SAISON 22/23

Formation

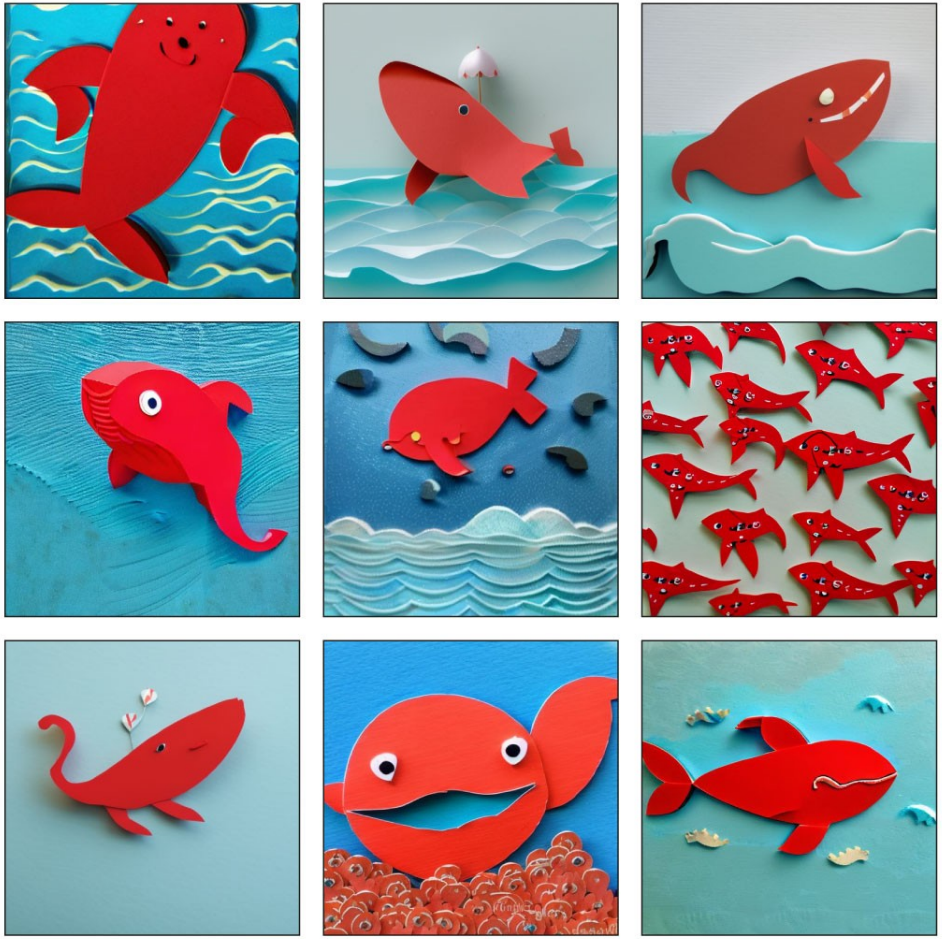
Introduction Deep Learning

Séquence 06

Données séquentielles et/ou temporelles. Réseaux de Neurones Récurrents (RNN)

FIDLE

https://fidle.cnrs.fr



Bonne et excellente année 2023 !

Que cette nouvelle année
puisse vous apporter santé, bonheur
et de nombreux projets riches de
sens !

Images générées via le modèle StableDiffusion 2.
"A Happy red whale on a blue ocean paper".
Idée et réalisation: Pierre C. (IDRIS)

https://beta.dreamstudio.ai/
https://beta.dreamstudio.ai/dream

Cette session va être enregistrée.
Retrouvez-nous sur notre chaine YouTube :-)

This session will be recorded.
Find us on our YouTube channel :-)

https://fidle.cnrs.fr/youtube

SAISON 22/23

Formation

Introduction Deep Learning

Séquence 06

Données séquentielles et/ou temporelles. Réseaux de Neurones Récurrents (RNN)

FIDLE

https://fidle.cnrs.fr

MIAI Grenoble Alpes

cnrs

UGA Université Grenoble Alpes

EFELIA - MIAI
ECOLE FRANÇAISE DE L'INTELLIGENCE ARTIFICIELLE

DEVLOG RÉSEAU DU DÉVELOPPEMENT LOGICIEL

Resinfo

sari

ANITI

GRICAD

SIMaP

Resources

https://fidle.cnrs.fr

Powered by CNRS CRIC, and UGA DGDSI
of Grenoble, Thanks !

- Course materials (pdf)
- Practical work environment*
- Corrected notebooks
- Videos (YouTube)

(*) Procedure via Docket or pip
Remember to get the latest version !

Resources

You can also subscribe to :

FIDLE http://fidle.cnrs.fr/listeinfo

DEVLOG RÉSEAU DU DEVELOPPEMENT LOGICIEL Recherche et Enseignement Supérieur https://listes.services.cnrs.fr/wws/info/devlog[1]

GROUPECALCUL https://listes.math.cnrs.fr/wws/info/calcul[2]

(1) List of ESR* developers,
(2) List of ESR* « calcul » group
Where ESR is Enseignement Supérieur et Recherche, french universities and public academic research organizations

Program

20 Séquences
du 17 novembre au 14 mai 2023

1. **History, Fundamental Concepts**
   - **Basic Regression** DNN
   - **Basic Classification** DNN
2. 3. **Hight Dimensionnal Data** CNN
4. Demystify **mathematics** for neural networks.
5. **Training strategies** Evaluation — **Sparse data (text)** Embedding
6. **Sequences data** RNN
7. PyTorch — A small detour with **PyTorch**.
8. **«Attention is All You Need»** Transformers
9. New ! **Graph Neural Network** GNN
10. **Autoencoder networks** AE
11. **Variational Antoencoder** VAE
12. **Project session** «My project in 180 s»
13. **Generative Adversarial Networks** GAN
14. New ! **Diffusion Model** Text to image
15. **AI, Law, Society** and **Ethics**
16. New ! **Model and training optimization** Resource efficiency
17. **Jean-Zay** GPU acceleration
18. New ! **Physics-Informed Neural Networks** PINNS
19. **Deep Reinforcement Learning** RL
20. **What will be tomorrow's AI** Review & perspectives !

FIDLE

SAISON 22/23

Roadmap

6

Sequences data
RNN

6.1 **Sequences data**

- Recurrent Neural Network
- LSTM and GRU

6.2 **Example 1 : Ladybug1**

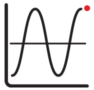
- Prediction of a virtual trajectory

6.3 **Example 2 : SYNOP1/3**

- Weather prediction at 3h and 12h

Roadmap

6



Sequences data
RNN

6.1 **Sequences data**

- → Recurrent Neural Network
- → LSTM and GRU

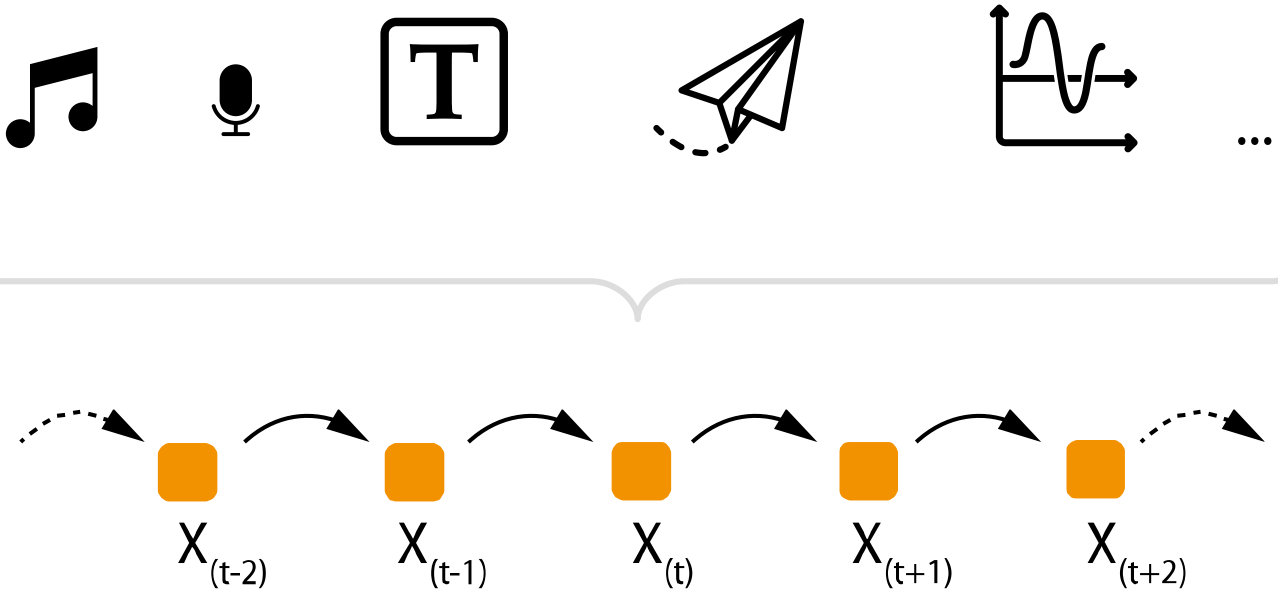
6.2 **Example 1 : Ladybug1**

- → Prediction of a virtual trajectory

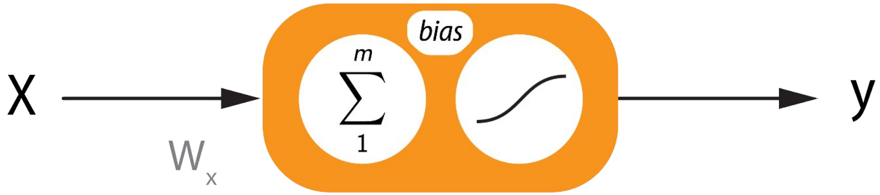
6.3 **Example 2 : SYNOP1/3**

- → Weather prediction at 3h and 12h

What if the world was just one big sequence ?



From classic to recurrent neuron

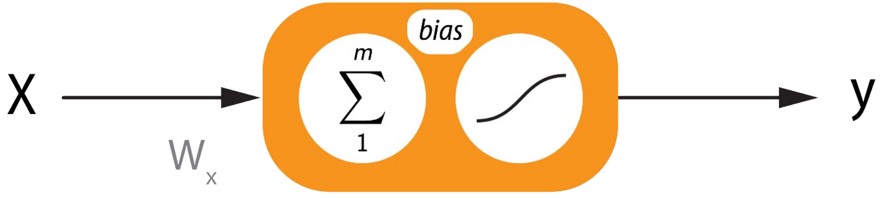


$$y = \sigma(W_x^T \cdot X + b)$$

Classical neuron.

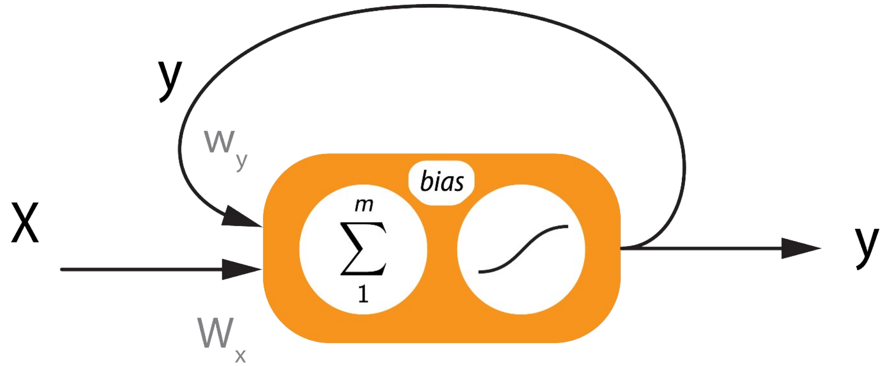
From classic to recurrent neuron

Where :
w_y : is a scalar
W_x : is a vector
b : is a scalar
y : is a scalar



Classical neuron.

$$y = \sigma(W_x^T \cdot X + b)$$

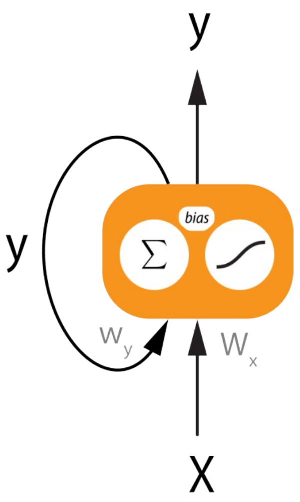


Recurrent neuron.

$$y_{(t)} = \sigma(W_x^T \cdot X_{(t)} + w_y \cdot y_{(t-1)} + b)$$

Simple reccurent neuron

Where :
w_y : is a scalar
W_x : is a vector
b : is a scalar
y : is a scalar

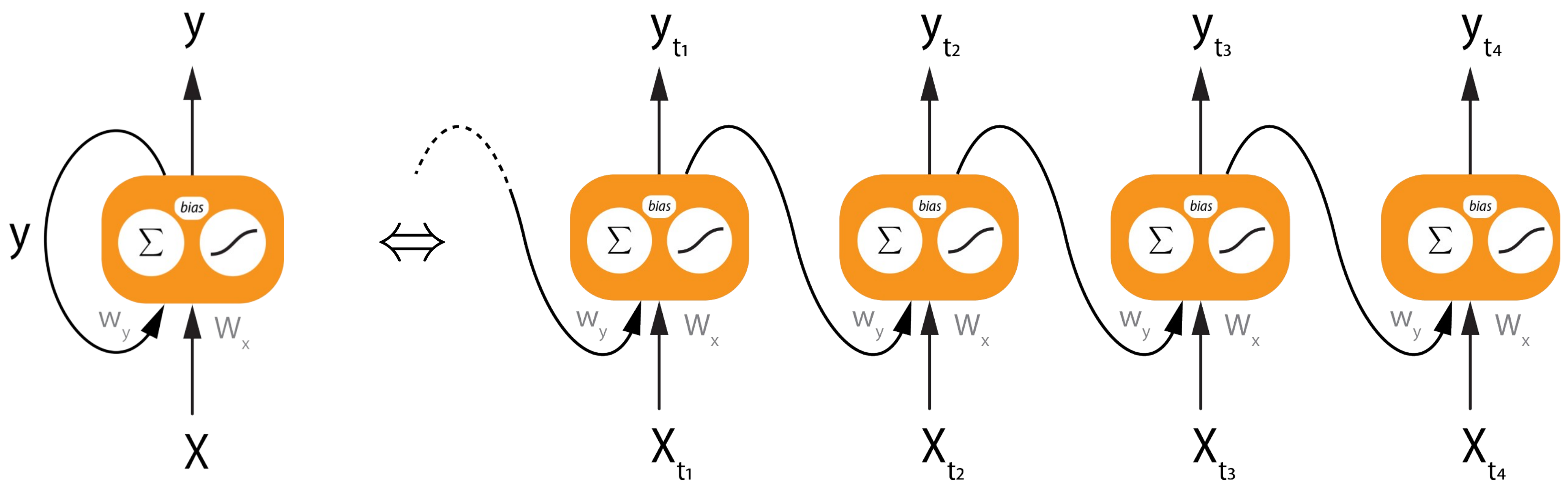


$$y_{(t)} = \sigma(W_x^T \cdot X_{(t)} + w_y \cdot y_{(t-1)} + b)$$

Recurrent neuron.

Simple reccurent neuron

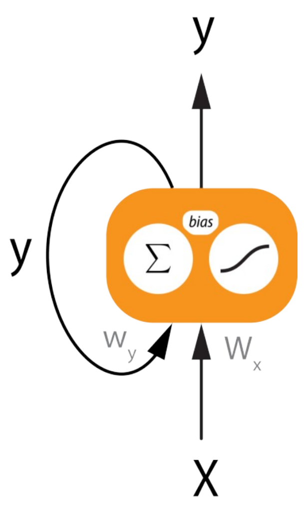
Where :
w_y : is a scalar
W_x : is a vector
b : is a scalar
y : is a scalar



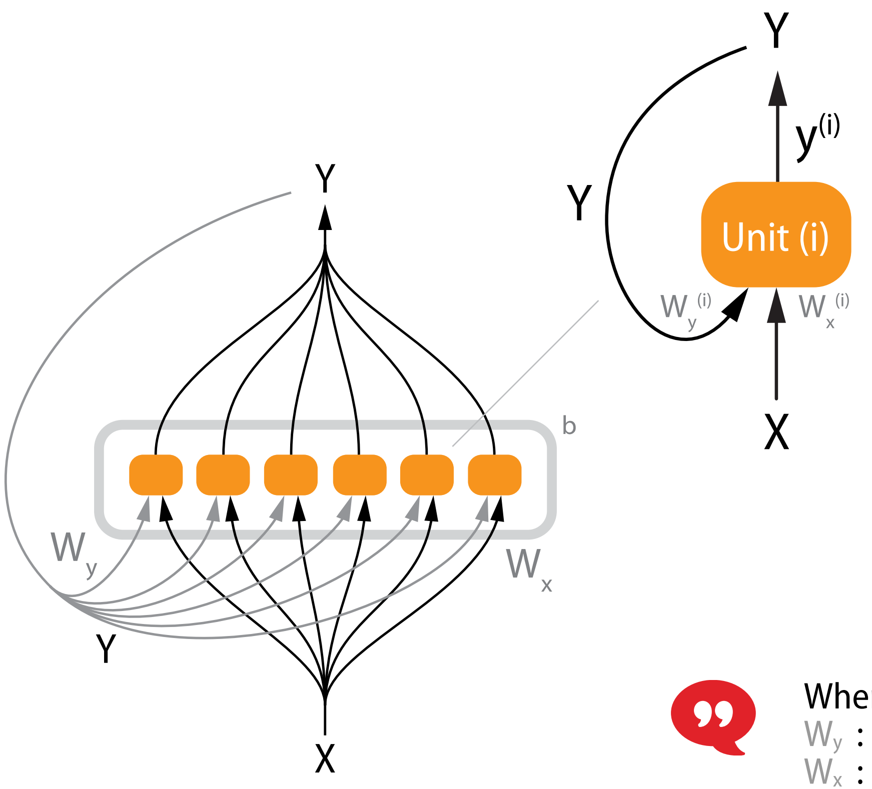
Recurrent neuron.

$$y_{(t)} = \sigma(W_x^T \cdot X_{(t)} + w_y \cdot y_{(t-1)} + b)$$

Recurrent Layer / Cell



Recurrent neuron.

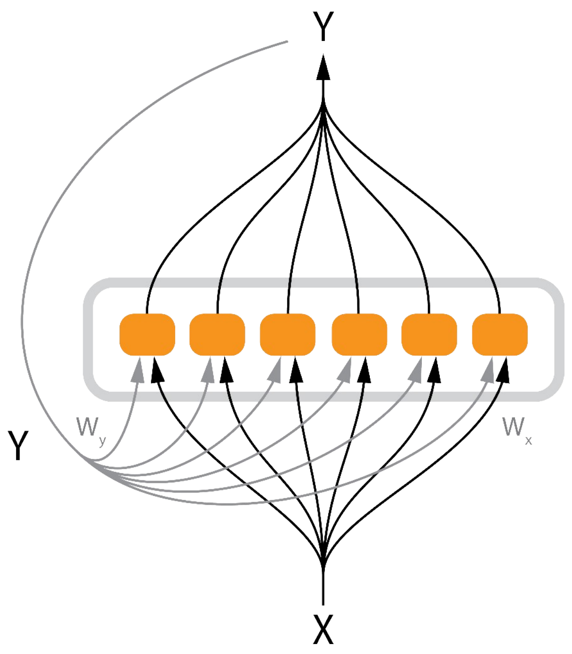


Recurrent Layer / Cell

Where :
W_y : is a tensor
W_x : is a tensor
b : is a vector
Y : is a vector

Recurrent Layer / Cell

Where :
W$_y$: is a tensor
W$_x$: is a tensor
b : is a vector
Y : is a vector

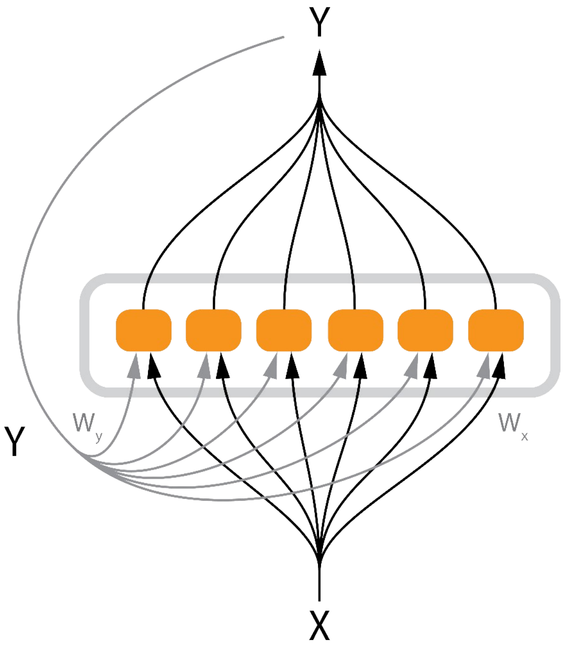


Recurrent Layer / Cell

$$Y_{(t)} = \Phi\left(W_x^T \cdot X_{(t)} + W_y^T \cdot Y_{(t-1)} + b\right)$$

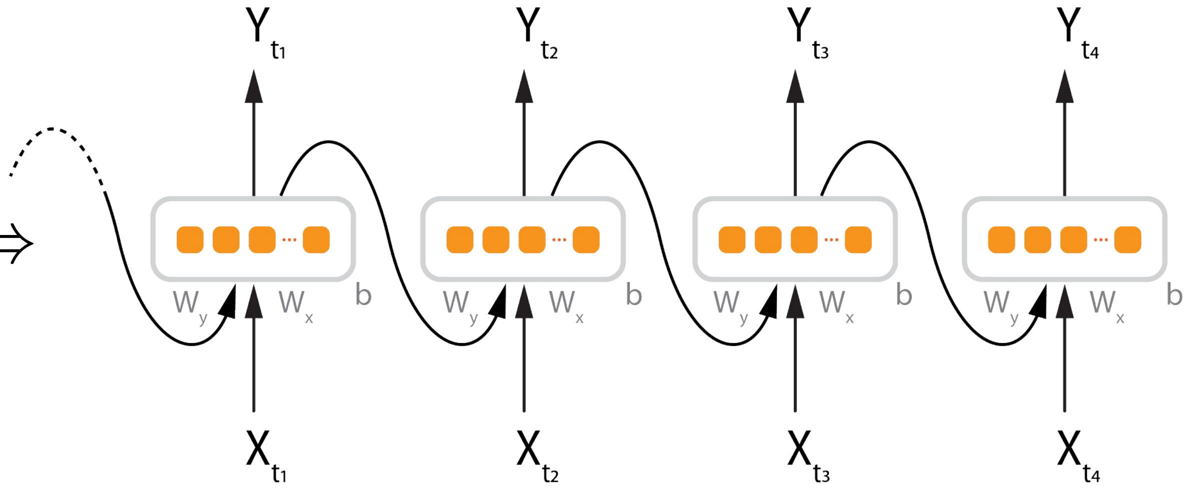
Recurrent Layer / Cell

Where :
W_y : is a tensor
W_x : is a tensor
b : is a vector
Y : is a vector



Recurrent Layer / Cell

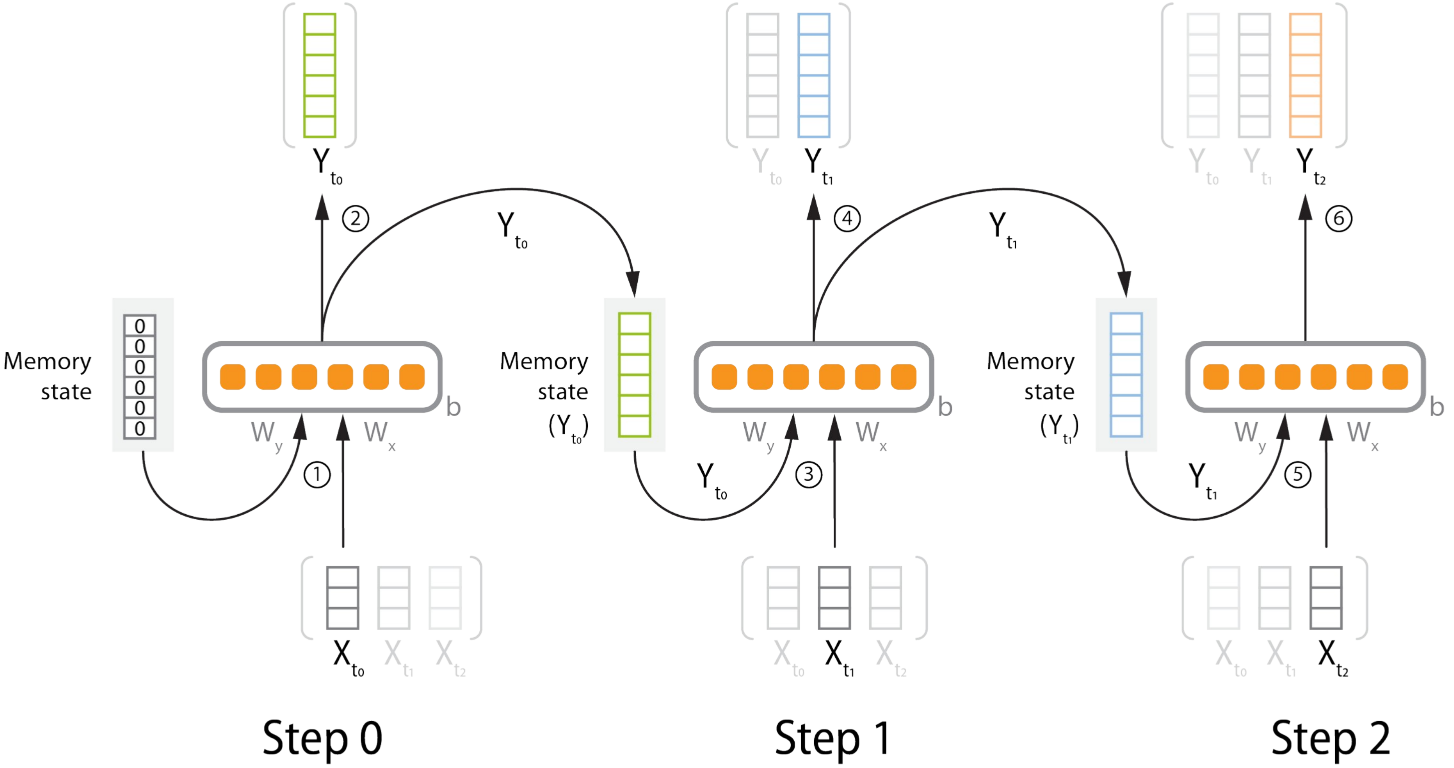
$\Leftrightarrow$



$$Y_{(t)} = \Phi \left(W_x^T \cdot X_{(t)} + W_y^T \cdot Y_{(t-1)} + b \right)$$

Recurrent Layer / Cell

W_x shape is : (nb units, x size)
W_y shape is : (nb units, nb units)
y shape is : (nb units,)

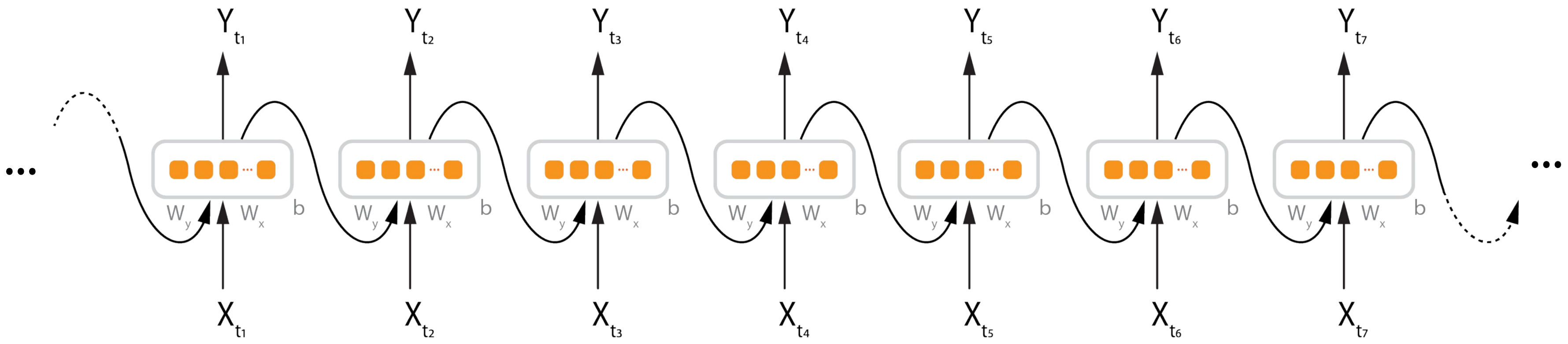


Recurrent Layer / Cell

$$Y_{(t)} = \Phi\left(W_x^T \cdot X_{(t)} + W_y^T \cdot Y_{(t-1)} + b\right)$$

Simple RNN limits...

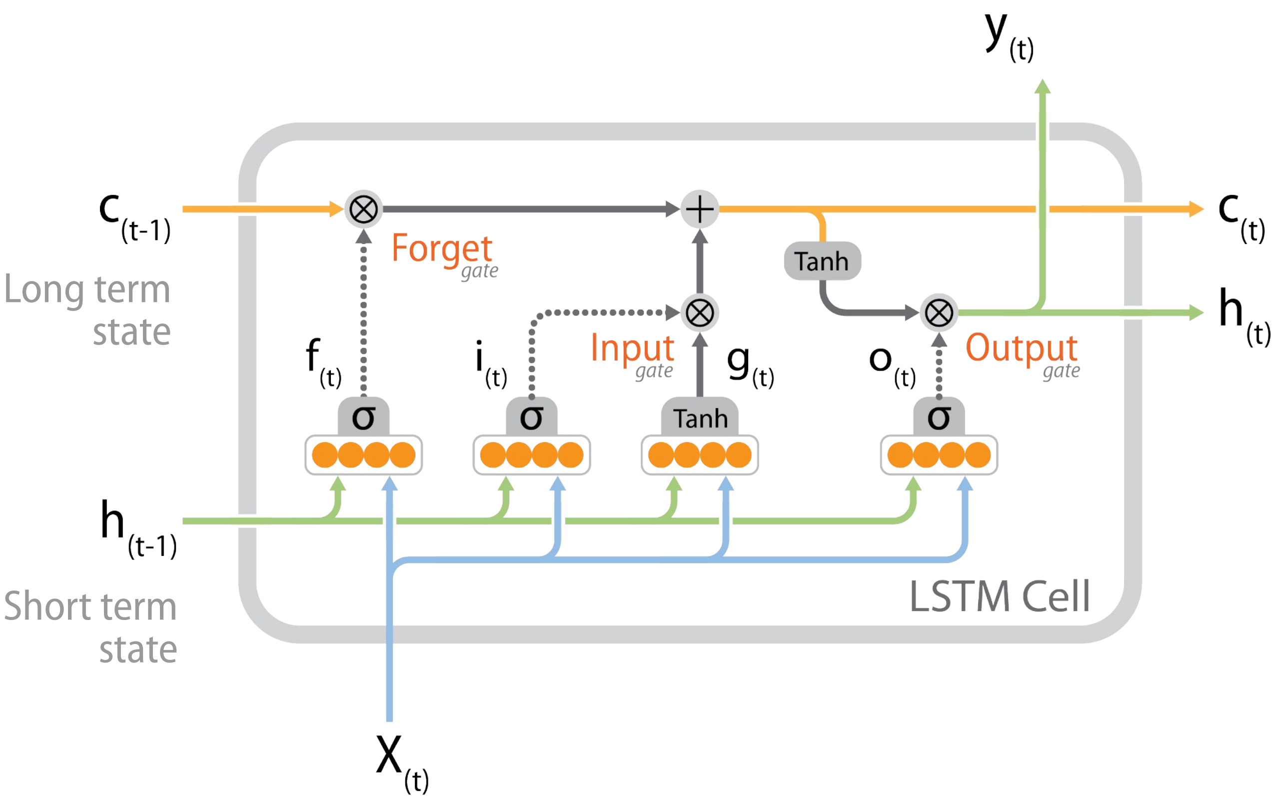
But….



Slow convergence,
Short memory,
Vanishing / exploding gradients

⇒ In short... it doesn't work !

Long Short-Term Memory (LSTM)



$$f_{(t)} = \sigma(W_{xf}^T X_{(t)} + W_{hf}^T h_{(t-1)} + b_f)$$
$$i_{(t)} = \sigma(W_{xi}^T X_{(t)} + W_{hi}^T h_{(t-1)} + b_i)$$
$$g_{(t)} = \tanh(W_{xg}^T X_{(t)} + W_{hg}^T h_{(t-1)} + b_g)$$
$$o_{(t)} = \sigma(W_{xo}^T X_{(t)} + W_{ho}^T h_{(t-1)} + b_o)$$
$$c_{(t)} = f_{(t)} \otimes c_{(t-1)} + i_{(t)} \otimes g_{(t)}$$
$$y_{(t)} = h_{(t)} = o_{(t)} \otimes \tanh(c_{(t)})$$

with :

$X_{(t)} \in \mathbb{R}^d$	input vector
$f_{(t)} \in \mathbb{R}^h$	forget gate's activation vector
$i_{(t)} \in \mathbb{R}^h$	input gate's activation vector
$o_{(t)} \in \mathbb{R}^h$	output gate's activation vector
$g_{(t)} \in \mathbb{R}^h$,	current entry vector
$h_{(t)}, y_{(t)} \in \mathbb{R}^h$	hidden state or output vector
$c_{(t)} \in \mathbb{R}^h$	cell state vector
$\otimes$	Hadamard product
σ	sigmoid function
W_k	weights matrix
b_k	bias vector

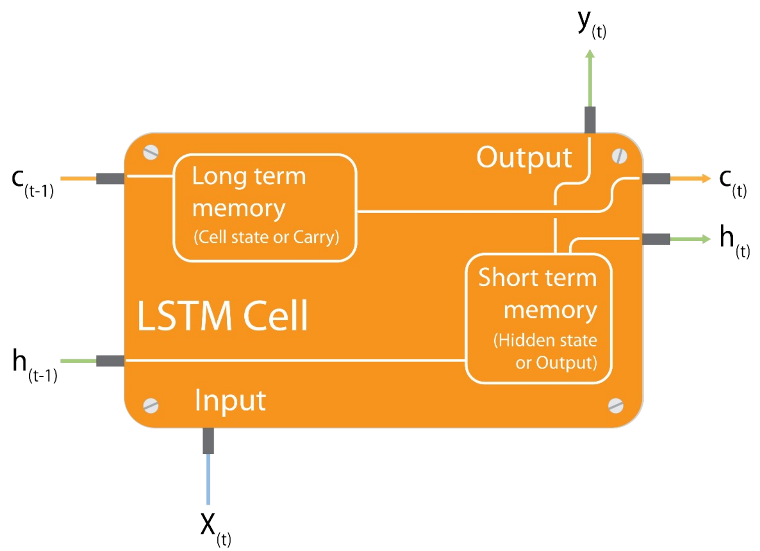
Long short-term memory (LSTM)[1]

Gated recurrent unit (GRU)[2]

[1] Sepp Hochreiter, Jürgen Schmidhuber, (1997) [LSTM]

[2] Kyunghyun Cho et al, (2014) [GRU]

Long Short-Term Memory (LSTM)




```
inputs  = tf.random.normal([32, 20, 8])

lstm    = tf.keras.layers.LSTM(16)
output  = lstm(inputs)
```

Serie to vector

```
Inputs shape is :  (32, 20, 8)
Output shape is :  (32, 16)
```

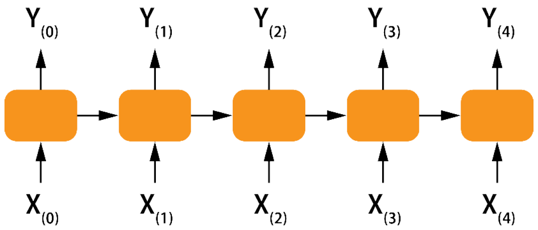
```
lstm = tf.keras.layers.LSTM(18, return_sequences=True, return_state=True)

output, memory_state, carry_state = lstm(inputs)
```

Serie to serie

```
Output shape :  (32, 20, 18)
Memory state :  (32, 18)
Carry  state :  (32, 18)
```

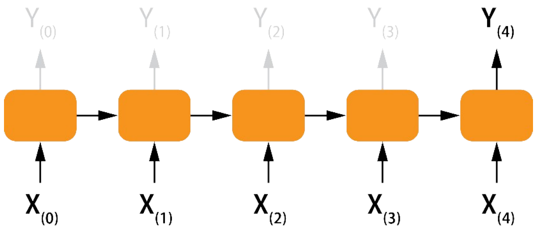
More about :
https://keras.io/api/layers/recurrent_layers/lstm/
https://www.tensorflow.org/api_docs/python/tf/keras/layers/LSTM

Reccurent Neural Network (RNN)



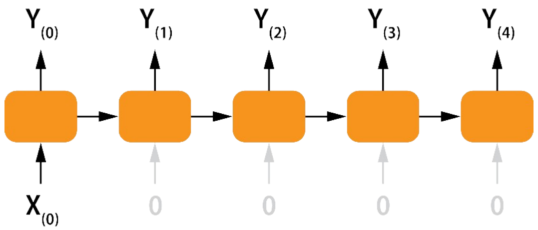
Serie to serie

Example : Time serie prediction



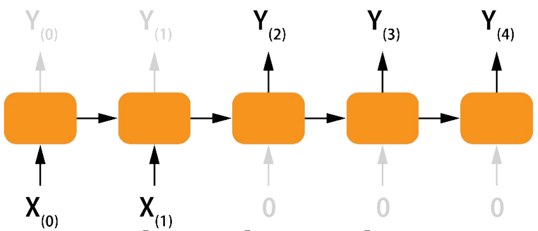
Serie to vector

Example : Sentiment analysis



Vector to serie

Example : Image annotation



Encoder-decoder

Example : Language Translation

How to predict a sequence ?

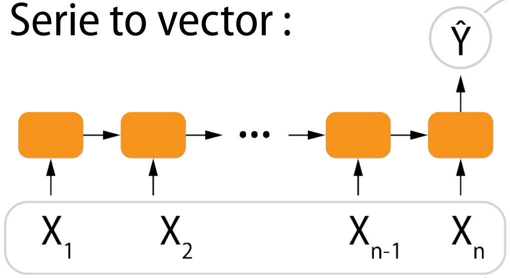
Known sequence : [$X_1, X_2, \ldots, X_n, X_{n+1}$]

Input Sequence : $X = [X_1, X_2, \ldots, X_n]$

Expected output : $Y = [X_{n+1}]$

The objective will be to train our RNN network to predict the n+1 vector of our input sequence :

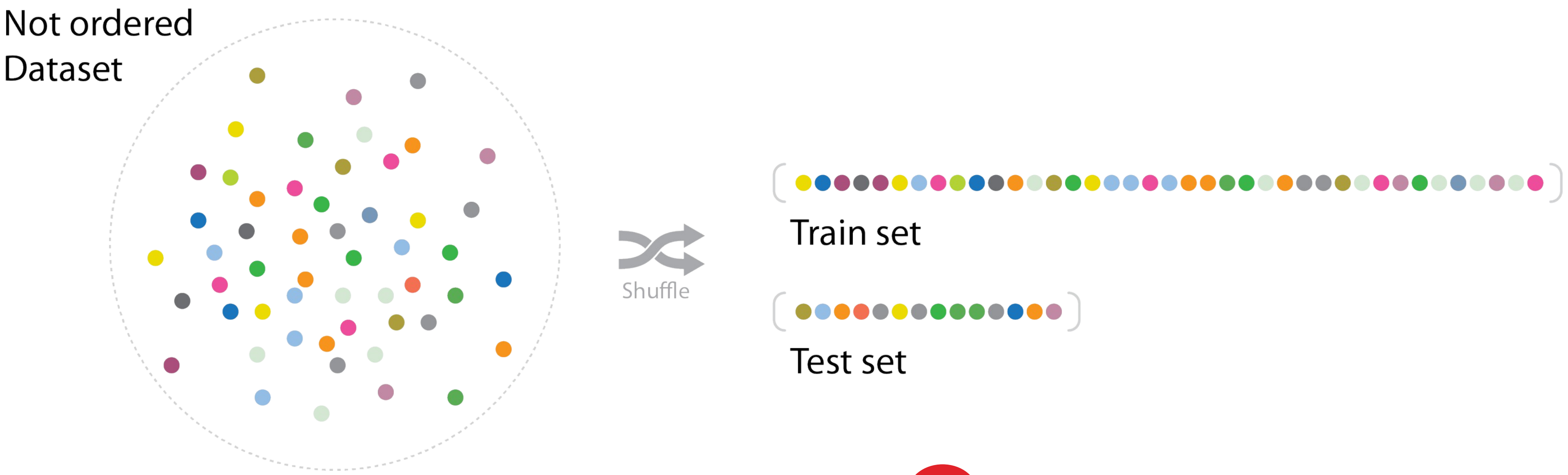
Serie to vector :



Predicted output
que l'on comparera à X_{n+1}

Input sequence
$[X_1, X_2, \ldots, X_{n-1}, X_n]$

Preparation of sequence data



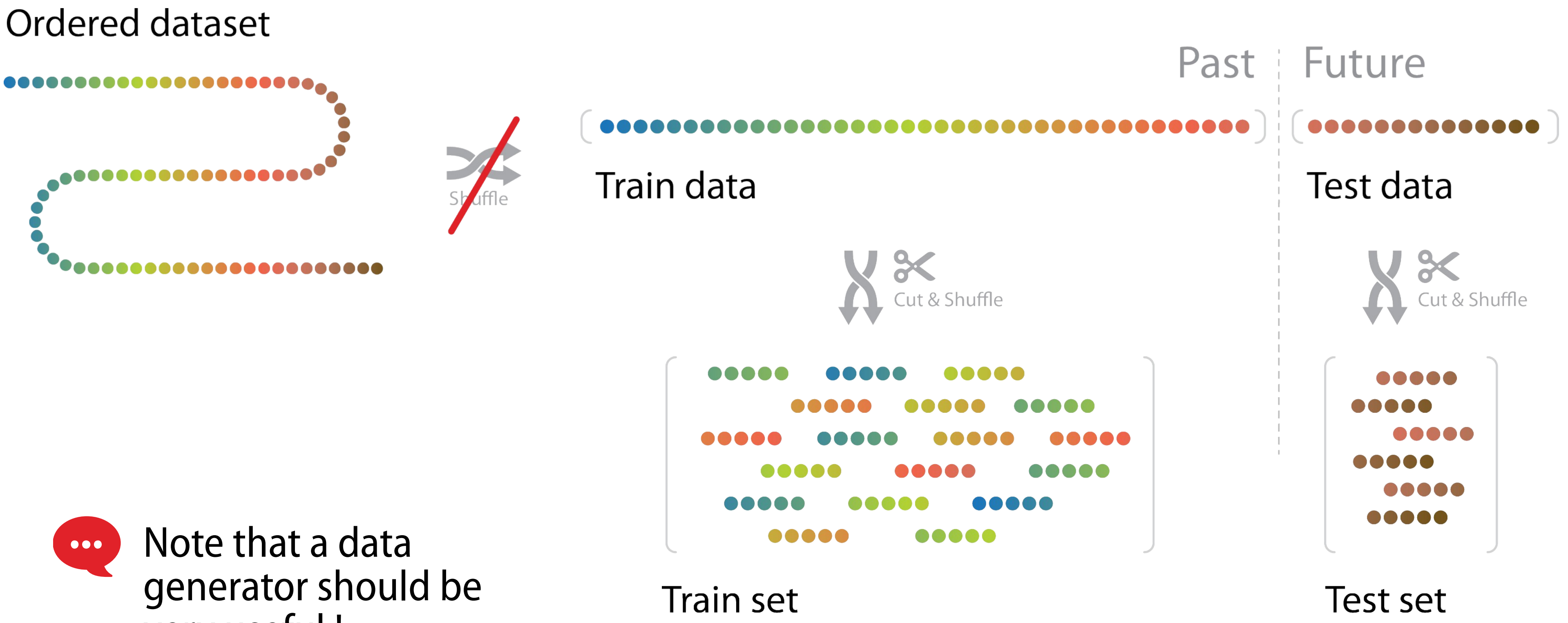
The distribution of data between trains and test series must be comparable.

Preparation of sequence data

Ordered dataset

Can the past explain the future ?

Preparation of sequence data



Note that a data generator should be very useful !

Roadmap

6

Sequences data
RNN

6.1 **Sequences data**

- Recurrent Neural Network
- LSTM and GRU

6.2 **Example 1 : Ladybug1**

- Prediction of a virtual trajectory

6.3 **Example 2 : SYNOP1/3**

- Weather prediction at 3h and 12h

Time series with RNN

Notebook : [LADYB1]

Objective :
Prediction of a 2D ladybug trajectory with a RNN

Dataset :
Generated trajectory

LADYB1

<60 s

train

test

START

Step 1 — Import and init

Step 2 — Generation of a dataset

Step 3 — Create a model

Step 4 — Compile and train

Step 5 — Try a trajectory prediction

END

Objectives :

Prediction of a 2D ladybug* trajectory with a RNN

Parameters

```
sequence_len = 20
predict_len  = 5

scale        = 1
train_prop   = .8
batch_size   = 32
epochs       = 5
```

*Disclaimer : No ladybugs were harmed during the making of this notebook.

Roadmap

6

Sequences data
RNN

6.1 **Sequences data**

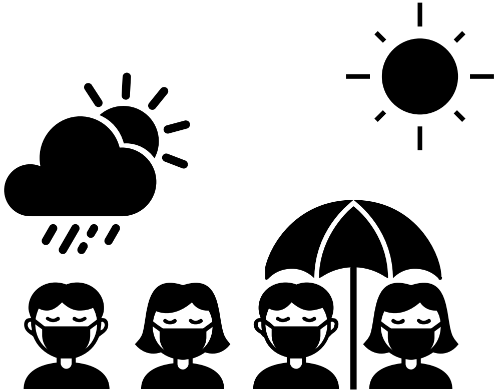
- → Recurrent Neural Network
- → LSTM and GRU

6.2 **Example 1 : Ladybug1**

- → Prediction of a virtual trajectory

6.3 **Example 2 : SYNOP1/3**

- → Weather prediction at 3h and 12h

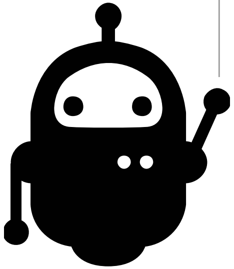


Time series with RNN

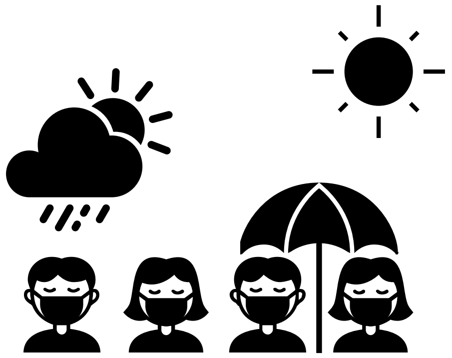
Notebook : [SYNOP1-3]

Objective :
Guess what the weather will be like !

Dataset :
SYNOP meteorological data*.
Data from LYS airport for the period 2010-2020



* Observational data from international surface observation messages (SYNOP) circulating on the World Meteorological Organization's (WMO) Global Telecommunication System (GTS).
https://public.opendatasoft.com/explore/dataset/donnees-synop-essentielles-omm/information/?sort=date



Time series with RNN

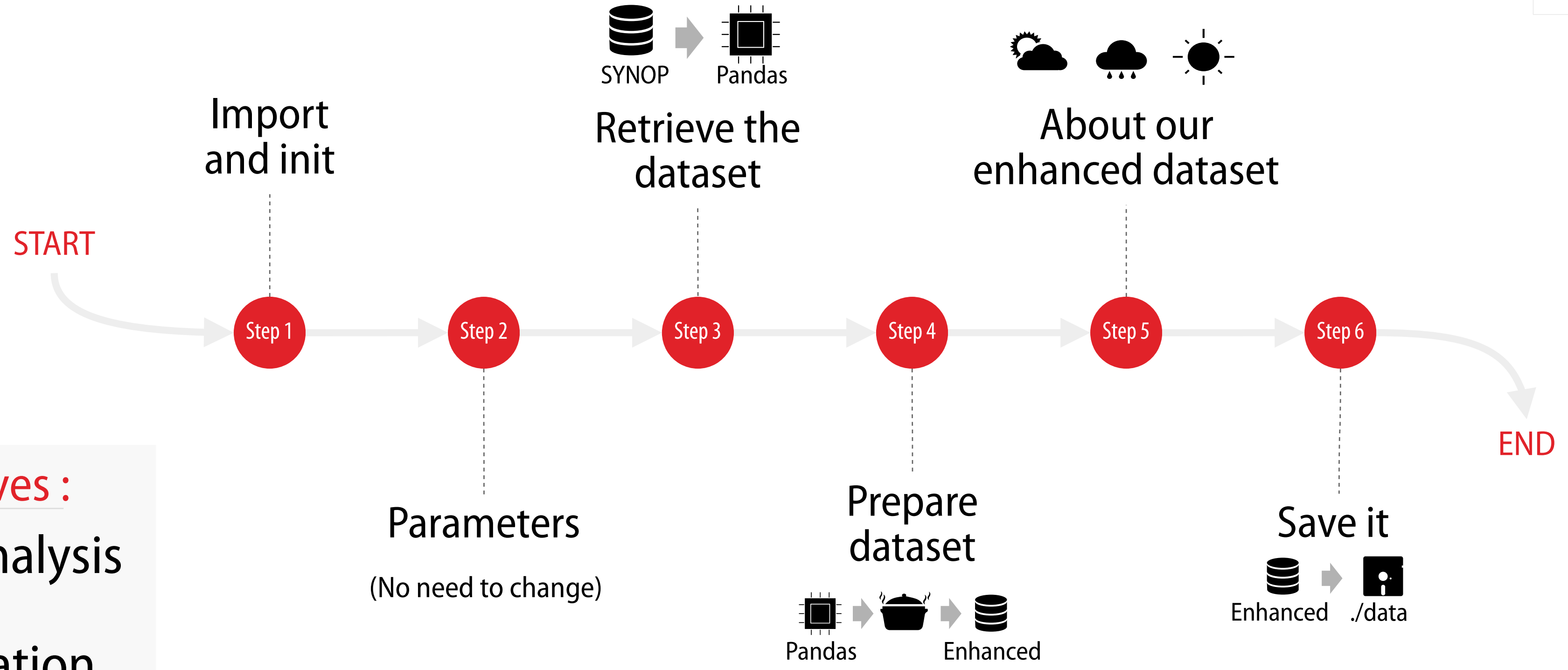
Notebook : [SYNOP1-3]

Episode 1 : Data analysis and creation of a **usable dataset**

Episode 2 : **Training** session and **first predictions**

Episode 3: Attempt to **predict** in the **longer term**

< 3 s !



Objectives :

Data analysis and preparation of a usable dataset

± 1’50s

```
scale         = 1
train_prop    = .8
sequence_len  = 16
batch_size    = 32
epochs        = 10
```

START

Step 1 — Import and init

Step 2 — Read and prepare dataset

./data

train

test

Step 3 — Create a model

Step 4 — Compile and train

Step 5 — Predict

END

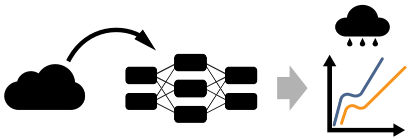
Objectives :

First weather prediction, using LSTM. Attempt at 3h

< 3s !

```
Iterations    = 4
scale         = 1
train_prop    = .8
sequence_len  = 16
batch_size    = 32
epochs        = 10
```

Import and init



Predict

START

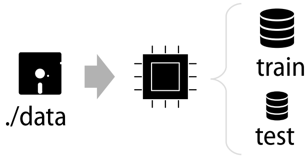
Step 1

Step 2

Step 5

END

Read and prepare dataset



Objectives :

First weather prediction, using LSTM. Attempt at 12h

Little things and concepts to keep in mind

- Can the past explain the future ?
- Understand the data, again and again !
- Beware of overfiting
- Remember that Pandas is good for you !
- The json files are cool, too
- Preparing your data can cost 70% of the work
- Think about data generators
- Matplotlib are also very good for you !
- There is a lot of sequential data
- Not everything can uses GPUs...

Next, on Fidle :

7

PyTorch

A small detour
by **PyTorch**

Jeudi 12 janvier,

Épisode 7 :
Un détour par PyTorch

Présentation générale
Principes et objets clés pour programmer sous PyTorch
Exemples : Classification et régression sous PyTorch

Durée : 2h

Next on Fidle :

7

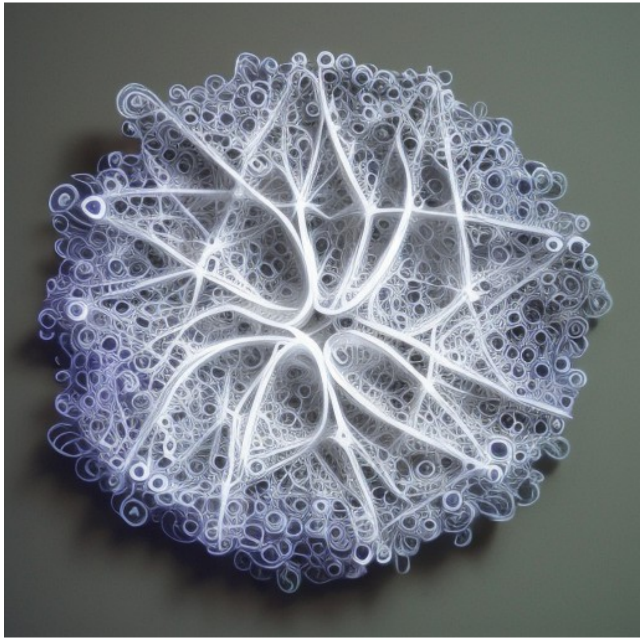
PyTorch

A small detour by **PyTorch**

Jeudi 12 janvier,

Séquence 7 :

Un détour par PyTorch



« A software framework to make neural networks, paper art pins »

To be continued...

Contact@fidle.cnrs.fr
FIDLE https://fidle.cnrs.fr
YouTube https://fidle.cnrs.fr/youtube

FIDLE