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]

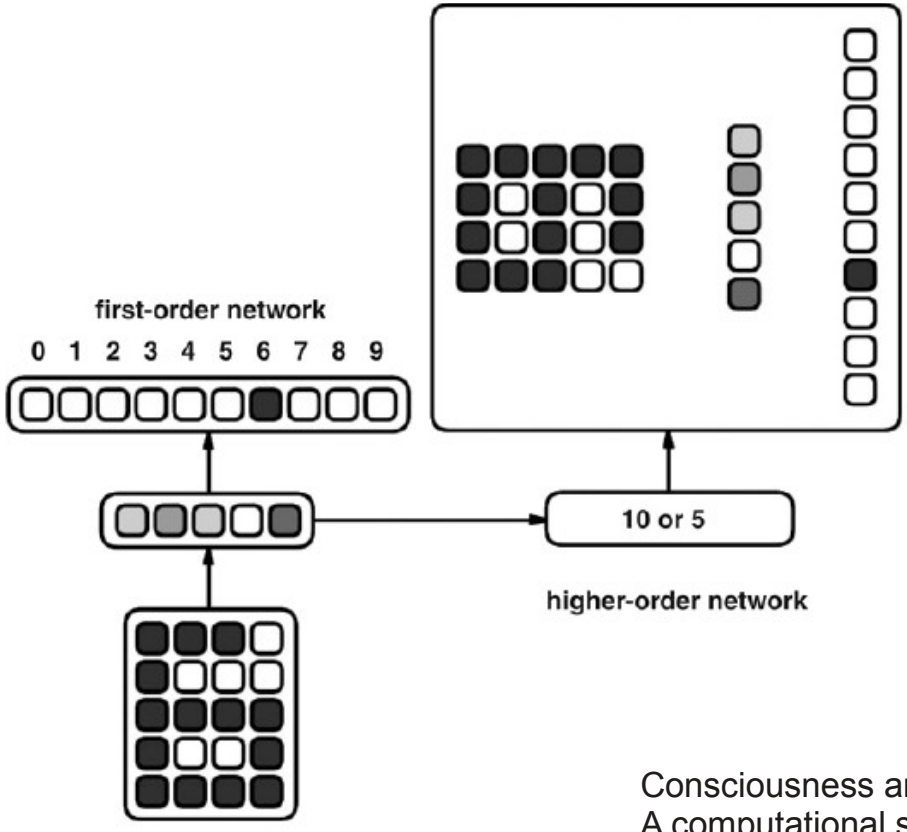
- Know thyself : Metacognitive networks and mesures of consciousness

  [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



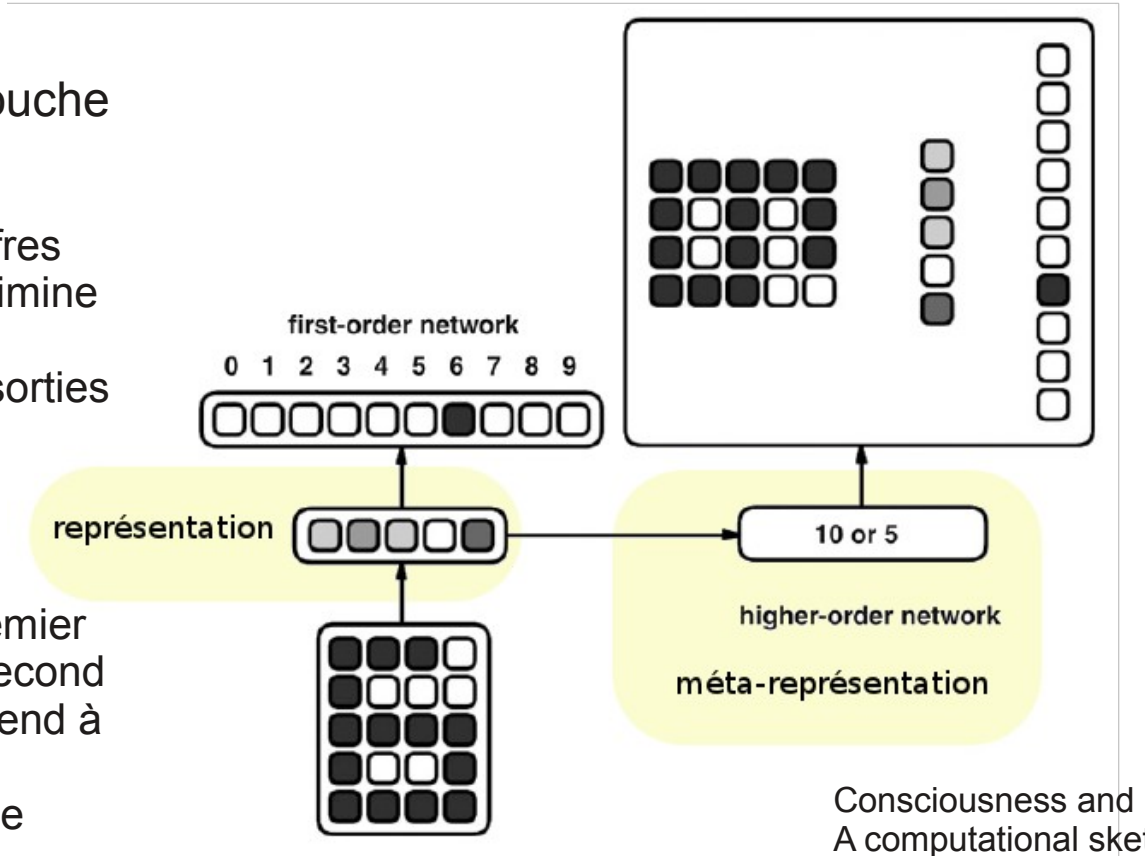
- La couche caché du premier réseau est l'entrée du second
- Le second réseau apprend à redonner l'état entier du premier (entrée / couche caché / sortie)

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

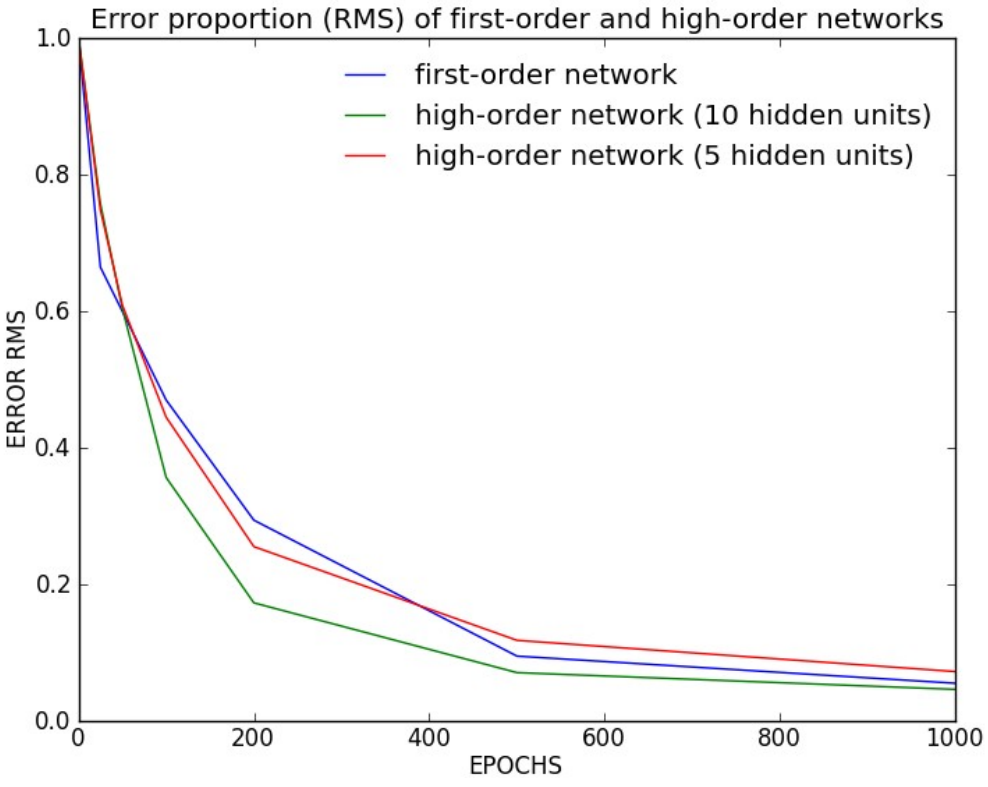


- La couche caché du premier réseau est l'entrée du second
- Le second réseau apprend à redonner l'état entier du premier (entrée / couche caché / sortie)

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

Résultats sur la base d'entrée de l'article

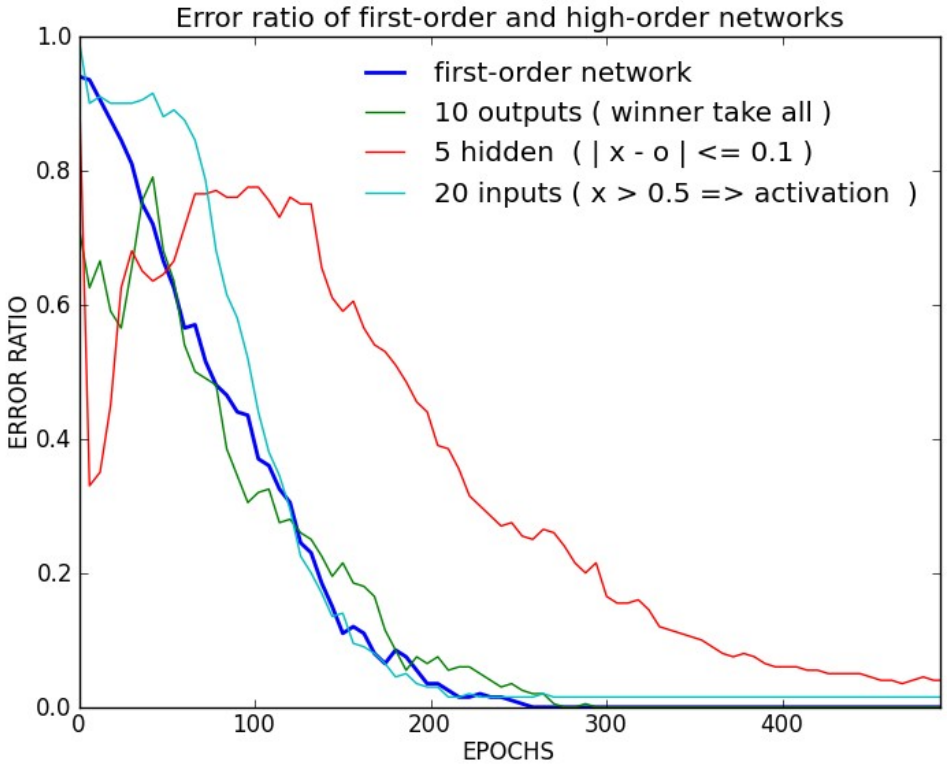
De l'article



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

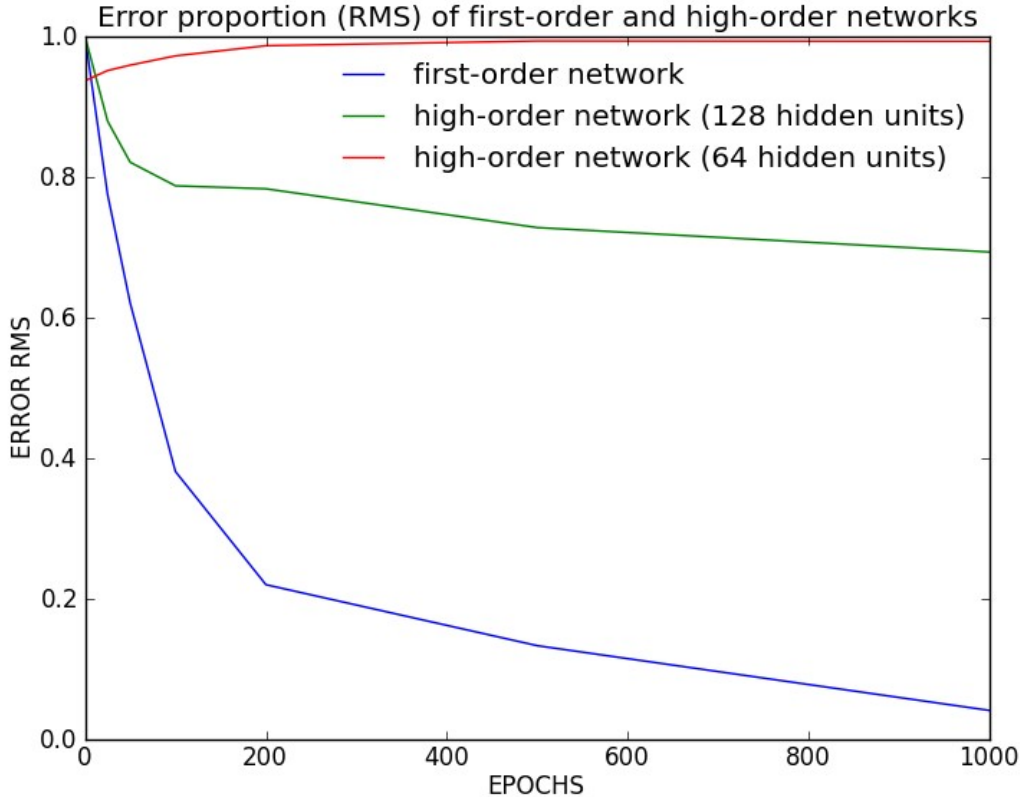
$$with \begin{cases} n : number\ of\ neurons\ on\ the\ output\ layer \\ o_{i,e} : value\ obtained\ for\ the\ i^{th}\ neuron\ at\ the\ e^{th}\ epoch \\ d_i : value\ desired\ for\ the\ i^{th}\ neuron \end{cases}$$

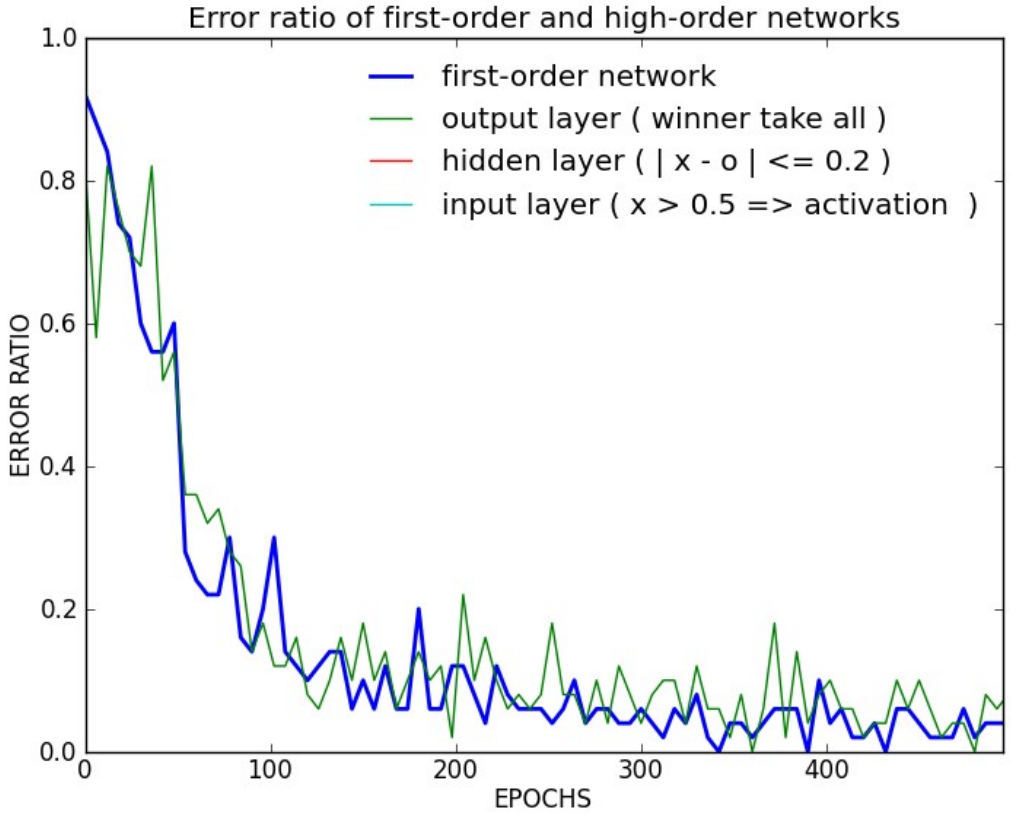
Notre touche personnelle



- 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

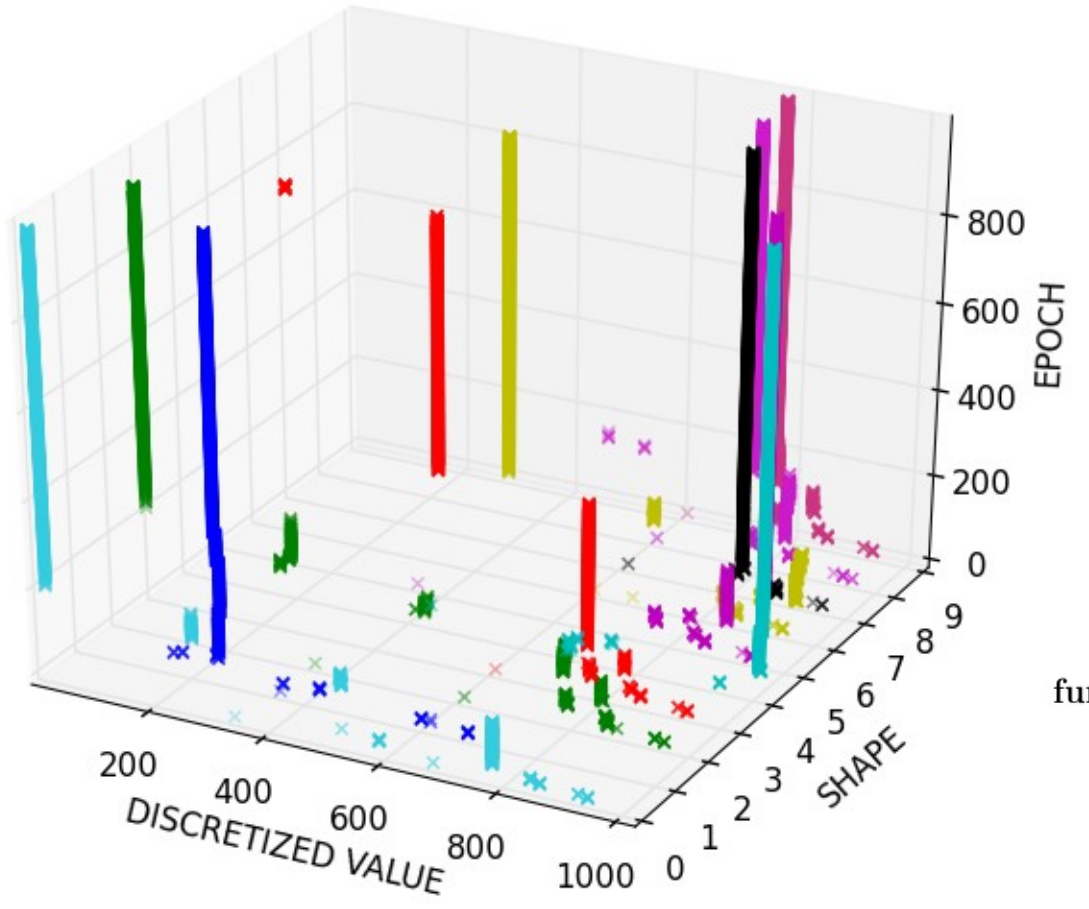


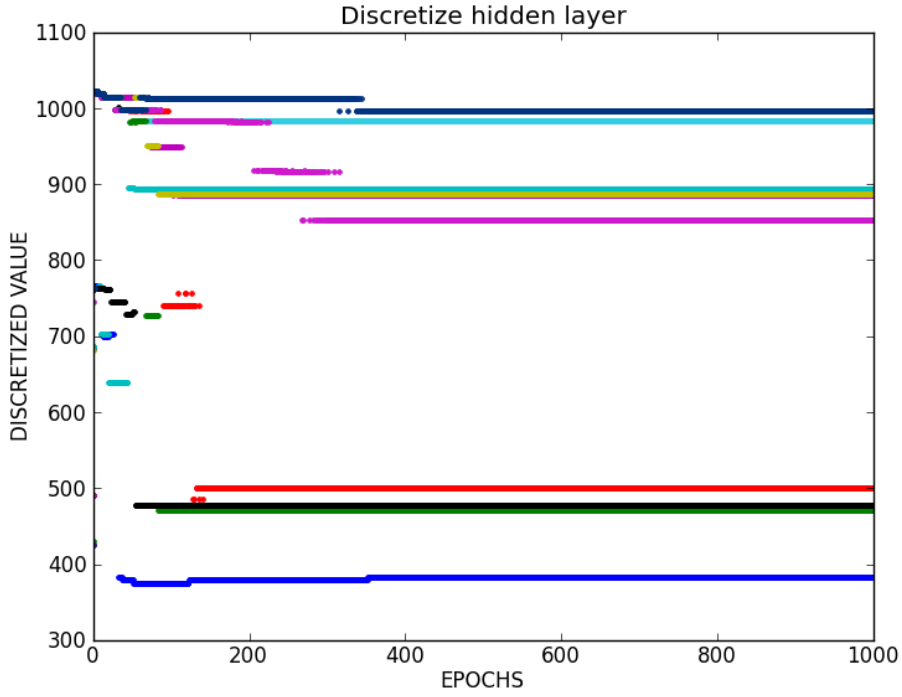


- 256 entrées
- 64 neurones cachés
- 10 sorties
- 1600 formes
- 10 formes/epoch

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

Pourquoi ça marche ?

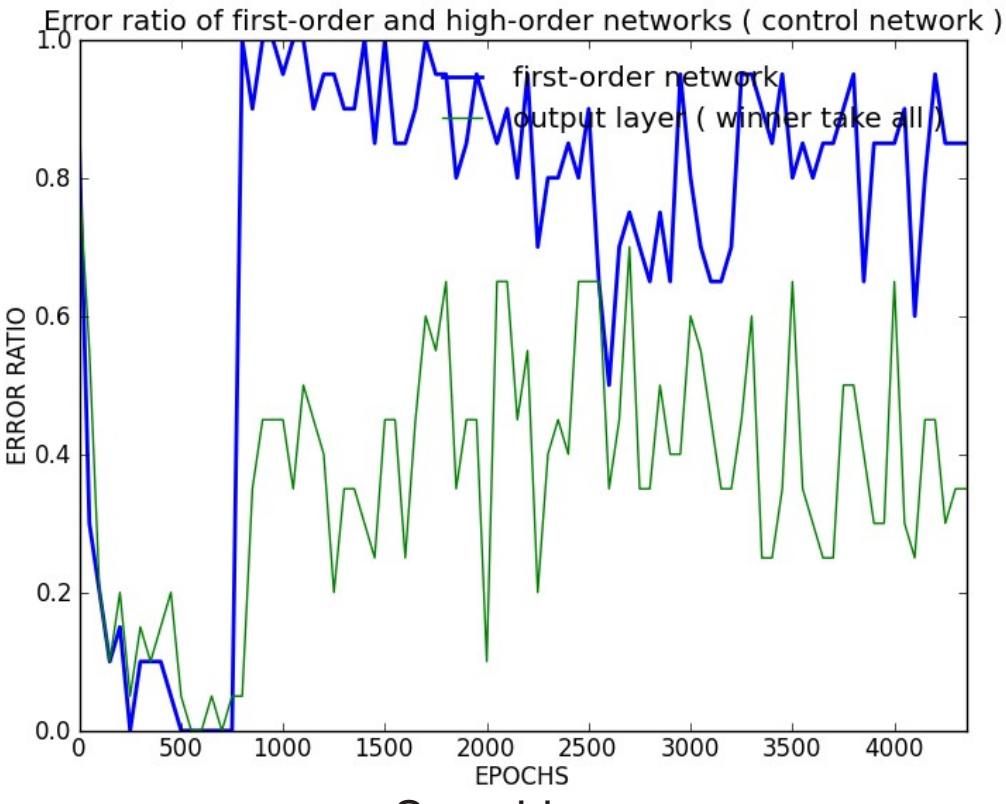




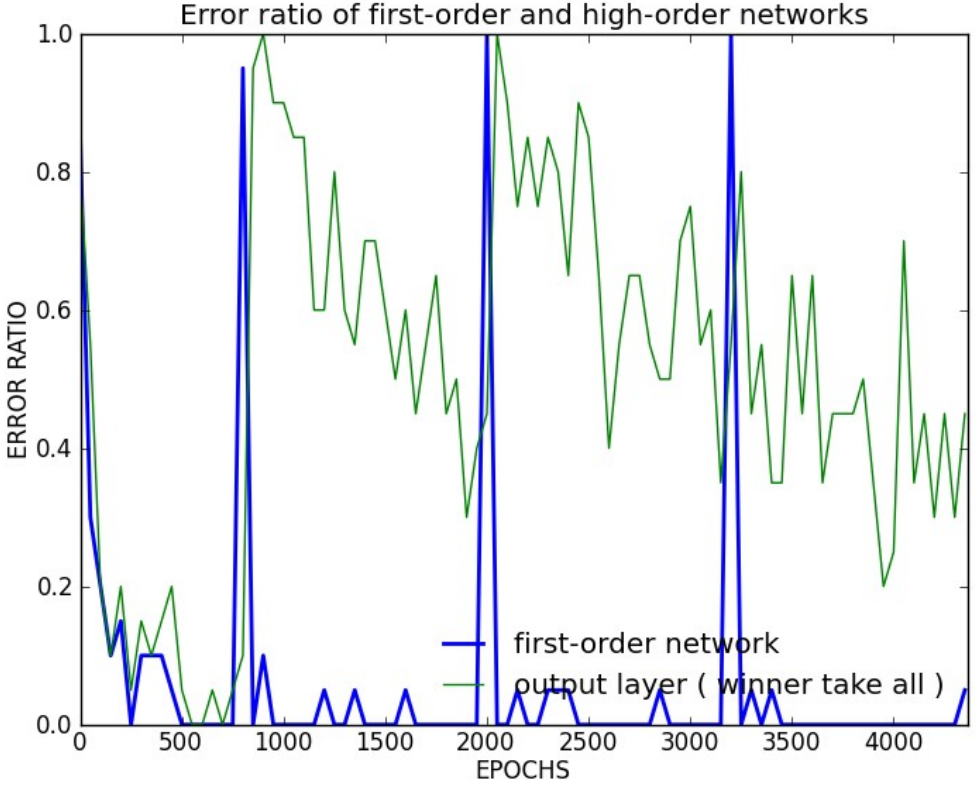
function DISCRETIZE($hiddenNeuron[], piece$)
  $result \leftarrow 0$
  for $i = 0 \rightarrow hiddenNeuron.length$ **do**
    $result \leftarrow result + piece^i \times cutting(hiddenNeuron[i], piece)$
    $i \leftarrow i + 1$
  end for
  return result
end function

Changement de tâche

Avec chiffres manuscrits



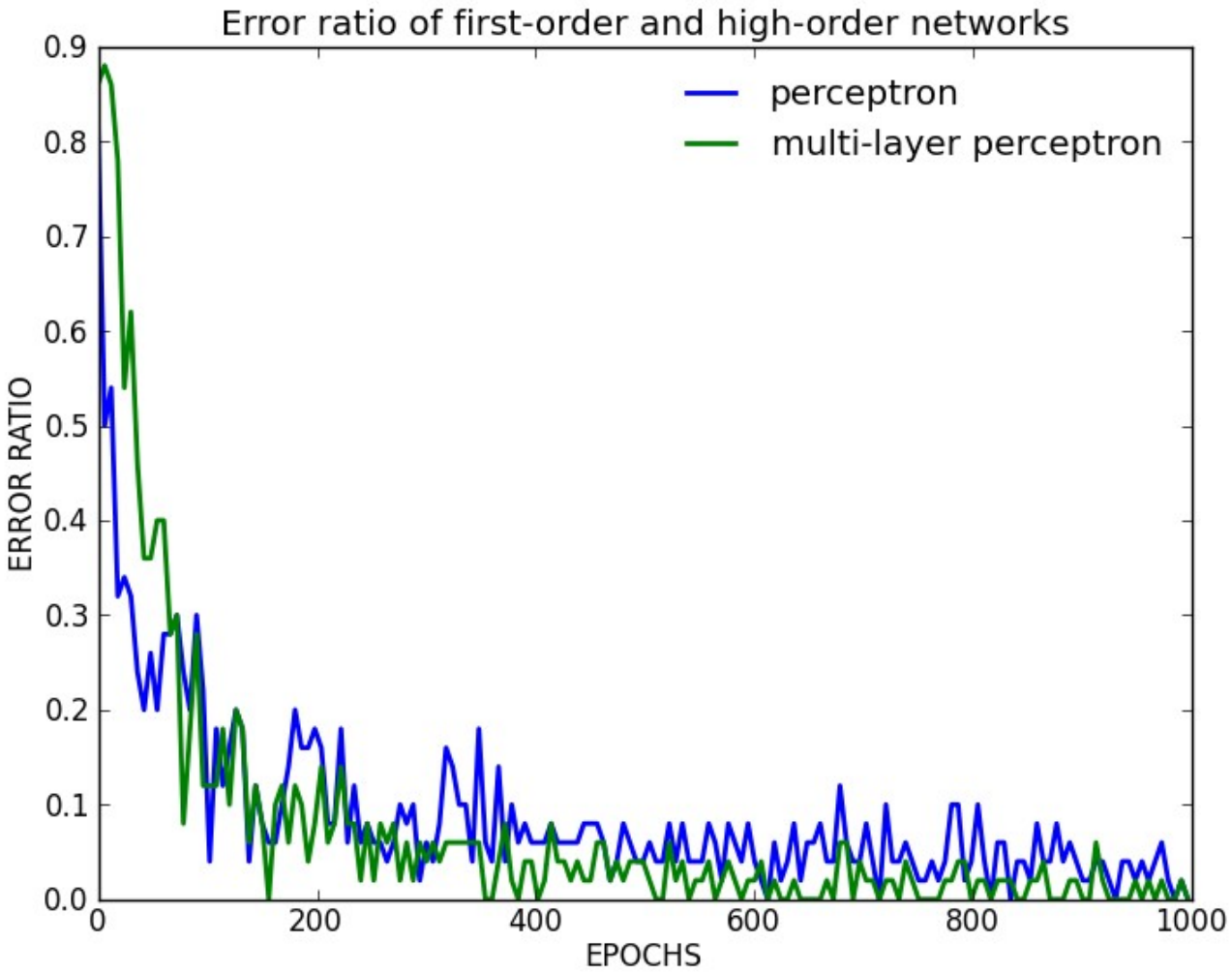
Sans blocage



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

Perceptron vs Multi-layer Perceptron

Avec chiffres manuscrits

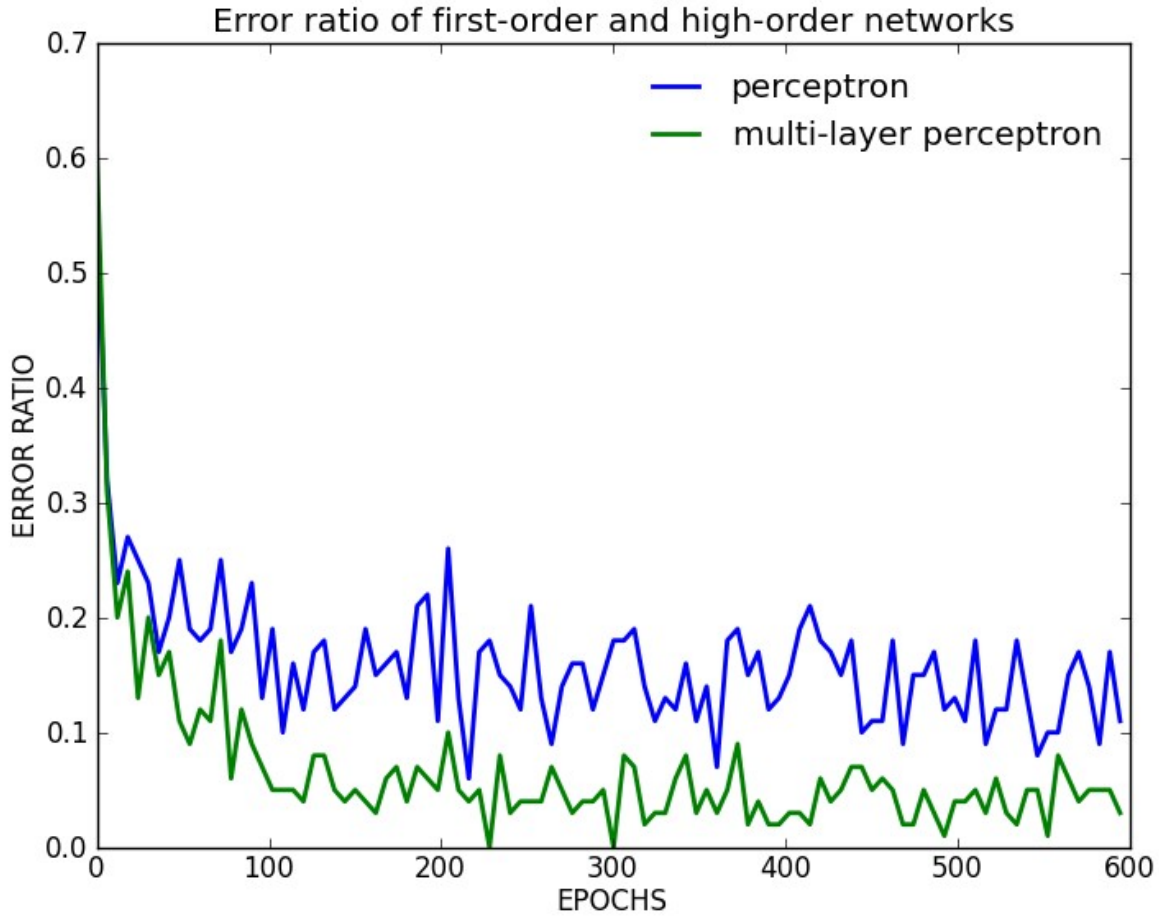


- Le perceptron apprend plus vite au début
- Le MLP a un meilleur taux de classification après 200 epochs

A-t-on besoin d'un MPL ?

Perceptron vs Multi-layer Perceptron

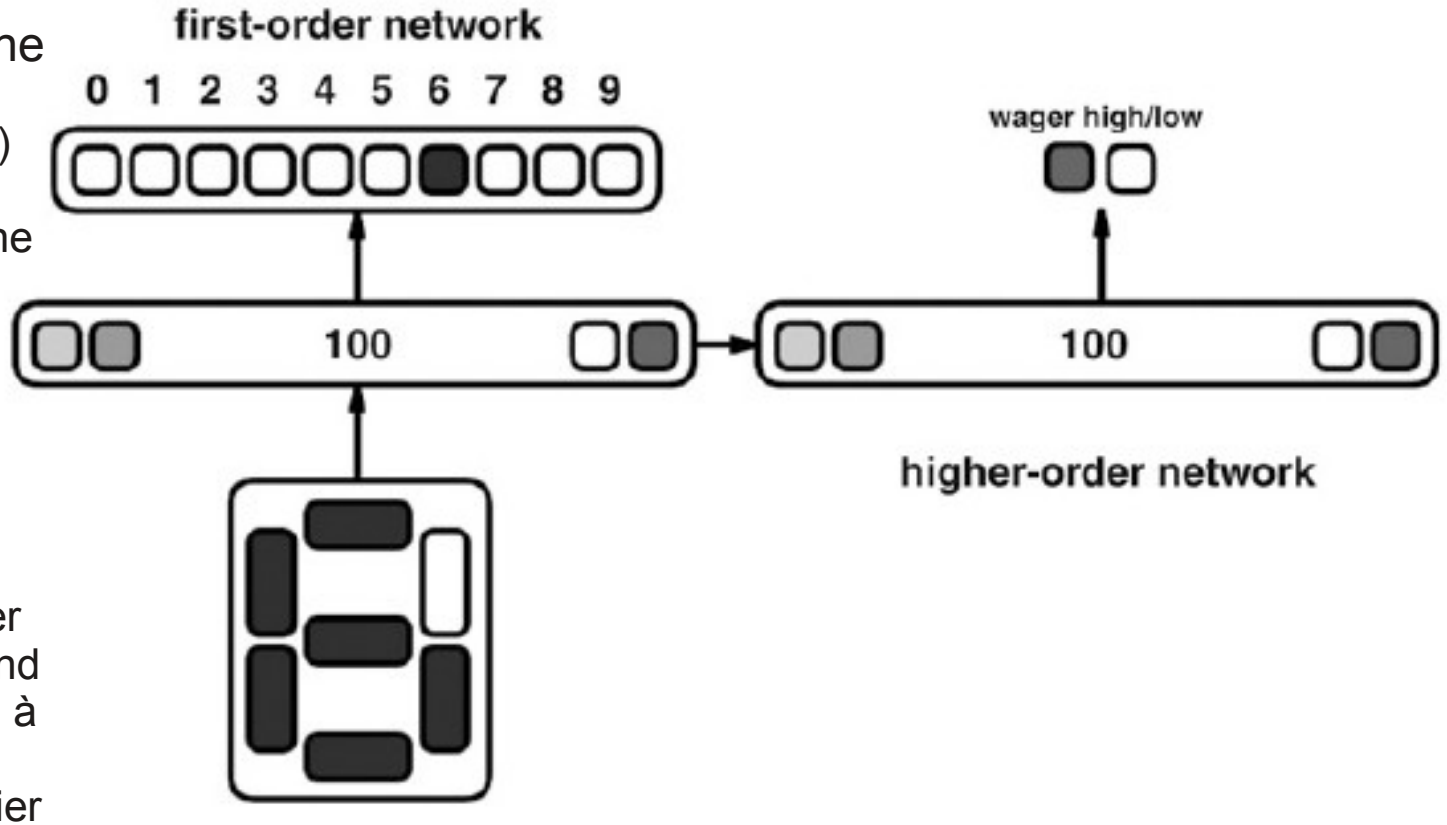
Avec des iris



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

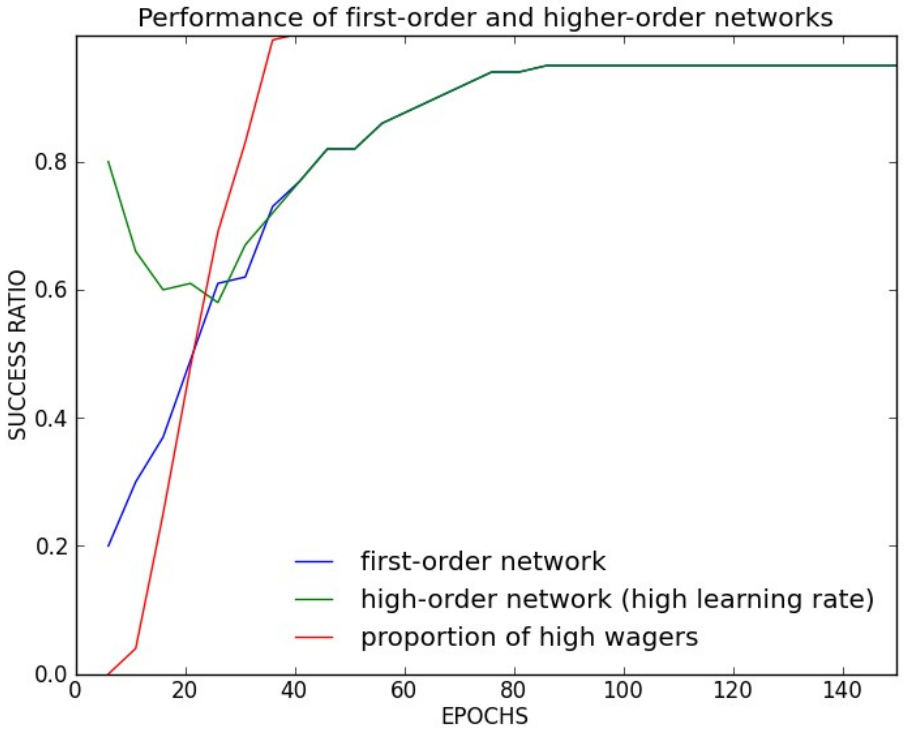


- La couche caché du premier réseau est l'entrée du second
- Le second réseau apprend à parier sur la qualité de la réponse donné par le premier

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

Résultats sur la base d'entrée de l'article

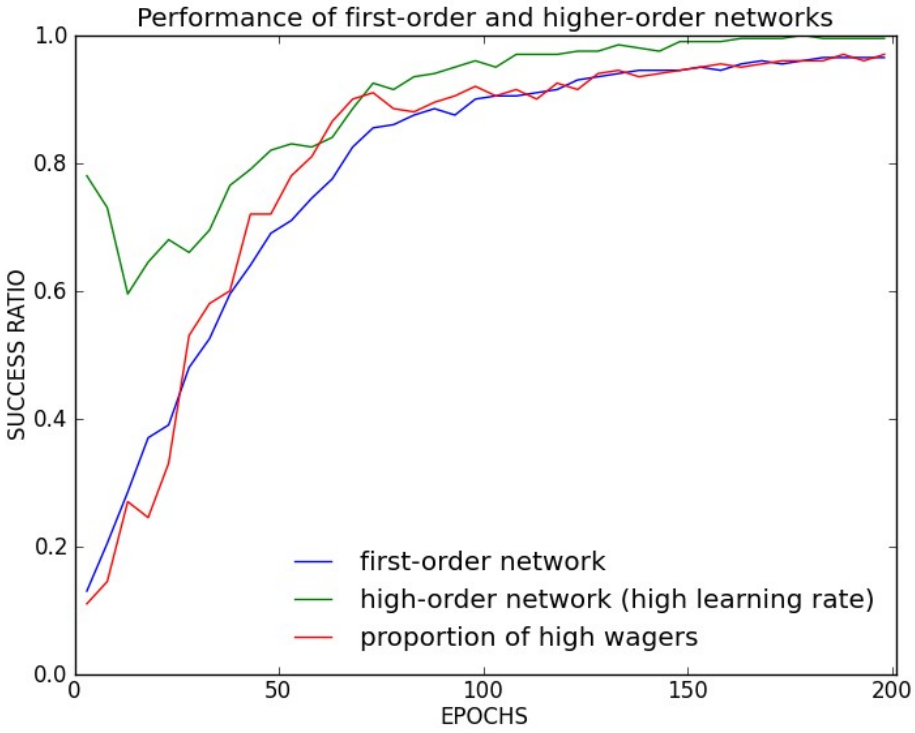
De l'article



Param. réseau supérieur :

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

Notre touche personnelle

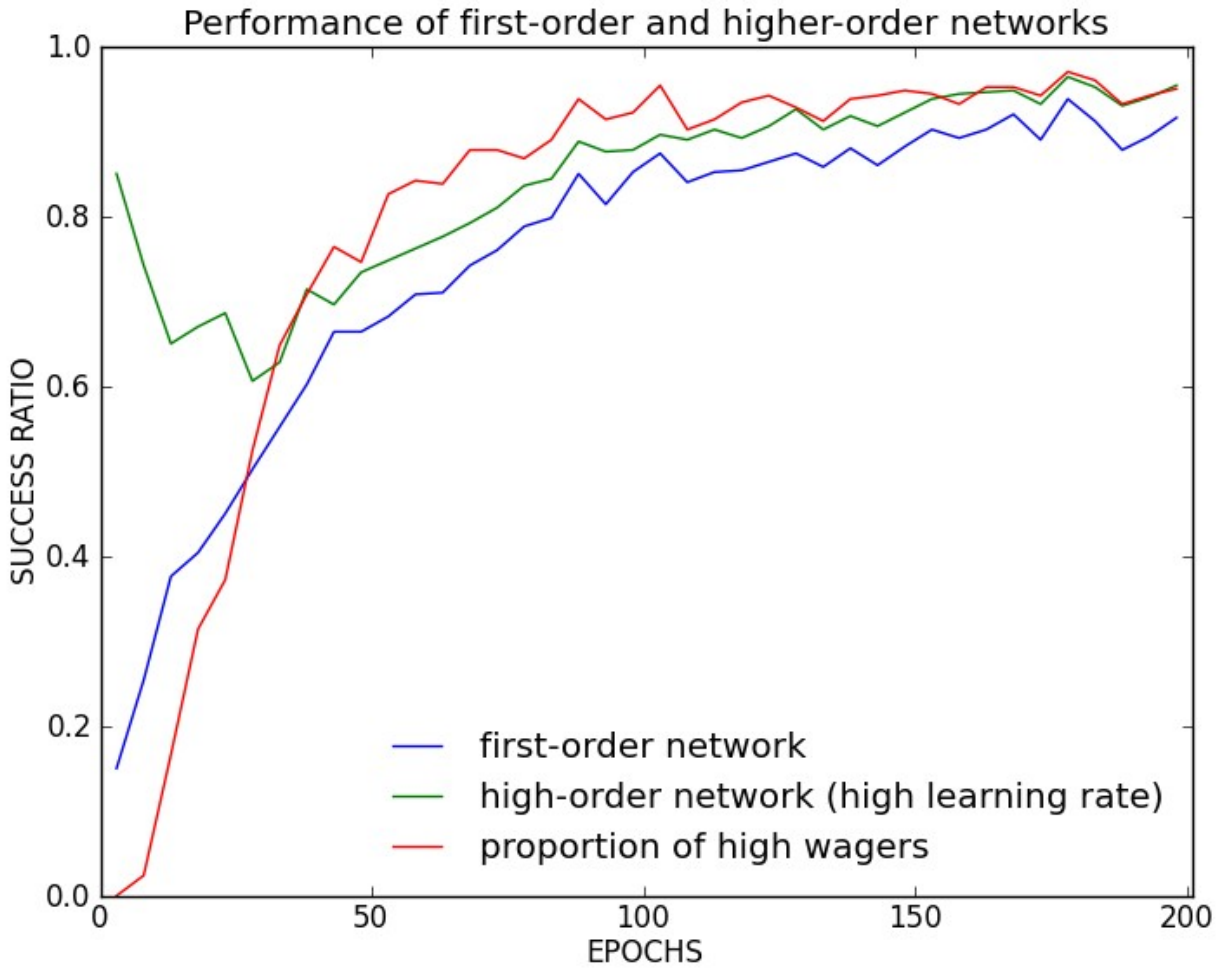


Param. réseau supérieur :

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

(comme le premier réseau)

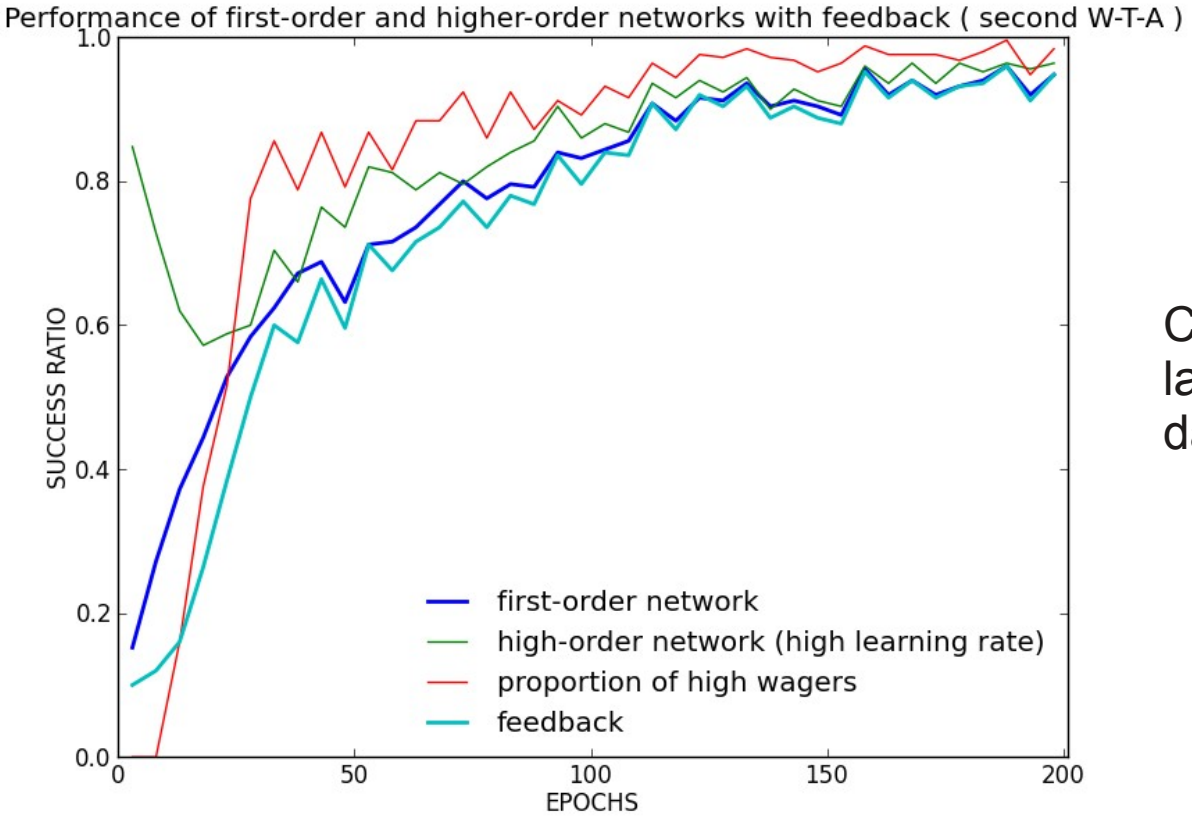
Approfondissement sur des chiffres manuscrits



- 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

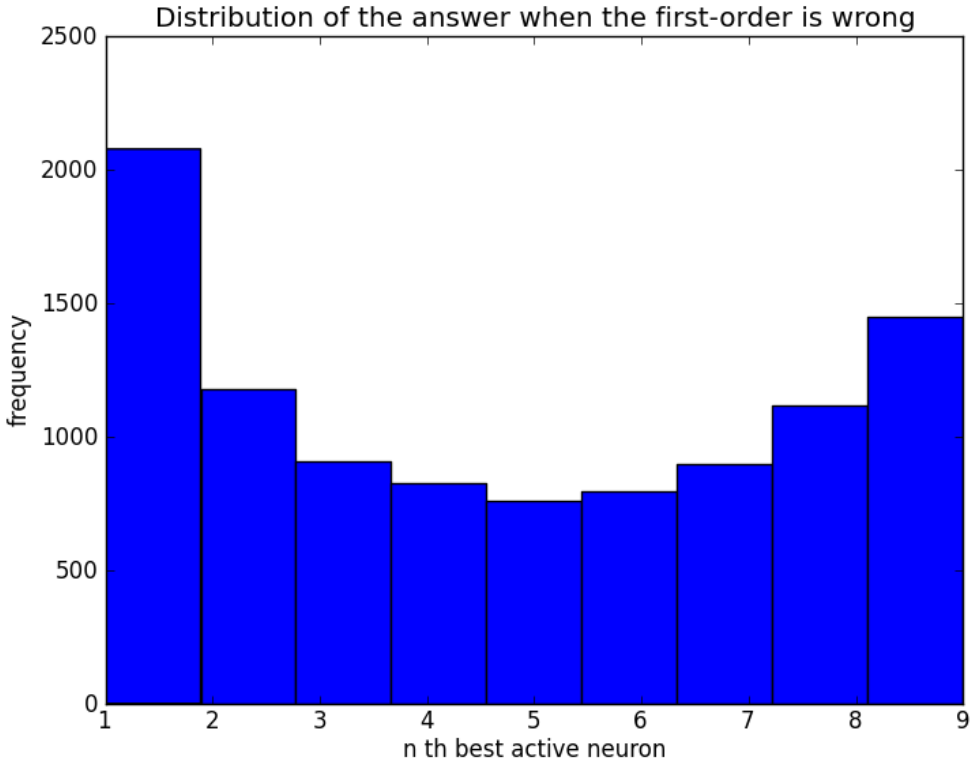
Avec chiffres manuscrits



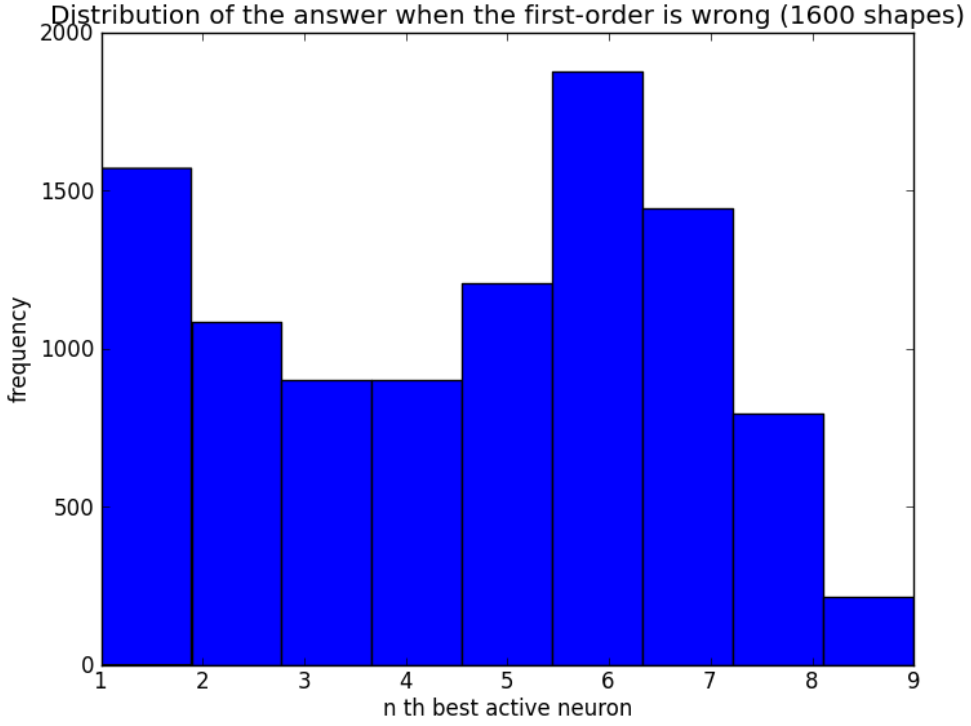
Conclusion :
la réponse ne se trouve pas toujours dans le second neurone le plus actif

Emplacement de la bonne réponse

Avec chiffres manuscrits



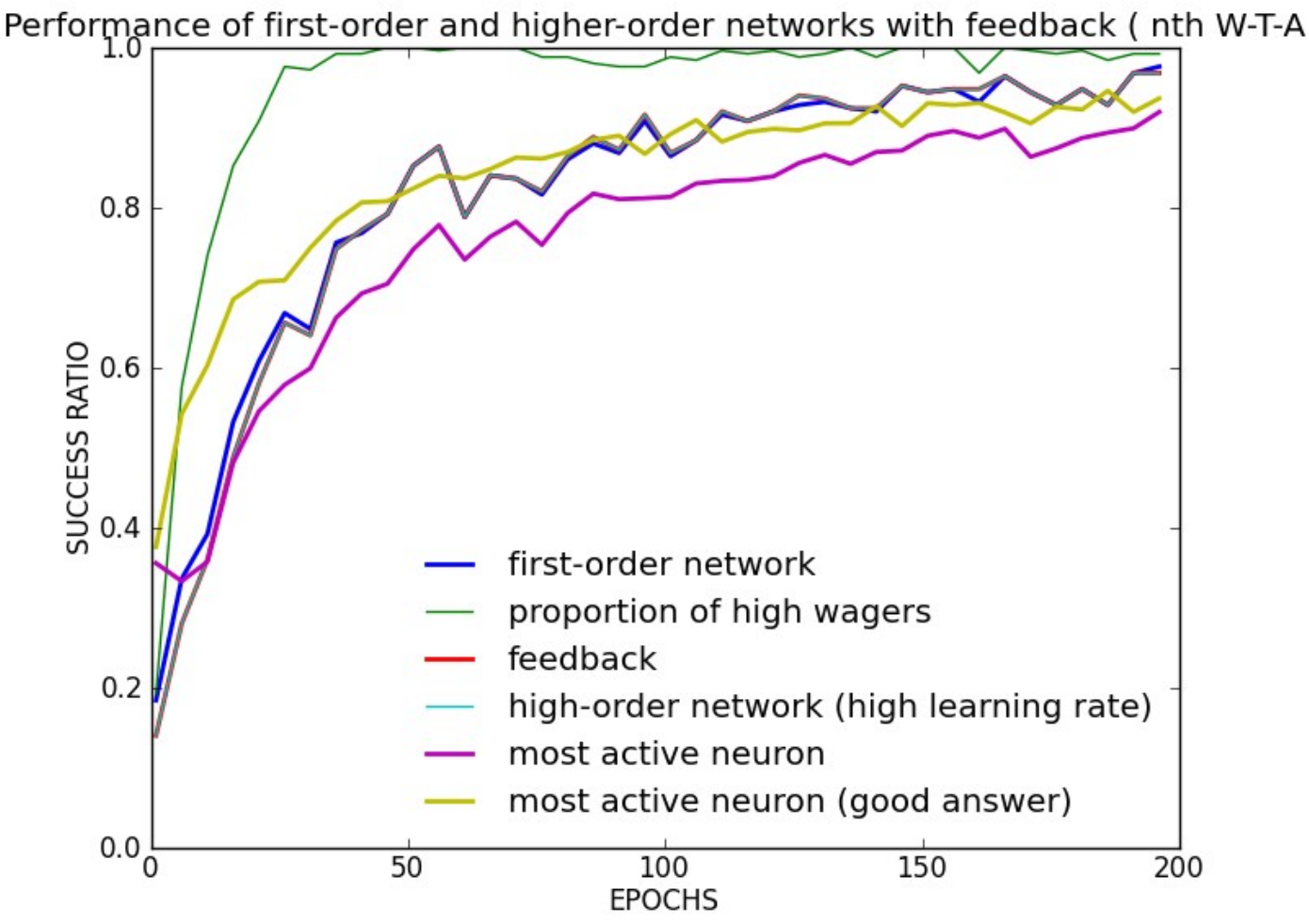
Durant tout l'apprentissage
(50 formes * 200 epochs)



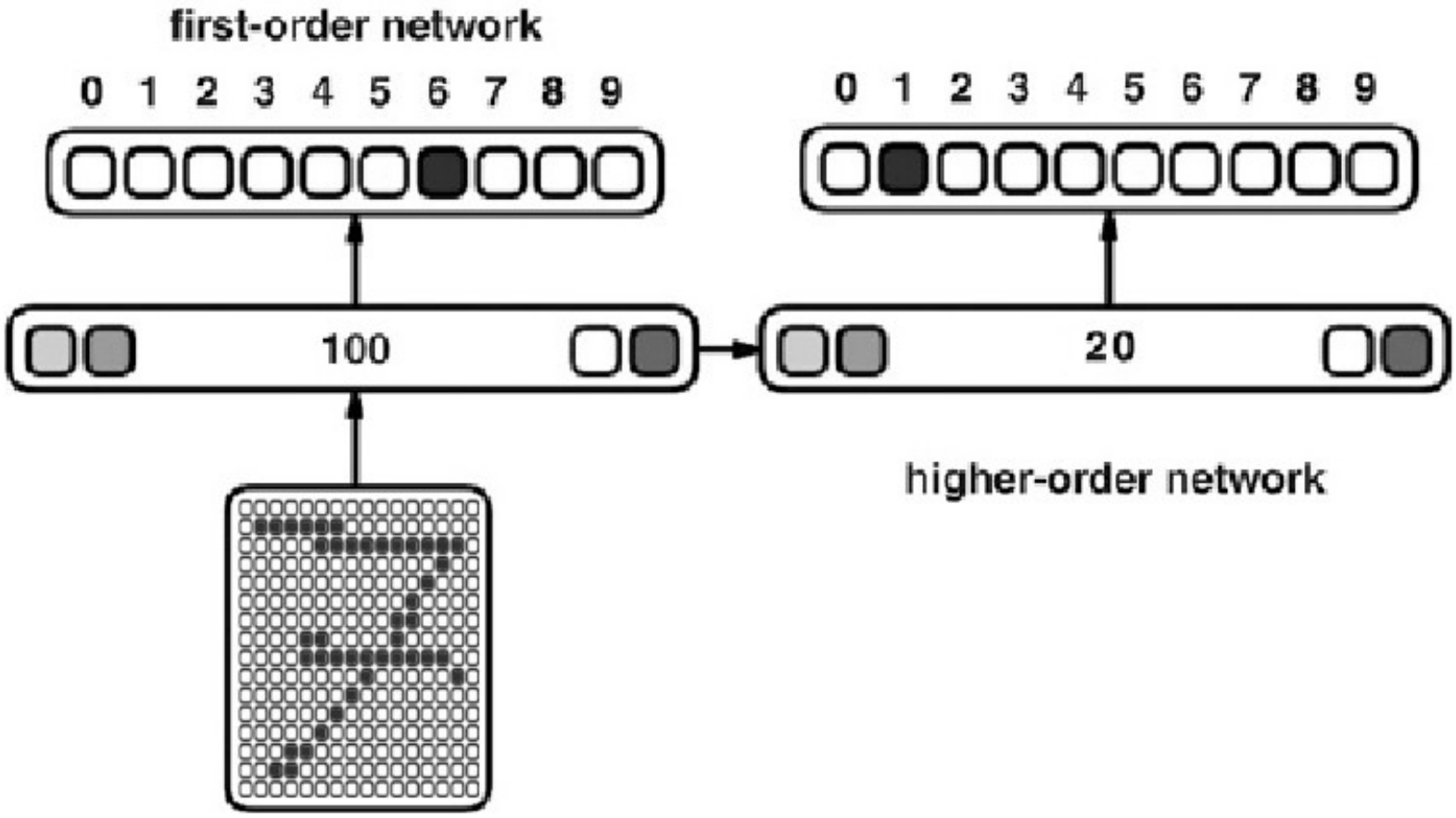
Après l'apprentissage sur
les 1600 formes

Valeur du neurone maximal

Avec chiffres manuscrits

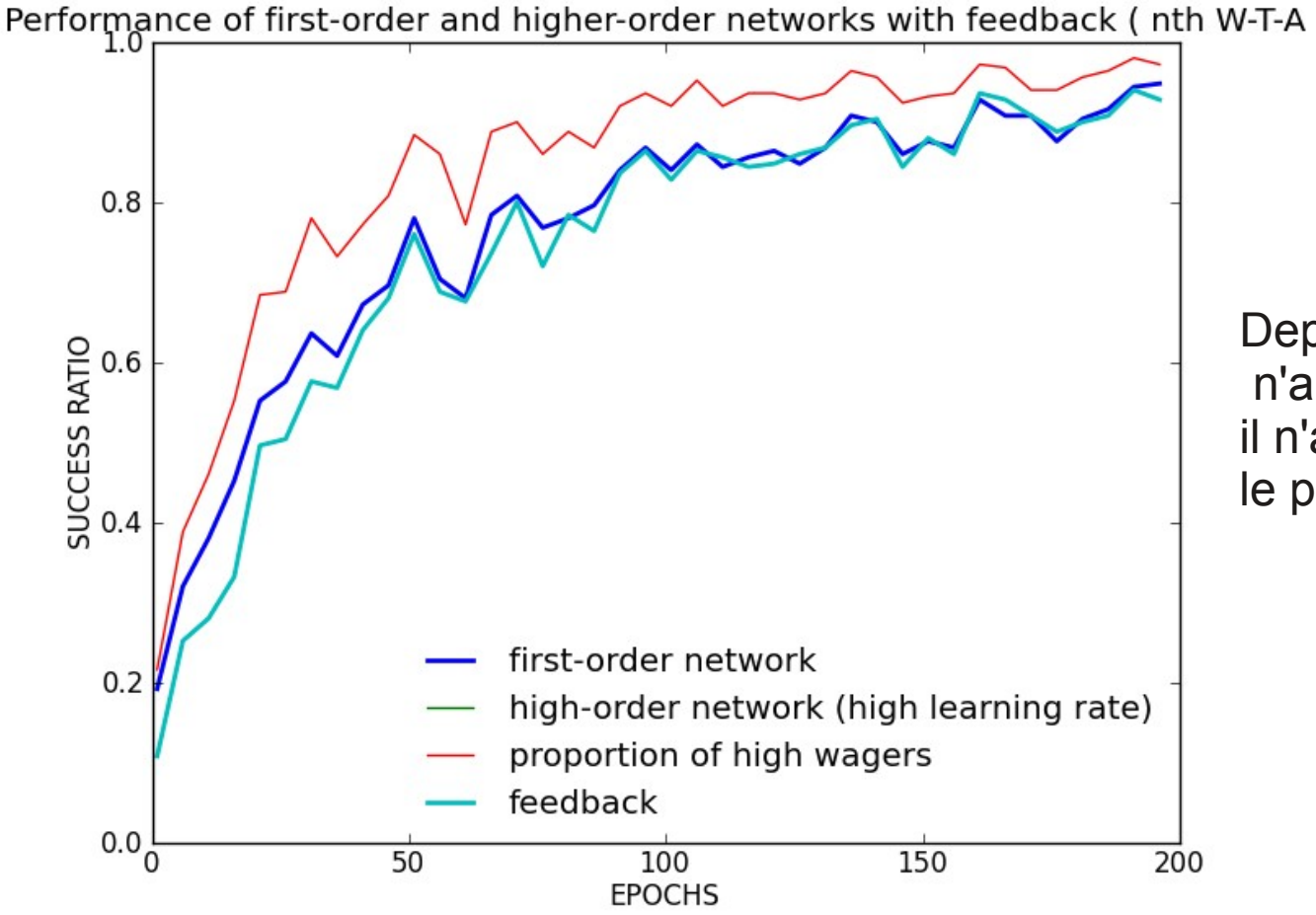


Feedback : n-ième neurone le plus actif



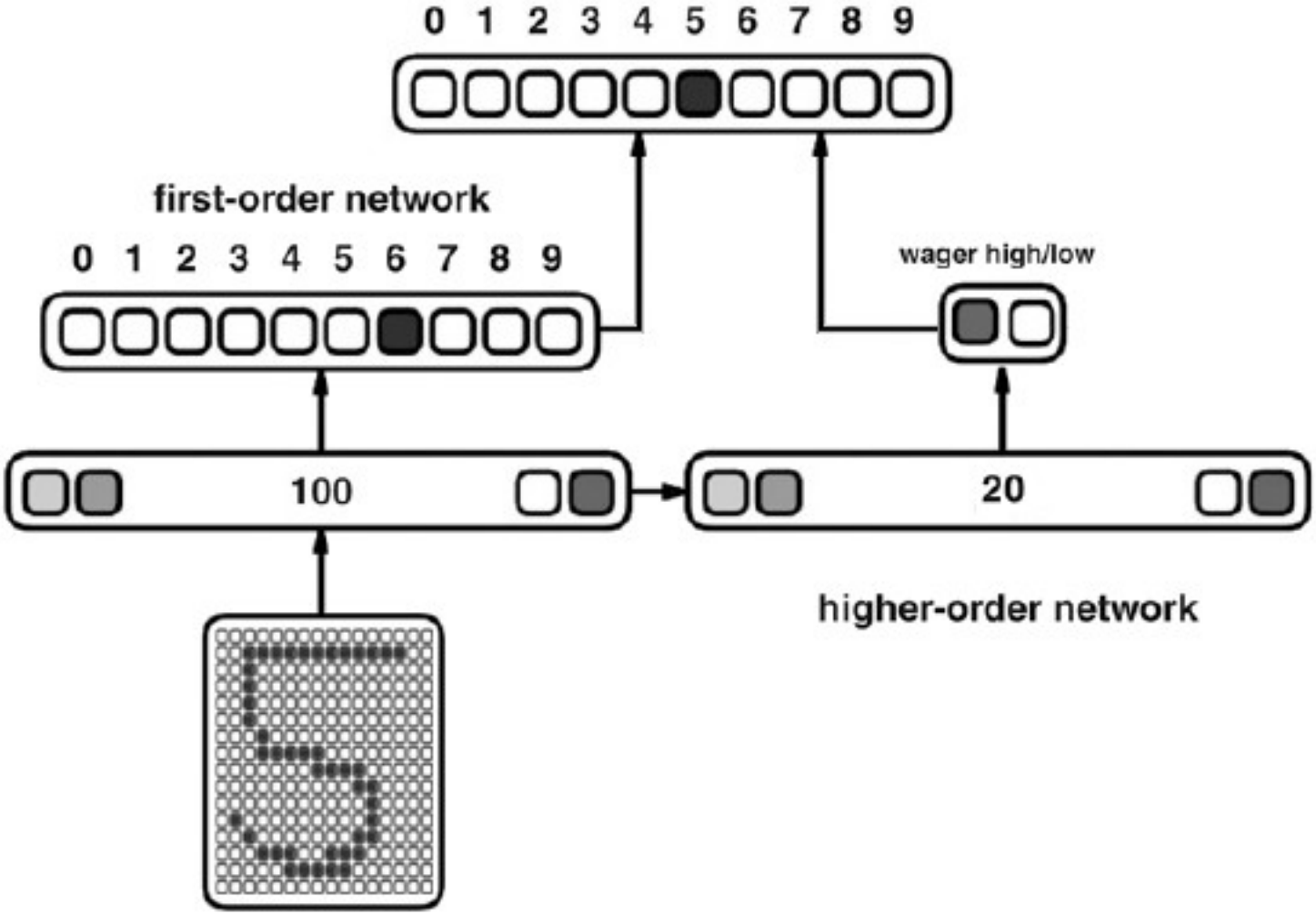
Feedback : n-ième neurone le plus actif

Avec chiffres manuscrits



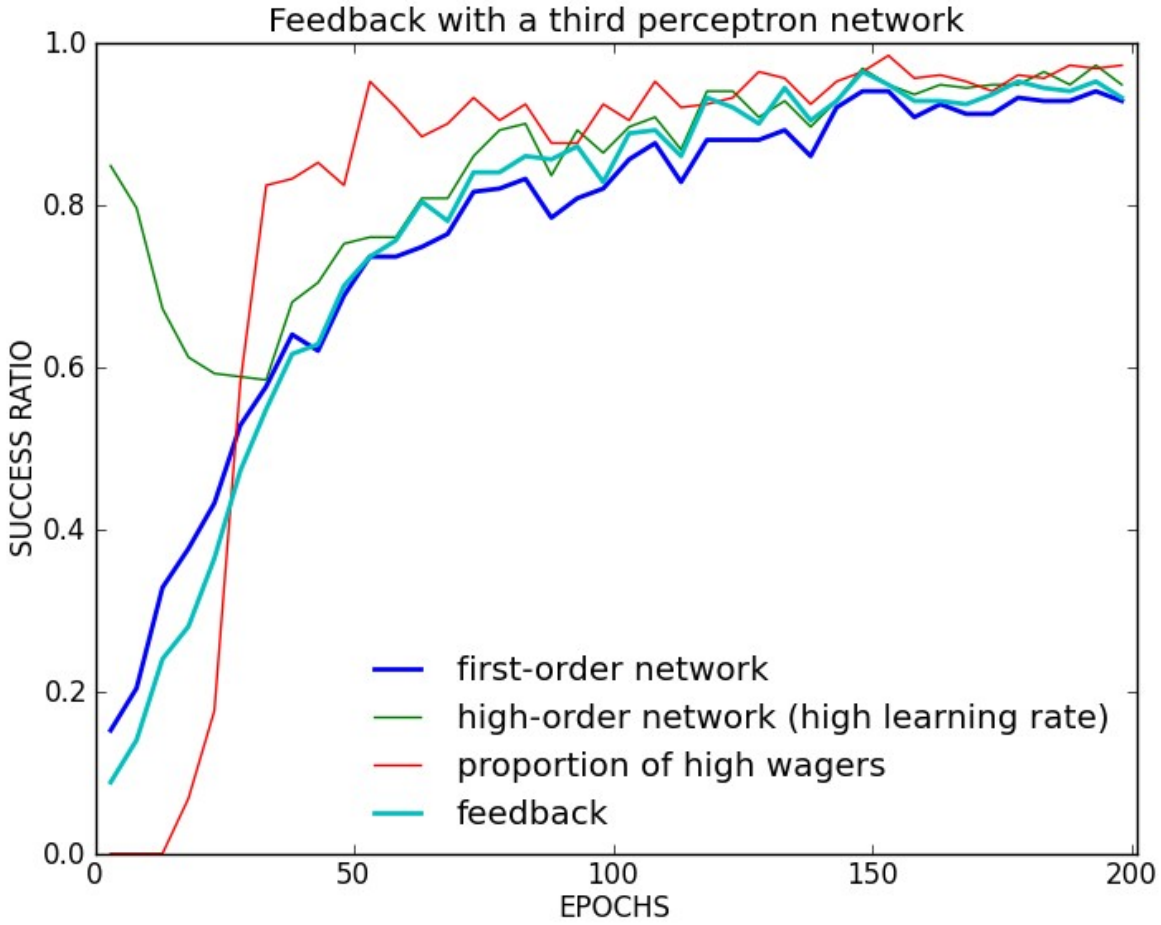
Depuis que le second réseau n'a plus 2 sorties mais 10 : il n'apprend pas plus vite que le premier réseau

Feedback : 3eme réseau



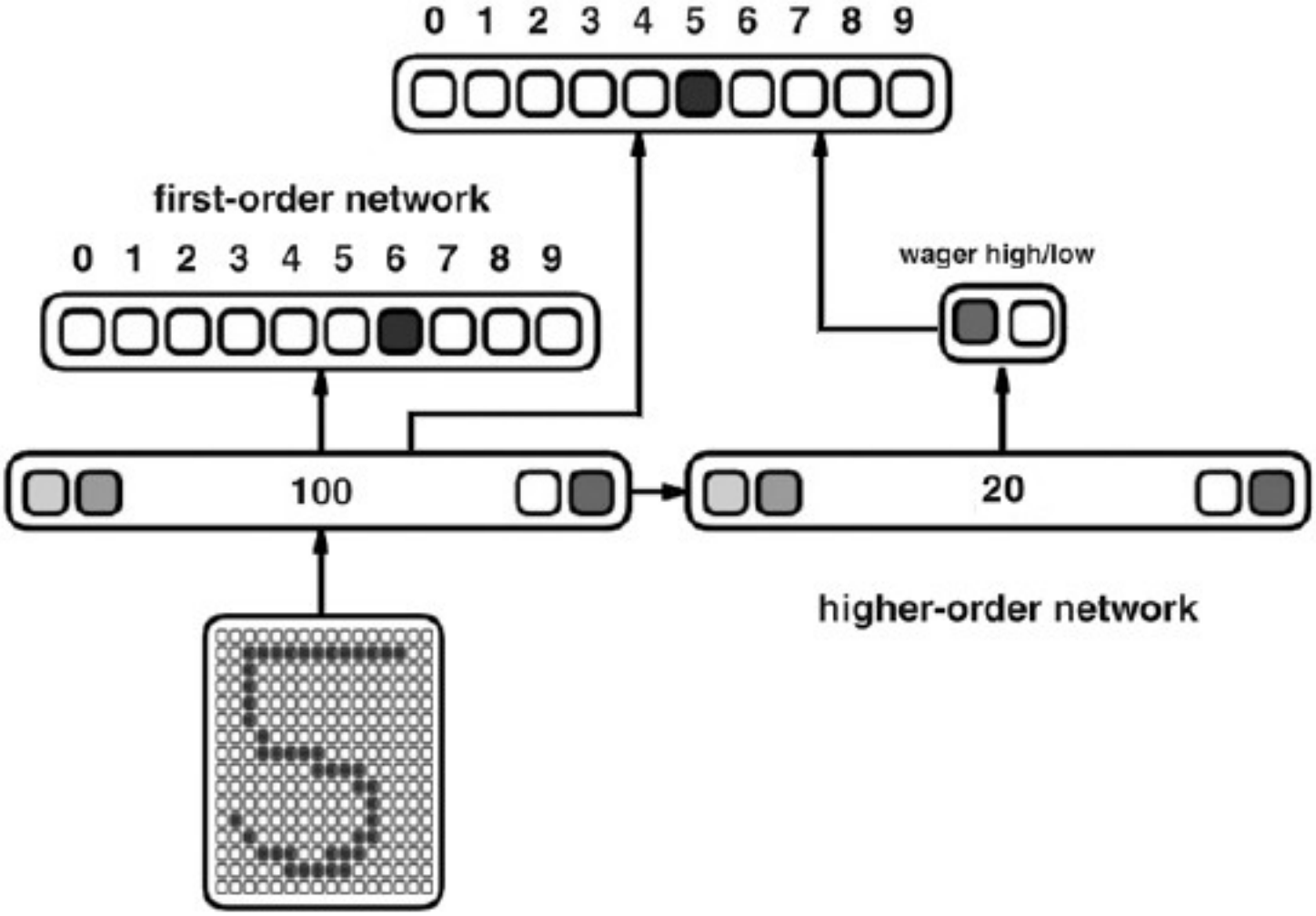
Feedback : 3eme réseau

Avec chiffres manuscrits



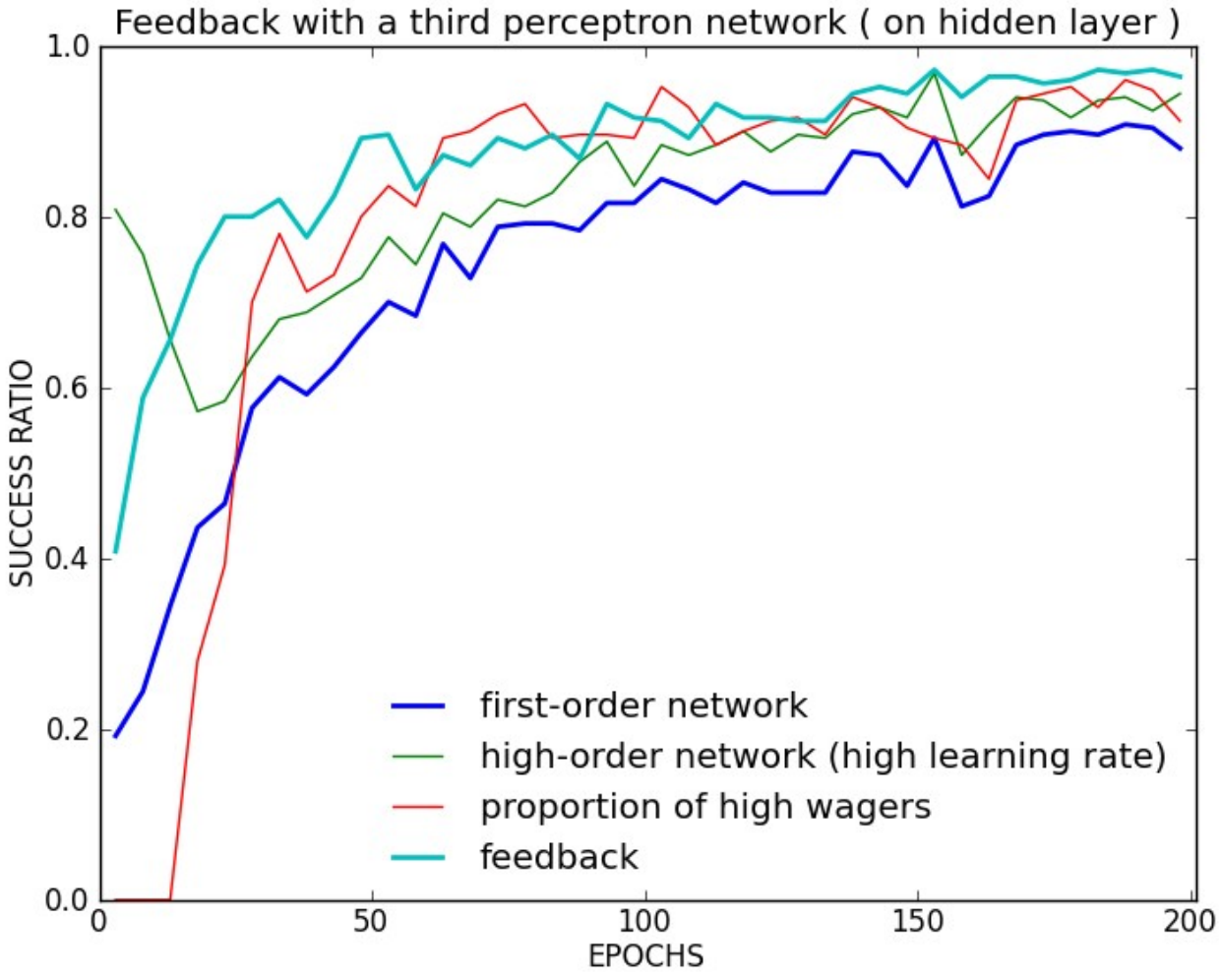
Les performances sont visiblement augmentées

Feedback : 3eme réseau (sur couche cachée)



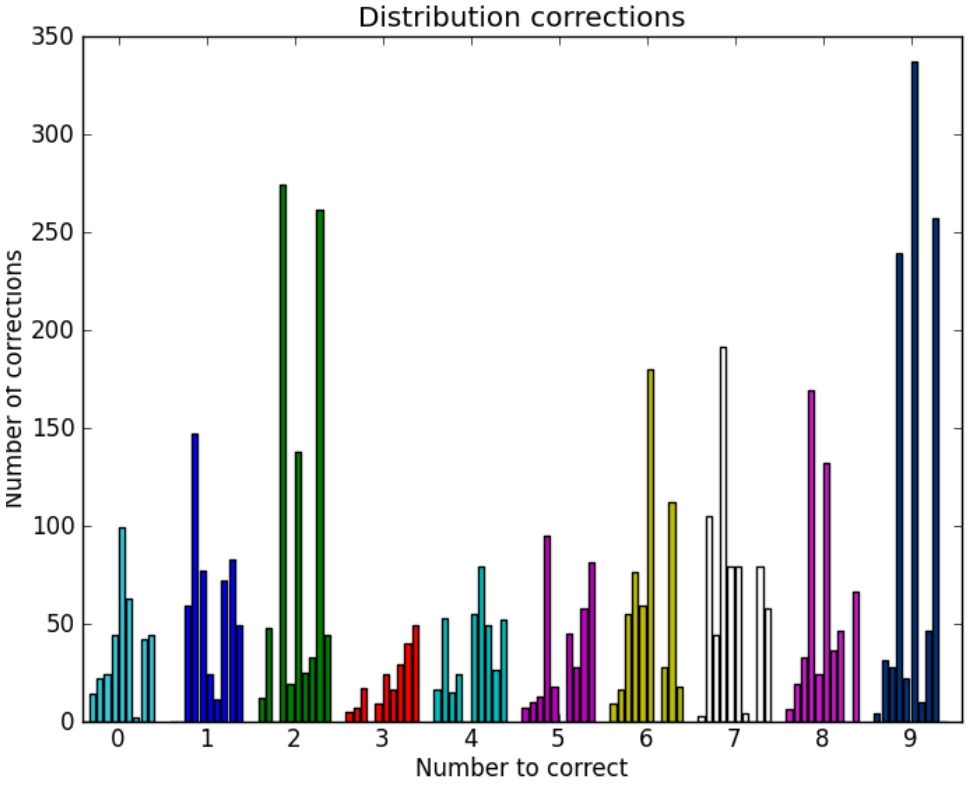
Feedback : 3eme réseau (sur couche cachée)

Avec chiffres manuscrits

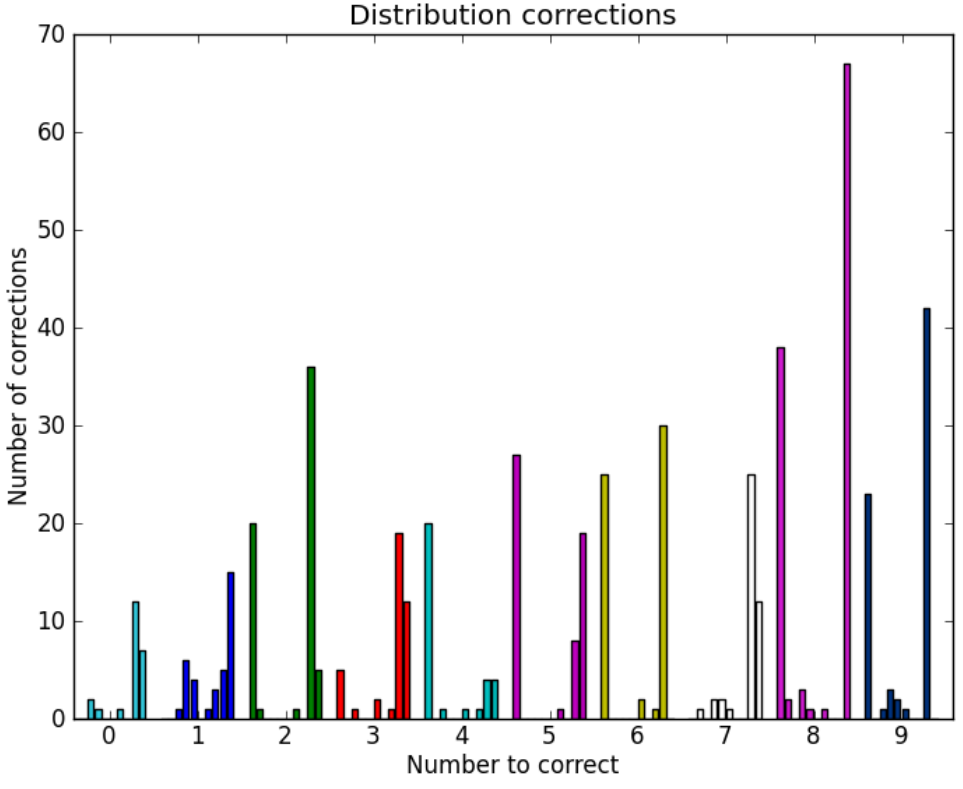


Les performances sont très nettement augmentées

Quelles corrections ?

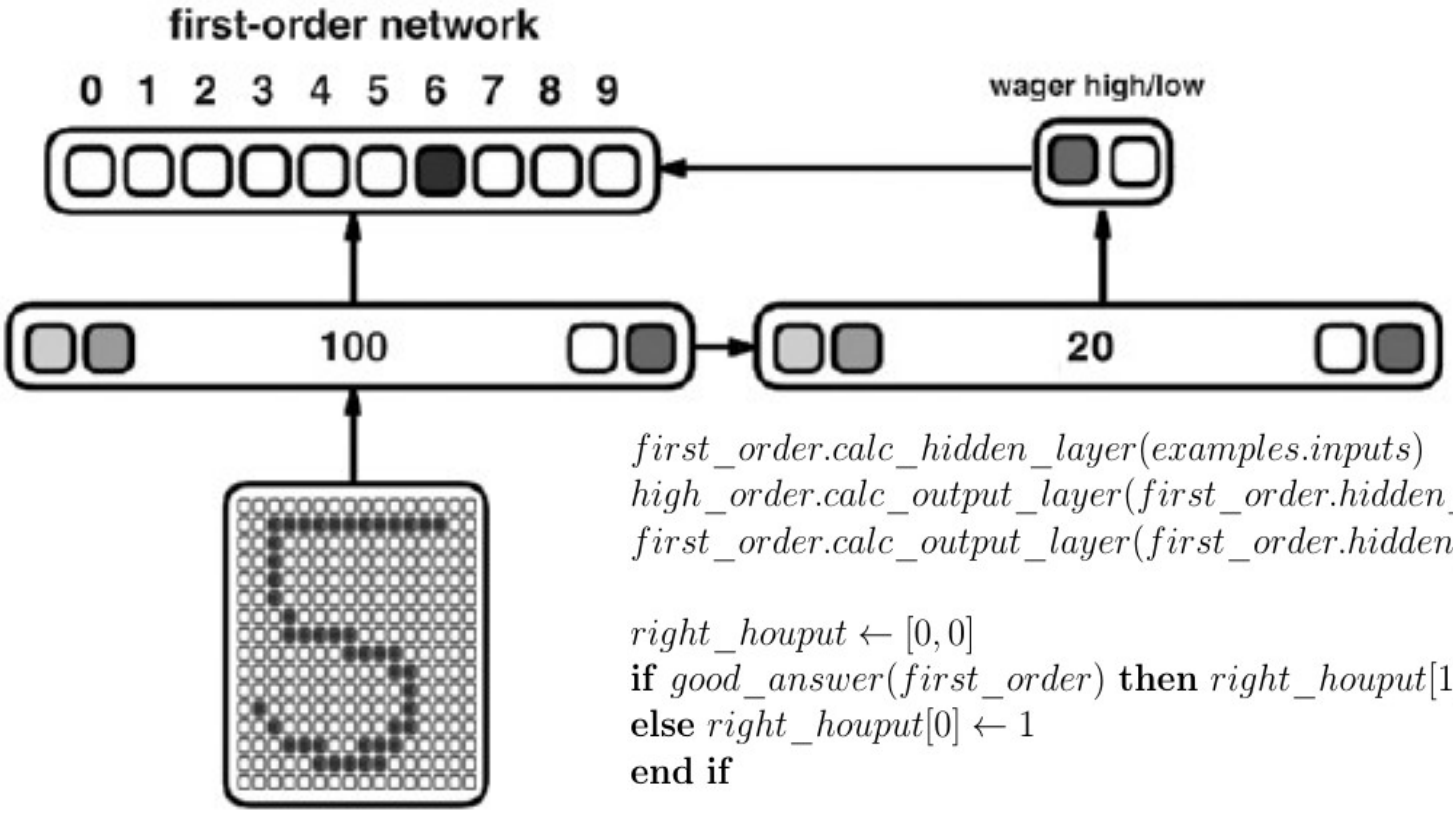


Durant tout l'apprentissage
(50 formes * 200 epochs)



Après l'apprentissage sur
les 1600 formes

Feedback : fusion



$first\_order.calc\_hidden\_layer(examples.inputs)$
$high\_order.calc\_output\_layer(first\_order.hidden\_layer)$
$first\_order.calc\_output\_layer(first\_order.hidden\_layer,\ [0, 0, ...0])$

$right\_houput \leftarrow [0, 0]$
if $good\_answer(first\_order)$ **then** $right\_houput[1] \leftarrow 1$
else $right\_houput[0] \leftarrow 1$
end if

$calc\_stats()$

$ah\_ouput \leftarrow ampli(high\_order.output\_layer)$
$high\_order.train(first\_order.hidden\_layer,\ right\_houput)$
$first\_order.train(examples.inputs,\ examples.outputs,\ ah\_output)$

Algo fusion

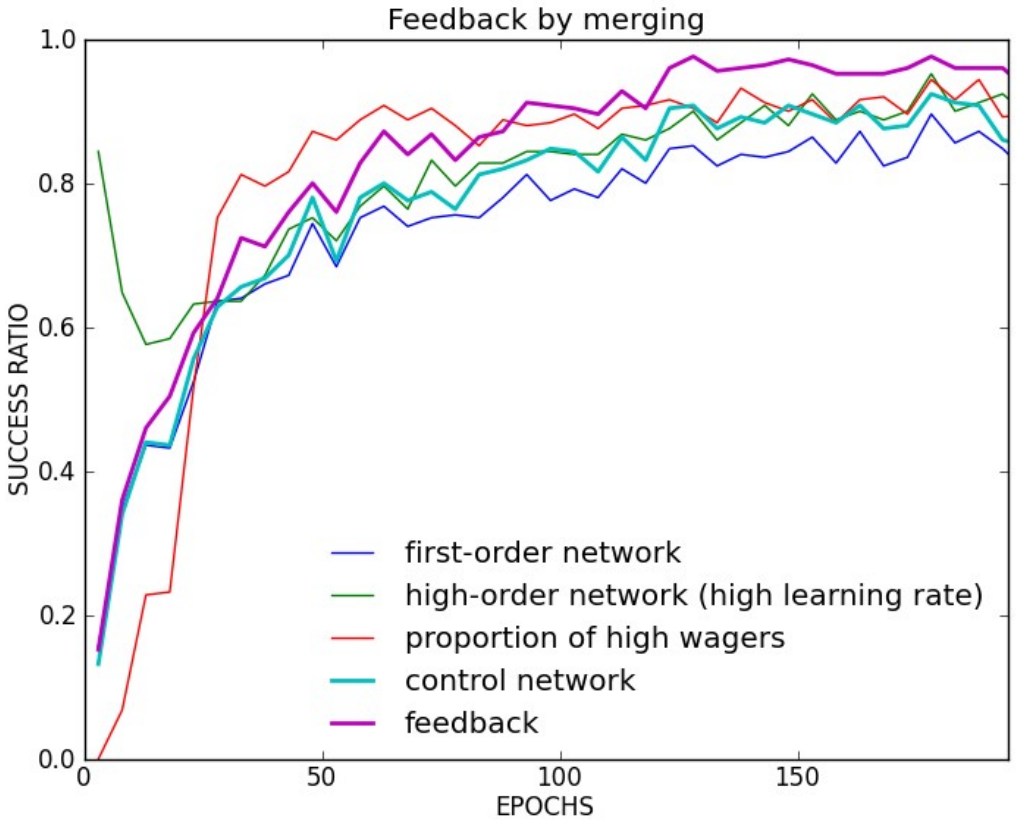
```
function train(inputs, outputs, add)
    y ← build_error_vector(...)
    update_weights_hidden_layer(...)

    for i = 0 → 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_gradient(error, intputs, add)
    calc_output(inputs + add)

    for j = 0 → inputs.length do
        dw ← weights[j] − last_weights[j]
        p ← error × inputs[j]
        weights[j] ← weights[j] + learning_rate × p + momentum × dw
    end for
end function
```

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

    for j = inputs.length → inputs.length + add.length do
        dw ← weights[j] − last_weights[j]
        p ← (goal − state) × add[inputs.length − j]
        weights[j] ← weights[j] + (learning_rate×p+momentum×dw) / add.length
    end for
end function
```

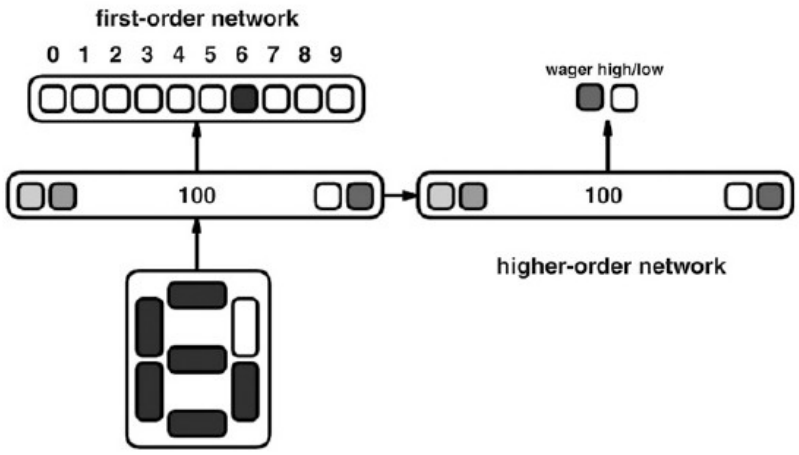
Feedback : fusion

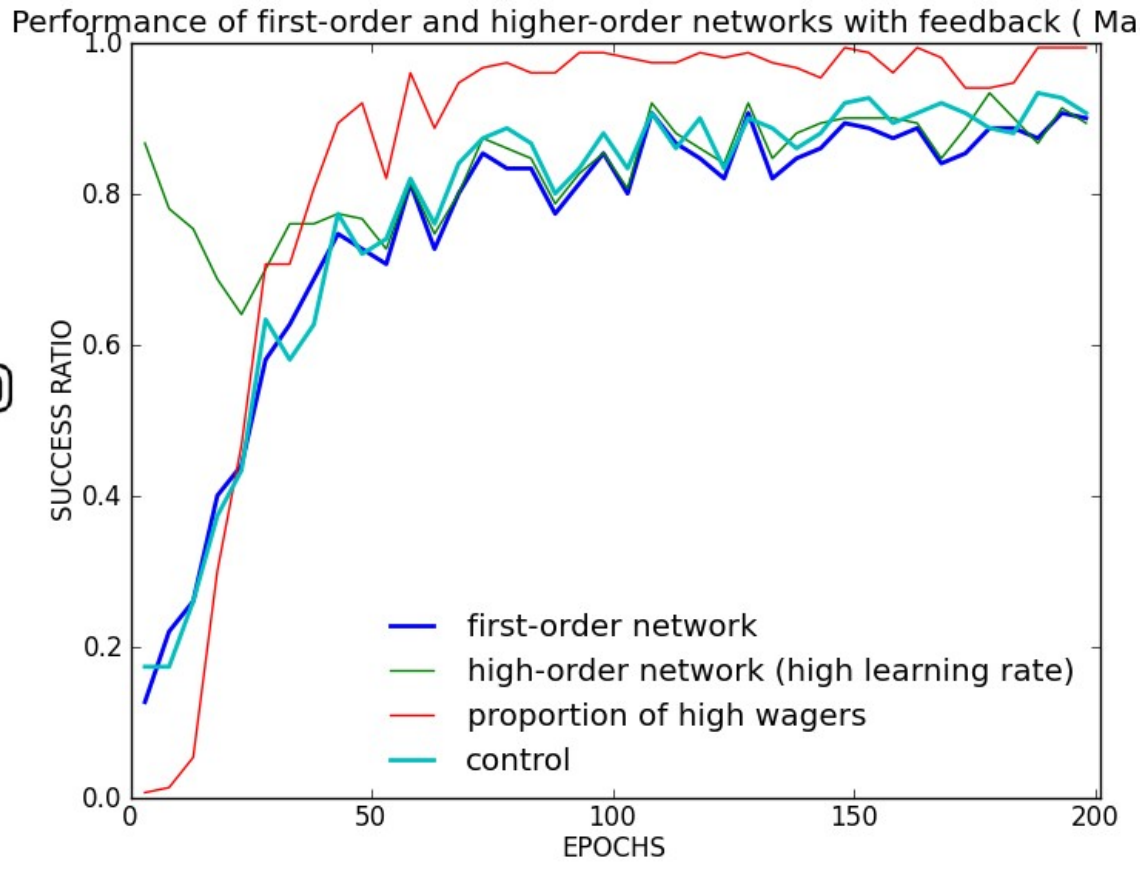
Avec chiffres manuscrits



Auto-supervisation

Avec chiffres manuscrits

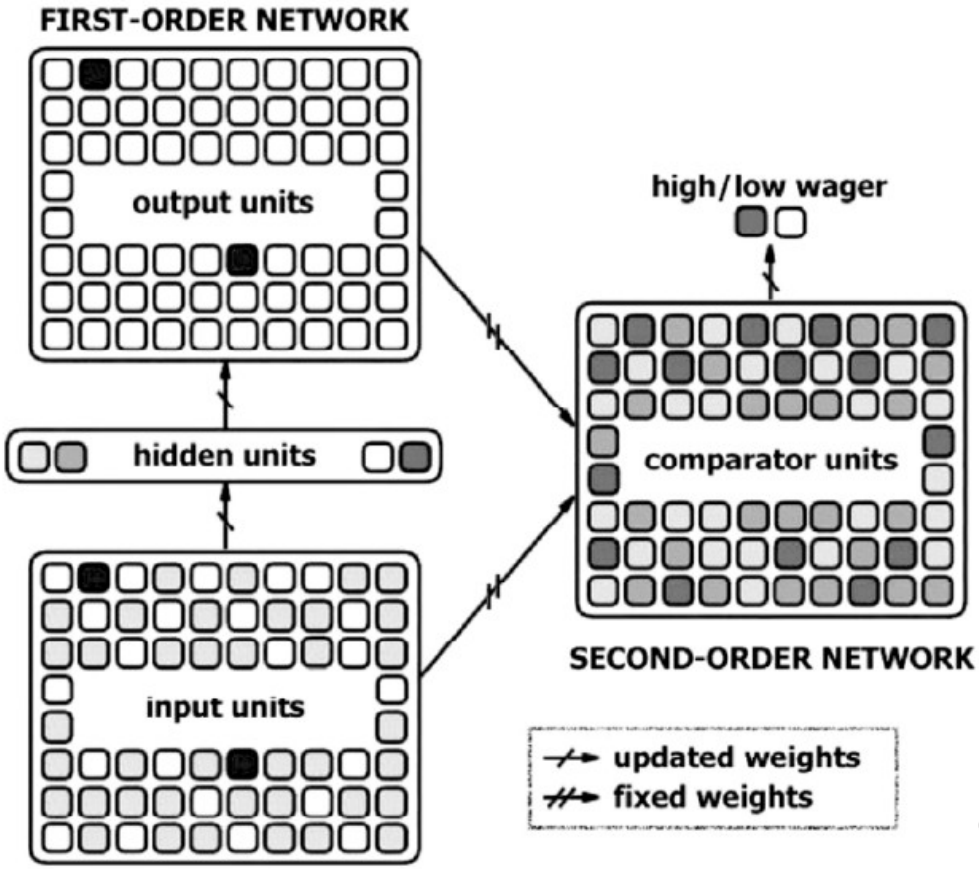




Vers où va t-on ?

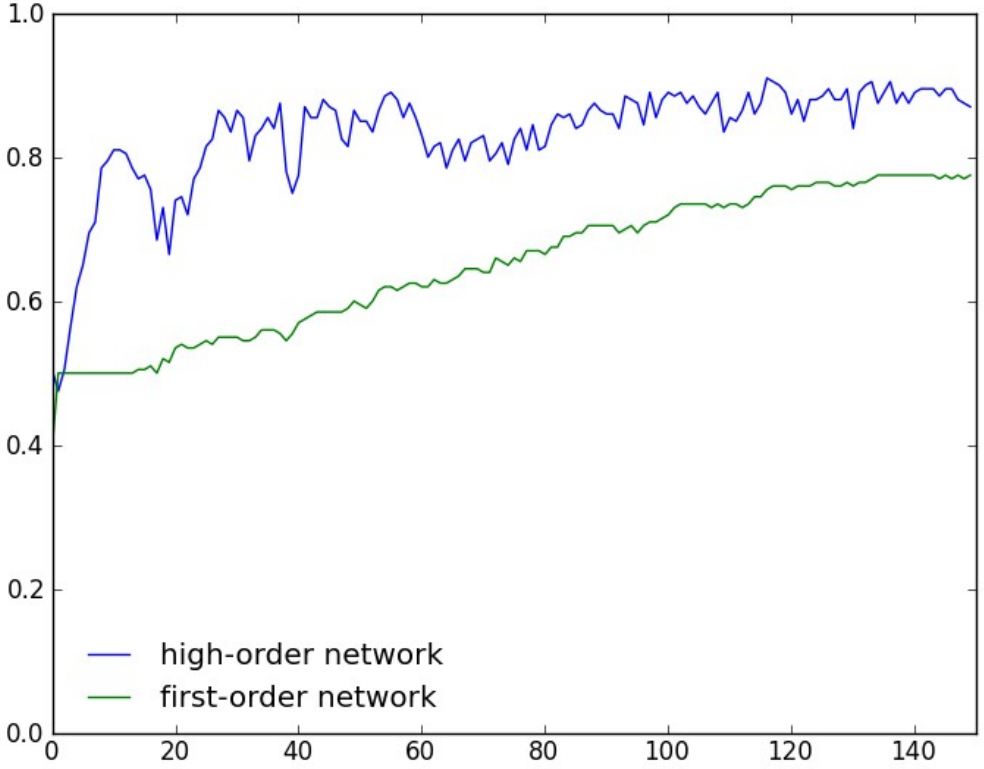
- ?

Simulation 3

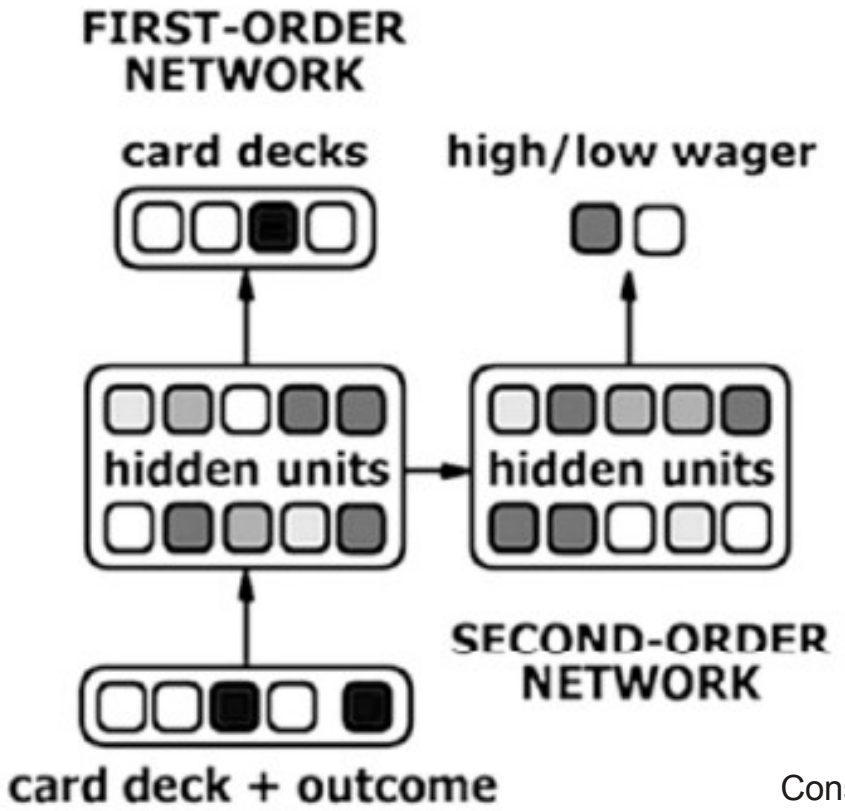


Consciousness and metarepresentation :
A computational sketch
[Alex Cleeremans, Bert Timmermans, Antoine Pasquali]

Simulation 3



Simulation 4



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

Simulation 4