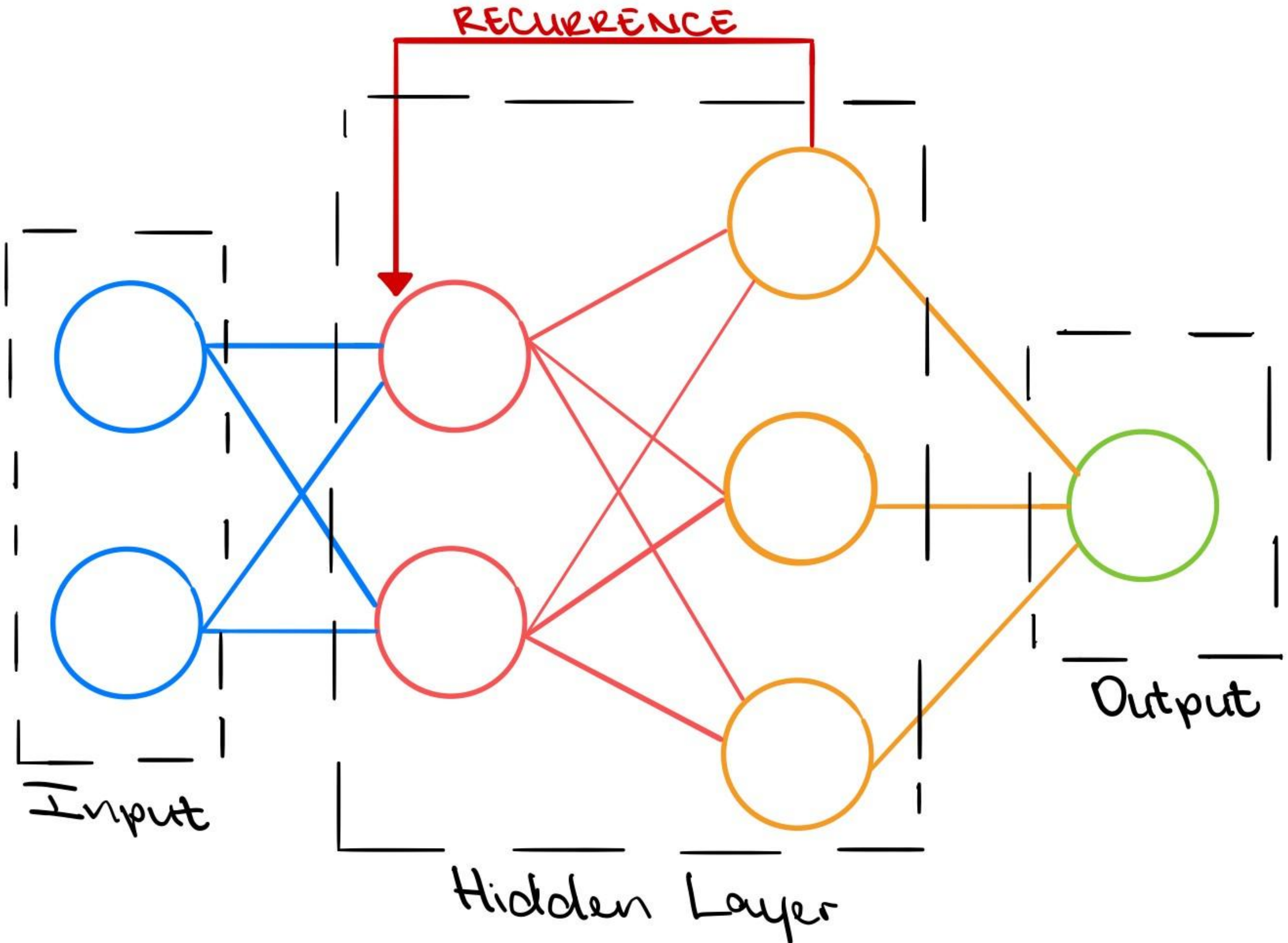
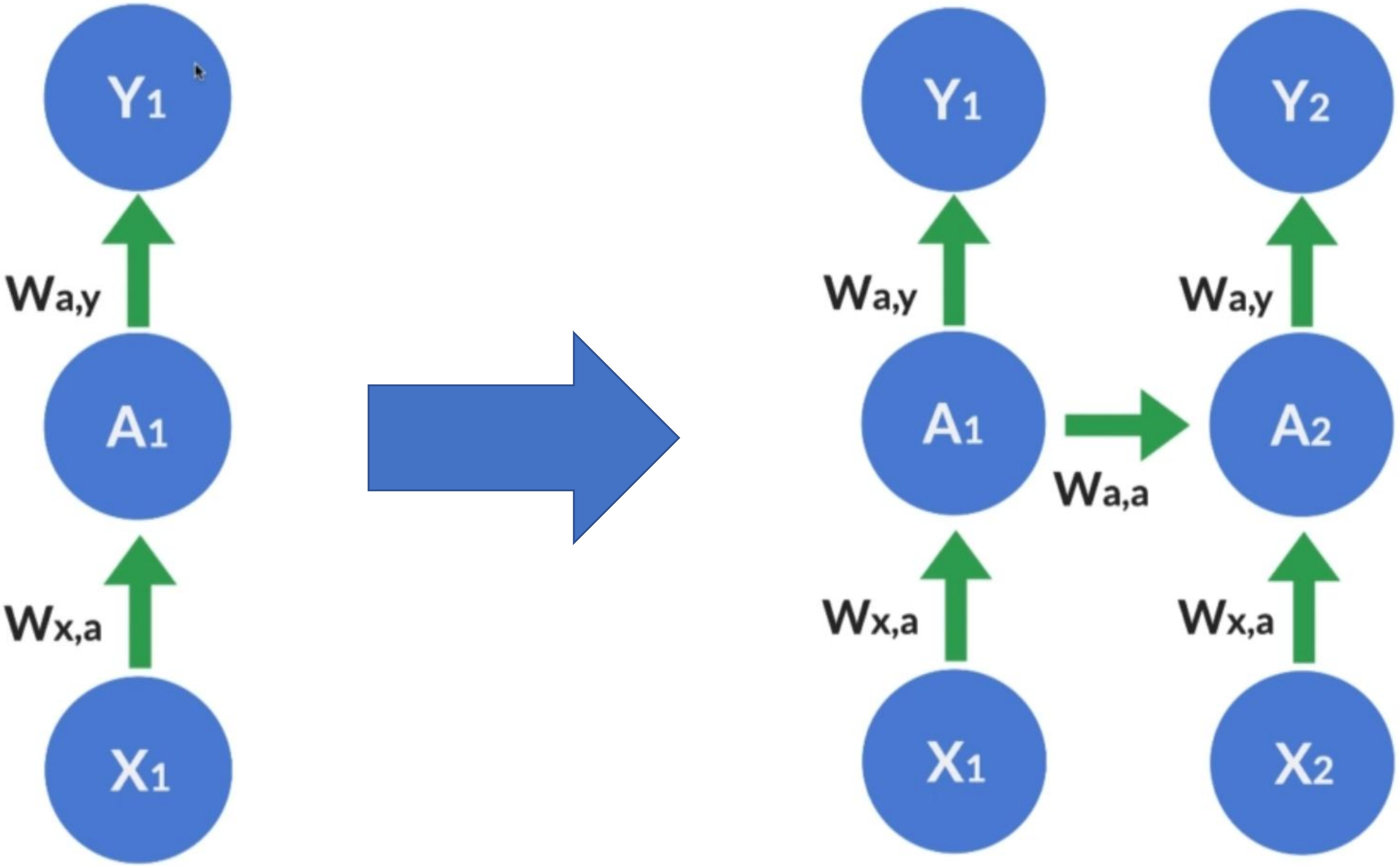
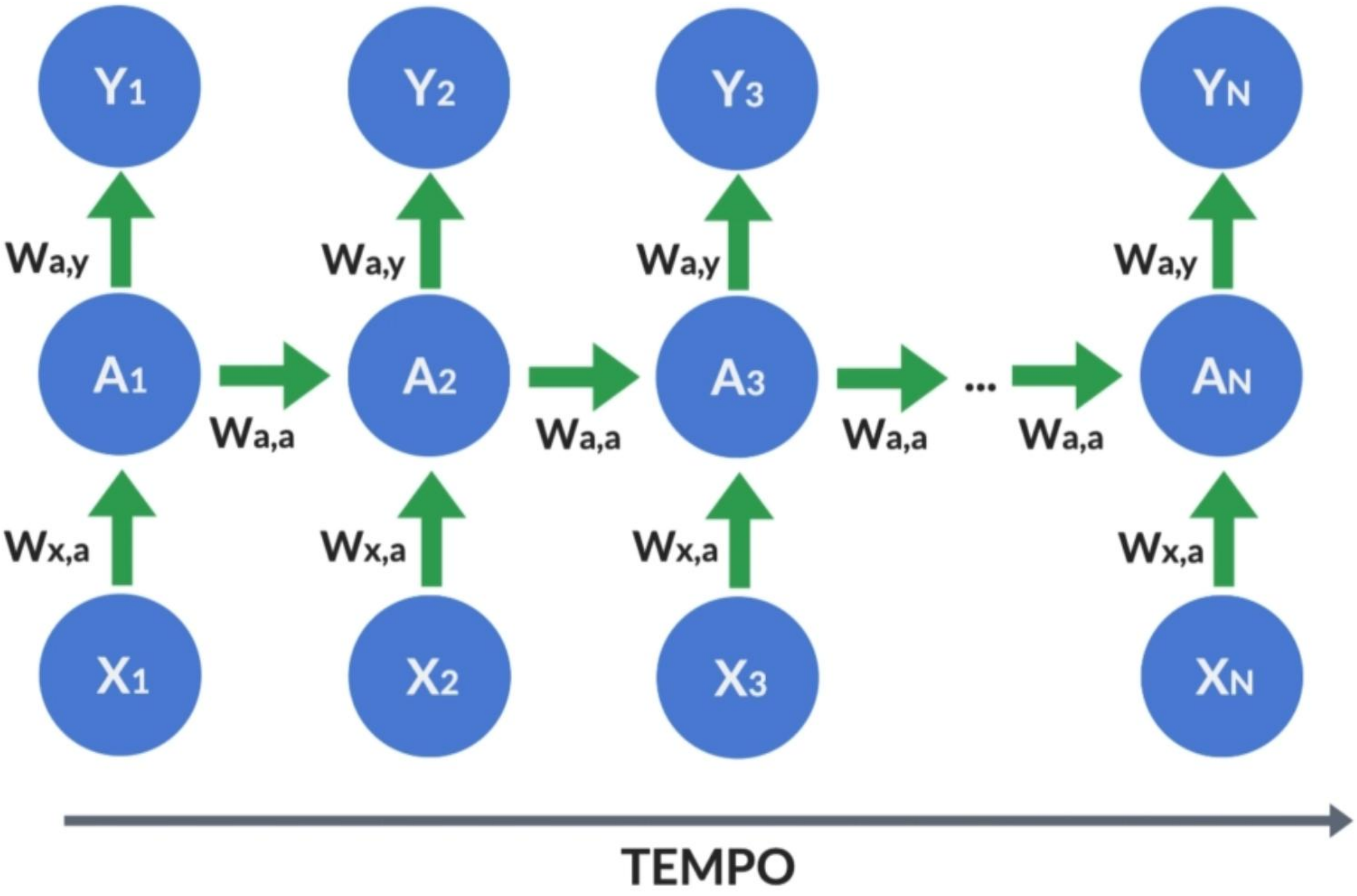
RECURRENT NEURAL NETWORKS

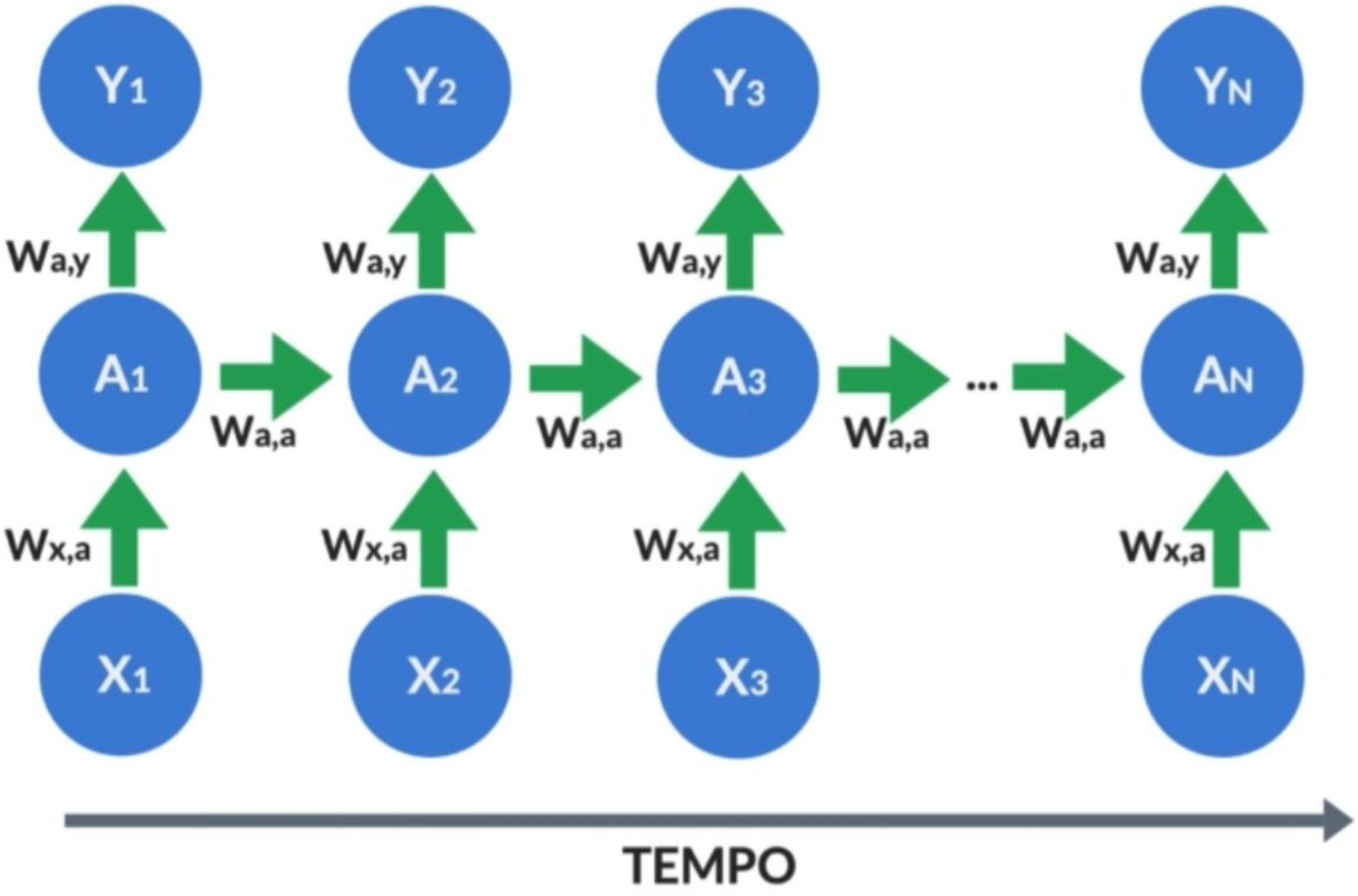




Y1
Wa,y
A1
Wx,a
X1
Y1
Y2
Wa,y
Wa,y
A1
A2
Wa,a
Wx,a
Wx,a
X1
X2

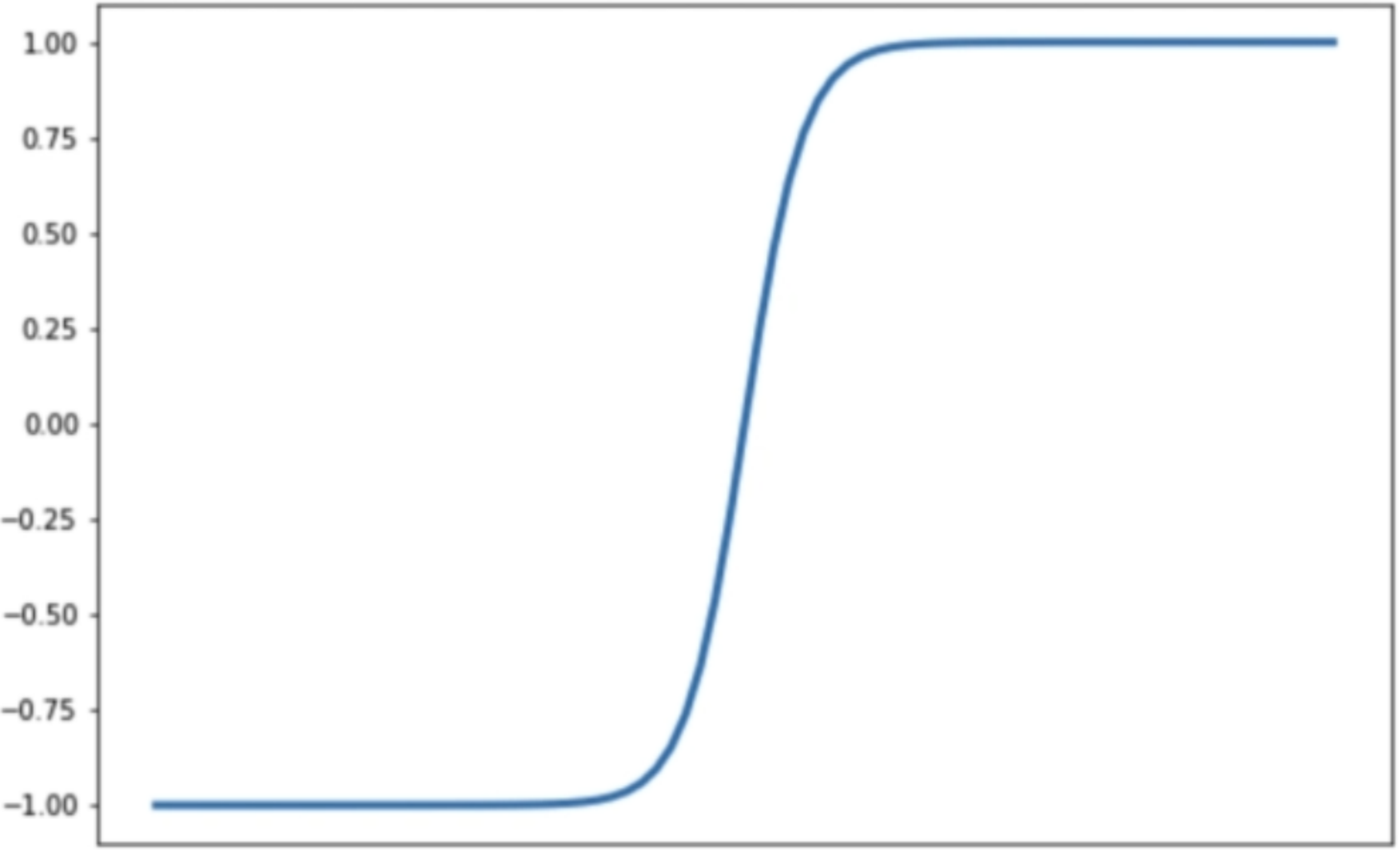


Y1
Y2
Y3
YN
Wa,y
Wa,y
Wa,y
Wa,y
A1
A2
A3
AN
Wa,a
Wa,a
Wa,a
...
Wa,a
Wx,a
Wx,a
Wx,a
Wx,a
X1
X2
X3
XN
TEMPO



$$a_1 = \tanh(W_{x,a}x_1 + b_a)$$
$$y_1 = \mathrm{softmax}(W_{a,y}a_1 + b_y)$$
$$a_2 = \tanh(W_{a,a}a_1 + W_{x,a}x_2 + b_a)$$
$$y_2 = \mathrm{softmax}(W_{a,y}a_2 + b_y)$$
$$a_3 = \tanh(W_{a,a}a_2 + W_{x,a}x_3 + b_a)$$
$$y_3 = \mathrm{softmax}(W_{a,y}a_3 + b_y)$$
$$\ldots$$
$$a_N = \tanh(W_{a,a}a_{N-1} + W_{x,a}x_N + b_a)$$
$$y_N = \mathrm{softmax}(W_{a,y}a_N + b_y)$$

Tangente iperbolica

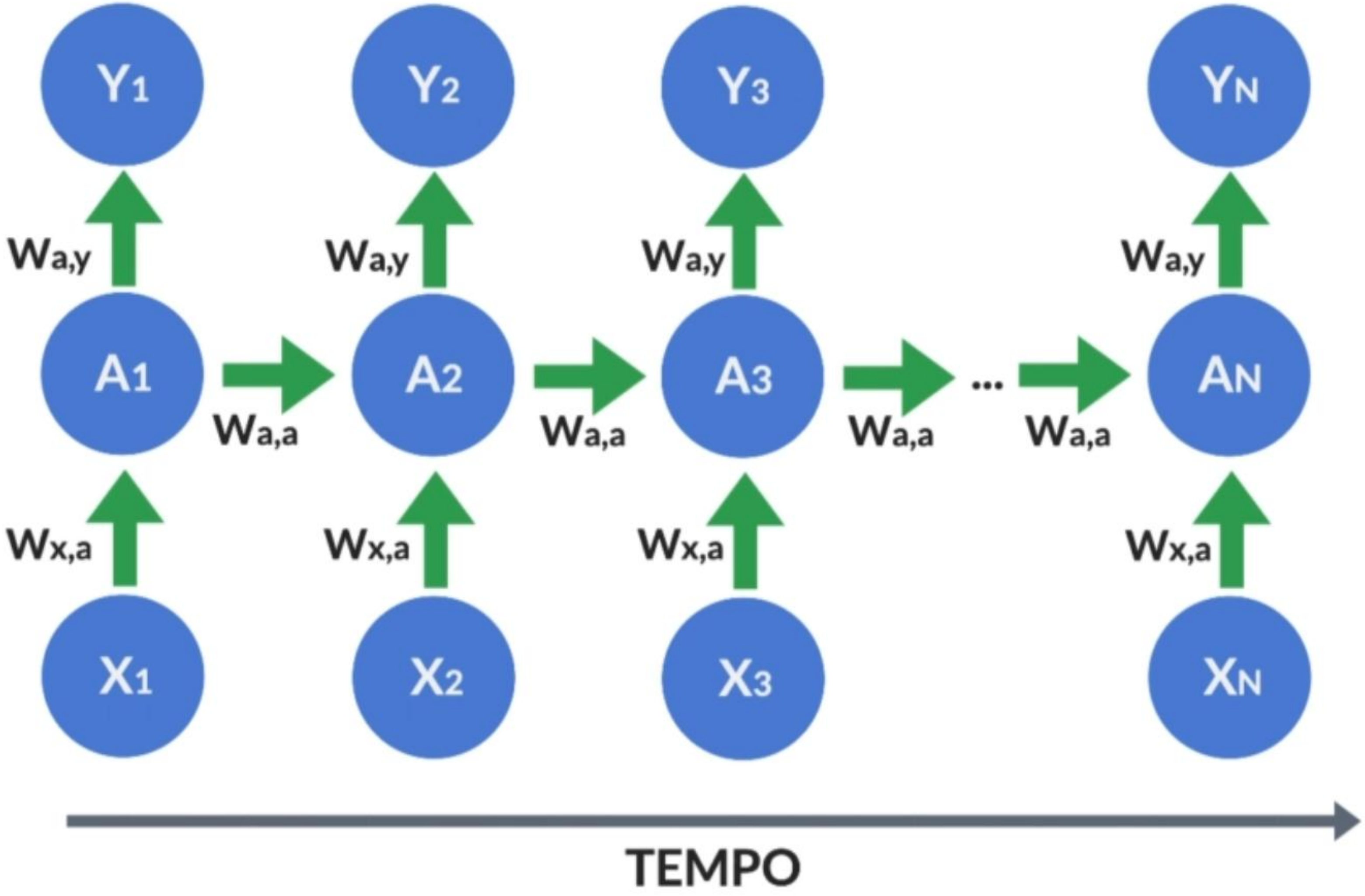


$$\phi(z) = \frac{1 - e^{-2z}}{1 + e^{-2z}}$$

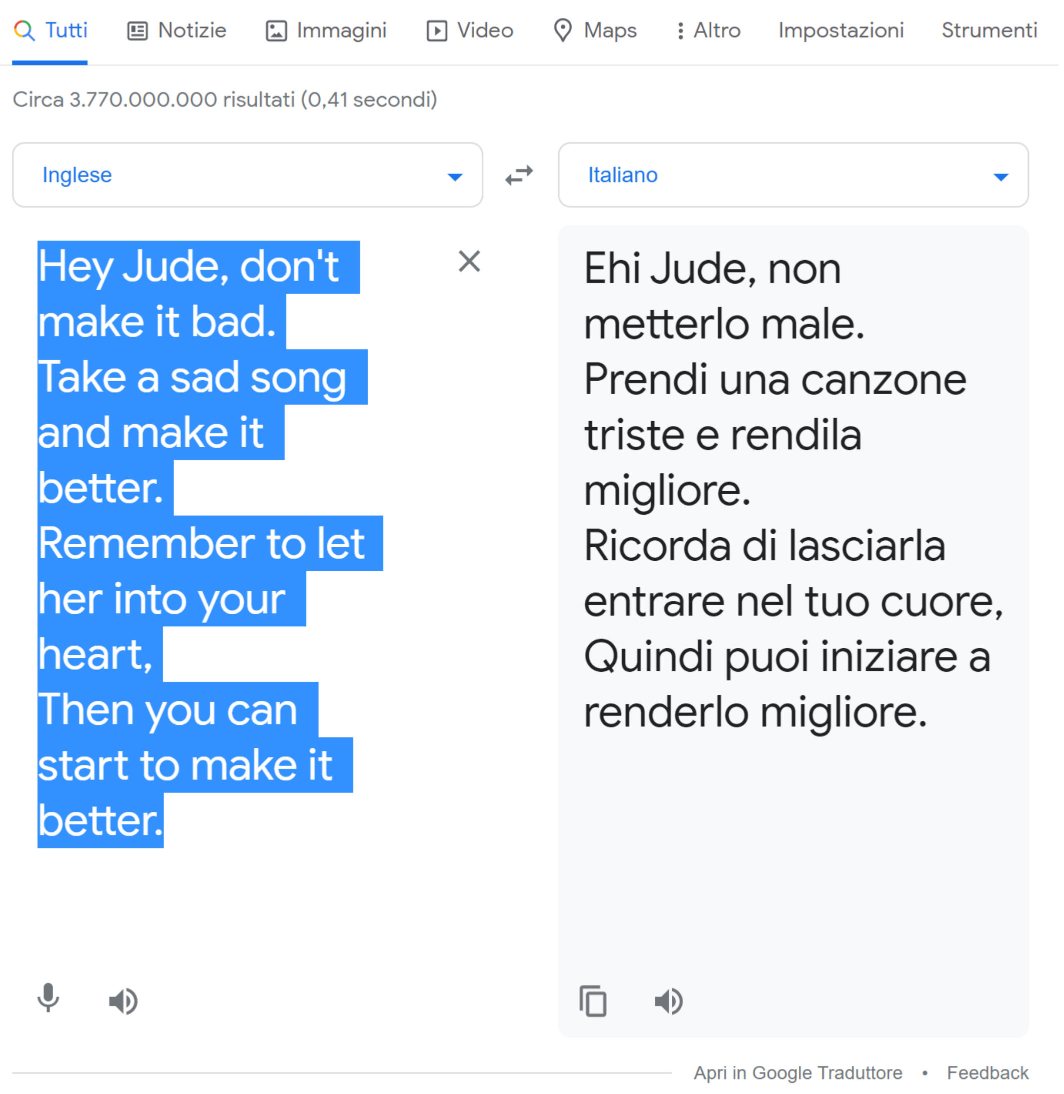
Una rete ricorrente è esposta al problema di scomparsa/esplosione del gradiente tra le diverse esecuzioni della stessa rete l'utilizzo della funzione di attivazione tangente iperbolica ci aiuta a limitare il problema (di più in seguito)

Relazione Uno a Uno

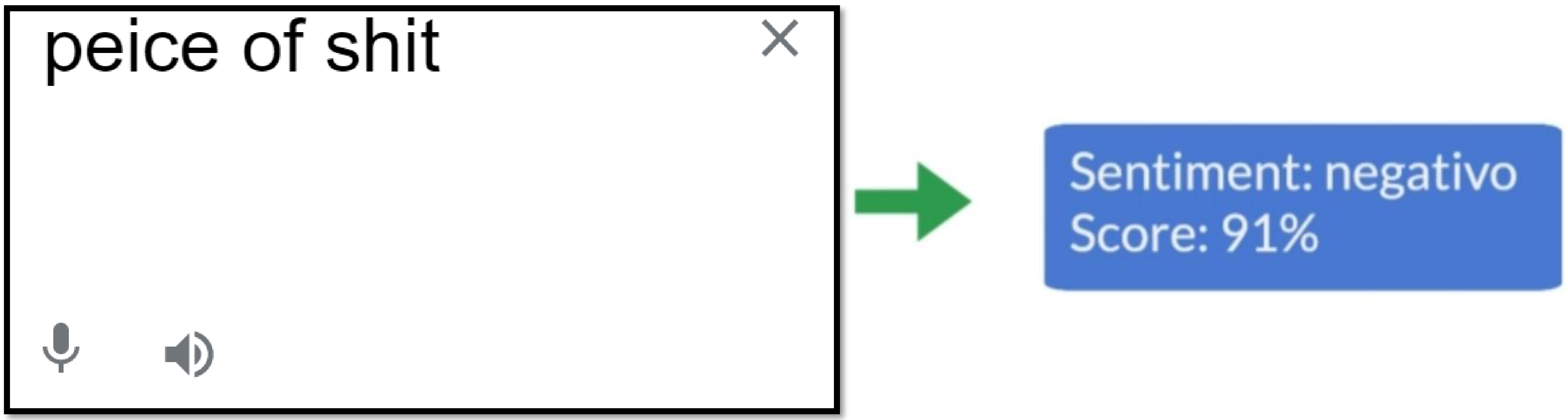
Relazione Molti a Molti



Relazione Molti a Molti



Relazione Molti a Uno



ESEMPIO

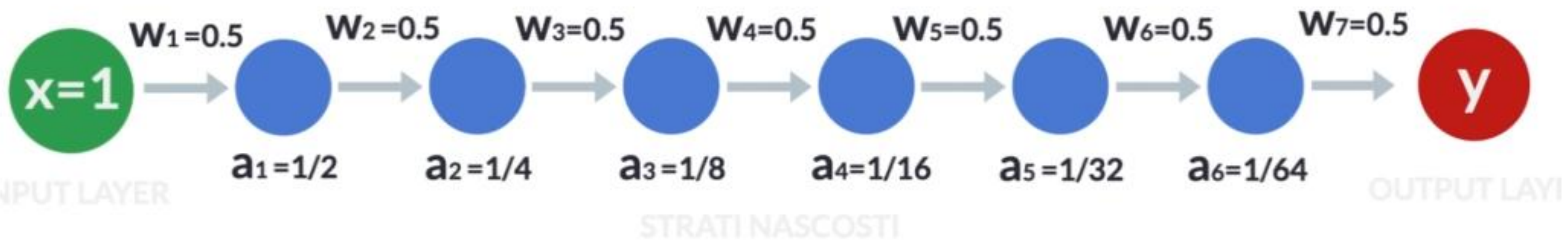
Sentiment Analysis

Backpropagation through time (BPTT)

Ci permette di addestrare una rete neurale ricorrente
propagando all'indietro il gradiente
non solo attraverso gli strati
ma anche attraverso le esecuzioni sequenziali della rete

Il problema del Vanishing Gradient nelle RNN

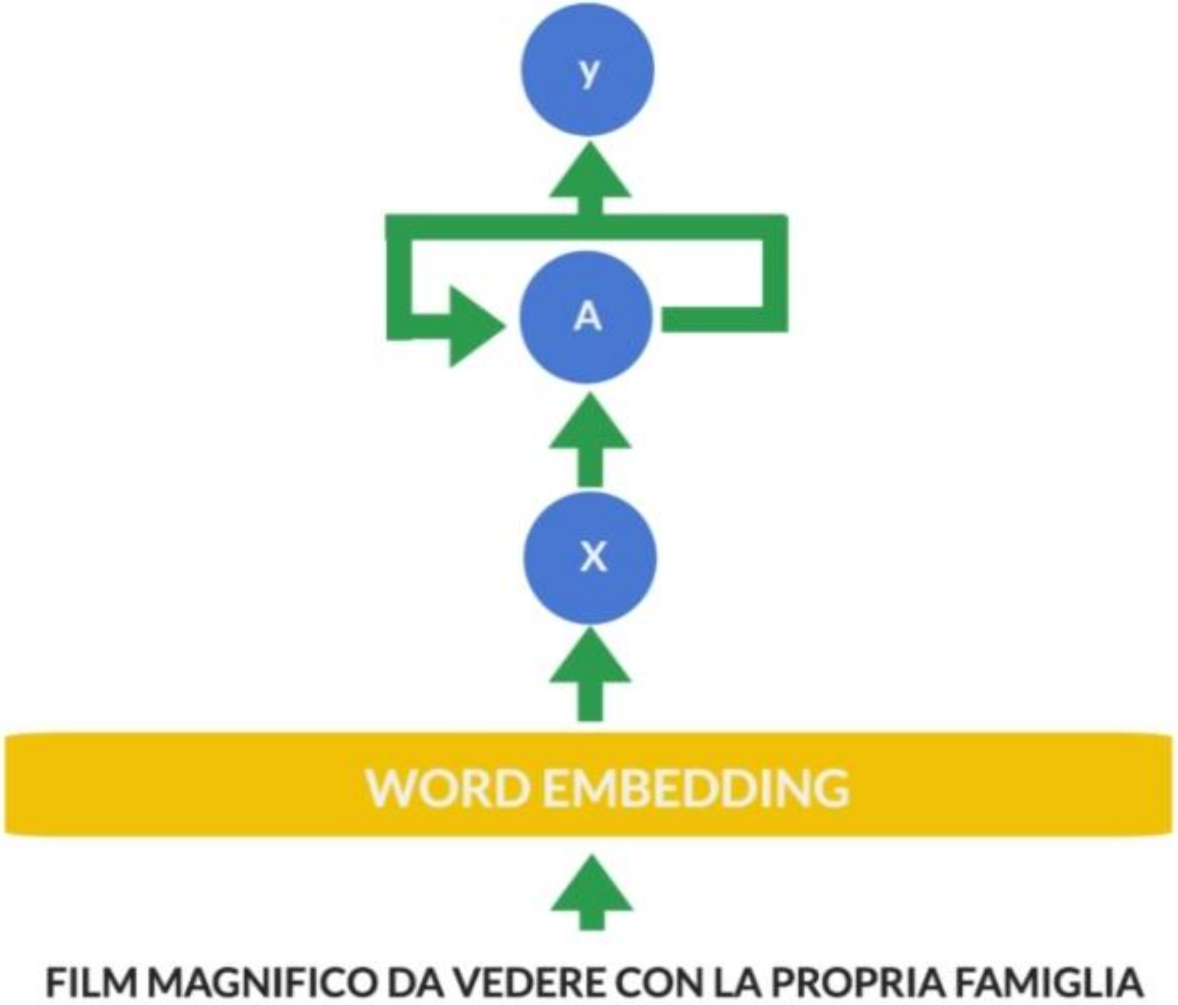
Il problema della scomparsa del Gradiente



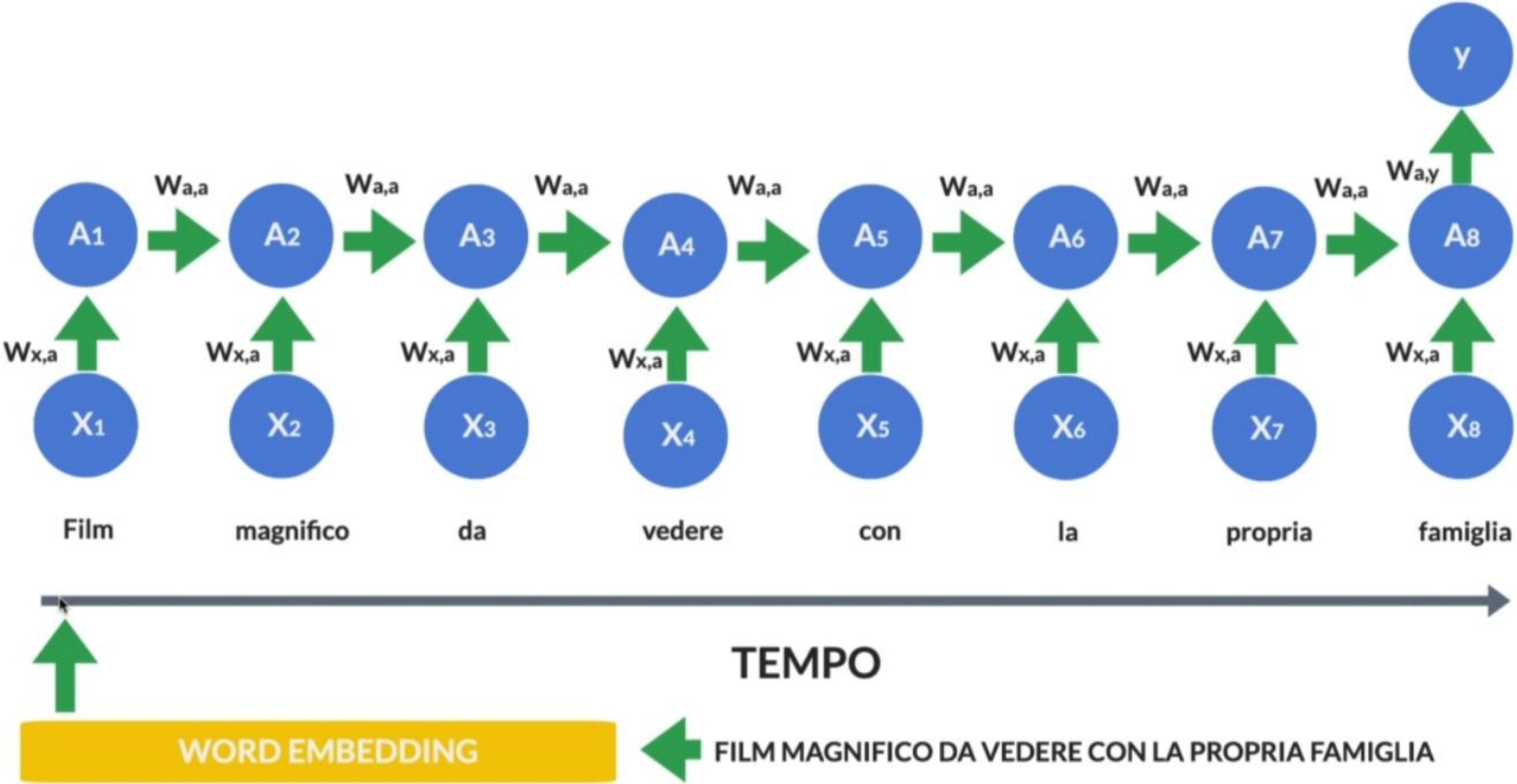
BACKWARD PROPAGATION

Effettuando moltiplicazioni ripetute per valori che tendono allo 0 il gradiente scomparirà (0).

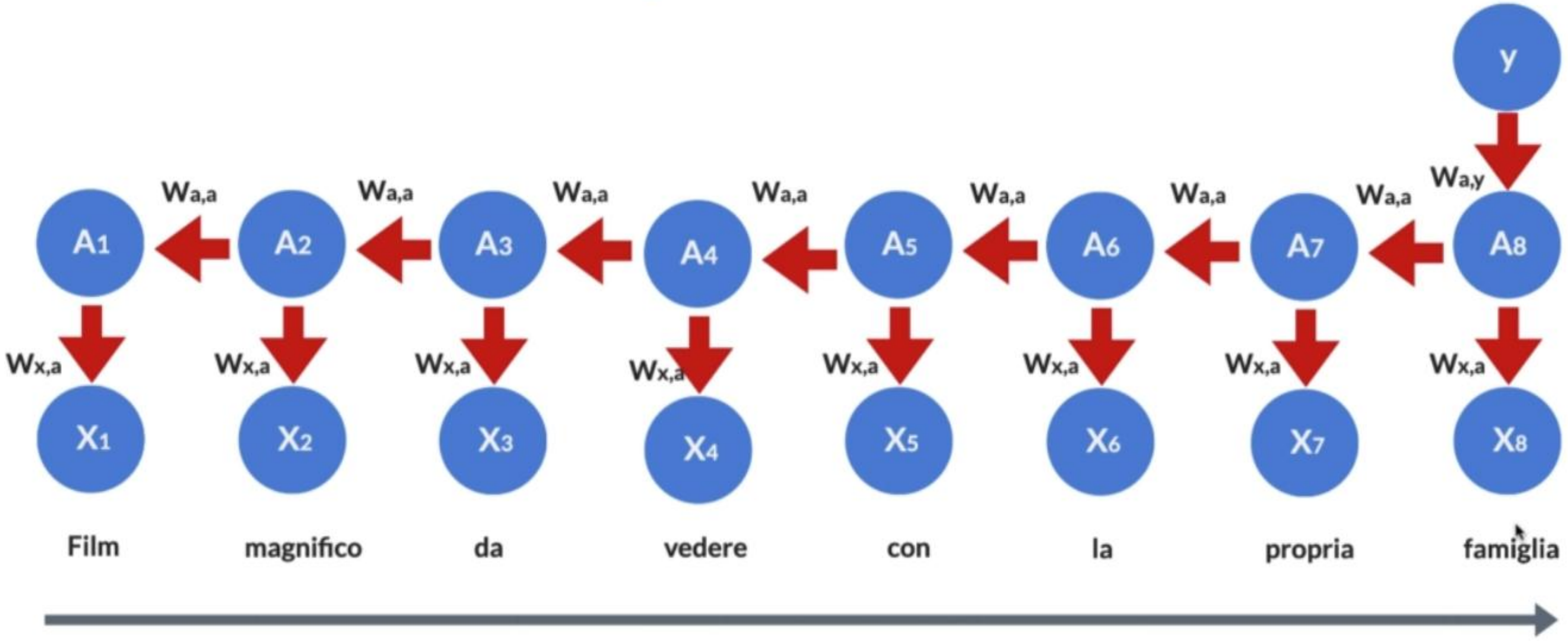
Vanishing Gradient e RNN



Vanishing Gradient e RNN



Vanishing Gradient e RNN

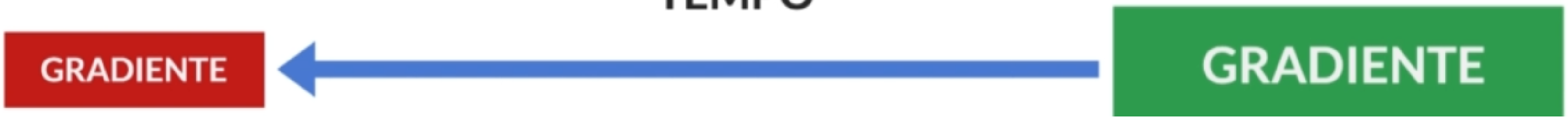


TEMPO

BACKPROPAGATION THROGH TIME

L'errore viene propagato non solo attraverso gli strati ma anche attraverso le esecuzioni

TEMPO



Le reti ricorrenti semplici (Vanilla RNN) vanno bene per brevi sequenze.

Ma a causa del problema della scomparsa del gradiente sono inadatte per sequenze più lunghe.

Understanding LSTM Networks

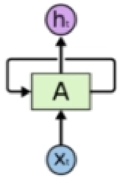
Posted on August 27, 2015

Recurrent Neural Networks

Humans don't start their thinking from scratch every second. As you read this essay, you understand each word based on your understanding of previous words. You don't throw everything away and start thinking from scratch again. Your thoughts have persistence.

Traditional neural networks can't do this, and it seems like a major shortcoming. For example, imagine you want to classify what kind of event is happening at every point in a movie. It's unclear how a traditional neural network could use its reasoning about previous events in the film to inform later ones. [5]

Recurrent neural networks address this issue. They are networks with loops in them, allowing information to persist. [1]



[8]

Recurrent Neural Networks have loops.

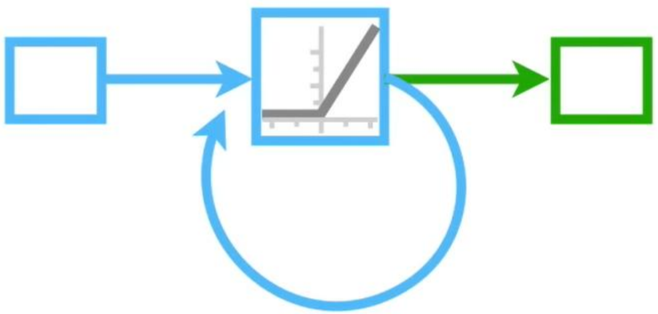
http://colah.github.io/posts/2015-08-Understanding-LSTMs/

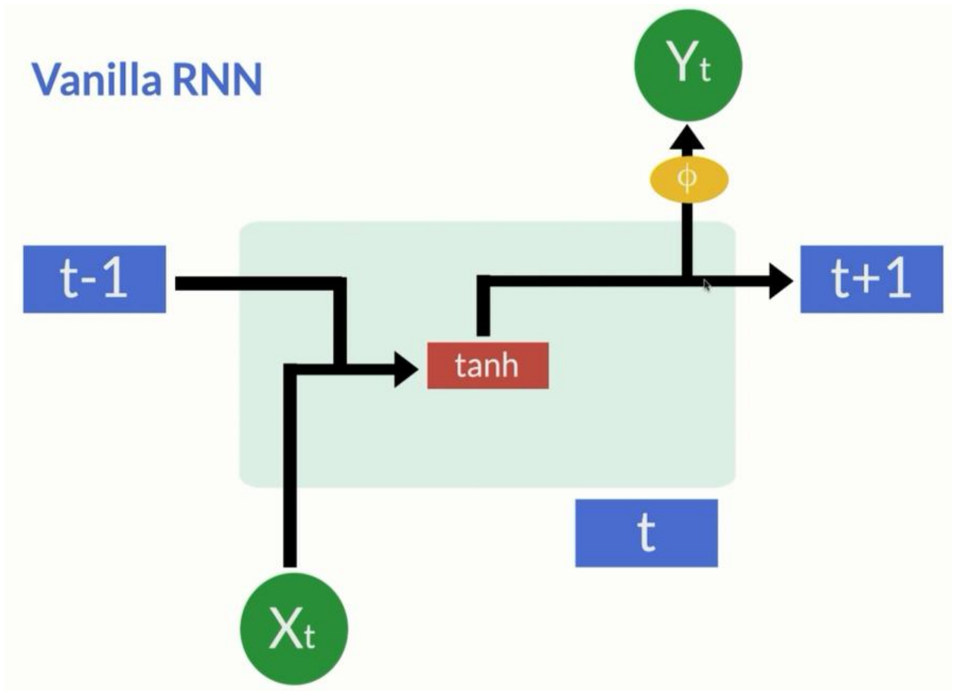
Cosa fa una LSTM ?

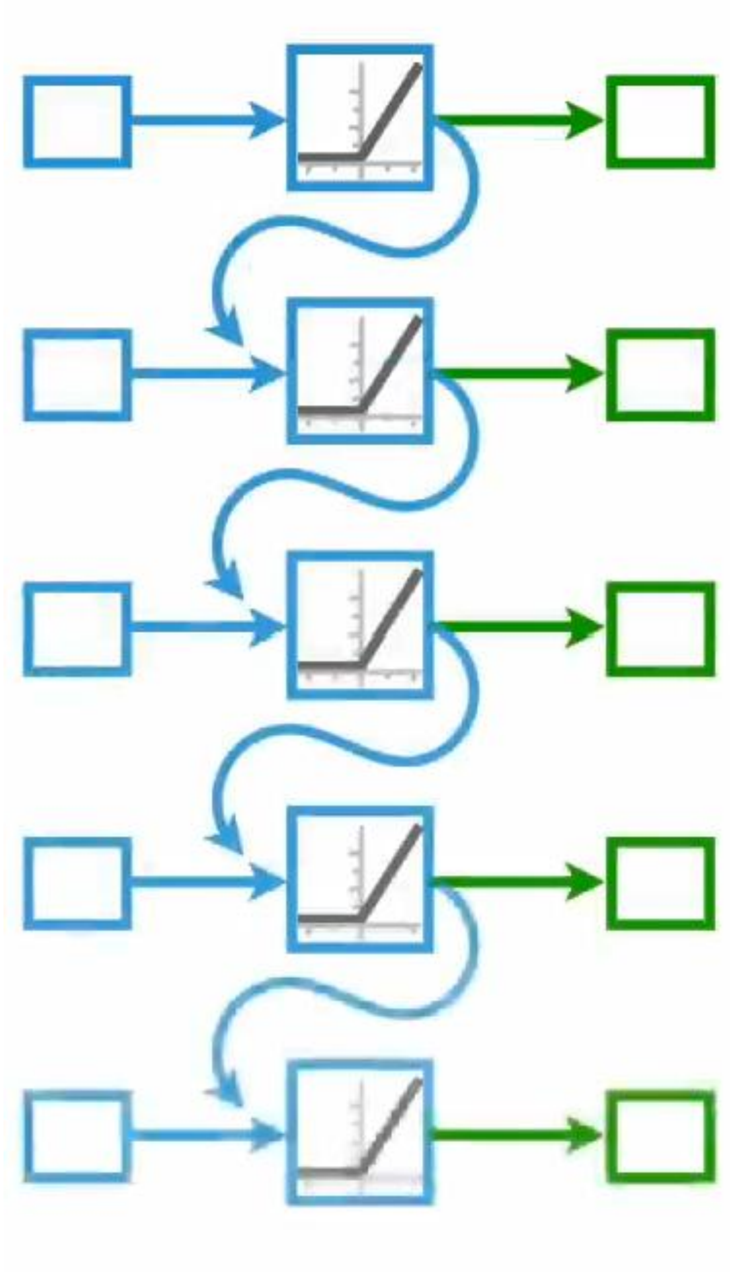
Un **capolavoro unico** ! Ho portato la mia ragazza al cinema per il suo compleanno, non avevo grosse aspettative ma devo ammettere che **ho amato questo film** fino all'ultima, anche se purtroppo mi sono perso i 5 minuti iniziali. **Lo consiglio a tutti.**

Una LSTM memorizza le informazioni importanti e scarta tutto il resto

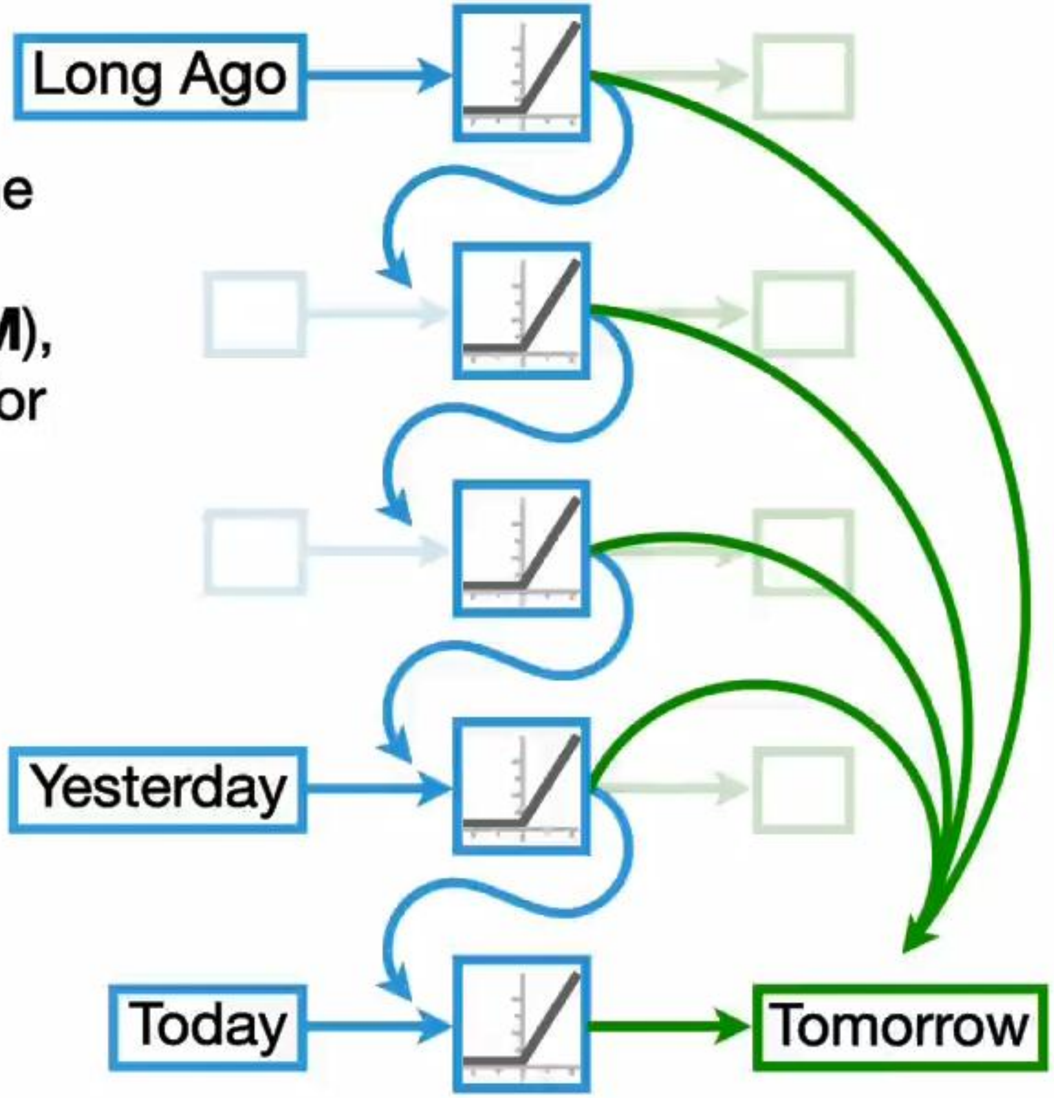
RNN

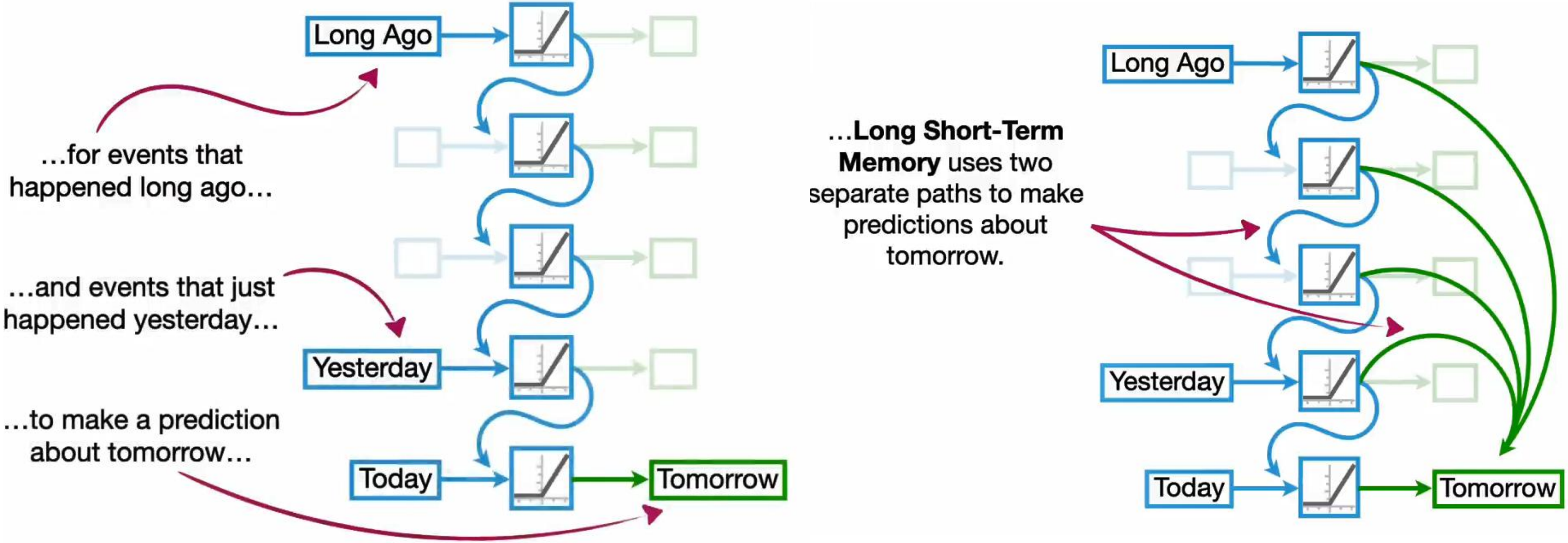






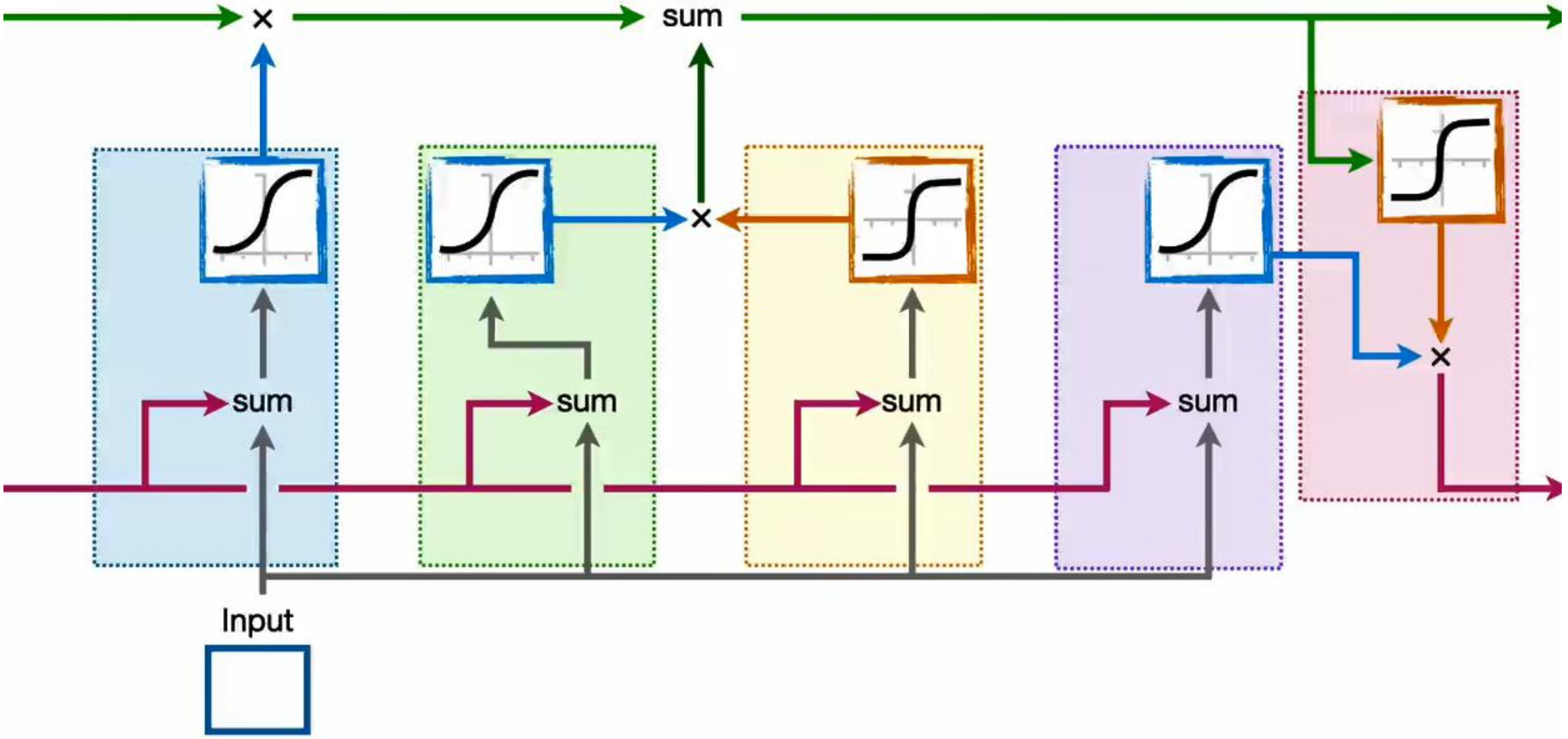
Now that we understand the **main idea** behind **Long Short-Term Memory (LSTM)**, that it uses different paths for **Long** and **Short-Term Memories**…



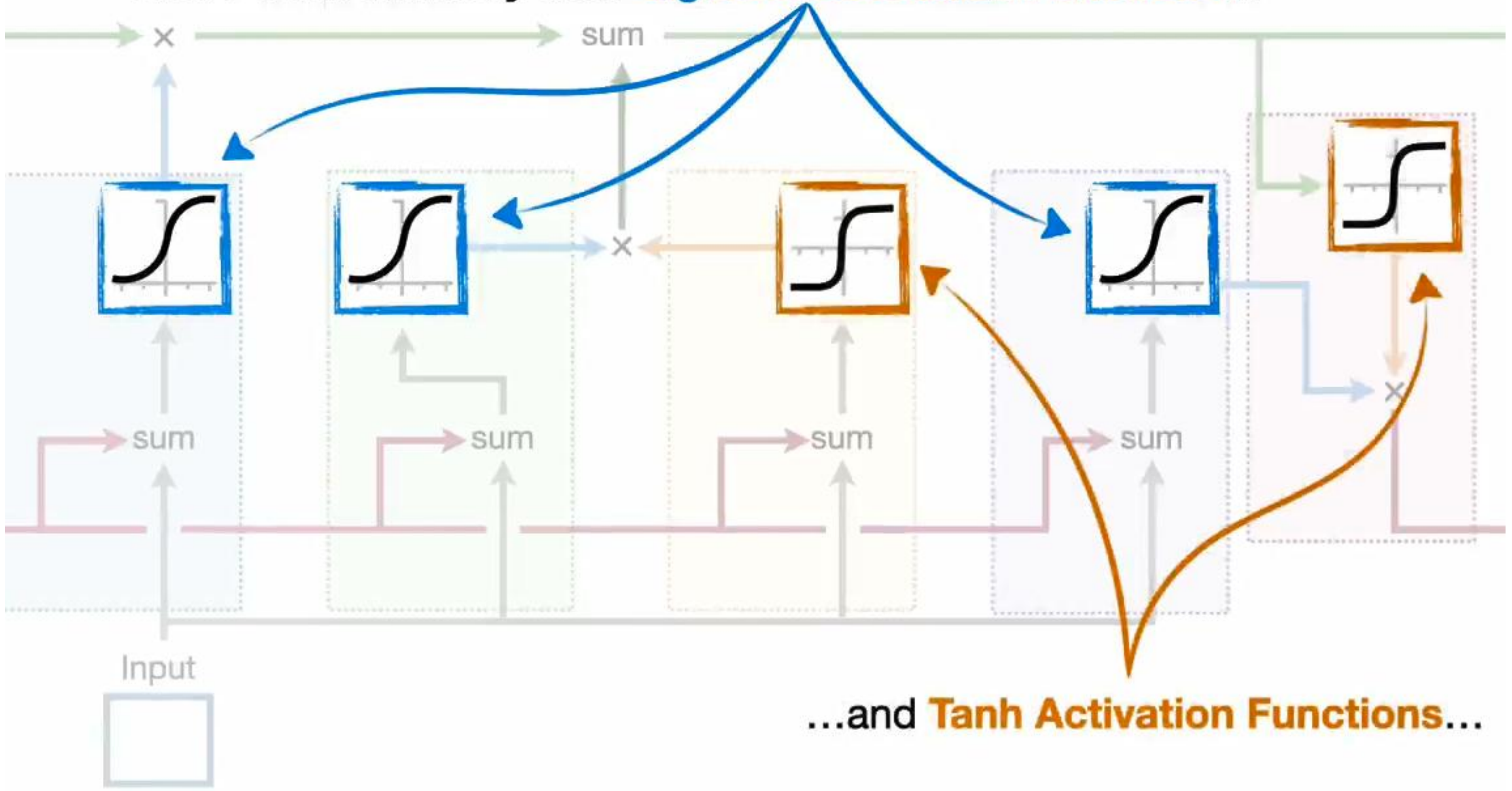


Long Ago
…for events that happened long ago…
…and events that just happened yesterday…
Yesterday
…to make a prediction about tomorrow…
Today
Tomorrow
…Long Short-Term Memory uses two separate paths to make predictions about tomorrow.
Long Ago
Yesterday
Today
Tomorrow

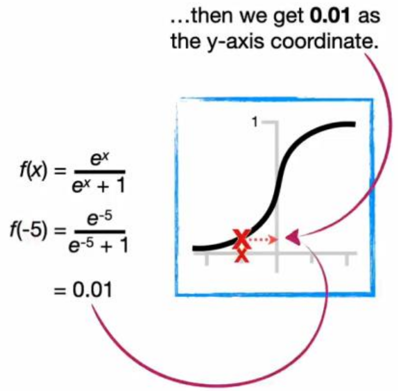
...**Long Short-Term Memory** is based on a much more complicated unit.



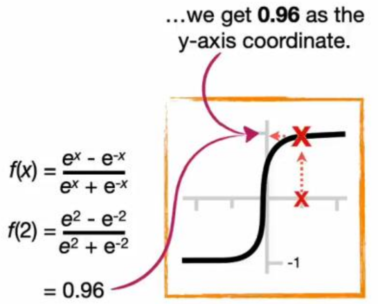
NOTE: Unlike the networks we've used before in this series, **Long Short-Term Memory** uses **Sigmoid Activation Functions**…



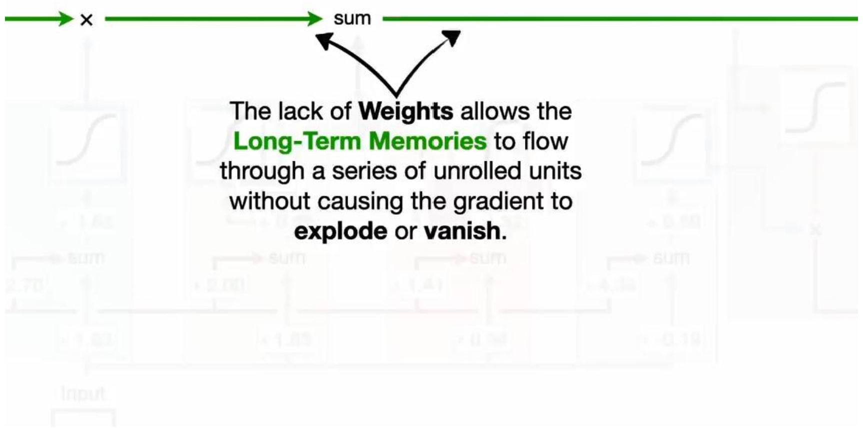
…and **Tanh Activation Functions**…



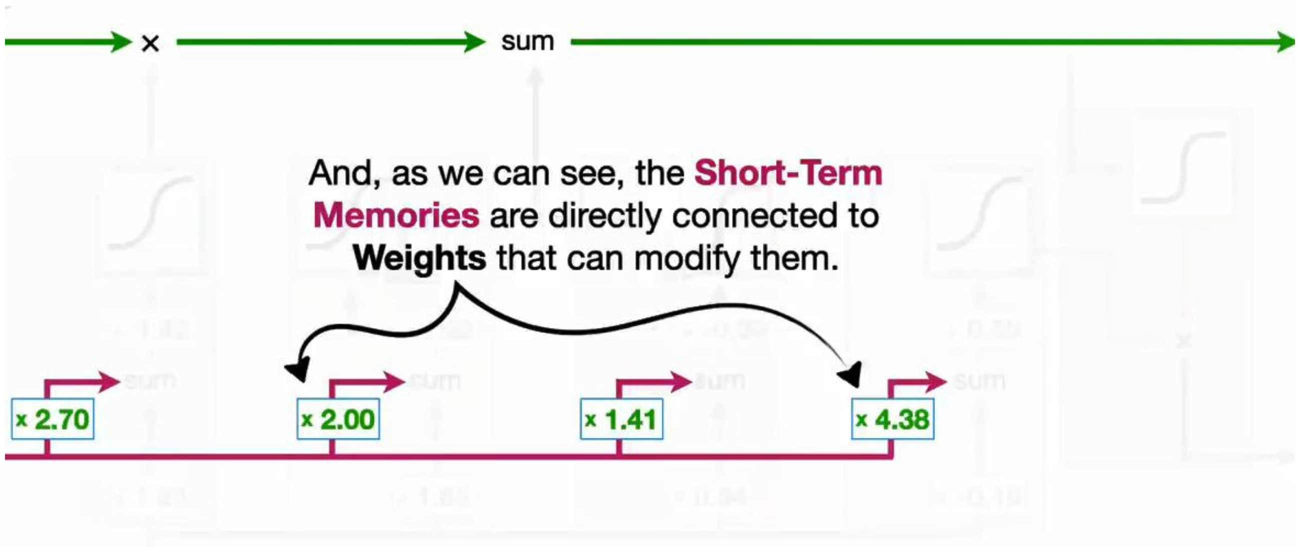
...then we get **0.01** as the y-axis coordinate.
$f(x) = \frac{e^x}{e^x + 1}$
$f(-5) = \frac{e^{-5}}{e^{-5} + 1}$
$= 0.01$
1



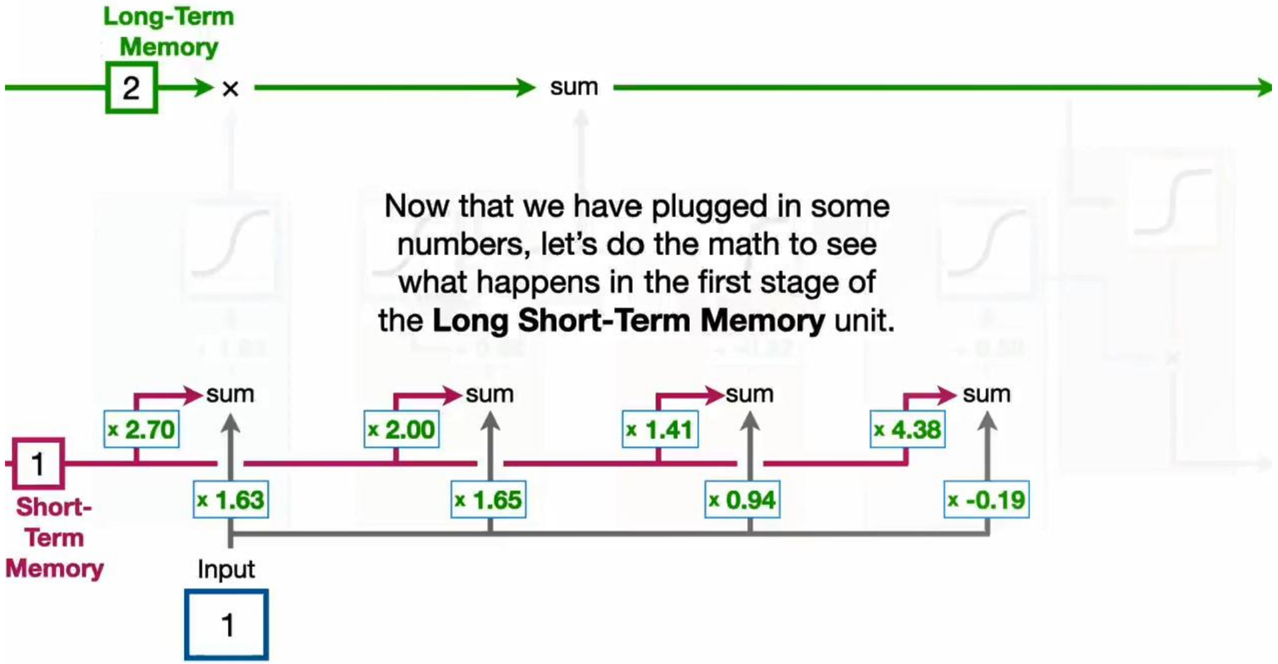
...we get **0.96** as the y-axis coordinate.
$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$
$f(2) = \frac{e^2 - e^{-2}}{e^2 + e^{-2}}$
$= 0.96$
-1



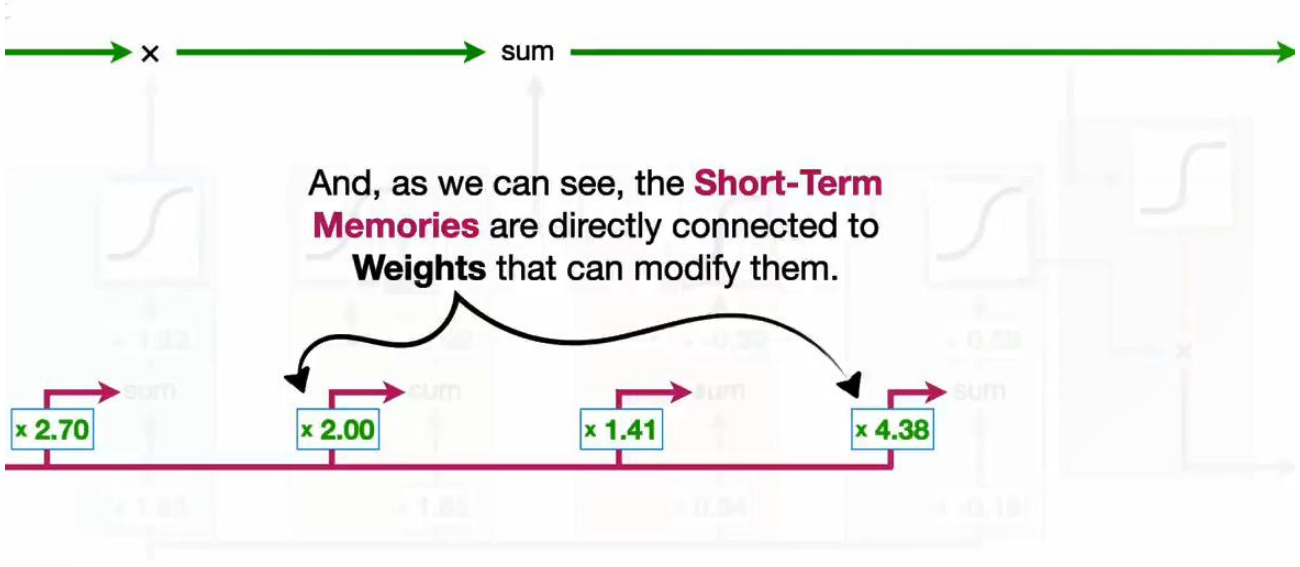
×
sum
The lack of **Weights** allows the **Long-Term Memories** to flow through a series of unrolled units without causing the gradient to **explode** or **vanish**.



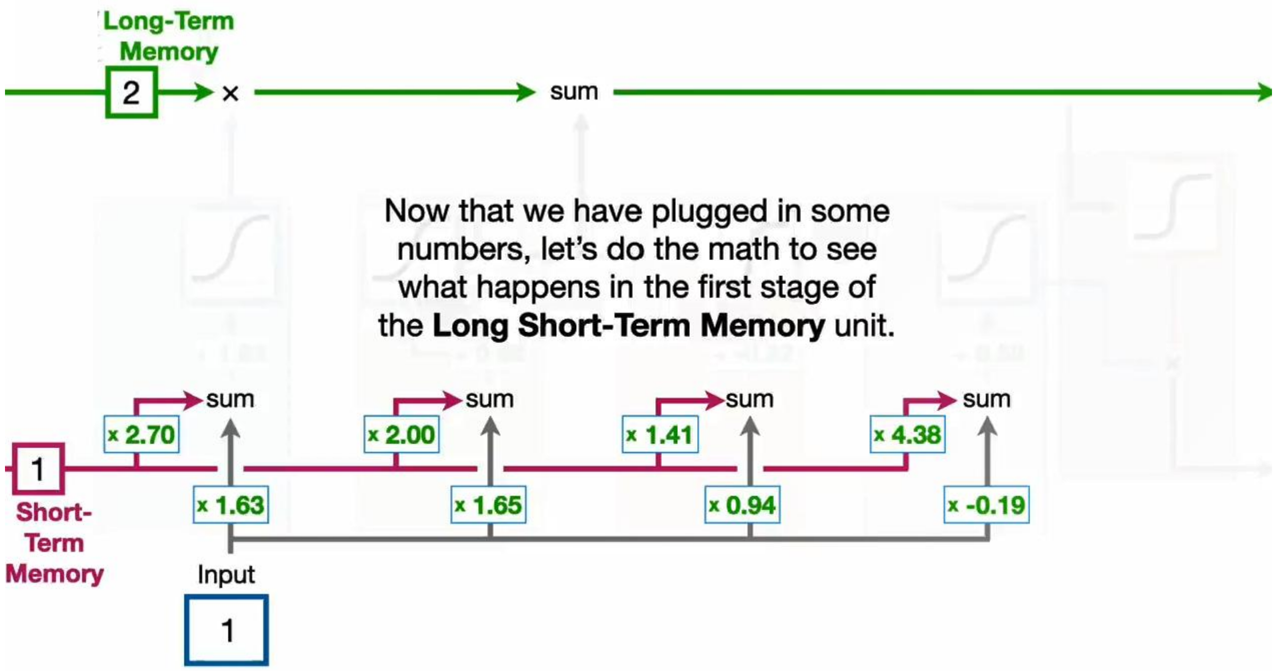
×
sum
And, as we can see, the **Short-Term Memories** are directly connected to **Weights** that can modify them.
x 2.70
x 2.00
x 1.41
x 4.38



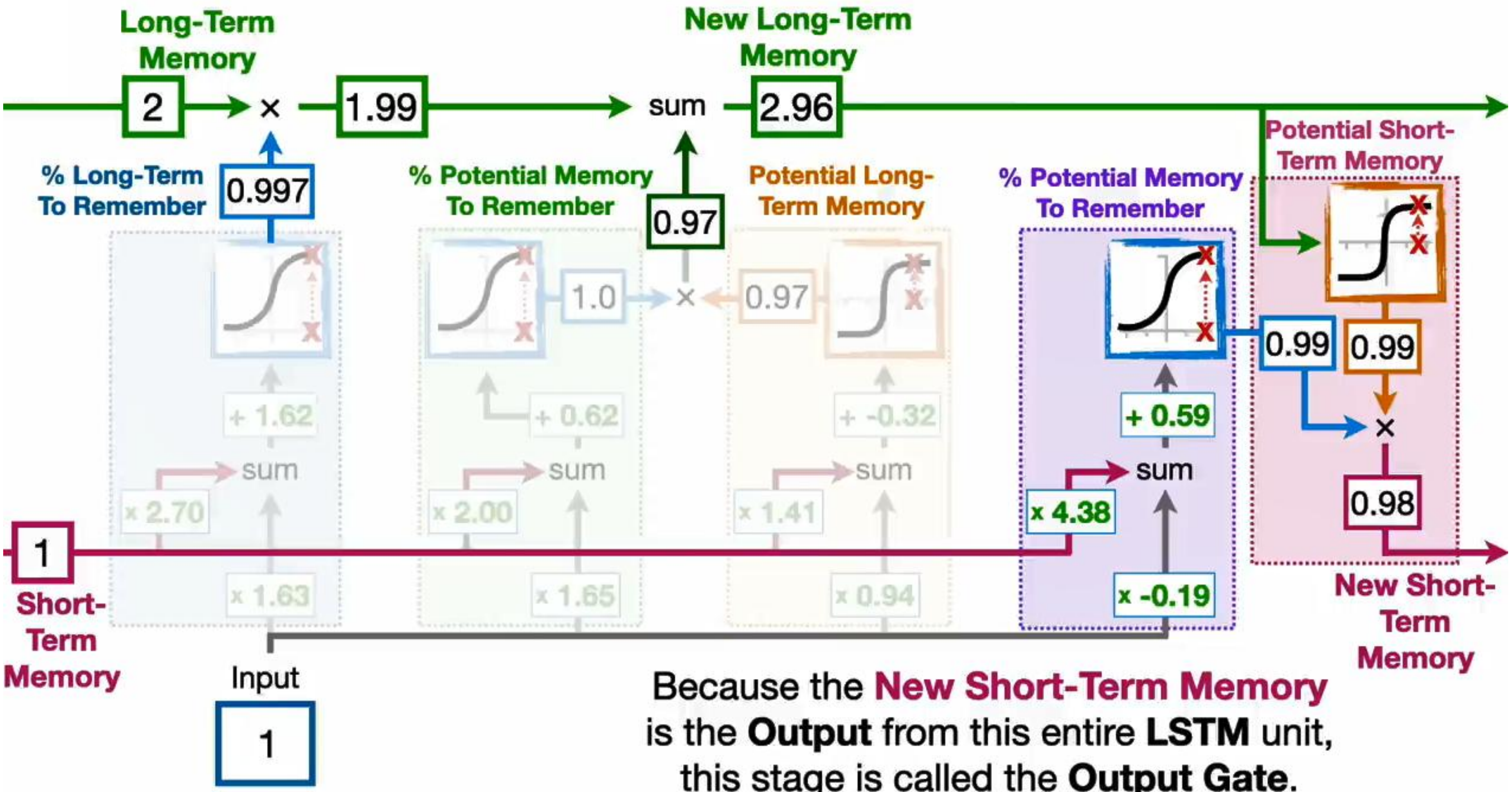
Long-Term Memory
2
×
sum
Now that we have plugged in some numbers, let's do the math to see what happens in the first stage of the **Long Short-Term Memory** unit.
sum
sum
sum
sum
x 2.70
x 2.00
x 1.41
x 4.38
1
x 1.63
x 1.65
x 0.94
x -0.19
Short-Term Memory
Input
1



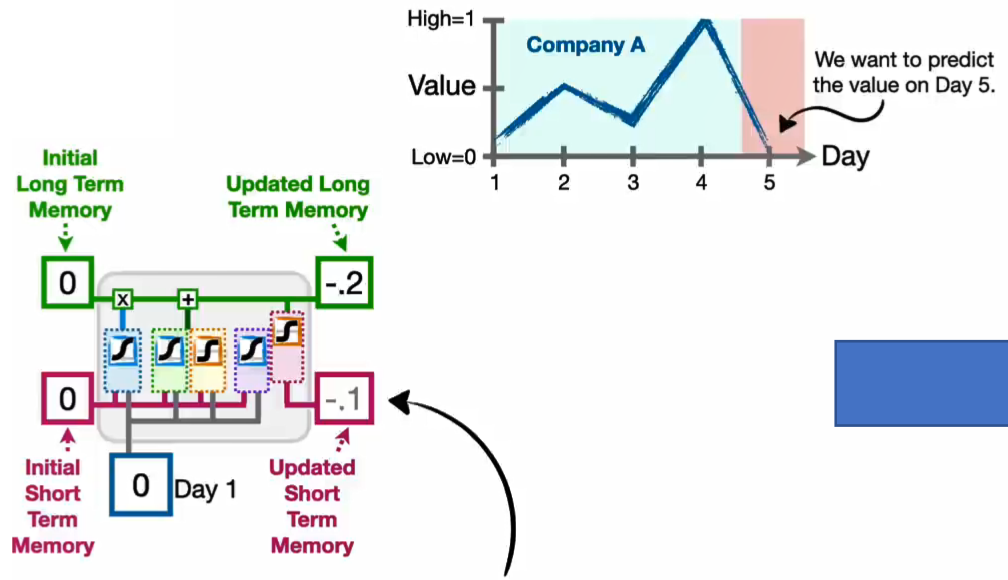
×
sum
And, as we can see, the Short-Term Memories are directly connected to Weights that can modify them.
x 2.70
x 2.00
x 1.41
x 4.38



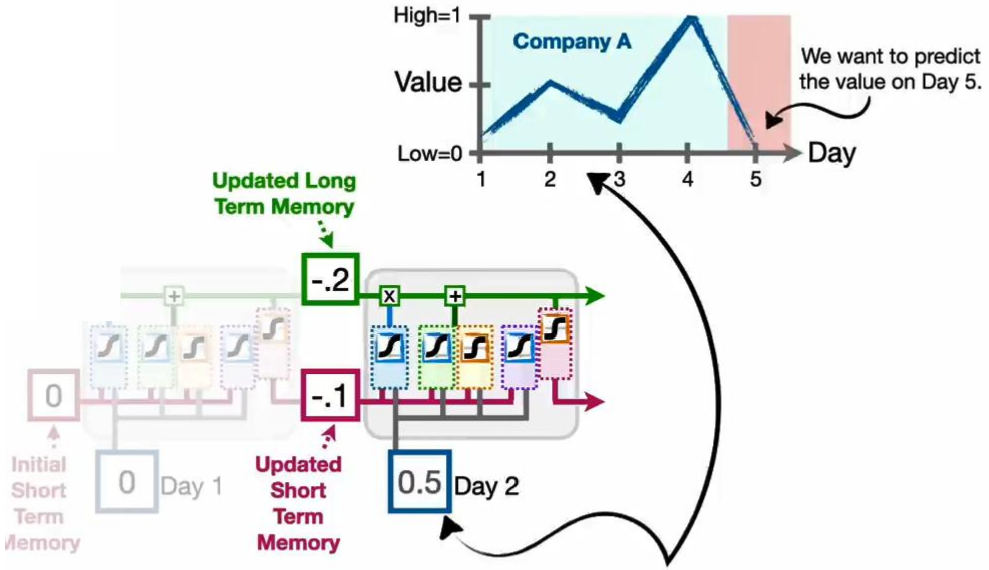
Long-Term Memory
2
×
sum
Now that we have plugged in some numbers, let's do the math to see what happens in the first stage of the Long Short-Term Memory unit.
sum
sum
sum
sum
x 2.70
x 2.00
x 1.41
x 4.38
1
Short-Term Memory
x 1.63
x 1.65
x 0.94
x -0.19
Input
1



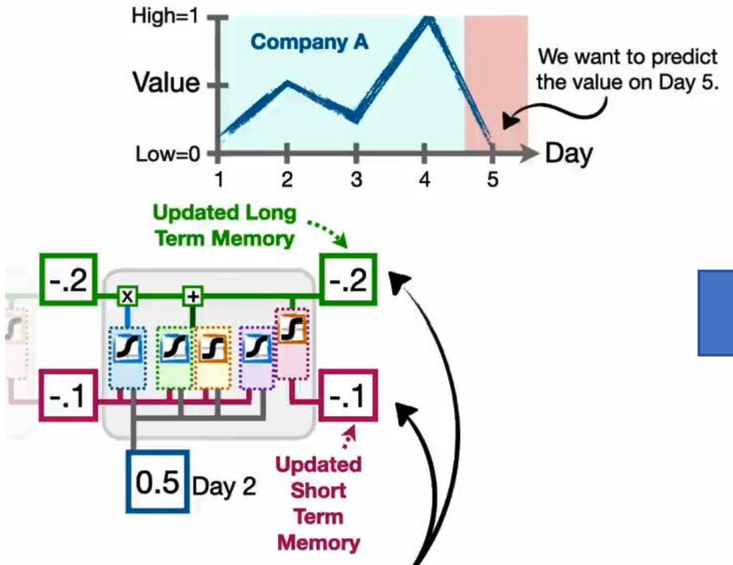
Because the **New Short-Term Memory** is the **Output** from this entire **LSTM** unit, this stage is called the **Output Gate**.



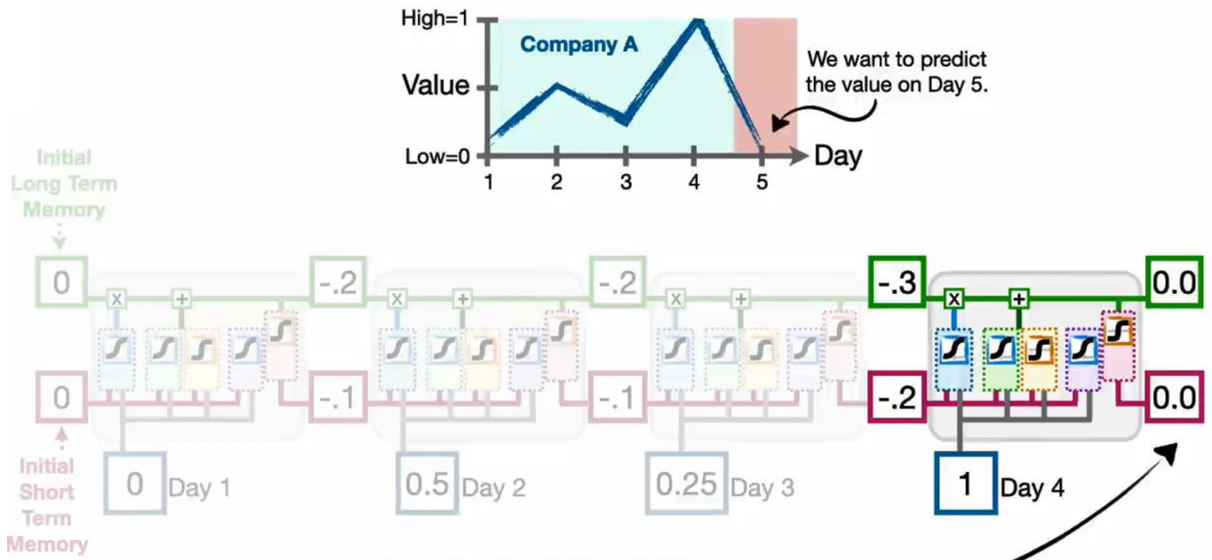
...and **-0.1** (rounded) for the updated **Short-Term Memory**.



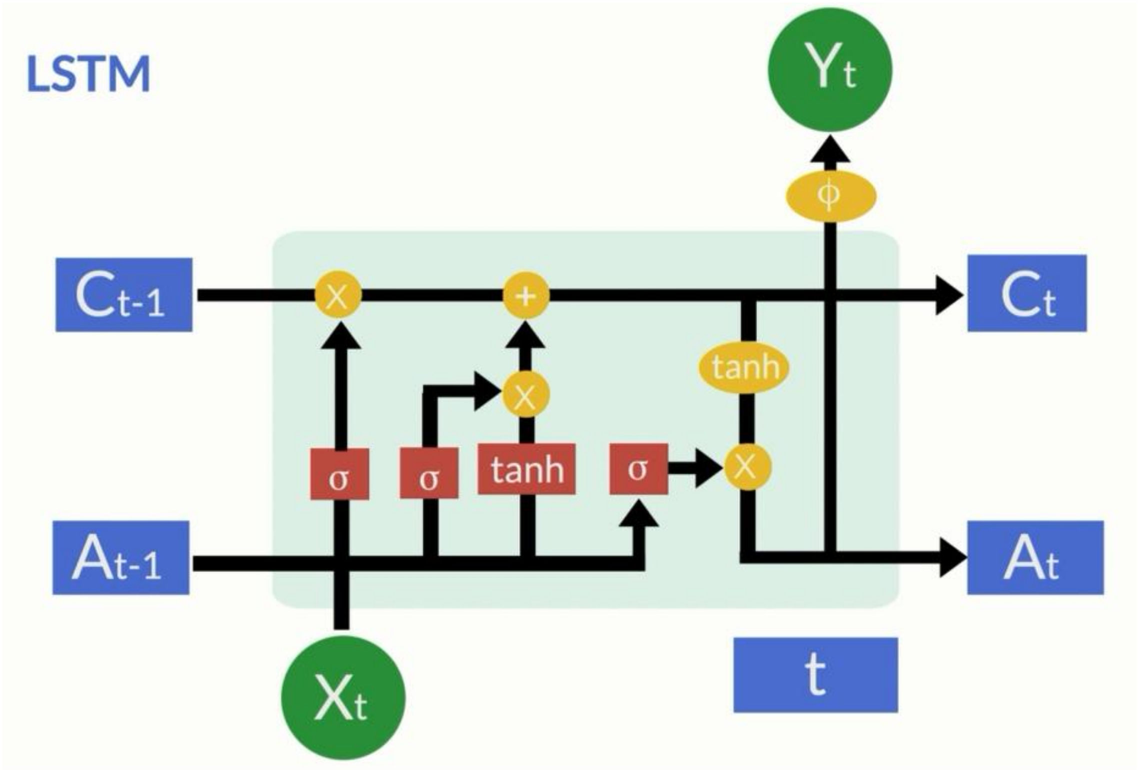
...and plug the value from **Day 2, 0.5,** into the **Input**.

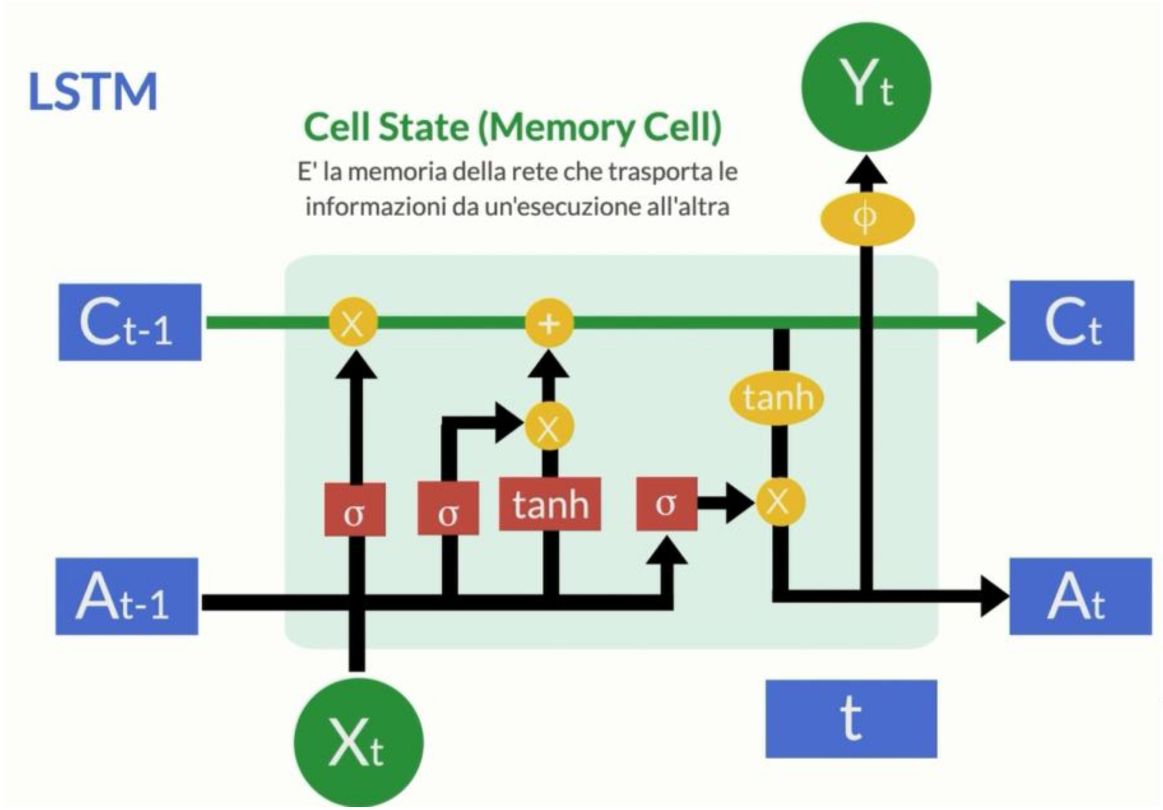


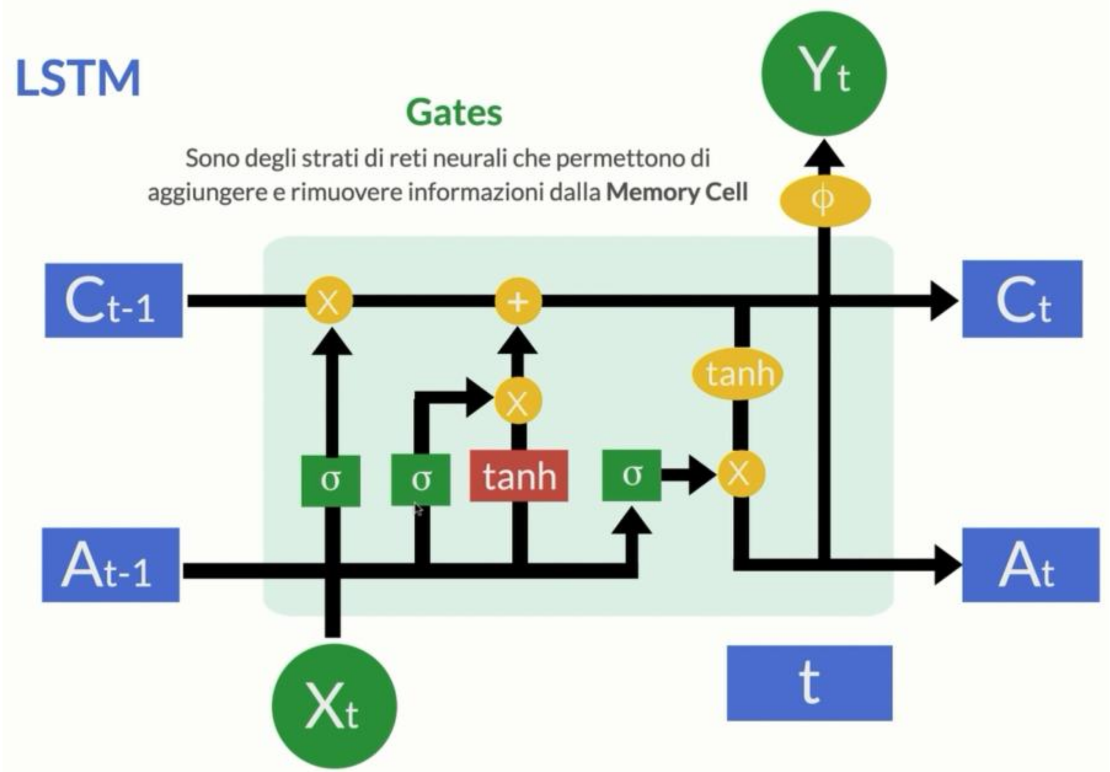
...and we end up with these updated **Long and Short-Term Memories.**

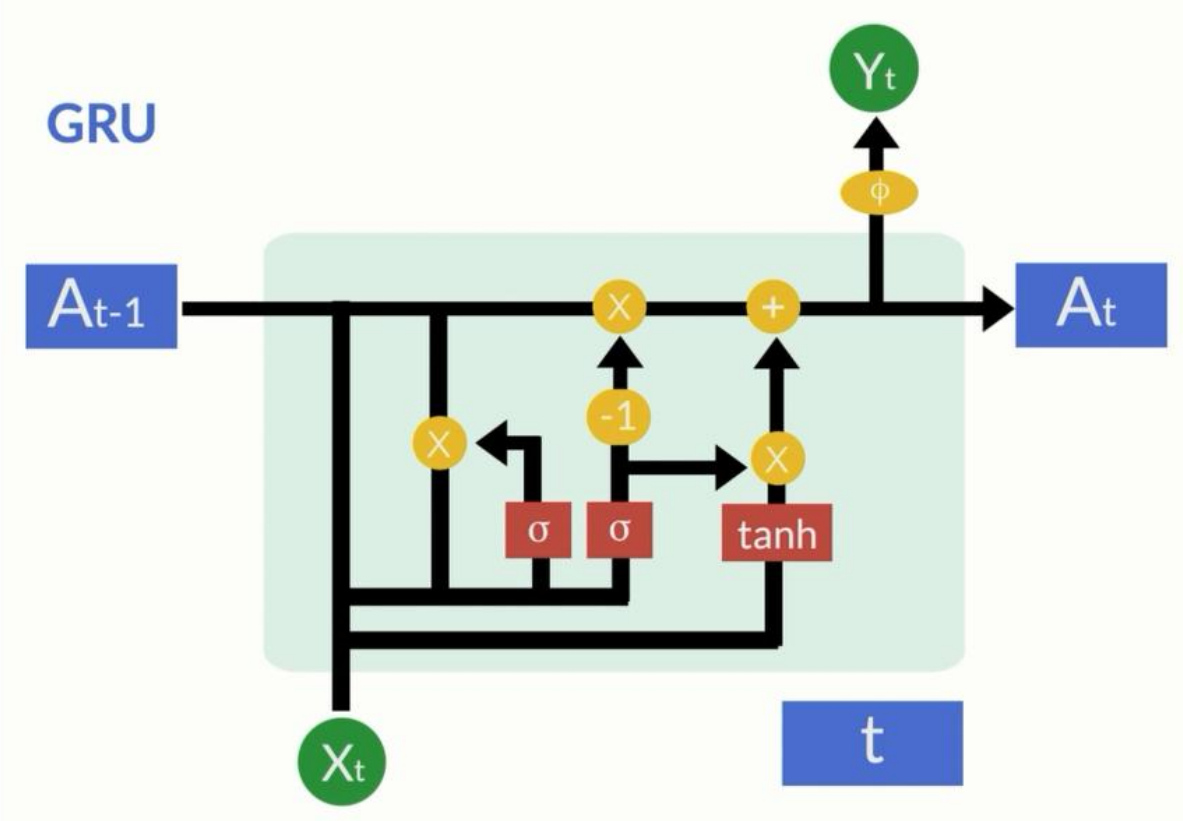


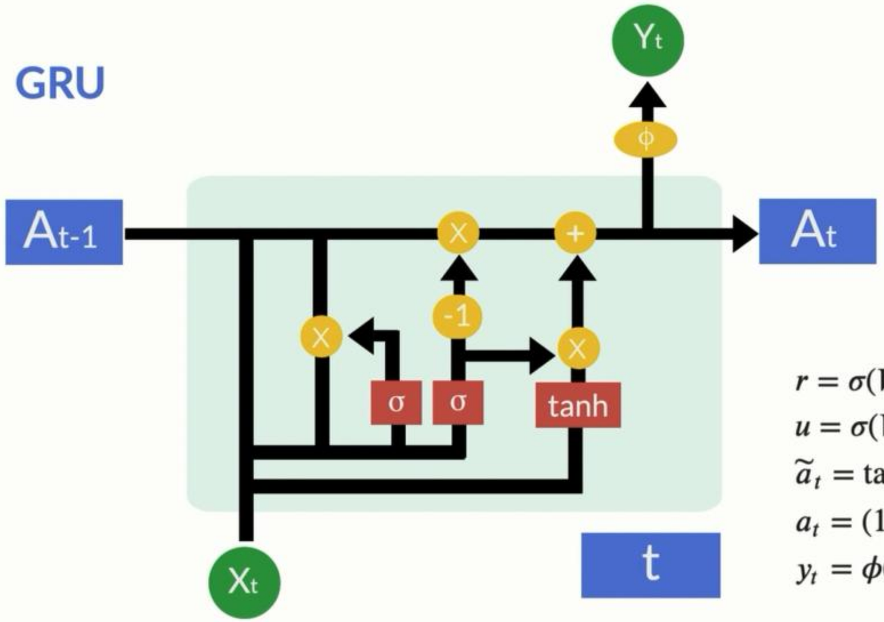
And the final **Short-Term Memory**, **0.0**, is the **Output** from the unrolled **LSTM**.


LSTM
Y_t
ϕ
C_{t-1}
C_t
tanh
σ
σ
tanh
σ
A_{t-1}
A_t
X_t
t


LSTM
Cell State (Memory Cell)
E' la memoria della rete che trasporta le informazioni da un'esecuzione all'altra
Y_t
ϕ
C_{t-1}
C_t
tanh
σ
σ
tanh
σ
A_{t-1}
A_t
X_t
t


LSTM
Gates
Sono degli strati di reti neurali che permettono di aggiungere e rimuovere informazioni dalla **Memory Cell**
Yt
φ
Ct-1
Ct
X
+
tanh
σ
σ
tanh
σ
X
At-1
At
Xt
t


GRU
Yt
φ
At-1
At
X
+
-1
X
X
σ
σ
tanh
Xt
t



RIEPILOGO

$$r = \sigma(W_r \cdot [a_{t-1}, x_t] + b_r)$$
$$u = \sigma(W_u \cdot [a_{t-1}, x_t] + b_u)$$
$$\widetilde{a}_t = \tanh(W_a \cdot [r * a_{t-1}, x_t] + b_a$$
$$a_t = (1 - u) * a_{t-1} + u * \widetilde{a}_t$$
$$y_t = \phi(a_t)$$

GRU vs LSTM

GRU: Richiedono meno calcoli tra tensori, quindi l'addestramento richiede meno tempo.

LSTM: Solitamente portano a risultati migliori e ricordano sequenze più lunghe.