

ML Sound - Programme Deep Learning d'identification de la faune sauvage

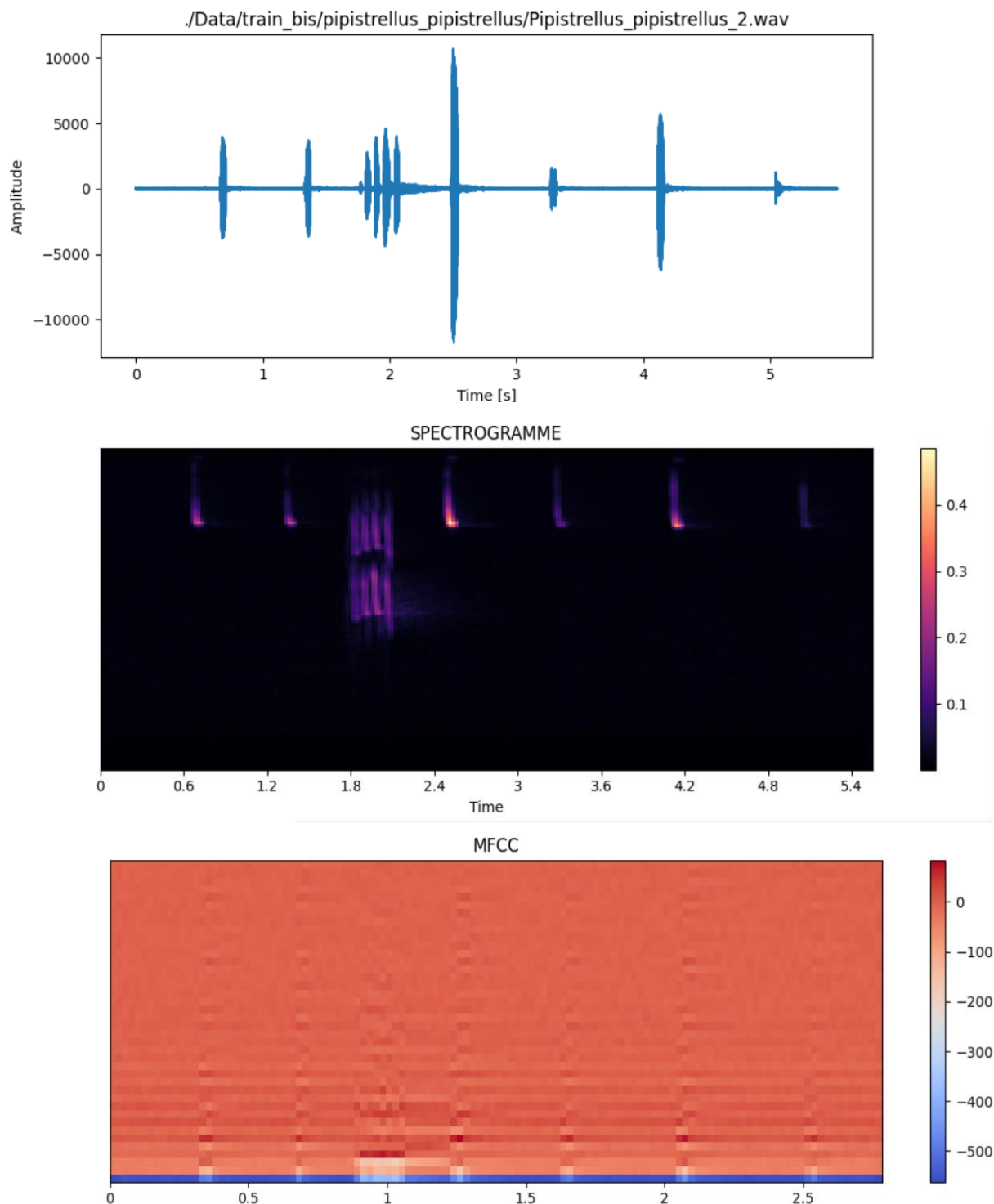


Table des matières

Introduction	3
I - Prise en main du sujet	3
1. Analyse des articles	4
2. Etablissement d'une stratégie.....	7
II - Prérequis	8
1. Python.....	8
2. Manipulation de fichiers audios	10
3. Machine Learning	12
III - Réalisation	15
1. Recherche de données.....	15
2. Préparation du son	18
3. Création du modèle.....	24
4. Interface utilisateur	28
5. Algorithmes annexes.....	36
IV - Bilan, gestion de projet	38
1. Fonctionnement de l'application	38
2. Gestion de projet	40
3. Bilan	44
3. Perspectives d'amélioration.....	46
Conclusion	47
Annexes	49
Table des illustrations	52
Sitographie	56
Résumé / Abstract	58

Introduction

Ce rapport intervient dans le cadre de la réalisation d'un projet de troisième année de licence informatique. Le but de ce rapport est d'expliquer la démarche mise en place, tant du point de vue théorique qu'expérimental. Le sujet est le suivant : mettre en pratique les approches proposées dans la littérature scientifique en proposant une interface conviviale. Cette littérature présente des approches pour identifier des espèces à partir de données collectées par l'homme. Ces données ont pour vocation à être analysés par une machine dans le but d'obtenir un résultat. Afin d'aborder le sujet, dans un premier temps, il est nécessaire de lire et d'analyser les articles mis à disposition dans le sujet. Ensuite, il faut établir une stratégie afin de répondre convenablement au sujet ([section 1](#)). Avant de commencer à programmer il est également nécessaire d'acquérir des prérequis et les compétences qui y sont associées ([section 2](#)). Une fois ces compétences et connaissances acquises, la réalisation du projet a commencé et s'est déroulée sur plusieurs mois. Ce rapport présente le déroulement du projet et son évolution au cours des mois ([section 3](#)). Finalement, pour qu'un projet progresse efficacement, il est essentiel de posséder une bonne organisation ainsi que des connaissances en gestion de projet ([section 4](#)).

I - Prise en main du sujet

Cette problématique ne date pas d'hier, en 1992, [le canton de Genève en Suisse, publiait un guide pour la protection des chauves-souris lors de la rénovation des bâtiments \[1\]](#). Ce guide s'adresse aux maîtres d'œuvre et propriétaires de bâtiments. Il explique l'importance des chauves-souris dans la biodiversité, en effet, grande consommatrice d'insectes, elle en ingère environ 15 kg par an. Sans elles cet équilibre serait bouleversé et c'est pourquoi elles sont protégées depuis 1966 en Suisse. En France, il est interdit de tuer ou de perturber une chauve-souris depuis 1976. Il est donc primordial de connaître la population à un endroit précis avant toute construction, rénovation ou modification d'un ouvrage. Ce programme a donc pour but de répondre à une problématique naturaliste

moderne. L'impacte de l'humain sur la biodiversité. Ses utilisateurs seront donc principalement des naturalistes réalisant des recensements d'espèces et d'individus.

1. Analyse des articles

Pour résumer le sujet du projet en de brefs lignes, il s'agit ici de réaliser un outil simple à prendre en main pour un naturaliste. Cet outil doit donc combiner un algorithme complexe sous une interface intuitive. Deux articles ont été proposés pour servir de base au projet. Le projet a donc débuté par une analyse de ces articles.

Ces deux articles traitent de l'application du Deep Learning à des problématiques naturalistes. Chacun de ces articles possède sa propre approche. En effet, le premier, [Bat Détective \[2\]](#), utilise le Deep Learning afin d'identifier et de localiser les cris de chauves-souris dans un enregistrement audio. [Le second \[3\]](#), utilise également le Deep Learning mais cette fois-ci, afin de déterminer l'espèce d'un oiseau à partir d'une image. Dans un premier temps il sera important d'aborder chaque article pour ensuite les comparer, et mettre ces deux visions et utilisations en parallèle.

[Bat Détective \[2\]](#), publié en mars 2018, se veut novateur dans le milieu des détecteurs de cris de chauves-souris. En effet, son but est de mesurer l'impact de la pollution et des changements environnementaux sur la biodiversité, et plus particulièrement l'évolution des populations de chauves-souris. Pour ce faire, il faut être capable d'analyser des enregistrements de chauves-souris. La plupart de ces enregistrements étant réalisés au bord des routes, ils comportent de nombreux bruits parasites. La problématique contemporaine concerne le coût des équipements. En effet, comme le disent les naturalistes, plus la méthode utilisée est efficace, plus son coût est élevé. Le fait de développer un programme accessible, utilisable et modifiable par tous est donc essentiel pour protéger la biodiversité. Afin d'illustrer cette problématique, voici un exemple concret. Cet article scientifique explique ensuite la démarche réalisée pour développer le programme. Tout d'abord, une description détaillée de la manipulation effectuée sur les fichiers audios. Tel que la

transformation en spectrogramme et son analyse en détectant les pics. Ensuite, une très grande partie est consacrée au jeu de données utilisé. En effet, trouver une base de données est très complexe. Ce point sera abordé en détail plus tard. Finalement, les résultats obtenus sont présentés et comme le montre le graphique ci-contre, [Bat Détective](#) [2]

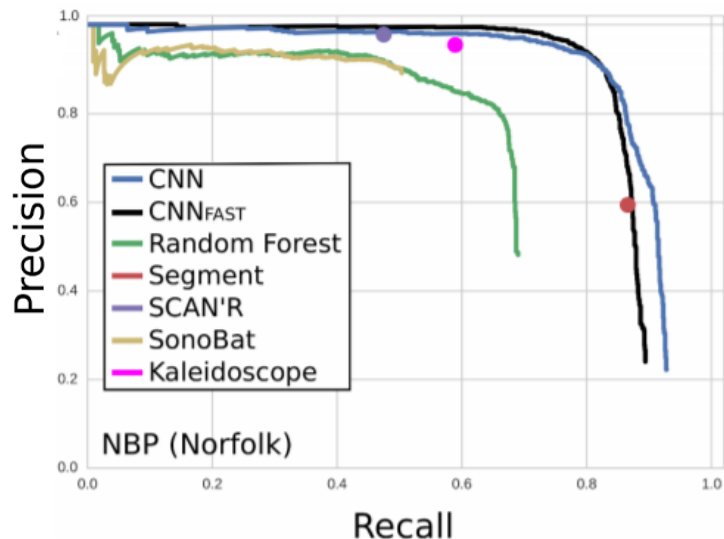


Figure 1 - courbe précision/recall sur la recherche de cris de chauves-souris

surpasse les autres logiciels actuellement présents sur le marché. Certains étant même payant tel que Sonobat, qui à l'heure actuelle a un coût situé entre \$384 et \$1536. Le graphique présente le ratio entre la précision et le recall. La précision représente le pourcentage avec lequel le programme est capable de prédire la vérité. Le recall quant à lui représente la proportion entre le nombre de vrai positif et de faux positif. Le schéma ci-dessous explique les calculs réalisés pour obtenir ces deux valeurs.

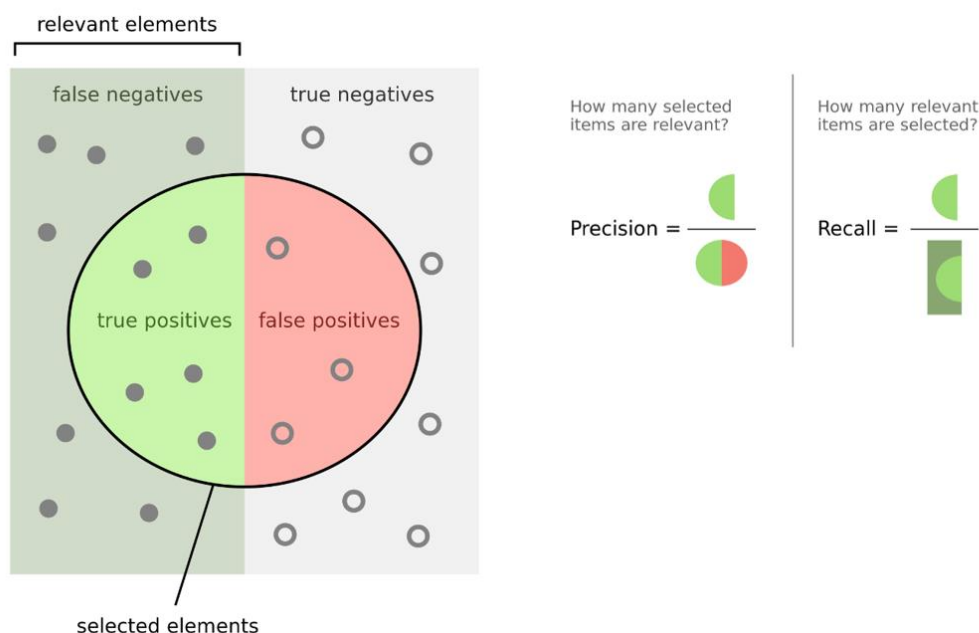


Figure 2 - Schéma explicatif des calculs de précision et de recall

[Le second article sur l'identification des espèces d'oiseaux grâce au Deep Learning \[3\]](#)

s'appuie lui sur des images. Cela comporte donc une approche réellement différente sur la préparation et la recherche de données. En effet, pour les oiseaux la problématique est différente. La protection n'est pas la même en fonction des espèces. En effet, certaines espèces tel que les Cigognes par exemple sont protégées et très surveillées. D'autres tel que le Corbeau Freux, la Corneille noire ou encore le Pigeon ramier qui sont catégorisés comme espèces susceptibles d'occasionner des dégâts. On constate ici deux poids deux mesures. Il est donc essentiel de pouvoir classifier les espèces et non pas simplement différencier un oiseau d'un autre groupe d'espèces animales. De plus cette différenciation est vraiment triviale à réaliser, notamment lorsqu'on la compare avec la localisation de cris de chauves-souris dans des enregistrements audios. Ici, l'approche de [l'article \[3\]](#) est différente, l'explication est plus axée sur l'explication du développement de l'algorithme de Deep learning. En effet, le jeu de données étant présent sur internet, seul son nom est présent. Ensuite, le jeu de données étant préparé pour être utilisé, la préparation des données a déjà été faite en amont. La seule partie qu'il reste donc à développer concerne l'implémentation du Deep Learning. Ils décrivent son utilisation avec beaucoup de termes scientifiques complexes. Les résultats sont ensuite présentés comme surpassant une nouvelle fois les logiciels traditionnels.

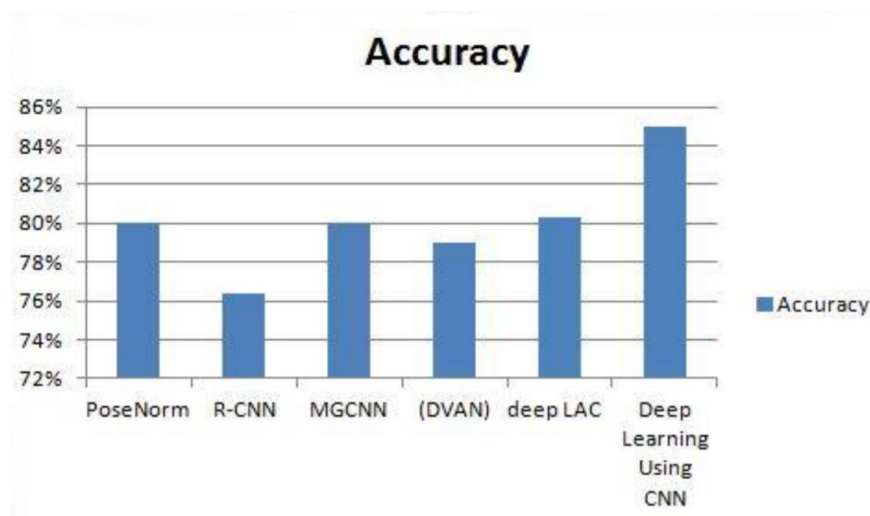


Figure 3 - Diagramme en bâton de la précision de différents programmes

Ces deux articles ont donc des applications et des buts très différents. Ils se rejoignent cependant sur l'utilisation du Deep Learning et de bibliothèques modernes afin d'arriver à leur but. La question qui se pose désormais est, quelle stratégie va être mise en place et quel sera le but du programme ? Sur quel article le projet va t'il principalement s'appuyer ?

2. Etablissement d'une stratégie

Une fois les documents lus, compris et analysés, il est important de savoir quelle direction donner au projet. Les premières semaines ont donc permis, au travers de réunions, de comprendre les attentes du tuteur de projet. À la suite de ces explications, une stratégie a pu être établie. Afin de réaliser ce projet, des réunions hebdomadaires ont été mises en place pour permettre un suivi régulier de l'avancement du projet. Au début, il a fallu s'exercer en réalisant un premier mini-projet. Ce mini-projet consiste à implémenter une interface graphique pour le programme [Bat Détective \[2\]](#). Cela permet d'acquérir de bonnes bases en python et de comprendre les différents enjeux. Ce mini-projet sera abordé plus en détails dans la partie prérequis. Une fois les outils pris en main, le sujet paraît plus clair. Il est donc temps d'établir une première liste d'objectifs. En effet, la méthode Agile étant la méthode sélectionnée pour ce projet, il n'y a pas de réel cahier des charges, il est préférable de travailler autour d'une idée commune et la faire évoluer.

La première version du cahier des charges était établie. Le sujet du projet porte sur les chauves-souris, en utilisant des enregistrements audios. Le but n'étant pas de réaliser une copie du projet [Bat Détective \[2\]](#) qui détecte tous les enregistrements de chauves-souris. Il s'agit là de réaliser un outil de classification permettant d'identifier une espèce de chauve-souris à partir d'un enregistrement audio. Cela à partir d'un modèle réalisé avec une bibliothèque de deep learning en python, Tensorflow. Ce modèle est entraîné avec des données collectées sur internet. Ce programme devra comporter, comme il l'était initialement demandé, une interface graphique intuitive et utilisable par tous. L'objectif est

donc de réaliser un algorithme étant capable d'égaliser, voire de surpasser les logiciels commerciaux.

II - Prérequis

1. Python

Afin de réaliser ce projet, le choix du langage de programmation s'est porté sur Python car il s'agit du langage privilégié par bon nombre de bibliothèques et d'articles scientifiques sur le Machine Learning. Notamment par [Bat Détective \[2\]](#), programme proposé par le tuteur de projet.

Tout d'abord, il a fallu choisir un outil afin de pouvoir coder. En ce qui concerne Python, les développeurs utilisent des environnements virtuels. Ces derniers comportent toutes les bibliothèques nécessaires au programme sans les installer directement sur la machine, chaque environnement possède ses propres bibliothèques. Ils ont également besoin qu'un interpréteur Python soit configuré. Le logiciel Pycharm semble être le logiciel le plus adapté au développement de ce projet. De plus, durant le module "Outils pour la programmation", le logiciel utilisé Idea développé par Jet Brain. Ce dernier étant très bien conçu, la décision a été prise de choisir le logiciel développé par Jet Brain pour programmer en python. En ce qui concerne l'interpréteur, le programme sera développé avec la dernière version disponible à l'heure actuelle, à savoir Python 3.8 avec Anaconda afin de simplifier la gestion des packages. La liste des bibliothèques utilisées est disponible en annexe.

Afin de prendre en main Python, un entraînement a été nécessaire. La réalisation de courts programmes a donc été réalisée afin de prendre en main le langage. Afin de pouvoir commencer ce projet, il a tout d'abord été important de comprendre et de faire fonctionner le programme [Bat Détective \[2\]](#). Ce programme est le sujet d'un des deux articles à partir desquelles le projet doit s'inspirer. Cependant ce dernier est réalisé en python 2.7 et donc

utilise des packages qui ne sont, pour certains, plus supportés ou plus disponibles. Un environnement de travail a donc été configuré afin de satisfaire les exigences du programme. Après plusieurs semaines à essayer de trouver des solutions et comprendre le code de ce programme, cela s'est soldé par l'obtention d'un programme qui fonctionne et qui permet de localiser les cris de chauves-souris dans des fichiers .wav. Ce mini-projet consiste à créer une fenêtre à l'aide de la bibliothèque TKinter, tout en faisant fonctionner l'algorithme de [Bat Détective \[2\]](#). Des classes ont été mises en place pour représenter les différentes parties de l'application. Ainsi que des boutons afin de pouvoir sélectionner un fichier .wav sur l'ordinateur de l'utilisateur, puis de détecter et afficher les time codes et la probabilité.

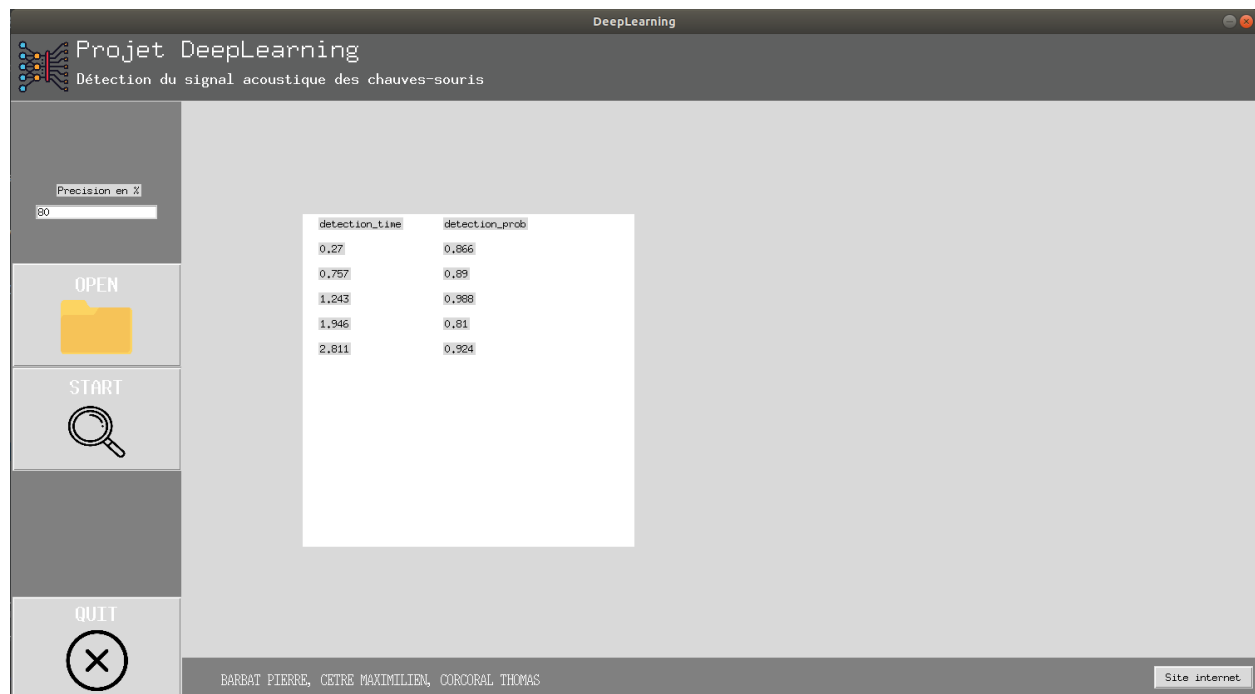


Figure 4 - Première version de UI du projet, implémentation de l'algorithme de Bat Détective

Il est donc essentiel de posséder des connaissances en programmation et en python afin de pouvoir comprendre les explications techniques et de pouvoir réaliser un projet portant sur le même sujet.

2. Manipulation de fichiers audios

Pour utiliser des données dans des programmes informatiques, il est nécessaire de les transformer et de les uniformiser.

Tout d'abord, afin de mieux comprendre le sujet, il faut lire beaucoup d'articles sur la manipulation de fichiers audio et plus particulièrement de fichiers .wav afin de transformer ces derniers en données. Pour utiliser le son, le programme retourne un tableau de valeurs (array). Cela signifie que, à partir d'un fichier audio .wav, grâce à des algorithmes, il a été possible de récupérer les différentes valeurs qui correspondent à l'oscillogramme de l'enregistrement. Cependant plusieurs problèmes se posent. [Les exemples cités par Mike Smales\[4\]](#) permettront d'illustrer ces problèmes. Dans son article sur la classification de sons urbains à l'aide de la bibliothèque [UrbanSound8K \[5\]](#), il y décrit les problèmes rencontrés. Cette base de données comporte 8732 sons, le premier problème qui se pose est l'utilisation de sons enregistrés en stéréo donc avec deux canaux.

```
# num of channels
print(audiodf.num_channels.value_counts(normalize=True))

2    0.915369
1    0.084631
```

Figure 5 - Proportion entre les enregistrements mono et stéréo dans le dataset UrbanSound8K

Sur cette image on peut voir que environ 91,5% des enregistrements utilisent deux canaux (stéréo) et 8,5% un seul canal (mono). Il faut donc être capable de pouvoir utiliser les deux types d'enregistrement de la même façon sans intervention humaine.

```
# sample rates

print(audiodf.sample_rate.value_counts(normalize=True))
```

44100	0.614979
48000	0.286532
96000	0.069858
24000	0.009391
16000	0.005153
22050	0.005039
11025	0.004466
192000	0.001947
8000	0.001374
11024	0.000802
32000	0.000458

Figure 6 - Proportion entre les différents sample rates présents dans le dataset UrbanSound8K

Ici, le but est de montrer le sample rate¹ de chaque enregistrement afin d'obtenir des statistiques. On peut donc voir qu'une grande disparité existe même si certains sont très rares. Par exemple on retrouvera seulement 0,045% d'enregistrement ayant un sample rate à 32000. Cependant, il ne faudra pas ignorer les sample rate minoritaire sous peine d'obtenir des erreurs lors de l'utilisation des données. En effet, par la suite, on ne pourra pas contrôler les entrées des utilisateurs. Il faut donc anticiper ces différences.

```
# bit depth

print(audiodf.bit_depth.value_counts(normalize=True))
```

16	0.659414
24	0.315277
32	0.019354
8	0.004924
4	0.001031

Figure 7 - Proportion entre les différents bit depth présents dans le dataset UrbanSound8K

¹ Fréquence (ou taux) d'échantillonnage.

Cette image représente les disparités au niveau du bit depth². On constate la même chose que les deux points précédents, la majorité utilisent des niveaux de qualité de 16 ou 24 bits. Ce qui est logique car ce sont des enregistrements possédant une très bonne qualité.

Pour résumer, il faudra prendre en compte trois points qui diffèrent en fonction des enregistrements :

- Le nombre de canaux d'enregistrement
- Le sample rate
- Le bit depth

3. Machine Learning

Afin de pouvoir créer un programme efficace de reconnaissance et de classification, la décision a été prise d'utiliser une bibliothèque python de machine learning. Cependant, il fallait au préalable comprendre son fonctionnement et être capable de pouvoir la mettre en place.

Dans le but de comprendre et de pouvoir par la suite mettre en place un tel programme, il faut effectuer des recherches sur le sujet. La première recherche et surement la plus naturelle fut se renseigner sur l'existence d'un programme réalisant une classification à partir de différents fichiers audios. Plusieurs projets correspondant aux attentes ont été trouvés. Et effectivement, ils utilisent tous la bibliothèque Tensorflow. La bibliothèque trouvée, il était temps de s'attaquer au cœur du projet, les réseaux de neurones. En effet, la bibliothèque permet d'implémenter des réseaux de neurones à l'aide des fonctions disponibles.

Lorsque l'on se renseigne sur les réseaux de neurones artificiels, on découvre un grand nombre d'articles scientifiques et grand public évoquant le sujet. Tout d'abord qu'est-ce qu'un neurone ? Il s'agit d'un élément d'un réseau qui comporte un poids qui va varier à

² Résolution : qualité de l'échantillonnage

chaque entraînement du modèle. Ce poids va permettre de faire varier la valeur de sortie du neurone.

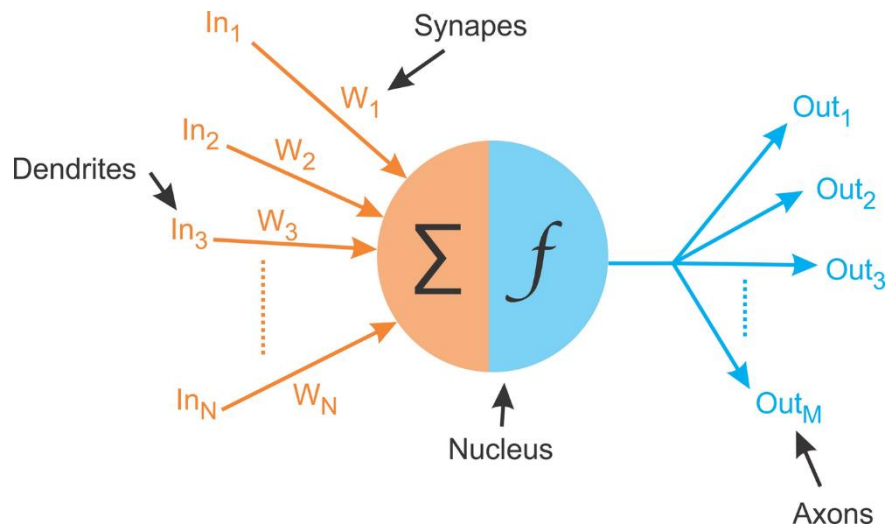


Figure 8 – Représentation d'un modèle informatique de neurone, il existe un noyau, des dendrites (entrées), un poids associé à chaque entrée et des axones (sorties des neurones).

Un neurone va appliquer une fonction qui peut être une sigmoïde, un Rectifier, linéaire, softmax (souvent utilisé pour la dernière couche d'un réseau de neurone ayant pour but de réaliser une classification) ou encore une fonction hyperbolique. Afin de mieux comprendre son fonctionnement, il existe un très bon [site \[6\] proposé par Tensorflow](#) qui permet de réaliser des simulations de réseaux de neurones sur différentes répartitions de données. Avant de présenter des exemples à partir de ce site, il est important de comprendre la relation entre tous les éléments présentés ci-dessus (entrées, poids, fonction d'activation et sortie). Pour ce faire, voici un schéma récapitulatif très simple.

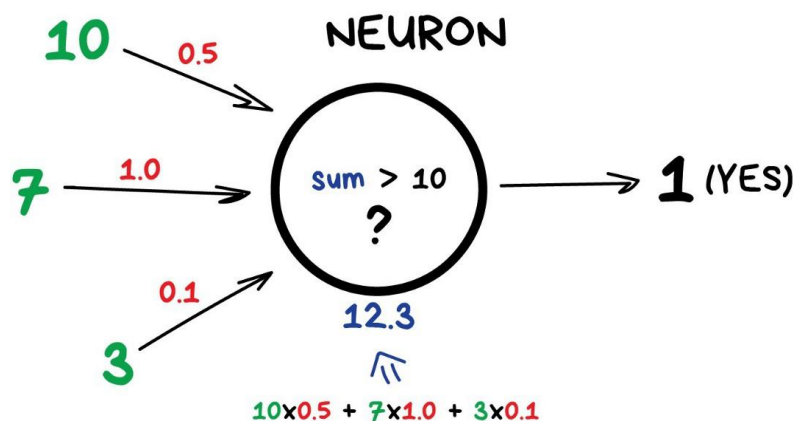


Figure 9 - Exemple très simple du fonctionnement d'un neurone

Si l'on souhaite classer deux populations bien distinctes le problème est très simple, il suffit de deux neurones en entrée, une couche hidden à un neurone. Les deux premiers neurones de la couche d'entrée réalisent le calcul pour la position horizontale et verticale et le neurone de sortie combine ses informations pour faire une sortie oblique. Il s'agit d'un problème linéaire. En effet la solution est une simple séparation par une surface plate.

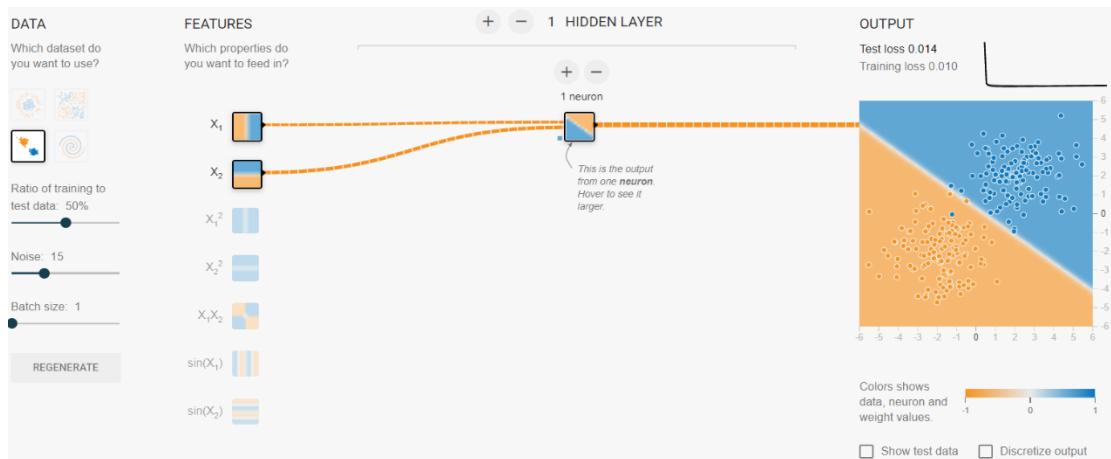


Figure 10 - Capture d'écran d'une classification simple

Étudions maintenant un cas où l'organisation des données est un peu plus chaotique. En effet, les données sont organisées en spirale, il faut donc beaucoup plus de couches de calcul avec également plus de neurones sur chaque couche. Chaque neurone

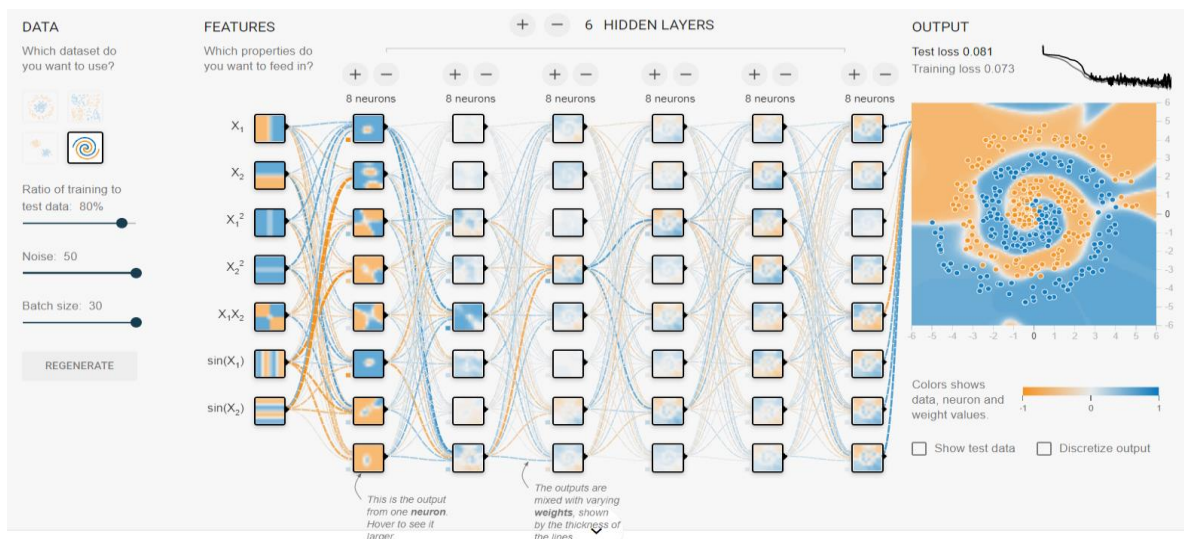


Figure 11 - Capture d'écran d'une classification complexe

Dans le machine learning, chaque type de réseau neuronal artificiel est adapté à certaines tâches. Il existe deux types de réseaux de neurones : les réseaux de neurones convolutifs (CNN) et les réseaux de neurones récurrents (RNN). La principale différence entre un CNN et un RNN est la capacité à traiter des informations temporelles ou des données qui arrivent en séquences, comme une phrase par exemple. En effet, un CNN va traiter les informations telles qu'elles sont sans tenir compte des données précédentes contrairement au RNN. Il va également être capable de traiter des données en 2D. Cela est très pratique et bien plus efficace pour classifier des images. De ce fait, un RNN sera principalement utilisé pour du traitement de données en temps réel lorsque les informations précédentes doivent jouer un rôle dans la prédiction. Par exemple, pour un logiciel de correction orthographique, le principe même de ce programme est de garder en mémoire le mot courant voir même la phrase afin d'éviter les fautes de syntaxe et de conjugaison. Cependant, un son d'oiseau ou de chauves-souris, - donnée concernée ici - en fonction de sa représentation sera plus facilement traité par un CNN car ce dernier est analysé d'un seul bloc.

III - Réalisation

Les données présentées ci-dessous ont été obtenues en exécutant les programmes avec la configuration suivante : Intel I7 9750H 2,6 GHz, 16 Go DDR4 2666 MHz, GeForce RTX 2060 6Go.

1. Recherche de données

Afin de pouvoir tester le programme sur des données plus concrètes, la décision a été prise d'utiliser des sons de chauves-souris. Cependant de nombreux problèmes ont été rencontrés. En effet, l'enregistrement de cris de chauves-souris est déjà un défi. Les chauves-souris émettent des ultrasons qui ne sont donc pas perceptibles par un micro standard. Il est donc nécessaire de posséder un matériel spécifique. Ce matériel est coûteux et donc rare sont les détenteurs d'un tel matériel. Un bon enregistreur capable d'être posé en extérieur peut facilement coûter plus de 1000 euros. Il s'agit principalement de biologistes et de laboratoires de recherche. De plus, ces technologies ont grandement progressé ces

dernières années. Les données sont donc rares et bien souvent réservées aux scientifiques et aux chercheurs appartenant aux laboratoires ayant créé les bases de données. Il est notamment de préciser qu'il existe tout de même du matériel plus accessible. Cependant, la qualité des enregistrements de ces derniers est de moins bonne qualité.

La première base de données utilisée fut celle du site [batcalls \[7\]](#), contenant les sons d'environ une dizaine de chauves-souris européennes. Néanmoins, du fait de son manque d'échantillons, son utilisation n'était pas appropriée. Ensuite, un accès à [une autre bibliothèque de sons \[8\]](#) a été tenté. Mais cette base de données était malheureusement bloquée par un système de membre payant. La troisième tentative fut une étude mentionnée sur [le site du musée des sciences naturelles de Berlin \[9\]](#), mais le lien vers la base de données mentionnée n'étant pas fonctionnel, cette piste n'a donc également pas été retenue. N'ayant que très peu de sons à disposition, Thomas eut l'idée de recourir à une bibliothèque nommée [UrbanSounds8K \[5\]](#), contenant des bruits provenant d'environnement urbains, afin de pouvoir tester le modèle avec des échantillons plus nombreux, même étant hors sujet.

Finalement, après une réunion, la décision a été prise de se tourner vers des bibliothèques de sons d'oiseaux. Etant donné que le mécanisme de détection est similaire, et que les bases de données de sons d'oiseaux sont plus accessibles, et fréquentes que celles de sons de chauves-souris. La première piste fut celle de [Kaggle \[10\]](#). Ce site organise des compétitions de créations d'algorithmes de reconnaissance, avec du Deep Learning. Il possède donc des bibliothèques de sons libres de droits.

Les dataset de [Kaggle \[10\]](#) n'étant pas suffisamment fournis, les recherches se sont poursuivies. Afin de trouver des données sur des oiseaux, il est important de les chercher au bon endroit. Pour ce faire, les recherches sont désormais portées sur des sites spécialisés en ornithologie. Cela a entraîné la découverte d'un site nommé [xeno-canto \[11\]](#) qui permet de mettre en relation des ornithologues à travers le monde et donc constitue une base de données gigantesque.

enregistrements de plusieurs minutes voir même parfois de plus d'une heure. L'option renseignée est donc la suivante len_{It}:30.

Lorsque le projet arrivait à son terme, M. Canon a réussi, par le biais de contacts dans le milieu naturaliste. A apporter des enregistrements de cris de chauves-souris accompagnés de fichiers recensant les différentes espèces entendues dans les enregistrements. Ces données ont permis de clore le travail effectué en amont avec différentes données ne correspondant pas réellement aux attentes fixées initialement. En effet, l'impossibilité de trouver des données accessibles librement a grandement fait évoluer le cahier des charges. Les enregistrements ayant été réalisés non loin de Besançon (25 Doubs). La majorité des enregistrements correspondent à des cris de Pipistrelle. De ce fait, la répartition entre les espèces n'est pas équitable. Il a donc été choisi de séparer les enregistrements en deux groupes équitables. Le premier contenant les enregistrements de Pipistrelles et le second ceux des autres espèces. De cette manière, les données pourront être utilisées par le programme.

2. Préparation du son

Il a été décrit dans les prérequis que la manipulation de l'audio allait entraîner de nombreux conflits de forme. Afin de résoudre tous ces problèmes, plusieurs recherches ont menés à la découverte de la bibliothèque librosa qui permet de pallier tous ces derniers :

- Le sample rate est convertit à 22050 Hz
- Le bit depth varie entre -1 et 1
- Aplatit l'audio en passant les audio enregistrés en stéréo en mono

Cependant la taille du résultat obtenu reste dépendante de la durée de l'enregistrement initial. Pour pallier ce problème, une des données est fixe. Il s'agit du n-mels. Afin de pouvoir uniformiser les données, la décision a été prise de faire la moyenne de données à l'aide de la fonction mean de la bibliothèque numpy. Cela va permettre de transformer un tableau 2D

de dimensions [n_mels,?] en un tableau 1D de taille n_mels. Désormais plusieurs choix se profilent.

Afin de pouvoir obtenir une identification efficace, il est essentiel d'utiliser la meilleure représentation possible. Deux solutions sont alors envisageables (cf. image de couverture). Ces dernières sont :

1. Utilisation du mel spectrogramme
2. Utilisation du mfcc

Le projet ayant avancé depuis la première méthode de préparation des enregistrements, il faut désormais être capable d'utiliser efficacement les nouvelles données trouvées. Tout d'abord, il faut aborder l'utilisation de fichiers .mp3 car, en effet jusqu'à maintenant il n'était question que de fichiers .wav.

Comme dit précédemment, notre programme ne prenait pas encore en charge les fichiers .mp3, et ne fonctionnait uniquement avec des fichiers .wav. Le programme a nécessité des modifications afin qu'il les prenne en charge. En effet, d'une part, l'utilisateur peut avoir besoin d'utiliser ces deux formats qui sont tous deux très répandus mais il fallait également ajouter cette fonctionnalité pour pouvoir utiliser les enregistrements de xeno-canto. Les recherches ne furent pas aussi simples. Beaucoup de solutions trouvées sur internet ne fonctionnaient pas ou plus. Finalement, utiliser un programme déjà mis en place dans beaucoup d'autres logiciels est une solution plus adaptée. Il s'agit de ffmpeg. Ce dernier est plutôt simple d'utilisation, il suffit de l'avoir installé sur son ordinateur avec une simple commande pour Linux. En ce qui concerne Windows, la démarche est un peu plus contraignante pour l'utilisateur. En effet, il faut que ce dernier télécharge des fichiers .exe (qui sont présents dans un fichier compressé 7z dans le projet). Puis il faut qu'il indique le chemin vers le dossier qui contient ces fichiers dans la variable d'environnement path. Par la suite une bibliothèque python nommé pydub se charge d'effectuer les appels des commandes. Ce programme permet de convertir un fichier .mp3 en un .wav. Ce .wav sera

utilisé puis lorsqu'un nouveau fichier .mp3 devra être utilisé par le programme il le remplacera. Il s'agit donc d'un fichier temporaire que l'utilisateur n'est pas censé utiliser.

L'obtention du mfcc d'un enregistrement audio permet de pallier ce problème en obtenant des données uniformes. Le mfcc obtenu correspond à un tableau à 2 dimensions. Cependant la longueur de ce tableau dépend de la longueur de l'enregistrement audio. Pour avoir des données uniformes, la moyenne des différentes valeurs sur l'axe vertical était réalisée. Cette méthode est très efficace pour différencier des sons constants tels qu'un moteur qui tourne, un climatiseur, ou encore un marteau piqueur. A l'opposé, des cris d'oiseaux ou de chauves-souris ne sont ni réguliers, ni constants. Cela explique donc le très mauvais taux de réussite. Afin de pallier ce problème il faut pouvoir alimenter le modèle avec toutes les données, sans faire de moyenne afin de ne pas perdre de l'information. Cependant le même problème se pose, le programme doit être capable d'utiliser des enregistrements

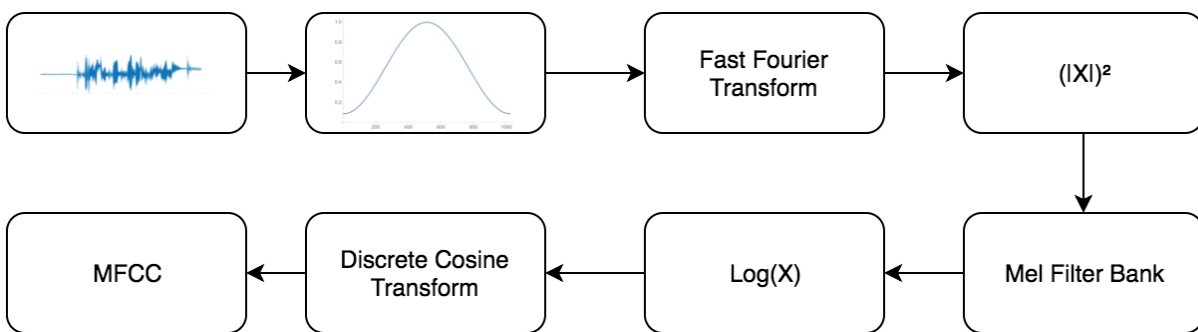


Figure 13 - Algorithme d'obtention du MFCC à partir d'un audio

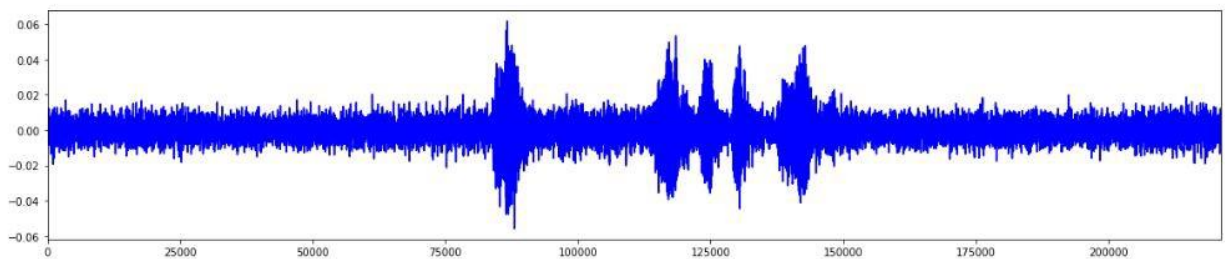
de longueur variable. La solution trouvée à ce problème a été de découper les enregistrements audios en plusieurs audio de plus petite taille, ici 1 seconde. Le programme utilise donc une fonction qui transforme un audio en son mfcc. Le mel-frequency cepstrum est une représentation du spectre de puissance à court terme d'un son, basée sur une transformée en cosinus linéaire d'un spectre de puissance log sur une échelle de fréquence mel non linéaire. Son obtention est relativement complexe. Voici un schéma pour en résumer les différentes étapes.

Ce mfcc correspond à un tableau à deux dimensions. La première correspond à la dimension renseignée en paramètre. La seconde est variable et dépend de la durée de

l'enregistrement audio. Grâce à la standardisation des données par librosa lors de la lecture des fichiers .wav, ces derniers possèdent les mêmes fonctionnalités. Grâce à cela, leur nombre de valeur par seconde est fixe, il est donc possible de le déterminer afin de pouvoir obtenir des données uniformes. Il est donc maintenant possible de récupérer le tableau du mfcc à 2 dimensions et de créer un tableau à 3 dimensions. Ce dernier contient en fait le premier tableau divisé en différents tableaux représentant chacun 1 seconde d'enregistrement. Il y a donc un problème. En effet, lorsque l'enregistrement ne fait pas une durée exacte en seconde, le dernier tableau n'est pas rempli. Pour pallier ce problème, le dernier tableau sera rempli avec des 0.

Afin d'avoir l'algorithme le plus efficace, la décision de modifier l'origine de la préparation des données a été prise. En effet, les bruits de fond sur les enregistrements de xeno-canto sont très présents. Pour contrer ce problème, il est préférable d'appliquer une réduction de bruit. Pour appliquer une réduction de bruit, la bibliothèque noise reducer est la plus répandue. Ce programme permet de nettoyer l'enregistrement audio avant d'obtenir son mfcc.

Original audio file:



Noise removed audio file:

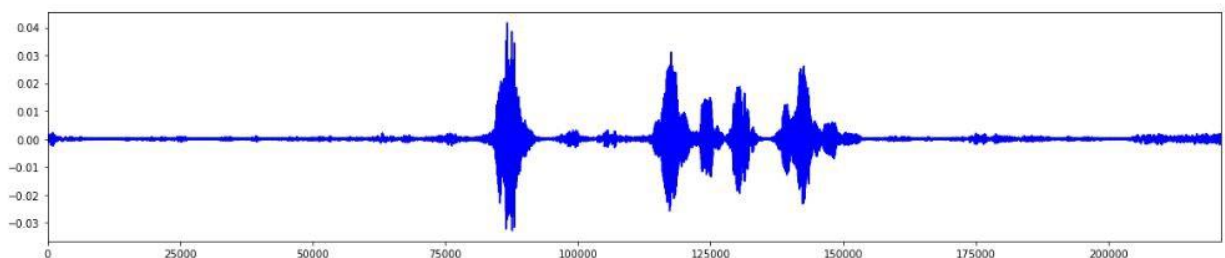


Figure 14 - Exemple d'enregistrement audio nettoyé de ses sons parasites

D'autre part, les enregistrements sont parfois brouillés par des passages de vide. Ces passages correspondent à des secondes de l'enregistrement ou aucun son n'a été perçu. Cela résulte par l'obtention de blancs entre les cris et ces derniers ne doivent pas être interprétés par le modèle. Afin d'ignorer ces passages, ils ne devront pas être sélectionnés. Pour ce faire, la moyenne de toutes les valeurs présentent sur le mfcc initial est calculée. Elle est comparée à chaque passage - d'une seconde -. Afin de rendre cette comparaison efficace et de ne sélectionner que les passages efficaces, il a été décidé d'appliquer un poids de 1,2 à la moyenne. Les enregistrements seront donc conservés uniquement si la moyenne des valeurs de la seconde concerné est supérieure à 1,2 fois la valeur de la moyenne globale de l'enregistrement.

La préparation du son est désormais fonctionnelle, cependant, un problème persiste. En effet, si l'utilisateur souhaite utiliser le spectrogramme pour entraîner le modèle alors que les données avaient été formatés avec le mfcc. Il doit reformater entièrement les données en sélectionnant le bouton radio correspondant au spectrogramme. La préparation étant très longue, cela n'est absolument pas confortable. La partie de l'algorithme qui nécessite le plus de temps n'est pas la conversion en mfcc ou en spectrogramme, ni la concaténation de ces derniers. Cette partie représente entre 4 et 5 pourcents du temps de calcul. La partie qui nécessite le plus de temps et de puissance de calcul est l'extraction du fichier audio. De plus, si le fichier est un .mp3, la conversion est également un processus long. De ce fait, il a été décidé de préparer les données à la fois en utilisant le mfcc et le spectrogramme. L'augmentation du temps de calcul est effectivement négligeable. De ce fait, l'utilisateur n'a qu'à formater une fois ses données, puis il peut entraîner le modèle en sélectionnant la préparation des données souhaitée. L'entraînement du modèle étant entre 10^2 et 10^3 fois plus rapide que le formatage des données, cela est beaucoup moins contraignant.










 class_label	17/03/2021 11:58	Document texte	31 Ko
 test_audio_mfcc.npy	17/03/2021 11:58	Fichier NPY	32 990 Ko
 test_audio_spec.npy	17/03/2021 11:58	Fichier NPY	31 646 Ko
 test_labels_mfcc.npy	17/03/2021 11:58	Fichier NPY	8 Ko
 test_labels_spec.npy	17/03/2021 11:58	Fichier NPY	8 Ko
 train_audio_mfcc.npy	17/03/2021 11:58	Fichier NPY	296 768 Ko
 train_audio_spec.npy	17/03/2021 11:58	Fichier NPY	284 792 Ko
 train_labels_mfcc.npy	17/03/2021 11:58	Fichier NPY	70 Ko
 train_labels_spec.npy	17/03/2021 11:58	Fichier NPY	67 Ko

Figure 15 - Nouveaux fichiers créés lors du formatage des données

Le surentraînement (Overfitting) constitue un des écueils de l'apprentissage. En effet, lorsque l'on cherche à entraîner un modèle, on va lui donner un jeu de données formaté en entrée. Puis à l'aide d'un nombre d'épochs, ces données seront données plusieurs fois au modèle dans le but de le rendre le plus efficace possible.

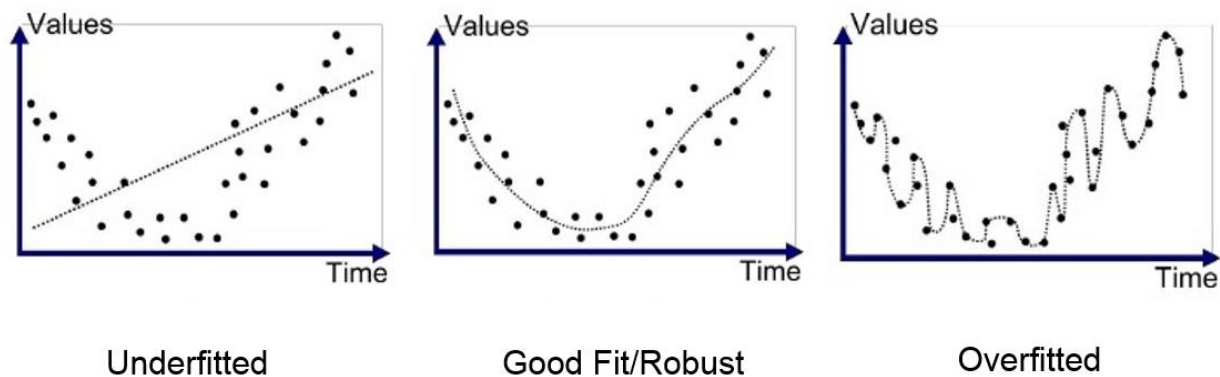


Figure 16 - Graphique présentant le concept de surentraînement (Overfitting)

Cependant, si le nombre d'époch est trop élevé comparé aux données, la précision du modèle continuera d'augmenter sur les données d'entraînement mais baissera sur des données inconnues. Afin de connaître le réel pourcentage de réussite du modèle, il est essentiel de réaliser des tests avec un jeu de données que ce dernier ne connaît pas. Pour ce faire, les données initiales seront séparées en un jeu d'entraînement et un jeu de test. La bibliothèque sklearn propose une fonction qui permet de séparer un jeu de données associées aux labels correspondants.

```
train_test_split(res, labels,  
                 test_size=ratio, random_state=rs)
```

Figure 17 - Appel de fonction permettant l'obtention d'un jeu d'entraînement et un second de tests

Cette fonction permet de séparer les données contenues dans res ainsi que les labels associés. L'option test_size permet de sélectionner la proportion de l'ensemble de données à inclure dans la division de test. random_state contrôle la lecture aléatoire appliquée aux données avant d'appliquer le fractionnement.

3. Création du modèle

Le programme est capable de créer un jeu de données exploitable. Afin de pouvoir obtenir un premier résultat, le modèle simple proposé par le site de [Tensorflow \[6\]](#) a été mis en place. Ce modèle est le suivant :

```
model = tf.keras.Sequential([  
    tf.keras.layers.Dense(128, activation='relu'),  
    tf.keras.layers.Dense(10)  
])
```

Figure 18 - Création d'un modèle simple

Il s'agit d'un modèle séquentiel avec deux couches de neurones. La première est composée de 128 neurones et la seconde de 10. La première couche fait appel à la fonction relu (Rectified Linear Unit) : $f(x) = \max(0, x)$. Sans précision de l'option la fonction utilisée est linéaire (linear) activation: $a(x) = x$.

Ensuite il faut compiler ce modèle, ici aussi l'option d'utiliser l'exemple [du tutoriel \[13\] sur le site de Tensorflow](#) a été privilégiée. Le modèle est le suivant :

```
model.compile(optimizer='adam',  
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),  
              metrics=['accuracy'])
```

Figure 19 - Compilation du modèle avec l'optimizer Adam

La fonction de compilation comporte plusieurs paramètres :

- l'optimiseur³ (optimizer)
- la fonction de perte⁴ (loss)
- Métriques⁵ (metrics)

L'optimiseur utilisé ici est fréquemment utilisé pour la création de CNN⁶, il s'agit de l'optimiseur nommé [adam \[14\]](#).

Etant donné que le dataset ne comporte que très peu d'enregistrement un nombre d'epoch⁷ (1000 dans ce cas, cela reste relativement rapide), le modèle va donc recevoir les données formatés ainsi que les labels associés

```
model.fit(train_audio, train_labels, epochs=1000)
```

Figure 20 - Entraînement du modèle avec les enregistrements, les labels et 1000 epochs

Le taux de réussite obtenu est de **4%**, ce chiffre est vraiment très mauvais, ce modèle n'est donc absolument pas adapté à ce cas de figure.

Afin d'augmenter le taux de réussite catastrophique, premièrement le modèle a été très légèrement modifié. Pour ce faire, le nombre de neurones par couche a été modifié. La première couche en comportera désormais 256, et la seconde 50, toutes les autres options restent les mêmes. Désormais le taux de réussite s'élève à **66%**, l'augmentation est significative mais pas pour autant suffisante. Désormais le modèle comporte 5 couches qui comportent respectivement 256, 128, 128, 64 et 32 neurones. Le résultat obtenu est : **89%**

Afin de pouvoir tester le programme, il est nécessaire de posséder un dataset complet. En effet les enregistrements de chauves-souris étant coûteux à mettre en place,

³ Méthode de mise à jour en fonction des données qu'il voit et de sa fonction de perte

⁴ Mesure la précision du modèle pendant l'entraînement. Il faut minimiser cette fonction

⁵ Surveille les étapes de formation et de test

⁶ Convolutional Neural Network

⁷ Il s'agit du nombre de fois où l'algorithme d'apprentissage fonctionnera sur l'ensemble de données d'apprentissage

leur disponibilité est très faible sur internet. La base de données [UrbanSound8K \[5\]](#) est apparue comme la meilleure des solutions. Cette base de données comporte plus de 8700 enregistrements de sons urbains. Premièrement, il a fallu adapter nos algorithmes de transformation et de récupération des données afin de pouvoir utiliser les différents enregistrements de la base de données.

Afin de pouvoir diviser nos données en deux parties, une première pour l'entraînement, une seconde pour les tests. Le modèle n'aura connaissance que des données d'entraînement et modifiera les équations de ses neurones en fonction des résultats obtenues à chaque epoch. Ensuite une fois toutes les epochs terminées, le modèle essaiera de catégoriser les données de test. la fonction *train_test_split* de la bibliothèque *sklearn.model_selection* permet de séparer aléatoirement les données.

Ensuite, exactement les mêmes paramètres ont été repris. Cependant la durée de l'entraînement du modèle fut très longue étant donnée le grand nombre de données ainsi que le nombre d'epoch mis à 1000. Après une longue attente (réduite grâce à l'utilisation de la carte graphique) le taux de réussite obtenu est de **92,1%**. Finalement, en modifiant le nombre d'epoch afin de faire des prédictions avec 100 epochs puis 10 epochs. Les résultats obtenus sont respectivement **85,5%** et **71,3%**. Les captures d'écran des résultats sont disponibles dans les annexes.

Afin de pouvoir obtenir le meilleur programme possible, des tests en utilisant les mfcc de chaque enregistrement audio ont également été réalisés. Les résultats sont les suivants, **80,5%** pour 10 epochs, **89,6%** pour 100 epochs et **91,3%** pour 1000 epochs. On note donc une augmentation significative lors de l'entraînement avec 10 et 100 epochs. Cependant le chiffre est comparable pour 1000 epochs. Etant donné que 1000 epochs nécessite une grande durée pour se réaliser, il est préférable d'utiliser le mfcc en ce qui concerne le dataset [UrbanSound8K \[5\]](#). Afin de pouvoir vérifier ces informations il faudrait que l'on puisse tester le programme sur une base de données comportant un grand nombre d'enregistrements de chauves-souris.

Le modèle a beaucoup évolué au cours du projet. En effet, le deep learning étant une notion complexe et difficile à prendre en main, de nouveaux concepts ont été acquis au fur et à mesure de la réalisation du projet. Cela a nécessité une formation et une documentation continue. Afin d'améliorer le modèle, une nouvelle couche neuronale variable a été ajoutée. Cette couche est variable car elle permet de déterminer quelle classe correspond le plus aux données reçues en entrée. La fonction qui est attribuée à cette couche neuronale est nommée Softmax. Cela a permis de grandement améliorer le modèle et d'obtenir une meilleure précision. De nouvelles couches neuronales en relu ont également été rajoutées pour agrandir le nombre de neurones disponibles et donc la capacité à évaluer une disposition complexe des données. Cela a été démontré dans la partie prérequis présentant le Machine Learning.

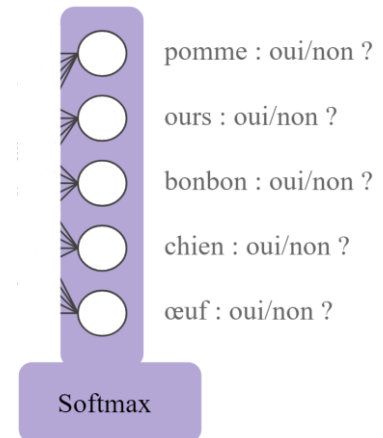


Figure 21 - Application concrète de la fonction softmax

Plusieurs données de xeno-canto ont été utilisées afin de pouvoir tester le modèle. La durée de préparation est désormais longue (de l'ordre de 45 - 60 minutes). Le même modèle a été appliqué, ainsi que la même préparation des données que celle précédemment utilisée sur [UrbanSound8K \[5\]](#). Pour rappel, le taux de réussite sur le jeu de test était de plus de 90%. Cependant, avec ces données le meilleur résultat obtenu fût d'environ 30% ce qui n'est vraiment pas bon et pourrait même s'apparenter à du hasard.

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^K e^{z_j}}$$

Figure 22 - Fonction Softmax

```
tf.keras.layers.Flatten(input_shape=(50, 43)),
```

Figure 23 - Application de Flatten à un Tableau 2D de dimension (50, 43)

Plusieurs problèmes se posent donc et une nouvelle préparation des données s'est imposée. Les données ont donc été retravaillées et avec des tableaux à deux dimensions le modèle a dû, lui aussi être modifié. Pour ce faire, une couche Flatten.

Cette couche permet de prendre en entrée un tableau correspondant aux dimensions indiquées dans la variable `input_shape`. Ici les tableaux comportent 50 valeurs par 43. 50 correspond à la donnée fournie lors de l'obtention du mfcc ou du spectrogramme. 43 correspond aux nombres de valeurs comprises dans une seconde d'enregistrement. Le taux de réussite final obtenu sur des sons d'oiseaux et des cris de chauves-souris oscillent entre 60% et 70%.

Afin que l'utilisateur puisse obtenir la meilleure précision possible lors de l'entraînement du modèle, une nouvelle fonction associée à un bouton a été développée. Cette fonction permet de déterminer le meilleur nombre d'epochs. Pour ce faire, l'algorithme va incrémenter un compteur d'epochs et entraîner un modèle fictif au fur et à mesure pour détecter à partir de quel point le sur entraînement commence à se faire ressentir. Le nombre d'epochs est automatiquement modifié dans le champ prévu à cet effet.

4. Interface utilisateur

Afin de pouvoir proposer ce programme au plus grand nombre, il est nécessaire d'avoir une interface graphique. Cette dernière permet à l'utilisateur d'interagir avec des algorithmes complexes sans qu'il ne s'en rende compte. Cela lui permet également de sélectionner différentes options afin que ce dernier puisse profiter de la meilleure expérience possible.

La bibliothèque TKinter semble la plus appropriée, principalement pour deux raisons. La première, est le fait que cette bibliothèque existe depuis 2003, de ce fait sa documentation ainsi que des exemples d'utilisation sont très facilement trouvables. La seconde, un des membres de l'équipe, Thomas, avait déjà utilisé cette bibliothèque pour un projet pour le Bac. Cette bibliothèque possède une utilisation simple et de ce fait est relativement limitée dans son utilisation. En effet, les options pour les widgets proposés (boutons etc...) ne sont pas très nombreuses. Il est donc important de bien maîtriser toutes les options disponibles afin de pouvoir exploiter au mieux la bibliothèque.

Ce programme va nécessiter d'un grand nombre de widget propre à TKinter. En effet chaque bouton, canvas, zone de texte correspond à un élément. Cela engendre donc une centaine d'éléments. Dans le but de les organiser, l'application est donc découpée en différentes parties. Ces parties sont représentées par des classes. Chaque classe comporte tous les éléments propres à une partie. De plus, il est de ce fait facile de créer des méthodes dans ces classes afin de pouvoir gérer les interactions graphiques telles que la modification du pourcentage de réussite. Cela permet également de récupérer les valeurs des variables tel que le nombre d'epochs. Au total, 8 classes ont été créées, elles représentent chacune une partie de l'application. Afin de créer toutes ces classes, chacune possède sa propre fonction d'initialisation. Cela permet de créer les classes dans la fonction principale du programme.

```
class Menu:...  
  
class Header:...  
  
class InfosMenu:...  
  
class Footer:...  
  
class AffichageSon:...  
  
class AffichageRes:...  
  
class RecapSelect:...  
  
class Console:...
```

Comme évoqué précédemment, TKinter est une bibliothèque très pratique d'utilisation, cependant certaines fonctionnalités ne sont pas disponibles et cela a été à l'origine de plusieurs problèmes. Premièrement l'utilisateur doit pouvoir visualiser l'enregistrement audio qu'il souhaite classifier. Pour ce faire, l'utilisateur doit être en mesure de visualiser le spectre sonore de l'enregistrement, son spectrogramme et son mfcc. Afin de pouvoir afficher ces représentations il faut pouvoir afficher des figures. Cela nécessite une bibliothèque spécialisée dans l'affichage de figures. Le choix s'est arrêté sur la collection pyplot de Matplotlib. En effet, au même titre que TKinter, cette bibliothèque est fortement documentée et très répandue. Le résultat obtenu est celui présenté sur la page de couverture du rapport. Cette bibliothèque a également été mise en place pour réaliser des barres de chargement lors des longues attentes présentes lors du formatage des données. Cette barre pourra être utilisée simplement dans de futurs programmes. Le fichier associé à ces fonctions est

Figure 24 - Liste des classes

progress_bar.py qui permet l'initialisation, la mise à jour et la fermeture de la barre de chargement.

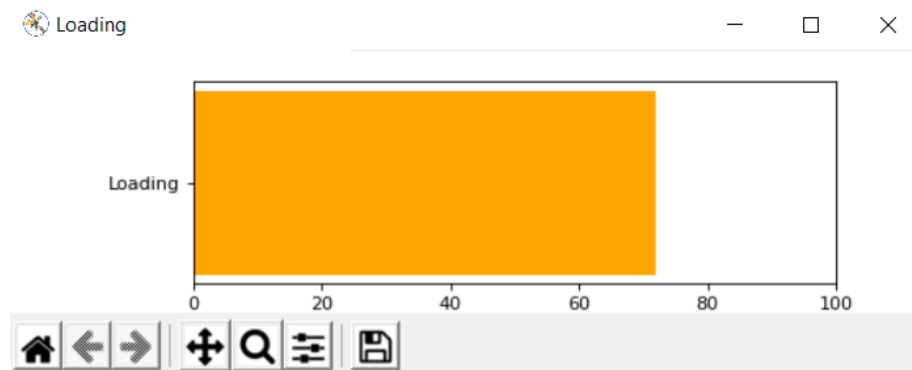


Figure 25 - Barre de chargement utilisée par le programme

Afin de pouvoir mettre un programme en production, il est primordial d'accorder un soin tout particulier aux fonctionnalités dédiées à l'utilisateur. Ici, la cible est principalement des biologistes. Ce ne sont donc pas des informaticiens et le but de ce programme est de leur faire gagner du temps en leur apportant une expertise supplémentaire. S'ils doivent perdre du temps à essayer de l'utiliser, ce dernier ne remplit pas son rôle.

Tout d'abord, il est essentiel de pouvoir effectuer les actions de base. Pour ce programme, il s'agit de pouvoir formater des données, d'entraîner un modèle avec ces données puis de pouvoir réaliser une prédiction. Afin de pouvoir formater les données, il faut que le programme ait connaissance du dossier contenant les données.

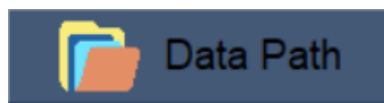


Figure 26 - Bouton pour sélectionner les données

Ensuite, afin de parcourir ce dossier et connaître les informations sur les données, l'utilisateur doit renseigner le chemin vers son fichier .csv (Un déjà existant, ou un auto-généré).



Figure 27 - Bouton pour sélectionner le fichier .csv

Ce fichier doit comporter plusieurs informations, toutes ces informations doivent donc être connues par l'utilisateur. Une fonctionnalité permettant d'afficher une fenêtre pop-up a été rajoutée. Cette fenêtre affiche l'aide pour expliquer le fonctionnement du programme. Cette rubrique aide est générée à partir d'un fichier .txt. Afin que les utilisateurs francophones puissent aisément lire cette aide, une traduction a également été réalisée.



Figure 28 - Boutons pour afficher l'aide

Ensuite une fois ces deux données renseignées et donc enregistrés par le programme, ce dernier doit recevoir l'ordre de réaliser le formatage des données afin que ces dernières soient uniformes. En effet, il est préférable de laisser le choix à l'utilisateur de choisir quand lancer ce formatage car c'est une action qui nécessite un long temps de réalisation. C'est d'ailleurs pour cette étape que la barre de chargement a été ajoutée.

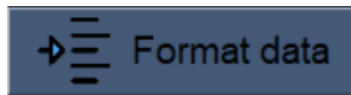


Figure 29 - Bouton pour formater les données

Une fois les données formatées et sauvegardées, il ne manque plus qu'à entraîner le modèle avec le ratio et RS indiqué.

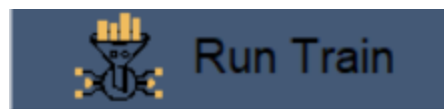


Figure 30 - Bouton qui lance l'entraînement du modèle

Une fois le modèle entraîné, l'utilisateur va avoir besoin de faire des prédictions, pour réaliser ces prédictions, il va devoir sélectionner un fichier puis réaliser la prédiction. L'utilisateur choisit déjà le fichier à tester avec 'Test Path', puis utilise les données formatées afin d'obtenir une estimation de l'espèce avec un pourcentage de précision à l'aide de 'Run Test'.



Figure 31 - Boutons pour sélectionner le fichier à tester et lancer la prédiction

Afin de pouvoir traiter les données fournies dans le data path, il est nécessaire d'avoir un fichier tableur de format .csv qui indique quel fichier correspond à quelle espèce d'oiseau / de chauve-souris. Néanmoins, sur certains dataset conséquents, faire ce tableur est une tâche longue et fastidieuse. Le bouton 'Generate csv' a pour but de générer automatiquement un fichier .csv contenant les informations nécessaires pour le bon fonctionnement des étapes suivantes, telle que le formatage des données. Il faut néanmoins bien faire attention à ce que les données du dataset soient bien organisées comme indiqué dans l'aide. En effet, cet algorithme parcourt l'entièreté d'un dossier en descendant dans son arborescence. Il génère également des numéros de classe et des noms de classe en fonction des noms des fichiers. Il prend en compte les fichiers .wav et .mp3. S'il rencontre un fichier d'un autre type, il ne génère pas le fichier et se termine en décrivant l'erreur.

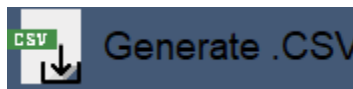


Figure 32 - Bouton pour générer le fichier .csv

Afin d'économiser du temps à l'utilisateur, il est aussi possible pour ce dernier d'importer des données déjà formatées, ainsi qu'un modèle d'entraînement déjà fonctionnel. Ceci est particulièrement utile pour les dataset conséquents, afin d'éviter un traitement pouvant parfois durer plus d'une heure, ou bien pour importer un modèle préexistant afin d'avoir des résultats stables par rapport à d'éventuelles déductions précédentes.



Figure 33 - Boutons pour importer les données formatées / modèle

Afin de formater les données, il faut un nombre d'epochs (répétitions) afin d'indiquer au programme le nombre de fois qu'il doit affiner le dataset lors du formatage. Il faut avoir un nombre d'epochs assez élevé pour avoir une précision améliorée, tout en faisant attention

à ne pas mettre un nombre trop grand, au risque de surentraîner le modèle, le rendant moins fiable. Pour les utilisateurs expérimentés, il est possible de changer le nombre d'epochs manuellement en fonction de leurs besoins, mais il est possible d'utiliser le bouton correspondant afin d'obtenir un nombre d'epochs considéré optimal par le programme.

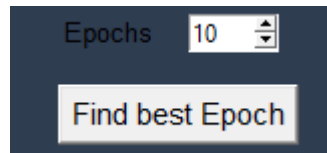


Figure 34 - Boutons pour choisir le nombre d'epochs manuellement, ou automatiquement

Lors du formatage, l'utilisateur peut choisir une méthode utilisant le mfcc, ou un spectrogramme, en faisant attention que si l'utilisateur souhaite changer de méthode de formatage, il devra reformater les données à nouveau.

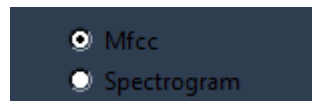


Figure 35 - Boutons pour choisir la méthode de formatage

Lors du formatage des données, une fonction particulière de la bibliothèque sklearn est utilisée afin de garantir une génération aléatoire des données. Cette génération utilise deux variables Ratio et Random State afin de déterminer l'échantillonnage utilisé lors du formatage et la façon dont sont sélectionnées les fichiers respectivement. Par défaut, le ratio est à 0.1 et le RS est à 42. En d'autres termes, cela veut dire que sur 10% du dataset, 1 fichier tous les 42 sera choisi lors du formatage. Ces boutons servent à ajuster ces valeurs pour les utilisateurs voulant un formatage particulier.



Figure 36 - Boutons pour changer les valeurs de ratio et RS

Si l'utilisateur doit quitter l'application, mais souhaite réutiliser ces données / modèles pour une utilisation ultérieure, il y a une option qui permet de sauvegarder les données ou le modèle afin d'éviter un formatage parfois très long.

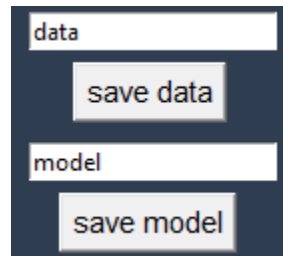


Figure 37 - Boutons pour sauvegarder les données formatés / modèle

Après avoir choisi le fichier de test, l'utilisateur a plusieurs options afin de pouvoir entendre ou/et visualiser le son qu'il souhaite tester. Il peut, par exemple, choisir de jouer l'enregistrement afin de pouvoir l'entendre directement sans avoir à recourir à un logiciel tiers, de voir la représentation graphique / spectrogramme / MFCC du fichier, afin d'avoir des informations avancées sur le fichier.

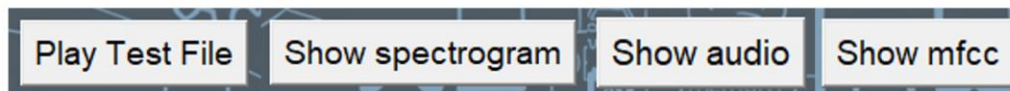


Figure 38 - Boutons pour entendre / visualiser le fichier sonore

Tous ces éléments ont été disposés dans le but de créer une interface conviviale et intuitive pour l'utilisateur. En effet, bien souvent les interfaces sont agressives au premier regard et n'ont pas pour objectif d'être accessibles à tous. Tkinter est une bibliothèque qui se veut simple et efficace. Elle n'a pas pour objectif d'être esthétique. Sur l'image ci-contre on retrouve tous

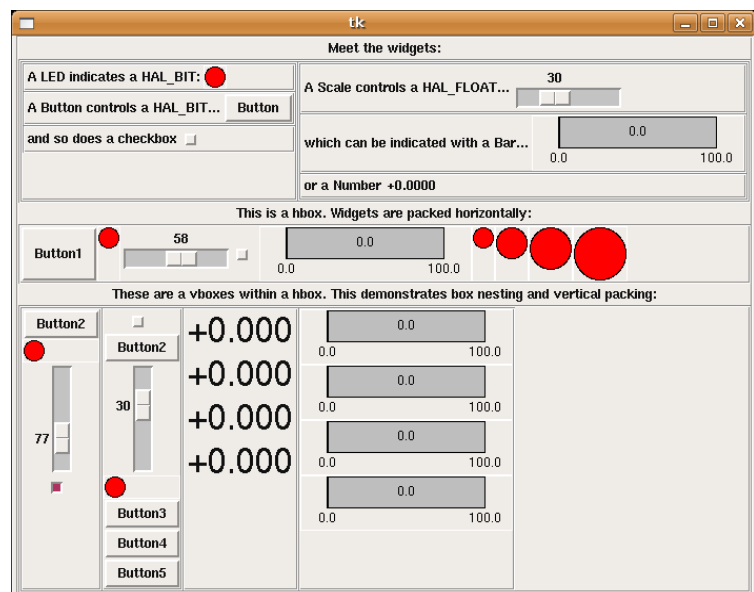


Figure 39 - Exemple d'interface graphique Tkinter

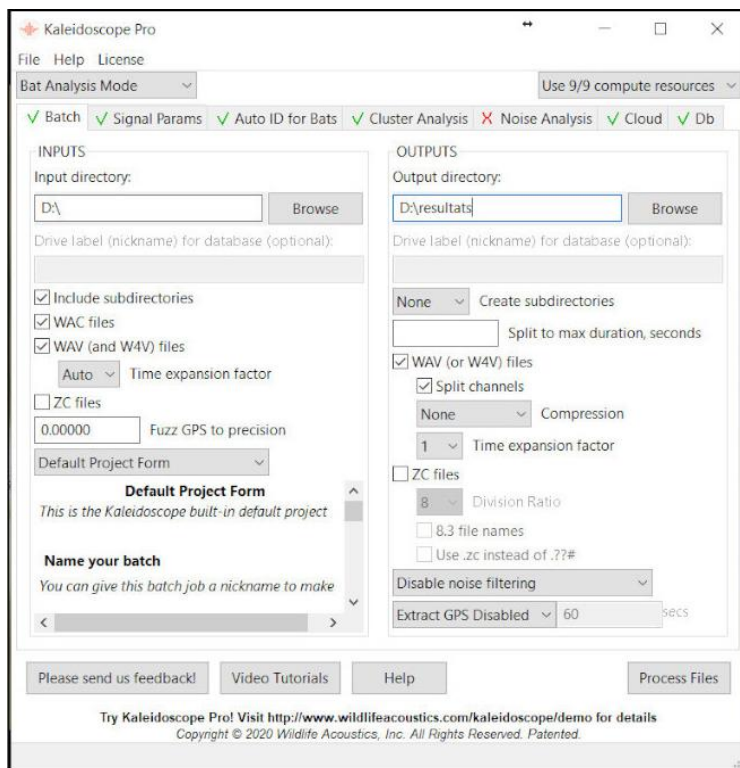


Figure 40 - Interface graphique de Kaléidoscope Pro

les éléments disponibles avec Tkinter sans aucune personnalisation. On retrouve des boutons, des checkbox etc... Ces derniers sont froids et peu accueillants. Ce principe de réaliser un logiciel efficace plutôt qu'esthétique est répandu dans le milieu scientifique. Les logiciels permettant l'analyse des cris de chauves-souris ne font pas exception. Il est possible de constater cela en observant le logiciel présenté ci-contre. Il s'agit

du logiciel Kaléidoscope Pro, proposé au prix de 439€. L'interface est monochrome et l'organisation quelque peu approximative. En partant de ce constat, la décision a été prise de réaliser une interface constatant les interfaces des logiciels présents sur le marché. Une grande partie du projet a donc été consacrée à concevoir, améliorer et perfectionner l'interface graphique. Les grandes lignes directrices étaient de réaliser une interface intuitive, simple et travaillée. De nombreux détails ont donc été apportés afin de permettre la meilleure expérience d'utilisation possible. Parmi ces détails, on retrouve les icônes sur les boutons principaux permettant de les localiser rapidement et de plus facilement comprendre leur utilité. Des parties consacrées à la compréhension de l'utilisateur ont également été ajoutées tel que la partie console qui communique les messages d'erreur et de succès à l'utilisateur. La partie située en dessous de la console permet quant à elle de récapituler les chemins fournis par l'utilisateur pour les données et le fichier .csv. L'entièreté du projet a été réalisée en anglais dans un souci de conformité internationale. Cependant, l'application étant particulièrement destinée à un public francophone du fait du lieu de sa réalisation, l'aide a été traduite en français. La disposition des boutons principaux en haut à

gauche n'a pas non plus été laissée au hasard. En effet, leur disposition a été réalisée par ordre chronologique. Ensuite les boutons mis côte à côte ont pour but de représenter un choix. Effectivement l'utilisateur peut choisir un chemin vers son propre fichier .csv ou bien le générer automatiquement. Il en est de même pour les boutons suivants. C'est donc après plusieurs semaines de réflexion et de modifications qu'est née l'interface graphique suivante.



Figure 41 - Graphique final de l'application ML Sound

5. Algorithmes annexes

Afin de pouvoir réaliser un programme, il arrive que des algorithmes soient nécessaires sans pour autant qu'ils soient directement intégrés au programme. Ces algorithmes ont tout de même nécessité un temps de réalisation important. Le premier algorithme réalisé dans ce but est l'algorithme de génération automatique de fichier .csv. Ce dernier a fini par être intégré au programme car son utilité pour les utilisateurs a finalement été démontrée. Cet algorithme utilise le principe de la récursivité pour aller chercher en profondeur dans le dossier. Quatre colonnes sont alors créées. Les deux premières permettent de retrouver le fichier. En effet, la première correspond au nom et la seconde au chemin (dossiers) vers ce fichier. Les deux dernières permettent de classer les fichiers. La

troisième attribue un numéro de classe fixé arbitrairement par ordre croissant d'apparition des fichiers. Finalement le nom de la classe est lui attribué en fonction du dossier mère dans lequel il est présent. Il a fallu rajouter une vérification du type du fichier afin de détecter une éventuelle mauvaise utilisation de la part de l'utilisateur. Initialement, cet algorithme était utilisé pour réaliser des tests sur le programme. En effet, les enregistrements de xeno-canto ne possédant pas de fichier .csv fournis. Le réaliser manuellement était trop long et fastidieux. Cet algorithme a donc été ajouté au logiciel et donc à l'interface graphique. Sa première version n'était pas très concluante car le fichier généré se trouvait dans un dossier qui été vidé lors de chaque formatage des données afin de pouvoir y stocker les nouveaux fichiers. Il devait donc aller déplacer le fichier généré dans un autre dossier et ensuite sélectionner dans l'application le chemin vers ce fichier qu'il avait préalablement déplacé. Afin de corriger ce problème, le dossier dans lequel était généré le fichier .csv a été modifié et son nettoyage également. De plus, pour éviter une étape supplémentaire à l'utilisateur, désormais le chemin vers le fichier est automatiquement renseigné lors de sa génération.

Ensuite, un second algorithme a été nécessaire, une fois le projet terminé. M. Canon nous a présenté des enregistrements de cris de chauves-souris. Ces derniers étaient tous contenues dans un seul dossier. Le fichier .csv récapitulant les informations ne respectait pas les normes nécessaires au programme pour prendre en compte le fichier. Pour ce faire, un algorithme a été développé. Ce dernier sépare les données en deux dossiers répartissant d'une part les enregistrements présentant des Pipistrelles et le second toutes les autres espèces. Ce choix a été commenté précédemment. Le fichier .csv fourni contient certaines informations très pratiques tel que la correspondance entre le nom du fichier et l'espèce associée. C'est d'ailleurs ces informations qui vont permettre à l'algorithme de déplacer les fichiers dans les dossiers correspondants. L'algorithme utilisé est semblable à l'algorithme qui permet la génération du fichier .csv.

IV - Bilan, gestion de projet

1. Fonctionnement de l'application

Le fonctionnement de l'application est désormais complet. Ce chapitre a pour but d'expliquer le fonctionnement du programme au travers d'un diagramme d'activité UML. Premièrement, l'utilisateur doit avoir installé le programme sur son ordinateur. Pour l'installer sous linux, il faut cloner le projet, installer les bibliothèques ainsi que ffmpeg puis lancer le programme. Sous Windows, il suffit de lancer l'installateur puis de lancer le raccourci créé sur le bureau. Ensuite l'utilisateur arrive sur l'interface graphique Plusieurs choix s'offrent à lui.

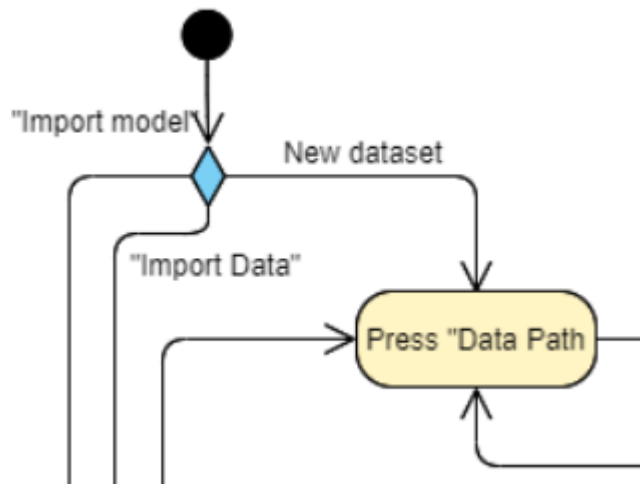


Figure 42 - Début du diagramme d'activité

L'utilisateur peut alors choisir entre les différentes options d'importation. Ces fonctionnalités ont été décrites précédemment. Le chemin traditionnel veut que ce dernier sélectionne un chemin vers le dossier contenant les données.

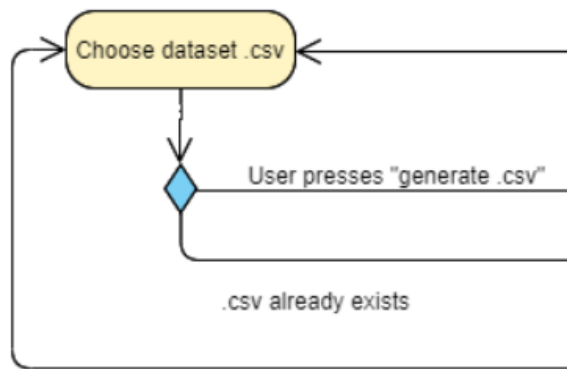


Figure 43 - Partie utilisateur concernant le choix du fichier .csv, extrait du diagramme d'activité

Ensuite l'utilisateur a le choix entre choisir son propre tableur .csv ou bien le générer automatiquement. Par la suite les étapes sont relativement simples pour l'utilisateur il doit formater ses données puis patienter. Le ratio dépend de la taille de ses données mais le ratio est d'environ 10 minutes pour 1 Go.

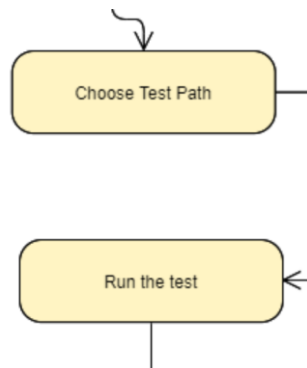


Figure 44 - Zoom sur la partie prédiction du programme, extrait du diagramme d'activité

Ici on observe la partie prédiction du programme. L'utilisateur sélectionne un fichier audio dont il ne connaît pas l'espèce de la chauve-souris présente sur ce dernier – le programme peut également fonctionner avec différents cris d'animaux (oiseau mais également fauves etc..) à condition que les données d'entrainements correspondent -. Ensuite, il presse le bouton permettant de lancer la prédiction.

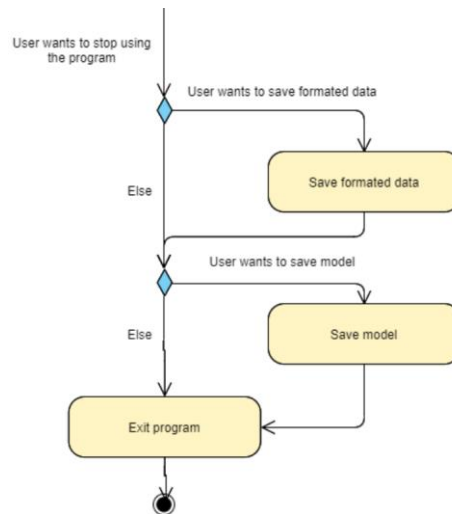


Figure 45 - Zoom sur la partie exportation du programme, extrait du diagramme d'activité

Lorsque l'utilisateur a fini d'utiliser le programme sur sa machine, il peut avoir besoin de sauvegarder certains éléments. Pour ce faire les boutons appropriés sont disponibles sur l'interface. Il est important de préciser que les données formatées et le model entraîné sont sauvegarder en local et sont rechargés à chaque ouverture du programme. Cela peut être comparer à une sauvegarde automatique pour éviter toute fermeture par inadvertance ou à la suite d'un problème technique. Ainsi, l'utilisateur n'a pas besoin de reformater les données. Le diagramme d'activité UML complet du programme est disponible en [annexe](#).

2. Gestion de projet

Durant une licence en informatique, de nombreux projets sur des durées n'excédant pas 3 mois sont attribués. Ils sont à réaliser en binôme ou en monôme. Ce projet était donc le premier en trinôme et également le premier sur une si longue durée (6 mois). De plus, c'était également le premier projet ou un tuteur était attribué. Le projet, contrairement au projet habituel à réaliser en licence, ne possède pas de cahier des charges. Une problématique est posée et le but est de résoudre cette dernière. Pour ce faire, la mise en place de la méthode Agile est primordiale. En effet, cette dernière consiste à mettre en relation l'avancé du développement avec le ou les clients. Ici le tuteur de projet. Pour ce faire, l'organisation du projet va se rapprocher de la méthode SCRUM en réalisant cependant des

sprints plus courts qu'habituellement. En effet, habituellement, un sprint SCRUM prend entre 1 et 4 semaines (4 à 5 pour les gros projets tel que le développement de Just Dance comme l'a affirmé M. Sylvain Rousseau Project Coordinator Live chez Ubisoft). Pour ce projet, les réunions étant hebdomadaires (hors vacances scolaires), les sprints ont donc une durée de 1 semaine. Une mêlée hebdomadaire est alors réalisée lors de chaque réunion. Durant ces réunions, le but est de présenter les avancées, de comparer les résultats obtenus avec les précédents si des phases de tests ont été réalisées. De montrer le résultat des recherches réalisées.

Afin de pouvoir mettre en commun le code réalisé, Github semble être la meilleure solution. En effet, durant le développement, il est simple de partager du code et de les fusionner. De plus, lors de chaque réunion, le tuteur peut récupérer le projet mis à jour et le tester directement sur sa machine. Cela a permis de détecter bon nombre de bugs et de dysfonctionnements. Github est un outil complexe qu'il faut tout d'abord apprendre à maîtriser. Son but est d'être un dépôt de source, il faut donc bien le paramétrer pour ne garder que les fichiers essentiels. Cet outil avait déjà été manipulé (principalement à cause du confinement) afin de pouvoir collaborer lors de précédents projets.

Comme il l'a été décrit précédemment, les réunions hebdomadaires ont pour but de faire le point sur l'avancée du projet, de valider les tâches réalisées et de déterminer les objectifs à atteindre. Pour ce faire, la mise en place d'un outil dédié est essentielle. Pour ce faire, l'outil Trello qui est un outil de gestion de projet en ligne est parfait. Ce logiciel s'inspire de la méthode Kanban (qui signifie étiquette) de Toyota. Cette méthode consiste à répartir les tâches selon des étiquettes ou tickets qui peuvent ensuite être attribués à un employé ou à une équipe. Ces tickets sont placés dans un tableau qui représente leur avancement. Au fur et à mesure que le projet avance, les tickets changeront de colonnes. Trello reprend ce principe avec une interface web. Le déroulement du projet s'est donc organisé autour d'un Trello. Lors de chaque réunion, les tickets situés dans la partie terminée étaient récapitulés et leur réalisation discutée. Les tickets n'ayant pas avancé ou ayant rencontré des problèmes

sont également au cœur des discussions. Ensuite, les nouveaux objectifs sont déterminés lors de la réunion puis attribués durant le déroulement du sprint.

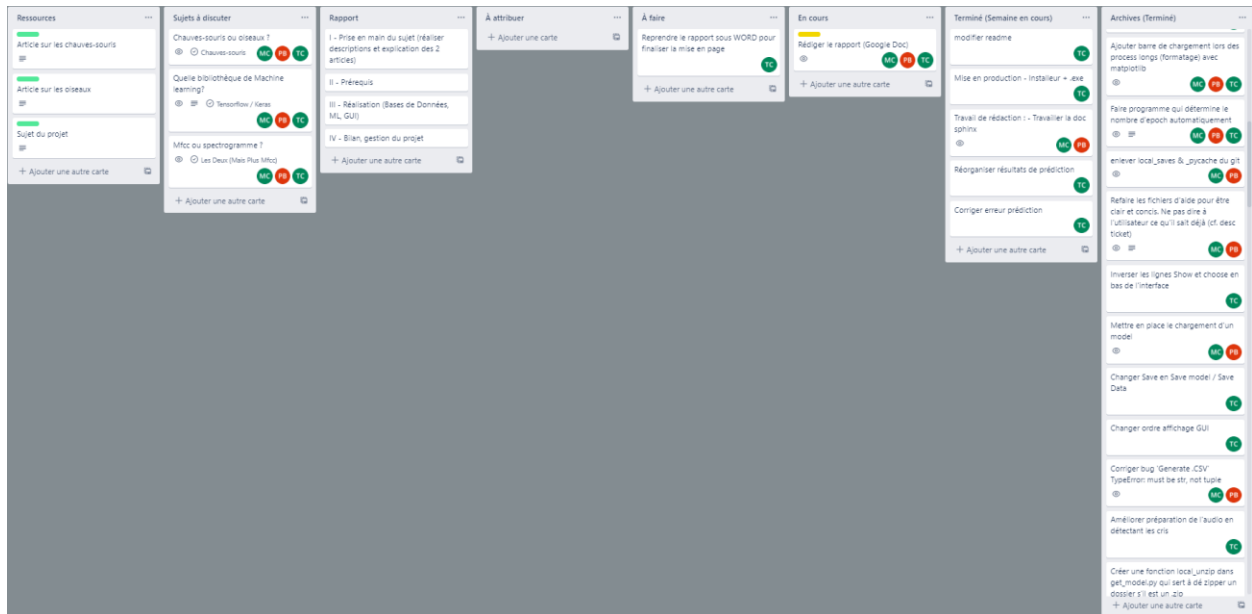
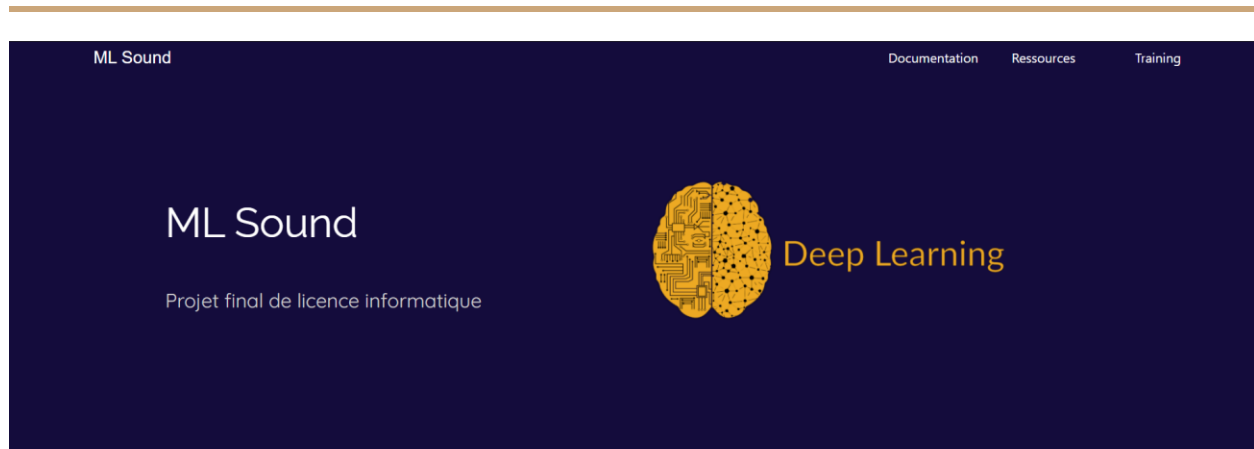


Figure 46 - Trello du projet

Au début du projet, un site internet a été réalisé. Ce dernier a permis de réunir toutes les ressources nécessaires afin de pouvoir y accéder depuis n'importe où. Cela avait pour but de ne pas encombrer le Github qui a pour vocation d'être uniquement un dépôt de sources. En effet, on retrouve sur ce derniers plusieurs documents utiles tel que le sujet du projet, les documentations scientifiques associées et les premiers résumés de réunion avant que la gestion ne se fasse directement via Trello. Ce site contient également des entraînements (nommés training) ayant pour but de former également chaque membre de l'équipe afin que chacun possède les mêmes pré requis. On retrouve des entraînements pour installer python et le logiciel Pycharm, pour prendre en main python et Tkinter et enfin un dernier pour configurer Github sous Pycharm. La page d'accueil répertorie les liens vers les différentes sections du site, elle présente également une Timeline du projet et récapitule les différents liens utiles à l'équipe (Github, Trello et Google Doc). Désormais, le projet est arrivé à son terme et ce site permet également de présenter la [documentation du projet \[15\]](#).



PAGES

Vous trouverez ici les différentes parties du site

Figure 47 - Site internet du projet

Jusqu'ici, bon nombre d'outils collaboratifs ont été utilisés, et la rédaction du rapport n'y échappe pas. En effet, nous avons utilisé une application web que l'on ne présente plus, Google Doc. La majorité du rapport a été rédigé grâce à cet outil qui s'est avéré très pratique pour présenter au tuteur l'avancée de ce dernier. Le rapport a ensuite été repris sur Word pour pallier certaines fonctionnalités absentes de google doc.

Tous ces outils ont été utilisés pour simplifier la collaboration des membres de l'équipe. D'autres outils ont également été utilisés pour simplifier l'utilisation du programme. C'est notamment le cas de Sphinx. Sphinx est un générateur de documentation libre développé pour la communauté python. En effet, il s'agit de la documentation officielle pour ce langage. Au même titre que la Java Doc pour le java. Cet outil a été très utile pour générer l'entièreté de la documentation du projet. Sa configuration a rencontré quelques difficultés au niveau de la configuration des fichiers initiaux. Cela était principalement dû à la répartition des fichiers du projet dans plusieurs dossiers, voire sous-dossiers. Plusieurs options sont disponibles tel que la possibilité de modifier le thème. Le thème utilisé ici est le : sphinx_rtd_theme. Ensuite, afin de générer un fichier html, il faut préalablement préparer le fichier .rst correspondant. Ce dernier permet de décrire à Sphinx l'information souhaitée afin que ce dernier puisse générer la page HTML par rapport au module renseigné par

exemple. Pour ce faire, plusieurs fichiers .rst ont dû être créés et modifiés afin d'obtenir le rendu ci-dessous.

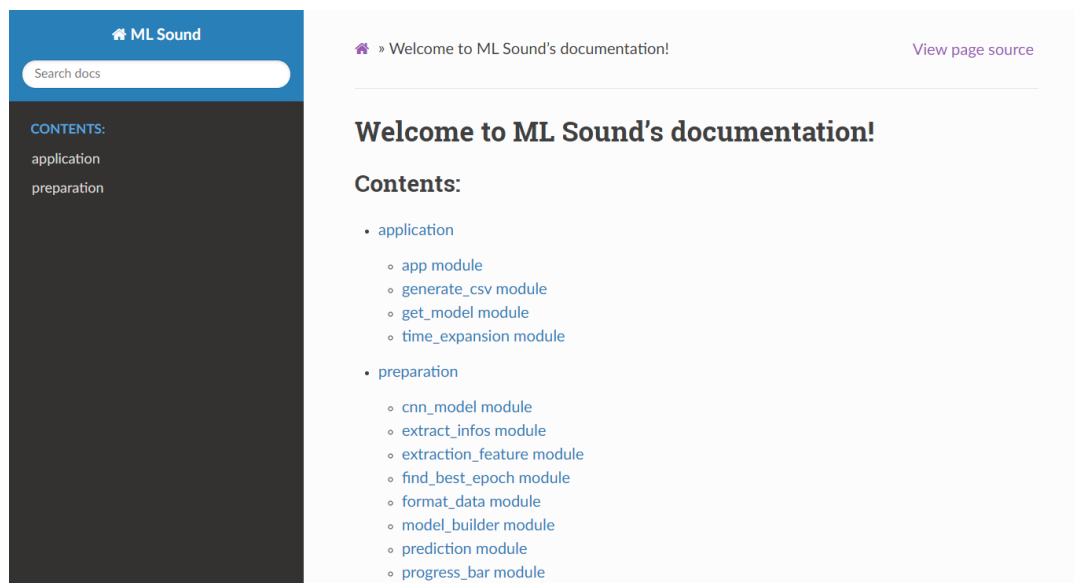


Figure 48 - Documentation du programme

Un second élément est primordial pour l'utilisateur, il s'agit du ReadMe disponible sur le Github du projet. En effet, c'est ce fichier qui sera affiché lors de l'arrivée sur la page d'accueil du Github. C'est donc ce fichier qui va devoir récapituler toutes les informations importantes sur le projet, son installation, son fonctionnement etc... Tout doit être expliqué de façon très concise. Dans une logique de continuité, ce fichier est également en anglais. Le langage couramment utilisé pour rédiger ce genre de fichier est le Markdown (extension .md). Il s'agit d'un langage de balisage léger, il peut également recevoir du code html. On retrouve également les dépendances et les références du projet.


3. Bilan

Une fois le projet terminé, il est simple pour un utilisateur Linux de copier le projet, installer les requirements et lancer le programme. Cependant pour un utilisateur Windows qui n'a, bien souvent, pas l'habitude de manipuler autre chose que des .exe, la tâche devient plus

compliquée. Pour ce faire, il a été décidé de réaliser, à partir du projet un fichier .exe (associé aux dossiers et fichiers complémentaires).

Plusieurs méthodes ont été essayées mais beaucoup se sont soldées par un échec. Notamment parce qu'elles n'étaient pas maintenues à jour et que le projet utilise la dernière version de python (lors de son commencement en octobre 2020). Finalement, le projet Auto PY to EXE a permis de générer tous les fichiers nécessaires au fonctionnement du .exe. A l'exception d'un dossier. Le dossier contenant la bibliothèque Librosa ne s'est jamais généré. Après localisation du problème, le dossier a été copié à partir du projet initial. Une fois ce problème réglé, le .exe fonctionnait.

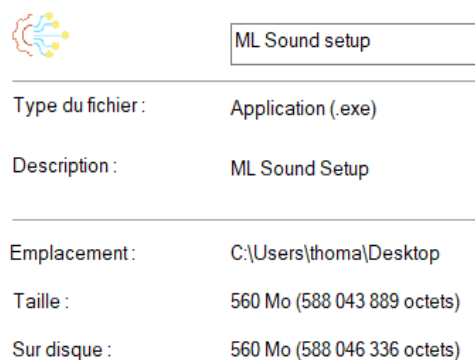
Désormais, un autre problème se pose. Le projet nécessitant bon nombre de bibliothèques et quelques images. Ce dernier atteignait presque les 1,5Go et les 2500 fichiers. Deux solutions se proposent alors. La première et la plus simple, réaliser une archive .zip, .rar ou .7z puis la mettre à disposition de l'utilisateur. Cependant, cette pratique n'est pas la plus répandue sous Windows et c'est cette réflexion qui amène à la



Type :	Dossier de fichiers
Emplacement :	C:\Program Files (x86)
Taille :	1,46 Go (1 573 713 107 octets)
Taille sur le disque :	1,46 Go (1 577 005 056 octets)
Contenu :	2 425 Fichiers, 207 Dossiers

Figure 50 - Propriétés du dossier du programme

seconde option. Cette seconde option consiste à créer un installateur qui lui aussi serait un .exe. L'utilisateur aurait juste à l'exécuter, suivre les étapes et à la fin, un raccourci sera créé



Type du fichier :	Application (.exe)
Description :	ML Sound Setup
Emplacement :	C:\Users\thoma\Desktop
Taille :	560 Mo (588 043 889 octets)
Sur disque :	560 Mo (588 046 336 octets)

Figure 49 - Propriétés du .exe permettant l'installation du programme

sur son bureau. Le logiciel Inno Setup Compiler a paru être une bonne option. Ce dernier possède son propre langage de script .iss (Inno Setup Script). C'est donc grâce à un script .iss (Ce script est disponible sur le dépôt [Github du projet \[16\]](#)) que l'installateur de l'application a été réalisé. Cela a permis de compresser les fichiers (560 Mo) et de rendre leur décompression simple pour l'utilisateur. Grâce à cela, l'utilisateur peut désormais utiliser le projet de la plus simple manière

qui soit, il lui suffit de le télécharger, de l'installer puis dans lancer l'application via le raccourci. Ensuite, il n'a plus qu'à lancer le programme et utiliser des technologies avancées de deep learning sans s'en rendre compte.

Durant ce projet, bon nombre de difficultés ont été rencontrées. La première à se profiler fût la difficulté à trouver des données. Ensuite, la prise en main des technologies complexes du deep learning a également été un gros frein. En effet, les documentations scientifiques, presque exclusivement en anglais n'aident pas à la compréhension d'un nouveau segment de l'informatique. De plus, en licence ces technologies ne sont pas abordées. Les bases de l'intelligence artificielle sont vues lors de certaines unités d'enseignements sur les algorithmes mais rien de tel. Il a donc fallu prendre en main, seul, des concepts reposant exclusivement sur des mathématiques poussées.

4. Perspectives d'amélioration

Un projet comme celui-ci représente un très grand investissement. Le sujet choisi, le Deep Learning est un domaine très vaste. En effet, un groupe d'étudiant de troisième année tel qu'il en est question ici peut très bien, avec beaucoup d'investissement personnel réaliser une production fonctionnelle. Mais cela peut également être un sujet de thèse pour un doctorant en intelligence artificielle. Ou bien encore le sujet de recherches d'ingénieurs dans une grande entreprise tel que Google. De nombreuses pistes d'améliorations sont donc possibles.

Premièrement, avec un meilleur réseau de scientifiques ou de laboratoires, il est possible d'obtenir de meilleures données. Des sons enregistrés avec des appareils plus performants auront un meilleur mfcc et spectrogramme. Cela signifie que ces derniers seront plus lisibles et donc le programme sera bien plus performant.

Ensuite, si le projet se poursuivait sur quelques mois, il aurait été possible d'améliorer la préparation des données. Afin de mieux préparer les données, il aurait été intéressant de se

former davantage à la manipulation du son. De ce fait, un meilleur positionnement des cristaux aurait sûrement permis une meilleure uniformisation des données et donc un meilleur taux de réussite.

Afin d'améliorer le modèle, il serait également intéressant de posséder de meilleures connaissances en mathématiques et en intelligence artificielle. Cela permettrait sûrement d'utiliser d'autres technologies qui pourraient s'avérer plus efficaces pour résoudre cette problématique.

Finalement, sur un tout autre aspect du projet, il serait également intéressant d'essayer d'utiliser une autre bibliothèque graphique. En effet, la bibliothèque Tkinter a été utilisée à son maximum. Des plus-values telles qu'un affichage dynamique ou une interface graphique encore plus moderne pourraient donc être apportées. Il est possible de réaliser des designs à l'aide de différents logiciels tels que InDesign ou Photoshop et de les animer ensuite avec des bibliothèques plus complètes telles que Pygame.

Conclusion

Pour conclure, le projet est arrivé à son terme en ayant répondu à la problématique. À savoir, la réalisation d'un outil simple d'utilisation avec une interface conviviale permettant l'identification de faune sauvage, qui utilise le deep learning.

La réalisation d'un projet de groupe est importante car il forme à de nombreuses choses. Tout d'abord, il améliore l'autonomie et la rigueur de ses membres. Il développe également des facultés d'adaptation et d'auto-formation. Il prépare à la vie professionnelle et à travers ce rapport et une soutenance, améliore la qualité d'expressions écrite et orale. Ce dernier point appartient aux soft skills et est devenu extrêmement important en entreprise aujourd'hui.

Ce projet a permis une approche concrète de l'intelligence artificielle par la création d'une application au travers d'un projet de groupe. Cette première approche pourrait, par exemple, être le sujet d'un master.

Annexes

Ensemble des annexes utiles à la compréhension et à l'illustration du projet

Liens pour accéder aux visualisations de réseaux de neurones

[Classification simple](#)

[Classification complexe](#)

Captures d'écran des tests avec une classification simple

```
Epoch 10/10
219/219 [=====] - 1s 4ms/step - loss: 0.9013 - accuracy: 0.6975
55/55 [=====] - 0s 2ms/step - loss: 0.9058 - accuracy: 0.7132
Pre-training accuracy: 71.3223%
```

Figure 51 - Capture d'écran du résultat de l'entraînement sur UrbanSound8K avec 10 Epochs

```
Epoch 100/100
219/219 [=====] - 1s 4ms/step - loss: 0.1945 - accuracy: 0.9328
55/55 [=====] - 0s 2ms/step - loss: 0.6629 - accuracy: 0.8552
Pre-training accuracy: 85.5180%
```

Figure 52 - Capture d'écran du résultat de l'entraînement sur UrbanSound8K avec 100 Epochs

```
Epoch 1000/1000
219/219 [=====] - 1s 3ms/step - loss: 0.0152 - accuracy: 0.9943
55/55 [=====] - 0s 2ms/step - loss: 0.6649 - accuracy: 0.9210
Pre-training accuracy: 92.1007%
```

Figure 53 - Capture d'écran du résultat de l'entraînement sur UrbanSound8K avec 1000 Epochs

Bibliothèques utilisées :

- Librosa : Utilisé pour la génération de MFCC ([Mel-frequency cepstral coefficients](#)) et de spectrogrammes.

-
- Matplotlib : Utilisé pour une visualisation des spectrogrammes, mfcc et audios.
 - NumPy / SciPy : Utilisé pour la gestion des sons convertis en tableaux de nombres entiers.
 - Scikit-learn : Utilisé pour l'analyse de données. NumPy, SciPy, Matplotlib et Scikit-learn sont compatibles entre eux de base étant donné qu'ils ont été faits sur les mêmes bases.
 - Tensorflow / Keras : Utilisé pour la partie Deep Learning du projet. Tensorflow sert purement à faire la partie Deep Learning, et Keras est un outil simple d'utilisation construit sur la base de Tensorflow 2.0, ce qui permet une bonne compatibilité.
 - Pandas : Utilisé pour la manipulation / analyse de données, et dans ce cas, la gestion des fichiers tableur type Excel, afin d'extraire les données qui s'y trouvent.
 - Playsound : Utilisé pour permettre à l'utilisateur de jouer le son sélectionné. Il a remplacé Winsound qui était uniquement compatible avec Windows, playsound étant compatible avec la plupart des OS.

Liste explicative des options de xeno-canto

- | | | |
|-----|-----------|----------------------------------------------------------------------------------------------------------------|
| -m | [filters] | Génération de la metadata |
| -dl | [filters] | Téléchargement des enregistrements |
| -d | [filters] | Suppression des enregistrements |
| -p | [num] | Nettoyer les dossiers contenant peu de lignes ou d'éléments |
| -g | [path] | génère la metadata pour les enregistrements présents dans le path.
Par défaut dans le dossier dataset/audio |

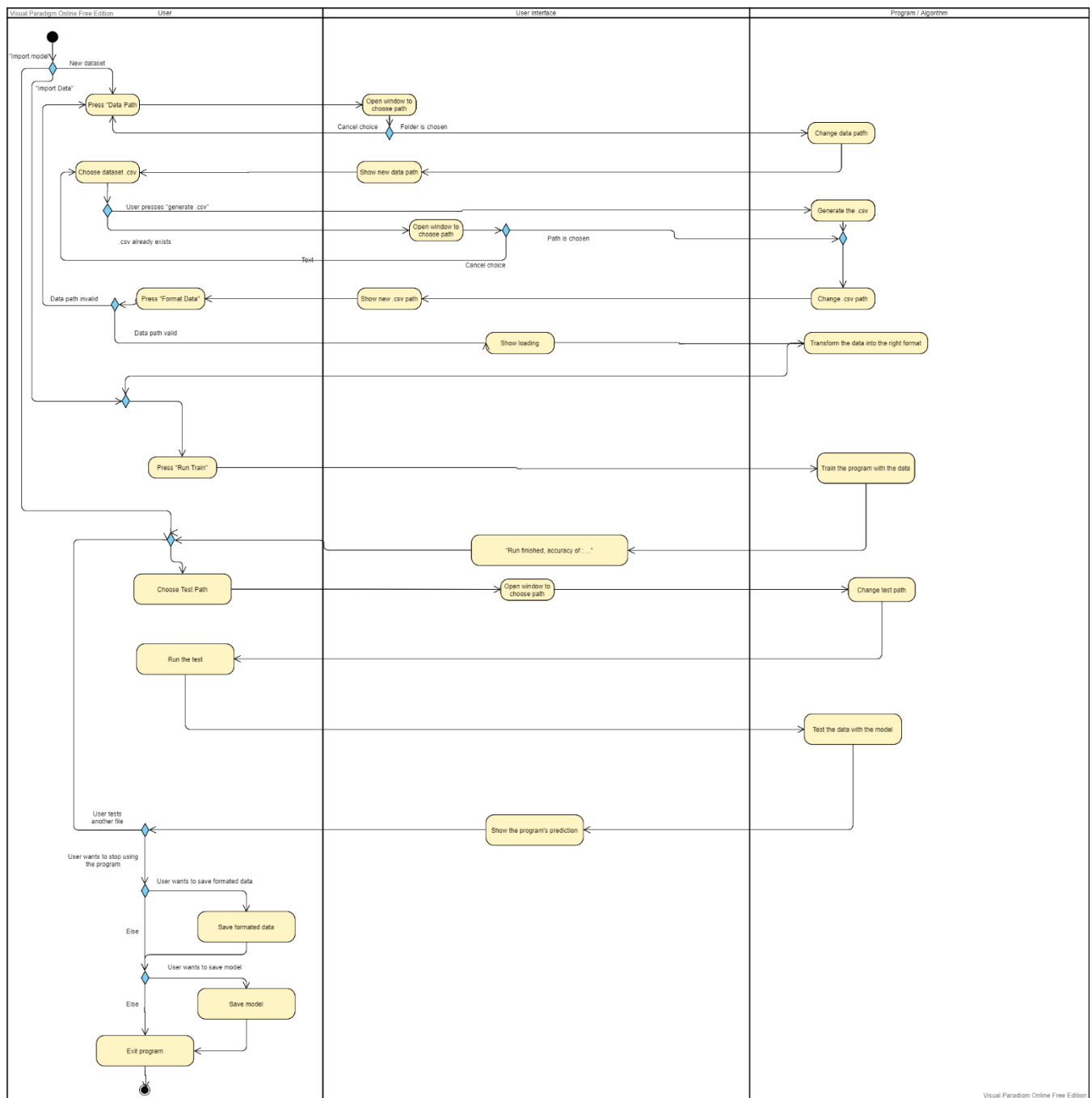


Figure 54 - diagramme d'activité UML de ML Sound

Table des illustrations

Figure 1 - courbe précision/recall sur la recherche de cris de chauves-souris	5
Figure 2 - Schéma explicatif des calculs de précision et de recall	5
Figure 3 - Diagramme en bâton de la précision de différents programmes.....	6
Figure 4 - Première version de UI du projet, implémentation de l'algorithme de Bat Détective	9
Figure 5 - Proportion entre les enregistrements mono et stéréo dans le dataset UrbanSound8K	10
Figure 6 - Proportion entre les différents sample rates présents dans le dataset UrbanSound8K	11
Figure 7 - Proportion entre les différents bit depth présents dans le dataset UrbanSound8K	11
Figure 8 – Représentation d'un modèle informatique de neurone, il existe un noyau, des dendrites (entrées), un poids associé à chaque entrée et des axones (sorties des neurones).	13
Figure 9 - Exemple très simple du fonctionnement d'un neurone	13
Figure 10 - Capture d'écran d'une classification simple	14
Figure 11 - Capture d'écran d'une classification complexe.....	14
Figure 12 - Algorithme d'obtention du MFCC à partir d'un audio	20
Figure 13 - Exemple d'enregistrement audio nettoyé de ses sons parasites.....	21
Figure 14 - Nouveaux fichiers créés lors du formatage des données.....	23

Figure 15 - Appel de fonction permettant l'obtention d'un jeu d'entrainement et un second de tests	24
Figure 16 - Création d'un modèle simple	24
Figure 17 - Compilation du modèle avec l'optimizer Adam	24
Figure 18 - Entrainement du modèle avec les enregistrements, les labels et 1000 epochs .	25
Figure 19 - Application concrète de la fonction softmax.....	27
Figure 20 - Fonction Softmax.....	27
Figure 21 - Application de Flatten à un Tableau 2D de dimension (50, 43)	27
Figure 22 - Liste des classes.....	29
Figure 23 - Barre de chargement utilisée par le programme	30
Figure 24 - Bouton pour sélectionner les données	30
Figure 25 - Bouton pour sélectionner le fichier .csv	30
Figure 26 - Boutons pour afficher l'aide	31
Figure 27 - Bouton pour formater les données	31
Figure 28 - Bouton qui lance l'entraînement du modèle	31
Figure 29 - Boutons pour sélectionner le fichier à tester et lancer la prédiction.....	32
Figure 30 - Bouton pour générer le fichier .csv.....	32
Figure 31 - Boutons pour importer les données formatées / modèle	32
Figure 32 - Boutons pour choisir le nombre d'epochs manuellement, ou automatiquement	33
Figure 33 - Boutons pour choisir la méthode de formatage	33

Figure 34 - Boutons pour changer les valeurs de ratio et RS.....	33
Figure 35 - Boutons pour sauvegarder les données formatés / modèle.....	34
Figure 36 - Boutons pour entendre / visualiser le fichier sonore.....	34
Figure 37 - Exemple d'interface graphique Tkinter	34
Figure 38 - Interface graphique de Kaléidoscope Pro	35
Figure 39 - Graphique final de l'application ML Sound	36
Figure 40 - Début du diagramme d'activité	38
Figure 41 - Partie utilisateur concernant le choix du fichier .csv, extrait du diagramme d'activité	39
Figure 42 - Zoom sur la partie prédiction du programme, extrait du diagramme d'activité	39
Figure 43 - Zoom sur la partie exportation du programme, extrait du diagramme d'activité	40
Figure 44 - Trello du projet	42
Figure 45 - Site internet du projet	43
Figure 46 - Documentation du programme	44
Figure 47 - Propriétés du dossier du programme	45
Figure 48 - Propriétés du .exe permettant l'installation du programme.....	45
Figure 49 - Capture d'écran du résultat de l'entraînement sur UrbanSound8K avec 10 Epochs	49
Figure 50 - Capture d'écran du résultat de l'entraînement sur UrbanSound8K avec 100 Epochs	49

Figure 51 - Capture d'écran du résultat de l'entraînement sur UrbanSound8K avec 1000 Epochs	49
Figure 52 - diagramme d'activité UML de ML Sound	51

Sitographie

Ensemble des sites internet utilisés pour réaliser ce rapport

- [1] Guide pour la protection des chauves-souris lors de la rénovation de bâtiments. (1992, mai). [www.ville-ge.ch. http://www.ville-ge.ch/mhng/cco/fileadmin/mhn/cco/documents/pdf/rapport_cco_batiment_renovier_guid_e.pdf](http://www.ville-ge.ch/mhng/cco/fileadmin/mhn/cco/documents/pdf/rapport_cco_batiment_renovier_guid_e.pdf)
- [2] [Bat Détective](#) – Deep learning tools for bat acoustic signal detection, Computational Biology, 2018
- [3] [Bird Species Identification](#) using Deep Learning, International Journal of Engineering Research & Technology, 2019
- [4] [Sound Classification using Deep Learning](#), Mike Smales, 2019
- [5] Base de donnée [UrbanSound8K](#)
- [6] <http://playground.tensorflow.org/>
- [7] Site référençant des enregistrements de chauves-souris : [Batcalls](#)
- [8] Bibliothèque en ligne d'enregistrements de chauves-souris : [Bats.org](#)
- [9] Présentation d'une librairie référençant des sons de chauves-souris : [Reference](#)
- [10] Site référençant différentes bases de données dans le but de réaliser des compétitions en lien avec le deep learning : [Kaggle](#)
- [11] xeno-canto::Sharing bird sounds from around the world. (2021). xeno-canto. <https://www.xeno-canto.org/>
- [12] xeno-canto, présentation des différentes options de l'API : <https://www.xeno-canto.org/help/search>
- [13] Premier tutoriel Tensorflow : [tensorflow.org/tutorials](https://www.tensorflow.org/tutorials)

[14] Description de optimizer Adams : towardsdatascience.com/adam

[15] [Documentation](#) du programme ML Sound

[16] [Github](#) du projet

Résumé / Abstract

Résumé du projet avec mots-clés

Résumé :

Il existe dans la nature de nombreux bruits ambiants, mais peu d'entre nous sont capables de lier tous ces sons à leur source. Néanmoins, les naturalistes cherchent toujours à perfectionner leurs technologies afin de pouvoir comprendre les animaux qui les entourent, que ce soit pour le loisir, ou pour des aménagements dans des zones où résident des animaux, parfois protégés. Comment faire afin de pouvoir prédire de manière efficace et rapide les espèces animales environnantes ? C'est sur cette question que nous nous sommes penchés, en particulier sur les oiseaux et les chauves-souris, et nous présentons nos résultats dans ce document.

Abstract:

There are lots of noises surrounding us in the wild, but few of us can accurately identify the source of each sound. However, naturalists are always looking for ways to perfect their technologies to better understand the animals surrounding them, either for leisure, or for more serious tasks such as building in an area where animals, sometimes endangered, can live. How could we predict in a fast and efficient way, the animal species around us? It's this very question that motivated this work, especially on bats and birds, and we are showing our answer to it in this document.

Mots clés / Keywords :

Deep Learning, Chauves-souris (Bats), Oiseaux (Birds), Interface Utilisateur (User Interface),
Projet L3