Annexe

On regroupe ici quelques informations diverses et variées.

1. Une brève histoire des réseaux de neurones

- **Neurone en biologie.** Entre 1840 et 1900 on découvre que la cellule est l'élément fondamental du vivant ; par observation Santiago Ramón y Cajal et d'autres mettent en évidence que les cellules du système nerveux forment un réseau : les neurones. On sait maintenant que le cerveau humain contient entre 80 et 100 milliards de neurones, mais nous avons aussi 500 millions de neurones dans le ventre (soit autant que dans le cerveau d'un chien). Un neurone est composé d'un élément central prolongé par un axone au bout duquel se trouvent les synapses. Les neurones sont organisés en réseau, un neurone étant en moyenne relié à 10 000 autres neurones et transmet ainsi un signal électrique à tous ses neurones voisins.
- **De la biologie vers l'informatique.** À la fin des années 1940, Donald Hebb émet l'hypothèse que lors de l'apprentissage certaines connexions synaptiques se renforcent et que ce renforcement persiste. Un modèle mathématique est déduit par Frank Rosenblatt : le perceptron (1958). L'algorithme est mis en œuvre dans une machine dédiée à la reconnaissance d'images. Le perceptron, qui correspond à un réseau formé d'un seul neurone, montre très vite ses limites à la fois théoriques (puisqu'il ne peut réaliser le « ou exclusif ») et aussi en termes de résultats pratiques. Ces deux problèmes sont résolus en considérant des réseaux à plusieurs couches, mais les calculs à mener sont trop longs à la fois en raison de la puissance des ordinateurs de l'époque et des algorithmes utilisés.
- **Rétropropagation.** Un progrès important est fait grâce l'algorithme de rétropropagation (par Rumelhart, Hinton et Williams en 1986) qui permet un calcul efficace des poids des neurones. On conserve encore aujourd'hui ce principe : des calculs essentiellement numériques, avec un minimum de calculs symboliques au niveau de chaque neurone (afin de calculer la dérivée), le tout avec un grand nombre d'itérations. Des améliorations se succèdent : pooling, dropout, calculs en parallèle, meilleures descentes de gradient.
- **Hiver.** Cependant les attentes de l'intelligence artificielle (peut être le nom a-t-il été mal choisi ?) sont largement déçues car ses applications restent limitées. L'intérêt des scientifiques diminue drastiquement. De 1980 à 2000 on parle de l'hiver de l'intelligence artificielle.
- ***Deep learning*.** À partir des années 2000 et surtout après 2010 les réseaux de neurones font des progrès fulgurants grâce à l'apprentissage profond. Yann Le Cun démontre l'efficacité des couches de convolution pour la reconnaissance des chiffres. On réalise et entraîne alors des réseaux ayant de plus en plus de couches grâce à des progrès matériels (par exemple le calcul sur les processeurs graphiques GPU) mais surtout grâce aux couches de convolution qui extraient des caractéristiques abstraites des images.
- **Présent et avenir.** Les réseaux de neurones s'appliquent à de nombreux domaines : la reconnaissance d'images (par exemple la détection de cancer sur une radiographie), les transports (par exemple la

conduite autonome des voitures), les jeux (les ordinateurs battent les champions du monde d'échecs, de go et des jeux vidéos les plus complexes), l'écriture (classement, résumé, traduction). Il y a cependant une méfiance vis à vis des décisions prises par une machine (sentence de justice, diagnostic médical, publicité ciblée). Une meilleure compréhension du fonctionnement des réseaux de neurones par tous est donc indispensable !

2. Références

- *tensorflow* est développé par *Google*. De nombreux tutoriels sont disponibles pour les débutants :
  tensorflow.org
- *keras* est maintenant intégré à *tensorflow* et facilite sa prise en main. Il a été développé par François Cholet auteur du livre *Deep learning with Python* (Manning publications, 2017). Il y a de nombreux exemples d'application :
  keras.io
- Le livre *Deep learning /L'apprentissage profond* par Goodfellow, Bengio, Courville contient des concepts plus avancés. Il est disponible en anglais et en français. Une version gratuite est consultable ici :
  deeplearningbook.org
- Vous pouvez récupérer l'intégralité des codes *Python* ainsi que tous les fichiers sources sur la page *GitHub* d'Exo7 : « GitHub : Deepmath ».
  En particulier vous trouvez sur ce site le module `keras_facile` qui vous aide à définir facilement des poids pour un réseau simple.
- Un peu de pub pour les livres Exo7 :
  — *Algèbre* et *Analyse* pour toutes les notions de mathématique niveau première année,
  — *Python au lycée* (tome 1 et tome 2) pour apprendre la programmation.
  Ils sont disponibles gratuitement en téléchargement :
  exo7.emath.fr et GitHub : exo7math
  et aussi en vente à prix coûtant sur amazon.fr.

3. Ce qu'il faut pour utiliser tensorflow/keras

Les activités de ce livre sont écrite pour *Python*, version 3. Cependant il faut installer un certain nombre de modules complémentaires qui ne sont pas présents par défaut.

- `tensorflow` qui contient le sous-module `keras`,
- `numpy` pour les tableaux,
- `matplotlib` pour l'affichage de graphiques,
- `scipy` pour la convolution,
- `ioimage` pour gérer la lecture et l'écriture d'images.

Un module s'installe simplement par :

```
pip install mon_module
```

Il existe des distributions *Python* (du type `conda`) qui contiennent les principaux modules scientifiques et permettent aussi de créer des « environnements » afin de gérer plusieurs versions de *Python* et de ses modules.

Pour ceux qui ne souhaitent rien installer, il est possible d'utiliser *tensorflow* en ligne :
Google Colab

4. Lexique français/anglais

La plupart des références à propos des réseaux de neurones sont en anglais. Voici donc un petit lexique français/anglais des termes principaux.

Réseau de neurones. *Neural network.*

- neurone artificiel/*artificial neuron*
- poids/*weights*, biais/*bias*
- fonction d'activation/*activation function*, fonction marche de Heaviside/*Heaviside step function*, fonction sigma σ ou sigmoïde/*sigmoid function*
- fonction d'erreur/*loss function* ou *cost function*
- entrée/*input*, sortie/*output*
- couches du réseau/*layers*
- réseau de neurones avec convolution/*convolutional neural network* abrégé en *cnn*
- un motif de convolution s'appelle aussi noyau/*kernel* ou filtre/*filter* ou masque/*mask*
- *dropout* se traduit par abandon ou décrochage
- *pooling* se traduit pas regroupement de termes

Mathématiques

- dérivée partielle/*partial derivative*
- gradient/*gradient* mais la notation anglo-saxonne de grad $f(x,y)$ est $\nabla f(x,y)$
- régression linéaire/*linear regression*
- tensor/*tensor*, taille/*shape*, nombre d'éléments/*size*

Descente de gradient. *Gradient descent.*

- descente de gradient (classique)/*(batch) gradient descent*
- descente de gradient stochastique/*stochastic gradient descent* abrégée en *sgd*
- descente de gradient par lots/*mini-batch gradient descent*
- pas δ/*learning rate*
- erreur quadratique moyenne/*minimal squared error* abrégée en *mse*
- moment/*momentum*
- époque/*epoch*

5. L'image de couverture

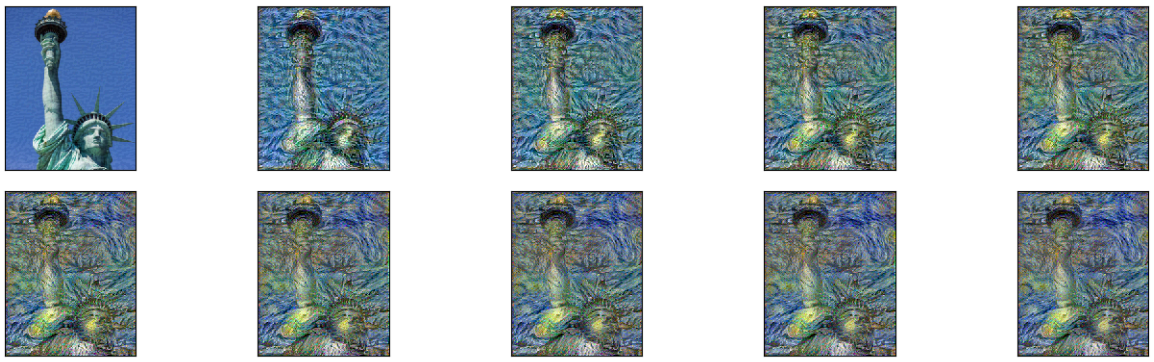
L'image de couverture du livre est générée automatiquement grâce à un réseau de neurones et la rétropropagation. Au départ il faut deux images :

- une image dont on veut conserver le contenu : ici la statue de la liberté,
- et d'une image dont on veut conserver le style : ici un tableau de Van Gogh.

L'image obtenue a le contenu de la première et le style de la seconde !

Références :

- Statue de la liberté (photographie par George Hodan, licence CC0, domaine public).
- Peinture de Vincent Van Gogh, *La nuit étoilée*, 1889.
- Le programme et les explications pour *tensorflow* : *Neural Style Transfer* avec TensorFlow d'après des travaux de Gatys, Ecker et Bethge.



Plus en détails :

- le réseau de neurones, nommé « VGG19 », est un réseau à 19 couches. On ne va pas calculer les poids, mais on utilise des poids qui ont déjà été calculés pour reconnaître les images de la base *ImageNet*.
- Dans ce réseau bien paramétré certaines couches se concentrent sur le contenu (contour, lignes, formes…) et d'autre se concentrent sur le style (couleur, flou…).
- En identifiant ces couches, on construit par itérations une image qui conserve le contenu de la première image et prend peu à peu le style de la seconde.
- Ces itérations se font par rétropropagation et descente de gradient selon une fonction d'erreur qui est la somme d'une fonction d'erreur associée au contenu et d'une fonction d'erreur associée au style.