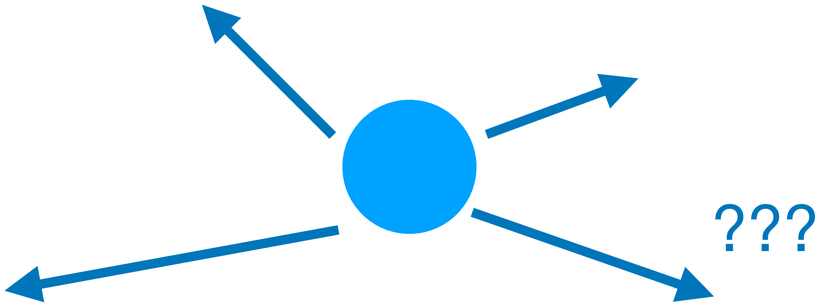
RNN

Dada la imagen de una pelota,

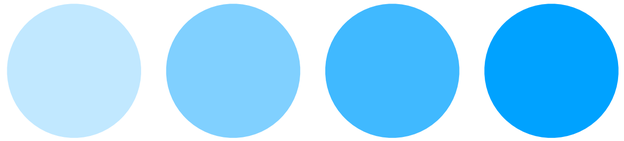
¿Puedes predecir a dónde irá a continuación?

RNN



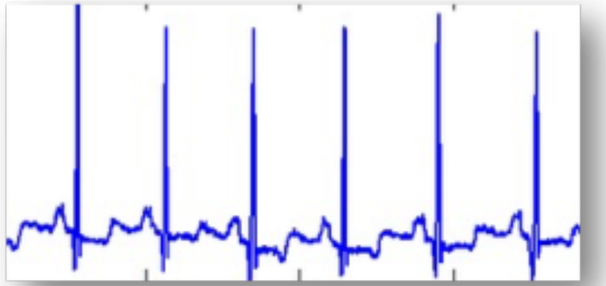
Sin información previa no sabemos hacia dónde va

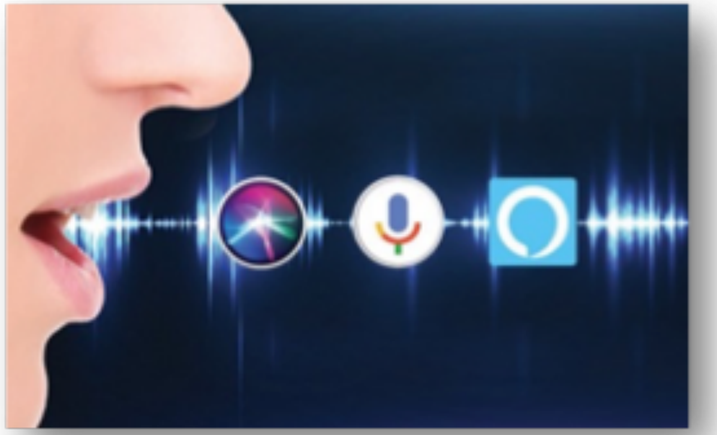
RNN

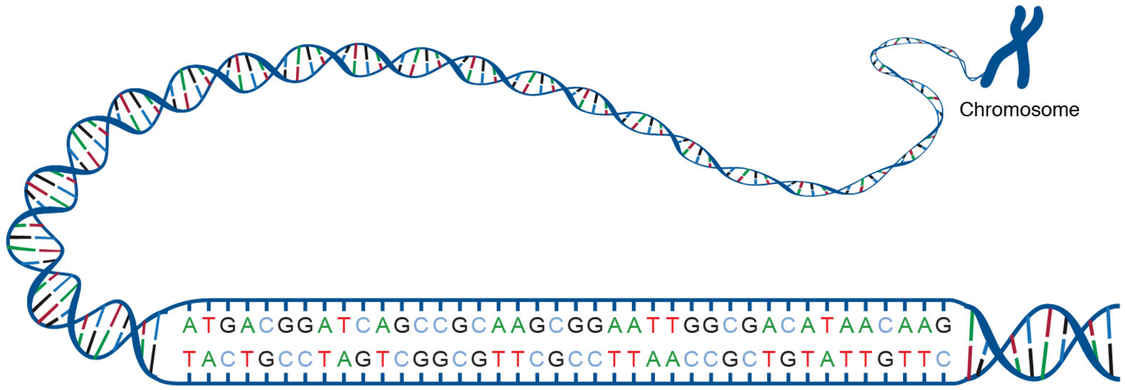


Si tenemos información de posiciones anteriores, podemos predecir mejor


Google
Translate



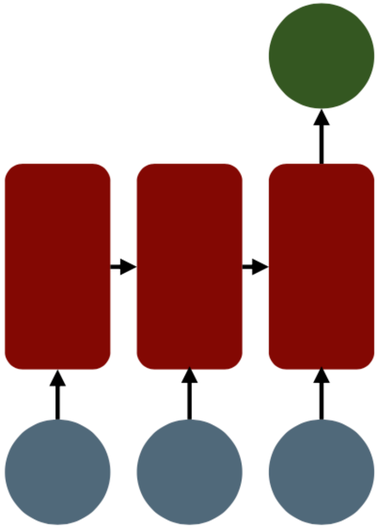



Chromosome
ATGACGGATCAGCCGCAAGCGGAATTGGCGACATAACAAG
TACTGCCTAGTCGGCGTTCGCCTTAACCGCTGTATTGTTC

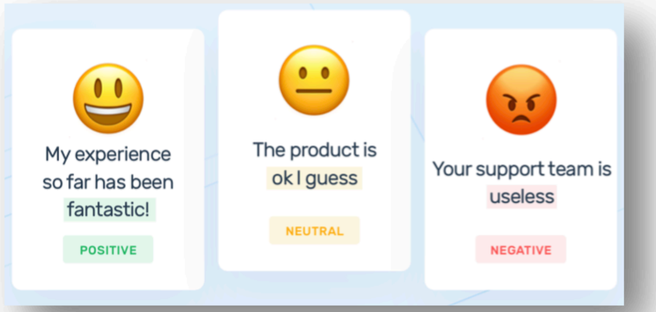
Aplicaciones

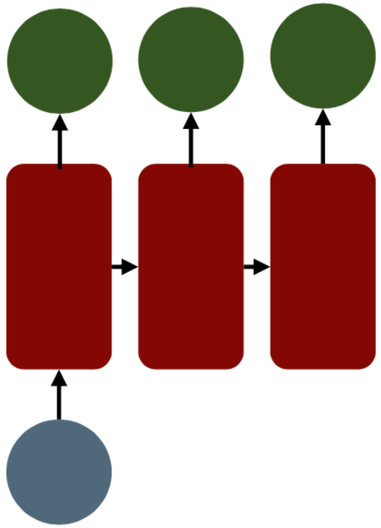
1 entrada, 1 salida
Clasificación binaria
(Redes neuronales totalmente conectadas)

Problema estático, no requiere información previa

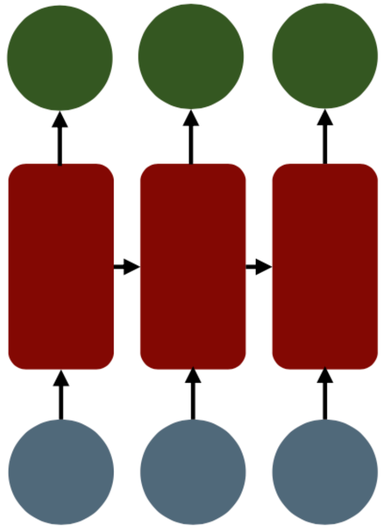


Varias entradas, 1 salida
Clasificación del sentimiento



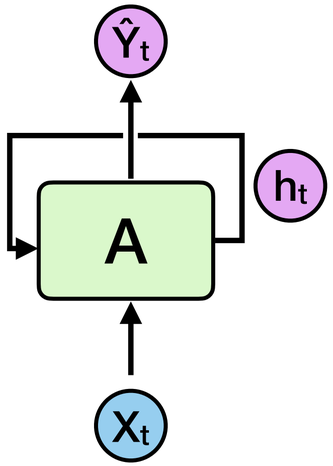


1 entrada, varias salidas
Descripción de una imagen

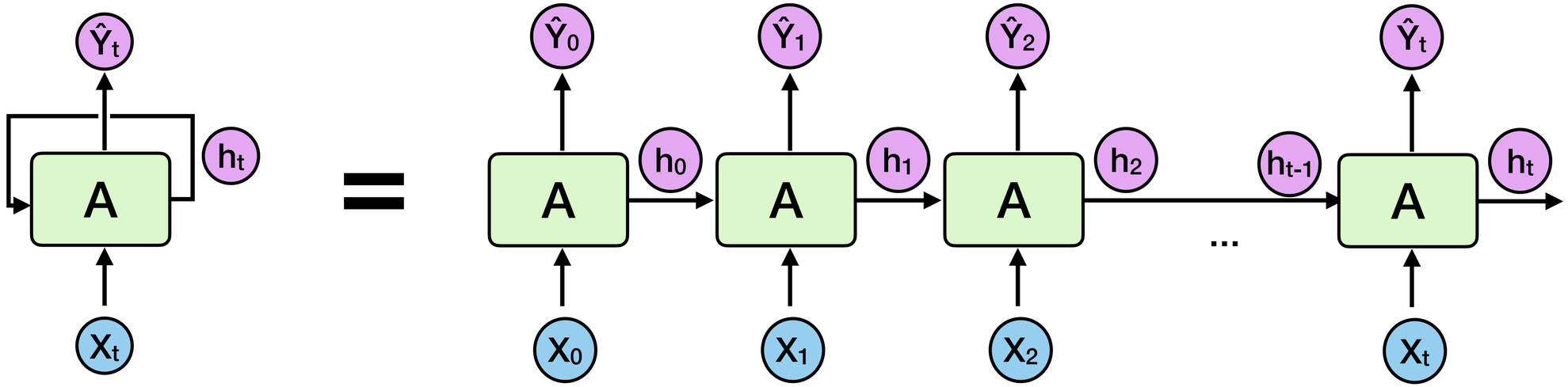


Varias entradas, varias salidas
Traducción

RNN

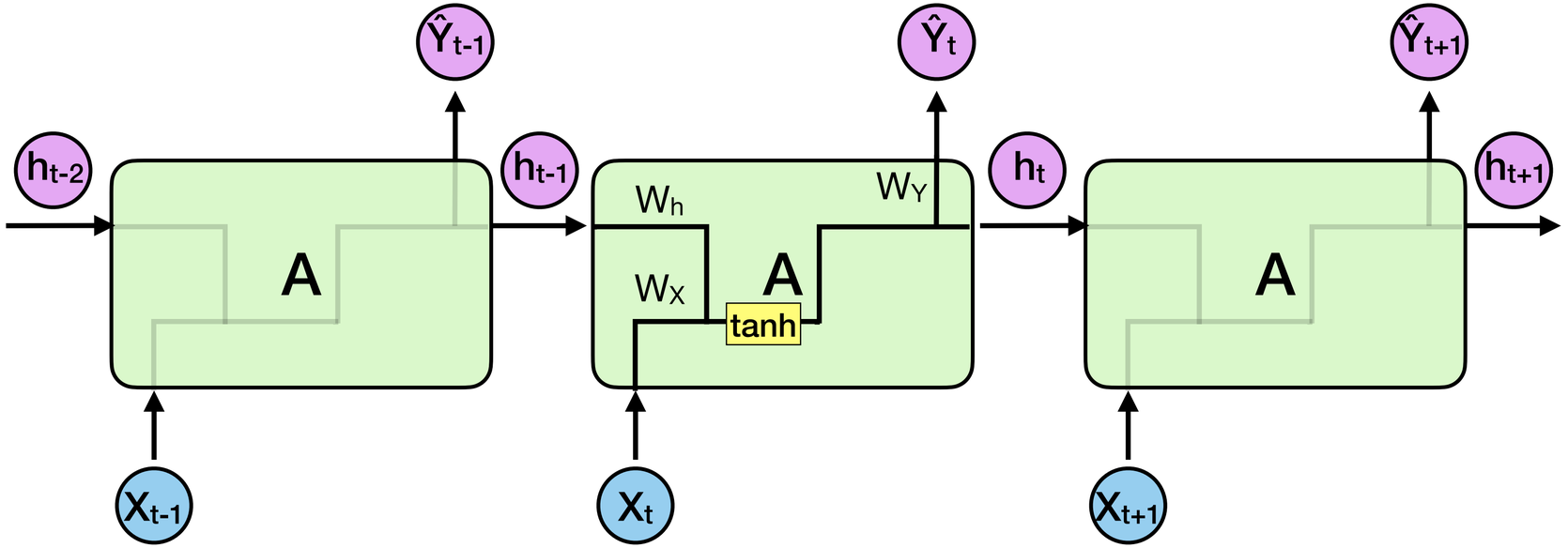


RNN


$\hat{Y}_t$
h_t
A
X_t
=
$\hat{Y}_0$
$\hat{Y}_1$
$\hat{Y}_2$
$\hat{Y}_t$
h_0
h_1
h_2
h_{t-1}
h_t
A
A
A
A
...
X_0
X_1
X_2
X_t

RNN

Realmente es así:



$$h_t = tanh(W_x^T x_t + W_h^T h_{t-1})$$

$$h_t = f_W(x_t, h_{t-1})$$

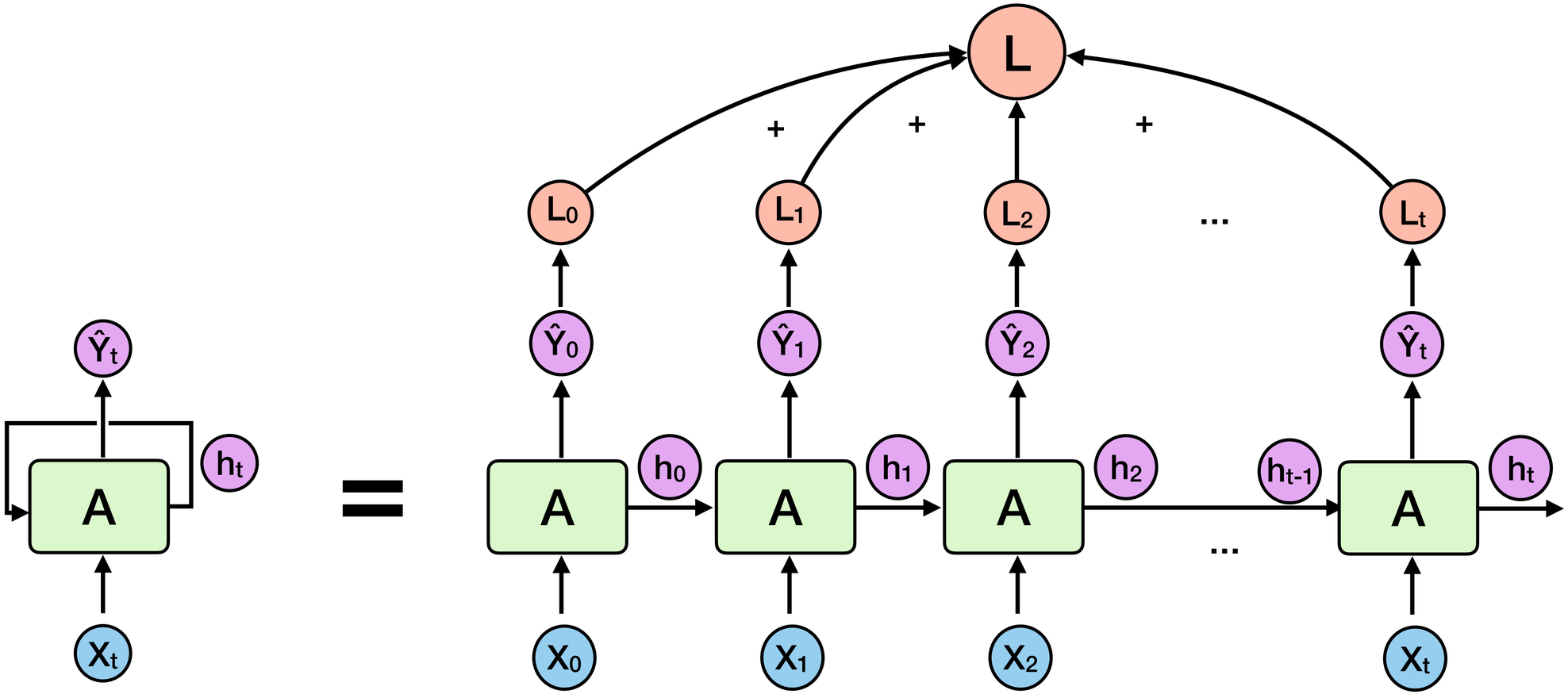
$$\hat{Y}_t = W_Y^T h_t$$

Salida Entrada Memoria

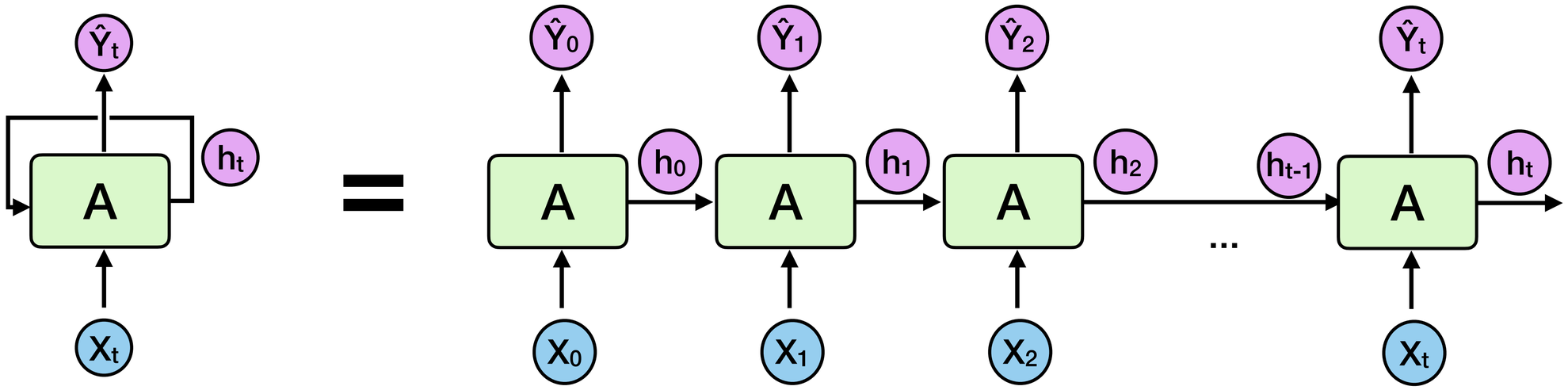
f -> función parametrizada por pesos W

RNN

Función de pérdida



RNN

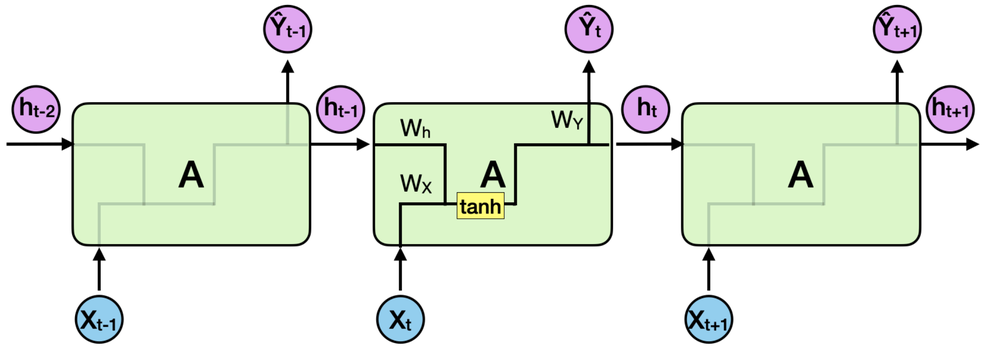



```
tf.keras.layers.SimpleRNN(rnn_units)
```

El número de unidades (rnn_units) es la dimensión del vector del estado oculto h_t

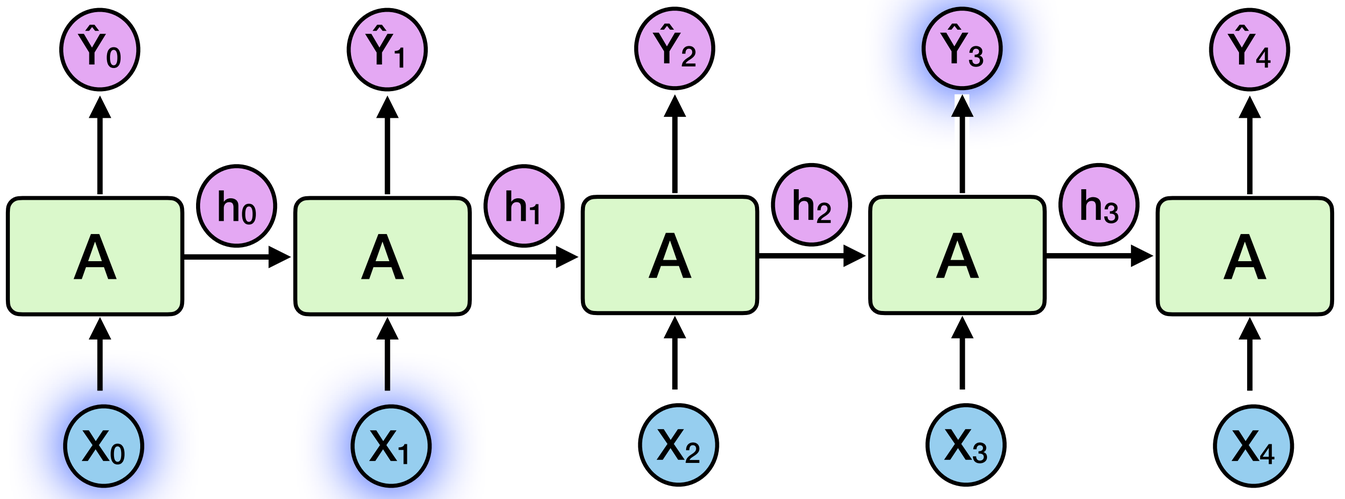
RNN

```
my_rnn = RNN()
hidden_state = [0, 0, 0, 0]

sentence = ["I", "love", "recurrent", "neural"]

for word in sentence:
    prediction, hidden_state = my_rnn(word, hidden_state)

next_word_prediction = prediction
# >>> "networks!"
```



$$h_t = f_W(x_t, h_{t-1})$$

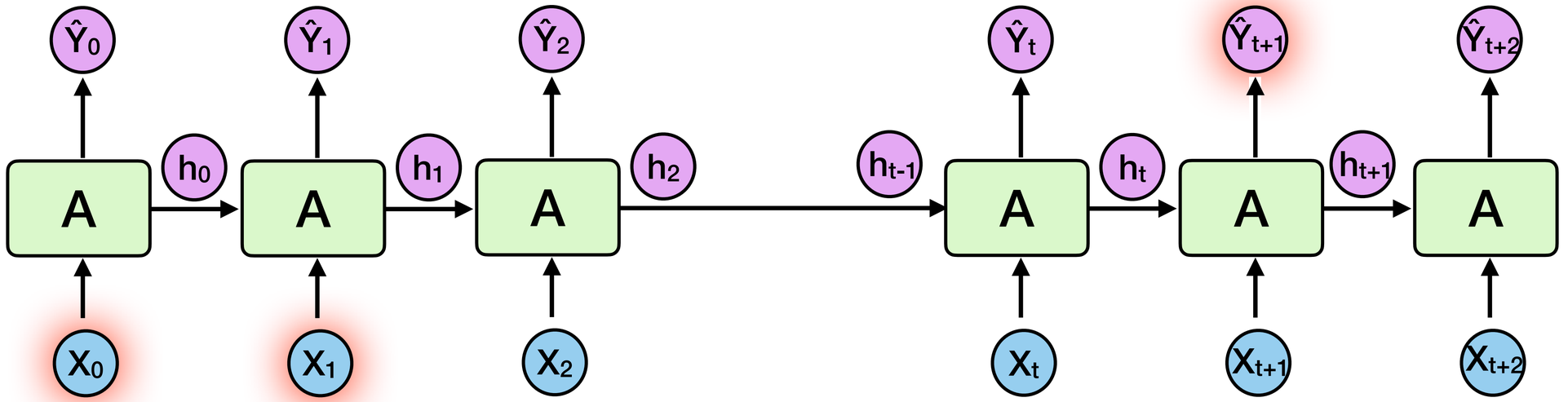
El problema de las RNN



No hay problema cuando la distancia entre una entrada y otra es corto

Ej. Intentar predecir la palabra sky aquí: “the clouds are in the *sky*,”

El problema de las RNN

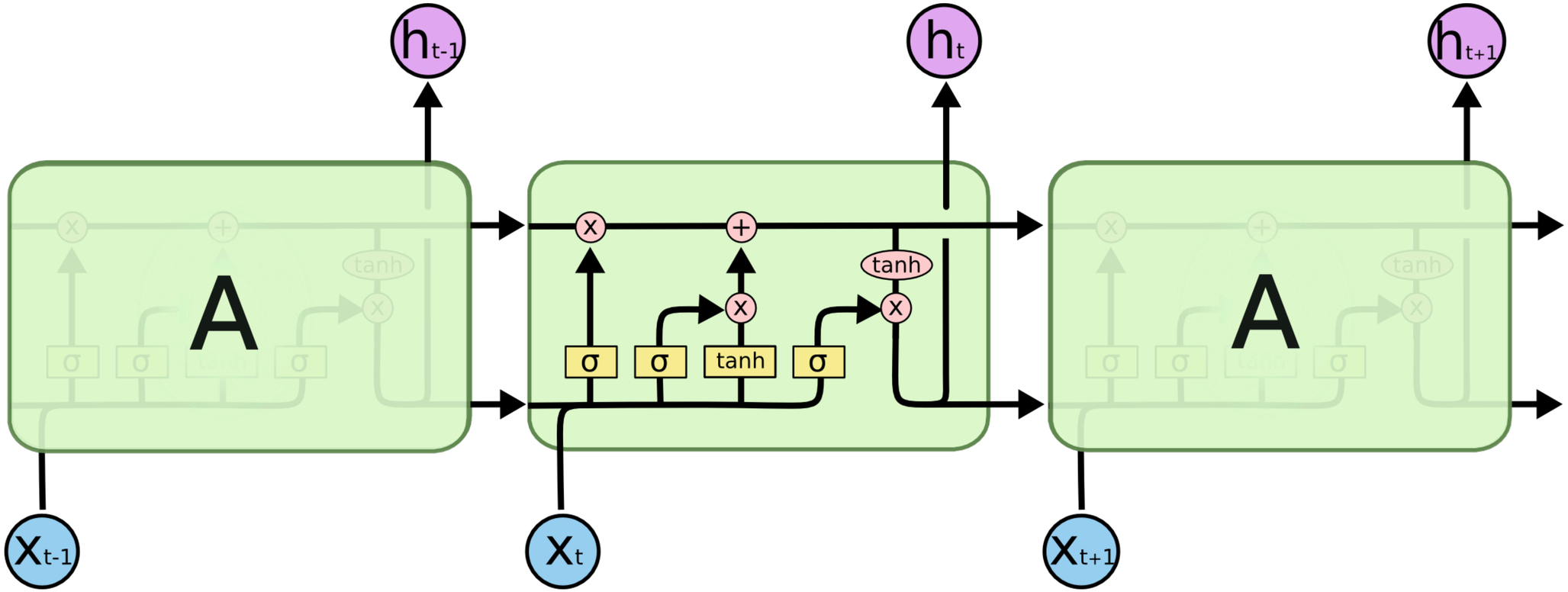


Cuando la distancia es grande, la información se va desvaneciendo, ya que pasa por muchas funciones de activación tanh

Ej. Intentar predecir la palabra French aquí: "I grew up in France many years ago, so I speak fluent *French*."

LSTM

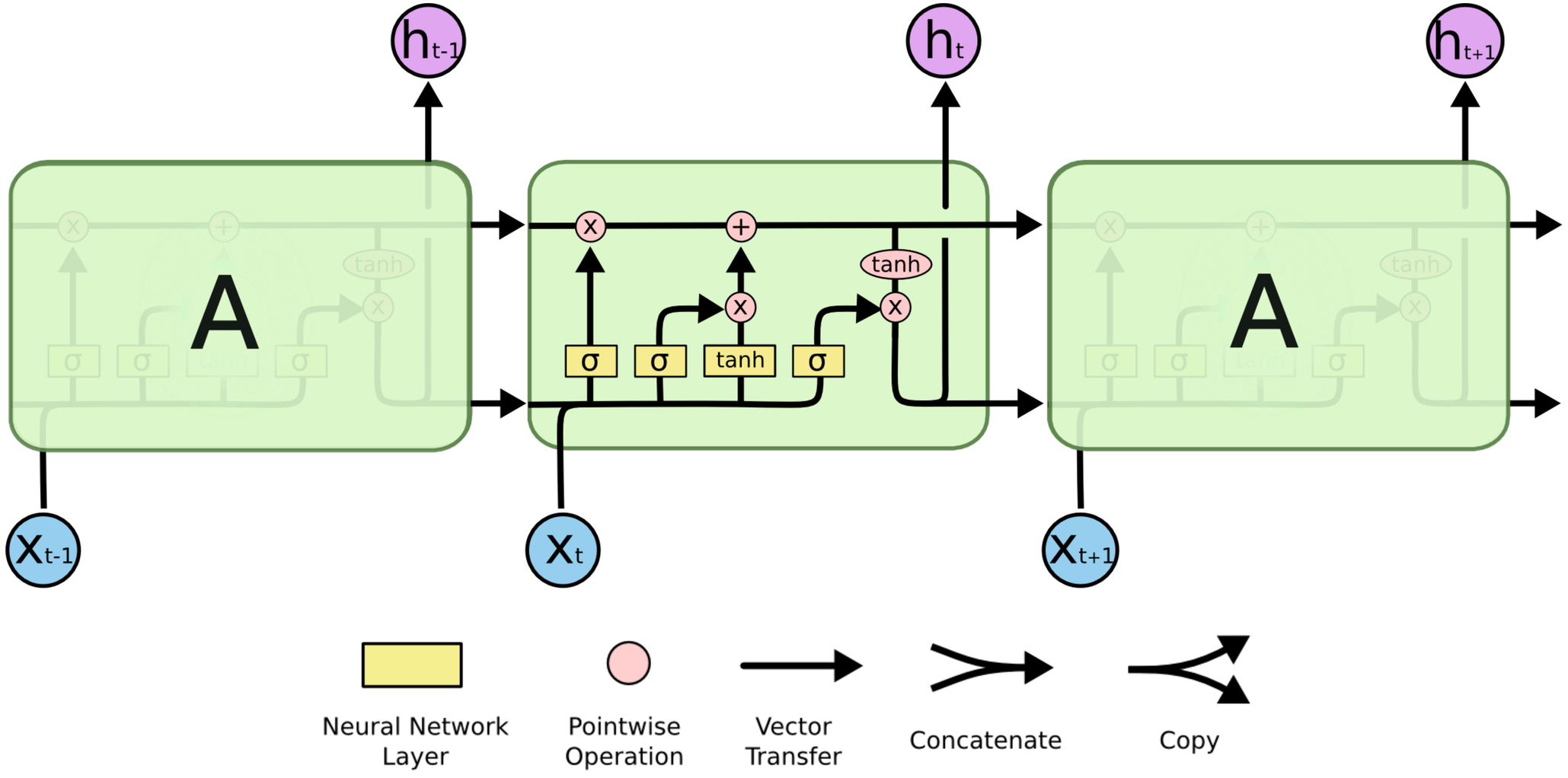
Long Short Term Memory



LSTM

Long Short Term Memory

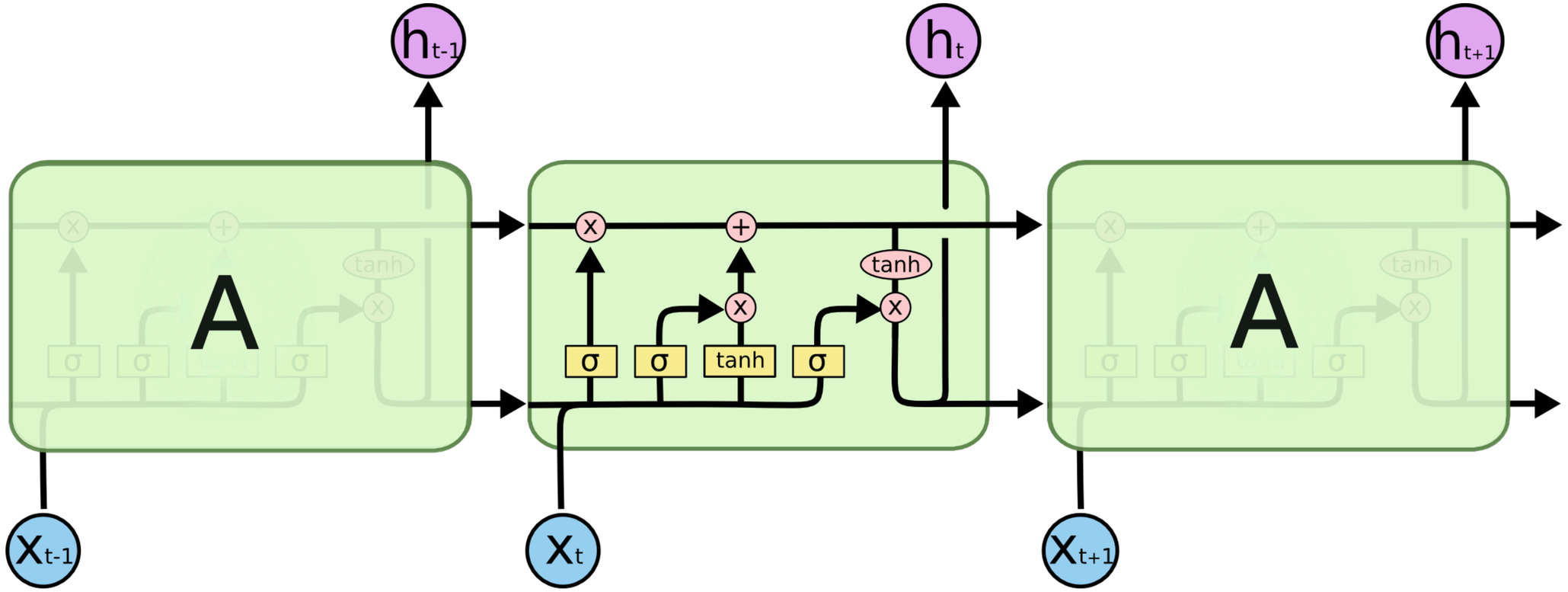
El módulo repetitivo en una LSTM contiene cuatro capas que interactúan.



LSTM

Long Short Term Memory

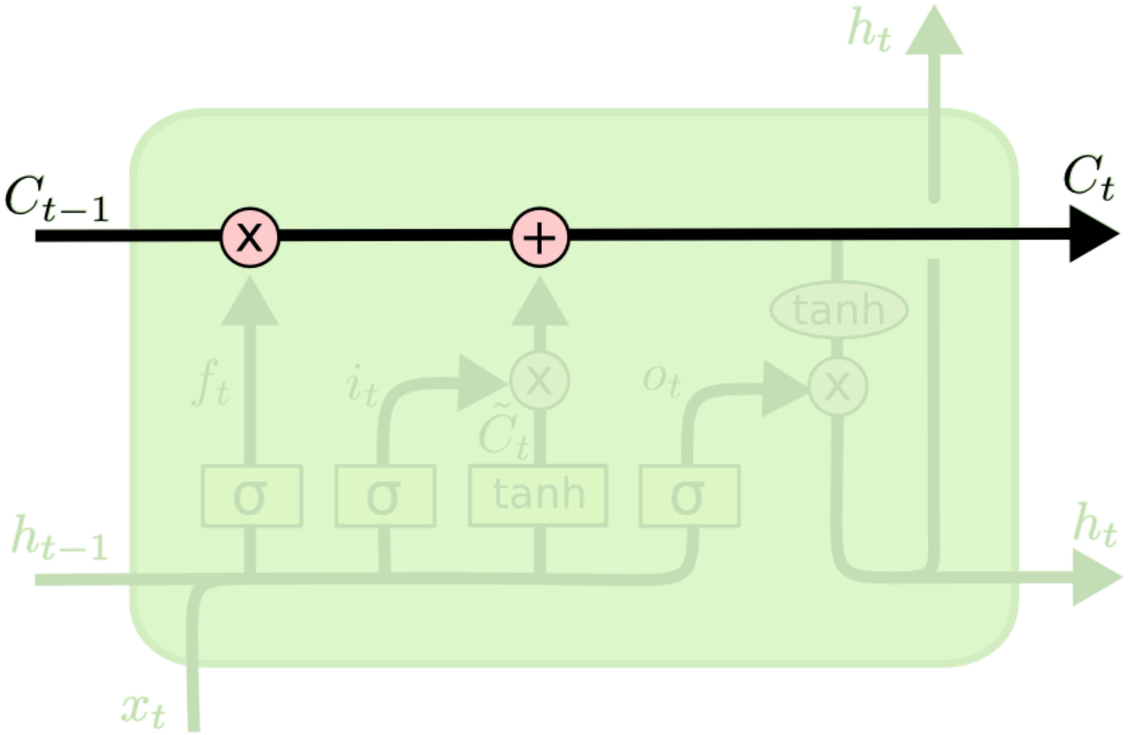
El módulo repetitivo en una LSTM contiene cuatro capas que interactúan.



Pointwise Operation

$$\underset{G}{\begin{bmatrix} 3 & 5 & 7 \\ 4 & 9 & 8 \end{bmatrix}} \circ \underset{H}{\begin{bmatrix} 1 & 6 & 3 \\ 0 & 2 & 9 \end{bmatrix}} = \underset{N}{\begin{bmatrix} 3\times1 & 5\times6 & 7\times3 \\ 4\times0 & 9\times2 & 8\times9 \end{bmatrix}}$$

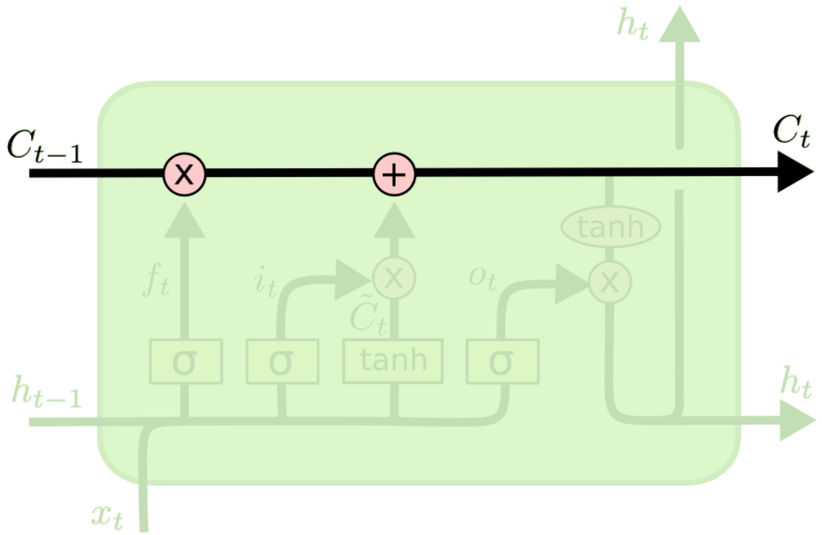
LSTM

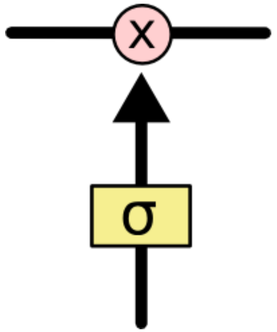


La clave de los LSTM es el **estado de la celda**, la línea horizontal que atraviesa la parte superior del diagrama.

El estado de la celda es como una cinta transportadora. Se ejecuta directamente a lo largo de toda la cadena, con solo algunas interacciones lineales menores. Es muy fácil que **la información fluya** sin cambios.

LSTM





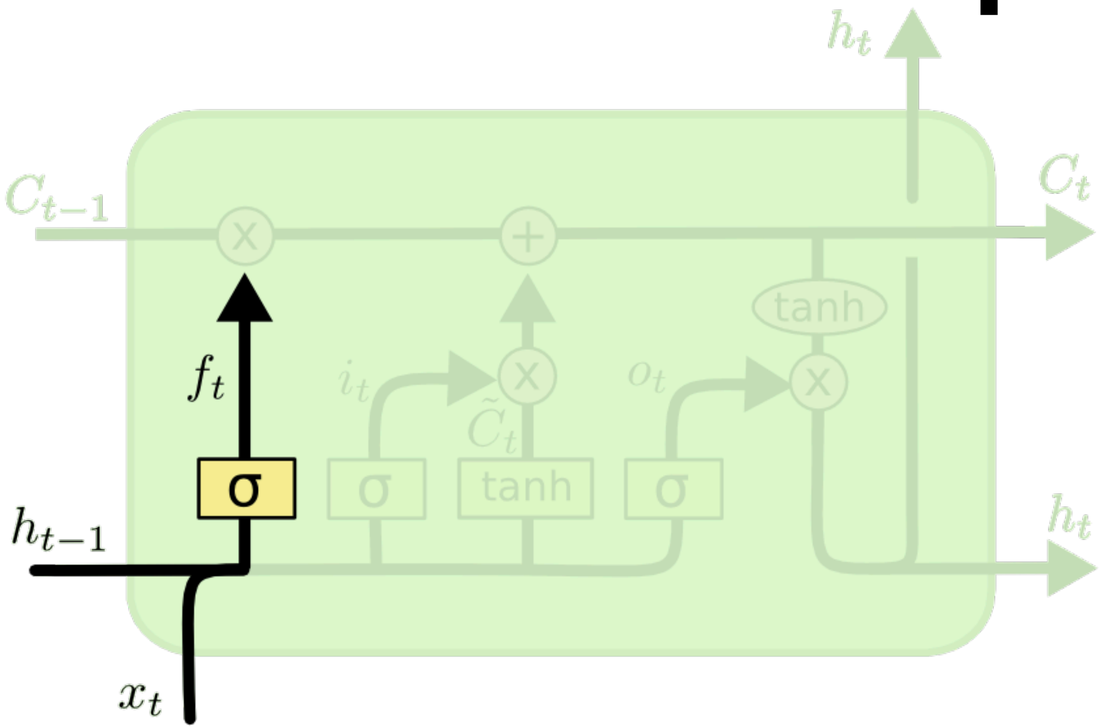
Las puertas son una forma de dejar pasar información opcionalmente. Están compuestos por una capa de red neuronal sigmoidea y una operación de multiplicación escalar.

La capa sigmoidea genera números entre cero y uno, que describen la cantidad de cada componente que se debe dejar pasar. Un valor de cero significa "no dejar pasar nada", mientras que un valor de uno significa "¡dejar pasar todo!"

Un LSTM tiene tres de estas puertas para proteger y controlar el estado de la celda.

1

LSTM paso a paso

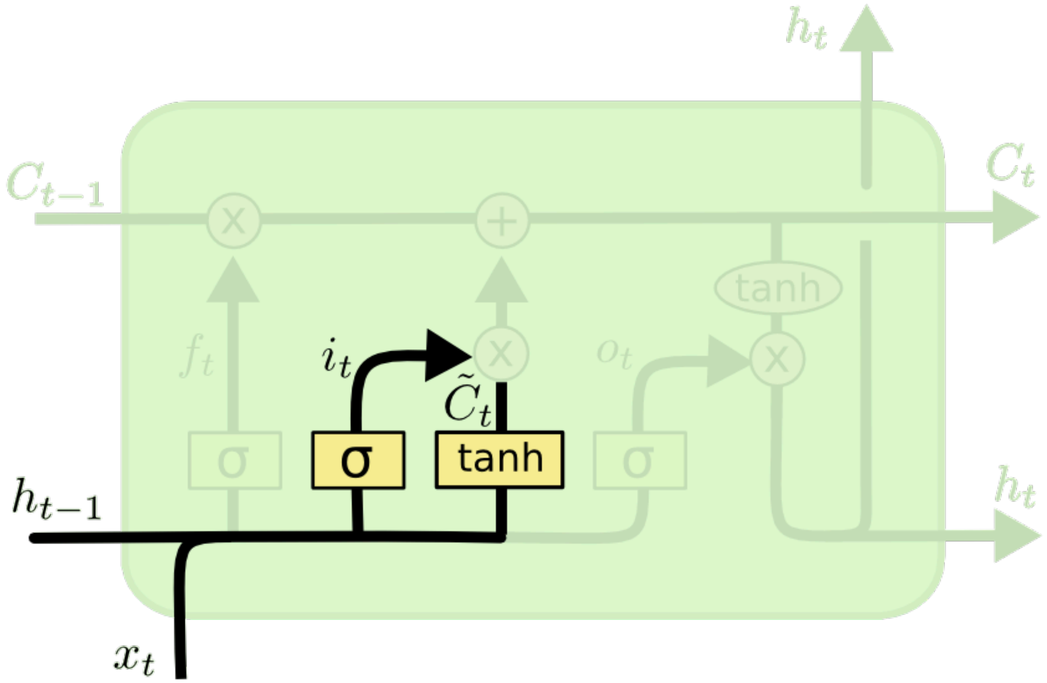


$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

El primer paso en el LSTM es decidir qué información vamos a desechar del estado de la celda. Esta decisión la toma una capa sigmoidea llamada "capa de puerta de olvido". Toma h_{t-1} y x_t, y genera un número entre 0 y 1 para cada número en el estado de celda C_{t-1}. Un 1 representa "mantener esto por completo", mientras que un 0 representa "deshacerse de esto por completo".

2

LSTM paso a paso



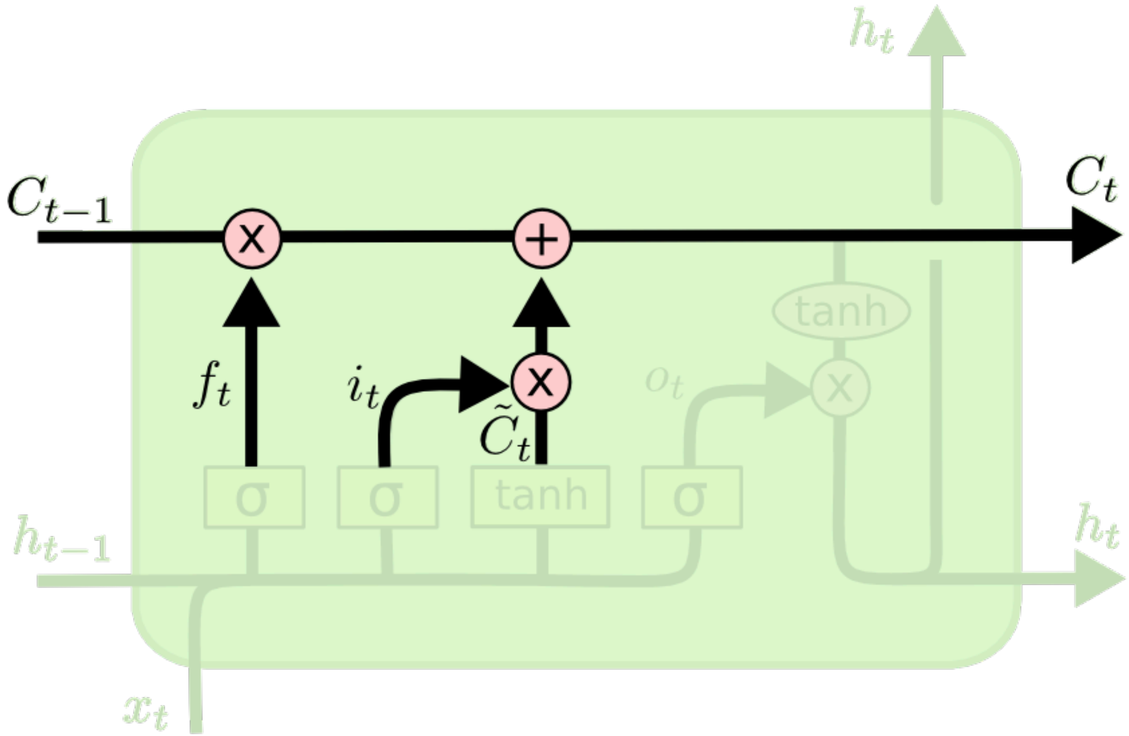
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

El siguiente paso es decidir qué nueva información vamos a almacenar en el estado de la celda. Esto tiene dos partes. Primero, una capa sigmoidea llamada "capa de puerta de entrada" decide qué valores actualizaremos. Luego, una capa tanh crea un vector de nuevos valores candidatos, $\tilde{C}_t$, que podrían agregarse al estado. En el siguiente paso, combinaremos estos dos para crear una actualización del estado.

3

LSTM paso a paso



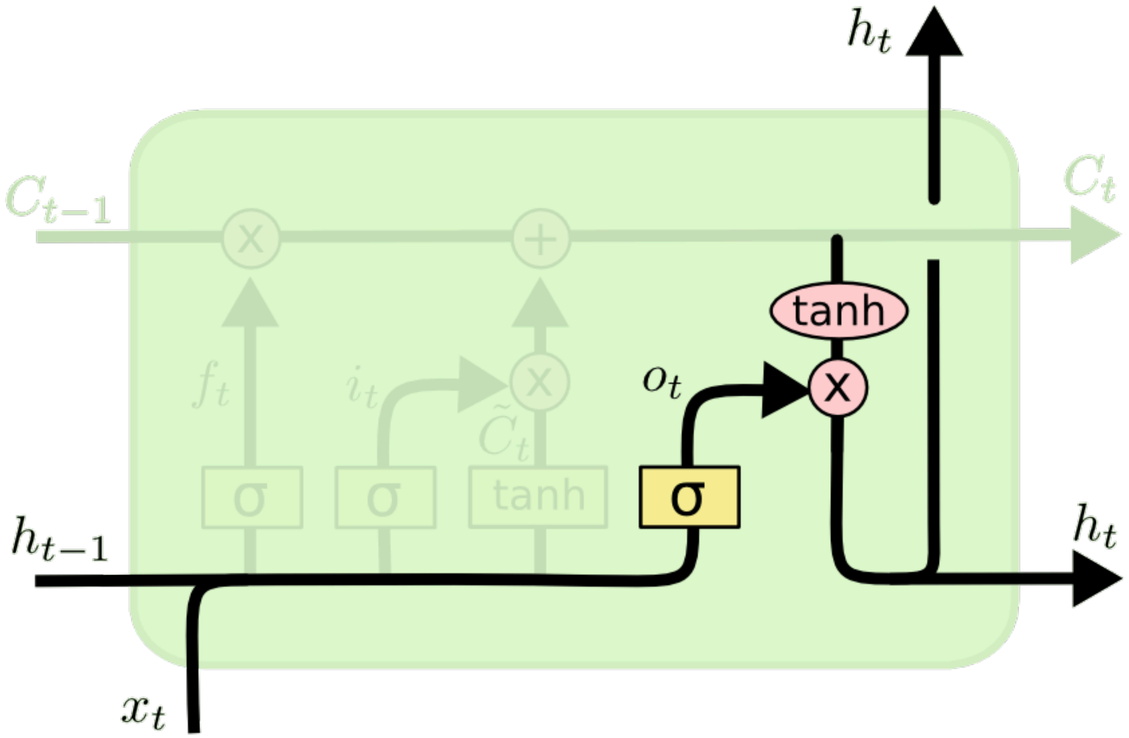
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Ahora es el momento de actualizar el estado de celda anterior, C_{t-1}, al nuevo estado de celda C_t. Los pasos anteriores ya decidieron qué hacer, solo necesitamos hacerlo.

Multiplicamos el estado anterior por f_t, olvidando las cosas que decidimos olvidar antes. Luego lo sumamos $i_t * \tilde{C}_t$. Estos son los nuevos valores candidatos, escalados por cuánto decidimos actualizar cada valor de estado.

4

LSTM paso a paso

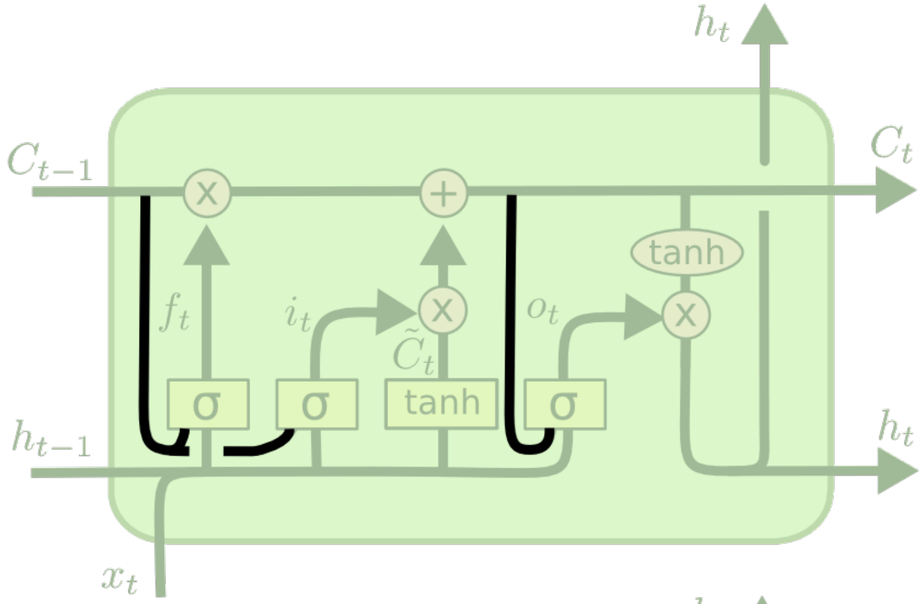


$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$

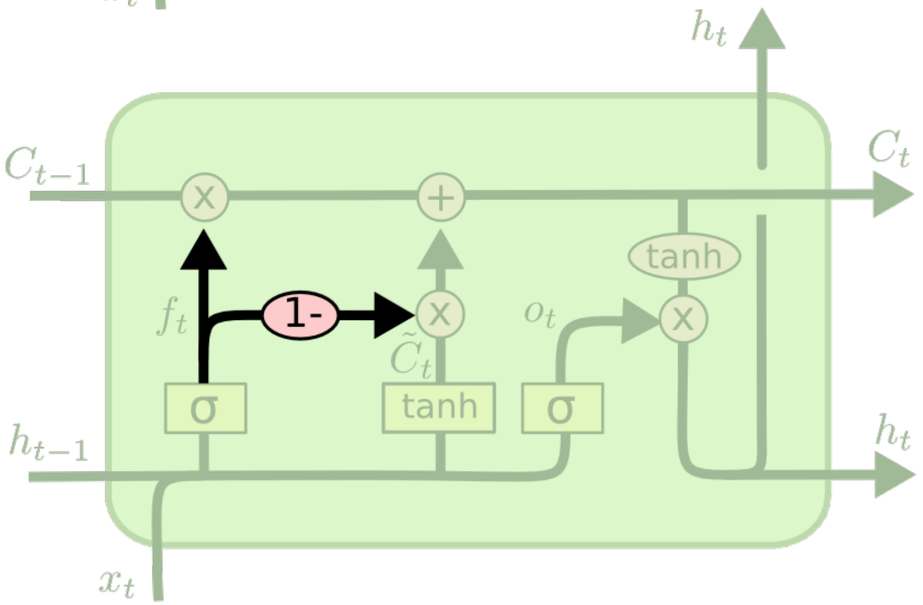
$$h_t = o_t * \tanh\left(C_t\right)$$

Finalmente, tenemos que decidir qué vamos a generar (capa de puerta de salida). Esta salida se basará en nuestro estado de celda, pero será una versión filtrada. Primero, ejecutamos una capa sigmoidea que decide qué partes del estado de la celda vamos a generar. Luego, ponemos el estado de la celda a través de tanh (para que los valores estén entre –1 y 1) y lo multiplicamos por la salida de la puerta sigmoidea, de modo que solo emitamos las partes que decidimos.

Variantes de LSTM



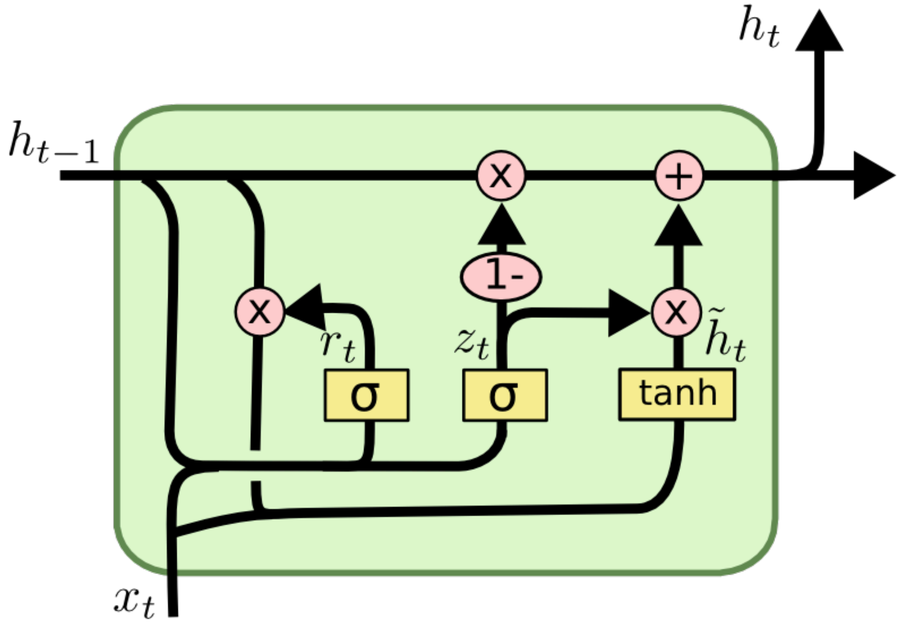
$$f_t = \sigma\left(W_f \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \; + \; b_f\right)$$
$$i_t = \sigma\left(W_i \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \; + \; b_i\right)$$
$$o_t = \sigma\left(W_o \cdot [\boldsymbol{C_t}, h_{t-1}, x_t] \; + \; b_o\right)$$



$$C_t = f_t * C_{t-1} + (\mathbf{1} - \boldsymbol{f_t}) * \tilde{C}_t$$

Variantes de LSTM

GRU (Gated Recurrent Unit)



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Todas obtienen similares resultados