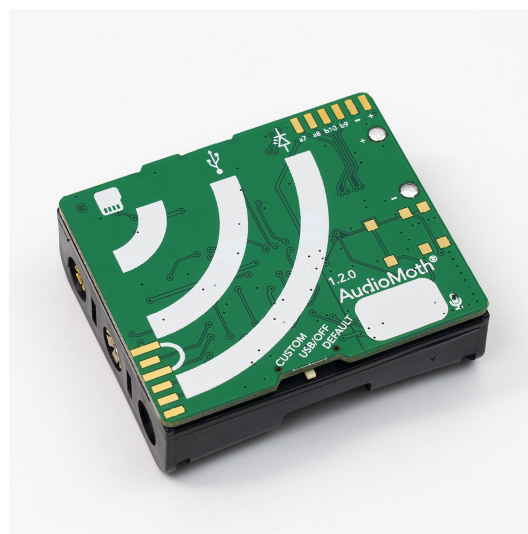


ENSTA Bretagne
2, rue François Verny
29806 BREST cedex
FRANCE
Tel +33 (0)2 98 34 88 00
www.ensta-bretagne.fr

Projet Système
Rapport
d'avancement
SOIA 2023
15 décembre 2021

Pollution sonore urbaine Implémentation d'une méthode de Machine Learning



T. ALBERT E. FERRAND H. FAUVEL

1 Préface

1.1 Abstract

The aim of this project is to build an interactive sound map of a district of Brest based on machine learning recognition methods. The different noises will be recognized and categorized in a database to establish the different sources of noise pollution. A daily noise monitoring (24h) will allow, after a statistical analysis, to determine the evolution of noise pollution during a day. More precisely, the probability of occurrence of such a noise at a given moment and its maximum intensity will be displayed.

1.2 Résumé

L'objectif de ce projet est de constituer une carte sonore interactive d'un quartier de Brest basée sur des méthodes de reconnaissance par apprentissage automatique. Les différents bruits seront reconnus et catégorisés dans une base de données afin d'établir les différentes sources de pollution sonore. Un suivi du bruit quotidien (24h) permettra, après une analyse statistique, de déterminer l'évolution des nuisances sonores lors d'une journée. Plus précisément, on affichera la probabilité d'apparition de tel bruit à un instant, et son intensité maximale.



Table des matières

1	Préface	2
1.1	Abstract	2
1.2	Résumé	2
2	Introduction	4
3	Organisation générale	5
3.1	Cahier des charges	5
3.2	Analyse du système par l'ingénierie système	7
3.2.1	Analyse externe du système	7
3.2.2	Analyse interne du système	11
3.2.3	Architecture physique du système	14
3.3	Diagramme de Gantt	15
3.4	Organisation de groupe par méthode agile	15
4	État de l'art	16
4.1	Machine Learning	16
4.1.1	Définitions	16
4.1.2	Les phases d'un projet de machine Learning	16
4.1.3	Différents types de machine learning	17
4.2	Deep Learning et Réseaux de Neurones	18
4.2.1	Réseaux de neurones artificiels: architecture et fonctionnement	18
4.2.2	Sur-apprentissage et sous-apprentissage	19
4.2.3	Les réseaux de neurones convolutifs	20
4.3	Estimation spatiale par krigeage ordinaire	21
4.3.1	Le krigeage ordinaire pour des coordonnées géographiques	21
4.3.2	Krigeage d'une fonction aléatoire sur une dimension temporelle	22
4.3.3	Modèle du variogramme	24
5	Relevé de prise audio dans un milieu urbain	26
6	Traitement des données audios par deep learning	27
6.1	Mise en forme des données	27
6.2	Exploitation des données par méthode de Deep Learning	29
6.2.1	Le réseau de neurones utilisé	29
6.2.2	Les prédictions	29
6.3	Résultats	30
6.3.1	Influence des classes choisies et de la variété des données	31
6.3.2	Influence de la structure du réseau de neurones convolutifs	32
6.3.3	Influence de la quantité de données	32
7	Visualisation de la pollution sonore urbaine par de la cartographie	34
7.1	Estimation spatiale par krigeage	34
7.2	Visualisation cartographique	35
7.3	Interface avec l'utilisateur avec une page web	35
8	Conclusion	36



2 Introduction

Le conseil de la colonie grecque de Sybaris au VI^{ème} siècle avant J.C. a ordonné le déplacement de commerçants et artisans en dehors du centre de la ville à cause du bruit excessif qu'ils produisaient. C'est le plus ancien décret connu sur les nuisances sonores. Cette ordonnance prononcée à la suite du développement des techniques artisanales de l'âge de fer nous indique que la pollution sonore est une problématique ancienne que Hippocrate avait identifiée au siècle suivant comme étant une cause des problèmes d'acouphènes et de bourdonnements. C'est seulement lorsque l'intensité sonore devient suffisamment importante ou que l'exposition à ses nuisances est prolongée que des gênes, et parfois même, des dégâts pour la santé apparaissent selon le médecin grec. La ville est un lieu caractérisant cette problématique en raison de sa forte concentration d'activités où est générée une pollution sonore intense.

Afin d'avoir un confort acoustique dans un milieu urbain, plusieurs travaux ont tenté de quantifier ce son, de le limiter, de le prévenir. L'aménagement du territoire proposé par [1] a pour but de limiter les nuisances sonores dans un milieu urbain. Encore faut-il pouvoir identifier ces sources sonores. Cette difficulté est soulevée par chaque étude entreprenant d'améliorer les conditions de vie en ville. Ces sources sont nombreuses et parfois difficiles à prévoir.

Avant d'envisager aménager le territoire, nous cherchons dans ce projet à reconnaître des sources sonores courantes, comme le bruit généré par la circulation routière, par des passants... et de les localiser pour adapter le cadre urbain de façon locale. Le projet se décompose en quatre parties principales.

La première partie consiste à réaliser des relevés dans un quartier. Ces relevés seront ensuite préalablement traités en fonction des besoins. La deuxième partie doit répondre à une préoccupation évoquée plus tôt, l'identification de la source sonore. Cette phase sera traitée à l'aide d'un algorithme de deep learning sous Python, afin d'automatiser cette tâche. La troisième partie répond à la seconde interrogation, la localisation des nuisances sonores qui peuvent a priori être hétérogènes d'une rue à l'autre. Enfin, la quatrième partie doit permettre d'informer sur le résultat obtenu. Nous choisissons de réaliser une carte pour permettre une visualisation spatiale de la pollution sonore urbaine.



3 Organisation générale

3.1 Cahier des charges

Nous concevons notre système en quatre grandes étapes : l'identification et la classification des exigences, l'analyse externe, l'analyse interne et l'architecture fonctionnelle. Nous utilisons pour cela le logiciel IBM Rhapsody. Nous allons présenter ici l'ensemble de notre réflexion, mais seuls les schémas du dossier les plus pertinents seront présentés.

Dans un premier temps, nous avons identifié les exigences. Nous devons identifier, en plus des exigences du client, toutes les exigences techniques supplémentaires indispensables au bon fonctionnement du système. Ces exigences sont classées dans différents groupes.

Enregistrements

- Le système doit permettre de collecter des sons en milieu urbain.
- Le système doit permettre de collecter des sons de manière périodique sur une certaine période de temps à préciser.
- Le système doit permettre de collecter des sons dans les fréquences audibles chez l'humain.
- Le système doit permettre d'enregistrer des sons quelques soient les conditions météorologiques.
- Le système doit faire abstraction des nuisances météorologiques (vent, pluie)
- Le système doit pouvoir stocker une certaine quantité de données à préciser.
- Le système doit recueillir les coordonnées géographiques (GPS)
- Le système doit recueillir la date d'enregistrement
- Le système doit mesurer l'intensité sonore pour un temps donné.
- Le système doit mesurer l'intensité sonore pour un temps et une fréquence donnée.

Caractérisation de la source sonore

- Le système doit identifier la source des sons enregistrés.
- Le système doit donner une fenêtre de temps pour laquelle un objet est détecté. -Le système doit détecter l'intensité sonore pour chaque source.

Analyse statistique

- Le système doit donner la fréquence d'apparition de chaque son identifié par tranche horaire
- Le système doit donner la fréquence d'apparition de chaque son identifié par tranche horaire et par site
- Le système doit donner la fréquence d'apparition de chaque son identifié par site
- Le système doit donner la fréquence d'apparition moyenne des sons identifiés, par tranche horaire
- Le système doit donner la fréquence d'apparition moyenne des sons identifiés, par site
- Le système doit donner la fréquence d'apparition moyenne des sons identifiés, par tranche horaire et par site
- Le système doit donner l'intensité sonore moyenne associée à chaque source identifiée, par tranche horaire
- Le système doit donner l'intensité sonore moyenne associée à chaque source identifiée, par tranche horaire et par site
- Le système doit donner l'intensité sonore moyenne associée à chaque source identifiée, par site
- Le système doit donner l'intensité sonore moyenne globale par tranche horaire, mais aussi d'autres descripteurs statistiques (permettant de réaliser une boîte à moustache par tranche horaire), et ce en moyenne sur un certain nombre de jours à préciser.



-Le système doit donner l'intensité sonore moyenne globale par tranche horaire et par site, mais aussi d'autres descripteurs statistiques (permettant de réaliser une boîte à moustache par tranche horaire), et ce en moyenne sur un certain nombre de jours à préciser.

-Le système doit donner l'intensité sonore moyenne globale par site, mais aussi d'autres descripteurs statistiques (permettant de réaliser une boîte à moustache par tranche horaire), et ce en moyenne sur un certain nombre de jours à préciser.

Sortie du système

-Le système doit fournir des résultats lisibles et synthétiques pour l'homme, sous forme d'une carte.

-Le système doit fournir une carte, qui pour chaque site, affiche la probabilité de présence moyenne de chaque source

-Le système doit fournir une carte, qui pour chaque site, affiche l'intensité sonore moyenne de chaque source

-Le système doit fournir une carte, qui pour chaque site, affiche la probabilité de présence moyenne d'un son

-Le système doit fournir une carte, qui pour chaque site, affiche l'intensité sonore moyenne

-Le système doit fournir une carte, qui pour chaque site et chaque tranche horaire, affiche la probabilité de présence moyenne de chaque source

-Le système doit fournir une carte, qui pour chaque site, et chaque tranche horaire, affiche l'intensité sonore moyenne de chaque source

-Le système doit fournir une carte, qui pour chaque site, et chaque tranche horaire, affiche la probabilité de présence moyenne d'un son

-Le système doit fournir une carte, qui pour chaque site, et chaque tranche horaire, affiche l'intensité sonore moyenne

-Le système doit fournir une carte, qui pour chaque tranche horaire, affiche la probabilité de présence moyenne de chaque source

-Le système doit fournir une carte, qui pour chaque tranche horaire, affiche l'intensité sonore moyenne de chaque source

-Le système doit fournir une carte, qui pour chaque tranche horaire, affiche la probabilité de présence moyenne d'un son

-Le système doit fournir une carte, qui pour chaque tranche horaire, affiche l'intensité sonore moyenne

-Le système doit avoir une interface accessible en ligne.

Législation

-Le système ne doit pas aller à l'encontre des lois sur la violation de la vie privée.

Implémentation et mise à jour

-Le système doit permettre de programmer un algorithme d'analyse statistiques.

-Le système doit pouvoir être implémenté et mis à jour en ligne par les membres de l'équipe.

-Le système doit permettre de coder l'algorithme de reconnaissance de son par deep-learning

-Le système doit être implémenté en Python.

-Le système doit pouvoir être modifié en parallèle de son utilisation.

-Le système doit permettre l'implémentation de la fonction d'apprentissage de l'algorithme de deep-learning

-Le système doit permettre l'implémentation de la fonction de test de l'algorithme de deep-learning.

-Le système doit permettre l'affichage de la fonction d'affichage.



Apprentissage

- Le système doit pouvoir être entraîné à la reconnaissance des sons avant et en parallèle de son utilisation.
- Le système doit arrêter sa phase d'entraînement après un certain temps à préciser.
- Le système doit reconnaître les sons avec une précision d'au moins une certaine valeur à définir.
- Le système doit pouvoir être entraîné à la reconnaissance de sons à partir de fichiers de sons d'une certaine durée à définir.
- Le système doit pouvoir être entraîné à la reconnaissance de sons à partir d'une bibliothèque de données prévues à cet effet.
- Le système doit pouvoir être testé pour la reconnaissance de sons à partir d'une bibliothèque de données prévues à cet effet.
- Le système doit permettre d'associer manuellement une source à un son.

Cartographie

- Le système doit afficher une carte d'un quartier de Brest
- Le système doit afficher des relevés expérimentaux sur la carte par des points
- Le système doit afficher des estimations par des rasters
- Le système doit permettre de naviguer sur la carte de façon interactive
- Le système doit permettre de visualiser la carte sur une page web

3.2 Analyse du système par l'ingénierie système

3.2.1 Analyse externe du système

Cycle de vie

L'analyse externe présentée par la suite doit permettre de bien distinguer le système de son environnement. Plusieurs phases de vie doivent d'abord être identifiées. Nous délimitons le système en quatre phases de vie: la récupération de données, le traitement des données par deep learning, le traitement des données par géostatistique et la visualisation cartographique des données recueillies et estimées par les deux méthodes précédentes.

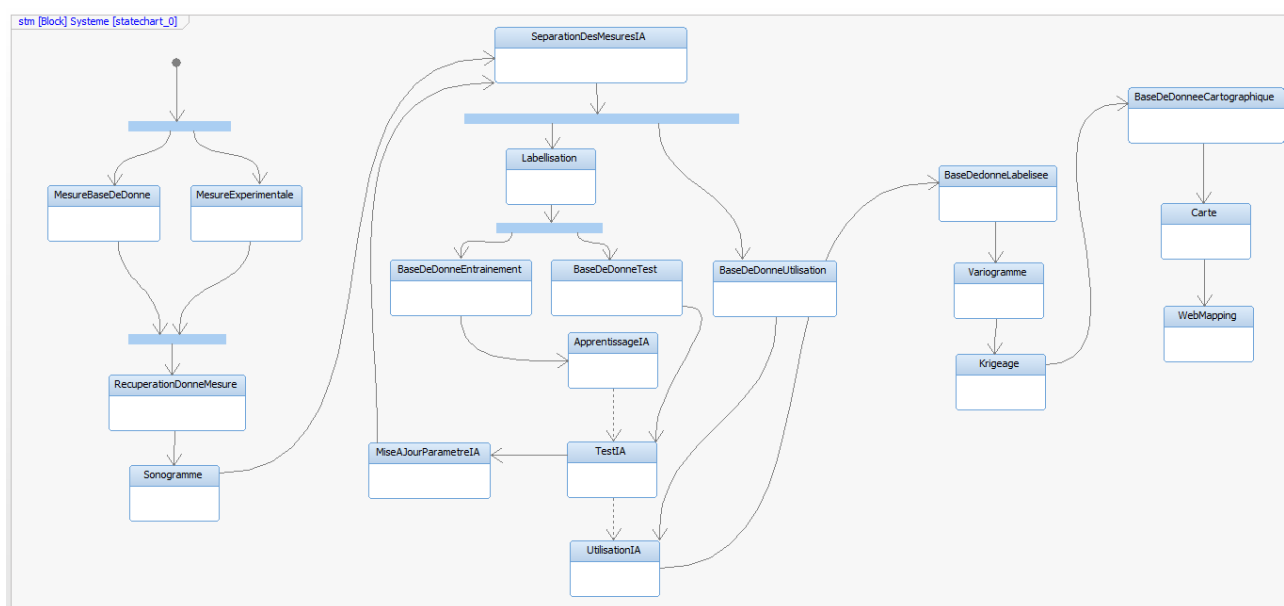


FIGURE 1 – Diagramme du cycle de vie du système séparé en quatre phases.



-Receuil des mesures audios

La première phase consiste à récupérer des mesures audios et à les stocker dans une base de données. Deux sources permettent de récupérer des enregistrements audios, par des mesures déjà recueillies et partagées. Elle sont partagées soit par notre encadrant M. CAZAU, soit sur une base de données publique. Ces fichiers audios peuvent être récupérés en parallèle, d'où les états mis en parallèle dans la figure du cycle de vie. Les mesures expérimentales sont récupérées par un capteur audiomoth, que nous présenterons avec plus de détails dans la suite de ce document.

Les enregistrements audios ne sont pas directement stockés. Plusieurs étapes sont nécessaires. Chaque fichier doit recevoir un identifiant avec les coordonnées géographiques dans le système WGS84, ainsi que la date à laquelle ces données sont récupérées. Ensuite, les fichiers se voient attribuer un sonogramme qui sera utilisé pour le traitement par deep learning.

-Traitement par intelligence artificielle

Une fois récupérées, les données sont séparées dans trois bases de données distinctes. L'une sert à entraîner l'intelligence artificielle, une seconde à vérifier les performances de l'intelligence artificielle, et la dernière contient des données qui seront labellisées de façon automatique par l'intelligence artificielle. Les deux premières bases de données ont donc besoin d'une étape supplémentaire consistant à labelliser chaque fichier. Ce label est ajouté par une interface graphique et correspond à la source sonore entendue dans l'enregistrement, comme une voiture. Les étapes suivantes se déroulent parallèlement mais avec une dépendance à sens unique, la vérification a par exemple besoin que l'étape d'entraînement de l'intelligence artificielle soit réalisée. Par ailleurs, cette étape de vérification permet d'observer les performances de l'apprentissage automatique et donc de décider s'il est nécessaire de mettre à jour les paramètres de la méthode employée.

Lorsque les paramètres de l'intelligence artificielle semblent donner des résultats satisfaisant, les fichiers de la base de données sont labellisés de façon automatique et stockés dans une nouvelle base de données pour le traitement géostatistique.

-Traitement géostatistique

Les données labellisées automatiquement permettent de connaître la source et l'intensité sonore au site de mesure. Pour estimer la pollution sonore en des zones non mesurées, les données sont utilisées pour une analyse variographique. Ce variogramme permet de déterminer l'influence qu'un site possède sur un autre. Ce variogramme sera un paramètre d'entrée dans l'algorithme de krigeage. Les données estimées en des points non mesurés sont stockées dans une base de données cartographique, avec toujours leur localisation et leur date, ainsi qu'avec leur estimation d'intensité par krigeage, et l'erreur commise par ce traitement.

-Visualisation par cartographie

La base de données est récupérée par un système d'information géographique, QGIS, correspondant à l'état carte, où cette dernière est élaborée. Lorsque cette carte est créée, elle est insérée dans une page web afin de visualiser les données provenant des enregistrements audio et des estimations.

Diagrammes en pieuvre

Dans la première phase, le système interagit avec un nombre d'acteurs important, auquel s'ajoute des dépendances. Les enregistrements audios sont récupérés par un capteur Audiomoth ou bien sur une base de données publique. Dans le premier cas, le concepteur doit placer son capteur chez un logeur, les deux étant liés par un contrat respectant la loi. De plus, les mesures sont affectées par les conditions



météorologiques, dont le vent ou la pluie. Des mesures pour limiter ces perturbations sont à prendre en compte. Enfin, ces enregistrements doivent être localisés par le moyen d'une constellation GNSS, par exemple le GPS. Dans le second cas, les mesures sont récupérées sur une base de donnée publique. Le concepteur doit vérifier son droit à récupérer et exploiter ses données en fonction de la loi. Dans les deux cas, chaque enregistrement doit pouvoir être interprété par un sonogramme. Les fonctions principales de cette première phases sont les suivantes. Leur but est de transformer les données recueillies par un capteur en une information exploitable par un algorithme de classification.

- Capteur: Récupération d'enregistrement audio par un capteur audiomoth.
- Base de donnée publique: Récupération d'enregistrement audio sur une base de données publique.
- Localisation: Récupération de la localisation d'un site de mesure.
- Sonogramme: Création d'un sonogramme.
- Législation: Respect de la loi.

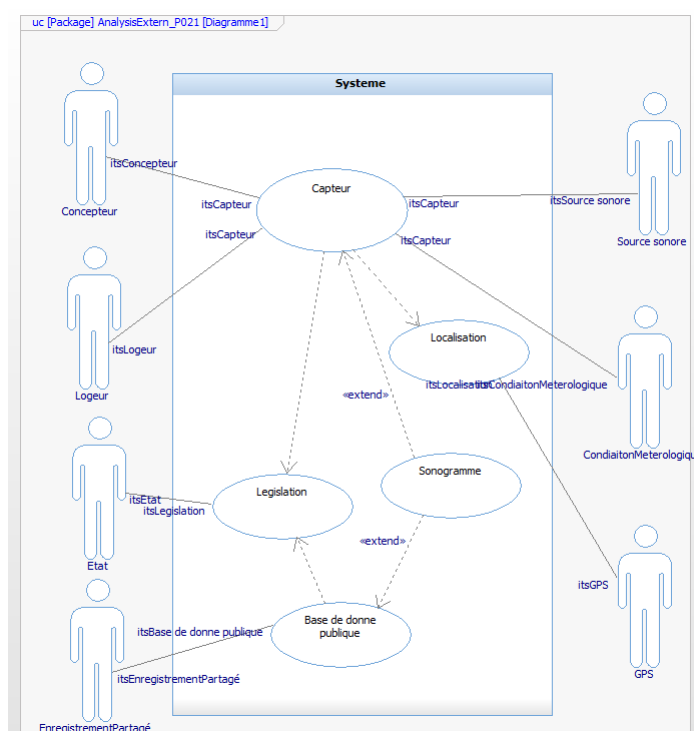


FIGURE 2 – Diagramme en pieuvre de la première phase de vie du système.

Après la création de la base de donnée contenant les sonogrammes des fichiers audios, la deuxième phase de vie correspond au traitement de ces données par intelligence artificielle. La fonction principale est l'algorithme de deep learning dont le but est de classer les fichiers audios par un label. Cette fonction principale possède de nombreuses contraintes et étapes qui seront détaillées dans la partie analyse interne et dans la partie sur le traitement des fichiers audios.



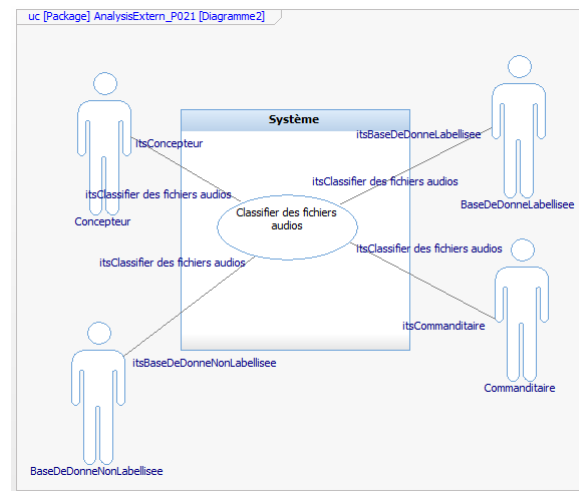


FIGURE 3 – Diagramme en pieuvre de la deuxième phase de vie du système.

Les données labellisées sont traitées par un programme Python pour obtenir le variogramme approché du modèle. Cette fonction est nécessaire pour réaliser l'estimation par krigeage. Les estimations sont stockées dans une base de données cartographique. Les fonctions principales de cette troisième phase sont les suivantes.

- Variogramme: Calcul du variogramme du modèle.
- Krigage: Estimation par krigeage.

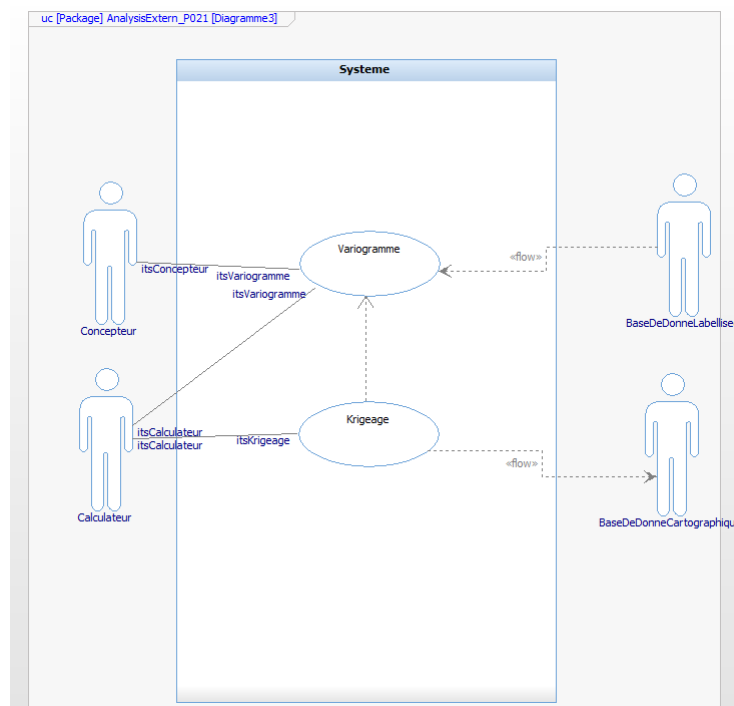


FIGURE 4 – Diagramme en pieuvre de la troisième phase de vie du système.

Les différentes étapes de traitement ont permis d'ajouter des informations à chaque enregistrement audio, dont la localisation et la date de la mesure, son label en utilisant un algorithme de deep learning, et par ailleurs de s'appuyer sur ces mesures pour estimer la même grandeur étudiée dans des zones voisines. A partir de ces informations regroupées dans un fichier texte, la dernière étape consiste à transmettre ces informations grâce à une carte. Cette carte doit avoir les deux fonctions principales



suivantes pour permettre à un utilisateur de visualiser la pollution sonore dans un quartier.

- Carte: Insertion de la carte sur une page web.
- Mise à jour carte: carte interactive.

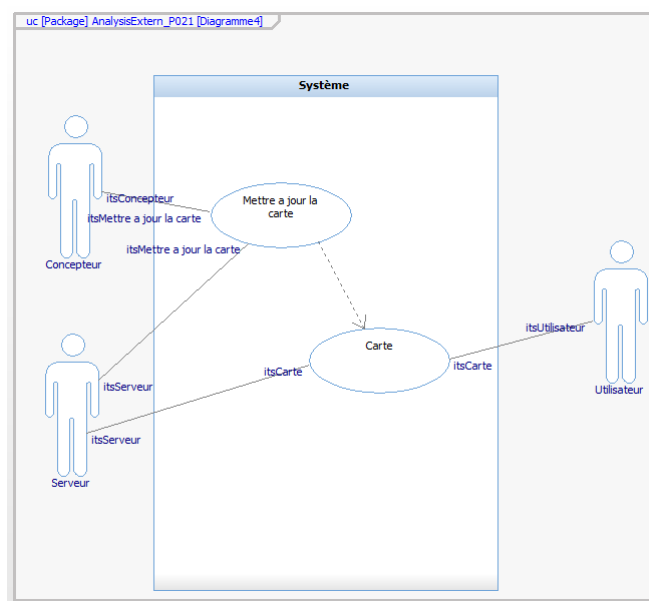


FIGURE 5 – Diagramme en pieuvre de la quatrième phase de vie du système.

3.2.2 Analyse interne du système

A partir de l'analyse externe présenté précédemment, une analyse plus approfondie des fonctions principales est nécessaire pour comprendre le fonctionnement du système. Une nouvelle fois, nous divisons cette partie en suivant la phase de vie du système.

Pour la première phase, les fonctions principales présentées dans l'analyse externe sont décrites ci-dessous.



FIGURE 6 – Diagrammes internes FAST de la première phase de vie.

La deuxième phase consiste à récupérer les fichiers, préalablement traités pour attribuer un sonogramme à chaque enregistrement audio, pour les classer en fonction des labels choisis. Cette partie utilise un algorithme de deep learning. La partie d'apprentissage a pour but de trouver un modèle répondant aux contraintes et donc de l'évaluer. Ensuite, la seconde partie réutilise ce programme pour prédire quelle est la source entendue dans l'enregistrement.

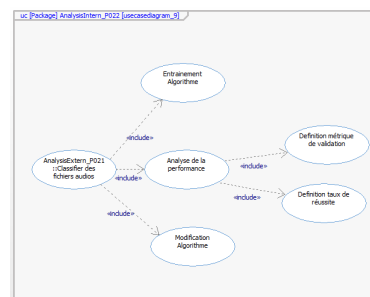


FIGURE 7 – Diagramme interne FAST de la deuxième phase de vie.



La troisième phase d'estimation nécessite la première phase avec un variogramme qui doit transmettre les paramètres au krigeage ordinaire. Ce krigeage permet de s'appuyer sur les mesures expérimentales pour prévoir ces grandeurs en des sites distincts.

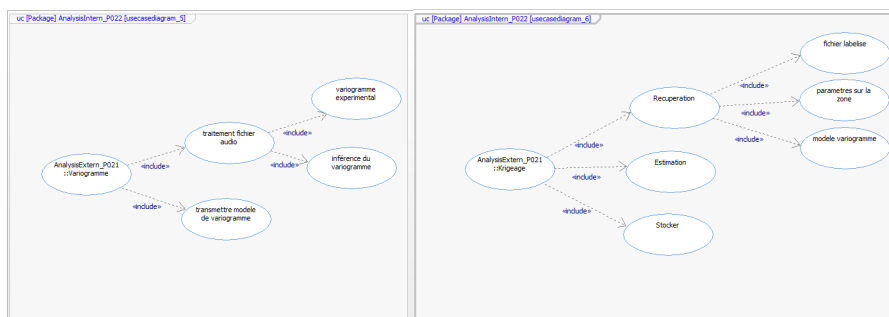


FIGURE 8 – Diagrammes internes FAST de la troisième phase de vie.

Enfin, la carte est élaborée afin de visualiser les données issues des traitements précédents.

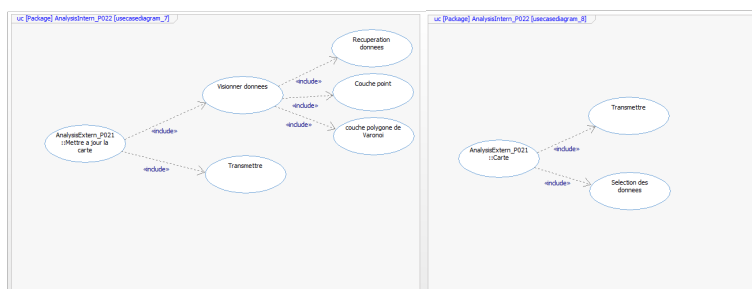


FIGURE 9 – Diagrammes internes FAST de la quatrième phase de vie.



3.2.3 Architecture physique du système

Les fonctions ont été présentées précédemment afin de répondre aux contraintes auxquelles doit répondre le système. Ces fonctions doivent être réellement présentes à l'aide de support physique ou immatériel. Tous ces objets sont connectés entre eux par des flux, souvent d'information, dans le but d'exploiter des données pour arriver à un modèle visuel. Ci-dessous, l'architecture physique est présente sous forme d'un diagramme. Certains blocs correspondent à des algorithmes Python, comme séparation fichier, algorithme, variogramme et krigeage. Ou encore, des logiciels comme audacity pour le sonogramme et QGIS pour la carte. D'autres sont des jeux de données comme tous les blocs débutants par BaseDeDonnée et les blocs apprentissage, test, utilisation. Les deux dernières parties se trouvent au début et à la fin. La première utilise un capteur audio nommé Audiomoth et un récepteur GPS se trouvant dans un smartphone, réalisant des mesures en absolu. Enfin, la carte SD est insérée dans l'Audiomoth afin de stocker les fichiers audios. La deuxième correspond à la page web. La carte générée sous QGIS est intégrée à une page web, qui pourra être partagée à un utilisateur.

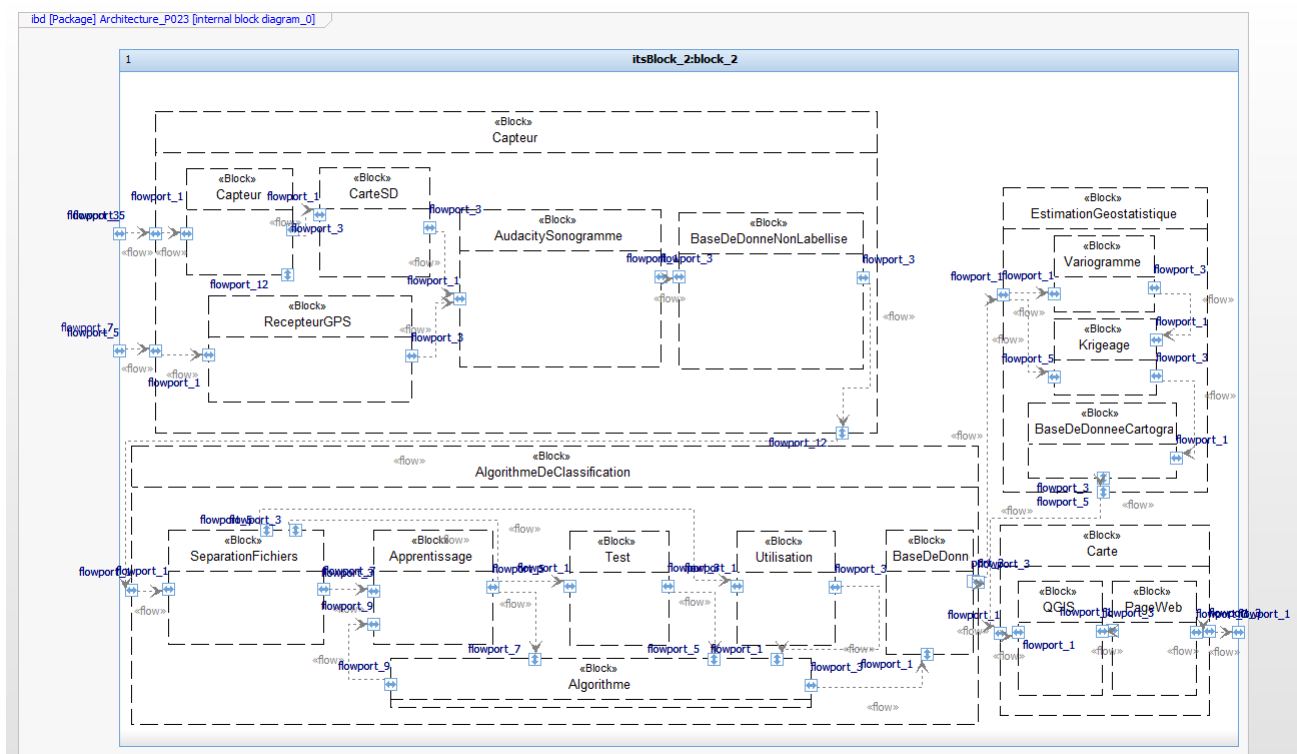


FIGURE 10 – Diagramme de l'architecture physique du système.



3.3 Diagramme de Gantt

Lors des premières semaines, un diagramme de Gantt a été réalisé afin d'avoir une première vue sur les besoins dont nous devons avoir au cours du projet. Certaines étapes ont été respectées. D'autres ont pris plus de temps ou parfois au contraire ont été plus courtes que prévues, voir presque supprimées comme la partie traitement du signal. Une partie sur l'estimation par krigeage a été ajoutée et n'apparaît pas dans le diagramme de Gantt initial. L'acquisition des données quant à elle n'a pas encore commencé et fera l'objet d'une attention particulière pour la deuxième phase du projet.

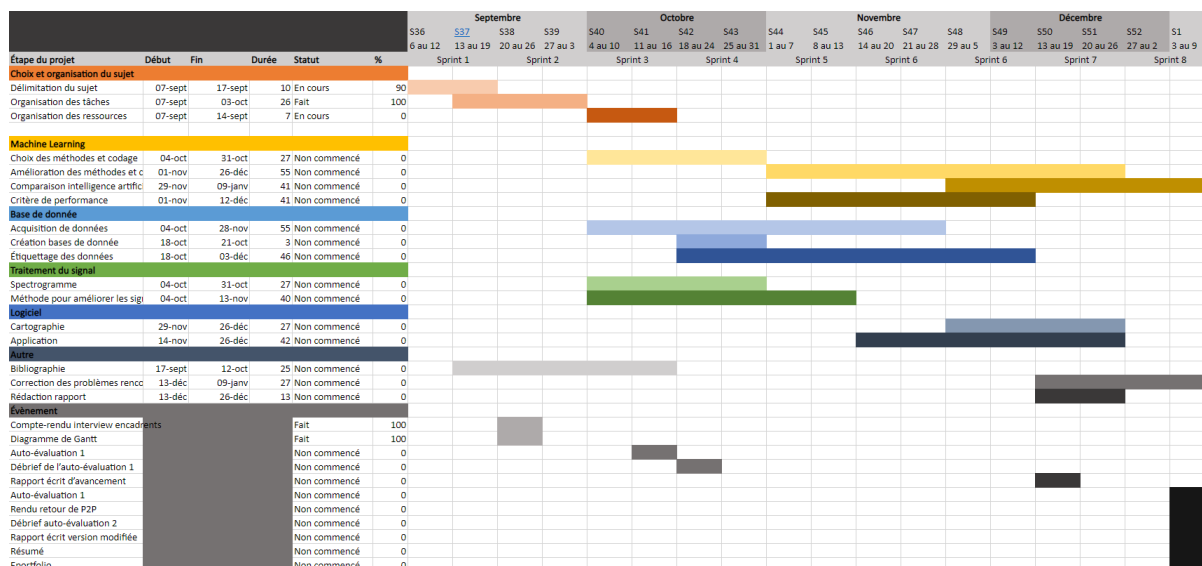


FIGURE 11 – Le diagramme de Gantt était un premier aperçu sur les étapes du projet.

3.4 Organisation de groupe par méthode agile

La méthode Agile a été utilisée pour le bon avancement du projet. Le consultant de l'organisation en groupe est Sylvain GUERIN. Deux rencontres ont eu lieu afin de s'organiser avec la méthode Agile. Les différents rôles ont été attribués, ils sont listés ci-dessous.

Product Owner : E. FERRAND

Scrum Master : T. ALBERT, H. FAUVEL

Équipe technique : T. ALBERT, E. FERRAND, H. FAUVEL, A.J. RAKAN

Chaque sprint a fait l'objet d'une réunion. De plus, deux outils ont permis de suivre l'avancement du projet afin de répondre au critère de transparence du mode Agile. Des comptes rendu sur une page environ ont été rédigés pour chaque sprint, afin que chaque membre rende compte de son travail et partage aux autres ce qu'il a fait de son côté. Pour que les membres du groupe se partagent le travail et pour faciliter la division du travail préconisé par la méthode Agile, Trello a été utilisé. Pour chaque sprint, les tâches à faire sont glissées dans l'onglet correspondant au sprint, en spécifiant le membre de l'équipe chargé de ce travail.



4 État de l'art

4.1 Machine Learning

4.1.1 Définitions

Une grosse partie du projet repose sur l'application d'algorithmes d'apprentissage automatique (en anglais *Machine Learning*) qu'il convient de définir préalablement. En effet, lorsqu'on parle d'intelligence artificielle (IA), il s'agit de l'ensemble de théories et de techniques mises en œuvre en vue de réaliser des machines capables de simuler l'intelligence humaine (définition Larousse). Une branche majeure de l'intelligence artificielle est l'apprentissage automatique. Arthur SAMUEL définit l'apprentissage automatique comme "*le domaine d'étude qui donne aux ordinateurs la capacité d'apprendre sans être explicitement programmés*". Il s'agit d'une définition ancienne et informelle. Mais cette définition permet une approche conceptuelle de la méthodologie mise en place plus tard. En effet, en programmation classique, le développeur va rentrer des *données* dans son algorithme et en définir les *règles* pour récupérer des *réponses*. Tandis qu'en apprentissage automatique, le développeur ne rentre que les *données* et *réponses* pour récupérer les *règles*. Ces règles peuvent ensuite être appliquées à de nouvelles données pour obtenir « automatiquement » des réponses cohérentes. Le Schéma suivant décrit la différence de paradigme.

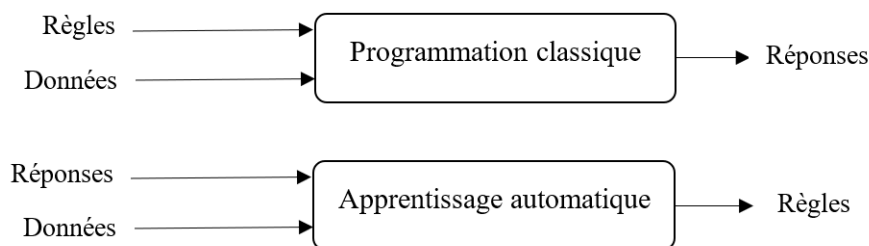


FIGURE 12 – Machine Learning, nouveau paradigme de programmation.

Tom Mitchell fournit une définition plus moderne : "On dit qu'un programme informatique apprend de l'expérience E en ce qui concerne une certaine classe de tâches T et une mesure de performance P, si sa performance aux tâches dans T, telle que mesurée par P, s'améliore avec l'expérience E." Un exemple classique est le jeu de dames, où E est l'expérience de nombreuses parties de dames. T la tâche consistant à jouer aux dames. Et P la probabilité que le programme gagne la prochaine partie. Ainsi on dira davantage d'un algorithme de machine learning qu'il est *entraîné* plutôt que programmé.

4.1.2 Les phases d'un projet de machine Learning

Une fois un besoin identifié, il est possible de définir une marche à suivre pour la réalisation d'un projet de Machine Learning. Il s'agit d'un cycle en cinq phases [6] :

I. Collecte des données

Tout d'abord, on peut se demander si la donnée existe déjà ou s'il faut prévoir une phase de captation de données. Il est souvent utile de se renseigner sur les législations en vigueur relatives aux données traitées.

II. Préparation des données

Le prétraitement des données (en anglais *data preprocessing*) consiste à mettre en forme les données brutes. Il convient d'effectuer des modifications préalables aux données brutes afin de les simplifier au maximum pour représenter plus précisément le problème sous-jacent que l'on souhaite résoudre. Il y



a plusieurs étapes dans cette phase [6] :

- L'évaluation des données : La qualité des données dépend des capteurs utilisés et peut parfois altérer l'analyse effectuée par le programme informatique. Si les valeurs manquantes d'une donnée sont trop importantes il est possible de les supprimer. Si par ailleurs la part de valeurs manquantes est faible, il est possible d'estimer cette valeur (moyenne, médiane...).
- L'échantillonnage des données : L'échantillonnage consiste à définir un certain nombre de sous-ensembles de données. Cela permet également une réduction de la consommation de mémoire et du temps de traitement. Il est indispensable d'obtenir un échantillon représentatif. Ses propriétés doivent donc être les mêmes que celles des données d'origine.
- Réduction de dimension : Le principe de la réduction de dimension est de regrouper certaines caractéristiques en une même entité. Les valeurs attribuées à quelques-unes des caractéristiques peuvent alors donner une même valeur pour cette entité.
- Feature Engineering : Le processus de feature engineering consiste à utiliser des compétences annexes (physique, chimie ...) afin de simplifier la compréhension des données par l'algorithme. Par exemple, un problème de lecture d'horloge dont la donnée brute serait la photo d'une horloge sera plus facilement traité si on la transforme en un jeu de coordonnées des aiguilles ou encore les angles des aiguilles. Une fois les données prétraitées, il est possible d'entraîner le modèle choisi. C'est l'objet de la phase suivante.

III. Entraînement d'un modèle

La phase d'entraînement consiste à développer un modèle de résolution ou bien à en utiliser un déjà existant (librairies keras et tensorflow sur Python par exemple).

IV. Analyse du modèle et des résultats

La phase d'analyse du modèle à partir des résultats consiste à évaluer le taux de réussite du modèle utilisé. La performance d'un même algorithme peut varier selon les différents dataset. En fonction des résultats, il est possible que le modèle soit sous-adapté (underfit) ou sur-adapté (overfit) par rapport au problème étudié. Cette question sera traitée lors des sous-/sur-/apprentissage dans la suite de ce rapport.

V. Amélioration du modèle

Enfin, fort des analyses du modèle, le modèle peut être amélioré. Le cycle se termine et recommence jusqu'à ce que la performance du modèle soit satisfaisante.

4.1.3 Différents types de machine learning

Il existe un grand nombre de méthodes de résolution dans le cadre d'apprentissage automatique. Les plus connus sont : la classification où des labels sont placés sur une donnée après analyse, le clustering où l'on cherche à rassembler des données voisines selon diverses caractéristiques, la régression où l'on prédit des coefficients liés aux valeurs d'entrées. La figure suivante décrit quelques exemples :

Dans le cadre de notre étude, nous nous tournons plutôt vers des méthodes de résolution de problèmes de classification multi-label.



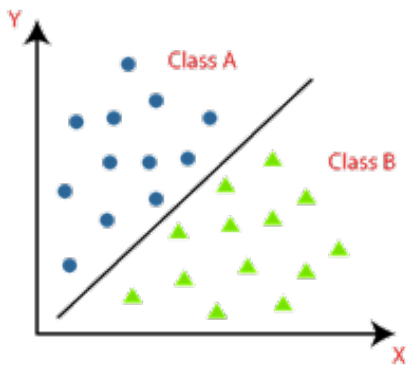


FIGURE 13 – Classification

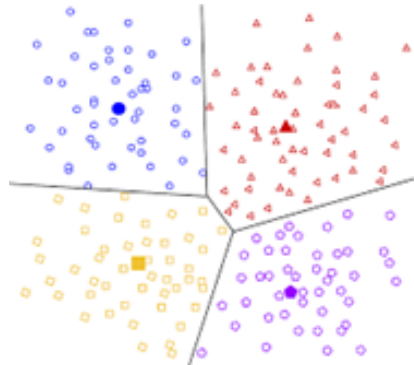


FIGURE 14 – Clustering

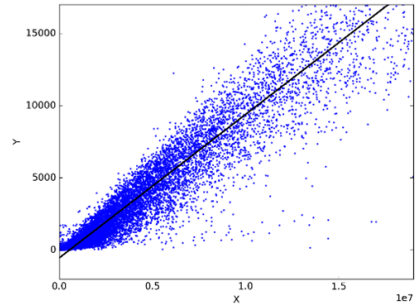


FIGURE 15 – Régression

4.2 Deep Learning et Réseaux de Neurons

Les réseaux de neurones artificiels tirent leur nom des réseaux de cellules nerveuses du cerveau. Bien qu'une grande partie des détails biologiques soit éliminée dans ces modèles informatiques, les réseaux neuronaux artificiels conservent suffisamment la structure observée dans le cerveau pour permettre de comprendre comment le traitement neuronal biologique peut fonctionner[7]. Cette partie s'intéresse à développer leur fonctionnement.

4.2.1 Réseaux de neurones artificiels: architecture et fonctionnement

Un neurone (au sens biologique) est constitué de trois éléments principaux : l'axone est une longue fibre ramifiée en sortie du neurone, les dendrites sont un ensemble de fibres ramifiées à l'entrée du neurone, la synapse est le point de connexion entre un axone et une dendrite. Lorsqu'une série d'impulsions est reçue dans les zones dendritiques d'un neurone, il en résulte généralement une probabilité accrue que le neurone cible envoie une impulsion dans son axone[7]. La figure 16 propose une représentation.

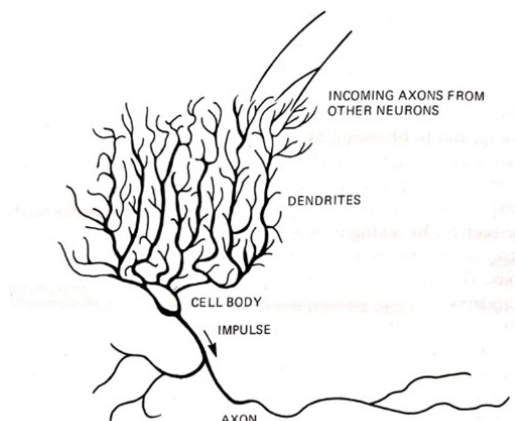


Figure 1-1. Schematic drawing of a biological nerve cell.

FIGURE 16 – Neurone naturel simplifié

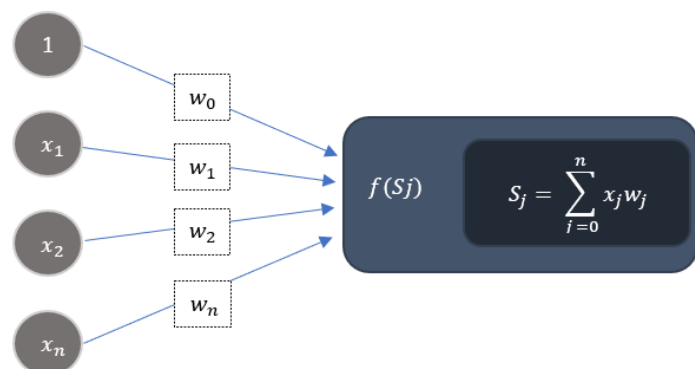


FIGURE 17 – Le perceptron

Gardons cette information en tête et, par analogie, nous allons définir réseau de neurones artificiel. Tout d'abord, un neurone artificiel est une unité qui reçoit un certain nombre de valeurs en entrée, qui effectue une somme pondérée qui est ensuite évaluée par une fonction spéciale appelée « fonction d'activation », le résultat étant la sortie du neurone. Dans la Figure 17 ci-dessus, la fonction en question est f , et on note qu'il y a un terme constant, qui sera toujours présent. Les familles de poids sont propres à chacun des neurones et sont les paramètres du modèle à régler.



Cette brique élémentaire est appelée le Perceptron et constitue la base d'un réseau de neurones. On peut alors construire une couche (*layer* en anglais) de neurones. Par la suite, on crée le réseau de neurones en empilant les couches, on parle parfois de perceptron multicouche. On s'intéressera ici à un réseau où la progression est à sens unique, de la gauche vers la droite. On a donc un réseau de neurones à propagation avant (*feedforward neural network* en anglais). À partir de cette structure, on crée des réseaux très variés, par exemple pour ce qui est du traitement d'image on a recours à des réseaux de neurones convolutifs.

Le perceptron multicouche (*multilayer* en anglais) peut être utilisé dans des problèmes de classification. La figure suivante en propose une illustration. Il faut remarquer que la couche de sortie propose autant de sorties que de classes prédéfinies, le nombre de couches cachées est également prédéfini.

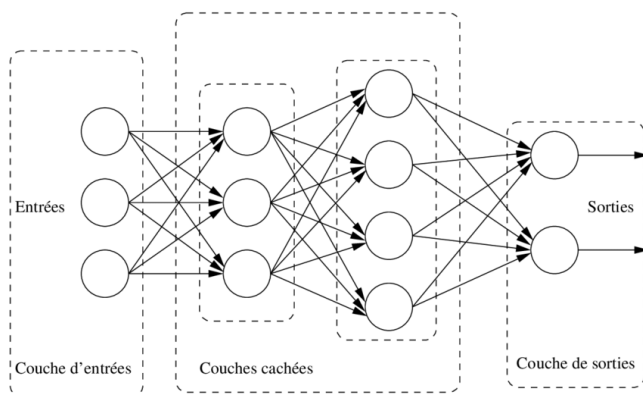


FIGURE 18 – Deux classes

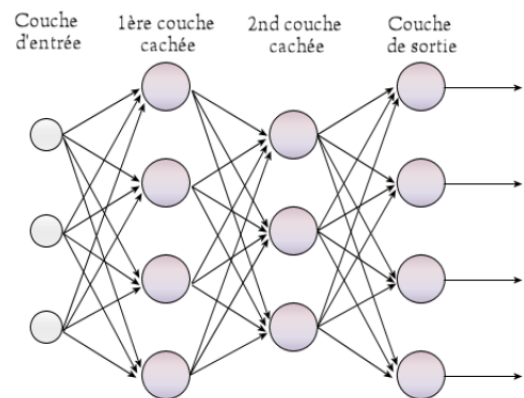


FIGURE 19 – Quatre classes

FIGURE 20 – Perceptrons multicouches dans des problèmes de classification

4.2.2 Sur-apprentissage et sous-apprentissage

Deux grands risques liés à l'apprentissage automatique sont le surapprentissage et le sousapprentissage (*overfitting* et *underfitting* en anglais). Ces deux phénomènes arrivent lorsque l'on souhaite développer un modèle mais que ce dernier n'arrive pas à généraliser (cas de surapprentissage) ou bien qu'il ne généralise pas suffisamment (cas de sous apprentissage).

Dans le domaine de l'apprentissage automatique, on parle d'*overfitting* ou de surapprentissage quand le modèle fonctionne bien sur les données d'apprentissage, mais qu'il ne généralise pas bien. On parle d'*underfitting* ou de sousapprentissage lorsque le modèle s'adapte mal et n'arrive pas à prendre en compte les corrélations des données d'entraînement [8]. Ces deux phénomènes sont illustrés dans les Figures 21 et 22 sur des cas de régressions linéaires.



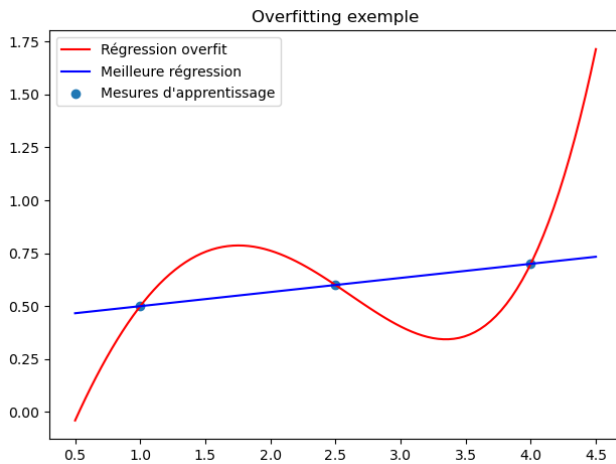


FIGURE 21 – Exemple de sur apprentissage

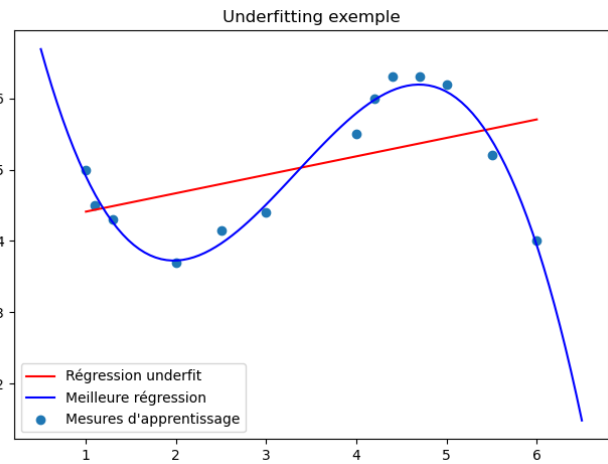


FIGURE 22 – Exemple de sous apprentissage

4.2.3 Les réseaux de neurones convolutifs

L'utilisation des réseaux de neurones convolutifs est la technique la plus souvent utilisée pour la reconnaissance d'image. Nous allons, nous aussi, utiliser cette technique.

On peut distinguer, dans un réseau de neurones convolutifs, deux grandes étapes : l'extraction des caractéristiques et la classification. Pour extraire les caractéristiques, on procède par l'application de deux couches. La première est une couche de convolution, la seconde est une couche dite de "pooling".

La première étape consiste à identifier et séparer les différentes caractéristiques à l'aide d'une couche de convolution, qui applique plusieurs filtres. [7] Ces filtres sont représentés par des matrices dont les éléments sont appelés des « poids ». Ce sont les mêmes que ceux présentés dans la partie précédente, sauf qu'ils sont liés les uns aux autres (à côté dans la matrice, ou à l'opposé). Ces poids vont permettre, par opération mathématique de convolution avec la matrice des niveaux de gris du png (valeurs de 0 à 255), d'obtenir des valeurs indiquant la ressemblance entre le filtre et la portion de matrice. Cette opération est répétée en déplaçant le filtre sur l'ensemble de l'image, ce qui fait ressortir les zones d'une image qui se rapprochent le plus du filtre. Ainsi, la convolution permet d'identifier la présence de patterns dans la matrice de gris donnée en entrée. Pour le but du deep-learning par réseau de neurone convolutifs et de trouver les patterns qui vont être discriminants des différentes classes, et dont on va ainsi chercher la présence, par l'intermédiaire des convolutions. C'est l'algorithme qui module les poids au fur et à mesure de l'apprentissage.

On peut enchaîner les étapes de convolutions, mais petit à petit, la matrice va diminuer en taille. Il peut alors être intéressant d'utiliser le "zero-padding", afin de continuer à avoir des filtres d'une certaine taille, et contenant un certain nombre de poids à moduler. Le zero-padding consiste à ajouter des zéro tout autour de la matrice qui subit les convolutions, avant une convolution.

Une fois l'opération de convolution par un filtre effectuée, on applique une fonction d'activation, "ReLU" dans notre cas. Elle permet mettre les valeurs négatives à 0 et conserve les valeurs positives afin de mettre en exergue les points d'apparitions des patterns notamment.

La seconde étape est l'étape de Pooling. Elle permet de ne pas accorder trop d'importances à des légères variations par rapport au pattern recherché, et donc de reconnaître celui-ci, même s'il n'est pas exactement identique. Concrètement, l'étape de Pooling, dans notre cas de Max Pooling, filtre les données par carrés 2x2, et conserve la valeur maximale dans la matrice de sortie. Cette valeur max est



la plus grande car la convolution l'a permis, parce que le pattern recherché et la pixellisation de la matrice au voisinage de ce point correspondaient. D'où le fait que cette étape est primordiale dans la reconnaissance du pattern, elle "oublie" les données insignifiantes et conservent celle qui témoignent d'une présence de celui-ci.

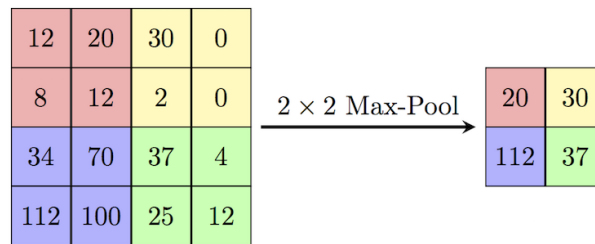


FIGURE 23 – Fonctionnement du Max Pooling.

Le réseau de neurones convolutifs est donc constitué, pour ce qui est de l'extraction des caractéristiques, d'une suite d'applications de ces deux types de couches (Convolution et Pooling). Cette opération est répétée dans des couches plus profondes, avec des filtres différents. Une fois cette succession de couches appliquée, le résultat est mis sous la forme d'un vecteur colonne. Ce vecteur colonne est alors traité par un réseau de neurone traditionnel, comme défini dans la partie précédente.

4.3 Estimation spatiale par krigeage ordinaire

4.3.1 Le krigeage ordinaire pour des coordonnées géographiques

Les mesures effectuées par un capteur ne peuvent être réalisées en tout point de l'espace. Tandis que la propagation du son est un phénomène continu, l'objectif de cette partie consiste à estimer, dans toute la zone étudiée, l'intensité sonore à partir seulement des mesures expérimentales. Deux approches sont possibles. L'une déterminisme où on ne se préoccupe que des mesures locales et où l'on essaie d'interpoler par une fonction dans \mathbb{R}^2 . Il existe de nombreuses techniques réalisant cette tâche. L'un des problèmes est qu'aucune information ne renseigne sur le champ étudié, et donc sans se préoccuper du phénomène physique, ces techniques ne tiennent pas compte de la réalité. La deuxième approche utilise des techniques de géostatistique prenant en considération l'influence des sites sur les autres. Ces techniques de krigeage étudient le phénomène physique pour estimer l'intensité sonore avec des outils statistiques. En un point distinct des zones mesurées expérimentalement, on possède donc à la fois une estimation de l'intensité sonore, mais aussi des indicateurs de la qualité de cette estimation, dont l'écart-type.

Avant tout traitement, nous devons apporter le plus grand soin à bien distinguer les mesures expérimentales des mesures prédites par des méthodes géostatistiques. L'abstraction que nous allons présenter dans la suite ne saurait décrire la réalité des phénomènes physiques sans prudence, encore moins les expliquer. Cette précaution doit nous conduire tout au long du processus à ne pas confondre modèle mathématique et environnement réel.

Une fois les mesures acquises en différent emplacement localisé dans un système cartographique, nous allons nous attacher à estimer la pollution sonore dans l'ensemble de l'espace. Bien que la propagation du son soit un phénomène continu dans l'espace, nous ne pouvons percevoir ce phénomène qu'à des points de mesures. En étant réduit à ces points, la visualisation de la pollution sonore est d'autant plus difficile que ce nombre de point est relativement faible. En cette raison, nous avons étudié des méthodes de statistique spatiale pour prédire l'intensité sonore dans un espace continu, ou au moins pour un nombre de point très important et qui sont proches les uns des autres.



Soit un champ D , dans notre étude cet espace représente la zone étudiée. Une fonction aléatoire Z sur D représente un processus aléatoire. En prenant un site s de D , on réalise un tirage aléatoire de $Z(s)$ qui donne la variable régionalisée $z(s)$. La fonction aléatoire Z est caractérisée par sa loi spatiale qui correspond à la donnée de $Z(s_1), Z(s_2), \dots$.

Pour interpoler l'intensité sonore en des points suggérés, nous avons utilisé des méthodes de krigeage. Mais avant toutes estimations, plusieurs informations sur le phénomène physique sont nécessaires. Dans un premier temps, nous avons besoin de connaître la disposition spatiale des capteurs afin de déterminer la distance d'une mesure s_i à une autre s_j . Ces distances sont stockées dans une matrice H . En se contentant de cette matrice et des intensités sonores évaluées, la première étape consiste à estimer le demi-variogramme du modèle. Ce graphique existe à condition que le processus stochastique Z soit intrinsèquement stationnaire, ce que nous ferons, c'est-à-dire que ses accroissements sont stationnaires du second ordre. Ce modèle permet de définir le demi-variogramme [4].

$$\gamma(h) = \frac{1}{2} \text{Var}(Z(s) - Z(s + h)) \quad (1)$$

Le demi-variogramme opère donc sur les accroissements. Il évolue en opposition avec la corrélation entre deux points. Le réseau des mesures expérimentales n'étant pas régulier, le calcul expérimental du demi-variogramme est donné par la formule suivante.

$$\hat{\gamma}(h) = \frac{1}{2 N(h)} \sum_{(i, j) \in \mathbb{N}^2} (z(s_i) - z(s_j))^2 \quad (2)$$

Où les sites s_i et s_j sont séparés par une distance entre $h - \frac{dh}{2}$ et $h + \frac{dh}{2}$, $N(h)$ le nombre de couple (s_i, s_j) .

Le demi-variogramme a pour objectif d'observer les interactions spatiales du modèle. Au lieu de le réutiliser directement, nous l'approchons par un modèle mathématique. Dans le cas de cette étude, le demi-variogramme est représenté ci-dessous. Nous avons identifié un modèle linéaire dont les paramètres sont obtenus par les moindres-carrés.

Une fois les caractéristiques du phénomène physique étudiées, nous pouvons l'ajouter au modèle de krigeage. Dans le cas d'une fonction aléatoire supposée strictement intrinsèque, le krigeage ordinaire s'applique.

$$\begin{cases} \sum_{\beta=1}^n \lambda_{\beta} \gamma(s_{\alpha} - s_{\beta}) - \mu = \gamma(s_{\alpha} - s_0), \text{ pour } \alpha = 1, \dots, n \\ \sum_{\alpha=1}^n \lambda_{\alpha} = 1 \end{cases} \quad (3)$$

Soit matriciellement,

$$\begin{pmatrix} \gamma(s_{\alpha} - s_{\beta}) & \mathbf{1} \\ \mathbf{1} & 0 \end{pmatrix} \begin{pmatrix} \lambda_{\beta} \\ -\mu \end{pmatrix} = \begin{pmatrix} \gamma(s_{\alpha} - s_0) \\ 1 \end{pmatrix} \quad (4)$$

μ est le multiplicateur de Lagrange. La variance du krigeage est $\sum_{\alpha=1}^n \lambda_{\alpha} \gamma(s_{\alpha} - s_0) - \mu$ [4].

Le krigeage ordinaire possède plusieurs propriétés, mais nous en citerons deux particulièrement intéressantes. Une estimation réalisée sur un site où a été réalisée une mesure expérimentale sera exacte. L'interpolation est exacte. Cela ne veut pas dire que les estimations réalisées en dehors des sites de mesure sont quant à elles exactes. De plus, les estimations ont tendance à être attirées par une valeur constante m dans le domaine D . Cette valeur correspond à la moyenne estimée de manière optimale par les mesures en entrées.

4.3.2 Krigeage d'une fonction aléatoire sur une dimension temporelle

Les mesures expérimentales réalisées sont identifiées par des coordonnées géographiques, mais aussi par une date. Cette deuxième information fait l'objet d'un traitement pour estimer l'intensité sonore maximale probable sur une plage horaire déterminée. Les mesures effectuées par l'audiomoth sont



réalisées sur une durée δT avec une fréquence f de 8.3×10^{-4} Hz, soit toutes les vingt minutes. Nous détaillerons avec plus d'application cette démarche de prise de mesure dans la partie suivante.

En rassemblant suffisamment de données en un site pour un même horaire, nous pouvons couvrir des mesures sur l'ensemble de la journée et de la semaine. Ces données ne seraient être suffisamment précises pour déterminer l'évolution de la pollution sonore à l'échelle de la minute. L'objectif suivant est de prédire l'intensité sonore entre deux enregistrements écartés d'une heure. Encore une fois, deux possibilités sont possibles pour cette estimation, une approche déterminisme dont les spiles, une approche statistique par krigeage.

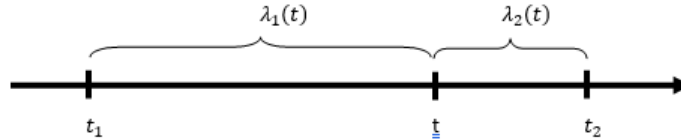


FIGURE 24 – Schéma du krigeage sur une dimension temporelle. Les poids $\lambda_1(t)$ et $\lambda_2(t)$ peuvent être interprétés comme l'influence exercée par les mesures aux instants t_1 et t_2 sur l'estimation à t .

En reprenant le krigeage ordinaire évoqué plus tôt, l'estimation de l'intensité sonore sur une dimension temporelle est réalisable. Soit Z une fonction aléatoire à une dimension supposée strictement intrinsèque. Le variogramme doit être déterminé comme dans la partie précédente. De plus, l'expression écrite en (1) sera approchée par discrétisation, en utilisant les mesures expérimentales.

La fonction aléatoire Z est connue par mesures expérimentales en deux instants t_1 et t_2 , $t_1 < t_2$. Soit S une durée inférieure à $t_2 - t_1$, la moyenne temporelle de Z sur S est $Z_S = \frac{1}{|S|} \int_S Z(t) dt$, où S : durée de S (5)

On notera le variogramme de la régularisé tel que ci-dessous

$$\bar{\gamma}(t, S) = \frac{1}{|S|} \int_S \gamma(t_1 - u) du \quad (6)$$

$$\bar{\gamma}(S, S) = \frac{1}{|S|^2} \int_S \int_S \gamma(u - u') du du' \quad (7)$$

Le système de krigeage (4) de $Z(S)$ est exprimé ci-dessous par les deux mesures déterminées $Z(t_1)$ et $Z(t_2)$

$$\begin{cases} \lambda_1(S) \gamma(t_1 - t_1) + \lambda_2(S) \gamma(t_1 - t_2) + \mu(S) = \bar{\gamma}(t_1, S) \\ \lambda_1(S) \gamma(t_2 - t_1) + \lambda_2(S) \gamma(t_2 - t_2) + \mu(S) = \bar{\gamma}(t_2, S) \\ \lambda_1(S) + \lambda_2(S) = 1 \end{cases} \quad (8)$$

Puisque le variogramme est nul en 0,

$$\begin{cases} \lambda_2(S) \gamma(t_1 - t_2) + \mu(S) = \bar{\gamma}(t_1, S) \\ \lambda_1(S) \gamma(t_2 - t_1) + \mu(S) = \bar{\gamma}(t_2, S) \\ \lambda_1(S) + \lambda_2(S) = 1 \end{cases} \quad (9)$$

Lorsque S est réduit à un instant t , $t_1 < t < t_2$ $S = t$,

$$\begin{cases} \lambda_2(t) \gamma(t_1 - t_2) + \mu(t) = \bar{\gamma}(t_1, t) \\ \lambda_1(t) \gamma(t_2 - t_1) + \mu(t) = \bar{\gamma}(t_2, t) \\ \lambda_1(t) + \lambda_2(t) = 1 \end{cases} \quad (10)$$



On pose $\lambda(t) = \lambda_1(t)$, soit $\lambda_2(t) = 1 - \lambda(t)$. Le système (10) s'écrit

$$\begin{cases} (1 - \lambda(t))\gamma(t_1 - t_2) + \mu(t) = \bar{\gamma}(t_1, t) \\ \lambda(t)\gamma(t_2 - t_1) + \mu(t) = \bar{\gamma}(t_2, t) \end{cases} \quad (11)$$

Le poids $\lambda(t)$ est déterminé en soustrayant les deux équations du système ci-dessus, tant dis que le multiplicateur de Lagrange est obtenu par addition de ces mêmes équations. [5].

$$\lambda(t) = \frac{1}{2} + \frac{\bar{\gamma}(t_2, t) - \bar{\gamma}(t_1, t)}{2\gamma(t_1 - t_2)} \quad (12)$$

$$\mu(t) = \frac{1}{2}(\bar{\gamma}(t_1, t) + \bar{\gamma}(t_2, t) - \gamma(t_1 - t_2)) \quad (13)$$

Par ailleurs, l'erreur de krigeage est de

$$\sigma_K^2(t) = -\bar{\gamma}(t, t) + \mu(t) + \lambda_1(t)\bar{\gamma}(t_1, t) + \lambda_2(t)\bar{\gamma}(t_2, t) \quad (14)$$

Les trois expressions précédentes permettent de déterminer pour tout instant t , $t_1 < t < t_2$, une estimation de l'intensité sonore maximale et son intervalle de confiance par l'erreur de krigeage. Dans le cas où nous supposons un variogramme linéaire $\gamma(t) = a|t|$, a un réel, le krigeage coïncide avec une interpolation linéaire entre les deux instants où ont été réalisées les mesures en t_1 et t_2 , dont les poids sont les suivants.

$$\lambda_1(t) = \frac{t_2 - t}{t_2 - t_1} \quad (15)$$

$$\lambda_2(t) = \frac{t - t_1}{t_2 - t_1} \quad (16)$$

4.3.3 Modèle du variogramme

Dans les deux parties précédentes, le choix du variogramme est réalisé expérimentalement en discrétisant l'équation (1). De plus, on supposera dans la suite que le variogramme est isotrope, traduisant qu'un site exerce la même influence dans toutes les directions. Seule la distance entre un site et un emplacement à estimer fait varier le variogramme. L'étude du variogramme n'a donc pas besoin de prendre en compte la direction entre deux sites, facilitant sa détermination par un modèle interpolateur que nous pourrions visualiser en deux dimensions, avec la distance en abscisse, la valeur du variogramme en ordonnée. Cette isotropie se justifie par le caractère isotrope du monde réel où une onde est supposée se propager de manière isotropique dans un milieu homogène. Cette onde accoustique n'est autre qu'une onde sphérique. [4].

Le choix du variogramme est réalisé après sa détermination expérimentale. Son approximation se fait par des fonctions connues, comme linéaire ci-dessus, ou une fonction exponentielle, parabolique... Le choix d'une fonction interpolant le variogramme expérimental doit avoir un résidu le plus faible possible. Mais entraîne également une variation de l'erreur de krigeage. Cette erreur $\sigma_K^2(t)$ dépend du modèle de variogramme choisie.

Le modèle de variogramme choisi peut être délicat. D'une part, il faut s'assurer que le modèle choisi est valide dans le domaine étudié. Par exemple, un modèle parabolique à tendance à ne pas être exploitable dans un sous domaine de \mathbb{R}^d , où $d > 3$. Dans certain cas où il vérifie pourtant la contrainte précédente, il n'est pas non plus utilisable, c'est par exemple le cas dans la figure ci-dessous où il ne répond pas au critère qu'un krigeage ordinaire soit exact aux points relevés.



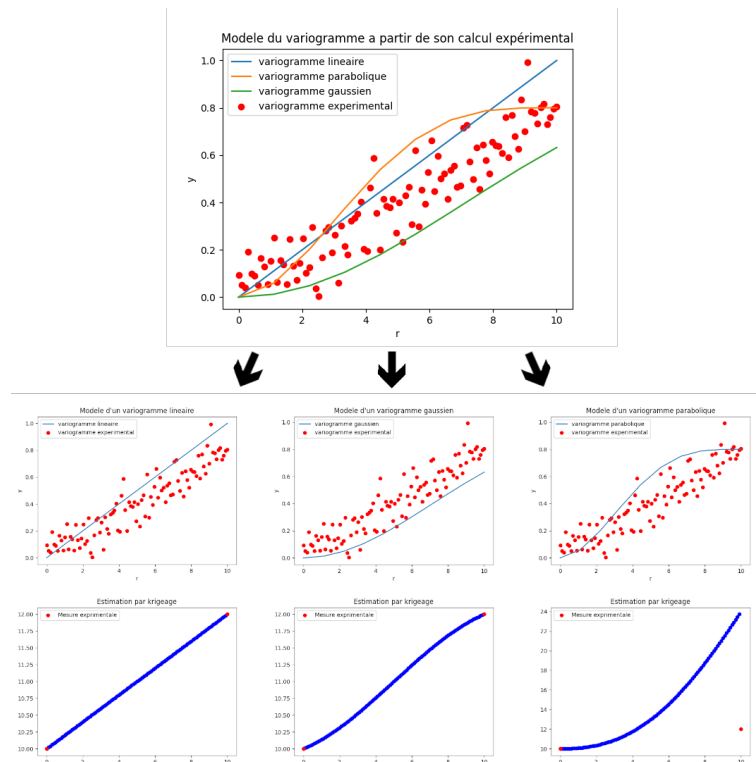


FIGURE 25 – Krigage ordinaire en une dimension temporelle. Les graphiques du haut montrent le choix d’interpolation sur le variogramme expérimental représenté par les points rouges. Les figures du bas permettent de visualiser l’estimation par krigage à partir des deux mesures expérimentales en rouge. On remarque que le variogramme parabolique n’est pas valide. Le modèle linéaire et gaussien ont une erreur de 0.333 et 0.065 respectivement, suggérant que le choix d’un variogramme gaussien est favorable dans ce cas, malgré son interpolation moins optimal avec le variogramme expérimental.

De plus, le modèle du variogramme choisi peut s’avérer ne pas être optimal. Dans la figure 2, un modèle linéaire semble favorable puisqu’il interpole mieux a priori le variogramme expérimental, mais l’erreur commise par le krigage est quant à elle plus grande qu’avec un modèle gaussien d’un facteur cinq. [3].

Certaines caractéristiques du variogramme expérimental peuvent aider à choisir un modèle. La portée, lorsqu’elle existe, indique la distance à partir de laquelle l’influence d’un site sur un autre est nul. Cette portée correspond à une variance nulle, les deux sites sont décorrélés, c’est à dire qu’au début d’un plateau sur le variogramme, la fonction devient constante. La hauteur de ce plateau est une deuxième caractéristique renseignant sur le comportement de la fonction aléatoire. Dans certains cas, le variogramme ne dispose cependant pas de portée et de plateau. D’autres caractéristiques peuvent alors être utilisées, dont le comportement à l’origine. Les modèles peuvent se distinguer par leur pente en 0, c’est par exemple le cas avec le modèle parabolique et gaussien dans l’exemple donné ci-dessus, qui ont une pente nulle, contrairement au modèle linéaire avec une pente non nulle à l’origine. Cependant, dans l’exemple présenté, il est difficile de déterminer le comportement à l’origine à cause du bruit sur le variogramme obtenu expérimentalement.

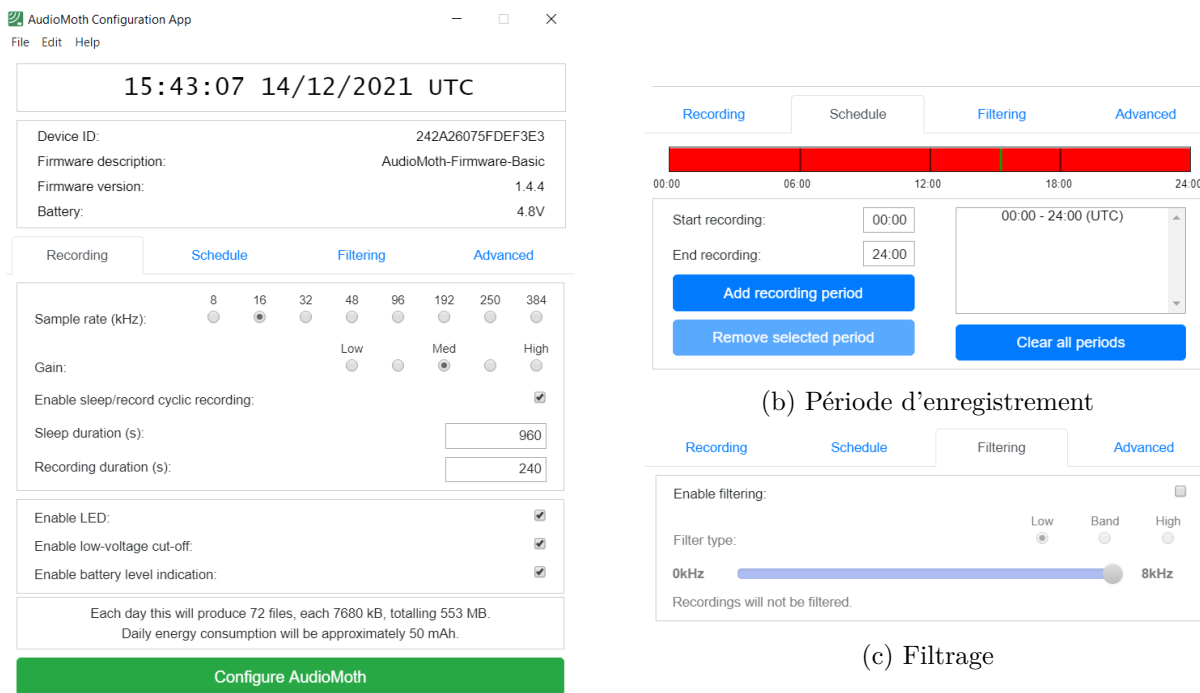


5 Relevé de prise audio dans un milieu urbain

Afin de créer une base de données de fichiers sonores suffisamment grande, nous avons besoin d'effectuer des mesures hebdomadaires. Pour ce faire, M. CAZAU nous a fourni un Audiomoth qui est un petit appareil d'acquisition sonore, facilement transportable et configurable.

Son utilisation nécessite une carte micro SD et 3 piles alcaline AA. Pour le configurer, il faut le brancher sur un ordinateur via un câble USB et lancer le logiciel permettant de régler l'appareil disponible sur *Open Acoustic Devices*. Le logiciel permet notamment de configurer la fréquence d'échantillonnage, la période et le planning d'enregistrement, ainsi que la bande fréquentielle d'enregistrement.

Notre configuration est décrite dans les figures 26a, 26b, 26c et est détaillée en-dessous.



The screenshot shows the AudioMoth Configuration App interface. It includes a status bar at the top with the time and date (15:43:07 14/12/2021 UTC). Below this, there's a section for device information (Device ID, Firmware description, Firmware version, Battery). The main configuration area is divided into four tabs: Recording, Schedule, Filtering, and Advanced. The Recording tab is active, showing settings for Sample rate (kHz), Gain, Enable sleep/record cyclic recording, Sleep duration (s), and Recording duration (s). The Schedule tab shows a timeline from 00:00 to 24:00 with recording periods marked in red. The Filtering tab shows filter type (Low, Band, High) and a frequency range from 0kHz to 8kHz. The Advanced tab shows checkboxes for Enable LED, Enable low-voltage cut-off, and Enable battery level indication. A green button at the bottom says 'Configure AudioMoth'.

(a) Paramètres d'enregistrement

(b) Période d'enregistrement

(c) Filtrage

FIGURE 26 – Paramétrage de l'audiomoth

Nous avons choisi d'enregistrer une bande fréquentielle de 0 à 8 KHz, car les sons que l'on cherche à détecter n'émettent que rarement au dessus de 8KHz. Ainsi nous avons fixé la fréquence d'échantillonnage à 16 kHz car celle-ci doit être au moins deux fois supérieure à la fréquence la plus élevée (Critère de Shannon). De plus, nous avons fixé la fréquence d'enregistrement de l'audiomoth à 4 minutes d'écoute toutes les 20 minutes, soit 16 minutes sans écoute. Enfin, la période d'enregistrement est fixée de 00 à 23h59 pour avoir une écoute tout au long de la journée. Ces paramètres fournissent par jour 72 fichiers de 4 minutes, dont la taille totale s'élève à 553 MB. Soit environ 3.5 GB par semaine.

Il nous reste encore à trouver une solution permettant d'enregistrer à l'extérieur sous des conditions météorologiques peu favorables (pluie, vent) sans endommager le matériel.



6 Traitement des données audios par deep learning

A ce stade du projet, nous n'avons pas encore recueilli nos propres données dans la ville de Brest. Ainsi, pour la partie Machine Learning et Cartographie, nous avons travaillé avec des données déjà enregistrées, celles de nos prédécesseurs. Celles-ci n'ont pas exactement les mêmes caractéristiques que celles que nous allons recueillir (durée, bande de fréquences, types de son enregistrés). Cela ne pose pas problème, en ce qui concerne le machine learning, car il s'agit d'abord de rendre les données exploitables pour l'algorithme, et de comprendre les paramètres qu'ils faudra faire évoluer tout au long de la phase d'apprentissage, à mesure que nous recueilleront de nouvelles données.

6.1 Mise en forme des données

Nous cherchons, à partir d'un jeu d'enregistrements .wav de quelques minutes, à détecter les bruits présents. Nous les décomposons en plusieurs types : voitures, chiens, voix, moto, oiseaux (classification à affiner et adapter pour avoir la meilleure performance). Le problème est alors un problème de classification multi-label : plusieurs bruits, représentés par les classes, peuvent être présents sur un même échantillon, et doivent être détectés.

Puisque nous écrivons un programme qui permet de dire si dans un échantillon, il y a telle ou telle classe de bruit, mais qui ne précise pas le nombre d'occurrence, on ne peut donner un échantillon de 5 minutes en entrée du programme d'apprentissage, car s'il y a 20 voitures en 5 minutes, on n'en détectera qu'une. Ainsi, nous découpons notre enregistrement en échantillons de 10 secondes. On ne sélectionne pas des passages sur l'enregistrement qui nous paraissent pertinents, ou qui contiennent un bruit, on exploite tout l'enregistrement, même les silences, même si un bruit de voiture chevauche deux échantillons : le but est que l'on donne un échantillon quelconque au programme entraîné, et qu'il soit capable de reconnaître ce qu'il contient. Il faut avoir un algorithme qui généralise au mieux.

Une fois les échantillons de 10 secondes obtenus, il faut les labelliser avec les classes choisies. On labellise seulement l'enregistrement de 10 secondes, mais l'on ne montre pas à l'algorithme quelle partie de spectrogramme correspond à tel label, il le «découvre» progressivement avec le réseau de neurones convolutifs.

On peut combiner les deux actions précédentes avec un programme Python 27, qui prend en entrée un fichier wav brut de 5 minutes et qui le découpe en fichier sons de 10 secondes, enregistre ces fichiers, ainsi que leur chemin d'accès et leur label dans un fichier nommé « data », Figure 28.

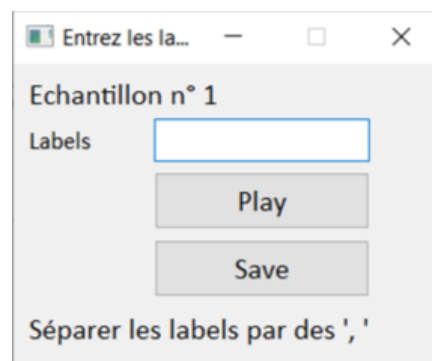
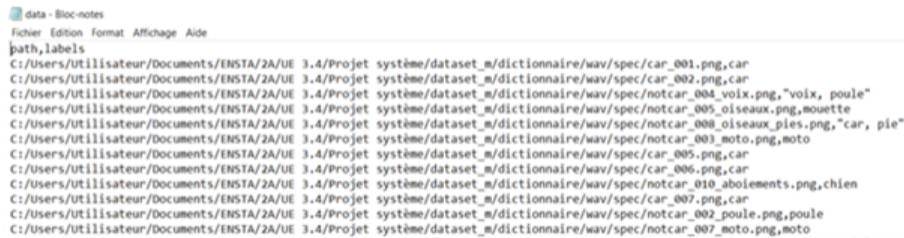


FIGURE 27 – Interface permettant de labelliser les échantillons sur Python, en les découpant.



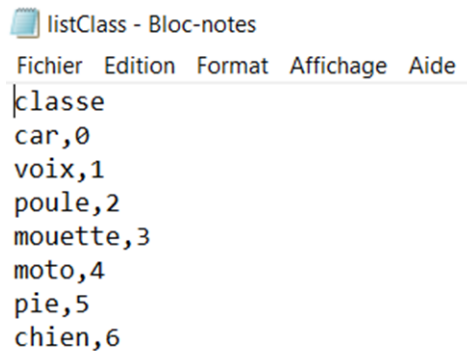
```

data - Bloc-notes
Fichier Edition Format Affichage Aide
path,labels
C:/Users/Utilisateur/Documents/ENSTA/2A/UE 3.4/Projet système/dataset_m/dictionnaire/wav/spec/car_001.png,car
C:/Users/Utilisateur/Documents/ENSTA/2A/UE 3.4/Projet système/dataset_m/dictionnaire/wav/spec/car_002.png,car
C:/Users/Utilisateur/Documents/ENSTA/2A/UE 3.4/Projet système/dataset_m/dictionnaire/wav/spec/notcar_004_voix.png,"voix, poule"
C:/Users/Utilisateur/Documents/ENSTA/2A/UE 3.4/Projet système/dataset_m/dictionnaire/wav/spec/notcar_005_oiseaux.png,mouette
C:/Users/Utilisateur/Documents/ENSTA/2A/UE 3.4/Projet système/dataset_m/dictionnaire/wav/spec/notcar_008_oiseaux_pies.png,"car, pie"
C:/Users/Utilisateur/Documents/ENSTA/2A/UE 3.4/Projet système/dataset_m/dictionnaire/wav/spec/notcar_003_moto.png,moto
C:/Users/Utilisateur/Documents/ENSTA/2A/UE 3.4/Projet système/dataset_m/dictionnaire/wav/spec/car_005.png,car
C:/Users/Utilisateur/Documents/ENSTA/2A/UE 3.4/Projet système/dataset_m/dictionnaire/wav/spec/car_006.png,car
C:/Users/Utilisateur/Documents/ENSTA/2A/UE 3.4/Projet système/dataset_m/dictionnaire/wav/spec/notcar_010_abolements.png,chien
C:/Users/Utilisateur/Documents/ENSTA/2A/UE 3.4/Projet système/dataset_m/dictionnaire/wav/spec/car_007.png,car
C:/Users/Utilisateur/Documents/ENSTA/2A/UE 3.4/Projet système/dataset_m/dictionnaire/wav/spec/notcar_002_poule.png,poule
C:/Users/Utilisateur/Documents/ENSTA/2A/UE 3.4/Projet système/dataset_m/dictionnaire/wav/spec/notcar_007_moto.png,moto

```

FIGURE 28 – Fichier «data» créé après labellisation des échantillons, associant à chaque spectrogramme ses labels.

En parallèle, on écrit dans un fichier "listIndices" les classes choisies 29.



```

listClass - Bloc-notes
Fichier Edition Format Affichage Aide
classe
car,0
voix,1
poule,2
mouette,3
moto,4
pie,5
chien,6

```

FIGURE 29 – Fichier «listClass» créé après labellisation des échantillons, contenant tous les labels.

Il nous a paru important de déterminer la durée optimale d'enregistrement (fixée à 10 sec.). Si elle est trop petite, les sons se retrouvent sur plusieurs spectrogrammes, et le nombre de détections est faussé. S'il est trop grand, on peut avoir plusieurs bruits identiques dans le même enregistrement, alors qu'on en détectera qu'un.

Mais dans un premier temps, le nombre de détection n'est pas la valeur qui nous intéresse. En effet, ce n'est pas un nombre ni une densité spatiale que l'on veut mesurer, mais une présence, qu'elle soit de 1 ou 10 voitures, la présence est approuvée. Il faut néanmoins découper en différents intervalles pour avoir une graduation de probabilités assez grande. Ex: si on ne découpe pas l'échantillon de 5 minutes, et qu'il détecte une voiture pendant 13 secondes, on aura une voiture dans l'échantillon, donc une probabilité de présence de la voiture qui sera de 1, à la date et au lieu de prise de l'échantillon.

A ce stade, les données sont des fichiers .wav, mais on va convertir ceux-ci en .png afin de pouvoir utiliser un algorithme de deep learning pour la reconnaissance d'images. Les fichiers png sont les spectrogrammes de l'échantillon : ils représentent l'intensité sonore en fonction de la fréquence et du temps, et leur représentation est semblable au spectrogramme de la figure 30.

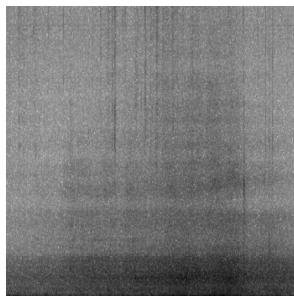


FIGURE 30 – Spectrogramme d'un échantillon, affichant l'intensité en fonction de la fréquence et du temps.

Ainsi, une fois ce travail d'adaptation des données à l'algorithme effectué, on les exploite.

6.2 Exploitation des données par méthode de Deep Learning

On définit un modèle d'apprentissage: un réseau de neurones convolutifs, pour une méthode d'apprentissage par deep-learning. En entrée de l'algorithme, on a le chemin vers le répertoire contenant les spectrogrammes, le fichier texte listClass et le fichier texte data.

6.2.1 Le réseau de neurones utilisé

Notre réseau utilise une succession de couches de Convolution et de Pooling, comme présenté figure 31. Une matrice de convolution 32x32 est suivie d'un MaxPooling 2x2, puis une matrice de convolution 64x64 est également suivie d'un Max Pooling 2x2. Enfin, on applatit les données (flatten) pour les rendre exploitables par la dernière couche de neurones. Cette dernière couche de neurone est de taille 7, et ses poids correspondent à la probabilité calculée de chacune des 7 classes. Au total, on entraîne 6 796 615 paramètres au cours de l'apprentissage.

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 498, 498, 32)	320
max_pooling2d (MaxPooling2D)	(None, 249, 249, 32)	0
conv2d_1 (Conv2D)	(None, 247, 247, 64)	18496
max_pooling2d_1 (MaxPooling2D)	(None, 123, 123, 64)	0
flatten (Flatten)	(None, 968256)	0
dense (Dense)	(None, 7)	6777799
=====		
Total params: 6,796,615		
Trainable params: 6,796,615		
Non-trainable params: 0		

FIGURE 31 – Réseau de neurones convolutifs mis en place.

6.2.2 Les prédictions

On utilise ImagaDataGenerator de TensorFlow pour générer la structure d'entraînement. On utilise ensuite un flow from dataframe sur cette structure à qui on donne en paramètre le nom (et le path)



de chaque image classée dans le répertoire d'entraînement, ainsi que les labels correspondants à ces images. Ceux-ci sont contenus dans le dataframe, que l'on a créé à partir du fichier data donné en paramètre du programme. On fait de même pour les données de validation et de test.

Enfin, on entraîne le modèle (`model.fit`) en précisant les structures d'entraînement et de validation définies ci-dessus, ainsi que le nombre d'époques, et de valeurs pour l'entraînement et pour la validation.

Une fois le modèle entraîné, on peut réaliser les prédictions (`model.predict generator`). La valeur renvoyée par la fonction de prédiction est un tableau de taille "nb classe", qui pour chaque classe, associe une probabilité de présence sur l'image. Si cette probabilité est supérieure à une valeur seuil fixée (et à déterminer), on considère que le bruit de la classe est présent sur l'échantillon (valeur `True`). Ainsi pour chaque échantillon png, l'algorithme donne les classes présentes dans celui-ci (sous la forme d'un tableau de booléen de taille nb classe).

De plus, les fichiers wav puis png ont des titres significatifs, contenant la localisation (coordonnées géographique, et la date d'enregistrement, éventuellement la rue, l'intensité maximale). On peut alors écrire un fichier texte, qui regroupe les données utiles à la réalisation de la carte: x,y, t, [booléen classe1, booléen classe2, etc], intensité. Un extrait de ce fichier texte est représenté figure 32.

x,y,t et intensité sont contenues dans le titre, recueillies par un récepteur GPS pour les coordonnées, et par l'audiomoth sinon. On prend l'intensité maximale de l'échantillon pour le moment, peu importe si elle vient d'un son que l'on cherche à détecter ou d'autre chose. Puisque l'on sait détecter des bruits, on pourra chercher à retrouver l'intensité maximale de chaque bruit dans un enregistrement.



```

donnees_pour_carte - Bloc-notes
Fichier Edition Format Affichage Aide
['x : un bout du titre de l image 0', 'y: un autre bout du titre de l image 0', t : encore un autre bout du titre de l'image 0, [false false True false false false false], l intensité
maximale a recueillir qq part
['x : un bout du titre de l image 1', 'y: un autre bout du titre de l image 1', t : encore un autre bout du titre de l'image 1, [false false True false false false false], l intensité
maximale a recueillir qq part
['x : un bout du titre de l image 2', 'y: un autre bout du titre de l image 2', t : encore un autre bout du titre de l'image 2, [false false True false false false false], l intensité
maximale a recueillir qq part
['x : un bout du titre de l image 3', 'y: un autre bout du titre de l image 3', t : encore un autre bout du titre de l'image 3, [false false True false false false false], l intensité
maximale a recueillir qq part
['x : un bout du titre de l image 4', 'y: un autre bout du titre de l image 4', t : encore un autre bout du titre de l'image 4, [false false True false false false false], l intensité
maximale a recueillir qq part

```

FIGURE 32 – Fichier «donnees pour carte» que l'on utilisera pour la cartographie.

Avec ces données, on peut tracer sur une carte les probabilités de présences des différents sons en différents points, et leurs intensité. La Cartographie permettra d'estimer ces paramètres en des points dont on n'a pas fait de mesures, façon à recouvrir tout un quartier de la ville de Brest, de façon continue.

6.3 Résultats

A ce stade, nous avons pu entraîner l'algorithme avec les données de M. CAZAU. Les résultats sont obtenus pour un jeu de données de 20 enregistrements de 10 secondes, regroupées en 7 classes. Comme on peut le voir, pour le moment, l'algorithme n'est pas performant. Nous avons modulé le réseau de neurones, de façon à voir comment il se comportait et mieux saisir l'impact de chaque paramètre dans l'apprentissage. Mais avec si peu de données et tant de classes, il est difficile de faire des prédictions correctes, et encore plus, de généraliser.

Comme le montre la figure 33, l'algorithme ne réussit pas à prédire les bons sons. Cela s'explique aisément par le fait que chaque image possède plusieurs labels, et que si un en manquant, ou un est en trop dans la prédiction, la prédiction est fausse. Cela arrive d'autant plus souvent que l'on s'entraîne sur 10 échantillons, et que l'on valide l'entraînement avec 5 échantillons, ceux-ci étant toujours les mêmes. L'algorithme n'a pas la possibilité de généraliser à d'autres configurations sonores, dont certaines lui sont d'ailleurs inconnues. Mais, Les données d'entraînements sont bonnes, cela signifie que l'algorithme parvient à apprendre, même s'il ne parvient pas du tout à généraliser. Par exemple, figure 33, l'accuracy de la 17ème epoch est à 1, tous les sons ont été correctement prédits.



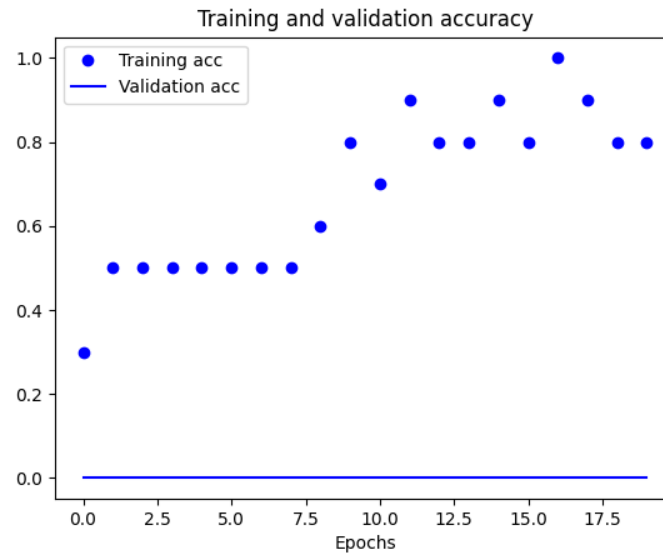


FIGURE 33 – Entraînement et Validation Accuracy

De même, la fonction de perte ne cesse de diminuer pour les données d'entraînements, prouvant la capacité d'apprentissage du réseau de neurones. Mais les pertes du jeu de validation ne sont pas concluantes. Elles diminuent jusqu'à l'époch 7, mais restent à un seuil élevé, et réaugmentent, ce qui témoigne de l'overfitting qui a lieu ensuite.

Nous tenons à préciser que la performance du réseau de neurone n'est pas le point sur lequel nous nous sommes concentrés jusqu'ici, et que nous nous focaliseront sur celui-ci une fois que nous aurons recueilli nos propres données. En effet, presque toutes les problématiques annexes au choix des paramètres de l'algorithme et de la structure du réseau de neurones ont été réglées dans cette première partie de projet.

Cependant qualité et la précision des prédictions dépendent de nombreux paramètres. Même si nous ne les avons pas tous optimisé à présent, il est important de les présenter ici, car c'est ce qui constitue la prochaine étape de notre projet.

6.3.1 Influence des classes choisies et de la variété des données

Selon le nombre de classes choisies et les caractéristiques de celles-ci, des biais peuvent être introduits dans les prédictions.

Par exemple, selon la fréquence d'apparition de certains bruits, les classes devront être modulées. Par exemple, si la voiture est présente sur chaque donnée, mais la moto sur une très petite proportion de données. Les spectrogrammes de ces deux sons sont assez semblables, et s'il on représentaient ces sons dans deux classes différentes, l'algorithme pourrait avoir du mal à reconnaître les motos, car avec la petite quantité de données de motos, il ne pourrait pas apprendre à repérer les subtiles différences entre les deux spectrogrammes. Il faudra alors peut-être regrouper ces deux classes, en "voiture ou moto". Mais rien n'est sûr pour le moment, cela dépend de l'affluence des motos dans les villes de Brest, et de la façon dont réagit le réseau de neurones.

En ce qui concerne les différents types d'oiseaux, la question de leur séparation en plusieurs classes peut se poser. En effet, la forme de leurs spectrogrammes se ressemblent, mais l'amplitude de fréquence



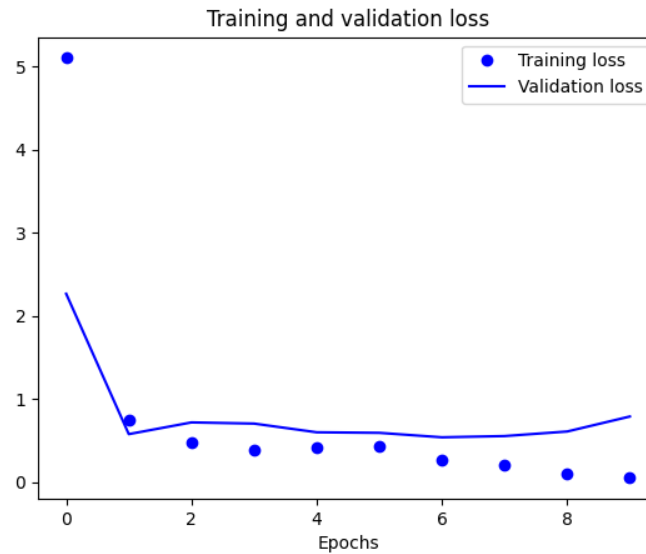


FIGURE 34 – Entrainement et Validation Loss

est différente entre les oiseaux. Si la fréquence d'apparition des sons appartenants aux potentielles futures classes d'oiseaux est suffisamment élevée pour chaque classe, et que nous sommes capables de différencier aisément les types d'oiseaux, nous les dissocieront en plusieurs classes. Cela dépend donc principalement de l'affluence des différents types d'oiseaux. De plus, si le réseau de neurones reconnaît sans peine n'importe quel oiseau, en tant que son de la classe "oiseau", nous n'aurons pas besoin de faire plusieurs classes. En effet, différencier les oiseaux n'est pas une fin en soi dans notre projet.

6.3.2 Influence de la struture du réseau de neurones convolutifs

De plus, la structure du réseau de neurones convolutifs joue un rôle important. Notre structure est adaptée à la classification multi-label, mais elle est basique. Elle est bien adaptée car la fonction d'activation, en sortie du réseau de neurones est la fonction Sigmoid qui permet, pour chaque classe, de donner une probabilité de présence de la classe sur l'image. De plus, la fonction de calcul des pertes est "binary crossentropy". Cela permet donc de repérer plusieurs classes sur la même image, si leur probabilité d'apparition dépasse un certain seuil. En effet, à contrario, la fonction d'activation Softmax, à une forme qui permet plutôt de distinguer une classe parmi les autres dans sa probabilité de présence.

Il nous faudra, pour la suite du projet, moduler chacun des paramètres du réseau et chacune de ses fonctions. Cela comprend les convolutions, la Max Pooling, le DropOut, et la modulation du seuil d'activation. Ceci permettra d'augmenter les permformance de notre modèle, en plus de l'accroissement des données et de la meilleure définition des classes, qui contribuerons également à cette meilleure performance.

6.3.3 Influence de la quantité de données

Nous n'avons travaillé que sur un jeu de 20 enregistrements, ce qui est bien plus faible que ce que nous donneront à l'algorithme une fois les enregistrements dans Brest réalisés. Il n'a pas pu ici saisir les subtiités des différents spectrogrammes. Nous avons prévu, pour notre relevé, d'obtenir XX enregistrements de X minutes, soit X enregistrements de 10 secondes. Cette quantité est significativement plus élevée que les 20 échantillons utilisés actuellement, et devrait permette des résultats bien plus performants.



Nous avons préféré travailler sur un petit jeu de données, de manière à bien comprendre le mécanisme de labellisation, de mise en forme des données, ainsi que le principe de l'algorithme mis en oeuvre. Même si les résultats, au stade de ce rendu intermédiaire, sont peu significatifs, notre démarche est prête pour la suite, et chaque rouage de notre système est prêt à fonctionner sur une grande quantité de données. Il restera à définir le nombre de classes et leurs valeurs, ainsi qu'à ajuster la structure du réseau de neurones, au fur et à mesure de l'apprentissage de l'algorithme.



7 Visualisation de la pollution sonore urbaine par de la cartographie

7.1 Estimation spatiale par krigeage

Au cours de l'étude menée jusqu'à présent, notre préoccupation a été de traiter et d'analyser des fichiers audios afin de reconnaître la source de l'onde sonore et son intensité. Désormais, nous devons visualiser les mesures effectuées pour faciliter l'utilisateur à visionner la pollution sonore urbaine. Nous nous contenterons de réaliser la carte sur le quartier Saint-Martin de Brest, où seront faites les mesures expérimentales. La première partie consiste à définir dans quelle région les mesures sont réalisées. Par ailleurs, pour chaque site de mesure, nous devons localiser et dater les mesures réalisées. Pour la localisation, nous utilisons le système de projection WGS84. Ce système est identifié par son code EPSG 3857. Certes, une projection Lambert serait plus précise à Brest. Nous justifions ce choix pour deux raisons. La première est que les données recueillies pour se positionner sont issues du système GPS, utilisant déjà cette projection. Ces données sont fournies soit par l'application NMEA Tools sur Android en mesure absolu, soit par Google Maps, dont la précision est de l'ordre du mètre. Cette incertitude métrique est supérieure à celle commise par la projection dans le quartier étudié. L'utilisation d'une projection Lambert n'aurait pas améliorée la précision sur la localisation [2]. Pour la date des mesures, elles sont renseignées par l'audiomoth. Ces données, localisation et date, sont tout au long du traitement attachées à chaque fichier audio.

La première visualisation auxquelles l'utilisateur est confrontée est l'intensité de la pollution sonore maximale. Les mesures expérimentales étant réalisées pour un nombre de site réduit, un premier traitement consiste à estimer statistiquement cette intensité sonore sur l'ensemble du quartier avec une précision longitude de $d\phi = 100$ et en latitude de $d\theta = 10$. Plus $d\phi$ et $d\theta$ sont faibles, plus la visualisation sera précise pour l'utilisateur puisque à défaut de pouvoir faire une estimation continue sur R^2 , le nombre de site estimé sera suffisamment grand pour s'approcher de cette continuité. Sur la zone étudiée, cela correspond à 40 estimations sur la longitude pour θ fixé, et 13 estimations sur la latitude pour ϕ fixé. Soit 520 estimations sur la zone étudiée. Bien évidemment, plus le nombre d'estimation augmente, plus le temps de calcul et le stockage s'accroissent. Un exemple de cette estimation est présenté ci-dessous en utilisant des valeurs en entrée choisis arbitrairement.

Estimation par krigeage ordinaire.

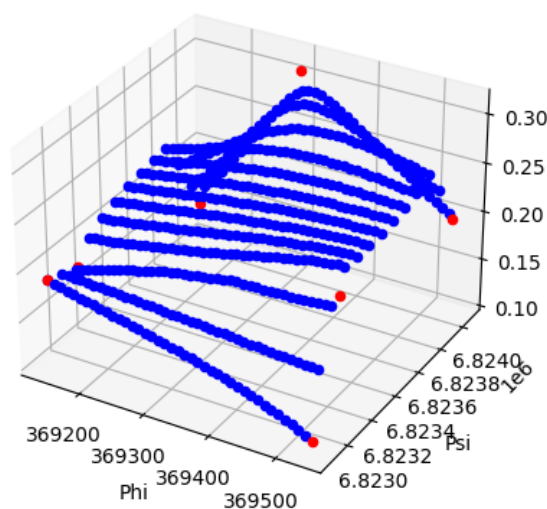


FIGURE 35 – Estimation par krigeage ordinaire. Les points rouges sont des mesures expérimentales, les points bleus sont les estimations par géostatistique.



Comme présenté précédemment, nous utilisons une estimation par krigeage ordinaire, ce qui implique que l'intensité sonore maximal est une fonction aléatoire strictement intrinsèque. Les deux étapes sont donc l'analyse variographique et le krigeage ordinaire, qui sont réalisés l'une après l'autre. Avant tout, une analyse variographique est nécessaire pour déterminer l'influence qu'un site a sur un site voisin. Le variogramme est approché expérimentalement, avec les mesures réalisées par l'audiomoth, dans le fichier Python *variogramme.py*. Ensuite, le krigeage ordinaire est effectué dans le programme *krigeage.py*. Ce programme comporte plusieurs paramètres à modifier si nécessaire. La fonction f correspondant au variogramme. La zone étudiée est déterminée par les paramètres ψ_{Min} , ψ_{Max} , ϕ_{Min} , ϕ_{Max} . La précision, déterminant le nombre d'estimation à effectuer, par $d\psi$ et $d\phi$. L'algorithme extrait les données stockées dans le fichier texte décrit précédemment, puis pour chaque site à estimer, il résout le système matricielle (4).

Les estimations réalisées permettent à la fois d'avoir une estimation de l'intensité sonore et la variance sur cette mesure. Ces données sont stockées dans un fichier texte *krigeageO.txt*. Ces données sont les suivantes, latitude, longitude, intensité maximal, variance d'estimation.

7.2 Visualisation cartographique

Toutes les données nécessaires à l'élaboration d'une carte de l'intensité sonore maximal sont stockées dans le fichier *krigeageO.txt*. La carte est réalisée sous le logiciel QGIS, un système d'information géographique. Ce programme permet de géoréférencer les données traitées tout au long du processus expliqué précédemment. La première étape reprend ce dont nous avons expliqué au début de cette partie à propos de la région étudiée, du système de projection utilisé. Ces informations sont préalablement renseignées lors de la création d'un projet sous QGIS, nommé *PSB*. Ensuite, plusieurs couches sont ajoutées, dont chacune représente une information particulière. Dans le cas de ce projet, une première couche est ajoutée faisant office de couche de fond pour que l'utilisateur puisse se repérer dans le quartier de Brest choisi. Cette couche provient d'un serveur d'OpenStreetMap. Elle est stockée en tant qu'une couche raster.

Les couches au-dessus correspondent aux données traitées dans le projet. La première couche représente les mesures expérimentales sous la symbologie de point. Ces données sont récupérées à partir du fichier texte!!! et convertie en une couche vectorielle. Ces points sont de couleurs différentes, informant sur le niveau d'intensité sonore maximal récupéré sur le site. La couche en dessous, est également une couche vectorielle sous forme de point, correspondant aux intensités sonores maximales estimées par le krigeage ordinaire. Elle renseigne par la couleur sur le niveau de l'intensité.

La couche en dessous représente les polygones de Varonoï. Ces figures sont créées à partir des points de l'estimation par krigeage ordinaire. Pour tout polygone P_i sur D est associé un site si d'une estimation. Un point s appartient à un polygone P_j si parmi tous les sites si estimés, le site le plus proche de s est s_j . Un algorithme intégré dans le logiciel QGIS effectue ce traitement. Il prend en entrée une couche vectorielle, dans ce cas les estimations par krigeage ordinaire et renvoie une couche vectorielle de polygone. Dans le cas de l'étude géostatistique réalisée, l'estimation a été faite sur une grille régulière, pour des pas le long de la latitude et de la longitude constantes. Donc, les polygones de Varonoï sont des rectangles dont les longueurs sont égales à $d\phi$ et $d\theta$.

7.3 Interface avec l'utilisateur avec une page web

A partir de la carte réalisée sous QGIS, les données localisées doivent être partagées. A l'aide de l'extension qgis2web, une page web est créée. L'avantage de cette page web est qu'elle peut être partagée facilement d'un utilisateur à un autre, sans logiciel préalable.

La carte affichée est interactive, ce qui permet d'apprécier les détails au choix de l'observateur. Les couches peuvent être masquer ou afficher en fonction de ce dont on souhaite. De plus, en cliquant sur les mesures expérimentales, représentées par des points, des données sont affichées dans une fenêtre, dont la position géographique et la valeur de l'intensité sonore maximale.



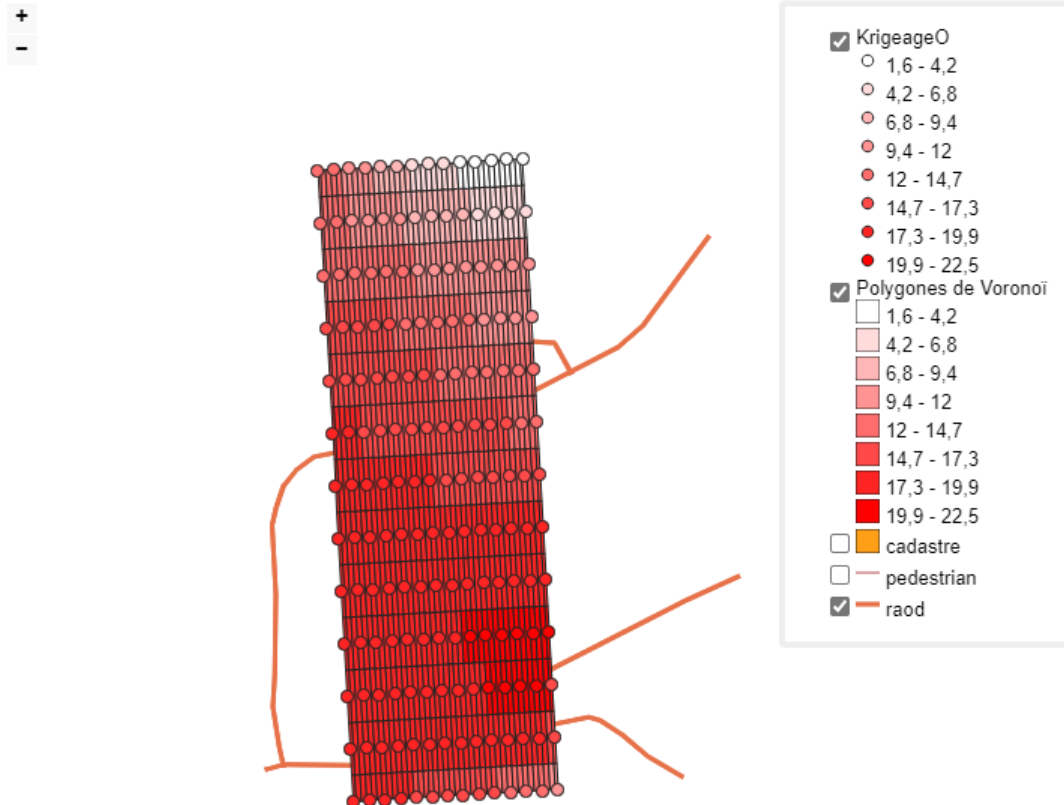


FIGURE 36 – Carte interactive issue du krigeage ordinaire. Les points représentent les 520 estimations par krigeage ordinaire. Les polygones de Varanoï sont des rectangles.

8 Conclusion

Les deux propositions que nous avons définies dès le début de ce projet, l'identification et la localisation, ont été prises en compte pour former le système exposé. Les phases n'ont par toutes été traitées avec un avancement égal, mais il reste suffisant pour affirmer que le système présenté peut répondre aux contraintes fixées dans le cahier des charges.

Avant de développer un logiciel informel comprenant chacune des phases du projet, nous nous sommes avant tout appliqués à s'assurer du bon fonctionnement des nombreux liens entre chaque partie. Par exemple, en créant des bases de données dont la forme et le contenu peuvent à la fois récupérer des données en entrée et les transmettre à une autre partie du projet en sortie.

Puis, en développant plusieurs programmes pour chaque partie, comme pour l'apprentissage automatique de reconnaissance ou d'estimation par krigeage, nous avons pu commencer à manipuler des concepts théoriques. Par exemple, en utilisant des premiers jeux de données pour tester l'algorithme de deep learning ou encore en estimant l'intensité sonore en un site distinct des zones de mesures expérimentales. Les données d'entrée n'étaient pas des données provenant de mesures expérimentales dans la zone étudiée. Ces données avaient l'objectif seulement de voir le fonctionnement des algorithmes utilisés et de simuler le processus de traitement, de la création d'un sonogramme avec toutes les données de localisation et de date, jusqu'à la mise en oeuvre d'une carte interactive intégrée sur une page web.

Les prochaines étapes auront pour but de trouver des paramètres si possible optimaux pour le réseau de neurone, et d'améliorer le krigeage par des algorithmes prenant en compte des phénomènes non stationnaires. Enfin, la récolte de données présentées devra être mise en oeuvre pour pouvoir donner des données réelles aux différentes parties du système.



Table des figures

1	Diagramme du cycle de vie du système séparé en quatre phases.	7
2	Diagramme en pieuvre de la première phase de vie du système.	9
3	Diagramme en pieuvre de la deuxième phase de vie du système.	10
4	Diagramme en pieuvre de la troisième phase de vie du système.	10
5	Diagramme en pieuvre de la quatrième phase de vie du système.	11
6	Diagrammes internes FAST de la première phase de vie.	12
7	Diagramme interne FAST de la deuxième phase de vie.	12
8	Diagrammes internes FAST de la troisième phase de vie.	13
9	Diagrammes internes FAST de la quatrième phase de vie.	13
10	Diagramme de l'architecture physique du système.	14
11	Le diagramme de Gantt était un premier aperçu sur les étapes du projet.	15
12	Machine Learning, nouveau paradigme de programmation.	16
13	Classification	18
14	Clustering	18
15	Régression	18
16	Neurone naturel simplifié	18
17	Le perceptron	18
18	Deux classes	19
19	Quatre classes	19
20	Perceptrons multicouches dans des problèmes de classification	19
21	Exemple de sur apprentissage	20
22	Exemple de sous apprentissage	20
23	Fonctionnement du Max Pooling.	21
24	Schéma du krigeage sur une dimension temporelle. Les poids $\lambda_1(t)$ et $\lambda_2(t)$ peuvent être Interprétés comme l'influence exercé par les mesures aux instants t1 et t2 sur l'estimation à t.	23
25	Krigeage ordinaire en une dimension temporelle. Les graphiques du haut montrent le choix d'interpolation sur le variogramme expérimental représenté par les points rouges. Les figures du bas permettent de visualiser l'estimation par krigeage à partir des deux mesures expérimentales en rouge. On remarque que le variogramme parabolique n'est pas valide. Le modèle linéaire et gaussien ont une erreur de 0.333 et 0.065 respectivement, suggérant que le choix d'un variogramme gaussien est favorable dans ce cas, malgré son interpolation moins optimal avec le variogramme expérimental.	25
26	Paramétrage de l'audiomoth	26
27	Interface permettant de labelliser les échantillons sur Python, en les découpant.	27
28	Fichier «data» créé après labellisation des échantillons, associant à chaque spectrogramme ses labels.	28
29	Fichier «listClass» créé après labellisation des échantillons, contenant tous les labels.	28
30	Spectrogramme d'un échantillon, affichant l'intensité en fonction de la fréquence et du temps.	29
31	Réseau de neurones convolutifs mis en place.	29
32	Fichier «donnees pour carte» que l'on utilisera pour la cartographie.	30
33	Entraînement et Validation Accuracy	31
34	Entraînement et Validation Loss	32
35	Estimation par krigeage ordinaire. Les points rouges sont des mesures expérimentales, les points bleus sont les estimations par géostatistique.	34
36	Carte interactive issue du krigeage ordinaire. Les points représentent les 520 estimations par krigeage ordinaire. Les polygones de Varanoï sont des rectangles.	36



Références

- [1] Nathan Belval, « Aménager l'environnement sonore : trois compositeurs à l'épreuve de l'urbanisme », thèse en aménagement et urbanisme, juin 2020.
- [2] Pierre Bosser, « Géodésie : Notions Fondamentales », 8 octobre 2020.
- [3] Pierre Bosser, « Introduction à l'interpolation spatiale et aux géostatistiques », Chapitre 3, 6 janvier 2020.
- [4] Pierre Chauvet, « Aide mémoire de géostatistique linéaire », Presses Des Mines, avril 2008.
- [5] Chantal De Fouquet, « Exercices corrigés de géostatistique », Presses Des Mines, pages 231-243, mail 2019.
- [6] F. Chollet, « Deep Learning with Python », 1st édition. Shelter Island, New York: Manning Publications, 2017.
- [7] J. DAYHOFF, « Neural Network Architectures », 1990, Hardcover – June 1.
- [8] A. Géron, « Deep Learning avec Keras et TensorFlow: Mise en oeuvre et cas concrets », Dunod, 2020. 6.

