Rapport TER - HLIN601

Réseau de neurones : que se cache-t-il dans la boîte noire ?

Auteurs :
Josua PHILIPPOT
Aurélien BESNIER
Hayaat HEBIRET
Quentin YECHE

Encadrant :
Pascal Poncelet

Université de Montpellier
Année 2019-2020

Remerciements

Nous tenons 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.

Table des matières

Table des figures

1 Introduction

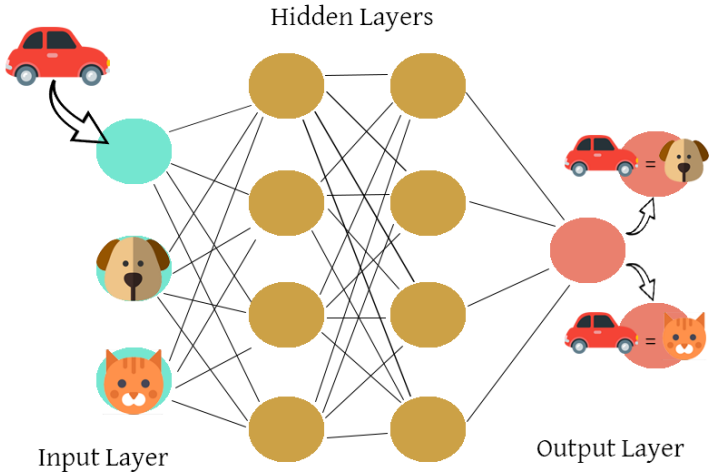
Dans un contexte où des données de plus en plus volumineuses font leurs apparitions chaque jour sur Internet, les réseaux de neurones et autres techniques d'exploitation big data ont de plus en plus de succès : on a besoin de répondre à une demande croissante pour traiter une grande quantité de données de façon efficace.

Dans ce contexte, les réseaux de neurones ont montré qu'ils étaient particulièrement efficaces pour, par exemple, traiter un grand volume d'images.

Depuis ses premières implémentassions à la fin du XXe siècle, les réseaux neuronaux n'ont pas cessés d'évoluer et d'attiser la curiosité des gens. On peut lui attribuer de multiples succès qui ont contribué à la renommée de ce procédé, notamment avec la création de ImageNet en 2009 qui permet la classification d'images dans plus de 20 000 catégories existantes, ou des exploits plus insolites comme en 2011 où une IA à battue pour la première fois un être humain au jeu télévisé Jeopardy, ou encore en 2016 où AlphaGo battit pour la première fois un professionnel sans handicap au jeu de Go.

Cependant, il arrive que les réseaux de neurones se trompent lors de l'analyse des programmes et donnent des résultats inattendus.

Prenons un exemple pour concrétiser ce type de cas :



Nous avons trois 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 un chien ou un chat. Suite à cela nous introduisions le troisième jeu de données et nous affichons les résultats obtenue.

Deux possibilités s'offrent 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 voiture s'apparentent 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 aimerait notamment savoir comment fonctionne le réseau et si on peut extraire des signatures caractéristiques de classes.

Une vidéo de démonstration est disponible à : https://youtu.be/-cfFWJnAjCo

Le rapport est organisé de la manière suivante. Dans la section 2 nous commencerons par des définitions. Dans la section 3, nous listerons les différents logiciels et bibliothèques utilisés au cours de ce projet. Dans la section 4 nous expliquerons les méthodes de traitement des données et les algorithmes utilisés. Dans la section 5 nous analyserons les résultats obtenus avec notre application. Dans la section 6 où 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

2.1 Définitions

— Jeux de données (dataset) [1] : un ensemble de données (ou valeurs) dans lequel on associe une variable (ou attribut) et une observation pour chacune d'entre elles.

— Neurone [2] : Le neurone formel est conçu comme un automate doté d'une fonction de transfert qui transforme ses entrées en sortie selon des règles précises.

— Fonction d'activation [3] : Fonction mathématique appliquée à un signal en sortie d'un neurone.

— Layer (Couche) [4] : Couche du réseau composée de neurones.

— Modèle (Structure du réseau) [5] : Résultat d'une méthodes d'apprentissage de jeux de données Elle est composé d'une succession de couches dont chacune prend ses entrées sur les sorties de la précédente.

— Cluster [6] : Regroupement de données avec des valeurs d'activations similaires.

— Partitionnement de données (data clustering) [7] : Une méthode en analyse des données. Elle vise à diviser un ensemble de données en différents clusters homogènes, en ce sens que les données de chaque sous-ensemble partagent des caractéristiques communes, qui correspondent le plus souvent à des critères de proximité

— K-moyennes (K-means) [8] : Une méthode de partitionnement de données, il permet de regrouper les données qui se ressemblent en un paquet.

1. https://fr.wikipedia.org/wiki/Jeu_de_donn%C3%A9es
2. https://fr.wikipedia.org/wiki/R%C3%A9seau_de_neurones_artificiels
3. https://deeplylearning.fr/cours-theoriques-deep-learning/fonction-dactivation/
4. https://fr.wikipedia.org/wiki/R%C3%A9seau_de_neurones_artificiels
5. Structureduréseau
6. https://fr.wikipedia.org/wiki/Partitionnement_de_donn%C3%A9es
7. https://fr.wikipedia.org/wiki/Partitionnement_de_donn%C3%A9es
8. https://fr.wikipedia.org/wiki/K-moyennes

3 Technologies utilisées

— Langage de programmation Python3 [9]

— Application Jupyter notebook [10] qui permet de faire des notebook contenant des cellules de codes Python et du Markdown.

— Bibliothèque Python destinée à l'apprentissage automatique scikit-learn [11]

— Bibliothèque Python pour la manipulation et l'analyse des données pandas [12]

— L'outil open source d'apprentissage automatique tensorflow [13]

— Bibliothèque logicielle libre et open source Python permettant la manipulation de matrice et de fonctions mathématiques numpy [14]

— Bibliothèque permettant d'interagir avec les algorithmes de réseaux de neurones profonds et de machine learning keras [15]

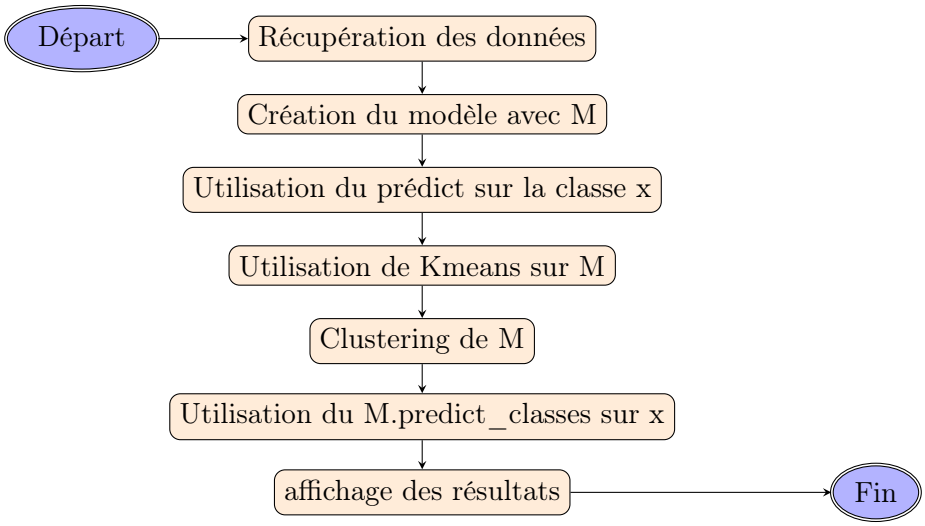
— Bibliothèque graphique JavaScript qui permet l'affichage de données numériques sous une forme graphique et dynamique D3.js [16]

9. https://www.python.org/ [dernière consultation le 4/5/2020] / version 3.6.9
10. https://jupyter.org/ [dernière consultation le 4/5/2020] / version 1.0.0
11. https://scikit-learn.org/stable/ [dernière consultation le 4/5/2020] / version 0.22.1
12. https://pandas.pydata.org/ [dernière consultation le 4/5/2020] / version 1.0.1
13. https://www.tensorflow.org/ [dernière consultation le 4/5/2020] / version 1.12.0
14. https://numpy.org/ [dernière consultation le 4/5/2020] / version 1.18.1
15. https://keras.io/ [dernière consultation le 4/5/2020] / version 2.2.4
16. https://d3js.org/ [dernière consultation le 4/5/2020] / version 4.x.x

4 Développements Logiciel : Conception, Modélisation, Implémentation

4.1 Ordre de déroulement

Soit E un ensemble de classes c tel que $E = \{c_1, c_2...c_n\}$ avec $n \in \mathbb{R}$.
Soit $x \subset E$ et $M = E \setminus \{x\}$



4.2 Structure des données

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

4.3 Traitement des données

Les données sont traitées dans un **modèle** [17] séquentiel de keras :

D'abord, 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** [18] et leur valeur d'activations par **layers** [19].
On divise ensuite les layers dans plusieurs dataframes.

4.3.1 Clustering

Á 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** [20] de la bibliothèque

17. définition page 8
18. définition page 8
19. définition page 8
20. définition page 8

scikit-learn.

On définit donc le nombre de **clusters**[21] et la fonction les positionne de façon optimale.

On obtient ainsi un résultat similaire à celui-ci :

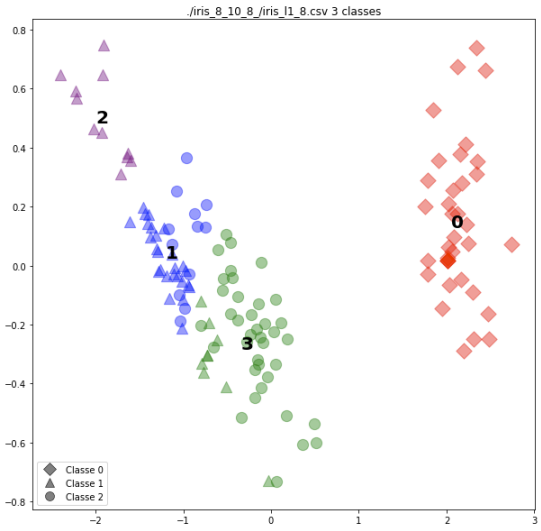


FIGURE 1 – Répartitions des classes dans les clusters

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 nœuds (clusters et inputs et output).
— La source et la cible de chacun des nœuds.
— La classe de ses dernières.
— Le nombre d'éléments regroupés par classes (les liens) qui les traverses.

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

21. definition page 8

5 Analyse des Résultats

5.1 IRIS

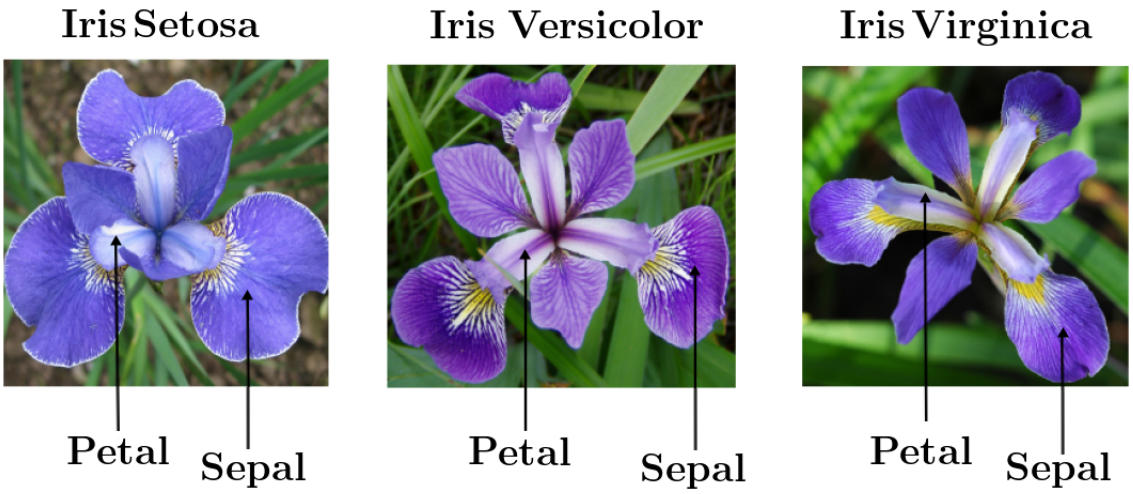


FIGURE 2 – Les trois Iris du jeu de données.

"IRIS Data Set" est un jeu de données contenant 3 espèces d'Iris (Iris Setosa,Iris Versicolor,Iris Virginica). Chacune des espèces possède 50 exemples.

La structure du jeu de données est la suivante :

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

5.1.1 Signatures pour les classes IRIS

Modèle crée à partir de : Iris Virginica, Iris Setosa et Iris Versicolor
Nombre de layer : 3
Nombre de clusters : 4
Les résultats obtenus :

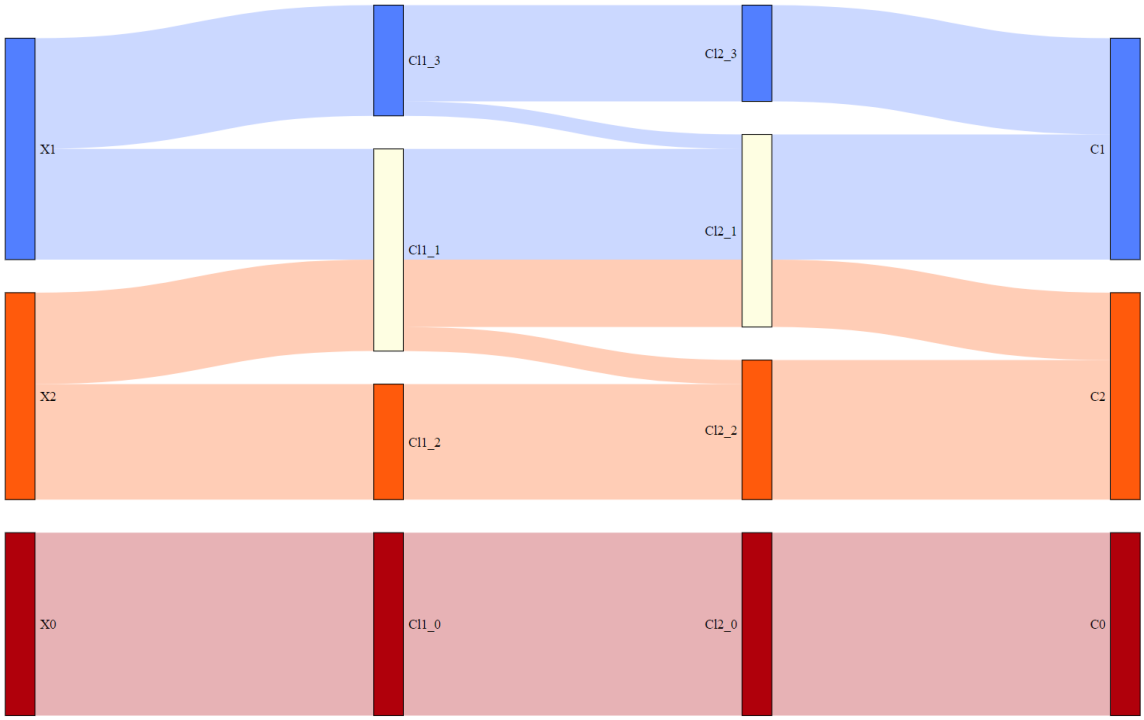


FIGURE 3 – Iris Virginica, Iris Setosa et Iris Versicolor (les combinaisons de clusters partagés sont en blanc)

Remarque(s) :
On peut voir une assez claire répartition des données, avec tout de même des clusters partagés entre Iris Virginica et Iris Versicolor. A la lecture de ce diagramme, on peut en déduire que le modèle reconnaît bien trois classes d'iris différentes.

5.1.2 Signature de Versicolor versus Virginica et Setosa

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

Classe introduite : Iris Versicolor
Nombre de layer : 3
Nombre de clusters : 4

Les résultats obtenus :

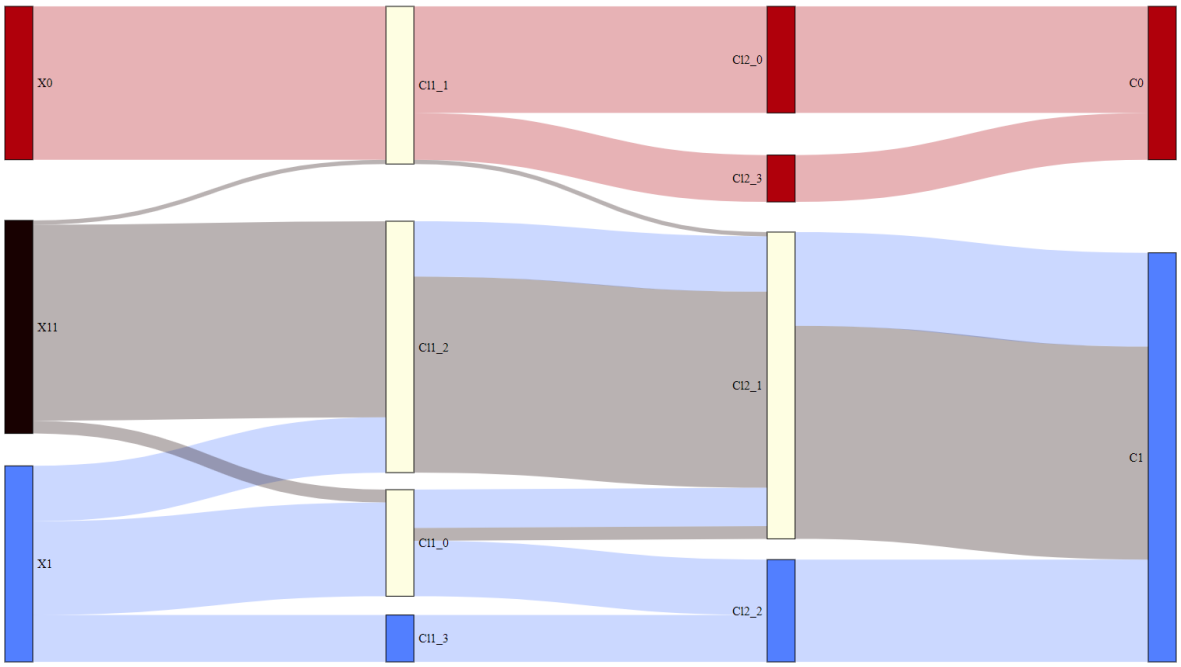


FIGURE 4 – Iris Virginica, Iris Setosa et Iris Versicolor

Remarque(s) :
On voit sur le diagramme une nette distinction entre Iris Setosa et Iris Virginica. La classe introduite, Iris Versicolor va elle être majoritairement regroupée avec Iris Virginica. On remarque deux classes en sortie, une contenant uniquement des éléments de Iris Setosa et l'autre contenant des éléments de Iris Virginica et Iris Versicolor. On peut en déduire que le modèle traite Iris Versicolor comme étant de la même classe que Iris Virginica.

5.1.3 Signature de Versicolor versus Virginica et Setosa avec de nombreux layers

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

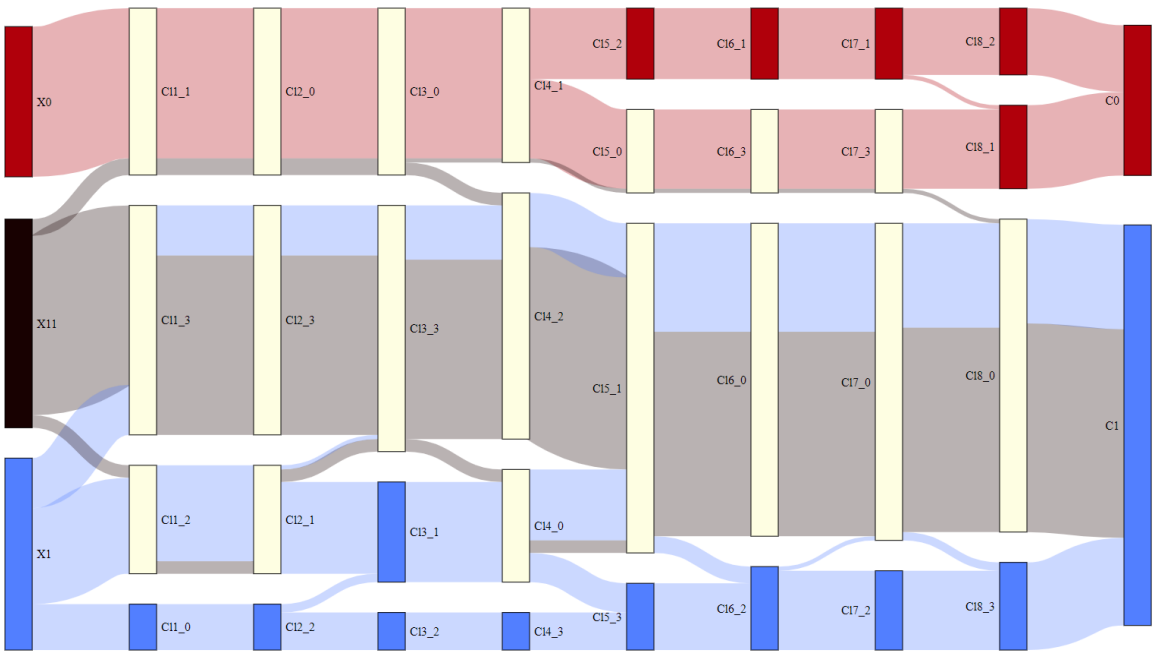


FIGURE 5 – Iris Virginica, Iris Setosa et Iris Versicolor

Remarque(s) :
On a les mêmes observations que pour le modèle précédent si ce n'est que le nombres de layers est plus grand. On a au final deux classes en sortie, une contenant uniquement des éléments de Iris Setosa et l'autre contenant des éléments de Iris Virginica et Iris Versicolor. On peut en déduire qu'encore une fois le modèle traite Iris Versicolor comme étant de la même classe que Iris Virginica, et donc que le nombre de layer ne semble pas avoir d'impact sur les prédictions finales du modèle.

Conclusion sur cette donnée : Lorsque l'on introduit une classe inconnue dans un modèle, il va essayer de la regrouper avec une classe qu'il connaît déjà. De plus, il semble qu'ajouter des layers n'apporte pas plus d'informations sur la trajectoire de la classe introduite.

5.2 MNIST

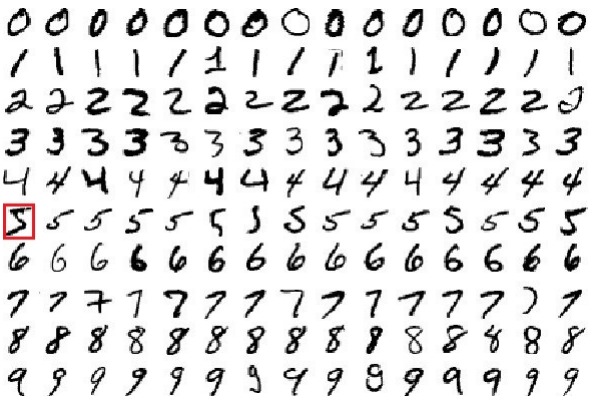


FIGURE 6 – Quelques éléments de MNIST tel qu'ils sont écrit.

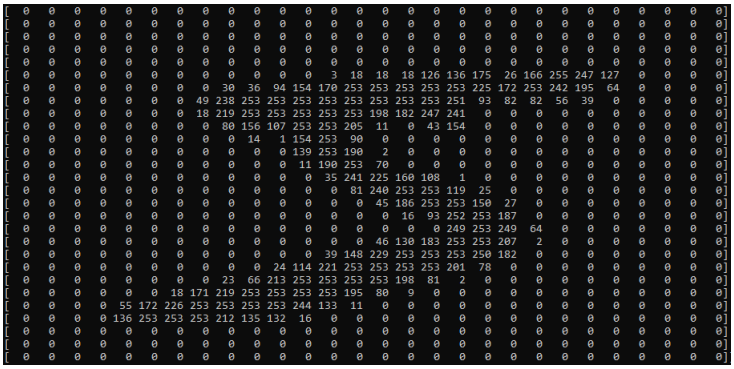


FIGURE 7 – Structure d'un chiffre 5 (encadré ci-dessus) dans notre jeu de données.

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

Les écrits sont ensuite normalisés, numérisés et représentés dans un tableau de dimension 28x28.(Les données de chaque cases représentant des niveau de gris)

MNIST à une collection de 60 000 chiffres et de 10 000 autres utilisé pour faire des jeux de tests.

5.2.1 Signature des chiffre 0, 1 , et 2

Modèle crée à partir : des chiffres 0, 1 et 2 .
Nombre de layer : 5
Nombre de clusters : 4

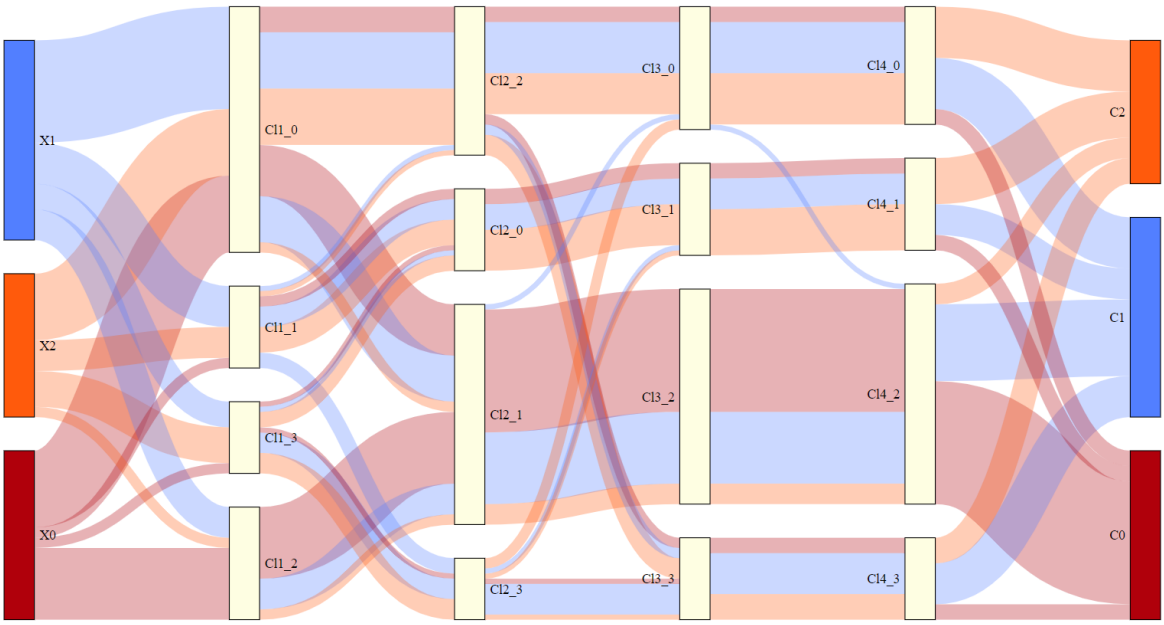


FIGURE 8 – Les chiffres 0 et 1 avec 2

Remarque(s) :
Le fait de ne mettre aucune autre classe dans le modèle nous permet d'avoir un témoin de son comportement, et de voir s'il arrive à classer les classes qu'il connaît déjà. Comme on peut l'observer sur le diagramme, bien qu'il ai tous ses clusters partagés, il a bel et bien trois classes distinctes en sortie.

5.2.2 Signature du chiffre 2 versus le modèle composé des chiffres 0 et 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 obtenus :

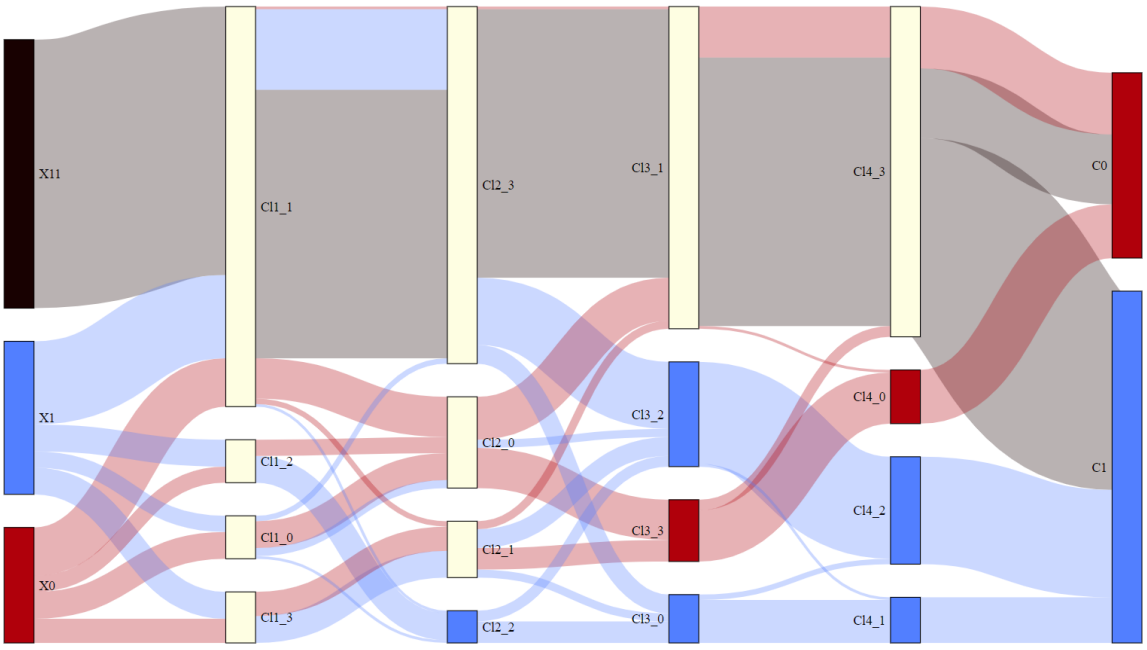


FIGURE 9 – Les chiffres 0 et 1 avec 2

Remarque(s) :
Sur le diagramme ci-dessus, on peut constater que la classe introduite va se regrouper avec les différentes classes apprises par le modèle, et on retrouve des éléments de la classe de test dans les deux clusters de sorties. On peut déduire de ce diagramme que le choix est plus dur à faire pour le modèle qui va prédire des éléments comme étant des 0 et d'autres dans 1. Bien qu'une majorité d'élément soient regroupés avec des 1, les proportions sont trop proches pour faire des conclusions plus probantes.

5.2.3 Signature d'un jeu de test versus les chiffres de 0 à 8

Modèle crée à partir : des chiffres de 0 à 8 (limite de nos machines)
Classe introduite : jeu de test avec des chiffres non classés, ici des 1.
Nombre de layer : 5
Nombre de clusters : 10
Les résultats obtenus :

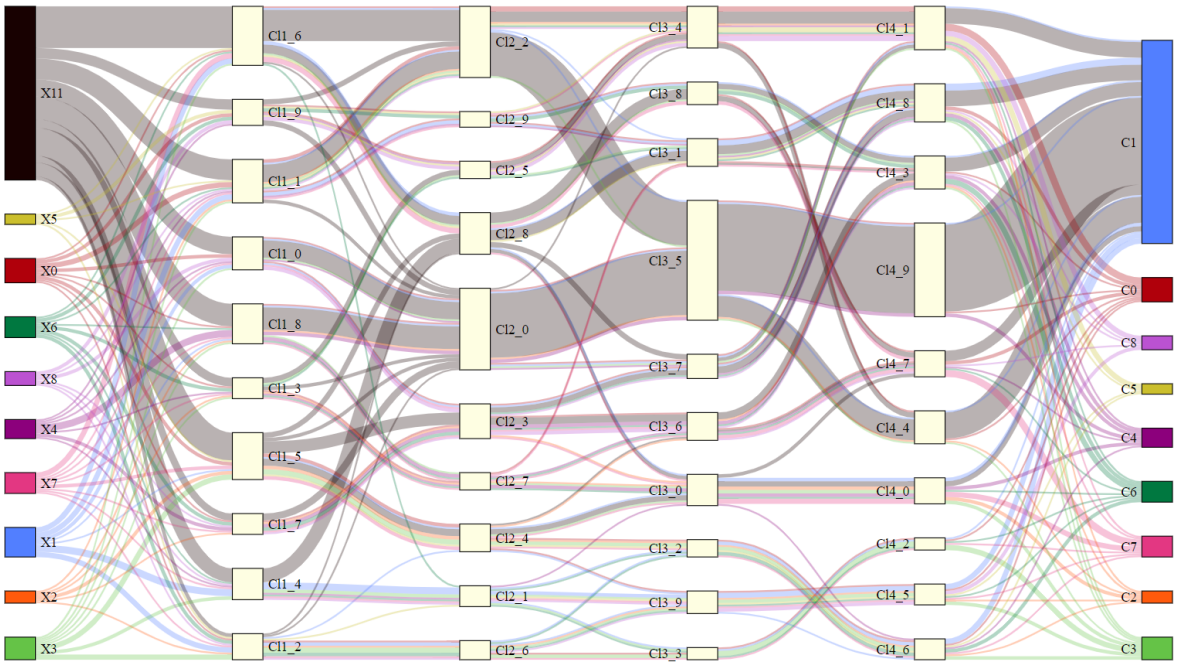


FIGURE 10 – Les chiffres de 0 à 8 avec des 1 non classés

Remarque(s) :
On observe des clusters mixtes pour toutes les classes, mais chacune des classes finissent dans le cluster qui leur est propre. Pour ce qu'il est de la classe de test, elle rejoint de multiples clusters avant de finir dans le cluster C1. On peut donc en déduire que le modèle fini par reconnaître les 1 inconnus comme étant effectivement de la même nature que les 1 qu'il connaît.

Conclusion sur cette donnée : A posteriori, les résultats ne sont pas aussi flagrants que sur iris. Cela est sûrement dû à la nature du jeu de données. Nous pouvons malgré tout suivre les chemins empruntés par les classes, et voir que le modèle réussi à classer effectivement les données si elles sont similaires, autrement les prédictions sont imprévisibles.

6 Gestion du Projet

6.1 Diagrammes de Gantt

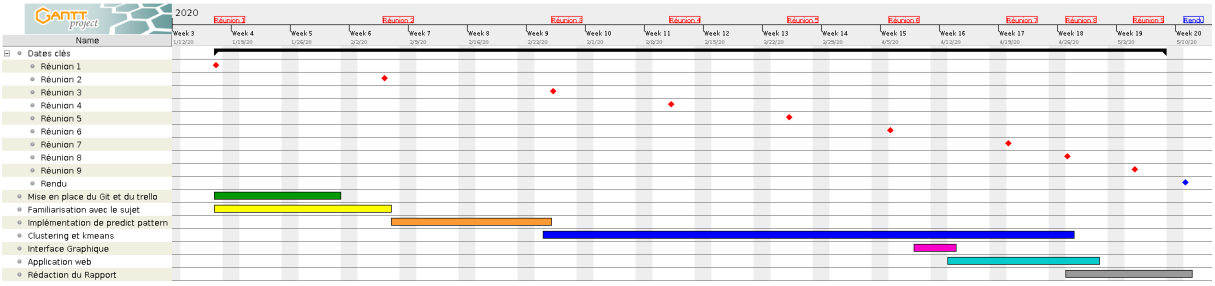


FIGURE 11 – Diagramme des tâches

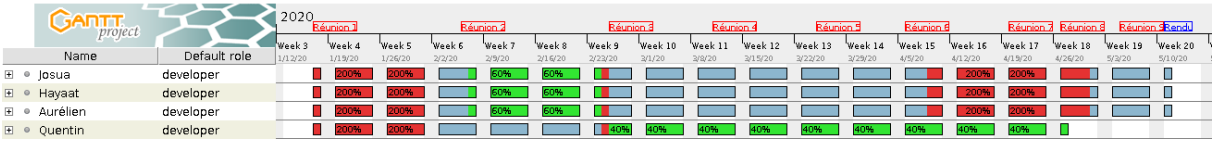


FIGURE 12 – Diagramme des ressources

6.2 Outils utilisés

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

Nous avons aussi organisé des réunions de groupe sur Discord qui nous ont permis de rester en contact même lors du confinement, malgré quelques soucis de communication dans cette période malgré tout.

Nous avons également utilisé GitHub pour les différents développements.

7 Conclusion

Au cours de ce TER, nous avons produit une application qui permet de visualiser les chemins empruntés par les éléments d'un jeu de données lors de leur classification, sous forme de diagrammes interactifs.

L'interface web de notre application pourrait être grandement améliorée avec plus de temps. Par exemple, fournir des diagrammes plus agréables à l'oeil et plus compréhensibles (comme le diagramme de l'onglet *Objectif d'affichage "optimal"*). De plus, on pourrait également englober le code dans un script plus ergonomique qui permettrait l'utilisation de ce dernier sur un jeu de données fourni par l'utilisateur.

Ce projet nous a permis de mieux comprendre le domaine des réseaux de neurones. Cependant, malgré la possibilité de voir le cheminement des données, 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 à rien de ce que l'on avait vu auparavant. Malgré tout le sujet reste une expérience intéressante et nous a donné des connaissances qui pourront potentiellement nous servir dans l'avenir.

Le code produit est disponible sur GitHub ici : https://github.com/JPhilippot/Patterns et la vidéo de présentation ici : https://youtu.be/-cfFWJnAjCo.

8 Bibliographie

Références

scikit-learn [en ligne]. The Iris Dataset [consulté le 25 janvier 2020]. https://scikit-learn.org/stable/auto_examples/datasets/plot_iris_dataset.html

MDN [en ligne]. Référence JavaScript [consulté le 2 avril 2020]. https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference

mohitatgithub [en ligne]. MNIST image classification with CNN Keras [consulté le 2 avril 2020]. https://mohitatgithub.github.io/2018-03-28-MNIST-Image-Classification-with-CNN-&-Keras/

bl.ocks.org [en ligne]. Sankey diagram using a csv file with v4 [consulté le 20 mai 2020]. https://bl.ocks.org/d3noob/06e72deea99e7b4859841f305f63ba85

UCI [en ligne]. Iris Data Set [consulté le 5 mai 2020]. Iris Data Set https://archive.ics.uci.edu/ml/datasets/iris

Wikipedia [en ligne]. MNIST database [consulté le 5 mai 2020]. https://en.wikipedia.org/wiki/MNIST_database

yann.lecun.com [en ligne]. THE MNIST DATABASE [consulté le 5 mai 2020]. http://yann.lecun.com/exdb/mnist/