Indian Institute of Information Technology, Allahabad

B.Tech 8th sem Major Project

"*Preprocessing Techniques for Speaker Recognition*"

Mentor

Dr. Ramesh Kumar Bukhya

Assistant Professor

Department of Electronics & Communication Engineering
Indian Institute of Information Technology, Allahabad
Deoghat, Jhalwa, Prayagraj-211012, Uttar Pradesh, India.
rkbukhya@iiita.ac.in

Ankit Singh - IEC2018076

Overview

List of Tables

List of figures

Abstract

Speaker recognition is the task of identifying persons from their voices.It is the task of confirming a user's identification by examining speaker-specific traits in their voice samples. This technique is very popular for biometric recognition in the world and can be very handy in areas where security is a major concern.

The recent success of deep learning in speech-related tasks has been a revolution in the field of speaker recognition. Its enormous success is also due to the availability of data. It is divided into two categories: (i) identification and (ii) verification. The speaker identification process determines which registered speaker makes a given speech, whereas the speaker identification process identifies which registered speaker makes a given utterance. The process of validating whether a particular speech is delivered by a postulated speaker is known as Speaker verification. Deep learning has a significant advantage over traditional approaches in terms of representation, since it can generate extremely abstract embedding characteristics from utterances.

The goal of our proposed study is to construct a Speaker identification model that can recognise the speaker utilizing multiple preprocessing processes, with an emphasis on the deep-learning-based model and MFCCs characteristics collected from the speech waves. This paper includes voice activity detection, Mel Frequency Cepstral Coefficients for feature extraction, then saving the pre-processed data so that processing time will be saved while executing. Then feeding this data to the 3-layer CNN model, which also consists of batch normalization and max-pooling. Max-pooling is applied to downsample the output of the convolutional layer by a factor of two. The activation function used is a rectified linear unit(ReLu). L2 Regularization is used for the overfitting issue. Tuning the hyperparameter-like learning rate, and epochs were very challenging as minute deflection take us up and down. The whole workflow is the first thing that loads the dataset and then splits it into a train, validation and test data splits then implements the CNN model. Then train the model, evaluate the model on the test splits, and finally, save the model so that it can be used afterward.

Index terms – speaker recognition, MFCCs, deep learning, CNN, speaker identification, speaker verification

Chapter 1: Introduction

1.1 Speaker Recognition

Speaker recognition is the method of automatically detecting who is speaking by analysing speaker-specific information included in voice waves in order to validate the identities of persons using systems. Speaker recognition is a relatively active research subject with important applications in disciplines including biometric authentication, forensics, security, and voice recognition, all of which have contributed to the discipline's sustained attention. It's divided into two categories: speaker verification and speaker identification.

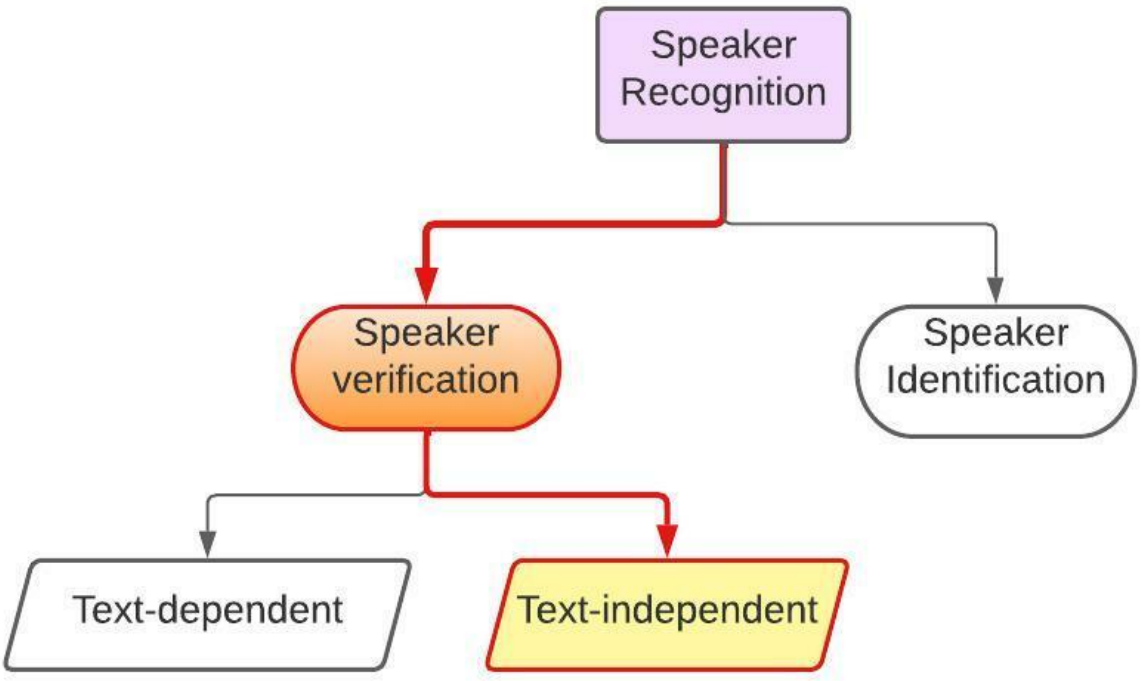


Figure 1: Classification of Speaker recognition

Speaker verification aims to verify whether a hypothetical person said a word based on their pre-recorded statements. The user interface converts a speech in the time domain or the time-frequency domain into a high-dimensional feature vector. This explains the recent advantage of deep learning-based speaker recognition. The backend first computes the similarity score between the test and speaker registration features and then compares the score with a threshold.

Speaker identification determines which registered speaker provides a given utterance.It basically does the job of matching/comparing the voice of the speaker with the database of reference.
It is further divided into two categories: (i) text-dependent and (ii) text-independent.

Text-dependent: Text-dependent is a method in which the test utterance corresponds to the text used during the enrolling phase. The test speaker has prior experience with the model. Low enrolment and trial stages are present in the local lexicon, resulting in an authentic result. Despite this, it confronts minimal scientific and technical obstacles. The first text-dependent speaker identification system established the key features of the present state of the art in the 1990s with feature extraction, speaker modelling, and score normalization using a likelihood ratio score. Since then, a number of architects have inquired on various occasions.

Text-independent: The voice signal's training and testing are completely unconstrained in a text-independent task. Both the training and testing phases of the speech signal normally take a long time to create. In this case, the test speaker has no prior knowledge of the enrollment phase samples in the testing phase.

A speaker's voice is regarded to include non-public developments of the speaker, given the specific pronunciation organs, and speaks to the speaker, for example, the specific vocal tract form, larynx size, accent, and rhythm. As a result, using a computer to identify a speaker from his or her voice is a possible option. No longer speak about speaker popularity through humans.

In this study a denoised vad algorithm is implemented to reduce the computational time and also discarding the robust noise. Then the hamming window is used so that information can be extracted for a smaller signal and can be examined in details. Otherwise it would be difficult to analyze the whole signal at once. Among the choice for feature extraction techniques for speaker

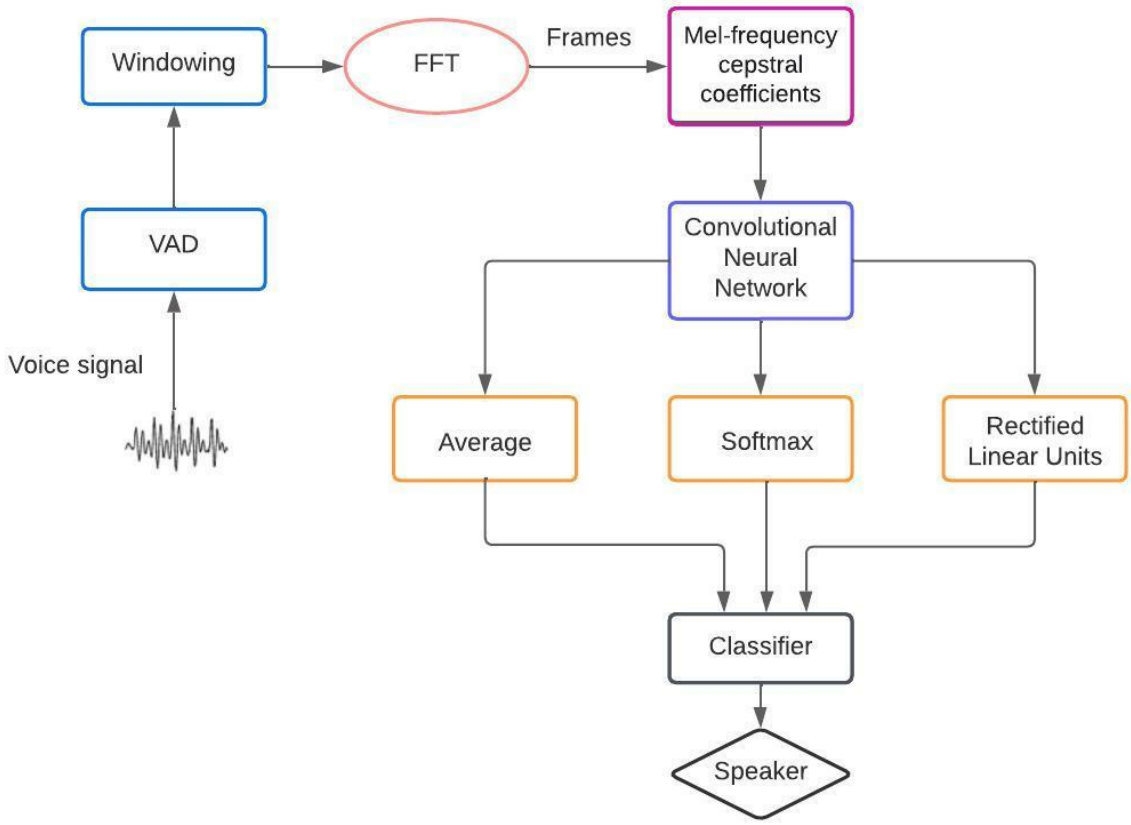


Figure 2: Speaker Recognition System

recognitiom, MFCCs feature extraction is used in this study as it is the newer and highly succesfull for the same . After Extraction of the features of the speaker, it is required to feed that information to the model which can be trained for the speaker recognition purpose. In this study, a CNN model is implemented. The fundamental characteristics of CNN are weight sharing, locality, and pooling and every single one of them has the potential to improve speaker recognition. Thus a CNN model can give better result . To update the weight and bais for the

neurons Rectified Linear Units activation function is used which adds piecewise non-linearity to the neurons. And thus the model can learn non-linearity as well. And at last a softmax output layer to compresses raw class scores into normalized positive values that add up to one which can be used for classification.

The speaker's voice is recognized to contain the speaker's own qualities. The form of the vocal tract, the size of the larynx, the accent, and the rhythm. As a result, the speaker's voice may be recognized. Speaker recognition research dates back to at least the 1960s. Many improved approaches aided the development of speech identification during the following 40 years.A collection of acoustic qualities, for example (eg, perceptual linear prediction coefficients and Mel frequency cepstrum coefficients). Voice control applications in everyday digital gadgets are gradually combining speech recognition into our daily lives as technology advances.

1.2 Motivation

- **The success of deep learning for speech recognition**

The huge success of deep learning in the domain of speech recognition motivated to do enhance the research on speaker recognition. Because the speech signal is vital in identifying the speaker from speech-based characteristics retrieved from the vocal tract, which are unique.

- **Vad can be used as part of systems for Speaker Recognition**

The focus is on robust VAD in noisy situations as VAD can help to improve the overall system. Automatic speech recognition (ASR), voice transmission, and other related applications Background noise is widespread and fluctuates greatly, making these particularly tough. Background noise can include automobile and machinery noises, nature sounds, gunshots, and a range of other sounds, depending on the area..

1.3 Literature Review

The relevant literature advises using distinct modeling strategies for each of the speaker modeling approaches. Popular modeling strategies for text-independent speaker recognition include

- Gaussian mixture models (GMMs)
- Neural networks.

It is not a novel strategy to use Gaussian mixture models for speaker recognition, but it is quite effective. GMMs are utilized in statistical data analysis, pattern identification, computer vision, speech and image processing, and machine learning, among other applications. It is one of the most popular models used for training in the processing of audio data. MFCC and GMM have been used together in previous studies to achieve the goal of correctly identifying speakers. Recent research follows deep learning-based methods for the implementation of the Speaker recognition system.

Neural networks are a relatively recent solution to speaker recognition that is beginning to gain traction in the industry. The speaker recognition technique employs a variety of neural networks. Deep neural networks (DNNs) are particularly good at speaker recognition, according to Richardson et al. Neural networks, according to Villalba et al., are the best performing act for speaker recognition.

1.4 Dataset Overview

Dataset availability is the reason for the success of the speaker recognition tasks. So finding a good dataset can lead us to better results. Dataset used is audioMNIST dataset which is freely available on the internet. There were a few more datasets that have been explored but they have a lesser amount of data and less number of users as well. One has a 2.5k audio sample with 6 users and another has 10k samples with 50 different users. Having more data can help users in better results thus considered this one as It contains 30k audio samples and 60 different users. Every user has more than 1k audio files in its corresponding folder. Audio files are available in .wav format which is quite good as python has lots of libraries to deal with the .wav extension. The length of audio files varies from 1 sec to 1 min. There is an additional file "audioMNIST_meta.txt" consisting of meta-information about the user such as age, and gender.
Highlight points:

- 30k Audio samples
- 60 users
- .wav format

Chapter 2: Implementation Overview

2.1 Block Diagram of Speaker Recognition System

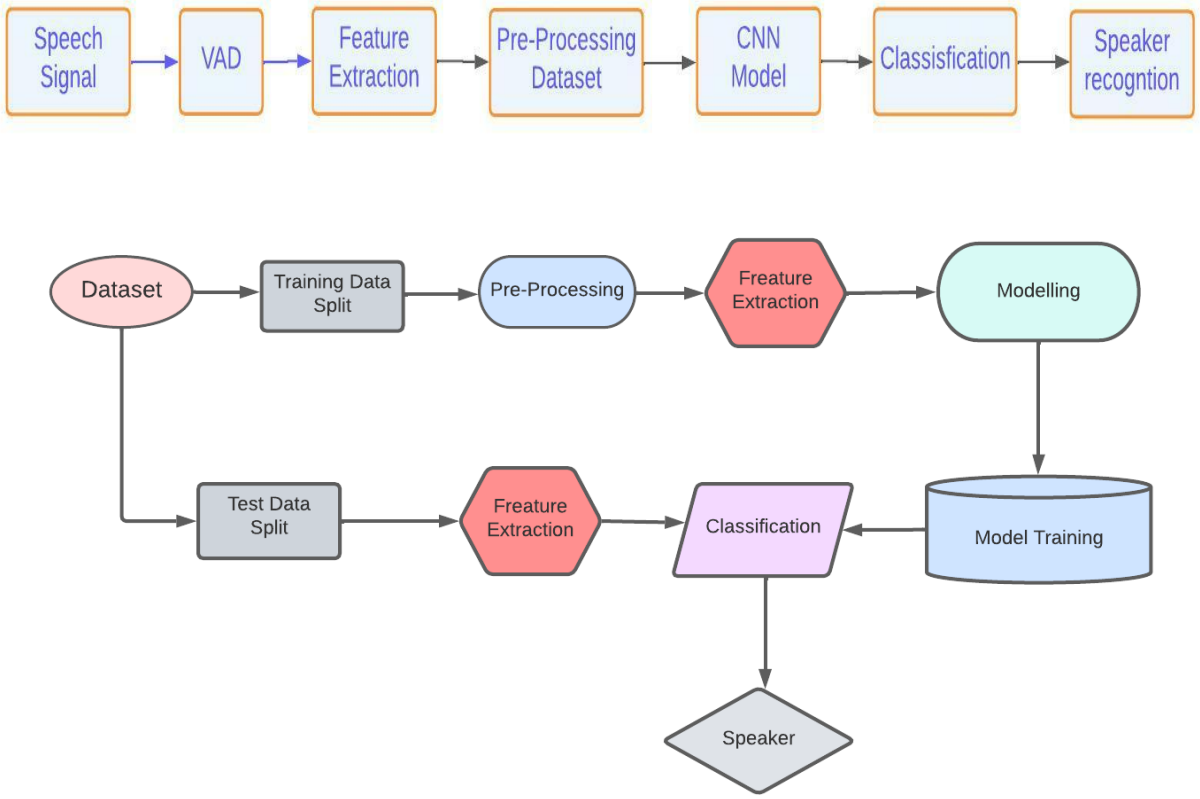


Figure 3: Block diagram of the Speaker Recognition system

The dataset consists of the speech signals which are directly not suitable for training purposes. Thus some preprocessing is required to make the dataset ready for training. Started with pre-emphasis of the audio signal. First voice activity detection is applied, which is here robust vad, to the whole dataset. The working of the vad is to detect the absence/presence of speech in an audio signal. If a non-speech part is detected then it will be rejected.

Then Iterate through all the audio files in the dataset, extract MFCCs which are the main features, and lately store in a JSON file all of that information. That data is going to be used by the deep learning model for training purposes. It is not done in real-time as it turns out that it takes quite a lot of time to pre-process all of this. It's better to do that offline than save all of that information in a JSON file and then retrieve it at training time.

Split the dataset into training, validation, and testing using sklearn. After that, I started the implementation of 3-layer CNN using TensorFlow. The first layer requires the input shape which is a 3d array as CNN requires a 3d input. It is important to specify the activation function. There are four choices after the analysis of the rectified linear unit activation function is used. The model has to know about the shape of our input as the first layer. And after that, it will be done programmatically. A regulariser, specifically an l2 regularizer is also added. It is used to improve the model and to tackle the issue of overfitting.
Two more layers are added to each convolution layer. First added a Batch normalization to speed up the process of training. Then, The output of the convolutional layer is downsampled by a factor of two using Max Pooling. All of the layers employ the same activation function, which is Rectified Linear Units (ReLu). After implementing 3rd convolution layer, the Output of this layer is flattened and fed into a dense layer. Then used a softmax output layer. It uses cross-entropy loss. The softmax function compresses raw class scores into normalised positive values that add up to one, allowing the cross-entropy loss to be applied.

2.2 Experimental setup

Google colab is used for running the experiment as it is freely available and has most of the libraries installed. It also provides the appropriate GPU necessary for performing calculations. A simple setup involves uploading the dataset to the drive and running the code on colab with the python 3.6+ version. Dataset consists of 30k audio files. Multiple splits of the dataset are used for comparison. Firstly train 70% and test 30% ,then train 80% and test 20% ,at last divide train 90% and test 10%. Necessary Libraries include TensorFlow, NumPy, matplotlib, sklearn, json, spicy, and math. Data accessing is also very easy by mounting the drive and uploading the dataset on the drive. The operating system used is Windows 10 or Ubuntu 20 with 8GB RAM and Intel i5 10th gen processor.

Chapter 3: Preprocessing

3.1 VAD

VAD, also known as speech detection, is commonly employed in real-world speech systems to improve resilience against additive sounds or to reject the non-speech component of a signal to decrease downstream processing computing costs. It determines the boundaries of the speech part and eliminates the mute segment. Because mute/noise doesn't contain any necessary information for our purpose thus eliminating it will buy us time for computation. There can be many small gap segments present in the data.It must discard each gap segment after identifying it in order to improve the overall performance of the systems. It may also minimise the mistake rates of speech identification systems by cutting the noise-only parts.

VAD involves feature extraction as well as categorization. Various speech characteristics, including as energy, zero-crossing rate, and Mel-frequency cepstral coefficient, have been studied in the literature (MFCC). Although there has been lots of research done in the field of VAD under a clear environment but accuracy under unseen environments is still an unsolved problem

Why**:**

- Reduces CPU
- memory, and
- battery consumption

for on-device speech recognition

Issue**:**

There are some issues that remain unsolved even after using this latest technique of robust voice activity detection:

- Loud noise is classified as speech
- Soft speech is classified as noise

This method uses two steps to filter the noise from the speech following the vad part. In the first step, the presence of pitch is determined using posteriori SNR. As the presence of pitch indicates, there is a speech segment otherwise if no pitch is detected then the segment should be considered as the noise segment and that segment will be eliminated. In the second step, the speech segment was enhanced. Several methods have been explored for this. Then the segments were merged which remains after removing the gap and noise segment from the speech. Then, to produce pitch segments, the adjacent frames containing pitch are put together. Assume that each speech segment will have a number of speech frames with pitch. First, they are grouped together to form a pitch frame into pitch segments.

Both ends of the pitch portions are expanded to contain both vocal and unvoiced sounds, as well as non-speech elements. After that, the expanded pitch segments of the denoised speech signal are subjected to an a posteriori SNR weighted energy difference to identify voice activity.

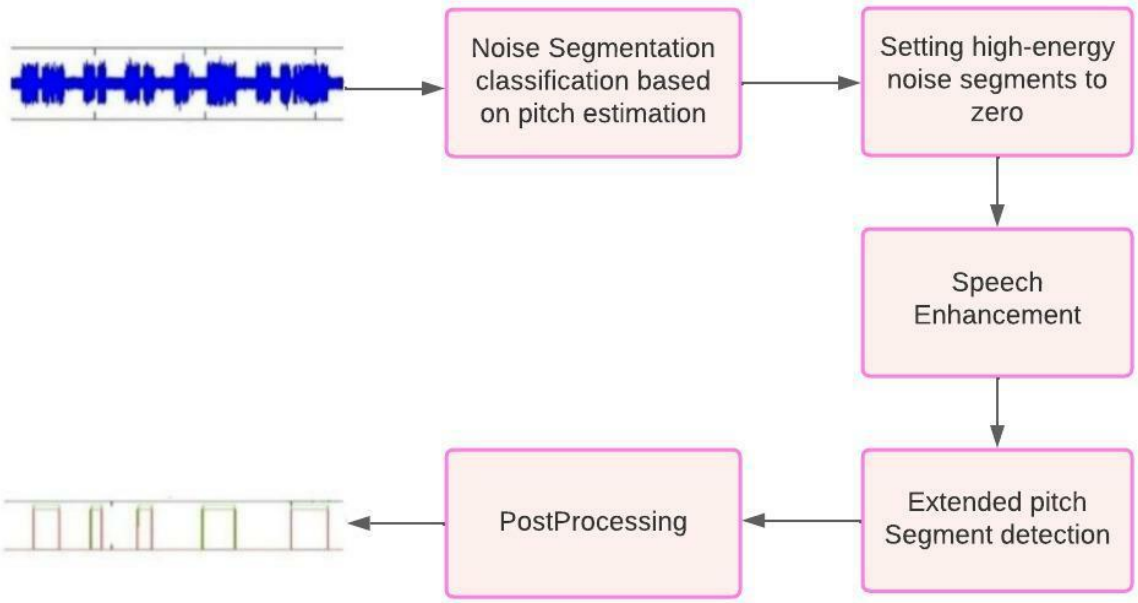


Figure 4: Block Diagram rVAD

Noise Segmentation Classification based on pitch detection

High-energy segments are detected by using a posteriori SNR weighted energy difference

$$m(t) = \sqrt{|l(t) - l(t-1)| * max(SNR(t), 0)}$$

where $l(t)$ the energy of the m th frame of noisy speech x(n), and SNR(t) is a posteriori SNR that is calculated as the logarithmic ratio of $l(t)$ to the estimated energy of the m th frame of noise signal v(n):

$$SNR(t) = 10 * log_{10} \frac{l(t)}{\overline{l_v}(t)}$$

Noise energy $\overline{l}_v(p)$ is calculated as the smooth version of $l_v(p)$ with forgetting factor of 0.9 as below

$$\overline{l}_v(p) = 0.9 * \overline{l}_v(p-1) + 0.1 * l_v(p)$$

Central-smooth the a posteriori SNR weighted energy difference, where N=18

$$\overline{d}(t) = \frac{1}{2N+1} \Sigma_{k=-N}^{N} d(t+k)$$

Setting high energy segment to zero and Speech Enhancement

Classify a frame as a high-energy frame if $\overline{d}(t)$ is greater than a threshold $\theta_{he}(t)$. For each super segment p (containing 200 frames), $\theta_{he}(p)$ is computed as follows:

$$\theta_{he}(l) = \gamma * max\{l((p-1) * 200 + 1), \ldots, l(p) \ldots, l(p * 200)\}$$

where $\gamma = 0.25$, $\theta_{he}(t)$ takes the threshold value $\theta_{he}(l)$ of the l th super-segment which the m th frame belongs to. Alternatively, the threshold can be calculated recursively using a forgetting factor.

Sequential high-energy frames are grouped together to generate high-energy segments.

If a high-energy segment contains no more than two-pitch frames, the segment is considered noise, and the segment's samples are set to zero.

Extended Pitch Segment Detection

The denoised speech and pitch information generated in the preceding phases are used in the VAD algorithm. All speech segments should have a number of speech frames with pitch, according to the basic assumption. Pitch frames are initially sorted into pitch segments, which are then extended by 60 frames from both ends depending on speech statistics to include voiced sounds, unvoiced sounds, and maybe non-speech elements.

Post Processing

If $\overline{d}(t)$ of the frame is greater than the threshold (θ_{vad}) then classify this frames as speech

$$\theta_{vad} = \eta * \frac{1}{N}\Sigma_{k=1}^{N}\overline{d}(k)$$

where N is the total number of frames with a pitch in the extended pitch segment and the default value for η is set to 0.3.

Non-speech frames are defined as those that are 33 frames to the left of the pitch segment and 47 frames to the right. Frames inside 12 frames to the right and 5 frames to the left of the pitch segments, on the other hand, are categorized as speech. Segments with less than 0.05 times the overall energy are also discarded.

3.2 Features Extraction

MFCCs (Mel. Frequency Cepstral Coefficient)

The most extensively used method for extracting features from an audio signal is MFCC.

Computation of MFCC

MFCCs are commonly derived as follows:

Step 1: Compute the Fourier transform of a signal (a windowed snippet).

Step 2: Using triangular overlapping windows, map the power of the spectrum produced on top of onto the mel scale.

Step 3: Record the power logs for each of the Mel frequencies.

Step 4: Use Mel log powers as signal, compute the discrete cosine transform.

Step 5: The MFCCs are the resulting spectrum's amplitudes. There could be variations on this method, such as changes to the form or spacing of the windows used to map the size, or the addition of dynamics capabilities such as "delta" and "delta-delta" (first- and second-order frame-to-body distinction) factors.

Block Diagram

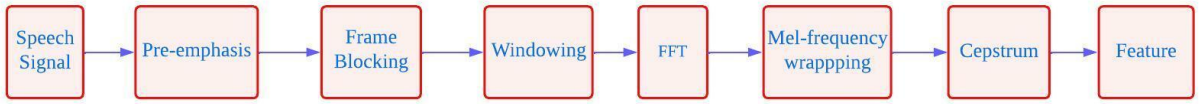


Figure 5: Computation of MFCCs

1. Pre-emphasis

   Pre-emphasis Filter is used for the amplification of higher frequencies. It is a filter used to obtain a smoother spectral form. It reduces the numerical calculation of Fourier Transform and improves the SNR of the signal.It can be calculated as

   $$y(n) = x(n) - \alpha x(n-1) \quad \ldots(1)$$

   where α varies from 0.9 to 1.0

2. Frame Blocking and Windowing

   It divides a continuous voice stream into N-sample frames, with neighboring frames separated by M (M<N). This acoustic stream is divided into numerous overlapped frames in this procedure, so no single signal is lost.

Windowing is used as there will be multiple phones in the given audio signal. The window is defined as w(n), where N is the number of samples in each frame

$$y_1(n) = x_1(n)w(n), \qquad 0 <= n <= N - 1 \qquad \ldots (2)$$

y(n) represents the convolved signal of the input signal with the hamming window.

Hamming Window:

$$w(n) = 0.54 - 0.46\, cos(\tfrac{2\pi n}{N-1}),\ 0 <= n <= N - 1 \qquad \ldots (3)$$

3. Fast Fourier Transform(FFT)

FFT is used to generated spectrum for the signal of the frame x_k .It is represented by

$$X_n = \Sigma_{k=0}^{N-1} x_k e^{-2\pi jkn/N}$$

Where n=0,1…N-1, j=$\sqrt{-1}$ and X_n is the n-frequency pattern

4. Mel-scaling (using Mel-filter bank shown in fig below)

Humans can't perceive sound linearly. The frequency Mel scale is the scale that works on the human ear. Thus Mel-scaling is used.

MEL SCALE
These formulas are used to switch between frequency and mel scale

Hz to Mel

$$m = 2595 * log\left(1 + \tfrac{f}{500}\right) \qquad \ldots\ldots\ldots\text{eq(i)}$$

Mel to Hz

$$f = 700 * \left(10^{\frac{m}{2595}} - 1\right) \qquad \ldots\ldots\ldots\text{eq(ii)}$$

A Mel-Filter Bank is used for mapping these frequencies.It is shown in below figure. The y axis represents Filter Multiplication Factor ranging from 0-1 and x axis downward represents the frequency in hertz and x axis upward maps to their respected Mel-scale Frequency

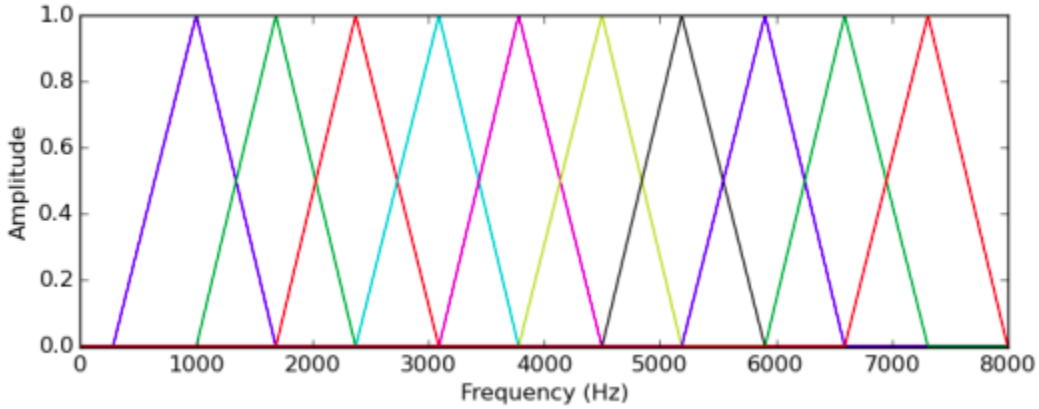


Figure 6: Mel-Filter Banks

The resulting Mel frequency coefficients are given as

$$F(k) = \sum_{l=0}^{N-1} |X(l)|^2 H_k(l), \qquad 0 \leq k \leq K-1$$

The frequency response of the *m*th Mel-scale filter is denoted as $H_m(l)$ and *M* is total number of triangular Mel-weight filters.

5. Discrete cosine Transform

The Discrete Cosine Transform is applied to the log of the Mel-spectral coefficients to obtain the Mel-Frequency. If MFCCs are denoted by c(n) then its formula is:

$$c(n) = \Sigma_{m=0}^{M-1} log(F(k))\, cos(\tfrac{\pi n(m-0.5)}{M}), \qquad n = 0, 1, \ldots . C-1$$

Where C is the number of coefficients, F(k) represents the Mel frequency coefficients

- Simplified version of the Fourier transform

- Get real-value coefficients
- Decorrelated energy in different mel-bands
- To depict a spectrum, the number of dimensions is reduced

How Many Coefficients?

- first 12-13 coefficient
- These coefficients keep most information (formants, spectral envelope)
- use delta & double delta MFCC (1st & 2nd order derivatives of MFCC Coeff.)
- Adding up, Total coefficients == 39

Important Features

- Identification of sound
- timbres
- phonemes
- formants

Delta and Double Delta Coefficient

- These are the derivatives of MFCCs
- These content dynamic features like the emotion of the speaker
- Delta coefficients can be calculated by subtracting MFCCs coefficients of that frame from the previous frame. For the nth frame

$$d_t = \frac{\Sigma_{l=1}^{N} l(s_{t+l} - s_{t-l})}{2\Sigma_{l=1}^{N} l^2}$$

- where d_t is a delta coefficient, for frame t computed in terms of the static coefficients s_{t+l} to s_{t-l}. A typical value for N is 2.

- Delta-Delta (Acceleration) coefficients are derived similarly to static coefficients but from deltas rather than static coefficients.
- Double Delta coefficients can be calculated by subtracting the Delta coefficients of that frame from the previous frame. For frame t,

$$dd_t = \frac{\Sigma_{l=1}^{N} l(d_{t+l} - d_{t-l})}{2\Sigma_{l=1}^{N} l^2}$$

3.3 Data Computation

Data needs to be built programmatically. So the first step is to loop through all the sub-directories. The Python package os.walk was used to recursively travel through a folder hierarchy, in this example from top to bottom, using the top-down tour. In the for loop, a tuple of three is passed, which stores some important information. In the first, the path of a directory is stored, and then in the second, all the names of the sub-directories in that directory. In the third index of tupple filenames that are in the first index of tuple has been saved which is going to be used inside further loops.

Now It is ensured that this level is not the dataset level so in other words It is not a root level so to do that conditional statements have been used to check Why is this important to specify? Because os.walk begins with the dataset path and then recursively descends to the sub-directories. Skipping dataset_path itself as it is not of our interest.

A dictionary is created to store important things like mapping, labels, MFCCs, and filenames. The first thing stored is the mappings and this is going to be equal to a list so mappings basically map to different users, for example, user1 is mapped with zero and user2 can be mapped with one and so on. Mapping is very important here as without mapping the data it will be difficult to pass it to the Neural network as Neural Network needs to pass numbers and so that's why mapping these users to numbers is an important step.

Then labels are created. Labels are a list in themself and so there is one value for each of the audio files that consist in the dataset and this value is like the target value that is expected. The first zero index data that is passed into the neural network for training is equal to some user whereas the second index data that is passed is an audio file that passes to the deep learning model is equal to one whose mapping is known. These are like the output that is expected target outputs.

Then the dataset is loaded. Two variables are required while loading an audio file using the librosa library that is the signal itself, and its sampling rate. If some specific value of sampling rate is required then it can be passed as an argument. After loading, the audio file must be longer by at least one second, or else it will be discarded. All the audio files that are shorter than one second have been discarded as there should be no data which means no required information if the audio file is less than one second.

All the audio files are iterated in the dataset and extracted MFCCs which are our main features. There are three arguments that are specific to MFCCs structure. The first one is quite self-explanatory, that is the number of coefficients. For this research, it is considered to be 13 so after computing MFCCs only 13 coefficients are taken. The second argument is called hop_length. Hop_length is measured in a number of frames and the value of hop length is set to

512 which is quite a customary hop length for extracting MFCCs.While extracting MFCCs, the audio signal is divided into equal segments and segments with equal length and then a snapshot is taken with MFCCs at each of these segments, and hop length tells how big the segment should be in a number of frames. The third argument is the number of fast Fourier transforms. After the computation of MFCCs, a fast Fourier transform is computed. FFT basically tells us how big the window for fast Fourier transform we want to consider and Its value is set to a customary 2048.

In the MFCCs coefficient, dynamic information of the speaker was missing thus Delta and Double delta coefficients of MFCCs are also included which add up to 39 coefficients. This function gives an n-d array. So first of all, transpose of that array is needed and then this has to be cast to a list because It can't be stored directly like a frame from an n-d array.

And lately data computed has been saved in a JSON file, with all of that information being utilised by a deep learning model for training. It is not done in real-time because it turns out that it takes quite a lot of time to pre-process all of this. It’s better to do that offline than save all of that information in a JSON file and then retrieve it at training time. This way lots of time can be saved.

Here is an overview, In the first iteration, all the subdirectories are explored. So that all the speakers must be included. Then ensure that it is not a root level and at this point, speakers are added to the mappings. Moving through all the files loaded the audio file and ensured that an audio file which is a signal has enough samples to consider which is equal to one second. Further, extract the MFCCs and store all of that information. The last thing that remains to do is to store that store.

Chapter 4: Model Overview

Model: A sequential Model is used for implementation.

- **Sequential Model**
  This model is based on machine learning and is used to input or output data sequences. Text streams, audio, and video snippets, and time-series data are all examples of sequential data.

In this paper, TensorFlow is used for the implementation of the neural network because it allows large-scale neural networks with many layers. What has been done is building the design/architecture of a convolutional neural network then compiled, trained, tested, and saved so that It can be reused. Here are the high-level entry points like the process that is covered. The next step is to build the CNN model so here I implemented the neural network followed by training and then evaluated the model using the test sets.

4.1 Model Implementation

In the model implementation function, two parameters are required. One is the input shape and the other one is the learning rate. Learning rate is a parameter that is used in the optimization algorithm. After the exploration, Adam is used as an optimizer and Its value is set to 0.0001 which is a quite customary value to have for a learning rate. The input shape is the shape of the input data that has to be fed into the CNN model so in order to get it some extraction has to be done

A three-dimensional array is required for input shape because while dealing with convolutional neural networks three-dimensional inputs are required. The first dimension is the number of segments, which are extracted MFCCs at evenly spaced segments, and the length of a segment is determined by the sampling rate, which is equal to the total number of samples in a particular

audio file divided by the hop length. This will reveal how many portions there are. The number of coefficients extracted is the second dimension. There are many coefficients in MFCCs but after the analysis, it was found that the first 13 coefficients contain most of the information that is required. It contains information like formants, timbres, and phonemes of the speaker. These 13 coefficients consist of only static information. To add the dynamic information delta and double delta coefficients which are the derivatives of MFCCs coefficient are added to form a total of 39. Delta coefficients can be calculated by subtracting MFCCs coefficients of that frame from the previous frame The third dimension is whose value varies from 1-to 3.So, as previously stated, this third dimension is critical for a CNN because it carries information about the depth or the so-called channel of an image and in the case, for example, of a grayscale image, the depth is equal to 1 if there are RGB images, then this channel would be equal to 3 but after research, it was discovered that MFCCs is in particular so it's like dealing with some data like a grayscale image, so depth equals to 1. So this is the input shape.

Table 1: Types of Activation Function

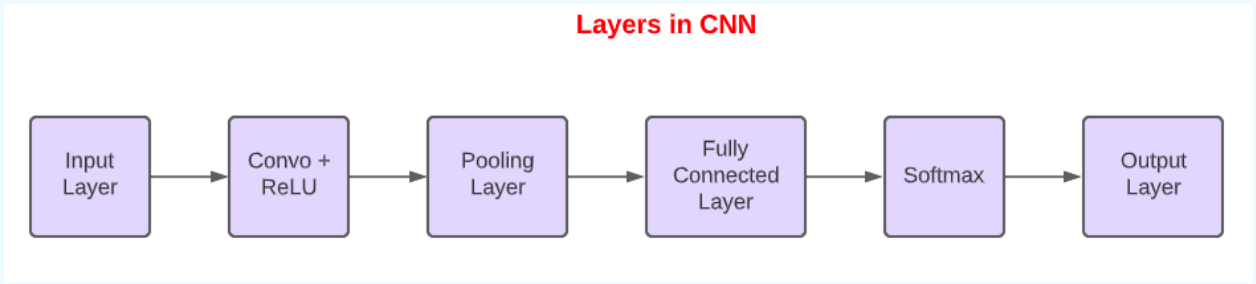
S.No	Activation Function	Formula
1	Sigmoid	$\sigma(x) = \frac{1}{1+e^{-x}}$
2	Tanh	$tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$
3	ReLu	$f(x) = max(0, x)$
4	Softmax	$S(y_i) = \frac{exp(y_i)}{\sum_{j=1}^{n} exp(y_j)}$ where y is an input vector to softmax function

To build a model, it is required to specify the input shape, the learning rate and the activation function. For the convolutional layers, the rectified linear unit activation function is used and for scaling numbers/logits into probabilities at the final layer, the softmax activation function is applied.

The model has to know about the shape of input as the first layer. And after that, it will be done programmatically. The first layer has 64 filters and a kernel of 3 by 3. A regulariser, specifically an l2 regularizer is also added to improve the model and tackle the issue of overfitting. Two more layers are added to the first convolution layer. First added a Batch normalization to speed up the process of training.The output of the convolutional layer is then downsampled by a factor of two using Max Pooling. A (3,3) filter and (2,2) strides is used in the max-pooling layer. Max value of 2*2 stride is carried forward to reduce the dimensionality.

The second convolutional layer has 32 filters with a kernel size of 3 by 3. ReLu activation function is used to include piecewise non-linearity. Max pooling layer has strides 2 by 2. The third layer has 32 filters with kernel sizes of 2 by 2. The activation function is Relu and the max-pooling layer has strides of 2 by 2.

After implementing the 3rd convolution layer, the Output of this layer is flattened and fed into a dense layer. Then used a softmax output layer. It uses cross-entropy loss. The softmax function is used to compress raw class scores into normalized positive values that sum to one when added together, allowing the cross-entropy loss to be applied.



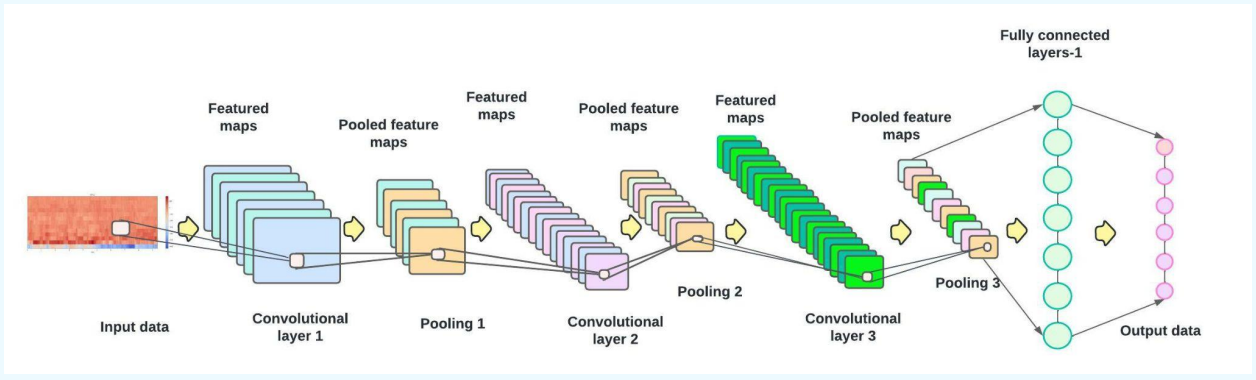


Figure 7: CNN Model Overview

Training

The model is built so now it requires training the speaker recognition system and in order to do that there is a method that comes directly from the Keras API and it's called.Fit(). It needed to pass in some different arguments so first of all, It needed to pass the training data to x_train(that was defined earlier) and both the inputs and the outputs. Then specify the number of epochs which is the number of times our network needs to train and It is set to a constant called EPOCHS with a value equal to 40. So now what's a network well a the number of epochs tells, how many times the network is going to see the whole data set for training purposes. In this case, It is set to 40 which basically means that the data set is passed for training purposes 40 times through the network. To speed up the process, data is divided into batches thus Another value that is another argument called batch size is defined and set this is equal to yet another constant that has been set to 32. So the batch size specifies the number of samples that the network will see before an update and runs like the backpropagation algorithm.

The system is trained. Now its performance is evaluated on the test set and so to get back from this evaluation process a test error that's also called loss is received. It also gives back a test accuracy as well and so this is equal to model dot evaluate where to evaluate is another method coming from the Keras API and It requires some argument like X_test and Y_test. This way model was evaluated. The results of this evaluation process are the Tensorflow test accuracy and the test error. It passed in test error and then a test accuracy and after passing the test accuracy and finally, saved this model. For saving there is another method from Keras API that's called a

model dot save and it is required to pass the saved data path. This is another constant that has been specified. Now set it equal to “model.h5” and the format of the saved model is h5.
The whole high-level process is defined so just to summarize what’s done. Loaded the train, validation & test data split as the first step then the CNN model is built with 3 convolutional layers and one dense layer. After the implementation, the model is trained, evaluated, and finally, saved so that it can be loaded when required for inference or prediction.

Max pooling:

It is a layer of CNN used to make the model more efficient and reduce the spatial size to minimize the number of parameters. 2*2 max-pooling is used where the max value of 2*2 matrix is carried forward to reduce the dimensionality.It is used to down sample the output of by the factor of 2 .

Algorithm: Measure Accuracy and loss

Input: the training dataset X_{train}, the evaluation dataset X_{eval} , GPU, and $epochs$

Output: Accuracy and the Loss of the Model

Steps

1. accuracy ← 0;
2. loss ← 0;
3. for i ← 1 to epochs do

    Train the CNN on X_{train} ;

    evalute_accuracy, evalute_loss ← Evaluate the accuracy and loss on X_{eval};

    if accuracy < evalute_accuracythen

        accuracy ← evalute_accuracy;

        loss ← evalute_loss;

    endif

4. endfor
5. *evaluate* ← evaluation score using loss function with accuracy, loss ;
6. **return** accuracy, loss.

4.2 Convolutional Layer's Structure

Input Shape:

The Model is fed with a 3-dimensional input and we can get that information from x[0], x[1],x[2] .Input layer size is (16,39,1)

16 → number of segments (number of samples/hop length)

39→ number of MFCCs coefficients we consider

1 → used for gray-scale images

Table 2 : Model Overview

Layer(type)	**Output Shape**	**Parameter**
Conv2D_1	(14,37,64)	640
batch_normalization_1	(14,37,64)	256
max_pooling2d_1	(7,19,64)	0
Conv2d_2	(5,17,32)	18464
batch_normalization_2	(5,17,32)	128
max_pooling2d_2	(3,9,32)	0
Conv2d_3	(2,8,32)	4128
batch_normalization_3	(2,8,32)	128

max_pooling2d_3	(1,4,32)	0
flatten	(128)	0
Dense	(60)	8256

Total Parameter: 35,900

Trainable Parameter: 35,644

Non-Trainable Parameter: 256

Chapter 4 : Results

4.1 Accuracy and Loss

Accuray and loss curve for different data split

For 70-30 split

- Test Accuracy : 88.79 %
- Test Loss : 0.57 %

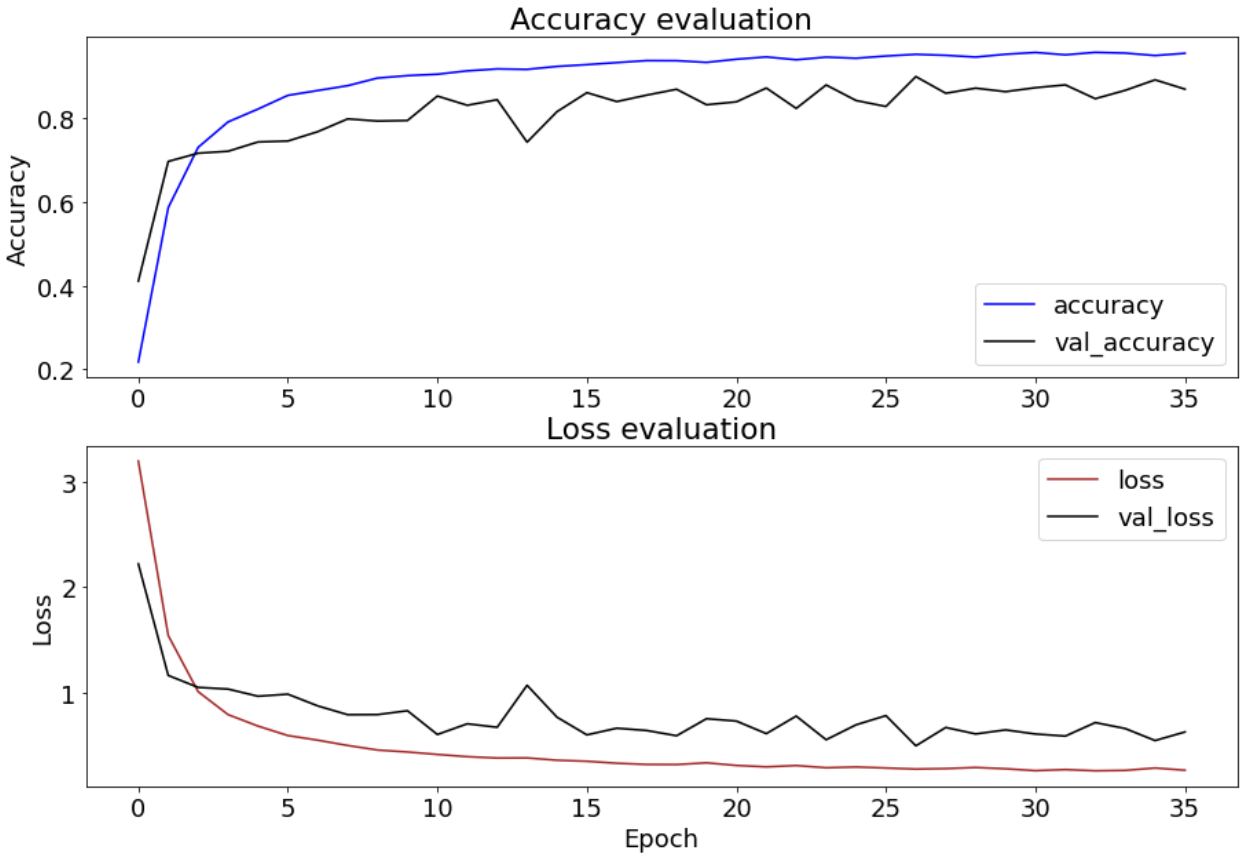


Figure 8 : Accuray and loss curve for train 70% and test 30%

For 80-20 split

- Test Accuracy: 90.03 %
- Test Loss: 0.49 %

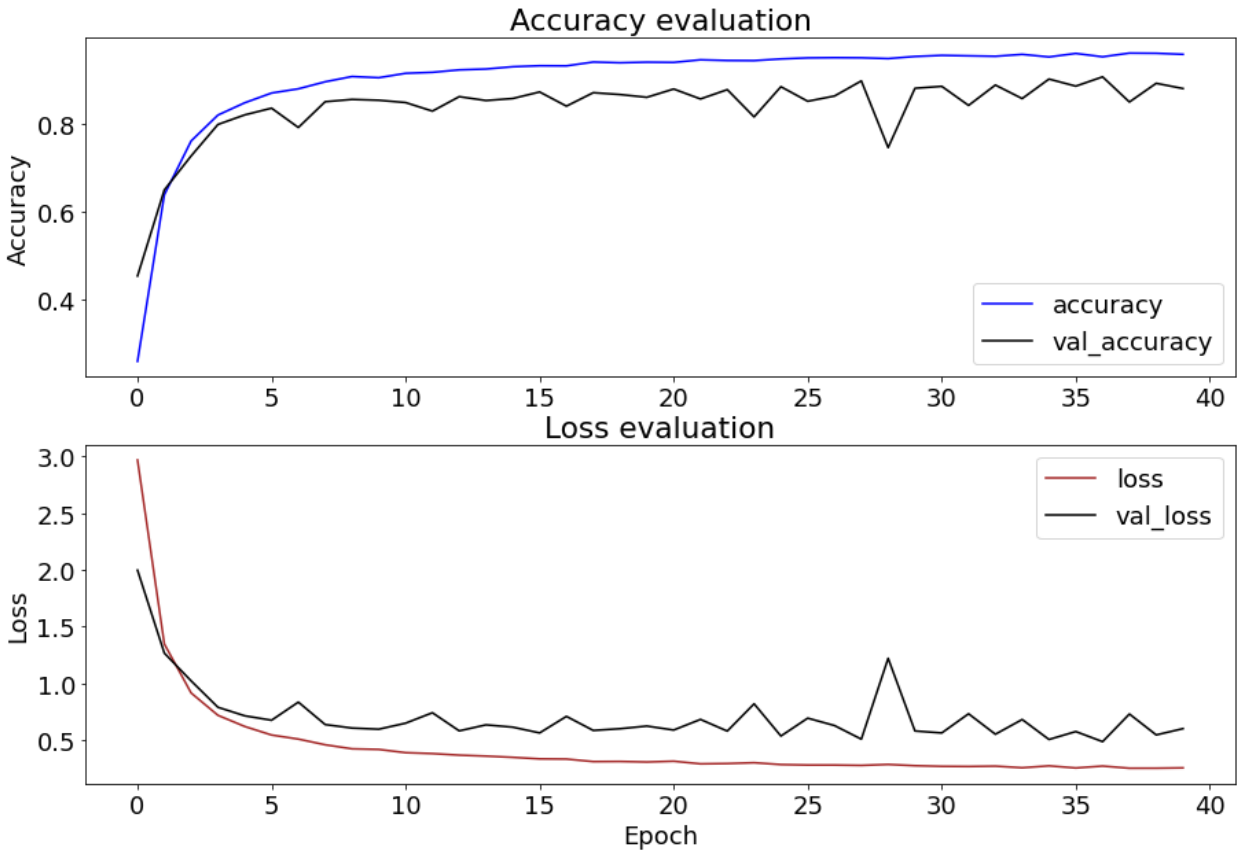


Figure 9: Accuray and loss curve for train 80% and test 20%

For 90-10 split

- Test Accuracy: 89.49 %
- Test Loss: 0.52 %

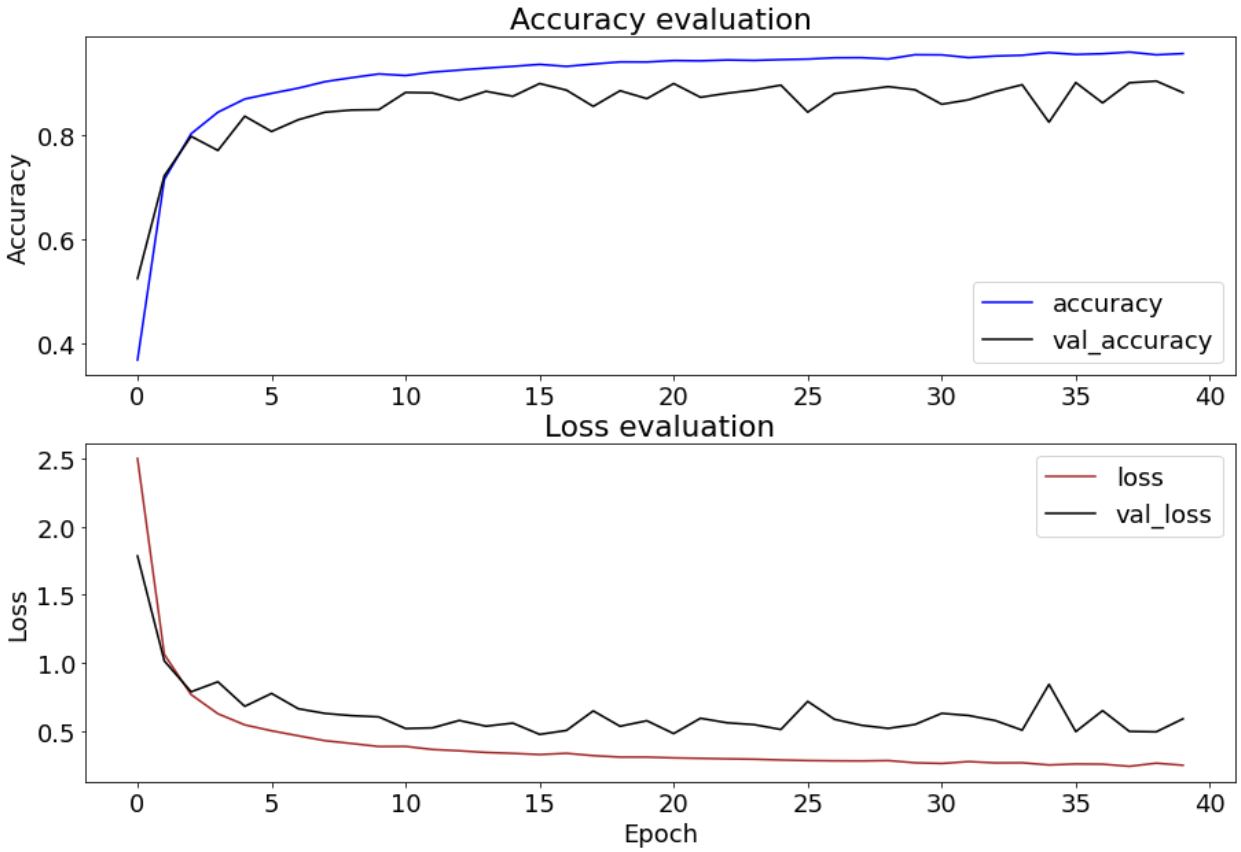


Figure 10: Accuray and loss curve for train 90% and test 10%

The model is trained for 40 epochs and after training, it was evaluated on the test data split. The given diagram shows the accuracy for multiple splits of data. The result of the evaluation is shown in Table 3. The above figures and curves show how accuracy and loss vary in a number of epochs.

Table 3: Result Comparison

Method	Model	Data Split	No. of Speaker	Accuracy
[2] CNN based speaker recognition in language and text-independent small scale system," 2019	CNN	Train: 80% Test: 20%	50	71%
	DNN	Train: 80% Test: 20%	50	61%
Proposed Method	CNN	Train: 70% Test: 30%	60	88.79%
	CNN	Train: 80% Test : 20%	60	90.03%
	CNN	Train: 90% Test : 10%	60	89.49%

Chapter 5: Conclusion

5.1 Conclusion

In this paper, the preprocessing techniques have been implemented for a deep learning-based speaker recognition system with MFCCs features. rVAD is implemented and applied to the dataset for robust voice activity detection and then extracted the MFCCs features to feed to the model. In this research, a three-layer CNN is used, with each convolutional layer containing batch-normalization and max-pooling. The output of the convolutional layer is then downsampled by a factor of two using Max Pooling. The third convolutional layer's output is flattened before being sent into a dense layer. Last but not least, a softmax output layer is included to generate the probability of each class label. Except for the last layer, which uses softmax,' all layers employ 'ReLu' as the activation function. Then train the model for 40 Epochs and test the model on the test split.

The Speaker recognition system was successfully implemented and got better results in terms of accuracy on the AudioMNIST Dataset of 60 different users. The result are compared with three data splits,For 70% train and 30% test split, accuracy is 88.79% and loss is 0.57% . For 80% train and 20% test split, accuracy is 90.03% and loss is 0.49% .For 90% train and 10% test split, accuracy is 89.49% and loss is 0.52%. The presence of irrelevant noise can degrade the accuracy of the system but our pre-processing steps help us overcome this problem.

5.2 Application

Speaker Recognition has many applications. It can be used for many real-life applications.

- Authentication
- Surveillance
- Speech Data Management

5.4 Challenges

- Accuracy
- Inconsistencies in the different types of audio
  and their quality
- Ability to efficiently isolate speakers

5.5 Future Work

There are a lot of thighs that can be added to this work and used to solve many real-life problems
.

- Work on the accuracy
- of Building Real-time System

References

[1] El-Moneim, Samia & El-Rabaie, El-Sayed & Nassar, Ma'En & Dessouky, M.I. & Ismail, Nabil & El-Fishawy, Adel & Abd El-Samie, Fathi. (2020). Speaker recognition based on pre-processing approaches. International Journal of Speech Technology.

[2] R. Jagiasi, S. Ghosalkar, P. Kulal and A. Bharambe, "CNN based speaker recognition in language and text-independent small scale system," 2019

[3] Bai, Zhongxin, and Xiao-Lei Zhang. "Speaker recognition based on deep learning: An overview." *Neural Networks* 140 (2021)

[4] D Anggraeni et al "The Implementation of Speech Recognition using Mel-Frequency Cepstrum Coefficients (MFCC) and Support Vector Machine (SVM) method based on Python to Control Robot Arm", 2018 IOP Conf. Ser.

[5] Kabir, Muhammad & Ph. D., M. & Shin, Jungpil & Jahan, Israt & Ohi, Abu. (2021)."A Survey of Speaker Recognition: Fundamental Theories, Recognition Methods and Opportunities". IEEE Access. PP. 1-1. 10.1109/ACCESS,2021

[6] M. Wang, T. Sirlapu, A. Kwasniewska, M. Szankin, M. Bartscherer, and R. Nicolas, "Speaker Recognition Using Convolutional Neural Network with Minimal Training Data," 2018 11th International Conference on Human System Interaction (HSI), 2018

[7] K. R. Farrell, R. J. Mammone and K. T. Assaleh, "Speaker recognition using neural networks and conventional classifiers," in IEEE Transactions on Speech and Audio Processing, vol. 2, no. 1, pp. 194-205, Jan. 1994

[8] Lei, Yun et al. "A novel scheme for speaker recognition using a phonetically-aware deep neural network." *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2014)

[9] Zheng-Hua Tan, Achintya kr. Sarkar, Najim Dehak,rVAD: An unsupervised segment-based robust voice activity detection method, Computer Speech & Language,Volume 59,2020,

[10] Bennani, Y., & Gallinari, P . (1994) . Connectionist approaches for automatic speaker recognition. In: Automatic Speaker Recognition, Identification and Verification

[11] X.-L. Zhang, J. Wu, Deep belief networks based voice activity detection, IEEE Transactions on Audio, Speech, and Language Processing 21 (4) (2013) 697–710.

[12]https://www.researchgate.net/publication/241686419_Speaker_Recognition_Advancements_and_Challenges

[13] Krˇcadinac, O.; Šoševi´c, U.; Starˇcevi´c, D. Evaluating the Performance of Speaker Recognition Solutions in E-Commerce Applications. Sensors 2021

[14] https://isl.anthropomatik.kit.edu/pdf/Beeking2020.pdf

[15] https://github.com/soerenab/AudioMNIST