Une image contenant texte

Description générée automatiquement**Faunascan**

**Projet PING n°47**

**2023-2024**

Une image contenant logo, Emblème, Marque, symbole

Description générée automatiquement

**Dossier de documentation**

**Membres :**

**ADEROMOU** Léonilde Ariel Oluwafèmi *– BDTN – FI*

**FAYE** Noroudine*– ISN – FI*

**GAGNON** Gasyd Hârif *– BDTN – FI*

**GUEYE** Mouhamed *– ISN – FI*

**MADJIDI** Zaénoul A. Aremou Erin *– CERT – FI*

**NDIAYE** Mouhamadou Fadel *– BDTN - FA*

**PATRY** Hugo *– BDTN – FA*

Table des matières

[Introduction 3](#_Toc2041684238)

[1 : Hardware 4](#_Toc537827746)

[2 : Récupération sonore 5](#_Toc1538754170)

[3 : Traitement des sons 6](#_Toc1568937917)

[4 : Développement d’une interface web 10](#_Toc347420933)

# Introduction

Ce projet a été réalisé dans le cadre du Projet Ingénieur (PING) de 3ème année de cycle ingénieur à l’ESIGELEC de Rouen. Pendant trois mois, notre équipe a travaillé sur le projet numéro 47 : Faunascan.

Faunascan s’inclut dans le cadre de l’étude de l’impact du réchauffement climatique sur la biodiversité des espaces publics mais également des jardins des particuliers, dans le but d’effectuer un comptage des espèces animales.

L'objectif principal de ce projet, tel qu’il a été conçu par le commanditaire M. Pluvinage, est de développer un système d'analyse de bruit basé sur l'intelligence artificielle qui peut identifier et classifier automatiquement les espèces animales en fonction de leurs vocalisations.

• Identification des Espèces : Le cœur du projet consiste à utiliser des enregistrements audios pour identifier les espèces animales présentes dans un environnement donné. Cela peut inclure des oiseaux, des insectes, des amphibiens, des mammifères et d'autres créatures.

• Surveillance Continue : Le système doit être capable de fonctionner en continu, en enregistrant les sons de l'environnement et en déclenchant une analyse dès qu'il détecte un son potentiellement d'origine animale.

• Apprentissage Automatique : L'IA utilisée pour l'identification des espèces doit être capable d'apprendre et de s'améliorer au fil du temps à mesure que de nouvelles données sont collectées.

• Interface Utilisateur Conviviale : Le système doit être accompagné d'une interface utilisateur conviviale qui permet aux utilisateurs de visualiser et d'interpréter les résultats de l'analyse, y compris des statistiques sur les espèces détectées et leur fréquence.

• Préservation de la Faune : Le projet vise à contribuer à la préservation de la faune en aidant les chercheurs, les écologistes et les amateurs de plein air à mieux comprendre la biodiversité locale.

# 1 : Hardware

Etudiant attitré à cette partie : Hugo PATRY

Trois équipements ont été utilisé pour ce projet : Un Raspberry Pi, un micro USB et un petit écran.

L’utilisation du Raspberry Pi avait été fortement encouragé dès la présentation du projet par le commanditaire. Et effectivement, il nous a semblé être la technologie répondant le plus à nos besoins. La capacité d’exécuter du code est évidemment obligatoire, le code du projet étant réalisé en Python. Qui dit Python et programmation, d’autant plus pour du Machine Learning, il faut un appareil ayant une certaine ram. Un stockage est également nécessaire afin de pouvoir enregistrer les sons captés par l’appareil. Et enfin, notre boîtier doit être simplement transportable.

En résumé, le Raspberry Pi répond parfaitement à ce besoin : Un appareil pouvant exécuter du code, ayant du stockage et de la ram, tout en étant simple à transporter.

L’autre avantage du Raspberry Pi est la multitude de périphérique compatible avec cette technologie. Ainsi, il n’a pas été compliqué de trouver un très bon micro sur les différents sites marchands. Même si au début du projet, l’utilisation de quel type de micro USB avait fait surface. Mes recherches indiquaient que faire soit même le microphone avec un microprocesseur pourrait offrir une meilleure qualité audio. Cependant différents projets disponibles sur l’internet ont utilisé un microphone USB dont le projet « BirdNet », projet au même objectif que le nôtre, mais spécialisée dans les espèces d’oiseaux. Ainsi, les microphones USB ayant fait leurs preuves, peu chère, et prêt à l’emploi, j’ai fait le choix d’utiliser cette technologie.

L’autre périphérique utilisé est un petit écran. Cet écran n’était pas prévu à l’origine. Cependant, cela nous a semblé être un bon ajout pour une personne qui souhaiterai utiliser l’appareil sans avoir à connecter à un ordinateur. En effet, nous sommes partis du principe que notre appareil doit être utilisable par tous, or certaines personnes n’ont pas la possession ou la connaissance afin de brancher un Raspberry Pi sur leurs ordinateurs. C’est pour cela qu’un petit écran tactile a été ajouté.

# 2 : Récupération sonore

Etudiant attitré à cette partie : Hugo PATRY

Dans le fonctionnement de Faunascan, la première brique utilisée et nécessaire est la récupération du son. Sans son, il ne peut pas y avoir d’analyse par l’équipe Data. Et sans analyse, il ne peut pas y avoir d’informations accessible à l’utilisateur sur la biodiversité de son environnement.

Dans un premier temps, j’étais parti sur un enregistrement en continu par l’appareil et la génération d’un fichier audio de deux heures. Qui par la suite allait être traité afin d’en sortir les sons sortant du lot, que l’on peut supposer animal et non parasite. De plus ceci aurait permis d’effectuer qu’une seule tâche de prétraitement afin d’améliorer la qualité sonore de l’enregistrement, sauvegardant ainsi de la puissance computationnelle.

Finalement, après des tests une fois le matériel reçu, j’ai pu me rendre compte que la qualité sonore produit par le microphone USB était excellente, encore meilleur que prévu. Aucun bruit parasite électronique, le son très net. L’audio brut en lui-même déjà a permis lors de tests de faire la reconnaissance d’espèce de chien. J’ai ainsi changé le code afin de ne plus avoir un gros enregistrement, le prétraitement pour améliorer la qualité n’étant plus nécessaire. De plus, ceci permet d’économiser du stockage.

Le code à présent pour l’enregistrement est composé de deux fonctions.

La première est « findpalier », qui comme son nom indique, à pour objectif de trouver un pallier. Plus précisément, le pallier de décibel à partir duquel notre appareil peut enregistrer le son. La fonction va tourner pendant une minute et enregistrer les décibels captés. Une fois la minute passée, la moyenne des décibels captés est calculée. A cette moyenne, on ajoute une marge de sécurité de 20% afin que ce soient des bruits vraiment majeurs qui soient bien enregistrés. Cette fonction est appelée au début de la fonction principale.

Cette fonction principale est « recordaudio ». Elle écoute en continue le son environnant, et si le son capté est supérieur à ce que la fonction pallier a défini, alors l’enregistrement commence. L’enregistrement continuera tant que pendant trois secondes d’affilées le son n’est pas inférieur à ce qui a été défini par le pallier. Une fois l’enregistrement terminée, le fichier audio (en format .wav) est créé dans un dossier spécifique. Afin d’effectuer un meilleur suivi, l’heure de l’enregistrement est utilisée afin de savoir quand est-ce que l’animal a été captée. Une fois tout cela terminée, nous retournons à la première étape en attente de capter un nouveau son.

# 3 : Traitement des sons

Etudiants attitrés à cette partie : ADEROMOU Ariel & GAGNON Gasyd

Pour le traitement des sons, nous avons décidé d’utiliser le modèle **HuBERT** développé par facebook research. Ce modèle est une nouvelle méthode développée pour apprendre de manière auto-supervisée des représentations de la parole. Le but principal est d'améliorer la reconnaissance et la compréhension de la parole de manière continue, en imitant la façon dont les bébés apprennent la langue en écoutant et en interagissant avec les autres. Cela nécessite non seulement d'analyser les mots prononcés, mais aussi de comprendre divers indices non verbaux tels que l'identité du locuteur, les émotions, les hésitations, les interruptions et de distinguer les bruits de fond tels que les rires, la toux, le claquement des lèvres, les véhicules en arrière-plan ou le chant des oiseaux. Le modèle se distingue par sa capacité à modéliser ces informations lexicales et non lexicales riches dans l'audio. Il égale ou dépasse les approches de pointe en termes d'apprentissage de représentations de la parole pour la reconnaissance, la génération et la compression de la parole.

Une image contenant texte, capture d’écran, diagramme, Police

Description générée automatiquement

Figure 1 : Architecture du modèle HuBERT

Dans cette architecture, il y a plusieurs composants clés qui traitent l'audio à différentes étapes :

**CNN Encoder :** Cette couche est responsable de l'extraction des caractéristiques de bas niveau à partir du signal audio brut. Ces réseaux de neurones convolutifs (CNN) sont utilisés pour extraire des représentations de faible niveau qui peuvent inclure, mais ne se limitent pas nécessairement aux **Mel-Frequency Cepstral Coefficients (MFCC)**.

**Transformer :** Le transformateur est une série de couches qui prennent les caractéristiques extraites par le CNN Encoder et les traitent à travers des mécanismes d'attention pour capturer des dépendances à long terme et des relations complexes dans les données.

**Acoustic Unit Discovery System :** Ce n'est pas une couche de **HuBERT** en soi, mais plutôt un système externe qui peut être utilisé pour découvrir des unités acoustiques dans le signal traité. Par exemple, l'application de K-means sur des MFCC pour la segmentation ou la classification des unités sonores.

Inspiré par l’architecture du modèle **HuBERT**, le prétraitement et l'extraction des caractéristiques audio ont été réalisé en utilisant les couches **CNN Encoder** et **Transformer** du modèle. Les vecteurs audio résultants sont tridimensionnels, comprenant :

- Une dimension **Batch**, qui représente le nombre d'exemples audio traités simultanément lors d'une opération. Cela permet de traiter plusieurs échantillons audios en parallèle, augmentant ainsi l'efficacité du traitement.

- Une dimension **Temporelle**, qui représente la séquence temporelle des caractéristiques audio. Chaque point dans cette dimension correspond à un moment particulier dans le signal audio, permettant au modèle de suivre l'évolution des caractéristiques audio au fil du temps.

- Une dimension de **Caractéristiques**, qui contient les informations représentatives des audios traités. Cela inclut les caractéristiques telles que les coefficients cepstraux de fréquence de Mél (MFCC) et d'autres attributs pertinents qui capturent les nuances et les détails importants du signal audio.

Il a donc fallu utiliser le **PCA**, ou Analyse en Composantes Principales (en anglais, Principal Component Analysis), qui est une technique statistique utilisée pour simplifier la complexité des espaces de données multidimensionnelles tout en préservant autant que possible leur variance. Cela se fait en transformant les données vers un nouveau système de coordonnées où les plus grandes variances par n'importe quelle projection des données viennent à se situer sur les premiers axes (appelés les premières composantes principales). Les axes suivants (ou composantes) sont orthogonaux (perpendiculaires) aux précédents et ont des variances décroissantes.

L'application du PCA dans notre contexte visait principalement :

**Réduction de Dimensionnalité :** Réduire les dimensions en conservant les aspects les plus importants des données, ce qui peut être utile pour améliorer la vitesse et réduire les besoins en stockage ou mémoire pour les tâches suivantes. Notamment en supprimant la dimension **Batch.**

**Filtrage du Bruit :** Le PCA a aidé à atténuer le bruit en reconstruisant les données à partir des composantes principales qui ont tendance à capturer les signaux les plus significatifs des audios.

Les vecteurs bidimensionnels normalisés sont intégrés dans la base de données vectorielles **FAISS** (Facebook AI Similarity Search). Cette base de données permet, lorsqu'un vecteur est fourni en entrée, d'effectuer une recherche de similitude pour trouver tous les vecteurs similaires à celui d'entrée, puis de les lister par ordre croissant de distance. Pour cette recherche de similitude, la distance euclidienne est utilisée, ce qui permet de calculer la distance linéaire entre deux vecteurs. Cette métrique est particulièrement adaptée pour évaluer la similarité entre des vecteurs audio, car elle mesure la proximité dans l'espace vectoriel.

Après avoir calculé la distance entre le vecteur d'entrée et ceux de la base de données, une transformation est appliquée en utilisant la fonction exponentielle négative pour convertir ces distances en mesures de similarité pour chaque enregistrement audio. L'utilisation de cette fonction est pertinente car elle permet de pondérer (dans l’intervalle [0,1]) les similarités de manière appropriée, mettant davantage en avant les enregistrements les plus proches en termes de similarité.

Enfin, pour chaque catégorie d'animaux (chiens, chats, oiseaux), des probabilités conditionnelles sont calculées en se basant sur les probabilités de similarité des échantillons les plus proches appartenant à chaque catégorie. Cela nécessite de prendre en compte les étiquettes associées aux vecteurs les plus proches pour déterminer leur catégorie.

Pour estimer les probabilités a priori P(animal) de chaque catégorie, on peut se baser sur la fréquence de chaque catégorie dans la base de données initiale. Par exemple, si 2000 enregistrements sur un total de 5000 sont des chiens, alors P(chien) = 2000/5000.

Enfin, en appliquant le **théorème de Bayes**, on peut calculer **P(animal | données)** pour chaque catégorie. La catégorie avec la probabilité la plus élevée sera la classification attribuée à l'échantillon audio. Cette approche permet de prendre en compte à la fois les caractéristiques des données et les probabilités a priori pour effectuer une classification précise.

# 4 : Développement d’une interface web

Etudiants attitrés à cette partie : FAYE Noroudine , GUEYE Mouhamed, MADJIDI Zaénoul

Lors du développement de l’interface web du projet Faunascan, le choix des technologies pour la mise en place des interfaces utilisateur et la gestion des données côté serveur est crucial. Dans cette documentation, nous explorerons les avantages de l'utilisation de FAST API côté serveur et React JS côté client pour créer des interfaces de résultats dynamiques et performantes.

**FAST API**

1. **Rapidité et Efficacité**

FAST API est un framework web Python asynchrone qui offre une exécution rapide des requêtes HTTP. Grâce à l'utilisation de la programmation asynchrone, FAST API permet de gérer simultanément de nombreuses connexions, ce qui améliore significativement la performance et l'efficacité de l'application.

2. **Documentation Automatique**

L'un des avantages majeurs de FAST API est la génération automatique de la documentation Swagger (OpenAPI). Cette documentation interactive facilite la compréhension des endpoints de l'API, rendant le processus de développement plus transparent et collaboratif.

3. **Sécurité Intégrée**

FAST API intègre des mécanismes de sécurité tels que la validation automatique des données, la gestion des tokens JWT, et d'autres fonctionnalités de sécurité avancées. Cela garantit la robustesse et la sécurité des applications développées.

**React JS**

1. **Composants Réutilisables**

React.js est une bibliothèque JavaScript qui permet de construire des interfaces utilisateur modulaires grâce à l'utilisation de composants réutilisables. Cette modularité facilite la maintenance du code et accélère le processus de développement.

2. **Virtual DOM pour une Rendu Rapide**

React utilise un Virtual DOM, ce qui signifie que seules les parties de l'interface qui ont été modifiées sont mises à jour. Cela permet d'optimiser le rendu des pages et de garantir une expérience utilisateur fluide même avec des données dynamiques.

3. **Écosystème et Communauté Active**

React bénéficie d'un vaste écosystème de bibliothèques tierces (React Router, Redux, etc.) et d'une communauté active. Cela facilite l'intégration de fonctionnalités avancées, le dépannage, et assure une évolution constante de la bibliothèque.

**Intégration de FAST API et React JS**

En combinant FAST API et React , on bénéficie d'une stack technologique complète, permettant une communication fluide entre le serveur et le client. L'utilisation de l'API RESTful fournie par FAST API avec React permet une mise à jour dynamique de l'interface en fonction des résultats du serveur.