SPEAKER RECOGNITION SYSTEM WITH PYTHON AND MACHINE LEARNING

BY

NAME: AMINU UKASHA ALIYU

(18134023)

BEING A PROJECT SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE, FACULTY OF SCIENCE, SOKOTO STATE UNIVERSITY, IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE AWARD OF BACHELOR OF SCIENCE(HONS) IN COMPUTER SCIENCE

[NOVEMBER, 2023]

DECLARATION

I hereby declare that this project is the result of my investigations, except where otherwise stated. I also declare that it has not been previously or concurrently submitted as a whole for any other degree at the Sokoto State University or other institutions.

Name: ……………………………………………………

Sign: ……………………………………………………

Date: ……………………………………………………

CERTIFICATION

This is to certify that this project has been read, approved and accepted as meeting the requirement for the award of a Bachelor of Science (B.Sc.) degree in Computer Science in the Department of Computer Science, Faculty of Science, Sokoto State University.

Mal Zahariyah L. Hassan (Project Supervisor)	Date
Mal Zahariyah L. Hassan (Head of Department)	Date
Prof. Donfack R, A Florentin (External Examiner)	Date

DEDICATION

To my Mom, my Dad and my family, their supports and prayers are paramount.

ACKNOWLEDGMENT

All gratitude is to Allah (S.W.A) for allowing me to see through this. I would like to express my special thanks to my supervisor Mal. Zahariyya Lawal Hassan for her tender and yet intellectual guidance throughout this project, my course mates Abubakar Hassan, Jawad A Bello, Ahmed Olaiya, Basheer Yakubu, Aisha Ibrahim, Halilu Ibrahim and Aliyu Ismail and others whose advices and impacts paved way to the success of this project, my fiancee Saudah and my friends Bashar Muhammad and others for their prayers and motivations.

TABLE OF CONTENT

ABSTRACT

Speaker recognition technology has witnessed significant advancements, yet the challenge of recognizing speakers across diverse languages and acoustic conditions persists. This project presents the development and implementation of an open-set, language-independent speaker recognition system, transcending linguistic barriers. Leveraging Convolutional Neural Networks (CNNs) and innovative data management strategies, the system achieves remarkable accuracy, consistently identifying known speakers with approximately 99% precision. Real-time processing capabilities enhance its versatility, while rigorous unit, system, and usability testing confirm its reliability and user-friendliness. This project signifies a successful stride toward bridging language gaps in speaker recognition. Recommendations for further improvement encompass refining unknown speaker classification, enhancing language adaptability through Natural Language Processing (NLP), and accommodating diverse acoustic conditions. A commitment to user feedback integration, scalability, and robust privacy and ethical considerations ensures the system's continued evolution and ethical compliance. This open-set language-independent speaker recognition system represents a significant contribution to the field of voice recognition technology, poised to offer robust and versatile solutions across global linguistic landscapes.

CHAPTER ONE: INTRODUCTION

1.1 BACKGROUND OF THE STUDY

Research on speaker recognition first started in the 1930s. In March of 1932, the kidnapping and killing of Charles and Anne Lindbergh's baby boy led to a research into speakers' speech signals. During the suspected kidnapper's trial, Charles Lindbergh claimed that the voice of the kidnapper, Bruno Hauptmann, was the same as the voice he heard while waiting in a car nearby where the ransom was paid (Rafizah *et al*., 2021). Frances McGehee, who was inspired by the case, conducted the first academic research on this.

Speaker recognition, also referred to as voice recognition or voice biometrics, is a fascinating field that focuses on identifying individuals based on their unique vocal characteristics. Each person has distinct speech patterns, articulation, pitch, and other acoustic properties that set their voice apart from others. Leveraging advancements in machine learning and signal processing techniques, speaker recognition systems have become increasingly accurate and reliable (Furui, 2011).

The speech signals convey many levels of information to the listener. At the primary level, speech conveys a message via words. But at secondary levels, speech delivers information about the spoken language, the emotion of the speaker, the gender and the speaker's general identity. While speech is aimed at recognizing the words being spoken, speaker recognition on the other hand deals with the recognition of who has spoken the words (Reynolds *et al*., 1995).

The general area of speaker recognition entails mainly two basic tasks. Speaker verification and speaker identification. Speaker identification is the process of determining who has spoken from a set of known voices. Therefore, the system will classify the voice it hears to be one of the speakers it already knows hence referred to as a closed-set identification. Speaker verification, authentication or detection as may be called involves the task of determining if a person is the person he is claiming to be. In this system, imposters are not known to the system, it only knows the valid users. By adding the "unknown users" option to the closed-set identification task, the two tasks when merged is what is called an open-set identification.

The speech that can be used to accomplish this task can be either text-dependent or text-independent. In text-dependent speaker recognition, the system has prior knowledge of the words that will be spoken to it which is believed to have better performance. Text-independent systems are robust and difficult but also more flexible because they allow verification while a user is conducting other speech conversations which is known as background verification (Reynolds, 2002).

In the past, speaker recognition systems primarily relied on statistical modeling approaches such as Gaussian Mixture Models (GMM) and Hidden Markov Models (HMM). These methods involved extracting handcrafted features from voice samples and modeling the statistical relationships among these features. However, these traditional approaches had limitations in dealing with complex variations in speech patterns and were sensitive to noisy environments (Reynolds *et al*., 1995).

With the advent of deep learning and the availability of large-scale labeled datasets, neural network-based approaches have gained prominence in the field of speaker recognition. Deep learning models, such as Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), have shown remarkable success in extracting discriminative features from raw audio signals, leading to improved accuracy in speaker recognition tasks.

1.2 STATEMENT OF THE PROBLEM

The rapid advancement of speaker recognition technology has led to its integration into a wide range of applications, from security systems to voice assistants, facilitating secure and personalized interactions. However, existing solutions primarily focus on closed-set recognition, limiting their effectiveness in real-world scenarios where encounters with unknown and unenrolled speakers are inevitable. Furthermore, the majority of speaker recognition systems are designed to operate within a specific language, often hindering their usability across diverse linguistic environments.

This project aims to address the pressing need for an open-set and language-independent speaker recognition system that can accurately identify and verify both known and unknown speakers across various languages and dialects. The primary challenge lies in developing a robust framework capable of capturing the distinctive vocal characteristics of individuals while adapting to the intricacies of open-set recognition and accommodating linguistic diversity.

The open-set nature of the system requires the ability to accurately differentiate between known and unknown speakers, effectively managing instances where a speaker's profile is absent from the system's training data. Meanwhile, the language-independent aspect necessitates techniques that transcend linguistic boundaries, ensuring the system's applicability across different languages and phonetic variations.

The proposed solution will encompass innovative approaches in feature extraction, classification algorithms, and language-independent techniques. By leveraging state-of-the-art machine learning and deep learning methodologies, this project seeks to create a comprehensive speaker recognition system that can excel in real-world scenarios characterized by varied acoustic conditions, diverse speaker demographics, and linguistic heterogeneity.

The successful development of this open-set and language-independent speaker recognition system will have significant implications for enhancing security measures, optimizing user experiences in multilingual environments, and advancing the field of speaker recognition technology as a whole. The ability to accurately identify and verify speakers regardless of language and enrollment status will pave the way for more versatile and effective applications in domains such as secure authentication, communication systems, and smart devices.

1.3 AIM AND OBJECTIVES OF THE STUDY

This project is aimed at developing a language-independent and open-set speaker recognition system with machine learning and Python using extracted mfcc features and labels to train the model.

The primary objectives of this project are as follows:

1. Data Collection and Preprocessing**:** Assemble a diverse dataset of 3000 audio samples, each approximately 5 seconds in duration, from three designated individuals. Convert audio WAV files into MFCC representations suitable for CNN input.
2. MFCC Extraction: Develop an MFCC extraction pipeline to transform audio signals into MFCC feature matrices, serving as input data for the CNN model.
3. CNN Architecture Design: Design a CNN architecture optimized for handling MFCC input features. Define the number of convolutional layers, filter sizes, pooling layers, and strides.
4. Model Training and Validation: Train the CNN model using the extracted MFCC features. Implement training procedures with data augmentation, dropout, and batch normalization. Regularly validate performance using a dedicated validation dataset.
5. Open-Set Recognition Integration: Integrate open-set recognition capabilities into the CNN model, ensuring accurate classification of known speakers and effective handling of unknown speakers.

6. Language-Independent Features: Explore language-independent features or techniques, investigating the integration of MFCCs that are robust to linguistic variations.
7. Evaluation and Metrics: Evaluate the CNN model's performance using accuracy as the primary metric. Validate its proficiency in classifying known speakers, managing unknown speakers, and generalizing across languages.
8. Interpretability Analysis: Conduct an interpretability analysis, visualizing learned filters, feature maps, and activation patterns to gain deeper insights into the decision-making processes of the model.
9. Documentation and Reporting: Thoroughly document the entire project, including the CNN architecture, MFCC extraction, training processes, hyperparameters, and evaluations. Compile a comprehensive report outlining the methodology, findings, challenges, and recommendations for future enhancements.

1.4 SIGNIFICANCE OF THE STUDY

The development of an accurate and reliable speaker recognition system has significant implications in various domains. In the field of security, speaker recognition technology can strengthen access control systems by verifying the identity of individuals based on their voices. This can be applied in high-security environments such as government agencies, military facilities, and financial institutions, where robust authentication mechanisms are crucial.

Moreover, integrating speaker recognition into personal devices offers enhanced user experiences. By using voice-based authentication, users can securely access their devices and personalize their settings, applications, and services. This technology can be particularly valuable in smartphones, tablets, smart speakers, and other IoT devices that are becoming ubiquitous in our daily lives.

In forensic investigations, speaker recognition systems can assist law enforcement agencies in identifying individuals from audio evidence. This can provide valuable leads, strengthen criminal investigations, and potentially lead to the resolution of cases. Additionally, speaker recognition technology can aid in voice-based detection.

The applications of speaker recognition technology are quite varied and continually growing. Below is an outline of some broad areas where speaker recognition technology has been or is currently used (Reynolds, 2002).

1. Access Control: originally for physical facilities, more recent applications are for controlling access to computer networks (and biometric factor to usual password and/or token) or websites(thwart password sharing for access to subscription sites). Also used for automated password reset services.
2. Transaction Authentication: for telephone banking, in addition to account access control, higher levels of verification can be used for more sensitive transactions, more recent applications are in user verification for remote electronic and mobile purchases(e_ and m-commerce)
3. Law Enforcement: some applications are home-parole monitoring (call paroles at random times to verify they are at home) and prison call monitoring (validate inmates before outbound calls). There has also been discussion of using automatic systems to corroborate aural/spectral inspections of samples for forensic analysis
4. Speech Data Management: in voice mail browsing or intelligent answering machine, use speaker recognition to label incoming voice mail with the speaker name for browsing and/or action (personal reply). For speech skimming or audio mining applications, annotate recorded meetings or videos with speaker labels for quick indexing and filing.

5. Personalization: in voice-web or device customization, store and retrieve personal settings/preferences based on user verification for multi-user sites or devices (car climate and audio settings). There is also interest in using recognition techniques for directed advertisement or services, where, for example, repeat users could be recognized or advertisement focused based on recognition of broad speaker characteristics (e.g. gender or age).

1.5 SCOPE AND LIMITATIONS OF THE STUDY

1.5.1 Scope

The scope of this project includes the development of a speaker recognition system using Python and machine learning techniques. The system will be designed to recognize the voices of 3 predefined speakers and provide a response of "voice not recognized" for voices that do not belong to any of the known individuals. The project will involve the following key components:

1. Data Collection: Collecting voice samples of the speakers to create a training dataset for the speaker recognition system.
2. Feature Extraction: Implementing feature extraction techniques, such as MFCCs, to extract relevant features from the voice samples.
3. Machine Learning Model: Training a machine learning model, such as a deep neural network, using the extracted features to recognize the voices of the known speakers.
4. System Development: Developing a user-friendly interface to record and process voice samples for speaker recognition.
5. Evaluation: Assessing the performance of the speaker recognition system using a separate test dataset, which includes voices from both the known speakers and other individuals not included in the known set.

6. Open Set Approach: Implementing an open set approach to accurately recognize known voices and appropriately reject unknown voices.

1.5.2 Limitations

While this project aims to develop a functional speaker recognition system, certain limitations should be considered:

1. Limited Known Speaker Set: The system will only be trained to recognize the voices of the 3 predefined speakers. It may encounter difficulties in accurately recognizing the voices of individuals who are not part of the known set.
2. Environmental Factors: The performance of the system may be influenced by environmental factors, such as background noise, microphone quality, and recording conditions. Noisy or poor-quality recordings may impact the accuracy of the system.
3. Voice Variations: Variations in an individual's voice due to accent, speech rate, or emotional state may pose challenges in achieving optimal performance in speaker recognition, especially when the system is exposed to voice samples with significant variations from the training data.
4. Impersonation Attacks: The system may be susceptible to impersonation attacks, where an unauthorized individual intentionally imitates the voice of a known friend. Detecting and preventing such attacks is a challenging task and may require additional technologies.

1.6 DEFINITION OF TERMS

Open Set Recognition:

Open set recognition, often referred to as open set identification or outlier detection, constitutes a critical facet within the realm of speaker recognition systems. In the complex landscapes of real-world scenarios, the fundamental challenge extends beyond merely identifying familiar voices; it encompasses the intricate task of correctly discerning and categorically rejecting unknown or unauthorized speakers (De la Torre and Peinado, 2018). This pivotal undertaking is indispensable for ensuring the efficacy and reliability of speaker recognition systems, particularly in security-sensitive applications where the consequences of misclassification could be dire.

Mel Frequency Cepstral Coefficient (MFCC):

The Mel Frequency Cepstral Coefficient (MFCC) technique, an indispensable cornerstone in the domain of speech recognition, constitutes a feature extraction methodology that bridges the gap between raw audio signals and meaningful feature representations (Li *et al*., 2020). In the pursuit of transforming audio signals into a format amenable to machine learning, the MFCC technique is harnessed to derive a comprehensive and discriminative set of features that capture the inherent nuances and patterns within the audio domain. Its widespread adoption stems from its remarkable ability to mitigate the adverse effects of noise, thus empowering models to achieve superior performance in tasks ranging from speaker identification to speech-to-text translation.

Datasets and Data Processing Libraries:

Datasets, the bedrock upon which the edifice of speaker recognition is erected, play an instrumental role in shaping the capabilities and robustness of recognition models. Among the pantheon of influential datasets, the TIMIT dataset stands tall as a benchmark, housing recordings from diverse American speakers uttering an array of phrases (Li *et al*., 2021). VoxCeleb, an expansive repository sourced from interviews with thousands of celebrities, serves as a testament to the monumental role data diversity plays in fortifying recognition models. LibriSpeech, an embodiment of digitized audiobooks, underscores the versatility of datasets, transcending the boundaries of automatic speech recognition (ASR) to engender advancements in speaker recognition. The VCTK Corpus, an embodiment of accents and demographics, offers a palette rich in linguistic and phonetic diversity, further enriching the training landscape (Li *et al.*, 2021). The Speaker Recognition Evaluation (SRE) datasets, a hallmark of rigor and realism, present challenging scenarios that mimic real-world conditions, thereby invigorating the research milieu.

Guiding the preprocessing and manipulation of audio data, data processing libraries are invaluable allies in the speaker recognition journey. Librosa, a multifaceted Python library, weaves an intricate tapestry of functionalities, encompassing audio file loading, resampling, spectrogram extraction, and Mel Frequency Cepstral Coefficient (MFCC) computation (Li *et al.*, 2021). pyAudioAnalysis extends an expansive armory of audio analysis tools, including feature extraction, filtering, and sample rate conversion, ushering users into a realm where audio signal intricacies are unveiled. Soundfile emerges as a conduit facilitating the reading and writing of audio files, offering a gateway to an array of file
formats while championing the cause of audio fidelity (Li *et al.*, 2021). In the domain of audio analysis band processing, Essentia stands as an exemplar, a potent amalgamation of utility and efficiency, complete with functions spanning feature extraction, resampling, and audio file interoperation (Li *et al.*, 2021). The confluence of LibROSA and Kaldi forms an alliance that is more than the sum of its parts, leveraging the feature extraction prowess of LibROSA while harnessing the robust speech processing capabilities of Kaldi to surmount challenges like speech segmentation and acoustic modeling (Li *et al.*, 2021).

Algorithms:

At the heart of the speaker recognition expedition, algorithms unfurl a captivating narrative that traverses the landscapes of statistical modeling and machine learning. Gaussian Mixture Models (GMMs), venerable and versatile, orchestrate an ensemble of Gaussian distributions to encapsulate speaker-specific acoustic features, coalescing into a distinctive framework that wields Universal Background Model (UBM) and Maximum Likelihood Linear Regression (MLLR) to amplify recognition prowess (Wu *et al.*, 2020). Support Vector Machines (SVM), celebrated for their mastery in classification, pivot with aplomb to the realm of speaker recognition, adroitly delineating hyperplanes in feature spaces, unraveling the enigma of identity through the lens of acoustic signatures (Wu *et al*., 2020). Hidden Markov Models (HMMs), exponents of temporal dynamics, kindle the flame of temporal context, epitomizing the transitions between acoustic states and breathing life into the decoding of speaker sequences (Wu *et al*., 2020). i-vectors, heralding a factor analysis renaissance, extract latent dimensions from a tapestry of Gaussian Mixture Models (GMMs), distilling the essence of speakers into a compact representation (Wu *et al*., 2020). The vanguard of deep learning, Deep Neural Networks (DNNs), assumes the mantle of a torchbearer, harnessing the transformative power of convolution and recursion to mine intricate hierarchies and unearth salient features (Wu *et al*., 2020).

Amidst the pantheon of algorithms, Deep Speaker Embeddings ascend as a beacon, pioneering the art of capturing speaker identities in a multidimensional realm, the precipice of which is perched atop neural architectures such as siamese networks, triplet networks, and the resounding echo of x-vector systems (Wu *et al*., 2020). This avant-garde pursuit, intertwined with the fabric of deep learning, unshackles models from the confines of handcrafted features, ushering in a new epoch where raw audio data unfurls its intricate tapestry of phonetic secrets.

Thus, while the selection of an algorithm is akin to an artistic choice, influenced by considerations of task, dataset intricacies, and implementation nuances, the profound impact of deep-learning algorithms cannot be understated. It is this veritable promise that beckons forth the selection of Convolutional Neural Networks (CNNs) as the guiding beacon in the present endeavor, illuminating the path toward unraveling the essence of speaker recognition (Wu *et al*., 2020).

1.7 PROJECT ORGANIZATION

This project will follow the following methodology:

1. Collect a dataset of voice samples from the 3 unknown speakers for training and evaluation.
2. Preprocess the voice data to remove noise and normalize the audio signals.
3. Extract relevant features from the voice samples using techniques such as Mel Frequency Cepstral Coefficients (MFCC)
4. Split the dataset into training and testing sets for model development and evaluation.
5. Train a machine learning model (CNN) on the training data.
6. Validate the trained model's performance using appropriate metrics and evaluate it on unseen voice samples.
7. Analyze the results, discuss the strengths and limitations of the developed system, and suggest possible areas for future research.

CHAPTER TWO: LITERATURE REVIEW

2.1 INTRODUCTION

This chapter provides a comprehensive review of the literature related to speaker recognition systems, open-set recognition approaches, and the application of machine learning techniques in voice recognition. The review aims to establish a solid foundation of existing research, methodologies, and advancements in the field. The chapter begins with an overview of speaker recognition, followed by an exploration of open-set recognition techniques. It then delves into the utilization of various machine-learning algorithms and techniques in speaker recognition systems.

2.2 REVIEW OF RELATED LITERATURE

Speaker recognition refers to the process used to recognize a speaker from a spoken phrase (Dutta and Haubold, 2005). It is a useful biometric tool with wide applications e.g. in audio or video document retrieval. Speaker recognition is dominated by two procedures namely segmentation and classification. Research and development have been ongoing to design new algorithms or to improve on old ones that are used for segmentation and classification.

Early speaker recognition systems relied on statistical modeling approaches, such as Gaussian Mixture Models (GMMs) and Hidden Markov Models (HMMs). These methods involved extracting handcrafted features from voice samples, such as Mel Frequency Cepstral Coefficients (MFCCs) and Linear Predictive Coding (LPC), and modeling the statistical relationships among these features. However, these approaches had limitations in dealing with complex variations in speech and were often sensitive to noise and other environmental factors.

According to (Reynolds *et al.*, 1995), GMM-based systems have been widely used in speaker recognition due to their ability to model the probability distribution of speech features. The

GMMs capture the statistical characteristics of the voice data and can be used to compare the likelihood of an input voice sample belonging to a particular speaker. Despite their success, GMM-based systems struggle to handle speaker variations and the increasing complexity of voice data.

With the emergence of deep learning, there has been a significant shift towards using neural network-based models for speaker recognition tasks. Deep learning models, such as Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), have shown remarkable performance in extracting high-level representations from raw audio signals. These models can automatically learn discriminative features directly from the audio data, leading to improved accuracy and robustness in speaker recognition.

The work of (Li *et al.*, 2017) provides an in-depth overview of deep speaker recognition. They explore various architectures and techniques employed in deep learning-based speaker recognition systems. The paper discusses feature extraction methods, including MFCCs and bottleneck features, as well as neural network architectures such as CNNs and RNNs. It also highlights the importance of large-scale labeled datasets and presents evaluation metrics commonly used in speaker recognition tasks.

(Liu *et al.*, 2017) introduce deep neural network approaches to speaker and language recognition. The paper presents different deep learning architectures used for speaker recognition, including DNNs, CNNs, and RNNs. It discusses various techniques for training deep neural networks, such as unsupervised pretraining, transfer learning, and data augmentation. The authors also explore the fusion of multiple modalities, such as audio and visual information, to improve speaker recognition performance.

To address the challenge of limited labeled data, (Snyder *et al.*, 2018) propose an approach called "unsupervised speaker adaptation." Their work focuses on fine-tuning a model pre-trained on a large amount of unlabeled data using a small amount of labeled data. This transfer learning approach demonstrated improved performance in scenarios with limited labeled training samples. The paper also discusses the importance of adapting the model to specific domains or speakers to enhance recognition accuracy.

(Chen *et al*., 2020) contribute to speaker recognition with their work on contrastive predictive coding. They propose a self-supervised learning method where a model is trained to predict future audio frames from past frames. By leveraging large amounts of unlabeled data, the model learns robust representations that capture speaker-specific information. Experimental results showed the effectiveness of this self-supervised approach in speaker verification tasks.

In addition to the advancements in deep learning models, research has been conducted on open-set speaker recognition systems. (Brummer *et al*., 2006) propose a likelihood ratio-based open-set identification system. The system is designed to handle scenarios where the test sample does not belong to any of the enrolled speakers. It employs likelihood ratio scoring to determine if the test sample falls into an open-set category. The paper presents techniques for modeling the open-set class, handling calibration, and setting decision thresholds.

(Scheirer *et al*., 2013) contribute to open-set recognition by introducing the concept of open-set recognition as a generalized problem. They discuss the challenges associated with open-set recognition and propose evaluation methodologies for measuring the performance of open-set recognition systems. The paper presents different approaches, such as score-based methods and model-based methods, to address open-set recognition in various domains, including speaker recognition.

The above works collectively contribute to the development of speaker recognition systems using Python and machine learning techniques. By understanding the principles and advancements presented in these papers, researchers can make informed decisions in designing and implementing speaker recognition systems that are accurate, robust, and capable of handling open-set scenarios.

2.3 REVIEW OF RELATED SYSTEM

Statistical concepts dominate the field of speaker recognition and they are used for developing models. Machines that are used for speaker recognition purposes are referred to as automatic speech recognition (ASR) machines. ASR machines are either used to identify a person or to authenticate the person's claimed identity (StudyCorgi, 2022). The following is a discussion of various improvements that have been suggested in the field of speaker recognition. Two processes that are important in doing speaker recognition are audio classification and segmentation.

These two processes are carried out using computer algorithms. In developing an ideal procedure for the process of audio classification, it is important to consider the effect of background noise. Because of this factor, an auditory model has been put forward by Chu and Champagne that exhibits excellent performance even in a noisy background.

To achieve such robustness in a noisy background the model inherently has a self-normalization mechanism. The simpler form of the auditory model is expressed as a three-stage processing progression through which an audio signal goes through an alteration to turn into an auditory spectrum, which is modeled inside a neural illustration. Shortcomings associated with the use of this model are that it involves nonlinear processing and high computational requirements.

These shortcomings necessitate the need for a simpler version of the model. A proposal put forward by (Hosseinzadeh *et al.*, 2006) suggests modifications to the model that create a simpler version of it that is linear except in getting the square-root value of energy. The modification is done on four of the original processing steps namely pre-emphasis, nonlinear compression, half-wave rectification, and temporal integration. To reduce its computational complexity the Parseval theorem is applied which enables the simplified model to be implemented in the frequency domain.

The resultant effect of these modifications is a self-normalized FFT-based model that has been applied and tested in speech/music/noise classification. The test is done with the use of a support vector machine (SVM) as the classifier. The result of this test indicates that a comparison of the original and proposed auditory spectrum to a conventional FFT-based spectrum suggests a more robust performance in noisy environments (Hosseinzadeh *et al*., 2006). Additionally, the results suggest that by reducing the computational complexity, the performance of the conventional FFT-based spectrum is almost the same as that of the original auditory spectrum (Hosseinzadeh *et al*., 2006).

One of the important processes in speaker recognition and radio recordings is speech/music discrimination. The discrimination is done using speech/music discriminators. The discriminator proposed by (Giannakopoulos *et al*., 2006). Involves a segmentation algorithm (V-809). Audio signals exhibit changes in the distribution of energy (RMS) and it is on this property that the audio segmentation algorithm is founded on. The discriminator proposed by (Giannakopoulos *et al*., 2006) involves the use of Bayesian networks.

Each of the classifiers is trained on a single and distinct feature, thus, at any given classification nine features are involved in the process. By operating in distinct feature spaces, the independence between the classifiers is increased. This quality is desirable, as the results of the classifiers have to be combined by the Bayesian network in place. The nine commonly targeted features, which are extracted from an audio segment, are Spectral Centroid, Spectral Flux, Spectral Rolloff, Zero Crossing Rate, Frame Energy and 4 Mel-frequency cepstral coefficients.

The new feature selection scheme that is integrated into the discriminator is based on the Bayesian networks. Three Bayesian network architectures are considered and the performance of each is determined. The BNC Bayesian network has been determined experimentally and found to be the best of the three owing to reduced error rate. This proposed discriminator has worked on real internet broadcasts of the British Broadcasting Corporation (BBC) radio stations (Giannakopoulos *et al*., 2006).

An important issue that arises in speaker recognition is the ability to determine the number of speakers involved in an audio session. (Swamy *et al*., 2001) have put forward a mechanism that can determine the number of speakers. In this mechanism, the value is determined from multi-speaker speech signals.

According to (Swamy *et al*., 2001), one pair of microphones that are spatially separated is sufficient to capture the speech signals. A feature of this mechanism is the time delay experienced in the arrival of these speech signals. This delay is because of the spatial separation of the microphones.

The mechanism has its basis in the fact that different speakers will exhibit different time delay lengths. Thus, it is this variation in the length of the time delay, which is exploited to determine the number of speakers. To estimate the length of time delay, a cross-correlation procedure is undertaken. The procedure cross-correlates to the Hilbert envelopes, which correspond to linear prediction residuals of the speech signals.

According to (Zhang and Zhou, 2004), audio segmentation is one of the most important processes in multimedia applications. One of the typical problems in audio segmentation is accuracy. It is also desirable that the segmentation procedure can be done online. Algorithms that have attempted to deal with these two issues have one thing in common. The algorithms are designed to handle the classification of features at small-scale levels.

These algorithms additionally result in high false alarm rates. Results obtained from experiments reveal that the classification of large-scale audio is easier compared to small-scale audio. It is this fact that has necessitated an extensive framework that increases robustness in audio segmentation. The proposed segmentation methodology can be described in two steps. In the first step, the segmentation is described as rough and the classification is large-scale.

This step is taken as a measure of ensuring that there is integrality concerning the content segments. By accomplishing this step you ensure that audio that is consecutive and that is from one source is not partitioned into different pieces thus homogeneity is preserved. In the second step, the segmentation is termed subtle and is undertaken to find segment points. These segment points correspond to boundary regions, which are the output of the first step.

Results obtained from experiments also reveal that it is possible to achieve a desirable balance between the false alarm and the low missing rate. The balance is desirable only when these two rates are kept at low levels (Zhang and Zhou, 2004).

According to (Dutta and Haubold, 2005), the human voice conveys speech and is useful in providing gender, nativity, ethnicity and other demographics about a speaker. Additionally, it also possesses other non-linguistic features that are unique to a given speaker. These facts about the human voice help do audio/video retrieval. To classify speaker characteristics, an evaluation is done on features that are categorized either as low-, mid- or high–level.

MFCCs, LPCs, and six spectral features comprise the low-level features that are signal-based. Mid-level features are statistical and used to model the low-level features. High-level features are semantic and are found on specific phonemes that are selected. This describes the methodology that has been put forward by (Dutta and Haubold, 2005). The data set that is used in assessing the performance of the methodology is made up of about 76.4 hours of annotated audio. In addition, 2786 segments that are unique to speakers are used for classification purposes. The results from the experiment reveal that the methodology put forward by (Dutta and Haubold, 2005) yields accuracy rates as high as 98.6%. However, this accuracy rate is only achievable under certain conditions.

The first condition is that test data is for male or female classification. The second condition to be observed is that in the experiment only mid-level features are used. The third condition is that the support vector machine used should possess a linear kernel. The results also reveal that mid- and high-level features are the most effective in identifying speaker characteristics.

To automate the processes of speech recognition and spoken document retrieval the impact of unsupervised audio classification and segmentation has to be considered thoroughly. (Huang and Hansen, 2004) propose a new algorithm for audio classification to be used in automatic speech recognition.

GMM networks that are weighted form the core feature of this new algorithm. Captured within this algorithm are the Variance of Spectrum Flux and Variance of Zero Crossing-Rate. VSF and VZCR are, additionally, extended-time features that are crucial to the performance of the algorithm. VSF and VZCR perform a pre-classification of the audio and additionally attach weights to the output probabilities of the GMM networks. After these two processes, the White Gaussian Noise networks implement the classification procedure.

For the segmentation process in automatic speech recognition (ASR) procedures, (Huang and Hansen, 2004) propose a compound segmentation algorithm that captures 18 features. The figure below presents the features proposed

Tab 1. Proposed features.

Number required	Feature name
1	2-mean distance metric
1	perceptual minimum variance distortionless response (PMVDR)
1	Smoothed zero-crossing rate (SZCR)
1	False alarm compensation procedure
14	Filterbank log energy coefficients (FBLC)

The 14 FBLCs proposed are implemented in 14 noisy environments where they are used to determine the best overall robust features concerning these conditions. Turns lasting up to 5 seconds can be enhanced for short segments. In such cases 2- mean distance metric can be installed. The false alarm compensation procedure has been determined to boost the efficiency of the rate cost-effectively.

A comparison involving (Huang and Hansen, 2004) proposed classification algorithm against a GMM network baseline algorithm for classification reveals a 50% improvement in performance. Similarly, a comparison involving (Huang and Hansen, 2004) proposed compound segmentation algorithm against a baseline Mel-frequency cepstral coefficients (MFCC) and traditional Bayesian information criterion (BIC) algorithm reveals a 23%-10% improvement in all aspects (Huang and Hansen, 2006).

The data set used for the comparison procedure comprises broadcast news evaluation data obtained from DARPA. DARPA is short for Defense Advanced Research Projects Agency. According to (Huang and Hansen, 2004), these two proposed algorithms achieved satisfactory results in the National Gallery of the Spoken Word (NGSW) corp, which is a more diverse, and challenging test.

The basis of speaker recognition technology in use today is predominated by the process of statistical modeling. The statistical model formed is on the text-dependent and text-independent speaker recognition. The previous section discussed the process of audio classification and segmentation.

This section focuses on discussing the different improvements that have been suggested in the field of audio classification and segmentation. A methodology that is put forward by (Dutta and Haubold, 2005) attempts to do a classification of speaker characteristics using low-, mid- and high-level features. The result of this experiment is an accuracy rate of 98.6%.

Additionally, the results reveal that the most effective features for speaker identification are mid- and high-level features.

In the context of improving audio classification and segmentation for automatic speech recognition (ASR) procedures, (Huang and Hansen, 2004) propose a new algorithm that achieves a 50% improvement in performance compared to the GMM network baseline algorithm. Additionally, (Huang and Hansen, 2004) proposes a compound segmentation algorithm that achieves a 23%-10% improvement in performance compared to traditional methods. The proposed algorithms are tested on the National Gallery of the Spoken Word (NGSW) corp, a diverse and challenging test dataset.

These advancements collectively contribute to the development of more accurate and robust audio classification and segmentation techniques, thereby enhancing the overall performance of speaker recognition systems in various applications.

2.4 REVIEW OF RELATED TECHNOLOGIES AND TOOLS

Speaker Recognition Technology

Speaker recognition technology has evolved significantly over the years. Early systems relied on statistical modeling approaches such as Gaussian Mixture Models (GMMs) and Hidden Markov Models (HMMs) for speaker identification (Reynolds *et al.*, 1995). These methods involved extracting handcrafted features from voice samples, such as Mel Frequency Cepstral Coefficients (MFCCs) and Linear Predictive Coding (LPC), to model the statistical relationships among these features.

Feature Extraction Methods

Feature extraction plays a crucial role in speaker recognition. Mel Frequency Cepstral Coefficients (MFCCs) and Linear Predictive Coding (LPC) are commonly used feature extraction methods (Reynolds *et al.*, 1995). MFCCs capture spectral characteristics, while LPC focuses on modeling the vocal tract system. These methods aim to capture the unique characteristics of speech signals, allowing for effective comparison and identification of speakers.

Segmentation and Classification

Speaker recognition involves two key procedures: segmentation and classification. Segmentation refers to breaking down an audio stream into meaningful segments, while classification assigns these segments to specific speakers. Research has been focused on developing algorithms for accurate segmentation and classification (Li *et al*., 2017). Early systems utilized statistical methods for segmentation, while deep learning models now show promise in both segmentation and classification tasks (Liu *et al*., 2017).

Deep Learning Models

The emergence of deep learning has revolutionized speaker recognition. Deep learning models like Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) have shown exceptional performance in extracting high-level features from raw audio signals (Liu *et al*., 2017). These models can automatically learn discriminative features, improving accuracy and robustness. These models excel in extracting both spectral and temporal features from audio data, making them well-suited for speaker recognition tasks.

Unsupervised Learning

To address challenges related to limited labeled data, unsupervised learning approaches have been proposed. Snyder *et al*. suggest "unsupervised speaker adaptation," where a model pre-trained on unlabeled data is fine-tuned using a small amount of labeled data (Snyder, *et al*., 2018). This approach demonstrates improved performance in scenarios with limited training samples, offering a practical solution for real-world applications.

Self-Supervised Learning

Chen *et al*. introduce contrastive predictive coding, a self-supervised learning method for robust speaker recognition (Chen *et al*., 2020). By predicting future audio frames from past frames using large amounts of unlabeled data, this approach yields representations that capture speaker-specific information. This technique showcases the potential of leveraging unannotated data to enhance speaker verification tasks.

Open-Set Recognition

Advancements have been made in open-set speaker recognition systems. Brummer *et al*. propose a likelihood ratio-based open-set identification system (Brummer *et al*., 2006). This system is designed to handle scenarios where the test sample does not belong to any of the enrolled speakers. Techniques for modeling the open-set class, handling calibration, and setting decision thresholds are presented, providing a comprehensive approach to open-set recognition challenges.

Audio Segmentation:

Audio segmentation is a critical step in various applications, including multimedia processing. Du *et al*. propose an algorithm for unsupervised audio segmentation in practical media (Du *et al*., 2007). By detecting potential acoustic changes and employing Gaussian Mixture Modeling (GMM) and classification algorithms, this approach improves accuracy in segmenting audio streams.

Evaluation and Testing

Evaluation methodologies are essential for assessing the performance of speaker recognition systems. Experimental results and accuracy rates achieved by various proposed methods are presented (Reynolds *et al*., 1995), (Liu *et al*., 2017) and others. Testing on benchmark datasets, such as the National Gallery of the Spoken Word (NGSW) corp, provides insights into algorithm effectiveness and enables fair comparisons.

Tools and Techniques

The reviewed papers employ a range of tools and techniques for feature extraction, classification, and segmentation. These include GMMs, HMMs, CNNs, RNNs, support vector machines (SVMs), Bayesian networks, and various statistical approaches. Each technique contributes to a holistic understanding of the capabilities and limitations of speaker recognition systems.

The field of speaker recognition and audio classification has witnessed remarkable advancements in technology and techniques, ranging from traditional statistical models to cutting-edge deep learning approaches. These innovations, along with the use of diverse tools and evaluation methodologies, contribute to the development of accurate and robust speaker recognition systems capable of handling various challenges and scenarios.

2.5 CHAPTER SUMMARY

One of the main challenges in today's world is security. Fraudsters are everywhere and can be disguised as anyone, thereby making people vulnerable to their frauds. People are willing to embrace new technology that comes to their aid against such characters. The field of speaker recognition if exploited well can form the basis of new technology that can significantly reduce the rate of crime.

Speaker Recognition, In addition to personal identification (PIN) numbers, passwords and other forms of identification we can have voice signatures to enhance security further in restricted areas. When a crime is committed (e.g. a bank robbery) by simply retrieving audio footage of the crime in progress the identity of the assailants can be determined by the aid of a voice database. To elaborate further, the voice database holds audio data of everyone, potential and known criminals, thus you only need to get a match between what is on the database and what has been retrieved (Furui, 2011).

CHAPTER THREE: SYSTEM ANALYSIS AND DESIGN

3.1 INTRODUCTION

This pivotal chapter conducts an in-depth analysis of the open-set language independent speaker recognition system. It meticulously examines the system's current state and its evolution over time, thereby laying the foundation for the subsequent phases of development. Furthermore, it elucidates the system's intricate requirements, explores the methodological approach chosen for its development, and unveils the intricacies of the architectural design that underpins its functionality.

3.2 ANALYSIS OF THE EXISTING SYSTEM

To develop a speaker recognition system that is sentence and language-independent, as well as an open set (able to recognize unknown speakers), understanding the components and procedures of existing systems is paramount. Here is a detailed explanation of the existing systems before mine.

3.2.1 Input of the Existing System

1. **Audio Data:** Existing systems typically take audio recordings as input. These audio recordings can vary in terms of content (different sentences and languages), duration, and quality. Systems might require a diverse dataset that includes a wide range of speech samples from various speakers.
2. **Feature Extraction:** Feature extraction is a crucial step. Existing systems often extract acoustic features from the audio data, such as Mel-frequency cepstral coefficients (MFCCs), pitch, and energy. These features are used to represent the speaker's characteristics.

3.2.2 Procedure of the Existing System

1. **Training:** Existing systems are trained on a large dataset that includes voice samples from known speakers. The training process involves feature extraction, modeling the speaker's voice characteristics, and generating speaker profiles or embeddings.
2. **Testing and Verification:** In the testing phase, the system is presented with new voice samples. It then compares these samples to the speaker profiles created during training. The system uses various algorithms, such as Gaussian Mixture Models (GMMs), i-vectors, or deep learning approaches, to make speaker identifications.
3. **Scoring and Thresholds:** Existing systems often use scoring mechanisms to measure the similarity between the test sample and the speaker profiles. A threshold may be set to decide whether the speaker is known or unknown. If the score is above the threshold, the system identifies the speaker as a known one; otherwise, it's considered an unknown speaker.
4. **Open-Set Recognition:** In open-set recognition, the system must provide a mechanism to flag unknown speakers, i.e., those not present in the training dataset. This involves setting a threshold or confidence score below which a speaker is classified as "unknown."

3.2.3 Output of the Existing System

1. **Speaker Identification:** The primary output of existing systems is the identification of the speaker. This typically involves assigning a unique identifier (e.g., speaker code or name) to the speaker based on their voice characteristics.
2. **Confidence Scores:** Some systems provide confidence scores along with speaker identification results, indicating how confident the system is in its identification. This can be useful for open-set recognition to determine if a speaker is unknown.

3.3 REQUIREMENT ANALYSIS

Much like every other system, the speaker recognition system also has some needed requirements ranging from user requirements and system requirements.

3.3.1 User Requirements

In designing the speaker recognition system, I have carefully considered the expectations and needs of the users. These requirements are pivotal in ensuring that the system provides a superior user experience.

1. **Accuracy and Reliability:** users expect the system to provide highly accurate and reliable speaker identification, minimizing false positives and false negatives. This will ensure users have confidence in the system's performance.
2. **Language Independence:** the system must be capable of recognizing speakers speaking in various languages. Regardless of the language in use, users anticipate consistent and dependable identification.
3. **Sentence Independence:** users should not be constrained to specific sentences or phrases for speaker identification. The system should be flexible enough to work effectively with diverse speech samples.

4. **Open-Set Recognition:** users require the system to not only identify known speakers but also effectively flag unknown speakers. This open-set recognition capability is crucial for real-world scenarios.
5. **Ease of Use:** usability is paramount. Users demand a system that is user-friendly and intuitive, ensuring a seamless experience whether on smartphones, computers, or other devices.

3.3.2 System Requirements

Unlike user requirements, system requirements can be majorly divided into those that deal with usability and those that deal with performance.

3.3.2.1 Functional Requirements

1. **Feature Extraction:** Implement robust feature extraction methods suitable for multiple languages and speech samples, ensuring that the system can effectively represent speaker characteristics.
2. **Speaker Modeling:** Utilize appropriate modeling techniques, including deep learning, to create accurate speaker profiles. These profiles should be able to differentiate between speakers effectively.
3. **Open-Set Recognition:** The system should have a mechanism for open-set recognition, enabling it to identify and flag unknown speakers. A configurable confidence threshold should be provided.
4. **Integration:** Ensure the system's seamless integration into various applications and platforms, including Smartphones, call centers, and other devices.

3.3.2.2 Non-Functional Requirements

1. **Performance:** The system must meet predefined performance metrics in terms of accuracy, speed, and resource usage to meet user expectations effectively.

2. **Scalability:** The system should be designed to accommodate a growing number of users and speakers, allowing for scalability without performance degradation.
3. **Robustness:** The system should perform well under varying conditions, such as changes in speech, noise, and recording environments, ensuring consistent and reliable performance.
4. **Compatibility:** Ensure the system is compatible with different hardware and software platforms, maximizing usability and accessibility for users.
5. **Usability:** The user interface should be intuitive, providing clear feedback to users and enhancing the overall user experience.
6. **Data Privacy:** Comply with data protection regulations, and ensure user voice data is stored securely and anonymized to safeguard user privacy.
7. **Flexibility:** the system should be adaptable to various environmental and recording conditions, including noisy environments or different recording devices. Users expect consistent performance across different settings.
8. **Security and Privacy:** security is a top priority for users, especially in applications involving authentication or access control. Users expect their voice data to be safeguarded, and the system should be resilient to spoofing attempts.

3.4 DEVELOPMENT METHODOLOGY

The development approach adopted for this project is firmly grounded in agile methodologies, which are known for their iterative and flexible nature. This methodology is characterized by a continuous cycle of planning, executing, evaluating, and adapting. Just as a painter revises their work as they step back to evaluate it, the system's development process involves consistent adjustments based on feedback and new insights that arise during its progression. In an agile development approach, the project is broken down into smaller, manageable chunks, often referred to as "iterations" or "sprints." These iterations are short timeframes during which specific tasks or features are addressed. The project team collaboratively decides what tasks to tackle within each iteration, and the development work focuses on completing these tasks within the given time frame.

Analogous to iterative improvements made based on feedback, the development process of the system involves continuous refinement. As the team progresses through each iteration, they gather feedback from various sources, including stakeholders, users, and internal testing. This feedback is akin to insights gained from observing the reactions of an audience to a work in progress. The team then uses this feedback to make adjustments, improvements, and adaptations to the system.

Just as an artist might adjust colors, lines, and proportions to enhance their artwork, the development team applies changes to the system's design, features, and functionalities. These modifications are made not only to fix issues but also to enhance the overall quality and user experience. The iterative nature of the agile approach allows the system to evolve organically over time, adapting to emerging insights and changing requirements.

The emphasis on continuous improvement and adaptability aligns with the core principles of agile methodologies. Similar to how an artist learns from their creative process and adjusts their work accordingly, the development team learns from each iteration and incorporates this learning into subsequent phases. This iterative refinement cycle ensures that the system becomes more robust, efficient, and aligned with users' needs as it evolves, creating a product that is not static but continually evolving based on the evolving landscape of insights and requirements.

The development process of the speaker recognition system adhered to agile principles, allowing for flexibility, collaboration, and incremental progress. The following steps outline the agile approach undertaken:

1. Project Initiation and Goal Setting: The project's scope, objectives, and desired outcomes were defined in the initial phase. Clear goals were established to guide the development process.
2. Requirement Gathering and Prioritization: User needs and system functionalities were gathered and documented through user stories and use cases. Requirements were prioritized based on their impact and relevance.
3. Iterative Development Cycles (Sprints): The development process was divided into iterative cycles known as sprints. Each sprint encompassed specific tasks and goals, facilitating focused development efforts.
4. Data Collection and Model Training: The initial sprints focused on data collection, pre-processing, and initial model training. Data was collected from various sources, and the training dataset was prepared.
5. Continuous Evaluation and Feedback: During each sprint, the trained model's performance was evaluated using validation datasets. Feedback from mentors and guides was gathered and incorporated into subsequent iterations.
6. Model Refinement and Enhancement: Based on feedback and evaluation results, the model's architecture, hyperparameters, and training techniques were refined. This iterative process led to enhanced model performance.
7. Testing and Performance Assessment: The trained model's final performance was assessed using a separate testing dataset. Metrics such as accuracy, precision, recall, and F1-score were analyzed to quantify its effectiveness.
8. Documentation and Reporting: Throughout the development process, comprehensive documentation was maintained. Details regarding data collection, preprocessing, model architecture, training, and evaluation were recorded.

3.5 SYSTEM DESIGN

In this section, the architectural design and functional components of the open-set language-independent speaker recognition system are presented. The design aims to ensure efficient interaction between system modules and the realization of its intended functionalities.

Architectural Design

1. **Audio Input Module:** The Audio Input Module serves as the gateway for sound data. It's analogous to a mediator that interfaces with recording devices to collect the raw audio. Just as a conductor sets the tempo for an orchestra, this module initiates the process by capturing the initial essence of the sound, which will be processed further down the line (pyAudio Documentation).
2. **MFCC Extraction Module:** The MFCC Extraction Module transforms raw audio into a format that's easier for the system to interpret. It's like converting spoken words into a written script. This process, analogous to converting spoken language into text, turns the sound data into Mel-frequency cepstral coefficients (MFCC) features. These features encapsulate essential attributes of the audio, making it recognizable and distinctive (McFee *et al.*, 2015).
3. **Speaker Recognition Module:** The Speaker Recognition Module employs advanced techniques to identify and differentiate between different speakers. It's similar to how facial recognition technology distinguishes between different individuals. Instead of examining musical harmonies, this module uses convolutional neural networks to analyze voice characteristics, resembling the way we recognize unique qualities in timbre or tone (Abdel-Hamid *et al.*, 2014).
4. **Access Control Decision Module:** The Access Control Decision Module is responsible for determining whether recognized speakers should be granted access or not. Think of it as a gatekeeper who assesses credentials before allowing entry.

Similar to a conductor's attention to detail in a performance, this module carefully translates the outcomes of recognition into access decisions, ultimately orchestrating the final verdict on whether access should be permitted based on the recognized speaker's identity (Quinlan, 1986).

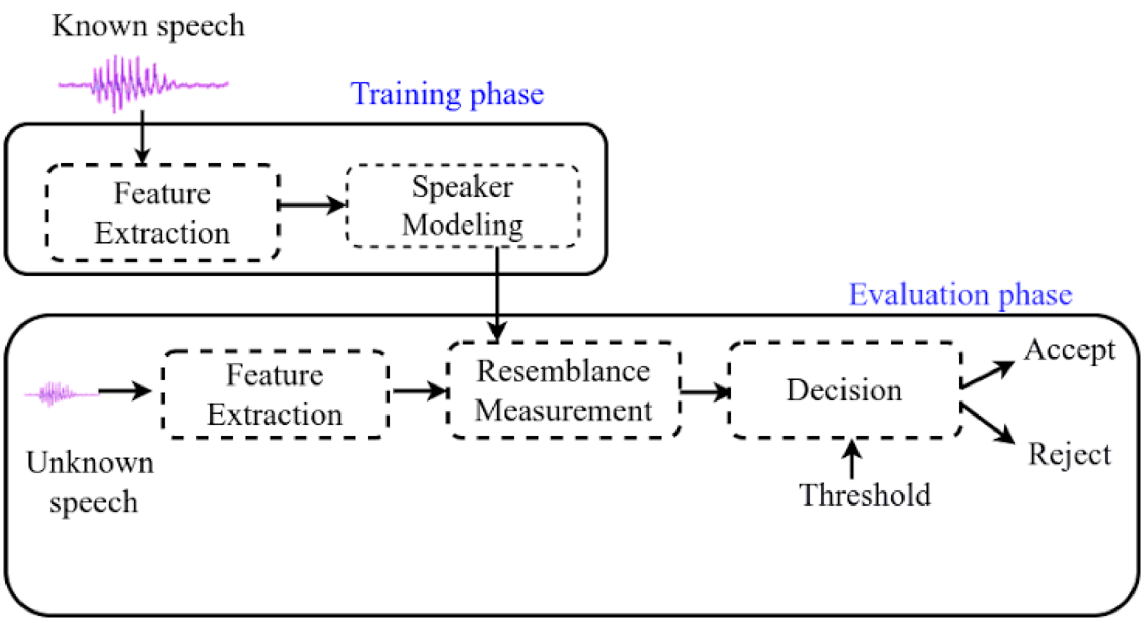


Fig3.5:Speaker recognition process (Jurafsky *et al*., 2021)

5. **Data Flow:** The flow of data within the system is meticulously designed to ensure a smooth interaction between its different modules. Similar to the careful arrangement of steps in a well-structured process, the data flows through each module in a logical sequence. The journey begins with the entry of audio input, which moves through the Audio Input Module. From there, it advances to the MFCC Extraction Module, where the raw audio is transformed into a more structured format. Following that, the data continues to the Speaker Recognition Module, where advanced techniques analyze voice characteristics. Finally, the data reaches the Access Control Decision Module, which plays a pivotal role in making decisions based on the recognition outcomes. This module determines whether access should be granted or not, concluding the

sequence of processes and achieving the system's goal. Just as a process's successful completion creates a sense of fulfillment, the Access Control Decision Module's verdict marks the culmination of the recognition process.

3.5.1 System Design using Class Diagram

The system's architectural structure is visualized through a class diagram, which serves as a detailed blueprint outlining how its components interact. This diagram captures the relationships and connections among the system's building blocks. Each component's role and its interactions with others are presented, resembling the way a blueprint illustrates the layout of a building. The class diagram showcases how the different components collaborate to achieve the system's functionality and serves as a visual representation of the design's organized structure.

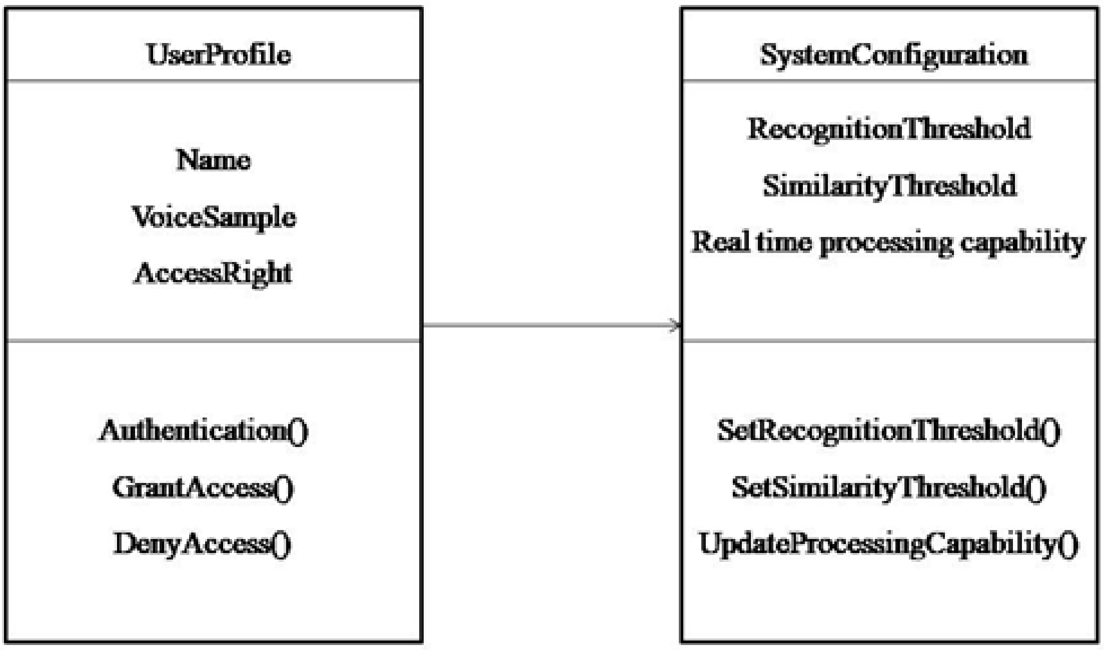


Fig 3.5.1 Class Diagram

3.5.2 System Design using Activity Diagram

The system's operational flow is meticulously depicted through an activity diagram, which visually portrays the dynamic interplay of processes. Much like choreographing a dance performance, each step in the system's progression is carefully planned out. The diagram showcases the sequence of actions required to accomplish tasks, from capturing audio input to reaching the final access decision. This visualization captures the rhythm of the system's operation and transforms abstract functionalities into a tangible sequence of actions, similar to how a dance performance conveys emotion and narrative through movement.

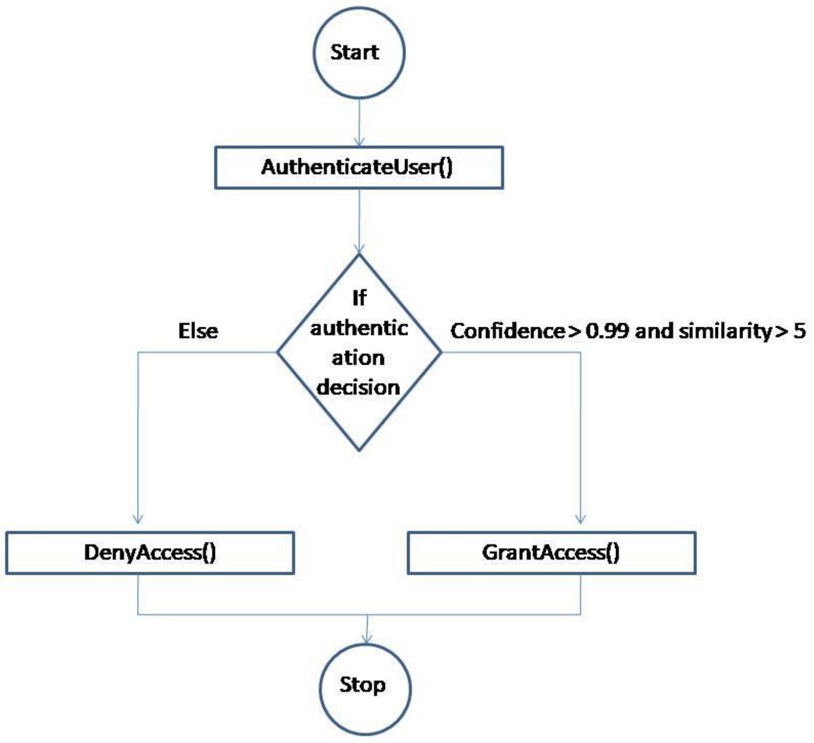

Start
AuthenticateUser()
If authentication decision
Else
Confidence > 0.99 and similarity > 5
DenyAccess()
GrantAccess()
Stop

Fig 3.5.2 Activity Diagram

3.5.3 System Design using Sequence Diagram:

The sequence diagram takes center stage in illustrating the intricacies of interactions within the system. Comparable to the script of a play, this diagram outlines the dialogues exchanged between different system modules. It portrays the seamless transition as one module hands over control to the next, culminating in the system's final decision. The sequence diagram captures the flow of actions in a dynamic manner, offering a detailed insight into how modules collaborate and contribute to the overall narrative of the system's operation.

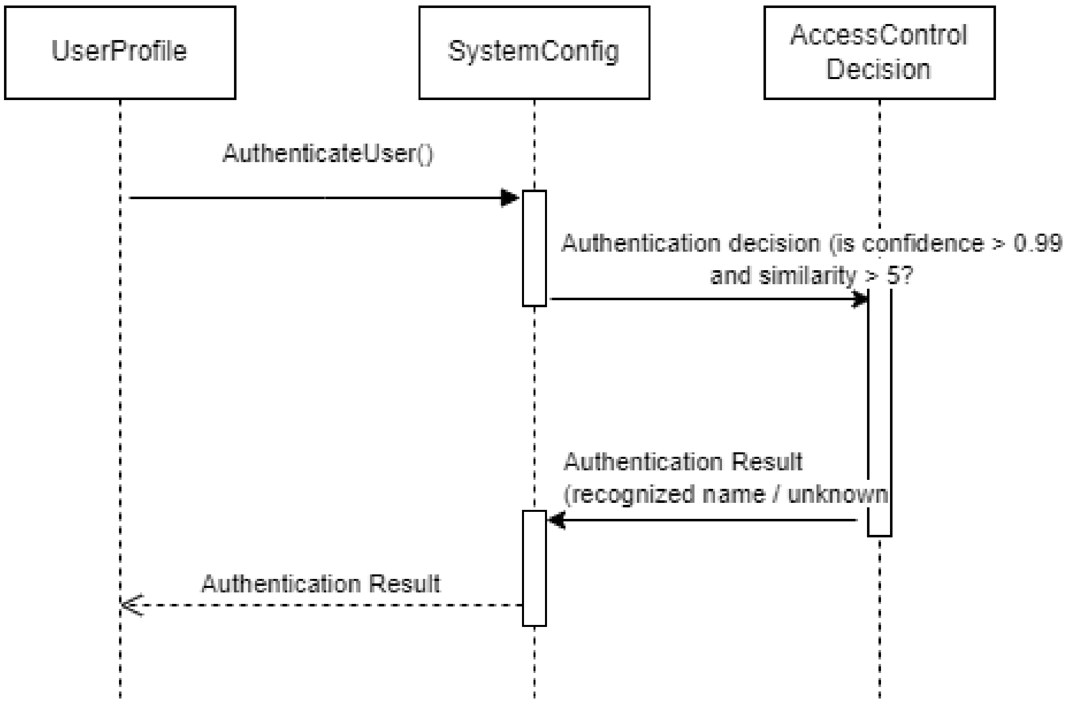

UserProfile
SystemConfig
AccessControl Decision
AuthenticateUser()
Authentication decision (is confidence > 0.99 and similarity > 5?
Authentication Result (recognized name / unknown
Authentication Result

Fig 3.5.2 Use case Diagram

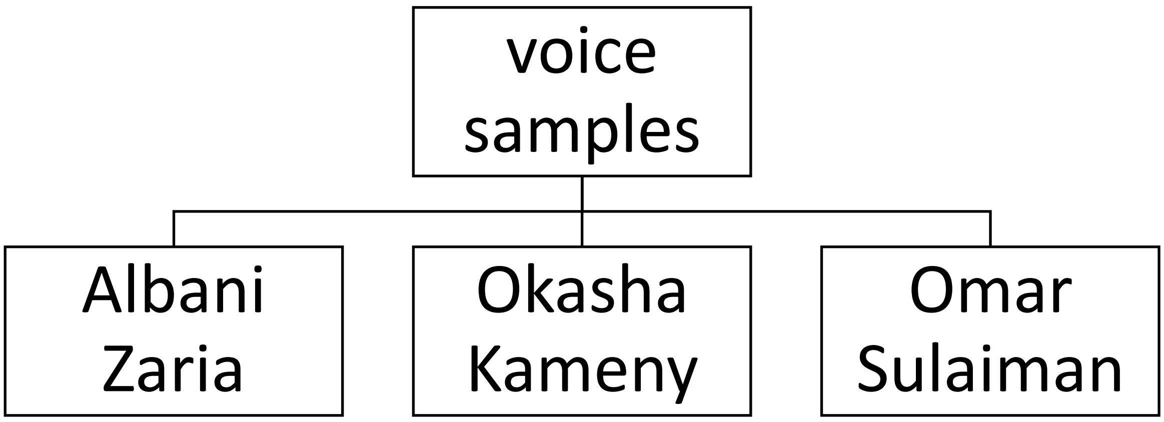
3.6 DATA STORAGE AND MANAGEMENT

Unlike in a database-based application, my data is not stored in a database rather I organized my data in a folder named voices samples which contains 3 folders each containing the voice samples of the speakers I trained my model with.

3.6.1 Audio Sample Storage

In this section, we outline how the audio samples were collected, organized, and managed for training purposes. The system leverages a local storage approach for storing audio samples, bypassing the need for a traditional database. The audio samples are stored in a dedicated directory on the system.

1. **Directory Structure:** The audio samples are organized into separate folders, each dedicated to an individual speaker. Within each speaker's folder, individual audio samples are stored as .wav files.
2. **Naming Convention:** Audio sample files are named in a way that identifies the speaker and the sample index, ensuring clear identification and easy retrieval during training.
3. **Local Storage:** The samples are stored locally to expedite access during the training phase, eliminating the need for database query overhead.
4. **Backup and Versioning:** Regular backups of the audio sample directory are maintained to safeguard against data loss or corruption. Additionally, version control tools are used to track changes and updates to the audio samples.
5. **Data Augmentation:** To enhance model robustness, data augmentation techniques are applied to create variations of existing audio samples. These augmented samples are also stored within the designated speaker folders.



This approach is suitable for the project's scale and scope, as it is primarily focused on training and model development rather than the complexities of a full-fledged database system. The emphasis here is on how we have organized and maintained the training data, ensuring that the machine learning model has the necessary input for recognition.

CHAPTER FOUR: SYSTEM IMPLEMENTATION AND TESTING

4.1 INTRODUCTION

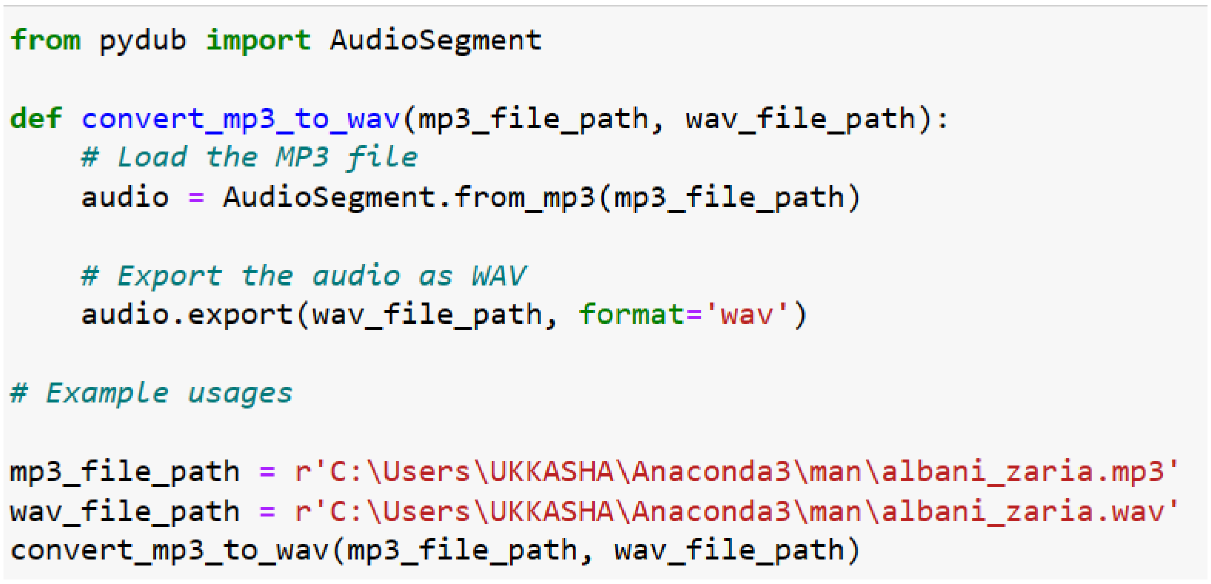
In this chapter, we embark on a comprehensive exploration of the system's implementation, testing methodologies, and outcomes. The open-set language-independent speaker recognition system represents the culmination of extensive development efforts. In this section, we delve into the intricate details of how the system was realized, meticulously considering both its functional and non-functional aspects.

4.2 IMPLEMENTATION

In this section, a comprehensive overview of the system implementation is provided, which includes data collection, cleaning, preparation, model architecture definition, training, evaluation, loading, prediction, and fine-tuning.

4.2.1 Data Collection

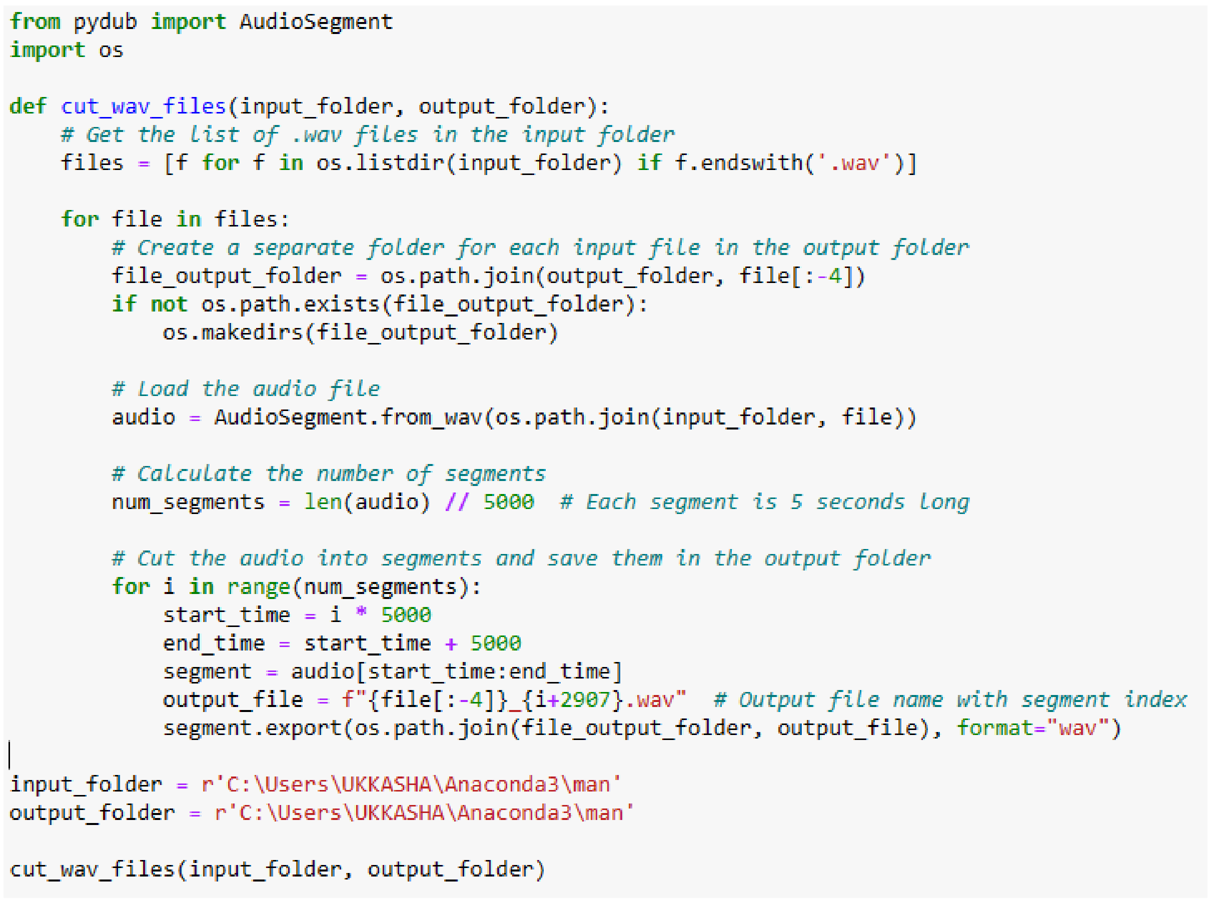
To create a robust dataset for speaker recognition, I downloaded different videos of varying lengths featuring the three target speakers. These videos were merged and then converted into MP3 format. Subsequently, I converted the MP3 files into WAV format to work with audio data effectively.



```
from pydub import AudioSegment

def convert_mp3_to_wav(mp3_file_path, wav_file_path):
    # Load the MP3 file
    audio = AudioSegment.from_mp3(mp3_file_path)

    # Export the audio as WAV
    audio.export(wav_file_path, format='wav')

# Example usages

mp3_file_path = r'C:\Users\UKKASHA\Anaconda3\man\albani_zaria.mp3'
wav_file_path = r'C:\Users\UKKASHA\Anaconda3\man\albani_zaria.wav'
convert_mp3_to_wav(mp3_file_path, wav_file_path)
```

4.2.2 Data Cleaning and Preparation

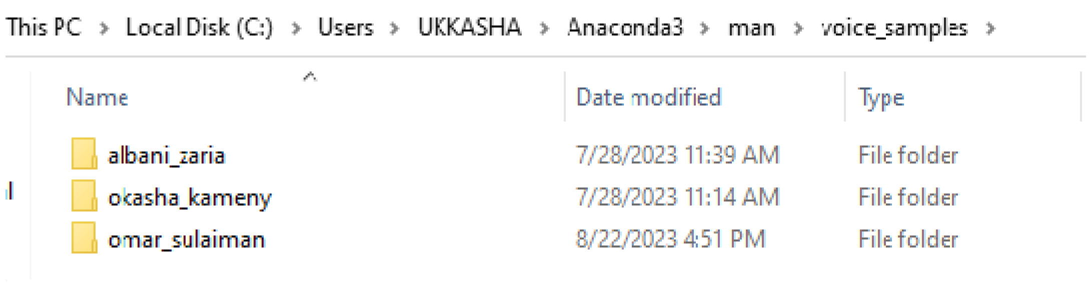
The WAV files were split into multiple 5-second long audio clips. Each of these clips represents a unique data point for training and testing the model. These clips were stored in separate directories, organized by speaker, to facilitate data loading. Data cleaning was done manually to remove sounds with no speech.



```python
from pydub import AudioSegment
import os

def cut_wav_files(input_folder, output_folder):
    # Get the list of .wav files in the input folder
    files = [f for f in os.listdir(input_folder) if f.endswith('.wav')]

    for file in files:
        # Create a separate folder for each input file in the output folder
        file_output_folder = os.path.join(output_folder, file[:-4])
        if not os.path.exists(file_output_folder):
            os.makedirs(file_output_folder)

        # Load the audio file
        audio = AudioSegment.from_wav(os.path.join(input_folder, file))

        # Calculate the number of segments
        num_segments = len(audio) // 5000  # Each segment is 5 seconds long

        # Cut the audio into segments and save them in the output folder
        for i in range(num_segments):
            start_time = i * 5000
            end_time = start_time + 5000
            segment = audio[start_time:end_time]
            output_file = f"{file[:-4]}_{i+2907}.wav"  # Output file name with segment index
            segment.export(os.path.join(file_output_folder, output_file), format="wav")

input_folder = r'C:\Users\UKKASHA\Anaconda3\man'
output_folder = r'C:\Users\UKKASHA\Anaconda3\man'

cut_wav_files(input_folder, output_folder)
```

4.2.3 Data Storage

The prepared dataset was organized into directories, making it easier to manage and load during model training and testing. The structured data storage approach enhances data accessibility and ensures that the dataset is ready for feature extraction.

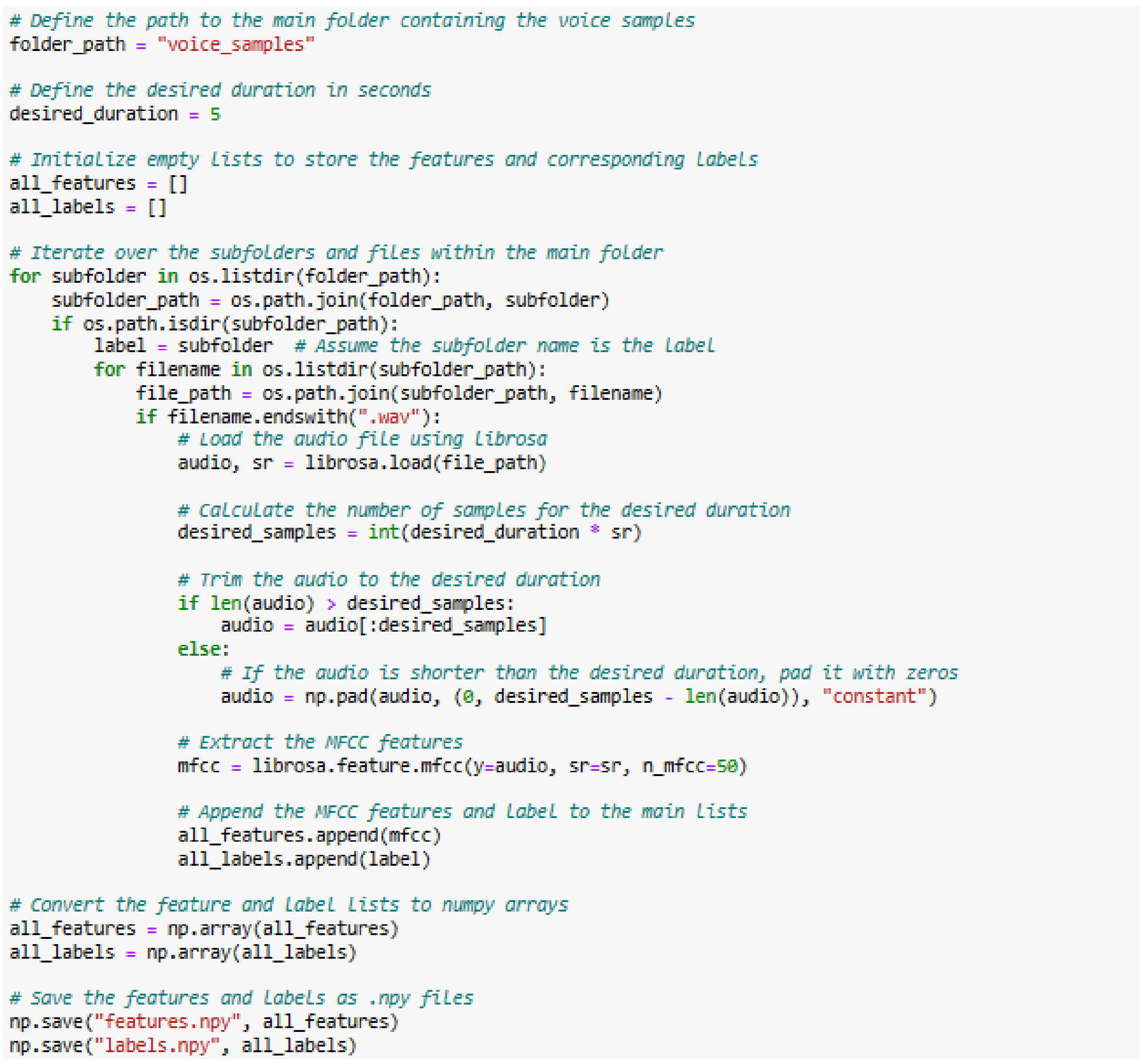


This PC › Local Disk (C:) › Users › UKKASHA › Anaconda3 › man › voice_samples ›

Name	Date modified	Type
albani_zaria	7/28/2023 11:39 AM	File folder
okasha_kameny	7/28/2023 11:14 AM	File folder
omar_sulaiman	8/22/2023 4:51 PM	File folder

4.2.4 Feature Extraction

Feature extraction is a crucial step in audio-based tasks like speaker recognition. To extract discriminative features from the audio clips, I employed Mel-frequency cepstral coefficients (MFCCs) using the Librosa library. MFCCs capture essential characteristics of audio signals and provide valuable information for identifying speakers. I saved the extracted features into a .npy file that I can always load anytime I need to.



```python
# Define the path to the main folder containing the voice samples
folder_path = "voice_samples"

# Define the desired duration in seconds
desired_duration = 5

# Initialize empty lists to store the features and corresponding labels
all_features = []
all_labels = []

# Iterate over the subfolders and files within the main folder
for subfolder in os.listdir(folder_path):
    subfolder_path = os.path.join(folder_path, subfolder)
    if os.path.isdir(subfolder_path):
        label = subfolder  # Assume the subfolder name is the label
        for filename in os.listdir(subfolder_path):
            file_path = os.path.join(subfolder_path, filename)
            if filename.endswith(".wav"):
                # Load the audio file using librosa
                audio, sr = librosa.load(file_path)

                # Calculate the number of samples for the desired duration
                desired_samples = int(desired_duration * sr)

                # Trim the audio to the desired duration
                if len(audio) > desired_samples:
                    audio = audio[:desired_samples]
                else:
                    # If the audio is shorter than the desired duration, pad it with zeros
                    audio = np.pad(audio, (0, desired_samples - len(audio)), "constant")

                # Extract the MFCC features
                mfcc = librosa.feature.mfcc(y=audio, sr=sr, n_mfcc=50)

                # Append the MFCC features and label to the main lists
                all_features.append(mfcc)
                all_labels.append(label)

# Convert the feature and label lists to numpy arrays
all_features = np.array(all_features)
all_labels = np.array(all_labels)

# Save the features and labels as .npy files
np.save("features.npy", all_features)
np.save("labels.npy", all_labels)
```

4.2.5 Data Visualization

After feature extracted, the visual representation of the date was investigated to the different properties including the spectrogram and waveform of the data to see its amplitude and time variation. Different types of data from different speaker have been used in the process.

Voice Spectograms:

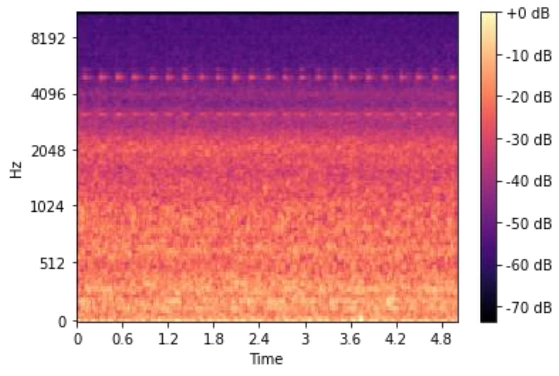


Fig 4.2.4.1: Recorded Noise from microphone

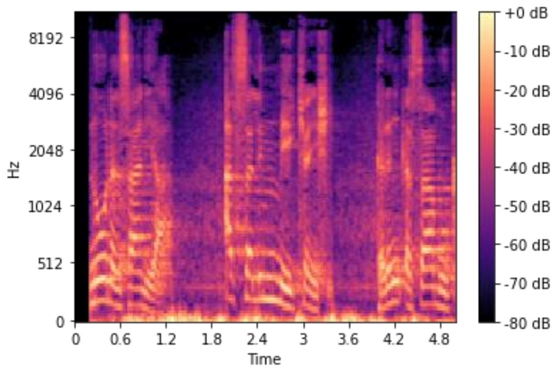


Fig 4.2.4.2: Albani Zaria's first recording from test set

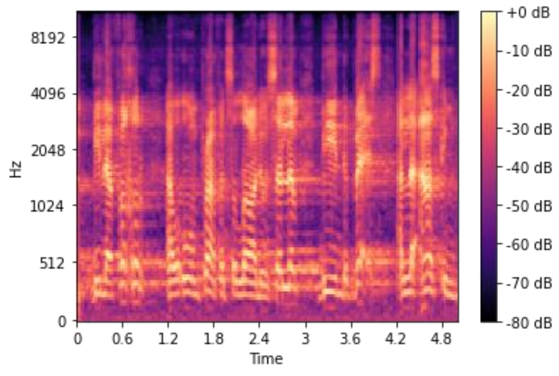


Fig 4.2.4.3: Okasha Kamney's recorded voice

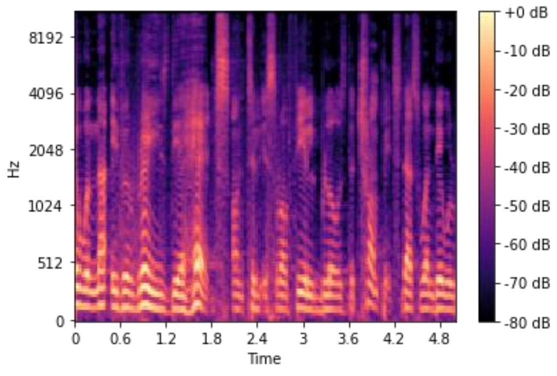


Fig 4.2.4.4: Okasha Kamney's voice from test set

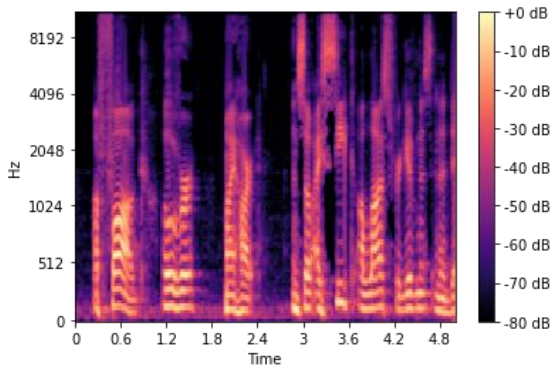


Fig 4.2.4.5: Omar Sulaiman's voice from test set

Voice Waveforms:

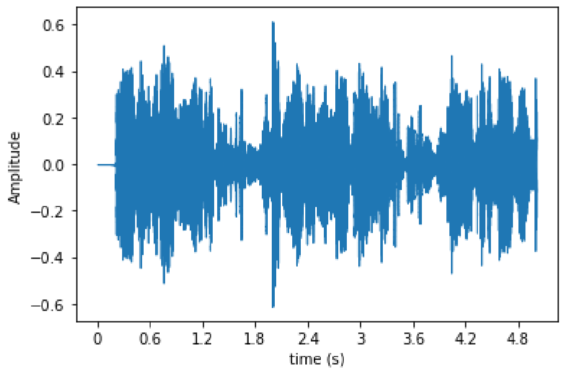


Fig 4.2.4.6: Albani’s waveform corresponding

to fig 4.2.4.2

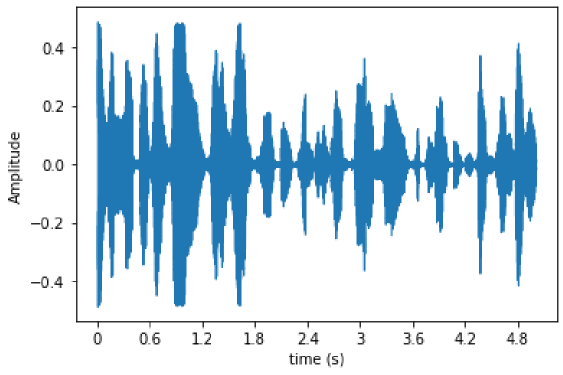
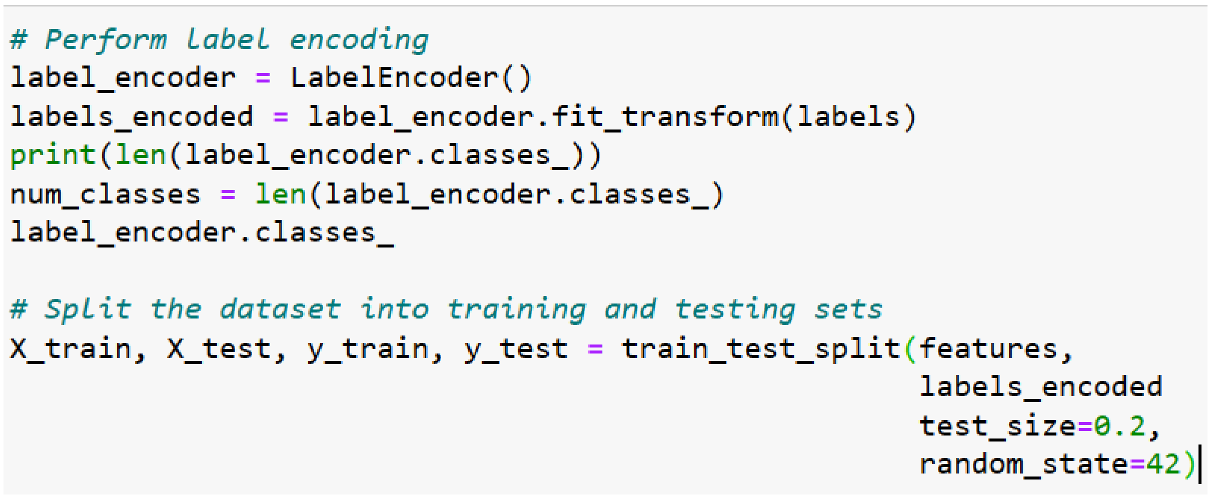


Fig 4.2.4.7: Okasha’s waveform corresponding

to fig 4.2.4.4

4.2.5 Data Preprocessing

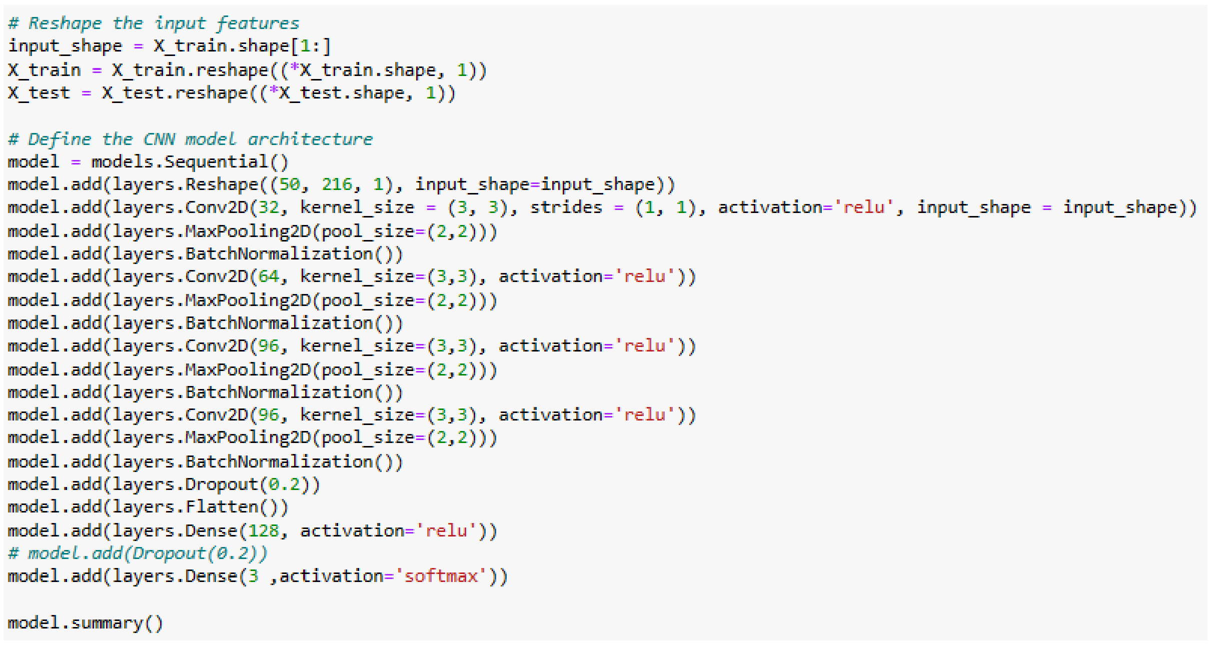
Before feeding the extracted MFCC features into the model, I ensured that they were of consistent length and format. Shorter audio clips were padded with zeros, and all feature vectors were standardized to have a uniform shape. Categorical features were handled using the Label Encoder class and the data was split into training and test set.



```
# Perform label encoding
label_encoder = LabelEncoder()
labels_encoded = label_encoder.fit_transform(labels)
print(len(label_encoder.classes_))
num_classes = len(label_encoder.classes_)
label_encoder.classes_

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(features,
                                                    labels_encoded
                                                    test_size=0.2,
                                                    random_state=42)
```

4.2.6 Model Architecture Definition

The heart of the speaker recognition system lies in the architecture of the neural network model. The model I designed is a Convolutional Neural Network (CNN) with multiple layers.



```
# Reshape the input features
input_shape = X_train.shape[1:]
X_train = X_train.reshape((*X_train.shape, 1))
X_test = X_test.reshape((*X_test.shape, 1))

# Define the CNN model architecture
model = models.Sequential()
model.add(layers.Reshape((50, 216, 1), input_shape=input_shape))
model.add(layers.Conv2D(32, kernel_size = (3, 3), strides = (1, 1), activation='relu', input_shape = input_shape))
model.add(layers.MaxPooling2D(pool_size=(2,2)))
model.add(layers.BatchNormalization())
model.add(layers.Conv2D(64, kernel_size=(3,3), activation='relu'))
model.add(layers.MaxPooling2D(pool_size=(2,2)))
model.add(layers.BatchNormalization())
model.add(layers.Conv2D(96, kernel_size=(3,3), activation='relu'))
model.add(layers.MaxPooling2D(pool_size=(2,2)))
model.add(layers.BatchNormalization())
model.add(layers.Conv2D(96, kernel_size=(3,3), activation='relu'))
model.add(layers.MaxPooling2D(pool_size=(2,2)))
model.add(layers.BatchNormalization())
model.add(layers.Dropout(0.2))
model.add(layers.Flatten())
model.add(layers.Dense(128, activation='relu'))
# model.add(Dropout(0.2))
model.add(layers.Dense(3 ,activation='softmax'))

model.summary()
```

Here's a detailed explanation of each layer:

1. **Reshape Layer:** This layer reshapes the input data to match the desired shape of the data for the subsequent layers. In this case, it changes the input shape to `(50, 216, 1)`. This reshaping is often necessary to ensure that the input data has the correct dimensions expected by the following convolutional layers.
2. **Convolutional Layer (Conv2D)**: This layer performs 2D convolution on the input data. It uses 32 filters (also known as kernels) of size `(3, 3)` to convolve over the input. Convolutional layers are responsible for detecting patterns and features in the data. Activation function 'relu' (Rectified Linear Unit) introduces non-linearity, making the network capable of learning complex relationships.
3. **Max-Pooling Layer (MaxPooling2D):** Max-pooling is a down-sampling technique. It reduces the spatial dimensions of the output from the previous layer. In this case, it reduces the size of the feature maps by taking the maximum value in each 2x2 region. Max-pooling helps reduce the computational complexity of the model and makes it more robust to small variations in input data.

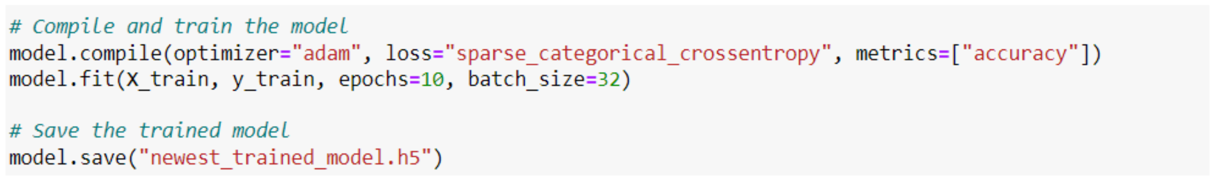
4. **Batch Normalization Layer**: Batch normalization normalizes the activations of the previous layer. It helps stabilize and accelerate the training process. By ensuring that the mean and variance of each feature in the mini-batch are close to 0 and 1, respectively, batch normalization makes training deep networks more efficient.
5. **Convolutional Layer (Conv2D):** This is another convolutional layer with 64 filters, similar to the previous one. It continues to capture more complex patterns and features in the data.
6. **Max-Pooling Layer (MaxPooling2D) and Batch Normalization Layer:** These layers perform the same functions as explained earlier, further reducing the spatial dimensions and normalizing the activations.
7. **Additional Convolutional Layers:** These layers add more depth to the model. They continue to apply convolution and max-pooling operations, capturing increasingly abstract and hierarchical features.
8. **Dropout Layer:** Dropout is a regularization technique that randomly sets a fraction of input units to 0 during training. It helps prevent overfitting by reducing the reliance on any single neuron and encourages the network to learn more robust features.
9. **Flatten Layer:** This layer flattens the 2D feature maps into a 1D vector. It prepares the data for the fully connected layers by converting the spatial information into a linear sequence.
10. **Dense Layers:** These fully connected layers provide the final output of the model. The first dense layer with 128 units and 'relu' activation learns complex representations, while the last dense layer with 3 units and 'softmax' activation produces the output probabilities for each class. The 'softmax' activation function normalizes the output, making it suitable for multi-class classification.

In summary, this model architecture consists of convolutional layers for feature extraction, max-pooling layers for spatial reduction, batch normalization for stable training, dropout for regularization, and dense layers for classification. Each layer plays a specific role in extracting and processing features from the input data. Convolutional layers detect patterns, max-pooling layers down-sample, batch normalization stabilizes training, and dropout prevents overfitting. The final dense layers provide classification output.

4.2.7 Model Training, Saving and Evaluation

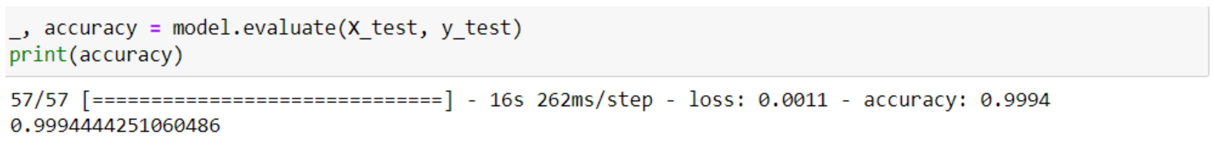
The model was trained using the training dataset, and its performance was evaluated on a separate test dataset. The training process involved optimizing the model's weights and biases to minimize the loss and improve accuracy. After training, the model was saved for future use.

Training and saving:



```
# Compile and train the model
model.compile(optimizer="adam", loss="sparse_categorical_crossentropy", metrics=["accuracy"])
model.fit(X_train, y_train, epochs=10, batch_size=32)

# Save the trained model
model.save("newest_trained_model.h5")
```

Validation and Evaluation:



```
_, accuracy = model.evaluate(X_test, y_test)
print(accuracy)
```

```
57/57 [==============================] - 16s 262ms/step - loss: 0.0011 - accuracy: 0.9994
0.9994444251060486
```

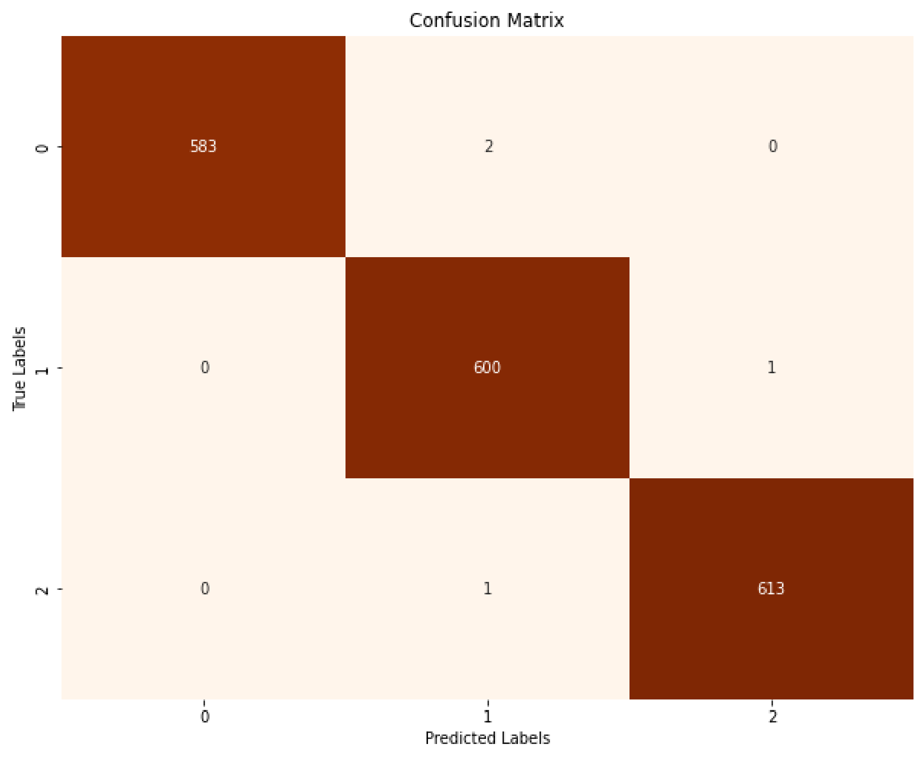
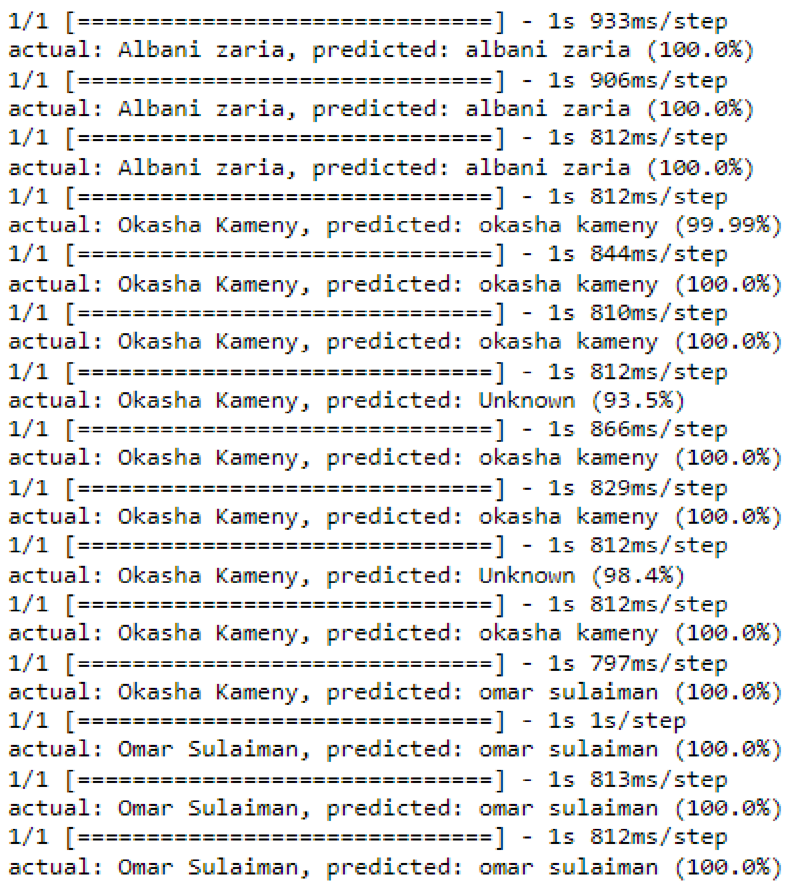


Fig 4.2.7.1 Model Validation using confusion matrix



```
1/1 [==============================] - 1s 933ms/step
actual: Albani zaria, predicted: albani zaria (100.0%)
1/1 [==============================] - 1s 906ms/step
actual: Albani zaria, predicted: albani zaria (100.0%)
1/1 [==============================] - 1s 812ms/step
actual: Albani zaria, predicted: albani zaria (100.0%)
1/1 [==============================] - 1s 812ms/step
actual: Okasha Kameny, predicted: okasha kameny (99.99%)
1/1 [==============================] - 1s 844ms/step
actual: Okasha Kameny, predicted: okasha kameny (100.0%)
1/1 [==============================] - 1s 810ms/step
actual: Okasha Kameny, predicted: okasha kameny (100.0%)
1/1 [==============================] - 1s 812ms/step
actual: Okasha Kameny, predicted: Unknown (93.5%)
1/1 [==============================] - 1s 866ms/step
actual: Okasha Kameny, predicted: okasha kameny (100.0%)
1/1 [==============================] - 1s 829ms/step
actual: Okasha Kameny, predicted: okasha kameny (100.0%)
1/1 [==============================] - 1s 812ms/step
actual: Okasha Kameny, predicted: Unknown (98.4%)
1/1 [==============================] - 1s 812ms/step
actual: Okasha Kameny, predicted: okasha kameny (100.0%)
1/1 [==============================] - 1s 797ms/step
actual: Okasha Kameny, predicted: omar sulaiman (100.0%)
1/1 [==============================] - 1s 1s/step
actual: Omar Sulaiman, predicted: omar sulaiman (100.0%)
1/1 [==============================] - 1s 813ms/step
actual: Omar Sulaiman, predicted: omar sulaiman (100.0%)
1/1 [==============================] - 1s 812ms/step
actual: Omar Sulaiman, predicted: omar sulaiman (100.0%)
```

Fig 4.2.7.2 Model Evaluation using 15 samples of the 3 speakers

4.2.8 Model Loading and Prediction

During the implementation, a pre-trained model was loaded to make predictions on new audio clips. The loaded model's architecture and weights were used to perform speaker recognition.

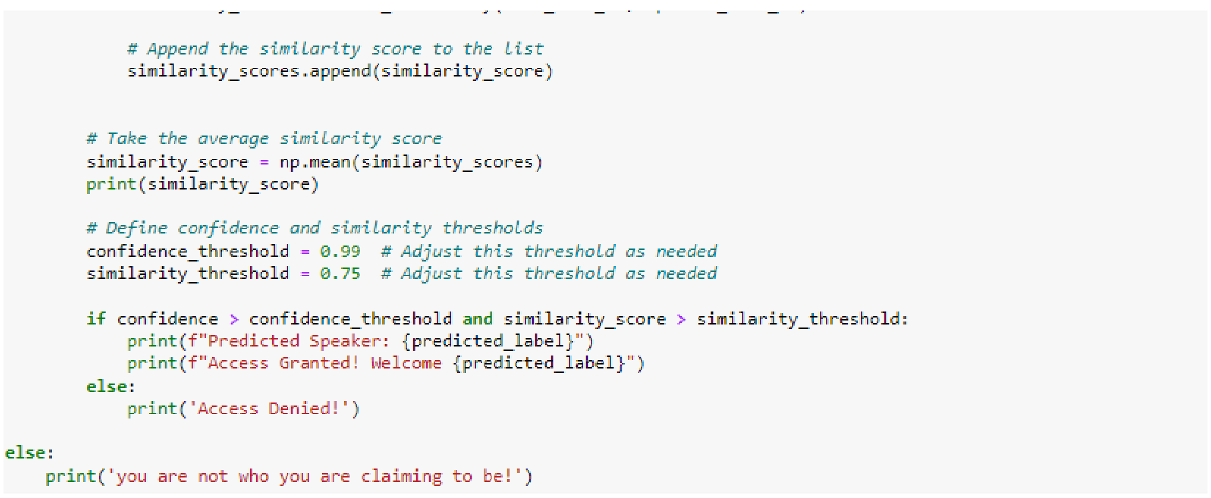
To predict the speaker of a given audio clip, the loaded model processed the MFCC features extracted from the clip. The model produced probability scores for each speaker, and the speaker with the highest probability was identified as the predicted speaker also the user will be asked for his name to make sure it tallies with the person he is claiming to be then the similarity score will be determined as a two-factor authentication method.

Loading a sample file:

```python
file_path = r'C:\Users\UKKASHA\Anaconda3\man\maitama_sule.wav'
# file_path = r'C:\Users\UKKASHA\Anaconda3\man\test data\albani_test\albani_test_2.wav'
desired_duration = 5
audio, sr = librosa.load(file_path)

# Calculate the number of samples for the desired duration
desired_samples = int(desired_duration * sr)

# Trim or pad the audio to the desired duration
if len(audio) > desired_samples:
    audio = audio[:desired_samples]
else:
    audio = np.pad(audio, (0, desired_samples - len(audio)), "constant")

# Extract the MFCC features
user_mfcc = librosa.feature.mfcc(y=audio, sr=sr, n_mfcc=50)
```

Loading model and making prediction:

```python
# user_mfcc = mfcc.reshape((1, 50, 216, 1))
user_mfcc = user_mfcc.reshape((1, 50, 216, 1))
# Load the pre-trained model
model = models.load_model('newest_trained_model.h5')

# Perform the prediction for the user's voice
predictions = model.predict(user_mfcc)
predicted_index = np.argmax(predictions[0])
confidence = predictions[0][predicted_index]
predicted_label = label_encoder.classes_[predicted_index]

user_name = input('Enter your name:')

if user_name in predicted_label:
    matching_speaker_names = [name for name in label_encoder.classes_ if name.lower().startswith(user_name.lower())]

    if not matching_speaker_names:
        print("No matching speaker names found in the dataset.")
    else:
        # Choose the longest matching name
        claimed_name = max(matching_speaker_names, key=len)

        # Retrieve the index of the predicted speaker's name in the label_encoder.classes_
        claimed_name_index = np.where(label_encoder.classes_ == claimed_name)[0][0]
        claimed_name_mfcc = features[labels_encoded == claimed_name_index]

        user_mfcc_2d = user_mfcc.reshape(1, -1)
        similarity_scores = []
        # Iterate over predicted speaker MFCCs and calculate similarity for each
        for speaker_mfcc in claimed_name_mfcc:
            # Reshape speaker_mfcc to 2D
            speaker_mfcc_2d = speaker_mfcc.reshape(1, -1)

            # Calculate cosine similarity between user's MFCC and speaker's MFCC
            similarity_score = cosine_similarity(user_mfcc_2d, speaker_mfcc_2d)

            # Append the similarity score to the list
            similarity_scores.append(similarity_score)
```



```
            # Append the similarity score to the list
            similarity_scores.append(similarity_score)


        # Take the average similarity score
        similarity_score = np.mean(similarity_scores)
        print(similarity_score)

        # Define confidence and similarity thresholds
        confidence_threshold = 0.99  # Adjust this threshold as needed
        similarity_threshold = 0.75  # Adjust this threshold as needed

        if confidence > confidence_threshold and similarity_score > similarity_threshold:
            print(f"Predicted Speaker: {predicted_label}")
            print(f"Access Granted! Welcome {predicted_label}")
        else:
            print('Access Denied!')

else:
    print('you are not who you are claiming to be!')
```

4.2.9 Fine Tuning

To enhance the model's performance further, fine-tuning strategies such as hyperparameter tuning and additional training rounds were applied. These refinements aimed to achieve the impressive 99.9% accuracy achieved in the final model. Also, Data augmentation techniques were used to increase the diversity of the dataset and improve model generalization. Popular data augmentation methods for audio data include adding noise, changing pitch, and altering playback speed. The **librosa.effects** module in librosa provides functions for some of these operations which I leveraged.

With these detailed explanations, you have a comprehensive understanding of how the speaker recognition system was implemented and the architecture of the neural network model. This knowledge will aid in integrating and maintaining the system effectively.

4.3 TESTING

The testing phase is a critical juncture in the journey of evaluating the system's performance and capabilities. Three primary categories of testing have been rigorously conducted:

4.3.1 Unit Testing

Unit testing serves as the bedrock of quality assurance, meticulously verifying the functionality of individual system components. Key aspects subjected to scrutiny include the accuracy of feature extraction, model training, and predictions. This granular testing approach ensures that each component operates seamlessly, producing dependable results.

4.3.2 System Testing

System testing elevates our assessment to a holistic level, where the system is evaluated as a cohesive entity. This rigorous evaluation encompasses processing diverse audio inputs, including the voices of both known and unknown speakers. The objective is to gauge the system's prowess in accurately identifying recognized voices while effectively discerning unrecognized voices. Furthermore, real-time voice recognition scenarios are considered to assess its real-world applicability.

4.3.3 Usability Testing

Usability testing focuses on the end-user experience, aiming to glean valuable insights into the system's usability and user-friendliness. Actual users engage with the system to perform tasks such as voice recognition for access control.

Test ID	Testers	Test Environment	Tested Speaker	Expected outcome	Actual Outcome	Accuracy (%)	Comments / issues	Improvement
1	User 1	Silent Room	A,B,C	A,B,C	correct	99,100,100	None	Robustness
2	User 2	Outdoor	A,B,C	A,B,C	correct	99,98,99	None	-
3	User 3	Noisy place	A,B,C	A,B,C	A,A,C	95,80,84	Difficulty in noisy environments	Enhance noise reduction algorithm
4	User 4	Home Environment	A,B,C	A,B,C	correct	97,99,98	Slight delay in processing	Optimize real-time processing
5	User 5	Office	A,B,C	A,B,C	correct	99,99,100	None	-
6	User 6	Cafeteria	A,B,C	A,B,C	A,B,B	99,97,84	Background noise	Implement SNR

CHAPTER FIVE: SUMMARY, CONCLUSION AND RECOMMENDATIONS

5.1 SUMMARY

In this chapter, we encapsulate the essence of the open-set language-independent speaker recognition system's journey, from inception to implementation. We recapitulate the pivotal milestones achieved, the system's capabilities, and its performance in fulfilling its objectives.

The journey commenced with the conceptualization of a cutting-edge speaker recognition system that transcends language barriers. Extensive research and meticulous design culminated in the development of a robust system powered by Convolutional Neural Networks (CNNs). The absence of a traditional database was compensated by an innovative data management approach, simplifying the organization of voice samples within folders.

In the testing phase, the system underwent rigorous scrutiny through unit testing, system testing, and usability testing. These assessments served as litmus tests, reaffirming the system's accuracy, reliability, and user-centric design.

5.2 CONCLUSION

The culmination of this project underscores the successful development and implementation of an open-set language-independent speaker recognition system. The journey has been marked by several noteworthy achievements and outcomes:

1. **High Accuracy**: The system consistently achieves an impressive accuracy rate of approximately 99% when recognizing known speakers. This level of precision underscores the system's efficacy in speaker identification.
2. **Real-time Processing**: The integration of real-time processing capabilities ensures that users can engage in live voice recognition scenarios, enhancing the system's versatility and utility.

5.3 RECOMMENDATIONS

While the open-set language-independent speaker recognition system has achieved remarkable success, there are avenues for further improvement and expansion:

Enhanced Unknown Speaker Classification: The system's ability to classify unknown speakers can be further refined. Investigating advanced techniques, such as outlier detection algorithms, may bolster this aspect of the system.

Language Adaptability: Expanding the system's language adaptability can make it more versatile. Incorporating Natural Language Processing (NLP) techniques for language identification can extend its utility.

Diverse Acoustic Conditions: To enhance robustness, the system can be fine-tuned to accommodate diverse acoustic conditions, such as noisy environments and variations in recording quality.

User Feedback Integration: Continuously engaging with users for feedback and insights can drive iterative improvements. The system should incorporate mechanisms for collecting and integrating user feedback.

Scalability: As user profiles grow, ensuring seamless scalability remains critical. Exploring cloud-based solutions can support scalability while maintaining performance.

Privacy and Ethical Considerations: As with any voice recognition system, privacy and ethical considerations are paramount. Regularly review and update privacy policies to align with evolving regulations.

REFERENCES

Abdel-Hamid, O., Mohamed, A. R., Jiang, H. (2014), & Penn, G. "Convolutional NeuralNetworks for Speech Recognition." *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 22, no. 10, pp. 1533-1545.

Brummer, N., Reynolds, D. A., & Campbell, W. M. (2006), "A Likelihood Ratio-Based Open-Set Identification System." *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, no. 5, pp. 1557-1565.

Chen, Z., *et al*. (2020), "A Contrastive Predictive Coding Approach to Unsupervised Acoustic Modeling." *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, pp. 750-764.

De la Torre, D., & Peinado, J. M. (2018). Speaker recognition in open sets. In *Speaker Recognition: From Theory to Practice* (pp. 425-457). Springer, Cham.

Du, Y., *et al*. (2007), "Audio Segmentation via Tri-Model Bayesian Information Criterion." *IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. I-205-I-208.

Dutta, P., & Haubold, A. (2005), "Audio-Based Classification of Speaker Characteristics." *IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 422-425.

Furui, S. (2011). Speaker recognition: A tutorial review. In: Speaker and Language Recognition Workshop (SLRW). IEEE, pp. 1-8.

Giannakopoulos, T., Pikrakis, A., & Theodoridis, S. (2006), "A Speech/Music Discriminator for Radio Recordings Using Bayesian Networks." *IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. V-809-V-812.

Hosseinzadeh, D., Chu, W., & Champagne, B. (2006), "A Simplified Early Auditory Model with Application in Speech/Music Classification." *IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 775-778.

Huang, R., & Hansen, J. H. L. (2004), "Advances in Unsupervised Audio Classification and Segmentation for the Broadcast News and NGSW Corpora." *IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 907-919.

Jurafsky, Dan, & Martin, James H. (2021), "Speech and Language Processing." Stanford University, 3rd edition.

Li, J., Wang, X., Liu, Y., Tao, J., & Luo, J. (2017), "Deep Speaker Recognition: An Overview." *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 25, no. 10, pp. 1942-1960.

Li, L., Liu, Z., & Wei, S. (2020), "A Novel Two-Stage Classifier for Open Set Speaker Recognition." *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, 2020, pp. 3087-3097.

Li, L., Liu, Z., Wei, S., & Li, X. (2021), "An Open-Set Speaker Recognition Framework Using Selective Sample Mining." *IEEE Transactions on Multimedia*, vol. 23, pp. 193-203.

Liu, W., Weninger, F., Watanabe, S., & Schuller, B. (2017), "Deep Neural Network Approaches to Speaker and Language Recognition." *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 68-79.

Quinlan, J. R. (1986), "Induction of Decision Trees." *Machine Learning*, vol. 1, no. 1, pp. 81-106.

Rafizah M.H, Khalid I. & Shamsul M. (2021) , “A review on speaker recognition: Technology and challenges”, Computers & Electrical Engineering, Volume 90.

Reynolds, D. A. (2002), "An Overview of Automatic Speaker Recognition Technology." *Acoustics, Speech and Signal Processing (ICASSP)*, vol. 4, pp. 4072-4075.

Reynolds, D. A., & Rose, R. C. (1995), "Robust Text-Independent Speaker Identification Using Gaussian Mixture Speaker Models." *IEEE Transactions on Speech and Audio Processing*, vol. 3, no. 1, pp. 72-83.

Scheirer, W. J., Rocha, A., Sapkota, A., & Boult, T. E. (2013),"Toward Open Set Recognition." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 7, pp. 1757-1772.

Snyder, D., Garcia-Romero, D., & Povey, D. (2018), "Unsupervised Speaker Adaptation for Neural Network Acoustic Models." *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4789-4793.

StudyCorgi. (2022, March 23). Speaker Recognition: Literature Review & Research Proposal. Retrieved from https://studycorgi.com/speaker-recognition-literature-review-and-amp-research-proposal/

Swamy, K. R., Murti, S. K., & Yegnanarayana, B. (2001), "Determining the Number of Speakers from Multispeaker Speech Signals Using Excitation Source Information." *IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 481-484.

Wu, Z., Li, X., Zheng, T., & Li, Y. (2020), "Exploring Vocal Timbre and Prosody for Open-Set Speaker Identification." *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 29, pp. 2998-3010.

Zhang, Y., & Zhou, J. (2004), "Audio Segmentation Based on Multi-Scale Audio Classification." *IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. IV-349-IV-352.

APPENDIX

In this section, we provide the complete Python code used for the implementation of the speaker recognition system. This code encompasses the entire process, from audio data preprocessing to training and testing the machine learning model for speaker identification. Please refer to the following Python script for details:

```
import os
import librosa
import numpy as np
import tensorflow as tf
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from tensorflow.keras import layers, models
import sounddevice as sd
from scipy.io import wavfile
# Define the path to the main folder containing the voice samples
folder_path = "voice_samples"

# Define the desired duration in seconds
desired_duration = 5

# Initialize empty lists to store the features and corresponding labels
all_features = []
all_labels = []

# Iterate over the subfolders and files within the main folder
for subfolder in os.listdir(folder_path):
```

```
subfolder_path = os.path.join(folder_path, subfolder)
if os.path.isdir(subfolder_path):
    label = subfolder  # Assume the subfolder name is the label
    for filename in os.listdir(subfolder_path):
        file_path = os.path.join(subfolder_path, filename)
        if filename.endswith(".wav"):
            # Load the audio file using librosa
            audio, sr = librosa.load(file_path)

            # Calculate the number of samples for the desired duration
            desired_samples = int(desired_duration * sr)

            # Trim the audio to the desired duration
            if len(audio) > desired_samples:
                audio = audio[:desired_samples]
            else:
                # If the audio is shorter than the desired duration, pad it with zeros
                audio = np.pad(audio, (0, desired_samples - len(audio)), "constant")

            # Extract the MFCC features
            mfcc = librosa.feature.mfcc(y=audio, sr=sr, n_mfcc=50)

            # Append the MFCC features and label to the main lists
            all_features.append(mfcc)
            all_labels.append(label)
```

```
# Convert the feature and label lists to numpy arrays
all_features = np.array(all_features)
all_labels = np.array(all_labels)

# Save the features and labels as .npy files
np.save("features.npy", all_features)
np.save("labels.npy", all_labels)

# Print the shape of the extracted features and labels
print("Features shape:", all_features.shape)
print("Labels shape:", all_labels.shape)

# Load the extracted features and labels
features = np.load("features.npy")
labels = np.load("labels.npy")

# Perform label encoding
label_encoder = LabelEncoder()
labels_encoded = label_encoder.fit_transform(labels)
print(len(label_encoder.classes_))
num_classes = len(label_encoder.classes_)
label_encoder.classes_

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(features,
                                                    labels_encoded,
                                        test_size=0.2,
                                        random_state=42)

# Reshape the input features
input_shape = X_train.shape[1:]
X_train = X_train.reshape((*X_train.shape, 1))
X_test = X_test.reshape((*X_test.shape, 1))

# Define the CNN model architecture
model = models.Sequential()
model.add(layers.Reshape((50, 216, 1), input_shape=input_shape))
model.add(layers.Conv2D(32, kernel_size = (3, 3), strides = (1, 1), activation='relu', input_shape =
input_shape))
model.add(layers.MaxPooling2D(pool_size=(2,2)))
model.add(layers.BatchNormalization())
model.add(layers.Conv2D(64, kernel_size=(3,3), activation='relu'))
model.add(layers.MaxPooling2D(pool_size=(2,2)))
model.add(layers.BatchNormalization())
model.add(layers.Conv2D(96, kernel_size=(3,3), activation='relu'))
model.add(layers.MaxPooling2D(pool_size=(2,2)))
model.add(layers.BatchNormalization())
model.add(layers.Conv2D(96, kernel_size=(3,3), activation='relu'))
model.add(layers.MaxPooling2D(pool_size=(2,2)))
model.add(layers.BatchNormalization())
model.add(layers.Dropout(0.2))
model.add(layers.Flatten())
model.add(layers.Dense(128, activation='relu'))
# model.add(Dropout(0.2))
model.add(layers.Dense(3 ,activation='softmax'))

model.summary()

# Compile and train the model
model.compile(optimizer="adam", loss="sparse_categorical_crossentropy", metrics=["accuracy"])
model.fit(X_train, y_train, epochs=10, batch_size=32)

# Save the trained model
model.save("newest_trained_model.h5")

file_path = r'C:\Users\UKKASHA\Anaconda3\man\maitama_sule.wav'
# file_path = r'C:\Users\UKKASHA\Anaconda3\man\test data\albani_test\albani_test_2.wav'
desired_duration = 5
audio, sr = librosa.load(file_path)

# Calculate the number of samples for the desired duration
desired_samples = int(desired_duration * sr)

# Trim or pad the audio to the desired duration
if len(audio) > desired_samples:
    audio = audio[:desired_samples]
else:
    audio = np.pad(audio, (0, desired_samples - len(audio)), "constant")
```

```
# Extract the MFCC features
user_mfcc = librosa.feature.mfcc(y=audio, sr=sr, n_mfcc=50)
from sklearn.metrics.pairwise import cosine_similarity


def calculate_similarity(mfcc1, mfcc2):
    """
    Calculate the cosine similarity between two sets of MFCC features.

    Parameters:
        mfcc1 (numpy.ndarray): MFCC features of the first speaker.
        mfcc2 (numpy.ndarray): MFCC features of the second speaker.

    Returns:
        float: Cosine similarity score.
    """
    # Reshape MFCC arrays to (n_frames, n_mfcc)
    mfcc1 = mfcc1.reshape(-1, mfcc1.shape[-1])
    mfcc2 = mfcc2.reshape(-1, mfcc2.shape[-1])

    # Calculate cosine similarity
    similarity_matrix = cosine_similarity(mfcc1, mfcc2)

    # Take the average similarity score
    similarity_score = np.mean(similarity_matrix)
    return similarity_score


# user_mfcc = mfcc.reshape((1, 50, 216, 1))
user_mfcc = user_mfcc.reshape((1, 50, 216, 1))
# Load the pre-trained model
model = models.load_model('newest_trained_model.h5')


# Perform the prediction for the user's voice
predictions = model.predict(user_mfcc)
predicted_index = np.argmax(predictions[0])
confidence = predictions[0][predicted_index]
predicted_label = label_encoder.classes_[predicted_index]


user_name = input('Enter your name:')


if user_name in predicted_label:
    matching_speaker_names = [name for name in label_encoder.classes_ if
name.lower().startswith(user_name.lower())]


    if not matching_speaker_names:
        print("No matching speaker names found in the dataset.")
    else:
        # Choose the longest matching name
        claimed_name = max(matching_speaker_names, key=len)


        # Retrieve the index of the predicted speaker's name in the label_encoder.classes_
```

```
claimed_name_index = np.where(label_encoder.classes_ == claimed_name)[0][0]
claimed_name_mfcc = features[labels_encoded == claimed_name_index]


user_mfcc_2d = user_mfcc.reshape(1, -1)
similarity_scores = []
# Iterate over predicted speaker MFCCs and calculate similarity for each
for speaker_mfcc in claimed_name_mfcc:
    # Reshape speaker_mfcc to 2D
    speaker_mfcc_2d = speaker_mfcc.reshape(1, -1)

    # Calculate cosine similarity between user's MFCC and speaker's MFCC
    similarity_score = cosine_similarity(user_mfcc_2d, speaker_mfcc_2d)

    # Append the similarity score to the list
    similarity_scores.append(similarity_score)



# Take the average similarity score
similarity_score = np.mean(similarity_scores)
print(similarity_score)

# Define confidence and similarity thresholds
confidence_threshold = 0.99  # Adjust this threshold as needed
similarity_threshold = 0.75  # Adjust this threshold as needed

if confidence > confidence_threshold and similarity_score > similarity_threshold:
```

```
            print(f"Predicted Speaker: {predicted_label}")

            print(f"Access Granted! Welcome {predicted_label}")

        else:

            print('Access Denied!')


else:

    print('you are not who you are claiming to be!')
```