

Résumé

Table des matières

Introduction générale	1
1 État de l'art	2
1.1 l'anatomie du discours	3
1.2 Extraction des caractéristiques de la voix	4
1.2.1 Objectifs de l'extraction des caractéristiques	4
1.2.2 Comparaison entre quelques techniques existantes	4
1.3 l'apprentissage profond	5
1.3.1 Du cerveau au réseau de neurones	5
1.3.2 Les réseaux de neurones profonds	6
1.3.3 Différentes architectures de l'apprentissage profond	7
2 Choix de la solution	8
2.1 Extraxtion des caractéristiques avec MFCC	9
2.1.1 Préaccentuation	10
2.1.2 Fenêtrage	10
2.1.3 Pondération par une fenêtre de Hamming	10
2.1.4 Estimation du spectre	11
2.1.5 Bancs du filtre Mel	11
2.1.6 Calcul du logarithme de chaque énergie	12
2.1.7 Calcul de la transformé en cosinus discrète inverse	13
2.1.8 Normalisation moyenne	13
2.1.9 Filtre bank VS MFCC	14
2.2 L'architecture CNN	14
2.2.1 Couche de convolution	14
2.2.2 Couche de regroupement "Pooling layer"	15
2.2.3 Couche d'activation	16
3 Simulation Et Réalisation	17
3.1 Base de données	18
3.2 Paramétrage	18
3.2.1 Les coefficients MFCCs	18
3.2.2 La fonction de perte "Loss Function"	18
3.2.3 L'optimiseur "Optimizer"	19

3.3	Etude comparative	20
3.3.1	Variation des couches de l'architecture	20
3.3.1.1	Architecture 1	20
3.3.1.2	Architecture 2	20
3.3.1.3	Architecture 3	21
3.3.1.4	Architecture 4	22
3.3.2	Variation des paramètres	22
3.3.2.1	Variation du nombre des époques	23
3.3.2.2	Variation du batch size	24
3.3.2.3	Variation du taux d'apprentissage "learning rate"	25
3.3.2.4	Variation de la taille du filtre	26
3.3.2.5	Variation de la fonction de perte	27
3.3.2.6	Variation de la valeur du Dropout	27
3.3.3	Architecture finale	28
3.4	Interface graphique	30
Conclusion Générale et Perspectives		32
Bibliographie		33

Table des figures

1.1	L'appareil vocal humain	3
1.2	Neurone humaine	5
1.3	Allure d'une architecture en deep learning	6
2.1	Signal d'entrée	9
2.2	Etapes d'extraction des caractéristiques	9
2.3	Signal après la préaccentuation	10
2.4	Fenêtre de Hamming	10
2.5	Les fréquences dans l'oreille humaine	11
2.6	Allure du filtre bank dans l'échelle mel	12
2.7	Application du filtre bank	12
2.8	Mel-spectrogram du signal	12
2.9	MFCCs	13
2.10	Filtre bank normalisé	13
2.11	MFCCs normalisés	13
2.12	Exemple de l'application d'une couche de convolution	14
2.13	Architecture CNN avec des convolutions	15
2.14	Exemple de l'application d'un max pooling	15
2.15	Liste de quelques fonctions d'activation	16
3.1	L'organisation de la base de données	18
3.2	Architecture finale	29
3.3	Interface graphique	30
3.4	Importer un fichier audio	30
3.5	Afficher le spectrogramme MFCC du fichier audio importé	30
3.6	Afficher le résultat de la prédiction	31
3.7	Afficher les probabilités de la prédiction	31

Liste des tableaux

1.1	Comparaison entre les techniques d'extraction des caractéristiques	4
1.2	Comparaison entre les architectures de l'apprentissage profond	7
3.1	Architecture 1 : Comparaison mel-spectrogram VS MFCC	20
3.2	Architecture 2 : Comparaison mel-spectrogram VS MFCC	21
3.3	Architecture 3 : Comparaison mel-spectrogram VS MFCC	21
3.4	Architecture 4 : Comparaison mel-spectrogram VS MFCC	22
3.5	Tableau comparatif : variation du nombre des époques	23
3.6	Tableau comparatif : variation du batch size	24
3.7	Tableau comparatif : variation du taux d'apprentissage	25
3.8	Tableau comparatif : variation de la taille du filtre	26
3.9	Tableau comparatif : variation de la fonction de loss	27
3.10	Tableau comparatif : variation de la valeur du Dropout	28

Introduction générale

Le cerveau est l'organe le plus incroyable du corps humain. Il dicte la façon dont nous percevons chaque vue, son, odeur, goût et toucher. Il nous permet de stocker des souvenirs, éprouver des émotions, et même rêver. Sans lui, nous serions des organismes primitifs, incapables de rien d'autre que le plus simple des réflexes. Le cerveau est ce qui nous rend intelligents.

Pendant des décennies, nous avons rêvé de construire des machines intelligentes avec des cerveaux comme le nôtre(assistants robotiques qui nettoie nos maisons, les voitures qui conduisent eux-mêmes, les microscopes qui détecte automatiquement les maladies..). Mais construire ces machines artificiellement intelligentes nous oblige à résoudre certains des problèmes de calcul les plus complexes que nous ayons jamais rencontrés; problèmes que notre cerveau peut déjà résoudre en quelques microsecondes. Pour résoudre ces problèmes, nous devons utiliser des techniques largement développées au cours de la dernière décennie.c'est un domaine extrêmement actif de l'intelligence artificielle, souvent appelé "l'apprentissage profond".

La reconnaissance vocale est une preuve de l'intelligence. Elle est devenue un domaine de recherche populaire depuis le premier brevet international [1]qui a été déposé en 1983 par Michele Cavazza et Alberto Ciaramella dans le cadre de la recherche sur les télécommunications de CSELT (Italie) pour servir à la fois les futurs services de télécommunications aux clients finaux et améliorer les techniques de réduction du bruit. Dès lors, plusieurs applications ont été mise allant de la sécurité à l'indexation multimédia. Pour la sécurité et le contrôle d'accès, tout d'abord, elle peut servir à contrôler l'accès physique à des bâtiments sensibles (banques, entreprises) ou à un véhicule, ou l'accès à distance avec par exemple la consultation de comptes bancaires par téléphone , mais en raison des performances limitées de cette reconnaissance, on l'utilisera préférentiellement en complément d'un code et d'un badge ou dans des situations moins sensibles. Une application potentielle est liée à la police criminelle et à l'identification de suspects. Le fait que la voix ne soit pas de nature purement biométrique limite là encore le champ d'application, et la communauté scientifique du traitement automatique de la parole appelle à la prudence dans ce cadre d'utilisation.

Enfin, la reconnaissance du locuteur est aussi un élément très utile d'amélioration des systèmes de transcription automatique de la parole, et permet la structuration en tours de paroles d'un flux audio pour l'indexation de documents multimédia.

Chapitre 1

État de l'art

Introduction

Dans ce chapitre nous allons détailler le concept de production de la parole, les techniques existantes pour extraire les caractéristiques de la voix de chaque personne et les différentes architectures présentées par l'apprentissage profond.

1.1 l'anatomie du discours

Pour tous, la voix est considérée comme le moyen direct et privilégié de communication avec l'autre : elle va refléter autant l'état du corps que celui de l'esprit : c'est une image sonore de nous-mêmes avec laquelle nul ne peut tricher. C'est notre signature sonore, au cœur de notre personnalité, qui permet de nous identifier. Mais l'utilisation du terme d'empreinte vocale laisse croire que la voix présenterait des caractéristiques aussi fiables que les empreintes digitales ou génétiques : il n'en est rien.

La parole a un mécanisme de production spécial dans l'appareil vocal de l'être humain, ce qui le distingue du reste des signaux audio en général. L'appareil vocal humain, comme le montre la figure 1, est divisé en trois parties principales : les poumons, le larynx et les cavités vocales. [5]

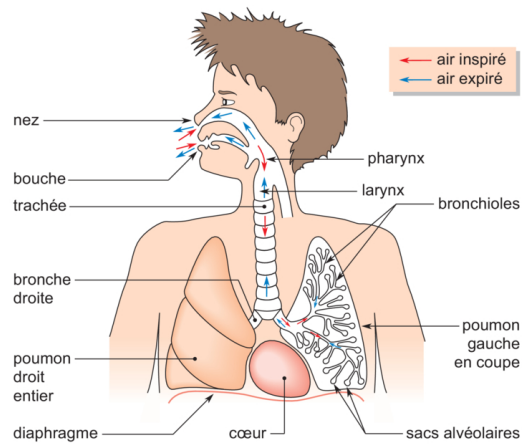


FIGURE 1.1 – L'appareil vocal humain

Ceux-ci contiennent en outre de nombreux différents organes et chaque organe a une fonction spécifique.

Le processus de production de la parole commence à partir des poumons. Ces derniers génèrent un flux d'air, qui après avoir traversé la trachée, ressemble à un bruit blanc.

Aux cordes vocales du larynx, la pression de l'air est accumulée, ce qui se traduit par une ouverture glottale et une vibration des cordes vocales. Les cordes vocales vibrent à une fréquence appelée fréquence fondamentale, f_0 et génèrent la série de bouffées d'air ou d'impulsions séparées par la période de tangage « pitch period » (inverse de la fréquence fondamentale).

Ces impulsions d'air après le passage à travers les cavités nasales, orales et buccales créent un son perceptible appelé discours.

C'est dû aux propriétés structurelles spécifiques de l'appareil vocal, le discours porte des informa-

tions relatives à la personne en tant que caractérisation intrinsèque.

Dernier élément à considérer : les paramètres « physiques » de la voix. Il y en a trois :

- **La fréquence ou hauteur de la voix**
- **L'intensité**
- **Le timbre**

1.2 Extraction des caractéristiques de la voix

1.2.1 Objectifs de l'extraction des caractéristiques

La variation de la nature du signal acoustique rend le traitement des données brutes issues de ce dernier très difficile. En effet, ces données contiennent des informations complexes, souvent redondantes et mélangées à du bruit.

De plus, la performance d'un système de reconnaissance vocale dépend fortement de la façon de la représentation de la parole en entrée du système. De ce fait, plusieurs efforts ont été faits pour modéliser le système de production de la parole et distiller les caractéristiques acoustiques sensibles aux variations des enceintes fonctionnalités telles que MFCC, PLP, RAST, LPCC, PCA, LDA, Wavelet, DTW mais surtout utilisé est MFCC.

L'extraction de caractéristiques, en résumant, transforme le son entrant en une représentation interne telle qu'il est possible de reconstruire le signal d'origine à partir de celui-ci.

1.2.2 Comparaison entre quelques techniques existantes

Méthode	propriétés	Utilisation
MFCC	Le spectre de puissance est calculé en effectuant une analyse de Fourier.	Cette méthode est utilisé pour trouver nos caractéristiques vocales.
PLP	Méthode d'extraction des caractéristiques non linéaire mais rapide.	Méthode traditionnelle. Bonne pour les données gaussiennes.
LPC [4]	Méthode d'extraction des caractéristiques statique, 10 à 16 ordre de coefficient inférieur.	Utilisé pour l'extraction des caractéristiques pour un ordre de coefficient bas.

TABLE 1.1 – Comparaison entre les techniques d'extraction des caractéristiques

1.3 l'apprentissage profond

Avec l'apparition et l'évolution de l'intelligence artificielle et l'augmentation des capacités technologiques, des nombreuses recherches sont apparues afin de simuler le comportement humain et également de reproduire ses capacités cognitives. Dans ce contexte le Deep Learning ou bien "l'apprentissage profond", est l'un des axes les plus explorés de nos jours.

1.3.1 Du cerveau au réseau de neurones

Notre cerveau constitue le centre du système nerveux, [7]il est en fait capable d'intégrer plusieurs informations, de contrôler la motricité et d'assurer les fonctions cognitives. Le neurone est considéré comme l'élément central du cerveau.

Un neurone se compose de :

- un corps cellulaire
- un axone qui représente le lien de transmission des signaux
- une synapse qui permet le déclenchement d'un potentiel d'action dans le neurone pour activer une communication avec un autre neurone.

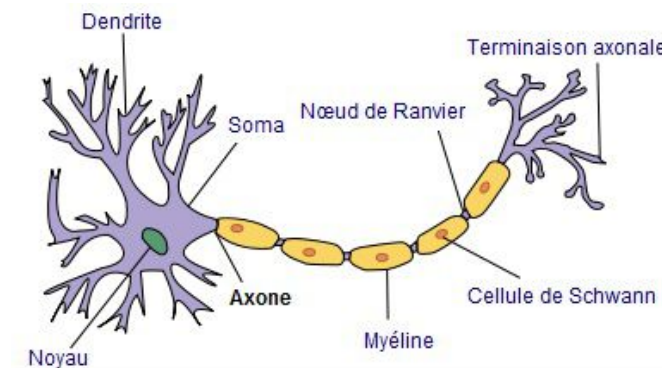


FIGURE 1.2 – Neurone humaine

La force d'un réseau de neurone se traduit par la bonne communication entre les neurones qui le constituent et qui possède chacun d'entre eux une fréquence qui influe directement sur la propagation des signaux au sein de ce réseau de neurones.

L'influx nerveux ou encore le signal électrique se propage de l'axone vers la terminaison synaptique. Plus la fréquence de celui-ci est importante, plus le neurone produit des substances chimiques : les neurotransmetteurs.

Contenus dans les vésicules, ces derniers sont libérés dans le milieu extracellulaire au niveau de la synapse. Ils vont à leur tour activer ou inhiber un second neurone au niveau de sa dendrite ou de son corps cellulaire. L'influx nerveux poursuit son chemin le long de ce second neurone et ainsi de suite.

Pour résumer et avoir un lien entre le cerveau et les réseaux de neurones artificiels nous pouvons remarquer que l'information est reçue par les dendrites, se rassemblent dans le corps de la cellule

et s'écoule vers le bas de l'axone. Chaque neurone est relié à plusieurs neurones en « entrée » (dendrites) et en « sortie » (axone). C'est à travers ce fonctionnement « entrée/traitement/sortie » que les chercheurs se sont inspirés pour reproduire un réseau de neurones de manière artificielle.

1.3.2 Les réseaux de neurones profonds

Les neurones artificiels présentent la base sur laquelle le Deep Learning est conçu. Il est basé sur des algorithmes qui permettent de traiter un ensemble de données en entrée qu'on appelle des observations pour prédire une sortie qu'on appelle une caractéristique.

Les réseaux de neurones sont composés par des milliers de neurones artificiels. [8] Ces derniers sont organisés en couches interconnectées grâce à des liens pondérés W_i . Deux éléments qui caractérisent chaque neurone qui sont :

- La fonction d'activation F_i
- La condition d'activation C_i

Le travail de l'apprentissage est donc de trouver la combinaison optimale des $W_i/F_i/C_i$ pour générer les meilleurs résultats en termes de prédiction et d'estimation.

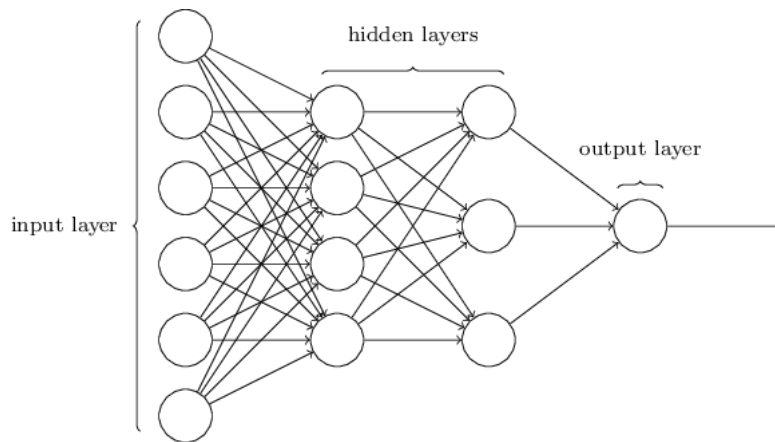


FIGURE 1.3 – Allure d'une architecture en deep learning

Pour assurer la non linéarité et entraîner le réseau et trouver les millions des poids nous devons avoir une solution qui est la « backpropagation ».

Dans la phase d'apprentissage, nous cherchons à construire un modèle qui nous servira à faire des prédictions. Comme nous l'avons déjà mentionnée, nous cherchons à trouver la meilleure combinaison $W_i/F_i/C_i$. Pour ce faire, on définit une fonction d'erreur qui calcule la différence entre la sortie réelle du réseau et sa sortie attendue après qu'un cas a circulé à travers le réseau.

Soit Y et Y' deux vecteurs respectivement la sortie réelle et la sortie attendue du réseau et l'erreur de rétropropagation est $E(Y, Y')$. On cherche alors à optimiser le modèle ($W_i/C_i/F_i$) de tel façon à minimiser $E(Y, Y')$. Dans un premier temps, les poids W_i sont initialisés aléatoirement ce qui mène à avoir une sortie qui permet de calculer $E(Y, Y')$.

Pour chaque neurone $k = 1, \dots, n$, on calcule l'erreur propre au neurone en question (k). On met à jour ensuite les poids et on propage ainsi de suite jusqu'à atteindre le seuil d'erreur préalablement fixé. On peut donc enchaîner directement vers la phase de prédiction une fois le modèle conçu.

1.3.3 Différentes architectures de l'apprentissage profond

Le nombre d'architectures et d'algorithmes utilisés dans l'apprentissage profond est vaste et varié. Cette section explore trois des architectures les plus populaires durant les 20 dernières années. Notamment, LSTM et CNN qui sont les deux approches les plus anciennes de cette liste mais aussi les plus utilisées dans diverses applications.

Ces architectures sont appliquées dans plusieurs scénarios, mais dans le tableau suivant nous allons indiquer les domaines d'application pour chacune.

Architecture	Domaines d'application
RNN [6]	Speech recognition , handwriting recognition
LSTM [2]	Natural language text compression, handwriting recognition, gesture recognition, speech recognition, image captioning
CNN	image recognition, video analysis, natural language processing, speaker recognition

TABLE 1.2 – Comparaison entre les architectures de l'apprentissage profond

Conclusion

Dans ce chapitre, nous avons cité les différents architectures qu'offre l'apprentissage profond et les différents méthodes que nous pouvons utiliser pour extraire les caractéristiques de la voix. Maintenant, nous avons une idée générale qui va nous aider à mieux faire le choix.

Chapitre 2

Choix de la solution

Introduction

Dans ce chapitre nous allons mentionner le choix de l'architecture du deep learning que nous allons adopter dans la réalisation de notre application de reconnaissance vocale et nous allons ainsi dévoiler le choix de la technique adéquate pour l'extraction des caractéristiques des voix.

2.1 Extraxtion des caractéristiques avec MFCC

La première étape de tout système de reconnaissance vocale consiste à extraire des caractéristiques, c'est-à-dire identifier les composants du signal audio qui permettent de distinguer une voix d'une autre et d'éliminer toutes les autres informations véhiculant comme le bruit de fond, l'émotion, etc.

Nous allons utiliser les Mel-Frequency Cepstrum Coefficients MFCC [3] qui ont été élaborés dans les années 80 par Davis et Mermelstein et qui restent aujourd'hui encore assez solides pour être couramment utilisés. Dans cette partie nous allons justifier notre choix pour MFCC en abordant ses principaux aspects.

Supposons que notre signal est de longueur 3.5s comme le montre la figure ci-dessous :

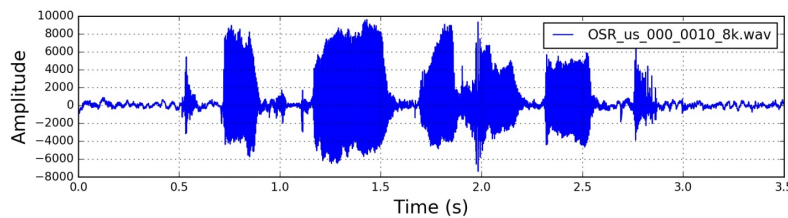


FIGURE 2.1 – Signal d'entrée

Pour le calcul des coefficients MFCC il suffit d'appliquer cette recette :

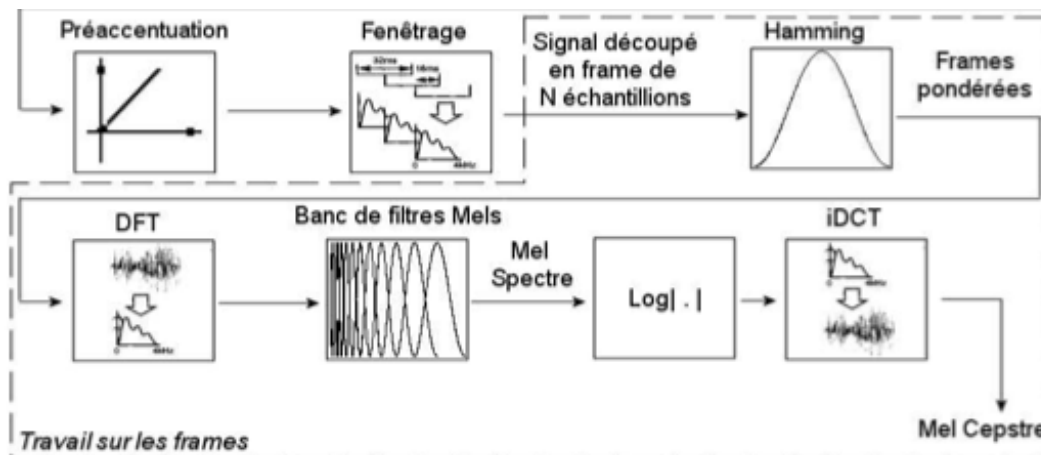


FIGURE 2.2 – Etapes d'extraction des caractéristiques

2.1.1 Préaccentuation

Le rôle de la préaccentuation est d'augmenter les hautes fréquences. Et ce pour se caler sur notre perception des aiguës. Il s'agit de faire ressortir les hautes fréquences avec un filtre passe-haut de la forme $H(z) = 1 - 0.9 Z.e(-1)$.

Il serait dommage de négliger ces zones qui contiennent beaucoup d'information (c'est à cet endroit que l'on retrouve l'énergie des fricatives). Après la préaccentuation le signal en entrée devient :

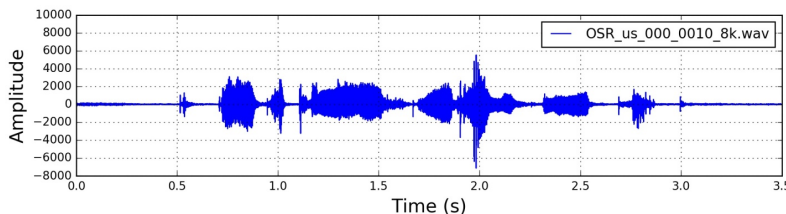


FIGURE 2.3 – Signal après la préaccentuation

2.1.2 Fenêtrage

Après la préaccentuation, nous devons diviser le signal en petits morceaux de 20 à 30 ms toutes les 10-15 ms c'est ce qu'on appelle « overlap ». La raison derrière cette étape est parce que nos lèvres, notre langue et notre glotte, n'arrêtent pas de bouger quand on parle. Hors, il est plus simple de s'appuyer sur des informations qui ne dépendent pas du temps. Et donc on estime qu'entre 20 et 30 ms notre bouche bouge peu et donc que le signal de parole produit sur cette durée possède des propriétés stochastiques variant peu. On dit alors que le signal est stationnaire. Le décalage de 10 à 15 ms quant à lui c'est simplement pour balayer finement le signal.

2.1.3 Pondération par une fenêtre de Hamming

Après avoir découpé le signal en des morceaux, un début et une fin de signal abruptes ont été introduits. Or une variation rapide dans le temps implique l'apparition de hautes fréquences parasites. Voilà pourquoi, pour éviter les effets de bord, on adoucit le début et la fin en multipliant chaque morceau. Une fenêtre Hamming a la forme suivante :

$$w[n] = 0.54 - 0.46 \cos \frac{2 \times \pi \times n}{N-1}$$

N est la longueur de la fenêtre. Où $0 \leq n \leq N - 1$

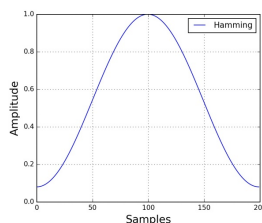


FIGURE 2.4 – Fenêtre de Hamming

2.1.4 Estimation du spectre

C'est le Passage du domaine temporel vers le domaine spectral ! Nous pouvons maintenant faire une FFT à N points sur chaque trame pour calculer le spectre de fréquence, qui est aussi appelé Transformée de Fourier à Court Terme (STFT), où N est typiquement 256 ou 512, $NFFT = 512$; puis calculer le spectre de puissance (périodogramme) en utilisant l'équation suivante :

$$P = \frac{|FFT(x_i)|^2}{N}$$

où, x_i est la i ème trame du signal x

2.1.5 Bancs du filtre Mel

Encore une fois nous devons s'approcher au maximum au fonctionnement de l'oreille. Nous n'entendons pas de la même façon les fréquences contiguës selon qu'elles soient graves ou aiguës.. Nous n'avons plus qu'à faire la même chose avec notre spectre.

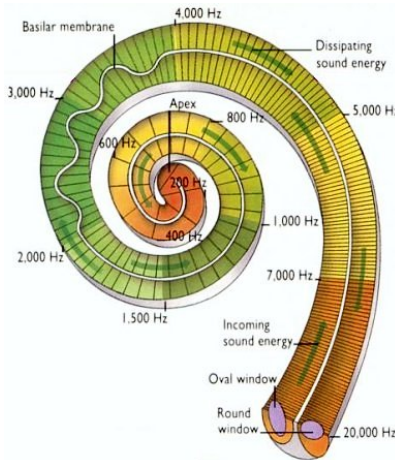


FIGURE 2.5 – Les fréquences dans l'oreille humaine

Pour cela nous allons appliquer des filtres triangulaires, typiquement 40 filtres, sur une échelle Mel au spectre de puissance pour extraire les bandes de fréquence. L'échelle Mel vise à imiter la perception non linéaire du son par l'oreille humaine, en étant plus discriminante à des fréquences plus basses et moins discriminantes à des fréquences plus élevées. Nous pouvons convertir entre Hertz (f) et Mel (m) en utilisant les équations suivantes :

$$m = 2595 \log_{10}\left(1 + \frac{f}{700}\right)$$
$$f = 700 \times 10^{\frac{m}{2595}} - 1$$

Chaque filtre du filter bank est triangulaire avec une réponse de 1 à la fréquence centrale et décroît linéairement vers 0 jusqu'à atteindre les fréquences centrales des deux filtres adjacents où la réponse est 0, comme montré sur cette figure :

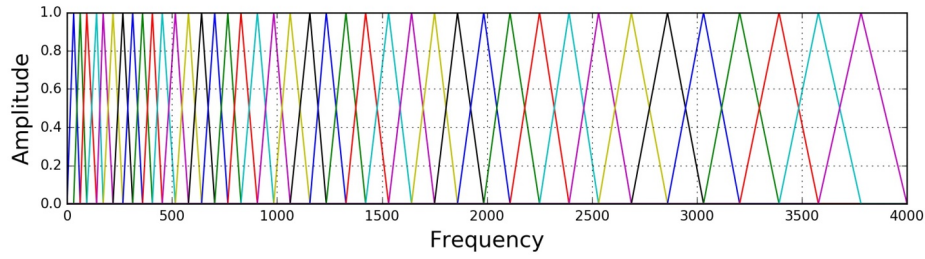


FIGURE 2.6 – Allure du filtre bank dans l'échelle mel

En calculant la somme des énergies contenues dans chaque filtre nous obtenons autant de valeurs représentatives que de filtres.

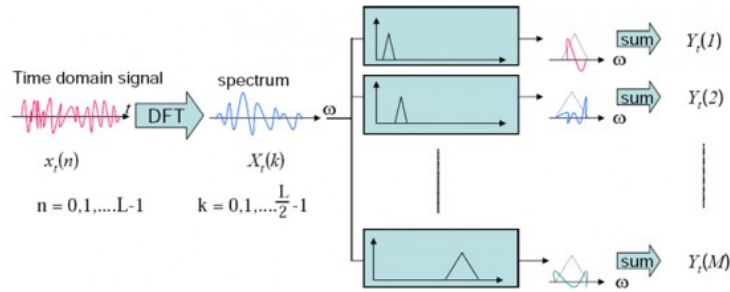


FIGURE 2.7 – Application du filtre bank

Après avoir appliqué le filtre bank au spectre de puissance (périodogramme) du signal, on obtient le spectrogramme suivant ce qu'on appelle mel-spectrogram :

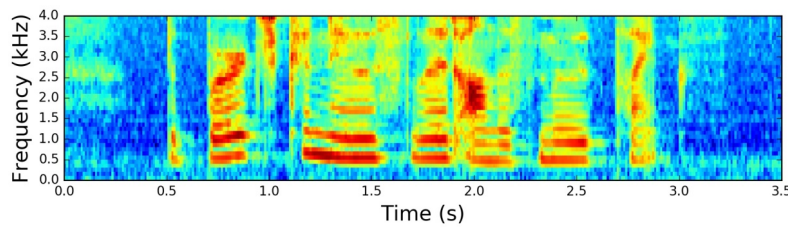


FIGURE 2.8 – Mel-spectrogram du signal

2.1.6 Calcul du logarithme de chaque énergie

Ceci est également motivé par l'ouïe humaine : nous ne percevons pas l'intensité sonore au travers d'une échelle linéaire. En gros, pour percevoir un son deux fois plus fort, il faut multiplier par huit sa puissance d'émission.

2.1.7 Calcul de la transformée en cosinus discrète inverse

Il s'avère que les coefficients du filtre bank calculés à l'étape précédente sont fortement corrélés, ce qui pourrait être une problématique dans certains algorithmes d'apprentissage. Par conséquent, nous pouvons appliquer la transformée en cosinus discrète (DCT) pour décorrélérer les coefficients du filtre bank et obtenir une représentation compressée des bancs de filtres.

En général on ne garde que les 13 premiers coefficients. Le premier étant proportionnel au log de l'énergie moyenne il est tout simplement remplacé par l'énergie de la trame. Question de représentativité.

Le résultat de cette étape est donné par cette figure :

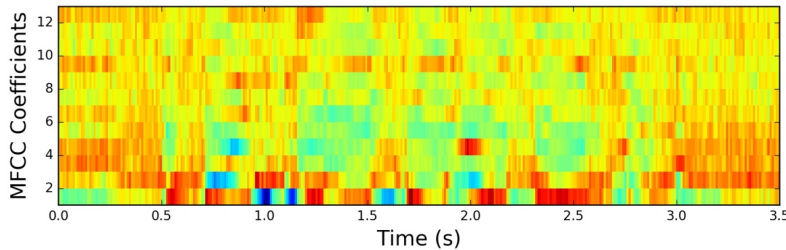


FIGURE 2.9 – MFCCs

2.1.8 Normalisation moyenne

Comme mentionné précédemment, pour équilibrer le spectre et améliorer le rapport signal / bruit (SNR), nous pouvons simplement soustraire la moyenne de chaque coefficient de toutes les trames. Nous appliquons cette normalisation moyenne (mean normalization) à la sortie du filtre bank et celle des coefficients MFCC :

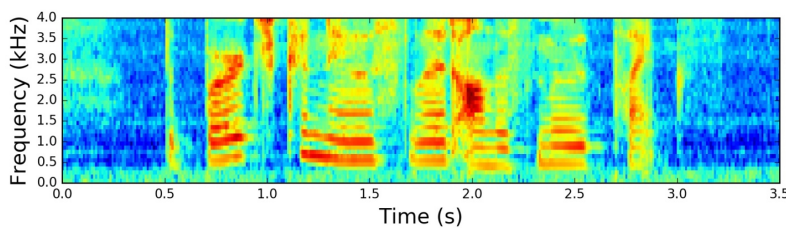


FIGURE 2.10 – Filtre bank normalisé

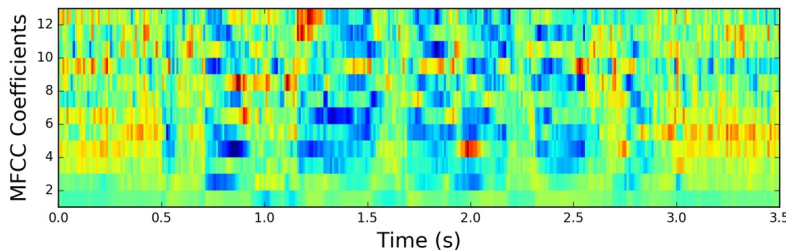


FIGURE 2.11 – MFCCs normalisés

2.1.9 Filtre bank VS MFCC

À ce stade, les étapes pour calculer les filtre banks et les MFCCs ont été discutées en termes de motivations et de mises en œuvre. Il est intéressant de noter que toutes les étapes nécessaires pour calculer les filtres banks étaient inspirées par la nature du signal vocal et la perception humaine de tels signaux. Au contraire, les étapes supplémentaires nécessaires pour calculer les MFCC ont été motivées par la limitation de certains algorithmes d'apprentissage. La transformée en cosinus discrète (DCT) était nécessaire pour décorrélérer les coefficients des filtre banks, un processus également appelé blanchiment « whitening ».

2.2 L'architecture CNN

Il est bien connu que les réseaux de neurones convolutifs (CNN ou ConvNets) ont été la source de nombreuses recherches dans le domaine de l'apprentissage profond ces dernières années, mais ils sont plutôt peu intuitifs à raisonner pour la plupart des gens. Dans cette partie nous allons décomposer les parties d'un ConvNet et voir à quoi ressemble une image après chaque étape. [9]

Un CNN est un réseau de neurones qui contient généralement plusieurs types de couches, dont une couche de convolution ainsi que des couches de regroupement (pooling) et d'activation.

2.2.1 Couche de convolution

Pour comprendre qu'est-ce qu'un CNN, nous devons comprendre comment fonctionnent les convolutions. Pour cela, nous imaginons que nous avons une image représentée comme une matrice de valeurs 5x5, et nous prenons une matrice 3x3 et nous faisons glisser cette fenêtre 3x3 autour de l'image. A chaque position des visites 3x3, nous multiplions les valeurs de notre fenêtre 3x3 par les valeurs dans l'image qui sont actuellement couvertes par la fenêtre. Il en résulte un nombre unique que représente toutes les valeurs dans cette fenêtre de l'image. Voici l'exemple ci-dessous pour la clarté :

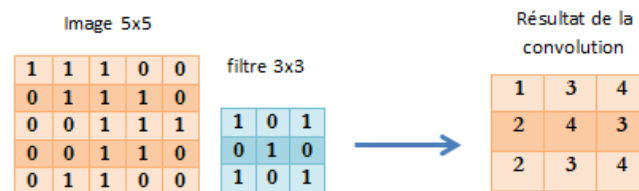


FIGURE 2.12 – Exemple de l'application d'une couche de convolution

Nous remarquons alors que chaque élément de la matrice de convolution résultante correspond à une section de l'image. La "fenêtre" qui se déplace sur l'image s'appelle un noyau (kernel). Les noyaux sont typiquement carrés et 3x3 est une taille de noyau assez commune pour les images de petite taille. La distance à laquelle la fenêtre se déplace à chaque fois s'appelle la foulée (stride). De plus, les images sont parfois complétées par des zéros (padding), autour du périmètre lors des convolutions ce qui atténue la valeur des convolutions autour des bords de l'image.

Notez que vous pouvez avoir différentes foulées (strides) horizontalement et verticalement. Vous pouvez utiliser les équations suivantes pour calculer la taille exacte de la sortie de la convolution pour :

- une entrée de taille (largeur = W , hauteur = H)
- un filtre de taille (width = F_w , hauteur = F_h)
- un padding P
- une foulée (stride) (S_w et S_h)

$$largeur = \frac{W - F_w + 2P}{S_w} + 1$$

$$hauteur = \frac{H - F_h + 2P}{S_h} + 1$$

Le but d'une couche de convolution est le filtrage. Lorsque nous nous déplaçons sur une image, nous vérifions les motifs dans cette section de l'image. Cela fonctionne à cause des filtres, des piles de poids représentés comme un vecteur, qui sont multipliés par les valeurs produites par la convolution. Quand on forme une image, ces poids changent, et quand il est temps d'évaluer une image, ces poids retournent des valeurs élevées si elle pense voir un motif qu'elle a déjà vu.

Pour les couches de convolutions, il faut bien noter que :

- La sortie de la convolution est plus petite (en largeur et en hauteur) que l'image originale
- Une fonction linéaire est appliquée entre le noyau (kernel) et la fenêtre d'image qui est sous le noyau
- Les poids dans les filtres sont appris en voyant beaucoup d'images

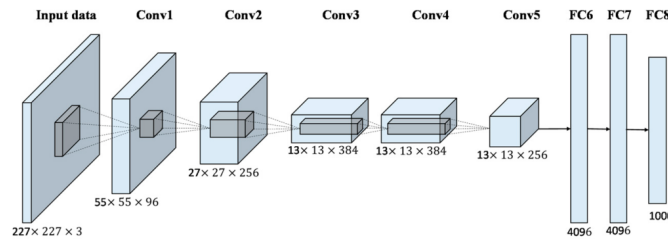


FIGURE 2.13 – Architecture CNN avec des convolutions

2.2.2 Couche de regroupement "Pooling layer"

La couche de regroupement ou encore le pooling layer est similaire au fonctionnement des couches de convolution, où nous prenons un noyau et le déplaçons sur l'image, la seule différence est la fonction qui est appliquée au noyau et encore la fenêtre d'image n'est pas linéaire. La max pooling et average pooling sont les fonctions les plus courantes. Max pooling prend la plus grande valeur de la fenêtre de l'image actuellement couverte par le noyau, tandis que l'average pooling prend la moyenne de toutes les valeurs dans la fenêtre.

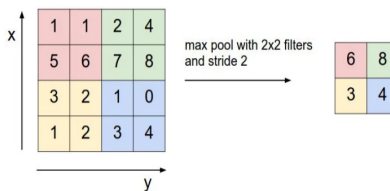


FIGURE 2.14 – Exemple de l'application d'un max pooling

2.2.3 Couche d'activation

Les couches d'activation fonctionnent exactement comme dans d'autres réseaux de neurones, une valeur est transmise à travers une fonction qui écrase la valeur dans une plage. Voici un tableau qui montre différentes fonctions d'activation les plus utilisés dans les réseaux de neurones.








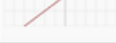

Name	Plot	Equation	Derivative
Identity		$f(x) = x$	$f'(x) = 1$
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}$
Logistic (a.k.a Soft step)		$f(x) = \frac{1}{1 + e^{-x}}$	$f'(x) = f(x)(1 - f(x))$
Tanh		$f(x) = \tanh(x) = \frac{2}{1 + e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$
ArcTan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2 + 1}$
Rectified Linear Unit (ReLU)		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Parameteric Rectified Linear Unit (PReLU) [2]		$f(x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Exponential Linear Unit (ELU) [3]		$f(x) = \begin{cases} \alpha(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} f(x) + \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
SoftPlus		$f(x) = \log_e(1 + e^x)$	$f'(x) = \frac{1}{1 + e^{-x}}$

FIGURE 2.15 – Liste de quelques fonctions d'activation

La fonction d'activation la plus utilisée dans les réseaux CNN est la relu (unité linéaire rectifiée). Il y en a plusieurs raisons pour lesquelles les gens préfèrent utiliser relus, mais le plus important est parce qu'ils ont vraiment un cout très réduit, en fait si le nombre est négatif : elle le remplace par zéro, sinon elle garde le nombre. Cette propriété rend plus rapide l'entraînement des réseaux.

Conclusion

Dans ce chapitre, nous avons détaillé l'architecture CNN que nous allons adopter dans notre travail et nous avons bien expliqué les étapes faites sur un enregistrement audio pour le transformer en un spectrogramme et le considérer comme l'entrée du modèle CNN.

Chapitre 3

Simulation Et Réalisation

Introduction

Dans ce chapitre nous allons faire une étude comparative pour étudier l'influence de quelques paramètres sur la performance de notre modèle ce qui va nous servir pour mieux choisir l'architecture finale.

3.1 Base de données

Nous avons réalisé une base de données composée de trois différents locuteurs. Chaque personne possède 200 enregistrements sous l'extension « .wav » avec une durée de 5 à 8 secondes pour chaque audio.

De plus, pour assurer l'efficacité de notre application, nous avons ajouter du bruit à 50 enregistrements de chaque locuteur.

Les enregistrements de chaque locuteurs sont regroupés dans un dossier avec un nom de celui du locuteur ce qui va être considéré comme un label dans notre traitement plus tard.

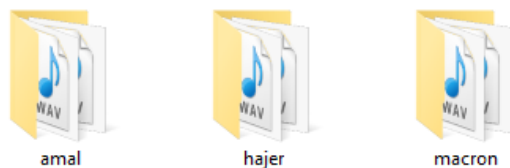


FIGURE 3.1 – L'organisation de la base de données

3.2 Paramétrage

Comme nous avons vu la description générale de l'architecture du réseau de neurones convolutif dans le chapitre précédent, nous avons choisi, à ce niveau, notre propre architecture que nous l'avons trouvée convenable avec notre dataset.

3.2.1 Les coefficients MFCCs

Tout d'abord, nous avons spécifié la taille de la matrice en entrée qui sera soit la matrice associée aux Mel-frequency cepstral coefficients ou à celle du mel spectrogramme. Nous avons réglé les paramètres cités dans la partie MFCC du deuxième chapitre comme suit :

- Taille de chaque segment : 40ms
- chevauchement : 50
- nombre de filter bank : 40
- durée de chaque enregistrement : 5s

La forme de la matrice en entrée est donc : $(249, 40, 1)$.

3.2.2 La fonction de perte "Loss Function"

La sortie du modèle est comparée, par suite, à la sortie réelle du système. À ce niveau, l'erreur est calculée selon une fonction appelée « loss-function ». Dans notre cas, nous avons utilisé la

fonction « Categorical-crossentropy » car elle est considérée la meilleure pour les problèmes de classification, sa formule est la suivant :

$$H(p, q) = - \sum_x (p(x) \times \log q(x))$$

3.2.3 L'optimiseur "Optimizer"

Les paramètres internes du modèle (les poids et les biais) jouent un rôle très important dans la formation efficace du modèle et la production des résultats précis. C'est pourquoi nous utilisons diverses stratégies et algorithmes d'optimisation pour mettre à jour et calculer les valeurs appropriées et optimales de ces paramètres. Dans notre cas on a utilisé l'algorithme « Adadelta ». [10]

Adadelta est une extension d'AdaGrad qui tend à supprimer le problème de décroissance du taux d'apprentissage. Au lieu d'accumuler tous les gradients au carré précédents, Adadelta limite la fenêtre des gradients passés accumulés à une taille fixe w .

Autrement dit, Au lieu de stocker de manière inefficace les gradients carrés précédents, la somme des gradients est récursivement définie comme une moyenne décroissante de tous les gradients carrés passés. La moyenne courante $E[g](t)$ à l'instant t dépend alors seulement de la moyenne précédente et du gradient courant.

Donc les paramètres sont mis à jour selon la formule suivant :

$$\begin{aligned} E[g^2](t) &= \gamma \times E[g^2](t-1) + (1-\gamma) \times g^2(t) \\ \Delta\theta(t) &= -\eta \times g(t, i) \quad [1] \\ \theta(t+1) &= \theta(t) + \Delta\theta(t) \end{aligned}$$

3.3 Etude comparative

3.3.1 Variation des couches de l'architecture

Nous avons commencé par entraîner le modèle avec les coefficients des mel-spectrogrammes associée aux fichiers audio.

Nous avons essayé quatre architectures en modifiant à chaque fois l'organisation, le type et le nombre des couches qui constituent le modèle et en générant les courbes qui correspondent au taux de précision (accuracy) de la validation en fonction du nombre des époques (epochs).

3.3.1.1 Architecture 1

Dans la première architecture nous avons mis en place la succession suivante des layers :

$Input(40, 249, 1) \Rightarrow Conv(32, 2 \times 2) \Rightarrow Conv(68, 2 \times 2) \Rightarrow Conv(120, 2 \times 2) \Rightarrow maxPooling(2 \times 2) \Rightarrow Conv(80, 2 \times 2) \Rightarrow maxPooling(2 \times 2) \Rightarrow Dropout(0.25) \Rightarrow Flatten \Rightarrow Dense(128) \Rightarrow Dropout(0.25) \Rightarrow Dense(64) \Rightarrow Dense(softmax) \Rightarrow Output$

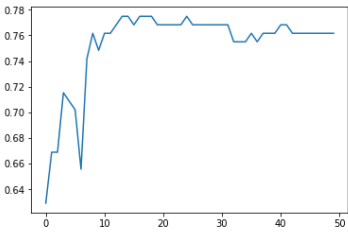
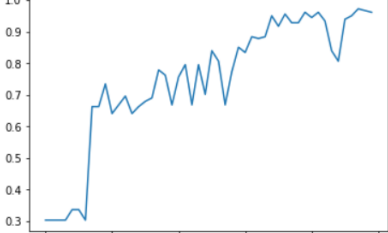
Entrée	Mel-Spectrogram	MFCC
Train acc	1.0	0.9832935560859188
Test acc	0.761589403974	0.9613259668508287
Courbe		

TABLE 3.1 – Architecture 1 : Comparaison mel-spectrogram VS MFCC

3.3.1.2 Architecture 2

Pour la deuxième architecture, nous avons varié le nombre des couches pour avoir comme résultat :

$Input(40, 249, 1) \Rightarrow Conv(32, 2 \times 2) \Rightarrow Conv(80, 2 \times 2) \Rightarrow Conv(120, 2 \times 2) \Rightarrow maxPooling(2 \times 2) \Rightarrow Conv(80, 2 \times 2) \Rightarrow Conv(120, 2 \times 2) \Rightarrow maxPooling(2, 2) \Rightarrow Conv(64, 2 \times 2) \Rightarrow maxPooling(2 \times 2) \Rightarrow Flatten \Rightarrow Dropout(0.25) \Rightarrow Dense(128) \Rightarrow Dropout(0.25) \Rightarrow Dense(64) \Rightarrow Dense(softmax) \Rightarrow Output$

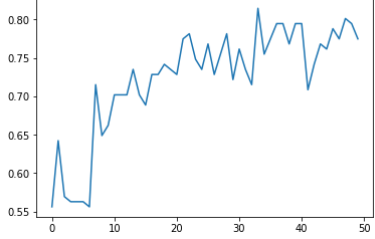
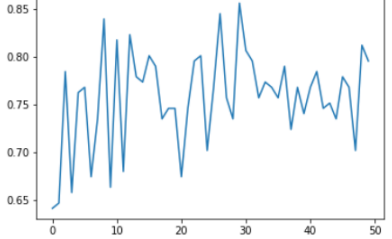
Entrée	Mel-Spectrogram	MFCC
Train acc	0.957020057307	0.7541766109785203
Test acc	0.774834437086	0.7955801104972375
Courbe		

TABLE 3.2 – Architecture 2 : Comparaison mel-spectrogram VS MFCC

3.3.1.3 Architecture 3

De même nous cherchons toujours à avoir la meilleure architecture pour notre problème et nous avons essayé l'architecture suivante :

$Input(40, 249, 1) \Rightarrow Conv(32, 2 \times 2) \Rightarrow Conv(48, 2 \times 2) \Rightarrow Conv(120, 2 \times 2) \Rightarrow maxPooling(2 \times 2)$
 $\Rightarrow Conv(48, 2 \times 2) \Rightarrow Conv(80, 2 \times 2) \Rightarrow Conv(120, 2 \times 2) \Rightarrow maxPooling(2 \times 2) \Rightarrow Conv(64, 2 \times 2)$
 $\Rightarrow Conv(80, 2 \times 2) \Rightarrow maxPooling(2 \times 2) \Rightarrow Dropout(0.25) \Rightarrow Flatten \Rightarrow Dense(128, sigmoid)$
 $\Rightarrow Dropout(0.25) \Rightarrow Dense(64, sigmoid) \Rightarrow Dropout(0.25) \Rightarrow Dense(48, sigmoid) \Rightarrow Dense(softmax) \Rightarrow Output$

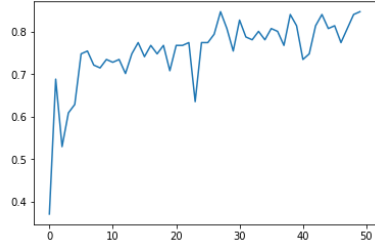
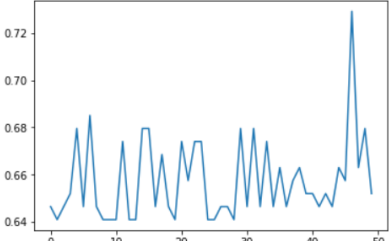
Entrée	Mel-Spectrogram	MFCC
Train acc	0.899713467049	0.6442516109945285
Test acc	0.847682119205	0.63042514287945285
Courbe		

TABLE 3.3 – Architecture 3 : Comparaison mel-spectrogram VS MFCC

3.3.1.4 Architecture 4

De même nous cherchons toujours à avoir la meilleure architecture pour notre problème et nous avons essayé l'architecture suivante :

$Input(40, 249, 1) \Rightarrow Conv(32, 2 \times 2) \Rightarrow Conv(68, 2 \times 2) \Rightarrow maxPooling(2 \times 2) \Rightarrow Conv(42, 2 \times 2) \Rightarrow Conv(80, 2 \times 2) \Rightarrow Dropout(0.25) \Rightarrow maxPooling(2 \times 2) \Rightarrow Conv(80, 2 \times 2) \Rightarrow Conv(120, 2 \times 2) \Rightarrow maxPooling(2 \times 2) \Rightarrow Flatten \Rightarrow Dense(128, sigmoid) \Rightarrow Dense(64, sigmoid) \Rightarrow Dropout(0.25) \Rightarrow Dense(48, sigmoid) \Rightarrow Dense(softmax) \Rightarrow Output$

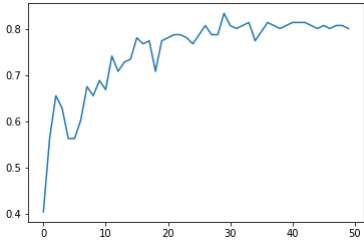
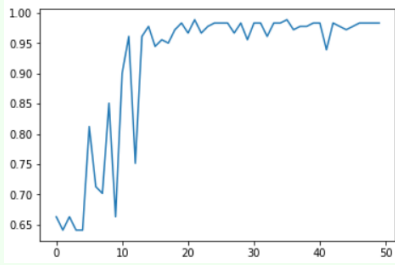
Entrée	Mel-Spectrogram	MFCC
Train acc	0.899713467049	1.0
Test acc	0.801324503311	0.9834254143646409
Courbe		

TABLE 3.4 – Architecture 4 : Comparaison mel-spectrogram VS MFCC

À ce niveau, nous avons décidé de choisir la 4ème architecture puisqu'elle donne un taux de précision qui est égale à 1 pour l'entraînement et en même temps elle offre un taux de précision très important pour le test qui est égale à 0.9834254143646409.

3.3.2 Variation des paramètres

L'étape suivante après le choix de l'architecture sera la modification de quelques paramètres qui influent sur la performance du modèle à savoir : type d'entrée au CNN, epochs, batch size, learning rate, taille du filtre, fonction de loss, valeur du dropout.

3.3.2.1 Variation du nombre des époques

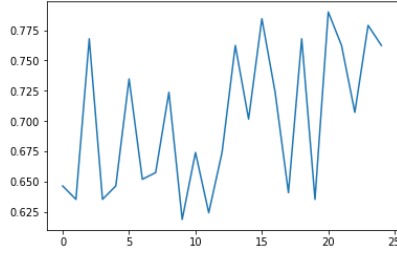
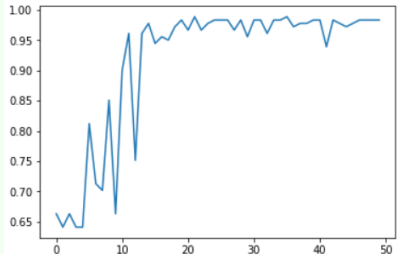
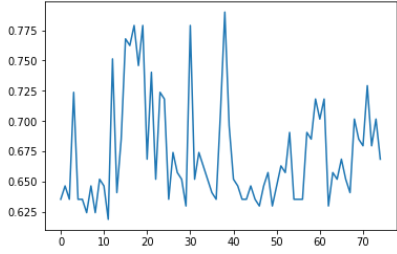
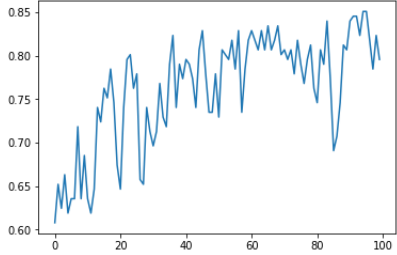
Epochs	Test acc	Train acc	Courbe
25	0.797136038	0.7624309392	
50	1.0	0.9834254143	
75	0.6968973747	0.6685082872	
100	0.887828162	0.7955801104	

TABLE 3.5 – Tableau comparatif : variation du nombre des époques

D'après ce test, nous remarquons que le taux de précision a commencé à croître jusqu'à la valeur 0.9834254143 du test de celui de 50 époques, puis il décroît avec les valeurs 75 et 100. Nous pouvons parler à ce stade des phénomènes connus en anglais "underfitting" et "overfitting".

la sous-alimentation (underfitting) fait référence à un modèle qui ne peut ni modéliser les données d'apprentissage ni généraliser de nouvelles données. Il n'est pas un modèle approprié car il aura de mauvaises performances sur les données d'entraînement.

Le sur-apprentissage (overfitting) fait référence à un modèle qui modélise trop bien les données d'entraînement. Il se produit lorsqu'un modèle apprend les détails et le bruit dans les données d'ap-

prentissage dans la mesure où il a un impact négatif sur la performance du modèle sur les nouvelles données. Cela signifie que le bruit ou les fluctuations aléatoires dans les données d'apprentissage sont captés et apprises en tant que concepts par le modèle.

Nous allons alors travailler avec un nombre d'époque qui donne le meilleur résultat qui est 50.

3.3.2.2 Variation du batch size

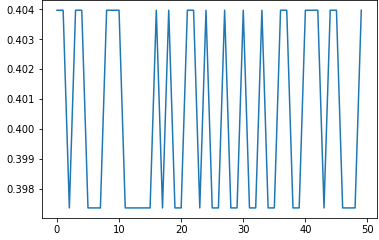
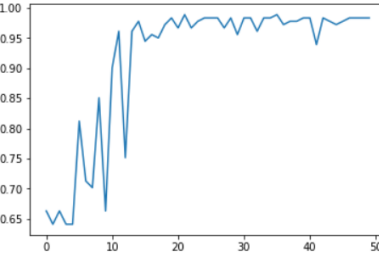
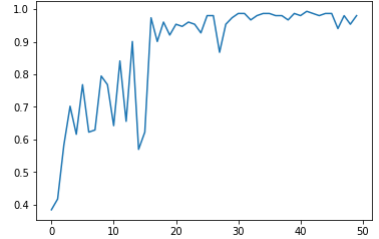
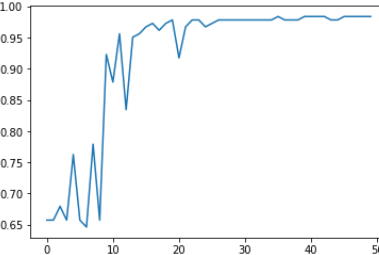
Taille du lot	Test acc	Train acc	Courbe
5	0.3982802292	0.4039735099	
10	1.0	0.9834254143	
20	0.991404011	0.980132450	
30	1.0	0.9834254143	

TABLE 3.6 – Tableau comparatif : variation du batch size

Les résultats donnés par la taille du lot 10 et 30 sont identiques sauf que la courbe n'est pas la même. Nous remarquons qu'elle est plus stable pour une taille du lot égale à 30 alors nous allons choisir cette valeur.

3.3.2.3 Variation du taux d'apprentissage "learning rate"

Ce paramètre indique à l'optimiseur jusqu'où déplacer les poids dans la direction du gradient pour un mini-lot.

Si le taux d'apprentissage est faible, la formation est plus fiable, mais l'optimisation prendra beaucoup de temps car les étapes vers le minimum de la fonction de perte sont minimes.

Si le taux d'apprentissage est élevé, la formation peut ne pas converger ou même diverger. Les changements de poids peuvent être si importants que l'optimiseur dépasse le minimum et aggrave la perte.

Nous allons effectuer quelques tests sur ce paramètre.

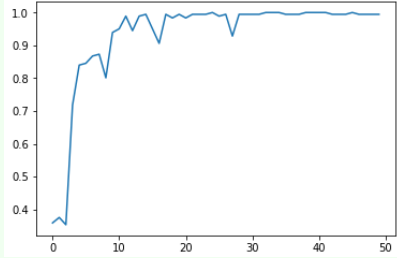
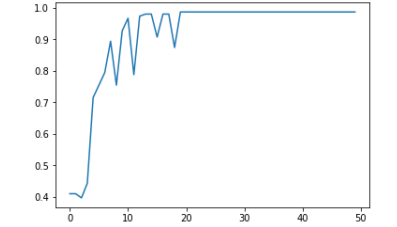
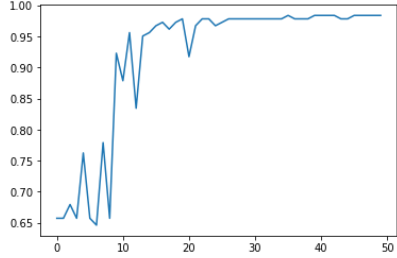
taux d'apprentissage	Test acc	Train acc	Courbe
0.1	1.0	0.9944751381	
0.5	1.0	0.986754966	
1	1.0	0.9867549668	

TABLE 3.7 – Tableau comparatif : variation du taux d'apprentissage

C'est génial! Nous avons arrivé à avoir un taux d'apprentissage très important donné par le taux d'apprentissage "0.1". Alors cette valeur sera prise dans le futur modèle final.

3.3.2.4 Variation de la taille du filtre

Dans cette section nous allons varier la taille du filtre de convolution pour déterminer son influence sur notre modèle. Il existe une infinité de combinaison pour les filtres. Nous avons choisi d'appliquer un filtre 2x2 , 4x4 et 8x8. Les résultats sont montrés dans le tableau ci-dessous :

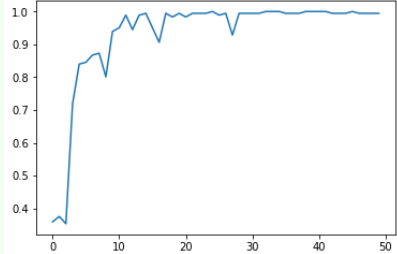
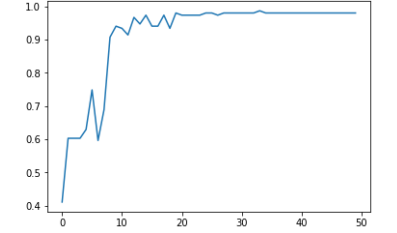
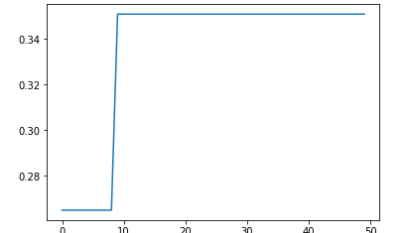
Taille du filtre	Test acc	Train acc	Courbe
2x2	1.0	0.9944751381	
4x4	1.0	0.9801324503	
8x8	0.4212034383	0.3509933774	

TABLE 3.8 – Tableau comparatif : variation de la taille du filtre

Nous remarquons que l'augmentation de la taille du filtre diminue la performance de l'application, ce qui est dû à une perte de données.
A ce niveau, nous allons choisir un filtre de taille 2x2.

3.3.2.5 Variation de la fonction de perte

La fonction de perte (loss function) est utilisée pour mesurer l'incohérence entre la valeur de prédiction et la valeur réelle. C'est une valeur non négative, où la robustesse du modèle augmente avec sa diminution.

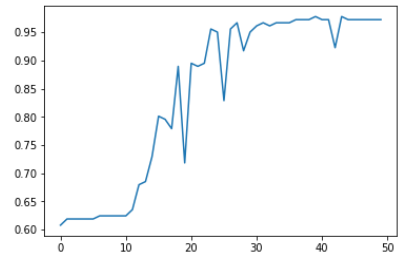
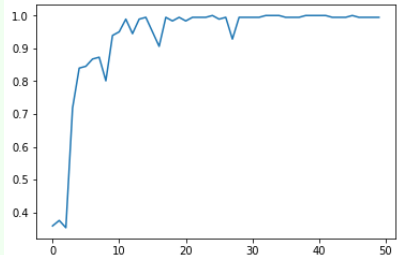
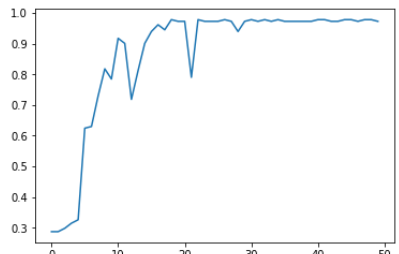
Fonction de loss	Test acc	Train acc	Courbe
mean squared error	0.9976133651	0.9723756906	
categorical crossentropy	1.0	0.9944751381	
poisson	1.0	0.9723756906	

TABLE 3.9 – Tableau comparatif : variation de la fonction de loss

Ces résultats affirment notre choix de la fonction de loss dès le début qui est 'categorical crossentropy'.

3.3.2.6 Variation de la valeur du Dropout

Le Dropout est une technique où les neurones sélectionnés au hasard sont ignorés pendant l'entraînement. Ils sont "dropped-out" au hasard. Cela signifie que leur contribution à l'activation des neurones est retirée temporairement et que les mises à jour de poids ne sont pas appliquées sur eux.

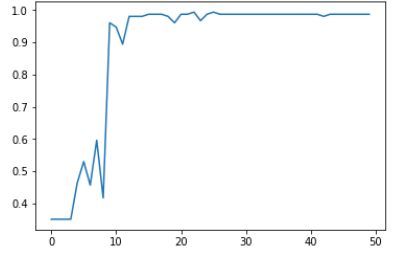
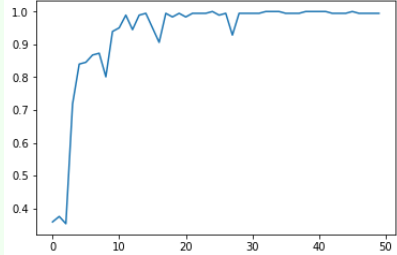
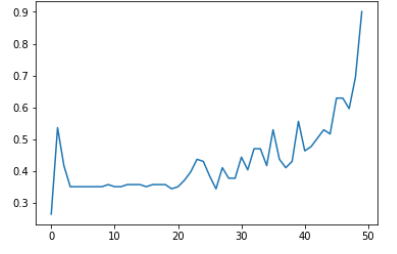
Valeur du dropout	Test acc	Train acc	Courbe
0	1.0	0.986754966	
0.25	1.0	0.9944751381	
0.8	0.948424068	0.900662251	

TABLE 3.10 – Tableau comparatif : variation de la valeur du Dropout

Nous allons finalement choisir la valeur 0.25 pour le Dropout puisqu'il nous donne la meilleur performance de notre modèle.

3.3.3 Architecture finale

Après la modification de certains paramètres dans notre modèle, nous pouvons finalement choisir l'architecture finale qui sera l'architecture 4 avec les configurations suivantes :

- Entrée : vecteur MFCC
- Nombre d'époques (epochs) : 50
- Taille du lot (batch size) : 30
- taux d'apprentissage (learning rate) : 0.1
- taille du filtre : 2x2
- fonction de loss : categorical crossentropy
- valeur du dropout : 0.25

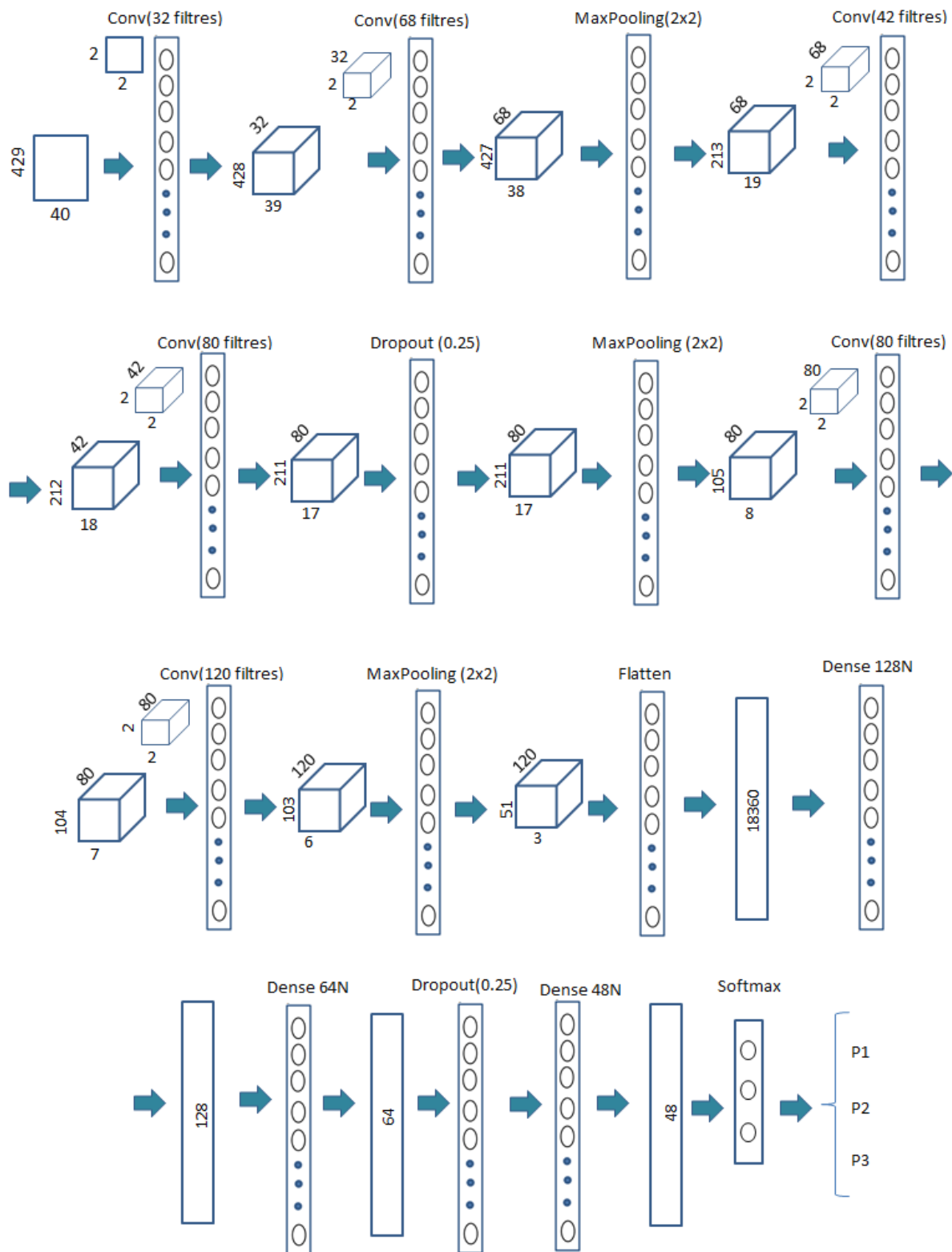


FIGURE 3.2 – Architecture finale

3.4 Interface graphique

Notre travail n'est pas encore fini, nous allons maintenant réaliser une interface graphique pour faciliter l'utilisation de notre application.

Cette interface présente quatre fonctions à savoir :

- Importer un fichier audio pour prédire le locuteur
- Afficher le spectrogramme MFCC du fichier audio importé
- Afficher le résultat de la prédiction
- Afficher les probabilités de la prédiction

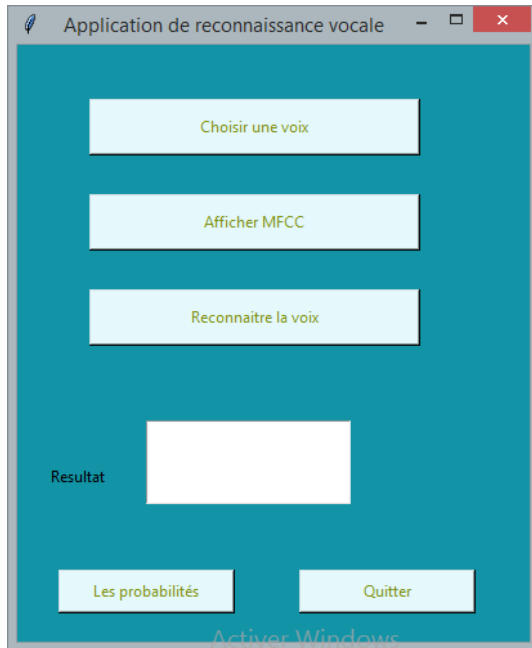


FIGURE 3.3 – Interface graphique

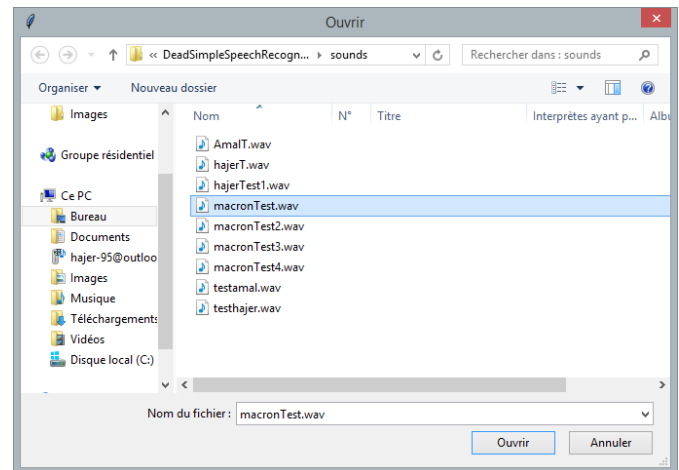


FIGURE 3.4 – Importer un fichier audio

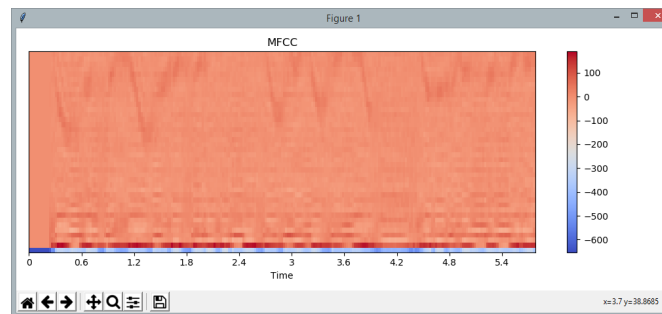


FIGURE 3.5 – Afficher le spectrogramme MFCC du fichier audio importé

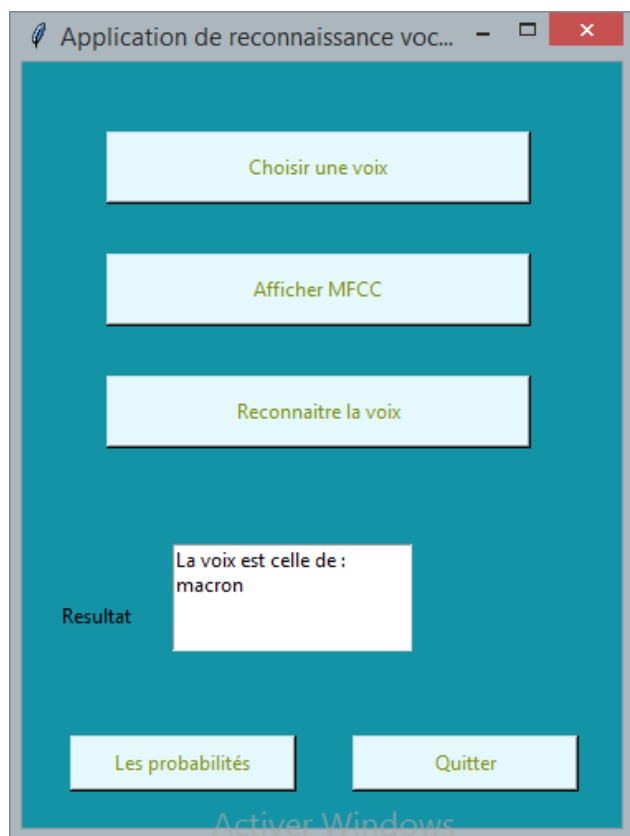


FIGURE 3.6 – Afficher le résultat de la prédiction

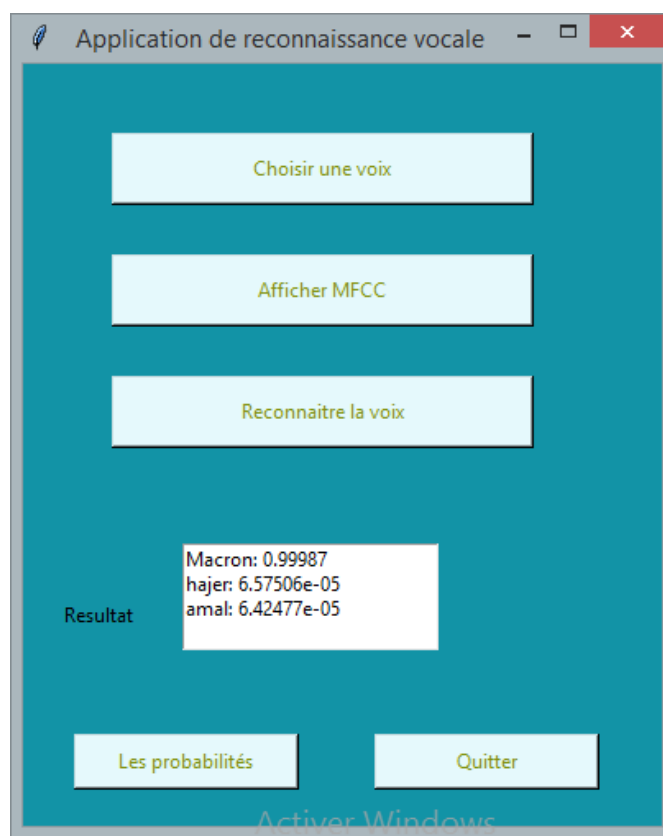


FIGURE 3.7 – Afficher les probabilités de la prédiction

Conclusion

Dans la partie pratique, nous avons arrivé à trouver une architecture que nous considérons la meilleure pour notre application de reconnaissance vocale grâce à l'étude comparative faite sur la modification des paramètres de notre modèle.

Conclusion générale

En domaine d'apprentissage profond, plusieurs applications peuvent être réalisées d'une manière très simple et avoir des résultats satisfaisants, mais il faut savoir que la reconnaissance de la voix du locuteur est un problème difficile. Il faut surmonter des défis presque illimités : des micros de mauvaise qualité, du bruit de fond, de l'écho, des variations d'accentuation, et ainsi de suite. Toutes ces questions doivent être présentées dans les données d'entraînement pour s'assurer que le réseau de neurones puisse les traiter.

L'un des plus grands défis dans le domaine de l'apprentissage profond pour les applications de reconnaissance vocale est le manque de données open source. La plupart des données vocales sont soit exclusives, difficiles d'accès, insuffisamment étiquetées... Dans de nombreux documents de recherche, les données insuffisantes ont été citées comme argument pour ne pas poursuivre des modèles et des applications plus complexes. Cet inconvénient présente un vrai challenge pour exceller dans ce domaine et apporter de nouveaux domaines d'application basés sur la reconnaissance vocale.

En outre, l'apprentissage profond est une technique très exigeante en ressources. Il nécessite des GPU plus puissants, des unités de traitement graphique de hautes performances, de grandes quantités de stockage pour former les modèles, etc. En outre, cette technique nécessite plus de temps pour s'entraîner par rapport à l'apprentissage automatique traditionnel ML.

Malgré tous ses défis, l'apprentissage profond découvre de nouvelles méthodes améliorées d'analyses de Big Data non structurées pour ceux qui ont l'intention de l'utiliser. C'est un domaine qui peut doit être présent dans tous les entreprises pour innover. En effet, les entreprises peuvent tirer des avantages significatifs de l'utilisation de l'apprentissage profond dans leurs tâches de traitement de données. Cependant, la question n'est pas de savoir si cette technique est utile, mais plutôt comment les entreprises peuvent la mettre en œuvre dans leurs projets pour améliorer la façon dont elles traitent les données.

À l'avenir, nous prévoyons d'améliorer notre application en formant une base de données beaucoup plus large en terme du nombre et durée d'enregistrement. de plus, nous allons diversifier la manière de prendre les enregistrements (à partir d'une vidéo, un microphone, un portable....) et nous allons, ainsi, varier les types de bruit présent dans chaque enregistrement pour assurer l'efficacité de l'application.

En outre, nous prévoyons intégrer cette application dans de nombreux domaines, pour pouvoir non seulement reconnaître la personne qui parle à partir de sa voix, mais aussi reconnaître son état psychique(heureux, peur, malheureux ..) et son état de santé (si cette personne souffre d'une maladie au niveau de son appareil vocal ou pas).

Bibliographie

- [1] M. Claude BARRAS. *Thèse*. URL : https://perso.limsi.fr/barras/habilitation_plus_annexes.pdf.
- [2] M. Tim JONES. *Article*. URL : <https://www.ibm.com/developerworks/library/cc-machine-learning-deep-learning-architectures/index.html>.
- [3] James LYONS. *Article*. URL : <http://www.practicalcryptography.com/miscellaneous/machine-learning/guide-mel-frequency-cepstral-coefficients-mfccs/#computing-the-mel-filterbank>.
- [4] Mme Nesma Settouti MME SOUMIA BENIKHLEF M.Bendimerad El Batoul. *HAL Archives*. URL : <https://hal.archives-ouvertes.fr/hal-00846805/document>.
- [5] OVERBLOG. *Blog*. URL : http://outilsrecherche.over-blog.com/pages/Notes_131_Lappareil_Phonatoire_Humain-3083095.html.
- [6] Ahmad SALMAN. *Thèse*. URL : https://www.research.manchester.ac.uk/portal/files/54522602/FULL_TEXT.PDF.
- [7] M. SPHOS. *Article*. URL : <https://news.sophos.com/en-us/2017/09/21/man-vs-machine-comparing-artificial-and-biological-neural-networks/>.
- [8] Stanford UNIVERSITY. *Cours*. URL : <http://cs231n.github.io/neural-networks-1/http://www.blog-formation-entreprise.fr/portfolio/on-the-job-training-o-j-t/>.
- [9] Stanford UNIVERSITY. *Cours*. URL : <http://cs231n.github.io/convolutional-networks/>.
- [10] M. Anish Singh WALIA. *Blog*. URL : <https://towardsdatascience.com/types-of-optimization-algorithms-used-in-neural-networks-and-ways-to-optimize-gradient-95ae5d39529f>.