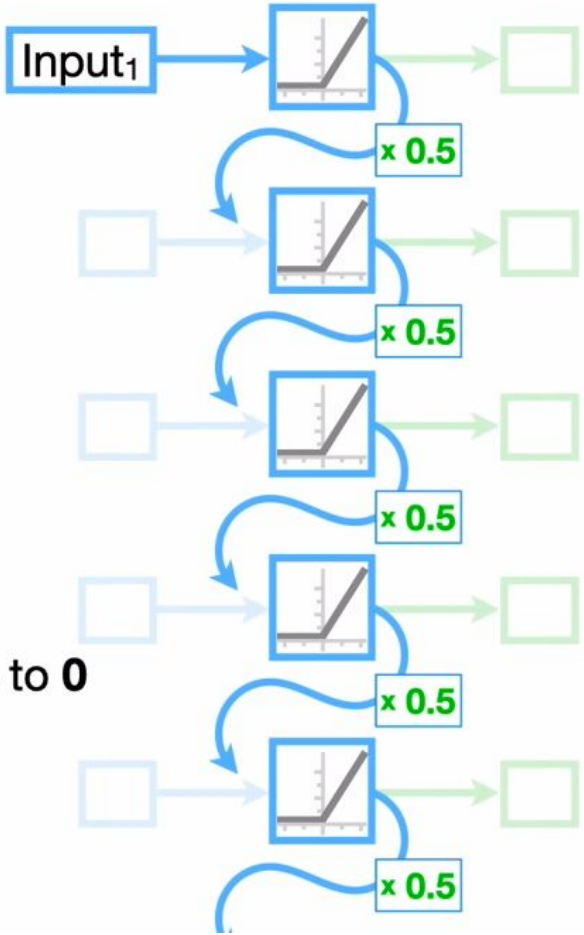
= **Input$_1$** × a number super close to **0**



…to vanish.

= **Input$_1$** × **0.5^{50}**

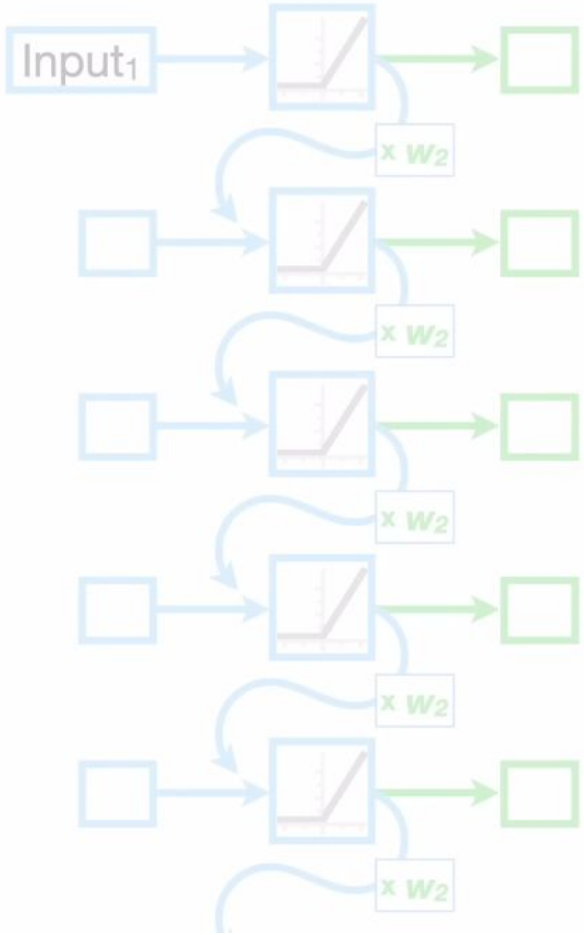
= **Input$_1$** × a number super close to **0**



In summary, basic, vanilla **Recurrent Neural Networks** are hard to train because the gradients can **explode**, or **vanish**.

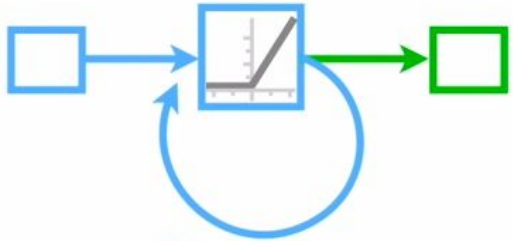
$= \textbf{Input}_1 \times \boldsymbol{w}_2^{\text{Num. Unroll}}$

The good news is that it doesn't take much to extend the basic, vanilla **Recurrent Neural Network** so that we can avoid this problem.

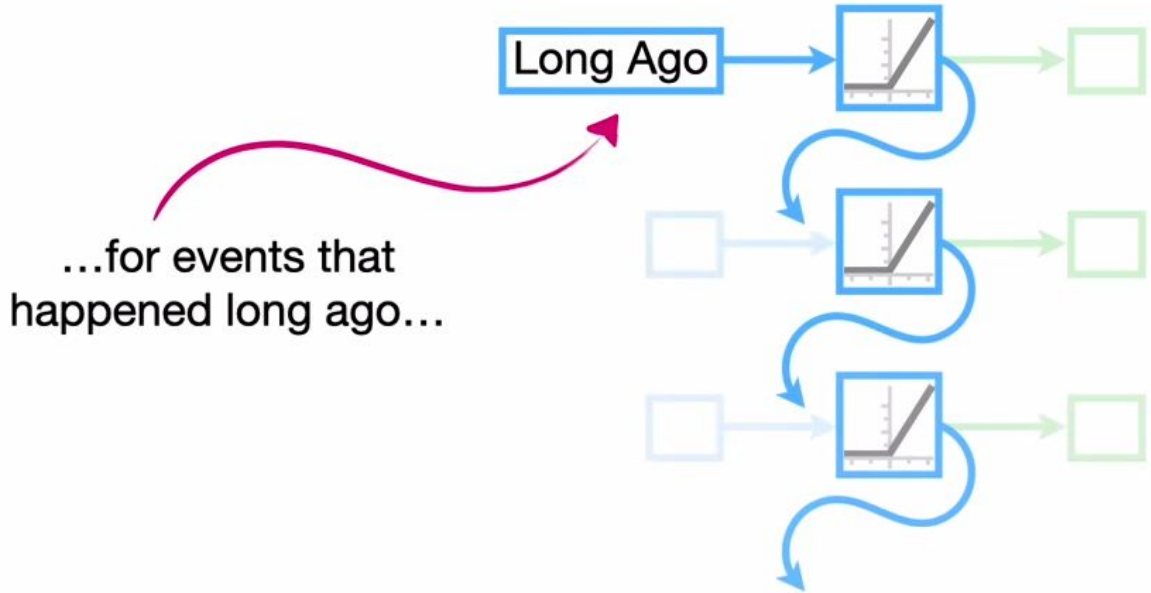


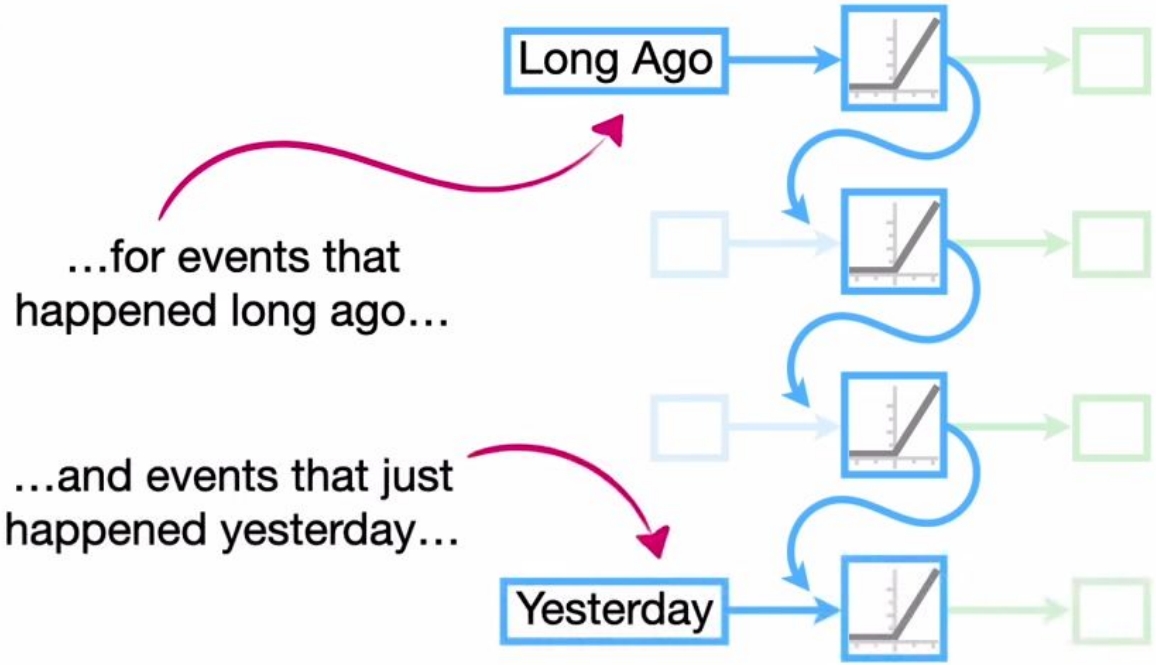
So, today, we're going to talk about **Long Short-Term Memory** (**LSTM**), which is a type of **Recurrent Neural Network** that is designed to avoid the **exploding/ vanishing** gradient problem.

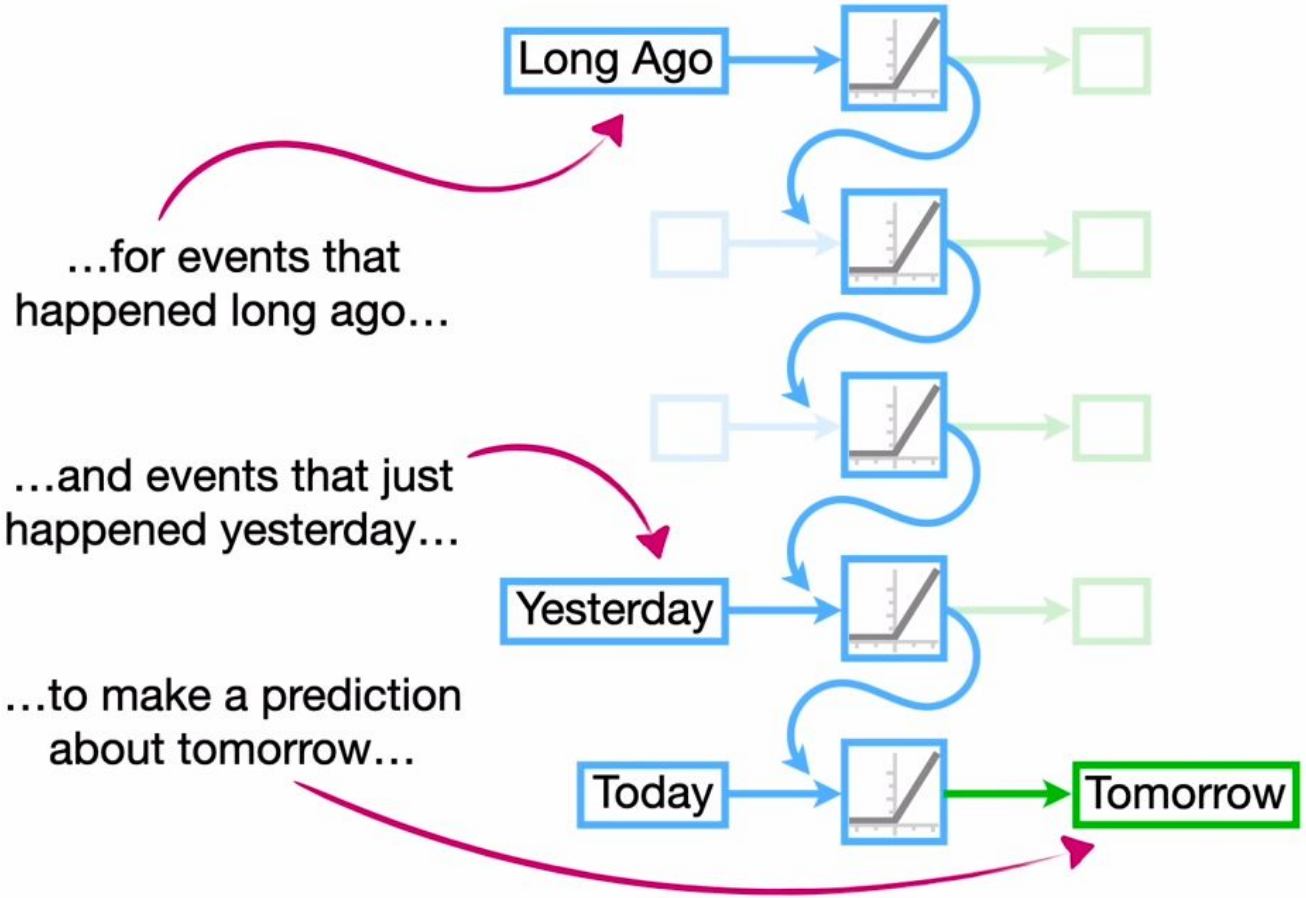
The main idea behind how
Long Short-Term Memory
(**LSTM**) works…



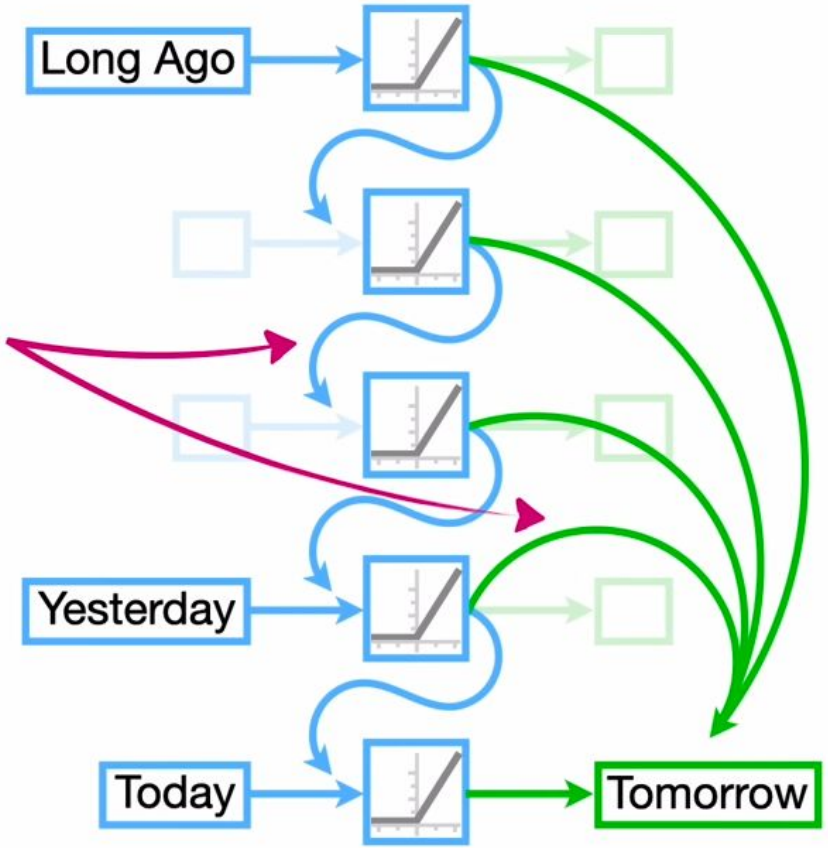
…is that instead of using the same feedback loop connection…


Long Ago
…for events that
happened long ago…

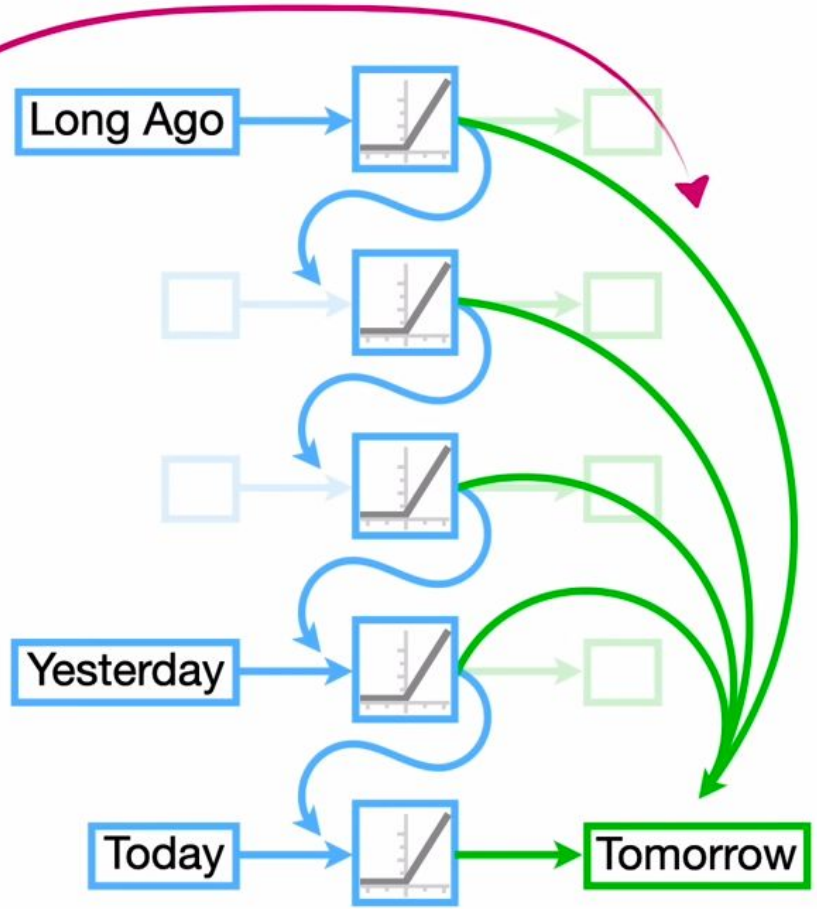

Long Ago
…for events that happened long ago…
…and events that just happened yesterday…
Yesterday


Long Ago
…for events that happened long ago…
…and events that just happened yesterday…
Yesterday
…to make a prediction about tomorrow…
Today
Tomorrow

…**Long Short-Term Memory** uses two separate paths to make predictions about tomorrow.

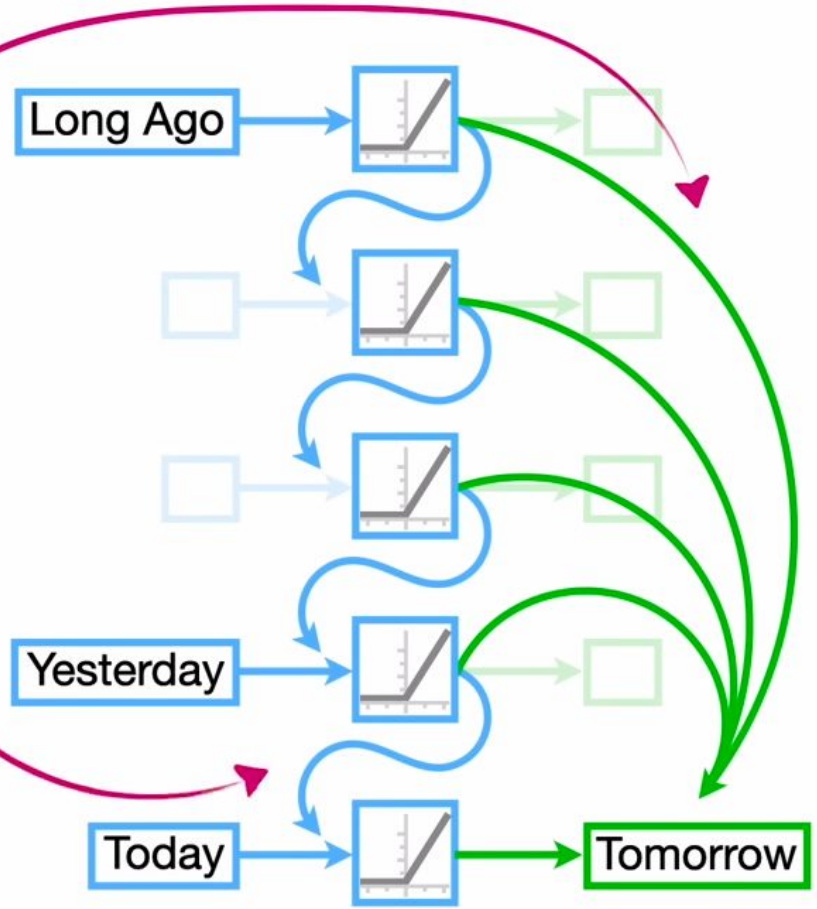


One path is for **Long-Term Memories**…

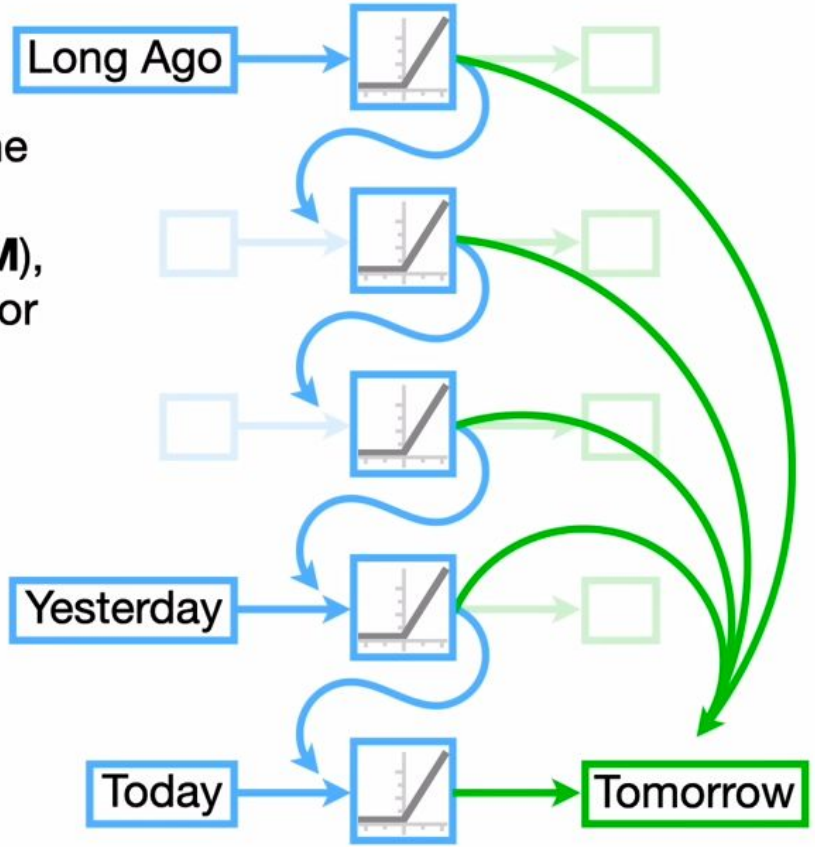


One path is for **Long-Term Memories**…

…and one is for **Short-Term Memories**.

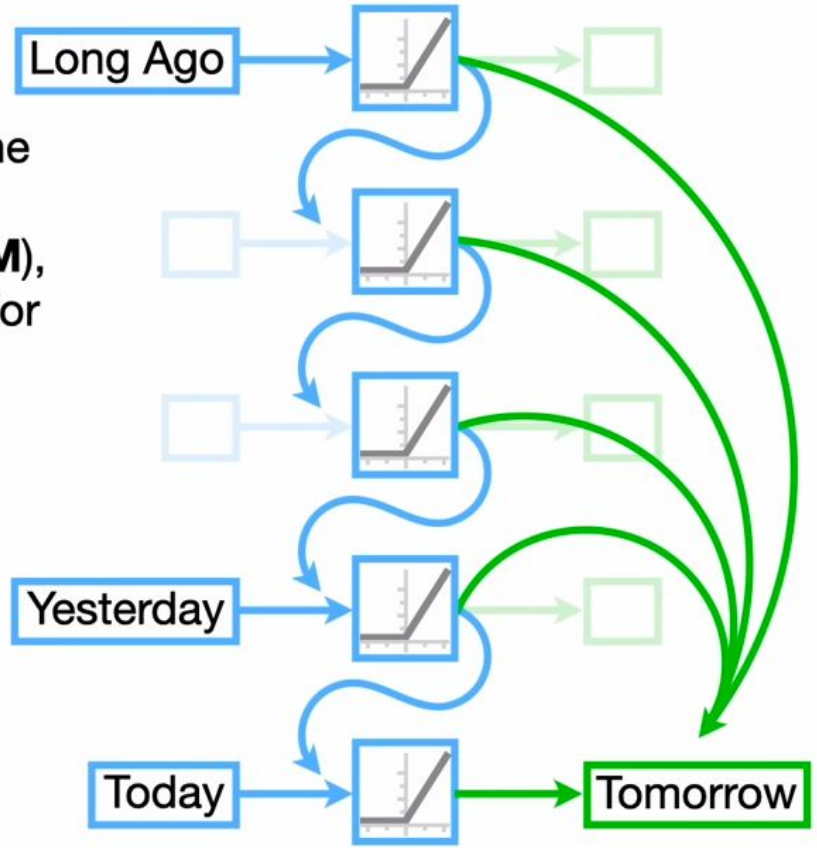


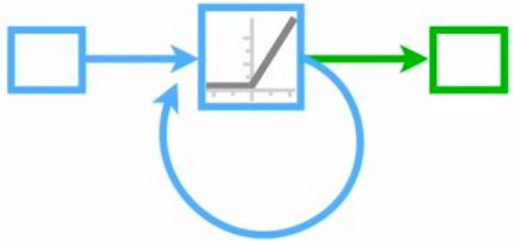
Now that we understand the **main idea** behind **Long Short-Term Memory** (**LSTM**), that it uses different paths for **Long** and **Short-Term Memories**…



Now that we understand the **main idea** behind **Long Short-Term Memory** (**LSTM**), that it uses different paths for **Long** and **Short-Term Memories**…

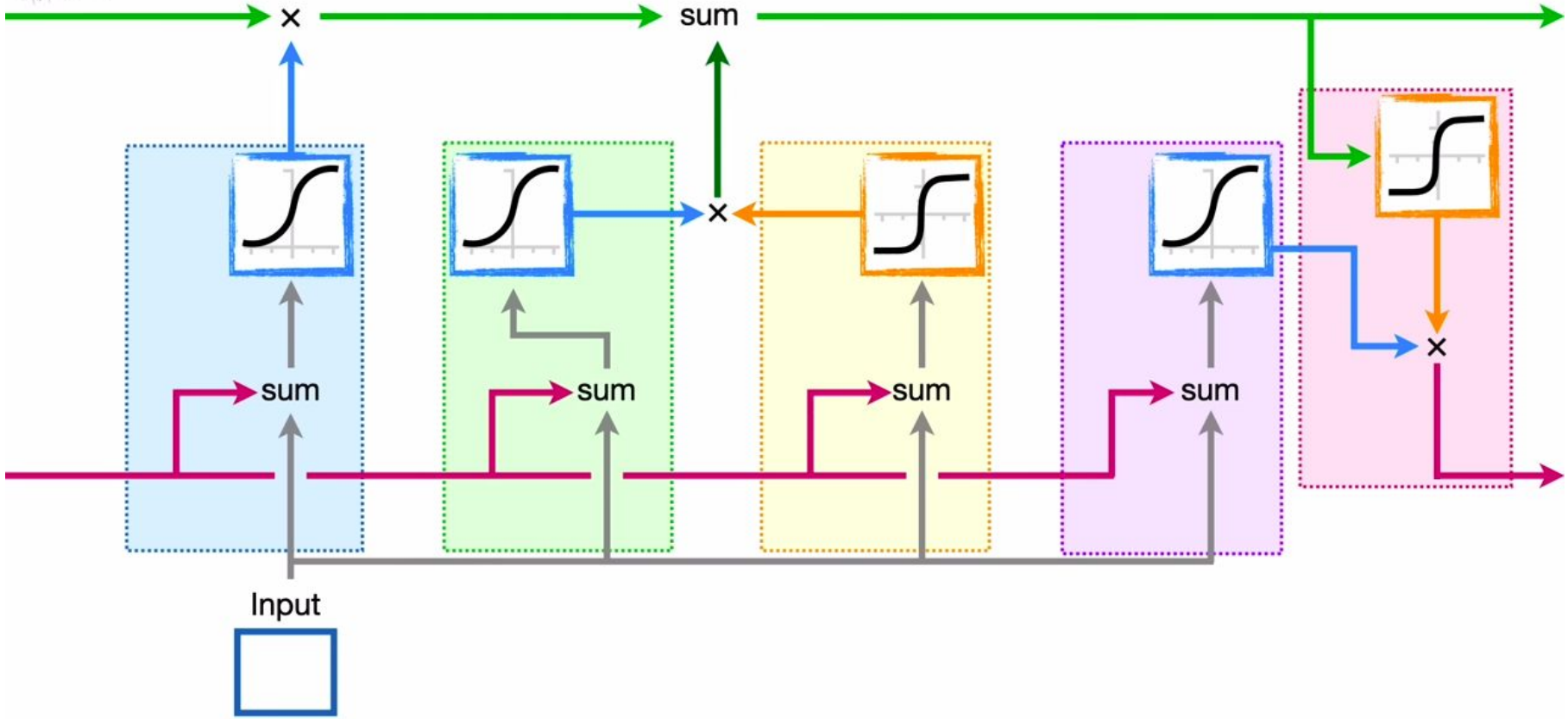
…let's talk about the **details**.



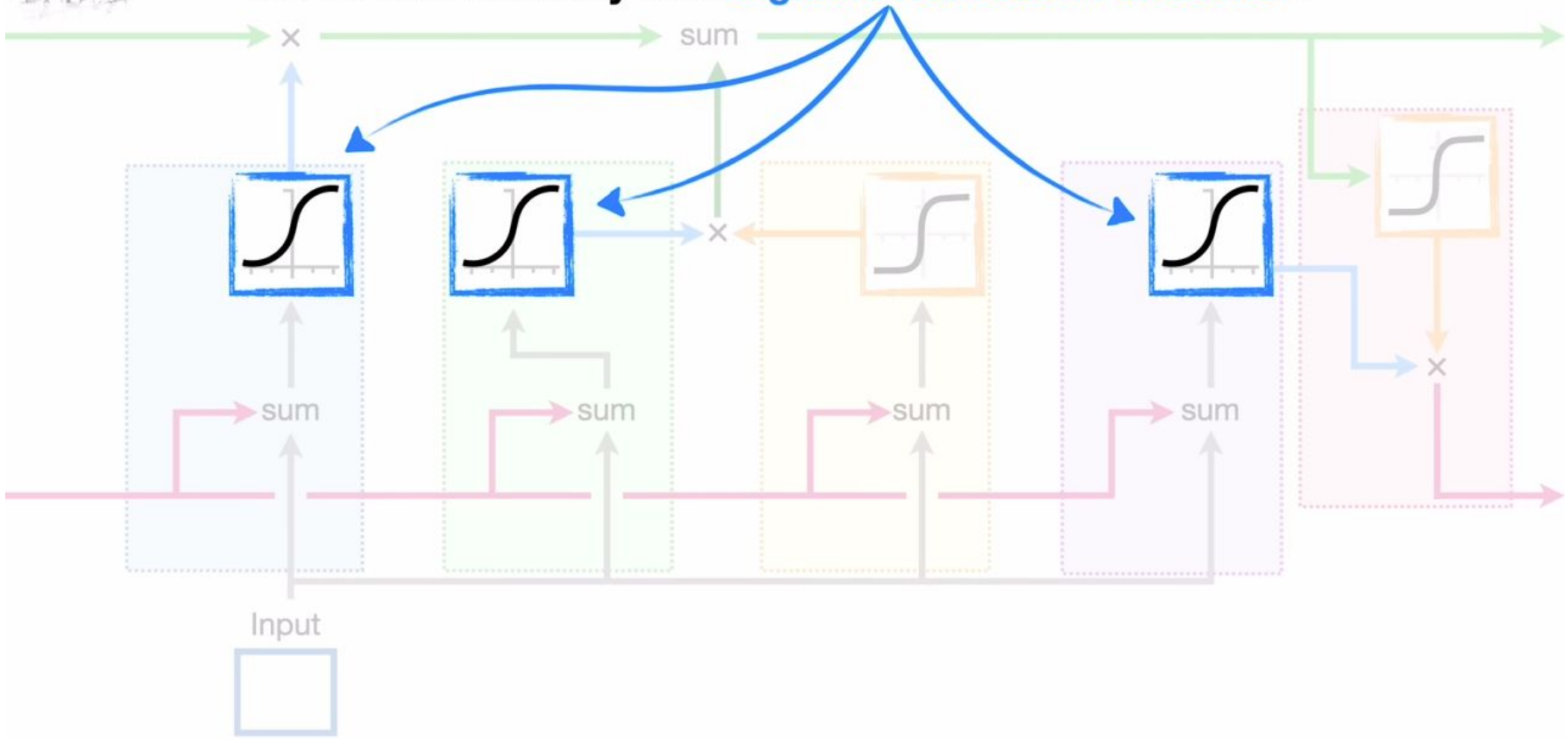


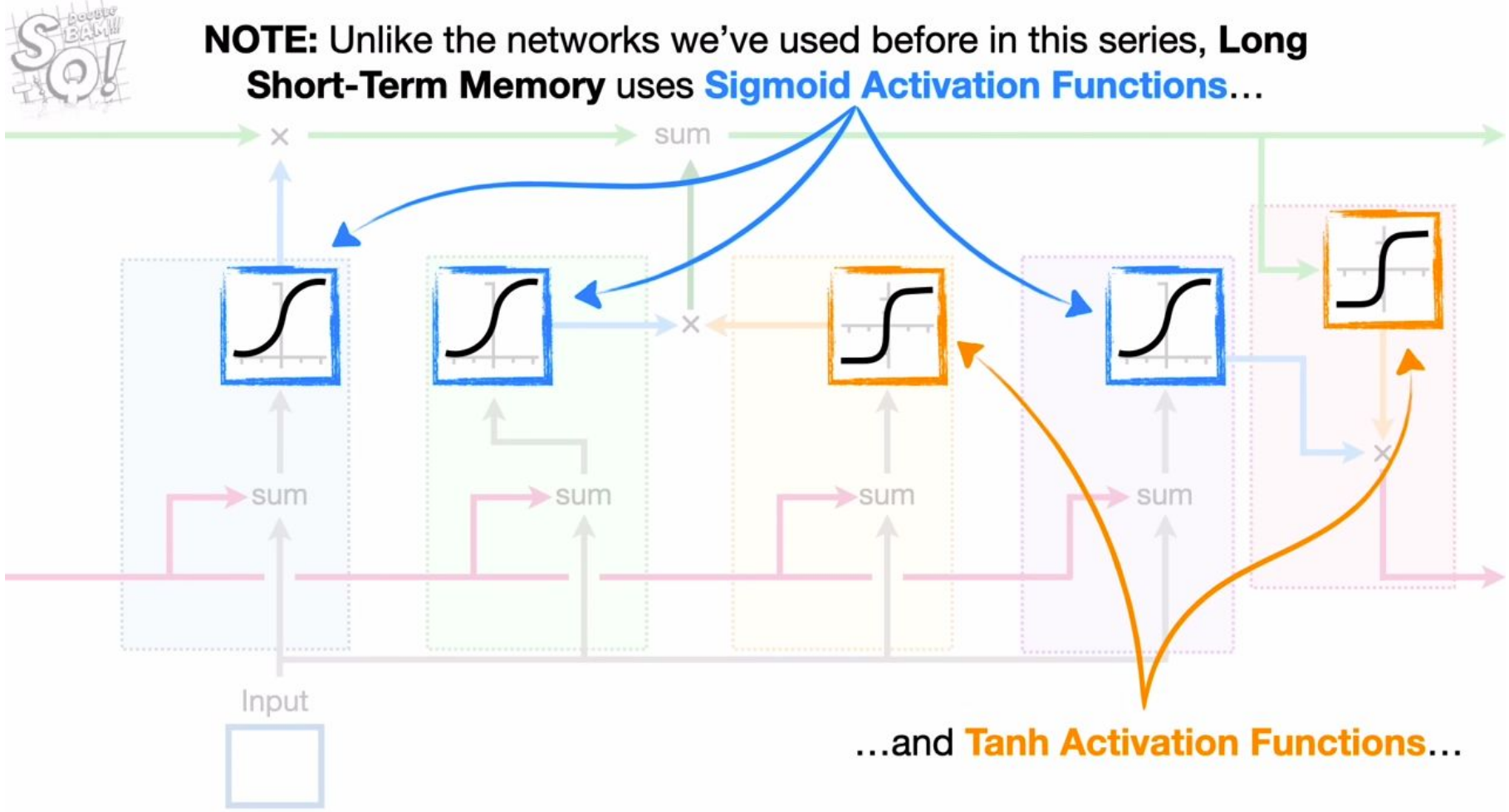
The bad news is that compared to a basic, vanilla **Recurrent Neural Network**, which unrolls from a relatively simple unit…

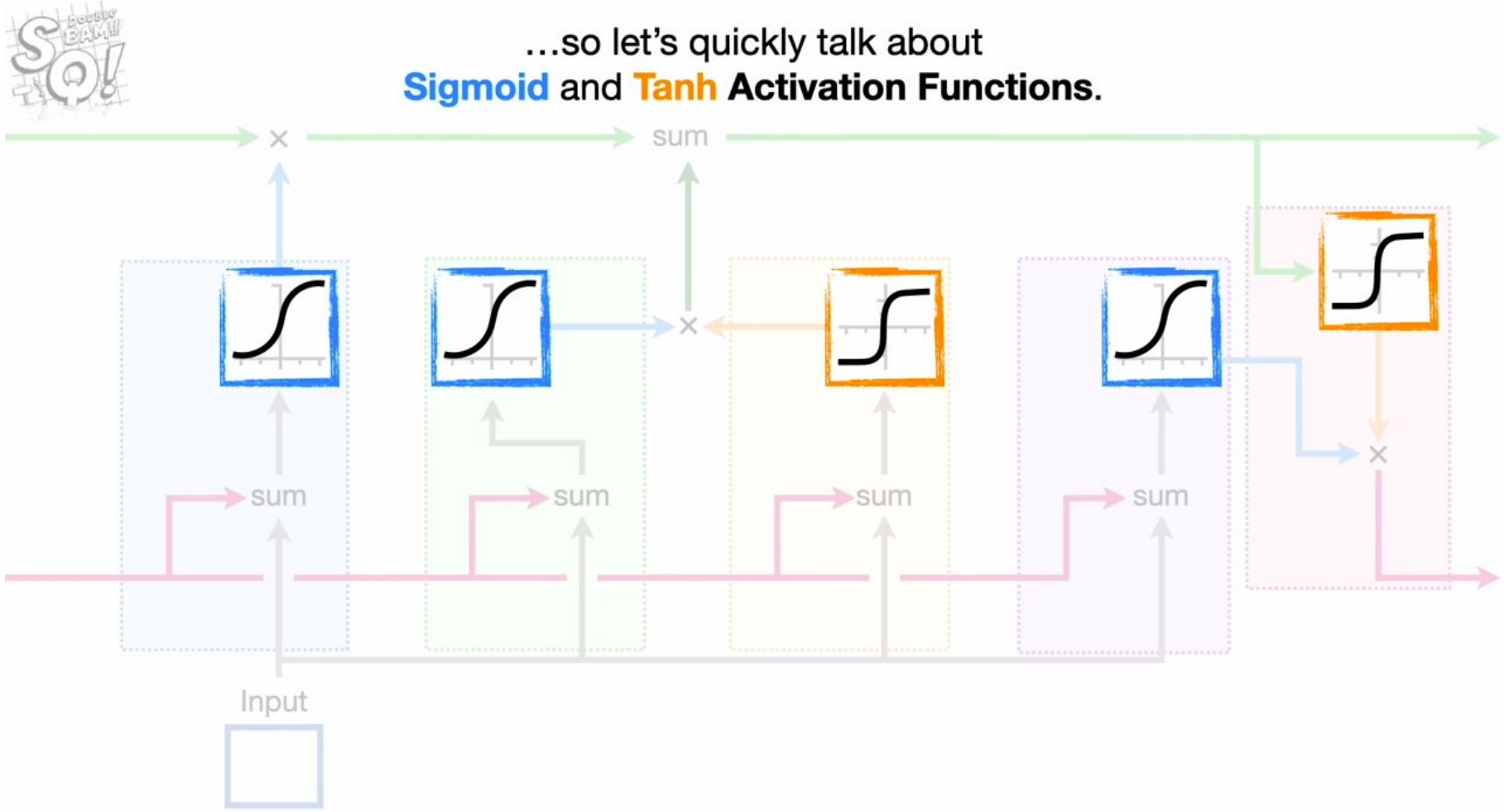
…**Long Short-Term Memory** is based on a much more complicated unit.



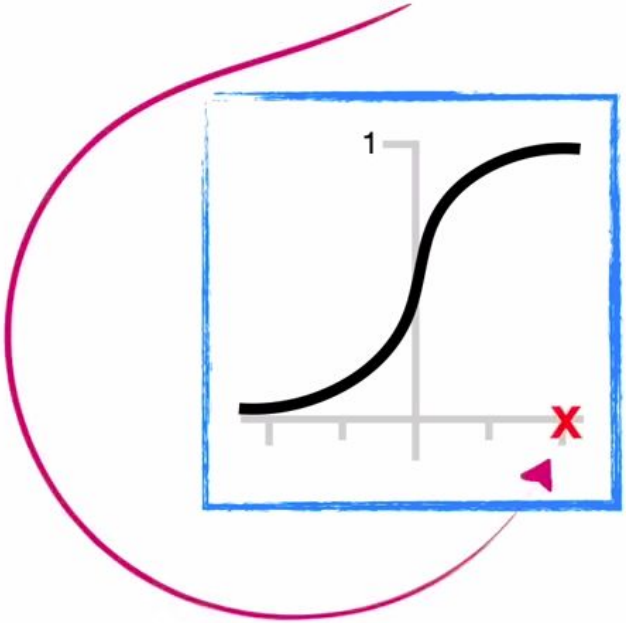
NOTE: Unlike the networks we've used before in this series, **Long Short-Term Memory** uses **Sigmoid Activation Functions**…



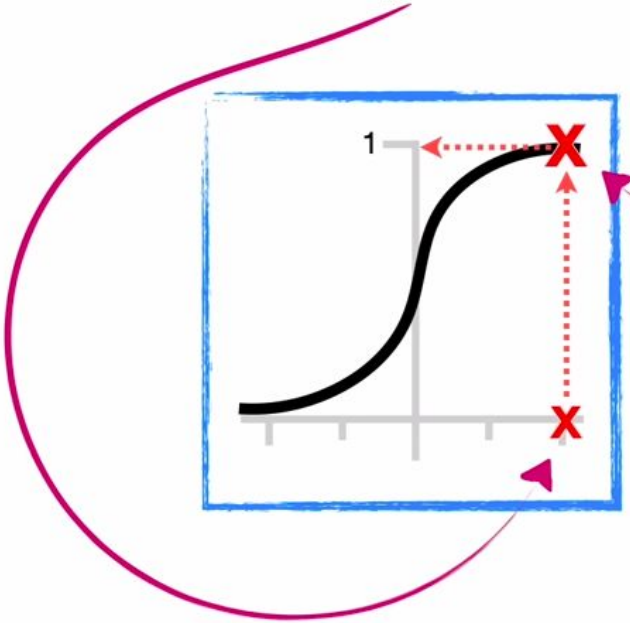

NOTE: Unlike the networks we've used before in this series, Long Short-Term Memory uses Sigmoid Activation Functions…
×
sum
sum
sum
sum
sum
×
×
Input
…and Tanh Activation Functions…


…so let's quickly talk about
Sigmoid and Tanh Activation Functions.
×
sum
sum
sum
sum
sum
×
×
Input

In a nutshell, the **Sigmoid Activation Function** takes any x-axis coordinate…



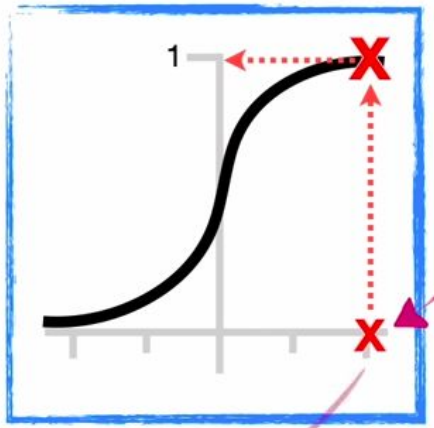
In a nutshell, the **Sigmoid Activation Function** takes any x-axis coordinate…



…and turns it into a y-axis coordinate between **0** and **1**.

For example, when we plug in this x-axis coordinate, **10**, into the equation for the **Sigmoid Activation Function**…

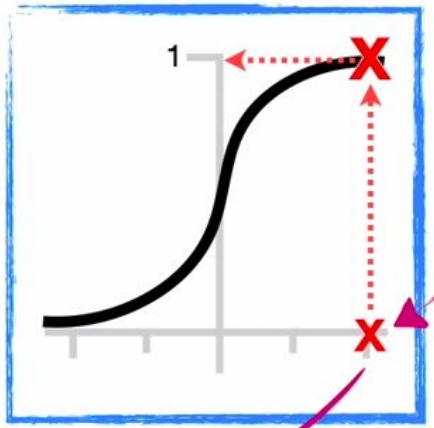
$$f(x) = \frac{e^x}{e^x + 1}$$



For example, when we plug in this x-axis coordinate, **10**, into the equation for the **Sigmoid Activation Function**…

$$f(x) = \frac{e^x}{e^x + 1}$$

$$f(10) = \frac{e^{10}}{e^{10} + 1}$$

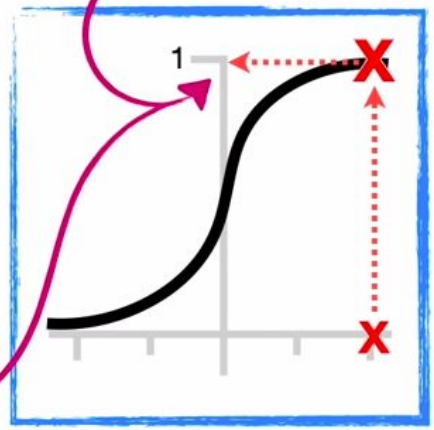


…we get **0.99995** as the y-axis coordinate.

$$f(x) = \frac{e^x}{e^x + 1}$$

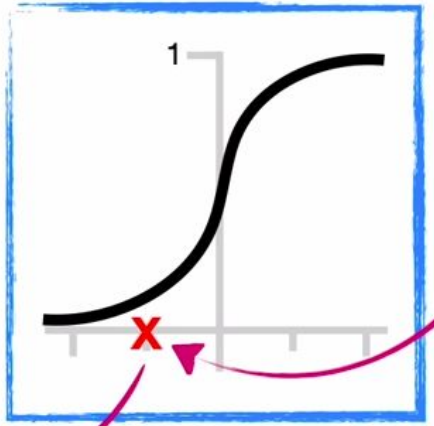
$$f(10) = \frac{e^{10}}{e^{10} + 1}$$

$$= 0.99995$$



And if we plug in this
x-axis coordinate, **-5**…

$$f(x) = \frac{e^x}{e^x + 1}$$

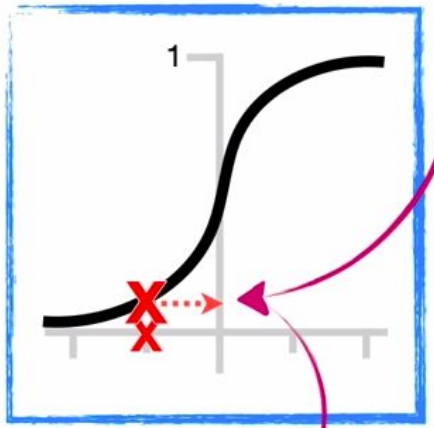


…then we get **0.01** as the y-axis coordinate.

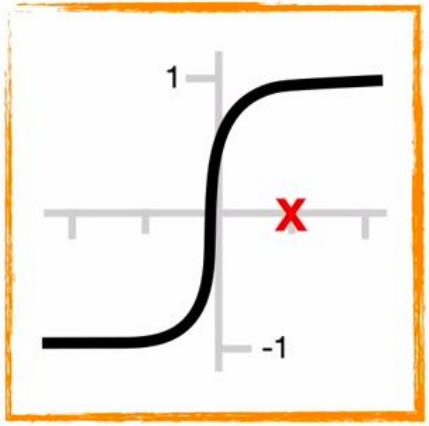
$$f(x) = \frac{e^x}{e^x + 1}$$

$$f(-5) = \frac{e^{-5}}{e^{-5} + 1}$$

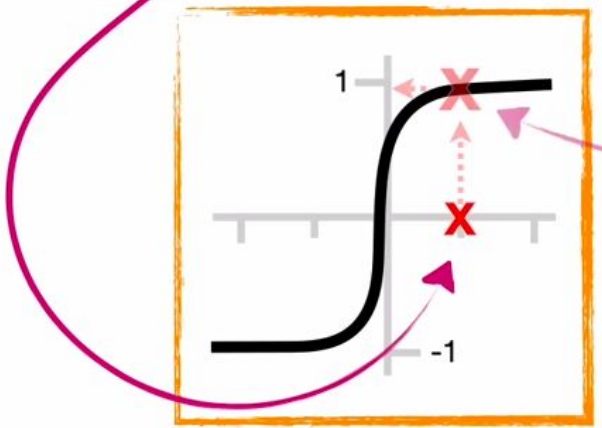
$$= 0.01$$



In contrast, the **tanh**, or **Hyperbolic Tangent**, **Activation Function** takes any x-axis coordinate…



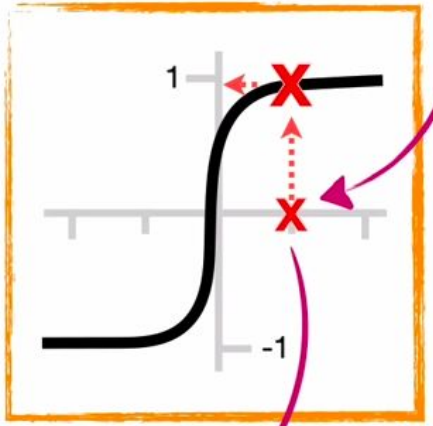
In contrast, the **tanh**, or **Hyperbolic Tangent**, **Activation Function** takes any x-axis coordinate…



…and turns it into a y-axis coordinate between **-1** and **1**.

For example, if we plug this x-axis coordinate, **2**, into the equation for the **Tanh Activation Function**…

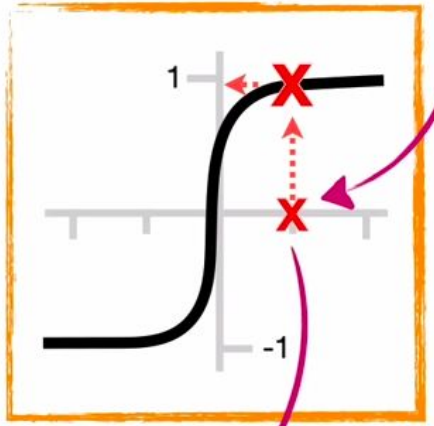
$$f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$



For example, if we plug this x-axis coordinate, **2**, into the equation for the **Tanh Activation Function**…

$$f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

$$f(2) = \frac{e^{2} - e^{-2}}{e^{2} + e^{-2}}$$

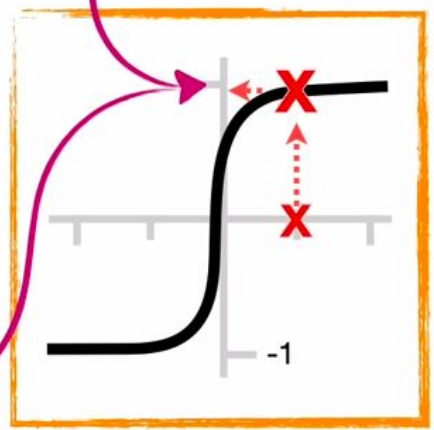


…we get **0.96** as the y-axis coordinate.

$$f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

$$f(2) = \frac{e^{2} - e^{-2}}{e^{2} + e^{-2}}$$

$$= 0.96$$

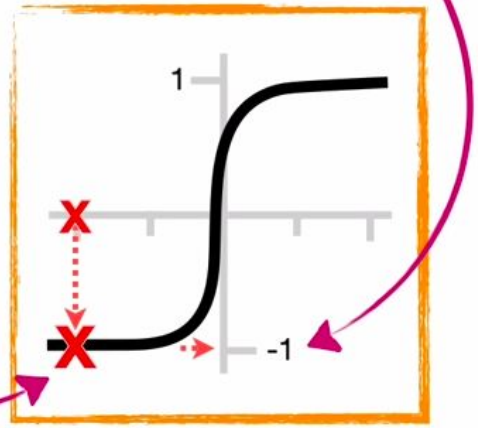


…we get **-1** as the y-axis coordinate.

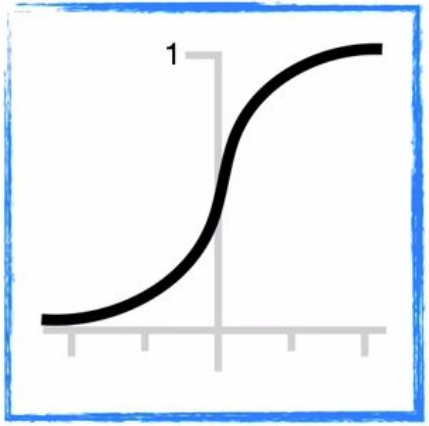
$$f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

$$f(-5) = \frac{e^{-5} - e^{5}}{e^{-5} + e^{5}}$$

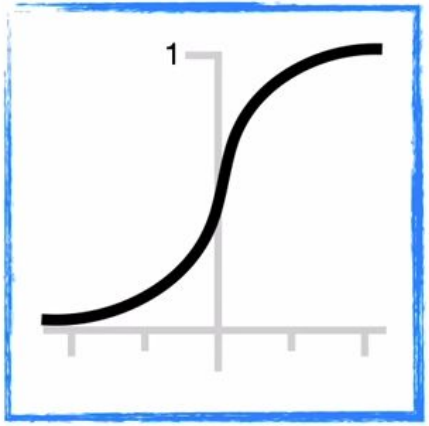
$$= -1$$



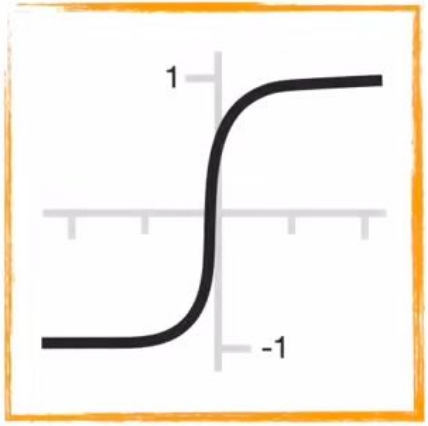
So, now that we know that the **Sigmoid Activation Function** turns any input into a number between **0** and **1**…

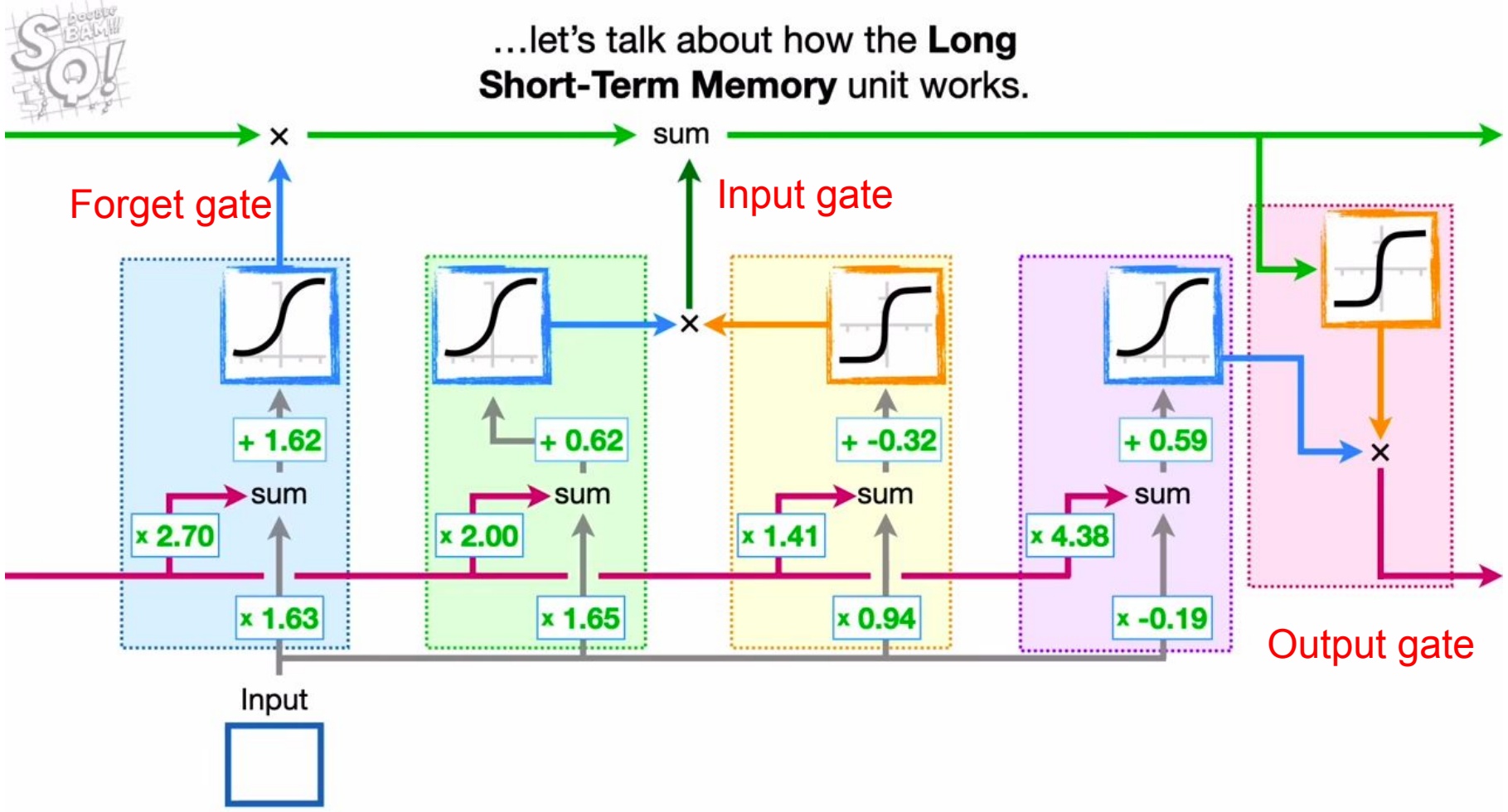


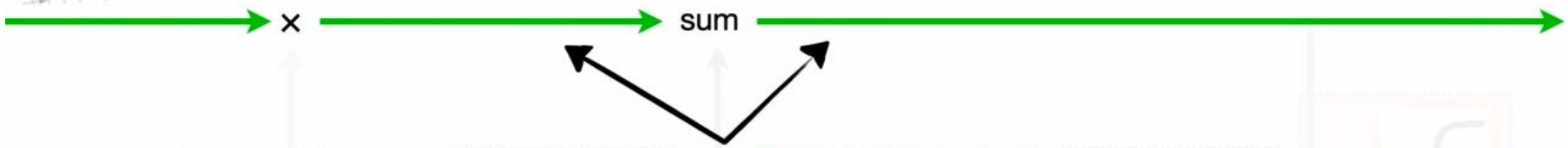
So, now that we know that the **Sigmoid Activation Function** turns any input into a number between **0** and **1**…



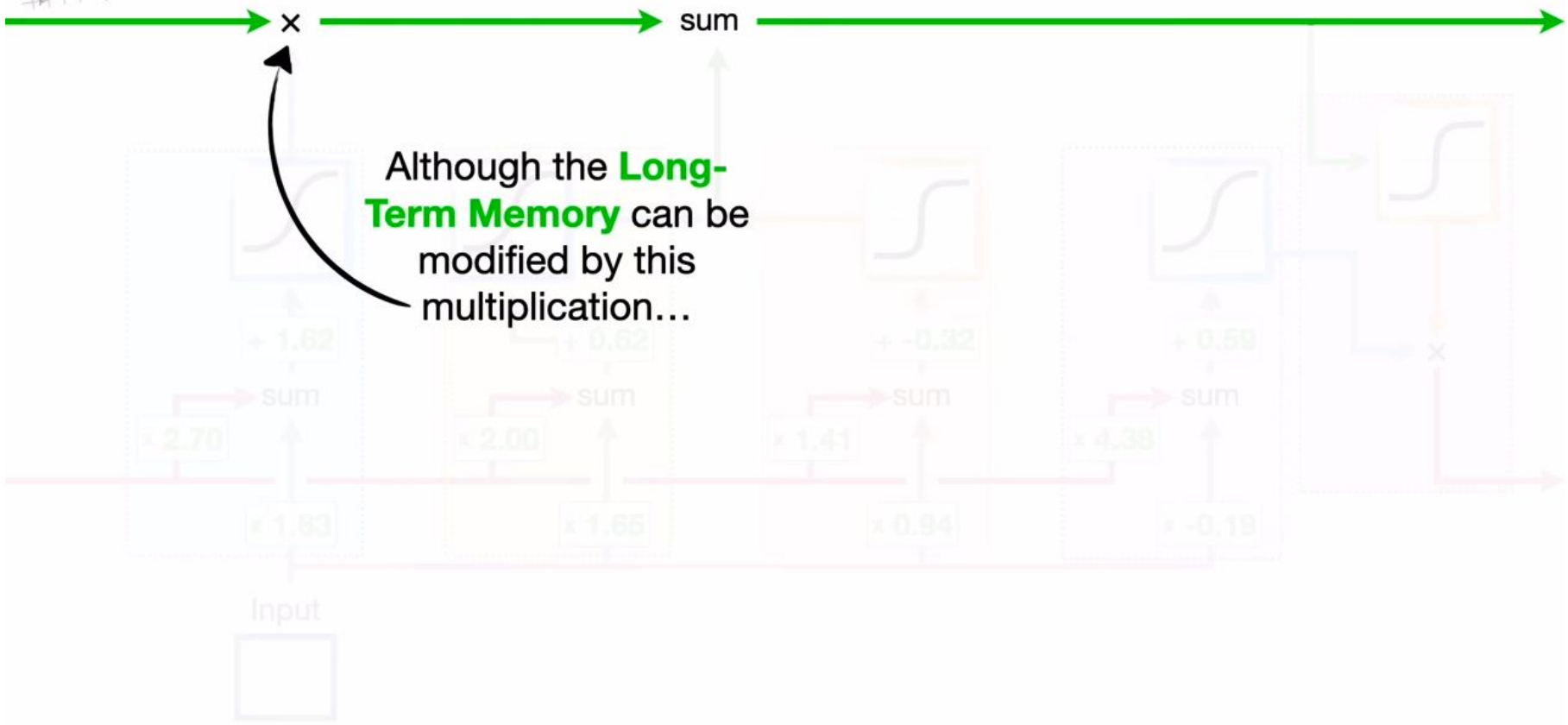
…and the **Tanh Activation Function** turns any input into a number between **-1** and **1**…

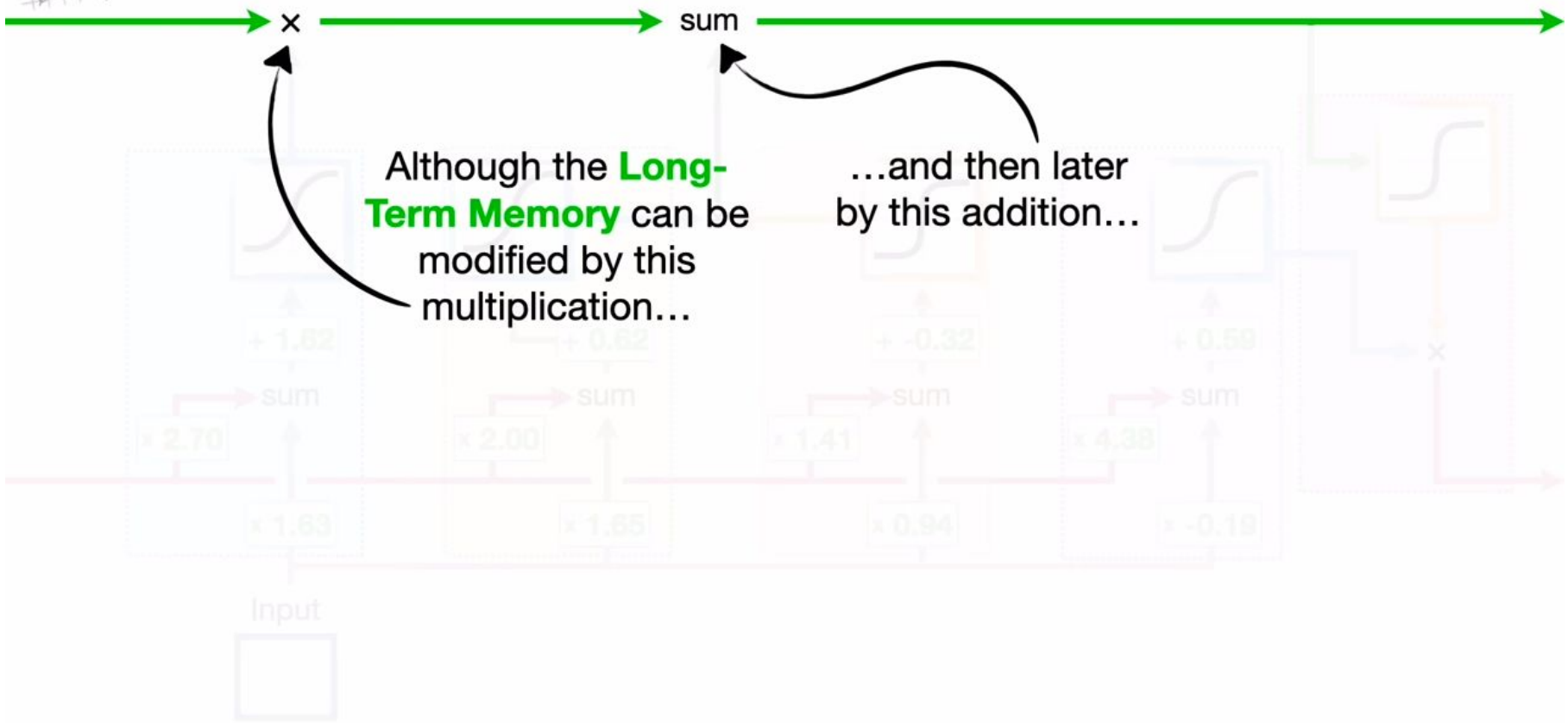


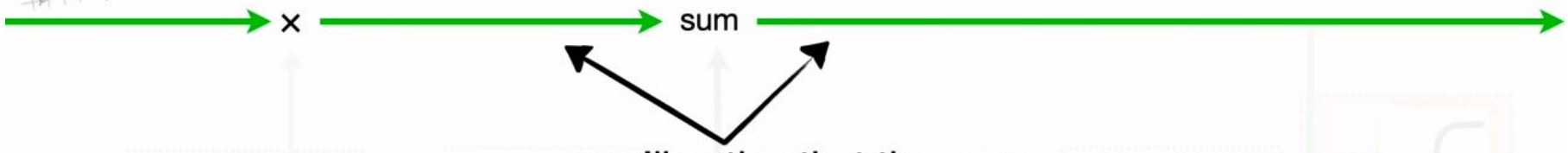

…let's talk about how the Long Short-Term Memory unit works.
×
sum
Forget gate
Input gate
+ 1.62
sum
× 2.70
× 1.63
+ 0.62
sum
× 2.00
× 1.65
×
+ -0.32
sum
× 1.41
× 0.94
+ 0.59
sum
× 4.38
× -0.19
×
Output gate
Input



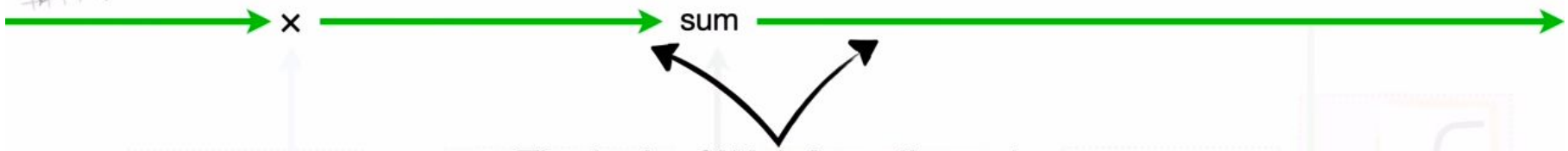
First, this **green line** that runs all the way across the top of the unit is called the **Cell State** and represents the **Long-Term Memory**.


×
sum
Although the Long-Term Memory can be modified by this multiplication…
+ 1.62
sum
× 2.70
× 1.63
+ 0.62
sum
× 2.00
× 1.65
+ -0.32
sum
× 1.41
× 0.94
+ 0.59
sum
× 4.38
× -0.19
×
Input

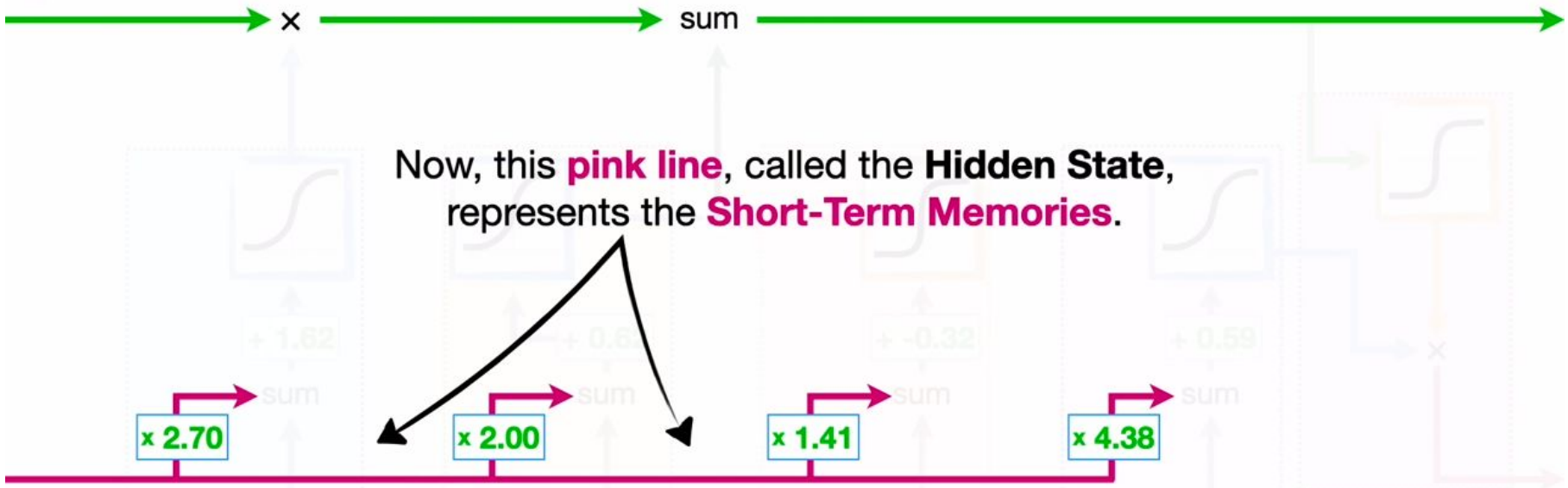

×
sum
Although the Long-Term Memory can be modified by this multiplication…
…and then later by this addition…
Input

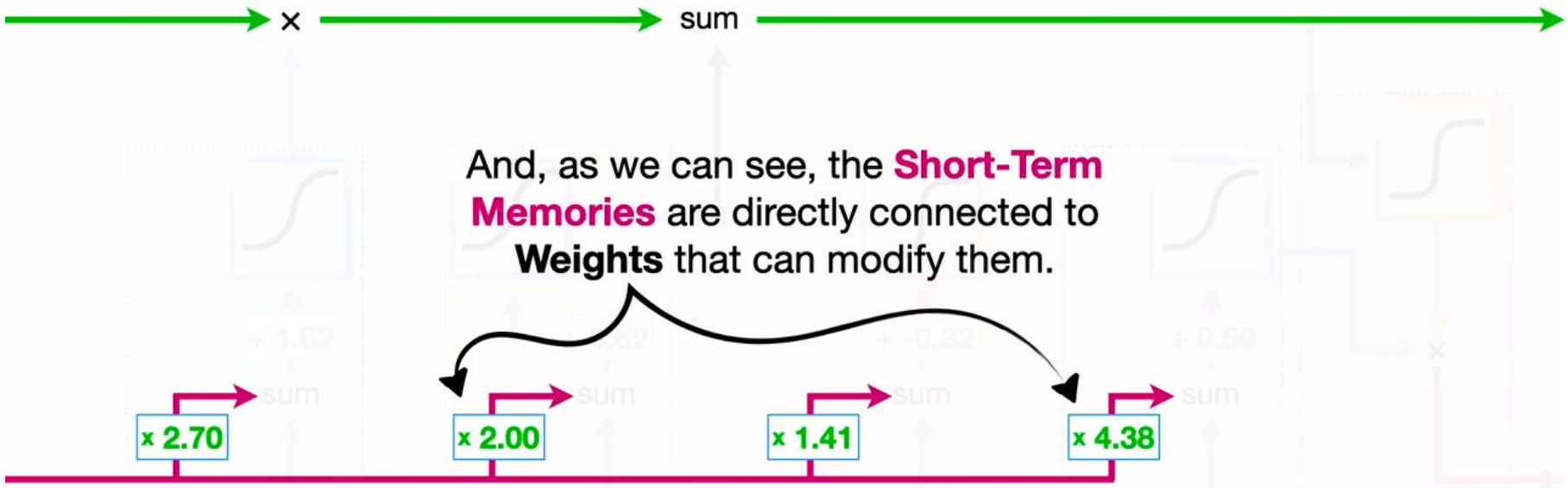


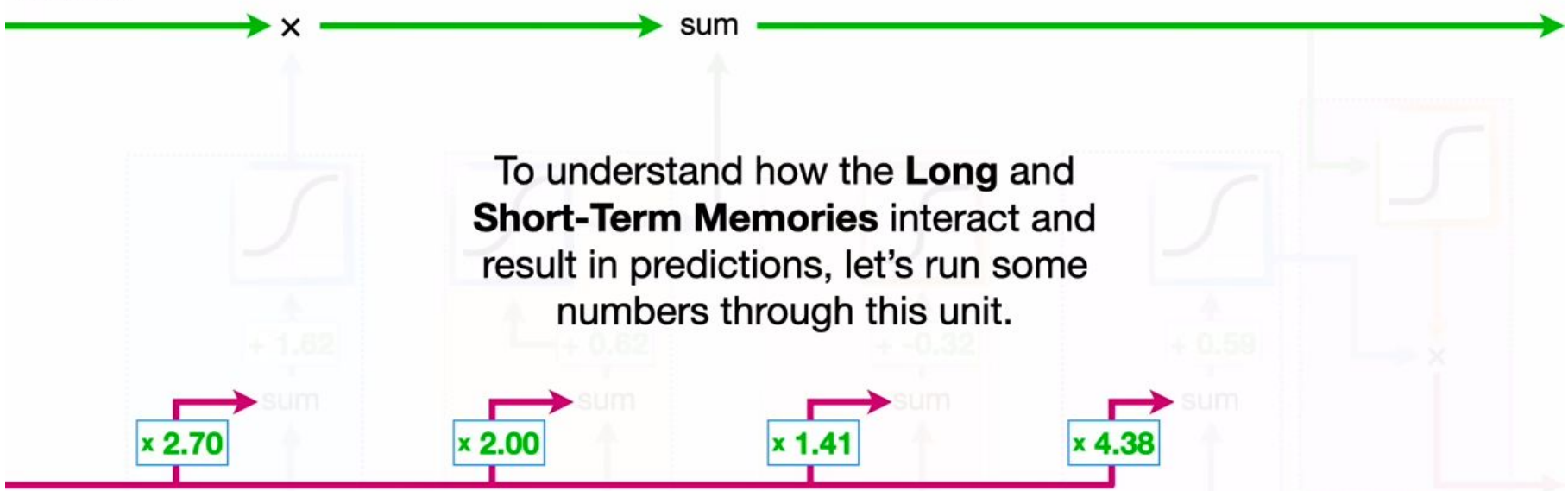
…you'll notice that there are no **Weights** and **Biases** that can modify it directly.

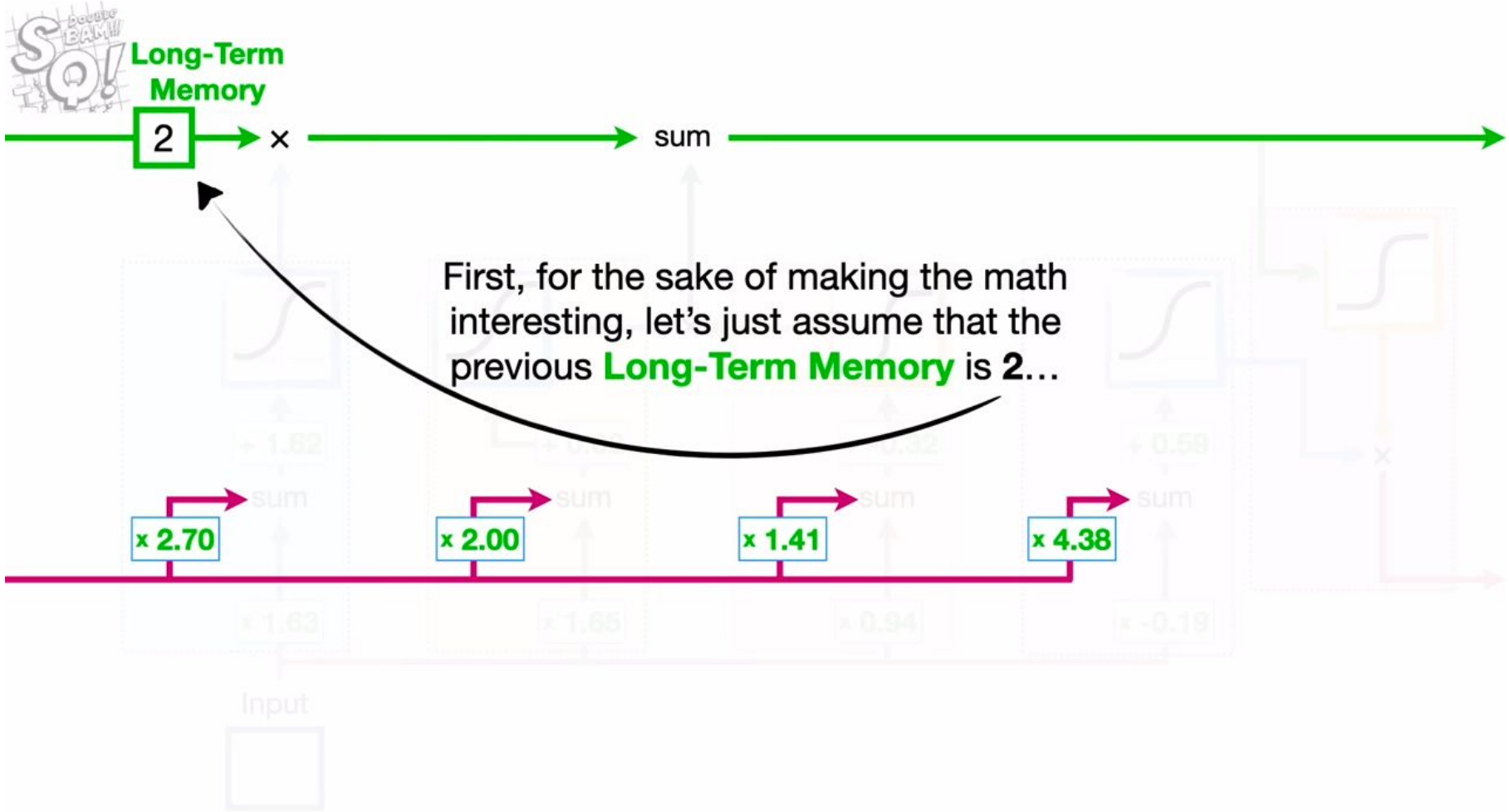


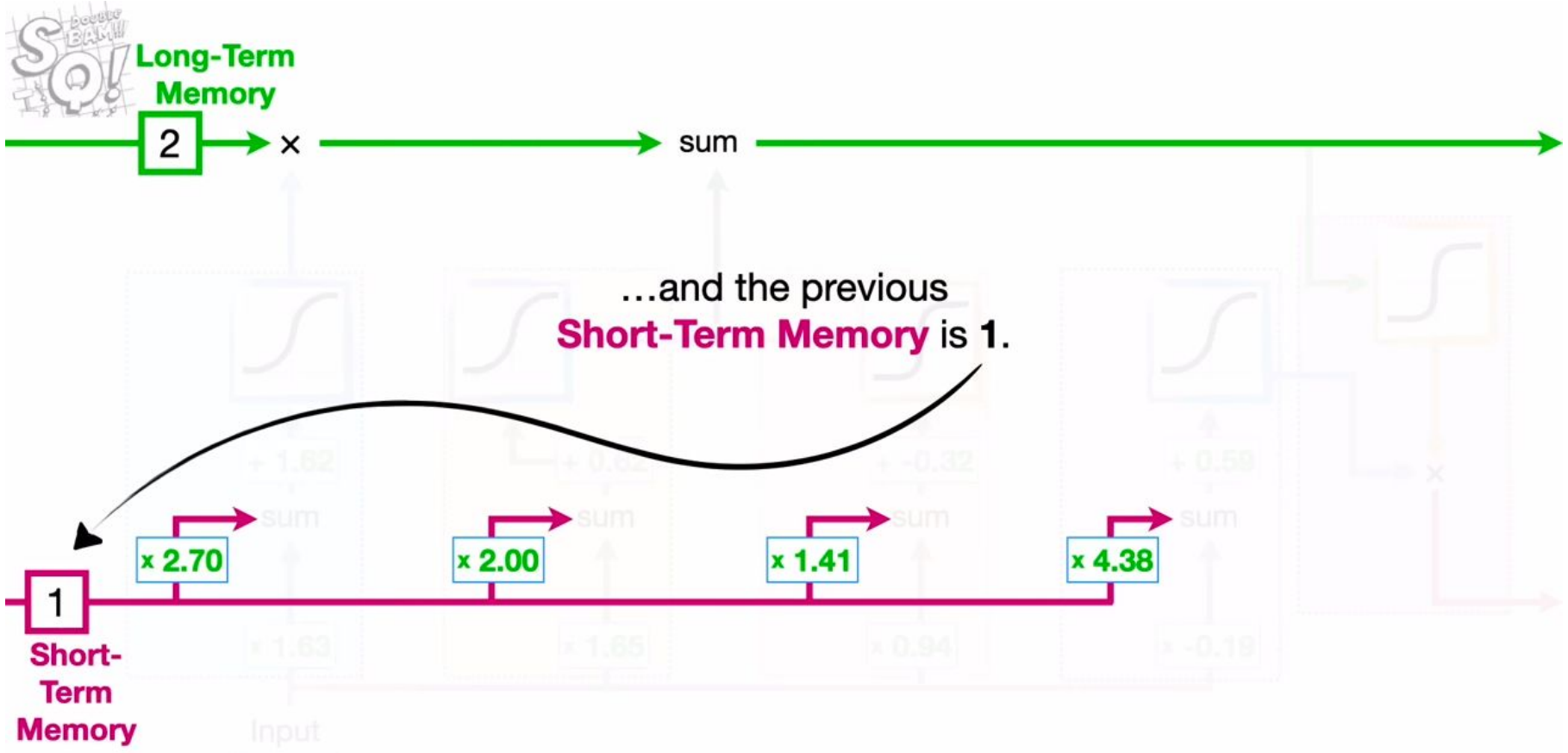
The lack of **Weights** allows the **Long-Term Memories** to flow through a series of unrolled units without causing the gradient to **explode** or **vanish**.

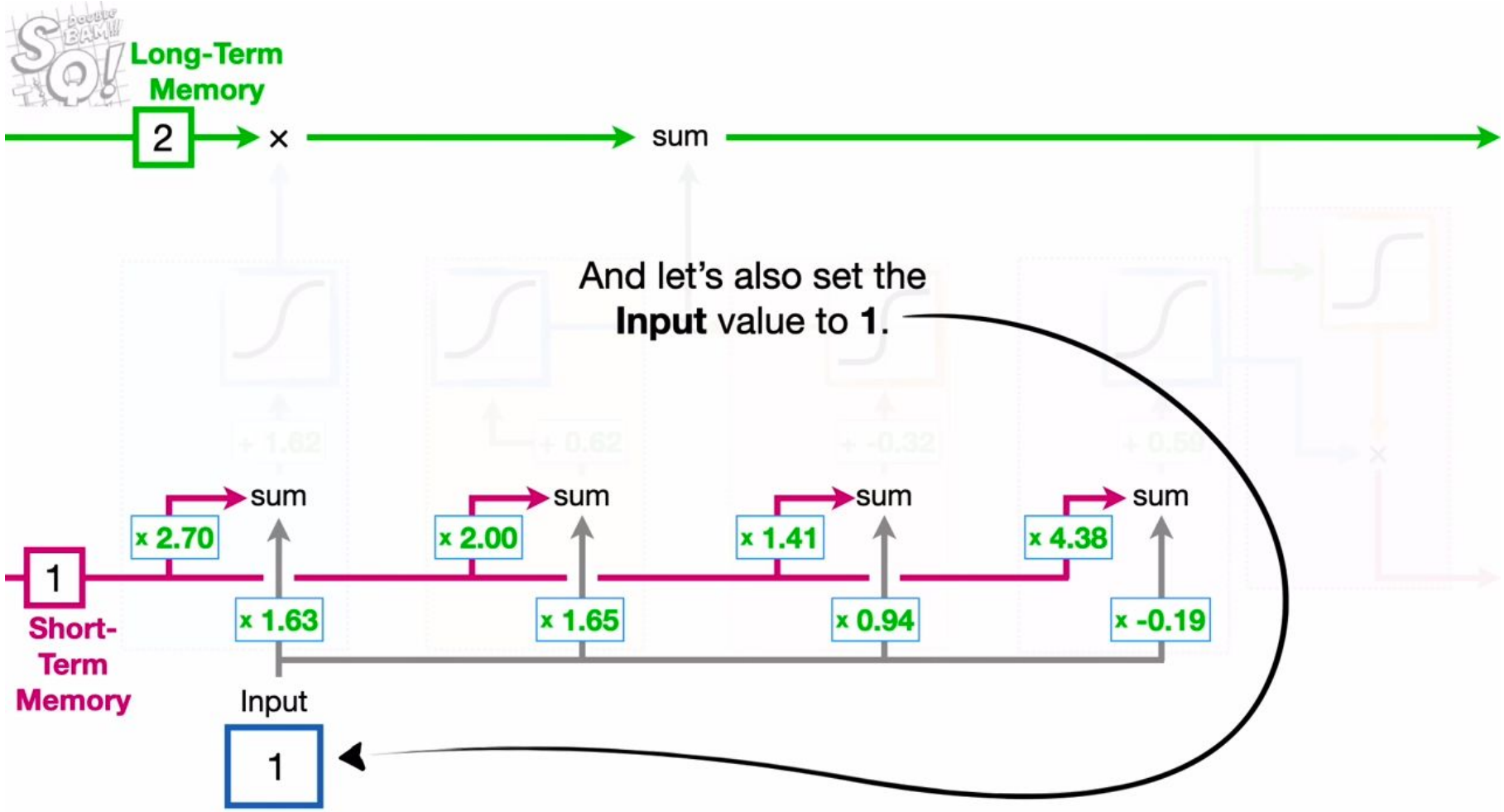

×
sum
Now, this pink line, called the Hidden State,
represents the Short-Term Memories.
x 2.70
x 2.00
x 1.41
x 4.38

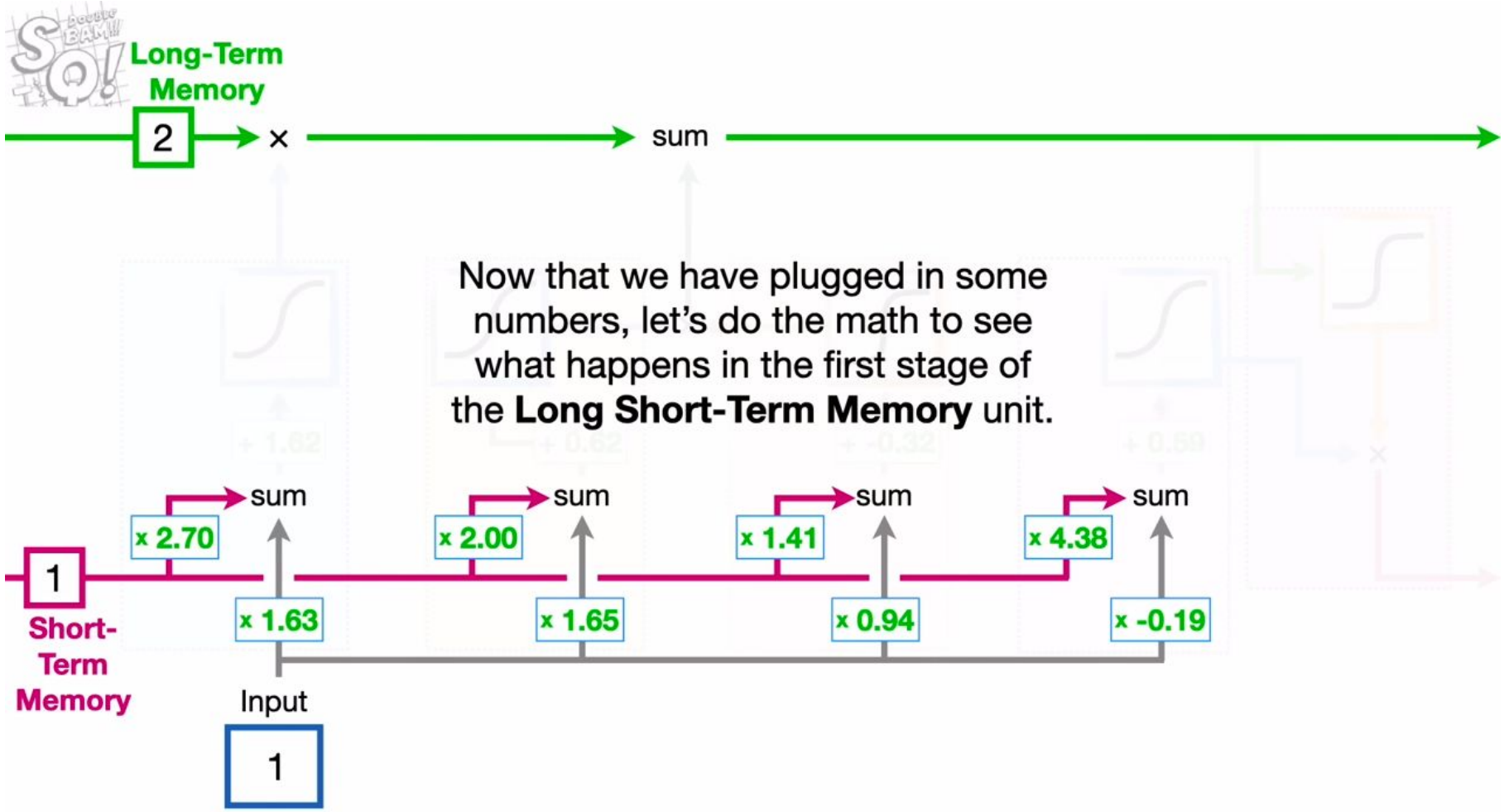

×
sum
And, as we can see, the Short-Term Memories are directly connected to Weights that can modify them.
x 2.70
x 2.00
x 1.41
x 4.38

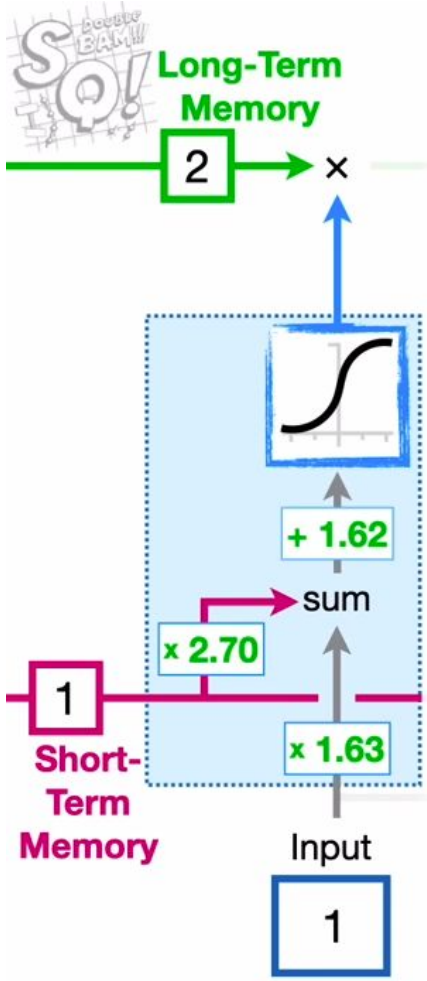

×
sum
To understand how the **Long** and **Short-Term Memories** interact and result in predictions, let's run some numbers through this unit.
x 2.70
x 2.00
x 1.41
x 4.38


Long-Term Memory
2
×
sum
First, for the sake of making the math interesting, let's just assume that the previous Long-Term Memory is 2…
× 2.70
× 2.00
× 1.41
× 4.38

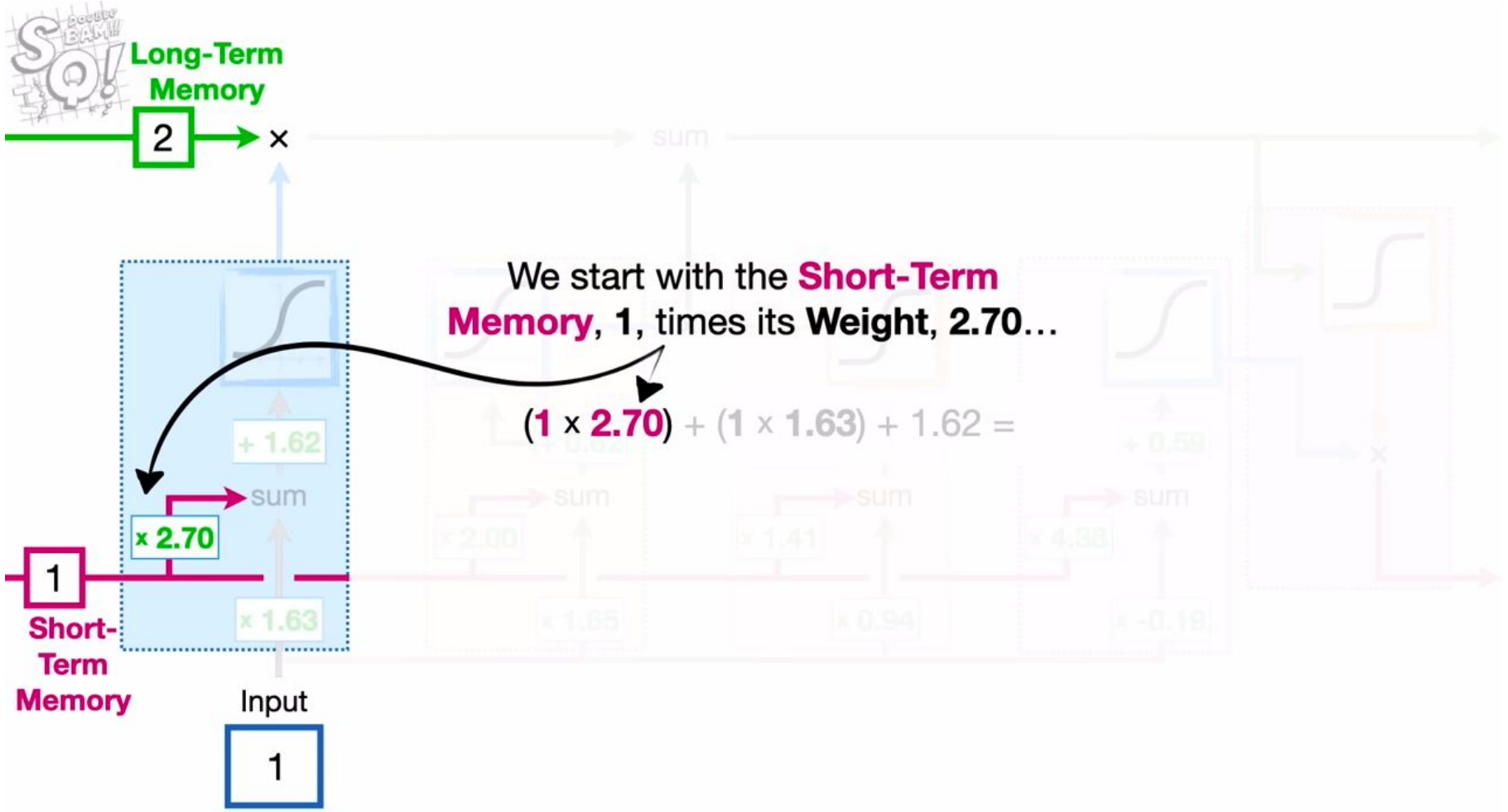

Long-Term Memory
2
×
sum
…and the previous
Short-Term Memory is 1.
x 2.70
x 2.00
x 1.41
x 4.38
1
Short-Term Memory

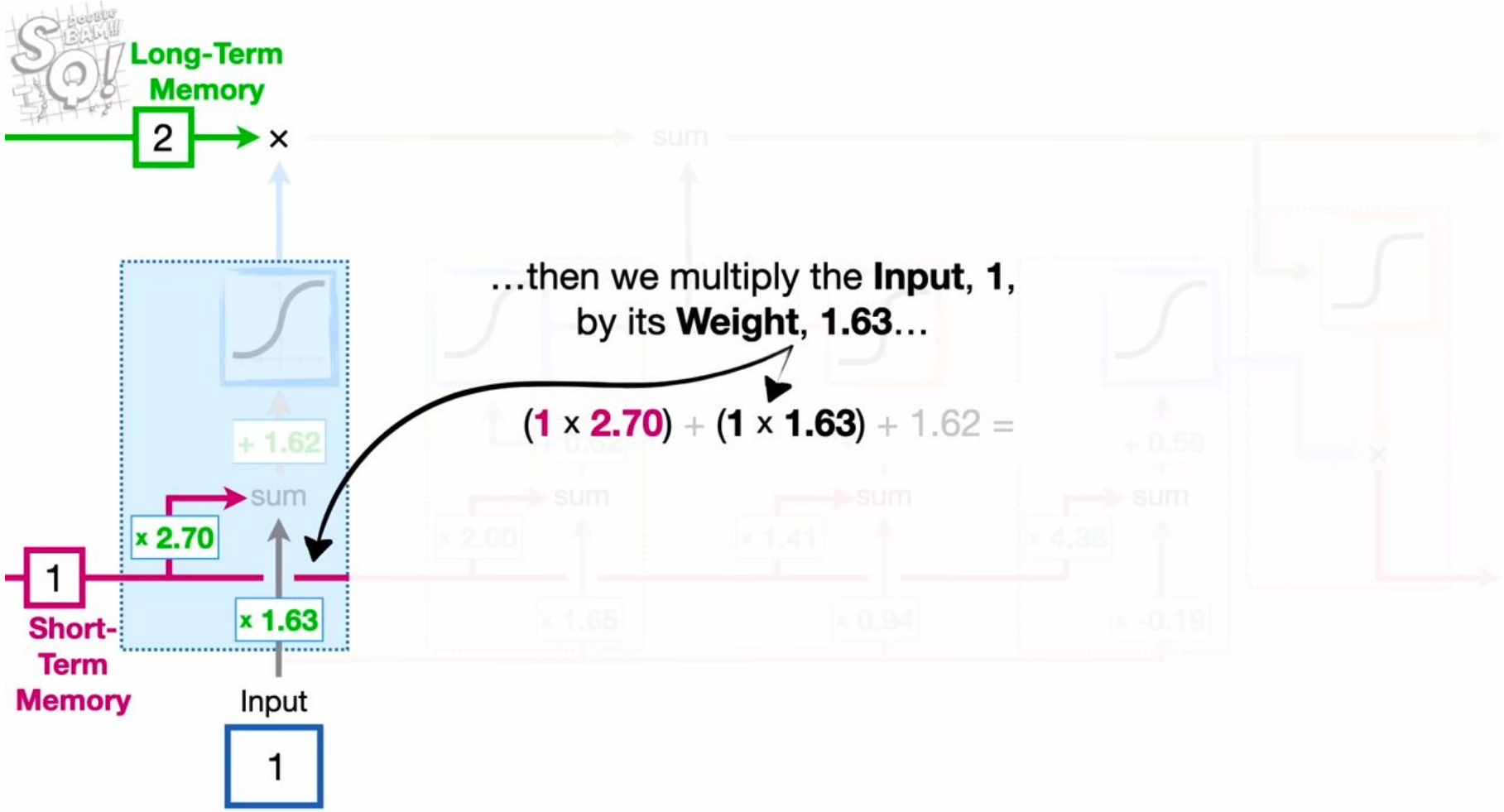

Long-Term Memory
2
×
sum
And let's also set the **Input** value to **1**.
sum
sum
sum
sum
x 2.70
x 2.00
x 1.41
x 4.38
1
x 1.63
x 1.65
x 0.94
x -0.19
Short-Term Memory
Input
1

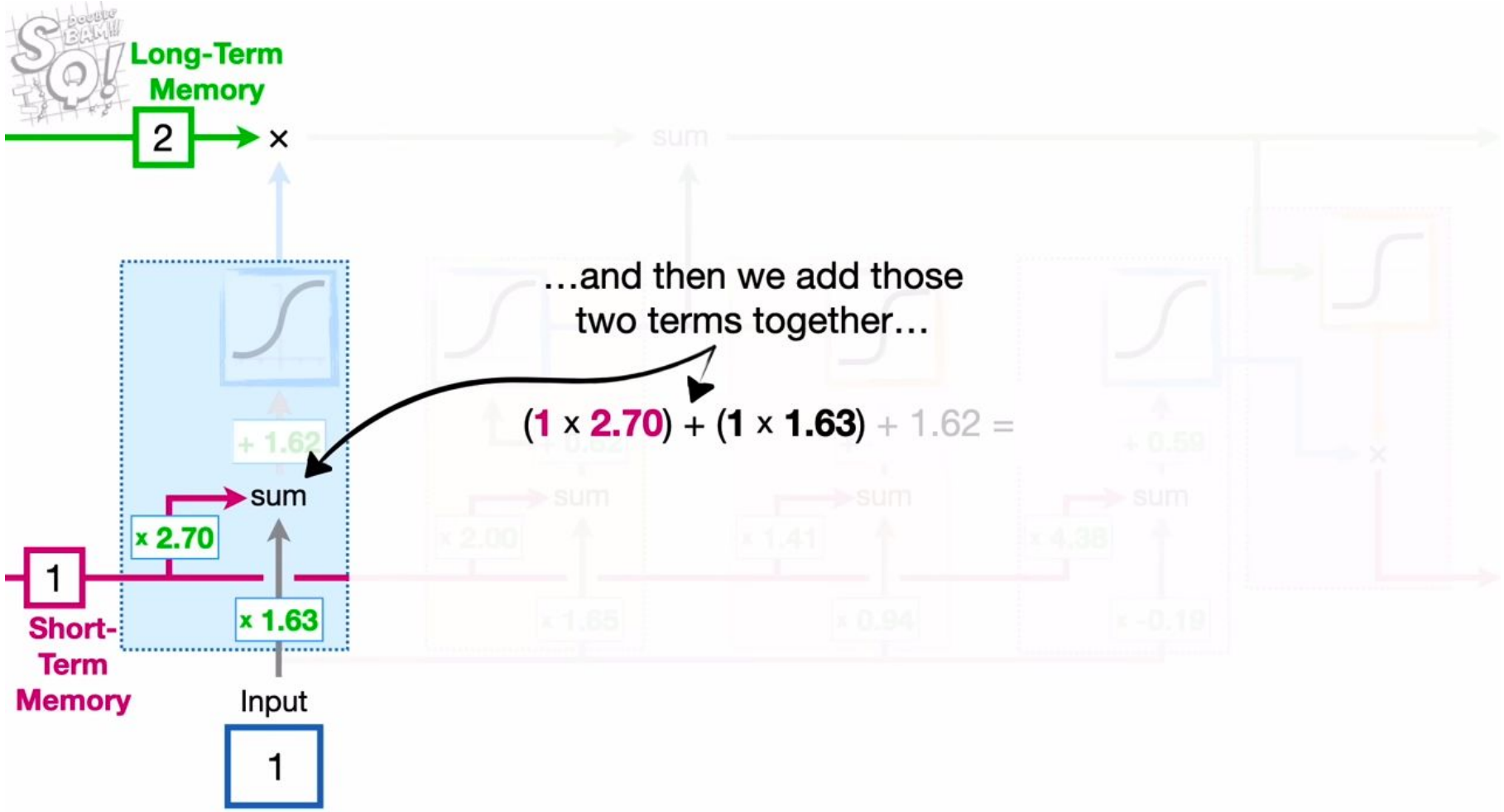

Long-Term Memory
2
×
sum
Now that we have plugged in some numbers, let's do the math to see what happens in the first stage of the **Long Short-Term Memory** unit.
sum
sum
sum
sum
x 2.70
x 2.00
x 1.41
x 4.38
1
x 1.63
x 1.65
x 0.94
x -0.19
Short-Term Memory
Input
1

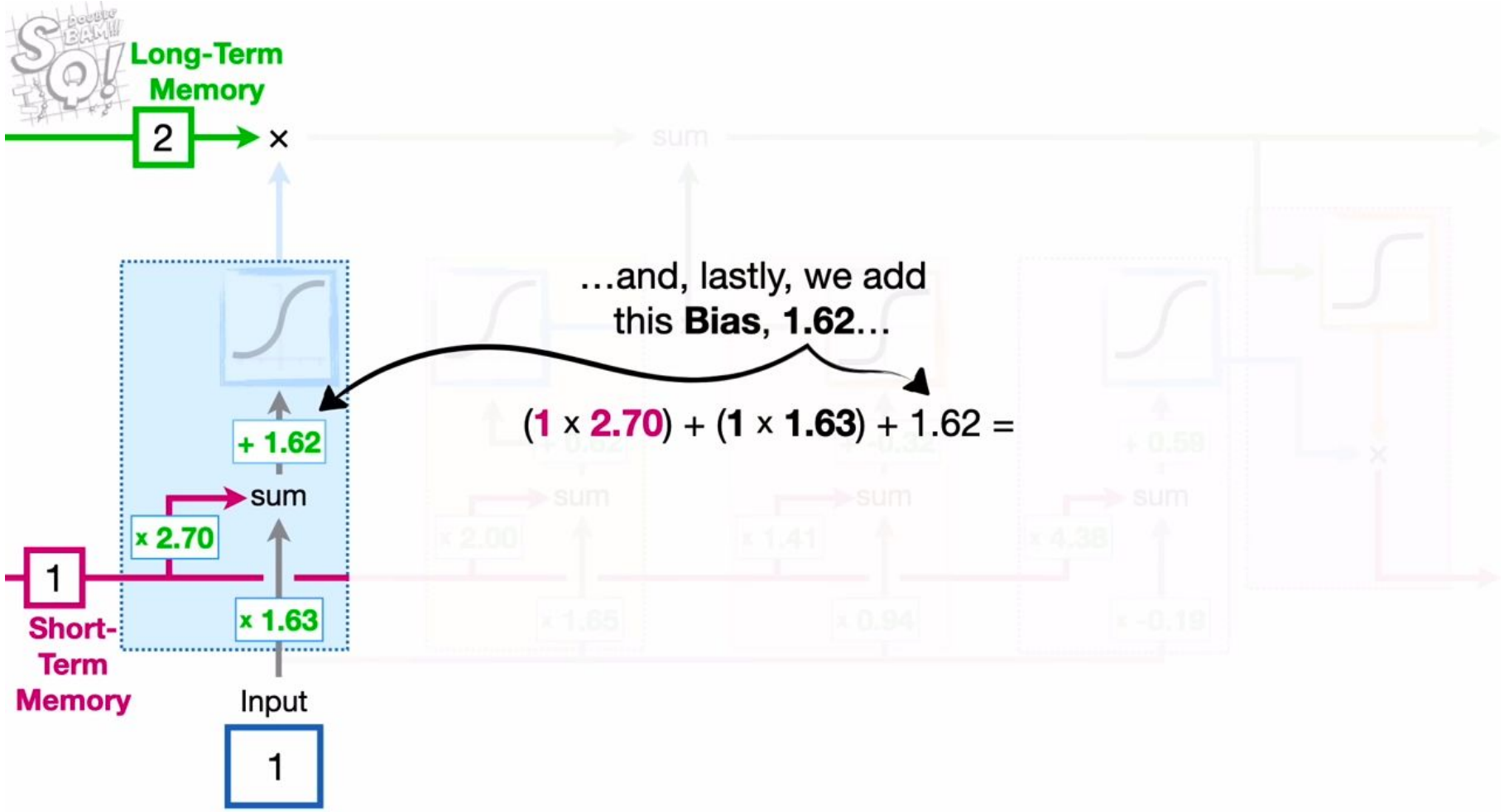


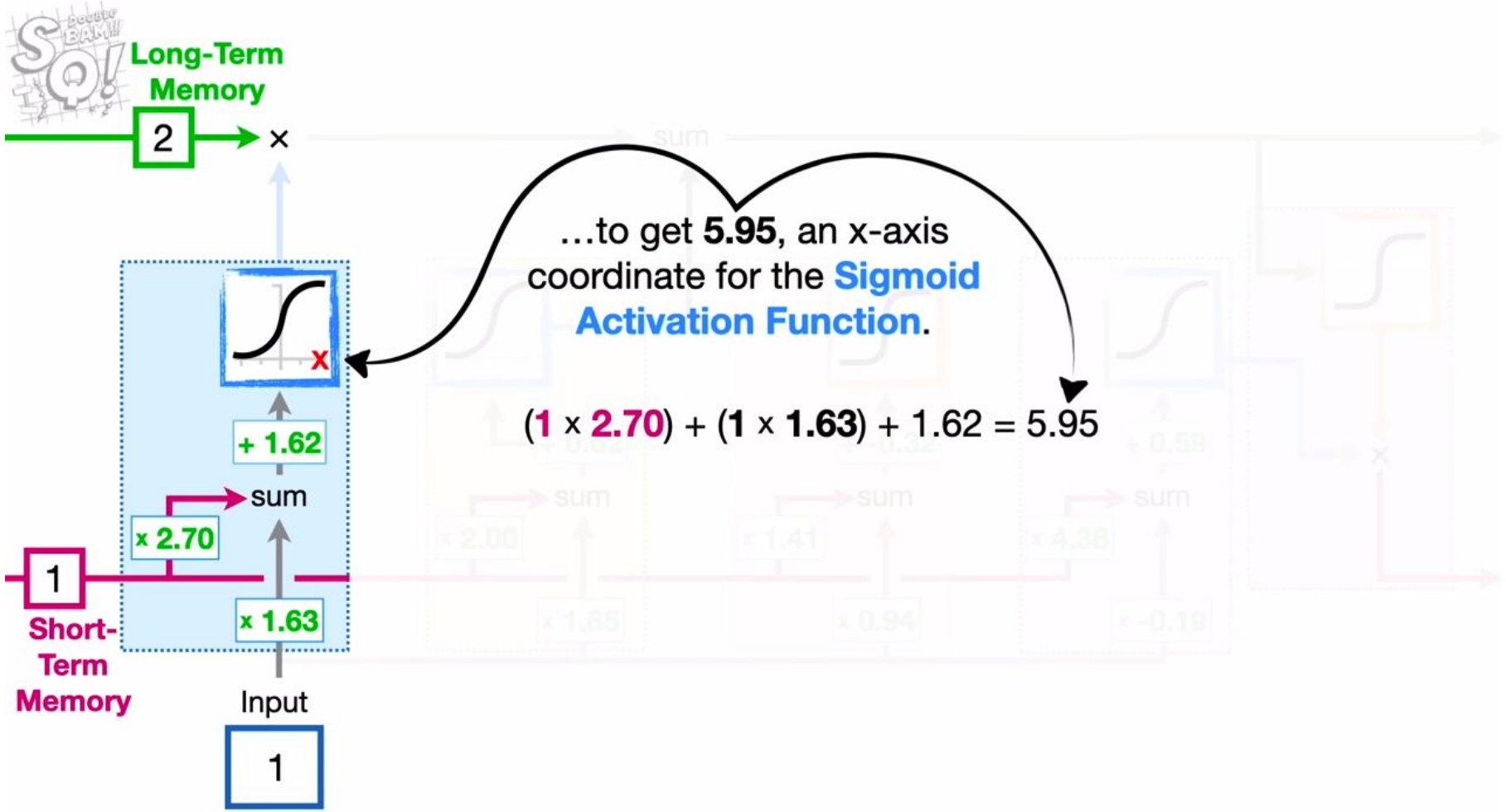
Now that we have plugged in some numbers, let's do the math to see what happens in the first stage of the **Long Short-Term Memory** unit.

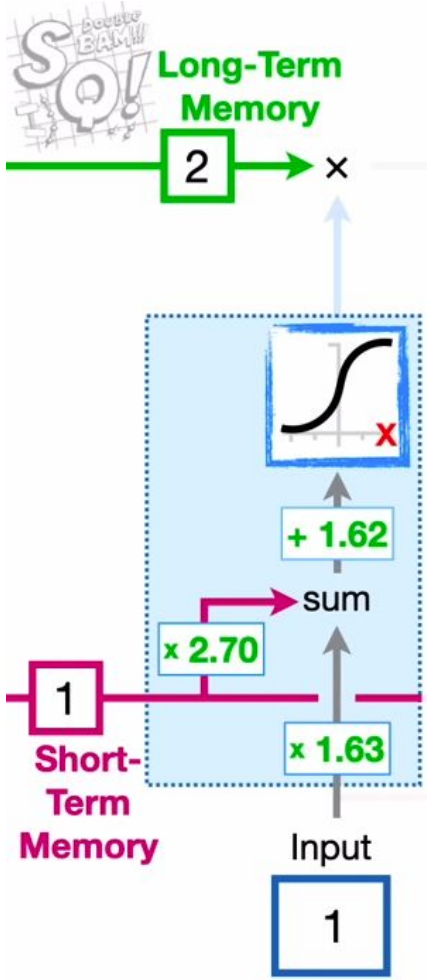

Long-Term Memory
2
×
+ 1.62
sum
× 2.70
1
Short-Term Memory
× 1.63
Input
1
We start with the Short-Term Memory, 1, times its Weight, 2.70…
(1 x 2.70) + (1 x 1.63) + 1.62 =


Long-Term Memory
2
×
+ 1.62
sum
x 2.70
1
Short-Term Memory
x 1.63
Input
1
…then we multiply the Input, 1, by its Weight, 1.63…
(1 x 2.70) + (1 x 1.63) + 1.62 =


Long-Term Memory
2
×
…and then we add those two terms together…
(1 x 2.70) + (1 x 1.63) + 1.62 =
+ 1.62
sum
x 2.70
1
Short-Term Memory
x 1.63
Input
1


Long-Term Memory
2
×
…and, lastly, we add this Bias, 1.62…
(1 x 2.70) + (1 x 1.63) + 1.62 =
+ 1.62
sum
x 2.70
1
Short-Term Memory
x 1.63
Input
1


Long-Term Memory
2
×
x
+ 1.62
sum
x 2.70
1
Short-Term Memory
x 1.63
Input
1
…to get 5.95, an x-axis coordinate for the Sigmoid Activation Function.
(1 x 2.70) + (1 x 1.63) + 1.62 = 5.95



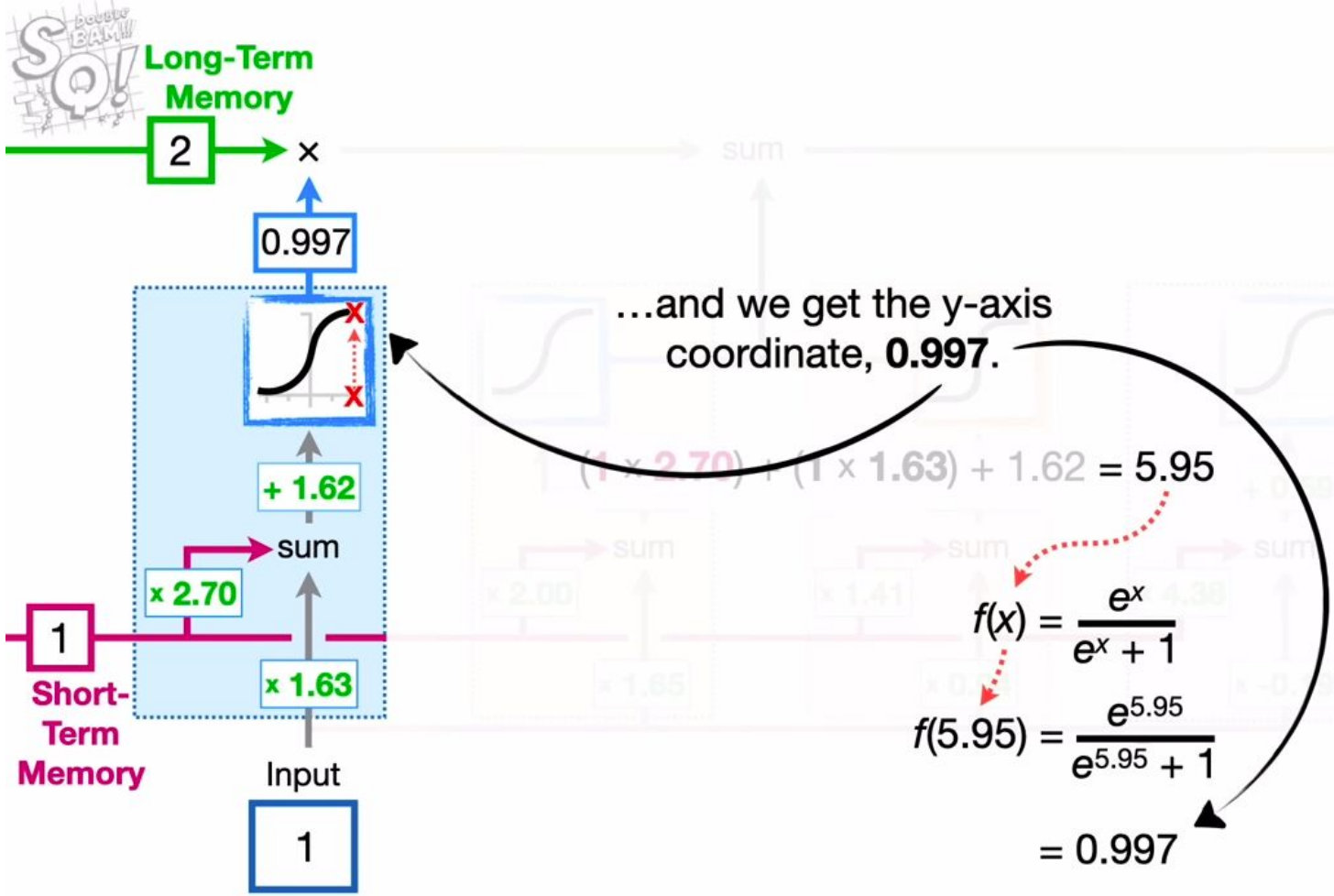
Now we plug the x-axis coordinate into the equation for the **Sigmoid Activation Function**…

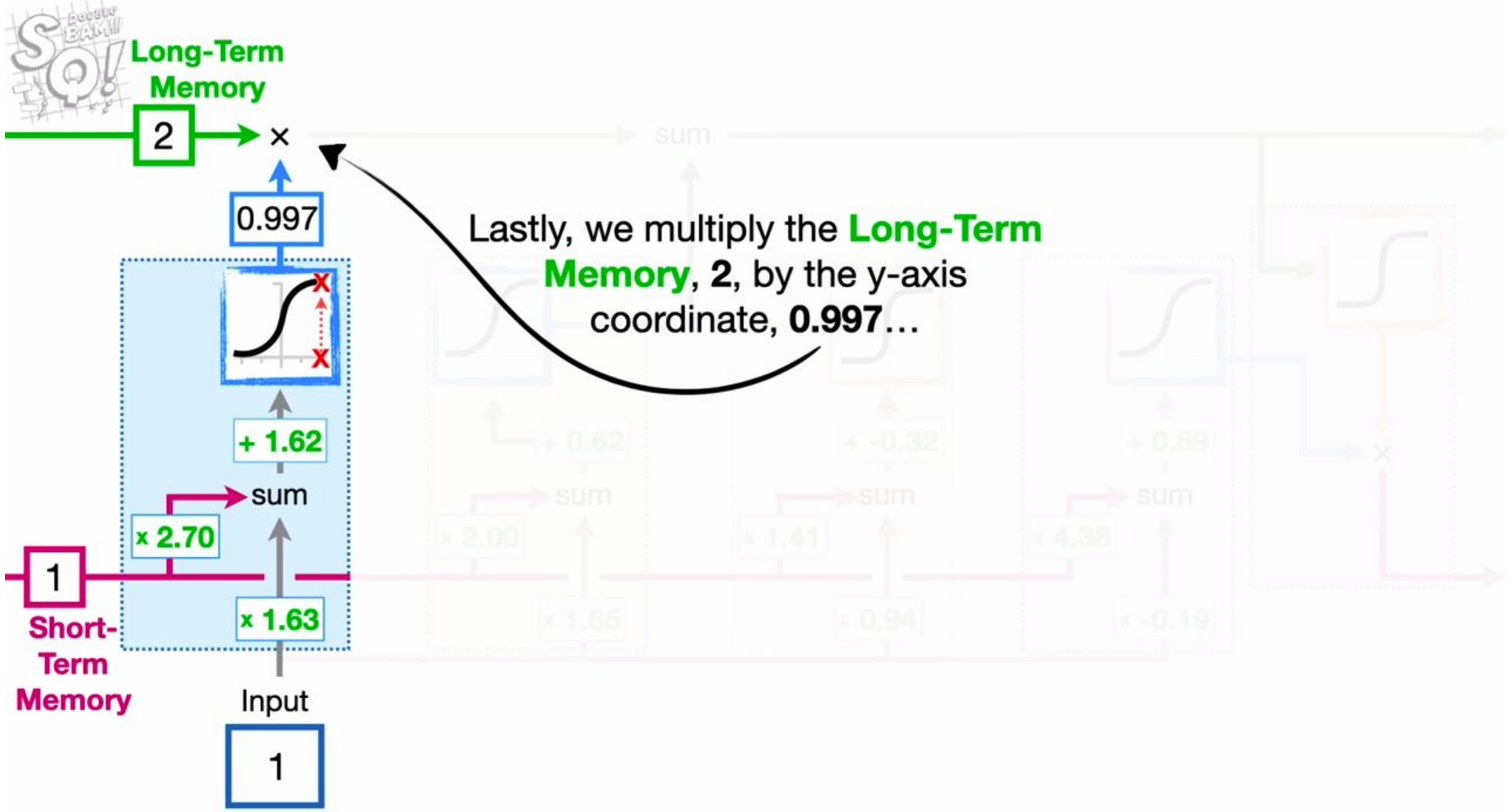
$$(1 \times 2.70) + (1 \times 1.63) + 1.62 = 5.95$$

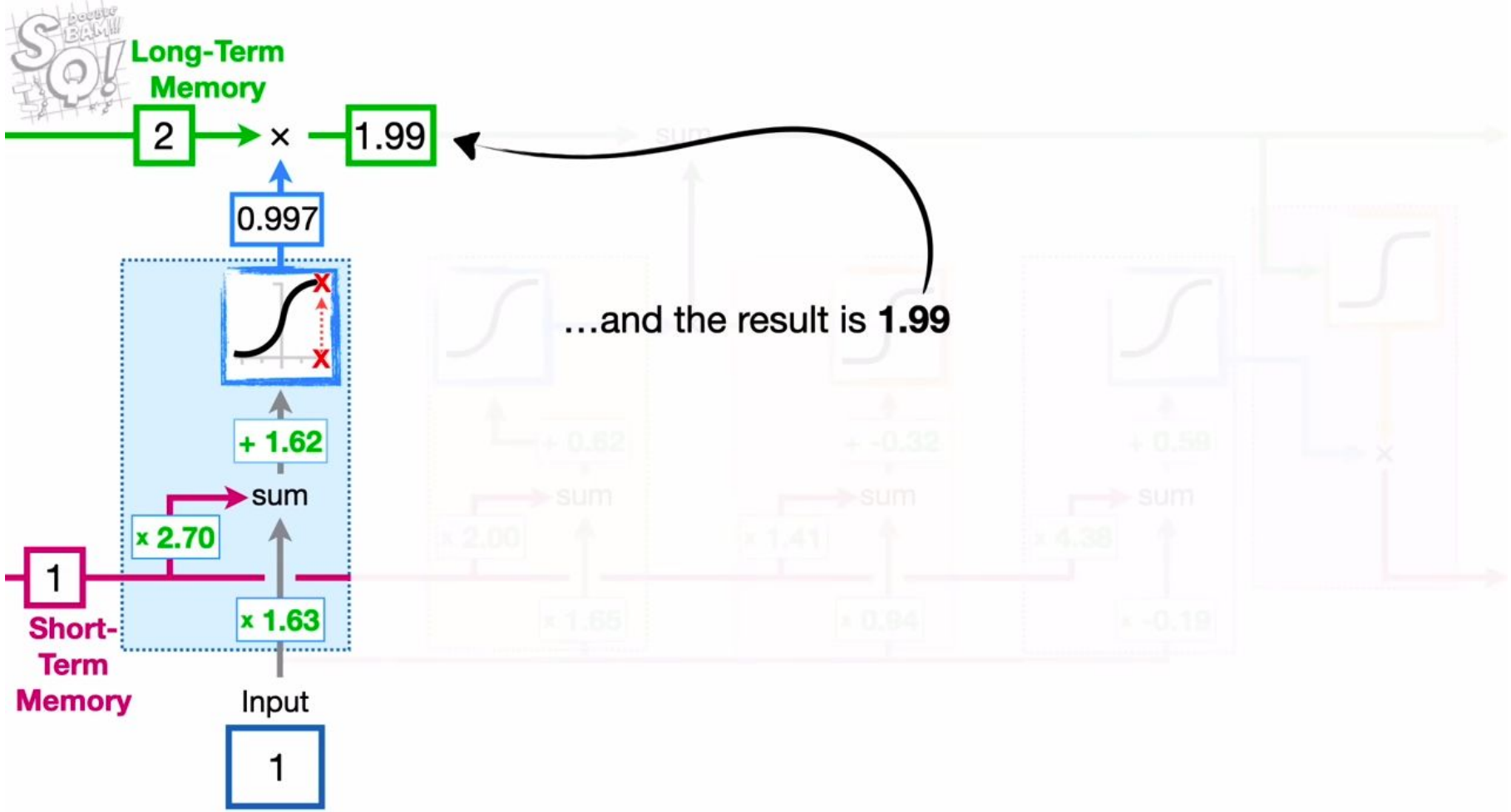
$$f(x) = \frac{e^x}{e^x + 1}$$

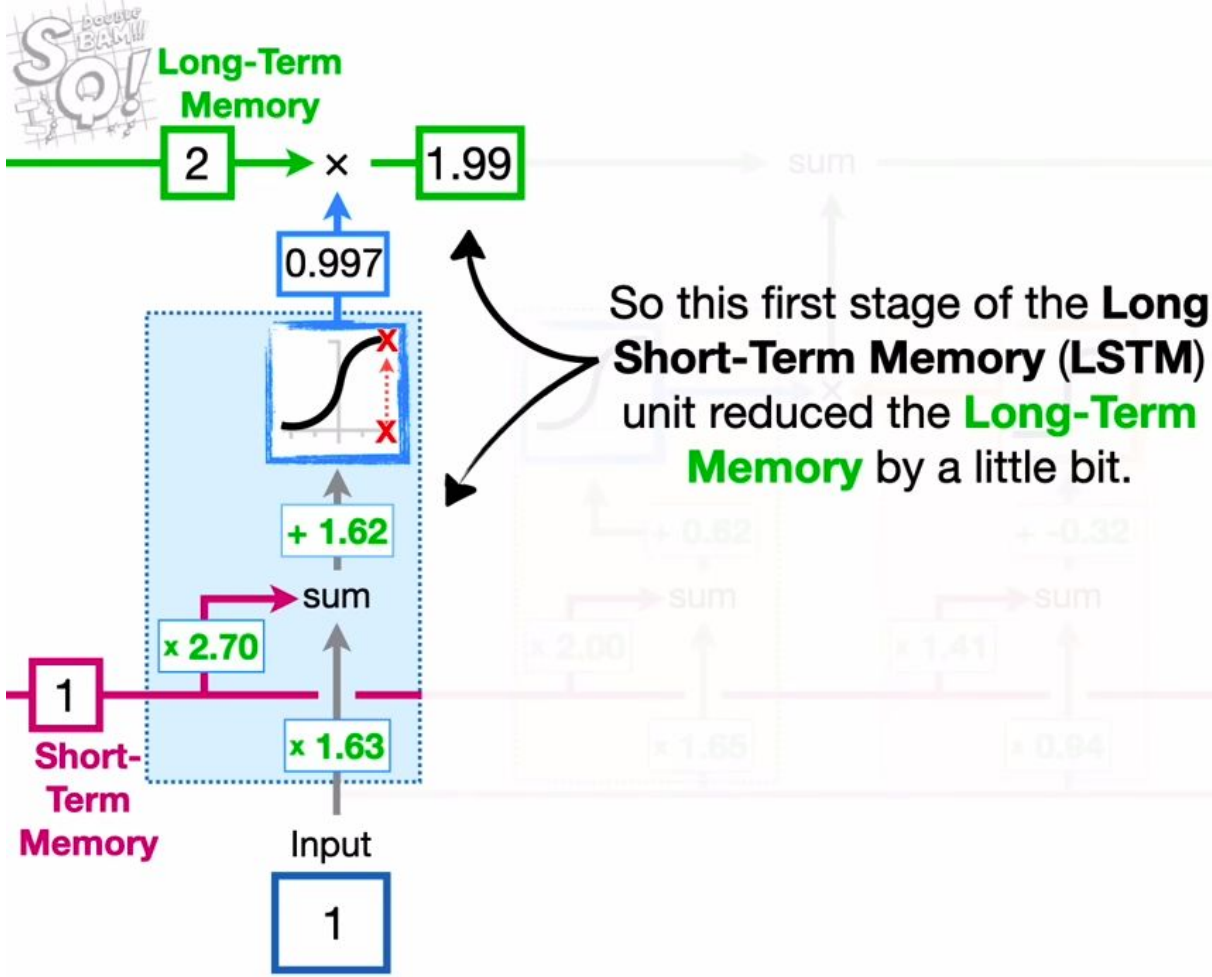
$$f(5.95) = \frac{e^{5.95}}{e^{5.95} + 1}$$

$$= 0.997$$

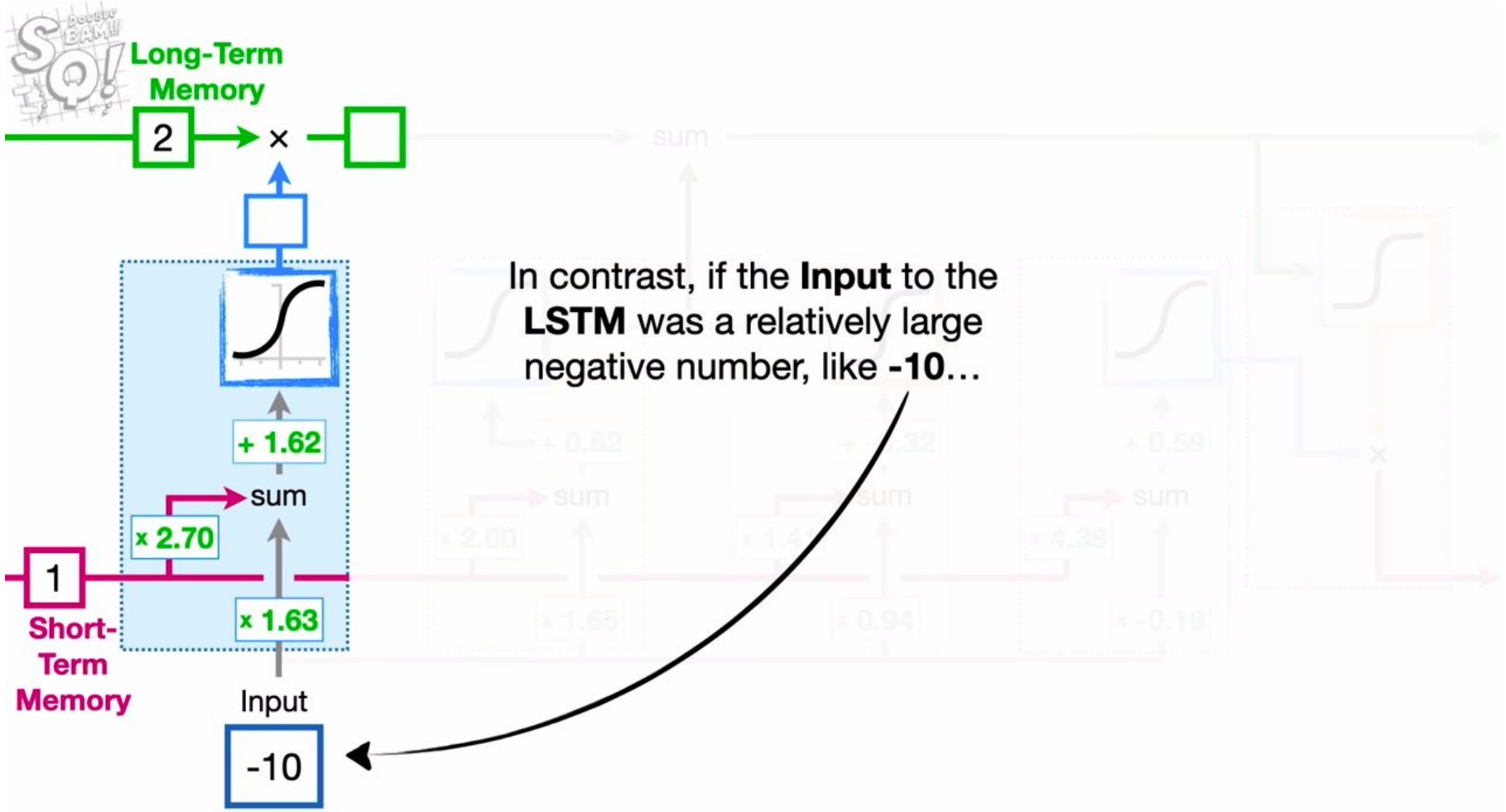

Long-Term Memory
2
×
0.997
…and we get the y-axis coordinate, 0.997.
+ 1.62
sum
× 2.70
1
Short-Term Memory
× 1.63
Input
1
$(1 \times 2.70) + (1 \times 1.63) + 1.62 = 5.95$
$f(x) = \frac{e^x}{e^x + 1}$
$f(5.95) = \frac{e^{5.95}}{e^{5.95} + 1}$
$= 0.997$

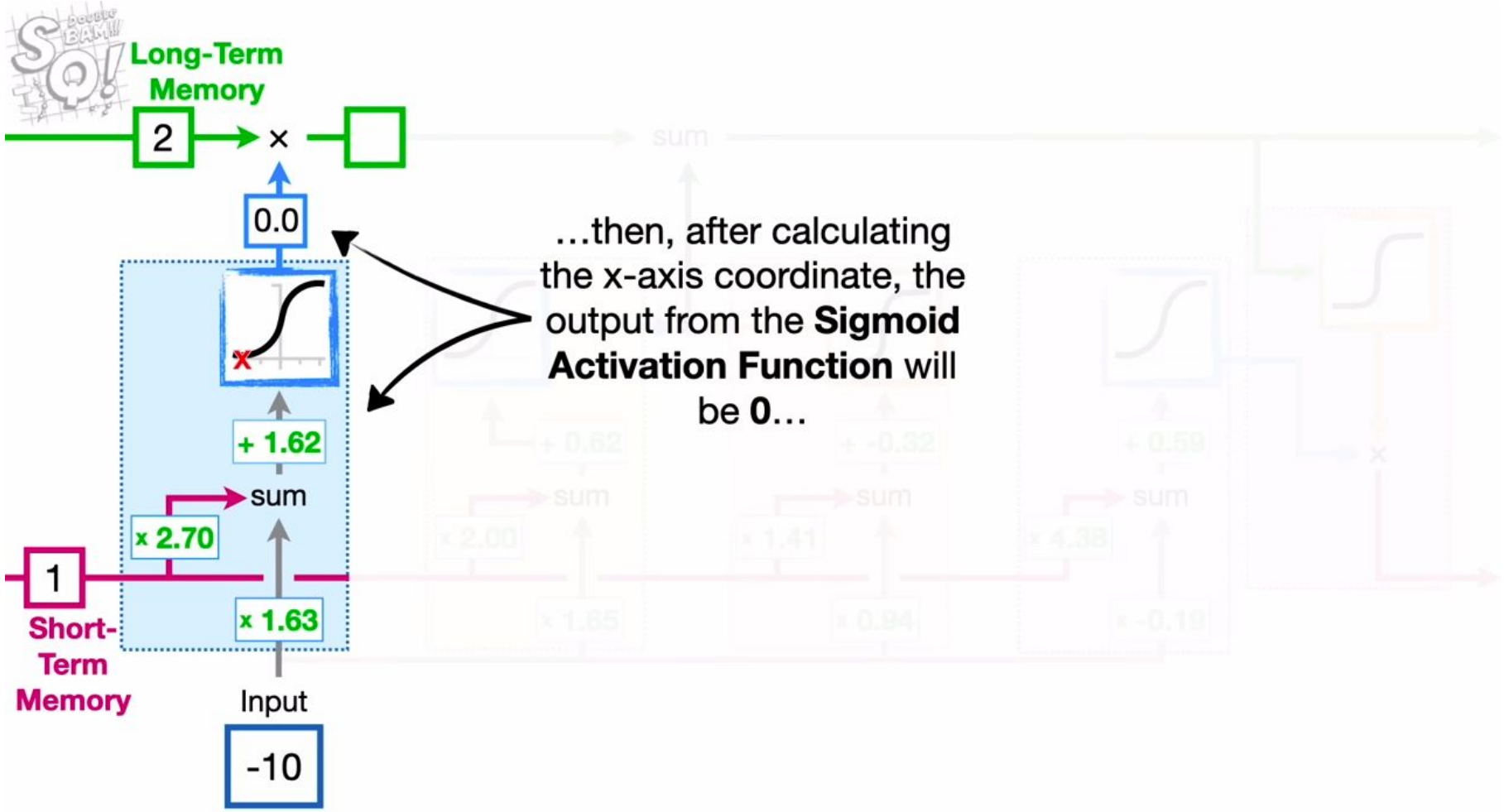

Long-Term Memory
2
×
0.997
Lastly, we multiply the Long-Term Memory, 2, by the y-axis coordinate, 0.997…
+ 1.62
sum
× 2.70
1
Short-Term Memory
× 1.63
Input
1

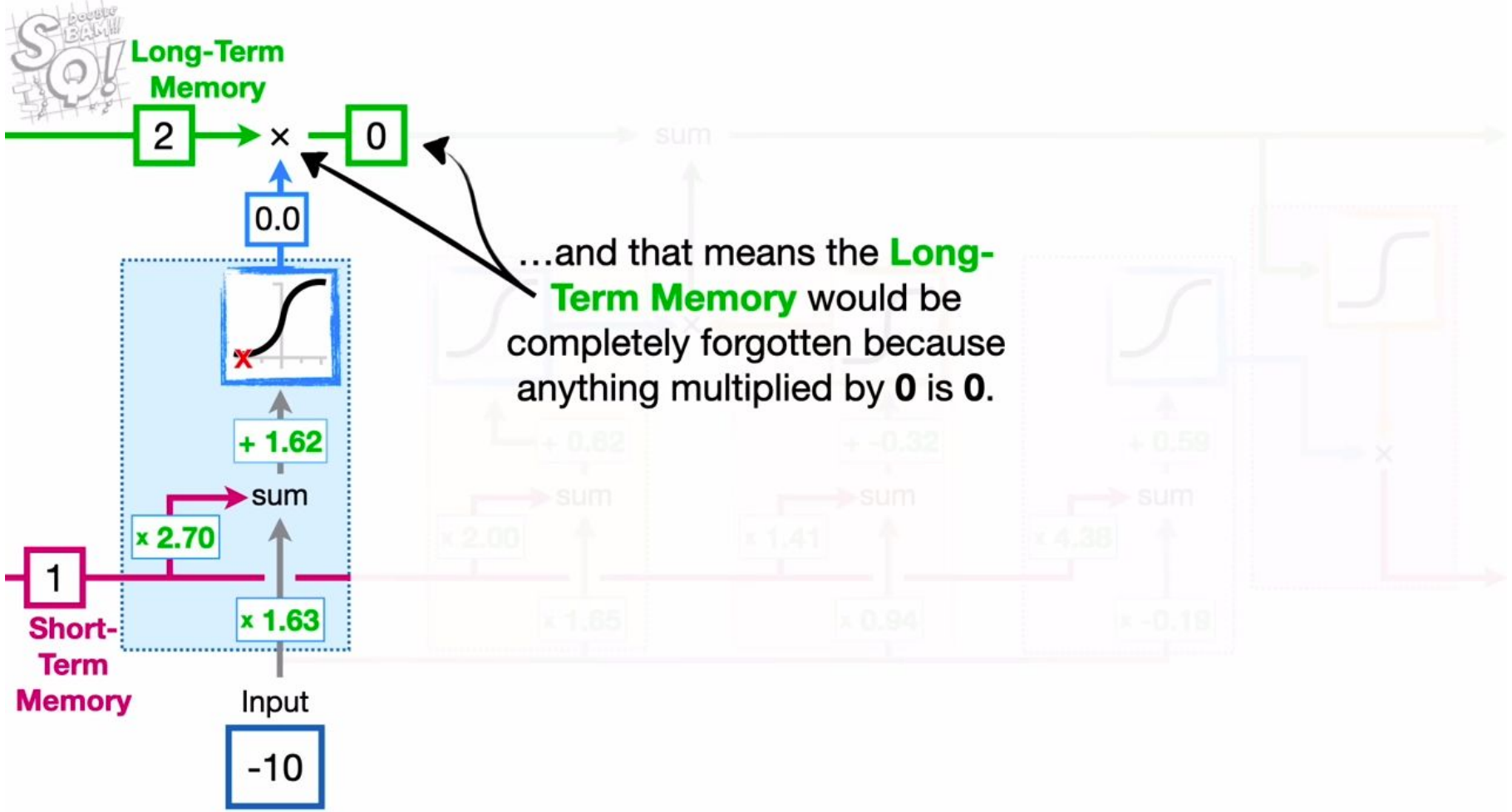

Long-Term Memory
2
×
1.99
0.997
…and the result is 1.99
+ 1.62
sum
× 2.70
1
Short-Term Memory
× 1.63
Input
1

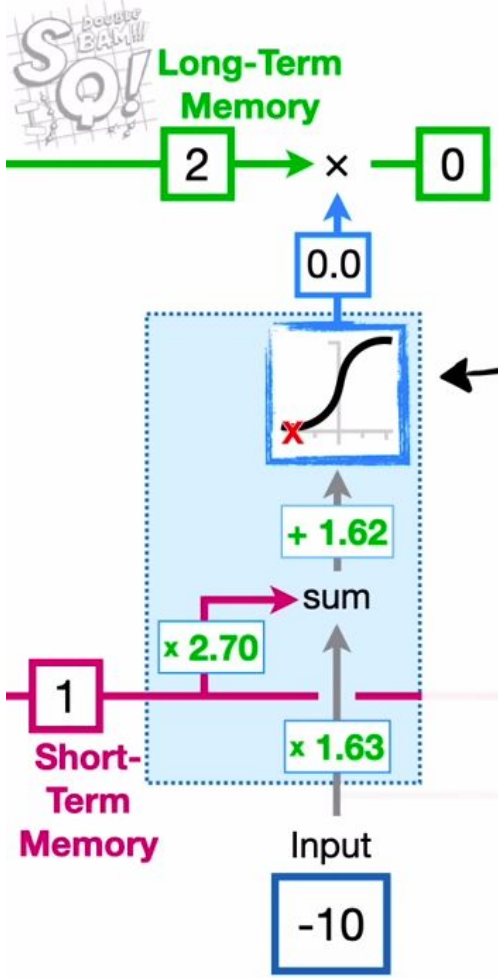


So this first stage of the **Long Short-Term Memory** (**LSTM**) unit reduced the **Long-Term Memory** by a little bit.

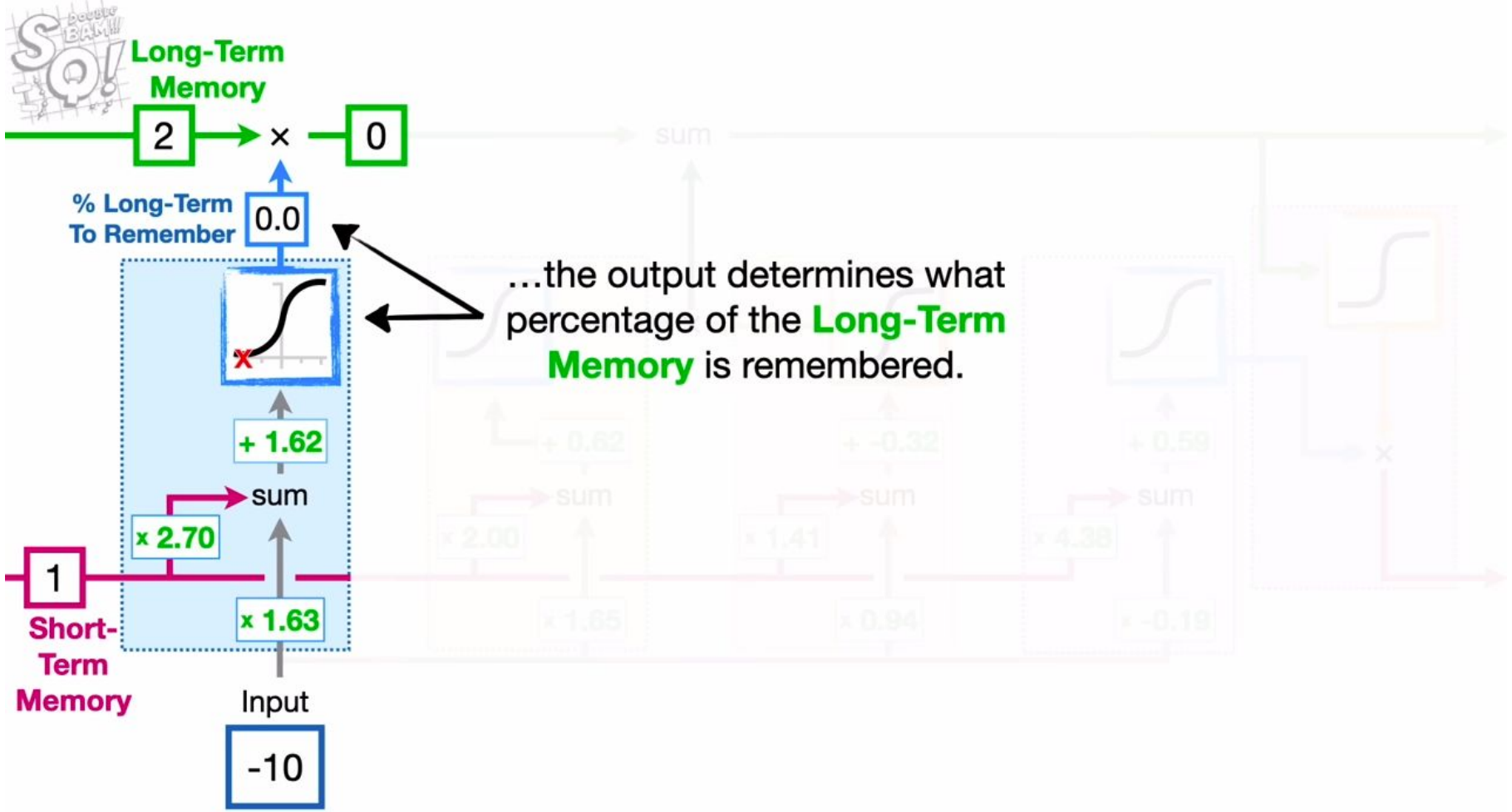

Long-Term Memory
2
×
+ 1.62
sum
× 2.70
1
Short-Term Memory
× 1.63
Input
-10
In contrast, if the **Input** to the **LSTM** was a relatively large negative number, like **-10**…

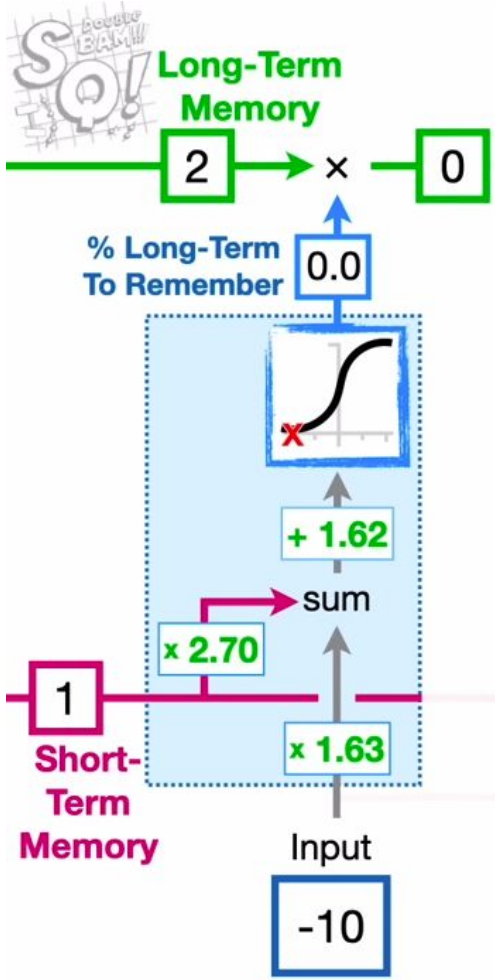

Long-Term Memory
2
×
0.0
x
+ 1.62
sum
x 2.70
1
Short-Term Memory
x 1.63
Input
-10
…then, after calculating the x-axis coordinate, the output from the **Sigmoid Activation Function** will be **0**…


Long-Term Memory
2
×
0
0.0
…and that means the Long-Term Memory would be completely forgotten because anything multiplied by 0 is 0.
+ 1.62
sum
x 2.70
1
Short-Term Memory
x 1.63
Input
-10

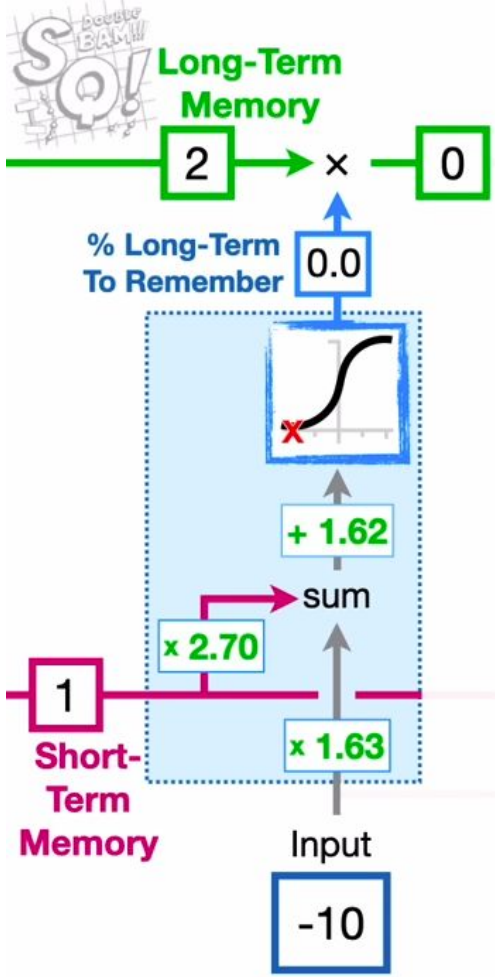


Thus, because the **Sigmoid Activation Function** turns any input into a number between **0** and **1**...

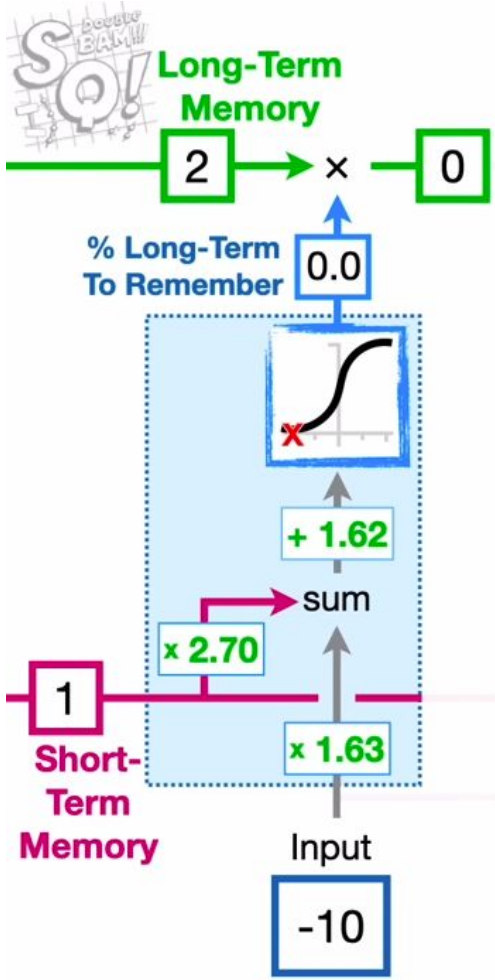

Long-Term Memory
2
×
0
% Long-Term To Remember
0.0
…the output determines what percentage of the Long-Term Memory is remembered.
+ 1.62
sum
x 2.70
1
Short-Term Memory
x 1.63
Input
-10



To summarize, the first stage in a **Long Short-Term Memory** unit determines what percentage of the **Long-Term Memory** is remembered.

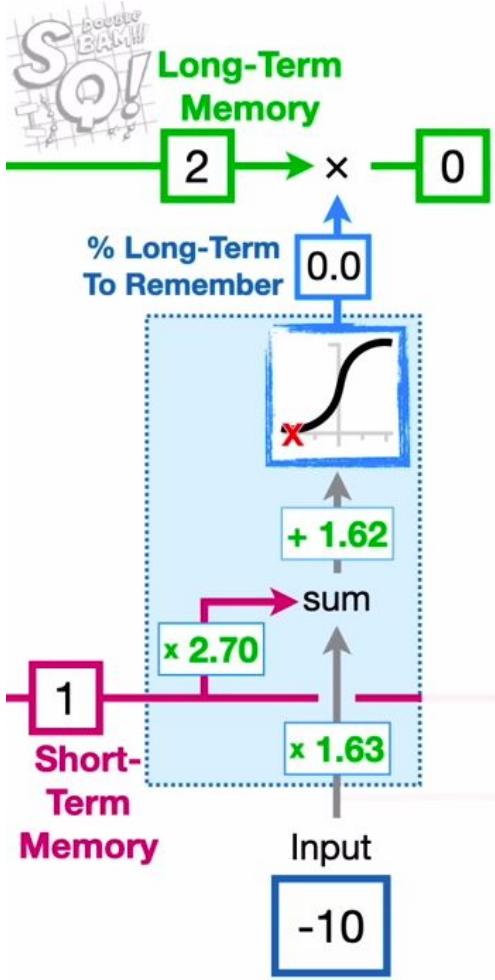


TERMINOLOGY ALERT!!!



TERMINOLOGY ALERT!!!

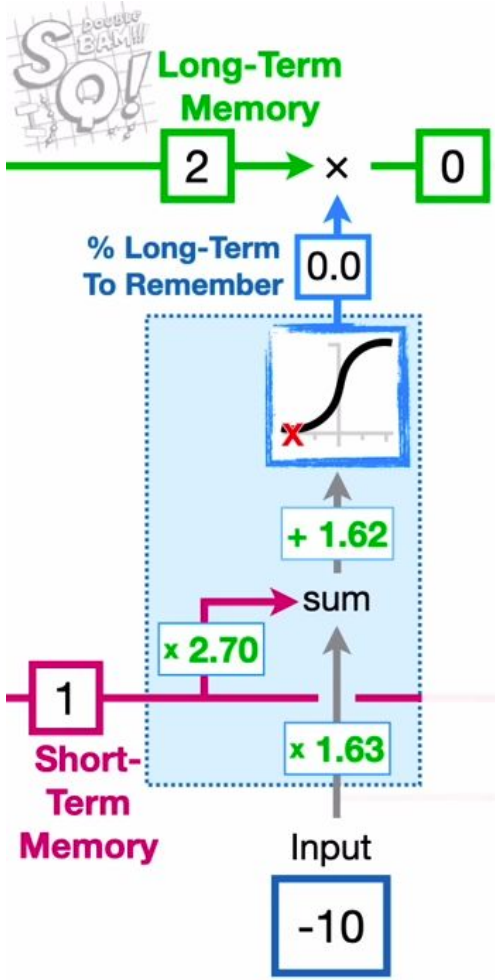
Even though this part of the **Long Short-Term Memory** unit determines what percentage of the **Long-Term Memory** will be remembered…



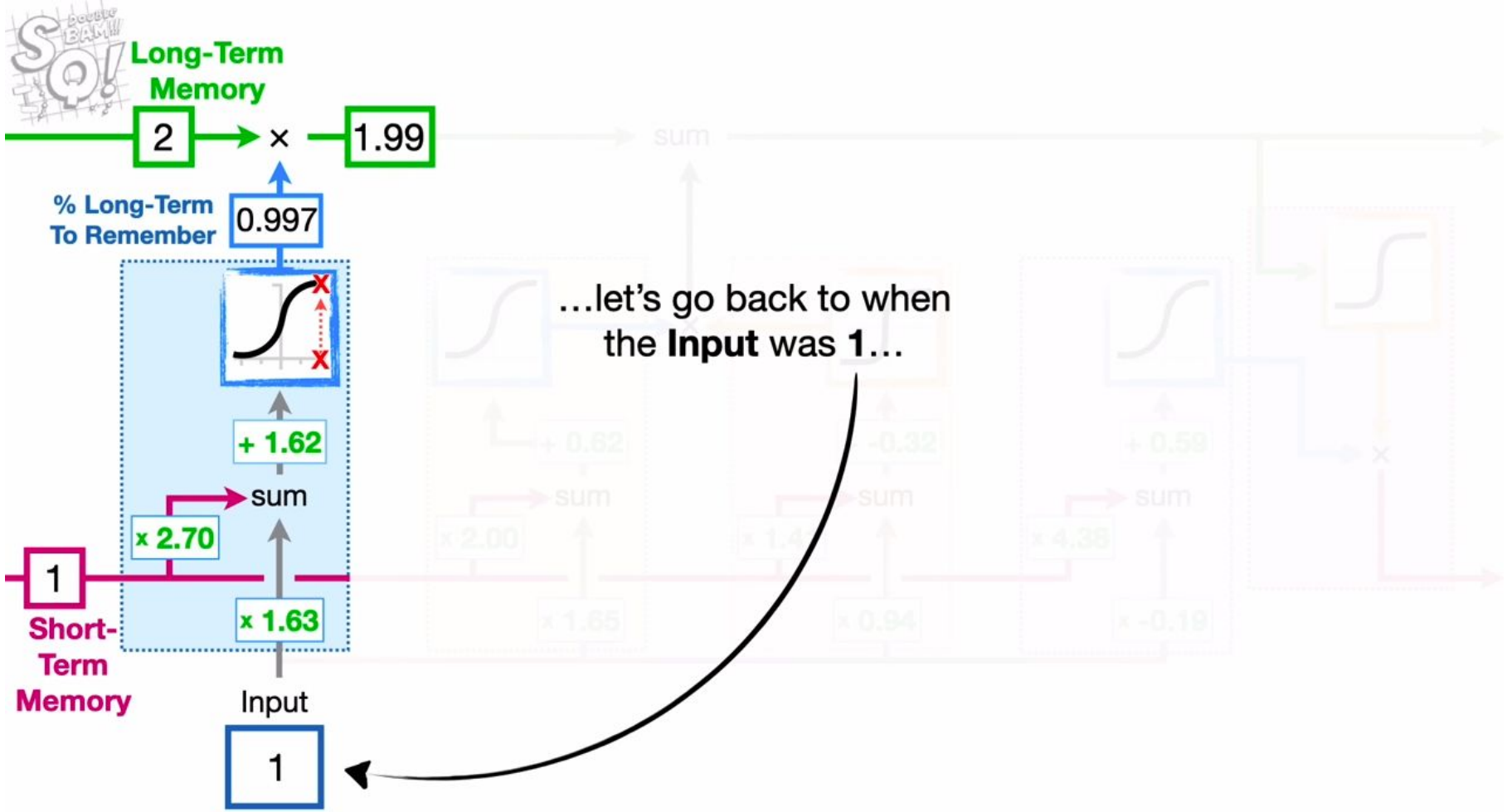
TERMINOLOGY ALERT!!!

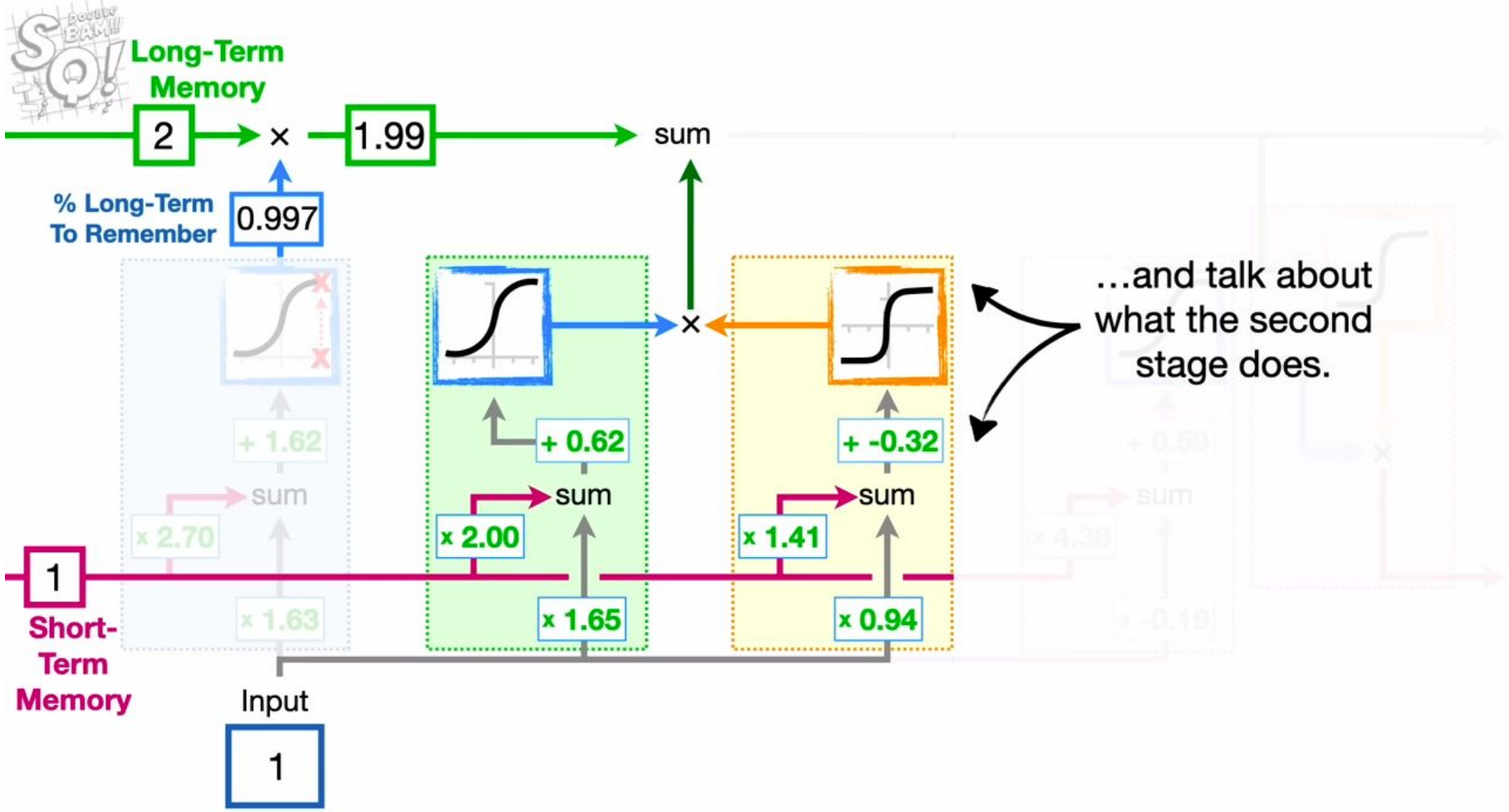
Even though this part of the **Long Short-Term Memory** unit determines what percentage of the **Long-Term Memory** will be remembered…

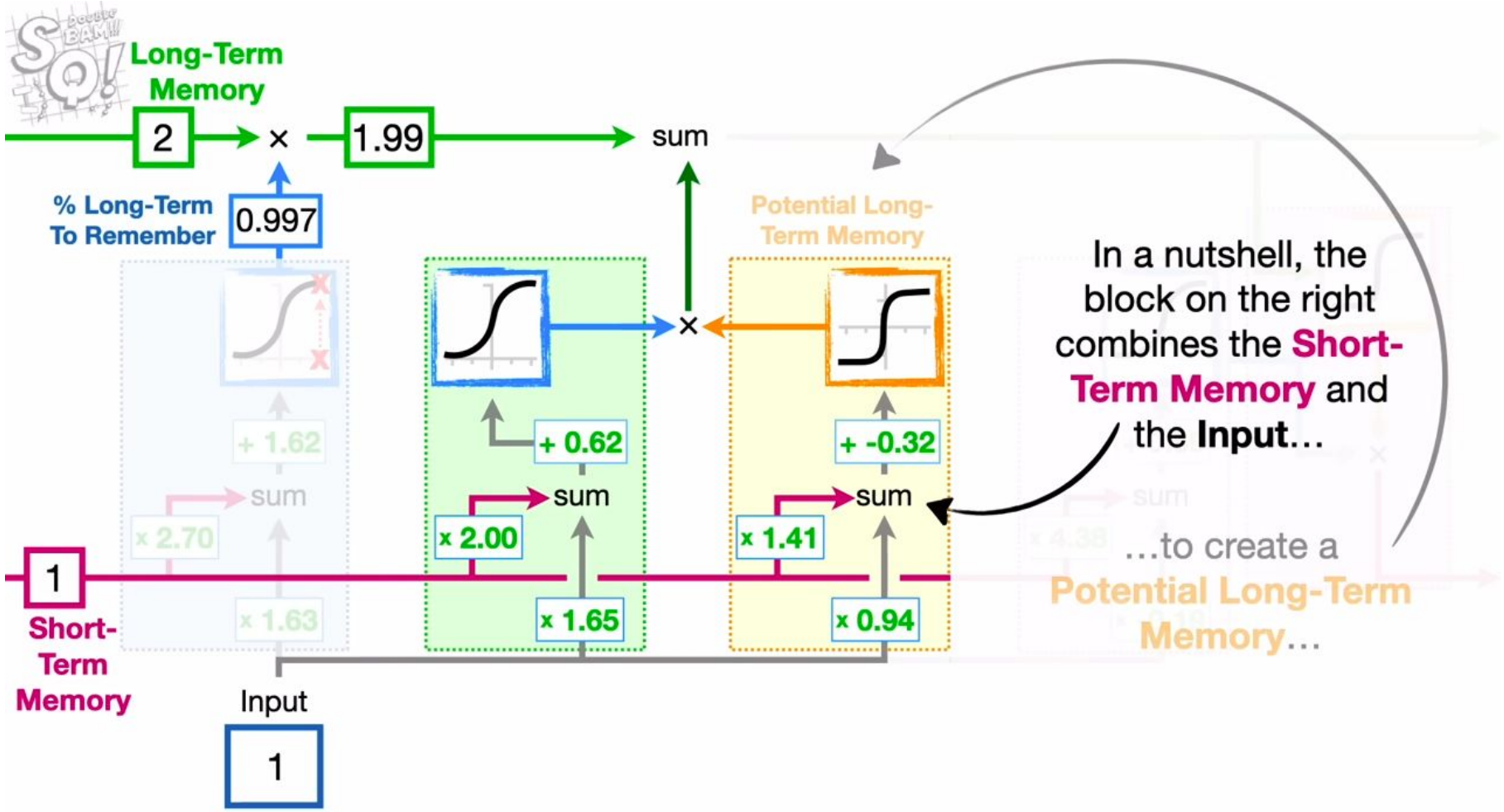
…it is usually called the **Forget Gate**.

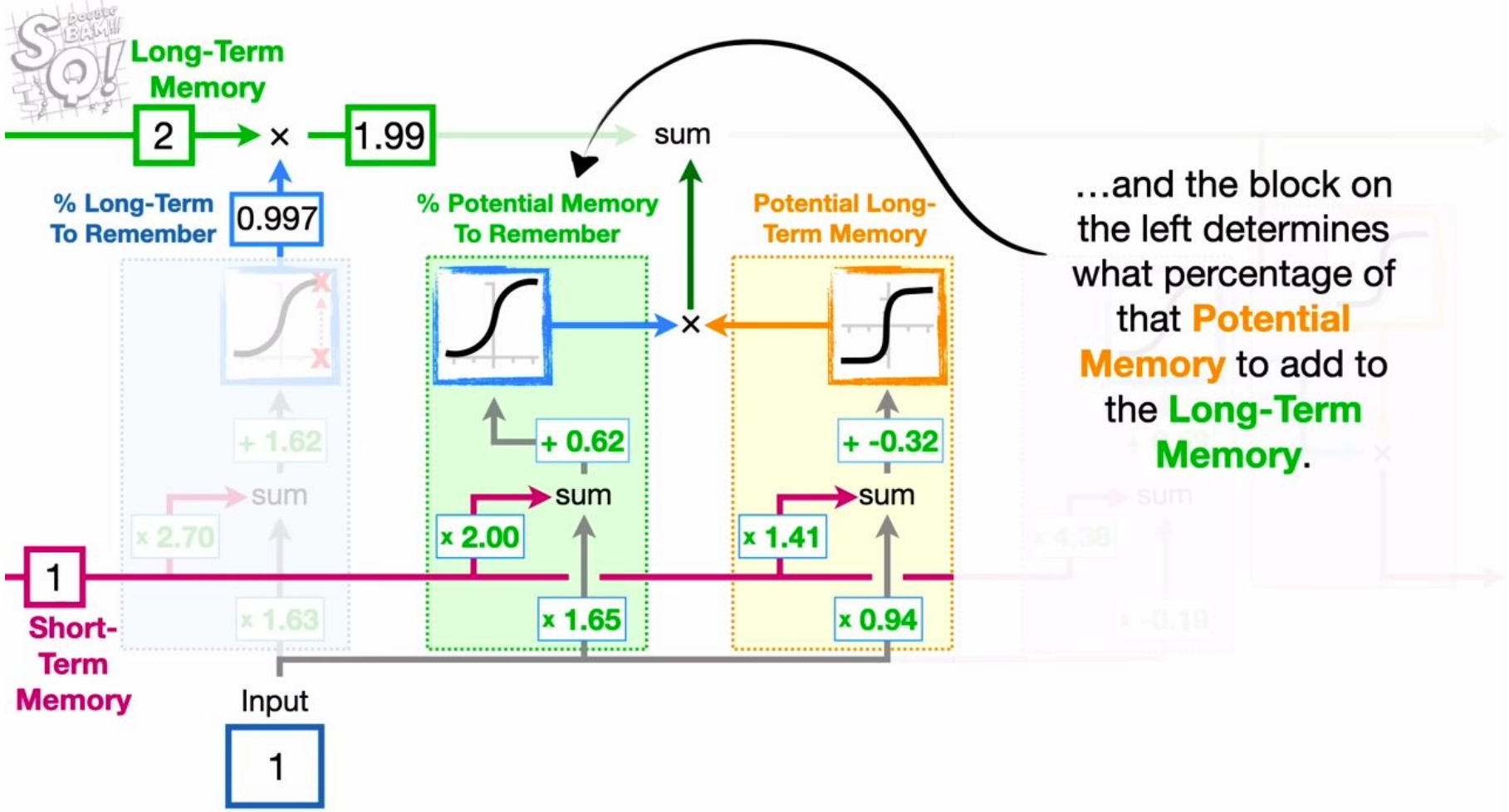


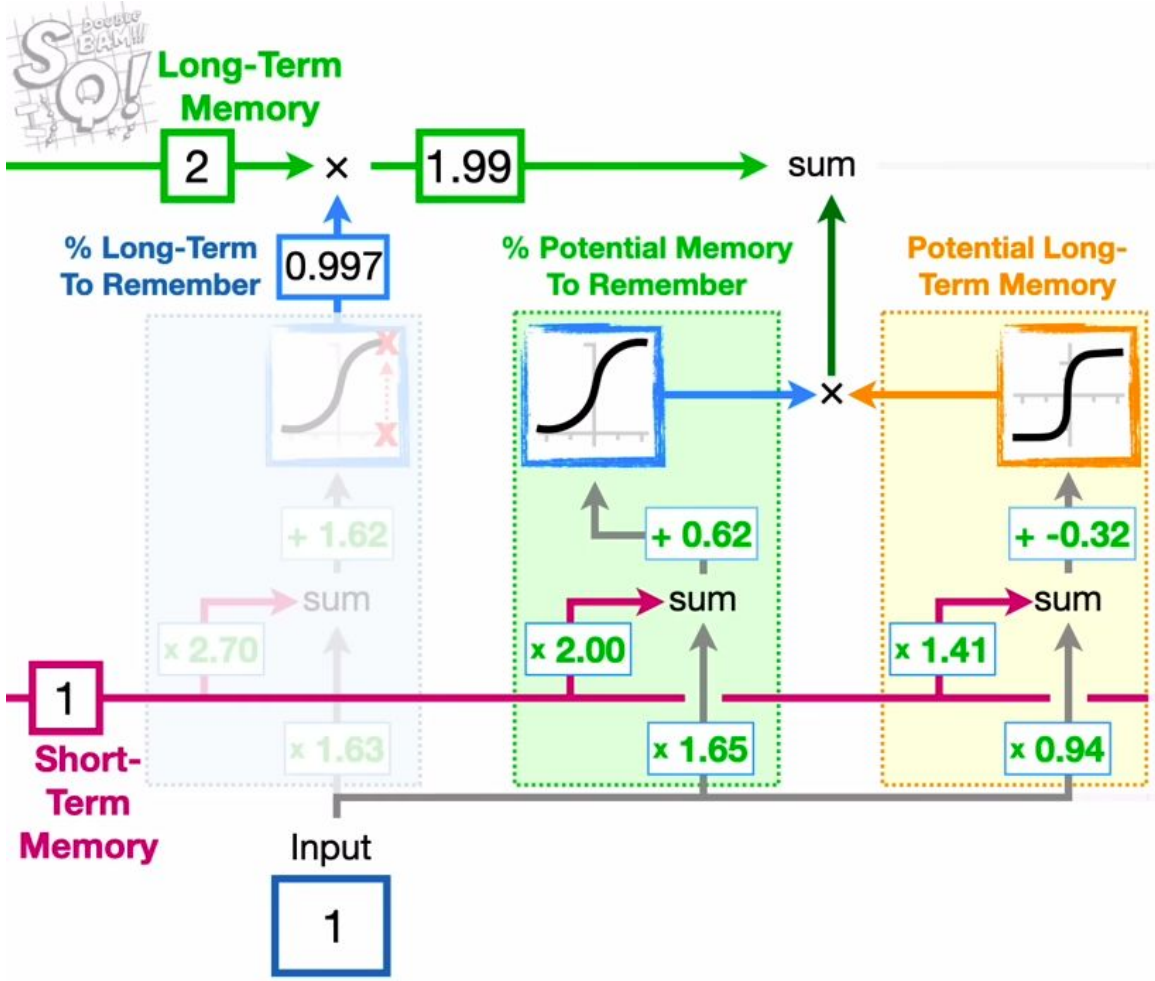
Now that we know what the first part of the **LSTM** unit does, it determines what percentage of the **Long-Term Memory** will be remembered…


Long-Term Memory
2
×
1.99
% Long-Term To Remember
0.997
+ 1.62
sum
× 2.70
1
Short-Term Memory
× 1.63
Input
1
…let's go back to when the **Input** was **1**…

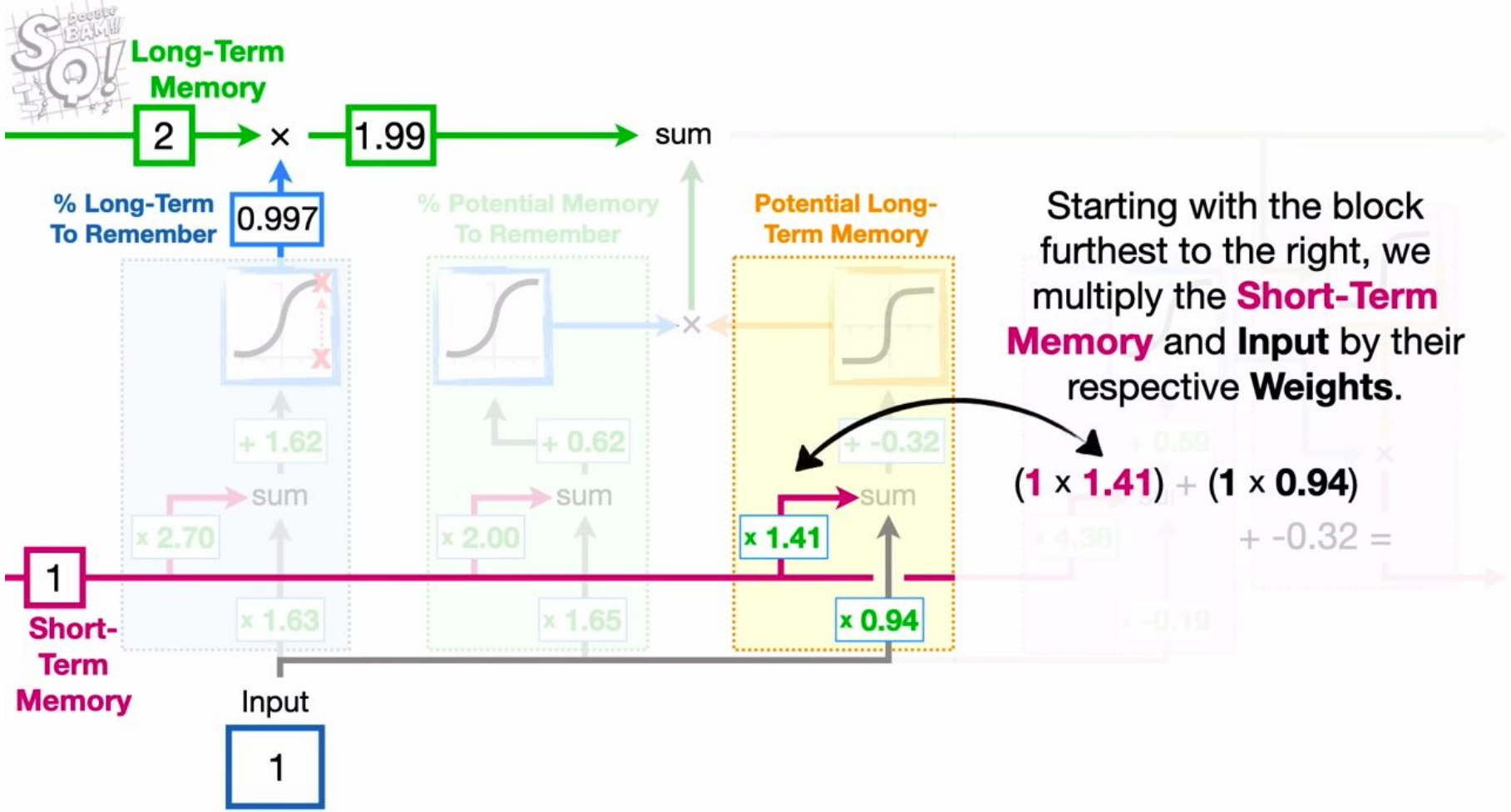

Long-Term Memory
2
×
1.99
sum
% Long-Term To Remember
0.997
+ 0.62
sum
× 2.00
× 1.65
+ -0.32
sum
× 1.41
× 0.94
×
…and talk about what the second stage does.
1
Short-Term Memory
Input
1

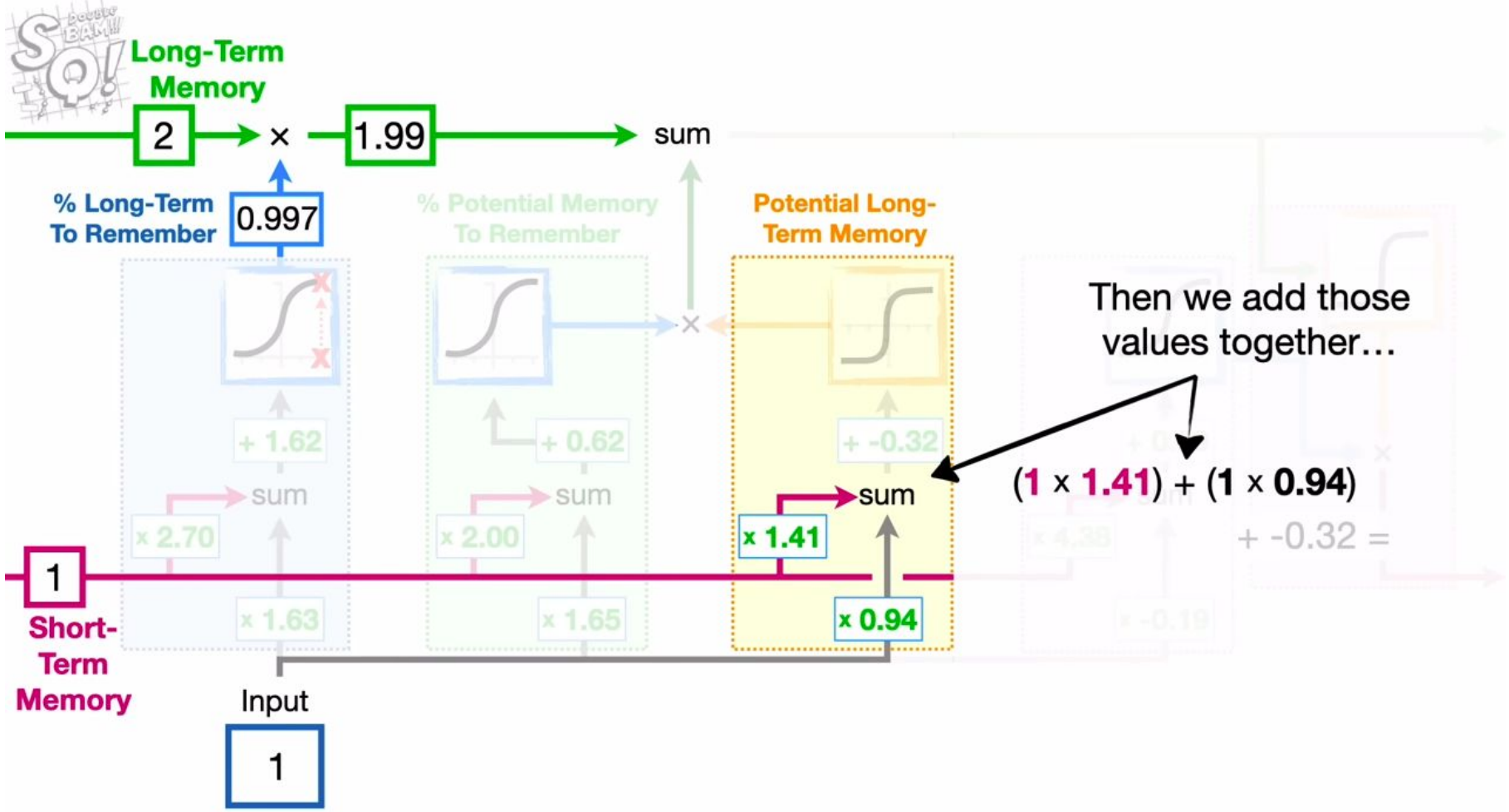

Long-Term Memory
2
×
1.99
sum
% Long-Term To Remember
0.997
Potential Long-Term Memory
In a nutshell, the block on the right combines the Short-Term Memory and the Input…
+ 0.62
+ -0.32
sum
sum
× 2.00
× 1.41
1
× 1.65
× 0.94
Short-Term Memory
Input
1
…to create a Potential Long-Term Memory…

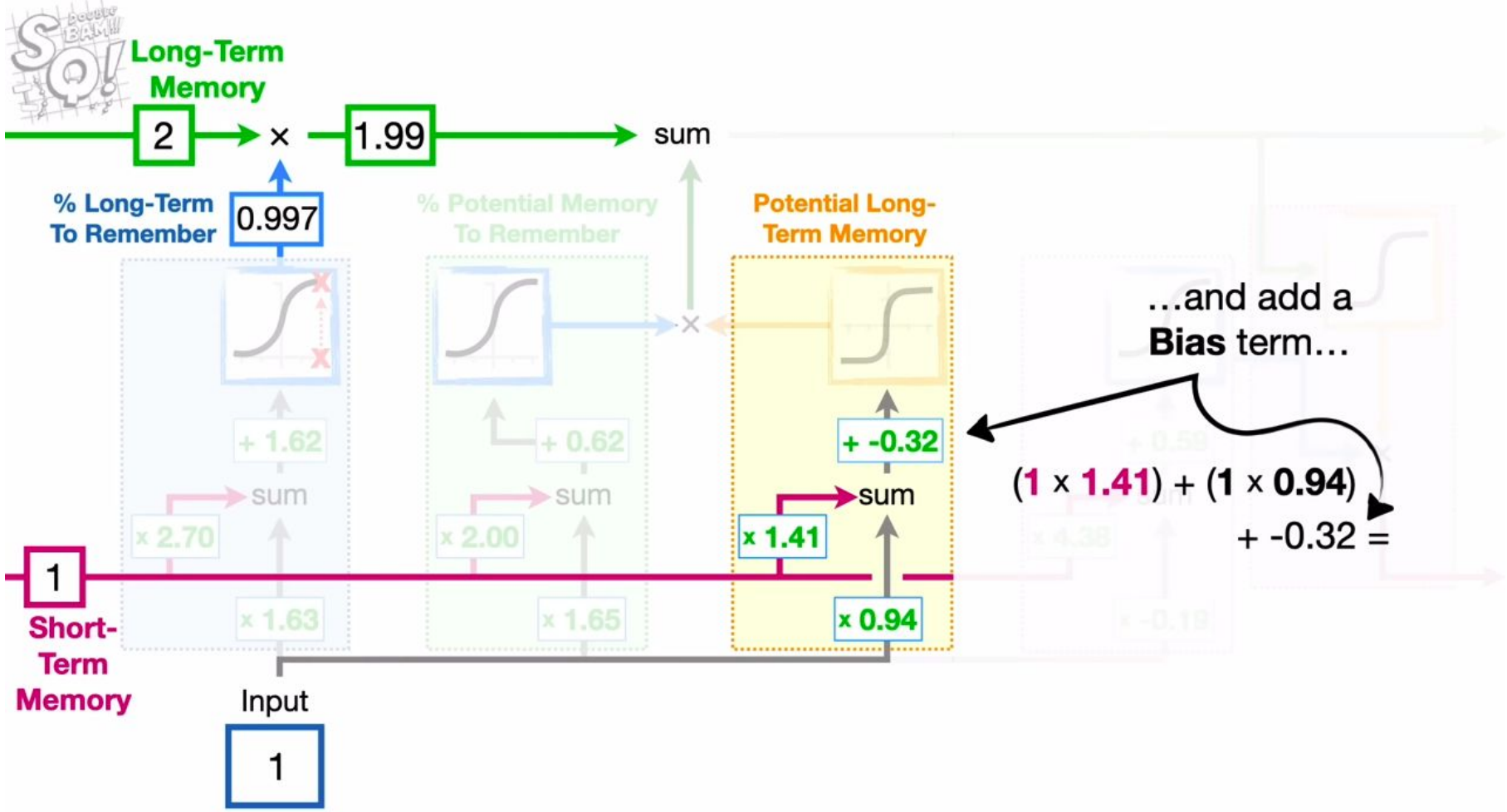

Long-Term Memory
2
×
1.99
sum
% Long-Term To Remember
0.997
% Potential Memory To Remember
Potential Long-Term Memory
×
+ 0.62
+ -0.32
sum
sum
× 2.00
× 1.41
1
Short-Term Memory
× 1.65
× 0.94
Input
1
…and the block on the left determines what percentage of that Potential Memory to add to the Long-Term Memory.

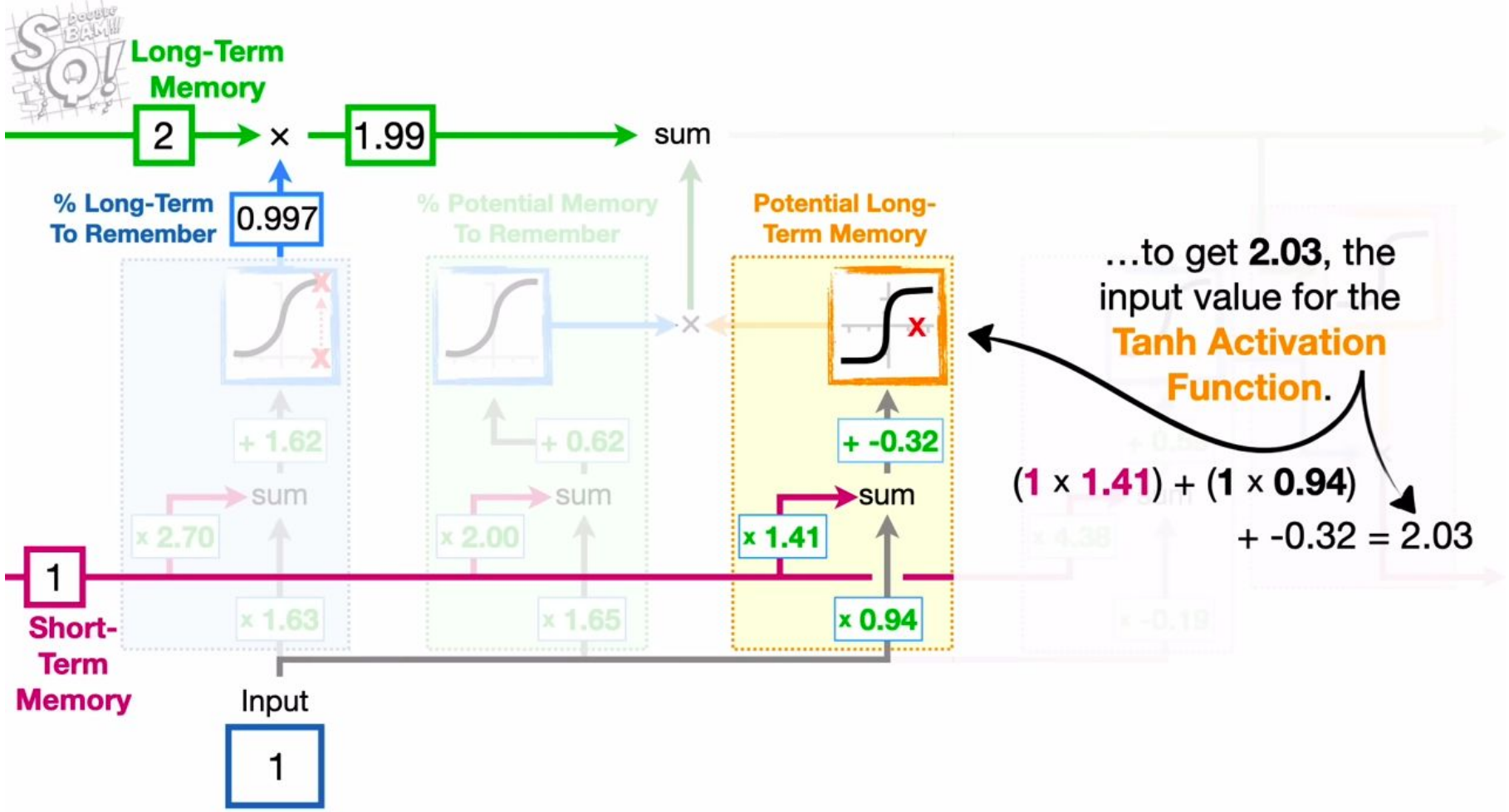


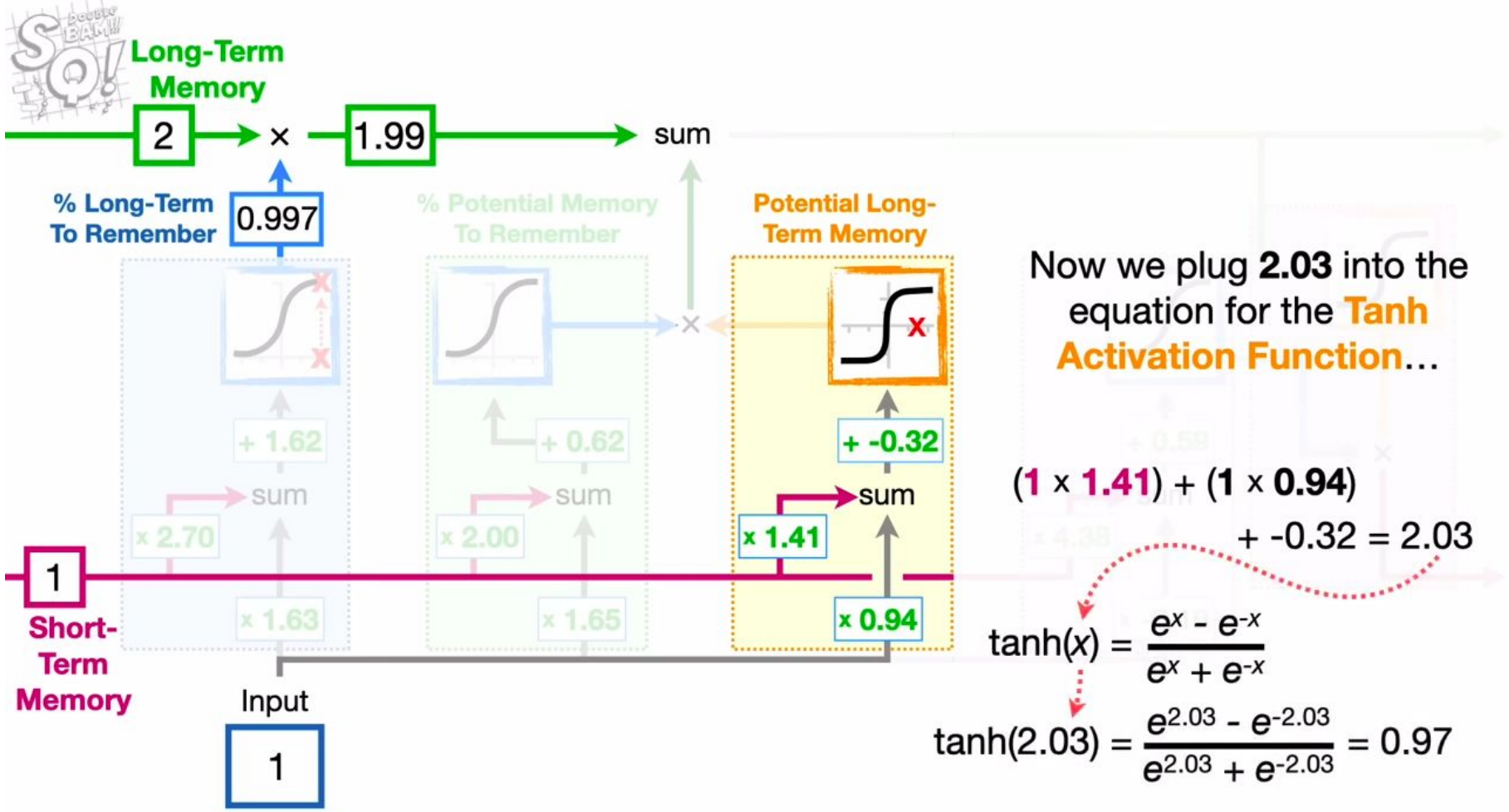
So let's plug the numbers in and do the math to see how a **Potential Memory** is created and how much of it is added to the **Long-Term Memory**.


Long-Term Memory
2
×
1.99
sum
% Long-Term To Remember
0.997
% Potential Memory To Remember
Potential Long-Term Memory
Starting with the block furthest to the right, we multiply the Short-Term Memory and Input by their respective Weights.
(1 x 1.41) + (1 x 0.94)
+ -0.32 =
+ 1.62
+ 0.62
+ -0.32
sum
sum
sum
x 2.70
x 2.00
x 1.41
1
Short-Term Memory
x 1.63
x 1.65
x 0.94
Input
1


Long-Term Memory
2
×
1.99
sum
% Long-Term To Remember
0.997
% Potential Memory To Remember
Potential Long-Term Memory
Then we add those values together…
(1 x 1.41) + (1 x 0.94)
+ 1.62
sum
x 2.70
x 1.63
+ 0.62
sum
x 2.00
x 1.65
+ -0.32
sum
x 1.41
x 0.94
1
Short-Term Memory
Input
1


Long-Term Memory
2
×
1.99
sum
% Long-Term To Remember
0.997
% Potential Memory To Remember
Potential Long-Term Memory
…and add a Bias term…
+ 1.62
sum
× 2.70
× 1.63
+ 0.62
sum
× 2.00
× 1.65
+ -0.32
sum
× 1.41
× 0.94
(1 x 1.41) + (1 x 0.94) + -0.32 =
1
Short-Term Memory
Input
1


Long-Term Memory
2
×
1.99
sum
% Long-Term To Remember
0.997
% Potential Memory To Remember
Potential Long-Term Memory
…to get 2.03, the input value for the Tanh Activation Function.
(1 x 1.41) + (1 x 0.94) + -0.32 = 2.03
+ 1.62
sum
x 2.70
x 1.63
+ 0.62
sum
x 2.00
x 1.65
+ -0.32
sum
x 1.41
x 0.94
1
Short-Term Memory
Input
1

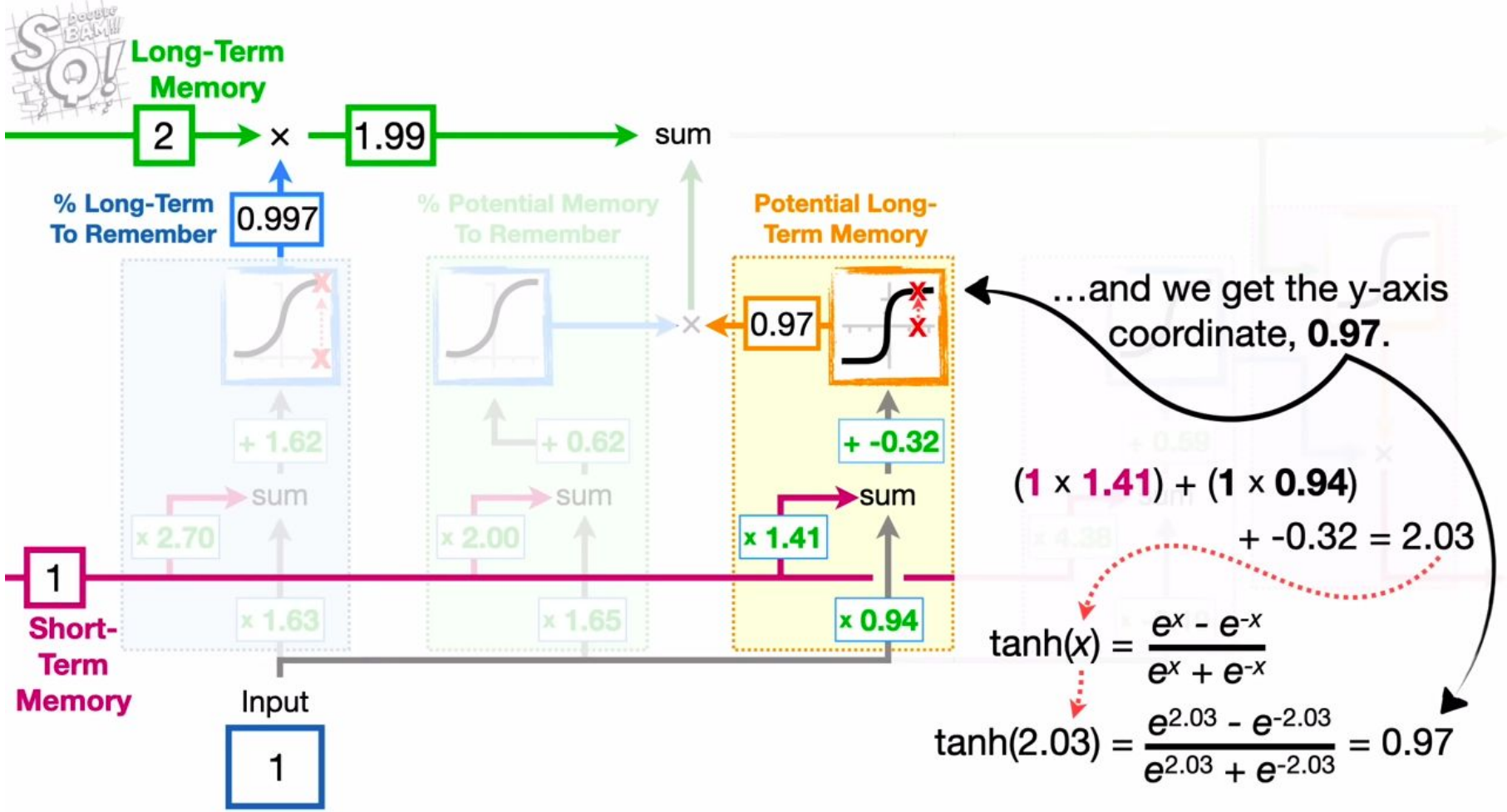


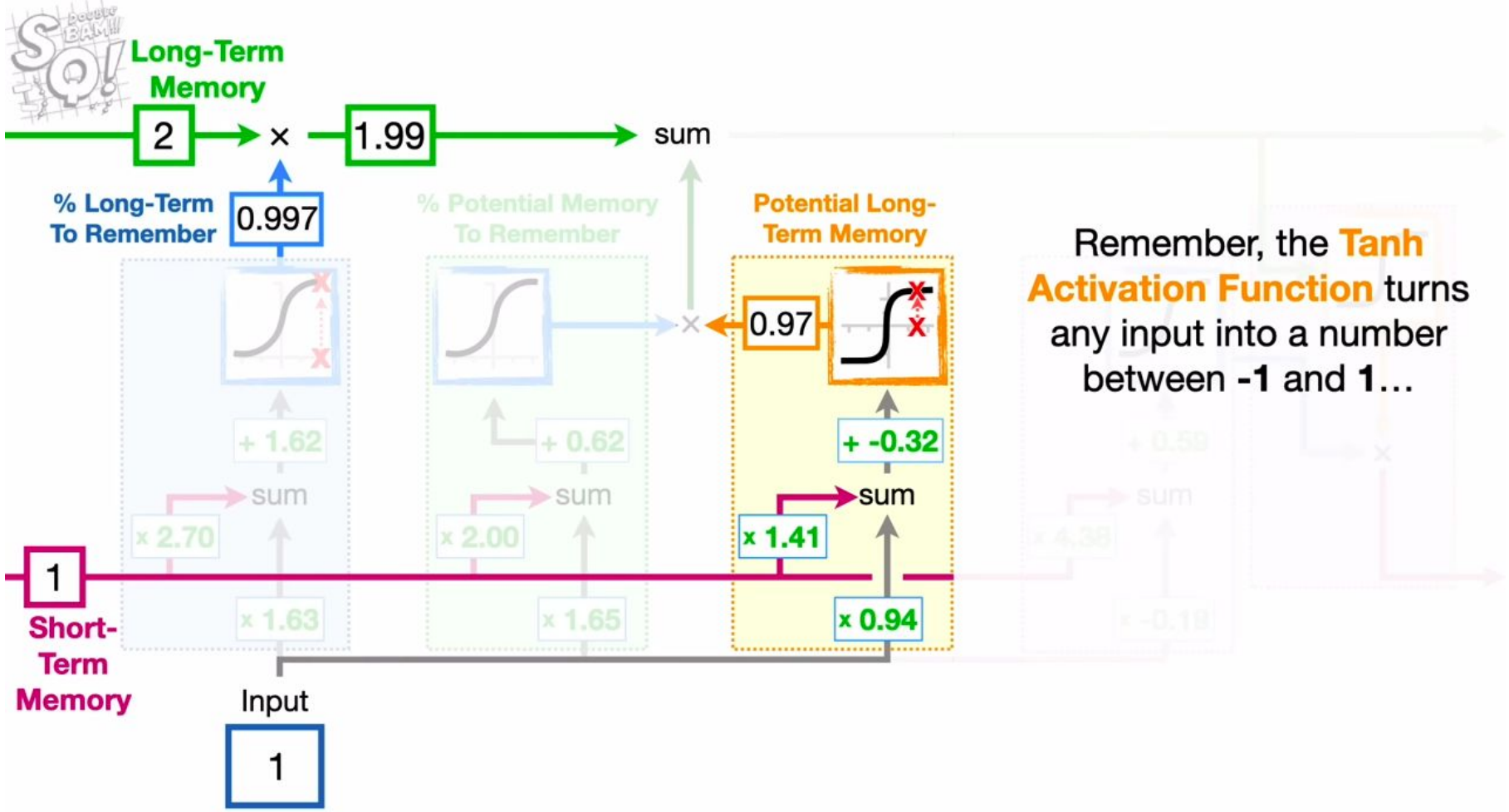
Now we plug **2.03** into the equation for the **Tanh Activation Function**…

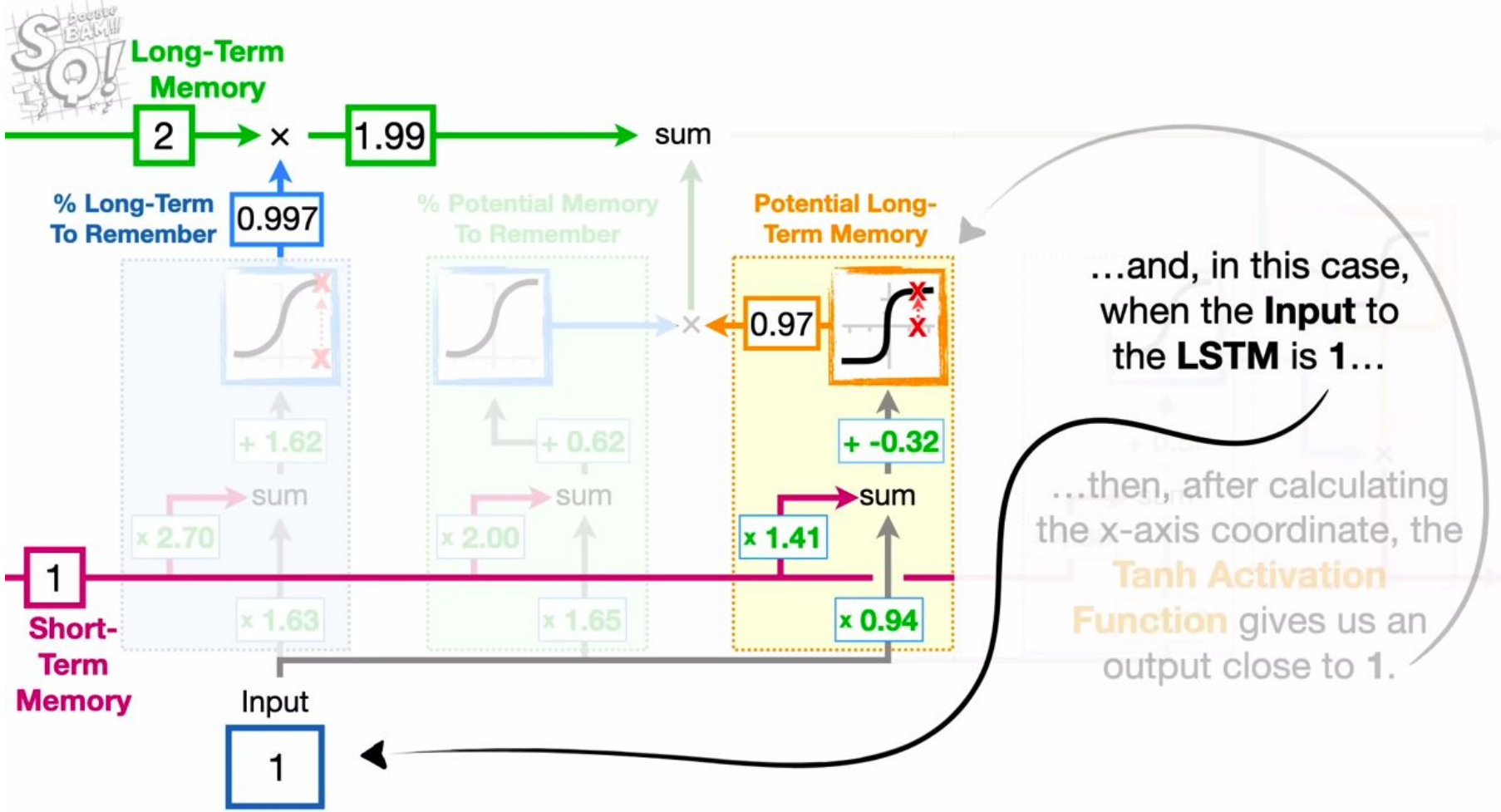
$$(1 \times 1.41) + (1 \times 0.94) + -0.32 = 2.03$$

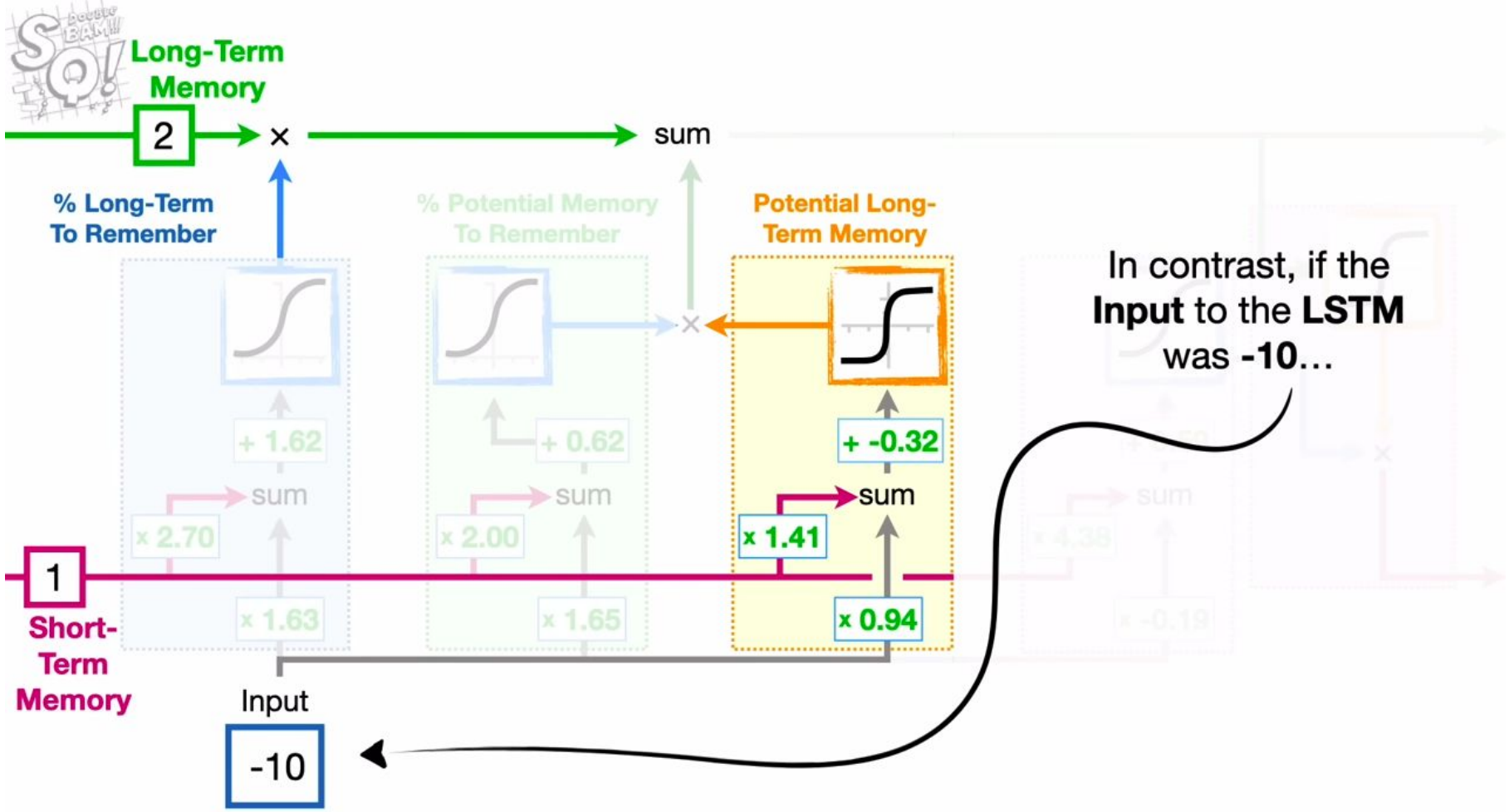
$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

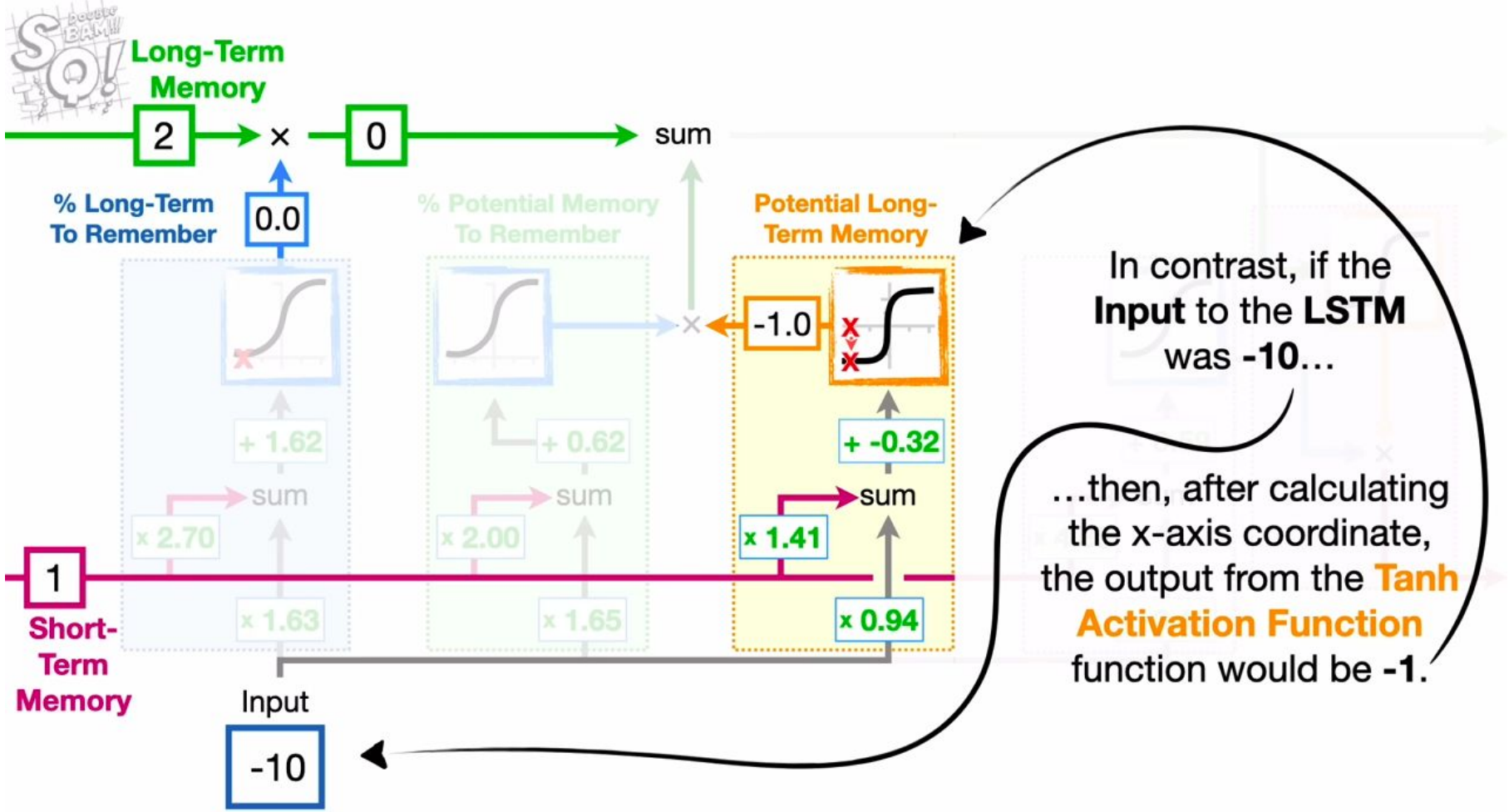
$$\tanh(2.03) = \frac{e^{2.03} - e^{-2.03}}{e^{2.03} + e^{-2.03}} = 0.97$$

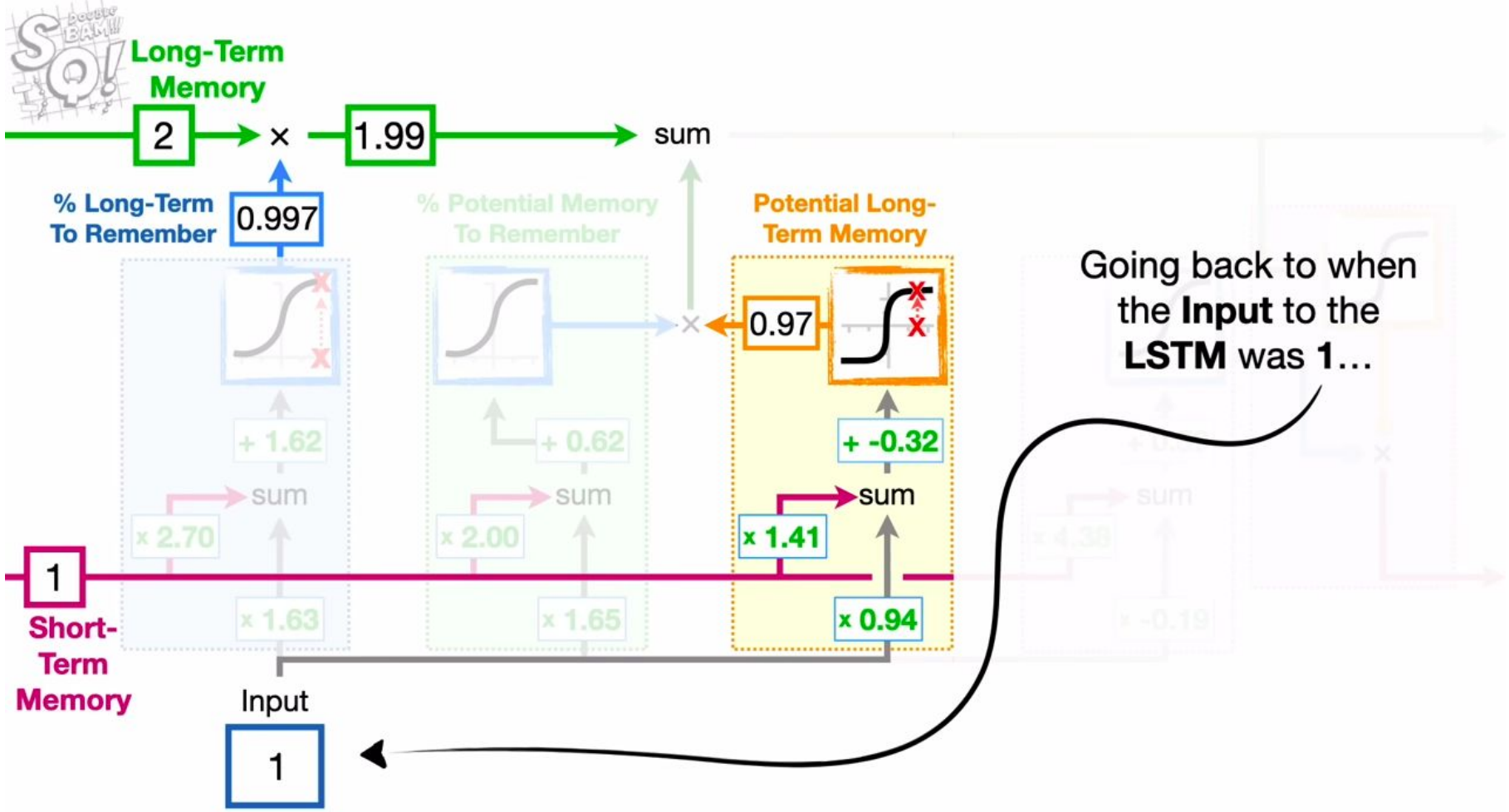

Long-Term Memory
2
×
1.99
sum
% Long-Term To Remember
0.997
% Potential Memory To Remember
Potential Long-Term Memory
0.97
…and we get the y-axis coordinate, 0.97.
+ 1.62
sum
× 2.70
× 1.63
+ 0.62
sum
× 2.00
× 1.65
+ -0.32
sum
× 1.41
× 0.94
1
Short-Term Memory
Input
1
(1 x 1.41) + (1 x 0.94)
+ -0.32 = 2.03
$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$
$\tanh(2.03) = \frac{e^{2.03} - e^{-2.03}}{e^{2.03} + e^{-2.03}} = 0.97$

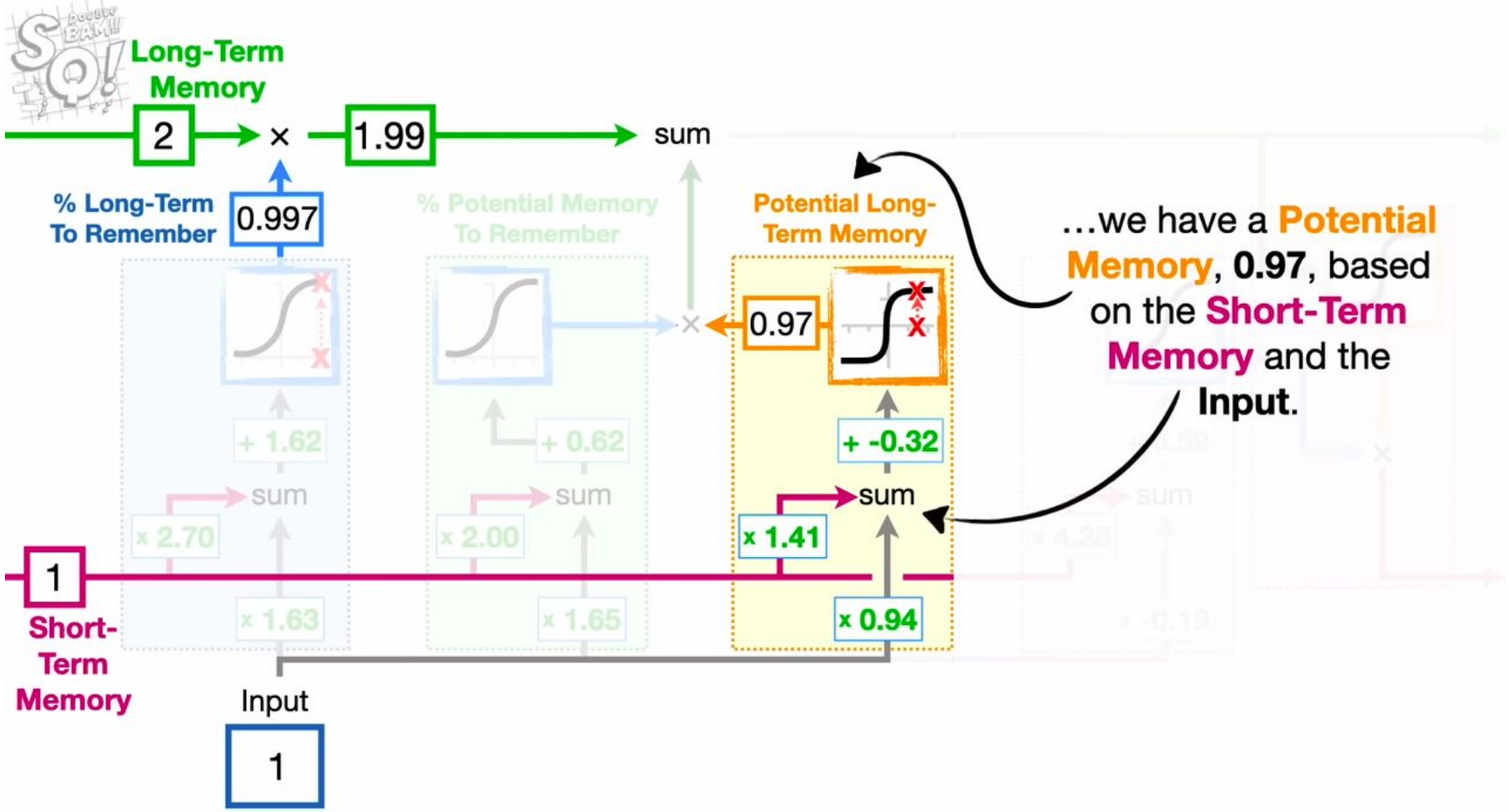

Long-Term Memory
2
×
1.99
sum
% Long-Term To Remember
0.997
% Potential Memory To Remember
Potential Long-Term Memory
0.97
+ 1.62
+ 0.62
+ -0.32
sum
sum
sum
× 2.70
× 2.00
× 1.41
1
× 1.63
× 1.65
× 0.94
Short-Term Memory
Input
1
Remember, the Tanh Activation Function turns any input into a number between -1 and 1…

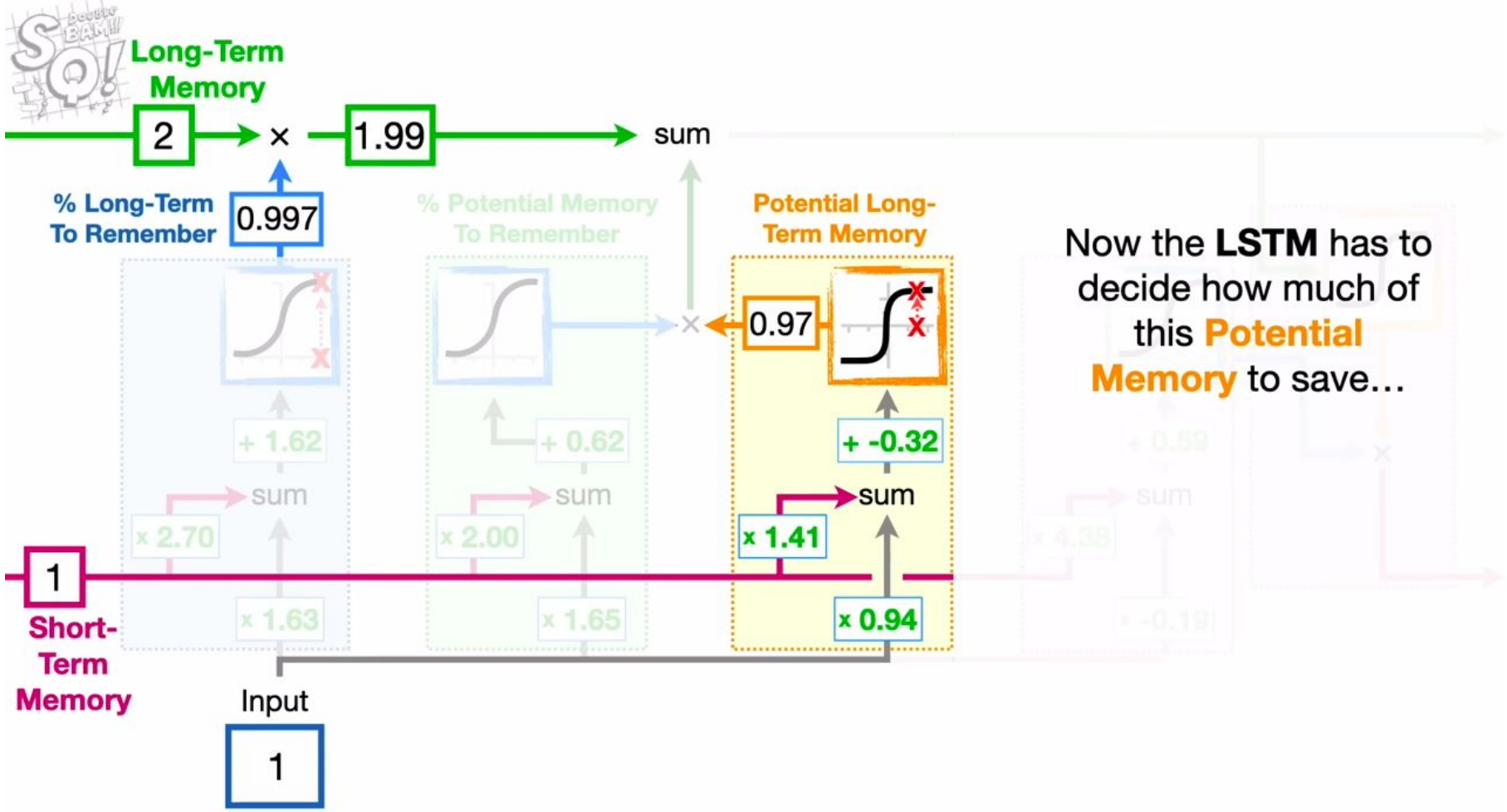

Long-Term Memory
2
×
1.99
sum
% Long-Term To Remember
0.997
% Potential Memory To Remember
Potential Long-Term Memory
0.97
+ -0.32
sum
× 1.41
× 0.94
+ 1.62
sum
× 2.70
× 1.63
+ 0.62
sum
× 2.00
× 1.65
1
Short-Term Memory
Input
1
…and, in this case, when the Input to the LSTM is 1…
…then, after calculating the x-axis coordinate, the Tanh Activation Function gives us an output close to 1.

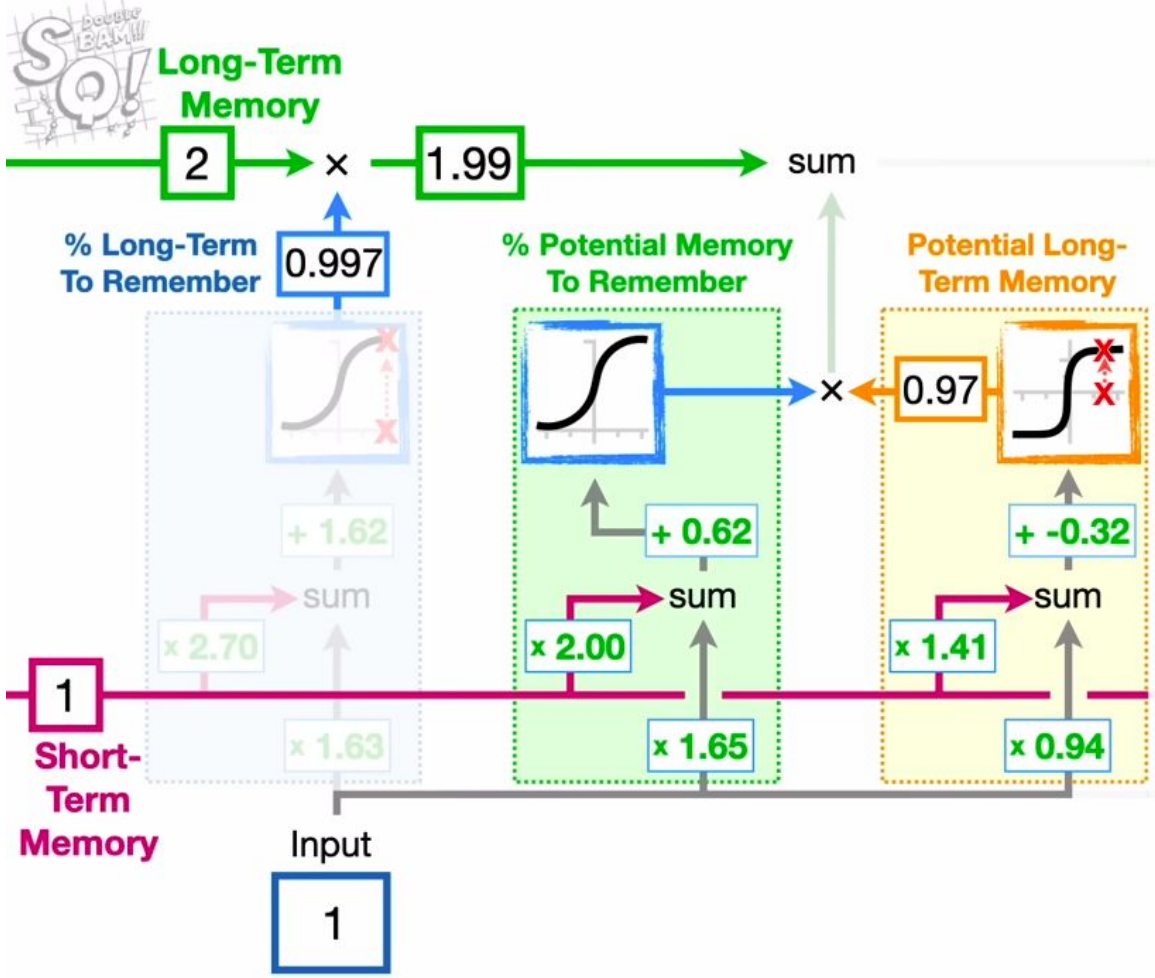

Long-Term Memory
2
×
sum
% Long-Term To Remember
% Potential Memory To Remember
Potential Long-Term Memory
In contrast, if the **Input** to the **LSTM** was **-10**…
+ 1.62
+ 0.62
+ -0.32
sum
sum
sum
× 2.70
× 2.00
× 1.41
1
× 1.63
× 1.65
× 0.94
Short-Term Memory
Input
-10


Long-Term Memory
2
×
0
sum
% Long-Term To Remember
0.0
% Potential Memory To Remember
Potential Long-Term Memory
-1.0
+ 1.62
sum
× 2.70
× 1.63
+ 0.62
sum
× 2.00
× 1.65
+ -0.32
sum
× 1.41
× 0.94
1
Short-Term Memory
Input
-10
In contrast, if the Input to the LSTM was -10…
…then, after calculating the x-axis coordinate, the output from the Tanh Activation Function function would be -1.

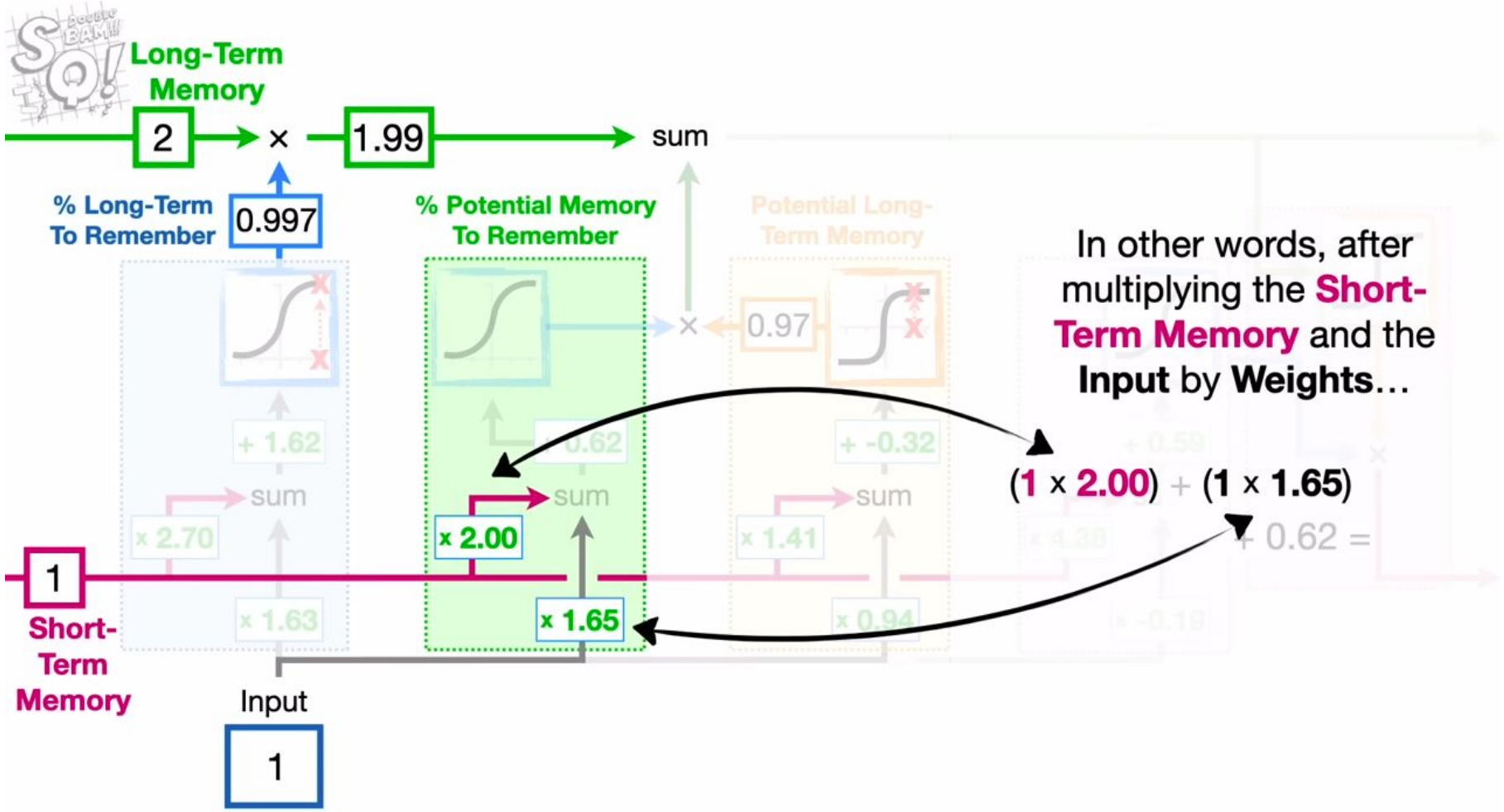

Long-Term Memory
2
×
1.99
sum
% Long-Term To Remember
0.997
% Potential Memory To Remember
Potential Long-Term Memory
0.97
+ 1.62
+ 0.62
+ -0.32
sum
sum
sum
× 2.70
× 2.00
× 1.41
1
Short-Term Memory
× 1.63
× 1.65
× 0.94
Input
1
Going back to when the Input to the LSTM was 1…

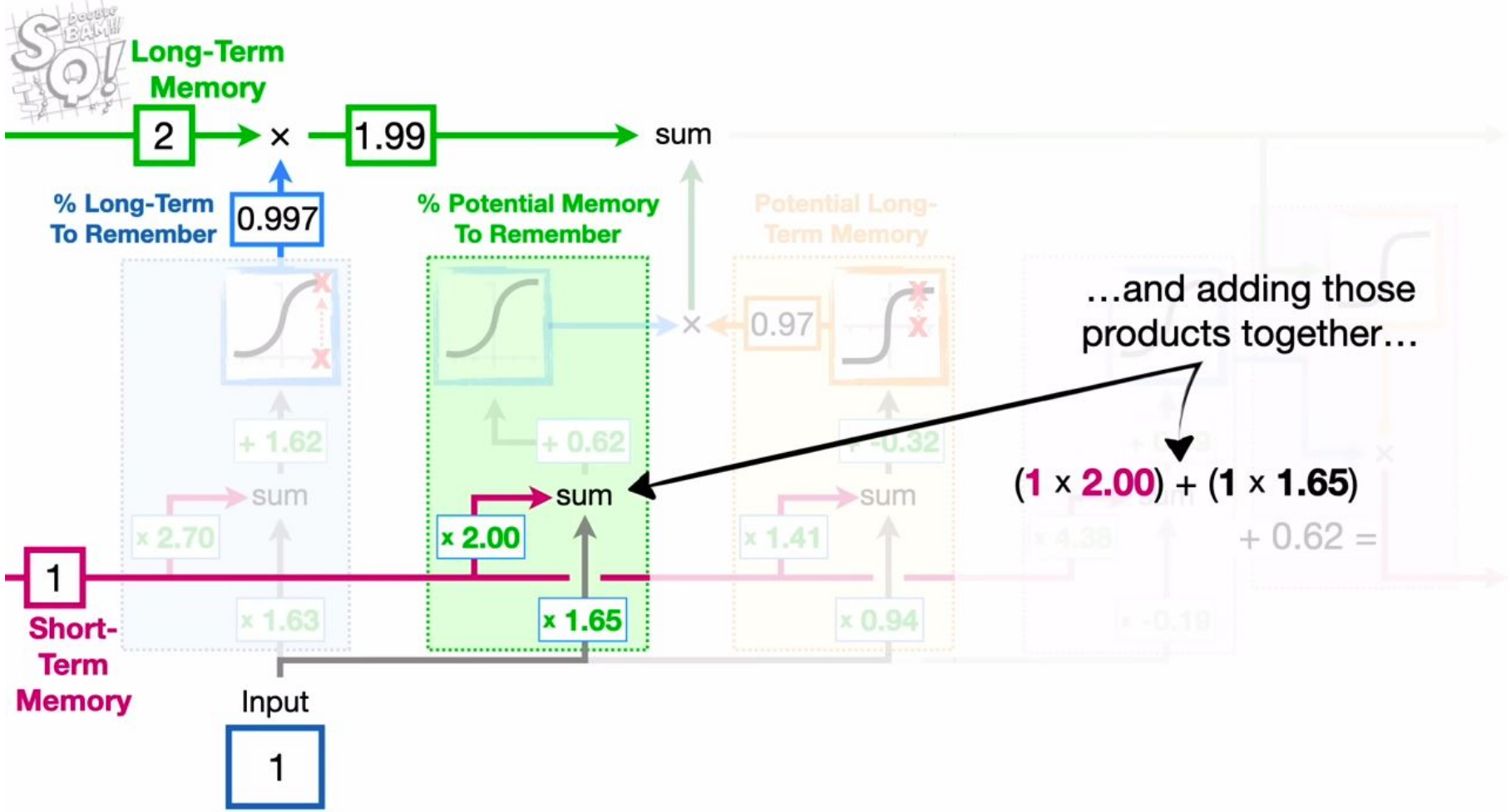

Long-Term Memory
2
×
1.99
sum
% Long-Term To Remember
0.997
% Potential Memory To Remember
Potential Long-Term Memory
0.97
…we have a Potential Memory, 0.97, based on the Short-Term Memory and the Input.
+ 1.62
+ 0.62
+ -0.32
sum
sum
sum
× 2.70
× 2.00
× 1.41
1
Short-Term Memory
× 1.63
× 1.65
× 0.94
Input
1

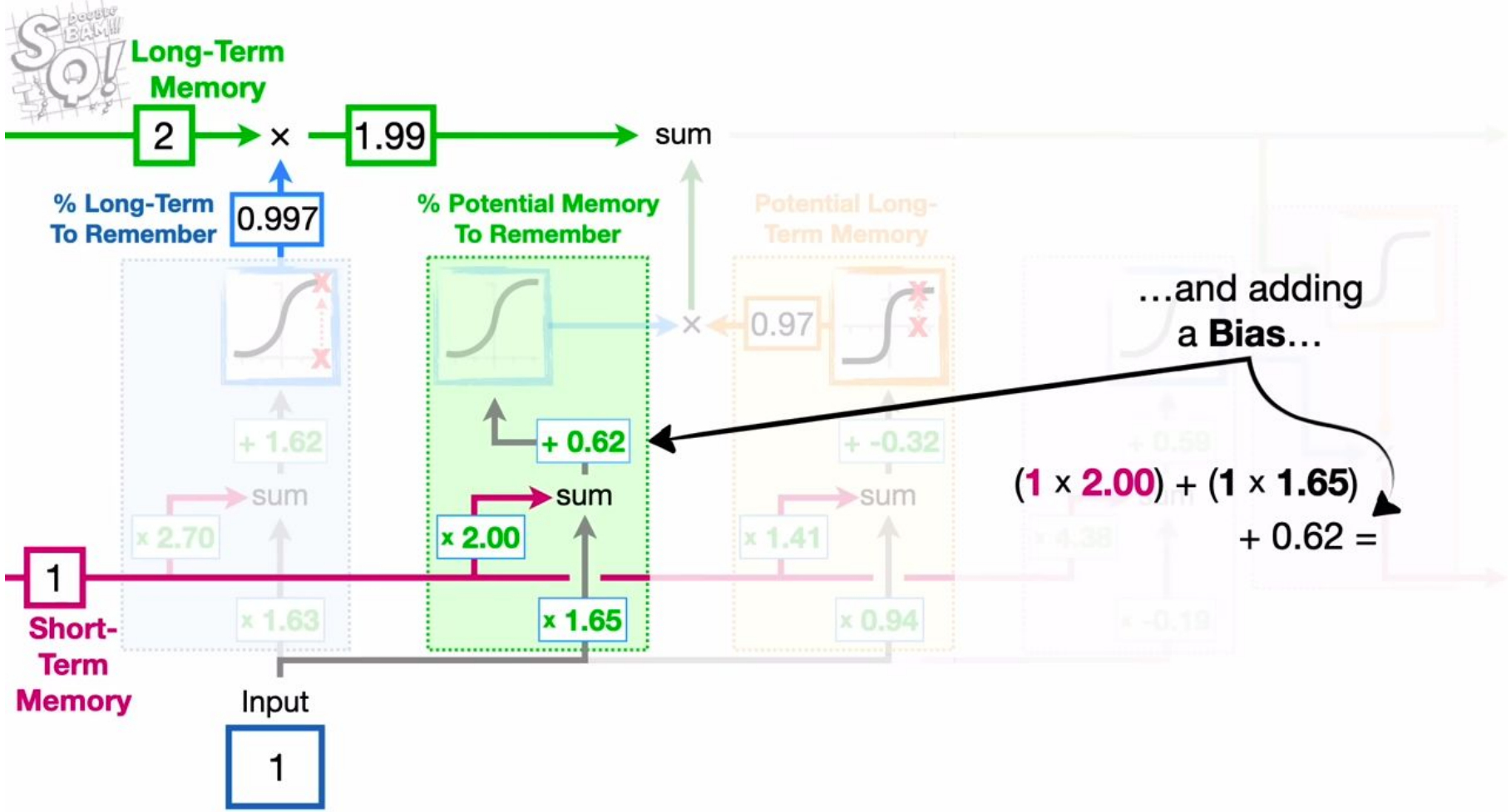

Long-Term Memory
2
×
1.99
sum
% Long-Term To Remember
0.997
% Potential Memory To Remember
Potential Long-Term Memory
0.97
+ -0.32
sum
× 1.41
× 0.94
+ 1.62
sum
× 2.70
× 1.63
+ 0.62
sum
× 2.00
× 1.65
1
Short-Term Memory
Input
1
Now the LSTM has to decide how much of this Potential Memory to save…

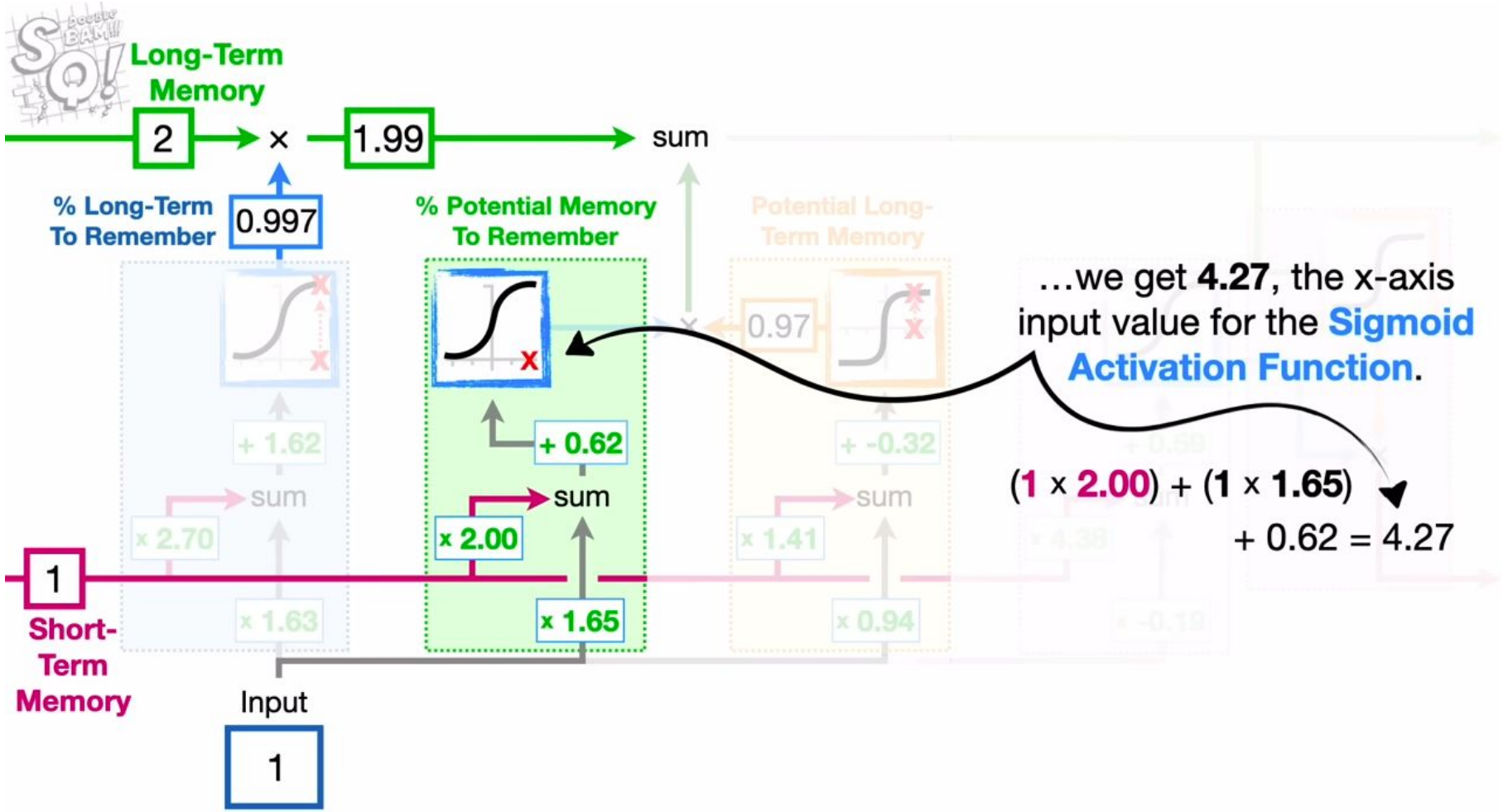


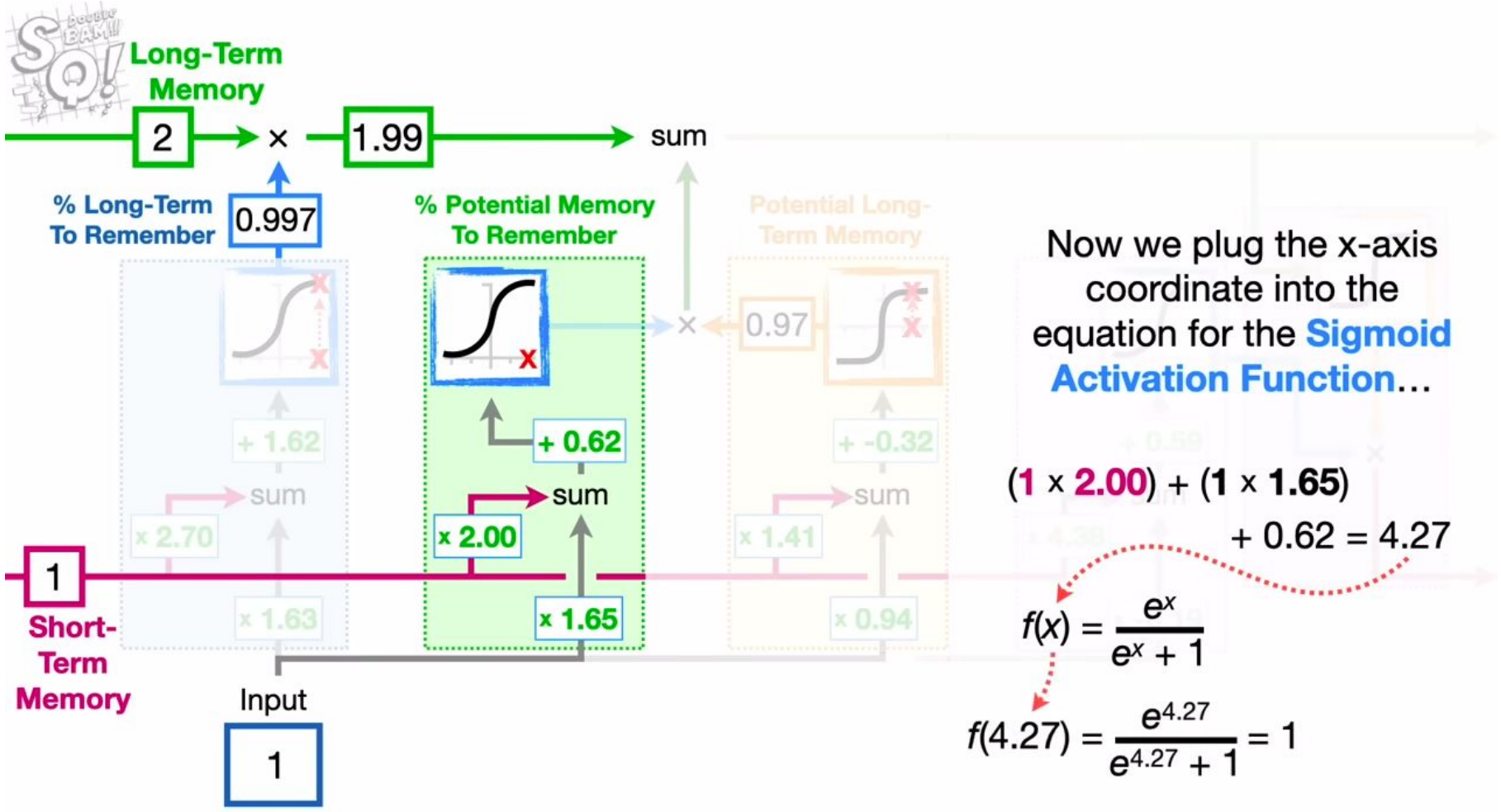
…and this is done using the exact same method we used earlier, when we determined what percentage of the **Long-Term Memory** to remember.


Long-Term Memory
2
×
1.99
sum
% Long-Term To Remember
0.997
% Potential Memory To Remember
Potential Long-Term Memory
0.97
In other words, after multiplying the Short-Term Memory and the Input by Weights…
(1 x 2.00) + (1 x 1.65)
+ 1.62
sum
× 2.70
× 1.63
+ 0.62
sum
× 2.00
× 1.65
+ -0.32
sum
× 1.41
× 0.94
1
Short-Term Memory
Input
1


Long-Term Memory
2
×
1.99
sum
% Long-Term To Remember
0.997
% Potential Memory To Remember
Potential Long-Term Memory
0.97
…and adding those products together…
(1 x 2.00) + (1 x 1.65)
+ 1.62
sum
× 2.70
× 1.63
+ 0.62
sum
× 2.00
× 1.65
+ 0.32
sum
× 1.41
× 0.94
1
Short-Term Memory
Input
1


Long-Term Memory
2
×
1.99
sum
% Long-Term To Remember
0.997
% Potential Memory To Remember
Potential Long-Term Memory
0.97
…and adding a Bias…
+ 0.62
sum
× 2.00
× 1.65
(1 x 2.00) + (1 x 1.65) + 0.62 =
1
Short-Term Memory
Input
1


Long-Term Memory
2
×
1.99
sum
% Long-Term To Remember
0.997
% Potential Memory To Remember
Potential Long-Term Memory
0.97
…we get 4.27, the x-axis input value for the Sigmoid Activation Function.
+ 1.62
+ 0.62
+ -0.32
sum
sum
sum
x 2.70
x 2.00
x 1.41
(1 x 2.00) + (1 x 1.65) + 0.62 = 4.27
1
x 1.63
x 1.65
x 0.94
Short-Term Memory
Input
1

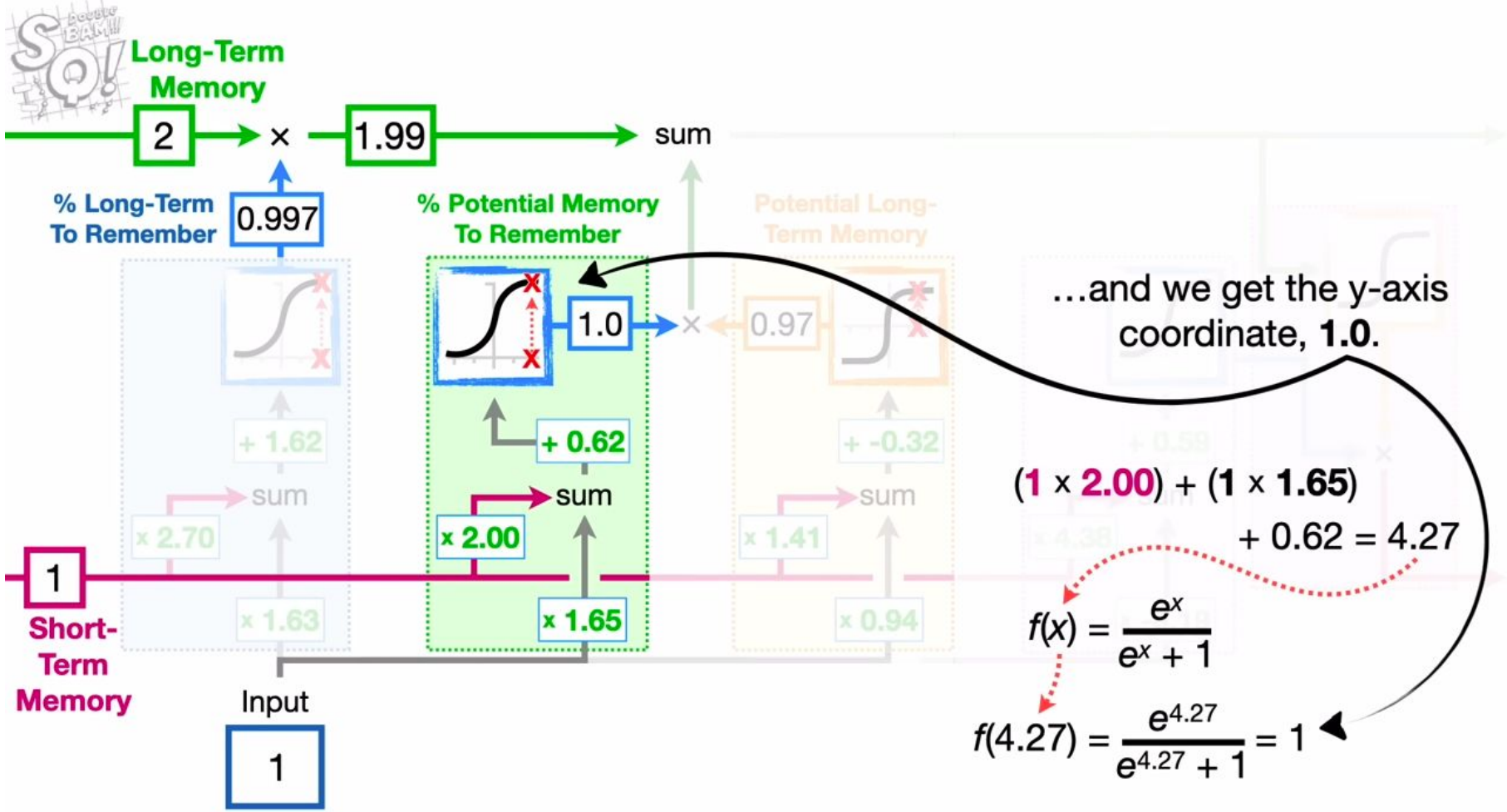


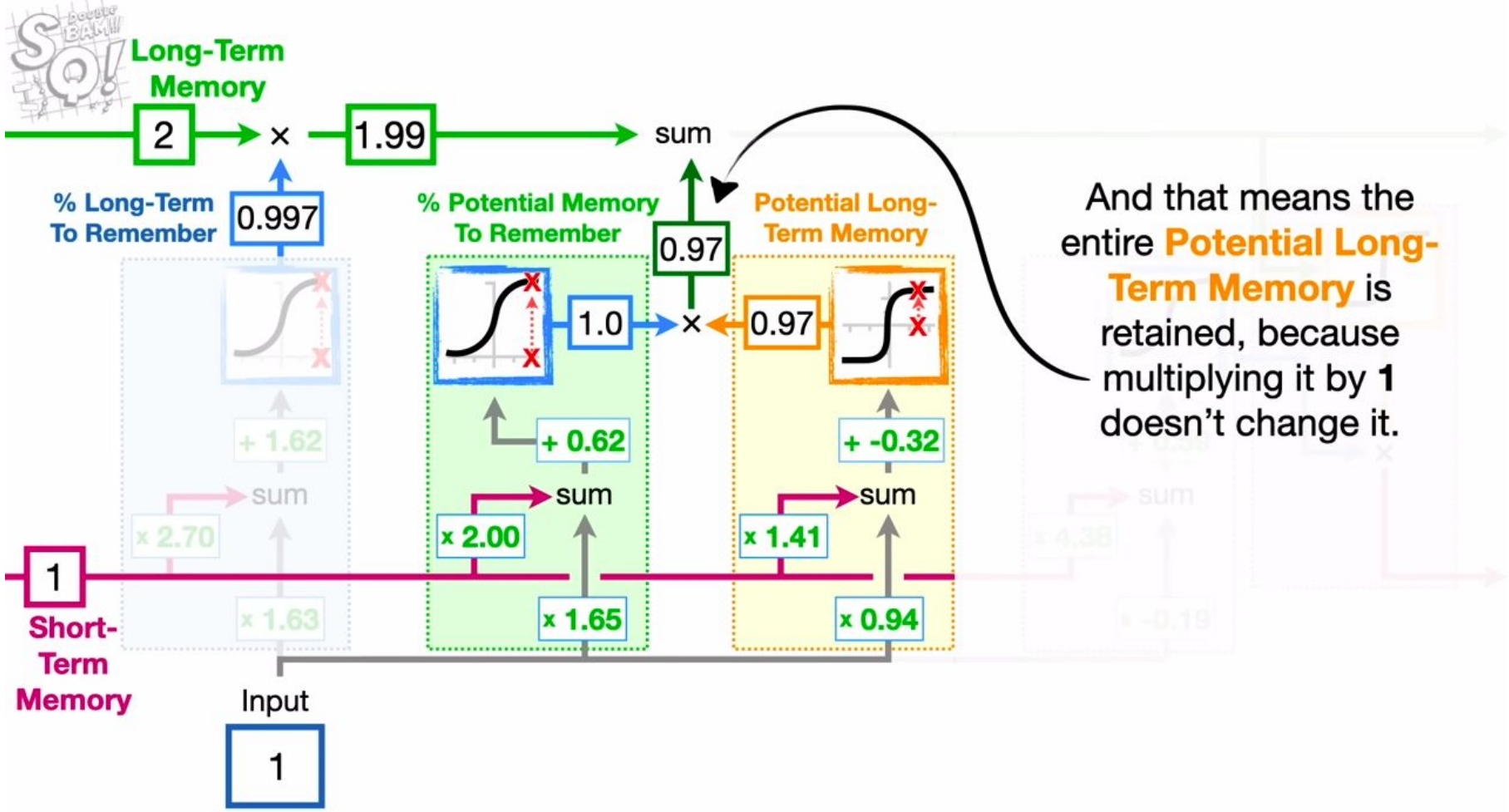
Now we plug the x-axis coordinate into the equation for the **Sigmoid Activation Function**…

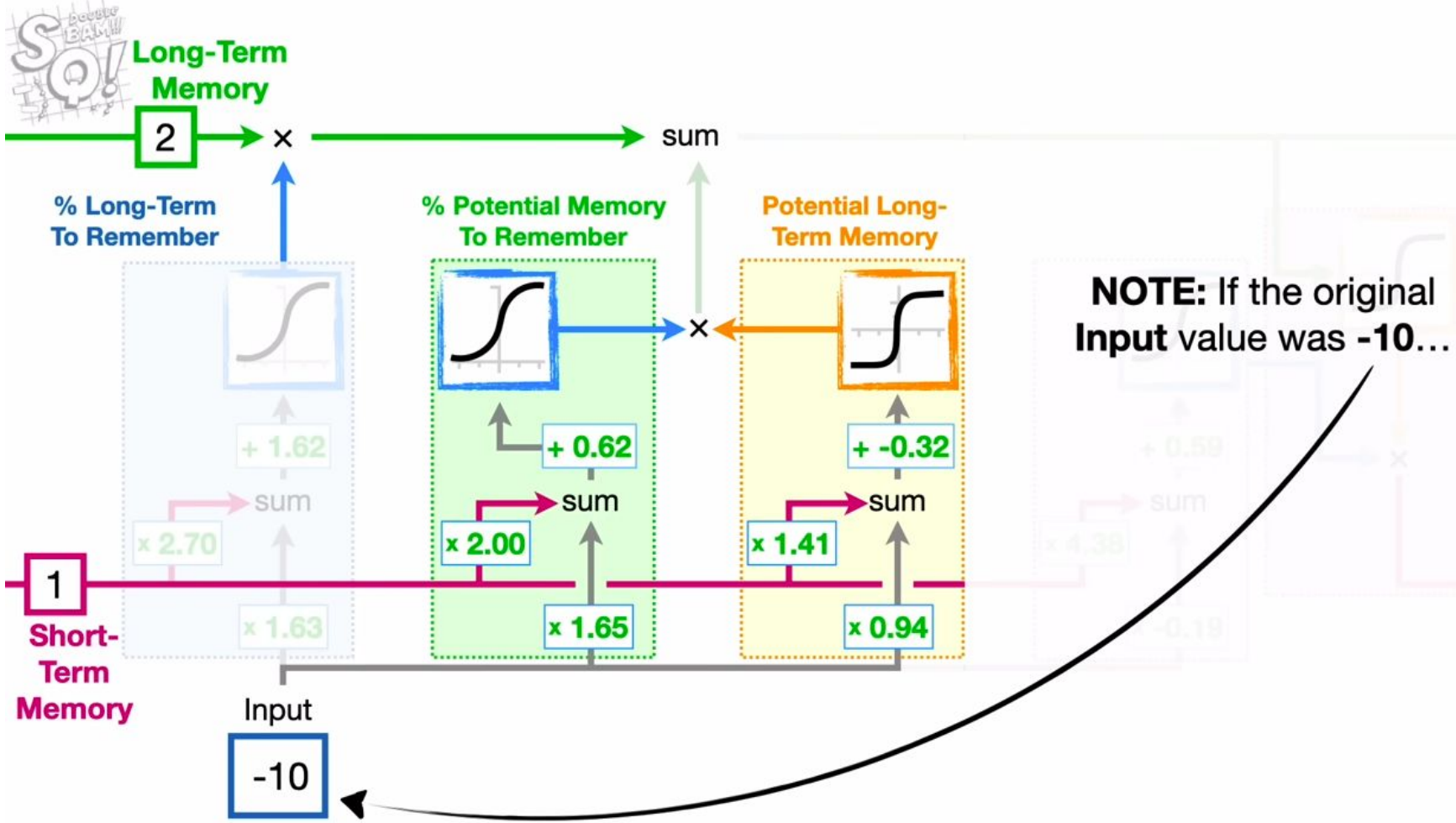
$$(1 \times 2.00) + (1 \times 1.65) + 0.62 = 4.27$$

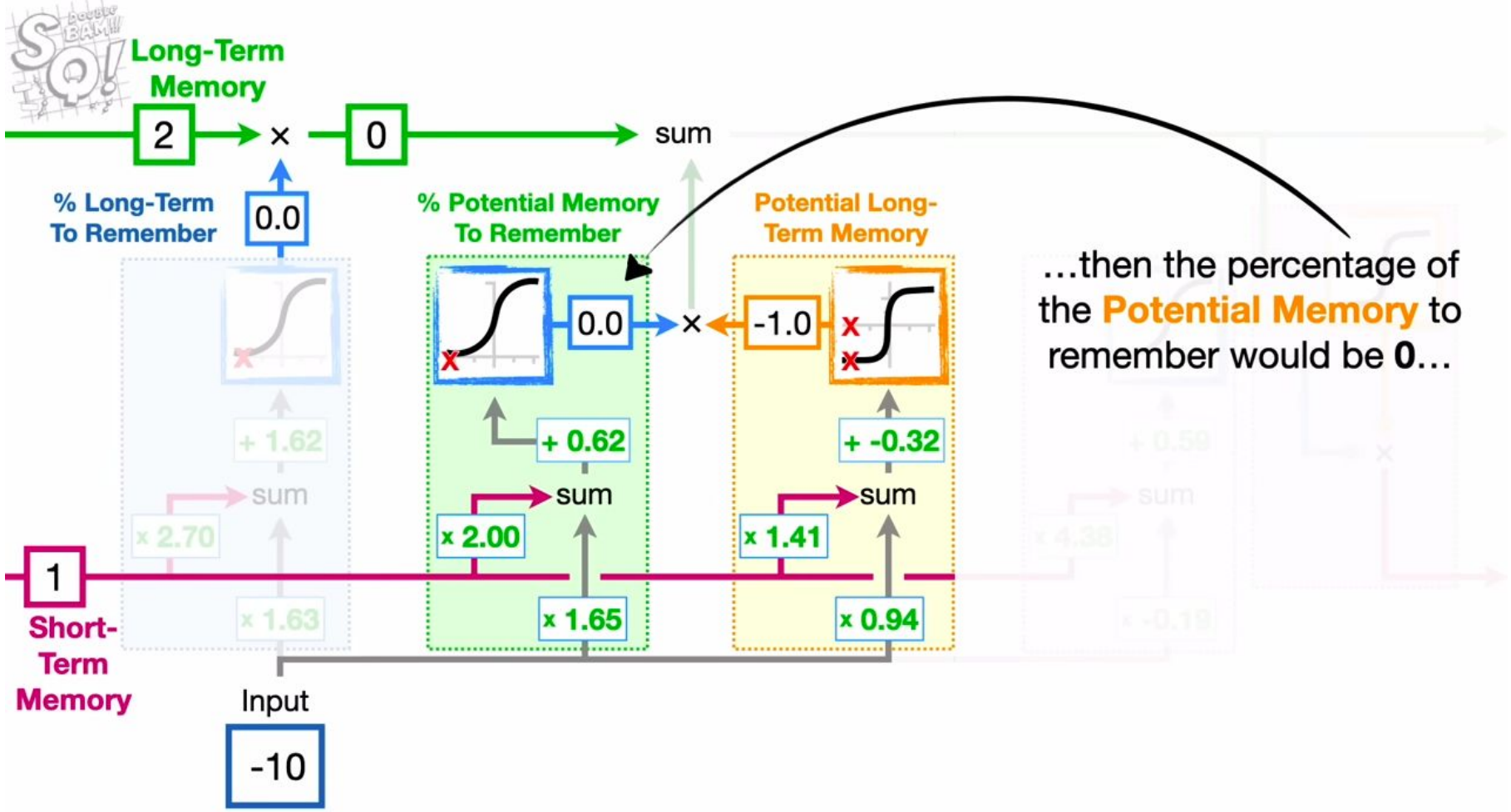
$$f(x) = \frac{e^x}{e^x + 1}$$

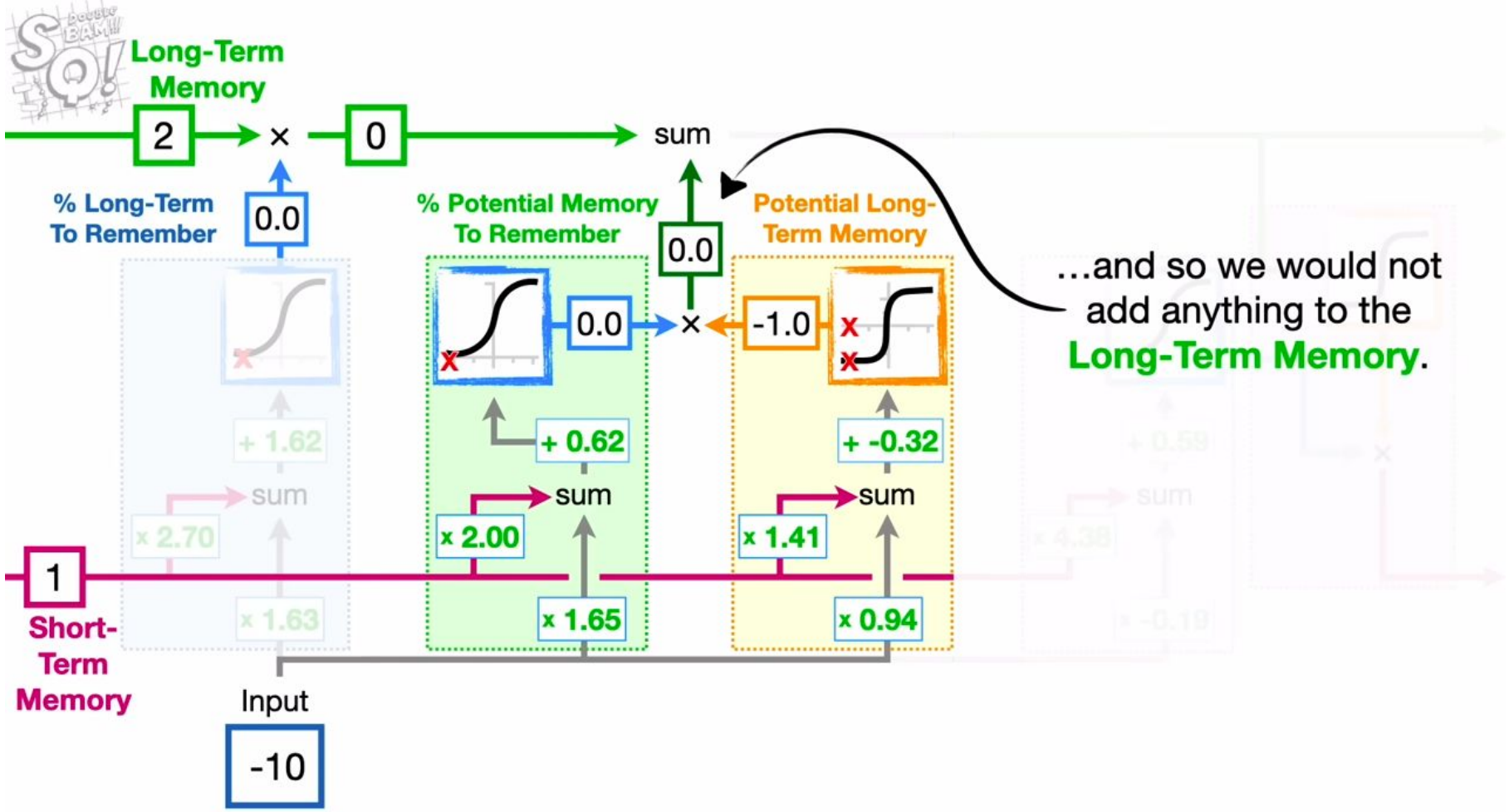
$$f(4.27) = \frac{e^{4.27}}{e^{4.27} + 1} = 1$$

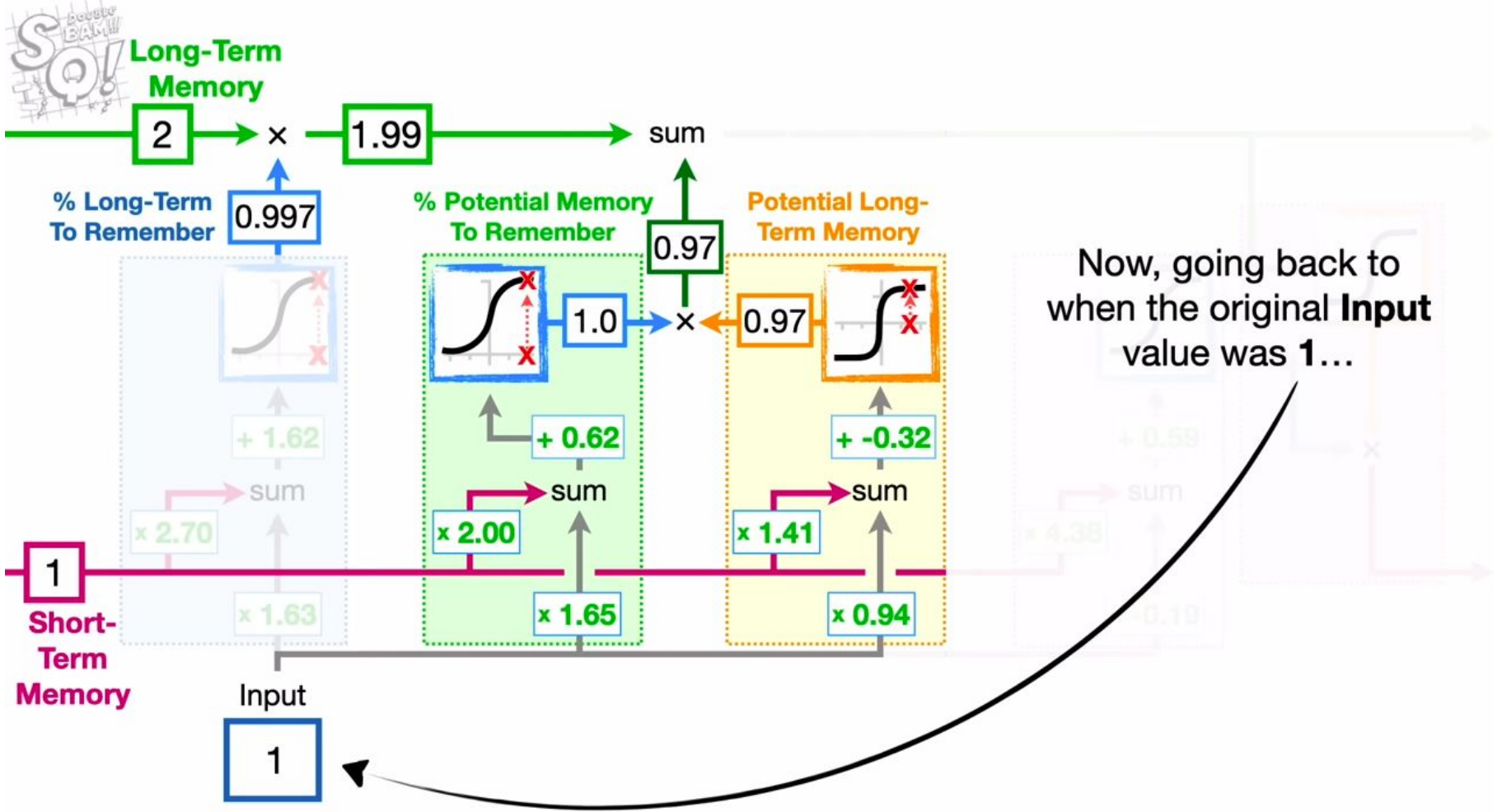

Long-Term Memory
2
×
1.99
sum
% Long-Term To Remember
0.997
% Potential Memory To Remember
Potential Long-Term Memory
1.0
0.97
+ 0.62
+ 1.62
+ -0.32
sum
sum
sum
× 2.00
× 2.70
× 1.41
× 1.65
× 1.63
× 0.94
1
Short-Term Memory
Input
1
…and we get the y-axis coordinate, 1.0.
(1 x 2.00) + (1 x 1.65)
+ 0.62 = 4.27
$f(x) = \frac{e^x}{e^x + 1}$
$f(4.27) = \frac{e^{4.27}}{e^{4.27} + 1} = 1$

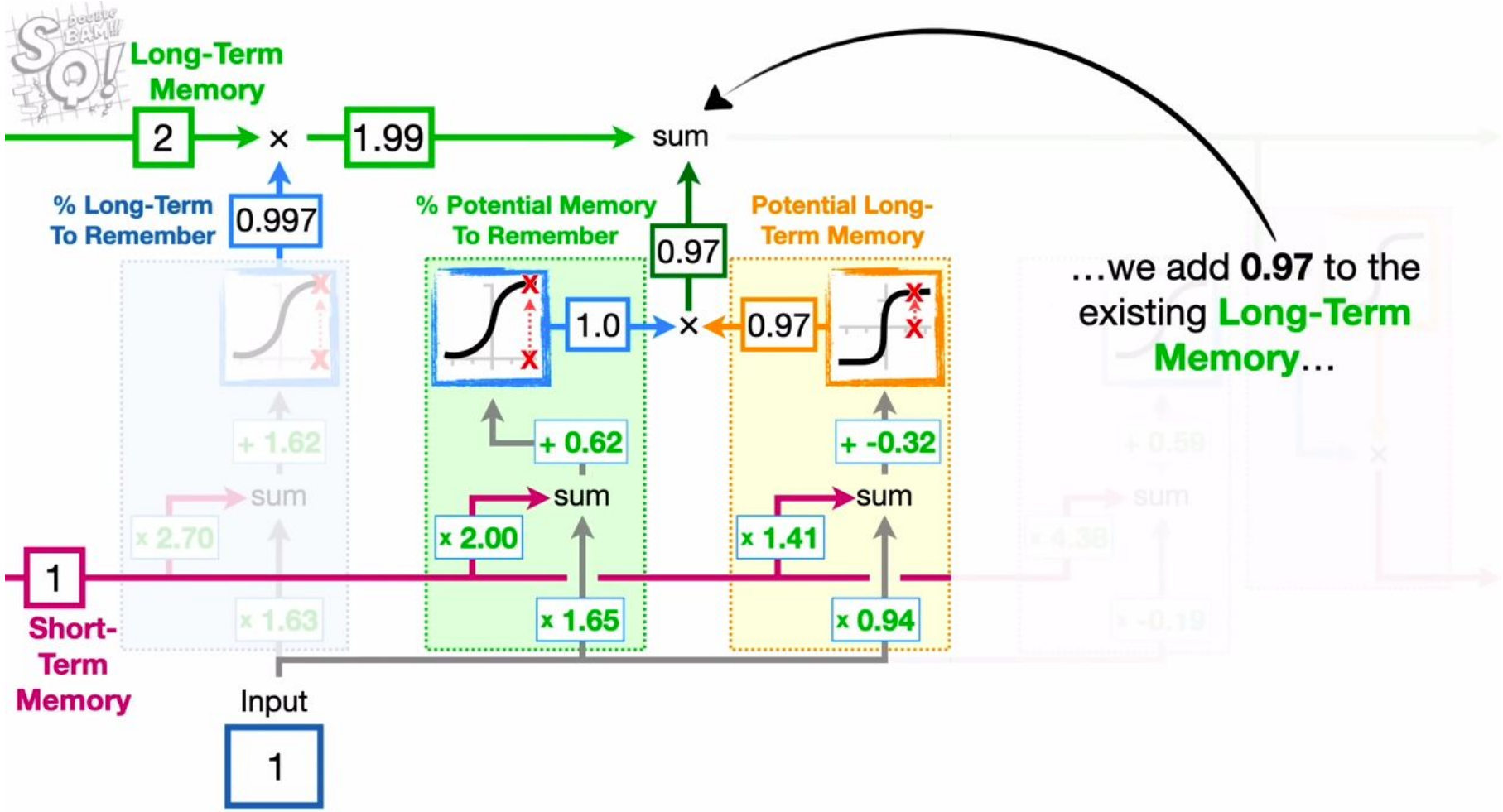

Long-Term Memory
2
×
1.99
sum
% Long-Term To Remember
0.997
% Potential Memory To Remember
0.97
Potential Long-Term Memory
1.0
×
0.97
+ 0.62
+ -0.32
sum
sum
× 2.00
× 1.41
1
Short-Term Memory
× 1.65
× 0.94
Input
1
And that means the entire Potential Long-Term Memory is retained, because multiplying it by 1 doesn't change it.

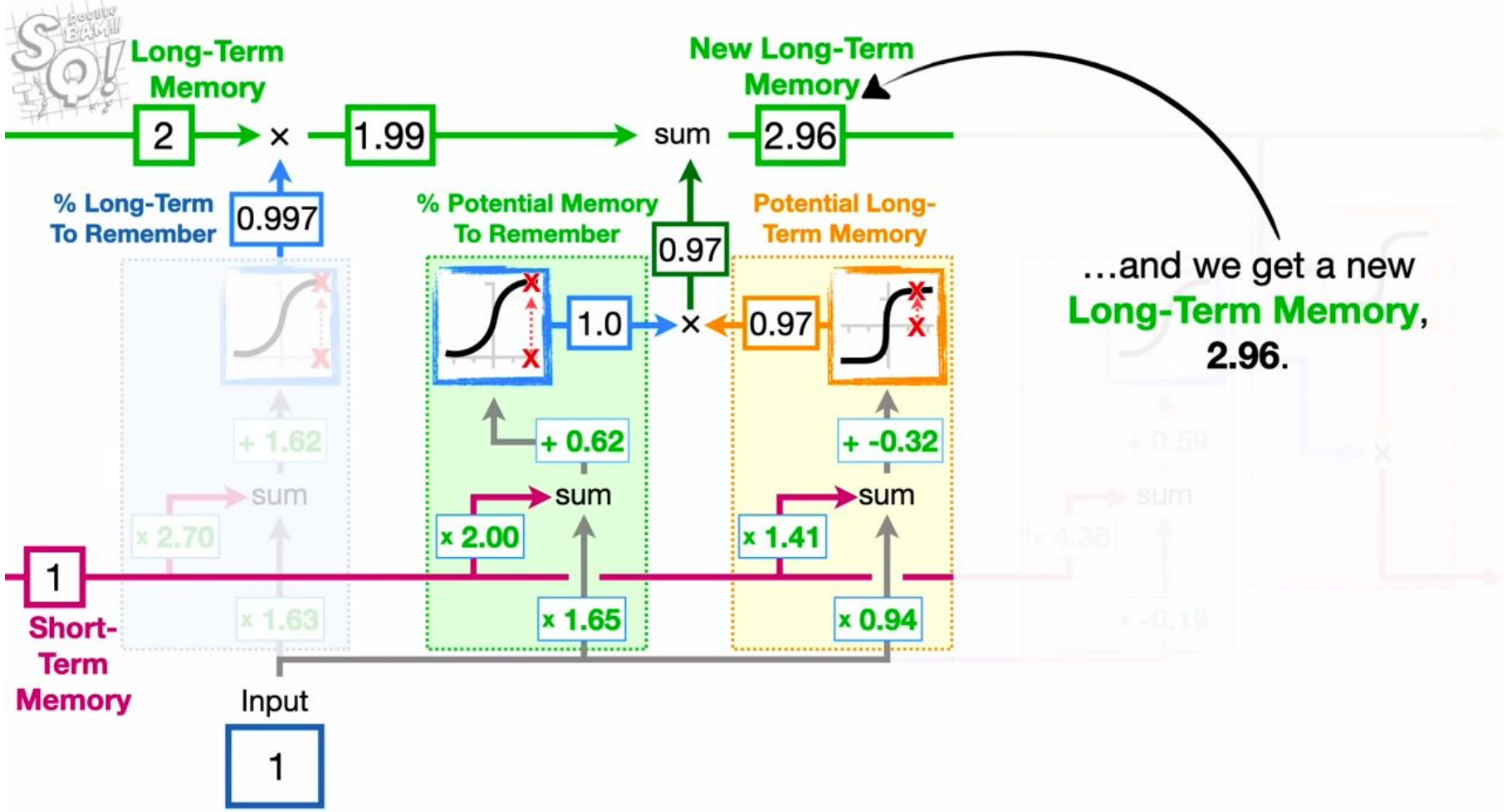

Long-Term Memory
2
×
sum
% Long-Term To Remember
% Potential Memory To Remember
Potential Long-Term Memory
×
+ 0.62
+ -0.32
sum
sum
× 2.00
× 1.41
1
Short-Term Memory
× 1.65
× 0.94
Input
-10
NOTE: If the original Input value was -10…

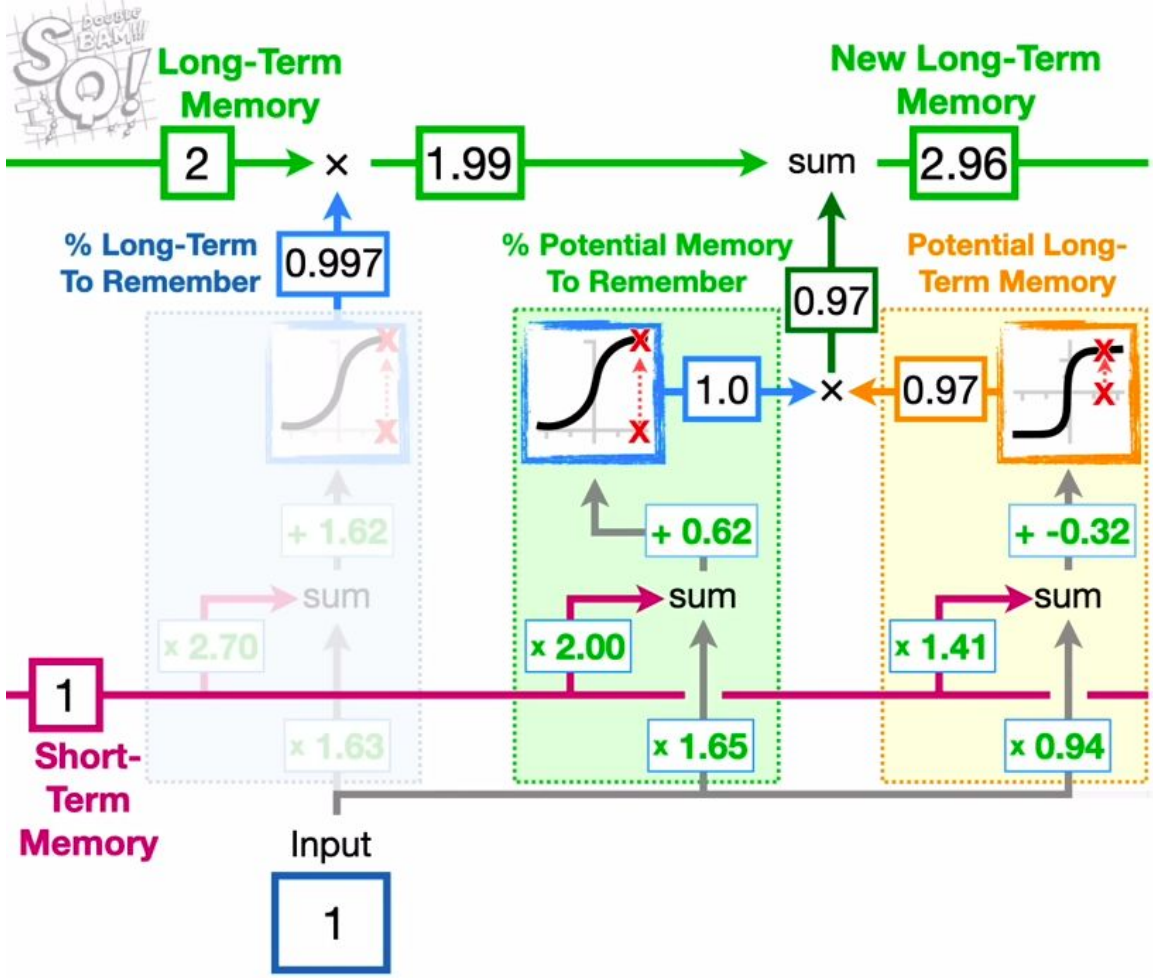

Long-Term Memory
2
×
0
sum
% Long-Term To Remember
0.0
% Potential Memory To Remember
Potential Long-Term Memory
0.0
×
-1.0
…then the percentage of the Potential Memory to remember would be 0…
+ 0.62
+ -0.32
sum
sum
× 2.00
× 1.41
1
× 1.65
× 0.94
Short-Term Memory
Input
-10


Long-Term Memory
2
×
0
sum
% Long-Term To Remember
0.0
% Potential Memory To Remember
0.0
Potential Long-Term Memory
0.0
×
-1.0
…and so we would not add anything to the Long-Term Memory.
+ 0.62
+ -0.32
sum
sum
× 2.00
× 1.41
1
× 1.65
× 0.94
Short-Term Memory
Input
-10


Long-Term Memory
2
×
1.99
sum
% Long-Term To Remember
0.997
% Potential Memory To Remember
0.97
Potential Long-Term Memory
1.0
×
0.97
+ 0.62
+ -0.32
sum
sum
× 2.00
× 1.41
1
Short-Term Memory
× 1.65
× 0.94
Input
1
Now, going back to when the original Input value was 1…

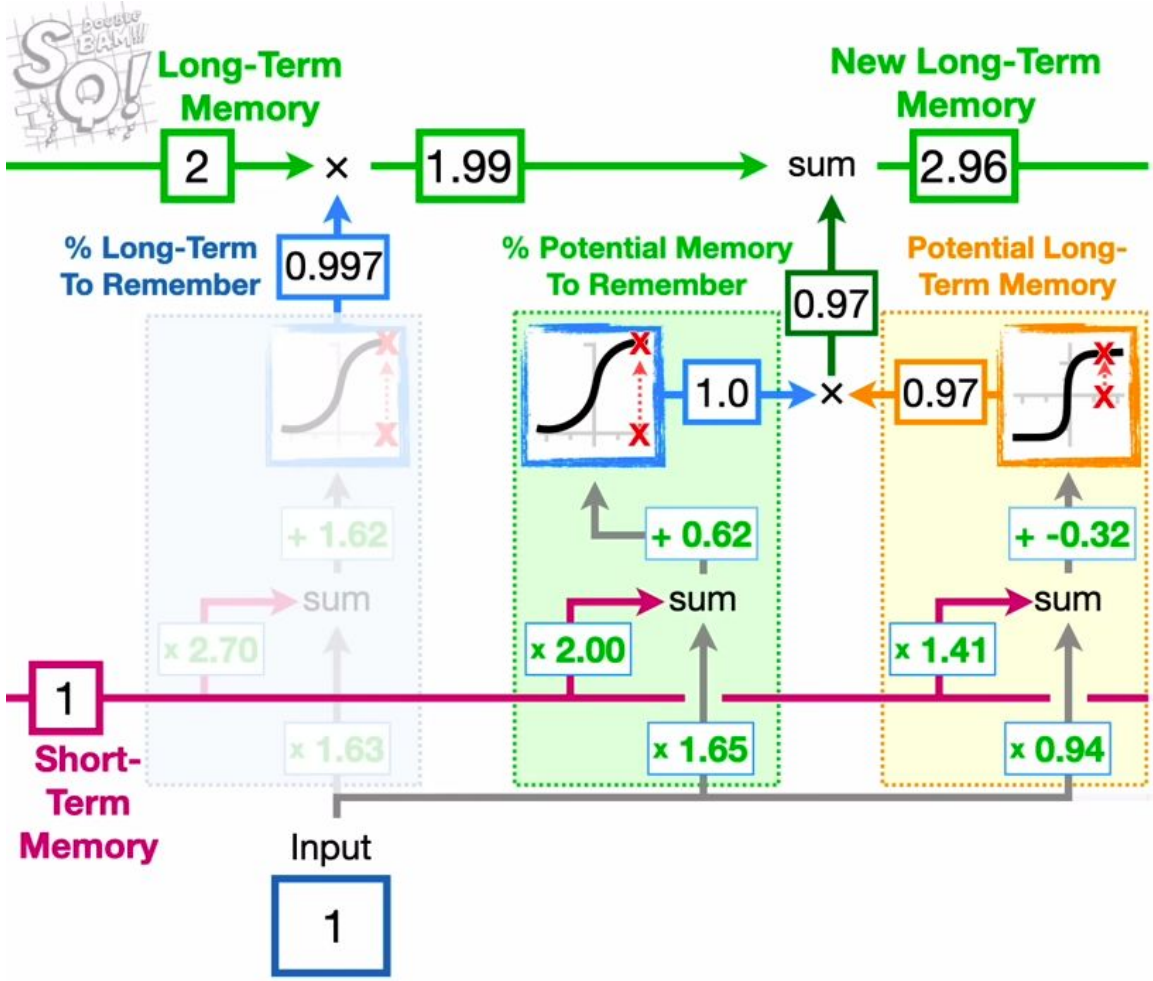

Long-Term Memory
2
×
1.99
sum
% Long-Term To Remember
0.997
% Potential Memory To Remember
0.97
Potential Long-Term Memory
1.0
×
0.97
…we add 0.97 to the existing Long-Term Memory…
+ 0.62
+ -0.32
sum
sum
× 2.00
× 1.41
1
Short-Term Memory
× 1.65
× 0.94
Input
1


Long-Term Memory
2
×
1.99
sum
New Long-Term Memory
2.96
% Long-Term To Remember
0.997
% Potential Memory To Remember
0.97
Potential Long-Term Memory
1.0
×
0.97
+ 1.62
sum
× 2.70
× 1.63
+ 0.62
sum
× 2.00
× 1.65
+ -0.32
sum
× 1.41
× 0.94
1
Short-Term Memory
Input
1
…and we get a new Long-Term Memory, 2.96.



TERMINOLOGY ALERT!!!

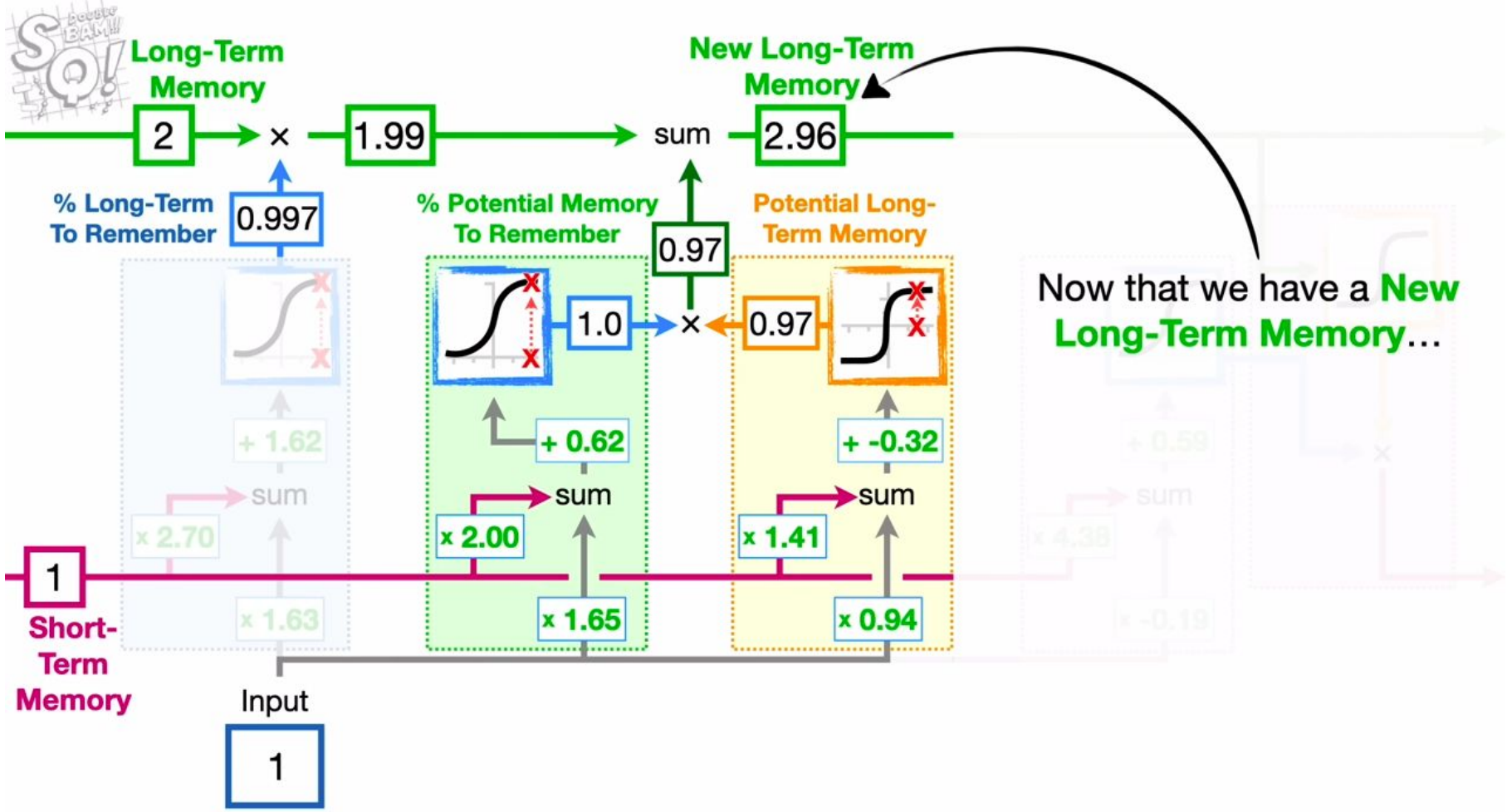
Even though this part of the **Long Short-Term Memory** unit determines how we should update the **Long-Term Memory**…

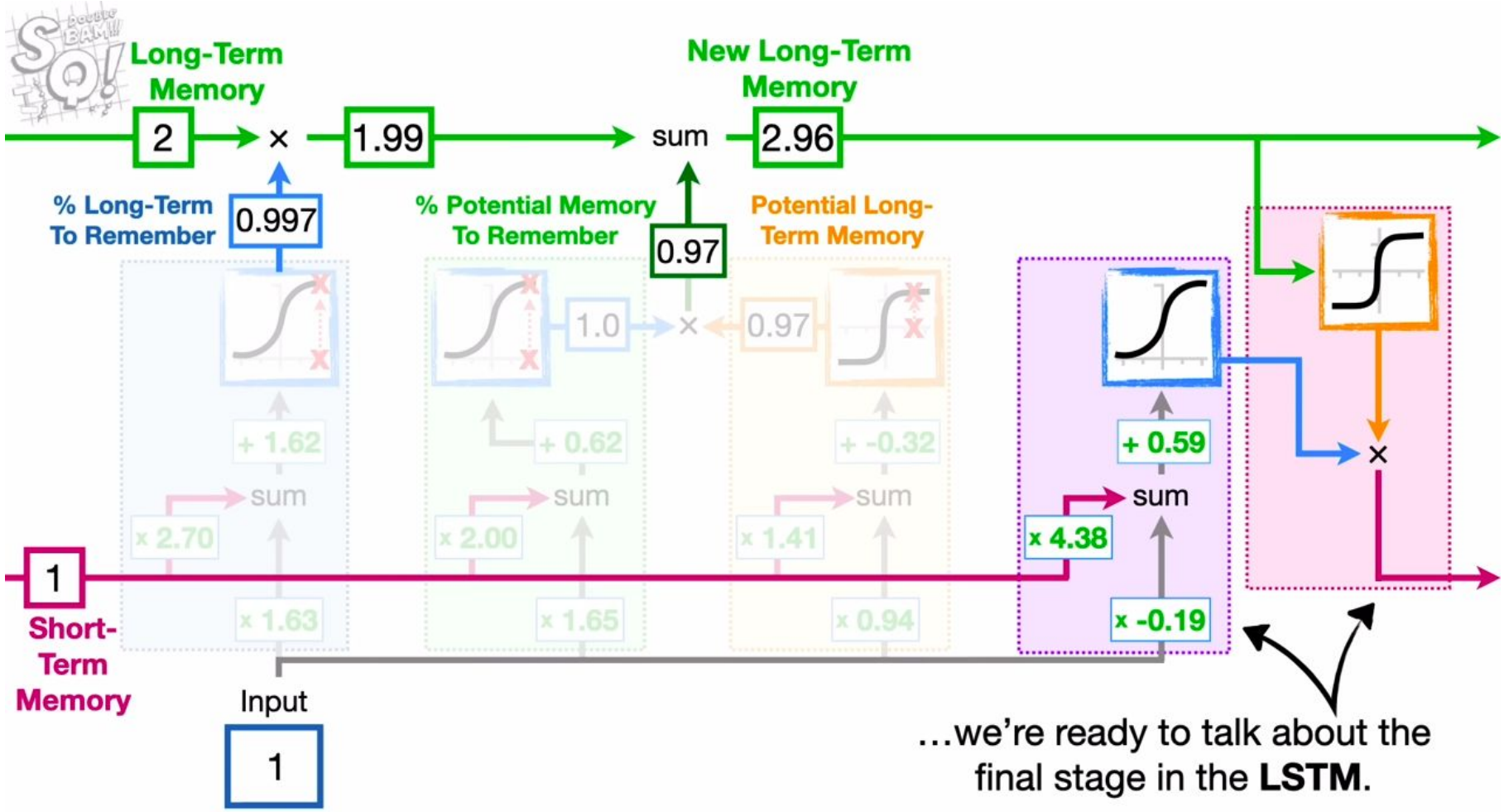


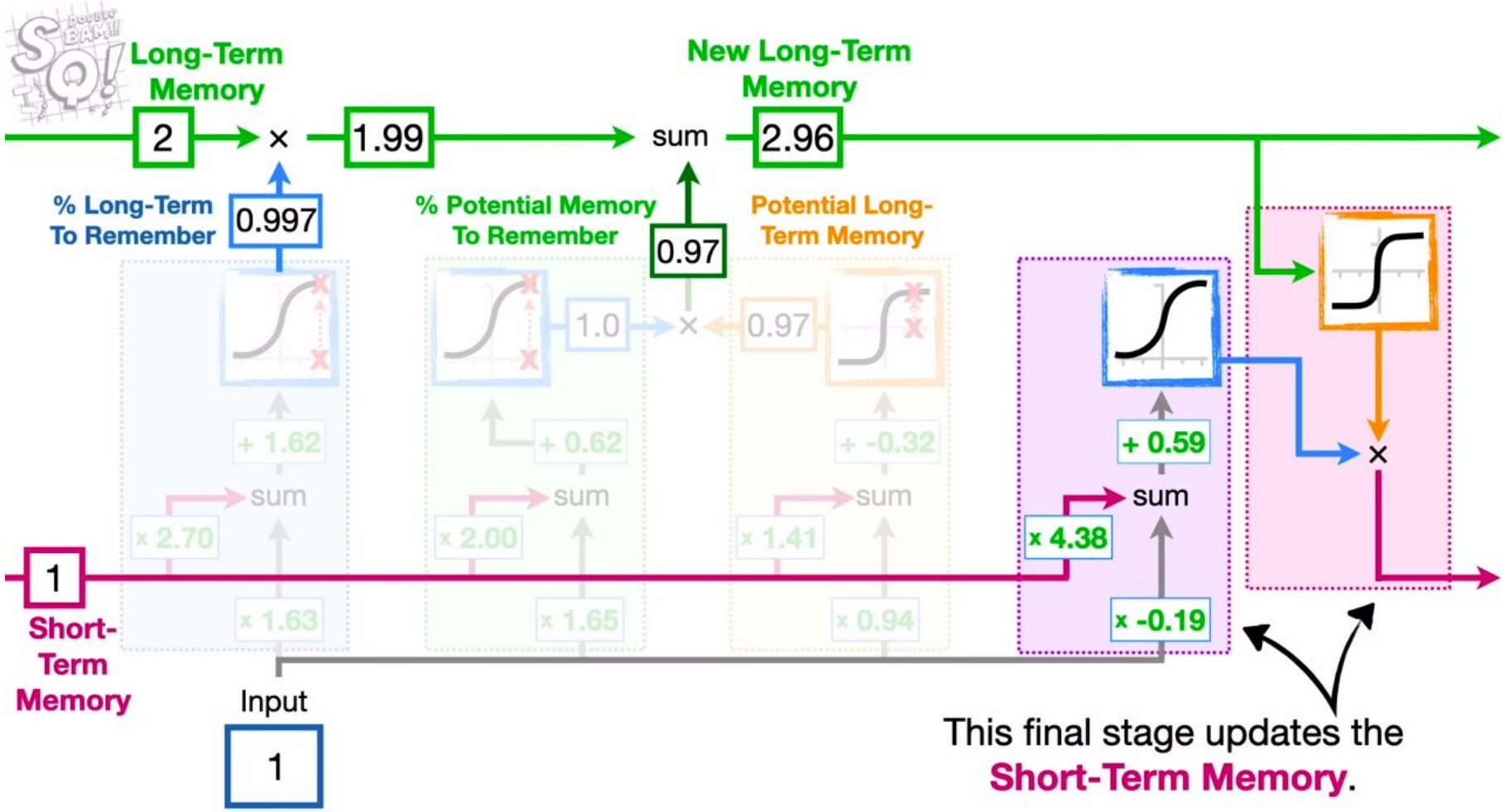
TERMINOLOGY ALERT!!!

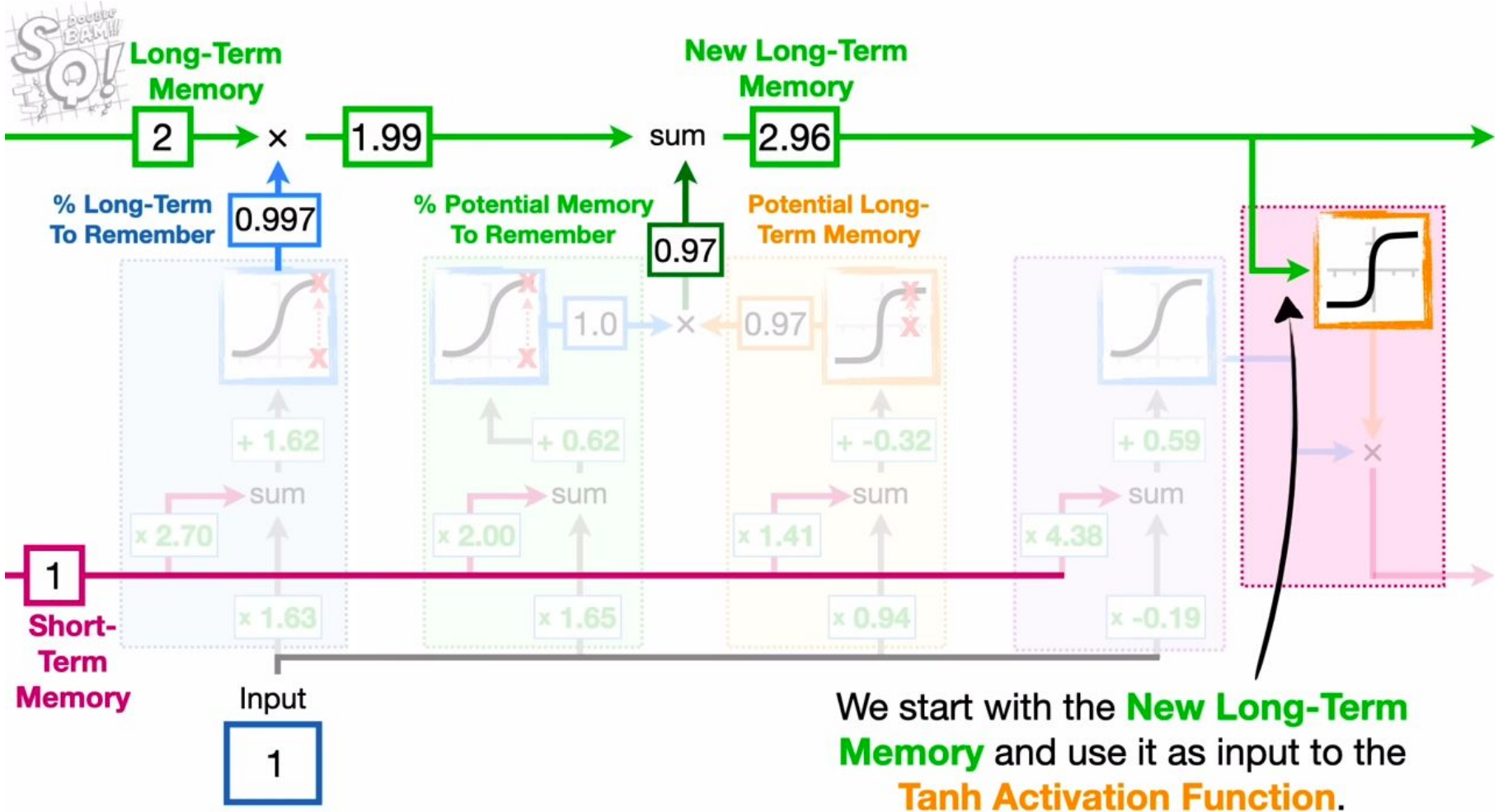
Even though this part of the **Long Short-Term Memory** unit determines how we should update the **Long-Term Memory**…

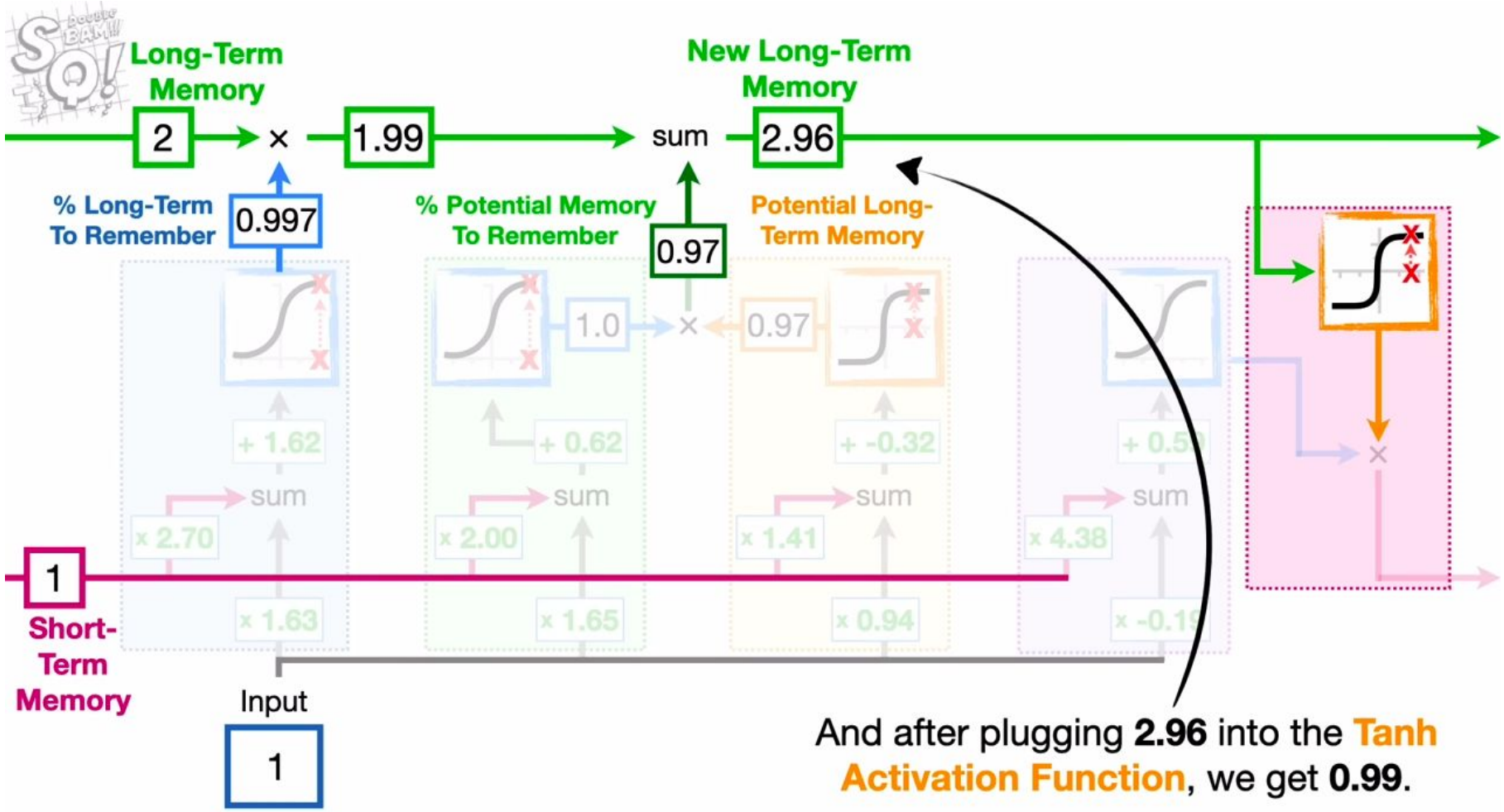
…it is usually called the **Input Gate**.

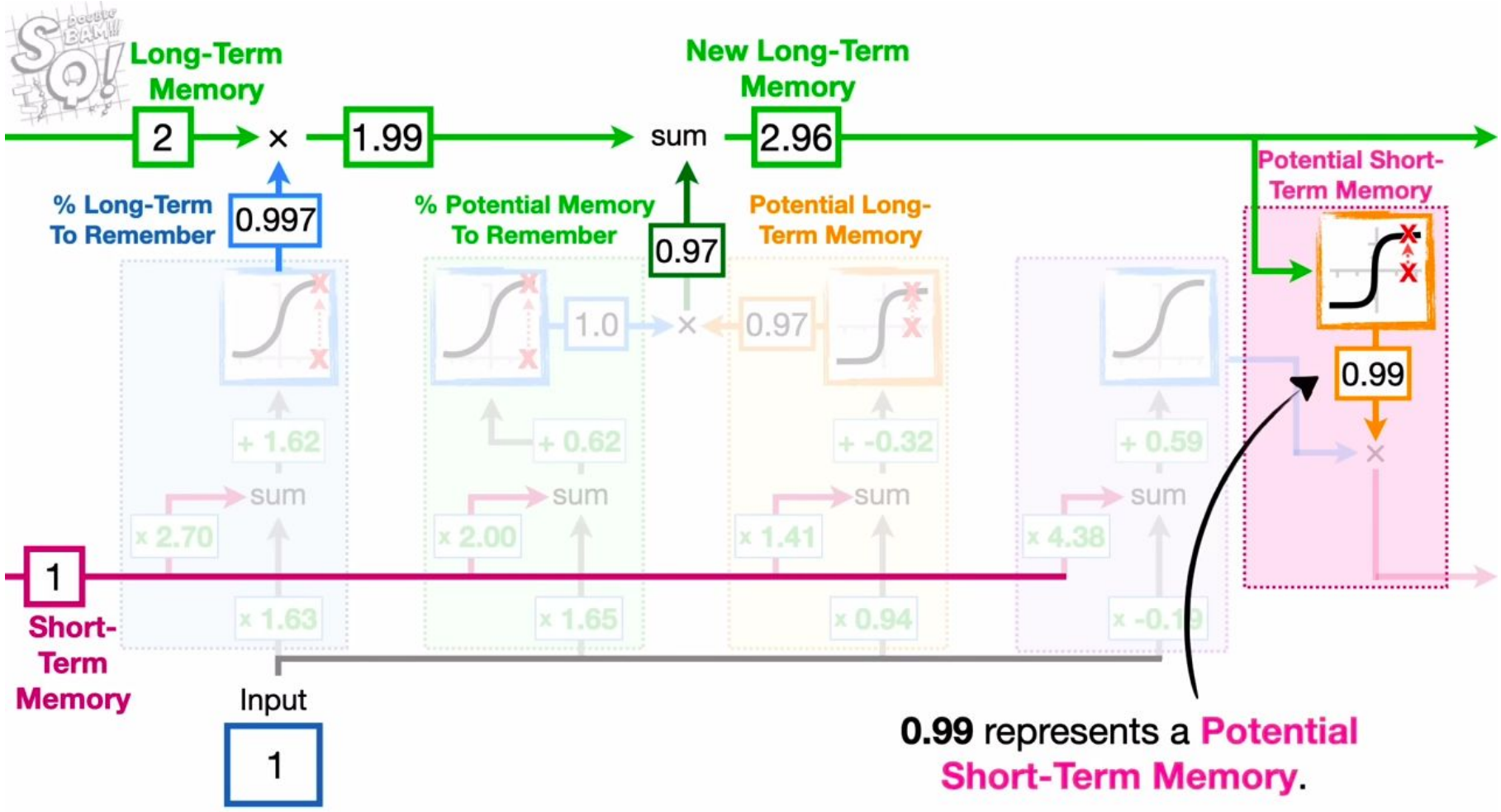

Long-Term Memory
2
×
1.99
sum
2.96
New Long-Term Memory
% Long-Term To Remember
0.997
% Potential Memory To Remember
0.97
Potential Long-Term Memory
1.0
×
0.97
+ 0.62
+ -0.32
sum
sum
× 2.00
× 1.41
1
Short-Term Memory
× 1.65
× 0.94
Input
1
Now that we have a New Long-Term Memory…

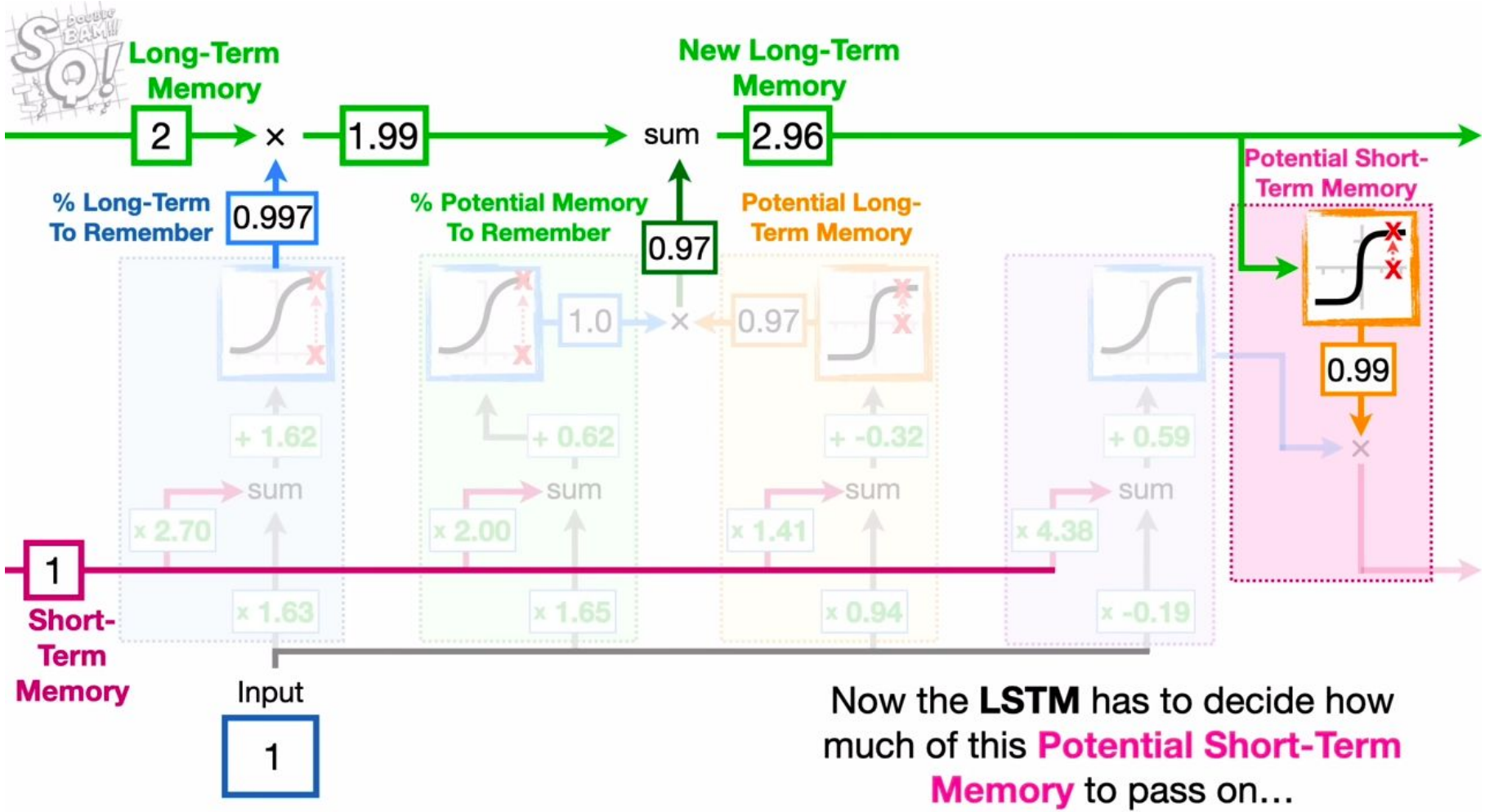

Long-Term Memory
2
×
1.99
sum
New Long-Term Memory
2.96
% Long-Term To Remember
0.997
% Potential Memory To Remember
0.97
Potential Long-Term Memory
1.0
×
0.97
+ 1.62
sum
× 2.70
× 1.63
+ 0.62
sum
× 2.00
× 1.65
+ -0.32
sum
× 1.41
× 0.94
+ 0.59
sum
× 4.38
× -0.19
×
1
Short-Term Memory
Input
1
…we're ready to talk about the final stage in the **LSTM**.

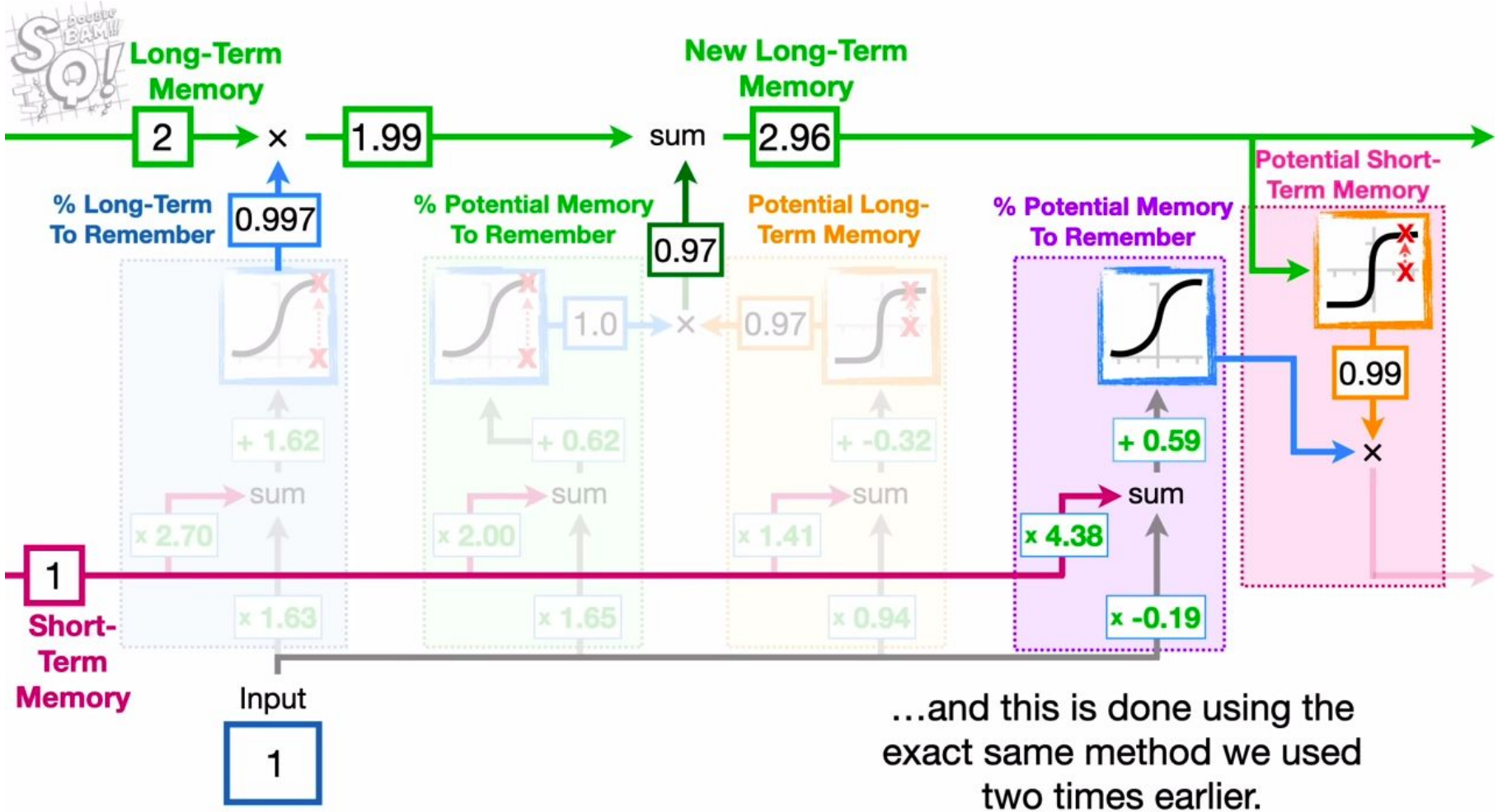

Long-Term Memory
2
×
1.99
sum
New Long-Term Memory
2.96
% Long-Term To Remember
0.997
% Potential Memory To Remember
0.97
Potential Long-Term Memory
1.0
×
0.97
+ 1.62
sum
× 2.70
× 1.63
+ 0.62
sum
× 2.00
× 1.65
+ -0.32
sum
× 1.41
× 0.94
+ 0.59
sum
× 4.38
× -0.19
×
1
Short-Term Memory
Input
1
This final stage updates the Short-Term Memory.

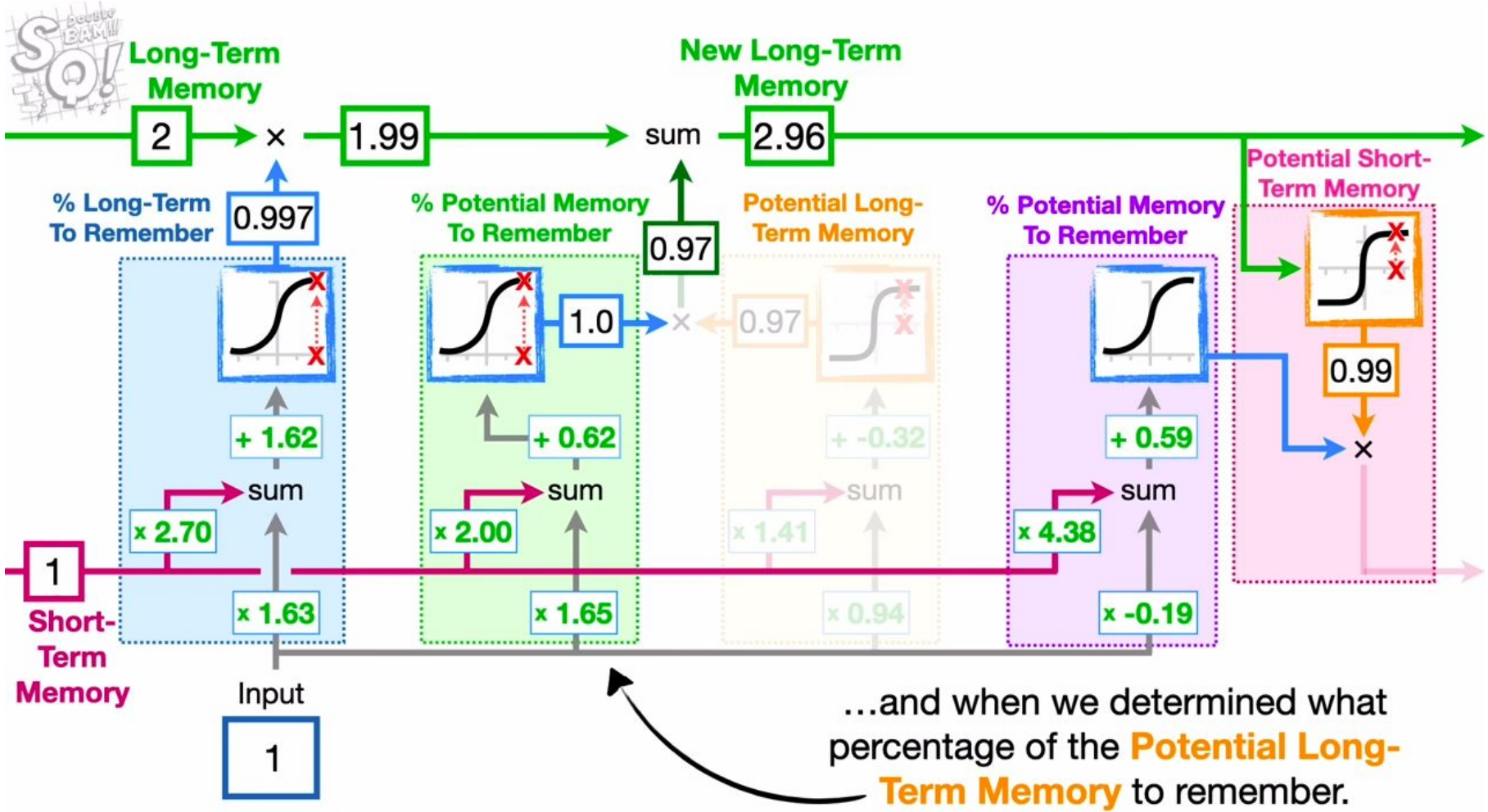

Long-Term Memory
2
×
1.99
sum
New Long-Term Memory
2.96
% Long-Term To Remember
0.997
% Potential Memory To Remember
0.97
Potential Long-Term Memory
1.0
×
0.97
+ 1.62
sum
× 2.70
× 1.63
+ 0.62
sum
× 2.00
× 1.65
+ -0.32
sum
× 1.41
× 0.94
+ 0.59
sum
× 4.38
× -0.19
1
Short-Term Memory
Input
1
We start with the New Long-Term Memory and use it as input to the Tanh Activation Function.

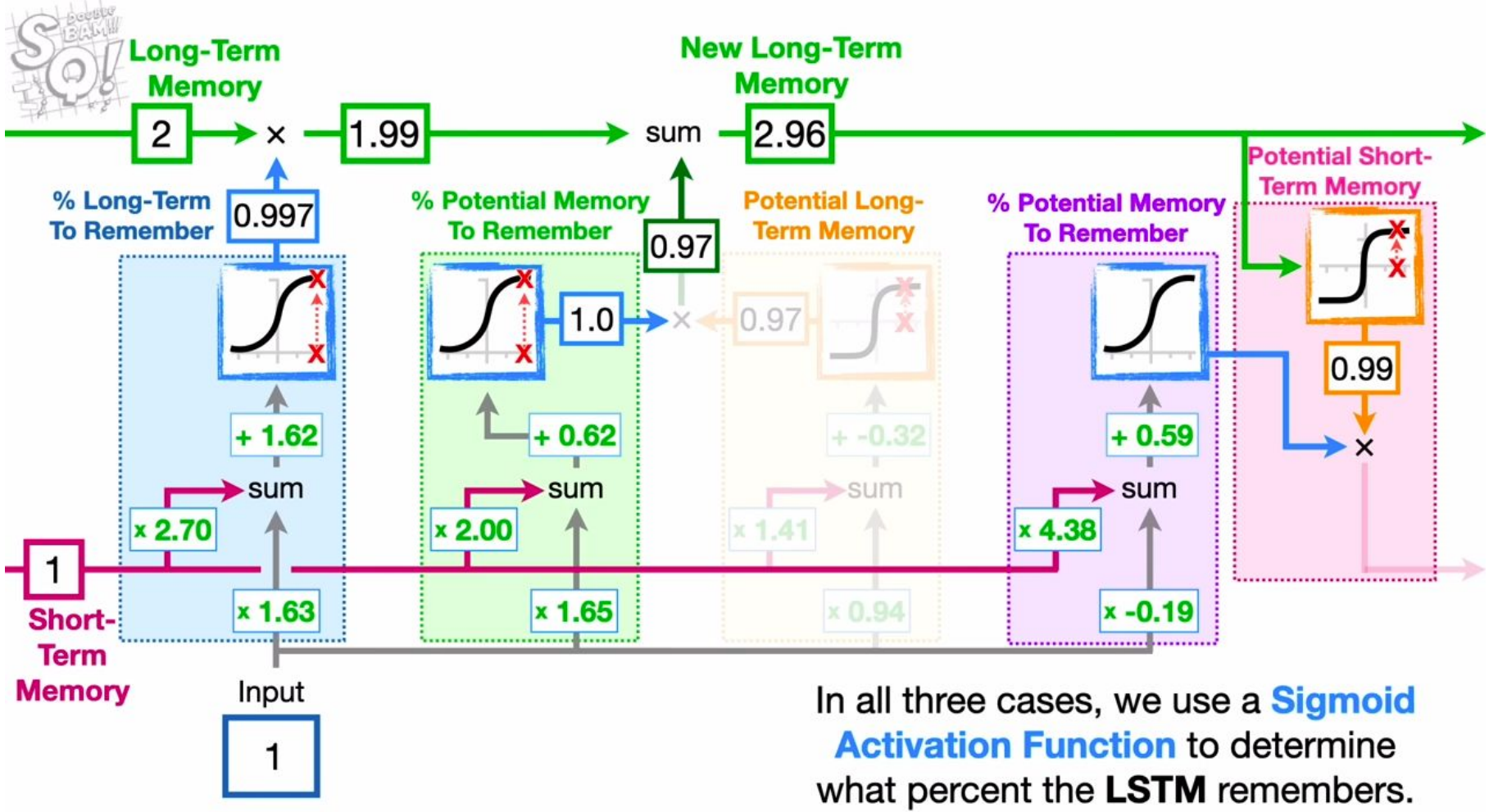

Long-Term Memory
2
×
1.99
sum
New Long-Term Memory
2.96
% Long-Term To Remember
0.997
% Potential Memory To Remember
0.97
Potential Long-Term Memory
1.0
×
0.97
+ 1.62
sum
× 2.70
× 1.63
+ 0.62
sum
× 2.00
× 1.65
+ -0.32
sum
× 1.41
× 0.94
sum
× 4.38
1
Short-Term Memory
Input
1
And after plugging 2.96 into the Tanh Activation Function, we get 0.99.

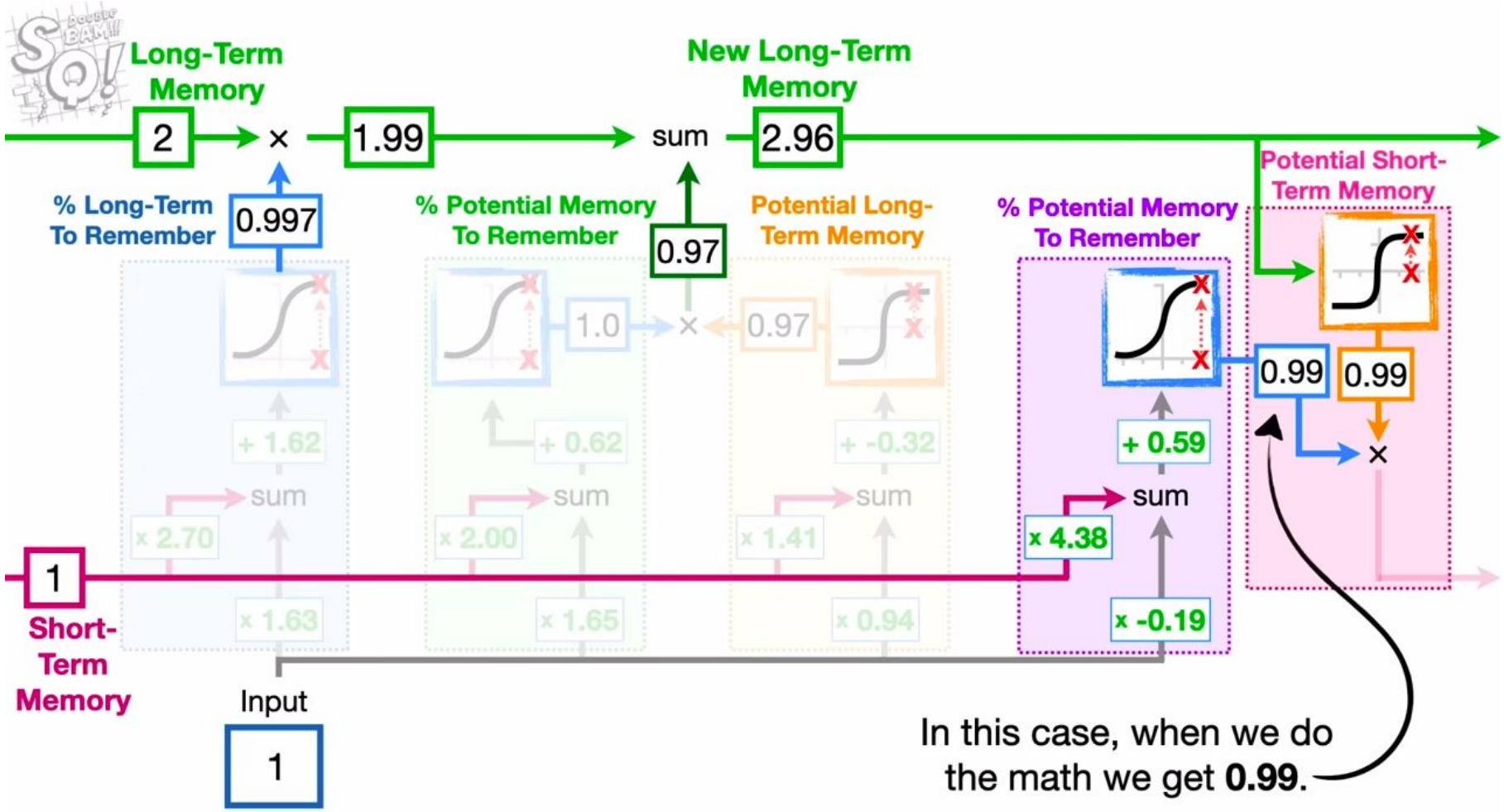

Long-Term Memory
2
×
1.99
sum
New Long-Term Memory
2.96
% Long-Term To Remember
0.997
% Potential Memory To Remember
0.97
Potential Long-Term Memory
Potential Short-Term Memory
0.99
1.0
0.97
+ 1.62
+ 0.62
+ -0.32
+ 0.59
sum
x 2.70
x 2.00
x 1.41
x 4.38
x 1.63
x 1.65
x 0.94
1
Short-Term Memory
Input
1
0.99 represents a Potential Short-Term Memory.

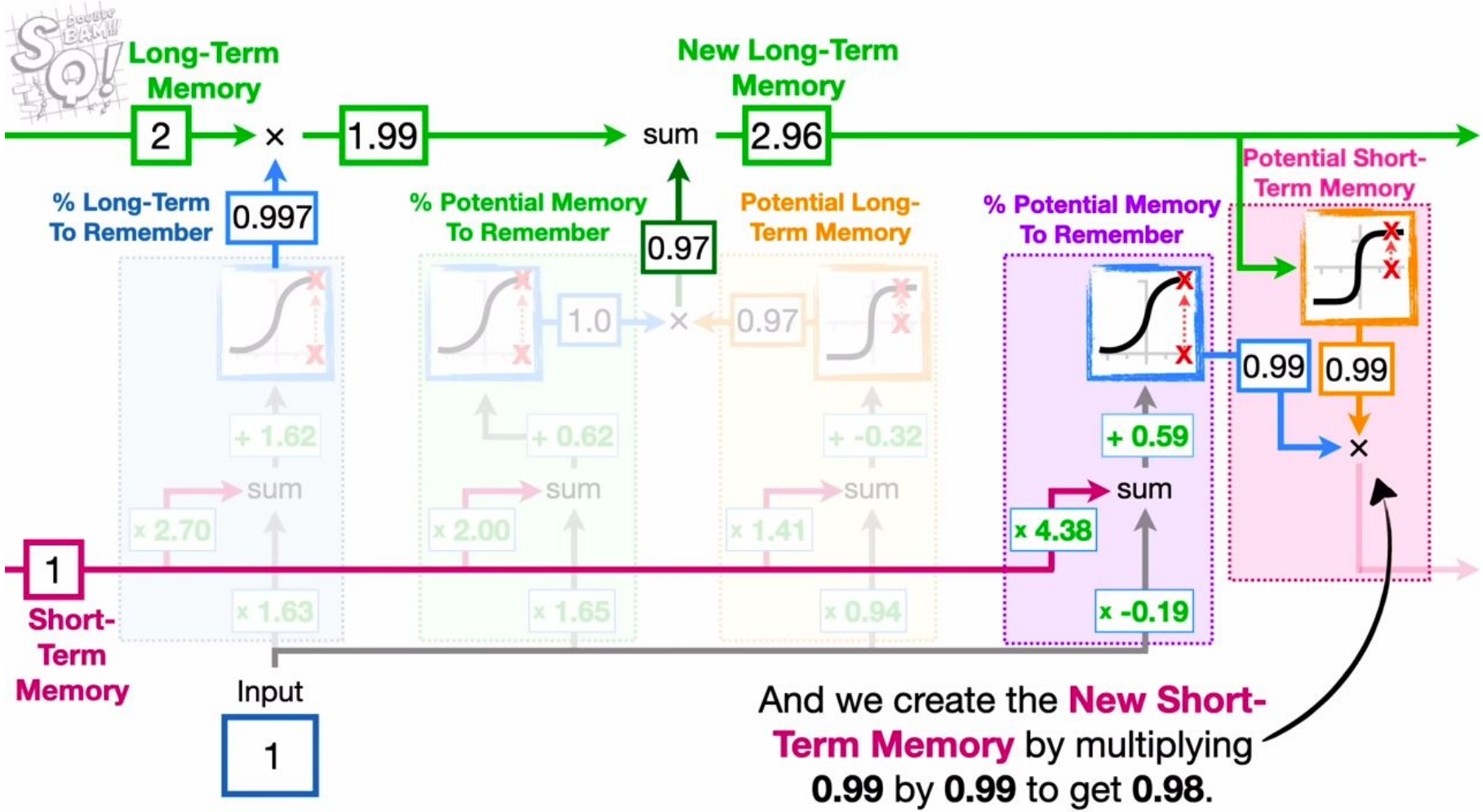

Long-Term Memory
2
×
1.99
sum
New Long-Term Memory
2.96
% Long-Term To Remember
0.997
% Potential Memory To Remember
0.97
Potential Long-Term Memory
Potential Short-Term Memory
0.99
1.0
0.97
+ 1.62
+ 0.62
+ -0.32
+ 0.59
sum
sum
sum
sum
x 2.70
x 2.00
x 1.41
x 4.38
x 1.63
x 1.65
x 0.94
x -0.19
1
Short-Term Memory
Input
1
Now the LSTM has to decide how much of this Potential Short-Term Memory to pass on…

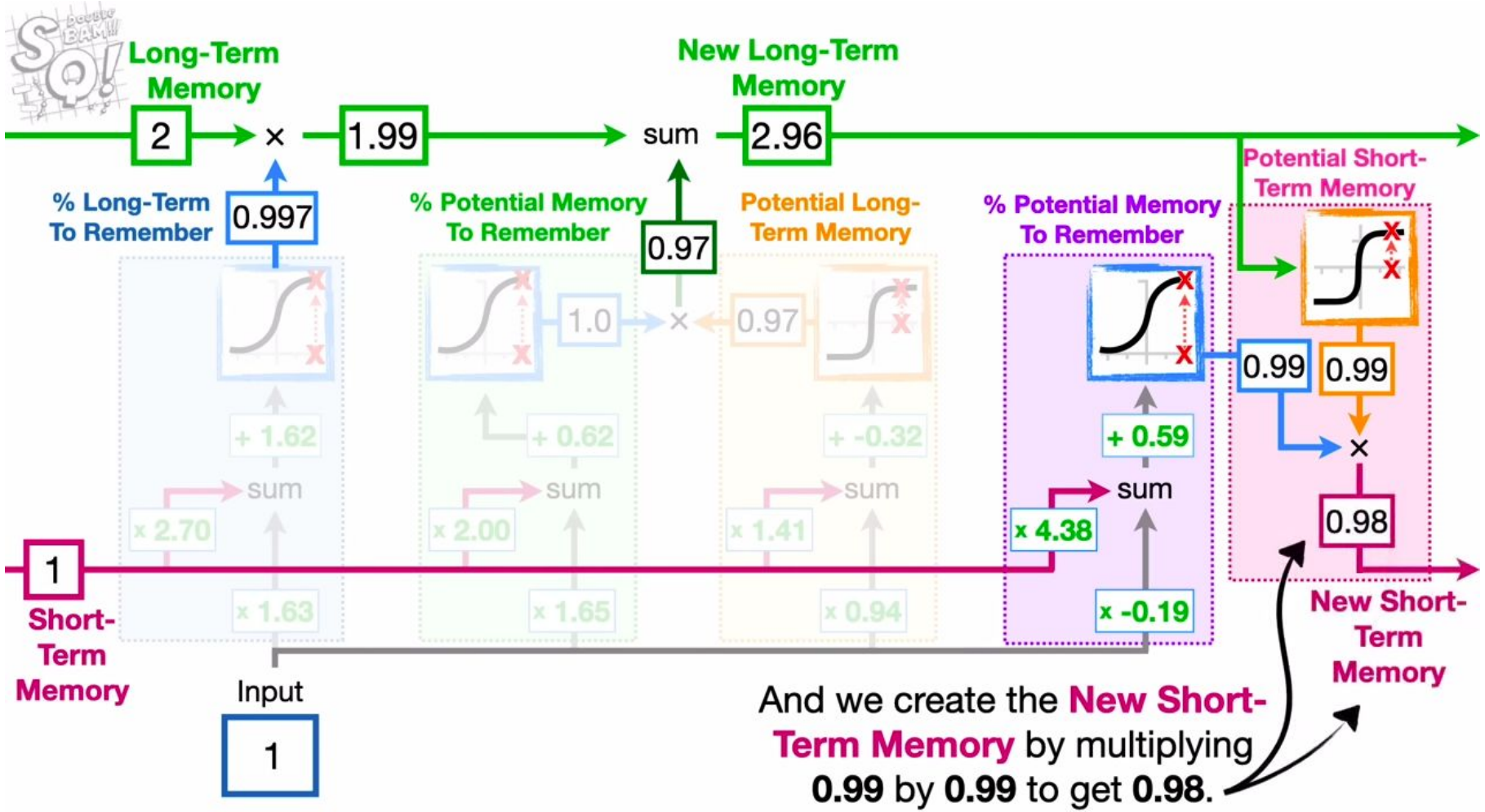

Long-Term Memory
2
×
1.99
sum
New Long-Term Memory
2.96
% Long-Term To Remember
0.997
% Potential Memory To Remember
0.97
Potential Long-Term Memory
% Potential Memory To Remember
Potential Short-Term Memory
0.99
1.0
0.97
+ 1.62
+ 0.62
+ -0.32
+ 0.59
sum
sum
sum
sum
x 2.70
x 2.00
x 1.41
x 4.38
x 1.63
x 1.65
x 0.94
x -0.19
1
Short-Term Memory
Input
1
…and this is done using the exact same method we used two times earlier.

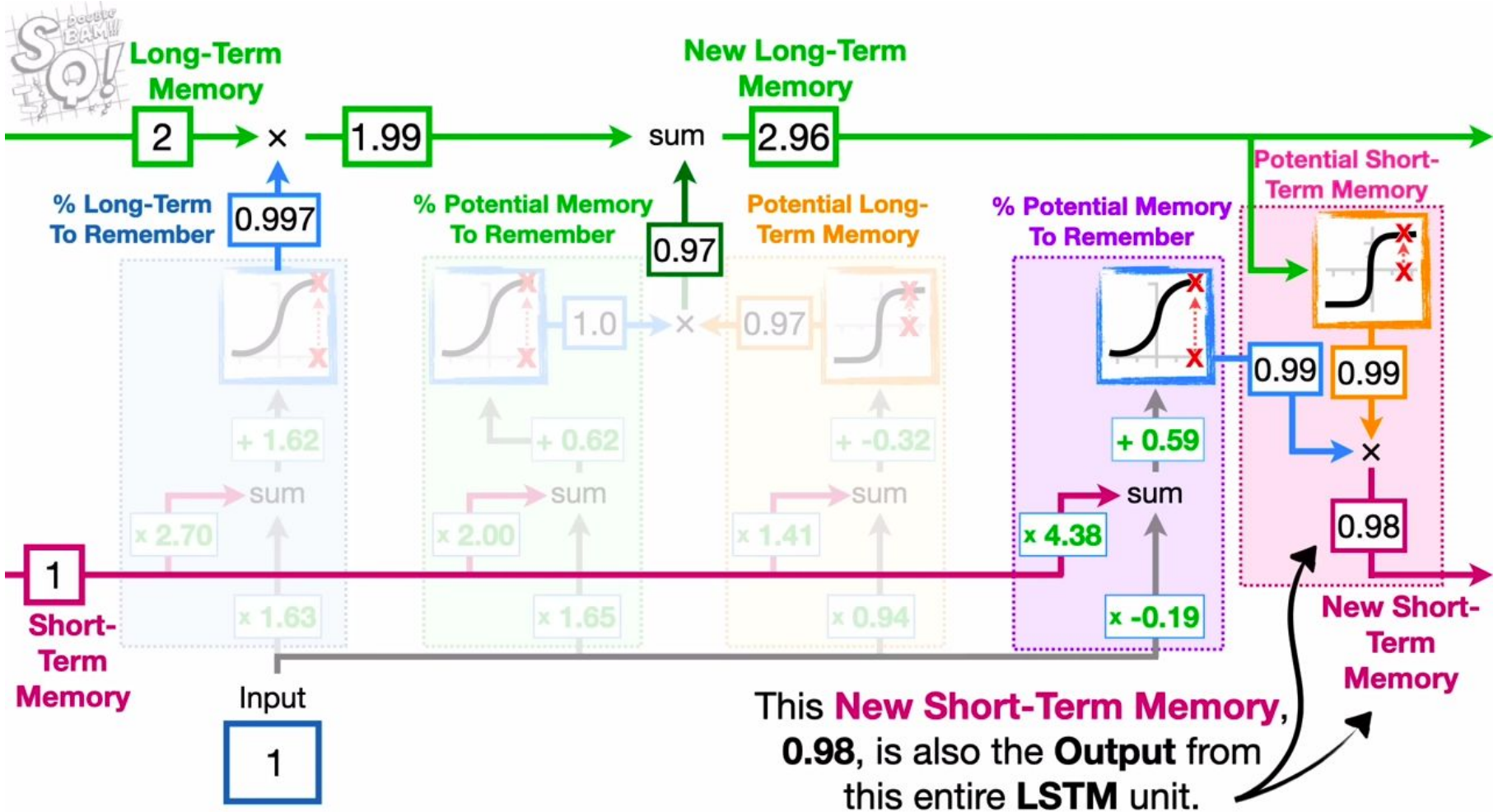

Long-Term Memory
2
×
1.99
sum
New Long-Term Memory
2.96
% Long-Term To Remember
0.997
% Potential Memory To Remember
0.97
Potential Long-Term Memory
% Potential Memory To Remember
Potential Short-Term Memory
1.0
0.97
0.99
+ 1.62
+ 0.62
+ -0.32
+ 0.59
sum
sum
sum
sum
× 2.70
× 2.00
× 1.41
× 4.38
1
Short-Term Memory
× 1.63
× 1.65
× 0.94
× -0.19
Input
1
…and when we determined what percentage of the Potential Long-Term Memory to remember.

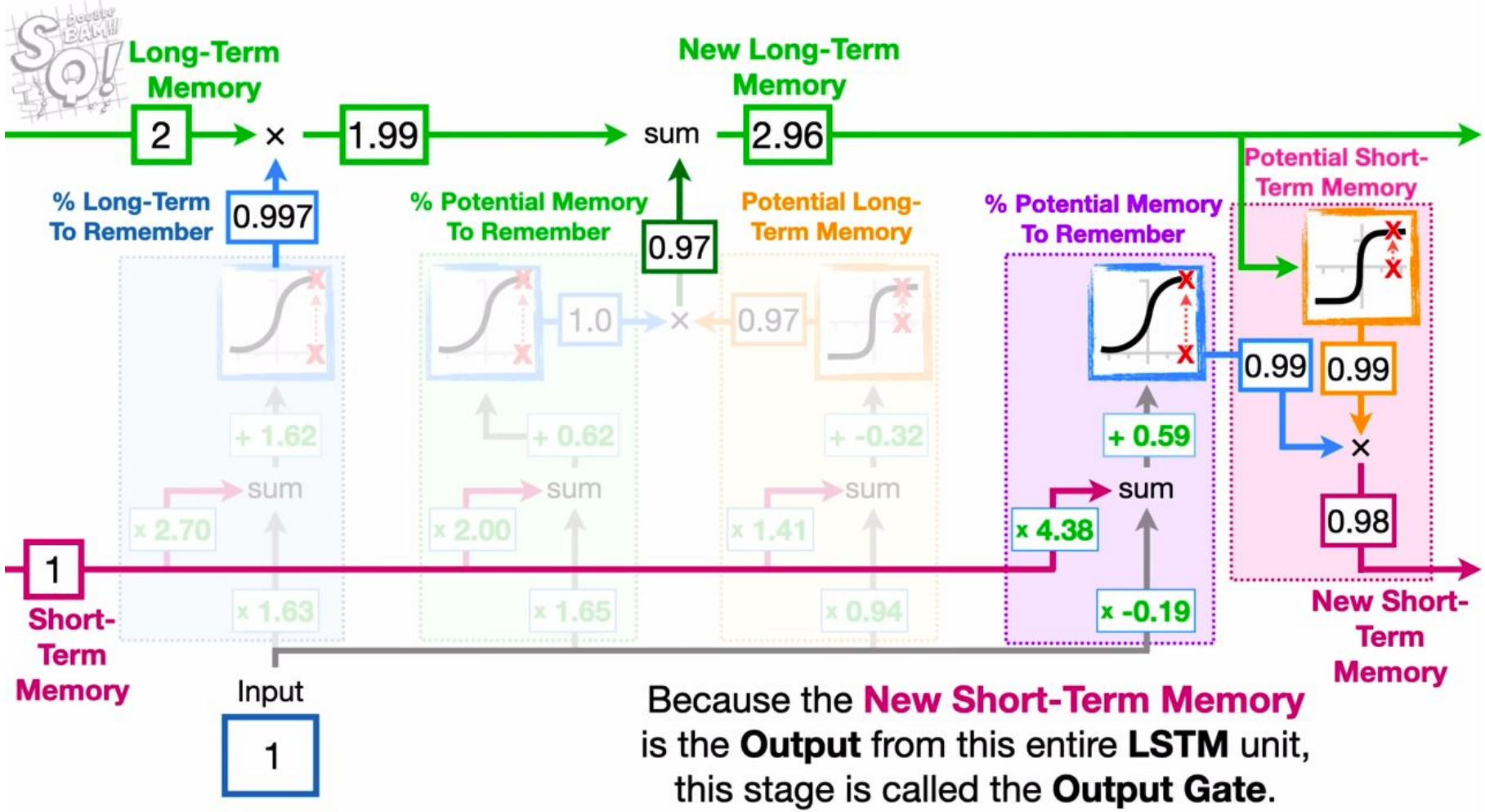

Long-Term Memory
2
×
1.99
sum
New Long-Term Memory
2.96
% Long-Term To Remember
0.997
% Potential Memory To Remember
0.97
Potential Long-Term Memory
% Potential Memory To Remember
Potential Short-Term Memory
1.0
0.97
0.99
+ 1.62
+ 0.62
+ -0.32
+ 0.59
sum
sum
sum
sum
× 2.70
× 2.00
× 1.41
× 4.38
1
Short-Term Memory
× 1.63
× 1.65
× 0.94
× -0.19
Input
1
In all three cases, we use a Sigmoid Activation Function to determine what percent the LSTM remembers.

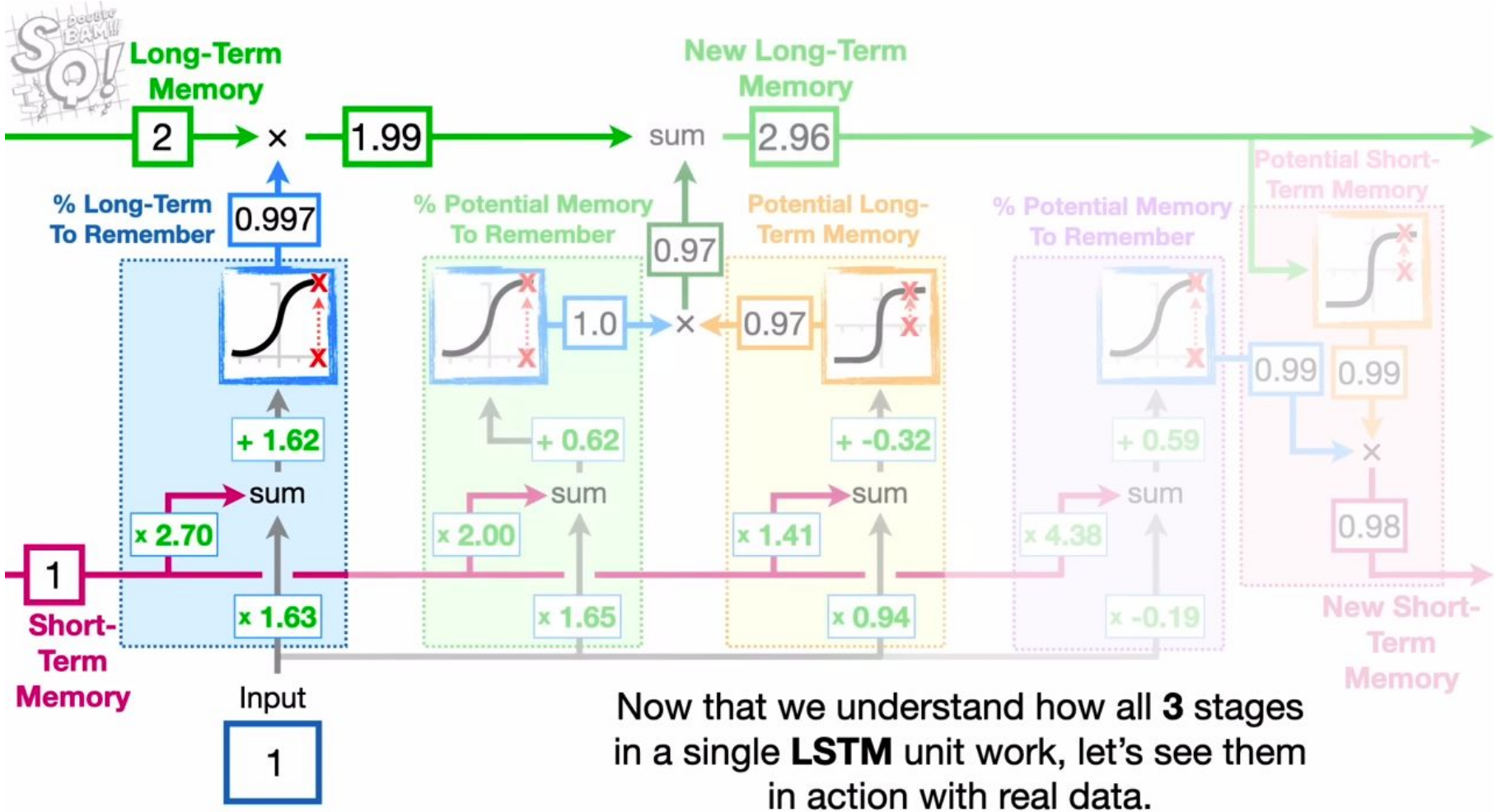

Long-Term Memory
2
×
1.99
sum
New Long-Term Memory
2.96
% Long-Term To Remember
0.997
% Potential Memory To Remember
0.97
Potential Long-Term Memory
% Potential Memory To Remember
Potential Short-Term Memory
1.0
×
0.97
0.99
0.99
+ 1.62
+ 0.62
+ -0.32
+ 0.59
sum
sum
sum
sum
x 2.70
x 2.00
x 1.41
x 4.38
x -0.19
1
Short-Term Memory
x 1.63
x 1.65
x 0.94
Input
1
In this case, when we do the math we get 0.99.

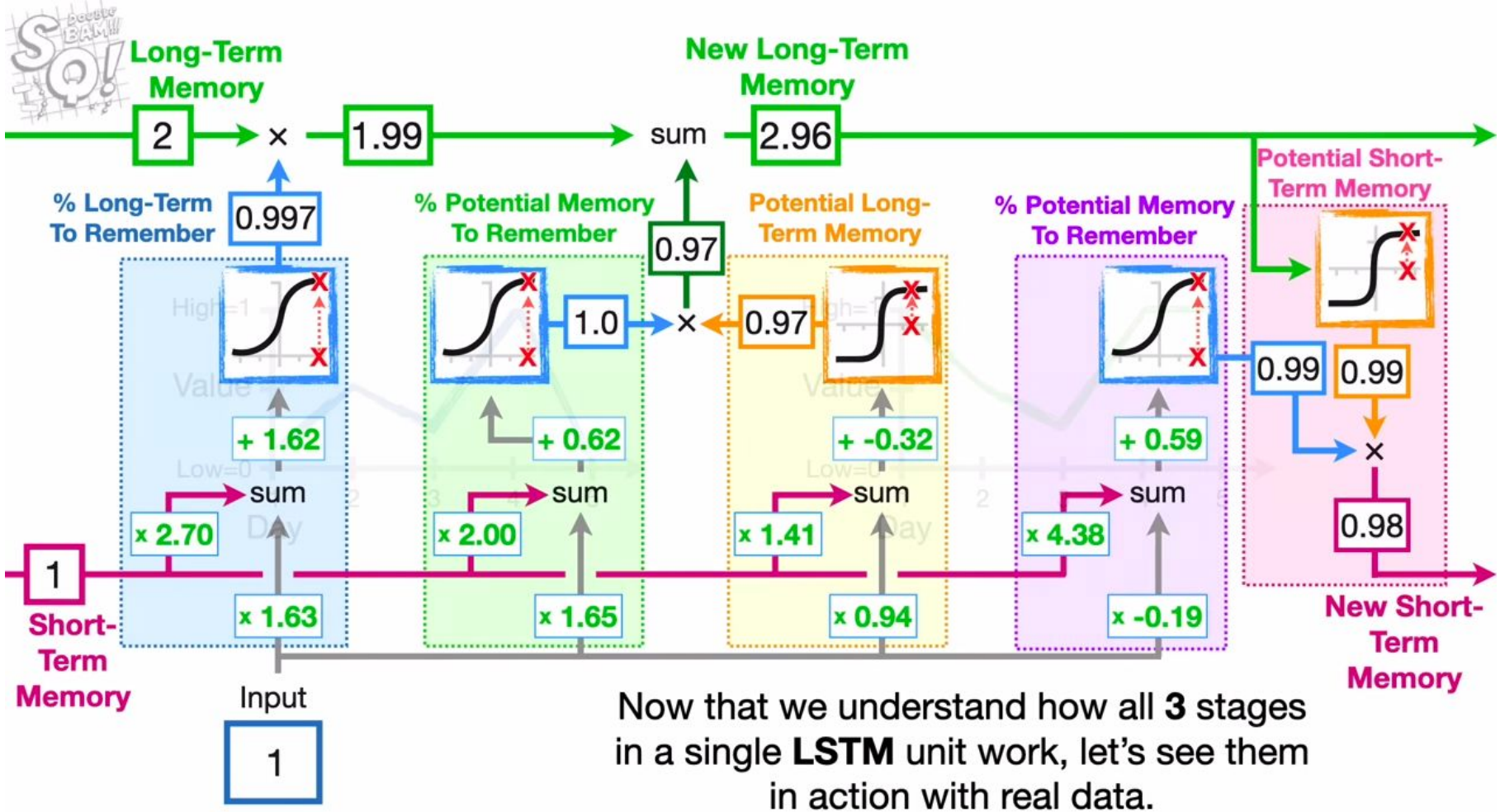

Long-Term Memory
2
×
1.99
sum
New Long-Term Memory
2.96
% Long-Term To Remember
0.997
% Potential Memory To Remember
0.97
Potential Long-Term Memory
% Potential Memory To Remember
Potential Short-Term Memory
1.0
×
0.97
0.99
0.99
+ 1.62
+ 0.62
+ -0.32
+ 0.59
sum
sum
sum
sum
x 2.70
x 2.00
x 1.41
x 4.38
x 1.63
x 1.65
x 0.94
x -0.19
1
Short-Term Memory
Input
1
And we create the New Short-Term Memory by multiplying 0.99 by 0.99 to get 0.98.


Long-Term Memory
2
×
1.99
sum
New Long-Term Memory
2.96
% Long-Term To Remember
0.997
% Potential Memory To Remember
0.97
Potential Long-Term Memory
% Potential Memory To Remember
Potential Short-Term Memory
1.0
×
0.97
+ 1.62
sum
× 2.70
× 1.63
+ 0.62
sum
× 2.00
× 1.65
+ -0.32
sum
× 1.41
× 0.94
+ 0.59
sum
× 4.38
× -0.19
0.99
0.99
×
0.98
1
Short-Term Memory
Input
1
New Short-Term Memory
And we create the New Short-Term Memory by multiplying 0.99 by 0.99 to get 0.98.


Long-Term Memory
2
×
1.99
sum
New Long-Term Memory
2.96
% Long-Term To Remember
0.997
% Potential Memory To Remember
0.97
Potential Long-Term Memory
% Potential Memory To Remember
Potential Short-Term Memory
1.0
×
0.97
+ 1.62
+ 0.62
+ -0.32
+ 0.59
sum
sum
sum
x 2.70
x 2.00
x 1.41
x 4.38
x -0.19
x 1.63
x 1.65
x 0.94
0.99
0.99
×
0.98
1
Short-Term Memory
Input
1
New Short-Term Memory
This New Short-Term Memory, 0.98, is also the Output from this entire LSTM unit.

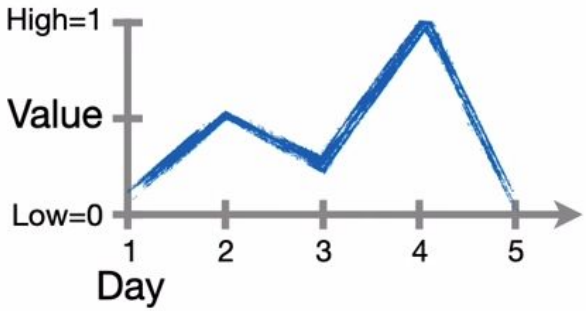

Long-Term Memory
2
×
1.99
sum
New Long-Term Memory
2.96
% Long-Term To Remember
0.997
% Potential Memory To Remember
0.97
Potential Long-Term Memory
% Potential Memory To Remember
Potential Short-Term Memory
1.0
×
0.97
+ 1.62
+ 0.62
+ -0.32
+ 0.59
sum
sum
sum
x 2.70
x 2.00
x 1.41
x 4.38
x -0.19
x 1.63
x 1.65
x 0.94
0.99
0.99
×
0.98
1
Short-Term Memory
Input
1
New Short-Term Memory
Because the New Short-Term Memory is the Output from this entire LSTM unit, this stage is called the Output Gate.

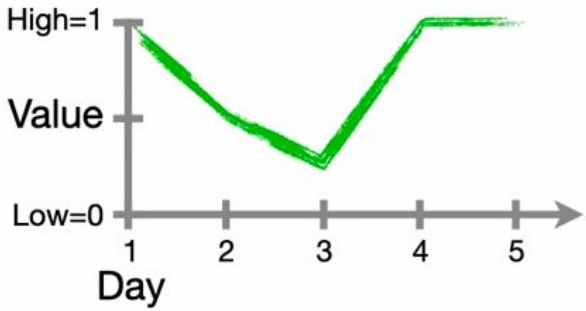

Long-Term Memory
2
×
1.99
New Long-Term Memory
sum
2.96
% Long-Term To Remember
0.997
+ 1.62
sum
× 2.70
1
Short-Term Memory
× 1.63
Input
1
% Potential Memory To Remember
1.0
+ 0.62
sum
× 2.00
× 1.65
0.97
Potential Long-Term Memory
0.97
+ -0.32
sum
× 1.41
× 0.94
% Potential Memory To Remember
+ 0.59
sum
× 4.38
× -0.19
Potential Short-Term Memory
0.99
0.99
×
0.98
New Short-Term Memory
Now that we understand how all 3 stages in a single LSTM unit work, let's see them in action with real data.



Now that we understand how all **3** stages in a single **LSTM** unit work, let's see them in action with real data.

Here we have stock prices for two companies, **Company A** and **Company B**.

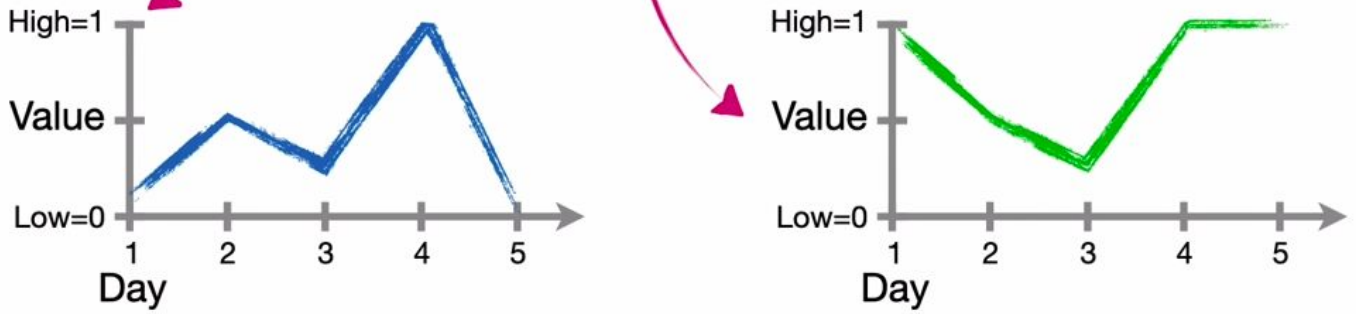




Company A =

Company B =

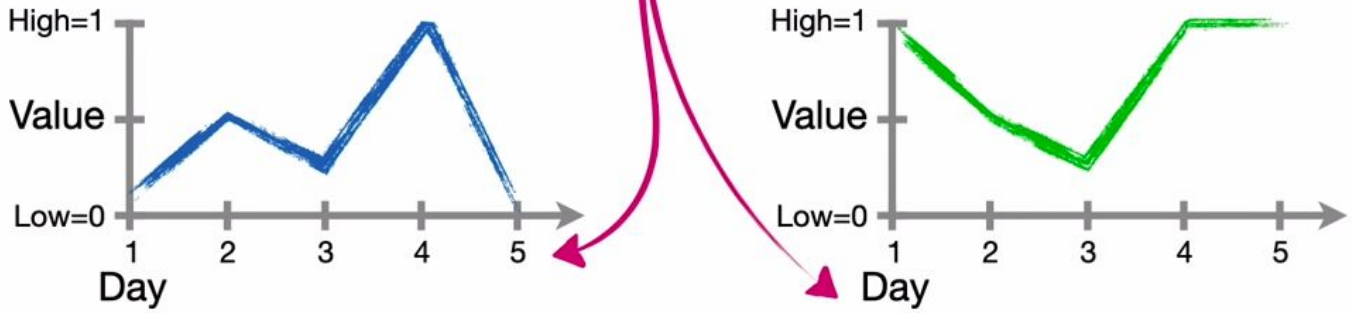
On the y-axis, we have the stock value…



Company A =

Company B =

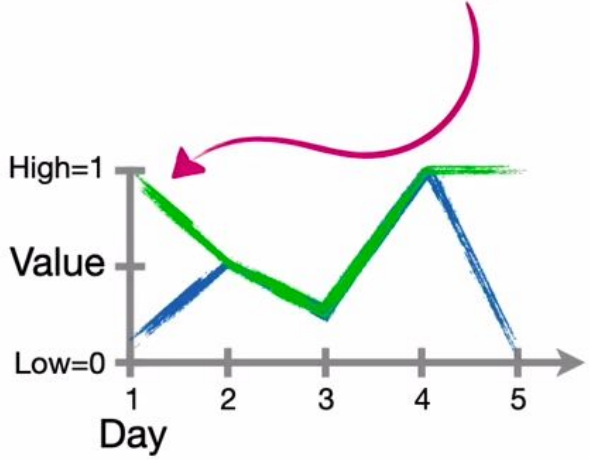
…and on the x-axis, we have
the day the value was recorded.



Company A =

Company B =

…we see that the only differences occur on **Day 1**…

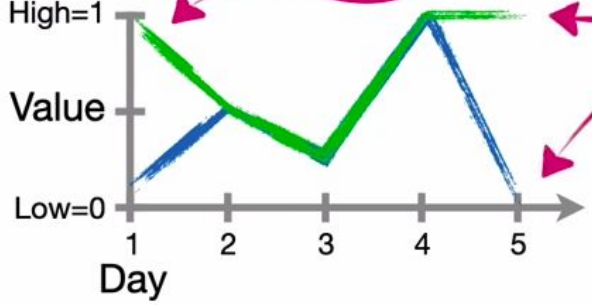


Company A =

Company B =

…we see that the only differences occur on **Day 1**…

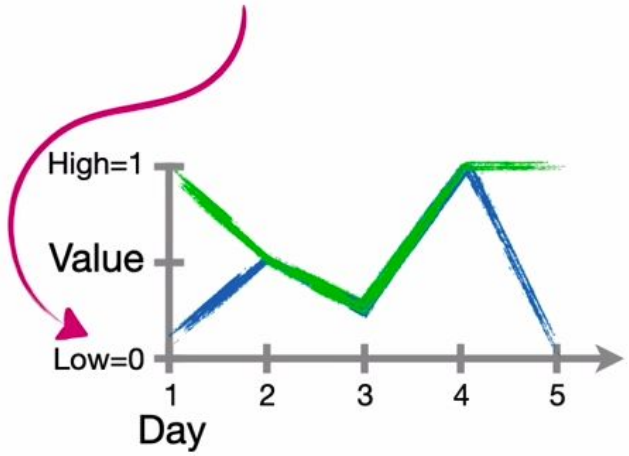
…and on **Day 5**.



Company A =

Company B =

On **Day 1**, **Company A** is at **0**…

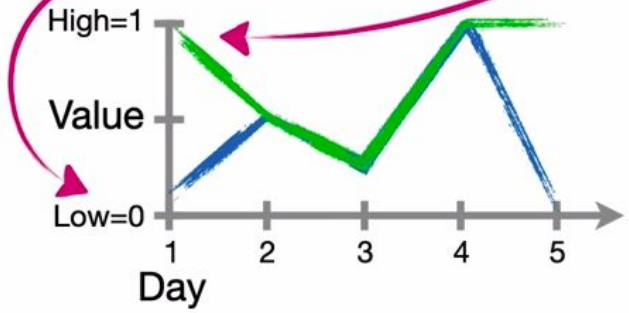


Company A =

Company B =

On **Day 1**, **Company A** is at **0**…

…and **Company B** is at **1**.

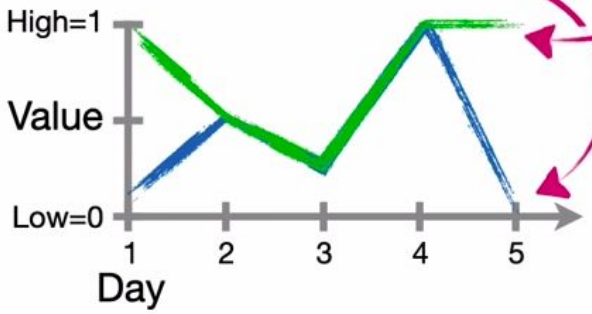


Company A =

Company B =

And on **Day 5**, **Company A** is returns to **0**…

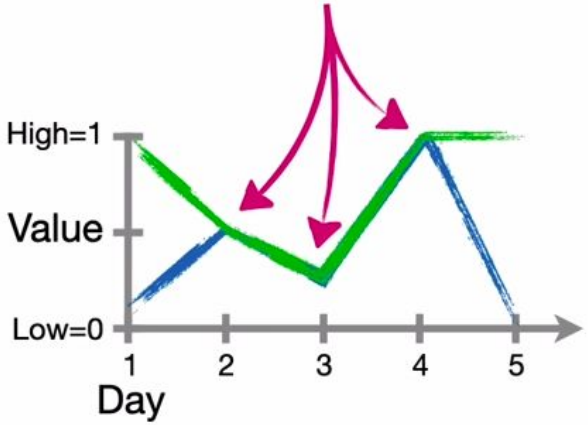
…and **Company B** returns to **1**.



Company A =

Company B =

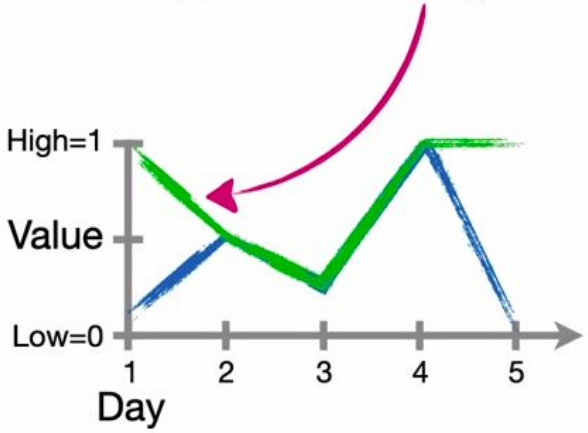
On all of the other days, **Days 2**, **3** and **4**, both companies have the exact same values.



Company A =

Company B =

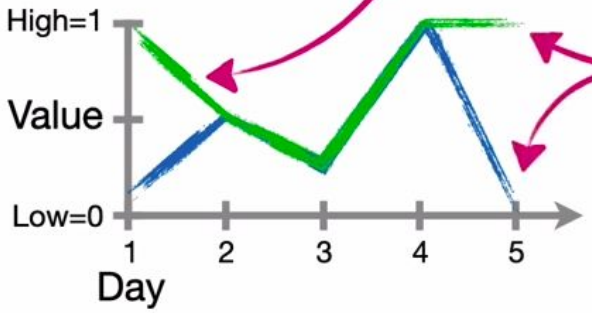
Given this sequential data, we want the **LSTM** to remember what happened on **Day 1**…



Company A =

Company B =

Given this sequential data, we want the **LSTM** to remember what happened on **Day 1**…

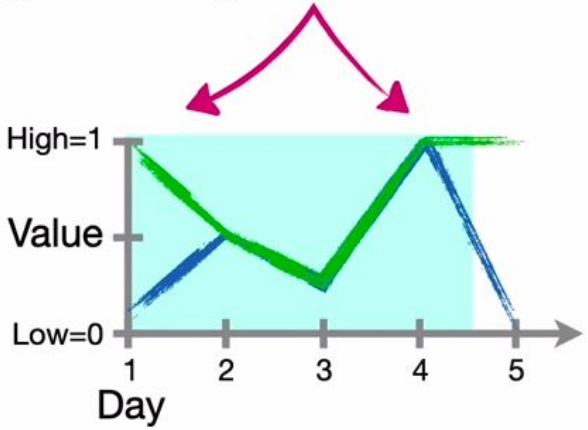


…so it can correctly predict what will happen on **Day 5**.

Company A =

Company B =

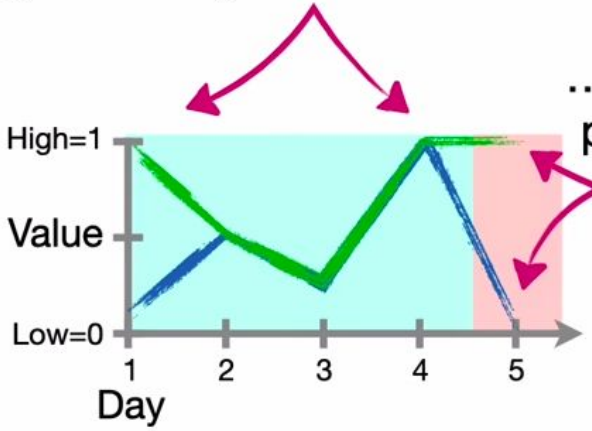
In other words, we're going to sequentially run the data from **Days 1** through **4** through an unrolled **LSTM**…



Company A =

Company B =

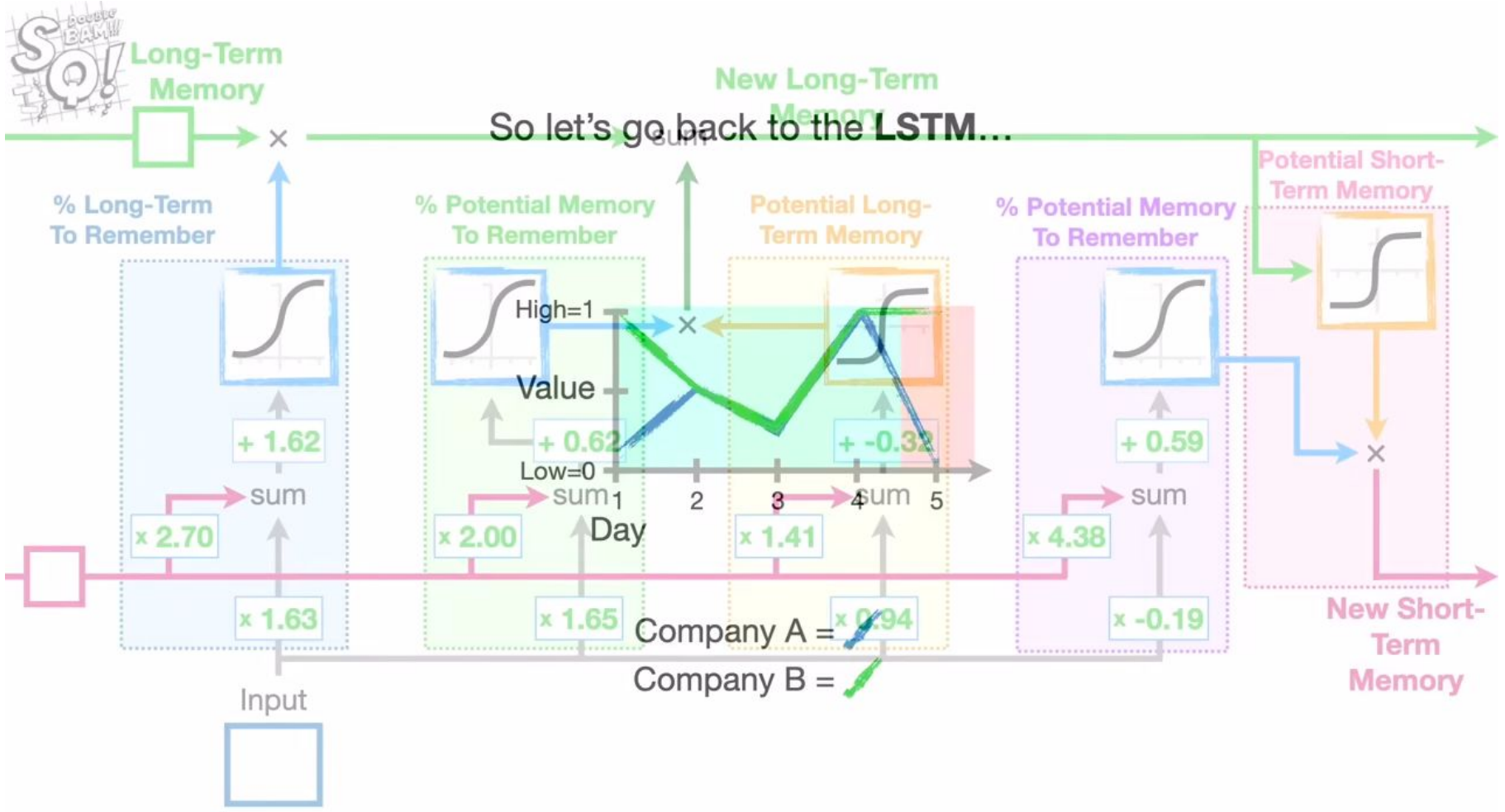
In other words, we're going to sequentially run the data from **Days 1** through **4** through an unrolled **LSTM**…

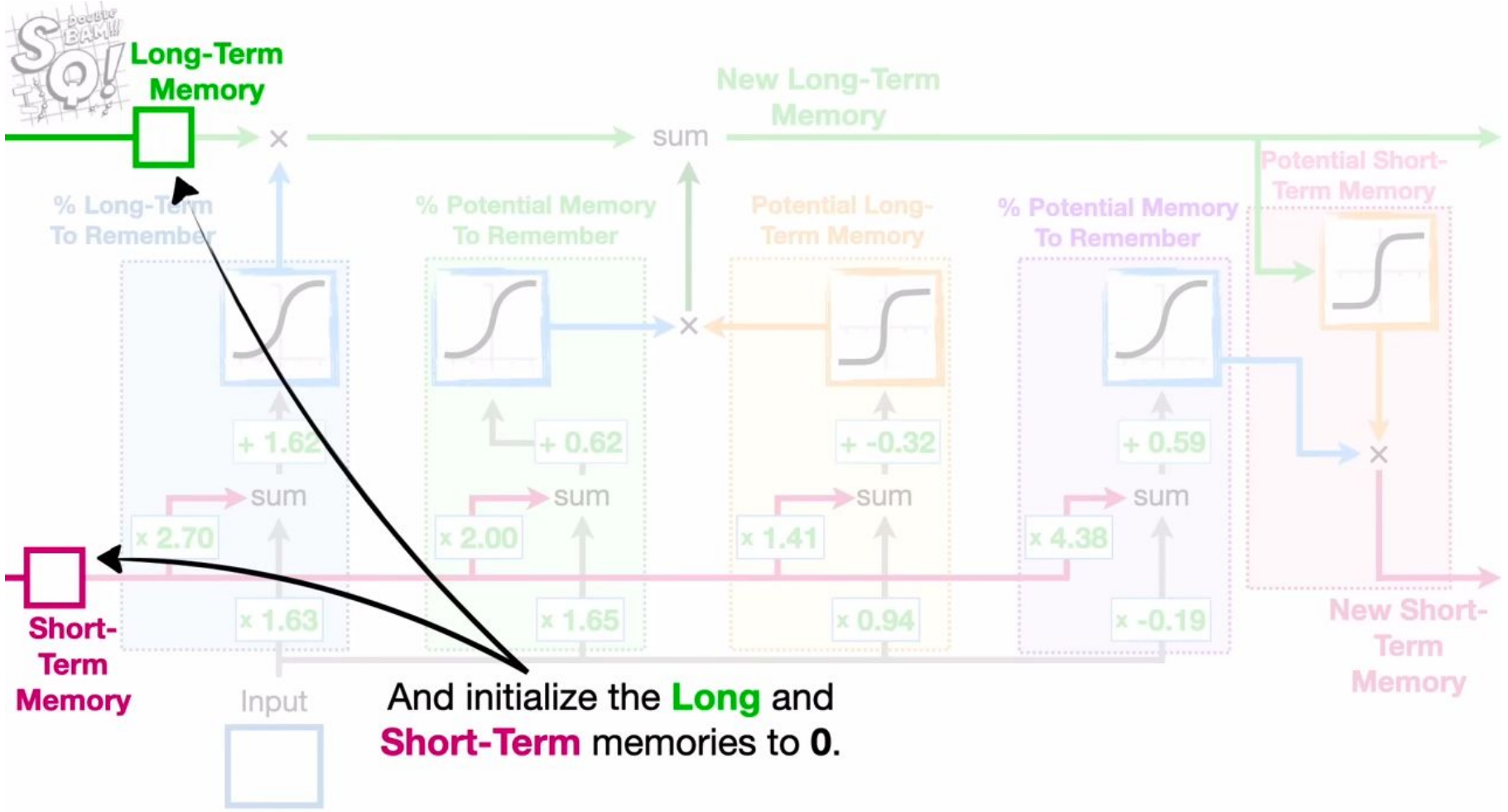


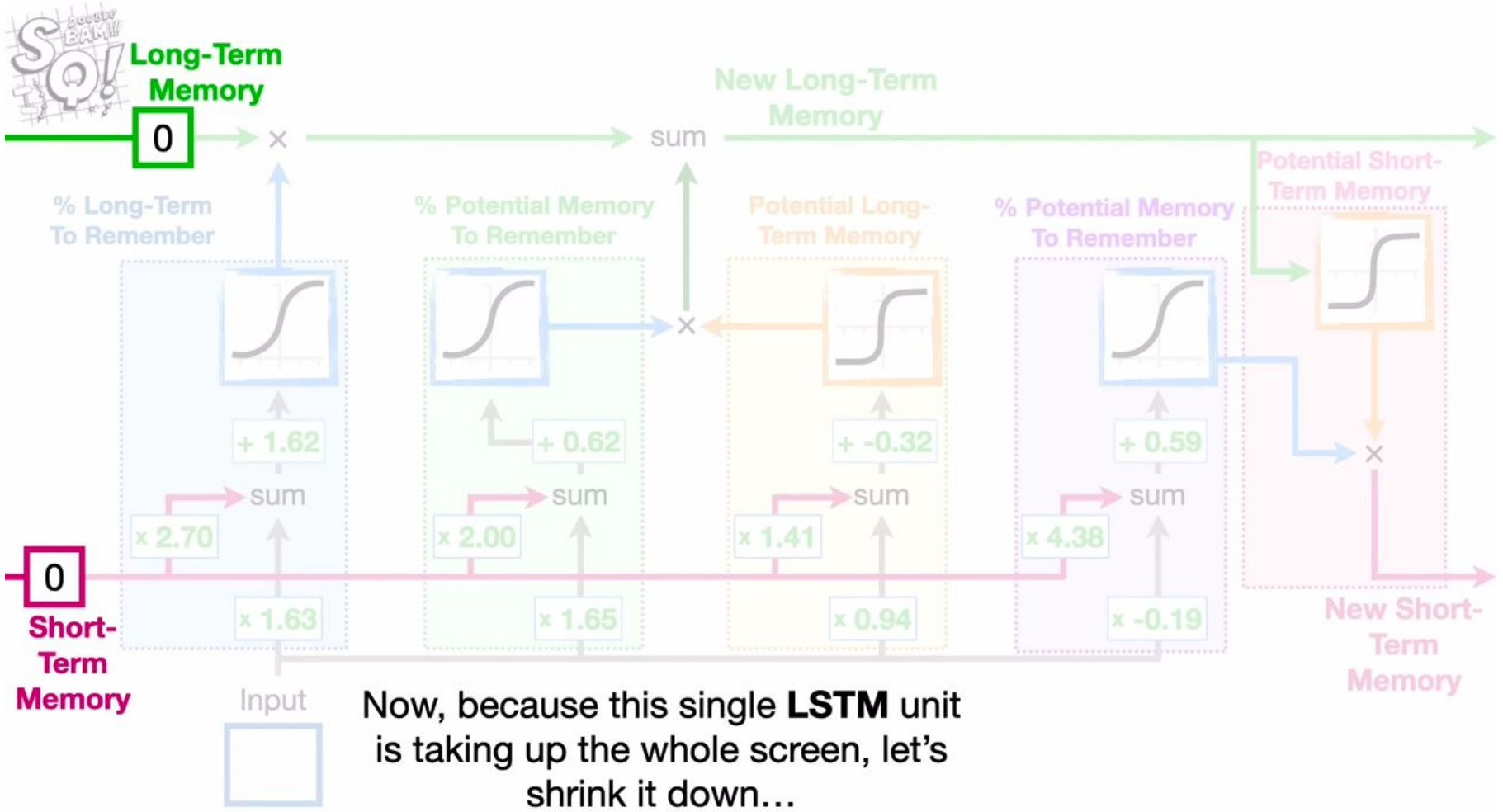
…and see if it can correctly predict the values for **Day 5** for both companies.

Company A =

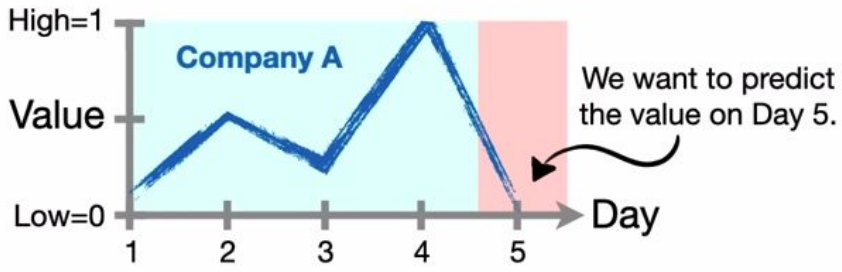
Company B =

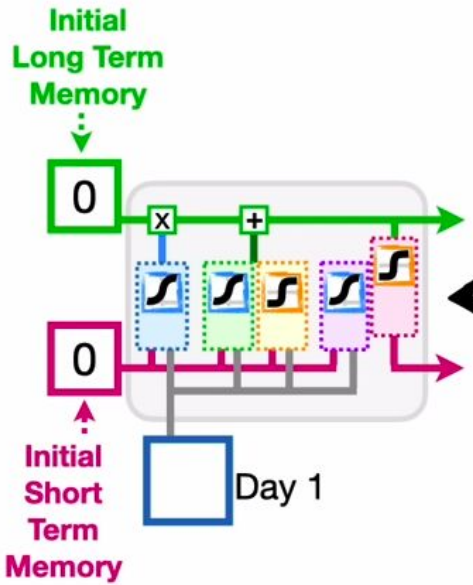

So let's go back to the LSTM…
Long-Term Memory
New Long-Term Memory
Potential Short-Term Memory
% Long-Term To Remember
% Potential Memory To Remember
Potential Long-Term Memory
% Potential Memory To Remember
High=1
Value
Low=0
1
2
3
4
5
Day
+ 1.62
+ 0.62
+ -0.32
+ 0.59
sum
sum
sum
sum
x 2.70
x 2.00
x 1.41
x 4.38
x 1.63
x 1.65
x 0.94
x -0.19
Company A =
Company B =
Input
New Short-Term Memory


Long-Term Memory
New Long-Term Memory
sum
Potential Short-Term Memory
% Long-Term To Remember
% Potential Memory To Remember
Potential Long-Term Memory
% Potential Memory To Remember
+ 1.62
+ 0.62
+ -0.32
+ 0.59
sum
sum
sum
sum
x 2.70
x 2.00
x 1.41
x 4.38
x 1.63
x 1.65
x 0.94
x -0.19
Short-Term Memory
New Short-Term Memory
Input
And initialize the Long and Short-Term memories to 0.

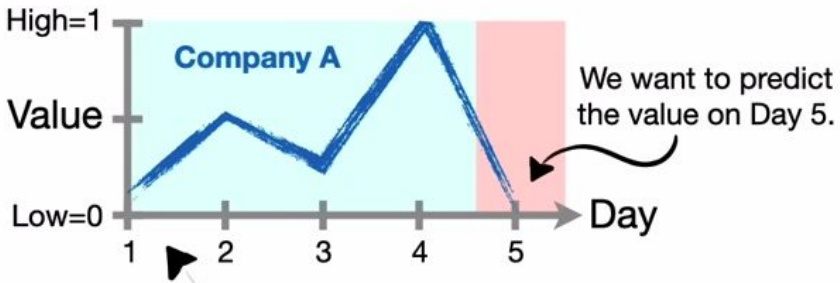


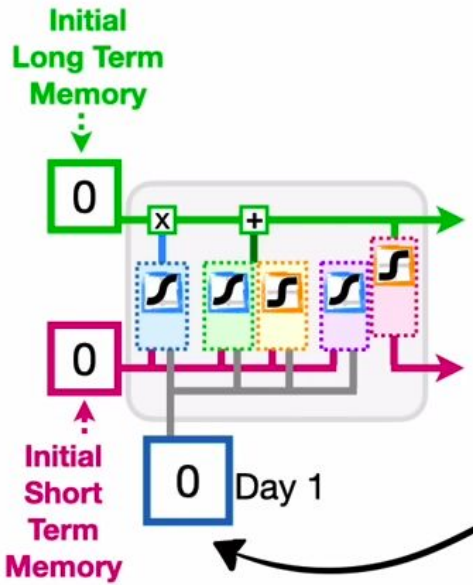
Now, because this single **LSTM** unit is taking up the whole screen, let's shrink it down…



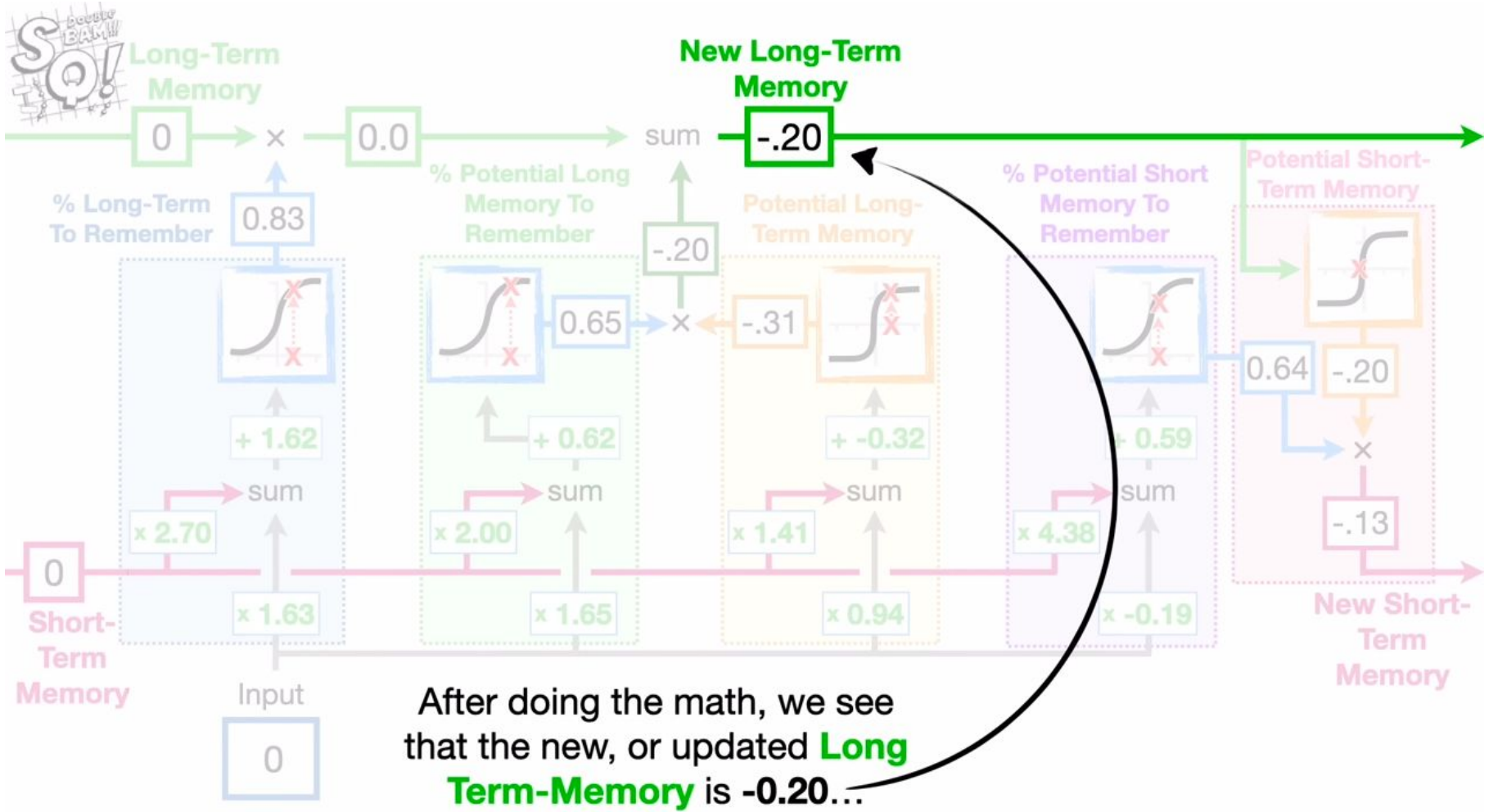


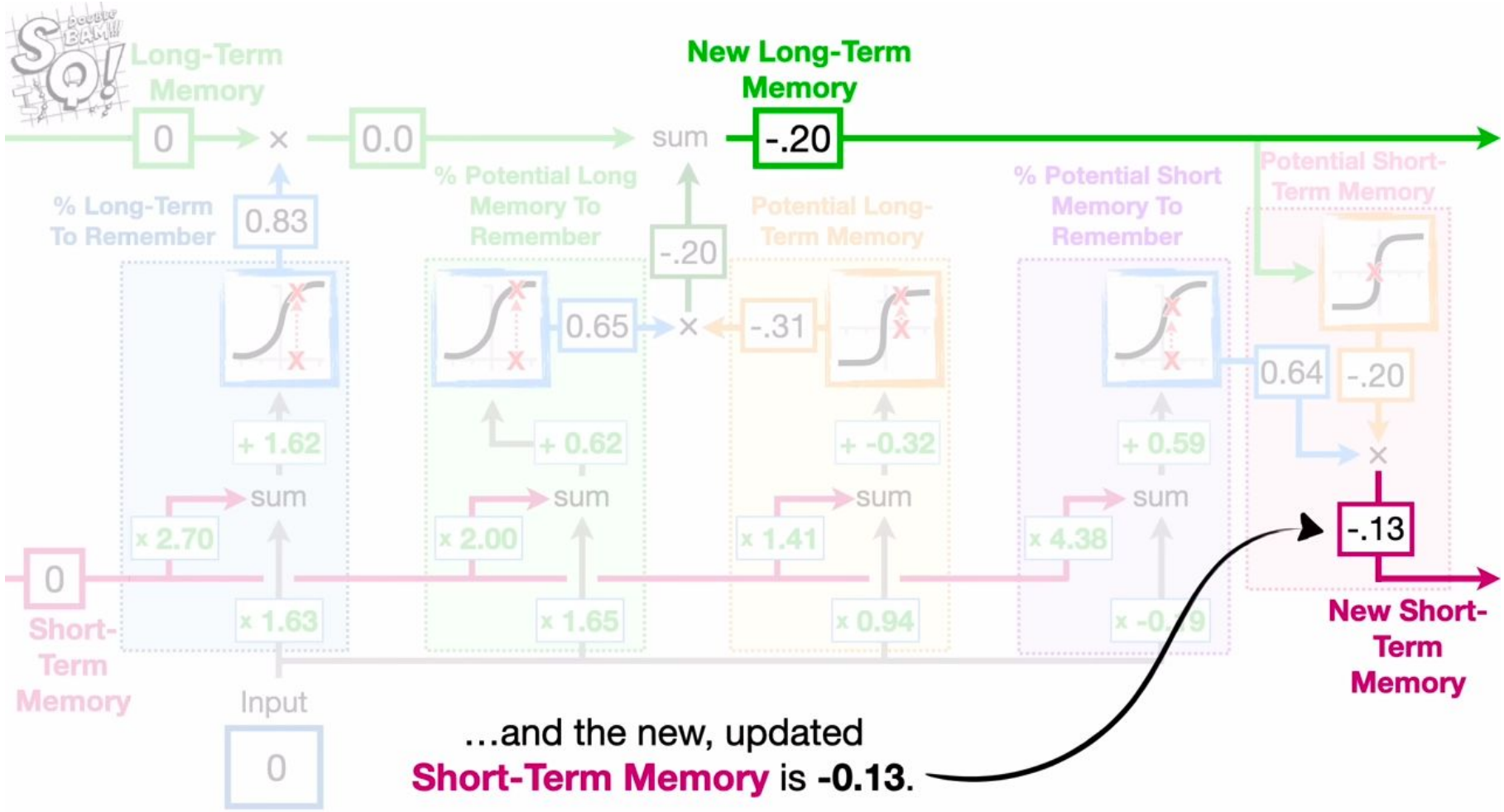
Now, if we want to sequentially run **Company A**'s values from **Days 1** through **4** through this **LSTM**…

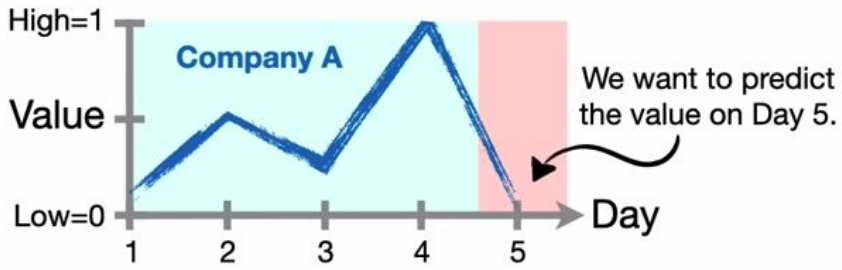


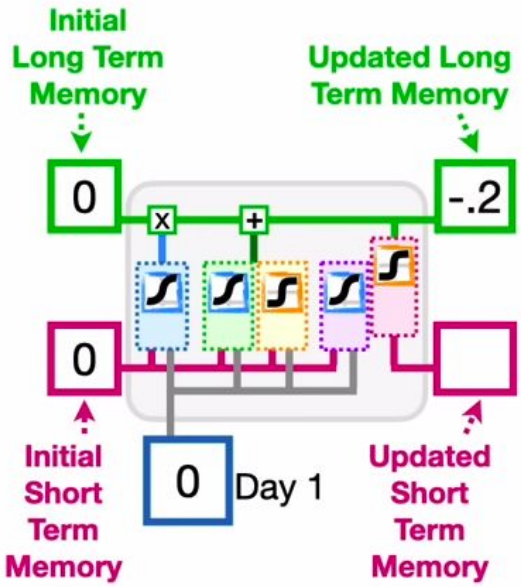


…then we'll start by plugging the value for **Day 1**, which is **0**, into the **Input**.

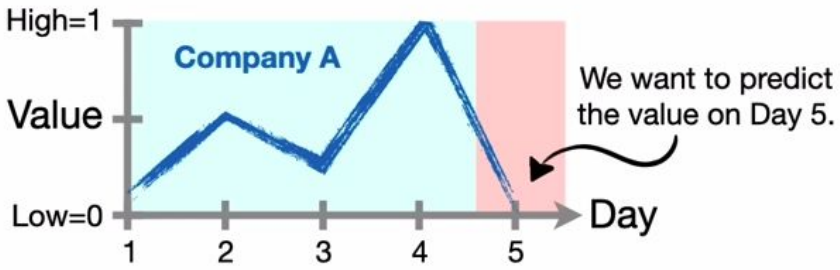

Long-Term Memory
0
×
0.0
sum
New Long-Term Memory
-.20
% Long-Term To Remember
0.83
% Potential Long Memory To Remember
0.65
-.20
×
Potential Long-Term Memory
-.31
% Potential Short Memory To Remember
Potential Short-Term Memory
0.64
-.20
×
-.13
New Short-Term Memory
+ 1.62
+ 0.62
+ -0.32
+ 0.59
sum
sum
sum
sum
× 2.70
× 2.00
× 1.41
× 4.38
0
Short-Term Memory
× 1.63
× 1.65
× 0.94
× -0.19
Input
0
After doing the math, we see that the new, or updated Long Term-Memory is -0.20…

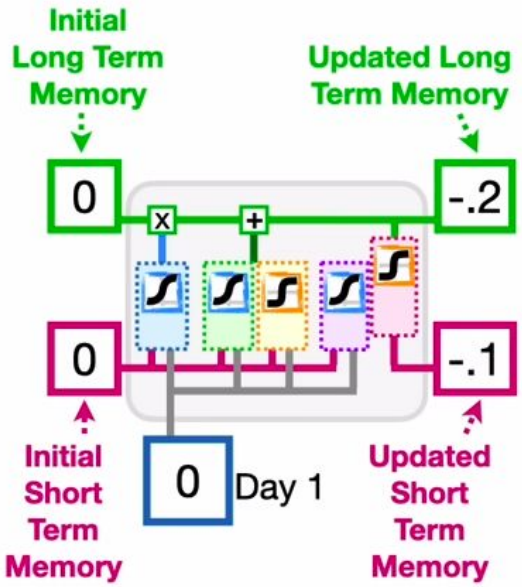

Long-Term Memory
0
×
0.0
sum
New Long-Term Memory
-.20
% Long-Term To Remember
0.83
% Potential Long Memory To Remember
0.65
-.20
Potential Long-Term Memory
-.31
% Potential Short Memory To Remember
Potential Short-Term Memory
0.64
-.20
+ 1.62
+ 0.62
+ -0.32
+ 0.59
sum
sum
sum
sum
× 2.70
× 2.00
× 1.41
× 4.38
0
Short-Term Memory
× 1.63
× 1.65
× 0.94
-.13
New Short-Term Memory
Input
0
…and the new, updated
Short-Term Memory is -0.13.



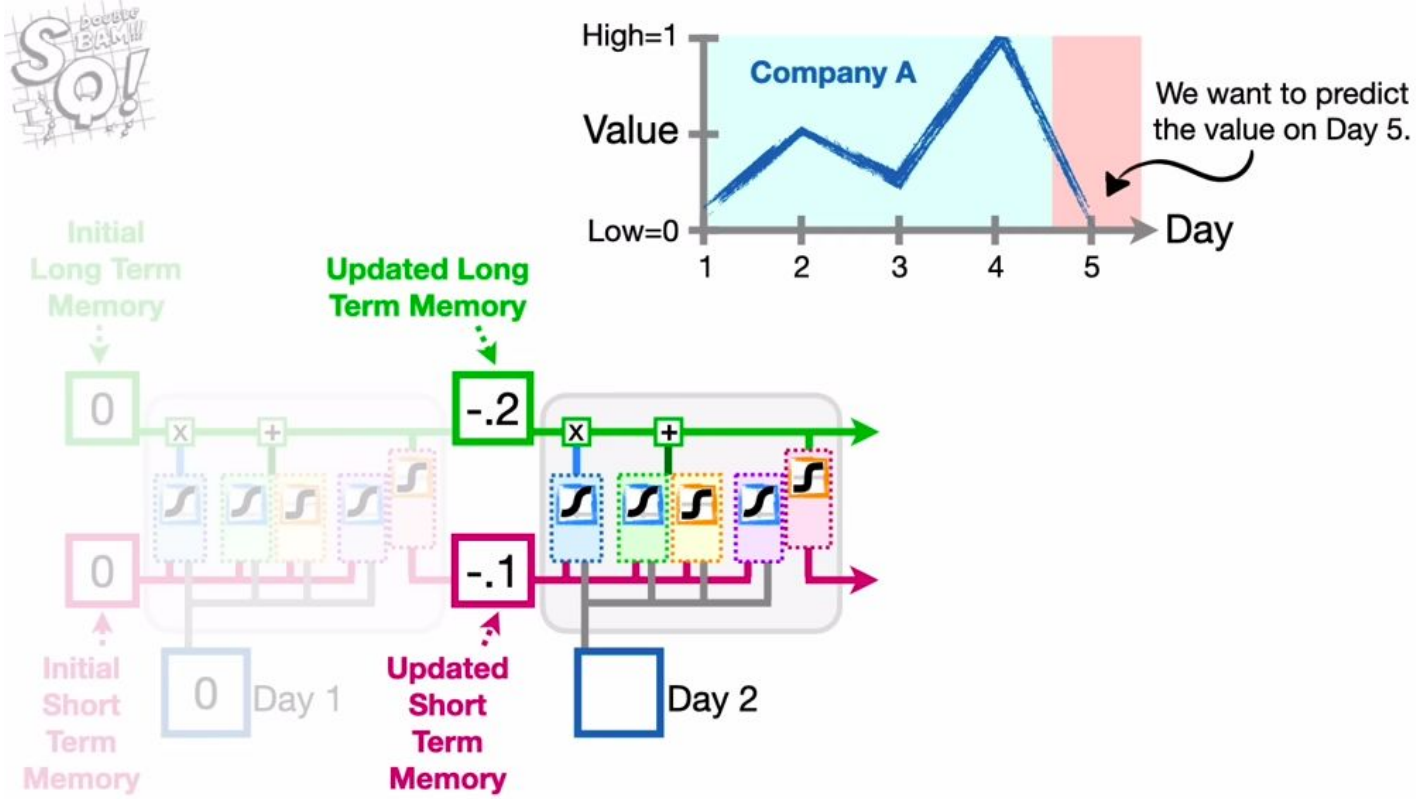


…so we plug in **-0.2** for the updated **Long-Term Memory**…

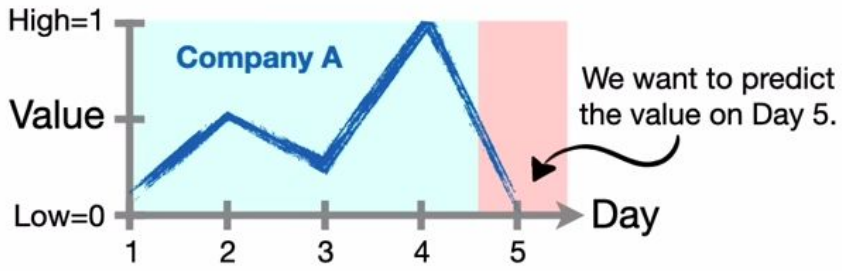


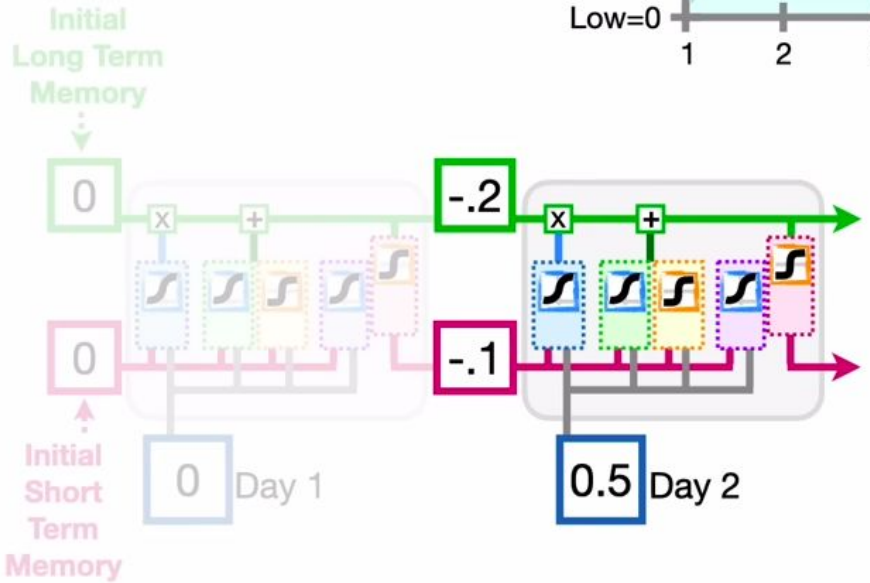


…and **-0.1** (rounded) for the updated **Short-Term Memory**.

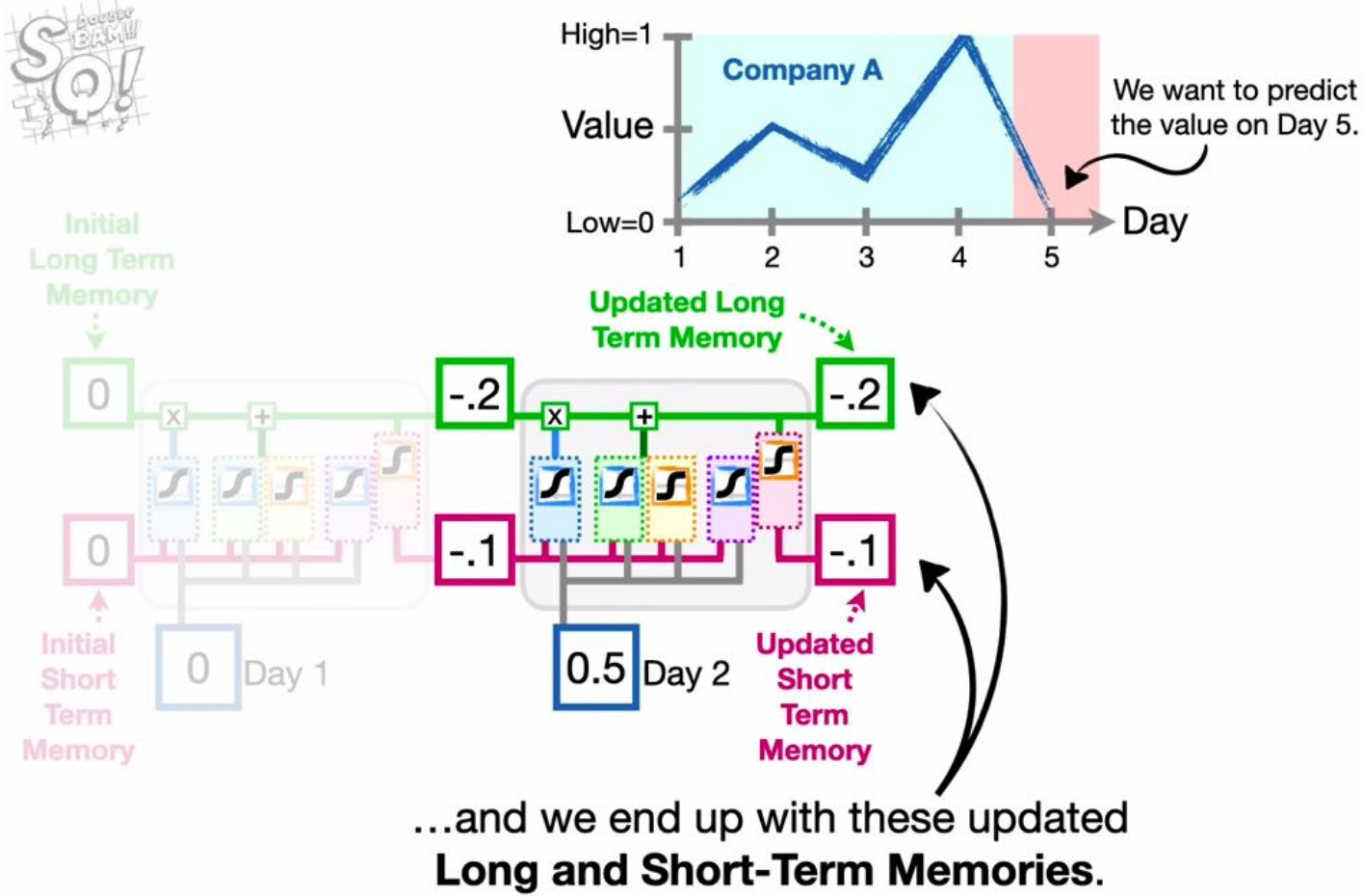


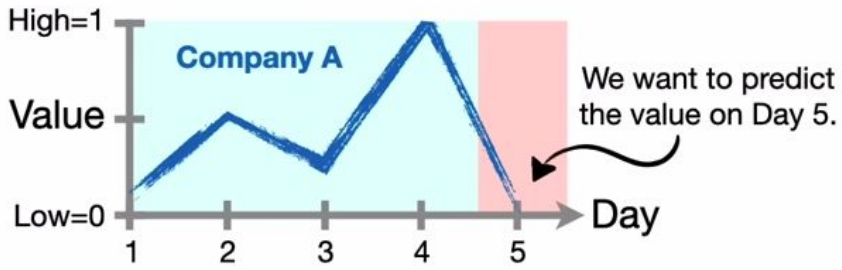
Now we unroll the **LSTM**, using the updated memories…

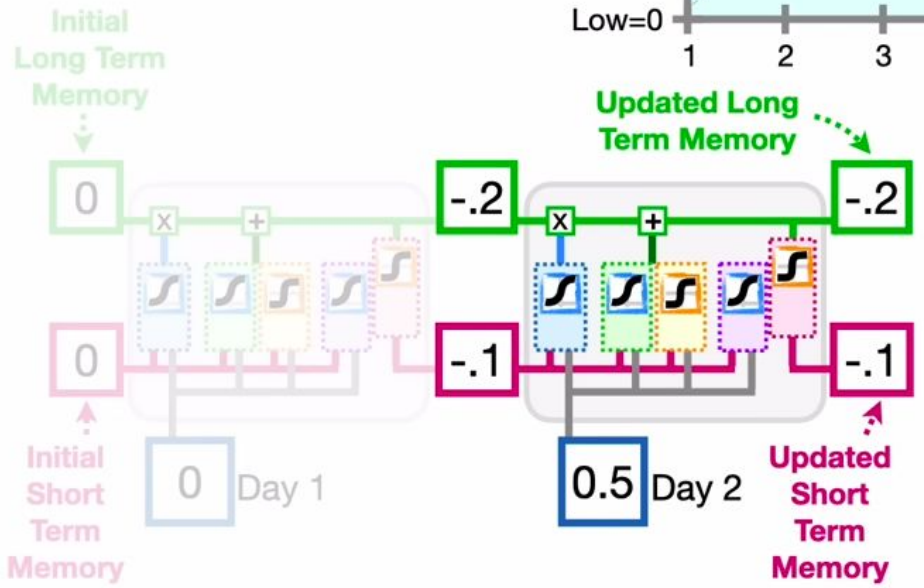


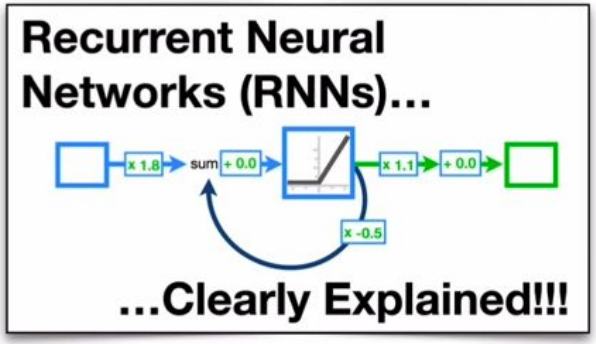


Then the **LSTM** does it's math,
using the exact same **Weights**
and **Biases** as before…

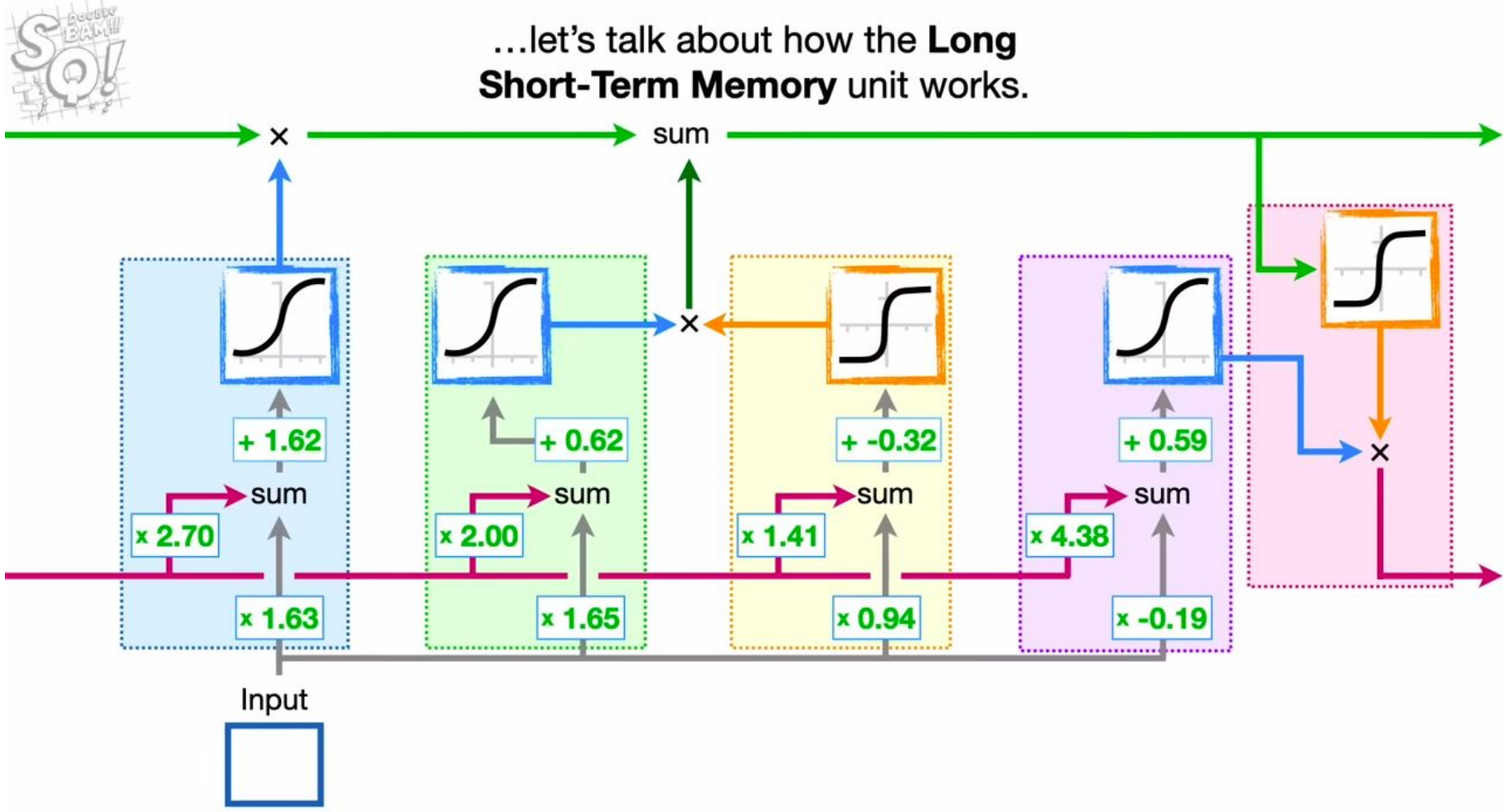

High=1
Value
Low=0
Company A
We want to predict the value on Day 5.
1
2
3
4
5
Day
Initial Long Term Memory
0
Updated Long Term Memory
-.2
-.2
0
-.1
-.1
Initial Short Term Memory
0
Day 1
0.5
Day 2
Updated Short Term Memory
…and we end up with these updated
Long and Short-Term Memories.

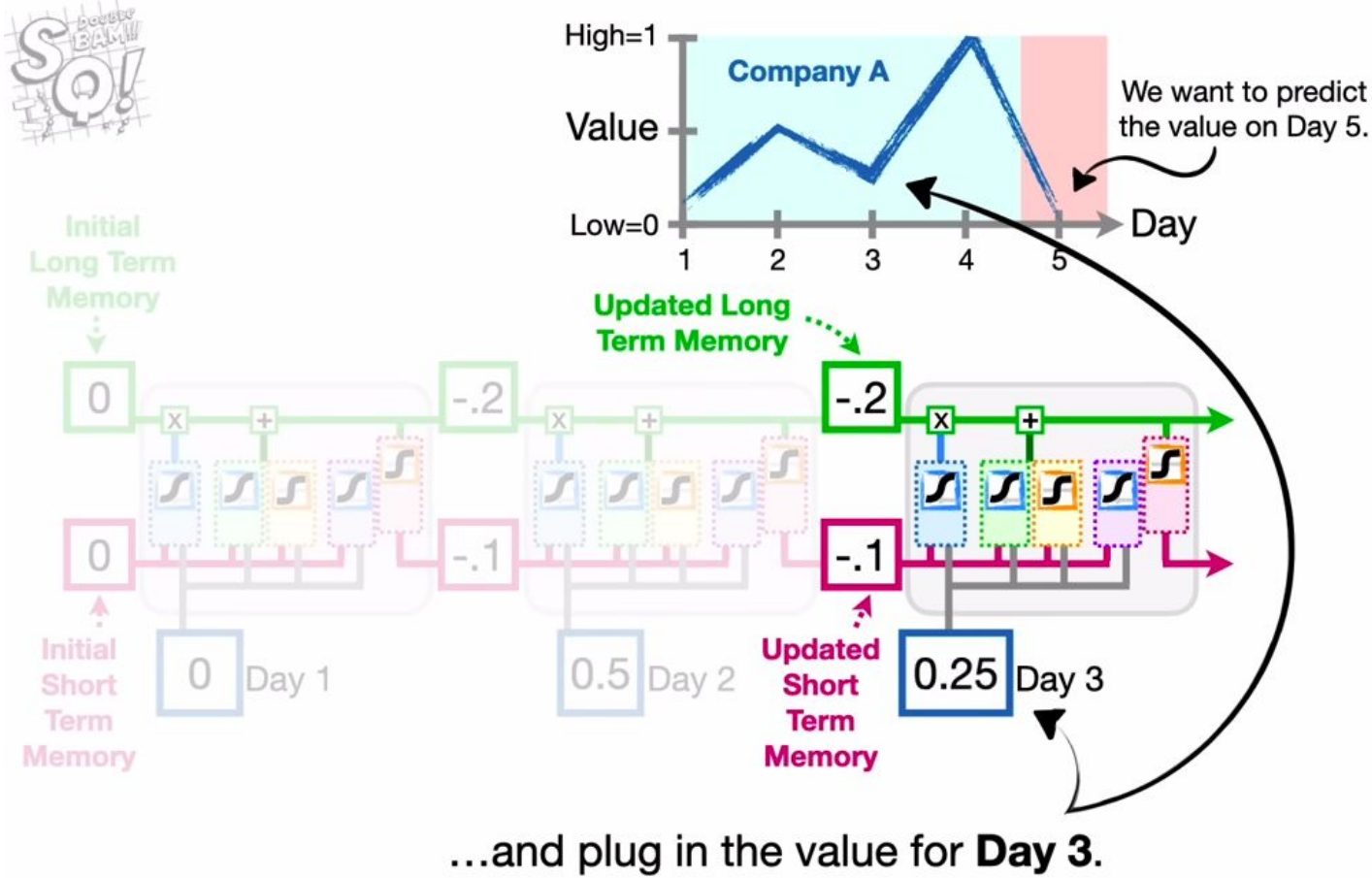




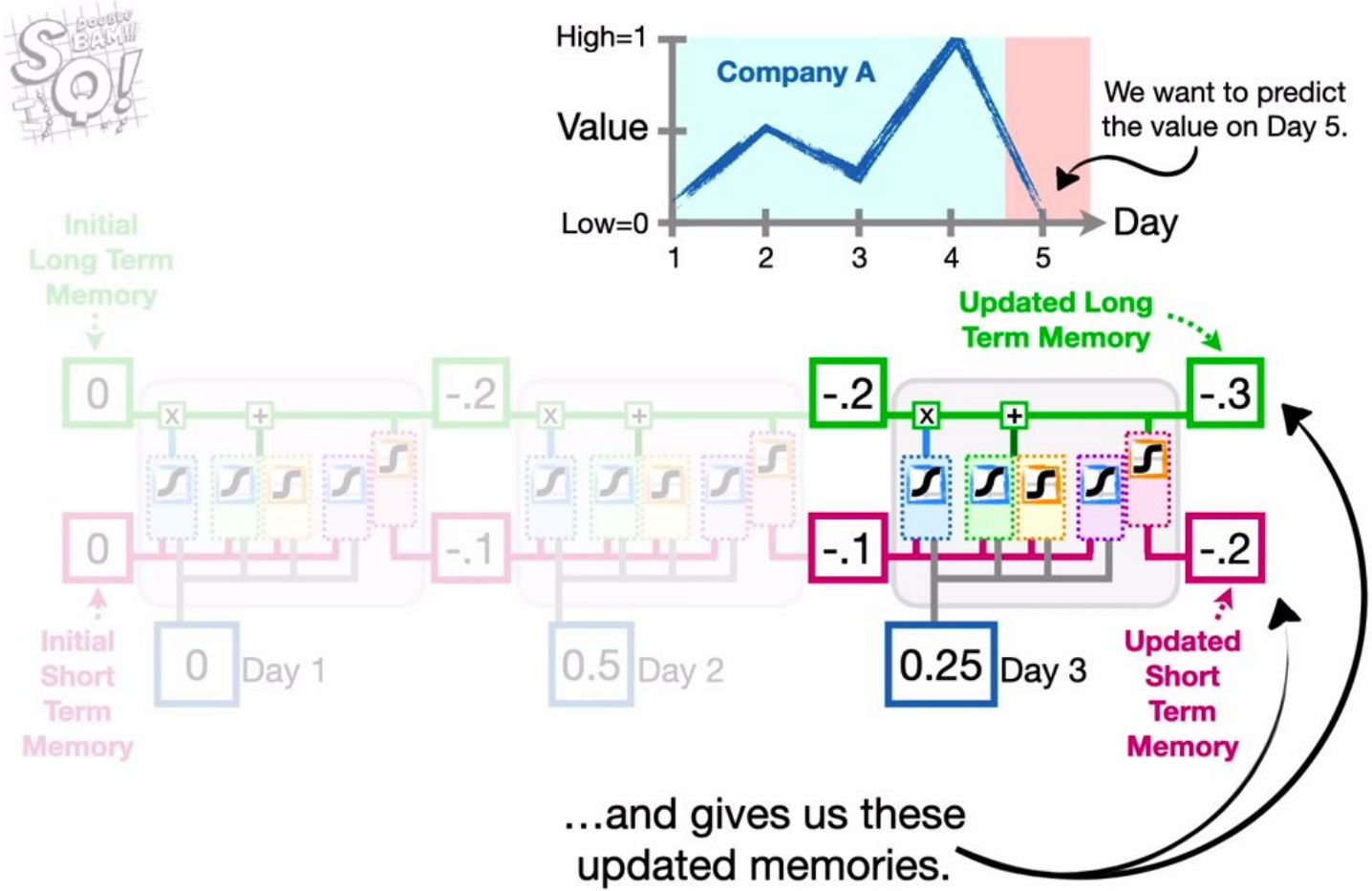


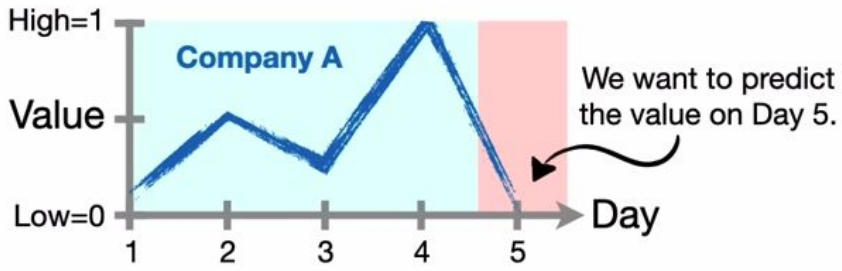
...the reason the **LSTM** reuses the exact same **Weights** and **Biases** is so it can handle data sequences of different lengths.

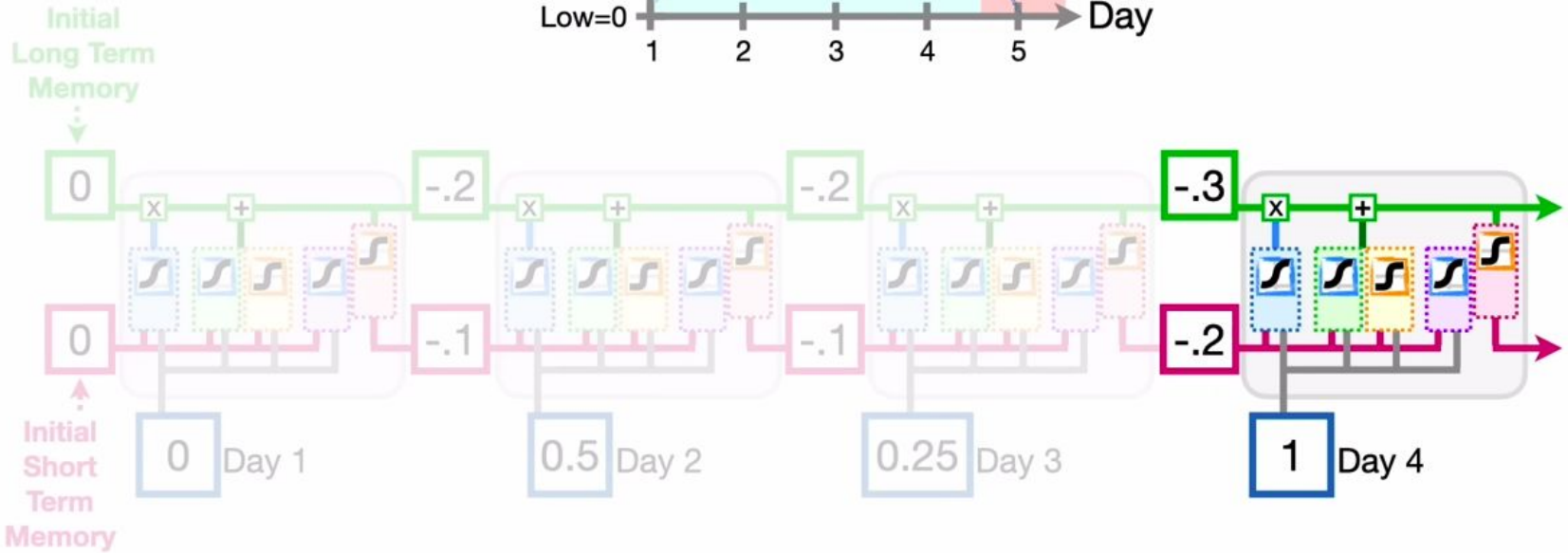

…let's talk about how the **Long Short-Term Memory** unit works.
×
sum
+ 1.62
sum
x 2.70
x 1.63
+ 0.62
sum
x 2.00
x 1.65
×
+ -0.32
sum
x 1.41
x 0.94
+ 0.59
sum
x 4.38
x -0.19
×
Input



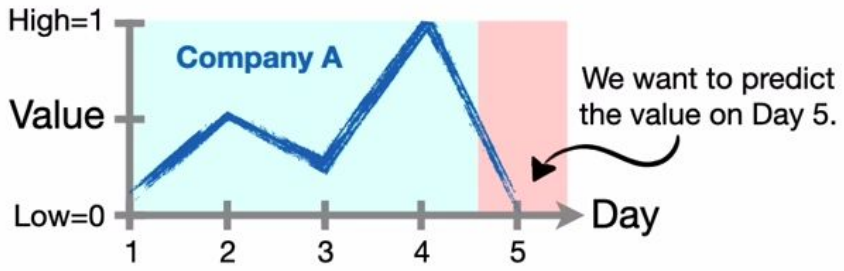
…and plug in the value for **Day 3**.

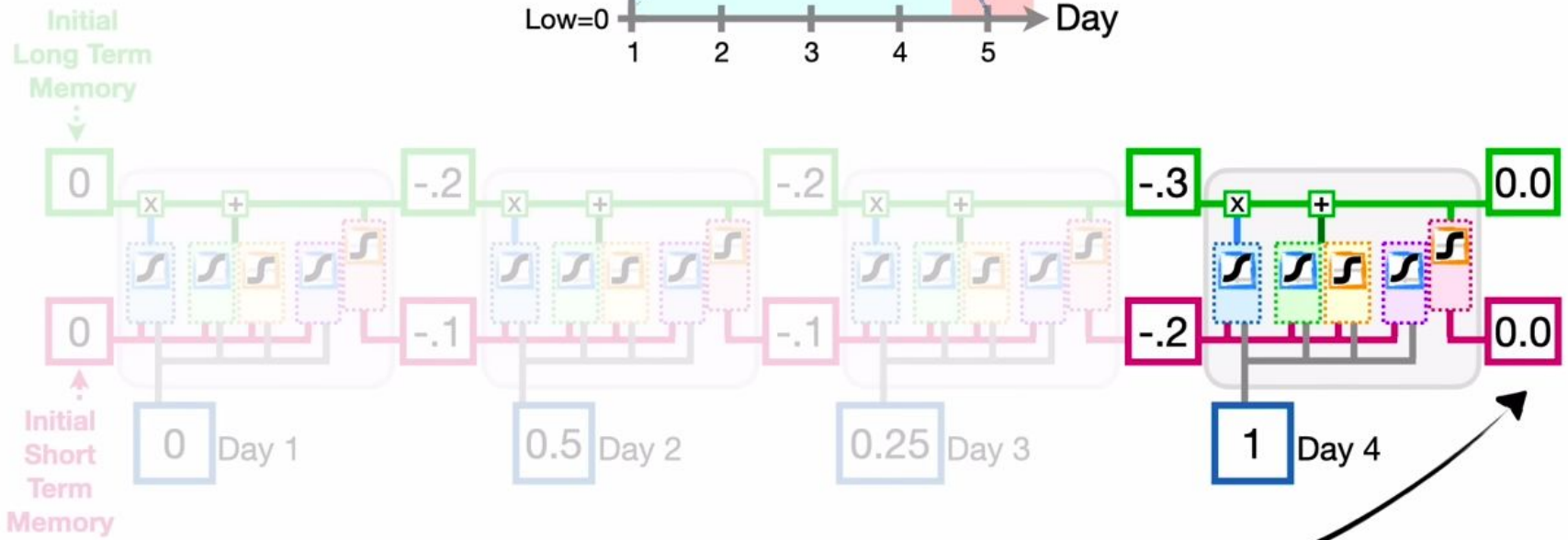

High=1
Value
Low=0
Company A
Day
1
2
3
4
5
We want to predict the value on Day 5.
Initial Long Term Memory
Updated Long Term Memory
0
-.2
-.2
-.3
0
-.1
-.1
-.2
Initial Short Term Memory
0 Day 1
0.5 Day 2
0.25 Day 3
Updated Short Term Memory
…and gives us these updated memories.



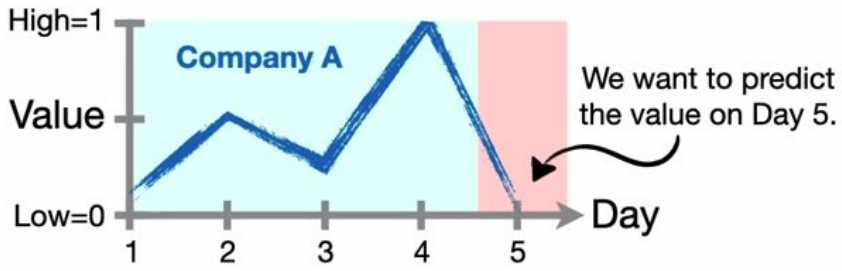


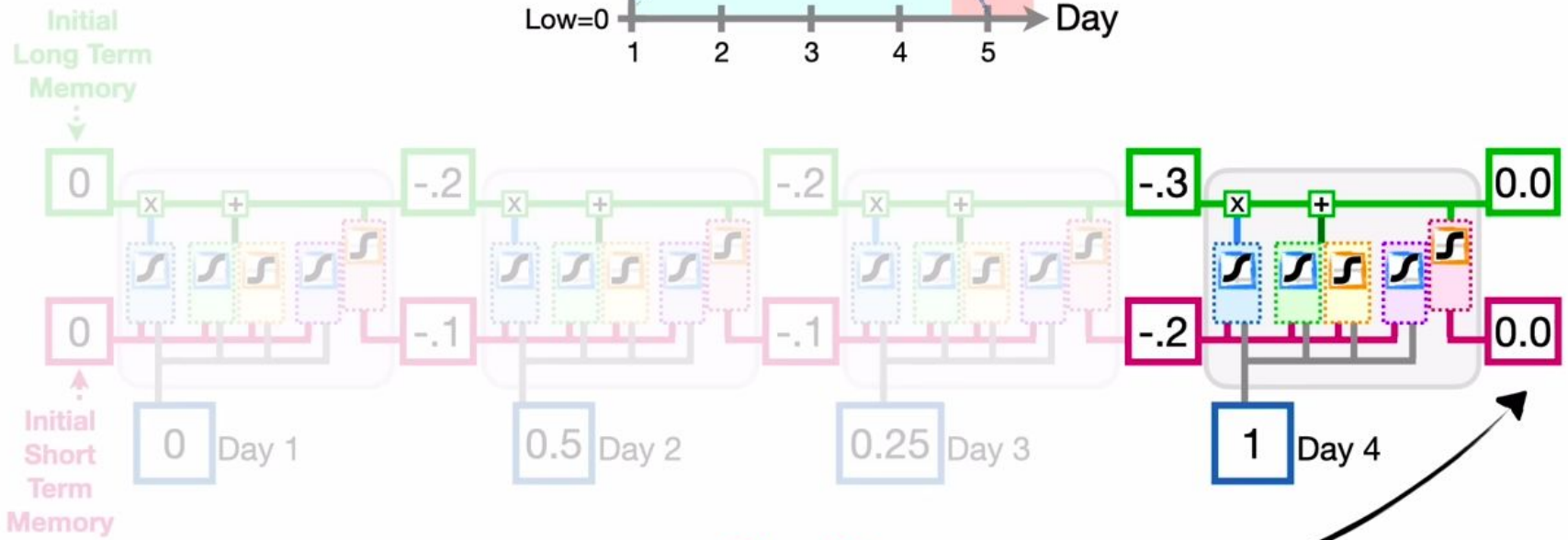
And the **LSTM** does the math, again, using the exact same **Weights** and **Biases**…



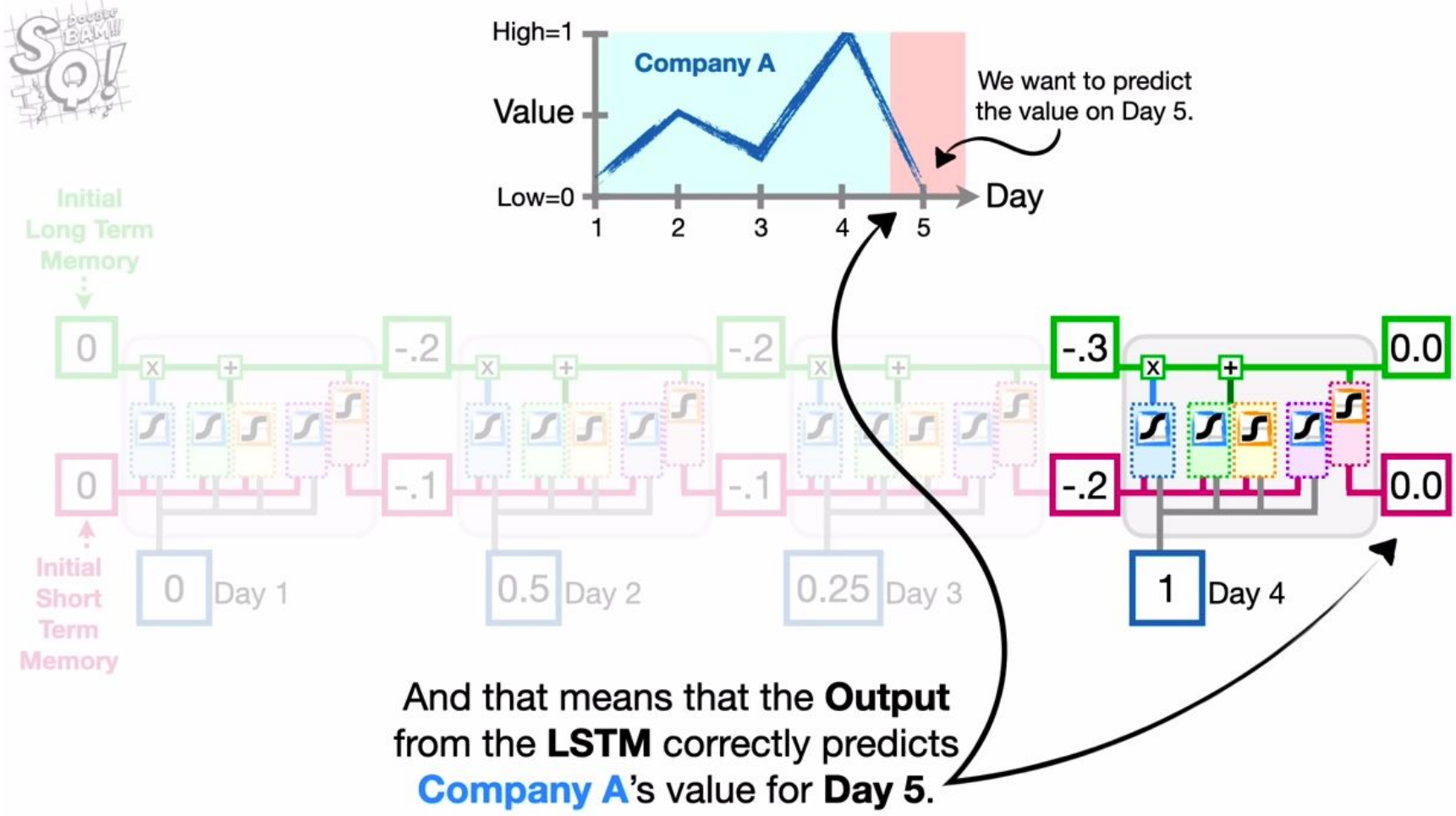


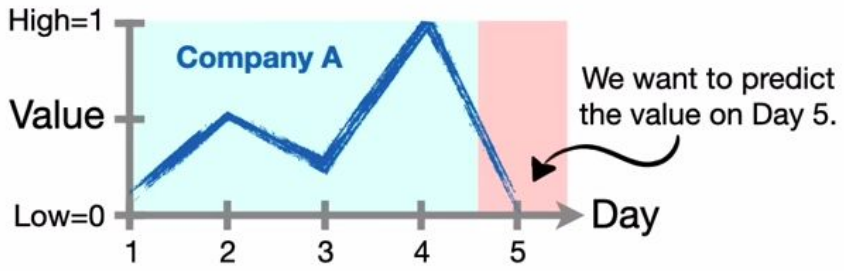
…and gives us the final memories.

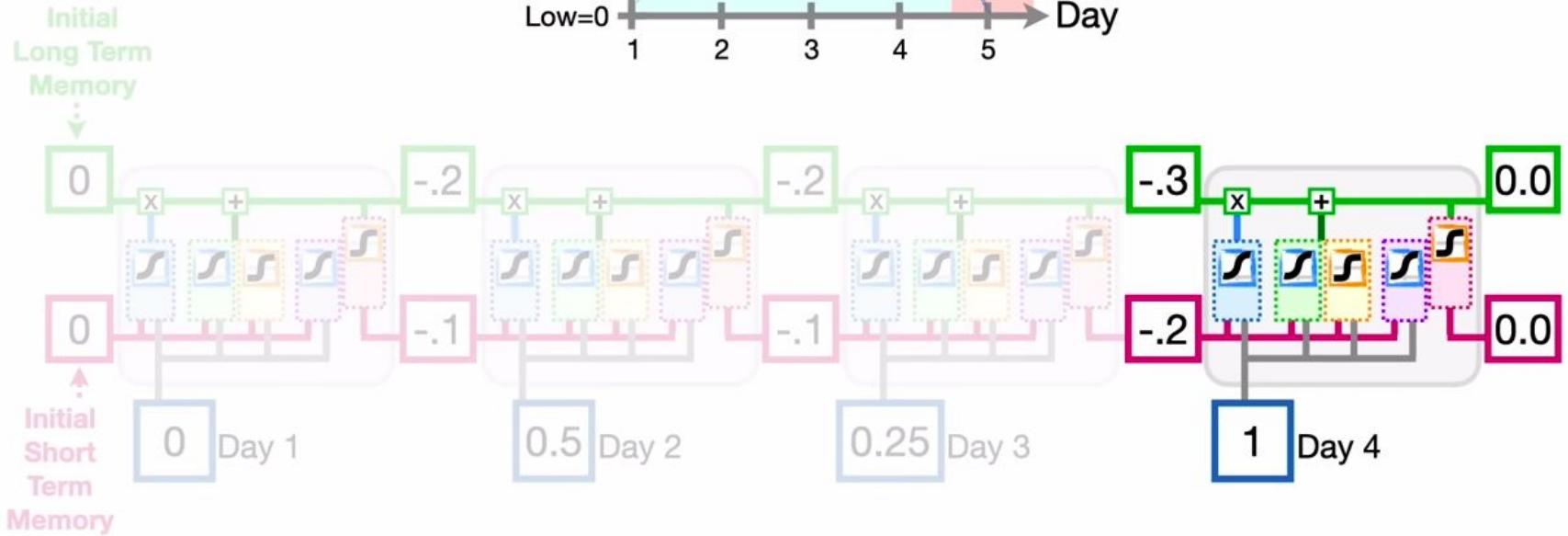




And the final **Short-Term Memory**, **0.0**, is the **Output** from the unrolled **LSTM**.

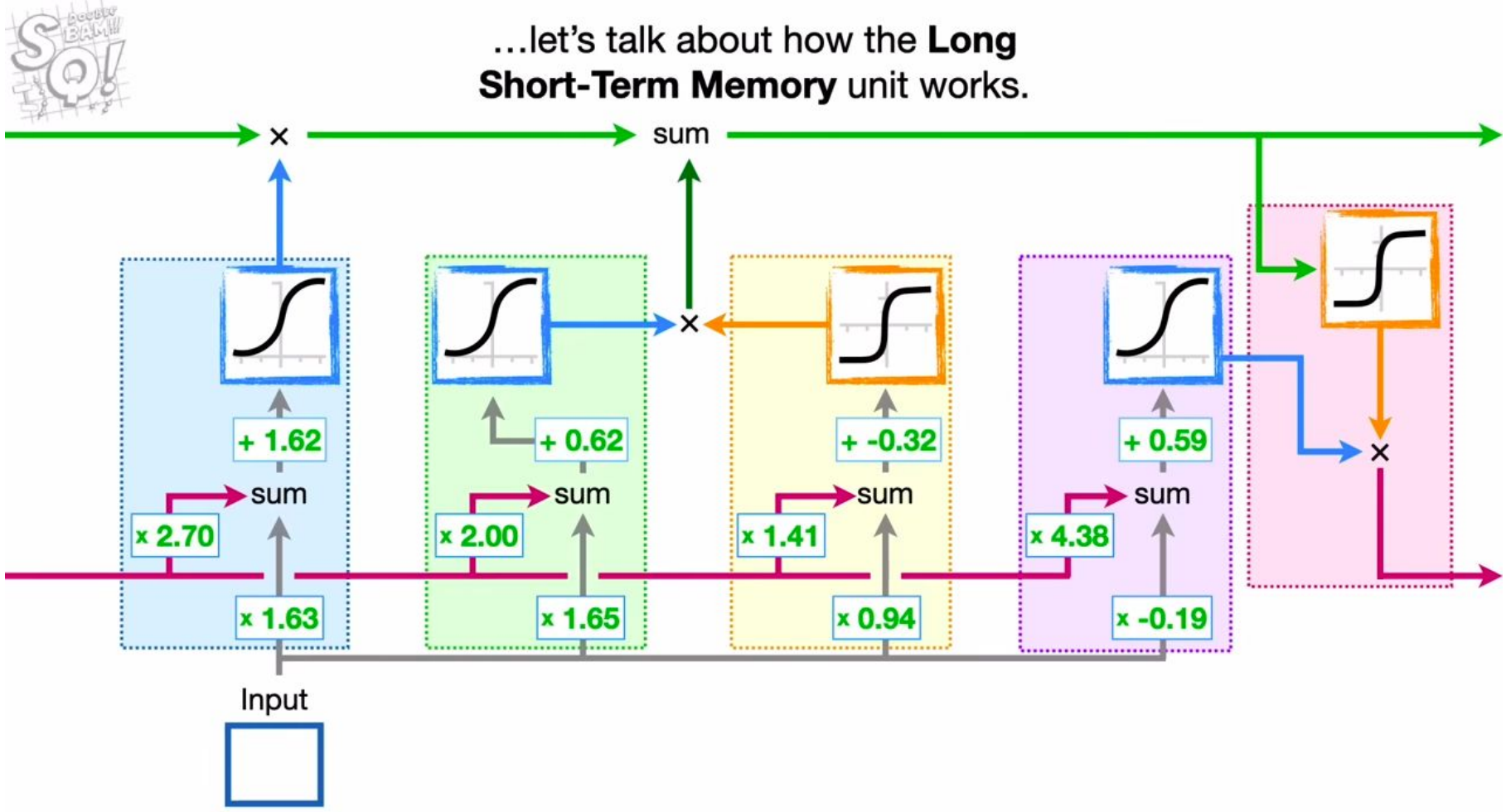

High=1
Value
Low=0
Company A
We want to predict the value on Day 5.
1
2
3
4
5
Day
Initial Long Term Memory
Initial Short Term Memory
0
0
-.2
-.1
-.2
-.1
-.3
-.2
0.0
0.0
0
Day 1
0.5
Day 2
0.25
Day 3
1
Day 4
And that means that the **Output** from the **LSTM** correctly predicts **Company A**'s value for **Day 5**.

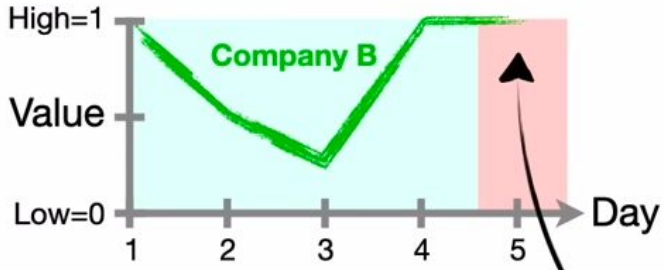




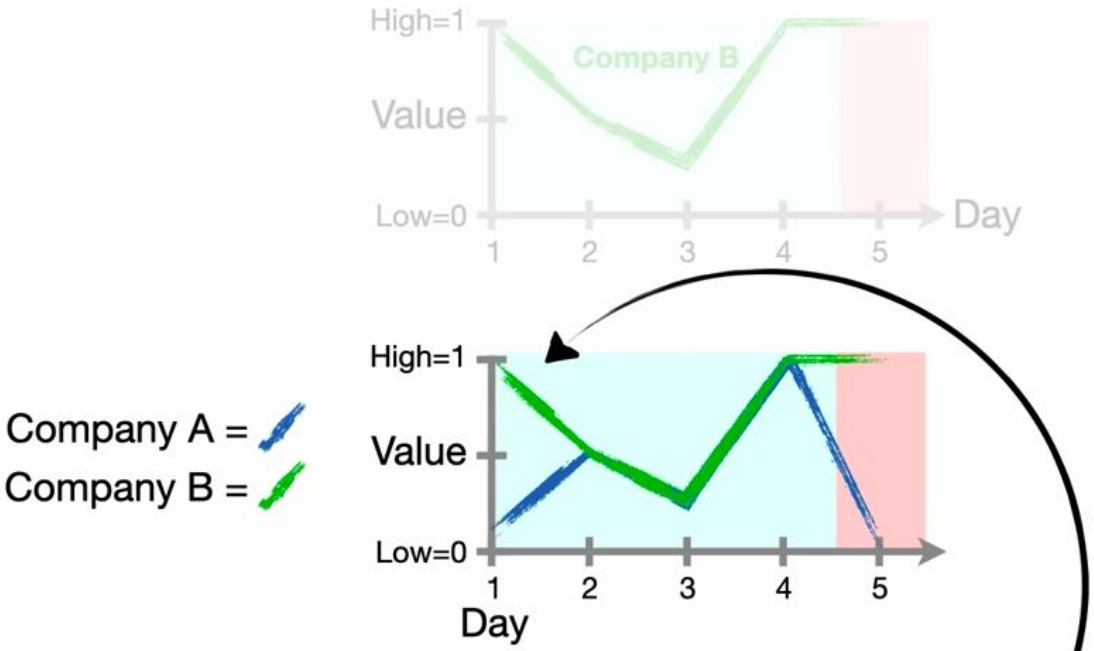
Now that we have shown that the
LSTM can correctly predict the value
on **Day 5** for **Company A**…

Quiz


…let's talk about how the **Long Short-Term Memory** unit works.
×
sum
+ 1.62
sum
x 2.70
x 1.63
+ 0.62
sum
x 2.00
x 1.65
×
+ -0.32
sum
x 1.41
x 0.94
+ 0.59
sum
x 4.38
x -0.19
×
Input



…let's show how the same **LSTM**, with the same **Weights** and **Biases**, can correctly predict the value on **Day 5** for **Company B**.



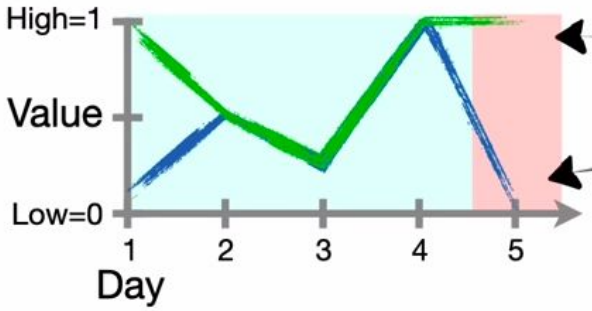
Company A =

Company B =

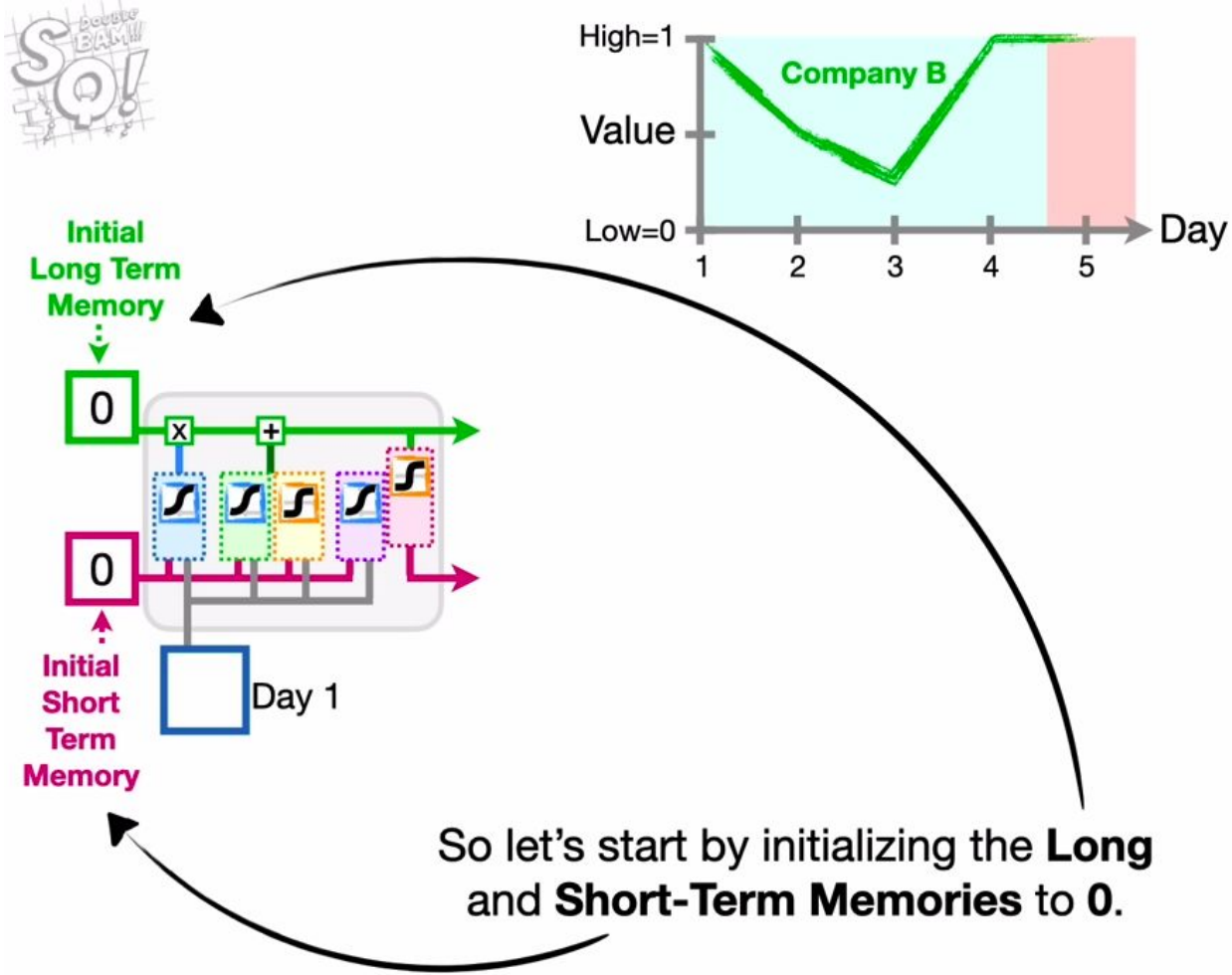
NOTE: Remember, on **Days 1** through **4**, the only difference between the companies occurs on **Day 1**…

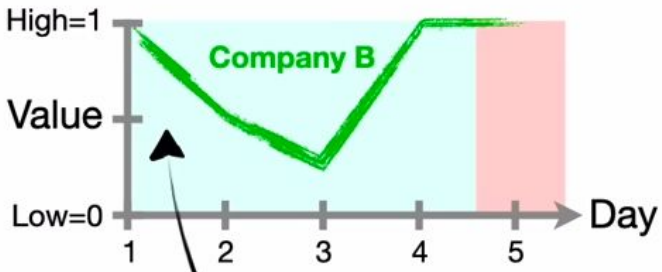
Company A =

Company B =

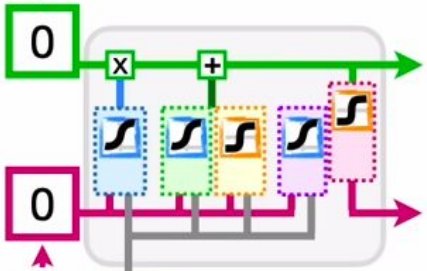


…and that means the **LSTM** has to remember what happened on **Day 1** in order to correctly predict the different output values on **Day 5**.


High=1
Value
Low=0
Company B
Day
1
2
3
4
5
Initial
Long Term
Memory
0
x
+
0
Initial
Short
Term
Memory
Day 1
So let's start by initializing the **Long** and **Short-Term Memories** to **0**.



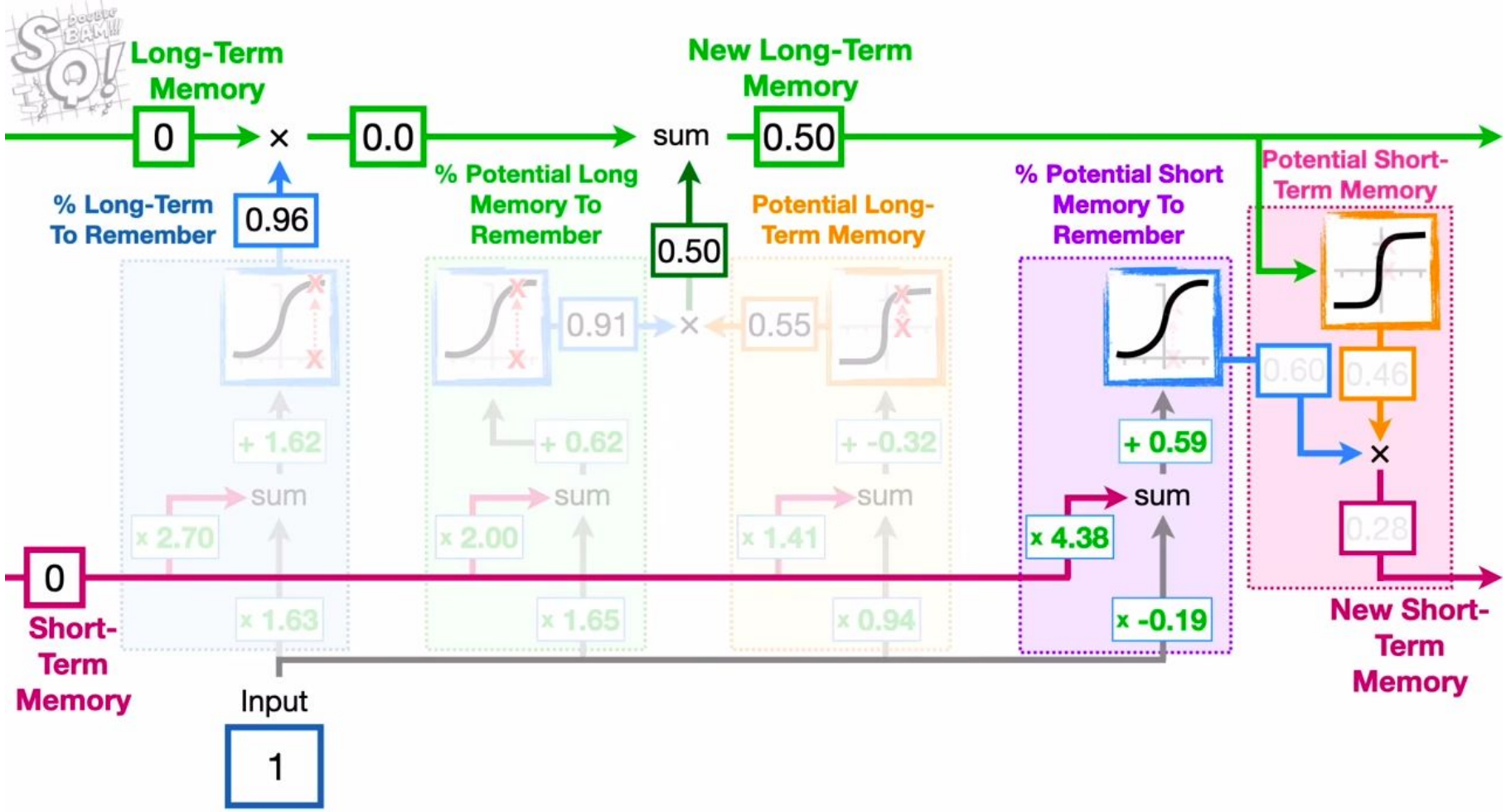
Initial Long Term Memory

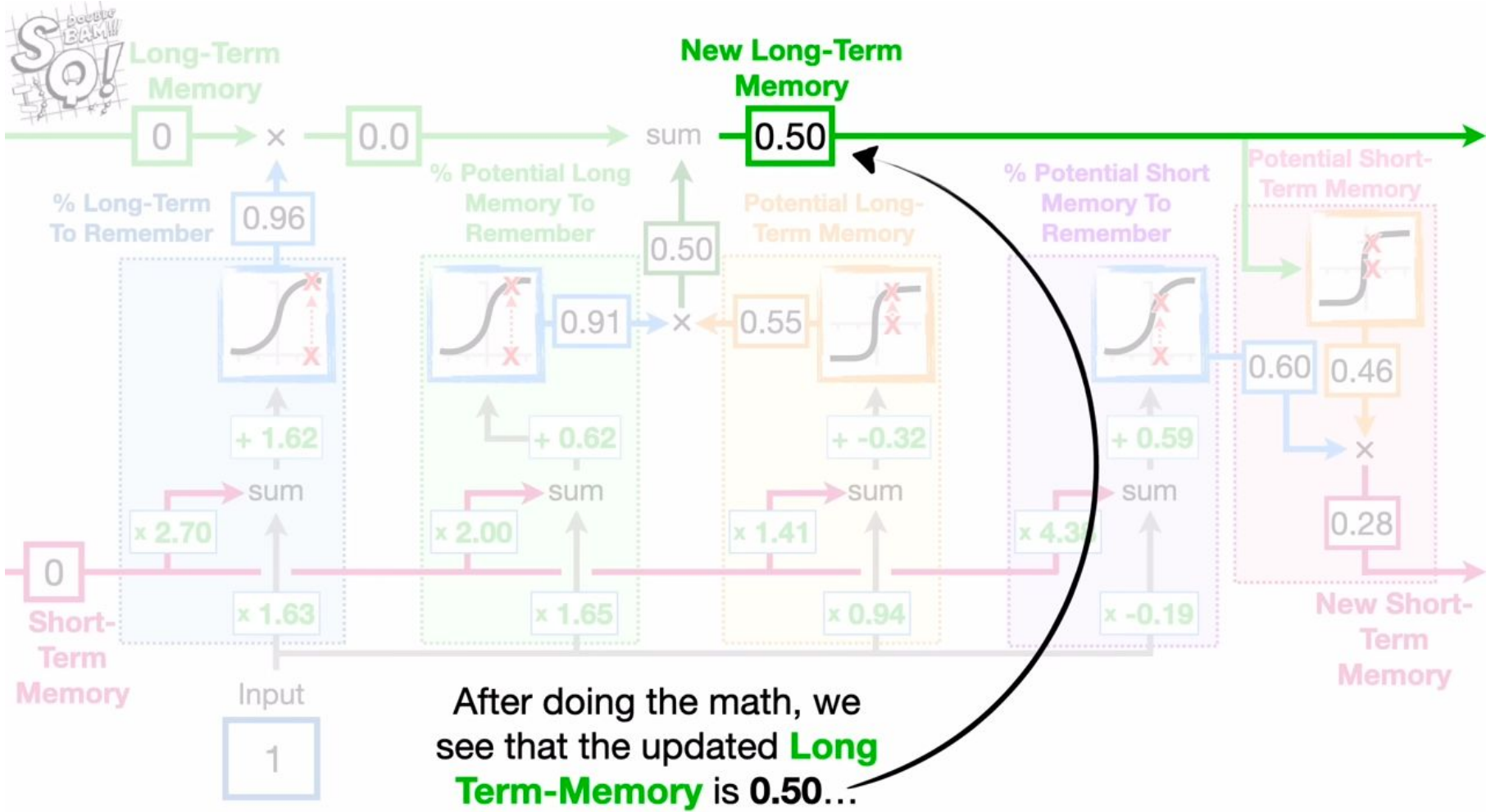


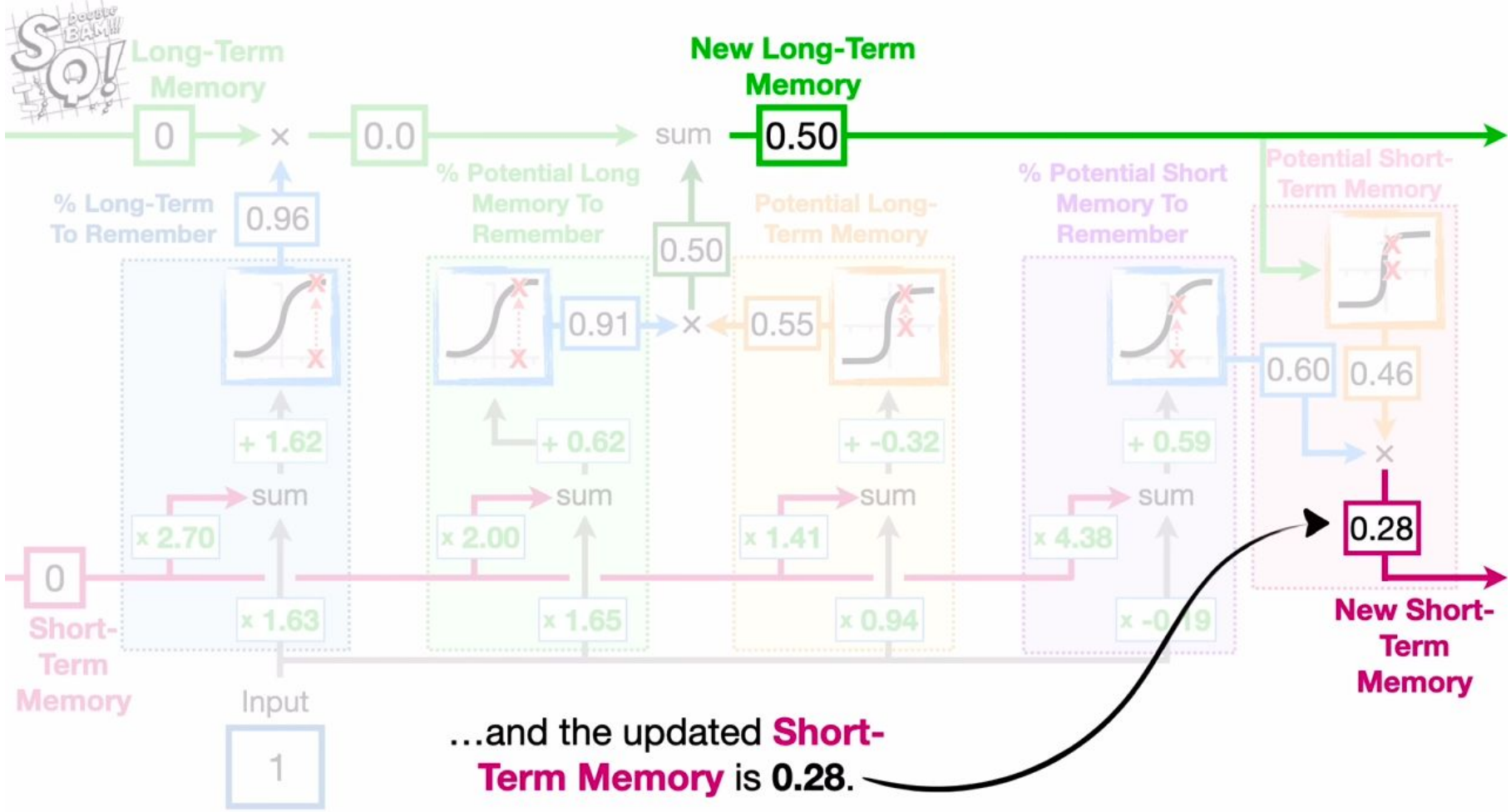
Initial Short Term Memory

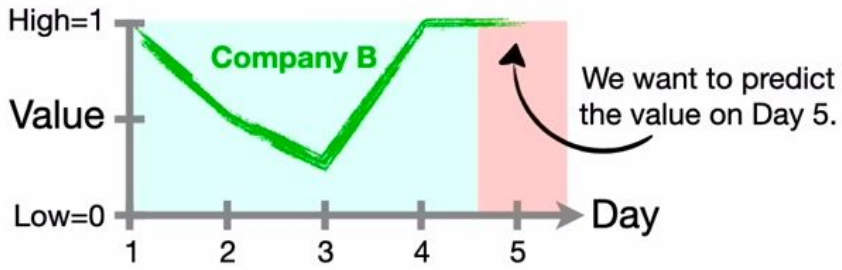
1 Day 1

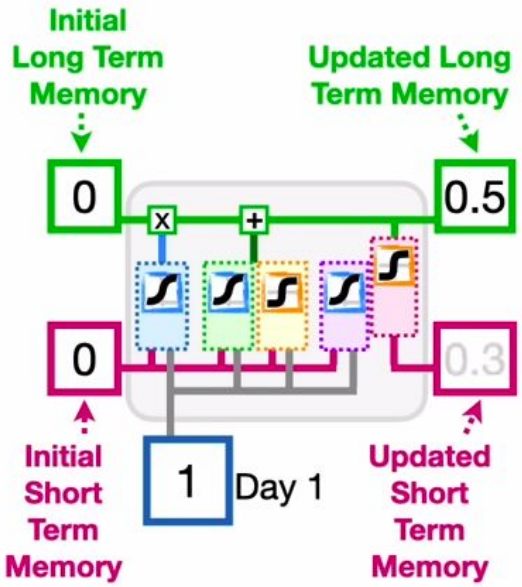
Now let's plug in the value for **Day 1** from **Company B**, **1**…


Long-Term Memory
0
×
0.0
New Long-Term Memory
sum
0.50
% Long-Term To Remember
0.96
% Potential Long Memory To Remember
0.50
Potential Long-Term Memory
% Potential Short Memory To Remember
Potential Short-Term Memory
0.91
×
0.55
+ 1.62
+ 0.62
+ -0.32
+ 0.59
sum
sum
sum
× 2.70
× 2.00
× 1.41
× 4.38
0.60
0.46
×
0.28
0
Short-Term Memory
× 1.63
× 1.65
× 0.94
× -0.19
New Short-Term Memory
Input
1

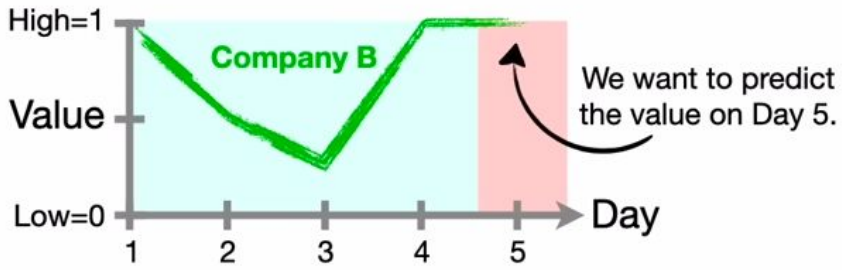

Long-Term Memory
0
×
0.0
sum
New Long-Term Memory
0.50
% Long-Term To Remember
0.96
% Potential Long Memory To Remember
0.91
0.50
×
Potential Long-Term Memory
0.55
% Potential Short Memory To Remember
Potential Short-Term Memory
0.60
0.46
×
0.28
+ 1.62
+ 0.62
+ -0.32
+ 0.59
sum
sum
sum
sum
× 2.70
× 2.00
× 1.41
× 4.39
0
Short-Term Memory
× 1.63
× 1.65
× 0.94
× -0.19
New Short-Term Memory
Input
1
After doing the math, we see that the updated Long Term-Memory is 0.50…

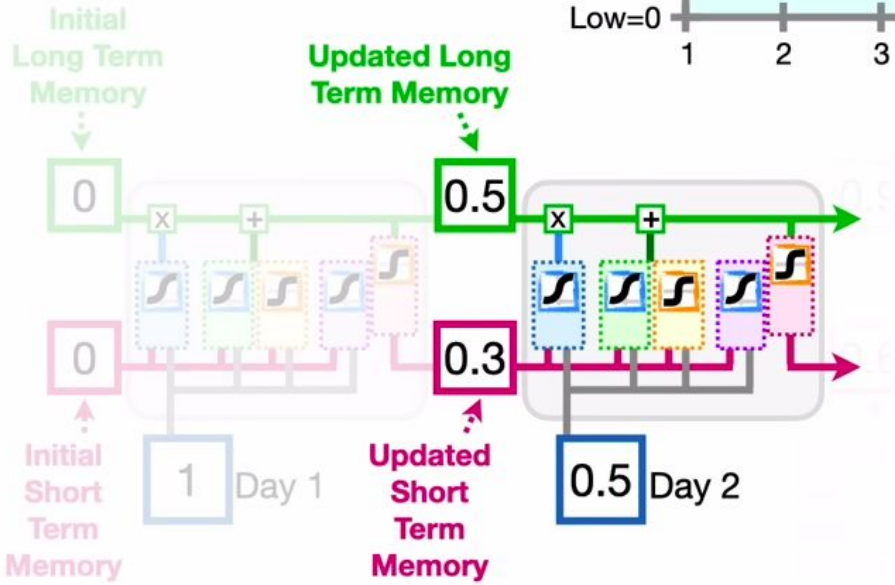

Long-Term Memory
0
×
0.0
sum
New Long-Term Memory
0.50
% Long-Term To Remember
0.96
% Potential Long Memory To Remember
0.91
0.50
Potential Long-Term Memory
0.55
% Potential Short Memory To Remember
Potential Short-Term Memory
0.60
0.46
+ 1.62
sum
× 2.70
× 1.63
+ 0.62
sum
× 2.00
× 1.65
+ -0.32
sum
× 1.41
× 0.94
+ 0.59
sum
× 4.38
× -0.19
0.28
New Short-Term Memory
0
Short-Term Memory
Input
1
…and the updated Short-Term Memory is 0.28.



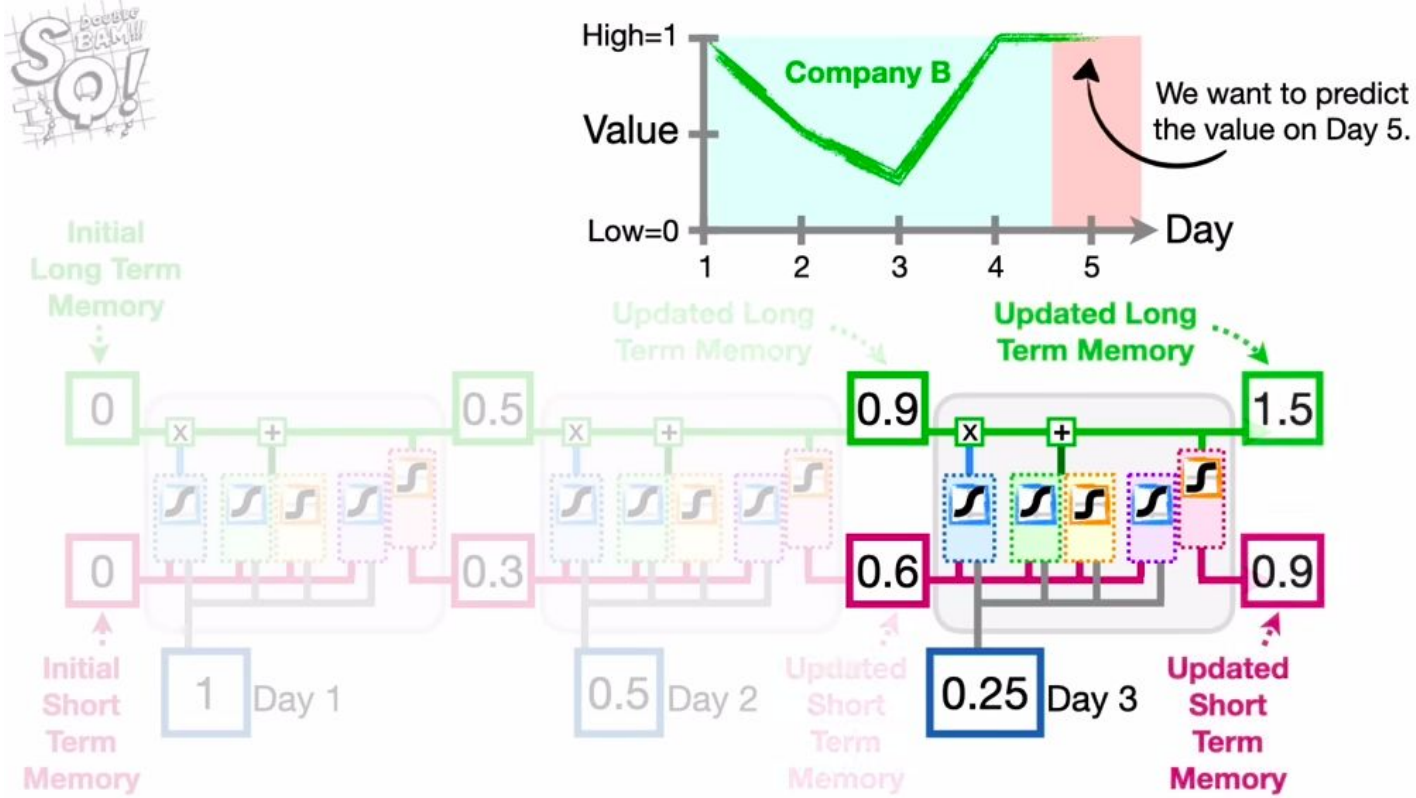


…and **0.3** (rounded) for the updated **Short-Term Memory**.

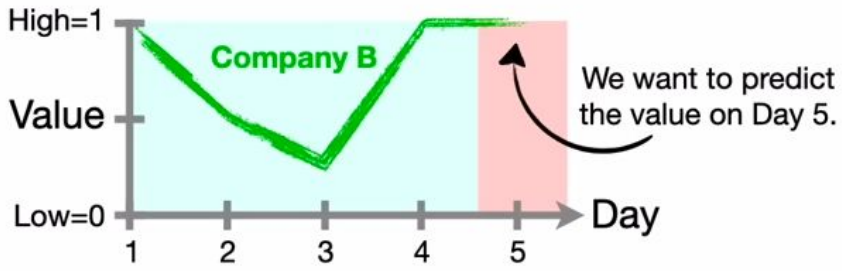


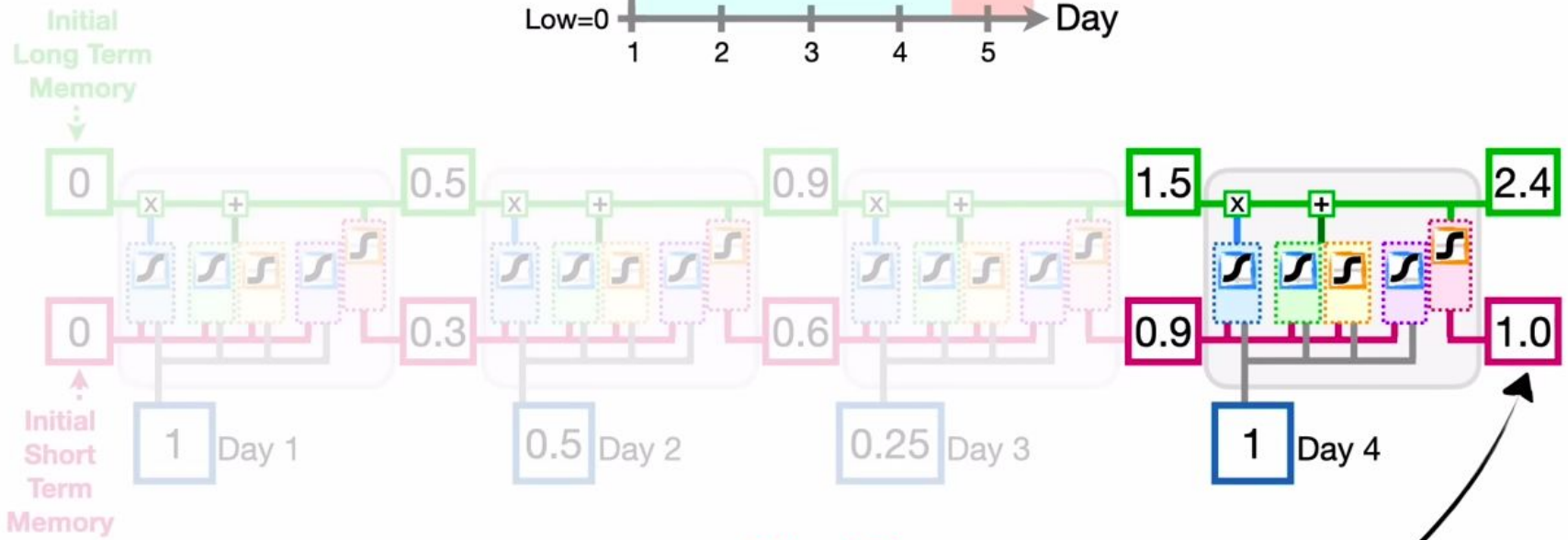


Now we unroll the **LSTM** and do the math with the remaining input values…

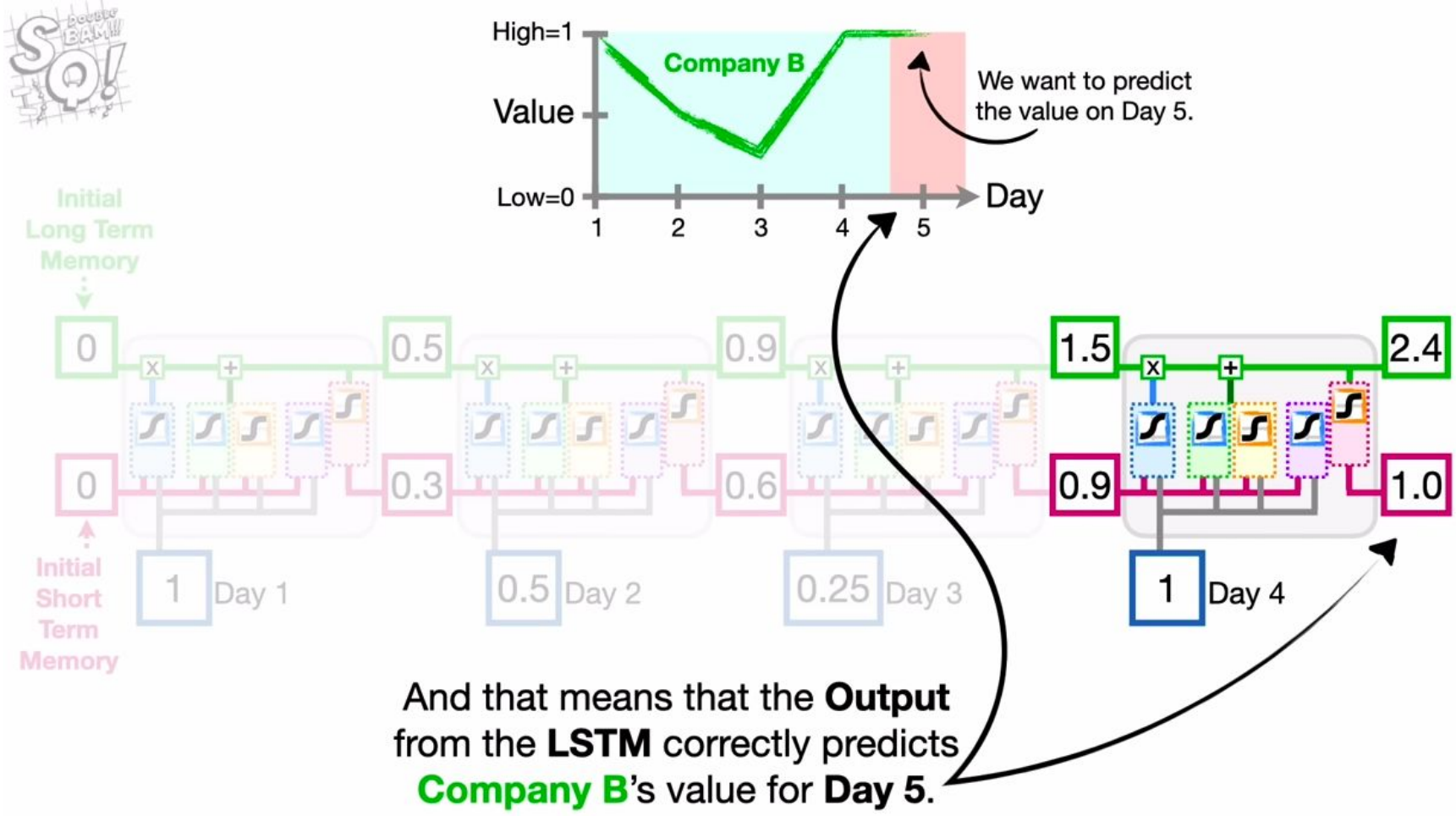


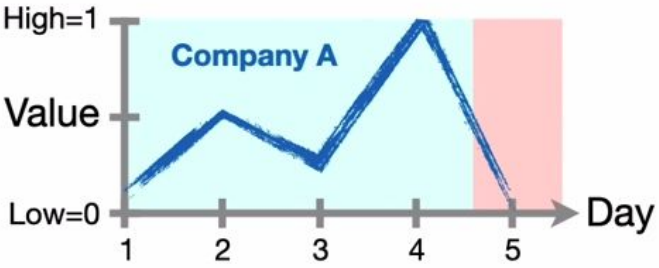
Now we unroll the **LSTM** and do the math with the remaining input values…

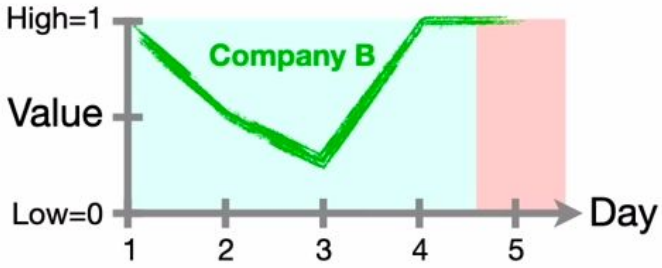




…and the final **Short-Term Memory**, **1.0**, is the **Output** from the **unrolled LSTM**.


High=1
Value
Low=0
Company B
We want to predict the value on Day 5.
1
2
3
4
5
Day
Initial Long Term Memory
Initial Short Term Memory
0
0
0.5
0.3
0.9
0.6
1.5
0.9
2.4
1.0
1
Day 1
0.5
Day 2
0.25
Day 3
1
Day 4
And that means that the Output from the LSTM correctly predicts Company B's value for Day 5.





Initial Long Term Memory

Initial Short Term Memory

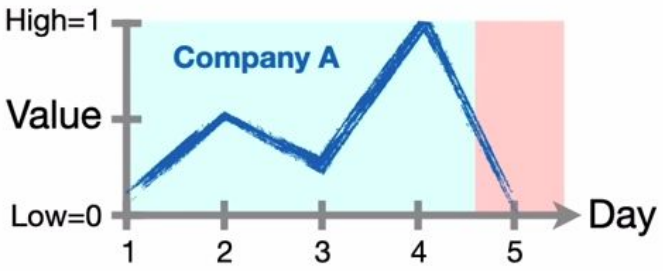
Day 1

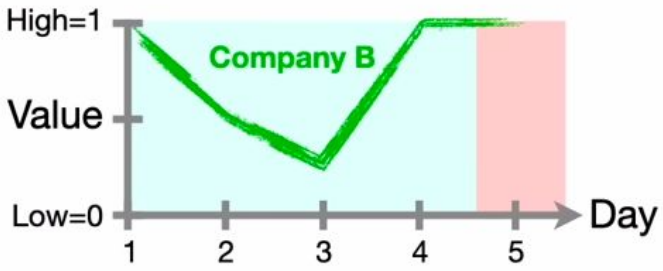
Day 2

Day 3

Day 4

In summary, using separate paths for **Long-Term Memories**…





Initial Long Term Memory

Initial Short Term Memory

Day 1

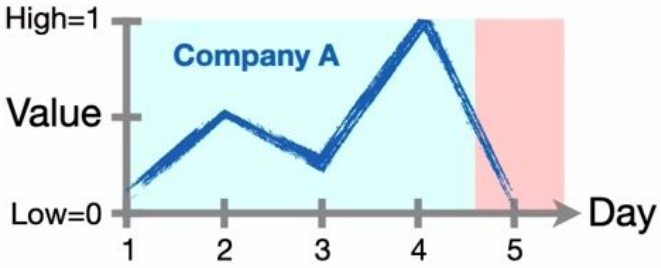
Day 2

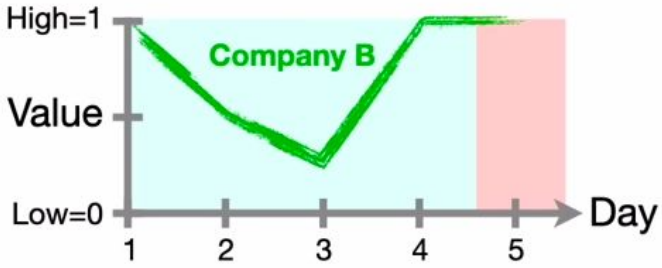
Day 3

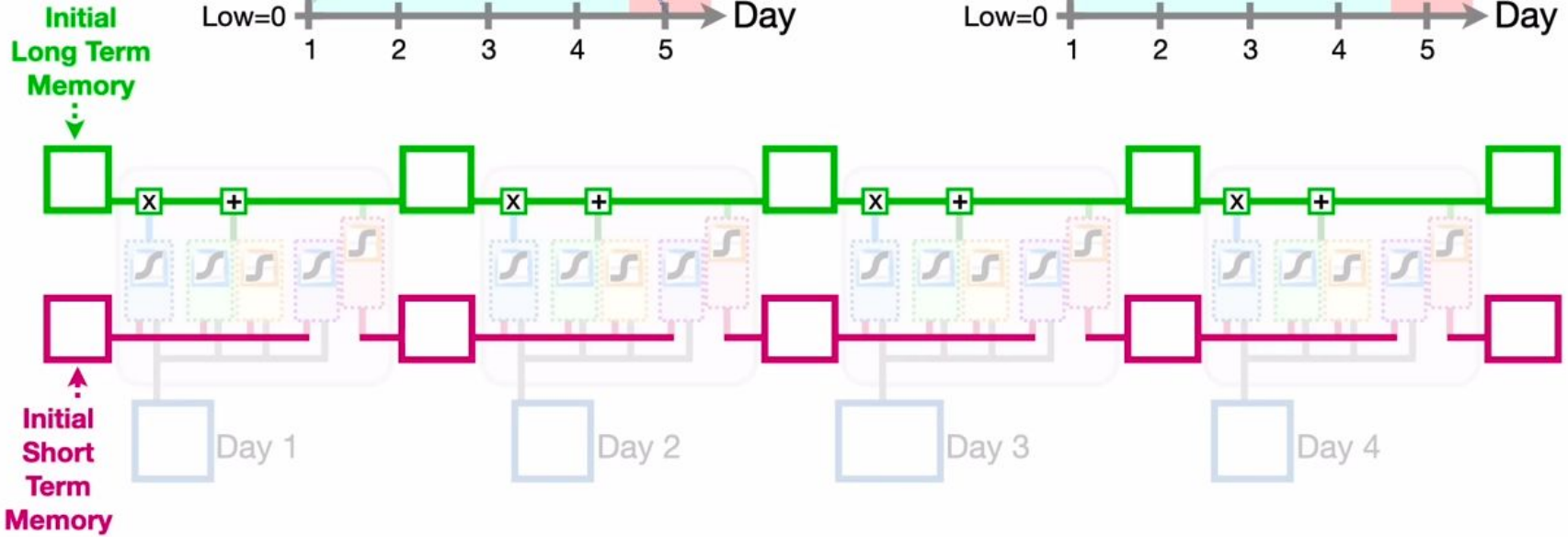
Day 4

In summary, using separate paths for **Long-Term Memories**…

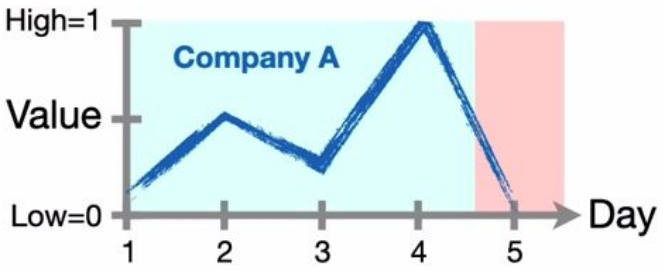
…and **Short-Term Memories**…

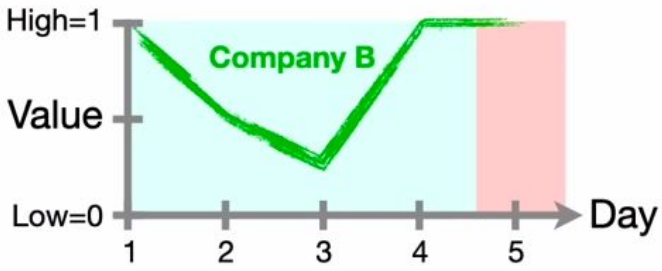


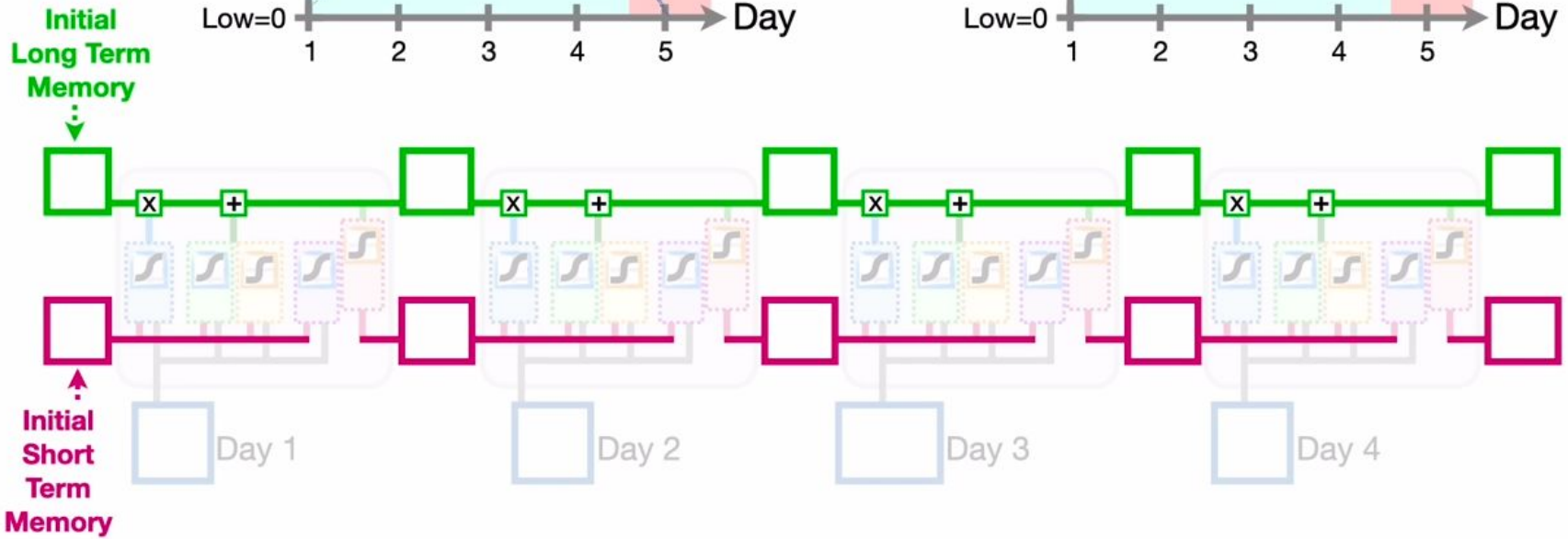




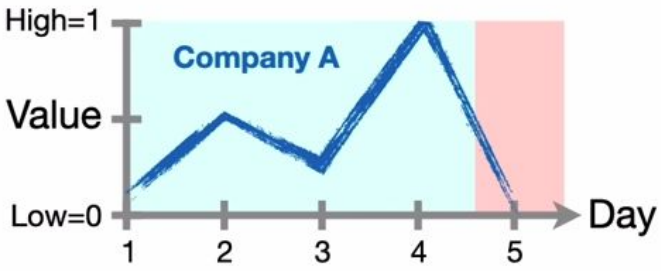
…**Long Short-Term Memory Networks** avoid the **exploding/ vanishing** gradient problem…

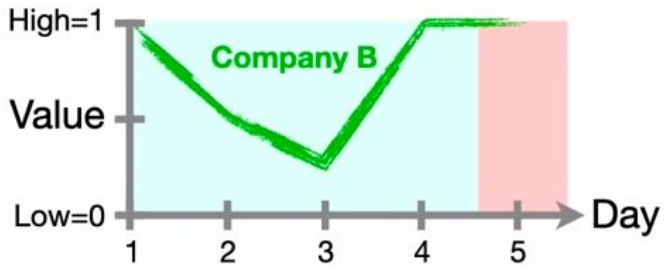






…and that means we can unroll them more times to accommodate longer sequences of input data than a vanilla **Recurrent Neural Network**.





Initial Long Term Memory

Initial Short Term Memory

Day 1

Day 2

Day 3

At first I was scared of how complicated the **LSTM** was, but now I understand.