RÉPUBLIQUE DU BÉNIN
MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR
ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITÉ D'ABOMEY-CALAVI

INSTITUT DE FORMATION ET DE RECHERCHE EN INFORMATIQUE

BP 526 Cotonou Tel : +229 21 14 19 88
http://www.ifri-uac.bj Courriel : contact@ifri.uac.bj

MÉMOIRE

pour l'obtention du

Diplôme de Master en Informatique

Option : Systèmes d'Information et Réseaux Informatiques

Présenté par :
Sero Abdel-Kader **SAKA**

Mise en place d'un cloud Rest API pour l'identification et l'authentification de locuteur par empreinte vocale

Bon pour dépôt.
Le 06/05/2024
John AOGA
97599277.

Sous la supervision :
Dr John **AOGA**

Membres du jury :

Eugène C. EZIN	Professeur titulaire	IFRI-UAC	Président
Chérif DIALLO	Professeur titulaire	IFRI-UAC	Examinateur
John AOGA	Docteur	IFRI-UAC	Rapporteur

Année Académique : **2020-2021**

Sommaire

Dédicace ii

Remerciements iii

Résumé iv

Abstract v

Table des figures vi

Sigles et abréviations viii

Glossaire x

Introduction 1

1 État de l'art de l'identification par biométrie vocale 3

2 Modélisation et développement de notre modèle d'intelligence Artificielle 22

3 Matériels et Méthodes 30

4 Résultats et Discussions 49

Conclusion 55

56

Bibliographie 56

Table des matières 58

Dédicace

A mon père,

Sero Abdel-kader SAKA

Remerciements

C’est avec un grand plaisir que je réserve ces quelques lignes en signe de gratitude et de profonde reconnaissance à tous ceux qui, de près ou de loin, ont contribué à la réalisation et l’aboutissement de ce travail.
Je tiens tout d’abord à remercier Le **Docteur John AOGA** pour son soutien, son accompagnement, et surtout pour son aide précieuse tout au long de l’élaboration de ce travail.
Notre parfaite gratitude au corps professoral de l’ **IFRI** en général et plus particulièrement au **Prof. Eugène C. EZIN, Directeur de l’Institut de Formation et de Recherche en Informatique (IFRI)**.
Je m’acquitte, enfin, volontiers d’un devoir de gratitude et de remerciements à **tous mes enseignants** pour la qualité de l’enseignement qu’ils ont bien voulu me prodiguer durant mes études afin de me fournir une formation efficiente.

Résumé

L'émergence de technologies de pointe, de plus en plus sophistiquées et accessibles à tous, a facilité la compromission des mots de passe et des systèmes d'authentification conventionnels. Face à ces nouveaux défis de sécurité, il devient impératif de repenser les méthodes d'authentification pour garantir une protection adéquate des données sensibles. Dans cette optique, notre projet vise à concevoir un système d'authentification et d'identification de locuteur (Identification de Locuteurs (ILs)) déployé dans le cloud, offrant une solution sécurisée et pratique pour l'ère des innovations numérique. Cette approche de pointe exploite la puissance des algorithmes d'apprentissage automatique pour analyser les caractéristiques vocales uniques de chaque individu, permettant une reconnaissance précise et fiable. Pour atteindre cet objectif ambitieux, nous avons d'abord développé un système basé sur des réseaux de neurones profonds pour l'authentification et l'identification des locuteurs (ILs). Ce modèle complexe a été soigneusement élaboré et entraîné à partir de vastes ensembles de données vocales afin d'assurer une précision optimale dans la reconnaissance des voix. Une fois le modèle finalisé, il a été intégré dans une architecture de type RESTful API (REST APIs), permettant ainsi son déploiement sur une infrastructure cloud. Cette interface intuitive permet une intégration facile dans diverses applications et services, rendant la technologie accessible à un large public. Grâce à cette approche innovante, notre solution offre une alternative fiable et efficace aux méthodes d'authentification traditionnelles, renforçant ainsi la sécurité des systèmes et des données dans un environnement nuxmérique en constante évolution.

Mots clés : Reconnaissance de la parole et des locuteurs, biométrie vocale, échantillon vocal, biométrie vocale, reconnaissance de la parole

Abstract

The digital landscape is constantly evolving, with sophisticated technologies becoming increasingly accessible to everyone. This accessibility, however, comes with a double-edged sword. While it empowers users, it also exposes them to new security threats. Passwords and traditional authentication systems, once considered secure, are now vulnerable to compromise. This necessitates a paradigm shift in how we approach user authentication to ensure the continued protection of sensitive data. With this in mind, our project aims to design a Speaker Recognition (Speaker Recognitions (SRs)) deployed in the cloud, offering a secure and convenient solution for the digital innovation era. This cutting-edge approach harnesses the power of machine learning algorithms to analyze the unique vocal characteristics of each individual, enabling precise and reliable recognition. To achieve this ambitious goal, we first developed a system based on deep neural networks for speaker authentication and identification (ILs). This complex model was carefully crafted and trained using extensive sets of voice data to ensure optimal accuracy in voice recognition. Once the model was finalized, it was integrated into a RESTful API (REST APIs) architecture, allowing for deployment on a cloud infrastructure. This intuitive interface enables easy integration into various applications and services, making the technology accessible to a wide audience. Through this innovative approach, our solution provides a reliable and effective alternative to traditional authentication methods, thereby enhancing the security of systems and data in an ever-evolving digital environment.

Key words: research in speech and speaker recognition, Speaker Recognition (SR), speech recognition, voice biometrics, voice sample

Table des figures

1.1 Architecture d'une IA de reconnaissance par biométrie vocale. 4
1.2 Ajout d'un individu à une IA de reconnaissance parempreinte vocale. 6
1.3 Intelligence Artificielle, Machin learning et Deep learning. 11
1.4 Machine learning, Nouveau paradigme de programmation. 12
1.5 Un Réseau de neurone pour la reconnaissance des caractères. 13
1.6 modèle de classification de caractères manuscrits. 14
1.7 Paramétrisation d'un réseau de neurones 15
1.8 Détermination de la fonction Loss 16
1.9 Le Loss Score utilisé pour optimiser le Weigth à l'entrée 17

2.1 Structure d'un RN de type TDNN 23
2.2 caractéristiques des enregistrements vocaux 24
2.3 Le dataset dansle bucket Cloud Storage 24
2.4 enregistrements vocaux d'un seul utilisateur 25
2.5 paramètres d'exécution du Colab 26
2.6 Ressource de l'environnement d'exécution Colab 27
2.7 Structure d'un RN de type TDNN 28

3.1 enregistrements vocaux d'un seul utilisateur 31
3.2 enregistrements vocaux d'un seul utilisateur 31
3.3 récupération des données 32
3.4 Fichier manifestes d'entrainement 33
3.5 Fichier des paramètres d'entrainement 34
3.6 paramètres de base d'entrainement 35
3.7 paramètres des chemins des fichiers manifestes 35
3.8 paramètres des chemins des fichiers manifestes 36
3.9 Autres paramètres 36
3.10 Paramètres d'augmentation des données avec les bruits 37
3.11 Paramètres d'augmentation des données avec les bruits 38
3.12 Code d'entrainement 39
3.13 commande de separation des données 39
3.14 Code de separation des données 40
3.15 ligne du fichier manifeste 41
3.16 Classe Brain 41
3.17 compute forward 42
3.18 augmentation des données 42
3.19 definition de la fonction de perte 43

3.20 Logs de l'entrainement variation de la function loss 44
3.21 Logs de l'entrainement, passage de l'Epoch 14 à l'Epoch 15 45
3.22 Les deux models 46
3.23 Préparation du model 47
3.24 Inférence sur le modèle pour l'authentification 47
3.25 Inférence sur le modèle pour l'identification 48

4.1 Liste des POI 50
4.2 Meilleure modèle sauvegardé sur Cloud Storage 51
4.3 récupération du meilleur modèle 51
4.4 Identification de locuteur 52
4.5 Identification de locuteur 53
4.6 Identification de locuteur 54

Sigles et abréviations

acroglo :

A la fois dans les deux *Glossaire :* acroglo

ADN :

Acide DésoxyriboNucléique 3

API :

Application Programming Interface 2

CNN :

Convolutional Neural Networks 9

CPU :

Recurrent Neural Networks 25

CSV :

Comma-Separated Values 32

DNN :

Deep Neural Networks 17

GPU :

Graphics Processing Unit 25

IA :

Intelligence Artificielle 2

IL :

Identification de Locuteur iv, v

IoT :

Internet of Things 3

JSON :

JavaScript Object Notation 32

MFA :

Multi-Factor Authentication 49

PIN :

Personal Identification Number, en français : numéro d'identification personnel 2

RNN :

Recurrent Neural Networks 9

SR :

Speaker Recognition v

TDNN :

Time-Delay Neural Network 26

Glossaire

acroglo :

A la fois dans les deux. viii

REST API :

REST (Representational State Transfer) est un style d'architecture logicielle qui définit un ensemble de contraintes à utiliser pour créer des services web. Une API RESTful (ou REST API) est une interface de programmation d'application qui respecte ces contraintes et permet la communication entre des systèmes distribués. iv, v, 2

Introduction Générale

1. Problématique

L'avènement de technologies avancées et de plus en plus accessibles a rendu la compromission des mots de passe et des systèmes d'authentification conventionnels plus aisée. Cette évolution soulève un défi majeur en matière de sécurité des données et nécessite une révision des méthodes d'authentification pour garantir une protection adéquate.

La complexité des mots de passe et leur longueur peuvent rendre leur utilisation fastidieuse, tandis que les codes PIN, bien que plus mémorables, sont souvent plus faciles à deviner. Ainsi, la question de l'authentification par mot de passe ou code PIN devient un délicat compromis entre sécurité et facilité d'utilisation, exigeant une approche prudente qui tienne compte des risques potentiels et des besoins des utilisateurs.

D'autre part, la biométrie vocale émerge comme une alternative prometteuse, applicable dans divers domaines tels que la sécurité, l'accès en ligne, la surveillance, et la gestion des appels téléphoniques. Toutefois, cette technologie soulève des préoccupations substantielles liées à la vie privée, à la fiabilité des systèmes, et à l'éthique. Les données vocales, une fois collectées, peuvent être exploitées de manière malveillante si elles tombent entre de mauvaises mains, suscitant des inquiétudes quant à la protection de la vie privée et des libertés individuelles.

2. Contexte et justification

Face à la fraude documentaire et au vol d'identité, aux menaces du terrorisme ou de la cybercriminalité, et face à l'évolution logique des réglementations internationales, de nouvelles solutions technologiques sont progressivement mises en œuvre. Parmi ces technologies, la biométrie s'est rapidement distinguée comme la plus pertinente pour identifier et authentifier les personnes de manière fiable et rapide, en fonction de caractéristiques biologiques uniques. Aujourd'hui, de nombreuses applications font appel à cette technologie. Ce qui était autrefois réservé à des applications sensibles telles que la sécurisation de sites militaires est devenue une application grand public en développement rapide.

Un simple geste de la main, une pression du doigt sur un capteur, une expression prononcée de la voix, ou un regard d'une seconde vers une caméra suffisent pour prouver son identité.

L'authentification biométrique vocale facilite la vie des consommateurs et citoyens qui sont de plus en plus mobiles et connectés en leur apportant une alternative simple aux mots de passe et aux codes Personal Identification Number, en français : numéro d'identification personnels (PINs) pour s'authentifier. Ainsi nous souhaiterons mettre en place un Cloud Application Programming Interfaces (APIs) pour l'authentification et l'identification des individus par biométrie vocale.

3. Objectifs

L'objectif global de notre recherche est de concevoir un système Cloud basé sur l'intelligence artificielle, accessible via une REST API. Cette interface intuitive permet une intégration facile dans diverses applications et services, rendant la technologie accessible à un large public. Ce système sera dédié à l'authentification et à l'identification des individus par le biais de leur biométrie vocale. Pour concrétiser cette vision, nous avons défini des objectifs spécifiques clés, à savoir :

- Mettre en place une procédure robuste de sauvegarde et d'organisation du dataset, garantissant la qualité et la pertinence des données utilisées pour l'apprentissage du modèle ;
- Sélectionner judicieusement les outils, technologies, et l'architecture appropriés pour le développement du système, en tenant compte des meilleures pratiques et des exigences spécifiques de la biométrie vocale ;
- Appliquer des techniques d'apprentissage profond pour former un modèle capable d'apprendre et de discerner les caractéristiques vocales uniques de chaque individu. Cela inclut l'exploration et l'optimisation des algorithmes d'apprentissage profond pertinents ;
- Concevoir et mettre en œuvre une interface RESTful permettant un accès efficace au système depuis diverses plateformes et applications. Ceci facilitera l'intégration avec d'autres services et assurera une expérience utilisateur fluide.

4. Organisation du mémoire

Outre l'introduction et la conclusion, le present memoire comprend quatre (04) chapitres. Le premier chapitre présente l'état de l'art de l'identification par biometrie vocale. Dans le deuxième chapitre nous explorons les materiels et méthodes. Le troisième chapitre nous permet d'exposer la modelisation et le developpement de notre model d'Intelligence Artificielle s (IAs). Le quatrième chapitre nous permet de présenter les resultats découlant de nos travaux. Nous faisons ensuite une discussion et présentons les difficultés rencontreées dans la réalisation de cette étude.

Chapitre 1

État de l'art de l'identification par biométrie vocale

Introduction

L'identification par biométrie vocale est le processus de reconnaissance et de vérification de l'identité d'une personne à partir de leur enregistrement vocal. Elle a des applications importantes telles que la sécurité a criminalistique, l'Internet of Things s (IoTs), les centres d'appel, et l'interaction homme-machine[1]. Elle a été révolutionnée par les techniques d'apprentissage profond ces dernières années, permettant une identification plus précise, même dans des environnements acoustiques difficiles avec des parasites sonores. Ces techniques exploitent de grandes quantités de données vocales pour former des modèles de réseaux neuronaux qui peuvent apprendre à extraire automatiquement des caractéristiques discriminantes des signaux vocaux.

Dans ce chapitre, nous clarifierons d'abord les concepts clés liés à notre étude. Par la suite nous présenterons les quelques techniques de mise en place de ce type de modèles d'Intelligence Artificielle.

1.1 Définitions

1.1.1 Biométrie

La biométrie comprend tout un ensemble de technologies et procédés de reconnaissance, d'authentification et d'identification des personnes à partir de certaines de leurs caractéristiques physiques (empreintes digitales, la forme de la main, du doigt, le réseau veineux, l'œil), biologiques (Acide DésoxyriboNucléique s (ADNs), le sang, la salive) ou comportementales (reconnaissance vocale, la dynamique des signatures). Ces caractéristiques doivent être à la fois [2] :

- universelles : pour être utilisables par tous ;
- uniques : pour distinguer les personnes sans équivoque ;
- invariables : pour permettre une utilisation tout au long de la vie ;

- mesurables : pour permettre la comparaison.

Les différentes techniques utilisées font l'objet de recherches régulières, de développements et bien entendu, d'améliorations constantes. Toutefois, les différentes sortes de mesures n'ont pas le même niveau de fiabilité. On estime que les mesures physiologiques ont l'avantage d'être plus stables dans la vie d'un individu. Par exemple, elles ne subissent pas autant les effets du stress, contrairement à l'identification par mesure comportementale [2].

1.1.2 Biométrie vocale

l'authentification par biométrie vocale consiste à utiliser la voix d'une personne comme caractéristique biologique d'identification unique afin de l'authentifier. La Biométrie vocale est considérée comme la méthode d'authentification la plus solide car elle lie l'identité à un individu réel **("ce que vous êtes" plutôt que "ce que vous savez" ou "ce que vous avez")**. Elle extrait un ensemble de caractéristiques audio qui garantissent qu'un locuteur est une personne réelle plutôt qu'un enregistrement ou un synthétique voix d'un individu [3].

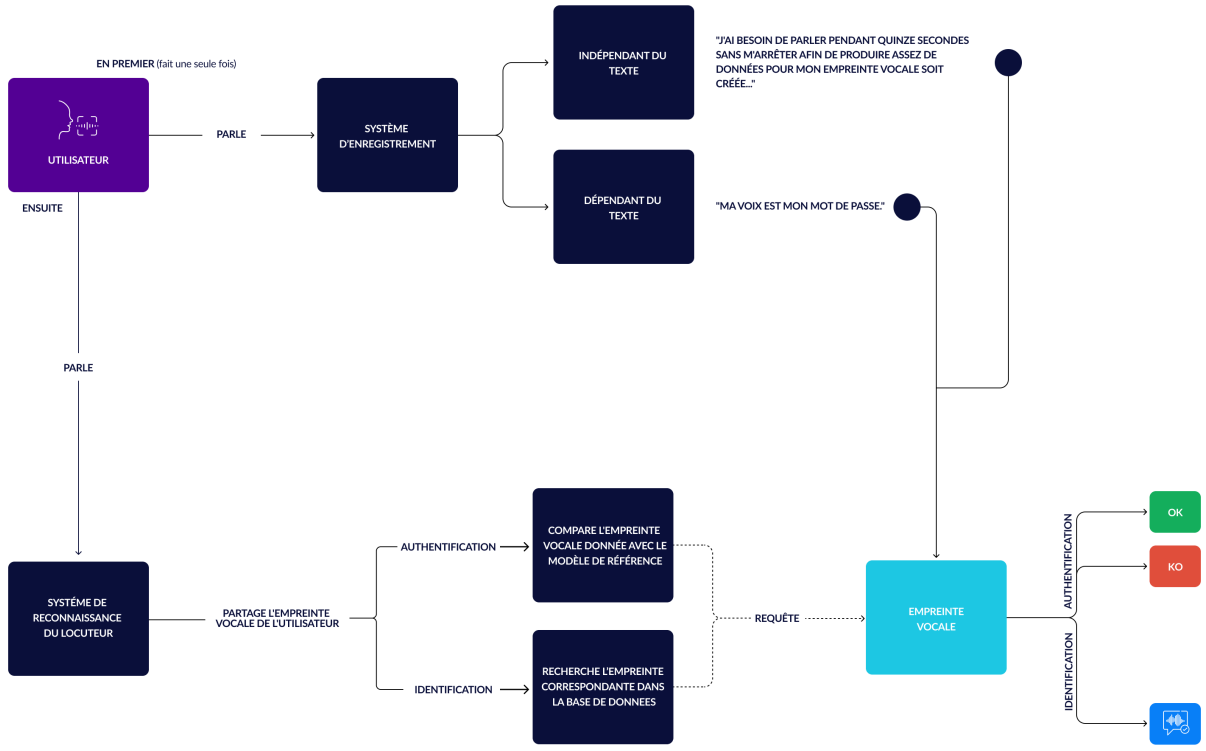


FIGURE 1.1 : Architecture d'une IA de reconnaissance par biométrie vocale.

Source: [4]

1.1.3 Identification et authentification biométrique

La biométrie permet l'identification et l'authentification d'une personne à partir de données reconnaissables et vérifiables, qui lui sont propres et qui sont uniques.

L'identification consiste à déterminer l'identité d'une personne. Il s'agit de saisir une donnée biométrique de cette personne, en prenant par exemple une photo de son visage, en enregistrant sa voix,

ou en captant l'image de son empreinte digitale. Ces données sont ensuite comparées aux données biométriques de plusieurs autres personnes qui figurent dans une base ; **ici, on essaye de répondre à la question : « qui êtes-vous ? »**.

L'authentification, appelée également vérification, est le processus qui consiste à comparer les données caractéristiques provenant d'une personne, au modèle de référence biométrique de cette dernière (« template »), afin de déterminer la ressemblance. Le modèle de référence est préalablement enregistré et stocké dans une base de données, dans un équipement ou objet personnel sécurisé. On vérifie ici que la personne présentée est bien la personne qu'elle prétend être. Dans ce cas on essaye de répondre à la question : « êtes-vous bien celui que vous prétendez être ? ».

1.1.4 Deep learning

Le deep learning ou apprentissage profond est une technique d'apprentissage automatique basée sur des réseaux de neurones artificiels. Dans le domaine de la reconnaissance vocale, l'apprentissage profond est largement utilisé pour améliorer la précision de la reconnaissance [5].

1.2 Historique de la biométrie vocale

La biométrie répond à une préoccupation très ancienne de prouver son identité, de manière irréfutable, et en utilisant nos différences et particularités.

1952 : le premier système de reconnaissance vocale a été développé par Lawrence G. Roberts et ses collègues au laboratoire Lincoln du MIT. Le système utilisait l'analyse spectrographique de fichiers vocaux comparé á une base de données de d'empreintes vocales de personnes bien identifiées[6].

1962 : les laboratoires Bell ont créé le système "voiceprint", qui utilise un spectrogramme pour analyser les modèles de fréquence uniques de la voix d'un locuteur afin de les identifier.

Années 1970 : Plusieurs systèmes d'identification de d'individus par empreinte vocale commerciaux ont été développés, y compris le système VeriVox de Raytheon et le système VoiceID de Bolt Beranek et Newman (BBN).

Années 1990 : Les chercheurs ont commencé à explorer l'utilisation de réseaux neuronaux pour l'identification des individus par voix, ce qui a permis une identification plus précise et plus fiable.

Années 2000 : L'identification des individus par voix est devenue plus largement utilisée dans les applications d'application de la loi et de sécurité, et les chercheurs ont commencé à explorer l'utilisation d'algorithmes d'apprentissage automatique pour identifier automatiquement les locuteurs sans avoir besoin d'une base de données préexistante de locuteurs connus.

Aujourd'hui, l'identification des individus par voix continue d'être un domaine de recherche actif, avec des applications dans divers domaines, notamment l'application de la loi, la sécurité et la criminalistique.

1.3 Types de reconnaissances vocales

Les principes de la biométrie vocale sont généralement classés en 2 catégories :

- dépendant du texte :
  Ce type de reconnaissance du locuteur exige que l'utilisateur dise exactement l'expression enregistré. Cela manque de flexibilité étant donné que l'utilisateur doit se souvenir de ladite expression, mais offre une précision et une rapidité intéressantes.
- independant du texte :
  Ce processus de vérification n'a pas la contrainte de contenu. L'utilisateur peut parler librement. Toutefois, l'entraînement et les tests des modèles prendront plus de temps pour atteindre la performance attendue [7].

1.4 Fonctionnement

La Biométrie vocale utilise des modèles de voix pour générer une identification unique pour chaque individu, en travaillant avec plus de 100 caractéristiques physiques et comportementales. Il comprend la vitesse de la parole, l'accent, la prononciation, l'emphase, en plus des facteurs physiques des voies, de la bouche et des voies nasales [8].

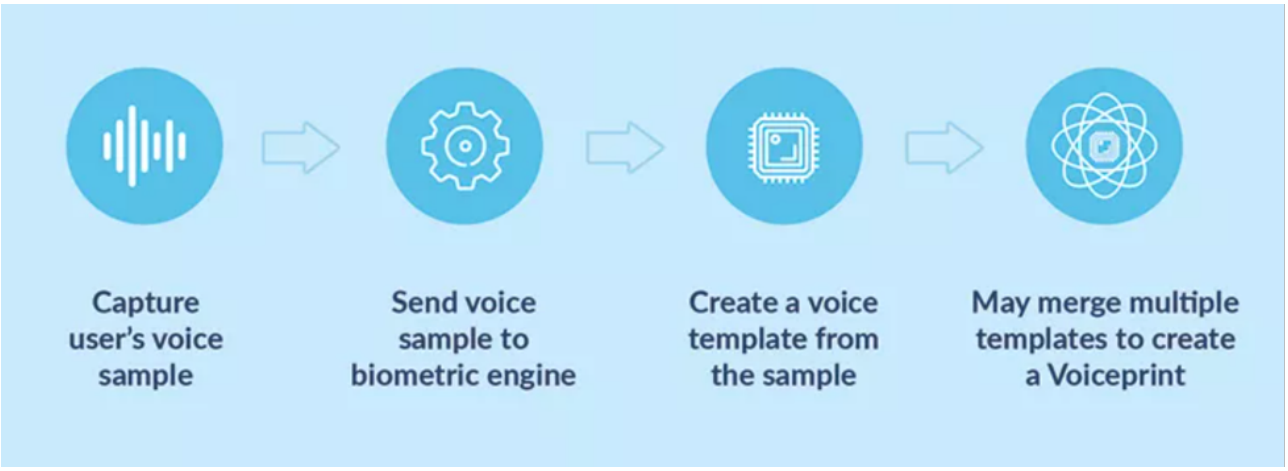


FIGURE 1.2 : Ajout d'un individu à une IA de reconnaissance parempreinte vocale.

Source: https ://imageware.io [8]

Selon la méthode d'authentification (dépendante du texte ou indépendante du texte), une IA de biométrie vocale collecte le modèle vocal d'un utilisateur [9]. Cependant, il n'authentifie pas ce que l'utilisateur dit. Il vérifie seulement qui parle. Il extrait les caractéristiques qui distinguent le discours d'une personne des autres. Le résultat est une empreinte vocale ou un modèle vocal, tel qu'on peut avoir une empreinte digitale. Cela signifie-t-il qu'une personne avec une voix similaire peut contourner le système ? La voix d'une personne est extrêmement difficile à forger à des fins de comparaison biométrique en raison de son caractère unique inhérent comme le dialecte, le style de parole etc.

Cela signifie simplement que même si une usurpation d'identité vocale est ressemblant pour une oreille humaine, une analyse détaillée de l'empreinte vocale effectuée à l'aide d'algorithmes informatiques peut aider à la distinguer de l'échantillon. Cela grâce a Plus de 100 caractéristiques physiques du corps, chacune avec une taille et une forme unique, qui contribuent à la façon dont une personne parle. La biométrie vocale repose sur des caractéristiques vocales fortement corrélées aux qualités physiologiques de la façon dont une personne crée la parole. Maintenant que nous avons étudié l'authentification vocale et son fonctionnement, jetons un coup d'œil à certaines de ses applications réelles.

1.5 Cas d'utilisation de la biométrie vocale

Des centres d'appel (call center), et des applications mobiles aux applications de messagerie et aux appareils domotiques intelligents, la biométrie vocale peut fonctionner dans divers cas d'utilisation [4].

Voici un aperçu détaillé de ceux-ci :

De toute évidence, l'authentification biométrique vocale semble rendre les choses beaucoup plus faciles. Cependant, avant de décider d'utiliser l'authentification vocale pour votre entreprise, examinons ses avantages et ses défis [10]

1.6 Les avantages de la biométrie vocale

La Biométrie vocale peut être utilisé dans le cadre d'un processus d'authentification à deux facteurs pour augmenter la sécurité et les 3 principaux avantages sont :

- Améliorer l'expérience client avec une authentification rapide et fluide
- Améliorer la sécurité et minimiser les violations dues aux mots de passe compromis, au phishing, etc…
- Identifier instantanément les utilisateurs et personnalisez l'interaction. Grâce à la technologie de Biométrie vocale, les appelants n'ont plus besoin de fournir de mots de passe ou de codes PIN ni de répondre à des questions de sécurité pour vérifier leur identité. Cela rend la biométrie vocale idéale pour les déploiements multicanaux. Une fois qu'un utilisateur est inscrit, on peut utiliser son empreinte vocale sur tous les canaux d'assistance d'une organisation [7].

1.7 Les inconvénients de la biométrie vocale

- Authentification via des contrefaçons audio :
  Les récents progrès de la technologie ont permis aux gens de créer des contrefaçons profondes. Il s'agit de fausses voix produites synthétiquement d'une personne, identiques à leur voix d'origine. Les contrefaçons profondes sont de plus en plus courantes et peuvent faire croire à un programme d'IA en son authenticité. Alors, comment empêcher les utilisateurs non autorisés

TABLE 1.1 : Cas d'utilisation de la biométrie vocale

Cas d'utilisation	**Description**	**Exemple**
les applications mobiles	Le principal cas d'utilisation de l'authentification vocale auprès des consommateurs est l'authentification mobile mains libres. Tout ce que vous avez à faire est de fournir une commande vocale pour vous connecter ou autoriser les achats, éliminant ainsi le besoin de mémoriser les identifiants et les mots de passe. Ceci est idéal pour les téléphones portables ou d'autres para- mètres où la reconnaissance faciale et d'autres formes d'authentification biométrique peuvent être gênantes.	Google Home, Alexa d'Amazon et Siri
les centres d'appels et systèmes IVR(Interactive Voice Response)	Les systèmes de biométrie vocale offrent une résistance contre l'imitation de la voix grâce à des algorithmes intrinsèques utilisés pour l'analyse bio-métrique et offrent une liste de blocage. Cela rend la technologie particulièrement utile dans le secteur de l'assistance téléphonique. On peut également utiliser la reconnaissance vocale comme authentificateur lors des appels d'assistance client. Les appelants peuvent trouver cela plus pratique et plus sûr que de partager des données personnelles telles que leur numéro de licence ou de carte de crédit à des fins de vérification d'identité.	Les Call center
les Applications web	On peut ajouter des systèmes de vérification vocale à des pages Web ou à des applications dans les secteurs de la banque et du commerce électronique. L'authentification vocale dans les applications Web peut être utile pour l'identification à distance des utilisateurs. De plus, l'inscription passive ou l'authentification indépendante du texte facilite l'intégration de nouveaux utilisateurs sans aucune inscription. Les clients sont automatiquement vérifiés en temps réel lorsqu'ils interagissent avec un agent IVR ou un centre de contact.	Call center et centres IVR
Les objets connectés	Les applications IoT offrent des moyens nouveaux et innovants de communication et d'interaction entre les humains et les machines. Une mise en œuvre appropriée de l'authentification vocale peut offrir une expérience utilisateur plus flexible que les méthodes traditionnelles telles que les écrans tactiles.	Coffre-fort , portes

d'entrer dans la base de données ? Vous pouvez créer une liste d'autorisation d'empreintes vocales et les stocker dans un répertoire actif. Au cours de ce processus, le système de reconnaissance vocale inscrit l'utilisateur dans une liste de membres autorisés. Ainsi, chaque fois

qu'un utilisateur essaie d'accéder au système, son empreinte vocale est comparée à la fois à la liste d'autorisation et à une liste de blocage des empreintes vocales des fraudeurs . Et pendant que l'authentification est en cours, la détection passive des fraudes peut envoyer des alertes si l'empreinte vocale correspond à la base de données de la liste noire [10].

- Manque de précision :
  Le bruit de fond ou son parasite est l'un des principaux facteurs qui affectent la reconnaissance automatique de la parole. Cela peut avoir un impact sur la qualité du modèle de voix de l'orateur et, à son tour, diminuer le niveau de précision du processus d'authentification. Un système d'authentification vocale peut ne pas être en mesure de faire la différence entre le discours de l'utilisateur, les autres personnes qui parlent et le bruit ambiant, ce qui entraîne des confusions et des erreurs [11].
  Cela signifie qu'il peut être difficile d'utiliser l'authentification vocale dans des environnements bruyants tels que des bureaux très fréquentés ou des espaces publics.
  Pour une authentification transparente, on peut utiliser des microphones proches ou des casques antibruit qui permettent au logiciel de se concentrer sur le discours de l'utilisateur.
  L'identification du locuteur est le processus de détermination de l'identité d'une personne en fonction de sa voix. Il s'agit d'une tâche essentielle dans de nombreuses applications, y compris l'application de la loi, la sécurité et la reconnaissance vocale.
  L'état de l'art en matière d'identification du locuteur implique l'utilisation de techniques d'apprentissage en profondeur, qui ont considérablement amélioré la précision et la fiabilité du processus. Les modèles d'apprentissage en profondeur les plus populaires pour l'identification du locuteur comprennent les réseaux de neurones convolutifs (Convolutional Neural Networks s (CNNs)), les réseaux de neurones récurrents (Recurrent Neural Networks s (RNNs)) et leurs variantes.
  L'un des principaux défis de l'identification du locuteur est de gérer la variabilité des signaux vocaux. Des facteurs tels que le bruit de fond, l'accent et l'état émotionnel peuvent affecter de manière significative la qualité des signaux vocaux, ce qui rend difficile l'identification précise du locuteur. Pour relever ce défi, les chercheurs ont développé diverses techniques d'extraction de caractéristiques, telles que les coefficients cepstraux de fréquence Mel (MFCC), qui sont couramment utilisés dans le traitement de la parole.
  Ces dernières années, les chercheurs ont également exploré l'utilisation d'autres modalités biométriques, telles que la reconnaissance faciale et la reconnaissance de l'iris, en conjonction avec l'identification du locuteur pour améliorer la précision globale du processus. Ces approches multimodales tirent parti des atouts de différentes modalités biométriques pour améliorer le processus d'identification.
  Dans l'ensemble, l'état de l'art en matière d'identification des locuteurs continue d'évoluer rapidement, les recherches en cours se concentrant sur le développement de modèles plus robustes et plus précis pour identifier les locuteurs dans des environnements difficiles [10].

1.8 Apprentissage profond

1.8.1 Notion d'intelligence artificielle

L'intelligence artificielle est née dans les années 1950, lorsqu'une poignée de pionniers du domaine naissant de l'informatique ont commencé à se demander si les ordinateurs pouvaient être amenés à "penser" - une question dont nous explorons encore les ramifications aujourd'hui. Une définition concise du domaine serait la suivante : l'effort d'automatisation des tâches intellectuelles normalement exécutées par les humains. En tant que tel, l'IA est un domaine général qui englobe l'apprentissage automatique et l'apprentissage en profondeur, mais qui comprend également de nombreuses autres approches qui n'impliquent aucun apprentissage. Les premiers programmes d'échecs, par exemple, n'impliquaient que des règles codées en dur conçues par des programmeurs et n'étaient pas considérées comme de l'apprentissage automatique.

Pendant assez longtemps, de nombreux experts ont cru que l'intelligence artificielle au niveau humain pouvait être obtenue en demandant aux programmeurs de créer à la main un ensemble suffisamment large de règles explicites pour manipuler les connaissances. Cette approche est connue sous le nom d'IA symbolique et a été le paradigme dominant de l'IA des années 1950 à la fin des années 1980. Il a atteint son apogée pendant le boom des systèmes experts des années 1980.

Bien que l'IA symbolique se soit avérée adaptée pour résoudre des problèmes logiques bien définis, comme jouer aux échecs, il s'est avéré impossible de trouver des règles explicites pour résoudre des problèmes plus complexes et flous, tels que la classification d'images, la reconnaissance vocale et le langage. Une nouvelle approche est apparue pour prendre la place symbolique de l'IA : l'apprentissage automatique.

1.8.2 Notion d'apprentissage automatique

L'apprentissage automatique découle de cette question : un ordinateur pourrait-il aller au-delà de "ce que nous savons lui ordonner d'exécuter" et apprendre par lui-même comment effectuer une tâche spécifiée ? Un ordinateur pourrait-il nous surprendre ? Plutôt que des programmeurs élaborant des règles de traitement de données à la main, un ordinateur pourrait-il automatiquement apprendre ces règles en examinant les données ? Cette question ouvre la porte à un nouveau paradigme de programmation.

En programmation classique, le paradigme de l'IA symbolique, les humains saisissent des règles (un programme) et des données à traiter selon ces règles, et en sortent des réponses (voir figure 1.2).

Avec l'apprentissage automatique, les humains saisissent les données ainsi que les réponses attendues des données, et sortent les règles. Ces règles peuvent ensuite être appliquées à de nouvelles données pour produire des réponses originales.

Un système d'apprentissage automatique est entraîné plutôt qu'explicitement programmé. Il est présenté avec de nombreux exemples pertinents pour une tâche, et il trouve une structure statistique

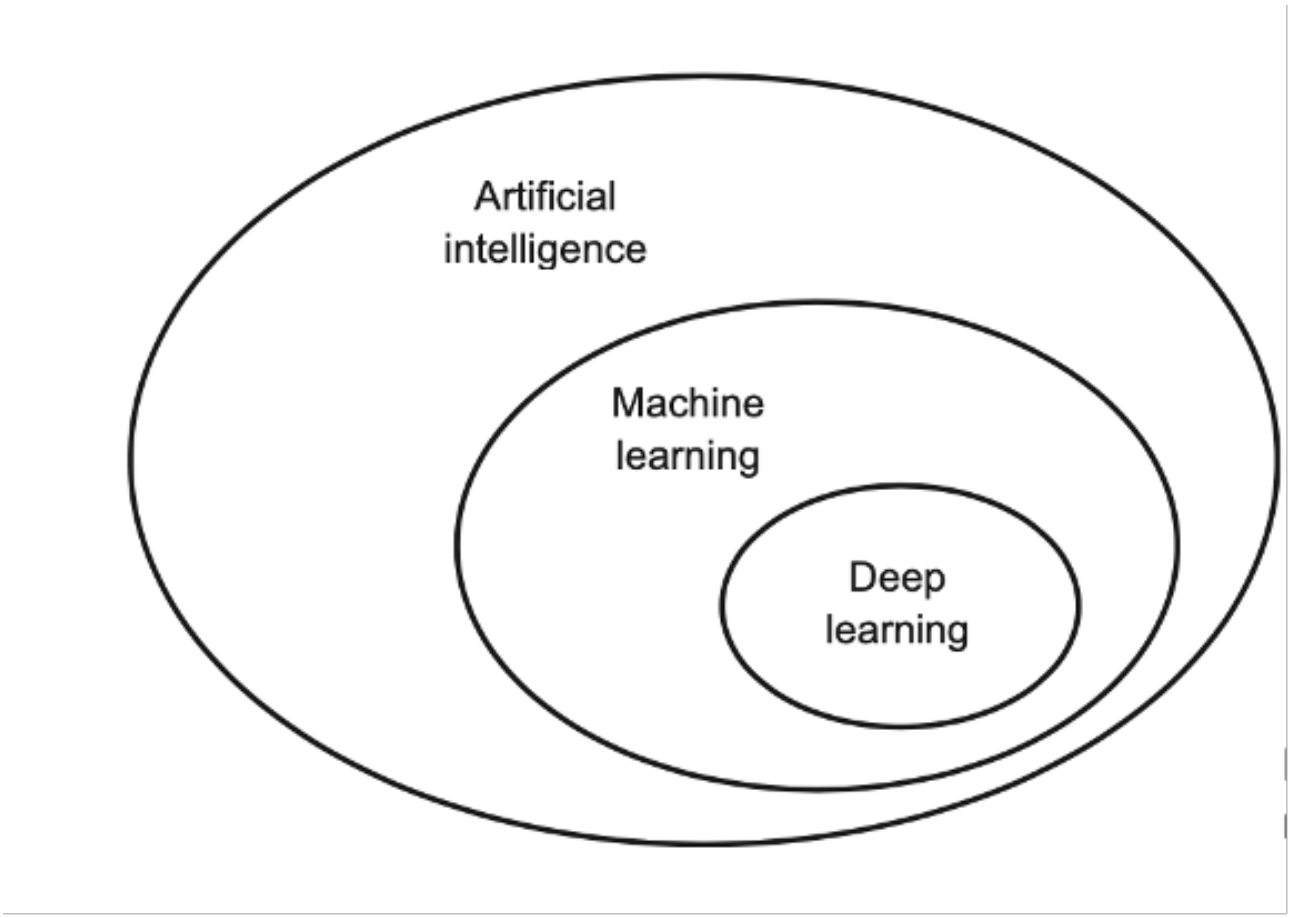


FIGURE 1.3 : Intelligence Artificielle, Machin learning et Deep learning.

Source: https ://imageware.io

dans ces exemples qui permet finalement au système de proposer des règles pour automatiser la tâche.

Par exemple, si l'on souhaite automatiser la tâche d'étiquetage de photos, on pourrait présenter un système d'apprentissage automatique avec de nombreux exemples d'images déjà étiquetées par des humains, et le système apprendrait des règles statistiques pour associer des images spécifiques à des étiquettes spécifiques.

Un système d'apprentissage automatique est entraîné plutôt qu'explicitement programmé. Il est présenté avec de nombreux exemples pertinents pour une tâche, et il trouve une structure statistique dans ces exemples qui permet finalement au système de proposer des règles pour automatiser la tâche. Par exemple, si vous souhaitez automatiser la tâche d'étiquetage de vos photos de vacances, vous pouvez présenter un système d'apprentissage automatique avec de nombreux exemples d'images déjà étiquetées par des humains, et le système apprendrait des règles statistiques pour associer des images spécifiques à des étiquettes spécifiques.

L'apprentissage automatique est étroitement lié aux statistiques mathématiques, mais il diffère des statistiques de plusieurs manières importantes. Contrairement aux statistiques, l'apprentissage automatique a tendance à traiter de grands ensembles de données complexes (tels qu'un ensemble de données de millions d'images, chacune composée de dizaines de milliers de pixels) pour lesquels une

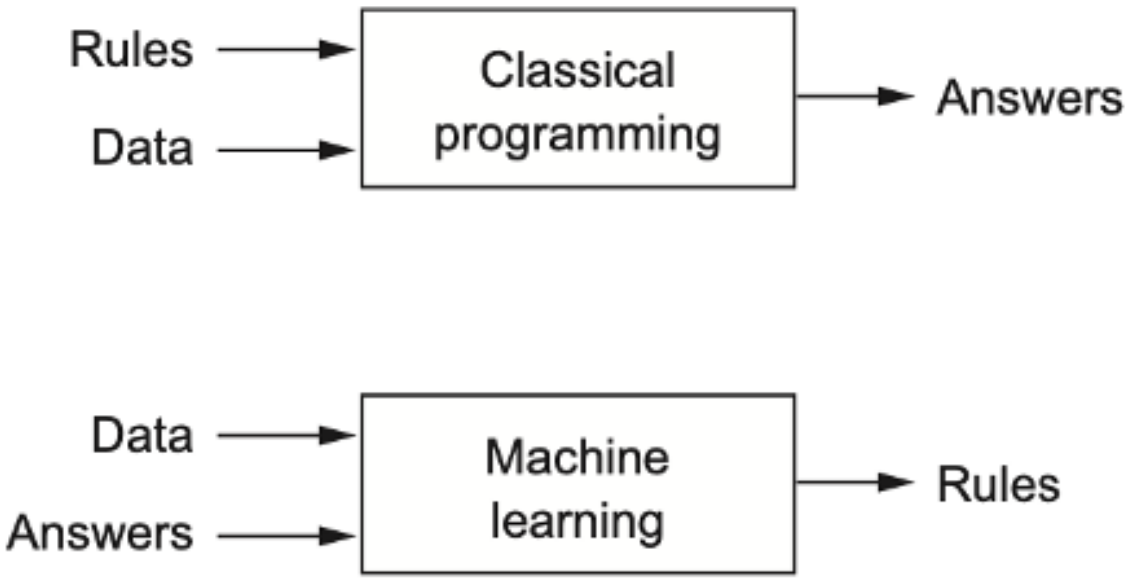


FIGURE 1.4 : Machine learning, Nouveau paradigme de programmation.

Source: https ://www.oreilly.com/library/view/deep-learning-with/9781788624336/a7a045c6-b0e2-437c-892d-1e61c11446bf.xhtml [12]

analyse statistique classique telle que l'analyse bayésienne ne serait pas pratique. En conséquence, l'apprentissage automatique, et en particulier l'apprentissage en profondeur, présente relativement peu de théorie mathématique peut-être trop peu - et est orienté vers l'ingénierie. C'est une discipline pratique dans laquelle les idées sont prouvées empiriquement plus souvent que théoriquement.

Pour résumer, Le machine learning (apprentissage automatique) est une branche de l'intelligence artificielle qui permet à un système informatique d'apprendre à partir de données et d'expériences passées, sans être explicitement programmé. Il s'agit d'une technique d'analyse de données qui utilise des algorithmes pour identifier des modèles et des tendances dans les données, puis les utilise pour faire des prédictions ou des recommandations. Le machine learning est utilisé dans de nombreux domaines, tels que la reconnaissance vocale, la reconnaissance faciale, la détection de fraudes, la recommandation de produits, etc.

1.8.3 Notions de Deep learning

1.8.3.1 définition

Le deep learning, ou apprentissage profond en français, est une branche de l'intelligence artificielle qui utilise des réseaux de neurones artificiels pour apprendre à partir de données complexes et massives. Il s'agit d'une technique d'apprentissage automatique qui permet aux machines d'apprendre à partir d'un grand nombre de données en les nourrissant à travers des couches de neurones interconnectés [13].

Dans l'apprentissage profond, ces représentations en couches sont (presque toujours) apprises via des modèles appelés réseaux de neurones, structurés en couches littérales empilées les unes sur les autres. Le terme réseau de neurones fait référence à la neurobiologie, mais bien que certains des concepts centraux de l'apprentissage en profondeur aient été développés en partie en s'inspirant de notre compréhension du cerveau, les modèles d'apprentissage en profondeur ne sont pas des modèles du cerveau. Il n'y a aucune preuve que le cerveau implémente quelque chose comme les mécanismes d'apprentissage utilisés dans les modèles modernes d'apprentissage en profondeur.

Le deep learning est utilisé dans de nombreux domaines tels que la reconnaissance d'image, la reconnaissance vocale, la traduction automatique, la prédiction de résultats, etc. Cette technologie est devenue très populaire ces dernières années grâce à ses performances exceptionnelles dans la reconnaissance de formes et dans l'analyse de données complexes.

Examinons comment un réseau de plusieurs couches de profondeur (voir la figure 1.5) transforme l'image d'un chiffre afin de reconnaître de quel chiffre il s'agit.

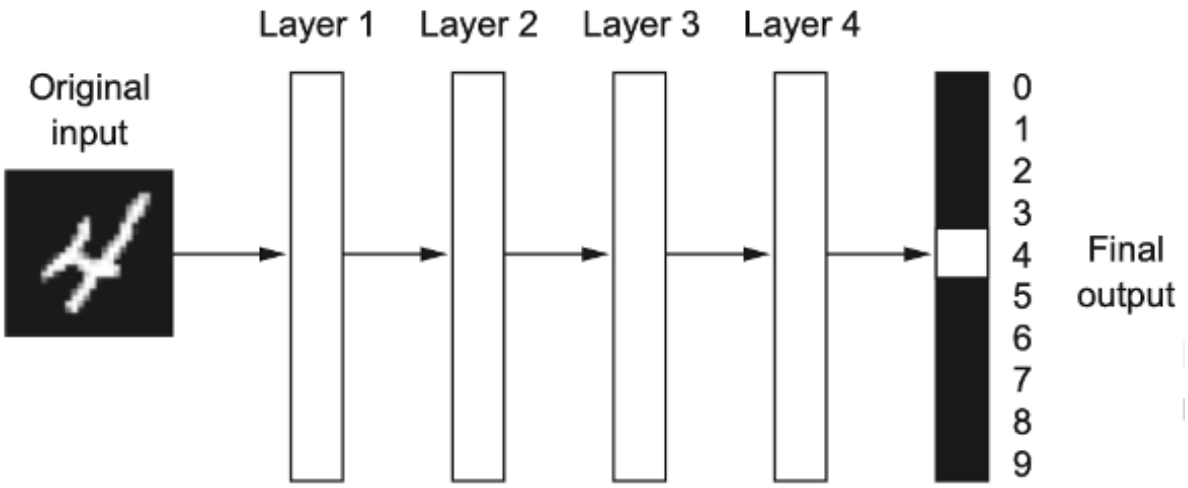


FIGURE 1.5 : Un Réseau de neurone pour la reconnaissance des caractères.

Source: [14]

Comme on peut le voir sur la figure 1.6 , le réseau transforme l'image numérique en représentations de plus en plus différentes de l'image originale et de plus en plus informatives sur le résultat final. On peut considérer un réseau profond comme une opération de distillation d'informations à plusieurs étapes, où les informations passent par des filtres successifs et sortent de plus en plus purifiées (c'est-à-dire utiles pour certaines tâches).

1.8.3.2 Fonctionnement

La spécification de ce qu'une couche fait à ses données d'entrée est stockée dans les poids de la couche, qui sont essentiellement un ensemble de nombres. En termes techniques, on dirait que la

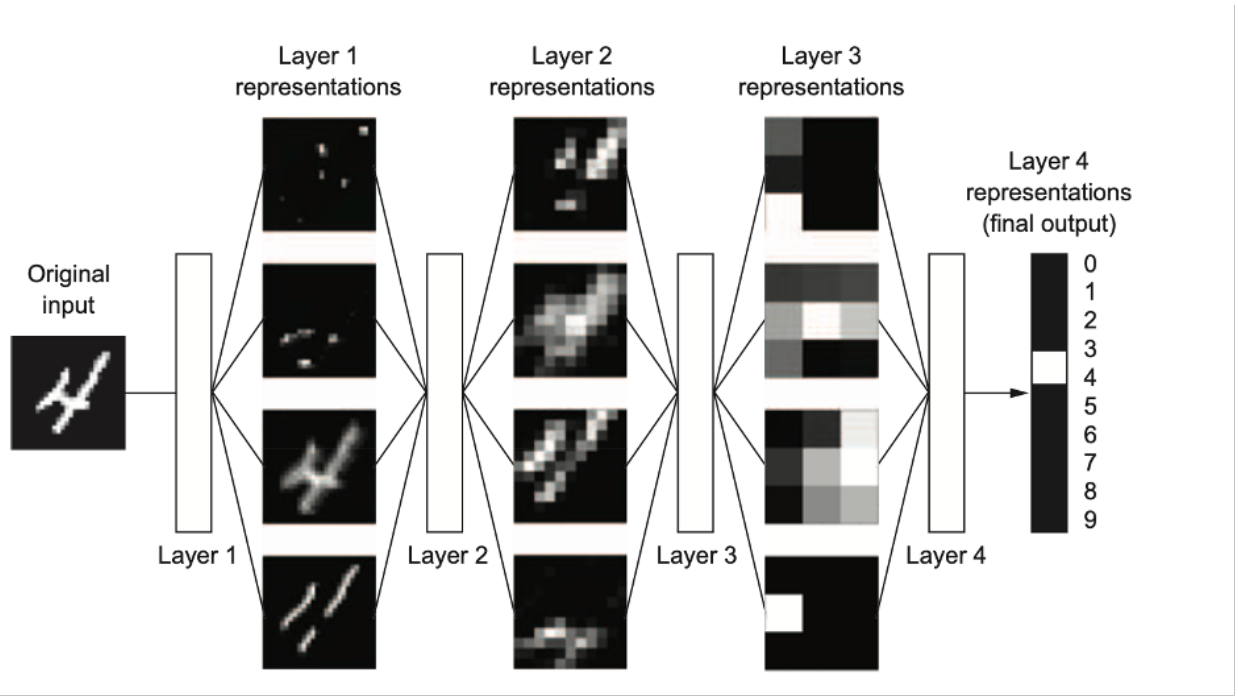


FIGURE 1.6 : modèle de classification de caractères manuscrits.

Source: [14]

transformation mise en œuvre par une couche est paramétrée par ses poids (voir figure 1.7). dans ce contexte, l'apprentissage signifie trouver un ensemble de valeurs pour les pondérations de toutes les couches d'un réseau, de sorte que le réseau mappe correctement les exemples d'entrées sur leurs cibles associées. Mais voici le problème : un réseau de neurones profonds peut contenir des dizaines de millions de paramètres. Trouver la valeur correcte pour chacun d'eux peut sembler une tâche ardue, d'autant plus que la modification de la valeur d'un paramètre affectera le comportement de tous les autres !

Pour contrôler la sortie d'un réseau de neurones, on doit être en mesure de mesurer à quelle distance cette sortie (la prédiction, Y') [13] est de ce que vous attendiez (le résultat attendu, Y). C'est le travail de la fonction de perte du réseau, également appelée fonction objectif. La fonction de perte prend les prédictions du réseau et la véritable cible (ce que vous vouliez que le réseau produise) et calcule un score de distance, capturant la performance du réseau sur cet exemple spécifique (voir figure 1.7).

L'astuce fondamentale dans l'apprentissage profond consiste à utiliser ce score comme un signal de rétroaction pour ajuster un peu la valeur des poids, dans une direction qui abaissera le score de perte pour l'exemple actuel (voir figure 1.9). Cet ajustement est le travail de l'optimiseur, qui implémente ce qu'on appelle l'algorithme de rétropropagation : l'algorithme central de l'apprentissage en profondeur.

Initialement, les poids du réseau reçoivent des valeurs aléatoires, de sorte que le réseau implémente simplement une série de transformations aléatoires. Naturellement, son rendement est loin de

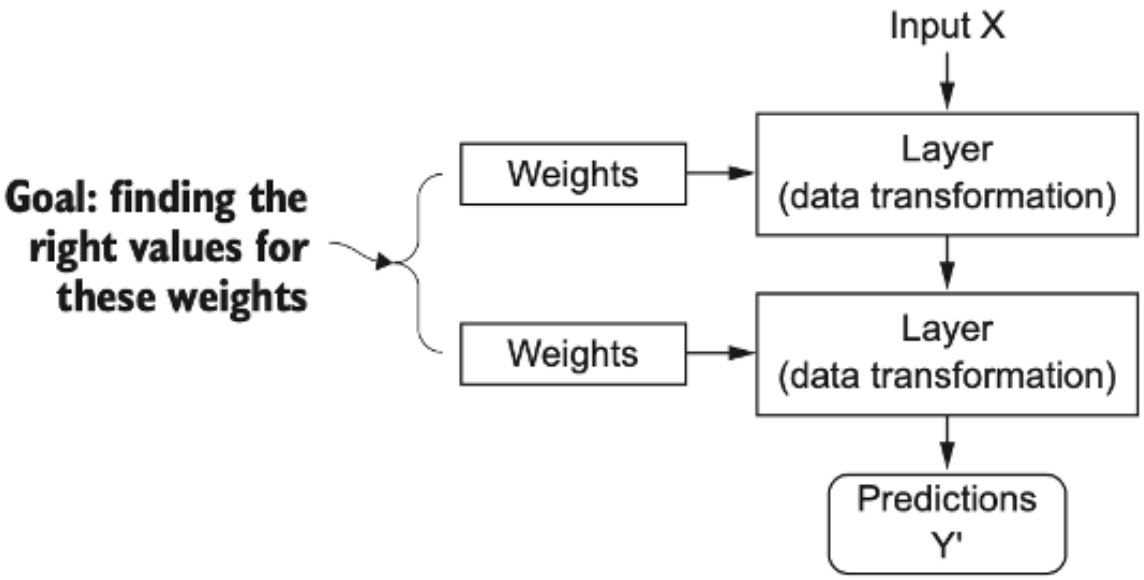


FIGURE 1.7 : Paramétrisation d'un réseau de neurones

Source: [14]

ce qu'il devrait être idéalement, et le score de perte est donc très élevé. Mais avec chaque exemple traité par le réseau, les pondérations sont légèrement ajustées dans la bonne direction et le score de perte diminue. Il s'agit de la boucle d'apprentissage qui, répétée un nombre suffisant de fois (généralement des dizaines d'itérations sur des milliers d'exemples), donne des valeurs de poids qui minimisent la fonction de perte. Un réseau avec une perte minimale est un réseau dont les sorties sont aussi proches que possible des cibles : un réseau entraîné (voir figure 1.7 et 1.8).

En résumé, le deep learning implique la collecte et la préparation des données, [13] la construction et l'entraînement du réseau de neurones, l'évaluation du modèle et enfin la prédiction. Le processus de deep learning est souvent itératif, et il peut être nécessaire d'ajuster les paramètres du modèle ou de modifier l'architecture du réseau de neurones pour améliorer les performances du modèle.

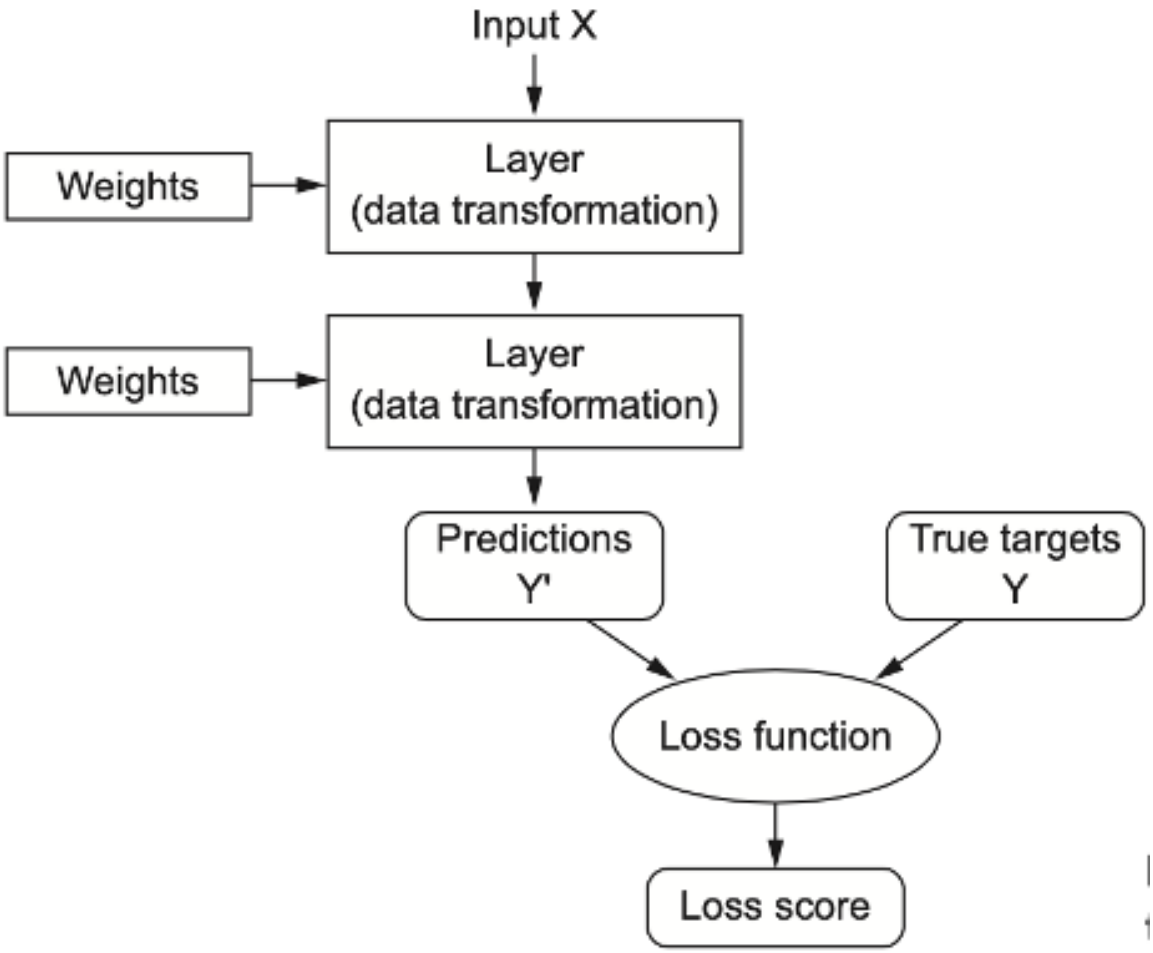


FIGURE 1.8 : Détermination de la fonction Loss

Source: [13]

1.9 Le Deep Learning pour la reconnaissance vocale

Dans le domaine de la reconnaissance de locuteur, le Deep Learning est utilisé pour améliorer la précision des systèmes de reconnaissance de la parole en analysant les caractéristiques du signal vocal. Plusieurs méthodes de Deep Learning sont utilisées pour la reconnaissance de locuteur :

- **Réseaux de Neurones Convolutionnels (CNN)** : Les CNN sont largement utilisés pour l'analyse d'images, mais ils ont également été appliqués à la reconnaissance de locuteur en utilisant des images spectrographiques du signal vocal. Les CNN peuvent extraire des caractéristiques significatives du signal vocal qui peuvent être utilisées pour identifier le locuteur.
- **Réseaux de Neurones Récurrents (RNN)** : Les RNN sont utilisés pour modéliser des séquences de données, telles que des séquences de phonèmes dans la parole. Les RNN peuvent être utilisés pour identifier le locuteur en analysant les modèles de parole spécifiques à chaque locuteur.

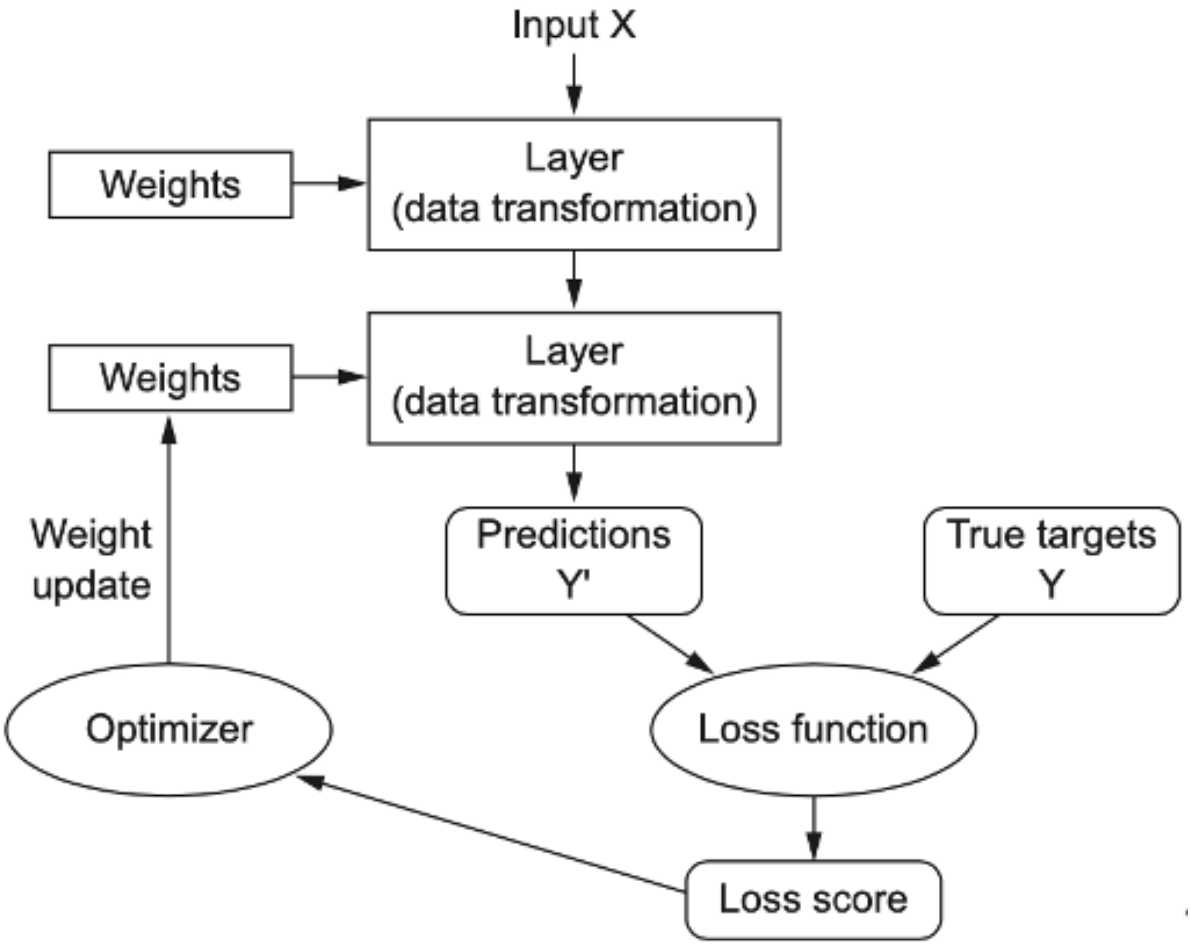


FIGURE 1.9 : Le Loss Score utilisé pour optimiser le Weigth à l'entrée

Source: [13]

- **Réseaux de Neurones Profonds (Deep Neural Networks s (DNNs))** : Les DNN sont utilisés pour l'apprentissage en profondeur en utilisant des couches de neurones artificiels. Les DNN ont été utilisés pour la reconnaissance de locuteur en analysant les caractéristiques du signal vocal telles que la fréquence, l'intensité, la durée et les modèles de parole.
- **Réseaux Adversaires Génératifs (GAN)** : Les GAN sont utilisés pour générer des données synthétiques en imitant les données réelles. Les GAN ont été utilisés pour la reconnaissance de locuteur en générant des images spectrographiques du signal vocal qui peuvent être utilisées pour entraîner des modèles de reconnaissance de locuteur.
- **Les ECAPA-TDNN (Ensemble Classifiers using Adaptive Partitioning Algorithm - Time-Delay Neural Network)** :sont un type de modèle de classification utilisé en intelligence artificielle et en reconnaissance de formes. Ils combinent les techniques d'ensemble learning (utilisation de multiples classificateurs pour améliorer la précision de la classification) avec les réseaux de neurones à retards temporels (TDNN) pour résoudre des problèmes de classification complexes. Les ECAPA-TDNN sont particulièrement adaptés pour la reconnaissance de formes dans des signaux temporels tels que les signaux audio, les séries chronologiques, les images vidéo, etc. Ils sont largement utilisés dans des domaines tels que la reconnaissance de

la parole, la reconnaissance de gestes et la vision par ordinateur [5].

- **Les PLDA (Probabilistic Linear Discriminant Analysis)** : sont des modèles d'apprentissage automatique utilisés dans la classification de données. Ils sont principalement utilisés pour l'analyse discriminante linéaire, qui est une méthode statistique qui permet de déterminer la catégorie à laquelle une observation appartient en se basant sur un ensemble de variables prédéfinies. Les PLDA sont souvent utilisés dans le domaine de la reconnaissance vocale pour l'identification des locuteurs, mais peuvent également être appliqués à d'autres types de données. Ils sont considérés comme un outil efficace pour la réduction de dimensionnalité et pour l'analyse des données à grande échelle. [5].

Les TDNN (Time-Delay Neural Networks) représentent une architecture de réseaux de neurones artificiels, qui utilisent des couches de neurones à délai temporel pour traiter des données séquentielles ou temporelles. Ces réseaux sont capables de modéliser des dépendances temporelles entre des événements, en prenant en compte l'ordre et le timing des entrées. Les TDNN sont souvent utilisés pour la reconnaissance de la parole, la classification de séries temporelles, la prédiction de séquences, etc.

En somme, ces techniques et méthodes de Deep Learning permettent d'améliorer la précision de la reconnaissance de locuteur en analysant les caractéristiques du signal vocal de manière plus fine et précise [13].

1.9.1 que représentent les x-vector dans la reconnaissance vocale ?

Les X-vector sont une représentation numérique de haut niveau d'un segment de signal vocal, qui est utilisée dans la reconnaissance vocale pour l'identification des individus. Les x-vector sont calculés à partir de caractéristiques acoustiques telles que les coefficients cepstraux de fréquence (MFCC) extraits d'un segment de parole [**?**].

Les X-vector sont généralement extraits en utilisant des réseaux de neurones profonds appelés réseaux de neurones convolutionnels (CNN) et des réseaux de neurones récurrents (RNN), qui sont entraînés à prédire l'identité du locuteur à partir des caractéristiques acoustiques. Le réseau de neurones est ensuite utilisé pour extraire les x-vector à partir de nouvelles données de parole, qui sont ensuite utilisées pour identifier l'individus.

Les X-vector ont plusieurs avantages par rapport aux méthodes traditionnelles de reconnaissance vocale basées sur les traits acoustiques. Ils peuvent être utilisés pour représenter des segments de parole de longueur variable, sont robustes aux variations de canal et peuvent être utilisés pour l'identification des individus dans des scénarios multi-locuteurs. Les x-vector ont été utilisés avec succès dans de nombreux systèmes de reconnaissance vocale, notamment dans les systèmes de sécurité et de surveillance.

1.10 Les bonnes pratiques en matière de reconnaissance vocale

Comme pour toute méthode de sécurité, il existe certaines meilleures pratiques de base que les organisations doivent suivre pour s'assurer que les informations de leurs clients ne sont pas en danger [15].

- établir une méthode d'authentification principale avant toute autre méthode ; cela signifie choisir une forme de validation, qu'il s'agisse d'un code PIN, d'un mot de passe ou d'une biométrie, avant d'accepter d'utiliser plus d'authentifications.
- obtenir le consentement explicite de l'utilisation prévue des utilisateurs pour la mesure de sécurité ; par exemple, accord d'utilisation de la reconnaissance faciale pour permettre les transactions de paiement.
- exiger la méthode d'authentification principale toutes les 72 heures.
- utiliser un pipeline entièrement sécurisé pour toutes les données biométriques et leur manipulation.
- conserver toutes les données biométriques dans un environnement sécurisé et isolé pour empêcher leur acquisition par des fraudeurs.

L'utilisation des étapes ci-dessus fournit une excellente base à toute entreprise pour assurer la sécurité des informations de ses clients. Il fournit également une base pour commencer à ajouter plus de niveaux d'authentification afin de créer un système de vérification encore plus puissant.

1.11 À quel point la reconnaissance vocale est-elle sécurisée ?

La sécurité des systèmes est généralement gérée à travers un ensemble de pratiques, de politiques et de mesures visant à protéger les données, les ressources et les utilisateurs contre les menaces et les attaques potentielles. La commodité qu'offre la reconnaissance vocale se fait-elle au détriment de la sécurité ? Il y a eu de nombreuses histoires dans les nouvelles sur les pirates capables d'infiltrer les maisons et les entreprises au détriment de leurs propriétaires en utilisant des problèmes de sécurité de reconnaissance vocale. Comme pour toute mesure de sécurité, croire qu'une seule mesure à elle seule est infaillible est imprudent [16].

Il reste encore du chemin à parcourir pour parvenir à une identification vocale absolument sécurisée. L'état actuel de la biométrie vocale est vulnérable. Des recherches ont montré que des échantillons de voix provenant de quelque chose comme une vidéo YouTube peuvent être acceptés comme modèles de parole approuvés. Les pirates ont pu enterrer des commandes malveillantes dans du bruit blanc pour contrôler les appareils à commande vocale.

Selon Lior Artzi, Directeur de la gestion des produits chez NICE, une entreprise internationale réputée dans la sécurité des systèmes informatiques, deux technologies majeures sont utilisées pour lutter contre la fraude dans l'authentification vocale : la détection de la vivacité et l'authentification continue[16].

La détection de la vivacité, comme son nom l'indique, garantit que l'empreinte digitale ou l'échantillon vocal utilisé est réel. Certaines tentatives pour contrecarrer cette mesure de sécurité sont des voix synthétiques et d'autres implémentations d'intelligence artificielle, mais elles n'ont pas encore suffisamment progressé pour provoquer une menace réelle[16].

L'authentification continue vérifie à plusieurs reprises l'identité d'un individu tout au long d'une session, plutôt qu'une seule fois. Cela aide à surmonter les problèmes potentiels tels que les appelants qui changent au milieu d'un appel téléphonique ou d'autres astuces qu'un mauvais acteur pourrait utiliser pour accéder au compte de quelqu'un[16].

Conclusion

L'authentification vocale rend les connexions aussi simples que parler. Cela peut sembler futuriste, mais c'est la sécurité qui protège les entreprises des violations quotidiennes. La reconnaissance vocale est-elle sécurisée ? Oui, la reconnaissance vocale est sécurisée, surtout par rapport aux connexions classiques qui nécessitent un nom d'utilisateur et un mot de passe. Semblable à d'autres biométries, la reconnaissance vocale est plus sécurisée car une personne doit interagir avec un identifiant plutôt que de simplement saisir un code. De toute évidence, l'authentification vocale offre de nombreux avantages, tels que des économies de coûts, une sécurité accrue et une expérience client améliorée. Cependant, cela vient avec son propre ensemble de défis. Par exemple, parce que la technologie peut être sensible aux bruits et aux signaux parasites. Elle reste tout de même un excellent choix pour accroitre la sécurités des système utilise comme une vérification supplémentaire, car elle ajoute des couches de protection supplémentaires que les codes d'accès manuels pourraient ne pas fournir. L'authentification vocale est bénéfique à la fois pour les clients et pour l'entreprise car elle élimine la frustration associée aux processus de connexion fastidieux. La commodité et la sécurité qu'elle offre en font une méthode de vérification dont l'utilisation ne fera qu'augmenter dans tous les secteurs. Dans les prochaines décennies, L'authentification par empreinte vocale est susceptible de gagner encore plus en popularité en raison des améliorations de la précision, causées en grande partie par les progrès de l'IA. Cette croissance est également due aux attentes accrues des clients pour un accès rapide et facile à l'information. La biométrie vocale permet un accès rapide, fluide et hautement sécurisé pour une gamme de cas d'utilisation allant des centres d'appels aux applications mobiles et en ligne, en passant par les chatbots, les appareils IoT et l'accès physique.

Chapitre 2

Modélisation et développement de notre modèle d'intelligence Artificielle

Introduction

Nous abordons, dans ce chapitre, la présentation de la méthodologie et des choix techniques. Nous voulons mettre en place un modèle de reconnaissance vocal. Ce model sera des réseaux de neurones profonde (DNN). Nous avons vu précédemment qu'il existait plusieurs méthodes de Deep Learning pour la reconnaissance par empreinte vocale. Dans ce chapitre, nous allons effectuer un choix et approfondir la méthode choisie. Nous allons commencer par la méthode de Deep Learning choisie et la structure que nous voulons mettre en place. Nous aborderons également les matériels ainsi que les technologies utilisées dans la mise en œuvre du modèle.

2.1 Le model

Tel que nous l'avons énoncé dans les parties précédentes, De nombreux modèles neuronaux peuvent être utilisés pour aborder ce type de tâche. Dans cette étude, nous nous concentrerons sur un classificateur TDNN (xvector).

Comme le montre la figure ci-dessous, l'architecture TDNN est basée sur la topologie à vecteur x (xvector), et introduit plusieurs améliorations pour les IA dans le domaine du traitement des signaux vocaux.

La couche de Pooling utilise un mécanisme d'attention dépendant du canal et du contexte, ce qui permet au réseau d'attirer l'attention sur différents frames par canal. Des blocs **SqueezeExcitation** unidimensionnels (SE) redimensionnent les canaux des cartes de caractéristiques de niveau de frame intermédiaires pour insérer des informations de contexte global dans les blocs de convolution qui fonctionnent localement. Ensuite, l'intégration de blocs Res2 unidimensionnels améliore les performances tout en réduisant simultanément le nombre total de paramètres en utilisant des convolutions groupées de manière hiérarchique.

Enfin, l'agrégation de fonctionnalités multicouches (MFA) fusionne des informations complémentaires avant le Pooling de statistiques en concaténant la carte de caractéristiques de niveau de frame finale avec des cartes de caractéristiques intermédiaires de couches précédentes.

Le réseau est entraîné en optimisant la perte AAMsoftmax sur les identités de locuteurs dans le corpus d'entraînement. L'AAM-softmax est une amélioration puissante par rapport à la perte softmax régulière dans le contexte des problèmes de classification et de vérification fine. Il optimise directement la distance cosinus entre les embeddings de locuteur.

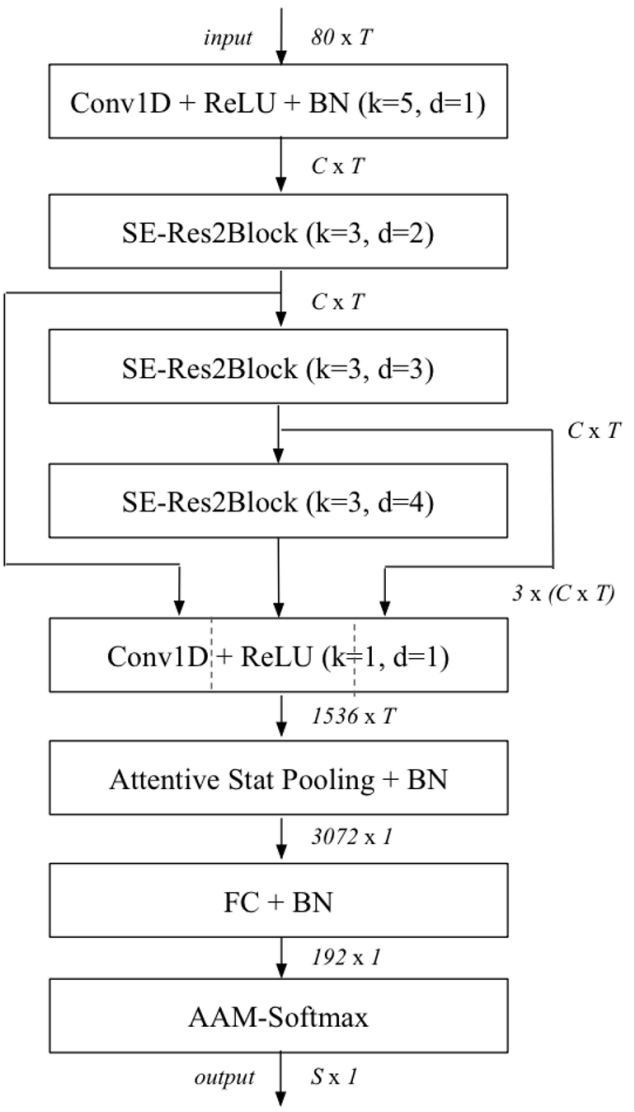


FIGURE 2.1 : Structure d'un RN de type TDNN

Source: https ://speechbrain.io

2.2 Les données d'entrainement

L'entrainement se fera avec un petit ensemble de données open source appelé minilibrispeech à laquelle nous avons ajouté des données vocales d'un certain nombre de personnalités politiques et médias. Tous les fichiers vocaux sont au format .flac, Channel Mono et de fréquence 16 KHz.



FIGURE 2.2 : caractéristiques des enregistrements vocaux

Nous avons ainsi, un dataset d'une trentaine d'empreinte vocale, qui ne contient que quelques heures de données de formation. Nous avons aussi mis en place un data pipeline qui à partir d'un API REST peux servir à étendre le dataset.

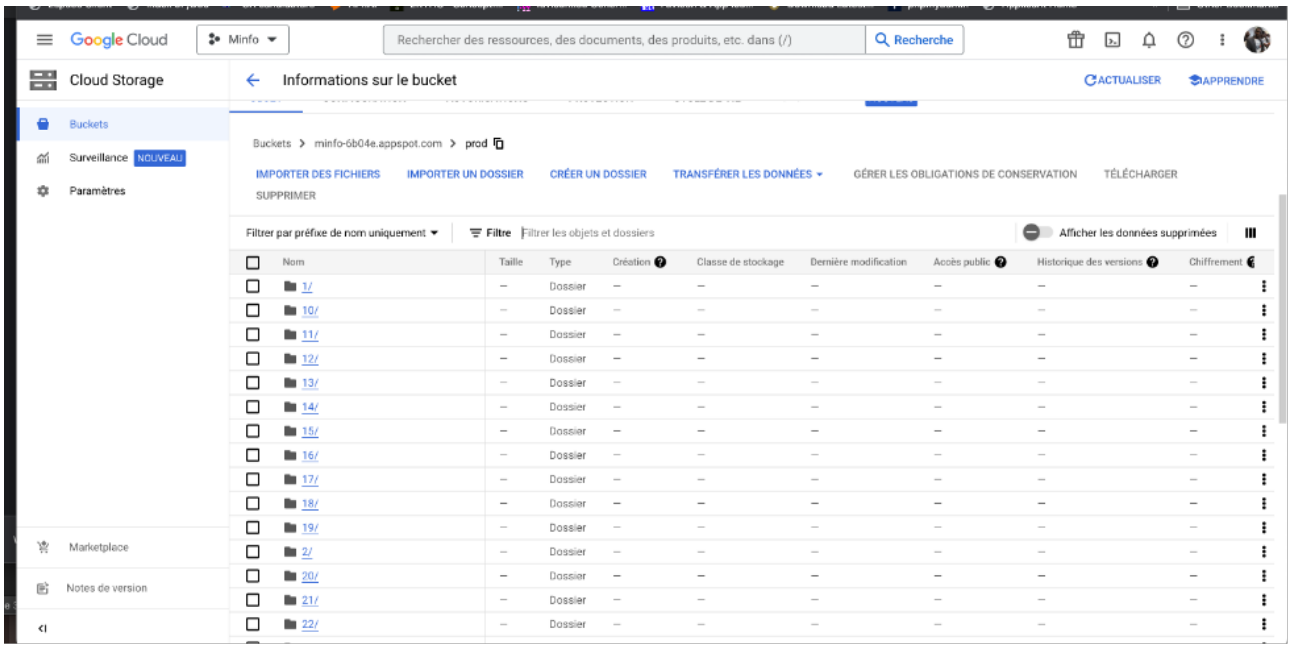


FIGURE 2.3 : Le dataset dansle bucket Cloud Storage

2.3 Le code

Pour l'entrainement, on va cloner le répertoire github.com/kadersaka/minfo-speaker-identification. On y trouve un ensemble de fichiers et de dossiers, dont :

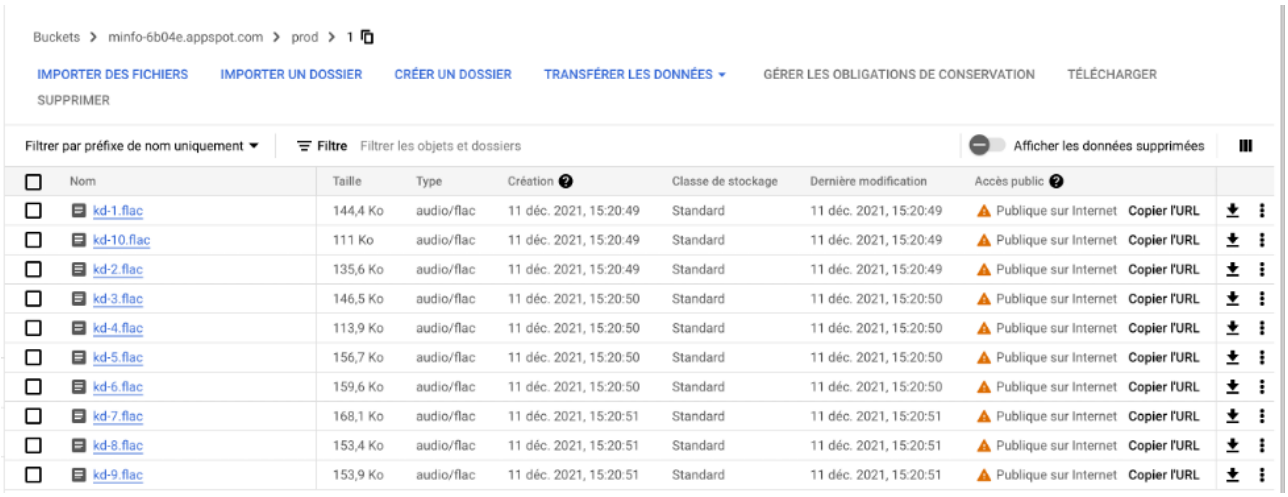


FIGURE 2.4 : enregistrements vocaux d'un seul utilisateur

- **train.py** : le fichier de code principal, décrit l'ensemble du processus d'entrainement.
- **train.yaml** : le fichier d'hyperparamètres, définit tous les paramètres d'exécution.
- **custom-model.py** : un fichier contenant la définition d'un module PyTorch dont on aura besoin pour l'inférence.
- **mini-librispeech-prepare.py** : si nécessaire, télécharge et prépare les manifestes de données si elles ne sont pas encore dans l'environnement d'entrainement.

2.4 Matériels

Le code d'entrainement sera exécuté sur Colab qui nous offre une machine linux avec toutes les ressources Recurrent Neural Networks s (CPUs), Graphics Processing Unit s (GPUs) nécessaire a une telle opération.

2.5 Description des données

Le code d'entrainement sera exécuté sur Colab qui nous offre une machine linux avec toutes les ressources CPUs, GPUs nécessaire a une telle opération.

2.6 Métriques d'évaluation

Le code d'entrainement sera exécuté sur Colab qui nous offre une machine linux avec toutes les ressources CPUs, GPUs nécessaire a une telle opération.

Le data-set sera sauvegardé sur Google Cloud Storage. En ce qui concerne le backend il sera déployé sur un backend sur le cloud.

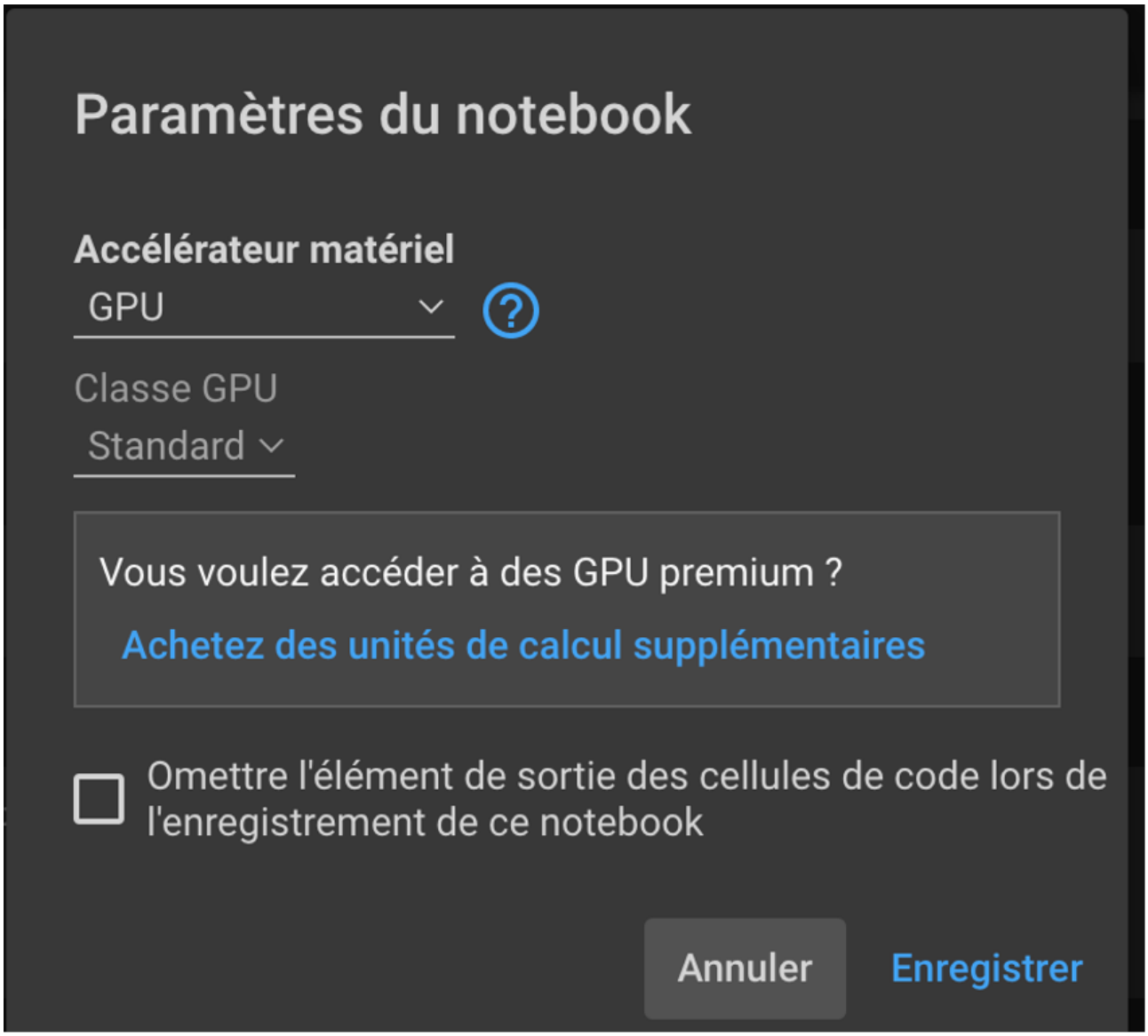


FIGURE 2.5 : paramètres d'exécution du Colab

2.7 Choix technologique

2.7.1 Définitions

La reconnaissance de locuteurs est utilisée dans de nombreux domaines, tels que la sécurité, la biométrie, la communication et l'analyse de données. Pour réaliser cette reconnaissance, plusieurs méthodes peuvent être utilisées, notamment la reconnaissance automatique de la parole, l'analyse acoustique, la modélisation de la voix et l'apprentissage automatique.

Les x-vectors sont des vecteurs d'embedding de haut niveau extraits à partir de caractéristiques acoustiques et utilisés pour la reconnaissance automatique de la parole et la vérification du locuteur. Pour extraire les x-vectors, un modèle de réseau de neurones à décalage temporel (Time-Delay Neural Network s (TDNNs)) est souvent utilisé. Le modèle TDNN pour les x-vectors est entraîné sur de grandes quantités de données de parole pour apprendre à extraire les caractéristiques acoustiques les

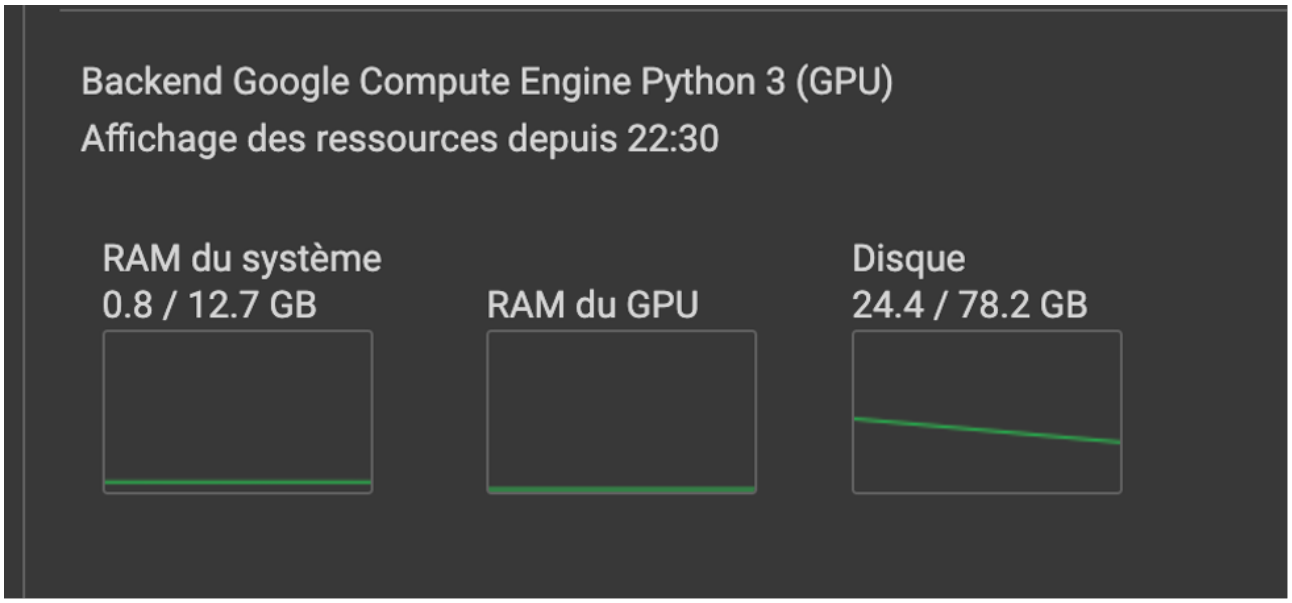


FIGURE 2.6 : Ressource de l'environnement d'exécution Colab

plus discriminantes pour la reconnaissance du locuteur. Le modèle est ensuite utilisé pour extraire les x-vectors des enregistrements de parole et ces vecteurs sont utilisés pour identifier les locuteurs ou vérifier leur identité. Nous avons vu dans le chapitre premier que plusieurs méthodes de deep learning pouvaient permettre d'entrainer des modèles de reconnaissance de locuteurs. Dans le cas de notre étude nous utiliserons les TDNN.

Le TDNN (Time-Delay Neural Network) est une méthode de traitement de la parole qui se concentre sur les caractéristiques temporelles de la parole plutôt que sur les caractéristiques fréquentielles. Le TDNN utilise une fenêtre temporelle glissante pour extraire des caractéristiques de la parole à partir de segments de durée fixe. Ces caractéristiques sont ensuite traitées par des couches de neurones, chaque couche traitant des caractéristiques de plus en plus abstraites. Le TDNN utilise également des couches de rétropropagation de l'erreur pour ajuster les poids des neurones en fonction de l'erreur de prédiction. Le TDNN est particulièrement utile pour la reconnaissance de locuteur car il est capable de traiter **les variations temporelles de la parole, telles que les pauses, les changements de ton et les variations de vitesse. Il peut également être utilisé pour détecter les caractéristiques distinctives de la voix d'un locuteur, telles que la hauteur, la durée et le débit de la parole.**

Le TDNN est également un type de réseau de neurones convolutif (CNN) qui prend en compte les caractéristiques temporelles dans les données d'entrée. Il utilise des convolutions sur des fenêtres de temps pour extraire des caractéristiques pertinentes de la parole. Le TDNN est ensuite suivi d'une couche d'agrégation globale pour résumer les caractéristiques de la parole dans un vecteur d'embedding de haut niveau (X-Vector).

En résumé, le TDNN est un outil puissant pour la reconnaissance de locuteur car il peut prendre en compte les informations temporelles de la parole et les variations de la parole dues aux différences de prononciation et de dialecte. Il peut être entraîné à reconnaître les locuteurs à partir d'enregistrements vocaux et peut être utilisé pour prédire le locuteur d'un enregistrement vocal inconnu.

2.7.2 Structure

Les TDNN sont constitués d'une couche d'entrée, de plusieurs couches cachées et d'une couche de sortie mais ils se différencie de par l'organisation des liaisons inter-couches. Les TDNN introduisent des contraintes qui leurs permettent d'avoir un certain degré d'invariance par décalage temporel et déformation. Celles-ci utilisent trois idées : poids partagés, fenêtre temporelle et délai. Les poids partagés permettent de réduire le nombre de paramètres du réseau neuronal et induisent ainsi une capacité de généralisation plus importante. Les poids sont partagés suivant la direction temporelle, c'est à dire que pour une caractéristique donnée, la fenêtre associée à celle-ci aura les mêmes poids selon la direction temporelle. De plus cette contrainte entraîne une capacité d'extraire les différences au fur et à mesure du balayage du signal. Le concept de fenêtre temporelle implique que chaque neurone de la couche l+1 n'est connecté qu'à un sous ensemble de la couche l (nous n'avons plus une connectivité totale). La taille de cette fenêtre est la même entre deux couches données. Cette fenêtre temporelle permet que chaque neurone n'est qu'une vision locale du signal, il peut être vu comme une unité de détection d'une caractéristique locale du signal. En plus des deux contraintes précédentes nous introduisons des délais entre deux fenêtre successive pour une couche donnée. De plus chaque couche a deux directions : une direction temporelle et une direction caractéristique [17].

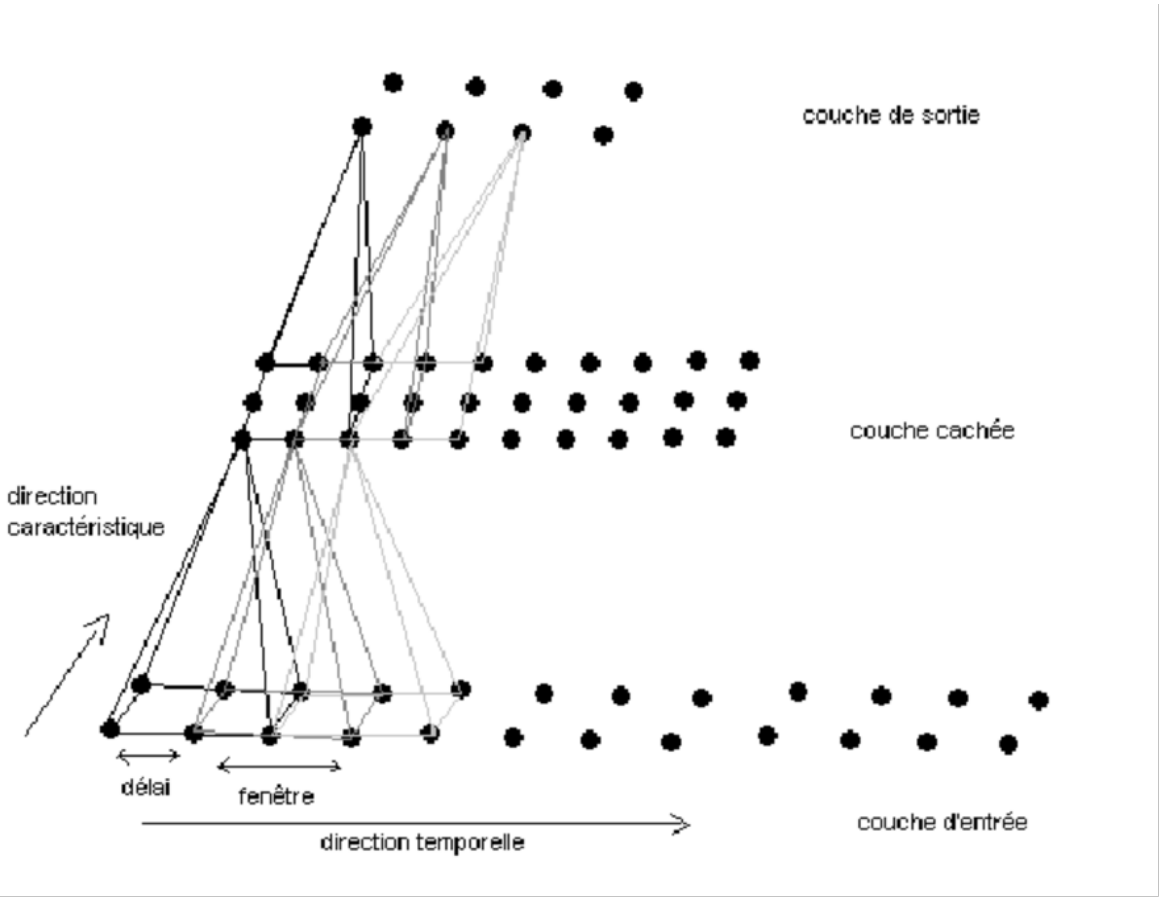


FIGURE 2.7 : Structure d'un RN de type TDNN

Source: The Marvin Project

2.7.3 Fonctionnement

Le but du TDNN est non pas d'apprendre basiquement le signal mais d'extraire les caractéristique de celui-ci. La première couche acquière le signal puis une ou plusieurs couches cachées transforment le signal en des vecteurs de caractéristiques(X-Vectors). Un neurone donné détecte une caractéristique locale de la variation de la courbe. Le champ de vision du neurone est restreint à une fenêtre temporelle limitée. Avec la contrainte des poids partagés, le même neurone est dupliqué dans la direction temps (la même matrice de poids dupliquée) pour détecter la présence ou l'absence de la même caractéristique à différentes places le long du signal. En utilisant plusieurs neurones à chaque position temporelle, le réseau de neurone effectue la détection de caractéristiques différentes : les sorties des différents neurones produisent un nouveau vecteur caractéristique pour la couche supérieure.

La composante temporelle du signal d'origine est peu à peu éliminée au fur et à mesure de sa transformation en caractéristique par les couches supérieures, pour compenser cette perte d'information on augmente le nombre de neurones dans la direction caractéristique [18].

Conclusion

Dans ce chapitre, nous avons présenté l'architecture de notre système, les matériels ainsi que les différentes technologies utilisées dans le cadre de notre travail. En ce qui concerne le choix de la méthodologie nous avons opté pour le TDNN.

Il faut retenir que le TDNN (Time Delay Neural Network) est une technique d'apprentissage profond qui est largement utilisée pour la reconnaissance de locuteurs. Cette méthode a l'avantage de traiter les signaux acoustiques dans leur temporalité, ce qui permet de prendre en compte les variations temporelles des caractéristiques acoustiques des locuteurs. Le TDNN est également capable de traiter des données de haute dimensionnalité, telles que les signaux audios, en utilisant des réseaux de neurones profonds, ce qui en fait une méthode performante pour la reconnaissance de locuteurs. Dans notre implémentation, il nous permettra d'extraire les x-vectors à partir des caractéristiques acoustiques afin de faire l'identification ou l'authentification du locuteur.

Chapitre 3

Matériels et Méthodes

Introduction

Ce chapitre est consacré à la modélisation et à l'implémentation de notre model d'IA et du backend Django. Le model permettre d'offrir deux REST API (un pour l'authentification et le second pour l'identification). Ainsi n'importe quel client web, mobile ou desktop pourra consommer les Apis. La modélisation fera l'analyse du contexte qui nous permet de sortir les différentes fonctionnalités que nous allons mettre en exergue à travers un diagramme de cas d'utilisation. Dans l'implémentation, nous abordons les détails relatifs à l'installation des outils de développement et de test, une esquisse du système, Enfin, nous présentons certaines parties du code source.

3.1 architecture générale du système

Le TDNN (Time Delay Neural Network) est un type de réseau de neurones récurrent utilisé pour la reconnaissance de la parole et la modélisation de la langue. Il est souvent utilisé dans les systèmes de reconnaissance automatique de la parole, tels que les systèmes de reconnaissance de locuteurs.

Les x-vectors sont des vecteurs de caractéristiques extraits de l'entrée audio, qui sont utilisés pour la reconnaissance de locuteurs. Les TDNN sont souvent utilisés pour extraire les x-vectors à partir de l'audio.

L'architecture d'un TDNN utilisé pour extraire les x-vectors est constituée de plusieurs couches de neurones, chacune avec une taille de fenêtre différente. Chaque couche est composée de plusieurs neurones qui sont connectés à des entrées audio avec un retard temporel.

Le retard temporel est utilisé pour permettre au réseau de traiter des informations audio qui se produisent à différents moments dans le temps. Par exemple, une couche peut être configurée pour traiter des informations audio qui ont été enregistrées 100 millisecondes avant l'instant actuel, tandis qu'une autre couche peut traiter des informations audio enregistrées 200 millisecondes avant l'instant actuel voir figure 2.7.

Les sorties de chaque couche sont ensuite combinées pour produire un vecteur de caractéristiques global, qui est utilisé comme x-vector pour la reconnaissance de locuteurs.

En résumé, l'architecture d'un TDNN utilisé pour extraire les x-vectors est généralement constituée de plusieurs couches de neurones, chacune avec une taille de fenêtre différente et connectée à des entrées audio avec un retard temporel. Les sorties de chaque couche sont combinées pour produire un vecteur de caractéristiques global, qui est utilisé comme x-vector pour la reconnaissance de locuteurs.

3.2 Installations

La première chose à faire est de créer un notebook Colab. Ensuite, nous allons activer l'exécution en mode GPU car nous en auront besoin pour l'entrainement. Puis il faudra cloner le code du répertoire GitHub ci-dessus ; (voir figure 3.1). Ensuite il faudra installer la librairie SpeechBrain (voir figure 3.2).

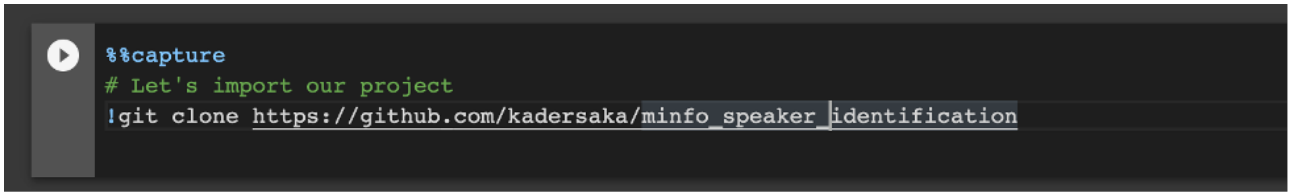


FIGURE 3.1 : enregistrements vocaux d'un seul utilisateur

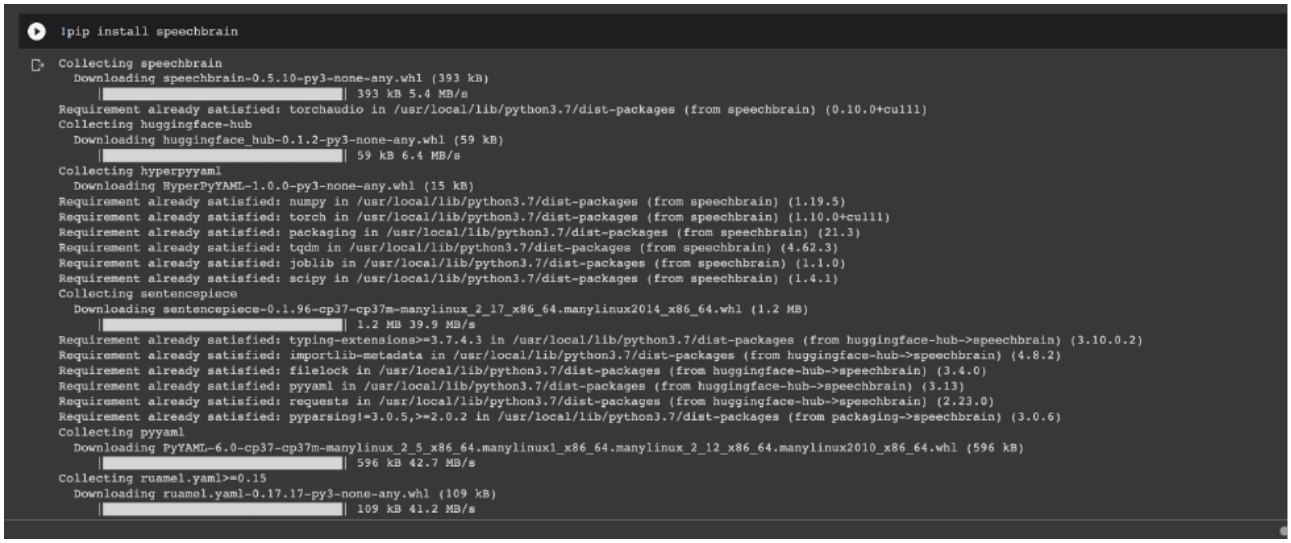


FIGURE 3.2 : enregistrements vocaux d'un seul utilisateur

3.3 Préparation des données

A cette étape la première chose à faire est de donner les autorisations au fichier Colab de pouvoir accéder au Bucket du Cloud Storage pour récupérer le dataset. Ensuite, nous utilisons GsUtils pour

la gestion des données du Bucket, notamment leur copie de cloud Storage vers l'environnement local du Colab (voir figure 3.3).

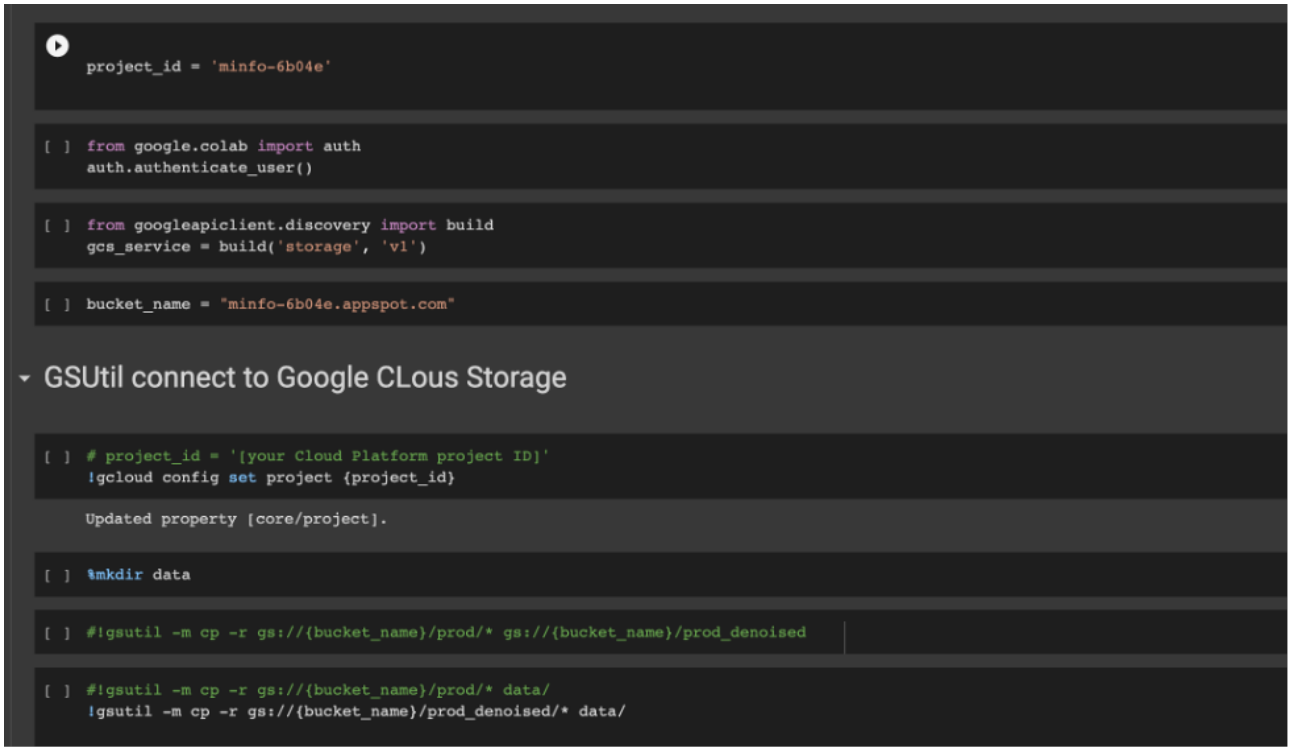


FIGURE 3.3 : récupération des données

Après avoir récupéré le dataset, nous pouvons maintenant préparer les données. L'objectif de la préparation des données est de créer les fichiers manifestes de données. Ces fichiers indiquent à SpeechBrain où trouver les données audio et leur classification au niveau de l'énoncé correspondante. Ce sont des fichiers texte écrits dans les formats populaires Comma-Separated Values s (CSVs) et JavaScript Object Notation s (JSONs) (voir figure 3.4).

Comme vous pouvez le voir, nous avons une structure hiérarchique spécifiant tous les champs nécessaires à la tâche adressée. Par exemple, nous rapportons le chemin de l'enregistrement vocal, sa durée en secondes (nécessaire si nous voulons trier les phrases avant de créer les mini-lots) et l'identité (ID) du locuteur dans l'enregistrement donné. Il faut préciser qu'a cette étape le dataset est divise en trois partitions, chacun avec son fichier manifest :

- le premier pour les l'entrainement
- le second pour les test
- et le troisième pour la validation du model

3.4 Entrainement du modèle TDNN

Nous allons former le modèle basé sur TDNN utilisé pour les X-Vector. La mise en commun statistique est utilisée au-dessus des couches convolutives (CNN) pour convertir un vocal de longueur variable en incorporations de longueur fixe. Les paramètres sont sur le fichier train.yaml

Dans ce fichier (voir figure 3.5) :

```
{
  "10dc0961-3d9b-4b5f-962e-9fe871254deb": {
    "wav": "{data_root}/28/10dc0961-3d9b-4b5f-962e-9fe871254deb.flac",
    "length": 10.0,
    "spk_id": "28"
  },
  "Roland1": {
    "wav": "{data_root}/23/Roland1.flac",
    "length": 10.96,
    "spk_id": "23"
  },
  "537d918d-7672-4a9d-95b7-402a15f5e37e": {
    "wav": "{data_root}/35/537d918d-7672-4a9d-95b7-402a15f5e37e.flac",
    "length": 9.3686875,
    "spk_id": "35"
  },
  "1088-134318-0001": {
    "wav": "{data_root}/8/1088-134318-0001.flac",
    "length": 12.235,
    "spk_id": "8"
  },
  "25210228-6f59-4c50-8e9a-191ddb60b60d": {
    "wav": "{data_root}/35/25210228-6f59-4c50-8e9a-191ddb60b60d.flac",
    "length": 12.4520625,
    "spk_id": "35"
  },
  "aea3c2eb-5bd7-4189-8e86-9eb2b2700f61": {
    "wav": "{data_root}/28/aea3c2eb-5bd7-4189-8e86-9eb2b2700f61.flac",
    "length": 10.0,
    "spk_id": "28"
  },
```


FIGURE 3.4 : Fichier manifestes d'entrainement

- Dans la première partie, nous spécifions quelques paramètres de base, tels que le seed, le chemin du dossier qui contiendra nos fichiers vocaux et le chemin du dossier de sortie (voir figure 3.6) :

- Nous spécifions ensuite le chemin des fichiers manifestes de données pour l'entrainement, la validation et le test voir figure 3.7) :
  Ces fichiers seront créés automatiquement lors de l'appel du script de préparation des données (mini_librispeech_prepare.py) à partir du fichier d'entrainement (train.py).

- Ensuite, nous configurons le train_logger et déclarons les objets error_stats qui recueilleront des statistiques sur le taux d'erreur de classification (voir figure 3.8) :

- Nous pouvons maintenant spécifier certains hyperparamètres d'entraînement tels que le nombre d'époques, la taille du lot, le taux d'apprentissage, le nombre d'époques et la dimensionnalité d'intégration (voir figure 3.9).

La variable CKPT_interval_minutes peut être utilisée pour enregistrer des points de contrôle toutes les N minutes au cours d'une période d'entraînement. Dans certains cas, une époque peut prendre plusieurs heures, et enregistrer périodiquement le point de contrôle est une bonne pratique sûre. Nous pouvons maintenant définir les modules les plus importants qui sont nécessaires pour former notre modèle (voir figure 3.10)

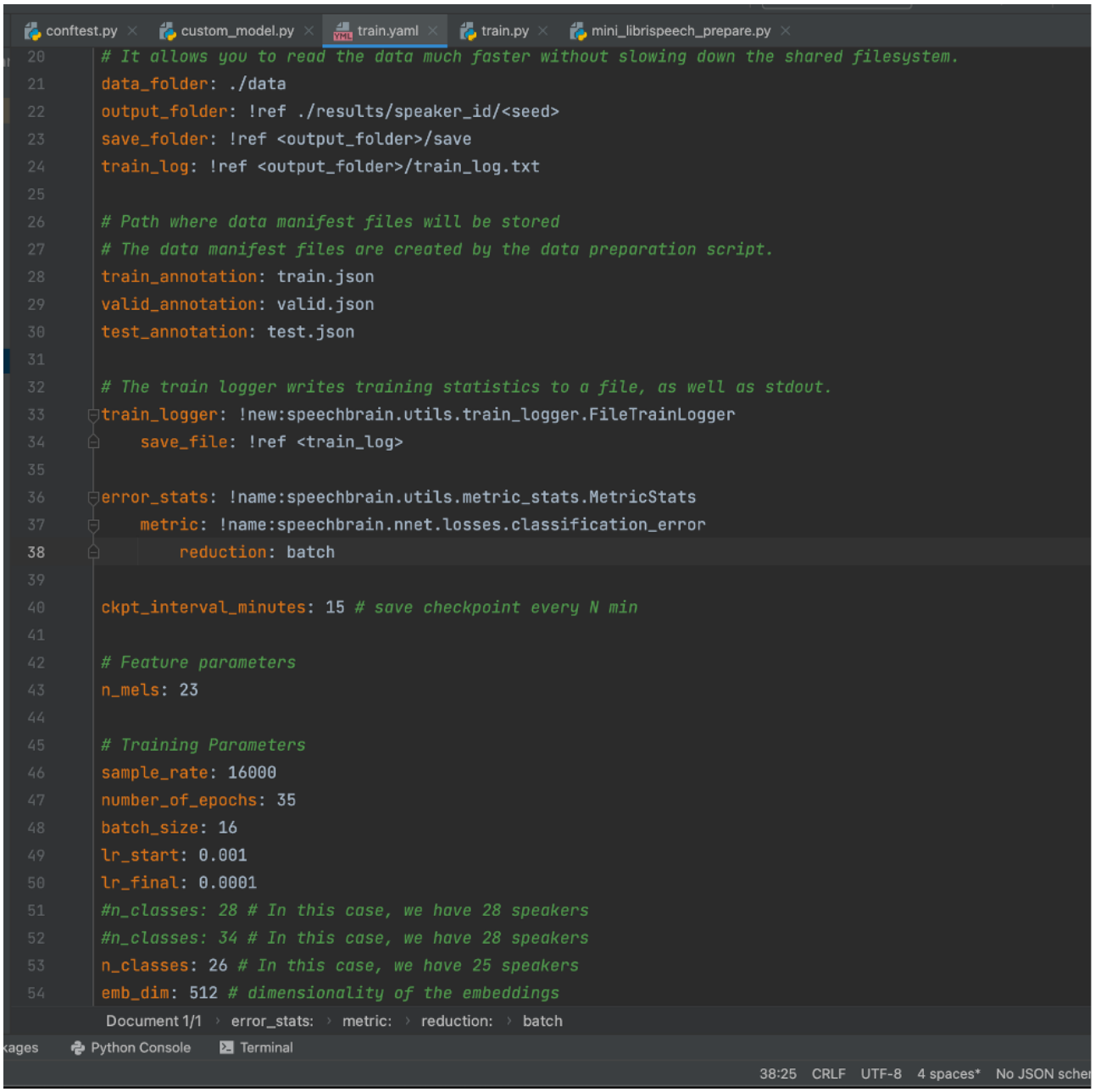


FIGURE 3.5 : Fichier des paramètres d'entrainement

La partie d'augmentation est basée à la fois sur env_corrupt (qui ajoute du bruit et de la réverbération) et augmentation (qui ajoute des pertes de temps/fréquence et un changement de vitesse). Nous terminons la spécification des hyperparamètres avec la déclaration de l'optimiseur, de l'ordonnanceur du taux d'apprentissage et du pointeur de contrôle :

Dans notre cas, nous utilisons Adam comme optimiseur et une décroissance linéaire du taux d'apprentissage sur les 15 époques. Adam (Adaptive Moment Estimation) est un optimiseur très populaire pour l'apprentissage profond, car il combine les avantages de deux autres optimiseurs, à savoir la descente de gradient stochastique (SGD) avec moment et la méthode d'estimation adaptative de la moyenne du carré des gradients (RMSProp). L'optimiseur Adam est très efficace pour résoudre des problèmes de gradient évanescents ou explosifs, car il utilise des moments d'ordre supérieur pour adapter le taux d'apprentissage pour chaque paramètre. En pratique, l'utilisation de la décroissance linéaire du taux d'apprentissage avec l'optimiseur Adam peut améliorer la stabilité et

```yaml
# Seed needs to be set at top of yaml, before objects with parameters are made
seed: 1986
__set_seed: !!python/object/apply:torch.manual_seed [!ref <seed>]

# If you plan to train a system on an HPC cluster with a big dataset,
# we strongly suggest doing the following:
# 1- Compress the dataset in a single tar or zip file.
# 2- Copy your dataset locally (i.e., the local disk of the computing node).
# 3- Uncompress the dataset in the local folder.
# 4- Set data_folder with the local path.
# Reading data from the local disk of the compute node (e.g. $SLURM_TMPDIR with SLURM-based clusters) is very important.
# It allows you to read the data much faster without slowing down the shared filesystem.
data_folder: ./data
output_folder: !ref ./results/speaker_id/<seed>
save_folder: !ref <output_folder>/save
train_log: !ref <output_folder>/train_log.txt
```

FIGURE 3.6 : paramètres de base d'entrainement

```yaml
# Path where data manifest files will be stored
# The data manifest files are created by the data preparation script.
train_annotation: train.json
valid_annotation: valid.json
test_annotation: test.json
```

FIGURE 3.7 : paramètres des chemins des fichiers manifestes

la convergence du modèle, en particulier pour les problèmes de grande envergure avec des données complexes. Cependant, il est important de noter que le taux d'apprentissage doit être choisi avec soin, car une décroissance trop rapide peut conduire à une convergence lente ou à des minima locaux, tandis qu'une décroissance trop lente peut entraîner une instabilité ou une divergence. Voici donc comment les objets, les fonctions et les hyperparamètres déclarés dans le fichier yaml sont utilisés dans train.py pour implémenter le classifieur. Parcourrons maintenant le fichier principal train.py (voir figure 3.12) :

Nous effectuons ici quelques opérations préliminaires telles que l'analyse de la ligne de commande, l'initialisation des données distribuées en parallèle (nécessaire si plusieurs GPU sont utilisés), la création du dossier de sortie et la lecture du fichier yaml afin de récupérer les paramètres par défaut. Après lecture du fichier yaml avec load_hyperpyyaml, tous les objets déclarés dans les fichiers d'hyperparamètres sont initialisés et disponibles sous forme de dictionnaire (ainsi que les autres fonctions et paramètres rapportés dans le fichier yaml). Par exemple, nous aurons hparams['embedding_model'], hparams['classifier'], hparams['batch_size'], etc. Nous exécutons également le script de préparation des données prepare_mini_librispeech qui crée les fichiers manifestes de données(distribuées à 80% pour l'entrainement, 10% pour la validation, et 10% pour les tests). Il est encapsulé avec sb.utils.distributed.run_on_main car cette opération écrit les fichiers manifestes sur le disque.
Nous appelons ensuite la fonction qui crée les objets de l'ensemble de données pour l'entrainement, la validation et le test.

La première partie est juste une déclaration de CategoricalEncoder qui sera utilisée pour convertir les étiquettes catégorielles en leurs index correspondants. Nous exposons ensuite les fonctions de traitement audio et label. La fonction audio_pipeline prend le chemin d'un signal audio (wav) et le lit. Il renvoie un objet tensor contenant la phrase vocale lue. L'entrée en paramètre de cette fonction (c'est-à-dire le fichier wav) doit avoir le même nom que la clé correspondante dans le fichier manifeste

```
# The train logger writes training statistics to a file, as well as stdout.
train_logger: !new:speechbrain.utils.train_logger.FileTrainLogger
    save_file: !ref <train_log>

error_stats: !name:speechbrain.utils.metric_stats.MetricStats
    metric: !name:speechbrain.nnet.losses.classification_error
        reduction: batch
```

FIGURE 3.8 : paramètres des chemins des fichiers manifestes

```
ckpt_interval_minutes: 15 # save checkpoint every N min

# Feature parameters
n_mels: 23

# Training Parameters
sample_rate: 16000
number_of_epochs: 35
batch_size: 16
lr_start: 0.001
lr_final: 0.0001
n_classes: 28 # In this case, we have 28 speakers
emb_dim: 512 # dimensionality of the embeddings
dataloader_options:
    batch_size: !ref <batch_size>
```

FIGURE 3.9 : Autres paramètres

de données généré : Ex :

De même, nous définissons une autre fonction appelée label_pipeline pour traiter les étiquettes au niveau de l'énoncé et les mettre dans un format utilisable par le modèle défini. La fonction lit la chaîne spk_id définie dans le fichier JSON et l'encode avec l'encodeur catégoriel. Nous créons ensuite un objet de type DynamicItemDataset et le connectons aux fonctions de traitement définies ci-dessus. Nous définissons les clés de sortie souhaitées à exposer. Ces clés seront disponibles dans la classe brain dans la variable batch comme :

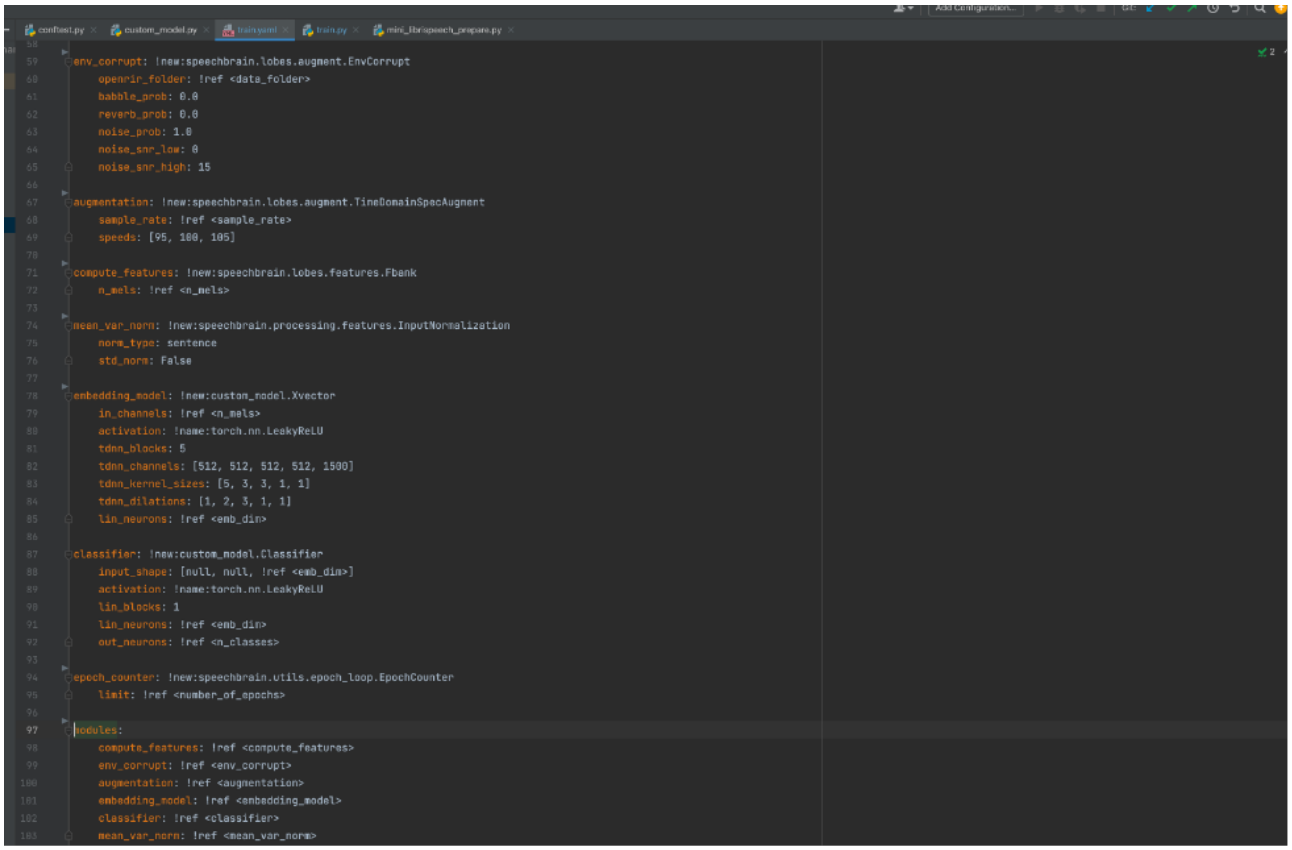
- batch.id
- lot.sig
- lot.spk_id_encoded

La dernière partie de la fonction est dédiée à l'initialisation de l'encodeur d'étiquettes. L'encodeur d'étiquettes prend en entrée le jeu de données d'apprentissage et attribue un index différent à toutes les entrées spk_id trouvées. Ces index correspondront aux index de sortie du classifieur. Après la définition des jeux de données, la fonction main peut procéder à l'initialisation et à l'utilisation de la classe brain :

La méthode d'ajustement effectue l'entrainement, tandis que le test est effectué avec celui d'évaluation. Les chargeurs de données d'entraînement et de validation sont fournis en entrée de la méthode d'ajustement, tandis que l'ensemble de données de test est introduit dans la méthode d'évaluation. Examinons maintenant les méthodes les plus importantes définies dans la classe brain :

Les Calculateurs

Commençons par la fonction forward, qui définit tous les calculs nécessaires pour transformer l'audio d'entrée en prédictions de sortie.



```
env_corrupt: !new:speechbrain.lobes.augment.EnvCorrupt
    openrir_folder: !ref <data_folder>
    babble_prob: 0.0
    reverb_prob: 0.0
    noise_prob: 1.0
    noise_snr_low: 0
    noise_snr_high: 15

augmentation: !new:speechbrain.lobes.augment.TimeDomainSpecAugment
    sample_rate: !ref <sample_rate>
    speeds: [95, 100, 105]

compute_features: !new:speechbrain.lobes.features.Fbank
    n_mels: !ref <n_mels>

mean_var_norm: !new:speechbrain.processing.features.InputNormalization
    norm_type: sentence
    std_norm: False

embedding_model: !new:custom_model.Xvector
    in_channels: !ref <n_mels>
    activation: !name:torch.nn.LeakyReLU
    tdnn_blocks: 5
    tdnn_channels: [512, 512, 512, 512, 1500]
    tdnn_kernel_sizes: [5, 3, 3, 1, 1]
    tdnn_dilations: [1, 2, 3, 1, 1]
    lin_neurons: !ref <emb_dim>

classifier: !new:custom_model.Classifier
    input_shape: [null, null, !ref <emb_dim>]
    activation: !name:torch.nn.LeakyReLU
    lin_blocks: 1
    lin_neurons: !ref <emb_dim>
    out_neurons: !ref <n_classes>

epoch_counter: !new:speechbrain.utils.epoch_loop.EpochCounter
    limit: !ref <number_of_epochs>

modules:
    compute_features: !ref <compute_features>
    env_corrupt: !ref <env_corrupt>
    augmentation: !ref <augmentation>
    embedding_model: !ref <embedding_model>
    classifier: !ref <classifier>
    mean_var_norm: !ref <mean_var_norm>
```

FIGURE 3.10 : Paramètres d'augmentation des données avec les bruits

Dans ce cas, la chaîne de calcul est très simple. Nous plaçons simplement le lot sur le bon appareil et calculons les caractéristiques acoustiques. Nous traitons ensuite les caractéristiques avec l'encodeur TDNN qui produit un objet tensor de taille fixe (notre. X-Vector). Ce dernier alimente un classifieur qui sort les probabilités a posteriori sur les N classes (ici les 31 locuteurs). L'augmentation des données est ajoutée dans la méthode prepare_features :

En particulier, lorsque la corruption (Fichiers parasites sonores) de l'environnement est déclarée dans le fichier yaml, nous concaténons dans le même lot la version propre et la version augmentée des signaux parasites. Cette approche double la taille du lot, mais elle implémente un régulateur très puissant. Le fait d'avoir à la fois la version propre et la version bruyante du signal dans le même lot force le gradient à pointer dans une direction de l'espace des paramètres qui est robuste contre les distorsions du signal.

Objectifs de calcul

Intéressons-nous maintenant à la méthode compute_objectives qui prend en entrée les cibles, les prédictions, et estime une fonction de perte :

Les prédictions en entrée sont celles calculées dans la méthode directe. La fonction de coût est évaluée en comparant ces prédictions avec les étiquettes cibles.

Les autres méthodes

Au-delà de ces deux fonctions importantes, nous avons d'autres méthodes qui sont utilisées par l'IA. Le on_state_starts est appelé au début de chaque époque et il est utilisé pour configurer les trackers de statistiques. Le on_stage_end est appelé à la fin de chaque étape (par exemple, à la fin de chaque

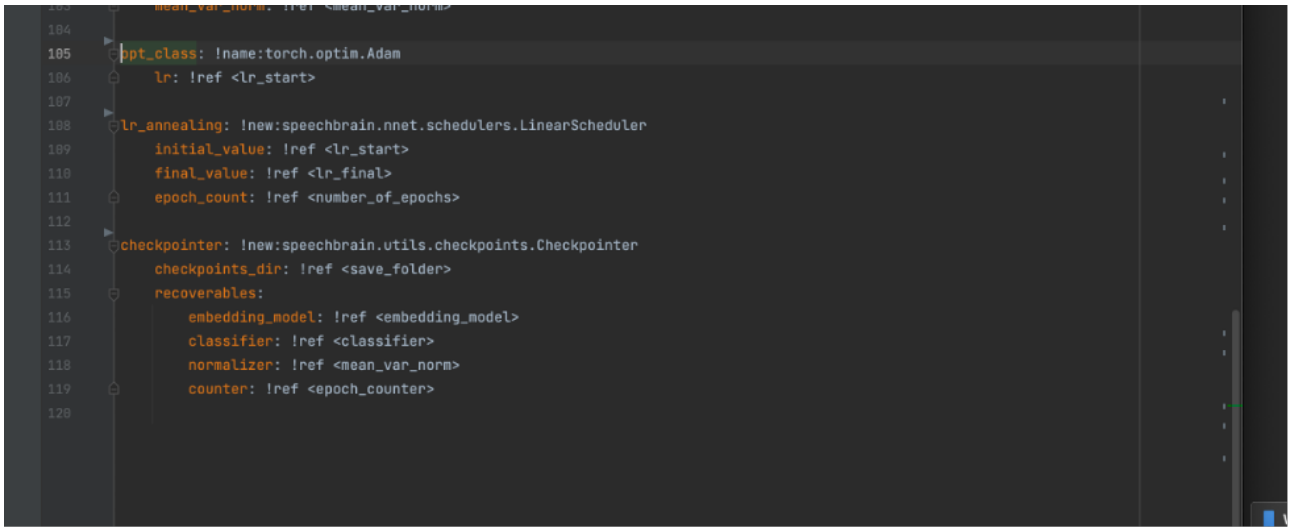


FIGURE 3.11 : Paramètres d'augmentation des données avec les bruits

période d'entraînement) et s'occupe principalement de la gestion des statistiques, du recuit du taux d'apprentissage et des points de contrôle.
Pour démarrer l'entraînement nous allons exécuter le code suivant : **python train.py train.yaml –number_of_epochs=15**

nous pouvons y passer des paramètres qui vont overrider les paramètres par défaut énoncés sur le fichier train.yaml On observe sur la console que les pertes de validation et d'entrainement (loss function) diminuent très rapidement dans les premières époques. Ensuite, nous voyons essentiellement quelques améliorations mineures et des oscillations de performances.

Voyons quels fichiers/dossiers sont générés dans le output_folder spécifié :

- **train_log.txt** : contient les statistiques (par exemple, train_loss, valid_loss) calculées à chaque époque.
- **log.txt** : est un enregistreur plus détaillé contenant les horodatages pour chaque opération de base.
- **env.log** : affiche toutes les dépendances utilisées avec leur version correspondante (utile pour la réplicabilité).
- **train.py, hyperparams.yaml** : sont une copie du fichier d'expérience avec les hyperparamètres correspondants
- **save** : est l'endroit où nous stockons le modèle généré.

Dans le dossier de sauvegarde, on a des sous-dossiers contenant les points de contrôle enregistrés lors de l'entraînement (au format CKPT+données+heure). Typiquement, on y trouve ici deux points de contrôle : le meilleur (c'est-à-dire le plus ancien) et dernier (c'est-à-dire le plus récent).

À l'intérieur de chaque point de contrôle, nous stockons toutes les informations nécessaires pour reprendre l'entrainement (par exemple, les modèles, les optimiseurs, les planificateurs, le compteur d'époques, etc.). Les paramètres des modèles d'intégration sont rapportés dans le fichier embedding_model.CKPT, tandis que ceux du classifieur sont dans classifier.CKPT. C'est juste un format binaire lisible avec torch.load. Le dossier de sauvegarde contient également l'encodeur d'étiquette (label_encoder.txt), qui mappe chaque entrée d'id de locuteur à leurs index correspondants. A la fin de l'entrainement, l'erreur de validation doit passer à zéro (ou très proche de zéro).

```
# Recipe begins!
if __name__ == "__main__":

    # Reading command line arguments.
    hparams_file, run_opts, overrides = sb.parse_arguments(sys.argv[1:])

    # Initialize ddp (useful only for multi-GPU DDP training).
    sb.utils.distributed.ddp_init_group(run_opts)

    # Load hyperparameters file with command-line overrides.
    with open(hparams_file) as fin:
        hparams = load_hyperpyyaml(fin, overrides)

    # Create experiment directory
    sb.create_experiment_directory(
        experiment_directory=hparams["output_folder"],
        hyperparams_to_save=hparams_file,
        overrides=overrides,
    )

    # Data preparation, to be run on only one process.
    sb.utils.distributed.run_on_main(
        prepare_mini_librispeech,
        kwargs={
            "data_folder": hparams["data_folder"],
            "save_json_train": hparams["train_annotation"],
            "save_json_valid": hparams["valid_annotation"],
            "save_json_test": hparams["test_annotation"],
            "split_ratio": [80, 10, 10],
        },
    )
```

FIGURE 3.12 : Code d'entrainement

```
    # Create dataset objects "train", "valid", and "test".
    datasets = dataio_prep(hparams)
```

FIGURE 3.13 : commande de separation des données

3.5 Inférence

À ce stade, nous pouvons utiliser notre meilleur model pour effectuer des prédictions sur de nouvelles données. Speechbrain dispose de la classe EncoderClassifier qui facilite l'inférence. La classe peut également être utilisée pour extraire certains plongements à la sortie de l'encodeur. Voyons d'abord comment pouvons-nous l'utiliser pour charger notre meilleur modèle xvector pour effectuer une classification des locuteurs :

Mais comment cela fonctionne-t-il avec notre model que nous avons formé auparavant ?

Exemple d'inférence pour l'authentification. le résultat est contenue dans la variable prédiction (voir figure 3.24)(True = le locuteur est bien celui dont il s'agit, False=Le locuteur n'est pas celui qu'il prétend).

Ici nous avons effectué l'inférence pour l'identification de locuteur ce qui nous a retourné le locuteur avec l'ID 23 (voir figure 3.25).

```python
def dataio_prep(hparams):
    """..."""

    ...
    label_encoder = sb.dataio.encoder.CategoricalEncoder()

    # Define audio pipeline
    @sb.utils.data_pipeline.takes("wav")
    @sb.utils.data_pipeline.provides("sig")
    def audio_pipeline(wav):...

    # Define label pipeline:
    @sb.utils.data_pipeline.takes("spk_id")
    @sb.utils.data_pipeline.provides("spk_id", "spk_id_encoded")
    def label_pipeline(spk_id):...


    # Define datasets. We also connect the dataset with the data processing
    # functions defined above.
    datasets = {}
    hparams["dataloader_options"]["shuffle"] = False
    for dataset in ["train", "valid", "test"]:
        datasets[dataset] = sb.dataio.dataset.DynamicItemDataset.from_json(
            json_path=hparams[f"{dataset}_annotation"],
            replacements={"data_root": hparams["data_folder"]},
            dynamic_items=[audio_pipeline, label_pipeline],
            output_keys=["id", "sig", "spk_id_encoded"],
        )

    ...
    lab_enc_file = os.path.join(hparams["save_folder"], "label_encoder.txt")
    label_encoder.load_or_create(
        path=lab_enc_file,
        from_didatasets=[datasets["train"]],
        output_key="spk_id",
    )

    return datasets
```

FIGURE 3.14 : Code de separation des données

3.6 Le développement des Apis

A cette étape, nous pouvons exporter notre model vers Cloud Storage afin de pouvoir l'utiliser sur notre backend Django. Le code se présente comme suit :

- UpdateModel cette classe permet de récupère le dernier model depuis Cloud Storage pour l'utilisation à des fins de reconnaissance.
  Requête : GET
  Route : https ://api.dev.minfo.com/api/speaker/updatemodel

```python
class UpdateModel(APIView):
    permission_classes = [AllowAny]
    def get(self, request):
        storage_client = storage.Client.from_service_account_json('minfo-6b04e-ed4a5427f384.json')
        buckets = list(storage_client.list_buckets())
        print(buckets)
```

```
  "10dc0961-3d9b-4b5f-962e-9fe871254deb": {
    "wav": "{data_root}/28/10dc0961-3d9b-4b5f-962e-9fe871254deb.flac",
    "length": 10.0,
    "spk_id": "28"
  },
```

FIGURE 3.15 : ligne du fichier manifeste

```
320
321 spk_id_brain = SpkIdBrain(
322     modules=hparams["modules"],
323     opt_class=hparams["opt_class"],
324     hparams=hparams,
325     run_opts=run_opts,
326     checkpointer=hparams["checkpointer"],
327 )
328
329 spk_id_brain.fit(
330     epoch_counter=spk_id_brain.hparams.epoch_counter,
331     train_set=datasets["train"],
332     valid_set=datasets["valid"],
333     train_loader_kwargs=hparams["dataloader_options"],
334     valid_loader_kwargs=hparams["dataloader_options"],
335 )
336
337 test_stats = spk_id_brain.evaluate(
338     test_set=datasets["test"],
339     min_key="error",
340     test_loader_kwargs=hparams["dataloader_options"],
341 )
342
```

FIGURE 3.16 : Classe Brain

```
7         bc_name = "minfo-6b04e.appspot.com"
8         bucket = storage_client.get_bucket(bc_name)
9
10        blobs2 = bucket.list_blobs(prefix="models/")
11        print('-----------------------blobs:')
12        print(blobs2)
13        for blob in blobs2:
14            print(blob.name)
15            # destination_uri = '{}/{}'.format("/best_model", blob.name)
16            blob.download_to_filename("speaker_identification/"+blob.name)
17
18        blobs3 = bucket.list_blobs(prefix="denoised_models/")
19        print('-----------------------blobs3:')
20        print(blobs3)
21        for blob in blobs3:
22            print(blob.name)
23            # destination_uri = '{}/{}'.format("/best_model", blob.name)
24            blob.download_to_filename("speaker_identification/"+blob.name)
25
26
27
28        data = {
29            "success": True,
30            'message': "Best trained model successfully updated",
31        }
32        return Response(data, status=status.HTTP_200_OK)
```

```
    def compute_forward(self, batch, stage):
        """..."""

        # We first move the batch to the appropriate device.
        batch = batch.to(self.device)

        # Compute features, embeddings, and predictions
        feats, lens = self.prepare_features(batch.sig, stage)
        embeddings = self.modules.embedding_model(feats, lens)
        predictions = self.modules.classifier(embeddings)

        return predictions
```

FIGURE 3.17 : compute forward

```
    def prepare_features(self, wavs, stage):
        wavs, lens = wavs

        ...
        if stage == sb.Stage.TRAIN:
            if hasattr(self.modules, "env_corrupt"):
                wavs_noise = self.modules.env_corrupt(wavs, lens)
                wavs = torch.cat([wavs, wavs_noise], dim=0)
                lens = torch.cat([lens, lens])

            if hasattr(self.hparams, "augmentation"):
                wavs = self.hparams.augmentation(wavs, lens)

        # Feature extraction and normalization
        feats = self.modules.compute_features(wavs)
        feats = self.modules.mean_var_norm(feats, lens)

        return feats, lens
```

FIGURE 3.18 : augmentation des données

- **IdentifySpeaker** : Cette classe permet d'identifier le locuteur.
  Requête : POST
  Route : https ://api.dev.minfo.com/api/speaker/id

```
class IdentifySpeaker(APIView):
    permission_classes = [AllowAny]
    def post(self, request):
        voice = self.request.FILES['voice']
        if voice:
            id_data = identify_speaker(voice)
            id_data_denoised = identify_speaker_denoised(id_data[5], id_data[4])
            id = id_data[0]
            accuracy = id_data[1]
            id2 = id_data[2]
            accuracy2 = id_data[3]
            from django.core.files.base import ContentFile, File
            denoised_voice = None

            did = id_data_denoised[0]
            daccuracy = id_data_denoised[1]
            did2 = id_data_denoised[2]
            daccuracy2 = id_data_denoised[3]

            if id:
```

```python
    def compute_objectives(self, predictions, batch, stage):
        _, lens = batch.sig
        spkid, _ = batch.spk_id_encoded

        if stage == sb.Stage.TRAIN and hasattr(self.modules, "env_corrupt"):
            spkid = torch.cat([spkid, spkid], dim=0)
            lens = torch.cat([lens, lens])

        loss = sb.nnet.losses.nll_loss(predictions, spkid, lens)

        self.loss_metric.append(
            batch.id, predictions, spkid, lens, reduction="batch"
        )

        if stage != sb.Stage.TRAIN:
            self.error_metrics.append(batch.id, predictions, spkid, lens)

        return loss
```

FIGURE 3.19 : definition de la fonction de perte

```python
                # id = int(id)+1 97073197 3600
                pooi = Pooi.objects.filter(id=id).first()
                pooi2 = Pooi.objects.filter(id=id2).first()

                pooi_data = PooiSerializer(pooi).data
                pooi2_data = PooiSerializer(pooi2).data

                dpooi = Pooi.objects.filter(id=did).first()
                dpooi2 = Pooi.objects.filter(id=did2).first()

                dpooi_data = PooiSerializer(dpooi).data
                dpooi2_data = PooiSerializer(dpooi2).data

                # scanResult = ScanResult(voice=voice, score=float(str(accuracy)), denoisedScore=float(str(accuracy2)), denoisedVoice=voice, pooi=pooi, denoisedPooi=pooi2, created_by=request.user)
                # scanResult = ScanResult(voice=voice, score=0.0, denoisedScore=0.0, denoisedVoice=voice, pooi=pooi,  denoisedPooi=pooi2, created_by=request.user)

                with open(id_data[4], 'rb') as f:
                    denoised_voice = File(f, name=os.path.basename(f.name))

                    print('--------------------------------')
                    print(type(denoised_voice))

                    scanResult = ScanResult(voice=voice,  denoisedVoice=denoised_voice, pooi=pooi,  denoisedPooi=pooi2, created_by=request.user)
                    dscanResult = ScanResult(voice=voice,  denoisedVoice=denoised_voice, pooi=dpooi,  denoisedPooi=dpooi2, created_by=request.user)
                    print('-----------brand logo type 0')
                    if request.user is not None and request.user.is_authenticated:
                        print('-----------brand logo type 1')
                        scanResult.created_by = request.user
                        print('-----------brand logo type  2')
                    print('-----------brand logo type 3')
                    scanResult.save()
```

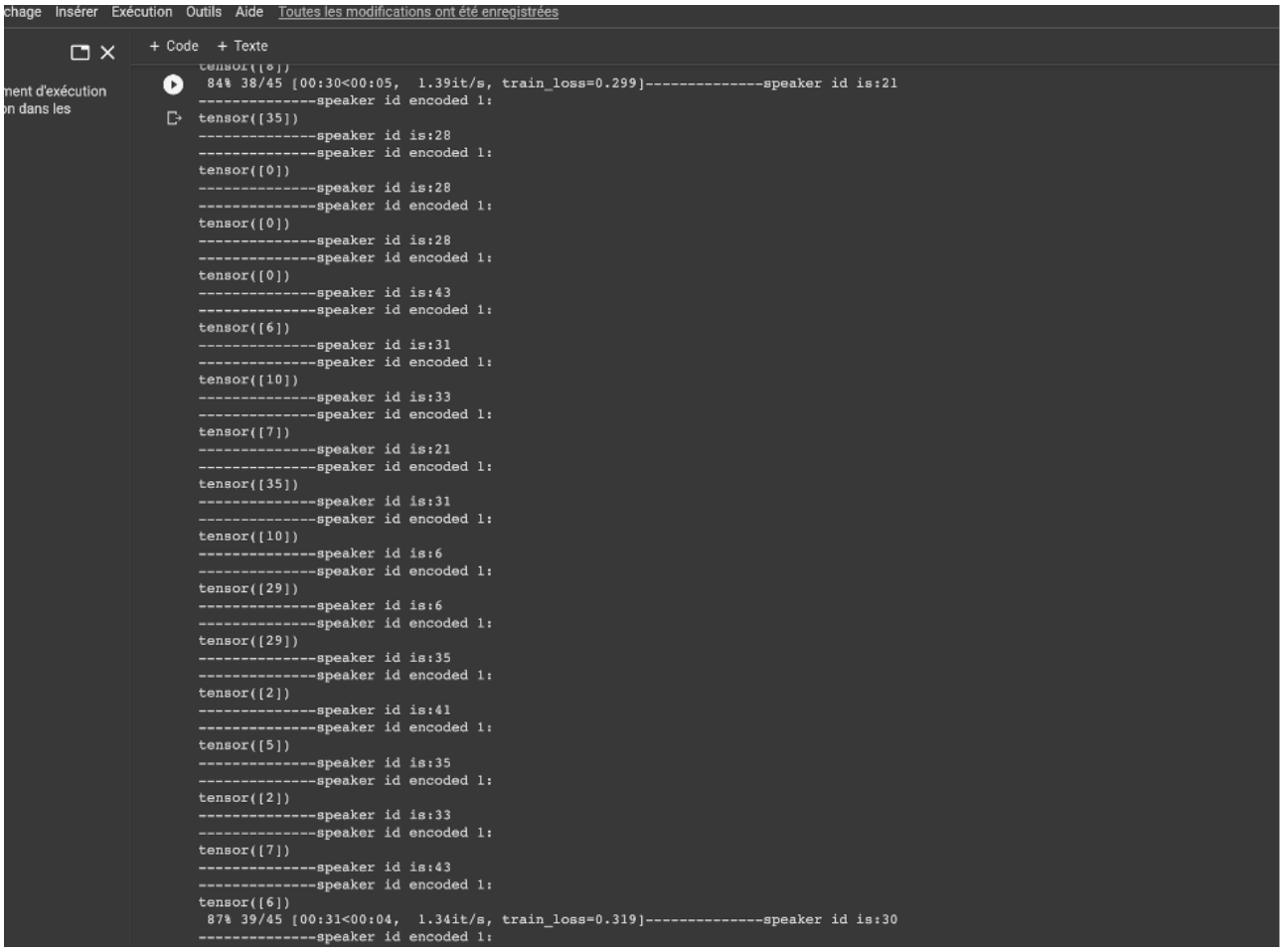


FIGURE 3.20 : Logs de l'entrainement variation de la function loss

```
                print('-----------brand logo type4')
            # score
            data = {
                'success': True,
                "score": accuracy,
                "id": id,
                "pooi": pooi_data,
                "denoised_score": accuracy2,
                "denoised_pooi": pooi2_data,
                "scanResult": ScanResultSerializer(scanResult).data,

                "dscore": daccuracy,
                "did": did,
                "dpooi": dpooi_data,
                "ddenoised_score": daccuracy2,
                "ddenoised_pooi": dpooi2_data,

            }
            return Response(data, status=status.HTTP_200_OK)
        else:
            data = {
                'message': 'POOI not found'
            }
            return Response(data, status=status.HTTP_400_BAD_REQUEST)
    else:
```

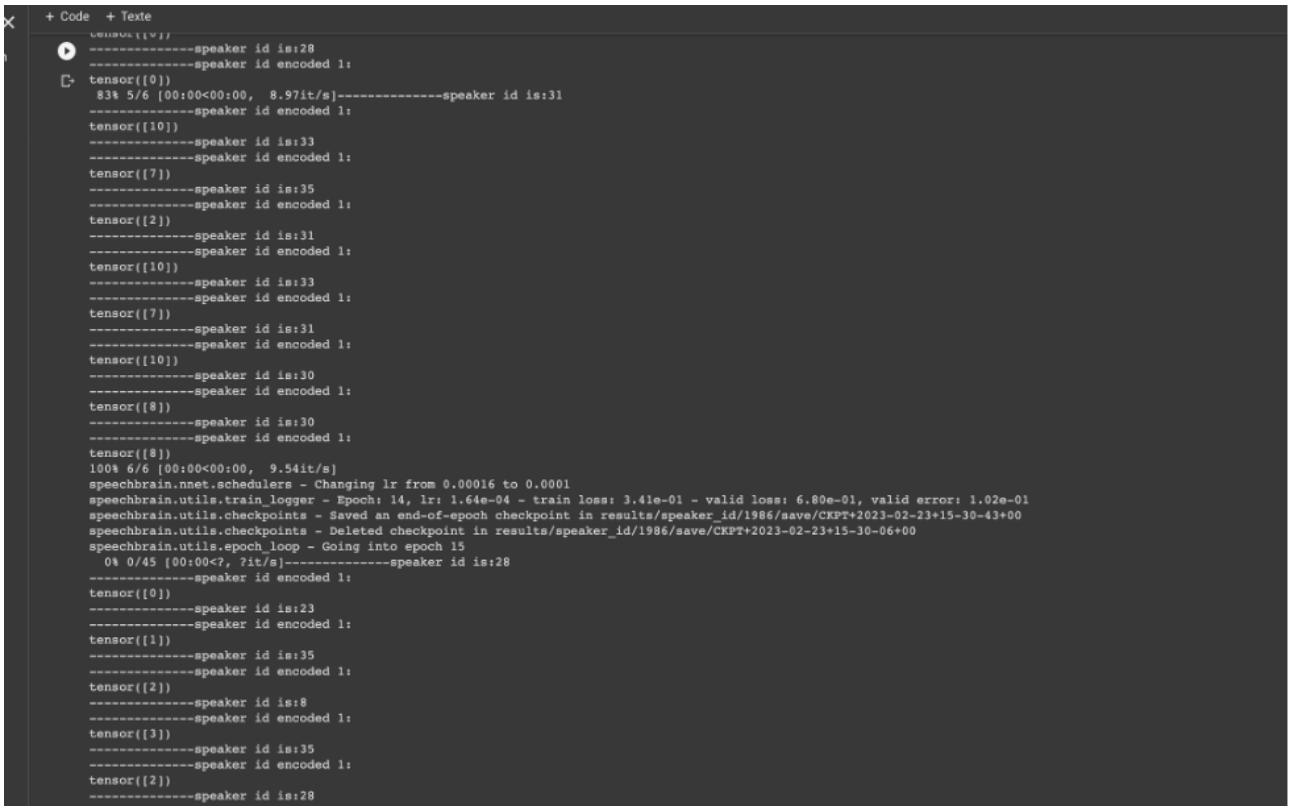


FIGURE 3.21 : Logs de l'entrainement, passage de l'Epoch 14 à l'Epoch 15

```
77              data = {
78                  'message': 'Voice file is required'
79              }
80              return Response(data, status=status.HTTP_400_BAD_REQUEST)
```

- **MatchVoices** : Elle permet d'authentifier un locuteur.
  Requête : POST
  Route : https ://api.dev.minfo.com/api/speaker/match

```
1  class MatchVoices(APIView):
2      permission_classes = [AllowAny]
3
4      def post(self, request):
5          from django.core.files.storage import FileSystemStorage
6          voice1 = self.request.FILES['voice1']
7          voice2 = self.request.FILES['voice2']
8
9          if voice2 and voice1:
10             fs = FileSystemStorage()
11
12             v1 = fs.save(voice1.name, voice1)
13             v1_path = fs.path(v1)
14
15             v2 = fs.save(voice2.name, voice2)
16             v2_path = fs.path(v2)
17
18             from speechbrain.pretrained import SpeakerRecognition
19             verification = SpeakerRecognition.from_hparams(source="speechbrain/
    spkrec-ecapa-voxceleb",
```

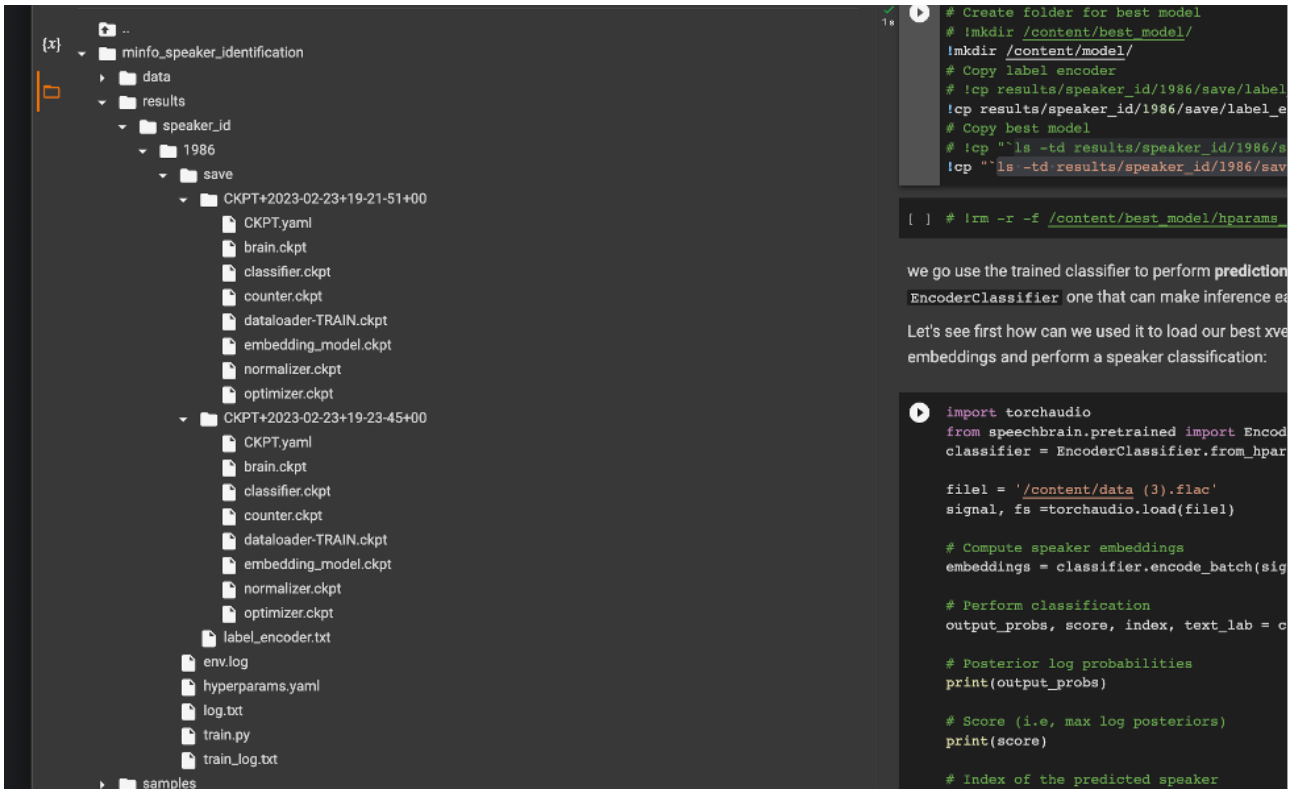


FIGURE 3.22 : Les deux models

```
                                                    savedir="
pretrained_models/spkrec-ecapa-voxceleb")
        score, prediction = verification.verify_files(v1_path, v2_path)
        # score, prediction = verification.verify_files(file1, file2)
        data = {
            # 'score': voice1.name,
            # 'prediction': voice2.name
            'score': score,
            'prediction': prediction
        }
        return Response(data, status=status.HTTP_200_OK)
    else:
        data = {
            'message': 'Hello Django REST API'
        }
        return Response(data, status=status.HTTP_400_BAD_REQUEST)
```

Conclusion

Dans cette partie, nous avons montré comment créer un model TDNN pour l'authentification et l'identification des locuteurs. Le système proposé contient tous les ingrédients de base pour développer un système de pointe (c'est-à-dire, l'augmentation de données, l'extraction de caractéristiques, l'encodage, la mise en pool statistique, le classificateur, etc.). Dans le prochain chapitre nous présenterons les résultats obtenus après notre implémentation.

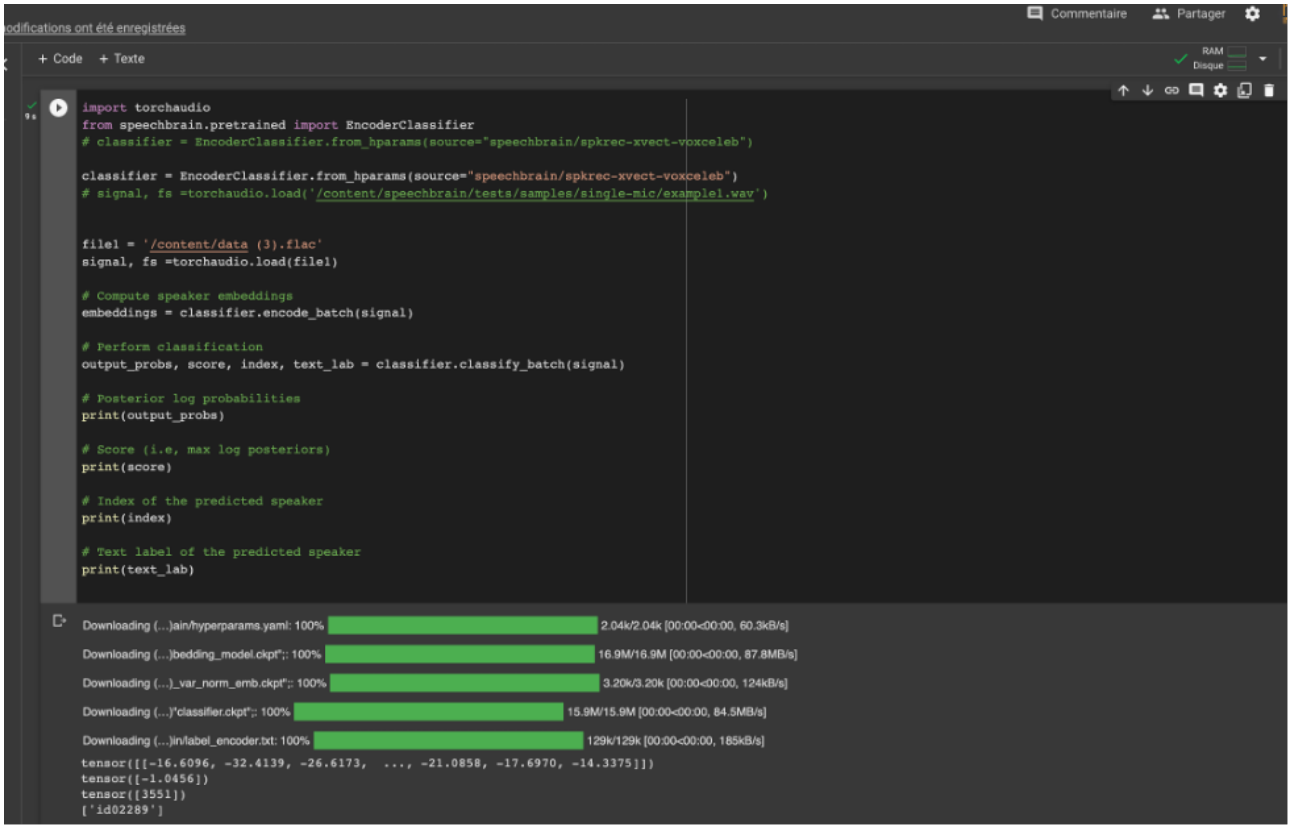


FIGURE 3.23 : Préparation du model

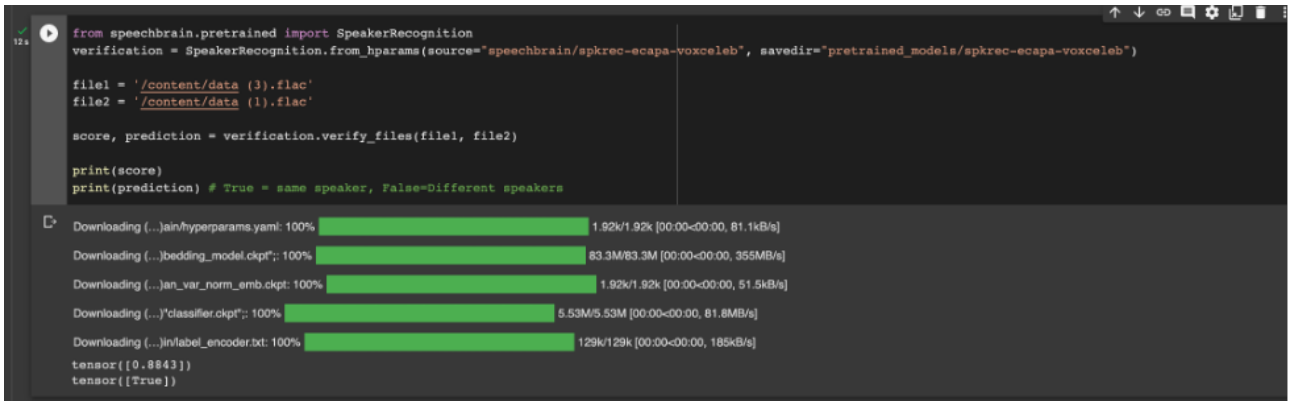


FIGURE 3.24 : Inférence sur le modèle pour l'authentification

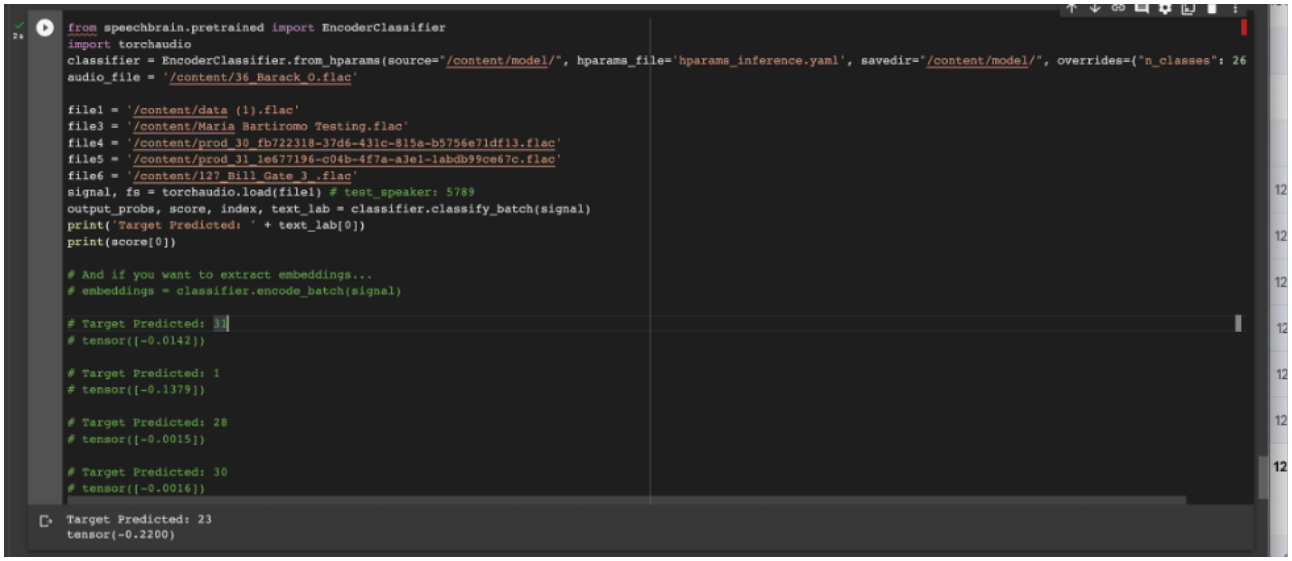


FIGURE 3.25 : Inférence sur le modèle pour l'identification

Chapitre 4

Résultats et Discussions

Introduction

Dans ce chapitre, nous présentons les différentes fonctionnalités de notre système. Les résultats obtenus sont assez satisfaisants. Ces résultats encourageants montrent des possibilités d'adoption de la reconnaissance de locuteur dans divers domaines, tels que la sécurité biométrique pour l'identification de personnes, la surveillance des appels téléphoniques, les centres d'appels et les interactions vocales dans les voitures connectées. Dans ce chapitre, nous repréciserons le cadre d'étude de notre système, ainsi que lesdits résultats, obtenus à l'issue de l'implémentation.

4.1 Présentation du cadre d'étude

Le présent système mis sur pieds a pour but la reconnaissance de locuteur. La reconnaissance de locuteur est une application de l'intelligence artificielle (IA) qui vise à identifier et à authentifier une personne en fonction de sa voix. Cette technologie peut être utilisée dans différents contextes tels que la sécurité, l'assistance vocale, la gestion de la relation client, etc. l'IA pourra être déployée et mise à la disposition des services de la criminologie pour servir lors des enquêtes judiciaires pour identifier des individus. Elle pourra également servir à sécuriser les systèmes en implémentant un Multi-Factor Authentication s (MFAs) (Multi-Factor Authentication) afin d'accroitre la sécurité des systèmes.

4.2 Résultats obtenus

Nous rappelons que la génération de nos modèles, à partir des données d'apprentissage a été possible grâce au service cloud de Google Colaboratory . Celui-ci est un outil intuitif donnant accès à des ressources informatiques permettant de travailler sur des projets en science des données. Par la suite ces modèles ont été sauvegardé sur Cloud Storage afin d'alimenter notre application Django qui offrira les APIs pour les opérations d'identification et d'authentification. L'utilisation de l'IA pour la reconnaissance de locuteur a connu une avancée significative au cours des dernières années grâce à l'amélioration des algorithmes d'apprentissage automatique et de traitement du signal vocal. Les

résultats obtenus sont encourageants et permettent d'identifier un locuteur avec une précision élevée. Les tests réalisés sur des enregistrements de voix ont montré que l'IA est capable de reconnaître un locuteur avec une précision pouvant varier en fonction de la qualité de l'enregistrement, du bruit ambiant et de la durée de la parole.

4.2.1 liste des locuteurs

La figure suivante présente la liste des locuteurs. Dans notre architecture, nous les avons appelé POI (Person Of Interest).

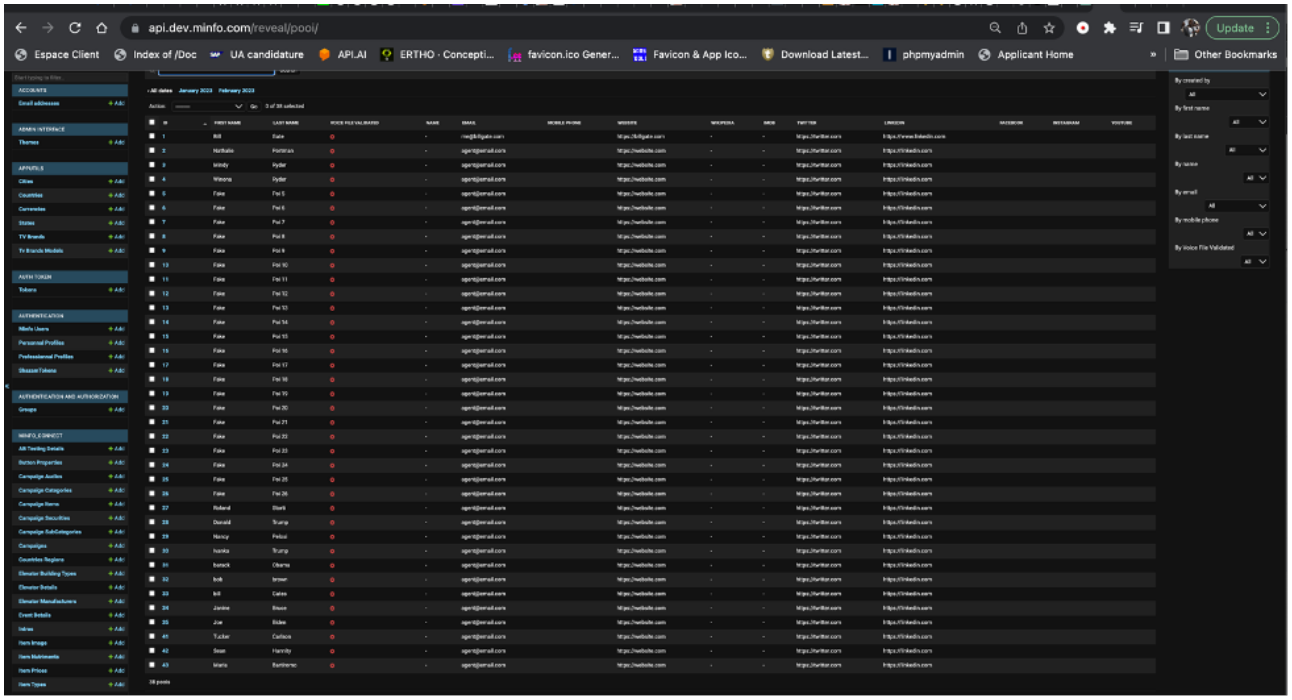


FIGURE 4.1 : Liste des POI

A partir de cette page nous pouvons facilement étendre la liste, ajouter des empreintes vocales pour les différents POIs.

4.2.2 mise à jour du model

Après l'entrainement sur Google Colab, le model est exporté vers Cloud Storage pour usage ultérieure.

Afin de récupérer ce model et l'utiliser en production dans notre projet Django, nous appelons la route **api/speaker/updatemodel**

4.2.3 Identification de locuteur

L'identification répond à la question : « Qui est cette personne ? ». Elle consiste à vérifier dans notre base de données à qui correspond une empreinte vocale.

Route : /api/speaker/id
Requête : POST
Paramètre : voice(le fichier vocal)
Le locuteur doit être dans la liste des POI.

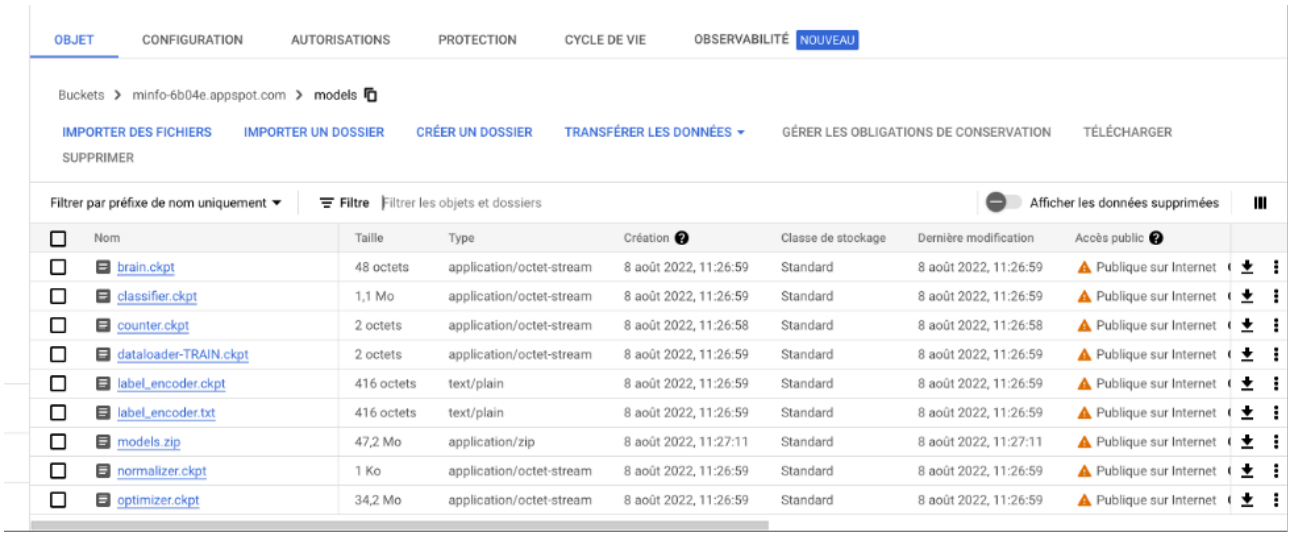


FIGURE 4.2 : Meilleure modèle sauvegardé sur Cloud Storage

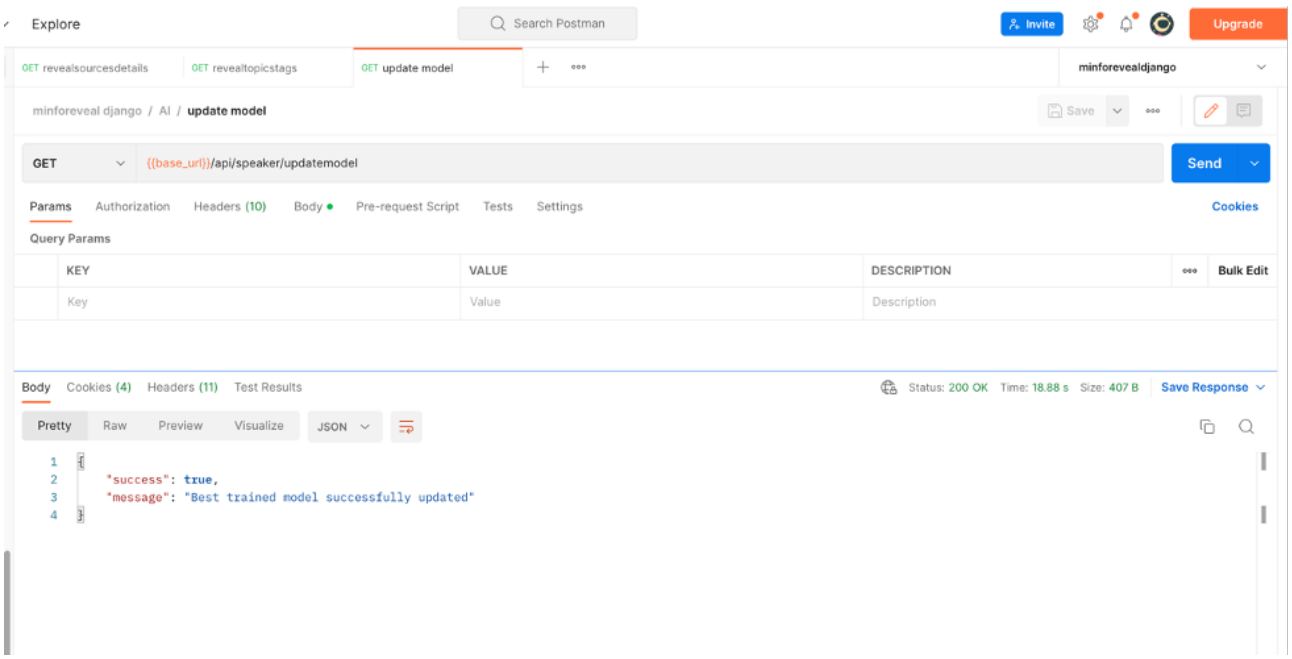


FIGURE 4.3 : récupération du meilleur modèle

4.2.4 authentification de locuteur

L'authentification permet de répondre à la question : « Le locuteur est-il vraiment celui qu'il prétend être ? ». Pour cet API on n'a pas besoin d'avoir les empreintes vocales du locuteur. Ainsi la plateforme qui intègre l'API peut conserver ses empreintes sur ses serveurs privés cela résout en même temps les problématiques de privilèges d'accès aux données, de sécurités et de confidentialité.

route /api/speaker/match

Requête : POST

Paramètre : voice1 et voice2(le fichier vocal empreinte et le fichier vocale enregistré à l'authentification)

Le locuteur n'a pas besoin d'être enregistré dans la base des POI.

La prédiction représente le résultat de l'authentification :

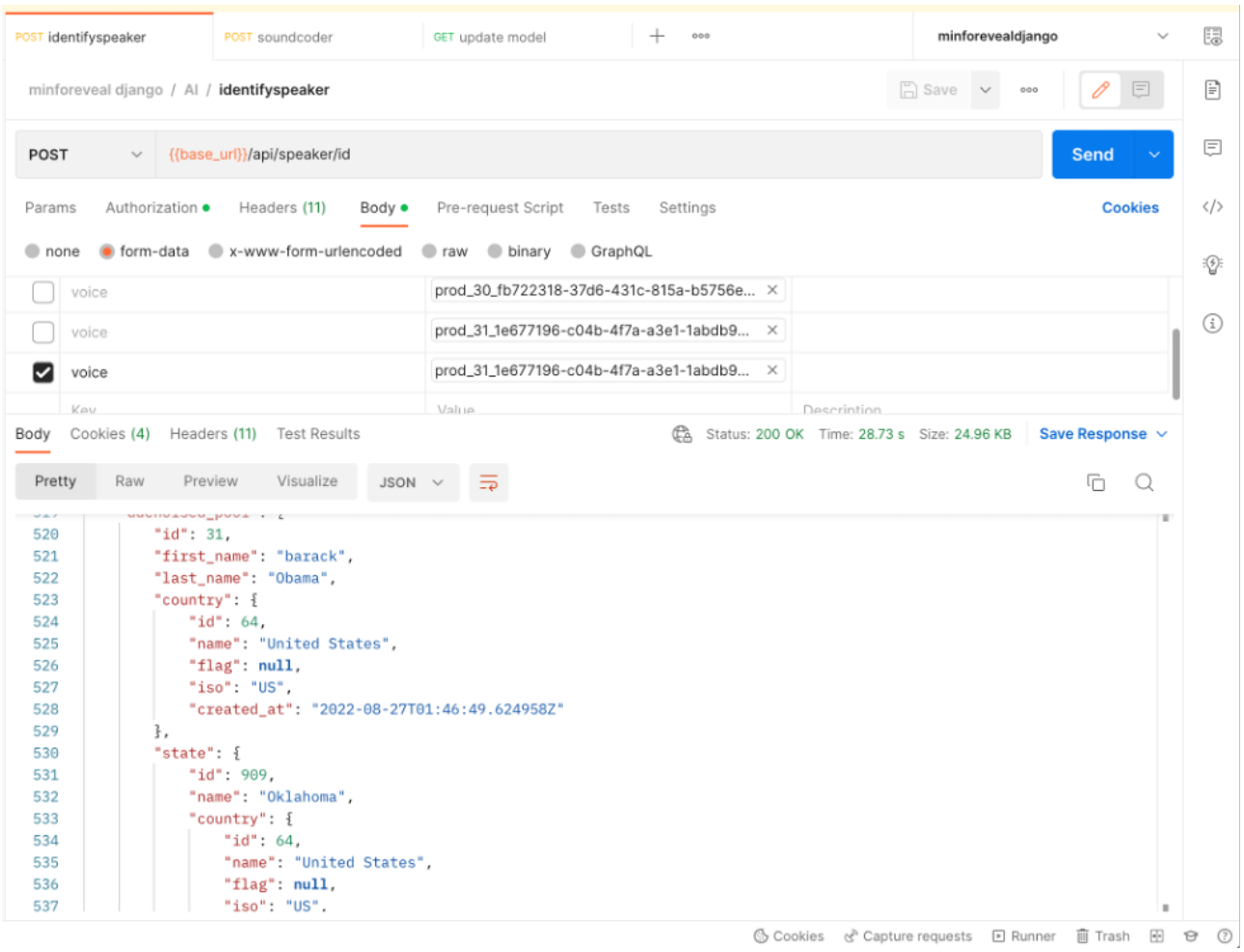


FIGURE 4.4 : Identification de locuteur

- True signifie que l'utilisateur est bien celui qu'il prétend être
- False signifie que l'utilisateur n'est pas le bon, c'est-à-dire que la voix ne correspond pas à l'empreinte vocale.

4.2.5 Analytic des connexions

Nous avons mise en place un service d'analytique pour le suivi et l'utilisation du système. Il enregistre la date, le fichier vocal et le résultat du scan.

4.3 avantage de notre architecture

Le TDNN (Time Delay Neural Network) est l'architecture de réseau de neurones artificiels qui a été utilisée dans. Le cas de notre étude pour la reconnaissance de locuteur. Concernant ses avantages, nous pouvons citer :

- Le TDNN est relativement rapide à entraîner et à utiliser une fois qu'il a été entraîné.
- Les TDNN peuvent prendre en compte les caractéristiques temporelles des signaux acoustiques. En effet, les TDNN sont capables de prendre en compte les données d'entrée passées et présentes, ce qui est important pour la reconnaissance de locuteur.

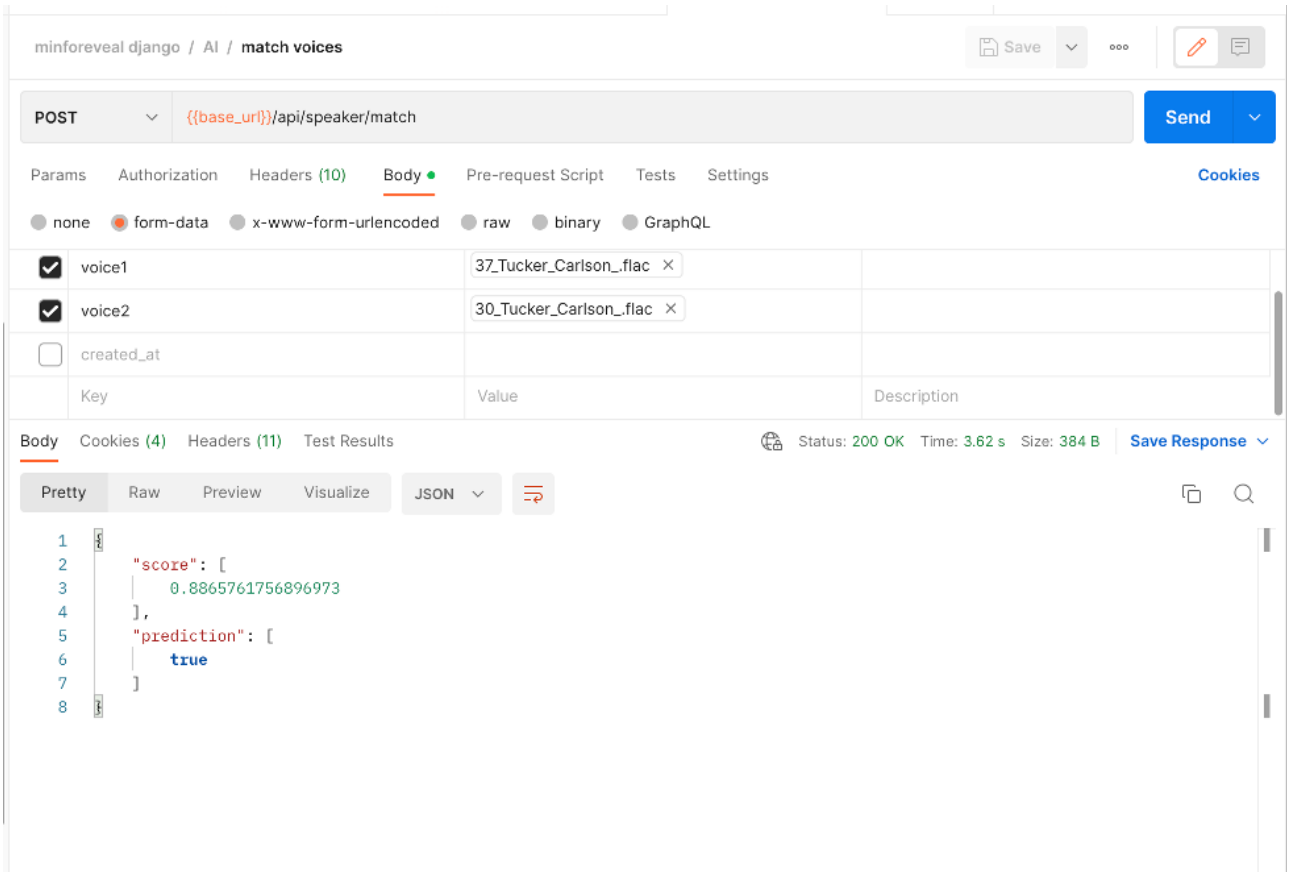


FIGURE 4.5 : Identification de locuteur

- Les TDNN peuvent être utilisés pour extraire des caractéristiques discriminantes des signaux acoustiques. Les TDNN peuvent apprendre des caractéristiques complexes des signaux acoustiques, ce qui est important pour la reconnaissance de locuteur.
- Les TDNN peuvent être entraînés sur de grandes quantités de données. Les TDNN sont capables d'apprendre à partir de grandes quantités de données, ce qui est important pour la reconnaissance de locuteur.

4.4 Les limites de notre architecture

Il convient de noter que les inconvénients du TDNN pour la reconnaissance de locuteurs peuvent varier en fonction du contexte d'utilisation et des caractéristiques des données de parole utilisées pour l'entraînement et le test. Les TDNN peuvent être sensibles aux variations de conditions d'enregistrement. Les TDNN peuvent être affectés par des facteurs tels que le bruit de fond, la qualité de l'enregistrement, la distance entre le locuteur et le microphone, etc. Les TDNN peuvent nécessiter une grande quantité de données d'entraînement. Pour obtenir des performances de reconnaissance de locuteur élevées, il peut être nécessaire de disposer d'une grande quantité de données d'entraînement. Les TDNN peuvent nécessiter une puissance de calcul importante. L'entraînement et l'utilisation de TDNN peut être coûteux en termes de puissance de calcul, surtout lorsqu'il s'agit de grandes quantités de données.

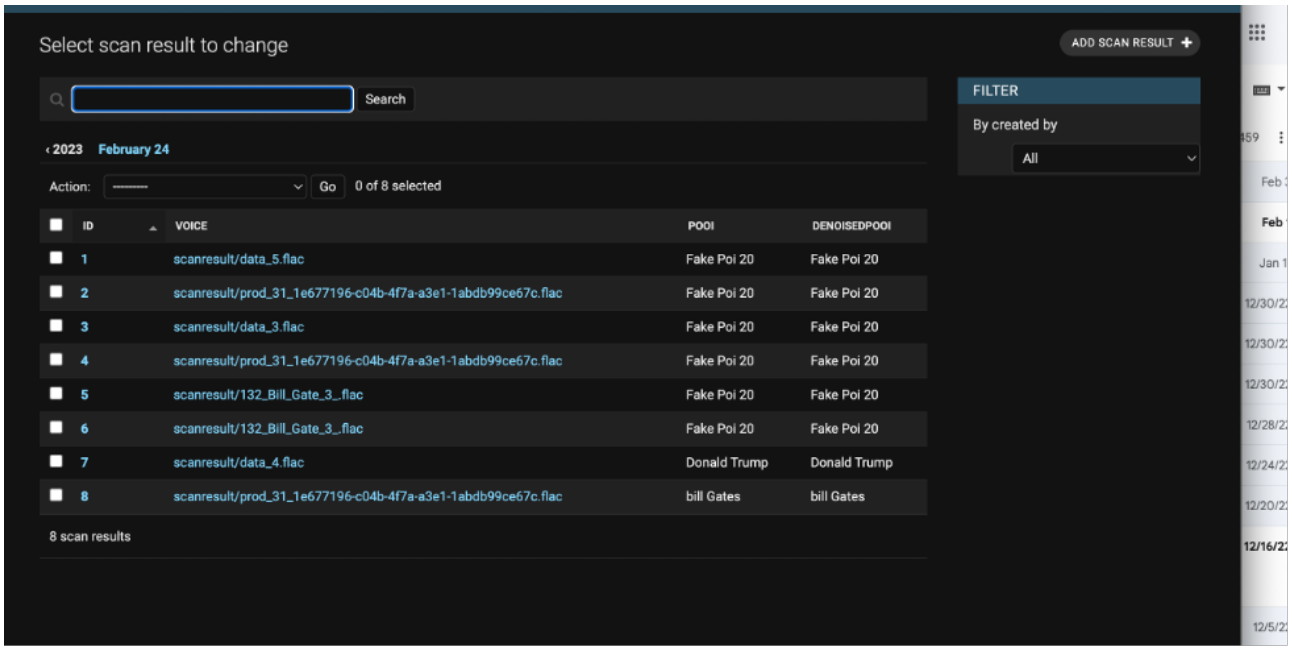


FIGURE 4.6 : Identification de locuteur

Conclusion

Dans ce chapitre, nous avons présenté les résultats obtenus à travers quelques figures présentant l'utilisation de nos différent APIs. Dans le cas où l'IA est bien entraînée avec des données de haute qualité et utilisée dans un contexte approprié, elle peut fournir une reconnaissance de locuteur précise et fiable. Cela peut être utile dans diverses applications, telles que la sécurité de l'authentification vocale, l'analyse de la parole pour la recherche et la reconnaissance de locuteur dans les applications de communication. Cependant, l'utilisation d'une IA pour la reconnaissance de locuteur peut également soulever des préoccupations en matière de vie privée et de sécurité. Si les données vocales utilisées pour entraîner l'IA sont mal protégées, elles peuvent être utilisées pour identifier les utilisateurs sans leur consentement, ou même être utilisées à des fins malveillantes. Il est donc important de prendre des mesures pour protéger la vie privée des utilisateurs et de veiller à ce que l'utilisation de l'IA pour la reconnaissance de locuteur soit justifiée et éthique. C'est pour ça que notre implémentation de l'authentification renécessite pas une sauvegarde par notre système des empreinte vocaux. En résumé, l'utilisation d'une IA pour la reconnaissance de locuteur peut être utile dans certains contextes, mais elle nécessite une attention particulière pour garantir la précision, la fiabilité et la protection de la vie privée des utilisateurs.

Conclusion Générale et perspectives

Ce mémoire de master offre une vue d'ensemble approfondie des perspectives et des défis de la mise en place de systèmes d'IA pour la reconnaissance de locuteur. La reconnaissance de locuteur à l'aide de l'intelligence artificielle est un domaine en constante évolution qui offre d'importantes opportunités pour diverses industries. Les avancées dans les domaines de l'apprentissage automatique, du traitement du signal et de la reconnaissance de la parole ont conduit au développement de systèmes de plus en plus précis et efficaces. Dans ce contexte, ce mémoire explore en détail les différentes applications de la reconnaissance de locuteur alimentée par l'IA. Parmi celles-ci, nous examinons notamment son rôle crucial dans les domaines de la sécurité, des centres d'appels, de l'assistance aux personnes âgées ou atteintes de troubles cognitifs, ainsi que dans les services financiers et les systèmes de vérification d'identité. Ces applications offrent des avantages significatifs en termes d'efficacité, de sécurité, de commodité et de qualité de vie, mais elles soulèvent également des questions éthiques et de confidentialité concernant le traitement des données des utilisateurs et le respect de la vie privée.

Notre étude met également en lumière l'évolution future de ces systèmes, soulignant leur potentiel d'intégration dans une gamme encore plus large de dispositifs et de technologies à l'avenir. Cette expansion pourrait permettre une interaction plus intuitive et sans friction entre les humains et les machines, avec des applications potentielles dans des domaines tels que les véhicules autonomes, les dispositifs médicaux, les appareils domestiques connectés et les systèmes de sécurité résidentiels. Cependant, cette intégration accrue pose des défis supplémentaires en matière de développement, de sécurité, de protection de la vie privée et d'acceptation sociale. Il sera essentiel de trouver un équilibre entre les avantages offerts par ces technologies et le respect des droits et des préoccupations des utilisateurs.

Cependant, la mise en place de ces systèmes soulève également des questions éthiques et de confidentialité. Il est important de garantir que les données des utilisateurs soient traitées de manière éthique et sécurisée, conformément aux lois et réglementations applicables.

En perspectives, nous comptons ajouter d'autres fonctionnalités, poursuivre nos travaux de recherche afin de perfectionner au mieux le model pour aboutir à une solution beaucoup plus optimale. Il s'agira aussi de le mettre en place avec d'autres méthodes et algorithme de Deep Learning pour comparer les performances.

Bibliographie

[1] Vivoka. La biométrie vocale est-elle un processus vraiment fiable ?, . URL https://vivoka.com/fr/la-biometrie-vocale-est-elle-un-processus-vraiment-fiable/.

[2] Idemia. Qu'est-ce que la biométrie ? (consulté le : 09.01.2023). URL https://www.idemia.com/fr/biometrie.

[3] Nice. Biométrie vocale embarquée, authentification et identification (consulté le : 09.01.2023). URL https://www.nice.com/glossary/what-is-contact-center-voice-authentication.

[4] Vivoka. Biométrie vocale embarquée, authentification et identification (consulté le : 09.01.2023), . URL https://vivoka.com/fr/biometrie-vocale-embarquee-authentification-identification/.

[5] Shakeel Sheikh Md Sahidullah Fabrice Hirsch Slim Ouni. Embeddings. 2022.

[6] Lawrence G. Roberts et Thomas G. Stockham Jr. The lincoln tx-0 voice input system. 1962.

[7] Time Doctor. What is voice authentication ? (pros, cons, faqs) ; (consulté le : 09.01.2023). URL https://www.timedoctor.com/blog/voice-authentication/?utm_term=&utm_campaign=General+Performance+Max&utm_source=google&utm_medium=cpc&hsa_acc=9406395779&hsa_cam=19250146465&hsa_grp=&hsa_ad=&hsa_src=x&hsa_tgt=&hsa_kw=&hsa_mt=&hsa_net=adwords&hsa_ver=3&gclid=CjwKCAiA85efBhBbEiwAD7oLQIQ4852C_NFGMEfZLHkS86XXgudzFHsIw7C_B16CRFwJpMw9gF1T8hoChfoQAvD_BwE#use-cases.

[8] imageware. Biometric voice recognition – everything you should know (consulté le : 09.01.2023). URL https://imageware.io/biometric-voice-recognition/.

[9] RCDev. Méthode d'authentification biométrique vocale améliorée (consulté le : 09.01.2023). URL https://www.rcdevs.com/fr/improved-voice-biometrics-authentication-method.

[10] thalesgroup. La biométrie au service de l'identification et l'authentification (consulté le : 09.01.2023). URL https://www.thalesgroup.com/fr/europe/france/dis/gouvernement/inspiration/biometrie.

[11] Aware. Biométrie vocale embarquée, authentification et identification (consulté le : 09.01.2023). URL https://www.aware.com/voice-authentication.

[12] Nouveau paradigme de programmation. Machine learning. Structures d'apprentissage & réseaux neuronaux reconnaissance de l'empreinte vocale - (consulté le : 22.01.2023). URL https://www.oreilly.com/library/view/deep-learning-with/9781788624336/a7a045c6-b0e2-437c-892d-1e61c11446bf.xhtml.

[13] FRANÇOIS CHOLLET. *Deep Learning with Python*. -, 2020.

[14] Kaggle. Les réseau de neurone - (consulté le : 24.01.2023). URL https://www.kaggle.com/discussions/getting-started/388478.

[15] Mathworks. Speaker recognition using x-vectors (consulté le : 09.01.2023). URL https://www.mathworks.com/help/audio/ug/speaker-recognition-using-x-vectors.html.

[16] Softjourn. Is voice authentication secure enough to be your new password ? (consulté le : 09.01.2023). URL https://softjourn.com/insights/security-considerations-in-voice-authentication.

[17] Frontiers. X-vectors : New quantitative biomarkers for early parkinson's disease detection from speech (consulté le : 09.01.2023). URL https://www.frontiersin.org/articles/10.3389/fninf.2021.578369/full#:~:text=X%2Dvectors%20are%20fixed%2Dlength,seen%20during%20the%20DNN%20training.

[18] The Marvin Project. Structures d'apprentissage & réseaux neuronaux reconnaissance de l'empreinte vocale - (consulté le : 09.01.2023). URL http://themarvinproject.free.fr/final/node3.html.

Table des matières

Dédicace **ii**

Remerciements **iii**

Résumé **iv**

. iv

. iv

Abstract **v**

. v

. v

Table des figures **vi**

Sigles et abréviations **viii**

Glossaire **x**

Introduction **1**

. 1

. 1

. 2

. 2

1 État de l'art de l'identification par biométrie vocale **3**

Introduction 3

1.1 Définitions 3

1.1.1 Biométrie 3

1.1.2 Biométrie vocale 4

1.1.3 Identification et authentification biométrique 4

. 4

. 5

1.1.4 Deep learning 5

1.2 Historique de la biométrie vocale 5

. 5

. 5

. 5

. 5

. 5

. 5
1.3 Types de reconnaissances vocales 6
1.4 Fonctionnement 6
1.5 Cas d'utilisation de la biométrie vocale 7
. 7
1.6 Les avantages de la biométrie vocale 7
1.7 Les inconvénients de la biométrie vocale 7
1.8 Apprentissage profond 10
1.8.1 Notion d'intelligence artificielle 10
. 10
. 10
. 10
1.8.2 Notion d'apprentissage automatique 10
. 10
. 10
. 10
. 10
. 11
. 11
. 11
. 12
1.8.3 Notions de Deep learning 12
1.8.3.1 définition 12
. 12
. 13
. 13
. 13
. 13
. 13
1.8.3.2 Fonctionnement 13
. 13
. 14
. 14
. 14
. 14
. 15
1.9 Le Deep Learning pour la reconnaissance vocale 16
. 16
. 18
. 18
1.9.1 que représentent les x-vector dans la reconnaissance vocale ? 18
. 18
. 18
. 18
1.10 Les bonnes pratiques en matière de reconnaissance vocale 19

. 19
1.11 À quel point la reconnaissance vocale est-elle sécurisée ? 19
. 19
. 19
. 19
. 20
. 20
Conclusion 21
. 21

2 Modélisation et développement de notre modèle d'intelligence Artificielle 22
Introduction 22
2.1 Le model 22
. 22
. 22
. 22
. 23
. 23
. 23
2.2 Les données d'entrainement 24
. 24
2.3 Le code 24
2.4 Matériels 25
2.5 Description des données 25
2.6 Métriques d'évaluation 25
. 25
2.7 Choix technologique 26
2.7.1 Définitions 26
. 26
. 26
. 27
. 27
. 27
2.7.2 Structure 28
2.7.3 Fonctionnement 29
. 29
. 29
Conclusion 29
. 29
. 29

3 Matériels et Méthodes 30
Introduction 30
3.1 architecture générale du système 30
. 30

. 30
. 30
. 30
. 31
. 31
3.2 Installations 31
3.3 Préparation des données 31
3.4 Entrainement du modèle TDNN 32
3.5 Inférence 39
3.6 Le développement des Apis 40
Conclusion 46

4 Résultats et Discussions **49**
Introduction 49
4.1 Présentation du cadre d'étude 49
4.2 Résultats obtenus 49
4.2.1 liste des locuteurs 50
4.2.2 mise à jour du model 50
4.2.3 Identification de locuteur 50
4.2.4 authentification de locuteur 51
. 51
4.2.5 Analytic des connexions 52
4.3 avantage de notre architecture 52
4.4 Les limites de notre architecture 53
Conclusion 53

Conclusion **55**
. 55
. 55
. 55
. 55

56

Bibliographie **56**

Table des matières **58**