



**HAL**  
open science

# ReservoirPy: Efficient Training of Recurrent Neural Networks for Timeseries Processing

Xavier Hinaut, Nathan Trouvain

► **To cite this version:**

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 Id: hal-03780006**

**<https://inria.hal.science/hal-03780006>**

Submitted on 18 Sep 2022

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

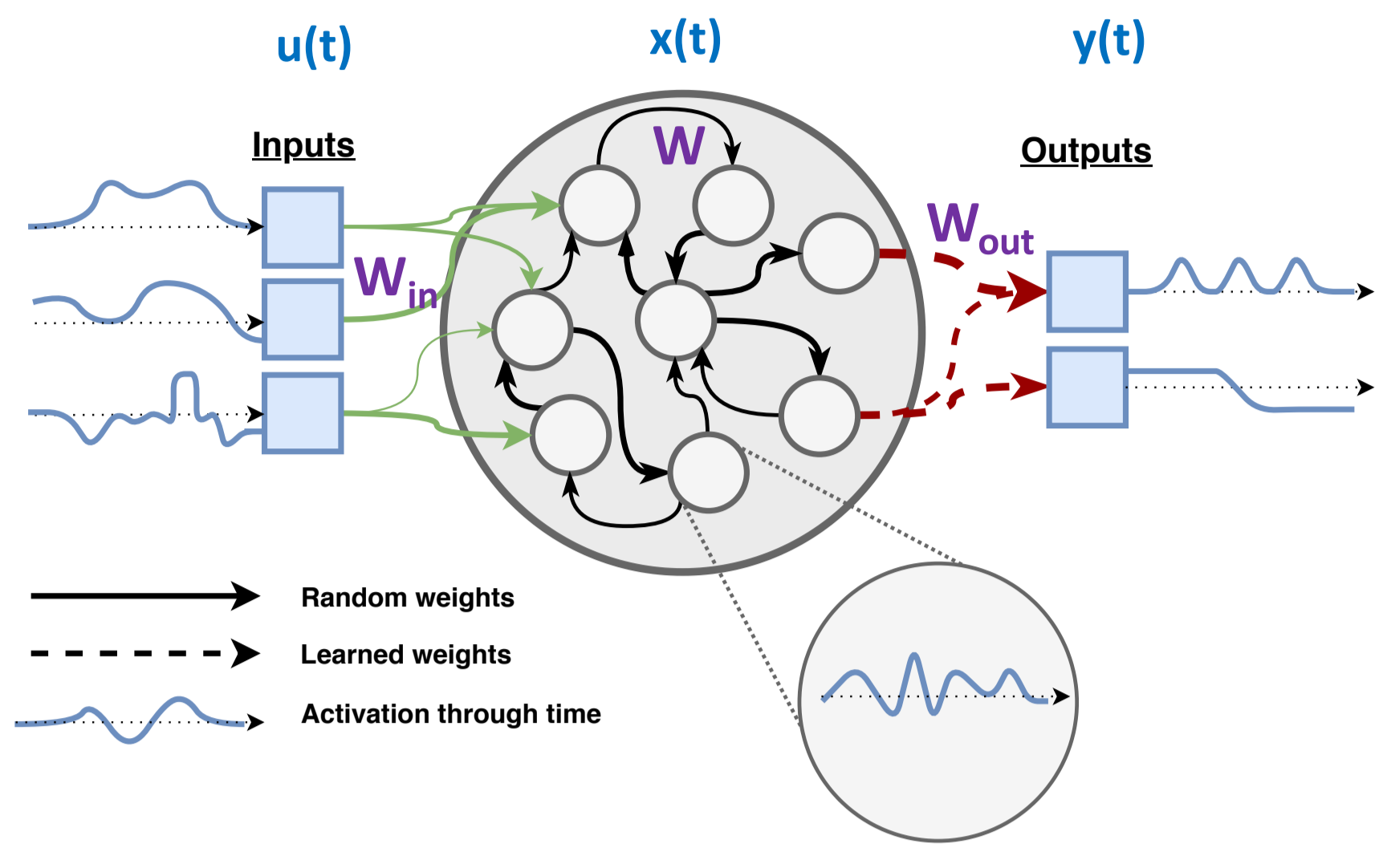


github.com/reservoirpy

### Abstract

ReservoirPy is a simple user-friendly library based on Python scientific modules. It provides a flexible interface to implement efficient Reservoir Computing (RC) [2] architectures with a particular focus on Echo State Networks (ESN) [1]. Advanced features of ReservoirPy allow to improve computation time efficiency on a simple laptop compared to basic Python implementation. Some of its features are: offline and online training, parallel implementation, sparse matrix computation, fast spectral initialization, advanced learning rules (e.g. Intrinsic Plasticity) etc. It also makes possible to easily create complex architectures with multiple reservoirs (e.g. deep reservoirs), readouts, and complex feedback loops. Moreover, graphical tools are included to easily explore hyperparameters with the help of the hyperopt library. It includes several tutorials exploring exotic architectures and examples of scientific papers reproduction.

### Reservoir Computing paradigm



### Reservoir state update

$$x(t) = \left(1 - \frac{1}{\tau}\right)x(t-1) + \frac{1}{\tau}f(W^{in}u(t) + Wx(t-1))$$

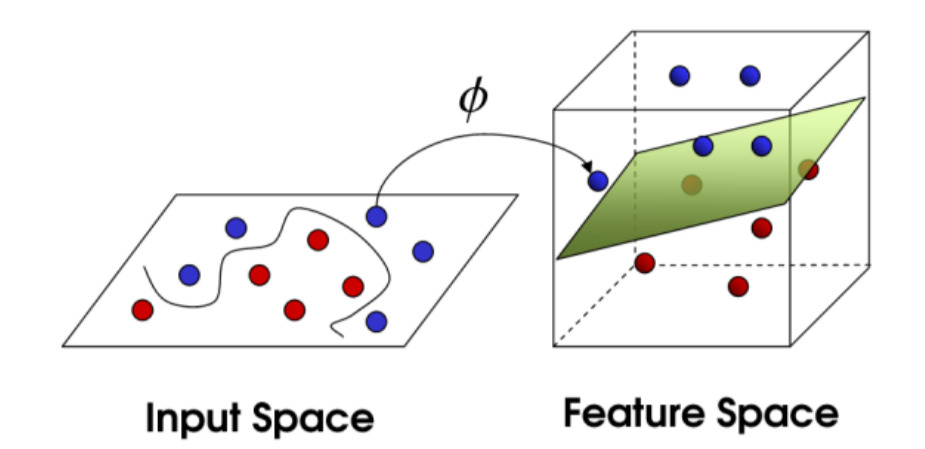
### Output (« read-out ») update

$$y(t) = W^{out}x(t)$$

- $u(t)$ : inputs
- $W^{in}$ ,  $W$ ,  $W^{out}$ : input, recurrent, output matrices
- $W^{in}$  and  $W$  matrices are kept random
- Only  $W^{out}$  is trained (e.g. ridge regression)
- $\tau$ : time constant of reservoir units
- leak rate (LR) is often used instead of  $\frac{1}{\tau}$
- $f$ : activation function (usually  $\tanh$ )

(see [1;2] for more details)

### Similar to temporal Support Vector Machines (SVMs)



### Intuition

The names "reservoir" for the recurrent layer, and "read-out" for the output layer, come from the fact that a lot of input combinations are made inside the recurrent layer (thanks to random projections): the "reservoir" is literally a reservoir of calculations (= "reservoir computing") that are non-linear. From this "reservoir" one linearly decodes (= "reads-out") the combinations that will be useful for the task to be solved. Reservoirs can be implemented on various kinds of physical substrates [9] (e.g. electronic, photonic, mechanical RC).



Founded at:  
*Inria*

License: MIT  
 pypi: <https://pypi.org/project/reservoirpy/>  
 GitHub: <https://github.com/reservoirpy/reservoirpy>  
 Documentation: <https://reservoirpy.readthedocs.io>  
 Related projects: <https://github.com/reservoirpy>  
 R interface to ReservoirPy: <https://github.com/reservoirpy/reservoirR>

### Why reservoirpy ?

- Few dedicated, modern, reusable tools.
- Last available dedicated tool: Oger (2012, Python 2).
- Other open source tools: PyRCN, EchoTorch, standalone scripts...
- Possibility of using existing machine learning frameworks:

**Machine Learning**  
 Scikit-learn (2011, Python 3) and extensions like Sktime (2019, Python 3)  
 → High quality codebase  
 → Lack of flexibility for RC purpose

**Deep Learning**  
 Tensorflow/Keras (2015, Python 3) PyTorch (2016, Python 3)  
 → Flexible  
 → Heavy codebase and unadapted

### Project philosophy

Python 3.8+ "scikit" tools: only based on Python standard scientific stack.  
 Everything is a recurrent network, everything is a timeseries.  
 Create complex models with simple building blocks.  
 Reach high computational efficiency.  
 Community driven development (tests, documentation, tutorials...).

### Targeted users

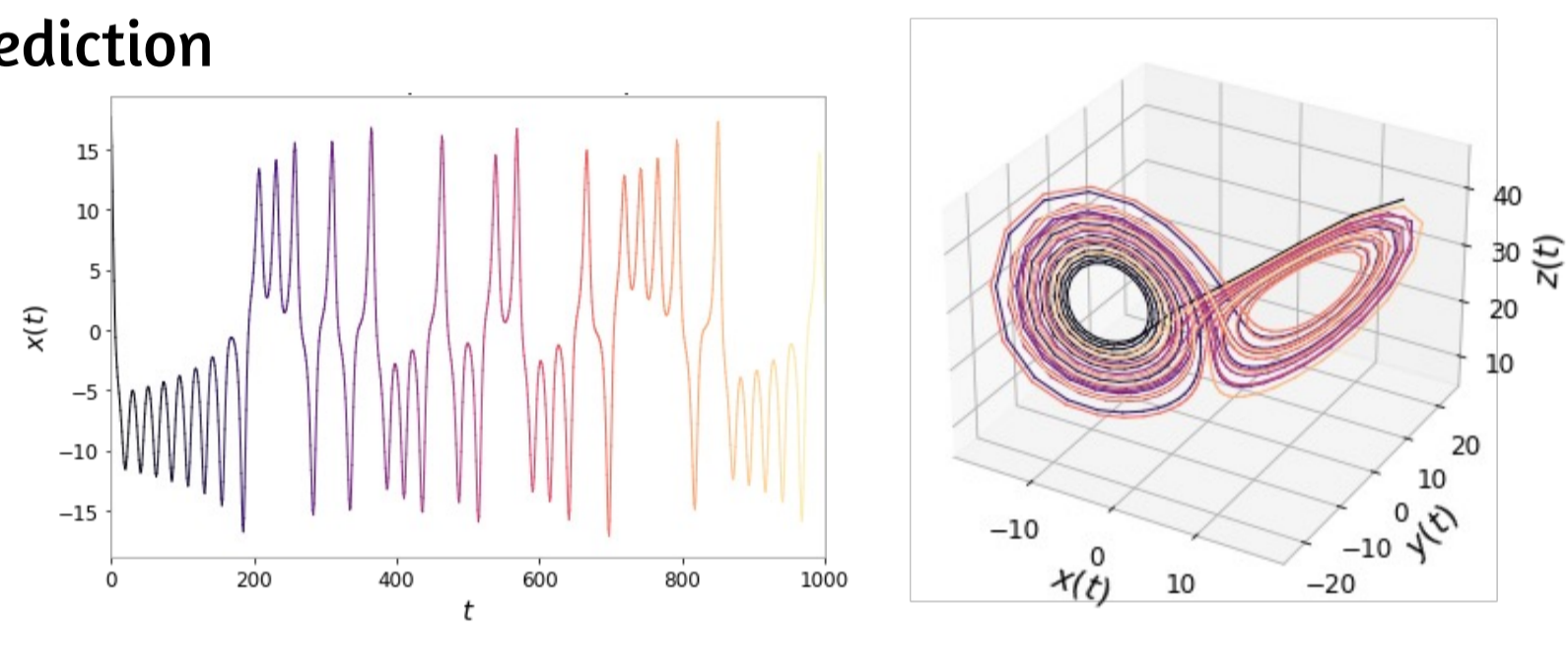
- Academic researchers
- Industry
- Machine learning students/beginners

### A simple application: Chaotic timeseries prediction

#### Importing the dataset (Lorenz attractor)

```
1 from reservoirpy.datasets import lorenz
2 X = lorenz(2500)
```

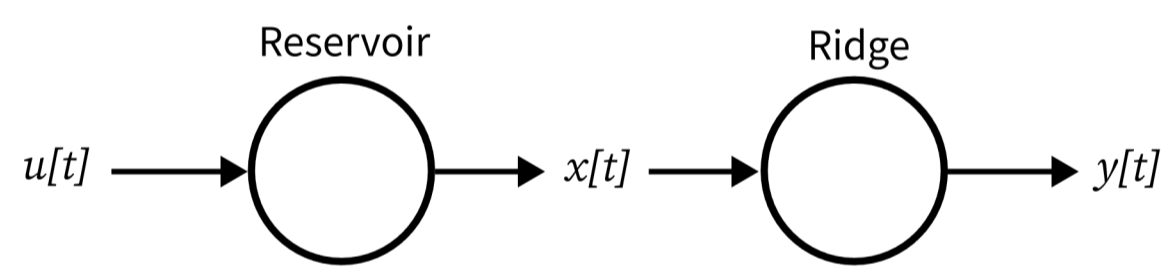
Given  $x[t]$ ,  $y[t]$ ,  $z[t]$ , can we predict  $x[t+10]$ ,  $y[t+10]$ ,  $z[t+10]$  ?



#### Node & model creation

A node is an independent operator, that applies a function on some data, potentially in a recurrent way. A model connects nodes together to compose operators.  
 ESN = Echo State Network (a specific instance of Reservoir Computing with firing rate neurons)

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



#### Model offline training & Evaluation

Offline training using linear regression: the model is trained in one shot on all available data.

```
1 esn.fit(X_train, Y_train)
```

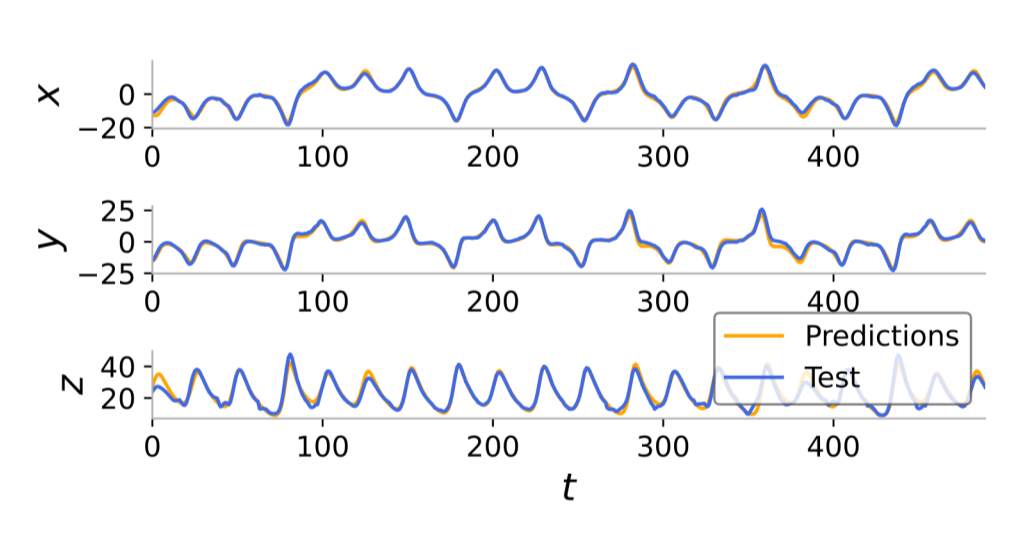
#### Model online training

Online using Recursive Least Squares (RLS) or Least Mean Squares (LMS).

```
1 esn.train(X_train, Y_train)
```

#### Evaluation

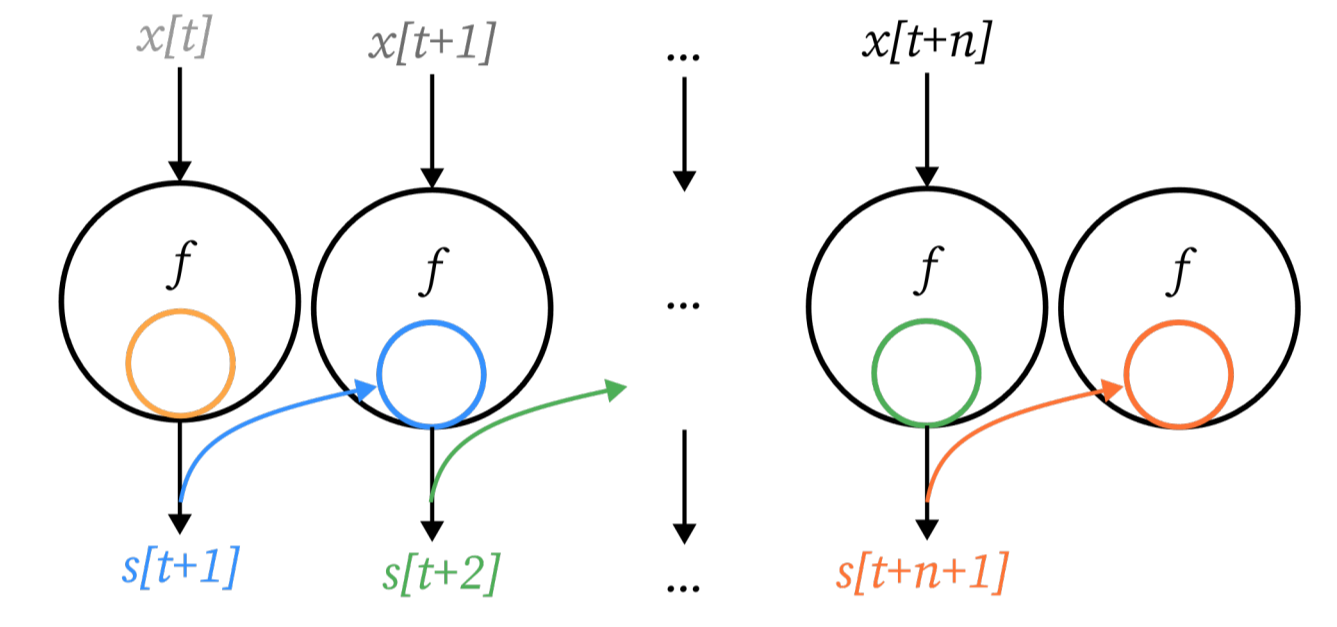
```
1 predictions = esn.run(X_test)
```



### Building blocks

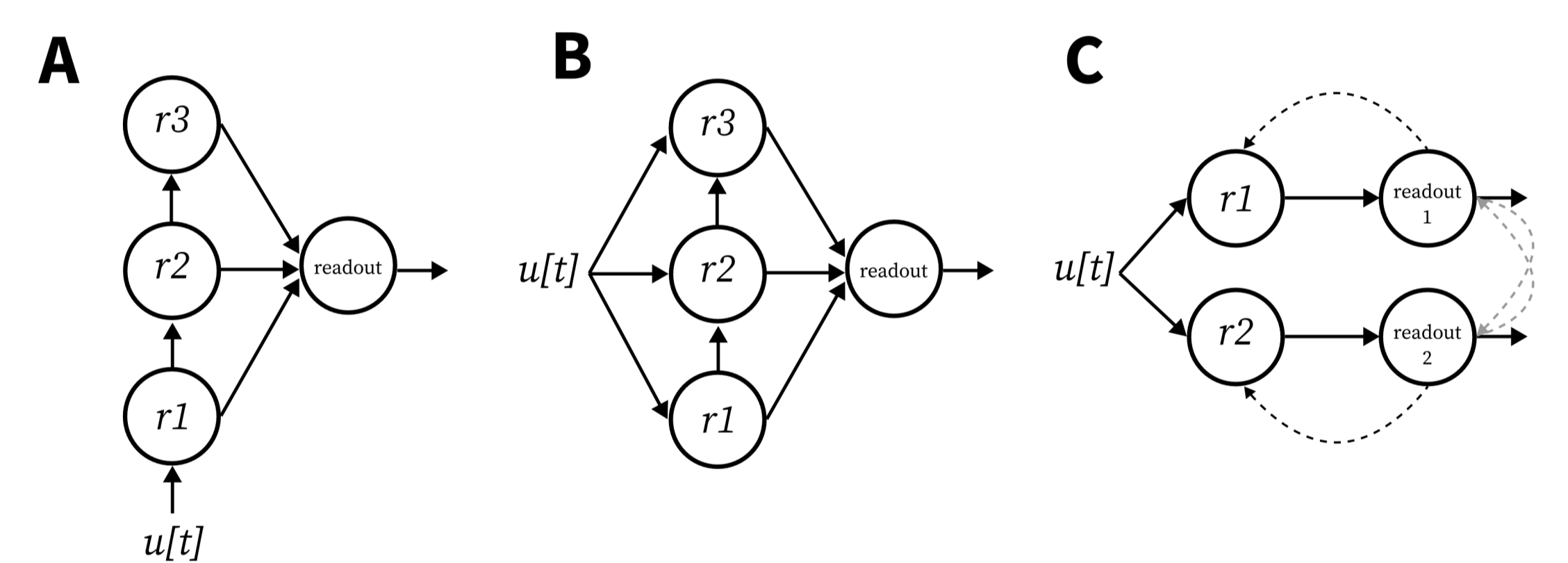
#### Node

Base object used to apply functions defined as  $f: s[t], x[t] \rightarrow s[t+1]$  to inputs  $x$  and modifying a state  $s$ . Can be parametrized and carry a learning rule to fit its parameters.



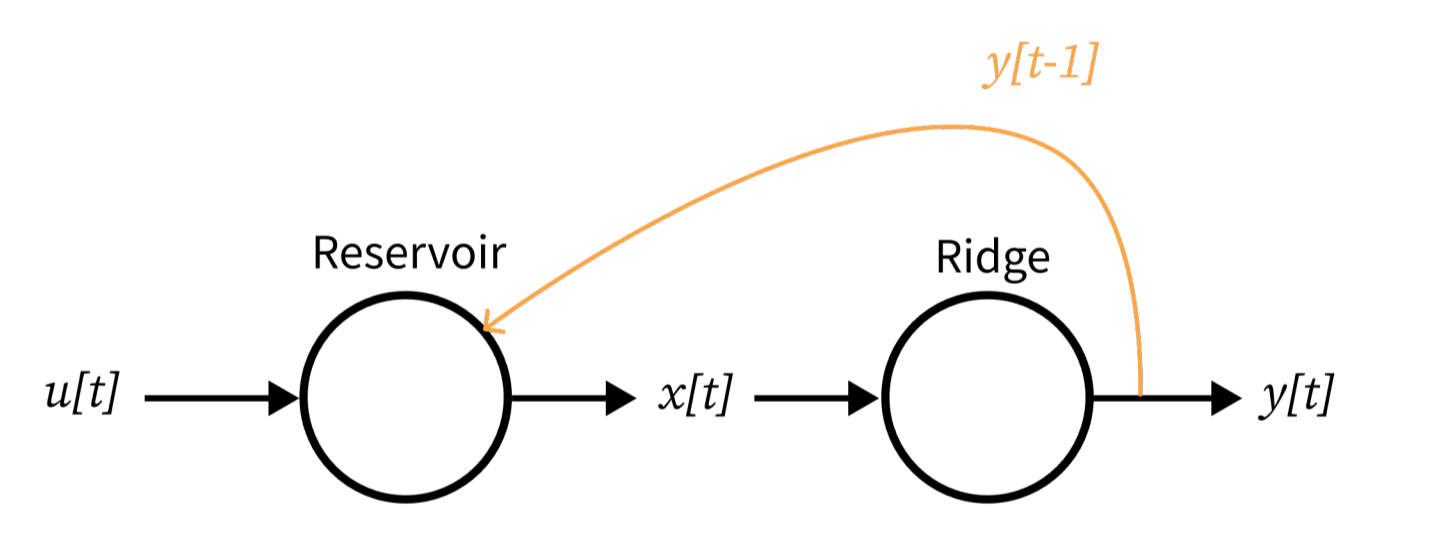
#### Model

Composition of Nodes



### Delayed connections (Feedback loops)

Can propagate different kind of information: → unit activities, error signals, yield ground truth values from an operator (teacher nodes), ... Can also be used just to add a delay in the network.



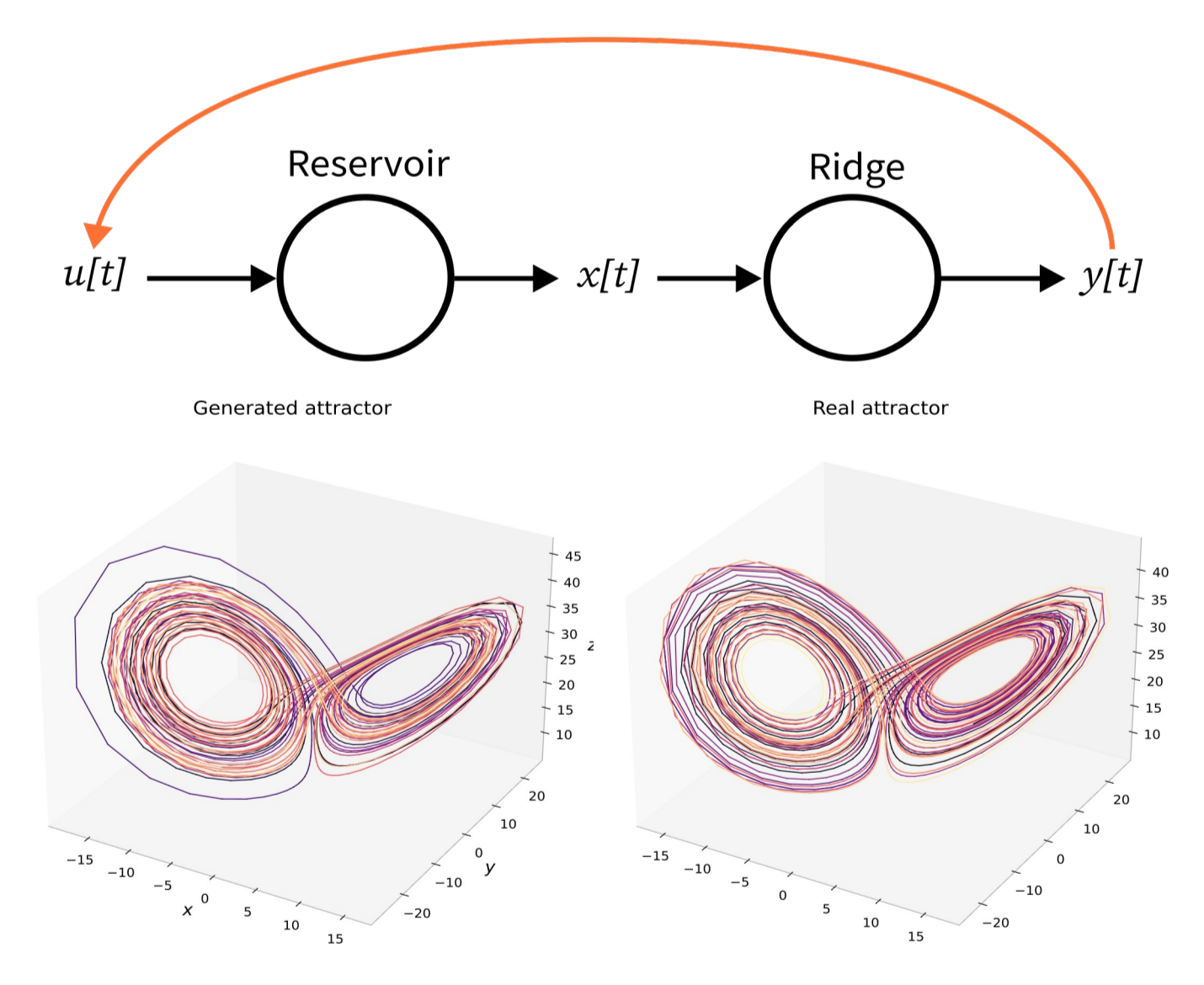
```
1 reservoir_with_fb = reservoir << readout
```

### Generative mode

#### Timeseries generation: Lorenz attractor

Generation of 2000 timesteps (consecutive to training timeseries).

```
1 N = 2000
2 y_gen = np.zeros((N, 3))
3 y = x_test[0]
4
5 for i in range(N):
6     y = model(y)
7     y_gen[i] = y
```

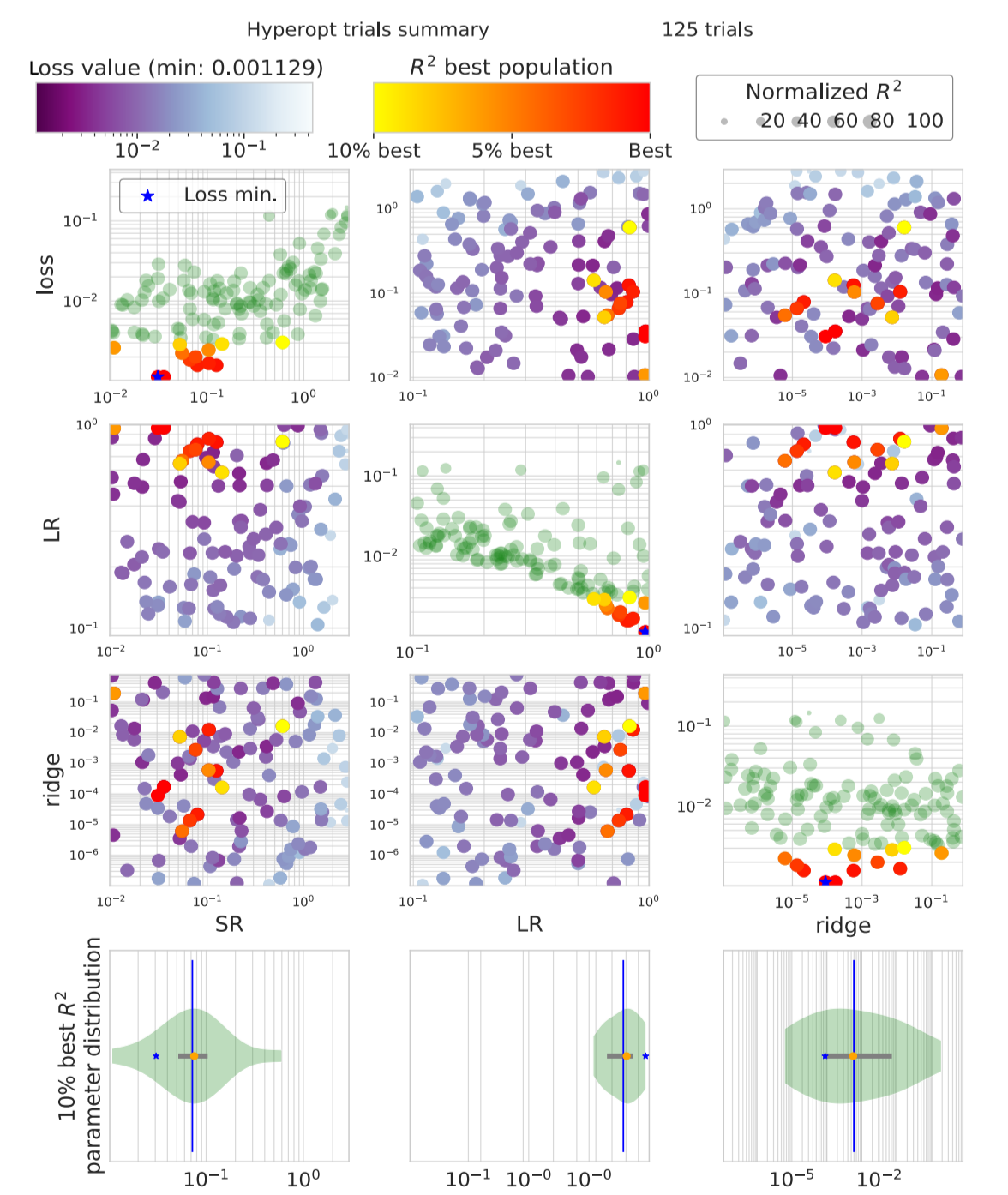


### Hyperparameter exploration

#### Random search on Lorenz attractor

Graphical tools to explore hyperparameters are included (using Hyperopt [6]). With 125 random sets of HPs you can already have a good idea of the landscape. SR: spectral radius, LR: leak rate, ridge: regularization param.

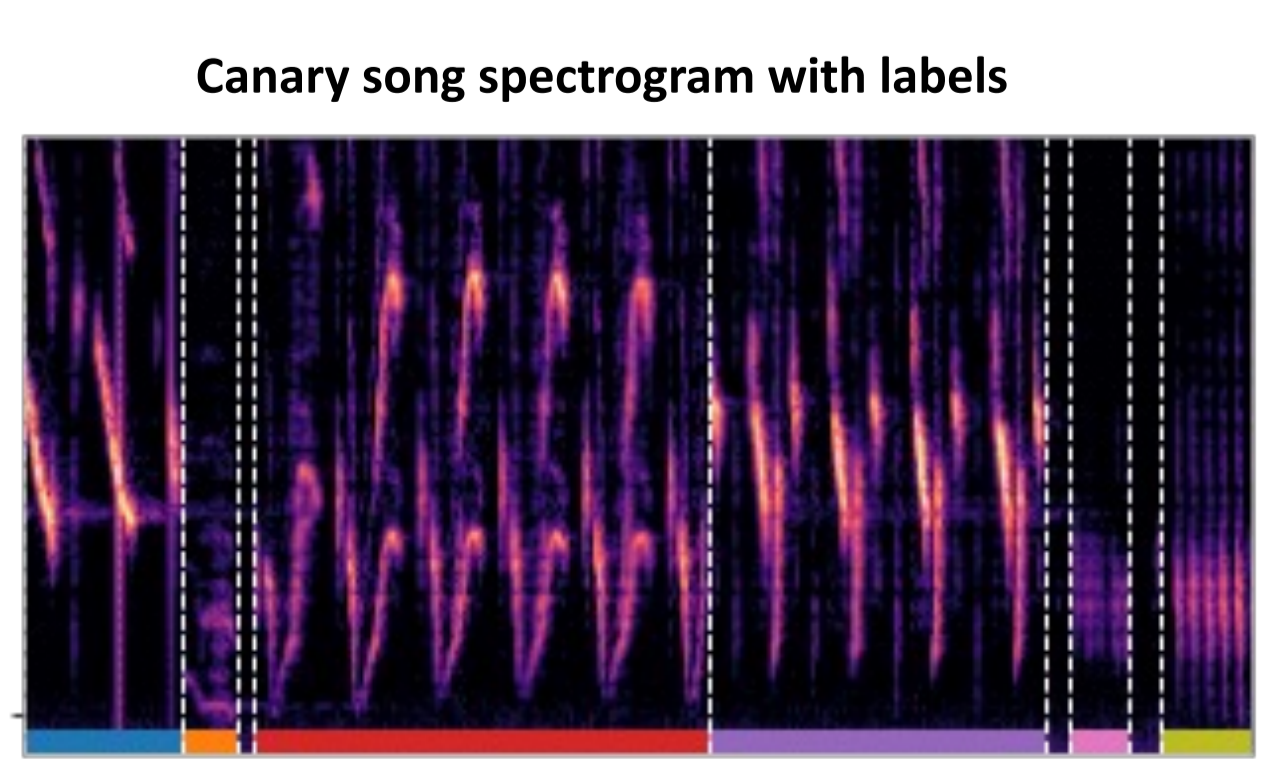
See [4] for more details on a general method to optimize HPs for reservoirs. See [7] to understand why random search is better than grid search.



### Sound classification example [5]

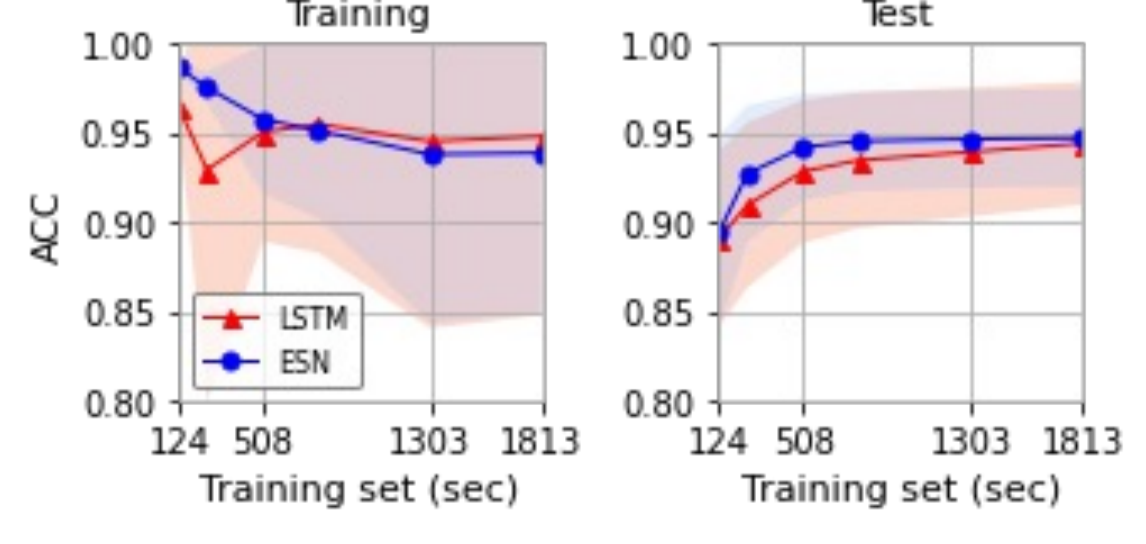
#### Dataset [9]

- 459 canary songs
- 40 labels
- MFCF preproc.



#### Train/test frame accuracies\*: LSTM vs. Reservoir

Reservoir learn with less data than LSTM (with equal number of trainable parameters) \*avg. 5 fold-CV;



#### CPU training time\* (sec.): LSTM vs. Reservoir

Reservoir benefited from distributed states computation of reservoirpy. \*Trained on 2h40 of songs on Intel i7-9850H with 12 cores at 2.60GHz and 32Gb RAM

Model	LSTM	ESN
Average training time (s)	2930 ± 222	35 ± 1

### Summary

#### Dedicated framework for Reservoir Computing

- Online/Offline learning
- Complex feedback loops
- Hierarchical models, deep reservoir computing
- Parallel computing
- Inspired from scikit-learn, Thinc and Keras

#### Perspectives

#### Upgrade framework capabilities

- Spiking neural networks (e.g. Liquid State Machines)
- Extended distributed computation strategies
- GPU hardware exploitation
- Scikit-learn tools integration

#### Reservoir Computing models toolbox

- Reservoir, Autoregressive reservoir (NVAR), Ridge regression, FORCE learning (RLS & LMS), Intrinsic Plasticity, ...

#### Complementary features

- Datasets, metrics, hyperoptimization tools using Hyperopt...
- Tutorials, examples, documentation...

#### Reservoir Computing Community

- Sharing and reuse models across community
- Replicate scientific results
- Implement new tools
- Find new use cases

### REFERENCES

- Jaeger & Haas, Science, 2004
- Lukosevicius, & Jaeger, Computer Science Review, 2009
- Lorenz, Deterministic Nonperiodic Flow. J. Atmo. Sci., 1963
- Hinaut & Trouvain, ICANN, 2021
- Trouvain & Hinaut, ICANN 2021
- Bergstra et al. ICML 2013
- Bergstra & Bengio, JMLR 2012
- Tanaka, Neural Networks, 2019
- Giraudon et al., Zenodo, 2021

### ACKNOWLEDGEMENTS

ReservoirPy development was supported by an Inria engineer "ADT" grant 2020–2022. We thank all contributors to ReservoirPy in particular Nicolas Rougier for his great ideas and guidance in project management. Many thanks to A. Strock, C. Mercier and H. Chateau-Laurent for their feedbacks on the poster.