Rapport TER - HLIN601

Visualisation de la boîte noire des réseaux de neurones

Auteurs :
Josua PHILIPPOT
Aurélien BESNIER
Hayaat HEBIRET
Quentin YECHE

Encadrant :
Pascal Poncelet

Université de Montpellier
Année 2019-2020

Table des matières

Table des figures

1 Introduction

Dans un contexte où des données de plus en plus volumineuses et de plus en plus nombreuses font leurs apparitions chaque jour sur Internet, les réseaux de neurones et autres techniques d'exploitation big data ont de plus en plus de succès : on a besoin de répondre a une demande croissante pour traiter une grande quantité de données de façon efficace. Dans ce contexte, le traitement de grosses données telles que des images haute résolution s'est vu utiliser les réseaux de neurones pour reconnaître des signatures caractéristiques de ce qu'elles représentent. Cependant il arrive que les réseaux de neurones se trompent lors de l'analyse des programmes donnent des résultats inattendus.

Prenons un exemple pour concrétiser ce type de cas :

Nous avons 3 jeux de données qui respectent un même format, la première est issue d'études sur des chats, la deuxième sur des chiens et la troisième sur des voitures.

Nous traitons les deux premières données afin d'obtenir un modèle qui permet de définir si un objet est une chien ou un chat. Suite à cela nous introduisons le troisième jeu de données et nous affichons les résultats obtenues. Deux possibilités s'offre à nous :
→ Le modèle conclut que la voiture est un chien.
→ Le modèle conclut que la voiture est un chat.

Or les deux cas sont absurdes car, par exemple , les proportions d'une voitures ne s'apparente à aucun des deux.

Ainsi on s'intéresse dans ce TER au fonctionnement du réseau de neurones afin de comprendre comment le modèle en est arrivé à cette conclusion lors du traitement de données. On aimerais notamment connaître la démarche des neurones et déterminer à l'avance le résultat escompté.

Une vidéo de démonstration est disponible à : <insèrelien>

Le rapport est organisé de la manière suivante. Dans la section 2 nous commenceront par des définitions.... Dans la section 3, nous listerons les différents logiciels et bibliothèques utilisées au cours de ce projet. Dans la section 4 nous expliquerons les méthodes l'implémentassions des données et des algorithmes . Dans la section 5 nous analyserons les résultats obtenus avec notre application. Dans la section 6 on nous parlerons de la manière dont on a abordé le sujet et les différents outils utilisés. Enfin dans la section 7 nous récapitulerons le travail effectué.

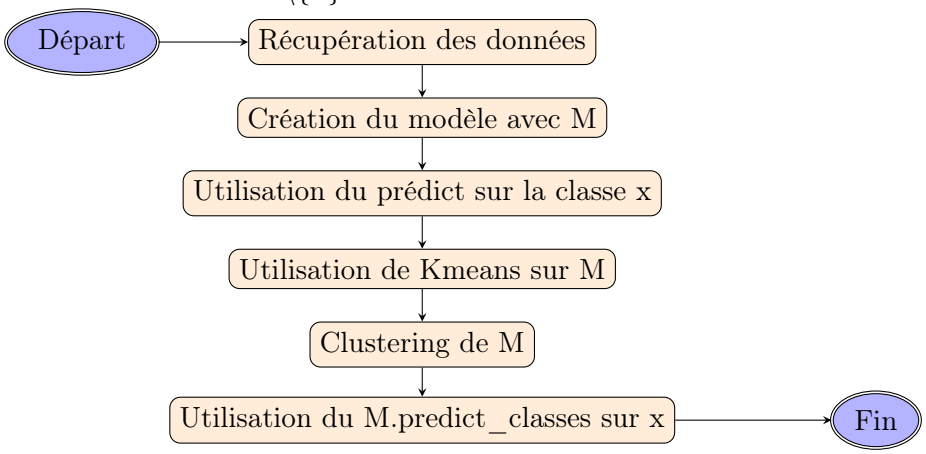
2 Réseaux de neurones et définitions préliminaires

(foward propagation : chat chien pour transmettre l'information , kmeans clustering permet d'optimiser selon le nombre de cluster + glossaire qui détail)

ORGANIGRAMME Les jeux de données utilisés sont dans un nombre satisfaisant, évitant ainsi un modèle qui donne un résultat différent à chaque fois qu'elle est construite.

Soit E un ensemble de classes c tel que E = c1,c2...cn avec n $\in \mathbb{R}$.

Soit x $\subset$ E et M = E $\setminus\{x\}$



pseudo organigramme : pour chaque layer : création d'un modèle en utilisant predict_classes, ainsi qu'un predict sur la classe à introduire. détermination d'un clustering avec kmeans et l'appliquer sur les 3 classes traitement des données.

pour chaque classe : déterminer son cheminement par cluster à travers les layers.

2.1 Définitions

— KMEANS :

—

—

3 Technologies utilisées

— Langage de programmation : Python3 [1]

— Application Jupyter notebook [2]

— Bibliothèque Python destinée à l'apprentissage automatique scikit-learn [3]

— Bibliothèque Python pour la manipulation et l'analyse des données pandas [4]

— L'outil open source d'apprentissage automatique tensorflow [5]

— Bibliothèque logicielle libre et open source Python permettant la manipulation de matrice et de fonctions mathématiques numpy [6]

— Bibliothèque permettant d'interagir avec les algorithmes de réseaux de neurones profonds et de machine learning keras [7]

— Bibliothèque graphique JavaScript qui permet l'affichage de données numériques sous une forme graphique et dynamique D3.js [8]

1. [dernière consultation le 4/5/2020] / version 3.6.9
2. [dernière consultation le 4/5/2020] / version 1.0.0
3. [dernière consultation le 4/5/2020] / version 0.22.1
4. [dernière consultation le 4/5/2020] / version 1.0.1
5. [dernière consultation le 4/5/2020] / version 1.12.0
6. [dernière consultation le 4/5/2020] / version 1.18.1
7. [dernière consultation le 4/5/2020] / version 2.2.4
8. [dernière consultation le 4/5/2020] / version 4.x.x

4 Développements Logiciel : Conception, Modélisation, Implémentation

4.1 Structure des données

Les données son stocké sous la forme d'une dataframe (fichier csv) et leur lecture se fait avec la bibliothèque pandas.

4.2 Traitement des données

Les données sont traitées dans un modèle séquentiel de keras . En premier, on met en place notre modèle en apprenant sur certaines classes (fonction *fit()*) puis on va faire une prédiction sur d'autres données non apprises (fonction *predict_ classes()*).
On obtient alors une dataframe pour la classe à introduire, une autre pour le modèle. Chacune ayant des neurones et leur valeur d'activations par layers.
On divise ensuite les layers dans plusieurs dataframe.

4.2.1 Clustering

A partir de cette étape le traitement se fait séparément selon le layer. Pour effectuer le clustering des classes on utilisera l'algorithme de clustering KMEANS [9] de la bibliothèque scikit-learn.
On définit donc le nombre de clusters et la fonction les positionne de façons optimale.

On obtient ainsi un résultat similaire à celui-ci :

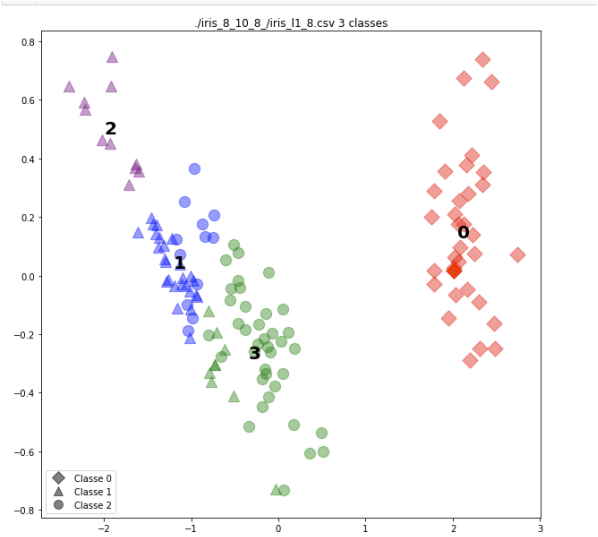


FIGURE 1 – Répartitions des classes dans les clusters

9. définition page 4

Le but de cette clusterisation est de réunir les objets ayants des signatures similaires et ainsi pouvoir déterminer le cheminement des classe layer par layer.

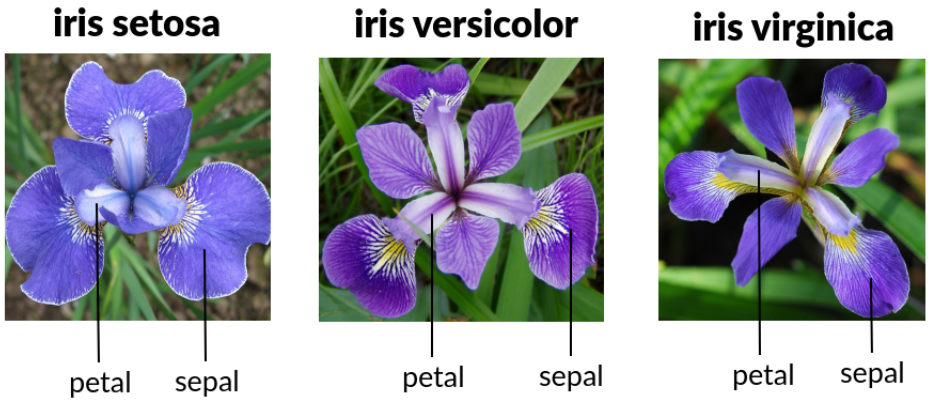
Avant de pouvoir analyser les résultats on enregistre les données sous leur format final : un fichier JSON. Ce fichier contient les informations essentielles au tracé de leur parcours dans le réseau de neurones, c'est à dire :

— Le nom des nodes (clusters et inputs).
— La source et la cible de chacune des nodes.
— La classe de ses dernières.
— Et le nombre d'éléments regroupés par classes (les links) qui les traverses.

Ce fichier est ensuite passé à notre application web qui va les lire et produire les schémas que nous étudirons dans la section suivante.

5 Analyse des Résultats

5.1 IRIS



"IRIS Data Set" est une dataset contenant 3 espèces d'Iris (Iris Setosa,Iris Versicolor,Iris Virginica). Chacune des espèces possède 50 exemples.

La structure du dataset est la suivante :

sepal length	sepal width	petal length	petal width	class
				Iris Setosa
...				
				Iris Versicolor
...				
				Iris Virginica
...				

Modèle crée à partir de : Iris Virginica, Iris Setosa .

Classe introduite : Iris Versicolor

5.1.1 DÉMARCHE 1

Nombre de layer : 3
Nombre de clusters : 4

Les résultats obtenues :

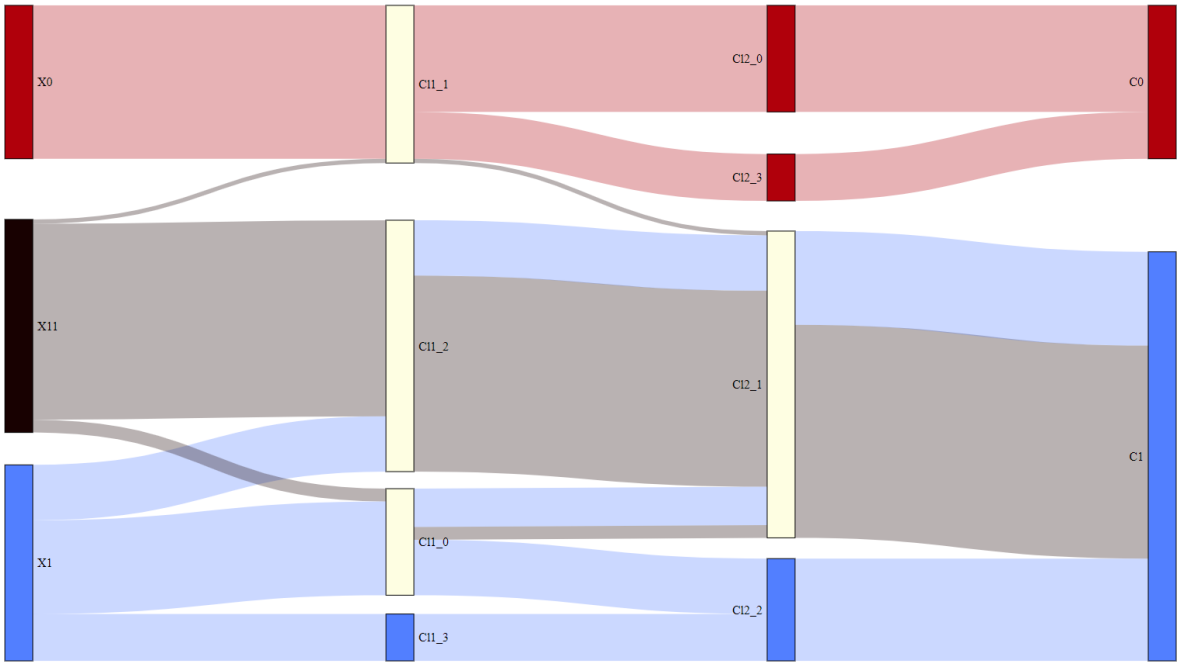


FIGURE 2 – Iris Virginica, Iris Setosa et Iris Versicolor (les combinaisons de clusters partagés sont en blanc)

Remarque(s) : On voit que Iris-Versicolor suit un trajet similaire à Iris-Virginica ce qui semble cohérent avec les images ci-dessus.

5.1.2 DÉMARCHE 2

Nombre de layer : 10
Nombre de clusters : 4
Les résultats obtenues :

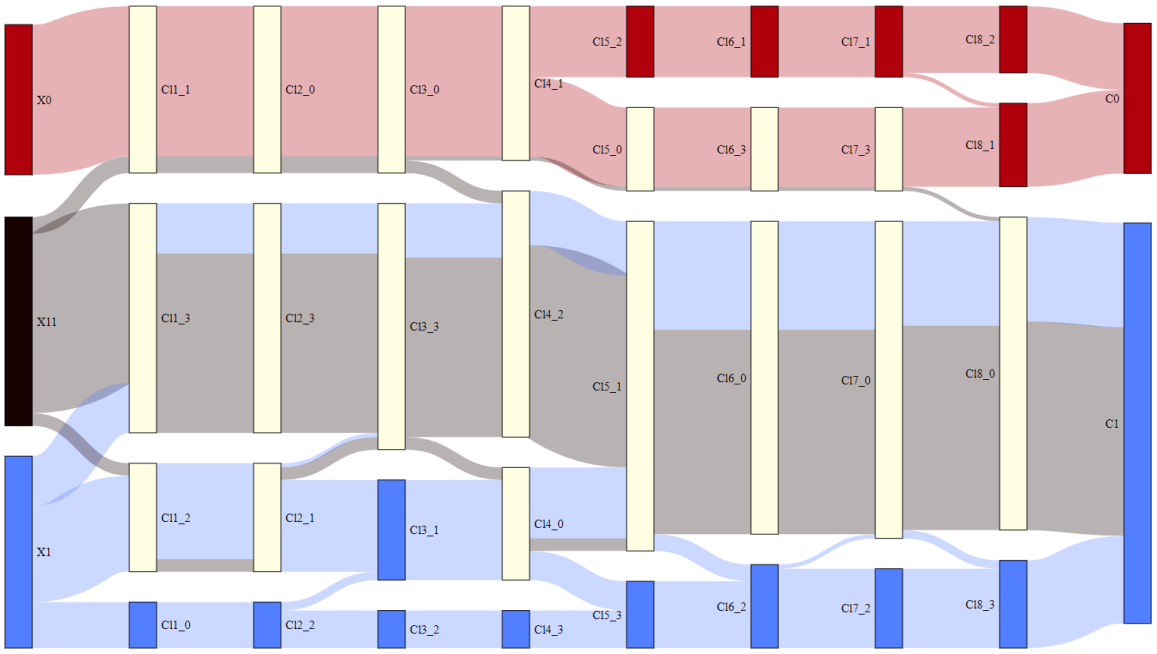
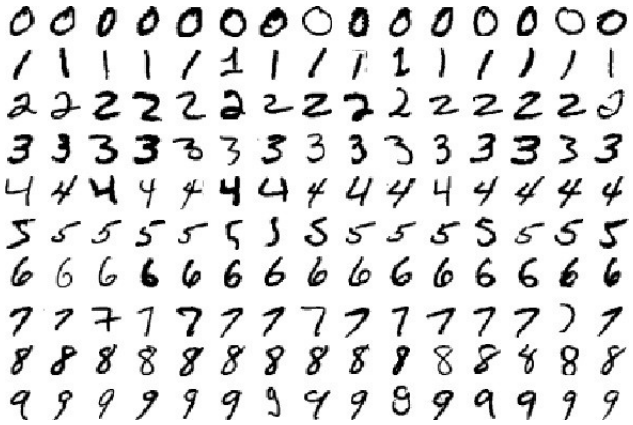


FIGURE 3 – Iris Virginica, Iris Setosa et Iris Versicolor

Remarque(s) : On en revient à la même conclusion.

Conclusion final sur cette donnée : A posteriori ajouter des layers n'apporte pas plus d'informations sur la trajectoire de la classe introduit.

5.2 MNIST



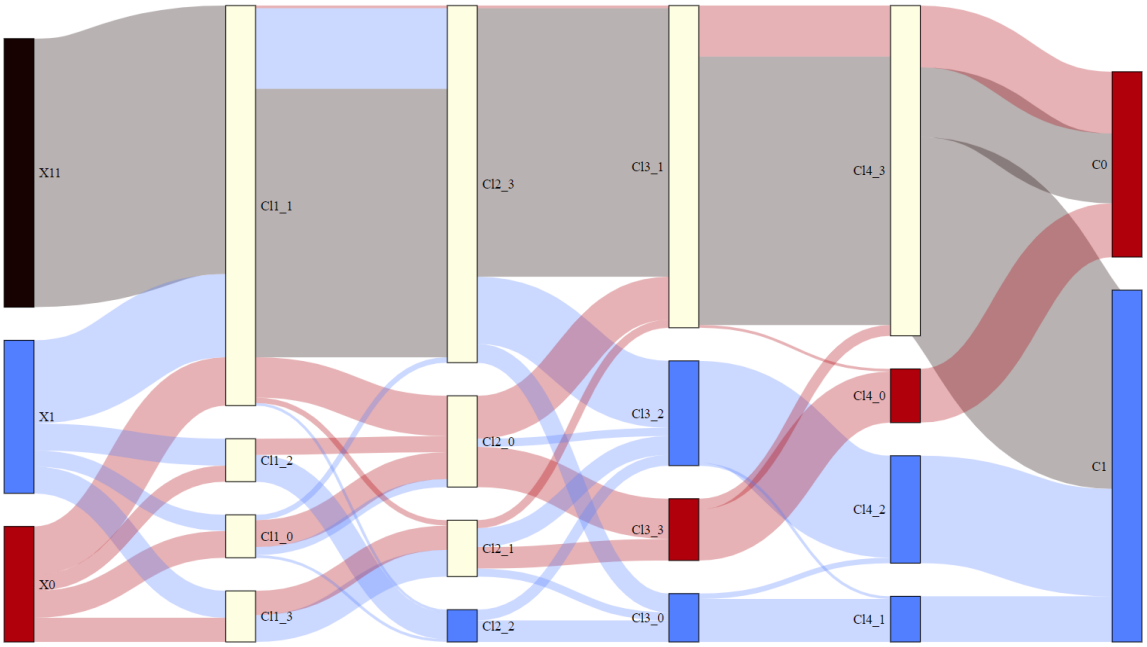
MNIST database est un database contenant une large collection de chiffres écrits à la main. Une partie proviens notamment de deux databases de MNIST

Les écrits sont ensuite normalisés, numérisés et représentés dans un tableau de dimension 28x28.(Les donnés de chaque cases représentant des niveau de gris)
MNIST à une collection de 60 000 chiffres et de 10 000 autres (moins bien lisible) qui sont utilisés pour déterminer les chiffres qu'ils représentent.

Dans notre cas nous nous intéressons qu'aux chiffres bien formés.

5.2.1 DÉMARCHE 1

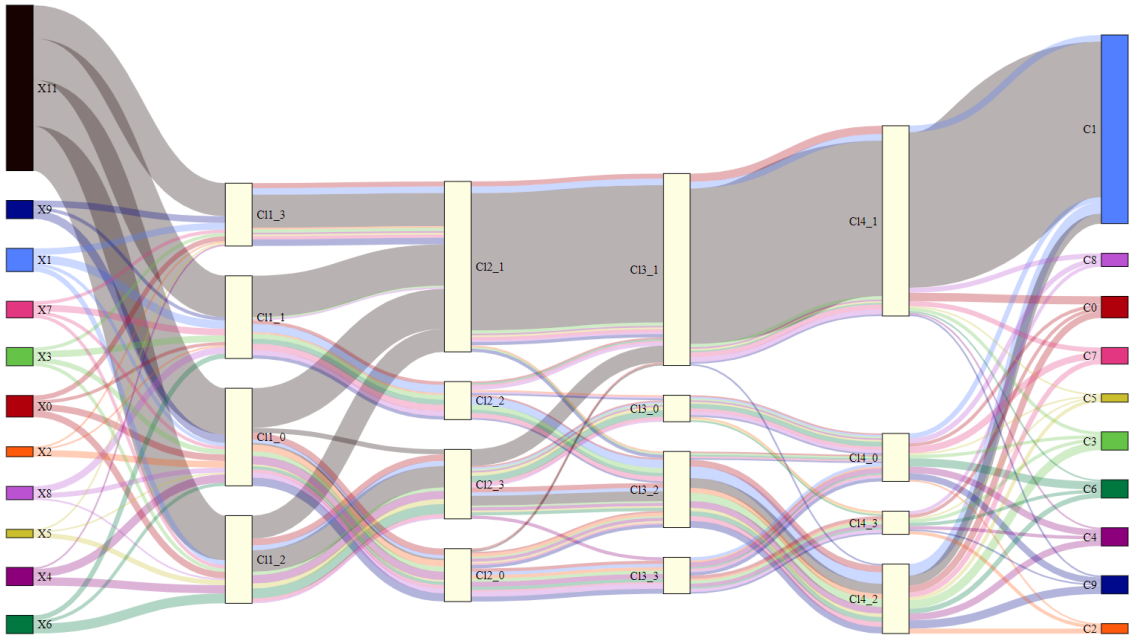
Modèle crée à partir : des chiffres 0 et 1.
Classe introduite : le chiffre 2
Nombre de layer : 5
Nombre de clusters : 4
Les résultats obtenues :



Remarque(s) :
La classe introduite va suivre des chemins variés et à la lecture seule de ce graphe, on pourra pas définir de manière certaine s'elle appartient à la classe 1 ou 0.

5.2.2 DÉMARCHE 2

Modèle crée à partir : des chiffres 0 à 9
Classe introduite : jeu de test avec des chiffres "mal formés".
Nombre de layer : 5
Nombre de clusters : 10
Les résultats obtenues :



Remarque(s) :
On observe un regroupement final de la classe de test (les chiffres mal formés) vers le cluster de classe C1 (le chiffre 1) on pourrait donc e déduire par la simple vue du graphe que la classe de test ne contient que des 1, or ce n'est pas le cas en réalité.

Conclusion final sur cette donnée : A posteriori les résultats ne sont pas aussi flagrants que sur iris. Cela est sûrement du a la nature du jeu de données. Nous pouvons malgré tout suivre les chemins empruntés par la classe

6 Gestion du Projet

6.1 Diagrammes de Gantt

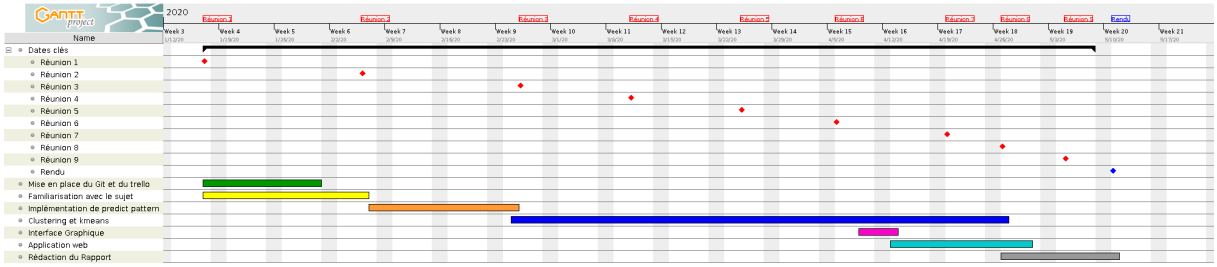


FIGURE 4 – Diagramme des tâches

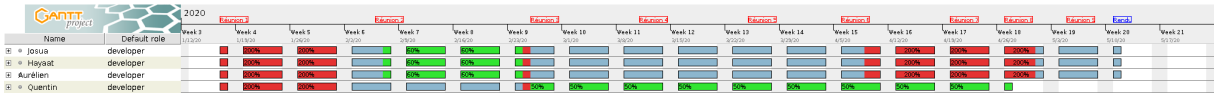


FIGURE 5 – Diagramme des ressources

6.2 Outils utilisés

Nous avons mis en place un trello bien que nous l'avons mis de côté pendant une grande partie du projet. De par la nature du sujet, nous n'avons pas vraiment su nous diviser le travail, donc nous avons préféré travailler tous ensemble.

Nous avons aussi organisé des réunions de groupe sur Discord qui nous ont permit de rester en contact même lors du confinement, bien que l'on a eu quelques soucis de communication dans cette période malgré tout.

Nous avons aussi un Github accessible ici : `https://github.com/JPhilippot/Patterns` sur lequel est disponible ce rapport, le notebook produit ainsi que l'application web de visualisation des clusters par layers.

7 Conclusion

Ce projet nous a permis de mieux comprendre le domaine des réseaux de neurones. Cependant, malgré la possibilité de voir le cheminement des donnés, on ne peut toujours pas prédire à l'avance la réponse du modèle lors de l'introduction d'une classe inconnue au modèle.
L'application a donc une réelle utilité quand il est accompagné d'autres outils d'analyses.

D'autre part d'un point de vue personnel il a été difficile pour nous d'appréhender le sujet au départ car il ne ressemblait a rien de ce que l'on avait vu auparavant. Malgré tout le sujet reste une expérience intéressante et nous a donné des connaissances qui pourrait potentiellement nous servir dans le futur.

Le code produit est disponible sur Github ici : https://github.com/JPhilippot/Patterns et la vidéo de présentation ici : <>.

Pour terminer, nous tenions à remercier chaleureusement Pascal Poncelet pour avoir proposé ce sujet et nous avoir guidé tout au long du projet en faisant preuve de patience et d'une réelle implication dans ce dernier.

8 Bibliographie

Références

https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference
https://scikit-learn.org/stable/auto_examples/datasets/plot_iris_dataset.html
https://scikit-learn.org/stable/index.html
https://mohitatgithub.github.io/2018-03-28-MNIST-Image-Classification-with-CNN-&-Keras/
https://bl.ocks.org/d3noob/06e72deea99e7b4859841f305f63ba85
Iris Data Set https://archive.ics.uci.edu/ml/datasets/iris (5/5/20 21h)
https://en.wikipedia.org/wiki/MNIST_database (5/5/20 22h)
http://yann.lecun.com/exdb/mnist/ (5/5/20 22h)