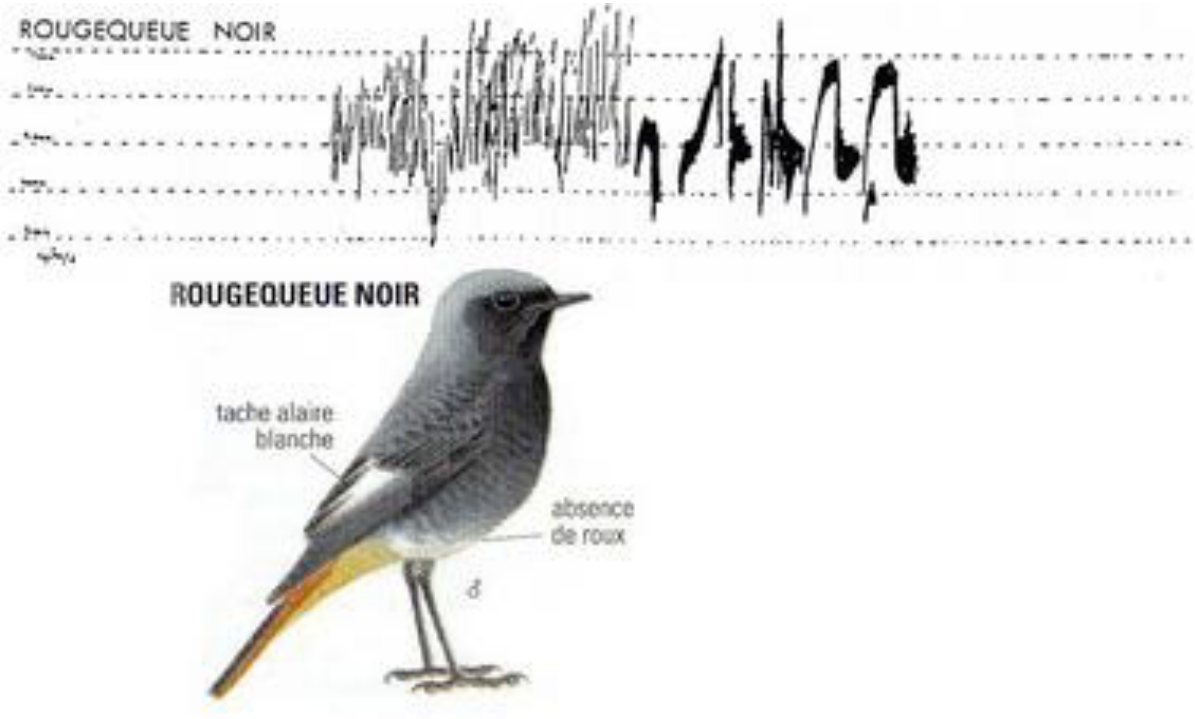
RAPPORT PLDAC

Détection sons d'oiseaux



CHANEMOUGAM Viniya

LUYAKANA Jordan

Table des Matières

Introduction

Dans le cadre du projet PLDAC, nous allons travaillé sur la détection de sons d'oiseaux (inspiré d'un concours 2017). Cela est une des tâches les plus importantes de la surveillance automatique de la faune. Nous devons élaborer un système d'apprentissage automatique qui selon un enregistrement d'audio donné rend un résultat binaire sur l'absence ou la présence d'un son d'oiseaux. Inspiré par différents modèles d'apprentissage proposés, nous allons tester les différentes méthodes avec des données d'entrées pré-traités sous différentes conditions. L'angle principale sur laquelle nous avons décidé de travailler notre sujet est la suivante: L'implémentation de code de nouveaux systèmes et amélioration des systèmes existents pour la detection de sons d'oiseaux dans les audios.

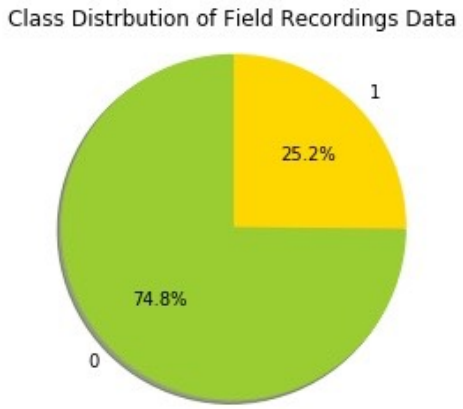
Pour ce projet nous avons décidé d'utiliser le dataset «ff1010bird» pour l'apprentissage qui était donnée sur le site du concours .

Notre objectif final serait de construire un reséau convolutif qui nous permettrait de détecter la présence d' un oiseau , mais avant de coder ce réseau , nous avons décidé d' utiliser d'autres algorithmes comme le perceptron par exemple pour montrer que les réseaux convolutifs sont les plus efficaces actuellement pour effectuer une classification audio .

Etude des données d'apprentissage

Ce dataset contient 7690 audio , il est accompagné d' un fichier CSV qui contient trois colonnes , la première colonne correspond aux id des audios par exemple pour l'audio 55.wav , l'item de cette audio sera 55 . La deuxième colonne on a simplement id du dataset dans notre cas l'id c'est le nom de notre dataset et pour terminer une troisième colonne nommé « hasbird » qui nous renseigne s' il y a un oiseau ou pas dans la séquence de l'audio , deux valeurs sont possibles , soit 1 s' il y en a bien un soit 0 s' il y en a pas du tout .

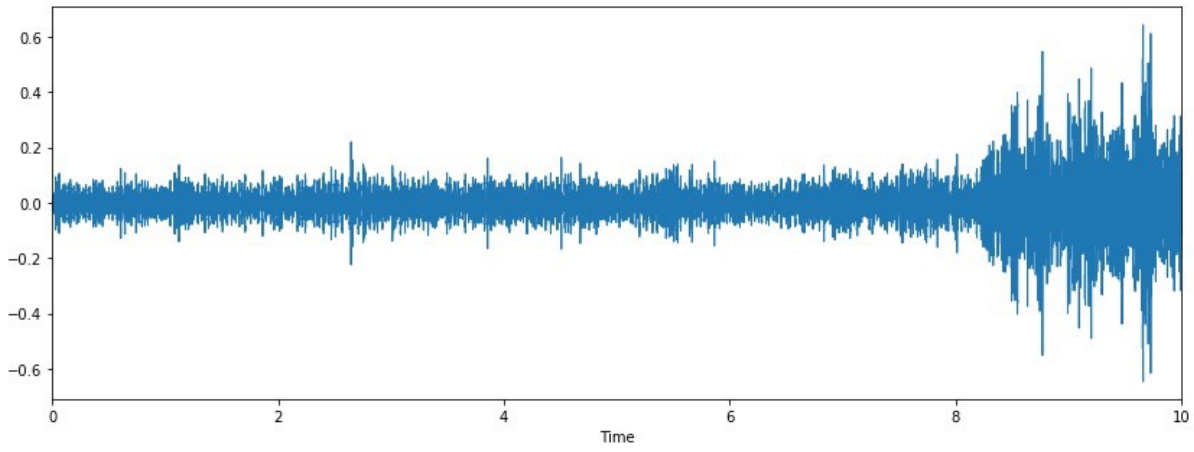
Distribution de Classe

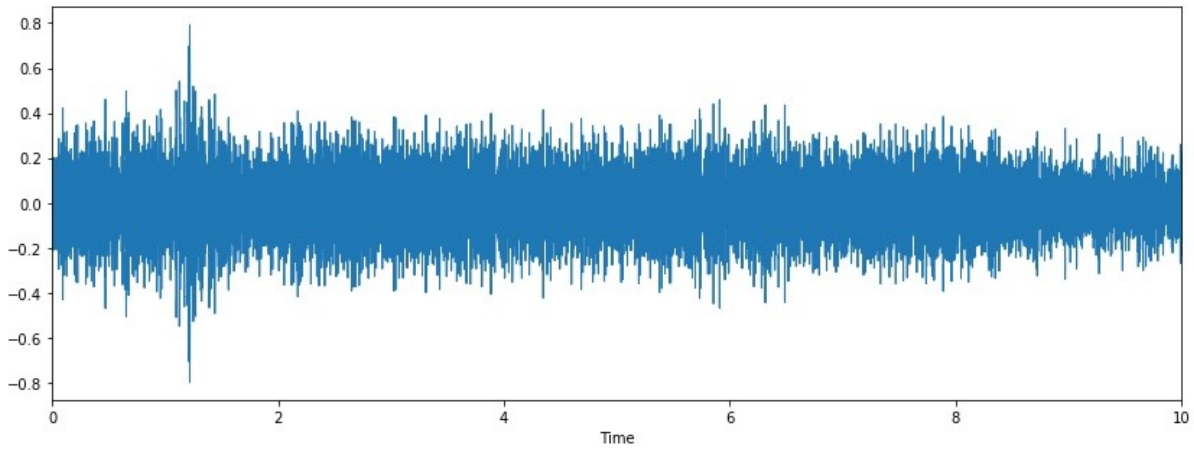


Nous avons utilisé la bibliothèque Librosa pour extraire les différentes formes de l'audio .

Nous avons pris un audio avec son d'oiseau et un autre sans son d'oiseau de notre donnée d'apprentissage et nous l'avons afficher sans aucune transformation on l'a laissé en format brut .

La première est la représentation d'une audio avec son, la deuxième sans:





Sous cette forme il est compliqué de faire de bonnes analyses , ça nous permet pas d'identifier et de savoir si le bruit est celui d' un oiseau ou d' une autre chose.

Il est donc nécessaire, comme tout autre système d'apprentissage automatique de reconnaissance vocale, d'extraire les charactéristiques. Nous avons ici testé les 3 méthodes suivantes:

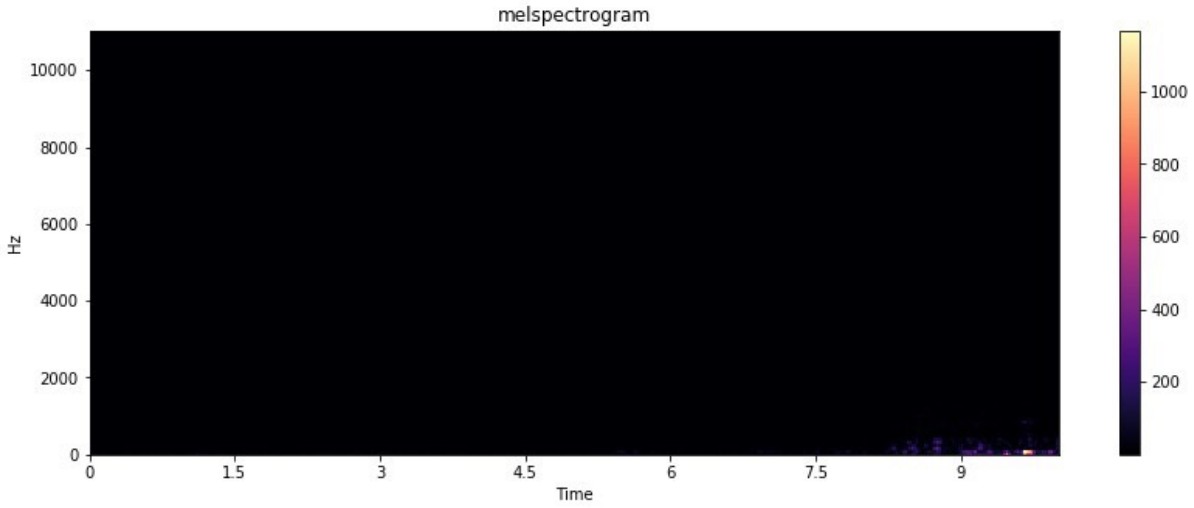
- Mel Spectogramme:
  - C'est une représentation temps-fréquence d'un son.Il est échantillonné en un certain nombre de points autour de temps ti et de fréquences fj équidistants (sur une échelle de fréquence Mel).

- Mfcc (Mel-frequency cepstral coefficients calculé de la manière suivante:
  - On prends la transformée de Fourier du signal
  - Ensuite on cartographie les puissances du spectre obtenues ci-dessus sur l'échelle mel
  - On prends les logs des puissances de chaque fréquence mel.
  - On effectue la transformation en cosinus discrète de la liste des "log puissances mel", comme s'il s'agissait d'un signal.
  - Les MFCC sont les amplitudes du spectre résultant

Cependant, les mfcc ne sont pas robustes lorsque il y a une importante présence de bruit / sons non intéressantes pour notre reconnaissance dans l'audio.

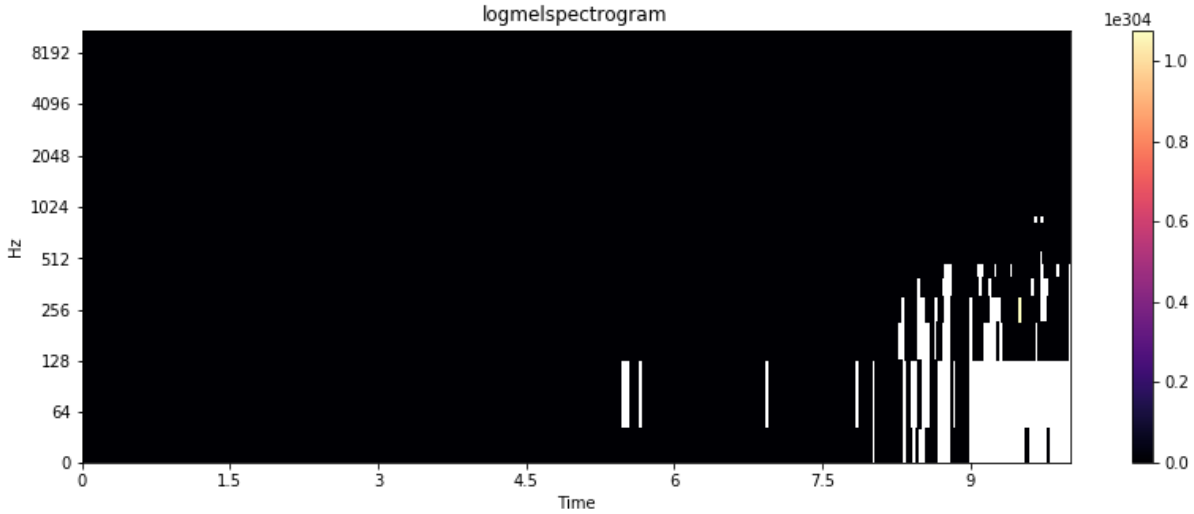
- Stft (Short Time Fourier Transformation)

Voici la représentation de notre audio en melspectrogramme :

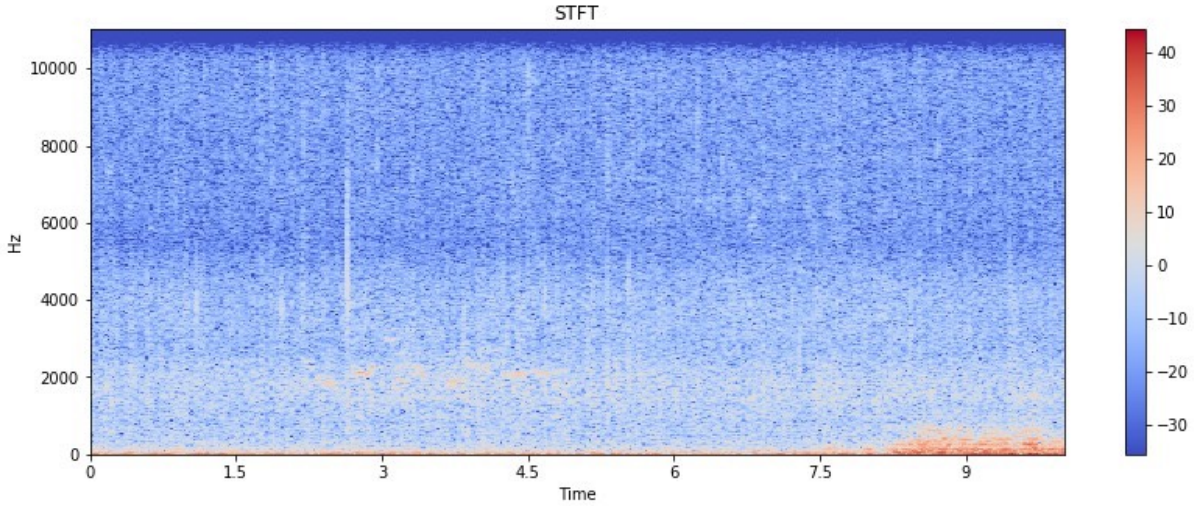


C'est beaucoup plus intuitif et on remarque par exemple dans cette exemple qu’il y a un son élève que vers la fin de la séquence .

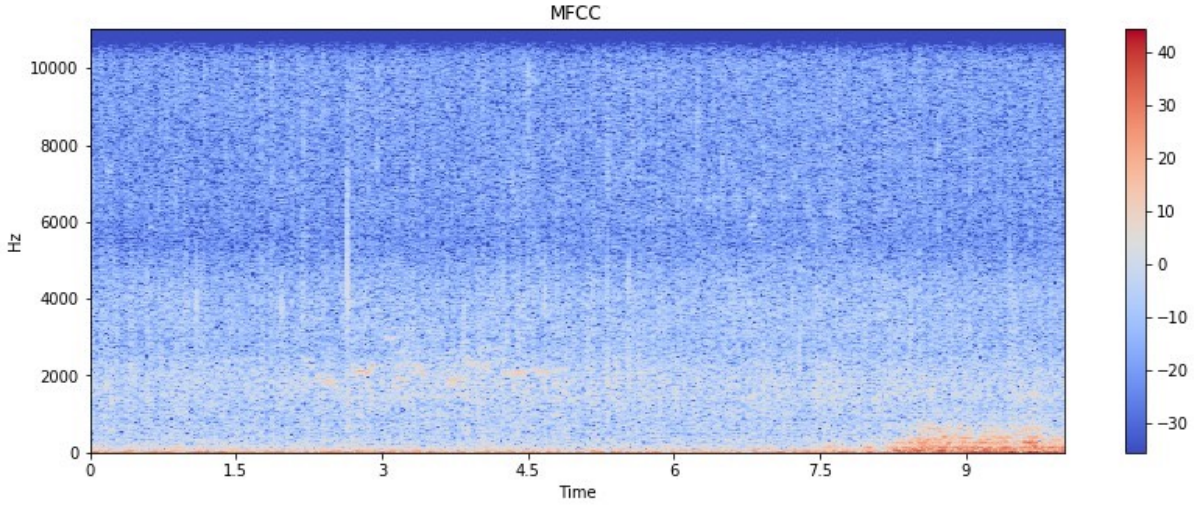
Voici notre représentation en log-melspectogramme :



Voici une autre représentation de notre audio mais cette fois-si en STFT :



Et notre dernière en MFCC :

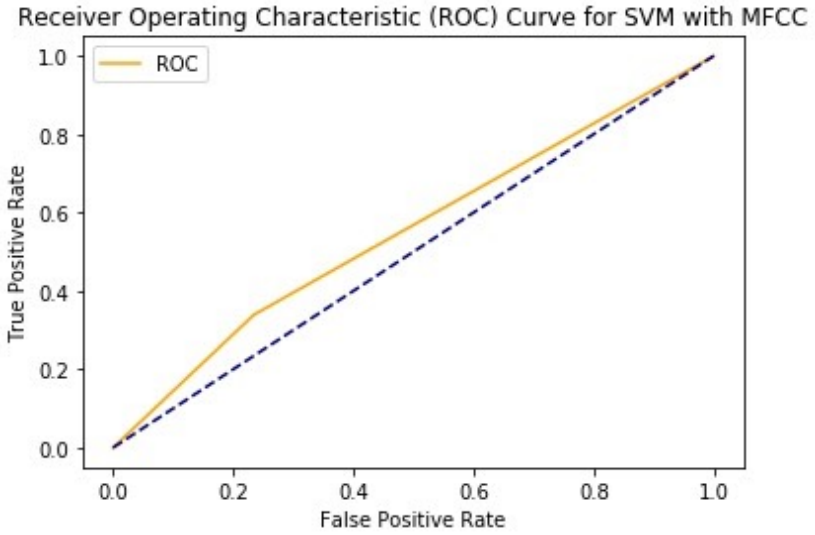


Ces différentes extractions nous permettent de mieux voir et surtout sont beaucoup plus intuitifs .

Pour la suite du projet nous utiliserons donc quatres entrées , nos données brutes, nos données en mel spectrogramme , des données en STFT et pour terminer en MFCC .

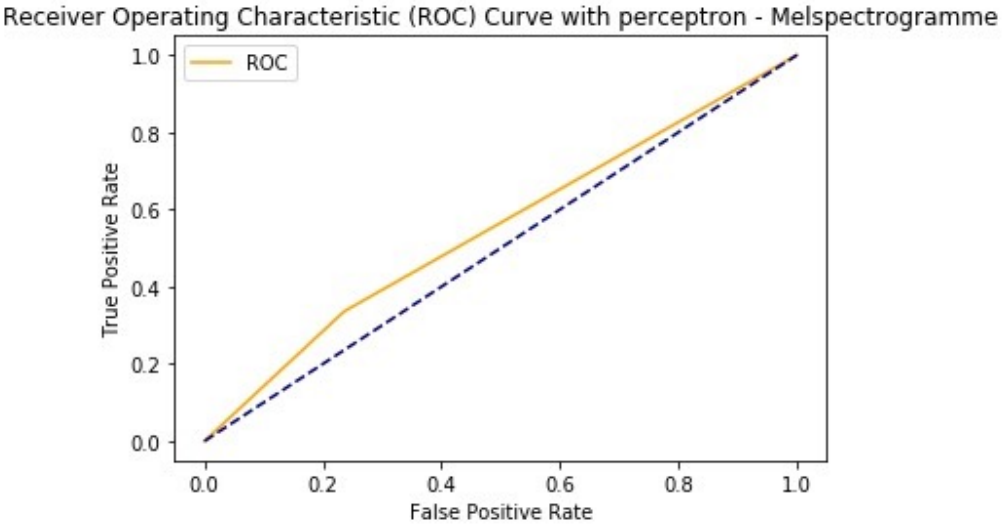
Méthode d'évaluation

Pour évaluer nos différents modèles, nous avons utilisé l'évaluation AUC-ROC. Cette dernière est la mesure de sélection du modèle pour le problème de classification binaires et multi-classes. ROC est une courbe de probabilité pour différentes classes. ROC nous dit à quel point le modèle est bon pour distinguer les classes données.



Nos Résultats :

Nous avons testé nos données sur un perceptron , un svm et un mlp , avec des données différentes (selon les différentes extractions qu' on a vu plus haut) ,



voici un petit tableau qui récapitule nos résultats avec une validation croisée :

	Mel Spectogramme	**Mfcc**	**Stft**
Perceptron	67 %	64 %	64 %
SVM	76 %	75 %	75 %
MLP	77 %	77 %	75 %

Ci-dessous, quelques exemples de score AUC de nos modèles:

On peut voir qu'un perceptron est moins efficace qu'un SVM et un MLP .
Ces résultats nous ont permis de voir la limite de certains models dans un problème de classification audio.

La prochaine étape consiste donc à coder notre premier CNN et de comparer les résultats avec nos premiers tests .

Réseau de neurone Convolutif

Après avoir testé nos données sur différents algorithmes d'apprentissages de bases, nous avons procédé à tester sur un réseau de neurone convolutif.

Pour commencer, nous avons coder notre CNN utilisant la librairie Keras. Nous avons utilisé le modèle "Sequential", qui nous permettra de construire notre modèle en ajoutant une couche à la fois. Nous avons fais une séquence de 4 convolutions comme montré sur le tableau ci-dessous. Les deux premières couches avec une filtre de taille 3*3, ensuite une taille de 3*1. Ensuite 3 couches denses. Nous avons utilisé la fonction d'activation Relu.

Un point à ne pas oublier lors de l'apprentissage est le fait de ne pas sur-apprendre, car nos données d'entrées sont diversifiés. Pour palier ce problème de sur-apprentissage, nous avons utilisé un taux de 50% de dropout sur les 3 dernières couches.

Nous avons entrainé notre CNN sur des Mel Spectogrammes. Les mel-Spectogrammes ont été construits avec les données "Field Recordings". Car les mfcc, et les stft sont moins performants avec le CNN.

Avec ce réseau, nous avons obtenu un score de 81%.

Cependant, notre prochaine étape est d'améliorer ce score, on effectuant des pré-traitements sur nos données d'entrées: Augmentation de données.
Pour commencer, nous pouvons faire des translations d'audios: Découper les morceaux d'audios et les mélanger de sorte qu'on a nouvelle audio (un nouveau exemple donc) suffisamment différent de ce qu'on avait en initial. Ce dernier, permettra donc à notre modèle d'être moins sensible aux endroits aux lesquelles se situent le son de l'oiseau dans l'audio.