

Voice Impersonation Detection using LSTM based RNN and Explainable AI

by

Kawshik Barua

17201034

Abdur Rahim

17301164

Prantozit Saha Parizat

17301171

Md.Asad Uzzaman Noor

17301128

Miftahul Jannah

13201085

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science

Department of Computer Science and Engineering
Brac University
October 2021


© 2021. Brac University
All rights reserved.

Declaration

It is hereby declared that

1. The thesis submitted is my/our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain any material that has been approved or submitted for any other university or other institution's degree or certificate.
4. All major sources of assistance have been acknowledged.

Student's Full Name & Signature:



Kawshik Barua
17201034



Abdur Rahim
17301164



Prantojit Saha Parizat
17301171



Md. Asad Uzzaman Noor
17301128



Miftahul Jannah
13201085

Approval

The thesis/project titled “Voice Impersonation Detection using LSTM based RNN and Explainable AI” submitted by

1. Kawshik Barua (17201034)
2. Abdur Rahim (17301164)
3. Prantozit Saha Parizat (17301171)
4. Md.Asad Uzzaman Noor (17301128)
5. Miftahul Jannah (13201085)

Of Summer, 2021 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on October 5, 2021.

Examining Committee:

Supervisor:
(Member)



Md.Golam Rabiul Alam,PhD
Associate Preofessor
Department of Computer Science and Engineering
Brac University

Thesis Coordinator:
(Member)

Md.Golam Rabiul Alam,PhD
Associate Professor
Department of Computer Science and Engineering
Brac University

Head of Department:
(Chair)

Sadia Hamid Kazi,PhD
Chairperson and Associate Professor
Department of Computer Science and Engineering
Brac University

Ethics Statement

The research is carried out in accordance with the BRAC University's laws and procedures on ethical norms. In our thesis, we use information from original sources. We're double-checking references and in-text citations for accuracy. We, the five co-authors, take full responsibility for any violations of the thesis code. To address problems, we used a variety of literature and internet resources.

We also recruited the help of several of our university's professors. Finally, we express our gratitude to everyone who helped us. To finish the thesis, we did not employ any misleading techniques. Our research complies with BRAC University's ethics norms.

Abstract

The advancing field of artificial synthetic media introduced deepfakes which made it easier to synthesize a person's voice, identical to their original voice mechanically to use it for negative means. People's voices are exposed to public as it is a proficient and more convenient media of exchanging information over various mediums, entertainment, speech delivering, news reading and so on, making it easier to collect voice samples for creating fake yet almost identical voice samples to trick people. So it has become vital to prevent this crime which led us to do this research paper for saving the victims of voice impersonation attacks where we used LSTM based RNN model in order to distinguished between real and synthesize voice. Furthermore, to compare the results we got from the mentioned process, we build a SVM classifier and finally we've explained the predicted outputs(fake or real) of both LSTM and SVM model by using an Explainable AI method named LIME. Our research resulted in 98.33% accuracy rate through our proposed model and very low percentage of error in detecting fake/synthesized voices.

Keywords: Deepfakes, Voice Impersonation Detection, LSTM based RNN, Feature Extraction, SVM, LIME, Explainable AI.

Acknowledgement

All gratitude be to the great Allah, who granted us to submit our thesis without any major setbacks.

We are grateful to our honorable sir Md. Golam Rabiul Alam, PhD for his supervision throughout the whole working time. He helped us a lot by sharing his invaluable knowledge with us, whenever we needed help. Without his guidance it would be so difficult for us to complete the paper properly.

Finally, without our parents' unwavering support, we may not be able to achieve our goals. We are currently on the verge of graduating thanks to their kind assistance and prayers.

Table of Contents

Declaration	i
Approval	ii
Ethics Statement	iii
Abstract	iv
Acknowledgment	v
Table of Contents	vi
List of Figures	viii
List of Tables	ix
Nomenclature	x
1 Introduction	1
1.1 Background	1
1.2 Problem Statement	2
1.3 Research Objective	3
1.4 Thesis Organization	3
2 Related Work	5
2.1 Literature Review	5
3 Methodology	11
3.1 Proposed Approach	11
3.2 Dataset Description	12
3.3 Feature Extraction	13
3.4 Data Preprocessing	16
3.5 Classifications	17
3.5.1 Long Short Term Memory (LSTM)	17
3.5.2 Support Vector Machine (SVM)	20
3.6 Explainable AI	22
3.6.1 Local Interpretable Model-agnostic Explanations	22

4	Evaluation and Performance Analysis	24
4.1	Implementation	24
4.1.1	LSTM Implementation	24
4.1.2	SVM Implementation	25
4.1.3	LIME Implementation	25
4.2	Result Analysis	26
4.3	Comparative Analysis	30
5	Conclusion	31
	Bibliography	35

List of Figures

3.1	Top level layout of the voice impersonation detection model	12
3.2	MFCC wave plots for single audio sample	14
3.3	Chroma vector wave plots for single audio sample	14
3.4	ZCR wave plots for single audio sample	15
3.5	Spectral Centroid wave plots for single audio sample	15
3.6	Spectral Roll off wave plots for single audio sample	16
3.7	Difference between RNN and Traditional Feed Forward Neural Network	17
3.8	Difference between working mechanism of RNN and LSTM	18
3.9	LSTM internal Architecture	19
3.10	A simple representation of SVM working mechanism	21
3.11	The idea behind local explanation methodology employed by LIME .	23
4.1	Confusion Matrix for LSTM and SVM model	27
4.2	ROC curve for LSTM and SVM	29
4.3	Explaining LSTM model's prediction result using LIME	29
4.4	Explaining SVM model's prediction result using LIME	30

List of Tables

4.1	Calculated Scores for LSTM and SVM	28
4.2	Accuracy comparison between different classifiers	30

Nomenclature

The next list describes several symbols & abbreviation that will be later used within the body of the document

CENS Chroma energy normalized statistics

CQT Constant Q Transform

EER Equal Error Rate

FN False Negative

FP False Positive

GFB Gammatone filter bank

HLL human log-likelihoods

LFCC Linear Frequency Cepstral Coefficient

LIME Local Interpretable Model-agnostic Explanations

LSTM Long Short-Term Memory

MFB mel-frequency rectangular filter banks

MFCC Mel-Frequency Cepstral Coefficients

RFE Recursive Feature Elimination

RMSE The Root-Mean-Square Energy

STFT Short Time Fourier Transform

SVM Support Vector Machine

TN True Negative

TP True Positive

XAI Explainable AI

Chapter 1

Introduction

1.1 Background

Voice impersonation detection is a system where fake voice data are identified in order to prevent any unwanted circumstances. It is known that voice impersonating specifically means imitating a person's voice for various purposes. Earlier impersonating someone's voice was limited in the field of entertainment where an impressionist or a mimic artist used to impersonate celebrities or famous peoples to make a funny act in order to entertain audience. It was harmless form of entertainment done by artists. It also helped reminiscing dead famous people through their voices. But after deepfake technology was developed, more apps/software are getting built to make fake voice presentations that promotes a type of entertainment, especially over social media. Voice mimicking is not limited to artists anymore. Apparently it may project the idea of entertainment but if we look at it from a higher view, we will notice it can be used for harmful purposes such as fraudulent [23], security breach, defaming famous personalities and so on.

With the thriving improvement of technology and availability of smart devices, voice based applications and software are becoming more common these days. The use of voice based biometric systems such as unlocking a device or a door, smart home assistance (Google Home, Amazon Alexa), access control of a device and security control are becoming more popular gradually. Additionally, in recent times, voice based texting system is introduced over many social media platforms such as Facebook messenger, Whatsapp, WeChat and many more.

Undoubtedly it has made communication and exchange of information less time consuming and easier for both educated and uneducated people but it is an undeniable fact that people with harmful intentions are getting more scope to send false data through voice texts by impersonating particular people. With all these voice based improved technology; we cannot ignore the underlying fact that our voices are getting easily exposed to the public more easily than ever.

Identifying fake voice data to save people from getting into dangerous consequences is our aim of study. In our thesis work, we collect both original and synthesized voice data as our base data set and then implement an LSTM based RNN model in order to differentiate between them. Voice spoofing software, apps, websites are increasing

day by day whereas enough app or software is not getting built to identify those synthesized voices. Our focus is to get control over future fraudulent occurrences that may happen due to this increased number of spoofing software. Our structured model is able to collect good amount of data set and our system is going to provide more accuracy while detecting fake voices than existing researches.

1.2 Problem Statement

Impersonation is a type of presentation attack in which a false biometric is used to impersonate a real person. Fraudsters aim to imitate a person's voice using advanced voice synthesis, voice conversion or imitation, and recorded playback. Although voice impersonation is one of the simplest assaults to carry out, since it just requires a recording of the victim's voice. It is extremely difficult to detect because the recorded voice samples shares many of the same features as the victim's real voice. Voice impersonation attacks may be used to make purchases using a victim's credit card information, operate Internet of Things linked devices such as smart appliances, and provide hackers with unwelcome access to sensitive consumer data such as financial information, home addresses, and more.

There are two types of synthesised voice available. First, non-DNN-based voice synthesizers produce voice that sounds more robotic and is easily recognized by human hearing. On the other hand, DNN-based voice synthesizers use the strategy to synthesis voices, produced more realistic and persuasive speech. It's getting more difficult to tell the difference between a real and a fake voice as the field of voice impersonation grows. When certain people abuse technology, the problem develops. Impersonating a person's voice using a deep neural network (DNN) may appear to be a difficult task, but it can be accomplished simply by providing some text as input to the system, which will then be transformed into voice that sounds exactly like the victim's using various methods and algorithms. As a result, distinguishing between a genuine voice uttered by the speaker and an AI synthetic voice that was transformed utilizing certain technologies created or accessible on the market is extremely difficult. So, the main components of a DNN model are layers and neurons, which were discussed in [22]. Furthermore, each layer in a deep neural network plays a unique function in the learning of the input representation. However, while numerous deep learning-based voice identification studies have been done in the past, none of them have been able to recognize mimic voices correctly. Considering all factors, there is still potential for progress in detecting a fake or real voice. There have been studies in this area, but the concept of identifying fake voices is new. As a result, no model can distinguish between fake and real voices without utilizing the correct methodologies for extracting features from voice samples and predicting fake or real voices.

Existing fake or real voice detection techniques, which typically use deep learning models, do not have the high accuracy level of detecting voice samples, because of data loss during feature extraction and lack of using proper methods. As a result, we proposed a model in our research that can more accurately distinguish between fake and real voice.

1.3 Research Objective

In this research we aimed to develop a detection system which will be able to differentiate between real and synthesized voices that are made from advance voice technologies like TTS and VC. For differentiating purpose we have used Long Short-term Memory (LSTM) based RNN that are capable of learn order dependency in sequence prediction tasks and this behavior is a need in a variety of different complex issue areas like machine translation, speech recognition. A model based on Support Vector Machine (SVM) will also build in order to compare its result with lstm based model. The purpose of using Explainable Ai here is to describe how black box decisions of this detection system are made meaning based on which features it's claiming a sample input voice as a real or fake voice and how much contribution of each features have behind that predicted output result.

- An intelligent system that is capable of identifying a real and fake voice.
- A separate detection model is developed which is based on SVM and LSTM to detect highly synthesized artificial voices.
- A model that explains how it came to its conclusion explainable ai methods is used on the development models which is capable of describing the desired output by the model.

1.4 Thesis Organization

This research report on detecting voice impersonation has provided a structured model to distinguish between fake voice data and original voice data more precisely than subsisting models. The goal of the authors is to establish a model that can be implemented later alongside voice based software, home appliances and security systems to stop fraudulent occurrences. The synopsis of this research work is presented in this section.

To begin, the introduction section (**Chapter 1**), briefly describes the background of the selected research topic and the underlying reasons that led the authors to address the issue statement. The research methodology of the paper is also presented concisely to provide an overall idea about the scheme.

In the literature review section (**Chapter 2**) all the existing relevant works done regarding voice impersonation and detection is acknowledged and evaluated. The goal of this section was to identify the flaws of these works that will help to come up with a better model. The flaws were detected and a better model is proposed where the authors solved the vanishing gradient problem of audio input.

In methodology section (**Chapter 3**) the research started to get into the original shape where the entire methodology of the paper is described with appropriate detailing. The proposed LSTM based RNN model along with SVM based model for comparison, a visual structure of the overall methodology, the procedure of data set collection, feature extraction detailing and data processing is presented in this

section. Later, relevant explainable AI and selected models are also described in this section.

In evaluation and performance analysis (**Chapter 4**), at first, the implementation of the proposed model is presented which includes LSTM implementation, SVM implementation and LIME implementation. Next, the output from these implemented models is analyzed under result analysis section. Lastly, a comparison is shown between our research result and the existing research results in the relevant field to compare the accuracy of our proposed model.

Chapter 2

Related Work

2.1 Literature Review

This section will concentrate on some relevant efforts in the subject of synthetic voice detection. We evaluate the variety of approaches utilized to accomplish the synthetic speech detection, and we demonstrate how several deep learning-based algorithms tried to distinguish between real and fake speech.

In order to comprehend these works, one must first grasp how synthetic voices are created. There are two types of synthesized voices available.:

- HMM and GMM are non-DNN based methods for learning and reproducing speech features.
- For synthesizing naturalness of speech and even unfamiliar words, DNN is used.

The first method (non-DNN based) uses speech concatenation technique, which combines several pre-recorded speech fragments to create a new clip voice [24]. The other format analysis approach generates robotic-sounding speech using acoustic models without the need of a human voice as input [5]. Another method for synthesizing speech is articulatory speech synthesis, which involves modeling the human vocal tract and vocal biomechanics [8]. Some studies investigate the use of HMM to regulate speech proprieties such as basic frequency and duration [4]. These approaches were frequently used in the early years of speech synthesis; however, they suffer from naturalness concerns that may be easily detected by human hearing. DNN-based speech synthesis algorithms solve this issue by directly mapping language information to acoustic data. Based on DNNs, several models (e.g., Boltzmann machines [7], deep belief networks [6], mixed density networks [1], and Bidirectional LSTM [16]) are proposed for synthesizing high-quality and natural speech. Some synthetic examples can be found online. Using these models recently many advance voice synthesis technologies has been developed. DeepMind's WaveNet [11] in 2016 and Google's Tacotron [19] in 2017 are two landmarks in voice synthesis. These two models use considerably advance DNN based speech synthesis, allowing for large-scale commercial applications for developing TTS and VC systems. Because of the strong capabilities of WaveNet and Tacotron, commercial systems like as Baidu TTS, Amazon AWS Polly [25], and Google Cloud TTS [28] have been created. Unfortunately, some attackers can utilize speech synthesis techniques to create fake voices

with harmful intent.

Coming back to the subject of works related with detection of these synthesized voices, A variety of researches regarding the identification of real and fake voices are accessible in the literature. Research work [18] suggested a collection of short-term spectral characteristics in mid-2017 that can significantly enhance the accuracy of fake speech detection. The authors examine the differences between synthetic and natural speech in great detail and uncover some fascinating facts, such as the lower frequencies (1kHz) and higher frequencies (greater than 7kHz) are the most useful for distinguishing between fake and natural speech.

Yu et al. released a study on the usage of Deep Neural Networks (DNNs) for voice spoofing detection as the popularity of DNN solutions grew [20]. The fundamental concept is to utilize DNNs to extract dynamic acoustic characteristics and identify an utterance as genuine or faked. According to the findings, the suggested technique outperforms standard static feature analysis using GMM classifiers. In the research work, they have used dynamic acoustic characteristics to train a five layer DNN spoofing detection classifier and present a unique, simple scoring approach for spoofing detection that only uses human log-likelihoods (HLLs). They showed mathematically that the novel HLL scoring system is better than traditional LLR scoring approach for spoofing detection, especially when the spoofing speech is highly similar to human speech. The ASVspoof 2015 database is used to test the performance of spoofing detection using various characteristics and classifiers. The performance of DNN classifiers is evaluated using dynamic SBCC and CQCC in this research work. MFCC and LFCC, for example are SBCC characteristics that have been widely employed in various speech processing applications. LFCC, MFCC, MFB, GFB, GFCC these five features are used in this paper for speech recognition task. Besides, GMM-LLR classifier which has been used widely in ASV systems for spoofing detection, DNN-LLR classifiers which contains five hidden layers to distinguish human or synthetic voice and DNN-HLL classifier was introduced. The experimental findings demonstrate that the DNN classifier can efficiently identify faked speech from human speech on S1- S9 attacks. On all of the examine characteristics the HLL outperforms the GMM-LLR model scoring with LLR. The DNN-HLL classifier based on CQCC can lower the average EER score on unknown attacks to 0.089 %. Furthermore, the average EER score on all attacks was 0.089 % which is better.

Mimicry voice can be detected using convolutional neural networks according to [26]. In the research work they focused on Automatic speaker verification system and shows how spoofing attacks are occurred on this ASV system. However, the main purpose of this research was to detect this kind of mimicry voice using CNN. The main problem of ASV biometric system is that it's powerless against mocking assaults. There are four types of mocking attacks and in a mimicry assault a fraudster tries to imitate the voice of a licensed speaker. The human voice is so versatile that it may be used to create imitation samples and using these samples, they can create a voice pattern of the targeted individual and try to get access to their personal information and cause them harm. However, there is no impersonation (mimicry) sample available publically so for mimicry detection purpose they

have created some mimicry samples generated by professional mimicry artist and some samples which are collected from the celebrity speech available on internet. Additionally, creating a high-quality mimic database is not an easy work and for this reason speech samples were collected at 8 kHz and for doing this they have selected two Telugu-speaking celebrities. The celebrity's speeches are recorded as short frame, each with a maximum duration of 3 sec. 68 voice samples from two celebrity speech were and mimic artist in order to make the dataset. Imitators that seek to imitate the target speaker's voice use prosodic elements such as intonation, stress, and rhythm and for this reason they have chosen spectral features like MFCC. Apart from this, for evaluating the result a classifier based on CNN model was established. This model is successfully adopted from the detection of speech presentation attacks and cost optimization analysis was performed. After doing all the above steps they got a testing accuracy of 0.6693 or 67% and the equal error rate was 0.3585.

A Deep Learning based solution on Synthetic Speech Detection was proposed in the research work [29]. The project's goal was to create a model that could extract voice's dynamic characteristics. and which will be able to differentiate between artificially generated voice and real human speech. To do so they have used a dataset from APTLY Lab named Fake-Or Real dataset of size 4GB which was freely available. However, the model was built on CNN and RNN. CNN aids in the learning of pixel positioning relationships and allows neural networks to recognize forms and patterns. When convolutional layers are combined with fully linked layers, a strong architecture emerges that can recognize patterns in everything forms to complex object, from lowest to the highest layers. RNNs, on the other hand, operate on the concept that the output of one layer becomes the input of another. However, It aids in recalling past and current decisions that are impacted by what has been learnt from previous experiences. The model's structure consisted of four CNNs, followed by max pooling and dropout. After that, a flatten layer is utilized to extract the audio's various characteristics. In addition to this, they have used three different approaches named STFT, MFCC and Mel Spectrogram on the audio file for extracting features. Finally, the loss function and accuracy were used to assess the proposed model's performance during and after training. The loss function indicates how good or bad a model performs after each round of optimization and It's a metric for how close the model's output forecast is to the actual data.. The model's accuracy was 94 percent while the loss value was 0.691 percent.

Research study [9] suggested a collection of unique short-term spectral features based on synthetic and natural speech characteristics that may effectively capture the discriminant features between natural and synthetic speech. Techniques that fundamentally employ spectral mapping processes are used to create voice converted speech and as a result synthetic speech often lacks spectral information that may be used to distinguish it from genuine speech. For evaluating the discriminative information between real and synthetic voice they have used F-ratio analysis and power spectrum coefficients courses. It's calculated as the ratio of power spectrum coefficients' class covariance to within class covariance for a particular frequency point k . The F-ratio distribution reveals that discriminating information may be found in both lower and higher frequency regions. Sharp peaks in the lower frequency range, typically within 1 kHz. The mel-scale representation of the higher frequency

range is poorer, but it has a significant degree of separability. However, they used the PESQ matrix to evaluate speech objective quality, which is derived by combining the average distribution and the average asymmetrical disturbance value between the reference and synthesis loudness spectra. Apart from this they have introduced some anti-spoofing features and their use in countermeasure framework such as Frequency wrapping based cepstral feature which includes Speech signal based frequency cepstral coefficients (SFCC), Formant-specific cepstral features using block transformation which includes MOBt and Speech-signal-based overlapped block transformation based cepstral features which includes SOBt, Inverted frequency warping scale based cepstral features, Inverted mel-warping overlapped block transformation (IMOBt) and Inverted speech-signal based overlapped block transformation (ISOBt). Furthermore, for evaluating the performance of short-term cepstral feature they have used ASVspoof 2015 corpus dataset. Speech frames with a frame size of 20ms and 50% overlap are used to extract features and the hamming window was used for windowing. All of the investigations use the 20-dimensional short term features like MFCC and SFCC vectors, as well as their delta and double delta coefficients. Three successive speech frames are used to determine dynamic characteristic. However, for classification purpose they have used GMM classifier for synthetic speech detection which was trained using maximum likelihood criteria. Each class parameter was estimated using ten iterations of EM method and the maximum number of Gaussian components was 512. Finally, the equal error rate (EER) is used to evaluate the system's performance in both speaker verification and synthesized speech recognition. Using the Bosaris toolbox and the receiver operating characteristics convex hull (ROCCH) approach, the EER is computed. The lower the EER number, the greater the anti-spoofing performance.

Previous researches such as [15] demonstrates that dynamic acoustical characteristics (such as filter banks, dynamic MFCCs, and dynamic linear prediction cepstral coefficients) are better participants for spoofing detection than standard static features (such as magnitude-based features and cosine normalized phase features). Based on this, as well as the fact that DNNs are widely recognized for their ability to extract dynamic characteristics, the researchers opted to use a 5-layer deep neural network to conduct output classification on the AVSpeech2015 dataset. Although this dataset does not include the most recent deep neural network based TTS systems (such as DeepVoice3), it is an excellent place to start when training a DNN to identify faked speech. The experiment findings demonstrate that DNNs with dynamic features outperform earlier approaches based on static features and GMM models.

Zhang et al. released a paper in mid-2017 on their research into deep-learning frameworks for speaker verification anti-spoofing [21]. The authors propose using CNNs in combination with RNNs to recognize synthetic speech in their study. The suggested technique shows the state-of-the-art performance for an end-to-end single system using the ASVspoof2015 dataset as a baseline. AI synthesized voice can be detected by using bispectral analysis. This technique is based on the observation that current speech synthesis algorithms introduce specific and unusual higher-order bispectral correlations that's not typically found in human speech. Google and Lyrebird from all other recordings following this strategy, five spectral logistic regression

classifiers are trained to distinguish each synthesized voice from all categories.

Spoofing speech can be detected using Temporal Deep CNN method [13]. The goal of spoofing speech detection is to tell the difference between spoofing and real speech. In the research work for spoofing speech detection they have suggested a classifier which is based on temporal Deep CNN. The feature trajectories are initially convolved with a series of filters using the temporal CNN and then uses a max-polling layer to obtain the highest response rate of these filters throughout a window. Due to max-polling layer, it helps to extract relevant features from a big temporal series without concatenating a huge number of neighboring frames as in feed forward deep neural network. The ASVspoof 2015 database is utilized here to evaluate the performance of the temporal CNN based classifiers. It contains ten different forms of spoofing attacks, numbered S1 through S10. Furthermore, 5 types of features including log magnitude spectrum (LSM), instantaneous frequency derivative (IF), baseband phase difference (BPD), Group delay (GD) and modified group delay (MGD) were extracted. The amplitude and phase spectra of each frame are first acquired by performing a short-time Fourier transform (STFT) on the speech signal with a 25ms computational window and a 15ms overlap. The EER rate was 0.41 %. As lower EER rate indicates better performance and it verifies the utility of the temporal CNN-based method for speech detection.

A solution of voice presentation attack detection based on End-to-End CNN was proposed in the research work [17]. The authors demonstrate in this research that the suggested architecture mostly learns discriminative information from lower and higher frequencies, which is consistent with prior studies that combined manual feature extraction with standard machine-learning methods. This demonstrates that the DNN can extract frequency characteristics directly from raw audio and that the DNN learns from the same spectrum areas as the conventional method, with comparable or greater accuracy. Basically they have used the CNN-based end-to-end acoustic modeling technique for automatic speech detection. The CNN used in this technique comprises of a feature stage modeled by N convolution layers followed by a classification stage modeled by a multilayer perceptron (MLP). The feature stage in the experiment consists of one convolution layer and the classifier stage consists of an MLP with a single hidden layer or a single layer perceptron (SLP). They utilized two publicly accessible datasets for their experimental setup avspoo and asvspoo datasets were used. The first dataset was about contact assaults and the second one was about logical attack. However, they have used two different techniques for evaluating the result of these two databases, EER for avspoo and hter for asvspoo. With the EER criteria, the threshold is determined individually for each type of assault in both the development and evaluation sets. The system's performance then assessed by average the EER over all assaults both known attacks and the unknown attacks. Apart from this, before feeding the raw speech signal to CNN, they have executed an energy based vocal activity detection to eliminate the quiet sections at the beginning and end of the utterances and normalize the signal in each segment of the kernel width kW by its mean and variance. On AVspoo -PA, Avspoo -LA and ASVspoof the suggested CNN based approach performs well.

As we can see, several deep learning-based works in the field of voice detection have

been done earlier, but none of the works have taken the vanishing gradient problem of voice input into account. As audio input is obtained in time domain, part of the input information may be lost due to the vanishing gradient problem, which can have a significant impact on the output. The LSTM-based RNN is an excellent model for tackling this problem, and we utilized it in our study to obtain high accuracy in recognizing false and real voices. Furthermore, to the best of our knowledge, no other research in the field has used the Explainable AI method to explain the models' predictions. However, in our study, we employed an Explainable AI called LIME to explain the predictions generated by our LSTM-based model, which will assist us in understanding how LSTM works while making the predictions.

Chapter 3

Methodology

3.1 Proposed Approach

We proposed a model based on LSTM based RNN which will be able to differentiate between real and fake voice. For this purpose first of all we have collected our dataset named FoR dataset which consists of total 13956 audio files of 2 sec. As all the files were audio file so we have extracted 8 important features like Mel Frequency Cepstral Coefficients (MFCC), Chroma Short Time Fourier Transformation (STFT), Chroma Constant Q Transform (CQT), Chroma Energy Normalized Statistics (CENS), Spectral Centroid, Spectral Roll Off, Zero Crossing Rate (ZCR), Root Mean Square Energy (RMSE). Then for data preprocessing purpose we have used Standard Scaler for scaling the data in range and Recursive Feature Elimination (RFE) to keep the more impactful or relevant features present in our dataset and then train the dataset into our model to get the desired result. A model based on SVM was also build in order to compare it's result with our proposed model. In addition, we explained the predicted output we obtained from our LSTM model and from the SVM model using an Explainable AI method called LIME. The following diagram is a top level layout for our working strategy:

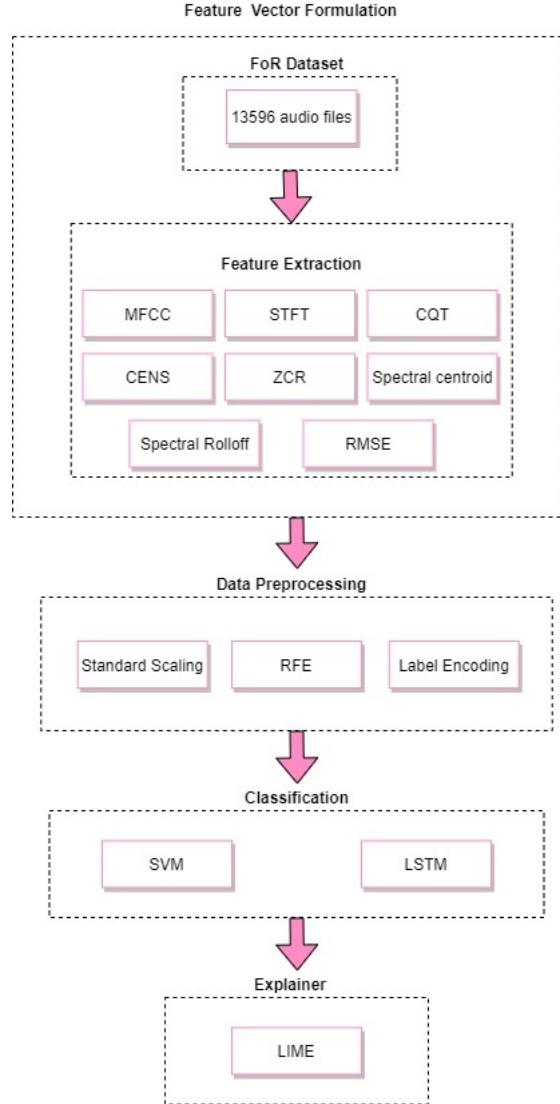


Figure 3.1: Top level layout of the voice impersonation detection model

3.2 Dataset Description

For training our proposed models to classify between fake and real voice, we needed a dataset of raw audio files containing both real human voice samples from various sources along with AI synthesized fake voice samples using latest techniques. Very few public datasets such as The “AsvSpoof” dataset [10], the “Anti-Spoofing for Text-Independent Speaker Verification: An Initial Database” [14] the “FoR (Fake or Real)” [27] dataset are available in this regard. Among these we chose the FoR dataset created by Aptly lab as it contains more number of utterances also because the fake utterances are synthesized with the advanced technology and voice synthesizer products such as Amazon AWS Polly, Google Cloud TTS, and Microsoft Azure TTS etc, which results in naturalness similar to a real person’s voice. Real voices in FoR dataset are gathered from publically available speech related datasets and

free videos/audios on internet mediums like Facebook, Youtube, Twitter, Ted Talks which cover a great variation of speaker age, sex, accent, tone etc. The FoR dataset is available in different versions publically . In our case we used the “for-2sec” version which has files truncated at 2 seconds. However, This 2-sec version of FoR dataset consists of a total 13,956 raw audio file and some of them are real voice and some of them are fake.

3.3 Feature Extraction

As discussed in 3.2 our dataset contains audio files with extension like .mp3, .wav etc. But these data’s are provided as a form of audio files which cannot be understood by the deep learning models directly. So in order to establish our model and train data we need to process audio files and convert them into data features so that deep learning machine learning models could use them . Feature extraction is one such process to convert audio files into an understandable format. In audio analysis process, an initial crucial step is to extract features from the audio files. The goal is to pull out a collection of features from the initial audio files containing dataset. These features extracted should be informative in respect to the qualities which defines an audio signal. The process of feature extraction may also be interpreted as a conversion technique because while doing feature extraction while we are getting tabular data from an audio signal which is in time frequency domain.

In this report, we have extracted eight features of audio files that will help our proposed model to differentiate between real and fake synthetic voice. We used python based Librosa library for analyzing and extracting features. These eight features and their extraction process using Librosa is discussed below:

- **Mel-Frequency Cepstral Coefficients (MFCC):** A signal’s Mel frequency cepstral coefficients (MFCCs) are generally a collection of characteristics (typically 10-20) that succinctly characterizes the all-inclusive form or shape of a spectrum. MFCCs are usually calculated by taking a signal’s Fourier transform and then mapping the spectrum’s powers onto the scale named mel scale .The mel scale is a perceptual scale of sounds that listeners consider to be equally spaced apart. A well-known formula for converting f hertz to m mels is:

$$\text{Mel}(h) = 2595 \log \left(1 + \frac{f}{700} \right) \quad (3.1)$$

then the log values of the magnitude at each of the mel frequencies are taken and converted by using discrete cosine transform technique. The amplitudes of the resulting spectrum is called MFCC.

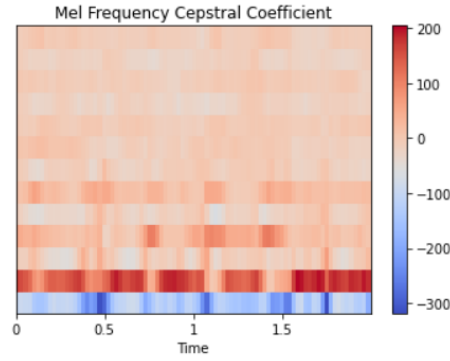


Figure 3.2: MFCC wave plots for single audio sample

- Chroma Vector:** Chroma-based characteristics, which are also familiar as "pitch class profiles," are a strong method for evaluating music with usefully classified pitches into twelve groups. The chroma vector is basically a twelve(12) element spectral energy representation in which the bins reflect the 12 equally toned down pitch classes of western-style music. Chroma_stft (Short Time Fourier Transform) is used for extracting chroma features with time intervals between the frequencies, While Chroma_cqt (Constant Q Transform) extracts features with geometrically placed frequency axis. Chroma energy normalized statistics (CENS) is statistics of energy distribution between the energy bands.

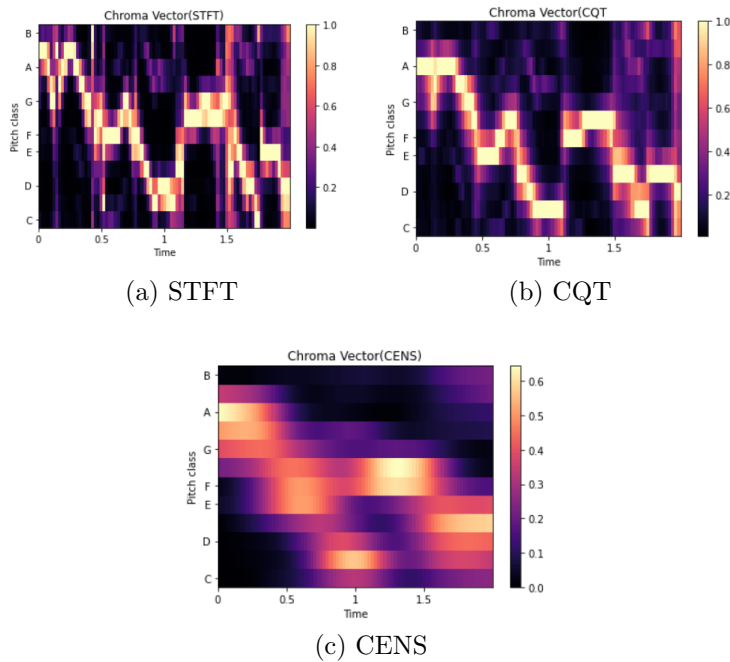


Figure 3.3: Chroma vector wave plots for single audio sample

- Zero-Crossing Rate (ZCR):** An audio frame's Zero-Crossing Rate (ZCR) is basically the value of rate at which the signal's sign changes during the

frame. To put it another way, it's the sum of number at which the signal's value shifts from positive side to negative side of an axis and vice versa. The ZCR is calculated using the following equation(3.2). Here s represents signal of length L . if $s(t) = 1$ if the signal has a positive amplitude at time t or 0 in the other case.

$$ZCR = \frac{1}{L} \sum_{t=1}^L |s(t) - s(t-1)| \quad (3.2)$$

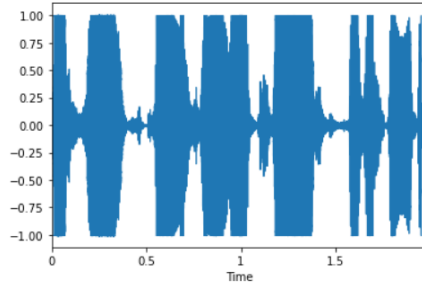


Figure 3.4: ZCR wave plots for single audio sample

- **Spectral Centroid:** Spectral Centroid is determined as the mean weight of the frequencies contained in the sound and shows where the sound's "center of mass" is located. If the frequencies in music remain consistent throughout, the spectral centroid will be towards the middle, and if the sound ends with high frequencies, the centroid will be near the end.

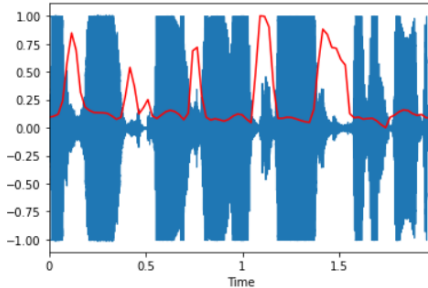


Figure 3.5: Spectral Centroid wave plots for single audio sample

- **Spectral Roll off:** The frequency below which a large proportion of total spectral energy, such as 85 percent, resides is referred to as spectral roll off. The spectral roll off point is the percentage of spectrum bins in which 85 percent of the power is at the lower side of frequencies. In other words, the roll-off frequency is the frequency below which 85 percent of the total spectral magnitude is gathered.

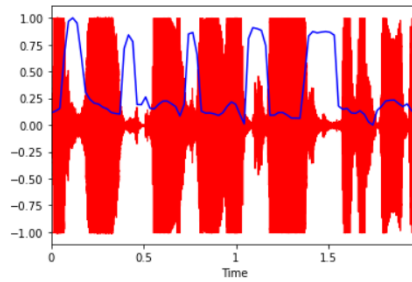


Figure 3.6: Spectral Roll off wave plots for single audio sample

- **Root-Mean-Square Energy (RMSE):** The most significant metric that indicates the magnitude of a signal is the Root Mean Square (RMS) value. For every frame of an audio sample, RMS energy is calculated separately. The overall magnitude of a signal correlates to the signal's energy. That basically correlates to the signal's loudness in audio signals

3.4 Data Preprocessing

Data preprocessing is an significant part of Deep learning Machine Learning since the data quality has a direct effect on a model's learning capacity. Since the range of values of our extracted data features varies widely, first we need to scale them to bind within a range. Among the available feature scaling technique, we choose Standard Scaler to scale the data. It follows Standard normal deviation (SND) and as a result of that the mean is reset to 0 and the data is scaled to unit variance. However, The Standard Scaler scales data such that the distribution centered on 0, with a standard deviation of 1. After performing the scaling operation we start to process our data which included label encoding, Recursive Feature Elimination (RFE). Label encoding was done because in the dataset column "sample" contains values which are string (real, fake). So we did label encoding here where 1 means real and 0 means fake. Recursive Feature Elimination (RFE) is a feature elimination technique with a wrapper which is effective at eliminating those features (columns) in a training dataset that are least relevant in predicting the target value. We eliminated a number off less relevant features by using RFE, so our Final CSV dataset contained 200 most relevant features instead off 266 features that were present in the initial dataset.

3.5 Classifications

3.5.1 Long Short Term Memory (LSTM)

Long Short Term Memory networks, abbreviated as LSTM are a kind of RNN Which has the ability to memorise long-term dependencies from its previous layers. Hochreiter and Schmidhuber (1997) introduced them, and many others improved and promoted them in subsequent research [2]. They function fantastically well on a wide range of problems and are now extensively utilized. Recently, audio classification using LSTM has lately gained popularity and LSTM has demonstrated an outstanding accuracy rate in the classification problems. The model contains many layers that are engaged in the training and testing processes and for optimizing these layers different activation functions such as ReLU,amp and Soft Max are used. LSTM's are deliberately designed to avoid the problem of long-term reliance. Remembering knowledge over extended periods of time is essentially their default habit. Although considered a component of RNN, LSTM actually overcomes the limitations of RNN. LSTM can handle information in memory for a longer amount of time compared to RNN.

A fundamental knowledge of feed-forward neural networks is necessary to understand how LSTM improves over RNN. In a feed forward neural network, information only goes in one direction: from input layer towards output layer through the hidden layers. Data crosses across the network in a linear path, never passing a node more than once. Feed forward neural networks have limited memory of the data they receive and perform averagely in classification prediction. Because it only looks at the current input, a feed-forward network has no sense of temporal order. It simply cannot recollect anything other than its instruction from the past. Here's when RNN comes in handy,An RNN's information travels through a circle or loop. While making a conclusion, it considers both the present input and what it has learned from earlier inputs. Because of their internal memory, RNNs can retain crucial facts about the input they received, allowing them to predict what will happen next with remarkable accuracy. Rather of maintaining all inputs independent of one another, this allows the deep neural network to transmit information over multiple time stamps.

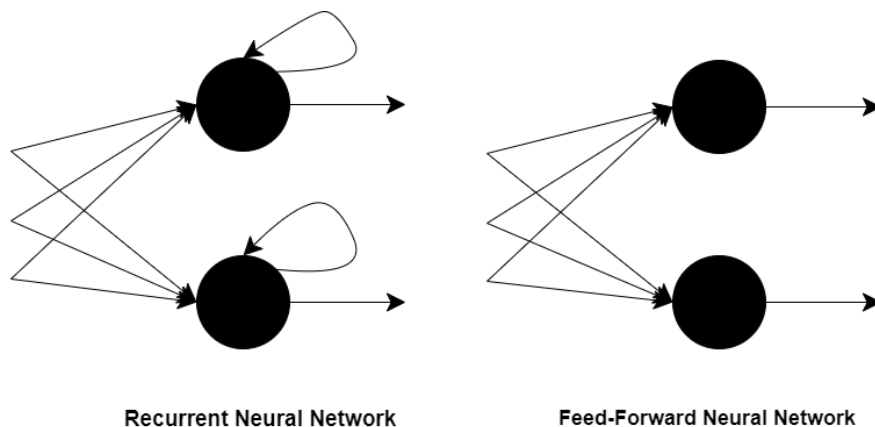


Figure 3.7: Difference between RNN and Traditional Feed Forward Neural Network

However, the traditional RNN suffers from vanishing/exploding gradients, which is a fundamental flaw. This particular flaw happens while back propagating the RNN during model training period, for networks with deeper layers this problem is more frequent. This happens because according to the chain rule gradients goes through continuous weight multiplications during back propagation, causing the gradient to either decline exponentially which is also known as vanishing gradient problem or increase exponentially which is known as exploding gradient. A gradient with a tiny value prevents the weights from upgrading which stops the model from learning/training more, on the other hand a huge gradient can drive the model to be unsteady. For these issues, RNNs are not able to deal with longer data sequences and preserve long-term dependencies, resulting in short-term memory; while LSTMs are a kind of RNN and operate similarly to normal RNNs, it is their gate mechanism that separates them. RNN's short-term memory problem is solved with this functionality. LSTMs solve the vanishing and inflating gradient concerns by including extra gates, such as input and forget gates, that provide the gradient flow more control and allow for better long-range dependency management.

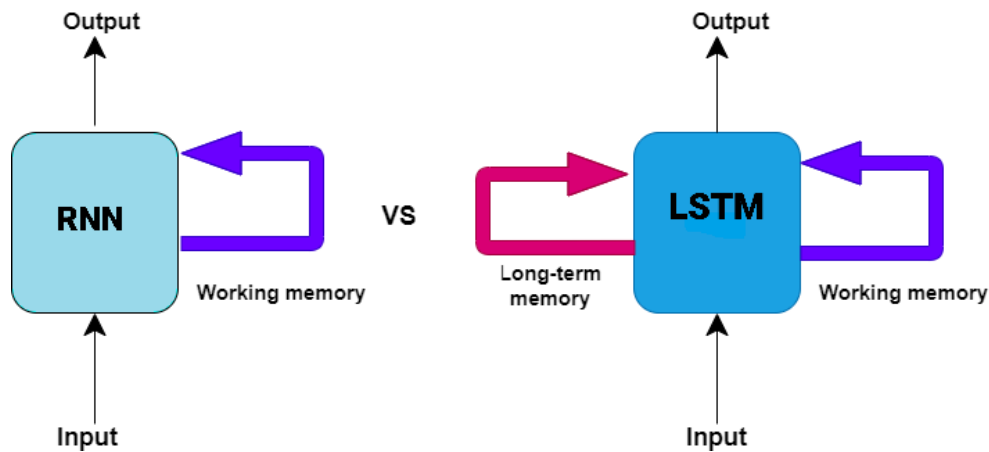


Figure 3.8: Difference between working mechanism of RNN and LSTM

LSTM Architecture

On a high level, LSTM functions similarly to an RNN cell. But internally LSTM has a slightly more complex structure. The LSTM is made up of three components and each one serves a specific purpose. The LSTM networks internal operation is seen below:

LSTMs are a kind of RNN and function in the same way as ordinary RNNs, their gating mechanism sets them apart. RNN's short-term memory problem is solved with this functionality. Additional gates, such as input and forget gates, are added to LSTMs to solve the vanishing gradient issues, which allows the model scope for better control over the gradient flow and memorising long-range dependencies. How these three gates work based on mathematical equation is briefly discussed below:

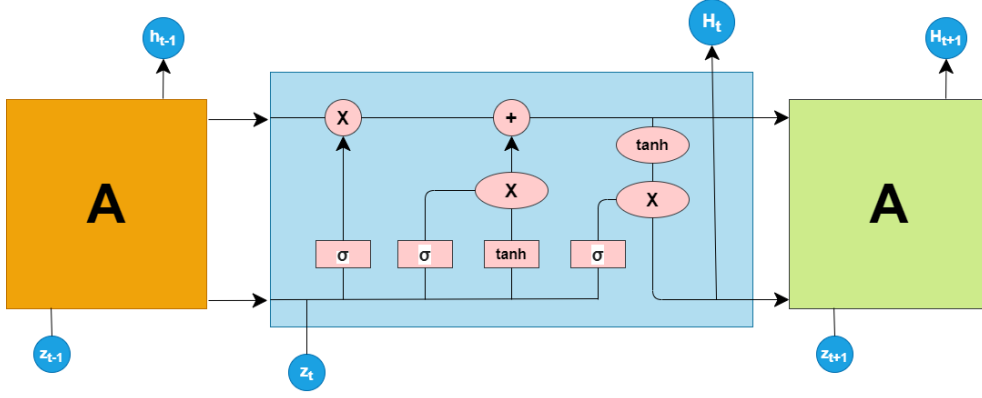


Figure 3.9: LSTM internal Architecture

- **Forget Gate:** First step in an LSTM cell is to determine whether it should keep or discard the information from the previous timestamp. The forget gate equation is as follows: (3.3)

$$f_{gt} = \sigma(z_t * U_a + H_{t-1} * W_a) \quad (3.3)$$

Here, z_t is the current timestamp's input, U_a is the input weight, H_{t-1} is the previous timestamp's hidden state, and W_a is the hidden state's weight matrix. An activation function is then applied to it. As a result, f_{gt} will be a value between 0 and 1. This f_{gt} is then multiplied by the preceding timestamp's cell state, as illustrated below(3.4):

$$C_{st-1} * f_{gt} = 0 \quad \dots \text{ if } f_{gt} = 0 \quad (3.4)$$

$$C_{st-1} * f_{gt} = C_{st-1} \dots \text{ if } f_{gt} = 1 \quad (3.5)$$

Here, C_{st-1} is cell state's current timestamp. If $f_{gt} == 0$ then it will forget everything and if $f_{gt} == 1$ it will forget nothing.

- **Input Gate:** Input gate decides which of new information should be stored in long-term memory. It only works with data from the present input and the short-term memory of the preceding time stamp. Here is the equation of the input gate(3.6):

$$I_{gt} = \sigma(z_t * U_g + H_{t-1} * W_g) \quad (3.6)$$

Here, U_g is the initial input weight, W_g is Weight of input co related with hidden state. Again, an activation function is applied over it. As a result, the value of I_g at the timestamp t will be between 0 and 1 and finally the information is going to be saved in long-term memory and utilized as an output is indicated by the final result of the input gate.

- **Output Gate:** Output gate's equation, is very much similar to the two previous gates discussed.(3.7)

$$O_{gt} = \sigma(z_t * U_0 + H_{t-1} * W_0) \quad (3.7)$$

value of output gate will also lie between 0 and 1 because of the activation function. In order to calculate the current hidden state O_{gt} and \tanh of the updated Hidden cell state(H_t) the equation is:(3.8)

$$H_t = o_t^* \tanh(C_{st}) \quad (3.8)$$

Finally, a choice is made regarding what will be the output. This result will be depending on the current condition of the cells. These gates' short-and long term memory will then be passed to the next cell, where the process will be repeated. Each time stamp's output may also be found in the short-term memory, also known as the hidden state.

3.5.2 Support Vector Machine (SVM)

Regression and Classification problems can be solved by using SVM.It's so much popular in machine learning for solving categorization tasks.

An SVM model is a representation of multiple classes in a hyperplane in multi-dimensional space. It will iteratively create the hyperplane in order to decrease the inaccuracy.The main objective of SVM is dividing the dataset into some classes for finding the best hyperplane.We can defined the support vectors as datapoints which are very close to the hyperplane and these points are more relavent for building up the classifiers properly and the hyperplane is a decision plane or space which divided a group of objects having different classes. The distance between two lines on a single data point of different classes is known as the margin.

SVM's basic goal is to segregate a dataset in the best feasible way and choose a hyperplane with the largest possible margin between support vectors in that dataset. To deal with nonlinear input spaces, SVM employs kernel techniques. A kernel transforms an input data space into the proper form.In other words,it converts non-separable issues into separable problems by adding more dimensions, which is particularly useful in non-linear separation problems.

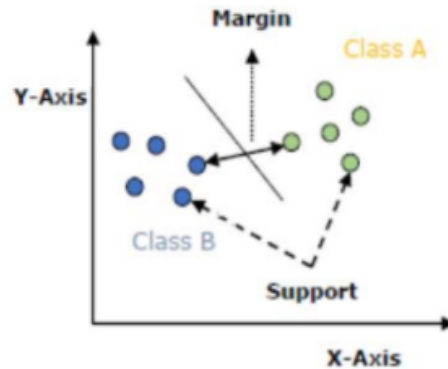


Figure 3.10: A simple representation of SVM working mechanism

There are various types of kernel functions that help to change the data dimension. Let's have a short discussion about the kernels.[3]

- **Linear Kernel:** The most basic type of kernel is linear kernel and it's own dimensional in nature. It's most commonly used for text-classification issues, as this type of categorization may be split linearly.(3.9)

$$k(y * y_i) = \text{sum}(y * y_i) \quad (3.9)$$

The product of two vectors (y, y_i) is the summation of multiplication on each input value pairs.

- **Polynomial Kernel:** We can describe the polynomial kernel as the more generalized version of the linear kernel which can tell whether the input space is curved or nonlinear.(3.10).

$$k(y, y_i) = 1 + \text{sum}(y * y_i)^d \quad (3.10)$$

Here, d is the degree of the polynomial and if the value of d is equal to 1 then it's the same as the linear kernel.

- **Gaussian Radial Basis Function(RBF):** It is the most common and favored kernel function in SVM, and it is typically used for nonlinear data. It assists in suitable separation when there is no prior knowledge of data and it can map input space into an infinitely dimensional space. (3.11)

$$k(y, y_i) = \exp(-\text{gamma} * \text{sum}(y - y_i^2)) \quad (3.11)$$

Gamma has a range of values from 0 to 1, with 0.1 being the most popular.

3.6 Explainable AI

In recent years, the huge field of Artificial Intelligence (AI) has undergone tremendous growth. With newer and more complicated models being released each year, AI models have begun to outperform human intelligence at a rate that no one could have imagined. However, as we obtain increasingly accurate and exact outcomes, it becomes more difficult to explain the reasoning behind the complicated mathematical decisions that these models make. This mathematical abstraction also does not assist people keep their trust in the decisions of a given model. This is where Explainable AI (also known as XAI) comes in. Explainable AI refers to tactics or methods that aid in explaining the decision-making process of a specific AI model. This freshly discovered branch of AI has immense potential, with more and more complex techniques being developed each year. SHAP (Shapley Additive explanations), DeepSHAP, DeepLIFT, CXplain and LIME are some of the most well-known XAI approaches. This report goes into great detail on LIME as we used LIME as our Explainable AI model to explain how our core models are making the predictions and to show which inputs are more important while predicting the output.

3.6.1 Local Interpretable Model-agnostic Explanations

Explainable AI approach LIME, which stands for local interpretable model-agnostic explanations, approximates any black box machine learning model with a local, interpretable model to explain each unique prediction.

Users must be able to comprehend AI models before they can trust it. AI interpretability reveals what's going on inside these systems and assists in the detection of potential issues such as data leakage, model bias, robustness, and causality. LIME is a broad framework for disclosing black boxes and explaining why AI-generated forecasts or recommendations are made, which aids in addressing the issue of how a model predicts its output. LIME modifies each unique data sample by gently altering feature values and evaluating the impact of each feature alteration on the prediction result. The main concept is to mimic a black-box model with a simpler glass-box model that is easier to comprehend locally. This method will be described in this chapter.

The idea is based on a work released in 2016 by [12], in which the writers intercept the original data points, input them into a black box model, and then evaluate the corresponding outputs. The extra data points are then weighted based on their

proximity to the original site. Finally, it fits a surrogate model, such as linear regression, to the dataset with changes using those sample weights. Each original data point may then be explained using the newly trained explanation model. The reasoning behind the LIME method is depicted in the diagram below. The colorful backdrop represents the original decision function, which is obviously nonlinear. The instance chosen to be explained is referred to as X . Initially, instances in the vicinity of X are simply disturbed and weighted based on their closeness to X . The sizes of the symbols circles and X indicate the weights here. In the neighborhood of X , it fits a linear model (black line) that approximates the model well. Lime then seeks to explain which features are impacting the forecasts the most by comparing it to the original model's prediction on these disturbed cases. Please keep in mind that the reasoning in this situation is only valid locally around instance X , not globally.

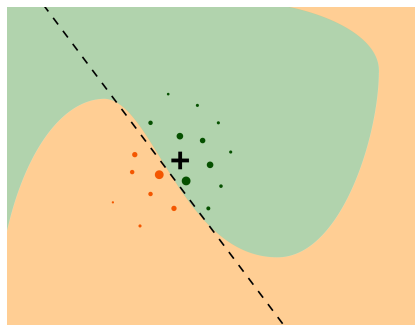


Figure 3.11: The idea behind local explanation methodology employed by LIME

The following general formula yields the explanation generated by LIME at a particular location x .(3.12):

$$\mathbf{Explanation}(x) = \underset{h \in H}{\operatorname{argmin}} L_f(f, h, \pi_x) + \Omega(h) \quad (3.12)$$

f can be denotes as a genuine function (known as ground truth), h denotes a surrogate function used to approximate f in the vicinity of x , and π_x denotes the locality. Various explanation families H , loss functions L_f , and complexity measures Ω can be utilized with this formulation. The assumption is that complexity and explainability are mutually exclusive. Normally, h would belong to the family of linear functions (low complexity), but this isn't always the case, and in the case of non-linear functions, the complexity $\Omega(h)$ rises and explainability falls. It is not easy to choose a complexity metric. It can be a degree of a polynomial function in some situations, but it can also be the calculation time for the original function f to determine the complexity. We can reduce the local mismatch between the original complex function f and the approximation function h using the loss function L_f .

The output type of lime is a list of explanation which enlighten us about the contribution of each features behind the predicted output. This allows for local interpretation and also indicates which features have the most impact behind predicting the result.

Chapter 4

Evaluation and Performance Analysis

4.1 Implementation

For Coding and implementing our proposed Model we used Python programming language. Python is the best fit for implementing most of the machine learning and AI-based algorithm because of its simplicity platform independence along with access to great libraries and frameworks for AI and machine learning (ML). Both ML and AI requires continuous data processing and Python's libraries easily let one to access, handle and transform data. From feature extraction till SVM implementation Libraries such as Scikit-Learn, Pandas, Matplotlib, Numpy, Librosa, Glob an open source deep learning library named keras, tensorflow etc. were consistently being used during all the stages.

4.1.1 LSTM Implementation

For implementing LSTM, we used Keras which is an publically available Deep Learning library that's written in Python. Keras wraps around the functionalities of other Machine Learning and Deep Learning libraries, including TensorFlow, Theano and CNTK.

To start implementing LSTM we used Sequential model, which basically refers to a simple stack of layers, but it only allows a single input, and a single output stacks of layers. In other words, a Sequential model is suited for a simple stack of layers with one input tensor and one output tensor for each layer. The first layer of our model is an LSTM layer with 200 memory units with Rectified Linear unit (ReLU) as activation function. The purpose of using ReLU as activation function is to allow the neural network to learn nonlinear dependencies and also return sequence is set as true. This is done to make sure that the following LSTM layer receives sequences rather than just random input.

A dropout layer with value of 0.01 is used after every LSTM layer to avoid overfitting of the model. The same process is repeated three more times and then, the output from the fourth LSTM layer is then given as input to a dense layer and finally, our last layer acts as a fully connected layer with an activation function of

'Sigmoid' .The main reason behind using sigmoid function is it activates between 2 points which is very useful for those model in which the probability is predicted as output.In our case the voice can be real or 1 and fake or 0 so sigmoid function is the best option here.

We also used Adam optimizer which is an algorithm for optimization technique for gradient descent with adaptive learning rate. Adam optimizer is helpful in reducing the gradient losses and to provide the most accurate results possible. After that the model is fit over epochs and batches, finally the score function of LSTMs or `lstm.score()` function is called for finding the accuracy of our implemented model.

Since LSTM works for sequential data, so window sliding the two seconds audio file and merging all the necessary extracted features for each slide can make the data sequential. Hence this would also make the dataset suitable for the LSTM model.However, we had 265 columns initially but if the audio file undergoes window sliding for 20 milliseconds we would have 6 slides for each audio files. Hence, each row in our dataset will have data of all the slides in a single row.

4.1.2 SVM Implementation

To utilize SVM,we've made the training and testing dataset. For training purpose we have used the python's sci-kit learn library which is built for solving SVM problem.However, we are doing the classification task meaning classify our data that's why we used SVC from sci-kit learns package and it only accepts the kernel type as an input. This parameter was set to linear in our SVM for kernelization. The SVC class's fit method is used to train the algorithm with the training data supplied as a parameter. The score function of the SVC class, or `svc.score`, is used to check the accuracy of our created model. Because it is the most frequently used metric for classification jobs, we employed the Confusion Matrix to further investigate the categorisation. To rapidly obtain the values for Confusion Matrix, utilize the confusion matrix function from Scikit-metrics Learn's package.

4.1.3 LIME Implementation

Reason behind using LIME is to explain the predictions of our LSTM and SVM Models.LIME works by picking a single instance in the validation dataset and gets their probability values for each feature. It also provides explanation as to the reason for assigning the probability.

For implementing LIME on our SVM model on python, an explainer class named 'LimeTabularExplainer' was used which explains predictions based on tabular data. It perturbs numerical features by sampling from a Normal (0,1) and executing the inverse operations of mean-centering and scaling depending on the training data's means and standard deviations. It also perturbs categorical features by sampling from the training distribution and producing a binary feature that is 1 when the value is the same as the case being described. Then for generating explanations

for a prediction, the `Explain_instance ()` method is used which generates neighbour data by randomly perturbing characteristics from the instance. Then it gets the neighboring data locations. Finally, using this neighboring data, it builds locally weighted linear models to describe each of the classes in an interpretable manner..

Then for creating Lime explainer on our main LSTM based model, a different class named ‘`RecurrentTabularExplainer`’ was used, which is an explainer for recurrent neural network models implemented in Keras/TensorFlow. This class simply just extends the `LimeTabularExplainer` class, reshaping the training data and feature names in such a manner that the training/testing data may be sent in precisely as it is in a recurrent neural network. `Explain_instance ()` method was also used here like previously.

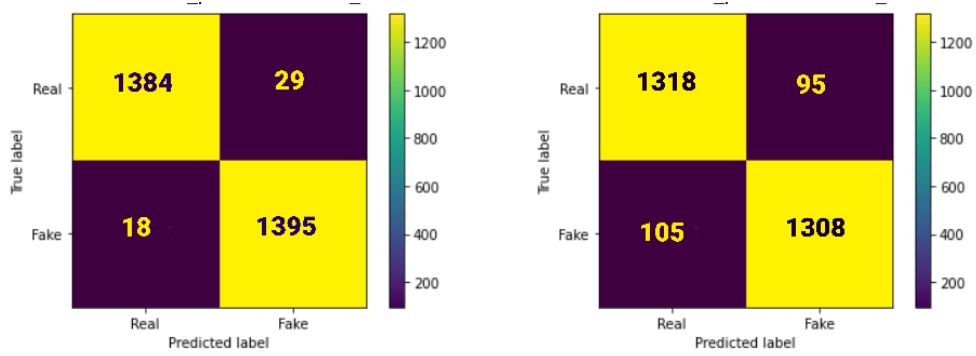
Finally, to visualize explanations created by LIME, `Show_in_notebook ()` function is called which shows the prediction value at left bar plot, each feature’s contribution to this prediction is shown in the right bar plot with colors assigned to signify the positive and negative impact of that feature on the target.

4.2 Result Analysis

After preprocessing the collected data and train it to our proposed model, now we need to evaluate the performance of our proposed model. In this purpose we have calculate the accuracy score,F-1 score, sensitivity score, specificity score and also plotted the confusion matrix for each of the classification model.

A confusion matrix is a simple and straightforward approach to illustrate a classifier’s prediction result. The matrix compares the actual target values to the machine learning model’s prediction which in result gives us a proper idea about how the model is working. However, the four basic things of a confusion matrix are described bellow:

- **True Positive:** This is the case when the predicted value matches with the actual value and both of the values are positive. In our case it means that our proposed model predicts the sample voice as a real voice and the sample voice is actually real.
- **True Negative:** In this case the predicted value also matches with the actual value and both the values are negative. This means that our model is predicted the voice sample as a fake voice and the sample voice is actually fake.
- **False Positive:** Actual value doesn’t matches with the predicted values. This is the case when the model predicted the voice sample as a real voice but the voice is actually a fake voice.
- **False Negative:** This is the case when the model predicted the sample voice as a fake voice but the sample voice is actually real.



(a) Confusion Matrix of our proposed LSTM based RNN model

(b) Confusion Matrix for SVM model

Figure 4.1: Confusion Matrix for LSTM and SVM model

Our Testing dataset was consist of a total 2826 audio files.If we look at the confusion matrix for the LSTM based RNN model we can see the TP value is 1384 meaning the sample input data was a real voice and our model is also claiming the voice as a real voice, the TN value is 1395 meaning sample input is a fake voice and our model also detect it as a fake voice. The FP value is 29 which means there are 29 cases where the input and output value mismatched.Our model detect this voice sample as a real voice but the sample voice is actually fake. The FN value is 18,in this case our model predicted the sample as a fake voice but the sample is actually real.On the other hand, for SVM model the TP and TN values are 1318 and 1308, FP value is 95 and the FN value is 105. So if we compare this two confusion matrix we can see our LSTM based RNN model performs quite well.

We can calculate the accuracy scores by dividing the total correct prediction made by the model by the total number of predictions meaning every possible decision which can be made by the model.It describes how regular the model anticipates accurate outputs and gives the method for assessing the accuracy.We can calculate accuracy by using the formula (4.1)

$$\text{Accuracy} = \frac{(TP + TN)}{(TP + FP + TN + FN)} \quad (4.1)$$

The F-score, also known as the F-1 score is a measure for how accurate a model is on a given dataset.It's a method of combining the model's precision and recall, and it's defined as the harmonic mean of the precision and recall.(4.4). Here Precision is the number of True Positive divided by the number of True Positive and Number of

False Positive (4.2) and we can calculate Recall by using formula (4.3). The number of true positive divided by the number of true positive and false negative.

$$Precision = \frac{TP}{TP + FP} \quad (4.2)$$

$$Recall = \frac{TP}{TP + FN} \quad (4.3)$$

$$F - Score = \frac{2Precision \cdot Recall}{Precision + Recall} \quad (4.4)$$

On the other hand, Sensitivity refers to the measure of the proportion of actual positive cases that got predicted as positive (True Positive) and Specificity refers to the proportion of actual negative which got predicted as Negative (True Negative). Table 4.1 shows these scores in detail:

Scores	Classifiers	
	SVM	LSTM
Accuracy	92.92%	98.33%
F-1	92%	98%
Sensitivity	92%	98%
Specificity	93%	99%

Table 4.1: Calculated Scores for LSTM and SVM

From the above table it's clear that our proposed LSTM based model have better score in every sectors than SVM classifier.

A receiver operating characteristic curve is a graph that shows the classification model's performance over all categorization threshold. It has 2 parameters.

- True Positive Rate
- False Positive Rate

True Positive Rate: TPR is the synonym for recall and it can be defined as follow (4.5):

$$TPR = \frac{TP}{TP + FN} \quad (4.5)$$

False Positive Rate: For calculating FPR we will use(4.6):

$$FPR = \frac{FP}{FP + TN} \quad (4.6)$$

An efficient, sorting-based technique called AUC (Area under the ROC curve) is used to compute the points in a ROC curve. It takes into account the full two-dimensional area beneath the ROC curve. AUC is a scale that ranges from 0 to 1. However, the model that has an AUC value 1 is 100 percent accurate and the model which has AUC value 0 is 100 percent inaccurate. We have also explained the predicted outputs of LSTM based model and SVM model by using a technique of Explainable AI called Lime. Local model interpretability is provided by Lime. It

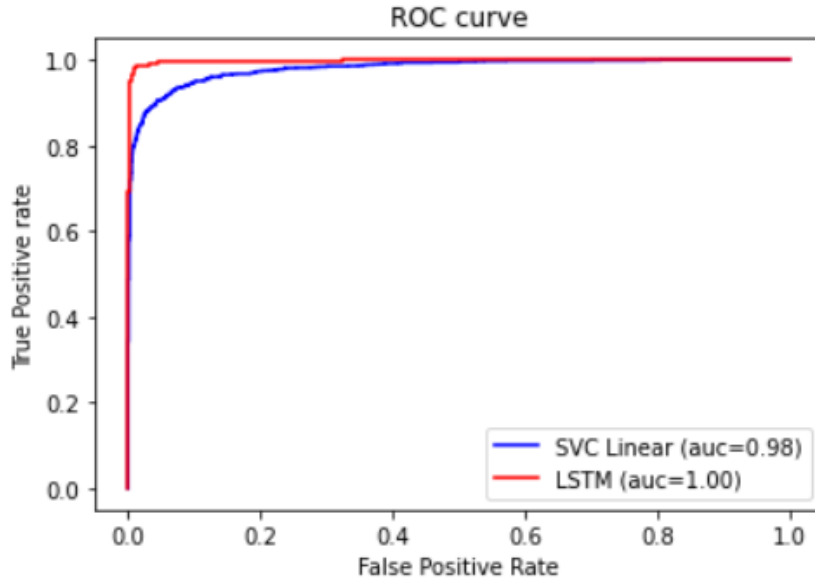


Figure 4.2: ROC curve for LSTM and SVM

modifies feature values in a single data sample before looking at the impact on the outcome. The result is shown in the left bar plot and each of the features contribution to predict the result is showed in the right bar plot with the color associated in it and it's showing about the contribution of each features to predict the result.

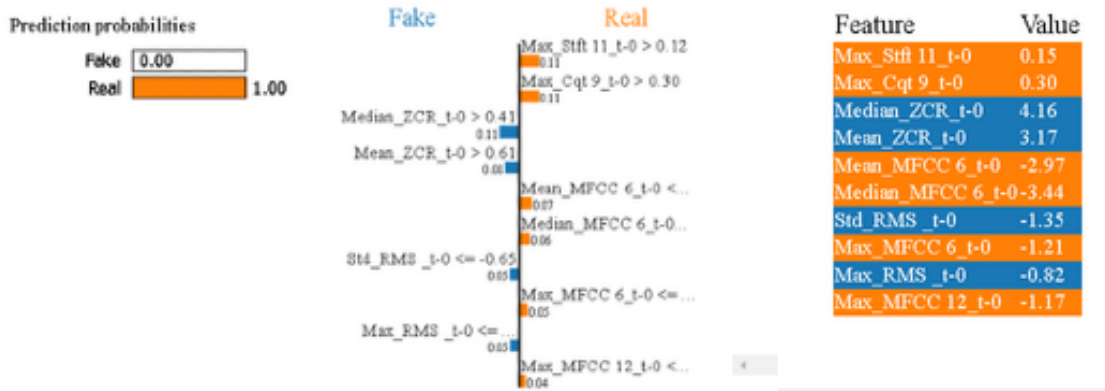


Figure 4.3: Explaining LSTM model's prediction result using LIME

Figure 4.3 shows lime explanation for LSTM based RNN model. Note that the sample voice was a real voice and lime is showing based on which features and how much contribution each features have behind predicting the result. Figure 4.4 shows the explanation behind predicting the output for SVM model. The contribution of each features behind predicting the output of SVM is shown in the figure.

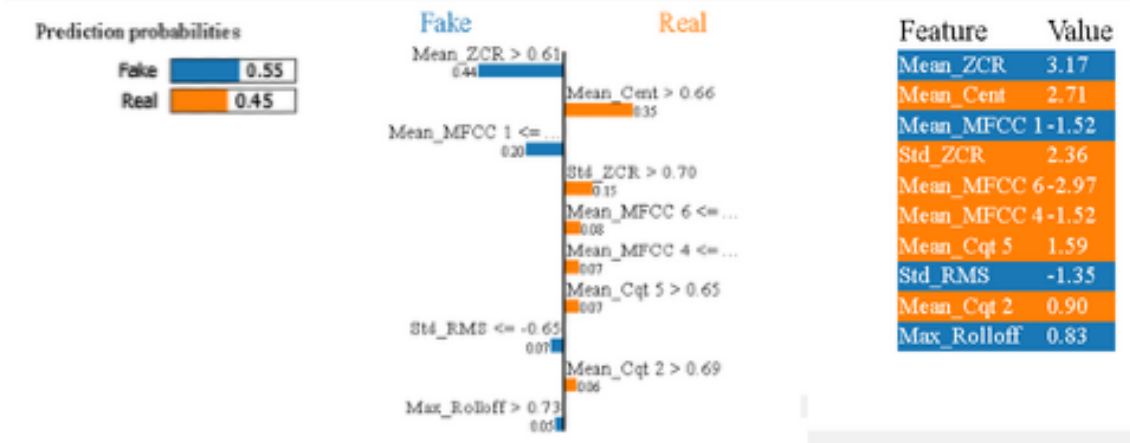


Figure 4.4: Explaining SVM model's prediction result using LIME

4.3 Comparative Analysis

We try to compare our study to other similar studies in this area. The accuracy scores were used to perform the comparison investigation. We used the publically available FoR dataset as the foundation for our learning model. We have found some model trained based on this dataset [27] and also found some reasearch related to voice detection based on other datasets [26] [29].

A comparison may be seen in the table below based on the accuracy. Several of the studies included in the table used various preprocessing and extraction methods. We've tried to extract all necessary features of voice and pre-processed our dataset in order to achieve the maximum level of result in our work. The table demonstrates that the techniques in our suggested model approach are more efficient and produce better outcomes than any previous study.

Existing Works	Classifier	Dataset	Accuracy Score	Explainability
Recardo et al[27]	Naive Bayes	FoR	67.27%	No
Recardo et al[27]	SVM	FoR	73.46%	No
Recardo et al[27]	Decision Tree(J48)	FoR	70.26%	No
Recardo et al[27]	Random Forest	FoR	71.47%	No
Neelima et.al[26]	CNN based	Own generated	67%	No
Wijethunga et.al[29]	CNN and RNN based	Fake-Or Real	94%	No
The proposed lstm model	LSTM based	FoR	98.33%	Yes

Table 4.2: Accuracy comparison between different classifiers

Chapter 5

Conclusion

Voice detection technology is capable of detecting between fake or real voices that are made from advanced voice synthesis technologies. As advanced AI-synthesized techniques are capable of producing extremely realistically sounding voices, it also raises security and privacy concerns to everyone. So, we proposed a model which can detect real and fake voices by using LSTM based RNN model and Explainable AI method.

In this paper, we have listed and examined various characteristics for impersonated voice identification. We have provided an overview of known techniques with the goal of categorizing features by speech properties that are used. For our research we used FoR dataset in which we got raw audio files containing both real human voice samples from various sources and also AI synthesized fake voices using latest techniques. In order to establish our model and train data, we need to extract features from the audio files and convert them, so that our proposed model could use them for training and testing purpose. After extracting features from voice samples, we combined features from various speech properties to get more accuracy to detect fake or real voices. Our analyses showed that earlier researchers did not take the vanishing gradient problem of voice input into account. The LSTM can erase, write and read data from the cell and also the LSTM architecture makes it easier for the RNN to preserve data across multiple time-steps. As the input audio files are obtained in time domain, so our proposed model based on LSTM based RNN model can easily track every bit of information from voice inputs to the cell. Thus, we are able to solve the vanishing gradient problem.

We also build a SVM classifier to compare the result we got from our proposed model. The SVM classifier showed less accuracy than our proposed LSTM based model. We calculated the F-1 score of both models which can measure the accuracy of a model on a dataset. The F-1 score showed promising percentage of our proposed model. From which we can understand that our proposed model can successfully differentiate between fake or real voices. We also calculated the accuracy, sensitivity and specificity score where in every score the percentage is high in LSTM than the SVM classifier. Moreover, it would seem logical that knowing the predicted output of specific predictions would help us decide whether to trust or distrust the prediction, or the classifier as a whole. So, we explained our predicted output using Explainable AI method called Lime which can adjust important feature values from every indi-

vidual data and explains the contributions of each features. The FoR dataset have used in various voice synthesis model [TABLE 4.2], but the accuracy we got from our proposed model is 98.33% which is way higher than the previous models. Also We implemented LIME in order to receive an explanation from our LSTM models prediction, so that a more detailed analysis should be performed in future works. The results of our research suggest that our analysis can be used in a larger scale to prevent the crimes by that takes place by misusing impersonated voice.

Bibliography

- [1] C. M. Bishop, “Mixture density networks,” 1994.
- [2] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
- [3] T. Evgeniou and M. Pontil, “Support vector machines: Theory and applications,” in *Advanced Course on Artificial Intelligence*, Springer, 1999, pp. 249–257.
- [4] H. Zen, T. Nose, J. Yamagishi, S. Sako, T. Masuko, A. W. Black, and K. Tokuda, “The hmm-based speech synthesis system (hts) version 2.0.,” in *SSW*, Citeseer, 2007, pp. 294–299.
- [5] P. Taylor, *Text-to-speech synthesis*. Cambridge university press, 2009.
- [6] S. Kang, X. Qian, and H. Meng, “Multi-distribution deep belief network for speech synthesis,” in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, IEEE, 2013, pp. 8012–8016.
- [7] Z.-H. Ling, L. Deng, and D. Yu, “Modeling spectral envelopes using restricted boltzmann machines and deep belief networks for statistical parametric speech synthesis,” *IEEE transactions on audio, speech, and language processing*, vol. 21, no. 10, pp. 2129–2139, 2013.
- [8] J. C. Lucero, J. Schoentgen, and M. Behlau, “Physics-based synthesis of disordered voices,” in *Interspeech*, Citeseer, 2013, pp. 587–591.
- [9] M. Sahidullah, T. Kinnunen, and C. Hanilçi, “A comparison of features for synthetic speech detection,” 2015.
- [10] Z. Wu, T. Kinnunen, N. Evans, J. Yamagishi, C. Hanilçi, M. Sahidullah, and A. Sizov, “Asvspoof 2015: The first automatic speaker verification spoofing and countermeasures challenge,” in *Sixteenth annual conference of the international speech communication association*, 2015.
- [11] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, “Wavenet: A generative model for raw audio,” *arXiv preprint arXiv:1609.03499*, 2016.
- [12] M. T. Ribeiro, S. Singh, and C. Guestrin, ““ why should i trust you?” explaining the predictions of any classifier,” in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 2016, pp. 1135–1144.
- [13] X. Tian, X. Xiao, E. S. Chng, and H. Li, “Spoofing speech detection using temporal convolutional neural network,” in *2016 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA)*, IEEE, 2016, pp. 1–6.

- [14] Z. Wu, P. L. De Leon, C. Demiroglu, A. Khodabakhsh, S. King, Z.-H. Ling, D. Saito, B. Stewart, T. Toda, M. Wester, *et al.*, “Anti-spoofing for text-independent speaker verification: An initial database, comparison of countermeasures, and human performance,” *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 24, no. 4, pp. 768–783, 2016.
- [15] H. Yu, A. Sarkar, D. A. L. Thomsen, Z.-H. Tan, Z. Ma, and J. Guo, “Effect of multi-condition training and speech enhancement methods on spoofing detection,” in *2016 First International Workshop on Sensing, Processing and Learning for Intelligent Machines (SPLINE)*, IEEE, 2016, pp. 1–5.
- [16] R. Li, Z. Wu, X. Liu, H. Meng, and L. Cai, “Multi-task learning of structured output layer bidirectional lstms for speech synthesis,” in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2017, pp. 5510–5514.
- [17] H. Muckenhirn, M. Magimai-Doss, and S. Marcel, “End-to-end convolutional neural network-based voice presentation attack detection,” in *2017 IEEE international joint conference on biometrics (IJCB)*, IEEE, 2017, pp. 335–341.
- [18] D. Paul, M. Pal, and G. Saha, “Spectral features for synthetic speech detection,” *IEEE journal of selected topics in signal processing*, vol. 11, no. 4, pp. 605–617, 2017.
- [19] Y. Wang, R. Skerry-Ryan, D. Stanton, Y. Wu, R. J. Weiss, N. Jaitly, Z. Yang, Y. Xiao, Z. Chen, S. Bengio, *et al.*, “Tacotron: Towards end-to-end speech synthesis,” *arXiv preprint arXiv:1703.10135*, 2017.
- [20] H. Yu, Z.-H. Tan, Z. Ma, R. Martin, and J. Guo, “Spoofing detection in automatic speaker verification systems using dnn classifiers and dynamic acoustic features,” *IEEE transactions on neural networks and learning systems*, vol. 29, no. 10, pp. 4633–4644, 2017.
- [21] C. Zhang, C. Yu, and J. H. Hansen, “An investigation of deep-learning frameworks for speaker verification antispoofing,” *IEEE Journal of Selected Topics in Signal Processing*, vol. 11, no. 4, pp. 684–694, 2017.
- [22] Y. Gao, R. Singh, and B. Raj, “Voice impersonation using generative adversarial networks,” in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2018, pp. 2506–2510.
- [23] D. Harwell, *An artificial-intelligence first: Voice-mimicking software reportedly used in a major theft*, Sep. 2019. [Online]. Available: <https://www.washingtonpost.com/technology/2019/09/04/an-artificial-intelligence-first-voice-mimicking-software-reportedly-used-major-theft/>.
- [24] X. Yuan, P. He, Q. Zhu, and X. Li, “Adversarial examples: Attacks and defenses for deep learning,” *IEEE transactions on neural networks and learning systems*, vol. 30, no. 9, pp. 2805–2824, 2019.
- [25] Amazon corporation. 2020. *amazon aws polly*, 2020. [Online]. Available: <https://aws.amazon.com/polly/>.
- [26] M. Neelima and I. Santiprabha, “Mimicry voice detection using convolutional neural networks,” in *2020 International Conference on Smart Electronics and Communication (ICOSEC)*, IEEE, 2020, pp. 314–318.

- [27] Ricardo and Vassilios, *For: A dataset for synthetic speech detection*, 2020. [Online]. Available: <http://bil.eecs.yorku.ca/publications/>.
- [28] *Text-to-speech: Lifelike speech synthesis — google cloud*, 2020. [Online]. Available: <https://cloud.google.com/text-to-speech>.
- [29] R. Wijethunga, D. Matheesha, A. Al Noman, K. De Silva, M. Tissera, and L. Rupasinghe, “Deepfake audio detection: A deep learning based solution for group conversations,” in *2020 2nd International Conference on Advancements in Computing (ICAC)*, IEEE, vol. 1, 2020, pp. 192–197.