Sesión 1.2

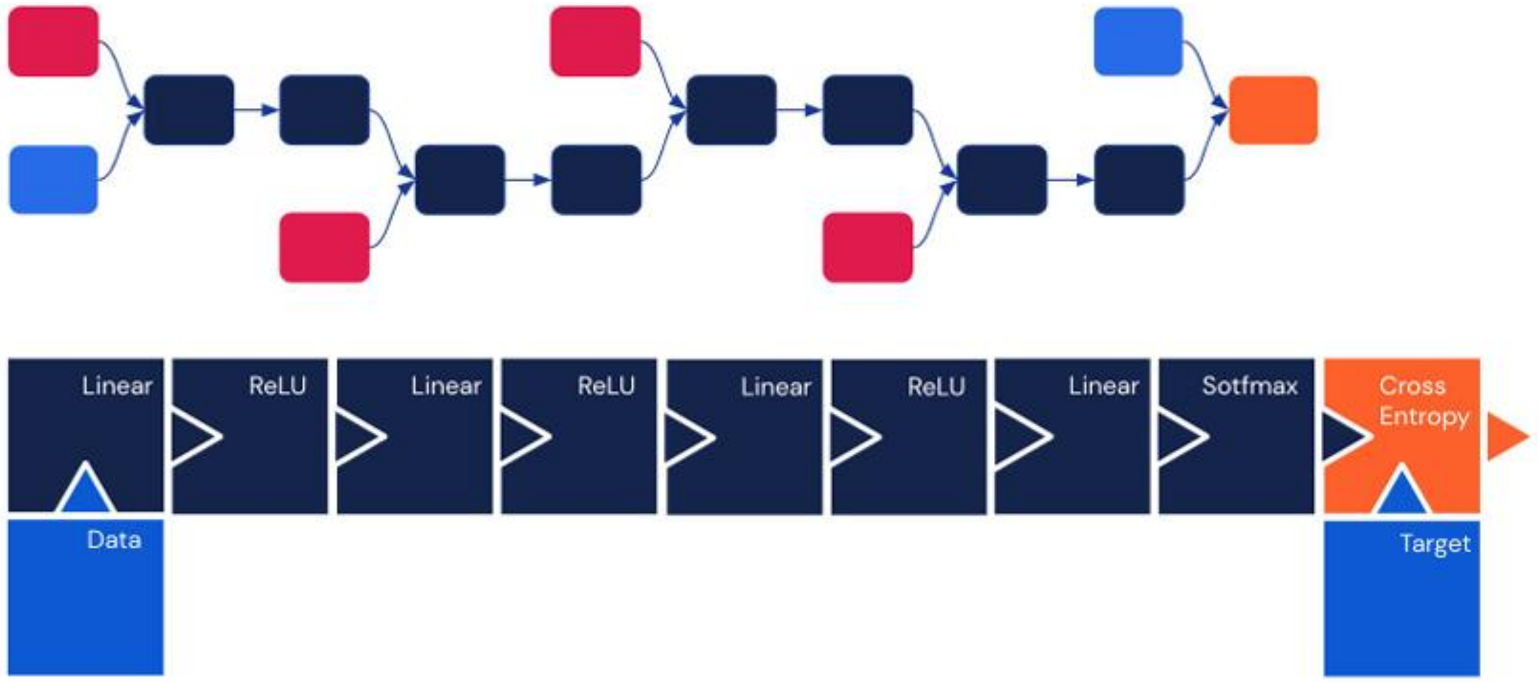
Redes recurrentes

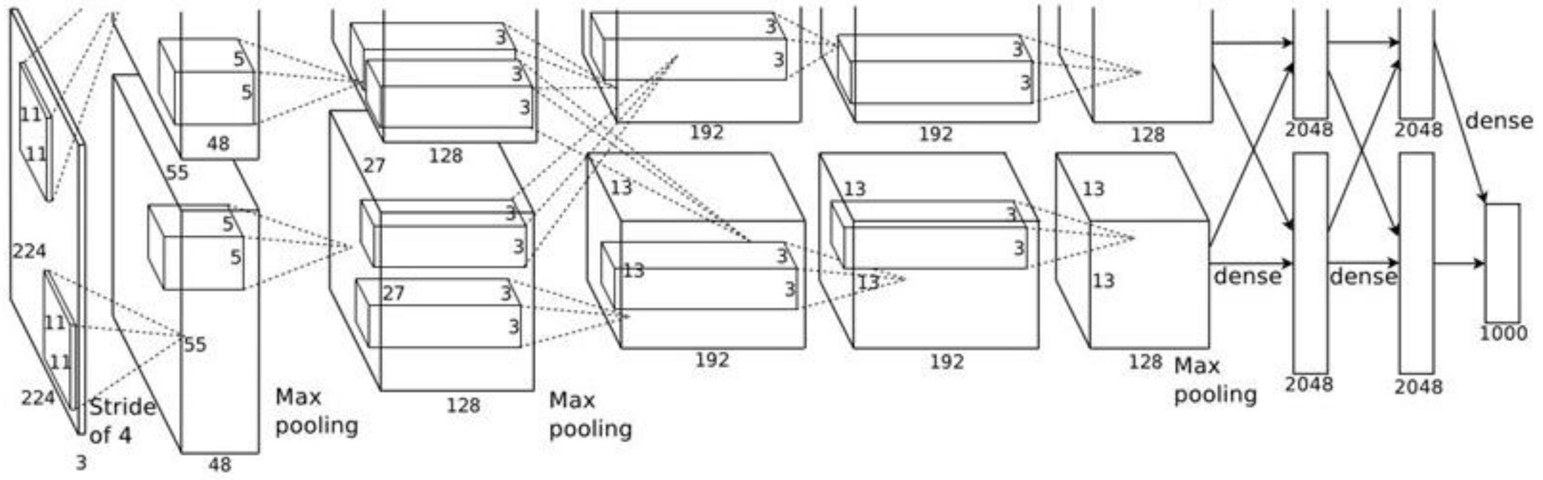
LSTM, GRU

> Reinventa el mundo <

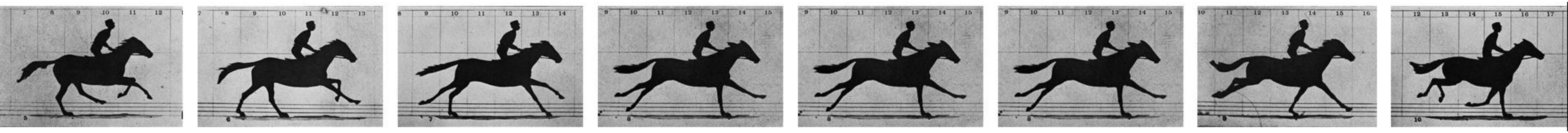
TRANSFORMATEC

Deep *Models*


Linear
ReLU
Linear
ReLU
Linear
ReLU
Linear
Sotfmax
Cross Entropy
Data
Target


224
224
3
11
11
Stride of 4
55
48
Max pooling
27
128
Max pooling
13
192
128
2048
dense
1000

Secuencias



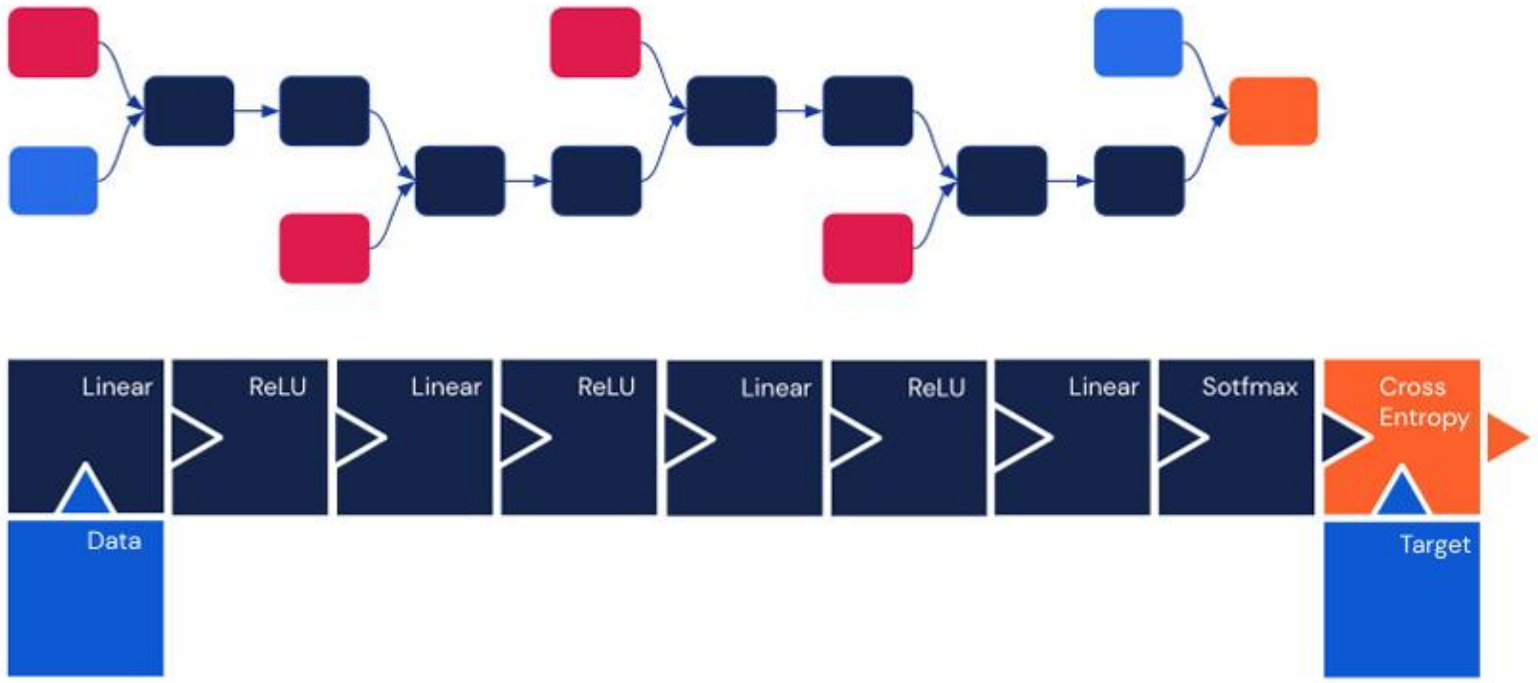
Colecciones de datos donde:

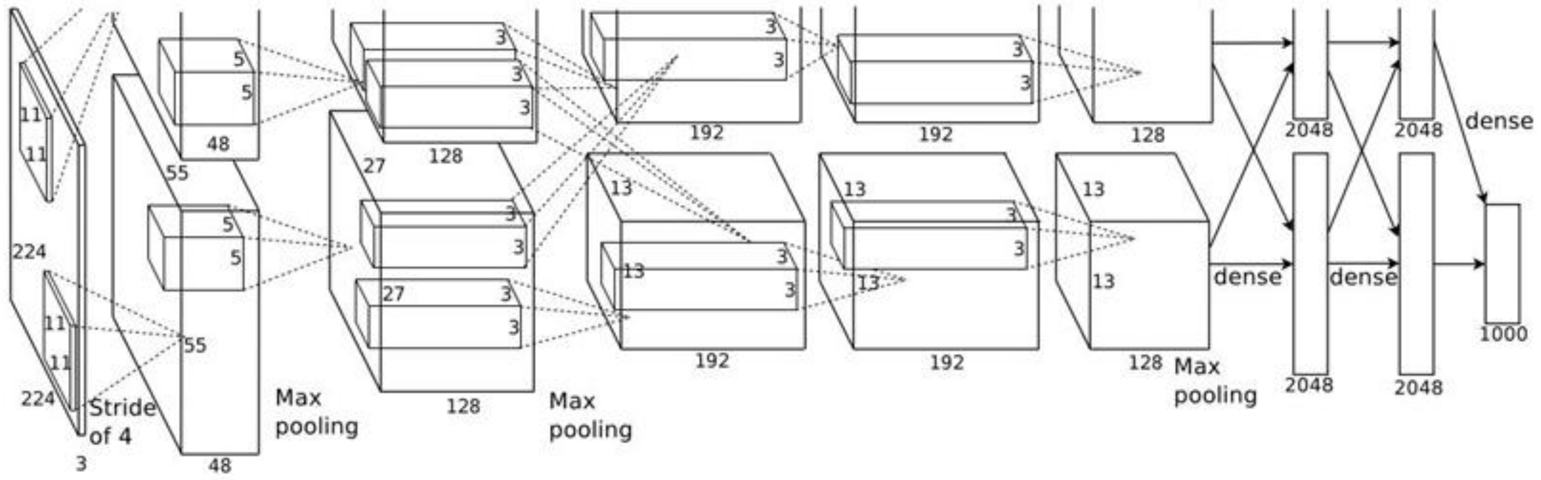
- Los elementos pueden **repetirse**
- El **orden** importa
- De longitud **variable** (potencialmente infinita)

Modelar *Secuencias*

- Los elementos pueden **repetirse**
- El **orden** importa
- De longitud **variable** (potencialmente infinita)

Los modelos previos no funcionan bien con datos secuenciales





Secuencias

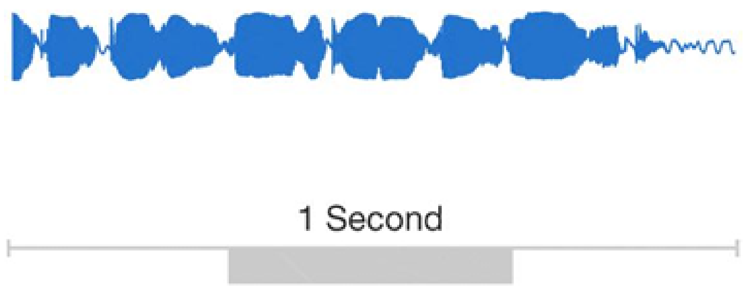
"Sequences really seem to be everywhere! We should learn how to model them. What is the best way to do that? Stay tuned!"

Palabras, letras

Secuencias

"Sequences really seem to be everywhere! We should learn how to model them. What is the best way to do that? Stay tuned!"

Palabras, letras

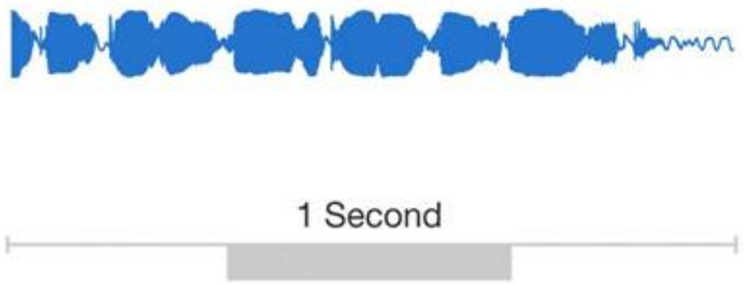


Habla

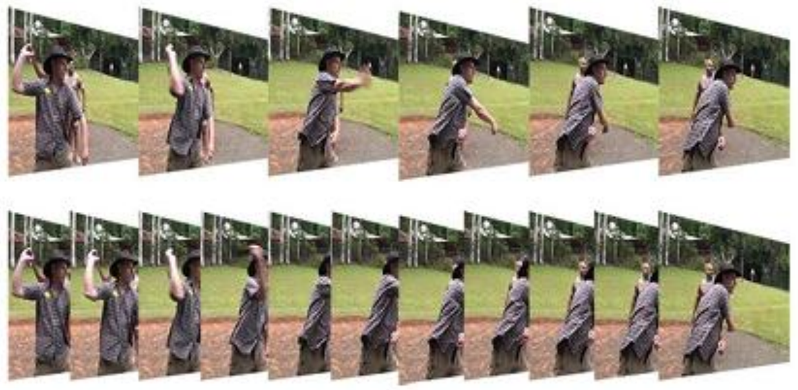
Secuencias

"Sequences really seem to be everywhere! We should learn how to model them. What is the best way to do that? Stay tuned!"

Palabras, letras



Habla

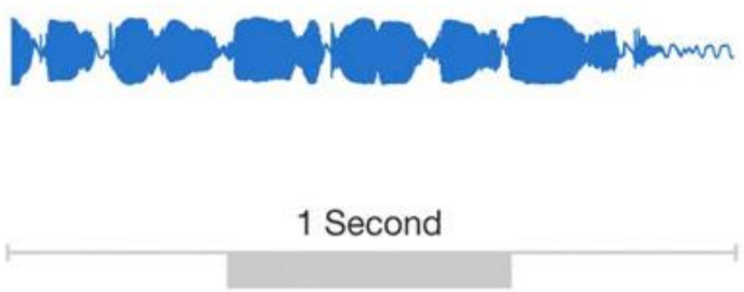


Video

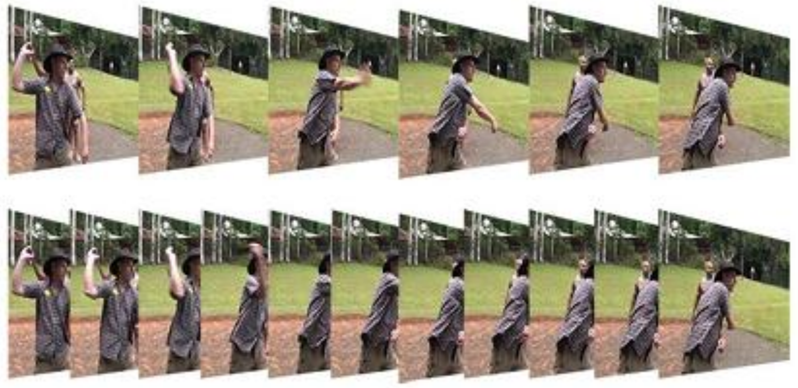
Secuencias

"Sequences really seem to be everywhere! We should learn how to model them. What is the best way to do that? Stay tuned!"

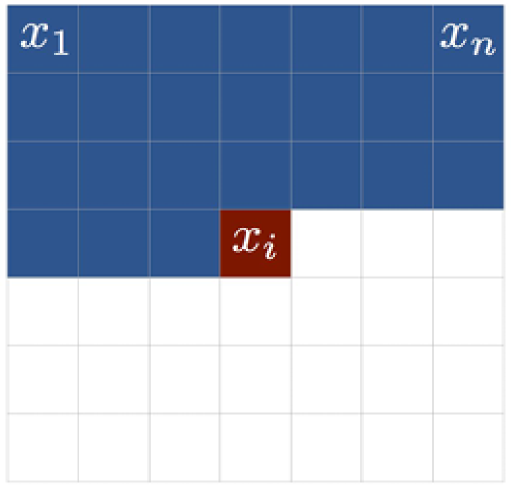
Palabras, letras



Habla



Video

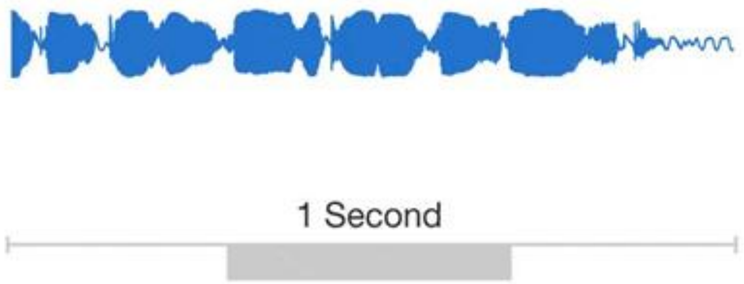


Imágenes

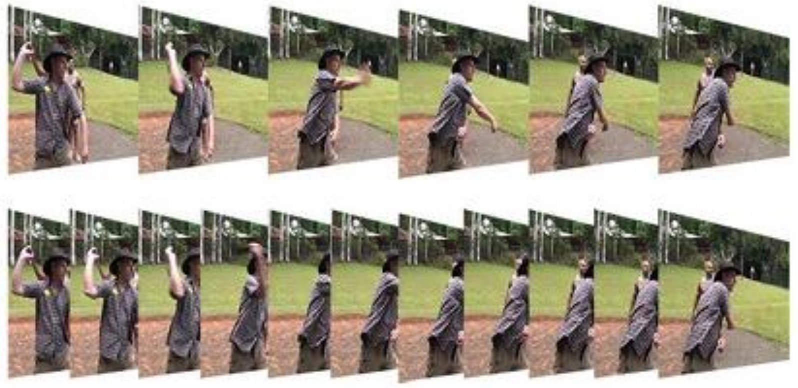
Secuencias

"Sequences really seem to be everywhere! We should learn how to model them. What is the best way to do that? Stay tuned!"

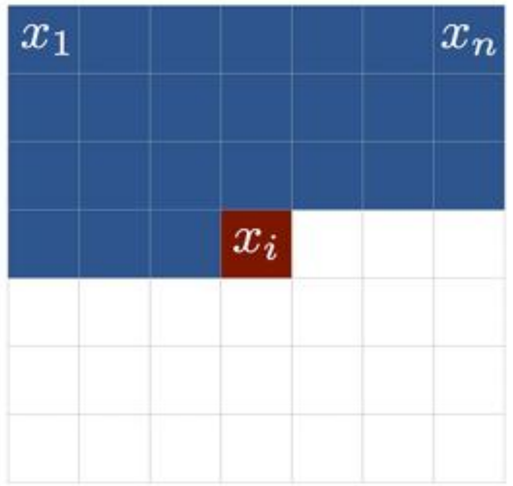
Palabras, letras



Habla



Video



Imágenes

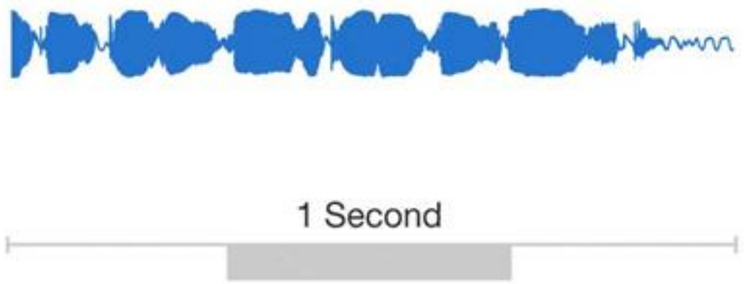
```python
import numpy as np
import matplotlib.pyplot as plt

def forward_backward_prop(w, T):
  hs = [0.5]
  for _ in range(T):
    hs.append(np.tanh(w*hs[-1]))

  dh = 1
  for t in range(T):
    dh = (1-hs[-1-t] ** 2) * w * dh

  return hs[-1], dh

T = 10  # sequence length
wlim = 4  #limit of interval over weights w

results = []
ws = np.linspace(-wlim, wlim, 1000)
for w in ws:
  results.append(forward_backward_prop(w, T))

plt.plot(ws, [r[0] for r in results], label='RNN state')
plt.plot(ws, [r[1] for r in results], label='Gradients')
```

Programas

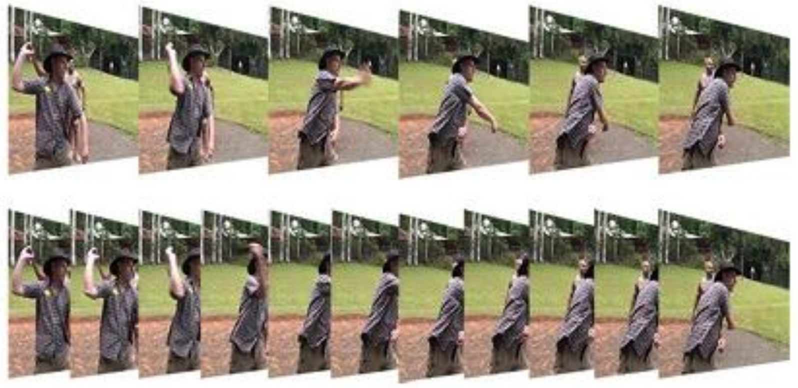
Secuencias

"Sequences really seem to be everywhere! We should learn how to model them. What is the best way to do that? Stay tuned!"

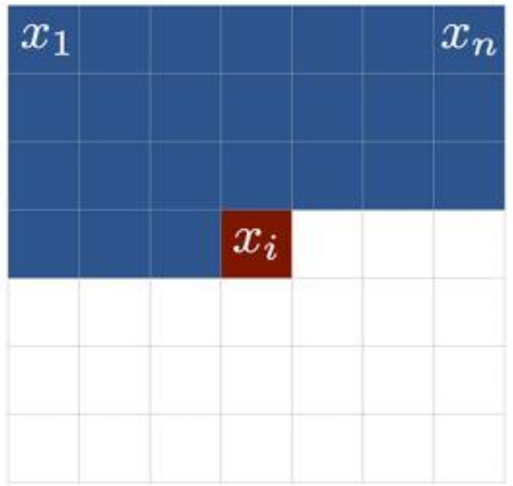
Palabras, letras



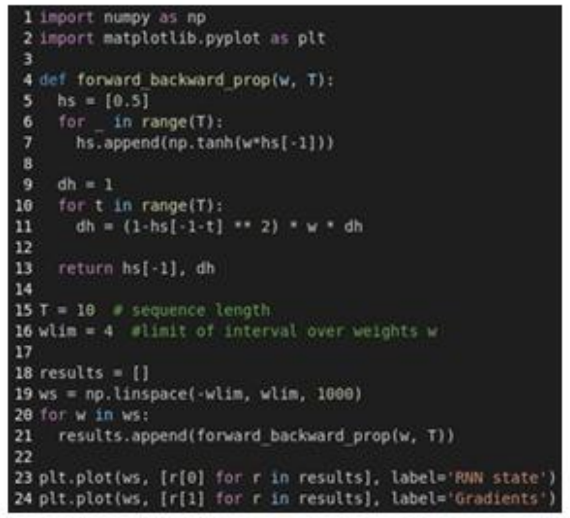
Habla



Video

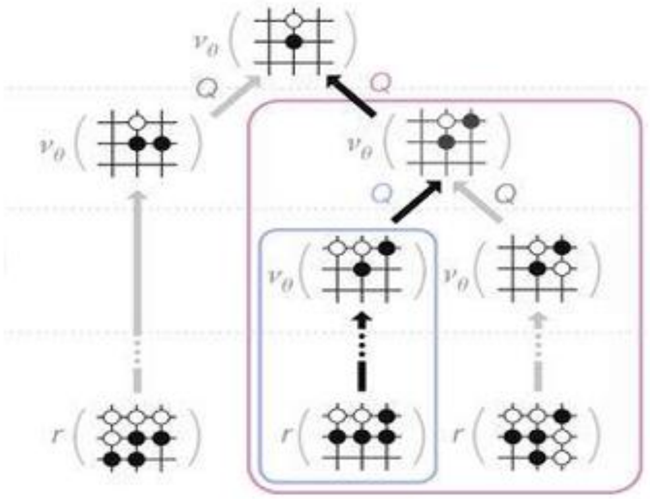


Imágenes



```python
import numpy as np
import matplotlib.pyplot as plt

def forward_backward_prop(w, T):
  hs = [0.5]
  for _ in range(T):
    hs.append(np.tanh(w*hs[-1]))

  dh = 1
  for t in range(T):
    dh = (1-hs[-1-t] ** 2) * w * dh

  return hs[-1], dh

T = 10  # sequence length
wlim = 4  #limit of interval over weights w

results = []
ws = np.linspace(-wlim, wlim, 1000)
for w in ws:
  results.append(forward_backward_prop(w, T))

plt.plot(ws, [r[0] for r in results], label='RNN state')
plt.plot(ws, [r[1] for r in results], label='Gradients')
```

Programas



Toma de decisiones

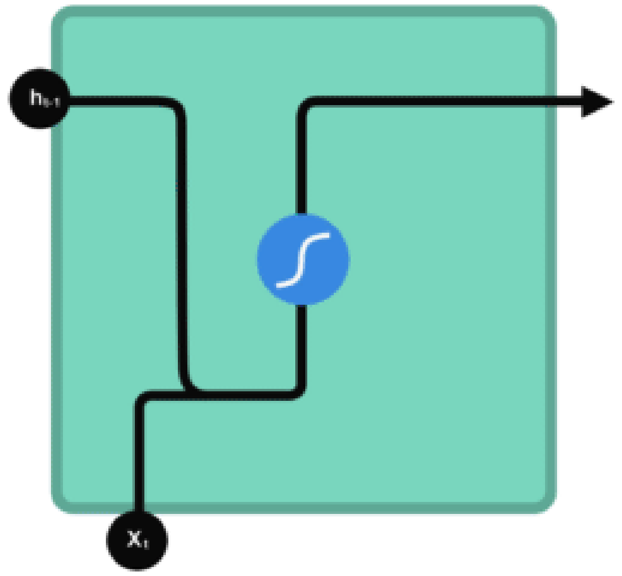
UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

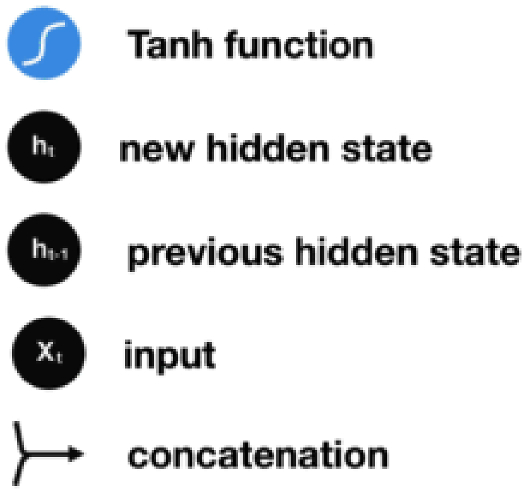
1.

RNN *Tradicionales*

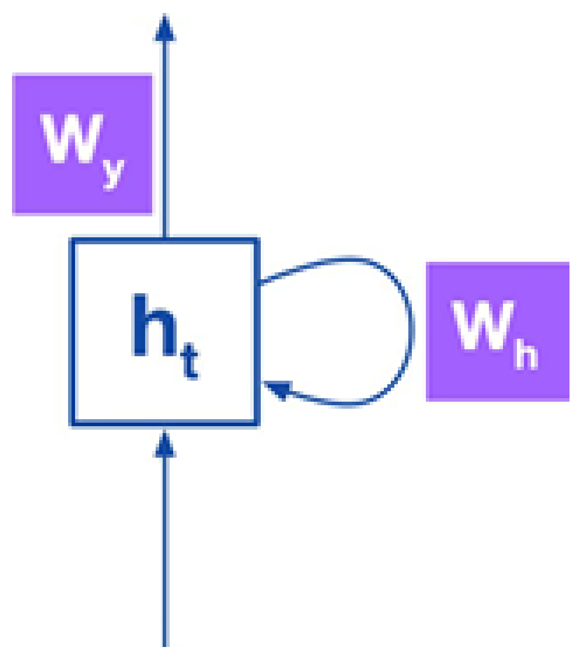
TRANSFORMATEC

Arquitectura *de las RNN*





Recurrent Neural Network



Alguna función
con parámetro W_h

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

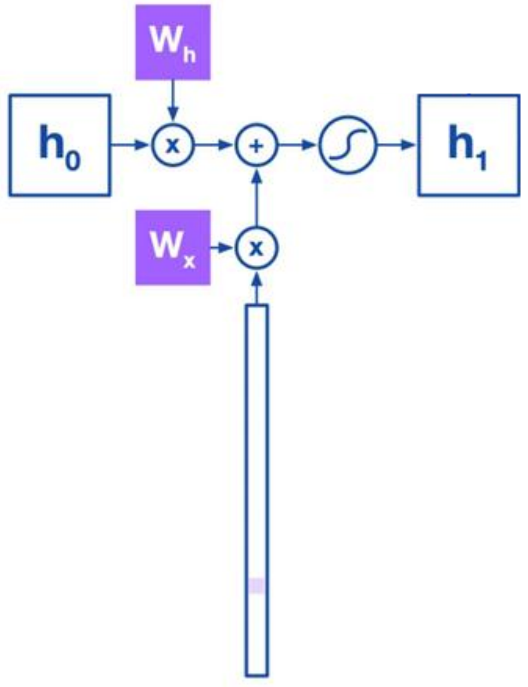
Nuevo estado Estado previo Nueva entrada

Alguna función
con parámetro W_y

$$y_t = f_{W_y}(h_t)$$

Salida actual Nuevo estado

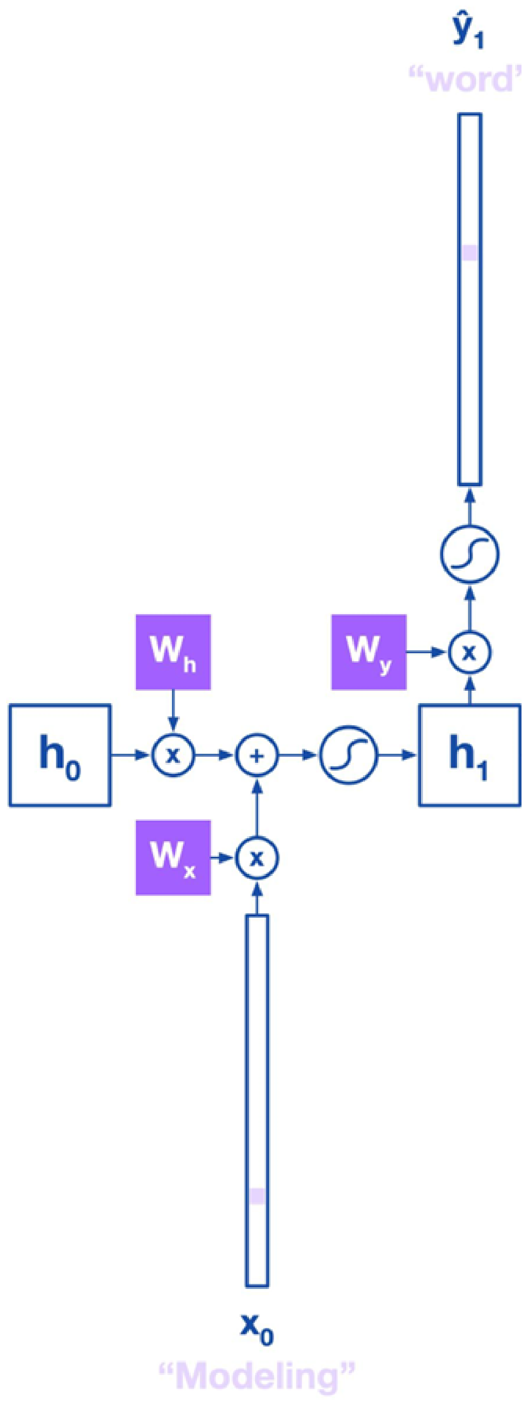
Recurrent Neural Network



Se recibe la primera palabra de entrada x_0.

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

Recurrent Neural Network

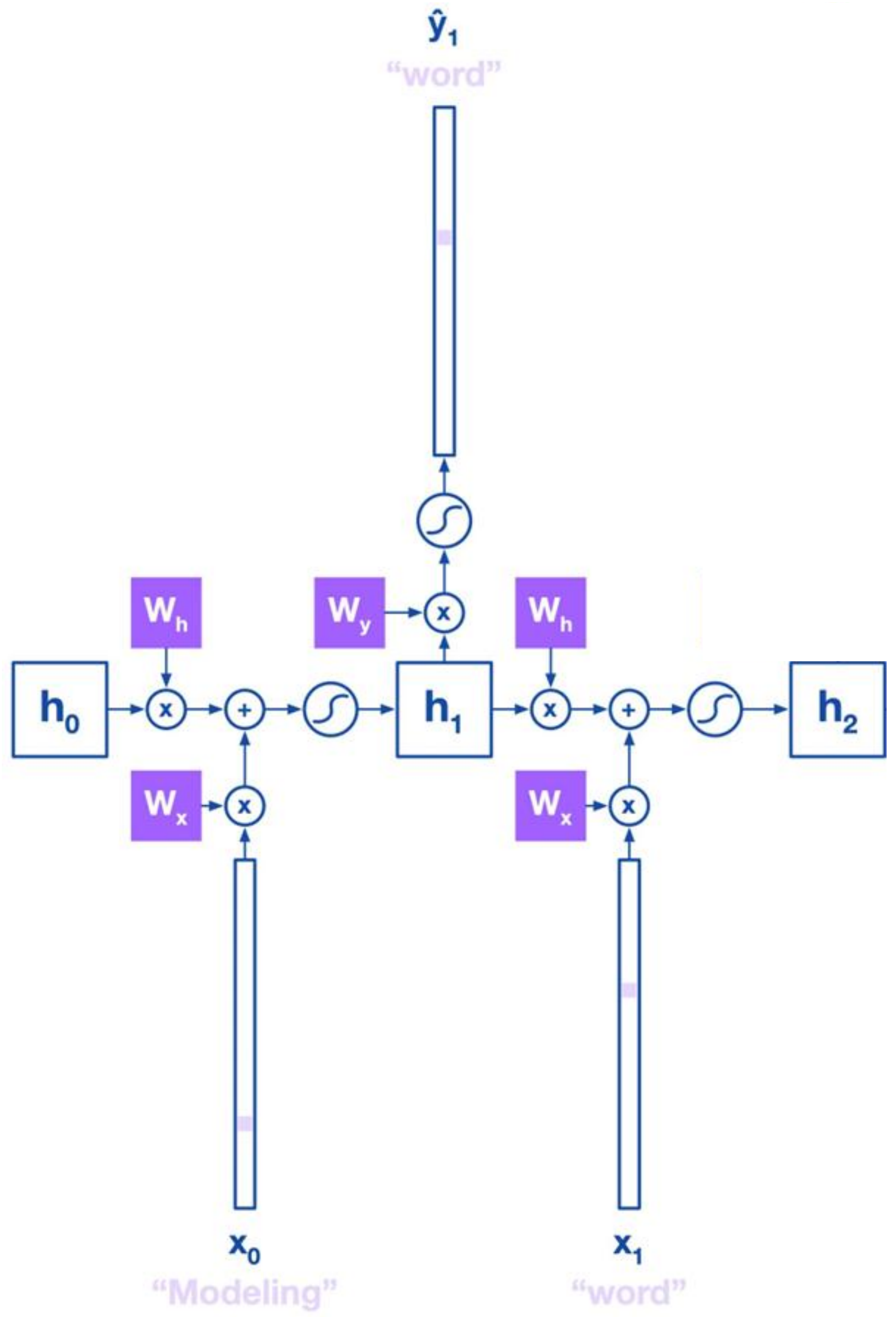


RNN predicen la salida y_1 (la siguiente palabra) a partir del estado h_1.

$$y_t = \text{Softmax}(W_y h_t)$$

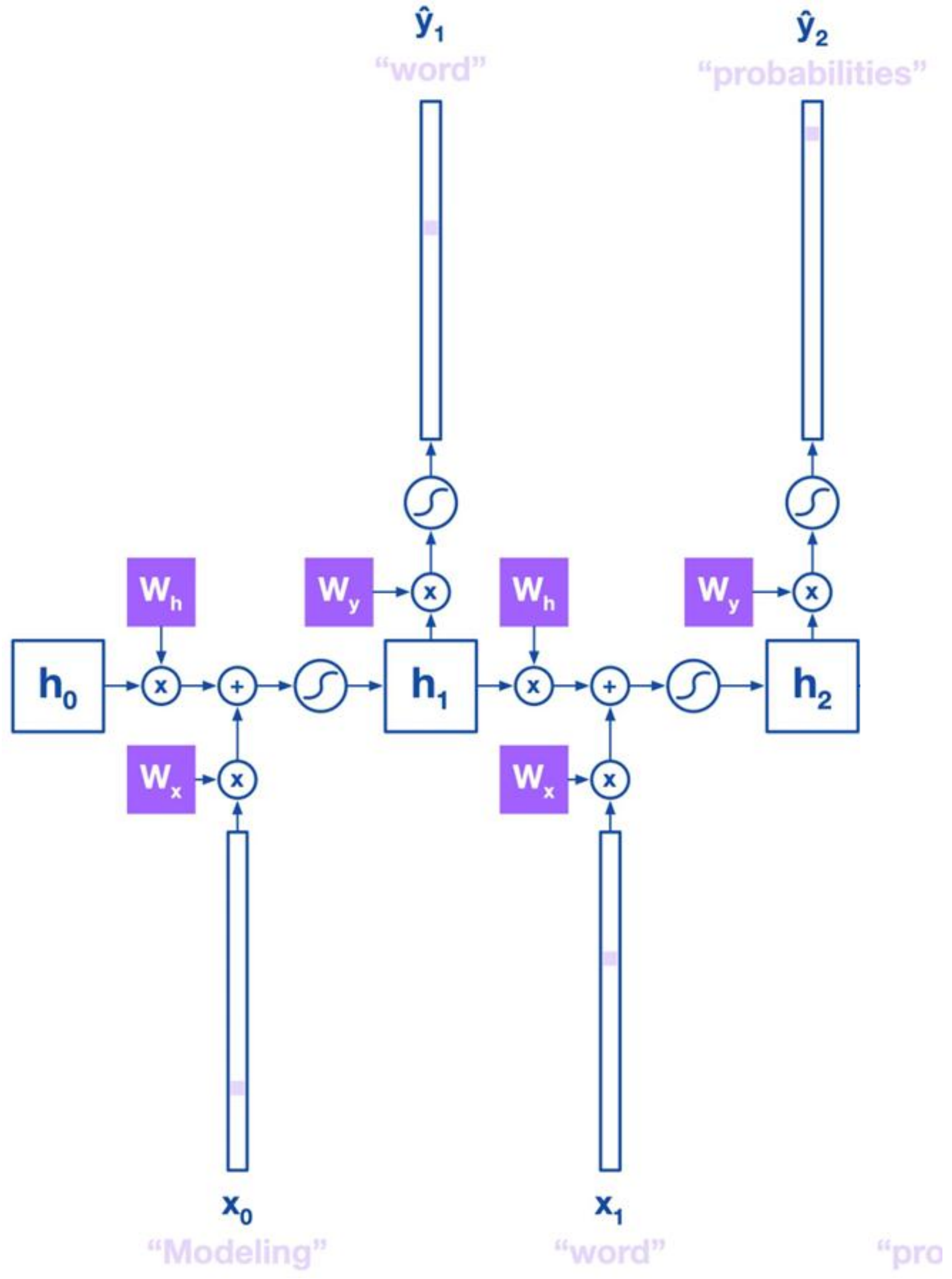
Softmax genera una distribución de probabilidad entre todas las palabras posibles.

Recurrent Neural Network

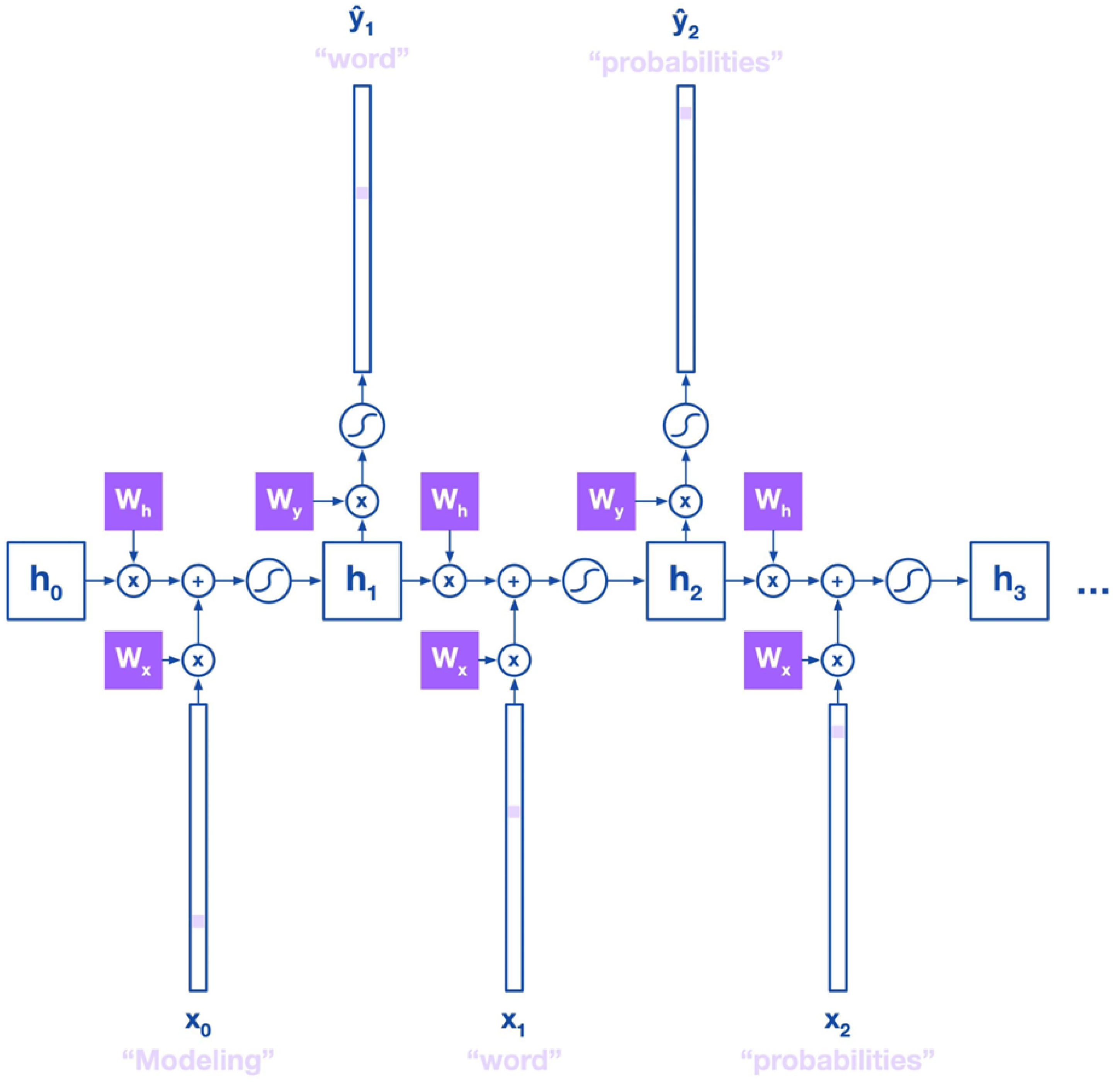


Siguiente palabra de la frase x_1 como entrada

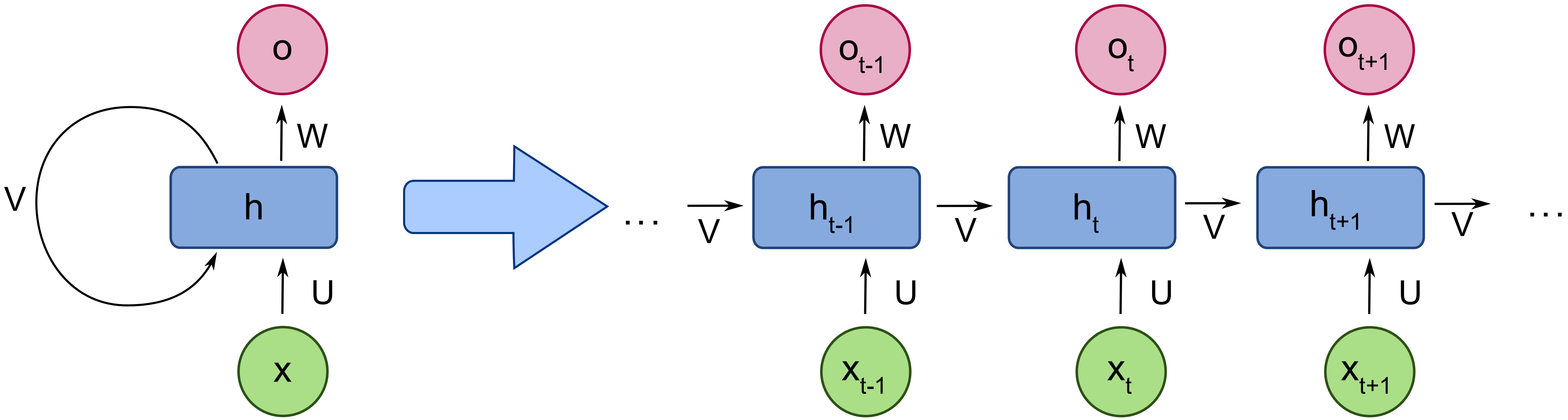
Recurrent Neural Network



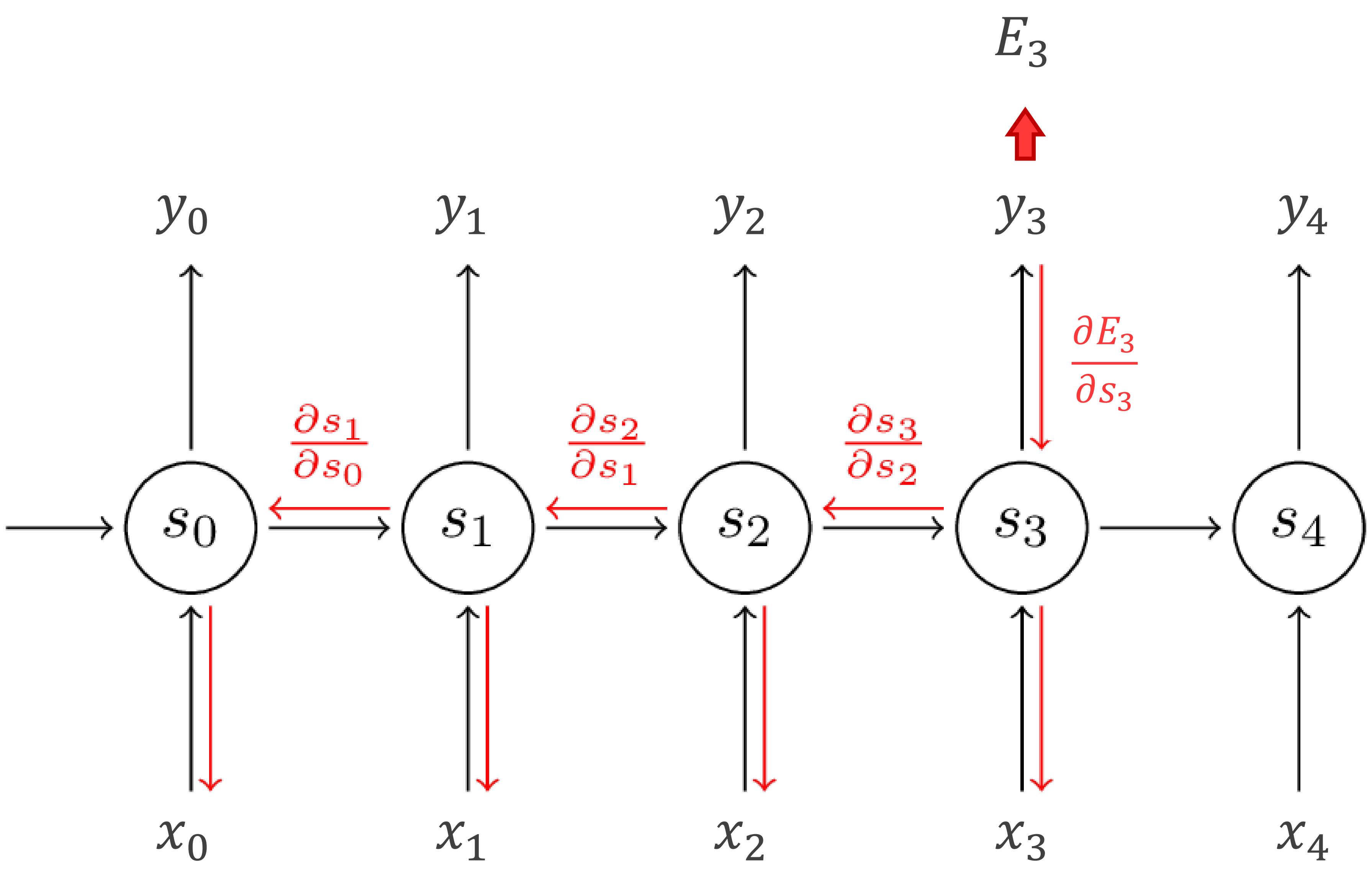
Recurrent Neural Network



Recurrent Neural Network



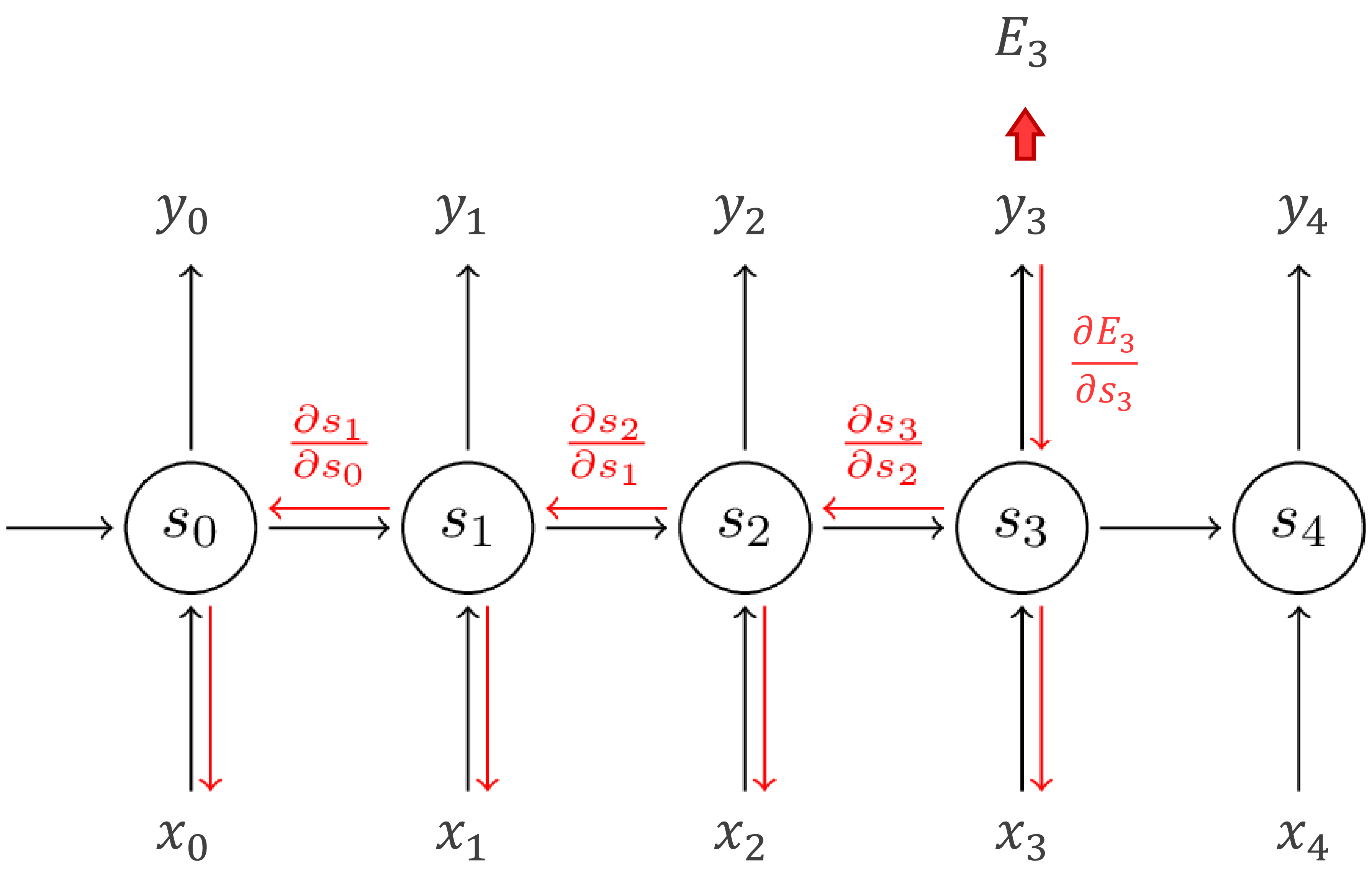
Vanilla RNN *Gradient Flow*



Función de error:

$$E_t(y_t, \hat{y}_t) = -y_t \log \hat{y}_t$$

Vanilla RNN *Gradient Flow*



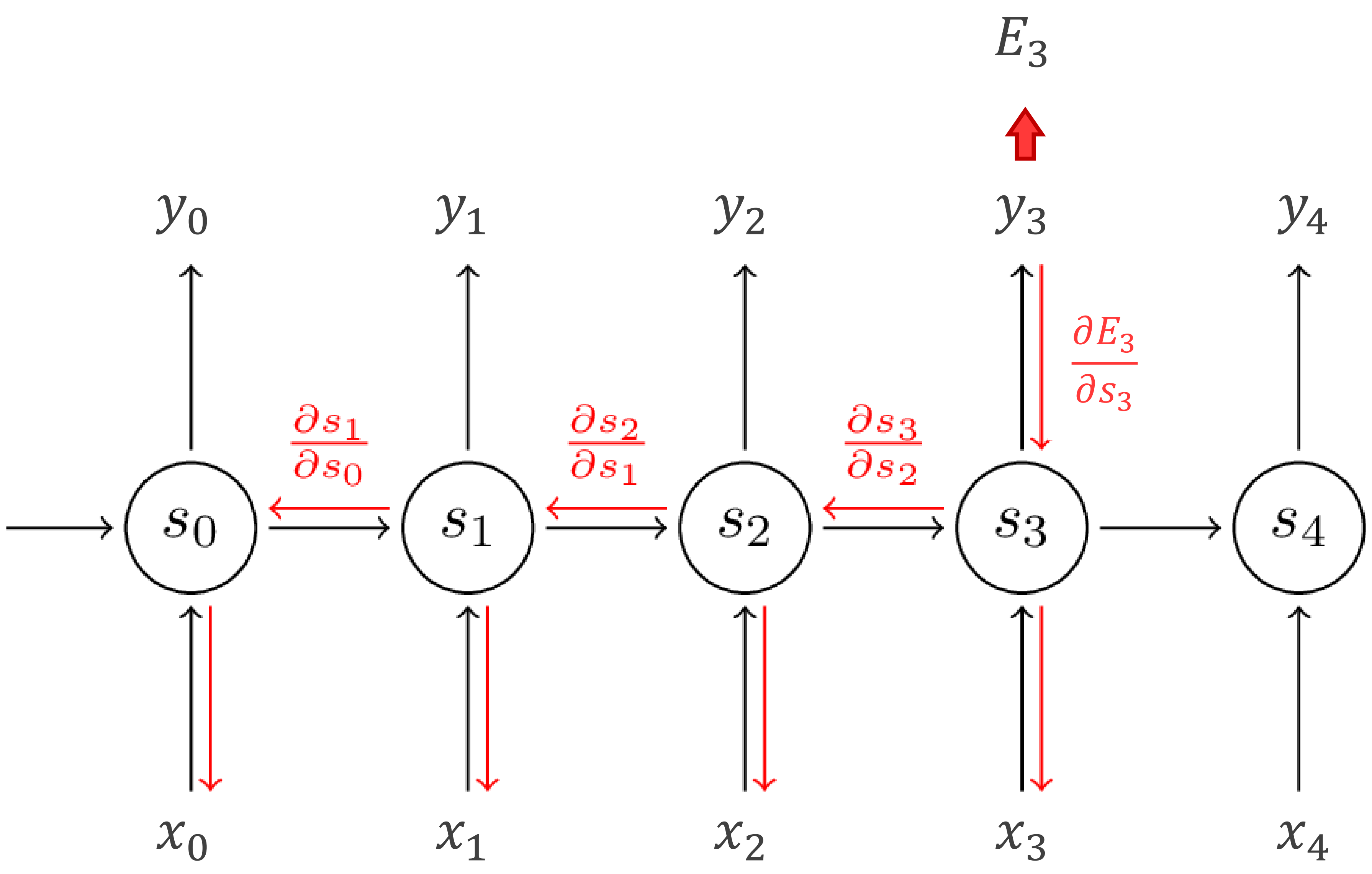
Función de error:

$$E_t(y_t, \hat{y}_t) = -y_t \log \hat{y}_t$$

Cálculo del gradiente:

$$\frac{\partial E}{\partial W} = \sum_{1 \leq t \leq T} \frac{\partial E_t}{\partial W}$$

Vanilla RNN *Gradient Flow*



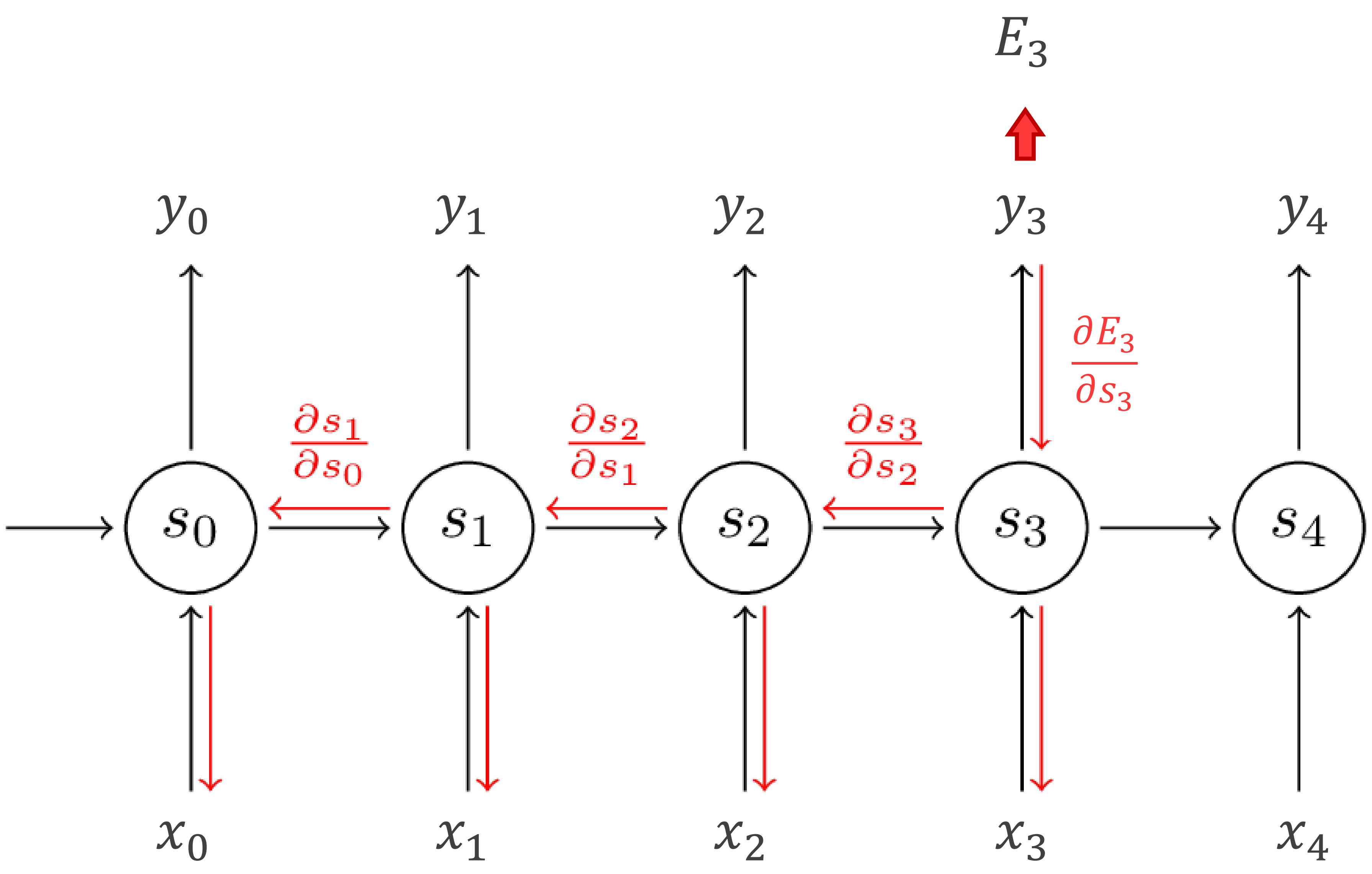
Función de error:

$$E_t(y_t, \hat{y}_t) = -y_t \log \hat{y}_t$$

Cálculo del gradiente:

$$\frac{\partial E}{\partial W} = \sum_{1 \le t \le T} \frac{\partial E_t}{\partial W} = \sum_{1 \le k \le t} \left(\frac{\partial E_t}{\partial x_t} \frac{\partial x_t}{\partial x_k} \frac{\partial x_k}{\partial W} \right)$$

Vanilla RNN *Gradient Flow*



Función de error:

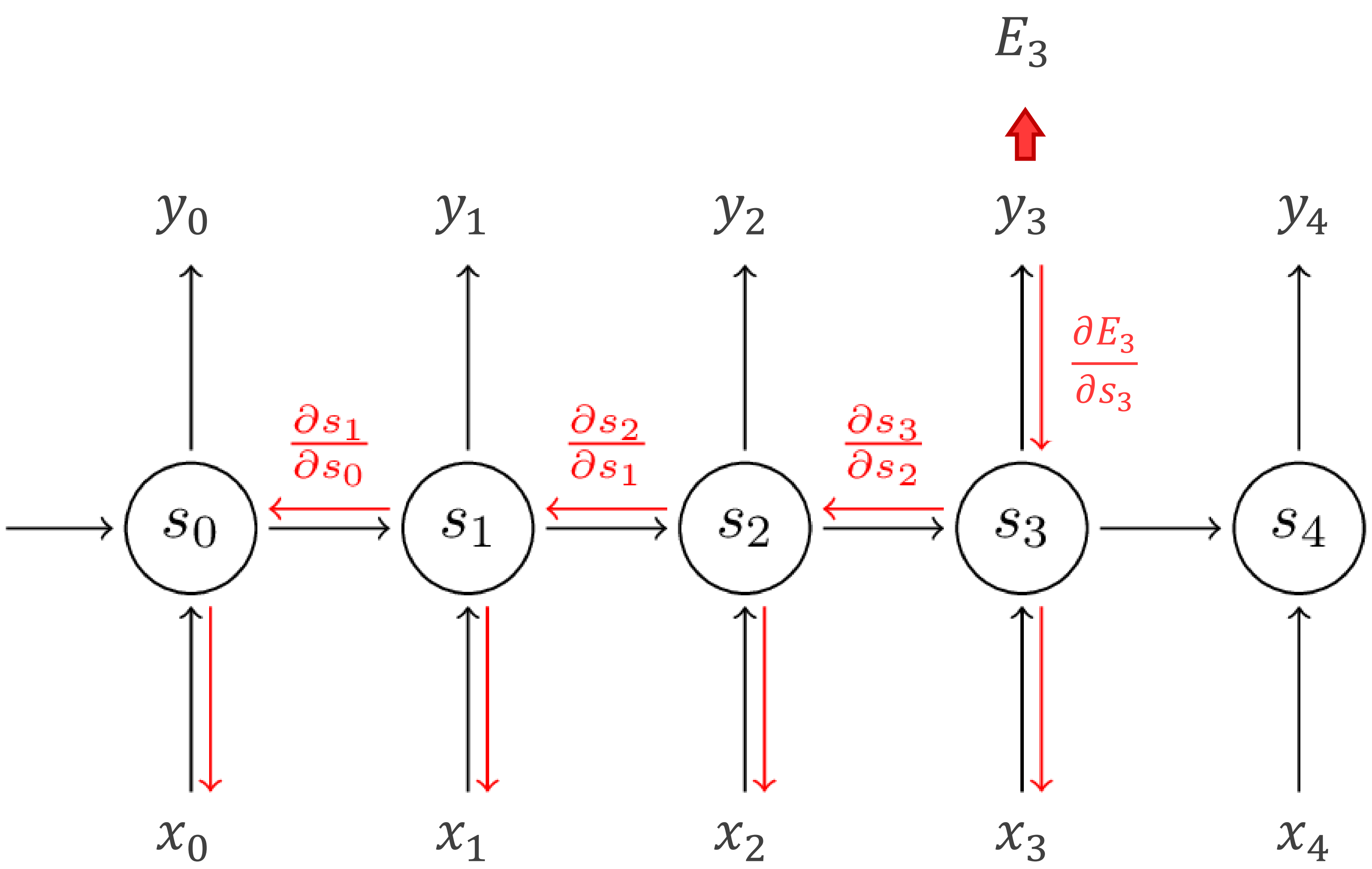
$$E_t(y_t, \hat{y}_t) = -y_t \log \hat{y}_t$$

Cálculo del gradiente:

$$\frac{\partial E}{\partial W} = \sum_{1 \le t \le T} \frac{\partial E_t}{\partial W} = \sum_{1 \le k \le t} \left(\frac{\partial E_t}{\partial x_t} \frac{\partial x_t}{\partial x_k} \frac{\partial x_k}{\partial W} \right)$$

$$\frac{\partial x_t}{\partial x_k} = \prod_{t \ge i > k} \frac{\partial x_i}{\partial x_{i-1}} = \prod_{t \ge i > k} W^T \operatorname{diag}(\sigma(x_{i-1}))$$

Vanilla RNN *Gradient Flow*



Función de error:

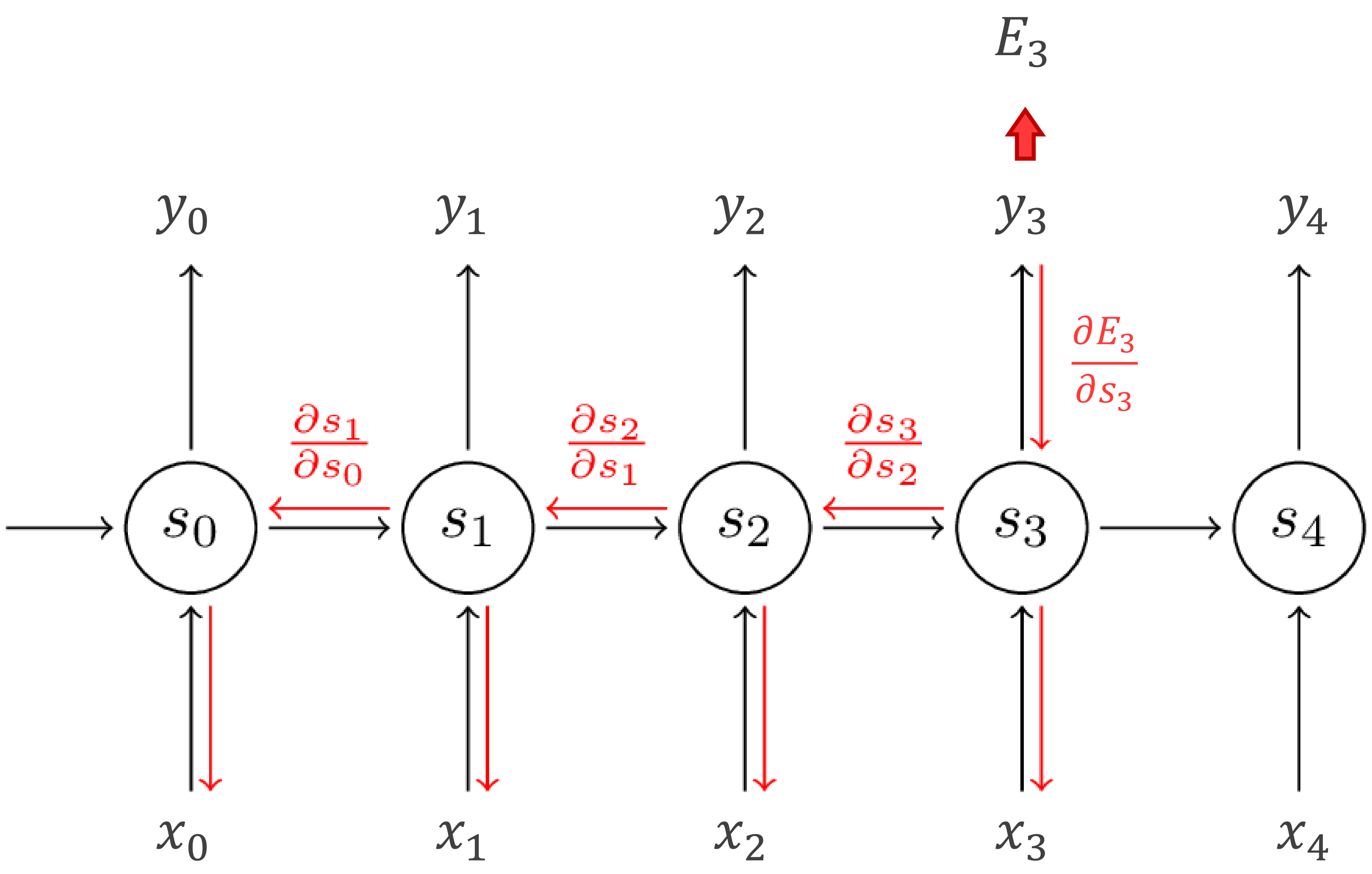
$$E_t(y_t, \hat{y}_t) = -y_t \log \hat{y}_t$$

Cálculo del gradiente:

$$\frac{\partial E}{\partial W} = \sum_{1 \le t \le T} \frac{\partial E_t}{\partial W} = \sum_{1 \le k \le t} \left(\frac{\partial E_t}{\partial x_t} \frac{\partial x_t}{\partial x_k} \frac{\partial x_k}{\partial W} \right)$$

$$\frac{\partial x_t}{\partial x_k} = \prod_{t \ge i > k} \frac{\partial x_i}{\partial x_{i-1}} = \prod_{t \ge i > k} \boxed{W^T} \operatorname{diag}(\sigma(x_{i-1}))$$

Vanishing *Gradients*



Función de error:

$$E_t(y_t, \hat{y}_t) = -y_t \log \hat{y}_t$$

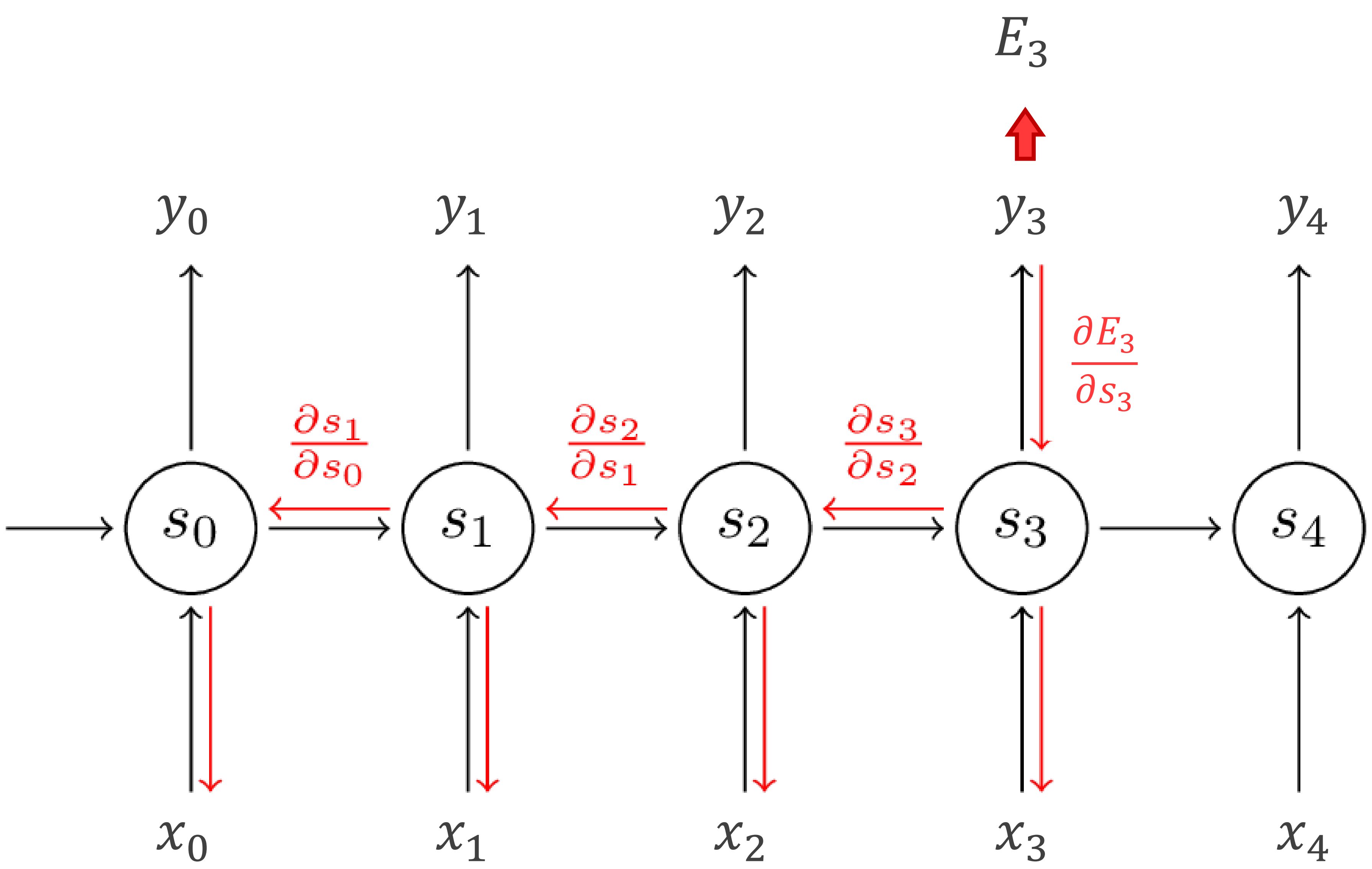
Cálculo del gradiente:

$$\frac{\partial E}{\partial W} = \sum_{1 \le t \le T} \frac{\partial E_t}{\partial W} = \sum_{1 \le k \le t} \left(\frac{\partial E_t}{\partial x_t} \frac{\partial x_t}{\partial x_k} \frac{\partial x_k}{\partial W} \right)$$

$$\frac{\partial x_t}{\partial x_k} = \prod_{t \ge i > k} \frac{\partial x_i}{\partial x_{i-1}} = \prod_{t \ge i > k} \boxed{W^T} \operatorname{diag}(\sigma(x_{i-1}))$$

Si $\|W^T\| < 1$: El gradiente **se desvanece.**

Exploding *Gradients*



Función de error:

$$E_t(y_t, \hat{y}_t) = -y_t \log \hat{y}_t$$

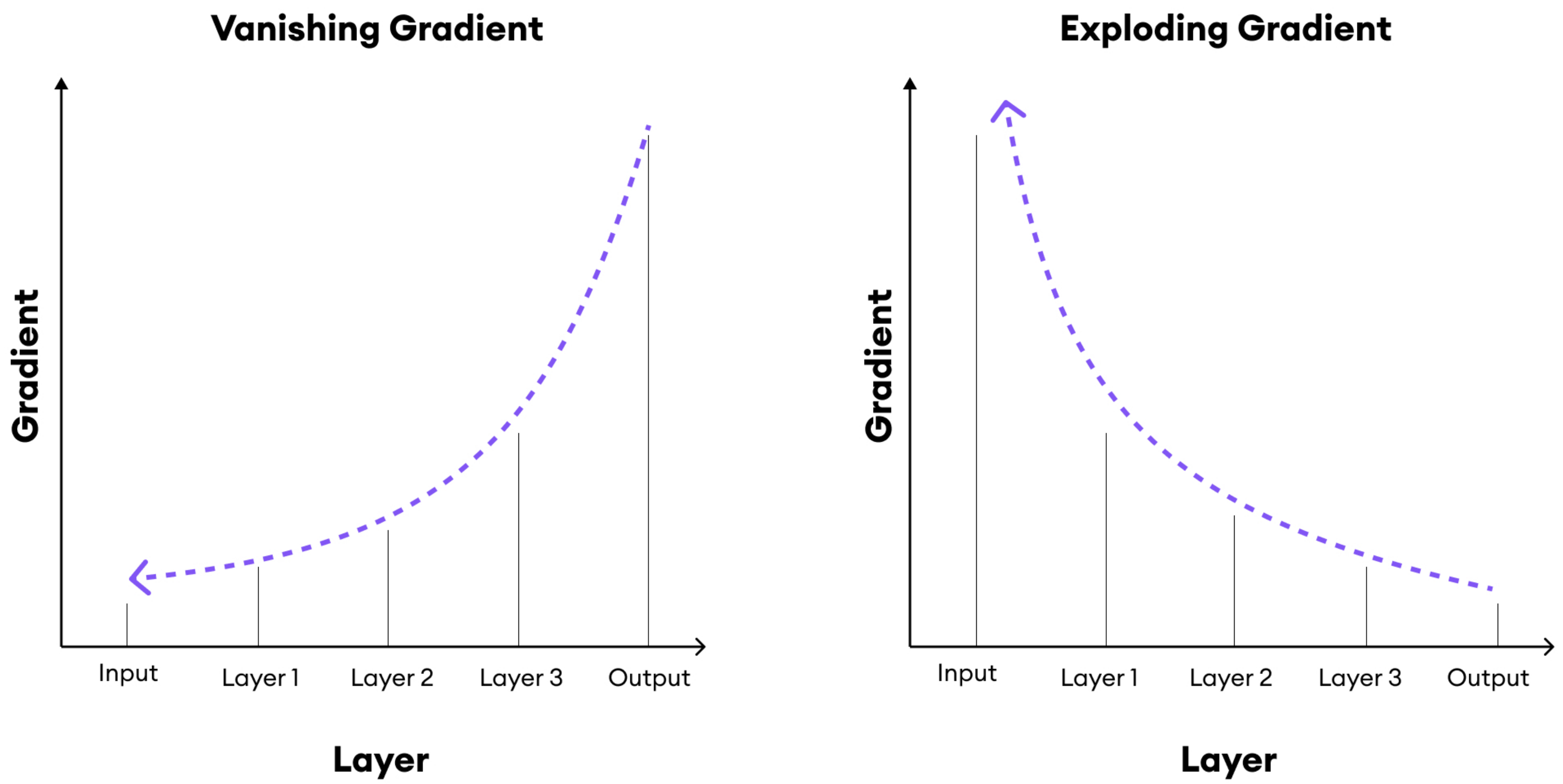
Cálculo del gradiente:

$$\frac{\partial E}{\partial W} = \sum_{1\le t\le T} \frac{\partial E_t}{\partial W} = \sum_{1\le k\le t} \left(\frac{\partial E_t}{\partial x_t} \frac{\partial x_t}{\partial x_k} \frac{\partial x_k}{\partial W} \right)$$

$$\frac{\partial x_t}{\partial x_k} = \prod_{t\ge i>k} \frac{\partial x_i}{\partial x_{i-1}} = \prod_{t\ge i>k} \boxed{W^T} \mathrm{diag}(\sigma(x_{i-1}))$$

Si $\|W^T\| > 1$: El gradiente **crece exponencialmente**

Vanilla RNN *Gradient Flow*

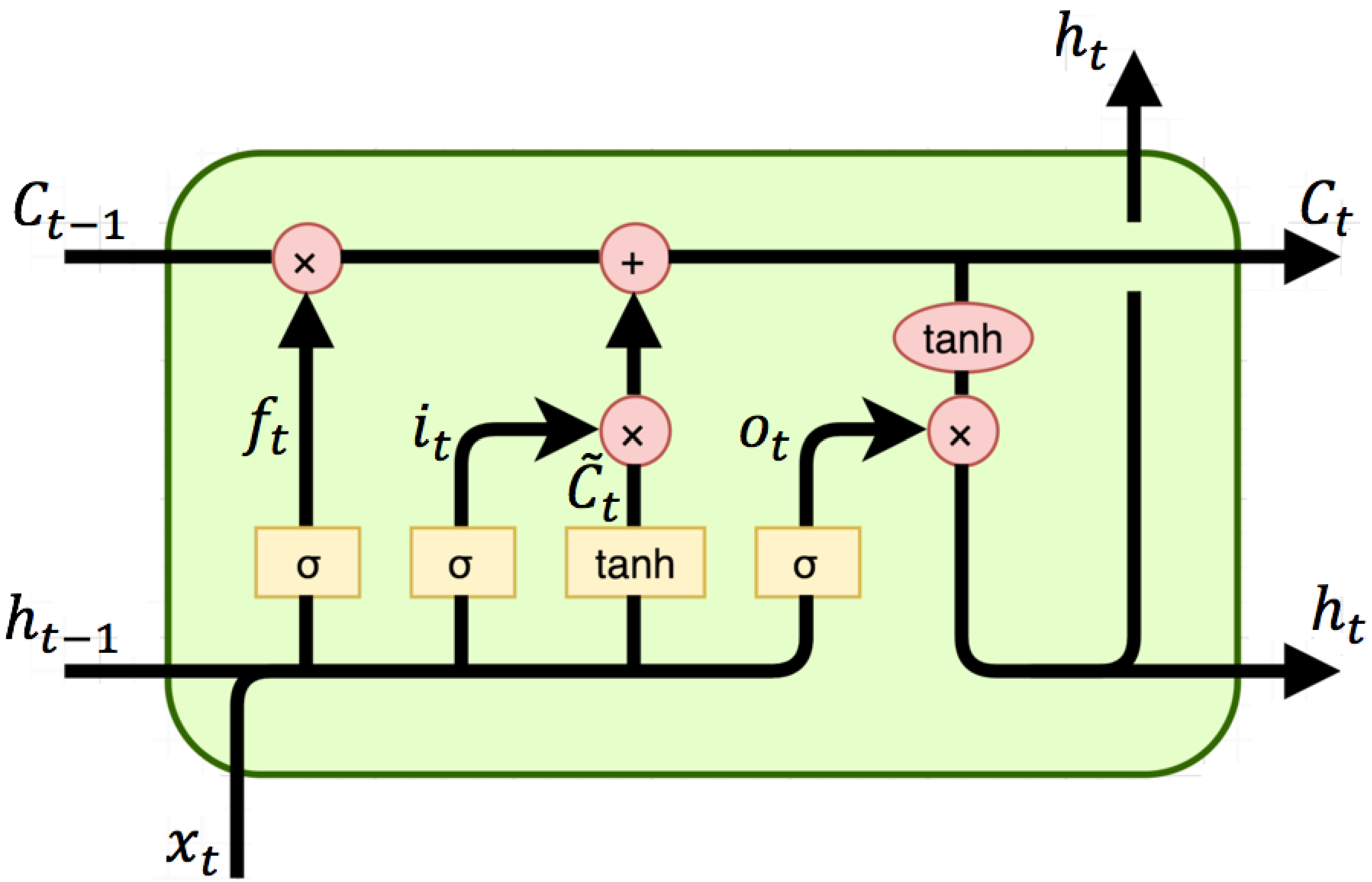


UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

2.

LSTM

Long short-term *memory (LSTM)*



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

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

$$o_t = \sigma(\omega_o \cdot [h_{t-1}, x_t] + b_o)$$

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

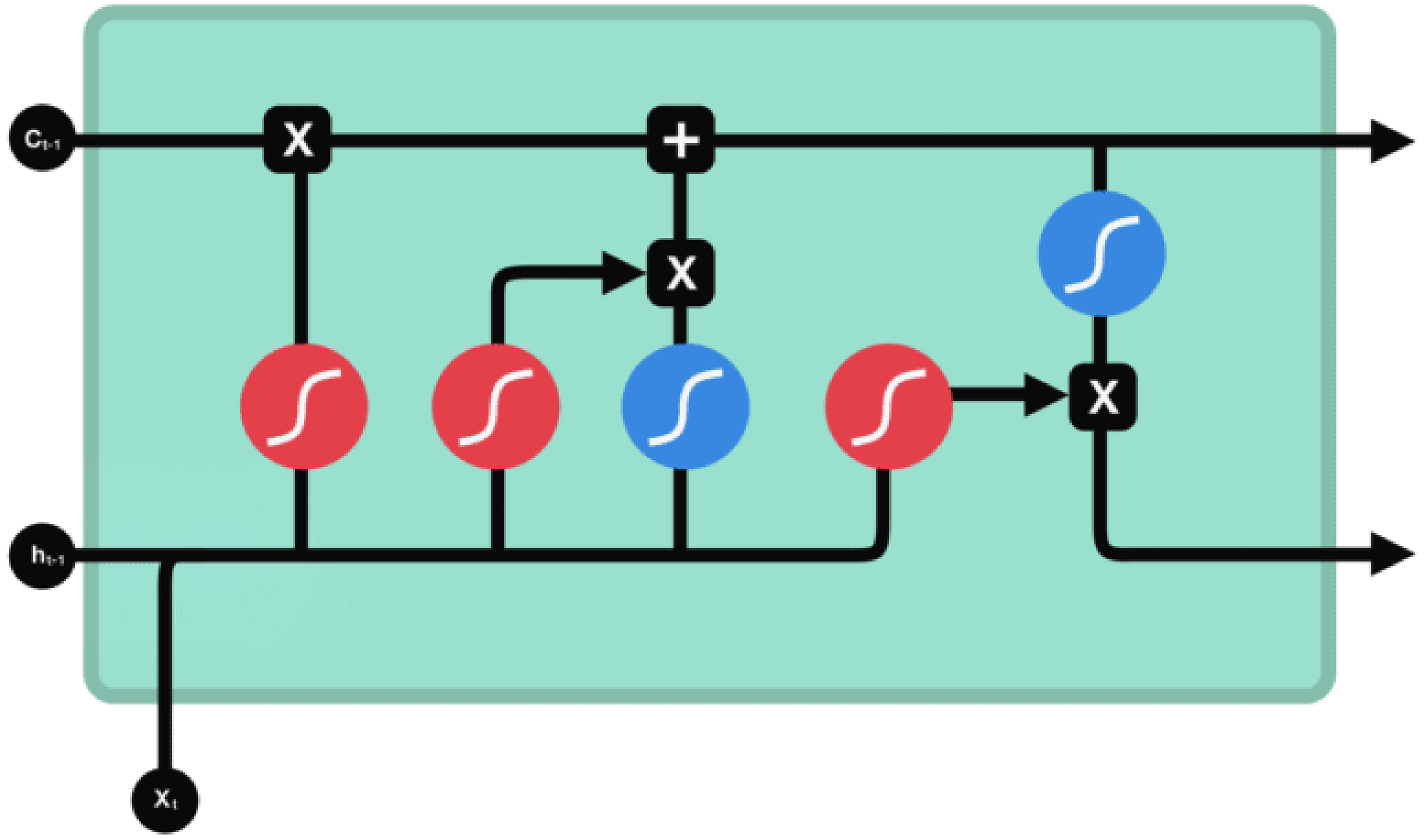
$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t$$

$$h_t = o_t \odot \tanh(C_t)$$

Sepp Hochreiter and Juergen Schmidhuber (1997) "Long Short-term Memory". Neural Computation MIT-Press.

Long short-term *memory (LSTM)*

Forget gate



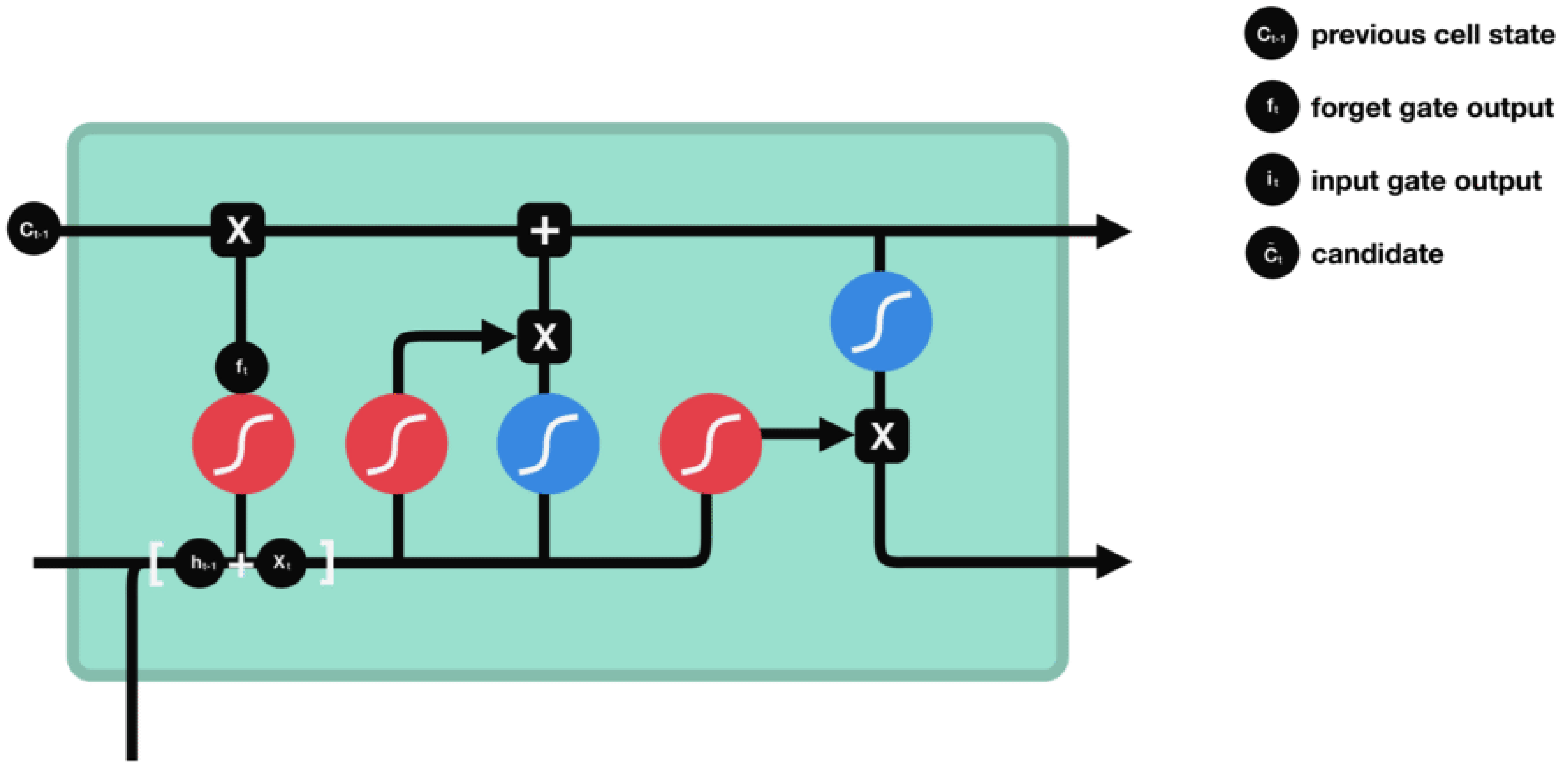
C_{t-1} previous cell state

f_t forget gate output

Sepp Hochreiter and Juergen Schmidhuber (1997) "Long Short-term Memory". Neural Computation MIT-Press.

Long short-term *memory (LSTM)*

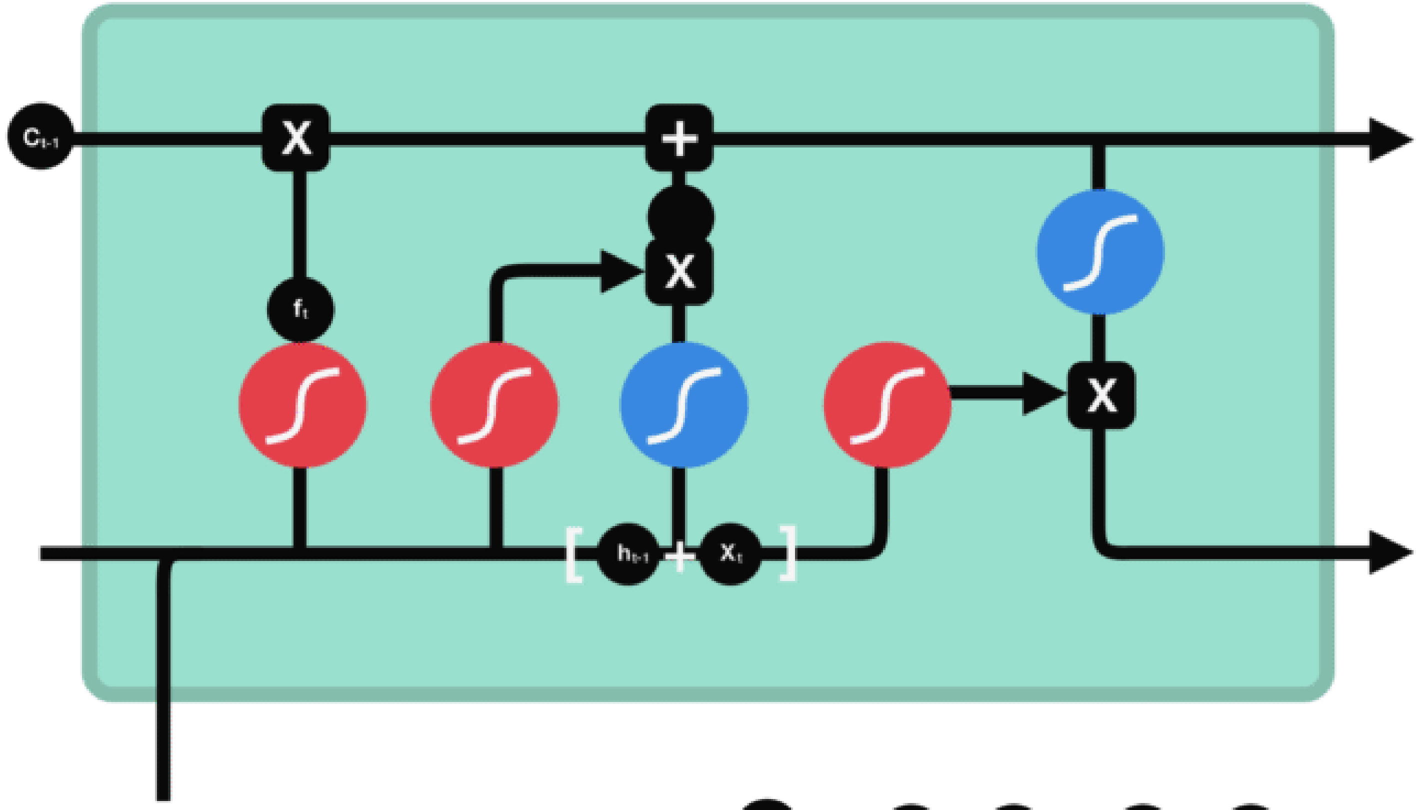
Input gate



Sepp Hochreiter and Juergen Schmidhuber (1997) "Long Short-term Memory".
Neural Computation MIT-Press.

Long short-term *memory (LSTM)*

Cell state



C_{t-1} previous cell state

f_t forget gate output

i_t input gate output

$\tilde{C}_t$ candidate

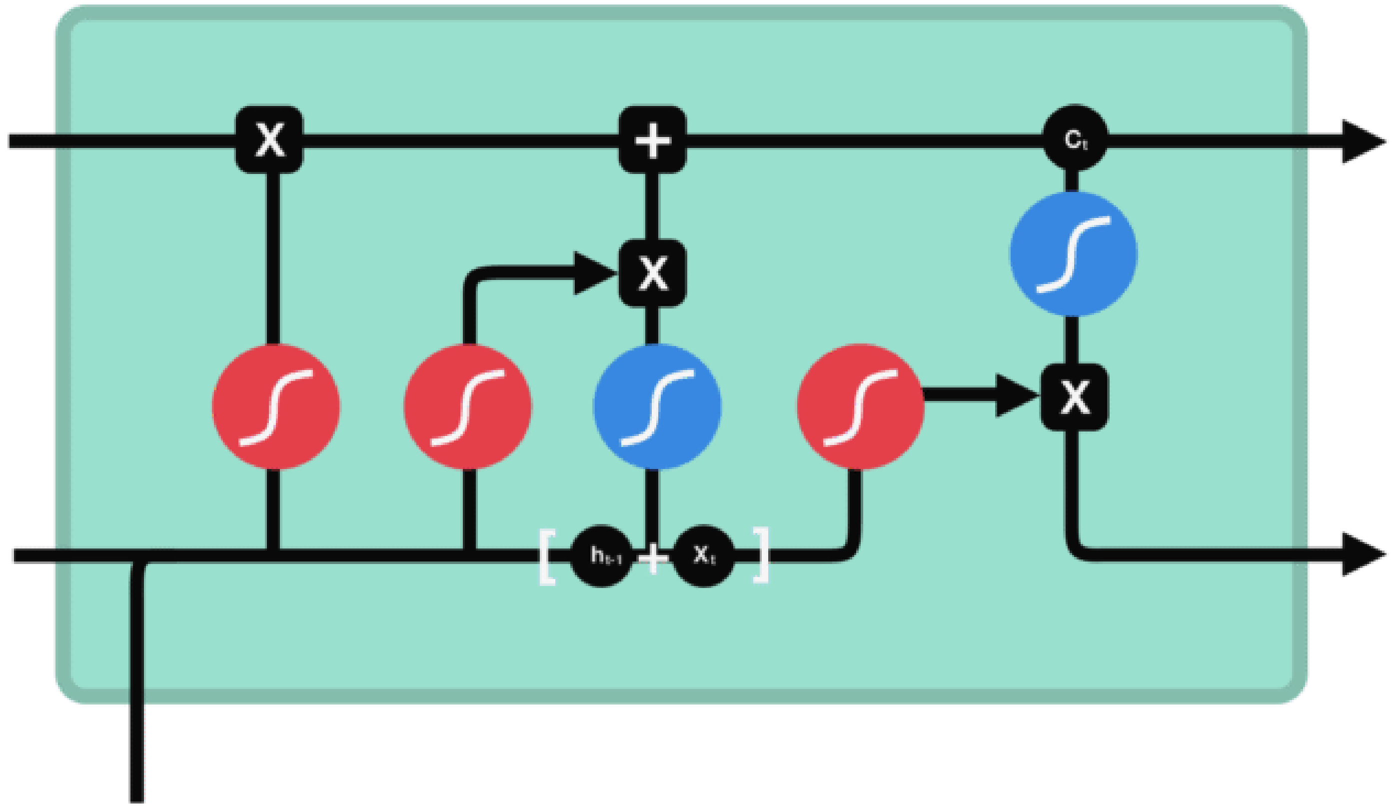
C_t new cell state

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

Sepp Hochreiter and Juergen Schmidhuber (1997) “Long Short-term Memory”. Neural Computation MIT-Press.

Long short-term *memory (LSTM)*

Output gate



C_{t-1} previous cell state

f_t forget gate output

i_t input gate output

$\tilde{C}_t$ candidate

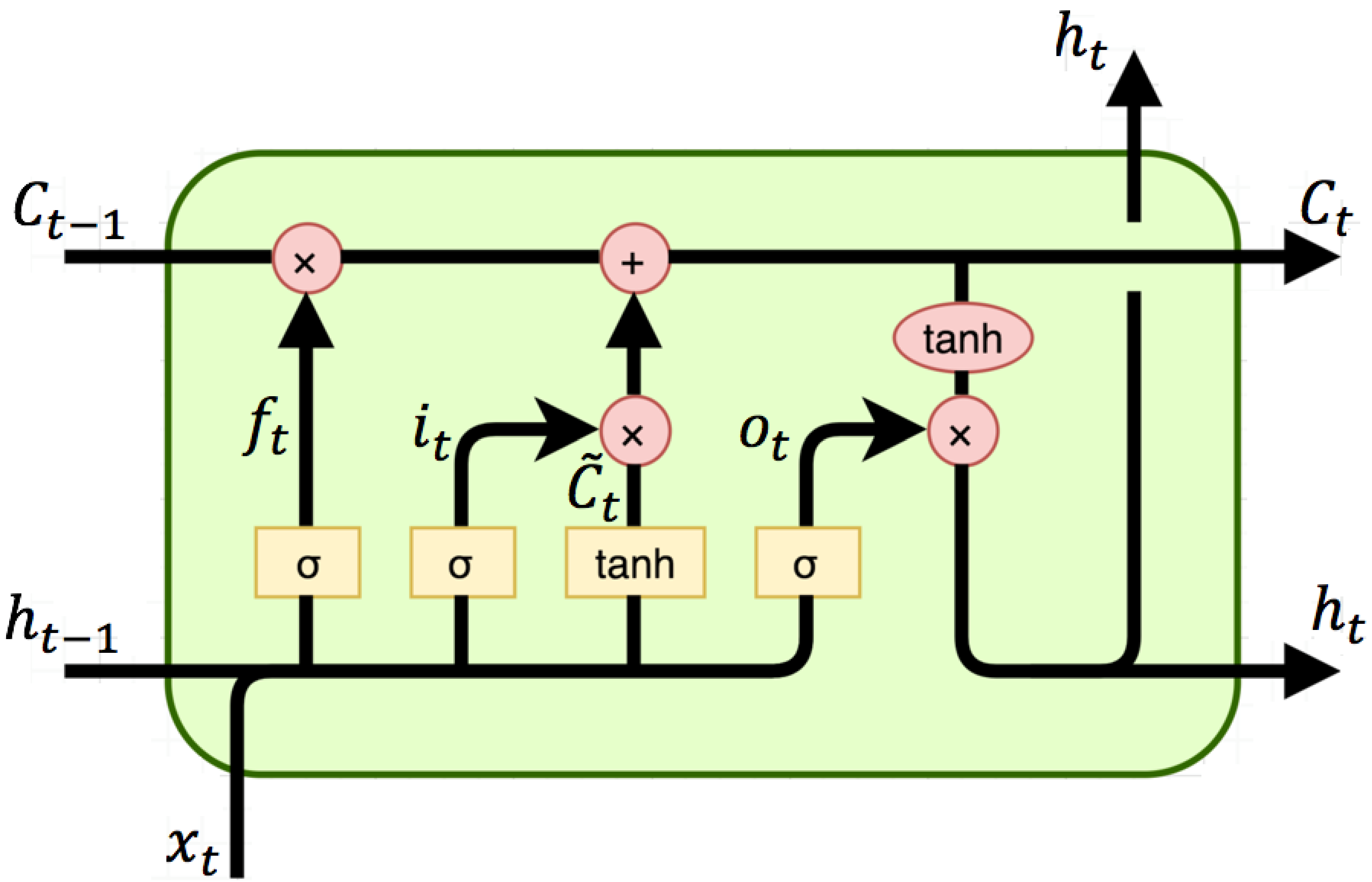
C_t new cell state

o_t output gate output

h_t hidden state

Sepp Hochreiter and Juergen Schmidhuber (1997) "Long Short-term Memory". Neural Computation MIT-Press.

Long short-term *memory (LSTM)*



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

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

$$o_t = \sigma(\omega_o \cdot [h_{t-1}, x_t] + b_o)$$

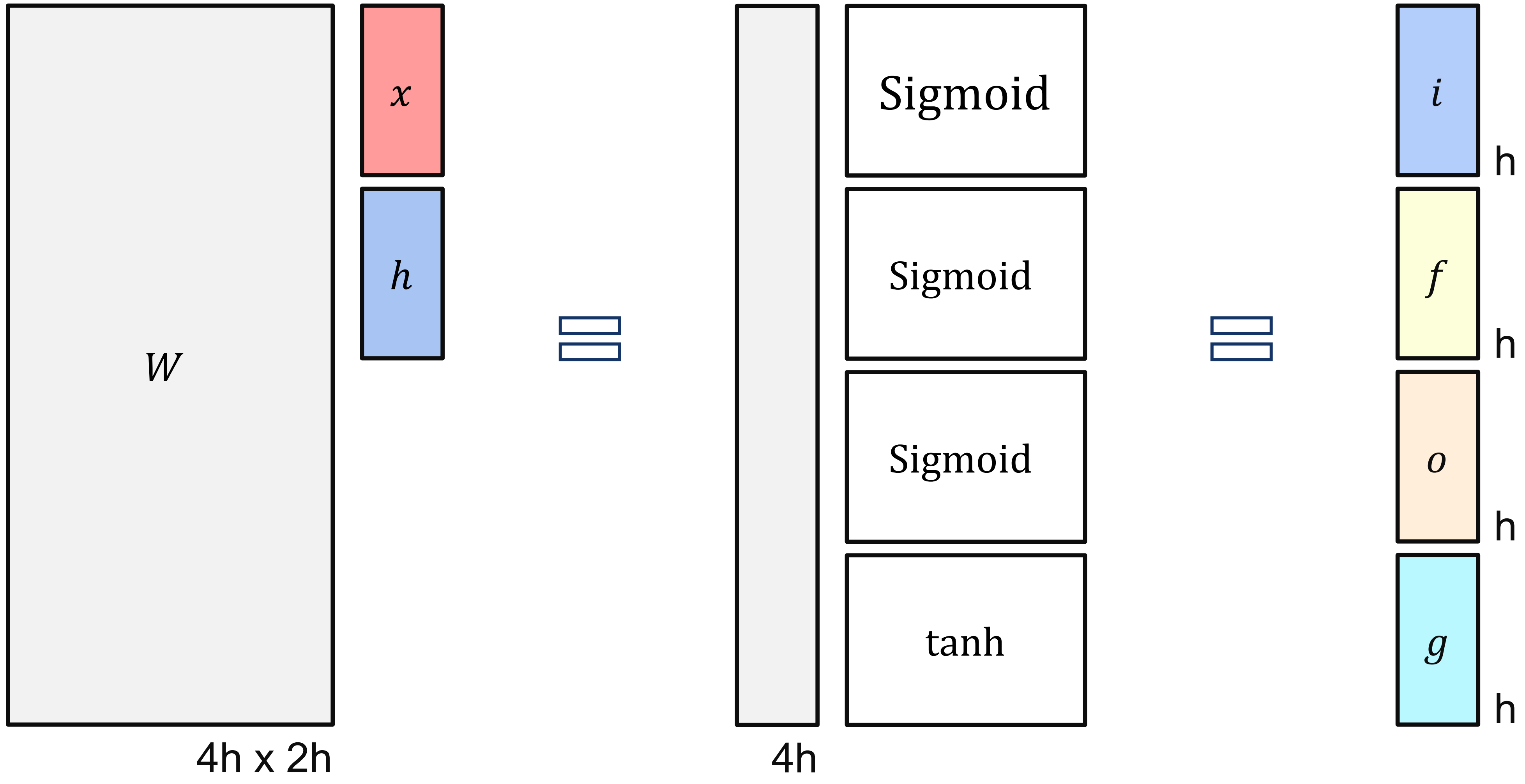
$$\tilde{C}_t = \tanh(\omega_C \cdot [h_{t-1}, x_t] + b_C)$$

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

$$h_t = o_t \odot \tanh(C_t)$$

Sepp Hochreiter and Juergen Schmidhuber (1997) "Long Short-term Memory". Neural Computation MIT-Press.

Long short-term *memory (LSTM)*

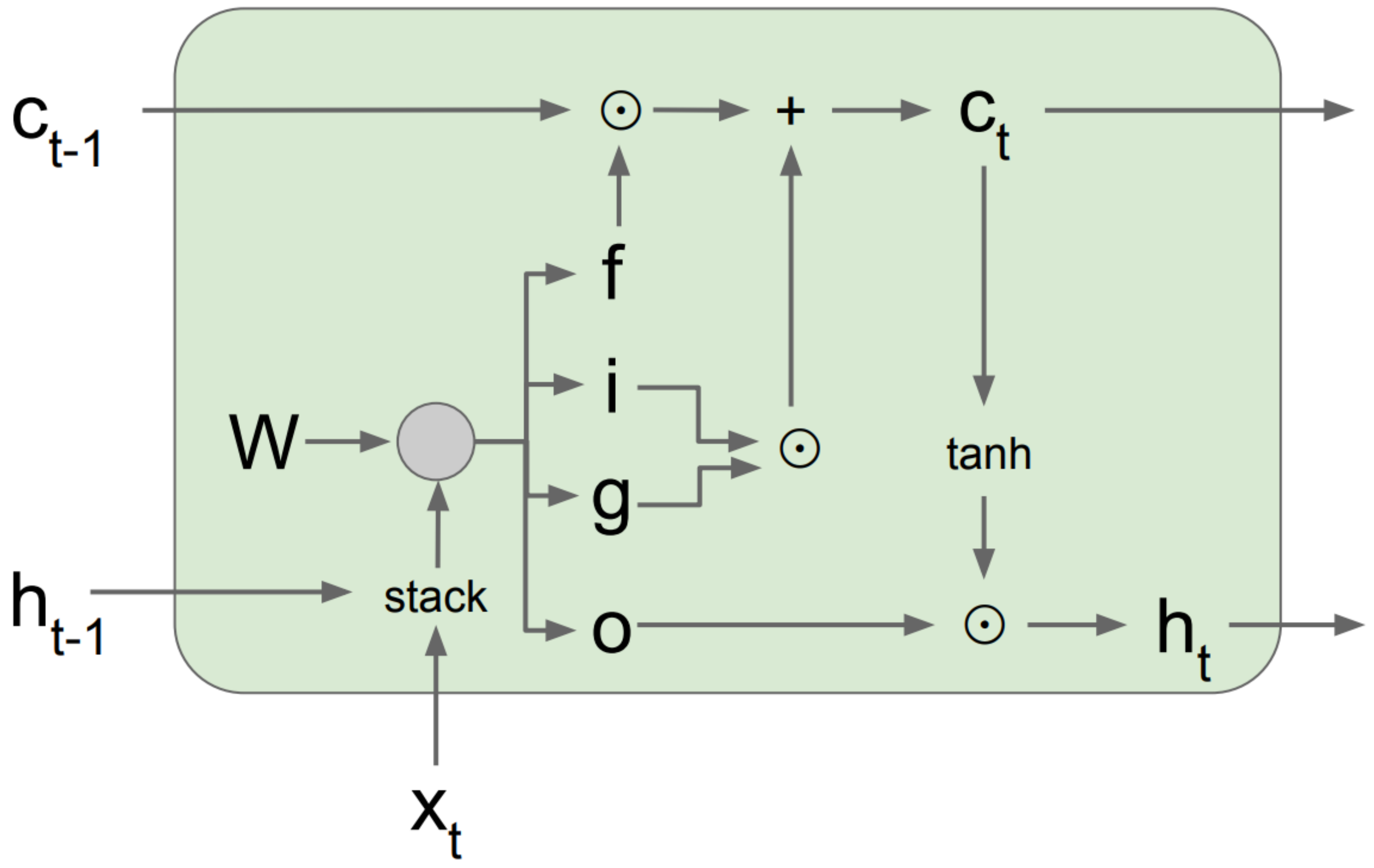


$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

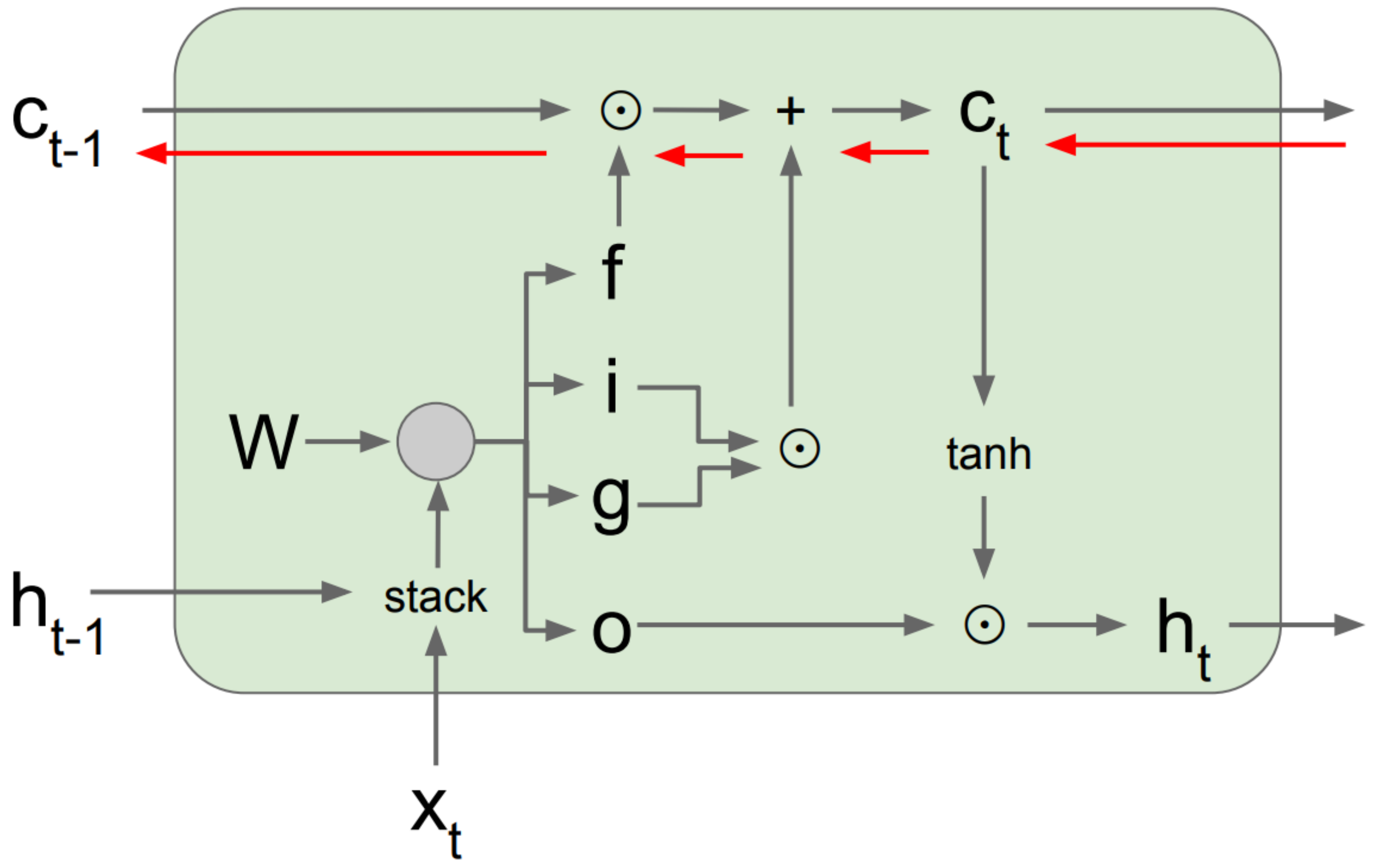
$$C_t = f_t \odot C_{t-1} + i_t \odot g_t$$

$$h_t = o_t \odot \tanh(C_t)$$

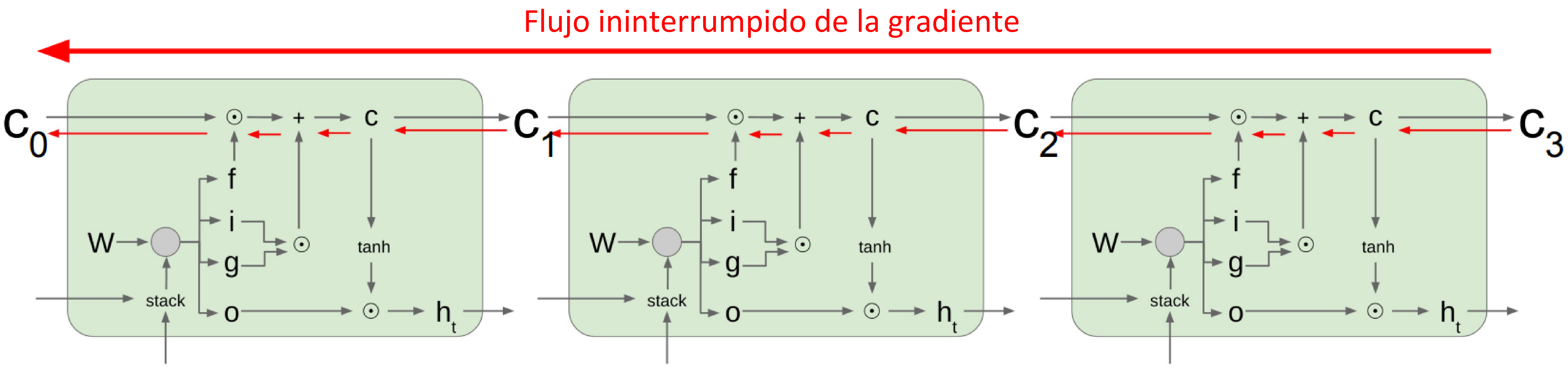
LSTM Gra*dient Flow*



LSTM Gradient Flow

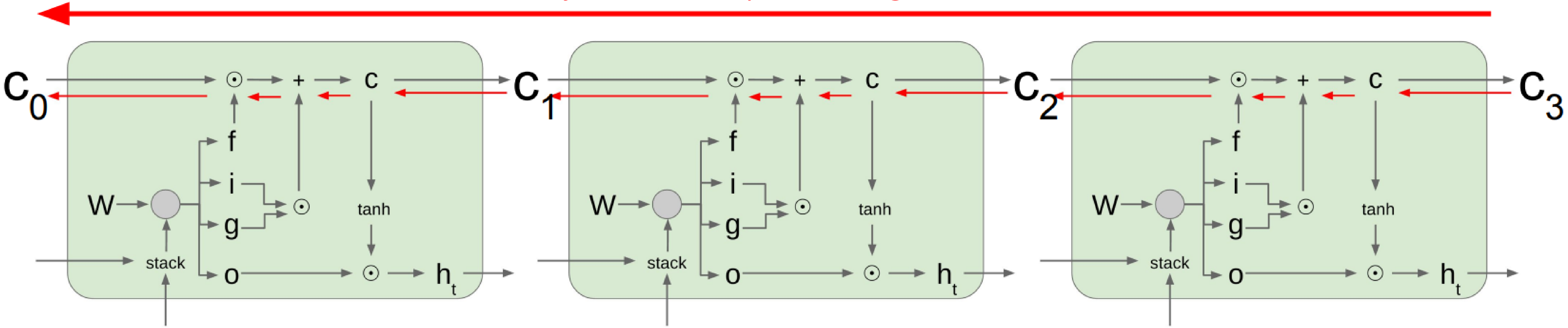


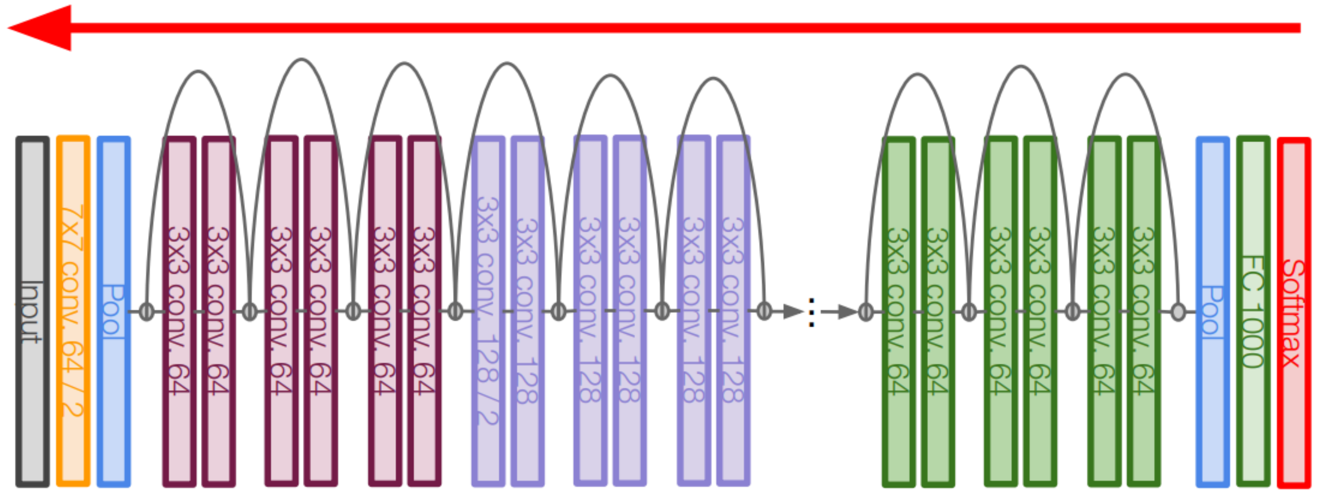
LSTM Gradient Flow



LSTM Gradient Flow

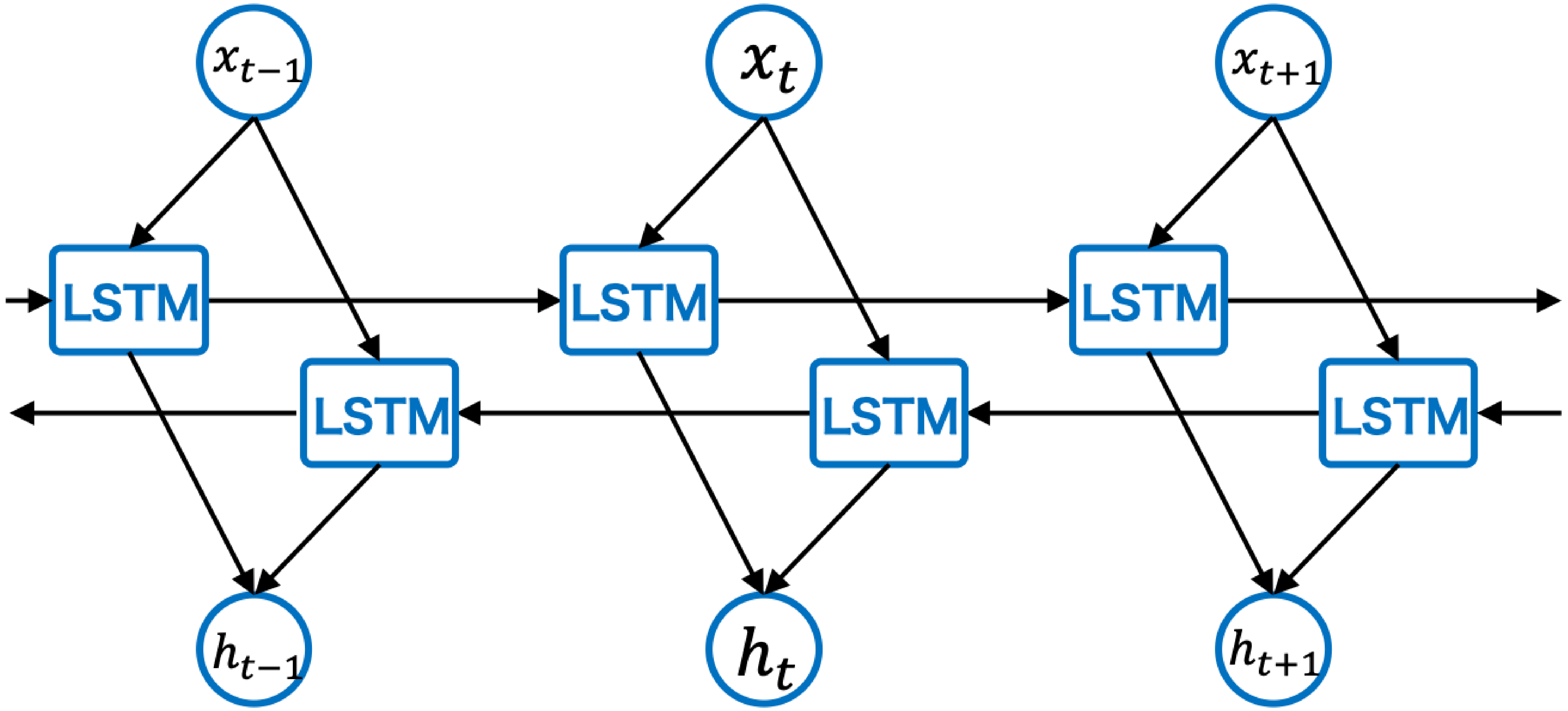
Flujo ininterrumpido de la gradiente





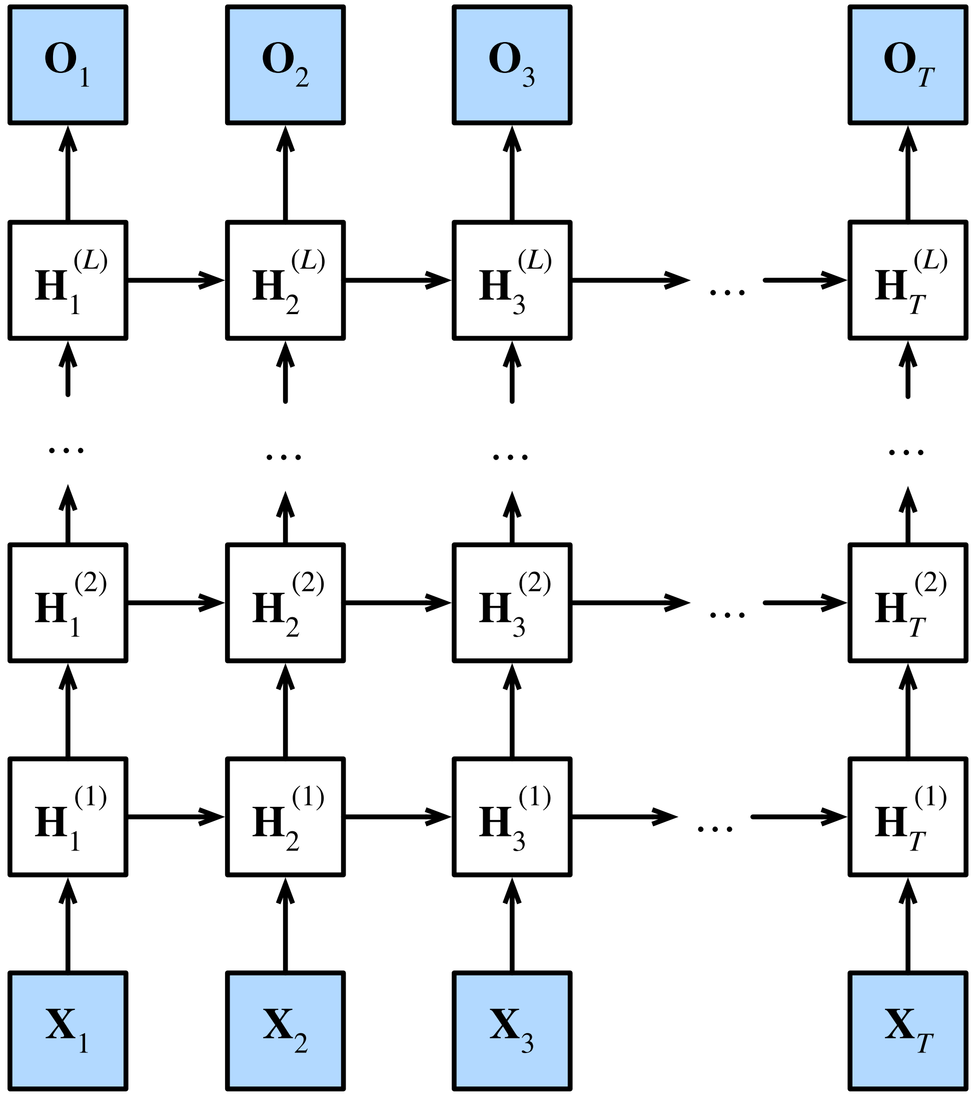
Similar a ResNet

Bidirectional *LSTM*



Alex Graves and Jürgen Schmidhuber (2005) "Framewise phoneme classification with bidirectional LSTM networks". Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005. (Vol. 4, pp. 2047-2052). IEEE.

Deep *LSTM*



$$h_t^{(l)} = \phi_l\left(h_t^{(l-1)} W_{xh}^{(l)} + h_{t-1}^{(l)} W_{hh}^{(l)} + b_h^{(l)}\right)$$

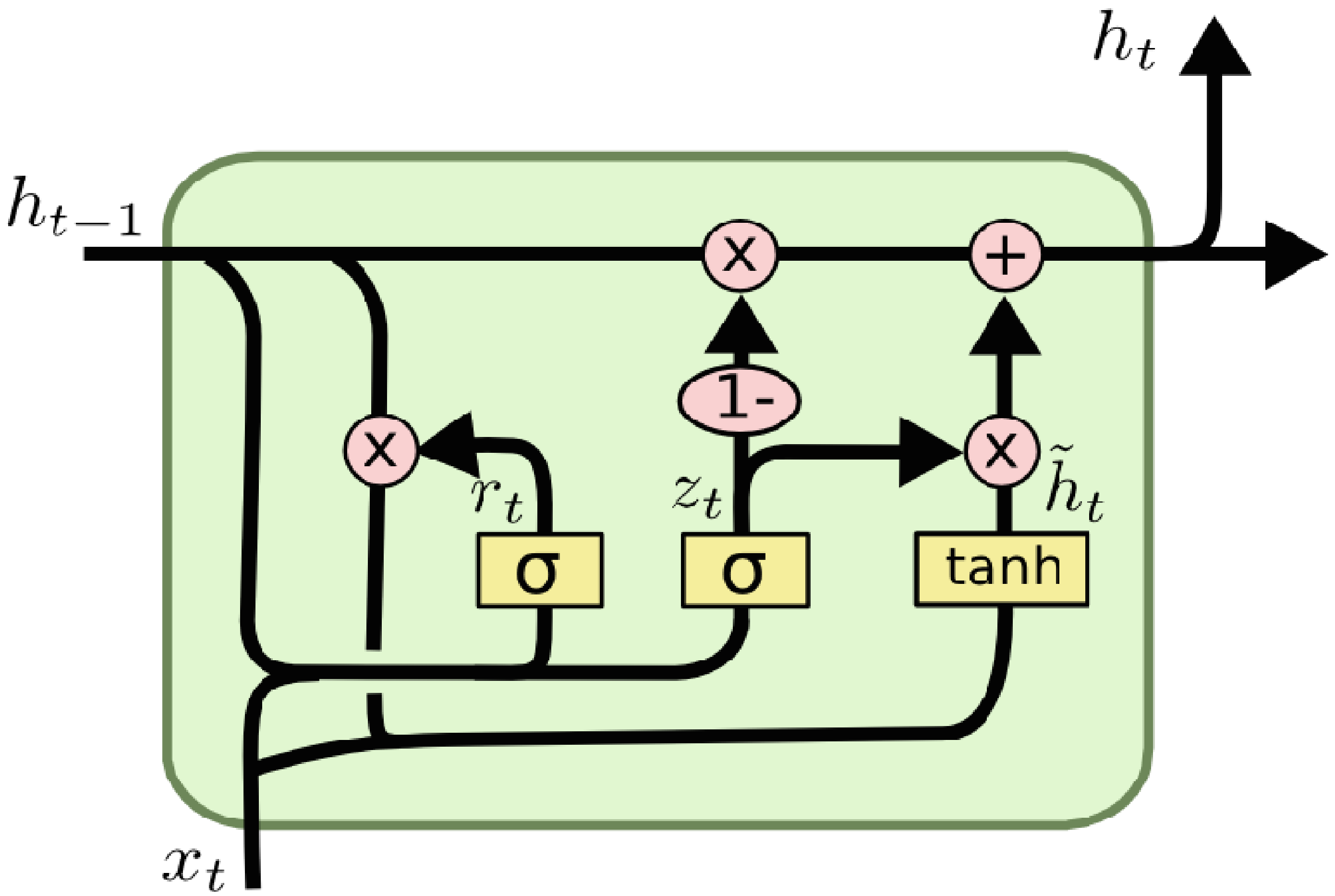
$$O_t = h_t^{(L)} W_{hq} + b_q$$

Felix Weninger, et al. (2014) "Feature enhancement by deep LSTM networks for ASR in reverberant multisource environments". Computer Speech & Language, 28(4), 888-902.

3.

GRU

GRU



Reset gate: $z_t = \sigma(\omega_z \cdot [h_{t-1}, x_t] + b_z)$

Update gate: $r_t = \sigma(\omega_r \cdot [h_{t-1}, x_t] + b_r)$

$$\tilde{h}_t = \tanh(\omega_h \cdot [r_t \odot h_{t-1}, x_t] + b_h)$$

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

Kyunghyun Cho et al. (2014) "On the Properties of Neural Machine Translation: Encoder-Decoder Approaches". arXiv preprint arXiv:1409.1259

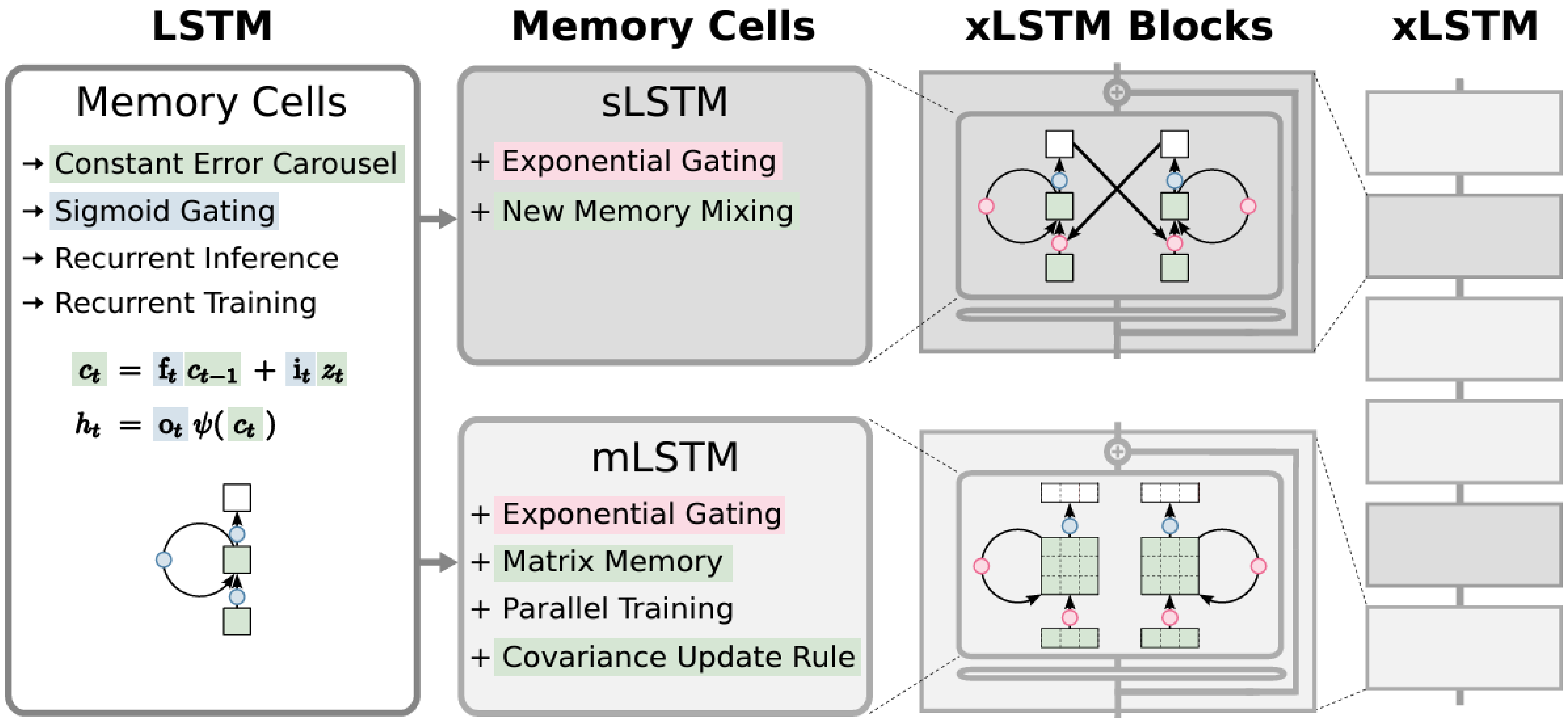
UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

4.

xLSTM

TRANSFORMATEC

xLSTM



Maximilian Beck et al. (2024) "xLSTM: Extended Long Short-Term Memory". arXiv preprint arXiv:2405.04517.

sLSTM

$$c_t = \mathrm{f}_t \; c_{t-1} + \mathrm{i}_t \; z_t \qquad \text{cell state} \qquad (8)$$

$$n_t = \mathrm{f}_t \; n_{t-1} + \mathrm{i}_t \qquad \text{normalizer state} \qquad (9)$$

$$h_t = \mathrm{o}_t \; \tilde{h}_t \,, \qquad \tilde{h}_t = c_t \,/\, n_t \qquad \text{hidden state} \qquad (10)$$

$$z_t = \varphi\left(\tilde{z}_t\right) \,, \qquad \tilde{z}_t = \boldsymbol{w}_z^\top \boldsymbol{x}_t + r_z h_{t-1} + b_z \qquad \text{cell input} \qquad (11)$$

$$\mathrm{i}_t = \exp\left(\tilde{\mathrm{i}}_t\right) \,, \qquad \tilde{\mathrm{i}}_t = \boldsymbol{w}_{\mathrm{i}}^\top \boldsymbol{x}_t + r_{\mathrm{i}} \, h_{t-1} + b_{\mathrm{i}} \qquad \text{input gate} \qquad (12)$$

$$\mathrm{f}_t = \sigma\left(\tilde{\mathrm{f}}_t\right) \text{ OR } \exp\left(\tilde{\mathrm{f}}_t\right) \,, \qquad \tilde{\mathrm{f}}_t = \boldsymbol{w}_{\mathrm{f}}^\top \boldsymbol{x}_t + r_{\mathrm{f}} \, h_{t-1} + b_{\mathrm{f}} \qquad \text{forget gate} \qquad (13)$$

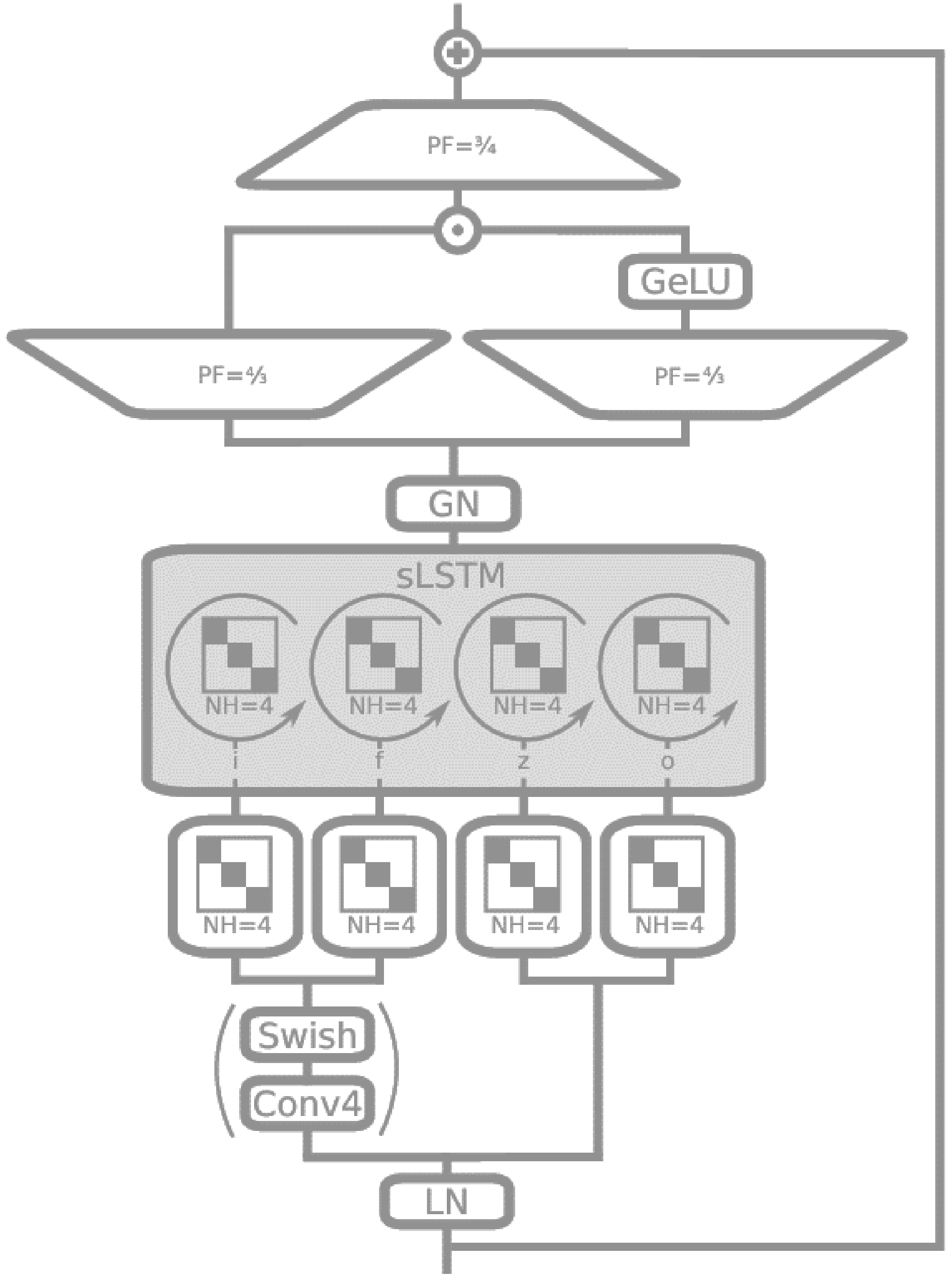
$$\mathrm{o}_t = \sigma\left(\tilde{\mathrm{o}}_t\right) \,, \qquad \tilde{\mathrm{o}}_t = \boldsymbol{w}_{\mathrm{o}}^\top \boldsymbol{x}_t + r_{\mathrm{o}} \, h_{t-1} + b_{\mathrm{o}} \qquad \text{output gate} \qquad (14)$$

Maximilian Beck et al. (2024) "xLSTM: Extended Long Short-Term Memory". arXiv preprint arXiv:2405.04517.

UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

> Reinventa el mundo <

sLSTM



TRANSFORMATEC

Maximilian Beck et al. (2024) "xLSTM: Extended Long Short-Term Memory".
arXiv preprint arXiv:2405.04517.

mLSTM

$$\boldsymbol{C}_t = \mathrm{f}_t \ \boldsymbol{C}_{t-1} + \mathrm{i}_t \ \boldsymbol{v}_t \ \boldsymbol{k}_t^\top \qquad \text{cell state} \quad (19)$$

$$\boldsymbol{n}_t = \mathrm{f}_t \ \boldsymbol{n}_{t-1} + \mathrm{i}_t \ \boldsymbol{k}_t \qquad \text{normalizer state} \quad (20)$$

$$\boldsymbol{h}_t = \mathbf{o}_t \odot \tilde{\boldsymbol{h}}_t , \qquad \tilde{\boldsymbol{h}}_t = \boldsymbol{C}_t \ \boldsymbol{q}_t \ / \ \max\left\{ \left| \boldsymbol{n}_t^\top \ \boldsymbol{q}_t \right|, 1 \right\} \qquad \text{hidden state} \quad (21)$$

$$\boldsymbol{q}_t = \boldsymbol{W}_q \ \boldsymbol{x}_t + \boldsymbol{b}_q \qquad \text{query input} \quad (22)$$

$$\boldsymbol{k}_t = \frac{1}{\sqrt{d}} \boldsymbol{W}_k \ \boldsymbol{x}_t + \boldsymbol{b}_k \qquad \text{key input} \quad (23)$$

$$\boldsymbol{v}_t = \boldsymbol{W}_v \ \boldsymbol{x}_t + \boldsymbol{b}_v \qquad \text{value input} \quad (24)$$

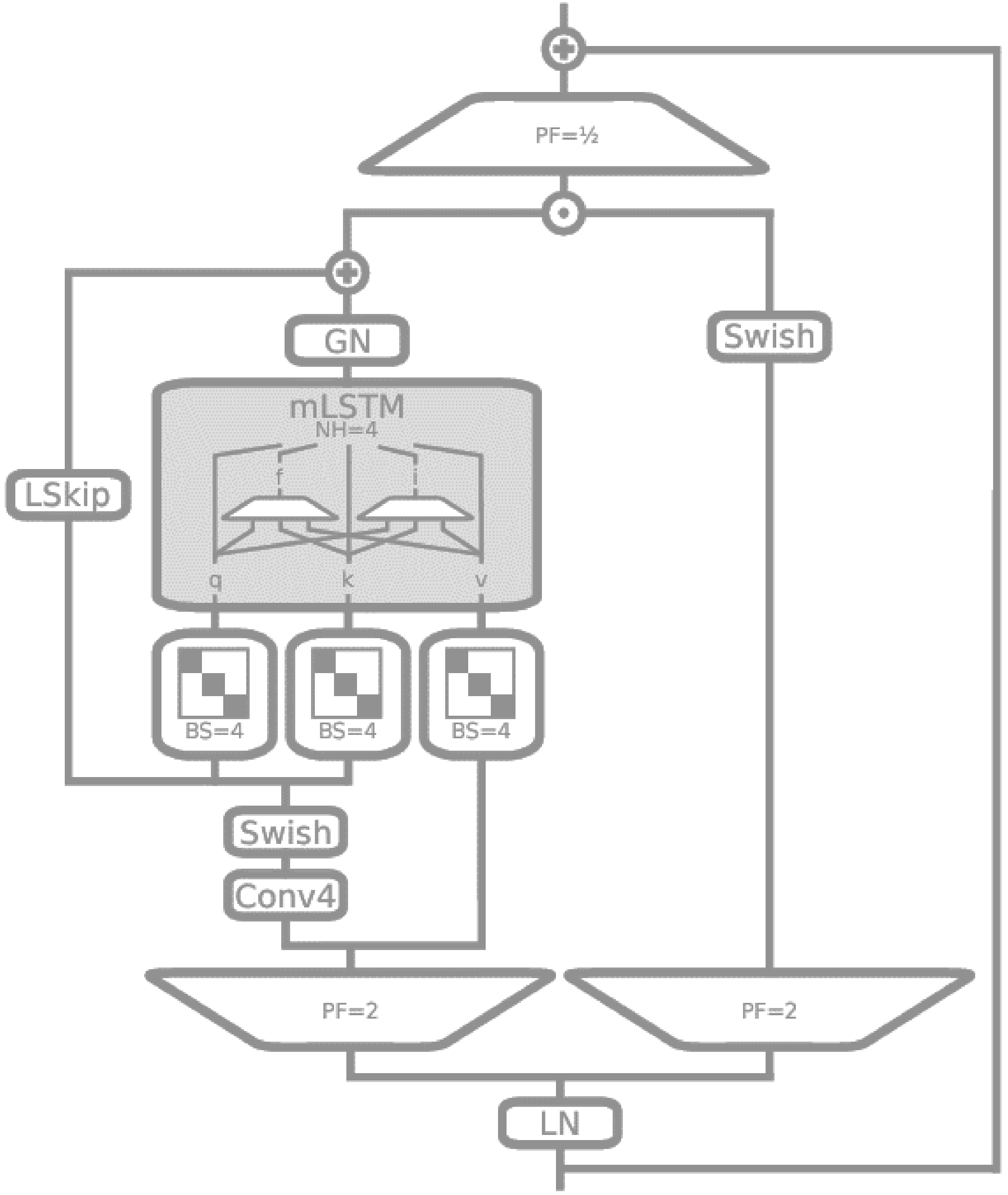
$$\mathrm{i}_t = \exp\left(\tilde{\mathrm{i}}_t\right) , \qquad \tilde{\mathrm{i}}_t = \boldsymbol{w}_{\mathrm{i}}^\top \ \boldsymbol{x}_t + b_{\mathrm{i}} \qquad \text{input gate} \quad (25)$$

$$\mathrm{f}_t = \sigma\left(\tilde{\mathrm{f}}_t\right) \text{ OR } \exp\left(\tilde{\mathrm{f}}_t\right) , \qquad \tilde{\mathrm{f}}_t = \boldsymbol{w}_{\mathrm{f}}^\top \ \boldsymbol{x}_t + b_{\mathrm{f}} \qquad \text{forget gate} \quad (26)$$

$$\mathbf{o}_t = \sigma\left(\tilde{\mathbf{o}}_t\right) , \qquad \tilde{\mathbf{o}}_t = \boldsymbol{W}_{\mathbf{o}} \ \boldsymbol{x}_t + \boldsymbol{b}_{\mathbf{o}} \qquad \text{output gate} \quad (27)$$

Maximilian Beck et al. (2024) "xLSTM: Extended Long Short-Term Memory". arXiv preprint arXiv:2405.04517.

mLSTM

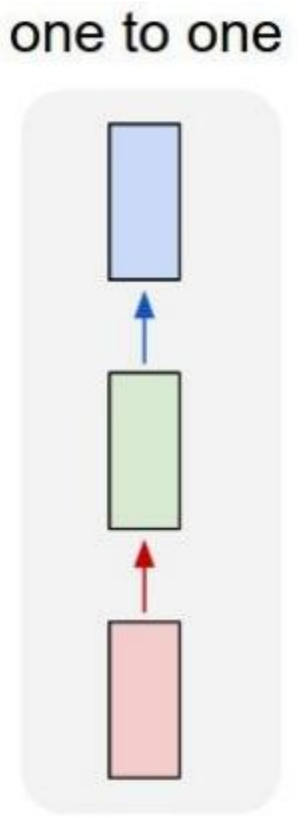


Maximilian Beck et al. (2024) "xLSTM: Extended Long Short-Term Memory". arXiv preprint arXiv:2405.04517.

5.

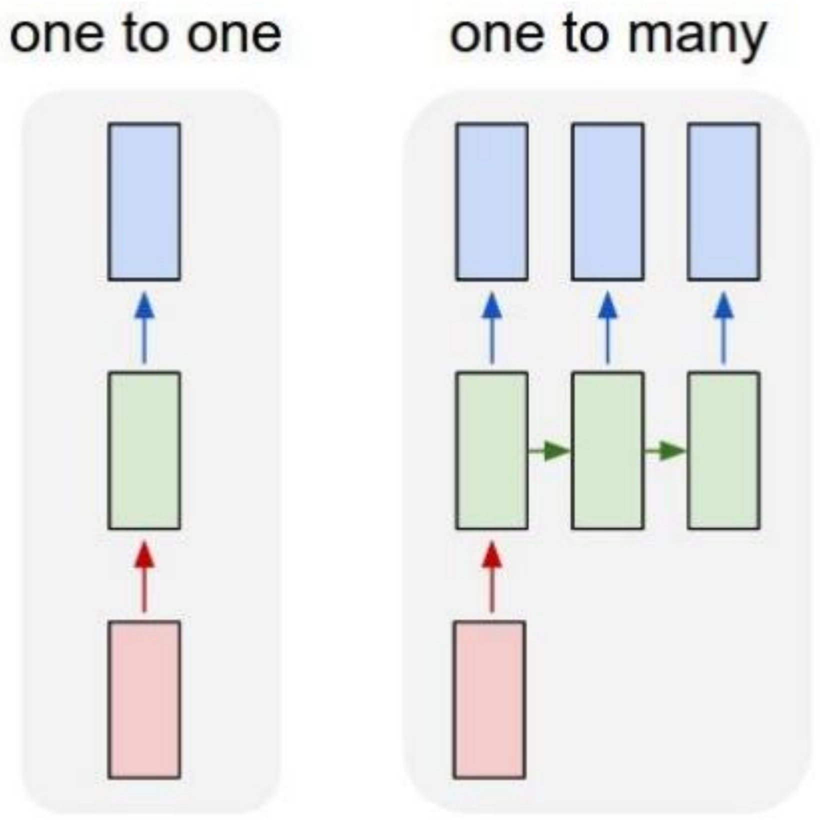
Configuraciones

Configuraciones



Ejemplo:
Clasificación
Regresión

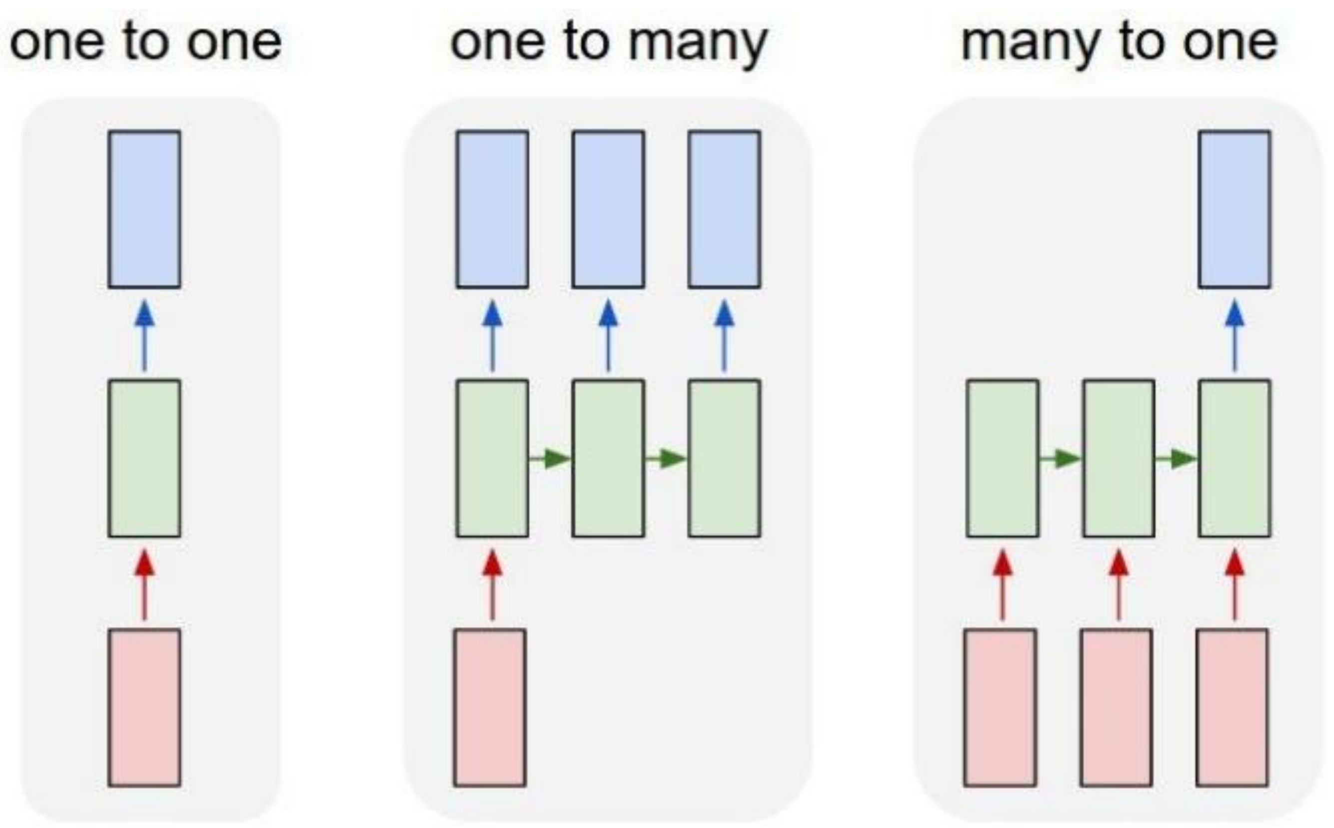
Configu*raciones*



Ejemplo:
Clasificación
Regresión

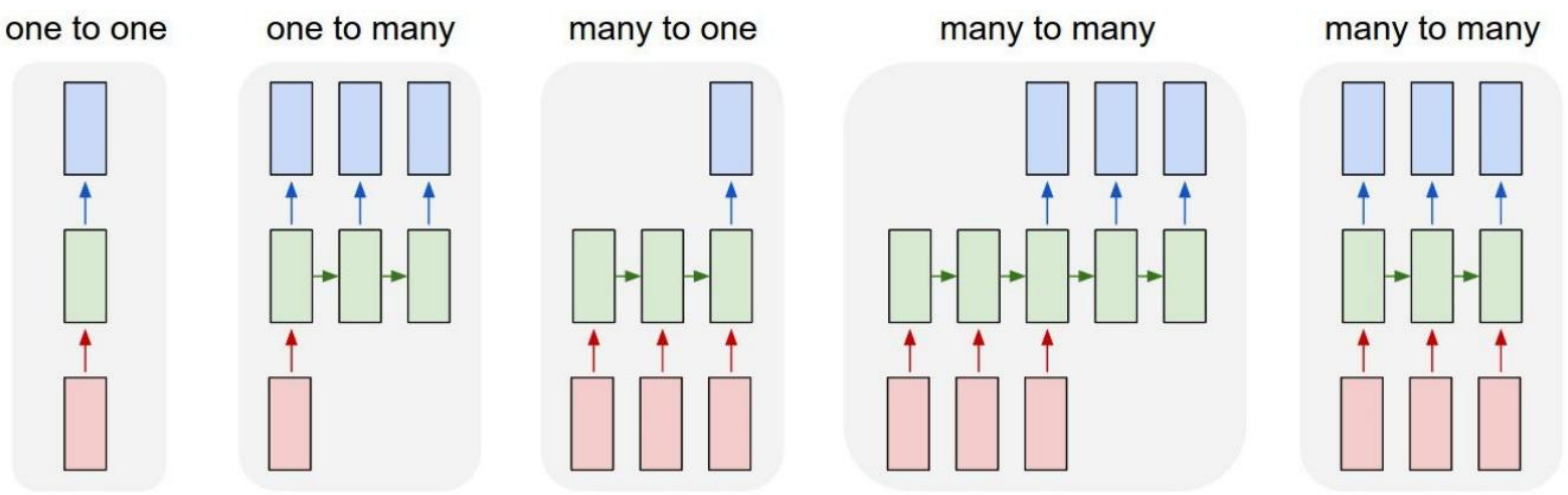
Ejemplo:
Image caption
Music generation

Configuraciones



Ejemplo: Clasificación Regresión	**Ejemplo:** Image caption Music generation	**Ejemplo:** clasificación de oraciones, respuesta a preguntas de opción múltiple

Configuraciones



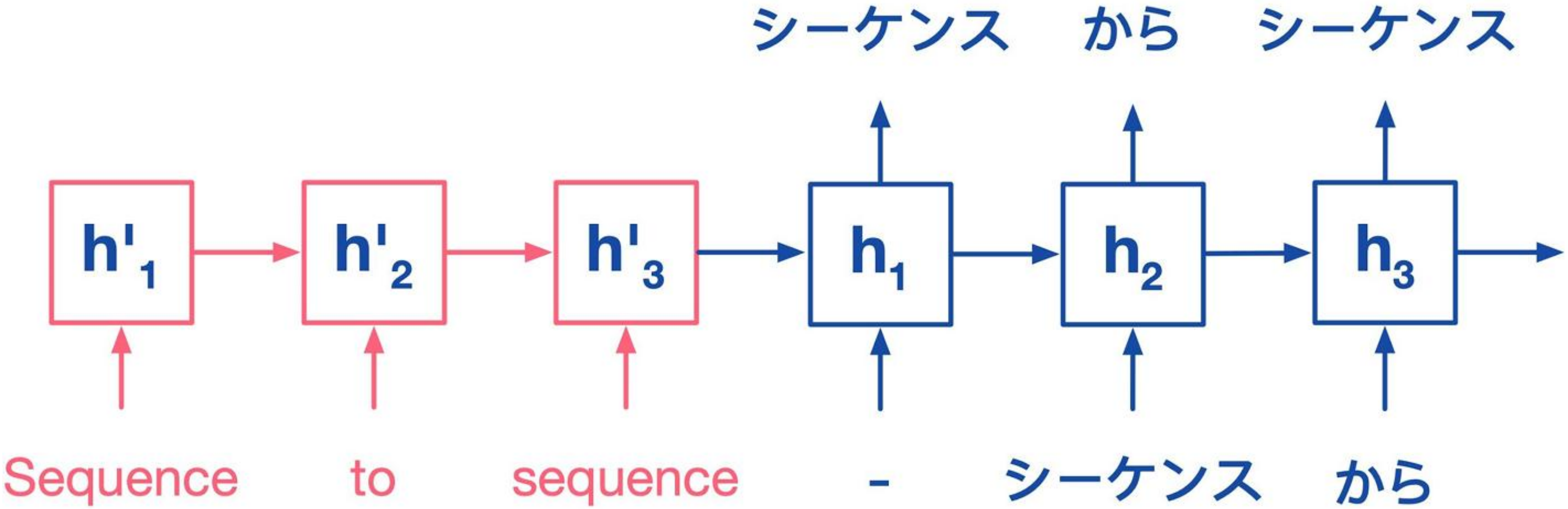
Ejemplo:
Clasificación
Regresión

Ejemplo:
Image caption
Music generation

Ejemplo:
clasificación de oraciones, respuesta a preguntas de opción múltiple

Ejemplo:
machine translation, video classification, video captioning,

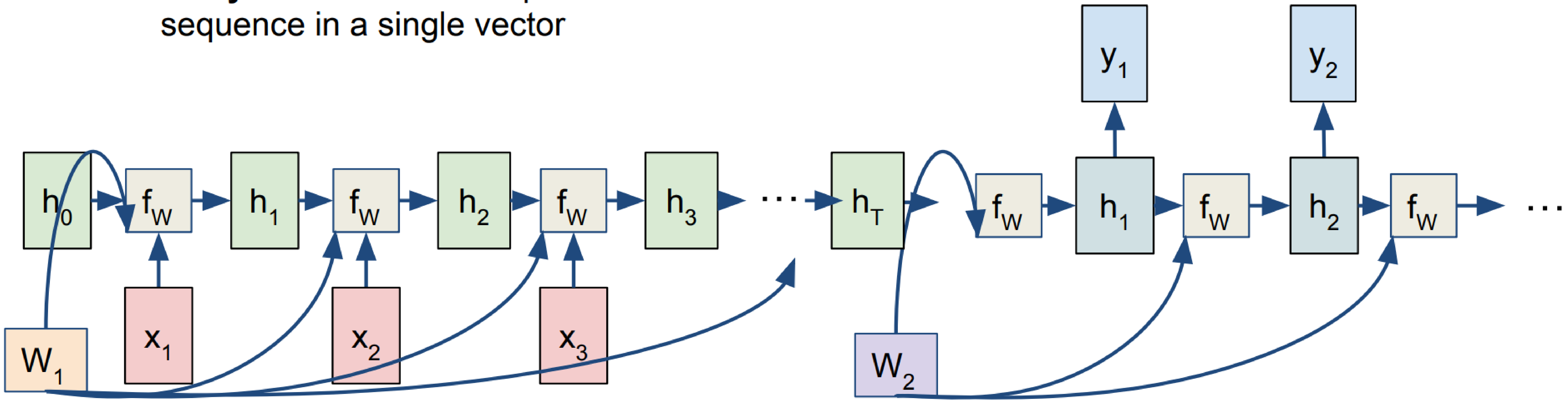
Sequence-to-*sequence models*



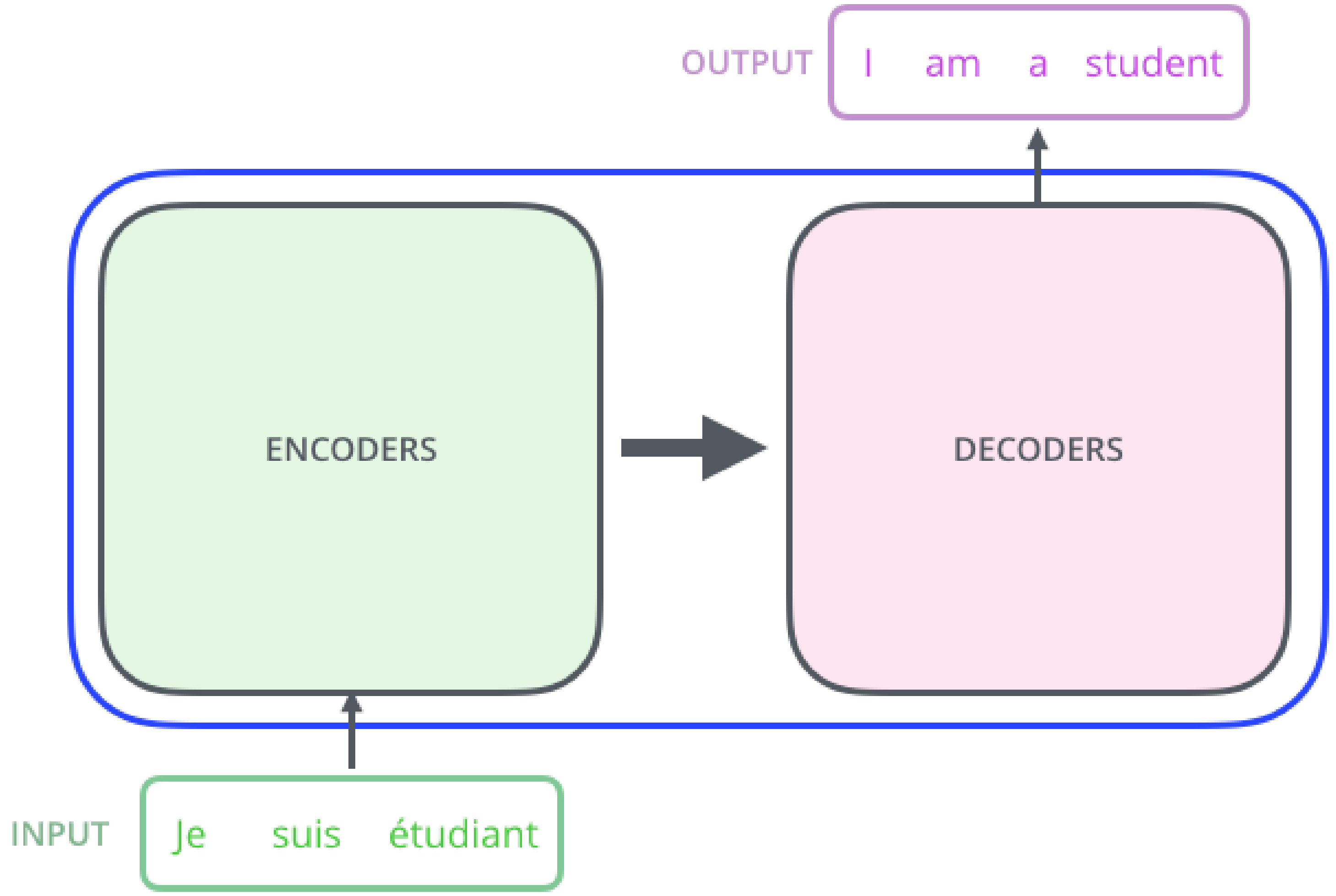
Sequence-to-*sequence models*

One to many: Produce output sequence from single input vector

Many to one: Encode input sequence in a single vector

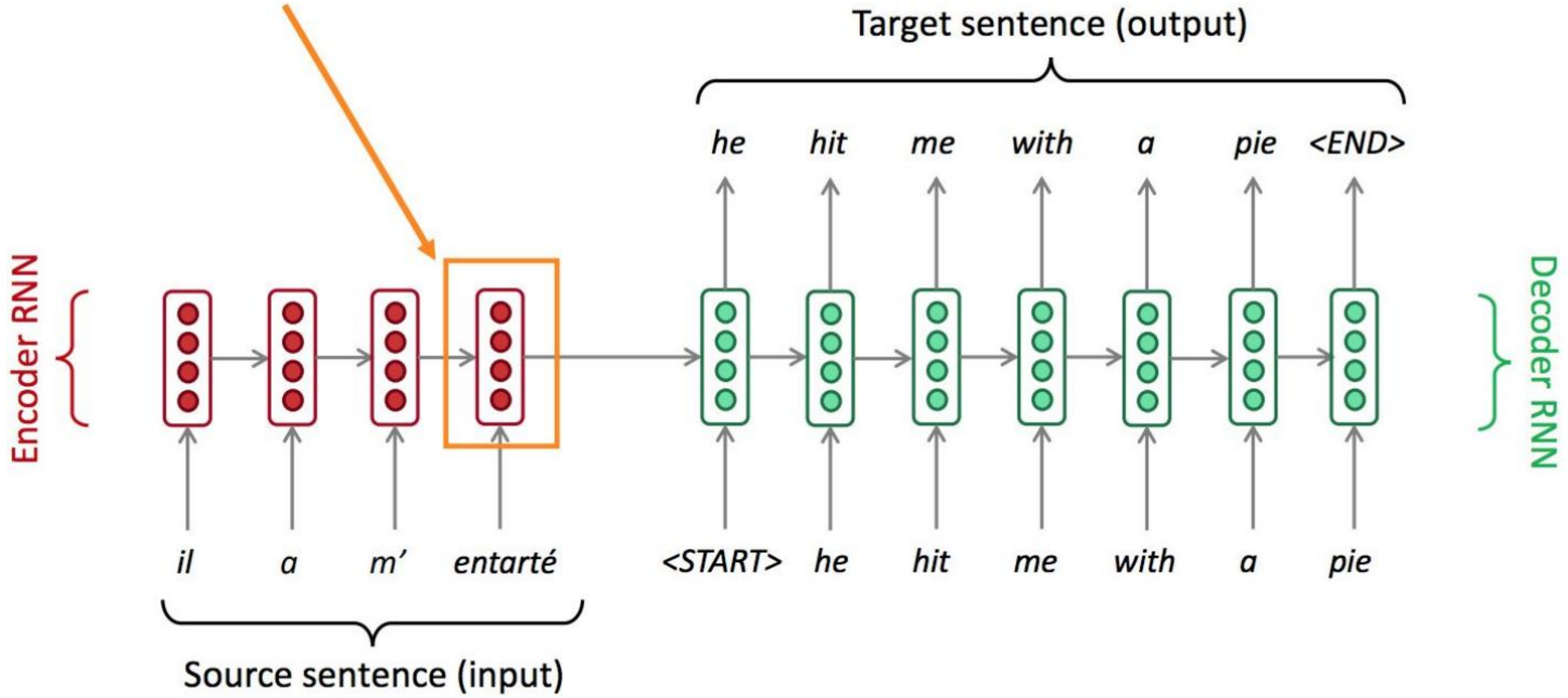


Sequence-to-*sequence models*



Sequence-to-*sequence models*

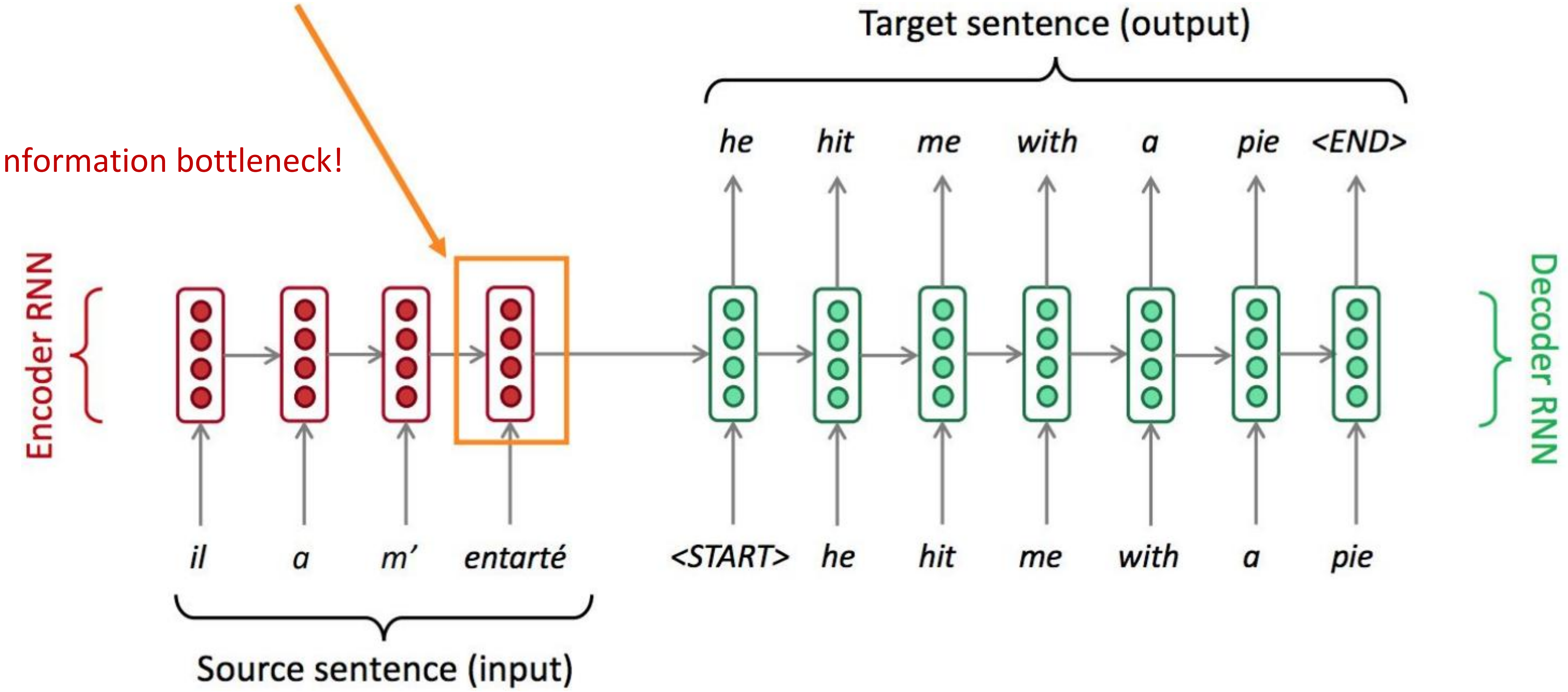
Codificación de la oración fuente.



Sequence-to-*sequence models*

Codificación de la oración fuente.
Debe capturar toda la información
sobre la oración fuente.

Information bottleneck!



UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

> Reinventa el mundo <

GRACIAS

Victor Flores Benites

TRANSFORMATEC