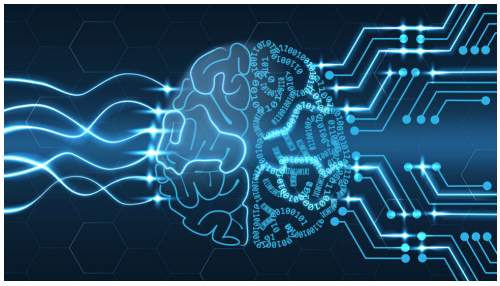
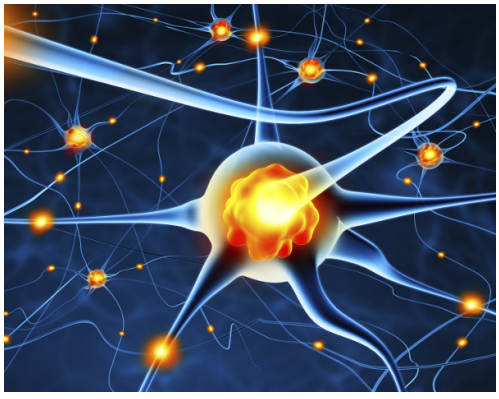


RAPPORT DE STAGE

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

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 . . . 4
  - 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 à mon arrivé . . . 5
  - 1.2.3 Réseau de neurones classique . . . 6
  - 1.2.4 Réseau de neurones convolutifs . . . 7
  - 1.2.5 TensorFlow . . . 9
- 1.3 Présentation du plan du rapport . . . 10

2 Analyse et traitement des données **11**
- 2.1 Les signaux . . . 11
  - 2.1.1 Les signaux bruts . . . 11
  - 2.1.2 Le zoom . . . 12
  - 2.1.3 Transformée de Fourier . . . 12
  - 2.1.4 Spectrogrammes . . . 13
- 2.2 Data augmentation . . . 13
  - 2.2.1 Intérêt théorique . . . 13
  - 2.2.2 Rajout de bruit blanc . . . 14
  - 2.2.3 Simulation de distance . . . 14
- 2.3 Traitement du signal . . . 15
  - 2.3.1 Filtre passe haut . . . 15
  - 2.3.2 Mise à l'échelle . . . 15
- 2.4 Les Pipelines . . . 15
- 2.5 Les PDF (Fiches d'analyse) . . . 16

3 Le travail à distance **17**
- 3.1 Organisation du travail à distance . . . 17
- 3.2 Outils utilisés . . . 17
  - 3.2.1 Présentation de GitHub . . . 17
  - 3.2.2 Présentation de Google Colab . . . 18
  - 3.2.3 TensorFlow . . . 19
  - 3.2.4 Présentation de LaTex . . . 19

4 Conclusion **20**
4.1 Conclusion de l'étude 20
4.2 Perspectives 20
4.2.1 20
4.3 Les apports du stage 20
4.3.1 les apports generaux 20
4.3.2 les apports personels 20

Chapitre 1

Introduction

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 public de l'enseignement supérieur qui entre dasn la catégorie des établissements à caractère scientifique, culturel et professionnel (EPSCP). Ces missions sont régies par l'article L123-3 du code de l'éducation :

- 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 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 Donnees ;

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

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 groupe de travail, nommé SpikeTrain, 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**.

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 **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.

1.2 Contexte du stage

1.2.1 Le Challenge : Classifier les Cachalots

Le challenge "Dyni Odontocete Click Classification" (https://challengedata.ens.fr/challenges/32) 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 d'apprentissage labelisée (comportant 113000 exemples) ainsi qu'une base de test non labelisée (comportant 20000 exemples) sur laquelle nous pouvions évaluer les performances réelles de nos classifieurs.

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 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 probléme uniquement basé sur du machine learning notament des réseaux de neurones et des réseaux de neurones convolutifs
— Divers traitements des signaux bruts
— De l'augmentation de données

Leurs meilleurs résultats étaient de l'ordre de 98% de réussite en accuracy sur la base labelisée contre 72 % de réussite sur la base non labelisée, 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 labelisée comme des données de meilleure qualité que celles de la base labelisée.

C'est pour mieux comprendre ce différentiel mais surtout le probléme 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.

Mais avant de pouvoir commencer cette partie du travail il a fallu commencer par comprendre un certaine 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 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 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.

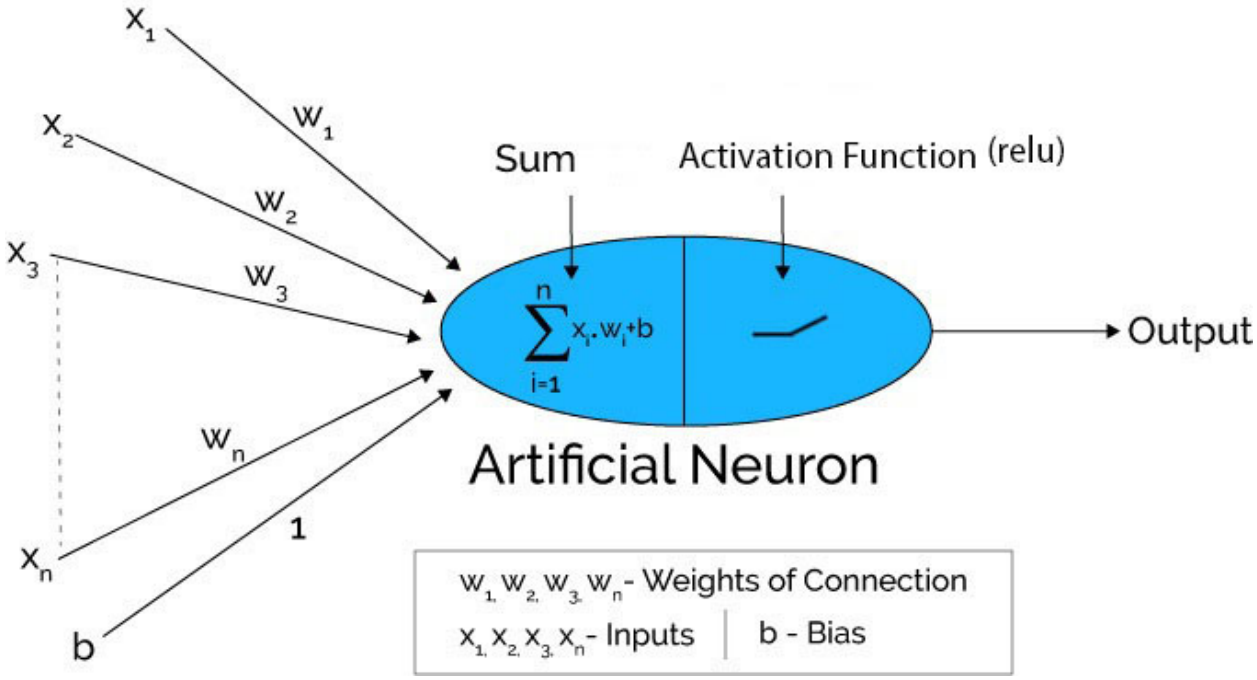


FIGURE 1.1 – Fonctionnement d'un neurone seul.

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 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.

Ce type d'architecture est présentée sur la figure 1.2.

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'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. 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 1.3).

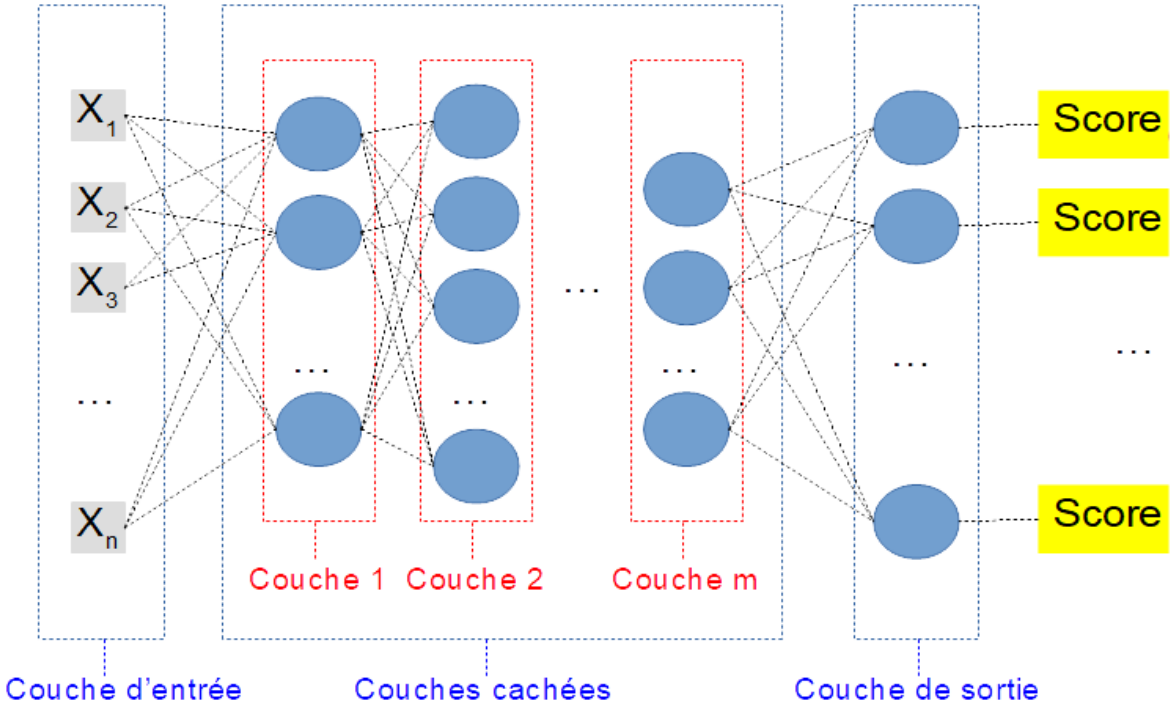


FIGURE 1.2 – réseau de neurones classique en couches.

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

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).

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.

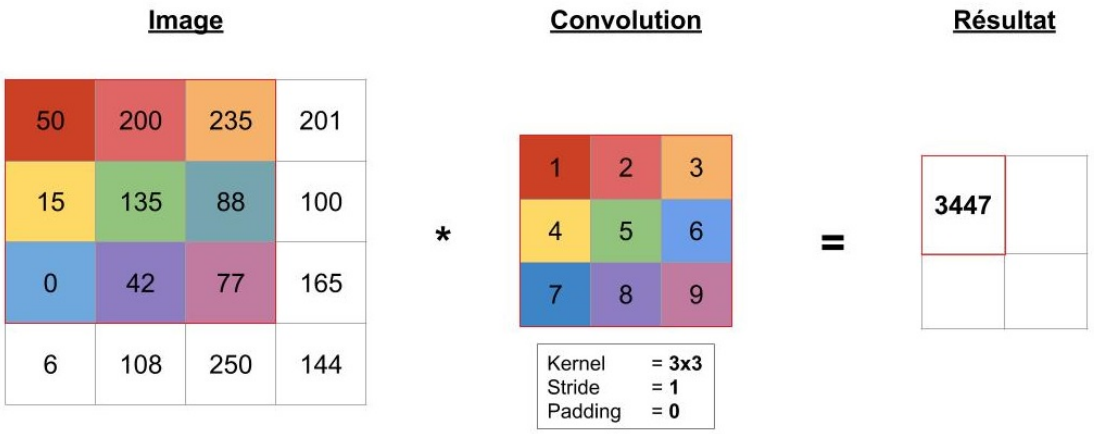


FIGURE 1.3 – L'image d'entrée est découpée en *patchs* sur lesquels est appliqué un filtre de convolution. La sortie de ce fitre appliqué à chacun des patchs donne la couche de sortie de la couche convolutive (appelée carte ou map). La convolution est paramètrée par : le *kernel* (la forme du patch), le *stride* (le déplacement du filtre) et le *padding* (la façon dont les bords de l'image vont être traités).

1.2.5 TensorFlow

à renvoyer vers la section outils

FIGURE 1.4 – 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.

1.3 Présentation du plan du rapport

Chapitre 2

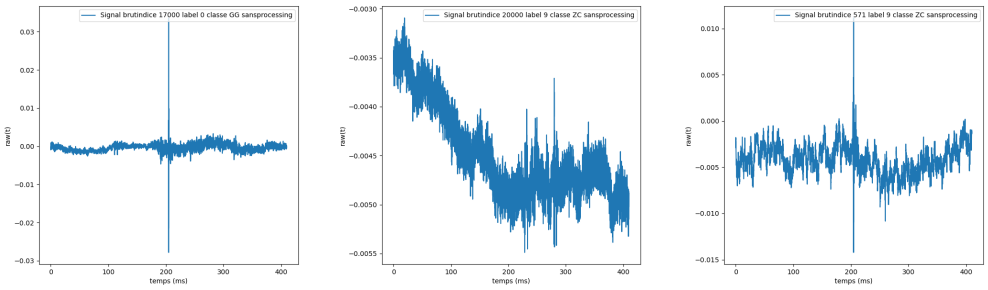
Analyse et traitement des données

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 labelisée ainsi que le n°571 de la base non labelisée) que nous observerons sous diverses formes puis sur lesquels nous effectuerons un certain nombre de traitements.

2.1.1 Les signaux bruts

Dans un premier temps on commence par observer les signaux sans traitement sous différentes formes.



(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 :

— Le clic semble durer 5 millisecondes.

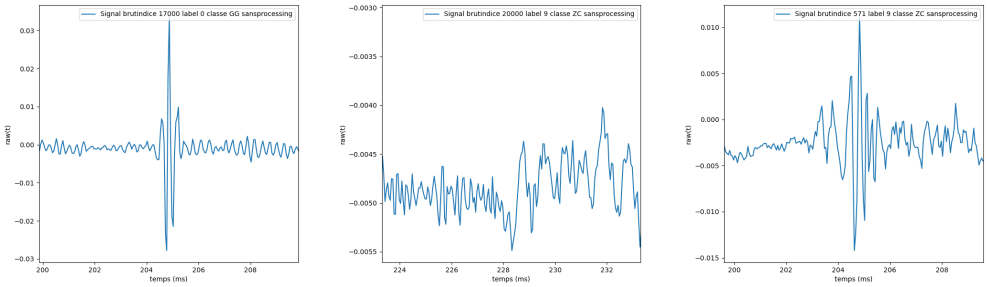
— L'amplitude des clics semble variable.

— Il arrive que le bruit soit parfois suffisament important par rapport au clic pour rendre son identification difficile voir impossible.

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 qui sera logiquement le milieu du clic puis on rajoute l'équivalent de la durée d'un clic avant et aprés ce maximum.



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

FIGURE 2.2 – Les signaux n°17 000, 20 000, 571 avec zoom temporel

Hors on constate que celà ne suffis pas 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 vas donc par la suite appliquer un filtre pour supprimer les bruits parasites (dont on reparlera dans la partie traitement du signal).

On peux déjà observer plusieurs phénoménes : -tout d'abord l'efficacité du zoom semble corélée à la qualité du signal de départ -Les "bons clics" semblent se situer aux alentours de 200 ms (ils sont donc bien centrés) -Leur intensité est très variable pouvant aller jusqu'a un facteur 10 entre 2 clics. Il parait donc pertinent par la suite de les normaliser. 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 pertinent.

2.1.3 Transformée de Fourier

Avant d'observer les spectrogrammes il convient de commencer par expliquer et observer les Transformés de Fourier de nos 3 signaux (attention les signaux ne sont pas du tout à la même échelle !) :

La Transformés de Fourier est un outil mathématique nous permettant de passer du domaine temporel au domaine fréquenciel, ainsi la transformé de fourier d'un enregistrement nous donneras un peux à l'image d'un histogramme la répartition fréquentielle de celui ci.

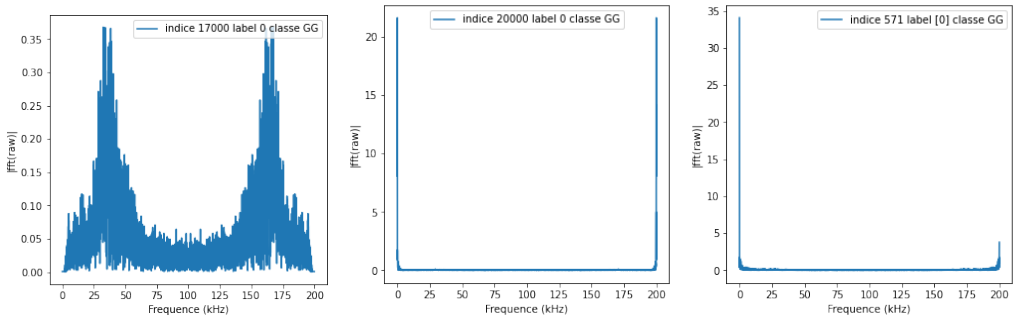


FIGURE 2.3 – Transformé de Fourier des signaux 17000, 20000 et 571

2.1.4 Spectrogrammes

Les Spectrogrammes sont simplement des transformé de fourier non pas sur l'ensemble de l'enregistrement mais a chaque échantillon, ainsi si l'échantillon est de 100 millisecondes par exemple le spectrogramme seras un enchainement de transformé de fourier toutes les 100 millisecondes.

On commence tout d'abord par observer leurs Spectrogrammes en 2D (a noter qu'ici pour éviter les phénoménes d'écrasement (les fréquences majoritaires pouvant écraser les autres) on utilise une échelle logarithmique) :

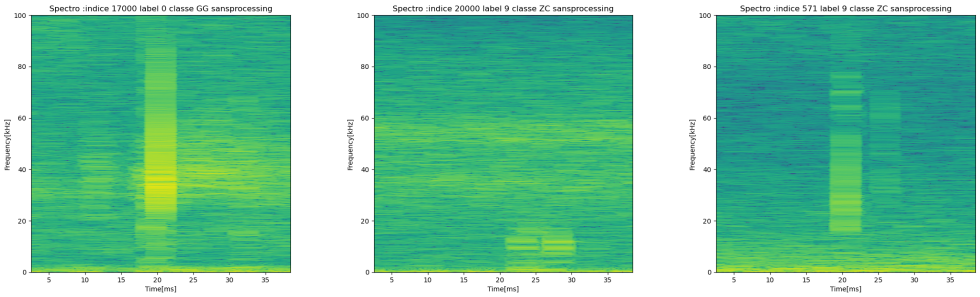


FIGURE 2.4 – Spectrogramme 2D des signaux n° 17 000, 20 000 et 571

On constate tout d'abord que les clics sont bien visibles et que leurs gamme de fréquence semble relativement variable d'un enregistrement à l'autre.

Puis on observe 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.

Pour cela selon la nature des données de cette base on va par exemple : -Rajouter du bruit

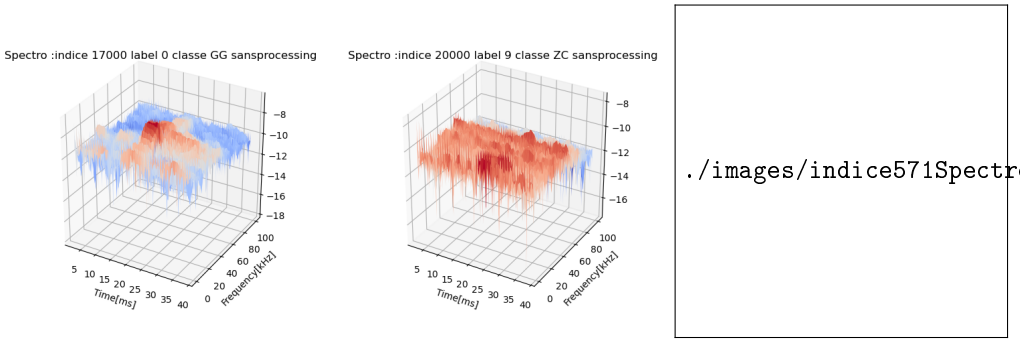


FIGURE 2.5 – Spectrogramme 3D des signaux n° 17 000, 20 000 et 571

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 probléme

L'intérêt le plus évident de cette opération est de simplement 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 avions besoin de l'utiliser pour une autre raison. Effectivement nous avions remarqué que certains exemples avaient subis de fortes dégradations notament 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 a l'une des classes) plutôt que de supprimer ces dégradations il nous a paru plus pertinent de d'abord essayer de rajouter des exemples avec des dégradations similaires issuent d'exemples choisis aléatoirement parmis nos classes sur lesquelles ont aurait fait de la data augmentation. En effet ces dégradations pouvant être simplement dues a des problémes pendant l'enregistrement des clics (bateau navigant a proximité du micro ou encore d'autres animaux émettant divers bruits à proximité par exemple) il est plus que problable que les enregistrements qui seront soumis par la suite à notre classifieur contiennent également des dégradations qui ne devront pas entraver le bon fonctionnement de celui-ci.

2.2.2 Rajout de bruit blanc

Comme on a pu l'observer précedement certains enregistrement sont plus ou moins bruités on vas donc dans un premier temps essayer d'en dimunuer l'impact via de la data augmentation. Autrement dit on vas aléatoirement rajouté des exemples issus d'enregistrements choisis aléatoirement auquels ont a rajouté artificiellement du bruit.

2.2.3 Simulation de distance

Lors de la prise des enregistrements audios les animaux peuvent se situer plus ou moins loin du ou des micro, ce qui peux potentiellement influencer notre classifieur. En effet certaines especes plus fuyardes peuvent par exemple rester systématiquement plus éloignées du micro que les autres poussant le classifeur a assimiler une grande distance a une éspece 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 vas rajouter aléatoirement des signaux choisis dans des classes aléatoires a une distance aléatoire (on simule les effets de la distance sur le signal).

2.3 Traitement du signal

Comme nous avons pu le voir précedement il arrive que certains enregistrements aient subis 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 suffis pas. Parcequ'ils seraient trop dégradés ils empêcheraient l'identification de l'espece, cela peut être un bruit tellement important qu'il recouvrirait le clic par exemple. Ainsi nous avons quand même dû faire du traitement du signal.

2.3.1 Filtre passe haut

Dans un premier temps pour résoudre les probléme vus précedement on vas 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 peux donc appliquer le filtre à l'enregistrement sans en altérer le clic. 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 le signal n° 20000 deviens parfaitement possible avec (à droite) :

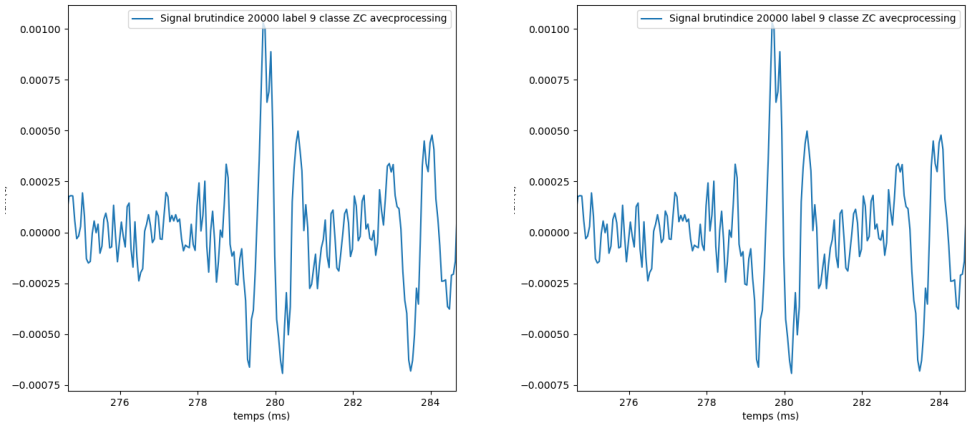


FIGURE 2.6 – Zoom du clic du signal 20 000 sans le filtre passe haut puis avec

2.3.2 Mise à l'échelle

En analysant les signaux on a constaté de grandes différences aussi bien en terme d'intensité que de fréquences sur leurs clics, hors pour bien fonctionner le réseau de neurones convolutif a besoin de données à la même échelle. C'est pourquoi on vas devoir mettre à l'échelle nos signaux, autrement dit on vas normaliser entre 1 et -1 l'ensemble de nos signaux.

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 directement dans le réseau de neurones.

Cette méthode présente plusieurs interets majeurs : -Premièrement elle nous évite de stocker le résultat des opérations intermédiaires faisant gagner beaucoup de mémoire -Deuxièmement elle nous permet d'optimiser grandement l'ensemble du processus de prétraitement des données. -Troisièmement ce procédé améliore grandement les performances de tensorflow

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)

Les bases de données contenant un très grand nombre d'exemples (environs 90 000 au total que l'on peut visualiser sous 12 formes différentes soit potentiellement 1 080 000 images) afin de pouvoir exploiter les analyses faites précedement il a fallu créer un certain nombre d'outils afin de pouvoir aisement trier les données. Pour cela je me suis inspiré du systéme de pipeline que nous venons de voir, ainsi dans un premier temps l'ensemble des fonctions nécessaires à l'analyse était stocké dans le cachalot_helper.

Dans un second temps j'ai créer dans un autre python une fonction paramètrable permettant tout d'abord de sélectionner ou un certain nombre de signaux dans un certain nombre de labels ou des signaux bien specifiques puis de générer (à l'aide du cachalot_helper) avec et ou sans preprocessing (les traitements du signal) avec ou sans zoom : -Des plots 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 créer un autre 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) 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 paramètrable qui va récupérer les fichiers tex précedement créés puis les réunir dans un seul fichier tex qui va automatiquement s'executer 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 paramètrable chargé de faire tourner l'ensemble des programmes vus précedement 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 génére 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 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 :

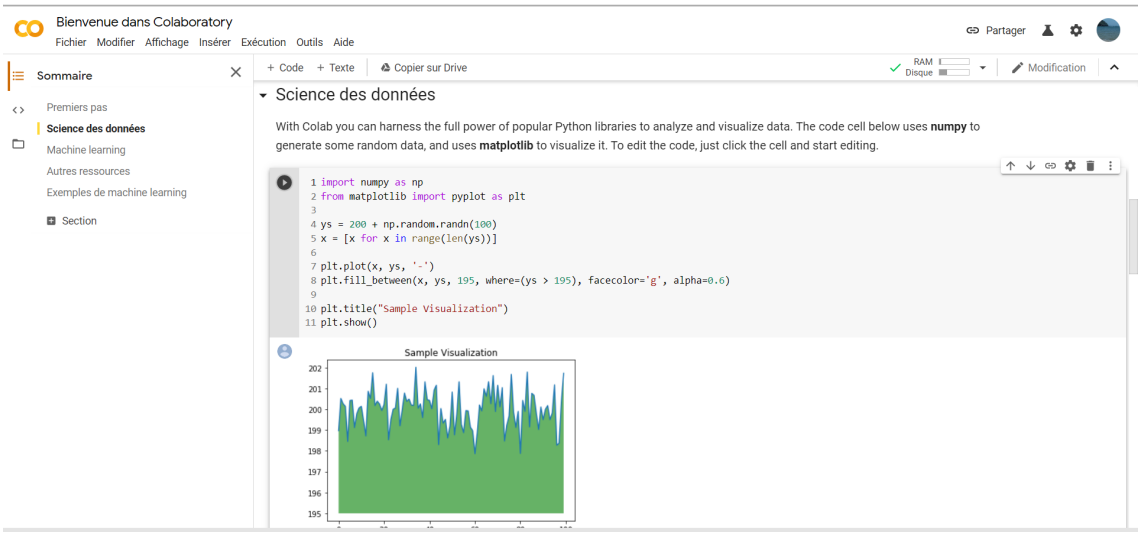


FIGURE 3.1 – Nous avons ici un exemple de code exécuté avec colab.

3.2.3 TensorFlow

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 s̈cientifiquec̈ela 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 :

4.2.1

Nous pouvons interpréter tout le réseau comme un utilisateur et cet entretient 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 entretient 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 nécéssaire 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.

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

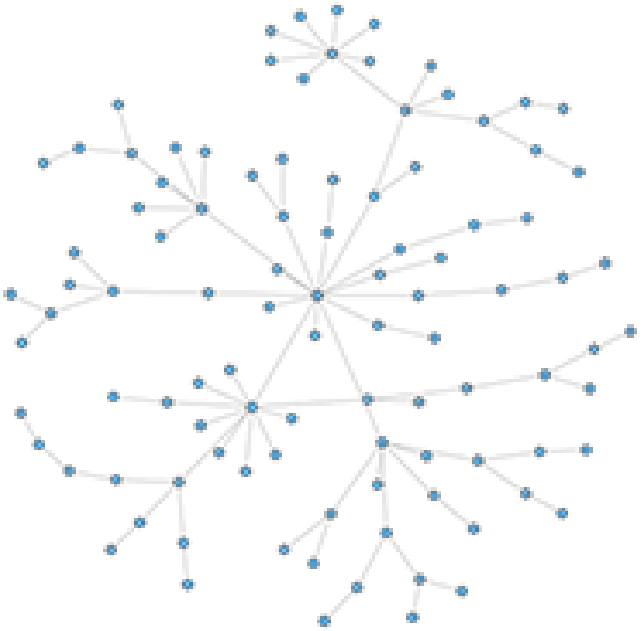


FIGURE 4.1 – Architecture scale-free