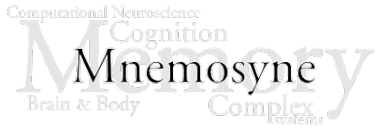
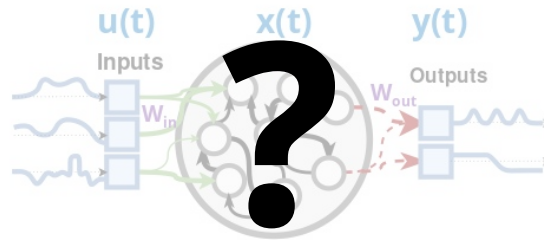
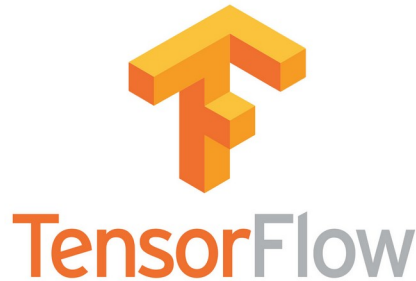


Reservoir Computing : passage à la pratique avec **ReservoirPy**



Introduction

Quel outil Python pour le
reservoir computing ?



Introduction

Library	Language	Main dependency	Last activity	Package	Doc.	Tests	Off.	On.	Fb.	Model type	Deep
PyRCN	Python 3	Scikit-learn	Nov 2022	pip	✓	✓	✓	x	x	ESN	✓
EchoTorch	Python 3	PyTorch	Sep. 2021	pip	✓	✓	✓	x	x	ESN, Conceptors	✓
★Res.Comp.jl ³	Julia	Julia	Sept 2023	Pkg	✓	✓	✓	x	x	ESN	x
Pytorch-esn	Python 3	Pytorch	Feb 2022	x	x	x	✓	x	x	ESN	✓
DeepESN	Matlab	Matlab	Feb. 2019	Matlab	✓	?	✓	x	x	ESN	✓
RCNet	C#	C#	Aug. 2021 (closed)	x	partial	x	✓	x	x	ESN, LSM	x
LSM	Python 3	Nest	Nov. 2020	x	x	x	✓	x	x	LSM	x
Oger	Python 2	mdp	2012 (obsolete)	x	x	✓	✓	✓	✓	LSM, ESN	✓
★reservoirpy (this package)	Python 3	Numpy	Sept 2023	pip	✓	✓	✓	✓	✓	ESN	✓

Table 1: Comparative table of some open source software for Reservoir Computing. This table might not be exhaustive. **Doc.** Complete documentation. **Off.** Offline learning strategies included. **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.

ReservoirPy: un module Python pour le reservoir computing



Basé sur l'écosystème NumPy + SciPy
→ léger

- Facile à prendre en main
- Flexible



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

Format des données

Toutes les données sont des tableaux NumPy.

shape = (timesteps, features)

```
from reservoirpy.datasets import mackey_glass
```

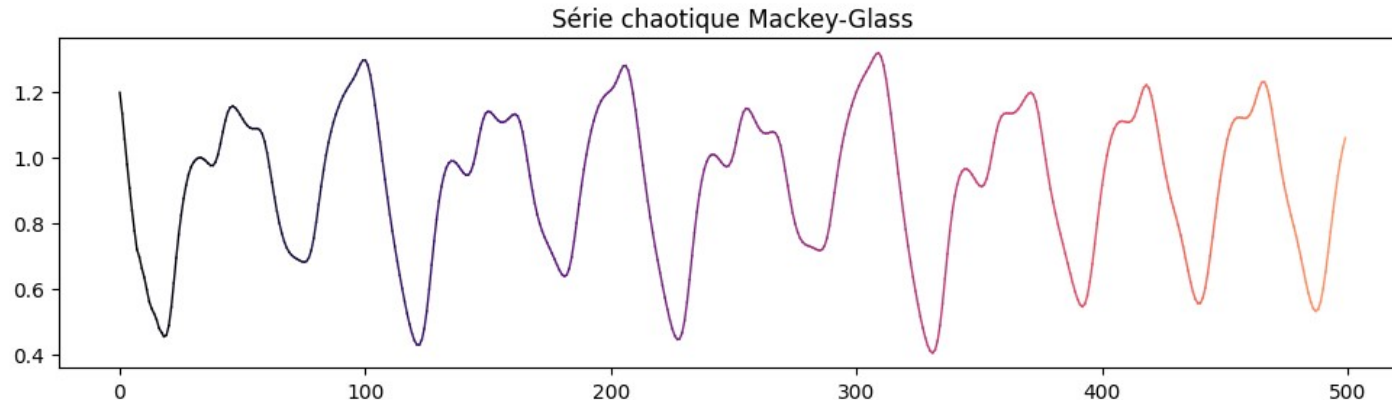
```
timeseries = mackey_glass(n_timesteps=2_000)
```

```
timeseries.shape
```

✓ 0.0s

Python

(2000, 1)

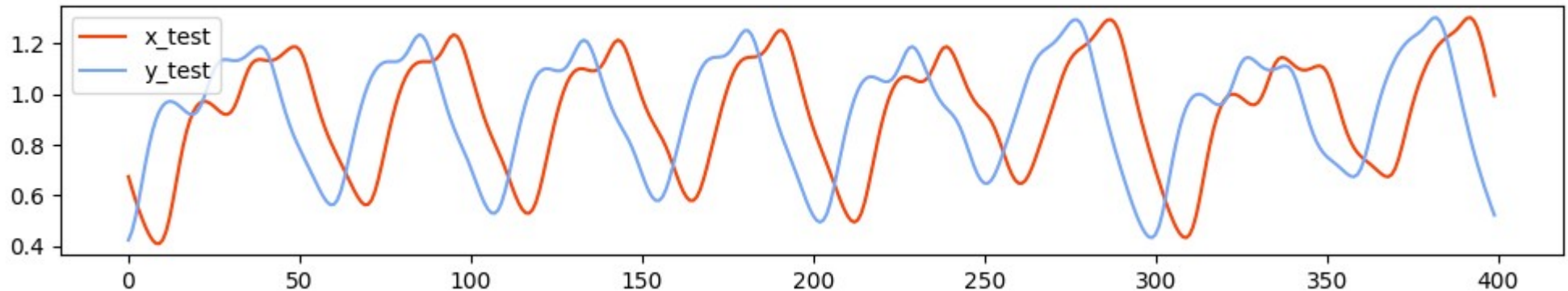


Préparer les données

```
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
```

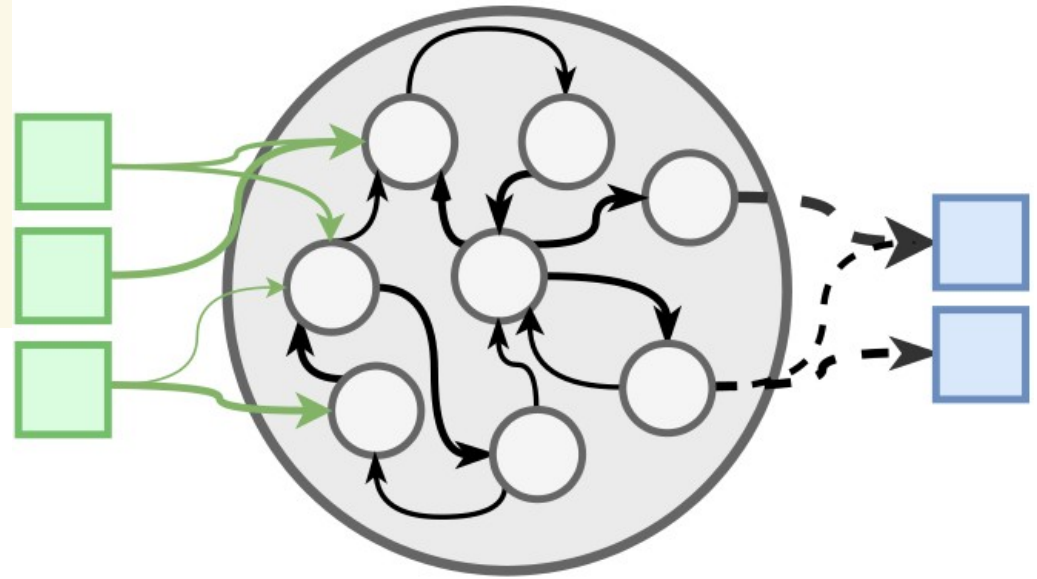
✓ 0.0s

entrées et sorties du modèle



Créer un ESN

```
from reservoirpy.nodes import ESN
model = ESN(
    units=100,
    lr=0.3,
    sr=1.25,
    ridge=1e-8
)
```



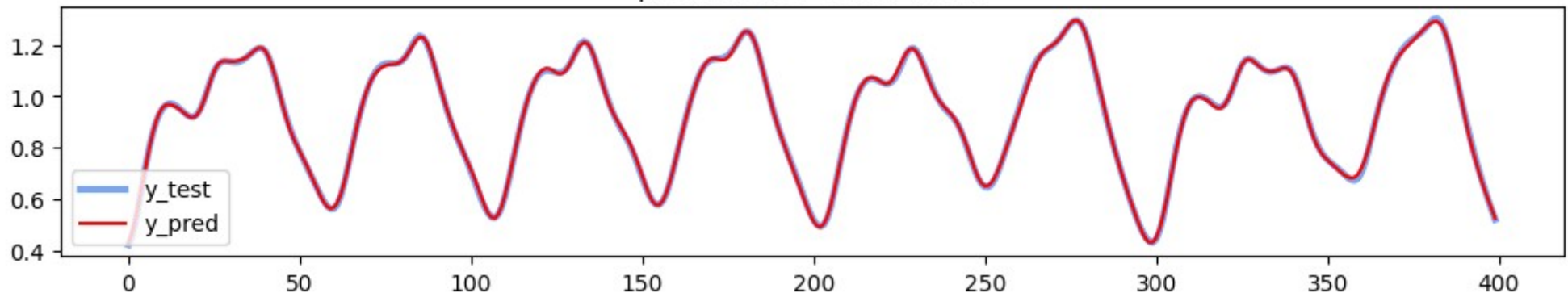
Entraîner et lancer le modèle

```
model.fit(x_train, y_train)  
y_pred = model.run(x_test)
```

✓ 0.4s

Python

prédiction et valeur réelle



... et avec une prédiction à $t+50$?

```
from reservoirpy.datasets import to_forecasting
```

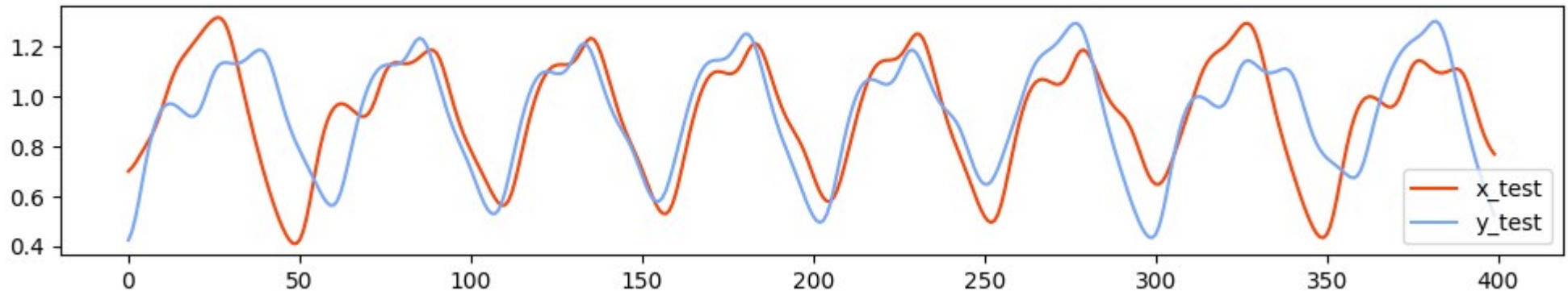
```
dataset = to_forecasting(X, forecast=50, test_size=0.2)
```

```
x_train, x_test, y_train, y_test = dataset
```

✓ 0.0s

Python

entrées et sorties du modèle

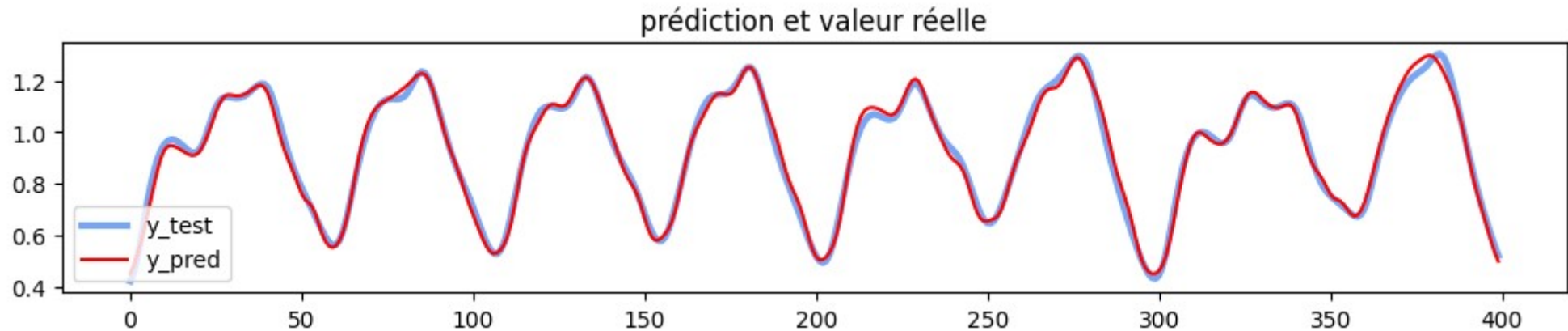


... et avec une prédiction à $t+50$?
le résultat

```
from reservoirpy.datasets import to_forecasting  
  
dataset = to_forecasting(X, forecast=50, test_size=0.2)  
x_train, x_test, y_train, y_test = dataset
```

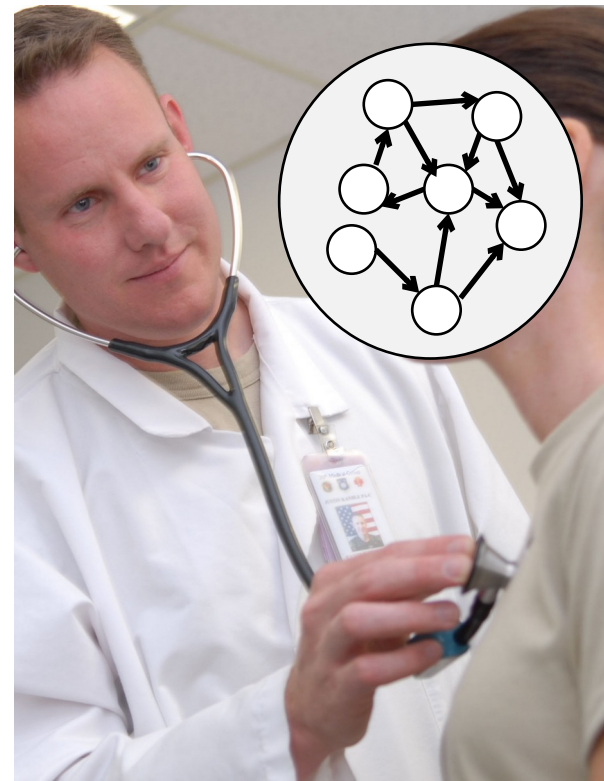
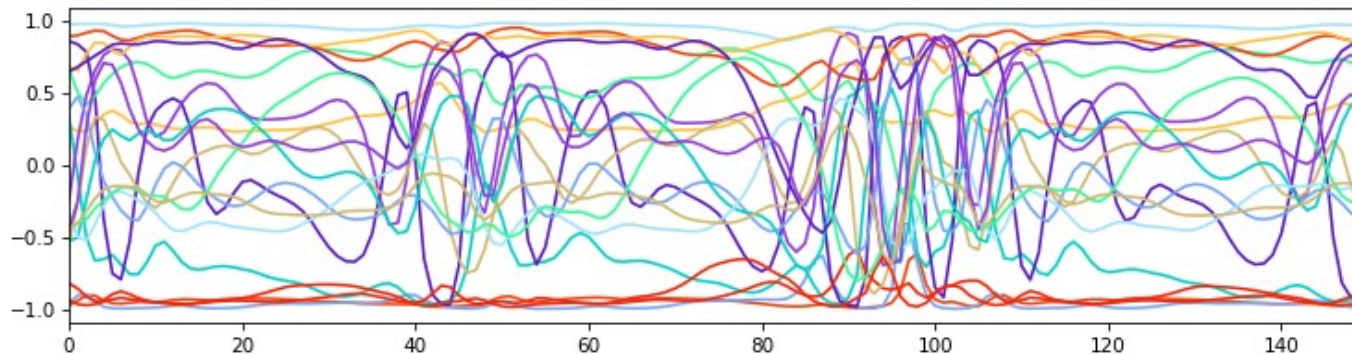
✓ 0.0s

Python



Qu'est-ce qui se passe dans le réservoir ?

```
reservoir = Reservoir(units=20)  
activity = reservoir.run(X_test)
```



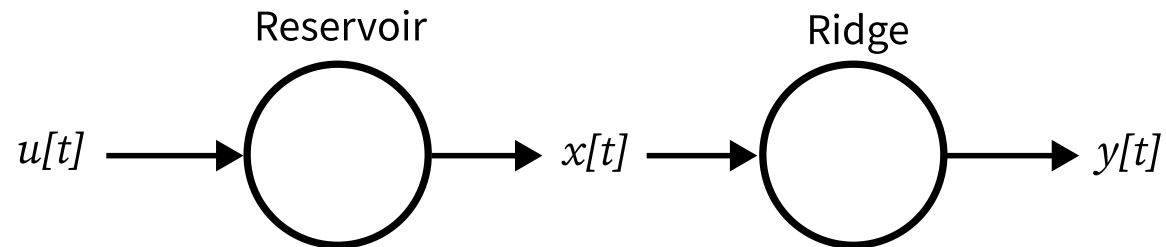
Crédits de l'image : Wikimedia

Récapitulatif

```
from reservoirpy.datasets import (  
    mackey_glass,  
    to_forecasting,  
)  
from reservoirpy.nodes import ESN  
  
timeseries = mackey_glass(n_timesteps=2_000)  
dataset = to_forecasting(timeseries, forecast=100, test_size=0.2)  
x_train, x_test, y_train, y_test = dataset  
  
model = ESN(  
    units=100,  
    lr=0.3,  
    sr=1.25,  
    ridge=1e-8,  
)  
  
model.fit(x_train, y_train)  
y_pred = model.run(x_test)
```

Les nœuds

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

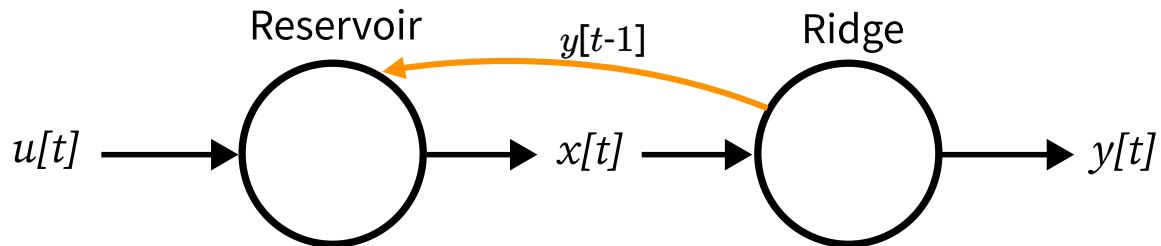


[Mettre tutoriel]

Connexion *feedback*

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

reservoir <<= readout
model = reservoir >> readout
```



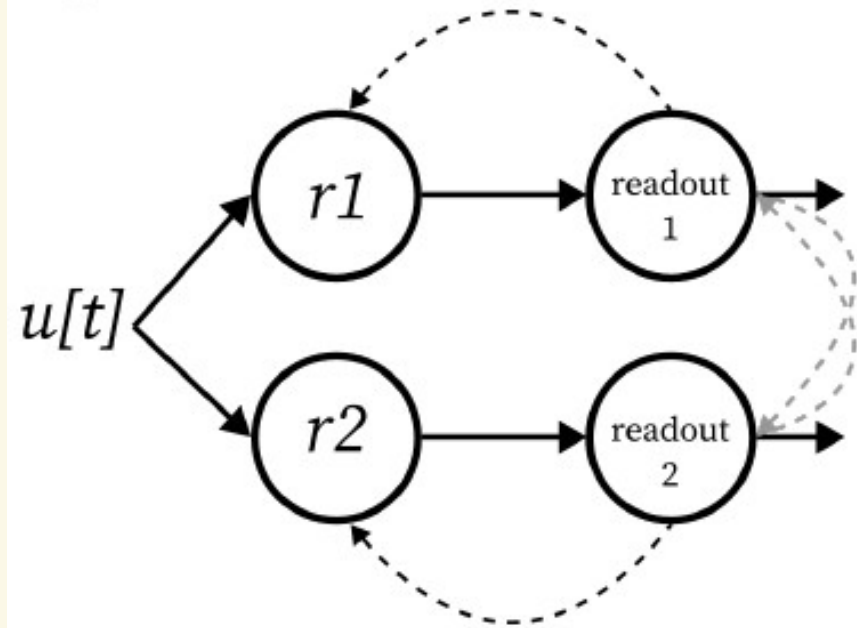
Architectures complexes

Créer des **modèles complexes** à partir de **briques simples**.

```
from reservoirpy.nodes import Input
in_node = Input(name="u")
r1 = Reservoir(units=100, name="r1")
r2 = Reservoir(units=220, name="r2")
readout1 = Ridge(ridge=1e-4)
readout2 = Ridge(ridge=1e-2)

r1 <<= readout1
r2 <<= readout2
readout1 <<= readout2
readout2 <<= readout1
```

```
model = in_node >> [r1 >> readout1, r2 >> readout2]
```

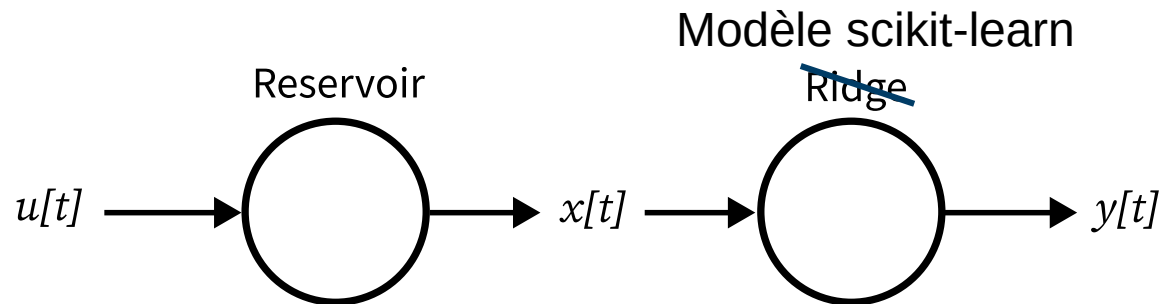


Les nœuds scikit-learn

```
from sklearn.linear_model import RidgeClassifier, LogisticRegression, Perceptron

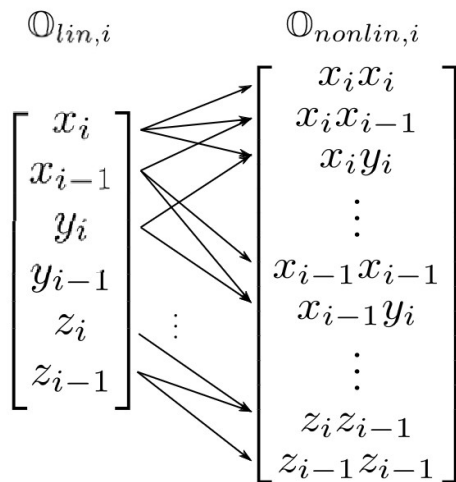
classifier      = ScikitLearnNode(RidgeClassifier)
logistic_regressor = ScikitLearnNode(LogisticRegression)
perceptron      = ScikitLearnNode(Perceptron)

model1 = reservoir >> classifier
model2 = reservoir >> logistic_regressor
model3 = reservoir >> perceptron
```

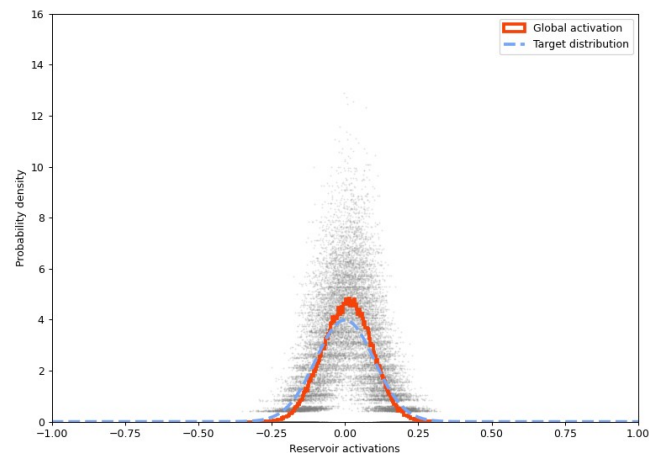


D'autres « réservoirs » plus exotiques...

`reservoirpy.nodes.NVAR`



`reservoirpy.nodes.IPReservoir`



Créer ses propres nœuds ReservoirPy

```
from reservoirpy import Node

def initialize(node: Node, x: np.ndarray, y: np.ndarray) -> None:
    ... # setting input and output dimension, initialize parameters

def forward(node: Node, x: np.ndarray) -> np.ndarray:
    ... # called at each timestep, return the new node state

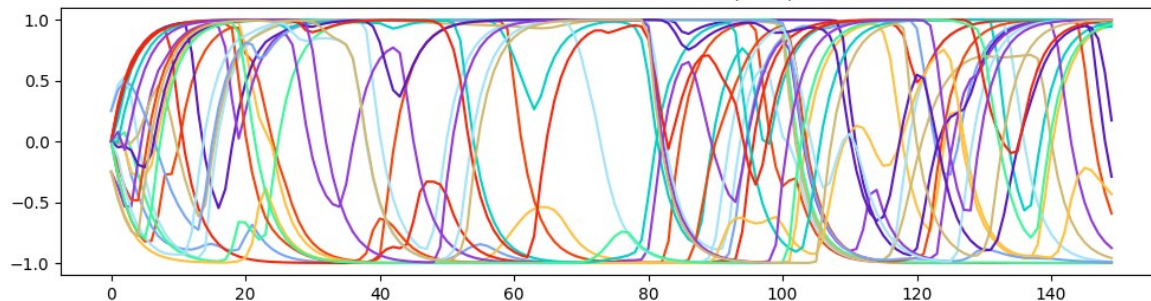
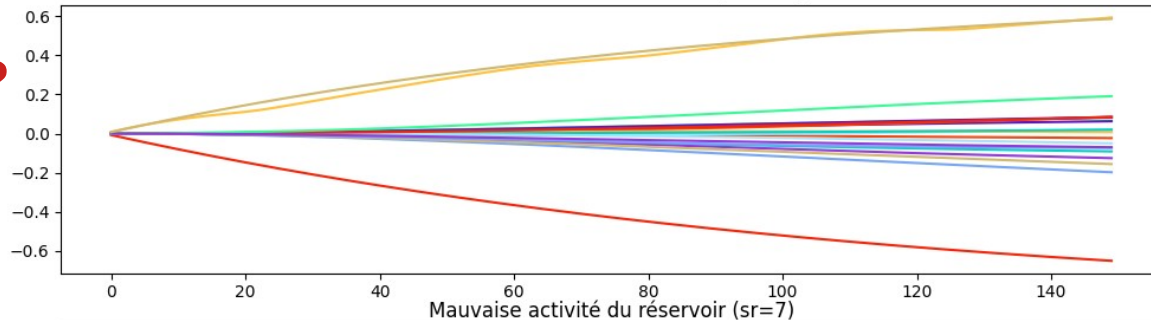
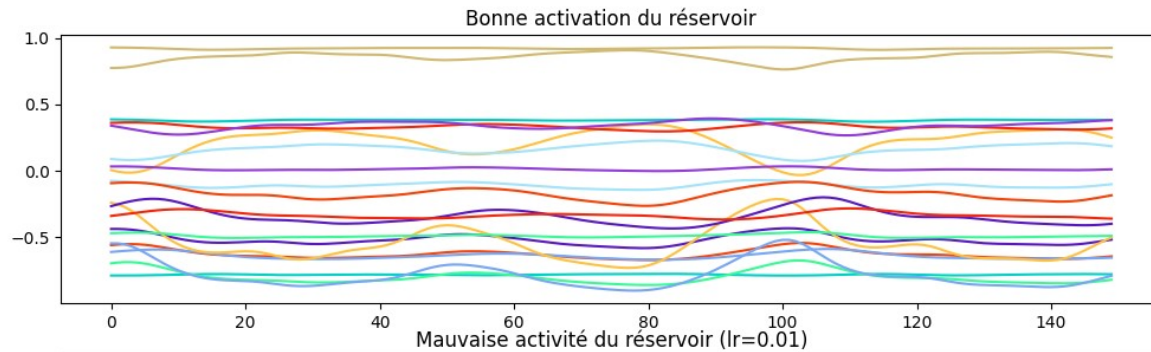
class MyNode(Node):
    def __init__(self, hyper1=1., name=None):
        super().__init__(
            forward = forward,
            initializer = initialize,
            hypers = {"hyper1": hyper1},
            params = {"param1": None},
            name = name,
        )
```

https://reservoirpy.readthedocs.io/en/latest/user_guide/create_new_node.html

De l'importance des hyper-paramètres

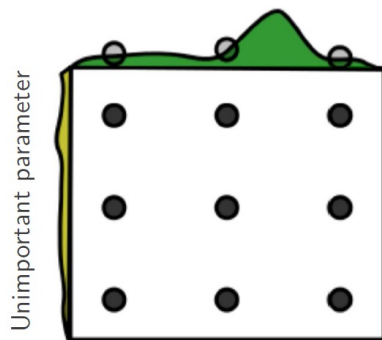
```
model = ESN(  
    units=100,  
    lr=0.3,  
    sr=1.25,  
    ridge=1e-8,  
)
```

???



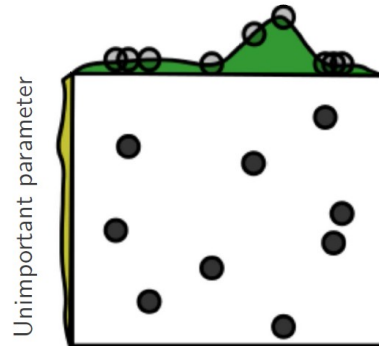
Recherche d'hyper-paramètres avec `reservoirpy.hipers`

Grid Layout



Important parameter

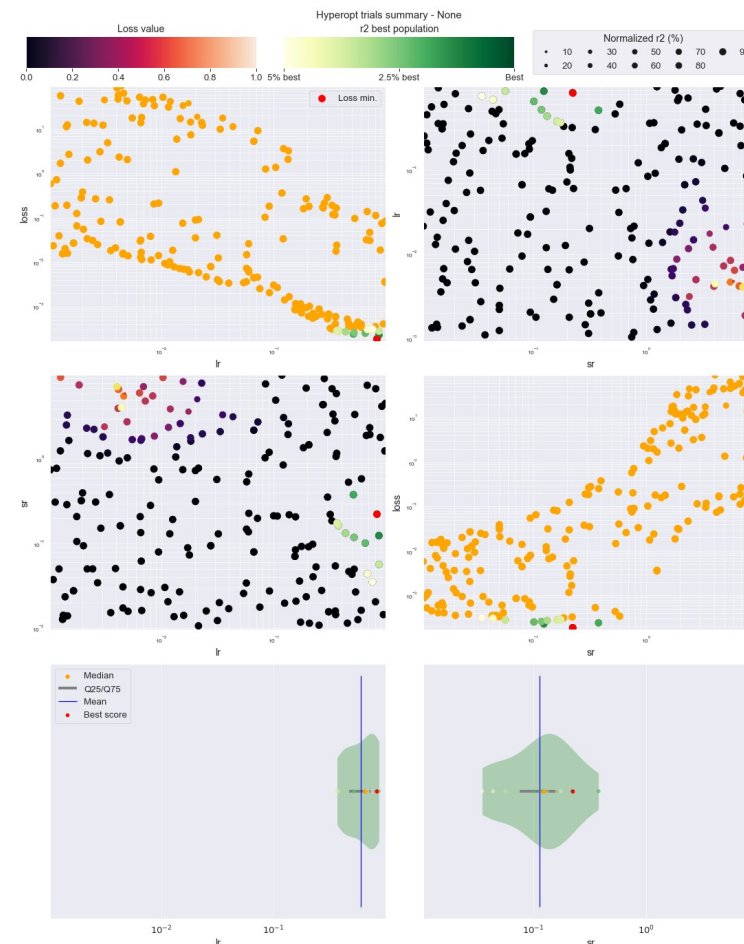
Random Layout



Important parameter

Voir le tutoriel n°4:

<https://github.com/reservoirpy/reservoirpy/tree/master/tutorials>



reservoirpy v0.3

Un *framework* Python dédié au Reservoir Computing:

- Apprentissage **en-ligne/hors-ligne**
- Boucles de **feedback**
- Modèles hiérarchiques, *deep reservoir computing*
- **Parallélisation** des calculs

Mais aussi:

- Des jeux de données, des métriques, des outils d'aide à l'optimisation des hyperparamètres avec *Hyperopt*.
- Des tutoriels, des exemples.

Une boîte à outil pour construire des modèles de Reservoir Computing:

- Reservoir,
- Reservoir autoregressifs (NVAR),
- Ridge regression,
- FORCE learning,
- Intrinsic Plasticity...

La philosophie de **ReservoirPy**



Libre et open-source (licence MIT)



Basé sur l'écosystème NumPy / SciPy



Entièrement documenté



De nombreux guides



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



Fork 88



Starred 330

Perspectives

- Intégration de nouvelles méthodes
- Calculs sur GPU (Jax)
- Réservoirs avec réseaux spiking

Ouverts aux suggestions et nouveaux cas d'applications !

Pour aller plus loin

`https://github.com/reservoirpy/reservoirpy/`

→ `tutorials/`

`https://reservoirpy.readthedocs.io/`

→ `User Guide`



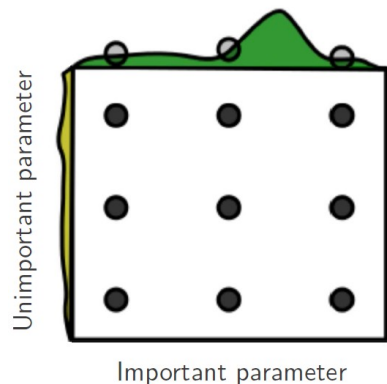
À vous !

Téléchargez le dépôt et le notebook

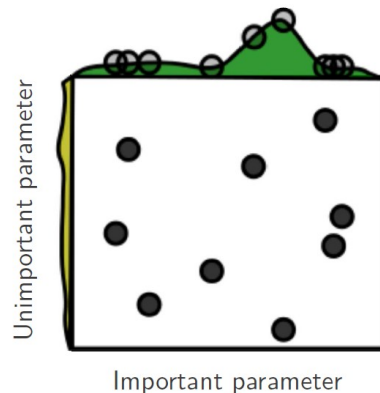
<https://github.com/PAUL-BERNARD/PFIA2024>

Optimisation des hyper-paramètres

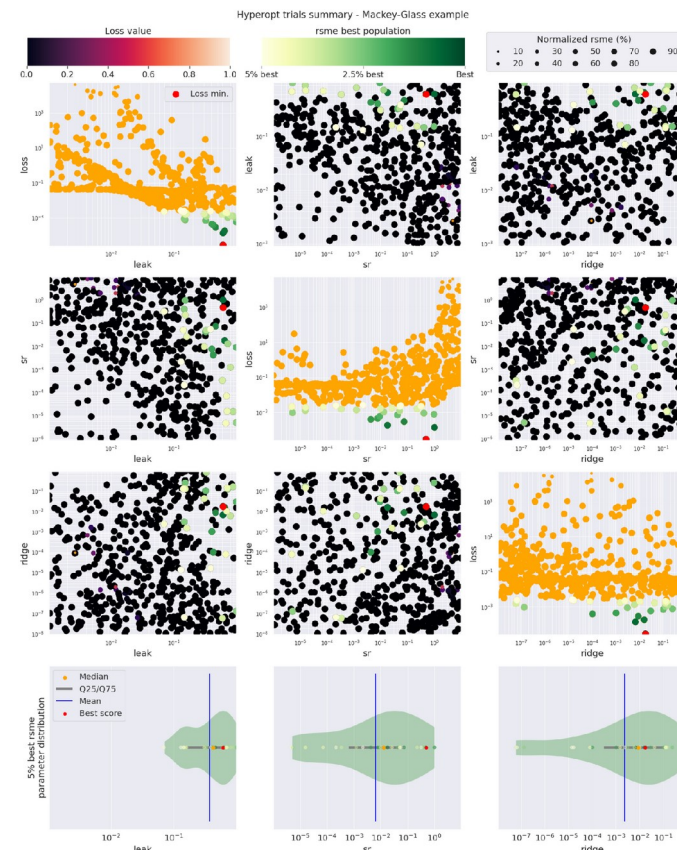
Grid Layout



Random Layout



https://github.com/reservoirpy/reservoirpy/blob/master/tutorials/4-Understand_and_optimize_hyperparameters.ipynb



Publications avec ReservoirPy

HAL publications related to this software ?

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. <i>ICANN 2020 - 29th International Conference on Artificial Neural Networks</i> , Sep 2020, Bratislava, Slovakia. (hal-02595026v2)
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. (hal-03533731)
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. (hal-03203318v2)
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. (hal-03482372)
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, 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. <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. (hal-03761440)
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. (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. <i>EuroSciPy 2022 - 14th European Conference on Python in Science</i> , Aug 2022, Basel, Switzerland. (hal-03780006)
hal-03945994	Nathan Trouvain, Xavier Hinaut. Reservoir Computing : traitement efficace de séries temporelles avec ReservoirPy. <i>Dataquityne</i> 2022, Feb 2022, Bordeaux, France. (hal-03945994)