Design of a Voice Recognition System Using Artificial Neural Network

Mayowa O. Daniel[1,*], Ibukunoluwa A. Olajide[2]
[1]Department of Computer Engineering, The Federal University of Technology, Akure, Nigeria
Email: danielmayotar@gmail.com
[2]Department of Electrical and Electronics Engineering, The Federal University of Technology, Akure, Nigeria
Email: iaadebanjo@futa.edu.ng
[*] *Corresponding author*

Abstract: **Voice recognition systems have gained significant prevalence in our everyday lives, encompassing a wide range of applications, from virtual assistants on smartphones to voice-controlled home automation systems. This research paper presents a comprehensive design and implementation of a voice recognition security system employing artificial neural networks. The system's training involved a dataset consisting of 900 audio samples collected from 10 distinct speakers, enabling the resulting model to accurately classify the speaker of a given audio sample. For the implementation of the voice recognition system, Python serves as the primary programming language. The system leverages the Keras library, which offers a high-level interface for constructing and training neural networks, with efficient computation facilitated by the TensorFlow back-end. Additionally, the Flask framework, a Python-based web framework, was utilized to create a user interface in the form of a web application for the voice recognition system. To effectively train the artificial neural network, the audio data undergoes preprocessing, involving the extraction of relevant features from the audio samples. Subsequently, during the preprocessing phase, the audio data is labelled, and the neural network is trained on this labelled dataset to learn the classification of different speakers. The trained model was rigorously tested on a set of previously unseen audio samples, yielding an impressive classification accuracy exceeding 96%. The finalized model will be integrated into the web application, enabling users to upload audio files and receive accurate predictions regarding the speaker's identity. This paper demonstrates the efficacy of artificial neural networks in the context of voice recognition systems, while also providing a practical framework for constructing such systems using readily available tools and libraries.**

Keywords: **ANN, Kera Libraries, Neural Networks, Security, Voice recognition**.

I. INTRODUCTION

Regarding interpersonal communication, speech stands out as the predominant, inherent, and highly efficient means. The process of automatic speech recognition (ASR) entails transforming an acoustic signal, captured via a microphone or telephone, into a sequence of words through the utilization of a specific algorithm that can be implemented as a system program [1]. This technique also encompasses the interpretation of spoken words by analyzing voice recordings to ascertain their intended meaning. Contemporary residential and commercial security measures commonly encompass diverse protective methods, such as passwords, user IDs, and PINs. Nonetheless, these systems lack robustness due to the susceptibility of PIN codes to hacking and the potential theft and replication of ID cards [2]. Consequently, the emergence of a novel technology called biometrics is expected to engender enhanced trust in security systems. Biometrics comprises various techniques employed for identifying individuals based on their distinctive physical and behavioural traits. Examples of such identifying features include fingerprints, voice patterns, facial characteristics, retinal and iris scans, signatures, hand geometry, and wrist veins [2].

Biometric technology operates by employing a user's unique physical attribute as the password or feature parameter. Human characteristics such as voice, face, and fingerprints are commonly utilized as feature parameters. The inherent absence of identical twins ensures user confidence and safety when employing voice recognition systems [2]. According to a survey conducted by Unisys, 32% of respondents favored voice recognition, 27% preferred fingerprints, 20% opted for facial scans, 12% favored hand geometries, and 10% expressed a preference for iris scans [3]. Given that the human voice embodies the most pervasive and instinctive form of human communication, voice recognition stands as a leading biometric technology. Consequently, voice recognition systems hold potential benefits for securing doors, vaults, confidential laboratories, and other restricted areas.

II. STATEMENT OF PROBLEM

Ensuring security is a paramount concern for both individuals and organizations, and with the continuous advancement of technology, various approaches have been employed to safeguard lives and assets. Traditional methods of protection, such as PINs and passwords, have long been relied upon as the foundation for securing properties. However, these conventional security measures are vulnerable to hacking and unauthorized access due to their inherent limitations and lack of robustness [4]. Furthermore, commonly used biometric technologies themselves are not immune to shortcomings. For instance, in the case of fingerprint recognition, the security of the system can be compromised if a user's finger is forcibly removed [5]. Similarly, facial recognition systems can be deceived by utilizing a user's photograph to bypass security measures [6]. Moreover, these methods often prove inadequate and incompatible for individuals with physical disabilities, posing additional challenges [7].

III. LITERATURE REVIEW

A. Speech Recognition Systems

Drenthen (2012) introduced a fundamental speech recognition system that encompasses several distinct stages, namely pre-processing, feature extraction, clustering, and classification [8]. In the pre-processing module, aimed at input speech signals, enhancing the signal-to-noise ratio is crucial. Subsequently, in the second step, relevant features of the signal are extracted through an appropriate technique for feature extraction. The third step involves determining the centroid by employing the k-means algorithm on the feature vectors. Lastly, a pattern-matching technique is utilized to recognize the speech signal, with the matching score dependent on both the chosen algorithm and the size of the training database [9].

B. Evolution of Speech Recognition Systems

The history of speech recognition technology dates back to significant milestones. In 1784, a scholar in Vienna created the first Acoustic Mechanical Speech Machine. Later, in 1879, Thomas Edison invented the first dictation machine. Advancing further, Bell Laboratories developed a speech recognition system in 1952 capable of accurately recognizing spoken digits, albeit limited to the inventor's voice. Notably, in 1970, a scholar introduced the Harpy System, which exhibited impressive capabilities, recognizing over 1000 words, different pronunciations, and certain phrases. The 1980s witnessed further advancements with the introduction of the Hidden Markov Model (HMM) in speech recognition. This mathematical approach revolutionized the analysis of sound waves and paved the way for numerous breakthroughs. IBM Tangora, in 1986, harnessed the power of HMM and successfully predicted upcoming phonemes in speech. In 2006, the National Security Agency (NSA) adopted speech recognition systems to segment keywords in recorded speech, highlighting its relevance in security applications.

The subsequent surge in speech recognition technology occurred when leading IT companies such as Facebook, Google, Amazon, Microsoft, and Apple ventured into offering this functionality across various devices. Services like Google Home, Amazon Echo, and Apple Siri exemplify their commitment to developing voice assistants that provide accurate responses and replies. These top tech companies strive to enhance the accuracy and efficiency of voice assistants, driving the continuous evolution of speech recognition technology.

IV. METHODOLOGY

The purpose of the speaker recognition system is to identify individuals based on their unique voice patterns. The methodology involves extracting various features from the audio signal, such as prosodic and spectral features. These features are then used to develop a speaker voice model using machine learning algorithms. The speaker recognition system compares the input audio to a speaker database in order to identify the specific speaker. Fig. 1 provides a visual representation of the step-by-step process to achieve the desired outcome. The implementation stages include data acquisition, pre-processing, feature extraction, training and evaluation, and classification of the voice input.

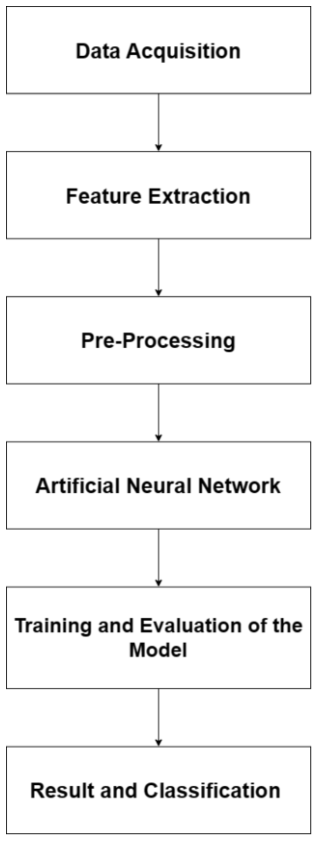
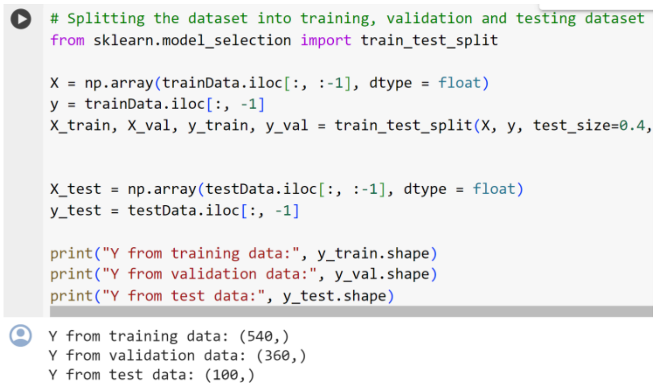

Data Acquisition
Feature Extraction
Pre-Processing
Artificial Neural Network
Training and Evaluation of the Model
Result and Classification

Fig. 1. The block diagram of the Voice Recognition System

A. Data Acquisition

The data collection process involved obtaining audio recordings from ten different users using a mobile phone as the recording device. A total of 1000 audio samples were collected, consisting of number-digits ranging from 0 to 9. The dataset was in waveform (.wav) format. To facilitate model training and evaluation, the data was split into training and testing subsets. The testing subset comprised 100 audio samples, while the training dataset contained 900 audio samples. Within the training dataset, a further split was made to create a validation set, with a ratio of 60:40 for training and validation respectively. This split ensured an appropriate distribution of data for model development and evaluation. Fig. 2 illustrates the allocation of the pre-processed data into the respective subsets, maintaining the appropriate proportions. Each of these split data groups—training, testing, and validation was utilized during the modelling stage for training, testing, and validation purposes respectively.



```
# Splitting the dataset into training, validation and testing dataset
from sklearn.model_selection import train_test_split

X = np.array(trainData.iloc[:, :-1], dtype = float)
y = trainData.iloc[:, -1]
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.4,


X_test = np.array(testData.iloc[:, :-1], dtype = float)
y_test = testData.iloc[:, -1]

print("Y from training data:", y_train.shape)
print("Y from validation data:", y_val.shape)
print("Y from test data:", y_test.shape)
```

```
Y from training data: (540,)
Y from validation data: (360,)
Y from test data: (100,)
```

Fig. 2. Splitting of the dataset

B. *Feature Extraction*

Various techniques can be employed to extract unique features from the audio dataset, including Linear Prediction Coefficients (LPC), Linear Prediction Cepstral Coefficients (LPCC), Mel Frequency Cepstral Coefficients (MFCC), Discrete Wavelet Transform (DWT), Perceptual Linear Prediction (PLP), and more. In the case of the dataset at hand, the technique chosen for feature extraction is Mel Frequency Cepstral Coefficients (MFCC). When extracting features from audio data, a range of characteristics can be considered, such as zero-crossing rate, energy, spectral roll-off, spectral flux, spectral entropy, chroma features, pitch, MFCC, spectral bandwidth, spectral centroid, and so on. These features provide valuable information about the properties and patterns within the audio signals. In the specific case of the dataset being analyzed, the MFCC feature extraction technique is employed, which focuses on capturing the Mel-frequency spectrum and cepstral coefficients of the audio signals.

C. *Pre-Processing*

Pre-processing plays a crucial role in the analysis of audio data, particularly when working with extracted features. Its main objective is to improve the quality of audio signals, reduce noise, and extract relevant information for subsequent analysis. In this particular scenario, pre-processing involves working with extracted features obtained from voice data samples stored in a comma-separated value (.csv) file. During the pre-processing, these techniques were employed to enhance audio signal quality and remove unwanted noise-filtering, normalization, denoising, and resampling.

D. *Explanatory Data Analysis and Correlation*

Exploratory data analysis (EDA) is an approach used to understand the information and patterns within a dataset, without necessarily focusing on formal modeling or hypothesis testing. It involves examining the data to gain insights, identify relationships, and uncover potential correlations among the extracted features. In the context of the current analysis, EDA was conducted to assess the correlation between the extracted features from the dataset. To visualize and analyze this correlation, a heatmap was employed. The heatmap, presented in Fig. 3 and Fig. 4, provides a graphical representation of the relationships between different features and allows for a quick assessment of their strength and direction.

By examining the heatmap, patterns and dependencies between the features can be identified. High correlations between features indicate a strong relationship, while low correlations suggest a weaker or no relationship.

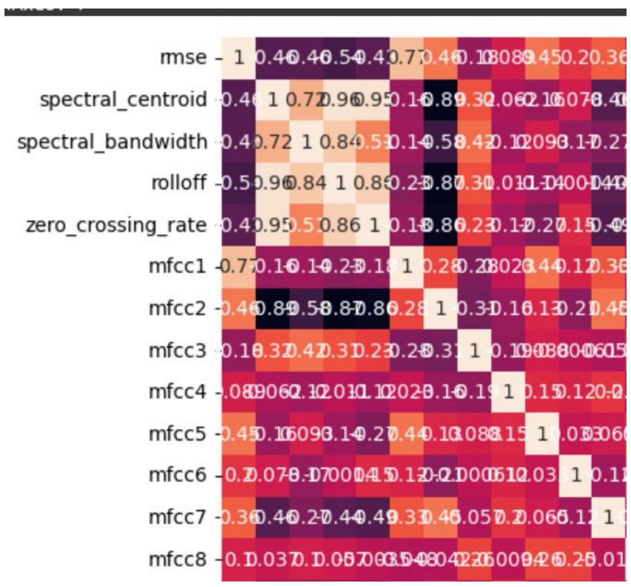


Fig. 3. Data correlation of the Audio feature using Heatmap A

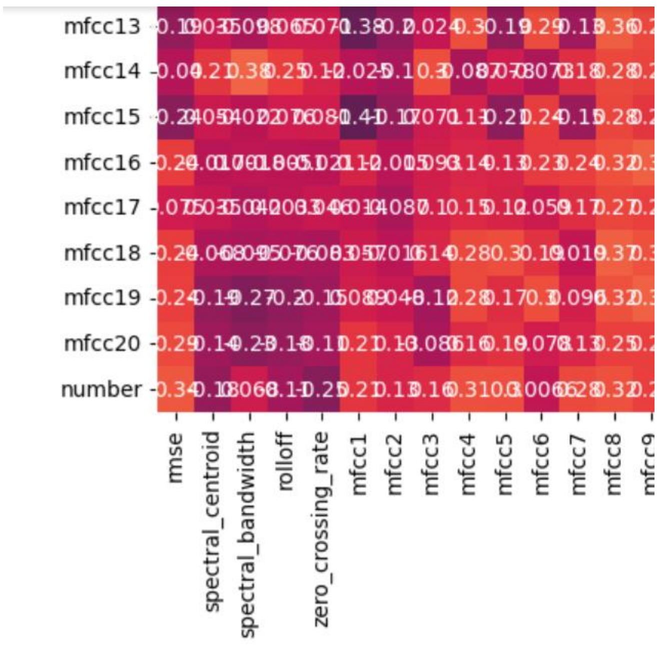


Fig. 4. Data correlation of the Audio feature using Heatmap B

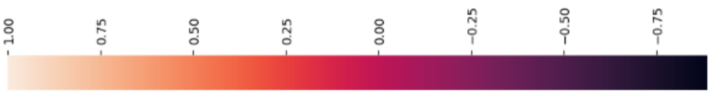


Fig. 5. Data correlation range

A heatmap is a visualization tool that employs a color-coded matrix to represent data. In the context of this analysis, the heatmap is used to depict the relationship between various features extracted from the audio data. Fig. 5 shows the correlation coefficient, ranging from -1 to 1, measures the strength and direction of the relationship between two features. A correlation coefficient of 1 indicates a perfect positive correlation, typically occurring when a feature is compared to itself. It signifies that as one variable increases, the other variable also increases at the same rate. On the other hand, a correlation coefficient of -1 indicates a complete negative correlation, where an increase in one variable corresponds to a decrease in the other variable at the same rate. If the correlation coefficient is 0, it implies the absence of any correlation between the two features. When observing the heatmap, cells with correlation coefficients closer to 1 indicate a stronger positive influence between those features in the model training process. This implies that variations in one feature are strongly associated with variations in the other, making them important for modeling and analysis purposes.

E. Modelling

The voice recognition model was constructed using a deep neural network architecture with the Keras library. Keras is a popular high-level deep learning framework that provides a user-friendly interface to build and train neural networks. The model architecture consisted of multiple layers of densely connected nodes, also known as fully connected layers. These layers allow information to flow in both forward and backward directions, enabling the network to learn complex patterns and relationships in the data. Each node in these layers is connected to every node in the previous and subsequent layers. The activation function used in the densely connected layers was the rectified linear unit (ReLU). ReLU is a widely used activation function that introduces non-linearity to the model. It transforms negative input values to zero and leaves positive values unchanged, enabling the network to learn more complex representations.

For the final classification step, a softmax activation function was employed. Softmax is commonly used in multi-class classification problems as it provides a probability distribution over the classes. It assigns probabilities to each class, indicating the likelihood of the input belonging to each class. By utilizing this deep neural network architecture with ReLU activation functions in the densely connected layers and a softmax activation function for classification, the model can effectively learn and classify voice patterns for speaker recognition tasks.

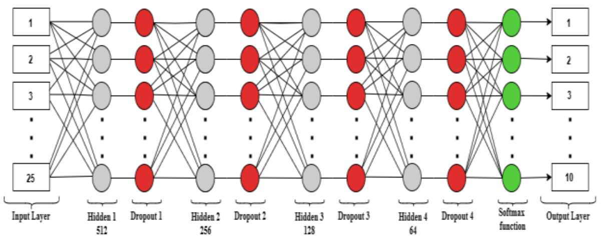


Fig. 6. The ANN Architecture

The artificial neural network (ANN) depicted in Fig. 6 consists of hidden layers that perform weighted computations on the received input and pass the results to the subsequent layers until reaching the output layer. Each layer in the neural network contains nodes, also known as neurons, which perform computations on the input data using weights and activation functions. The hidden layers of the neural network conduct a set of weighted computations on the input data they receive. These computations involve multiplying the input values by corresponding weights and summing them to obtain an intermediate result. This result is then passed through an activation function, which introduces non-linearity to the network and enables it to learn complex patterns.

Dropout layers are incorporated in the neural network architecture to address the issue of over-fitting during training. Over-fitting occurs when the model becomes too specialized to the training data and fails to generalize well to unseen data. Dropout layers randomly deactivate a fraction of the neurons during training, forcing the network to learn more robust and generalizable representations by preventing co-adaptation among neurons. The output layer of the neural network employs the softmax function. The softmax function takes the raw outputs from the preceding layers and converts them into probabilities for each class. These probabilities represent the likelihood of the input belonging to each class. The input is then classified based on the class with the highest probability, providing the final output of the network.

By utilizing this neural network architecture with hidden layers, dropout layers for regularization, and the softmax function in the output layer, the model can effectively learn and classify data into different classes, taking raw input and producing probabilistic predictions.

Artificial neural network (ANN) was used because of its effectiveness in processing and classifying complex audio data [10]. The benefits of using ANN include:

1. Pattern Recognition: ANNs are well-suited for recognizing and learning patterns in data, making them ideal for voice recognition tasks where subtle variations in speech need to be detected and classified [10].

2. Non-linearity: The use of hidden layers allows the network to capture non-linear relationships in the data, which is crucial for accurately modelling the complex nature of human speech [10].

3. Generalization: By utilizing dropout layers, the network is less prone to over fitting, meaning it can better generalize to unseen data and perform reliably in real-world scenarios.

However, the major drawback of ANN is that it tends to over train [11], which requires you to tune some parameters such as learning rate, dropout rate, etc. to achieve optimal performance.

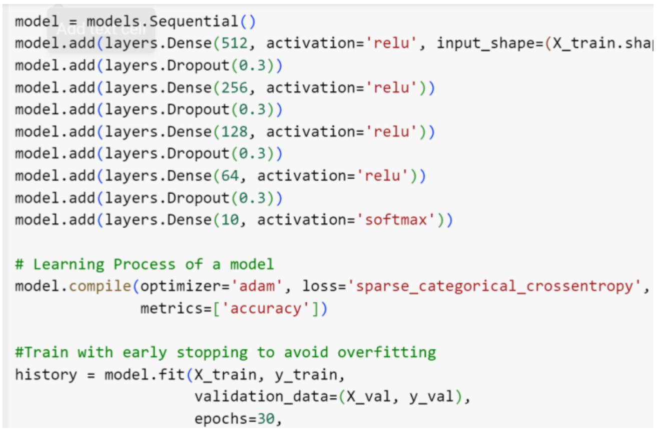
The application of artificial neural network (ANN) in voice recognition security systems is used to secure vaults, server rooms, for smart homes or smart hospital [12].

F. Training

The training process of the neural network model was conducted using the stochastic gradient descent (SGD) optimizer. SGD is a widely used optimization algorithm in deep learning. It updates the model's weights iteratively based on the gradients computed on small batches of training data. The specific configuration of the SGD optimizer used in this training process includes a learning rate of 0.01, which determines the step size for weight updates. A higher learning rate allows for larger weight updates, potentially leading to faster convergence, but it can also risk overshooting the optimal solution. Conversely, a lower learning rate reduces the risk of overshooting but may slow down the convergence process. The momentum value of 0.9 was employed in the SGD optimizer. Momentum helps accelerate the training process by accumulating the gradients from previous steps and adding a fraction of it to the current gradient update. This allows the optimizer to navigate through flat areas or shallow local minima more efficiently.

The sparse categorical cross-entropy loss function was chosen for this training process. This loss function is commonly used when dealing with multi-class classification problems. It computes the cross-entropy loss between the predicted probabilities and the true class labels. The "sparse" aspect indicates that the true class labels are provided as integers instead of one-hot encoded vectors. The training process was executed over 30 epochs. An epoch represents a complete iteration over the entire training dataset. Training over multiple epochs allows the model to gradually refine its weights and improve its performance. A batch size of 20 was utilized during training. The batch size determines the number of samples processed before updating the model's weights. A smaller batch size introduces more frequent weight updates but can result in noisy gradients. Conversely, a larger batch size reduces the frequency of weight updates but provides a more stable estimation of the gradients.

Fig. 7 likely represents a visualization or plot showcasing the training progress, such as the training loss or accuracy, over the 30 epochs. This visual representation aids in understanding the model's convergence and performance throughout the training process.



```
model = models.Sequential()
model.add(layers.Dense(512, activation='relu', input_shape=(X_train.sha
model.add(layers.Dropout(0.3))
model.add(layers.Dense(256, activation='relu'))
model.add(layers.Dropout(0.3))
model.add(layers.Dense(128, activation='relu'))
model.add(layers.Dropout(0.3))
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dropout(0.3))
model.add(layers.Dense(10, activation='softmax'))

# Learning Process of a model
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

#Train with early stopping to avoid overfitting
history = model.fit(X_train, y_train,
                    validation_data=(X_val, y_val),
                    epochs=30,
```

Fig. 7. Training of the Model

G. Testing

The purpose of the testing process was to assess the model's generalization capabilities and its accuracy in predicting outcomes using unseen data. This evaluation involved utilizing a fresh set of voice recordings, which underwent the same preprocessing steps as the training dataset. Subsequently, the model's performance was evaluated based on its ability to accurately classify the new recordings into their respective classes.

Fig. 8 illustrates the difference in epoch loss between the training data and the test data. Towards the later portion of the graph, the test and train lines closely converge, indicating that the model is effectively generalizing to the data used in the testing process. This convergence suggests that the model is successfully adapting to new, unseen data and can make accurate predictions. This alignment between the test and train lines underscores the model's robustness and its potential for real-world applications.

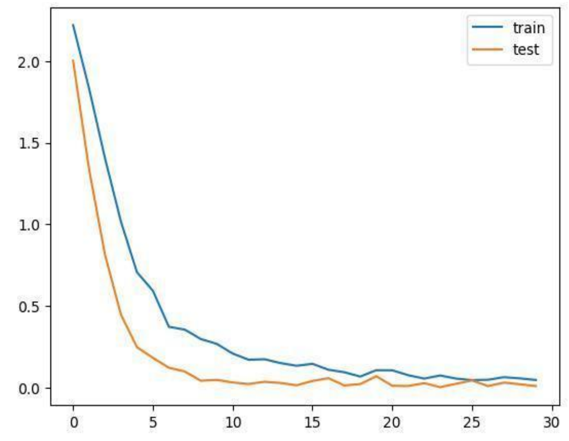
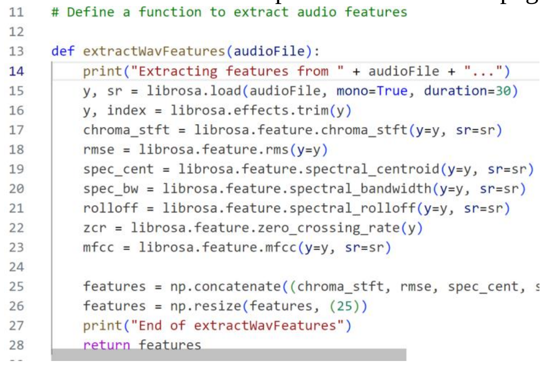


Fig. 8. Graph showing the epoch losses between the train data and test data

H. Web Application

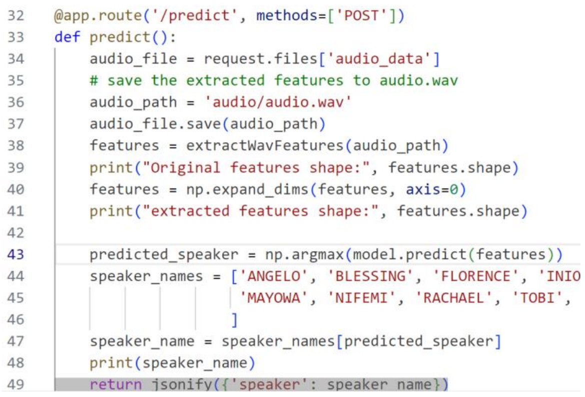
The web application was created utilizing Flask, a Python-based lightweight web framework. The application incorporated a web form that enabled users to upload an audio recording of their voice and submit it for prediction. The uploaded audio recording underwent preprocessing using the same techniques employed during the training phase. Subsequently, the preprocessed audio was passed into the saved voice recognition model for prediction. The predicted speaker name was then displayed on the web page.

To construct the web interface, HTML, CSS, and JavaScript were utilized for designing and implementing the web-page. The Python Flask API was employed to run the server, enabling communication between the web application and the back-end voice recognition model. This combination of technologies facilitated the seamless integration of user interaction, data processing, and prediction generation in the web application. Fig. 9 and 10 gives the feature extraction and prediction code flow incorporated into the web page.



```
# Define a function to extract audio features

def extractWavFeatures(audioFile):
    print("Extracting features from " + audioFile + "...")
    y, sr = librosa.load(audioFile, mono=True, duration=30)
    y, index = librosa.effects.trim(y)
    chroma_stft = librosa.feature.chroma_stft(y=y, sr=sr)
    rmse = librosa.feature.rms(y=y)
    spec_cent = librosa.feature.spectral_centroid(y=y, sr=sr)
    spec_bw = librosa.feature.spectral_bandwidth(y=y, sr=sr)
    rolloff = librosa.feature.spectral_rolloff(y=y, sr=sr)
    zcr = librosa.feature.zero_crossing_rate(y)
    mfcc = librosa.feature.mfcc(y=y, sr=sr)

    features = np.concatenate((chroma_stft, rmse, spec_cent, s
    features = np.resize(features, (25))
    print("End of extractWavFeatures")
    return features
```

Fig. 9. Feature extraction code on the web



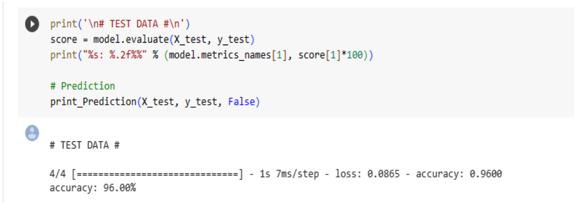
```
@app.route('/predict', methods=['POST'])
def predict():
    audio_file = request.files['audio_data']
    # save the extracted features to audio.wav
    audio_path = 'audio/audio.wav'
    audio_file.save(audio_path)
    features = extractWavFeatures(audio_path)
    print("Original features shape:", features.shape)
    features = np.expand_dims(features, axis=0)
    print("extracted features shape:", features.shape)

    predicted_speaker = np.argmax(model.predict(features))
    speaker_names = ['ANGELO', 'BLESSING', 'FLORENCE', 'INIO
                     'MAYOWA', 'NIFEMI', 'RACHAEL', 'TOBI',
                    ]
    speaker_name = speaker_names[predicted_speaker]
    print(speaker_name)
    return jsonify({'speaker': speaker_name})
```

Fig. 10. Prediction code

V. RESULT AND DISCUSSION

The performance of the voice recognition system was assessed using a dedicated test dataset comprising 100 audio files. This dataset contained 10 audio files for each of the 10 speakers present in the training dataset. The evaluation of the system on this test dataset yielded an accuracy of 96%, as depicted in Fig. 11. Out of the 100 audio files, only 3 were misclassified by the system. Fig. 12 provides a visualization of the classification report generated from the evaluation of the test data on the model. The classification report offers detailed insights into the performance of the model for each class in the test dataset. It typically includes metrics such as precision, recall, F1-score, and support, which collectively provide a comprehensive assessment of the model's performance on each class.

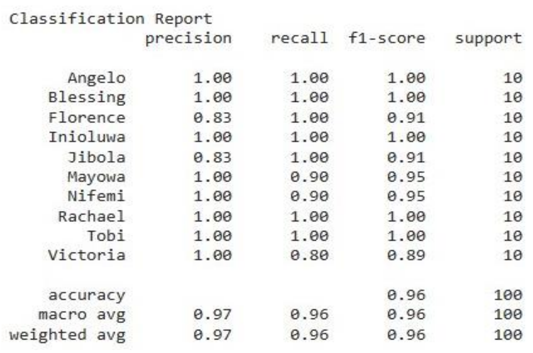
The reported accuracy of 96% and the low number of misclassified audio files indicate that the voice recognition system achieved a high level of accuracy and effectiveness in accurately identifying and classifying speakers in the test dataset. These results suggest that the model trained on the training dataset generalized well to unseen data and demonstrates its potential for real-world voice recognition applications.



```
print('\n# TEST DATA #\n')
score = model.evaluate(X_test, y_test)
print("%s: %.2f%%" % (model.metrics_names[1], score[1]*100))

# Prediction
print_Prediction(X_test, y_test, False)
```

```
# TEST DATA #

4/4 [==============================] - 1s 7ms/step - loss: 0.0865 - accuracy: 0.9600
accuracy: 96.00%
```

Fig. 11. Accuracy of the model



```
Classification Report
                precision    recall  f1-score   support

      Angelo       1.00      1.00      1.00        10
    Blessing       1.00      1.00      1.00        10
    Florence       0.83      1.00      0.91        10
    Inioluwa       1.00      1.00      1.00        10
      Jibola       0.83      1.00      0.91        10
      Mayowa       1.00      0.90      0.95        10
      Nifemi       1.00      0.90      0.95        10
     Rachael       1.00      1.00      1.00        10
        Tobi       1.00      1.00      1.00        10
    Victoria       1.00      0.80      0.89        10

    accuracy                           0.96       100
   macro avg       0.97      0.96      0.96       100
weighted avg       0.97      0.96      0.96       100
```

Fig. 12. Classification Report

A. Training Analysis

Fig. 13 presents the confusion matrix corresponding to the classification results depicted in Figure 12, specifically highlighting the classification outcomes for each speaker. In the confusion matrix, the diagonal elements signify the number of correctly classified audio files for each speaker, while the off-diagonal elements represent the number of misclassified audio files. The matrix reveals that the majority of misclassifications occurred between speakers who possessed similar accents or shared voice characteristics. This observation aligns with the challenges commonly encountered in voice recognition tasks, where distinguishing between speakers with similar traits can be more intricate. The occurrence of such misclassifications is not unexpected, given the inherent complexities associated with voice recognition systems. The confusion matrix provides valuable insights into the specific patterns of misclassifications and the relationships between speakers. By analyzing these patterns, further refinements or adjustments can be made to improve the model's performance, such as incorporating additional training data from speakers with similar accents or employing advanced techniques for accent or voice characteristic normalization.

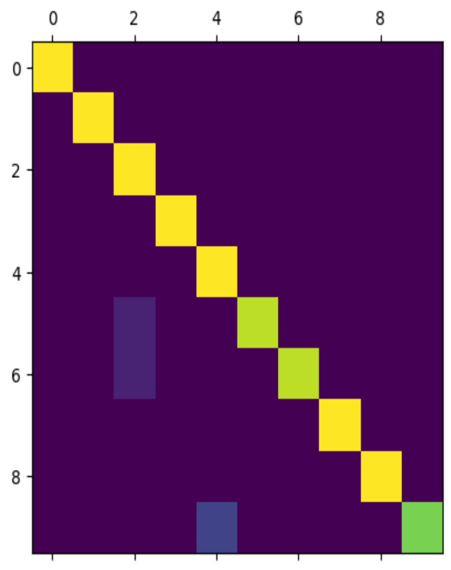


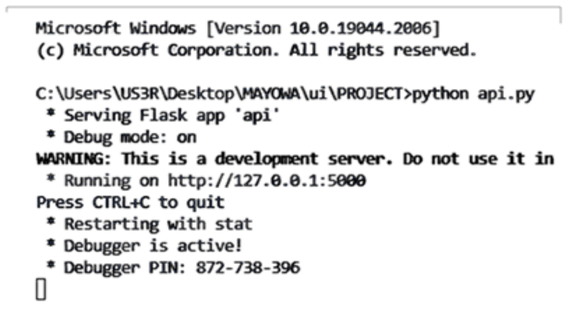
Fig. 13. Confusion Matrix Analysis of the Classified Output

B. Web application result

The web application utilizes a pre-trained speaker recognition model to make predictions based on audio files. The model is developed using the Keras package, while the application itself is implemented using the Python Flask API.

The web application consists of two main components: the front-end and the back-end. The front-end is responsible for creating the user interface, allowing users to upload an audio file for prediction. It is developed using HTML, CSS, and JavaScript, providing a visually appealing and interactive interface for users. On the other hand, the back-end is built using the Flask API. When a user uploads an audio file through the front-end, the back-end receives the file, extracts the relevant audio features, and utilizes the pre-trained model to identify the speaker. The back-end processes the audio data and produces the prediction results. Fig. 14 displays the output details on the terminal, indicating whether the code has run successfully or encountered an error. In the case of a successful run, the terminal output includes the IP address, indicating that the program executed without issues. To access the web application, the IP address from the terminal output needs to be copied and pasted into a web browser. This action loads the web-page, as illustrated in Fig. 15, providing the user interface where audio input can be provided.

To test the program, an audio file is required to be uploaded through the web-page. After uploading the audio file, the user clicks the "predict" button. The back-end processes the audio data using the pre-trained model and generates a visual output on the screen, displaying the predicted speaker's name, as depicted in Fig. 16. These steps ensure a seamless user experience, from uploading the audio file to obtaining the predicted speaker's name through the web application's front-end and back-end integration.



```
Microsoft Windows [Version 10.0.19044.2006]
(c) Microsoft Corporation. All rights reserved.

C:\Users\US3R\Desktop\MAYOWA\ui\PROJECT>python api.py
 * Serving Flask app 'api'
 * Debug mode: on
WARNING: This is a development server. Do not use it in
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 872-738-396
```

Fig. 14. Starting the API for the web server

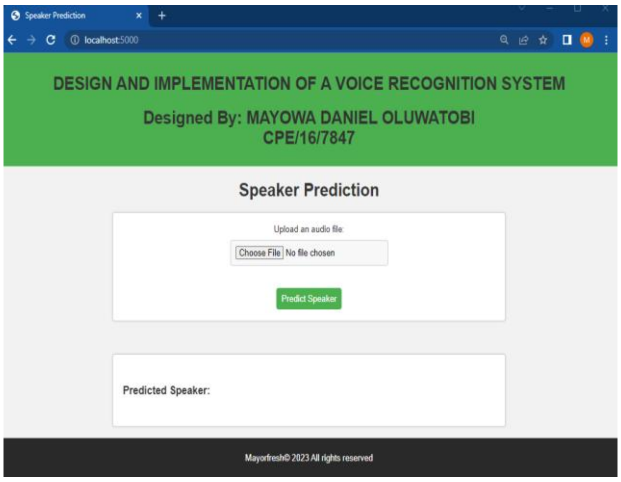
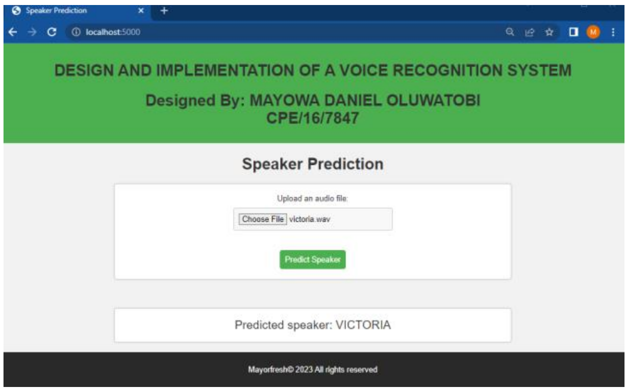

Speaker Prediction
localhost:5000
DESIGN AND IMPLEMENTATION OF A VOICE RECOGNITION SYSTEM
Designed By: MAYOWA DANIEL OLUWATOBI
CPE/16/7847
Speaker Prediction
Upload an audio file:
Choose File No file chosen
Predict Speaker
Predicted Speaker:
Mayorfresh© 2023 All rights reserved

Fig. 15. Webpage View


Speaker Prediction
localhost:5000
DESIGN AND IMPLEMENTATION OF A VOICE RECOGNITION SYSTEM
Designed By: MAYOWA DANIEL OLUWATOBI
CPE/16/7847
Speaker Prediction
Upload an audio file:
Choose File victoria.wav
Predict Speaker
Mayorfresh© 2023 All rights reserved

Predicted speaker: VICTORIA

Fig. 16. Web page showing the predicted speaker

VI. CONCLUSIONS

In conclusion, this paper has successfully presented an implemented system for predicting the speaker of a speech. Through a comprehensive review of relevant literature and a thorough comparison of various machine learning methods, it was determined that an artificial neural network would be the most suitable approach for realizing the system. The results obtained from the implemented system have demonstrated the potential of artificial neural networks in developing robust and reliable voice recognition systems. The system's accuracy and performance showcase its viability for deployment in diverse domains, including security, authentication, and communication. By achieving its goals and objectives, this paper has contributed to the advancement of voice recognition technology. The successful implementation of the system highlights the effectiveness of artificial neural networks in accurately identifying speakers and opens up new possibilities for their application in real-world scenarios. The findings of this work encourage further exploration and refinement of artificial neural network-based voice recognition systems. Future research can focus on enhancing the system's capabilities, such as handling diverse accents, improving accuracy, and addressing challenges related to speaker verification and identification.

Overall, this paper highlights the potential and significance of artificial neural networks in the development of robust and reliable voice recognition systems. The presented work contributes to the existing body of knowledge in the field and paves the way for future advancements in voice recognition technology.

REFERENCES

[1] T. Gulzar, A. Singh, D. K. Rajoriya and N. Farooq, "A Systematic Analysis of Automatic Speech Recognition: An Overview," *International Journal of Current Engineering and Technology*, vol. 4, no. 3, pp. 1664-1675, 2014.

[2] H.N. Mohd. Shah, M. Z. Ab Rashid, M.F. Abdollah, M.N. Kamarudin, C.K. Lin and Z. Kamis, "Biometric Voice Recognition in Security System," *Indian Journal of Science and Technology*, vol. 7, no. 2, pp. 104-112, 2014.

[3] A. Olubukola, A. Adeoluwa, O. Abraham, B. Oyetunde and O. Ayorinde, "Voice Recognition Door Access Control System," *IOSR Journal of Computer Engineering (IOSR-JCE)*, vol. 21, no. 5, pp. 1-12, 2019.

[4] Cypress, Data Defense, "6 Password Security Risks and How to Avoid Them," June 2020. [Online].Available: https://theconversation.com/passwords-security-vulnerability-constraints-93164.

[5] University of York, "Researchers expose vulnerabilities of password managers," 16 March 2020. [Online]. Available: https://www.york.ac.uk/news-and-events/news/2020/research/expose-vulnerabilities-password-managers/.

[6] P. Neil, "PIN Authentication Passkeys – Say Goodbye to Passwords," 25 April 2023. [Online]. Available: https://vaultvision.com/blog/pin-authentication-passkeys

[7] L.R. Rabiner and B.H. Juang, "Speech recognition: Statistical methods," in K. Brown (Ed.), *Encyclopedia of Language & Linguistics*, pp. 1-18, Amsterdam: Elsevier, 2006.

[8] H.F. Pai and H.C. Wang, "A two-dimensional cepstrum approach for the recognition of mandarin syllable initials," *Pattern Recognition*, vol. 26, no. 4, pp. 569-577, 1993.

[9] S. Furui, "History and development of speech recognition," In *Speech Technology*, pp. 1-18, New York: Springer, 2010.

[10] Y. LeCun, Y. Bengio and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, 2015.

[11] M. Malik, M.K. Malik, M.K., K. Mehmood and I. Makhdoom, "Automatic speech recognition: a survey," *Multimedia Tools and Applications*, vol. 80, pp. 9411-9457, 2021.

[12] A. Ismail, S. Abdlerazek and I.M. El-Henawy, "Development of Smart Healthcare System Based on Speech Recognition Using Support Vector Machine and Dynamic Time Warping," *Sustainability*, vol. 12, pp. 2403, 2020.