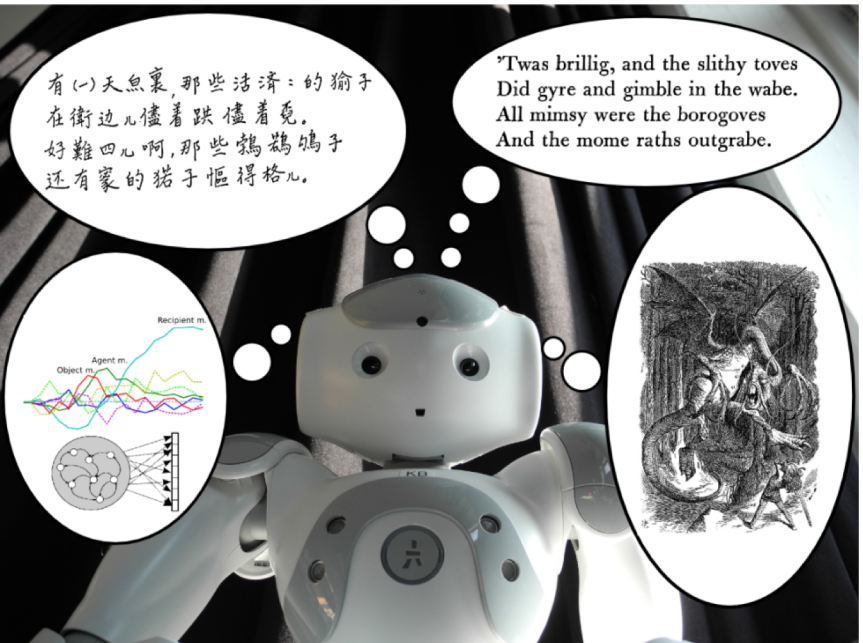
Atelier Machine Learning

- Arnaud Revel: Machines à Vecteurs Support
- Xavier Hinaut: Réseaux de Neurones Récurrents
- Nathan Trouvain: Notebook sur le Reservoir Computing
  - Notebook Python pour la partie tutoriel à télécharger: https://github.com/reservoirpy/R42021-reservoir-computing-tutorial

Réseaux de Neurones Récurrents (RNN)



Xavier HINAUT

Chercheur Inria informatics mathematics

Équipe Mnemosyne

xavier.hinaut@inria.fr

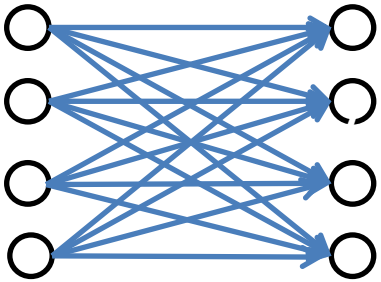
@neuronalX

github.com/neuronalX

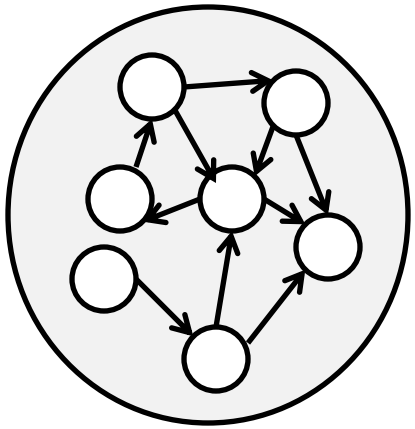
LaBRI

imn institut des maladies neurodégénératives

2 types de réseaux



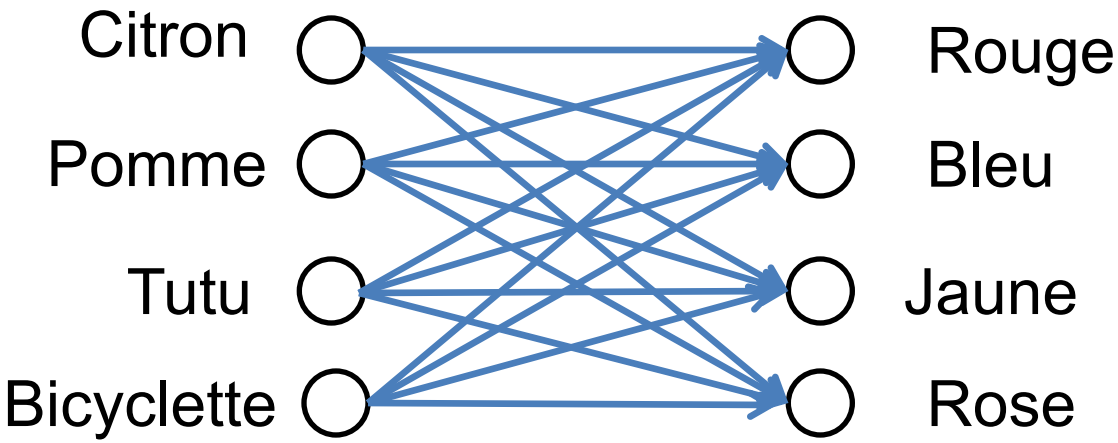
- Réseau de neurones en couches
  - Connexions entre neurones de couches différentes
  - Associations



- Réseau de neurones récurrent
  - Mémoire dynamique
  - Représentation du contexte temporel

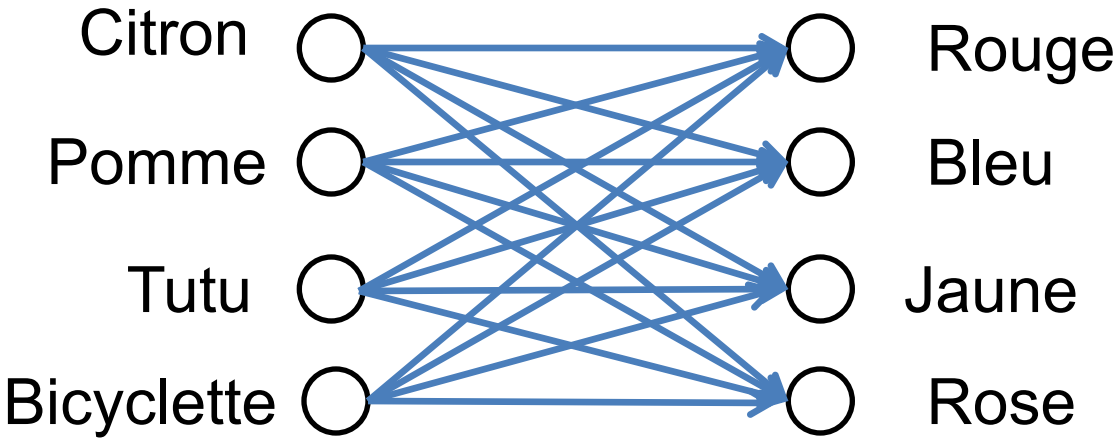
Neurone ⟶ Axone (connexion entre deux neurones)

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 »

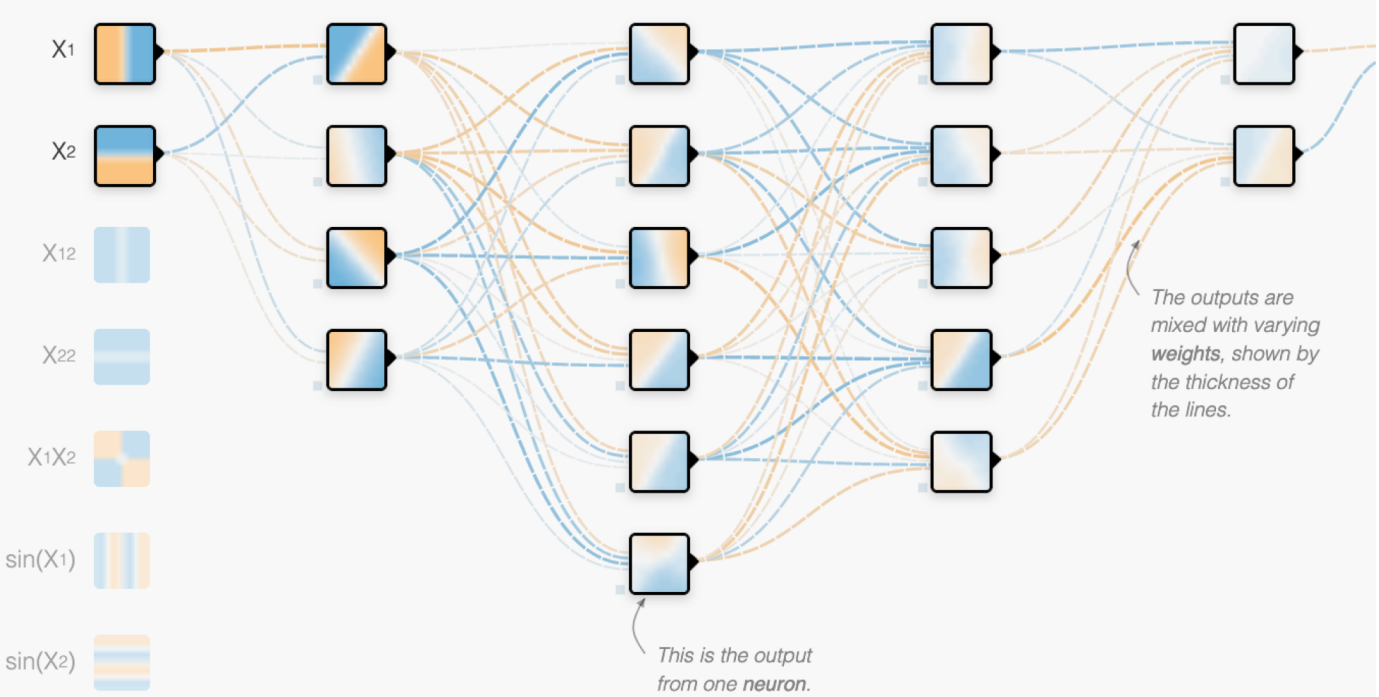
Un réseau de neurones en couches



- Loi de Hebb (1949)
  - lorsque 2 neurones sont actifs en même temps leurs connexions sont renforcés
    - « Neurons that fire together, wire together »
  - Permet de faire des associations (ex : tutu – rose)

Un réseau de neurones en couches

- Avec l'algorithme de rétro-propagation du gradient, on peut apprendre avec un empilement de couches



playground.tensorflow.org

Pourquoi utiliser des réseaux récurrents ?

« Je déteste les lundis »

« J'aime la prose »

« La bicyclette est bleue »

« Comme c'est bizarre ! »

◯ Mme Martin

◯ Jean

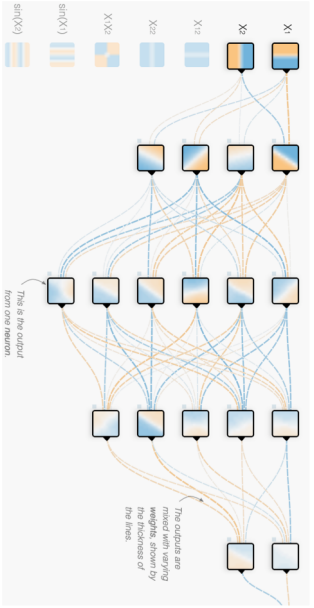
◯ Garfield

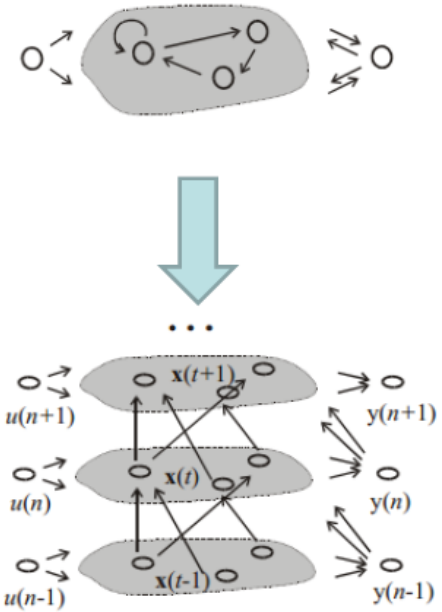
◯ Mr Jourdain

- Comment associer des phrases avec les personnes associées ?
  - Les mots sont dits les uns à la suite des autres
  - Il faut donc une mémoire de l'ordre des mots

Comment apprendre dans un réseau récurrent ?

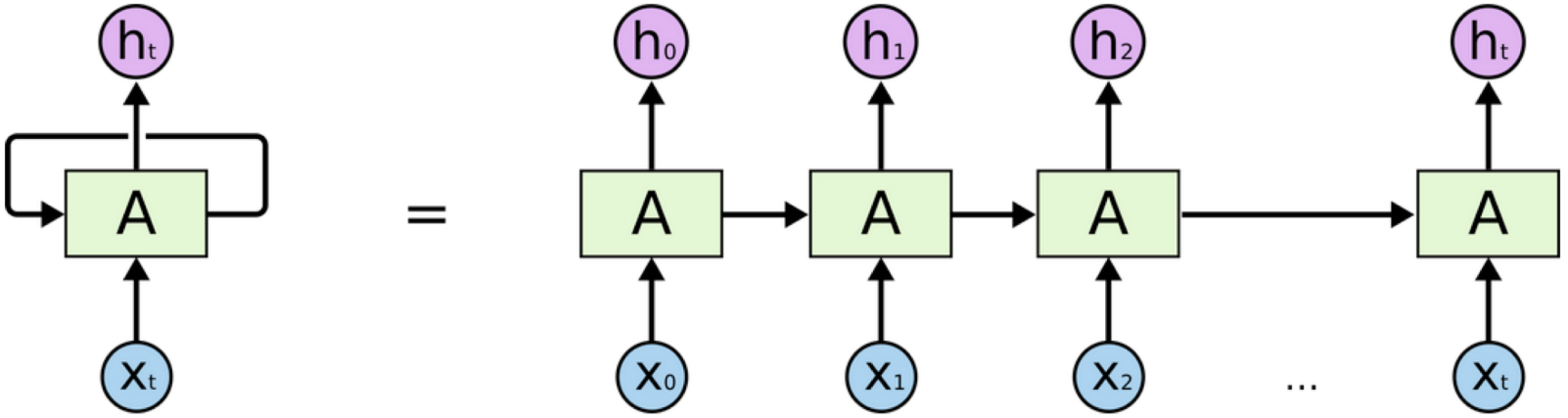
- Si on veut appliquer l'algorithme de rétro-propagation du gradient à un réseau récurrent,
- cela implique de « virtualiser » le temps
- et donc de dupliquer le réseau à chaque pas de temps
- afin appliquer l'algo. comme si c'était un réseau en couches





Réseau de neurone recurrent “déplié”

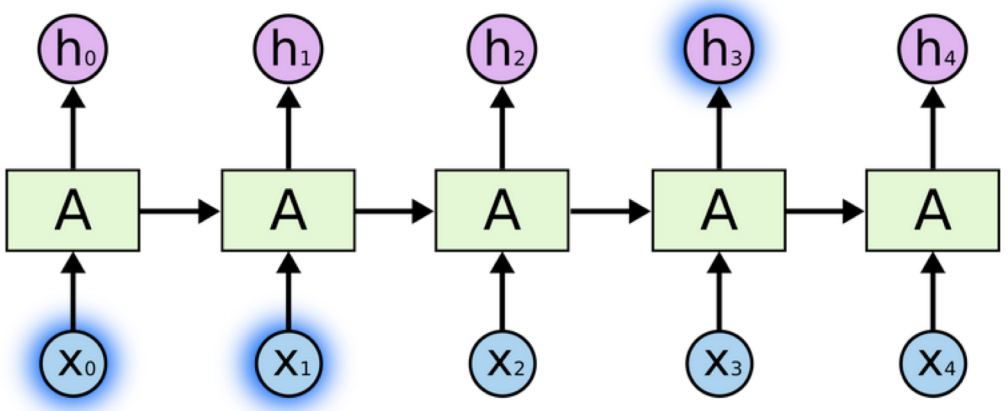
- Un réseau de neurone “déroulé” (*unrolled*) dans le temps



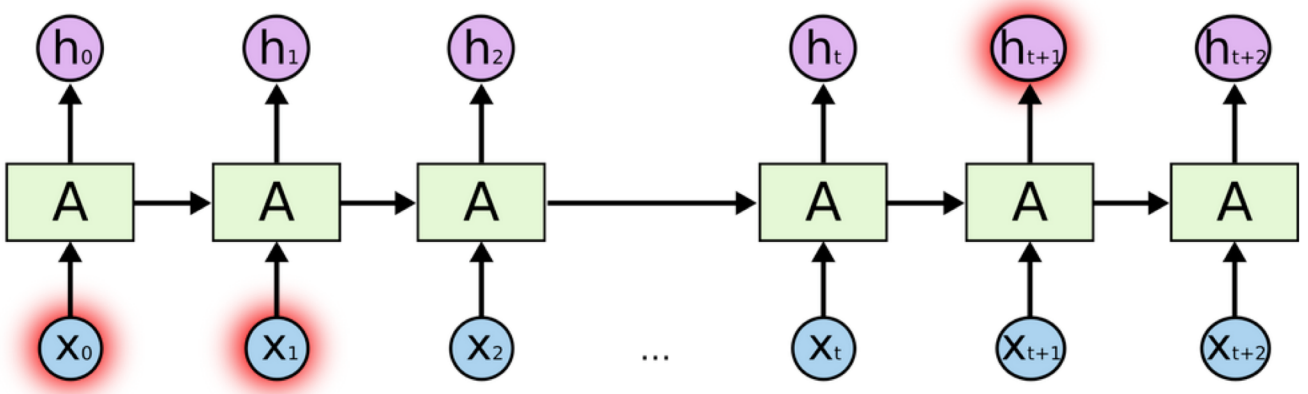
An unrolled recurrent neural network.

How to Deal with Short or Long Time Dependencies?

Short



Long



Plus loin que les RNNs classiques

- Pour pallier au problème de mémoire à long terme dans les RNNs
  - Tani et al. on utilisé les MTRNN (multi-time scale RNN). Plos Comp Bio 2008

Use RNN with Multiple Time-Scales

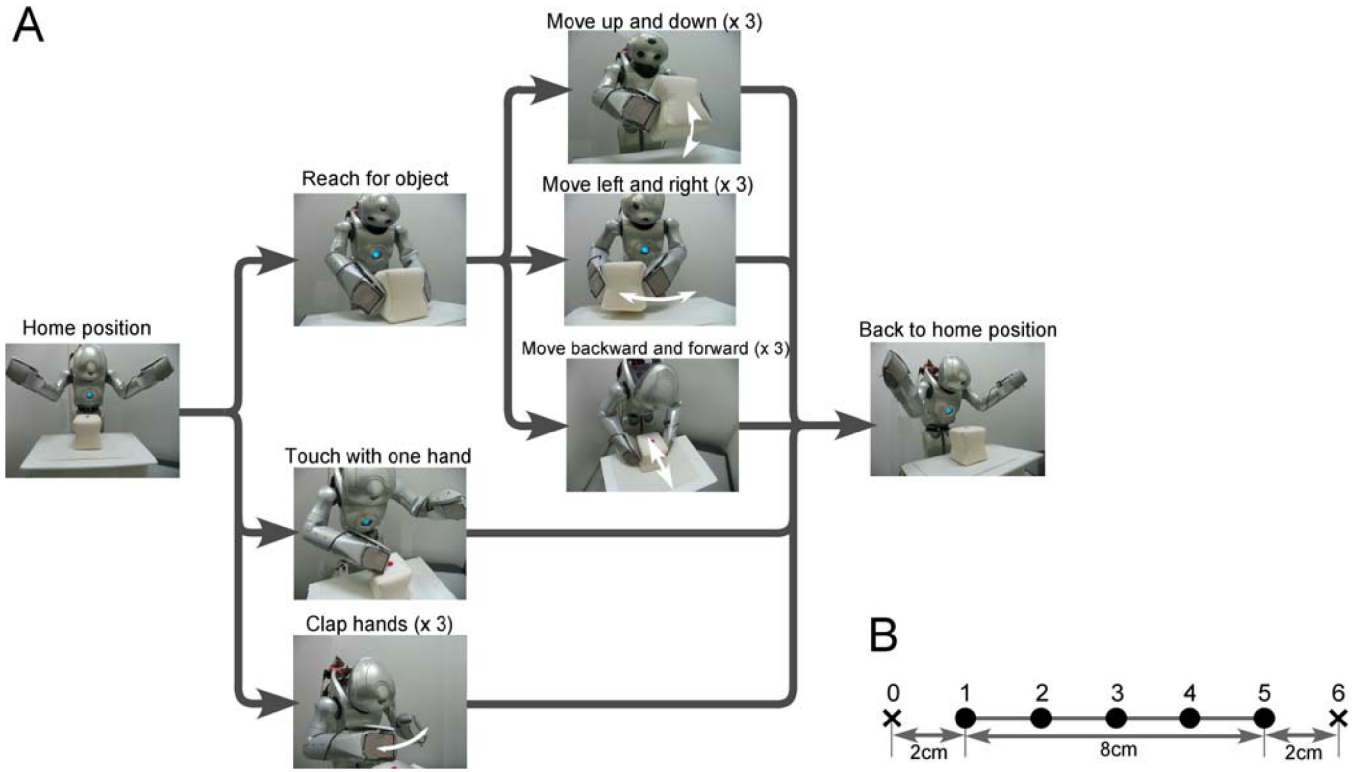
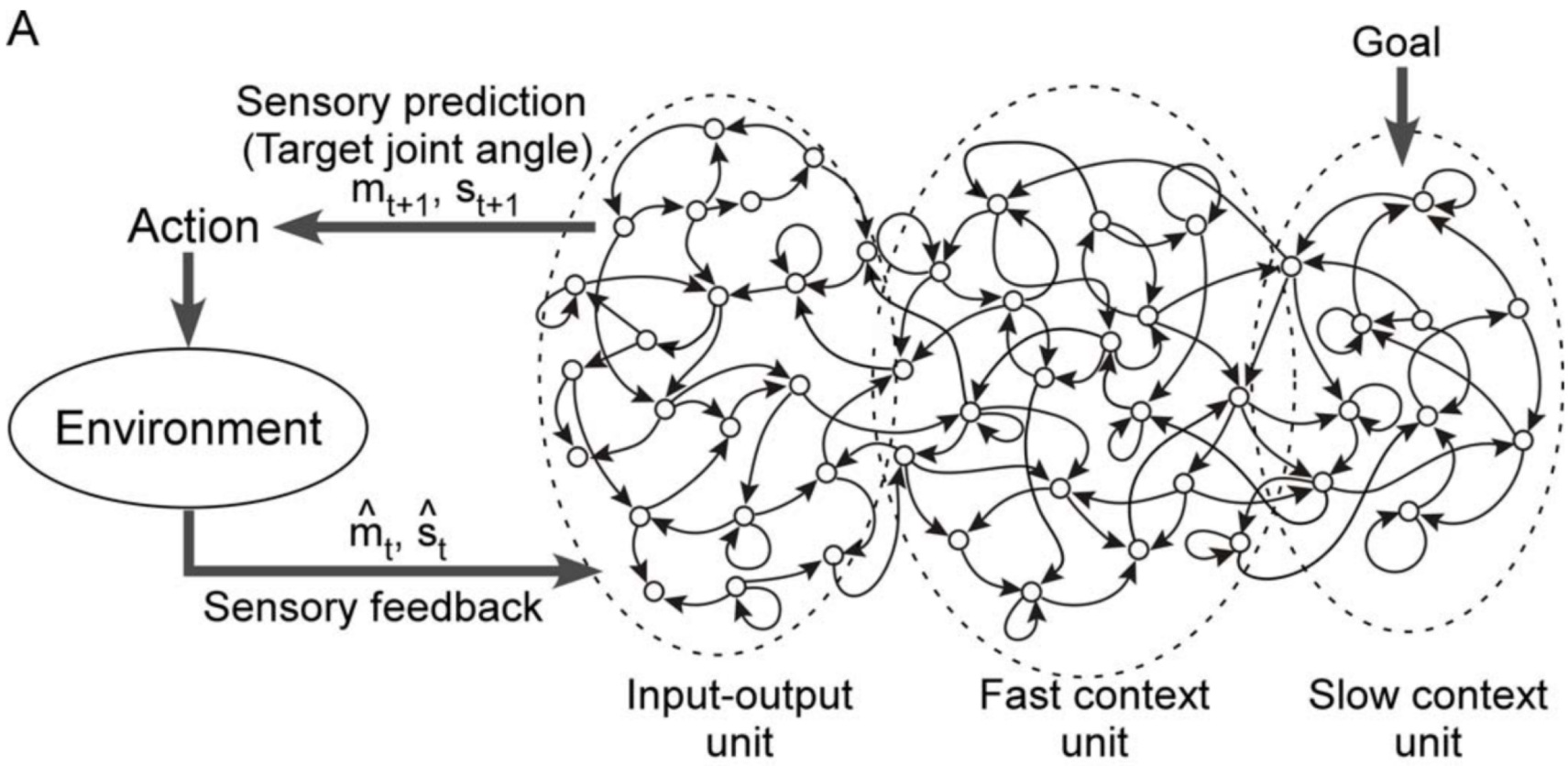


Figure 2. Task design. (A) A humanoid robot was fixed to a stand. In front of the robot, a workbench was set up, and a cubic object (approximately 9×9×9 cm) was placed on the workbench to serve as the goal object. The task for the robot was to autonomously generate five different types of behavior: (1) move the object up and down three times, (2) move the object left and right three times, (3) move the object backward and forward three times, (4) touch the object with one hand, and (5) clap hands three times. For each behavior, the robot began from the home position and ended at the same home position. (B) For each behavior other than the clapping hands task, the object was located at five different positions (positions 1–5). Since the clapping hands behavior was independent of the location of the object, the object was located at the center of the workbench (position 3) and was never moved for this task.
doi:10.1371/journal.pcbi.1000220.g002

Yamashita & Tani 2008

Use RNN with Multiple Time-Scales



https://www.youtube.com/watch?v=n9NYcG8xlYs

Yamashita & Tani 2008

Use RNN with Multiple Time-Scales

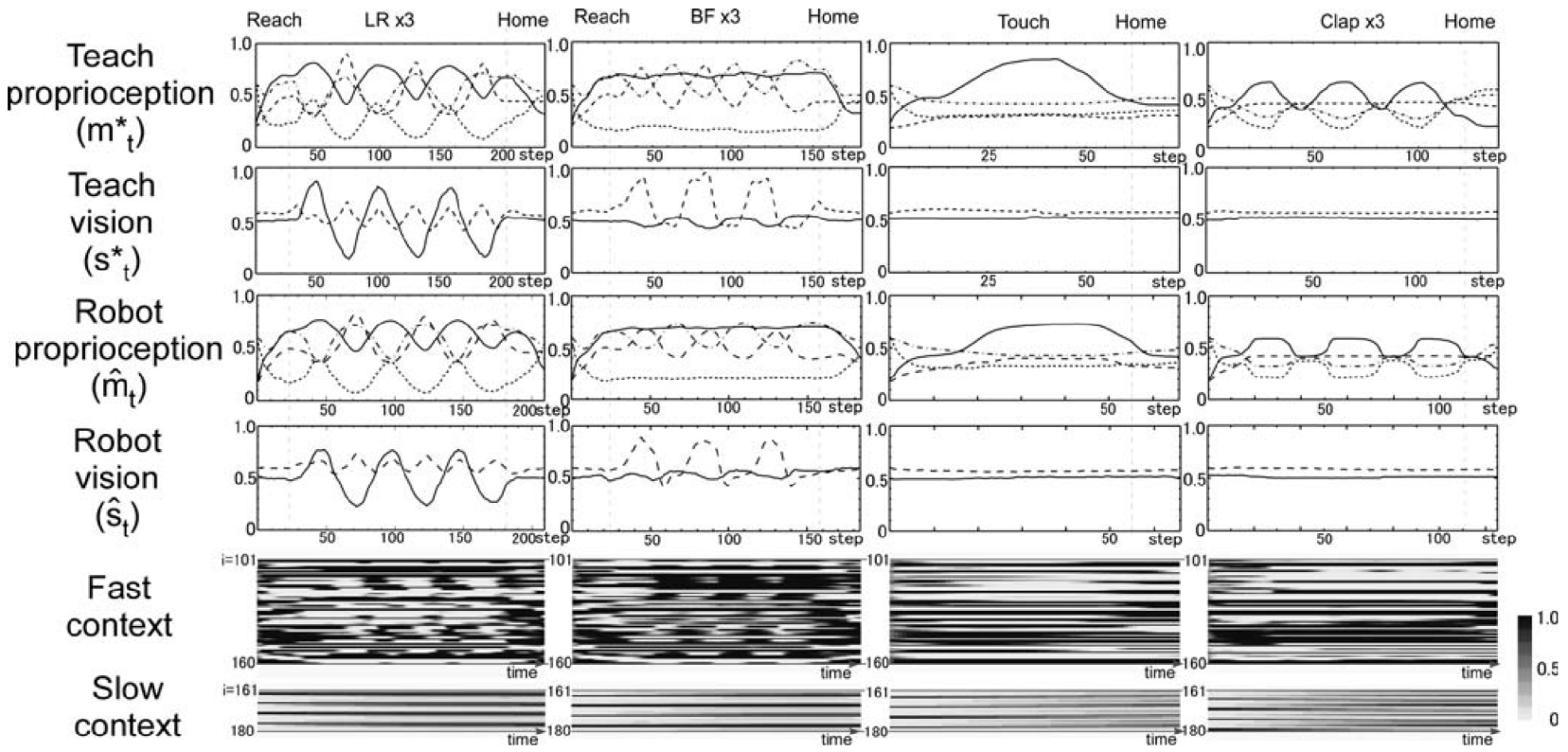


Figure 5. Example of behavior sequences for other basic behavior. Proprioception, vision, fast and slow context activation of teaching signal and actual values in physical environment during left-right (LR: first column), backward-forward (BF: second column) touch with single hand (Touch: third column) and clapping hands (Clap: fourth column) behavior at position 3 are shown. Correspondences for line types in each graph are the same as in Figure 4. Reach: reach to the object, Home: return to the home position.
doi:10.1371/journal.pcbi.1000220.g005

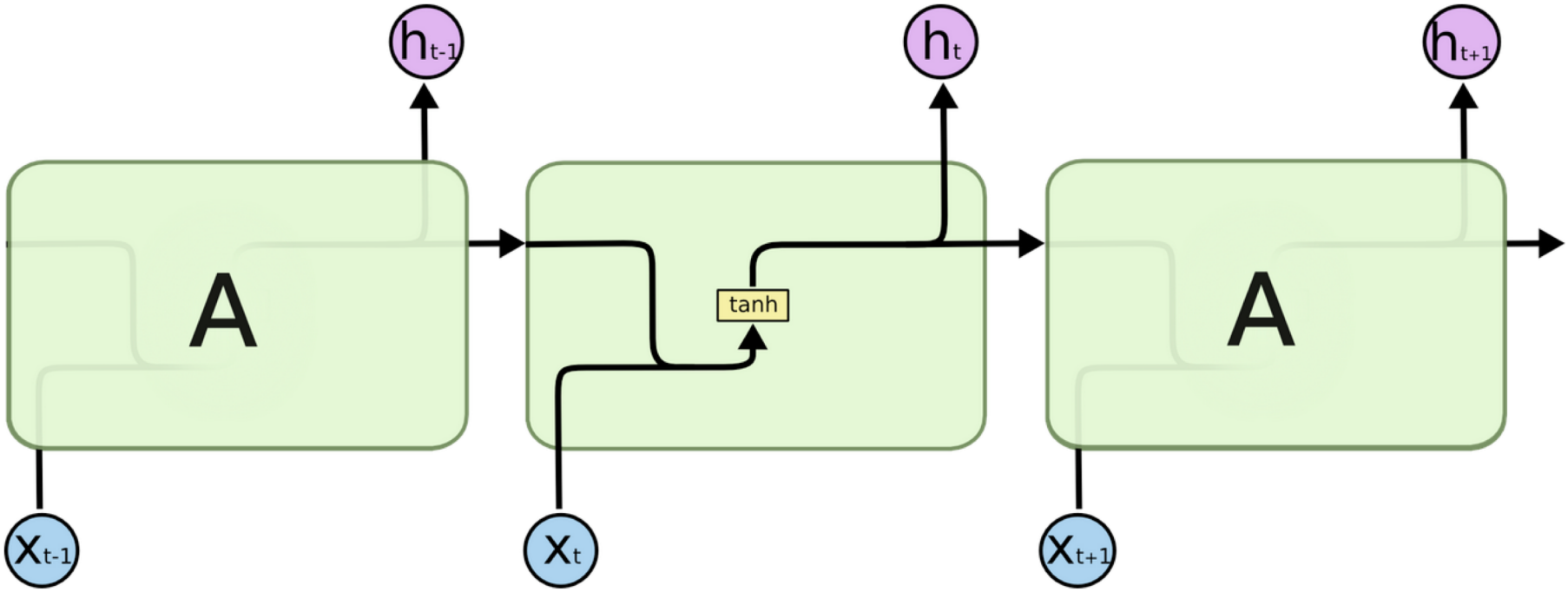
Yamashita & Tani 2008

Plus loin que les RNNs classiques

- Pour pallier au problème de mémoire à long terme dans les RNNs
  - Tani et al. on utilisé les MTRNN (multi-time scale RNN). Plos Comp Bio 2008

- Pour pallier au problème d'« explosion ou évanescence » du gradient (de la *back-propagation*)
  - LSTM: Long-Short Term Memory network (1997)

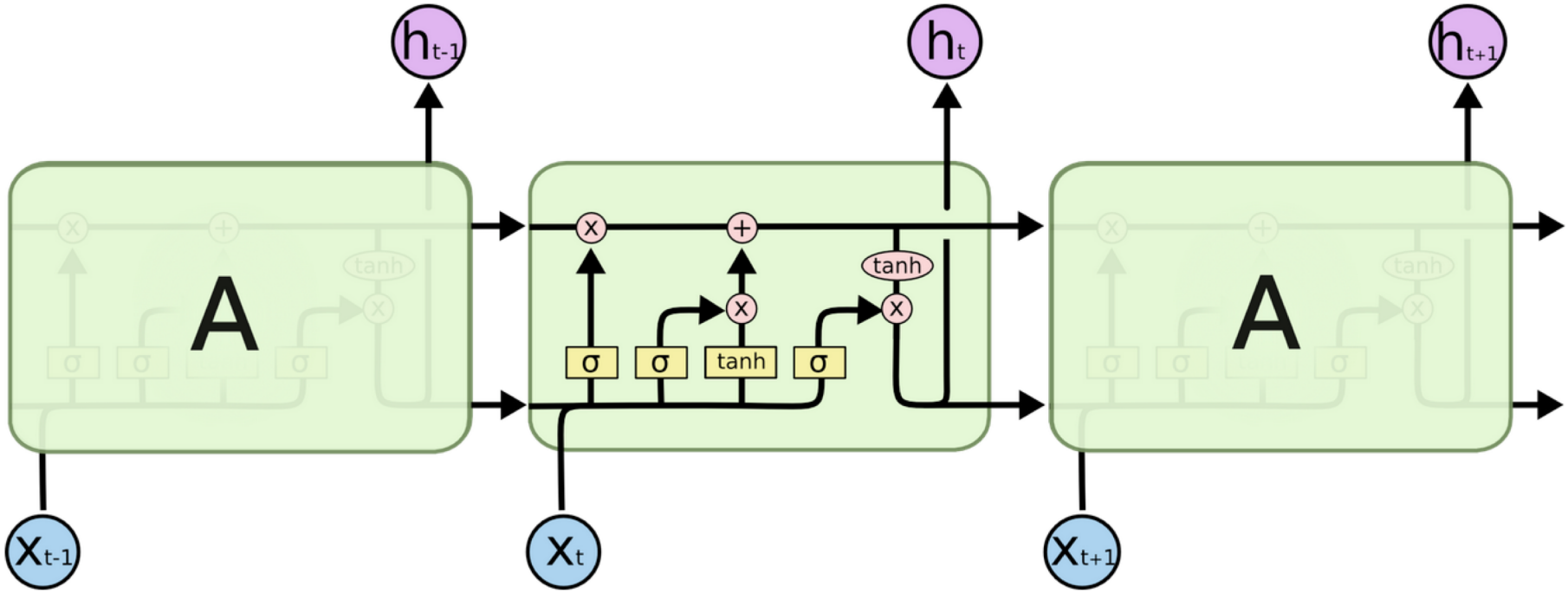
Classical RNNs



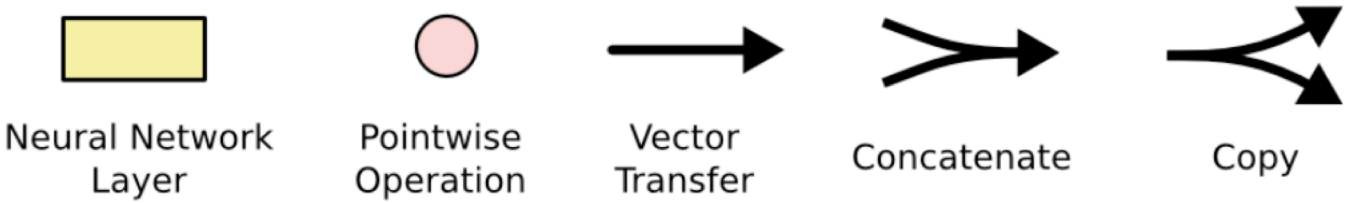
The repeating module in a standard RNN contains a single layer.

LSTM: Long Short-Term Memory Networks

Hochreiter & Schmidhuber (1997)

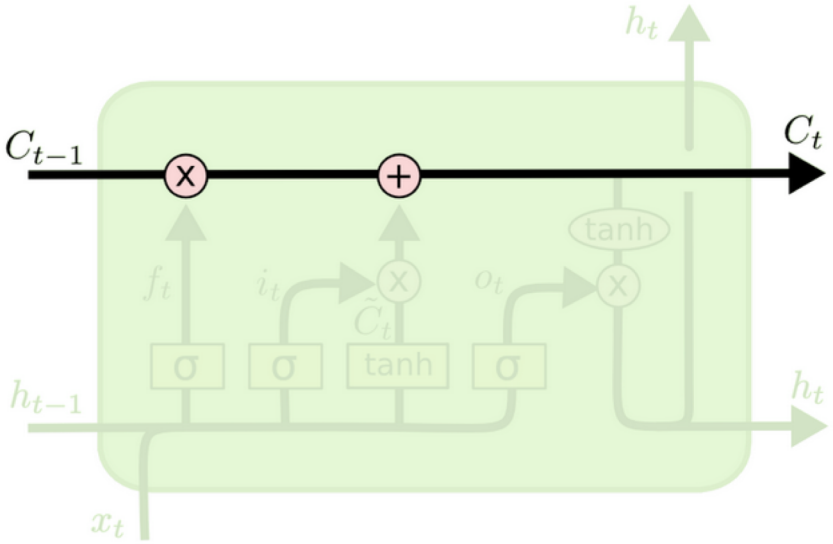


The repeating module in an LSTM contains four interacting layers.



LSTM: the key component

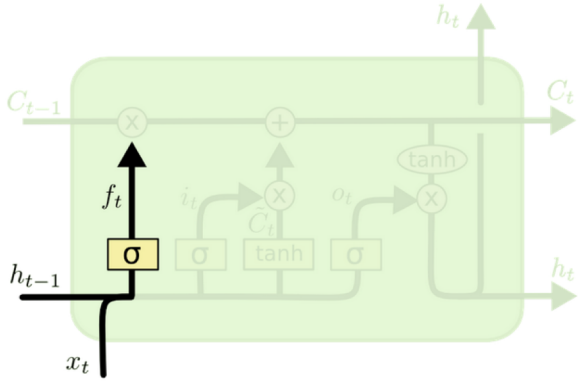
Composant clé:
« cell state »



La « Cell state » permet de garder le gradient constant au cours du temps (lorsque c'est nécessaire).

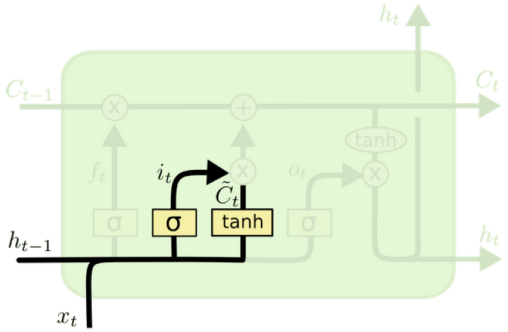
Cela permet de résoudre le problème « d'explosion ou d'évaporation » du gradient.

LSTM: 4 components



forget gate

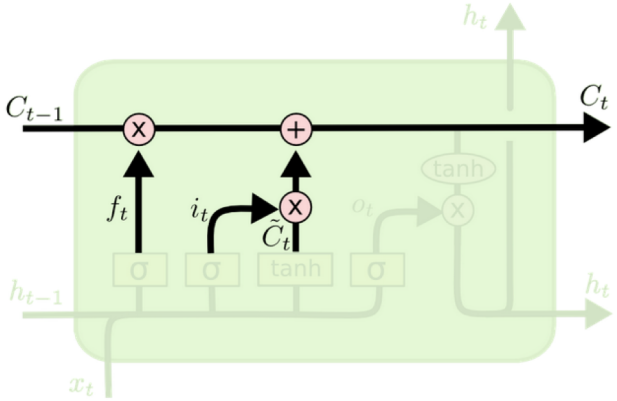
$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$



input gate

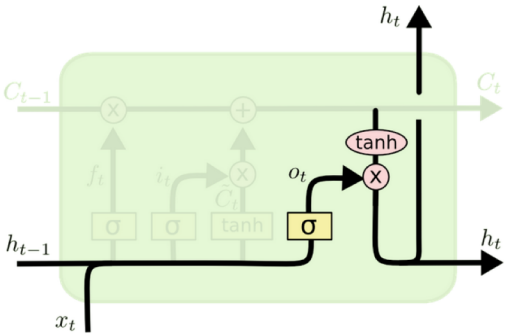
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$



cell state

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$



output gate

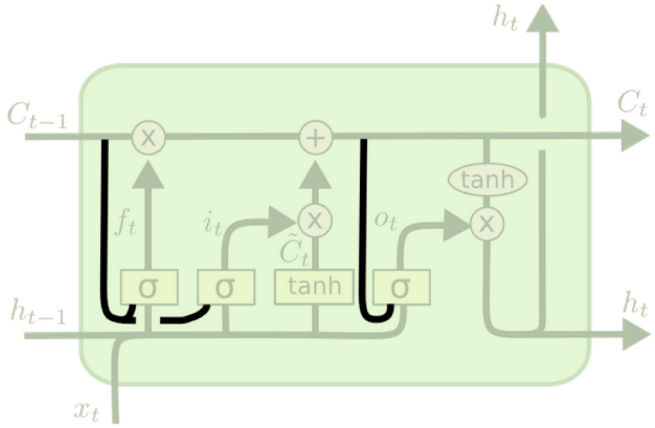
$$o_t = \sigma\left(W_o \; [h_{t-1}, x_t] \; + \; b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

LSTM: Long Short-Term Memory Networks

- Il y a différentes versions de LSTM !
  - Version originale: Hochreiter & Schmidhuber (1997)
  - Version “2000”: Gers & Schmidhuber (2000)
  - GRU (Gated Recurrent Unit): Cho, et al. (2014)
  - …

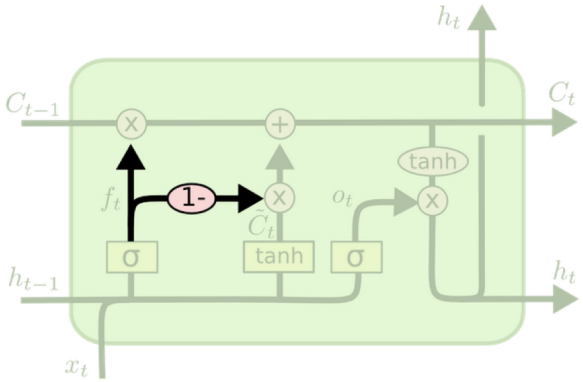
LSTM Variants



Gers & Schmidhuber (2000)

$$f_t = \sigma\left(W_f \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \; + \; b_f\right)$$
$$i_t = \sigma\left(W_i \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \; + \; b_i\right)$$
$$o_t = \sigma\left(W_o \cdot [\boldsymbol{C_t}, h_{t-1}, x_t] \; + \; b_o\right)$$

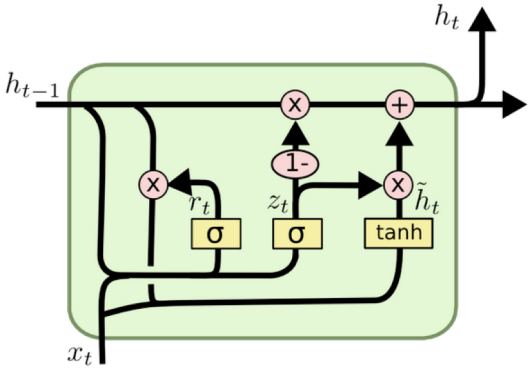
adding "peephole connections"



coupled forget and input gates

$$C_t = f_t * C_{t-1} + (1 - \boldsymbol{f_t}) * \tilde{C}_t$$

On décide simultanément ce que vont faire l'*input* et la *forget gate*:
On oublie seulement lorsque l'on va remplacer l'état par l'entrée.



GRU: Gated Recurrent Unit

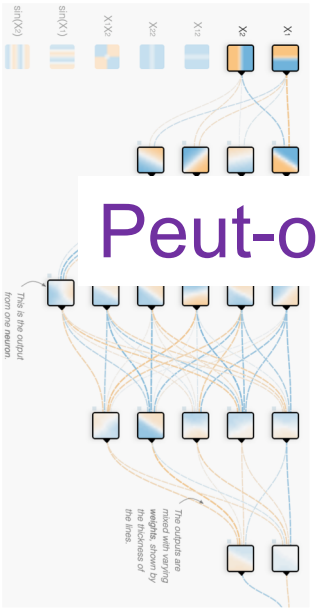
$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

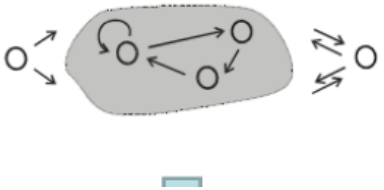
Comment apprendre dans un réseau récurrent ?

- Si on veut appliquer l'algorithme de rétro-propagation du gradient
- cela impl
- et donc de dupliquer le réseau à chaque pas de temps
- afin appliquer l'algo. comme si c'était un réseau en couches

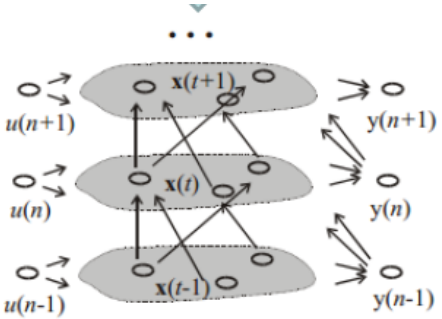
Mais c'est couteux en calculs !

(et pas biologiquement plausible)



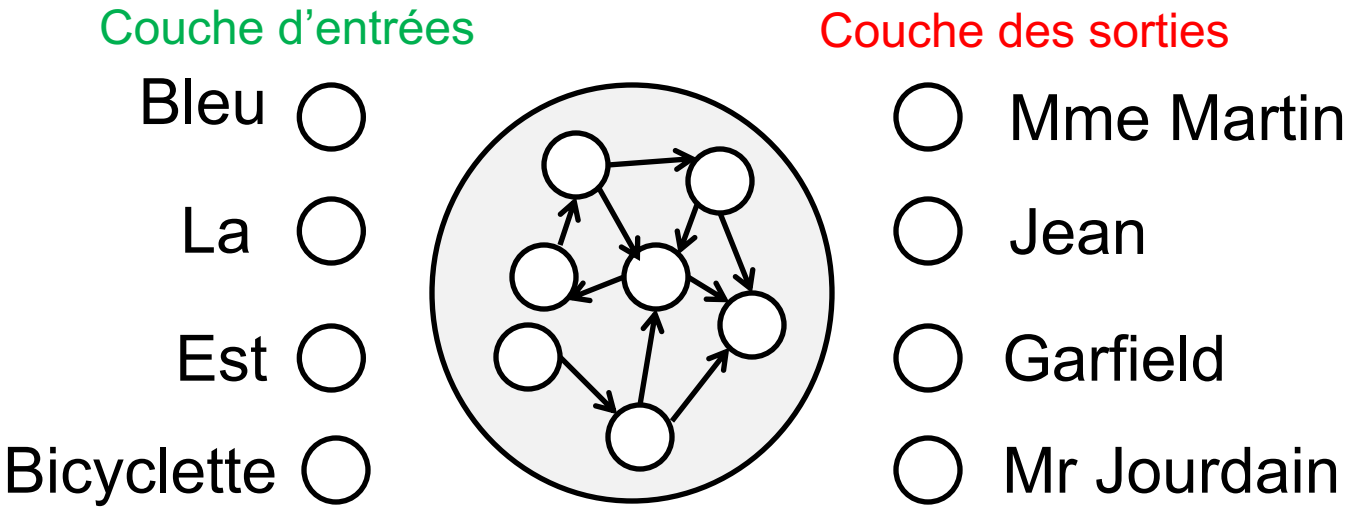


Peut-on faire autrement ?



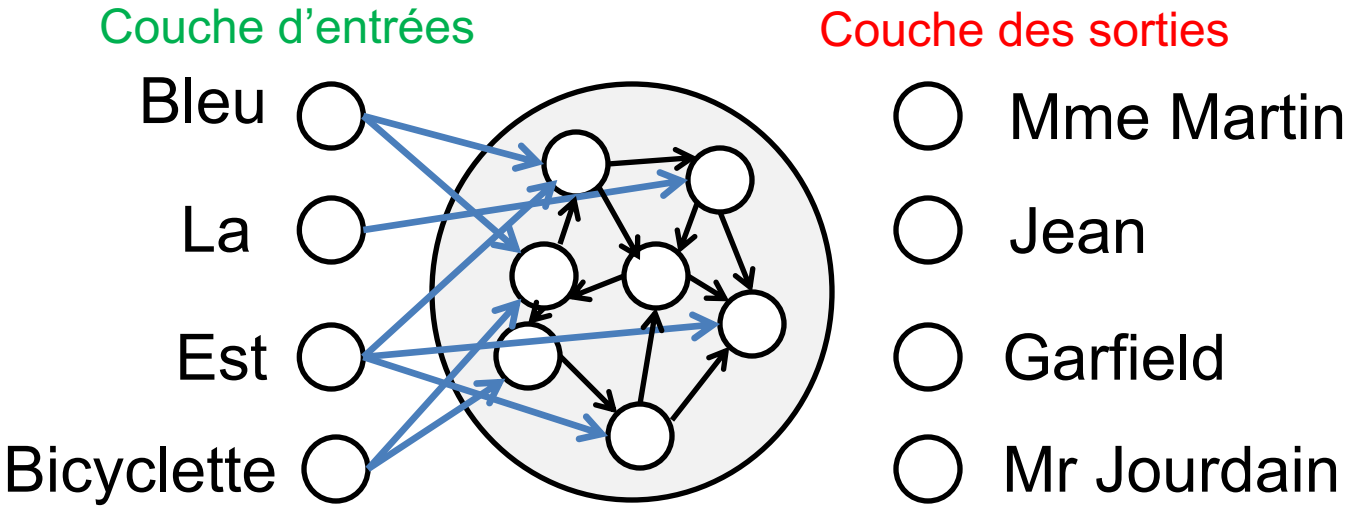
Réseau récurrent pour les séquences

« La bicyclette est bleue »



Réseau récurrent pour les séquences

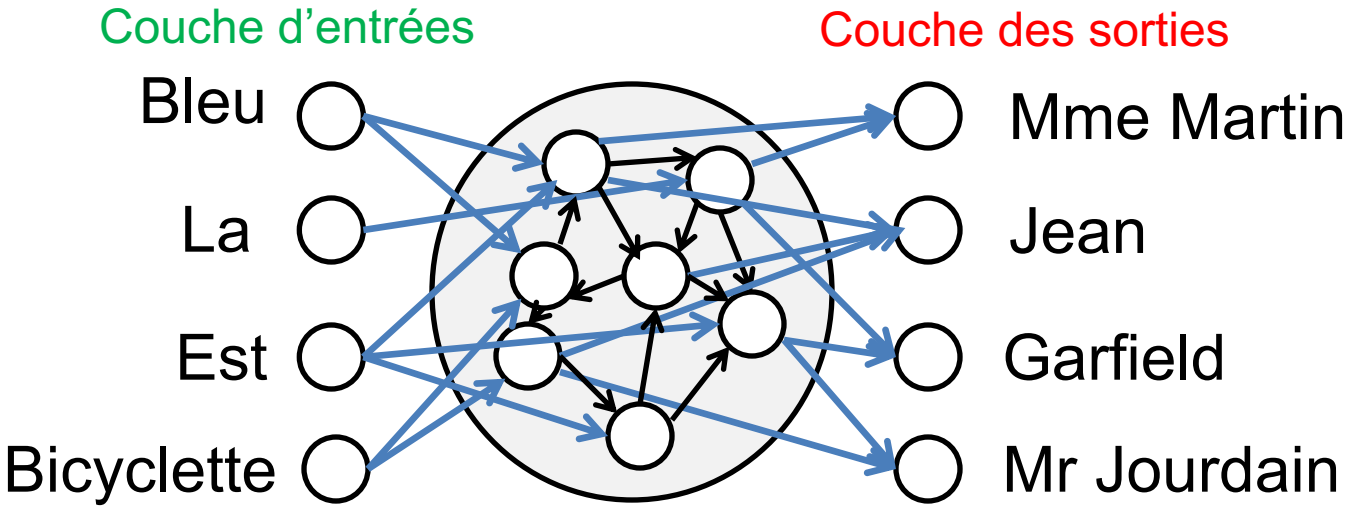
« La bicyclette est bleue »



- On va connecter la couche d'entrée (les mots possibles de la phrase) au réseau de neurone récurrent (le réservoir).

Réseau récurrent pour les séquences

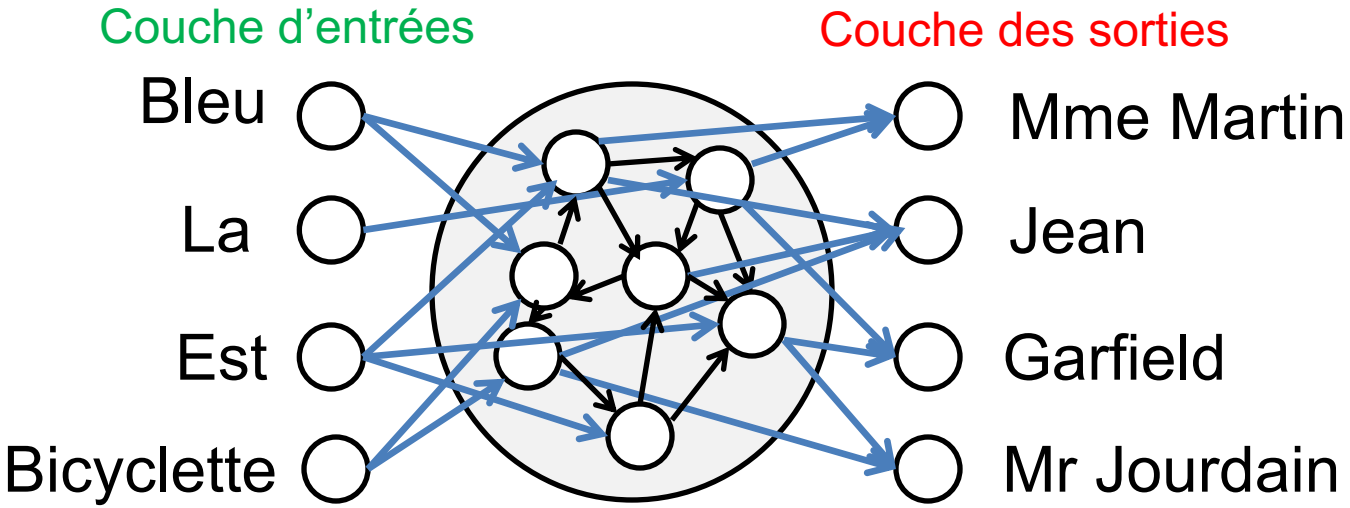
« La bicyclette est bleue »



- On va connecter la couche d'entrée (les mots possibles de la phrase) au réseau de neurone récurrent (le réservoir).
- Puis on va connecter le réseau de neurones récurrent à la couche de sortie (les noms des auteurs possibles)

Réseau récurrent pour les séquences

« La bicyclette est bleue »

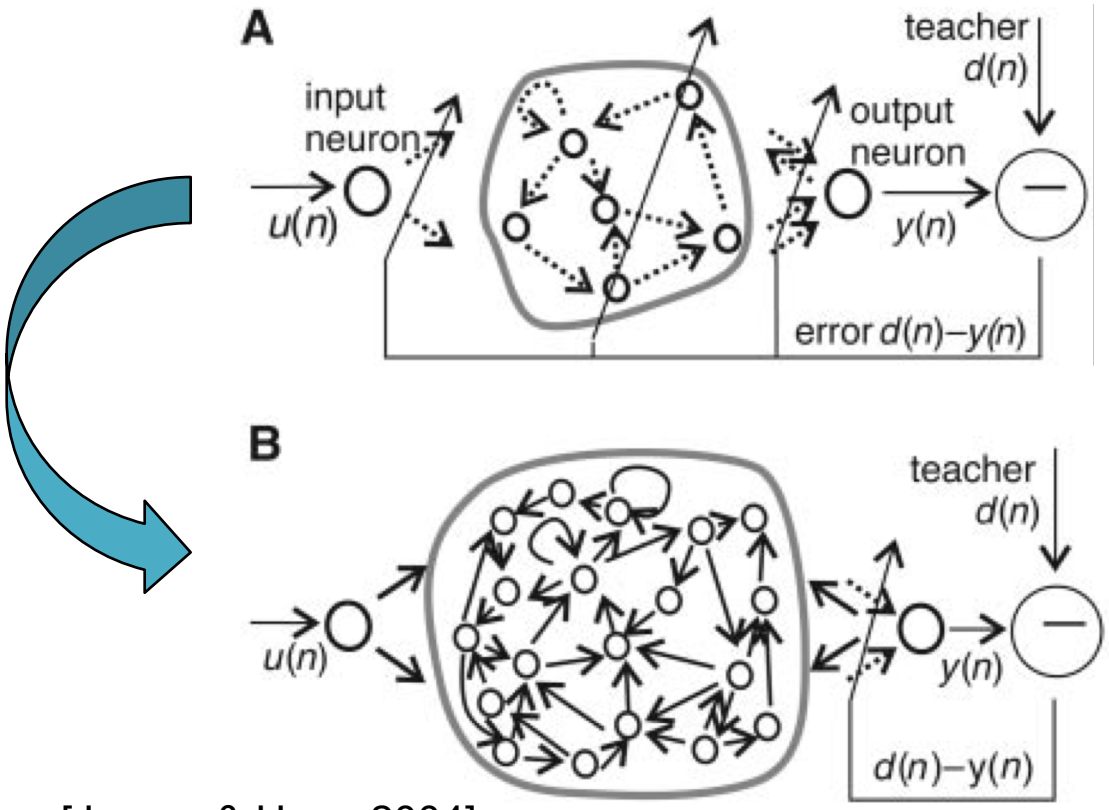


- On rentre les mots un par un.
- Les activations de neurones vont générer une mémoire dynamique des mots de la phrase.
- Représentation du contexte temporel.

Accélérer la phase d'apprentissage d'un RNN ?

- Ne pas modifier toutes les connexions
  - (e.g. ***juste la couche de sortie***)
- ***Tirer aléatoirement*** les connexions non entrainées (la couche d'entrée et la couche de sortie)

- Utiliser des ***méthodes linéaires*** pour l'apprentissage (e.g. la régression linéaire)

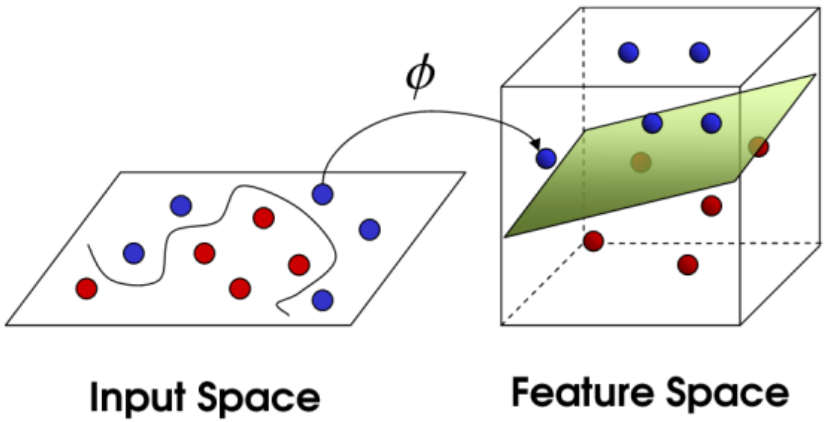


[Jaeger & Hass 2004]

RNN = Recurrent Neural Network

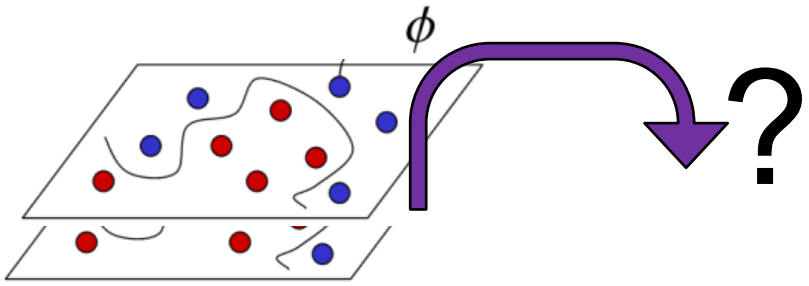
Pourquoi est-ce que ça marche ?

- Grande couche cachée (100~1000 neurones)
  - Projection des entrées dans un ***espace de haute dimension***
  - Beaucoup de ***calculs non-linéaires*** sont effectués sur la base des entrées
  - ***Dynamiques temporelles riches*** (même avec un réseau aléatoire fixé)
  - Réservoir = (couche cachée) d'un ESN
  - Comme un SVM temporel (Support Vector Machine)



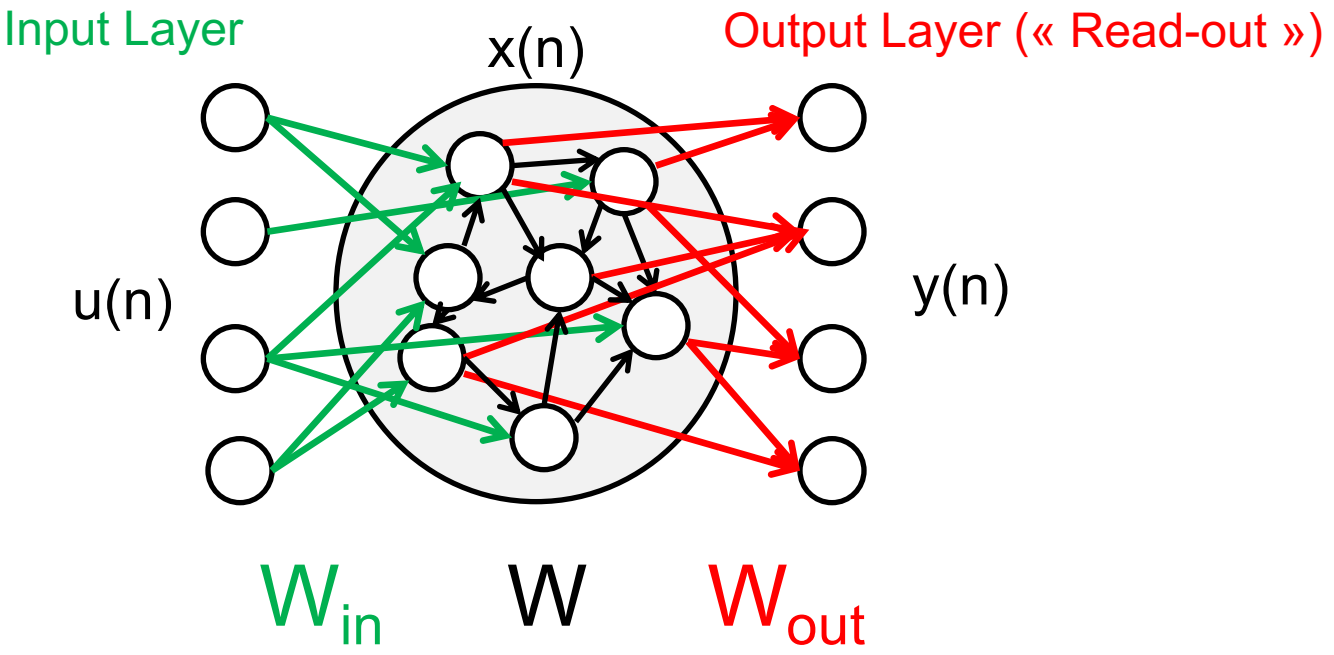
Quand est-ce que ça marche moins bien ?

- Plus les entrées vont être de grande dimension, plus la dimension du réservoir devra être grande !
- Le problème est que les capacités computationnelles du réservoir n'augementent pas linéairement avec l'augmentation du nombre de neurones (du réservoir)
- Au delà de 1000 neurones, on augmente plus beaucoup les performances en augmentant la taille du réservoir. Intuitivement :
  - passer de 100 à 200 neurones aura plus d'effet sur les performances
  - que de passer à 1100 à 1200 neurones.



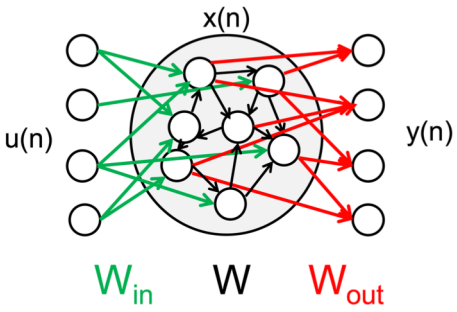
(too big) **Input Space**

Reservoir Computing

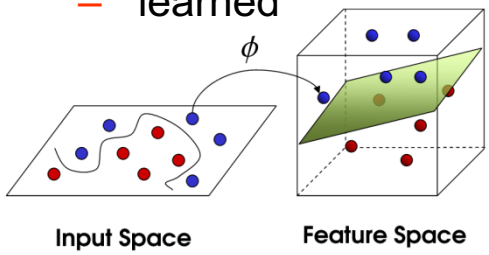


- W_{in}, W, W_{out} : connection weight matrices
- W_{in} (input->reservoir), W (recurrent), W_{out} (reservoir->outputs)
- W_{in}, W: randomly generated (sparse)
- W_{out} : weights are found with training (supervised learning)

Résumé sur l'apprentissage hors-ligne



- W_{in}, W:
  - randomly generated
- W_{out} :
  - learned



Jaeger 2001, Jaeger et al. 2007, Lukosevicius 2012

Reservoir states update for *leaky* integrator neurons

$$\mathbf{x}(t) = (1 - \frac{1}{\tau})\mathbf{x}(t-1) + \frac{1}{\tau} f(\mathbf{W}^{in} u(t) + \mathbf{W}\mathbf{x}(t-1)) \quad (1)$$

- x the reservoir state vector,
- u the input vector,
- W the reservoir recurrent weight matrix,
- W^{in} the weight matrix from input layer to the reservoir,
- τ the time constant of a reservoir unit,
- f the activation function (i.e. *tanh*) of a reservoir unit.

Output layer (read-out) state update

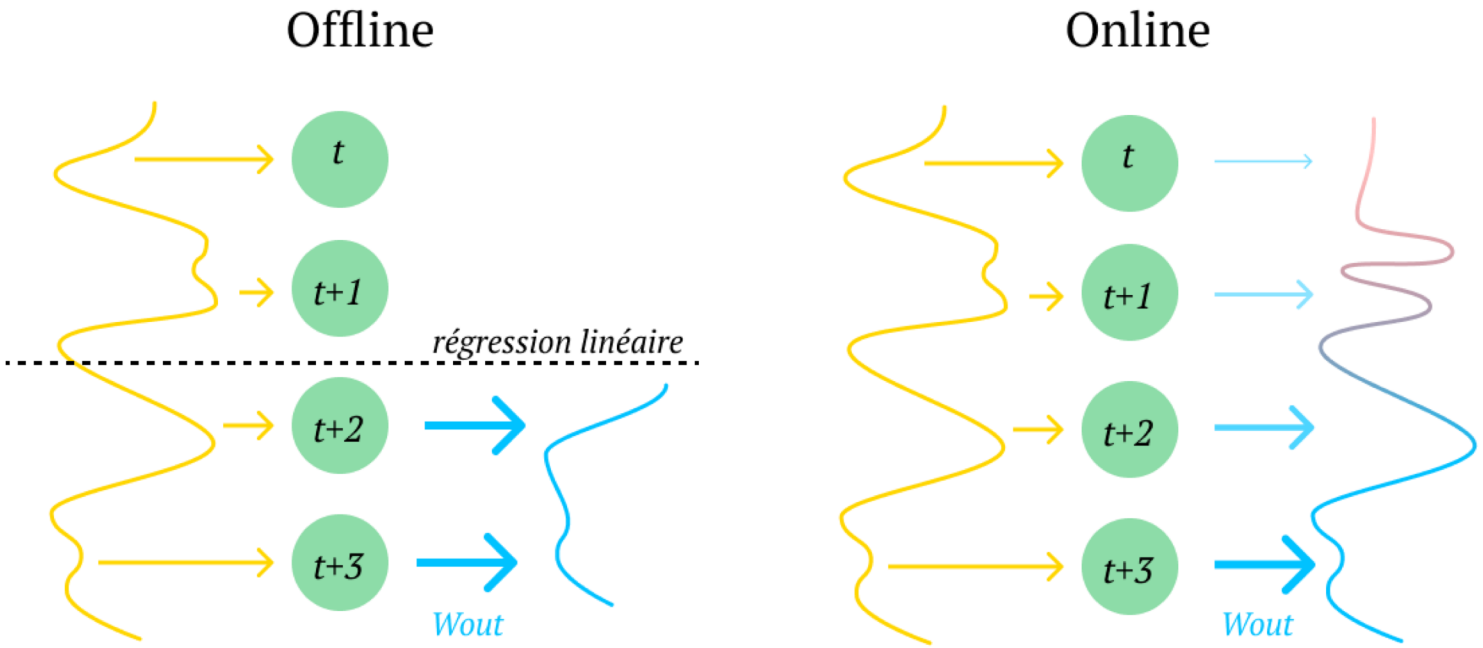
$$\mathbf{y}(t) = \mathbf{W}^{out}\mathbf{x}(t) \quad (2)$$

Offline learning of output weights

$$\mathbf{W}^{out} = Y^d Z^+ \quad (3)$$

- Y^d the concatenation of the desired outputs,
- Z the concatenation of reservoir states, with $Z = [X; 1]$
- $^+$ denotes the pseudo-inverse.

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



FORCE Learning (incremental)

Calcul de la sortie : Le réseau forme des sorties au cours du temps au fur et à mesure qu'il reçoit de nouvelles entrées. En recevant au cours du temps le signal $(\mathbf{x}_1, \cdots, \mathbf{x}_n)$, le réseau calcule, à chaque un instant $t \in [\![1; n]\!]$, l'activité de sa couche récurrente $\mathbf{r}_t$ (son état interne), $\mathbf{s}_t$, et sa sortie $\mathbf{y}_t$, selon les équations (1) et (2), afin de renvoyer le signal $(\mathbf{y}_1, \cdots, \mathbf{y}_n)$.

$$\mathbf{r}_t \leftarrow (1-\alpha)\ \mathbf{r}_{t-1}\ +\ \alpha\ \mathbf{tanh}(\mathbf{W}^{rec}\mathbf{r}_{t-1} + \mathbf{W}^{in}\mathbf{x}_t + \mathbf{W}^{fb}\mathbf{y}_{t-1}) \tag{1}$$

$$\mathbf{s}_t \leftarrow \begin{pmatrix} 1 \\ \mathbf{x}_t \\ \mathbf{r}_t \end{pmatrix} \qquad \mathbf{y}_t \leftarrow \mathbf{W}^{out}\mathbf{s}_t \tag{2}$$

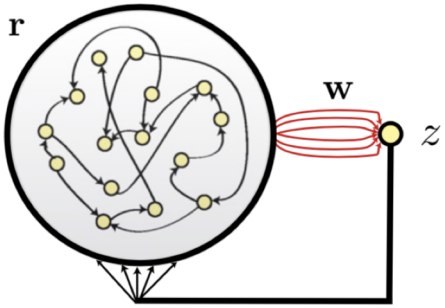
Vecteurs :

- $\mathbf{x}_t$: entrée fournie au réseau à un instant t.
- $\mathbf{r}_t$: activité de la couche récurrente du réseau après lecture de $\mathbf{x}_t$
- $\mathbf{s}_t$: concaténation d'un biais constant, de $\mathbf{x}_t$, et de $\mathbf{r}_t$
- $\mathbf{y}_t$: sortie du réseau après lecture de $\mathbf{x}_t$

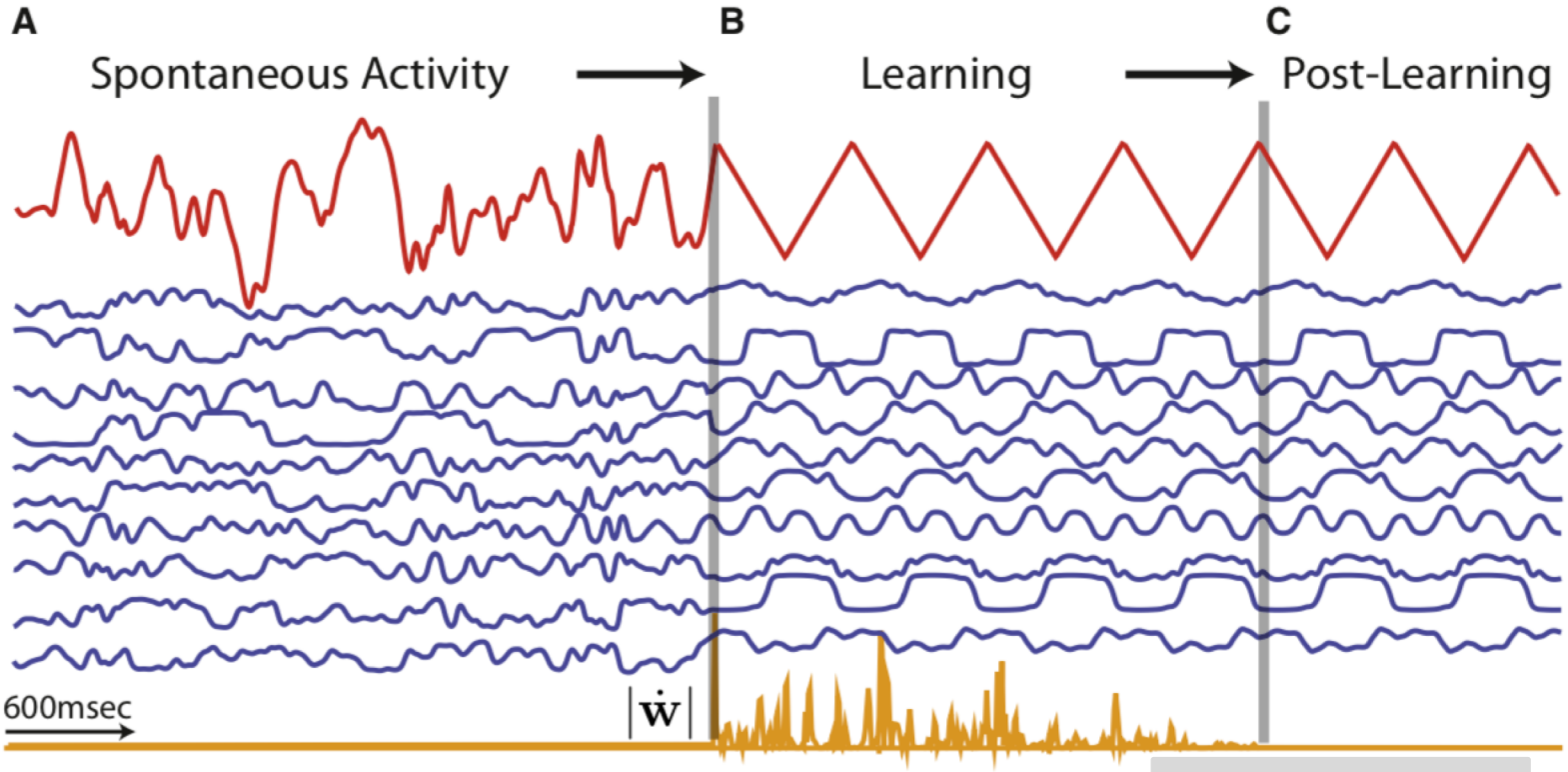
Juven & Hinaut 2020

Sussillo & Abbott 2009

FORCE LEARNING

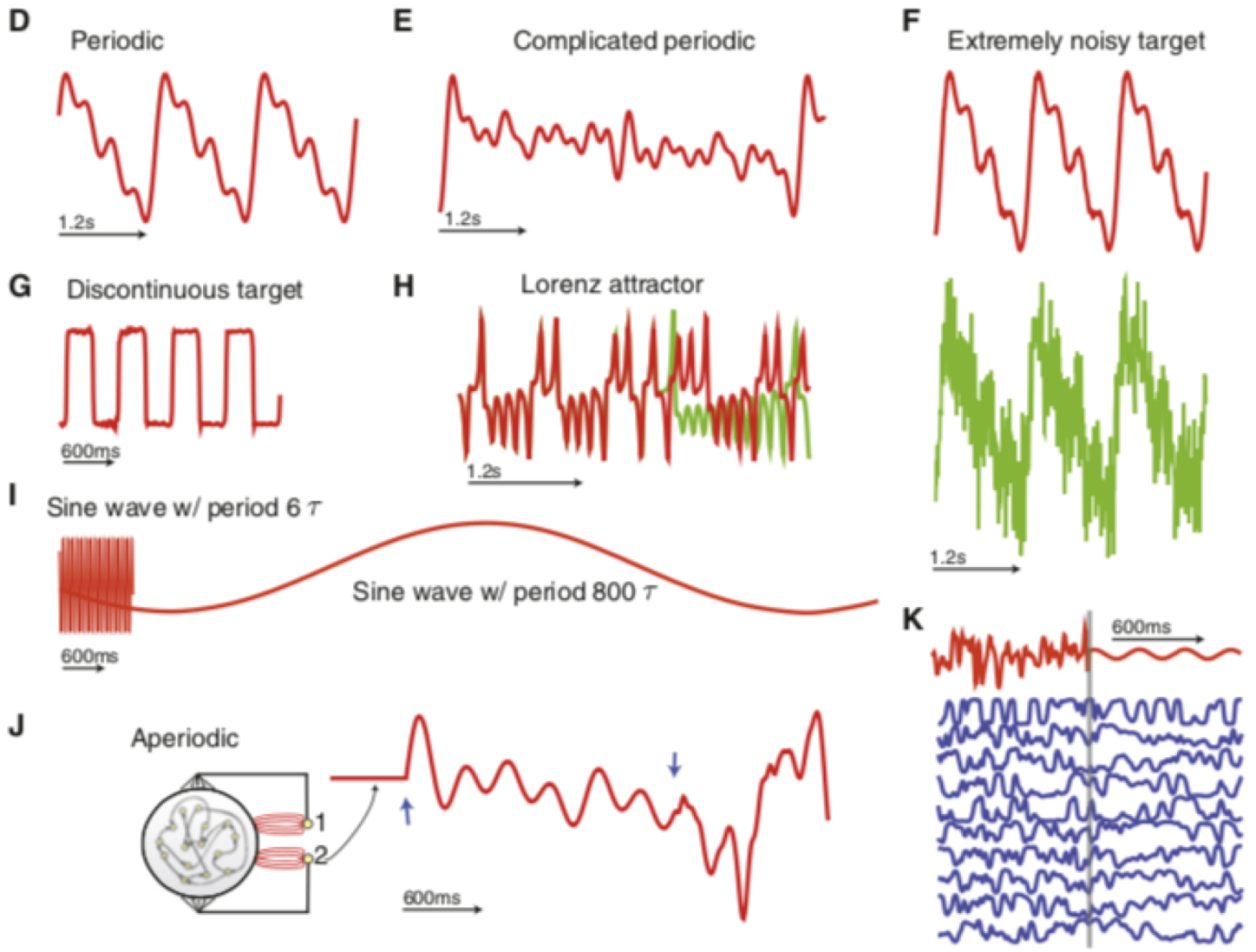


A good exemple of online learning
(Sussillo & Abbott, Neuron 2009)



Sussillo & Abbott 2009

FORCE LEARNING



Sussillo & Abbott 2009

FORCE learning

sortie du réseau (avant mise à jour des poids) $\mathbf{w}^T(t - \Delta t)\mathbf{r}(t)$

sortie désirée $f(t)$

erreur (avant mise à jour) $e_-(t) = \mathbf{w}^T(t - \Delta t)\mathbf{r}(t) - f(t)$

erreur (après mise à jour) $e_+(t) = \mathbf{w}^T(t)\mathbf{r}(t) - f(t)$

where $\mathbf{P}(t)$ is an $N \times N$ matrix that is updated at the same time as the weights according to the rule

$$\mathbf{P}(t) = \mathbf{P}(t - \Delta t) - \frac{\mathbf{P}(t - \Delta t)\mathbf{r}(t)\mathbf{r}^T(t)\mathbf{P}(t - \Delta t)}{1 + \mathbf{r}^T(t)\mathbf{P}(t - \Delta t)\mathbf{r}(t)}. \quad (5)$$

mise à jour des poids $\mathbf{w}(t) = \mathbf{w}(t - \Delta t) - e_-(t)\mathbf{P}(t)\mathbf{r}(t)$

The algorithm also requires an initial value for $\mathbf{P}$, which is taken to be

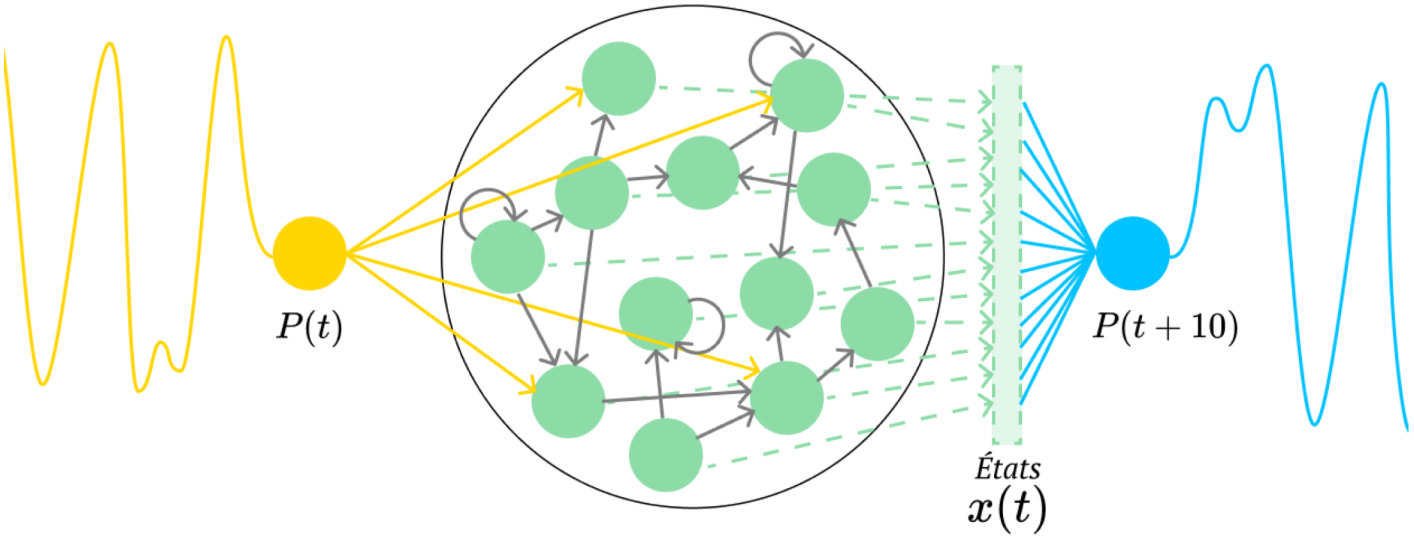
$$\mathbf{P}(0) = \frac{\mathbf{I}}{\alpha}, \quad (6)$$

where $\mathbf{I}$ is the identity matrix and α is a constant parameter.

"Qu'est-ce qu'on en fait ?”

- Prédiction de séries temporelles
- Discrimination de sons
- Apprendre le langage à un robot

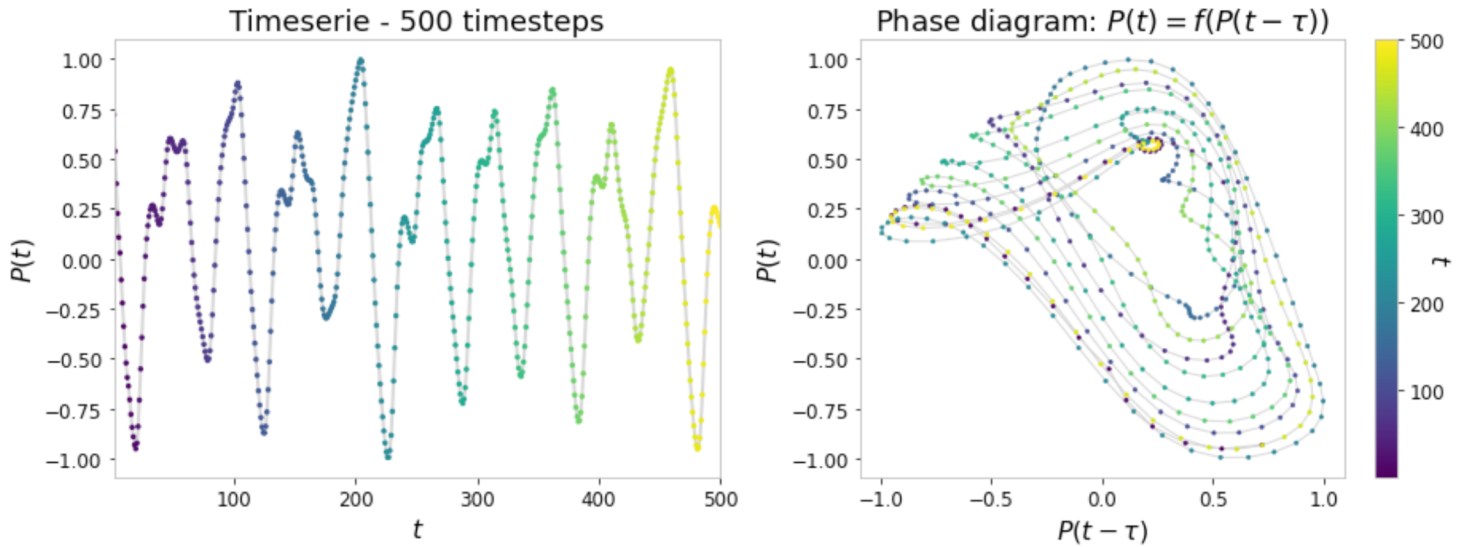
Prédiction de series temporelles



Prédiction de series temporelles

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

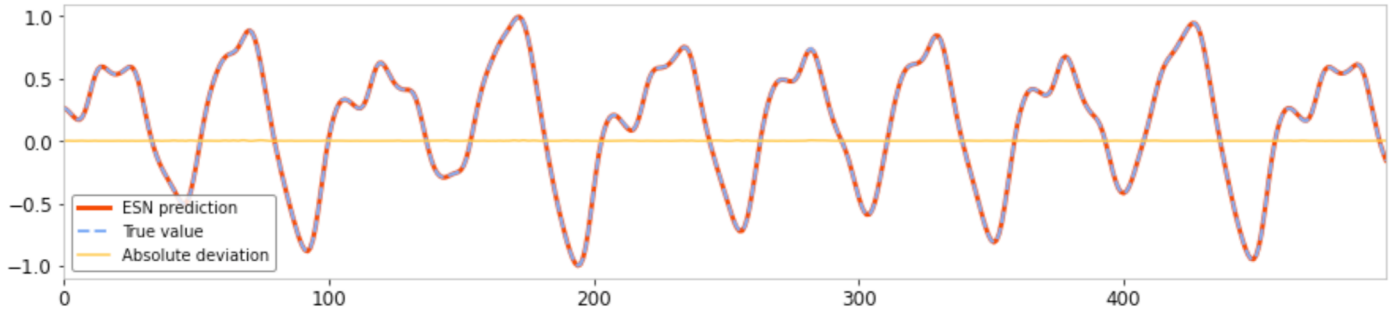
```
plot_mackey_glass(X, t, 500, tau)
```



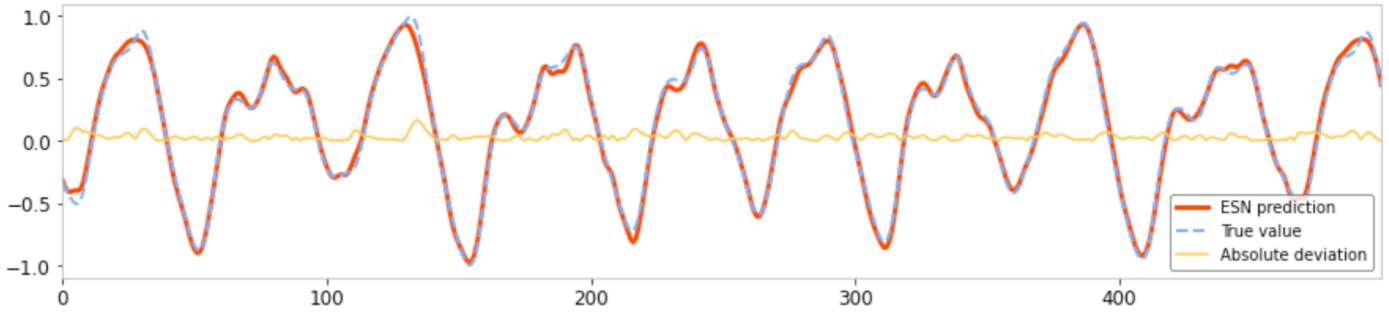
Exemple tiré de github.com/reservoirpy/reservoirpy

Prédiction de series temporelles

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



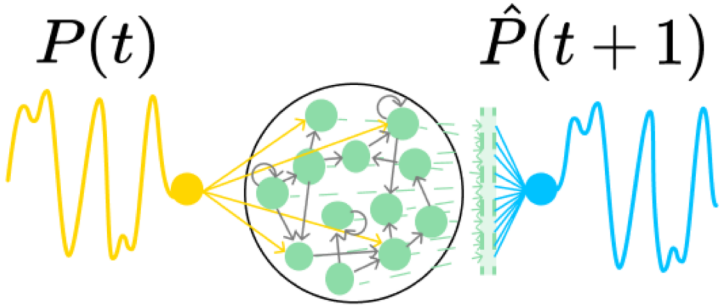
- Prédictions obtenues pour 50 pas de temps dans le futur (T+50)



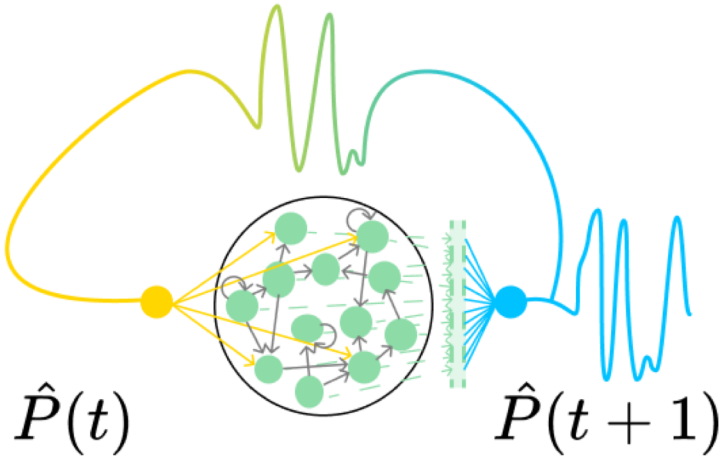
Exemple tiré de github.com/reservoirpy/reservoirpy

Génération de series temporelles

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



Mode Prédiction

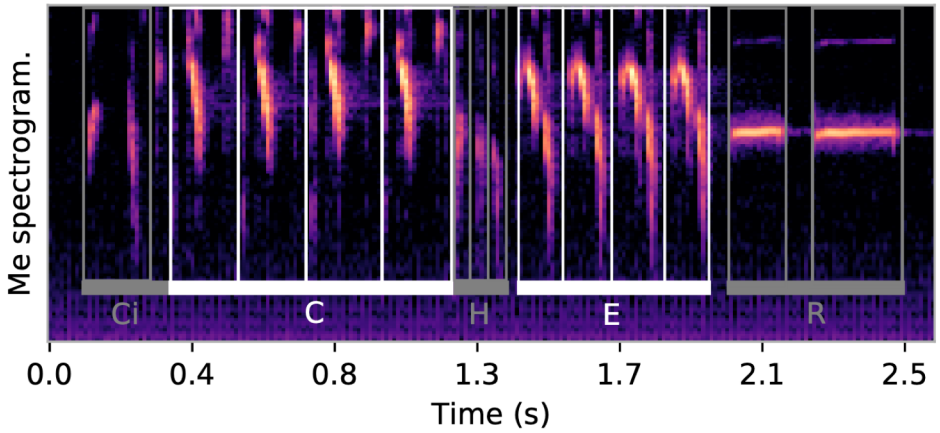


Mode Génération

"Qu'est-ce qu'on en fait ?”

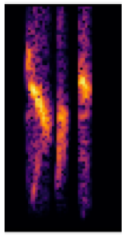
- Prédiction de séries temporelles
- Discrimination de sons (Reservoirs vs. LSTMs)
- Apprendre le langage à un robot (Reservoirs vs. LSTMs)

Discrimination de phrases de canaris

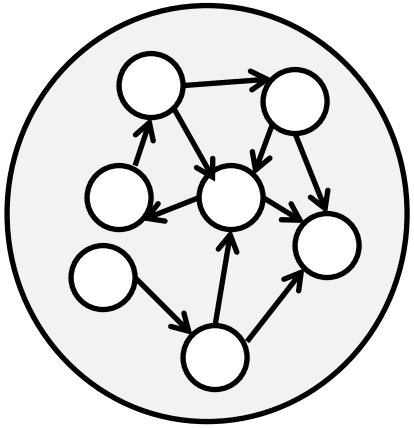


Input preproc. as MFCCs

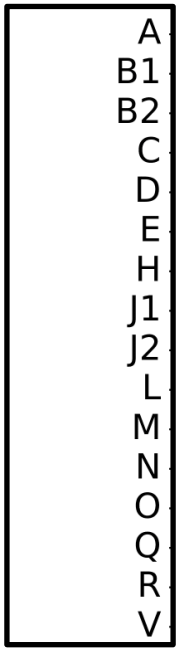
RNN training/testing



Input
(one raw
at a time)



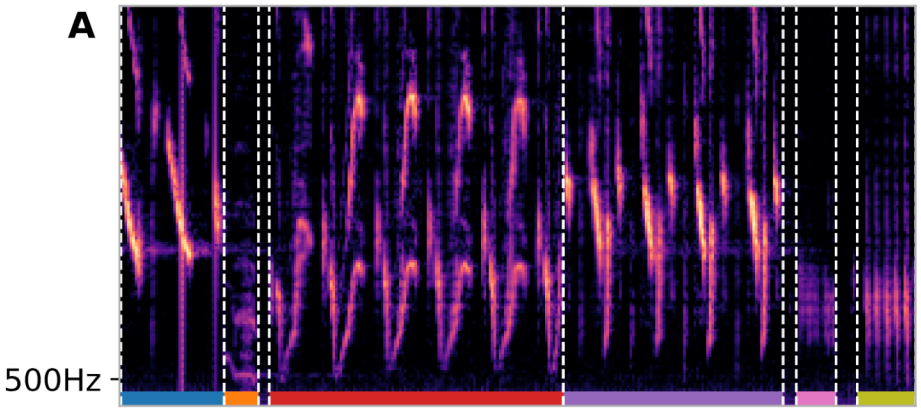
Reservoir
or
LSTM



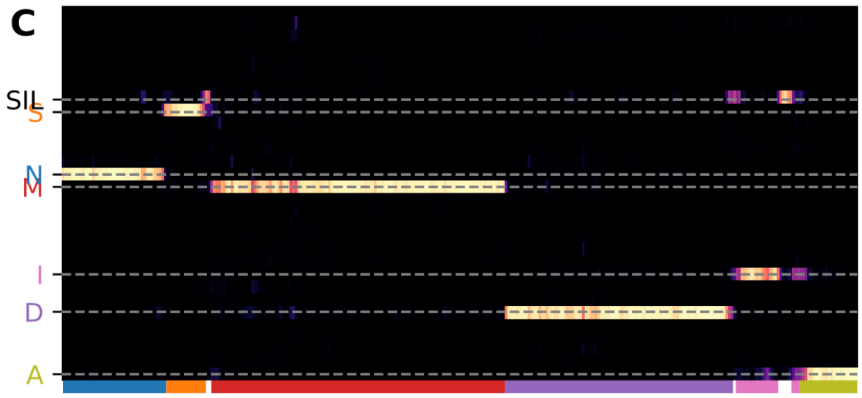
Output

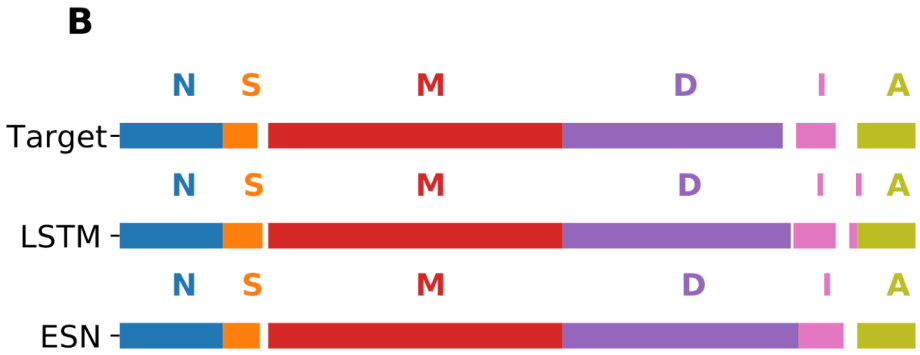
Discrimination de phrases de canaris

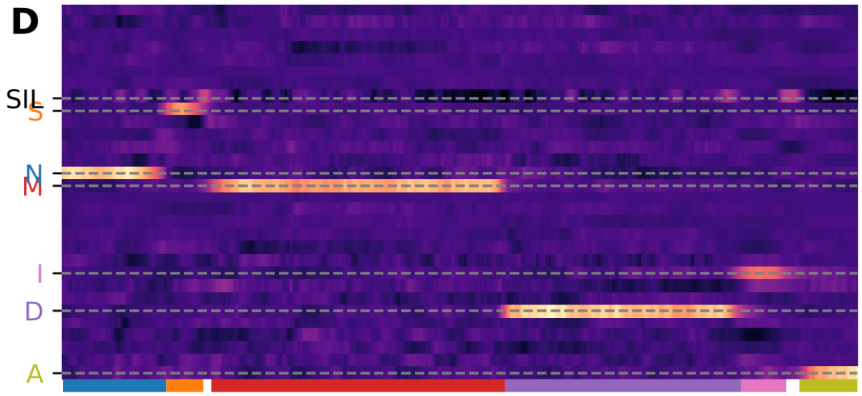
Mel-scale spectrogram



LSTM chromagram-like output







Target and predicted annotation sequence

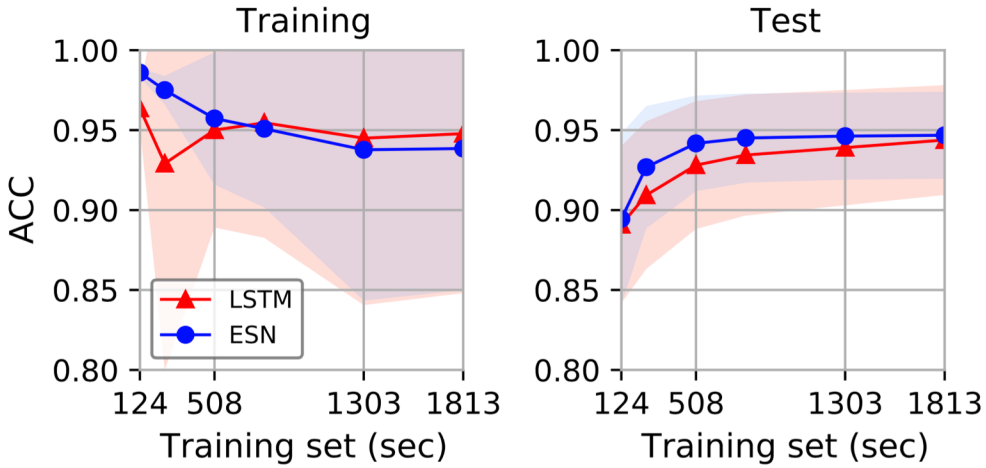
Reservoir chromagram-like output

Discrimination de phrases de canaris

Table 1. Average scores obtained with a 5-fold cross-validation over all training songs and several models instances.

Model	Average frame accuracy (ACC)	Median frame accuracy	F1 (macro avg.)
LSTM	0.931 ± 0.104	0.951	0.865
ESN	0.935 ± 0.09	0.952	0.877

Frame accuracy vs. data training size



Average training time (10-fold CV)

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

Les performances en généralisation sont similaires entre les LSTM et les reservoirs, mais :

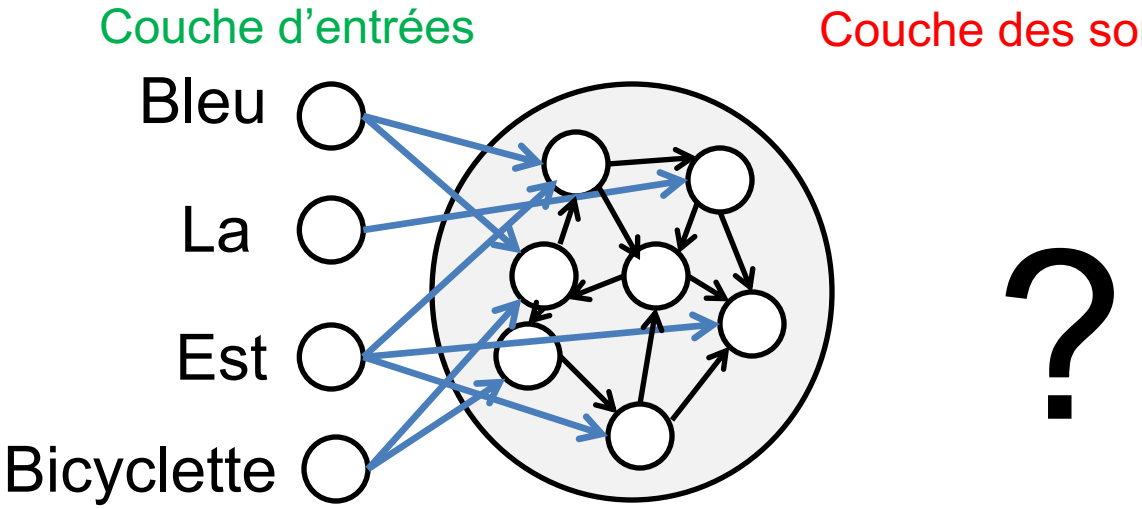
- les réservoirs généralisent avec moins de données
- les réservoirs sont bien plus rapides à entraîner

"Qu'est-ce qu'on en fait ?”

- Prédiction de séries temporelles
- Discrimination de sons (Reservoirs vs. LSTMs)
- Apprendre le langage à un robot (Reservoirs vs. LSTMs)

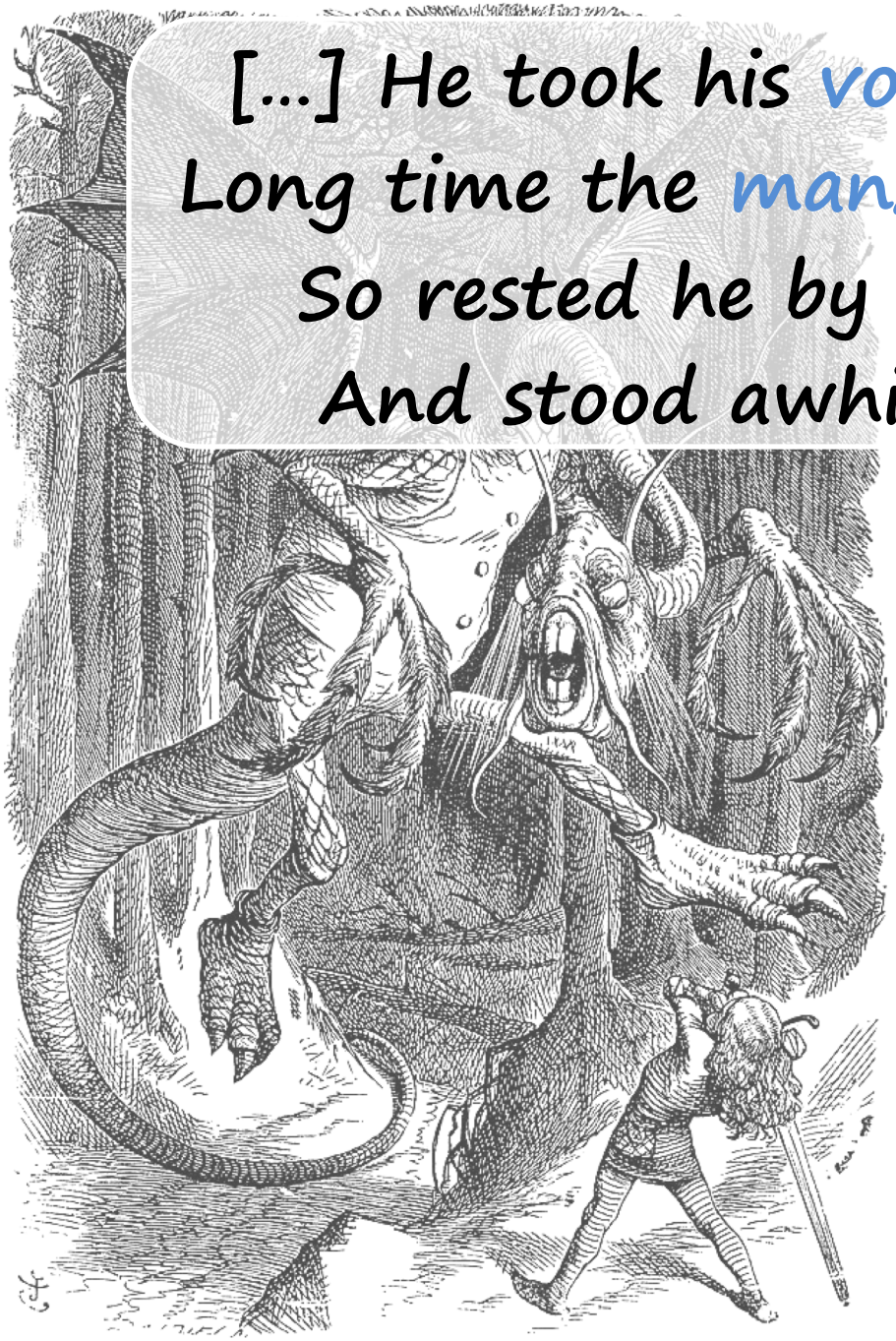
Modèle de compréhension du langage

« La bicyclette est bleue »



- Et maintenant, comment faire si l'on veut apprendre à notre réseau de neurones à comprendre des phrases ?

[...] He took his vorpal sword in hand:
Long time the manxome foe he sought—
So rested he by the Tumtum tree,
And stood awhile in thought. [...]

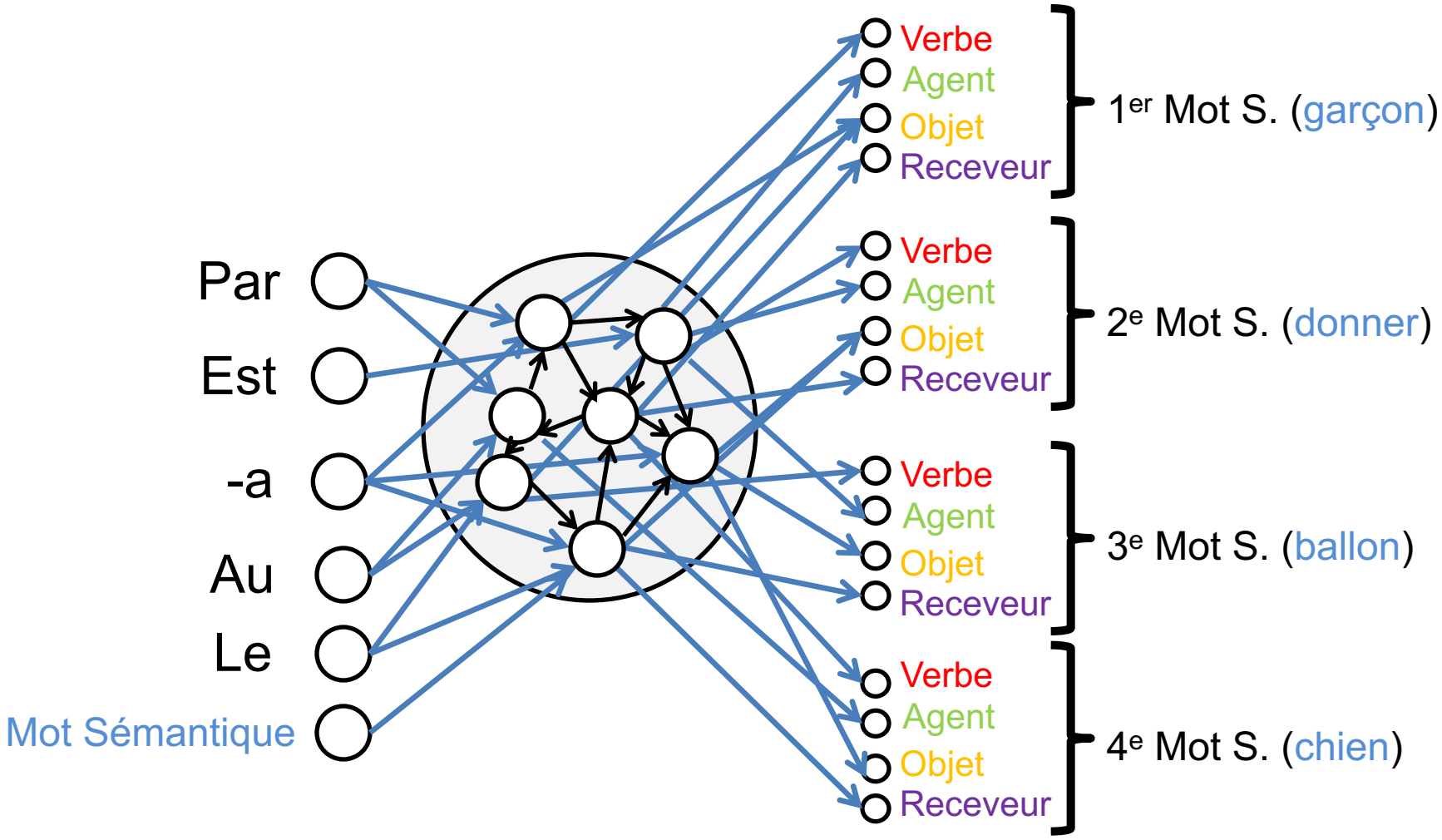


[…] He took his vorpal sword in hand:
Long time the manxome foe he sought—
So rested he by the Tumtum tree,
And stood awhile in thought. […]

Jabberwocky, Lewis Carroll
Through the Looking-Glass,
and What Alice Found There (1871)

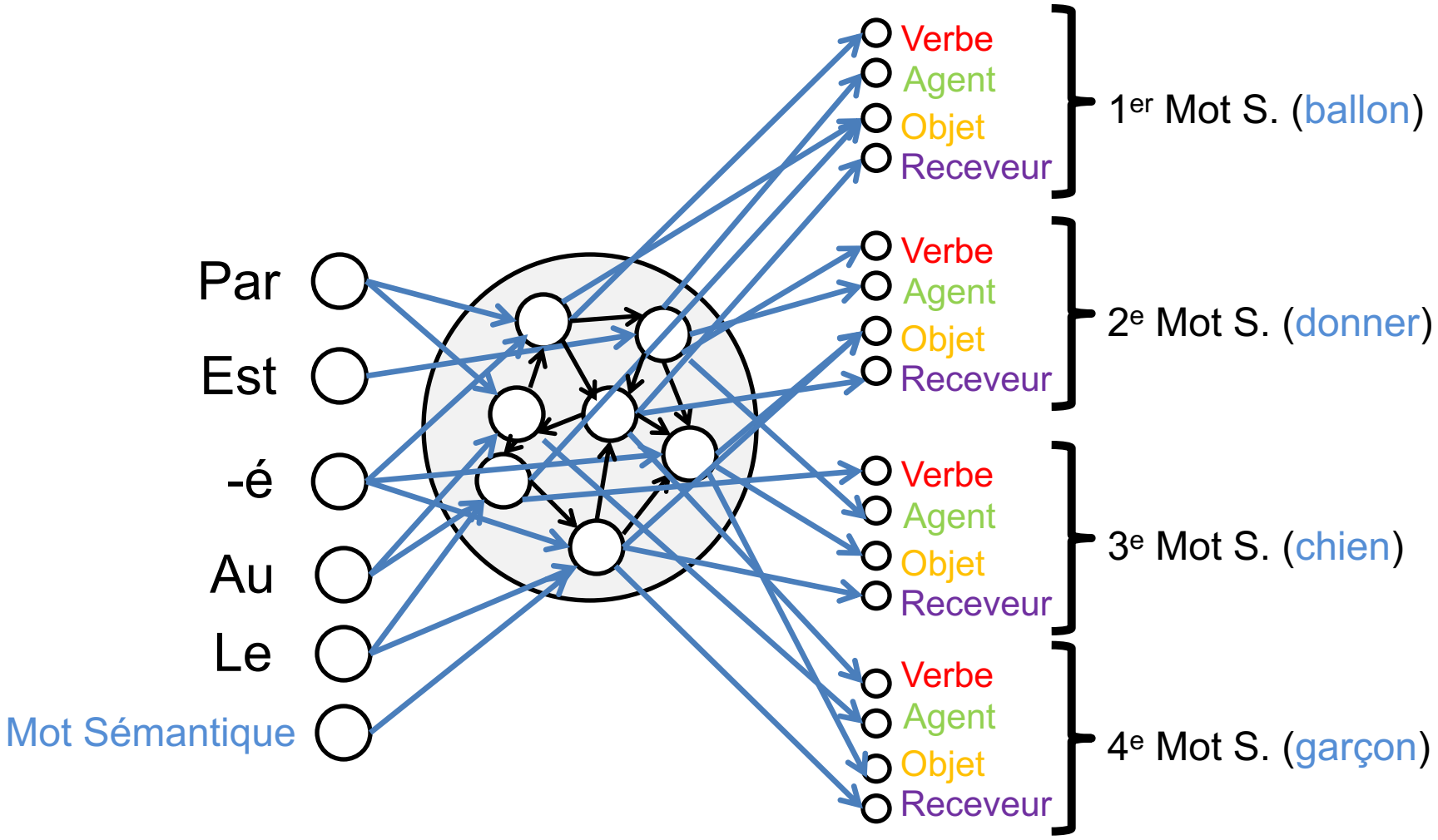
Phase d'apprentissage

« Le garçon donn-a le ballon au chien »

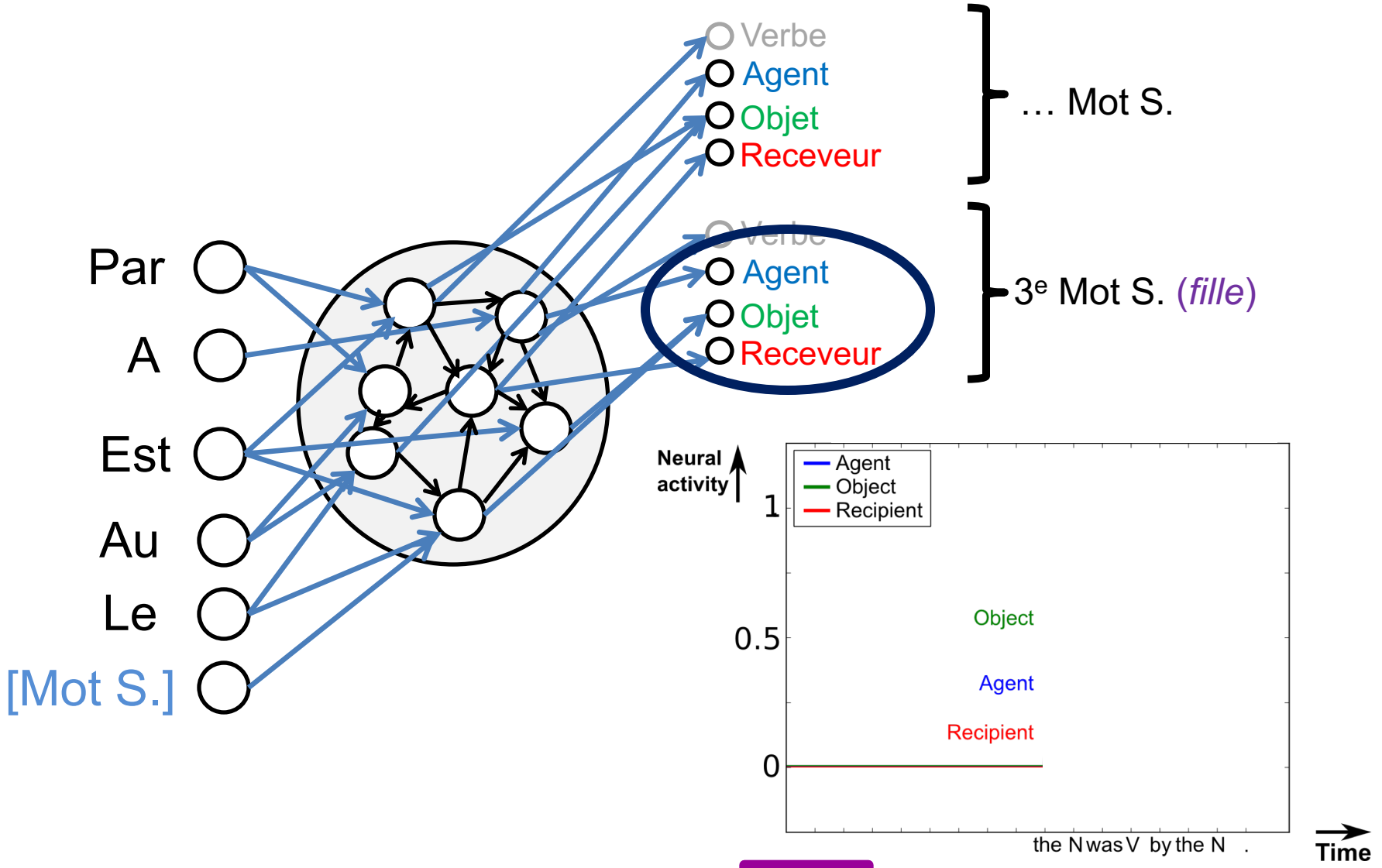


Phase d'apprentissage

« Le ballon est donn-é au chien par le garçon »



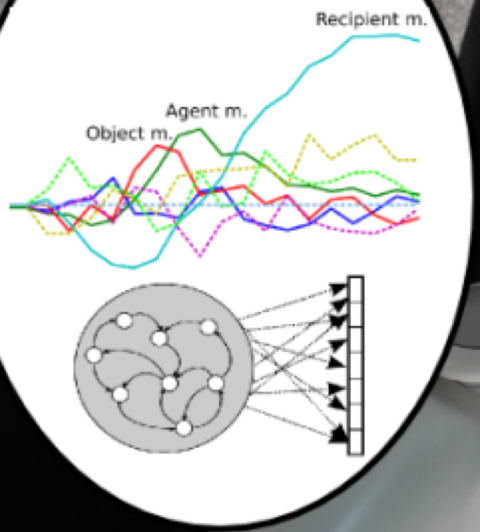
Phase de test



« Le ***chat*** a été ***nourri*** par la ***fille*** »

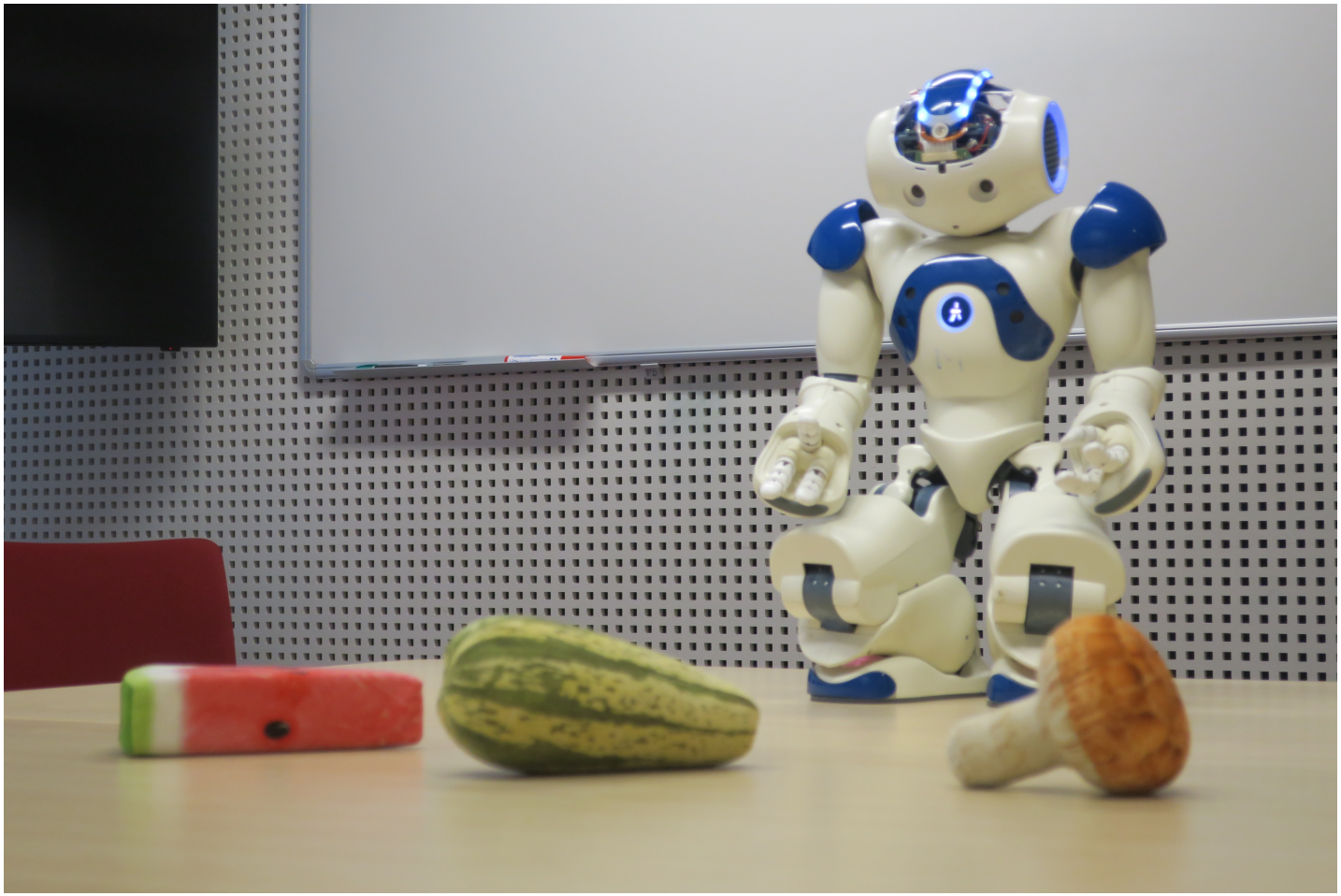
有(一)天鱼裏，那些活济：的猶子
在衛边儿儘着跌儘着兎。
好難四儿啊，那些鷓鴣鴿子
还有家的猪子慪得格儿。

'Twas brillig, and the slithy toves
Did gyre and gimble in the wabe.
All mimsy were the borogoves
And the mome raths outgrabe.

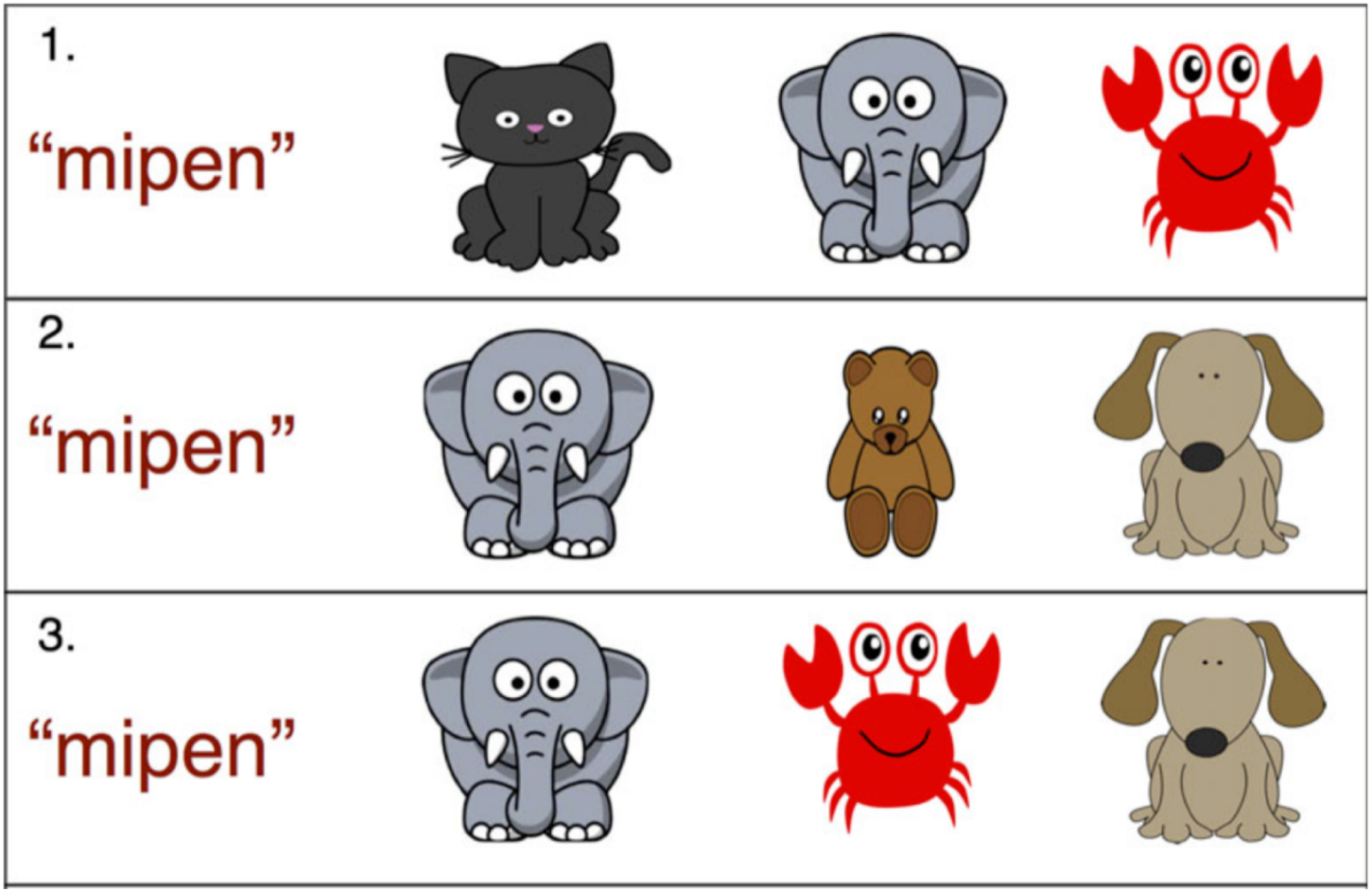


https://youtu.be/AUbJAupkU4M

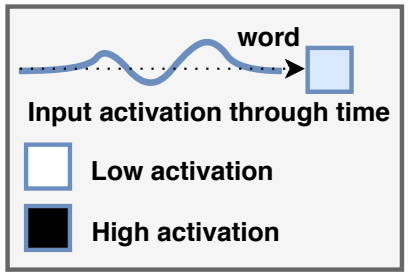
Apprendre le langage à un robot ?



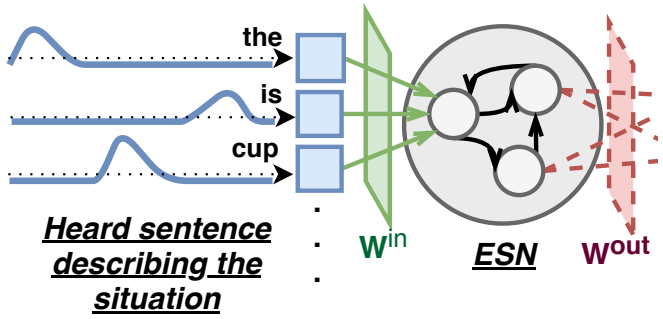
Cross-situational learning experiment

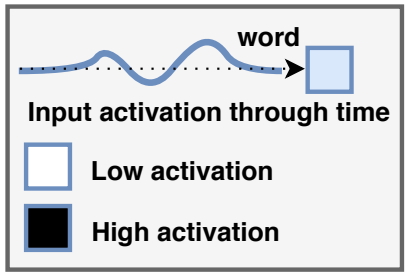


Stevens et al. 2017

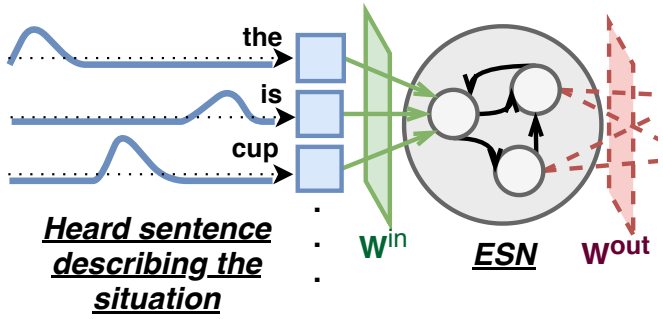


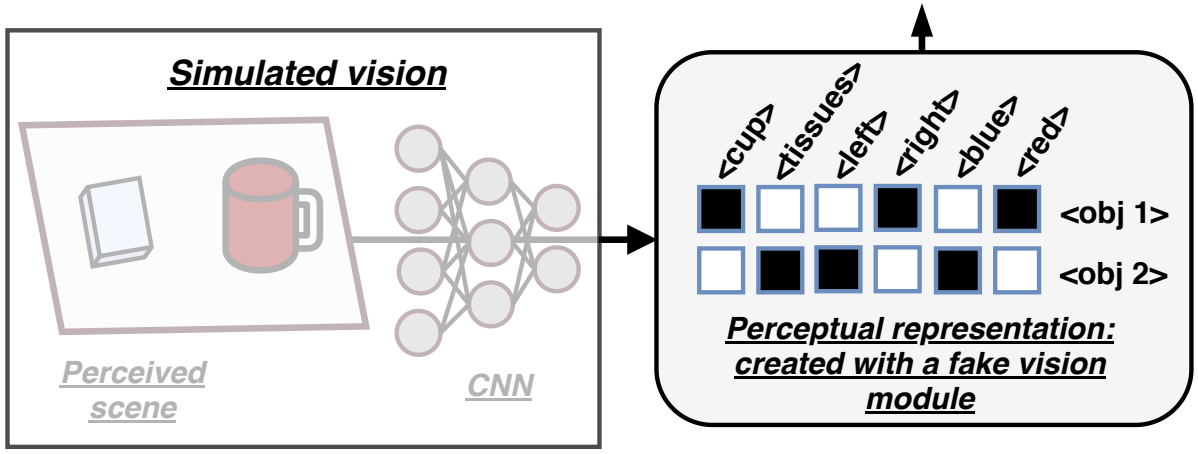
"The cup is on the right"

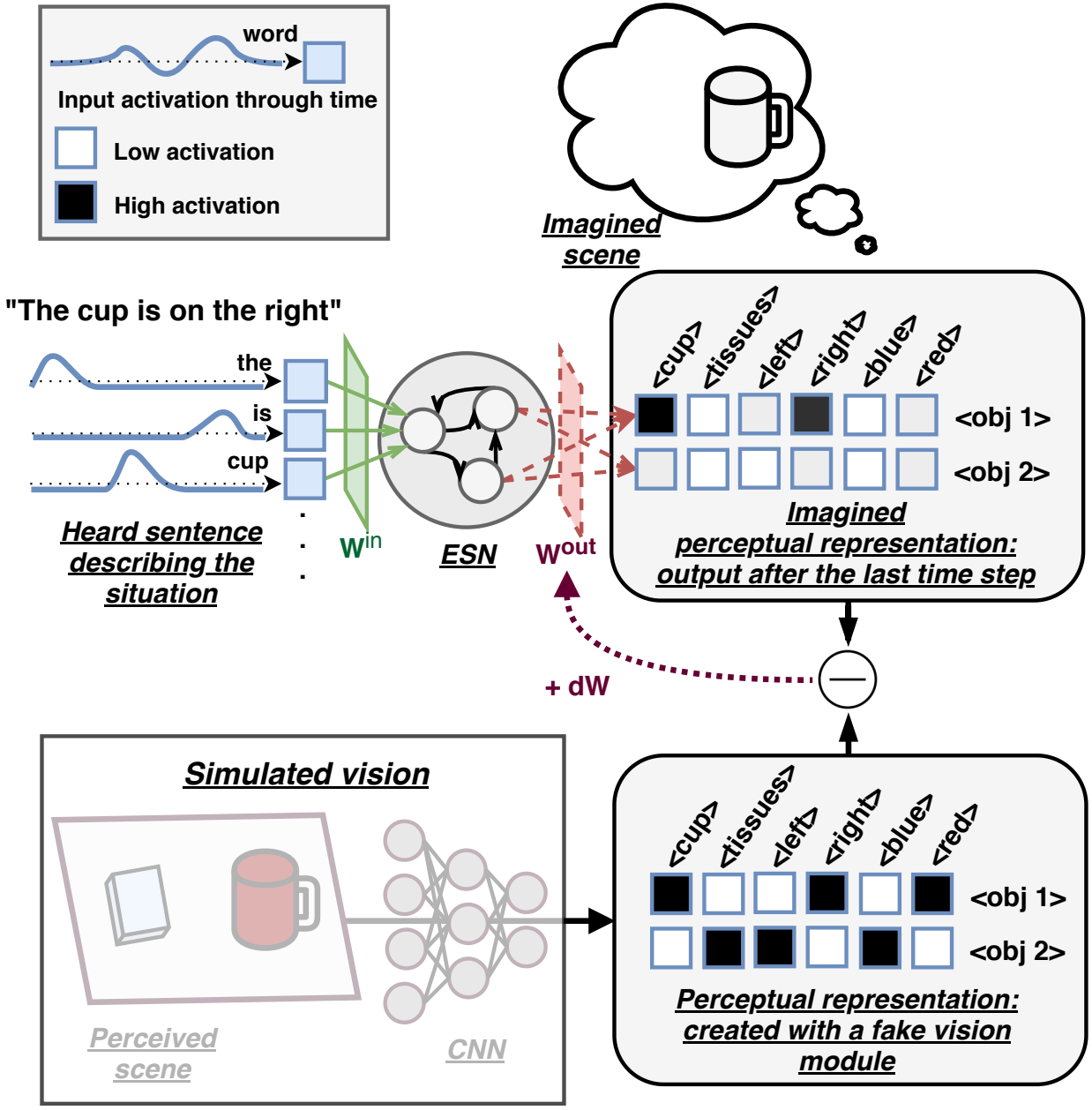




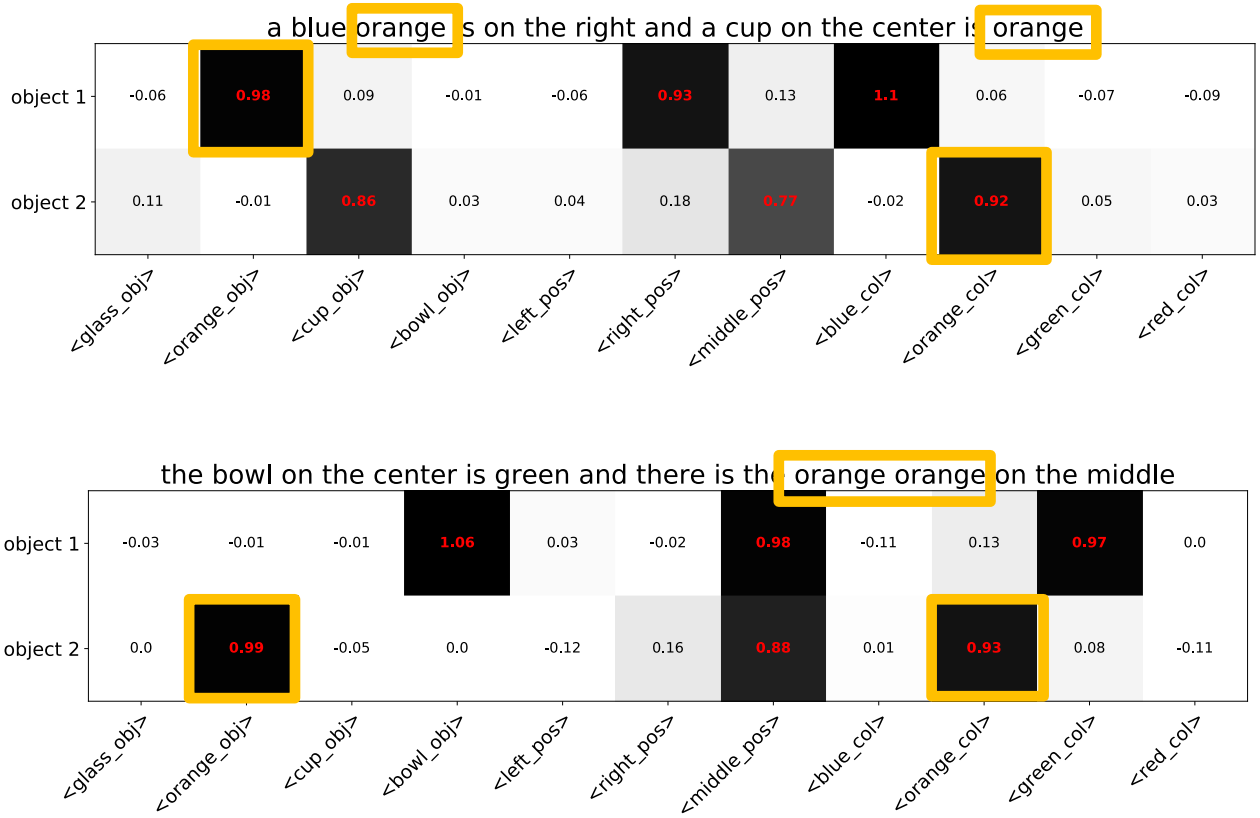
"The cup is on the right"



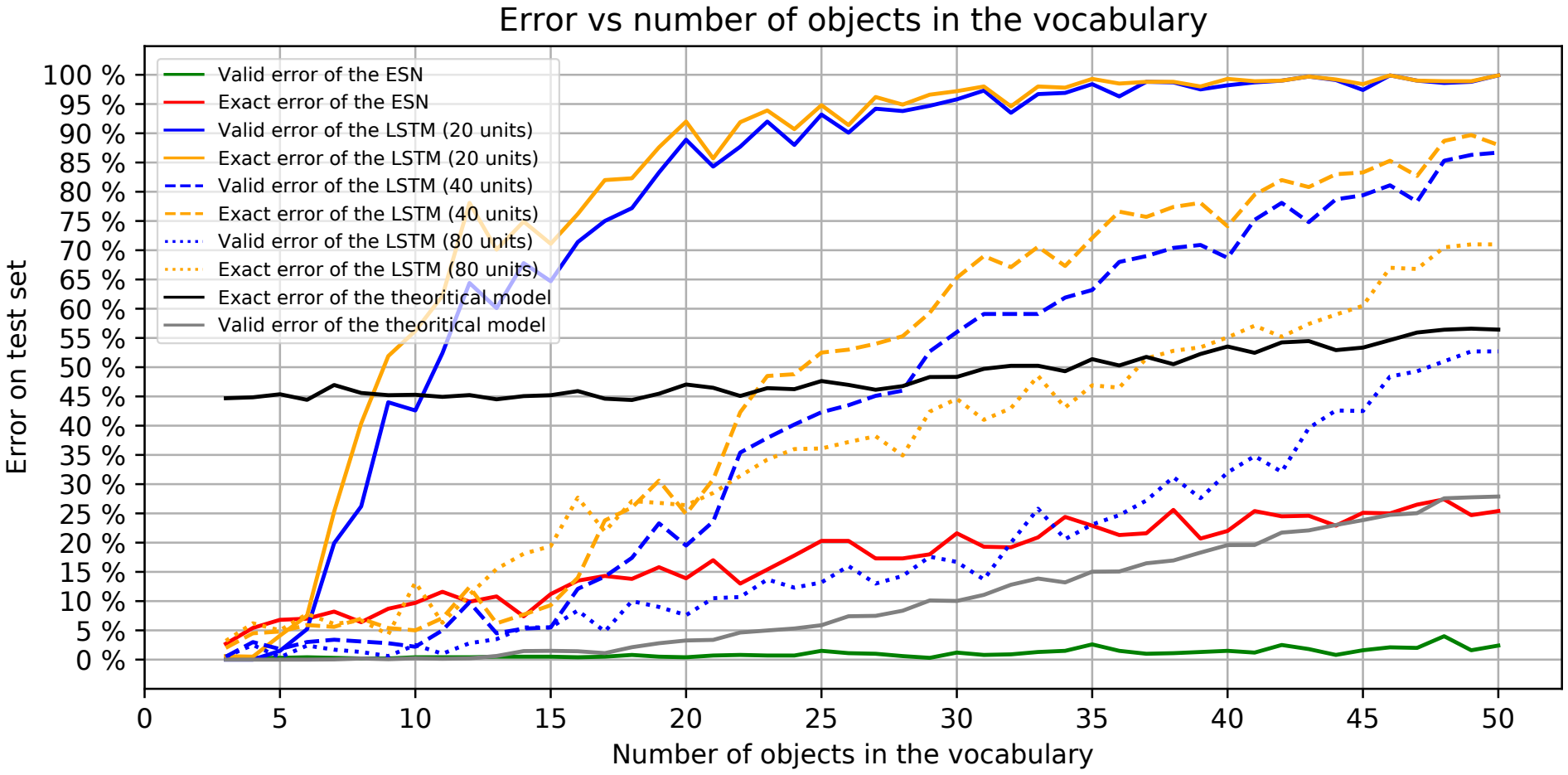



word
Input activation through time
Low activation
High activation
Imagined scene
"The cup is on the right"
the
is
cup
Heard sentence describing the situation
Win
ESN
Wout
<cup>
<tissues>
<left>
<right>
<blue>
<red>
<obj 1>
<obj 2>
Imagined perceptual representation: output after the last time step
+ dW
Simulated vision
Perceived scene
CNN
Perceptual representation: created with a fake vision module

Learning polysemous words



Increasing size of corpus: Reservoirs vs. LSTMs

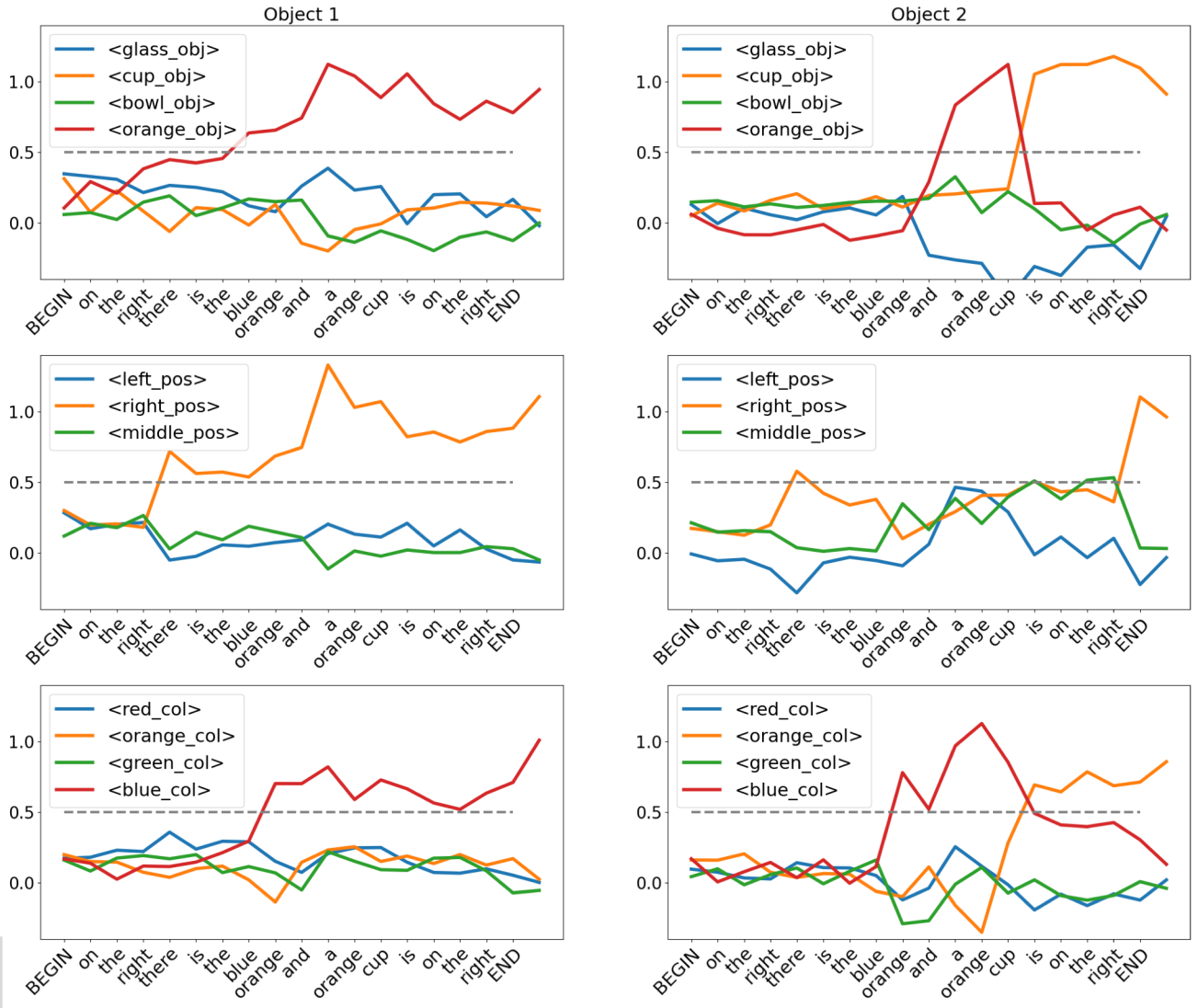


→ Les reservoirs généralisent mieux que les LSTM !

Variengien & Hinaut (preprint 2020)

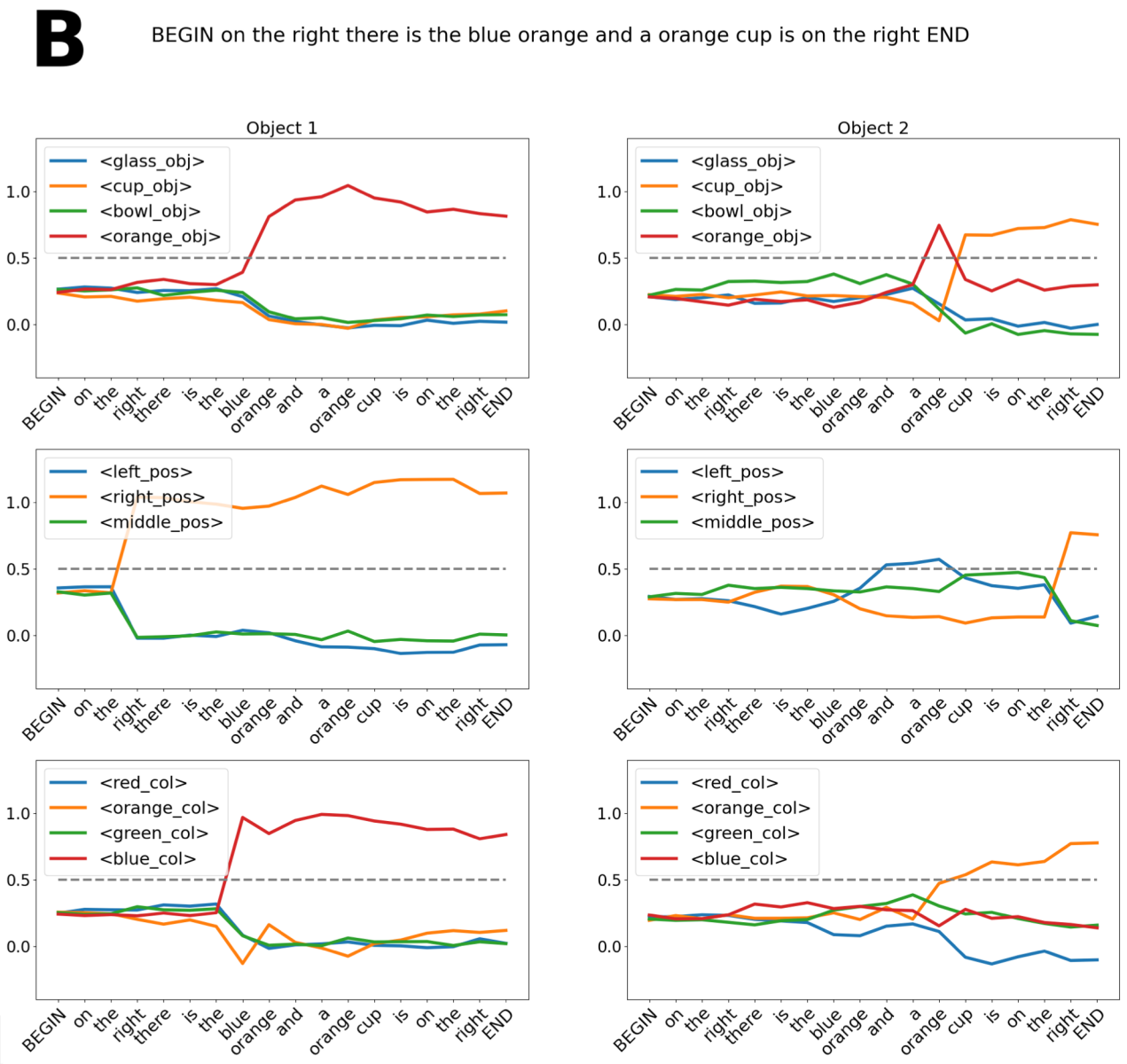
LSTM outputs

BEGIN on the right there is the blue orange and a orange cup is on the right END



Variengien & Hinaut (preprint 2020)

Reservoir outputs



Variengien & Hinaut (preprint 2020)

Collaborators

- **(Lyon)**

P.F. Dominey
M. Petit
…

- **Hamburg**

S. Wermter
J. Twiefel
L. Mici
S. Magg
…

- **Paris**

C. del Negro
A. Cazala
...

- **Tokyo**

M. Spranger

- **Bordeaux**

S. Pagliarini
A. Strock
F. Alexandre
N. Rougier
A. Leblois
A. Juven
L. Pedrelli
N. Trouvain
S. Oota
G. Passault
...

github.com/neuronalX