DESIGN AND DEVELOPMENT OF DEEP-LEARNING ENABLED AUDIO SPOOF DETECTOR

MAIN PROJECT REPORT

Submitted by

GOURAV GOPAL	**1905015**
NALIN SURIYA S	**1905031**
VISHAL KARTHIK S	**1905060**
YOKESH R S	**1905062**

in partial fulfilment for the award of the degree of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

COIMBATORE INSTITUTE OF TECHNOLOGY

(Government Aided Autonomous Institution Affiliated to Anna University)

COIMBATORE – 641 014

ANNA UNIVERSITY – CHENNAI - 600 025

DECEMBER 2022

COIMBATORE INSTITUTE OF TECHNOLOGY

(A Govt. Aided Autonomous Institution Affiliated to Anna University)

COIMBATORE – 641 014

BONAFIDE CERTIFICATE

Certified that this project **"DESIGN AND DEVELOPMENT OF DEEP LEARNING ENABLED AUDIO SPOOF DETECTOR"** is the bonafide work of **GOURAV GOPAL (1905015), NALIN SURIYA S (1905031), VISHAL KARTHIK S (1905060), YOKESH R S (1905062)** under my supervision during the academic year 2022-2023.

SIGNATURE

Dr. G. KOUSALYA, M.E., Ph.D.,
HEAD OF THE DEPARTMENT
Department of CSE
Coimbatore Institute of Technology
Coimbatore – 641 014

SIGNATURE

Dr.M. MOHANAPRIYA, M.E., Ph.D
ASSOCIATE PROFESSOR
Department of CSE
Coimbatore Institute of Technology
Coimbatore – 641 014

Certified that the candidates were examined by us in the **19CS65 Main Project** viva voce examination held on ____________.

Internal Examiner **External Examiner**

Place :

Date :

ACKNOWLEDGEMENT

ACKNOWLEDGEMENT

We whole-heartedly thank the management of Coimbatore Institute of Technology for providing us the necessary infrastructure which was very much helpful to complete our project.

We heart fully thank our Secretary **Dr.R.Prabhakar B.Tech.,(IIT, Madras), M.S.(OSU.,USA), Ph.D.(Purdue, USA),** Advisor **Dr. V. Selladurai**, **M.E., Ph.D.,** and our principal **Dr.A.Rajeswari, M.E., Ph.D.,** for providing us with the necessary facilities which was useful in the completion of our project.

We sincerely thank the efforts taken by **Dr.Kousalya G, M.E., Ph.D.,** Professor and Head, Department of Computer Science and Engineering, Coimbatore Institute of Technology, for her valuable ideas in completing our project efficiently.

We also express our profound gratitude to our supervisor **Dr.M.Mohanapriya,M.E.,Ph.D** Associate Professor, Department of Computer Science and Engineering, Coimbatore Institute of Technology, for guiding us to carry out our project successfully.

We also express our sincere thanks to our Senior Tutor, Tutors and Project Coordinators, Department of Computer Science and Engineering, Coimbatore Institute of Technology, for guiding us all through the successful completion of our project.

We also extend our thanks to all our teaching faculty and non-teaching staff of our department for their kind attitude towards us all through our project work.

ABSTRACT

ABSTRACT

Authentication has become very important aspect of our day to day lives starting from normal lock screen pin to human retina based authentication systems. One among those popular and complex authentication systems are the audio based authentication systems where in people use certain words to unlock devices and objects like mobiles, doors etc., Audio Authentication generally involves authentication based on words and voices. Issue in the existing system is that the system verifies and extracts the features of words and voices but it does not classify human voice and recorded human voices. The above issue can be overcome by using models like RNN and LSTM for classification. The proposed system classifies bonafide and spoof using Bidirectional LSTM using MFCC feature extraction and obtains an accuracy of 85%.

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO
	ACKNOWLEDGEMENT	**iv**
	ABSTRACT	**vi**
1	**INTRODUCTION**	
	1.1 INTRODUCTION	**9**
	1.2 OBJECTIVE	**9**
	1.3 PROBLEM STATEMENT	**10**
	1.4 GOOGLE TREND ANALYSIS	**10**
2	**LITERATURE SURVEY**	**12**
3	**SYSTEM SPECIFICATION**	
	3.1 SOFTWARE SPECIFICATION	**21**
	3.2 HARDWARE SPECIFICATION	**21**
4	**SYSTEM ANALYSIS**	
	4.1 INTRODUCTION	**23**
	4.2 PROPOSED SYSTEM	**23**
	4.3 DATASET USED	**24**
	4.4 METHODOLOGIES USED	**24**
5	**SYSTEM IMPLEMENTATION**	
	5.1 INTRODUCTION	**29**
	5.2 MODULES	**30**
6	**RESULTS AND DISCUSSION**	**41**
7	**CORPORATE AND SOCIAL RESPONSIBILITY**	**43**
8	**CONCLUSION AND FUTURE WORK**	**45**
9	**REFERENCES**	**47**

INTRODUCTION

CHAPTER 1

INTRODUCTION

1.1. INTRODUCTION

Development of VCD’s, have boosted the realization of smart homes, voice-controlled authentication systems etc.,.These VCDs are vulnerable to different spoofing attacks .Audio Authentication is becoming very essential part of our lives. Audio Authentication spoofing is becoming an issue. High-quality audio recorders enable bypassing this audio authentication system by just recording the human voice and reusing them for accessing the same system. Thus, there exists a need to develop a voice anti-spoofing framework capable of detecting multiple audio spoofing attacks.

1.2. OBJECTIVE

1. To develop a Deep-Learning Enabled Audio Spoof Detector which uses Recurrent Neural Networks(RNN) and Long Short Term Memory(LSTM) to classify human voice from recorded voices.

2. The proposed model will defend from the following attacks:
   a) Replay attack
   b) Voice conversion attack

1.3 PROBLEM STATEMENT

The increase in the advancement of audio editing softwares , provide an easy way to the accessibility of voice controlled authentication systems which makes the Voice controlled devices(VCD) vulnerable for audio spoof attacks.

Audio spoof attack is the manipulation of genuine signal through recording or modifying to trick an audio verification system.

1.4 GOOGLE TREND ANALYSIS

The diagram shown below displays the google trend analysis about the domain of the project.

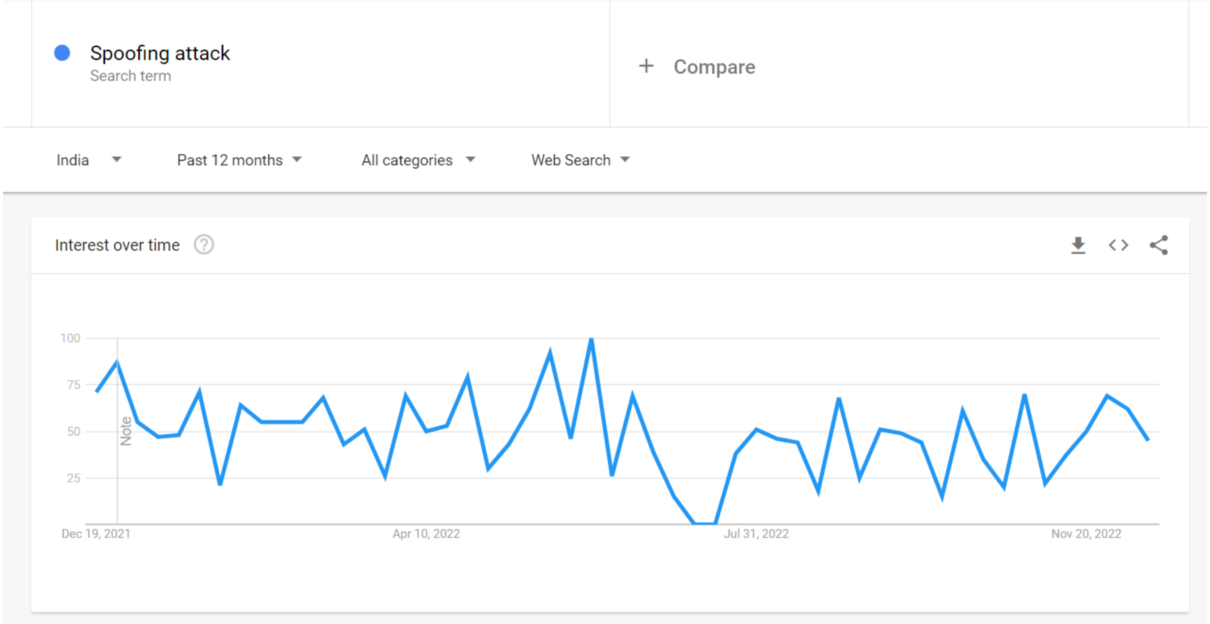


Figure 1.1 Google Trend analysis of spoof attacks over the time.

LITERATURE SURVEY

CHAPTER 2

LITERATURE SURVEY

2.1 Fusion of BiLSTM and GMM-UBM Systems for Audio Spoofing Detection (2019)

Authors: Ivan Rakhmanenko , Alexander Shelupanov , Evgeny Kostyunchenko

Description:

In this study, Bidirectional Long Short Term Memory (BiLSTM) networks with constant Q cepstral coefficients (CQCC) are used to classify real audio from fake audio in anti-spoofing systems.

By fusing the BiLSTM and GMM-UBM systems, a fusion mechanism is used to increase the variability of the systems' decision-making processes and their accuracy.

Over the baseline systems, these proposed systems significantly improved performance.

Advantages:

- Uses time domain dependency relation in audio and gives better results.
- Uses fusion mechanism to improve accuracy.

Disadvantages

- It doesn’t provide classification model for classification of various attacks.

2.2 Improving anti-spoofing with octave spectrum and short-term spectral statistics information (2020)

Authors: Jichen Yang , Rohan Kumar Das

Description:

In this paper, feature level exploration is done and a new feature is formed that can capture improved discriminative information between real and fake audio from that of the existing features that are already explored.

Similar to most of the systems being used nowadays, the spoofing detection system used for this work contains a front end feature extractor and a back-end classifier.

The feature extractor extracts effective information and whereas the classifier identifies whether an audio is genuine or spoofed.

Advantages:

- The systems combines linear power spectrum and Octave power spectrum to produce better to form better novel feature eCQCC indeed produces good results.

- Works for Synthetic and Playback Spoof attacks.

Disadvantages:

- Does not prove to be much effective in classification of audio spoofed data.

2.3 Spoofed Speech Detection with Weighted Phase Features and Convolutional Networks (2022)

Authors: Gökay DISKEN

Description:

In this paper, features are extracted using a frame-wise weighted magnitude spectrum and these are found to be effective on replay attacks. Along with frame-wise weighted spectrum, a cosine normalized phased spectrum is used to attain better performance as phase-based features have shown decent performance for this particular task.

These extracted features are then sent to the Convolutional Neural Network as input and then are classified as either synthetic or replay attacks.

Advantages:

- Frame –wise weighted magnitude spectrum results in more effective method for spoof detection.
- Discusses various Attacks like synthetic, replay attacks and S10.

Disadvantages:

- Takes much computational cost for training.

2.4 Detection of Various Speech Forgery Operations Based on Recurrent Neural Network (2020)

Author: Diqun Yan and Tingting Wu.

Description:

In this paper, feature extraction methods like LFCC and MFCC are used to extract audio features and then these are sent as input to the Recurrent Neural Network (RNN) frame with two-layer LSTM to detect four common audio forgery operations.

These are experimented mainly on TIMIT and UME databases and various evaluations like intra-database evaluation as well as cross-database evaluation are done and the detection accuracies of each of the above are identified.

Advantages:

- Feature extraction techniques like MFCC and LFCC are used to extract audio features.
- In this work, RNN is used as it can capture the correlation between the frames in a speech recognition application.
- Hence it is considered better than CNN as it does not capture the sequential correlation well.

Disadvantages:

- The cross-database evaluation accuracy could be improved in this model.

2.5 Fake Audio Speech Detection (2020)

Author: Shilpa Lunagaria, Mr. Chandresh Parekh

Description:

In this paper, deep fake audio forgery is identified using Deep Learning algorithms. Audio files are taken as input and model is trained to uniquely identify features for voice creation and voice detection. The model could then classify between whether the audio is real or fake.

The accuracy obtained for this model during training and validation phases are pretty high but the testing accuracy could be improved more by extracting more features and using different algorithms.

Advantages:

- The real and fake voices can be identified.
- The accuracies obtained for training, validation are considerably high (99%, 95% respectively).

Disadvantages:

- This work only focuses on deep fake audio forgery and it doesn't detect or identify other audio forgery operations.

- The testing accuracy in this model could be improved with better algorithms (just 85% accuracy).

2.6 Deep Learning based DFWF Model for Audio Spoofing Attack Detection (2020)

Author: Kottilingam Kottursamy

Description:

In this paper, fake audios are detected without continual-learning based methods and fakes are also detected without forgetting so as to develop a model that can detect spoofing attacks in an incremental fashion.

To retain the original model memory, a method known as knowledge distillation is introduced. This model using Deep Learning techniques reduces the error rate up to 80%.

Advantages:

- Overcomes the demerit of loss of previous data by introduction of knowledge distillation loss.

Disadvantages:

- Poor performance in detection of unseen data.
- Requires both original and new data.
- Storage of data.

2.7 Voice spoofing countermeasure for voice replay attacks using deep learning. (2022)

Author: Jincheng Zhou, Tao Hai1, Dayang N. A. Jawawi, Dan Wang, Ebuka Ibeke and Cresantus Biamba

Description:

The paper discusses about the immense usage of Automatic Speaker Verification (ASV) system which verifies users with their voices and it's susceptibility to voice spoofing attacks - logical and physical access attacks. A secured voice spoofing countermeasure to detect voice replay attacks is proposed. This has enhanced the ASV system security by building a spoofing countermeasure dependent on the decomposed signals that consist of prominent information. It uses two main features— the Gammatone Cepstral Coefficients (GCC) and Mel-Frequency Cepstral Coefficients (MFCC) — for the audio representation. For the classification of the features, Bi-directional Long-Short Term Memory Network in the cloud, a deep learning classifier.

Numerous audio features and respective feature's capability to obtain the most vital details from the audio for it to be labelled genuine or a spoof speech is examined. Furthermore, it uses various machine learning algorithms to illustrate the superiority of the system compared to the traditional classifiers. The results of the experiments were classified according to the parameters of accuracy, precision rate, recall, F1-score, and Equal Error Rate (EER). The results were 97%, 100%, 90.19% and 94.84%, and 2.95%, respectively.

Advantages:

- Avoids replay attacks in ASV.
- Voice biometrics.
- More accurate with the method of Speech Decomposition.

Disadvantages:

- Does not discuss about many audio spoof attacks, focuses on Replay attacks only.

2.8 LSTM and CNN based ensemble approach for spoof detection task in automatic speaker verification systems (2022)

Author: Mohit Dua, Chhavi Jain, Sushil Kumar

Description:

In this paper , the system that is proposed tries to address the problem of classifying legitimate user and the malicious attacks using deep learning (DL) methods and ensemble of different neural networks. The first model that is discussed is a combination of time-distributed dense layers and long short-term memory (LSTM) layers.

The other two deep neural networks (DNNs) are based on temporal convolution (TC) and spatial convolution (SC). Finally, an ensemble model comprising of these three DNNs has also been analysed. All these models are analysed with Mel frequency cepstral coefficients (MFCC), inverse Mel frequency cepstral coefficients (IMFCC) and constant Q cepstral coefficients (CQCC) at the frontend, where the proposed ensemble performs best with CQCC features.

The proposed work uses ASVspoof 2015 and ASVspoof 2019 datasets for training and testing, with the evaluation set having speech synthesis (SS) and voice conversion (VC) attacked utterances. Performance of proposed system trained with ASVspoof 2015 dataset degrades with evaluation set of ASVspoof 2019 dataset, whereas performance of the same system improves when training is also done with the ASVspoof 2019 dataset.

Advantages:

- LSTM models are used which increases performance.
- Ensemble method is used to get a consolidated decision from models.

Disadvantages:

- Performance on combined dataset is low.
- Deep CNN can be used to improve performance.

SYSTEM SPECIFICATION

CHAPTER 3

SYSTEM SPECIFICATION

3.1. SOFTWARE SPECIFICATION

Operating system: Windows 10

IDE: Google Colaboratory / Jupyter Notebook

Coding Language: Python

3.2. HARDWARE SPECIFICATION

Processor: Intel core i5

Hard disk: 512GB

RAM: 8GB

SYSTEM ANALYSIS

CHAPTER 4

SYSTEM ANALYSIS

4.1. INTRODUCTION

Development of Voice Control Devices(VCDs), have boosted the realization of smart homes, voice-controlled authentication systems etc.,.These VCDs are vulnerable to different spoofing attacks .Audio Authentication is becoming very essential part of our lives. Audio Authentication spoofing is becoming an issue. High-quality audio recorders enables bypassing this audio authentication system by just recording the human voice and reusing them for accessing the same system. Thus, there exists a need to develop a voice anti-spoofing framework capable of detecting multiple audio spoofing attacks.

4.2. PROPOSED SYSTEM

The proposed system uses a Bidirectional LSTM model to classify the audio samples. Features fed into bidirectional LSTM model are generated using MFCC methodologies.

The proposed system performs binary classification of audio data which are mapped to two classes

a) Authentic/Bonafide
b) Spoofed

4.3. DATASET USED.

The Dataset used in the model is Automatic Speaker Verification(ASV) 2019.

ASV Spoof dataset contains two types of Audio files

- Physical Access - Bonafide utterances are made in a real, physical space in which spoofing attacks are captured and then replayed within the same physical space using replay devices of varying quality.
- Logical Access - Bonafide and spoofed utterances generated using text-to-speech (TTS) and voice conversion (VC) algorithms are communicated across telephony and VoIP networks with various coding and transmission effects

The dataset includes genuine and spoofed speech from 20 speakers (8 male, 12 female).Each spoofed utterance is generated according to one of 2 voice conversion and 3 speech synthesis algorithms.

The voice conversion systems include those based on (i) neural-network-based and (ii)transfer-function-based methods.

The speech synthesis systems were implemented with (i) waveform concatenation, (ii) neural-network-based parametric speech synthesis using source-filter vocoders and (iii) neural-network-based parametric speech synthesis using Wavenet.

4.4 METHODOLGIES USED.

Mel Frequency Cepstral Coefficients(MFCCs) :

In general for any audio based task the raw audio signal cannot be given to the model as input because there will be a lot of noise in the audio signal. It is observed that extracting features from the audio signal and using it as input to the base model will produce much better performance than directly considering raw audio signal as input. MFCC is the widely used technique for extracting the features from the audio signal.

Mel-frequency cepstral coefficients (MFCC) which have 39 features. The feature count is small enough to force us to learn the information of the audio. 12 parameters are related to the amplitude of frequencies. It provides us enough frequency channels to analyze the audio.

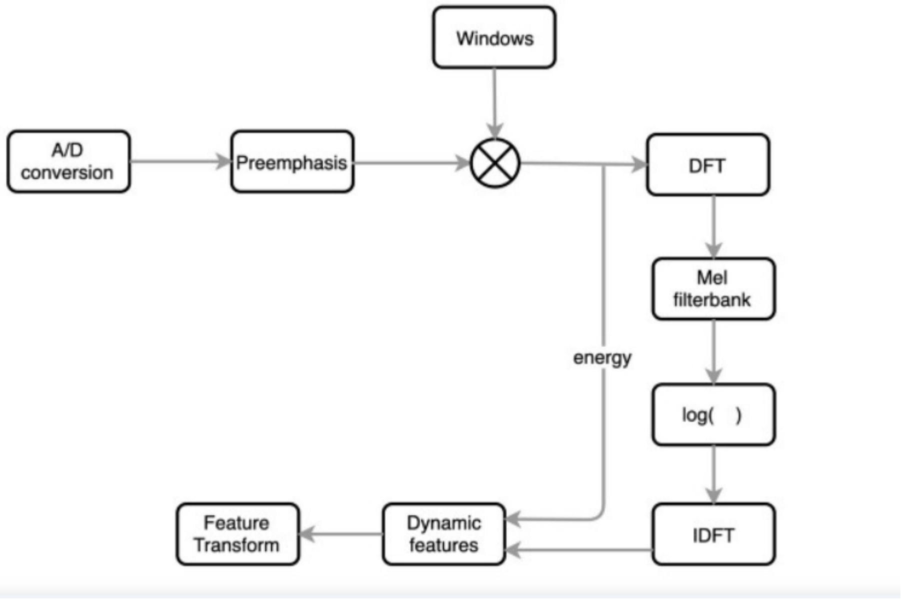


Figure 4.1 MFCC feature extraction flow chart

Long Short Term Memory Network:

Long Short Term Memory networks(LSTM) – are a special kind of RNN, capable of learning long-term dependencies.

LSTMs provides a large range of parameters such as learning rates, and input and output biases. The complexity to update each weight is reduced to O(1) with LSTMs, similar to that of Back Propagation Through Time (BPTT), which is an advantage.A Long Short Term Memory Network consists of four different gates for different purposes as described below:-

a) Forget Gate(f): It determines to what extent to forget the previous data.

b) Input Gate(i): It determines the extent of information be written onto the Internal Cell State.

c) Input Modulation Gate(g): It is often considered as a sub-part of the input gate and much literature on LSTM's does not even mention it and assume it is inside the Input gate. It is used to modulate the information that the Input gate will write onto the Internal State Cell by adding non-linearity to the information and making the information Zero-mean. This is done to reduce the learning time as Zero-mean input has faster convergence. Although this gate's actions are less important than the others and are often treated as a finesse-providing concept, it is good practice to include this gate in the structure of the LSTM unit.

d) Output Gate(o): It determines what output(next Hidden State) to generate from the current Internal Cell State.

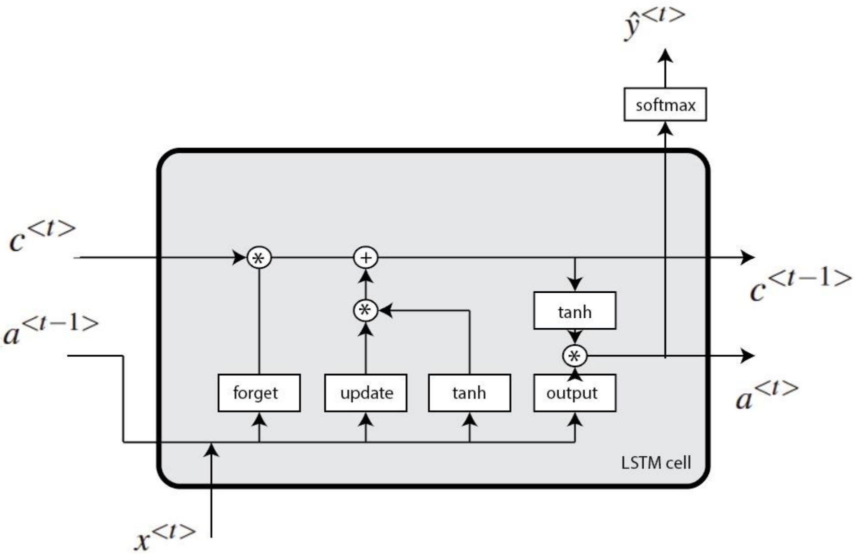


Figure 4.2 LSTM architecture

Bi-directional Long Short Term Memory Network:

Bidirectional LSTM (BiLSTM) is a recurrent neural network used primarily on natural language processing. Unlike standard LSTM, the input flows in both directions, and it's capable of utilizing information from both sides. It's also a powerful tool for modeling the sequential dependencies between words and phrases in both directions of the sequence.

BiLSTM adds one more LSTM layer, which reverses the direction of information flow. Briefly, it means that the input sequence flows backward in the additional LSTM layer. Then the outputs from both LSTM layers are combined in several ways, such as average, sum, multiplication, or concatenation.

To illustrate, the unrolled BiLSTM is presented in the figure below:

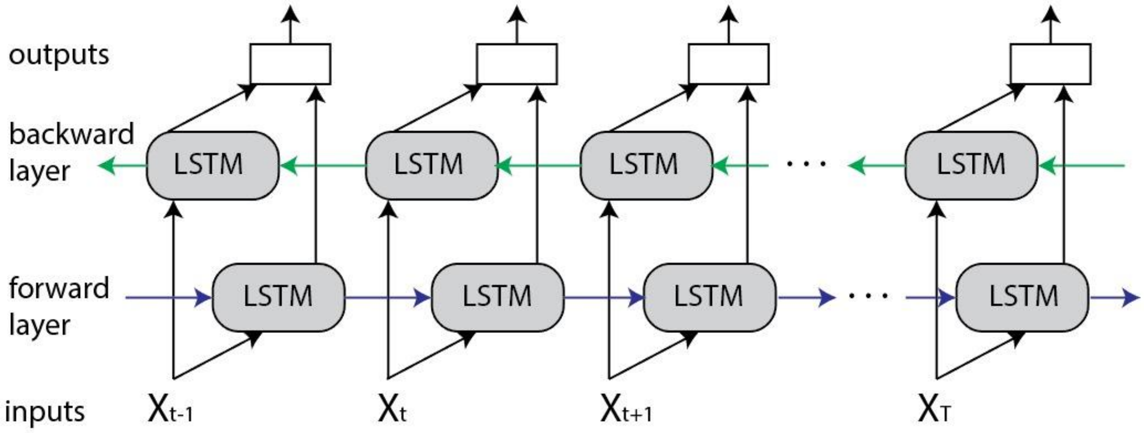


Figure 4.3 Bi-directional LSTM architecture

Advantages of Bidirectional LSTM

This type of architecture has many advantages in real-world problems, especially in NLP. The main reason is that every component of an input sequence has information from both the past and present. For this reason, BiLSTM can produce a more meaningful output, combining LSTM layers from both directions.

BiLSTM will have a different output for every component (word) of the sequence (sentence). As a result, the BiLSTM model is beneficial in some NLP tasks, such as sentence classification, translation, and entity recognition. In addition, it finds its applications in speech recognition, protein structure prediction, handwritten recognition, and similar fields.

SYSTEM IMPLEMENTATION

CHAPTER 5

SYSTEM IMPLEMENTATION

5.1 INTRODUCTION

The system is designed to take audio as input from the user and identifies the authenticity of the voice using the proposed LSTM model and then classifies it into bonafide class or spoofed voice class. On identification of spoofed voice it denies the access to the authentication system.

The system uses the Mel Frequency Cepstral Coefficients(MFCCs) for feature extraction and feed it to the LSTM model for classification.

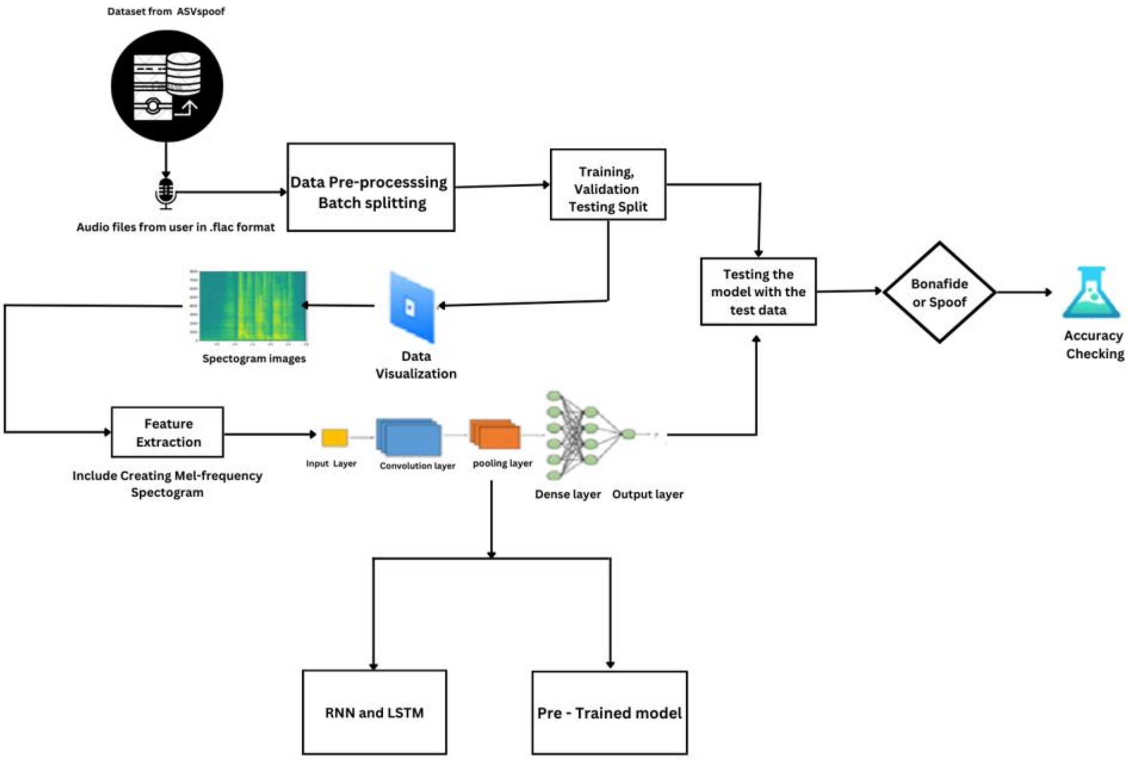


Figure 5.1 Architecture of the Proposed System

5.2 MODULES

The modules used in the system are

i. Import Dataset and Data Pre-processing.

ii. Data visualization.

iii. Feature extraction

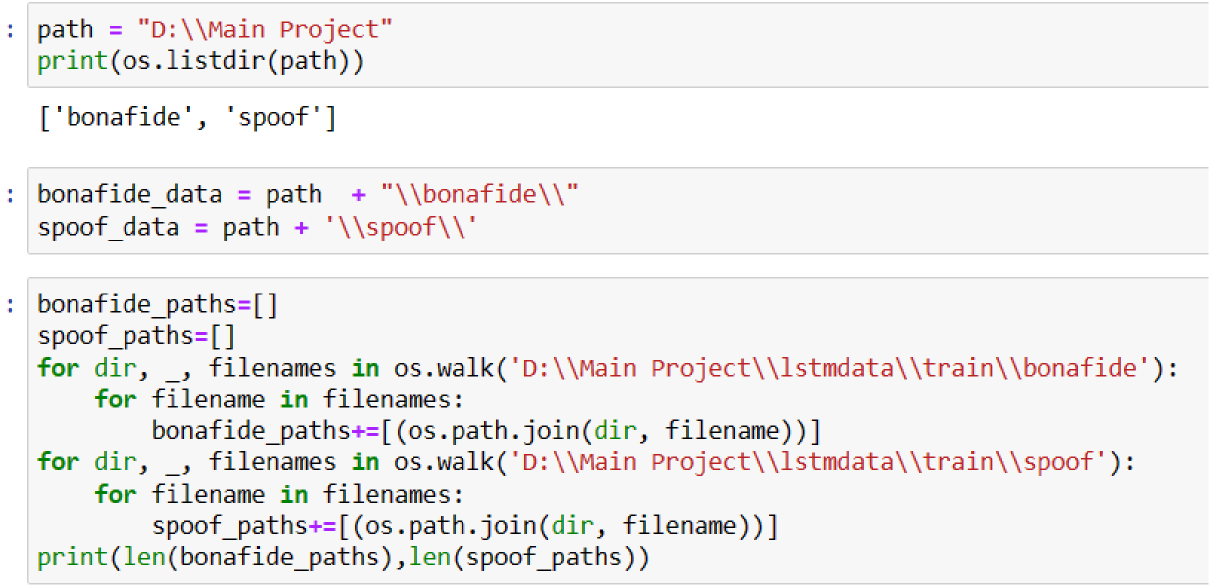
iv. Training LSTM model with the abstracted features.

v. Validating the model.

vi. Import the prepared model and provide the audio input for classification.

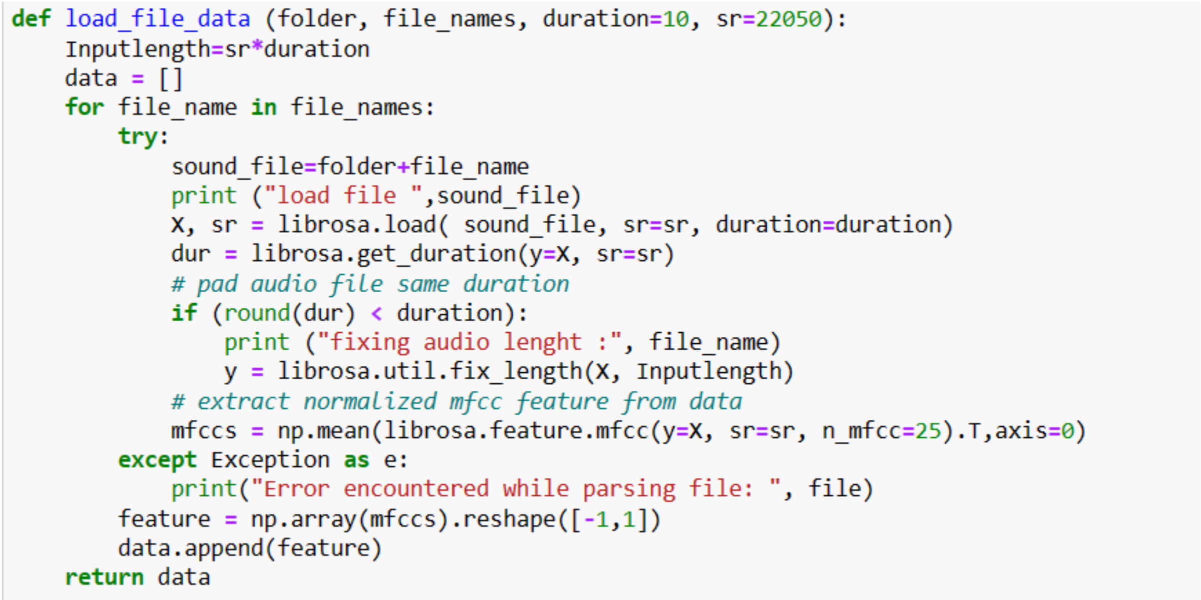
5.2.1 IMPORTING DATASET AND DATA PREPROCESSING

Importing Dataset:



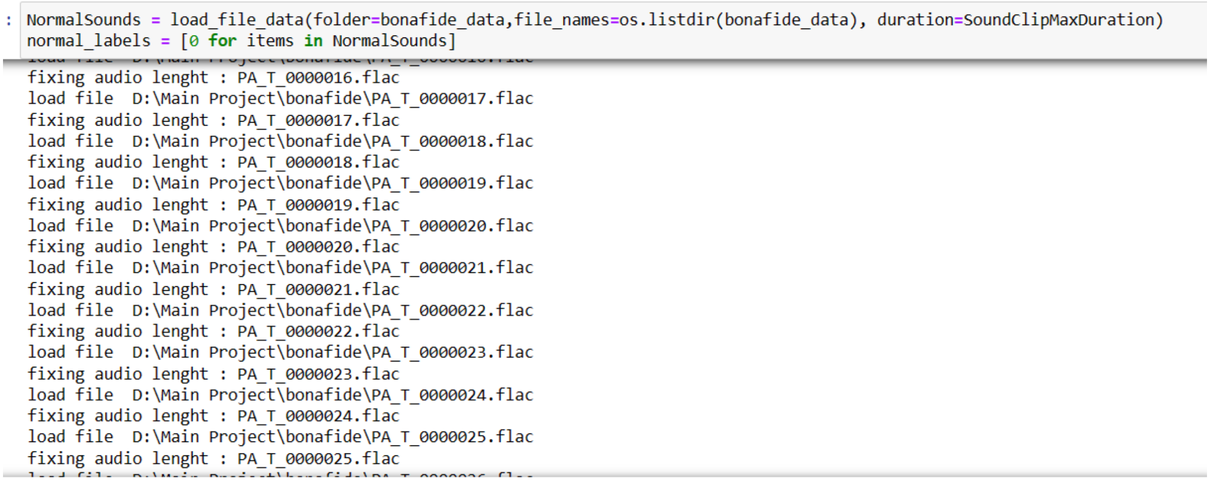
```
: path = "D:\\Main Project"
  print(os.listdir(path))

  ['bonafide', 'spoof']

: bonafide_data = path  + "\\bonafide\\"
  spoof_data = path + '\\spoof\\'

: bonafide_paths=[]
  spoof_paths=[]
  for dir, _, filenames in os.walk('D:\\Main Project\\lstmdata\\train\\bonafide'):
      for filename in filenames:
          bonafide_paths+=[(os.path.join(dir, filename))]
  for dir, _, filenames in os.walk('D:\\Main Project\\lstmdata\\train\\spoof'):
      for filename in filenames:
          spoof_paths+=[(os.path.join(dir, filename))]
  print(len(bonafide_paths),len(spoof_paths))
```

Figure 5.2 Importing dataset from the file directory.



```python
def load_file_data (folder, file_names, duration=10, sr=22050):
    Inputlength=sr*duration
    data = []
    for file_name in file_names:
        try:
            sound_file=folder+file_name
            print ("load file ",sound_file)
            X, sr = librosa.load( sound_file, sr=sr, duration=duration)
            dur = librosa.get_duration(y=X, sr=sr)
            # pad audio file same duration
            if (round(dur) < duration):
                print ("fixing audio lenght :", file_name)
                y = librosa.util.fix_length(X, Inputlength)
            # extract normalized mfcc feature from data
            mfccs = np.mean(librosa.feature.mfcc(y=X, sr=sr, n_mfcc=25).T,axis=0)
        except Exception as e:
            print("Error encountered while parsing file: ", file)
        feature = np.array(mfccs).reshape([-1,1])
        data.append(feature)
    return data
```

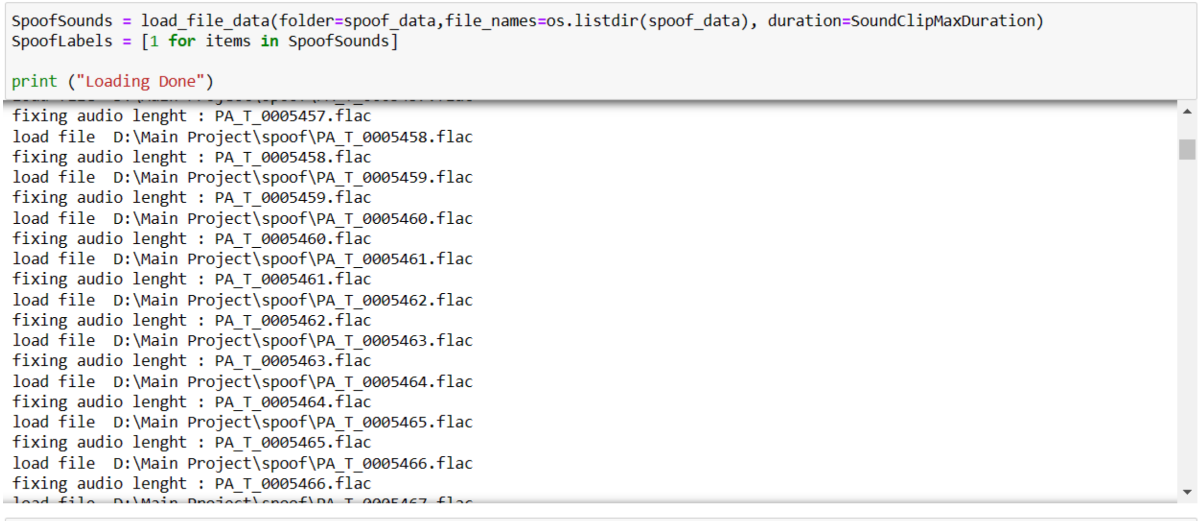
Figure 5.3 Function to load data.



```python
NormalSounds = load_file_data(folder=bonafide_data,file_names=os.listdir(bonafide_data), duration=SoundClipMaxDuration)
normal_labels = [0 for items in NormalSounds]
```

```
fixing audio lenght : PA_T_0000016.flac
load file  D:\Main Project\bonafide\PA_T_0000017.flac
fixing audio lenght : PA_T_0000017.flac
load file  D:\Main Project\bonafide\PA_T_0000018.flac
fixing audio lenght : PA_T_0000018.flac
load file  D:\Main Project\bonafide\PA_T_0000019.flac
fixing audio lenght : PA_T_0000019.flac
load file  D:\Main Project\bonafide\PA_T_0000020.flac
fixing audio lenght : PA_T_0000020.flac
load file  D:\Main Project\bonafide\PA_T_0000021.flac
fixing audio lenght : PA_T_0000021.flac
load file  D:\Main Project\bonafide\PA_T_0000022.flac
fixing audio lenght : PA_T_0000022.flac
load file  D:\Main Project\bonafide\PA_T_0000023.flac
fixing audio lenght : PA_T_0000023.flac
load file  D:\Main Project\bonafide\PA_T_0000024.flac
fixing audio lenght : PA_T_0000024.flac
load file  D:\Main Project\bonafide\PA_T_0000025.flac
fixing audio lenght : PA_T_0000025.flac
```

Figure 5.4 Loading Normal or Genuine Audio file in .flac format.

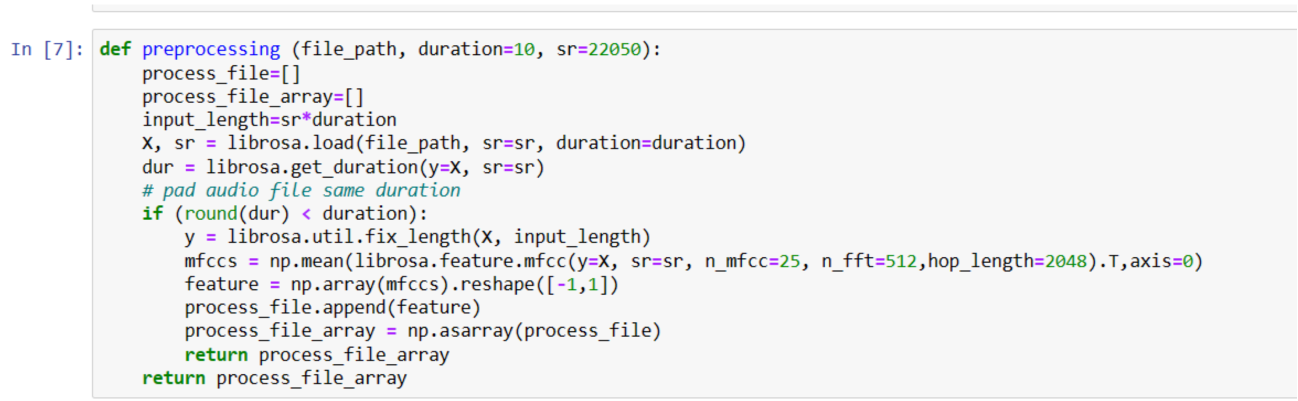


```python
SpoofSounds = load_file_data(folder=spoof_data,file_names=os.listdir(spoof_data), duration=SoundClipMaxDuration)
SpoofLabels = [1 for items in SpoofSounds]

print ("Loading Done")
```

```
fixing audio lenght : PA_T_0005457.flac
load file  D:\Main Project\spoof\PA_T_0005458.flac
fixing audio lenght : PA_T_0005458.flac
load file  D:\Main Project\spoof\PA_T_0005459.flac
fixing audio lenght : PA_T_0005459.flac
load file  D:\Main Project\spoof\PA_T_0005460.flac
fixing audio lenght : PA_T_0005460.flac
load file  D:\Main Project\spoof\PA_T_0005461.flac
fixing audio lenght : PA_T_0005461.flac
load file  D:\Main Project\spoof\PA_T_0005462.flac
fixing audio lenght : PA_T_0005462.flac
load file  D:\Main Project\spoof\PA_T_0005463.flac
fixing audio lenght : PA_T_0005463.flac
load file  D:\Main Project\spoof\PA_T_0005464.flac
fixing audio lenght : PA_T_0005464.flac
load file  D:\Main Project\spoof\PA_T_0005465.flac
fixing audio lenght : PA_T_0005465.flac
load file  D:\Main Project\spoof\PA_T_0005466.flac
fixing audio lenght : PA_T_0005466.flac
```

Figure 5.5 Loading Spoof Audio file in .flac format.

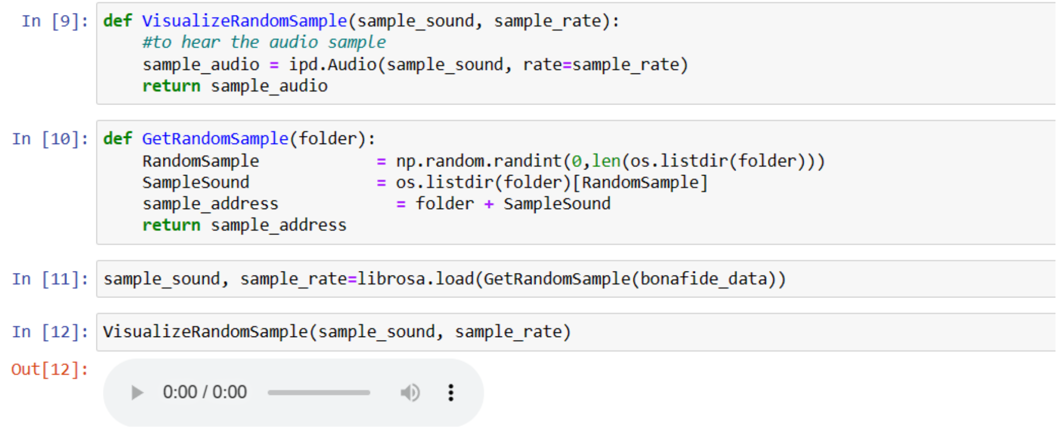
Data Preprocessing :



```
In [7]: def preprocessing (file_path, duration=10, sr=22050):
            process_file=[]
            process_file_array=[]
            input_length=sr*duration
            X, sr = librosa.load(file_path, sr=sr, duration=duration)
            dur = librosa.get_duration(y=X, sr=sr)
            # pad audio file same duration
            if (round(dur) < duration):
                y = librosa.util.fix_length(X, input_length)
                mfccs = np.mean(librosa.feature.mfcc(y=X, sr=sr, n_mfcc=25, n_fft=512,hop_length=2048).T,axis=0)
                feature = np.array(mfccs).reshape([-1,1])
                process_file.append(feature)
                process_file_array = np.asarray(process_file)
                return process_file_array
            return process_file_array
```

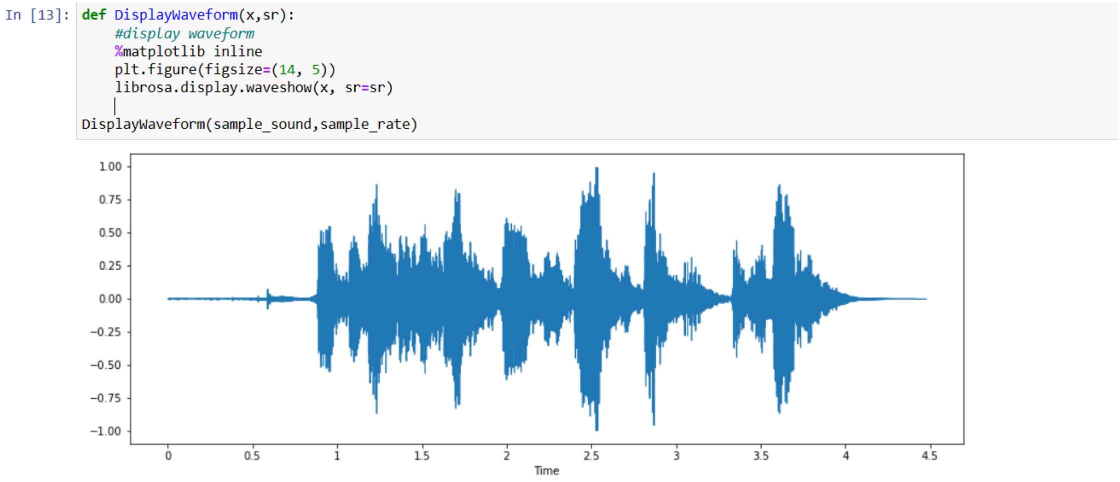
Figure 5.6 Audio File preprocessing.

5.2.2 DATA VISUALIZATION



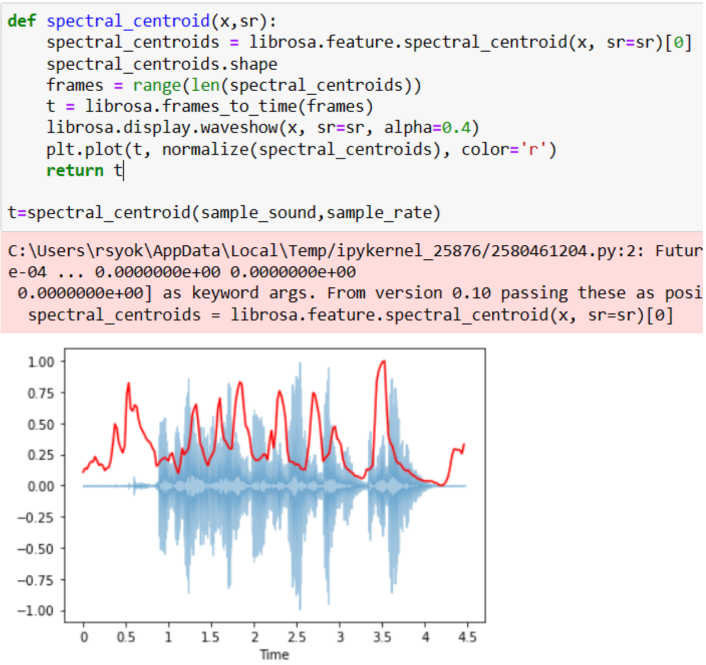
```
In [9]: def VisualizeRandomSample(sample_sound, sample_rate):
            #to hear the audio sample
            sample_audio = ipd.Audio(sample_sound, rate=sample_rate)
            return sample_audio

In [10]: def GetRandomSample(folder):
             RandomSample            = np.random.randint(0,len(os.listdir(folder)))
             SampleSound             = os.listdir(folder)[RandomSample]
             sample_address            = folder + SampleSound
             return sample_address

In [11]: sample_sound, sample_rate=librosa.load(GetRandomSample(bonafide_data))

In [12]: VisualizeRandomSample(sample_sound, sample_rate)

Out[12]:
```

Figure 5.7 Visualizing the Audio Sample using ipd package.



```
In [13]: def DisplayWaveform(x,sr):
             #display waveform
             %matplotlib inline
             plt.figure(figsize=(14, 5))
             librosa.display.waveshow(x, sr=sr)

         DisplayWaveform(sample_sound,sample_rate)
```

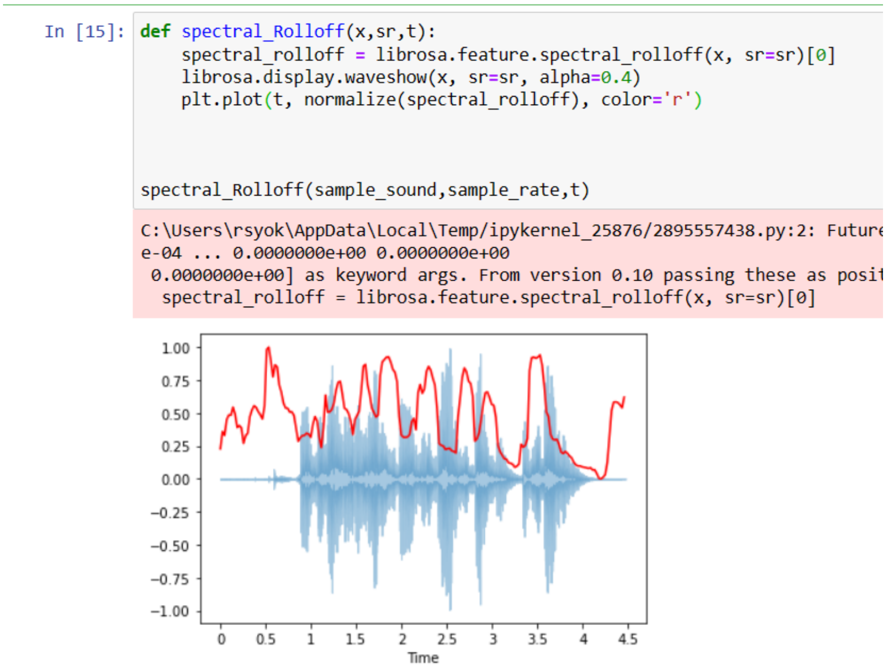
Figure 5.8 Waveform of a audio file.



```
def spectral_centroid(x,sr):
    spectral_centroids = librosa.feature.spectral_centroid(x, sr=sr)[0]
    spectral_centroids.shape
    frames = range(len(spectral_centroids))
    t = librosa.frames_to_time(frames)
    librosa.display.waveshow(x, sr=sr, alpha=0.4)
    plt.plot(t, normalize(spectral_centroids), color='r')
    return t

t=spectral_centroid(sample_sound,sample_rate)
```

```
C:\Users\rsyok\AppData\Local\Temp/ipykernel_25876/2580461204.py:2: Futur
e-04 ... 0.0000000e+00 0.0000000e+00
 0.0000000e+00] as keyword args. From version 0.10 passing these as posi
  spectral_centroids = librosa.feature.spectral_centroid(x, sr=sr)[0]
```

Figure 5.9 Spectral Centroid representation of audio file.

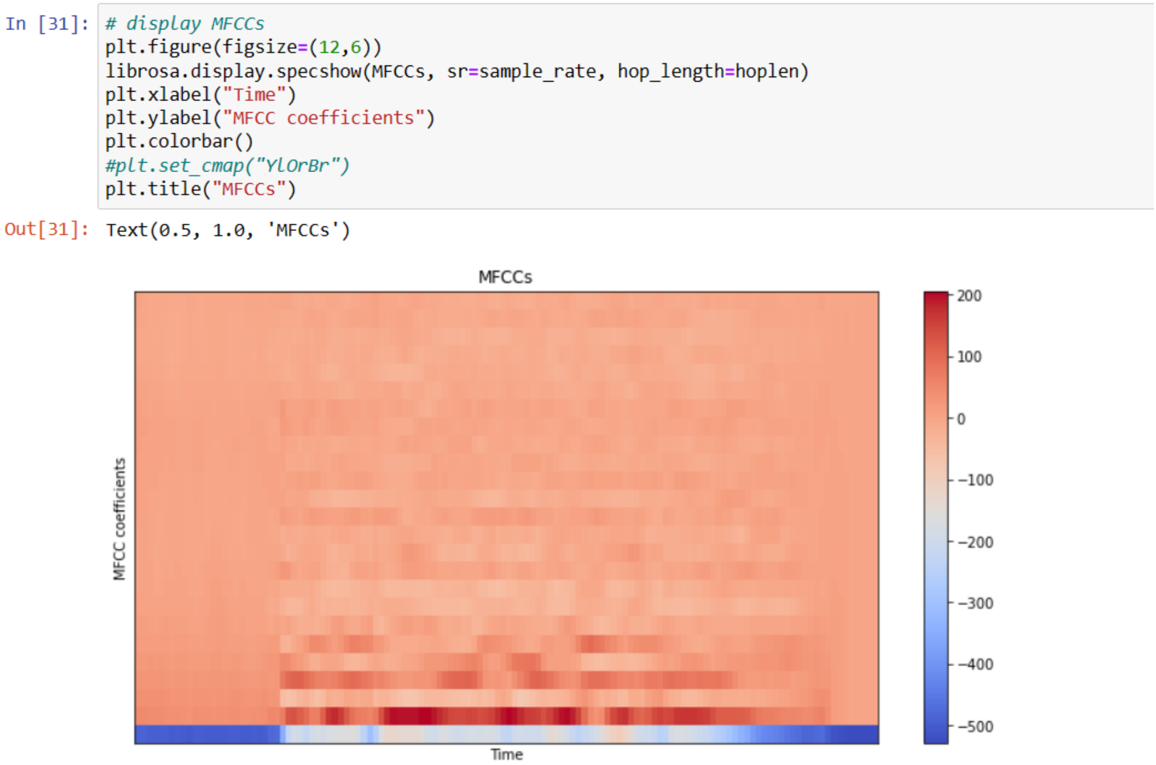


```
In [15]: def spectral_Rolloff(x,sr,t):
             spectral_rolloff = librosa.feature.spectral_rolloff(x, sr=sr)[0]
             librosa.display.waveshow(x, sr=sr, alpha=0.4)
             plt.plot(t, normalize(spectral_rolloff), color='r')



         spectral_Rolloff(sample_sound,sample_rate,t)
```

```
C:\Users\rsyok\AppData\Local\Temp/ipykernel_25876/2895557438.py:2: Future
e-04 ... 0.0000000e+00 0.0000000e+00
 0.0000000e+00] as keyword args. From version 0.10 passing these as posit
  spectral_rolloff = librosa.feature.spectral_rolloff(x, sr=sr)[0]
```

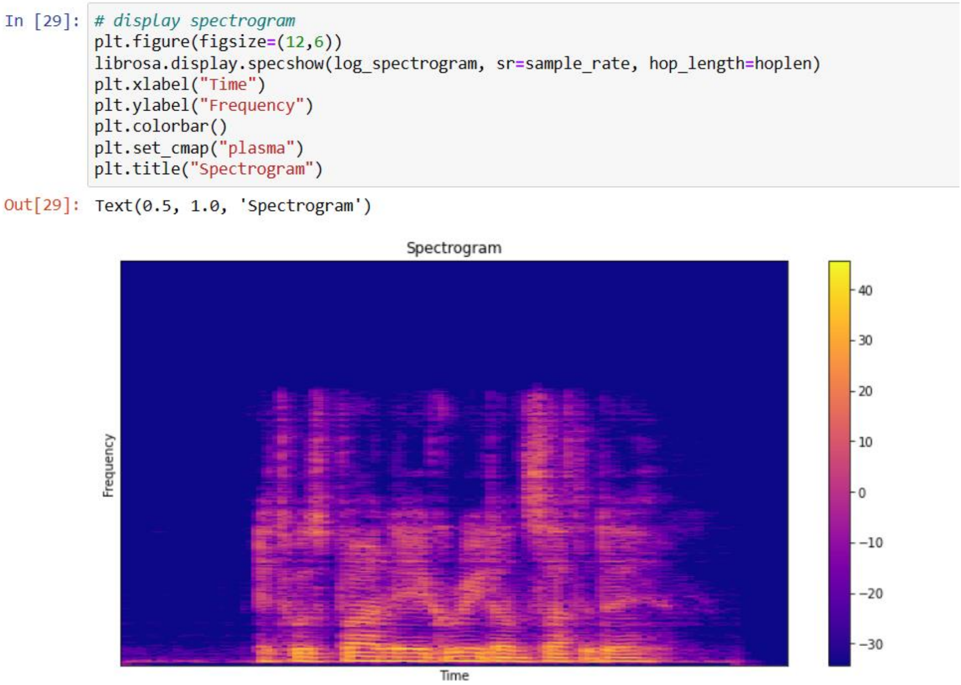
Figure 5.10 Spectral Rolloff representation of audio file.

5.2.3 FEATURE EXTRACTION



```
In [31]: # display MFCCs
         plt.figure(figsize=(12,6))
         librosa.display.specshow(MFCCs, sr=sample_rate, hop_length=hoplen)
         plt.xlabel("Time")
         plt.ylabel("MFCC coefficients")
         plt.colorbar()
         #plt.set_cmap("YlOrBr")
         plt.title("MFCCs")

Out[31]: Text(0.5, 1.0, 'MFCCs')
```

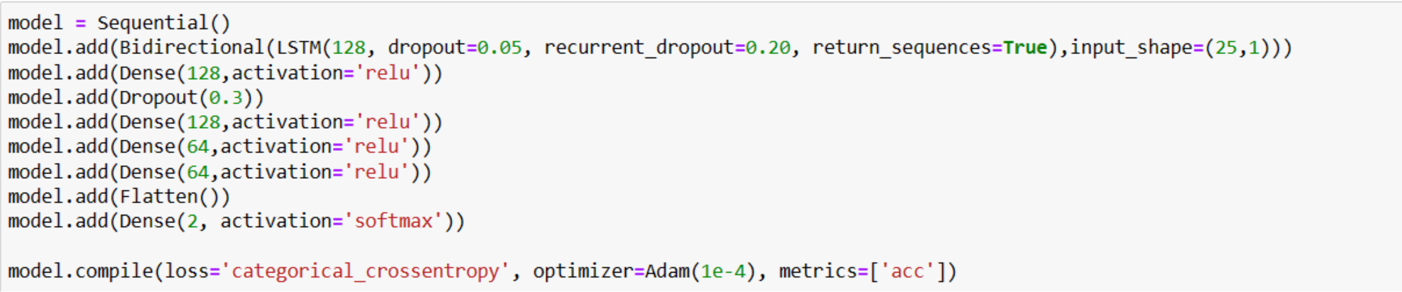
Figure 5.11 MFCC feature extraction from audio file using librosa package



```
In [29]: # display spectrogram
         plt.figure(figsize=(12,6))
         librosa.display.specshow(log_spectrogram, sr=sample_rate, hop_length=hoplen)
         plt.xlabel("Time")
         plt.ylabel("Frequency")
         plt.colorbar()
         plt.set_cmap("plasma")
         plt.title("Spectrogram")

Out[29]: Text(0.5, 1.0, 'Spectrogram')
```

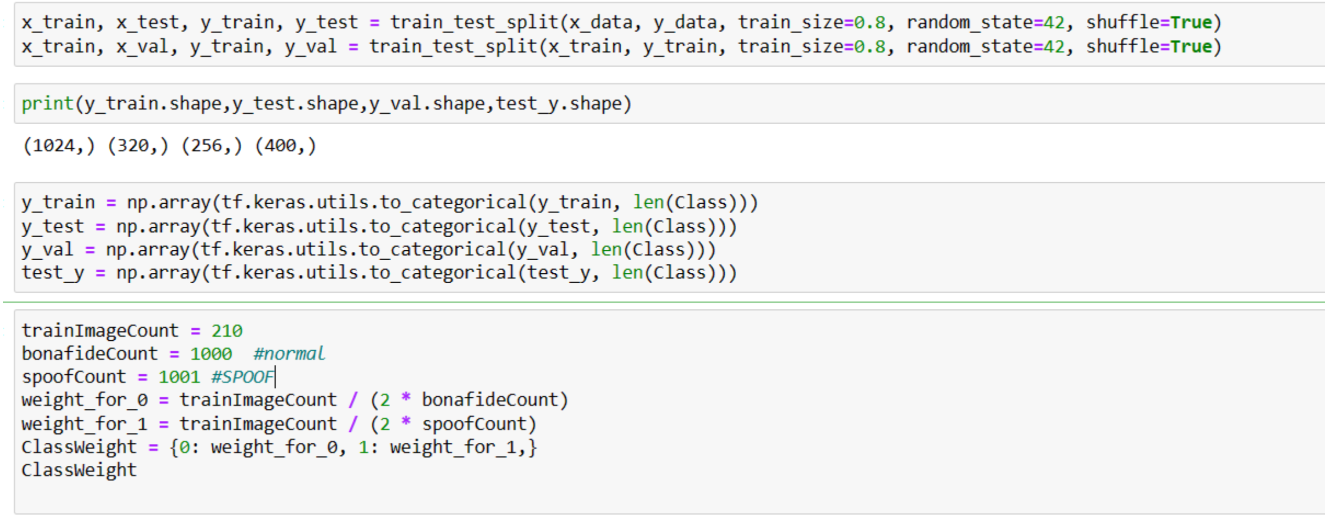
Figure 5.12 Spectrogram visualization

5.2.4 TRAINING LSTM MODEL USING THE ABSTRACTED FEATURES.



```
model = Sequential()
model.add(Bidirectional(LSTM(128, dropout=0.05, recurrent_dropout=0.20, return_sequences=True),input_shape=(25,1)))
model.add(Dense(128,activation='relu'))
model.add(Dropout(0.3))
model.add(Dense(128,activation='relu'))
model.add(Dense(64,activation='relu'))
model.add(Dense(64,activation='relu'))
model.add(Flatten())
model.add(Dense(2, activation='softmax'))

model.compile(loss='categorical_crossentropy', optimizer=Adam(1e-4), metrics=['acc'])
```

Figure 5.13 Model building



```
x_train, x_test, y_train, y_test = train_test_split(x_data, y_data, train_size=0.8, random_state=42, shuffle=True)
x_train, x_val, y_train, y_val = train_test_split(x_train, y_train, train_size=0.8, random_state=42, shuffle=True)
```

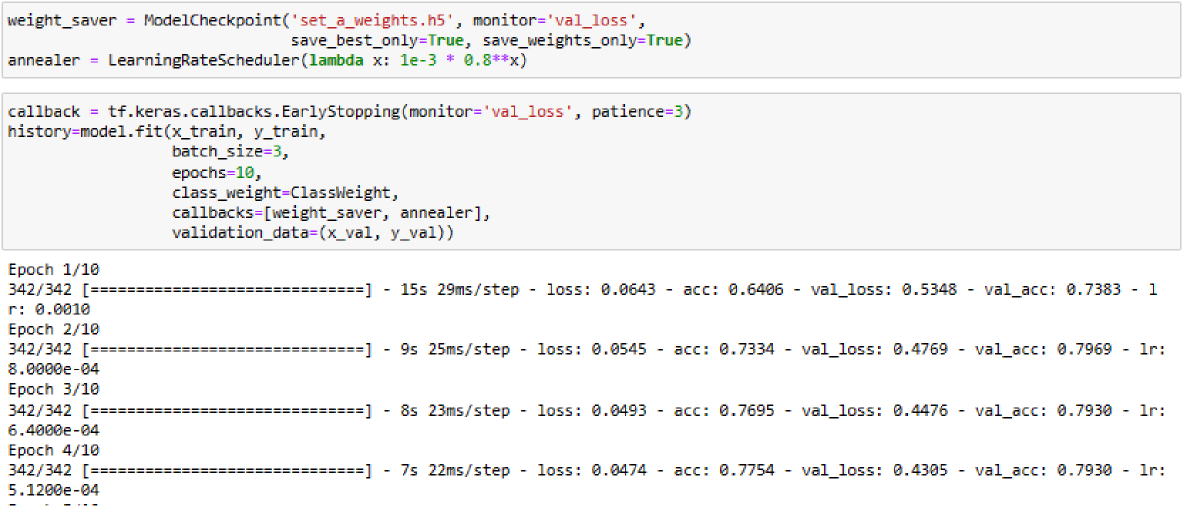
```
print(y_train.shape,y_test.shape,y_val.shape,test_y.shape)
```

```
(1024,) (320,) (256,) (400,)
```

```
y_train = np.array(tf.keras.utils.to_categorical(y_train, len(Class)))
y_test = np.array(tf.keras.utils.to_categorical(y_test, len(Class)))
y_val = np.array(tf.keras.utils.to_categorical(y_val, len(Class)))
test_y = np.array(tf.keras.utils.to_categorical(test_y, len(Class)))
```

```
trainImageCount = 210
bonafideCount = 1000  #normal
spoofCount = 1001 #SPOOF
weight_for_0 = trainImageCount / (2 * bonafideCount)
weight_for_1 = trainImageCount / (2 * spoofCount)
ClassWeight = {0: weight_for_0, 1: weight_for_1,}
ClassWeight
```

Figure 5.14 Splitting Dataset



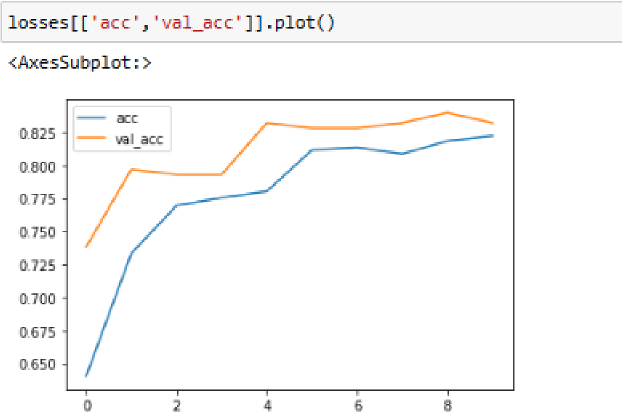
```
weight_saver = ModelCheckpoint('set_a_weights.h5', monitor='val_loss',
                               save_best_only=True, save_weights_only=True)
annealer = LearningRateScheduler(lambda x: 1e-3 * 0.8**x)
```

```
callback = tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=3)
history=model.fit(x_train, y_train,
                  batch_size=3,
                  epochs=10,
                  class_weight=ClassWeight,
                  callbacks=[weight_saver, annealer],
                  validation_data=(x_val, y_val))
```

```
Epoch 1/10
342/342 [==============================] - 15s 29ms/step - loss: 0.0643 - acc: 0.6406 - val_loss: 0.5348 - val_acc: 0.7383 - l
r: 0.0010
Epoch 2/10
342/342 [==============================] - 9s 25ms/step - loss: 0.0545 - acc: 0.7334 - val_loss: 0.4769 - val_acc: 0.7969 - lr:
8.0000e-04
Epoch 3/10
342/342 [==============================] - 8s 23ms/step - loss: 0.0493 - acc: 0.7695 - val_loss: 0.4476 - val_acc: 0.7930 - lr:
6.4000e-04
Epoch 4/10
342/342 [==============================] - 7s 22ms/step - loss: 0.0474 - acc: 0.7754 - val_loss: 0.4305 - val_acc: 0.7930 - lr:
5.1200e-04
```

Figure 5.15 Training Model

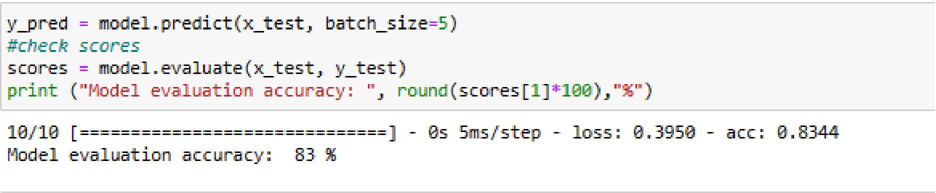
5.2.5 Validating Model.



```
losses[['acc','val_acc']].plot()
```

```
<AxesSubplot:>
```

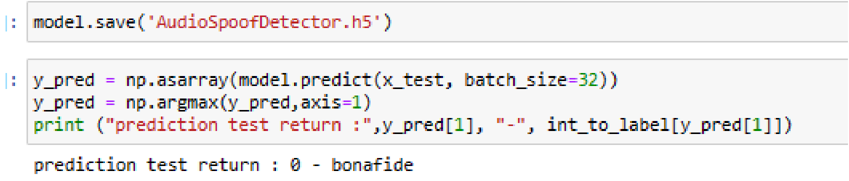
Figure 5.16 Plotting Loss Values



```
y_pred = model.predict(x_test, batch_size=5)
#check scores
scores = model.evaluate(x_test, y_test)
print ("Model evaluation accuracy: ", round(scores[1]*100),"%")
```

```
10/10 [==============================] - 0s 5ms/step - loss: 0.3950 - acc: 0.8344
Model evaluation accuracy:  83 %
```

Figure 5.17 Evaluating model

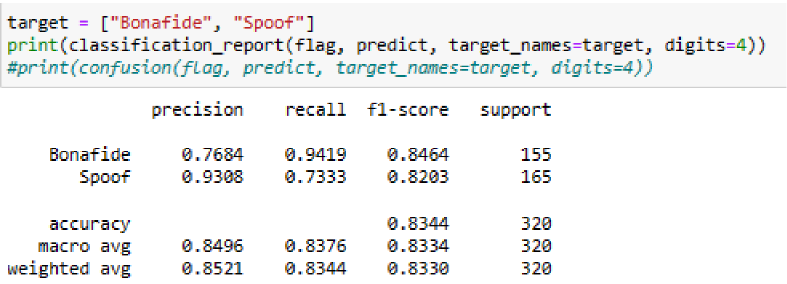


```
model.save('AudioSpoofDetector.h5')
```

```
y_pred = np.asarray(model.predict(x_test, batch_size=32))
y_pred = np.argmax(y_pred,axis=1)
print ("prediction test return :",y_pred[1], "-", int_to_label[y_pred[1]])
```

```
prediction test return : 0 - bonafide
```

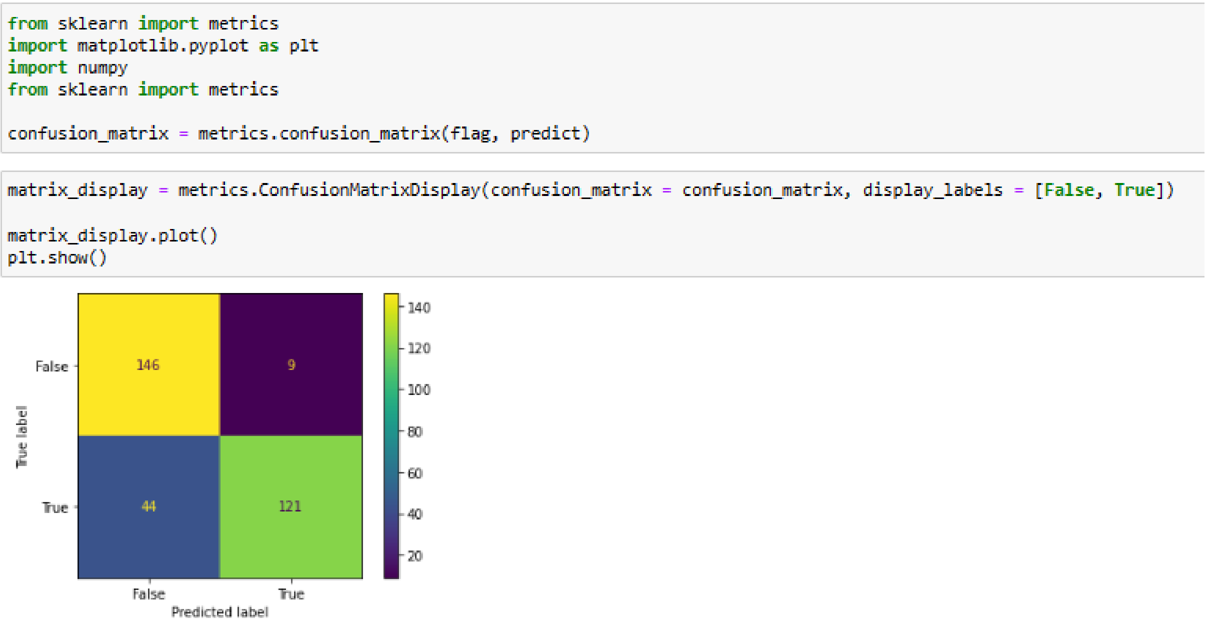
Figure 5.18 Saving Model and predicting for a sample audio file



```
target = ["Bonafide", "Spoof"]
print(classification_report(flag, predict, target_names=target, digits=4))
#print(confusion(flag, predict, target_names=target, digits=4))
```

```
              precision    recall  f1-score   support

    Bonafide     0.7684    0.9419    0.8464       155
       Spoof     0.9308    0.7333    0.8203       165

    accuracy                         0.8344       320
   macro avg     0.8496    0.8376    0.8334       320
weighted avg     0.8521    0.8344    0.8330       320
```

Figure 5.19 classification report

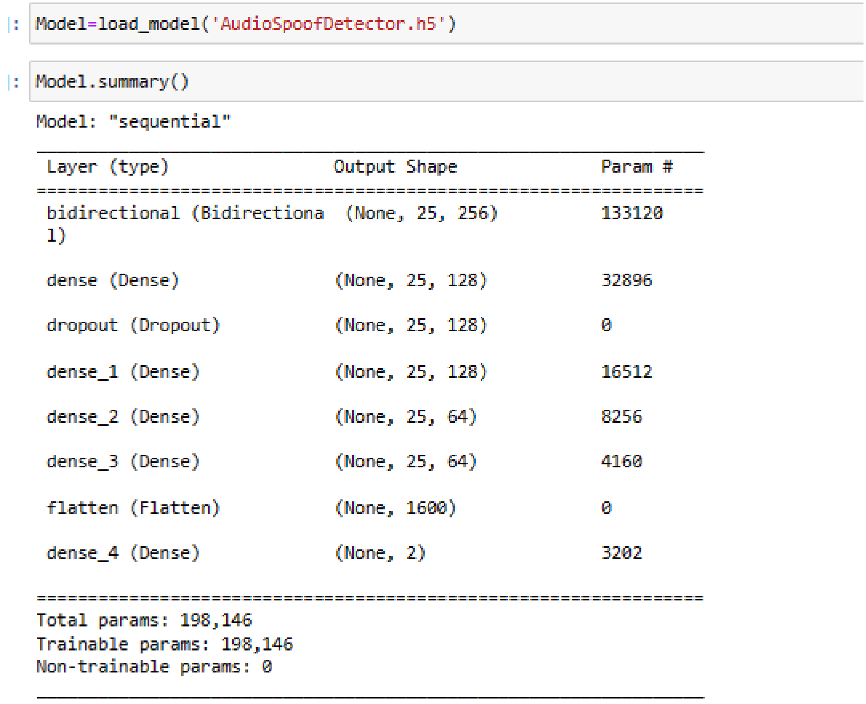


```
from sklearn import metrics
import matplotlib.pyplot as plt
import numpy
from sklearn import metrics

confusion_matrix = metrics.confusion_matrix(flag, predict)
```

```
matrix_display = metrics.ConfusionMatrixDisplay(confusion_matrix = confusion_matrix, display_labels = [False, True])

matrix_display.plot()
plt.show()
```

Figure 5.20 Confusion Matrix

5.2.6 Import the prepared model and provide the audio input for classification.

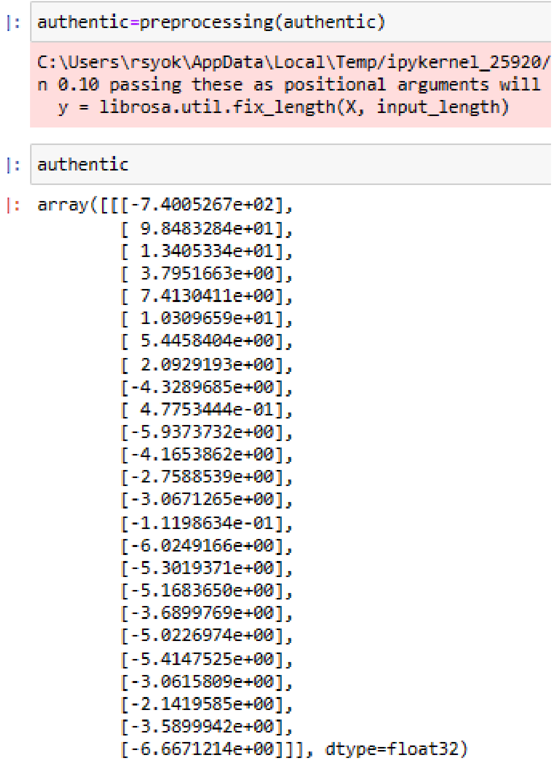


```
|: Model=load_model('AudioSpoofDetector.h5')
```

```
|: Model.summary()
```

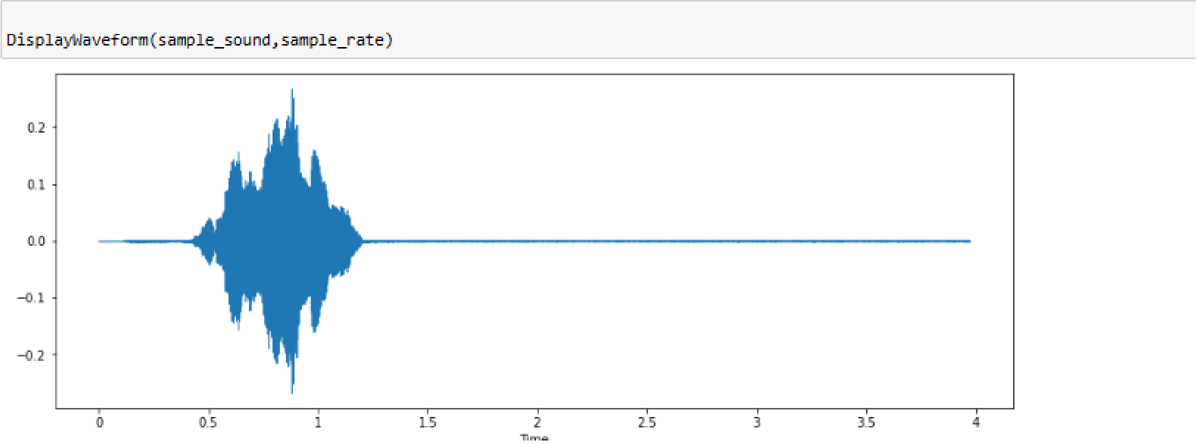
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 bidirectional (Bidirectiona  (None, 25, 256)          133120
 l)

 dense (Dense)               (None, 25, 128)           32896

 dropout (Dropout)           (None, 25, 128)           0

 dense_1 (Dense)             (None, 25, 128)           16512

 dense_2 (Dense)             (None, 25, 64)            8256

 dense_3 (Dense)             (None, 25, 64)            4160

 flatten (Flatten)           (None, 1600)              0

 dense_4 (Dense)             (None, 2)                 3202

=================================================================
Total params: 198,146
Trainable params: 198,146
Non-trainable params: 0
_________________________________________________________________
```

Figure 5.21 Importing saved Model



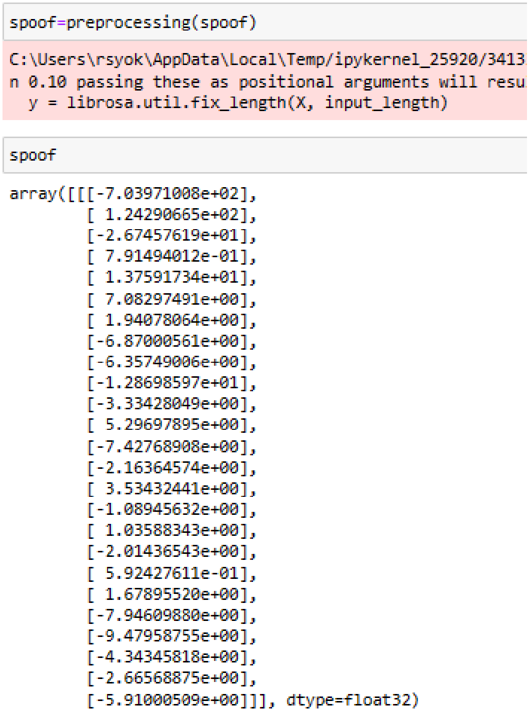
```
[ ]: authentic=preprocessing(authentic)

C:\Users\rsyok\AppData\Local\Temp/ipykernel_25920/
n 0.10 passing these as positional arguments will
  y = librosa.util.fix_length(X, input_length)

[ ]: authentic

[ ]: array([[[-7.4005267e+02],
        [ 9.8483284e+01],
        [ 1.3405334e+01],
        [ 3.7951663e+00],
        [ 7.4130411e+00],
        [ 1.0309659e+01],
        [ 5.4458404e+00],
        [ 2.0929193e+00],
        [-4.3289685e+00],
        [ 4.7753444e-01],
        [-5.9373732e+00],
        [-4.1653862e+00],
        [-2.7588539e+00],
        [-3.0671265e+00],
        [-1.1198634e-01],
        [-6.0249166e+00],
        [-5.3019371e+00],
        [-5.1683650e+00],
        [-3.6899769e+00],
        [-5.0226974e+00],
        [-5.4147525e+00],
        [-3.0615809e+00],
        [-2.1419585e+00],
        [-3.5899942e+00],
        [-6.6671214e+00]]], dtype=float32)
```

Figure 5.22 preprocessing authentic voice data



```
DisplayWaveform(sample_sound,sample_rate)
```

Figure 5.23 Waveform Representation of the authentic voice data



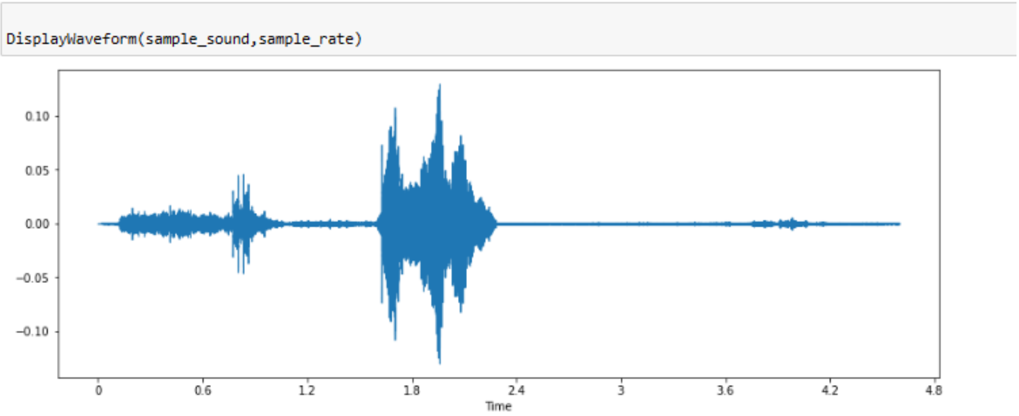
```
spoof=preprocessing(spoof)
```

```
C:\Users\rsyok\AppData\Local\Temp/ipykernel_25920/3413
n 0.10 passing these as positional arguments will resu
  y = librosa.util.fix_length(X, input_length)
```

```
spoof
```

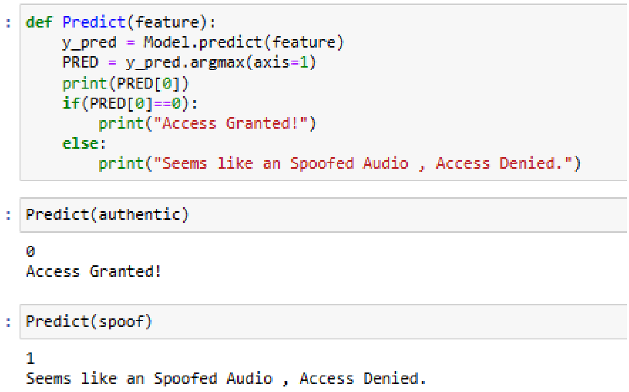
```
array([[[-7.03971008e+02],
        [ 1.24290665e+02],
        [-2.67457619e+01],
        [ 7.91494012e-01],
        [ 1.37591734e+01],
        [ 7.08297491e+00],
        [ 1.94078064e+00],
        [-6.87000561e+00],
        [-6.35749006e+00],
        [-1.28698597e+01],
        [-3.33428049e+00],
        [ 5.29697895e+00],
        [-7.42768908e+00],
        [-2.16364574e+00],
        [ 3.53432441e+00],
        [-1.08945632e+00],
        [ 1.03588343e+00],
        [-2.01436543e+00],
        [ 5.92427611e-01],
        [ 1.67895520e+00],
        [-7.94609880e+00],
        [-9.47958755e+00],
        [-4.34345818e+00],
        [-2.66568875e+00],
        [-5.91000509e+00]]], dtype=float32)
```

Figure 5.24 Preprocessing of spoofed voice data



```
DisplayWaveform(sample_sound,sample_rate)
```

Figure 5.25 Waveform representation of spoofed audio sample.



```
def Predict(feature):
    y_pred = Model.predict(feature)
    PRED = y_pred.argmax(axis=1)
    print(PRED[0])
    if(PRED[0]==0):
        print("Access Granted!")
    else:
        print("Seems like an Spoofed Audio , Access Denied.")
```

```
Predict(authentic)
```

```
0
Access Granted!
```

```
Predict(spoof)
```

```
1
Seems like an Spoofed Audio , Access Denied.
```

Figure 5.26 Prediction of sample authentic and spoof audio samples.

RESULTS AND DISCUSSIONS

CHAPTER 6

RESULTS AND DISCUSSIONS

The proposed system performs better than the existing systems as the dataset used in training the model ASV 2019 which contains of various samples including many attacks and spoof audios like synthetic attacks and replay attacks.The system produces an accuracy of 85%, which can be further integrated with any authentication system to produce better secure access.

CORPORATE SOCIAL RESPONSIBILITY

CHAPTER 7

CORPORATE SOCIAL RESPONSIBILITY

The present development in the technology of voice recorders pave easy way of voice spoof attacks on any authentication system. Hence, the proposed system classifies the spoofed voice from the authentic voice allowing only genuine users to access the authentication system. This is very important because secure mechanism of using voice as biometric key for any authentication system and protecting it from spoof attacks is crucial.

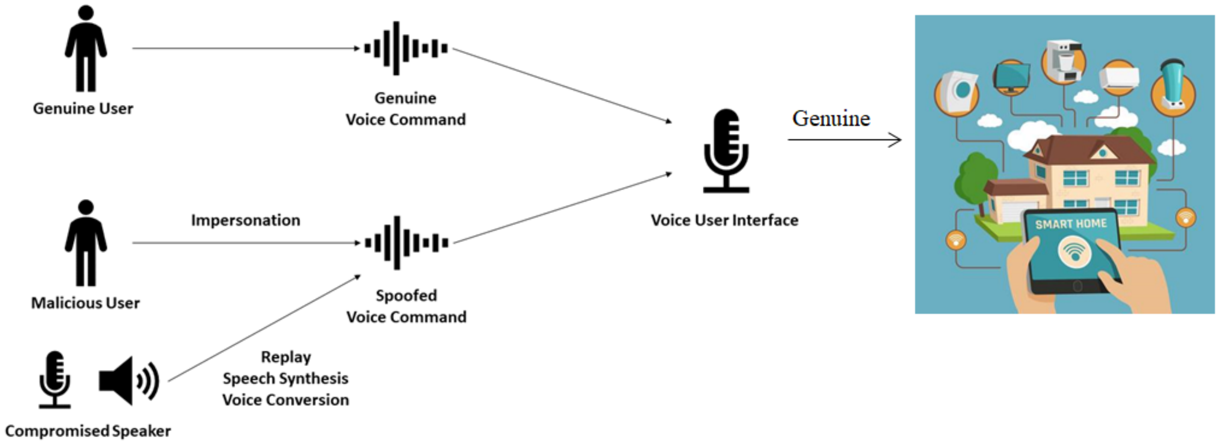
CONCLUSION AND FUTURE WORK

CHAPTER 8

CONCLUSION AND FUTURE WORK

The proposed system can classify human voices from recorded voices. The proposed system also defends from replay attacks and voice conversion attacks. The accuracy obtained for the proposed system is 85%.

The proposed system can be deployed and used alongside the audio authentication system to classify human voices from recorded voices. The authentication system is used to authenticate the legitimate user.The proposed system will be integrated into home automation IOT systems which contains voice-controllable features.



REFERENCES

REFERENCES

- Dua, M., Jain, C. & Kumar, S. LSTM and CNN based ensemble approach for spoof detection task in automatic speaker verification systems. *J Ambient Intell Human Comput* **13**, 1985–2000 (2022). https://doi.org/10.1007/s12652-021-02960-0

- Yamagishi, Junichi; Todisco, Massimiliano; Sahidullah, Md; Delgado, Héctor; Wang, Xin; Evans, Nicolas; Kinnunen, Tomi; Lee, Kong Aik; Vestman, Ville; Nautsch, Andreas. (2019). ASVspoof 2019: The 3rd Automatic Speaker Verification Spoofing and Countermeasures Challenge database, [sound]. University of Edinburgh. The Centre for Speech Technology Research (CSTR). https://doi.org/10.7488/ds/2555.

- https://www.simplilearn.com/tutorials/deep-learning-tutorial/rnn

- https://colah.github.io/posts/2015-08-Understanding-LSTMs/

- Sak, H., Senior, A.W., & Beaufays, F. (2014). Long short-term memory recurrent neural network architectures for large scale acoustic modeling. *INTERSPEECH*.

- Ankur, Tanjemoon & Kundu, Bipasha & Foysal, Md & Ortiz, Bengie & Chong, Jo. (2022). LSTM-Based COVID-19 Detection Method Using Coughing. 10.21203/rs.3.rs-2106413/v1.

- Akyol K, Şen B. Automatic Detection of Covid-19 with Bidirectional LSTM Network Using Deep Features Extracted from Chest X-ray Images. Interdiscip Sci. 2022 Mar; 14(1):89-100. doi: 10.1007/s12539-021-00463-2. Epub 2021 Jul 27. PMID: 34313974; PMCID: PMC8313418.

- Ivan Rakhmanenko Fusion of BiLSTM and GMM-UBM Systems for Audio Spoofing Detection August 2019 International Journal of Advanced Trends in Computer Science and Engineering 6(4):1741-1746.

- Jichen Yang, Rohan Kumar Das,Improving anti-spoofing with octave spectrum and short-term spectral statistics information,Applied Acoustics, Volume 157,2020,107017,ISSN 0003-682X,

- https://ijirt.org/master/publishedpaper/IJIRT149877_PAPER.pdf

- Mittal, Aakshi & Dua, Mohit. (2022). Automatic speaker verification systems and spoof detection techniques: review and analysis. International Journal of Speech Technology. 25. 10.1007/s10772-021-09876-2.

- https://irojournals.com/aicn/article/pdf/4/3/4

- https://journals.pan.pl/dlibra/publication/141648/edition/123487/content/archives-of-acoustics-2022-vol-47-no-2-spoofed-speech-detection-with-weighted-phase-features-and-convolutional-networks-br-disken-gokay?language=en

- Zhou, J., Hai, T., Jawawi, D.N.A. *et al.* Voice spoofing countermeasure for voice replay attacks using deep learning. *J Cloud Comp* **11**, 51 (2022). https://doi.org/10.1186/s13677-022-00306-5