Design Documentation

Identify Impersonation Challenge

Team marinaBeach

Introduction

In the banking domain, identity personification could be used to impersonate customers and gain access to their accounts. This could lead to financial losses for customers and banks. Banks need to be able to protect their customers from identity theft and fraud. One way to do this is to develop solutions that can detect and prevent AI-generated voice impersonation.

System Overview

The system aims to bolster voice verification capabilities by deploying artificial intelligence algorithms to distinguish between genuine human voices and AI-generated counterparts. This involves analyzing speech patterns, intonation, and subtle vocal cues to detect anomalies indicative of AI-generated speech. Additionally, liveness detection techniques are incorporated to ensure that voice samples originate from live individuals rather than recordings or simulations. Through the integration of advanced signal processing, machine learning models, and real-time verification mechanisms, the system strives to enhance security and reliability in authentication processes while addressing emerging threats posed by AI-generated speech.

Design Consideration

1. Generalizability - The system's ability to detect AI-generated voice prints across various voice domains, accents, and speaking styles.
2. Feature Engineering - Advanced feature techniques considered to make the model more adaptable, accurate and resilience against adversarial techniques. Features like **Spectral Rolloff, Spectral Bandwidth**, **Zero Crossing Rate**, **Mel-frequency cepstral coefficients (MFCC)** and **Chroma Frequencies** were considered while training the model.
3. Language and Accent Detection - Utilize advanced language models and Natural Language Processing (NLP) techniques to accurately identify the language spoken in the speech samples. Implement models trained on diverse language datasets to ensure robustness across various languages and accents.Incorporate accent detection algorithms to distinguish between different accents within the identified language. Train models on accent-specific datasets to improve accuracy in accent classification, considering regional variations and dialects.

4. Background Noise Level - Utilize **Root Mean Square RMS** analysis as a metric to quantify the level of background noise present in the speech signal. Apply threshold-based methods to classify noise levels and trigger appropriate noise reduction algorithms.

5. Emotional Level - Develop emotion recognition models using machine learning techniques such as deep neural networks or ensemble methods. Train models on emotion-labeled datasets to accurately detect and classify emotional states expressed in speech samples

6. Liveness Detection Mechanism - To Detect Audio Liveness, We plan to use **user location Data**. Location data can potentially help identify whether an audio recording is natural or artificial by cross-referencing the recorded audio with known locations of sound sources or ambient noises. For example, if the recorded audio contains sounds that are consistent with a specific location (such as street noises, bird chirping, etc.), and the location data confirms that the recording was made in that particular area, it can provide evidence that the audio is genuine.

7. Algorithm Selection and Optimization - To Choose the Best Algorithm, Multiple AI techniques were used like Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), Support Vector Machine and

8. Model Training and Evaluation - Train the AI models using labeled datasets and employ rigorous evaluation metrics such as accuracy, precision, recall, and F1-score to assess their performance.

9. Overfitting - Implement cross-validation techniques to validate the generalization capabilities of the models and mitigate overfitting issues.

Design Specification

Requirement	Description
R1	Develop AI algorithms capable of distinguishing between authentic and AI-generated voice prints
R2	Incorporate liveness detection techniques to ensure that voice samples are produced by a live person and not a recording or AI simulation
R3	Identify the language spoken in the speech
R4	Detect the accent in the Speech
R5	Process the Background Noise and its level
R6	Identify the emotional tone in the speech
R7	Identify if any speech is present in the audio
R8	API Endpoints (Flask)
R9	Deployment

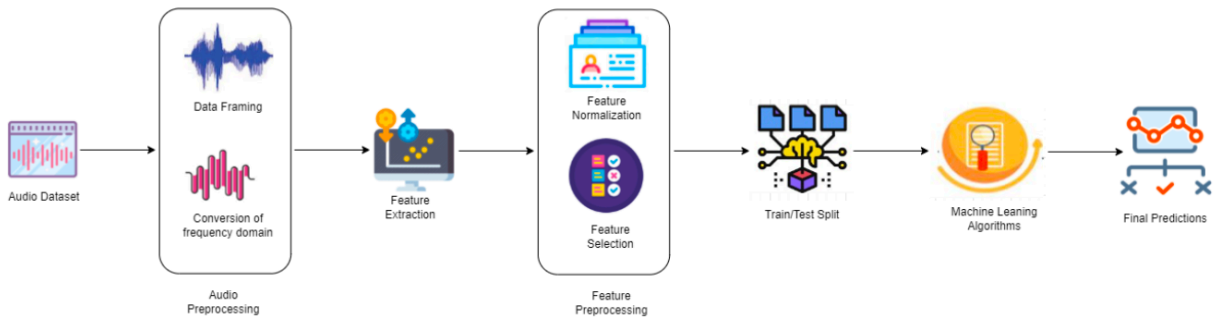
Note: ***Azure Speech Services used for R3, R4 and R7. Azure App Services used for R9***

Github Link - https://github.com/Hackathon2024-March/marinabeach

Detailed Design (R1)

1. Data Collection - Data collection for the problem was done in two parts-
   - Real Voices - We used CommonVoice Dataset to collect real Voices. We choose three languages - English, French and Spanish and for each language, we took care of different accent, gender and age.
   - AI Generated Voices - We used two ways to generate the AI voices -
     - Eleven Labs - We used to eleven labs to generate AI Audios with different gender, language and accent.
     - gTTS - We used 12000 AI Audio files using Google Text-to-Speech translation.
2. Audio Preprocessing - We remove the background noise, resample and trim the audio voice and remove the frame of audio where there is no speech.
3. Feature extraction - **Spectral Rolloff, Spectral Bandwidth**, **Zero Crossing Rate**, **Mel-frequency cepstral coefficients (MFCC)**, and **Chroma frequencies** using librosa library.
4. Train/Test Split - We used the conventional 0.2 split and applied K-Fold = 5 strategy to cross validate and train our model.
5. Machine Learning Algorithms - Convolutional Neural Network, Recurrent Neural Network, Support Vector Machine, Random Forest, Logistic Regression and XG Boosting were applied. Out of which on the basis of evaluation metrics, **Logistic Regression** performed best.
6. Evaluation - To Evaluate the Binary Classification problem, We chose Confusion matrix, roc arc score and accuracy as our base of criterion on test dataset.

Architectural Diagram



Liveness Detection (R2) - To Detect Audio Liveness, Location data can potentially help identify whether an audio recording is natural or artificial by cross-referencing the recorded audio with known locations of sound sources or ambient noises. For example, if the recorded audio contains sounds that are consistent with a specific location (such as street noises, bird chirping, etc.), and the

location data confirms that the recording was made in that particular area, it can provide evidence that the audio is genuine.

Language Detection, Accent Detection , Speech Detection (R3, R4, R7) - **Azure Speech Services** are used to identify the language and Accent.

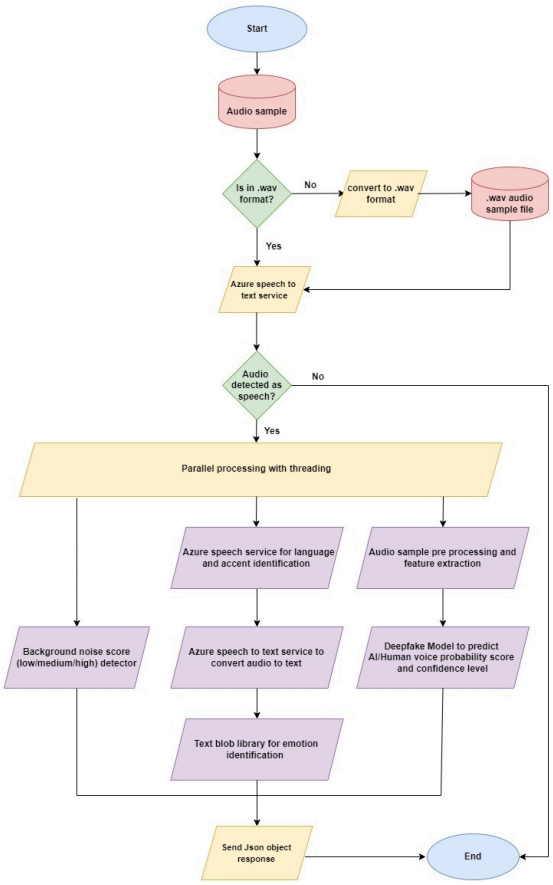
Justification for Azure Speech Service - Azure Speech Services offers accurate language and accent detection alongside robust speech recognition capabilities, leveraging Azure's scalability, security, and comprehensive language support for effective integration into diverse applications.

Background Noise (R5) - Using **Root Mean Square (RMS) of energy** analysis as a metric to quantify the level of background noise present in the speech signal. Apply threshold-based methods to classify noise levels and trigger appropriate noise reduction algorithms.

Sentimental Analysis (R6) - TextBlob (Open source) primarily provides polarity and subjectivity scores for sentiment analysis, Utilising the polarity metric, we map the emotion of the speech.

API Endpoint (R8) - Flask Framework is used to create the backend. Two Endpoints are made -

- **/ping [GET METHOD]-** This is used to check Server's Health
- **/voice/analyze [POST METHOD] -** This take audio file as an input and analyses the audio file and give the response. Check Below **API Flow Diagram -**



Deployment (R9) - Flask Server is deployed on Azure via Azure App Services.

Justification for Azure App Service - Deploying a Flask server on Azure App Services provides a scalable and managed platform for hosting Python web applications. It offers seamless integration with Azure's ecosystem, automatic scaling, and easy deployment workflows, ensuring efficient and reliable deployment of Flask-based applications.

Testing

Manual testing was done via Postman. Unit test cases and Coverage report are there to ensure the integrity of the system.