



le cnam

Stage de Master 1 TRIED
Deux mois

***Détection de la parole rejouée en appliquant
l'approche lightCNN***

Réalisé par **Saoussan TRIGUI**

Encadrant : Moez AJILI
Encadrant pédagogique : Nicolas THOME

Table des matières :

1. Introduction	2
2. Environnement de travail	2
2.a. L'entreprise	2
2.b. Les outils utilisés	3
3. État de l'art.....	3
3.a. La Reconnaissance Automatique de Locuteur (RAL)	3
3.b. ASV et l'usurpation de l'identité	5
3.c. Détection de la parole rejouée (Spoofing).....	7
3.d. Les réseaux de neurones.....	10
3.e. Décision et évaluation :	15
4. Expériences et résultats	16
4.a. Prétraitements	16
4.b. Modélisation	20
5. Conclusion :	24

1. Introduction

Dans ces dernières années, le domaine de la reconnaissance automatique du locuteur (RAL) a vécu une révolution remarquable vu son utilité dans divers domaines. Les systèmes de RAL sont employés pour sécuriser les informations confidentielles lors des transactions bancaires par téléphone, les services téléphoniques d'information et de réservation, etc. Le domaine légal a également profité de ces avancées dans divers contextes de crimes (appels par un Kidnappeur, menaces de mort, fausse alerte à la bombe...). Plus récemment, cette technologie est utilisée par les assistants personnels intelligents (par exemple, Google Home et Amazon Alexa) pour identifier les utilisateurs préenregistrés.

On sait depuis longtemps que les systèmes de reconnaissance biométrique sont vulnérables à la manipulation par usurpation, également appelée « Presentation Attack ». En reconnaissance du locuteur, il y a quatre types d'attaques principales d'usurpation d'identité, à savoir, l'imitation (impersonation), conversion de la voix (voice conversion), synthèse de la parole (speech synthesis) et la parole rejouée (replayed speech). Je m'intéresse au cours de ce stage à la problématique de détection de la parole rejouée.

Dans ce travail, j'ai étudié l'état de l'art de la reconnaissance automatique du locuteur et de la détection de la voix usurpée, ainsi que l'état de l'art de l'apprentissage profond (Deep Learning ou DL). Enfin, j'ai construit et testé un système de détection de la parole rejouée basé sur le DL intitulé LightCNN.

2. Environnement de travail

2.a. L'entreprise

J'ai effectué mon stage au sein de l'entreprise « My Voice AI » qui est une startup spécialisée en authentification biométrique, fondée en 2017. Elle est basée à Londres et possède une filiale à Paris. Elle est formée d'une dizaine de scientifiques et d'ingénieurs.

« My Voice AI » offre principalement des technologies basées sur l'identification du locuteur. Parmi les produits conçus par cette entreprise, on peut citer l'authentification des passagers du covoiturage mais également la sécurisation des paiements et des transactions en mode mains libres dans un véhicule. « My Voice AI » fournit d'autres fonctionnalités supplémentaires centrées sur la voix, comprenant la détection du sexe et de l'âge, l'isolement du locuteur et la détection des émotions.

Au début de ce stage, mon encadrant et moi avons défini les buts à atteindre et les étapes qui me permettent de réaliser la mission. Durant toute la période, on a fait des réunions pour évaluer le travail réalisé et pour discuter des futures tâches.

2.b. Les outils utilisés

De point de vue matériel, My voice AI a mis à ma disposition un accès sur un serveur contenant une carte mémoire de 256 Go, 3 cartes graphiques (Graphical Processing Unit ou GPU) Nvidia de type « GeForce GTX 1080 Ti » (voir la Figure 1) et 4 processeurs de type « Intel(R) Xeon(R) CPU E7-8880 v3 @ 2.30GHz ». Chaque processeur contient 18 cœurs et 36 threads, cette combinaison nous permet de travailler sur 144 processus.

```
trigui@myvoiceai: /users/trigui
Every 2.0s: nvidia-smi
myvoiceai

Wed Oct 23 14:37:28 2019

+-----+
| NVIDIA-SMI 430.50          Driver Version: 430.50          CUDA Version: 10.1   |
+-----+-----+-----+-----+-----+-----+
| GPU  Name                Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp   Perf          Pwr:Usage/Cap|  Memory-Usage | GPU-Util  Compute M. |
+-----+-----+-----+-----+-----+-----+
|  0   GeForce GTX 108...    Off      | 00000000:04:00:0 | Off      | N/A   |
| 51%    84C    P2      179W / 250W | 5460MiB / 11178MiB | 99%      | Default |
+-----+-----+-----+-----+-----+-----+
|  1   GeForce GTX 108...    Off      | 00000000:87:00:0 | Off      | N/A   |
| 36%    62C    P2      85W / 250W | 2567MiB / 11178MiB | 64%      | Default |
+-----+-----+-----+-----+-----+-----+
|  2   GeForce GTX 108...    Off      | 00000000:C4:00:0 | Off      | N/A   |
| 34%    59C    P2      82W / 250W | 2487MiB / 11178MiB | 75%      | Default |
+-----+-----+-----+-----+-----+-----+

+-----+-----+-----+-----+-----+-----+
| Processes:                                     GPU Memory |
|  GPU       PID    Type    Process name      Usage   |
+-----+-----+-----+-----+-----+-----+
|    0      60369     C       python             2677MiB |
|    0      80125     C       python             2773MiB |
|    1      80125     C       python             2557MiB |
|    2      80125     C       python             2477MiB |
+-----+-----+-----+-----+-----+-----+

[0] 0:watch*Z
```

Figure 1 : GPU Nvidia GeForce GTX 1080 Ti

De point de vue logiciel, j'ai utilisé le langage python (version 3.7.3) et la librairie PyTorch (version 1.1.0) pour l'apprentissage profond. Cette librairie est basée sur la technologie CUDA (version 10.1) qui permet d'exploiter les GPU (calculs matriciels...).

3. État de l'art

3.a. La Reconnaissance Automatique de Locuteur (RAL)

La RAL fonctionne en scannant les aspects de la parole qui diffèrent entre les individus. Chacun a une façon de parler qui lui est propre, c'est le résultat de leur physiologie (forme et taille de la bouche et de la gorge) et de leurs comportements (hauteur de la voix, accent, style de conversation, etc).

Dans la RAL, il existe deux types de tâches, à savoir, la vérification du locuteur (Automatic Speaker Verification ou ASV) et l'identification du locuteur (Automatic Speaker Identification ou SID). La première consiste à comparer deux enregistrements vocaux afin

de déterminer si ceux-ci appartiennent (ou pas) au même locuteur. Elle est surtout utile dans des cas d'investigation judiciaires. La deuxième consiste à identifier une personne (à partir de sa voix) parmi un ensemble de locuteurs connus. Elle peut servir à identifier les interlocuteurs dans une discussion [1]. Dans la suite, je m'intéresse uniquement à l'ASV.

La ASV est formée de trois composants principaux (voir la Figure 2). Au début, il y a l'étape d'extraction des paramètres acoustiques du locuteur « authentique » (enrolment). La seconde étape est la modélisation. Elle consiste à former un modèle qui résume les informations spécifiques au locuteur obtenues à partir de l'échantillon (l'enregistrement authentique). La dernière étape consiste à comparer les mêmes paramètres qui sont extraits à partir d'un segment de test inconnu avec celui du modèle du locuteur authentique. Cette comparaison fournit un score, une valeur scalaire, qui indique si les deux enregistrements vocaux sont prononcés par le même locuteur (c'est-à-dire la même source) ou non. Le principe est donc simple : si ce score est supérieur (ou inférieur) à un seuil choisi, le système accepte (ou rejette) le locuteur.

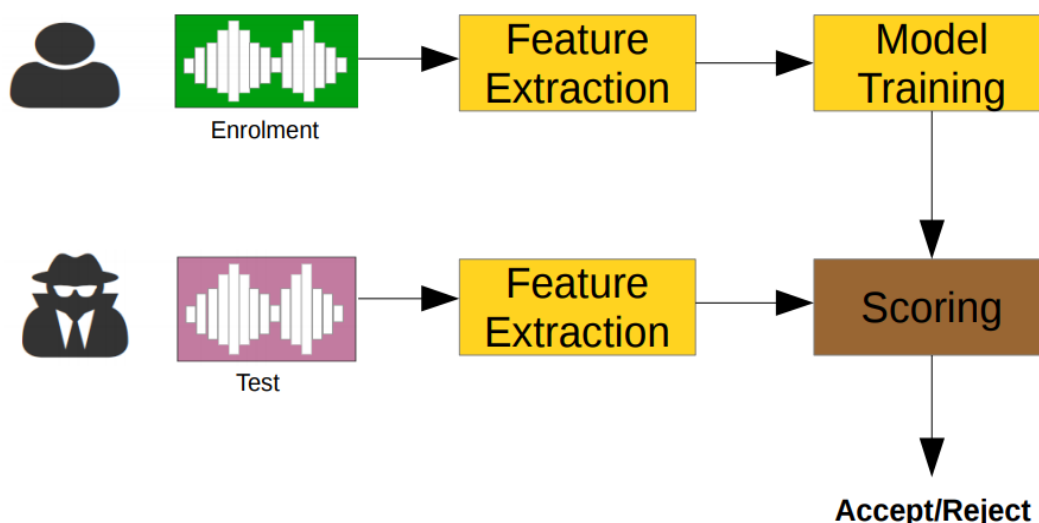


Figure 2 : Les principaux composants de l'ASV

A) Extraction et prétraitement des paramètres :

La paramétrisation de la parole consiste à transformer le signal de parole en un ensemble des vecteurs des paramètres dans lesquels les propriétés spécifiques du locuteur sont accentuées [AJILI, 2017].

Les paramétrisations les plus utilisées se basent sur une représentation cepstrale de la parole. Ce processus consiste à extraire des vecteurs de caractéristiques sur une partie locale du signal par l'application d'une fenêtre coulissante temporelle.

Parmi les représentations les plus connues, on distingue les MFCCs (Mel Frequency cepstral coefficients), les LFCC (Linear frequency cepstral coefficients), les LPCC (Linear predictive cepstral coefficient) ou les paramètres PLPs (Perceptual Linear Prediction).

B) Modélisation

Les modèles à mixture de Gaussiennes (GMM) (Reynolds et al., 2000; Reynolds, 1995) et les modèles de Markov cachés (HMM) sont les modèles stochastiques les plus connues. Ces modèles étaient dominants dans le domaine de l'ASV pendant plusieurs années. D'autres approches sont apparues comme l'analyse factorielle (Kenny et Dumouchel, 2004; Kenny et al., 2007) et la méthode i-vector (Dehak et al., 2011) qui est devenue l'état de l'art pour l'ASV. Dans ces dernières années les réseaux de neurones profonds (DNN) ont donné encore de meilleures performances (Variani et al., 2014; Rouvier et al., 2015; Matejka et al., 2016).

C) Scoring :

L'étape de scoring produit un score qui simule la probabilité d'appartenance au même locuteur à partir de l'étape de modélisation [AJILI, 2017].

Les méthodes de scoring sont choisies selon l'approche utilisée. En effet, pour les méthodes de GMM, un score de probabilité peut être utilisé pour calculer la similitude entre un GMM de locuteur et les données d'un locuteur inconnu (Hansen et Hasan., 2015). Pour l'approche de i-vector et celle de DNN, la PLDA (Probabilistic Linear Discriminant Analysis) est une des méthodes les plus efficaces. Pour effectuer la classification (décision entre genuine et impostor), on doit fixer un seuil de décision (threshold) sur le score produit.

3.b. ASV et l'usurpation de l'identité

Les systèmes ASV, font face à certaines attaques d'usurpation d'identité qui peuvent considérablement affecter leurs performances. Ces attaques peuvent être réalisées avec différentes techniques [2] :

1. Imitation (impersonation) : se base sur l'imitation humaine sans utilisation de la technologie.
2. Synthèse de la parole (speech synthesis) : connue aussi sous le nom "text-to-speech" (TTS), une technologie qui génère la voix à partir d'un texte.
3. Conversion de la voix (voice conversion) : consiste à convertir la voix enregistrée d'une personne X à la voix de la personne à usurper.
4. Parole rejouée (Replay) : consiste à lire un enregistrement vocal de la personne à usurper à l'aide d'haut-parleurs (ex : Smart phone).

Dans ce travail, je m'intéresse à la technique d'usurpation par parole rejouée. Cette technique est considérée parmi les plus dangereuses car elle est très facile à produire et ne nécessite quasiment pas de connaissances en acoustique.

Selon [Wu et al.], la différence entre la parole authentique et la parole rejouée est minime. Une comparaison a été réalisée entre un enregistrement d'un locuteur X (Target speech) et celui rejoué avec le haut-parleur d'un ordinateur (Replay speech). La Figure 3 illustre la grande similarité entre les spectrogrammes des deux enregistrements.

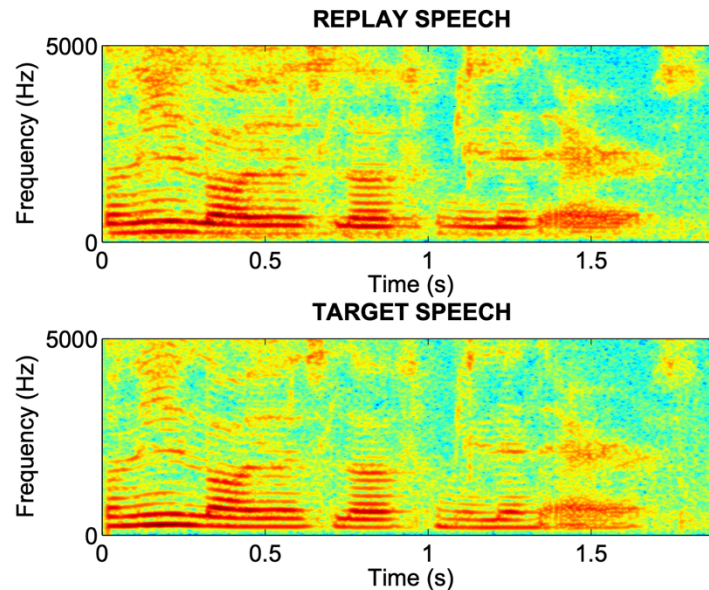


Figure 3 : Comparaison entre spectrogramme d'un enregistrement authentique (Target speech) et son enregistrement rejoué (Replay speech)

Dans le cadre de l'ASV, plusieurs chercheurs ont évalué la menace de l'attaque par enregistrement rejoué. Pour ce faire, ils ont modifié chaque couple d'enregistrements ayant des locuteurs différents (Enr Loc X, Enr Loc Y) en remplaçant l'enregistrement du locuteur Y (Enr Loc Y) par la version rejouée de l'enregistrement du locuteur X (Enr Loc X).

En 1999, une étude est réalisée par la chercheuse [Lindberg J]. Elle a utilisé un système ASV basé sur le modèle de Markov caché (Hidden Markov Model ou HMM) sur un texte déterminé à l'avance (text-dependent) formé par des chiffres enregistrés. Cette étude montre une augmentation significative des erreurs qui sont multipliés par 27 pour les locuteurs féminins et par 12 pour les locuteurs masculins. Pour un même seuil, le système considère comme authentique tous les enregistrements qui viennent d'un locuteur différent.

Dans un contexte text-independent, une étude est réalisée par les chercheurs [Villalba et Lleida] en 2011 en utilisant un ASV se basant sur l'analyse factorielle (JFA). Les attaques par enregistrements rejoués ont causé une multiplication des erreurs de 95 fois.

Il est montré donc, à travers ces études, que les systèmes de ASV sont très sensibles à l'attaque par un enregistrement rejoué. Ce sont principalement les fausses acceptations

qui augmentent dans ces expériences car les systèmes de ASV considèrent, à tort, les nouveaux enregistrements usurpés par parole rejouée comme authentiques.

3.c. Détection de la parole rejouée (Spoofing)

Afin d'empêcher l'augmentation des fausses acceptations de l'ASV suite à l'attaque par enregistrement rejoué, les scientifiques ont proposé des contremesures pour détecter l'attaque. Un tel système est composé de trois étapes, à savoir, l'extraction de paramètres acoustiques, la modélisation et la décision.

A) Extraction de paramètres acoustiques

Un signal acoustique est par définition un signal analogique représentatif de l'onde sonore. Dans le domaine de traitement de la parole, on utilise souvent des méthodes d'extraction de paramètres basés généralement sur la transformation de fourrier. Cette transformation permet de passer du domaine temporel (signal) vers le domaine fréquentiel (spectre) afin d'obtenir une nouvelle représentation plus compacte et plus adaptée à la modélisation du locuteur. Le spectre est la représentation des amplitudes des différentes fréquences présentes dans le signal. La Figure 4 schématise la production du spectre par la transformée de fourrier [3].

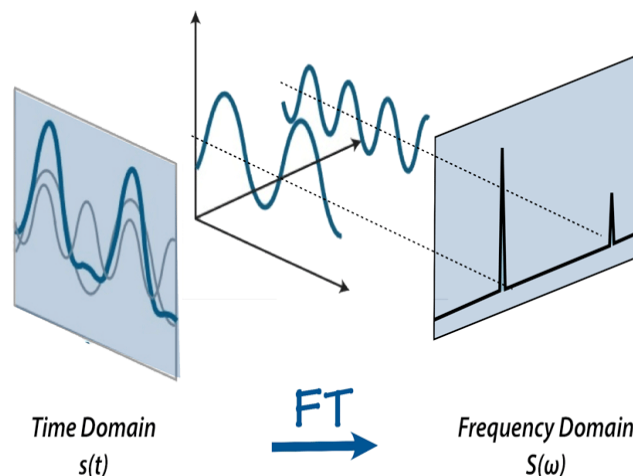


Figure 4 : Schématisation d'une transformée de fourrier

La Figure 5 représente le spectrogramme d'un enregistrement sonore. Le temps et les fréquences sont étalés respectivement sur l'axe x et y. L'amplitude de chaque fréquence f à une trame (frame) de l'instant t est dessinée selon l'échelle de couleur en dB [4].

Les MFCCs sont les paramètres acoustiques les plus utilisés. Ils ont été inventés par Davis et Mermelstein dans les années 1980 et ils sont utilisables depuis ce temps-là. Il s'agit de coefficients cepstraux calculés par une transformée en cosinus discrète appliquée au spectre de puissance d'un signal. Les bandes de fréquence de ce

spectre sont espacées logarithmiquement selon l'échelle de Mel pour simuler les spécificités de la perception humaine du son [4].

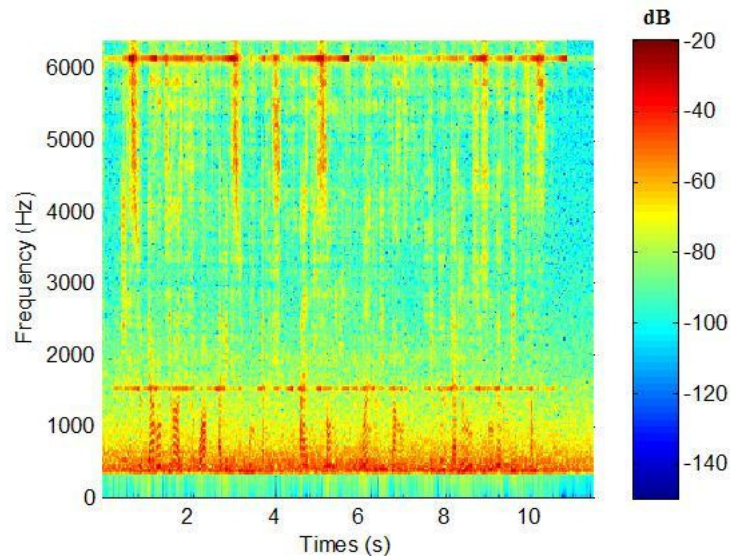


Figure 5 : Exemple d'un spectrogramme

La représentation cepstrale du signal est extraite des vecteurs de caractéristiques sur une partie locale du signal par l'application d'une fenêtre coulissante temporelle (voir la Figure 6). Cette fenêtre est appliquée depuis le début du signal, puis déplacée plus loin et ainsi de suite, jusqu'à la fin du signal. Chaque application de la fenêtre à une partie du signal de parole fournit un vecteur caractéristique. La longueur de la fenêtre est généralement fixée entre 20 et 30 millisecondes. D'un autre côté, la valeur du pas est choisie afin d'avoir un chevauchement entre deux fenêtres consécutives. Un pas de 10 millisecondes est généralement utilisé. Afin de capturer la variation temporelle, les coefficients de dérivation temporelle (deltas Δ et doubles deltas Δ^2) sont généralement ajoutés aux les vecteurs caractéristiques.

La sortie de cette étape est une séquence de vecteurs de paramètres représentant un segment de parole $S = \{x_1, \dots, x_T\}$, où x_t est un vecteur caractéristique résultant de l'application de la fenêtre, t ($t \in [1, 2, \dots, T]$) [AJILI, 2017].

B) Modélisation :

En 2011, [Villalba et Lleida] ont construit une contremesure basée sur les Machines à vecteurs de support (SVM). Ce système se base sur des informations liées à la réverbération et au bruit pour détecter les enregistrements rejoués. Une réduction des erreurs est constatée dans un système de l'ASV « text-independent » de 4 fois.

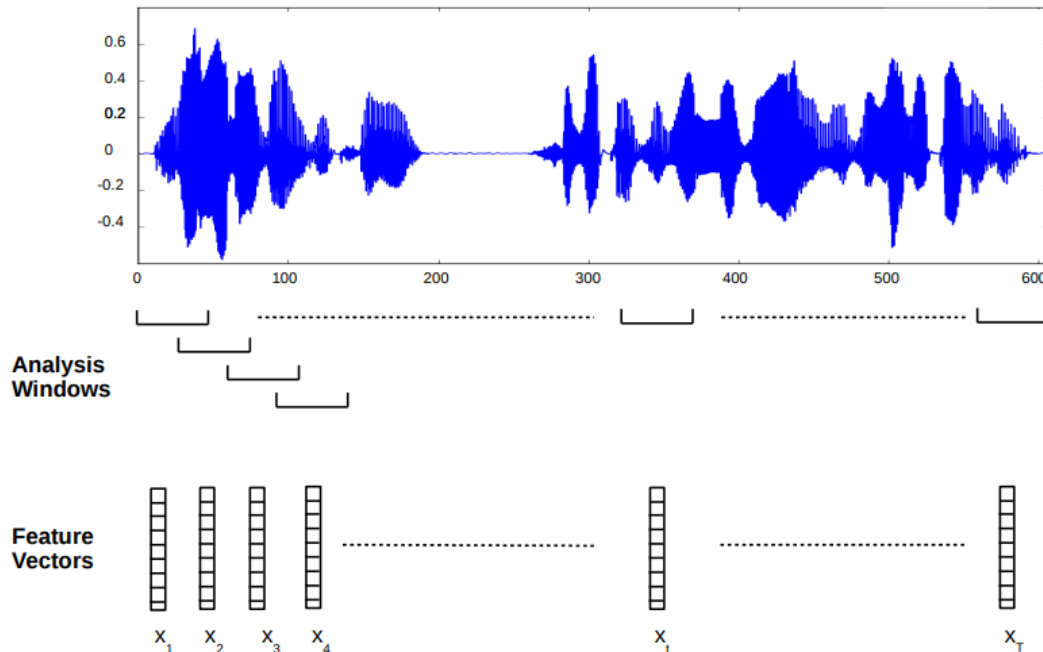


Figure 6 : Analyse à court terme et paramétrage d'un signal de parole

Une autre contremesure est proposée par [Wang et al]. Pour détecter les enregistrements rejoués, des informations sur le bruit généré par le mécanisme d'enregistrement et celui causé par l'éventuel haut-parleur (utilisé pour rejouer l'enregistrement) ont été utilisés. Le résultat montre que le taux d'erreur du système d'ASV basé sur un modèle GMM-UBM (Universal Background Model - Gaussian Mixture Model) a diminué de 4 fois.

Plus récemment, les scientifiques se sont intéressés plus particulièrement à la tâche de détection de la parole usurpée « spoofing detection ». Cet intérêt a fait naissance au défi ASVspoof2015.

Le meilleur système dans ce défi se base sur un modèle GMM qui utilise deux caractéristiques acoustiques CFCC et CFCC-IF. La deuxième contremesure se base sur un modèle SVM avec noyau linéaire qui utilise les i-vectors. Ce dernier est extrait à partir des trois types de caractéristiques MFCCs, MFPC et CPP. Quant à la troisième contremesure, elle exploite plusieurs caractéristiques liées à la phase et à l'amplitude. Chaque type de caractéristique est utilisé par un réseau de neurones (perceptron multicouches, MLP). Le score final représente la moyenne des scores de chaque MLP. Ces trois contremesures atteignent toutes un taux d'erreur très faible (inférieur à 2,6%) [5].

Si l'édition 2015 du défi s'est intéressée à la conversion et la synthèse de la voix, l'édition 2017 a été consacrée entièrement à l'usurpation par parole rejouée. Les approches utilisées dans ce défi peuvent être réparties en deux catégories. La première catégorie concerne les modèles génératifs (expl. GMM) et les systèmes i-vectors/PLDA. La

deuxième catégorie concerne les modèles discriminatifs tels que les modèles SVMs et l'apprentissage profond (DNN). Les systèmes à base de DNNs ont été les plus performants. En effet, cinq parmi les dix meilleurs systèmes utilisent les DNNs.

3.d. Les réseaux de neurones

Le Machine Learning est un des champs d'étude de l'intelligence artificielle. Il s'agit d'une discipline scientifique concernée par le développement, l'analyse et l'implémentation de

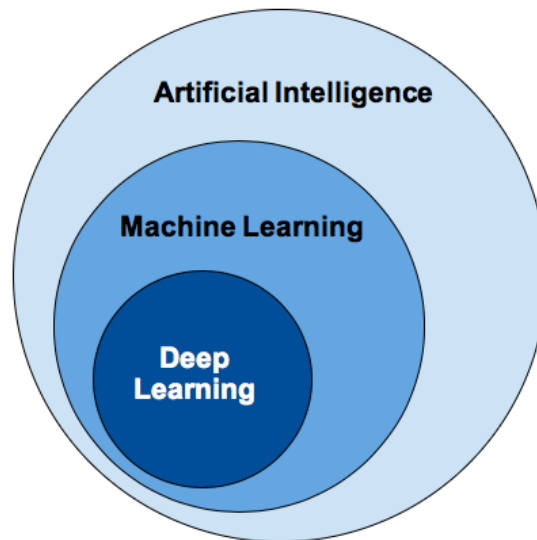


Figure 7 : Deep learning

méthodes automatisables qui permettent à une machine (au sens large) d'évoluer grâce à un processus d'apprentissage, et ainsi de remplir des tâches qu'il est difficile ou impossible de remplir par des moyens algorithmiques plus classiques [6].

Quatre catégories principales de méthodes d'apprentissage automatique sont connues, à savoir, l'apprentissage supervisé, l'apprentissage non supervisé, l'apprentissage semi-

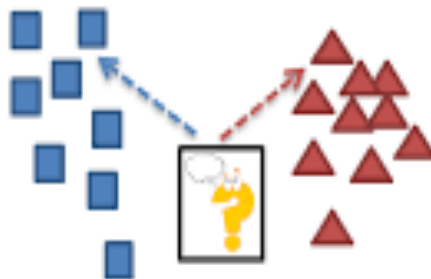


Figure 8 : Classification supervisée : Prédiction

supervisé et l'apprentissage par renforcement. La détection de la parole rejouée représente une tâche d'apprentissage supervisé, et plus précisément, une tâche de classification supervisée (classement).

L'apprentissage supervisé consiste à entraîner un modèle sur une partie étiquetée des données (c.à.d. qui contient les classes de référence) et ensuite à prédire la classe de chaque nouvelle donnée (non étiquetée) en utilisant ce modèle (voir la Figure 8). Les données requises doivent donc être divisées principalement en deux parties, à savoir, une partie pour l'apprentissage et une partie pour le test.

L'apprentissage profond (Deep Learning, ou DL) représente une catégorie d'approches de Machine Learning (voir la Figure 7), basées sur les réseaux de neurones.

A) Architecture d'un réseau de neurones

Un réseau de neurones est une architecture composée de neurones formels interconnectés permettant la résolution de problèmes complexes tels que la reconnaissance du locuteur, des formes, ou le traitement du langage naturel, grâce à

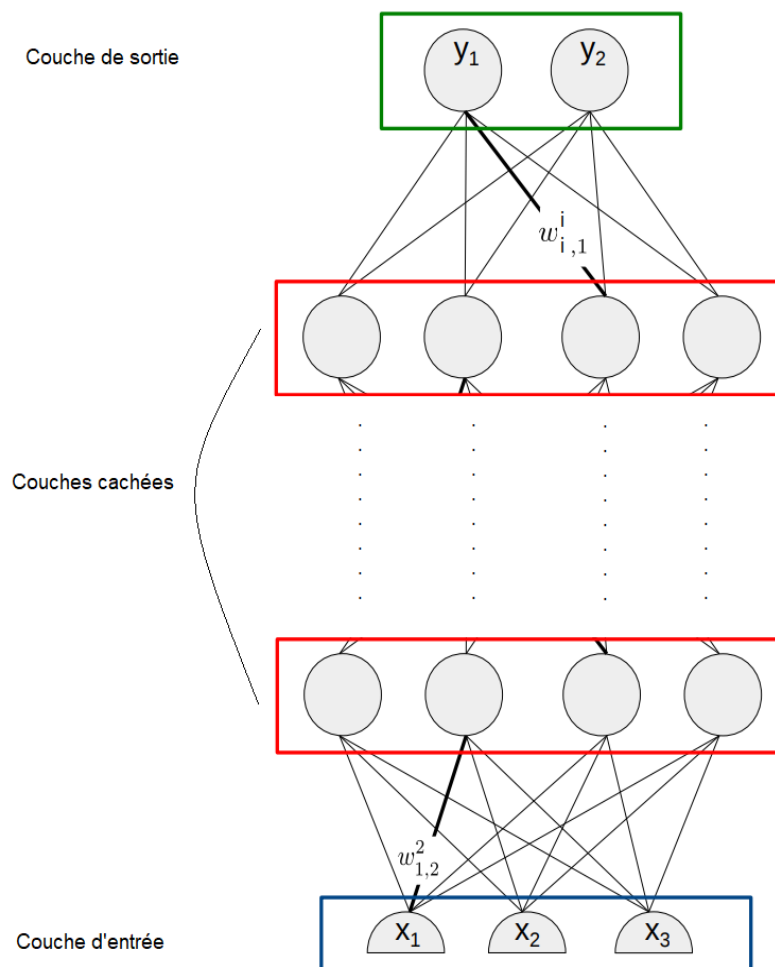


Figure 9 : Représentation d'un réseau de neurones

l'ajustement des coefficients de pondération dans une phase d'apprentissage. Un réseau neuronal s'inspire du fonctionnement des neurones biologiques et prend corps dans un ordinateur sous forme d'un algorithme [7].

Le Schéma de la Figure 9 est une représentation très simplifiée d'un réseau neuronal. Les trois neurones x_1 , x_2 et x_3 reçoivent les données et forment la couche d'entrée (Input layer). Le traitement de ces données est déterminé par leurs connexions avec les neurones internes qui forment les couches cachées (Hidden layers). L'information finale est envoyée sur les derniers neurones y_1 et y_2 qui forment la couche de sortie (Output layer). Chaque neurone est un perceptron, le schéma de la Figure 10 illustre son fonctionnement.

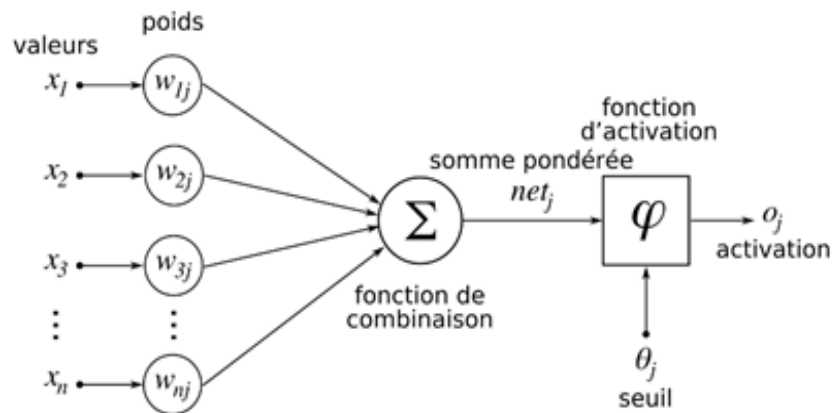


Figure 10 : Fonctionnement d'un perceptron

L'entraînement du réseau consiste à apprendre les pondérations automatiquement à partir des données de sorte que la sortie prévue (y_1 et y_2) soit proche de celle ciblée (réelle) pour toutes les entrées (x_1 , x_2 et x_3).

Dans un perceptron multi-couches (MLP) comme celui schématisé dans la Figure 9, la propagation des données commence par l'affectation de chaque instance de données aux sorties des neurones de la première couche.

Ces sorties représentent les entrées de la première couche cachée et sont pondérées avec les poids w_{ij} . Ces pondérations correspondent aux paramètres du réseau.

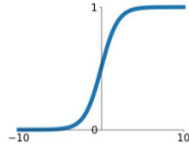
$$x_j = \sum_{i=1}^N w_{ij} y_i + b_j$$

Ensuite, une fonction d'activation f est appliquée à cette somme pondérée pour produire les sorties de ladite couche. La Figure 11 illustre les fonctions d'activation les plus utilisées.

$$y = f(x)$$

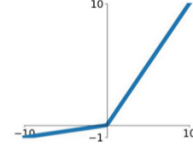
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



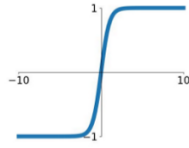
Leaky ReLU

$$\max(0.1x, x)$$



tanh

$$\tanh(x)$$

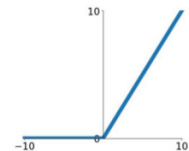


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ReLU

$$\max(0, x)$$



ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$

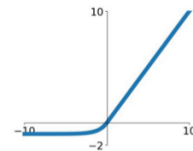


Figure 11 : Exemples de fonctions d'activation

La propagation est effectuée pour le reste du réseau en se basant sur ces deux dernières formules jusqu'à obtenir la sortie finale du réseau.

B) Apprentissage d'un réseau de neurones

L'apprentissage est effectué en trois étapes. La première étape est la propagation où les données d'entrée sont propagées dans le réseau comme décrit ci-dessus.

Ensuite, on évalue dans quelle mesure nous sommes proches de l'objectif à l'aide d'une fonction d'erreur E . L'une des plus courantes est l'erreur quadratique :

$$E(y_{\text{sortie}}, y_{\text{cible}}) = \frac{1}{2}(y_{\text{sortie}} - y_{\text{cible}})^2$$

Enfin, l'étape de rétropropagation décide dans quelle mesure chaque pondération du réseau doit être mise à jour selon l'erreur calculée. Le gradient de l'erreur (dérivée) est calculé et propagé jusqu'à la couche d'entrée. Les poids sont mis à jour selon le gradient de l'erreur de chaque neurone :

$$w_{ij} = w_{ij} - \alpha(dE/dw_{ij})$$

où α est une constante positive, appelée taux d'apprentissage. Elle a pour but d'ajuster la mise à jour des poids.

C) Les réseaux de neurones convolutionnels

Les réseaux de neurones convolutionnels forment une approche de Deep Learning qui repose sur des filtres de convolution. Un filtre est une matrice de poids utilisée pour scanner et analyser une donnée. Ces réseaux s'inspirent de la perception visuelle dans le cerveau humain.

Dans le contexte des images, chaque couche applique un filtre pour identifier des patterns ou des éléments spécifiques. Les premières couches détectent les principaux attributs, tandis que les dernières couches repèrent les détails les plus subtils et les organisent en

éléments concrets. Ainsi, ces réseaux convolutifs sont en mesure d'identifier des attributs de haut niveau sémantique, comme la forme des pupilles ou la distance entre le nez et les yeux dans une tâche de reconnaissance de visage.

Lorsqu'une matrice de données (exp. paramètres acoustiques MFCC) de taille $L \times M$ est présentée comme entrée à une couche de convolutions, un filtre (ou kernel) de taille $n \times n$ parcourt toute la surface de la matrice comme une fenêtre glissante. A chaque pas du filtre, les nombres (réels) du bloc de données de taille $n \times n$ sont combinés linéairement en les multipliant par les poids du filtre. A la fin du parcours, une nouvelle matrice formée par les sommes pondérées précédemment calculées est générée. Enfin, une fonction d'activation peut être appliquée sur chacun de ces nombres. Le résultat est appelé « Channel ». Si on applique x filtres à une donnée, on obtient x channels en sortie. Ces channels pourront être présentées en entrée à la couche suivante.

On applique souvent sur les channels en sortie une opération de « Pooling », dont le fonctionnement est semblable aux filtres, afin de réduire leur taille en sélectionnant les

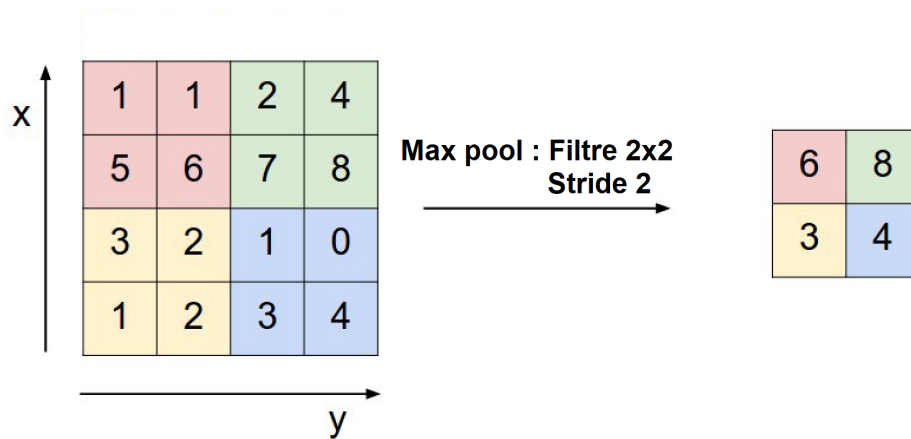


Figure 12 : Exemple d'application d'un Max Pooling

informations les plus importantes. Il y a plusieurs types de Pooling, les deux méthodes les plus utilisées sont le Max Pooling et le Average Pooling. Le Max Pooling (comme celui schématisé dans la Figure 12) consiste à extraire, pour chaque bloc de taille $n \times n$, la valeur la plus élevée tandis que le Average Pooling en prend la moyenne.

La succession de couches de convolutions et de Pooling est suivie par une/des couche(s) entièrement connectées (fully connected). Avant la première de ces couches, un « aplatissement » des channels de sortie de la dernière couche de convolution doit être effectué afin de construire un vecteur d'entrée d'une seule dimension (voir la Figure 13). La particularité d'une couche entièrement connectée est que ses neurones sont connectés à tous les neurones de la couche précédente mais aussi à tous les neurones de la couche suivante.

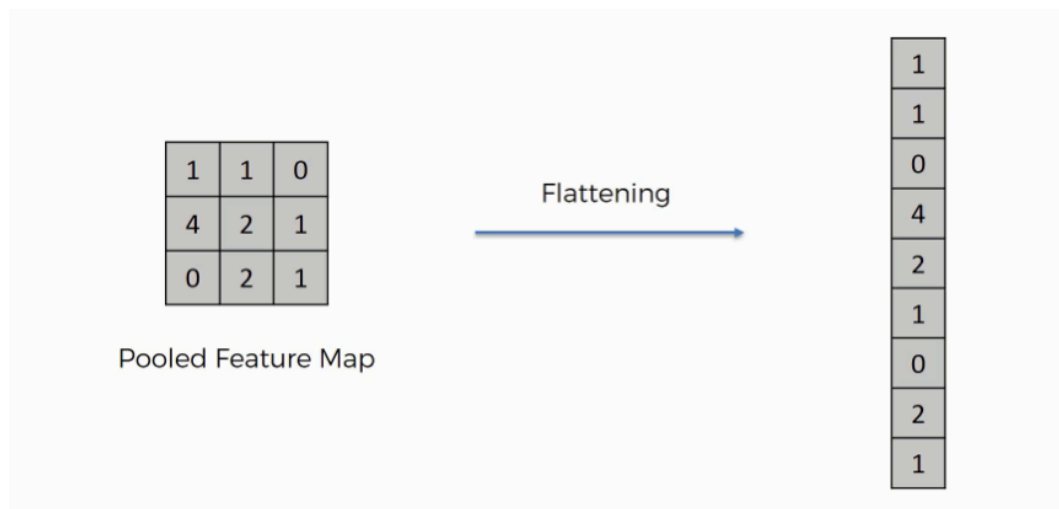


Figure 13 : Aplatissement des channels pour une couche complètement connectée

Dans une tâche de classification, une couche de Softmax est souvent utilisée pour produire une valeur de probabilité pour les différentes classes. Chaque classe aura une valeur réelle, appelée aussi « score », comprise entre 0 et 1.

3.e. Décision et évaluation :

Les contremesures produisent un score qui simule la probabilité que l'enregistrement soit authentique. A l'instar des systèmes d'ASV, la décision entre les deux classes (genuine et spoof) est effectuée en fixant un seuil sur le score produit.

Métriques d'évaluation

On peut dessiner la distribution des scores produits par le système de détection de la parole rejouée pour chacune des classes de référence, à savoir, genuine et spoofed (voir la Figure 14). Plus le système est capable de distinguer correctement entre les deux classes, moins les deux distributions sont chevauchées (moins d'incertitude).

Il y a deux types d'erreurs :

- False Rejection (FR) : Le système considère, à tort, qu'un enregistrement est rejoué.
- False Acceptance (FA) : Le système considère, à tort, qu'un enregistrement est authentique.

Pour évaluer la performance d'un système de détection de la parole rejouée, les métriques d'évaluation les plus connues sont :

- False Rejection Rate (FRR) : La proportion des FR qui correspond à la distribution cumulative (des enregistrements authentiques) à gauche du seuil.
- False Acceptation Rate (FAR) : la proportion des FA qui correspond à la distribution cumulative (des enregistrements usurpés) à droite du seuil.

- Equal Error Rate (EER) : La valeur du FRR et FAR quand ils sont égaux, c'est-à-dire, quand on choisit un seuil qui passe par l'intersection des deux courbes [8].

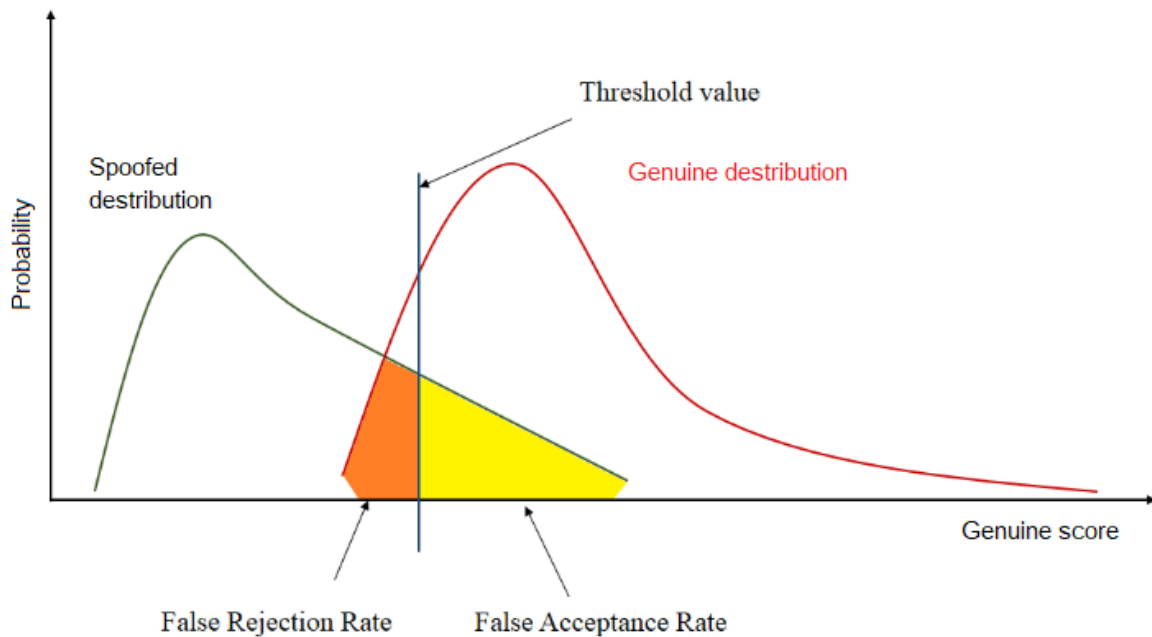


Figure 14 : Scores des classes et métriques d'évaluation d'un système de détection de la parole rejouée

4. Expériences et résultats

Le système que j'ai réalisé a pour but de prédire si un enregistrement audio est authentique (genuine) ou usurpé par parole rejouée (spoo). La construction de ce système est composée de trois étapes fondamentales : prétraitement, apprentissage et évaluation.

Pour ce faire, j'ai utilisé un corpus de données préparé dans le cadre du défi de contremesures pour l'usurpation de la voix (ASVspoof2017). Ce corpus est composé de fichiers audios en format « Wav ». Ils sont répartis sur trois groupes. La première partie est dédiée à l'apprentissage, la deuxième est dédiée à la phase de développement et la troisième sert à évaluer les modèles appris. Le taux d'échantillonnage de ces fichiers audio est de 16 kHz et ils sont représentés sous des entiers de 16 bits.

4.a. Prétraitements

Les fichiers du corpus de données sont annotés selon la classe (genuine/spoo). J'ai utilisé 13306 fichiers pour l'apprentissage et 3016 pour l'évaluation.

En première étape, j'ai construit pour chaque partie un fichier contenant deux colonnes, le « base name » et le chemin du fichier « path ». Exemple pour la partie train :

D_1001353 /users/trigui/dataSetTrain/D_1001353.wav

J'ai réalisé ceci à l'aide de la commande suivante :

```
ls | cut -f1 -d. | awk '{print $1" /users/trigui/dataSetTrain/"$1".wav"}' > train.path
```

Les fichiers « train.path » et « eval.path » seront utilisés ensuite pour la lecture des données sous python. La liste des couples « baseName/Path » sera convertie en une structure de données de type dictionnaire « *dict_wav_path* » qui sera enregistrée temporairement dans la mémoire, afin d'accélérer la lecture des données.

Extraction des paramètres MFCCs :

La première étape dans la tâche de détection de parole jouée est l'extraction des caractéristiques acoustiques de type MFCC (dans notre cas).

Pour ce faire, j'ai appliqué trois étapes :

- 1- Lecture du fichier wav sous forme d'un tableau numpy d'entiers (signal) accompagné de la fréquence d'échantillonnage Fs (sampling frequency) à l'aide de la librairie « SciPy » :

```
fs, signal = wav.read(dict_wav_path[key])
```

- 2- Application d'une pré-amplification (augmenter l'amplitude des hautes fréquences, voir la Figure 15 et la Figure 16) à l'aide de la librairie « SpeechPy » :

```
signal = speechpy.processing.preemphasis(signal, cof=0.97)
```

- 3- Extraction des paramètres du signal « mfcc » à l'aide de la librairie SpeechPy :

```
feat = speechpy.feature.mfcc (signal, sampling_frequency=fs, frame_length=0.020, frame_stride=0.01, num_cepstral=23, ...)
```

Les paramètres « MFCC » obtenus sont représentés sous la forme d'un tableau « NumPy » bidimensionnel de taille (nombre de trames * nombre de coefficients cepstraux).

Les paramètres « MFCC » d'un fichier « wav » sont enregistrés sur le disque sous une forme compressée (npz) à l'aide de la librairie NumPy :

```
np.savez(os.path.join(destinationPath, file_name), feature)
```

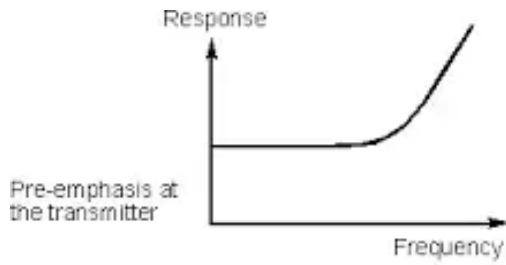


Figure 16 : Pré-amplification d'un signal

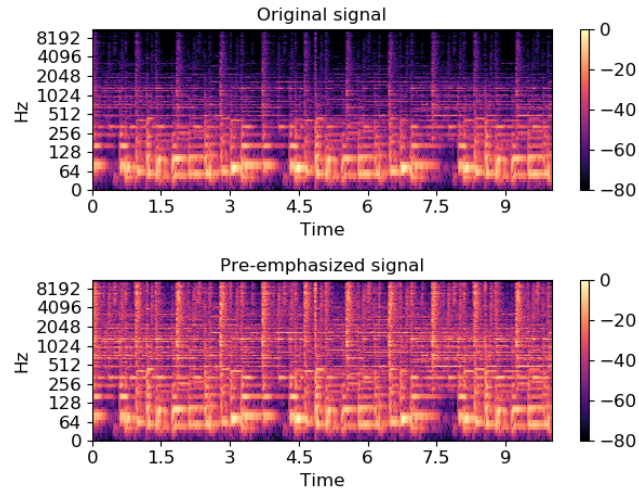


Figure 15 : Résultat de la pré-amplification sur les hautes fréquences

Préparation des entrées (MFCC) au réseau de neurones :

La bibliothèque « Pytorch » offre des fonctionnalités de construction et d'apprentissage de réseaux de neurones ainsi que des fonctionnalités de chargement de données. Pour nourrir la couche d'entrée du réseau, j'ai créé un objet de type « DataLoader » défini par la taille du batch (nombre de matrices mfcc à traiter en même temps) et l'option shuffle (qui permet de mélanger les données).

```
dl = torch.utils.data.DataLoader(DataSetObject, batch_size, shuffle=True)
```

L'objet DataLoader prend aussi un objet de type DataSet. Ce dernier permet principalement de charger une matrice MFCC (à partir d'un fichier npz) à l'aide de la fonction « getItem ».

```
def __getitem__(self, index):
    baseName = self.list_baseNames[index]
    dataPath = self.dictBaseName_Path[baseName]
    label = self.dictBaseName_LabelsFile[baseName]
    fileLoaded = np.load(dataPath)
    mfcc_Loaded = fileLoaded[ 'arr_0' ]
    if self.transform:
        mfcc = self.transform( mfcc_Loaded )
    return ((mfcc, label))
```


La fonction « getItem » fait appel à l'objet de type Compose (une classe qui contient un tableau de transformations). Ce dernier combine les transformations dont on a besoin (paddingTransform et TotensorTransform). Ce besoin est dû au fait que la couche d'entrée d'un réseau de neurone n'accepte que des données de même structure (taille et format). Ces transformations produisent des paramètres MFCC(s) ayant la longueur du plus long fichier wav (padding) et les convertissent en des tensors. Le padding consiste à

ajouter des zéros aux frontières des matrices de taille inférieur à la taille maximale (zero-padding) (Exemple de MFCCs : voir la Figure 17).

```
class paddingTransform (object):
    def __init__(self, nbFrameMax):
        self.nbFrameMax = nbFrameMax
    def __call__(self, mfcc):
        nbFrame = mfcc.shape[0]
        if nbFrame < self.nbFrameMax:
            nbFrameWithZeroToAdd = self.nbFrameMax - nbFrame
```

```
class TotensorTransform (object):
    def __call__(self, mfcc):
        mfcc_tensor = torch.from_numpy(mfcc).float()
        return (mfcc_tensor)
```

```
T = transforms.Compose ([ paddingTransform, toTensorTransform ])
DS = mfccDataset(dict_wav_npzFilePath, dict_wav_label, transform= T)
```

 trigui@myvoiceai: /users/trigui

```
[[[ 1.9715e+01,  8.3356e+00,  1.7452e+00, ..., -1.6047e+00,
    1.4719e-01, -6.1251e-01],
  [ 1.9965e+01,  7.1807e+00,  8.9180e-01, ..., -1.2663e+00,
    -2.4206e-01, -9.1934e-01],
  [ 2.0178e+01,  6.6822e+00,  8.1380e-01, ..., -1.1706e+00,
    -1.4880e+00,  6.3907e-03],
  ...,
  [ 0.0000e+00,  0.0000e+00,  0.0000e+00, ...,  0.0000e+00,
    0.0000e+00,  0.0000e+00],
  [ 0.0000e+00,  0.0000e+00,  0.0000e+00, ...,  0.0000e+00,
    0.0000e+00,  0.0000e+00],
  [ 0.0000e+00,  0.0000e+00,  0.0000e+00, ...,  0.0000e+00,
    0.0000e+00,  0.0000e+00]]],

[[[ 2.4306e+01,  7.6847e+00,  2.1266e+00, ...,  3.3111e-01,
    1.8897e-01,  1.9033e-01],
  [ 2.4944e+01,  7.5087e+00,  2.0195e+00, ...,  6.8408e-02,
    1.4562e-01,  2.7570e-01],
  [ 2.4751e+01,  6.4603e+00,  1.0407e+00, ..., -6.8974e-01,
    -1.0668e+00, -1.1031e+00],
  ...,
  [ 0.0000e+00,  0.0000e+00,  0.0000e+00, ...,  0.0000e+00,
    0.0000e+00,  0.0000e+00],
  [ 0.0000e+00,  0.0000e+00,  0.0000e+00, ...,  0.0000e+00,
    0.0000e+00,  0.0000e+00],
  [ 0.0000e+00,  0.0000e+00,  0.0000e+00, ...,  0.0000e+00,
    0.0000e+00,  0.0000e+00]]],
```

Figure 17 : Un échantillon de MFCCs

4.b. Modélisation

Pour construire mon système de détection de la parole rejoué, j'ai utilisé le modèle LCNN. Ce dernier a été construit pour une tâche de reconnaissance d'images avec des données bruitées (le corpus de données MS-Celeb-1M) [Xiang Wu].

L'architecture du modèle Light CNN :

Light CNN est une extension du réseau de neurones convolutionnel. La contribution principale consiste à l'ajout d'une opération de Maximum Feature Map (MFM) qui permet de réduire le nombre de paramètres dans le réseau.

MFM est considérée aussi comme une fonction d'activation qui supprime les neurones de faible activation. Le fonctionnement de l'opération MFM est schématisé dans la Figure 18. A la sortie d'une couche de convolution produisant N channels, une couche de MFM répartit ces dernières en deux groupes. Chaque groupe contient $N/2$ channels. Ensuite, elle applique une activation Maxout sur ces deux groupes pour produire un seul groupe de $N/2$ channels en sortie [9].

L'architecture du LCNN représentée dans le tableau 1 est constituée de neuf couches de convolutions. Sur chacune de ces couches une opération de MFM est appliquée. Le réseau intègre une couche de maxPooling après la première couche MFM puis après chaque bloc de deux couples de couches de convolution-MFM [13].

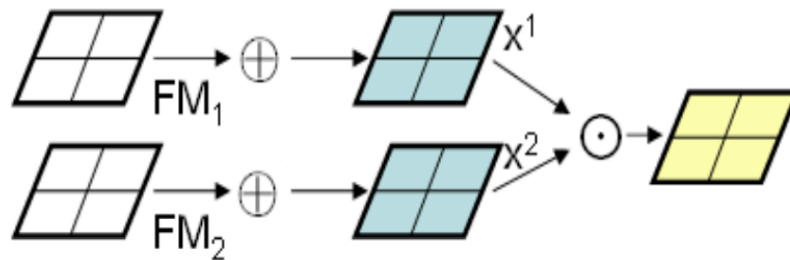


Figure 18 : Illustration de l'opération MFM

La sortie de cette partie de réseau est aplatie (flatten), cela permet la transformation de ce dernier d'un espace de trois dimensions à un espace de deux dimensions (vecteur). A partir du corpus de données MSCeleb-1M, la taille de ce vecteur est de 8192 ($8 \times 8 \times 128$). En utilisant notre corpus ASVspoof, la taille du vecteur devrait être 4352. J'ai adapté l'entrée de la couche suivante avec cette nouvelle taille.

Une couche entièrement connectée de 256 neurones avec une fonction linéaire prend la sortie de ce réseau. Enfin, une couche de sortie entièrement connectée produit un vecteur de taille 2 (deux neurones) sous un encodage one-hot.

Opération	Taille filtre	Stride	Padding	Nb chaîne Sortie
Convolution1	5x5	1	2	96
MFM 1	-	-	-	48
Pooling 1	2x2	2	-	48
Convolution 2	1x1	1	-	96
MFM 2	-	-	-	48
Convolution 3	3x3	1	1	192
MFM 3	-	-	-	96
Pooling	2x2	2	-	96
Convolution 4	1x1	1	-	192
MFM 4	-	-	-	96
Convolution 5	3x3	1	1	384
MFM 5	-	-	-	192
Pooling	2x2	2	-	192
Conv 6	1x1	1	-	384
MFM 6	-	-	-	192
Convolution 7	3x3	1	1	256
MFM 7	-	-	-	128
Convolution 8	1x1	1	-	256
MFM 8	-	-	-	128
Convolution 9	3x3	1	1	256
MFM 9	-	-	-	128
Pooling	2x2	2	-	128

Tableau 1 : L'architecture du modèle LCNN

Apprentissage :

Le réseau est appris en se basant sur l'entropie croisé (CrossEntropyLoss) comme fonction de coût et sur la descendante de gradient stochastique comme algorithme d'optimisation avec un taux d'apprentissage dynamique (qui change au fil des epochs) qui commence par la valeur 0,01.

Test :

Dans cette partie, j'ai introduit le corpus de test au réseau afin de prédire les deux valeurs de sortie. On va se baser sur ces deux valeurs dans ce qui suit pour analyser les résultats.

Le résultat est formé d'un vecteur de scores (une estimation du probabilité) de deux colonnes, une colonne représente les scores (probabilité) de l'hypothèse spoof et la seconde colonne représente les scores de l'hypothèse genuine. Ces deux colonnes sont complémentaires (somme égale à 1). J'ai donc considéré une seule colonne pour l'analyse, à savoir, la colonne genuine.

En premier temps, j'ai réparti les scores de cette colonne en deux sous-ensembles en fonction de la classe de référence (la valeur réelle). Ensuite, j'ai présenté la distribution des scores de chaque sous-ensemble en appliquant une fonction gaussienne. Les deux distributions gaussiennes des probabilités (score) de l'hypothèse genuine sont représentées dans la Figure 19.

La courbe rouge représente la distribution des scores des enregistrements qui sont réellement de classe spoof. La courbe verte représente celle des enregistrements qui sont réellement de la classe genuine.

On peut constater que ces deux courbes se chevauchent d'une manière très importante. Donc c'est très difficile de choisir un seuil de décision qui permet de déterminer la classe de chaque enregistrement en fonction de son score. On remarque aussi que la courbe genuine se trouve légèrement à gauche de la courbe spoof. Ceci indique que les scores de la classe de référence genuine sont en moyenne plus petit que ceux de la classe de référence spoof alors qu'on attend l'inverse puisque l'axe x représente les scores (probabilité) de l'hypothèse genuine.

Les résultats obtenus ne sont pas consistants et on n'a pas trouvé une explication concrète de l'origine de cette anomalie. En revanche, on a constaté que le nombre des paramètres est très grand (40 millions). Ceci pourrait être la cause de ce problème car un réseau pourrait être surdimensionné relativement à la taille des données et à la complexité de la tâche. Pour réduire le nombre de paramètres, j'ai suivi deux pistes. La première consiste à diminuer le nombre de couches de convolution et la deuxième consiste à diminuer leur taille.

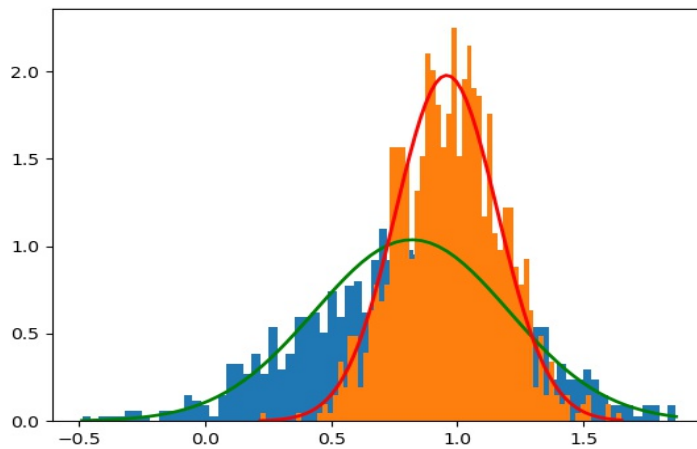


Figure 19 : Distribution des scores avec le modèle initial

Pour la première idée (Figure 20), j'ai enlevé les deux dernières couches pour obtenir un réseau de 7 couches au lieu de 9. Pour la seconde idée (Figure 21), j'ai réduit le nombre de filtres dans les couches de convolution et donc le nombre de channels produits.

A partir de la Figure 20 et la Figure 21, on peut constater que la répartition des scores produits par les deux modèles est meilleure que le modèle initial. En effet, la courbe spoof se trouve maintenant à gauche de la courbe genuine. Le mauvais positionnement relatif des deux distributions a été résolu. On peut remarquer aussi que le chevauchement des deux courbes a beaucoup diminué par rapport au premier modèle. Ceci confirme notre hypothèse sur l'impact de la taille de l'architecture sur la consistance des scores produits.

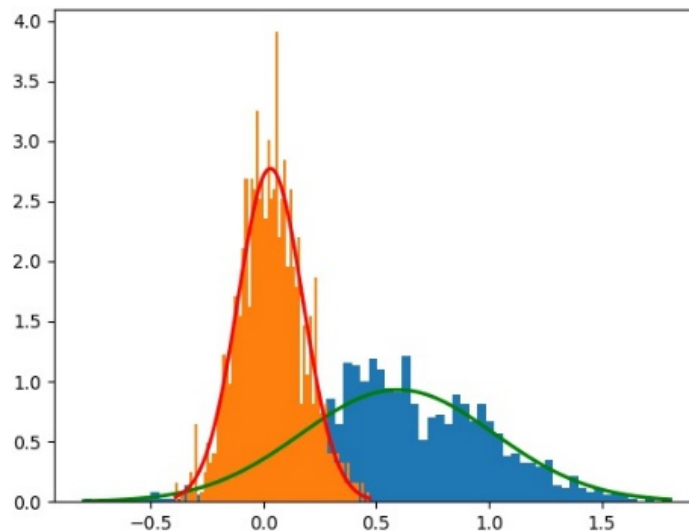


Figure 20 : Distribution des scores avec 7 couches de convolution

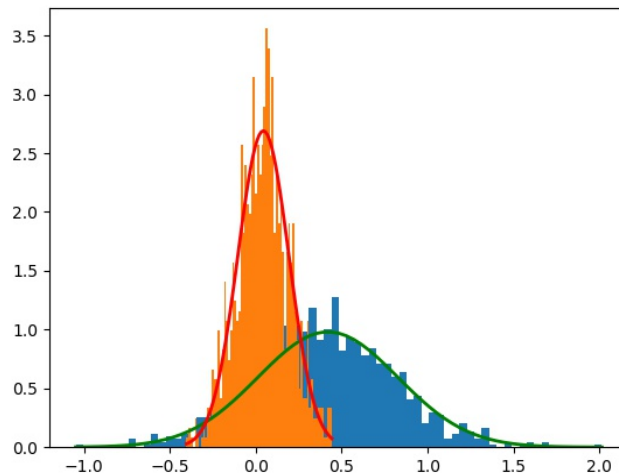


Figure 21 : Distribution des scores en réduisant le nombre de filtres

A ce stade, je me suis concentrée sur l'analyse de la distribution du score (production d'un résultat préliminaire et interprétable). La suite de ce travail portera sur l'interprétation des résultats selon les métriques objectives connus, à savoir, le FAR, le FER et le EER. Ceci fait partie des perspectives de ce stage.

5. Conclusion :

Je suis très ravie d'avoir eu l'opportunité de réaliser un stage au sein de la startup « MY Voice IA ». Le cadre professionnel et l'ambiance m'ont toujours encouragée à faire de mon mieux et à dépasser les problèmes que j'ai rencontrés. Dans ce stage, j'ai eu la chance de travailler avec une équipe d'ingénieurs-chercheurs, d'en apprendre l'esprit de résolution de problèmes et d'appliquer ce dernier sur un sujet contemporain, à savoir la détection de la parole rejouée.

En lisant des articles scientifiques en anglais, j'ai pu formuler un état de l'art sur l'usurpation d'identité dans le cadre de l'ASV avec les contremesures proposées dans la littérature. J'ai ainsi découvert la révolution qu'a offerte l'apprentissage profond durant ces dernières années notamment avec les réseaux de neurones convolutifs.

Je me suis intéressée dans ce travail à une extension récente du modèle CNN, nommée Light CNN, proposée initialement pour une tâche de traitement d'images. J'ai adapté cette approche à notre tâche ce qui m'a donné l'occasion de manipuler pour la première fois des données audios. Ce stage m'a donc permis d'approfondir mes connaissances académiques dans le domaine de l'intelligence artificielle en acquérant des aspects théoriques ainsi que des aspects pratiques de Deep Learning. J'ai exploité par exemple des fonctionnalités des bibliothèques NumPy, SciPy et SpeechPy pour la manipulation de mes données. J'ai également utilisé la librairie Pytorch pour concevoir et entraîner des modèles

LCNN sur des cartes graphiques. J'ai ensuite adapté les hyperparamètres du modèle LCNN afin d'obtenir de meilleurs résultats.

La durée de ce stage ne m'était pas suffisante d'achever l'évaluation du système appris. Cela reste parmi mes prochaines perspectives au sein de My Voice AI.

Références

- [1] : <https://www.totalvoicetech.com/difference-between-voice-recognition-and-speech-recognition/>
- [2] : <http://www.eurecom.fr/~evans/papers/pdfs/4436.pdf>
- [3] : <https://fr.wikipedia.org/wiki/Cepstre#MFCC>
- [4] : <http://www.epsic.ch/cours/electronique/techn99/acous/AQSIGNAL.html>
- [5] : <https://pdfs.semanticscholar.org/1c8b/4e300f86555b514ec1e0c3fd58763524c2ac.pdf>
- [6] : https://fr.wikipedia.org/wiki/Apprentissage_automatique
- [7] : <https://www.futura-sciences.com/tech/definitions/informatique-reseau-neuronal-601/>
- [8] : https://hal.archives-ouvertes.fr/hal-00990617/file/InTech-Evaluation_of_biometric_systems.pdf
- [9] : <https://arxiv.org/pdf/1511.02683.pdf>
- [10] : <https://blog.groupe-sii.com/ral/>
- [11] : <http://hectordelgado.me/wp-content/uploads/Delgado2018c.pdf>
- [12] : <https://arxiv.org/pdf/1511.02683.pdf>
- [13] : <https://github.com/AlfredXiangWu/LightCNN>
- [Wu et al.] : Wu, Z., Evans, N., Kinnunen, T., Yamagishi, J., Alegre, F., & Li, H. (2015). Spoofing and countermeasures for speaker verification: A survey. *speech communication*, 66, 130-153.
- [Lindberg J] : Lindberg, J., & Blomberg, M. (1999). Vulnerability in speaker verification-a study of technical impostor techniques. In *Sixth European Conference on Speech Communication and Technology*.
- [Villalba et Lleida] : Villalba, J., Lleida, E., 2011a. Detecting replay attacks from far-field recordings on speaker verification systems, in: Vielhauer, C., Dittmann, J., Drygajlo, A., Juul, N., Fairhurst, M. (Eds.), *Biometrics and ID Management*. Springer. *Lecture Notes in Computer Science*, pp. 274–285.
- [Galina Lavrentyeva] : Lavrentyeva, G., Novoselov, S., Malykh, E., Kozlov, A., Kudashev, O., & Shchemelinin, V. (2017, August). Audio Replay Attack Detection with Deep Learning Frameworks. In *Interspeech* (pp. 82-86).
- [Wang et al.] : Wang, Z. F., Wei, G., & He, Q. H. (2011, July). Channel pattern noise based playback attack detection algorithm for speaker recognition. In *2011 International conference on machine learning and cybernetics* (Vol. 4, pp. 1708-1713). IEEE.
- [Xiang Wu] : Wu, X., He, R., Sun, Z., & Tan, T. (2018). A light cnn for deep face representation with noisy labels. *IEEE Transactions on Information Forensics and Security*, 13(11), 2884-2896.
- [Davis et Mermelstein, 1980] : S. Davis & P. Mermelstein, 1980. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE transactions on acoustics, speech, and signal processing* 28(4), 357–366.
- [AJILI, 2017] : Ajili, M. (2017, November). Reliability of voice comparison for forensic applications. Avignon.

[Hansen et Hasan., 2015] : Hansen, J. H., & Hasan, T. (2015). Speaker recognition by machines and humans: A tutorial review. IEEE Signal processing magazine, 32(6), 74-99.