RECURRENT NEURAL NETWORK
UTEC

Cristian López Del Alamo

2021

Agenda

1. Classification
2. Neural Network
3. CNN
4. Sequential Models
5. Ejemplo en Pytorch

Objetivo:

1. Entender el funcionamiento de las redes neuronales recurrentes.
2. Como se usan y para qué sirven

1 Introduction

Classification

Source [: Click]

Wavelets, Harris, SIFT, etc

Image

0.1	0.8	1.3	.3	3.1	4.2

Feature Vector

0.1	0.8	1.3	.3	3.1	4.2

v1: Feature Vector

d(v1, v2) < e

0.2	0.9	1.1	.2	3.5	4.0

v2: Feature Vector

UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

0.1	0.8	1.3	.3	3.1	4.2

v1: Feature Vector

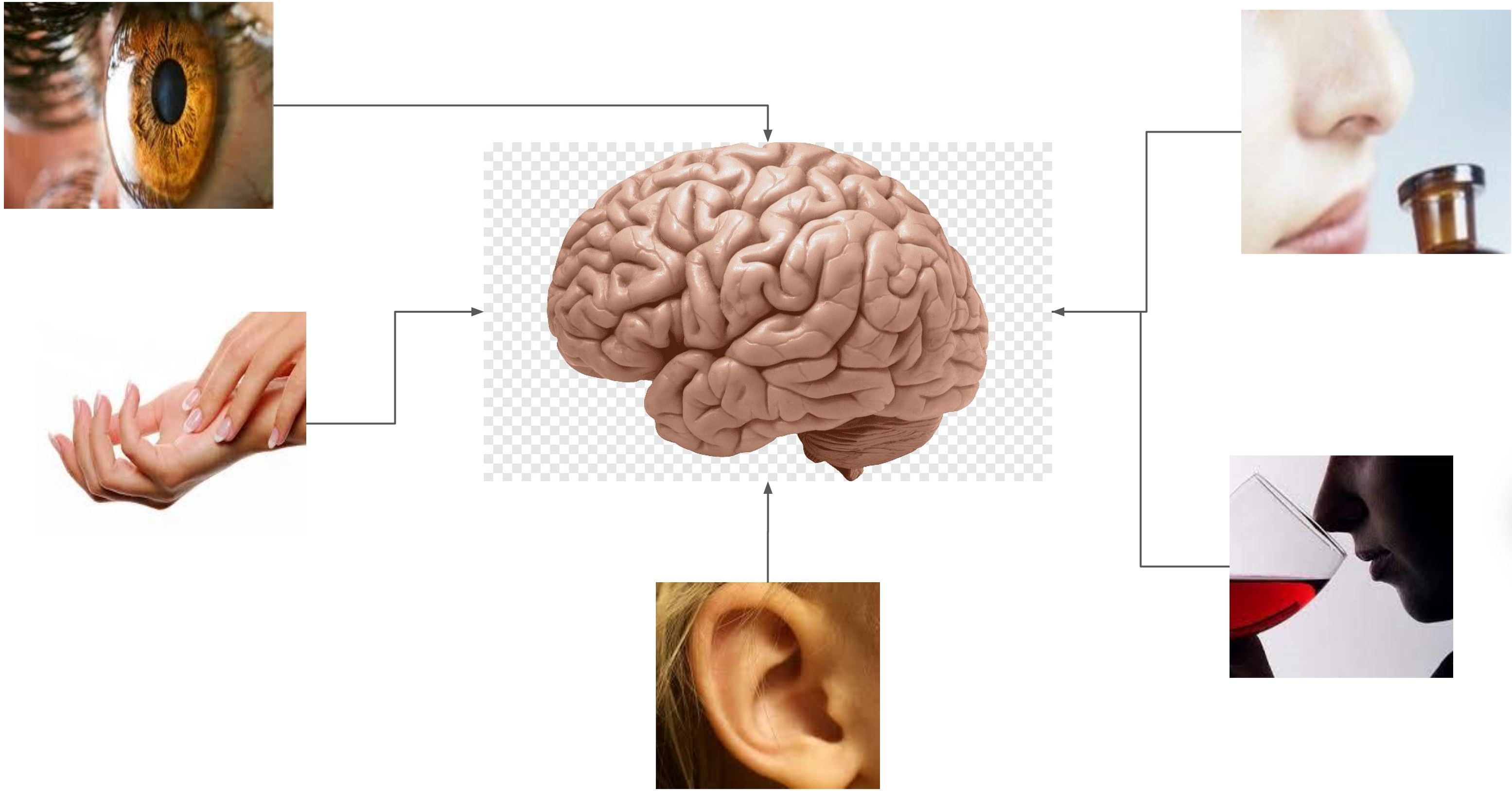
$d(v1, v2) > e$

2.2	3.9	1.6	2.2	6.5	1.0

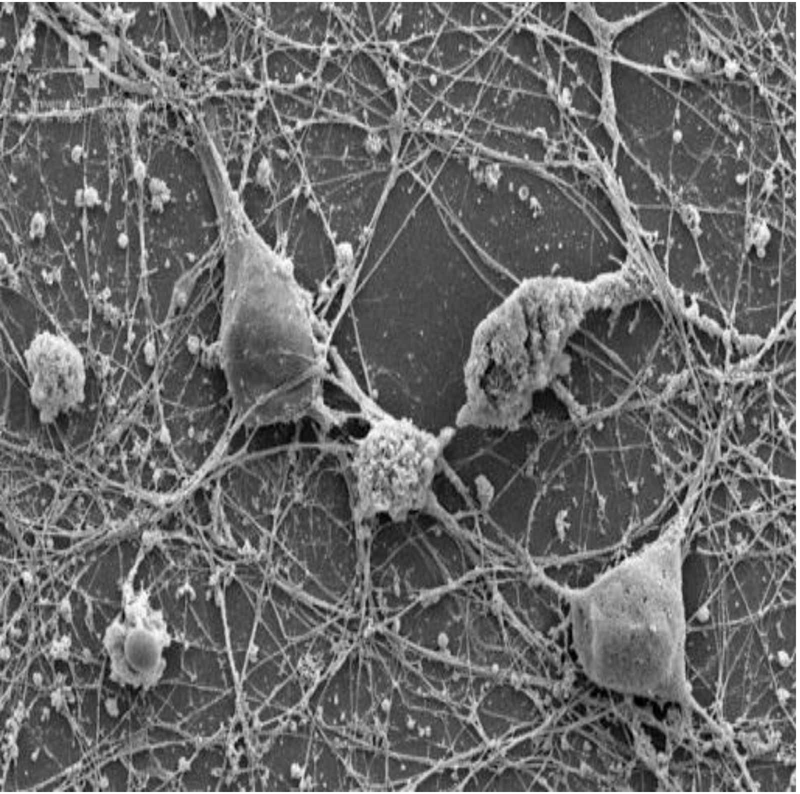
v2: Feature Vector

2 Neural Network

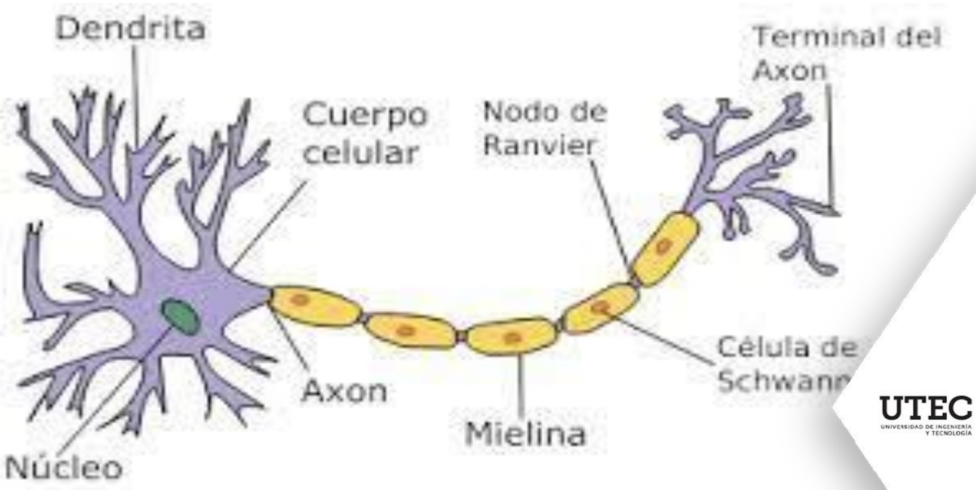
Neural Network



Neural Network



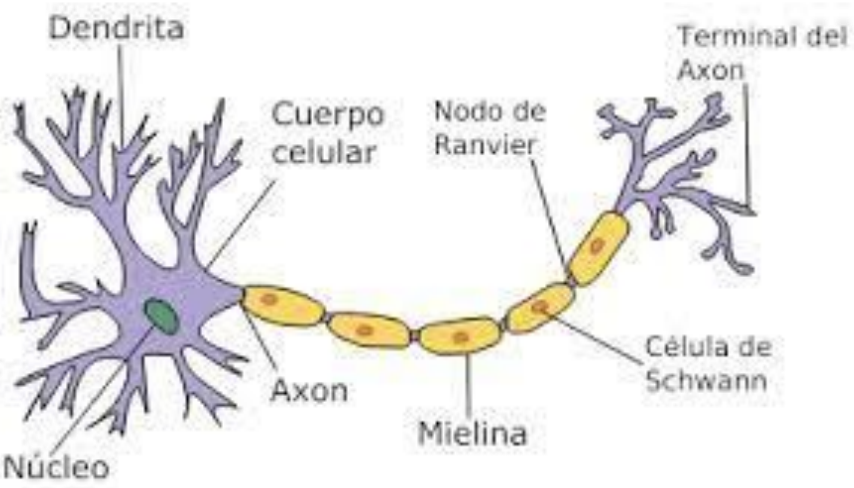
Source: Click



Source: Click

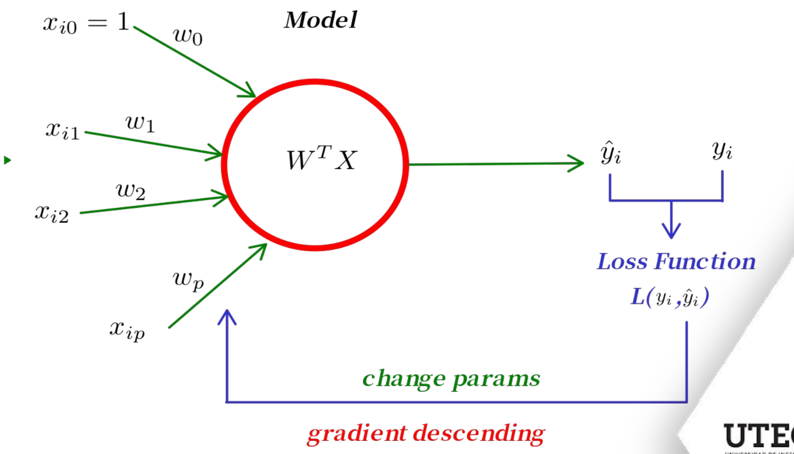
Neural Network

Biological Neuron



Source: Click

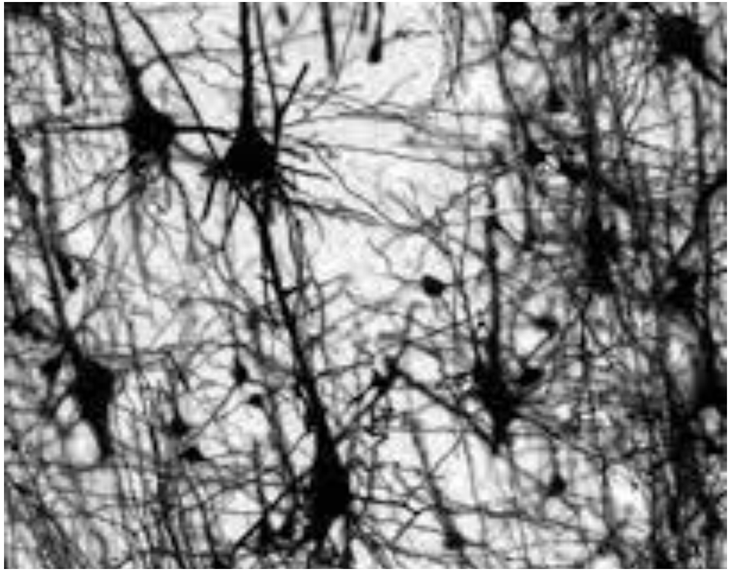
Artificial Neuron



Source: Click

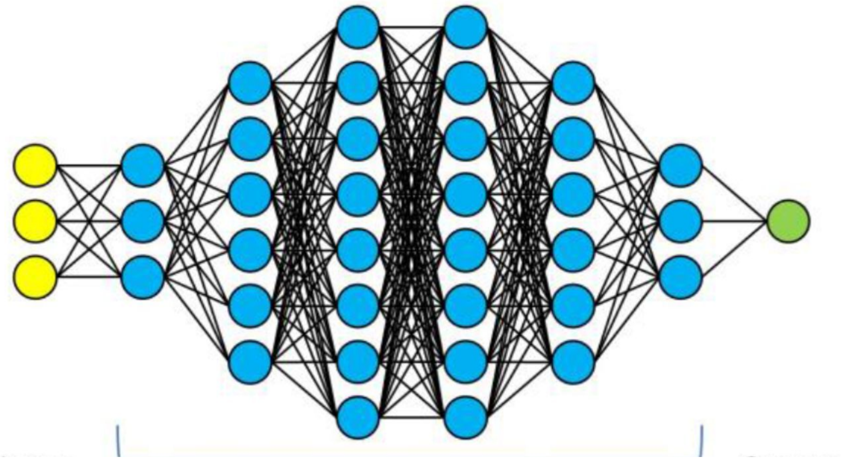
Neural Network

Biological Neural Network



Source: [Click]

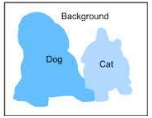
Artificial Neural Network



Source: [Click]

Neural Network

f() = “Dog”

f() = 

f() = 

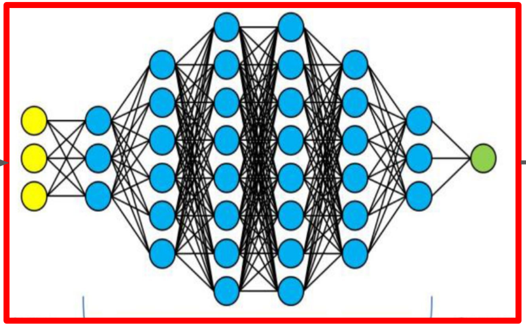
f() = 

f() = 

f() = “Pelican”

Neural Network

x

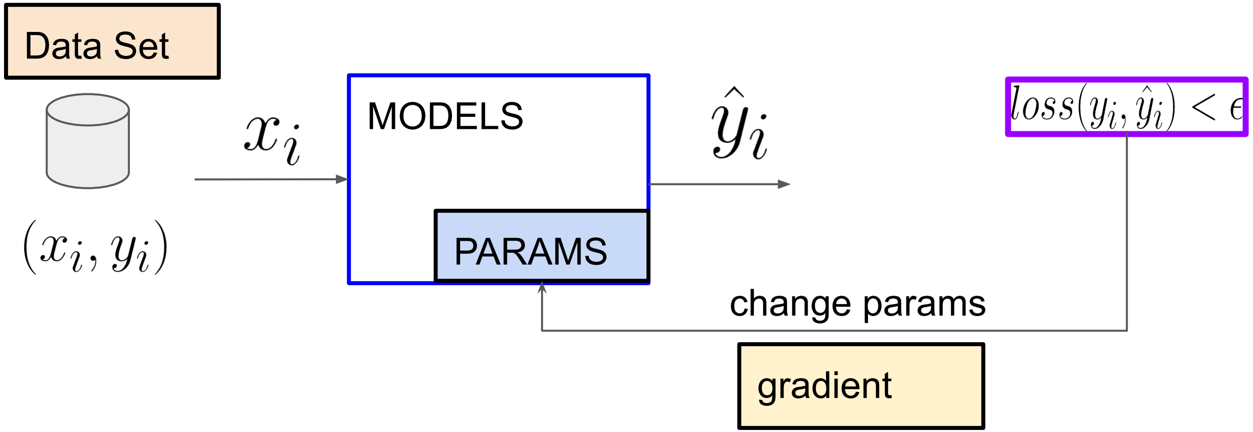


y

f = Model

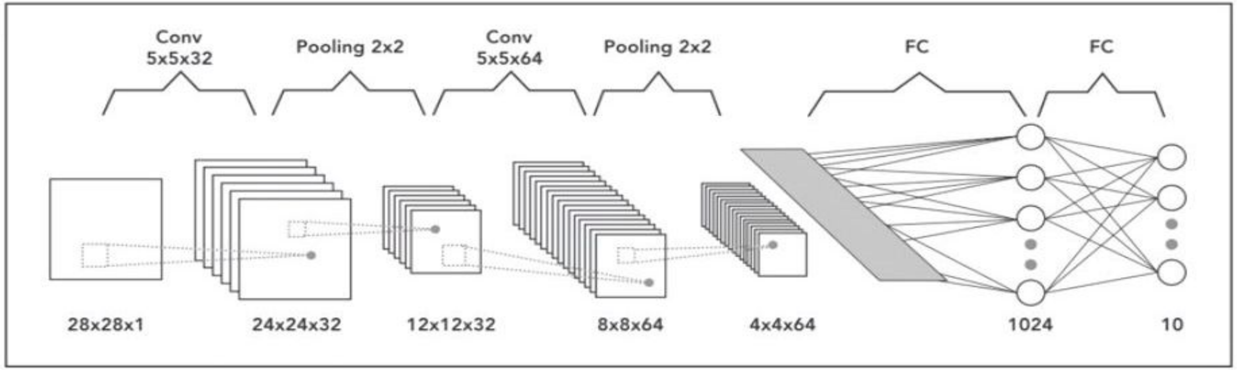
$$f(x) = y$$

Neural Network



3 CNN

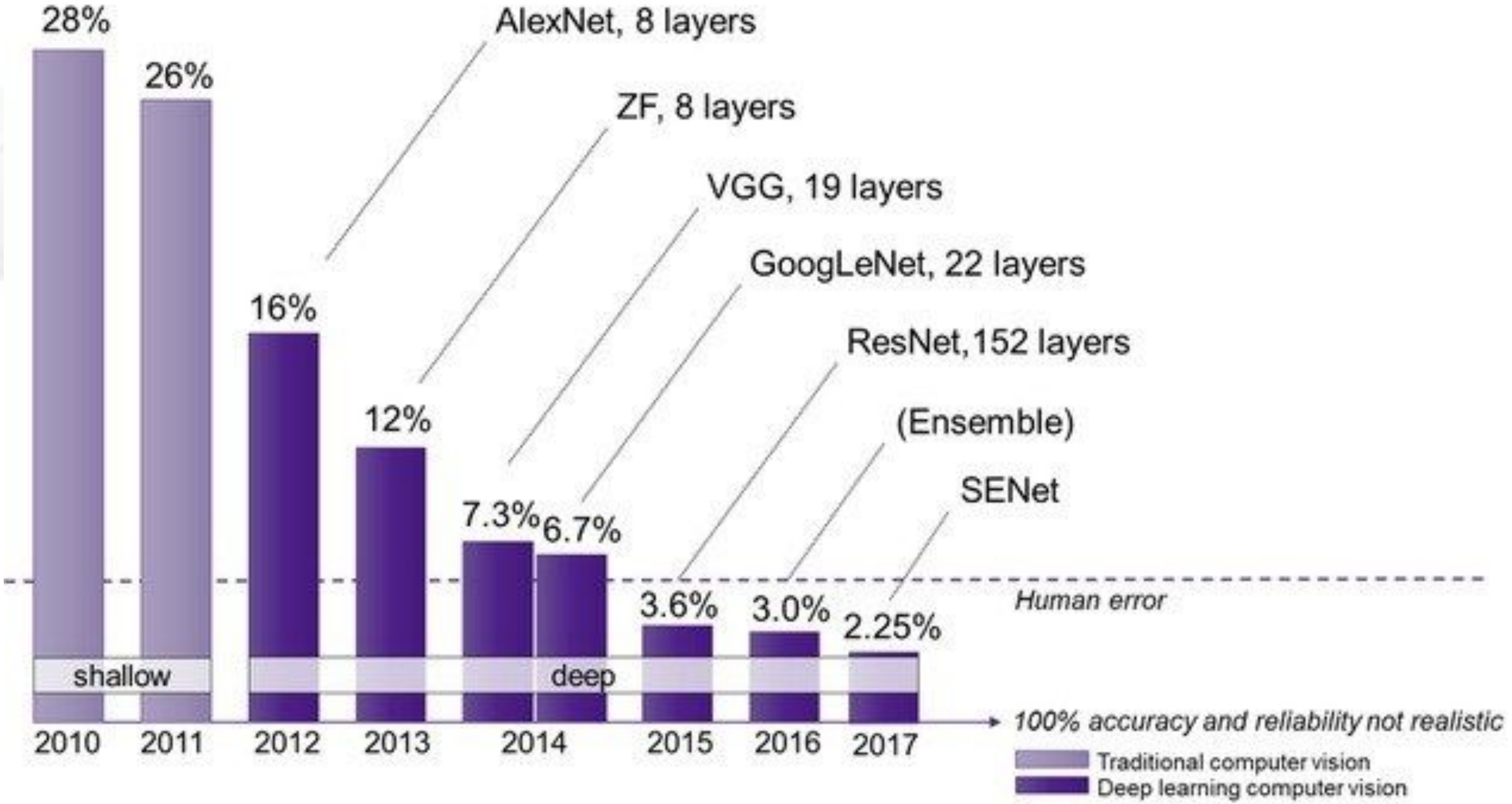
INTRODUCTION



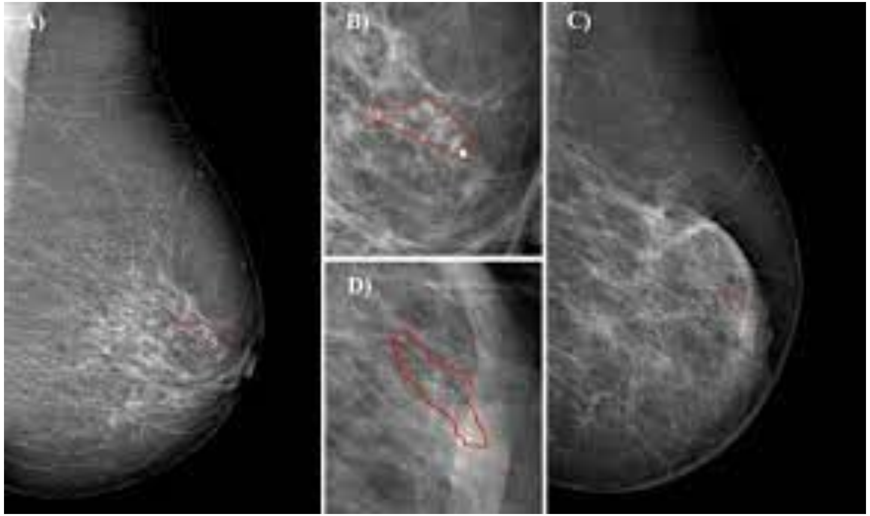
CNN: Neural Networks specialized in images

$f($  $) = \hat{y} \quad |y - \hat{y}|_p^p < \epsilon$

INTRODUCTION



ImageNet Large Scale Visual Recognition Challenge results show that deep learning is surpassing human levels of accuracy.
Source: click



[Clasificación de lesiones de mama con deep learning]

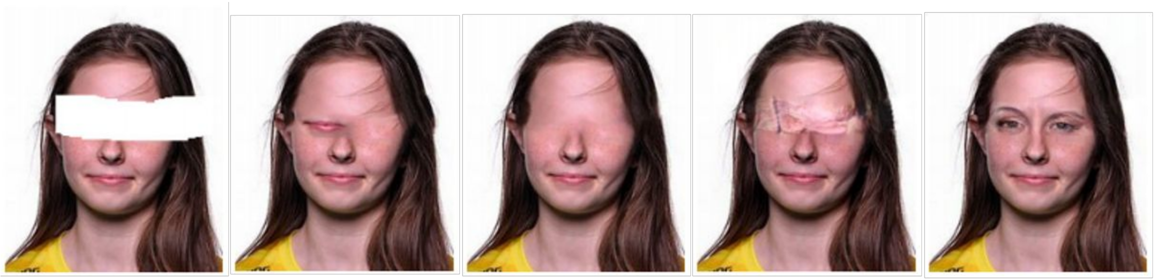
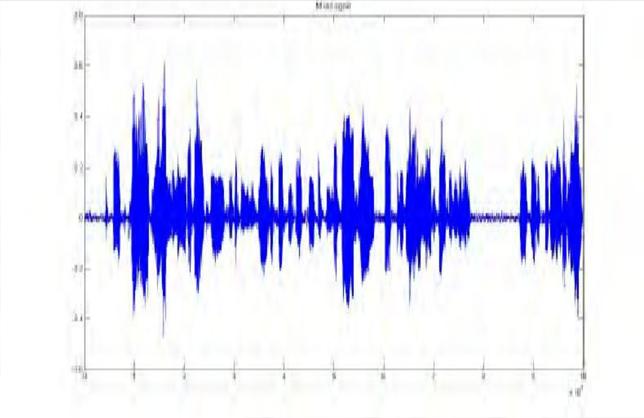


Image completion: Impating

4 Sequential Model

Time Series



Audio Signal

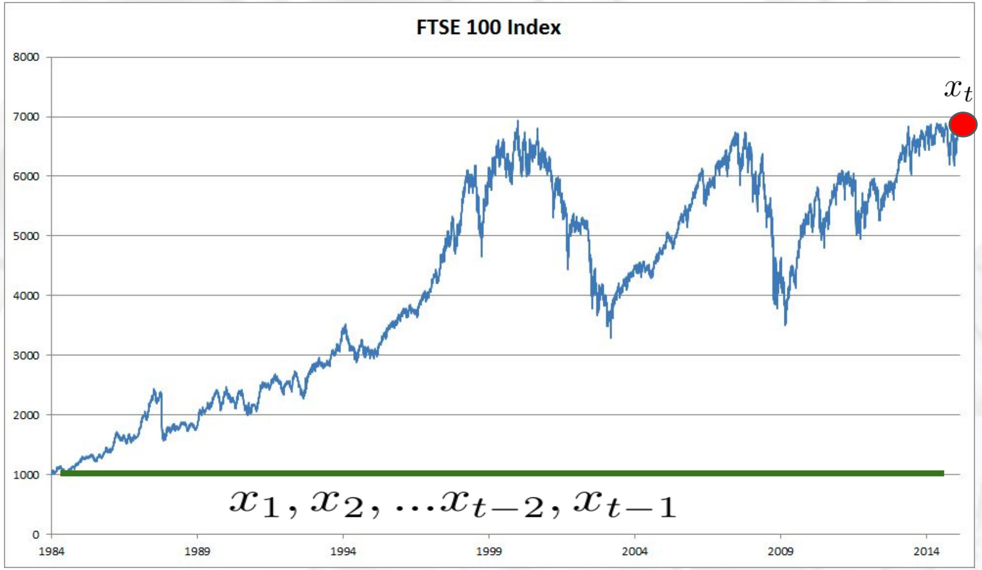
Sales

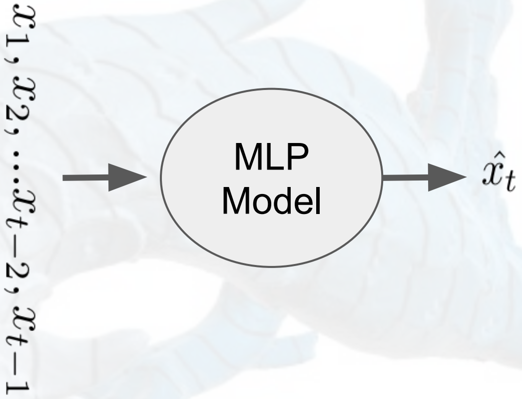
electrocardiogram

Deep Learning and applications

Text

Sequential model : Use all pass information





source: [Click]

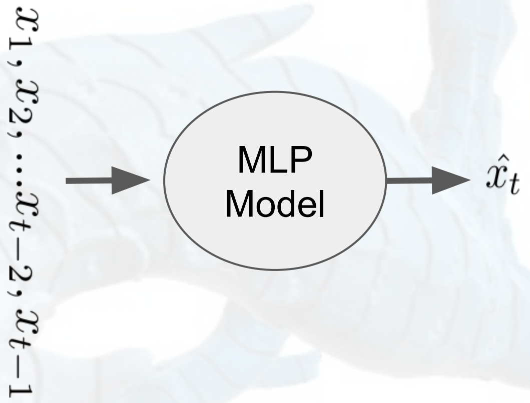
$$x_t \sim P(x_t | x_{t-1}, x_{t-2}, \ldots, x_1)$$

Sequential model : Use all pass information

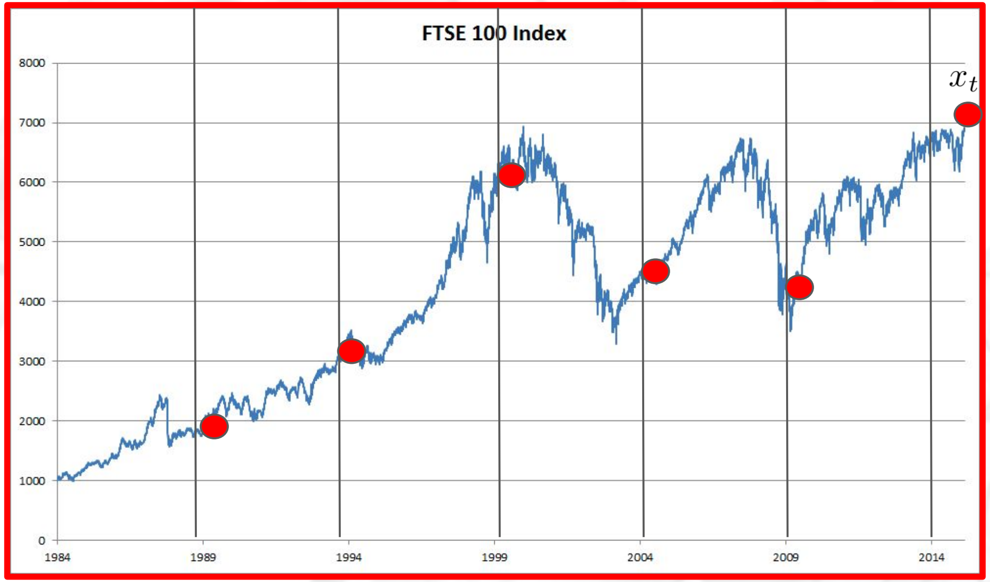
$$x_t \sim P(x_t | x_{t-1}, x_{t-2}, ..., x_1)$$

S1: Well done is better than well said

s2: Success in management requires learning as fast as the world is changing



Sequential model : Use constant length sequence



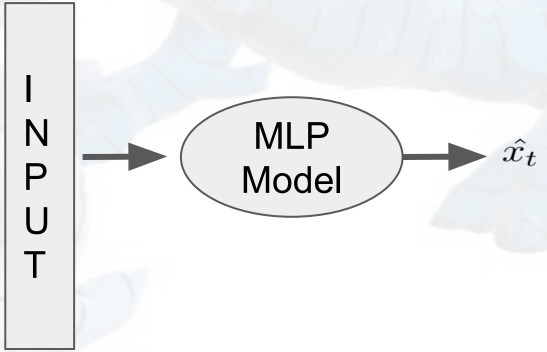
$$x_t \sim P(x_t | x_{t-1}, x_{t-2}, \ldots, x_{t-\tau})$$

$$X_1 = (\{x_1, x_2, .., x_\tau\}, \{x_{\tau+1}\})$$

$$X_2 = (\{x_{\tau+1}, x_{\tau+1}, \ldots, x_{2\tau}\}, \{x_{\tau+2}\})$$

$$\ldots$$

$$X_n = (\{x_{t-\tau}, \ldots, x_{t-2}, x_{t-1}\}, \{x_{t+1}\})$$



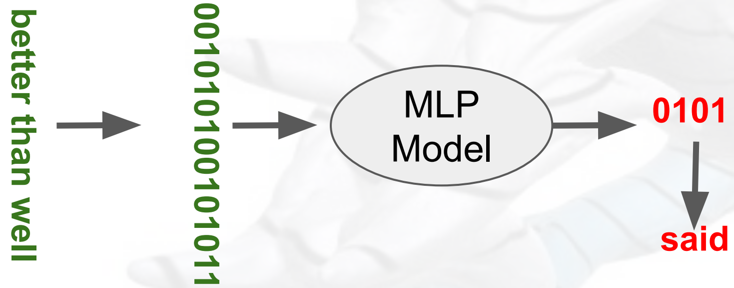
$$size = \tau$$

Sequential model : Use constant length sequence

$$x_t \sim P(x_t | x_{t-1}, x_{t-2}, \ldots, x_{t-\tau})$$

S1: Well done is better than well said

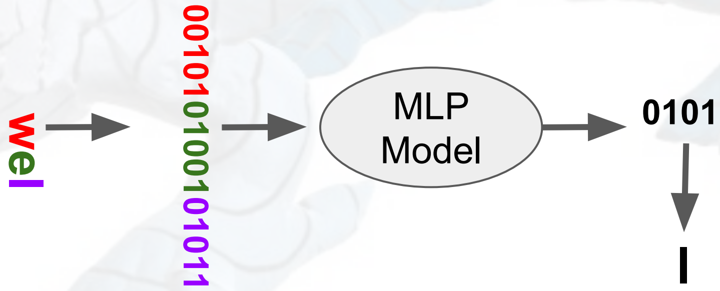
S2: Success in management requires learning as fast as the world is changing



S1: Well done is better than well said

(wel,l) (ell,) (do,n) (one,) , ….. (sad, d)

S2: Success in management requires learning as fast as the world is changing



Problem with constant length sequence

O irmão do meu pai se chama Julio. Julio é meu _____

Sequential model : Using Bag of Words

Bag of Words Example

Document 1

The quick brown fox jumped over the lazy dog's back.

Document 2

Now is the time for all good men to come to the aid of their party.

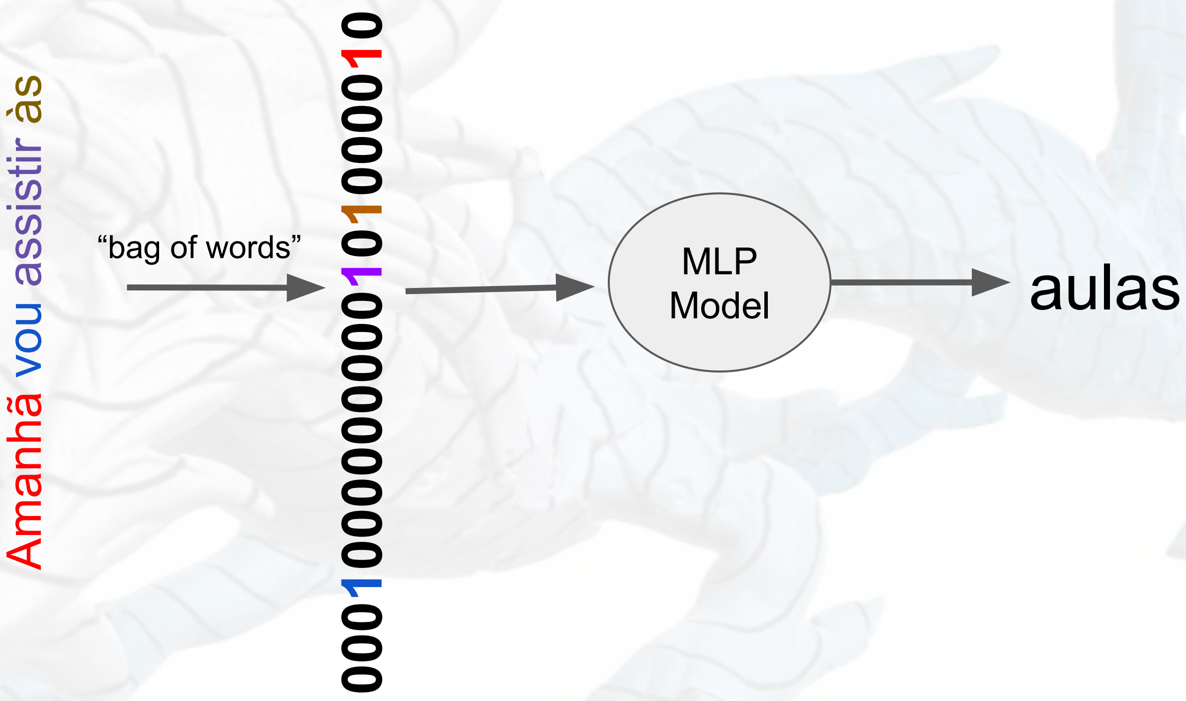
Term	Document 1	Document 2
aid	0	1
all	0	1
back	1	0
brown	1	0
come	0	1
dog	1	0
fox	1	0
good	0	1
jump	1	0
lazy	1	0
men	0	1
now	0	1
over	1	0
party	0	1
quick	1	0
their	0	1
time	0	1

Stopword List

for
is
of
the
to

source: click

Sequential model : Using Bag of Words



Bag of Words Problem

viver para trabalhar

0001000000000101000010

trabalhar para viver

0001000000000101000010

Sequential model : Using a big Fixed Windows

Amanhã vou assistir às aulas

01000 00010 000001 → MLP MODEL → aulas

Problem to solve

- Handle variable-length sequences
- Track long-term dependencies
- Maintain information about the order
- Share parameters across the sequence

More information: click

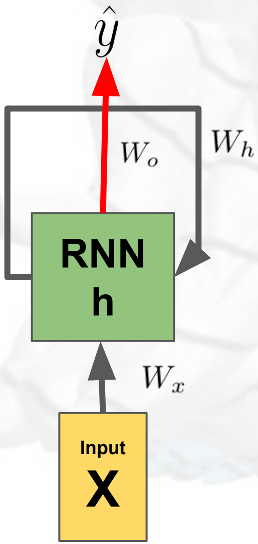
RECURRENT NEURAL NETWORK (RNNs)

- In the previous slides we speak about n-gramas.
- What happen if we use a little n-gramas ?
- What happen if we use a lot n-gramas ?

RECURRENT NEURAL NETWORK (RNNs)

n-grams model : $$x_t \sim P(x_t | x_{t-1}, x_{t-2}, ..., x_{t-\tau})$$

Latent variable model : $$P(x_t | x_{t-1}, x_{t-2}, ...x_1) \approx P(x_t | h_{t-1})$$

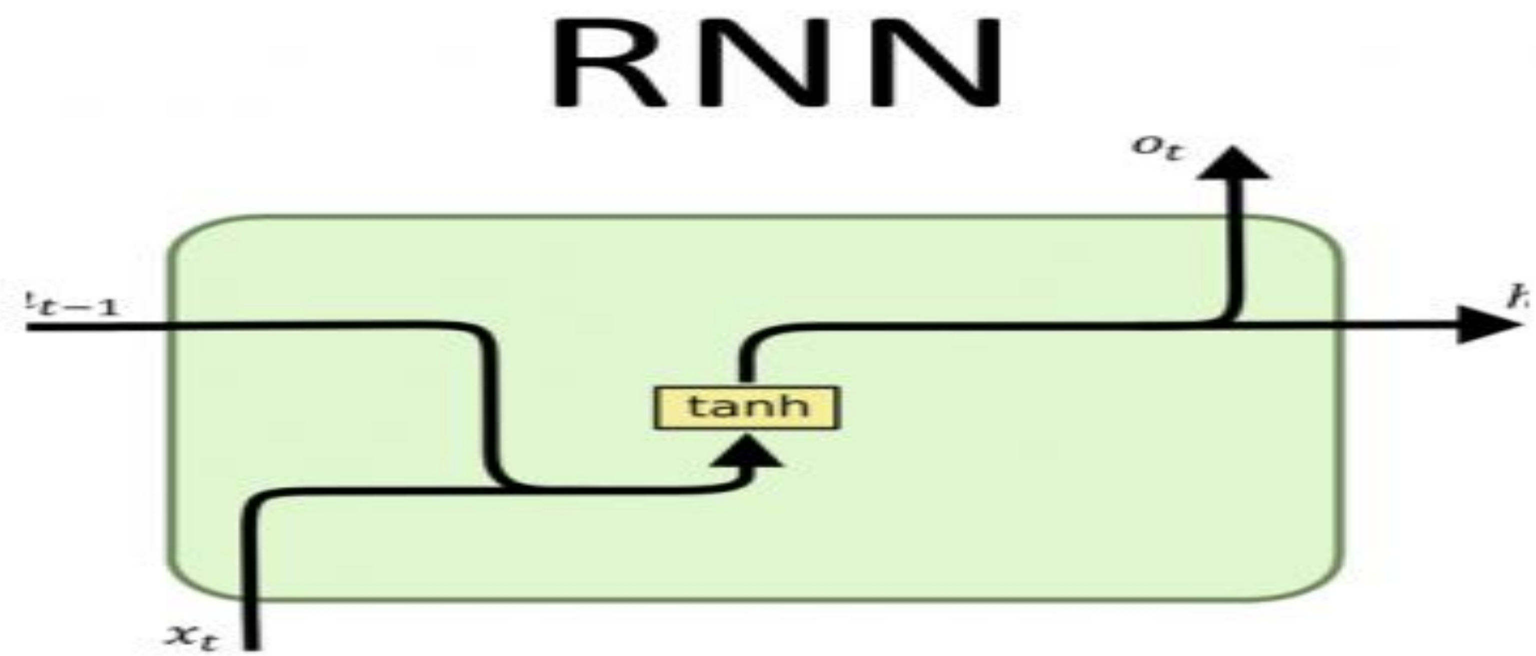


$$h_t = f_h(x_t, h_{t-1})$$

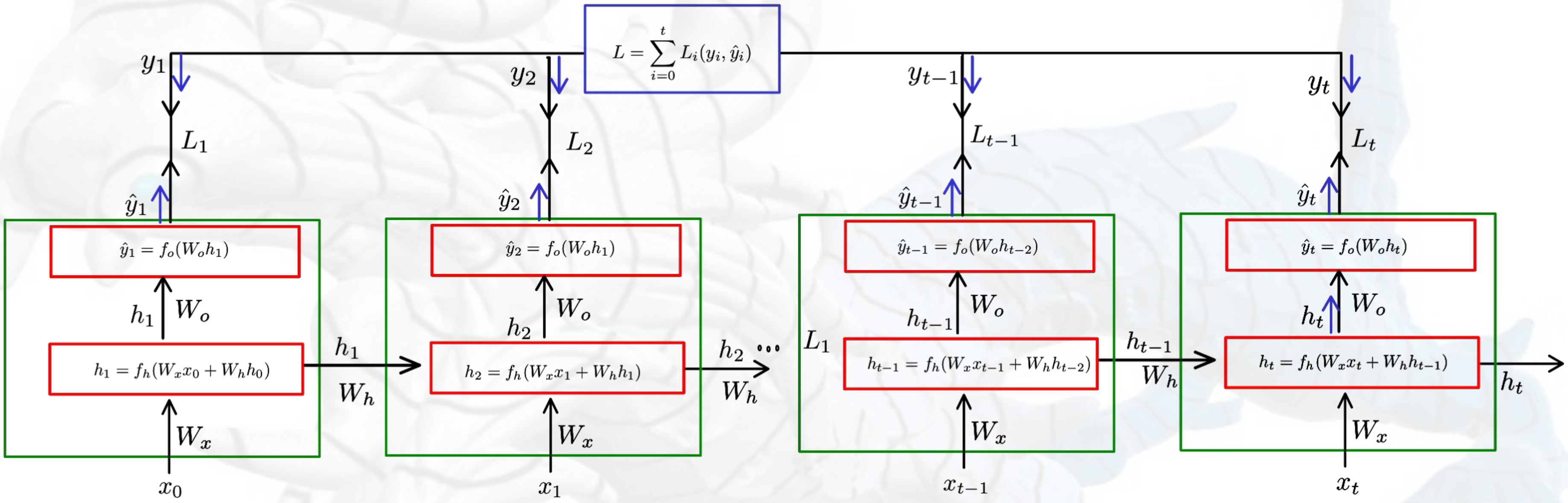
$$h_t = f_h(W_x x_t + W_h h_{t-1} + b_h)$$

$$o_t = f_o(W_o h_t + b_o)$$

RECURRENT NEURAL NETWORK (RNNs)



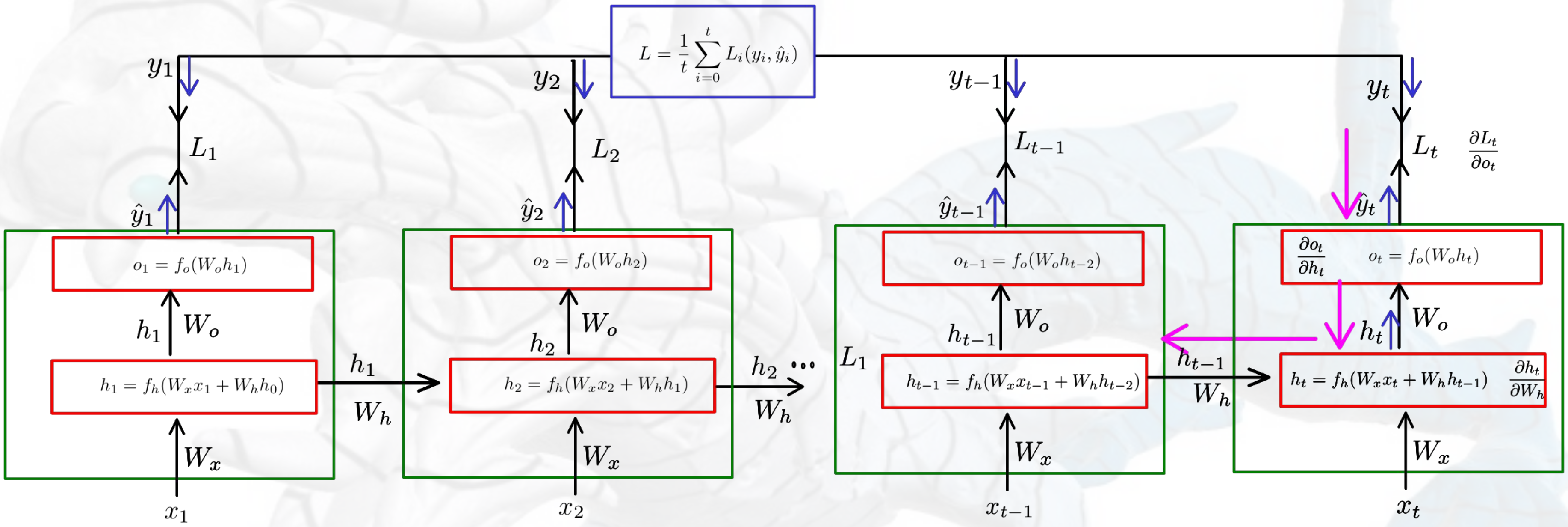
RECURRENT NEURAL NETWORK (RNNs)



RNNs: Backpropagation through time

- Exploding gradients : (1.1)^200
- Vanishing gradients : (0.9)^200

RNNs: Backpropagation through time

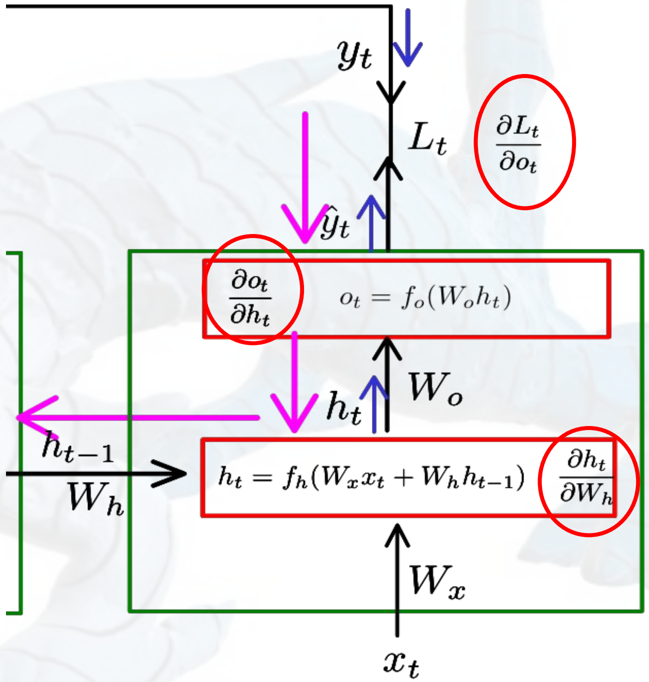


RNNs: Backpropagation through time

Loss function : $L = L_1 + L_2 + \ldots + L_t$

$$L = \frac{1}{t}\sum_{i=0}^{t} L_i(y_i, \hat{y}_i)$$

$$\frac{\partial L_t}{\partial W_h} = \frac{\partial L_t}{\partial o_t}\frac{\partial o_t}{\partial h_t}\frac{\partial h_t}{\partial W_h}$$



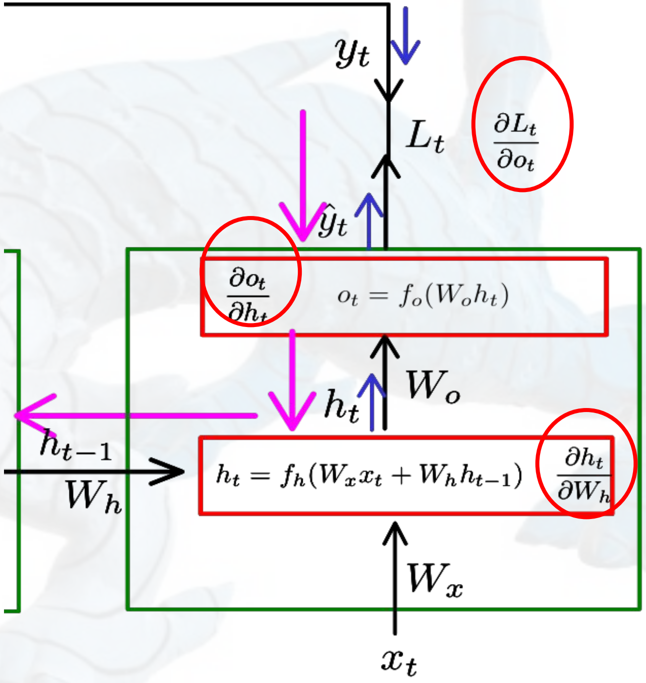
RNNs: Backpropagation through time

Loss function : $L = L_1 + L_2 + ... + L_t$

$$L = \frac{1}{t}\sum_{i=0}^{t} L_i(y_i, \hat{y}_i)$$

$$\frac{\partial L_t}{\partial W_h} = \frac{\partial L_t}{\partial o_t}\frac{\partial o_t}{\partial h_t}\frac{\partial h_t}{\partial W_h}$$

$$\frac{\partial h_t}{\partial W_h} = ???$$



RNNs: Backpropagation through time

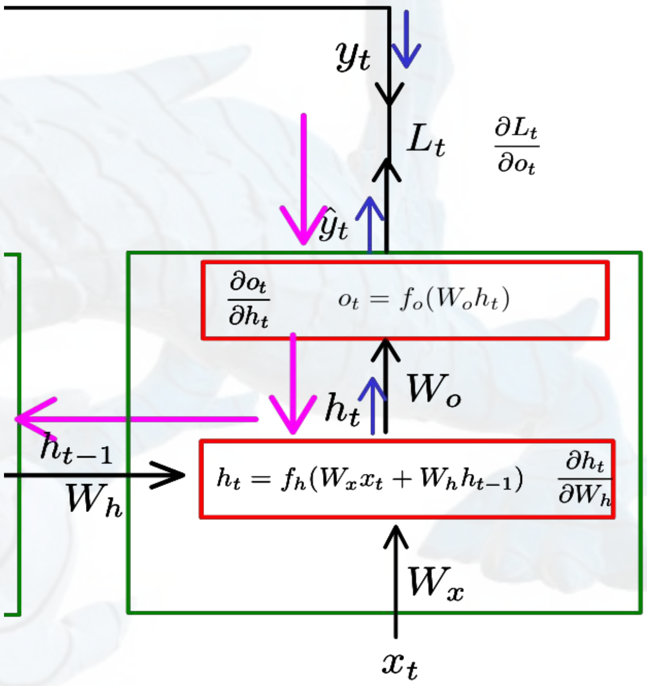
$$\frac{\partial h_t}{\partial W_h} = \frac{\partial f_h(W_x x_t + W_h h_{t-1})}{\partial W_h} + \frac{\partial f_h(W_x x_t + W_h h_{t-1})}{\partial h_{t-1}} \frac{\partial h_{t-1}}{\partial W_h}$$

$$a_t = \frac{\partial h_t}{\partial W_h} \qquad c_t = \frac{\partial f_h(W_x x_t + W_h h_{t-1})}{\partial h_{t-1}} \qquad b_t = \frac{\partial f_h(W_x x_t + W_h h_{t-1})}{\partial W_h}$$

$$a_t = b_t + c_t a_{t-1}$$

$$a_t = b_t + b_{t-1}c_t + b_{t-2}c_t c_{t-1} + b_{t-3}c_t c_{t-1} c_{t-2} + \dots + b_1 \prod_{j=1}^{t} c_j$$

$$a_t = b_t + \sum_{i=1}^{t-1} \Big(\prod_{j=i+1}^{t} c_j \Big) b_i$$



RNNs: Backpropagation through time

$$\frac{\partial h_t}{\partial W_h} = \frac{\partial f_h(W_x x_t + W_h h_{t-1})}{\partial W_h} + \sum_{i=1}^{t-1} \left(\boxed{\prod_{j=i+1} \frac{\partial f_h(W_x x_j + W_h h_{j-1})}{\partial h_{j-1}}} \right) \frac{\partial f_h(W_x x_i + W_h h_{i-1})}{\partial W_h}$$

$$\left| \cancel{\prod_{j=i+1}^{t}} \frac{\partial f_h(W_x x_j + W_h h_{j-1})}{\partial h_{j-1}} \right| > 1.1$$

$$\left| \cancel{\prod_{j=i+1}^{t}} \frac{\partial f_h(W_x x_j + W_h h_{j-1})}{\partial h_{j-1}} \right| < 0.9$$

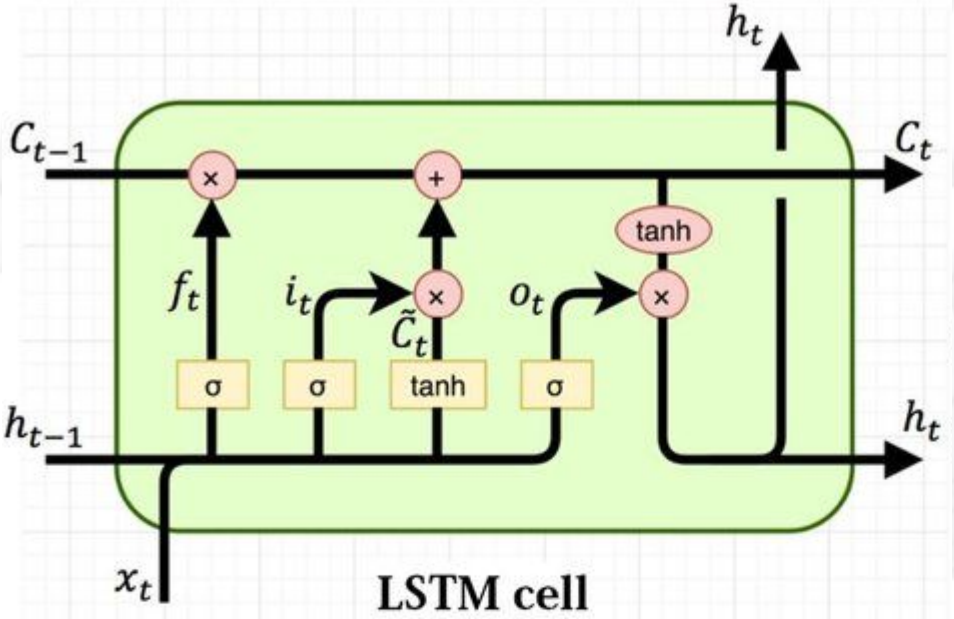
RNNs: Backpropagation through time (Truncating Time Steps)

- Truncate the sum
- Approximation of the true gradient
- In practice this works quite well.

$$\frac{\partial h_t}{\partial W_h} = \frac{\partial f_h(W_x x_t + W_h h_{t-1})}{\partial W_h} + \sum_{i=1}^{t-\tau} \left(\prod_{j=i+1} \frac{\partial f_h(W_x x_j + W_h h_{j-1})}{\partial h_{j-1}} \right) \frac{\partial f_h(W_x x_i + W_h h_{i-1})}{\partial W_h}$$

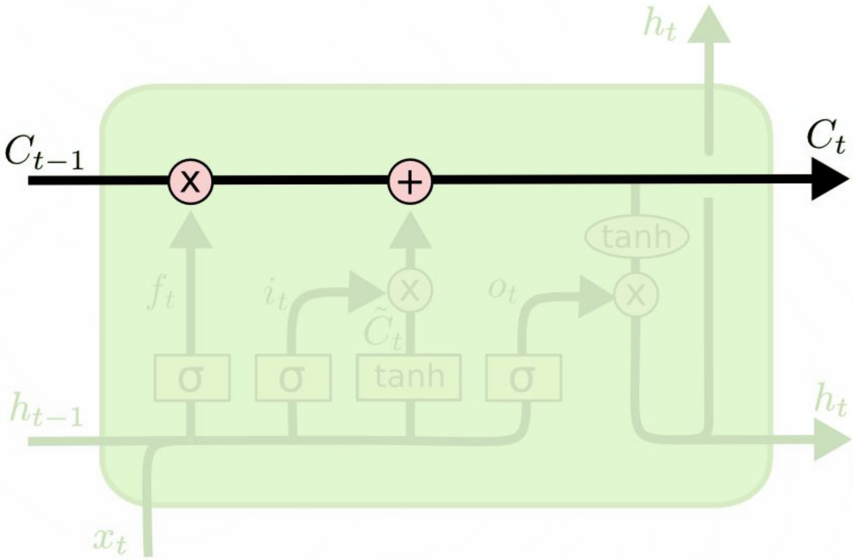
Truncated backpropagation through time

LSTM : Long Short-Term Memory (Hochreiter and Schmidhuber (1997))



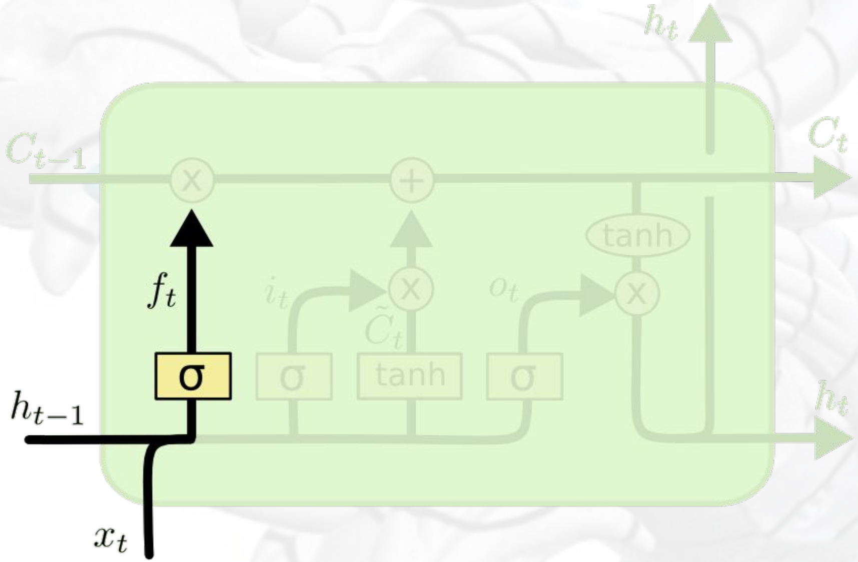
$$
\begin{aligned}
f_t &= \sigma(W_{fh}h_{t-1} + W_{fx}x_t + b_f), \\
i_t &= \sigma(W_{ih}h_{t-1} + W_{ix}x_t + b_i), \\
\tilde{c}_t &= \tanh(W_{\tilde{c}h}h_{t-1} + W_{\tilde{c}x}x_t + b_{\tilde{c}}), \\
c_t &= f_t \cdot c_{t-1} + i_t \cdot \tilde{c}_t, \\
o_t &= \sigma(W_{oh}h_{t-1} + W_{ox}x_t + b_o), \\
h_t &= o_t \cdot \tanh(c_t).
\end{aligned}
$$

LSTM : cell state



source: click

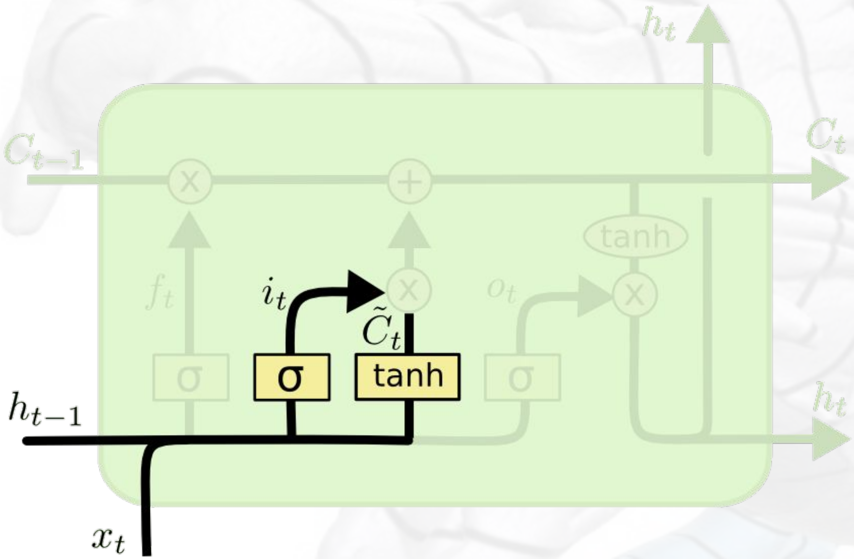
LSTM : Forgett Layer



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

source: click

LSTM : input gate layer

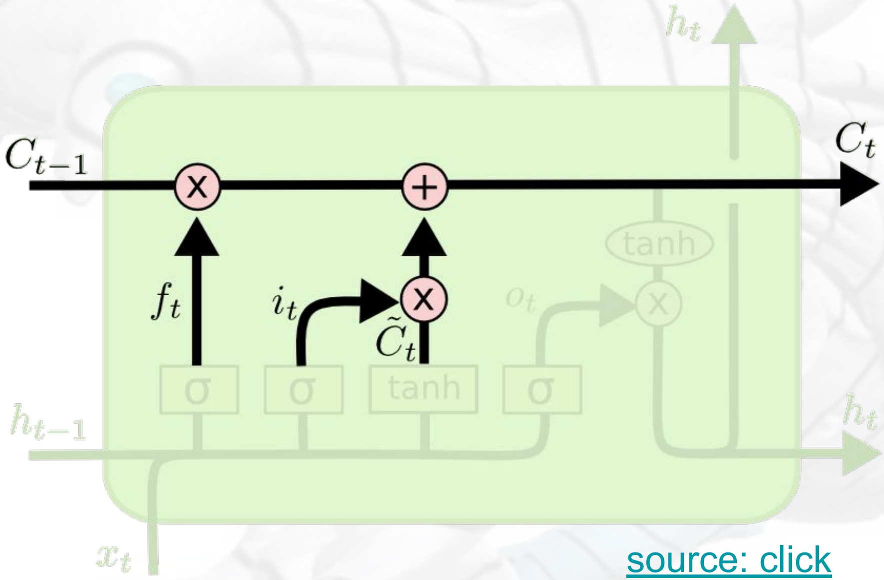


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$

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

source: click

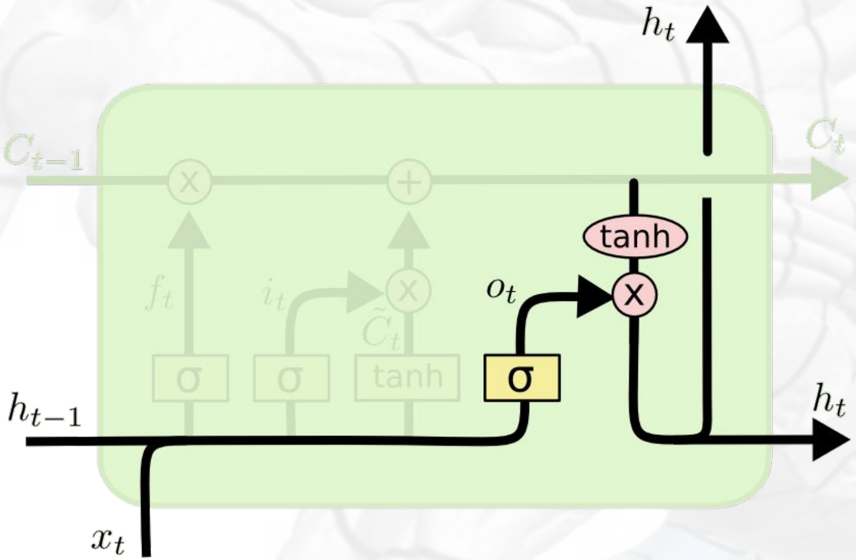
LSTM : Update the old cell state



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

source: click

LSTM : Output Layer

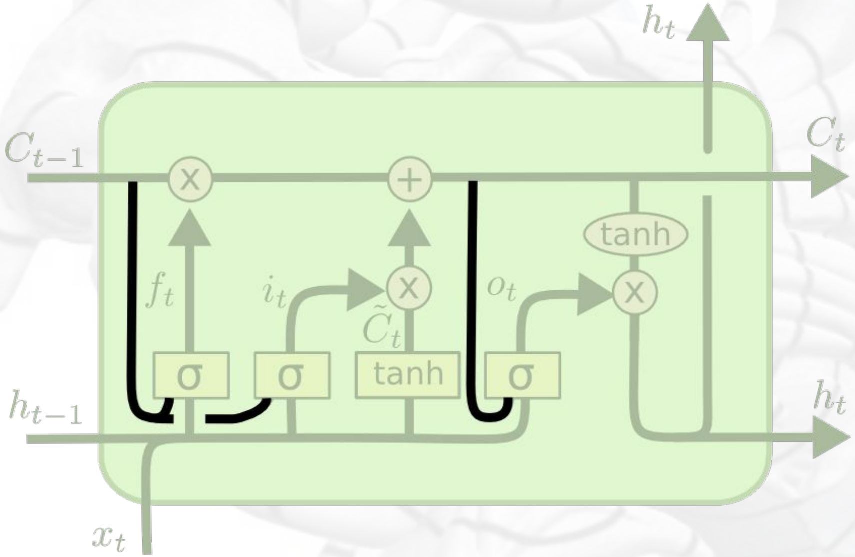


$$o_t = \sigma\left(W_o \ [h_{t-1}, x_t] \ + \ b_o\right)$$

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

source: click

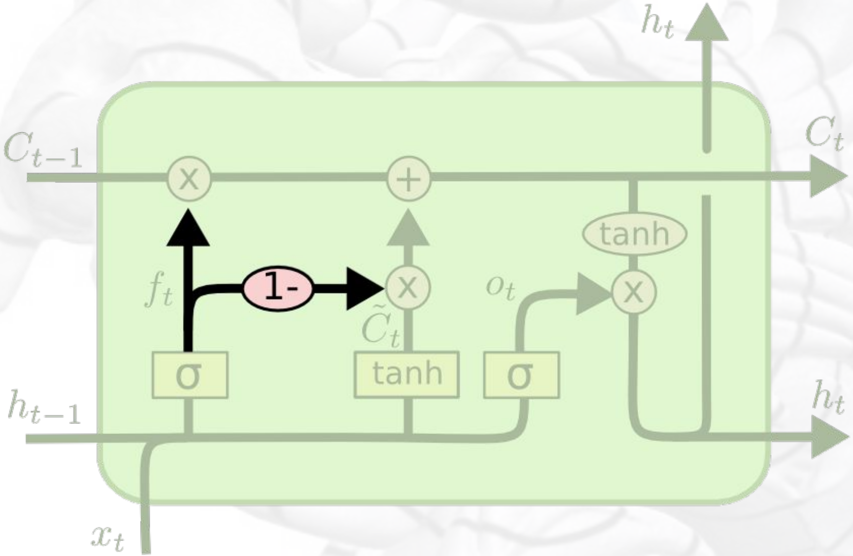
LSTM variants: peephole connections(Gers & Schmidhuber (2000))



$$
\begin{aligned}
f_t &= \sigma(W_{fh}h_{t-1} + W_{fx}x_t + P_f \cdot c_{t-1} + b_f),\\
i_t &= \sigma(W_{ih}h_{t-1} + W_{ix}x_t + P_i \cdot c_{t-1} + b_i),\\
\tilde{c}_t &= \tanh(W_{\tilde{c}h}h_{t-1} + W_{\tilde{c}x}x_t + b_{\tilde{c}}),\\
c_t &= f_t \cdot c_{t-1} + i_t \cdot \tilde{c}_t,\\
o_t &= \sigma(W_{oh}h_{t-1} + W_{ox}x_t + P_o \cdot c_t + b_o),\\
h_t &= o_t \cdot \tanh(c_t),
\end{aligned}
$$

[source: click]

LSTM variants



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

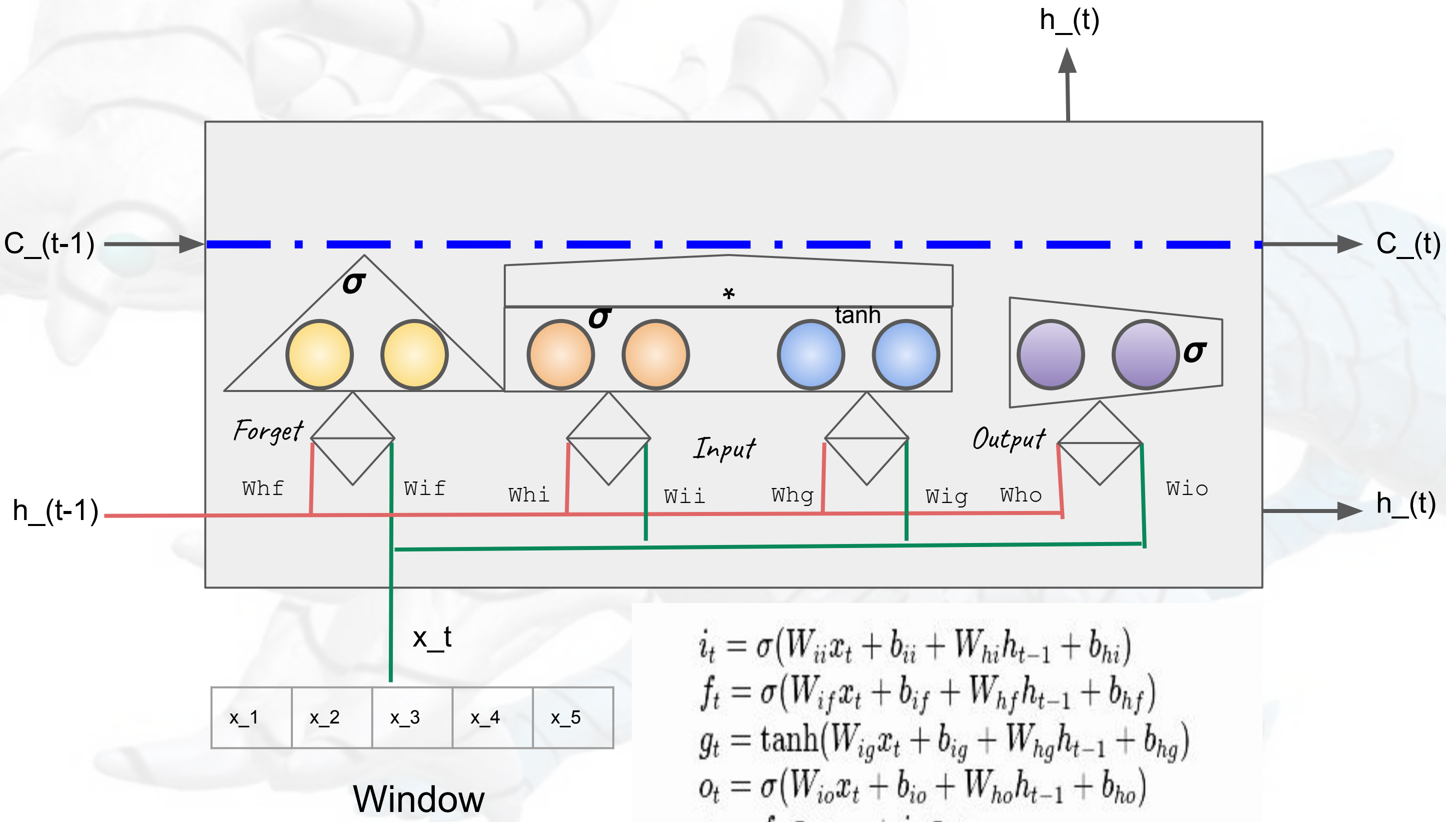
source: click

Programa

```
x = torch.rand(1000,1,5)
lstm = nn.LSTM(input_size = 5,hidden_size = 2, num_layers = 1)
print(lstm.eval())
p , _= lstm(x)
print(p.shape)
print([p.shape for p in lstm.parameters()])
```

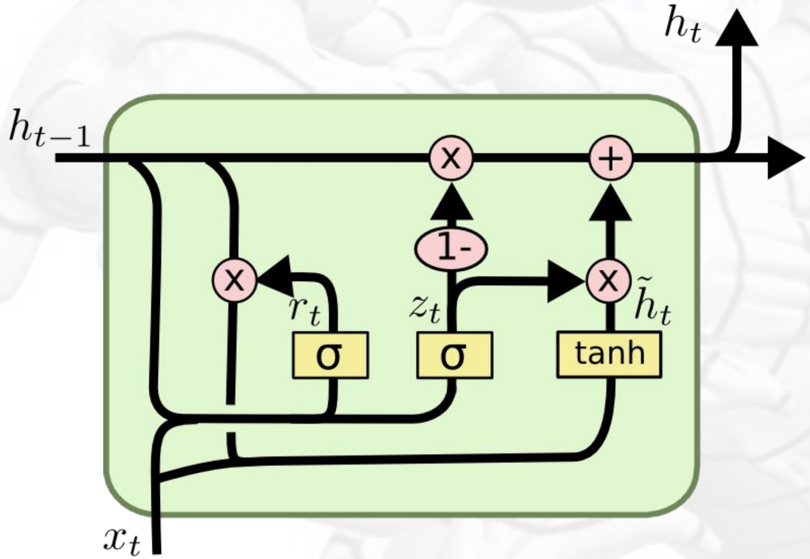
Salida

```
torch.Size([1000, 1, 5])
LSTM(5, 2)
torch.Size([1000, 1, 2])
[torch.Size([8, 5]), torch.Size([8, 2]), torch.Size([8]),
torch.Size([8])]
```



$$
\begin{aligned}
i_t &= \sigma(W_{ii}x_t + b_{ii} + W_{hi}h_{t-1} + b_{hi}) \\
f_t &= \sigma(W_{if}x_t + b_{if} + W_{hf}h_{t-1} + b_{hf}) \\
g_t &= \tanh(W_{ig}x_t + b_{ig} + W_{hg}h_{t-1} + b_{hg}) \\
o_t &= \sigma(W_{io}x_t + b_{io} + W_{ho}h_{t-1} + b_{ho}) \\
c_t &= f_t \odot c_{t-1} + i_t \odot g_t \\
h_t &= o_t \odot \tanh(c_t)
\end{aligned}
$$

GRU (Cho et al. (2014))

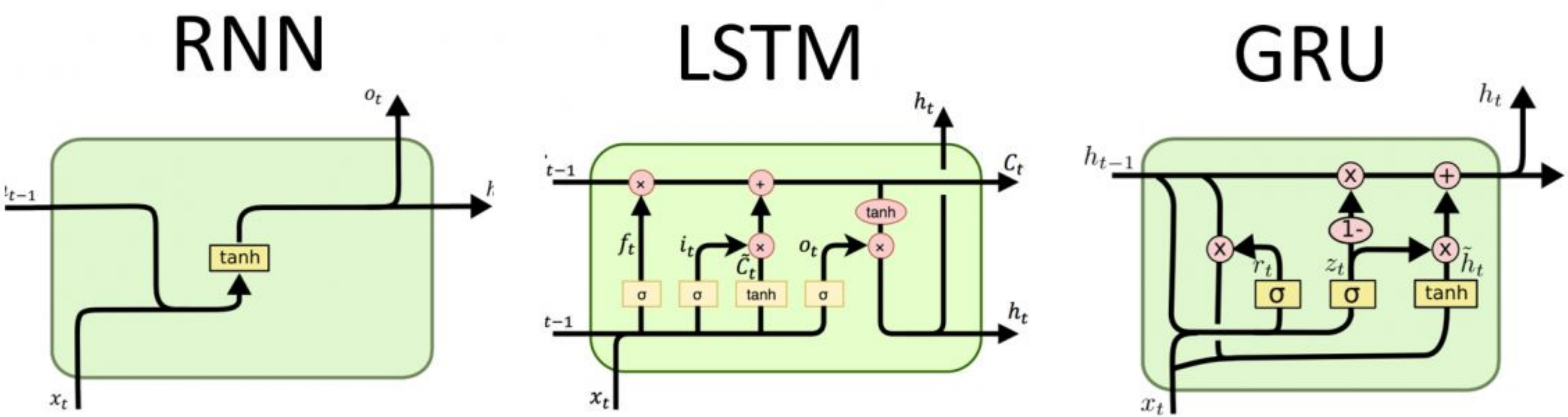


$$r_t = \sigma(W_{rh}h_{t-1} + W_{rx}x_t + b_r),$$

$$z_t = \sigma(W_{zh}h_{t-1} + W_{zx}x_t + b_z),$$

$$\tilde{h}_t = \tanh(W_{\tilde{h}h}(r_t \cdot h_{t-1}) + W_{\tilde{h}x}x_t + b_z),$$

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



[Pytorch example]

Next Class

- LSTM with Text
- Embedding word method
    - Binary Encoding
    - TF Encoding
    - TF-IF Encoding
    - Word2Vec Embedding
    - Glove
    - Beth
- Application

RNNs: Backpropagation through time (Full Computation)

RECURRENT NEURAL NETWORK

Cristian López Del Alamo

January 10, 2021