HLIN601

Projet de Programmation

—

Simulateur Deep

—

Melvin Bardin
Maël Bonneaud
Malika Lin-Wee-Kuan

—

Encadré par M. Poncelet

—

2019-2020

Table des matières

Première partie

Présentation du projet

1 Introduction

Dans le cadre du module HLIN601 de Projets de Programmation de L3, nous nous sommes orientés vers le sujet "Simulateur Deep". Le sujet de M. Poncelet sur le Deep Learning nous a intéressé car il s'agissait d'un sujet dont on ne connaissait que très peu de choses et nous étions curieux d'en apprendre d'avantage sur celui-ci.
En effet les réseaux de neurones sont comme une boite noire, elles prennent des données, retournent un résultat, mais en aucun cas nous ne connaissons la méthode employée. Notre objectif était donc de réaliser une application Web permettant de simuler le comportement d'un réseau de neurones, pour en expliciter le comportement, mieux le saisir et découvrir les subtilitées.
Le code ainsi qu'une vidéo de démonstration sont disponibles à l'adresse :
" https://github.com/Kyrial/Simulateur_Deep "

2 Les Réseaux de neurones

2.1 L'histoire des réseaux de neurones

Les réseaux de neurones sont de plus en plus présent en informatique, que se soit pour la reconnaissance faciale à la classification des données en passant par la prédiction météorologique. Ils existent réellement que depuis le XX$^{\text{ème}}$ siècle. On pourrait alors se demander quand est ce que les réseaux de neurones ont commencés ? La première théorie avec le neurone formel est apparu en 1943 et il a fallu attendre les années 2000 pour voir les prémisse dans l'informatique.

2.2 Rappel sur les réseaux de neurones

Interrogeons nous sur le fonctionnement d'un réseau de neurone qui sera important pour la compréhension de notre projet. Un neurone, comme en biologie, est une cellule qui va prendre en entrée des signaux (ici une donnée) et qui va renvoyer en sortie des signaux différents (via ce qu'on peut appeler des synapses). Un réseau de neurone c'est une organisation de plusieurs neurones.

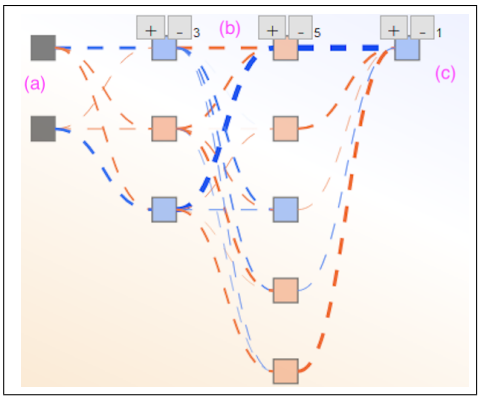

FIGURE 1 – Les différentes types de couches

La figure 1 illustre un réseau de neurones composés de 4 *layers*. Le premier *layer* (généralement noté *Layer 0*) et appelé *input layer* correspond à la couche d'entrée du réseau et contiendra les variables prédictives. Le dernier correspond à l'*output layer* et donne le résultat de la classification. Au milieu nous avons les *Hidden layer* qui eux vont permettre de modifier les valeurs récupérées et c'est là que l'apprentissage du réseau de neurone s'effectue.

2.2.1 Apprentissage du réseau de neurone

Pour apprendre, le réseau de neurone va avoir besoin de plusieurs tentatives et de nombreuses données. En effet les signaux (ou données) envoyés entre chaque synapses va être modifié avec ce qu'on appelle un poids, qui va selon sa valeur activer ou non le neurone en récupérant le signal.
Ensuite le neurone va appliquer à ce signal un biais qui permettra de favoriser (ou pas) l'activation de la synapse dans certains cas. Après, chaque neurones a ce qu'on appelle une fonction d'activation qui va permettre de déterminer l'activation (ou pas) de la synapse.
Il existe de nombreuses fonction d'activation comme la "sigmoïde" et la "ReLU" que nous utilisons dans notre projet. Ce signal est ensuite transmis au neurone suivant jusqu'à arriver au dernier layer (*"Output Layer"*) de sortie.

Dans un premier temps le jeu de donnée est divisé en deux parties, les données d'évaluation servirons de vérifications des performance du modèle tout au long des calculs et les donnée d'apprentissage permettrons l'apprentissage de ce modèle.
Pour apprendre, le réseau va donc en premier lieu utiliser des valeurs aléatoires car il ne saura pas quel poids appliquer pour arriver à un résultat correcte. Lors du premier passage (Forward propagation), la valeur de sortie est fausse par rapport au résultat attendu. Nous allons donc appliquer ce qu'on appelle une fonction de coût qui sera plus où moins élevée si on est éloigné ou non du résultat attendu. Cette fonction s'appliquera sur chaque neurone dans le sens inverse (Backward propagation). Chaque nouvelle propagation est appelée époque (*epoch*).
Ainsi nous cherchons à avoir un coût le plus faible possible pour s'approcher du résultat attendu. C'est pour cela que nous utilisons la technique de la descente de gradient qui utilise la dérivée partielle afin d'obtenir le plus petit coût possible pour chaque neurone. Cette fonction est associée à un taux d'apprentissage (*learning rate*) qui correspond à la vitesse de convergence de la descente de gradient vers le minimum de la fonction coût. Ainsi le réseau de neurone apprendra petit à petit.
Notre objectif consiste à partir de tous les composants d'un réseau de neurones de mettre en évidence

comment se propagent les informations (mises à jour des poids) pour parvenir à une classification.

3 Technologies utilisées

Durant notre projet nous utilisé de nombreux outils technologies répondant à nos besoins. Nous allons voir l'utilisation de Jypyter, Python, My Neural Network et enfin HTML.

3.1 Jupyter notebook

Dans un premier temps nous avons utilisé Jupyter notebook. Le bloc-notes Jupyter [1] est une application Web open source qui permet de créer et de partager des documents contenant du code en direct, des équations, des visualisations et du texte narratif. Nous l'avons donc principalement utilisé pour la présentation des algorithmes en lien avec les réseaux de neurones.

3.2 Python (Flask)

Ensuite tous les algorithmes générant les réseaux de neurones utilisés lors de notre projet utilisent Python. Cette technologie est un langage de programmation contenant de nombreuses librairies qui nous ont étaient très utiles pour notre projet. Nous avons donc pour la conception de notre site utilisé une libraire Python dont le nom est Flask [2] , qui nous a permit de mettre en place un serveur Python afin d'exécuter nos algorithmes.
Après avoir testé différentes technologies pour exécuter nos script Python, nous nous sommes orienté vers Flask pour héberger notre serveur car c'est celui qui correspondait au mieux à nos besoins. En effet Flask est compatible avec les machines de la faculté et il est léger, ce qui est parfait pour notre projet qui était déjà un peu lourd de base.

3.3 My Neural Network

My Neural Network[3] est un groupement de fonctions Python utilisant les outils de calculs de réseaux de neurones (Keras, Tensorflow, etc..) fournit par notre Référent permettant d'expliciter les étapes de calcul pour mieux cerner leurs fonctionnements et leurs utilités.

3.4 HTML / CSS / JAVASCRIPT

Concernant les technologies Web utilisées pour la conception de notre site nous avons utilisés HTML et CSS pour la partie *frontend* (visuelle : conception / interface) du projet. Ensuite nous avons utilisé principalement Javascript pour modifier le DOM et utiliser les données renvoyées par le serveur afin de construire le réseau de neurone et afficher les données, les courbes et la simulation du fonctionnement du réseau de neurone. Concernant Javascript nous avons également utilisé AJAX afin d'empêcher le rechargement de la page Web lors de l'exécution des algorithmes Python.

1. https ://jupyter.org/[dernière consulation le 23/04/2020].
2. https ://flask.palletsprojects.com/ [dernière consulation le 23/04/2020].

Deuxième partie
Développements Logiciel

4 Conception, Modélisation et Implémentations

4.1 Conception

La conception de notre site Web s'est axé sur un serveur qui effectue les calculs et une page web qui récupère les résultats et les affiches graphiquement.

4.2 Entrées et paramètres de l'application

À l'exécution, aucune entrée n'est nécessaire au bon fonctionnement du site Web. Lorsqu'une requête est envoyée au serveur pour effectuer des calculs, tous les paramètres ainsi que la configuration du réseau de neurones sont passées en paramètre grâce à un formulaire qui vérifie l'intégrité des données.
Après le traitement, le serveur retourne les résultats de l'algorithme sous forme de trois dictionnaires. Ces derniers correspondent à l'historique des calculs permettant d'effectuer l'animation des neurones ainsi que de dessiner les courbes de précision et de coût. L'image s'affichant en fin d'animation n'est pas retournée avec la réponse du serveur, elle est sauvegardée sur la machine puis est récupérée par la page Web.

5 L'interface graphique

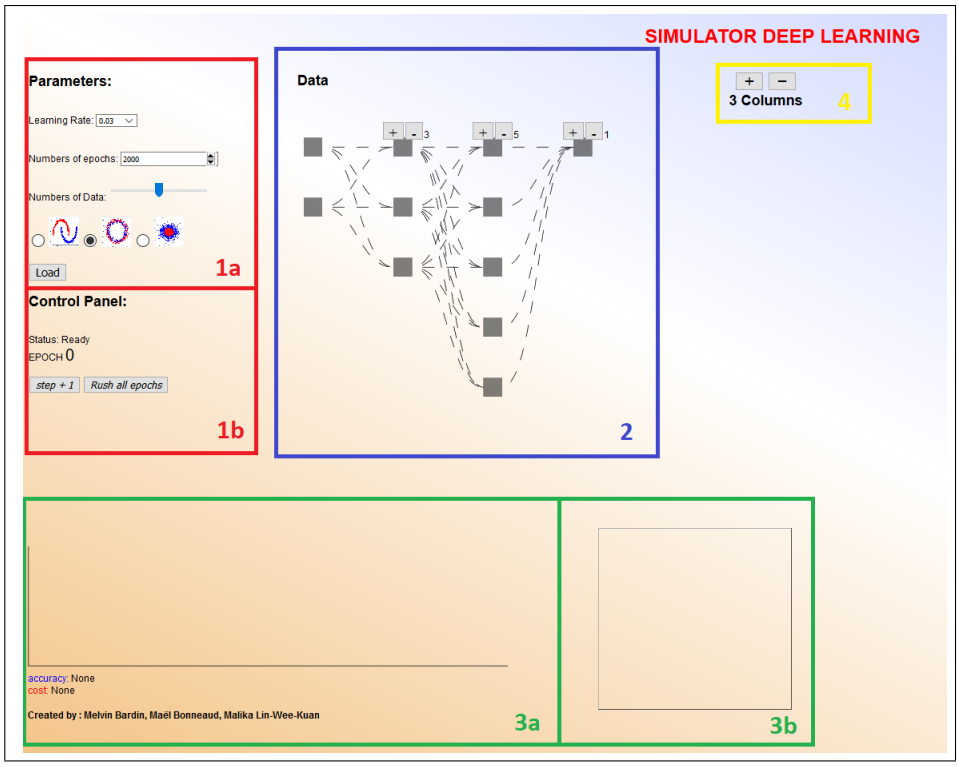

FIGURE 2 – L'interface de l'application

Notre site Web est composé d'une seule page. Celle ci est composée d'une zone d'édition (c.f. Figure 2.2) permettant de configurer le réseau de neurone que l'on souhaite.
On peut ajouter des neurones dans les *layers* avec les boutons "+" et "-", sachant qu'il faut que le dernier layer soit à 1. Il est aussi possible de rajouter des *layers* avec les boutons "+" et "-" (c.f. Figure 2.4).
Nous avons aussi un cadre permettant de renseigner les paramètres de calcul (c.f. Figure 2.1a). On peut y choisir son jeu de données , le nombre de données , le taux d'apprentissage ainsi que le nombre d'époques. Tous ces paramètres sont très importants pour faire fonctionner le réseau de neurones. Le bouton *"Load"* permet de charger tout ces paramètres une fois qu'ils ont été choisis.
En dessous, nous avons également un panneau de contrôle (c.f. Figure 2.1b) pour visualiser l'avancement des époques et de lancer l'animation.
Dans la partie inférieure, deux courbes se dessinent (c.f. Figure 2.3a) permettant de visualiser la précision des calculs ainsi que le coût durant l'évolution des *Epochs*. Nous obtenons aussi le résultat à la fin sous forme d'image des calculs effectués (c.f. Figure 2.3b).On pourra ainsi visualiser la frontière de décision.

5.1 Courbes

6 Structures de Données

Les principales structures de données sont du coté serveur des dictionnaires et coté Web, des Json. En effet lorsque le serveur effectue les calculs, il garde les valeurs comme trace dans trois dictionnaires pour l'envoyer sous forme de Json au site web. Ce dernier va les extraire pour effectuer le rendu graphique. Ces données pouvant être très volumineuse, dépassant le méga-Octet dans certains cas, les générer et les transférer sont les premières causes de ralentissement observé.

7 Analyse des Résultats

7.1 Performance et efficacité du logiciel

Les temps de calculs côté serveur pouvant être assez long, nous avons utilisé un formulaire asynchrone avec Ajax afin d'éviter d'avoir la page figée le temps de la requête et d'avoir le réseau de neurones qui se relance à sa version par défaut.

7.2 Analyse du réseau de Neurones

Nous pouvons remarquer que des couches semblent redondantes car les valeurs exprimées sur les différents arcs se répètent de manière symétriques et donc ces couches n'apportent pas vraiment de modification. Elles sont mises en évidence dans la figure 4. Cette information peut aider le concepteur du modèle à se poser des questions sur le fait qu'il y ait trop de *layers*.

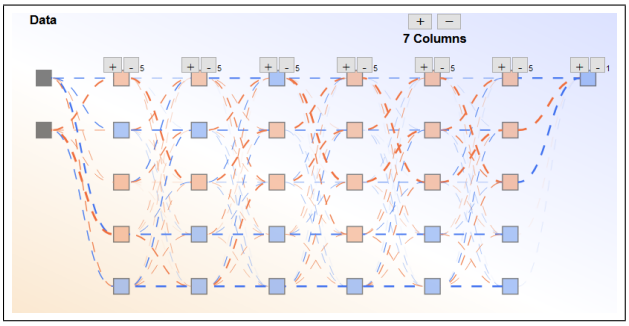

FIGURE 3 – Réseau avec des layers redondantes

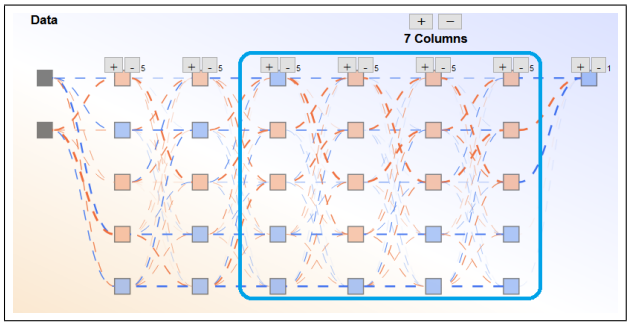

FIGURE 4 – Mise en évidence de layer redondantes

Si nous diminuons le nombre de *layers* comme l'illustre la figure 5, nous constatons que *l'accuracy* de la figure 6 n'a pas bougé et que justement*l'accuracy* reste tout à fait correcte par rapport au réseau avec 7 *layers*.

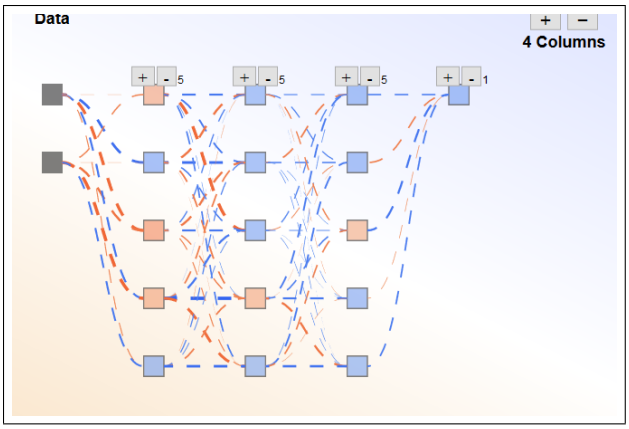

FIGURE 5 – Réseau avec layers non redondants

FIGURE 6 – Résultat avec layers non redondants

Concernant le nombre de neurones par couche, il est moins évident de constater un motif répétitif mais il semblerait que certains neurones soit moins actives voir même pas activées (figure 8 cadre bleu), ils n'influenceraient pas la solution et peuvent être enlevés.

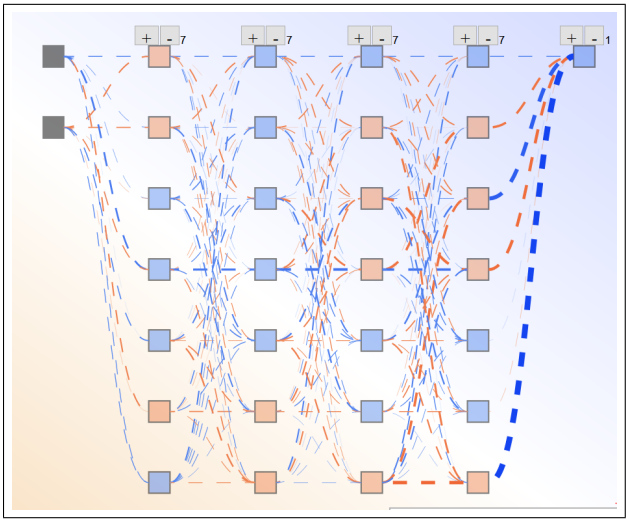

FIGURE 7 – Réseau avec des neurones inactifs

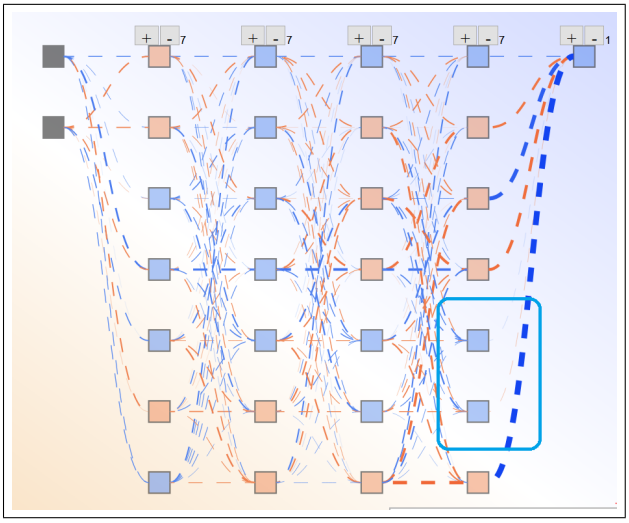

FIGURE 8 – Mise en évidence de neurones inactifs

Comme nous l'avons vu précédemment, le fait de pouvoir analyser les évolutions permet de mieux comprendre comment les résultats finaux sont obtenus. Cependant, cette application permet également d'avoir des informations importantes sur la composition même du réseau.

Troisième partie

Gestion du Projet

8 Le diagramme de Gantt

En résumé, un diagramme de Gantt répertorie toutes les tâches à accomplir pour mener le projet à bien, et indique la date à laquelle ces tâches doivent être effectuées (le planning). La colonne de gauche du diagramme énumère toutes les tâches à effectuer, tandis que la ligne d'en-tête représente les unités de temps les plus adaptées au projet (jours, semaines, mois etc). Chaque tâche est matérialisée par une barre horizontale, dont la position et la longueur représentent la date de début, la durée et la date de fin.
Voici notre diagramme de Gantt du projet :

	Task	Assigned To	Start	End	Dur
	Projet	Maël, Malika, Melvin	17/1/20	10/5/20	81
1	Etude du Deep Learning	Tous	17/1/20	5/2/20	14
2	Etude des fonctions python fournie *(MyNeural Network)*	Tous	17/1/20	5/2/20	14
3	Conception du site Web		23/1/20	1/5/20	72
3.1	Création du Tamplate du Site	Tous	23/1/20	5/2/20	10
3.2	Mise en place des courbes du réseau de neurones	Melvin	5/2/20	28/2/20	18
3.3	Mise en place des couches de neurones	Maël	5/2/20	28/2/20	18
3.4	Mise en place des neurones	Malika	5/2/20	28/2/20	18
3.5	Ajout du formulaire pour les paramètres	Tous	5/2/20	28/2/20	18
3.6	Analyse et compréhension des donnée d'un Json	Melvin, Malika	28/2/20	12/3/20	10
3.7	Animation du réseau de neurones en fonction des données	Tous	12/3/20	26/3/20	11
3.8	Colorisation des courbes	Malika	14/3/20	26/3/20	9
3.9	Les paramètres affectent les calculs du Programme python	Melvin	26/3/20	7/4/20	9
3.10	Mise en forme du site	Mael	26/3/20	20/4/20	18
3.11	Passage du formulaire en Assynchrone (Ajax)	Melvin	7/4/20	20/4/20	10
3.12	Récupération de l'état final des calculs sous forme d'image	Melvin, Malika	21/4/20	28/4/20	6
3.13	Changement de la méthode de colorisation des courbes	Malika	20/4/20	1/5/20	10
3.14	Finalisation et dernières retouches	Tous	25/4/20	30/4/20	4
4	Conception du Serveur		28/2/20	21/4/20	38
4.1	Déploiement du serveur Python (Flask)	Maël	28/2/20	12/3/20	10
4.2	Communication entre le formulaire et le serveur	Melvin	26/3/20	7/4/20	9
4.3	Ajout d'un nouveau jeu de donnée	Malika	8/4/20	21/4/20	10
5	Rédaction du rapport	Tous	20/3/20	10/5/20	36

FIGURE 9 – Diagramme de Gantt du Projet

9 Trello

Trello est une application web permettant d'organiser l'avancement d'un projet très simplement. En effet Trello permet de gérer les différents étapes de développement , ou d'avancement du projet sous formes de tableaux , de cartes etc. L'avancement du projet devient donc beaucoup plus clair et simple, il est facile pour les développeurs de savoir ce qu'il y a à faire et ce qui a était fais par les autres.

9.1 Exemple d'échanges lors du projet

mael bonneaud <mael190399@hotmail.fr>
À pascal.poncelet@lirmm.fr, melvin.bardin@etu.umontpellier.fr, malika.lin-wee-kuan@etu.umontpellier.fr

Compte Rendu de la réunion du 28 Février.
Lieu: département informatique.
Présent: Melvin Bardin, Maël Bonneaud, Malika Lin-Wee-Kuan.

Point abordés:
- Réponses aux questions sur le serveur python
- Explication de la communication entre les programmes python et le site web HTML
- Explication des ajouts à apportés et de comment les mettre en place (variables renvoyé par les programmes python à utiliser)

Décisions:
Construction du site Web avec :
- Backward Propagation
- couleur des courbes de Béziers
- Mise en place de la communication des programmes python avec le site Web
- Améliorer les fonctionnalités du site (ajout des paramètres , ajout des jeux de données ...)

Documents/code échangés:
Fichier historique envoyé par mail après la réunion.

Date et heure prochaine réunion : Jeudi 12 Mars 2020 à 11h30
Lieu de la prochaine réunion : département informatique.

Cordialement,
Bonneaud Maël.

FIGURE 10 – Compte rendu durant le projet

Quatrième partie
Conclusions

10 Problèmes rencontrés / Amélioration possibles

Durant la réalisation de notre application web nous avons rencontrés diverses difficultés.En effet nous avons du choisir entre plusieurs technologie possible lors de la conception du site. Nous devions faire des choix et ce n'était pas toujours évident de savoir quelle technologie sera meilleure. Par exemple pour faire notre serveur nous avons hésité entre plusieurs technologie comme FileZilla ou Wamp. Mais notre choix s'est finalement porté sur Flask puisqu'il répondait d'avantage à nos besoins.Actuellement nos avons également un soucis au niveau de SVG. En effet

11 Bilan

Durant ce projet de programmation nous avons atteint les objectifs donnés par M. Poncelet, en réalisant les différentes tâches du cahier des charges.

Malgré les problèmes et les difficultés que nous avons rencontrés (et vu précédemment), nous sommes satisfait de notre travail.

Dans le but d'une évolution,

Pour réaliser ce projet nous avons utilisé divers moyens de communications et de partage de documents, entre autre "Discord" , Gantt, Trello et le dépot "Git".

Pour conclure sur les réseaux de neurones, Le *Deep Learning* est utilisé dans de nombreux domaines :

Robotique Ce qui a permis aux robots de détecter et de réagir à leur environnement.

Agriculture Permet de déployer des équipements capables de repérer et de différencier les plantes cultivées et les mauvaises herbes.

Étude d'images Ce qui ramène à différents points :

- Classification d'image/donnée (par exemple : savoir si la photo représente un chat)
- Reconnaissance faciale en Asie.
- Conduire une voiture
- **Imagerie Médicale**

  Permet la reconnaissance de cancer sur une radio, et permet d'assister dans les décisions médicales

  Dans la situation actuelle, à l'aide d'une photo permet de savoir si une personne à de la fièvre ou non

12 Bibliographie

Références

[1] The SciPy COMMUNITY. *numpy.ndarray.* URL : `https://www.miximum.fr/blog/introduction-au-deep-learning-2/` (visité le 07/05/2020).

[2] Thibault JOUANNIC. *Deep learning : la rétropropagation du gradient.* URL : `https://docs.scipy.org/doc/numpy/reference/generated/numpy.ndarray.html` (visité le 07/05/2020).

[3] Pascal PONCELET. *My Neural Network (Jupyter NoteBook).* (Visité le 08/05/2020).

[4] Florent SIMON. *Deep Learning, les fonctions d'activations.* URL : `https://www.supinfo.com/articles/single/7923-deep-learning-fonctions-activation` (visité le 08/05/2020).

[5] Daniel SMILKOV et Shan CARTER. *Simulateur Deep Learning.* URL : `https://playground.tensorflow.org/` (visité le 07/05/2020).

Table des figures

13 Annexes

13.1 Fonctionnement du site

13.1.1 Prérequis pour l'execution du projet

Python 3.6 ou plus récent (projet testé uniquement sur python 3.6, 3.7 et 3.8).

Les librairies suivantes sont nécessaires :

- Flask ;
- numpy ;
- sklearn ;
- matplotlib ;
- panda ;
- seaborn ;
- sys ;
- json ;
- math ;

Installation des librairies via le terminal via la commande **pip install** ou **pip3 install**.

13.1.2 Execution du projet

Pour lancer le serveur : **py view2.py** ou **python view2.py** ou **python3 view2.py**, dans un terminal.

Ouvrez un navigateur et allez a l'adresse " http://127.0.0.1:5000/index.html/ "

Le site a été testé sur les navigateurs :
— Chrome
— Mozilla Firefox.

Le site a été testé sur les systèmes d'exploitation :
— Linux (Xubuntu)
— Window 10
— Mac (mojave 10.14)

13.1.3 Utilisation

Configurer votre réseau comme bon vous semble (maximum 9 layer). ATTENTION : la dernière layer doit posséder un unique neurone !

Choisissez vos préférence de calcul dans les paramètres.

Appuyer sur *"Load"* lorsque vous êtes satisfait.

Lorsque ceci est fait, dans le panneau de contrôle *'status :ready'* passe en *'Loading...'* indiquant que le serveur travaille. ATTENTION : évitez de *"Load"* lorsque le statut est *"Loading..."*, le serveur pourrait planter.

Lorsque les calculs sont finis et que les données on été retourné au site, *'status : Loading...'* redevient *'status : Ready !'*.

Vous pouvez maintenant décidé d'avancer étape par étape en appuyant sur *"step + 1"* ou lancer l'animation avec *"rush all epochs"*.

Le réseau de neurone s'anime, les courbes de précision et de coût se dessine et lorsque ceci est finis, le résultat via une image s'affiche.