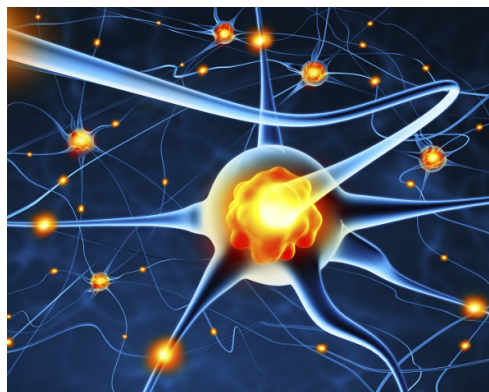


RAPPORT DE STAGE

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 figures

1.1	Figure 1 Schéma de l'organisation interne du LAMIA : ¹ Apprentissages Interactions Données. ² Data analytics and big data gathering with sensors. ³ Mathématiques (analyse variationnelle, analyse numérique, EDP, analyse statistique, mathématiques discrètes). [Pleaseinsertintopreamble]Membres permanents : Suzy Gaucher-Casalis (MCF), Engueran Grandchamp (MCF–HDR), Jean-Luc Henry (MCF), Jimmy Nagau (MCF), Vincent Pagé(MCF), Helene Paugam Moisy (PR), Sébastien Régis (MCF), Céline Rémi (MCF). .	4
1.2	Fonctionnement d'un neurone seul	6
1.3	réseau de neurones classique en couches	7
1.4	Schéma de la convolution.	8
1.5	Logo de Tensorflow	8
2.1	Signaux 17000, 20000 et 571 bruts	10
2.2	Les signaux zoom	11
2.3	Transformé de Fourier des signaux 17000, 20000 et 571	11
2.4	Spectrogramme 2D des signaux	12
2.5	Spectrogramme 3D du signal 17000	12
2.6	Spectrogramme 3D du signal 20000	12
2.7	Spectrogramme 3D du signal 571	12
3.1	Nous avons ici un exemple de code exécuté avec colab.	16
4.1	Architecture scale-free	17

Table des matières

1	Introduction	3
1.1	Présentation de la structure d'accueil	3
1.1.1	L'Université des Antilles	3
1.1.2	Le laboratoire LAMIA	3
1.1.3	Le groupe Spiketrain	5
1.2	Contexte du stage	5
1.2.1	Le Challenge : Classifier les Cachalots	5
1.2.2	Etat des lieux a mon arrivé	5
1.2.3	Réseau de neurones classique	6
1.2.4	Réseau de neurones convolutifs	7
2	Analyse et traitement des données	10
2.1	Les signaux	10
2.1.1	Signaux Bruts	10
2.1.2	Le zoom	10
2.1.3	Transformé de Fourier	11
2.1.4	Spectrogrammes	11
2.2	Data augmentation	11
2.2.1	Intérêt théorique	11
2.2.2	Rajout de bruit blanc	13
2.2.3	Simulation de distance	13
2.3	Traitement du signal	13
2.3.1	Filtre passe haut	13
2.3.2	Mise à l'échelle	13
2.4	Les Pipelines	13
2.5	Les PDF (Fiches d'analyse)	13
3	Le travail à distance	15
3.1	Organisation du travail à distance	15
3.2	Outils utilisés	15
3.2.1	Présentation de GitHub	15
3.2.2	Présentation de Google Colab	15
3.2.3	Présentation de LaTeX	16
4	Conclusion	17
4.1	Conclusion de l'étude	17
4.2	Perspectives	17
4.2.1	18
4.3	Les apports du stage	18
4.3.1	les apports generaux	18
4.3.2	les apports personnels	18

Chapitre 1

Introduction

1.1 Présentation de la structure d'accueil

Durant la période de mon stage , j'ai été accueilli au **Laboratoire de Mathématiques Informatique et Application (LAMIA)** de l'Université des Antilles (UA).

Pour présenter cette structure, il me faut tout d'abord présenter l'Université à laquelle il est rattaché.

1.1.1 L'Université des Antilles

Voici quelques chiffres que je ne connaissais pas et qui donnent la mesure de sa taille :

L'Université des Antilles s'organise autour deux pôles universitaires régionaux autonomes : le « Pôle Guadeloupe » et le « Pôle Martinique ».

Sur ces pôles, l'Université assure des missions d'*enseignement* et de *recherche*, assistées par des *administratifs et des techniciens*.

Administration et personnel technique

L'UA emploie 414 Administratifs et Techniciens (environ 200 personnes pour l'administration 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 : - autour de 20 enseignants-chercheurs - autour de 120 étudiants

Recherche

La recherche est structurée en laboratoires auxquels sont rattachés les enseignants chercheurs qui peuvent 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, ATER, 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 Données ;

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

L'équipe avec laquelle j'ai principalement travaillé est celle d'**Apprentissages Interactions Données** qui développe des méthodes de traitements et d'analyse de données hétérogènes : images (classique, multi-spectrale), séquences vidéos, séries temporelles et spatio-temporelles, dont la responsable est **Mme. Hélène Paugam-Moisy**.

Indépendamment de ces équipes, 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 projet est nommé de façon informelle projet "**Spikes**" et concerne l'utilisation de **réseaux de neurones impulsionnels** pour l'apprentissage automatique (ces notions seront définies plus loin). Le groupe de travail associé réunit à l'heure actuelle :

- 1 Professeur des Universités
- 2 MCF avec HDR
- 3 MCF
- 1 ingénieur d'études.

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

Ci dessous, un schéma présentant la structure du laboratoire me rattachement à cette structure. (L'équipe de travail **Spikestrain** étant informelle, elle ne figure pas sur ce schéma.)

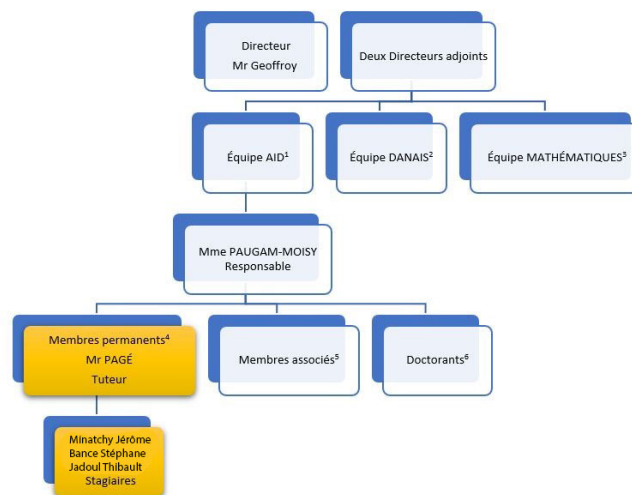


FIGURE 1.1 – Figure 1 Schéma de l'organisation interne du LAMIA : ¹Apprentissages Interactions Données. ²Data analytics and big data gathering with sensors. ³Mathématiques (analyse variationnelle, analyse numérique, EDP, analyse statistique, mathématiques discrètes). ⁴Membres permanents : Suzy Gaucher-Casalis (MCF), Enguerran Grandchamp (MCF–HDR), Jean-Luc Henry (MCF), Jimmy Nagau (MCF), Vincent Pagé(MCF), Helene Paugam Moisy (PR), Sébastien Régis (MCF), Céline Rémi (MCF).

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

Le groupe **Spiketrain** 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 (échecs, Go).

1.2 Contexte du stage

1.2.1 Le Challenge : Classifier les Cachalots

Le challenge "Dyini Odontocete Click Classification, 10 species [DOCC10] by Universite de Toulon" (<https://challengedata.ens.fr/challenges/32>) consistait simplement à réaliser un classifieur qui classe des mammifères marins en une dizaine d'espèces à partir de leurs "Clics" pour cela nous avions à disposition une base d'apprentissage labélisée ainsi qu'une base de test non labélisée sur laquelle nous pouvions évaluer les performances réelles de nos classifieurs. Pour cela nous labélisions les exemples de la base non labélisée puis nous envoyions nos prédictions sur le site du challenge qui nous renvoyait nos performances ainsi qu'un classement des performances de tous les participants.

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 200Hz. Dans le cas de la base non labélisée appelée base de test chaque enregistrement contient normalement un clic centré au milieu de la fenêtre tandis que dans la base labélisée appelée base d'apprentissage le clic n'est pas spécialement centré et peut se situer à divers moments de l'enregistrement de plus ceux-ci 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 précision est de l'ordre du mètre.

1.2.2 Etat des lieux à mon arrivé

Quand j'ai commencé mon stage mes tuteurs avaient déjà commencé le challenge depuis un moment déjà c'est pourquoi j'ai 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 problème uniquement basé sur du machine learning notamment des réseaux de neurones et des réseaux de neurones convolutifs
- De la Data Augmentation
- Divers traitements du signal

Leurs meilleurs résultats étaient de l'ordre de 95 pourcents de réussite sur la base labélisée contre 72 pourcents de réussite sur la base non labélisée, soit un écart de 20 pourcents de réussite entre les deux bases qui 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 labélisée comme des données de meilleure qualité que celles de la base labélisée.

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 grandement l'analyse des données qui nous ont été fournies lors de ce challenge.

Mais avant de pouvoir commencer cette partie du travail il a fallu commencer par comprendre un certain nombre d'outils et de concepts que nous allons voir ensemble.

Neurone artificiel

Tout d'abord avant de parler de réseaux de neurones il faut expliquer le principe du 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 laquelle vient s'ajouter un **biais** spécifique à chaque neurone (cf. equation 1.1).

$$y = \sum_i^n w_i \times x_i + b \quad (1.1)$$

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

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

Un schéma reprenant ces explications est présenté dans la figure ?? :

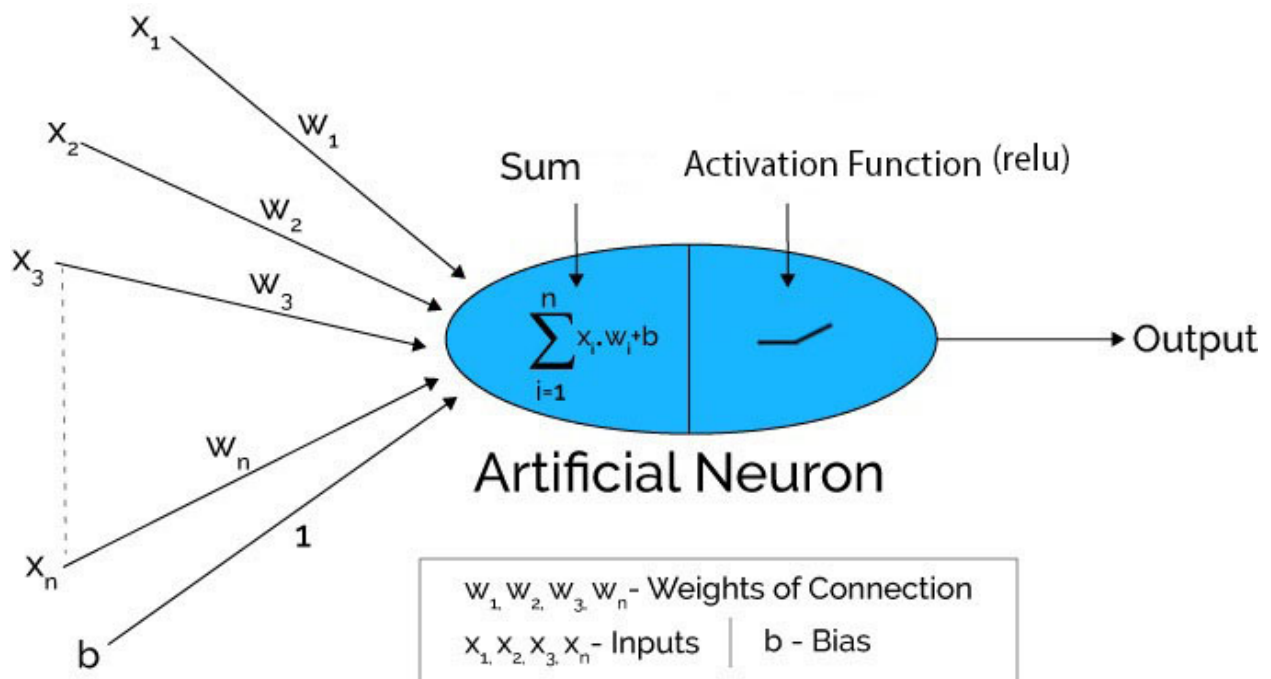


FIGURE 1.2 – Fonctionnement d'un neurone seul

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

1.2.3 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 réseaux, les neurones sont organisés en **couches**. 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 que nous pouvons qualifier de *full connected*, c'est à dire que chaque neurone d'une couche est connecté avec chaque neurone de la couche suivante. Nous allons encore une fois, pour le bien de ce rapport, ne pas épiloguer sur les autres types de connexions existantes.

Nous allons, pour cette partie encore, utiliser une figure (1.3) pour illustrer nos propos :

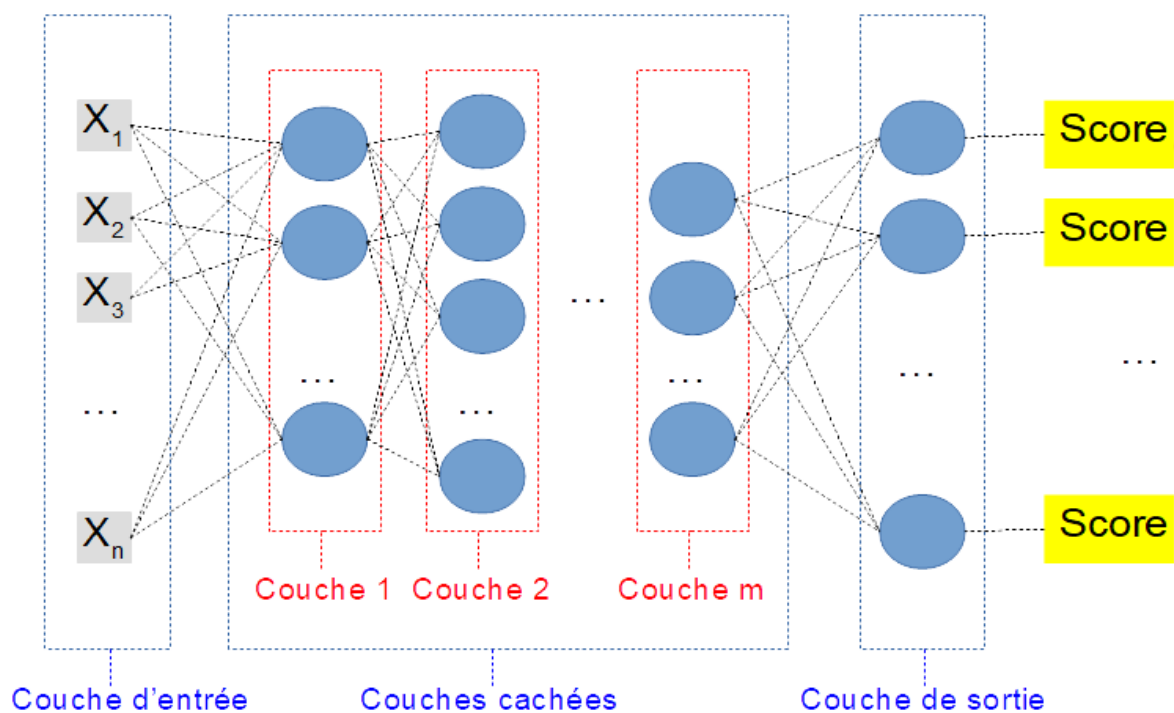


FIGURE 1.3 – réseau de neurones classique en couches

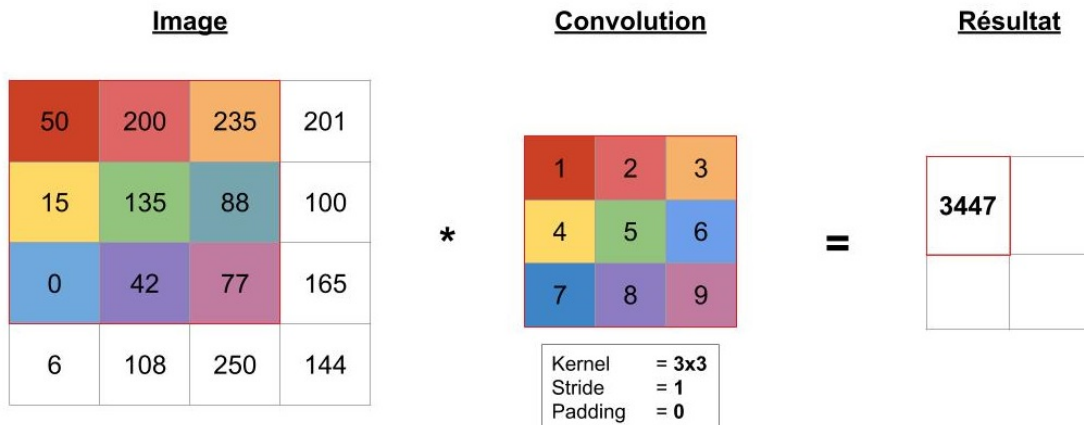
1.2.4 Réseau de neurones convolutifs

le réseau de neurones convolutifs est un type de réseau de neurones inspiré par le cortex cérébral des animaux. Il possède de larges applications dans la reconnaissance d'image, de vidéo et de son.

Ce réseau se présente comme un réseau classique, une couche d'entrée, une couche de sortie et des couches cachées. Ces couches cachées possèdent des couches **convolutives** pour lesquelles chaque neurone va appliquer un filtre convolutif sur une partie de l'image en entrée afin d'analyser toute l'image (voir figure 1.4).

Ces neurones ont souvent une fonction d'activation *relu* qui borne l'activation du neurone à 0. Ce sont donc des neurones dont l'activation ne peut être que positive.

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).



Pour calculer, on fait :
pixel1 de l'image x pixel1 de la convolution + pixel2 de l'image x pixel2 de la convolution + ...

Ici, cela donne : $50 \cdot 1 + 200 \cdot 2 + 235 \cdot 3 + 15 \cdot 4 + 135 \cdot 5 + 88 \cdot 6 + 0 \cdot 7 + 42 \cdot 8 + 77 \cdot 9 = 3447$

FIGURE 1.4 – L'image d'entrée est découpée en *patches* sur lesquels est appliqué un filtre de convolution. La sortie de ce filtre appliqué à chacun des patches donne la couche de sortie de la couche convolutive (appelée carte ou map). La convolution est paramétrée par : le *kernel* (la forme du patch), le *stride* (le déplacement du filtre) et le *padding* (la façon dont les bords de l'image vont être traités).

L'intérêt majeur 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.

TensorFlow



FIGURE 1.5 – Logo de Tensorflow

TensorFlow est un framework open source, multiplateforme d'apprentissage automatique développée par Google Brain, en Python et en C++. Sa première 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 complètement transparente pour les utilisateurs. Les gains de vitesse peuvent atteindre ainsi un facteur 10 quand le

même modèle est exécuté sur une carte graphique.
Présentation du plan du rapport

Chapitre 2

Analyse et traitement des données

2.1 Les signaux

Afin d'améliorer la lisibilité des chapitres suivants nous prendrons 3 signaux (le n°17000 et le n°20000 de la base labélisée ainsi que le n°571 de la base non labélisée) que nous observerons sous diverses formes puis sur lesquels nous effectuerons un certain nombre de traitements.

2.1.1 Signaux Bruts

Dans un premier temps on commence par observer les signaux sans traitement sous différentes formes. D'abord de manière brute :

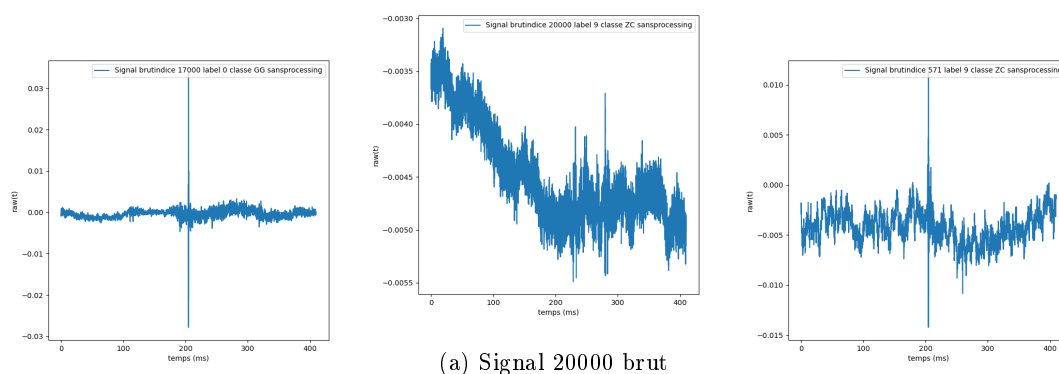


FIGURE 2.1 – Signaux 17000, 20000 et 571 bruts

On constate 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 on peut tirer quelques observations : -Le clic semble durer 5 millisecondes -L'amplitude des clics semble variable -Il arrive que le bruit soit parfois suffisamment important par rapport au clic pour rendre son identification complexe voire impossible. Pour affiner notre analyse il paraît pertinent de commencer par zoomer sur ce clic. Pour cela il va donc falloir commencer par trouver un moyen d'isoler le clic.

2.1.2 Le zoom

Pour zoomer sur le clic on commence par appliquer un filtre notamment pour supprimer les bruits parasites (dont on parlera dans la partie traitement du signal) puis on identifie le maximum qui sera logiquement le milieu du clic puis on rajoute l'équivalent de la durée d'un clic avant et après ce maximum.

On constate plusieurs choses tout d'abord l'efficacité du zoom semble corréler à la qualité du signal de départ, de plus les "bons clics" semblent se situer aux alentours de 200 ms (ils sont donc bien centrés)

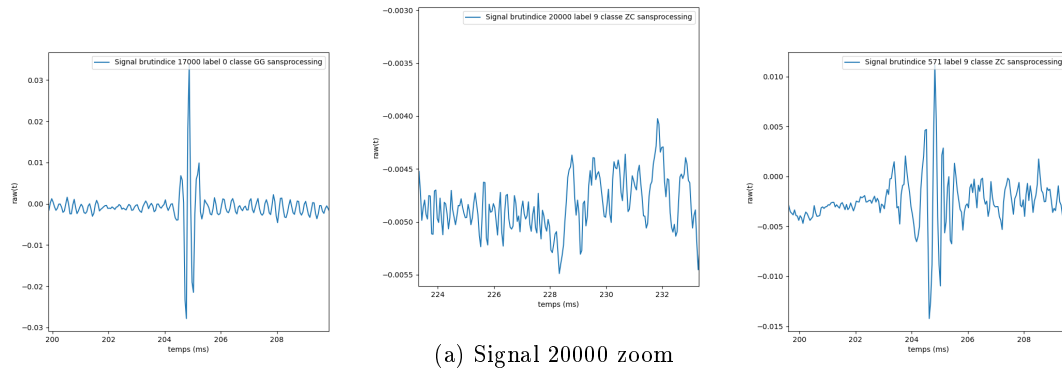


FIGURE 2.2 – Les signaux zoom

même si leur intensité est très variable pouvant aller jusqu'à un facteur 10 entre 2 clics. De plus sous cette forme nos observations semblent quand même limitées. On va donc commencer par 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 le plus pertinent.

2.1.3 Transformé de Fourier

domaine temporel -> fréquentiel Avant d'observer les spectrogrammes il convient de commencer par expliquer et observer les Transformés de Fourier de nos 3 signaux :

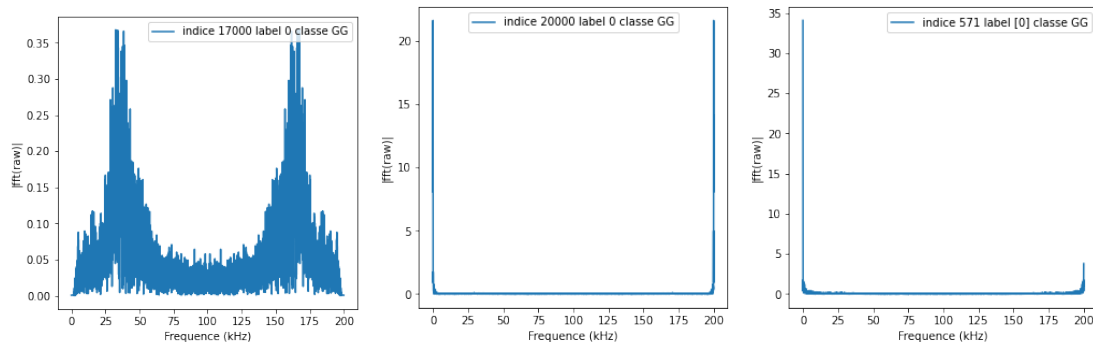


FIGURE 2.3 – Transformé de Fourier des signaux 17000, 20000 et 571

2.1.4 Spectrogrammes

On commence tout d'abord par observer leurs Spectrogrammes en 2D :
Puis les Spectrogrammes 3D :

2.2 Data augmentation

2.2.1 Intérêt théorique

Basiquement la Data augmentation regroupe un ensemble de méthodes permettant d'augmenter "artificiellement" la taille de la base sur laquelle notre ia va apprendre. Ainsi en plus de nos exemples initiaux on viendra rajouter de nouveaux exemples qui seront des versions "modifiées" des exemples initiaux.

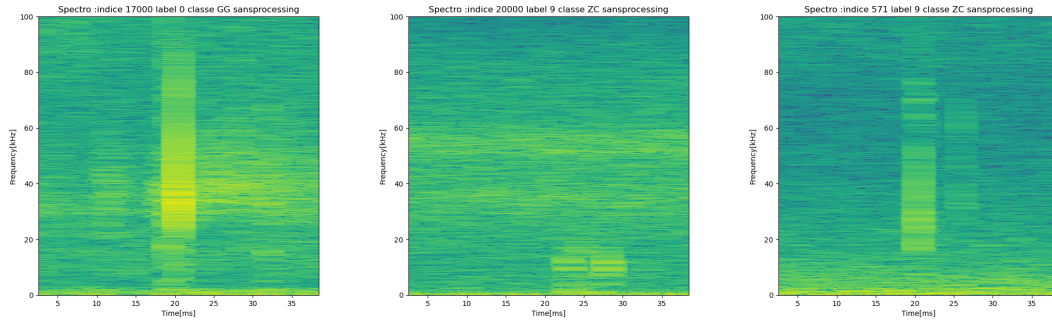


FIGURE 2.4 – Spectrogramme 2D des signaux

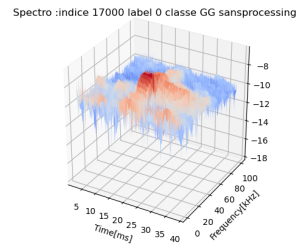


FIGURE 2.5 – Spectrogramme 3D du signal 17000

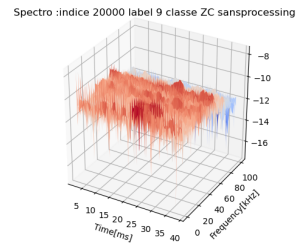


FIGURE 2.6 – Spectrogramme 3D du signal 20000

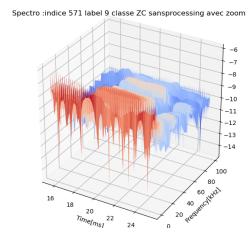


FIGURE 2.7 – Spectrogramme 3D du signal 571

Pour cela selon la nature des données de cette base on va par exemple : -Rajouter du bruit sur les exemples -Flouter les exemples s'il s'agit d'images -Effectuer une rotation sur les exemples -Modifier la luminosité dans le cas d'images -Déplacer le clic dans le cas de notre problème

L'intérêt le plus évident de cette opération est de simplement multiplier le nombre d'exemples dis-

Dans un troisième temps j'ai créer un autre python également paramétrable 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) puis ils sont stockés dans un ou plusieurs fichiers tex en fonction de leur nombre.

Dans un quatrième temps j'ai créer un autre python encore une fois paramétrable qui va récupérer les fichiers tex précédement créés puis les réunir dans un seul fichier tex qui va automatiquement s'exécuter pour générer un fichier pdf contenant l'ensemble des courbes qui étaient stockées dans les fichiers tex. Ce fichier pdf sera alors enregistré son nom correspondant à ce qu'il contient ainsi un fichier contenant les spectrogrammes 2D du label 6 sans processing et zoomer : sera nommer : Spectro2Dlabel6sansprocessingaveczoom.pdf

Et enfin un programme principal nommé apdfmaker également paramétrable chargé de faire tourner l'ensemble des programmes vus précédement afin de générer directement des fichiers pdf contenant ce que l'on désire. A noter que celui ci ne se contente pas de générer un pdf à la fois mais peut au contraire en générer une multitude à chaque run. Ainsi si l'on veut par exemple qu'il génère toutes les représentations graphiques possibles (soit 1 080 000 images) de tous les enregistrements puis qu'il les stock dans des pdf le plus détaillés possibles c'est parfaitement possible.

Chapitre 3

Le travail à distance

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 particulières ont grandement affecté notre travail particulièrement au début où nous avons dû régler de nombreux problèmes 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 whatsapp 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 versionner ses programme et d'y apporter des modification en temps réel à plusieurs.

Pourquoi Github

Car cela 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 rapport 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'exécution 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 fait 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 exemple d'exécution avec colab :

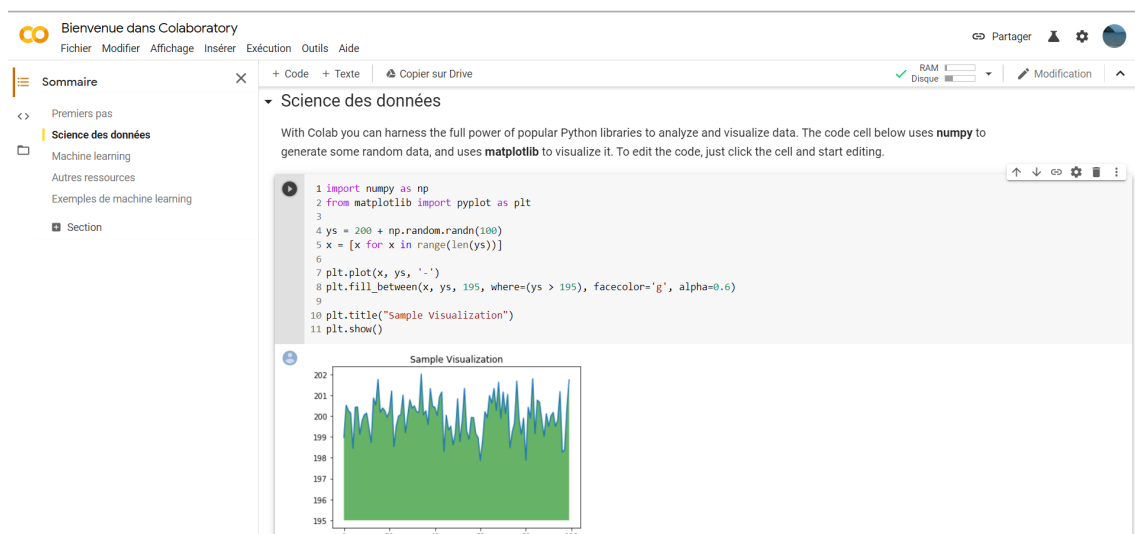


FIGURE 3.1 – Nous avons ici un exemple de code exécuté avec colab.

3.2.3 Présentation de LaTeX

L^AT_EX

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 scientifique cela 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.

Chapitre 4

Conclusion

4.1 Conclusion de l'étude

Nous avons tout d'abord vu avec notre modèle de simulation qu'il fallait tout d'abord limité l'influence de l'aléatoire de notre simulations

4.2 Perspectives

Nous avons envisagé de changer la **topologie**¹. Pour passer à une topologie dite **scalefree**. Qui est justement celle utilisé dans l'article d'origine. Cette architecture ressemble à ça :

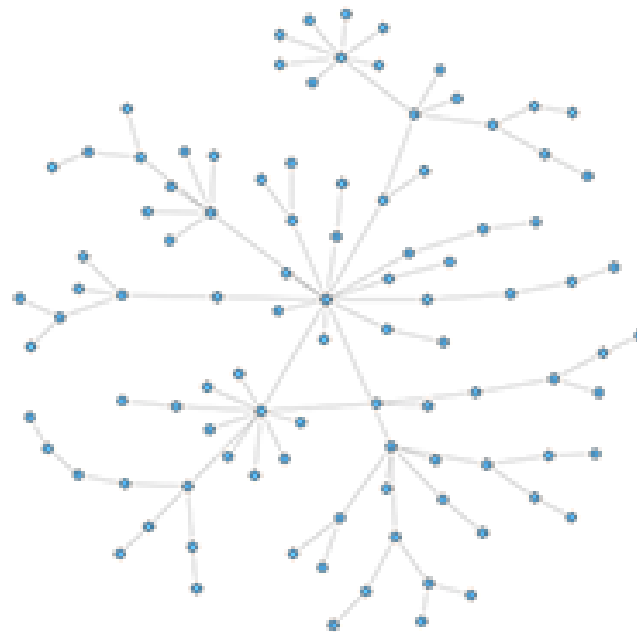


FIGURE 4.1 – Architecture scale-free

1. Topologie : Tel que la topologie d'un graffe il s'agit de la façon dont sont connecté nos neurones entre eux.

4.2.1

Nous pouvons interpréter tout le réseau comme un utilisateur et cet entretien comme le temps qu'il accorde à la rumeur.

Cela permettrait

4.3 Les apports du stage

4.3.1 les apports generaux

Grâce à ce stage les chercheurs du laboratoire auront une idée plus précise de la diffusion d'un echo dans un réservoir et comment y créer un entretien ce qui sera utile pour leurs futures expériences. Cela permettra de mieux appréhender certains problèmes.

4.3.2 les apports personels

Comme dit pendant mon introduction, avant ce stage je n'avais aucune connaissance des réseaux de neurones, ce stage m'a donc permis de m'ouvrir a ce nouveau sujet passionnant, et m'a permis d'obtenir des compétence nécessaire à mon cursus universitaire. Il m'a permis d'affiné mes méthodes de travail grâce à l'apprentissage de l'utilisation de GitHub et Colab. J'ai amélioré ma rédaction grace à l'approfondissement du LaTeX. Il m'a permis d'affiner mon analyse de par l'analyse de mes résultats.