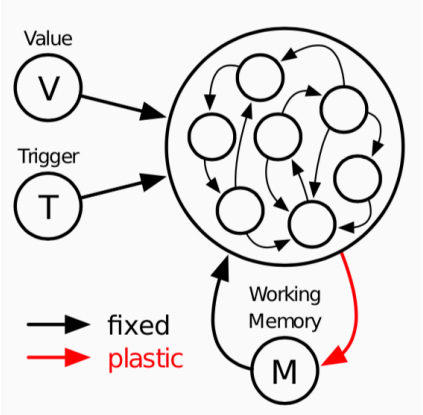
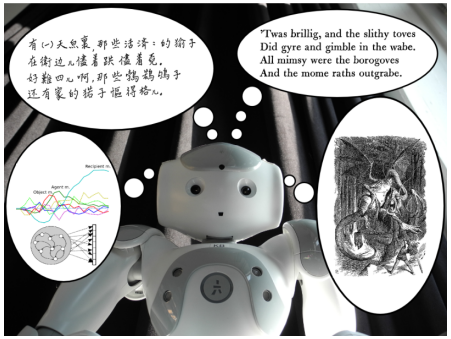
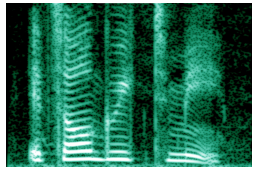






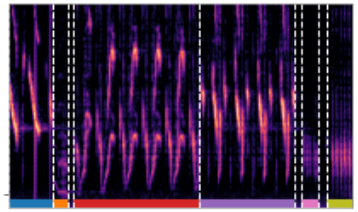
Xavier Hinaut & Paul Bernard

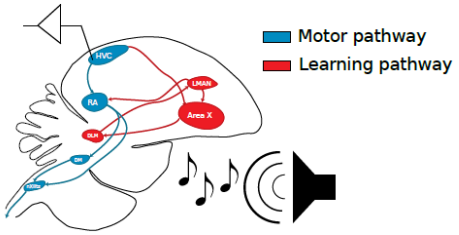
Researcher & Engineer @Inria, Bordeaux, France

{xavier.hinaut / paul.bernard}@inria.fr

@neuronalX & @PAUL-BERNARD

@hinaut @reservoirpy





RÉPUBLIQUE FRANÇAISE
Liberté Égalité Fraternité

Inria LaBRI université de BORDEAUX

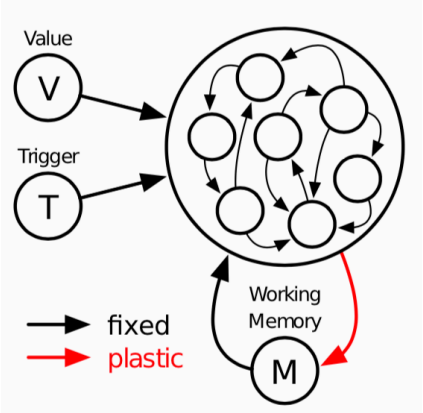
BORDEAUX neurocampus IMN

HACK1 ROBO


HACK1
ROBO
WHERE DID I COME FROM?
WHERE SHOULD I GO?
reservoirpy
RoboCup'23
Mon Fab Lab :
la fabrique à projets

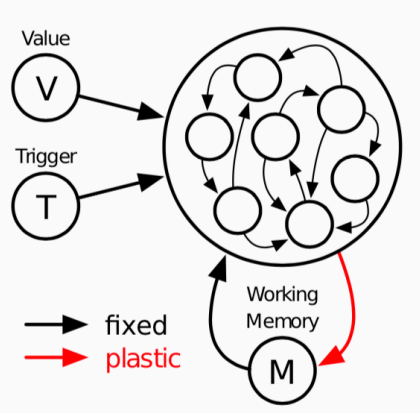
Have you ever dreamed of being a DJ?

Alison Wonderland

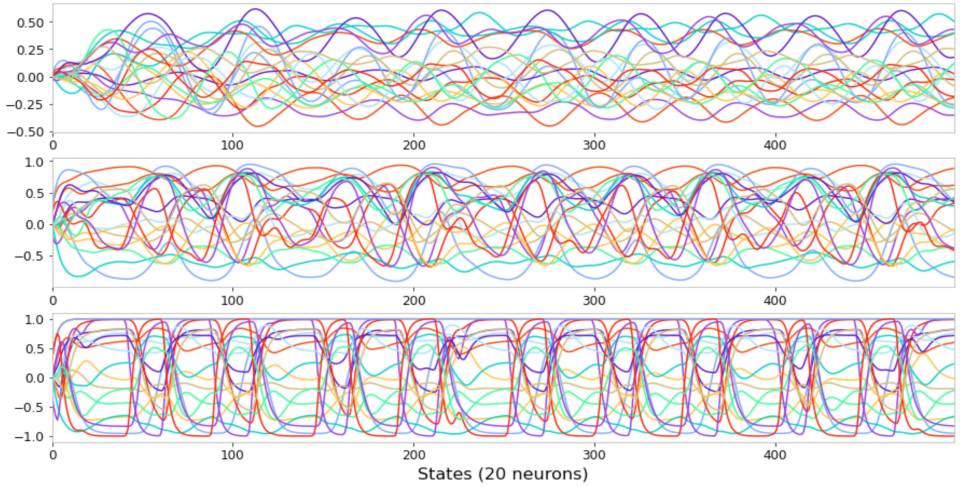


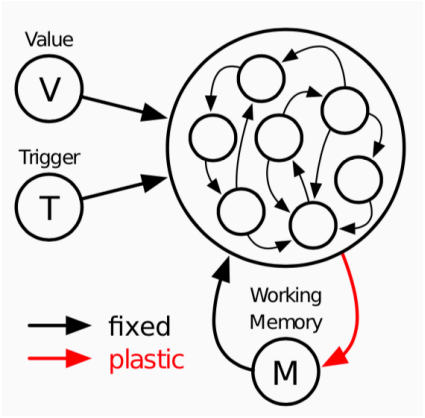
?

Alison Wonderland



```
from reservoirpy.nodes import Reservoir, Ridge, Input

data = Input(input_dim=1)
reservoir = Reservoir(100, lr=0.3, sr=1.1)
readout = Ridge(ridge=1e-6)

esn = data >> reservoir >> readout

forecast = esn.fit(X, y).run(timeseries)
```

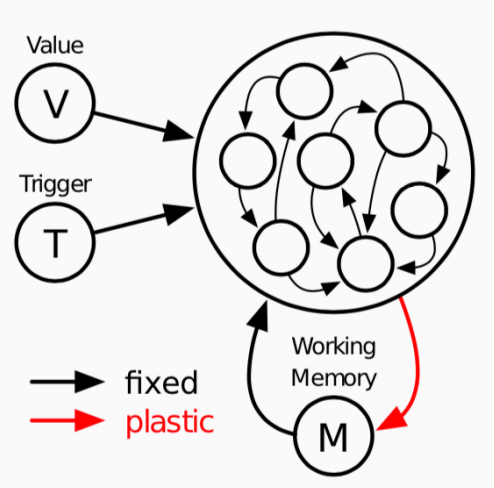


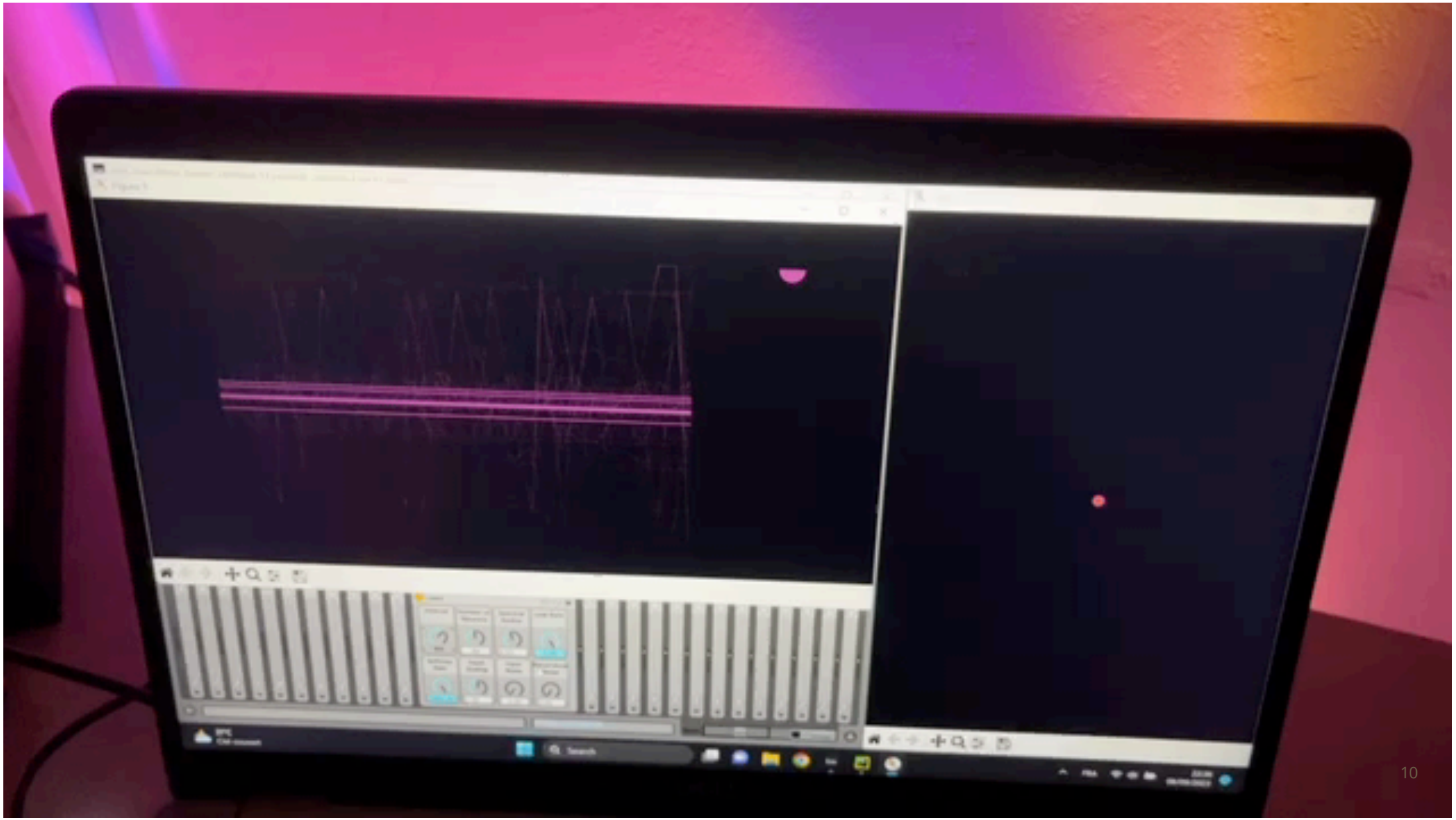

Value
V
Trigger
T
Working
Memory
M
fixed
plastic


SEQUENTIAL

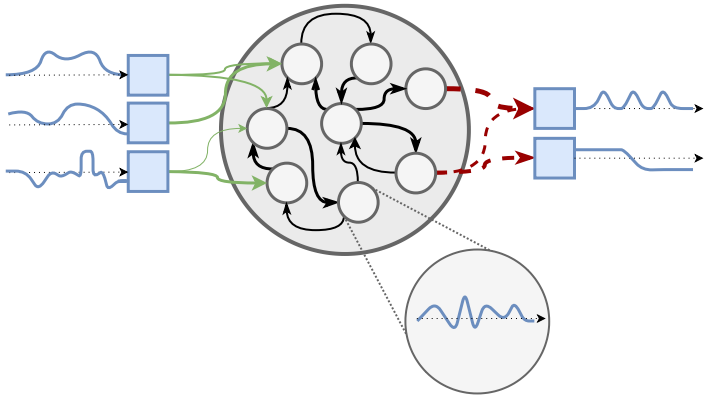

SEQUENTIAL

Have you ever dreamed of playing with a reservoir?

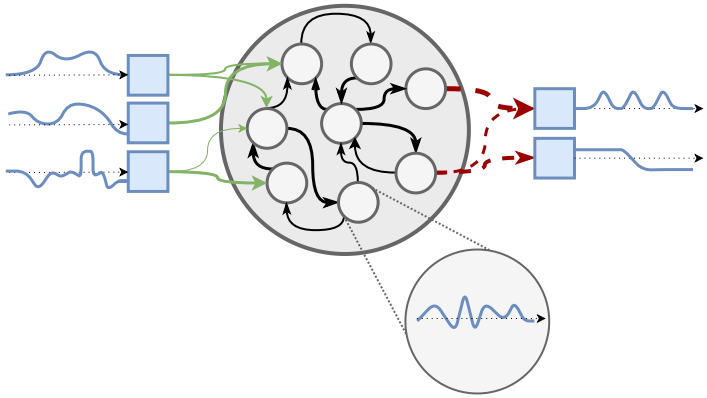




Random networks can generate fascinating dynamics

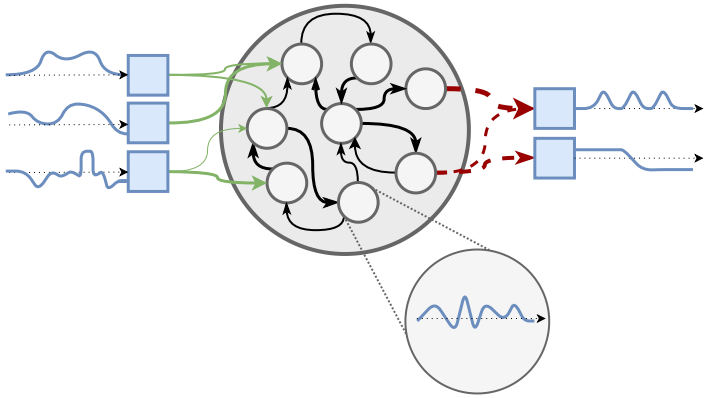


Random networks can generate fascinating dynamics

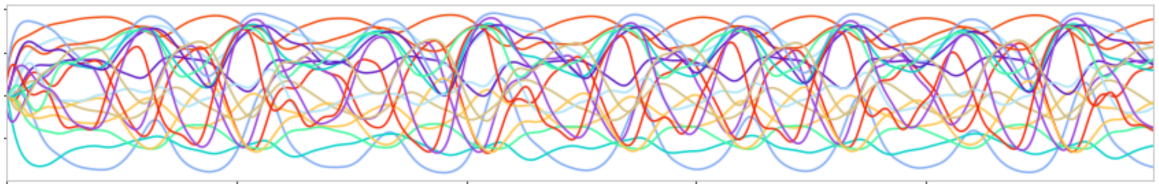


Reservoir Computing exploits
natural dynamics of random networks

Random networks can generate fascinating dynamics



Reservoir Computing exploits
natural dynamics of random networks

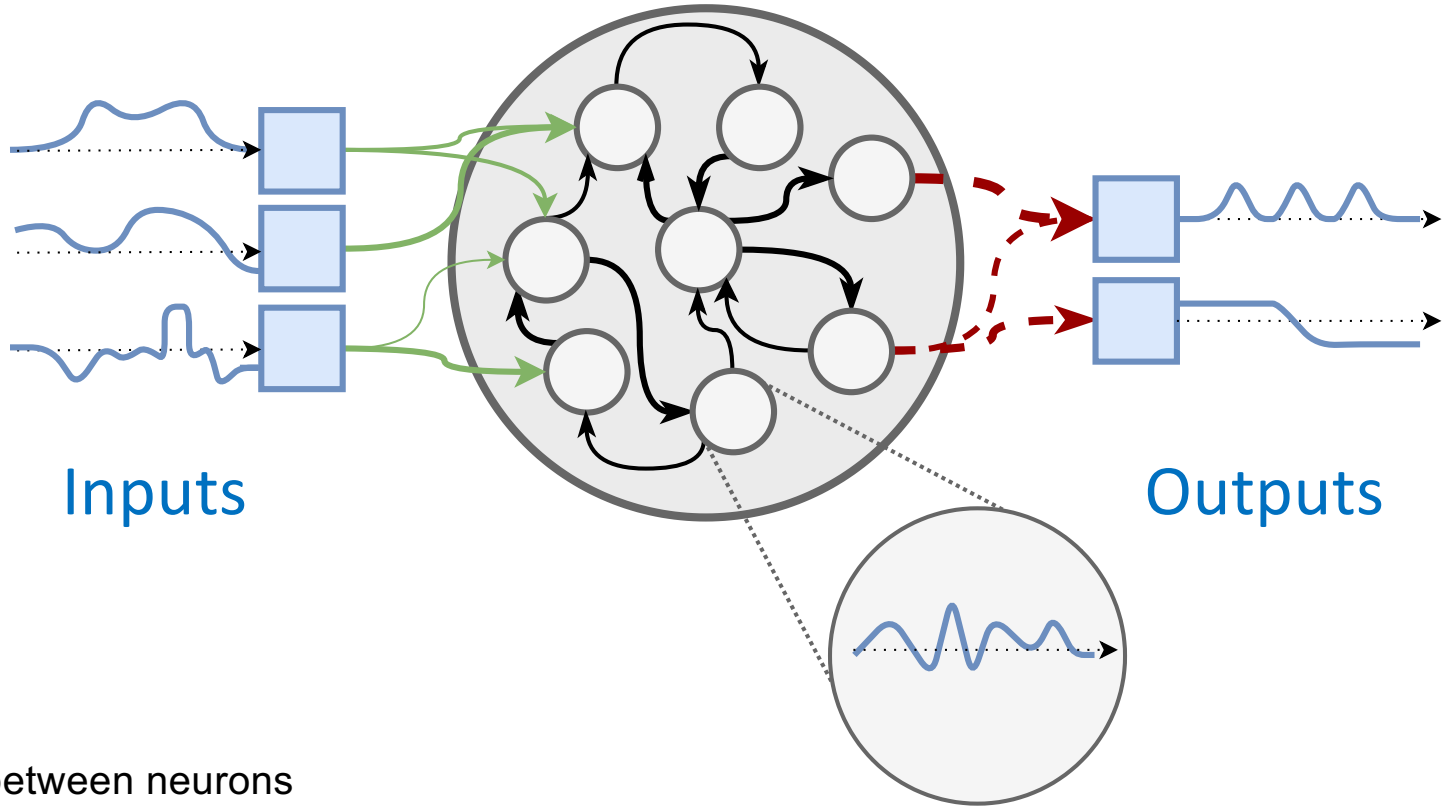


Quick installation for ReservoirPy demo

https://github.com/reservoirpy/reservoirpy

pip install reservoirpy[hyper]

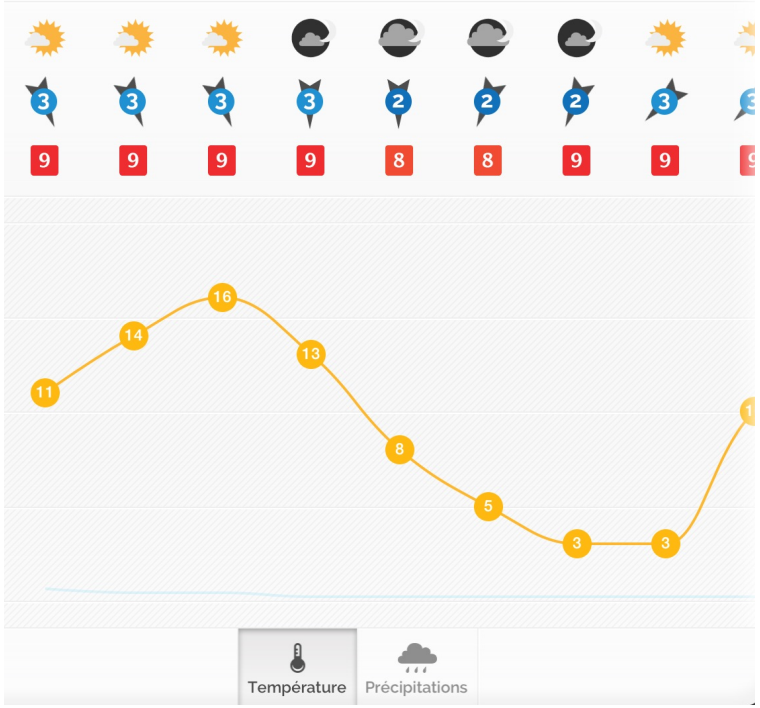
Reservoir Computing



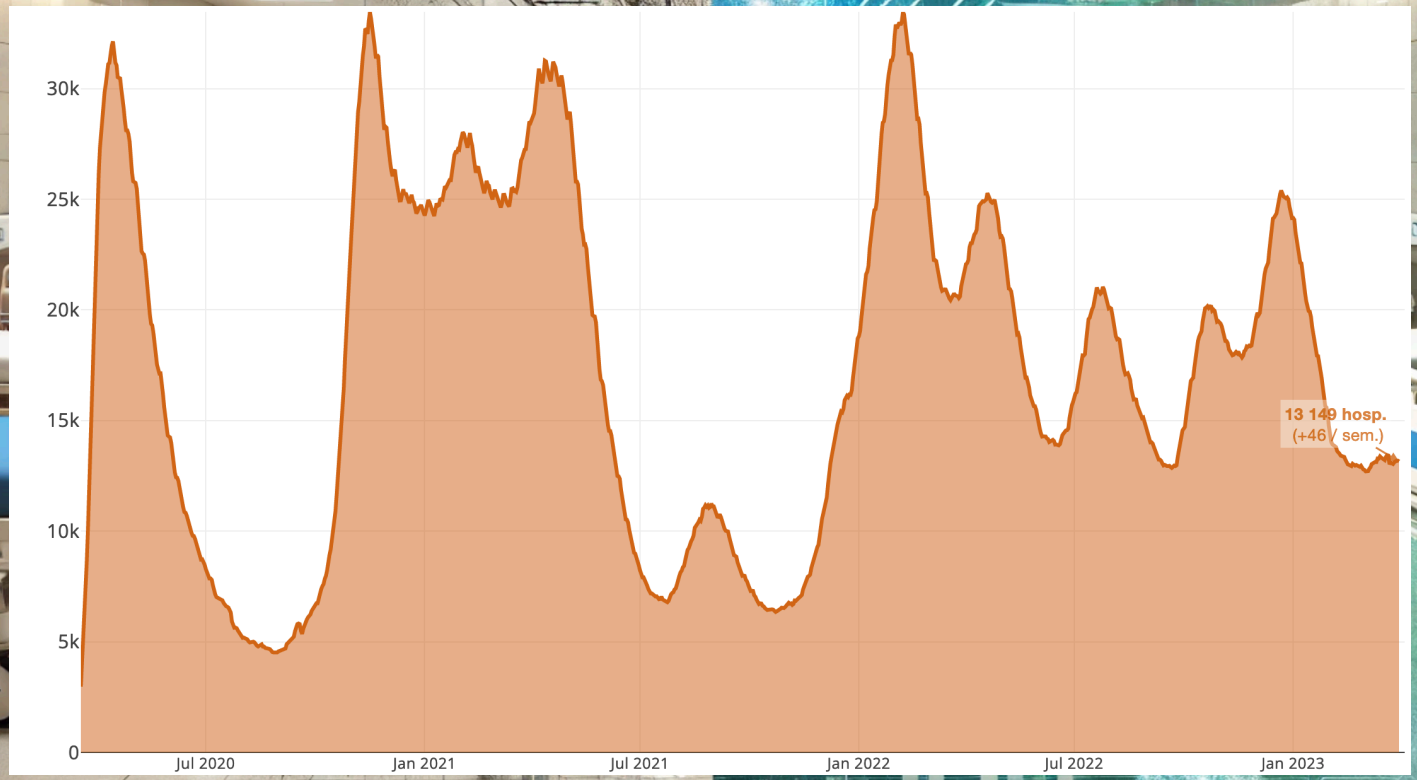
Neuron

Connection between neurons

Time series prediction



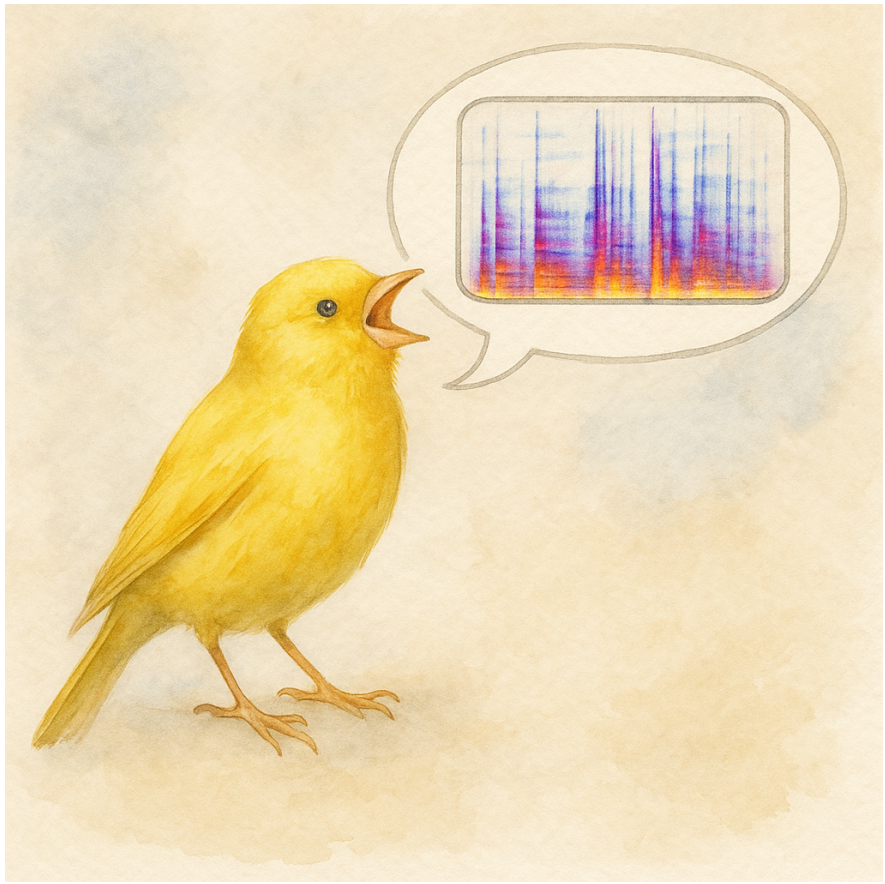
2 week forecasting of COVID hospitalisations

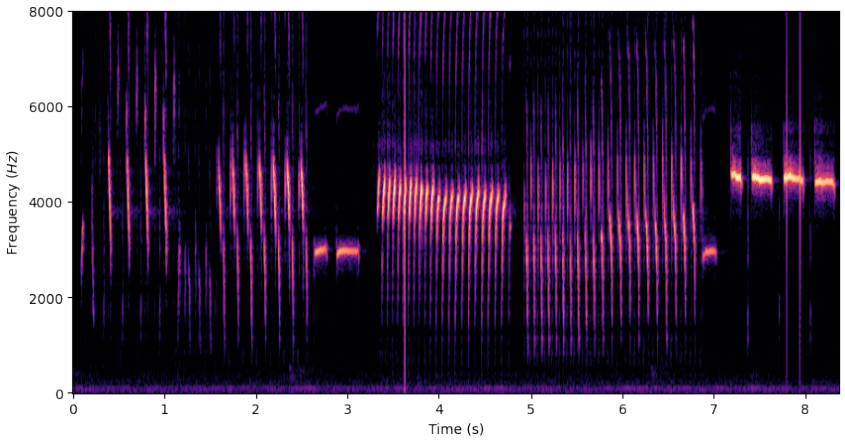


https://covidtracker.fr/france/#hospitalisations

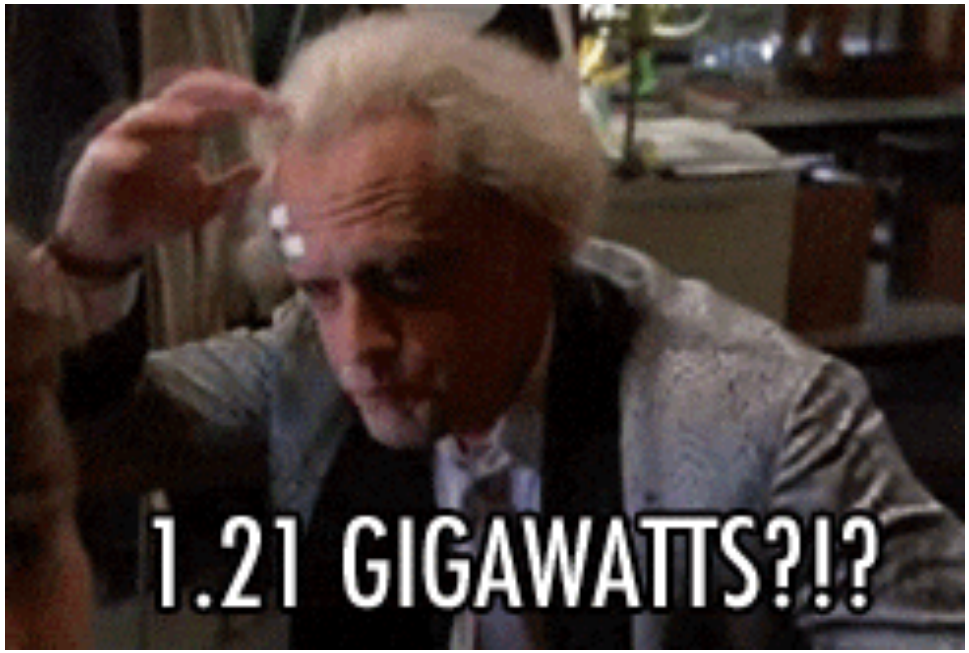
Catégoriser des sons

(“discriminer” des syllabes de canari)

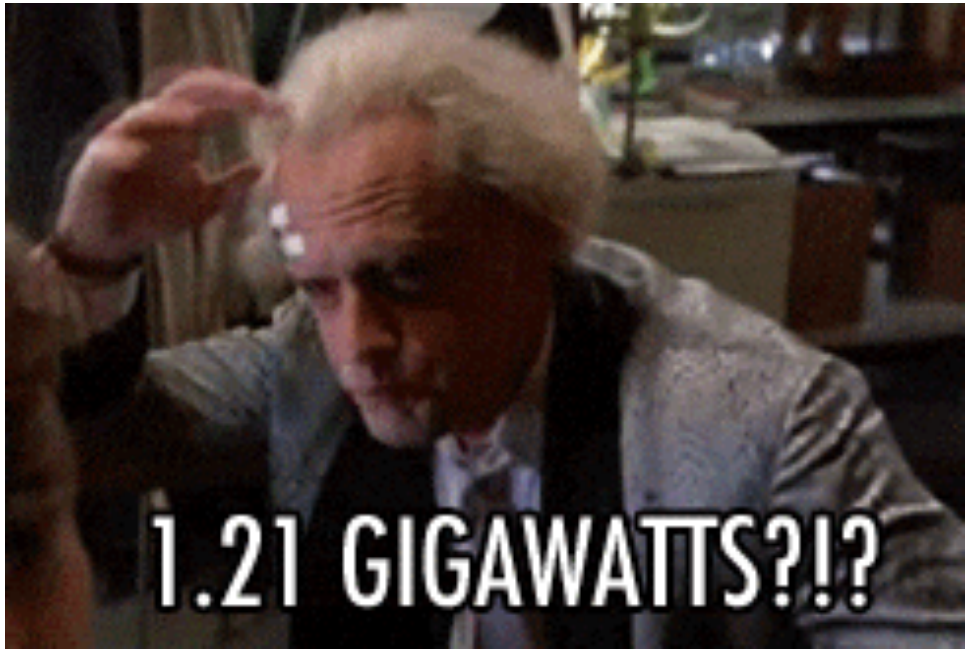


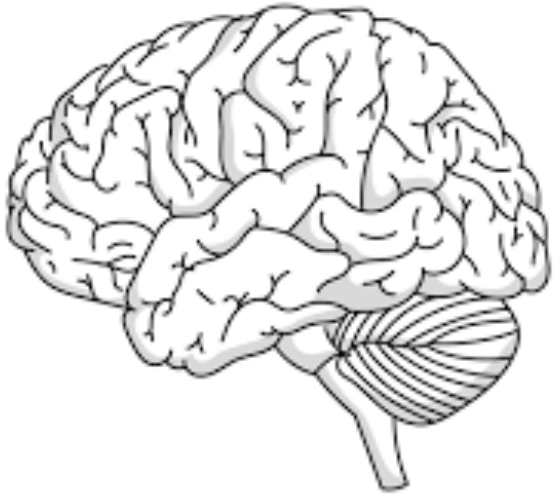


Training a recurrent network is expensive!



Training a recurrent network is expensive!

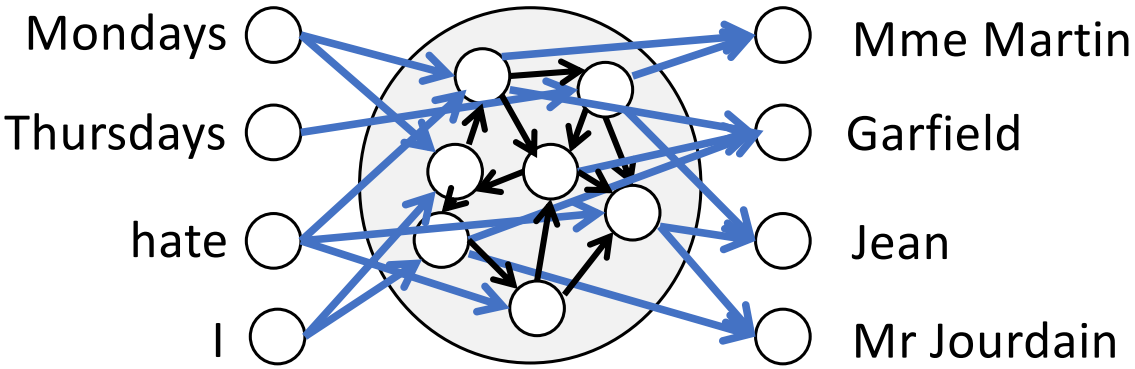




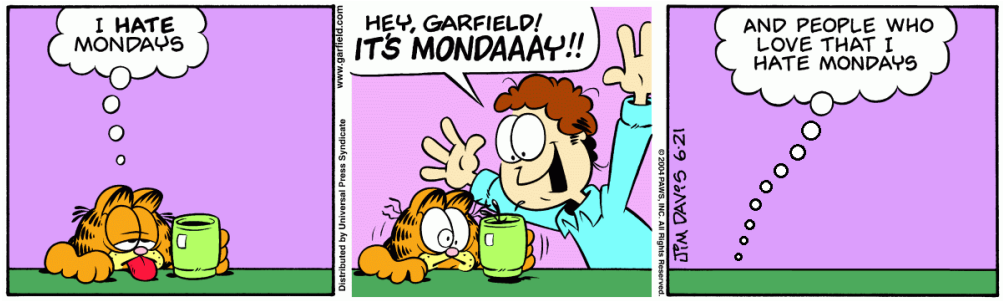
Only 20 W !
(2 energy-saving light bulbs)

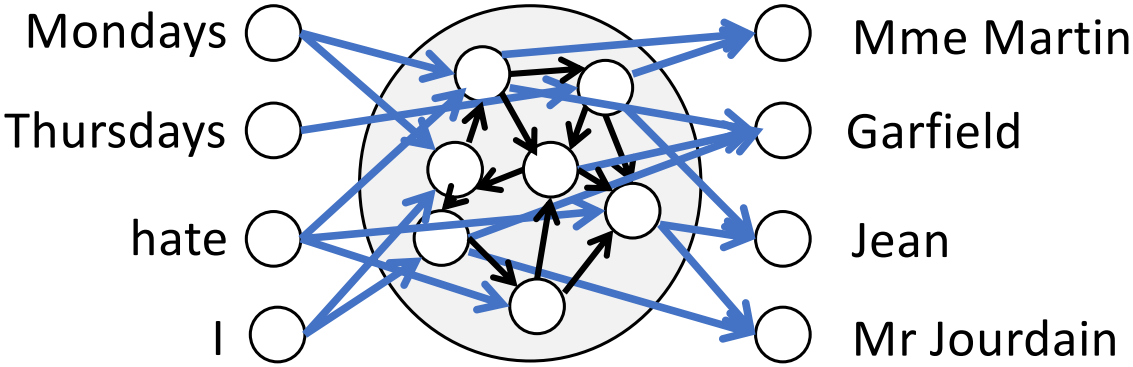
-> How to balance performance and energy?

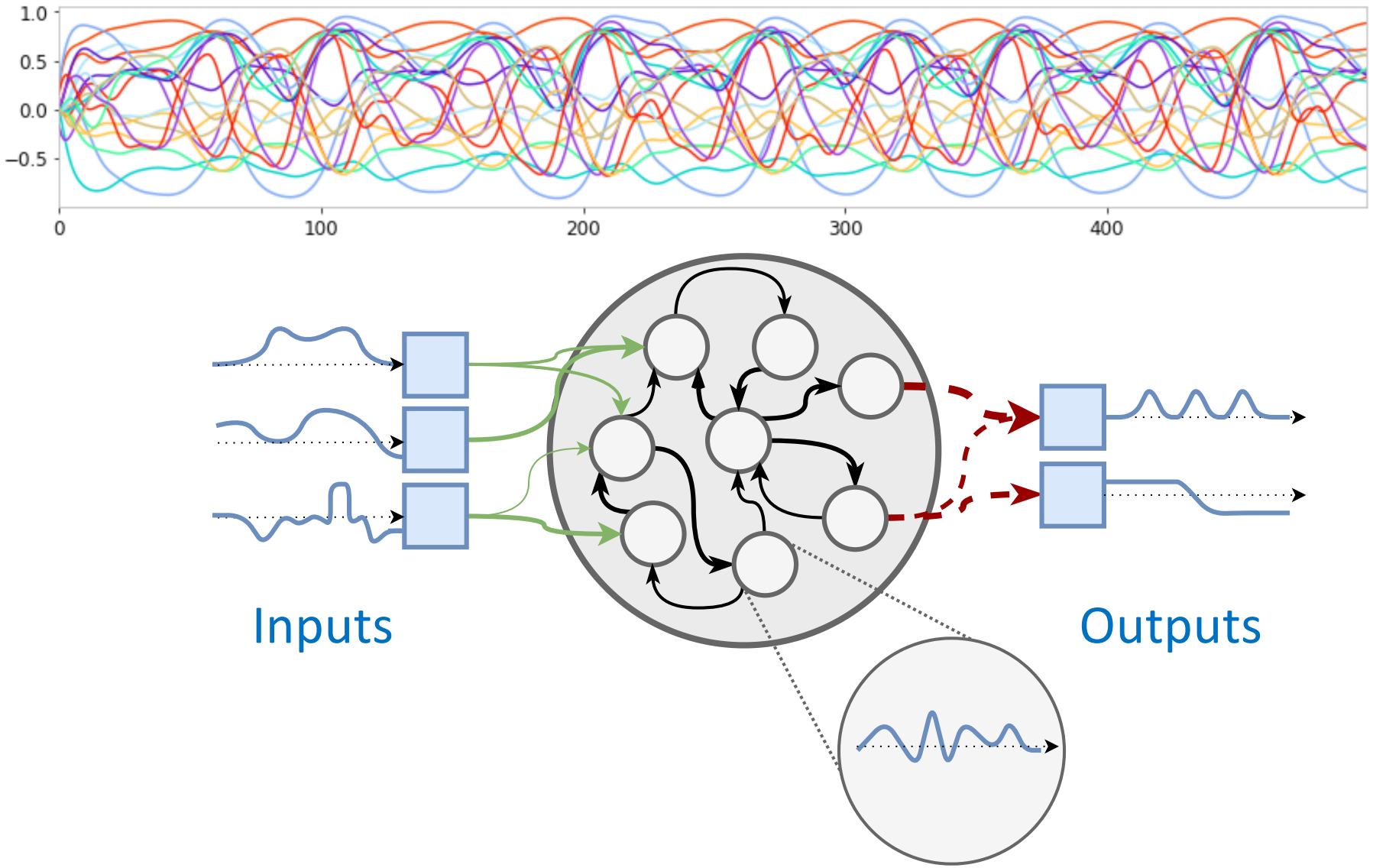
Reservoir Computing exploits natural dynamics of a recurrent neural network



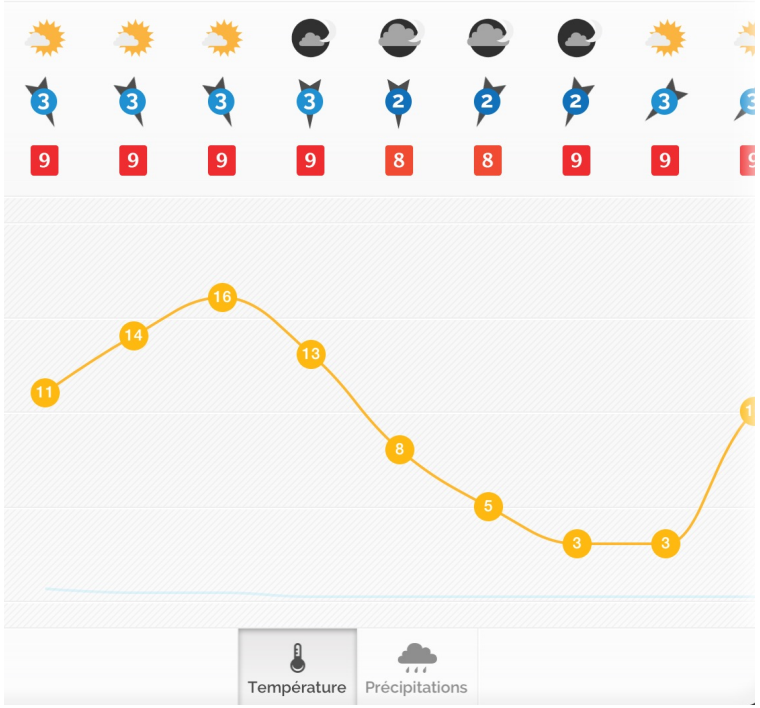
Reservoir Computing exploits natural dynamics of a recurrent neural network



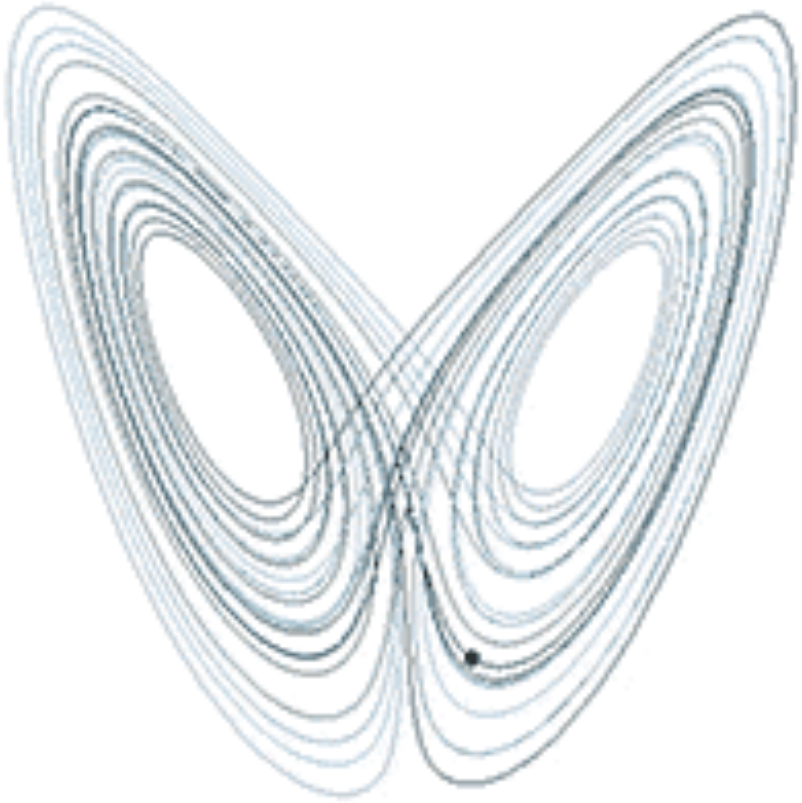



1.0
0.5
0.0
−0.5
0
100
200
300
400
Inputs
Outputs

Time series prediction



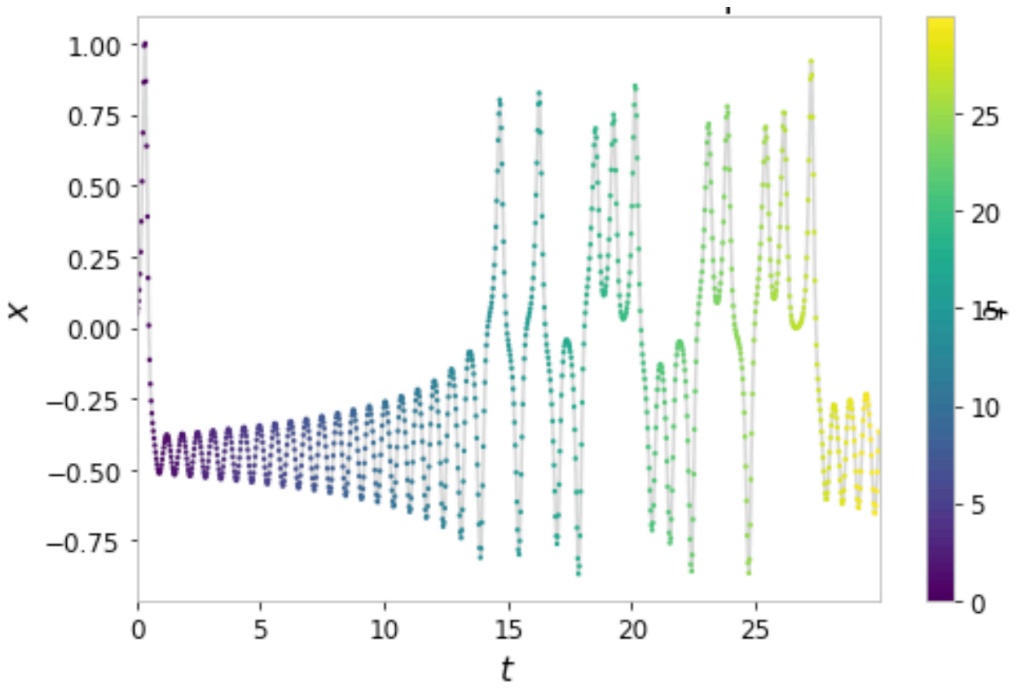
Lorenz chaotic strange attractor

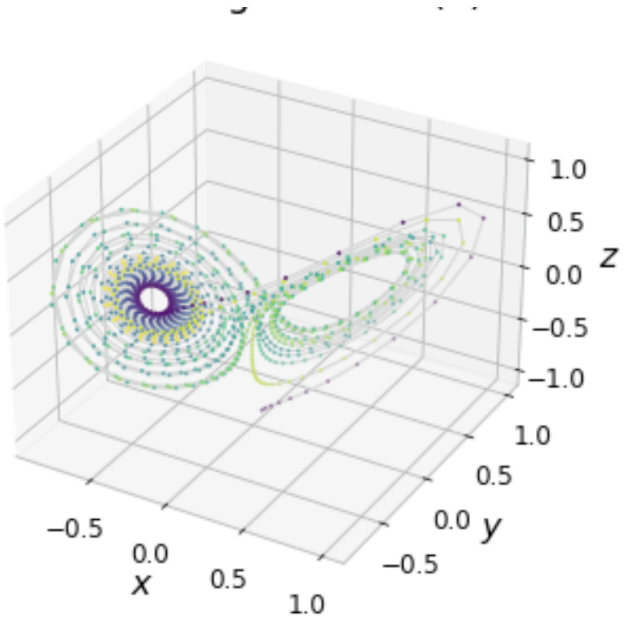


Lorenz chaotic strange attractor

- We try to predict one of the values of the Lorenz dynamical system defined by 3 variables (x, y, z) that change over time

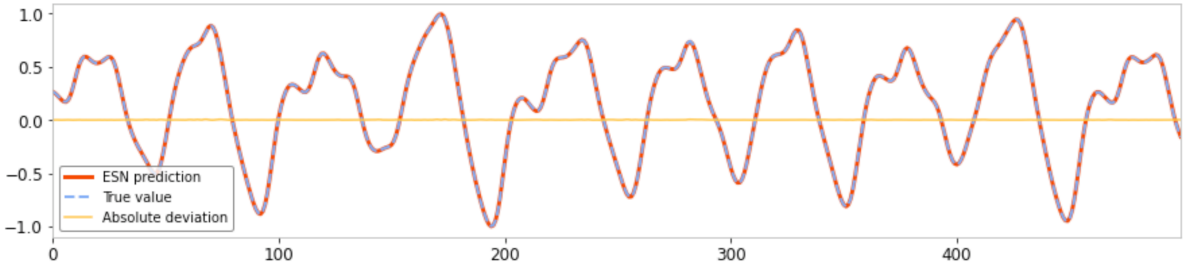
Here we want to predict x across time



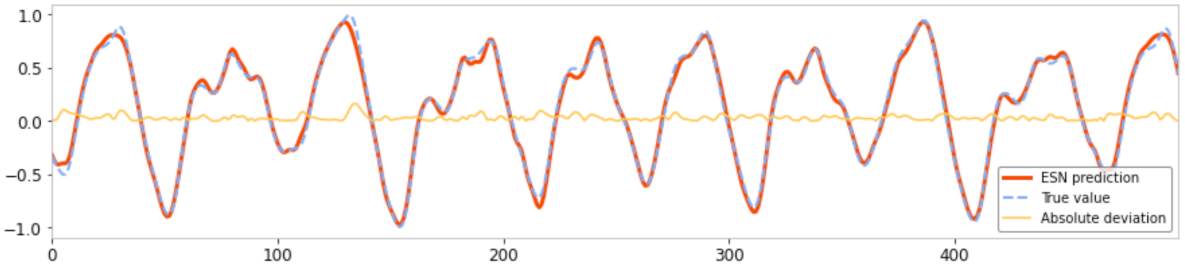


Time series prediction

– Predictions obtained for 10 steps ahead (T+10)



- Predictions obtained for 10 steps ahead (T+50)



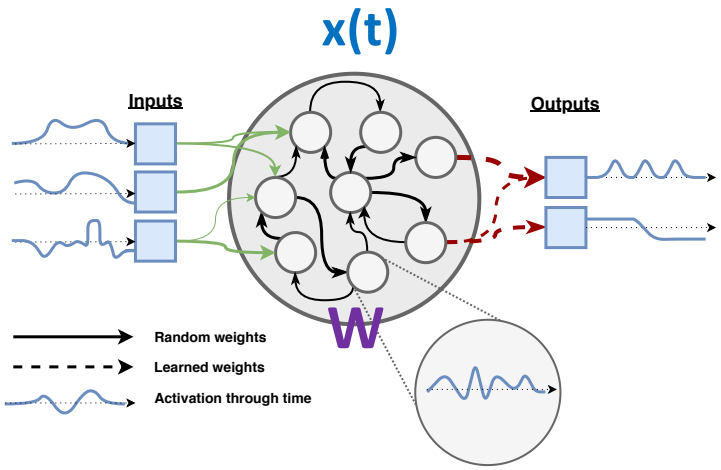
Why does it work?

Why does it work?

Why does it work?

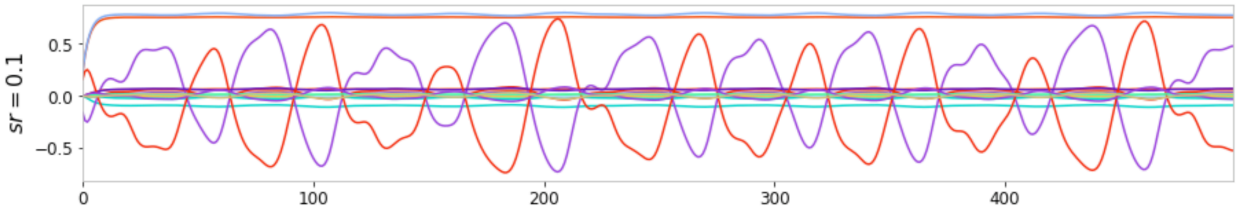
Spectral Radius (SR)

= scaling of recurrent connections

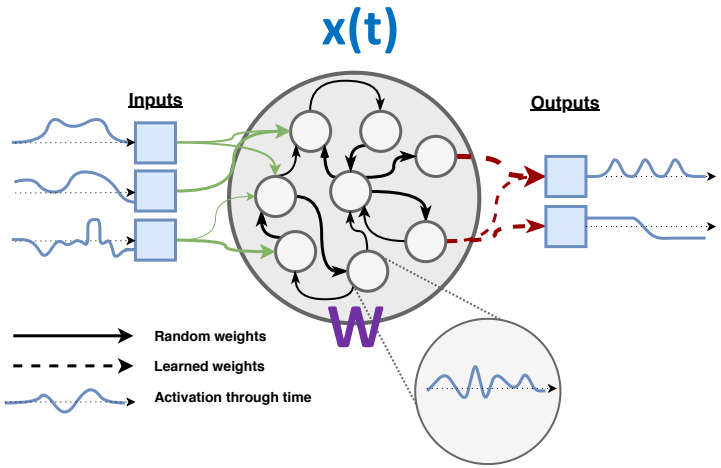


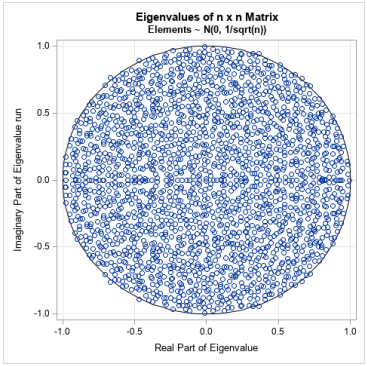
Spectral Radius (SR)

= scaling of recurrent connections



x(t)



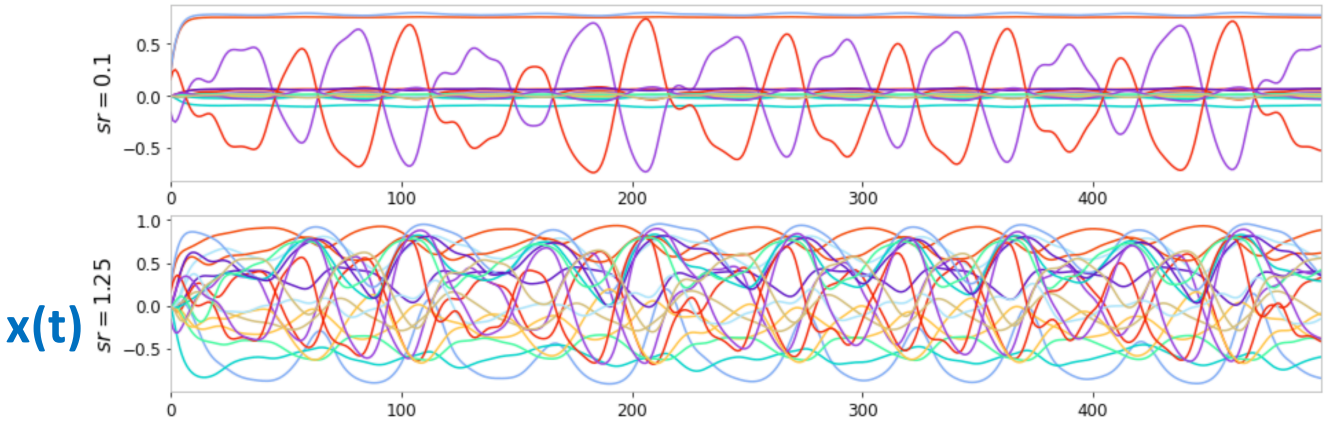


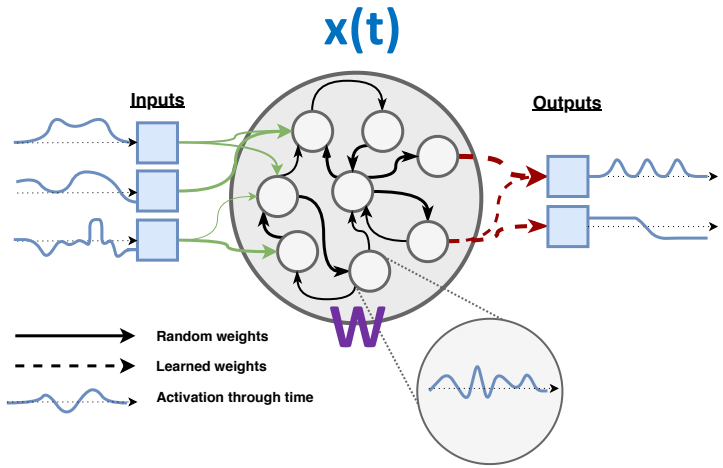
Rayon spectral de la matrice W

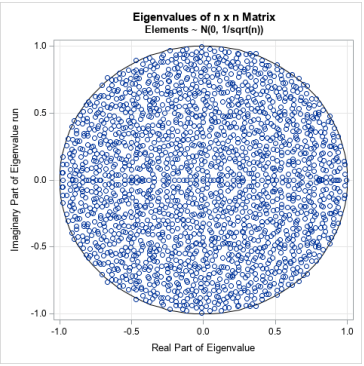
https://blogs.sas.com/content/iml/2020/04/27/circular-law-eigenvalues.html

Spectral Radius (SR)

= scaling of recurrent connections





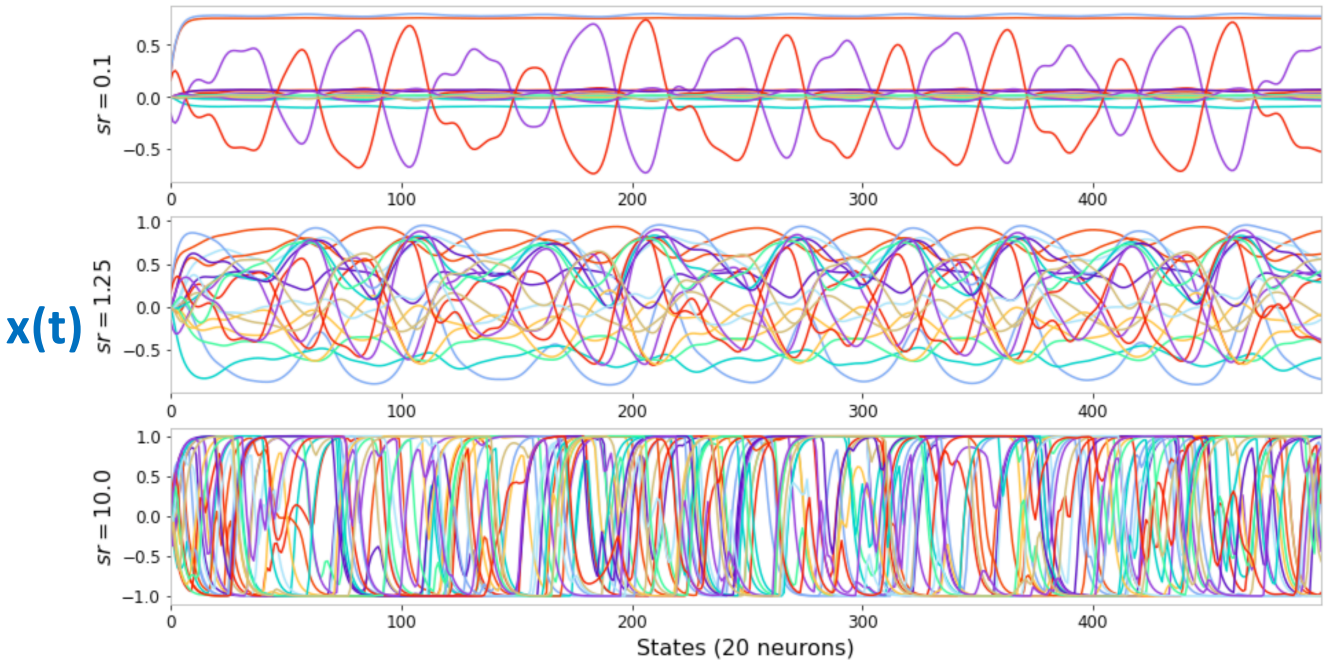


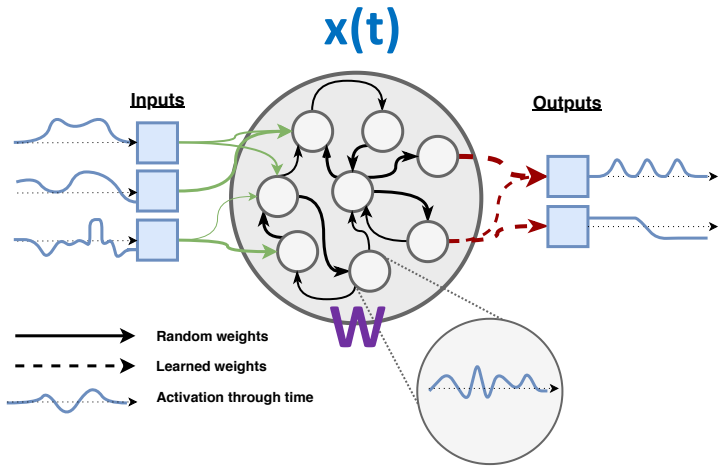
Rayon spectral de la matrice W

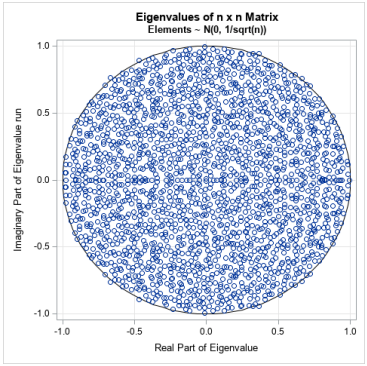
https://blogs.sas.com/content/iml/2020/04/27/circular-law-eigenvalues.html

Spectral Radius (SR)

= scaling of recurrent connections





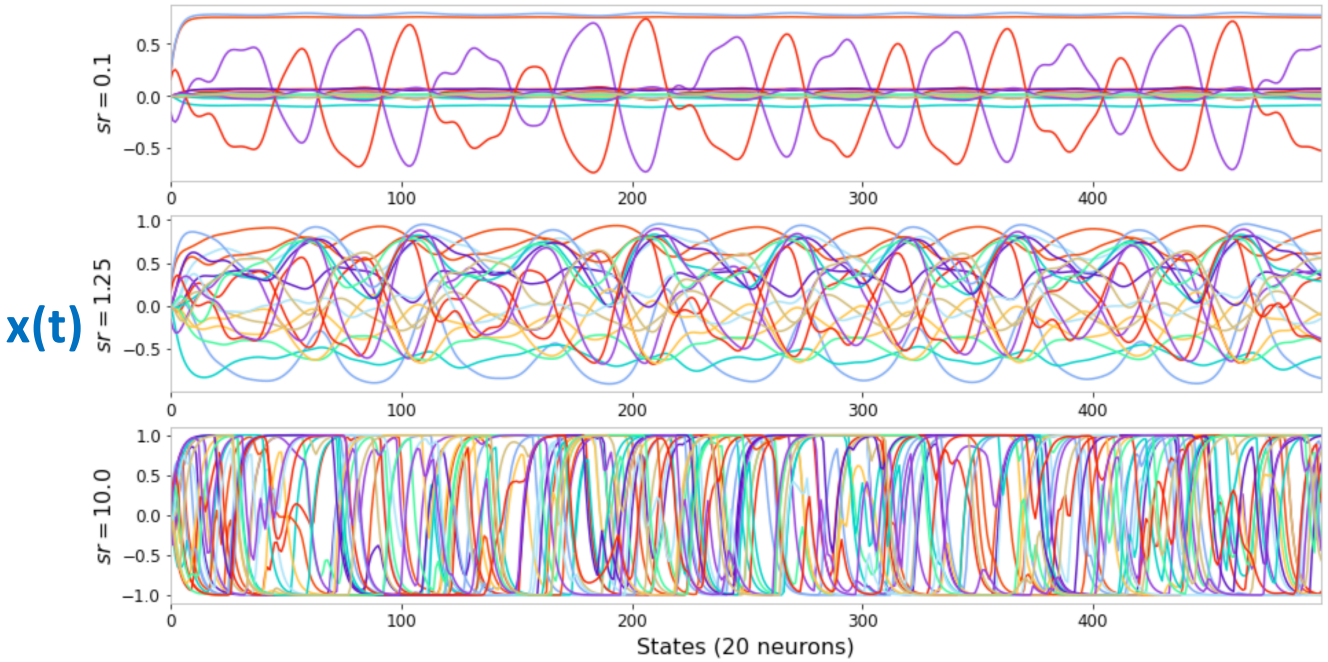


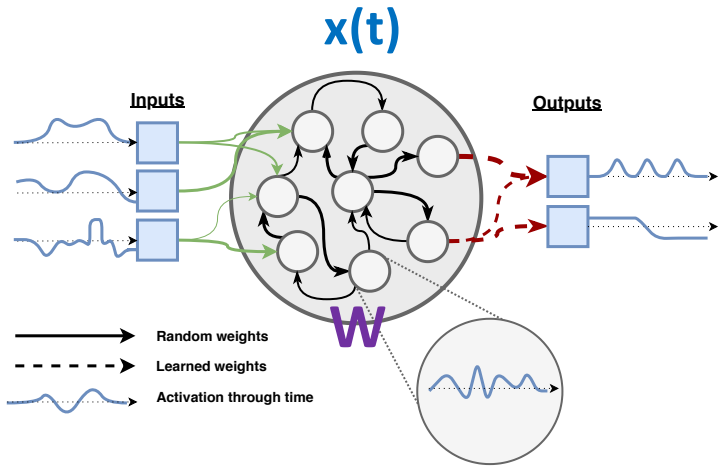
Rayon spectral de la matrice W

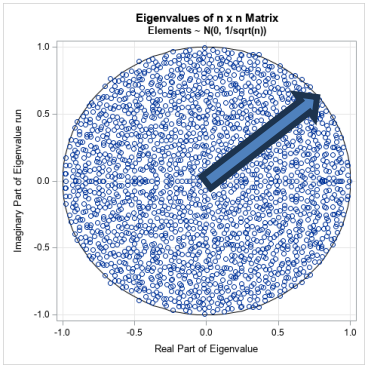
https://blogs.sas.com/content/iml/2020/04/27/circular-law-eigenvalues.html

Spectral Radius (SR)

= scaling of recurrent connections





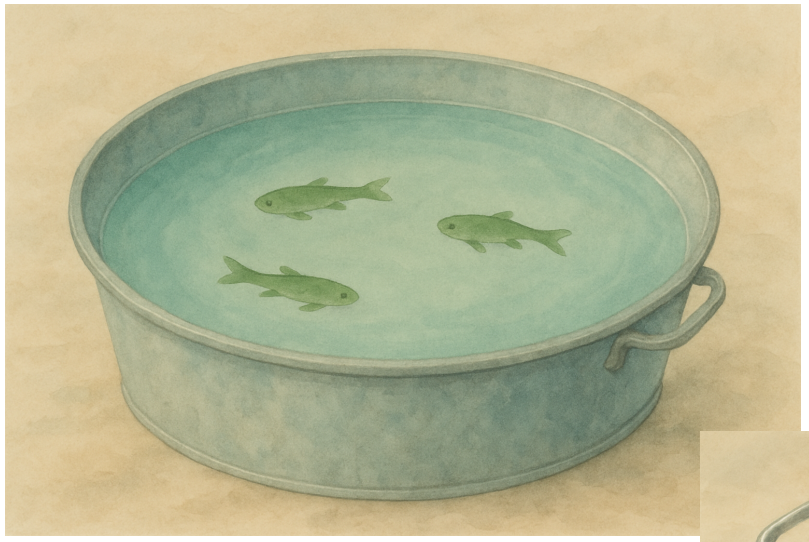


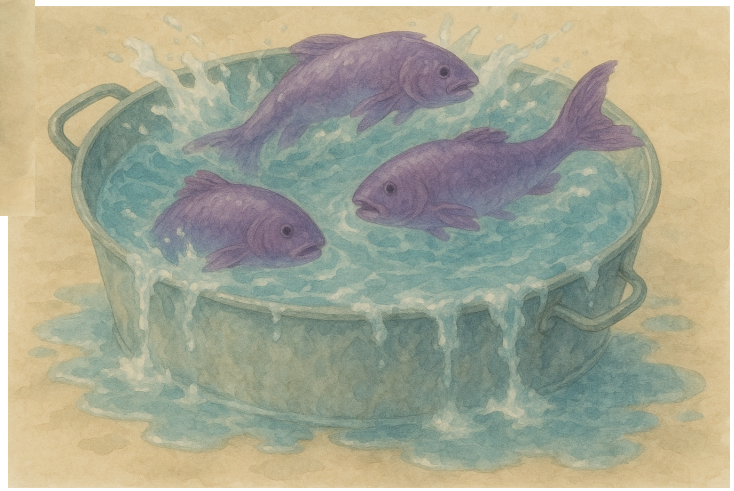
Rayon spectral de la matrice W

— spectral radius → **stable** dynamics

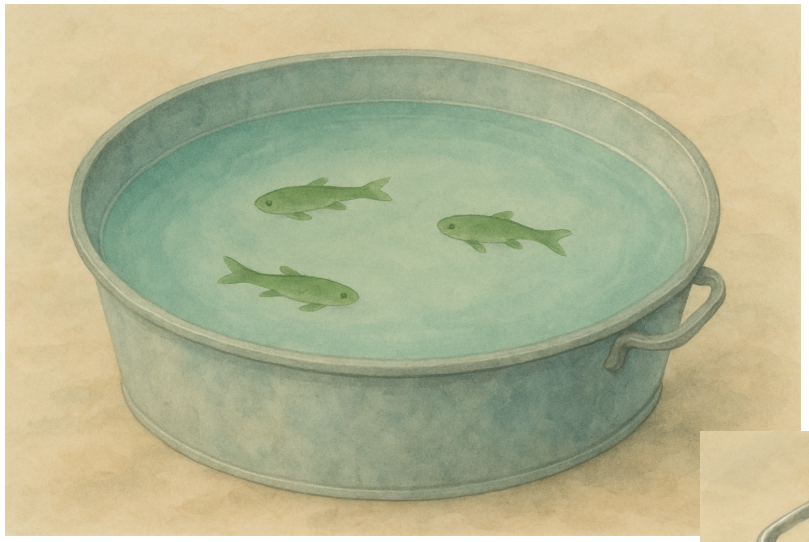
+ spectral radius → **chaotic** dynamics

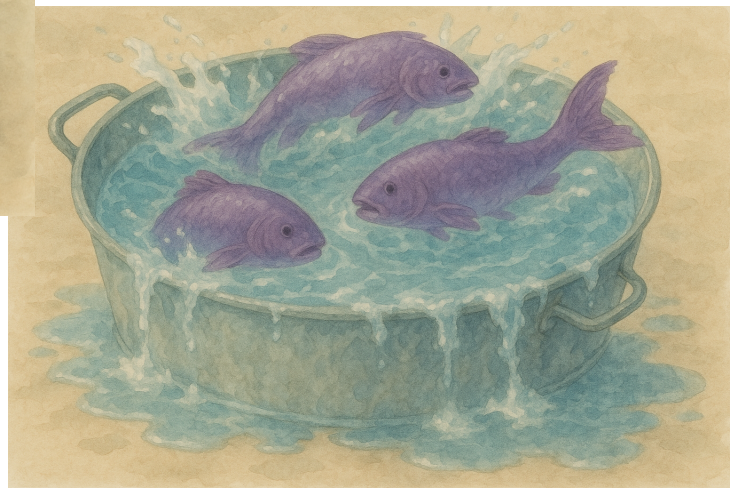
https://blogs.sas.com/content/iml/2020/04/27/circular-law-eigenvalues.html





and in practice?
(are there other
“Hypster-parameters”?)



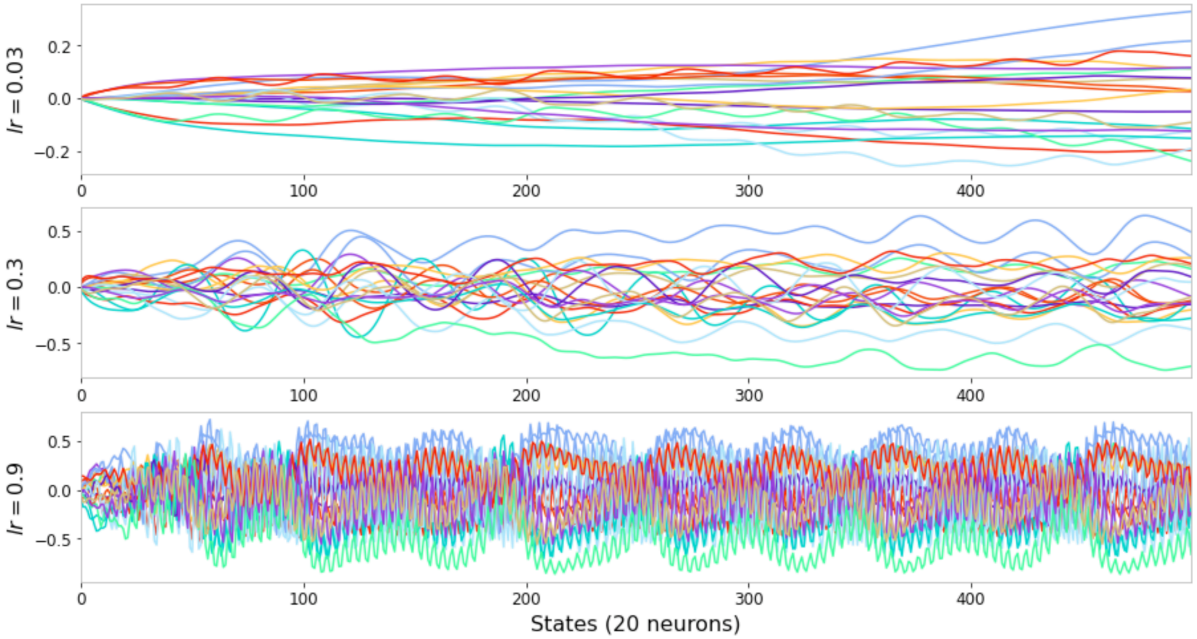


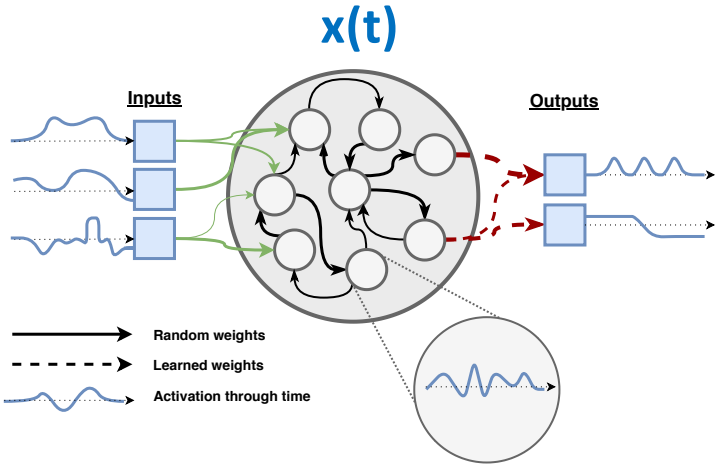
For instance, are there other hyperparameters that could “tame chaos”?

Leak(ing)-rate (LR)

= inverse of a time constant

x(t)





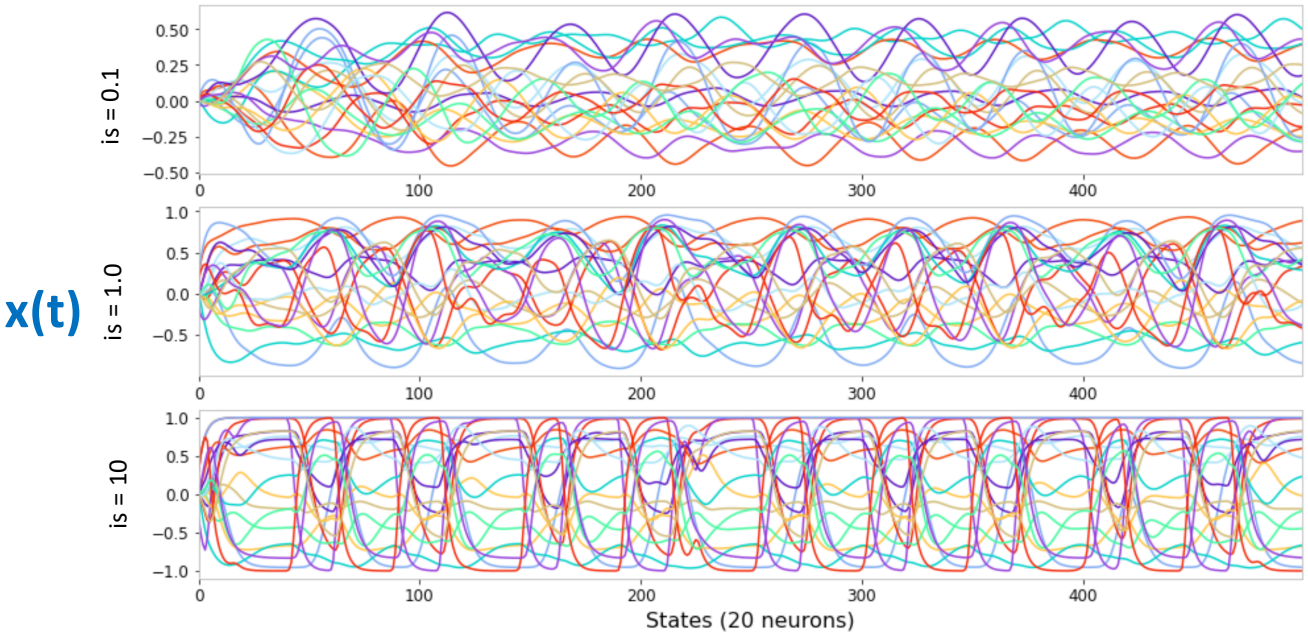
The leaking rate (α) controls the "memory feedback" of the ESN. The ESN states are indeed computed as:

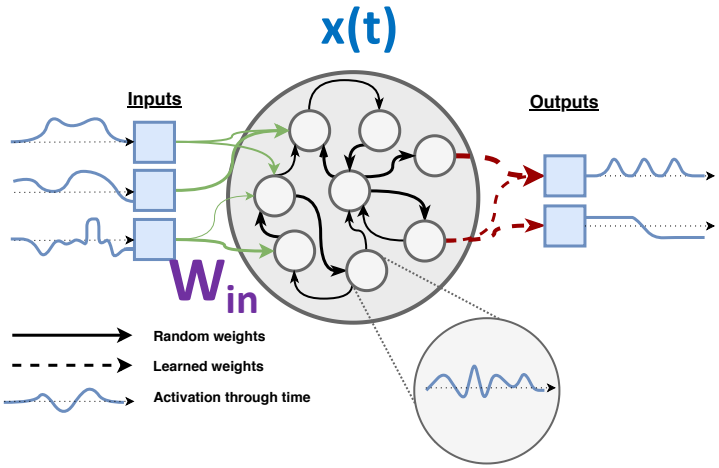
$$s(t+1) = \underbrace{(1-\alpha)s(t)}_{\text{previous states}} + \underbrace{\alpha f(u(t+1), s(t))}_{\text{new states}}$$

- $+$ leaking rate $\rightarrow$ **low inertia**, little memory of previous states
- $-$ leaking rate $\rightarrow$ **high inertia**, big memory of previous states

Input Scaling (IS)

= scaling of input connections





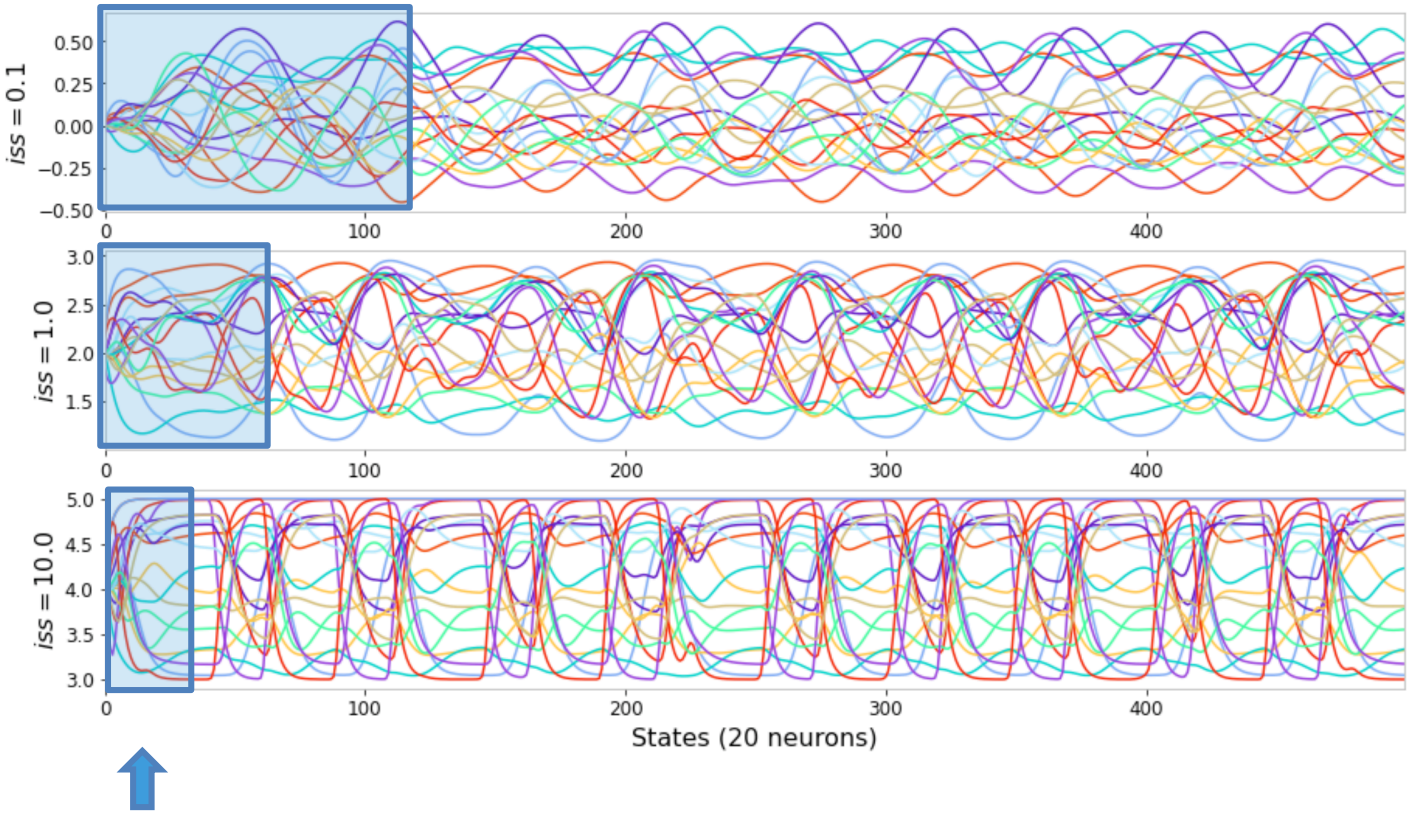
The input scaling controls how the ESN interact with the inputs. It is a coefficient appliyed to the input matrix W_{in}.

- $+$ input scaling $\rightarrow$ **input-driven** activities
- $-$ input scaling $\rightarrow$ **free** activities

The input scaling can also be used to rescale the inputs and ajust their influences.

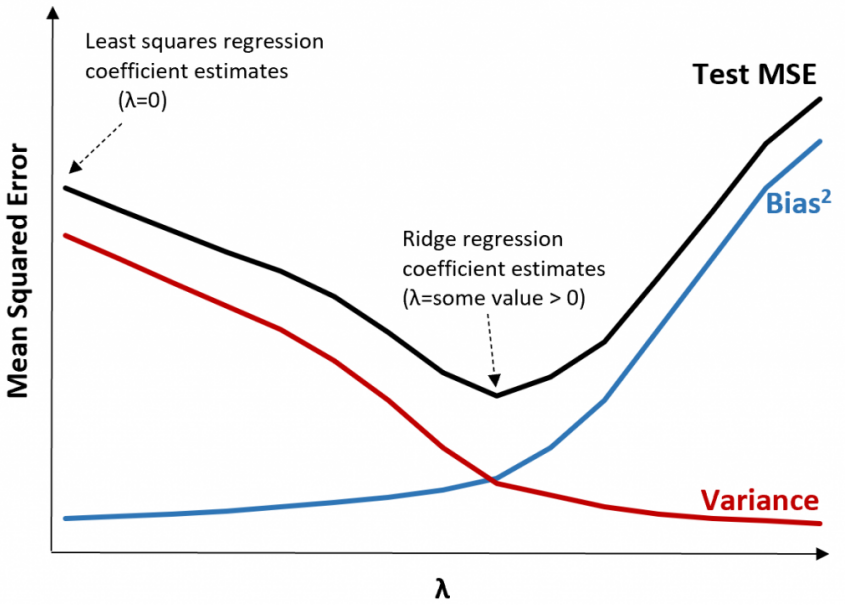
Warm(ing)-up

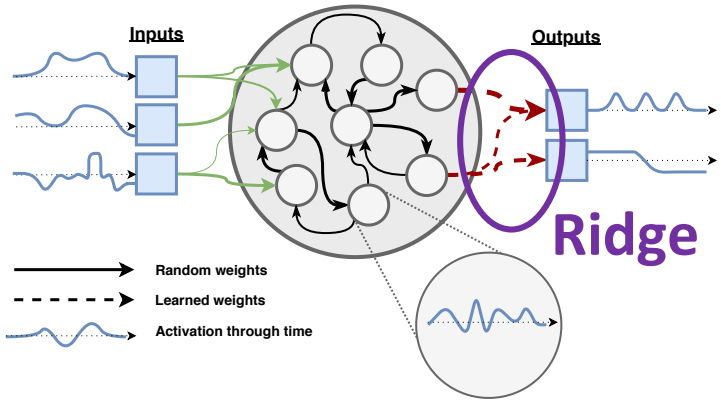
= transitory period during the « heating » of the system



Ridge

= regularisation of the read-out linear regression

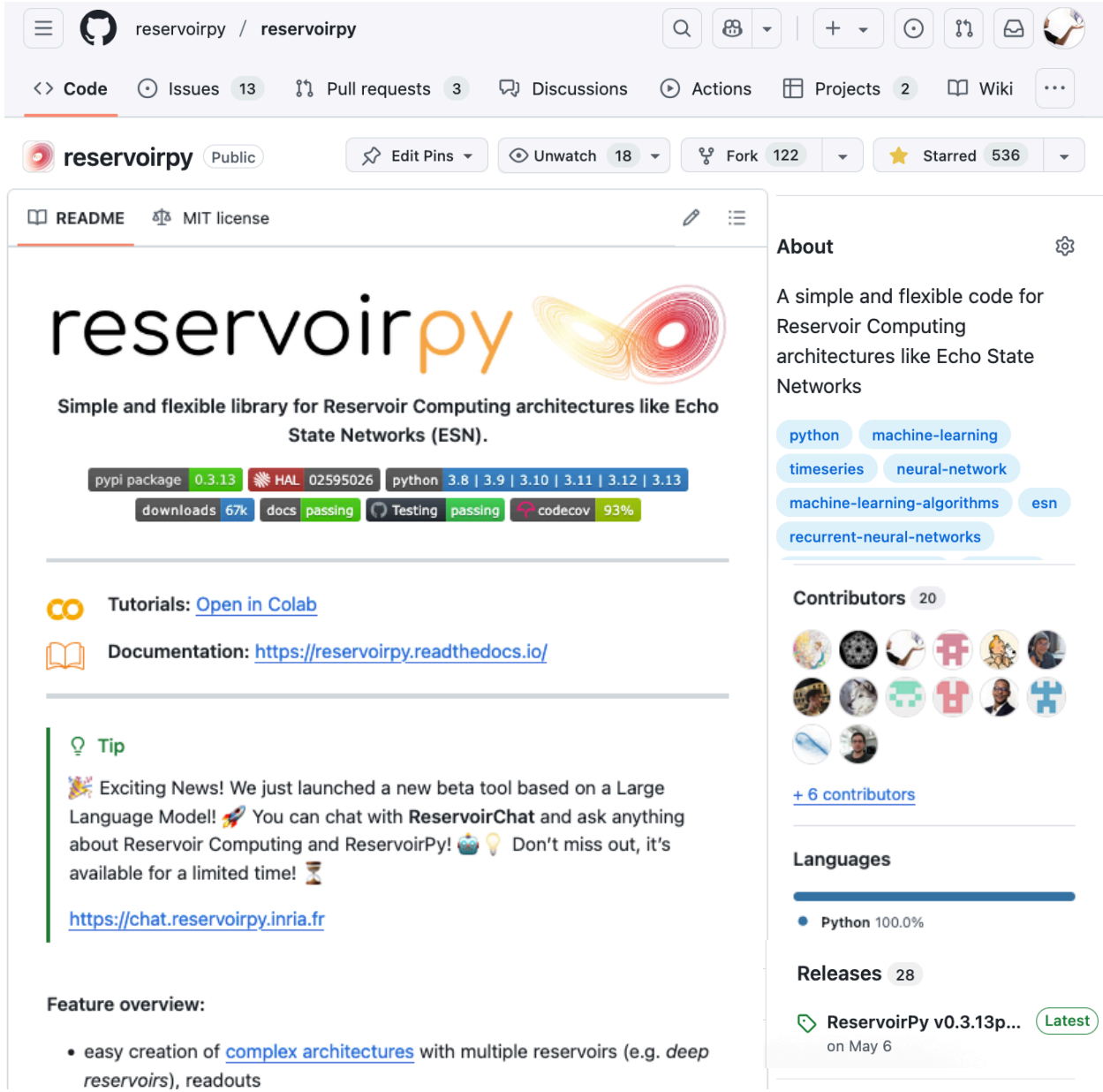




ReservoirPy

ReservoirPy
Pure Python scientific libraries

Few dependencies: mainly **Numpy** (+ optional **JAX backend** very soon!)



github.com/reservoirpy/reservoirpy

Library	Main dependency	Last release	Doc.	Tests	On.	Fb.	Model type
★ **reservoirpy (this package)**	`numpy, jax` (optional)	May 2025	✓	✓	✓	✓	ESN, DeepESN, NVAR Intrinsic Plasticity, LSM, Local Plasticity
EchoTorch	`PyTorch`	Jan. 2021	✓	✓	✓	✓	ESN, DeepESN Conceptor
★ reservoir-computing [1]	`numpy`	Jan. 2025	✓	×	×	×	ESN, state reduction
★ ReservoirComputing.jl	`Julia`	Mar. 2025	✓	✓	×	×	ESN, DeepESN, HybridESN
easyesn	`numpy / CuPy / PyTorch`	Jan. 2021	×	×	×	✓	ESN, Spatio-Temporal ESN
DeepESN	`numpy`	Apr. 2025	×	×	×	×	DeepESN
PyRCN	`scikit-learn`	Jul. 2024	✓	✓	✓	×	ESN, EulerSN
echoes	`scikit-learn`	Jun. 2025	✓	✓	×	×	ESN
RcTorch	`PyTorch`	Jul. 2022	✓	×	✓	×	ESN

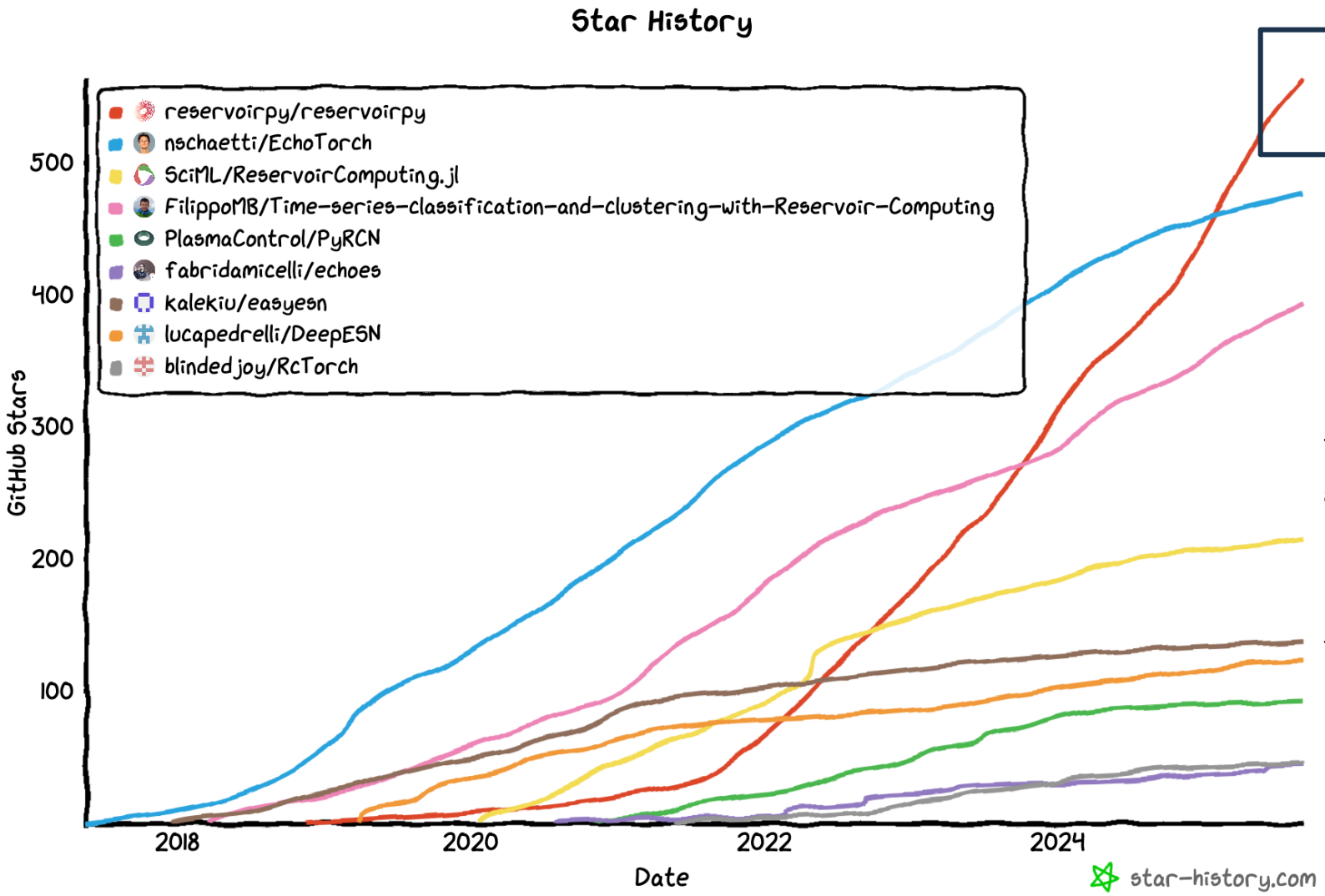
Table 1: Comparative table of some open source software for Reservoir Computing. This table might not be exhaustive. **Doc.** Complete documentation. **On.** Online learning strategies included. **Fb.** Feedback and delayed connections. **Deep** The software allows the design of complex models where basic RC elements such as reservoirs and readouts can be stacked to form so-called "deep" networks.

★ Projects with recent major updates

1: https://github.com/FilippoMB/Time-series-classification-and-clustering-with-Reservoir-Computing

Trouvain, Bernard* & Hinaut* (preprint)

78



→ ReservoirPy offers a solution that supports steady and lasting growth.

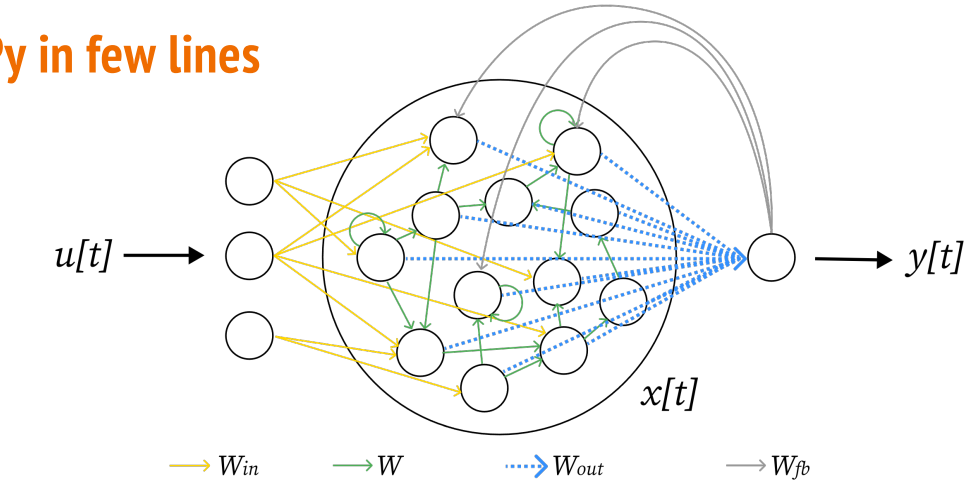
→ 2 times for citations in 2024 than in 2023 (G. Scholar)

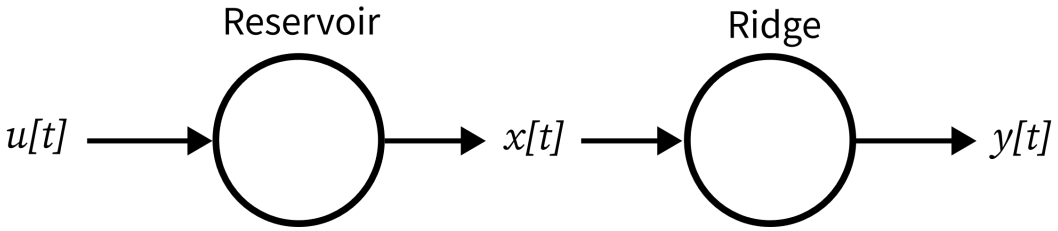
[Image source "star history"]

ReservoirPy in few lines

Node creation

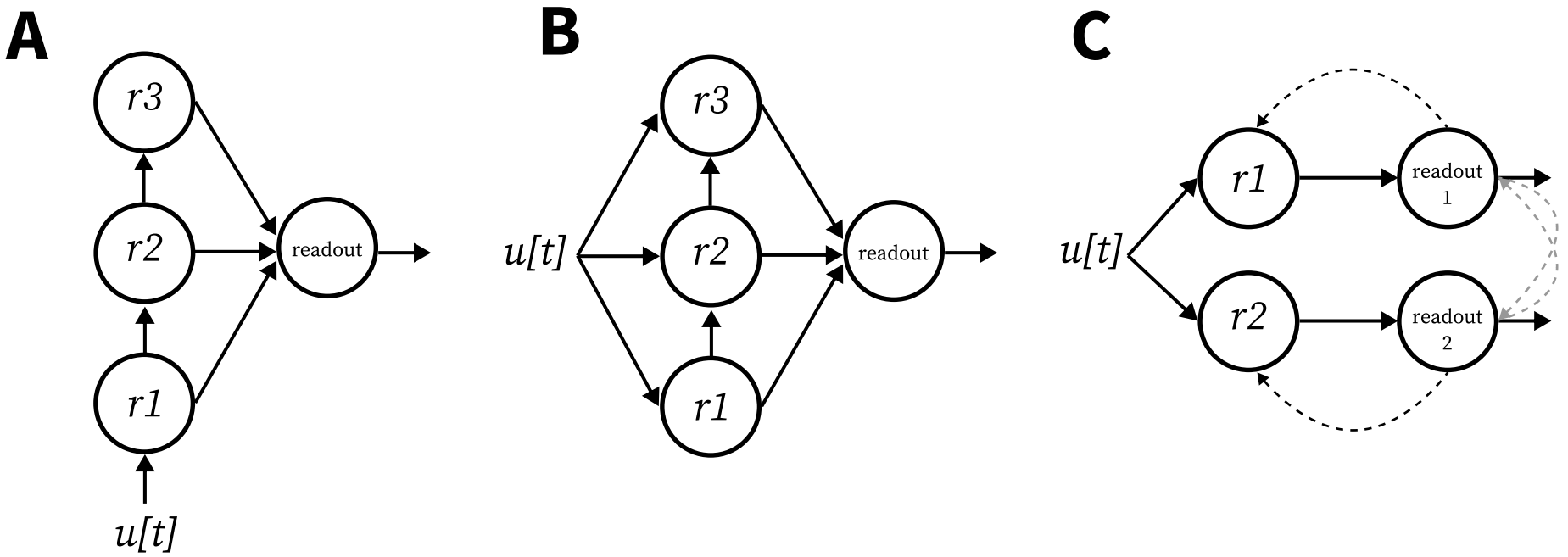
A *node* is an independent operator, that applies a function on some data, potentially in a recurrent way.

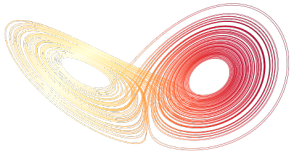





```
from reservoirpy.nodes import Reservoir, Ridge

reservoir = Reservoir(100, lr=0.3, sr=1.25, input_scaling=0.1)
readout = Ridge(ridge=1e-5)
```

Enable complex architectures



reservoirpy  v0.4.0

Reservoir dedicated framework

- **Online/Offline** learning
- **Feedback loops**
- **Hierarchical** models
  - *e.g. Deep Reservoir Computing*
- **Parallel** computing
- R language, (interface)
- Datasets, metrics, …
  - e.g. Memory Capacity
- Hyperoptimization tools
- ***ReservoirR***: interface in R language
- Tutorials, examples, documentation, …

➡️ Share your models in ReservoirPy

Reservoir models

- Echo State Networks (ESN)
- Autoregressive reservoir (NVAR)
- Ridge regression
- FORCE learning
- Intrinsic Plasticity
- Usual reservoir topologies (ring, orthogonal, …)
- Scikit-learn models, (interface)
- Simulated physical models (optical RC)

New • ES^{2}N
New • Local Plasticity (github contribution @Finebouche)
New • Arbitrary delays (between nodes)
New • LIF spiking neurons
New • Cluster & Small-world topologies
New • **v0.4: main refactoring and speed-up**
Soon! • (end Sept 2025) JAX backend

Hands-on ReservoirPy

Follow with the Introduction Notebook

github.com/reservoirpy/presentations/…

main presentations / ICANN-2025 /

PAUL-BERNARD Renamed files + improve README e226ec0 · 29 minutes ago

Name	Last commit message	Last commit date
..		
01_Introduction.ipynb	Renamed files + improve README	29 minutes ago
02_Hackathon.ipynb	Renamed files + improve README	29 minutes ago
README.md	Renamed files + improve README	29 minutes ago

README.md

ICANN 2025 - Tutorial on ReservoirPy

This is a Tutorial presented at the 3rd Reservoir Computing workshop held at ICANN 2025.

The aim is to:

- make participant familiar with the key concepts of Reservoir Computing;
- make participant familiar with ReservoirPy library the most popular reservoir library in Python;
- give you quick insights on ReservoirPy key features;
- show you how to optimize dynamically your hyperparameter search
- propose a quick hackathon and see which participant manages to reach the best performances while testing other reservoir nodes:
  - ES2N node
  - Intrinsic Plasticity
  - Next Generation Reservoir Computing

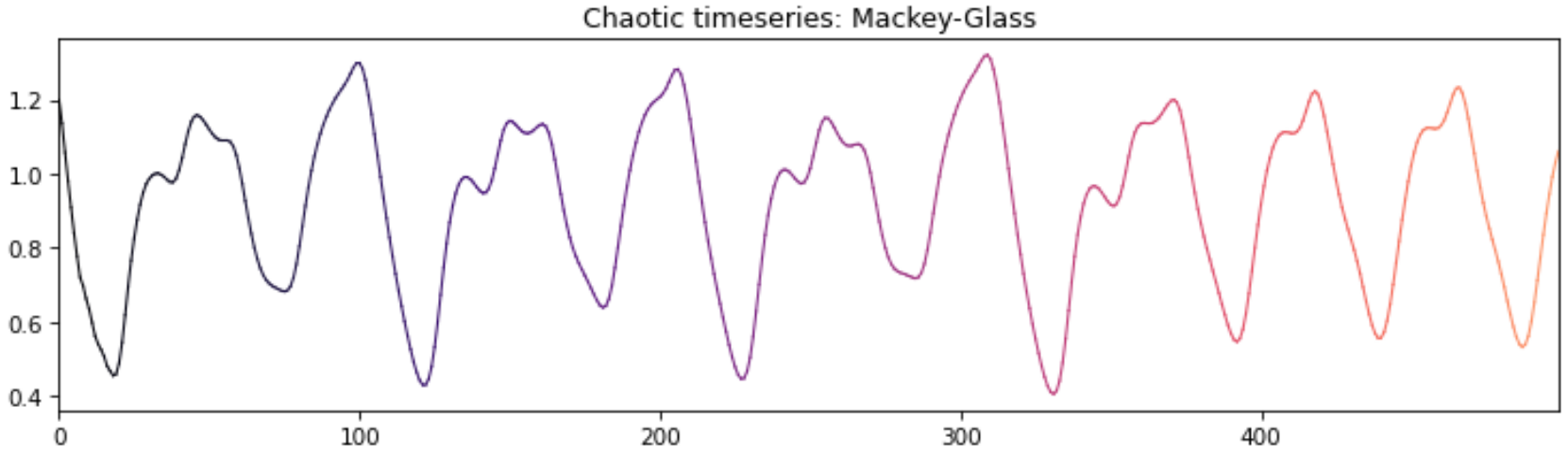
Data format

Everything is a numpy array.
shape = (timesteps, features)

```
from reservoirpy.datasets import mackey_glass

timeseries = mackey_glass(n_timesteps=2_000)
timeseries.shape
```

✓ 0.0s Python

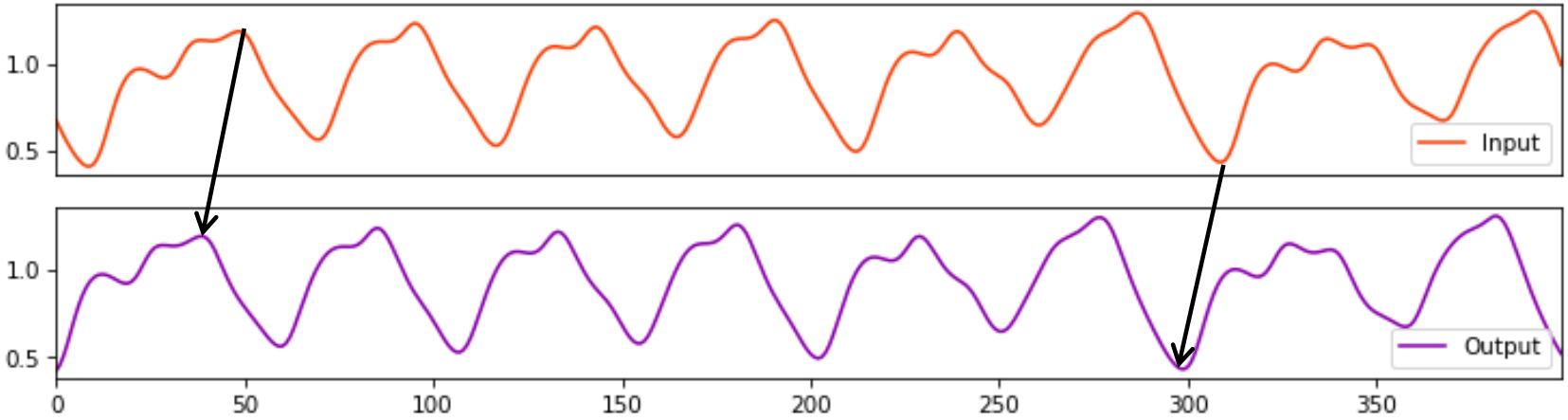
```
(2000, 1)
```



Data pre-processing

```
from reservoirpy.datasets import to_forecasting

dataset = to_forecasting(X, forecast=10, test_size=0.2)
x_train, x_test, y_train, y_test = dataset
```

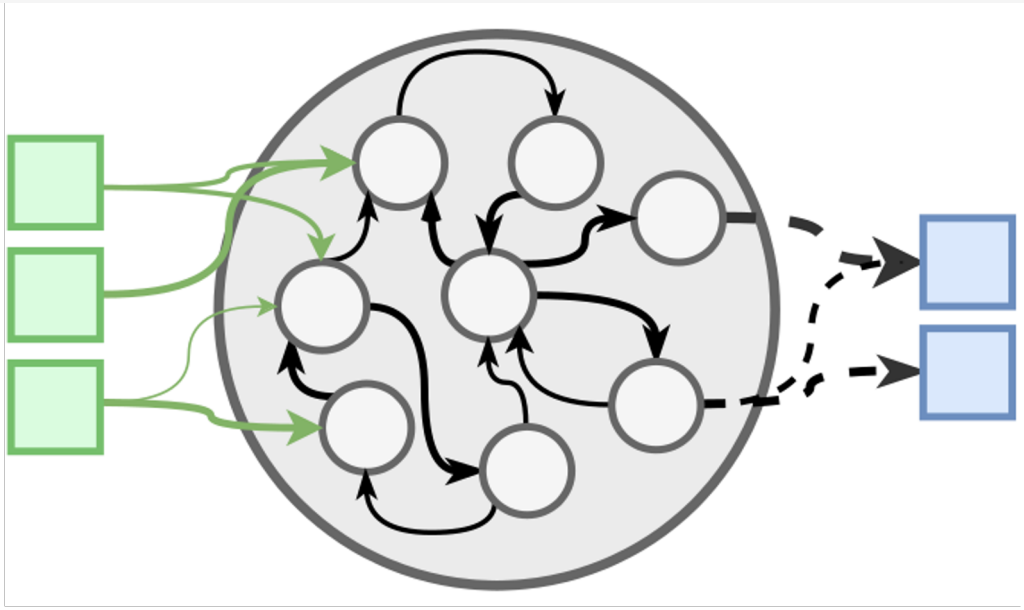
✓ 0.0s Python



Echo State Network creation

```
model = ESN(
    units=100, # neurons inside the reservoir
    lr=0.3, # leak rate
    sr=1.25, # spectral radius of the weight matrix
    ridge=1e-8) # Regularization parameter
```

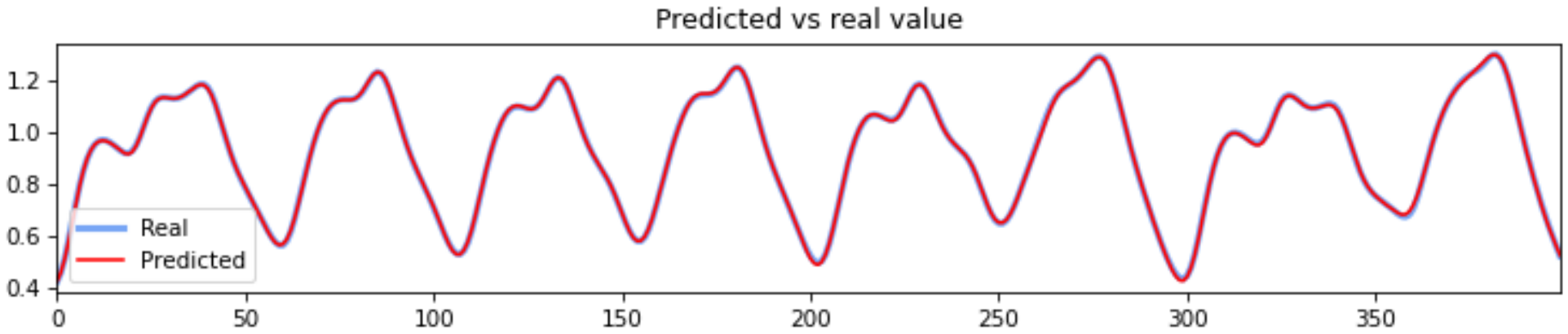
✓ 0.0s Python



Fit and run the model

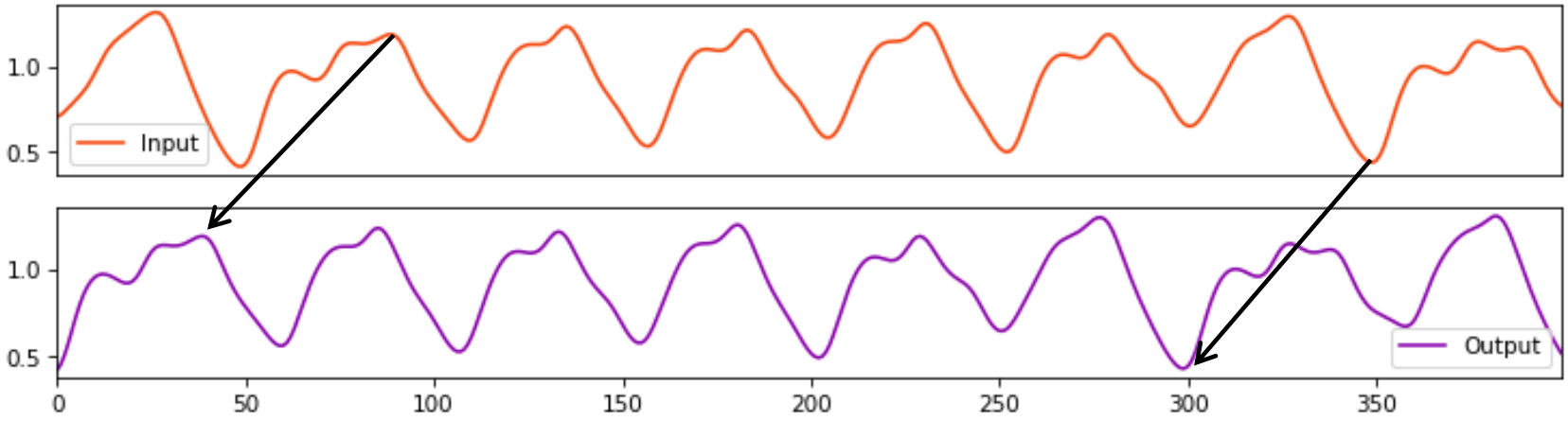
```
model.fit(x_train, y_train) # Learning phase
y_pred = model.run(x_test) # Prediction
```

✓ 0.5s Python



… what about a t+50 prediction?

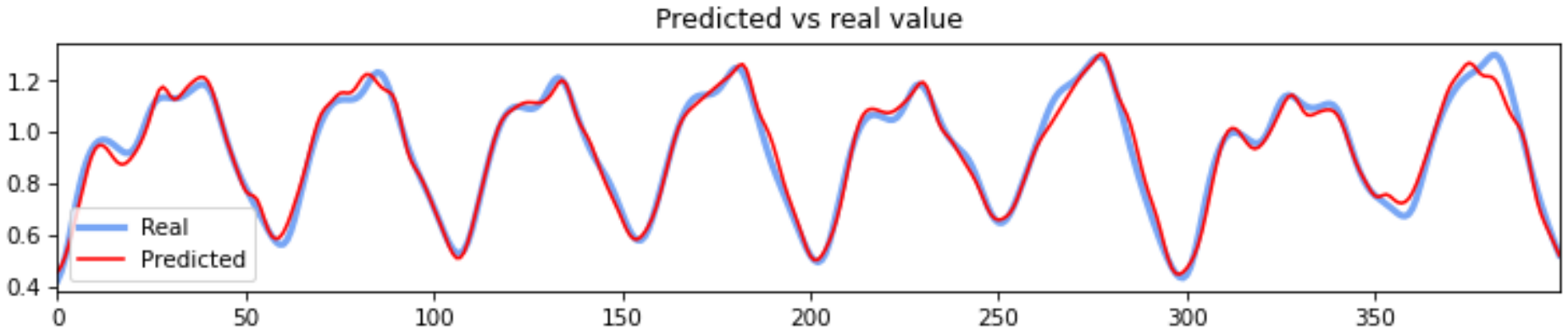
```
dataset = to_forecasting(X, forecast=50, test_size=0.2)
x_train, x_test, y_train, y_test = dataset
```



… what about a t+50 prediction?
The result

```
# Fitting the model
model.fit(x_train, y_train)
# Testing the model
y_pred = model.run(x_test)
```

✓ 0.4s Python



Summary

```python
model = ESN(
    units=100, # neurons inside the reservoir
    lr=0.3, # leak rate
    sr=1.25, # spectral radius of the weight matrix
    ridge=1e-8, # Regularization parameter
)

# Fitting the model
model.fit(x_train, y_train)
# Testing the model
y_pred = model.run(x_test)
```

✓ 0.3s Python

What is happening inside the reservoir?

```
activity = model.reservoir.run(x_test)
```

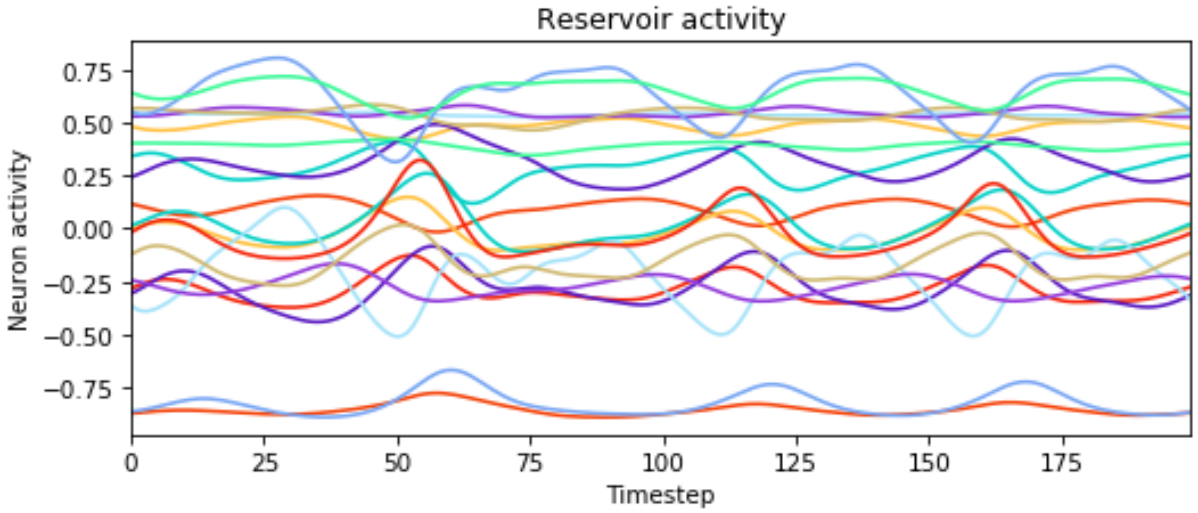
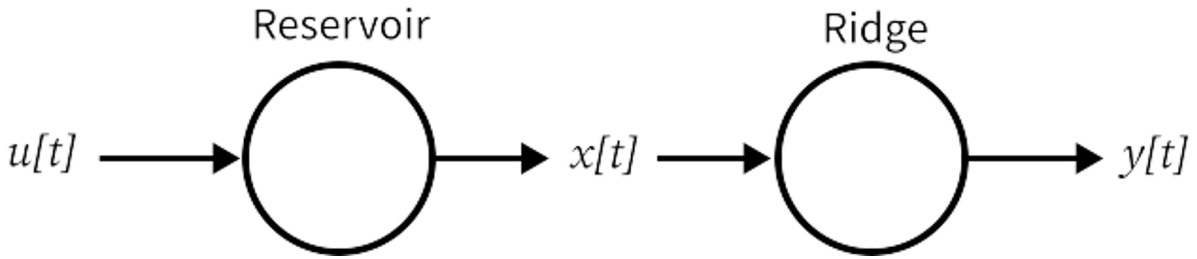


Image credits : Wikimedia

The nodes

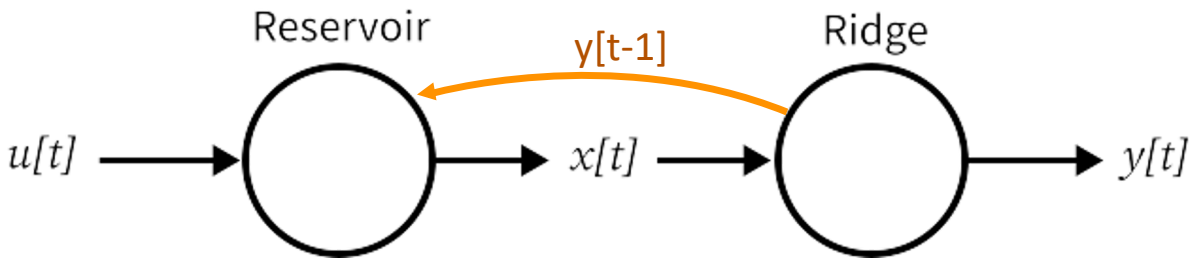
```
from reservoirpy.nodes import Reservoir, Ridge

reservoir = Reservoir(units=10)
readout = Ridge(ridge=1e-4)

model = reservoir >> readout
```

✓ 0.0s Python



Feedback connections

```
reservoir = Reservoir(units=10)
readout = Ridge(ridge=1e-4)

# create feedback
model = (reservoir >> readout) & (reservoir << readout)
```

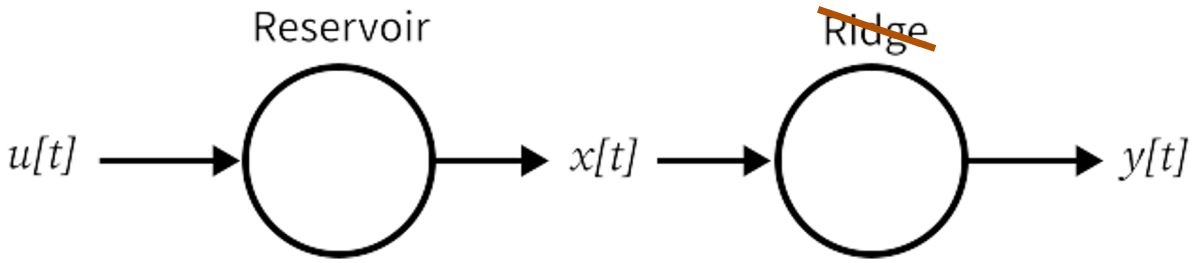


The scikit-learn node

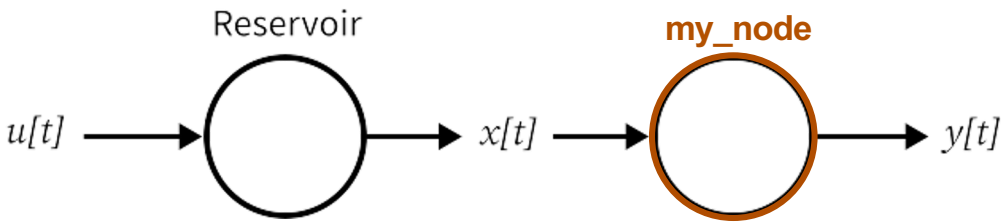
```python
from reservoirpy.nodes import ScikitLearnNode
from sklearn.linear_model import (
    RidgeClassifier,
    LogisticRegression,
)
reservoir = Reservoir(units=10)

classifier = ScikitLearnNode(RidgeClassifier)
logistic_regressor = ScikitLearnNode(LogisticRegression)

model1 = reservoir >> classifier
model2 = reservoir >> logistic_regressor
```

✓ 0.0s Python

Any scikit-learn model

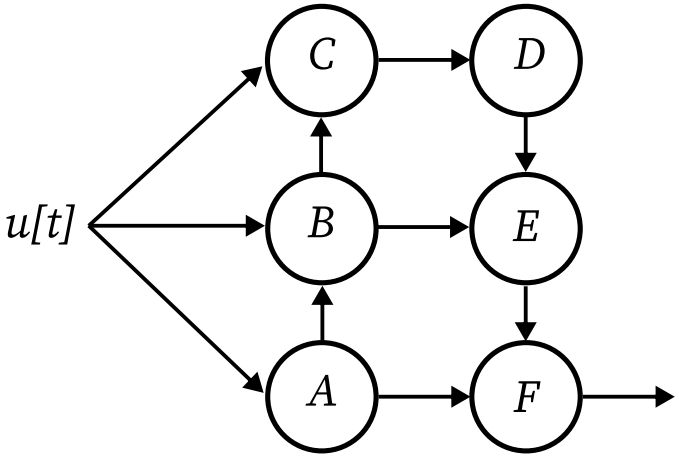


Create custom node




```python
from reservoirpy import Node


class MyNode(Node):
    def __init__(self, param1, param2):
        ...

    def initialize(self, x):
        ...

    def _step(self, state, x):
        ...
```

Enable complex architectures




```python
from reservoirpy.nodes import Input, Output

A, B, C, D, E, F = (Node() for _ in range(6))

path1, path2 = A >> F, B >> E
# One-to-many connection using a list
path3 = Input() >> [A, B, C]
# Chain of connections
path4 = A >> B >> C >> D >> E >> F >> Output()
# Merge all pathways to create the full computational graph
model = path1 & path2 & path3 & path4
```

Tools for Hyperparameter (HP) optimization

Let’s make hyperparameter search easy and reproducible!

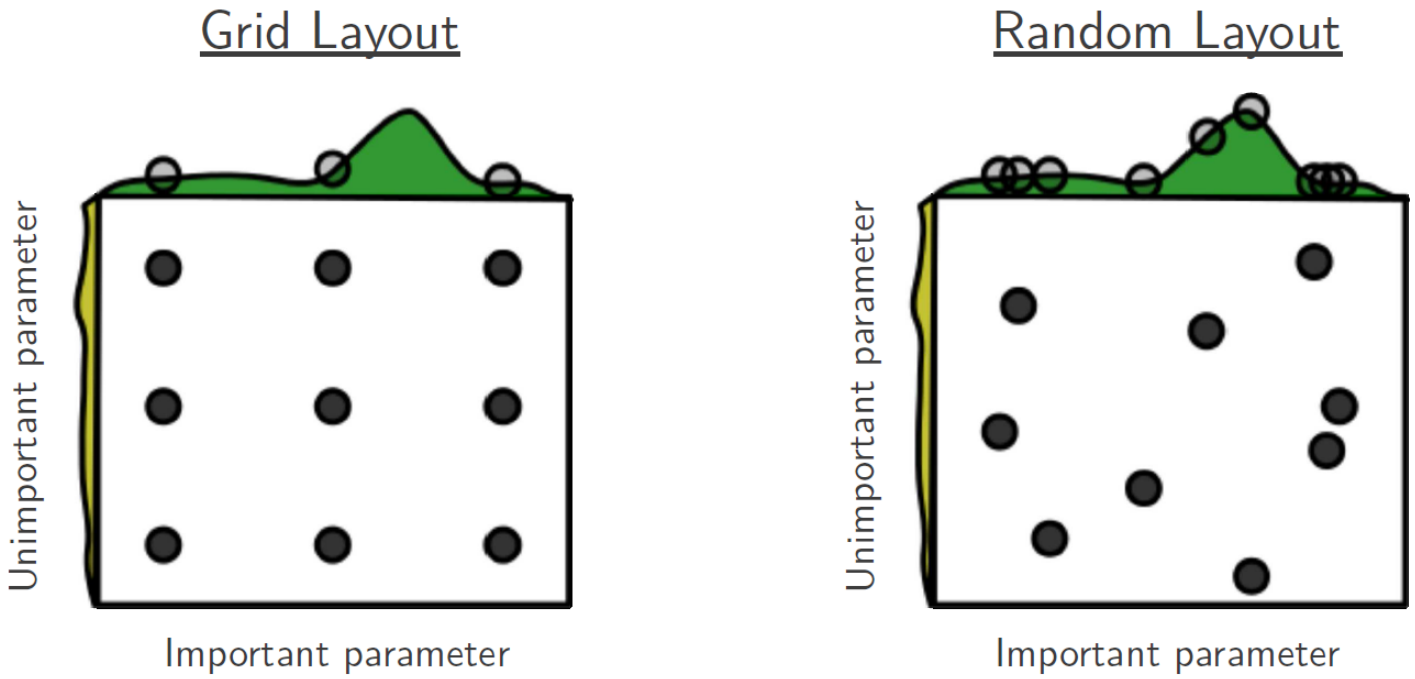
Don’t try at home without proper tools!

(🙏 don’t use grid search!)

Practical advices for parameter optimization

➡️ Grid search often fails in high dimension spaces

- « Grid search allocate **too many trials** to the exploration of **dimensions that do not matter …**
- … and suffer from **poor coverage in dimensions that are important**. » (Bergstra, 2012)



Which Hyperparameters (HP) should you tune?

- Important hyperparameters
  - Number of neurons (N)
  - **Spectral Radius (sr)**
  - **Leak-Rate (lr**)
  - **Input Scaling (input_scaling)**
  - **Ridge** (ridge)
  - Warm-up (= « transitory phase »)
  - (Feedback scaling)

Most important ones

Which Hyperparameters (HP) should you tune?

- Important hyperparameters
  - Number of neurons (N)
  - **Spectral Radius (sr)**
  - **Leak-Rate (lr)**
  - **Input Scaling (input_scaling)**
  - **Ridge** (ridge)
  - Warm-up (= « transitory phase »)
  - (Feedback scaling)

Most important ones

```
# Build your model given the input parameters
reservoir = Reservoir(
    units=N,
    sr=sr,
    lr=lr,
    input_scaling=input_scaling,
    seed=variable_seed
)

readout = Ridge(ridge=ridge)

model = reservoir >> readout
```

Which Hyperparameters (HP) should you tune?

- Important hyperparameters
  - Number of neurons (N)
  - **Spectral Radius (sr)**
  - **Leak-Rate (lr)**
  - **Input Scaling (input_scaling)**
  - **Ridge** (ridge)
  - Warm-up (= « transitory phase »)
  - (Feedback scaling)

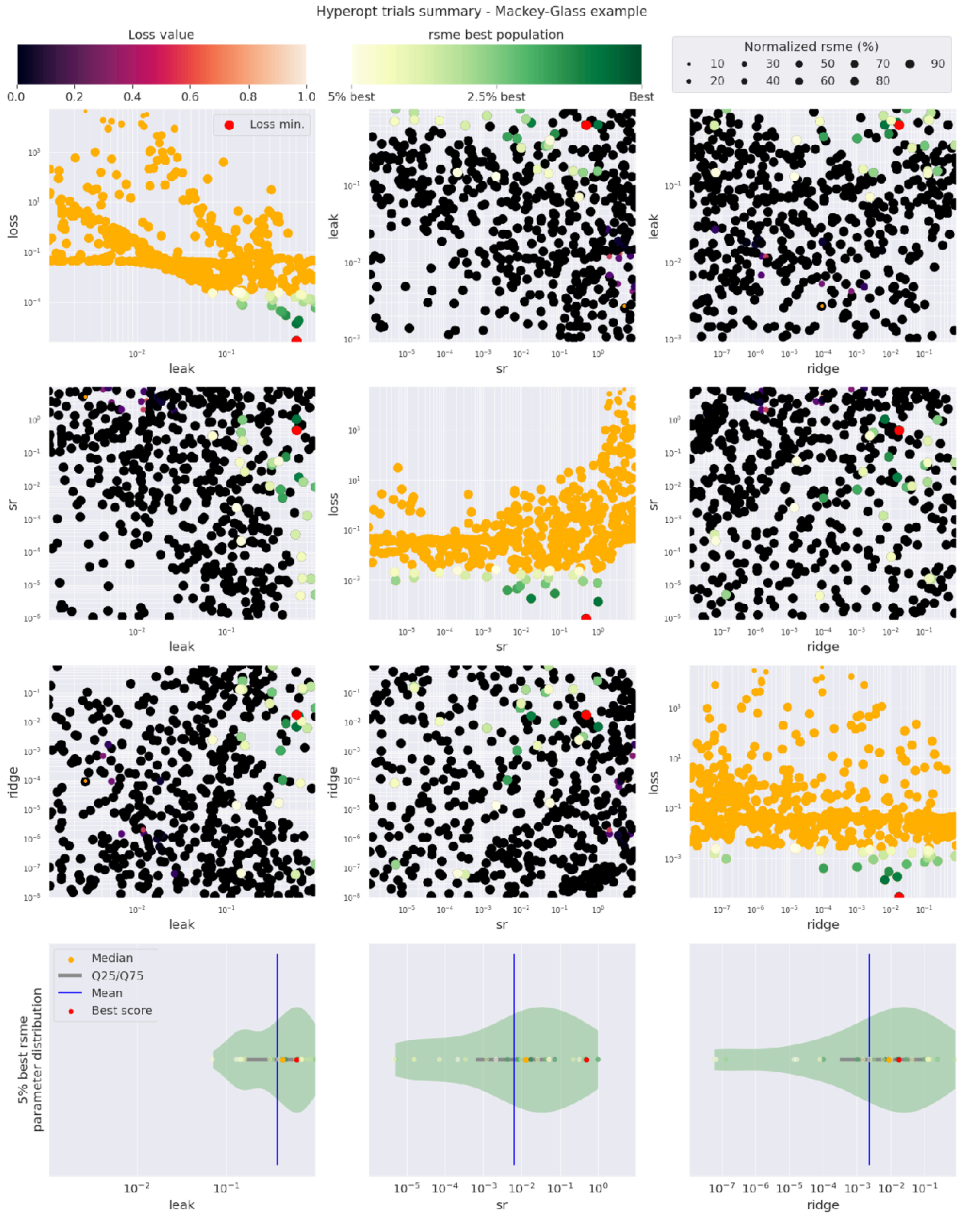
Most important ones

```python
# Build your model given the input parameters
reservoir = Reservoir(
    units=N,
    sr=sr,
    lr=lr,
    input_scaling=input_scaling,
    seed=variable_seed
)

readout = Ridge(ridge=ridge)

model = reservoir >> readout
```

Define logarithmic ranges & fix some HP
(e.g. N and input_scaling)

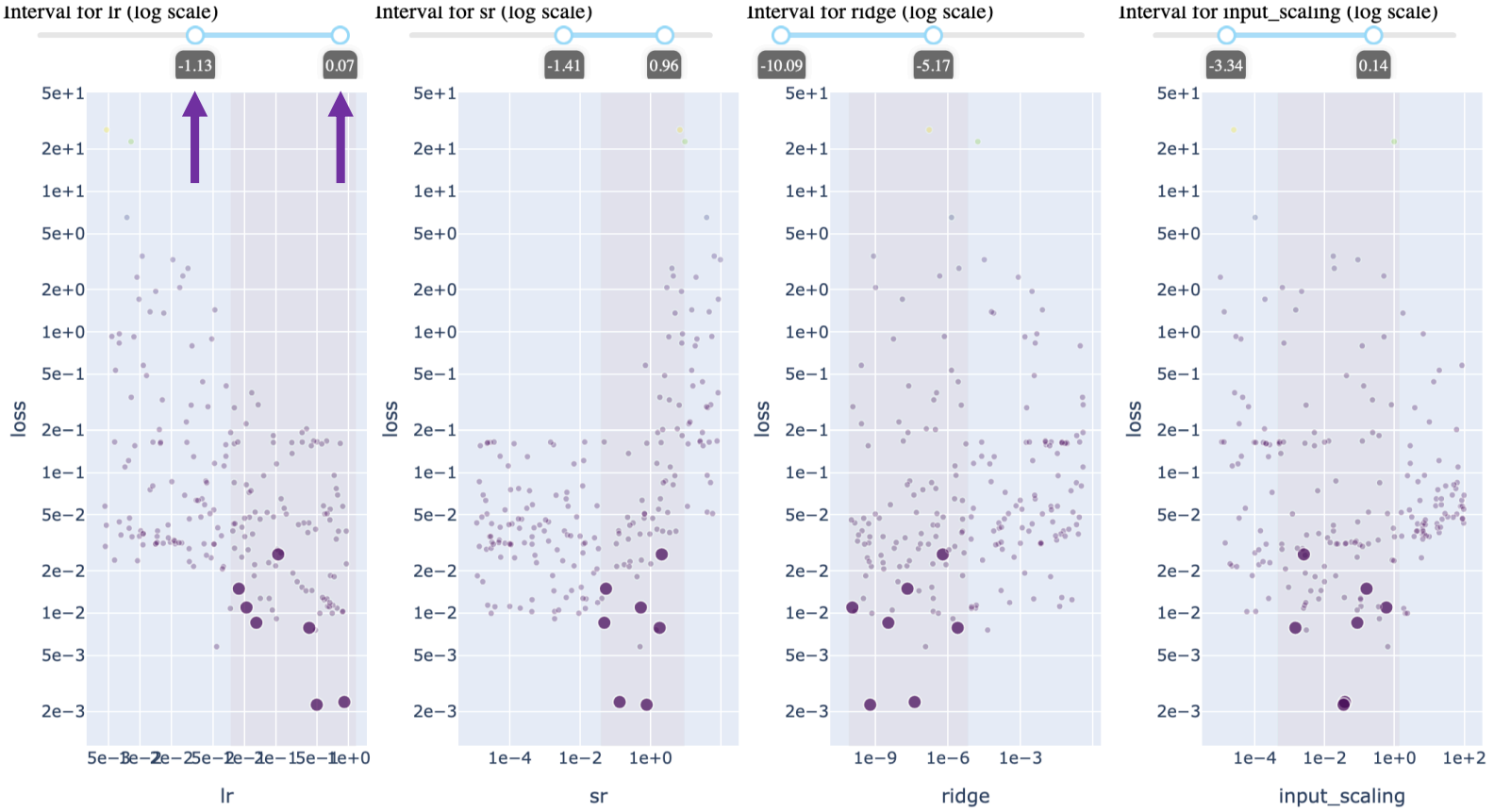
```python
hyperopt_config = {
    "exp": "hyperopt-doublescroll",    # the experimentation name
    "hp_max_evals": 200,               # the number of differents sets of parameters hyperopt has to try
    "hp_method": "random",             # the method used by hyperopt to chose those sets (see below)
    "seed": 42,                        # the random state seed, to ensure reproducibility
    "instances_per_trial": 1,          # how many random ESN will be tried with each sets of parameters
    "hp_space": {                      # what are the ranges of parameters explored
        "N": ["choice", 100],              # the number of neurons is fixed to 100
        "sr": ["loguniform", 1e-5, 1e2],   # the spectral radius is log-uniformly distributed from 1e-5 and 100
        "lr": ["loguniform", 1e-5, 1.0],    # idem with the leaking rate, from 1e-5 to 1
        "input_scaling": ["choice", 1.0],   # the input scaling is fixed
        "ridge": ["loguniform", 1e-6, 1e6],        # regularization parameter is explored widely
        "seed": ["randint", 0, 2**32]          # seeds used for random initialisation of weights matrices
    }
}
```

Graphical view of hyperparameter optimization



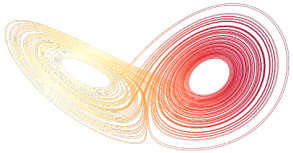
Hyperparameter Dynamic Exploration

```
interactive_report(hyperopt_config['exp'], ['lr', 'sr', 'ridge', 'input_scaling'], log_loss=True, out_path='limit.json')
```



→ quick & robust selection of hyperparameters ranges

Discussion

reservoirpy  v0.4.0

Reservoir dedicated framework

- **Online/Offline** learning
- **Feedback loops**
- **Hierarchical** models
  - *e.g. Deep Reservoir Computing*
- **Parallel** computing
- R language, (interface)
- Datasets, metrics, ...
  - e.g. Memory Capacity
- Hyperoptimization tools
- ***ReservoirR***: interface in R language
- Tutorials, examples, documentation, ...

➡️ Share your models in ReservoirPy

Reservoir models

- Echo State Networks (ESN)
- Autoregressive reservoir (NVAR)
- Ridge regression
- FORCE learning
- Intrinsic Plasticity
- Usual reservoir topologies (ring, orthogonal, ...)
- Scikit-learn models, (interface)
- Simulated physical models (optical RC)

New • ES^{2}N
New • Local Plasticity (github contribution @Finebouche)
New • Arbitrary delays (between nodes)
New • LIF spiking neurons
New • Cluster & Small-world topologies
New • **v0.4: main refactoring and speed-up**
Soon! • (end Sept 2025) JAX backend

Perspectives

Upgrade framework capabilities

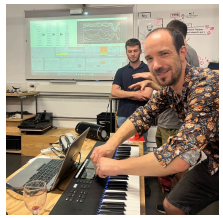
- **JAX back-end** (Sept 2025)
  - Extended distributed computation
  - GPU acceleration

- **Simulated physical Reservoir models**
  - More optical reservoir models
  - Bacterial reservoir (collab with MICALIS, INRAE, Paris-South)

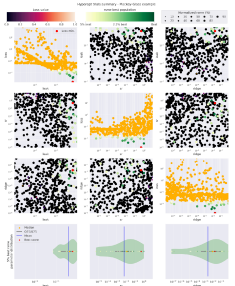
Sharing wihin RC community

- Replicate scientific results
- Implement new tools
- Increase user and expert community
- **Find new use cases**

Summary on reservoirpy



- Easy and quick to use
- Rich documentation and many tutorials (e.g. HPs optim.)



- Rich set of features + Upcoming new features
- Possible to interface with other languages
- We can build advanced and robust architectures by sharing models

https://github.com/reservoirpy/reservoirpy


Many thanks to
Nathan Trouvain
& Paul Bernard
+ all developers

⚠ Looking for a post-doc!

Thank you!

➡ Share your models in ReservoirPy
➡ Ideas of new features to implement?

xavier.hinaut@inria.fr & paul.bernard@inria.fr
@neuronalX & @PAUL-BERNARD
@hinaut @reservoirpy

➡ Add papers on the
Reservoir "Awesome list" ➡

github.com/reservoirpy/awesome-reservoir-computing

Play with the Hackathon Notebook

github.com/reservoirpy/presentations/…

main presentations / ICANN-2025 /

PAUL-BERNARD Renamed files + improve README e226ec0 · 29 minutes ago

Name	Last commit message	Last commit date
..		
01_Introduction.ipynb	Renamed files + improve README	29 minutes ago
02_Hackathon.ipynb	Renamed files + improve README	29 minutes ago
README.md	Renamed files + improve README	29 minutes ago

README.md

ICANN 2025 - Tutorial on ReservoirPy

This is a Tutorial presented at the 3rd Reservoir Computing workshop held at ICANN 2025.

The aim is to:

- make participant familiar with the key concepts of Reservoir Computing;
- make participant familiar with ReservoirPy library the most popular reservoir library in Python;
- give you quick insights on ReservoirPy key features;
- show you how to optimize dynamically your hyperparameter search
- propose a quick hackathon and see which participant manages to reach the best performances while testing other reservoir nodes:
  - ES2N node
  - Intrinsic Plasticity
  - Next Generation Reservoir Computing

Some references using ReservoirPy

HAL id	HAL citation
hal-03699931	Nathan Trouvain, Xavier Hinaut. reservoirpy: A Simple and Flexible Reservoir Computing Tool in Python. 2022. ⟨hal-03699931⟩
hal-02595026	Nathan Trouvain, Luca Pedrelli, Thanh Trung Dinh, Xavier Hinaut. ReservoirPy: an Efficient and User-Friendly Library to Design Echo State Networks. *ICANN 2020 - 29th International Conference on Artificial Neural Networks*, Sep 2020, Bratislava, Slovakia. ⟨hal-02595026v2⟩
hal-03533731	Nathan Trouvain, Xavier Hinaut. Reservoir Computing : théorie, intuitions et applications avec ReservoirPy. *Plate-Forme Intelligence Artificielle (PFIA)*, Jun 2021, Bordeaux, France. ⟨hal-03533731⟩
hal-03203318	Xavier Hinaut, Nathan Trouvain. Which Hype for my New Task? Hints and Random Search for Reservoir Computing Hyperparameters. *ICANN 2021 - 30th International Conference on Artificial Neural Networks*, Sep 2021, Bratislava, Slovakia. ⟨hal-03203318v2⟩
hal-03482372	Silvia Pagliarini, Arthur Leblois, Xavier Hinaut. Canary Vocal Sensorimotor Model with RNN Decoder and Low-dimensional GAN Generator. *ICDL 2021- IEEE International Conference on Development and Learning*, Aug 2021, Beijing, China. ⟨hal-03482372⟩
hal-03203374	Nathan Trouvain, Xavier Hinaut. Canary Song Decoder: Transduction and Implicit Segmentation with ESNs and LTSMs. *ICANN 2021 - 30th International Conference on Artificial Neural Networks*, Sep 2021, Bratislava, Slovakia. pp.71--82, ⟨10.1007/978-3-030-86383-8_6⟩. ⟨hal-03203374v2⟩
hal-03761440	Nathan Trouvain, Nicolas P. Rougier, Xavier Hinaut. Create Efficient and Complex Reservoir Computing Architectures with ReservoirPy. *SAB 2022 - FROM ANIMALS TO ANIMATS 16: The 16th International Conference on the Simulation of Adaptive Behavior*, Sep 2022, Cergy-Pontoise / Hybrid, France. ⟨hal-03761440⟩
tel-03946773	Xavier Hinaut. Reservoir SMILES: Towards SensoriMotor Interaction of Language and Embodiment of Symbols with Reservoir Architectures. Artificial Intelligence [cs.AI]. Université de Bordeaux (UB), France, 2022. ⟨tel-03946773⟩
hal-03628290	Subba Reddy Oota, Frédéric Alexandre, Xavier Hinaut. Cross-Situational Learning Towards Robot Grounding. 2022. ⟨hal-03628290v2⟩
hal-03780006	Xavier Hinaut, Nathan Trouvain. ReservoirPy: Efficient Training of Recurrent Neural Networks for Timeseries Processing. *EuroSciPy 2022 - 14th European Conference on Python in Science*, Aug 2022, Basel, Switzerland. ⟨hal-03780006⟩
hal-03945994	Nathan Trouvain, Xavier Hinaut. Reservoir Computing : traitement efficace de séries temporelles avec ReservoirPy. *Dataquitaine 2022*, Feb 2022, Bordeaux, France. ⟨hal-03945994⟩