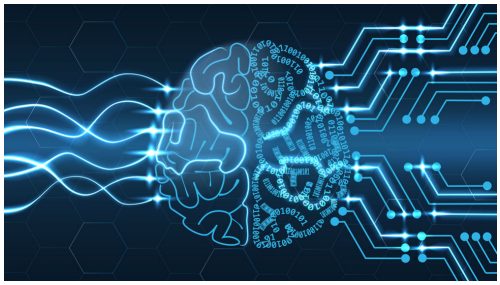
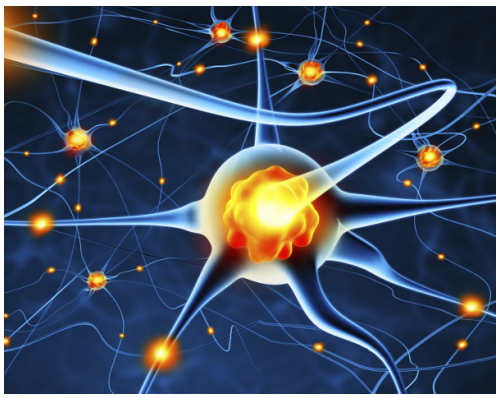


RAPPORT DE STAGE

Classification des Cachalots par leurs clics

Arekion Alexis
M1 Informatique
Année universitaire 2019/2020



Université des Antilles
Laboratoire de Mathématiques Informatique et Application (LAMIA)

Table des matières

Chapitre 1

Introduction

Dans ce premier chapitre, je vais dans un premier temps présenter la structure d'accueil à savoir l'**Université des Antilles**, son laboratoire **LAMIA** et le groupe **SpikeTrain** dans laquelle j'ai été accueilli. Puis je m'attarderais sur le contexte du stage et le challenge sur lequel travaillaient mes tuteurs lorsque j'ai commencé mon stage, ainsi que leur état d'avancement sur celui-ci. Ensuite, je présenterais les outils théoriques indispensables à la résolution du challenge à savoir les neurones artificiels, les réseaux de neurones qui en découlent et enfin les réseaux de neurones convolutifs que l'on va utiliser pour résoudre notre probléme. Enfin, je présenterais le plan de ce rapport.

1.1 Présentation de la structure d'accueil

Durant la période de mon stage, j'ai été accueilli au sein du groupe **SpikeTrain** du **Laboratoire de Mathématiques Informatique et Applications (LAMIA)** de l'**Université des Antilles (UA)**. Je vais donc présenter ces trois entités, en commençant par l'Université des Antilles

1.1.1 L'Université des Antilles

L'Université des Antilles est un établissement public de l'enseignement supérieur qui entre dans la catégorie des établissements à caractère scientifique, culturel et professionnel (EPSCP). Il est en charge de missions, régies par l'article L123-3 du code de l'éducation et qui concernent :

- La formation initiale et continue tout au long de la vie ;
- La recherche scientifique et technologique, la diffusion et la valorisation de ses résultats au service de la société ;
- L'orientation, la promotion sociale et l'insertion professionnelle ;
- La diffusion de la culture humaniste, en particulier à travers le développement des sciences humaines et sociales, et de la culture scientifique, technique et industrielle ;
- La participation à la construction de l'Espace européen de l'enseignement supérieur et de la recherche ;
- La coopération internationale.

L'Université des Antilles s'organise autour deux pôles universitaires régionaux dotés d'une certaine autonomie : le *Pôle Guadeloupe* et le *Pôle Martinique*. Sur ces pôles, l'Université assure des missions de *formation* et de *recherche*, par ses *enseignants-chercheurs*, Maîtres de Conférences

ou Professeurs des Universités, ses *chercheurs* et ses *enseignants*, assistés par des *personnels administratifs et techniques*.

Administration et personnel technique

l'UA emploie 414 agents administratifs et techniques (environ 200 personnes pour l'administation centrale et 100 répartis sur chaque pôle)

Enseignements

L'UA délivre des diplomes de la licence au doctorat dans de nombreux domaines. Au total, cela représente :

- 484 enseignants-chercheurs (environ 240 pour chaque pôle)
- 12 000 étudiants (environ 7000 pour la Guadeloupe , 5000 pour la Martinique)

Pour l'informatique, cela représente : environ 20 enseignants-chercheurs pour 200 étudiants.

Recherche

La recherche à l'Université des Antilles est structurée en laboratoires auxquels sont rattachés les enseignants chercheurs qui peuvent notamment former de futurs chercheurs : les doctorants.

L'Université compte ainsi au total :

- 17 laboratoires
- 320 doctorants

Pour ma part, comme signalé précédemment, j'ai effectué mon stage dans le laboratoire LAMIA que je vais maintenant présenter.

1.1.2 Le laboratoire LAMIA

Le **Laboratoire de Mathématiques Informatique et Application (LAMIA)**, comme son nom l'indique, se concentre sur les recherches en informatiques et mathématiques.

Il compte une soixantaine de membres (Professeurs des Universités, Maitres de Conférences, Attachés Temporaires d'Enseignement et de Recherche, Doctorants) répartis sur deux pôles (Guadeloupe et Martinique) au sein de trois équipes internes :

- Equipe **Mathématiques** (analyse variationnelle, analyse numérique, EDP, analyse statistique, mathématiques discrètes) ;
- Equipe Informatique **DANAIS** : Data analytics and big data gathering with sensors ;
- Equipe Informatique **AID** : Apprentissages Interactions Donnees ;

De plus, le LAMIA accueille en son sein un groupe de chercheurs associés travaillant en Epidémiologie Clinique et Médecine.

Indépendamment de ces équipes, depuis 2019, les travaux de recherche du laboratoire se répartissent en **projets** qui peuvent réunir des membres de plusieurs équipes en **groupes de travail**. Mon stage était en fait plus attaché à un projet et un groupe de travail qu'à une équipe.

Ce groupe de travail, nommé **SpikeTrain**, concerne l'utilisation de **réseaux de neurones**, et en particulier leur variante **impulsionnelle** pour l'apprentissage automatique. Ce groupe de travail réunit à l'heure actuelle :

- 1 Professeur des Universités
- 2 Maîtres de Conférences avec une Habilitation à Diriger des Recherches
- 3 Maïtres de Conférences
- 1 Ingénieur d'Études.

.

C'est avec ces personnes que j'ai travaillé tout au long du stage et plus particulièrement avec mes tuteurs de stage qui sont **M. Vincent PAGÉ et M. Manuel CLERGUE**.

La prochaine section sera consacrée à la présentation de la thématique de recherche du groupe SpikesTrain et de mon stage.

1.1.3 Le groupe SpikeTrain

Comme signalé plus haut, le groupe **SpikesTrain** s'intéresse aux techniques d'**Intelligence Artificielle**, plus spécifiquement à l'**apprentissage automatique** dont l'objectif est de créer des programmes capable d'apprendre à partir de bases d'exemples.

Actuellement, parmi les techniques permettant l'apprentissage automatique, une se démarque et est très populaire : les **réseaux de neurones artificiels**, notamment dans leur version *profonde*, qui sont très utilisés par exemple par **Facebook™** pour sa **reconnaissance faciale** ou encore par **Google™** pour ses **robots** qui apprennent par **répétitions** à jouer à des jeux de stratégies comme les échecs ou le Go.

Les réseaux de neurones artificiels, bien que très performants, souffrent d'un défaut majeur : leur consommation électrique est très importante. Cela risque de poser à terme de nombreux problèmes environnementaux, si ces techniques se généralisent. La version impulsionnelle des réseaux de neurones, plus proche du fonctionnement des neurones naturels et du cerveau, est plus parcimonieuse. Actuellement, sur quelques applications, les réseaux de neurones impulsionnels arrivent à des performances équivalentes aux réseaux de neurones artificiels classique, avec une consommation dix à cent fois moindre. Cette propriété les rend particulièrement adaptés pour faire du traitement *in situ* (à la sortie des capteurs) dans des applications de surveillance de l'environnement par exemple. C'est justement une telle application qui a amené le groupe SpikeTrain à s'intéresser au challenge présenté dans la section suivante.

Bien qu'un des axes de recherche du groupe **SpikeTrain** concerne les neurones impulsionnels, ce ne sont pas ceux que nous avons utilisés dans ces travaux, et ils ne seront pas évoqués dans ce rapport. Il s'agit de travaux préliminaires pour déterminer les performances des techniques classiques afin de pouvoir les comparer avec ce qu'il pourra être fait avec les réseaux de neurones impulsionnels.

1.2 Contexte du stage

1.2.1 Le Challenge : Classifier les Cachalots

Le challenge "Dyni Odontocete Click Classification"[1] consiste à réaliser un classifieur qui classe des mammifères marins de dix espèces différentes à partir de leurs "clics", c'est à dire le son qu'ils émettent avec leur machoires.

Pour celà nous avions à disposition une **base labelisée** (comportant 113000 exemples) ainsi qu'une **base de test**, non labelisée (comportant 20000 exemples). Comme dans tout challenge de ce type, ces deux bases ont des rôles très différents :

- La **base labelisée** sera découpée en deux par les participants :
  - une **base d'apprentissage** permettant de régler les paramètres des algorithmes.
  - une **base de validation** permettant d'évaluer les performances des algorithmes sur des exemples qu'ils n'ont jamais vus.

1. https ://challengedata.ens.fr/challenges/32

- La **base de test** sert pour l'évaluation par les organisateurs du challenge. Quotidiennement, les participants peuvent déposer leurs prédictions sur cette base (Deux fois par jour maximum). On peut ainsi définir les performances des différents participants.

Ci dessous, une traduction approximative du descriptif du challenge tel qu'il est présenté sur le site du challenge :

> Les deux bases sont constituées d'enregistrements audios des clics des différentes espèces. Chaque enregistrement contient 8192 mesures faites à une fréquence d'échantillonnage de 200KHz. Dans le cas de la base non labelisée appellée base de test chaque enregistrement contient normalement un clic centré au milieu de la fenêtre tandis que dans la base labelisée appellée base d'apprentissage, le clic n'est pas spécialement centré et peut se situer à divers moments de l'enregistrement. De plus, les enregistrements peuvent contenir divers bruits.
>
> L'objectif est de classer chaque enregistrement en fonction de l'espèce émettrice correspondante. Les 10 espèces sont :
>
> 0 Gg : Grampus griseus- Dauphin de Risso
>
> 1 Gma : Globicephala macrorhynchus- Baleine pilote à nageoires courtes
>
> 2 La : Lagenorhynchus acutus- Dauphin à flancs blancs de l'Atlantique
>
> 3 Mb : Mesoplodon bidens- Baleine à bec de Sowerby
>
> 4 Me : Mesoplodon europaeus- Baleine à bec de Gervais
>
> 5 Pm : Physeter macrocephalus - Cachalot
>
> 6 Ssp : Stenella sp.Dauphin stenellide
>
> 7 UDA : Delphinidés de type A - un groupe de dauphins (espèces non encore déterminées)
>
> 8 UDB : Delphinidés de type B - un autre groupe de dauphins (espèces non encore déterminées)
>
> 9 Zc : Ziphius cavirostris - Baleine de Cuvier à bec
>
> La performance du classifieur est évaluée sur la base de test (les labels sont envoyés au site du challenge) par une mesure d'accuracy (le nombre de bien classés sur le nombre total d'exemples).

1.2.2 Etat des lieux à mon arrivée

Quand j'ai commencé mon stage, mes tuteurs avaient déjà commencé le challenge depuis un mois. J'ai ainsi dû, dans un premier temps, me mettre à jour sur le challenge et ce qu'ils avaient fait, à savoir :

- Un grand nombre de tentatives de résolution du probléme uniquement basé sur du machine learning, notament des réseaux de neurones et des réseaux de neurones convolutionnels ;
- Divers traitements des signaux bruts ;
- De l'augmentation de données.

Leurs meilleurs résultats étaient les suivants :

- de l'ordre de 98% de réussite en accuracy sur la base de validation
- de l'ordre de 72% de réussite sur la base de test

Cet écart de 20 points entre les deux bases persistait même en utilisant d'autres méthodes de machine learning donnant de moins bons résultats. Cet important différentiel est d'autant plus surprenant que les auteurs du challenge nous présentent les données de la base non labelisée comme des données de meilleure qualité que celles de la base labelisée.

C'est pour mieux comprendre ce différentiel mais surtout le problème dans son ensemble que j'ai été chargé de créer un certain nombre d'outils facilitant l'analyse et la visualisation des données et des effets des traitements que nous leurs appliquons.

Avant de pouvoir commencer cette partie du travail, il m'a fallu prendre en main un certain nombre de concepts et d'outils, concernant les **réseaux de neurones**, que nous allons voir ensemble maintenant.

1.3 Réseaux de neurones artificiels

Les réseaux de neurones artificiels sont une technique d'apprentissage automatique particulièrement puissante. Ils parviennent efficacement à détecter les régularités statistiques des entrées qui leurs sont présentés afin de les répartir en classes. Cette technique a enregistré des succès phénoménaux ces dix dernières années, dans tous les domaines, de la médecine aux jeux de stratégie, en passant par la reconnaissance de formes. La description précise de leur fonctionnement dépasse largement le cadre de ce rapport. Nous nous contenterons de donner quelques éléments permettant de comprendre ce fontionnement.

Nous débuterons par la présentation de l'unité de base, le neurone artificiel, puis de comment les neurones artificiels sont combinés en réseau permettant de réaliser des tâches de classification. Enfin, nous présenterons les réseaux de neurones convolutifs, utilisés pour faire de la reconnaissance de formes et de la classification d'images ou de sons.

1.3.1 Neurone artificiel

Un neurone artificiel est pourvu d'un certain nombre d'**entrées**. Dans le cas des neurones classiques, ces entrées sont des nombres réels. Le neurone calculera, en fonction de ces entrées, une unique valeur en **sortie**. Détaillons la façon dont ces calculs sont effectués :

Chacune de ces entrées circule sur une connection, laquelle est caractérisée par un **poids** qui définit l'importance de l'entrée pour le neurone.

Le neurone calcule dans un premier temps la somme de ses entrées, pondérée par leurs poids respectifs, à laquel vient s'ajouter un **biais** spécifique à chaque neurone (cf. equation 1.1).

$$y = \sum_{i}^{n} w_i \times x_i + b \tag{1.1}$$

Le résultat de cette somme passe alors dans une **fonction d'activation** qui permet d'introduire une non-linéarité dans les calculs. La sortie s du neurone est donc calculée conformément à l'équation 1.2

$$s = f(\sum_{s=0}^{n_x} x_n w_n + b) \tag{1.2}$$

Un schéma reprenant ces explications est présenté dans la figure 1.1.

Notre apprentissage se fera en modifiant les poids de ses différentes connexions (et le biais) de façon à obtenir une sortie proche de celle voulue.

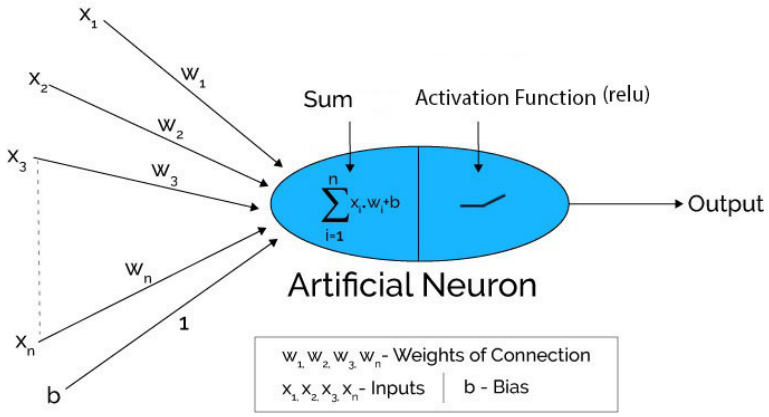


FIGURE 1.1 – Fonctionnement d'un neurone seul.

1.3.2 Réseau de neurones classique

Les neurones présentés précédemment prennent tout leur intérêt lorsqu'ils sont utilisés en groupes, dans des **réseaux de neurones**.

Le premier de ces réseaux, encore utilisé de nos jours, est appelé **perceptron**. Le principe du perceptron n'est pas nouveau et date des années 1960.

Dans ces types de réseaux, les neurones sont organisés en **couches** (une seul couche pour le perceptron et plusieurs pour le perceptron multicouche). La première couche correspond à celle qui permettra d'introduire des informations dans le réseau (comme la rétine par exemple). Elle est nommée **couche d'entrée**. La dernière couche permettra de lire les décisions du réseau. Elle est appelée **couche de sortie**. Dans les applications classiques, à chaque neurone de la couche de sortie correspond une décision possible et le neurone qui est le plus activé sur la couche de sortie l'emporte. Entre ces couches, on trouve souvent un nombre variable de couches intermédiaires appelées **couches cachées**.

Entre deux couches, on établit le plus souvent un schéma de connexion qualifiée de *full connected* ou *dense*. Dans ce cas, chaque neurone d'une couche est connecté avec chaque neurone de la couche suivante. Pour le bien de ce rapport, nous n'épiloguerons pas sur les autres types de connexions existantes. Ce type d'architecture est présentée sur la figure 1.2.

1.3.3 Réseau de neurones convolutifs

Les réseaux de neurones **convolutifs** sont un type de réseau de neurones inspirés par le cortex cérébral des animaux. Ils possèdent de larges applications dans la reconnaissance d'images, de vidéos et de sons.

Ce réseau se présente comme un réseau classique , une couche d'entrée, une couche de sortie et des couches cachées. Neanmoins, il s'en distingue par la composition des couches cachées.

Ces réseaux ayant été conçus initialement pour des applications a des images, nous illustrerons notre propos ici sur ce type de données, avant de passer à l'implémentation utilisée dans le cas du challenge.

les features map

En entrée et en sortie d'une couche d'un réseau convolutif, on trouve des **features map**, que nous nommerons ici **cartes** pour cartes de caractéristiques. Par exemple, en entrée d'un réseau

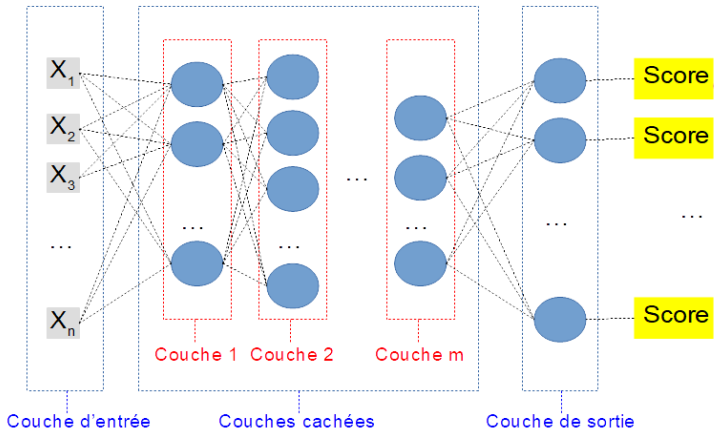


FIGURE 1.2 – réseau de neurones classique en couches.

traitant des images RGB, les entrées seraient composée de trois cartes, correspondant aux trois canaux de l'image.

Une sortie de la première couche d'un réseau convolutif est également un ensemble de cartes qui pourraient, par exemple, correspondre aux emplacements, dans l'image de départ, de traits verticaux (carte 1), de traits horizontaux (carte 2)...

Ces cartes, dans les frameworks actuels, sont représentées sous forme de tenseurs (des tableaux de dimension variable. Le tenseur représentant une image RGB peut être imaginé comme un cube de données de dimension 3 (hauteur, largeur, canal).

les couches convolutives

Les réseaux convolutifs possèdent des couches **convolutives** pour lesquelles chaque neurone va appliquer un filtre convolutif sur les cartes d'entrée.

Plus clairement : Imaginons, qu'une couche doive notamment produire une carte contenant les emplacements des traits verticaux. Dans un réseau classique, cette couche serait composée d'autant de neurones que l'image d'entrée. Pour que les neurones détectent tous des traits verticaux à différents emplacements de l'image d'entrée, il faudrait qu'ils fassent tous la même opération (qu'ils aient les même poids), mais opèrent chacun sur une portion variable de l'image d'entrée.

De fait, cette opération d'extraction de traits verticaux est une convolution et peut être faite par un neurone unique qui "se déplace" dans l'image. Dans le cas des neurones convolutifs, on gagne de nombreuses choses par rapport à une convolution classique :

- le neurone peut travailler sur toutes les cartes d'entrées en meme temps (il mixe les informations de ces cartes)
- il intègre une fonction d'activation qui est non linéaire, ce que ne fait pas la convolution de base.
- le neurone va apprendre quelles caractéristiques extraire des cartes d'entrées

La définition d'une couche convolutive est donc très simple : on lui indique le nombre de cartes d'entrées, et le nombre de cartes de sorties désirée. Le réseau possèdera alors, pour cette couche, autant de neurones que de cartes de sorties voulues.

Cette opération est illustrée dans la figure 1.3, avec une couche convolutive placée juste après

l'image d'entrée.

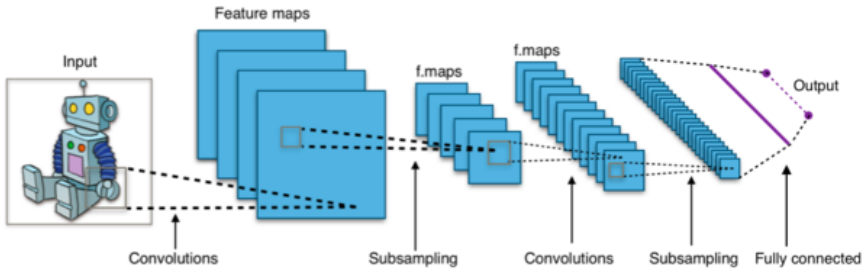


FIGURE 1.3 – Exemple de réseau convolutif. Celui ci présente 2 couches convolutives, entre lesquelles on trouve des couches de pooling, notée ici *subsampling*. La couche de sortie est notée *output*.

En empilant les couches de convolutions, on constate que les caractéristiques extraites par chaque couche sont de plus en plus sémantiques avec la profondeur des couches. Par exemple, la premiere couche va extraire des bords d'orientations diverses. La seconde va combiner ces bords pour extraire des coins. La troisième pourrait détecter des formes basiques (ronds, cercles) et une dernière pourrait combiner tout cela pour differencier des images de robots d'images de chats....

les couches de pooling

Les réseaux convolutifs possèdent aussi des couches de **pooling**. Le **pooling** est une opération simple qui consiste à remplacer une zone de pixels (généralement 2×2 ou 3×3), par une valeur unique (généralement le max ou la moyenne). De cette manière, l'image diminue en taille et se retrouve simplifiée (lissée).

Cette opération est illustrée dans la figure 1.3, avec une couche de pooling placée en seconde position après l'image d'entrée.

En général, les réseaux convolutifs alternent plus ou moins couches de convolution et couche de pooling, avant de passer à la prise de décision finale.

la couche de sortie

La lecture de la décision d'un réseau de neurones convolutifs est rarement lue directement sur les couches convolutives. Il est nécessaire d'utiliser un classifieur pour associer les cartes de caractèristiques à des classes. Dans la plupart des cas, c'est un perceptron multicouche qui est utilisé comme classifieur. Il est possible d'utliser la même méthode d'apprentissage pour la partie convolutive et pour la partie classifieur.

L'un des intérêts majeurs des réseaux convolutifs est qu'ils permettent de réaliser une invariance de translation : un motif appris sur une zone de l'image sera reconnue quelque soit sa position dans l'image. Un autre intérêt est qu'ils nécessitent l'apprentissage de beaucoup moins de paramètres (les poids) que les réseaux de neurones de type Perceptron Multicouche, pour des performances en classification équivalentes ou supérieures.

A titre d'exemple, notre meilleur résultat sur le challenge à été obtenu par un réseau de neurones convolutifs composé de 10 couches convolutives à 1 dimension, alternées avec des couches de pooling, suivies d'une perceptron multicouches à 64 entrées, une couche cachée de 32 neurones et une couche de sortie à 10 neurones (un par classe). Le nombre total de paramètres à apprendre était de 457258. Le temps d'apprentissage était de environ 4 heures sur une carte graphique. Les

performances (accuracy) obtenues étaient de 0.98 sur la base d'apprentissage et de 0.80 sur la base de test.

1.4 Présentation du plan du rapport

Ceci ayant été posé, nous pouvons maintenant présenter le plan de ce rapport, qui sera structuré comme suit :

Après ce chapitre d'introdution, qui présentait le contexte opérationnel, la mission et les pré-requis théoriques de ce stage, le chapitre 2 présentera l'essentiel des travaux que j'ai réalisés durant ce stage.

Mon objectif était de permettre au groupe SpikeTrain de visualiser les signaux injectés dans les réseaux de neurones. Les signaux issus des bases d'exemples du challenge seront présentés dans la section 2.1. Ces signaux passent ensuite au travers d'une multitude de filtres. Certains de ces filtres ont un objectif de prétraitement et seront vus en section 2.3). D'autres filtres sont là pour faire de l'**augmentation de données**, ce qui occupera la section 2.2 que nous avons laissée sous son nom anglophone (*Data Augmentation*) comme c'est le cas dans toutes les documentations que nous avons trouvées. Pour organiser le passage des signaux des bases d'exemples vers l'entrée des réseaux de neurones, les frameworks actuels de réseaux de neurones utilisent la notion de **pipeline de données**, que mes tuteurs et moi même avons fouillés au cours de ce stage. Ils font l'objet de la section 2.4. Enfin, l'objectif final de mes travaux était de produire des outils permettant de générer rapidement des fiches synthétisant les visualisations des signaux à divers emplacement de ce pipeline. Ces fiches et les outils permettant de les créer sont décrits dans la section 2.5.

Cette année 2020 a été très particulière en raison du confinement lié au COVID-19. Ce stage, comme l'ensemble des activités mondiales, a été impacté par ce confinement et a été réalisé en intégralité sous forme de travail à distance. Il nous a donc semblé judicieux d'ajouter à ce rapport un chapitre dédié à ces spécificités (cf. chapitre 3).

Enfin, nous terminerons ce rapport par une conclusion et des perspectives.

Chapitre 2

Analyse et traitement des données

Dans cette partie on va voir l'essentiel du travail que j'ai effectué durant mon stage à savoir la génération de fiches d'analyse de nos données. Dans un premier temps nous verrons les signaux, puis à l'aide d'un zoom les clics émis par les animaux. Ensuite afin d'affiner notre analyse nous verrons leurs transformées de Fourier grâce auxquelles nous obtiendrons leurs spectrogrammes 2D et 3D. Dans un second temps nous verrons l'une des principales méthodes de préparation des données que nous avons effectuée à savoir la *data augmentation* d'abord d'un point de vue théorique puis d'un point de vue pratique avec la data augmentation que nous avons effectuée sur notre base à savoir le rajout de bruit blanc puis la simulation de distance. Dans un troisième temps nous verrons les prétraitements de données que nous avons dûs effectuer pour le bon fonctionnement de nos IA à savoir l'application d'un filtre passe haut puis une mise à l'échelle. Ensuite nous verrons les pipelines qui nous ont permis d'optimiser l'injection des flux de données dans nos réseaux de neurones. Et enfin nous verrons le résultat final regroupant ces différents éléments pour la génération des fiches d'analyse.

2.1 Les signaux

Les données sont fournies par les organisateurs du challenge sous la forme de tableaux d'échantillons. Chaque exemple, un son ou signal temporel correspondant à un clic, est ainsi composé de 8192 échantillons de l'amplitude sonore captée à une fréquence de 200KHz par un hydrophone. Dans la suite du rapport, nous appelerons "signal" un exemple de la base d'apprentissage ou de test, pour conserver à l'esprit la nature des données que nous devons traiter.

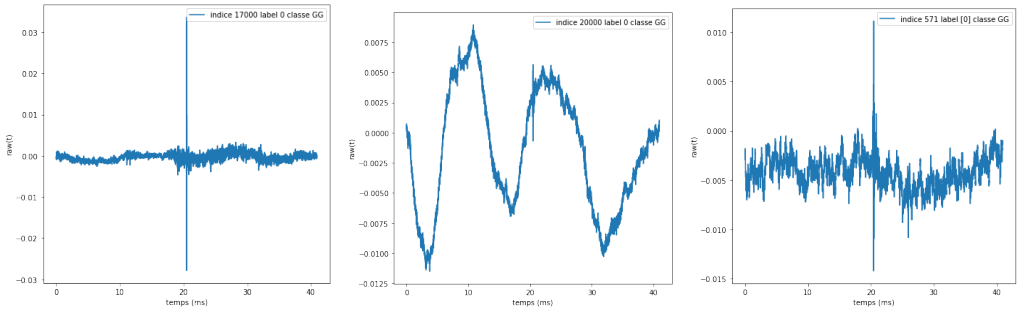
Afin d'améliorer la lisibilité des sections suivantes nous prendrons comme fil rouge les trois mêmes signaux (le n°17000 et le n°20000 de la base labelisée ainsi que le n°571 de la base non labelisée). Nous les avons choisis car ils sont représentatifs de la diversité des signaux que l'on a dans notre base à savoir :

- Des signaux très propres
- Des signaux un peu bruités ou altérés
- Des signaux très dégradés

Nous les observerons sous diverses formes puis nous effectuerons dessus un certain nombre de traitements.

2.1.1 Les signaux bruts

Dans un premier temps, nous commençons par observer les signaux sans traitement, comme des signaux sonores, c'est à dire comme l'évolution dans le temps de l'amplitude du son.



(a) (b) (c)

FIGURE 2.1 – Signaux bruts : (a) n°17000 et (b) n°20000 de la base d'apprentissage ; (c) n°571 de la base de test.

Sur la figure 2.1, nous constatons plusieurs choses : tout d'abord il semble y avoir une certaine disparité entre les signaux, certains étant beaucoup plus bruités que d'autres ; de plus contrairement à ce que l'on pouvait penser, le clic n'est pas toujours facile à distinguer et celui-ci n'est pas non plus toujours bien centré.

Cependant, nous avons pu tirer quelques enseignements de l'observation de ces signaux :

- L'amplitude des clics semble variable : elle est de 0.06 pour le signal 17000 et de 0.020 pour le signal 571, par exemple.
- Il arrive que le bruit soit suffisament important par rapport au clic pour rendre son identification difficile voir impossible, comme pour le signal 20000.

Pour afiner notre analyse, il parait pertinent de commencer par zoomer sur ce clic. Pour cela il va donc faloir commencer par trouver un moyen d'isoler le clic.

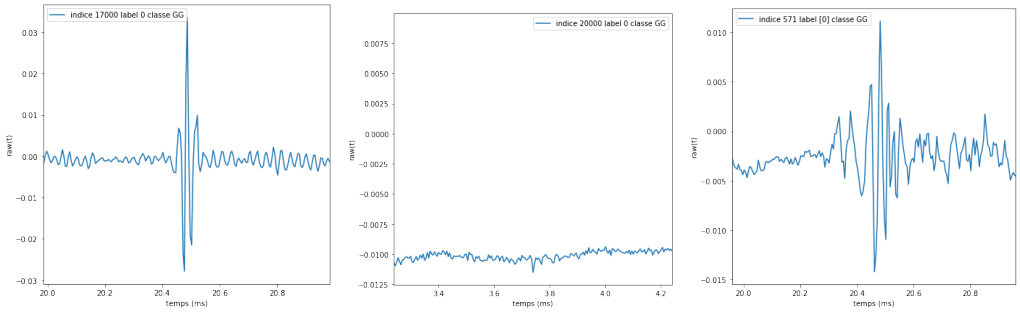
2.1.2 Le zoom

Pour zoomer sur le clic, on identifie le maximum du signal qui sera, idéalement, le milieu du clic puis on rajoute l'équivalent de la durée d'un clic (qui est de l'ordre de 5 millisecondes) avant et aprés ce maximum.

On peut observer le résultat de cette opération sur la figure 2.2. Cela nous permet de constater que cette procédure n'est pas efficace dans tous les cas. En effet si pour les signaux n° 17 000 et 571 le zoom semble bien fonctionner dans le cas du signal n° 20 000 qui est très bruité, le bruit semble avoir pris le dessus sur le clic conduisant à l'echec de l'identification du clic. On va donc par la suite appliquer un filtre pour supprimer une partie des bruits parasites (dont on reparlera dans la partie traitement du signal).

On peut déjà observer plusieurs phénoménes :

- L'efficacité du zoom semble corrélée à la qualité du signal de départ : un nettoyage du signal semble donc nécessaire pour améliorer la détection du clic.
- L'intensité des clics est très variable, cette variation pouvant aller jusqu'a un facteur 10 entre 2 clics : il parait donc pertinent par la suite de les normaliser.
- La durée d'un clic est bien de l'ordre de 0.5 millisecondes.
- Les clics les plus nets semblent bien centrés autour de 200 ms.



(a) 17000 (b) 20000 (c) 571

FIGURE 2.2 – Signaux bruts : (a) n°17000 et (b) n°20000 de la base d'apprentissage ; (c) n°571 de la base de test. avec un zoom temporel

Sous cette forme, nos observations semblent quand même limitées. Nous allons donc les observer sous d'autres formes puis on cherchera à améliorer la qualité de nos signaux via diverses techniques. Etant donné la nature de nos données à savoir des enregistrements audios, observer leurs spectrogrammes semble être pertinent.

2.1.3 Transformée de Fourier

Avant d'observer les spectrogrammes il convient de commencer par expliquer et observer les spectres obtenus par la transformée de Fourier de nos 3 signaux.

La transformée de Fourier est un outil mathématique nous permettant de passer du domaine temporel au domaine fréquenciel, en décomposant le signal de départ en somme de signaux sinusoidaux. Cela permet d'observer les différentes composantes fréquentielles du signal.

Les transformées de Fourier de nos signaux sont présentées sur la figure 2.3 (les courbes ne sont pas du tout à la même échelle en ordonnée).

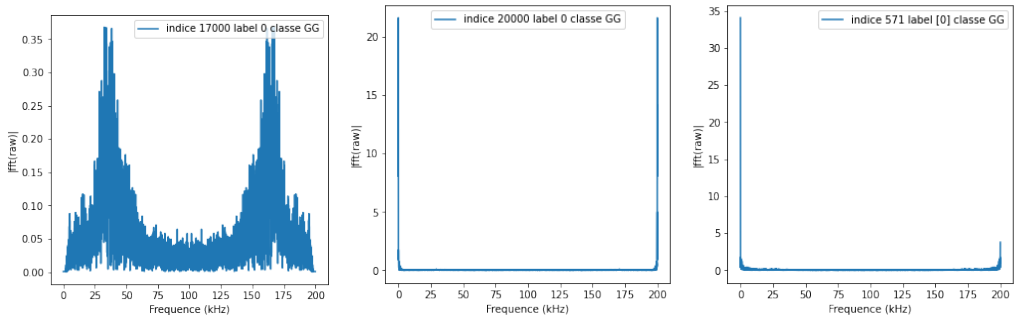


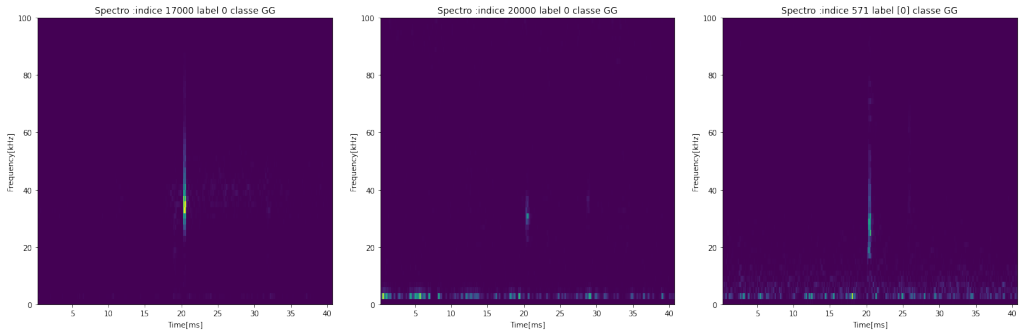
FIGURE 2.3 – Transformé de Fourier des signaux : (a) n°17000 et (b) n°20000 de la base d'apprentissage ; (c) n°571 de la base de test.

Sur la figure 2.3 on constate particuliérement sur le signal n° 17000 un pic entre les fréquence 25KHz et 50KHz, dont on peut supposer qu'il correspond au clic. Par contre, les autres spectres, tels qu'ils sont représentés, avec des pics importants aux très basses fréquences, ne permettent pas de trouver le clic.

2.1.4 Spectrogrammes

Les spectrogrammes sont simplement des transformées de Fourier effectuées à chaque pas de temps, sur une fenêtre temporelle. Cela permet de visualiser l'évolution des fréquences au cours du temps.

On commence par observer, sur la figure 2.4, les spectrogrammes en 2D, avec l'amplitude qui est représentée par une échelle de couleur (a noter qu'ici pour éviter les phénoménes d'écrasement, les fréquences majoritaires pouvant écraser les autres, on utilise une échelle logarithmique pour les amplitudes).



(a) (b) (c)

FIGURE 2.4 – Spectrogramme 2D des signaux : (a) n°17000 et (b) n°20000 de la base d'apprentissage ; (c) n°571 de la base de test.

On constate tout d'abord que leurs gammes de fréquences semblent relativement variable d'un enregistrement à l'autre et que la visibilité des clics semble bien correlée à leur qualité. Ainsi, pour le signal n° 17000 qui est de bonne qualité, le clic est très clairement visible : la bande verticale au centre de la figure de couleur vive. Pour le signal n° 571 qui est peu dégradé le clic est encore assez visible. A l'inverse, pour le signal n° 20000 qui est le plus dégradé, le clic est très peu visibleet plutôt sur les basses fréquences.

On constate également, ce qui n'apparaisait ni sur les signaux temporels ni sur les spectres que le clic principale sempble accompagné de répliques, ce qui est cohérent avec la biologie : il y a un phénomène d'écho à l'intérieur du nez des odontocètes lors de l'émission des clics.

On peut aussi visualiser les spectrogrammes en 3D, comme le montre la figure 2.5. Les informations sont les mêmes, mais représentées différemment. Cela permet d'avoir une meilleure idée de l'importance relative des pics de fréquence. De plus, un intérêt qui n'est pas évident sur un affichage statique, cette visualisation en 3D permet de faire varier l'angle de vue, ce qui permet d'avoir des points de vue différents sur les courbes.

2.2 Data augmentation

2.2.1 Intérêt théorique

Basiquement, l'augmentation de données, ou data augmentation, regroupe un ensemble de méthodes permettant d'augmenter "artificiellement" la taille de la base sur laquelle le réseau de neurones va apprendre. Ainsi en plus de nos exemples initiaux on viendra rajouter de nouveaux exemples qui seront des versions "modifiées" des exemples initiaux.

Pour cela selon la nature des données de cette base on va par exemple :

- Flouter les exemples s'il s'agit d'images
- Effectuer une rotation sur les exemples
- Modifier la luminosité dans le cas d'images

Dans notre cas, dans la mesure où nos données sont des enregistrements audios, nous allons plutôt :

- Rajouter du bruit sur les exemples
- Déplacer le clic
- Simuler une modification de la distance entre l'animal et l'hydrophone

Nous avons décidé de ne pas inclure l'effet Doppler (modification des fréquence en fonction des vitesses relatives) dans la data augmentation, qui semble être marginal.

L'intérêt le plus évident de cette opération est de multiplier le nombre d'exemples disponibles afin d'éviter le surapprentissage mais elle peut avoir beaucoup plus d'utilité. En effet dans notre cas, nous n'avons rencontré aucun probléme de surapprentissage mais nous avons besoin de l'utiliser pour une autre raison. Nous avons pu remarquer que certains exemples avaient subi de fortes dégradations notamment dues à du bruit ou bien à un fort décalage temporel du clic par exemple. Afin d'éviter que ces dégradations n'altérent le processus d'apprentissage de nos réseaux de neurones (le réseau pouvant par exemple assimiler une de ces dégradations à l'une des classes), plutôt que de les supprimer, il nous a paru plus pertinent d'essayer de rajouter aléatoirement ces pertubations sur tous les exemples.

2.2.2 Rajout de bruit blanc

Comme on a pu l'observer précedement certains enregistrement sont plus ou moins bruités. Nous allons donc dans un premier temps essayer d'en diminuer l'impact via de la data augmentation. Autrement dit nous allons artificiellement créer des exemples issus d'enregistrements choisis aléatoirement auxquels du bruit a été artificiellement ajouté. Le type de bruit que nous avons choisi, en dehors de toute indications sur les bruits marins, est un bruit blanc ou gaussien : au signal temporel est ajouté une valeur aléatoire de densité normale (moyenne 0, écart-type variable).

2.2.3 Simulation de distance

Lors de la prise des enregistrements audio les animaux peuvent se situer plus ou moins loin du ou des micro, ce qui peut potentiellement influencer notre classifieur. En effet certaines espèces plus fuyardes peuvent par exemple rester systématiquement plus éloignées du micro que les autres poussant le classifeur à assimiler une grande distance à une espèce en particulier. Afin d'éviter ce biais nous avons décidé de rajouter la simulation de distance dans la data augmentation. Le principe est simple on va rajouter aléatoirement des signaux choisis dans des classes aléatoires à une distance aléatoire (on simule les effets de la distance sur le signal). L'effet de la distance dans un milieu liquide est approximé une atténuation exponentielle des hautes fréquence. Pour simuler cette atténuation, nous nous sommes basés sur des études en hydro-accoustique.

2.3 Pré-traitement du signal

Comme nous avons pu le voir précedement il arrive que certains enregistrements aient subi d'importantes dégradations, si dans un premier temps nous avons fait de la data augmentation il pouvait y avoir certains enregistrements pour lesquels cela ne suffit pas. Parce qu'ils seraient trop dégradés ils empêcheraient l'identification de l'espèce, cela peut-être un bruit tellement important qu'il recouvrirait le clic, comme nous avons pu le constater sur la figure 2.1. Ainsi nous avons

mis en place un pré-traitement des signaux. Ces pré-traitements sont appliqués de la même façon et indifféremment sur l'ensemble de signaux. Ils ne remplacent pas la data augmentation mais viennent simplement en complément de celle-ci

2.3.1 Filtre passe haut

Dans un premier temps pour résoudre les probléme vus précedement on va commencer par appliquer un filtre passe haut aux enregistrements. En effet les clics des différents animaux se situant dans une gamme de fréquence au dessus des 8 KHz on peut donc appliquer le filtre à l'enregistrement sans en altérer le clic.

Sur la réponse fréquencielle du filtre 2.6 on observe bien qu'au niveau de la fréquence de coupure qui est de 8kHz le signal est réduit de moitié.

Ainsi en appliquant simplement ce filtre on constate par exemple que l'identification du clic pour le zoom qui était impossible sans le filtre (à gauche sur 2.7) sur le signal n° 20000 devient parfaitement possible avec (à droite sur 2.7).

2.3.2 Mise à l'échelle

En analysant les signaux, nous avons constaté de grandes différences aussi bien en terme d'intensité que de fréquences sur leurs clics. Cependant, pour bien fonctionner le réseau de neurones convolutifs (comme la grande majorité des classifieurs) nécessite des données normalisées. C'est pourquoi on va devoir mettre à l'échelle nos signaux, autrement dit nous allons normaliser automatiquement entre -1 et 1 l'ensemble de nos signaux. Afin de réaliser celà on fait un choix pragmatique qui était de repérer le maximum puis diviser le signal par ce maximum. Cette phase de normalisation se faisant après le filtrage passe haut des signaux, il est raisonnable d'espèrer que la maximum du signal corresponde bien au clic.

2.4 Les Pipelines

A l'image des pipelines utilisés pour transporter le gaz ou le pétrole, les pipelines en informatique servent à transporter un flux de données. Flux de données sur lequel on va effectuer un certain nombre d'opérations, flux qui sera ensuite injecté dans le réseau de neurones. Cette méthode constitue maintenant la norme pour les applications en apprentissage automatique.

Elle présente plusieurs interêts majeurs :

- Elle évite d'avoir à stocker l'ensemble des résultats des opérations intermédiaires, permettant ainsi d'économiser beaucoup de mémoire.
- Elle nous permet d'optimiser grandement l'ensemble du processus de prétraitement des données.
- Elle favorise la réutilisation des procédures de traitement des données, par la normalisation dans leur définition.
- Elle améliore grandement les performances lors de l'utilisation de frameworks disposant de fonctions adpatées pour les pipelines, comme TensorFlow, notamment pour l'implentation des traitements sur les cartes graphiques.

En pratique l'ensemble de nos fonctions étaient stockées dans un fichier python nommé cachalot_helper, et à chaque essai on faisait passer notre flux de données par les fonctions désirées avant de l'injecter dans le réseau de neurones.

2.5 Les PDF (Fiches d'analyse)

2.5.1 Génération des PDF

Les bases de données contiennent un très grand nombre d'exemples (environ 130 000 au total que l'on peut visualiser sous 12 formes différentes soit potentiellement pls d'un million d'images). Afin de pouvoir exploiter les analyses faites précédemment, il a fallu développer un certain nombre d'outils afin de pouvoir aisement trier et manipuler les données. Pour cela je me suis inspiré du systéme de pipeline que nous venons de voir, non pas pour effectuer une tâche de classification, mais pour générer automatiquement des fiches d'analyse des signaux. L'avantage de cette méthode est que les fonctions de traitement utilisées pour l'apprentissage et pour la visualisation sont les mêmes, réduisant ainsi le risque d'erreurs lié aux doublons.

J'ai donc créé dans un script python définissant fonction paramètrable permettant tout d'abord de sélectionner un ou un plusieurs signaux dans un certain nombre de classes ou des signaux bien specifiques puis de générer automatiquement avec et ou sans preprocessing (les traitements du signal) avec ou sans zoom :

- Des courbes des signaux sélectionnés
- Des spectrogrammes 2D des signaux sélectionnés
- Des spectrogrammes 3D des signaux sélectionnés

Une fois générés, ils sont enregistrés sous forme de png dont le nom correspond à leur description. Ce qui donne par exemple pour le spectrogramme 3D sans processing et sans zoom de l'enregistrement numéro 17 000 de la classe GG dont le label est 0 :
indice17000Spectro3Dlabel0classeGGsansprocessingsanszoom.png.

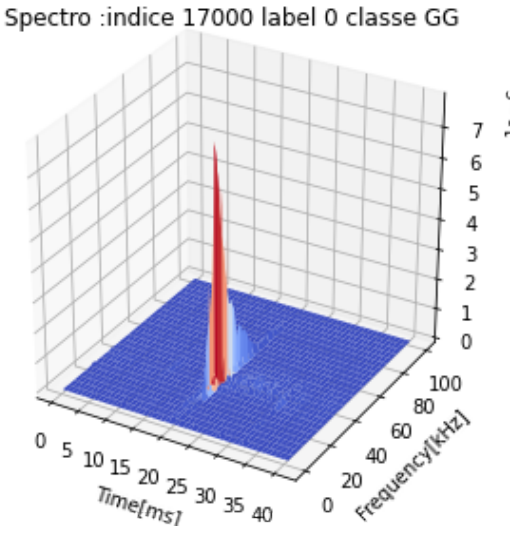
A noter que les plots simples sont enregistrés sous spectro1D pour des raisons pratiques.

Dans un troisième temps, j'ai créé un autre script en python également paramètrable permettant de sélectionner des png en fonction de leur label, de leur type (spectrogramme 1D ou 2D ou 3D) et leurs options (avec ou sans zoom et avec ou sans processing). Et de les intégrer dans un ou plusieurs fichiers latex en fonction de leur nombre.

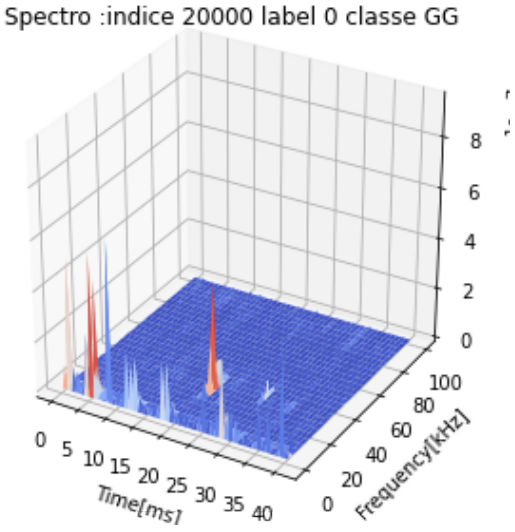
Dans un quatrième temps, j'ai créé un autre script python, encore une fois paramètrable qui va récupérer les fichiers latex précedement créés, puis va les réunir dans un seul fichier latex. Ce fichier latex est automatiquementcompilé par le script pour générer un fichier pdf qui contient l'ensemble des courbes générées. Ce fichier pdf est alors enregistré avec un nom correspondant à ce qu'il contient. Ainsi un fichier contenant les spectrogrammes 2D du label 6 sans processing et avec zoom sera nommé :
Spectro2Dlabel6sansprocessingaveczoom.pdf

Enfin un programme principal nommé apdfmaker, également paramètrable, est chargé de coordonner l'ensemble des scripts vus précedement. A noter que ce programme ne se contente pas de générer un pdf à la fois mais peut en générer une multitude à chaque exécution, en fonction des paramètres. Ainsi si l'on veut par exemple qu'il génére toutes les représentations graphiques possibles (soit 1 080 000 images) de tous les enregistrements puis qu'il les stocke dans des pdf les plus détaillés possibles c'est théoriquement possible (même si cela n'est pas souhaitable car très vite l'espace disque serait saturé).

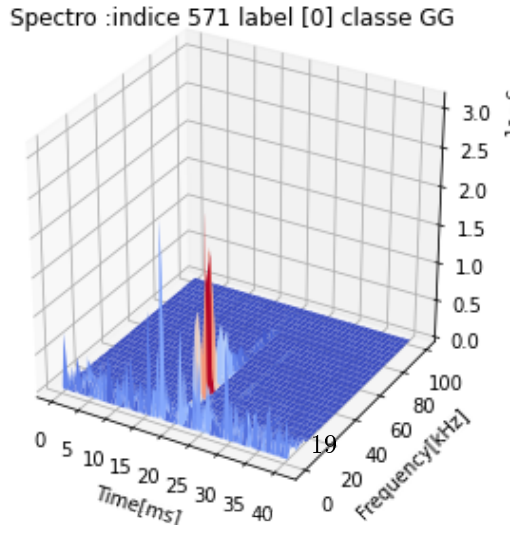
Les pdf finaux ressemblerons à la figure 2.8.



(a)



(b)



(c)

FIGURE 2.5 – Spectrogramme 3D des signaux : (a) n°17000 et (b) n°20000 de la base d'apprentissage ; (c) n°571 de la base de test.

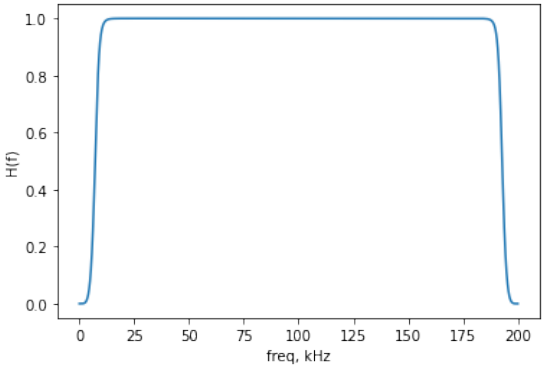
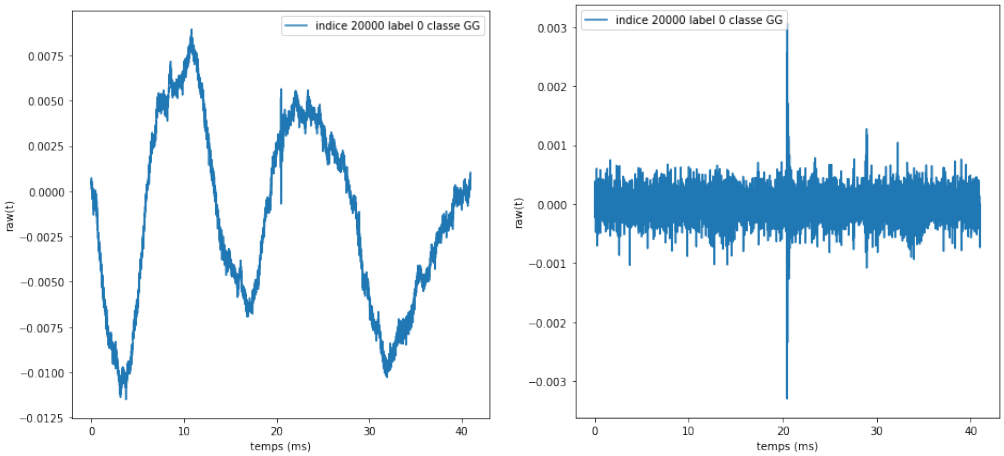


FIGURE 2.6 – Réponse fréquencielle du passe haut à 8kHz



(a) (b)

FIGURE 2.7 – Spectrogramme 3D des signaux : (a) n°20000 de la base d'apprentissage non filtré et (b) n°20000 de la base d'apprentissage filtré

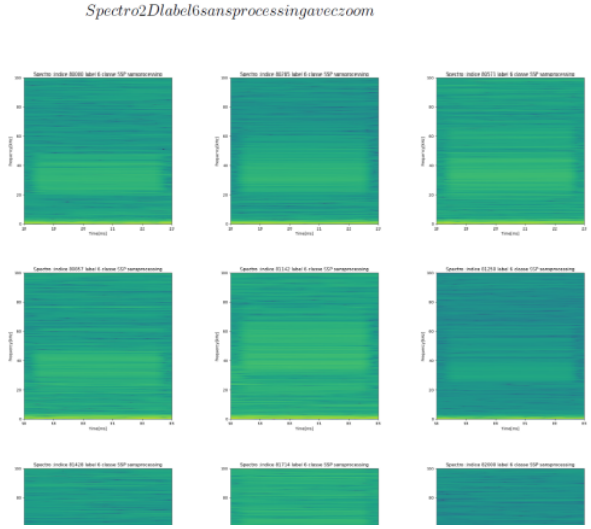


FIGURE 2.8 – Réponse fréquencielle du passe haut à 8kHz

Chapitre 3

Le travail à distance

Dans cette partie nous verrons dans un premier temps les méthodes qui nous on permis de pouvoir continuer a travailler à distance. Puis dans un second temps nous verrons les outils que nous avons utilisé pour celà ainsi que leur interet. A savoir GitHub pour le stockage des programmes et de la documentation. Puis collab qui nous grâce auquel nous avons pu coder en ligne. Et logiquement on enchainera avec tensorflow qui fonctionne particuliérement bien avec colab. Puis on finira avec LaTex qui a été l'un de mes principaux outils de travail.

3.1 Organisation du travail à distance

Comme vous le savez certainement durant cette année 2020 nous avons été touché par la crise du coronavirus qui nous à conduit à être confinés nous forçant à travailler uniquement à distance.

Ces circonstances très particuliéres ont grandement affecté notre travail particulérement au début où nous avons dû régler de nombreux problémes techniques et organisationels. Cependant en nous forçant à nous adapter à ces nouvelles conditions, cette crise nous a permis de grandement augmenter nos competances en "télétravail".

Ainsi malgré des débuts léthargiques nous avons mis en place une "routine de travail" qui était la suivante : -Des visio-conférences quotidiennes nous permettant d'organiser et de synchroniser notre travail -Un groupe whatsap dédié à mon stage afin de communiquer le plus efficacement possible -Un Github privé dédié afin de partager l'ensemble du projet -Un partage régulier de google collab via google drive

3.2 Outils utilisés

3.2.1 Présentation de GitHub

Nous pouvons définir GitHub comme une plateforme de développement de projet in formatique en groupe. Elle s'implifie grandement le développement de projets. Elle permet de versioner

ses programme et d'y apporter des modification en temps réel à plusieurs.

Pourquoi Github

Car celà permet une certaine synergie avec nos autres outils que nous verrons plus tard. Cette plateforme permet une facilité de développement de par sa fonctionnalité de versionnage de notre code à chaque changement ce qui permet une mise à jour dynamique ainsi qu'une relative facilité à retourner à un état antérieur de notre programme ce qui permet une facilité de débogage.Nous pouvons d'ailleurs dire que ce rappport est entreposer sur Github et qu'il peut-être récupérer facilement.Cette plateforme est aussi très connue dans le monde de la programmation ce qui sera utile pour notre futur professionnel.

3.2.2 Présentation de Google Colab

Colab peut-être défini comme étant une plateforme d'éxécution pour notre code il permet du fait que ce soit la puissance de calcul d'ordinateur géré par Google une vitesse d'exécution ainsi qu'une vitesse de téléchargement de base de données supérieure à celle qui nous est disponible en local.

Pourquoi Colab

En premier lieu pour faciliter l'exécution du code car elle ne se fais pas en local ce qui permet une exécution quasi immédiate du code sans aucune installation. Il est aussi facile de mettre sur github du code produit avec colab car ces deux plateformes sont liées. Il permet de par l'utilisation du format Jupyter notebook de mélanger code et texte (peu aussi comporter des images) dans notre notebook.

Voilà un exmple d'exécution avec colab :

3.2.3 TensorFlow

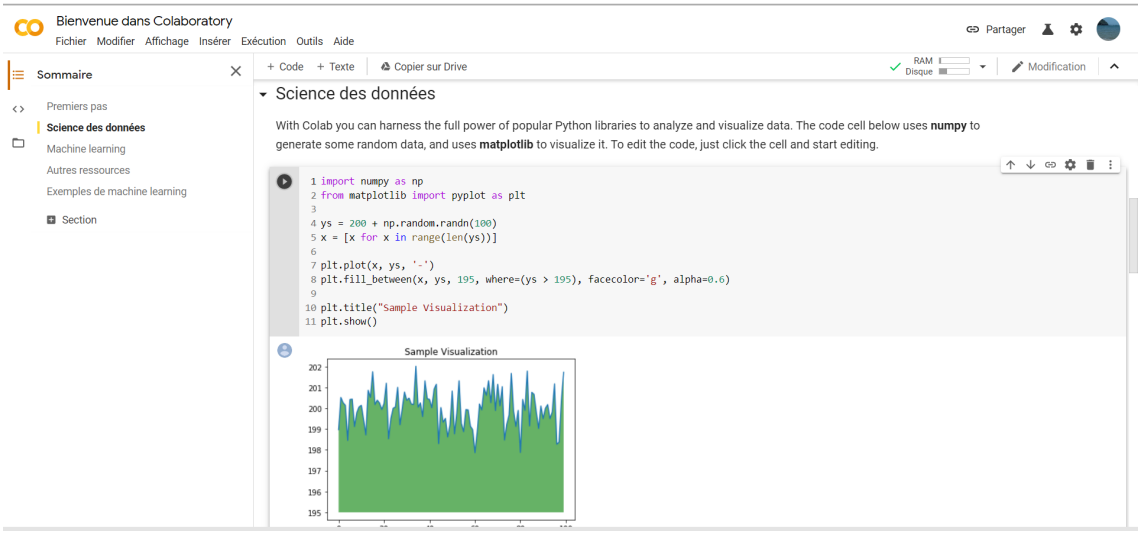


FIGURE 3.1 – Nous avons ici un exemple de code exécuté avec colab.

FIGURE 3.2 – Logo de Tensorflow

TensorFlow est un framework open source, multiplateforme d'apprentissage automatique développée par Google Brain, en C++ avec une interface pour Python. Sa premiere version est publiée par Google le 9 novembre 2015. L'intérêt de ce framework est de limiter les coûts de développement des solutions à base de réseaux de neurones, en réunissant des fonctionnalités permettant, en quelques lignes de code de construire des réseaux complexes. Tensorflow permet donc de simplifier beaucoup de choses dans le domaine de l'apprentissage automatique. Son autre intérêt est de fournir des interfaces pour pouvoir exécuter les calculs sur des accélérateurs graphiques et surtout de les prendre en charge de façon complétement transparente pour les utilisateurs. Les gains de vitesse peuvent atteindre ainsi un facteur 10 quand le même modèle est exécuté sur une carte graphique.

Le concurrent principal de TensorFlow est pyTorch, utilisé par FaceBook.

3.2.4 Présentation de LaTex

LaTeX

Nous pouvons dire que LaTex est un langage de traitement de texte tel que le markdown qui permet de mettre en forme notre texte de manière ščientifiqueëcela veut dire que. LaTex permet une facilité d'écriture des équations et de toutes les écritures mathématiques. Ce langage permet de par ses nombreux packages une quasi-infinité de possibilités. Il permet également de générer

des fichiers pdf facilement et de manière automatique. Raison qui m'as logiquement poussé à choisir ce langage afin de générer mes pdf.

Chapitre 4

Conclusion

4.1 Conclusion de l'analyse

L'analyse des fiches nous à permis de constater que :

- Certaines espèces semblent émettre à certaines gammes de fréquences
- Contrairement à ce qui nous avait été annoncé dans la description du challenge de nombreux clics de la base de test n'étaient pas centrés.
- Les signaux de la base d'apprentissage sont en très grande majorité bien centrés.
- Un nombre important d'enregistrements de la base de test semblent avoir subbis d'importantes dégradations.
- Sur certains enregistrements de la base de test le clic est impossible a identifier sans prétraitement.

Ce qui peux expliquer au moins en partie le différentiel de performances de nos IA sur la base labelisée et la base non labelisée.

En somme l'objectif est atteint.

4.2 Perspectives

Actuellement l'ensemble des IA existantes donnant de bons résultats ont un point commun, elle demandent une quantité titanesque de données afin d'être efficace. Fort de ce constat les outils d'analyse et de tris des données permettant de naviguer facilement dans d'importantes quantités de données semble présenter un grand intérêt. Ainsi même si pour l'instant ces outils sont adaptés à un probléme et une base de données particulière, la transformation de ceux-ci en outils génériques utilisables sur diverses bases de données ne semble pas absurde.

4.3 Les apports du stage

4.3.1 les apports generaux

Comme dit précedement si à l'heure actuelle ces outils ne sont adaptés qu'à une base de données et à un probléme particulier il est possible qu'ils puissent par la suite être transformés en outils génériques facilitant l'analyse des données des futurs problémes auxquels seront confrontés l'équipe SpikeTrain.

4.3.2 les apports personels

Bien que les tâches auxquelles j'ai été affecté ne s'attardent que peu sur les IA que nous avons utilisés, de par leur nature elle m'ont permis d'acquérir une large gamme de compétences. D'abord en programmation puisqu'avant de commencer mon stage je n'avais qu'un faible niveau en python et de maigres connaissances sur google colab, ainsi qu'aucune connaissance en LaTex, je ressors donc avec une certaine maitrise de ces trois langages. Ensuite en terme de méthodologie, mon travail avec des chercheurs sur un sujet de recherche à distance m'as permis de m'initier et de m'exercer aux méthodes de recherche, de travail à distance. J'ai pu aussi, apprendre à rédiger des documents de recherche grâce à la création des fiches d'analyse ainsi que la rédaction de ce rapport. Les problémes techniques inhérents au conditions exceptionnelles dans lesquelles mon stage s'est déroulé ont rendu son déroulement compliqué et laborieux. Heureusement, la grande maîtrise et la remarquable fléxibilité de mes tuteurs en ont fait une expérience aussi intense qu'enrichissante.