Projet 18: Reseaux profonds

Kaan Sen, Ari Cilingiroglu, Fanny Goupil, Fabien Caballero
3e année de licence informatique

Encadrant : M. Poncelet

Année 2021-2022

Table des matières

1 Introduction

De nos jours, avec le développement des nouvelles technologies, de plus en plus d'applications dans des domaines aussi divers que les voitures autonomes, les jeux, les chatbot, etc. utilisent l'intelligence artificielle (IA) et parmi celle-ci des approches basées sur de l'apprentissage profond.
Il est vrai que pour des problèmes compliqués, les résultats obtenus favorisent ce développement. Par exemple, la classification d'images médicales avec des approches d'intelligence artificielle permet d'avoir des résultats plus efficaces qu'avec des êtres humains (une IA développée par google est capable de détecter le cancer du poumon mieux que les experts, pour en savoir plus se référer à [7]).

Outre les problèmes éthiques qui peuvent être associés à son utilisation. Par exemple, l'évaluation de la sécurité publique (ASP) est basée sur de l'IA qui prédit les potentielles arrestations criminelles pendant la mise en liberté provisoire. Elle se base sur 750 000 cas dans 300 juridictions à l'échelle nationale et prédit les personnes qui serons réincarcérées ou libérées. D'autres problèmes concernent la confiance que l'on peut réellement avoir dans les résultats obtenus.

L'objectif de ce projet est de mieux comprendre le fonctionnement des réseaux de neurones.

Dans le cas de notre projet, il s'agit de savoir si notre système nous classifie bien l'image et ce qu'il se passe si l'image ne correspond pas à un des résultats possible.

FIGURE 1 – classifieur d' image

Par exemple, imaginons qu'un classifieur soit appris pour faire la différence entre une image de tigre, de jaguar ou de guépard.
Dans la figure 1, si nous lui passons une image de tigre, le modèle prédira qu'il s'agit d'une image de tigre.
Par contre que se passe-t'il si on lui passe une image de voiture ? le classifieur ne sait faire la différence qu'entre 3 classes (tigre, jaguar et guépard) et il ne pourra prédire que l'une de ces trois classes.

C'est dans ce contexte que se situe notre travail : mieux comprendre comment fonctionne la classification d'images et voir s'il est possible d'émettre des doutes sur le résultat de la classification.

Le rapport est organisé de la façon suivante :
Les outils que nous avons utilisés pour le projet sont présentés dans la section 2.
Un rappel des principaux composants des réseaux de neurones est proposé dans la section3.
La section 4 présente les travaux que nous avons menés pour mieux comprendre comme fonctionne une classification.
Les différents résultats sont intégrés dans un site web qui est décrit dans la section 5 avec des détails concernant la librairie LIME utilisée pour mettre en évidence les parties de l'images utilisées pour la classification.
L'organisation du projet est décrite dans la section 6. Enfin, nous concluons le rapport.

2 Outils utilisés

Dans cette section, nous décrivons les principaux outils et librairies que nous avons utilisés pour réaliser ce projet. Le projet étant très accès science des données nous avons employé des librairies specifiques.

— tensorflow : plate-forme Open Source dédiée au machine learning.
— scikit learn : cette bibliothèque est destinée à l'apprentissage automatique.
— plotly : permet de faire des graphes de manière simple et efficace.
— Lime : permet d'interpréter les modèles de machine learning.
— Keras : API de prototypage rapide de réseau de neurones artificiel.

Nous avons exécuter notre code grâce à l'outil Google Collab qui permet d'exécuter du code python. Cela nous a permis de créer notre réseau de neurones. Et enfin, nous avons réalisé notre rapport avec Overleaf qui est un éditeur Lattex.

3 Rappels sur les réseaux de neurones

Le projet que nous avons mené utilise des réseaux de neurones aussi de manière à faciliter la compréhension du rapport, nous proposons un rappel sur les principaux concepts associés aux réseaux de neurones. Un réseau de neurones artificiels copie le cerveau humain pour favoriser l'apprentissage. Il s'agit donc d'un système qui se base sur le fonctionnement du cerveau humain pour l'adapter à des ordinateurs équipés de fonctions d'intelligence artificielle.

La création d'un réseau de neurones passe par trois principales étapes :

— La constitution d'une base de données qui serviront à l'entraînement et à l'évaluation du réseau
— Le choix d'une architecture de réseau de neurone
— L'entraînement du réseau à partir des données

Nous aborderons ces 3 points dans les prochaines parties.

3.1 Constitution d'une base de données d'images

L'entraînement d'un réseau de neurones requiert un nombre important de données. Dans notre cas, nous nous sommes plus particulièrement intéressés à des données de type images.
Par la suite, nous considérons que les données sont de type image.

Pour l'apprentissage supervisé, l'entraînement du réseau à besoin d'indications qui indiquent le résultat attendu sur les images d'entraînement. Dans notre cas (classification d'images), l'indication va indiquer à quoi correspond l'image.
exemple : image tigre -> tigre

Dans notre contexte, nous considérons que nous travaillons dans un cadre d'apprentissage supervisée, i.e. il existe un ensemble d'images possédant un label (e.g. tigre, jaguard et guépard). L'objectif de l'apprentissage supervisée est à partir d'un sous ensemble de ces images (jeu d'apprentissage), d'apprendre un modèle et de pouvoir évaluer la qualité du modèle sur le reste du jeu de données (jeu de test).

3.2 Entraînement réseau de neurone

Une fois que la base d'images est collectée et annotée, il faut entraîner le réseau de neurones.

Pour entraîner un modèle, on considère généralement 80% ou 70% des données en apprentissage et 20% ou 30% en test afin d'évaluer le modèle.

3.3 Architecture réseau de neurone

Un réseau de neurones est composé d'une série de couches de neurones, de sorte que tous les neurones de chaque couche se connectent aux neurones de la couche suivante.

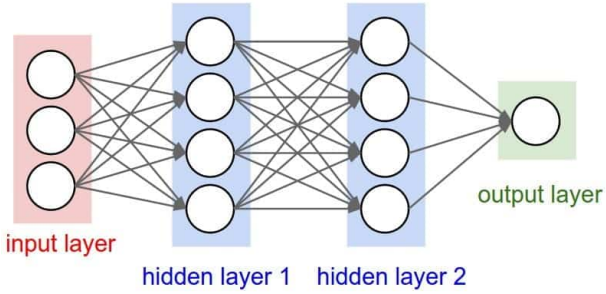


FIGURE 2 – réseau de neurones

Comme l'illustre la Figure 2, il existe plusieurs types de couches :

— La couche d'entrée (input layer) est un ensemble de neurones qui portent le signal d'entrée.
Elle contient un ensemble de neurones correspondant aux variables descriptives (i.e. les données de la base d'apprentissage) .
Tous les neurones de cette couche sont ensuite reliés à ceux de la couche suivante.

— La/les couches cachée(s) (hidden Layer) contiennent des neurones artificiels qui prennent un ensemble d'entrées pondérées et produisent une sortie via une fonction d'activation.
Cette transformation sera différente selon chaque neurone car chaque neurone possède un poids différent.

Une fonction d'activation sert avant tout à modifier de manière non-linéaire les données.
Il existe différents type de fonction d'activation. Voici les principales :
— relu : permet d'effectuer un filtre sur nos données (C.f. Figure 3) Elle laisse passer les valeurs positives ($x > 0$) dans les couches suivantes du réseau de neurones.

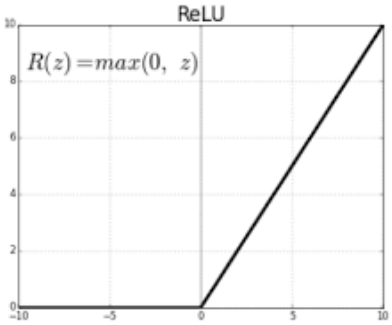


FIGURE 3 – graphe relu

— sigmoid : donne une valeur entre 0 et 1, une probabilité. Elle est très utilisée pour les classification binaire (C.f. figure 4).

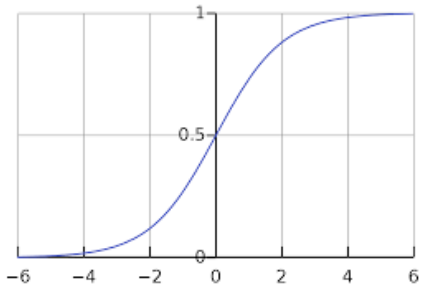


FIGURE 4 – graphe sigmoid

— softmax : permet de transformer un vecteur réel en vecteur de probabilité (C.f. figure 5).

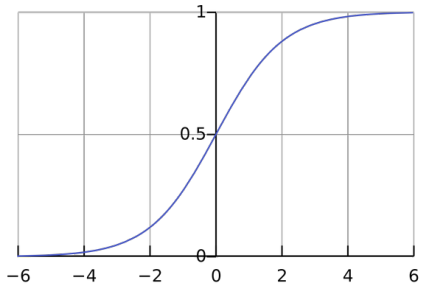


FIGURE 5 – graphe softmax

— La couche de sortie (output layer) : à une valeur qui correspond à la valeur sortant de la fonction d'activation. C'est elle qui permet la classification de l'image.

Dans le cadre de nos travaux, nous avons travaillé sur des images, i.e. une image est une matrice de pixels. Ainsi, par rapport à l'architecture que nous venons de décrire, il est indispensable de rajouter une ou plusieurs couches de convolutions (CNN - *Convolutional Neural Network (en français, réseau neuronal convolutif)*). Le lecteur intéressé par le fonctionnement de ces couches peut se reporter à [11]

4 Travail réalisé

Nous avons vu en introduction que notre objectif était de mieux comprendre comment fonctionnait les réseaux de neurones et étudier si nous pouvions expliquer les raisons d'une classification.

Nous venons de voir qu'un réseau de neurones possède plusieurs couches dont le résultat de la classification dépend de celles-ci.

Pour vérifier à quel point ce résultat est fiable, nous pouvons regarder comment chacun des objets se comportent et se positionnent entre eux.

4.1 Comment récupérer les données en sortie de fonction d'activation ?

La première chose à faire est de concevoir un modèle pour traiter des images. Nous avons choisi au début le jeu de donnée tigre,guépard ou jaguar (cf : figures 6,7,8,9 qui illustrent quelques exemples d'images du jeu de donnée).

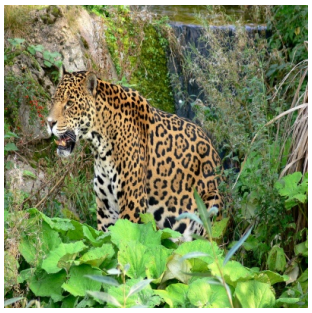


FIGURE 6 – Jaguar

FIGURE 7 – Guépard

FIGURE 8 – Guépard

FIGURE 9 – Tigre

Nous avons un modèle Tigre, Guépard ou Jaguar caractérisé par une couche d'entrée, 6 couches cachées et une couche de sortie (cf figure 10).

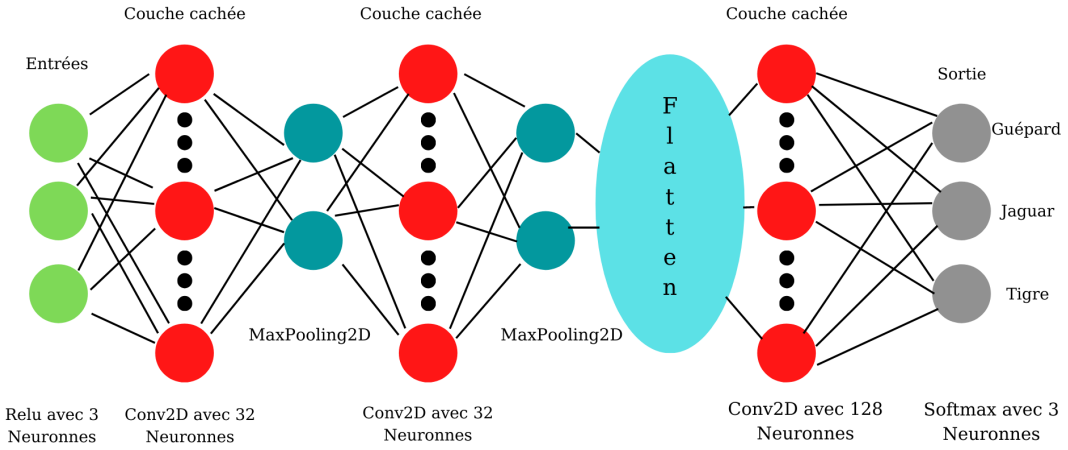


FIGURE 10 – CNN

Via Keras, il est possible de facilement définir un modèle comme l'illustre le listing suivant.

```
model = tf.keras.Sequential([
    tf.keras.layers.Conv2D(32, (3, 3), activation = 'relu', input_shape = (150, 150, 3)),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Conv2D(32, (3, 3), activation = 'relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(128, activation=tf.nn.relu),
    tf.keras.layers.Dense(3, activation=tf.nn.softmax)
])
```

Par la suite il faut pouvoir extraire la valeur des fonctions d'activation de chaque couche si nous voulons pouvoir mieux comprendre ce qu'il se passe et comment les données évoluent au sein du réseau. Heureusement pour cela, Keras propose également une méthode :

```

 from keras import models
layer_outputs = [layer.output for layer in model.layers]

activation_models=models.Model(inputs=model.input,outputs=layer_outputs)
activations = activation_models.predict(test_images,verbose=0)

pred_labels=np.argmax(activations[6],axis=1)

activations[0-6]

test_layers=[]
for i in range(len(activations)):
  test_layers.append(activations[i])
```

4.2 Utilisation d'un algorithme d'apprentissage non supervisé

Jusqu'à présent, nous étions dans un contexte d'apprentissage supervisé (les données traitées sont étiquetées).
Notre objectif maintenant est d'utiliser un algorithme d'apprentissage non supervisé pour permettre de faire des clusters des données à la sortie d'une couche.
Le principe d'un algorithme non supervisé est de créer ou regrouper des objets pour constituer des groupes où la distance à l'intérieur d'un groupe est minimale et la distance entre groupes est maximale. Il existe de très nombreux algorithmes non supervisé, le lecteur intéressé peut se reporter à [21]

Dans notre contexte, nous avons choisi d'utiliser K-means car il est simple d'utilisation. Cependant il nécessite de préciser le nombre de clusters.
Une méthode consiste à utiliser la méthode du coude (cf figure 11). Le lecteur intéressé pourra en apprendre plus ici : [23])
Dans le cadre de notre projet, nous avons choisi le nombre de clusters pour pouvoir voir la part d'erreur plus clairement et générer plus facilement des images mal classifiées (ex : image de tigre donne guépard).

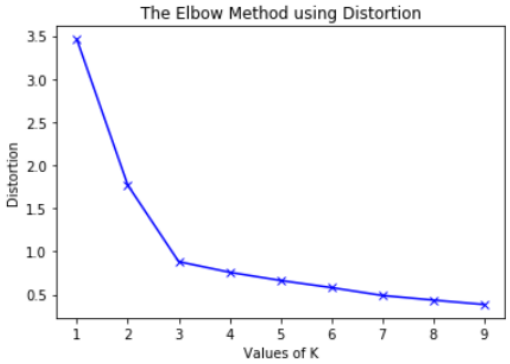


FIGURE 11 – méthode du coude

Dans la Figure 11, nous voyons que le nombre de clusters optimal à choisir est de 3.

Comme l'indique notre listing ci-dessous, nous avons choisi 4 clusters pour notre 1ere couche.
Nous avons groupé notre sortie de la couche par leur distance aux centres des clusters sur l'algorithme kmeans, et nous avons enregistré les coordonnées de chaque cluster.

```
nb clusters=4
kmeans = KMeans (n_clusters=nb_clusters, init= 'k-means++',
            max_iter = 300,
            n_init = 10,
            random_state = 0)
df0=kmeans.fit_predict(df0)

centers0= kmeans.cluster_centers_
center_index0=[]

for i in range(len(centers0)):
      center_index0.append(i)
center_index0 = [str(x) for x in center_index0]
center_index0 = pd.DataFrame (center_index0, columns=['cluster']);
```

Maintenant il serait intéressant de les visualiser. On le fait avec ACP qui permet d'observer les corrélation entre nos données.
Les données étant en grande dimension, il n'est pas possible de les visualiser. Pour pouvoir le faire, nous avons utilisé une analyse en composante principale afin de projeter les données dans un espace en 2D.
La figure 12 illustre un exemple de sortie des données après ACP :

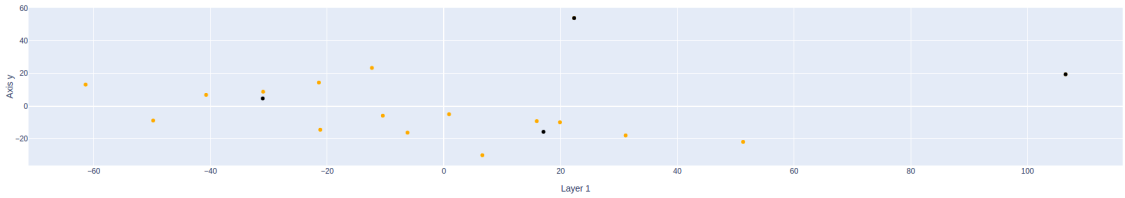


FIGURE 12 – Couche cachée 1 du modèle Tigre, Jaguar, Guépard avec 4 clusters

On peut apercevoir sur la figure 12 des points noirs qui correspondent à nos clusters et des points jaunes qui sont les données obtenus en sortie de la fonction d'activation de la 1ère couche.

4.3 Comment offrir une visualisation du comportement global ?

Pour visualiser le comportement global on utilise des graphes parallèles. Pour cela on utilise Plotly qui est utilisé pour la visualisation de données et qui prend en charge divers graphiques dont les graphes parallèles.

La figure 13 illustre un exemple de graphe parallèle pour notre modèle Tigre Guépard Jaguar.

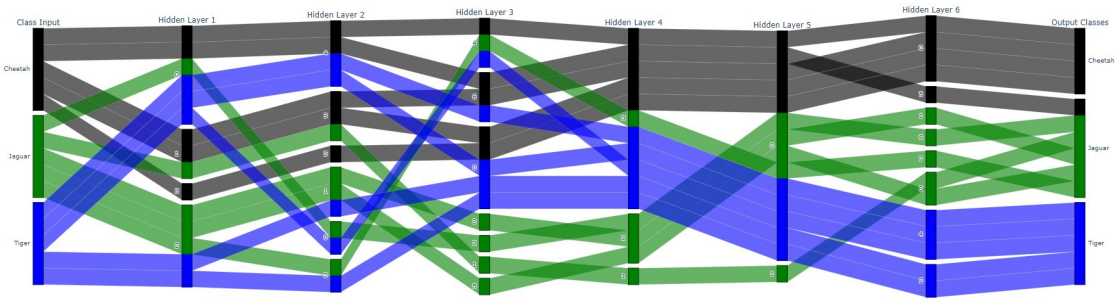


FIGURE 13 – Graphe Parallèle Modèle Tigre Jaguar Guépard

Cet exemple visualise comment chaque élément de chaque classe est interprété par chaque couches. Ici, nous pouvons émettre quelques doutes sur la fiabilité du résultat car les chemins empruntés sont pour certains éléments très variable. Cela signifie que durant le parcours du réseau de neurone, l'élément a été reconnu comme différentes classes.
Suivant la figure 13, un élément peut avoir un parcours du réseau "aléatoire" mais être bien classifié. Cela ne veut pas dire pour autant que le modèle est fiable.

4.4 Intégration dans le Site Web

Pour visualiser et mieux comprendre les réseaux de neurones, nous avons réalisé un site web en HTML, CSS et Javascript.

Vous y trouverez 3 différents modèles :
— Paysages
— Transports
— Jaguars, Tigres et Guépards
Dans le but d'utiliser le travail effectué sur google Colab (permet d'écrire et exécuter du code python dans un navigateur), nous avons importé chacun de nos modèle en HTML.

Ainsi, on a pu réutiliser les images et les schémas de chacun de nos modèles pour les inclure dans notre site web.

Vous trouverez notre site web à l'adresse : `https://fabiencaballero.fr/ProjetML/Accueil.html`

5 Expérimentations

5.1 LIME

Pour plus de compréhension, nous avons utiliser la bibliothèque LIME pour avoir une meilleure idée de pourquoi une image à été classifiée comme étant un tigre plutôt qu'un guépard.
En effet, cette bibliothèque met en surbrillance la zone de l'image utilisée lors de la classification.

FIGURE 14 – Image de Guépard

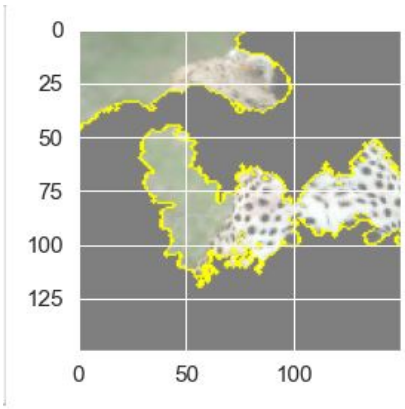


FIGURE 15 – Résultat donné par LIME avec Fig 15

Cette image de guépard est classifiée comme étant un jaguar.
En effet, la différence entre un guépard et un jaguar n'est pas très grande et on ne visualise pas bien la tête du guépard.
Grâce au modèle lime on voit que l'intelligence artificielle c'est basé sur le haut de la tête du guépard semblable à celle d'un jaguar et aux taches sur sa fourrure aussi présentes chez le jaguar.
Les deux étant très proches, l'élément se retrouve entre 2 clusters. C'est pourquoi il est classé comme étant un jaguar au lieu d'un guépard.

Vous retrouverez sur le site d'autres images mal classifiées avec lesquelles on a utilisé LIME pour essayer de comprendre pourquoi la classification n'a pas donné le résultat attendu.

5.2 Quelques images du site

Dans cette section, vous pourrez voir certains visuels du site et le détail des pages.

FIGURE 16 – Accueil

La figure 16 correspond à la page d'accueil, vous y trouverez une phrase d'accroche concernant les réseaux de neurones ainsi que 3 images cliquables. Chacune d'elle correspond à un des modèles et renvoie sur la page de leur modèle.

En cliquant sur le point d'interrogation en haut à droite, vous pourrez trouver des informations détaillées sur un réseau de neurones.
Vous pouvez naviguer à travers les différentes pages via le menu apparaissant en cliquant sur l'icône en haut à gauche.

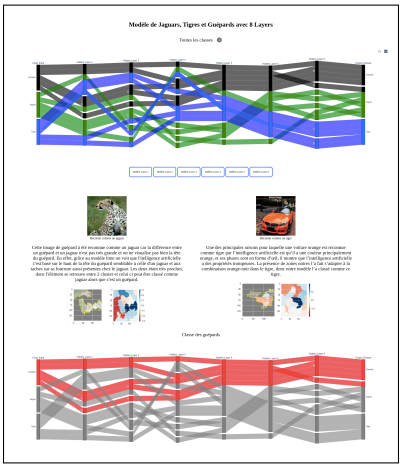


FIGURE 17 – Page Jaguar Tigre Guépard - 8 layers

La figure 17 correspond à un exemple de page d'un modèle. Celle-ci correspond au modèle de Jaguar, tigre guépard avec 8 Layers.

Vous pouvez y trouver un graphe parallèle interactif qui permet d'observer la classification de tous les objets de ce modèle et de pouvoir les comparer entre eux (cf figure 18).

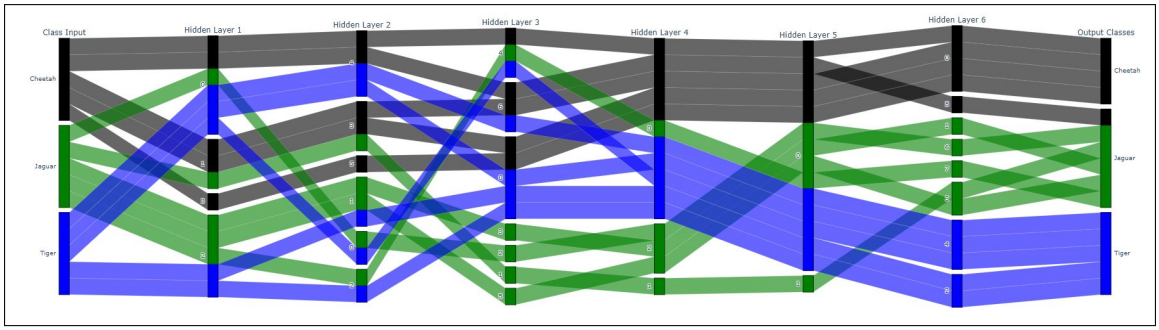


FIGURE 18 – Page Jaguar Tigre Guépard 8 layers - graphe parallèle

Pour les boutons en dessous de celui-ci, ils permettent lors d'un clic de visualiser la répartition des données dans le layer correspondant. (cf figure 19)

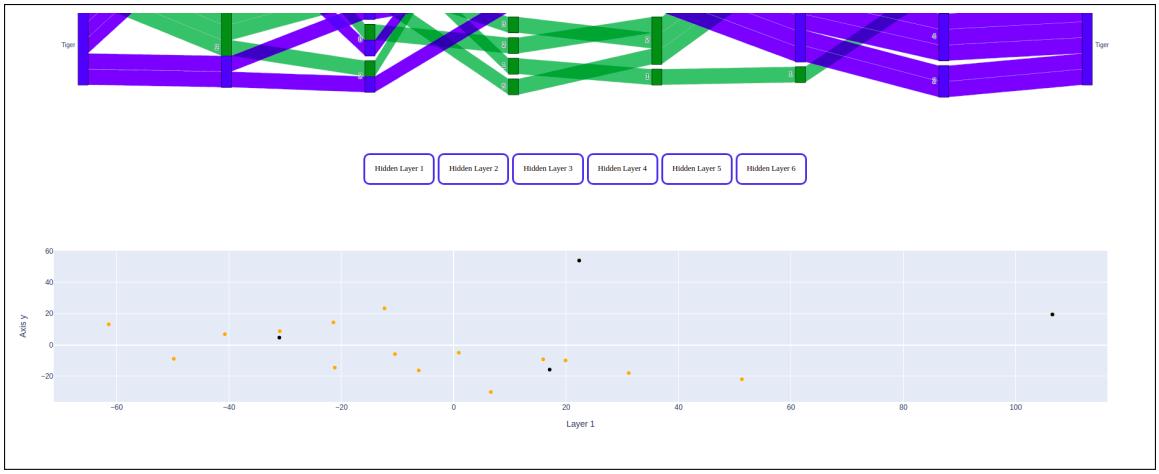


FIGURE 19 – Page Jaguar Tigre Guépard 8 layers - nuage de point

Vous pourrez également retrouvé deux images mal classifiées, une appartenant au modèle et une autre qui n'y appartient pas (cf figure 20).

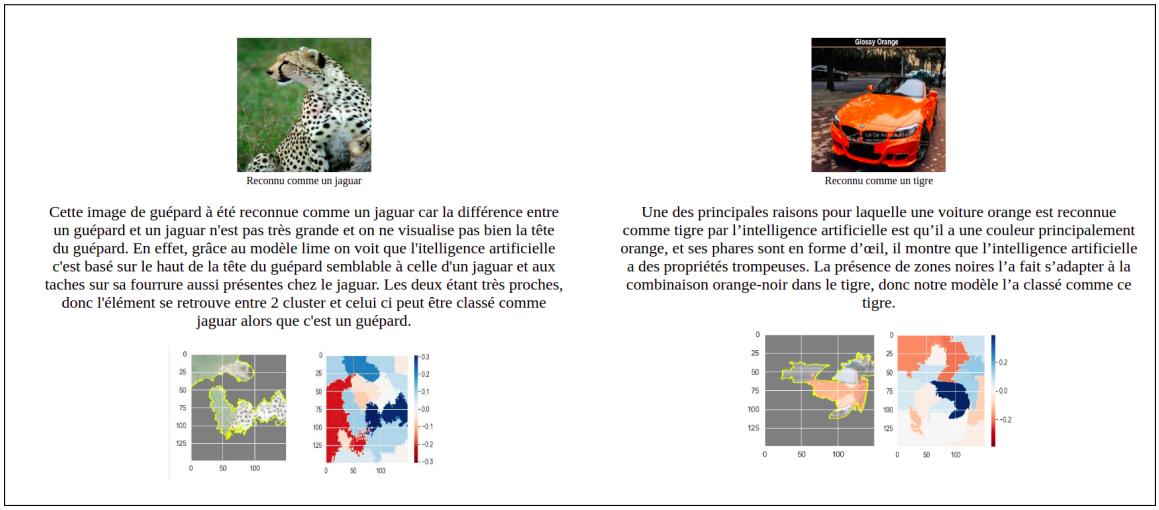


FIGURE 20 – Page Jaguar Tigre Guépard 8 layers - images mal classifiées

Lorsque vous cliquez sur l'image appartenant au modèle (dans notre cas le guépard),celle ci renvoie sur le graphe parallèle correspondant à sa classe réelle (ici, le guépard).
En dessous de nos images mal classifiées, vous pourrez y trouvez une brève description sur la raison pour laquelle l'image est mal classifiée. Pour cela nous nous sommes appuyé sur les résultats obtenus grâce à une analyse avec LIME.

Vous pourrez trouver d'autres exemples de modèles avec plus ou moins de couches en navigant grâce à notre menu (cf figure 21).

FIGURE 21 – Menu

6 Organisation du travail

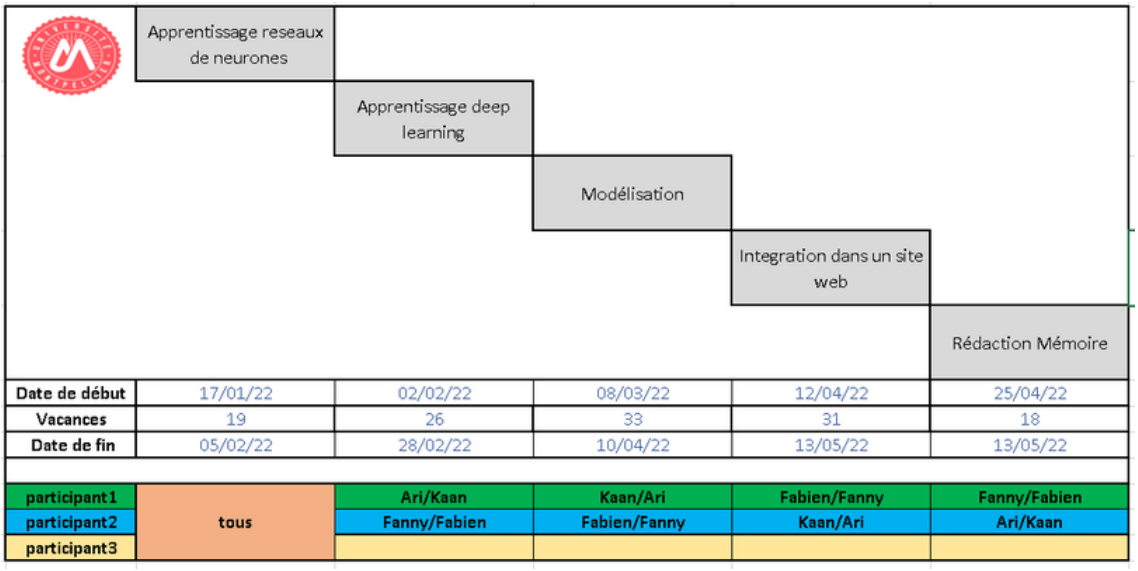
Dans cette section, vous trouverez des détails concernant l'organisation du projet.
Le projet c'est déroulé en 2 parties :

— Une première où l'on a appris ce que sont les réseaux de neurones via des réunions organisées quotidiennement (toutes les deux semaines) ainsi que par internet.
Pour cette première partie, tout le groupe travaillait ensemble. L'objectif étant de réaliser un premier réseau de neurones d'un modèle simple (iris) avec des graphes pour visualiser le comportement des objets du modèle (cf travail réalisé). Pour cela nous avons fixé des horaires où l'on se rejoignait en salle informatique pour travailler en groupe.

— La seconde partie à été d'appliquer ce que l'on a appris dans la 1ere partie à des modèles plus complexes pour réaliser une classification d'image. Nous avons réaliser un premier modèle tous ensemble. Puis Ari et Kaan ont appliqué la même chose pour différents modèles.
Pendant ce temps, Fanny et Fabien s'occupaient du site et du rapport. Cette partie était en majorité réalisée en distanciel. Nous avons communiqué et partagé les ressources créés et l'avancée du projet via les réseaux.

Lorsqu'un ensemble de tâche était finie, nous le partagions à l'ensemble du groupe qui le vérifiait ou partageait leurs avis afin de potentiellement apporter des modifications.

Vous trouverez ci-dessous le diagramme de Gantt (cf figure 22) :



Date de début	17/01/22	02/02/22	08/03/22	12/04/22	25/04/22
Vacances	19	26	33	31	18
Date de fin	05/02/22	28/02/22	10/04/22	13/05/22	13/05/22
participant1	**tous**	**Ari/Kaan**	**Kaan/Ari**	**Fabien/Fanny**	**Fanny/Fabien**
participant2		**Fanny/Fabien**	**Fabien/Fanny**	**Kaan/Ari**	**Ari/Kaan**
participant3					

FIGURE 22 – Diagramme de Gantt

7 Conclusion

Pour conclure, lors de ce projet nous avons pu voir ce qu'était un réseau de neurones et comment il fonctionne pour classifier une image.
Cependant, comme on a pu l'observer, la classification dépend de nombreux facteurs comme la création du modèle, des données d'entraînement équilibrées ainsi que de fortes ressemblances entre deux classes du modèle.

Pour vérifier si le résultat obtenu est fiable, nous pouvons regarder comment chacun des objets se positionnent et se comportent entre eux grâce à des graphes.

Par exemple dans le schéma ci dessous :

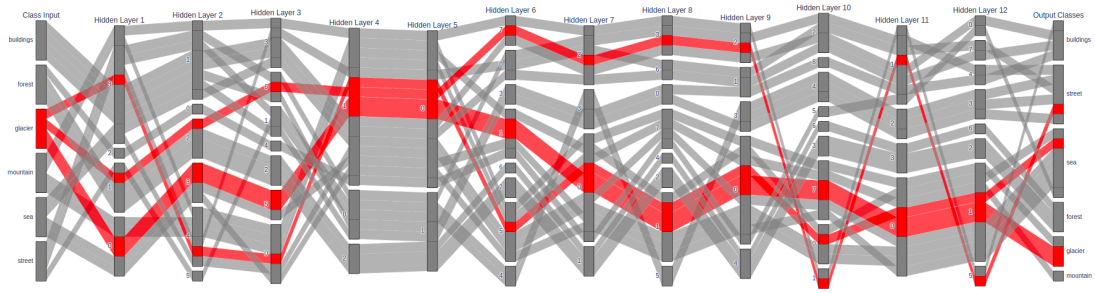


FIGURE 23 – graphe parallèle - modèle Paysages

On voit que les images de glaciers ont des caractéristiques communes avec la rue, et la mer en plus des glaciers.
En effet, prenons maintenant une image (fig 24) de notre modèle classifiée comme étant la mer au lieu d'un glacier.

FIGURE 24 – Glacier considéré comme mer

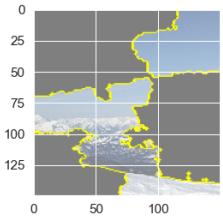


FIGURE 25 – Zones de l'image utilisé pour classification

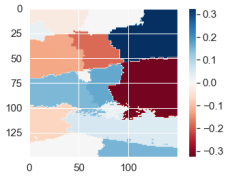


FIGURE 26 – Poids des zones de l'images

On peut observer grâce à LIME que la partie de l'image utilisée pour classifier est majoritairement le ciel (cf figure 25) car celui-ci a le poids le plus élevé (fig 26). Le ciel étant bleu, c'est pourquoi il le classifie comme étant la mer.

Grâce à ces outils on peut émettre des doutes sur la fiabilité des résultats.

Pour aller plus loin, nous avions penser à intégrer à notre site web, une page permettant à l'utilisateur d' importer son propre modèle et de voir les résultats de celui-ci.
De même, nous avons pensé à utiliser les connaissances acquises lors de ce projet à un système de reconnaissance faciale.

8 Bibliographie

Références

[1] Application Python avec des concepts d'entraînement, de validation et de test dans l'apprentissage automatique. https://yigitsener.medium.com/makine-C3B6C49Frenmesinde-train-validation-ve-test-kavramlarC4B1-ile-python-uygulamasC4B1-a60214fb5 Dernier Accès : 05-05-2022.

[2] Classification Example with Linear SVC in Python. https://www.datatechnotes.com/2020/07/classification-example-with-linearsvm-in-python.html?m=1. Dernier Accès : 05-05-2022.

[3] Click Events in Python. https://plotly.com/python/click-events. Dernier Accès : 05-05-2022.

[4] CNN tutorial(keras-nb2). https://www.kaggle.com/bavalpreet26/cnn-tutorial-keras-nb2. Dernier Accès : 05-05-2022.

[5] Convert ipynb notebook to HTML in Google Colab. https://stackoverflow.com/questions/53460051/convert-ipynb-notebook-to-html-in-google-colab. Dernier Accès : 05-05-2022.

[6] Convolutional Neural Network : Feature Map and Filter Visualization. https://towardsdatascience.com/convolutional-neural-network-feature-map-and-filter-visualization-f75012a5a49c. Dernier Accès : 05-05-2022.

[7] Google a développé une intelligence artificielle capable de détecter les cancers du poumon. https://sante.lefigaro.fr/article/google-a-developpe-une-intelligence-artificielle-capable-de-detecter-les-cancers-du-poumon/#:~:text=Google20a20dC3A9veloppC3A920une20intelligence20artificielle20capable20de20dC3A9tecter20les2-Par20Agathe20Delepaut&text=Une20C3A9quipe20de20chercheurs20a,de20prC3A9cision20que20des20mC3A9decins. Dernier Accès : 05-05-2022.

[8] How to print the output of each layer of CNN model after passing an image. https://stackoverflow.com/questions/54632416/how-to-print-the-output-of-each-layer-of-cnn-model-after-passing-an-image. Dernier Accès : 05-05-2022.

[9] How to Use LIME to Understand sklearn Models Predictions. https://coderzcolumn.com/tutorials/machine-learning/how-to-use-lime-to-understand-sklearn-models-predictions. Dernier Accès : 05-05-2022.

[10] Image Classification using Deep Neural Networks — A beginner friendly approach using TensorFlow. https://medium.com/@tifa2up/image-classification-using-deep-neural-networks-a-beginner-friendly-approach-using-tensorflow-94b0a Dernier Accès : 05-05-2022.

[11] Image Classification with Convolutional Neural Networks. https://towardsdatascience.com/image-classification-with-convolutional-neural-networks-12a7b4fb4c91. Dernier Accès : 05-05-2022.

[12] Interactive HTML Export in Python. https://plotly.com/python/interactive-html-export. Dernier Accès : 05-05-2022.

[13] Introducing Plotly Express. https://medium.com/plotly/introducing-plotly-express-808df010143d. Dernier Accès : 05-05-2022.

[14] k-NN classifier for image classification. https://www.pyimagesearch.com/2016/08/08/k-nn-classifier-for-image-classification. Dernier Accès : 05-05-2022.

[15] Kaggle Intel Image Classification. https://medium.com/@FastAIButMakeItSlow/kaggle-intel-image-classification-94-accuracy-project-49b2848c335a. Dernier Accès : 05-05-2022.

[16] L'apprentissage automatique est amusant ! Partie 3 : Apprentissage en profondeur et réseaux de neurones convolutifs. https://medium.com/deep-learning-turkiye/makine-C3B6C49Frenimi-eC49Flencelidir-3-kC4B1sC4B1m-2253856ad79c. Dernier Accès : 05-05-2022.

[17] Les réseaux de neurones convolutifs. https://www.natural-solutions.eu/blog/la-reconnaissance-dimage-avec-les-rseaux-de-neurones-convolutifs. Dernier Accès : 05-05-2022.

[18] Parallel Categories Diagram in Python. https://plotly.com/python/parallel-categories-diagram. Dernier Accès : 05-05-2022.

[19] Plotting Predictions With A Confusion Matrix. https://www.kaggle.com/bavalpreet26/cnn-tutorial-keras-nb2#Plotting-Predictions-With-A-Confusion-Matrix. Dernier Accès : 05-05-2022.

[20] Plotting Predictions With A Confusion Matrix. https://towardsdatascience.com/visualizing-intermediate-activation-in-convolutional-neural-networks-with-keras-260b36d60d0. Dernier Accès : 05-05-2022.

[21] Principaux algorithmes d'apprentissage non supervisé. https://fr.linedata.com/principaux-algorithmes-dapprentissage-non-supervise. Dernier Accès : 05-05-2022.

[22] Principaux algorithmes d'apprentissage non supervisé. https://datascientest.com/algorithme-des-k-means. Dernier Accès : 05-05-2022.

[23] Tout ce que vous voulez savoir sur l'algorithme K-Means. https://mrmint.fr/algorithme-k-means. Dernier Accès : 05-05-2022.

[24] Vegetable Image Classification using CNN. https://medium.com/mlearning-ai/vegetable-image-classification-using-cnn-6f1d1be75cfb. Dernier Accès : 05-05-2022.

[25] Visualizing intermediate activation in Convolutional Neural Networks with Keras. https://towardsdatascience.com/visualizing-intermediate-activation-in-convolutional-neural-networks-with-ke Dernier Accès : 05-05-2022.