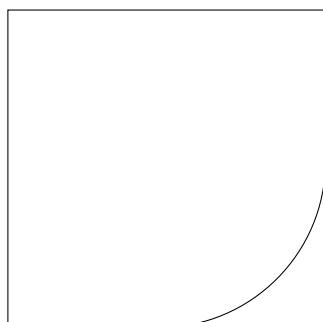


ENSTA Bretagne
2, rue François Verny
29806 BREST cedex
FRANCE
Tel +33 (0)2 98 34 88 00
www.ensta-bretagne.fr



Projet Système
Rapport
d'avancement
SOIA 2023
26 avril 2022



Pollution sonore urbaine

Implémentation d'une méthode de

Machine Learning

Réalisation d'une carte sonore



Préface

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. Measurements by an audio sensor capable, after a statistical analysis, of constituting a map allowing the maximum sound intensity that can be encountered in the study space.

Résumé

L'objectif de ce projet est de constituer une carte sonore dans 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. Des mesures par un capteur audio permettront, après une analyse statistique, de constituer une carte affichant l'intensité sonore maximale que l'on peut rencontrer dans l'espace d'étude.



Table des matières

1 Remerciements	5
2 Introduction	6
3 Organisation générale	7
3.1 Cahier des charges	7
3.2 Analyse du système par l'ingénierie système	9
3.2.1 Analyse externe du système	9
3.2.2 Analyse interne du système	13
3.2.3 Architecture physique du système	16
3.3 Diagramme de Gantt	17
3.4 Organisation de groupe par la méthode Agile	17
4 État de l'art	18
4.1 Deep Learning et Réseaux de Neurones	18
4.1.1 Prétraitement du spectrogramme	18
4.1.2 Réseaux de neurones pré-entraînés	19
4.2 Indices de performance pour la classification	20
4.2.1 La matrice de confusion	20
4.2.2 Classification Accuracy, fausse bonne idée	20
4.2.3 Précision, rappel et F1 score	21
4.2.4 Sensibilité, spécificité et courbe ROC	21
4.3 Estimation spatiale	22
5 Traitement des données audios par deep learning	26
5.1 Mise en forme des données	26
5.1.1 Choix du type de spectrogramme (explication log etc, différents choix)	27
5.1.2 Augmentation de données	28
5.1.3 Dataset	30
5.2 Exploitation des données par méthode de Deep Learning	31
5.2.1 Premiers réseaux de neurones utilisés	31
5.2.2 Réseau de neurones pré-entraînés	32
5.2.3 Réalisation des prédictions et passage à la cartographie	32
5.3 Résultats	33
5.3.1 Analyse de résultats via la Matrice de confusion	33
5.3.2 Conclusions sur la partie prédition	34
6 Visualisation de la pollution sonore urbaine par cartographie	35
6.1 Les moyens pour transmettre l'information par la cartographie	35
6.2 Quelques précisions sur la démarche	36
6.3 Traitement des données	39
6.3.1 Krigeage	40
7 Tests réalisés sur le système	43
7.1 Tests unitaires	43
7.2 Tests d'intégration	43



8 Réalisation de la partie Web	44
8.1 Introduction	44
8.2 Identification des acteurs	44
8.3 Identification des besoins fonctionnels	44
8.3.1 Besoins fonctionnels de l'administrateur	44
8.3.2 Besoins fonctionnels de l'utilisateur	44
8.4 Identification des besoins non fonctionnels	44
8.5 Analyse des besoins	45
8.5.1 Cas d'utilisation « S'authentifier »	45
8.5.2 Cas d'utilisation « Gérer les emplacements des capteurs»	46
8.6 Réalisation	47
8.6.1 Environnement de développement	47
9 Bilan sur la gestion de projet	49
9.1 Diagramme de Gantt	49
9.2 Organisation de groupe par la méthode Agile	49
10 Conclusion	50



1 Remerciements

Tout d'abord, nous remercions M.CAZAU, notre encadrant principal, pour son accompagnement et son implication tout au long de nos séances de projet. Ses nombreux et précieux conseils nous ont permis d'avancer dans ce projet.

Nous tenons également à remercier Mme. THOMAS, responsable du module d'enseignement lié au projet, pour son écoute attentive et sa disponibilité.

Enfin, nous remercions M. GUERIN, notre consultant en méthode Agile, pour nous avoir enseigné le fonctionnement de la méthodologie Agile et nous avoir posé des nombreuses questions pertinentes nous permettant de prendre du recul sur ce projet.

Nous remercions également M. THEODOROV, enseignant en ingénierie système, de nous avoir expliqué comment réaliser la phase de conception d'un produit en utilisant le logiciel IBM Rhapsody.



2 Introduction

Le conseil de la colonie grecque de Sybaris au VIème 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 prononcé 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 identifié 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 prolongé 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.

Afin d'avoir un confort acoustique en ville, plusieurs travaux ont tenté de quantifier les sources de pollution sonore, de les limiter, de les 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, puis de les quantifier. D'une part, il est difficile de les caractériser du fait qu'une pollution sonore peut provenir de sources aussi exotiques que notre imagination. D'autre part, la quantification de ces sources posent d'autres problèmes, comment peut-on faire des mesures sur l'ensemble d'une agglomération, dont les conditions changent d'heure en heure, si ce n'est que la ville elle-même mute dans le temps en connaissant l'explosion de l'urbanisation ces dernières années. Notre étude est bien moins ambitieuse. Elle se contentera de caractériser des sources que l'on rencontre régulièrement en ville, comme des véhicules. Elle quantifiera les pollutions sonores à l'échelle de quelques bâtiments.

Le rapport présent fait suite au rapport d'avancement de janvier 2022. Il rend compte de l'avancement du projet pollution sonore urbaine depuis le précédent rendu. Les deux objectifs principaux du projet sont de déterminer les sources et leur intensité sonore en milieu urbain qui peuvent engendrer des gênes et dans le pire des cas des risques pour la santé des habitants locaux. Deux axes ont été définis tôt dans ce projet : une partie de deep learning, s'étalant de la récolte des données audios à leurs traitements par un algorithme d'intelligence artificielle. Son but répond au premier objectif, déterminer, caractériser les sources de pollution sonore. Cette première étape doit nous permettre de préciser de façon automatique la provenance d'un son perçu dans un milieu urbain, comme une voiture ou un oiseau. La seconde partie répond à l'objectif de déterminer l'intensité de ces pollutions sonores. Elle consiste tout particulièrement à faciliter la visualisation de ces pollutions sonores dans un quartier, afin d'aider par exemple un décideur public à prendre des mesures pour atténuer une pollution sonore indésirable dans un quartier d'une ville. Par la cartographie, nous donnerons un aperçu de ces pollutions sonores, permettant de faciliter la compréhension de la provenance et des conséquences 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 devions 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.

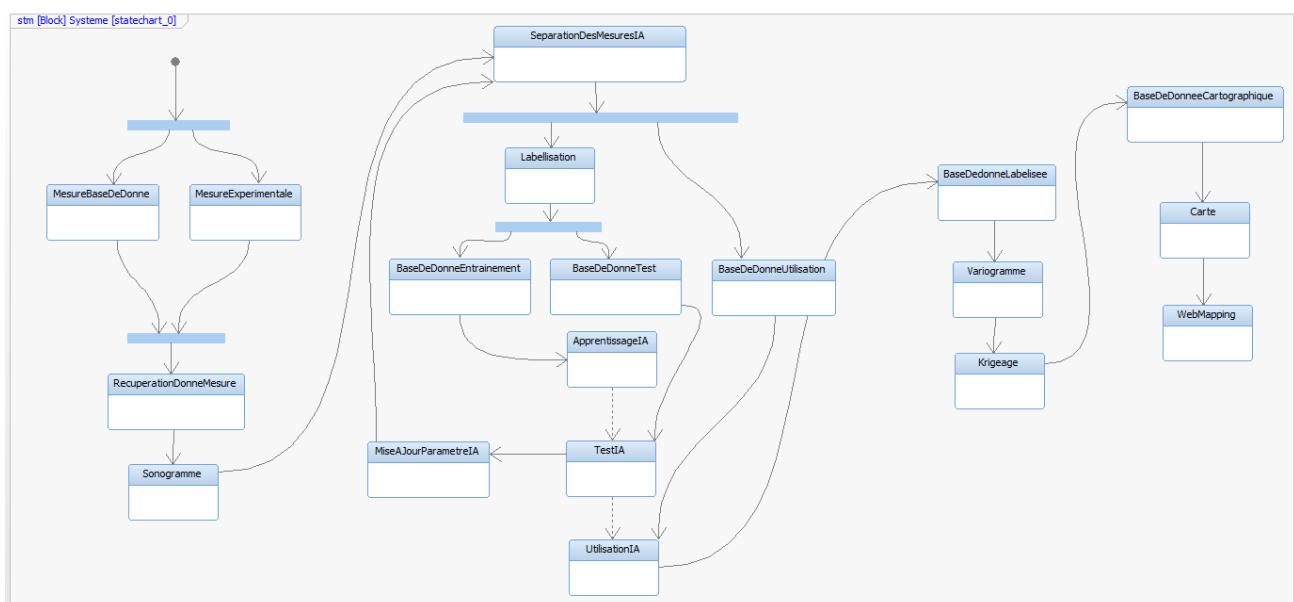


FIGURE 1 – Diagramme du cycle de vie du système séparé en quatre phases.



-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 receuillies 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 audiometer, 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 krigage. 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 krigage, 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 Audiometer 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, comme 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.

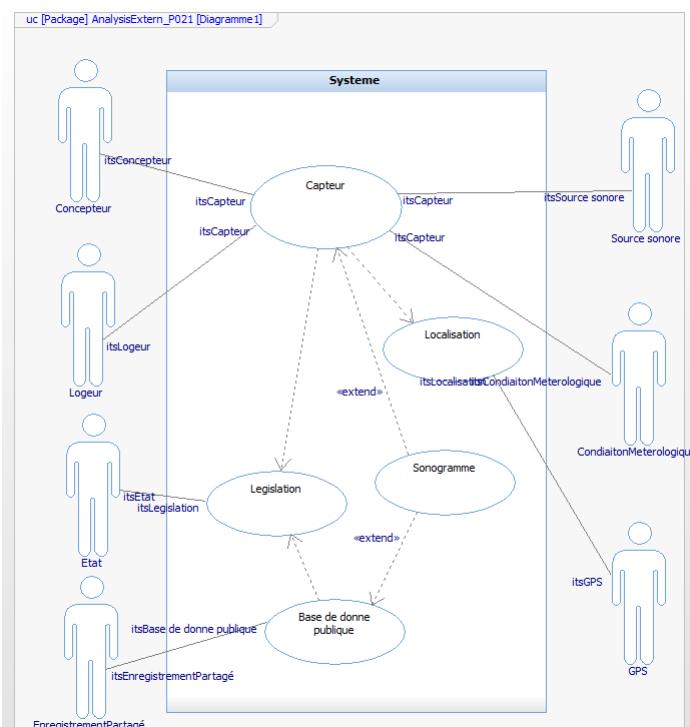


FIGURE 2 – Diagramme en pieuvre de la première phase de vie du système.

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 classifier 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.

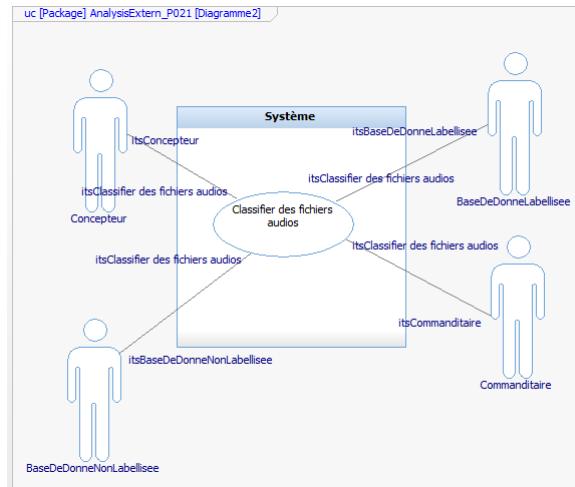


FIGURE 3 – Diagramme en pieuvre de la deuxième phase de vie du système.

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 krigage. 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 krigage.

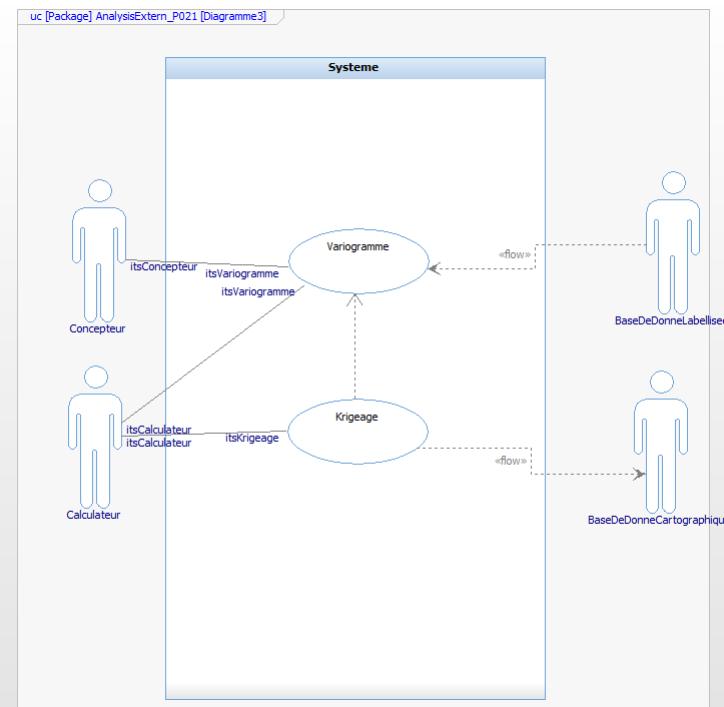


FIGURE 4 – Diagramme en pieuvre de la troisième phase de vie du système.

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.

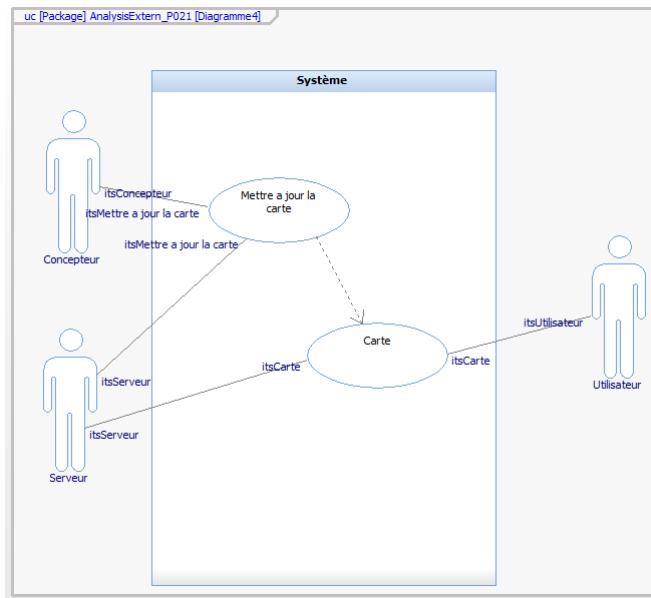


FIGURE 5 – Diagramme en pieuvre de la quatrième phase de vie du système.

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 classifier 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.



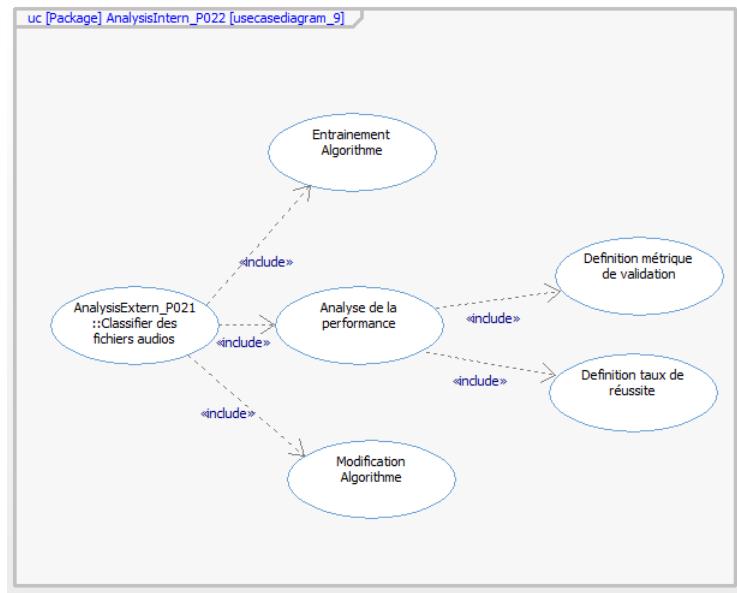


FIGURE 7 – Diagramme interne FAST de la deuxième phase de vie.

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

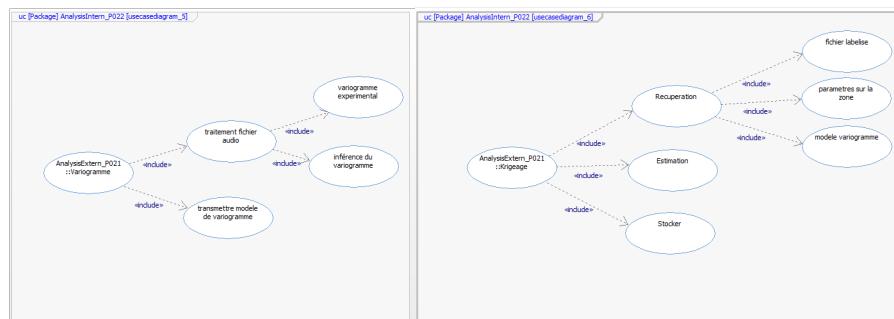


FIGURE 8 – Diagrammes internes FAST de la troisième phase de vie.

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



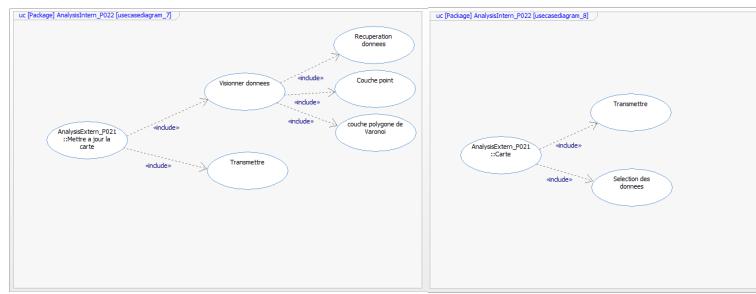


FIGURE 9 – Diagrammes internes FAST de la quatrième phase de vie.

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 krigage. 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 partagé à un utilisateur.

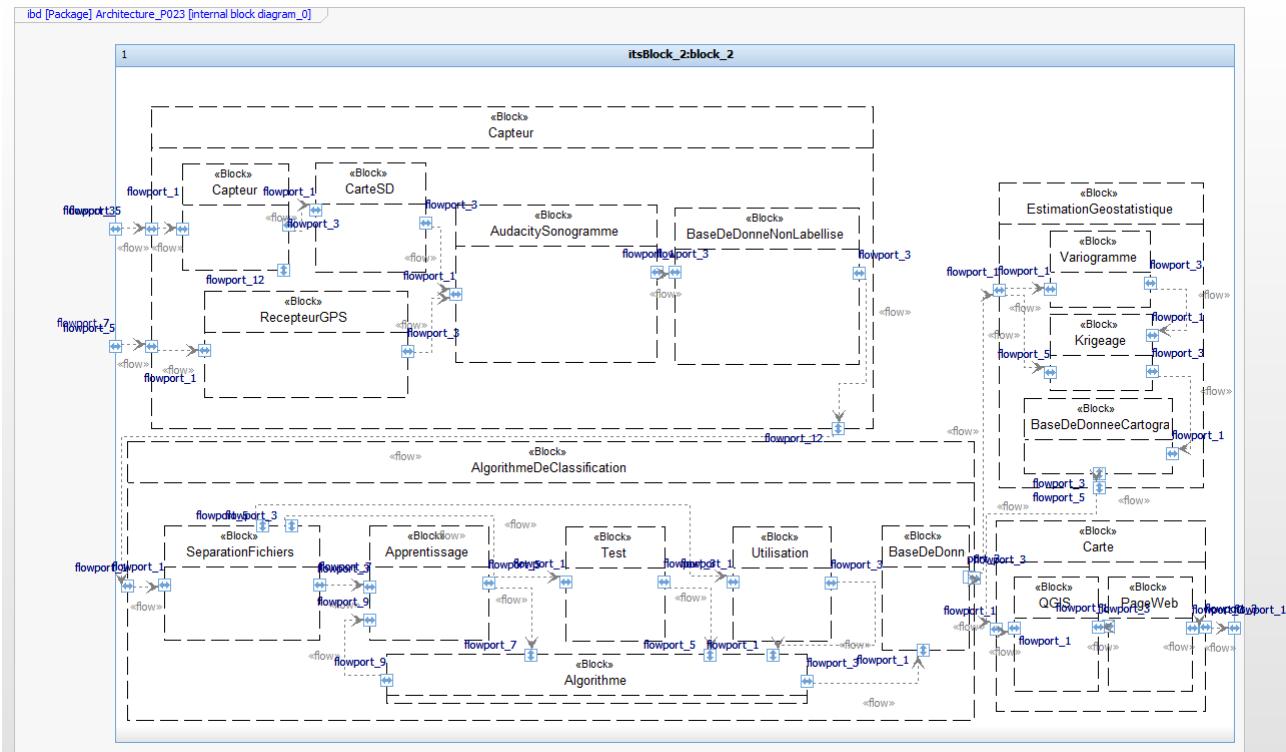


FIGURE 10 – Diagramme de l'architecture physique du système.



3.3 Diagramme de Gantt

3.4 Organisation de groupe par la méthode Agile



4 État de l'art

4.1 Deep Learning et Réseaux de Neurones

Les notions générales de deep learning ont été abordées dans le rapport d'avancement, du semestre 3. Cependant, nous n'avons pas cessé de rechercher si d'autres méthodes, que nous ne connaissons pas encore, pourraient nous être utiles. Celles-ci sont présentées dans les paragraphes suivants.

4.1.1 Prétraitement du spectrogramme

On peut se demander s'il est nécessaire, ou au moins avantageux, de réaliser un pré-traitement des spectrogrammes avant leur utilisation dans le réseau de neurone. Ce traitement d'image peut avoir deux buts opposés, soit améliorer l'image en atténuant le bruit, donc en augmentant le SNR, soit détériorer le spectrogramme pour augmenter le dataset. C'est une opération compliquée que nous avons essayé de réaliser, mais les résultats sont peu convaincants.

Une méthode est de détériorer l'image, en augmentant le bruit, donc en diminuant le SNR. L'objectif dans ce cas est d'augmenter le dataset qui sera utilisé par le réseau de neurone. Par exemple, à partir d'un spectrogramme, on peut réaliser un filtrage de ce dernier afin d'ajouter un ou plusieurs images du même spectrogramme mais avec un ou des SNR plus faible. Cette opération permet d'augmenter la taille du dataset, de répondre aux problèmes du faible nombre de données que nous pouvons rencontrer.

On peut aussi imaginer l'utilisation de nombreux filtres, plus ou moins complexes. Premièrement, puisque le but est de détériorer le SNR en gardant la structure du spectrogramme, pour que le réseau de neurones puisse la repérer malgré l'augmentation du bruit, on préférera utiliser un filtrage simple. On a donc choisi l'utilisation d'opérations ponctuelles, c'est-à-dire un filtre s'impliquant pixel par pixel. De plus, les opérations de convolution sont particulièrement utilisées en traitement d'image, donc nous choisirons un filtre par convolution. Enfin, le but étant d'augmenter le bruit tout en gardant la structure du spectrogramme initial, nous devons préférer l'utilisation d'un filtre passe bas, qui rejette les hautes fréquences. L'ajout de hautes fréquences n'est pas souhaitable afin de préserver la structure. Nous allons présenter quelques filtres ponctuels, par convolution et passe bas. [11]

Le premier filtre est le filtre moyenneur. Son principe est de donner la valeur moyenne à chaque pixel des ses pixels voisins. Les pixels voisins sont à définir, par exemple en prenant tous les pixels situés à une distance pixel inférieure à P , comme dans la formule suivante.
$$h(x, y) = \begin{cases} \frac{1}{(2P+1)^2} \sum_{si-P < x,y < P} s_i, & \text{si } P > 0 \\ 0, & \text{sinon} \end{cases}$$

Un filtre gaussien peut également être utilisé. Deux paramètres sont ici à choisir, le nombre de voisin et la variance de la loi gaussienne, supposée centrée. La variance dépend de la fréquence de coupure. On peut donc à partir de la fréquence de coupure déterminer la variance. Le nombre de pixel voisin sur lesquelles s'applique à filtre est lui optimal en fonction de la variance. Il n'y a donc qu'un seul paramètre à choisir, la fréquence de coupure. Sa formule est la suivante :
$$h(x, y) = \frac{1}{1\pi\sigma^2} \exp\left(-\frac{(x^2+y^2)}{2\sigma^2}\right).$$

Avec comme fréquence de coupure.

$$\gamma_c = \frac{0.187}{\sigma}$$

L'inconvénient de l'utilisation de ces deux filtres est qu'il réalise des opérations identiques dans toutes les orientations, selon la longueur et la hauteur de l'image. Sur un spectrogramme, cela se traduit par des opérations identiques selon le temps, l'abscisse, et les fréquences, l'ordonnée. On peut se demander si on doit réaliser des opérations identiques pour deux grandeurs non semblables. Risque-t-on de déformer la structure reconnue par le réseau de neurones par des opérations sur le temps et la fréquence ? Un filtre de Gabor peut être utilisé comme un filtre gaussien, mais avec des orientations privilégiées, soit une loi gaussienne en deux dimensions pour une image. Dans notre cas, nous considérons qu'il ne faut limiter la détérioration du spectrogramme temporellement, cela ajouteraI



un effet de résonnance. Par contre, on peut détériorer fréquemment le spectrogramme, comme ci on diminuait sa résolution fréquentielle. Le filtre de Gabor utilise donc un paramètre supplémentaire, γ dans le programme fourni. $h(x, y) = \exp(-2j\pi(x + y) + \phi) \exp\left(-\left(\frac{(x-x_0)^2}{\sigma_x^2} + \frac{(y-y_0)^2}{\sigma_y^2}\right)\right)$

4.1.2 Réseaux de neurones pré-entraînés

Nous avions présenté dans le rapport précédent les réseaux de neurones convolutifs et différentes techniques permettant de maximiser l'efficacité de ceux-ci : la convolution, le MaxPooling, le DropOut. Ainsi, les nombreux paramètres du réseau s'ajustaient au fur et à mesure des batchs (lot d'échantillons) et des epochs (nombre de fois que l'on apprend sur tout les échantillons), par descente de gradient. Ainsi une epoch peut contenir plusieurs batchs, si ceux-ci étant de taille inférieure au nombre total d'échantillon d'apprentissage.

Dans le réseau, au lieu de moduler soi-même tous les paramètres de notre réseau (et parce que la performance peut être mauvaise, avec un petit dataset), on peut mettre en œuvre la technique de "transfert learning", en utilisant des réseaux de neurones convolutionnels pré-entraînés. Concrètement, on exploite les valeurs de paramètres d'un certain modèle, calculés antérieurement sur un problème de classification général et avec un dataset de grande taille, pour l'appliquer à notre problème particulier, ne disposant potentiellement que d'un petit dataset. Différents modèles sont disponibles dans la bibliothèque Keras, comme le modèle VGG-16, fréquemment utilisé.

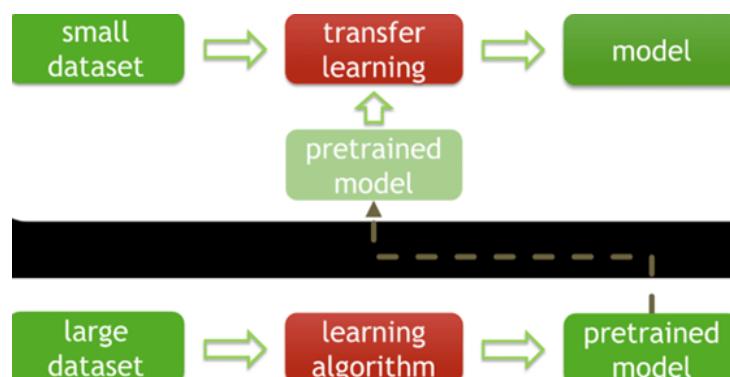


FIGURE 11 – Insertion d'un réseau de neurones pré-entraîné pour un problème particulier de classification

On peut ensuite ajouter une couche de neurones connectés de manière dense (classifieur), juste après le réseau pré-entraîné, et juste avant la dernière couche. Cette dernière couche contient autant de neurones que de classes (comme toujours en sortie de réseau, chaque neurone désignant la probabilité de présence d'une classe précise).

Ainsi le réseau pré-entraîné a déjà la capacité de reconnaître certains patterns, plus ou moins génériques et précis. Pour garder le bénéfice d'un apprentissage en amont (gain de temps et de qualité, surtout en cas de petit dataset), on fige les valeurs de paramètres pour les couches situées le plus en entrée. Ceux plus en profondeur détectent des patterns de plus en plus précis, voir trop (dans notre cas on ne détecte pas les mêmes patterns que pour un visage par exemple). Ils ne sont alors pas figés et pourront évoluer comme des neurones "classiques".

Cette technique a été mise en œuvre, nous la présenterons plus précisément par la suite.



4.2 Indices de performance pour la classification

Nous avons cherché à définir des mesures de performances nous permettant par la suite de décrire avec pertinence les résultats de notre réseau de neurones, dans le cas d'une classification binaire (l'extension à une classification multilabels étant instantanée avec le 1 versus all).

4.2.1 La matrice de confusion

La matrice de confusion (en anglais confusion matrix) n'est pas un indice de performance en soi, c'est un outil de visualisation de données utile en classification. La matrice de confusion représente le nombre (ou le taux) de prédiction d'une classe en fonction de la classe réelle.

The Confusion Matrix

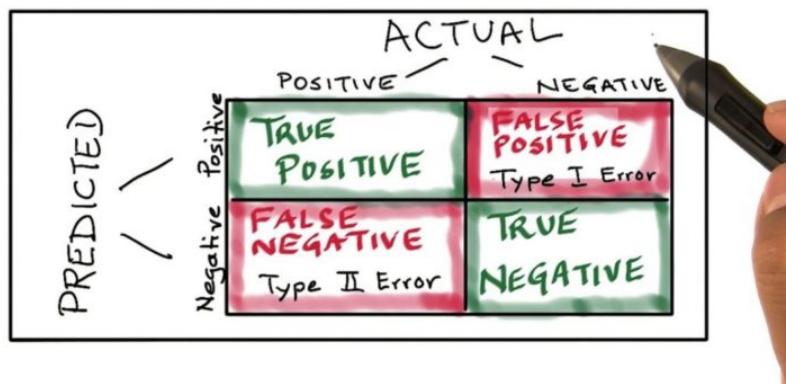


FIGURE 12 – Matrice de confusion - Classification binaire

source : <https://www.lebigdata.fr/confusion-matrix-definition>

Pour l'implémenter, il est nécessaire d'avoir y_true et y_pred :

```

1 import matplotlib.pyplot as plt
2 import tensorflow as tf
3 import seaborn as sns
4
5 confusion_mtx = tf.math.confusion_matrix(y_true, y_pred)
6 plt.figure(figsize=(10, 8))
7 sns.heatmap(confusion_mtx,
8             xticklabels=commands,
9             yticklabels=commands,
10            annot=True, fmt='g')
11 plt.xlabel('Prediction')
12 plt.ylabel('Label')
13 plt.show()

```

Listing 1 – Matrice de confusion

documentation :

https://www.tensorflow.org/api_docs/python/tf/math/confusion_matrix
<https://seaborn.pydata.org/generated/seaborn.heatmap.html>

4.2.2 Classification Accuracy, fausse bonne idée

L'accuracy est le nombre de prédictions correctes divisé par le nombre total de prédictions.



$$\text{Accuracy} = \frac{\text{Correct}}{\text{Correct} + \text{Incorrect}} = \frac{\text{Correct}}{\text{Total}}$$

Il existe de nombreux cas dans lesquels l'accuracy n'est pas un bon indicateur des performances d'un modèle. L'un de ces scénarios est celui où la distribution des classes est déséquilibrée (une classe est plus fréquente que les autres). Dans ce cas, prédire tous les échantillons comme appartenant à la classe la plus fréquente donnera nécessairement un taux de précision élevé, ce qui n'a aucun sens (car le modèle n'apprend rien et se contente de prédire tout comme la classe supérieure). Dès lors, il faut trouver une nouvelle métrique.

4.2.3 Précision, rappel et F1 score

La précision est le nombre de bonnes prédictions (*True Positive*, *TP*) divisé par la somme des bonnes et mauvaises prédictions (*False Positive*, *FP*).

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

La précision permet de mesurer le coût des faux positifs, c'est-à-dire ceux détectés par erreur. Si l'on cherche à limiter les faux positifs, c'est cet indicateur que l'on va chercher à minimiser.

Le rappel (ou sensibilité)

$$\text{Rappel} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

Le calcul du rappel permet d'estimer combien de documents réellement positifs nous avons réussi à capturer et combien sont passés au travers des mailles du filet. Application : détection de cancer, fraude bancaire nécessitent un rappel qui tend vers 1.

Ces deux indicateurs (précision et rappel) permettent de mesurer l'impact des faux négatifs et faux positifs sur notre modèle. Il existe un indicateur permettant de mesurer la globalité du modèle : le F1 Score.

Le F1-Score est plus pertinent que l'accuracy notamment dans le cas d'une distribution inégale des classes.

$$F_1\text{score} = 2 \cdot \frac{\text{Precision} \cdot \text{Rappel}}{\text{Precision} + \text{Rappel}}$$

4.2.4 Sensibilité, spécificité et courbe ROC

Une courbe ROC (receiver operating characteristic) est un graphique représentant les performances d'un modèle de classification pour tous les seuils de classification. Le seuil est généralement défini dans les algorithmes de ML entre 0 et 1 et représente la probabilité qu'un objet appartienne à la classe 1 (dans le cas d'une classification binaire).

En plus de la sensibilité, on définit la spécifité :

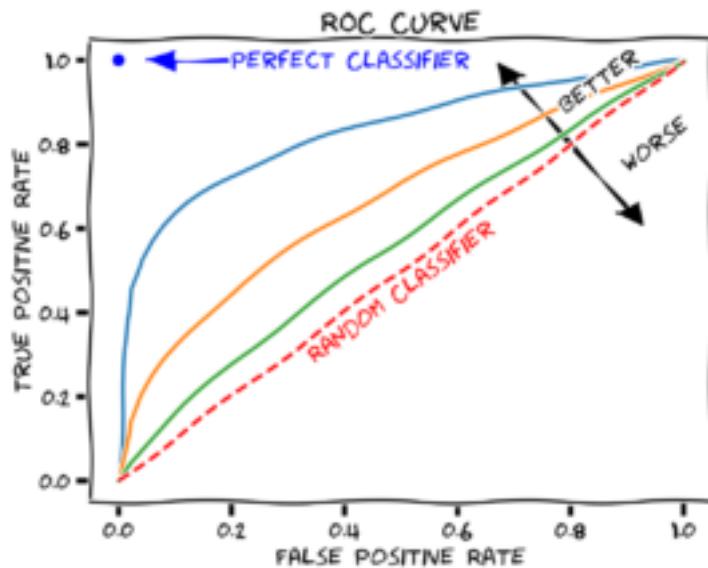
$$\text{Sensibilité} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$$\text{Spécificité} = \frac{\text{TN}}{\text{TN} + \text{FP}}$$



Ces deux indicateurs sont complémentaires, un bon modèle devra être sensible et spécifique (i.e. minimiser les FN et FP).

Pour comparer les modèles, on pourra représenter la sensibilité en fonction de la spécificité en faisant varier le seuil de 0 à 1 : c'est la courbe ROC. Dès lors, on peut entraîner plusieurs modèles (plusieurs classifiers dans notre cas) et les comparer à l'aide de cet outil.



Visuellement, on a un outil qui permet de comparer différents modèles. Pour pouvoir comparer numériquement ces mêmes modèles, on peut calculer l'aire sous la courbe appelée AUC (Area under the curve). L'AUC est comprise entre 0.5 et 1. Dans notre projet, nous avions comme idée de développer différents réseaux de neurones et d'utiliser l'AUC pour les comparer mais n'avons pas eu le temps d'explorer cette partie. L'implémentation peut s'effectuer à partir de keras (tensorflow) dont la documentation est disponible ici

4.3 Estimation spatiale

Les mesures sur des sites d'observation permettent de connaître des informations sur ces positions, mais aucunement sur l'ensemble de l'espace, de façon continue. Pour résoudre ce problème, afin d'estimer par exemple l'intensité sonore sur l'ensemble de l'espace et de proposer une meilleure visualisation, nous avons décidé de réaliser un algorithme de krigage. Cette procédure que nous allons présenter est plus détaillée dans le rapport d'avancement. Cette méthode nous semble favorable à un algorithme d'interpolation de déterminisme local, par exemple à un diagramme de Vornoï, par des méthodes barycentriques ou des splines qui ne fournissent aucune justification. Dans le sens où, une fois un résultat obtenu par ces algorithmes, il n'y a aucune autre information qui puisse nous indiquer si l'estimation réalisée est correcte, si elle s'approche du phénomène physique étudié. Dans le cas du krigage, cette méthode statistique nous fournira la variance pour chaque estimation, ce qui nous permettra de contrôler davantage cette extension du discret au continu.

Le krigage est une méthode de géostatistique s'appuyant sur des sites d'observations, des mesures dont on connaît la position spatiale et que l'on notera $z(x_i)$. Ces valeurs observées sont considérées comme des tirages d'une fonction aléatoire que l'on note Z . Rappelons qu'une fonction aléatoire est un processus stochastique défini sur un espace probabilisé, comme une loi aléatoire, et sur un espace D , qui sera dans notre contexte l'espace d'étude, un espace spatial sur \mathbb{R}^2 . Cette fonction aléatoire est à valeur dans \mathbb{R} , dans notre étude il s'agit de l'intensité sonore en décibel. À partir de ces sites, on peut estimer les valeurs que prendrait la fonction aléatoire Z en x , avec x différent de x_i . La méthode



d'interpolation locale consiste en deux grandes étapes dont nous rappelons dans la suite les grandes lignes. [3] [4] [10]

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].

$\gamma(h) = \frac{1}{2} \text{Var}(Z(s) - Z(s + h))$ (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 krigage. Dans le cas d'un fonction aléatoire supposée strictement intrinsèque, le krigage 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,

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

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

Le krigage ordinaire possède plusieurs propriétés, nous en citerons deux particulièrement intéressantes. Une estimation réalisée sur un site où a été réalisé 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 tendances 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.

Comme nous l'avons vu, les méthodes de géostatistique transitive, soit déterministes, ont l'avantage de pas avoir recours à des hypothèses dont il est difficile de vérifier leur respect. Cependant, les estimations local déterministes souffrent de leur manque d'information. Elle ne fournissent pas de moyen pour prendre du recul sur les résultats, en l'absence de variance par exemple. Les résultats donnés sont à prendre ou à laisser si on ne dispose pas de d'autres moyens pour évaluer la fiabilité de ces estimations. En revanche, dans le cas d'une estimation globale, la géostatistique transitive est bien plus judicieuse. Il s'agit de caractériser la variable régionalisée par une valeur unique et représentative du champ entier. Le calcul dont nous souhaitons faire est l'abondance. Cet indicateur prend un sens seulement s'il est calculé pour une zone d'étude cohérente, restreinte. Par exemple, il n'y aurait pas de sens à calculer l'abondance de l'intensité sonore sur l'ensemble d'un quartier où certaine zone sont



proche d'un axe routier bruyant, et à l'opposé, le calme règne. Notre zone d'étude doit donc être une zone où on peut considérer que l'intensité sonore est peu variable, ce qui suggère une aire assez faible et une portion libre d'obstacle, comme des bâtiments ou des murs.

L'abondance nous permettra d'avoir en supplément de la cartographie, issue d'estimations locales, un indicateur suggérant si la zone d'étude est fortement perturbée par la pollution sonore ou non. Bien entendu, cet indicateur n'a de sens que s'il est calculé en plusieurs zones, puisque l'abondance d'une intensité sonore n'a pas de sens physique. Dans cette étude, nous le calculerons que dans une zone définie dans la suite par manque de temps. Il conviendra par la suite de calculer l'abondance en d'autre lieu afin d'indiquer si une zone est bruyante ou non. [10]

D'abord, l'abondance q est définie par

$$\int_D z(x) dx$$

où z est l'intensité sonore maximale sur le site x appartenant au domaine D . Seulement, nous ne connaissons z que de façon discrète, sur un nombre de sites discrets. La définition de la grille d'implantation des capteurs sur la zone d'étude va modifier le calcul de l'abondance.

Cette abondance est inaccessible puisque nous ne connaissons des mesures que sur des sites d'observations. Nous devons estimer l'abondance par un estimateur statistique. La définition de la grille d'implantation des capteurs sur la zone d'étude va modifier les caractéristiques de l'estimateur de l'abondance. Nous donnerons dans la suite l'expression des estimateurs de l'abondance en fonction de la grille composé des sites d'observations, leurs biais et leurs variances. L'objectif est de choisir la forme de la grille des sites d'observation que nous utiliserons dans la suite en minimisant la variance de l'estimateur de l'abondance. Cette grille sera par exemple régulière, aléatoire stratifiée, aléatoire uniforme... En d'autres termes, nous ne détaillerons pas la position des sites d'observations dès maintenant, mais seulement la grille, l'emplacement de ces sites les uns par rapport à l'autres. Une fois la forme de la grille définie, les parties suivantes répondront à d'autres questions sur le choix de l'emplacement des sites d'observations, dont la zones d'étude dans Brest, l'écart entre les sites, leurs nombre...

On suppose dans un premier temps que la grille est régulière, soit que les observations sont réparties régulièrement sur la zone d'étude, supposé elle-même rectangulaire par simplification (voir figure 13). D'un site observé à un autre, il y a une distance a selon la longitude et b selon la latitude. En prenant arbitrairement une origine en un site x_0 , les autres sites sont atteints par les indices suivants.

L'estimateur de l'abondance vaut alors $q^*(x_0) = |a||b| \sum_{\eta, \nu \in \mathbb{Z}} z(x_0 + \eta a + \nu b)$

Cet estimateur est simple à calculer et intuitif puisque à chaque point on affecte un poids égal. Aucun site n'est surestimé dans l'abondance par rapport à un autre. Dans le cas où la grille serait randomisée, c'est-à-dire que l'origine X_0 serait choisie à partir d'un tirage d'une loi aléatoire, l'expression de l'estimateur serait semblable. Son biais serait nul, et sa variance ferait intervenir le covariogramme $g(h)$. La variance est donc d'autant plus faible que a et b sont petits, ce qui revient à augmenter le nombre de site d'observation pour une même zone d'étude, et que la fonction g est régulière. Le covariogramme g étant un choix important comme on l'a vu auparavant, l'un des critères qui peut déterminer ce choix entre deux fonctions est la régularité de celles-ci. A noter que l'on reste dans cadre de la géostatistique transitive. Ce n'est pas la variable régionalisé Z qui est aléatoire, mais la grille d'échantillonnage. On peut également choisir une grille irrégulière formée par les sites d'observations. Par exemple, un échantillonnage aléatoire stratifié ou bien aléatoire uniforme. Dans ces deux cas, le biais de l'estimateur est toujours nul, mais l'expression de la variance est modifiée. Cette variance dépend dans tous les cas du covariogramme. On pourra lors du choix des sites d'implantation prendre



en compte ce critère, afin de minimiser la variance. Pour une grille régulière, la variance est égale à $\text{Var}(q^*(x_0) - q) = |\mathbf{a}||\mathbf{b}| \sum_{\eta,\nu \in \mathbb{Z}} g(\eta\mathbf{a} + \nu\mathbf{b}) - \int_D g(h)dh$

Nous donnons les résultats de l'estimateur, du biais et de la variance pour les grilles aléatoire stratifiée et aléatoire uniforme. Une grille aléatoire stratifiée consiste à découper l'espace d'étude D en autant de bloc que de sites d'observation et de même taille. Puis de réaliser un tirage aléatoire d'une loi uniforme dans chaque bloc pour déterminer les emplacements des sites d'observations. Une grille aléatoire consiste à réaliser autant de tirage d'une loi aléatoire uniforme sur l'espace D.

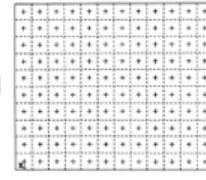
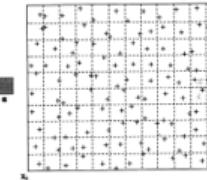
Grille régulière	Grille aléatoire stratifiée	Grille aléatoire uniforme
		
$q^*(\mathbf{X}_0) = \mathbf{a} \mathbf{b} \sum_{\eta,\nu \in \mathbb{Z}} z(\mathbf{X}_0 + \eta\mathbf{a} + \nu\mathbf{b})$	$q^*(\mathbf{X}_0, \{\mathbf{x}_{\eta,\nu}\}) = \mathbf{a} \mathbf{b} \sum_{\eta,\nu} z(\mathbf{x}_{\eta,\nu})$	$q^*(\mathbf{x}_1, \dots, \mathbf{x}_n) = \frac{1}{\theta} \sum_{a=1}^n z(\mathbf{x}_a)$. Où $\theta = \frac{n}{ D }$
Biais nul	Biais nul	Biais nul
$\text{var}[q^*(\mathbf{X}_0) - q] =$ $\underline{ \mathbf{a} \mathbf{b} \sum_{\kappa,\ell \in \mathbb{Z}} g(\kappa\mathbf{a} + \ell\mathbf{b}) - \int g(\mathbf{h})d\mathbf{h}}$	$\text{var}[q^*(\mathbf{X}_0, \{\mathbf{x}_{\eta,\nu}\}) - q] = \mathbf{a} \mathbf{b} (g(\mathbf{0}) - \bar{g}_{[\mathbf{a},\mathbf{b}]})$ Où \bar{g} valeur moyenne de g $\bar{g}_{[\mathbf{a},\mathbf{b}]} = \frac{1}{ \mathbf{a} ^2 \mathbf{b} ^2} \int_{[\mathbf{a},\mathbf{b}]} \int_{[\mathbf{a},\mathbf{b}]} g(\mathbf{x} - \mathbf{y}) d\mathbf{x} d\mathbf{y}$.	$\text{var}[q^*(\mathbf{x}_1, \dots, \mathbf{x}_n)] =$ $= \frac{1}{\theta} [g(\mathbf{0}) - \frac{1}{ D } \int g(\mathbf{h})d\mathbf{h}]$

FIGURE 13 – Table des estimateurs de l'abondance en fonction de la grille des emplacements des sites d'observations.

Pour chacun de ces estimateurs, on remarque qu'ils n'ont pas de biais, ce qui ne sera pas un critère pour décider quelle grille choisir. Leur variance dépend du covariogramme transitive g. En particulier, plus g est régulière, plus les sites d'observations sont corrélés, plus la variance sera faible. Dans notre étude, nous nous attendons à un covariogramme transitif régulier puisque l'atténuation des ondes sonores dans l'air est faible sur la distance dont nous envisageons, de quelques dizaines de mètre. De plus, pour une grille aléatoire uniforme, la variance dépend du domaine D, le choix des dimensions de l'aire d'étude sera donc important. Il est difficile de dire à priori quelle estimatrice offre la variance la plus faible. Par soucis de simplicité, nous poursuivront l'étude avec une grille régulière, qui consiste à calculer une seule intégrale et sans la dépendance de l'aire de D. Nous détaillerons la position des emplacements des sites d'observation dans la suite.



5 Traitement des données audios par deep learning

Nous avons rencontré plusieurs difficultés pour recueillir les données prévues. Premièrement, nous voulions les récupérer dans plusieurs endroits dans Brest, mais le risque de vol était présent, ainsi que le risque d'avoir enregistrements pollués par de nombreux sons non classifiés (hélicoptère, klaxon, sirène).

Nous avons ainsi préféré cartographier la zone vie de l'école. Les sons sont connus et assez réguliers (voiture, voix, musique, vélo, aboiement, oiseux). De plus, nous étions ainsi proches de l'audiomoth.

Mais nous n'avons récupéré l'audiomoth qu'assez tard, et nous avons passé un certain temps à réfléchir à la manière de le disposer pour qu'il ne prenne pas l'eau et que le son ne soit pas altéré. Malgré cela et quelques tentatives, l'audiomoth et sa pochette étaient souvent envolés par le vent à plusieurs mètres du lieu d'accroche, et remplis d'eau. Les données ne pouvaient alors pas être lues.

Nous avons alors décidé de continuer à travailler sur des données sonores recueillies en ligne, comme nous l'avions fait depuis lors, pour créer le meilleur réseau de neurone. Cela n'empêchait pas la pratique de la cartographie sur des données générées artificiellement, mais que nous aurions probablement pu générer nous-mêmes avec plus de temps.

5.1 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.

Comme présenté précédemment, on a alors écrit un premier programme permettant d'écouter par tranches de 10 secondes des sons de tailles quelconques, de les labelliser et de couper l'échantillon, et de créer le spectrogramme associé (cf prochaine sous-section). Également, le programme enregistre dans un fichier texte "data" le path du spectrogramme ainsi que les labels associés, Figure 14.

```
C:/Users/Utilisateur/Documents/ENSTA/2A/UE 3.4/Projet système/Machine_learning/Donnees_label.wav/spec/recording1.png,"pie, chien"
C:/Users/Utilisateur/Documents/ENSTA/2A/UE 3.4/Projet système/Machine_learning/Donnees_label.wav/spec/recording2_r1.png,pie
C:/Users/Utilisateur/Documents/ENSTA/2A/UE 3.4/Projet système/Machine_learning/Donnees_label.wav/spec/recording3_r1.png,"pie, chien, car"
C:/Users/Utilisateur/Documents/ENSTA/2A/UE 3.4/Projet système/Machine_learning/Donnees_label.wav/spec/recording4_r1.png,"car, pie"
C:/Users/Utilisateur/Documents/ENSTA/2A/UE 3.4/Projet système/Machine_learning/Donnees_label.wav/spec/recording5_r1.png,"car, pie, moto"
C:/Users/Utilisateur/Documents/ENSTA/2A/UE 3.4/Projet système/Machine_learning/Donnees_label.wav/spec/recording6_r1.png,pie
C:/Users/Utilisateur/Documents/ENSTA/2A/UE 3.4/Projet système/Machine_learning/Donnees_label.wav/spec/recording29_r1.png,pie
C:/Users/Utilisateur/Documents/ENSTA/2A/UE 3.4/Projet système/Machine_learning/Donnees_label.wav/spec/recording30_r1.png,pie
C:/Users/Utilisateur/Documents/ENSTA/2A/UE 3.4/Projet système/Machine_learning/Donnees_label.wav/spec/recording9_r1.png,"pie, car"
C:/Users/Utilisateur/Documents/ENSTA/2A/UE 3.4/Projet système/Machine_learning/Donnees_label.wav/spec/recording10_r1.png,"pie, car"
C:/Users/Utilisateur/Documents/ENSTA/2A/UE 3.4/Projet système/Machine_learning/Donnees_label.wav/spec/recording11_r1.png,pie
C:/Users/Utilisateur/Documents/ENSTA/2A/UE 3.4/Projet système/Machine_learning/Donnees_label.wav/spec/recording12_r1.png,pie
C:/Users/Utilisateur/Documents/ENSTA/2A/UE 3.4/Projet système/Machine_learning/Donnees_label.wav/spec/recording13_r1.png,"pie, chien"
```

FIGURE 14 – Fichier «data» créé après labellisation des échantillons, associant à chaque spectrogramme ses labels.



5.1.1 Choix du type de spectrogramme (explication log etc, différents choix)

Pour pouvoir être exploités par les réseaux de neurones, les fichiers .wav doivent être convertis en .png. Les fichiers png sont les spectrogrammes de l'échantillon : ils représentent l'intensité sonore en fonction de la fréquence et du temps, figure 15.

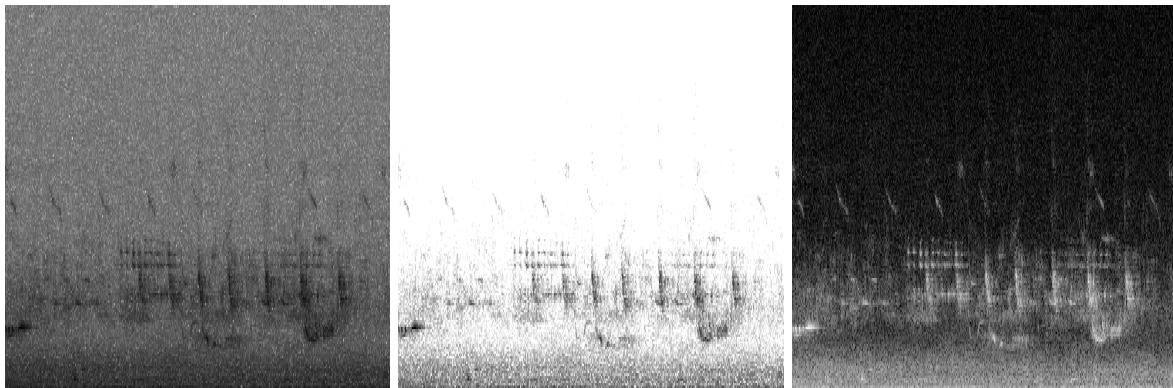


FIGURE 15 – Trois spectrogrammes d'un même échantillon, affichant l'intensité en fonction de la fréquence et du temps.

Ces trois spectrogrammes sont issus d'un même son. Ils sont différents en fonction des traitements appliqués aux entiers (de 0 à 255) représentants le niveau de gris de chaque pixel. Différents fonctions/ seuillages ont été testés, jusqu'à obtenir l'image qui semblait faire ressortir au mieux les pics d'intensité sonore, et masquer au mieux le bruit. Cette image est la troisième.

Pour l'obtenir, on a appelé la fonction `signal.spectrogram` avec en paramètre la fréquence d'échantillonage du son, le tableau des intensités sonores (en 1D si audio et en 2D si stéréo), et en précisant qu'on ne souhaite pas de zero-padding. On obtient alors un tableau des fréquences, un tableau des temps, et pour la troisième dimension (image), une matrice des intensités sonores, sur le maillage des temps X fréquences.

N'importe quel spectrogramme est issu de cette première opération. Ensuite, nous le spectrogramme trois, les fréquences inférieures à 100 Hz et celles supérieures à 15 kHz sont supprimées (inaudibles). On recueille l'opposé du log des valeurs des intensités, que l'on condense et centre pour qu'elle appartiennent à [0;255] et puisse représenter des pixels. Enfin tous les pixels de valeurs supérieures à 120 sont seuillés à 120. On les considèrent comme "absents", et cela permet d'étaler les pixels de grande taille inférieure sur l'échelle des entiers lors du `plt.colormesh()`. On laisse ainsi apparaître plus de nuances pour les valeurs d'intensité moyenne élevée à élevée, tout en levant le bruit faible.

Ce sont les valeurs élevées qui sont seuillées, mais elles correspondent aux intensités faibles (-log au début). La raison du passage à l'opposé est que le `colormesh` affecte du noir aux grandes valeurs et blancs aux petites (`cmap = "binary"`). Or, sur l'image ensuite obtenue, l'`array` représentant l'image contient des valeurs élevées pour les blancs, et faibles pour les noirs, comme toutes les images. Ce sont ainsi bien les pixels blancs qui nous intéressent, et qui sont censés connoter une présence sonore, car ils sont reconnaissables par le réseau de neurone, puisque leur valeur est élevée et non proche de 0.

C'est aussi pour cela que la troisième image semble la plus pertinente, car les zones remarquables, car la zone blanche correspond à la signature du son, et non au bruit ou aux intensités nulles.



5.1.2 Augmentation de données

La labellisation est très longue, il faut écouter au moins une fois la séquence de 10 secondes pour pouvoir affecter des labels à cette séquences. Ainsi, au début du projet, un son de dix secondes donnait naissance à un spectrogramme, un échantillon dans le dataset.

Mais à cette vitesse, il faudrait passer énormément de temps à labelliser pour obtenir un dataset suffisamment grand. Nous avons alors découvert et mis en oeuvre l'augmentation de données. Celle-ci repose sur un principe simple : on crée artificiellement des variations sur l'image de chaque spectrogramme, de façon à obtenir de nouvelles images, à la fois proches de l'originale et donc cohérentes avec la réalité sonore, à la fois différente de celle initiale et permettant à l'algorithme d'apprendre sur un scénario sonore différent.

Pour notre projet, nous avons réussi à obtenir 9 images à partir d'un seul spectrogramme issu de la labellisation. Celle-ci allait donc "9 fois plus vite", ou permettant d'obtenir 9 fois plus d'échantillons pour un temps de labellisation égal. Les deux techniques mises en œuvres sont la translation temporelle et le moyennage de certaines zones fréquentielles et temporelles.

Translation temporelle

Pour un enregistrement d'un certain nombre de secondes, on coupe celui-ci toutes les 10 secondes et on labellise et génère le spectrogramme. On peut facilement multiplier par 3 le nombre d'image en générant le spectrogramme d'un son de dix secondes, mais translaté de 1.25 secondes vers les temps antérieurs et postérieurs, figure 16. On ne translate pas de plus de temps car on utilise les labels générés lors de l'écoute du son initial, il faut donc que les classes présentes dans le son décalé restent les mêmes. On ne décale pas de moins car sinon les spectrogrammes seraient trop proches, et cette augmentation ne provoquerait que l'apparition de biais pour l'algorithme, faussant ainsi tout l'apprentissage.

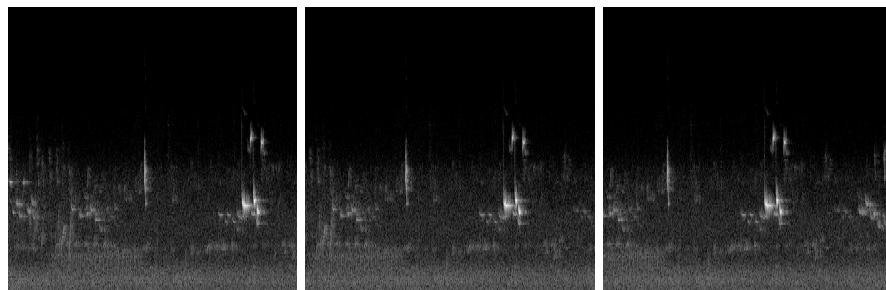


FIGURE 16 – Translation temporelle bilatérale de 1.25 secondes, respectivement antérieurement à gauche, postérieurement à droite, et sans translation au centre



Masquage de zones fréquentielles et temporelles

Pour chacun des trois sons correspondants aux décalages temporels, on peut effectuer des opérations de moyennage sur l'image. On sélectionne aléatoirement un temps, et pour une bande de largeur fixe centrée en ce temps, on fixe la valeur des pixels à la moyenne de tous les pixels sur l'image. On procède de même pour les fréquences, figure 17.

On obtient alors une image identique à l'originale, à la différence que des zones fréquentielles et temporelles sont comme "masquées", car moyennées, elle ne représentent plus de valeurs significatives pour l'algorithme.

L'algorithme doit alors considérer ce qu'il lui reste de visible sur l'image pour parvenir à la bonne estimation. Cela contribue à rendre l'algorithme plus performant pour généraliser, en considérant maints critères provoquant la présence d'une classe.

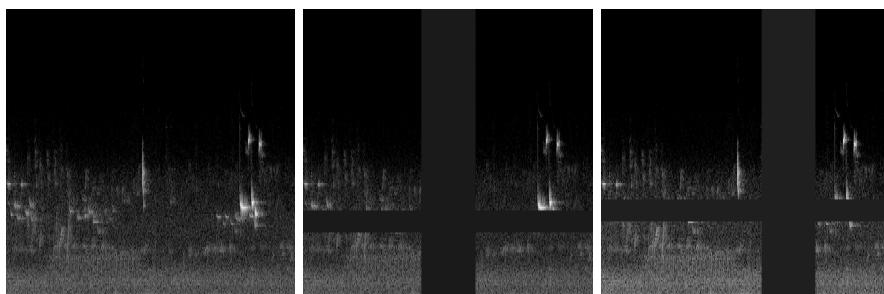


FIGURE 17 – Moyennage de zones fréquentielles et temporelles (figures 2 et 3), à partir d'un spectrogramme original.

On peut noter que l'on n'a pas effectué par exemple de symétries ni de rotations, car celles-ci ne sont pas du tout adaptées aux sons. Un spectrogramme "la tête en bas" n'a plus aucune réalité physique, et ne correspond à coûts sûrs pas aux signatures des classes présentes dans le son associé.

5.1.3 Dataset

Après labellisation de nombreux son recueillis en ligne et augmentation d'apprentissage, le dataset obtenu contient 2100 échantillon, et regroupe 7 classes : car (voiture), moto, pie, chien, voix, vélo, mouette. Les données ne sont malheureusement pas toutes présentes à fréquence égale. Dès lors, la métrique d'accuracy (cf 4.2.2) ne sera pas pertinente pour évaluer les performances de nos modèles. La somme de toutes les occurrences est supérieure à 2100 puisque plusieurs labels peuvent être présents pour un même son.

Classe	car	moto	pie	chien	voix	vélo	mouette
Occurrence	720	42	607	67	1248	900	246

Les figures 18 et 19 présentent les spectrogrammes des sons de chacunes des classes. Nous avons dissocié les oiseaux en général "pie", et la mouette dans nos classe. En effet, les spectrogramme sont assez éloignés, cela aurait pu induire en erreur le modèle si ces sons étaient mélangés. La mouette a un cri de fréquence plus aigue et semble émettre chaque note plus longtemps que la pie. la dissociation de ces classes était une des interrogations que nous avions lors du rendu du précédent rapport

La voiture et la moto ont des fréquences d'émissions plus basses et sans interruptions. Effectivement, la où l'oiseau alterne un son et un silence dans son cris, les moteurs "ronronnent" en continue. La bande de fréquence plus aigue pour la moto, en début d'échantillon, correspond à l'accélération. La voix recouvre toutes les fréquences de l'audible, comme le vélo, mais de façon beaucoup plus irrégulière que le bruit d'un dérailleur de vélo. Quelques rares éclats de voix montent dans les aigus intenses. Enfin, les différents aboiements du chien sont assez espacés et concentrés en une petite gamme de fréquence (rond rouges).

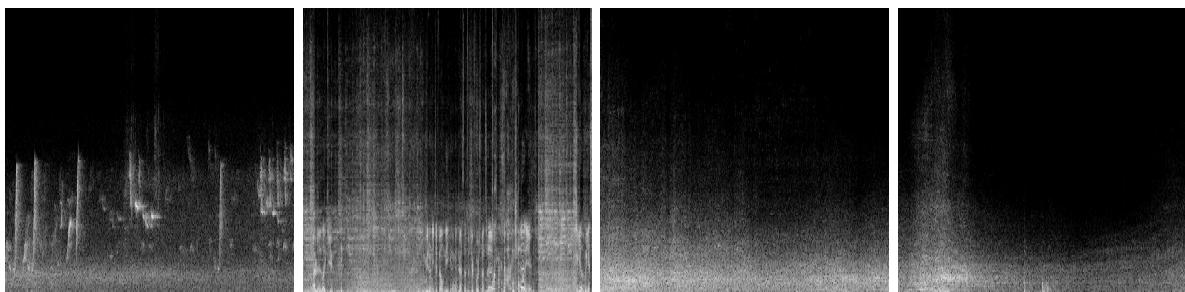


FIGURE 18 – Spectrogrammes de la pie, de la mouette, de la voiture et de la moto.

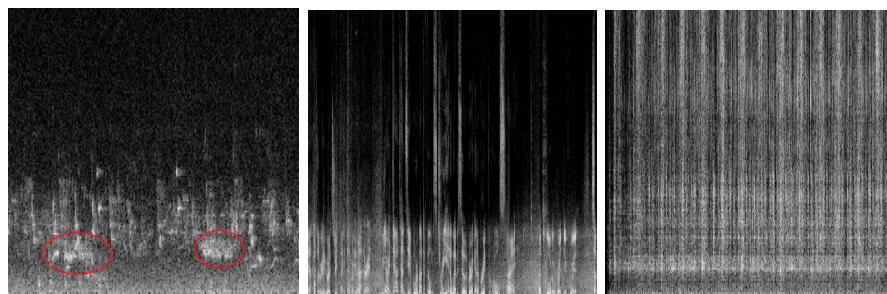


FIGURE 19 – Spectrogrammes du chien, de la voix et du vélo

5.2 Exploitation des données par méthode de Deep Learning

Nous sommes en mesure de stocker, écouter, convertir en spectrogramme, labelliser les données sonores, et regrouper les paths, labels, classes dans les fichiers listClass et data, et dans le répertoire spec (pour les spectrogrammes). Nous allons maintenant pouvoir insérer ces données dans nos réseaux de neurones, pour en tirer des modèles d'apprentissages qui permettront par la suite de réaliser des prédictions sur des données sonores.

5.2.1 Premiers réseaux de neurones utilisés

Notre premier réseau utilise une succession de couches de Convolution et de Pooling, comme présenté figure 21. Enfin de réseau, on aplatis les données (flatten) pour les rendre exploitables par la dernière couche de neurones. Cette dernière couche de neurone est de taille 7, et chaque poids correspond à la probabilité calculée de présence d'une classe. On ajoute une couche de 32 neurones, suivi d'un DropOut, pour limiter l'overfitting (phénomène de sur-apprentissage du modèle sur les données d'apprentissage, conduisant à une non-capacité de généraliser le modèles sur des données autres, et à des mauvaises prédictions). Au total, on entraîne 49 543 paramètres au cours de l'apprentissage.

Nous avons eu beaucoup de difficultés à lancer l'apprentissage, surtout sur nos ordinateurs personnels. Nous avons utilisé le matériel de l'école (Curry, VPN), nous permettant tout de même de générer et stocker nos modèles.

Model: "sequential"			Model: "sequential"		
Layer (type)	Output Shape	Param #	Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 996, 996, 32)	832	conv2d (Conv2D)	(None, 996, 996, 32)	832
max_pooling2d (MaxPooling2D)	(None, 249, 249, 32)	0	max_pooling2d (MaxPooling2D)	(None, 249, 249, 32)	0
conv2d_1 (Conv2D)	(None, 247, 247, 16)	4624	conv2d_1 (Conv2D)	(None, 247, 247, 16)	4624
max_pooling2d_1 (MaxPooling2D)	(None, 61, 61, 16)	0	max_pooling2d_1 (MaxPooling2D)	(None, 61, 61, 16)	0
conv2d_2 (Conv2D)	(None, 59, 59, 16)	2320	conv2d_2 (Conv2D)	(None, 59, 59, 16)	2320
max_pooling2d_2 (MaxPooling2D)	(None, 14, 14, 16)	0	max_pooling2d_2 (MaxPooling2D)	(None, 14, 14, 16)	0
conv2d_3 (Conv2D)	(None, 12, 12, 32)	4640	conv2d_3 (Conv2D)	(None, 12, 12, 32)	4640
max_pooling2d_3 (MaxPooling2D)	(None, 6, 6, 32)	0	max_pooling2d_3 (MaxPooling2D)	(None, 6, 6, 32)	0
flatten (Flatten)	(None, 1152)	0	flatten (Flatten)	(None, 1152)	0
dense (Dense)	(None, 32)	36896	dropout (Dropout)	(None, 1152)	0
dropout (Dropout)	(None, 32)	0	dense (Dense)	(None, 512)	590336
dense_1 (Dense)	(None, 7)	231	dense_1 (Dense)	(None, 7)	3591
<hr/>			<hr/>		
Total params: 49,543			Total params: 606,343		
Trainable params: 49,543			Trainable params: 606,343		
Non-trainable params: 0			Non-trainable params: 0		

FIGURE 20 – Réseaux de neurones convolutifs mis en place.

Un second réseau de neurones a été testé. Il ressemble au précédent, avec en avant dernière couche un réseau dense de 512 neurones, au lieu de 32 précédemment. On entraîne alors beaucoup plus de paramètres, 606 343.



De nombreuses modulations on été tentées sur le réseau, mais il est très long de run un tel code, et d'autant plus long que nous commençons à avoir beaucoup de données. Nous en avons ici retenues 3, dont nous détaillerons les résultats par la suite.

5.2.2 Réseau de neurones pré-entraînés

Devant les résultats peut concluants, nous avons utilisé un réseau de neurones pré-entraînés, vgg-16. Sa configuration se présente comme suit :

Layer (type)	Output Shape	Param #
<hr/>		
input_1 (InputLayer)	[None, 1000, 1000, 3]	0
block1_conv1 (Conv2D)	(None, 1000, 1000, 64)	1792
block1_conv2 (Conv2D)	(None, 1000, 1000, 64)	36928
block1_pool (MaxPooling2D)	(None, 500, 500, 64)	0
block2_conv1 (Conv2D)	(None, 500, 500, 128)	73856
block2_conv2 (Conv2D)	(None, 500, 500, 128)	147584
block2_pool (MaxPooling2D)	(None, 250, 250, 128)	0
block3_conv1 (Conv2D)	(None, 250, 250, 256)	295168
block3_conv2 (Conv2D)	(None, 250, 250, 256)	590080
block3_conv3 (Conv2D)	(None, 250, 250, 256)	590080
block3_pool (MaxPooling2D)	(None, 125, 125, 256)	0
Model: "sequential"		
block4_conv1 (Conv2D)	(None, 125, 125, 512)	1180160
block4_conv2 (Conv2D)	(None, 125, 125, 512)	2359808
block4_conv3 (Conv2D)	(None, 125, 125, 512)	2359808
block4_pool (MaxPooling2D)	(None, 62, 62, 512)	0
block5_conv1 (Conv2D)	(None, 62, 62, 512)	2359808
block5_conv2 (Conv2D)	(None, 62, 62, 512)	2359808
block5_conv3 (Conv2D)	(None, 62, 62, 512)	2359808
block5_pool (MaxPooling2D)	(None, 31, 31, 512)	0
<hr/>		
Total params: 77,695,815 Trainable params: 70,060,551 Non-trainable params: 7,635,264		
<hr/>		
Total params: 14,714,688		

FIGURE 21 – Modèle VGG-16 et Insertion de se réseau dans notre propre réseau

En sortie de ce réseau, nous avons ajouté une couche de 128 neurones. Après avoir figé tous les neurones du réseau vgg-16, sauf le dernier bloc de convolution/MaxPooling, nous obtenons un réseau avec 77 695 815 paramètres, dont 70 860 551 entraînables.

Nous aurions aimé avoir des résultats à exploiter, mais le nombre de neurones est très élevé, même avec Curry, la simulation était prédite à parfois plusieurs jours. Lorsque nous diminuons la taille du dataset d'entraînement, Curry pouvait réaliser le calcul, mais les résultats n'étaient absolument pas concluants. De plus, au delà du temps de calcul, il est arrivé fréquemment que le processus s'arrête pour manque de mémoire du système.

5.2.3 Réalisation des prédictions et passage à la cartographie

Après avoir entraîné le modèle (détails techniques dans le rapport précédent), on peut le sauvegarder. On écrit ensuite un nouveau programme, qui va charger le modèle et déterminer, grâce à celui-ci, les classes présentes dans chaque son (spectrogramme concrètement) d'un ensemble de données fournies à l'algorithme.



Nous avions prévus que les fichiers wav puis png aient des titres significatifs, contenant la localisation (coordonnées géographique, et la date d'enregistrement, éventuellement la rue, l'intensité maximale). Puisque nous n'avons pas enregistré nous-même des données à Brest, ces informations n'existent pas, mais nous pouvons les générer artificiellement pour pouvoir faire le lien avec la partie cartographie. On génère alors un fichier texte, qui regroupe les données utiles à la réalisation de la carte: x,y, t, [booléen classe1, booléen classe2, etc], intensité.

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.

5.3 Résultats

Dans le rapport précédent, nous avons utilisé l'accuracy pour rendre compte des performances de nos modèles. Mais nous avons précisé plus haut qu'au vu de l'inégalité d'occurrence des classes, cela n'était pas pertinent. On peut alors utiliser la Matrice de confusion (présentée dans l'état de l'art).

5.3.1 Analyse de résultats via la Matrice de confusion

Un des indices de performance d'un réseau de neurones est la matrice de confusion. On dira qu'un CNN est performant au sens de la matrice de confusion si cette dernière est similaire à un matrice diagonale. En effet, cela voudra dire que la majorité des échantillons ont bien été prédits.

La figure 22 correspond aux matrices de confusion sur 2095 échantillons testés sur nos deux premiers modèles (cf 6.2.1). Les 2 matrices sont très semblables. Cela s'explique par le fait que les données à estimer sont les mêmes (donc même présence réelle de chaque combinaison de classe dans les 2 matrices). Mais les présences estimées sont aussi très ressemblantes, comme si certaines confusions ou erreurs n'avais pas forcément pour source un réseau non adapté.

On remarque d'emblée que ce n'est pas une diagonale, et on pouvait s'y attendre avec l'analyse de l'accuracy (bien que non pertinentes dans les détails, elle peut être évocatrice dans les grandes lignes). Cependant, la matrice de confusion possède un autre indicateur : elle permet de voir entre quels type d'échantillons l'algorithme se confond.

Dans notre cas, on remarque que les *voitures* sont souvent prédites comme les *pies* ou la *voix*. Ou encore que la *voix* et le *vélo* sont très corrélées ce qui est compréhensible lorsqu'on regarde leurs spectrogrammes qui sont très similaires (voir Figure 19). Par ailleurs, le réseau de neurones n'arrive jamais à prédire les labels *voiture+vélo+voix* et *vélo+voix* ensemble, mais il arrive à les classifier dans les labels simples *voiture* ou *voix*. On peut donc dire que le CNN arrive à isoler des features mais pas à les associer ensemble.

Légende : pie: 0; car: 1; chien: 2; moto: 3; voix: 4; velo: 5; mouette: 6; car+velo+voix: 7; velo+voix: 8; car+voix: 9; car+pie+voix: 10



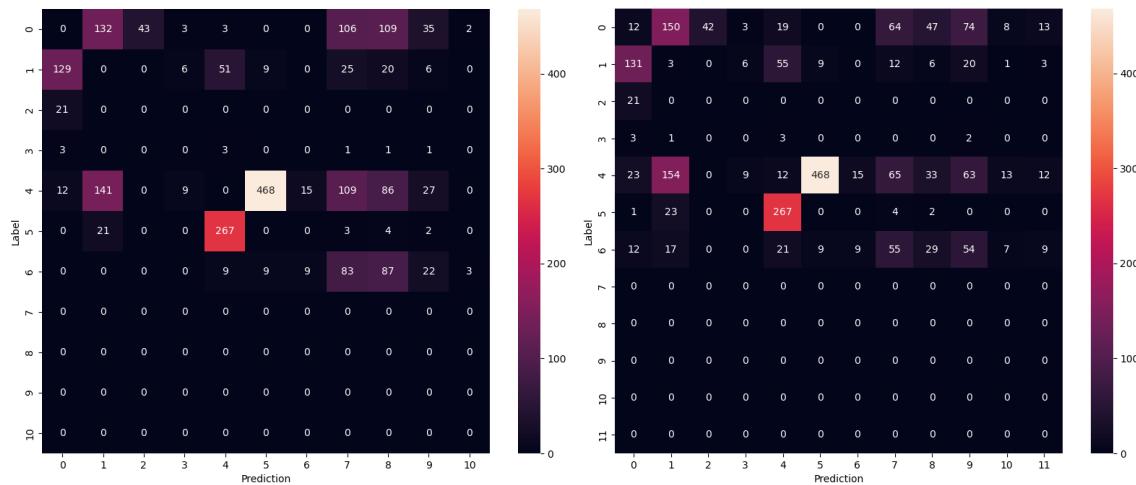


FIGURE 22 – Matrices de confusion

5.3.2 Conclusions sur la partie prédition

Certes, nous ne sommes pas parvenus à trouver un modèle nous permettant de réaliser des prédictions fiables, malgré nos différentes tentatives. Mais nous sommes malgré tout parvenu à combiner les étapes de découpage, labellisation, passage à l'image, traitement de l'image, augmentation de données, insertion dans un modèle, récupération de prédictions, et générations de fichiers textes pour validation des performances (fichier "données pour performances", contenant les labels réels et estimés) et pour cartographie (fichier "donnes pour carte", contenant les labels estimés, de sons inconnus, des positions et temps artificiels, qui seront exploités directement dans la partie cartographie).

Sauf la partie performance algorithmique, les étapes se déroulent sans accroc, et le concept de réseaux de neurones a bien été compris. La récolte de nos propres données, dans des conditions toujours semblables, et la labellisation sur un plus petit nombre de classes (au moins en premier lieu) auraient certainement amélioré significativement les performances de l'algorithme.

6 Visualisation de la pollution sonore urbaine par cartographie

6.1 Les moyens pour transmettre l'information par la cartographie

Le réseaux de neurone présenté précédemment était un moyen pour permettre d'établir une cartographie de l'intensité sonore moyenne sur un site. L'objectif à présent est de permettre à un décideur de l'aider à prendre des mesures pour atténuer la propagation d'ondes sonores perturbantes dans le milieu urbain. La visualisation par cartographie à l'avantage de conduire par l'image à une synthèse des lieux où l'on mesure une intensité sonore élevée par rapport au seuil préconnisé ou au contraire suffisamment faible pour éviter au décideur de lancer des aménagements qui ne seraient pas utiles.

Pour une visualisation plus confortable, nous avons décidé de réaliser des estimations par krigage en s'appuyant sur les mesures effectuées. Tout d'abord, nous devons définir la zone d'étude. Plusieurs points sont à prendre en compte pour que l'information que le client observera et analysera soit fiable. Premièrement, les mesures effectuées par le capteur audiomoth doivent être précises. Il n'y a pas d'erreur importante commise par ces mesures, la précision de l'audiomoth étant de ? dB. Cependant un étalonnage du capteur pour éviter un biais trop important est souhaitable. Toujours sur ces mesures, l'estimation par krigage requiert des contraintes importantes afin que l'estimation ait une variance faible, en dessous de ? db. Les mesures sont réalisées en des sites distincts, formant une grille sur la zone d'étude. D'une part, ces sites de mesures doivent être les plus nombreux possible. Ces l'une des difficultés que nous avons rencontré au cours de ce projet, puisque cette condition nécessite un temps de traitement des données significatif. Pour cette raison en particulier, nous avons décidé de faire des mesures sur une zone restreinte, d'une centaine de mètre carré. Cette restriction du champ étudié nous permettra tout de même de faire à la fois des estimations localisées par krigage et une estimation global pour chaque label. D'une autre part, cette grille doit être suffisamment régulière, pour ne pas délaisser certaines aires de la zone d'étude ou sur-représenter des aires localisées par rapport au reste de la zone d'étude. On peut se poser la question si l'emplacement des sites doit comporter un processus aléatoire comme on l'a vu dans l'estimation de l'abondance plus tôt. Dans notre cas, le phénomène physique, la propagation d'ondes sonores, ne pose pas de problème de structure. L'onde se propage de façon continue dans l'espace. L'atténuation de l'onde ne crée pas non plus de structure discontinue. Nous pouvons donc, pour simplifier le modèle, choisir des sites de mesures de façon régulière.

Ensuite, l'une des limites à l'estimation locale par krigage est qu'elle ne tient pas compte du support physique. En d'autres termes, on fait l'hypothèse que le champ d'étude est continue et homogène. Rappelons que le phénomène étudié est la propagation d'ondes sonores dans l'air. La zone étudiée est suffisamment restreinte pour considérer le milieu homogène, les conditions météorologiques entre deux sites distants d'une centaine de mètre ne vont pas brusquement varier au point que la propagation des ondes sonores sera affectée et discernable par un passant. En revanche, les bâtiments doivent nous contraindre dans le choix des sites d'emplacement du capteur. D'une part, on ne peut pas mettre un capteur à l'intérieur ou derrière un bâtiment, puisque les murs forment un espace matérielle différent que l'air, ce qui invaliderait l'hypothèse de milieu homogène. D'une autre part, on évitera de placer un capteur près d'un bâtiment, au risque de recevoir des échos d'une onde réfléchie par un mur qui pourraient perturber le réseau de neurones notamment. Les sites d'implantation du capteur doivent donc se trouver suffisamment éloignés des bâtiments, dans un champ libre.

Les résultats du réseaux de neurones auront une influence élevée sur la fiabilité de la cartographie. Une mauvaise classification pourrait entraîner des valeurs aberrantes. Par exemple, si un fichier audio provenant du son d'une voiture est reconnu par l'intelligence artificielle comme un oiseau, qui est censé produire une intensité sonore bien plus faible, la moyenne de l'intensité sonore maximale des oiseaux sera biaisée par cette valeur aberrante.

Par ailleurs, l'étude consiste à récupérer des fichiers audios dans l'espace public. Ce qui signifie



enregistrer et traiter des données pouvant provenir de l'espace privé d'un individu. Pour respecter la loi française, nous pouvons agir par deux moyens. D'une part, limiter la zone d'étude dans un espace peu fréquenté ou au moins un espace où l'on ne devrait pas pouvoir distinguer des conversations par exemple. Ensuite, avertir les passagers que le lieu est enregistré par un capteur audio, afin que ces derniers soit conscients que leur conversations peuvent être enregistrées.

Nous nous sommes donc limités à une aire sur le site de l'ENSTA Bretagne. Cette zone se trouve près de la résidence de l'établissement, s'étendant sur environ 1500 m². La grille des implantations des mesures sera régulière et composée d'une dizaine de sites. De plus, cette grille se trouvera au sud de la résidence, nous préciserons la raison de ce choix dans le paragraphe suivant.

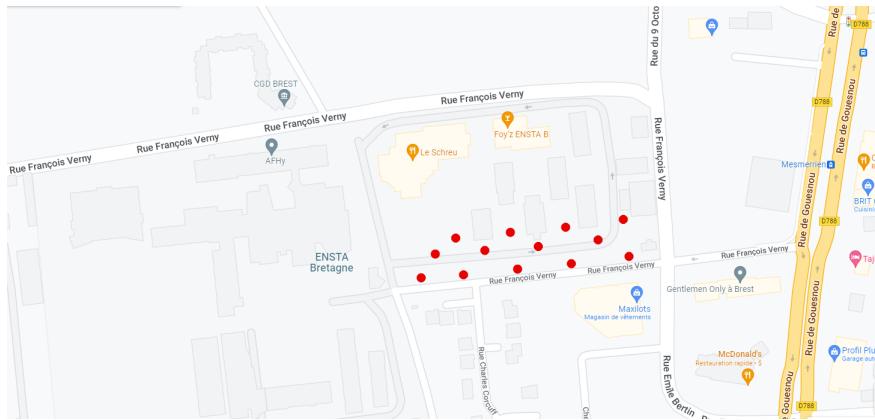


FIGURE 23 – Grille d'implantation du capteur.

6.2 Quelques précisions sur la démarche

Afin d'estimer l'intensité sonore en chaque point de l'espace, nous nous appuyons sur des sites dont l'étude précédente avait pour but de déterminer l'intensité sonore reçue pour chaque label défini. Nous pourrons nous appuyer sur ces sites de référence pour réaliser un krigage. Mais encore faut-il connaître la position de ces sites. Plusieurs moyens permettant de déterminer une position dans l'espace sont décrites dans la suite. La première solution est d'utiliser un récepteur GNSS, en particulier GPS, précisant les coordonnées géographiques du capteur grâce aux signaux envoyés par satellites. Le principe du positionnement par GNSS repose sur la trilateration. Cela consiste à déterminer aux moins trois sphères dont on connaît le rayon. Ce rayon correspond à la distance entre un satellite de la constellation GNSS utilisé et le récepteur. Cette longueur est calculée d'une part à partir du temps entre l'émission du signal et sa réception à l'aide du codage transporté dans le signal, d'une autre part en utilisant le postulat de la célérité de la lumière dans le vide. Cette distance est entachée d'erreurs à cause des hypothèses posées, en considérant l'onde se propageant dans le vide et non dans un milieu matériel par exemple, ou encore la déviation de l'onde dans l'atmosphère... Une correction est donc apportée. L'intersection des trois sphères est la position du récepteur. Quelques remarques sont à prendre en considération afin d'améliorer la précision de la position du récepteur. Plus le nombre de satellite visible par le récepteur est grand, plus la précision sera fine. La précision a donc d'avantage à être meilleure dans une zone dégagée. Par exemple, l'un des obstacles pouvant altérer la précision est la présence de bâtiment près du capteur. La zone dans laquelle nous faisons les mesures en contient.

Cependant, pour éviter cet écueil, nous pouvons remarquer que les satellites de la constellation GPS se trouvent sur des orbites quasiment circulaires, inclinés à 55° par rapport à l'équateur. En observant le ciel, nous ne voyons donc pas de satellite vers le pôle nord puisque nous nous trouvons dans le cadre de cette étude dans l'hémisphère nord. Les mesures doivent donc être réalisées de préférence sur la façade sud des bâtiments. Ils ne cacheront que le ciel en direction du nord, ce qui n'est pas gênant



	GNSS absolue (NMEA Tools)	GNSS absolue (récepteur GPS)
Précision	mètres	mètres
Prix		Dizaines d'euros
Remarques		

	GNSS différentiel (récepteur et station)	Topométrie	Google Maps
Précision	millimètres à décimètres	millimètres à centimètres	dizaines de mètres
Prix	Plus d'une centaine d'euro	Plus d'une centaine d'euro	
Remarques	Manipulation assez compliquée	Manipulation compliquée	

puisque il n'y a pas de satellite. Par ailleurs, la correction est elle aussi plus ou moins précise, elle dépend de différents facteurs, dont l'un est sur la mesure réalisée. Jusqu'à présent, la description ne s'est attaché qu'à la mesure de positionnement absolu, consistant à déterminer des distances entre des satellites et le récepteur GPS. Cette technique fournit une précision de l'ordre du mètre au mieux. Une mesure différentielle est une technique permettant une meilleure précision jusqu'au millimètre. Cette technique est semblable aux mesures absolues, mais en s'appuyant sur une station au sol recevant elle également des signaux provenant de satellites, les corrections apportées sont beaucoup plus significatives.

Les récepteurs GPS réalisant des mesures absolues sont abordables, soit en utilisant directement son téléphone portable qui possède généralement un récepteur GPS, il est possible de recevoir des trames NMEA par l'intermédiaire d'une application comme TrameNMEA, soit en disposant d'un récepteur dédié, dont la précision ne s'améliore que très peu par rapport à un téléphone portable. Son prix est de quelques dizaines d'euros. Les récepteur GPS permettant de réaliser des mesures différentielles sont quant à eux plus précis, en fonction des mesures différentes réalisées et de l'environnement du récepteur, mais son prix est plus important, au moins de quelques centaines d'euros.

Une autre technique est de s'appuyer sur des sites repérables et répertoriés dans des archives. A partir de ces objets identifiables, il est possible de déterminer la position de site visible par le repère avec un mètre ruban, un distancemètre, un tachymètre ou une station totale, sachant que ces instruments sont de plus en plus précis dans l'ordre dont ils sont énoncés. Une station totale peut atteindre des précisions de quelques millimètres. Il existe des points de référence dans la zone d'étude. Les instruments ne sont pas abordables et demande une formation au préalable. Enfin, on peut s'aider de ressources numériques, comme Google Map, afin de déterminer la position de sites sans coût. Seulement, la précision est de quelques dizaines de mètres dans la zone d'étude.

Par ailleurs, au-delà des erreurs commises par le système GPS, une autre erreur peut s'ajouter. La constellation GPS modélise la surface terrestre par une ellipsoïde, une surface dont on connaît mathématiquement dans l'espace euclidien. Cette forme de la Terre est une approximation de la surface terrestre réelle. Il existe une infinité d'ellipsoïde. En fonction du cadre d'étude, on utilise l'une plutôt que les autres pour limiter les erreurs commises. La constellation GPS s'appuie sur l'ellipsoïde WGS84, un système mondial et géocentrique. L'erreur commise en utilisant ce système est de l'ordre du mètre. Ce qui nous a décidé à privilégier l'utilisation du GPS en absolue, par l'intermédiaire de l'application NMEA Tools. Cette méthode est en plus simple d'utilisation, facile à transporter d'un site à un autre. L'erreur de l'ordre du mètre est assez importante mais acceptable puisque les sites de mesures sont suffisamment écartés. Dans les exigences de départ, nous avions fixé avec une grande incertitude cet écart à une quinzaine de mètre. Dans ce cas, deux sites seront donc identifiables. L'erreur commise par le positionnement GPS ne sera pas un problème majeur. [2]

Les coordonnées provenant d'un récepteur GPS sont transmises par des trames NMEA. Plusieurs informations sont transmises, avec plus ou moins de détails. La transmission des données se fait par des trames composées avec des caractères ASCII. Bien qu'utilisées dans de nombreuses applications, dans



le cas d'un récepteur GPS, ces trames commencent toutes par \$GP suivi par trois lettres identifiant la structure des informations communiqués. En ce qui concerne le positionnement, cette information se trouve dans toutes les trames GGA. Par ailleurs, on retrouve dans ces trames la date. Le positionnement sont fournies par l'intermédiaire de la longitude et de la latitude en coordonnées géographiques, une sous famille de coordonnées sphériques. Ces coordonnées peuvent être directement utilisées par le SIG. Dans notre cas, nous pourrons nous contenter d'enregistrer ces coordonnées pour chaque site ou de réaliser une transformation de coordonnées dans une représentation plane de Lambert 93 comme expliqué précédemment. [5]

Afin de se rendre compte de l'erreur possiblement commise par le positionnement en GPS absolue, qui atteint en général l'ordre du mètre, nous avons décidé de réaliser une classe stat héritant de la classe GGA. Cette classe permet d'afficher les ellipses d'incertitude sur les mesures reçues par la récepteur GPS. La zone délimitée par l'ellipse correspond à la probabilité que le site d'observation se situe à 90%, 95%, 99%... dans cette aire. Cette ellipse est obtenue en calculant les vecteurs propres et les valeurs propres de la matrice de covariance, les premiers nous donnent les directions privilégiées, le grand axe et le petit axe de l'ellipse, tandis que les valeurs propres donnent la longueur de ces axes respectivement. On obtient une précision comme annoncé plus tôt de l'ordre du mètre.

Par ailleurs, on observe que les données pourraient être considérées comme de tirage d'une loi aléatoire normale multidimensionnelle, d'ordre 2. Le diagramme quantile-quantile l'indique aussi. Pour confirmer cette hypothèse, nous avons réalisé un test statistique pour vérifier si les positions données peuvent être supposées comme suivant une loi gaussienne dans les deux directions privilégiées des ellipses. Le nombre de données étant suffisamment grand, mais restreint, de l'ordre de 30 mesures pour un site d'observation, nous avons réalisé un test de Shapiro-Wilk, plus robuste que d'autres test de normalité pour des petits échantillons. [9] Ce test consiste à calculer la statistique W, donnée ci-dessous, puis à la comparer à une table. Si la valeur obtenue par W est inférieure à celle renseignée dans les tables, alors l'hypothèse que l'échantillon ait été tiré d'une loi normale est rejeté. Ce test n'a pas été vérifié pour tous les sites d'observations.

$$W = \left(\frac{\sum_{k=1}^n a_i z_i}{\sum_{k=1}^n z_i - \bar{z}} \right)^2$$

où les coefficients a_i sont données dans des tables.

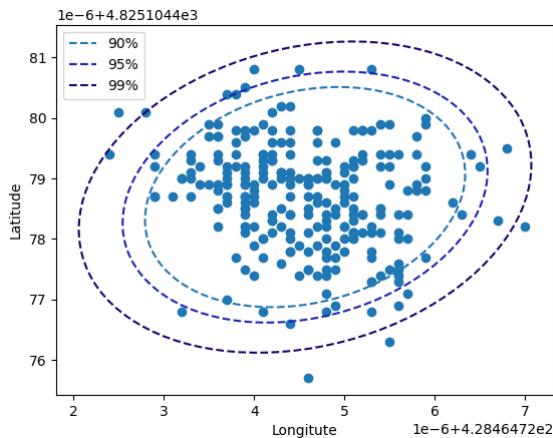


FIGURE 24 – Figure de l'ellipse d'incertitude sur les données GPS.



6.3 Traitement des données

Une fois les mesures de positionnement réalisées et ces fichiers traités par le réseau de neurones, il faut les inclure dans un SIG. Plusieurs SIG sont possibles, du plus basique comme en utilisant Python au plus élaboré comme GRASS. Dans notre étude, qui ne nécessite pas d'outil complexe, nous avons choisi de réaliser un logiciel en Python, afin d'avoir un projet entièrement dans ce langage. Mais aussi en utilisant des fonctions déjà programmé en R, qui nous permettrons d'améliorer nos estimations par krigeage par rapport aux programmes Python. Le langage R a été choisi du fait qu'il est majoritairement utilisé pour des programmes de statistique et notamment de géostatistique.

Le premier choix à faire est le système de référence à prendre. Encore une fois, il existe de nombreuses possibilités. Pour limiter les erreurs, et puisque notre étude se limite à une zone restreinte, nous pouvons choisir une ellipsoïde limitant les erreurs, sur la zone de Brest, d'avantage que l'ellipsoïde WGS84 qui donne une approximation dans l'ensemble du globe, ce qui n'est pas notre objectif. Par exemple, une des ellipsoïdes Lambert93, particulièrement utilisé en France métropolitaine, permet d'avoir une erreur plus faible entre la géoïde terrestre et ladite ellipsoïde. Cette erreur selon la longitude et la latitude peut être quantifié à l'aide du module linéaire, qui est alors plus petit à Brest avec une ellipsoïde Lambert93 que WGS84. Nous avons donc décidé d'utiliser le système RGF93, dont la représentation plane est un Lambert 93 pour minimiser l'erreur de précision. La projection plane associé au Lambert93 est conique et conforme, plus adapté pour les latitudes auxquelles se situe l'étude. L'erreur sera en dessous du mètre, faible par rapport aux erreurs liées au positionnement par GPS. Nous aurons par contre besoin de transformer les coordonnées issues du récepteur GPS, exprimé en degré sur l'ellipsoïde WGS84 à des coordonnées planes Lambert 93. La fonction *WGS84toLambert93* réalise cette conversion dans la classe GGA. Cette méthode utilise des fonctions de conversions du module *pyproj*.

Après avoir été analysé dans la partie deep learning, chaque enregistrement possède sa position. Tous les fichiers audios se trouvant au même site pour une heure similaire sont inclus dans un site. Ce site possède les sources de pollution sonore reconnu, et pour chacune d'entre elle, la moyenne de l'intensité sonore maximal reçu par le capteur audio. C'est à partir de ces données, position, heure, moyenne de l'intensité sonore maximal en ce site pour chaque source, que l'estimation spatial de cette dernière valeur sera faite. En effet, on ne dispose que d'un nombre fini de site où des mesures ont été prises. Afin d'estimer la moyenne de l'intensité sonore maximal reçu sur l'ensemble de la zone d'étude, nous devons réaliser une estimation spatiale.

Plusieurs moyens permettent de réaliser un krigeage et certaines seront présentés et comparer dans la suite. Le premier moyen est de programmer l'algorithme présenté plus tôt, à savoir la détermination du variogramme, afin de prendre en compte correctement les corrélations spatiales, et le krigeage en lui-même. Pour avoir une homogénéisation des outils utilisés, nous avons programmer ces algorithmes sous Python. Ces procédures ont été divisés en deux grandes parties, pour reprendre la division variogramme et krigeage, avec les classes respectifs SemiVariogramme Krigeage. Cette division est d'une part plus facile à comprendre pour l'utilisateur puisqu'il reprendre le schéma des procédures et permet de respecter le principe S de SOLID, une responsabilité pour une classe. De plus, le principe O est respecté en créant une troisième classe Krige. Cette dernière possède les fonctions que partagent le variogramme et le krigeage comme pour le calcul de distance entre des sites. L'héritage permettra facilement, sans modifier les classes déjà présentent, d'étendre le logiciel afin de lui ajouter des options. Par exemple, nous n'avons écrit des programmes seulement pour un krigeage ordinaire, l'extension du logiciel pourra de programmer un krigeage universel ou non stationnaire par exemple.

Un autre moyen est d'utiliser des programmes déjà existants. Il en existe plusieurs sous différents langages. Nous avons décider d'utiliser l'un de ces moyens afin de le comparer avec les résultats obtenus avec notre programme Python. Des bibliothèques sont proposées en Python, mais pour nous avons préféré nous orienter vers d'autres solutions. D'une part, ces programmes permettent d'estimer des valeurs sur



une grille d'étude de façon laborieuse, de fonctionnalité existe mais leur mise en œuvre n'est pas évidente.

D'autre part, ces programmes ne sont pas utilisés couramment dans la communauté des statisticiens. Du fait qu'il est difficile de se fier à ces programmes qui ne sont pas certifiés par les utilisateurs courants, nous avons privilégié l'utilisation d'outils admis par ceux-ci. C'est la raison pour laquelle nous avons décidé de réaliser un programme en R. Cet algorithme est facile à mettre en œuvre et les fonctions sont fréquemment utiliser par une communauté large. Dans ce langage, les données pour un site (positionnement, heure, moyenne de l'intensité sonore maximale) sont stockées dans une variable de type `data.frames`, représentant les données sous forme de tableau, comme un Python, mais ajoutant des informations en classant les données dans le tableau par colonne. Dans notre cas, nous avons deux colonnes pour la position géographique, une pour la date, une pour la moyenne de l'intensité sonore maximale. Il faut dans un premier temps transformer ces données dans un objet *Spatial* qui sera plus adapté pour la suite du programme. Deux bibliothèques permettent de réaliser cette conversion. La première `sp`, largement utilisé et stable, l'autre `sf` plus récente et de plus en plus utilisé. La différence entre les deux vient surtout du fait que la bibliothèque `sf` permet d'ajouter des options. Dans notre étude, nous n'aurons pas besoin de tel option, nous avons donc décider d'utiliser la bibliothèque `sp`. Lorsque les données sont stockées dans un objet *Spatial*, nous pouvons réaliser le variogramme puis le krigeage en procédant à quelques étapes intermédiaires dont la plupart consiste à mettre en forme les données. Les étapes du variogramme et du krigeage proviennent eux de la bibliothèque `gstat`.

Le programme en R nous permet de comparer et apprécier si le logiciel sous Python donne des résultats corrects. Autant les fonctions en R sont déjà implémentées et utilisées régulièrement par une communauté importante, et donc elles ne doivent pas être sujette à des erreurs. Ce n'est pas forcément le cas des programmes sous Python. En plus de comparer les résultats sous R et sous Python, nous avons réalisé des tests unitaires qui sont détaillés dans la fiche *Validation Fonctionnelle Usine*.

6.3.1 Krigeage

Les données présentées dans la suite ont pour but d'illustrer le krigeage. L'histogramme résume ces valeurs d'intensité sonore en décibel auxquelles on ajoute la carte des positions de ces mesures.

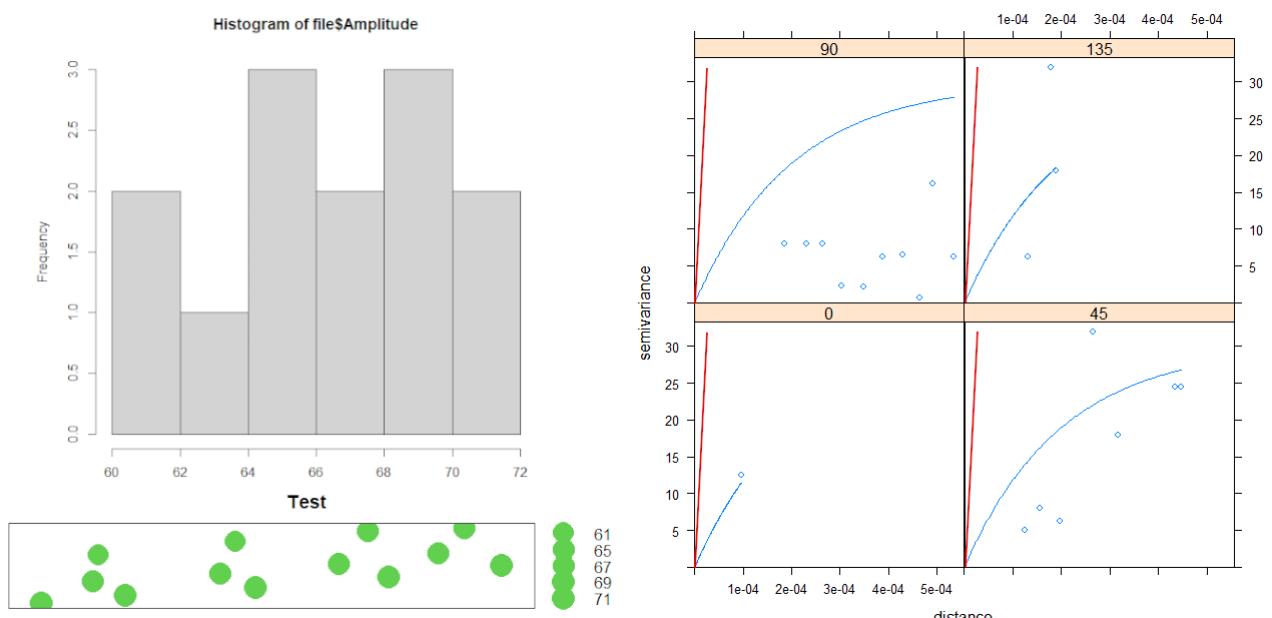


FIGURE 25 – Présentation des données, en haut à gauche, histogramme des données, en bas à gauche, carte de l'emplacement des sites d'observation, à droite, variogramme expérimentale avec le variogramme décidé en rouge, linéaire, et en bleu, sphérique.



D'abord, nous traçons le variogramme, au moyen de son estimateur, sur les treize sites d'observations. A cause du nombre de sites d'observation faible, il est difficile d'affirmer que le variogramme expérimental tracé soit proche du variogramme réelle. C'est un point fragile sur lequel nous devons prendre garde, au moins à repenser si les résultats ne sont pas concevables. Nous supposerons que le variogramme est isotrope aux vues de l'estimateur. Nous testerons deux modèles, l'un linéaire, l'autre sphérique, afin de montrer que le choix du variogramme est crucial. Dans le premier cas, le modèle linéaire n'est pas adapté au variogramme expérimental et donc ne pourra pas être valider. On donnera tout de même cet exemple pour essayer de voir quelles caractéristiques peuvent nous permettre de voir si un krigage est correct ou non. Le second variogramme sera quant à lui valide. Le modèle sphérique est représenté sur le variogramme expérimentale. Les deux variogrammes sont tracés sur le variogramme expérimentale de la figure 26. Pour les deux krigage présentés dans la suite, un pour chaque modèle, on réalisera une centaine d'estimation, plus d'une dizaine selon la longitude et la latitude.

Dans le programme sous Python, nous allons tester le krigage par une fonction linéaire de pente 12 dont la portée est suffisamment grande pour ne pas prendre en compte de palier. Le krigage donne le résultat de la figure.

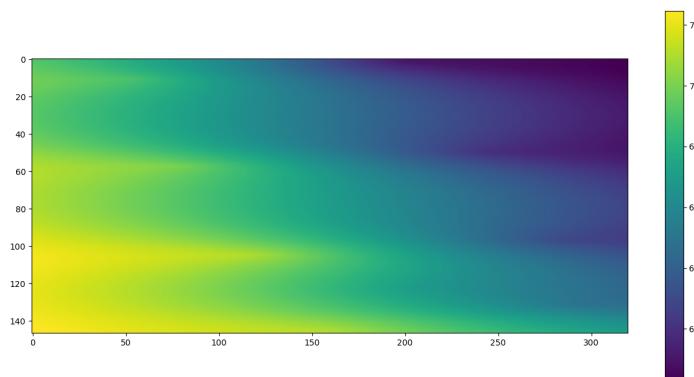


FIGURE 26 – Krigage par un modèle linéaire de pente 12.

Le krigage est correct puisque les sites d'observations sont bien interpolés. Par contre deux remarques remettent en cause le modèle linéaire utilisé. D'une part, l'écart-type que fournie l'algorithme pour chaque estimation est de l'ordre de 70 ! L'écart-type est bien trop important pour pouvoir considérer le krigage comme valide. D'une deuxième part, le krigage ordinaire entraîne une estimation qui à tendance à converger vers la moyenne des valeurs prises aux sites d'observation, comme rappelé dans le rapport d'avancement. On voit bien ce phénomène dans la figure 27, où les traces jaunes en bas à gauche sont des zones proches d'un site d'observation, donc leur valeur est fortement influencé par ce site. Mais dès que l'on en éloigne, en particulier entre ces traces jaunes, les estimations s'approchent de valeur à l'alentour de 66, en bleu. Ces estimations en bleu ne nous conviennent pas puisqu'elles forment des structures dont on ne peut imaginer qu'elles représentent le phénomène physique sous-jacent. On devrait avoir une liberalité au lieu d'observer des valles en bleu entouré de sommet en jaune. Ce phénomène fait apparaître le mauvais choix du variogramme linéaire avec une pente de 12. En effet, comme on peut le voir sur le variogramme expérimental, la fonction linéaire ne passe pas par les points de la figure 25. La pente de la fonction linéaire est beaucoup trop importante, ce qui signifie que l'on considère que l'influence d'un site sur son environnement diminue beaucoup plus rapidement que la réalité. Plus le variogramme prend une valeur grande, plus l'influence est faible, inversement à la covariance qui est plus faible lorsque deux jeux de données sont corrélés. Ces remarques nous incitent à rejeter le modèle linéaire avec une pente de 12 et de prendre garde à cette étape du variogramme.

Dans le second modèle, la fonction sphérique est plus adaptée au variogramme expérimental. De plus, il est favorable de prendre une fonction sphérique, dérivable sur les réels strictement positifs, que



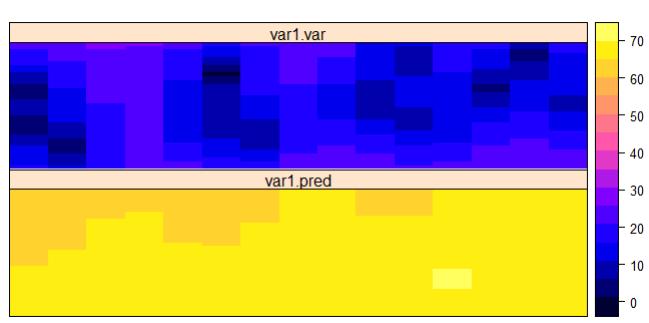


FIGURE 27 – Krigeage par un modèle sphérique de paramètres d'influence 0.39527463, palier 953.8942, de portée 0.06105141. En haut, la vriance sur les estimations, en bas, les estimations.

le modèle linéaire qui risque par ailleurs d'augmenter l'écart type de l'estimateur comme rappelé plus tôt. Cette fois ci, l'estimation fourni un écart type de moins de 30 dB, de moyenne 7,8 dB, graphique du haut de la figure 27. L'écart-type de l'estimateur est bien plus faible que le modèle précédent, mais reste significatif. Pour le réduire, il faut augmenter le nombre de site d'observation. On observe par contre une libéralité recherchée dans les estimations, car représentatif du phénomène physique. En d'autres termes, il y a une absence de structure visible comme précédemment.



7 Tests réalisés sur le système

7.1 Tests unitaires

Pour vérifier le fonctionnement correcte des algorithmes de krigeage, nous avons réalisé des tests unitaires des méthodes de chaque classe sous Python. Ce sont les fichiers débutant par le nom *test_* dans le projet sous notre GIT.

Nous avons dans ces tests repris des données issues d'applications que l'on peut calculer à la main, avec peu de sites observés et donc peu de calcul. Les méthodes ont été écrites de façon à ce qu'elles n'aient qu'une responsabilité, une tâche à réaliser, ce qui rend en plus de la clarté au code, une facilité à réaliser un test unitaire pour chaque étape de l'algorithme de krigeage. On s'intéresse particulièrement aux méthodes calculant les matrices de distance, matrice H, et l'estimation finale afin de valider les algorithmes écrits. Tous les tests sont corrects.

7.2 Tests d'intégration

Pour réaliser le lien entre les deux axes du projet, le traitement des fichiers audios par deep learning et la cartographie, on travaille avec un fichier texte, produit à l'issu des prédictions, et les conteneants, ainsi que la localisation spatiale du son recueilli. Dans la suite, toutes les données d'une même ligne seront séparées par une virgule, ce qui permet de séparer les données dans un programme sous Python facilement. Les données sur chaque fichier audio donné dès le début sont reprises et on y ajoute les labels estimés par deep learning.

Ensuite, un programme python permet de calculer la moyenne des intensités sonores par label et par position. Ces nouvelles données sont stockées dans un nouveau fichier texte. Un fichier correspond à un label, par exemple la catégorie voiture à un fichier propre. On retrouve dans un fichier texte plusieurs lignes, dont chacune correspond à un site d'observation. Dans une ligne on retrouve donc trois informations, la longitude, la latitude et l'intensité sonore. Les méthodes dans les classes concernant le krigeage peuvent par elle-même traiter ces données sous le format expliqué.



8 Réalisation de la partie Web

8.1 Introduction

La traduction des besoins des utilisateurs en documents conceptuels et techniques est une étape primordiale pour la mise en œuvre du projet, c'est avec cette traduction qu'on peut avoir une idée claire et concise sur les différentes fonctionnalités de l'application. Nous commençons ce chapitre par l'identification des acteurs, suivie par l'identification des besoins fonctionnels, ainsi que les besoins non fonctionnels. Ensuite, à partir de ces spécifications nous allons élaborer une analyse des besoins et cela par l'illustration des différents diagrammes de cas d'utilisations.

8.2 Identification des acteurs

Notre application va interagir avec 2 acteurs qui vont être regroupés selon leurs fonctionnalités :

- L'administrateur : c'est l'acteur principal de l'application, il gère la majorité des fonctionnalités.
- Utilisateur : c'est un acteur secondaire qui bénéficie de certaines fonctionnalités dans les limites des droits d'accès qu'il possède.

8.3 Identification des besoins fonctionnels

Les besoins fonctionnels représentent les fonctionnalités que l'application va offrir aux acteurs prédefinis précédemment. Ces fonctionnalités seront regroupées dans ce qui suit selon ces 2 types d'utilisateurs.

8.3.1 Besoins fonctionnels de l'administrateur

- L'authentification: par la saisie de son pseudo et de son mot de passe.
- La gestion des utilisateurs: ajouter, modifier, consulter, supprimer, archiver les utilisateurs.
- La gestion des villes et quartiers: ajouter , modifier, consulter, supprimer, et archiver les villes et des quartiers.
- La gestion des emplacements des capteurs: ajouter , modifier, consulter, supprimer, et archiver les emplacements des capteurs.

8.3.2 Besoins fonctionnels de l'utilisateur

- L'authentification: par la saisie de son pseudo et de son mot de passe.
- La consultation de son profil : un utilisateur peut consulter son profil, modifier et mettre à jour ses informations personnelles.
- La consultation des statistiques concernant la pollution sonore urbaine par emplacement, par région, par ville, et par intervalle du temps.

8.4 Identification des besoins non fonctionnels

Les exigences non fonctionnelles décrivent les caractéristiques générales d'un système et ils sont nécessaires pour juger le fonctionnement d'un programme. Notre application doit satisfaire les facteurs de qualité suivants :

- L'ergonomie : l'application doit présenter une interface et une hiérarchie d'informations claires pour garantir la compréhension du déroulement des différents processus.
- La facilité d'utilisation : l'utilisation doit être simple et ne demande aucun acquis ou connaissance supplémentaire.
- La maintenabilité : le code doit être développé de telle sorte qu'il soit facile à maintenir, et cela va être garanti dans notre cas, grâce à l'utilisation du framework Django.
- Portabilité : l'application devrait fonctionner à partir de n'importe quel appareil et sur n'importe environnement de développement.



- Convivialité : les utilisateurs doivent pouvoir satisfaire leurs besoins de manière efficace et efficiente en utilisant notre application. Cette dernière doit être facile à manipuler.

8.5 Analyse des besoins

Pour approfondir les implications des besoins, nous les modélisons en utilisant des diagrammes de cas d'utilisation raffinés avec une brève description textuelle.

8.5.1 Cas d'utilisation « S'authentifier »

L'authentification est l'une des fonctionnalités indispensables de l'application, c'est à travers laquelle le système peut différencier entre ses différents acteurs ainsi que leurs droits d'accès. La figure suivante illustre le diagramme de cas d'utilisation de cette fonctionnalité.

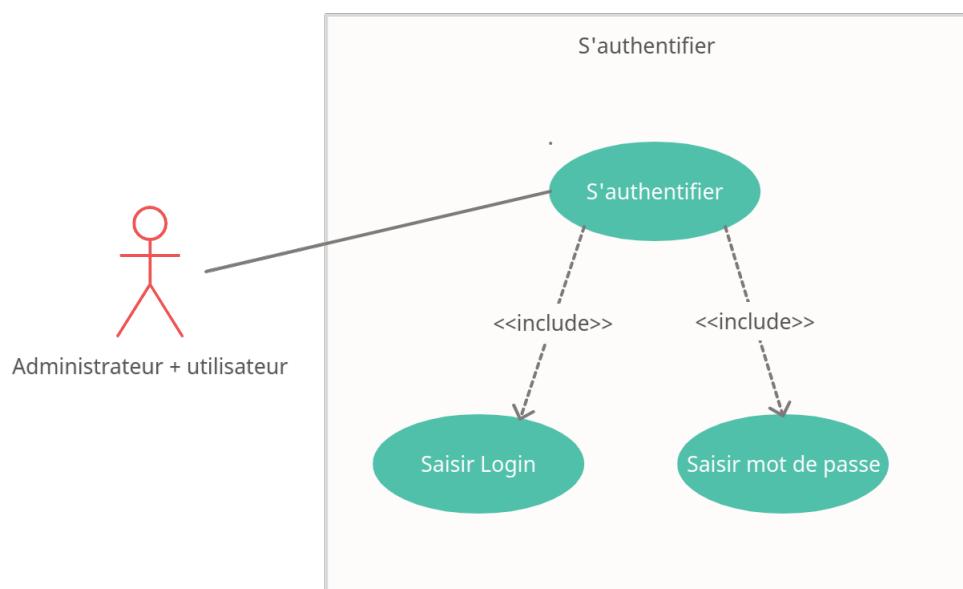


FIGURE 28 – Diagramme de cas d'utilisation « S'authentifier »

Le tableau suivant décrit la description textuelle de cette fonctionnalité:

NOM	S'authentifier
Acteur	Administrateur – Utilisateur
Description	L'utilisateur saisit son pseudo et son mot de passe afin d'accéder à l'application.
Pré-condition	Être déconnecté.
Scénario nominal	1 - L'utilisateur lance l'application. 2 - Il se dirige vers l'onglet d'authentification. 3 - Il saisit son pseudo et son mot de passe. 4 - Il clique sur « login ».
Scénario alternatif	Quand l'utilisateur utilise plus d'une fois l'application sur le même terminal, il peut ne pas mentionner son login qui est déjà mentionné (mais reste modifiable) car le système se rappelle du dernier login utilisé
Scénario d'exception	Car le système se rappelle du dernier login utilisé. Scénario d'exception Le système n'accepte pas l'authentification si le pseudo ou le mot de passe saisi est incorrect

8.5.2 Cas d'utilisation « Gérer les emplacements des capteurs»

Le diagramme illustré par la figure suivante permet d'expliquer de plus le processus de la gestion des emplacements des capteurs sonores.

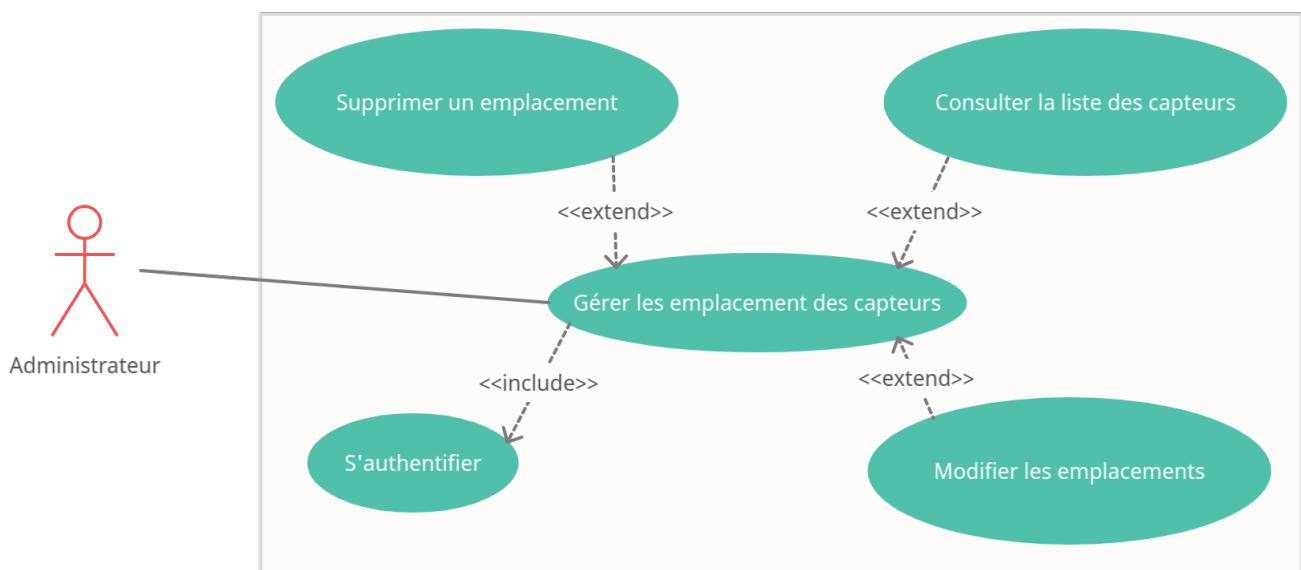


FIGURE 29 – Diagramme de cas d'utilisation « emplacement capteur »

Le tableau suivant décrit la description textuelle du processus de la gestion des emplacements des capteurs.



Nom	Gérer les emplacements des capteurs
Acteur	Administrateur
Description	Permettre à l'administrateur de faire une gestion des emplacements des capteurs.
Pré-condition	L'administrateur est déjà authentifié.
Scénario nominal	1 - L'administrateur clique sur le menu « Emplacement Capteur ». 2 - L'administrateur peut consulter la liste des capteurs dans une ville précise et dans un quartier précis. 3 - Il peut choisir un capteur pour modifier ses informations ou pour le supprimer à travers des boutons spécifiques
Post-condition	- Les informations d'un capteur ont été modifiées. - Le capteur est archivé.

8.6 Réalisation

Après avoir traité l'étude conceptuelle du projet, nous allons entamer dans ce présent chapitre la partie réalisation et mise en œuvre des résultats obtenus. Pour cela, nous commençons par la présentation de l'environnement matériel et logiciel choisi pour le développement de la solution proposée.

8.6.1 Environnement de développement

Environnement matériel:

Pour la réalisation du projet, nous avons utilisé un ordinateur portable HP qui présente les caractéristiques suivantes :

- Processeur : Intel(R) Core™ i5-4300U CPU @2.50 GHz
- Mémoire installée (RAM) : 8,00Go.
- Système d'exploitation : Ubuntu 20.04

Environnement logiciel:

Pour mener à terme notre projet, nous avons opté pour les outils logiciels suivants :

- Visual studio code : c'est un éditeur de code multiplateforme qui offre un environnement flexible à manipuler ainsi qu'il possède un système d'auto-complétion et une console de débogage qui facilite la gestion des fichiers de code volumineux.
- MySQL est un serveur de bases de données relationnelles SQL développé dans un souci de performances élevées en lecture, ce qui signifie qu'il est davantage orienté vers le service de données déjà en place que vers celui de mises à jour fréquentes et fortement sécurisées. Il est multi-thread et multi-utilisateur.
- Creately : c'est un logiciel de conception en ligne qui permet de modéliser les traitements informatiques et leurs bases de données associées et de réaliser tous les types de modèles informatiques.

Choix technologiques:

Faire le choix technologique est une étape primordiale dans la mise en œuvre d'un projet et qui demande d'être bien étudié afin d'aboutir à une meilleure productivité.

Après une étude d'une variété de technologies web, nous avons choisi de travailler avec les frameworks suivants :

- Django: est une infrastructure d'application (aussi appelé framework) côté serveur extrêmement populaire et dotée de beaucoup de fonctionnalités, écrite en Python. Ce module vous montrera pourquoi Django fait partie des frameworks web les plus populaires ainsi que comment l'installer, le mettre en place, et s'en servir afin de créer vos propres applications web.
- Inversion de contrôle (IOC) : c'est un principe de conception qui se base sur l'inversion du contrôle. Ici, ce n'est plus l'application qui appelle les fonctions d'une librairie, mais un framework



qui, à l'aide d'une couche abstraite mettant en œuvre un comportement propre, va appeler l'application en implémentant. L'idée principale est que les modules de hauts et bas niveaux doivent dépendre d'interfaces, et non les modules dépendent les uns des autres. L'inversion de contrôle les rend donc indépendants entre eux. Ainsi les composants ou les services, qui sont des composants distants, forment des entités autonomes et réutilisables sans avoir modifier l'ensemble des codes sources.

- Injection de dépendances (DI) : c'est un modèle qui permet d'implémenter IOC, où le contrôle à inverser est appliqué dans le réglage des dépendances des objets. Le conteneur est chargé d'injecter les dépendances lorsqu'il crée le haricot. En utilisant le principe DI. L'objet n'est plus obligé à rechercher ses dépendances car le DI s'en chargera, par conséquent, le code devient plus propre et le découplage est plus efficace.
- Bootstrap : c'est un framework CSS qui garantit la mise en forme des pages web. Son avantage principal est d'assurer un aspect unique quel que soit le navigateur grâce à une compatibilité totale intégrée. Mais le second avantage, le plus illustré, est le fait de pouvoir disposer d'une application responsive, c'est-à-dire une application qui s'adapte automatiquement en fonction de l'écran ou du périphérique utilisé.



9 Bilan sur la gestion de projet

9.1 Diagramme de Gantt

Lors des premières semaines, un diagramme de Gantt a été réalisé afin d'avoir une première vue sur les besoins que nous devrions 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 (choix du meilleur aspect du spectrogramme), voir presque supprimées pour cause de difficultés, comme la partie recueil de données réelles.

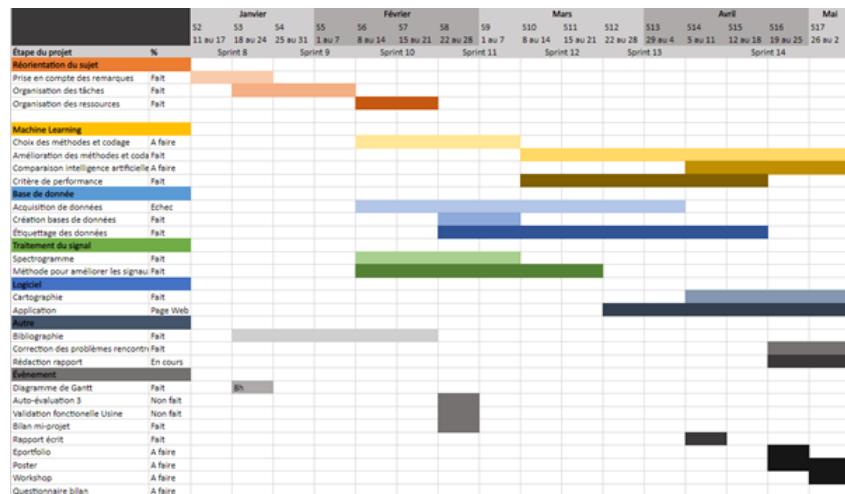


FIGURE 30 – Le diagramme de Gantt était un premier aperçu sur les étapes du projet.

9.2 Organisation de groupe par la méthode Agile

La méthode Agile a été utilisée pour le bon avancement du projet. Le consultant de l'organisation en groupe est Sylvain GUERIN. Trois rencontres ont eu lieu au total 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, R. AL JOHANI

La partie intelligence artificielle et base de données a été principalement réalisée par T. ALBERT et E. FERRAND. La partie sur l'estimation spatiale et la cartographie par H. FAUVEL. L'application web a été réalisée par R. AL JOHANI.

Concernant l'organisation des sprints, 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-rendus 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. Pour plus d'informations sur notre organisation, voici les liens vers notre Teams, Eportfolio et Trello.



10 Conclusion

D'un point de vue technique, la classification multi-labels, avec 8 classe différentes était ambitieuse, certainement trop, et il aurait fallu procéder avec moins de labels en premier lieu. Cela nous aurait permis de mieux analyser le comportement des différentes couches du réseau de neurones. En ce qui concerne la cartographie, on peut retenir que les méthodes statistiques sont plus fiables que les méthodes déterminismes, et qu'elles ont été performantes pour répondre à notre problématique.

Même si nous n'avons pas pu aboutir à un fonctionnement de bout en bout, de nombreux enchaînements de fonctions sont possibles, et chacun a œuvré pour qu'ils le soient. Au-delà du résultat, certes en dessous de nos espérances, nous sommes très satisfaits d'avoir pu nous épanouir, chacun dans notre domaine, à la compréhension et mise en œuvres des différentes techniques nécessaires. Nous n'en avons pas pour autant oublié de collaborer, et la mise en commun de nos apprentissages a permis à chacun de tirer une plus-value pédagogique polyvalente de ce projet.



Table des figures

1	Diagramme du cycle de vie du système séparé en quatre phases.	9
2	Diagramme en pieuvre de la première phase de vie du système.	11
3	Diagramme en pieuvre de la deuxième phase de vie du système.	12
4	Diagramme en pieuvre de la troisième phase de vie du système.	12
5	Diagramme en pieuvre de la quatrième phase de vie du système.	13
6	Diagrammes internes FAST de la première phase de vie.	14
7	Diagramme interne FAST de la deuxième phase de vie.	15
8	Diagrammes internes FAST de la troisième phase de vie.	15
9	Diagrammes internes FAST de la quatrième phase de vie.	16
10	Diagramme de l'architecture physique du système.	16
11	Insertion d'un réseau de neurones pré-entraîné pour un problème particulier de classification	19
12	Matrice de confusion - Classification binaire	20
13	Table des estimateurs de l'abondance en fonction de la grille des emplacements des sites d'observations.	25
14	Fichier «data» créé après labellisation des échantillons, associant à chaque spectrogramme ses labels.	26
15	Trois spectrogrammes d'un même échantillon, affichant l'intensité en fonction de la fréquence et du temps.	27
16	Translation temporelle bilatérale de 1.25 secondes, respectivement antérieurement à gauche, postérieurement à droit, et sans translation au centre	28
17	Moyennage de zones fréquentielles et temporelles (figures 2 et 3), à partir d'un spectrogramme original.	29
18	Spectrogrammes de la pie, de la mouette, de la voiture et de la moto.	30
19	Spectrogrammes du chien, de la voix et du vélo	30
20	Réseaux de neurones convolutifs mis en place.	31
21	Modèle VGG-16 et Insertion de se réseau dans notre propre réseau	32
22	Matrices de confusion	34
23	Grille d'implantation du capteur.	36
24	Figure de l'ellipse d'incertitude sur les données GPS.	38
25	Présentation des données, en haut à gauche, histogramme des données, en bas à gauche, carte de l'emplacement des sites d'observation, à droite, variogramme expérimentale avec le variogramme décidé en rouge, linéaire, et en bleu, sphérique.	40
26	Krigeage par un modèle linéaire de pente 12.	41
27	Krigeage par un modèle sphérique de paramètres d'influence 0.39527463, palier 953.8942, de portée 0.06105141. En haut, la vriance sur les estimations, en bas, les estimations. .	42
28	Diagramme de cas d'utilisation « S'authentifier »	45
29	Diagramme de cas d'utilisation « emplacement capteur »	46
30	Le diagramme de Gantt était un premier aperçu sur les étapes du projet.	49



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 », 2020.
- [3] Pierre Bosser, « Introduction à l'interpolation spatiale et aux géostatistiques », Chapitre 3, 2020.
- [4] Pierre Chauvet, « Aide mémoire de géostatistique linéaire », Presses Des Mines, avril 2008.
- [5] Pierre Bosser, « Global Navigation Satellite Systems », 2020.
- [6] F. Chollet, « Deep Learning with Python », 1st édition. Shelter Island, New York: Manning Publications, 2017.
- [7] J. Dayhoff, « Neural Network Architectures », 1990, Hardcover, 1 juin.
- [8] A. Géron, « Deep Learning avec Keras et TensorFlow: Mise en oeuvre et cas concrets », Dunod, 2020. 6 juin.
- [9] Ricco Rakotomalala, « Tests de normalité. Techniques empiriques et tests », Université Lumière Lyon 2, version 2.0, 2011.
- [10] Xavier Emery, « Géostatistique linéaire », Presses Des Mines, 2001.
- [11] Philippe Réfrégier, « Théorie du signal : Signal-Information-Fluctuations », Masson, 1993.

