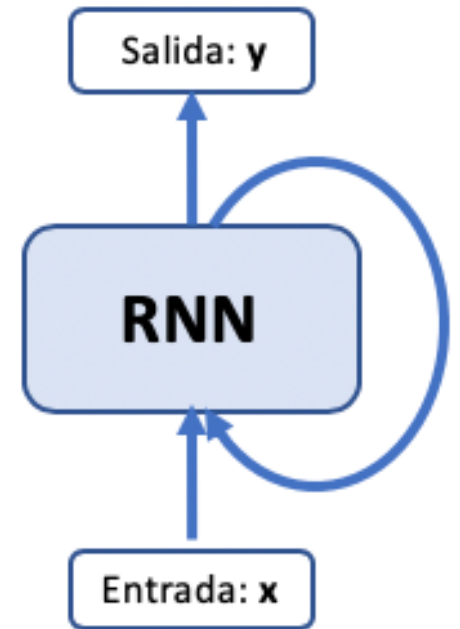


# REDES NEURONALES RECURRENTE



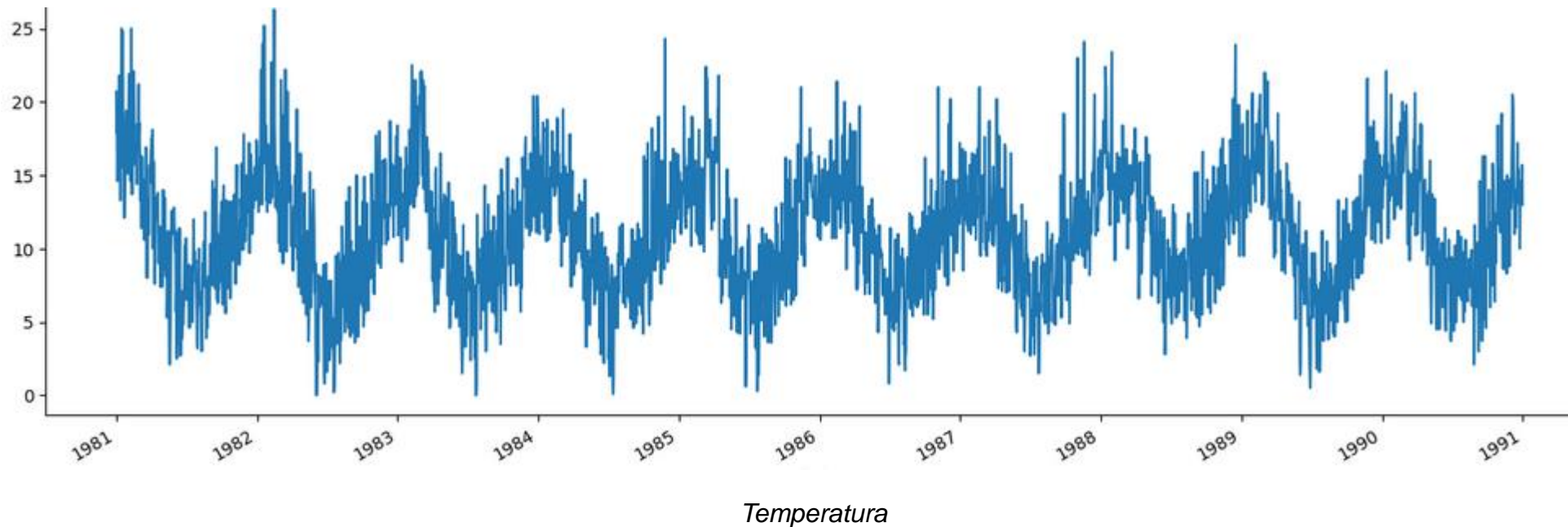
# Red Neuronal Recurrente (RNN)

- Una red neuronal recurrente es un tipo de red diseñada para procesar secuencias de datos, donde la información tiene una estructura temporal o secuencial.
- Las RNN utilizan conexiones recurrentes que les permiten mantener una especie de "memoria" de información previa y utilizarla para procesar entradas futuras en la secuencia.




# Serie Temporal

- Una serie temporal es un conjunto de datos ordenados en el tiempo.
- Los datos están igualmente espaciados en el tiempo, lo que significa que se registraron cada hora, minuto, mes o trimestre.



# Dataset: weather\_dataset.csv

- $p$  (mbar): presión del aire en milibars
- $T$  (degC): temperatura del aire en °C 
- $T_{pot}$  (K): temperatura potencial
- $T_{dew}$  (degC): temperatura de punto de rocío (a la cual el vapor de agua se condensa)
- $rh$  (%): humedad relativa
- $VP_{max}$  (mbar): presión de vapor de agua de saturación
- $VP_{act}$  (mbar): presión de vapor de agua real
- $VP_{def}$  (mbar): déficit de presión de vapor de agua
- $sh$  (g/kg): humedad específica
- $H_2O_C$  (mmol/mol): nivel de concentración del vapor de agua
- $\rho$  (g/m<sup>3</sup>): densidad del aire
- $wv$  (m/s): velocidad del viento
- $max. wv$  (m/s): velocidad del viento máxima
- $wd$  (deg): dirección del viento

*Se construirá un modelo capaz de predecir la temperatura*

# Preprocesamiento de los datos

- Antes de comenzar a trabajar es preciso analizar
  - ▣ Verificar que no haya **valores faltantes**. De ser necesario interpolar.
  - ▣ Verificar la periodicidad de las muestras
    - Eliminar duplicados.
    - Si hay diferencias en la periodicidad, reinterpoliar el data set.
  - ▣ Verificar que la media de cada atributo no ha sufrido grandes modificaciones con todos estos cambios.


*01\_Preproceso\_dataset.ipynb*

# Red Neuronal Recurrente (RNN)

- Según la cantidad de series temporales que se tengan en cuenta como entrada
  - ▣ Univariada
  - ▣ Multivariada
- Según la cantidad de valores que se predigan en cada instante de tiempo
  - ▣ One-step
  - ▣ Multi-step

# Red Neuronal Recurrente (RNN)

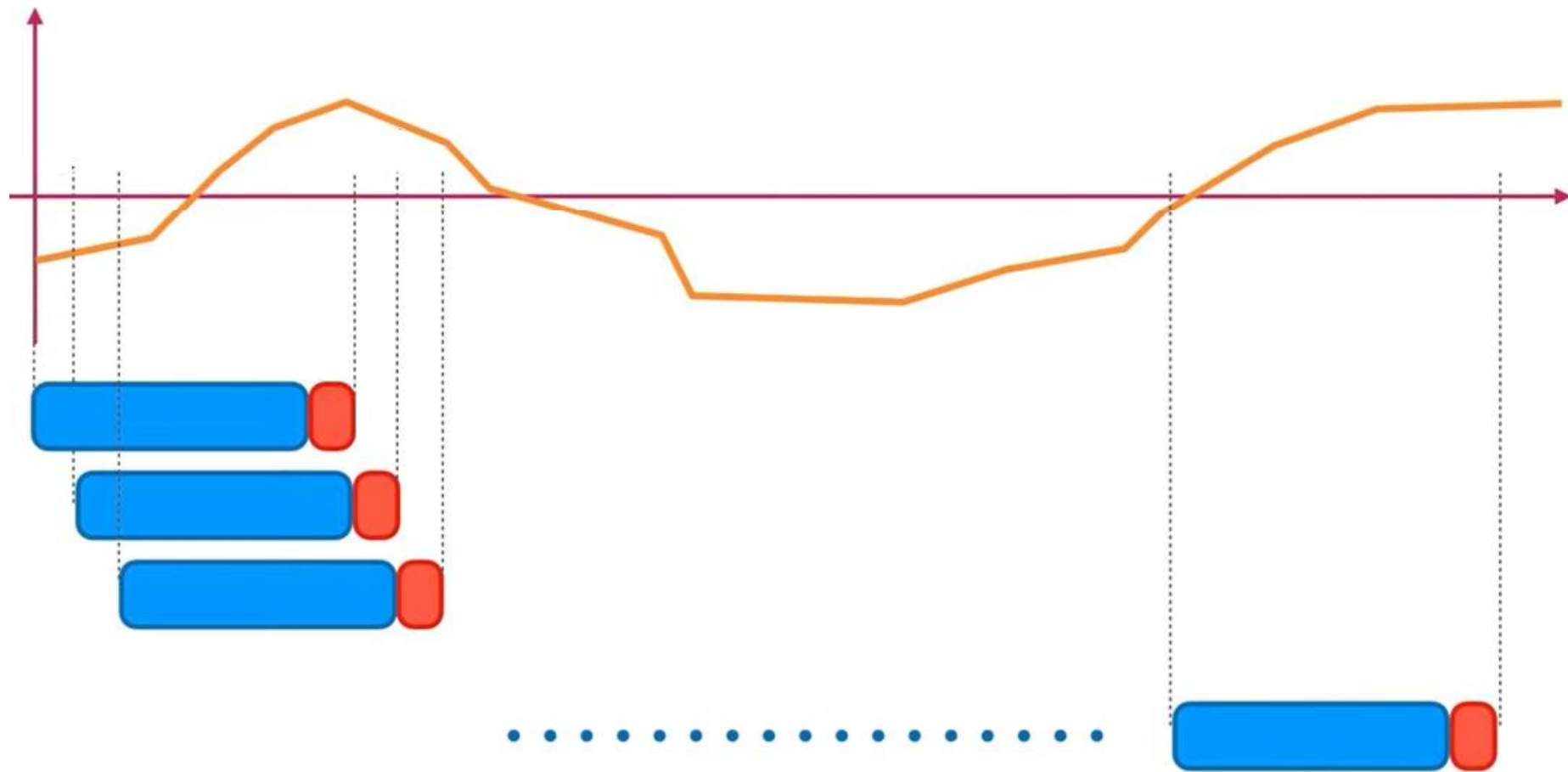
## □ Enfoques

- ▣ Univariado + one-step 
- ▣ Univariado + multi-step
- ▣ Multivariado + one-step
- ▣ Multivariado + multi-step

## □ Tipos de RNN

- ▣ SimpleRNN
- ▣ LSTM

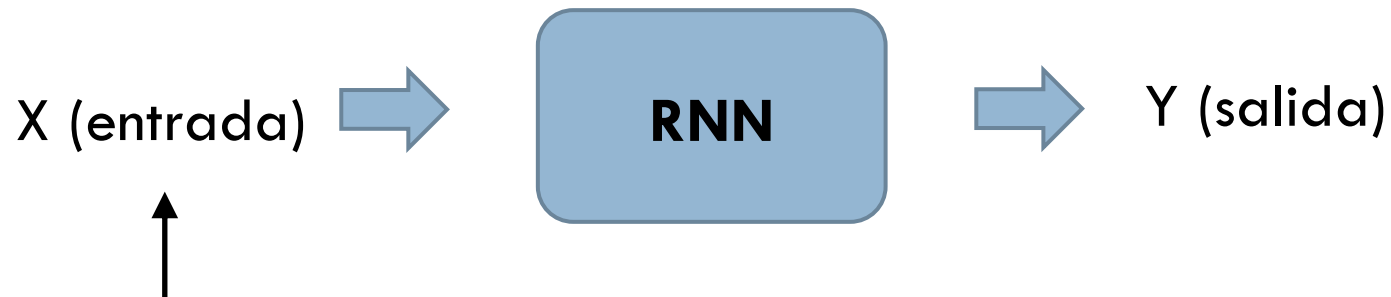
# Preparando los datos de entrada





# Ingresando los datos a la RNN

- Según la documentación de Keras



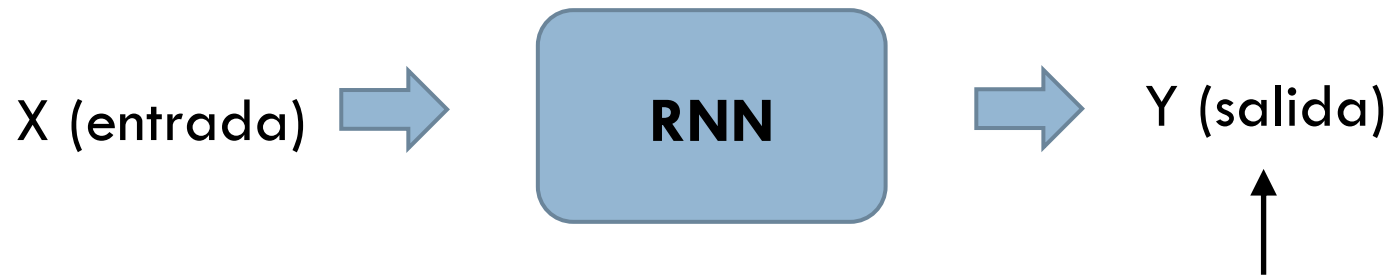
**(Batches, Input\_Length, Features)**

donde

- **Batches** : cant.de datos (ventanas) de entrenamiento
- **Input\_Length**: Longitud de la ventana de entrada
- **Features**: cantidad de series o variables a utilizar

# Ingresando los datos a la RNN

- Según la documentación de Keras



**(Batches, Output\_Length, Features)**

donde

- **Batches** : cant.de predicciones (ídem a los de entrada)
- **Output\_Length**: Longitud de la ventana de salida
- **Features**: cantidad de series o variables a predecir

# Preparación de los datos

```
x, y = crear_dataset_supervisado(datos, 5, 1) ←
```

*Genera secuencias de entrada de longitud 5 y de salida de longitud 1*

```
Datos = [-8.05 -8.88 -8.81 -9.05 -9.63 -9.67 -9.17 -8.1 ... ]
```

```
x = [[[-8.05]
      [-8.88]
      [-8.81]
      [-9.05]
      [-9.63]]
      [[-8.88]
      [-8.81]
      [-9.05]
      [-9.63]
      [-9.67]]]
```

```
y = [[[-9.67]]
      [[-9.17]]]
```

02a\_UniVar\_OneStep\_SimpleRNN.ipynb

# Preparación de los datos

- Si se ingresan secuencias de longitud 5 y se predice el valor siguiente

```
secuencia = [-8.05 -8.88 -8.81 -9.05 -9.63 -9.67 -9.17 -8.1 ... ]
```

```
entrada = [[[-8.05]
             [-8.88]
             [-8.81]
             [-9.05]
             [-9.63]]
            [[-8.88]
             [-8.81]
             [-9.05]
             [-9.63]
             [-9.67]]
            ...]
```

(Batches, Input\_Length, Features)  
# sec            5            1

```
salida = [[[-9.67]]
           [[-9.17]]
           ... ]
```

(Batches, Output\_Length, Features)  
# sec            1            1

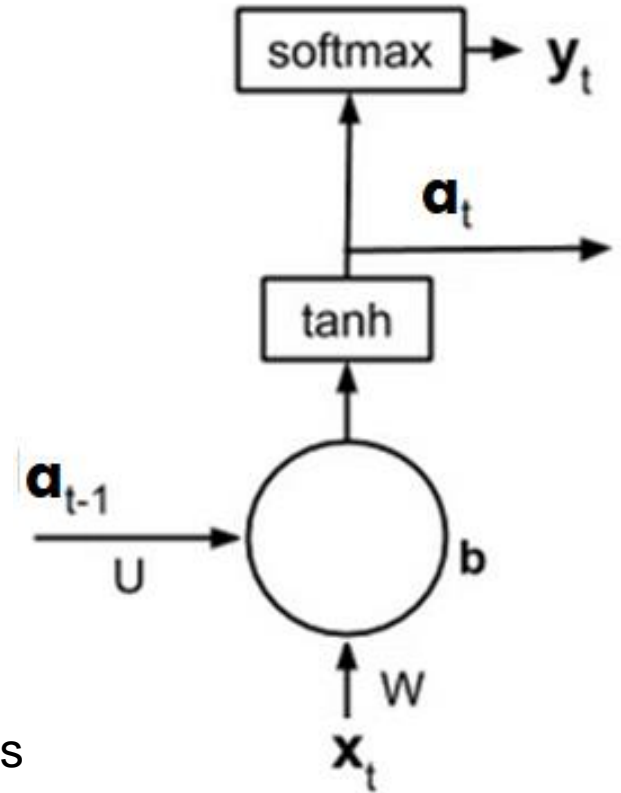
# SimpleRNN

$$h_t = \tanh(Wx_t + Ua_{t-1} + b)$$

$$y_t = \text{softmax}(a_t)$$

$$x_t \in \mathbb{R}^d; W \in \mathbb{R}^{h \times d}; U \in \mathbb{R}^{h \times h}; b \in \mathbb{R}^h$$

La dimensión del vector  $a_t$  coincide con la cantidad  $h$  de neuronas ocultas



```

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Input, SimpleRNN, Dense

modelo = Sequential(name='modelo')
modelo.add(SimpleRNN(4, input_shape=(24,1),name='Oculto'))
modelo.add(Dense(1, activation='linear', name='Salida'))

modelo.summary()

```

Model: "modelo"

| Layer (type)                   | Output Shape | Param # |
|--------------------------------|--------------|---------|
| =====                          |              |         |
| Oculto (SimpleRNN)             | (None, 4)    | 24      |
| Salida (Dense)                 | (None, 1)    | 5       |
| =====                          |              |         |
| Total params: 29 (116.00 Byte) |              |         |

$$x_t \in \mathbb{R}^d;$$

$$W \in \mathbb{R}^{h \times d}; b \in \mathbb{R}^h$$

$$U \in \mathbb{R}^{h \times h}$$

*1 de c/u por capa*

```

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Input, SimpleRNN, Dense

modelo2 = Sequential(name='modelo2')
modelo2.add(Input(shape=(24,1), name='Entrada'))
modelo2.add(SimpleRNN(10, return_sequences=True, name='Ocultaa1'))
modelo2.add(SimpleRNN(10, name='Ocultaa2'))
modelo2.add(Dense(1, activation='linear', name='Salida'))

modelo2.summary()

```

Model: "modelo2"

| Layer (type)         | Output Shape   | Param # |
|----------------------|----------------|---------|
| Ocultaa1 (SimpleRNN) | (None, 24, 10) | 120     |
| Ocultaa2 (SimpleRNN) | (None, 10)     | 210     |
| Salida (Dense)       | (None, 1)      | 11      |

Total params: 341 (1.33 KB)

$$x_t \in \mathbb{R}^d;$$

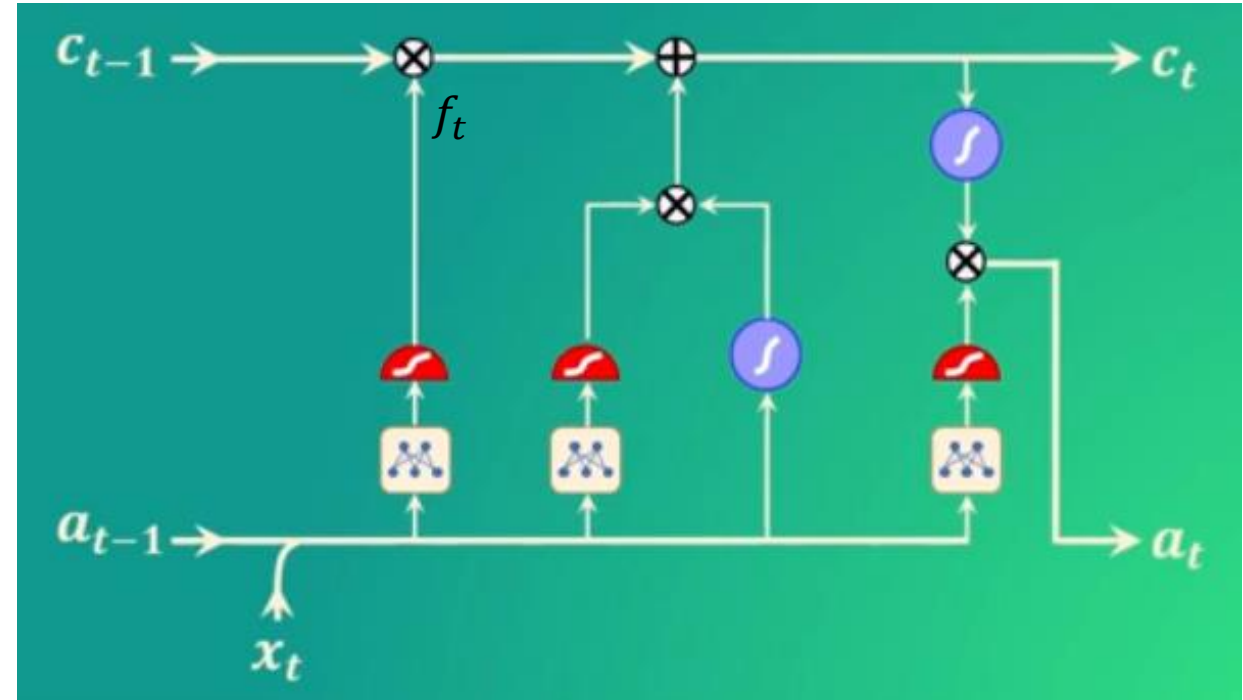
$$W \in \mathbb{R}^{h \times d}; b \in \mathbb{R}^h$$

$$U \in \mathbb{R}^{h \times h}$$

1 de c/u por capa

# LSTM

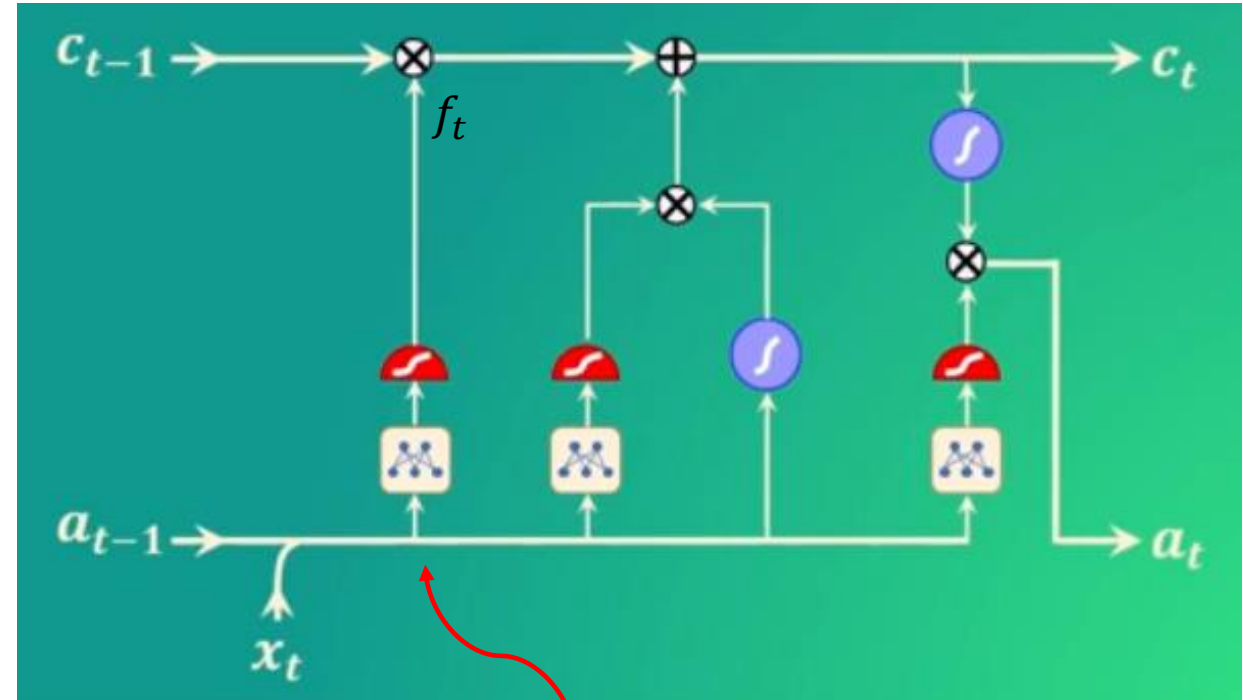
- La red LSTM agrega una celda de estado o MEMORIA para resolver el problema de olvido de la red recurrente simple





# LSTM

$$f_t = \text{sig}(W_f x_t + U_f a_{t-1} + b_f)$$

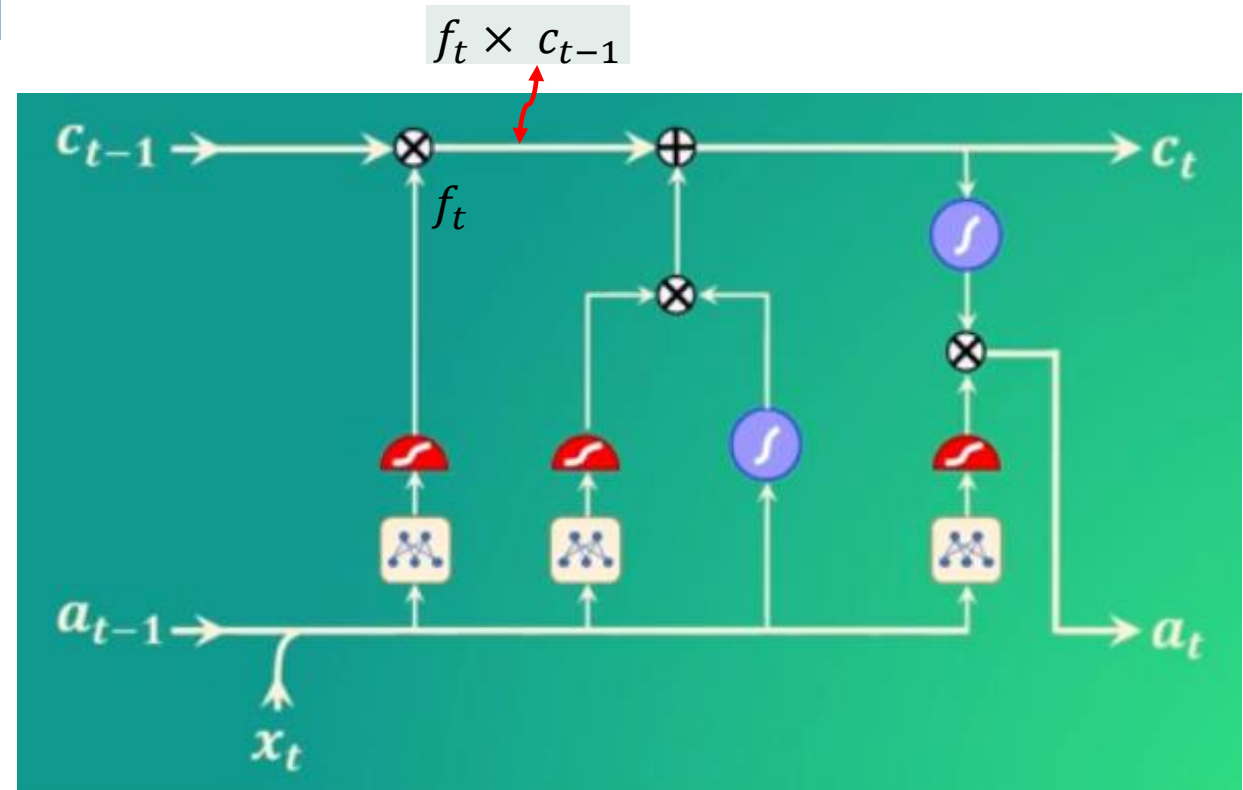


## **Forget Gate**

Retorna un vector con valores entre 0 y 1.  
Con valores cercanos a 0 “olvida” y con los  
cercanos a 1 “recuerda”

# LSTM

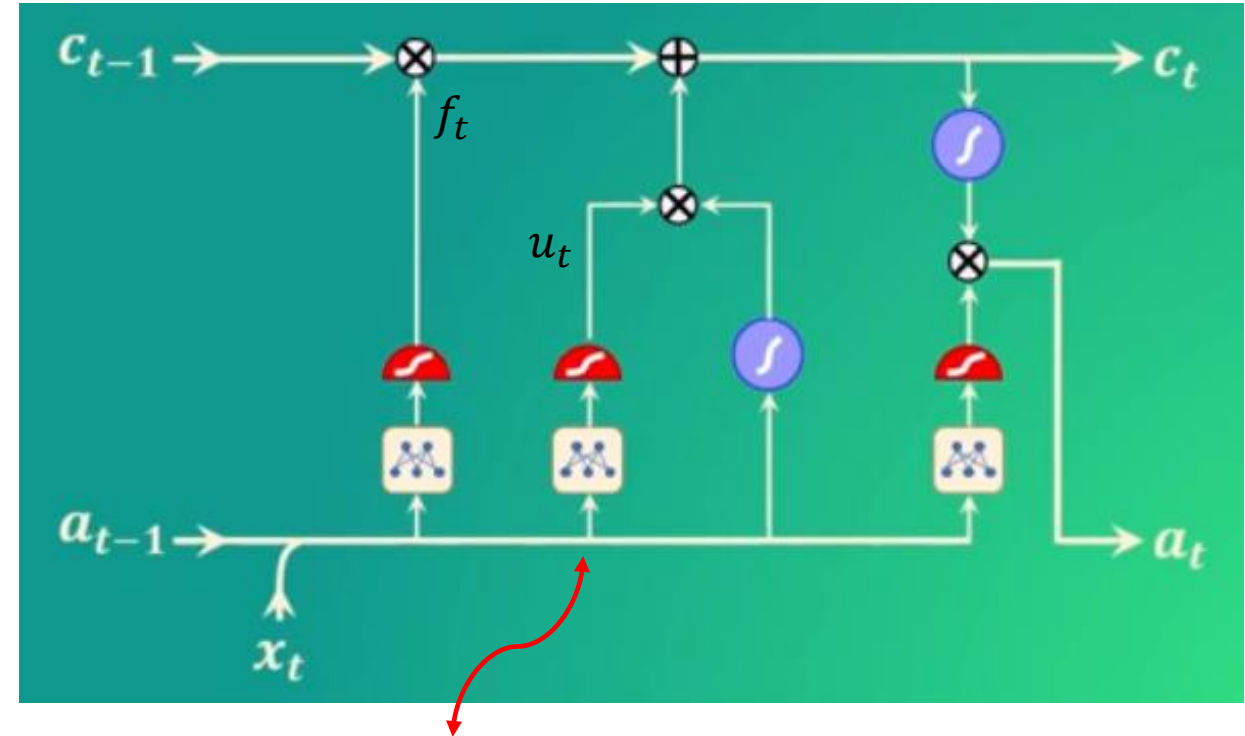
$$f_t = \text{sig}(W_f x_t + U_f a_{t-1} + b_f)$$



# LSTM

$$f_t = \text{sig}(W_f x_t + U_f a_{t-1} + b_f)$$

$$u_t = \text{sig}(W_i x_t + U_i a_{t-1} + b_i)$$



## Update Gate

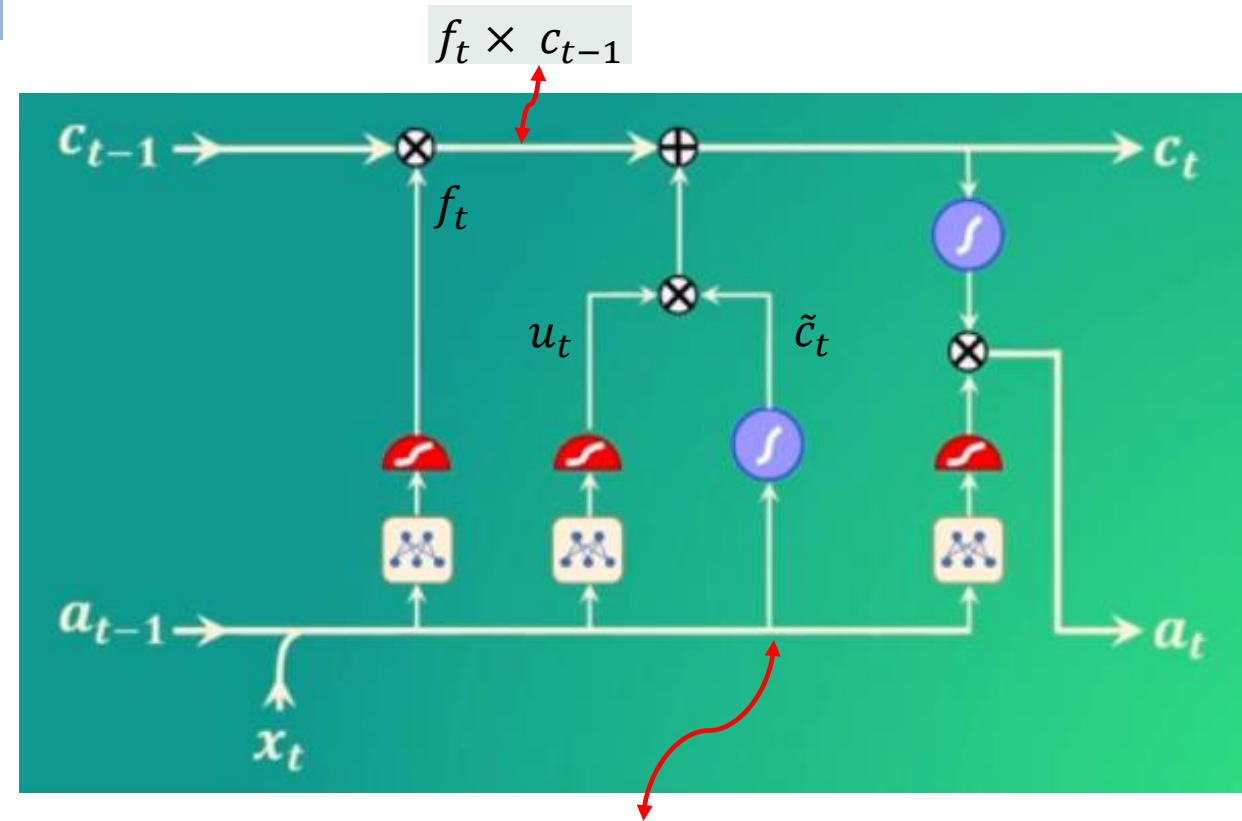
Retorna un vector con valores entre 0 y 1.  
Con valores cercanos a 1 serán preservados

# LSTM

$$f_t = \text{sig}(W_f x_t + U_f a_{t-1} + b_f)$$

$$u_t = \text{sig}(W_i x_t + U_i a_{t-1} + b_i)$$

$$\tilde{c}_t = \tanh(W_c x_t + U_c a_{t-1} + b_c)$$



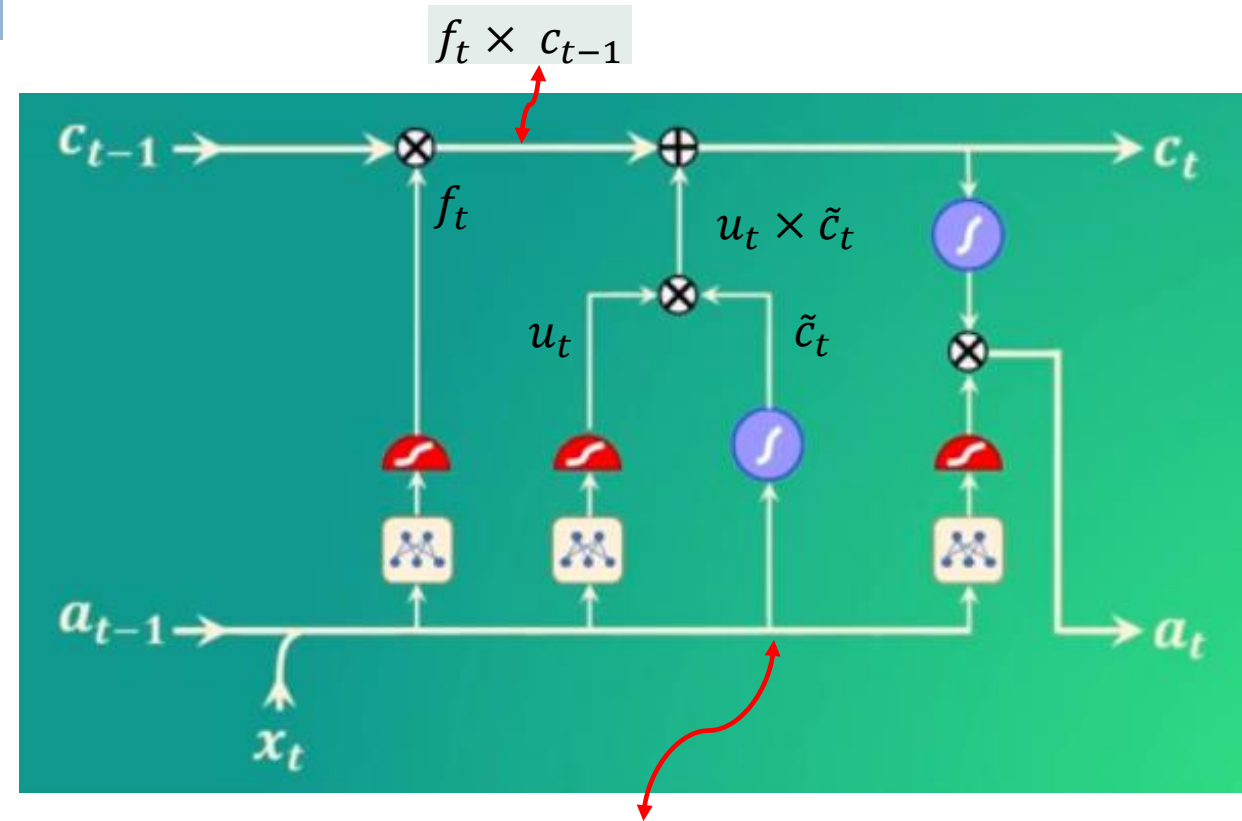
$\tilde{c}_t$  es un vector de valores candidatos a formar parte de la nueva memoria

# LSTM

$$f_t = \text{sig}(W_f x_t + U_f a_{t-1} + b_f)$$

$$u_t = \text{sig}(W_i x_t + U_i a_{t-1} + b_i)$$

$$\tilde{c}_t = \tanh(W_c x_t + U_c a_{t-1} + b_c)$$



$\tilde{c}_t$  es un vector de valores candidatos a formar parte de la nueva memoria.

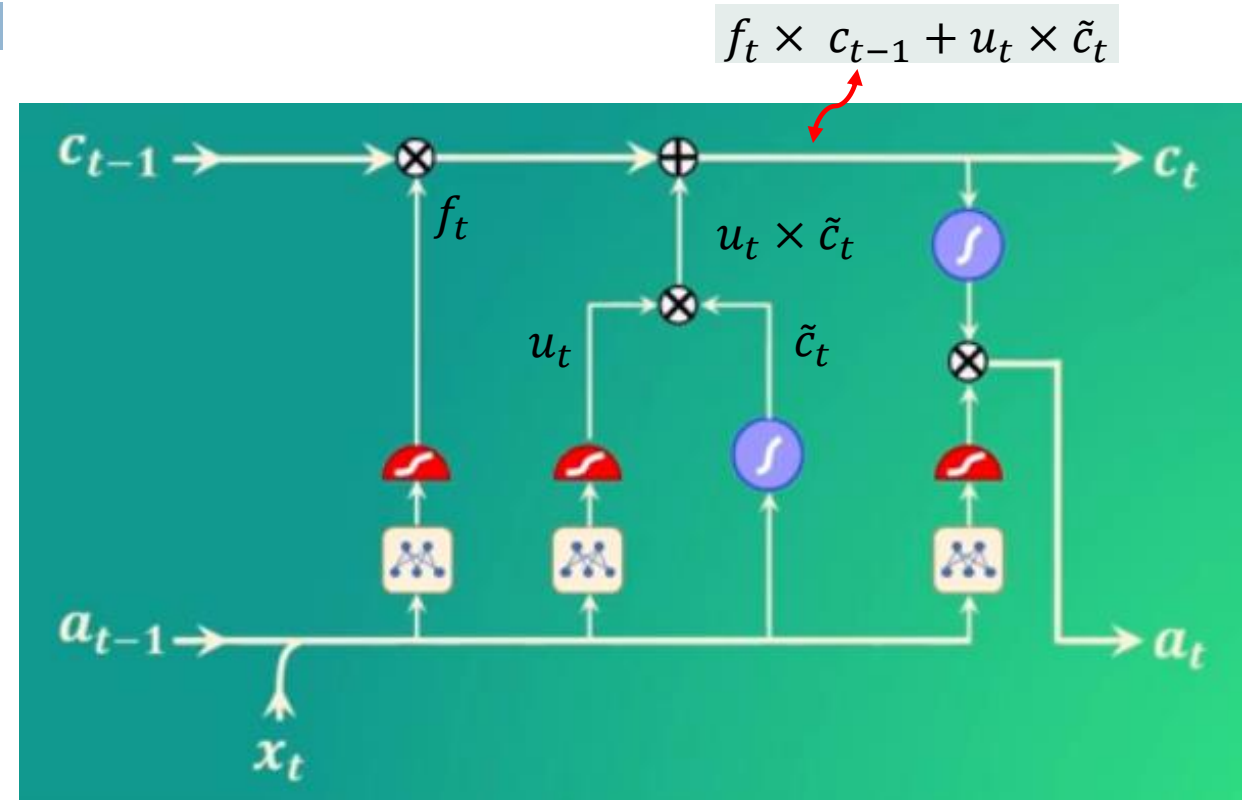
Estos valores se filtran utilizando  $u_t$

# LSTM

$$f_t = \text{sig}(W_f x_t + U_f a_{t-1} + b_f)$$

$$u_t = \text{sig}(W_i x_t + U_i a_{t-1} + b_i)$$

$$\tilde{c}_t = \tanh(W_c x_t + U_c a_{t-1} + b_c)$$



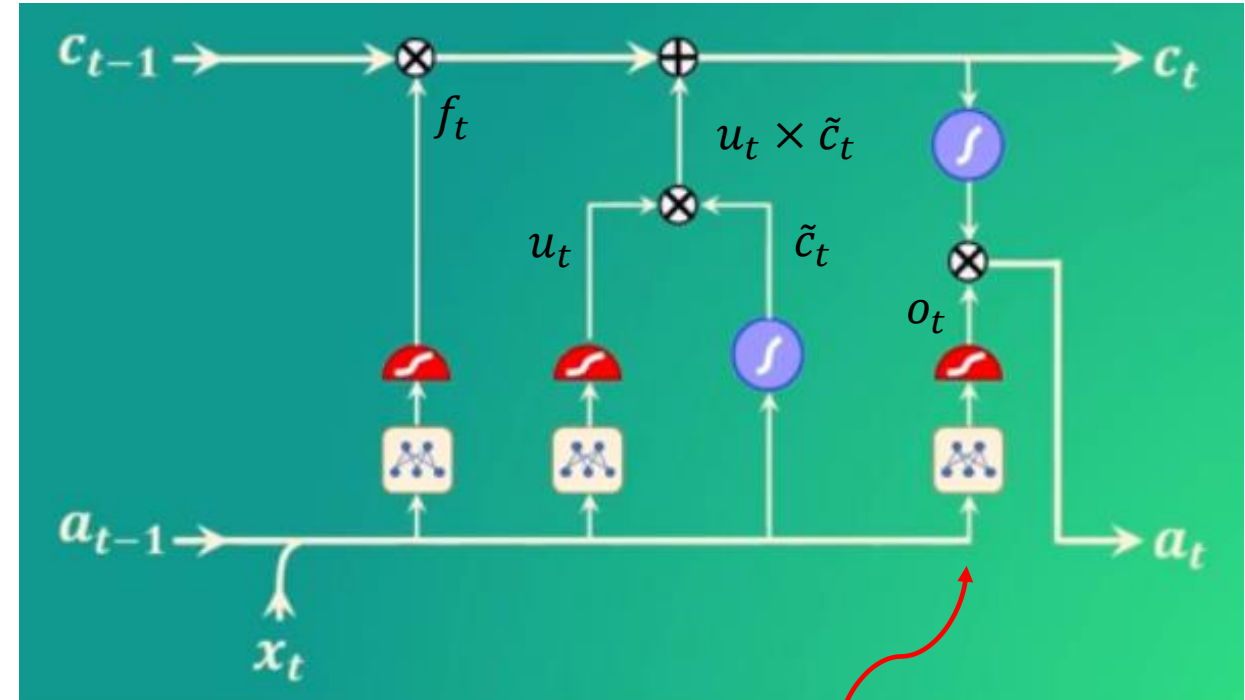
# LSTM

$$f_t = \text{sig}(W_f x_t + U_f a_{t-1} + b_f)$$

$$u_t = \text{sig}(W_i x_t + U_i a_{t-1} + b_i)$$

$$\tilde{c}_t = \tanh(W_c x_t + U_c a_{t-1} + b_c)$$

$$o_t = \text{sig}(W_o x_t + U_o a_{t-1} + b_o)$$



Determina qué valores de la memoria formarán parte del nuevo estado oculto  $a_t$

# LSTM

02b\_UniVar\_OneStep\_LSTM.ipynb

$$f_t = \text{sig}(W_f x_t + U_f a_{t-1} + b_f)$$

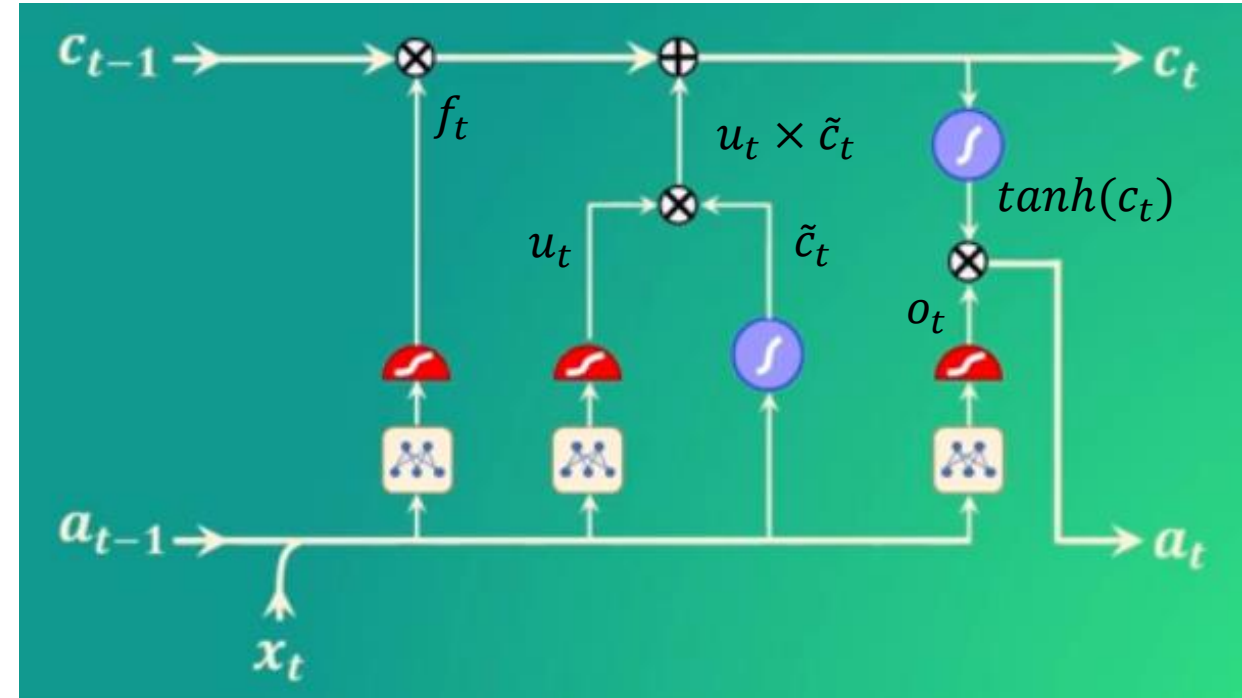
$$u_t = \text{sig}(W_i x_t + U_i a_{t-1} + b_i)$$

$$\tilde{c}_t = \tanh(W_c x_t + U_c a_{t-1} + b_c)$$

$$o_t = \text{sig}(W_o x_t + U_o a_{t-1} + b_o)$$

$$c_t = f_t \times c_{t-1} + u_t \times \tilde{c}_t$$

$$a_t = o_t \times \tanh(c_t)$$



La dimensión de los vectores  $f_t$ ,  $u_t$ ,  $\tilde{c}_t$  y  $o_t$  coincide con la cantidad  $h$  de neuronas ocultas

$$x_t \in \mathbb{R}^d; W \in \mathbb{R}^{h \times d}; U \in \mathbb{R}^{h \times h}; b \in \mathbb{R}^h$$



```

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense

INPUT_SHAPE = (24,1)
OUTPUT_LENGTH = 1
N_UNITS = 1

modelo = Sequential()
modelo.add(LSTM(N_UNITS, input_shape=INPUT_SHAPE))
modelo.add(Dense(OUTPUT_LENGTH, activation='linear'))
modelo.summary()

```

$$x_t \in \mathbb{R}^d;$$

$$W \in \mathbb{R}^{h \times d}; b \in \mathbb{R}^h$$

$$U \in \mathbb{R}^{h \times h}$$

*4 de c/u por capa*

Model: "sequential"

| Layer (type)                  | Output Shape | Param # |
|-------------------------------|--------------|---------|
| =====                         |              |         |
| lstm (LSTM)                   | (None, 1)    | 12      |
| dense (Dense)                 | (None, 1)    | 2       |
| =====                         |              |         |
| Total params: 14 (56.00 Byte) |              |         |

```

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense

INPUT_SHAPE = (24,1)
OUTPUT_LENGTH = 1
N_UNITS = 3

modelo = Sequential()
modelo.add(LSTM(N_UNITS, input_shape=INPUT_SHAPE))
modelo.add(Dense(OUTPUT_LENGTH, activation='linear'))
modelo.summary()

```

$$x_t \in \mathbb{R}^d;$$

$$W \in \mathbb{R}^{h \times d}; b \in \mathbb{R}^h$$

$$U \in \mathbb{R}^{h \times h}$$

*4 de c/u por capa*

Model: "sequential"

| Layer (type)                   | Output Shape | Param # |
|--------------------------------|--------------|---------|
| =====                          |              |         |
| lstm (LSTM)                    | (None, 3)    | 60      |
| dense (Dense)                  | (None, 1)    | 4       |
| =====                          |              |         |
| Total params: 64 (256.00 Byte) |              |         |