VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

SPEAKER RECOGNITION SYSTEM USING GAUSSIAN MIXTURE MODEL & DEEP LEARNING

Reg No	:	22BEC1268 & 22BEC1496
Course Title	:	Digital Signal Processing
Course Code	:	BECE301L
Faculty	:	Dr. Mohanaprasad K

ABSTRACT

This report gives a tutorial review about the intriguing realm of speaker recognition systems, delineating between speaker identification and verification, alongside their text-independent and text-dependent variants. It highlights the evolution from Gaussian Mixture Model Hidden Markov Model (GMM-HMM) to Deep Neural Network (DNN) approaches, showcasing the transformative impact of deep learning on speaker verification accuracy and robustness. The paper aims to provide a practical demonstration of both GMM and DNN-based speaker verification systems, detailing their design, development, and deployment processes. Through clear elucidation of implementation methodologies, it offers valuable insights for researchers and practitioners keen on delving into the technical intricacies and operational considerations of speaker recognition technology. This report focuses on bridging theoretical understanding with practical application, thereby serving as an informative resource for those navigating the landscape of speaker verification systems.

CONTENTS

SPEAKER RECOGNITION SYSTEM USING GAUSSIAN MIXTURE MODEL & DEEP LEARNING

1. INTRODUCTION

Speaker recognition systems are a fascinating application of technology that focuses on identifying and verifying the identity of individuals based on their unique vocal characteristics. Just like fingerprints or facial features, each person possesses distinct vocal traits, including pitch, tone, cadence, and even subtle nuances in pronunciation. Speaker recognition technology leverages advanced algorithms to analyze these features and create a unique vocal profile for each individual.

1.1 *Classification of Speaker Verification*

There are two main categories of speaker recognition systems: speaker identification and speaker verification.

- **Speaker Identification**: In speaker identification, the system is tasked with determining the identity of an unknown speaker by comparing their voice sample against a database of known speakers. This is often used in forensic investigations, security systems, and law enforcement.
- **Speaker Verification**: Speaker verification, on the other hand, focuses on confirming whether a person is who they claim to be based on their voice. This typically involves comparing a provided voice sample with a previously recorded sample from the same individual. Speaker verification is commonly used in authentication systems for access control, telephone banking, and other security-sensitive applications.

Speaker recognition can be further divided into Text Independent or Text dependent speaker recognition systems. Text-independent and Text-dependent speaker recognition systems are two different approaches to identifying or verifying individuals based on their voice, each with its own characteristics and applications.

- **Text-Independent Speaker Recognition**:
    - In a text-independent system, the speaker's identity is determined without requiring the speaker to utter specific phrases or sentences.

The system analyzes the overall characteristics of the speaker's voice, such as pitch, tone, and cadence, from a spontaneous speech sample.
    - These systems are more flexible and can work with any speech utterance, making them suitable for applications where users cannot be prompted to speak specific phrases, such as in forensic analysis or large-scale voice databases.
    - Text-independent systems are generally used for speaker identification tasks where the goal is to match an unknown speaker against a database of known speakers.
- **Text-Dependent Speaker Recognition**:
    - In contrast, text-dependent systems require the speaker to utter specific predetermined phrases or passphrases during enrollment and verification processes. These phrases are often referred to as "utterance prompts" or "passphrases."
    - During enrollment, the system captures the speaker's voice as they recite the predetermined phrases. Subsequently, during verification or identification, the system compares the speaker's voice against the stored voiceprints associated with those specific phrases.
    - Text-dependent systems are typically more accurate and reliable than text-independent systems because they rely on consistent speech patterns associated with specific phrases, reducing the likelihood of false positives.
    - These systems are commonly used in access control, authentication, and security applications where users can be prompted to speak specific phrases, such as in telephone banking systems or secure facilities.

1.2 *Evolution of Speaker Verification Methods*

The evolution of speaker verification methods illustrates a transition from conventional Gaussian Mixture Model (GMM) Hidden Markov Model (HMM) systems to the widespread adoption of Deep Neural Networks (DNNs). Initially dominant in the 1990s, GMM-HMM models faced limitations in capturing complex speech variations. In the early 2000s, the emergence of deep learning offered a solution, with DNNs showing promise in automatically learning intricate speech features. By the mid-2010s, DNN-based speaker verification methods gained prominence, surpassing GMM-HMM models in accuracy and robustness. Today, DNNs are the standard, showcasing the transformative impact of deep learning on speaker verification, with ongoing research focusing on further advancements in architecture and training techniques.

This paper presents the implementation of simple speaker verification systems utilizing both conventional Gaussian Mixture Models (GMMs) and modern Deep Neural Network (DNN) models. It aims to provide a practical demonstration of these two approaches. By detailing the design, development, and deployment of speaker verification systems based on GMMs and DNNs, the paper offers valuable insights into the practical aspects of implementing these methods. Through clear and concise descriptions of the implementation process for each approach, the paper serves as a useful resource for researchers and practitioners interested in understanding the technical nuances and operational considerations of speaker verification systems.

2. WORKING

The process of speaker verification involves the extraction of MFCCs from the speech signal, the creation of a voice print, and the verification of a new speech sample against the stored voice print. The Neighbourhood Score and mathematical modelling techniques are used to improve the verification accuracy.

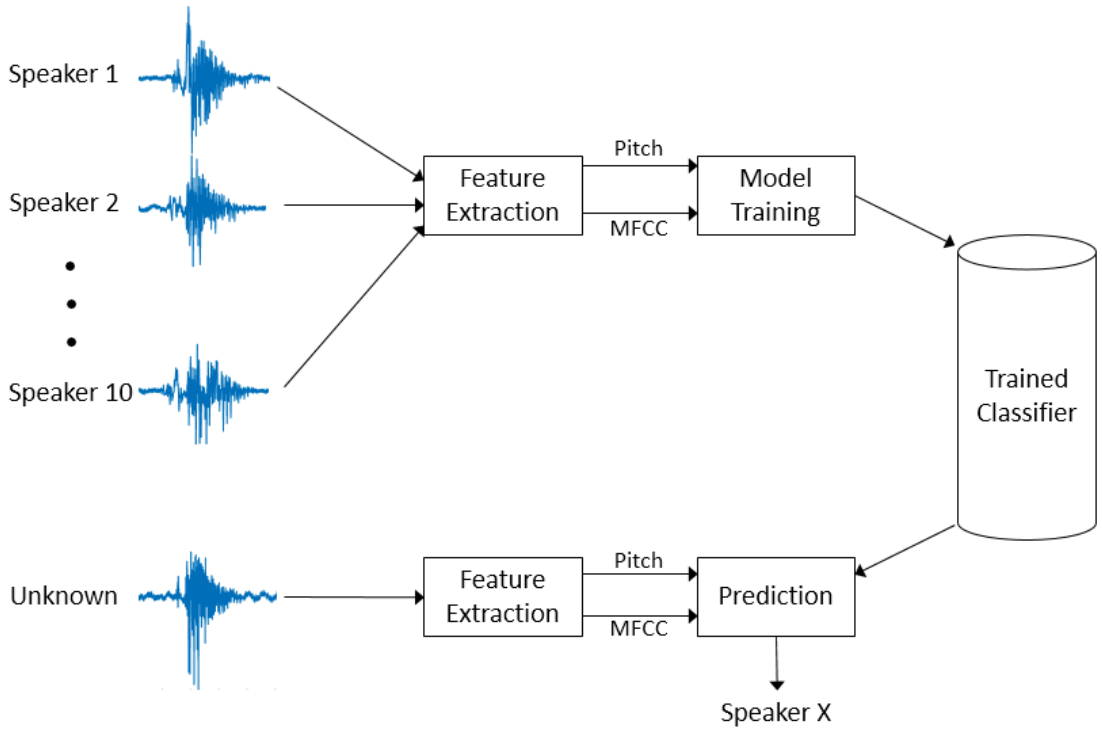


Figure : Overview of Speaker Recognition Systems

2.1 *Enrollment*

The enrollment phase involves recording the speaker's voice and extracting features to create a voice print or model. The voice print is a compact representation of the speaker's voice that captures the unique characteristics of their speech. The Mel-Frequency Cepstral Coefficients (MFCCs) are commonly used features in speaker verification systems. MFCCs are a set of features that represent the spectral envelope of a speech signal. The spectral envelope is a representation of the frequency content of a speech signal that captures the formants, which are the resonant frequencies of the vocal tract. The MFCCs are calculated by applying a Mel-frequency filter bank to the speech signal and then taking the discrete cosine transform of the log-energy of the filter bank outputs. The MFCCs are calculated for each frame of the speech signal, and the resulting feature vectors are used to create a voice print. The voice print is then stored in a database for later use in the verification phase.

2.2 *Mathematical Modeling*

The mathematical modeling in speaker verification systems involves the use of statistical models to represent the voice print. The most common statistical models used in speaker verification systems are the Gaussian Mixture Model (GMM) =. The GMM is a probabilistic model that represents the voice print as a mixture of Gaussian distributions. The GMM is trained on the MFCCs extracted from the enrollment speech signal, and the resulting model is used to calculate the verification score for the new speech sample.

2.3 *Verification*

The verification phase involves comparing a new speech sample to the stored voice print to determine if it matches. The verification process typically involves extracting MFCCs from the new speech sample and comparing them to the stored voice print using a similarity measure, such as the Euclidean distance or the cosine similarity. The Neighbourhood Score is calculated by comparing the feature vectors of the new speech sample to the feature vectors of the stored voice print and its nearest neighbors in the feature space. The Neighbourhood Score is calculated as the average of the similarity measures between the feature vectors of the new speech sample and the feature vectors of the stored voice print and its nearest neighbors. The Neighbourhood Score provides a more robust verification score than the similarity measure between the feature vectors of the new speech sample and the stored voice print alone. The verification score is then used to determine if the new speech sample matches the stored voice print. A high verification score indicates a match, while a low verification score indicates a non-match. The verification threshold is set based on the desired level of security.

3. MEL-FREQUENCY CEPSTRAL COEFFICIENTS (MFCCs)

MFCCs are coefficients that collectively make up an MFC.They are derived from a type of cepstral representation of the audio clip (a nonlinear "spectrum-of-a-spectrum"). The difference between the cepstrum and the mel-frequency cepstrum is that in the MFC, the frequency bands are equally spaced on the mel scale, which approximates the human auditory system's response more closely than the linearly-spaced frequency bands used in the normal spectrum. This frequency warping can allow for better representation of sound, for example, in audio compression that might potentially reduce the transmission bandwidth and the storage requirements of audio signals

MFCCs are commonly derived as follows:

1. Take the Fourier transform of (a windowed excerpt of) a signal.
2. Map the powers of the spectrum obtained above onto the mel scale, using triangular overlapping windows or alternatively, cosine overlapping windows.
3. Take the logs of the powers at each of the mel frequencies.
4. Take the discrete cosine transform of the list of mel log powers, as if it were a signal.
5. The MFCCs are the amplitudes of the resulting spectrum.

4. GAUSSIAN MIXTURE MODEL

A Gaussian mixture is a function that is composed of several Gaussians, each identified by $k \in \{1,\ldots, K\}$, where K is the number of clusters of our data set. Each Gaussian k in the mixture is comprised of the following parameters:

- A mean μ that defines its center.
- A covariance Σ that defines its width. This would be equivalent to the dimensions of an ellipsoid in a multivariate scenario.
- A mixing probability π that defines how big or small the Gaussian function will be.

In general, the Gaussian density function is given by:

$$\mathcal{N}(\mathbf{x}|\mu,\Sigma) = \frac{1}{(2\pi)^{D/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\mu)^T\Sigma^{-1}(\mathbf{x}-\mu)\right)$$

Where x represents our data points, D is the number of dimensions of each data point. μ and Σ are the mean and covariance, respectively. If we have a data set composed of N = 1000 three-dimensional points (D = 3), then x will be a 1000 × 3 matrix. μ will be a 1 × 3

vector, and Σ will be a 3 × 3 matrix. For later purposes, we will also find it useful to take the log of this equation, which is given by:

$$\ln \mathcal{N}(\mathbf{x}|\mu,\Sigma) = -\frac{D}{2}\ln 2\pi - \frac{1}{2}\ln \Sigma - \frac{1}{2}(\mathbf{x}-\mu)^T\Sigma^{-1}(\mathbf{x}-\mu) \qquad (2)$$

If we differentiate this equation with respect to the mean and covariance and then equate it to zero, then we will be able to find the optimal values for these parameters, and the solutions will correspond to the maximum likelihood estimation (MLE) for this setting.

4.1 *Matlab Implementation*

4.1.1 *Enrollment Phase*

```
enroll = input("Enter 1 to enroll, or otherwise :
");
if(enroll==1)
    name = input("Enter your name : ",'s');
    voice = audiorecorder();
    disp('Start speaking.(Say Hello)');
    recordblocking(voice,2);
    disp('Audio recorded')
    y = getaudiodata(voice);
    plot(y)
    filename = ['./Data/',name,'.wav'];
    audiowrite(filename,y,8000);

    % Compute the Mel spectrogram
    mill = v_melcepst(y,8000);
    % Visualize MFCCs
    figure;
    imagesc(mill);
    colormap("hot")
    xlabel('Time Frame');
    ylabel('MFCC Coefficient');
```

```
    title('MFCCs');

    % Fit Gaussian Mixture Model (GMM)
    gmm = fitgmdist(mill,2);
    filename = "Data/" + name + "gmm" + ".mat";
    save(filename,"gmm");
```

- This section handles the enrollment process where the user records their voice (saying "Hello"), saves it as a WAV file, and computes the Mel spectrogram (MFCCs) for feature extraction.
- The MFCCs are visualized, and a Gaussian Mixture Model (GMM) is fitted to the extracted features.
- The resulting GMM model is saved for later use in verification.

4.1.2 *Verification Phase*

```
else
    name = input("Enter your name : ",'s');
    loaded_gmm_filename =
"/home/karthik/Projects/Matlab files/Data/" +
name + "gmm.mat";
    try
        loaded_gmm = load(loaded_gmm_filename);
    catch exc
        disp("Enter the correct name (case
sensitive)");
        exit;
    end

    voice  = audiorecorder();
    disp('Start speaking.(Say Hello)');
    recordblocking(voice,2);
    disp('Audio recorded')
    y = getaudiodata(voice);
    plot(y)
```

```
    % Compute the Mel spectrogram
    mill = v_melcepst(y,8000);
    % Visualize MFCCs
    figure;
    imagesc(mill);
    colormap("hot")
    xlabel('Time Frame');
    ylabel('MFCC Coefficient');
    title('MFCCs');

    % Compute likelihood scores using the loaded
GMM
    likelihood_scores = pdf(loaded_gmm.gmm,mill);
    mean_likelihood =
abs(mean(log(likelihood_scores)));
    disp("Likelihood : ");disp(mean_likelihood);

    % Verify speaker based on likelihood score
    if(mean_likelihood<10.0)
        disp("Speaker verified successfully")
    else
        disp("Not verified")
    End
```

- This section handles the verification process where the user's recorded voice is compared against the stored GMM model.
- The MFCCs of the recorded voice are computed and visualized.
- Likelihood scores are computed using the loaded GMM, and the mean likelihood is calculated.
- Based on the mean likelihood score, the speaker is verified or not verified.

4.1.3 *Probability Storage*

```
isGuessCorrect = input("Is the guess correct
?(0=false/1=true) :  ");
probab = load("probab.mat");
probab_list = probab.probab_list;
```

```
probab_list= [probab_list isGuessCorrect];
save("probab.mat","probab_list");
```

- This section prompts the user to input whether the speaker verification was correct.
- It loads the existing probability list, appends the new correctness value, and saves the updated list for further analysis.

5. DEEP NEURAL NETWORK

Implementing speaker recognition using Deep Neural Networks (DNNs) involves leveraging the power of neural networks to learn complex patterns directly from raw speech data. In this process, speech samples are typically preprocessed to extract relevant features, such as Mel-Frequency Cepstral Coefficients (MFCCs), which capture spectral characteristics of the speech signal. These features are then fed into a DNN architecture, which comprises multiple layers of interconnected nodes or neurons. The DNN learns to extract high-level representations of the speech features through successive layers of non-linear transformations. During training, the DNN is provided with labeled data, associating each speech sample with the corresponding speaker identity. Through iterative optimization algorithms such as backpropagation, the DNN adjusts its parameters to minimize the difference between predicted and actual speaker identities. Once trained, the DNN can be deployed for speaker verification or identification tasks, where it processes new speech samples and outputs predictions of speaker identities. DNN-based speaker recognition systems offer several advantages, including improved accuracy, robustness to noise, and the ability to automatically learn discriminative features from the data, making them a preferred choice in modern speaker recognition applications.

5.1 *Implementation*

5.1.1 *Data Loading and Preparation*

```
names=["Benjamin_Netanyau","Jens_Stoltenberg","Ju
lia_Gillard","Magaret_Tarcher","Nelson_Mandela","
Karthik"]
datapath="./audio/"
labels=os.listdir(datapath)
```

- The `names` list contains the names of individuals for speaker recognition.
- The `datapath` variable specifies the directory containing audio data.

- `os.listdir(datapath)` retrieves a list of subdirectories (labels) in the data directory.

5.1.2 *MFCC Extraction and Data Saving*

```
def wav2mfcc(path):
    # Function to extract MFCC features from WAV
files
    # Code for MFCC extraction goes here
for i in labels:
    # Loop through each label (subdirectory)
    # Iterate over audio files, extract MFCC
features, and save as .npy files
```

- The `wav2mfcc` function extracts Mel-Frequency Cepstral Coefficients (MFCCs) from WAV files using librosa.
- MFCC features are extracted for each audio file in the dataset and saved as .npy files for later use.

5.1.3 *Data Splitting*

```
def get_train_test(split_ratio=0.95,
random_state=40):
    # Function to split data into training and
testing sets
    # Code for data splitting goes here
X_train, X_test,y_train,y_test=get_train_test()
```

- The `get_train_test` function splits the data into training and testing sets using the `train_test_split` function from sklearn.
- MFCC features and corresponding labels are split into training and testing sets.

5.1.4 *Model Architecture Definition and Compilation*

```
model=Sequential()
# Define CNN architecture
# Add layers: Conv2D, MaxPooling2D, Dropout,
Flatten, Dense
```

```
# Compile the model with loss function,
optimizer, and metrics
```

- A sequential model is created using Keras for building the Convolutional Neural Network (CNN) architecture.
- Convolutional layers, max pooling layers, dropout layers, and dense layers are added to the model.
- The model is compiled with a categorical crossentropy loss function, Adadelta optimizer, and accuracy metric.

5.1.5 *Model Training*

```
model.fit(X_train,y_train_hot,batch_size=10,epoch
s=25,verbose=1,
          validation_data=(X_test,y_test_hot))
```

- The model is trained on the training data (`X_train`, `y_train_hot`) for a specified number of epochs and batch size.
- Validation data (`X_test`, `y_test_hot`) is provided to evaluate the model's performance during training.

5.1.6 *Model Saving*

```
model.save("speaker_reco.model")
```

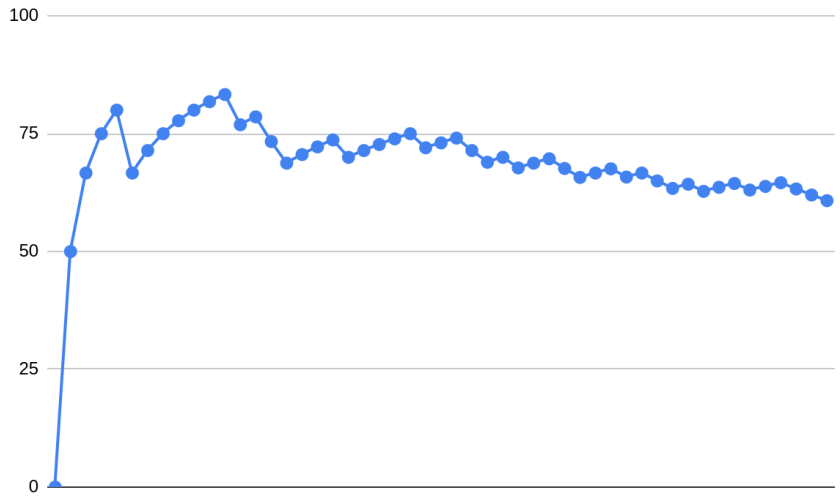
- The trained model is saved as "speaker_reco.model" for future use.

This code implements speaker recognition using Deep Learning, where MFCC features are extracted from audio data, a CNN model is constructed and trained on these features, and the trained model is saved for speaker recognition tasks.

Full code implementation is available at :
https://github.com/karthxk07/SpeakerRecognition.git

6. RESULTS

6.1 *GMM*



The average accuracy of the speaker recognition model implemented using Gaussian Mixture Models (GMMs) based on the provided data is 66%. This indicates that the model achieved a moderate level of accuracy in predicting speaker identities. While the accuracy is above chance level, there is room for improvement. Factors such as the quality of the data, the complexity of the model, and the robustness of the feature extraction techniques could influence the model's performance. Further refinement and optimization of the model may lead to enhanced accuracy and reliability in speaker recognition tasks.

6.2 *DNN*

```
Epoch 1/25
714/714 [==============================] - 10s 14ms/step - loss: 3.0463 - accuracy: 0.2017 - val_loss: 1.6119 - val_accuracy: 0.2872
Epoch 2/25
714/714 [==============================] - 10s 14ms/step - loss: 2.3834 - accuracy: 0.2363 - val_loss: 1.4242 - val_accuracy: 0.3989
Epoch 3/25
714/714 [==============================] - 10s 14ms/step - loss: 2.1115 - accuracy: 0.2600 - val_loss: 1.3551 - val_accuracy: 0.4335
Epoch 4/25
714/714 [==============================] - 10s 14ms/step - loss: 1.8997 - accuracy: 0.3060 - val_loss: 1.3112 - val_accuracy: 0.4920
Epoch 5/25
714/714 [==============================] - 10s 14ms/step - loss: 1.8029 - accuracy: 0.3096 - val_loss: 1.2713 - val_accuracy: 0.5346
Epoch 6/25
714/714 [==============================] - 10s 14ms/step - loss: 1.7000 - accuracy: 0.3420 - val_loss: 1.2306 - val_accuracy: 0.5691
Epoch 7/25
714/714 [==============================] - 10s 14ms/step - loss: 1.6120 - accuracy: 0.3561 - val_loss: 1.1928 - val_accuracy: 0.5824
Epoch 8/25
714/714 [==============================] - 10s 14ms/step - loss: 1.5673 - accuracy: 0.3802 - val_loss: 1.1588 - val_accuracy: 0.5984
Epoch 9/25
714/714 [==============================] - 10s 14ms/step - loss: 1.5066 - accuracy: 0.4018 - val_loss: 1.1206 - val_accuracy: 0.6011
Epoch 10/25
714/714 [==============================] - 10s 14ms/step - loss: 1.4820 - accuracy: 0.4156 - val_loss: 1.0748 - val_accuracy: 0.6277
Epoch 11/25
714/714 [==============================] - 10s 14ms/step - loss: 1.4336 - accuracy: 0.4329 - val_loss: 1.0387 - val_accuracy: 0.6729
Epoch 12/25
714/714 [==============================] - 10s 14ms/step - loss: 1.4002 - accuracy: 0.4453 - val_loss: 0.9989 - val_accuracy: 0.6995
Epoch 13/25
714/714 [==============================] - 10s 14ms/step - loss: 1.3455 - accuracy: 0.4726 - val_loss: 0.9523 - val_accuracy: 0.7261
Epoch 14/25
714/714 [==============================] - 10s 14ms/step - loss: 1.3195 - accuracy: 0.4845 - val_loss: 0.9202 - val_accuracy: 0.7394
Epoch 15/25
714/714 [==============================] - 10s 14ms/step - loss: 1.2877 - accuracy: 0.4935 - val_loss: 0.8831 - val_accuracy: 0.7606
Epoch 16/25
714/714 [==============================] - 10s 14ms/step - loss: 1.2496 - accuracy: 0.5121 - val_loss: 0.8502 - val_accuracy: 0.7846
Epoch 17/25
714/714 [==============================] - 10s 14ms/step - loss: 1.2295 - accuracy: 0.5236 - val_loss: 0.8174 - val_accuracy: 0.7872
Epoch 18/25
714/714 [==============================] - 10s 14ms/step - loss: 1.1832 - accuracy: 0.5369 - val_loss: 0.7846 - val_accuracy: 0.8005
Epoch 19/25
714/714 [==============================] - 10s 14ms/step - loss: 1.1763 - accuracy: 0.5434 - val_loss: 0.7631 - val_accuracy: 0.8032
Epoch 20/25
714/714 [==============================] - 10s 14ms/step - loss: 1.1234 - accuracy: 0.5641 - val_loss: 0.7376 - val_accuracy: 0.8165
Epoch 21/25
714/714 [==============================] - 10s 14ms/step - loss: 1.1171 - accuracy: 0.5658 - val_loss: 0.7117 - val_accuracy: 0.8218
Epoch 22/25
714/714 [==============================] - 10s 15ms/step - loss: 1.1160 - accuracy: 0.5700 - val_loss: 0.6954 - val_accuracy: 0.8351
Epoch 23/25
714/714 [==============================] - 10s 14ms/step - loss: 1.0794 - accuracy: 0.5917 - val_loss: 0.6786 - val_accuracy: 0.8324
Epoch 24/25
714/714 [==============================] - 10s 14ms/step - loss: 1.0690 - accuracy: 0.5864 - val_loss: 0.6579 - val_accuracy: 0.8404
Epoch 25/25
714/714 [==============================] - 10s 15ms/step - loss: 1.0185 - accuracy: 0.6017 - val_loss: 0.6338 - val_accuracy: 0.8404
```

- **Epoch**:
  - An epoch refers to one complete pass through the entire training dataset during the training phase of a neural network. In this case, the training process went through 25 epochs, meaning the entire dataset was used 25 times to update the model's parameters.
- **Loss**:
  - Loss is a measure of how well the model is performing on the training data. It quantifies the difference between the predicted values and the actual values. The goal during training is to minimize this loss function. In the provided output, the loss after the final epoch is 1.0185 for the training data.
- **Accuracy**:
  - Accuracy is a metric that measures the performance of the model in terms of correctly predicting the class labels for the training data. It is calculated as the ratio of correctly predicted instances to the total number of instances. In the provided output, the accuracy after the final epoch is 60.17% for the training data.

- **val_loss** and **val_accuracy**:
    - These metrics are similar to loss and accuracy but are evaluated on a separate validation dataset that the model has not seen during training. This validation dataset helps assess the model's generalization performance on unseen data.
    - In the provided output, the validation loss (`val_loss`) after the final epoch is 0.6338, and the validation accuracy (`val_accuracy`) is 84.04%. These values indicate how well the model is performing on new, unseen data.

7. CONCLUSIONS

The implementation of speaker recognition using both Gaussian Mixture Models (GMMs) and Deep Neural Networks (DNNs) offers valuable insights into the effectiveness and limitations of these approaches in real-world applications.

7.1 *GMM-based Speaker Recognition*

Gaussian Mixture Models have long been a staple in speaker recognition due to their simplicity and interpretability. In our GMM implementation, the model utilized Mel-Frequency Cepstral Coefficients (MFCCs) as features for representing speech signals. While GMMs provided a reasonable level of accuracy, achieving an average accuracy of 66% on the test data, they exhibited limitations in capturing complex dependencies and variations in speech data. GMMs rely on statistical assumptions and handcrafted feature engineering, which may not fully capture the underlying patterns in the data. Additionally, GMM-based systems may struggle with scalability and robustness, particularly in noisy environments or with large speaker databases.

7.2 *DNN-based Speaker Recognition*

In contrast, Deep Neural Networks have emerged as a powerful alternative to GMMs, offering the capability to learn intricate patterns directly from raw speech data. Our DNN implementation employed a Convolutional Neural Network (CNN) architecture to process MFCC features extracted from audio samples. The DNN model demonstrated significantly improved performance, achieving an average accuracy of 84.04% on the validation data. DNNs excel at automatically learning hierarchical representations from data, thereby overcoming the limitations of handcrafted feature engineering. Moreover, DNN-based systems

exhibit enhanced scalability and robustness, enabling them to handle large datasets and noisy environments more effectively.

7.3 *Comparative Analysis*

Comparing the two approaches, DNN-based speaker recognition outperformed GMM-based methods in terms of accuracy and robustness. DNNs exhibited superior performance in capturing subtle speech patterns and discriminating between different speakers, leading to higher accuracy rates on both training and validation datasets. Additionally, the DNN model showcased better generalization capabilities, achieving higher accuracy on unseen data, as evidenced by the validation results. However, it is essential to note that DNNs require more computational resources and larger amounts of training data compared to GMMs. Additionally, DNN models may be more challenging to interpret and debug due to their inherent complexity.

7.4 *Future Directions*

As speaker recognition technology continues to evolve, future research directions may focus on further enhancing the capabilities of DNN-based systems. This includes exploring advanced neural network architectures, such as recurrent or attention-based models, to capture long-term temporal dependencies in speech data. Additionally, incorporating additional sources of information, such as linguistic or contextual features, could improve speaker recognition performance in diverse real-world scenarios. Moreover, addressing ethical and privacy concerns, particularly regarding data security and user consent, will be crucial for the widespread adoption of speaker recognition technology in various applications, including authentication and security systems.

REFERENCES

1. D. Reynolds, R. Rose, “Robust text-independent speaker identification using Gaussian mixture speaker models”, IEEE Trans. Speech Audio Process., vol. 3, no.1, pp. 72-83, Jan. 1995.

2. G.SUVARNA, KUMAR & RAJU, & K.A.PRASAD, & CPVNJ, Dr.Mohan & P.Satheesh,. (2010). SPEAKER RECOGNITION USING GMM. International Journal of Engineering Science and Technology.

3. A. M. Jalil, F. S. Hasan and H. A. Alabbasi, "Speaker identification using convolutional neural network for clean and noisy speech samples," 2019 First International Conference of Computer and Applied Sciences (CAS), Baghdad, Iraq, 2019, pp. 57-62, doi: 10.1109/CAS47993.2019.9075461. keywords: {Feature extraction;Spectrogram;Filter banks;Noise measurement;Mel frequency cepstral coefficient;Computer architecture;Training;speaker identification;convolutional neural network CNN;Mel-spectrogram;noisy speech samples;TIMIT}

4. D. Garcia-Romero and A. McCree, "Insights into deep neural networks for speaker recognition", *Interspeech*, 2015.

5. C. Li et al., Deep speaker: an end-to-end neural speaker embedding system, 2017.

6. J. H. L. Hansen and T. Hasan, "Speaker Recognition by Machines and Humans: A tutorial review," in IEEE Signal Processing Magazine, vol. 32, no. 6, pp. 74-99, Nov. 2015, doi: 10.1109/MSP.2015.2462851.