

# Rapport du projet Grey's

Ajmi Doriane

Bouge Yoann

Makhlouf Mohamed El Mehdi



## **Lettre aux membres du jury :**

Mesdames, Messieurs, les membres du jury,  
Il a été pour nous très enrichissant de réaliser ce projet. Cela nous a permis d'expérimenter des outils qui ne font pas parties de notre formation actuelle. Durant ces mois de travail, nous avons tous été challengés sur nos capacités d'organisation, de recherches de données, nos prises de décision et notre créativité. Nous avons aussi appris à travailler en équipe, à s'écouter et donner chacun de soi-même pour l'aboutissement de ce projet. Ce projet nous tient particulièrement à cœur car il s'agit d'un petit pas pour nous, dans un univers qui nous passionne et qui, jusque-là, nous était mystérieux. Bonne lecture.

L'équipe Grey's



## **Remerciement :**

Nous tenons à exprimer nos remerciements aux enseignants Rémy Cazabet, Fabien Rico, Samir Aknine, Matthieu Moy et Yannick Vacher pour la mise en place des projets et le suivi au cours de ce semestre. Nous souhaitons également remercier Mr Alexandre Meyer qui a suivi toutes les étapes de ce projet, qui nous a accompagné et éclairé jusqu'à la fin.

# **Table des matières**

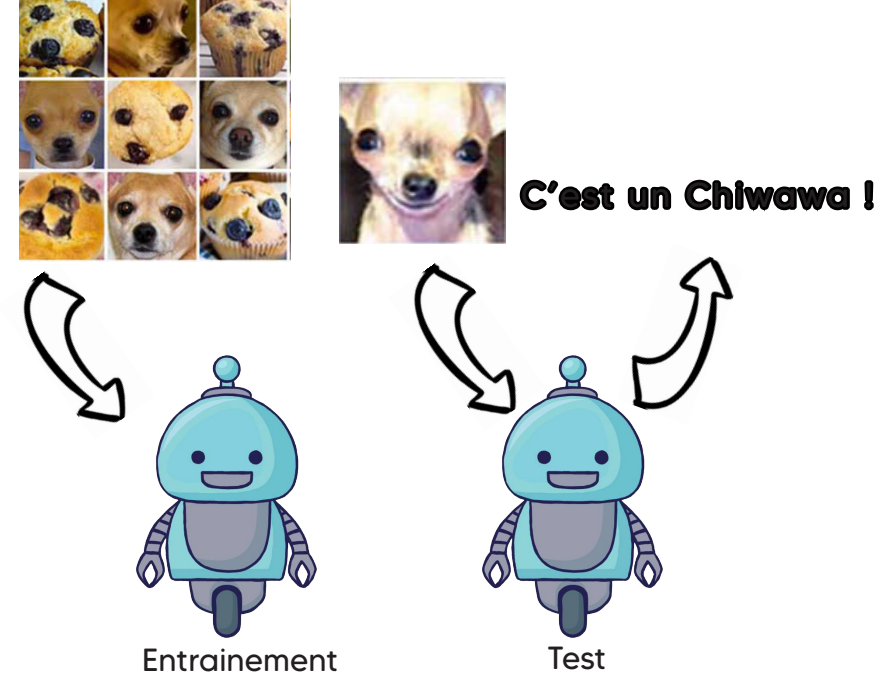
|   |  |    |
|---|--|----|
|    | 1- Sujet/Enjeux .....                  | 3  |
|    | 2- Environnement de travail .....      | 4  |
|    | 3- Bases de données et traitements.... | 4  |
|    | 4- Réseau de neurones.....             | 5  |
|    | 5- Algorithme .....                    | 6  |
|    | 6- É valuation .....                   | 7  |
|    | 7- Interface .....                     | 8  |
|   | 8- Conclusion/ Perspectives .....      | 9  |
|  | 9- Annexes .....                       | 10 |

# Sujet/Enjeux

L'intelligence artificielle ne date pas d'aujourd'hui. En effet, sa véritable naissance pourrait coïncider avec **l'apparition des ordinateurs dans les années 1940-1950**. Le **deep learning** est un type d'intelligence artificielle qui dérive du machine learning. Il s'appuie sur ce qu'on appelle **un réseau de neurones** et s'alimente de données. Aujourd'hui, l'accès aux données est facilité.

Nous avons donc pu réaliser des algorithmes de reconnaissance d'images grâce à un entraînement sur une multitude d'images. Pour ce faire, on présente à l'algorithme, un très grand nombre d'images en lui **indiquant à chaque fois ce qu'elles contiennent** : c'est ce qu'on appelle **l'apprentissage supervisé (voir figure 1)**. L'algorithme pourra ensuite reconnaître une image qu'il n'a jamais vue avant, à condition qu'il ait déjà été entraîné pour ce type. En effet, avec l'apprentissage supervisé, l'algorithme est guidé : les informations lui sont fournies et sont annotées de la bonne manière.

Il existe un autre type d'apprentissage qui est **l'apprentissage non supervisé**. Dans ce cas, la machine est totalement autonome : les données lui sont fournies mais ne sont pas annotées, ainsi la machine va les trier d'elle-même par catégorie.



**Figure 1 : Apprentissage supervisé**

Actuellement, le deep learning est un **outil utilisé en médecine**, particulièrement en imagerie médicale. Cela a pour but d'aider les médecins à établir un diagnostic mais aussi à aider dans les **pratiques cliniques du quotidien**. En effet, une multitude de données d'images médicales existent actuellement. Celle-ci sont stockées et annotées. L'accès à ces données permet aux machines de s'entraîner à reconnaître des anomalies en **comparant un cas sain à un cas pathologique**. Il est également possible de classer les différentes anomalies et de les faire apprendre (par un apprentissage supervisé) à la machine. Celle-ci pourra, à la vue d'une nouvelle image, reconnaître s'il y a une anomalie, et dans ce cas, déterminer de quel type il s'agit.

Ainsi, les professionnels de santé pourront utiliser le deep learning comme un **réel accompagnateur de prise de décision** : cette technique est précise et permet de réduire les risques d'erreurs. De plus, c'est un outil qui, contrairement à l'homme, **ne se fatigue pas et ne nécessite pas de pause**. La machine sera alors tout aussi efficace peu importe le moment où elle est sollicitée. Par exemple l'intelligence artificielle créée par **Therapixel** permet de dépister le cancer du sein avec un taux de détection supérieur à 99% en un temps inférieur à 5 millisecondes grâce à une base de données de 640 000 images : cela amène à une capacité de détection supérieure à la moyenne des radiologues.

Le deep learning nécessite des **données parfaitement qualifiées et annotées cliniquement**. Actuellement, un grand nombre de données en imagerie médicale sont biaisées, notamment par une mauvaise annotation. Ces erreurs ne sont pas corrigées par la machine et sont apprises telles quelles. Ce qui amène à se demander si l'on peut **réellement faire confiance à une machine en ce qui concerne le diagnostic**. C'est pourquoi l'idée de laisser la décision à un ordinateur **n'est pas envisageable**, mais cela reste une assistance aux professionnels de santé.

Pour notre projet, nous avons décidé de construire **un réseau de neurones artificiels** qui permet de réaliser de **l'apprentissage supervisé** sur différents types d'**imageries médicales**. Nous avons donc pu étudier le fonctionnement d'un réseau de neurones en fonction de la base de données et **évaluer son efficacité en conséquence**.

# Environnement de travail

Pour réaliser notre projet nous avons choisis **Python** comme langage de programmation. Il comporte plusieurs choix de bibliothèques utilisables pour créer un réseau de neurones et une forte communauté. Nous avons donc mis en place un environnement sur **Anaconda**. Nous y avons ajouté les bibliothèques **Numpy**, **Matplotlib** et **PyTorch**.

L'entraînement du réseau de neurones se fait sur **la carte graphique** et nos machines ne sont pas adaptées. Nous avons donc opté pour lancer nos entraînements sur un **serveur en ligne : Kaggle**.

Nous avons par la suite, réalisé une interface utilisateur avec **TKinter**. Nous l'avons choisi car c'est une bibliothèque graphique rapide à prendre en main et suffisante pour l'utilisation qu'on veut en faire. Elle nous permet d'avoir une **interface propre avec tous les éléments dont nous avons besoin**.



## Bases de données et traitements

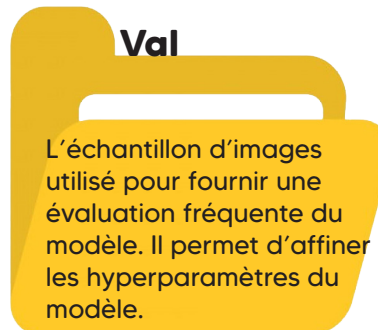
Nous avons choisi quatre bases de données qu'on a récupéré sur Kaggle : **mélanome**, **tumeur au cerveau**, **cellule sanguine** et **OCT (Tomographie en cohérence optique)**. Le contenu de chaque base de données est décrit en annexe 1. Pour la base de données mélanome, toutes les images étaient dans un même répertoire et nommées par des nombres. Ce répertoire était accompagné d'un document Excel où chaque image (donc son numéro) était associée à son type. Nous avons donc réalisé un script en Python. Ce script permet, à **partir d'un fichier Excel, de ranger chaque image dans la classe qui lui correspond**.

Nous avons également remarqué que les classes de certaines bases de données n'étaient pas équitables en nombres d'images. Nous avons estimé, afin d'avoir de meilleurs résultats, qu'il valait mieux **homogénéiser le nombre d'images pour chaque classe**. Nous avons donc réalisé un script en Python qui modifie les images. Nous avons **modifié la luminosité, le contraste, la brillance et la rotation**. Ainsi, la machine perçoit ces images comme des **nouvelles images** et s'entraîne avec. Nous avons donc pu homogénéiser toute nos classes. Dans chaque base de données, nous avons **trois dossiers qui comportent toutes les classes et un fichier Json** :



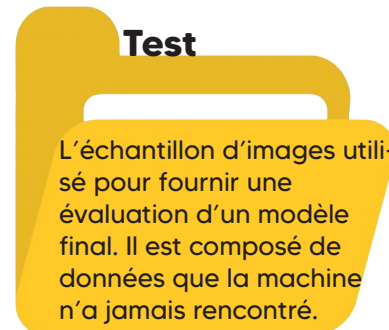
**Train**

L'échantillon d'images pour entraîner le modèle. Il va apprendre à partir de ces données annotées.



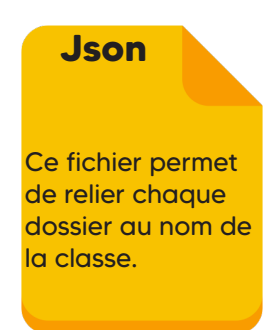
**Val**

L'échantillon d'images utilisé pour fournir une évaluation fréquente du modèle. Il permet d'affiner les hyperparamètres du modèle.



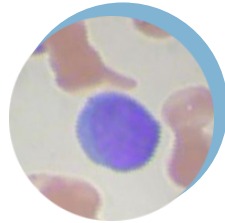
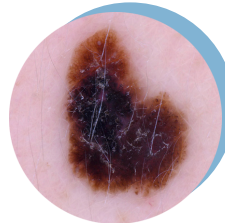
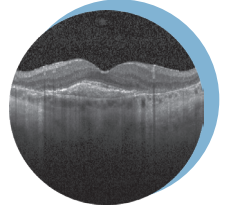
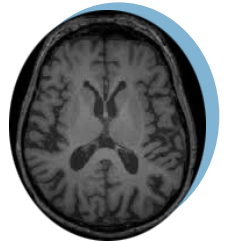
**Test**

L'échantillon d'images utilisé pour fournir une évaluation d'un modèle final. Il est composé de données que la machine n'a jamais rencontré.



**Json**

Ce fichier permet de relier chaque dossier au nom de la classe.



# Réseau de neurones

Pour notre projet, nous avons réalisé un apprentissage supervisé grâce un réseau de neurones artificiels (voir figure 2). Il se compose de trois parties :

- **La couche d'entrée** (Input layer) : c'est la première couche d'un réseau de neurones. C'est par cette couche que vont rentrer les données dont on dispose.
- Des **couches cachées** (Hidden layer): en générale un réseau de neurones contient plusieurs couches cachées inter-connectées les unes avec les autres (la sortie de la première couche cachée liée à l'entrée de la deuxième couche cachée).
- **La couche de sortie** (output) : cette couche va nous fournir le résultat (dans notre cas ce sont des probabilités d'appartenance pour chaque classe).

Chaque neurone comporte un signal (une valeur numérique) et un poids pour chaque liaison avec un autre neurone de la couche précédente. Ces poids et signaux sont attribués au début aléatoirement. Nous réalisons un calcul en faisant la somme pondérée de tous les signaux de la couche précédente auxquels on applique une fonction d'activation pour avoir **le signal de chaque neurone**. Cela va nous donner une valeur en sortie qui va permettre de calculer une **marge d'erreur**. Cette marge d'erreur est une valeur qui correspond à la différence entre la valeur proposée et la valeur attendue. Elle va permettre au réseau d'**ajuster ces poids par une fonction de rétropropagation**. Ce cheminement est **répété au cours de l'entraînement jusqu'à obtenir des résultats satisfaisants**.

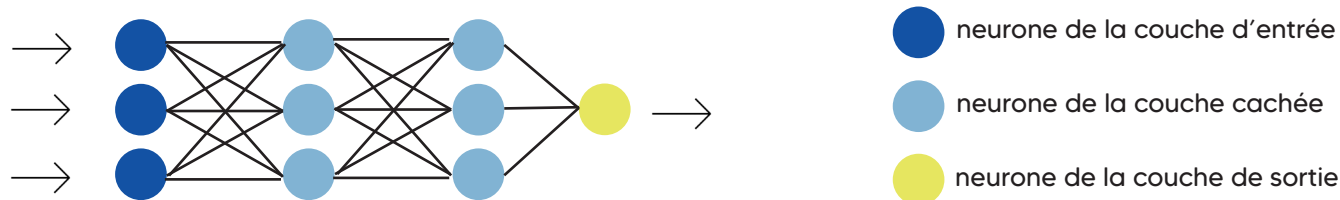


Figure 2 : Réseau de neurones artificiels

Aujourd'hui il existe plusieurs variantes de réseau de neurones avec différentes architectures. Chacune est dédiée à des applications précises : les réseaux de neurones convolutifs (**CNN**), les réseaux de neurones récurrents (**RNN**) et les autos-encodeurs. Dans notre cas il s'agit d'un réseau de neurones convolutifs (construit à partir de **VGG16** ou **Densenet**).

## Les réseaux de neurones Convolutifs (CNN)

Le nom réseau de neurones convolutifs renvoie à un terme mathématique : le **produit de convolution**. Ces réseaux sont utilisés pour le traitement des données de type grille. Ce qui est adapté pour les images 2D qui sont composées d'une grille de pixel. En termes simples, l'idée est d'appliquer des **filtres à l'image d'entrée**.

Un CNN contient en générale quatre différentes couches :

- **Couche convolutionnelle (CONV)** : le but est de repérer les différentes caractéristiques de l'image. Si l'image est en couleur comme dans notre cas, cette couche applique des filtres qui scannent l'image en entrée sur les trois couches (R, G, B). Cela est réalisé par des opérations de convolution paramétrable en ajustant la taille du filtre F, le pas du filtre S, le padding ou bien la taille de sortie du filtre O.
- **Pooling (POOL)** : c'est une couche utilisée en générale après une couche convolutionnelle. C'est une opération de sous-échantillonnage. Les deux types les plus utilisés sont **Max pooling** qui prend la valeur maximum de la surface et l'**Avg pooling** qui prend la moyenne d'une surface. Nous pouvons l'ajuster avec deux paramètres : la taille F et le pas S.
- **ReLU** : c'est une couche de correction des signaux entre les couches. Elle va appliquer une fonction mathématique d'activation à notre sortie. Par exemple la fonction  $\text{ReLU}(x) = \max(0, x)$ .
- **Fully Connected (FC)** : Ces couches se trouvent typiquement à la fin des architectures de CNN. Elles s'appliquent sur une entrée préalablement aplatie. Elle permet d'optimiser l'erreur de notre modèle en passant par une phase d'entraînement. On peut ajouter au CNN des **dropout** afin d'éviter un sur-apprentissage (modèle qui colle trop aux données sur lesquelles on apprend, et qui est trop sensible à la moindres variations).

## L'intelligence artificielle (IA) :

Il s'agit d'algorithmes plus ou moins évolués qui imitent des actions humaines.

## Le machine Learning (ML) :

Application de méthodes statistiques aux algorithmes pour les rendre plus intelligents. Elle repose sur la capacité des algorithmes à recevoir beaucoup de données et à corriger les courbes d'approximation.

## Le deep Learning (DL) :

C'est un sous domaine du machine Learning. Il se construit à partir de **réseaux de neurones artificiels**. Le DL va permettre de pousser l'analyse plus loin et savoir extraire les données soi-même.

## Un réseau de neurones artificiels (Artificial Neural Network) :

C'est un système dont la conception est inspirée du fonctionnement des neurones biologiques, et qui par la suite s'est rapproché des méthodes statistiques.



# Algorithme

C'est grâce aux concepts définies précédemment, nous avons pu construire notre modèle d'apprentissage. Ce modèle repose sur l'algorithme suivant :

**Import des données** et définition des transformations des images avec **Pytorch** (rotation, rognage, normalisation).

**Création du modèle de réseau de neurones (transfert learning)** : le transfert learning permet de transférer la connaissance acquise sur un jeu de données "source" pour mieux traiter un nouveau jeu de données dit "cible". En effet, la construction d'un réseau de neurones ainsi que son entraînement est une tâche qui demande beaucoup de temps et des machines puissantes. Pour cette raison, nous avons utilisé des modèles existants déjà entraîné à reconnaître des objets tels que VGG16 et Densenet. Le principe du transfert learning est de figer les première couches du modèle et d'y ajouter d'autres couches, ou de modifier le classifieur du model (pour Densenet 169). Ainsi, nous pouvons lancer un entraînement uniquement sur les dernières couches du modèle.

**L'entraînement du réseau de neurones** : Dans cette partie nous passons en entrées nos données, sous forme de **Batch**<sup>(1)</sup>. On essaye d'optimiser les poids de notre réseau de neurones. Les poids ont été générés aléatoirement au moment de la création du modèle. Afin d'ajuster ces poids, on réalise une **rétropropagation**<sup>(2)</sup>, qui va dépendre de «l'optimizer» choisi parmi plusieurs variantes. Nous avons utilisé **SGD** car c'est l'un des «optimizer» les plus utilisés (bonne convergence). Nous avons aussi utilisé **ADAM optimizer** qui est une des variantes de SGD la plus évoluée de par sa rapidité et ses bons résultats. Ces deux «optimizer» utilisent la méthode de **descente de gradient**<sup>(3)</sup>. Ce processus est répété nombre d'**epoch**<sup>(4)</sup> fois.

**Sauvegarde des paramètres du réseau de neurones** dans un fichier checkpoint.pth. Les **checkpoints** sont essentiels dans tout notre algorithme. Ce sont des fichiers PTH qui permettent la sauvegarde des paramètres du modèle (**fonction save\_checkpoint(model)**) après un entraînement. Après avoir chargé le checkpoint (**fonction load\_model(file)**) nous pouvons relancer l'algorithme sans refaire l'entraînement afin de faire une prédiction sur une image. Mais nous pouvons aussi continuer un entraînement, au point où il a été arrêté pour la dernière fois. Le fractionnement de l'entraînement en plusieurs fois peut être très utile pour faire des entraînements longs. Nous pouvons ainsi récupérer ce fichier checkpoint et relancer l'entraînement jusqu'à obtenir des résultats satisfaisants grâce à nos deux fonctions **save\_checkpoint(model)** et **load\_model(file)**.

Les paramètres du réseau de neurones étant satisfaisants et sauvegardés dans le checkpoint.pth. Nous pouvons lancer la fonction **test\_accuracy()** qui va permettre d'obtenir une évaluation du modèle (cette fonction nous renvoie la probabilité de trouver une bonne réponse). Ensuite, nous pouvons lancer la **prédiction et afficher les résultats**.

Pour une meilleure représentation de notre algorithme, nous avons réalisé une **hiérarchie de notre projet** (voir annexe 2)

**(1) Le Batch** : c'est un ensemble d'exemples utilisés dans une itération (c'est-à-dire, une mise à jour du gradient). C'est à dire une sous partie de notre dataset contenant batch\_size images.

**(2) La rétropropagation (backpropagation)**: c'est l'algorithme principalement utilisé pour exécuter la descente de gradient sur des réseaux de neurones.

**(3) La descente de gradient** : c'est un algorithme d'optimisation différentiable, destiné à minimiser une fonction (l'erreur de notre réseau). Il est itératif et procède donc par améliorations successives. La descente de gradient prend un paramètre appelé le **taux d'apprentissage (learning rate)** qui représente simplement la taille du pas à chaque itération en se déplaçant vers le minimum (Il faut qu'il soit ni trop grand, ni trop petit). Pour notre projet nous avons fixé **LR=0.001**.

**(4) Epoch**: Une epoch représente le passage de toutes nos données dans le réseau de neurones et l'ajustement des poids par la rétropropagation.

# Évaluation

Dans un premier temps nous avons réalisé **50 tests** pour chaque modèle. Chaque image de test est sélectionnée de façon **aléatoire**, depuis n'importe quelle classe. Nous obtenons ainsi pour chaque test, un pourcentage qui correspond à la « **certitude** » de sa réponse. Nous avons noté à chaque fois le pourcentage qui correspond à la réponse attendue dans un tableau Excel. Ainsi, nous avons pu **réaliser des courbes et comparer nos modèles**. Nous avons également utilisé la fonction **test\_accuracy()** de notre algorithme afin d'obtenir un score de certitude pour chaque base de donnée.

On observe que nos résultats **varient beaucoup en fonction de la base de données**. En effet, on observe (voir figure 3) que pour la base de données mélanome les valeurs sont majoritairement insatisfaisantes. Cette base de données comporte 7 classes différentes et malgré l'augmentation en images de la base de données nous n'avons pas pu obtenir de meilleur résultat. Cependant il semblerait que cette problématique soit complexe. En effet, un **article\*** décrit une étude sur une intelligence artificielle pour une base de donnée dermatologique. On peut voir que leur score de réussite est d'environ 72%. **Le score obtenu par notre réseau de neurone est de 57%**. On peut aussi observer que lorsque le nombre de classes est faible (comme pour brain-tumor) les résultats sont meilleurs.

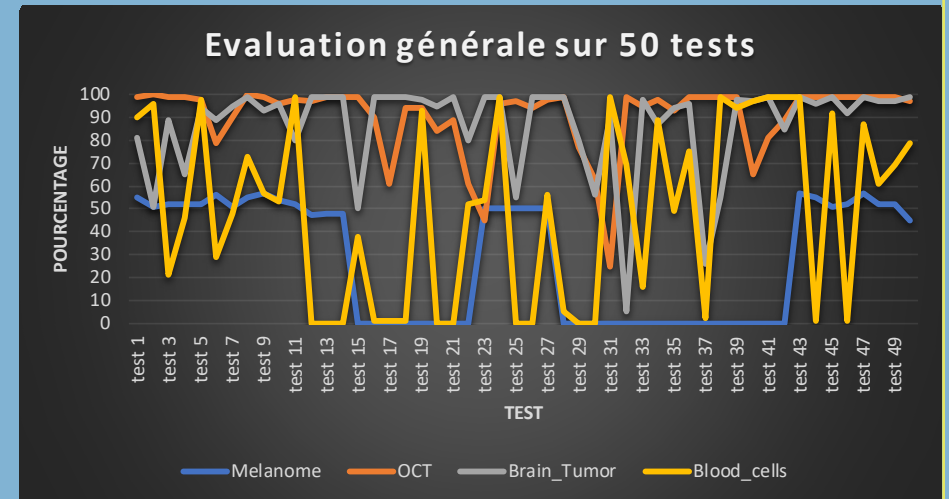


Figure 3 : évaluation générale sur 50 tests

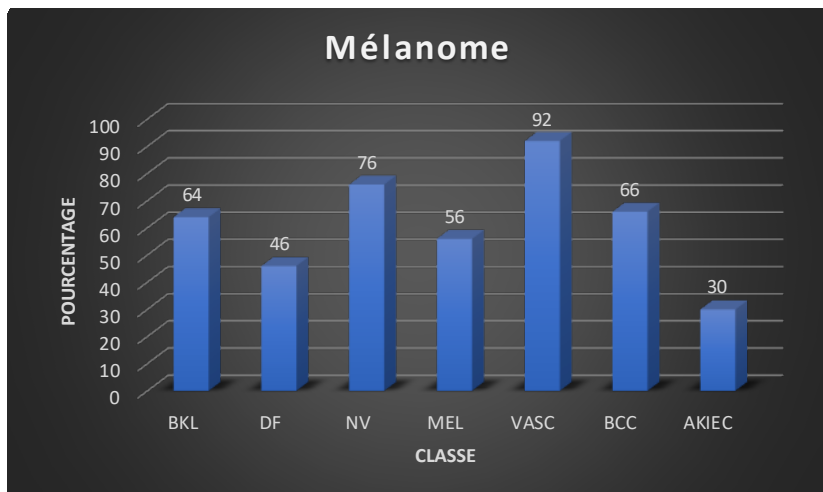


Figure 4 : évaluation de la base de données mélanome

Ensuite, nous avons réalisé un script en Python qui permet **d'évaluer notre modèle sur chaque classe de notre base de données**. Ce script permet de calculer la probabilité de trouver la bonne classe. Pour cela, il lance une analyse sur chacune des images du répertoire de test. Cela nous permet de connaître la probabilité de réussite.

Grâce cette dernière évaluation, nous pouvons remarquer (voir figure 4) l'effet de **l'inhomogénéité des nombres d'images par classe** (par exemple pour la base de donnée mélanomes initial)

\*[https://www.nature.com/articles/nature21056.epdf?author\\_access\\_token=8oxIcYW-f5UNrNpHsUhd2StRgN0jAjWel9jnR3ZoTv0NXpMHRAJy8Qn10ys2O4tuPakXos4U-hQAFZ750CsBNMMsISFHIKinKDMKjShCpHIIYPYUUhNzkn6pSnOCt0Ftf6](https://www.nature.com/articles/nature21056.epdf?author_access_token=8oxIcYW-f5UNrNpHsUhd2StRgN0jAjWel9jnR3ZoTv0NXpMHRAJy8Qn10ys2O4tuPakXos4U-hQAFZ750CsBNMMsISFHIKinKDMKjShCpHIIYPYUUhNzkn6pSnOCt0Ftf6)

# Interface

Nous avons fait en sorte que l'interface (voir figure 5) soit la plus **simple possible** tout en permettant à l'utilisateur d'accéder à toutes les fonctionnalités de l'algorithme. Pour accompagner, l'utilisateur, des bulles d'aides apparaissent au survole des boutons importants. Cette interface a été réalisée avec la librairie graphique de base de Python, **TKinter**. Elle est composée de 3 cadres au sein de la fenêtre, qui représentent les **3 onglets de l'application** (Entraîner, Analyser, Statistiques). Un bouton « quitter » est proposé en bas à droite et un espace en bas au centre permet d'afficher les messages d'alertes. Les cadres sont des instances de la classe « Frame » de TKinter.

**L'onglet «Entraîner»**, est lui-même composé de deux cadres (invisibles pour l'utilisateur), et d'un label qui sert de titre.

- Le premier cadre contient un **label très bref de consigne destiné à l'utilisateur**, une liste déroulante (classe « Combobox »), une zone de liste (classe « ListBox »), ainsi qu'un bouton « importer ».

- Le second cadre possède également un bref label, suivi de 3 boutons et un indicateur du **nombre d'époch sélectionné**. Ce sont trois instances de la classe « Button » et un label qui est modifié en fonction du nombre d'époch choisit par les boutons + et -. Le dernier bouton « Entraîner » appelle le code image\_classifier.py avec les paramètres suivants : le nom de la base de données, le fichier json sélectionné précédemment grâce à la liste déroulante, le nombre d'époch choisi, le booléen train = true qui indique que l'on veut **faire un entraînement** et le nom du modèle utilisé par la base de données choisit (VGG16 ou DENSENET).

**L'onglet «Analyser»**, est le cœur de notre application et est donc situé au centre. Il est constitué de deux labels, une liste déroulante et une zone de texte qui affiche le **chemin de l'image choisie**. A droite de cette zone de texte est placée un bouton « Parcourir » pour choisir une image. Ce bouton appelle la fonction **fileDialog()** qui ouvre la fenêtre de dialogue, insère le chemin de l'image dans la zone de texte et affiche un **aperçu de l'image en dessous**. Le dernier bouton « Analyser » appelle à nouveau le code du «classifier» mais avec des paramètres qui lancent uniquement **l'analyse d'une image donnée**.

**Enfin, le dernier onglet, «Statistiques»** affiche quelques informations sur les résultats obtenus avec nos entraînements. Nous avons la précision de nos tests sur **chacune des bases de données**. Une liste déroulante permet de sélectionner une base de données en particulier pour avoir un **graphique des résultats dédié aux classes qu'elle contient**.

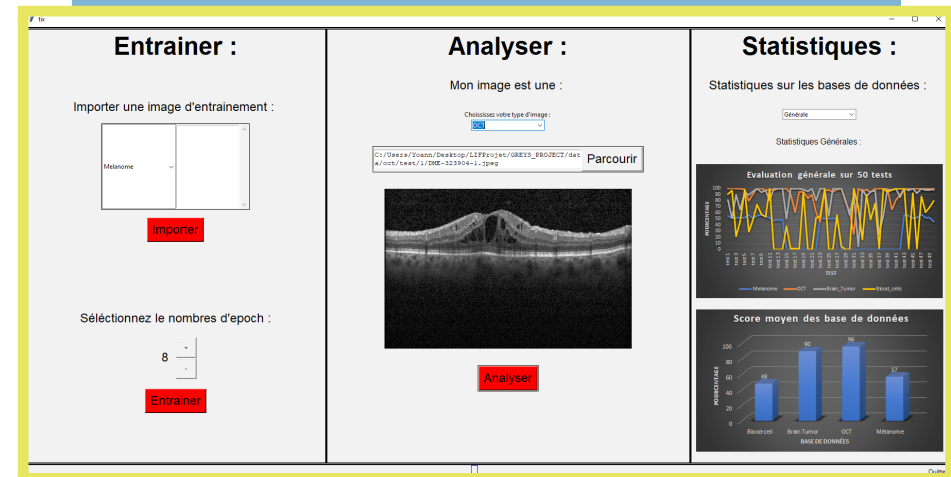


Figure 5 : Capture d'écran de l'interface



# Conclusion / Perspectives

Ce projet nous a permis de construire un modèle de réseau de neurone que l'on a pu tester sur différentes bases de données. Après avoir effectué l'évaluation, nous avons pu voir que le nombre de classes et d'images dans la base de données influe sur les performances du modèle.

Une meilleure utilité de l'interface serait d'en faire un véritable outil d'apprentissage pour les étudiants en médecine par exemple. Des quizz seraient proposés sur toutes les bases de données sous forme de questionnaire à choix multiple. Cela leur permettrait de s'entraîner à l'établissement d'un diagnostic sur des images réelles.

De plus, quelques fonctionnalités pourraient être ajoutées, telle que la barre de progression qui permet de visualiser l'avancement de l'entraînement, ou encore pouvoir effectuer un «Drag'n'drop» pour importer des images.

Une autre idée serait de passer à un apprentissage non supervisé. Cela pourrait se faire par l'intermédiaire d'un auto-encodeur qui va réduire les images de chaque classe dans des ensembles réduits.

Pour l'évaluation de nos résultats nous avons souhaité tracer une courbe ROC (fonction d'efficacité) pour chaque base de données en utilisant la librairie de `sklearn.metrics` et `pyplot` mais faute de temps, cela n'a pas été réalisé.

# Annexe

## Annexe 1:

**Melanome** : Il s'agit d'un ensemble d'images dermatoscopiques multi-sources de lésions cutanées pigmentées courantes. **Akiec:** kératoses actiniques (kératoses solaires) et le carcinome intraépithélial (maladie de Bowen). Les deux néoplasmes présentent généralement une desquamation de la surface et sont généralement dépourvus de pigment. **Bcc :** Le carcinome basocellulaire est une variante courante du cancer épithélial de la peau. **Bkl :** La "kératose bénigne" est une classe générique. **Df:** Le dermatofibrome est une lésion cutanée bénigne considérée comme une prolifération bénigne ou une réaction inflammatoire à un traumatisme minimal. **Nv:** Les naevus mélanocytaires sont des néoplasmes bénins des mélanocytes. Contrairement aux mélanomes, ils sont généralement symétriques en ce qui concerne la distribution de la couleur et de la structure. **Mel:** Le mélanome est un néoplasme malin dérivé des mélanocytes qui peut apparaître sous différentes variantes. **Vasc:** Ce sont des lésions vasculaires de la peau. Les hémorragies sont également incluses dans cette catégorie.

**Tumeur cerveau :** Il s'agit d'IRM du cerveau en coupe axiale.

**Oui:** Il y a une tumeur au cerveau

**Non:** Il n'y a pas de tumeur au cerveau

**Cellule sanguine :** Le diagnostic des maladies à base de sang implique souvent l'identification et la caractérisation des échantillons de sang. Nous avons ici des observations à partir de microscope optique d'échantillon sanguin.

**Eosinophile:** Les éosinophiles sont un type de globules blancs qui jouent un rôle important dans la réponse de l'organisme aux réactions allergiques, à l'asthme et aux infections parasitaires. Les éosinophiles représentent moins de 7 % des globules blancs circulants (100 à 500 éosinophiles par microlitre de sang).

**Monocyte:** Elles font partie des leucocytes, aussi appelées globules blancs, en charge des défenses immunitaires de l'organisme.

**Lymphocyte:** Les lymphocytes font partie de la famille des leucocytes (globules blancs). Ils prennent l'apparence de petites cellules rondes et possèdent un noyau. On les retrouve dans le sang, la moelle osseuse (où ils sont produits) et les tissus lymphoïdes (rate, ganglions lymphatiques).

**Neutrophile:** Des cellules sanguines appartenant à la lignée blanche. En effet, ce sont des globules blancs (leucocytes) qui ont un rôle majeur dans le système immunitaire.

**OCT :** Ce sont des images de rétine réalisées à partir de tomographie par cohérence optique. C'est un outil qui permet de mesurer de façon complètement inoffensif l'œil, avec précision.

**DME (Diabetic Macular Edema):** C'est une complication du diabète causée par l'accumulation de liquide dans la macula, la partie centrale de l'œil, qui provoque un gonflement de la macula.

**CNV (Neovascularisation choroïdienne):**

C'est la création de nouveaux vaisseaux sanguins dans la couche choroïde de l'œil.

**DRUSEN:** Minuscules accumulations jaunes ou blanches de matière extracellulaire qui s'accumulent entre la membrane de Bruch et l'épithélium pigmentaire rétinien de l'œil.

**NORMAL:** C'est l'état normal de la rétine

## Annexe 2 :

