

Exploration de la notion de méta-apprentissage

Dans quelle mesure un système apprenant peut « prendre conscience » de ses performances et altérer son comportement ?

Cortex / Maia

Yann Boniface
Alain Dutech
Nicolas Rougier

Matthieu Zimmer

À quoi sert le méta-apprentissage et les méta-représentations ?

- Évaluer les connaissances
- Améliorer l'apprentissage
- Un pas possible vers un début de conscience

« Higher-Order Thought theory », David Rosenthal

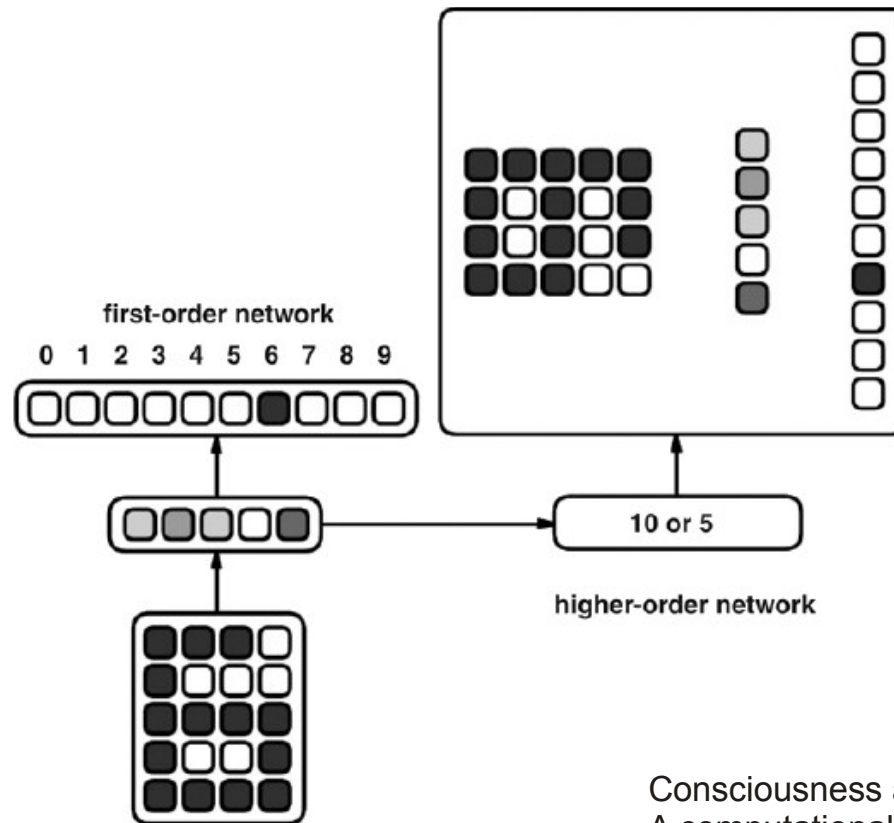
Inspiration : Conscience et méta-représentations

- Consciousness and metarepresentation : A computational sketch
[Alex Cleeremans, Bert Timmermans, Antoine Pasquali]
- Know thyself : Metacognitive networks and measures of consciousness
[Antoine Pasquali, Bert Timmermans, Alex Cleeremans]

Simulation 1 : Perceptron multicouche

2 perceptrons multicouche

- Matrice 5x4 entrées représentant les 10 chiffres
- Le premier réseau discrimine les 10 chiffres
- Winner-take-all sur les sorties



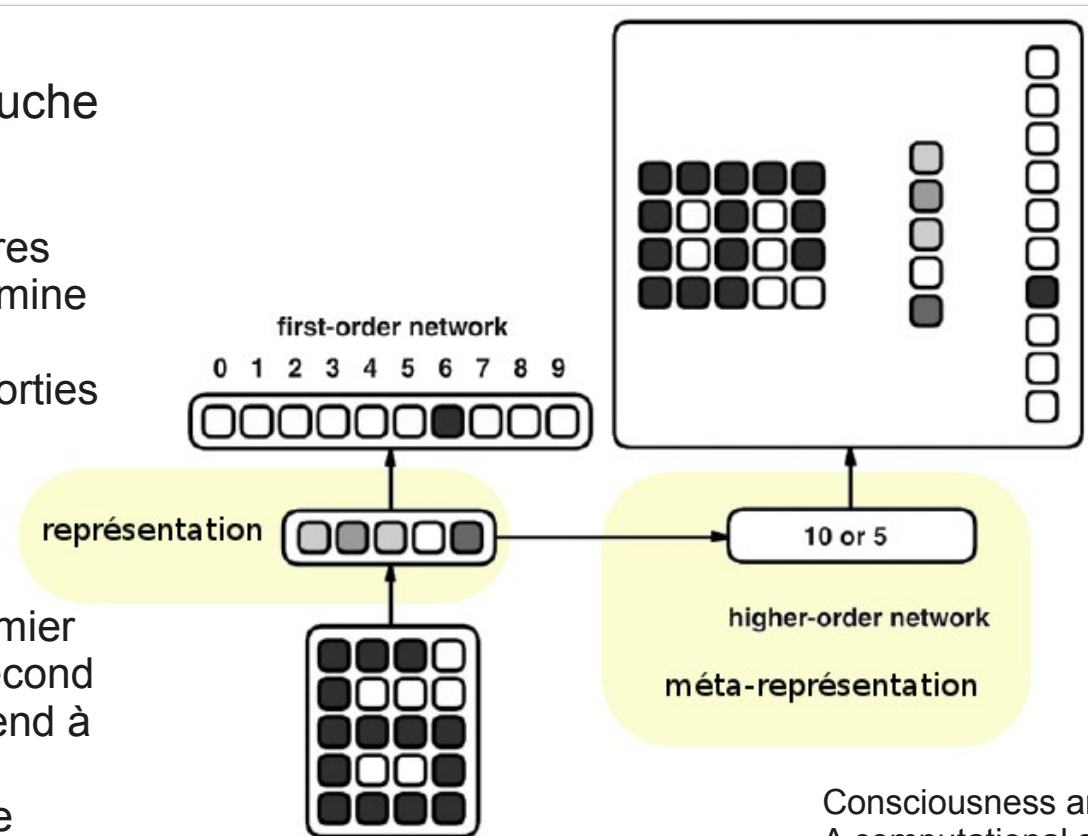
- La couche caché du premier réseau est l'entrée du second
- Le second réseau apprend à redonner l'état entier du premier (entrée / couche caché / sortie)

Consciousness and metarepresentation :
A computational sketch
[Alex Cleeremans, Bert Timmermans, Antoine Pasquali]

Simulation 1 : Perceptron multicouche

2 perceptrons multicouche

- Matrice 5x4 entrées représentant les 10 chiffres
- Le premier réseau discrimine les 10 chiffres
- Winner-take-all sur les sorties

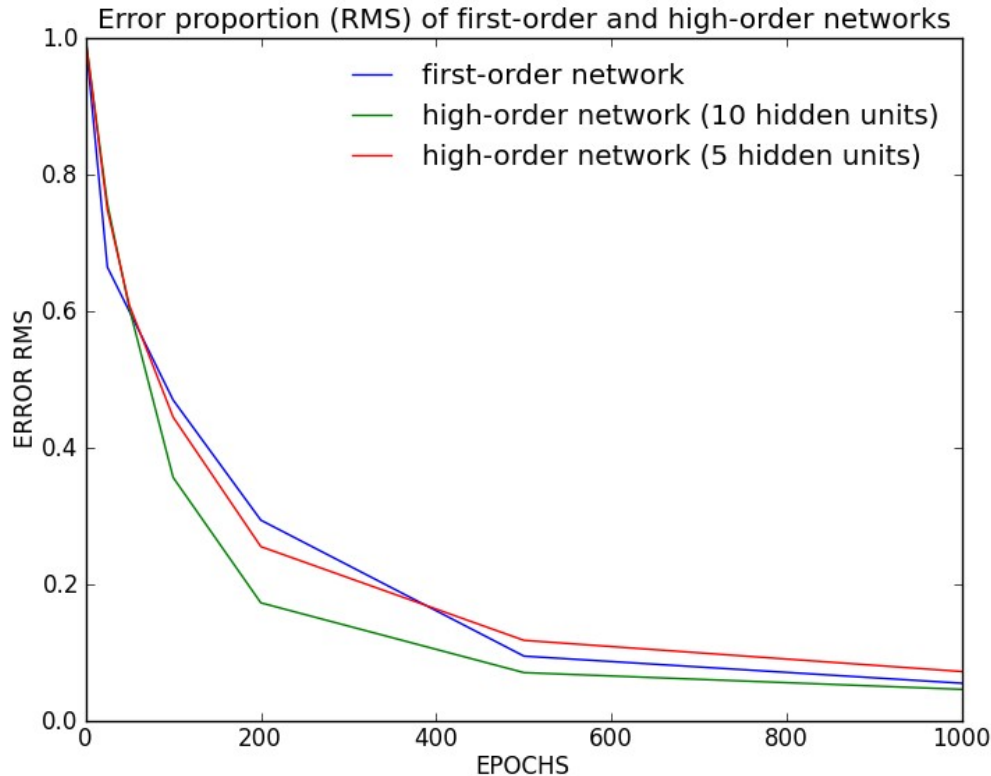


- La couche caché du premier réseau est l'entrée du second
- Le second réseau apprend à redonner l'état entier du premier (entrée / couche caché / sortie)

Consciousness and metarepresentation :
A computational sketch
[Alex Cleeremans, Bert Timmermans, Antoine Pasquali]

Résultats sur la base d'entrée de l'article

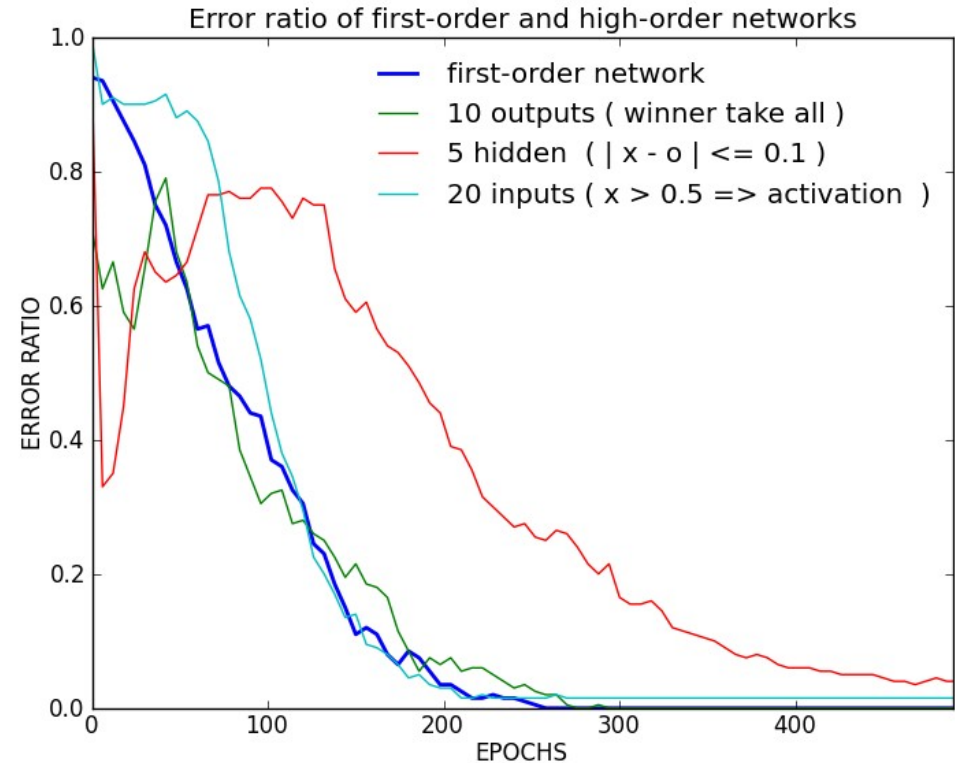
De l'article



$$rms\ proportion_e = \frac{rms_e = \sqrt{\frac{1}{n} \sum_{i=1}^n (o_{i,e} - d_i)^2}}{\max(rms_e), \forall e \in epochs}$$

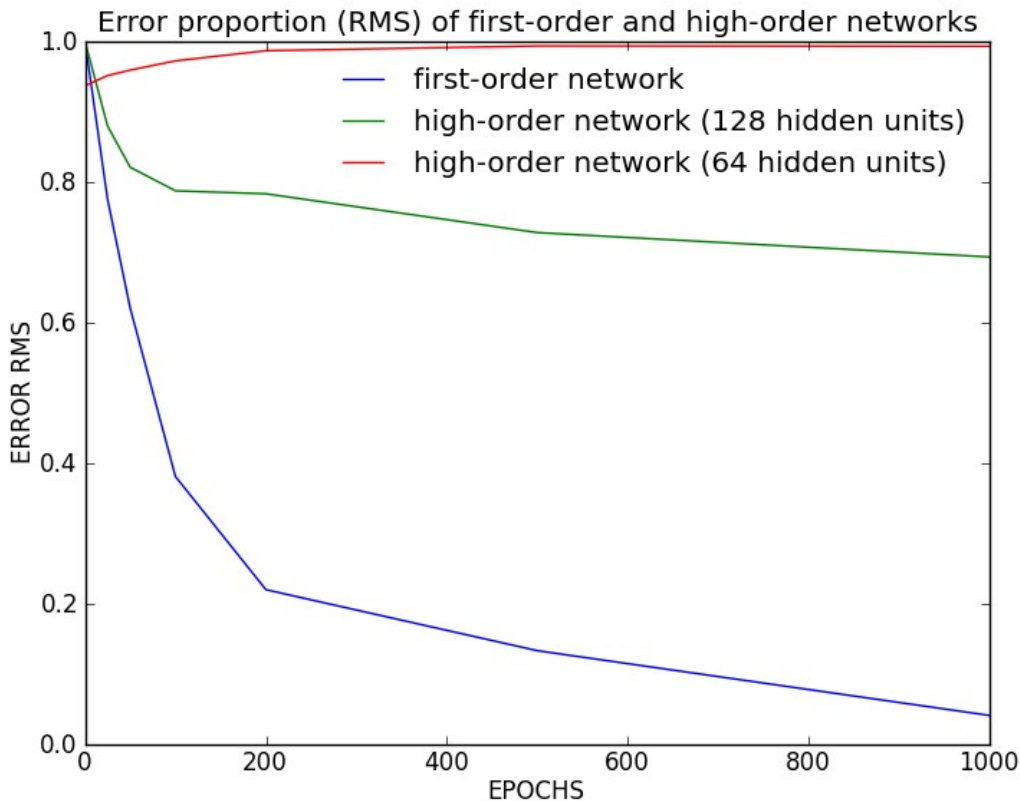
with $\begin{cases} n : \text{number of neurons on the output layer} \\ o_{i,e} : \text{value obtained for the } i^{th} \text{ neuron at the } e^{th} \text{ epoch} \\ d_i : \text{value desired for the } i^{th} \text{ neuron} \end{cases}$

Notre touche personnelle

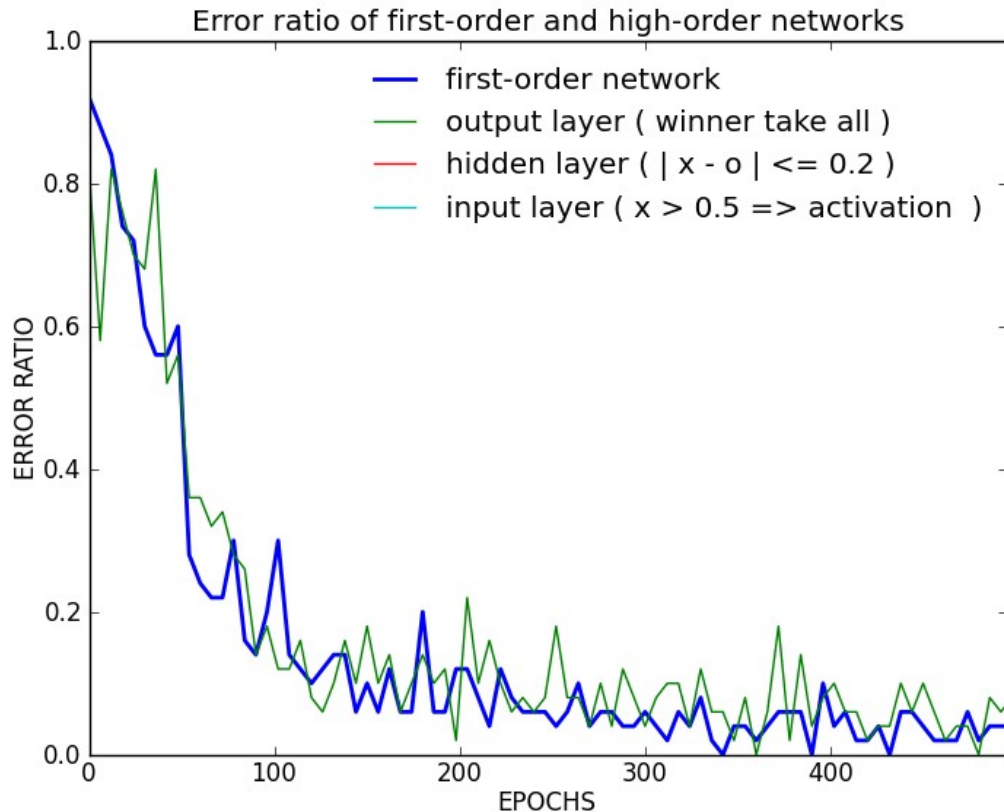


- Pourcentage d'erreur de classification
- Considère uniquement le réseau à 10 unités cachées
- Séparation des 3 couches à apprendre
- Entrées ont plus de poids

Approfondissement sur des chiffres manuscrits

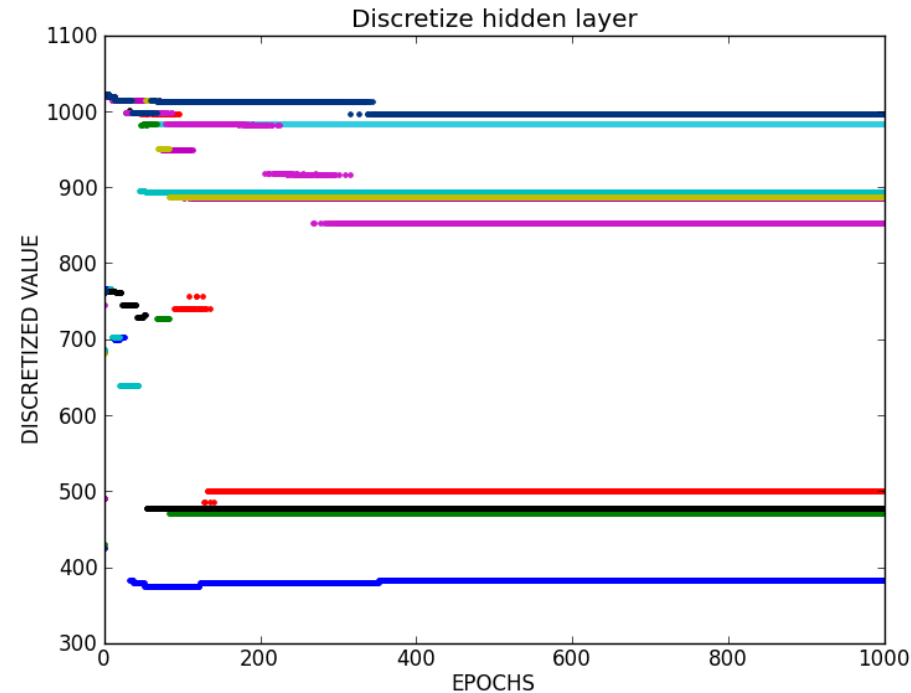
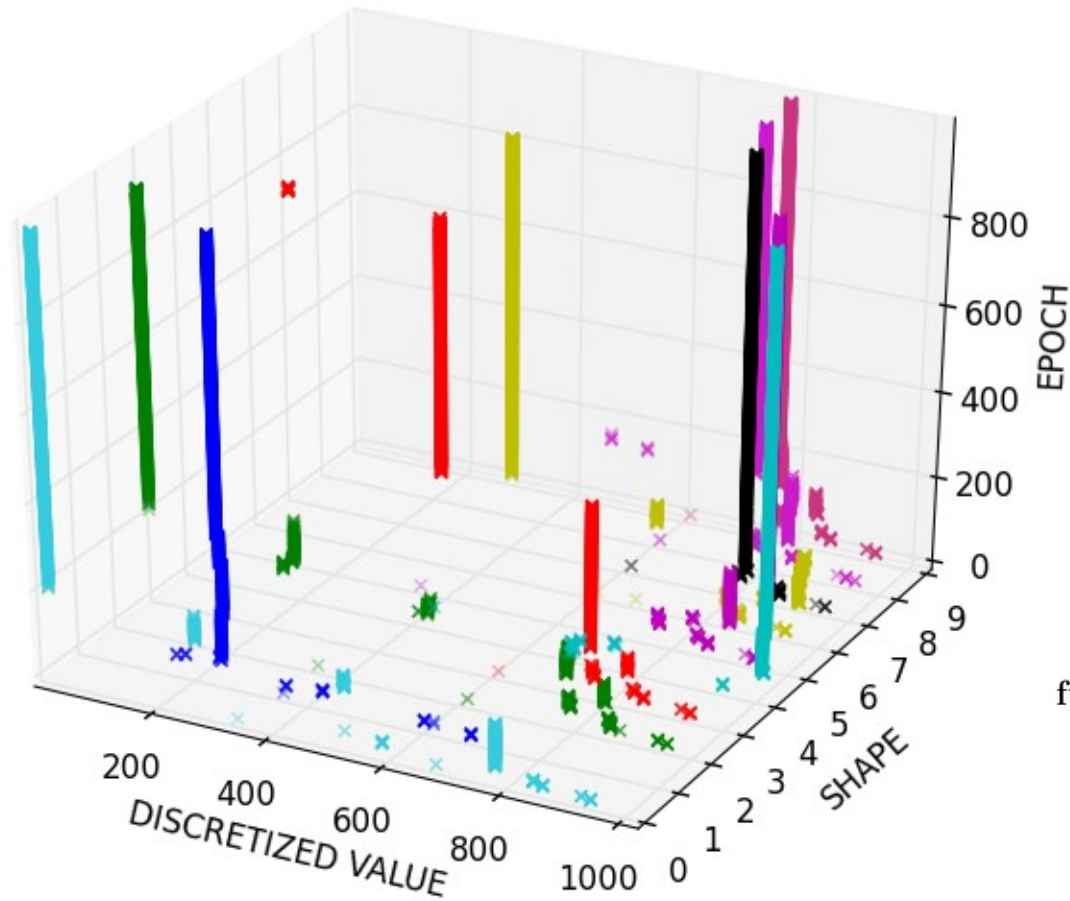


- 256 entrées
- 64 neurones cachés
- 10 sorties
- 1600 formes
- 10 formes/epoch



- 64 entrées
- 64/128 neurones cachés (low/high)
- 256 + 64 + 10 sorties

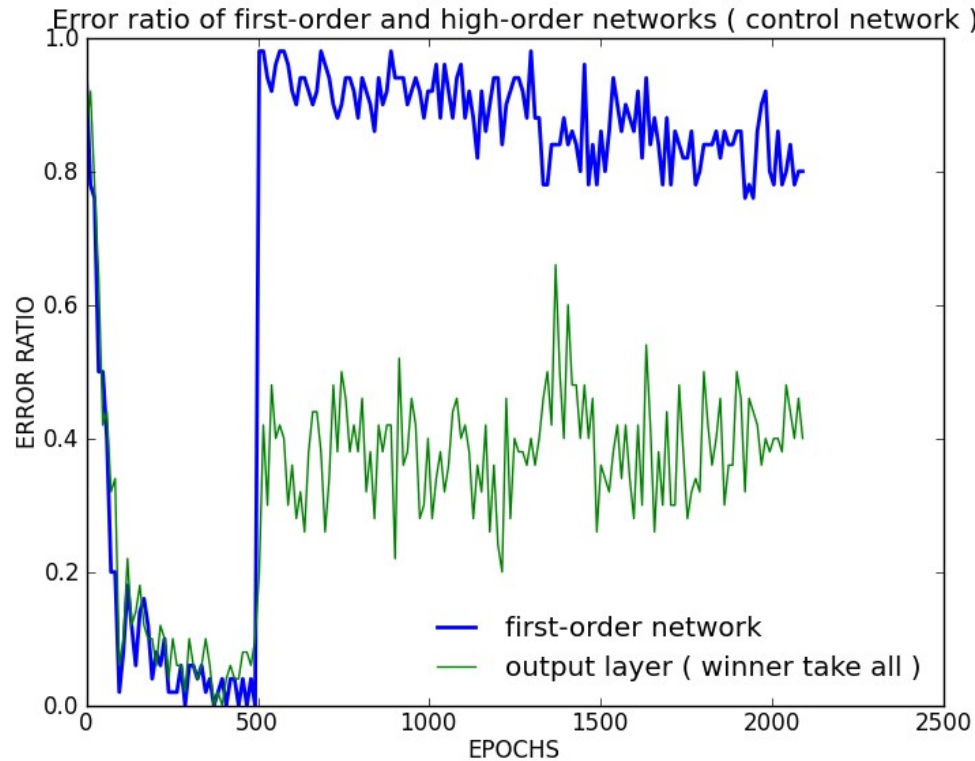
Pourquoi ça marche ?



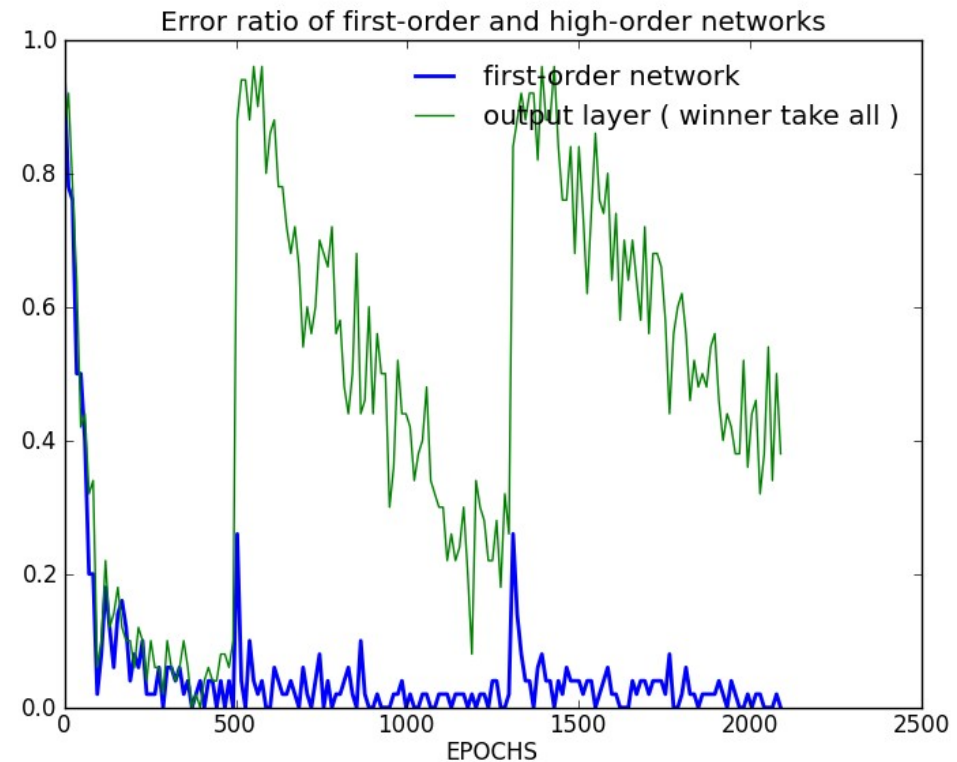
```
function DISCRETIZE(hiddenNeuron[], piece)
    result ← 0
    for i = 0 → hiddenNeuron.length do
        result ← result + piecei × cutting(hiddenNeuron[i], piece)
        i ← i + 1
    end for
    return result
end function
```


Changement de tâche

Avec chiffres manuscrits



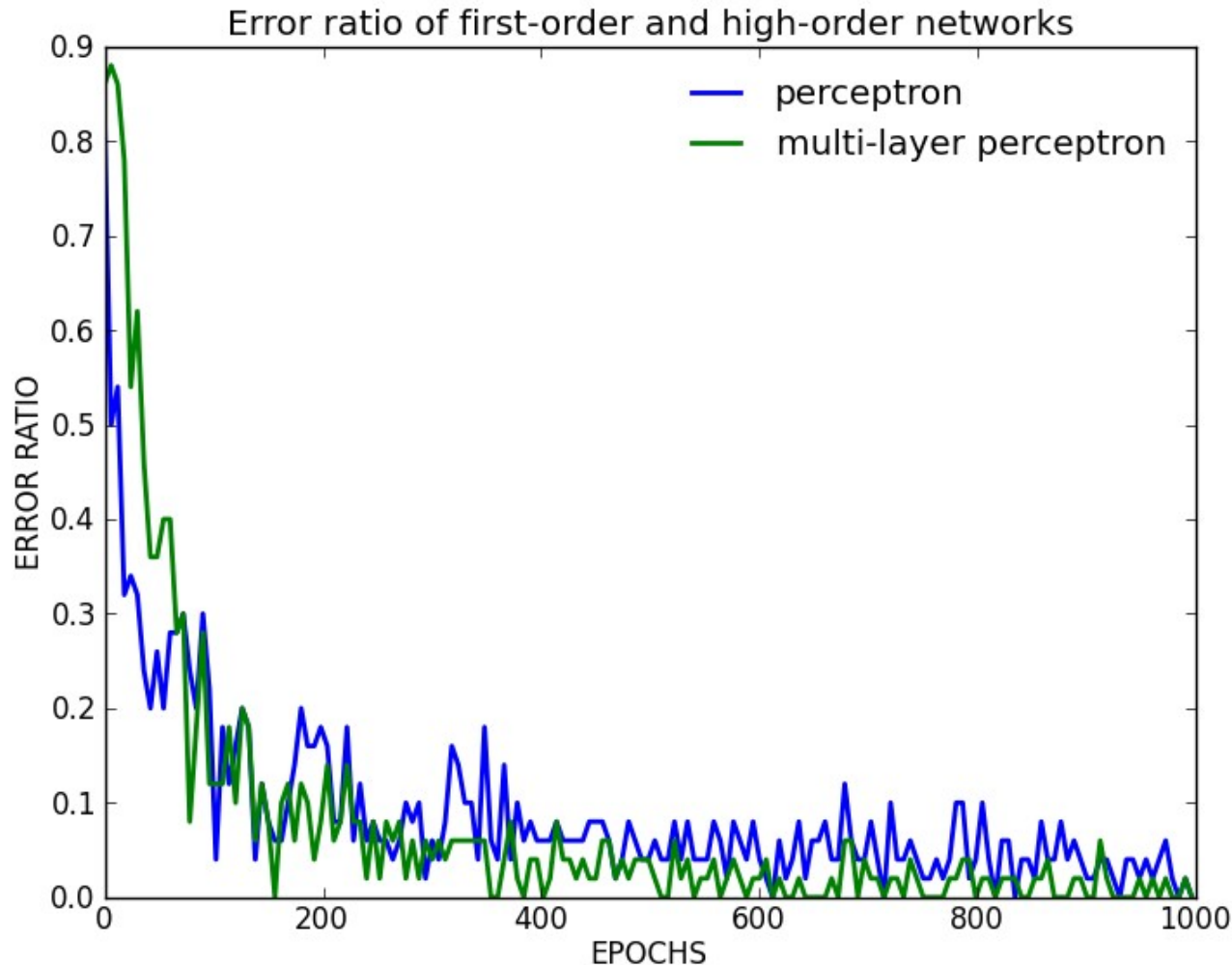
Sans blocage



Avec blocage de l'apprentissage
de la couche caché du premier réseau

Perceptron vs Multi-layer Perceptron

Avec chiffres manuscrits



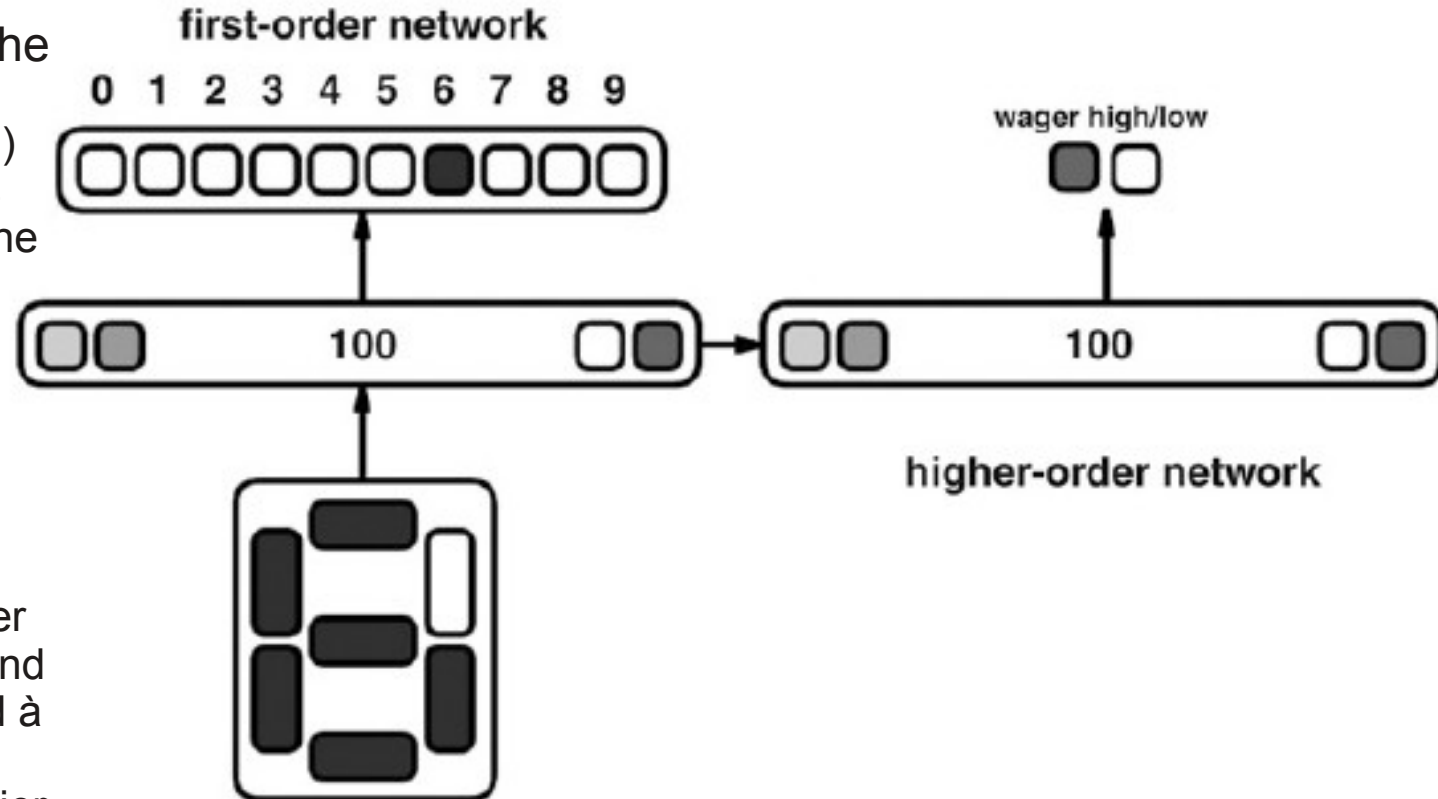
- Le perceptron apprend plus vite au début
- Le MLP a un meilleur taux de classification après 200 epochs

A-t-on besoin d'un MPL ?

Simulation 2

2 perceptrons multicouche

- 7 entrées (afficheur digital) représentant les 10 chiffres
- Le premier réseau discrimine les 10 chiffres
- Winner-take-all sur les sorties

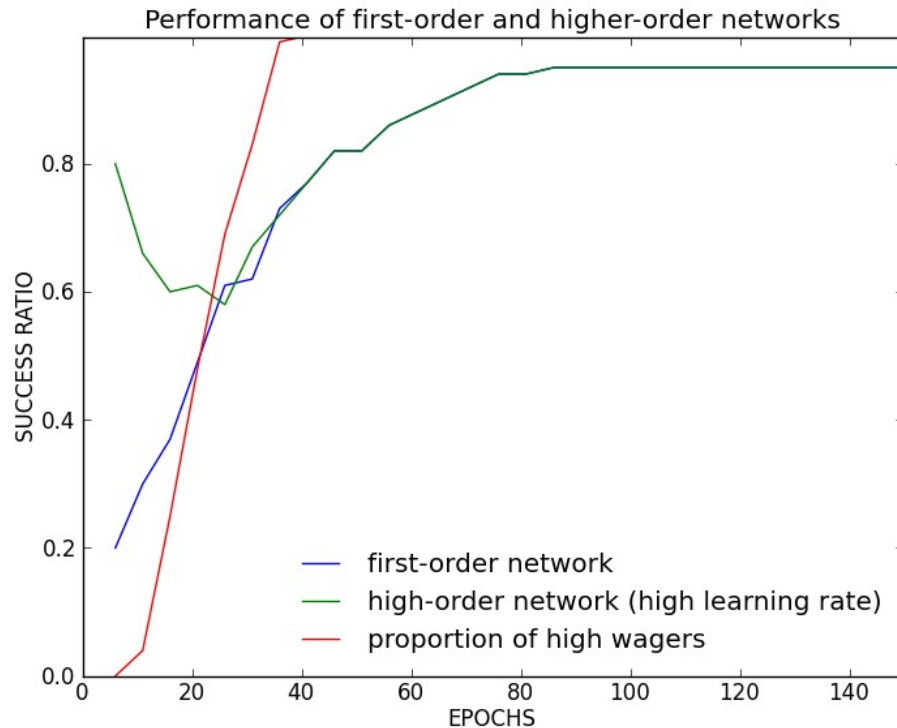


- La couche caché du premier réseau est l'entrée du second
- Le second réseau apprend à parier sur la qualité de la réponse donné par le premier

Consciousness and metarepresentation :
A computational sketch
[Alex Cleeremans, Bert Timmermans, Antoine Pasquali]

Résultats sur la base d'entrée de l'article

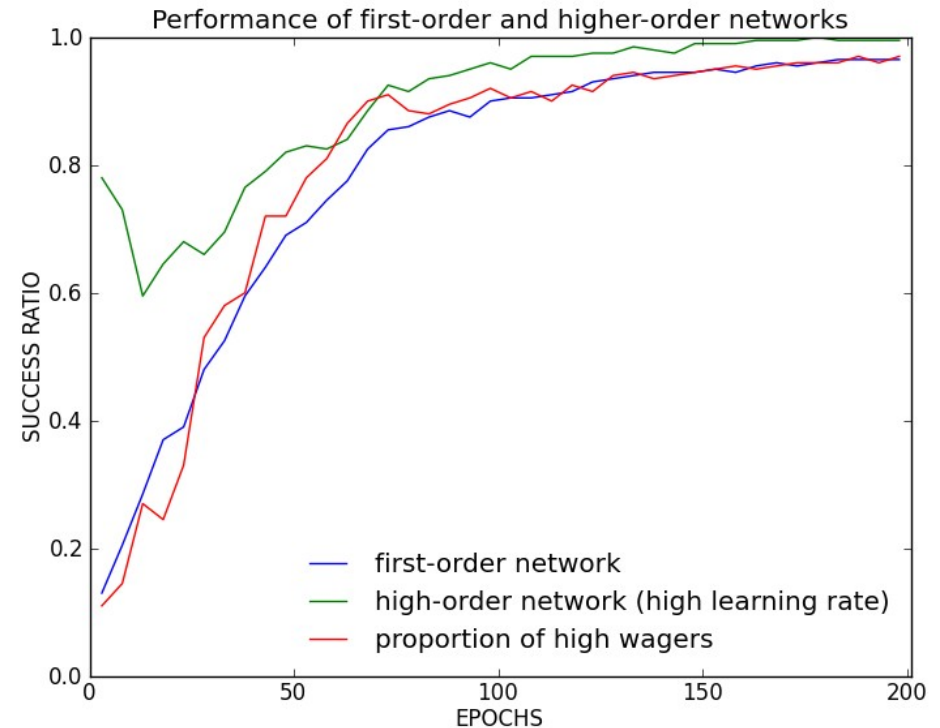
De l'article



Param. réseau supérieur :

- Poids initialisés sur $[-0,25; 0,25]$
- Momentum : 0

Notre touche personnelle

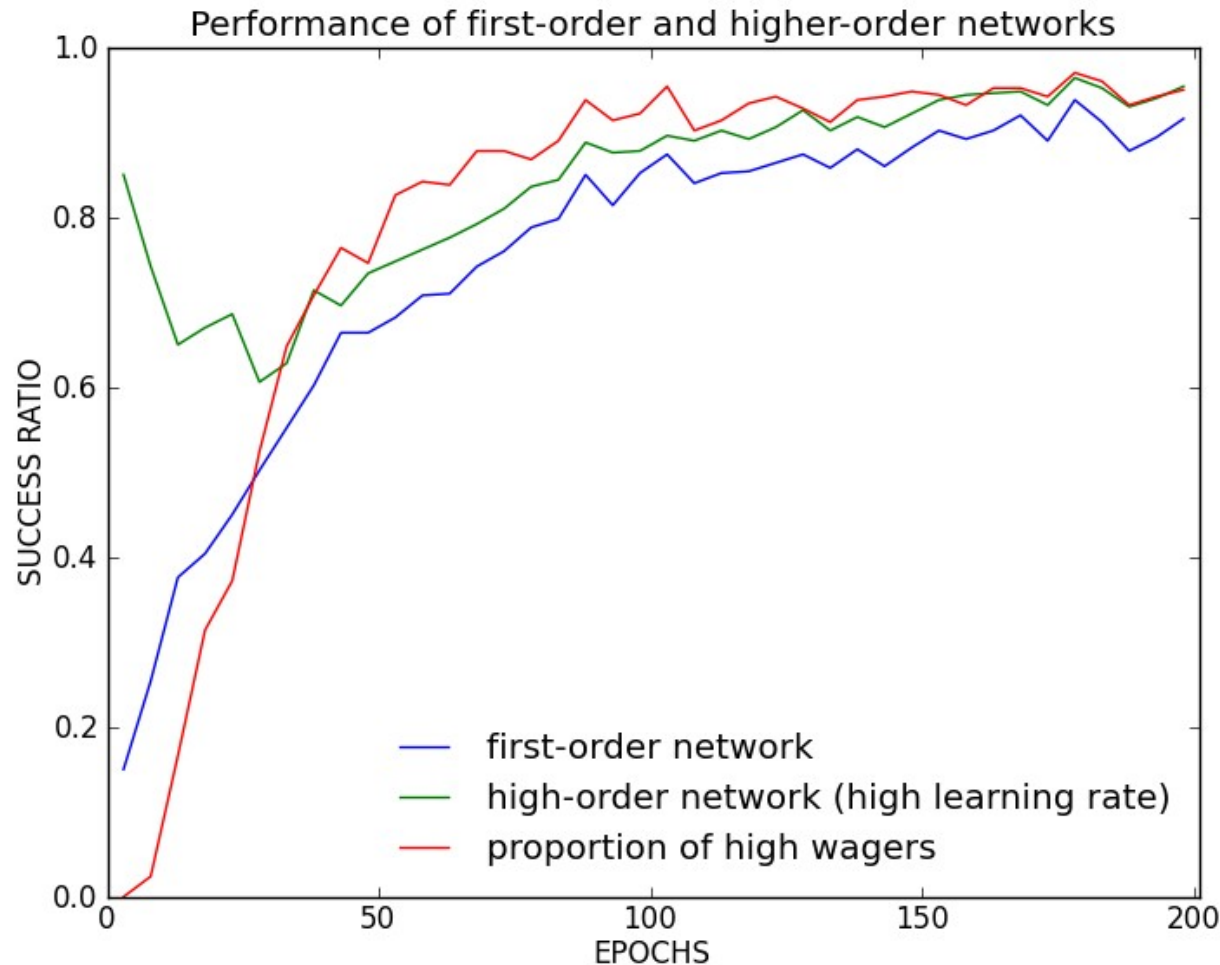


Param. réseau supérieur :

- Poids initialisés sur $[-1; 1]$
- Momentum : 0,5

(comme le premier réseau)

Approfondissement sur des chiffres manuscrits

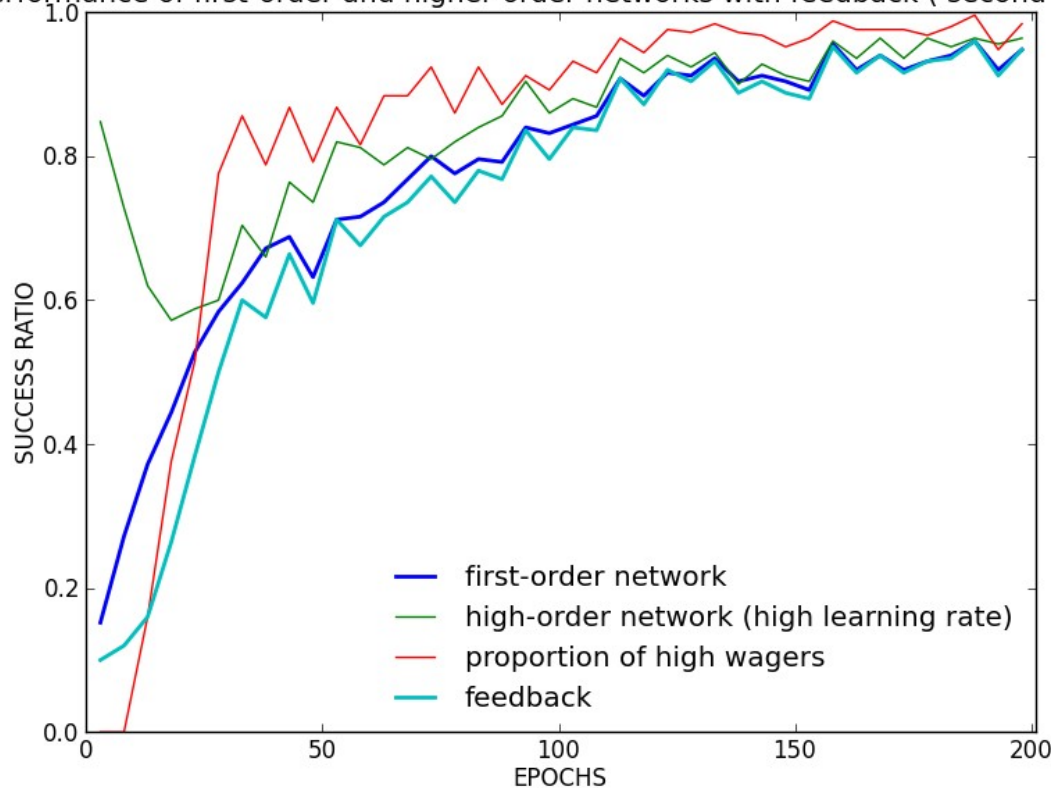


- 100 neurones cachés pour le premier réseau
- 20 neurones cachés pour le second réseau
- Taux de paris hauts trop élevé

Feedback : second neurone le plus actif

Avec chiffres manuscrits

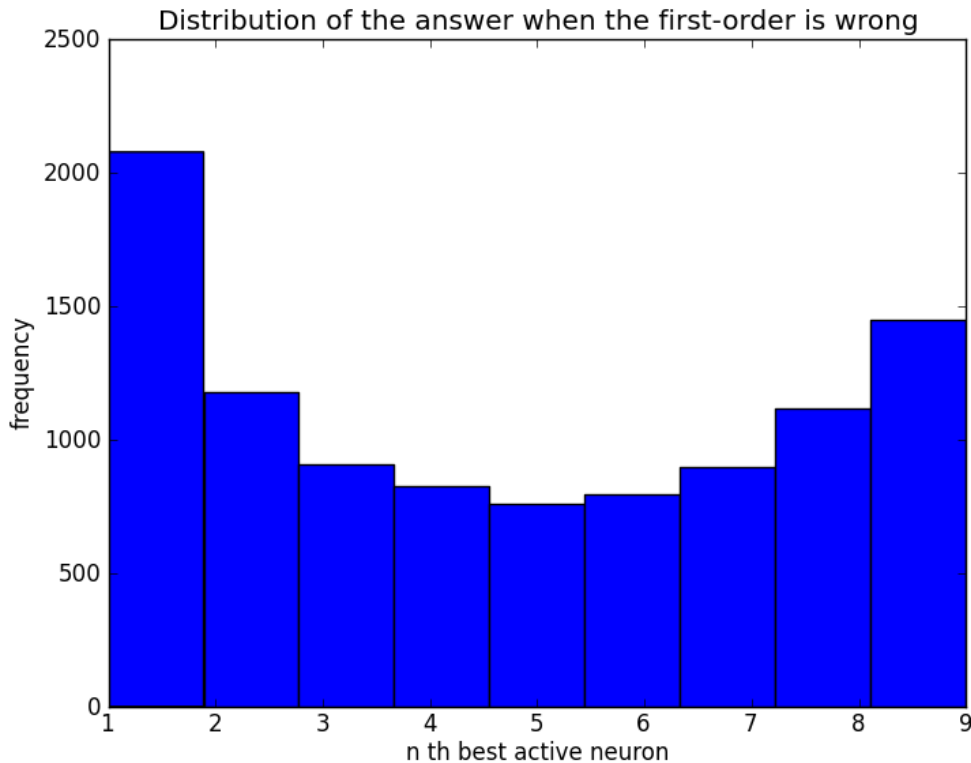
Performance of first-order and higher-order networks with feedback (second W-T-A)



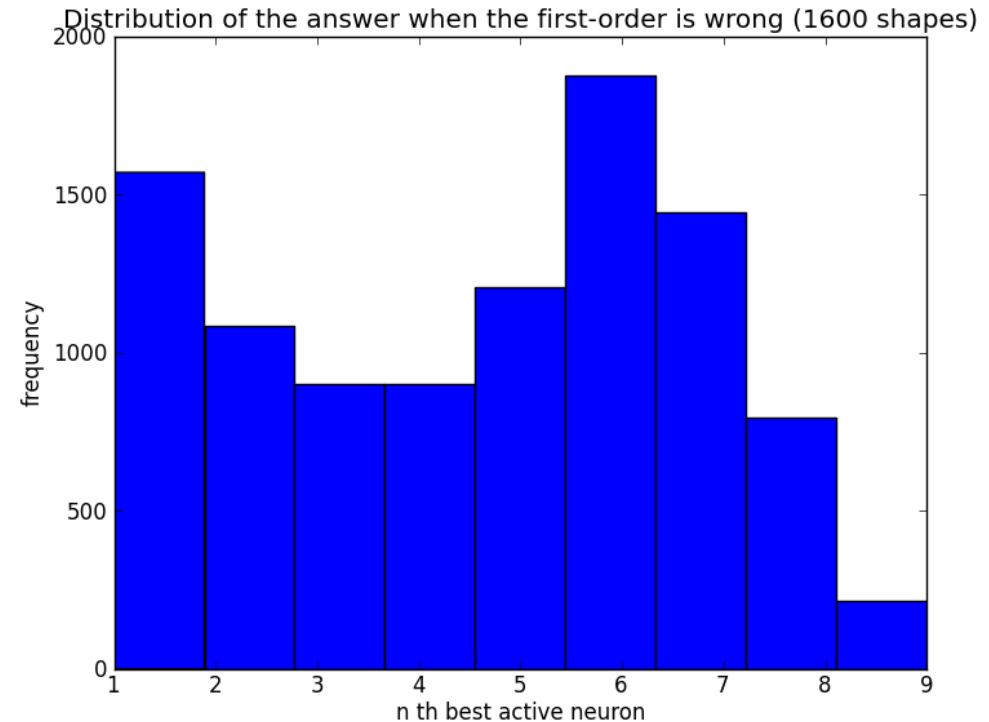
Conclusion :
la réponse ne se trouve pas toujours
dans le second neurone le plus actif

Emplacement de la bonne réponse

Avec chiffres manuscrits



Durant tout l'apprentissage
(50 formes * 200 epochs)

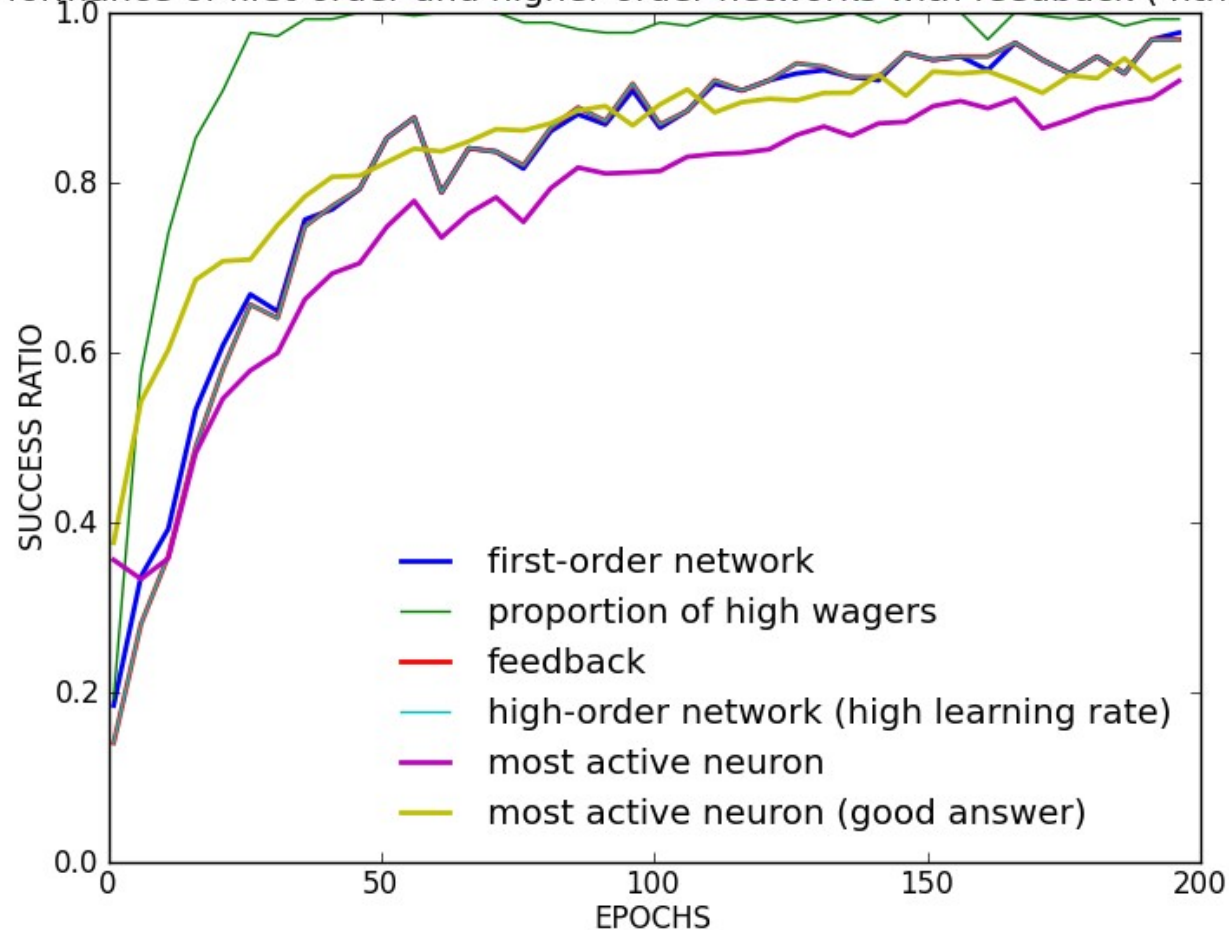


Après l'apprentissage sur
les 1600 formes

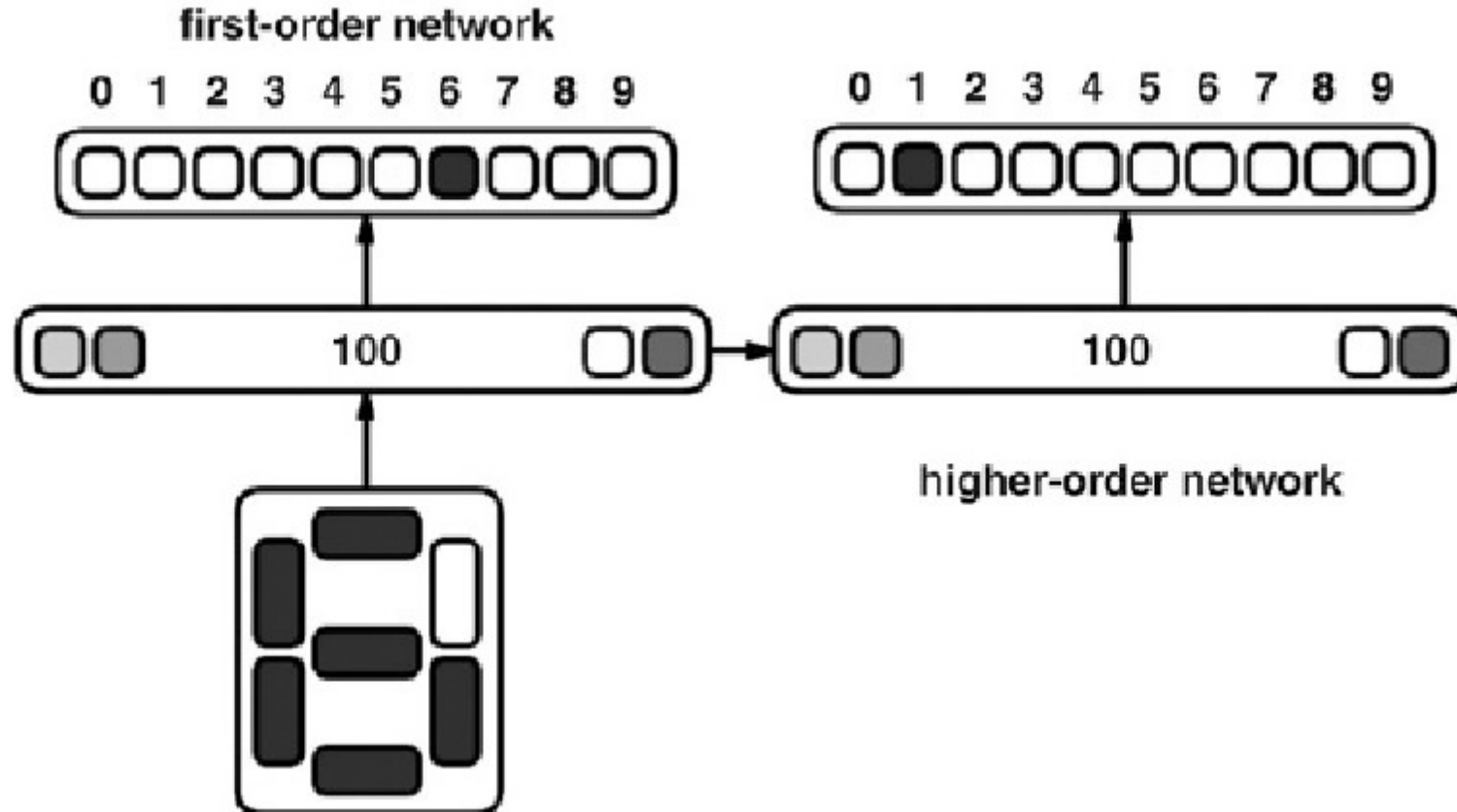
Valeur du neurone maximal

Avec chiffres manuscrits

Performance of first-order and higher-order networks with feedback (nth W-T-A



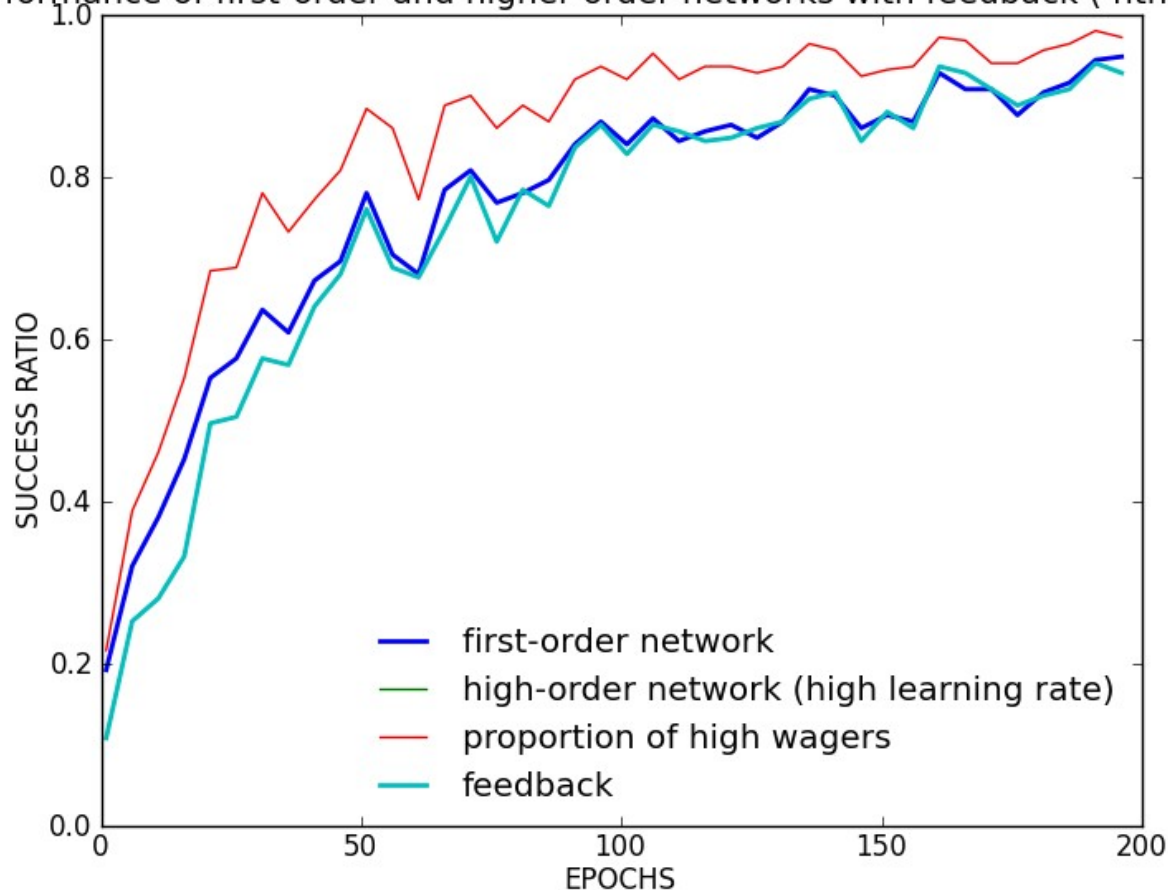
Feedback : n-ième neurone le plus actif



Feedback : n-ième neurone le plus actif

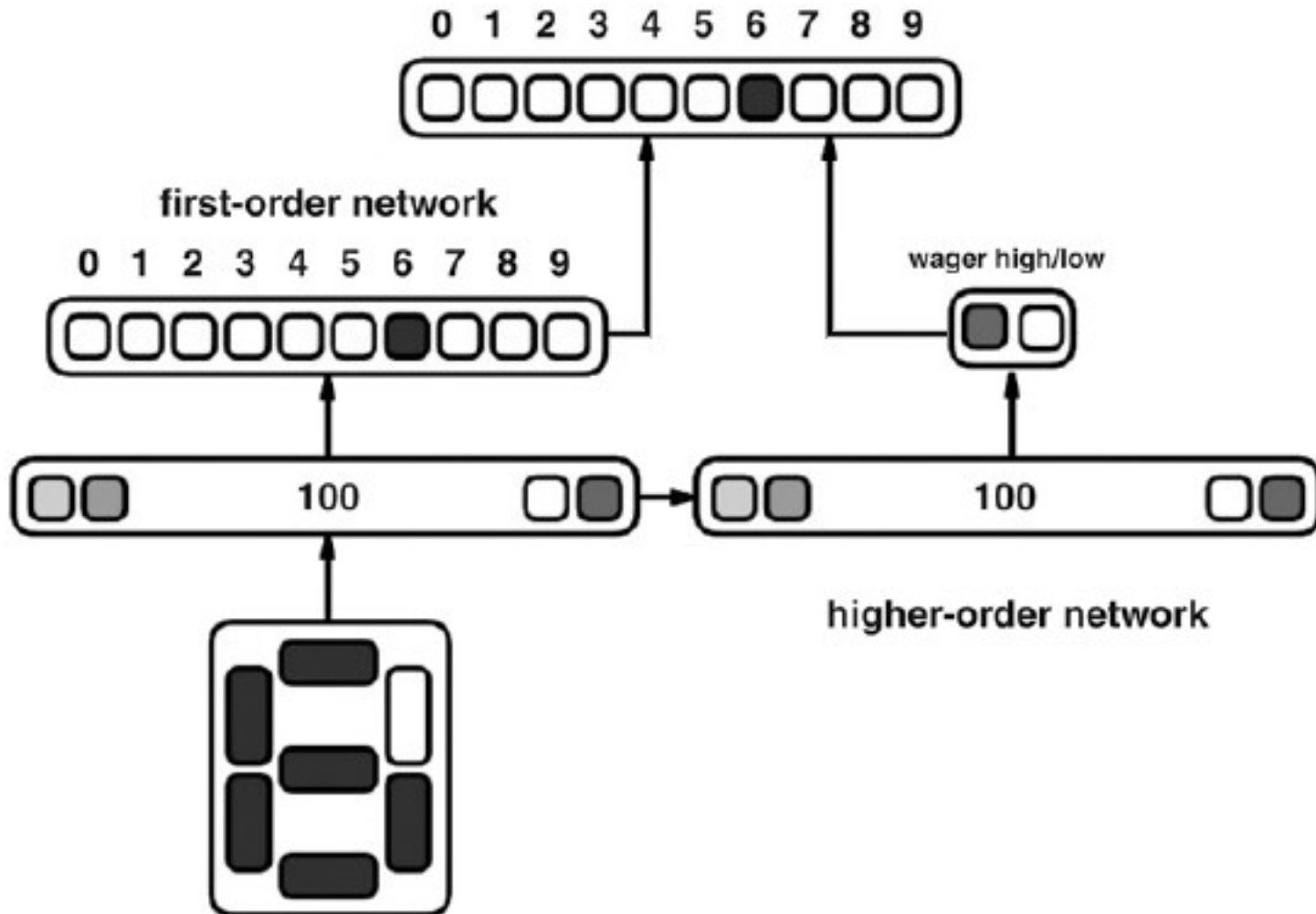
Avec chiffres manuscrits

Performance of first-order and higher-order networks with feedback (nth W-T-A



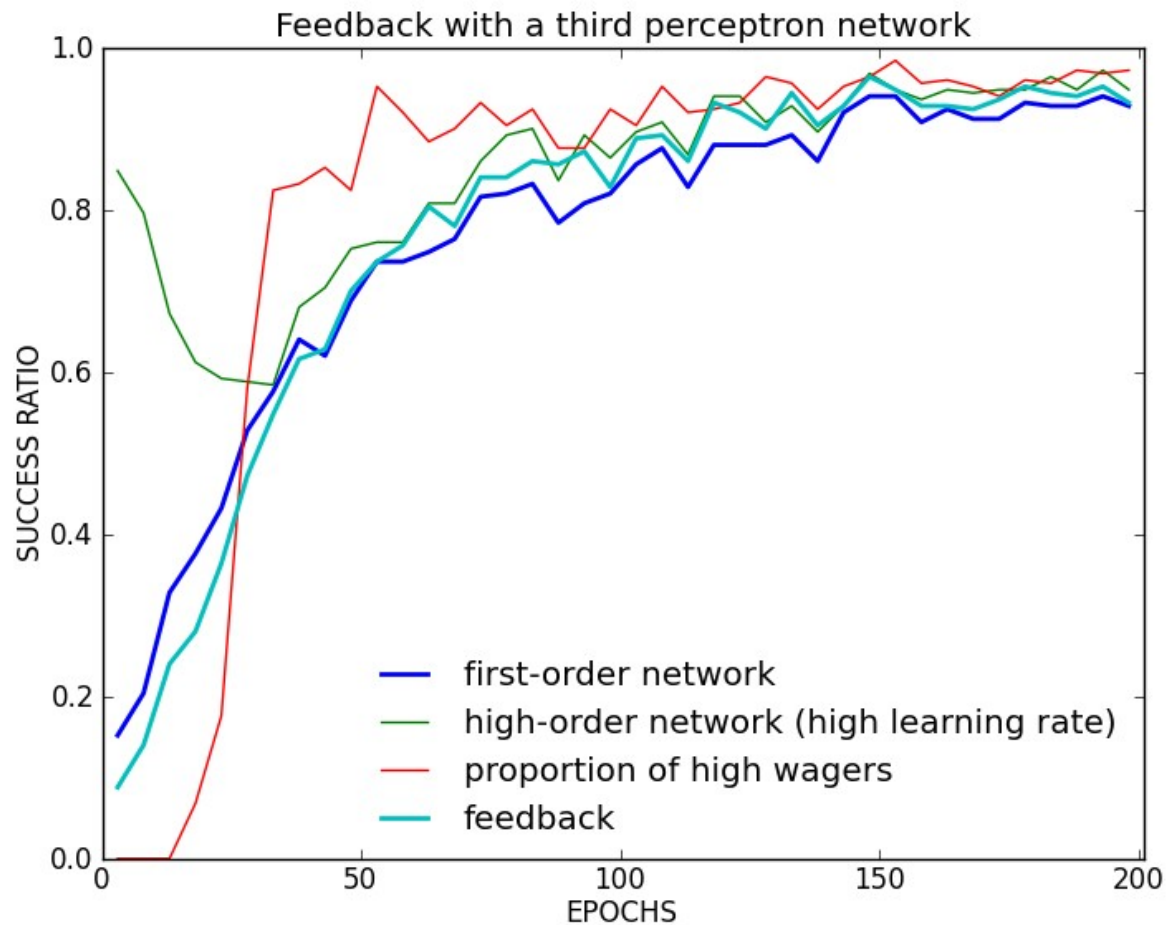
Depuis que le second réseau n'a plus 2 sorties mais 10 : il n'apprend pas plus vite que le premier réseau

Feedback : 3eme réseau



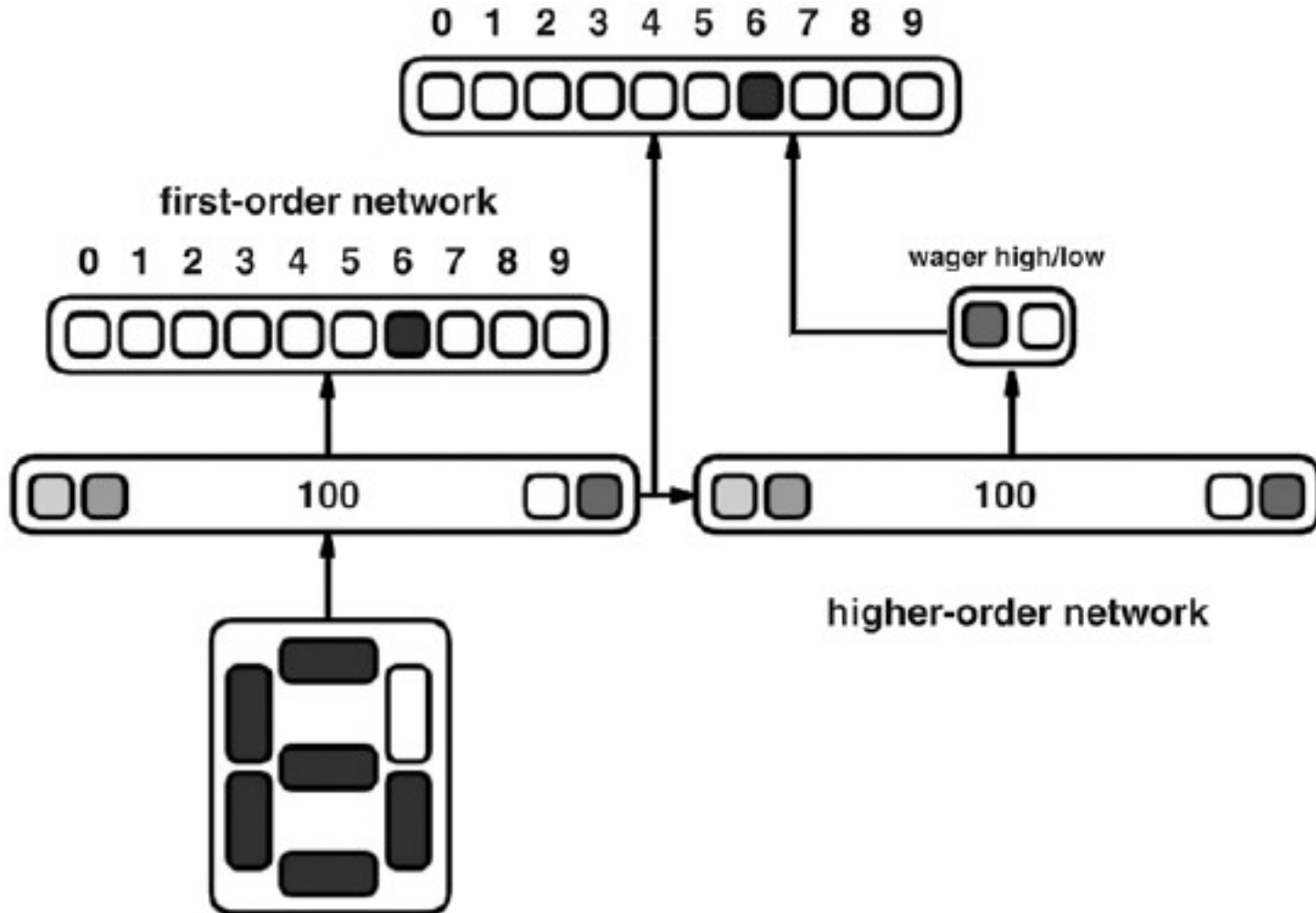
Feedback : 3eme réseau

Avec chiffres manuscrits



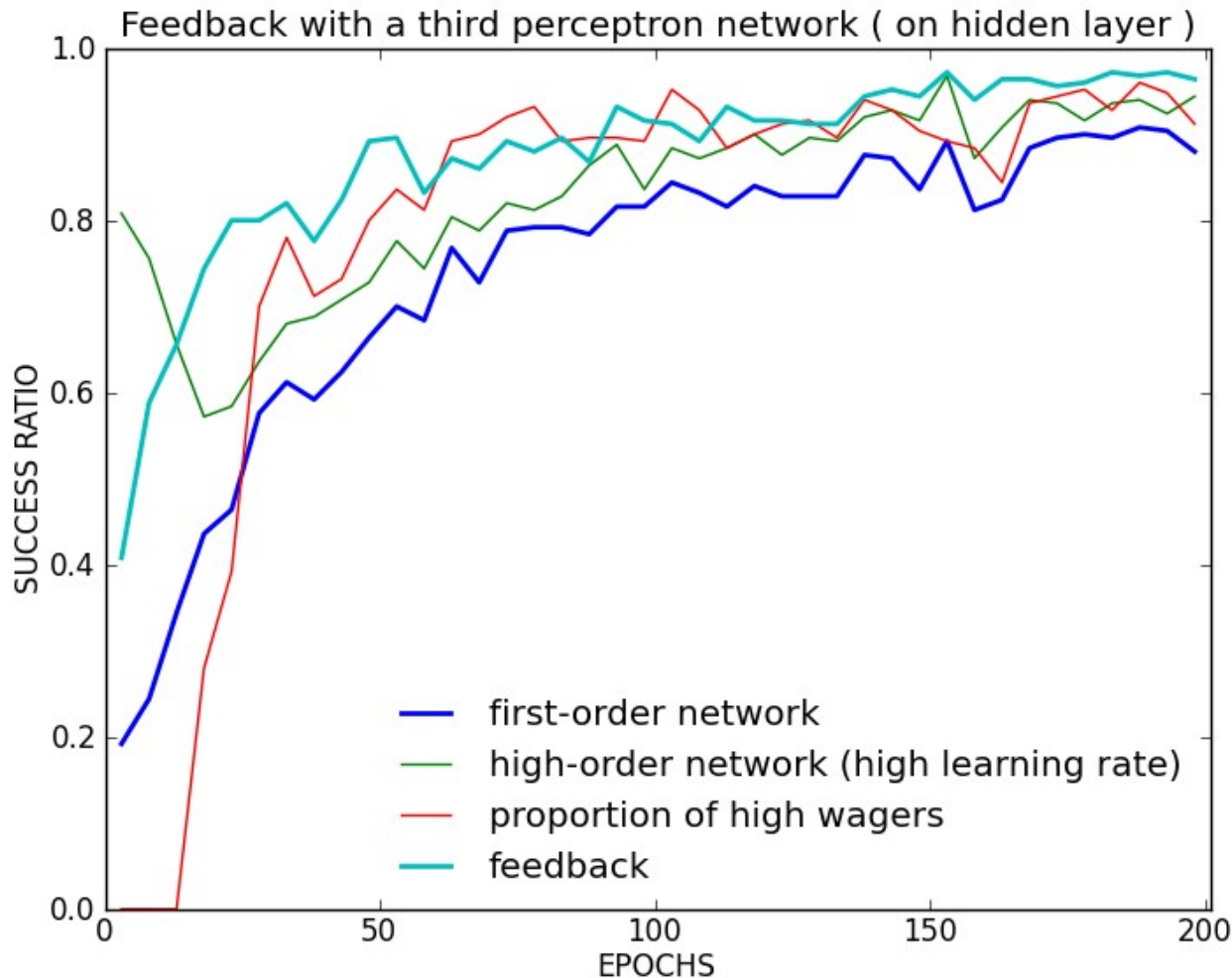
Les performances sont visiblement augmentées

Feedback : 3eme réseau (sur couche cachée)



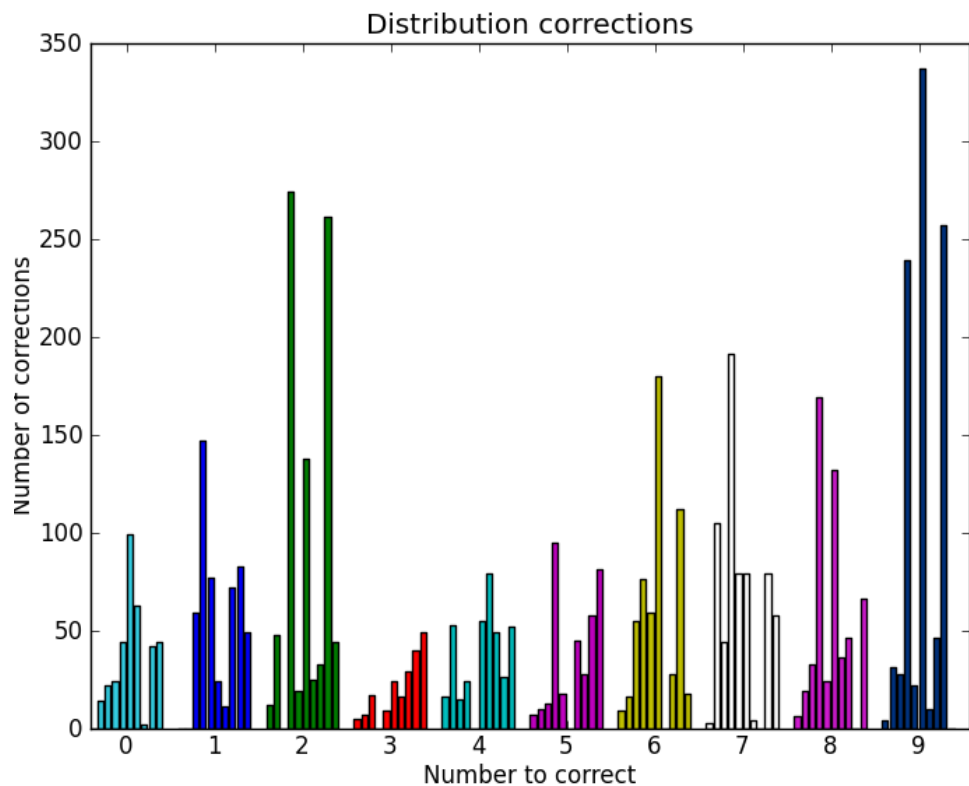
Feedback : 3eme réseau (sur couche cachée)

Avec chiffres manuscrits

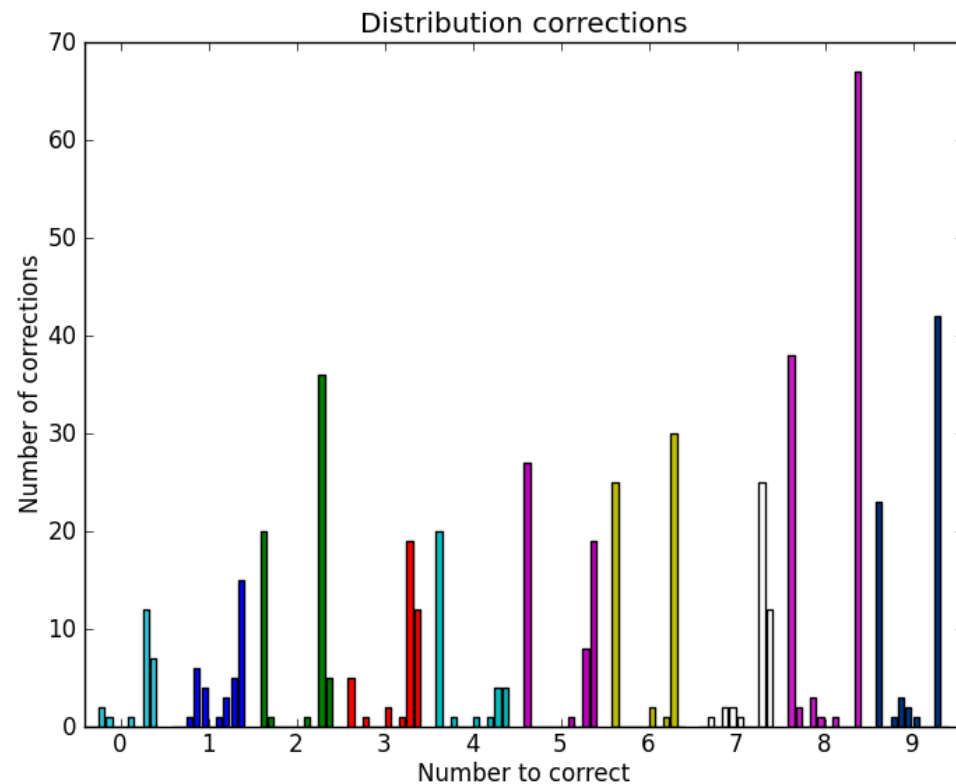


Les performances sont très nettement augmentées

Quelles corrections ?

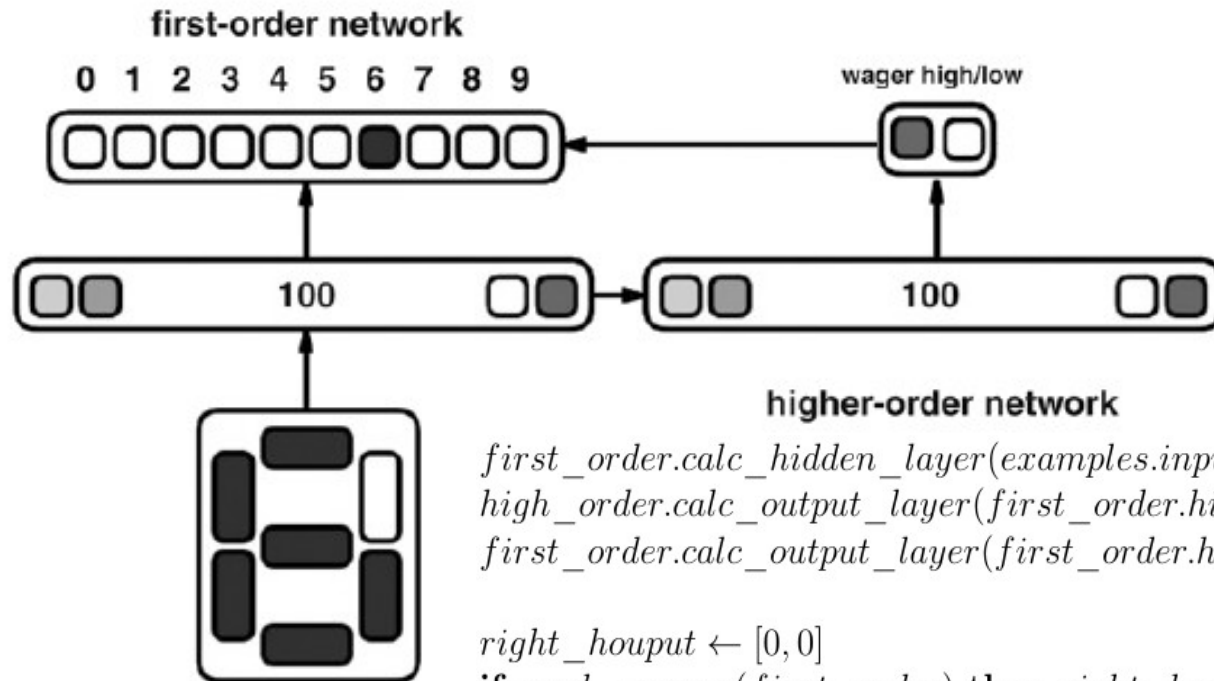


Durant tout l'apprentissage
(50 formes * 200 epochs)



Après l'apprentissage sur
les 1600 formes

Feedback : fusion



```

first_order.calc_hidden_layer(examples.inputs)
high_order.calc_output_layer(first_order.hidden_layer)
first_order.calc_output_layer(first_order.hidden_layer, [0, 0, ...0])

```

```

right_houput ← [0, 0]
if good_answer(first_order) then right_houput[1] ← 1
else right_houput[0] ← 1
end if

```

```

calc_stats()

```

```

ah_ouput ← ampli(high_order.output_layer)
high_order.train(first_order.hidden_layer, right_houput)
first_order.train(examples.inputs, examples.outputs, ah_output)

```


Algo fusion

```
function update_weights_gradient(error, inputs, add)
    calc_output(inputs + add)
```

```
    for  $j = 0 \rightarrow \text{inputs.length}$  do
         $dw \leftarrow \text{weights}[j] - \text{last\_weights}[j]$ 
         $p \leftarrow \text{error} \times \text{inputs}[j]$ 
         $\text{weights}[j] \leftarrow \text{weights}[j] + \text{learning\_rate} \times p + \text{momentum} \times dw$ 
    end for
end function
```

```
function train(inputs, outputs, add)
     $y \leftarrow \text{build\_error\_vector}(\dots)$ 
    update_weights_hidden_layer(...)
```

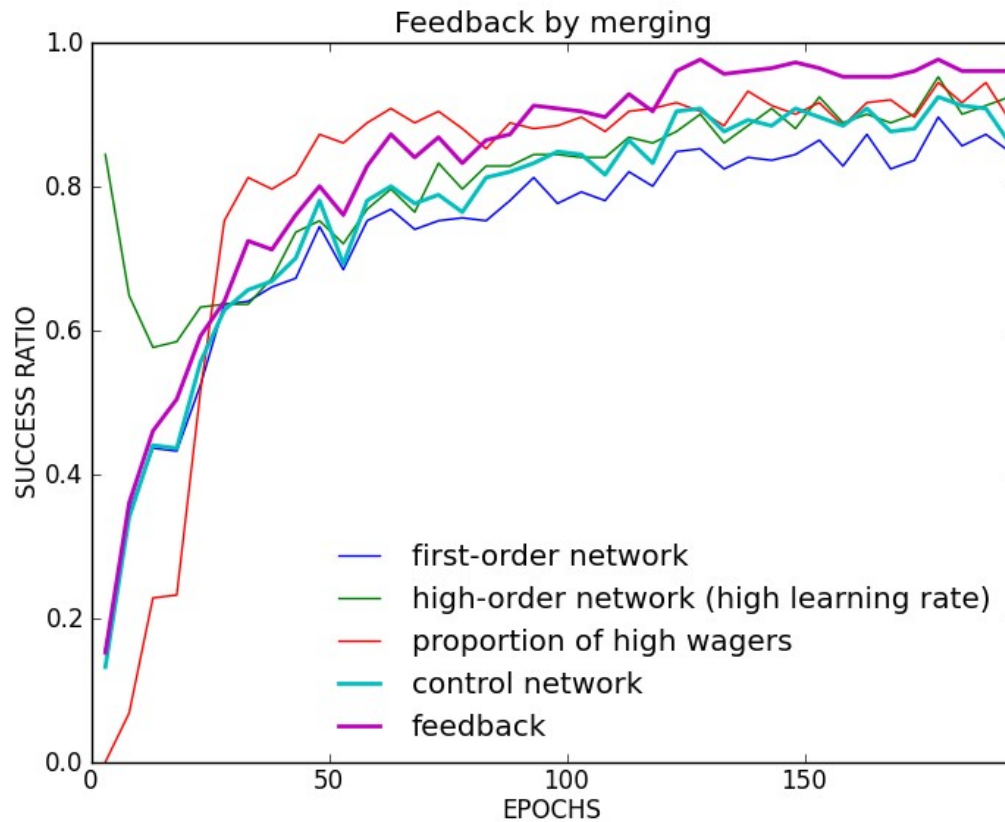
```
    for  $i = 0 \rightarrow \text{output\_neurons.length}$  do
        output_neurons[i].update_weights_gradient( $y[i]$ , hidden_neurons, add)
        output_neurons[i].update_weights_perceptron(outputs[i], hidden_neurons, add)
    end for
end function
```

```
function update_weights_perceptron(goal, inputs, add)
    calc_output(inputs + add)
```

```
    for  $j = \text{inputs.length} \rightarrow \text{inputs.length} + \text{add.length}$  do
         $dw \leftarrow \text{weights}[j] - \text{last\_weights}[j]$ 
         $p \leftarrow (\text{goal} - \text{state}) \times \text{add}[\text{inputs.length} - j]$ 
         $\text{weights}[j] \leftarrow \text{weights}[j] + \frac{\text{learning\_rate} \times p + \text{momentum} \times dw}{\text{add.length}}$ 
    end for
end function
```

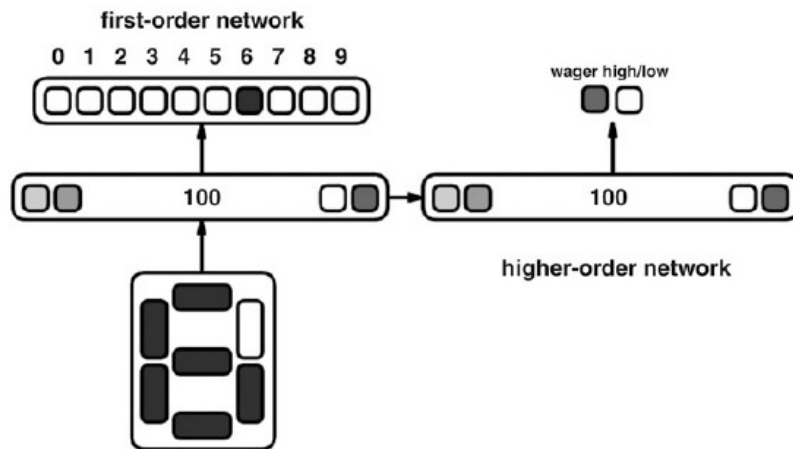
Feedback : fusion

Avec chiffres manuscrits

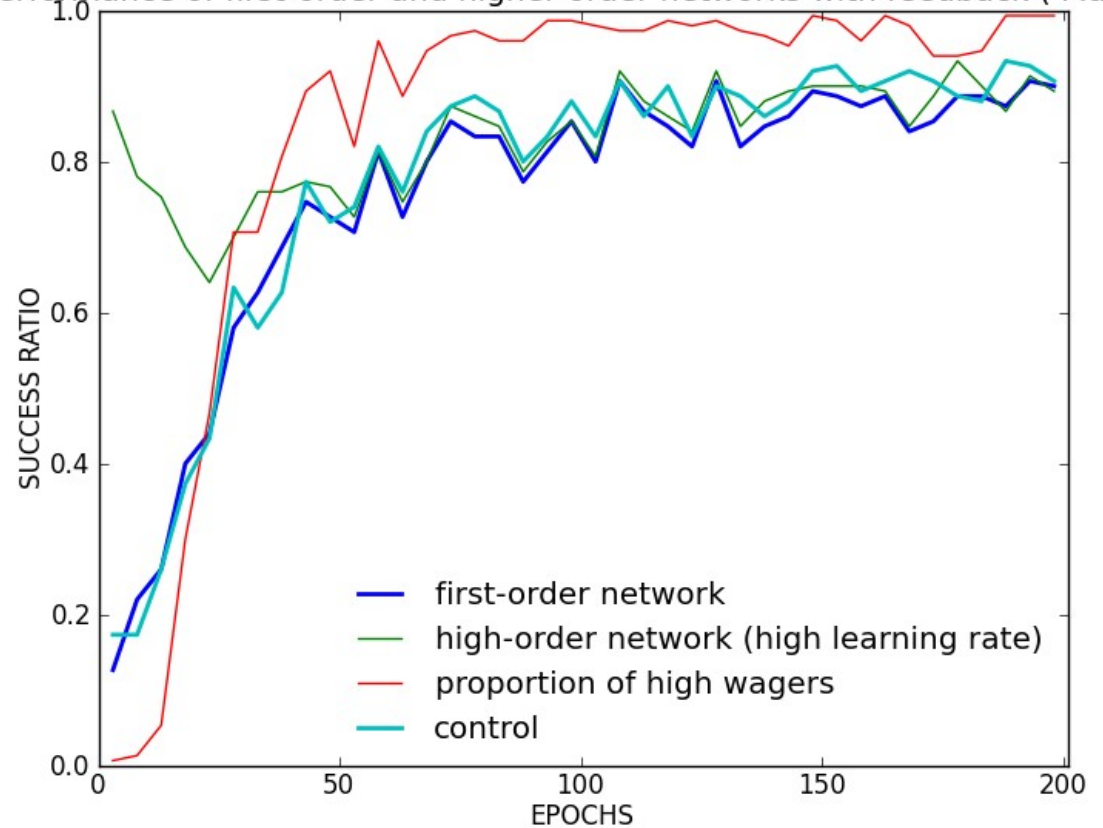


Auto-supervisation

Avec chiffres manuscrits



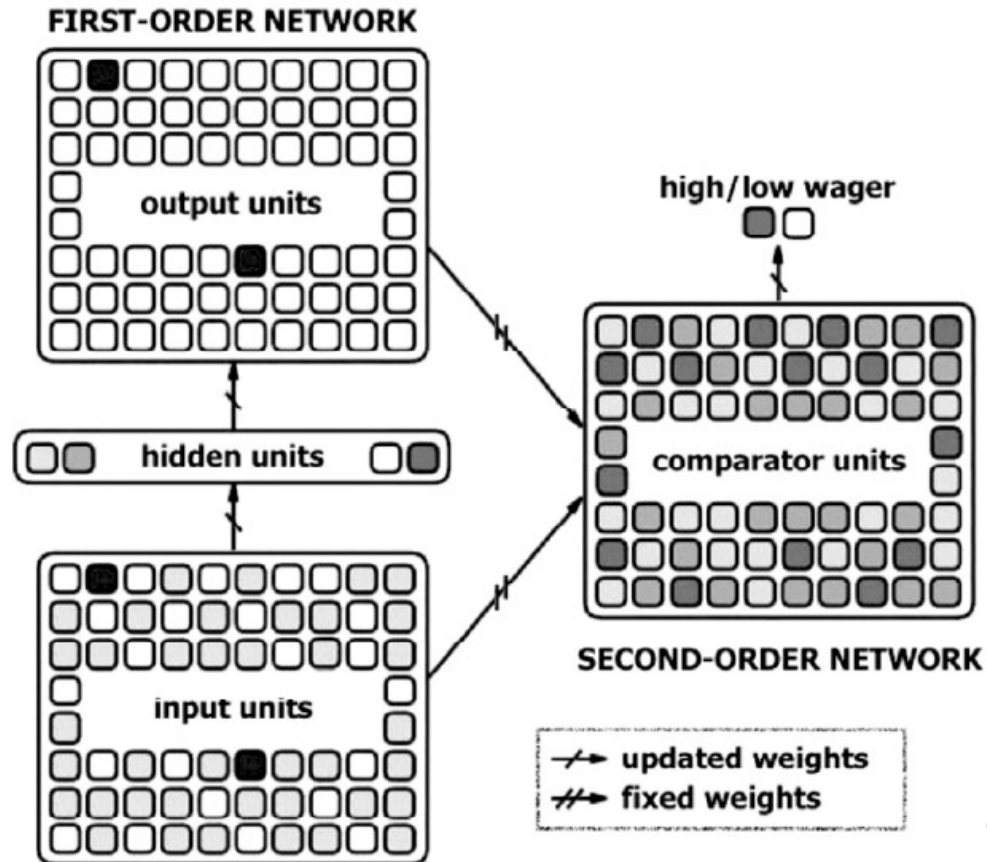
Performance of first-order and higher-order networks with feedback (Master)



Vers où va t-on ?

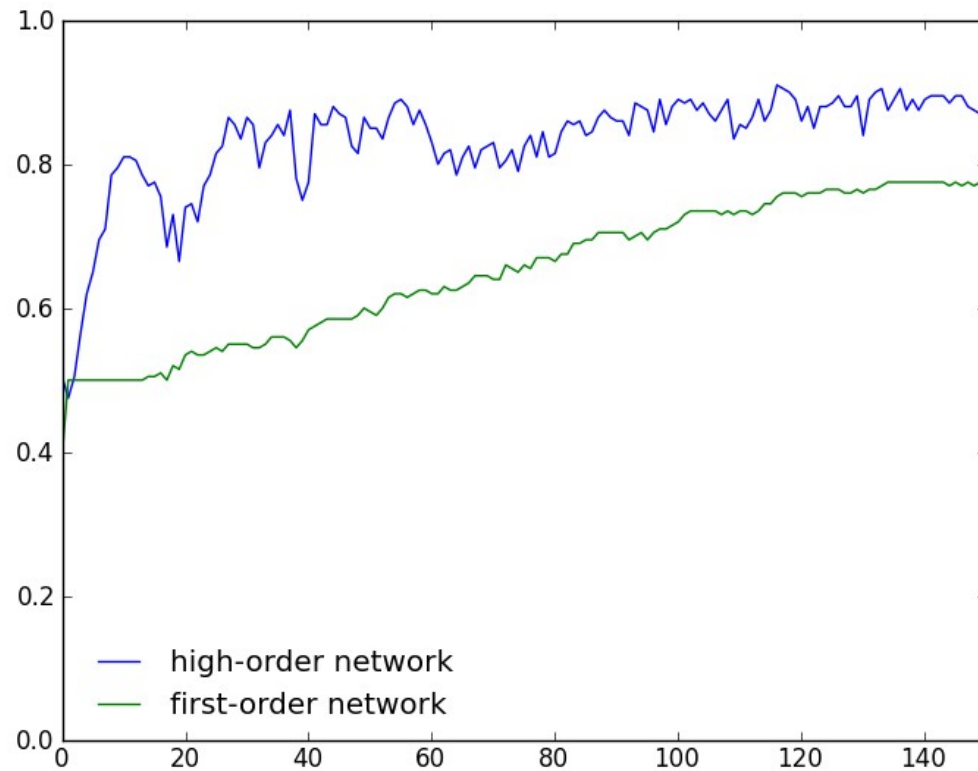
- ?

Simulation 3

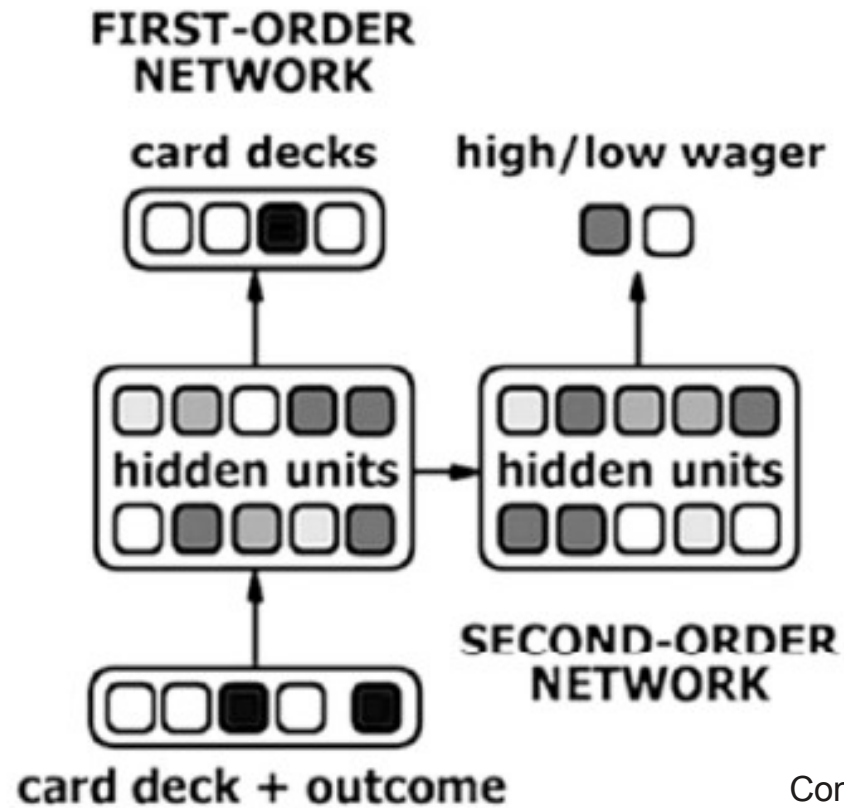


Consciousness and metarepresentation :
A computational sketch
[Alex Cleeremans, Bert Timmermans, Antoine Pasquali]

Simulation 3



Simulation 4



Consciousness and metarepresentation :
A computational sketch
[Alex Cleeremans, Bert Timmermans, Antoine Pasquali]

Simulation 4

