Dominic JOBIN   
Dominic MICHAUD  
Groupe 0001

**Outils d’expérimentation**  
  
Travail présenté à M. François Bertrand  
Exploration des nouvelles technologies  
420-501-SF

Département de l’informatique  
Cégep de Sainte-Foy  
Vendredi le 17 novembre 2017

Table des matières

[**Démarche d’expérimentation** 2](#_Toc498690379)

[Réseau de neurones 2](#_Toc498690380)

[L’interface graphique et la grille 2](#_Toc498690381)

[Création des données 2](#_Toc498690382)

[Entraînement des réseaux 2](#_Toc498690383)

[Incertitudes et éléments inconnus 3](#_Toc498690384)

[**Devis techniques des outils** 4](#_Toc498690385)

[Outils utilisés 4](#_Toc498690386)

[Installation des outils utilisés 4](#_Toc498690387)

[**Exécutable, sources et documentation** 5](#_Toc498690388)

[Source 5](#_Toc498690389)

[Guide d’utilisation 5](#_Toc498690390)

# **Démarche d’expérimentation**

## Réseau de neurones

Nous avons commencé par chercher une librairie qui permet de créer un réseau de neurones. Nous avons essayé PyBrain, Theano et NeuroLab. Nous avons abandonné rapidement Theano parce qu’il est trop compliqué à utiliser. Nous avons utilisé PyBrain au départ.

## L’interface graphique et la grille

Nous avons utilisé Pygame et Image pour l’interface graphique. Ce fut plus facile pour nous de faire la grille et les interactions. La première dimension qu’on a donnée à la grille était de 48 carrés par 48 carrés. Nous avons rencontré des problèmes de performances avec l’apprentissage du réseau de neurones parce qu’il y avait trop d’éléments à traiter. Nous avons diminué la dimension de la grille à 16 carrés par 16 carrés.

## Création des données

Une fois l’interface graphique implémentée, nous avons ajouté une fonctionnalité qui nous a permis de rapidement dessiner et enregistrer des caractères en appuyant sur la touche correspondante. Par exemple, en dessinant un « 2 » et ensuite en appuyant sur la touche « 2 », les données de la grilles sont enregistrées dans un fichier texte en tant qu’une suite de « 0 » et de « 1 ». La grille est ensuite réinitialiser pour nous permettre de passer à un autre caractère rapidement. Nous avons ainsi dessiné plus de 2000 caractères

## Entraînement des réseaux

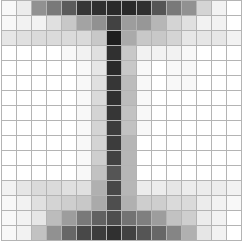
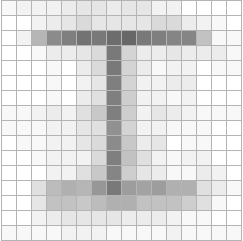
Nous avons donné l’information des caractères au réseau de neurones (appelé « cerveau »). Le premier essai était de lui faire apprendre 36 caractères dans un seul cerveau. Cela a échoué car il y avait trop d’éléments à apprendre en même temps.

La deuxième tentative fut de créer plusieurs cerveaux qui regroupent des caractères qui se ressemblent et un cerveau central qui fait le lien avec ces cerveaux. Nous croyions qu’avec cette manière de faire, que l’efficacité de l’apprentissage augmenterait à un niveau permettant d’avoir des résultats pertinents. Le réseau de neurones a fait du sur-ajustement, c’est-à-dire que les cerveaux donnaient les résultats attendues lorsqu’on lui donnait un caractère qu’il connaît, mais dès qu’on lui donnait un nouveau caractère, la liste de caractères proposé montrait des caractères inattendus (exemple : l’utilisateur dessine un « 8 » et le second choix proposé est un « M »).

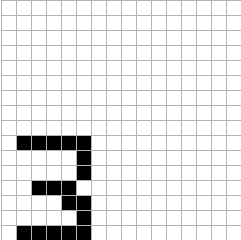
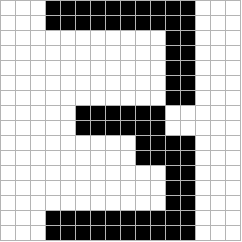
Nous avons ensuite tenté de faire plusieurs cerveaux qui regroupent des caractères qui se diffèrent. Dans cette expérimentation, il y a eu moins de sur-ajustement, mais la reconnaissance des caractères était faible (exemple : l’utilisateur dessine un « R » et le premier choix proposé est un « K » et le « R » est suggéré en quatrième choix).

Nous avons changé pour NeuroLab et avons ressayé ces étapes d’expérimentation. Avec les cerveaux qui regroupent des caractères similaires, nous avons eu des résultats qui correspondent à ce qu’on souhaitait.

Nous avons rencontré un problème lorsque nous avons tenté d’entraîner les cerveaux, les caractères ne se ressemblaient pas dû à leur positionnement et leur taille lorsque nous avons dessiné les 2000 caractères. Nous avons remédié à ce problème en redimensionnant et en centrant tous les caractères. Par exemple, voici la moyenne de tous les « I » que nous avions dessinés, avant et après l’ajustement :



De cette façon, le caractère est automatiquement redimensionné avant d’être envoyé dans les cerveaux, ce qui permet à l’utilisateur de dessiner un caractère de la taille qu’il veut, et où il désire. Par exemple, voici ce que l’utilisateur dessine (à gauche) et voici ce que les cerveaux reçoivent (à droite) :

Ce processus est entièrement transparent à l’utilisateur.

## Incertitudes et éléments inconnus

L’élément majeur qui reste incertain serait au niveau de l’algorithme d’entraînement des réseaux de neurones, qui a été en grande partie « caché » par les librairies d’intelligence artificielle utilisée (*PyBrain* et ensuite *NeuroLab*). Bien que nous soyons arrivés à des résultats satisfaisants en grande partie, nous n’avons pas eu à toucher au côté plus technique qui aurait nécessité du calcul avancé.

Du côté de la librairie Numpy, cette incertitude a été résolue car nous avons eu à chercher comment l’installer, configurer et utiliser pour bien manipuler les intrants et extrants des réseaux de neurones, tant au niveau de l’apprentissage que l’utilisation finale de l’application.

Notre expérimentation avec les réseaux de neurones ressemble beaucoup à ce que nous nous attendions : l’apprentissage selon des échantillons et les résultats des caractères qui se ressemblent (exemple : nous avions prédit que les réseaux de neurones confondraient les « O » et les « 0 »)

# **Devis techniques des outils**

## Outils utilisés

L’outil principal de cette recherche a été le langage de programmation Python 3.6, exécuté sous Windows 8.1 et Windows 10.

L’outil de reconnaissance de caractères a été développé dans l’environnement de développement intégré de PyCharm. De plus, les librairies suivantes ont été installées :

* *Pygame* pour l’interface graphique de l’application;
* *NeuroLab* pour les réseaux de neurones et l’entraînement;
* *Numpy* une dépendance de *NeuroLab*.

Nous avons également utilisé les librairies standards suivantes :

* *pickle* pour enregistrer et charger les objets de la classe *Brain;*
* *operator* pour trier les listes de résultats;
* *threading* pour entraîner plusieurs cerveaux à la fois.

## Installation des outils utilisés

Python 3.6 est disponible à l’adresse suivante :

<https://www.python.org/downloads/>

Il faut d’abord installer la libraire Numpy pour installer NeuroLab. Pour Windows, un fichier d’installation est disponible à l’adresse suivante :

<https://www.lfd.uci.edu/~gohlke/pythonlibs/#numpy>

Ensuite, il suffit d’exécuter la commande suivante pour installer NeuroLab et Pygame:

python –m pip install neurolab

python –m pip install pygame

# **Exécutable, sources et documentation**

## Source

Le code source python est disponible à l’adresse suivante :

<https://github.com/DomJob/LesDominics/raw/master/Outils.zip>

## Guide d’utilisation

Pour démarrer l’application, il faut double-cliquer sur le fichier **\_\_main\_\_.py** et l’interface s’affichera et permettra à l’utilisateur de dessiner sur la grille. Les résultats sont affichés à la droite de l’interface en temps-réel à mesure que le caractère est dessiné.

Pour entraîner de nouveaux cerveaux, l’application permet d’enregistrer les caractères dessinés simplement en dessinant le caractère et en appuyant sur la touche correspondante sur le clavier. Le caractère sera redimensionné, centré et enregistrer dans le fichier **data/characters.txt**. Le fichier zip fourni contient déjà le fichier de caractères qui a été utilisé lors de notre expérimentation.

Il faut ensuite exécuter les fichiers **brain/train.py** et **brain/train\_central.py** pour entraîner les cerveaux et le cerveau central, respectivement.