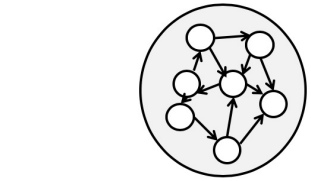
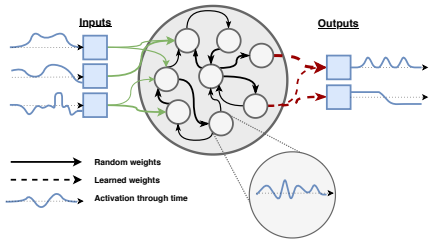
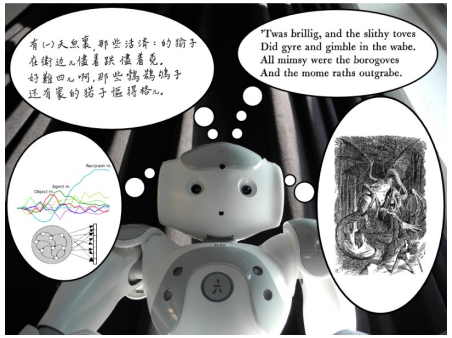
Vendredi 28 mars 2025 – CY Tech Pau – M2

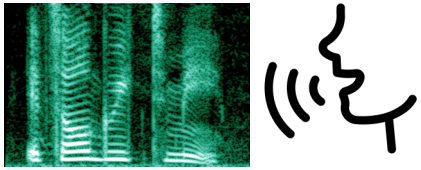
Le Reservoir Computing

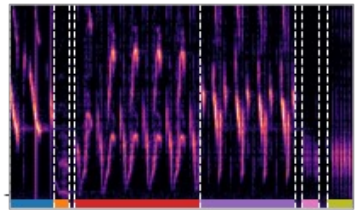
Prédire, générer, discriminer des sequences ...

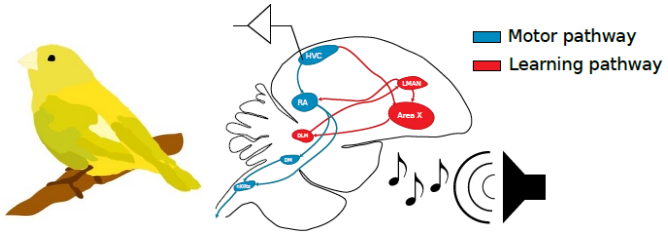










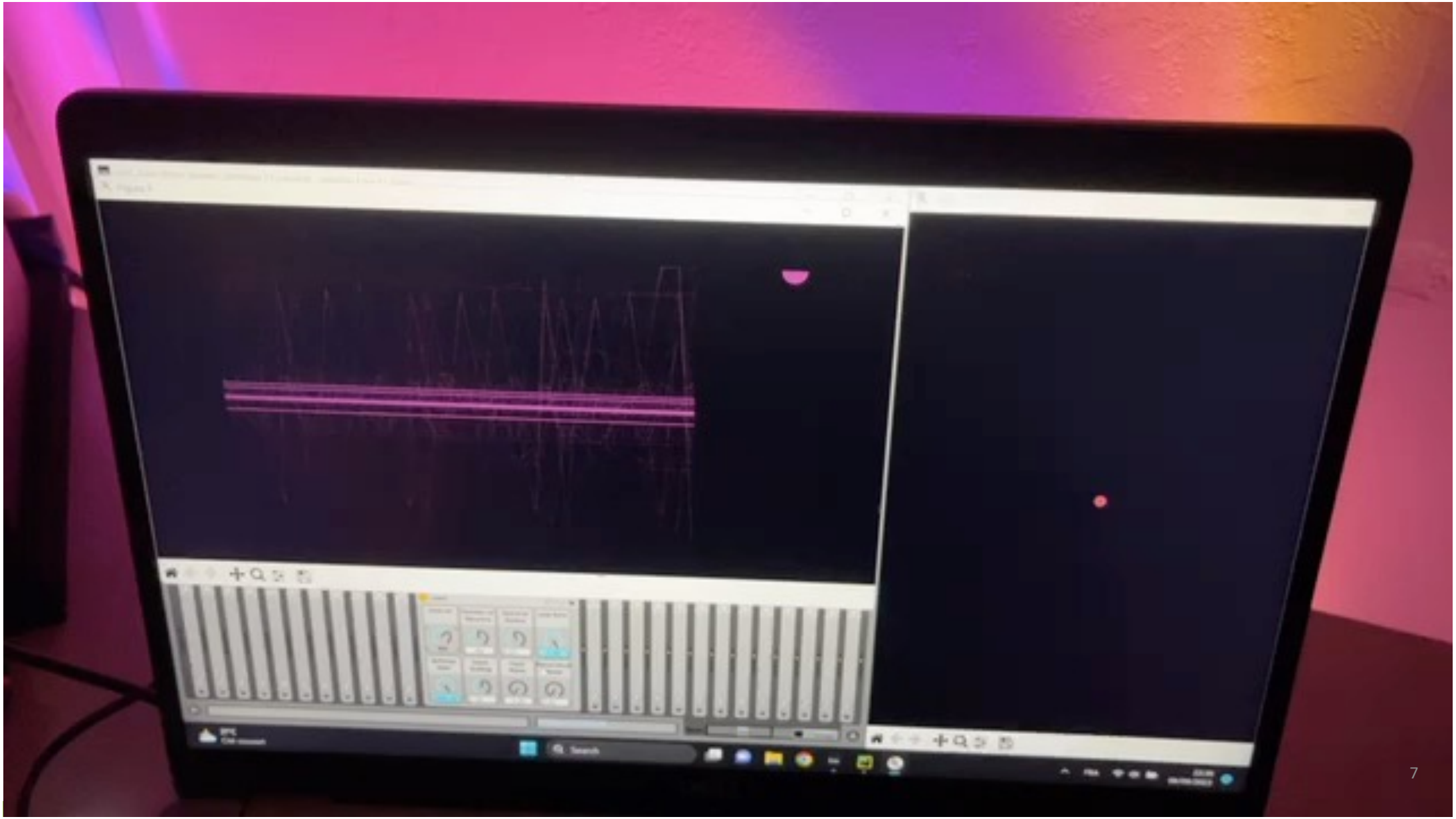


Xavier Hinaut

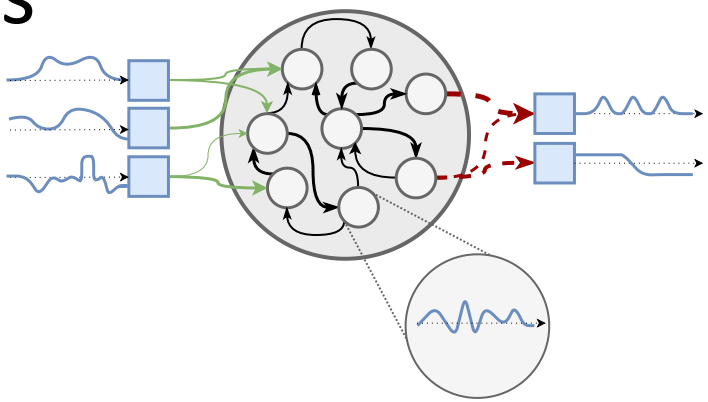
Inria, Bordeaux, France
Researcher @Mnemosyne team

xavier.hinaut@inria.fr
@neuronalX
@hinaut @reservoirpy

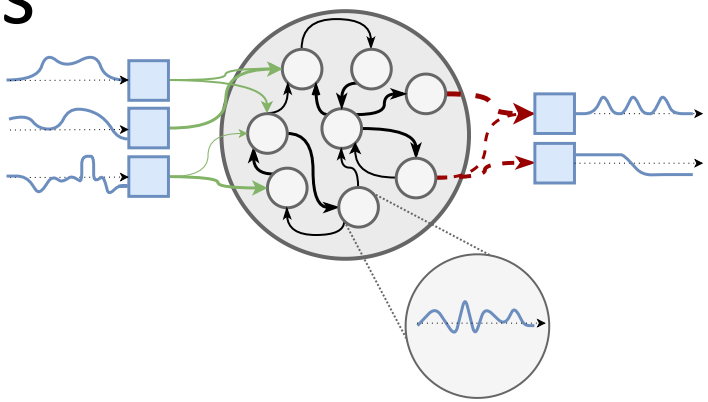
reservoirpy



Les reseaux aléatoires peuvent générer des dynamiques fascinantes

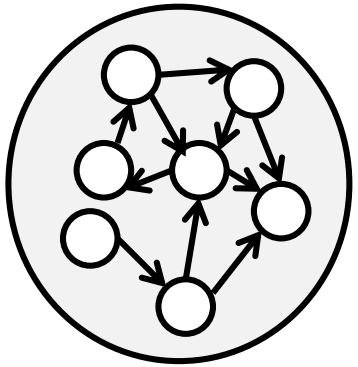


Les reseaux aléatoires peuvent
générer des dynamiques fascinantes



Le reservoir computing exploite
les dynamiques naturelles d'un réseau aléatoire

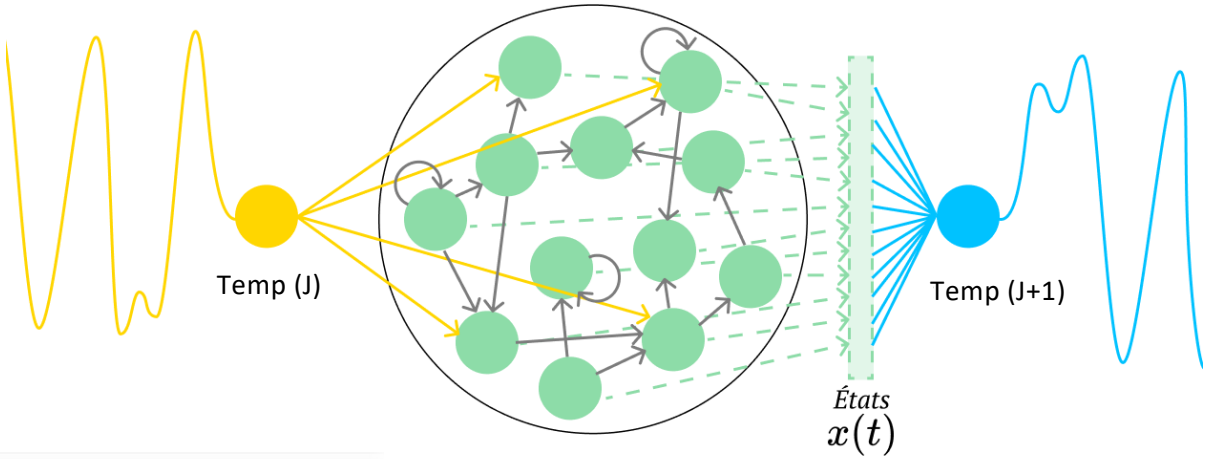
A quoi servent ces Réseaux de Neurones Récurrents ?

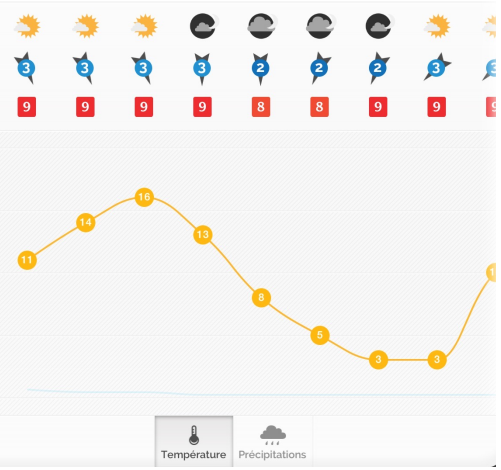


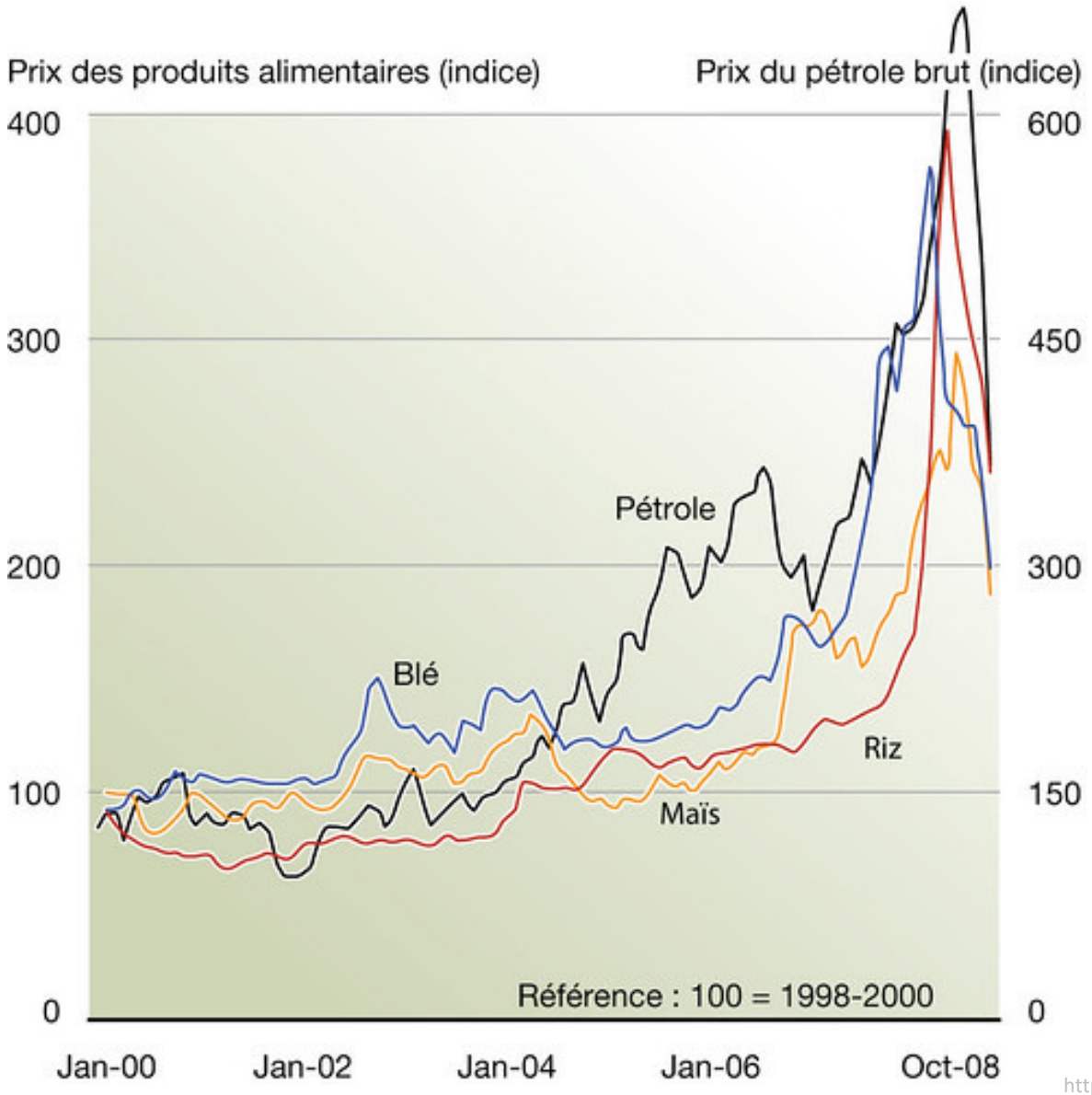
Neurone

Connexion entre deux neurones

Prédiction de séries temporelles

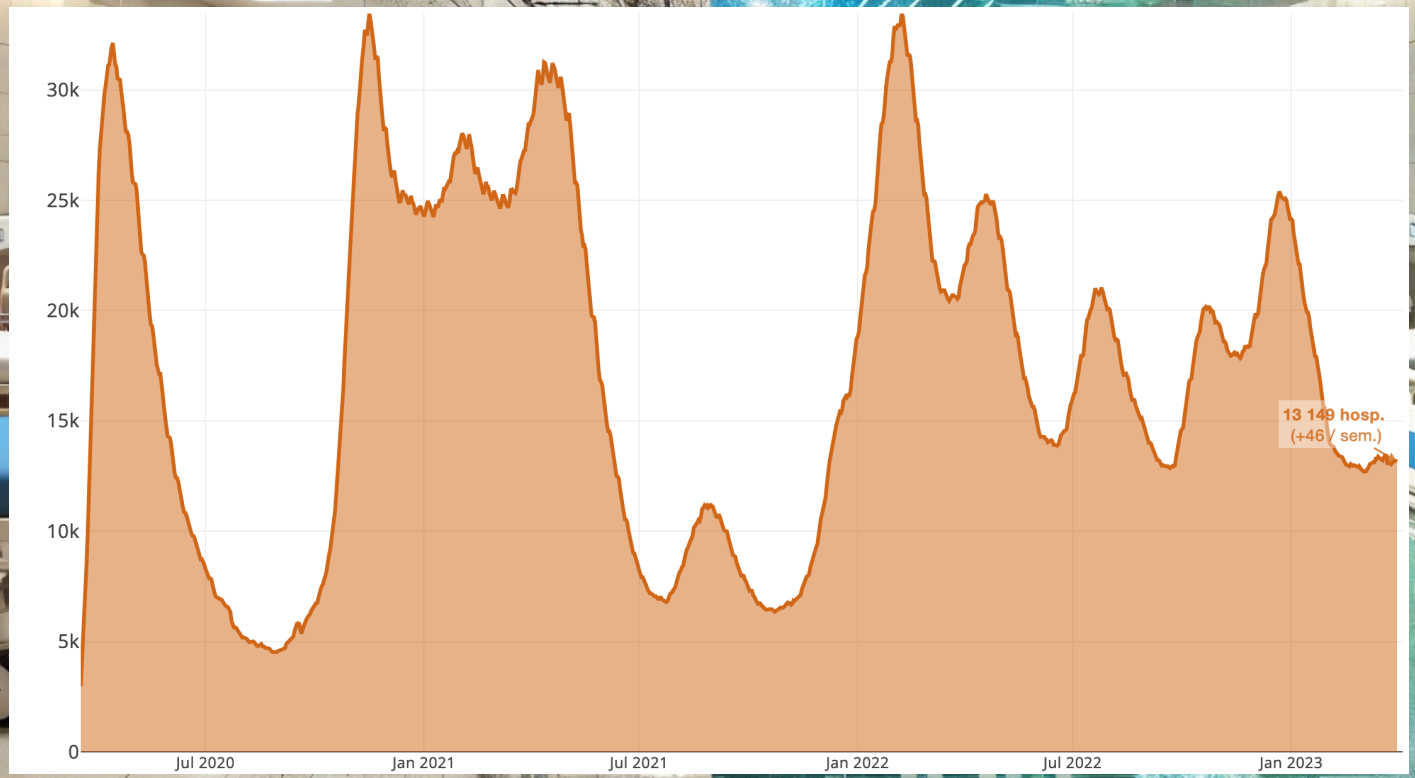






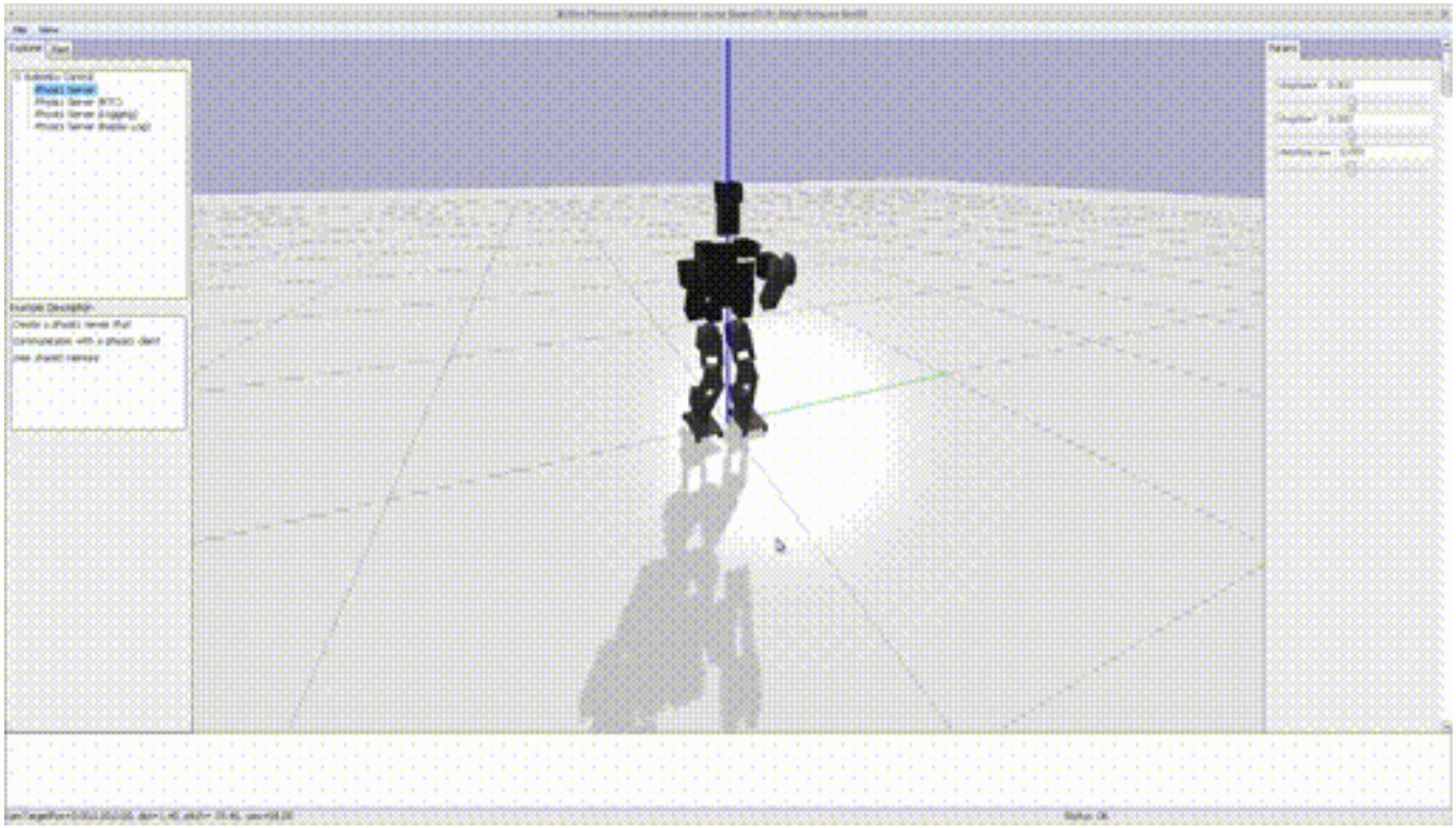
https://www.flickr.com/photos/gridarendal/32211647472 (by-nc-sa/2.0/)

Prédiction d'hospitalisations COVID



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

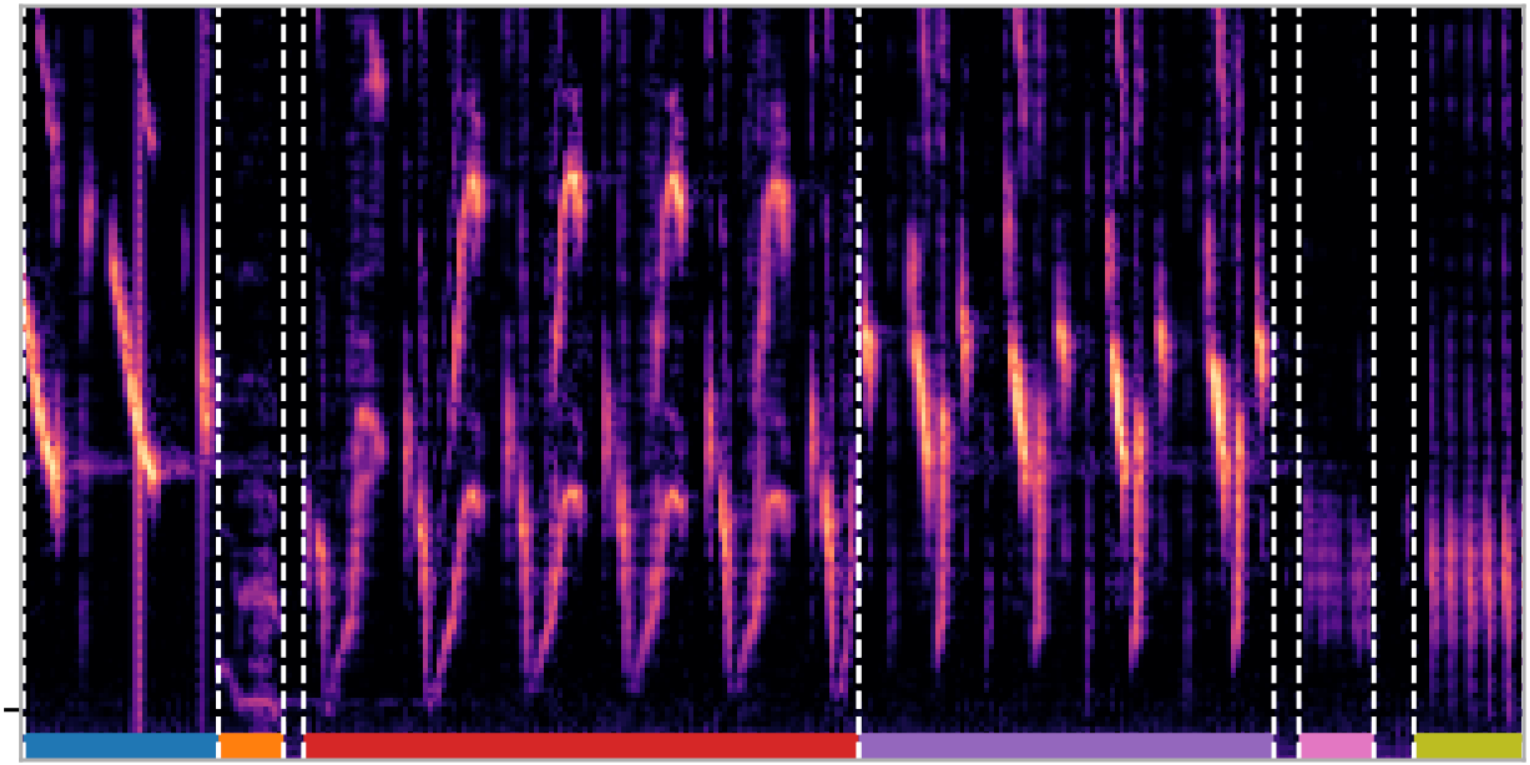
Prédiction d'événements (chute, panne, …)



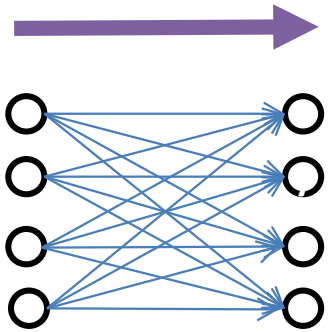
Données : Grégoire Passault, Rhoban, Labri, Univ. Bordeaux

Classifier des sons

(syllabes de canari)



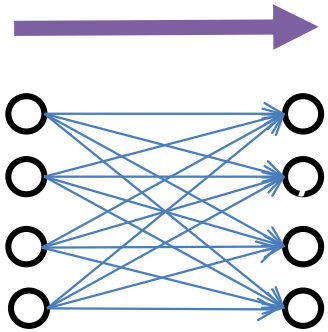
2 types de réseaux



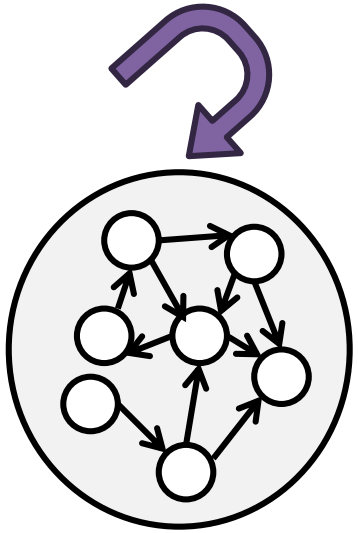
- Réseau de neurones en couches
  - Connexions d'une couche à la suivante
  - ex : Faire des associations (Image → « Chat »)

Neurone → Connexion entre neurones

2 types de réseaux



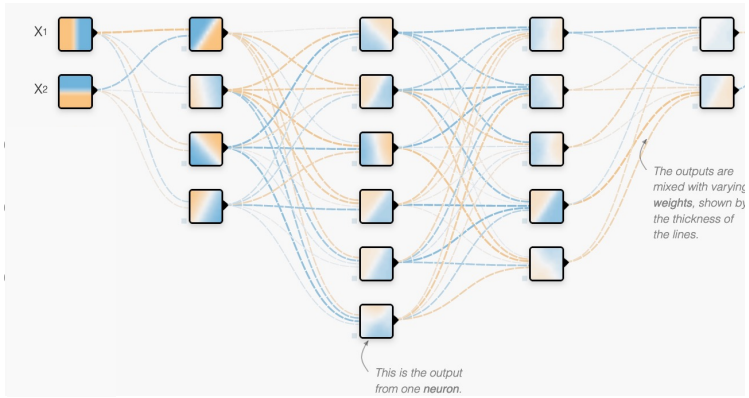
- Réseau de neurones en couches
  - Connexions d'une couche à la suivante
  - ex : Faire des associations (Image → « Chat »)



- Réseau de neurones récurrent
  - Connexions au sein d'une même couche
  - Mémoire dynamique du contexte temporel
  - ex : reconnaitre une séquence (mouvement)

giphy.com

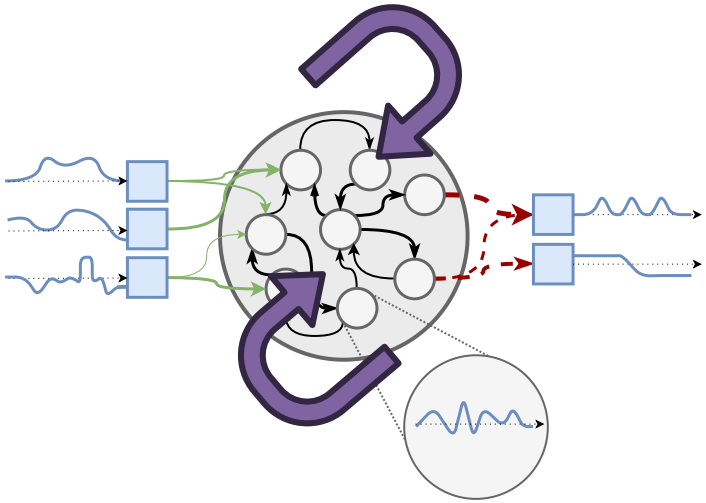
2 types de réseaux



Chat

Chien

Chaque stimulus est indépendant du précédent



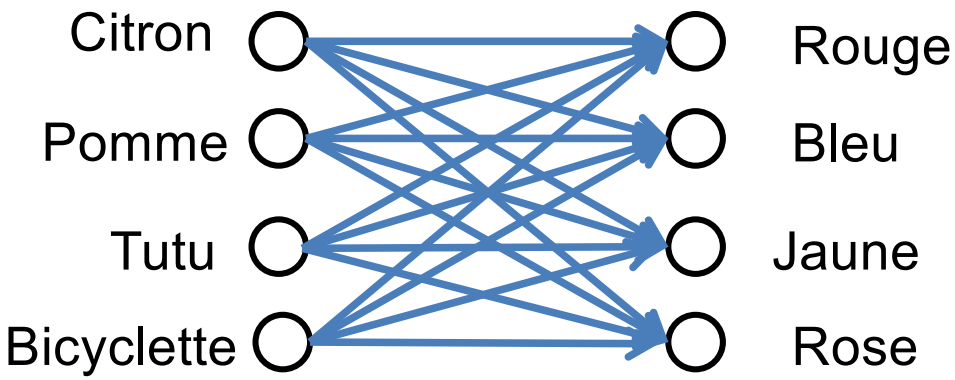
Geste « bonjour »

Geste « aimer »

Chaque stimulus est dépendant du précédent

giphy.com

Un réseau de neurones en couches



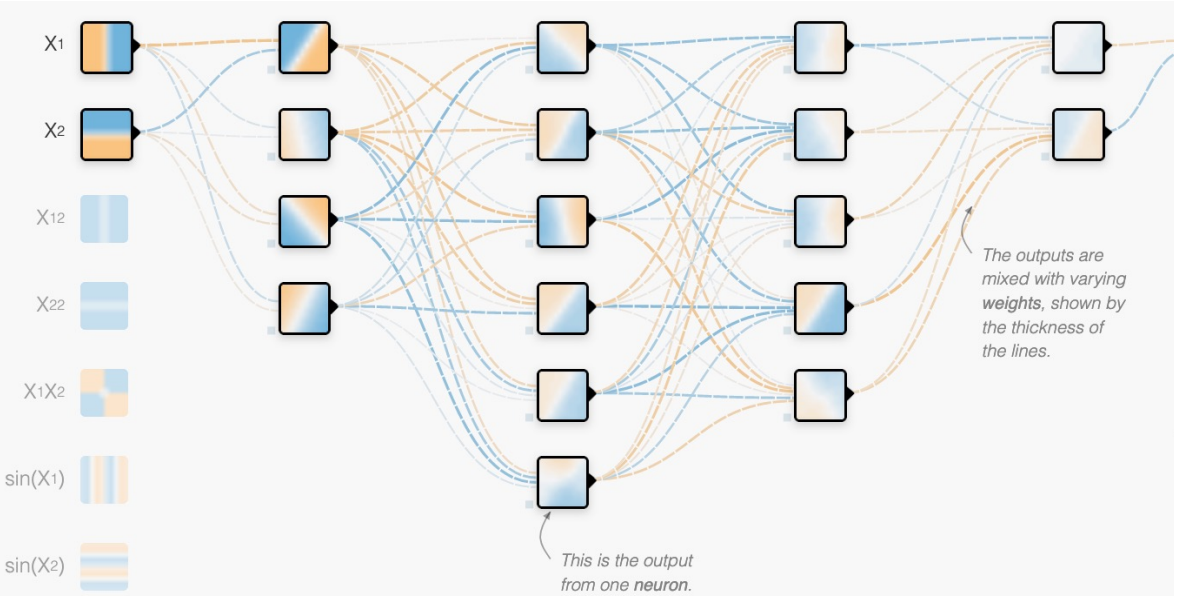
- Loi de Hebb (1949)
  - lorsque 2 neurones sont actifs en même temps leurs connexions sont renforcés

    « Neurons that fire together, wire together »

Un réseau de neurones en couches

L'algorithme de rétro-propagation du gradient permet d'apprendre avec plusieurs couches

→ Deep Learning

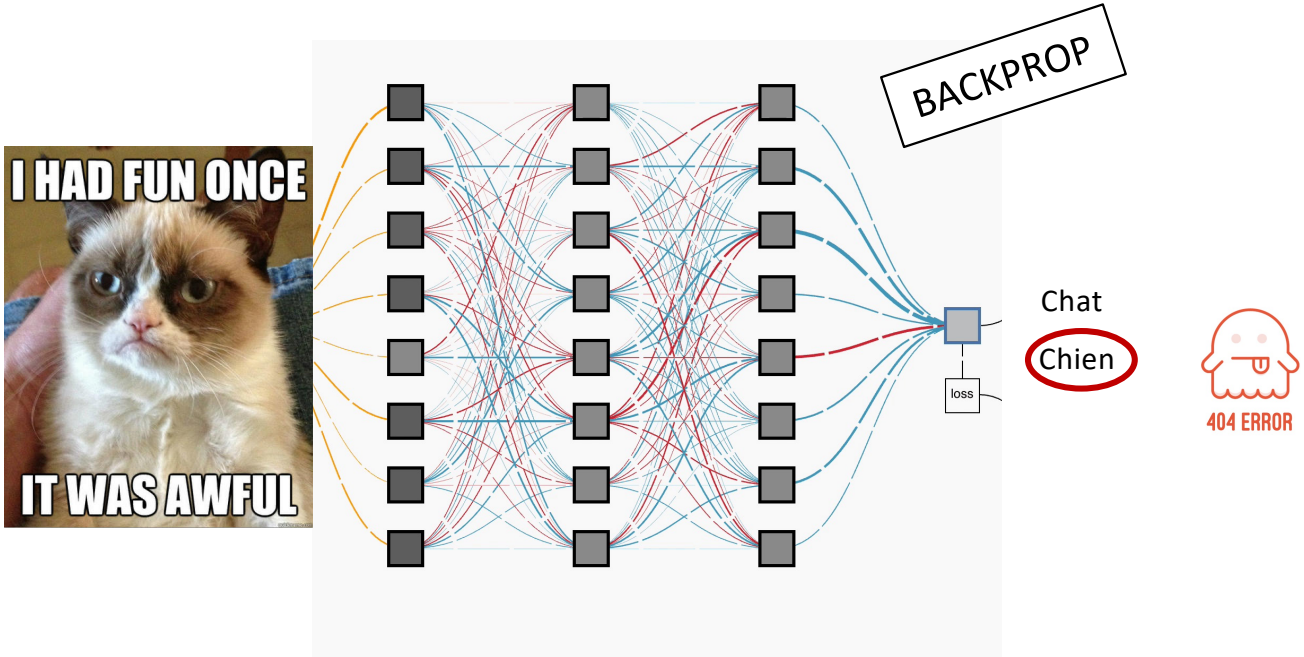


playground.tensorflow.org

Un réseau de neurones en couches

L'algorithme de rétro-propagation du gradient permet d'apprendre avec plusieurs couches

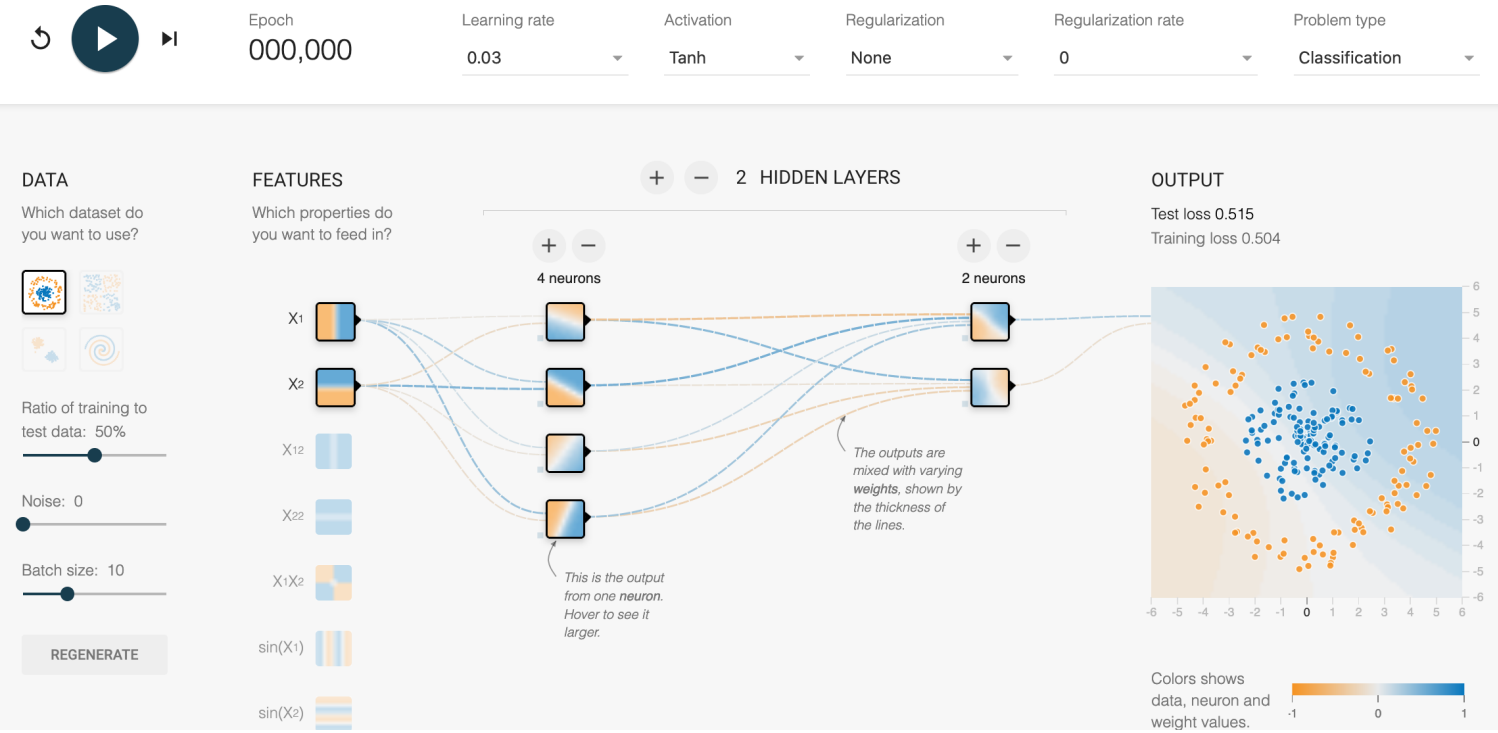
→ Deep Learning



github.com/xnought/backprop-explainer

giphy.com

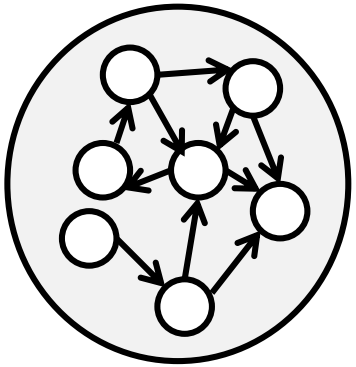
Testez votre intuition des réseaux de neurones

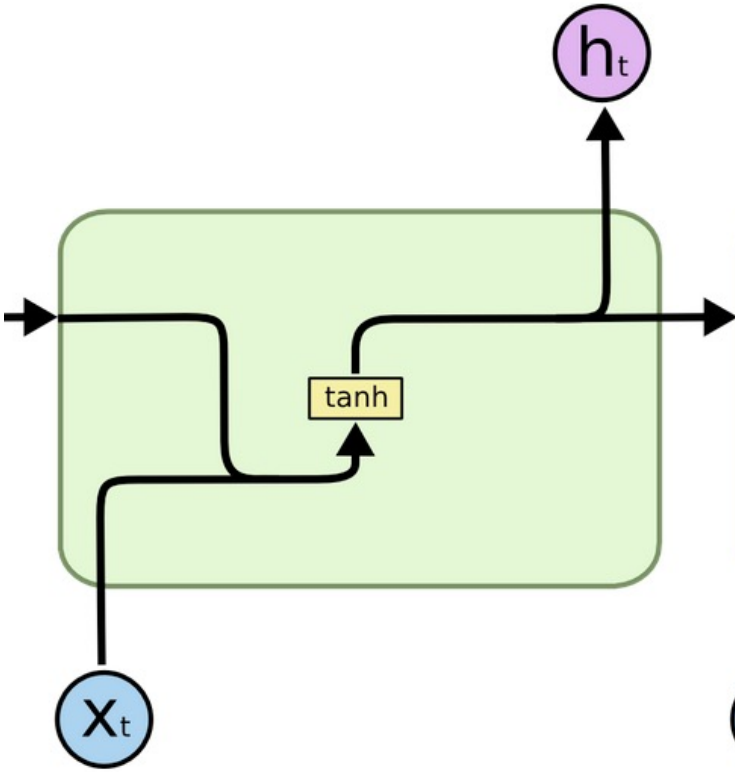


- http://playground.tensorflow.org
- http://binaire.blog.lemonde.fr/2017/10/20/jouez-avec-les-neurones-de-la-machine/

RNN

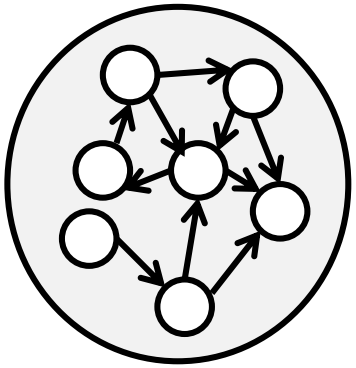
(Recurrent Neural Network)

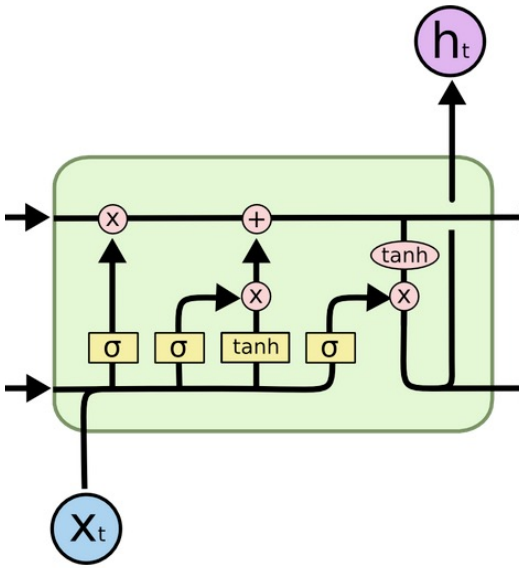




LSTM

(Long Short Term Memory network)





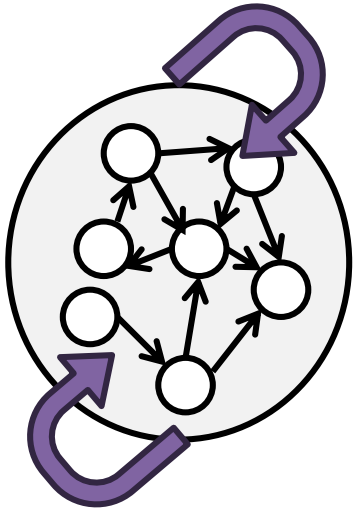
Entraîner un RNN/LSTM avec la backprop

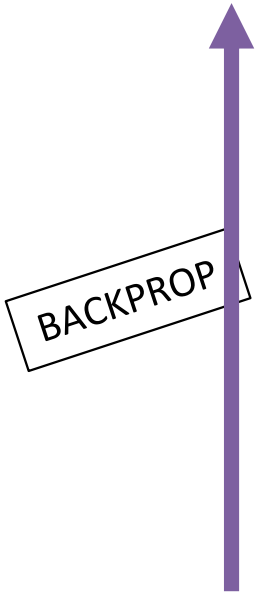
= virtualiser le temps

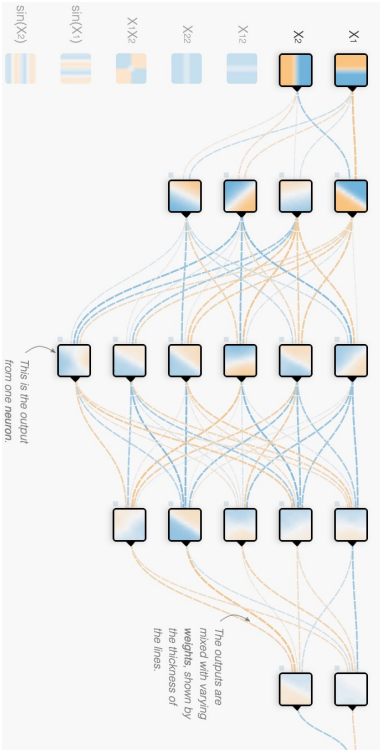
= déplier les récurrences

→ les mettre à plat comme un reseau en couches

t-N, …., t-2, t-1, t, ….







t-N

…

t-3

t-2

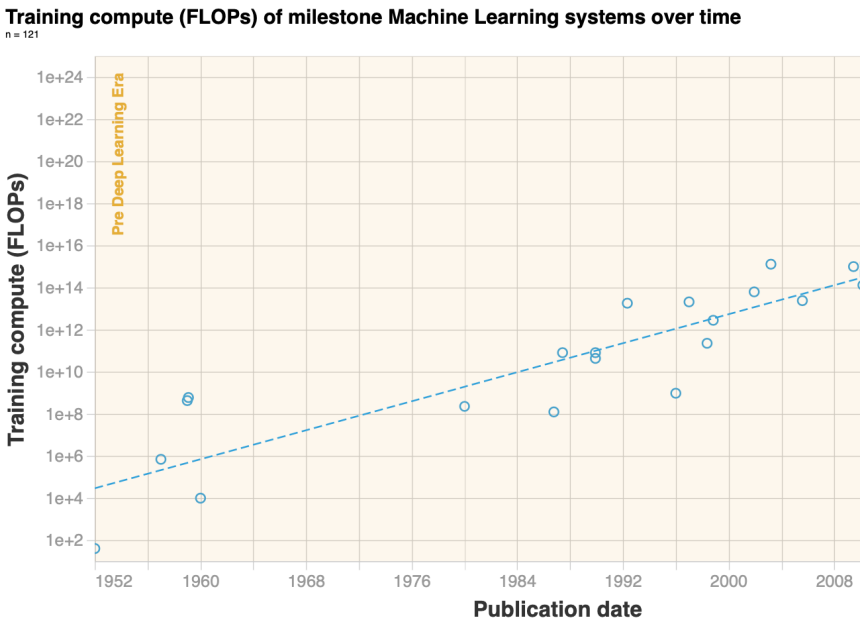
t-1

t

Nombre de couches = taille du contexte temporel → N

BACKPROP

reservoirpy
BACKPROP



Doublement de la puissance de calcul

Loi de Moore

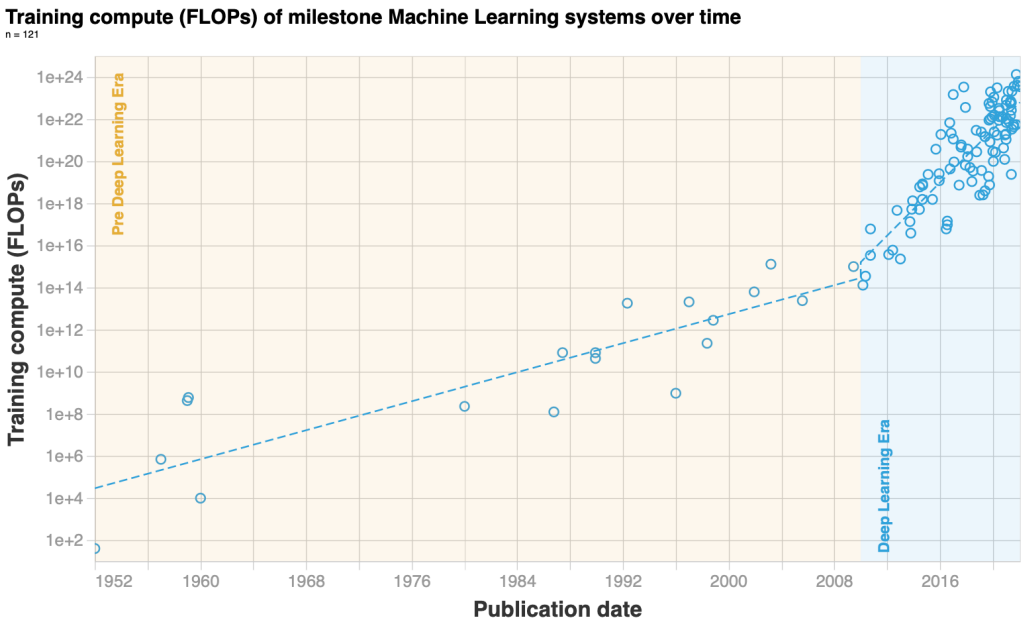
$\approx 2\ ans$

Algo. de DL

$\approx 3\ mois$

Sevilla, Jaime, et al. "Compute Trends Across Three Eras of Machine Learning." *arXiv e-prints* (2022): arXiv-2202.

Thanks to Claudio Gallichio for the slide



Doublement de la puissance de calcul

Loi de Moore

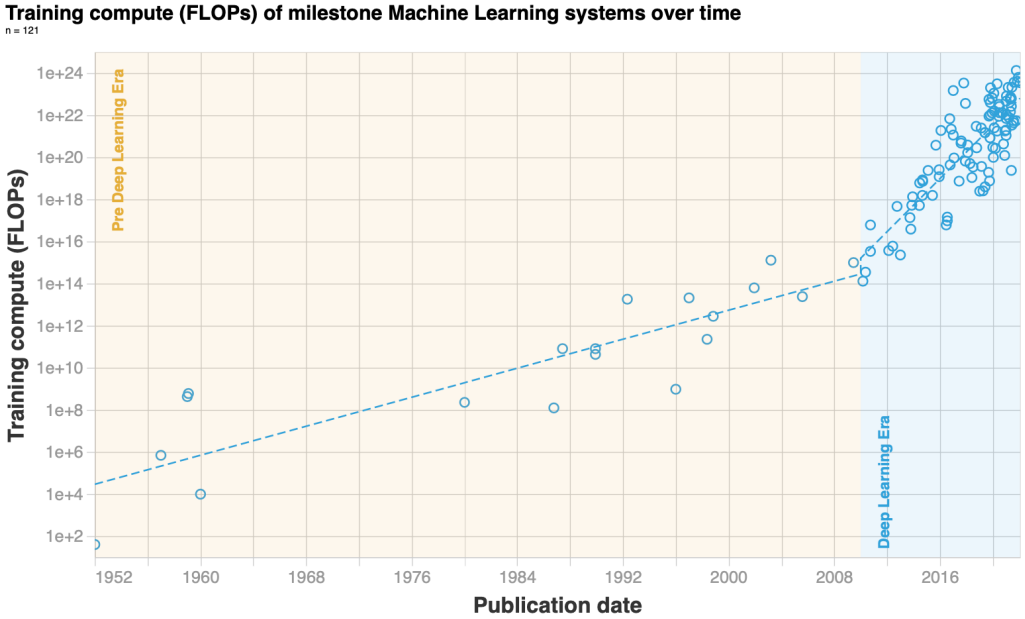
$\approx 2\ ans$

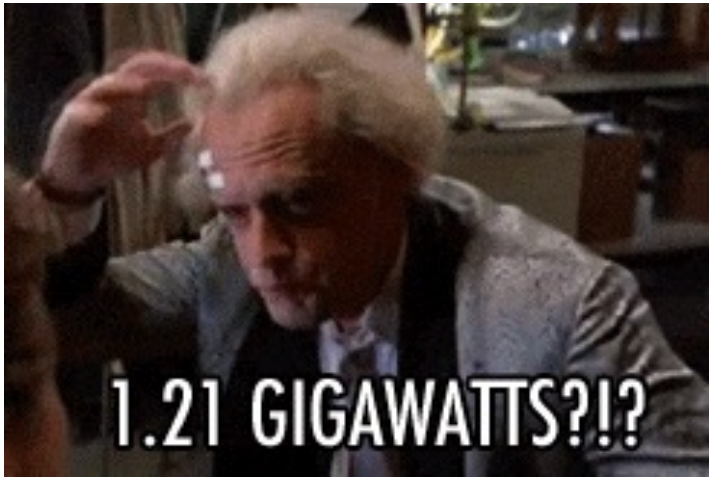
Algo. de DL

$\approx 3\ mois$

Sevilla, Jaime, et al. "Compute Trends Across Three Eras of Machine Learning." *arXiv e-prints* (2022): arXiv-2202.

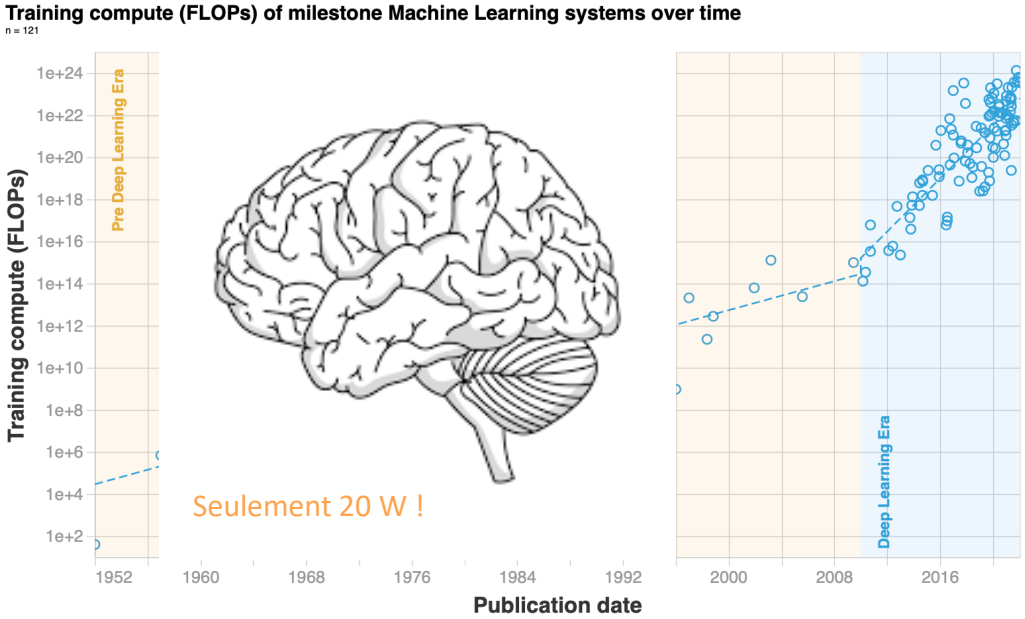
Thanks to Claudio Gallichio for the slide

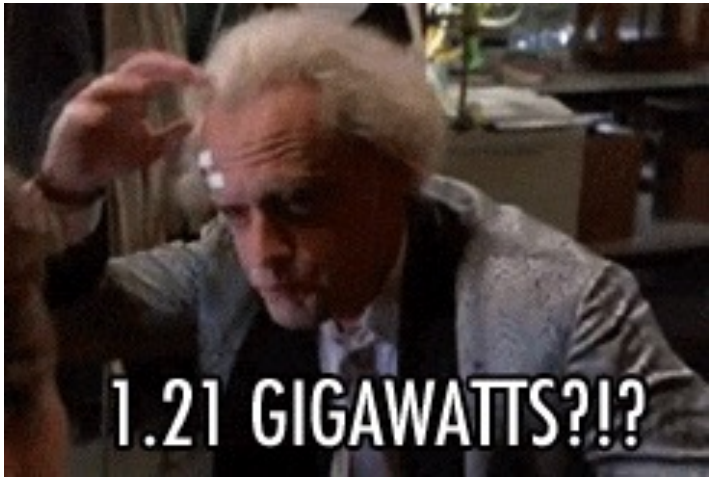




Sevilla, Jaime, et al. "Compute Trends Across Three Eras of Machine Learning." *arXiv e-prints* (2022): arXiv-2202.

Thanks to Claudio Gallichio for the slide





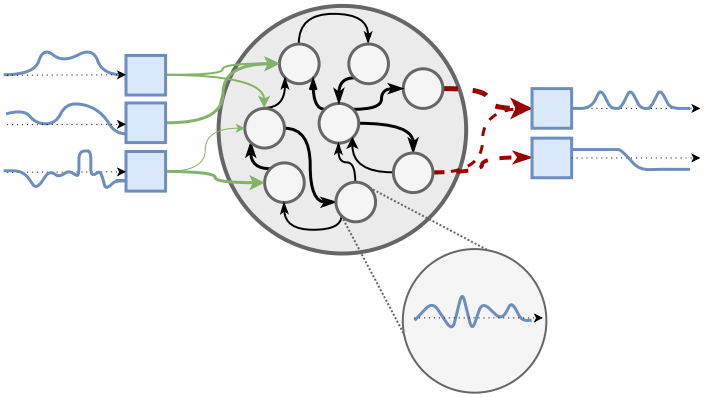
Sevilla, Jaime, et al. "Compute Trends Across Three Eras of Machine Learning." *arXiv e-prints* (2022): arXiv-2202.

Thanks to Claudio Gallichio for the slide

Peut-on avoir un bon équilibre entre performances

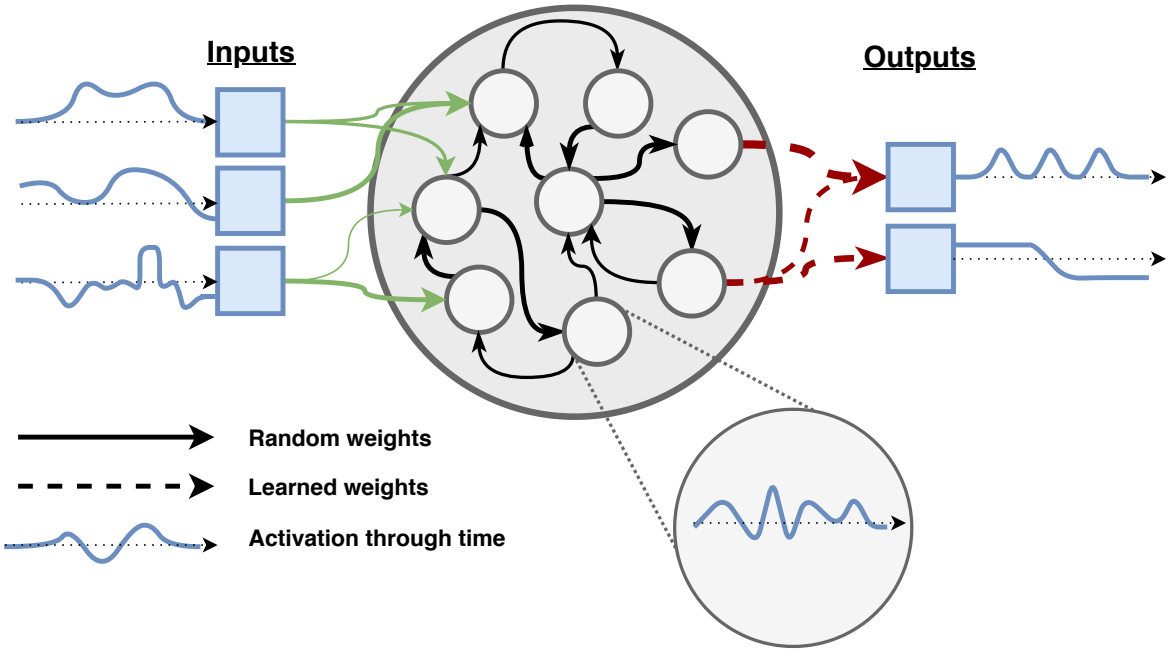
et

efficacité énergétique
+ temps de calcul ?



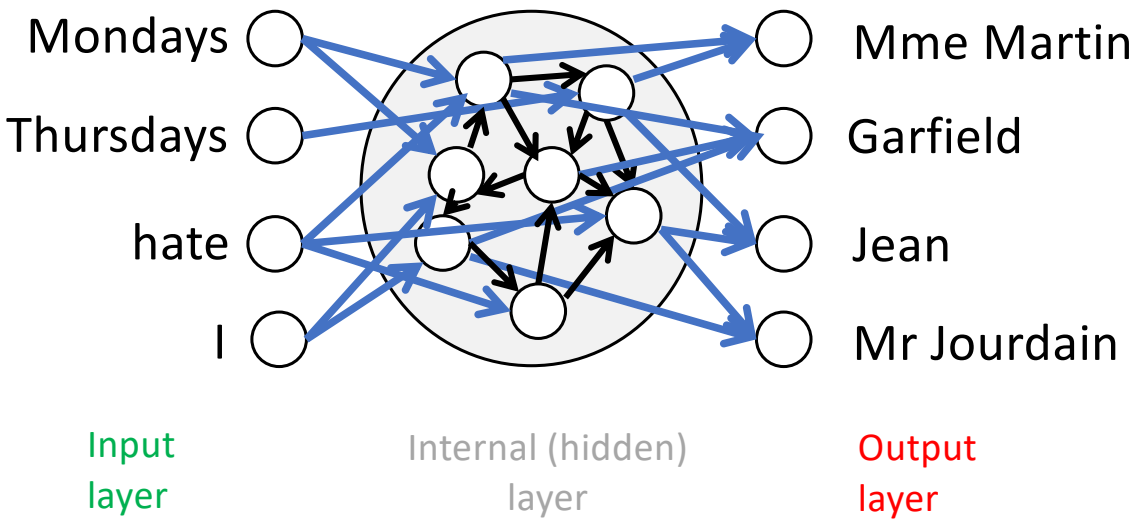
Reservoir Computing

Ceci n'est pas un réseau de neurones récurrent (RNN) classique !

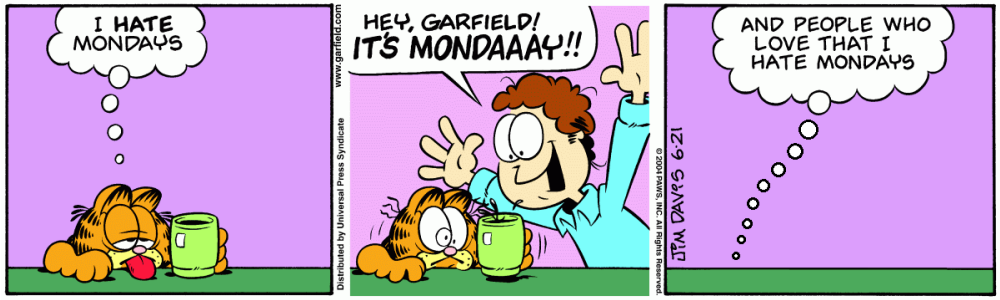


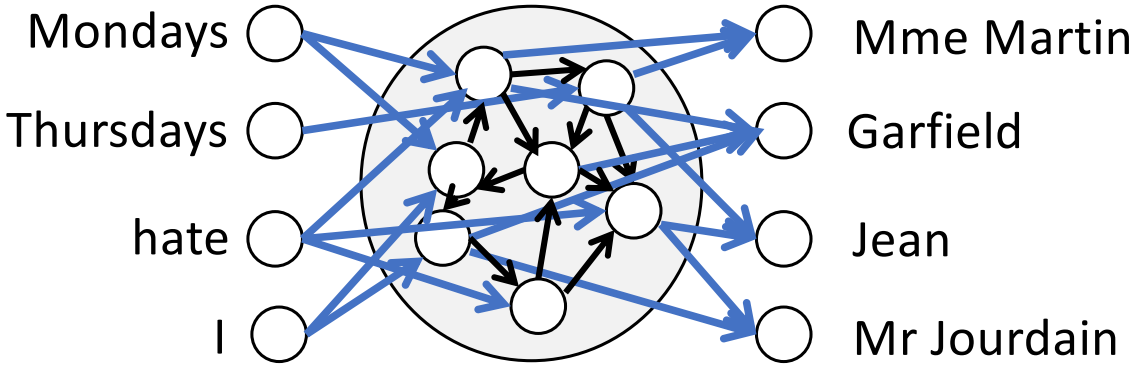
Juven & Hinaut 2020

Le Reservoir Computing exploite les dynamiques naturelles d'un réseau récurrent aléatoire



Le Reservoir Computing exploite les dynamiques naturelles d'un réseau récurrent aléatoire





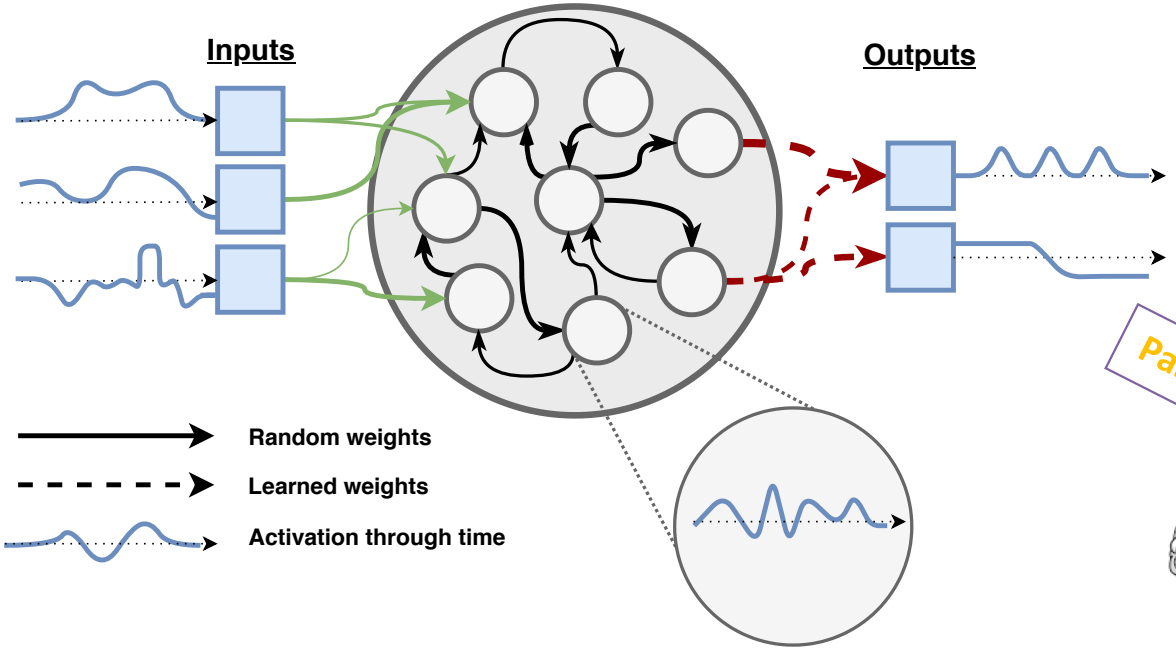
Input layer

Internal (hidden) layer

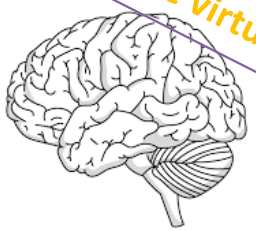
Output layer

Reservoir Computing

Poids aléatoires, MAIS dynamiques déterministes !



Pas besoin de virtualiser le temps

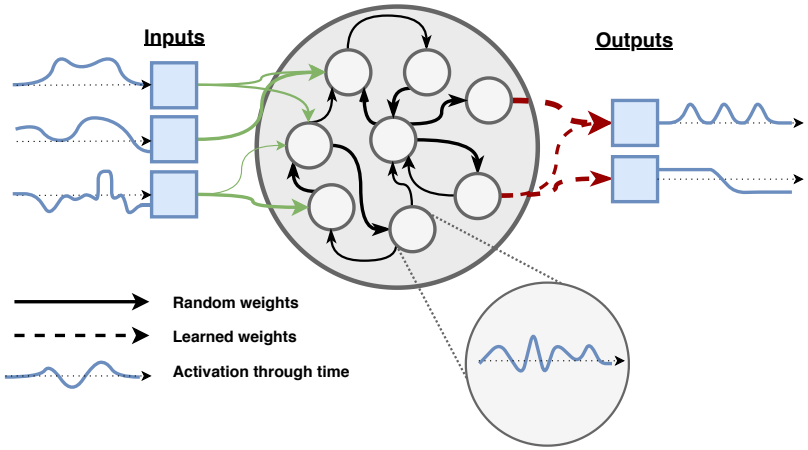


Poids aléatoires = pas d'a priori sur les connexions

Seulement 20 W !

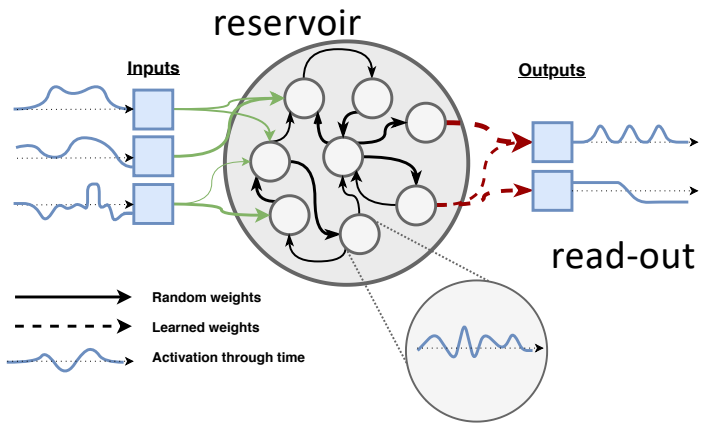
Juven & Hinaut 2020

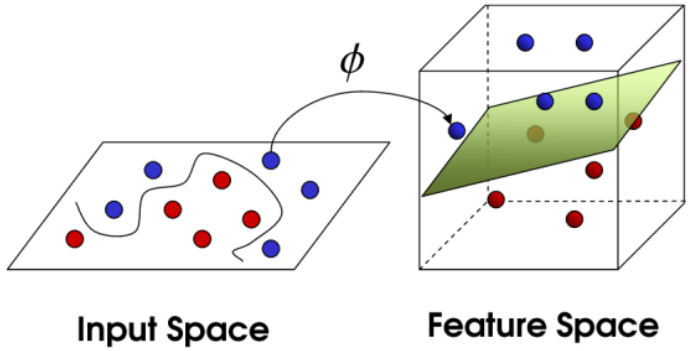
Dominey 1995, Jaeger 2001, Maas et al. 2002 ...


Inputs
Outputs
Random weights
Learned weights
Activation through time

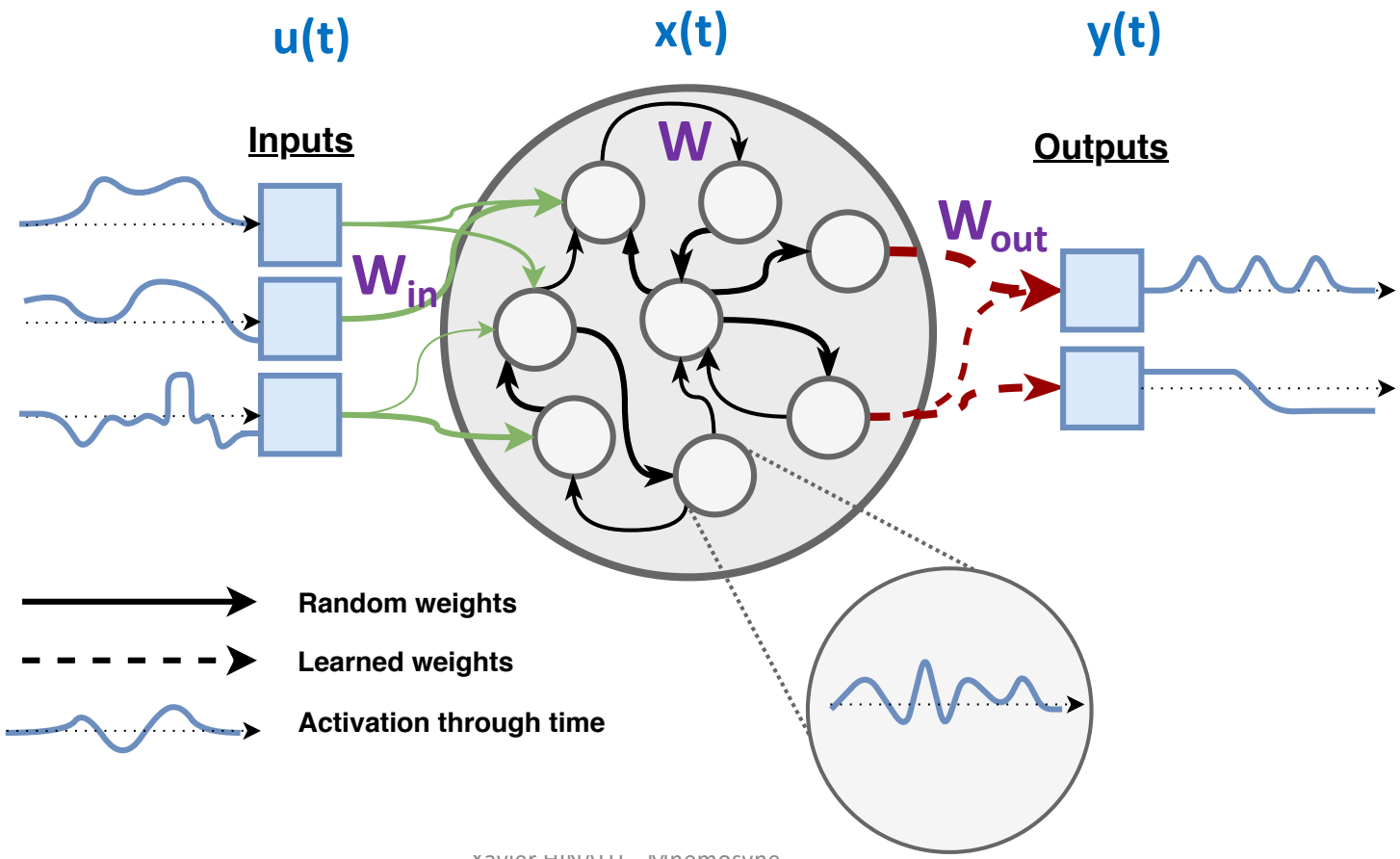
Pourquoi ce nom ?

- Réservoir = couche récurrente
- read-out = couche de sortie
- beaucoup de combinaisons des entrées sont faites dans le reservoir
  - grâce aux projections aléatoires
- “réservoir” est littérallement un réservoir de calculs (= “reservoir computing”)
  - calculs non-linéaires
- dans lequel on vient “piocher”/décoder (= ”read-out”) linéairement
  - les combinaisons qui vont être utiles pour le problème





Reservoir Computing

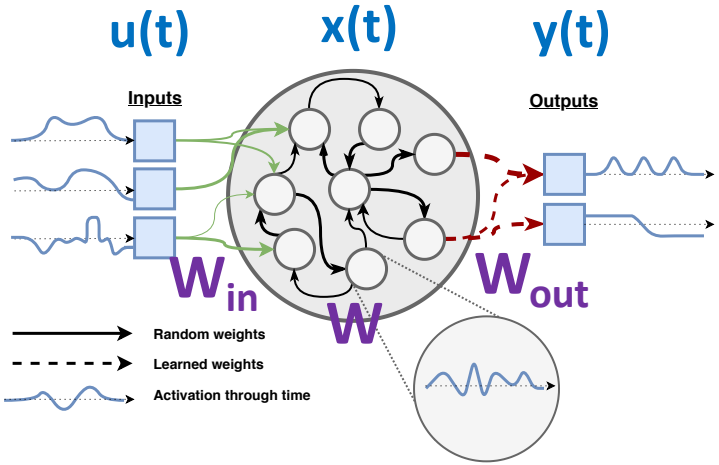


NE REGARDEZ PAS LA PROCHAINE DIAPO !

Reservoir Computing: Echo State Network (ESN)

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

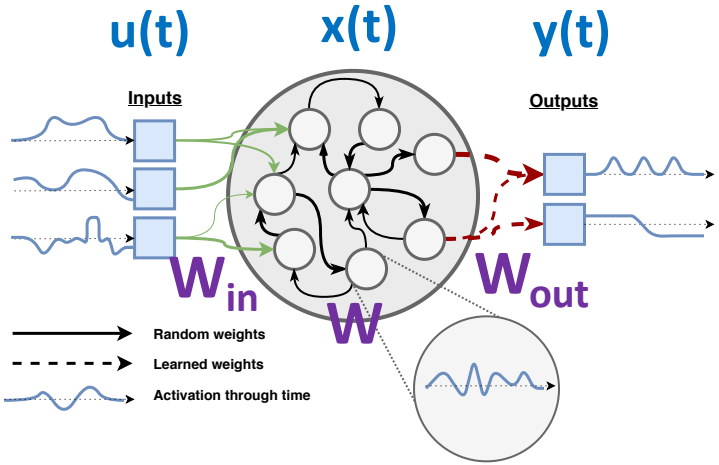
Output (« read-out ») update



Reservoir Computing: Echo State Network (ESN)

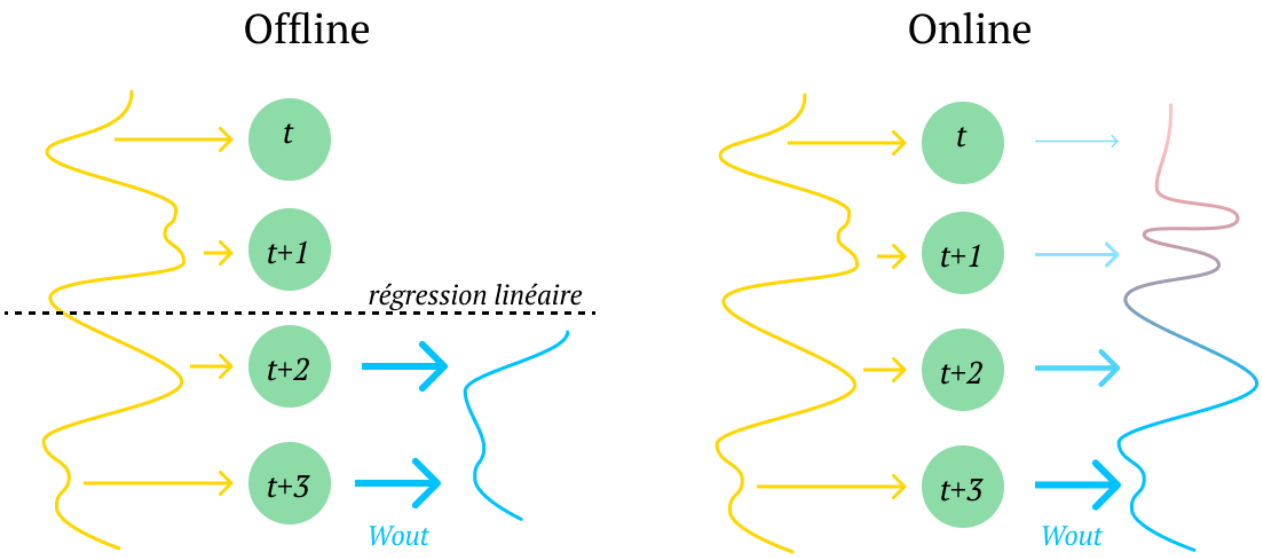
Reservoir state update $x(t) = \left(1 - \frac{1}{\tau}\right) x(t-1) + \frac{1}{\tau} f(W^{in} u(t) + W x(t-1))$

Output (« read-out ») update $y(t) = W^{out} x(t)$

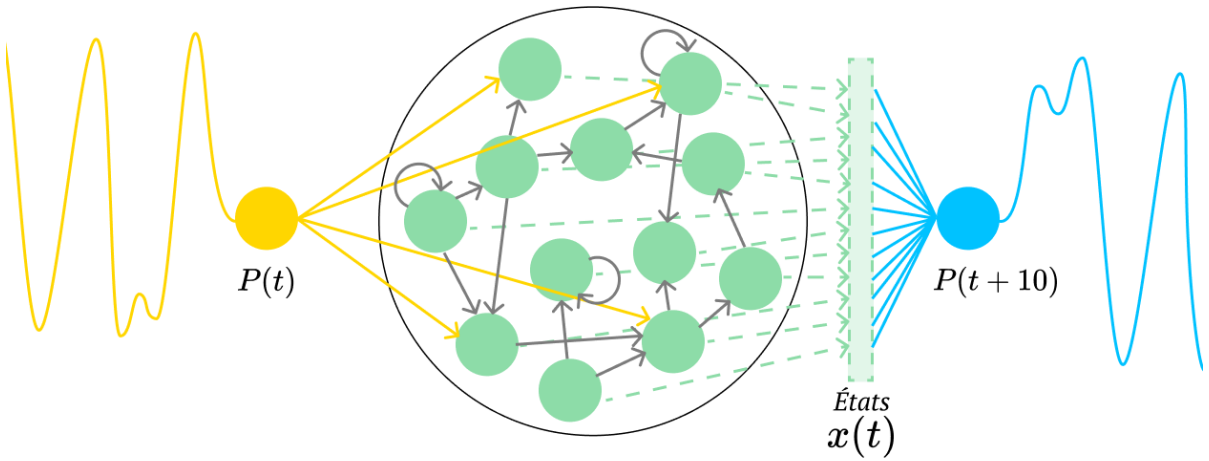


- $u(t)$: inputs
- W^{in}, W, W^{out}: input, recurrent, output matrices
- W^{in} and W matrices are kept random
- Only W^{out} is trained (e.g. ridge regression)
- τ : time constant of reservoir units
  - *leak rate* (LR) is often used instead of $\frac{1}{\tau}$
- f : activation function (usually $tanh$)

Apprentissage “hors-ligne” vs. “en-ligne”



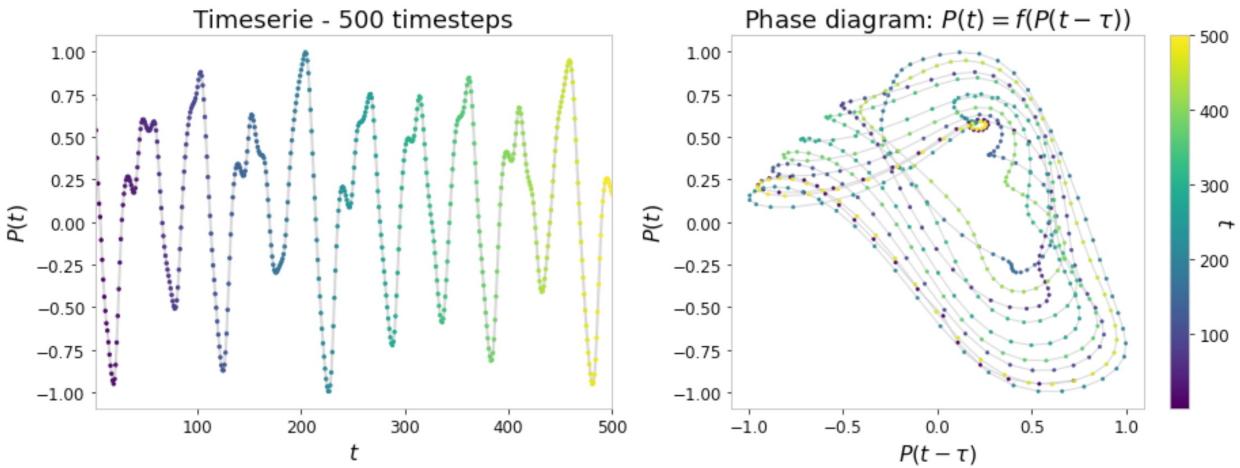
Prédiction de series temporelles



Prédiction de series temporelles

- Les réservoirs sont réputés pour leurs bonnes prédiction de séries temporelles difficiles
  - ex: séries chaotiques (ici Mackey & Glass)

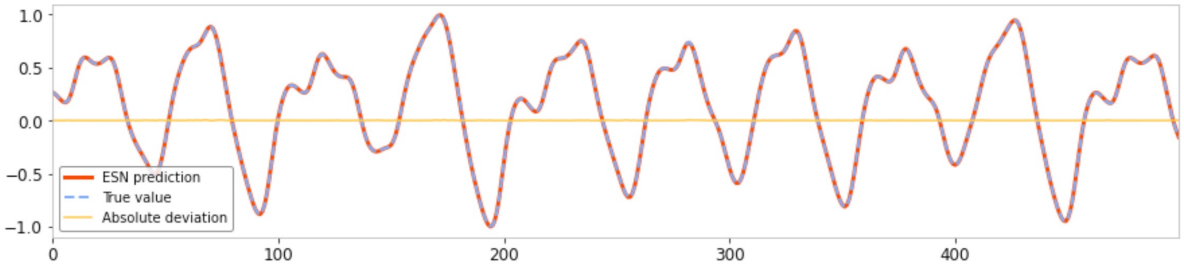
```
plot_mackey_glass(X, t, 500, tau)
```



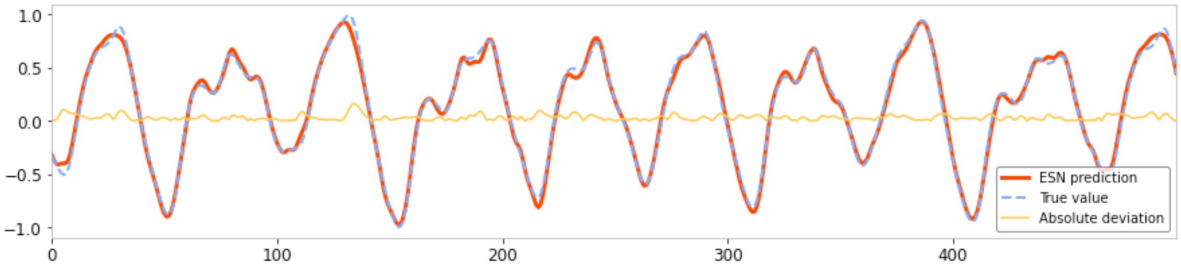
Exemple tiré de github.com/reservoirpy/reservoirpy

Prédiction de series temporelles

- Ex : série chaotique de Mackey & Glass
  - Prédictions obtenues pour 10 pas de temps dans le futur (T+10)



- Prédictions obtenues pour 50 pas de temps dans le futur (T+50)

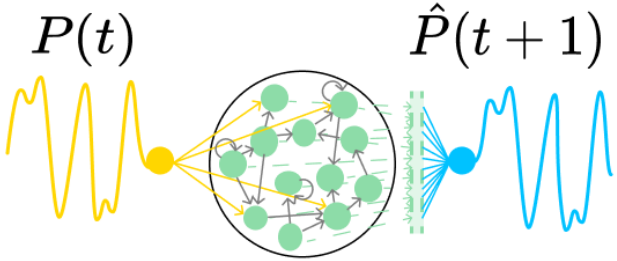


Exemple tiré de github.com/reservoirpy/reservoirpy

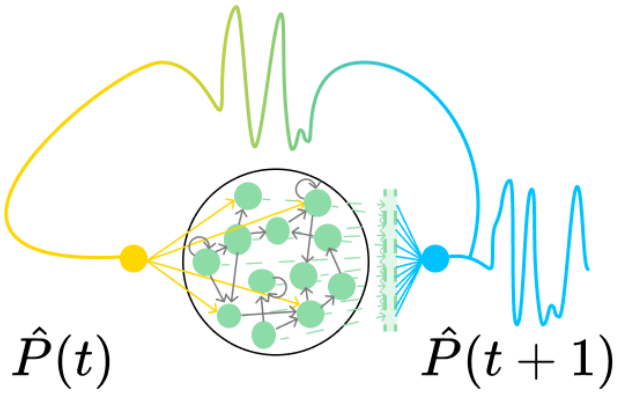
Génération de series temporelles

Génération de séquences en « faisant boucler » la sortie sur les entrées

= Mode « Autorégressif » comme ChatGPT

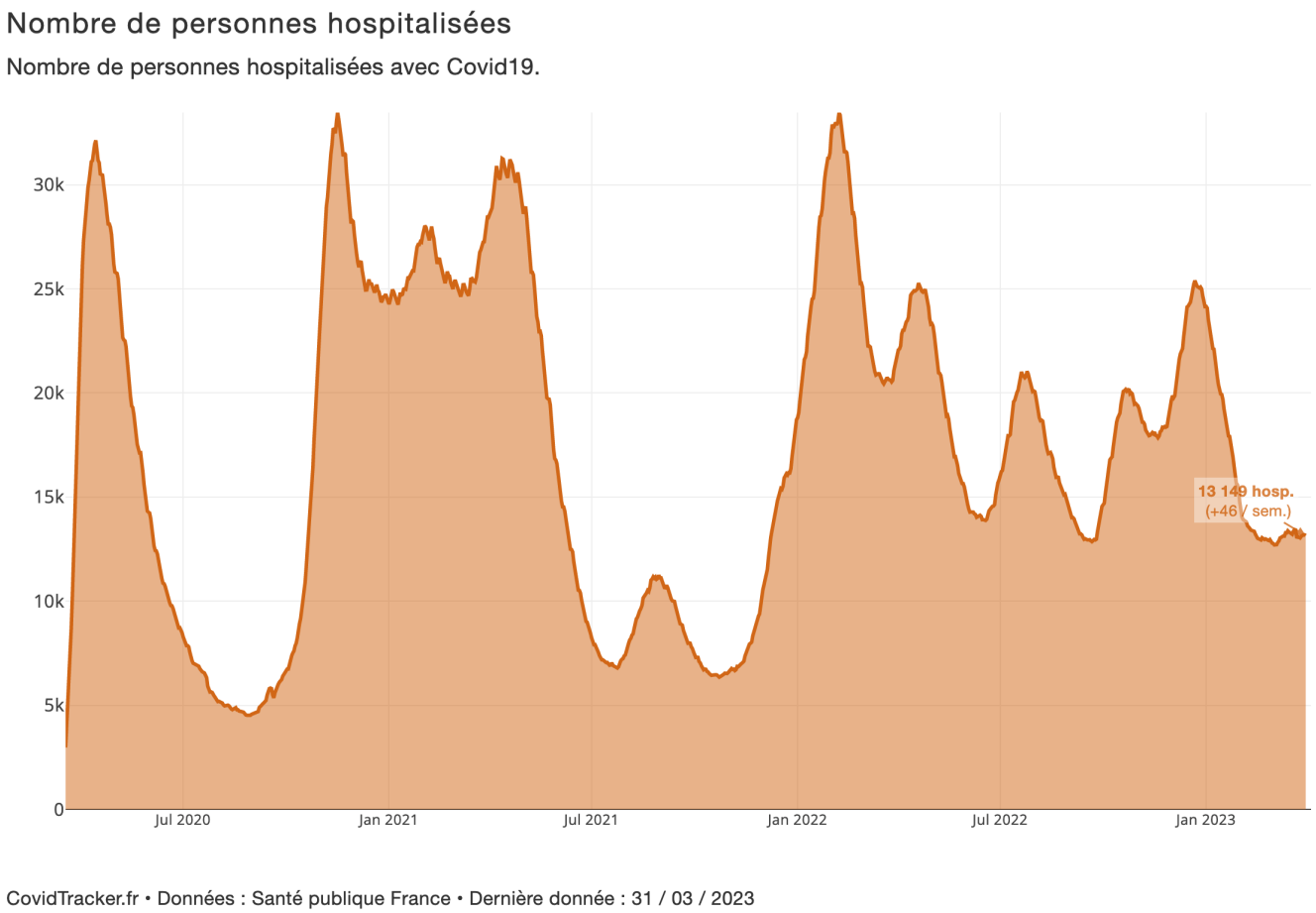


Mode Prédiction



Mode Génération

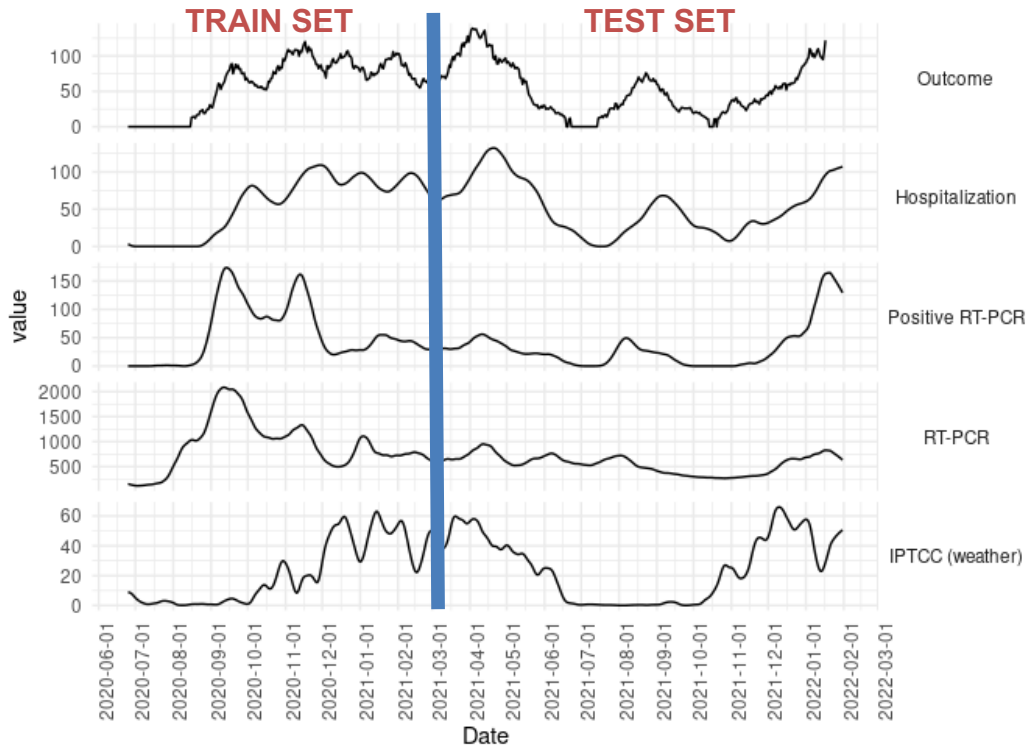
Prédiction hospitalisation COVID à t+14



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

Reservoir Computing + Algorithme Génétique

- **Données** : Données agrégées provenant des dossiers médicaux électroniques + SARS-CoV-2 publics + météo = 409 *features* sur 586 jours.
- **Haute dimension** : 409 *features* ≃ nombre de variables observées chaque jour.
- **Résultat** : Hospitalisations à 14 jours au CHU de Bordeaux.
- **Métrique** : Erreur Absolue Moyenne (MAE)
- **Modèle** : Reservoir Computing (médiane de prédictions de 40 réservoirs)



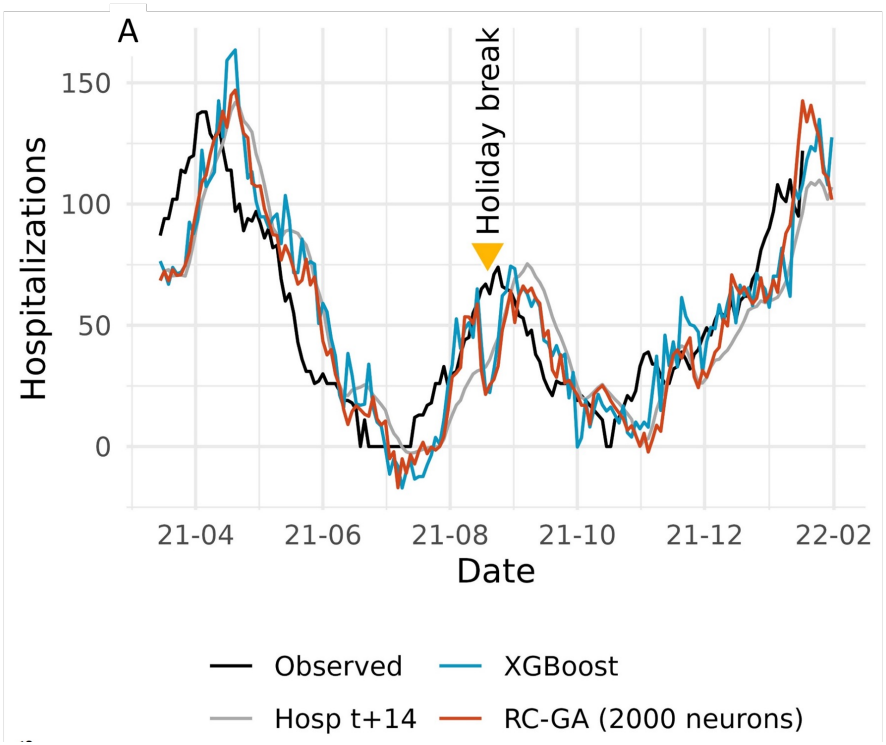
Prédiction hospitalisation COVID à t+14

Model	Mean Absolute Error (± std)
RC-GA (500 neurons)	15.27 (± 12.88)
RC-GA (2000 neurons)	**14.66 (± 12.63)**
RC-RS (500 neurons)	18.45 (± 13.27)
RC-GA-no Feature selection (500 neurons)	17.59 (± 13.35)
Elastic-net	15.83 (± 12.41)
XGBoost	15.45 (± 13.91)
LSTM-PCA	15.74 (± 13.98)
Transformers-PCA	19.28 (± 16.80)
Informer-PCA	18.70 (± 14.11)
PatchTST-PCA	18.34 (± 18.14)
Prophet	21.30 (± 16.33)

Notre approche

Prédiction hospitalisation COVID à t+14

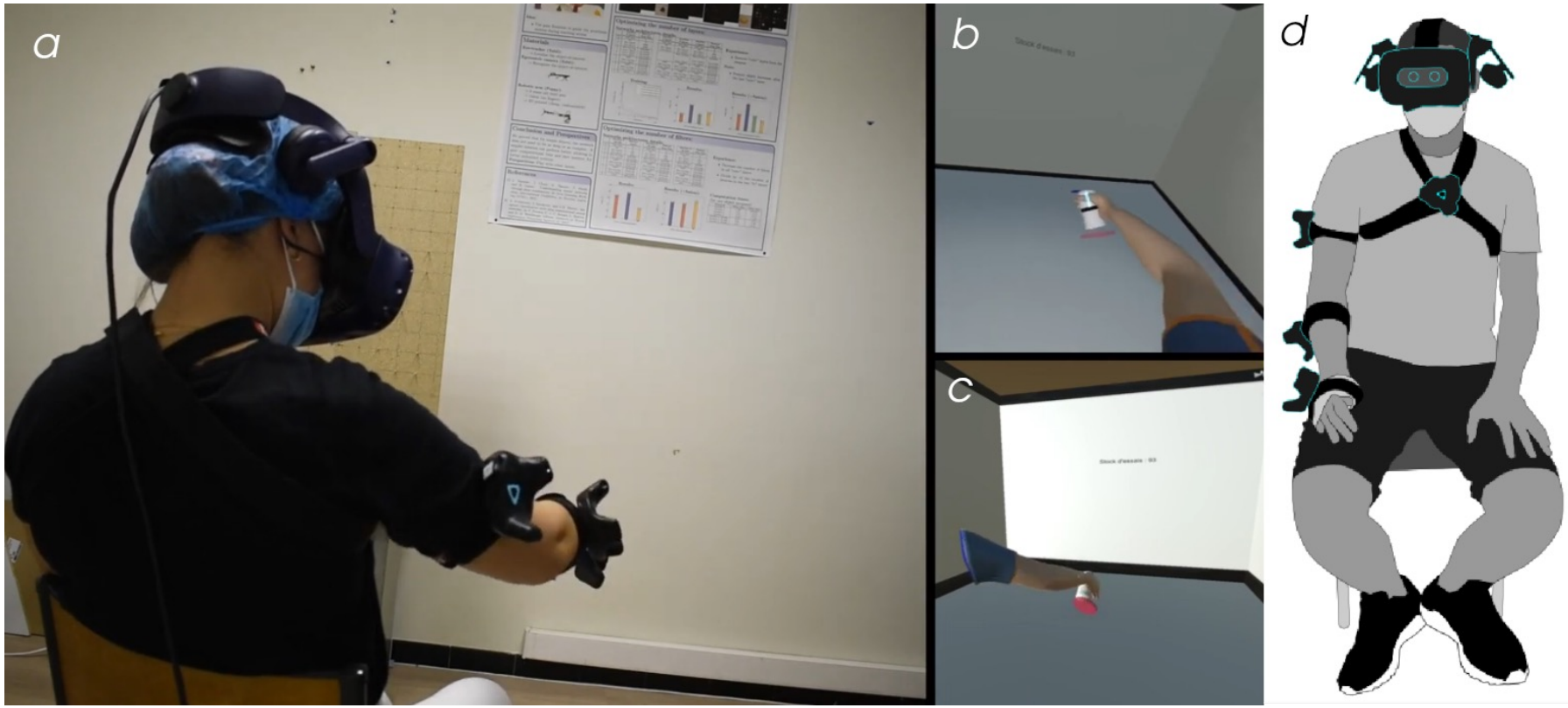
Model	Mean Absolute Error (± std)
RC-GA (500 neurons)	15.27 (± 12.88)
RC-GA (2000 neurons)	**14.66 (± 12.63)**
RC-RS (500 neurons)	18.45 (± 13.27)
RC-GA-no Feature selection (500 neurons)	17.59 (± 13.35)
Elastic-net	15.83 (± 12.41)
XGBoost	15.45 (± 13.91)
LSTM-PCA	15.74 (± 13.98)
Transformers-PCA	19.28 (± 16.80)
Informer-PCA	18.70 (± 14.11)
PatchTST-PCA	18.34 (± 18.14)
Prophet	21.30 (± 16.33)



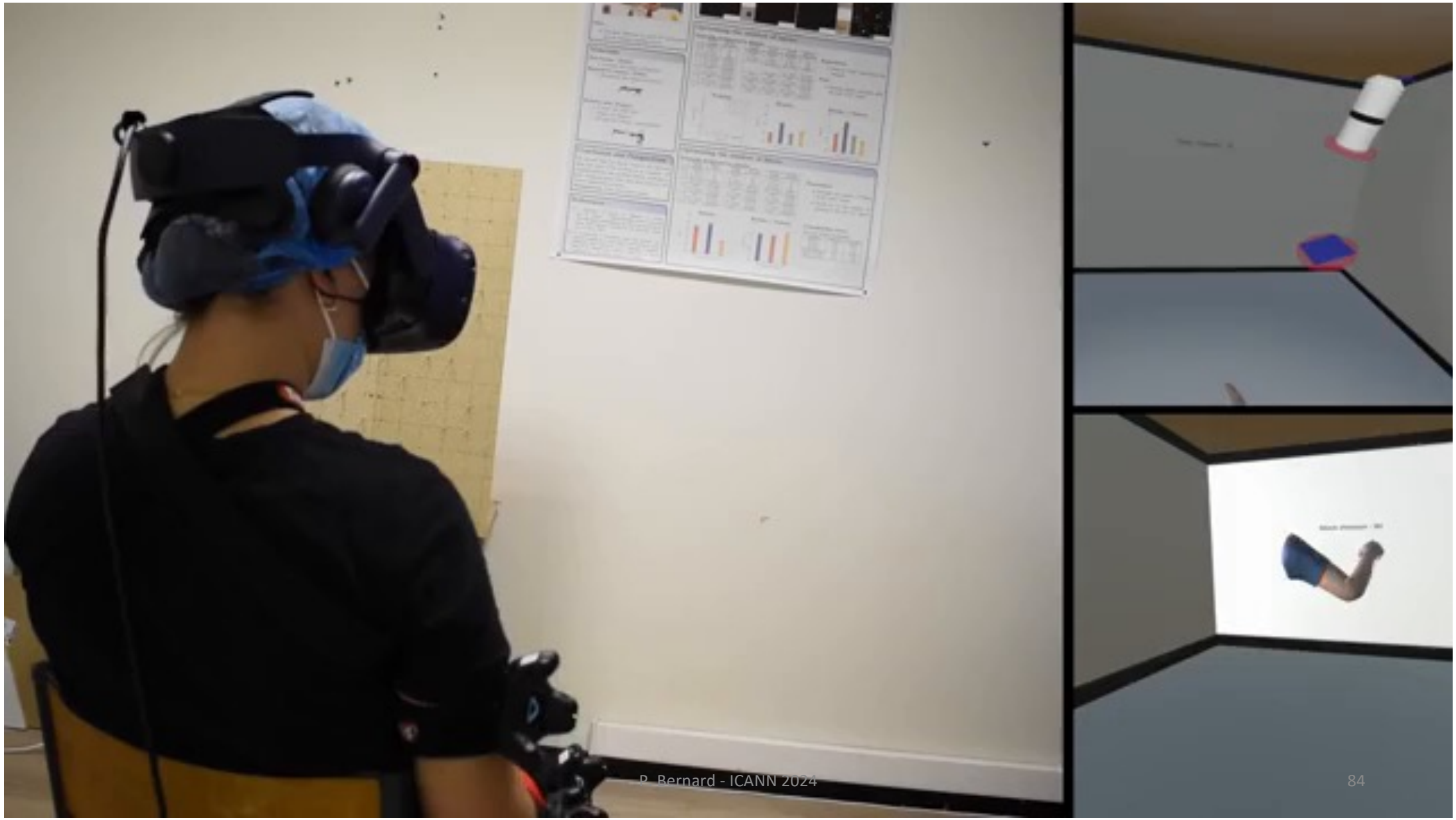
EXEMPLE D’APPLICATION

CONTROLE DE PROTHÈSE BIOMIMÉTIQUE

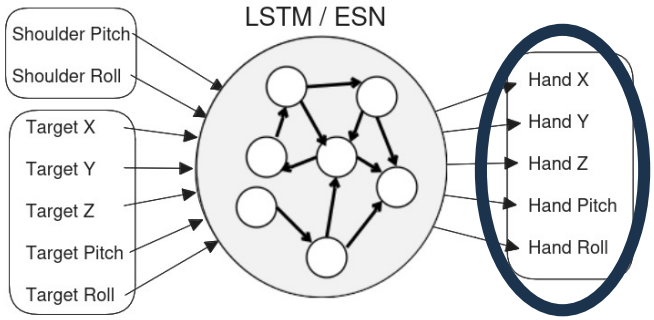
Controle de prothèse biomimétique

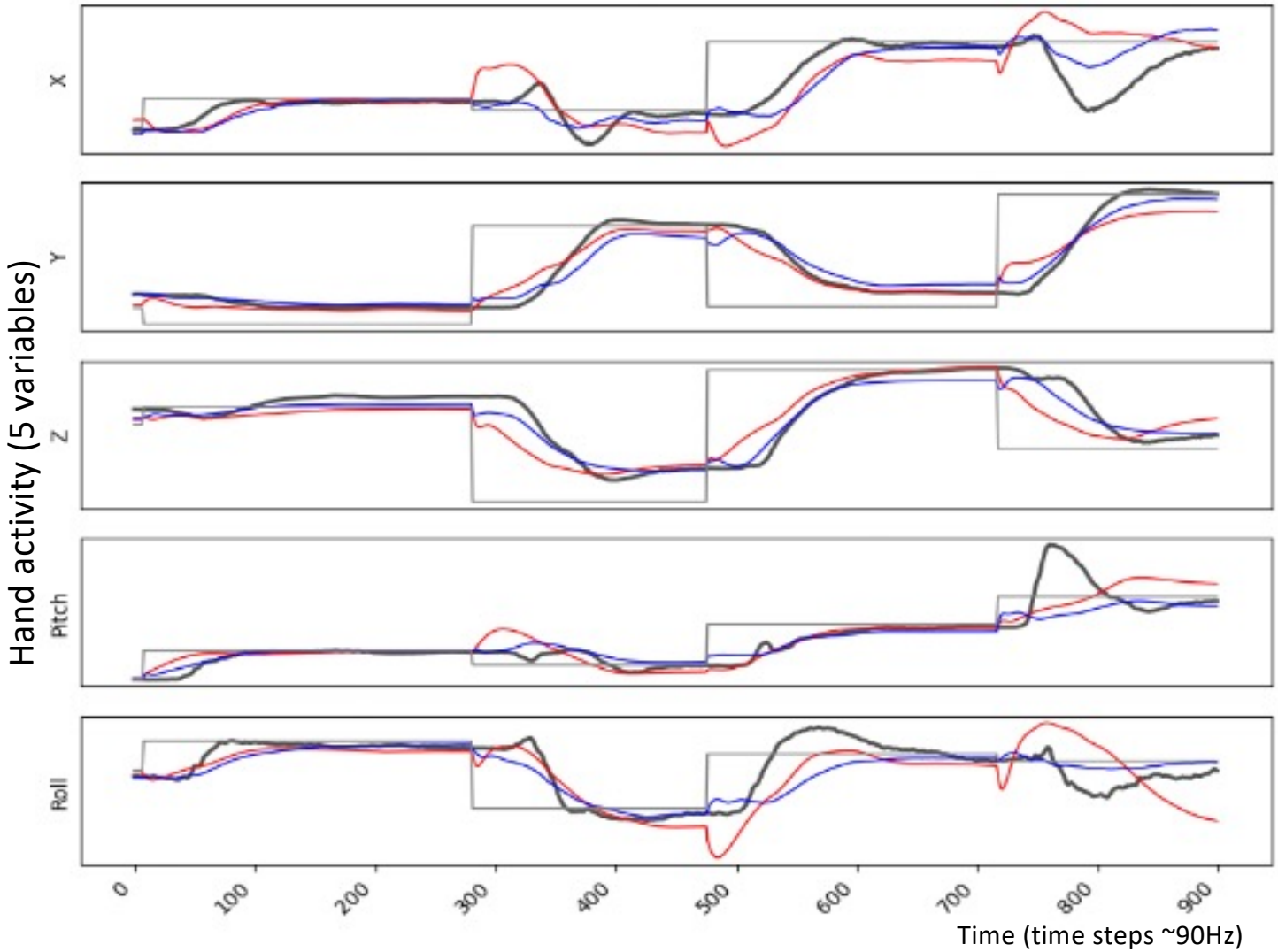


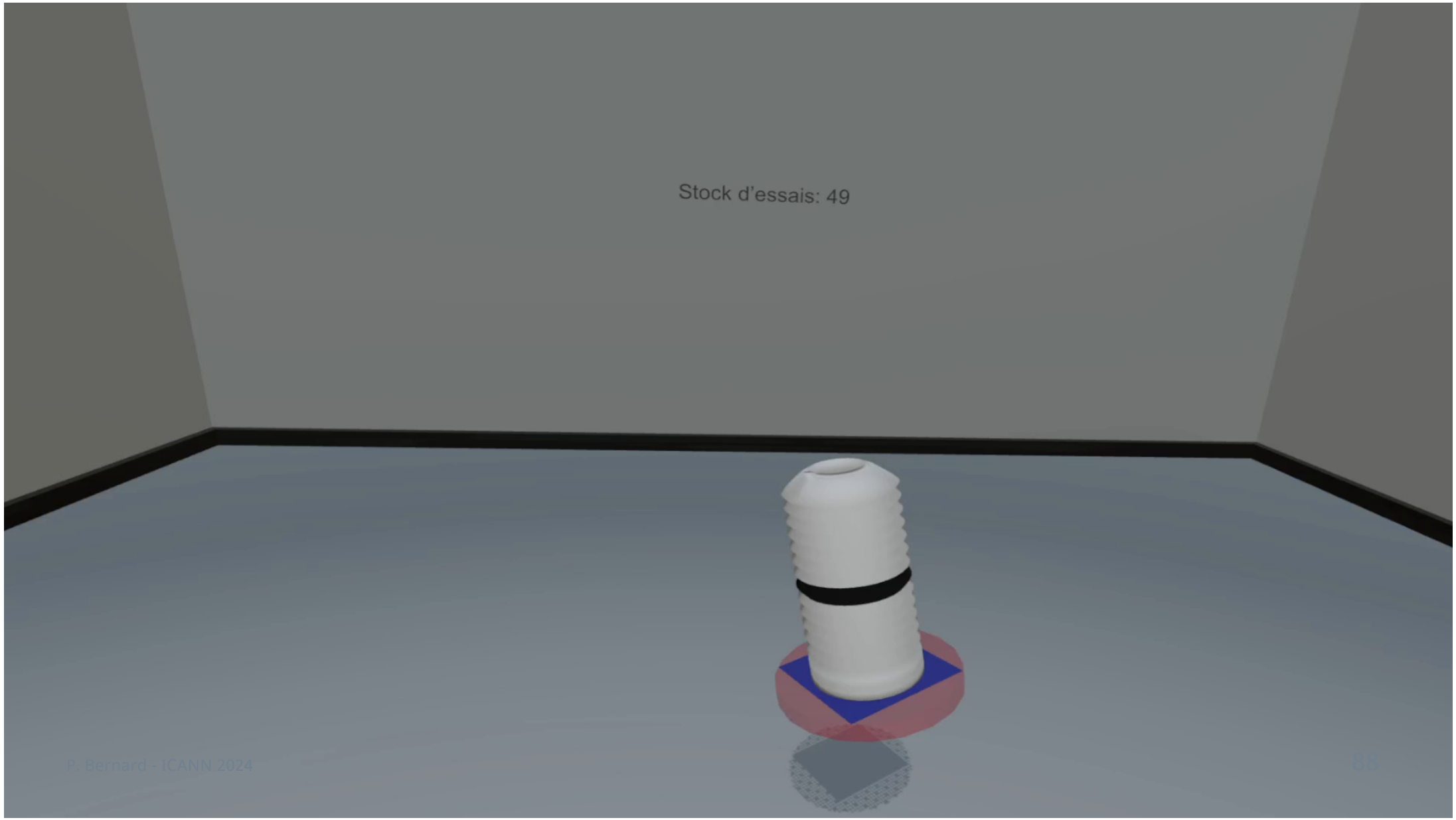
Segas, 2023



Résultats de l'apprentissage sur un seul sujet

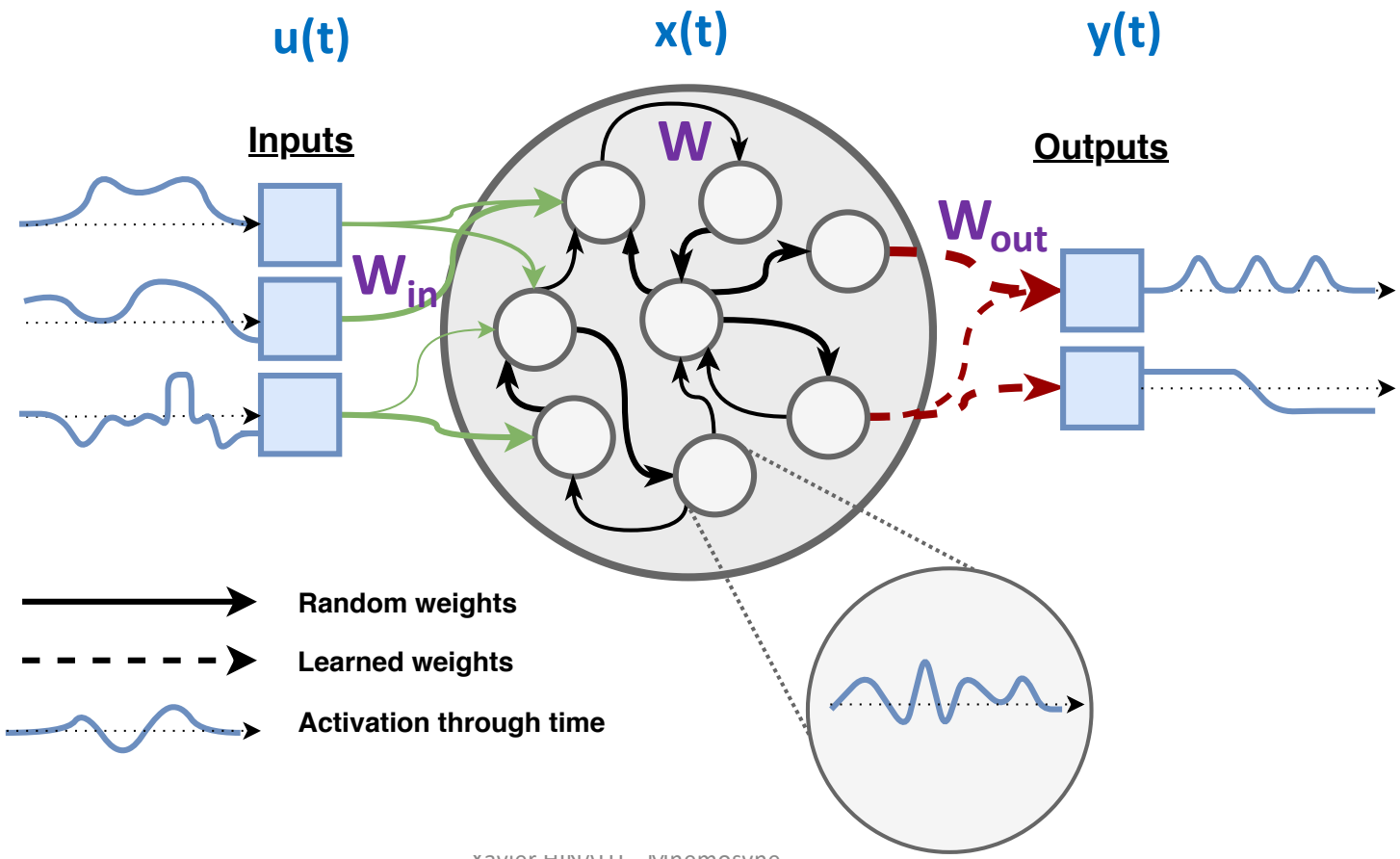





Stock d'essais: 49

Plongeons dans les profondeurs du Reservoir Computing

Reservoir Computing



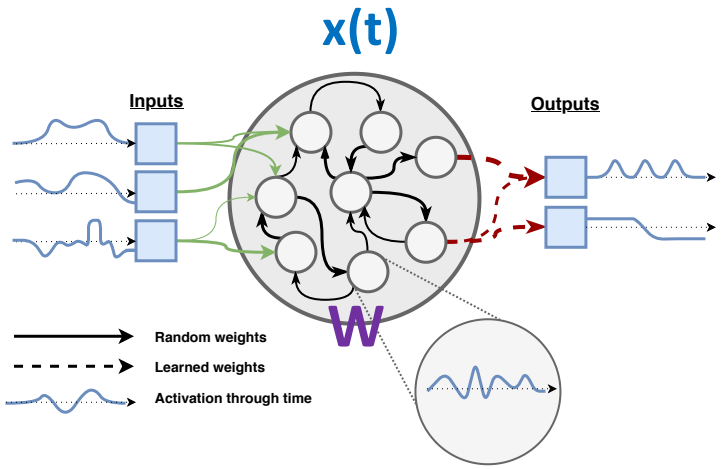
Pourquoi est-ce que ça marche ?

Pourquoi est-ce que ça marche ?

Pourquoi est-ce que ça marche ?

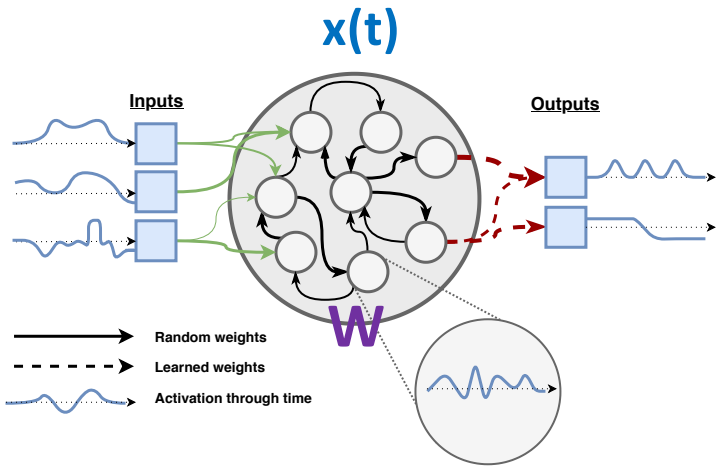
Spectral Radius (rayon spectral)

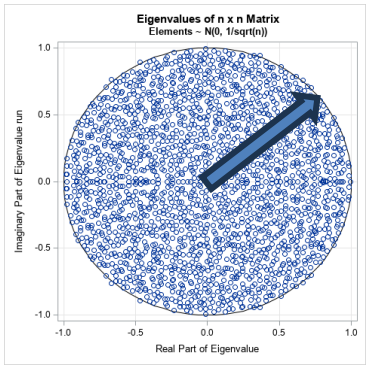
= scaling des connexions récurrentes



Spectral Radius (rayon spectral)

= scaling des connexions récurrentes



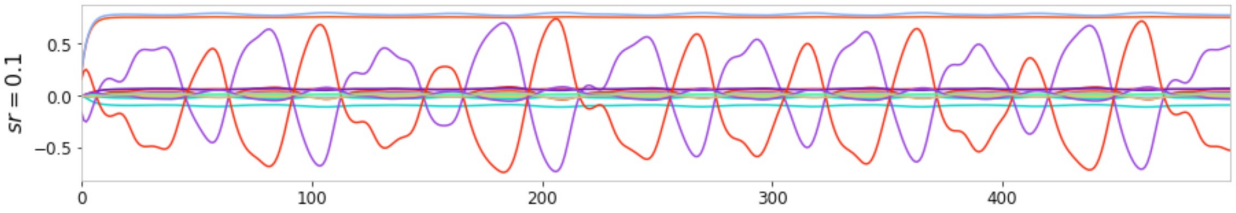


Rayon spectral de la matrice W

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

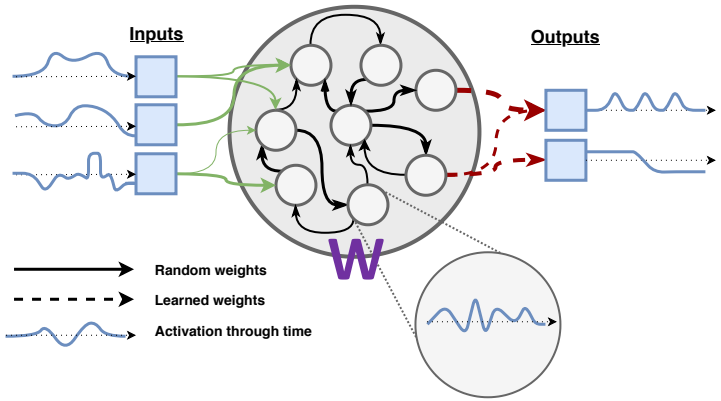
Spectral Radius (rayon spectral)

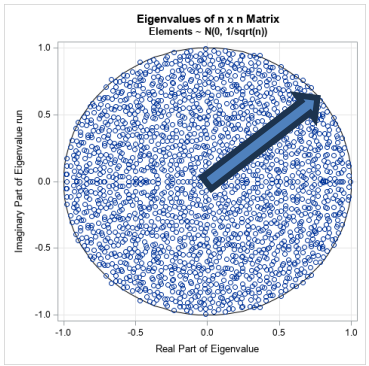
= scaling des connexions récurrentes



x(t)

x(t)



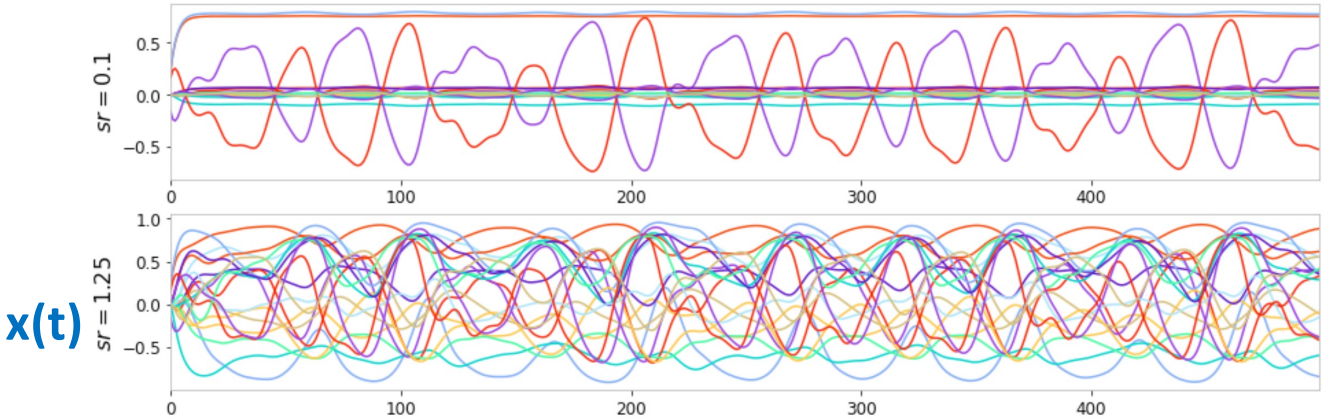


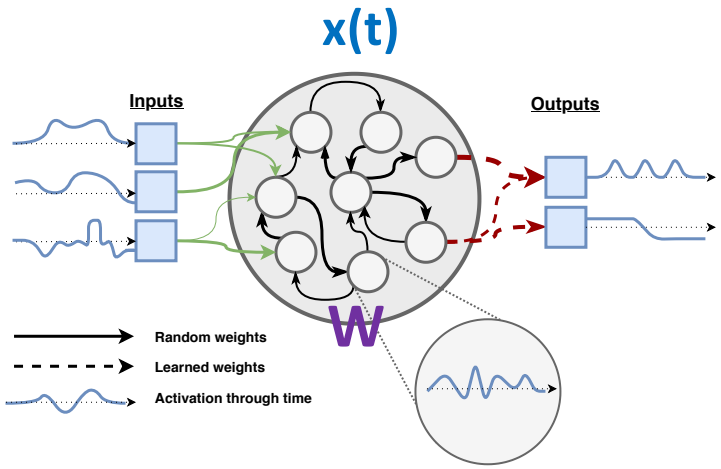
Rayon spectral de la matrice W

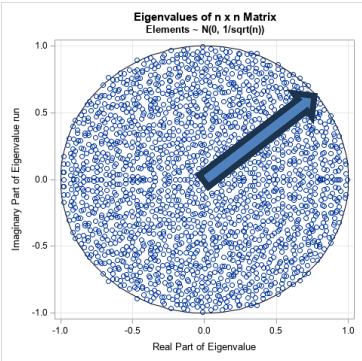
https://blogs.sas.com/content/iml/2020/04/27/circular-law-eigenvalues.html

Spectral Radius (rayon spectral)

= scaling des connexions récurrentes



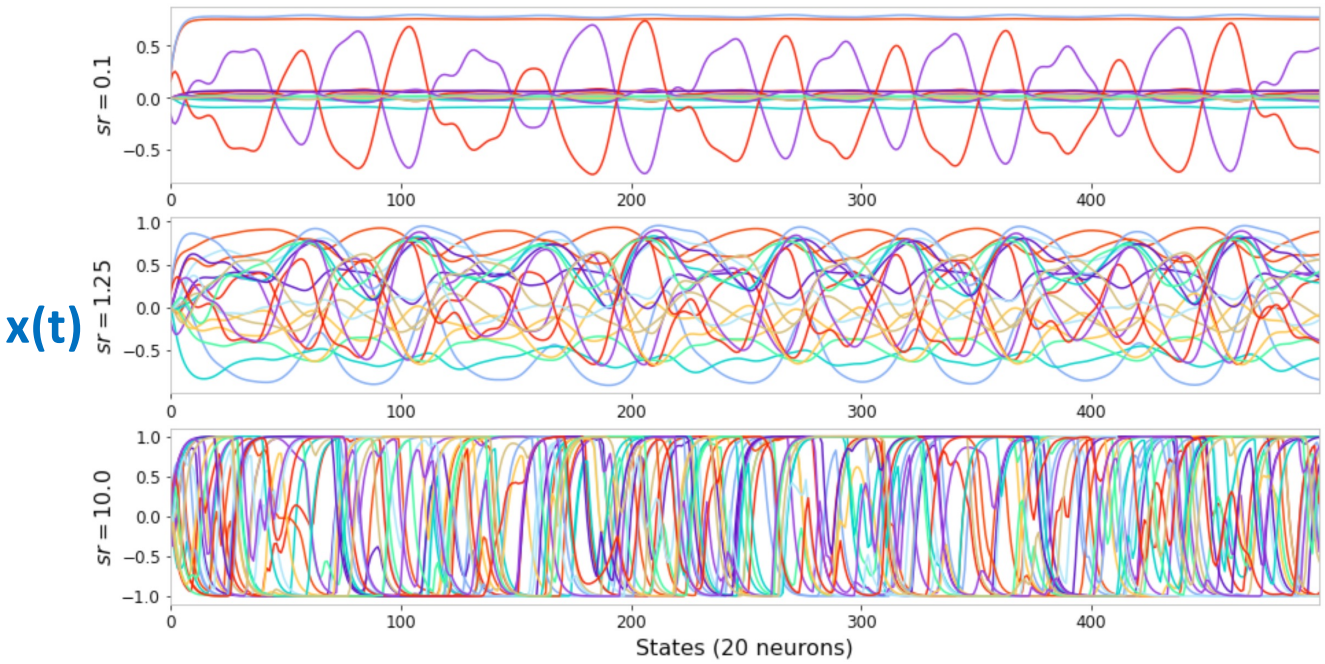


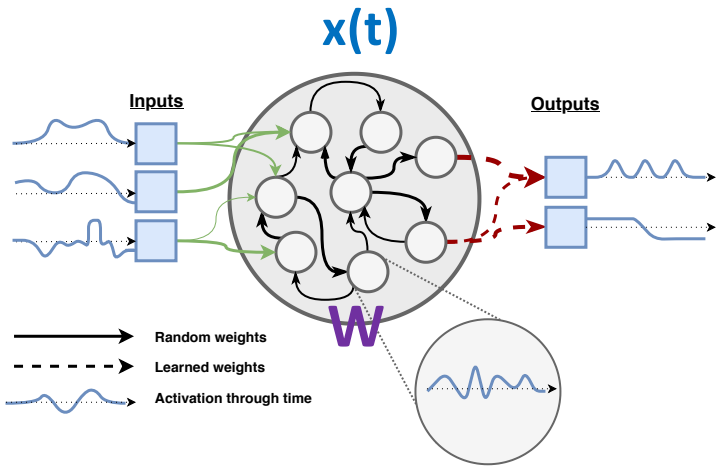


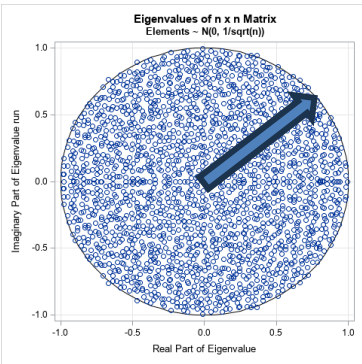
Rayon spectral de la matrice W

Spectral Radius (rayon spectral)

= scaling des connexions récurrentes







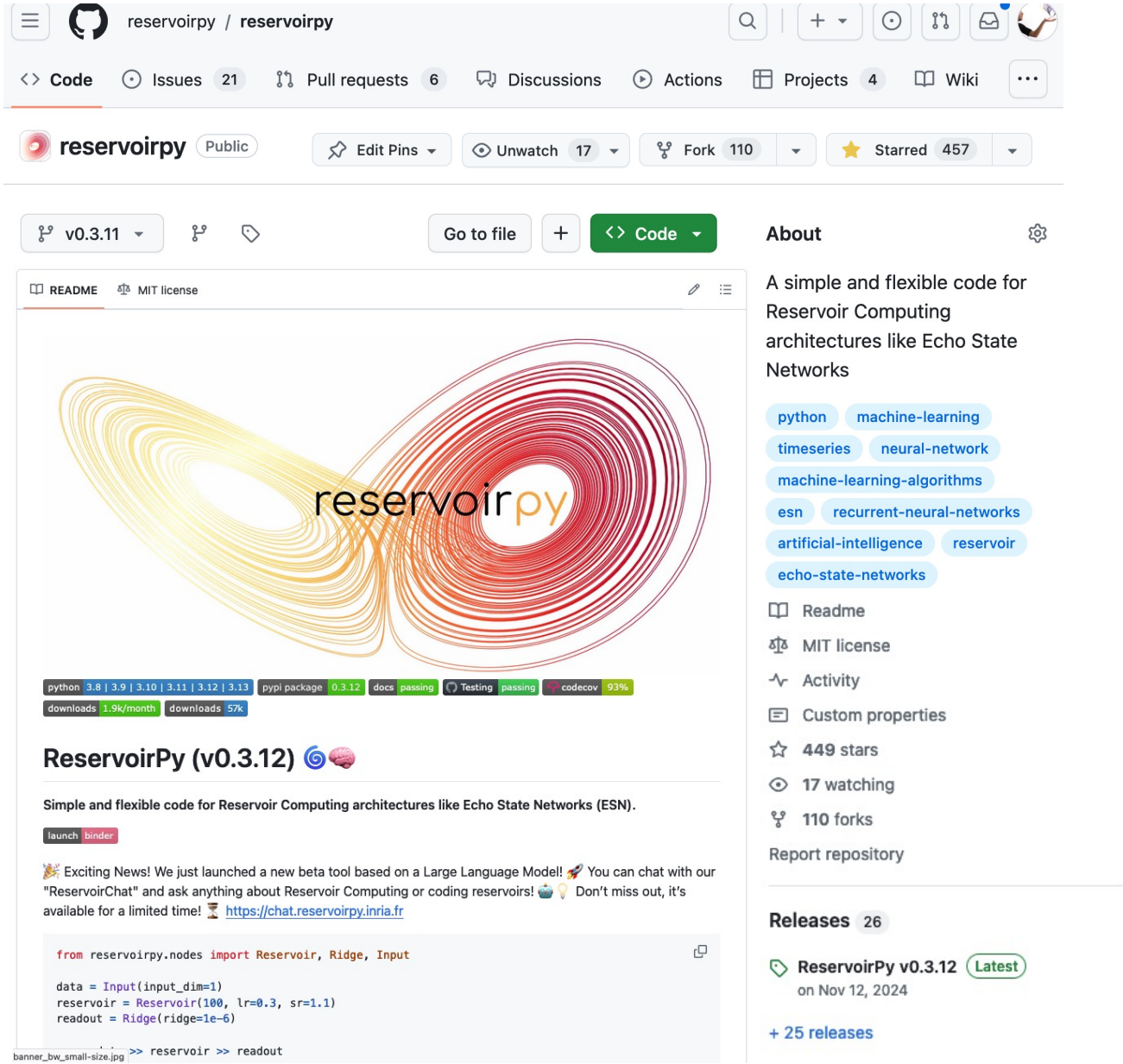
Rayon spectral de la matrice W

RESERVOIRPY ET EXEMPLE D'APPLICATION

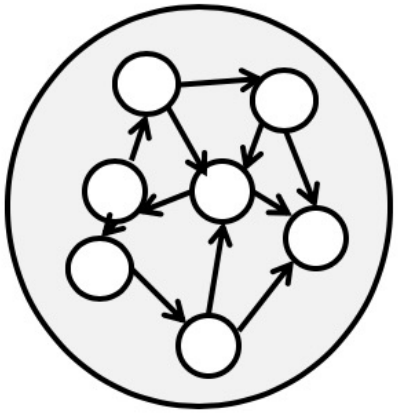
GÉNERATION DE MUSIQUE
+ INFLUENCE DES HYPERPARAMÈTRES

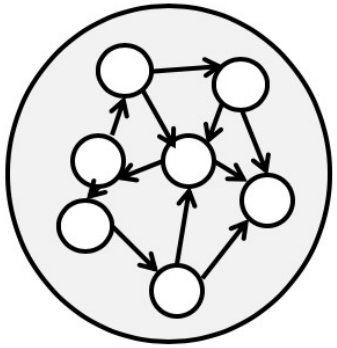
reservoirpy

github.com/reservoirpy/reservoirpy



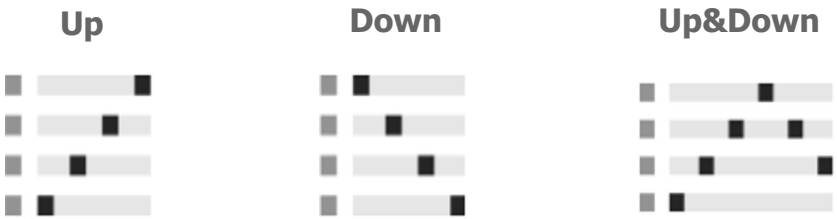
Avez-vous déjà rêvé de jouer d'un réservoir ?



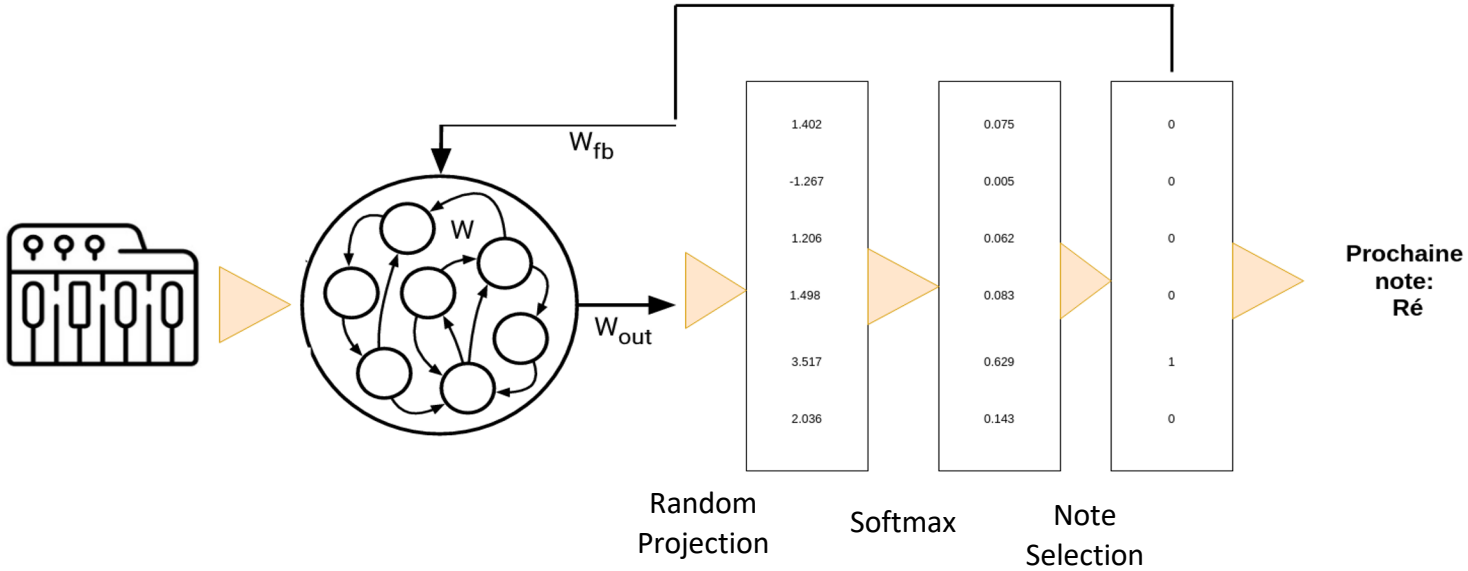


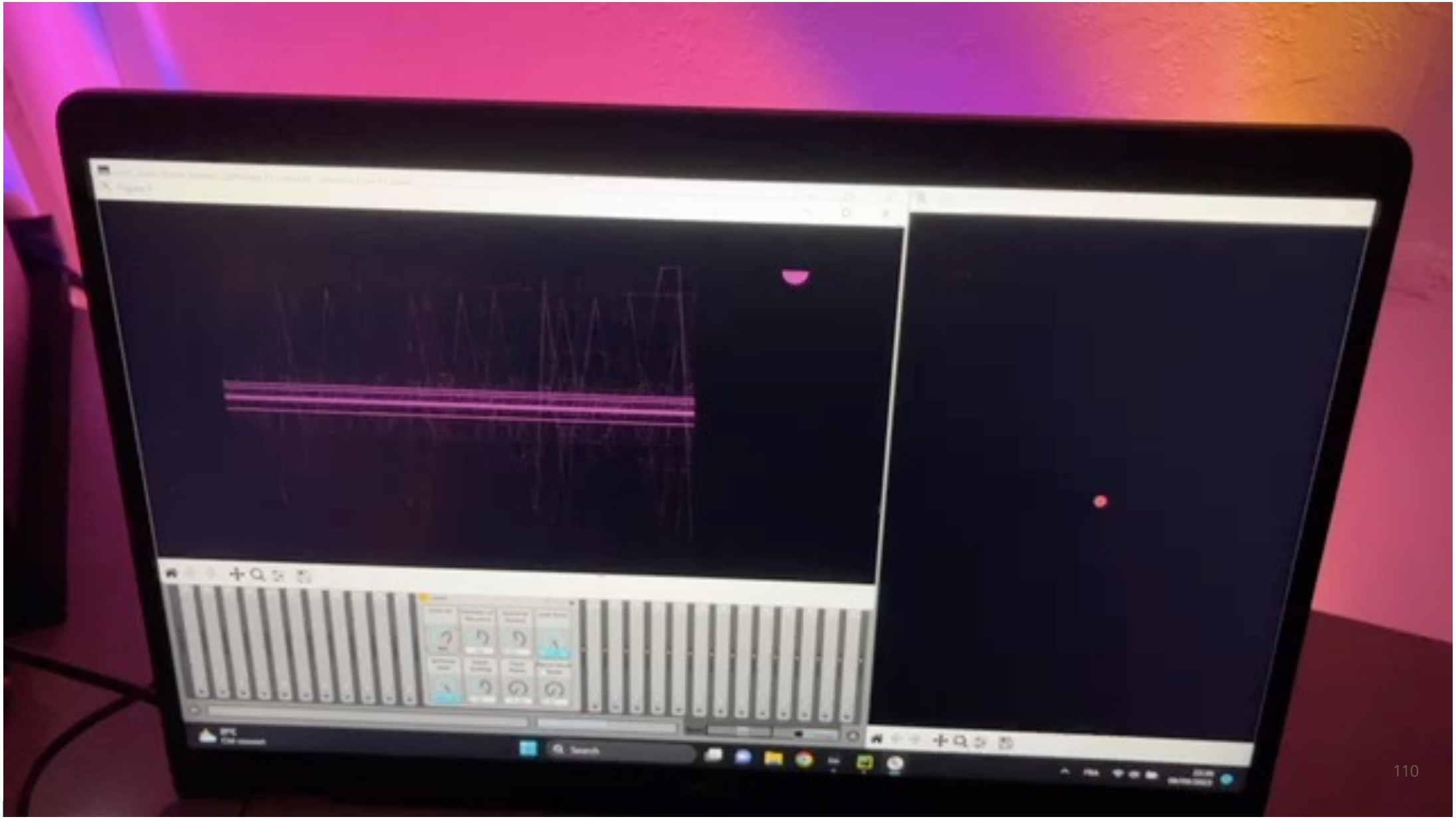
SEQUENTIAL

Classical arpegiators



“RéMi”: the Reservoir MIDI

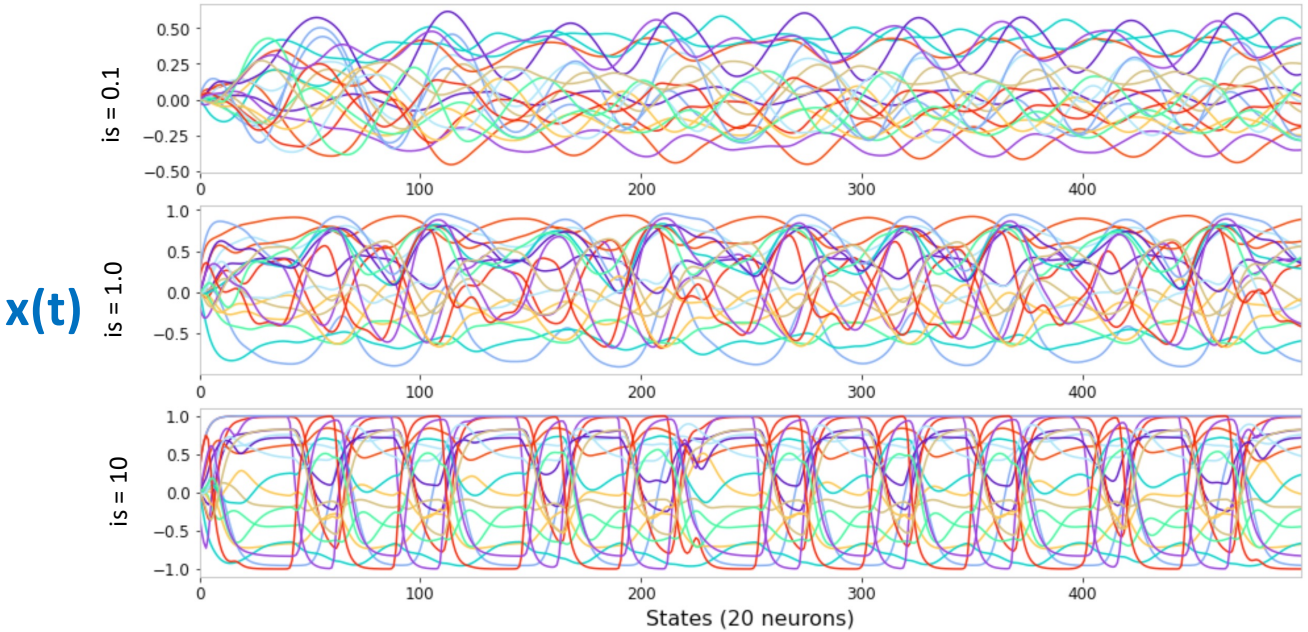


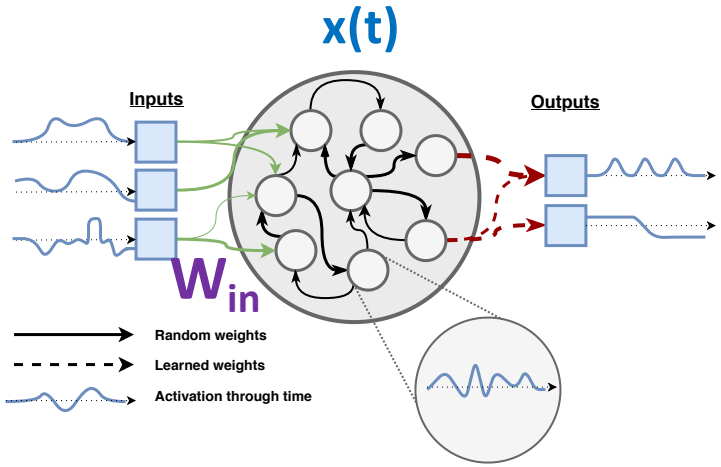


En pratique ? (Hypster-paramètres)

Input Scaling

= scaling des connexions entrantes

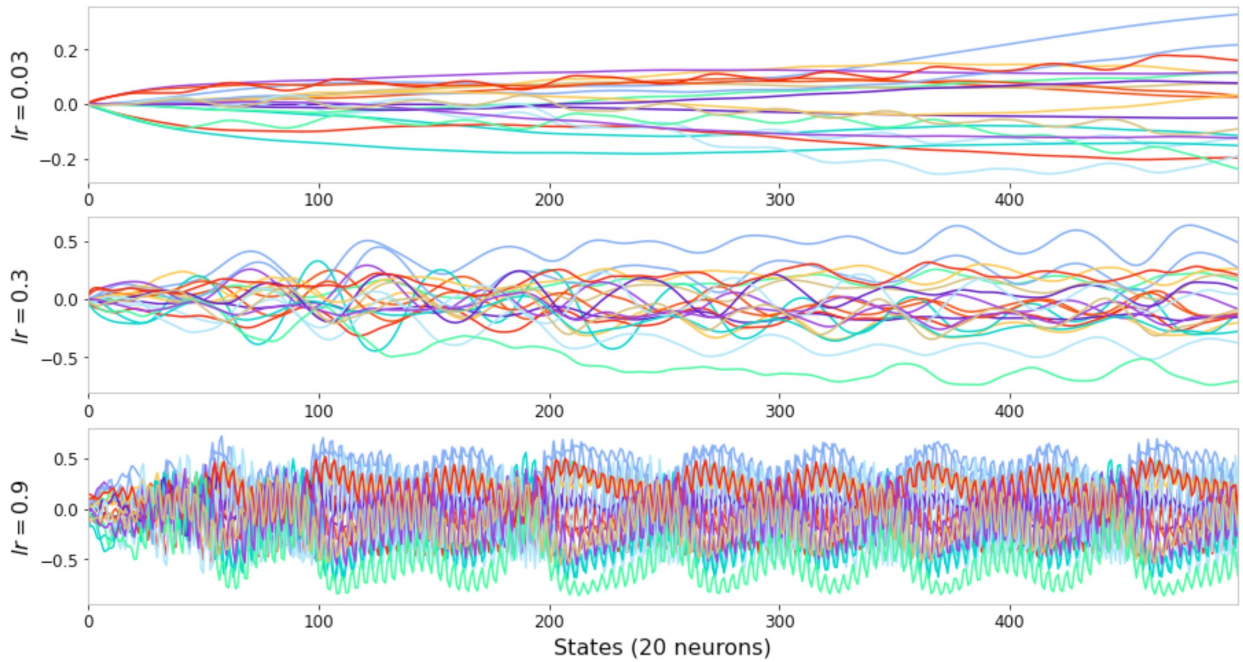




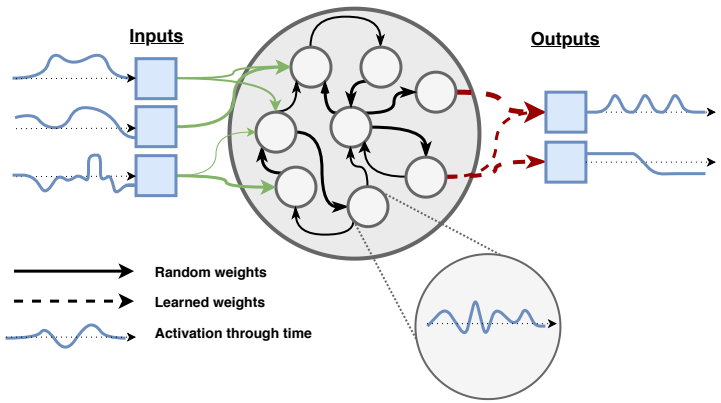
Leak-rate (=taux de fuite)

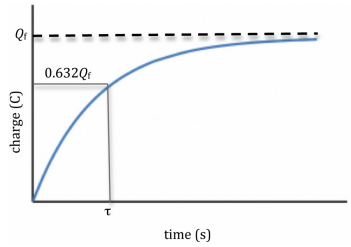
= inverse de la constante de temps des neurones

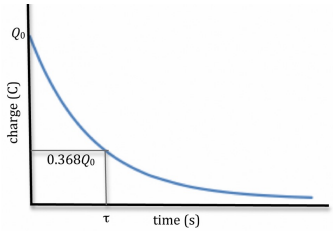
x(t)



x(t)

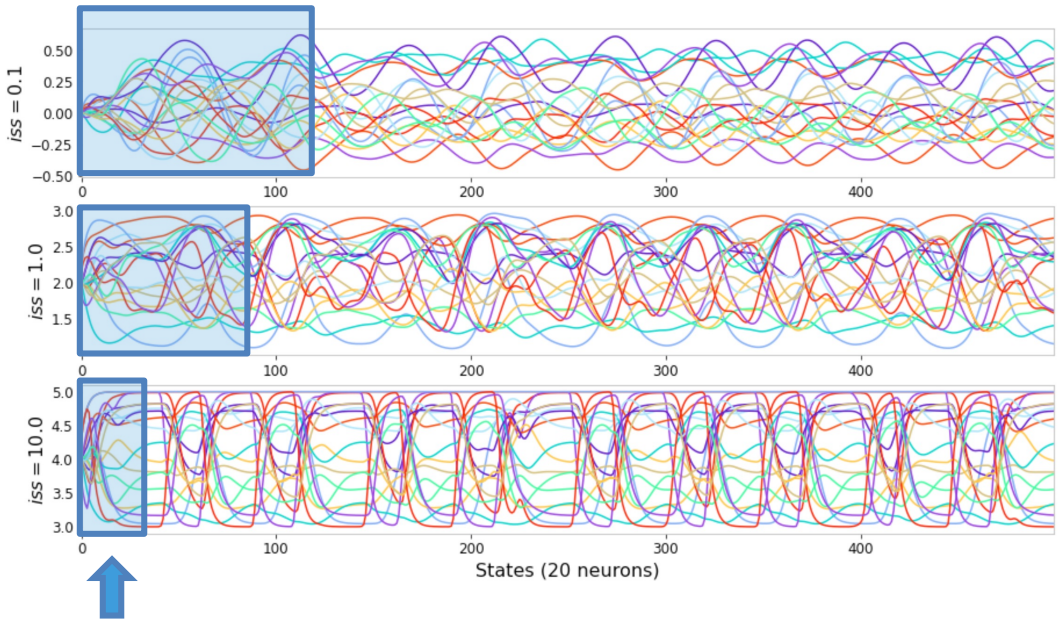






« Warming up »

warmup
ou
warming up

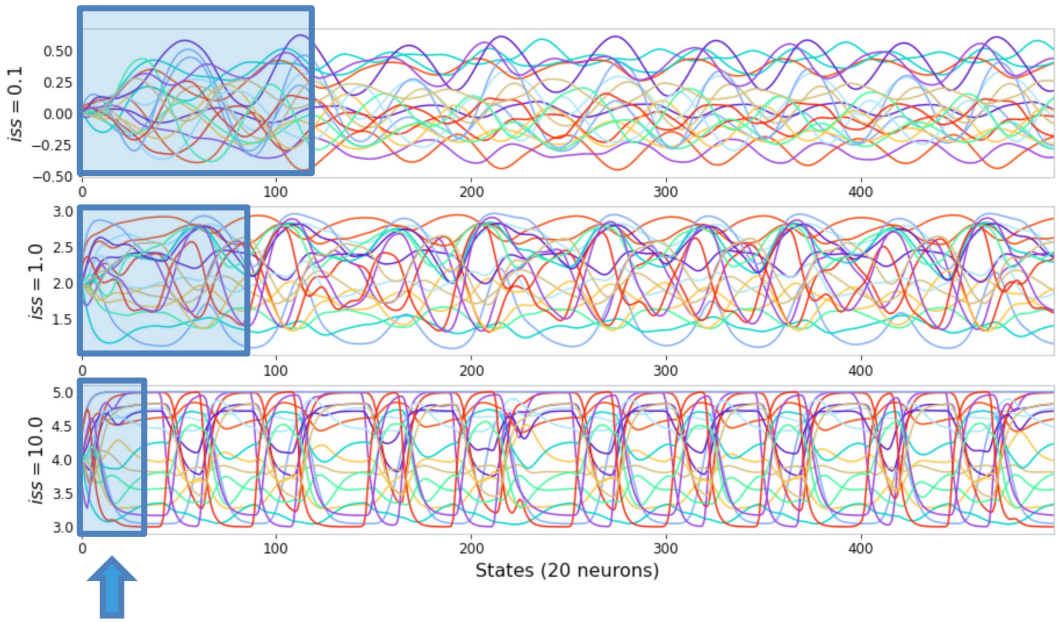


x(t)

- Le *warmup* est le « temps de chauffe » nécessaire pour que les dynamiques soient dans un rythme de « croisière ».
  - = zone bleue sur chaque figure
- Cela correspond au temps nécessaire pour que le réseau ne fasse plus vraiment la différence entre le début ou le milieu de la série temporelle (c-à-d après la zone bleue).
- = si je copie/colle un bout de graphique après le *warmup* vous ne serez pas capable de distinguer s'il s'agit du début/milieu/fin du graphique.

« Warming up »

warmup
ou
warming up



x(t)

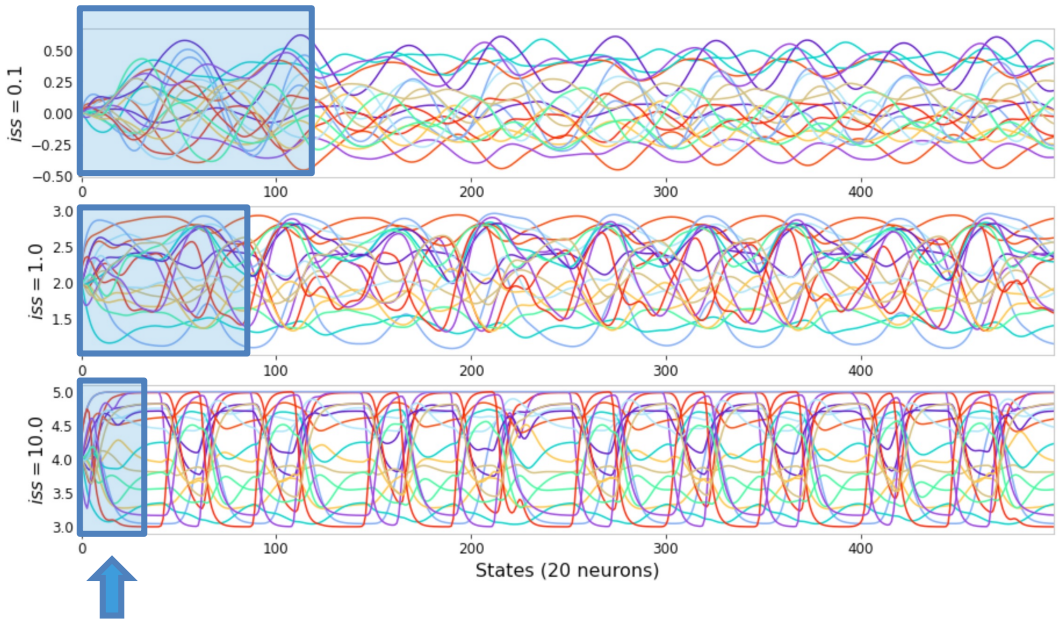
A priori, vous ne voulez pas utilisez ces dynamiques initiales du réservoir qui peuvent perturber l'apprentissage de votre tâche.

→ En conséquence, il faut définir l'hyperparamètre *warmup*

- « à l'œil » par rapport aux activités internes du réservoir
- et/ou en testant par essai/erreur pour quel *warmup* minimal vous avez des performances qui ne changent plus

« Warming up »

warmup
ou
warming up



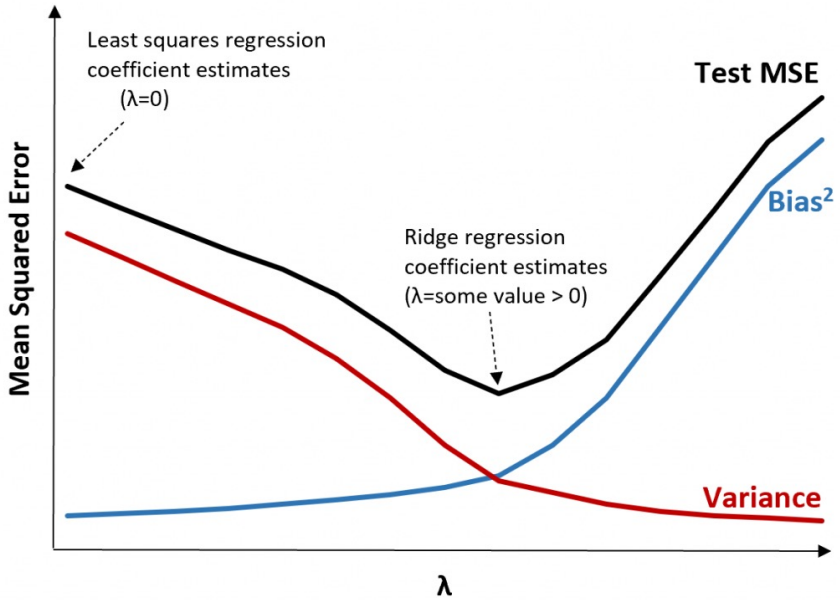
x(t)

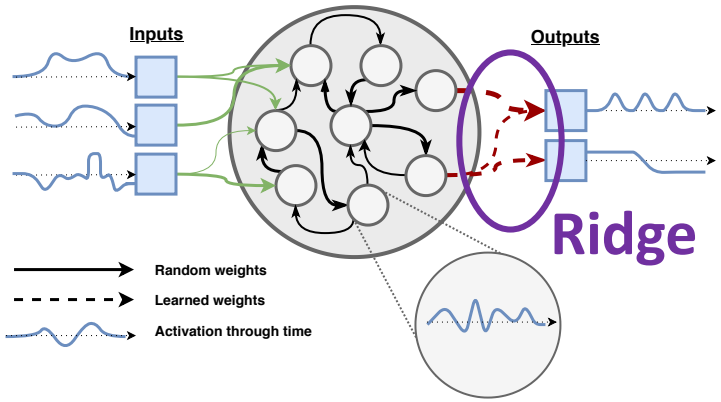
Ce *warmup* / « temps de chauffe » dépend

- des hyperparamètres du réservoir
  - (ici l'input scaling *iss*)
- des entrées / des données que vous donnez au réservoir
  - (par ex. si le taux d'échantillonnage change)

Ridge

= régularisation de la régression linéaire





- Modèle avec variance élevée est très flexible
  - et s'adapte trop bien aux données d'entraînement, y compris le bruit
- Modèle avec biais élevé simplifie trop le problème
  - le modèle est trop simple pour capturer les relations sous-jacentes dans les données

Méthodes pour la recherche d'hyperparamètres

- « Random search » répétés et adpatés
  - Hinaut & Trouvain, ICANN 2021
  - Plus efficace que « grid search »

Which Hype for my New Task? Hints and Random Search for Echo State Networks Hyperparameters

Xavier Hinaut[1,2,3,*][0000−0002−1924−1184] and Nathan Trouvain[1,2,3][0000−0003−2121−7826]

[1] INRIA Bordeaux Sud-Ouest, France.
[2] LaBRI, Bordeaux INP, CNRS, UMR 5800.
[3] Institut des Maladies Neurodégénératives, Université de Bordeaux, CNRS, UMR 5293.
*Corresponding author: xavier.hinaut@inria.fr

Abstract. In learning systems, hyperparameters are parameters that are not learned but need to be set a priori. In Reservoir Computing, there are several parameters that needs to be set a priori depending on the task. Newcomers to Reservoir Computing cannot have a good intuition on which hyperparameters to tune and how to tune them. For instance, beginners often explore the reservoir sparsity, but in practice this parameter is not of high influence on performance for ESNs. Most importantly, many authors keep doing suboptimal hyperparameter searches: using

https://inria.hal.science/hal-03203318

- Algorithmes génétiques avec 400 variables d'entrées
  - Ferte, …, Hinaut, ICML 2024
  - Prédiction hospitalisations COVID à +14 jours

Reservoir Computing for Short High-Dimensional Time Series: an Application to SARS-CoV-2 Hospitalization Forecast

Thomas Ferté, Dan Dutartre, Boris P Hejblum, Romain Griffier, Vianney Jouhet, Rodolphe Thiébaut, Pierrick Legrand, Xavier Hinaut

Published: 02 May 2024, Last Modified: 25 Jun 2024 ICML 2024 Poster Everyone Revisions BibTeX
CC BY-NC 4.0

Abstract:
In this work, we aimed at forecasting the number of SARS-CoV-2 hospitalized patients at 14 days to help anticipate the bed requirements of a large scale hospital using public data and electronic health records data. Previous attempts led to mitigated performance in this high-dimension setting; we introduce a novel approach to time series forecasting by providing an alternative to conventional methods to deal with high number of potential features of interest (409 predictors). We integrate Reservoir Computing (RC) with feature selection using a genetic algorithm (GA) to gather optimal non-linear combinations of inputs to improve prediction in sample-efficient context. We illustrate that the RC-GA combination exhibits excellent performance in forecasting SARS-CoV-2 hospitalizations. This approach outperformed the use of RC alone and other conventional methods: LSTM, Transformers, Elastic-Net, XGBoost. Notably, this work marks the pioneering use of RC (along with GA) in the realm of short and high-dimensional time series, positioning it as a competitive and innovative approach in comparison to standard methods.

https://openreview.net/forum?id=CY0IFwD4qx

En pratique ? (Hypster-paramètres)

- Les hyperparamètres importants à régler
  - Nombre de neurones
  - Input scaling
  - Spectral radius (= rayon spectral)
  - (Feedback scaling si présence de feedback)
  - Leak-rate (= taux de fuite = 1 / constance de temps)
  - Ridge (régularisation de la régression)
  - Warm-up (= « temps de chauffe »)

→ Voir le tutoriel 4 de ReservoirPy sur GitHub

reservoirpy / tutorials / 4-Understand_and_optimize_hyperparameters.ipynb

VirgileBoraud and PAUL-BERNARD Correction of some minor mistakes on the tutorials (#161) 78b89al

2.29 MB

Understand and optimize ESN hyperparameters

This chapter gives very basic clues on how to interpret the action of some first importance hyperparameters of an networks.

It also present a very basic example of optimization using hyperopt and reservoirpy.hyper tools.

Understand ESN hyperparameters

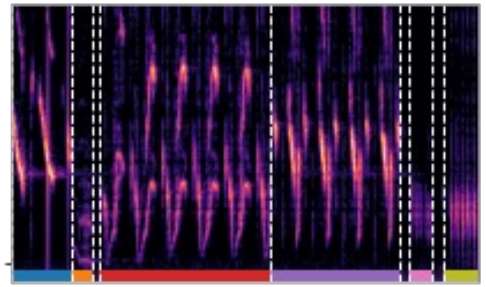
In [1]:

```
UNITS = 100                # - number of neurons
LEAK_RATE = 0.3            # - leaking rate
SPECTRAL_RADIUS = 1.25     # - spectral radius of W
INPUT_SCALING = 1.0        # - input scaling
RC_CONNECTIVITY = 0.1      # - density of reservoir internal matrix
INPUT_CONNECTIVITY = 0.2   # and of reservoir input matrix
REGULARIZATION = 1e-8      # - regularization coefficient for ridge regression
SEED = 1234                # for reproductibility
```

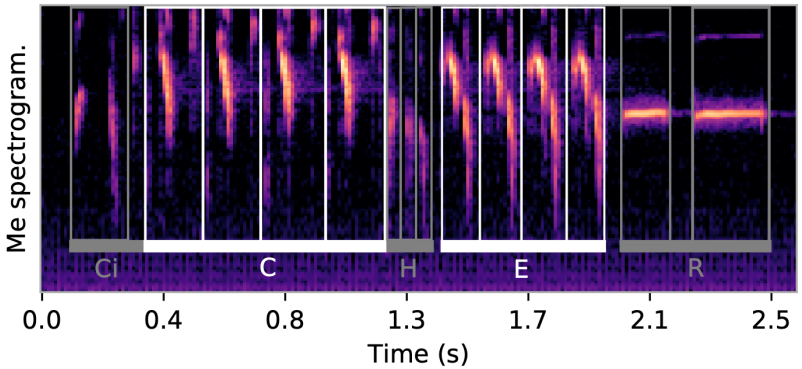
https://github.com/reservoirpy/reservoirpy/blob/master/tutorials/4-Understand_and_optimize_hyperparameters.ipynb

EXEMPLE D'APPLICATION

DISCRIMINATION CHANT CANARIS
RESERVOIR VS. LSTM

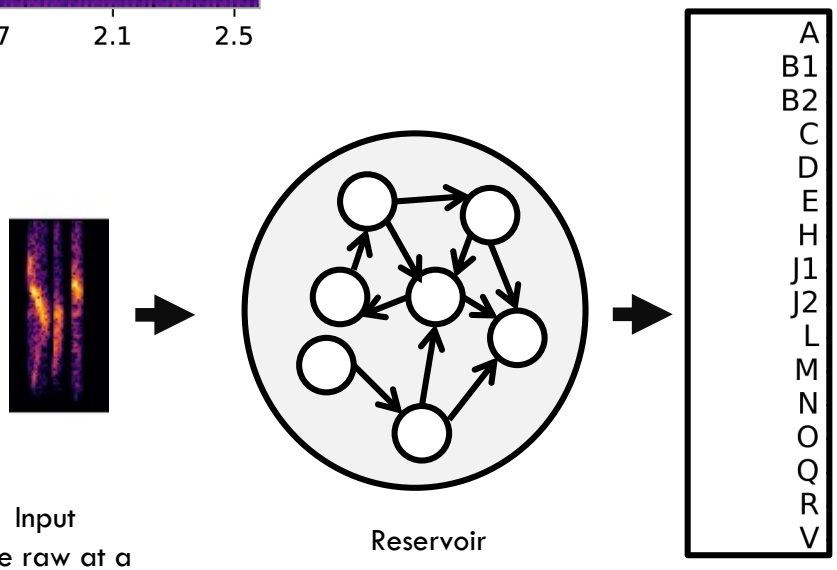


Discrimination de phrases de canaris



Input preproc. as MFCCs

RNN training/testing



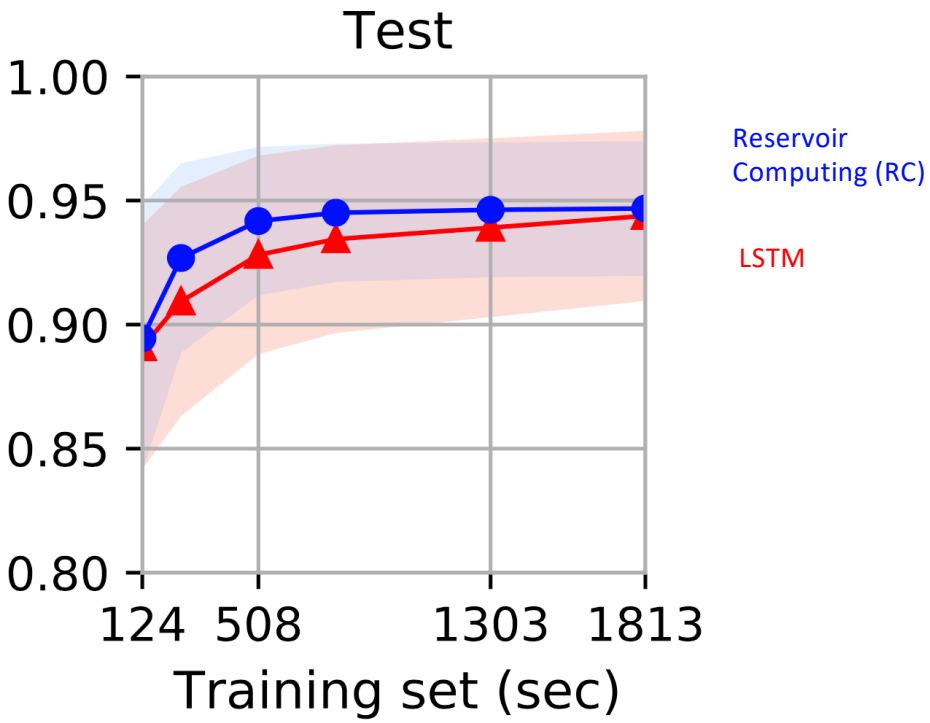
Input
(one raw at a
time)

Reservoir
or
LSTM

Output

Discrimination de phrases de canaris

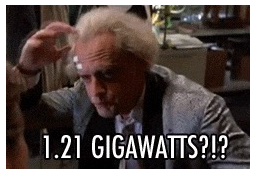
Taux de précision vs. taille des données d'entrainement

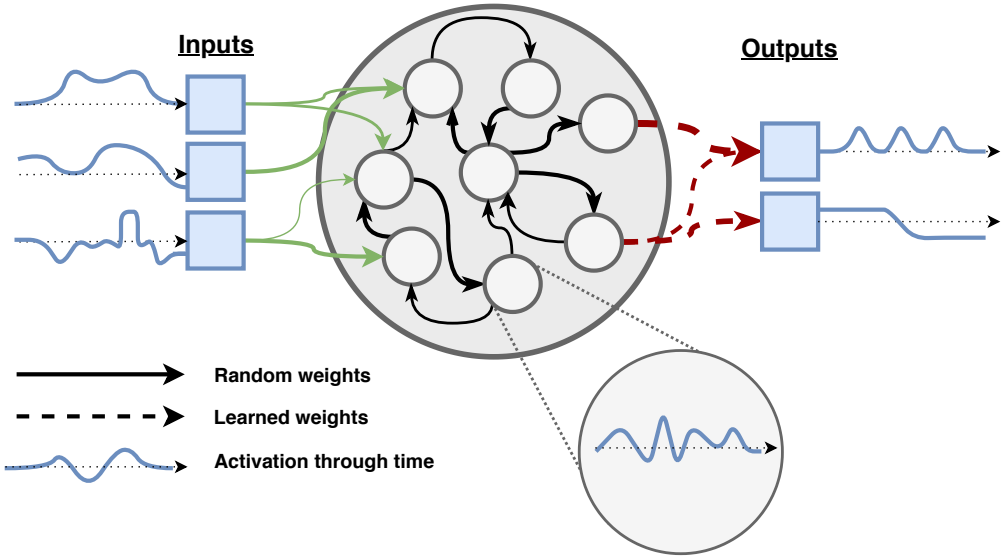


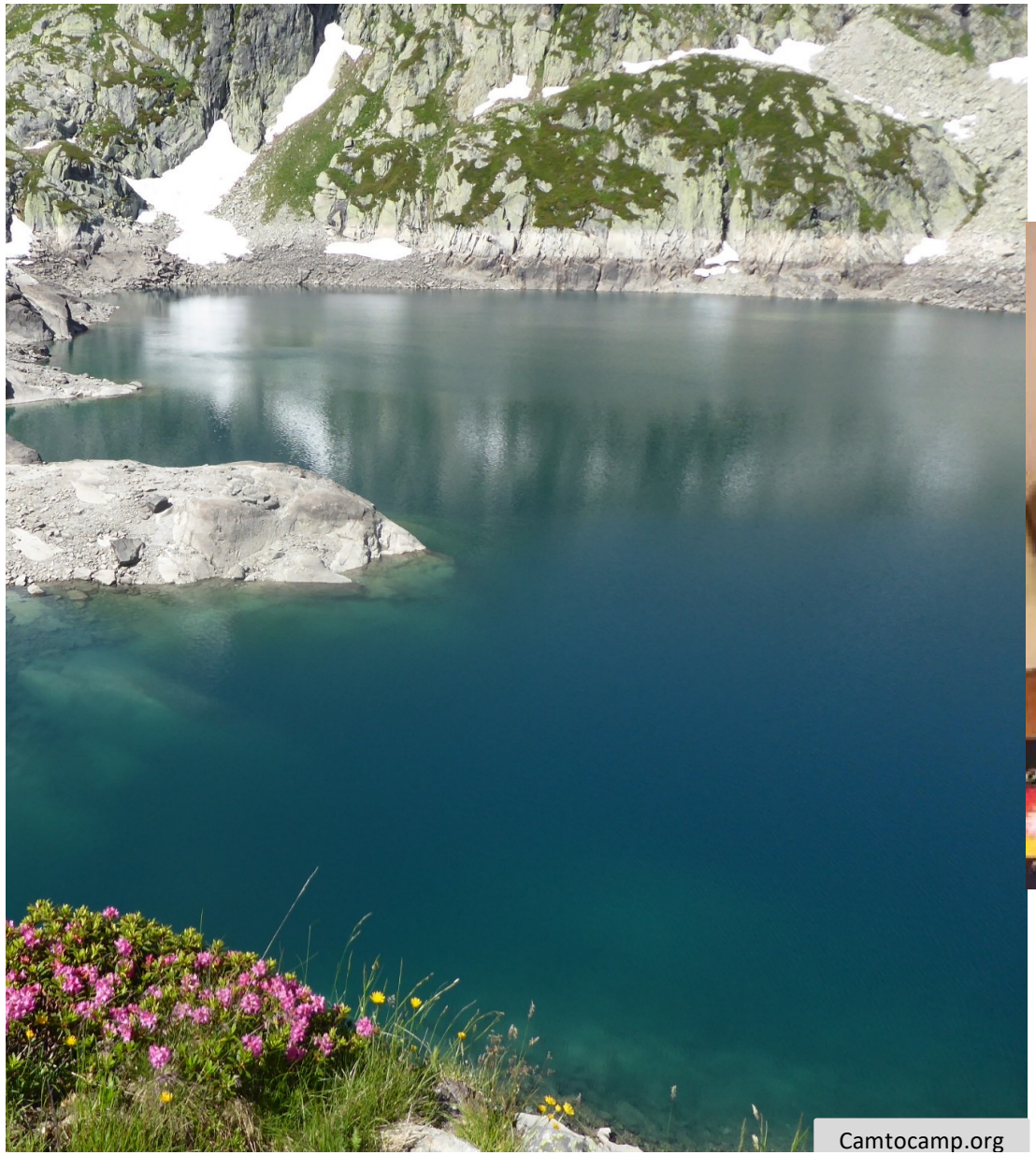
Temps moyen pour l'entrainement sur CPU (10-fold CV)

Model	Temps d'entrainement en sec.
LSTM	2930 ± 222
RC	35 ± 1

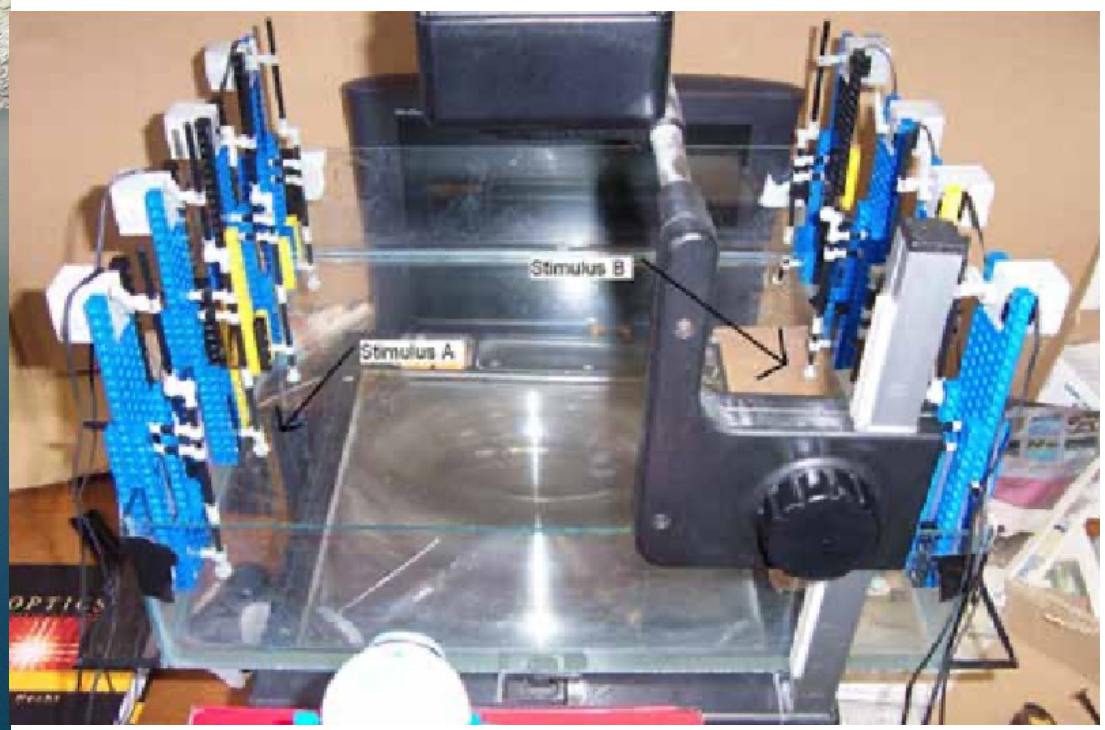
→ 40 min !




Inputs
Outputs
Random weights
Learned weights
Activation through time

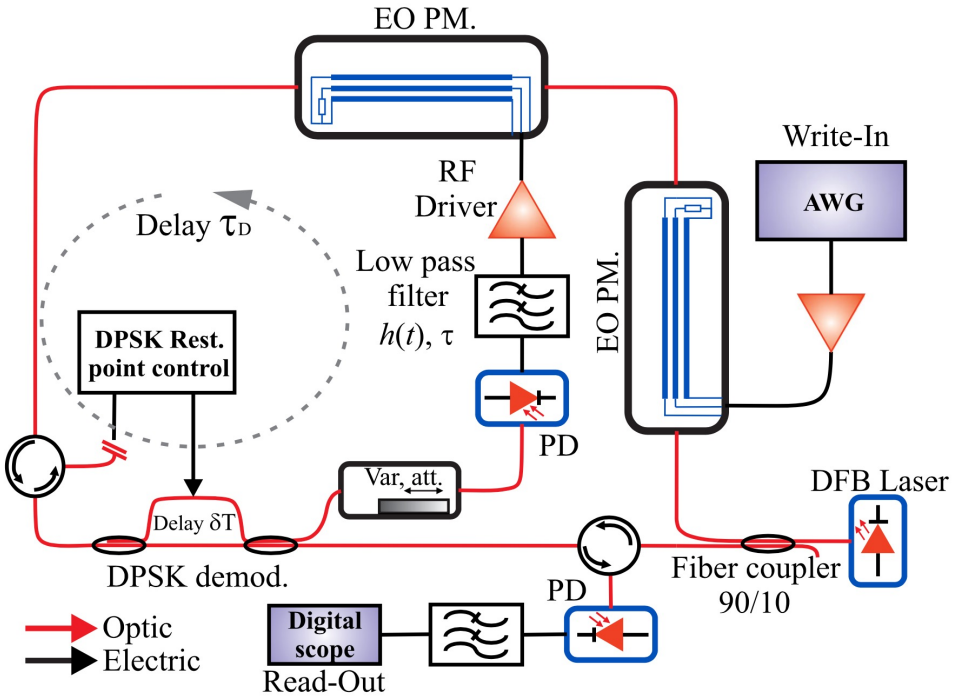


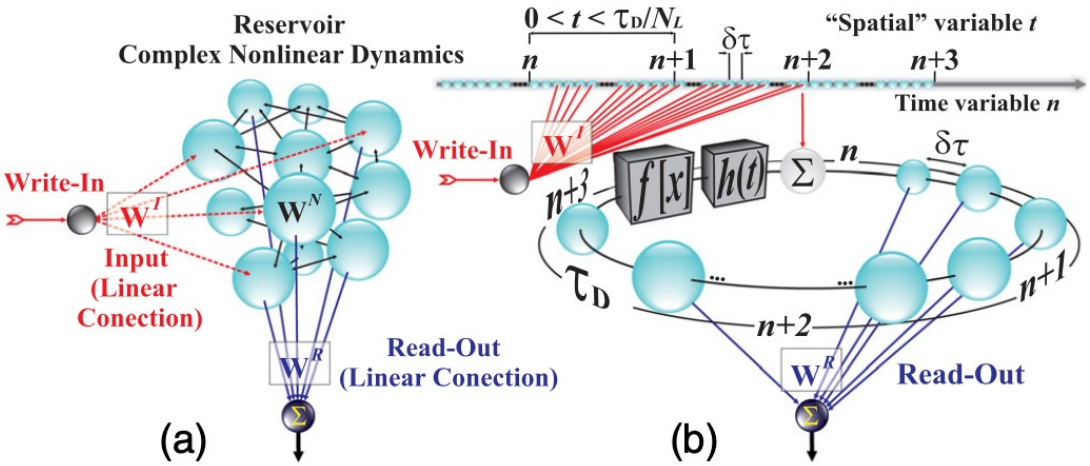
Camtocamp.org



Fernando & Sojakka, 2003

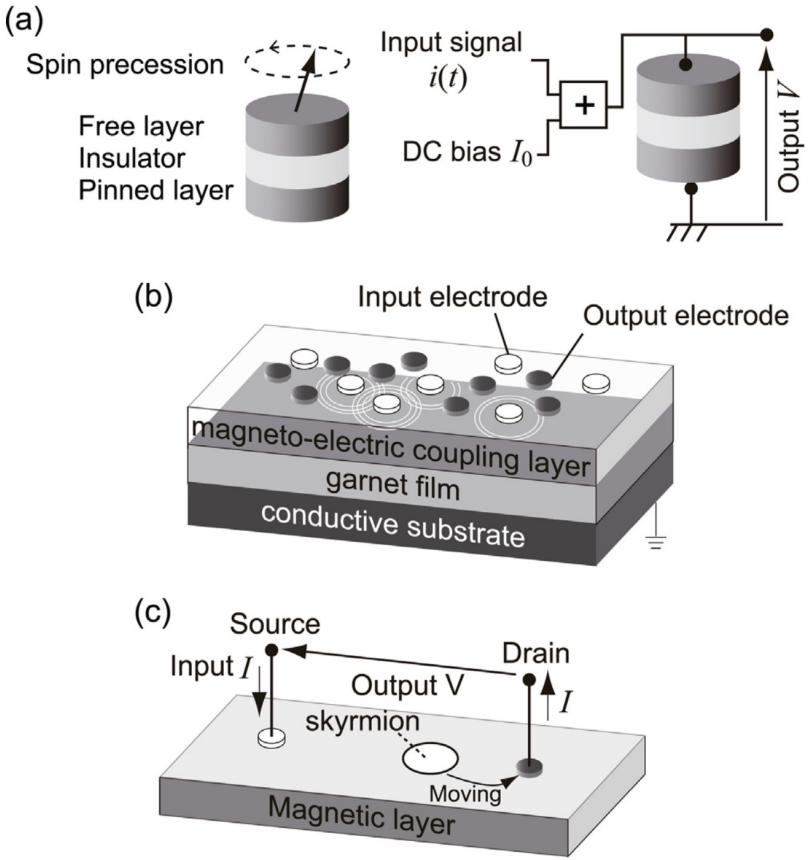
Reservoir Computing Optique



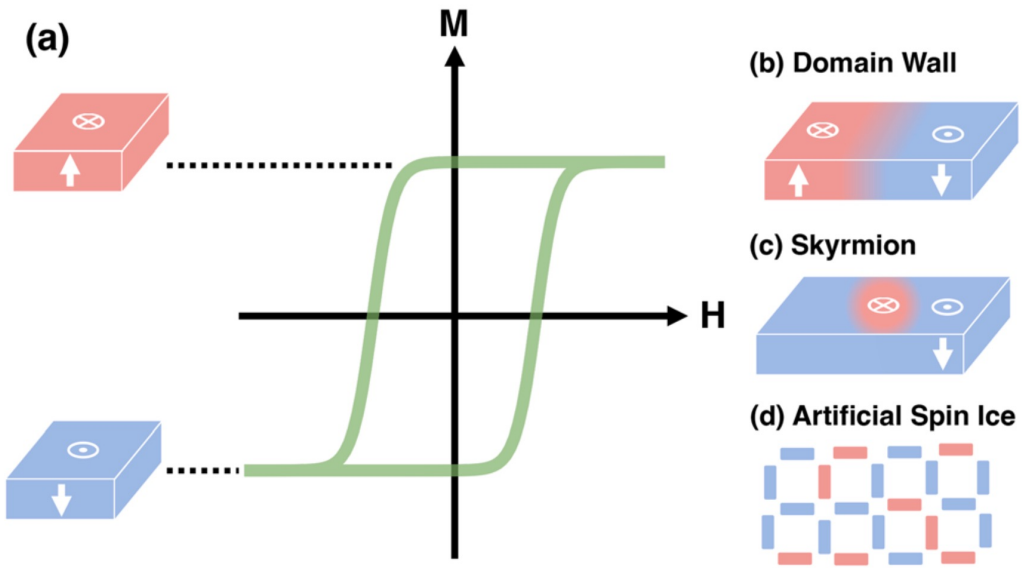


Larger et al. 2017

Reservoir Computing Physiquen (*skyrmions*)



Tanaka et al. 2019



Allwood et al. 2023

Choisissez le bon outil pour chaque problème

Si on a un marteau tous les problèmes sont des clous !

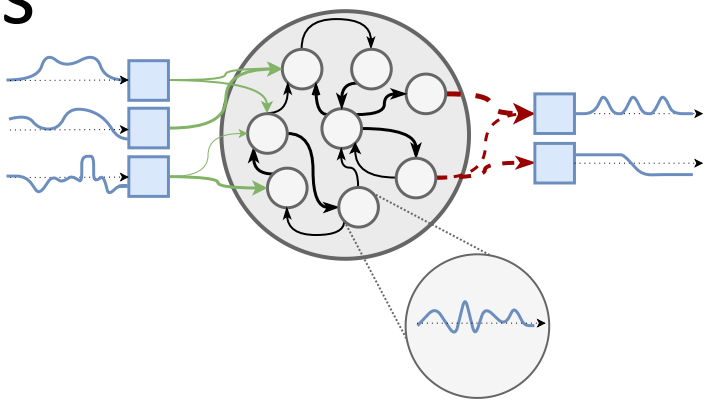
En Résumé

- Le Reservoir Computing permet de
  - Prédire / Générer / Classifier / Simuler
  - Entraînement *online* et *offline*
  - → données temporelles (et non-temporelles)
  - Se combine avec CNN, RL, …

- ReservoirPy propose une IA frugale
  - Bonnes performances même avec peu de données nécéssaires
  - Economie de calculs / temps
  - → Economie d'énergie !

- Transférable sur différents types de *hardware*
  - systèmes physiques optiques, magnétiques, biologiques, etc.

Les reseaux aléatoires peuvent
générer des dynamiques fascinantes

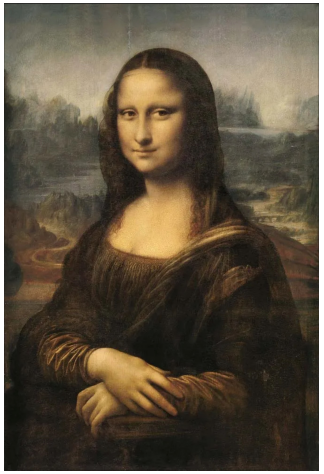


Le reservoir computing exploite
les dynamiques naturelles d'un réseau aléatoire

Les reseaux aléatoires peuvent
générer des dynamiques fascinantes

– Ça ne coute pas cher !

– Et ça permet d'avoir une grande palette



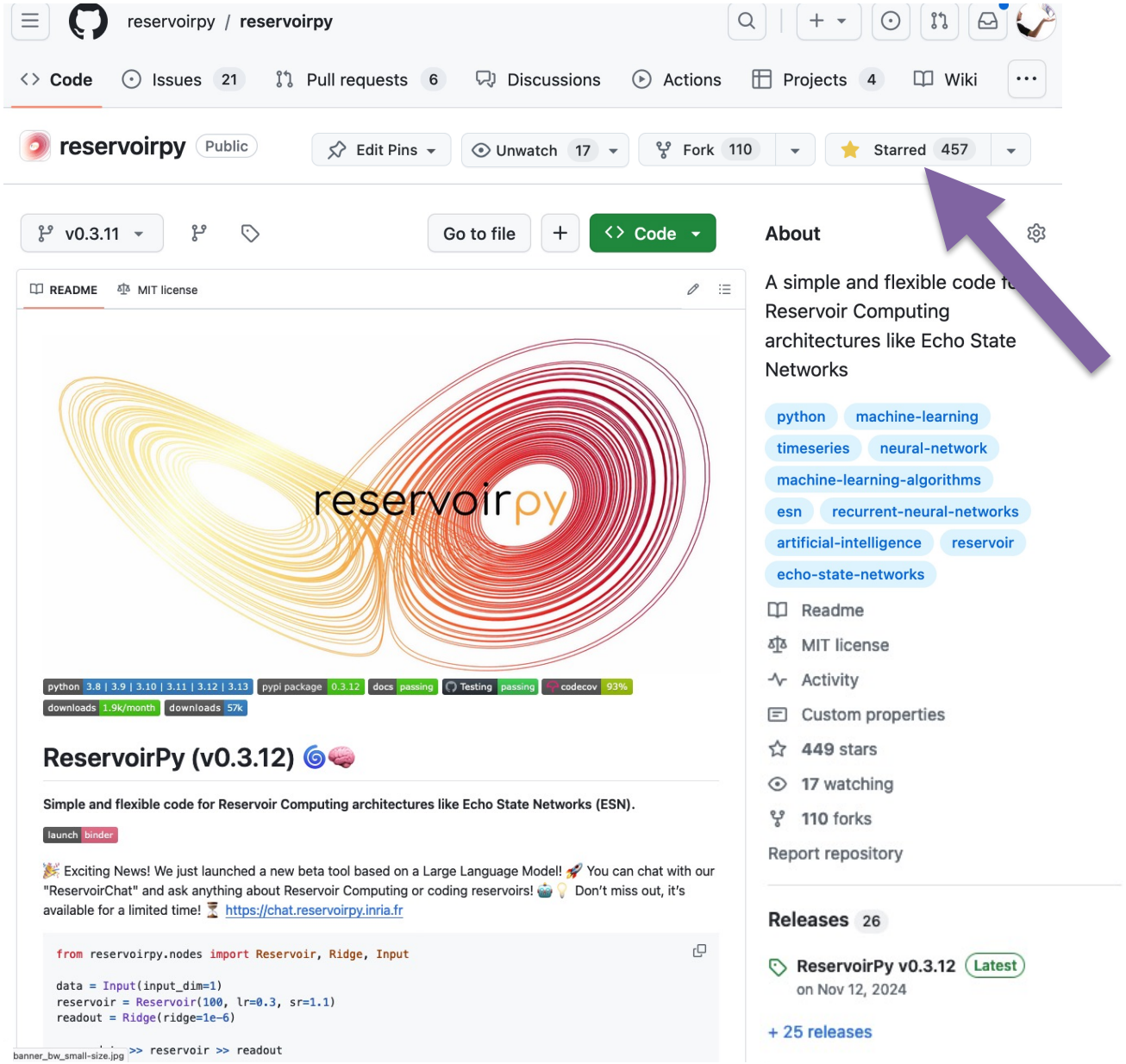
Le reservoir computing exploite
les dynamiques naturelles d'un réseau aléatoire

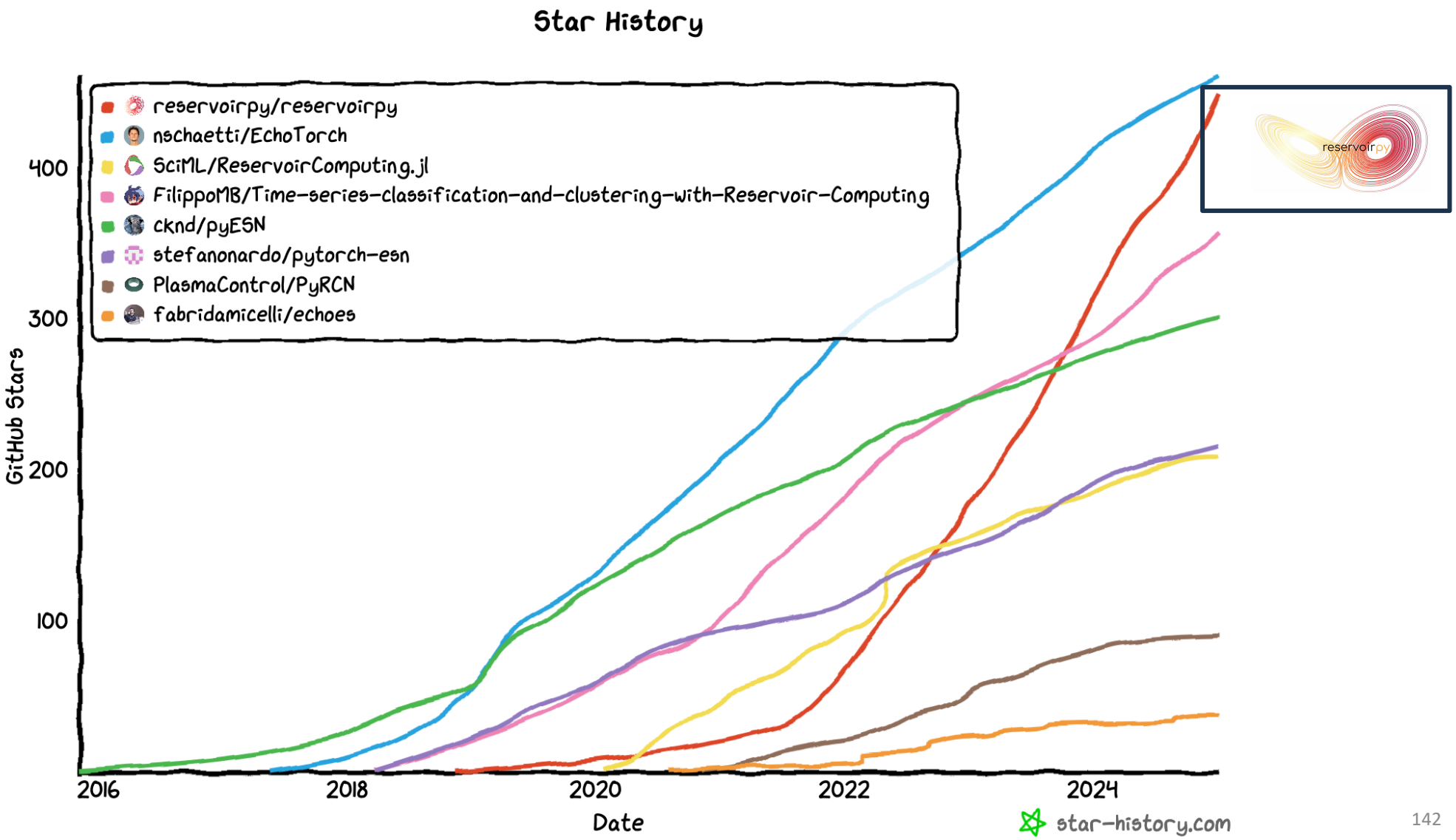
C'est juste une question de bonnes dynamiques !

Testez ReservoirPy sur l'application qui vous intéresse et dites-moi !

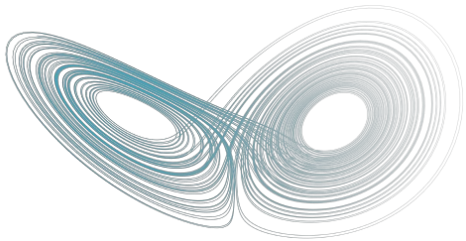
Je suis curieux de savoir si ça marche !

github.com/reservoirpy/reservoirpy

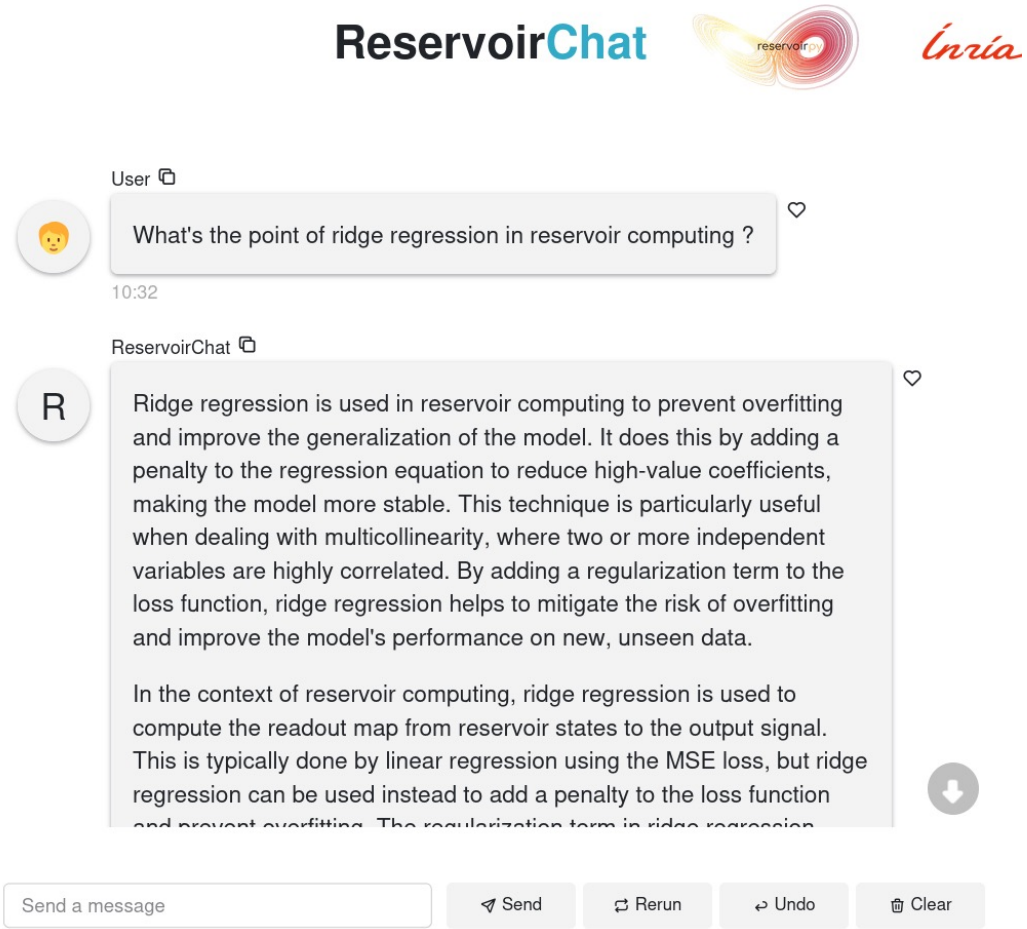



Star History
reservoirpy/reservoirpy
nschaetti/EchoTorch
SciML/ReservoirComputing.jl
FilippoMB/Time-series-classification-and-clustering-with-Reservoir-Computing
cknd/pyESN
stefanonardo/pytorch-esn
PlasmaControl/PyRCN
fabridamicelli/echoes
GitHub Stars
100
200
300
400
2016
2018
2020
2022
2024
Date
star-history.com
reservoirpy

ReservoirChat



chat.reservoirpy.inria.fr



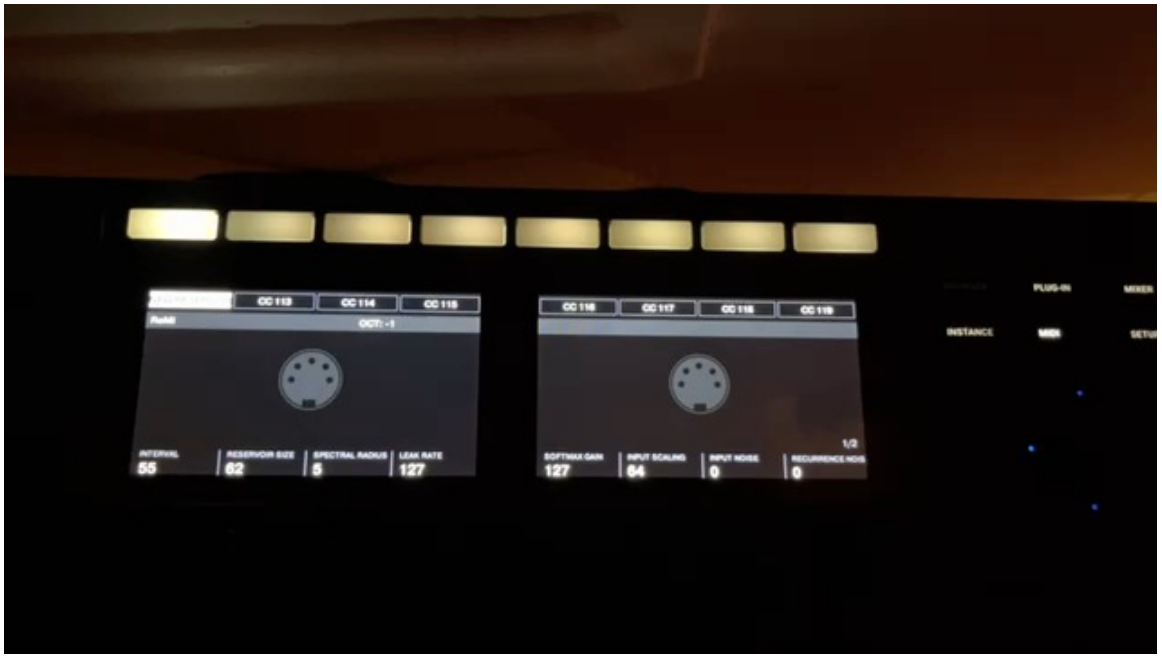
Merci pour votre attention !

xavier.hinaut@inria.fr
@neuronalX
@ hinaut @reservoirpy

sites.google.com/view/hack1robo
https://www.instagram.com/p/C2SkVGKMchu

github.com/reservoirpy/reservoirpy

chat.reservoirpy.inria.fr



Suite pour la partie TP

Installez ReservoirPy

- pip install reservoirpy[hyper]
- (pour installer les packages pour l'optimisation d'hyperparamètres en plus)

github.com/reservoirpy/reservoirpy

Documentation ReservoirPy

- ReadTheDocs

reservoirpy.readthedocs.io

Téléchargez le tutoriel pour le TP

→ tester ReservoirPy sur votre machine

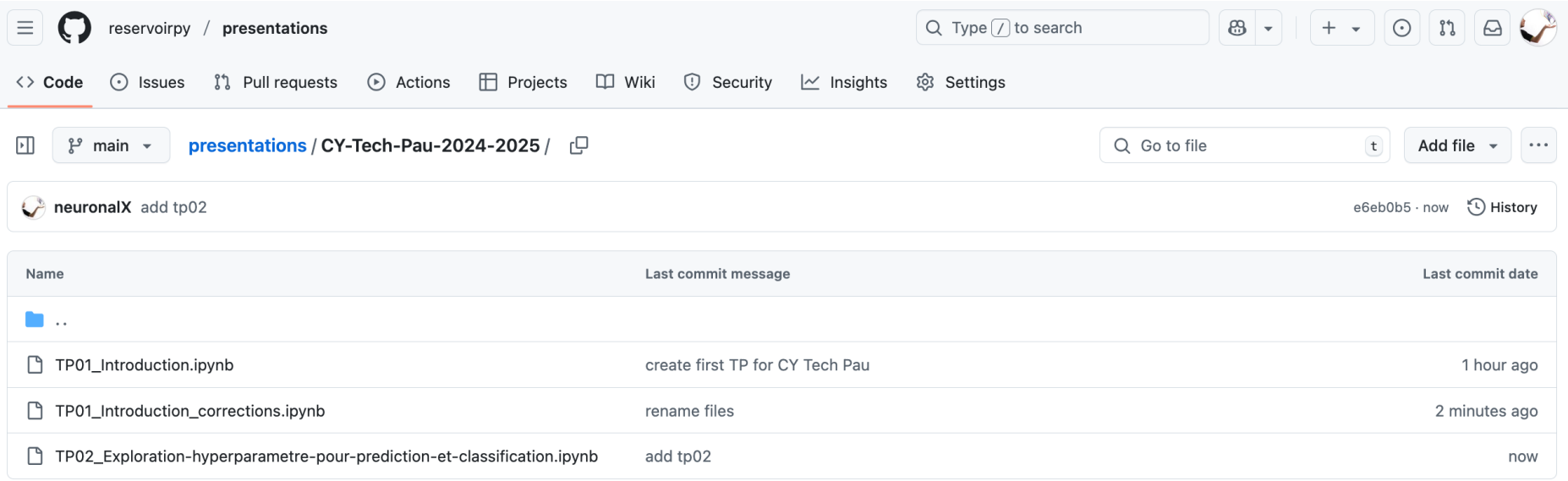
https://github.com/reservoirpy/presentations
→ Dossier "CY-Tech-Pau-2024-2025"

github.com/reservoirpy/presentations/tree/main/CY-Tech-Pau-2024-2025

Suite pour la partie TP

github.com/reservoirpy/presentations/tree/main/CY-Tech-Pau-2024-2025

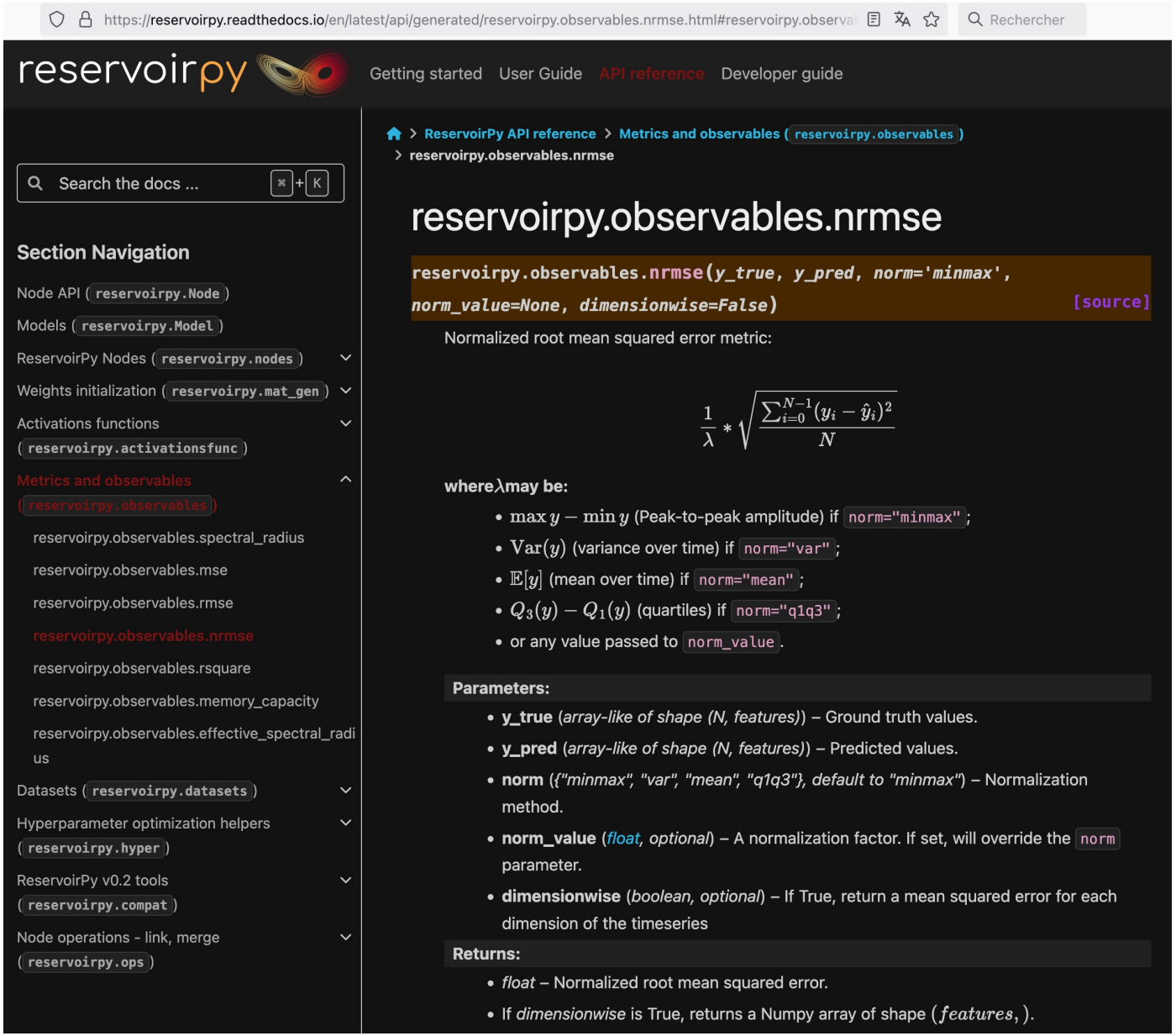
Vous devriez arriver sur une page qui ressemble à ça :



Aidez-vous de la documentation API

Par ex., vous avez plusieurs méthodes disponibles dans « observables » qui vous seront utiles

https://reservoirpy.readthedocs.io/en/latest/api/generated/reservoirpy.observables.nrmse.html#reservoirpy.observables.nrmse

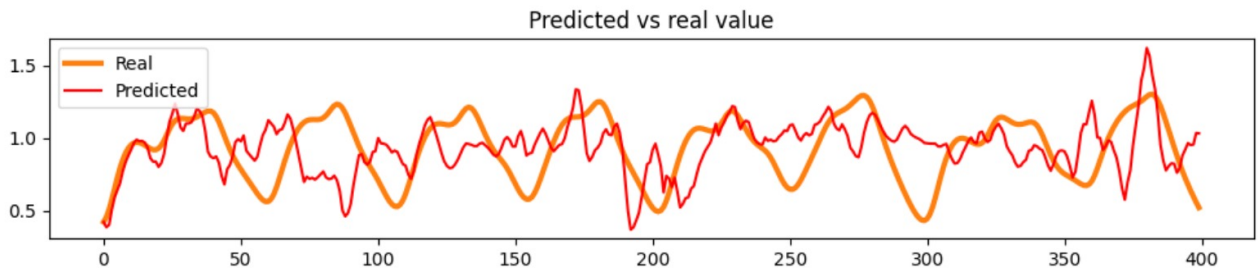


Astuces pour la partie TP: difference ESN vs. modèle perso

Attention ! Observer l'activité d'un modèle que vous définissez vous-même et observer l'activité d'un modèle fait avec un nœud ESN ne se fait pas de la même façon !

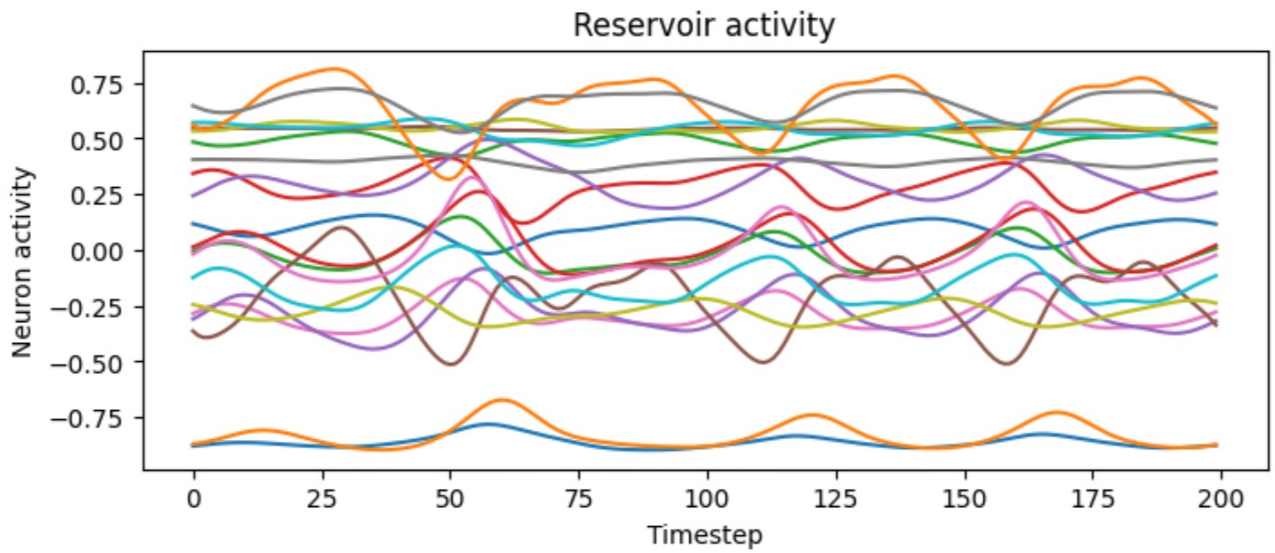
```
# A vous de jouer !
# Définissez et entrainez ci-dessous un modèle d'ESN en changeant les paramètres précédents

modelbis = ESN(
    units=100, # neurones dans le réservoir
    lr=0.3, # taux de fuite
    sr=10, # rayon spectral de la matrice de poids
    ridge=1e-8, # paramètre de régularisation
)


# Ajustement du modèle
modelbis.fit(x_train, y_train)

# Test du modèle
y_pred = modelbis.run(x_test)
```

```
# Observez les changements sur les valeurs prédites
plot_figure(y_test, y_pred)
```



Astuces pour la partie TP: difference ESN vs. modèle perso

```
# Observez les changements sur l'activité des neurones

activity = reservoir.run(x_test)

plt.figure(figsize=(8, 3))
plt.plot(activity[:200, :20]) # 200 pas de temps, 20 neurones visualisés
plt.title("Reservoir activity")
plt.xlabel("Timestep")
plt.ylabel("Neuron activity")
plt.show()
```



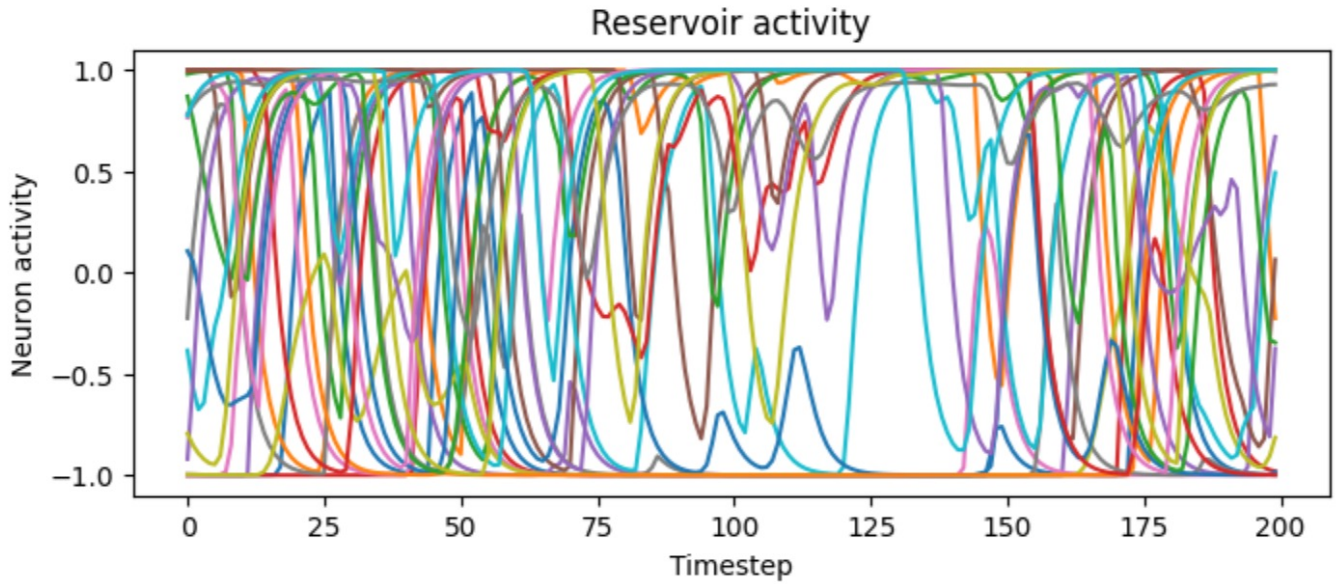
Astuces pour la partie TP: difference ESN vs. modèle perso

```python
# Ah mais c'est étrange !
# Cela ressemble exactement à l'activité que l'on avait avant ! Pourquoi ?
# -> Car nous avons regardé l'activité du 1er reservoir que nous avions créé
#    et pas l'ESN que nous utilisons actuellement !
# Pour analyser l'activité d'un ESN il faut récupérer les activités d'une autre façon :
activitybis = modelbis.run(x_test,
    return_states=["reservoir"]) # On ajoute la liste des noeuds dont on veut récupérer l'activité.

# /!\ Attention, ce que l'on récupère lorsque l'on spécifie "return_states" est un dictionnaire.
print(activitybis)

# Il faut donc récupérer l'activité de la façon suivante:
## activitybis['reservoir']
```

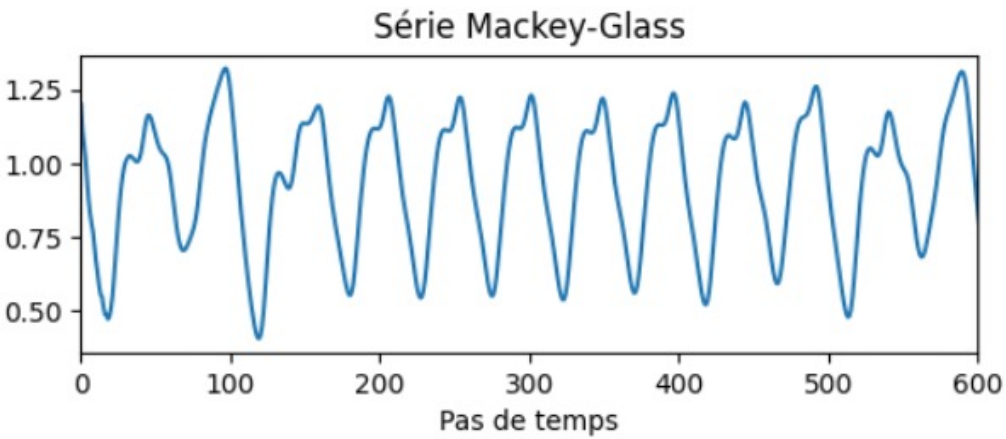
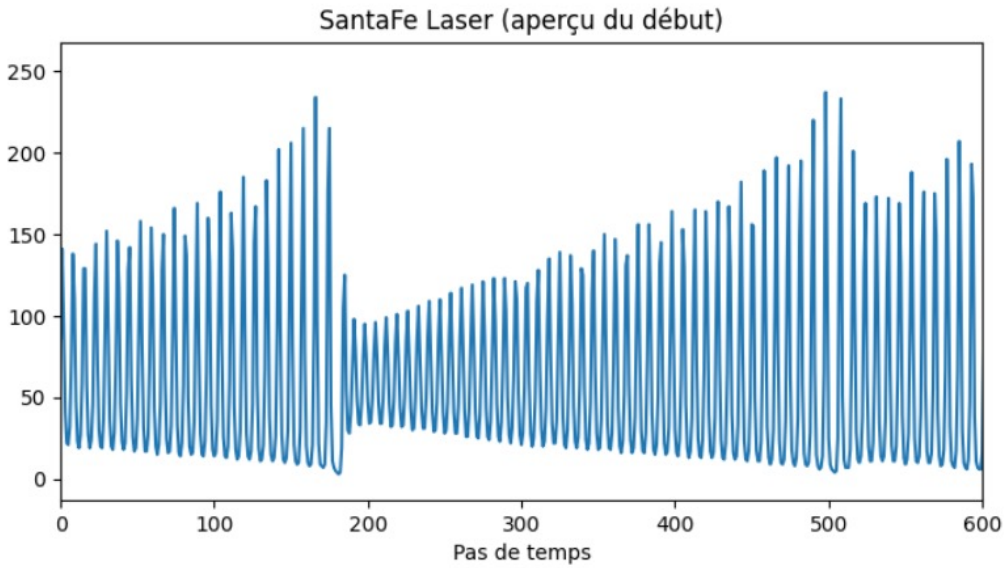
```
{'reservoir': array([[-0.99951318,  0.99862913,  0.86969742, ...,  0.99090729,
         0.99995465,  0.99953305],
       [-0.99965923,  0.99904   ,  0.75188303, ...,  0.99363316,
         0.99996795,  0.99907841],
       [-0.99976146,  0.99932762,  0.61087059, ...,  0.99554181,
         0.99997704,  0.86182414],
       ...,
       [-0.96744104, -0.03334604,  0.89825405, ..., -0.83357127,
        -0.45955163, -0.9704059 ],
       [-0.97634478, -0.323341  ,  0.92874884, ..., -0.88349989,
        -0.60656752, -0.97928413],
       [-0.96882611, -0.52633859,  0.95003025, ..., -0.91844992,
        -0.41795876, -0.98549889]])}
```

Astuces pour la partie TP: difference ESN vs. modèle perso

```
activitybis = modelbis.run(x_test,
    return_states=["reservoir"])

plt.figure(figsize=(8, 3))
plt.plot(activitybis['reservoir'][:200, :20]) # 200 pas de temps, 20 neurones visualisés
plt.title("Reservoir activity")
plt.xlabel("Timestep")
plt.ylabel("Neuron activity")
plt.show()
```




```
# Et effectivement là on obtient bien un autre type d'activité !
```

Astuces pour la partie TP: regardez l'échelle des vos données

Regardes les valeurs en ordonnées (les « y »), ce n'est pas la même échelle …
donc dans un cas on va être obligé d'adapter (2 solutions possibles) ☺