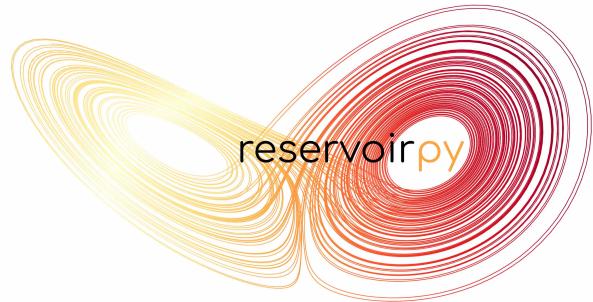


## **Chapter 3: Software libraries and applications**

Easily create, tune and share reservoir models



# Xavier Hinaut & Paul Bernard

Researcher & Engineer @Inria, Bordeaux, France

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

 @neuronalX & @PAUL-BERNARD

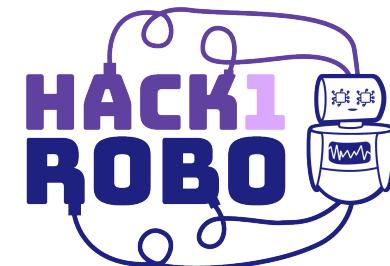
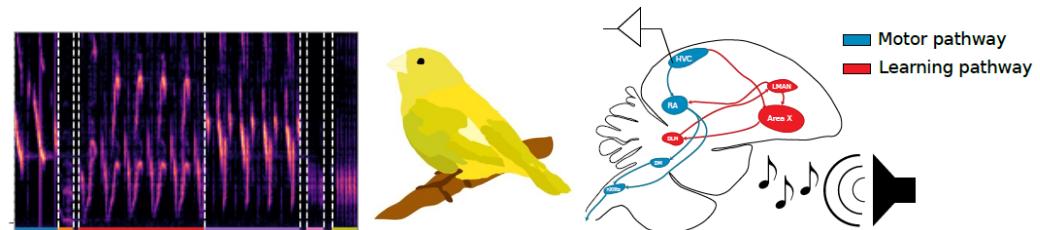
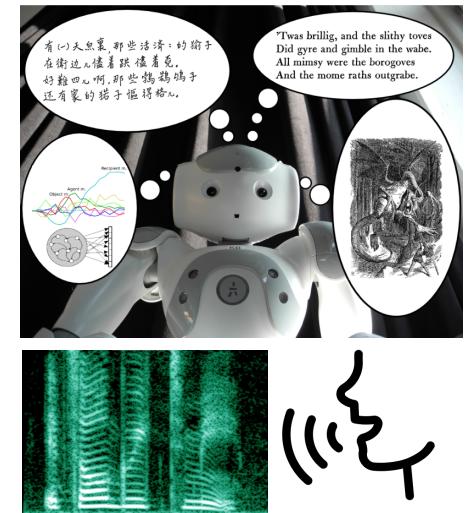
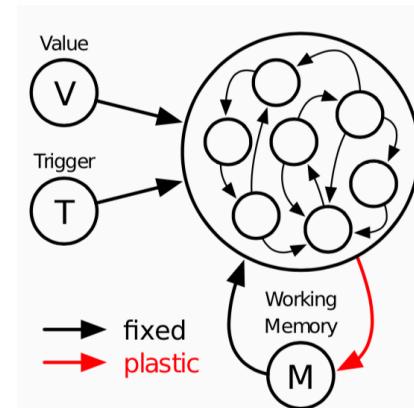
 @hinaut @reservoirpy

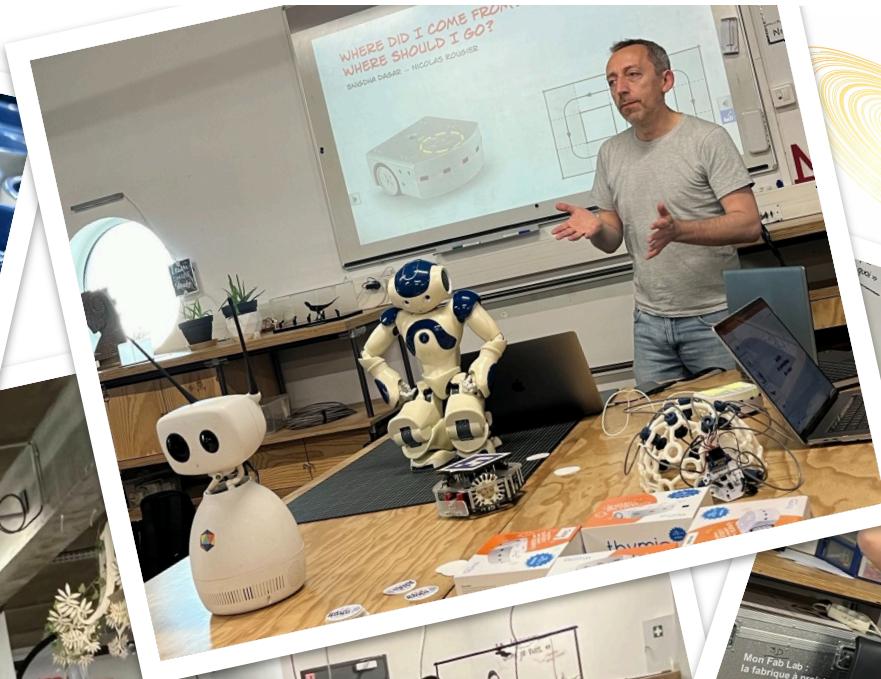


*Inria*

**LaBRI**

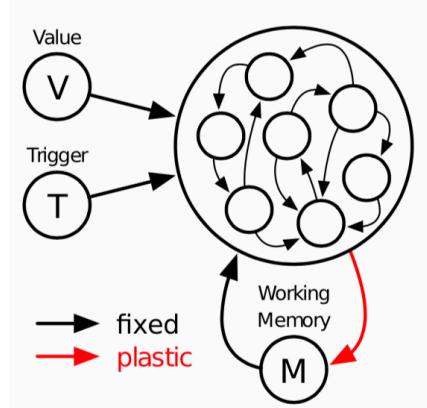
université  
de BORDEAUX



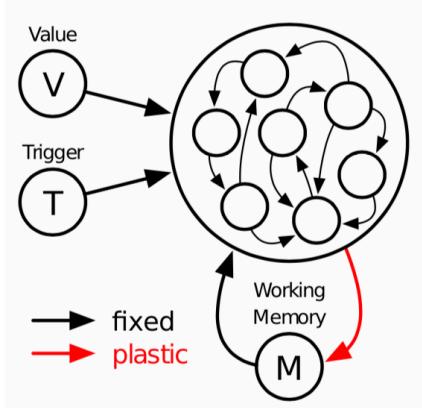


# Have you ever dreamed of being a DJ?





Alison Wonderland



```

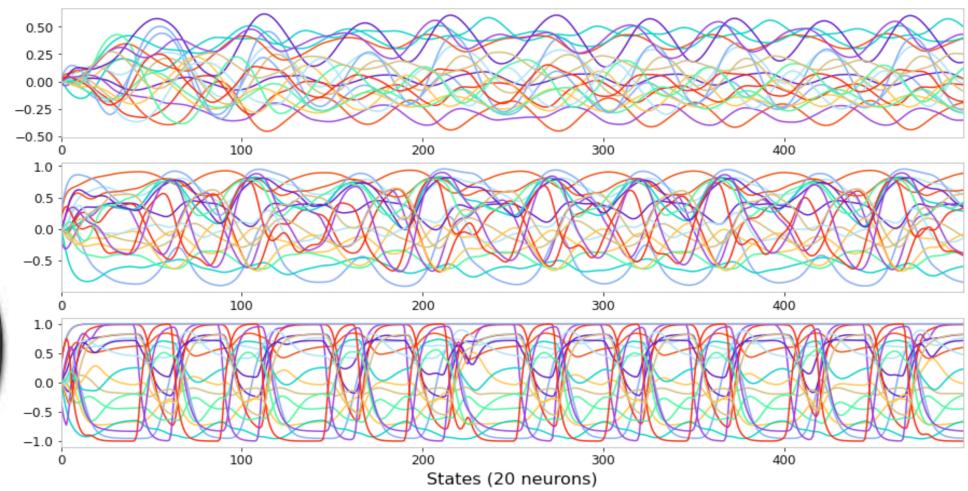
from reservoirpy.nodes import Reservoir, Ridge, Input

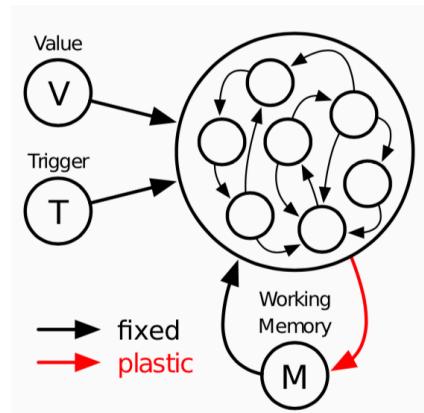
data = Input(input_dim=1)
reservoir = Reservoir(100, lr=0.3, sr=1.1)
readout = Ridge(ridge=1e-6)

esn = data >> reservoir >> readout

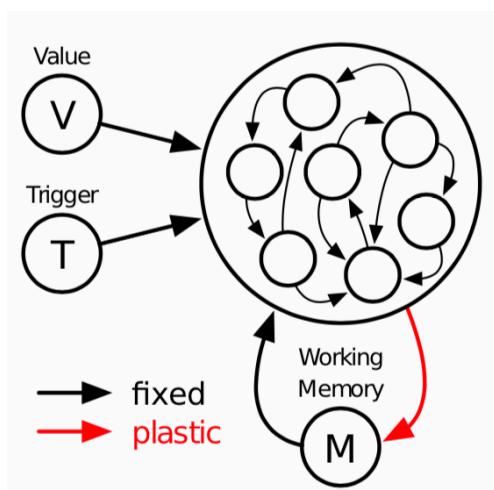
forecast = esn.fit(X, y).run(timeseries)

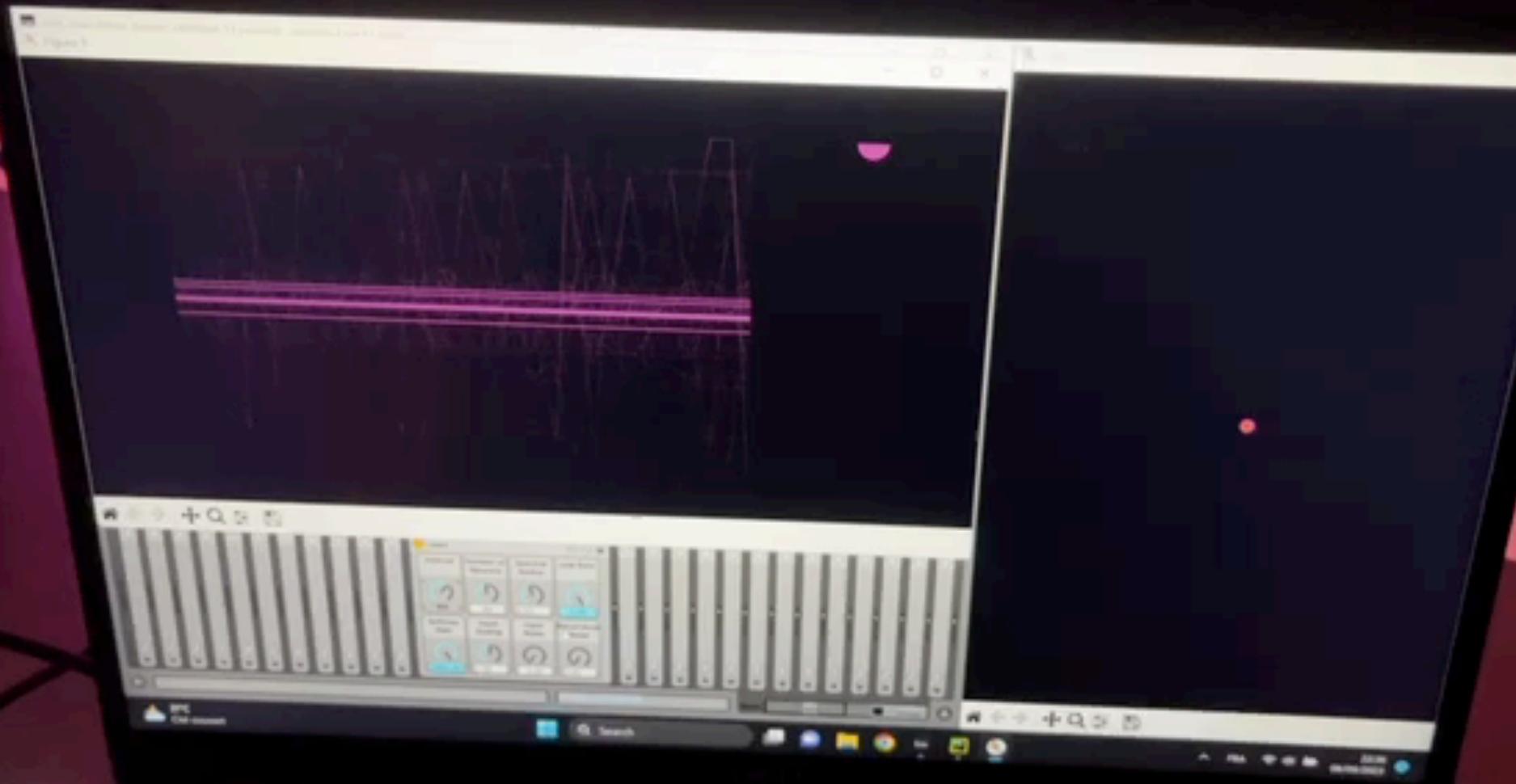
```



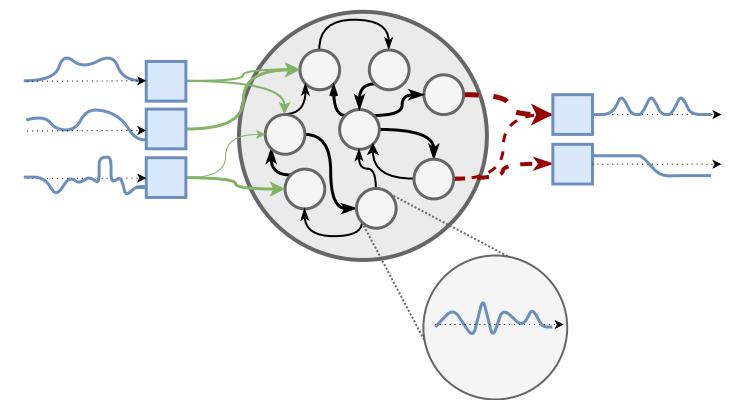


# Have you ever dreamed of playing with a reservoir?

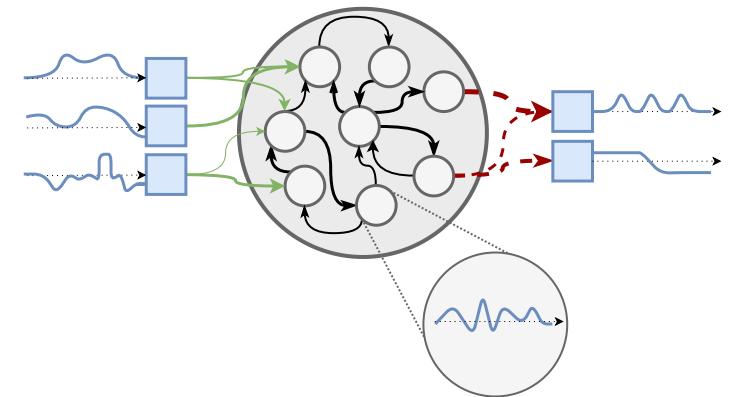




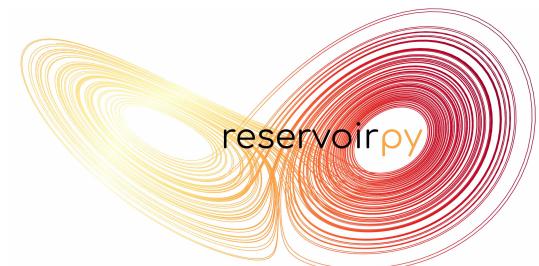
Random networks can generate fascinating dynamics



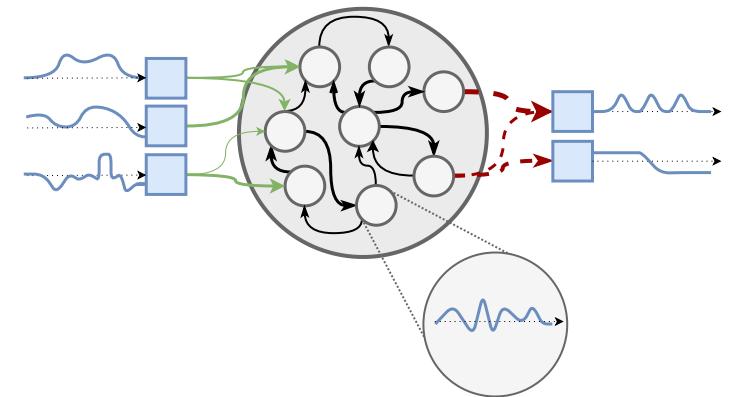
Random networks can generate fascinating dynamics



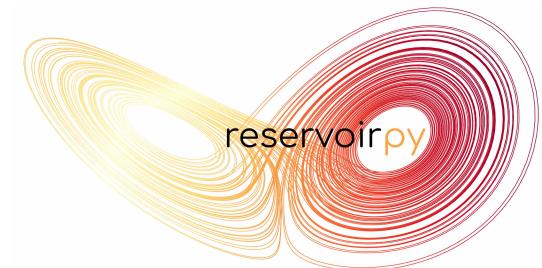
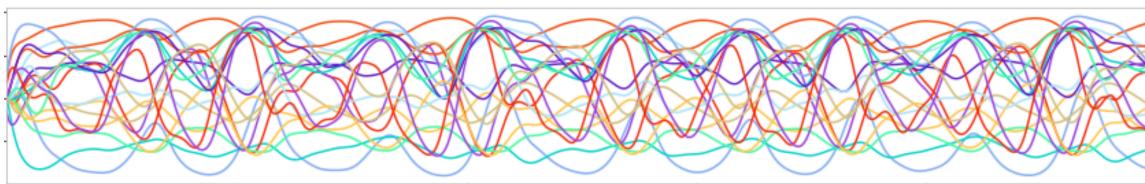
Reservoir Computing exploits  
natural dynamics of random networks



Random networks can generate fascinating dynamics



Reservoir Computing exploits  
natural dynamics of random networks



## Quick installation for ReservoirPy demo



<https://github.com/reservoirpy/reservoirpy>

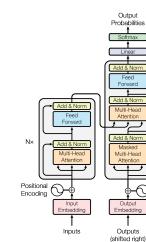
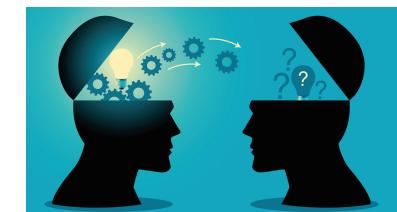
**pip install reservoirpy[hyper]**

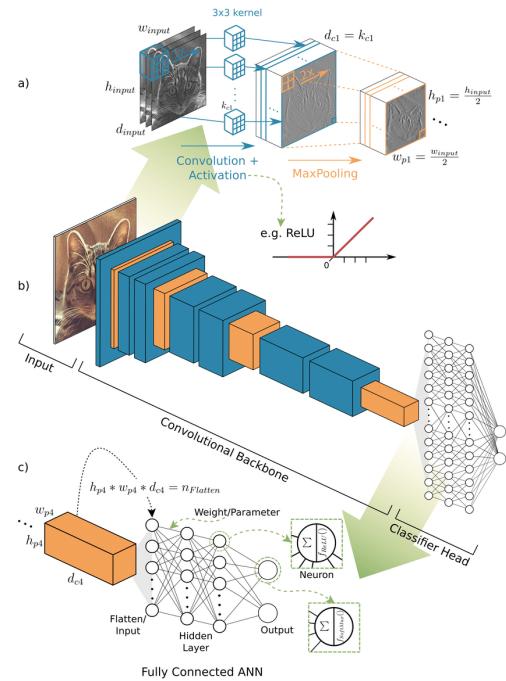
# Reservoir Computing Libraries Overview

# **Why is a library important to strengthens a community?**

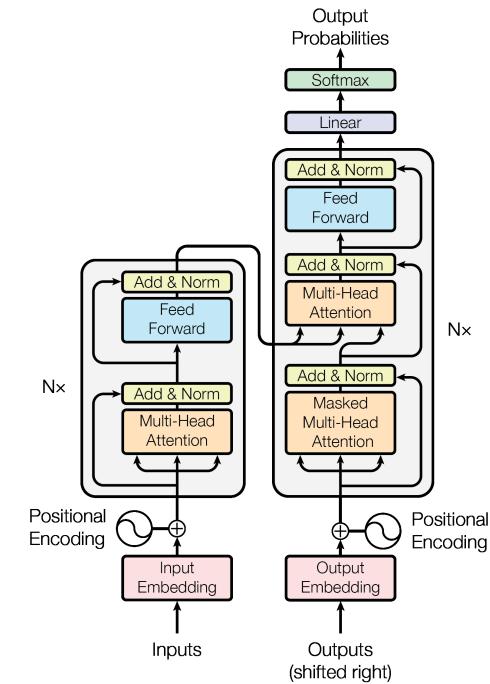
# Why a library is important to strengthens a community?

- What enabled Deep Learning community to grow so big since 2010's?
- Well known ingredients:
  - Massive amount of data
  - Massive computations available with GPUs
- Additional ingredients:
  - Several libraries backed by big institutions / companies
  - Platforms to share datasets and models 😊 **Hugging Face**
- Scikit-learn had a similar effect on the ML community
- ➔ Enabled to build complex architectures!





## Convolutional Neural Networks (CNNs)



## Transformers



### Aims of ReservoirPy

- Speed-up computations
- **Share implementation hints**
- **Share/reuse models and datasets**
- Empower to **build complex architectures** such as Transformers



# ReservoirPy

# ReservoirPy

## Pure Python scientific libraries

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



[github.com/reservoirpy/reservoirpy](https://github.com/reservoirpy/reservoirpy)

reservoirpy / reservoirpy

Code Issues Pull requests Discussions Actions Projects Wiki ...

reservoirpy Public Edit Pins Unwatch Fork Starred 567

README MIT license

# reservoirpy



Simple and flexible library for Reservoir Computing architectures like Echo State Networks (ESN).

pypi package 0.4.0 HAL 02595026 python 3.9 | 3.10 | 3.11 | 3.12 | 3.13  
downloads 85k docs passing Testing passing codecov 94%

Tutorials: [Open in Colab](#)  
Documentation: <https://reservoirpy.readthedocs.io/>

---

Feature overview:

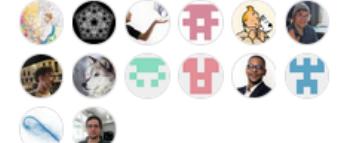
- easy creation of [complex architectures](#) with multiple reservoirs (e.g. *deep reservoirs*), readouts
- [feedback loops](#)
- [offline and online training](#)
- [parallel implementation](#)
- sparse matrix computation
- advanced learning rules (e.g. [Intrinsic Plasticity](#), [Local Plasticity](#) or [NVAR \(Next-Generation RC\)](#))
- interfacing with [scikit-learn](#) models

About

A simple and flexible code for Reservoir Computing architectures like Echo State Networks

python machine-learning timeseries neural-network machine-learning-algorithms esn recurrent-neural-networks

Contributors 20



+ 6 contributors

Languages

Python 100.0%

ReservoirPy v0.4.0 Latest 3 weeks ago

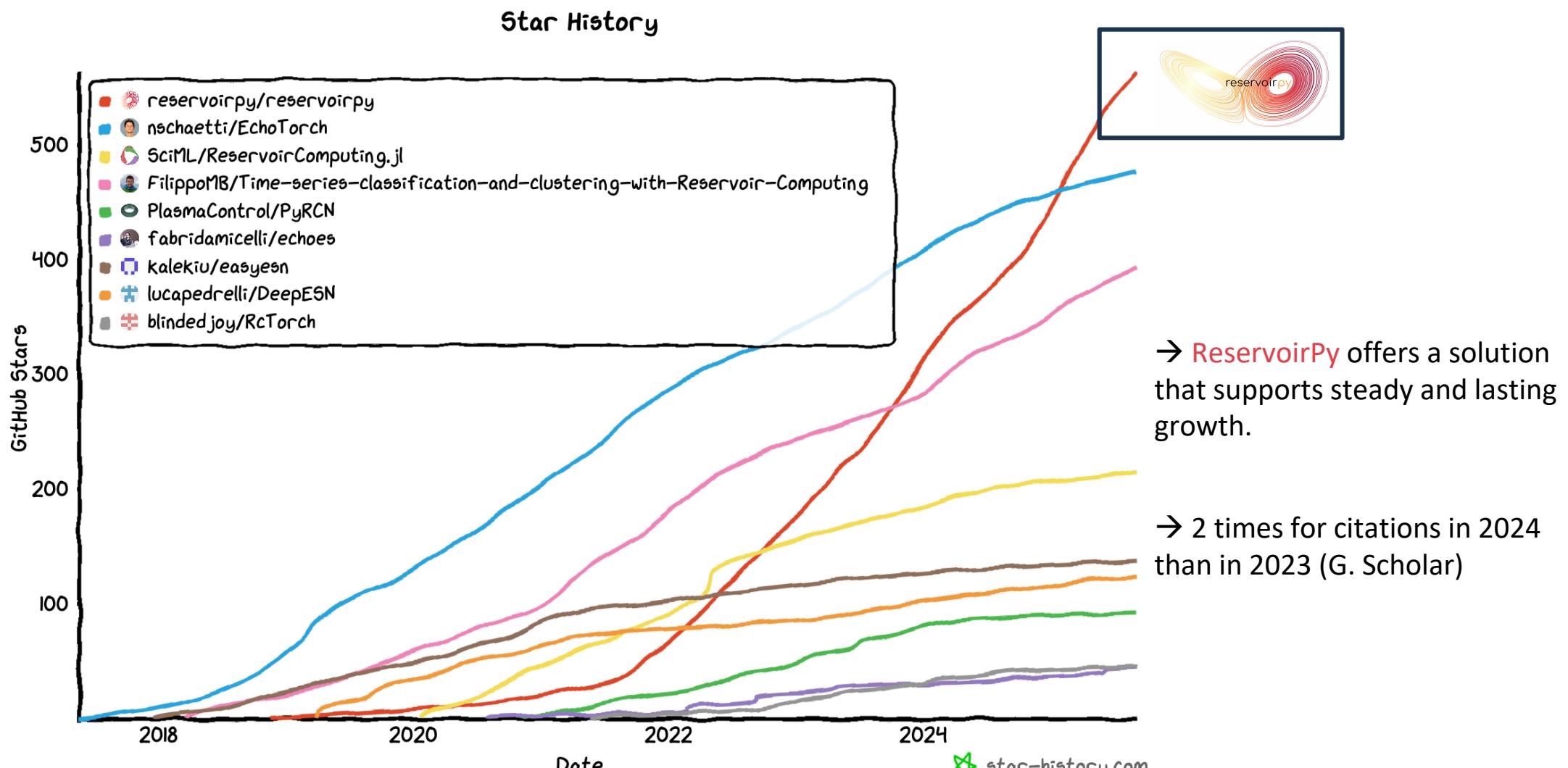
+ 29 releases

Library	Main dependency	Last release	Doc.	Tests	On.	Fb.	Model type
reservoirpy <b>(this package)</b>	numpy, jax (optional)	May 2025	✓	✓	✓	✓	ESN, DeepESN, NVAR Intrinsic Plasticity, LSM, Local Plasticity
EchoTorch	PyTorch	Jan. 2021	✓	✓	✓	✓	ESN, DeepESN Conceptor
reservoir-computing <sup>1</sup>	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>

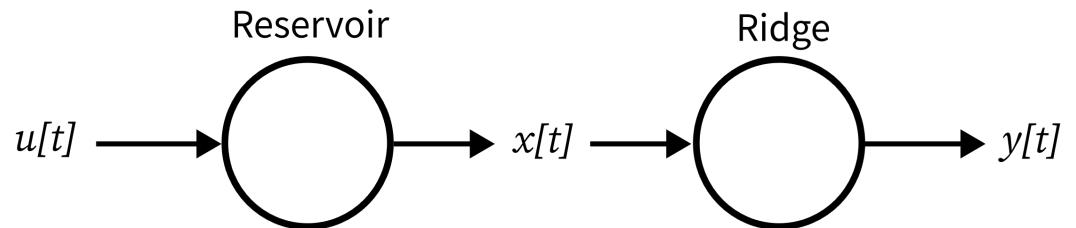
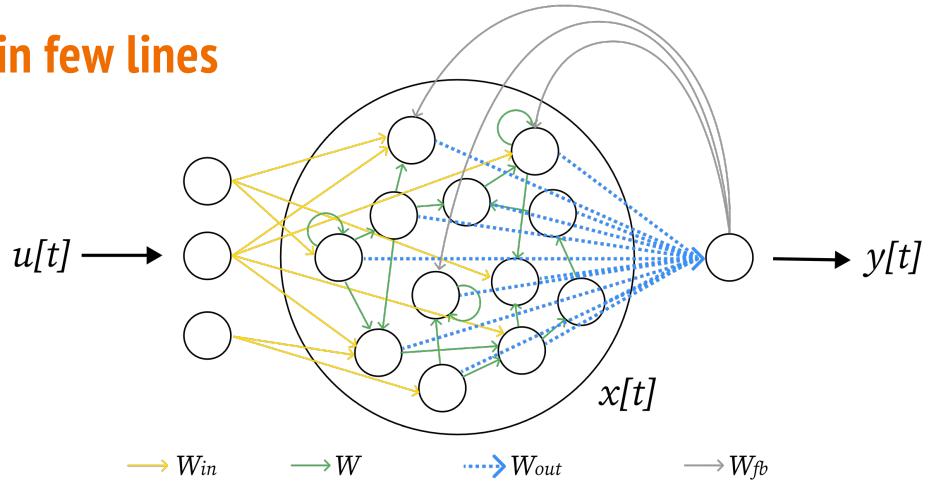


[Image source "star history"](#)

## Node creation

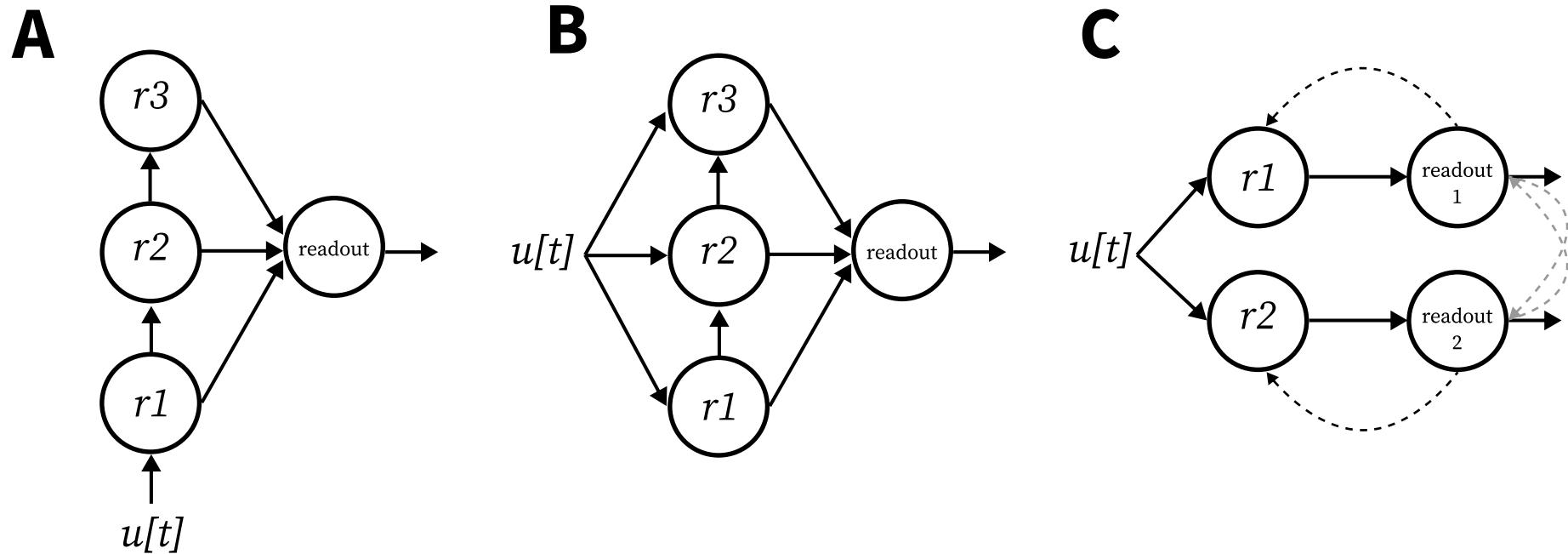
A *node* is an independent operator, that applies a function on some data, potentially in a recurrent way.

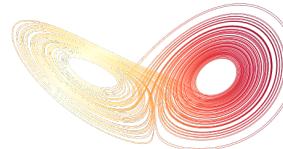
## ReservoirPy in few lines



```
1 from reservoirpy.nodes import Reservoir, Ridge  
2  
3 reservoir = Reservoir(100, lr=0.3, sr=1.25, input_scaling=0.1)  
4 readout = Ridge(ridge=1e-5)
```

## Enable complex architectures





## 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

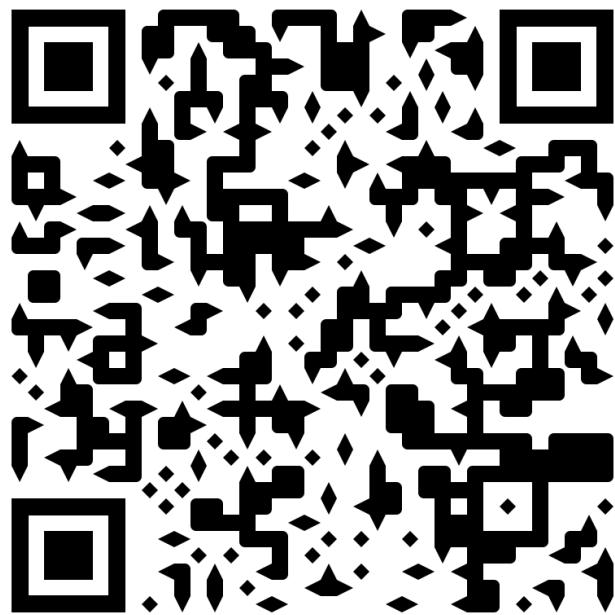
- Echo State Networks (ESN)
- Autoregressive reservoir (NVAR)
- Ridge regression
- FORCE learning
- Intrinsic Plasticity
- Usual reservoir topologies (ring, orthogonal, ...)
- Scikit-learn models, (interface)
- Simulated physical models (optical RC)

New

- ES<sup>2</sup>N
- Local Plasticity (github contribution @Finebouche)
- Arbitrary delays (between nodes)
- LIF spiking neurons
- Cluster & Small-world topologies
- **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/...](https://github.com/reservoirpy/presentations/)

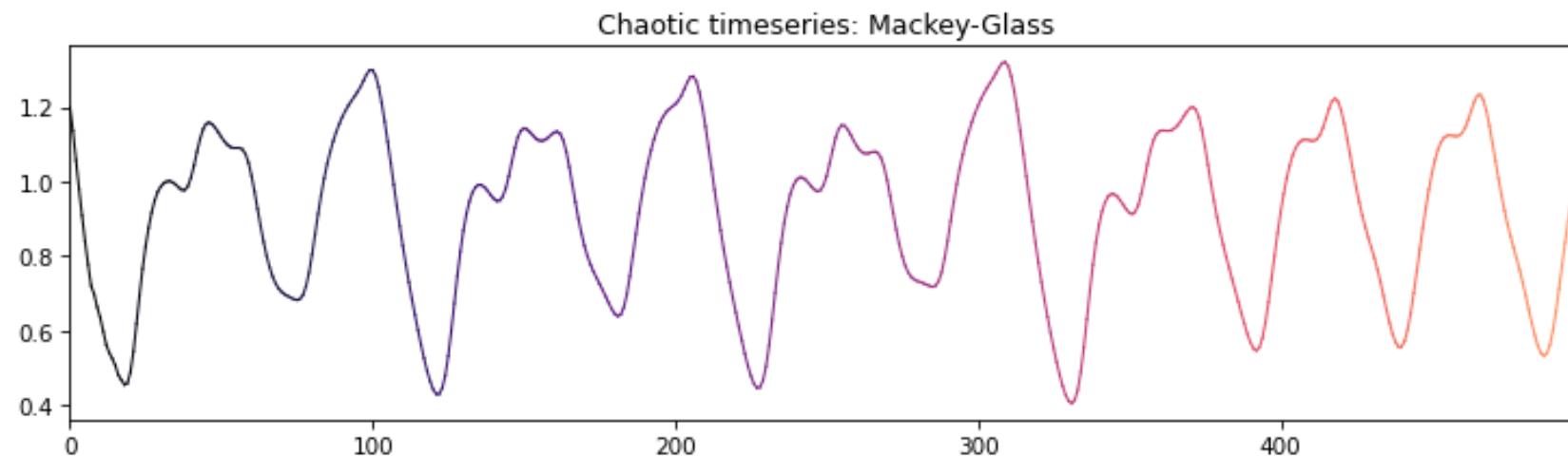
The screenshot shows a GitHub repository page for the user 'reservoirpy' named 'presentations'. The 'Code' tab is selected. The repository path is 'presentations / ECML-PKDD-2025'. A commit by 'PAUL-BERNARD' was made 4 minutes ago, adding the 'ECML-PKDD-2025' folder. The commit message is 'Added ECML-PKDD-2025'. Below the commit, there is a table showing three files: '...', 'Introduction.ipynb', and 'interactive\_report.py'. The 'Introduction.ipynb' file is highlighted with a purple border, indicating it is the target of the QR code.

Name	Last commit message	Last commit date
...		
Introduction.ipynb	Added ECML-PKDD-2025	4 minutes ago
interactive_report.py	Added ECML-PKDD-2025	4 minutes ago

## 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  
(2000, 1) Python
```

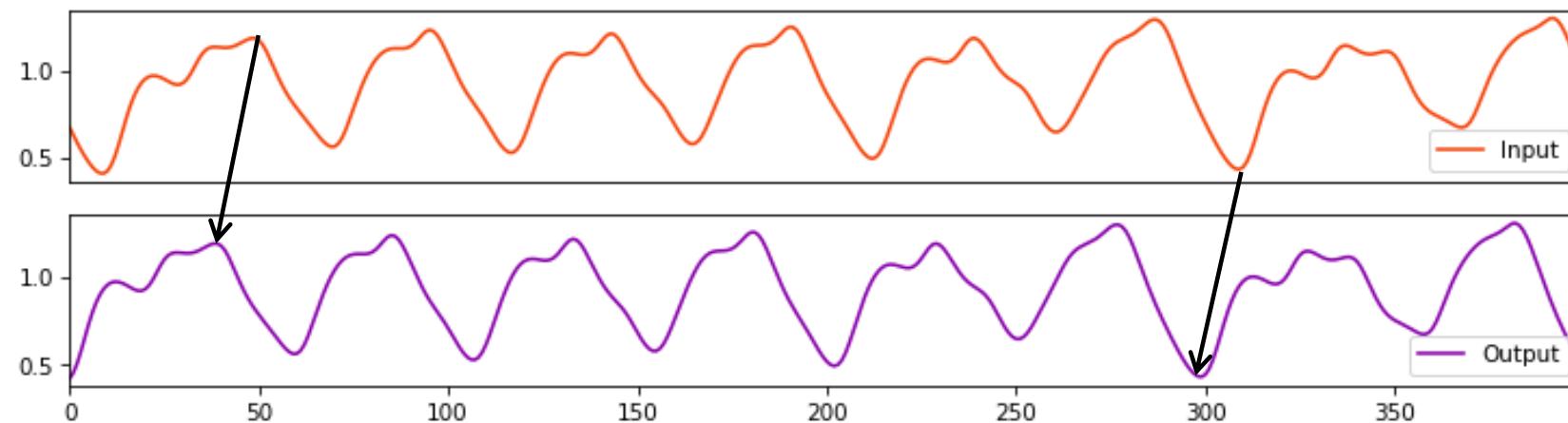


## Data pre-processing

```
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

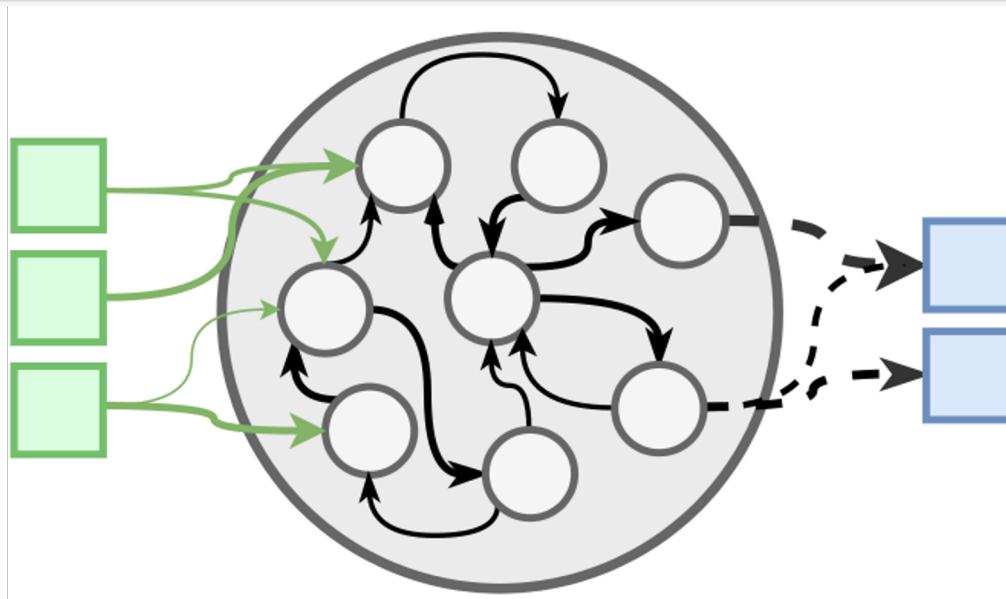


## Echo State Network creation

```
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

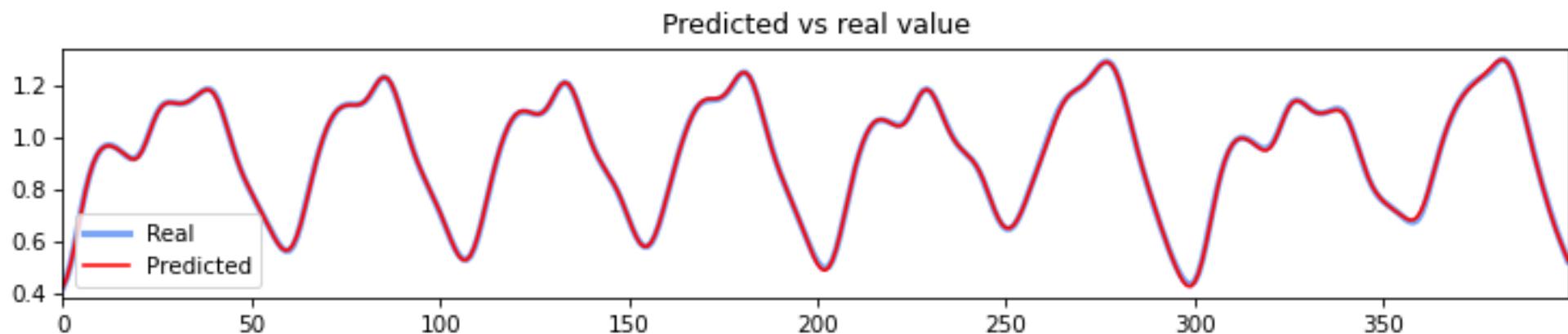


## Fit and run the model

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

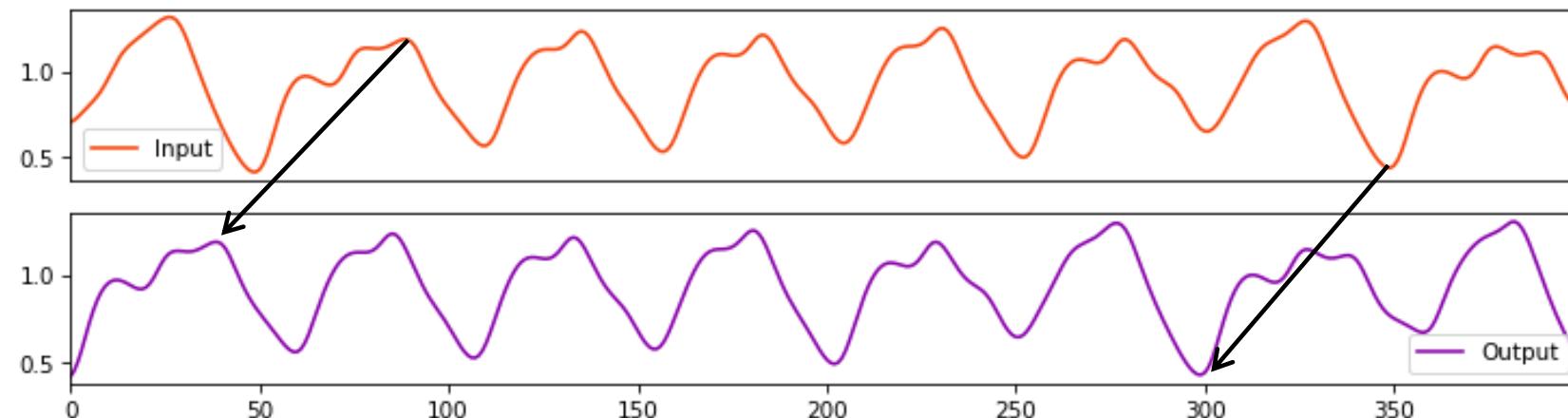
✓ 0.5s

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
```

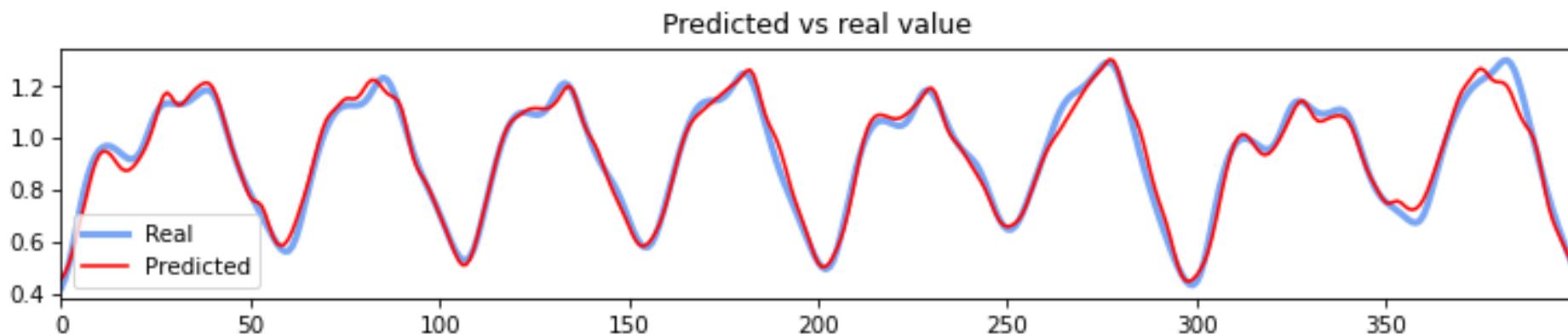


## ... what about a t+50 prediction? The result

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

✓ 0.4s

Python



# Summary

```
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?

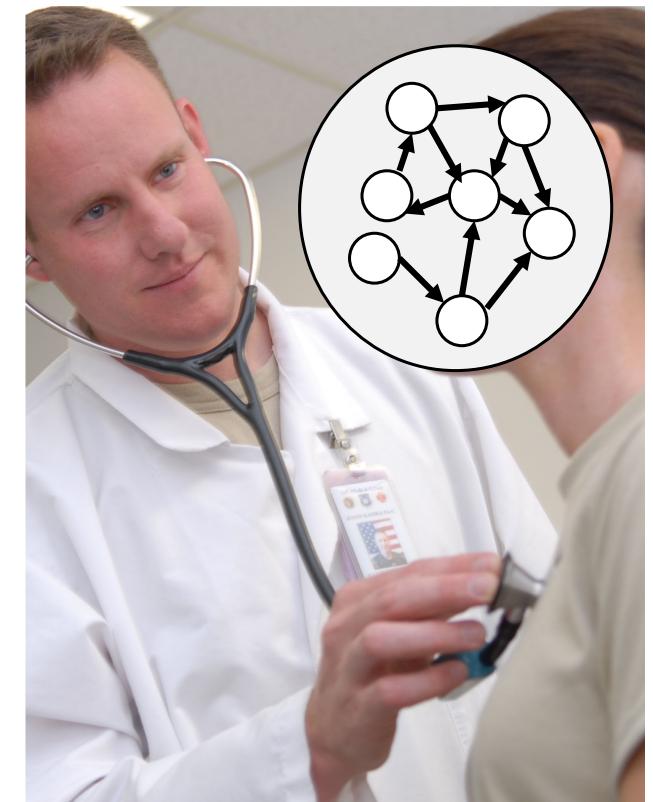
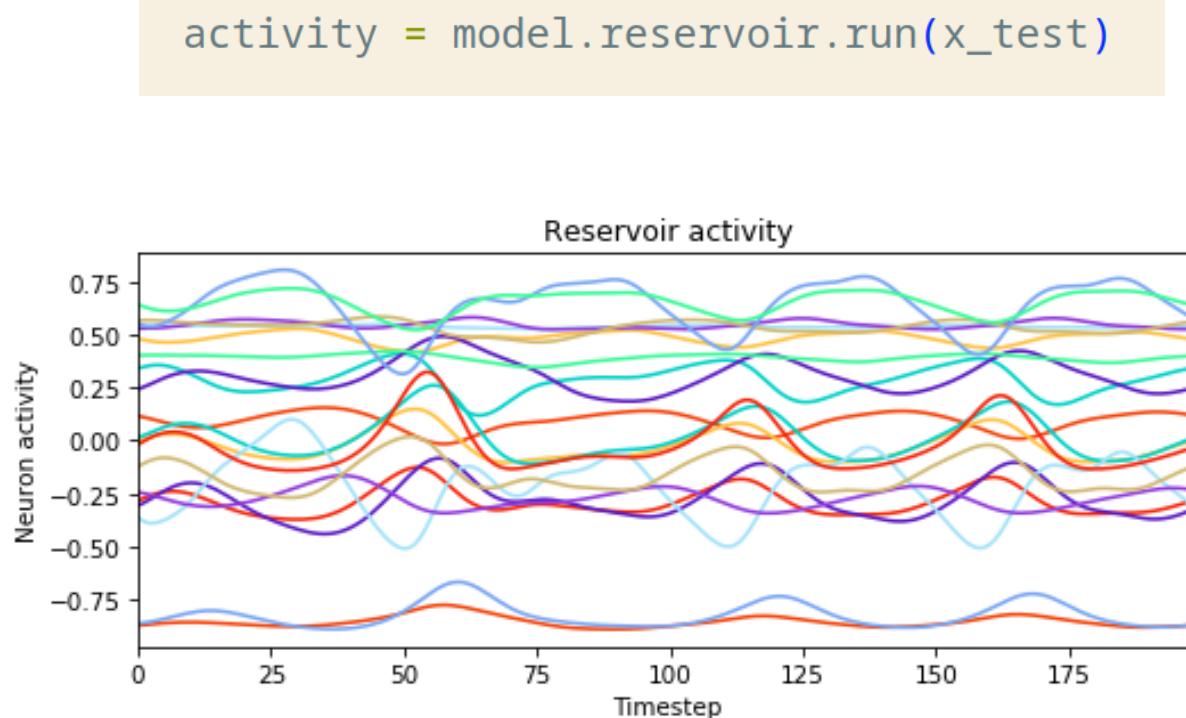


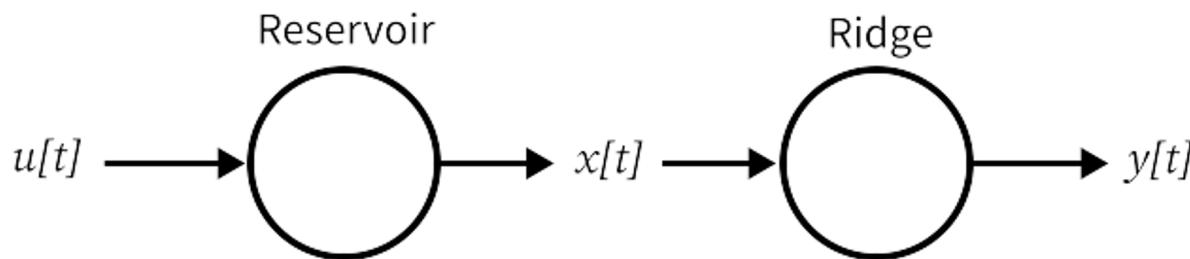
Image credits : Wikimedia

## The nodes

```
from reservoirpy.nodes import Reservoir, Ridge  
  
reservoir = Reservoir(units=10)  
readout = Ridge(ridge=1e-4)  
  
model = reservoir >> readout
```

✓ 0.0s

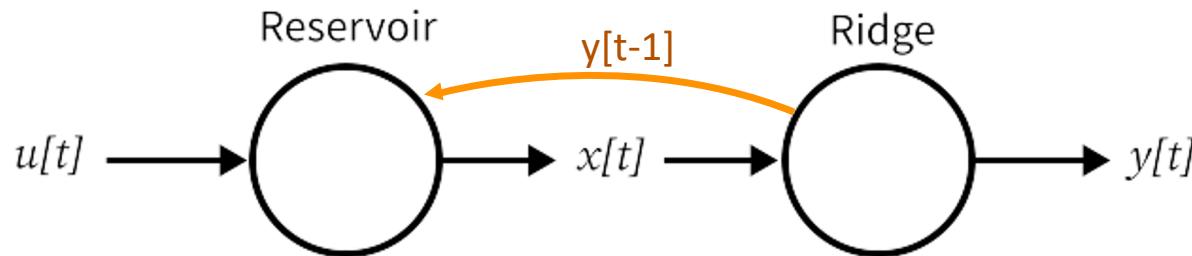
Python



## Feedback connections

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

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



# The scikit-learn node

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

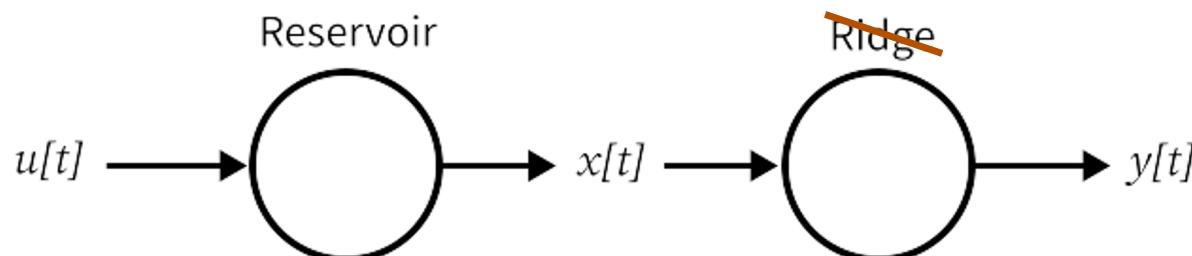
classifier = ScikitLearnNode(RidgeClassifier)
logistic_regressor = ScikitLearnNode(LogisticRegression)

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

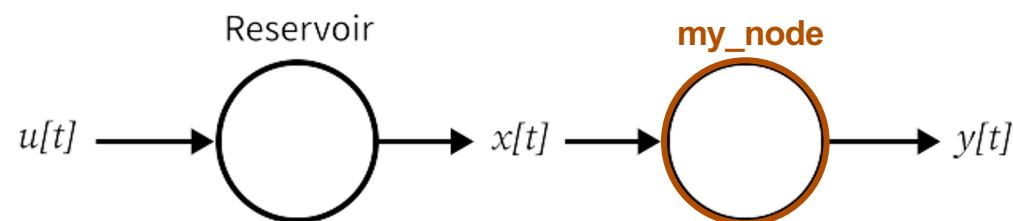
✓ 0.0s

Python

Any scikit-learn model



## Create custom node



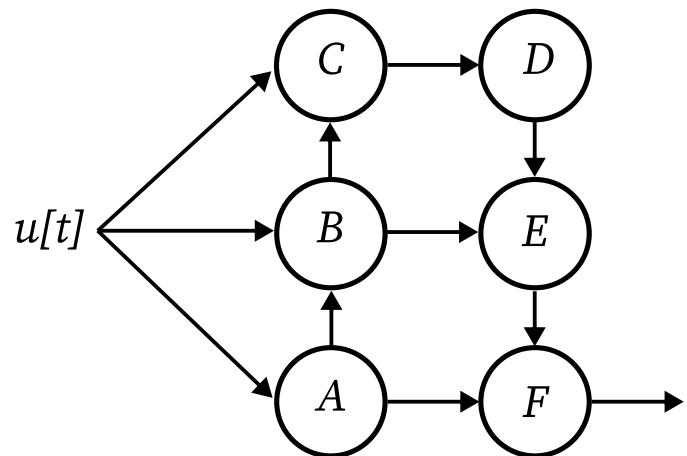
```
from reservoirpy import Node

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

    def initialize(self, x):
        ...
        ...

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

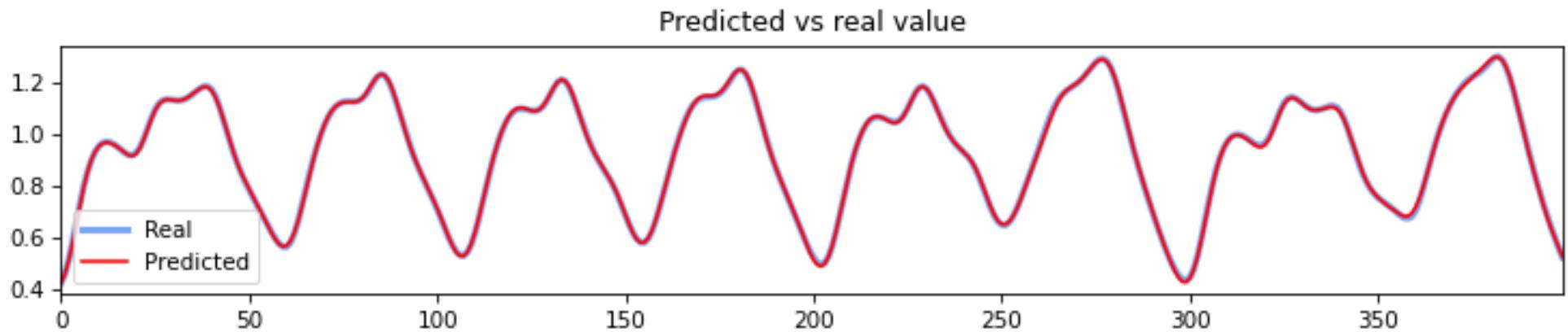
## Enable complex architectures



```
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
```

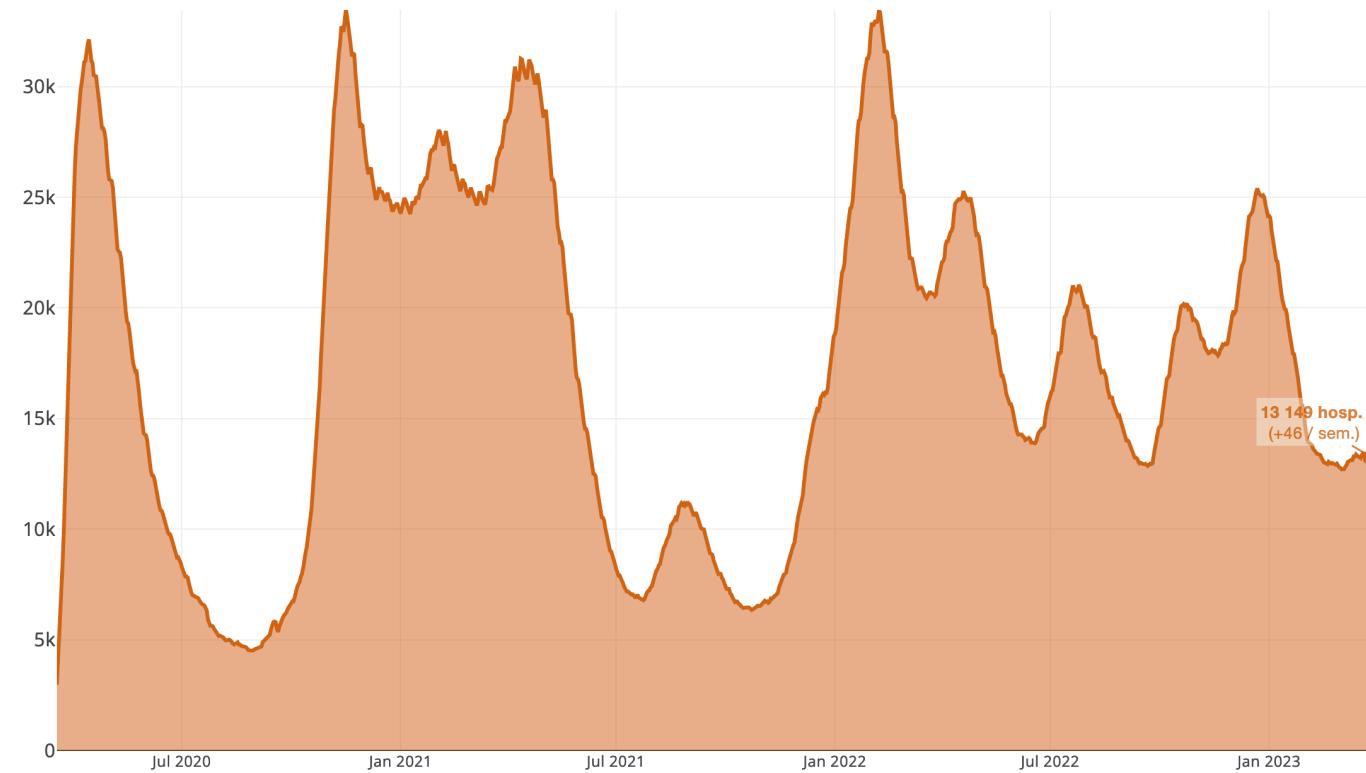
Few tasks/examples  
done with Reservoirpy

# Time series prediction

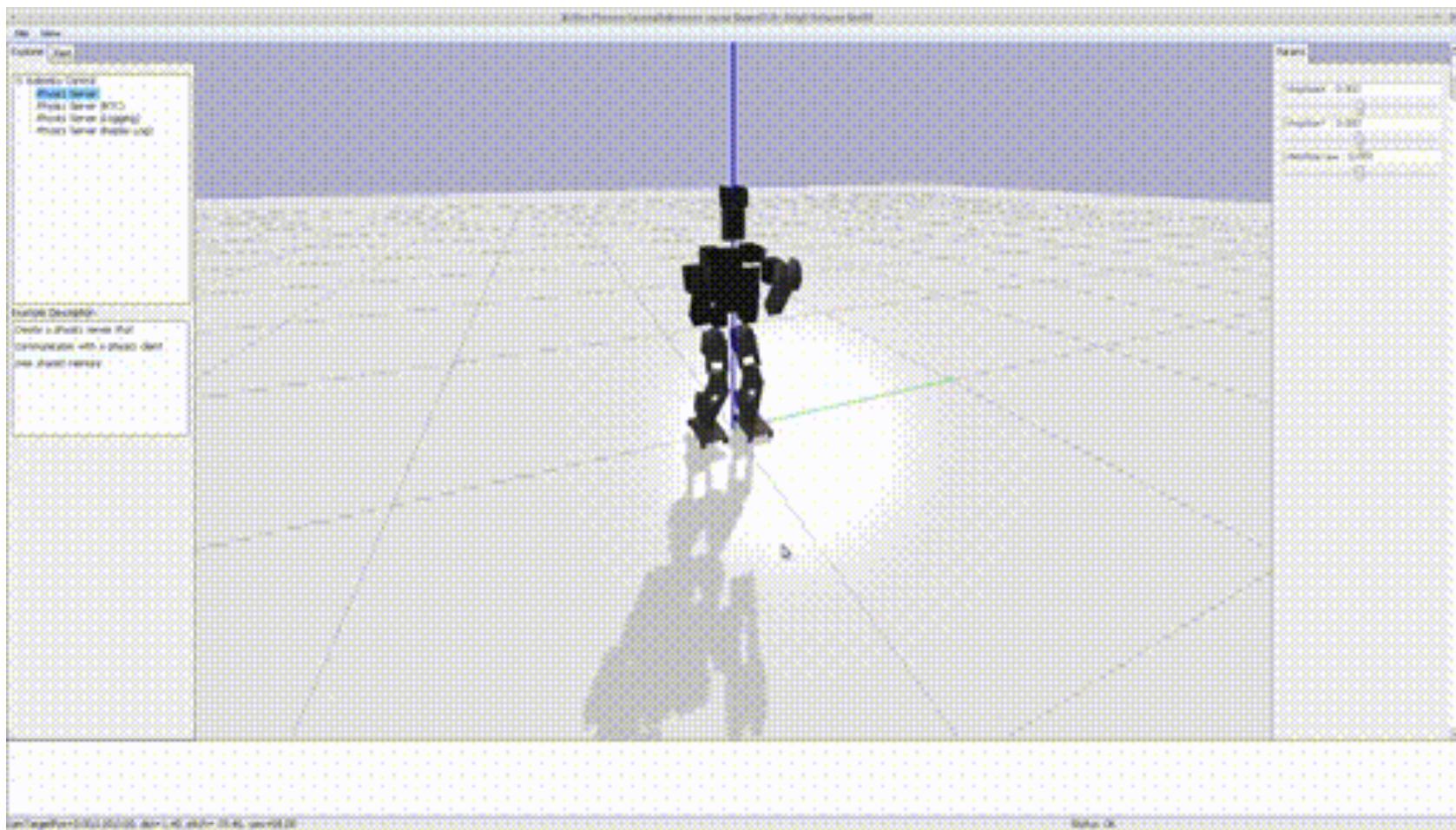




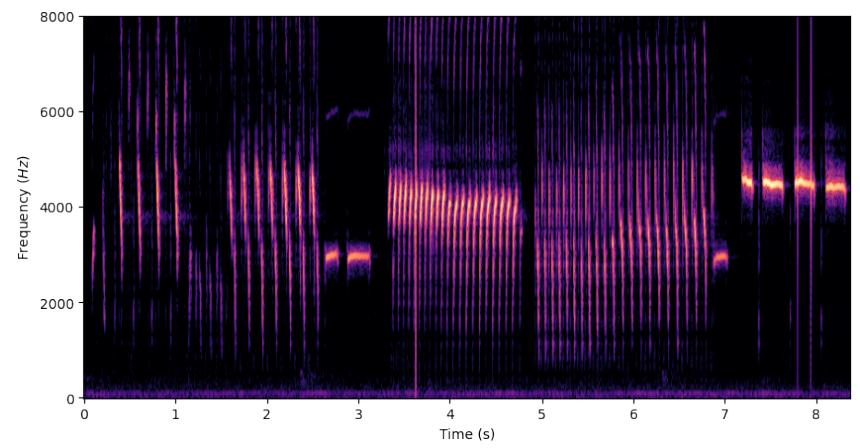
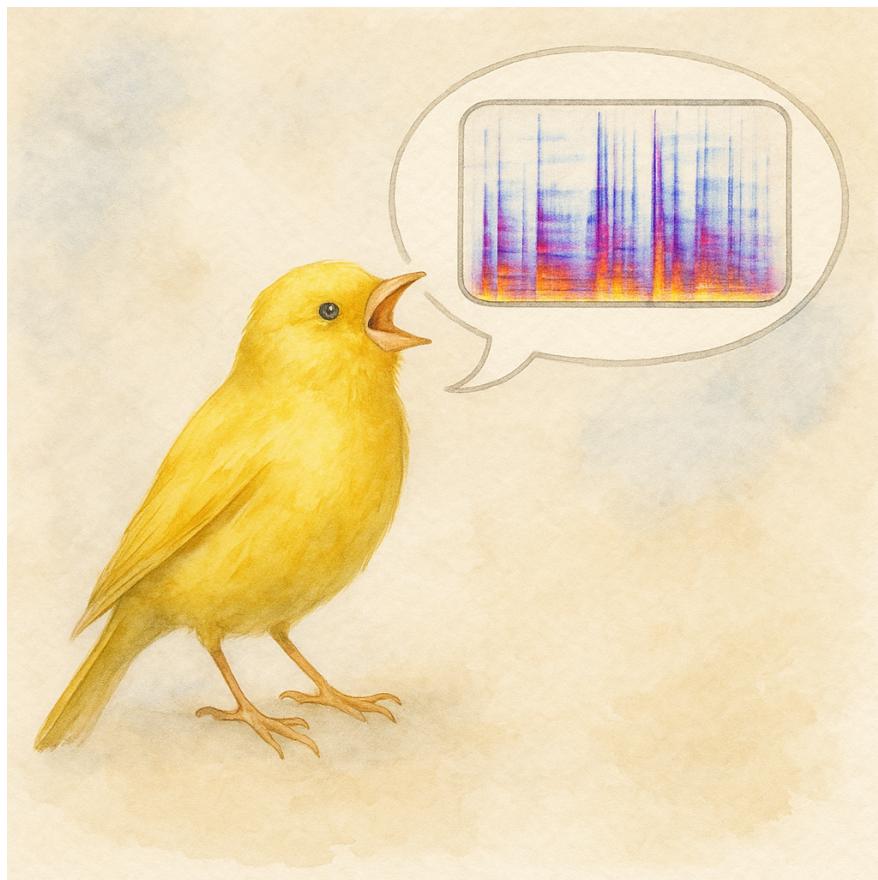
## 2 week forecasting of COVID hospitalisations



# Event classification / prediction



# Sound classification



# Why does it work?

# Why does it work?



# Why does it work?

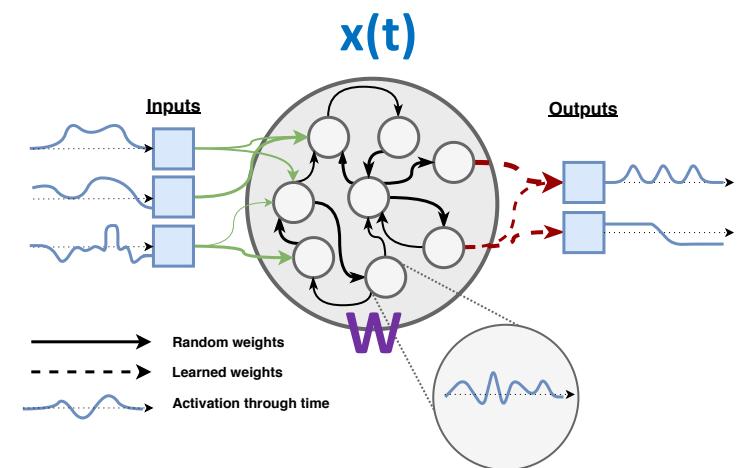


# Why does it work?



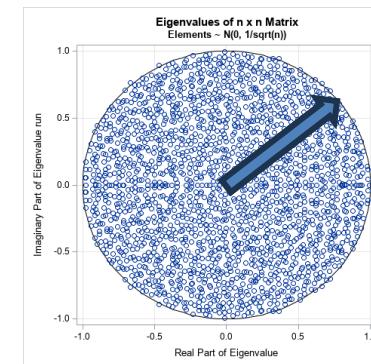
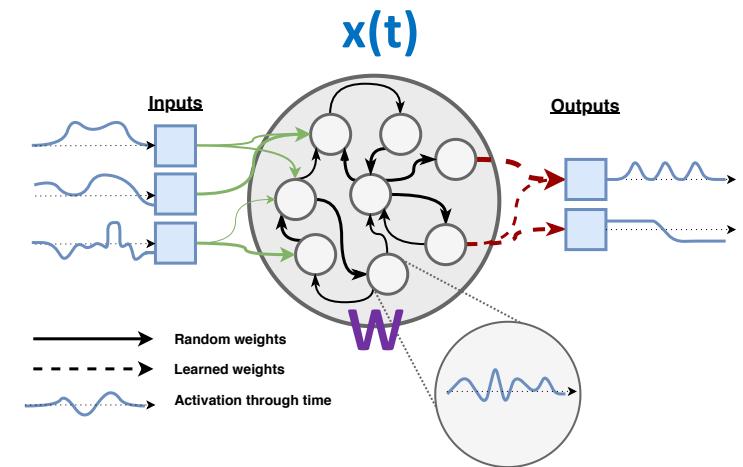
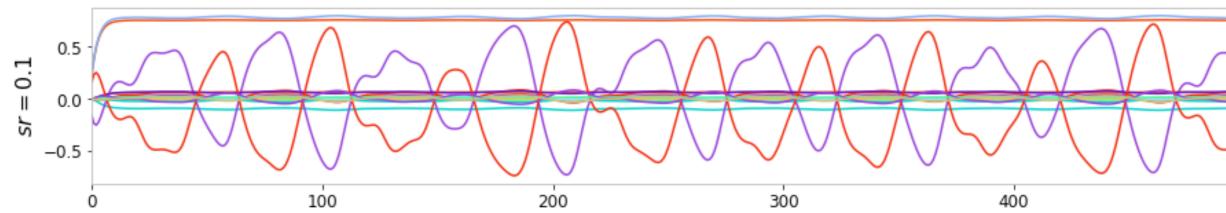
# Spectral Radius (SR)

= scaling of recurrent connections



# Spectral Radius (SR)

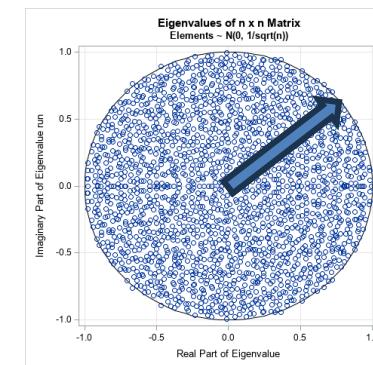
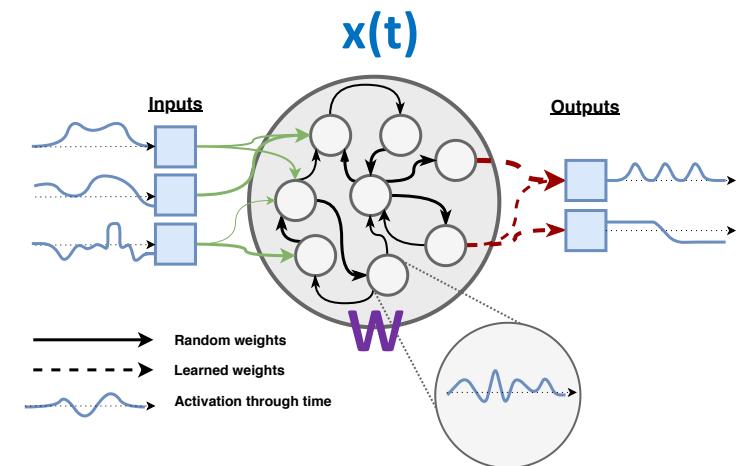
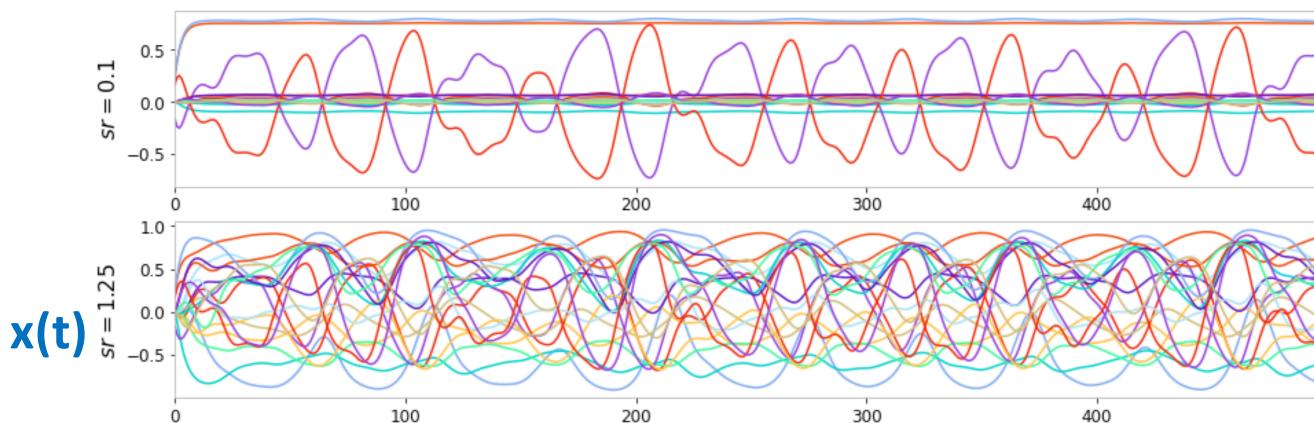
= scaling of recurrent connections



Spectral radius of matrix  $W$

# Spectral Radius (SR)

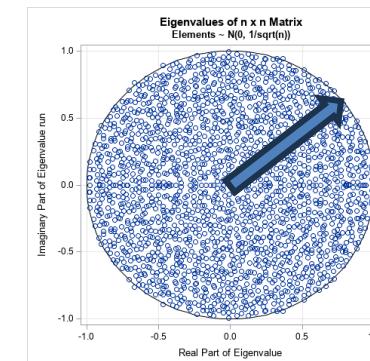
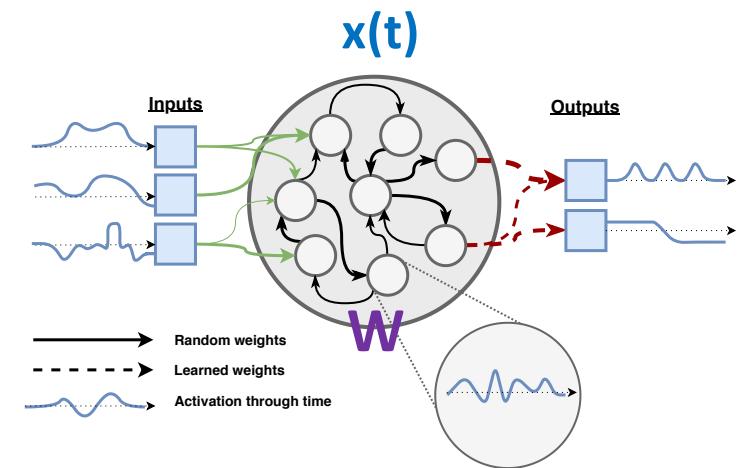
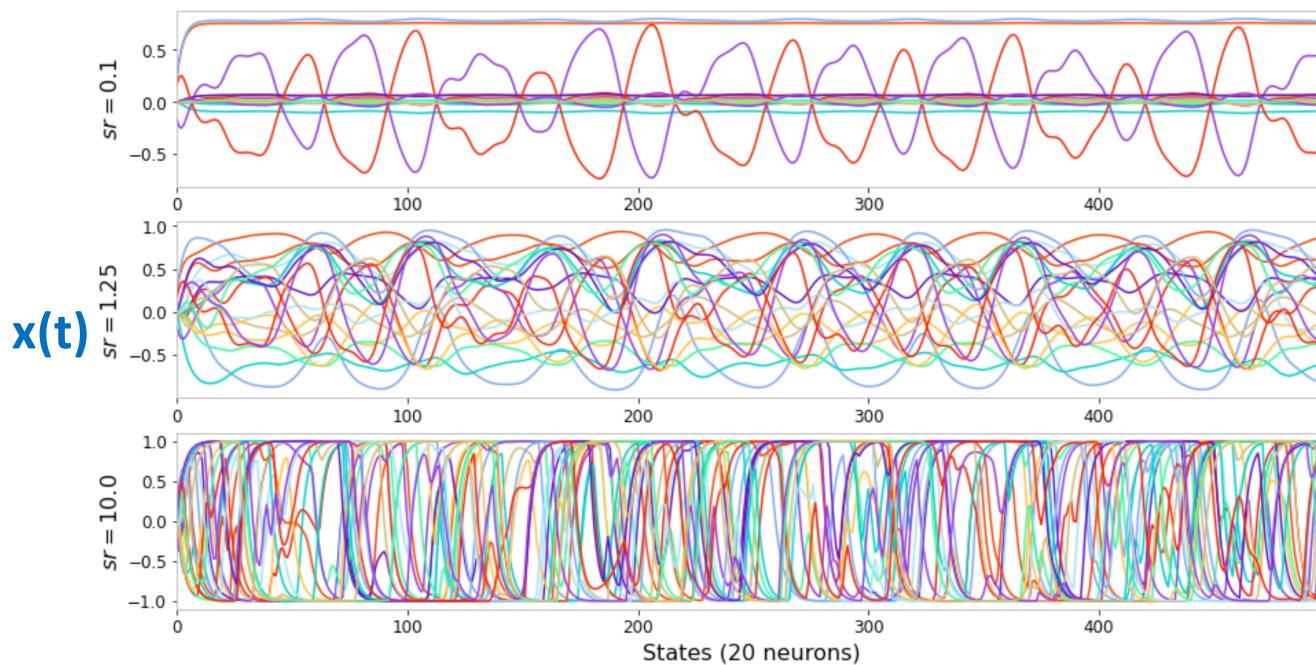
= scaling of recurrent connections



Spectral radius of  
matrix  $W$

# Spectral Radius (SR)

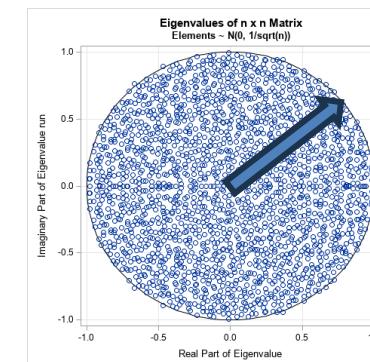
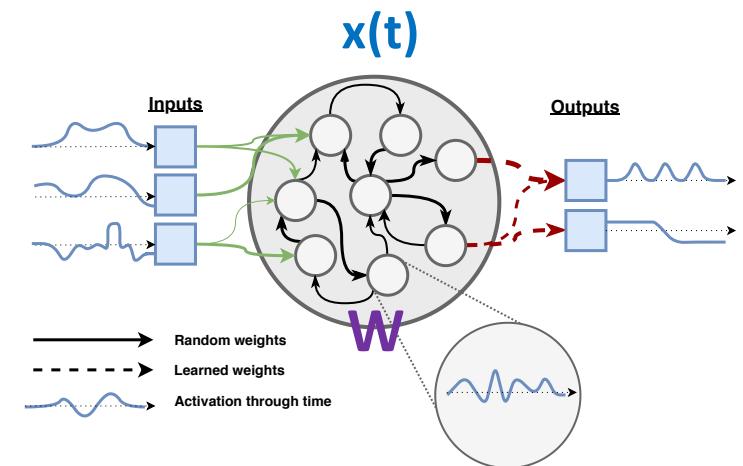
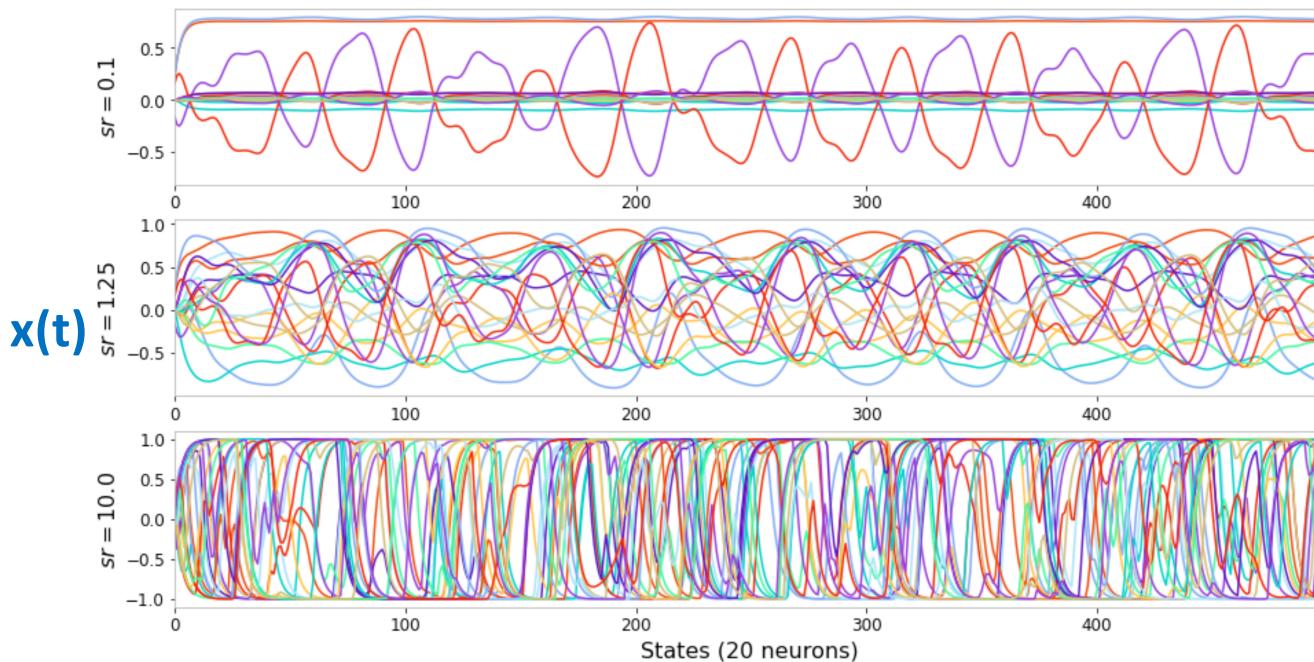
= scaling of recurrent connections



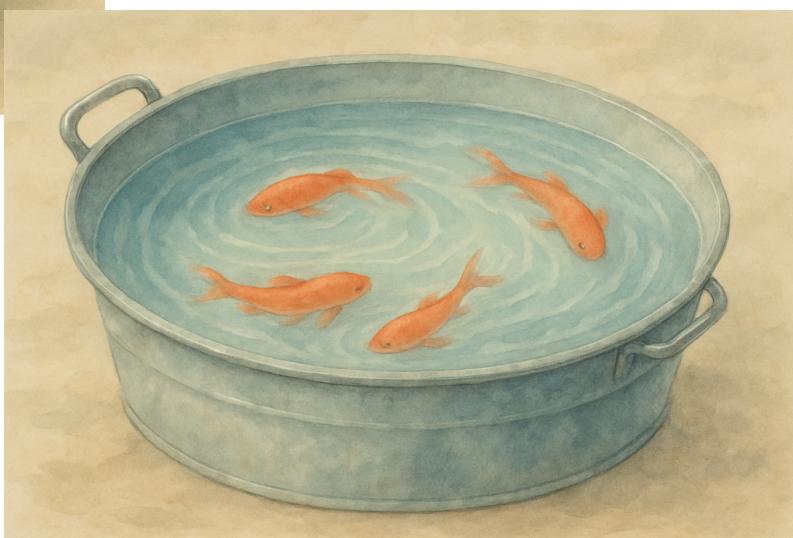
Spectral radius of matrix W

# Spectral Radius (SR)

= scaling of recurrent connections



Spectral radius of matrix  $W$



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

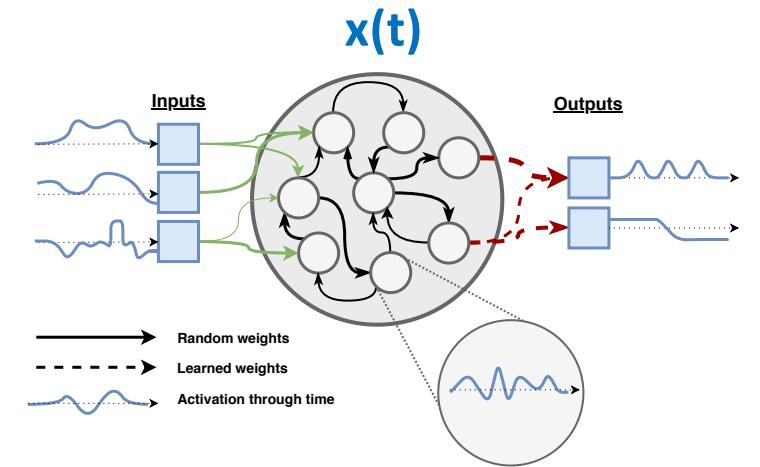
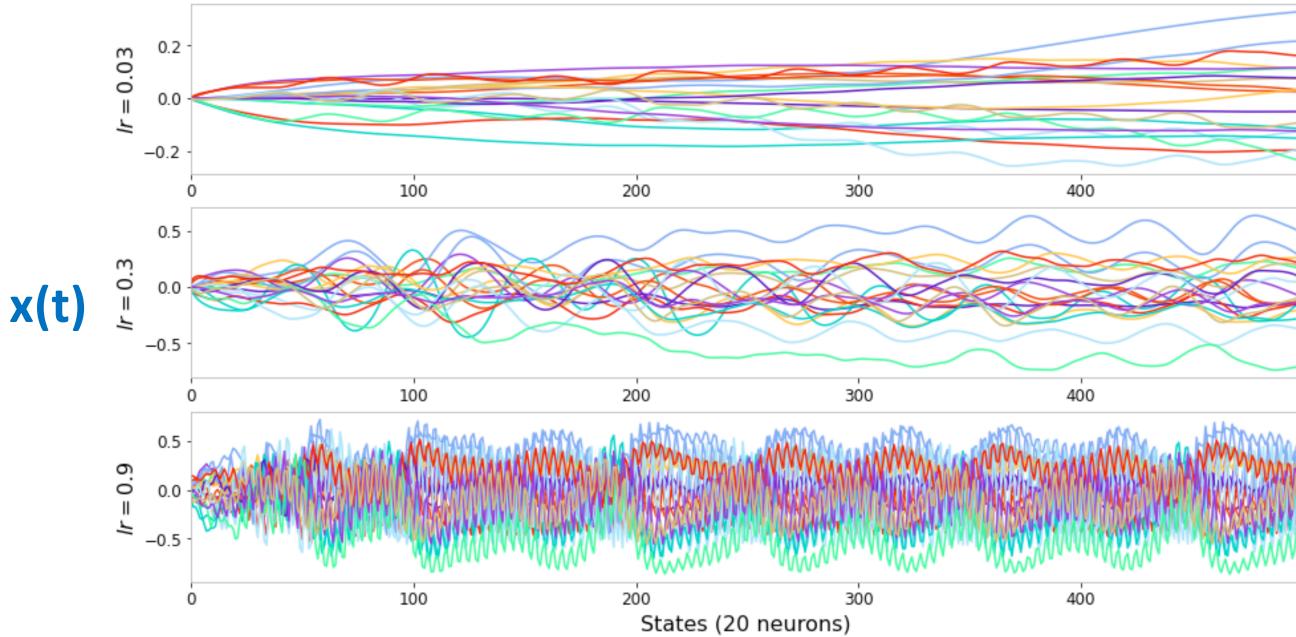


For instance, are there other hyperparameters  
that could “tame chaos”?



# Leak(ing)-rate (LR)

= inverse of a time constant



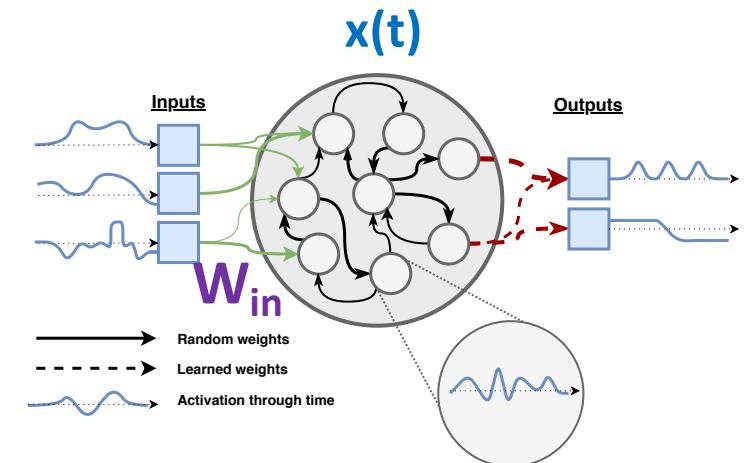
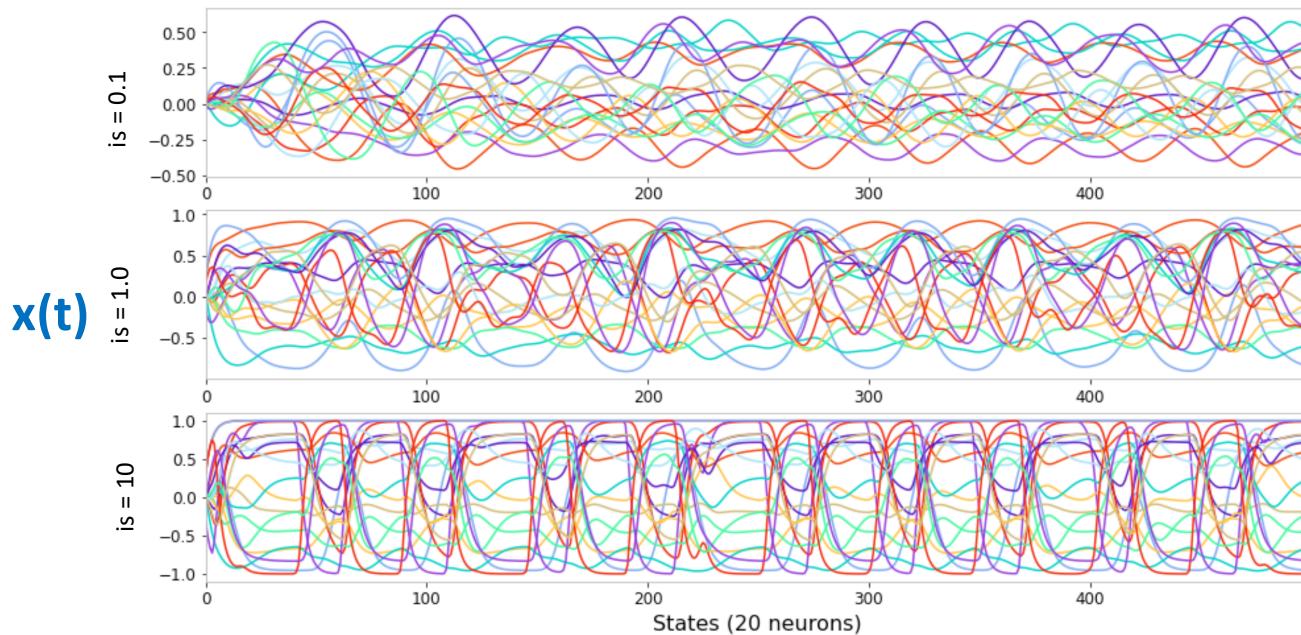
The leaking rate ( $\alpha$ ) controls the "memory feedback" of the ESN. The ESN states are indeed computed as:

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

- + leaking rate → **low inertia**, little memory of previous states
- - leaking rate → **high inertia**, big memory of previous states

# Input Scaling (IS)

= scaling of input connections



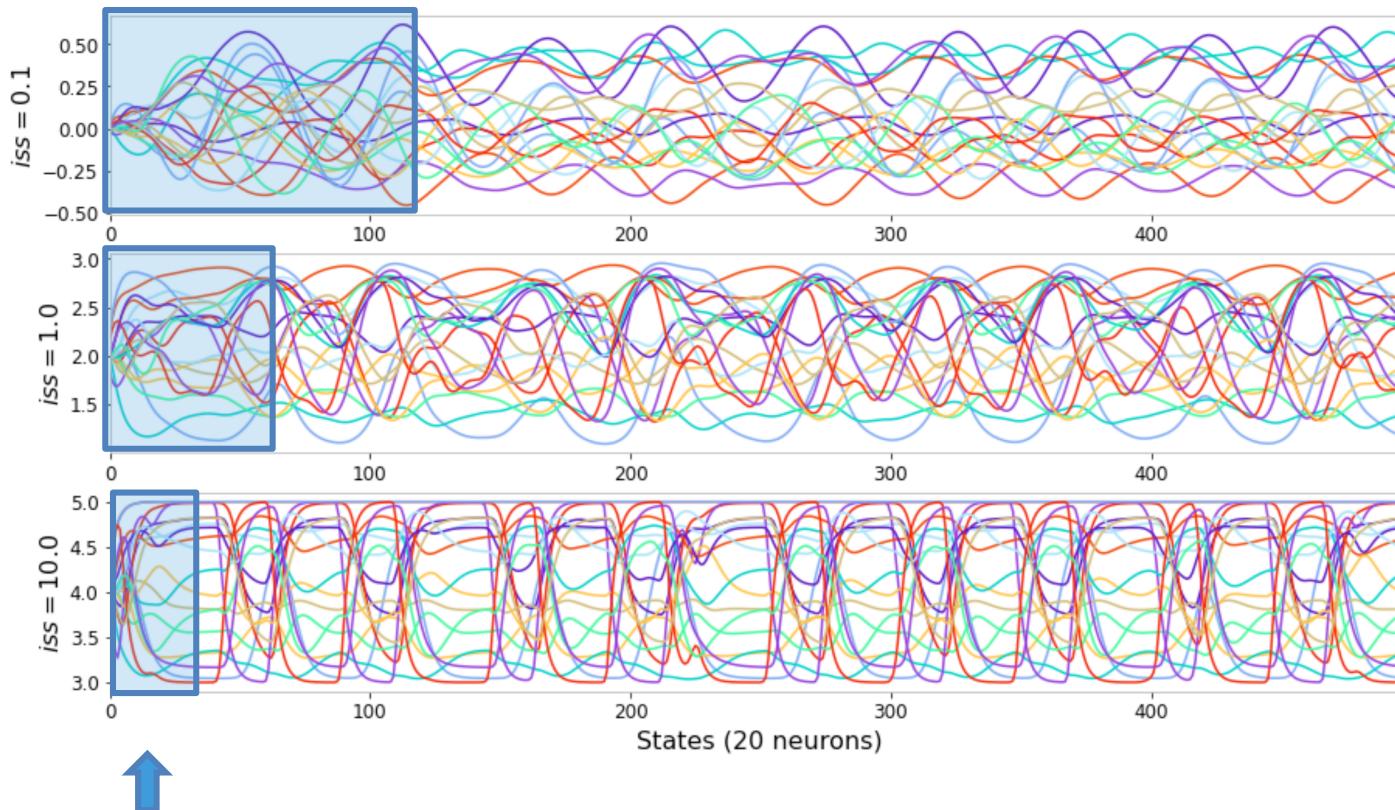
The input scaling controls how the ESN interact with the inputs. It is a coefficient applied to the input matrix  $W_{in}$ .

- + input scaling → **input-driven** activities
- - input scaling → **free** activities

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

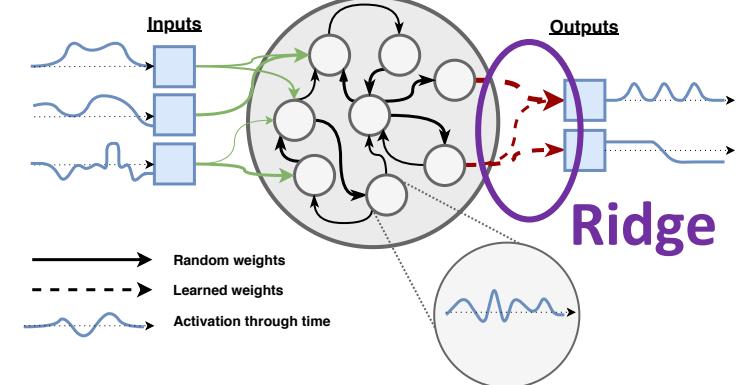
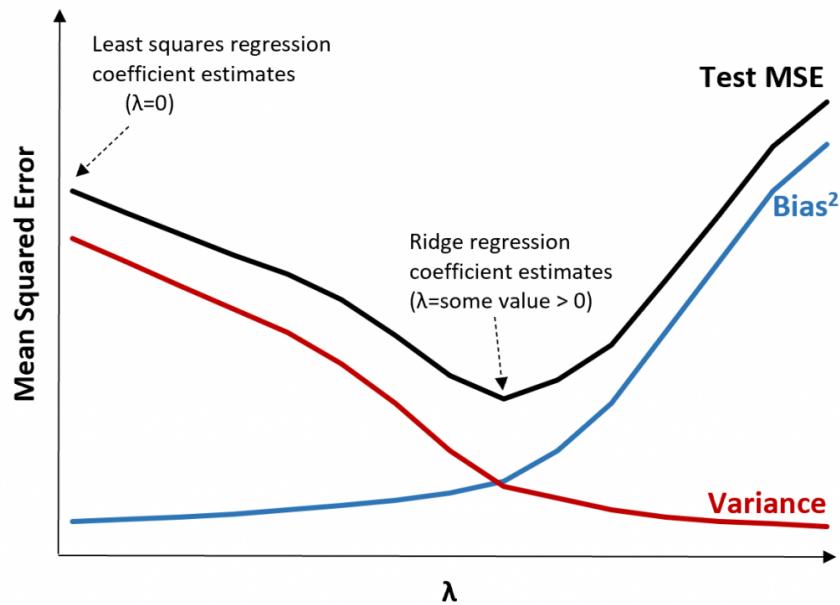
# Warm(ing)-up

= transitory period during the « heating » of the system



# Ridge

= regularisation of the read-out linear regression



# Tools for Hyperparameter (HP) optimization

# Let's make hyperparameter search easy and reproducible!

Don't try at home without proper tools!

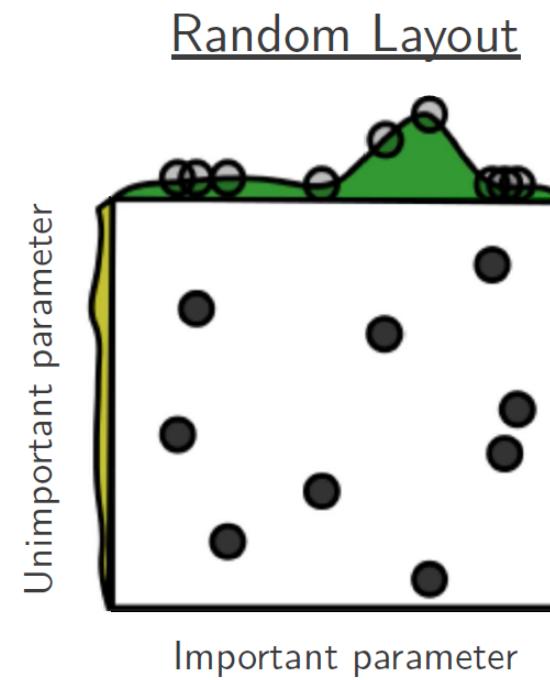
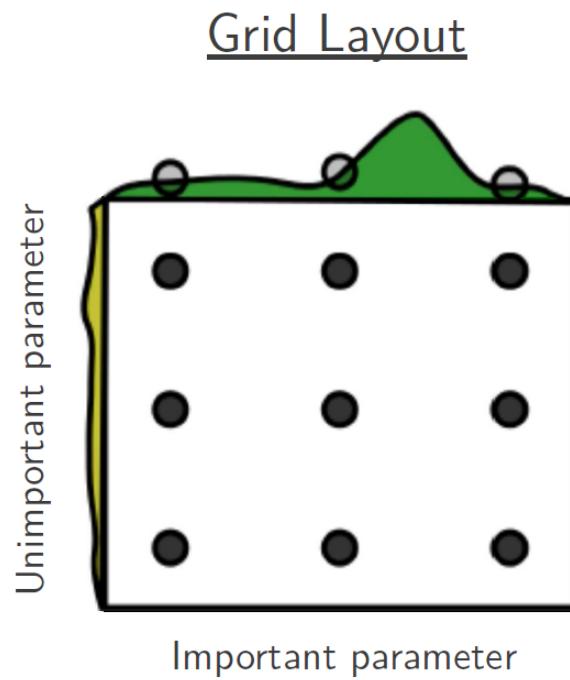


(⚠️ don't use grid search!)

# 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?

Most important ones

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

# Which Hyperparameters (HP) should you tune?

Most important ones

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

```
# 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

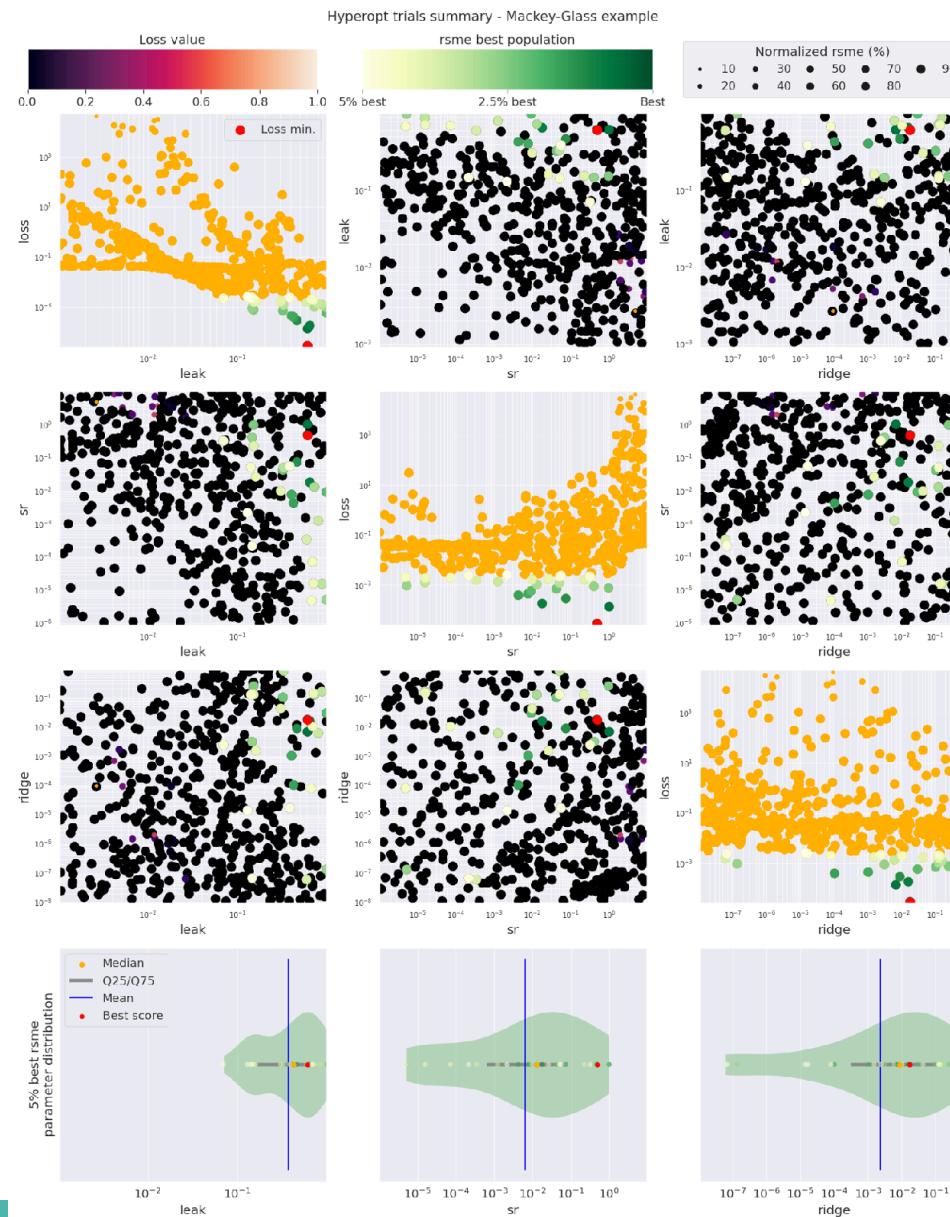
```
# Build your model given the input parameters
reservoir = Reservoir(
    units=N,
    sr=sr,
    lr=lr,
    input_scaling=input_scaling,
    seed=variable_seed
)
readout = Ridge(ridge=ridge)

model = reservoir >> readout
```

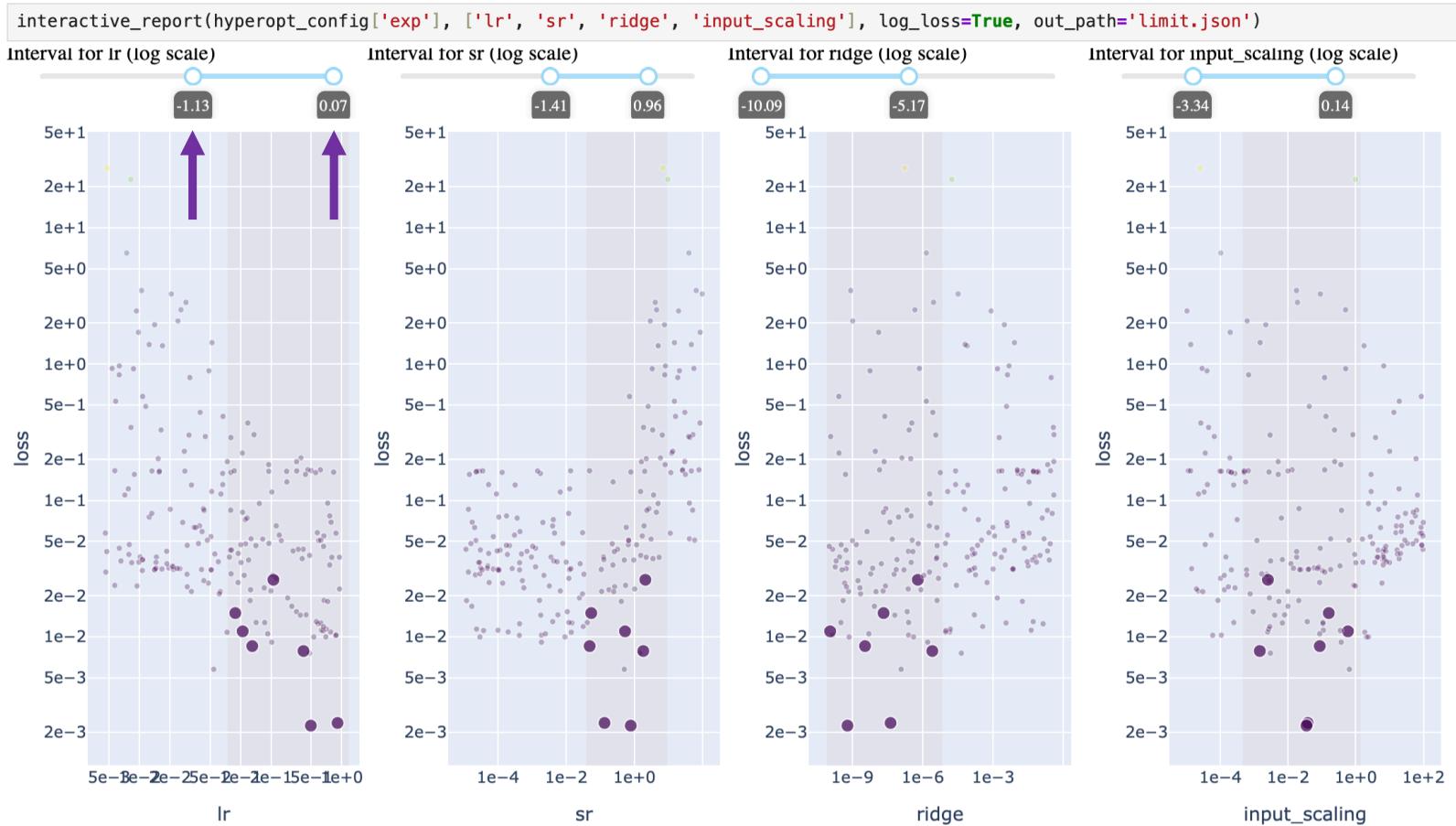
```
hyperopt_config = {
    "exp": "hyperopt-doublescroll",           # the experimentation name
    "hp_max_evals": 200,                      # the number of different sets of parameters hyperopt has to try
    "hp_method": "random",                    # the method used by hyperopt to chose those sets (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
    }
}
```

Define logarithmic ranges  
& fix some HP  
(e.g. N and input\_scaling)

# Graphical view of hyperparameter optimization

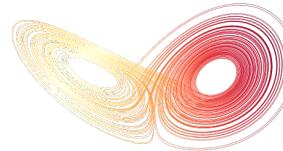


# Hyperparameter Dynamic Exploration



→ quick & robust selection  
of hyperparameters ranges

# Discussion



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

# Perspectives

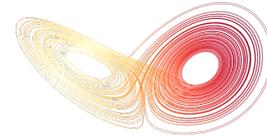
## Upgrade framework capabilities

- **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**

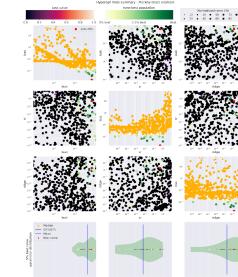
# Summary on reservoirpy



- 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>



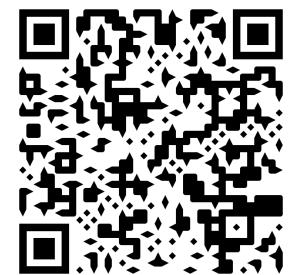
Many thanks to  
Nathan Trouvain  
& Paul Bernard  
+ all developers



Inria

LaBRI

université  
de BORDEAUX



Looking for a post-doc!

reservoirpy



xavier.hinaut@inria.fr & paul.bernard@inria.fr

@neuronalX & @PAUL-BERNARD  
 @hinaut @reservoirpy

# Thank you!

- ➡ Share your models in [ReservoirPy](#)
- ➡ Ideas of new features to implement?

- ➡ Add papers on the  
**Reservoir “Awesome list”**

[github.com/reservoirpy/awesome-reservoir-computing](https://github.com/reservoirpy/awesome-reservoir-computing) 150



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