Faculté de Génie – Branche III

Spécialité : Génie électrique et électronique

Option : Télécommunications

MINI PROJET DU SEMESTRE 8

Préparé par :

Batoul ALKAMMOUNI(5451)

Rim SAYEGH (5499)

Gender Recognition using Voice

Présenté au : Dr.Mohamed Awdi

Tables des matières

Introduction générale

L'intelligence artificielle peut se définir comme « l'ensemble de théories et de techniques mises en œuvre en vue de réaliser des machines capables de simuler l'intelligence », selon le Larousse. Soit des ordinateurs ou des programmes capables de performances habituellement associées à l'intelligence humaine, et amplifiées par la technologie : Capacité de raisonner , Capacité de traiter de grandes quantités de données, Faculté de discerner des patterns et des modèles indétectables par un humain ,Aptitude à comprendre et analyser ces modèles ,Capacités à interagir avec l'homme ,Faculté d'apprendre progressivement ,Et d'améliorer continuellement ses performances. « L'intelligence artificielle » couvre donc un vaste sujet, en perpétuel mutation. Et aux progrès fulgurants depuis 1950 , année fondatrice de l'IA . L'intelligence artificielle a franchi une étape décisive, parvenant à identifier les mots dans une conversation orale aussi bien qu'un être humain, ouvrant de nouvelles perspectives pour la reconnaissance vocale et la traduction automatique dans la vie courante. Puis, nouvelle prouesse : l'IA dépasse les humains lors de différents d'exercices de lecture et de compréhension, dans le célèbre test de lecture de l'université de Stanford. Cela permettra à l'intelligence artificielle, demain, d'interagir encore plus facilement avec les humains, pour leur apporter de l'information de manière plus naturelle.

L'intelligence artificielle a un impact considérable sur de nombreux domaines comme la reconnaissance du genre humain à l'aide de la voix.

Notre objectif est de reconnaitre le genre humain à partir de leur voix , ce qui permet de rendre la communication avec la machine plus intelligent.

Ce rapport est divisé en deux chapitres. Dans le premier chapitre , nous abordons les réseaux de neurones profonds que nous avons utilisés dans notre travail. Dans le deuxième chapitre, tout d'abord un système de reconnaissance de genre des voix est proposé. Ensuite nous montrons les résultats obtenus, les comparons et les analysons avec une analyse statistique. Enfin, la conclusion résume tout le travail et donne un bref aperçu de ce qui a été accompli.

Chapitre I:

Les réseaux de neurones

1.1-Introduction

Au cours des dernières années , les techniques d'apprentissage automatique ,en particulier lorsqu'elles ont appliquées aux réseaux de neurones, ont joues un rôle très important dans la conception des systèmes de reconnaissance de forme. Le réseau de neurone de convolution (CNN :convolution neural Network) permet aux machines de prendre des décisions précises sans l'aide des humains.

Nous allons introduites dans ce chapitre les réseaux de neurones artificiels, et nous allons décrire son architectures et son modèle mathématique.

1.2-Perceptron

Dans le domaine des technologies de l'information , un réseau de neurone est un système logiciel et matériel qui imite le fonctionnement des neurones biologiques. Un neurone formel est initialement introduit par McCulloch. Le perceptron(neurone formel) est concu pour illustrer certaines des propriétés fondamentales des systèmes intelligents en général. L'analogie entre le perceptron et les systèmes biologiques doit être évidente pour le lecteur. Ce dernier possède plusieurs entrées et un seul sortie qui correspondent respectivement aux dendrites et au cône d'émergence du neurone biologique (point de départ de l'axone). Les figures 1 et 2 représentent respectivement le neurone biologique et neurone artificiel en montrant l'analogie entre les deux structures.

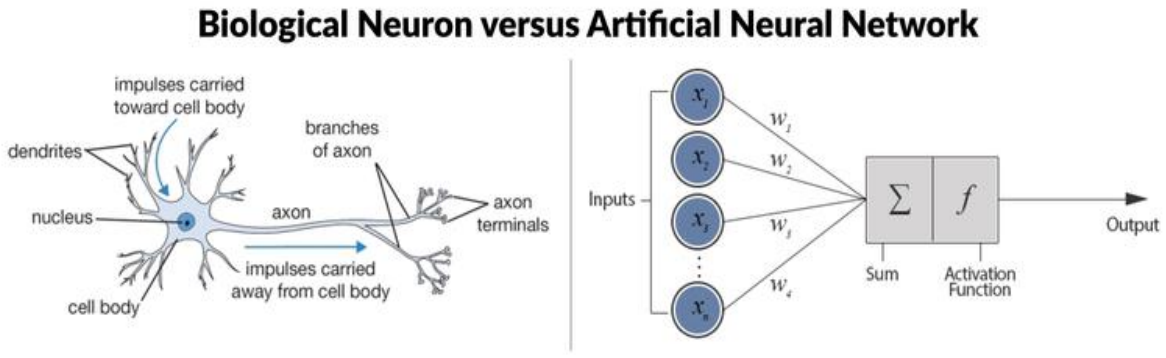


Figure1 : Structure d'un neurone biologique

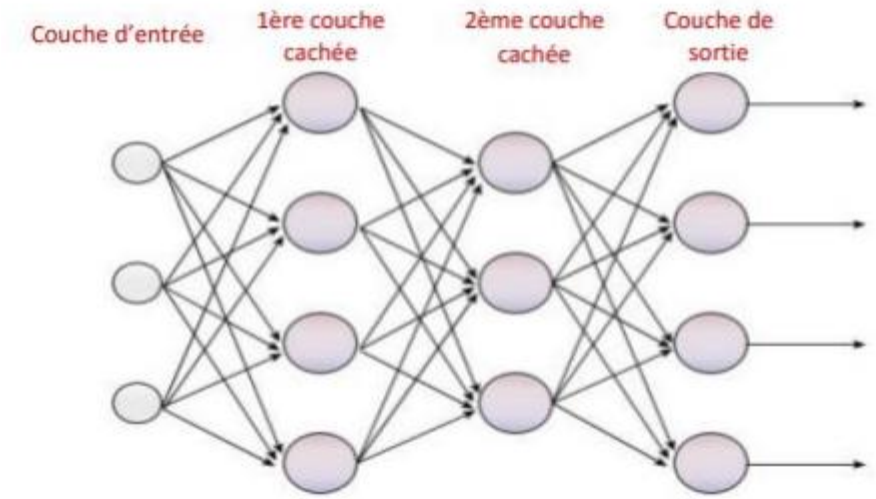


Figure 2 :L'architecture d'un perceptron multicouche à deux couches cachées.

La sortie du neurone formel est une fonction simple de la somme des signaux d'entrée x1,x2,…,xn , pondérée par des poids w1,w2,…,wn .

$y = \varphi$ ($\sum wi\ n\ i$=1 $.xi$ + b)

Avec φ est la fonction d'activation du neurone , elle peut être linéaire ou non linéaire selon le domaine d'application.

Cette fonction ,c'est ce que l'on appelle la fonction d'activation ,elle est la pour detérminer si la valeur doit passer ou non au prochaine neurones. C'est donc une analogie à ce que l'on trouve dans notre cerveau, en effet tous les neurones ne s'active pas , il y a certains qui s'activent et d'autres qui sont en pause. Pour être complet il faut savoir que le neurone a lui aussi la possibilité d'ajouter à l'entrée de la fonction d'activation ce que l'on appelle un biais , , c'est un petit valeur qu'il est aussi multiplié par la poids, cela permet aux neurones d'avoir de l'influence sur l'activation.

Linéairement séparable signifie qu'il existe un hyperplan (qui divise un espace d'entrée en deux demi-espaces) de sorte que tous les points de la pemière classe sont dans un demi-espace et ceux de la deuxième classe sont dans l'autre demi-espace.

En deux dimensions, cela signifie qu'il y a une ligne qui sépare les points d'une classe des points de l'autre classe. Et en plusieurs dimensions, cela signifie qu'il existe un hyperplan qui sépare les points d'une classe des points de l'autre classe.

Le principe de la linéairement séparable est applique sur l'opérateur <OU> et<ET> ou le perceptron a prouvé son efficacité en séparenant en 2 classes , mais il a échoué le problème n'est pas linéairement séparable(exemple : operateur <XOR>) alors on a besoin de plusieurs perceptrons interconnectes.

1.3. Perceptron multicouche

Premièrement ,la perceptron a un couche peut résoudre beaucoup des problèmes de classifications , mais elle souffre parfois de mal généralisation , ou il donne des bonnes résultats avec les exemples de l'appprentissage et donne des mals résultats avec les exemples des tests . Le perceptron multicouche (MLP), est introduite pour résoudre ces problèmes .

Le perceptron multicouche est un réseau oriente de neurones artificiels organise en couches et ou l'information voyage dans un seul sens, de la couche d'entrée vers la couche de sortie. La figure 1 donne l'exemple d'un réseau contenant une couche d'entée, deux couches cachées et une couche de sortie. La couche d'entrée représente toujours une couche virtuelle est associée aux entrées du système. Elle ne contient aucun neurone. Les couches suivantes sont des couches de neurones. Dans l'exemple illustre, il y a 3 entrées, 4 neurones sur la première couche cachée, trois neurones sur la deuxième et quatre neurones sur la couche de sortie. Les sorties des neurones de la dernière couche correspondent toujours aux sorties du système. Dans le cas genéral, un perceptron multicouche peut posséder un nombre de couches quel- conque et un nombre de neurones (ou d'entrées) par couche est également quelconque. Les neurones sont reliés entre eux par des connexions pondérées. Ce sont les poids de ces connexions qui gouvernent le fonctionnement du réseau et "programment" une application de l'espace des entrées vers l'espace des sorties ´ a l'aide d'une transformation non linéaire. La création d'un perceptron multicouche pour résoudre un problème donne passe donc par l'inférence de la meilleure application possible telle que définie par un ensemble de données d'apprentissage constituées de paires de vecteurs d'entrées et de sorties désirées. Cette inférence peut se faire, entre autre, par l'algorithme dit de rétropropagation.
Maintenant , on va determiner les poids des connesxions pondérées de manière aléatoire , donc forcement la réponse a très peu de chance d'être juste, donc il faut trouver les paramétres d'un tel réseau en minimisant une fonction d'erreur . Cette inférence peut se faire par l'algorithme de la retropropagation de gradient (Back Propagation).
Il faut voir le réseau à neurones artificiels comme un humain qui n'aurait aucune connaissance a priori. On veut que cette personne apprenne à reconnaître une forme, un objet, quelque chose. Pour cela on va lui montrer un maximum d'entrées représentant l'objet à reconnaître (ou autre). Idéalement ces entrées doivent toutes être différentes afin de généraliser au maximum ce que l'on souhaite reconnaître.

1.4. La couche dense de Keras

1.4.1.Qu'est-ce-qu'une couche dense dans le réseau neuronale ?

La couche dense est une couche de réseau neuronal qui est connectée profondément, ce qui signifie que chaque neurone de la couche dense reçoit l'entrée de tous les neurones de sa couche précédente. La couche dense s'avère être la couche la plus couramment utilisée dans les modèles.
En arrière-plan, la couche dense effectue une multiplication matrice-vecteur. Les valeurs utilisées dans la matrice sont en fait des paramètres qui peuvent être formés et mis à jour à l'aide de rétropropagation.
La sortie générée par la couche dense est un vecteur de dimension 'm'. Ainsi, la couche dense est essentiellement utilisée pour modifier les dimensions du vecteur. Les couches denses appliquent également des opérations telles que la rotation, la mise à l'échelle, la translation sur le vecteur.

1.4.2. Paramètres de la couche dense de Keras

Voyons les différents paramètres de la fonction de couche dense de Keras ci-dessous :

1.4.2.1.Unités

Paramètre le plus basique de tous les paramètres, il utilise un entier positif comme valeur et représente la taille en sortie de la couche.

C'est le paramètre d'unité lui-même qui joue un rôle majeur dans la taille de la matrice de poids avec le vecteur de biais**.**

1.4.2.2.Activation

Le paramètre activation est utile pour appliquer la fonction d'activation par élément dans une couche dense. Par défaut, l'activation linéaire est utilisée, mais nous pouvons modifier et passer à l'une des nombreuses options que Keras fournit pour cela.

1.4.2.3.Use_Bias

Un autre paramètre simple, **use_bias** aide à décider si nous devons inclure un vecteur de biais à des fins de calcul ou non. Par défaut, **use_bias** est défini sur true.

1.4.2.4.Initialiseurs

Comme son nom l'indique, le paramètre initializer est utilisé pour fournir une entrée sur la façon dont les valeurs de la couche seront initialisées. Dans le cas de la couche dense, la matrice de poids et le vecteur de biais doivent être initialisés.

1.4.2.5.Régularisateurs

Les régularisations contiennent trois paramètres qui effectuent la régularisation ou la pénalité sur le modèle. En règle générale, ces paramètres ne sont pas utilisés régulièrement, mais ils peuvent aider à la généralisation du modèle.

1.4.2.6.Contraintes

Ce dernier paramètre détermine les contraintes sur les valeurs que la matrice de pondération ou le vecteur de biais peut prendre.

1.4.3.Opération de couche dense de Keras

La fonction de couche dense de Keras implémente après l'opération :
output = activation(point(input, noyau) + biais)

Dans l'équation ci-dessus, **l'activation** est utilisée pour effectuer **l'activation par élément** et le **noyau** est la matrice **de poids** créée par la couche, et le **biais** est un vecteur de biais créé par la couche.
La couche dense de Keras sur la couche de sortie effectue **le produit scalaire** du **tenseur d'entrée** et de **la matrice du noyau de poids.**
Un vecteur de biais est ajouté et l'activation par élément est effectuée sur les valeurs de sortie.

1.4.4.Exemples de couche dense de Keras

Nous allons vous montrer deux exemples de couche dense Keras, le premier exemple vous montrera comment construire un réseau neuronal avec une seule couche dense et le deuxième exemple expliquera la conception de réseau neuronal ayant plusieurs couches denses.

1.4.4.1. Construction d'un réseau neuronal peu profond avec une couche dense de Keras

Voyons maintenant comment un modèle Keras avec une seule couche dense est construit. Ici, nous utilisons le modèle Keras intégré, c'est-à-dire séquentiel.
Tout d'abord, nous fournissons la couche d'entrée au modèle, puis une **couche dense** avec l'activation **ReLU** est ajoutée.
La couche en sortie contient également une couche dense, puis nous examinons la forme de la sortie de ce modèle.

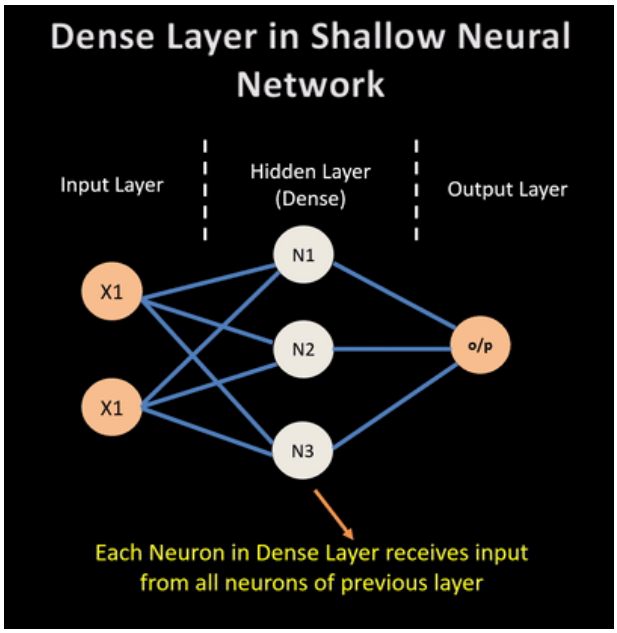


Figure 3 :Dense Layer in shallow neural network

1.4.4.2.Création d'un réseau neuronal profond avec des couches denses Keras

Dans cet exemple, nous examinons un modèle où plusieurs couches masquées sont utilisées dans des réseaux neuronaux profonds. Ici, nous utilisons la fonction d'activation ReLu dans les neurones de la couche dense cachée.
N'oubliez pas que l'on ne peut pas encore trouver les poids et le résumé du modèle, d'abord le modèle reçoit des données d'entrée, puis nous examinons les poids présents dans le modèle.
Enfin, le résumé du modèle affiche les informations sur les **couches en entrée,**la **forme des couches en sortie** et le nombre total de **paramètres.**

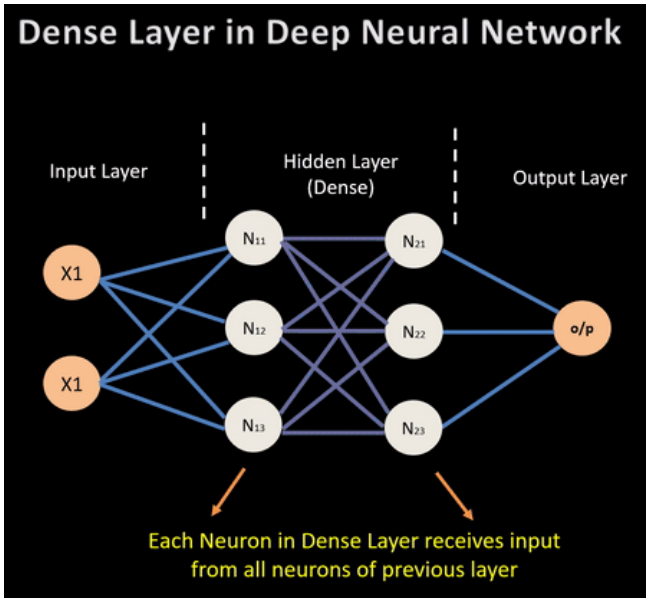


Figure 4 :Dense Layer in Deep neural network

1.5. Le spectrogramme Mel

Il est bien connu que les CNN ont des performances élevées pour la classification des images. On a pu montrer les performances des architectures CNN populaires, telles que AlexNet, VGG, Inception et ResNet, lorsqu'elles sont utilisées pour la classification audio. Leur approche consistait à décomposer la série temporelle audio avec une transformée de Fourier à court terme pour créer un spectrogramme qui a été utilisé comme entrée pour le CNN. Le problème auquel nous avons été confrontés avec beaucoup de ces modèles est qu'ils sont volumineux et que de nombreux paramètres peuvent être entraînés.

Une approche pour contourner ce problème est l'apprentissage par transfert. Cela implique d'utiliser un réseau pré-entraîné, de geler la plupart des poids de couche, et en ne recyclant que les dernières couches sur nos données d'entraînement audio. C'est l'une des approches que nous avons suivies pour ce projet.
Les forme d'onde sont d'abord traitée en rognant les sections silencieuses du clip, puis en rognant davantage ou en ajoutant des zéros pour équivaloir à une durée de 2 secondes. Ceci est nécessaire car nos modèles nécessitent une dimension d'entrée statique, et 2 secondes est la durée moyenne des données audio. Ensuite, chaque clip traité est transformé en sa représentation de spectrogramme Mel. Un spectrogramme est une représentation visuelle de la composition fréquentielle d'un signal au fil du temps. L'échelle de Mel fournit une échelle linéaire pour le système auditif humain, et est liée à Hertz par la formule suivante, où m représente Mels et F représente Hertz :

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

Le spectrogramme Mel est utilisé pour fournir à nos modèles des informations sonores similaires à ce qu'un humain percevrait. Les formes d'onde audio brutes sont passées à travers des bancs de filtres pour obtenir le spectrogramme Mel. Après ce processus, chaque échantillon a une forme de 128 x 128, indiquant 128 banques de filtres utilisées et 128 pas de temps par clip. Les figures 1 et 2 affichent respectivement un clip audio brut et la représentation de fréquence Mel correspondante.

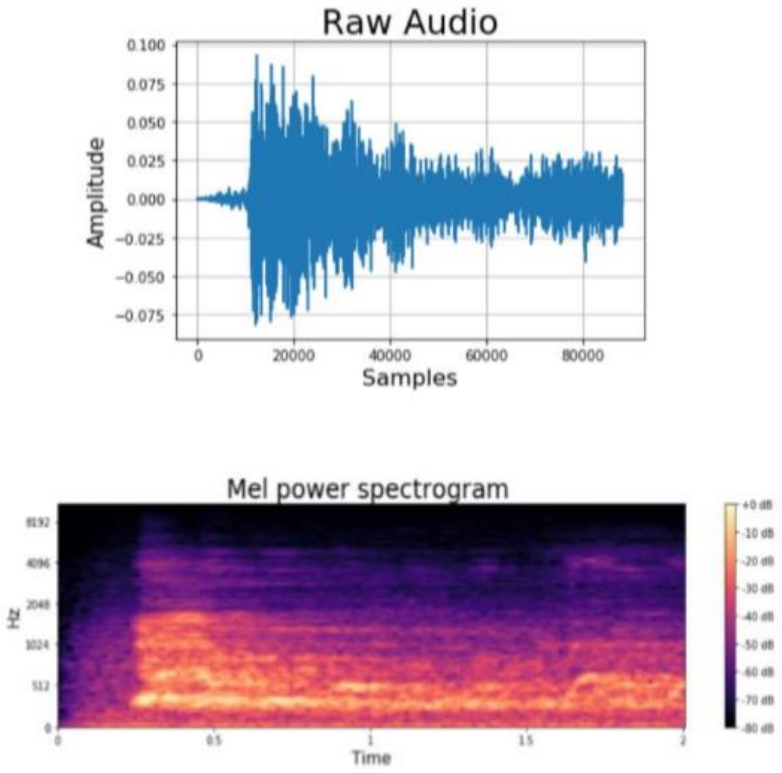


Figure 5 :Mel Power spectrogram

1.4.6. MFCC

La reconnaissance vocale est une tâche d'apprentissage supervisé. Dans le problème de reconnaissance vocale, l'entrée sera le signal audio et nous devons prédire le texte à partir du signal audio. Nous ne pouvons pas utiliser le signal audio brut en entrée de notre modèle car il y aura beaucoup de bruit dans le signal audio. Il est observé que l'extraction de caractéristiques du signal audio et son utilisation comme entrée dans le modèle de base produiront de bien meilleures performances que de considérer directement le signal audio brut comme entrée. MFCC est la technique largement utilisée pour extraire les caractéristiques du signal audio.

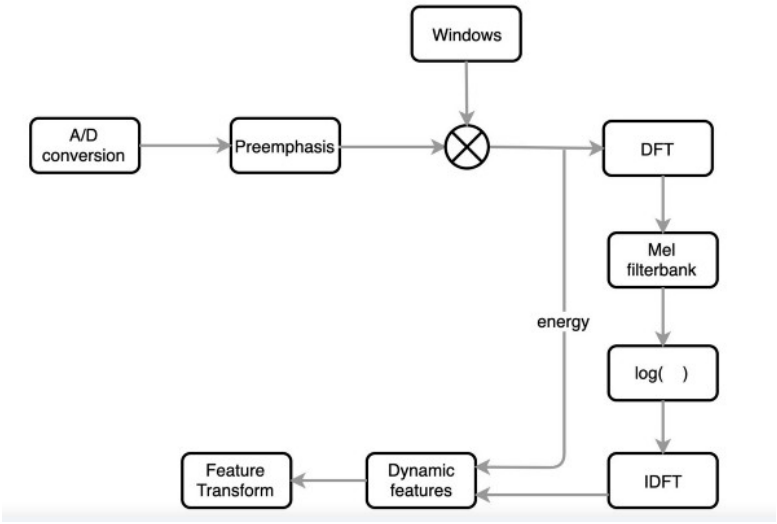
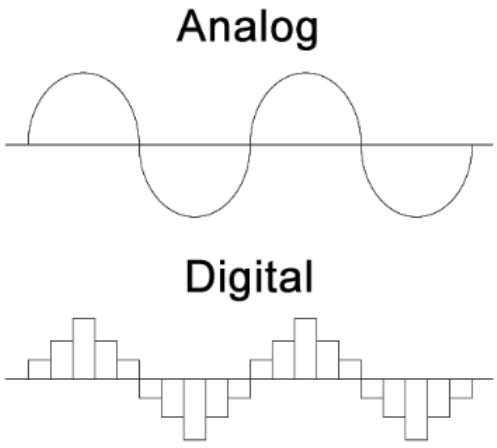


Figure 6 :MFCC diagram

Nous examinerons chaque étape par étape.

1. **Conversion A/N:** Dans cette étape, nous allons convertir notre signal audio du format analogique au format numérique avec une fréquence d'échantillonnage de 8 kHz ou 16 kHz.



2. **Préaccentuation :**

La préaccentuation augmente l'amplitude de l'énergie dans la fréquence plus élevée. Lorsque nous regardons le domaine fréquentiel du signal audio pour les segments voisés comme les voyelles, on observe que l'énergie à une fréquence plus élevée est bien inférieure à l'énergie à des fréquences plus basses. Augmenter l'énergie dans les fréquences plus élevées améliorera la précision de détection du téléphone, améliorant ainsi les performances du modèle.
La préaccentuation est effectuée par le filtre passe-haut du premier ordre comme indiqué ci-dessous. Le domaine fréquentiel du signal audio pour la voyelle « aa », avant et après la préaccentuation est indiqué ci-dessous.

3. **Fenêtrage :**

La technique MFCC vise à développer les caractéristiques du signal audio qui peuvent être utilisées pour détecter les téléphones dans la parole. Mais dans le signal audio donné, il y aura de nombreux téléphones, nous allons donc diviser le signal audio en différents segments, chaque segment ayant une largeur de 25 ms et le signal étant distant de 10 ms.En moyenne, une personne prononce trois mots par seconde avec 4 téléphones et chaque téléphone aura trois états, ce qui donne 36 états par seconde ou 28 ms par état, ce qui est proche de notre fenêtre de 25 ms.
De chaque segment, nous extrairons 39 caractéristiques. De plus, en cassant le signal, si on le coupe directement sur les bords du signal, la chute brutale d'amplitude sur les bords va produire du bruit dans le domaine des hautes fréquences. Ainsi, au lieu d'une fenêtre rectangulaire, nous utiliserons des fenêtres de Hamming/Hanning pour hacher le signal qui ne produira pas de bruit dans la région des hautes fréquences.

4. **DFT (Transformée de Fourier discrète) :**

Nous allons convertir le signal du domaine temporel au domaine fréquentiel en appliquant la transformée dft. Pour les signaux audio, l'analyse dans le domaine fréquentiel est plus facile que dans le domaine temporel.

5. **Banque de filtres Mel :**

La façon dont nos oreilles percevront le son est différente de la façon dont les machines percevront le son. Nos oreilles ont une résolution plus élevée à une fréquence plus basse qu'à une fréquence plus élevée. Donc, si nous entendons un son à 200 Hz et 300 Hz, nous pouvons le différencier facilement par rapport aux sons à 1500 Hz et 1600 Hz même si les deux avaient une différence de 100 Hz entre eux. Alors que pour la machine la résolution est la même à toutes les fréquences. Il est à noter que la modélisation de la propriété auditive humaine au stade de l'extraction des caractéristiques améliorera les performances du modèle.
Nous allons donc utiliser l'échelle mel pour mapper la fréquence réelle à la fréquence que les êtres humains percevront.

6. **Journal d'application :**

Les humains sont moins sensibles au changement d'énergie du signal audio à une énergie plus élevée par rapport à une énergie plus faible. La fonction log a également une propriété similaire, à une faible valeur d'entrée x le gradient de la fonction log sera plus élevé mais à une valeur élevée de la valeur de gradient d'entrée est moindre. Nous appliquons donc log à la sortie du filtre Mel pour imiter le système auditif humain.

7. **IDFT :**

Dans cette étape, nous effectuons la transformation inverse de la sortie de l'étape précédente. Avant de savoir pourquoi nous devons faire une transformation inverse, nous devons d'abord comprendre comment le son produit par les êtres humains.

Le son est en fait produit par la glotte qui est une valve qui contrôle le flux d'air entrant et sortant des voies respiratoires. La vibration de l'air dans la glotte produit le son. Les vibrations se produiront en harmoniques et la plus petite fréquence produite est appelée fréquence fondamentale et toutes les fréquences restantes sont des multiples de la fréquence fondamentale. Les vibrations qui sont produites seront transmises dans la cavité vocale. La cavité vocale amplifie et amortit sélectivement les fréquences en fonction de la position de la langue et des autres articulateurs. Chaque son produit aura sa position unique de la langue et des autres articulateurs.

Le modèle MFCC prend les 12 premiers coefficients du signal après avoir appliqué les opérations idft. Avec les 12 coefficients, il prendra l'énergie de l'échantillon de signal comme caractéristique.

$$Energy = \sum_{t=t_1}^{t_2} x^2[t]$$

8. **Fonctionnalités dynamiques :**

Parallèlement à ces 13 caractéristiques, la technique MFCC considérera la dérivée du premier ordre et les dérivées du second ordre des caractéristiques qui constituent 26 autres caractéristiques.

Les dérivés sont calculés en prenant la différence de ces coefficients entre les échantillons du signal audio et cela aidera à comprendre comment la transition se produit.

Ainsi, la technique MFCC globale générera 39 caractéristiques à partir de chaque échantillon de signal audio qui seront utilisées comme entrée pour le modèle de reconnaissance vocale.

Chapitre 2 :

Modèles de reconnaissance le genre humain à partir de leur voix

2.1. Libraries

- **NumPy :**

NumPy est une bibliothèque pour le langage de programmation Python, l'ajout du support pour de grands tableaux et matrices multidimensionnels, ainsi qu'une grande collection de fonctions mathématiques de haut niveau pour opérer sur ces tableaux.

NumPy cible l'implémentation de référence CPython de Python, qui est un interpréteur de bytecode non-optimisant. Les algorithmes mathématiques écrits pour cette version de Python s'exécutent souvent beaucoup plus lentement que leurs équivalents compilés. NumPy résout le problème de la lenteur en partie en fournissant des tableaux et des fonctions multidimensionnels et des opérateurs qui fonctionnent efficacement sur les tableaux ; leur utilisation nécessite de réécrire du code, principalement des boucles internes, à l'aide de NumPy.

Les liaisons Python de la bibliothèque de vision par ordinateur largement utilisée OpenCV utilisent des tableaux NumPy pour stocker et exploiter les données. Étant donné que les images avec plusieurs canaux sont simplement représentées sous forme de tableaux tridimensionnels, l'indexation, le découpage ou le masquage avec d'autres tableaux sont des moyens très efficaces d'accéder à des pixels spécifiques d'une image. Le tableau NumPy en tant que structure de données universelle dans OpenCV pour les images, les points de caractéristiques extraits, les noyaux de filtre et bien d'autres simplifie considérablement le flux de travail de programmation et le débogage.

audio2numpy charge un fichier audio et sort directement les données audio sous forme de tableau numpy et son taux d'échantillonnage. Prend en charge .wav, .aiff via la bibliothèque standard de python et .mp3 via ffmpeg.

- **TensorFlow :**

TensorFlow est une interface de programmation évolutive et multiplateforme pour la mise en œuvre et l'exécution d'algorithmes d'apprentissage automatique, y compris des wrappers pratiques pour l'apprentissage en profondeur.

Pour améliorer les performances des modèles d'apprentissage machine d'entraînement, TensorFlow permet l'exécution sur les CPU et les GPU. Cependant, ses plus grandes capacités de performances peuvent être découvertes lors de l'utilisation de GPU. TensorFlow prend officiellement en charge les GPU compatibles CUDA. La prise en charge des appareils compatibles OpenCL est encore expérimentale. Cependant, OpenCL sera probablement officiellement pris en charge dans un proche avenir.

TensorFlow est construit autour d'un graphe de calcul composé d'un ensemble de nœuds. Chaque nœud représente une opération qui peut avoir zéro ou plusieurs entrées ou sorties. Les valeurs qui traversent les bords du graphe de calcul sont appelées tenseurs. Les tenseurs peuvent être compris comme une généralisation de scalaires, vecteurs, matrices, etc. Plus concrètement, un scalaire peut être défini comme un tenseur de rang 0, un vecteur comme un tenseur de rang 1, une matrice comme un tenseur de rang 2 et des matrices empilées dans une troisième dimension comme des tenseurs de rang 3. Une fois qu'un graphe de calcul est construit, le graphe peut être lancé dans une session TensorFlow pour exécuter différents nœuds du graphe.

- **Scikit-learn :**

Scikit-learn est probablement la bibliothèque la plus utile pour l'apprentissage automatique en Python. La bibliothèque sklearn contient de nombreux outils efficaces pour l'apprentissage automatique et la modélisation statistique, notamment la classification, la régression, le clustering et la réduction de la dimensionnalité.

Veuillez noter que sklearn est utilisé pour créer des modèles d'apprentissage automatique. Il ne doit pas être utilisé pour lire les données, les manipuler et les résumer. Il existe de meilleures bibliothèques pour cela (par exemple NumPy, Pandas, etc.)

Scikit-learn est livré avec de nombreuses fonctionnalités. En voici quelques-uns pour vous aider à comprendre la propagation :

1. **Algorithmes d'apprentissage supervisé :** pensez à tout algorithme d'apprentissage automatique supervisé dont vous avez peut-être entendu parler et il y a de fortes chances qu'il fasse partie de scikit-learn. Des modèles linéaires généralisés (par exemple, régression linéaire), des machines à vecteurs de support (SVM), des arbres de décision aux méthodes bayésiennes - tous font partie de la boîte à outils scikit-learn. La propagation des algorithmes d'apprentissage automatique est l'une des principales raisons de l'utilisation élevée de scikit-learn. J'ai commencé à utiliser scikit pour résoudre des problèmes d'apprentissage supervisé et je le recommanderais également aux personnes novices en scikit/machine learning.
2. **Validation croisée :** il existe différentes méthodes pour vérifier l'exactitude des modèles supervisés sur des données invisibles à l'aide de sklearn.
3. **Algorithmes d'apprentissage non supervisé :** Encore une fois, il existe une large gamme d'algorithmes d'apprentissage automatique dans l'offre, du clustering à l'analyse factorielle, en passant par l'analyse des composants principaux, jusqu'aux réseaux de neurones non supervisés.
4. **Divers ensembles de données de jouets :** cela s'est avéré utile lors de l'apprentissage de scikit-learn. J'avais appris SAS en utilisant divers ensembles de données académiques (par exemple, ensemble de données IRIS, ensemble de données sur les prix des maisons de Boston). Les avoir à portée de main tout en apprenant une nouvelle bibliothèque a beaucoup aidé.
5. **Extraction de caractéristiques :** Scikit-learn pour extraire des caractéristiques à partir d'images et de texte (par exemple, un sac de mots).

- **Pandas :**

Pandas est principalement utilisé pour l'analyse de données. Pandas permet d'importer des données à partir de divers formats de fichiers tels que des valeurs séparées par des virgules, JSON, SQL, Microsoft Excel. Pandas permet diverses opérations de manipulation de données telles que la fusion, le remodelage, la sélection, ainsi que le nettoyage des données et les fonctionnalités de gestion des données.

- **PyAudio :**

Pour commencer à lire et à enregistrer de l'audio sous Windows, Linux et MacOS dans un environnement Python, vous devriez envisager d'utiliser la bibliothèque PyAudio. PyAudio est un ensemble de liaisons Python pour PortAudio, une bibliothèque C++ multiplateforme s'interfaçant avec les pilotes audio.

- **Librosa :**

Librosa est un package Python pour l'analyse musicale et audio. Librosa est essentiellement utilisé lorsque nous travaillons avec des données audios comme dans la génération de musique (à l'aide de LSTM), la reconnaissance vocale automatique. Il fournit les blocs de construction nécessaires pour créer les systèmes de recherche d'informations musicales.

2.2.Préparation de l'ensemble de données

Nous n'utiliserons pas de données audios brutes, car les échantillons audios peuvent être de n'importe quelle longueur et peuvent être problématiques en termes de bruit. En conséquence, nous devons effectuer une sorte d'extraction de caractéristiques avant de l'introduire dans le réseau de neurones.

L'extraction de caractéristiques est toujours la première phase de toute tâche d'analyse vocale, elle prend essentiellement un audio de n'importe quelle longueur en entrée et génère un vecteur de longueur fixe qui convient à la classification. Quelques exemples de méthodes d'extraction de caractéristiques sont le MFCC et le spectrogramme de Mel.

le « Common Voice Dataset de Mozilla » a été utilisé, c'est un corpus de données vocales lues par les utilisateurs sur le « site Common Voice » , son but est de permettre l'apprentissage et le test de la reconnaissance vocale automatique. Cependant, après avoir jeté un coup d'œil à l'ensemble de données, de nombreux échantillons sont en fait étiquetés dans la colonne genre.

Voici les étapes pour préparer l'ensemble de données pour la reconnaissance du genre :

- Tout d'abord, seulement les échantillons qui sont étiquetés dans le champ genre sont filtrés.

- Après cela, l'ensemble de données est équilibré afin que le nombre d'échantillons féminins soit égal à celui des échantillons masculins, cela aidera le réseau neuronal à ne pas sur adapter à un sexe particulier.
- Enfin, on a utilisé la technique d'extraction du spectrogramme de Mel pour obtenir un vecteur de la longueur 128 de chaque échantillon de voix.

En outre, on a inclus le script qui est chargé de préparer et prétraiter l'ensemble de données (à partir .mp3 dans les .npyfichiers).

Pour commencer, on doit installez les bibliothèques qu'on va avoir besoin. Ensuite, on doit ouvrir un nouveau bloc-notes et importez les modules nécessaires. On va parler un peu de ces modules :

Modèle séquentiel : Un Sequentialmodèle est approprié pour une pile simple de couches où chaque couche a exactement un tenseur d'entrée et un tenseur de sortie.

Modèle dense : Une couche de réseau de neurones densément connecté. Dense implémente l'opération activation (matmul(input, weight) + bias) , où le poids est une matrice de poids, le biais est un vecteur de biais et l'activation est une fonction d'activation par élément. Cette couche prend également en charge les tenseurs de poids 3D avec des matrices de polarisation 2D

LSTM : Le STM (Long Short-Term Memory network) est un type de réseau de neurones récurrents capable de mémoriser les informations passées et tout en prédisant les valeurs futures, il prend en compte ces informations passées.

Dropout : La couche Dropout définit de manière aléatoire les unités d'entrée sur 0 avec une fréquence de rate à chaque étape pendant le temps d'entraînement, ce qui permet d'éviter le surapprentissage. Les entrées non définies sur 0 sont augmentées de 1/(1 - taux) de sorte que la somme de toutes les entrées reste inchangée.

ModelCheckpoint : le rappel est utilisé en conjonction avec l'entraînement en utilisant model.fit()pour enregistrer un modèle ou des poids (dans un fichier de point de contrôle) à un certain intervalle, de sorte que le modèle ou les poids peuvent être chargés ultérieurement pour continuer l'entraînement à partir de l'état enregistré.

TensorBoard : est un outil permettant de fournir les mesures et les visualisations nécessaires au cours du flux de travail d'apprentissage automatique. Il permet de suivre les métriques d'expérience telles que la perte et la précision, de visualiser le graphique du modèle, de projeter les plongements dans un espace de dimension inférieure, et bien plus encore

Earlystopping : est une méthode qui vous permet de spécifier un grand nombre arbitraire d'époques d'entraînement et d'arrêter l'entraînement une fois que les performances du modèle ne s'améliorent plus sur un ensemble de données de validation suspendu.

Train_test_split : La procédure de fractionnement train-test est utilisée pour estimer les performances des algorithmes d'apprentissage automatique lorsqu'ils sont utilisés pour faire des prédictions sur des données non utilisées pour former le modèle.

Maintenant, pour obtenir le sexe de chaque échantillon, il existe un fichier de métadonnées CSV qui relie le chemin du fichier de chaque échantillon audio à son sexe approprié.

Une fois un grand nombre d'échantillons audio équilibrés, la fonction suivante charge tous les fichiers dans un seul tableau, nous n'avons besoin d'aucun mécanisme de génération car il s'adapte à la mémoire

La fonction load_data() est responsable de la lecture de ce fichier CSV et du chargement de tous les échantillons audios dans un seul tableau, cela a pris un certain temps la première fois quand on l'a exécuté, mais ce tableau fourni est enregistré dans un resultsdossier, ce qui a fait gagner du temps dans les autres Cours.

label2int mappe simplement chaque genre à une valeur entière, nous en avons besoin dans la fonction load_data()pour traduire les étiquettes de chaîne en étiquettes entières.

Nous devons diviser notre ensemble de données en ensembles d'entraînement, de test et de validation, la fonction split_data () le fait.

Nous utilisons train_test_split () qui est une fonction pratique de sklearn , elle mélangera notre ensemble de données et le divisera en ensembles d'entraînement et de test, puis nous l'exécutons à nouveau sur l'ensemble d'entraînement pour obtenir l'ensemble de validation.

2.3. Modèle

2.3.1. Générateur du modèle

Maintenant, ce dictionnaire de données contient tout ce que nous avons besoin pour ajuster notre modèle, construisons le modèle . pour ce tutoriel, nous allons utiliser un réseau neuronal d'avance profond avec 5 couches cachées, ce n'est pas l'architecture parfaite, mais il fait le travail jusqu'à présent.

Nous utilisons un taux d'abandon de 30% après chaque couche entièrement connectée, ce type de régularisation empêchera, espérons-le, le surajustement sur le jeu de données d'apprentissage.

Une chose importante à noter ici est que nous utilisons une seule unité de sortie (neurone) avec une fonction d'activation sigmoïde dans la couche de sortie, le modèle sortira le scalaire 1 (ou proche de celui-ci) lorsque le haut-parleur de l'audio est un mâle, et femelle quand il est plus proche de 0.

En outre, nous utilisons l'entropie croisée binaire comme fonction de perte, car il s'agit d'un cas particulier d'entropie croisée catégorique lorsque nous n'avons que 2 classes à prédire. Utilisons cette fonction pour construire notre modèle .

2.3.2. Formation du modèle

Maintenant que nous avons créé le modèle, entraînons-le à l'aide du jeu de données précédemment chargé.

Nous avons défini deux rappels qui seront exécutés après la fin de chaque époque :

- Le premier est le tensorboard, nous allons l'utiliser pour voir comment le modèle va pendant l'entraînement en termes de perte et de précision.
- Le deuxième rappel est l'arrêt anticipé,cela arrêtera l'entraînement lorsque le modèle cessera de s'améliorer, j'ai spécifié une patience de 5,ce qui signifie qu'il arrêtera l'entraînement après 5 époques de ne pas s'améliorer, la définition de restore_best_weights sur True restaurera les poids optimaux enregistrés pendant l'entraînement et l'affectera aux poids du modèle.

Enregistrons ce modèle et Voici ma sortie :

```
_________________________________________________________________
Train on 54219 samples, validate on 6025 samples
Epoch 1/100
54219/54219 [==============================] - 8s 143us/sample - loss: 0.5514 - accuracy: 0.7651 - val_loss: 0.3807 - val_accuracy: 0.850
Epoch 2/100
54219/54219 [==============================] - 5s 93us/sample - loss: 0.4159 - accuracy: 0.8326 - val_loss: 0.3464 - val_accuracy: 0.8536
Epoch 3/100
54219/54219 [==============================] - 5s 93us/sample - loss: 0.3860 - accuracy: 0.8466 - val_loss: 0.3112 - val_accuracy: 0.8744
<..SNIPPED..>
Epoch 16/100
54219/54219 [==============================] - 5s 96us/sample - loss: 0.2864 - accuracy: 0.8936 - val_loss: 0.2387 - val_accuracy: 0.9087
Epoch 17/100
54219/54219 [==============================] - 5s 95us/sample - loss: 0.2824 - accuracy: 0.8945 - val_loss: 0.2464 - val_accuracy: 0.9110
Epoch 18/100
54219/54219 [==============================] - 6s 103us/sample - loss: 0.2887 - accuracy: 0.8920 - val_loss: 0.2406 - val_accuracy: 0.907
Epoch 19/100
54219/54219 [==============================] - 5s 95us/sample - loss: 0.2822 - accuracy: 0.8939 - val_loss: 0.2435 - val_accuracy: 0.9080
Epoch 20/100
54219/54219 [==============================] - 5s 96us/sample - loss: 0.2813 - accuracy: 0.8957 - val_loss: 0.2567 - val_accuracy: 0.8993
Epoch 21/100
54219/54219 [==============================] - 5s 89us/sample - loss: 0.2759 - accuracy: 0.8962 - val_loss: 0.2442 - val_accuracy: 0.9112
```

Figure 7 :La sortie du modèle

Comme vous pouvez le voir, l'apprentissage du modèle s'est arrêté à l'époque 21et a atteint environ 0,2387 perte et près de 91% de précision de validation (c'était à l'époque 16).

2.4. Test du modèle

Étant donné que le modèle est maintenant formé et que les poids sont optimaux, testons-le à l'aide de notre jeu de tests que nous avons créé précédemment , Incroyable, nous avons atteint une précision de 91% sur des échantillons que le modèle n'avait jamais vus auparavant! C'est super!

Si vous ouvrez tensorboard (en utilisant la commande: tensorboard --logdir ="logs »),vous verrez des courbes de perte et de précision similaires à ceci:

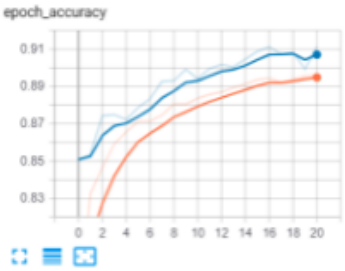


Figure 8 :La courbe de précision

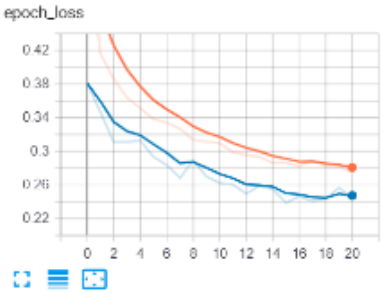


Figure 9 : la courbe de perte

La courbe bleue est l'ensemble de validation, tandis que l'orange est l'ensemble d'entraînement, vous pouvez voir que la perte diminue au fil du temps et que la précision augmente, c'est exactement ce à quoi nous nous attendions. Apres avoir tester le modèle pour des échantillons qu'il n'a jamais vue auparavant, il a atteint une précision de 91%.

2.5. Test du modèle avec votre propre voix

On a créé un script qui enregistre la voix jusqu'à ce que on arrête de parler et l'enregistre dans un fichier, puis extrait les fonctionnalités de cet audio et le transmet au modèle pour récupérer les résultats.

La fonction extract_feature () est la fonction qui est chargée de charger le fichier audio et d'en extraire les fonctionnalités.

Nous utilisons le module argparse pour analyser le chemin du fichier transmis à partir des lignes de commande, si le fichier n'est pas transmis (en utilisant --fileou en -fparamètre), le script commencera à enregistrer en utilisant votre microphone par défaut.

Nous créons ensuite le modèle et chargeons les poids optimaux que nous avons entraînés auparavant, puis nous extrayons les caractéristiques de ce fichier audio transmis (ou enregistré) et nous utilisons model.predict() pour obtenir les prédictions résultantes.

conclusion

Au cours du progrès de l'humain, les techniques d'apprentissage automatique en pris grand rôle dans l'amélioration de la performance et dans la diminution du temps et de l'effort donne au travail.

L'objectif de ce travail est la reconnaissance du genre d'une personne à partir de sa voie. La solution proposée se base sur l'utilisation des réseaux de neurones à convolution CNN. Ce choix est motivé par leur capacité à résoudre des problèmes de son très complexes grâce à l'extraction de caractéristiques sonores discriminantes de haut niveau. Cependant, ces réseaux ont besoin d'un grand nombre de données d'apprentissage annotées pour être performants. L'originalité de ce travail réside dans l'apprentissage de caractéristiques avec la technique de mfcc.

Premièrement nous avons abordé aux réseaux de neurones. On a décrit leurs architectures et leurs modèle mathématiques. Ensuite on a expliqué le spectrogramme de mel et la technique MFCC utilise pour détecter les caractéristiques du son. Apres nous avons présentés les étapes suivies en implémentant le code python. Enfin, nous avons présenté les résultats du système de reconnaissance proposé en montrant la capacité des réseaux de neurones à convolution à extraire des caractéristiques sonores et son succès dans le domaine de la classification de ces sons.

Une perspective intéressante serait d'étendre notre travail à identifier les maladies liées a l'appareil respiratoire, a l'aide de leur voie, en se basant sur les caractéristiques sonores liées a ces maladies.

Annexe

Github.com/rimsayegh/Gender-Voice-Recognition