VoiceAuth: Deepfake Voice Detection Using Neural Networks

line 1: 1st Given Name Surname
line 2: *dept. name of organization (of Affiliation)*
line 3: *name of organization (of Affiliation)*
line 4: City, Country
line 5: email address or ORCID

line 1: 2nd Given Name Surname
line 2: *dept. name of organization (of Affiliation)*
line 3: *name of organization (of Affiliation)*
line 4: City, Country
line 5: email address or ORCID

line 1: 3rd Given Name Surname
line 2: *dept. name of organization (of Affiliation)*
line 3: *name of organization (of Affiliation)*
line 4: City, Country
line 5: email address or ORCID

line 1: 4th Given Name Surname
line 2: *dept. name of organization (of Affiliation)*
line 3: *name of organization (of Affiliation)*
line 4: City, Country
line 5: email address or ORCID

line 1: 5th Given Name Surname
line 2: *dept. name of organization (of Affiliation)*
line 3: *name of organization (of Affiliation)*
line 4: City, Country
line 5: email address or ORCID

line 1: 6th Given Name Surname
line 2: *dept. name of organization (of Affiliation)*
line 3: *name of organization (of Affiliation)*
line 4: City, Country
line 5: email address or ORCID

***Abstract*— The amount of deepfake videos and voice messages in internet is extremely huge. This number of voice messages has the potential to mislead people. Consider a video in which social influencer misleads a group people, or deepfake video of Prime Minister or President of country declaring war on other country the social and economical impact would unimaginably large. In the era of Artificial intelligence it is impossible to stop generating deepfake videos or reduce those from internet. So, we have developed a neural network which would identify the deepfake videos or voice messages using Convolutions Recurrent neural network (RCNN) consisting of few convolution layers and recurrent layers of Long-Short term memory (LSTM) and a finally a dense layer with sigmoid activation function which is responsible for classification by producing probability score. These networks are trained using datasets of Kaggle's deepfake detection challenge. The dataset given us the audio files to extract the features like Mel-Frequency Cepstral Coefficients, Intonation, Rhythm, Speech rate, Formants. These features are extracted and added in training dataset, the testing of the network is done using Kaggle deepfake detection challenge where it has more than (470GB) very huge for network to handle. The network achieved the metrics before training like validation accuracy (how it performs for unseen data) is 50.01%, validation F1 score of 99.01%, accuracy 98.85% and loss of 3.26%. The same network on 20th epoch achieved impressively high metrics like loss of 3.21%, accuracy 98.85%, validation loss of 2.85%, validation accuracy of 99.02%, test accuracy of 99.11% and test loss of 2.85%. The network performed extremely well higher accuracy cannot achieved that is 100%, which may lead to overfitting of model on dataset.**

Keywords—CNNs, RNNs, Combination of CNN and RNN, CRNNs, neural network, LSTM, deepfake, deep-voice, MFCCs, formants, deepfake detection challenge.

I. Introduction

In recent years, the proliferation of deepfake technology has raised significant concerns regarding its potential misuse in various domains, including audio manipulation. Deepfake audio, generated using sophisticated machine learning models, can mimic the voice of an individual with remarkable accuracy, often leading to misinformation, fraud, and privacy breaches. In response to this emerging threat, the development of robust techniques for detecting deepfake audio has become imperative.

The project aims to address this challenge by leveraging Convolutional Recurrent Neural Networks (CRNNs) to detect deepfake audio. CRNNs are a powerful class of deep learning models capable of capturing both spatial and temporal features, making them well-suited for sequential data such as audio signals. By combining convolutional layers for feature extraction with recurrent layers for sequence modeling, CRNNs can effectively discern patterns indicative of deepfake audio.

The primary objective of this project is to train a CRNN model on a dataset of audio samples, each labeled as either authentic or deepfake. The dataset comprises various features extracted from the audio, including chroma features, root mean square (RMS) amplitude, spectral centroid, spectral bandwidth, and Mel-frequency cepstral coefficients (MFCCs). These features serve as inputs to the CRNN model, which learns to distinguish between genuine and manipulated audio recordings.

The project encompasses several key components, including data preprocessing, model development, training, evaluation, and performance analysis. We employ Python programming language along with libraries such as TensorFlow and Keras for implementing the CRNN model and conducting experiments. Additionally, we utilize tools for data visualization and performance metrics calculation to assess the efficacy of the developed model.

By the project's conclusion, we aim to achieve a CRNN model capable of accurately detecting deepfake audio with high precision and recall. The successful implementation of such a model holds significant implications for combating the spread of misinformation and safeguarding the integrity of audio content across various platforms and applications.

II. Literature Survey

A. *Audio processing:*

1. ***Fundamentals of Music Processing: Audio, Analysis, Algorithms, Applications by Meinard Müller:***
   This comprehensive book covers various aspects of audio processing, including feature extraction, audio analysis, and applications such as music information retrieval.
2. ***Speech and Audio Signal Processing: Processing and Perception of Speech and Music by Ben Gold and Nelson Morgan:***
   This book provides an in-depth understanding of both speech and audio signal processing, covering topics such as spectral analysis, filter design, and speech recognition.
3. ***Discrete-Time Signal Processing by Alan V. Oppenheim and Ronald W. Schafer:***
   While not exclusively focused on audio processing, this classic book is a cornerstone for understanding digital signal processing, including topics like sampling, filtering, and spectral analysis, which are fundamental to audio processing.

B. *Convolutional Neural Networks(CNNs):*

1. ***Deep Learning by Ian Goodfellow, Yoshua Bengio, and Aaron Courville:***
   This seminal book offers a comprehensive introduction to deep learning, including chapters dedicated to CNNs. It provides both theoretical insights and practical guidance on implementing CNNs for tasks like image recognition.
2. ***Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow by Aurélien Géron:***
   This book offers practical examples and exercises, including sections on CNNs, making it suitable for those looking to apply CNNs in real-world projects.
3. ***Convolutional Neural Networks in Visual Computing by Jürgen Gall, Ed.:***
   This book focuses specifically on the application of CNNs in visual computing tasks, including image classification, object detection, and segmentation, providing insights into both theory and practical implementation.

C. *Recurrent Neural Networks(RNNs):*

1. ***Recurrent Neural Networks for Short-Term Load Forecasting: An Overview by Dimitrios Vagropoulos and Vassilis G. Agelidis:***
   This book focuses on the application of RNNs in short-term load forecasting, providing insights into both theoretical concepts and practical implementations.
2. ***Sequence Learning: Paradigms, Algorithms, and Applications by Sumit Chopra, Raia Hadsell, and Yann LeCun:***
   While not exclusively focused on RNNs, this book covers sequence learning paradigms, including RNNs, LSTM networks, and their applications in various domains such as natural language processing and time series analysis.
3. ***"Recurrent Neural Networks in Atmospheric Remote Sensing" by Samarendra K. Mohanty:***
   This book explores the application of RNNs in atmospheric remote sensing, offering insights into how RNN architectures can be adapted for time-series data analysis in this domain.

III. Dataset collection

A. *Kaggle's deepfake detection challenge:*

The dataset collected is enormously large such that it is 470GB made up of .mp4 files. We have extracted .mp3 from those files. These data is to train the network.

B. *Google AVA dataset:*

The Audio Visual Actions (AVA) dataset by Google includes a wide variety of YouTube video clips with audio tracks, some of which may contain manipulated or synthesized audio.

C. *Common Voice dataset by Mozilla:*

This dataset by Mozilla contains thousands of hours of speech data contributed by volunteers. It includes recordings in multiple languages and accents. This dataset inclusion added the robust nature for the network because of multilingual nature of dataset and different accents.

D. *VoxCeleb dataset:*

VoxCeleb is a large-scale dataset containing audio samples of celebrities speaking from various sources such as interviews, podcasts, and speeches. This dataset helped the network to identify the voice of the celebrity easily without any complex oprations.

IV. Feature Extraction From Datasets

A. *Mel-Frequency cepstral co-efficients(MFCCs):*

Mel frequency Cepstral Coefficients (MFCCs) are widely used in audio signal processing. They are used in the processes like speech and speaker recognition systems.

1) Mel Scale:

The Mel Scale is a perceptual scale of pitches that approximates the human ear's response to different frequencies. It is nonlinear and is based on the perceived difference in pitch between tones.

2) Cepstral Analysis:

Cepstral analysis is used to separate the information about the vocal tract system from other aspects of the speech signal.

3) MFCCs:

MFCCs are coefficients that collectively represent the short-term power spectrum of a sound. They are derived from the Mel frequency scale of the power spectrum of the signal.

4) MFCC1:

Represents the overall energy or loudness of the audio signal. It's akin to the zeroth coefficient of a Discrete Cosine Transform (DCT), capturing the average energy of the signal.

5) MFCC2 - MFCC13:

These coefficients typically represent the spectral envelope of the audio signal. They capture the shape of the short-term power spectrum of the signal, with lower coefficients generally representing broader features (such as pitch or formants) and higher coefficients representing finer details.

6) MFCC14 - MFCC20:

These coefficients are often called "higher-order" coefficients. They capture additional information beyond the spectral envelope, such as variations and dynamics in the audio signal. They can represent aspects like vocal tract resonances and other temporal dynamics.

These MFCCs are extracted from dataset by using the python library [1] librosa which used for audio file processing. It has a feature called mfcc and setting its parameter num_mfcc to 20 we can extract up to 20 features on mfcc only.

B. Chroma Short-Time Fourier Tranform(Chroma STFT):

Chroma Short-Time Fourier Transform (Chroma STFT) is a representation of audio signals in the chroma domain, where the pitch content or tonal information is emphasized. It is derived from the Short-Time Fourier Transform (STFT) of the audio signal, which represents the signal's frequency content over time.

1) Short-Time Fourier Transform(STFT):

STFT is a technique used to analyze the frequency content of a signal over short, overlapping time windows. It involves dividing the signal into short segments, applying the Fourier Transform to each segment, and then combining the results.

2) Chroma:

In music, chroma refers to the set of 12 distinct pitch classes (corresponding to the 12 notes in the Western musical scale). Chroma features capture the distribution of these pitch classes in the audio signal, regardless of their octave.

3) Chroma STFT:

Chroma STFT is obtained by first computing the STFT of the audio signal and then summarizing the frequency content within each time frame into chroma features. This is typically achieved by mapping the frequency bins of the STFT to the 12 pitch classes, often using a weighted sum or template matching approach.

Continuous short-time Fourier transform:

$$\mathbf{STFT}\{x(t)\}(\tau,\omega) \equiv X(\tau,\omega) = \int_{-\infty}^{\infty} x(t)w(t-\tau)e^{-i\omega t}\,dt$$

Discrete short-time Fourier transform:

$$\mathbf{STFT}\{x[n]\}(m,\omega) \equiv X(m,\omega) = \sum_{n=-\infty}^{\infty} x[n]w[n-m]e^{-i\omega n}$$

Chromagram features of short-time Fourier transforms can be obtained by the formula:

$$CG(m, k) = |X(m, k)|$$

where X – Short-time Fourier Transform.

Applications:

Music Information Retrieval: Chroma features are commonly used in tasks such as music genre classification, chord recognition, and melody extraction.

Audio Analysis: Chroma STFT can also be useful in audio analysis tasks where tonal content is important, such as speech processing and environmental sound recognition.

The average of 12 Chroma STFT is identified using the python library librosa and added to the features of the datasets.

C. Spectral Centroid:

The spectral centroid is a measure used in digital signal processing to characterize the center of mass or the average frequency of a signal. It provides valuable information about the spectral distribution of the signal.

1) Calculation of Spectral Centroid:

a) Compute to frequency domain:

Use a Fourier Transform (usually the Short-Time Fourier Transformer STFT) to convert the audio signal from the time domain to frequency domain.

b) Compute the Spectral Centroid:

The spectral centroid is calculated as the weighted mean of the frequencies present in the power spectrum, where the weights are given by the magnitudes of the frequencies.

$$\mathrm{SC}(m) = \frac{\sum_k k \cdot \mathrm{SG}(m,k)}{\sum_k \mathrm{SG}(m,k)}.$$

D. Spectral Bandwidth:

Spectral bandwidth is important feature in digital signal processing that provides information about the spread or width of the frequency content in a signal's power spectrum. It quantifies the range of frequencies present in the signal and can be used to characterize the overall spectral distribution.

1) Calculation of Spectral Bandwidth:

Spectral bandwidth is typically calculated as a measure of the spread of the power spectrum around the spectral centroid. It can be computed using various methods, such as the standard deviation of the frequencies around the centroid, or by fitting a Gaussian curve to the power spectrum and measuring its width.

$$\mathrm{SB}(m) = \sqrt{\frac{\sum_k (k - \mathrm{SC}(m))^2 \cdot \mathrm{SG}(m,k)}{\sum_k \mathrm{SG}(m,k)}}.$$

E. *Spectral Roll-Off:*

The spectral roll-off (or roll-off) is spectral feature used in digital signal processing to characterize the shape or distribution of the power spectrum of an audio signal. It indicates the frequency below which a certain percentage of the total spectral energy lies.

1) *Calculatios:*

a) Determine the Percentage:

Typically, spectral roll-off is calculated as the frequency below which a certain percentage (e.g., 85%) of the total spectral energy lies.

b) Find the Rolloff Frequency:

The rolloff frequency is then determined as the frequency below which the specified percentage of the total spectral energy is contained.

F. *Zero Crossing Rate:*

Zero crossing rate is a simple yet effective feature used in digital signal processing to characterize the rate at which a signal changes its sign. It is commonly employed in audio and speech processing applications to extract information about the temporal characteristics of the signal, such as its periodicity and energy distribution.

1) *Calculations of ZCR(Zero Crossing Rate):*

a) Discretization:

The continuous audio signal is discretized into frames of a fixed duration.

b) Counting Zero Crossings:

Within each frame, the number of times the signal crosses the zero amplitude line (i.e., changes sign) is counted.

c) Normalization:

The count of zero crossings is often normalized by dividing it by the length of the signal frame or the signal duration.

$$\mathrm{ZCR}(m) = \frac{1}{2N} \sum_{n=1}^{N-1} |sgn(x[n]) - sgn(x[n-1])|.$$

G. *Root Mean Square Value (RMS):*

The Root Mean Square (RMS) value of an audio signal is a measure of its overall amplitude or energy content. It is a commonly used feature in audio signal processing to quantify the loudness or power of the signal.

1) *Calculations of RMS value:*

a) Squared Amplitude:

The amplitude of each sample in the audio signal is squared.

b) Mean Calculation:

The mean (average) of the squared amplitudes is computed.

c) Square Root:

The square root of the mean squared amplitude is taken to obtain the RMS value.

$$\mathrm{RMS}(m) = \sqrt{\frac{1}{N} \sum_{n=0}^{N-1} x[n]^2}.$$

H. *Intonation:*

Intonation in the context of an audio signal typically refers to the variation in pitch or the melody contour of speech or musical sounds. It's a fundamental aspect of communication and music, contributing to the conveyance of emotions, meaning, and expression.

1) *Calculating Intonation:*

a) Pitch Tracking:

Intonation analysis often involves tracking the pitch or fundamental frequency (F0) of the signal over time. This can be done using techniques such as autocorrelation, cepstral analysis, or using dedicated pitch detection algorithms.

b) Melodic Contour:

Intonation analysis may also focus on extracting the melodic contour of the signal, capturing the rising and falling patterns of pitch changes.

I. *Rythym:*

Analyzing the rhythm of an audio file involves identifying and characterizing its temporal patterns, such as the beat, tempo, and rhythmic structure. This analysis is essential in music processing, dance analysis, and various audio-based applications.

1) *Rythym Calculations:*

a) Beat Tracking:

Detecting the underlying beat or pulse of the music, which often forms the basis of the rhythm. This can involve techniques such as onset detection, autocorrelation, or dynamic programming.

b) Tempo Estimation:

Estimating the tempo or speed of the music, typically measured in beats per minute (BPM).

c) Rhythmic Patterns:

Identifying recurring rhythmic patterns, such as drum patterns or rhythmic motifs, which contribute to the overall feel of the music.

J. *Speech Rate:*

Speech rate, also known as speaking rate or speech tempo, refers to the speed at which spoken language is produced, typically measured in words per minute (WPM). Analyzing the speech rate of an audio file involves estimating the rate of speech utterances over time.

1) Speech Rate Calculations:

a) Speech Segmentation:

Divide the audio signal into speech segments, often using techniques like voice activity detection (VAD) to identify regions of speech.

b) Word Counting:

Count the number of words or syllables in each speech segment.

c) Time Calculations:

Calculate the duration of each speech segment.

d) Speech Rate Estimations:

Estimate the speech rate by dividing the total number of words or syllables by the total duration of speech.

The Speech rate is calculated using the speech recognition library to perform automatic speech recognition by the google API then split the sentence to words and to find the speech rate.

K. Formants:

Formants in the context of audio signals refer to the resonant frequencies of the vocal tract during speech production. They play a crucial role in shaping the timbre and intelligibility of speech sounds. Analyzing formants in an audio file involves identifying and characterizing these prominent spectral peaks.

1) Analyzing Formants:

a) Speech Segmentation:

Divide the audio signal into segments corresponding to individual speech sounds or phonemes.

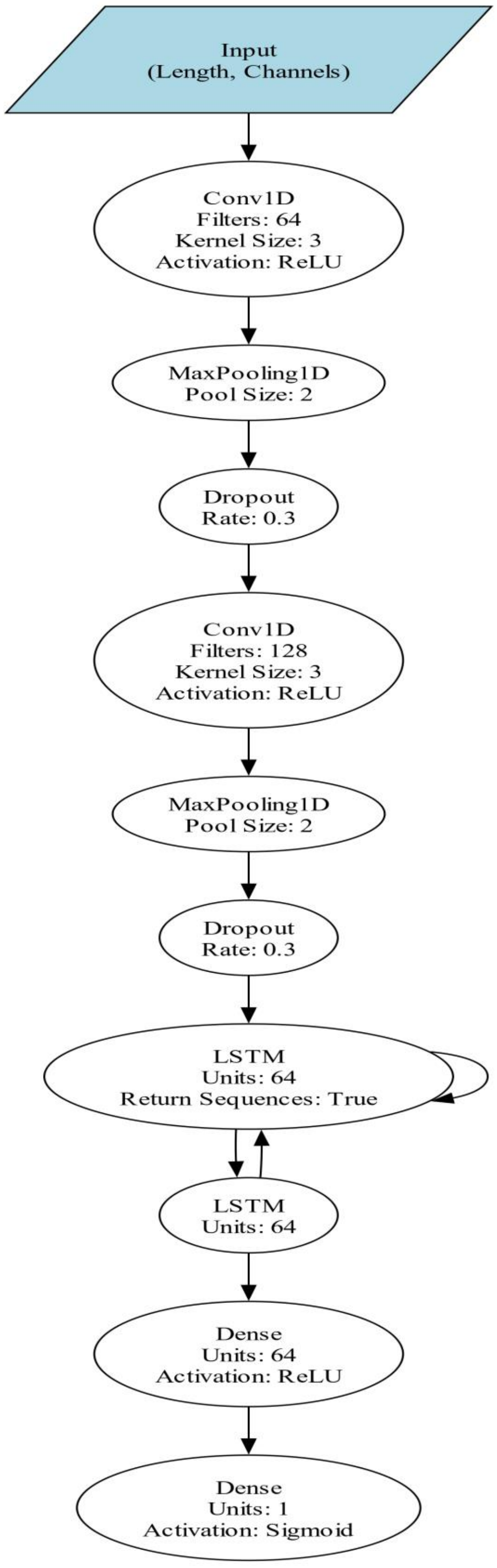
b) Spectral Analysis:

Compute the spectrum of each speech segment using techniques such as the Fourier Transform or Short-Time Fourier Transform (STFT).

c) Formants Detection:

Identify the peaks in the spectrum that correspond to the resonant frequencies of the vocal tract. These peaks are known as formants.

V. Block Diagram:



VI. Working Principle

The proposed system uses Convolutional-Recurrent Neural Networks(CRNN) which means Convolutional Neural Networks(CNN) followed by Recurrent Neural Networks(RNN) of Long-Short Term Memory(LSTM) to classify the voices into fake or real voice label. The model consists of 10 layers including 1 input layer, 2 convolutional layers, 2 MaxPooling layers, 2 Dropout layers, 2 LSTM layers, 2 Dense layers including output layers.

A. *Layers of the Network:*

1) 1st layer:

Convolutional layer – Layer that works on one-dimensional input data. It is tuned with 64 filters or kernels and has the convolutional window of size 3. The activation function used in the layer is ReLU activation function which introduces non-linearity into the model.

2) 2nd layer:

MaxPooling1D layer – Layer which has the pool size of 2 which performs maximum pooling operations on the temporal data. This maximum pooling enables the network to learn required features and prevents from overfitting.

3) 3rd layer:

Dropout layer - Dropout is a regularization technique where randomly selected neurons are ignored during training. It helps prevent overfitting by forcing the model to learn redundant representations. It is set to the value of 0.3 which fraction of inputs that has been dropped during training.

4) 4th layer:

Convolutional 1Dimension layer – Same as the first layer but here the filters are added up to 128 filters with same kernel size of 3 and same activation function as ReLU.

5) 5th layer:

Dropout layer - Dropout is a regularization technique where randomly selected neurons are ignored during training. It helps prevent overfitting by forcing the model to learn redundant representations. It is set to the value of 0.3 which fraction of inputs that has been dropped during training.

6) 6th layer:

Max Pooling layer – Same as the second layer with pool size of 2 to reduce overfitting of the model during previous convolutional layer.

7) 7th layer:

LSTM layer – Long Short Term Memory is a type of recurrent neural network (RNN) architecture capable of learning long-term dependencies. Here it consists of 64 neurons and return sequence is set to true which means it returns full sequence to the next because the next layer is also a LSTM layer.

8) 8th layer:

LSTM layer – Same as the previous layer with 64 neurons and return sequence is set to false which means it returns only the last output.

9) 9th layer:

Dense layer – Layer in which every neuron from previous is connected to next layer is known as dense layer. It has 64 neurons in the layer and activation function used in the layer is ReLU activation function.

10) 10th layer:

Dense layer – The final dense layer with single neuron which is connected with the all the other neurons from previous layer. The activation function of the neuron is sigmoid function which splits the input and output between 1 and 0 for this binary classification problem.

B. Activation functions:

1) ReLU Activation Functions:

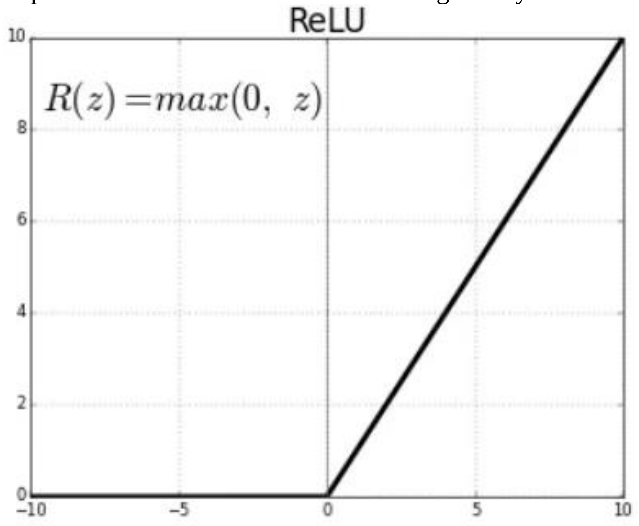
It is a piecewise linear function that returns 0 for negative input and the same input value when it is greater than 0. It avoids the vanishing gradient problem and allows the network to learn quickly. Activation function of ReLU is:

$$f(x) = \max(0, x)$$

The derivative of function is given by:

$$f(x) = \begin{cases} 0, & x < 0 \\ 1, & x \geqslant 0. \end{cases}$$

Graph of the ReLU activation function is given by:



2) Sigmoid Activation Function:

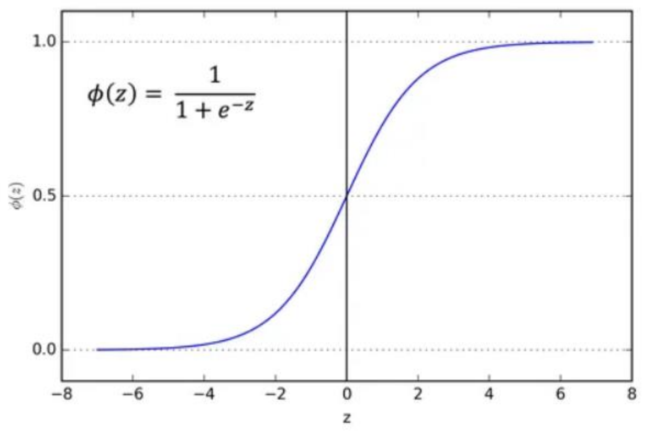
As the name suggests it is an S-shaped curve which maps the input value to any value between 0 and 1. It is mostly used in the output layer for the binary classification where probability of class considered. Activation function is given by:

$$f(x) = \frac{1}{1 + e^{-x}}$$

The derivative of the function is given by:

$$f'(x) = f(x).(1 - f(x)).$$

Graph of the sigmoid activation function is given by,



C. Epochs:

Epoch represent how many times the model sees the dataset. When we set epoch to 10 it sees the datasets 10 times and when we set it to 20 model sees the datasets 20 times.

1) Why 20 epochs:

The choice of number epochs depends on various factors, including the complexity of the problem, the size of the dataset, the architecture of the model. For voice detection we initialize from 0 to 20 where on 20 epochs it reached its saturation.

2) Monitoring Performance:

During training, it's necessary to observe the model's performance on a validation dataset to avoid overfitting. You can do this by calculating metrics such as accuracy, precision, recall, and F1-score on the validation set after each epoch. When the performance on validation set is starts decreasing it is an alarming sign that model overfits on the datasets. Hence, further increase in the epochs can only cause decrease in the performance.

3) Early stopping:

To automatically stop training when the performance starts to degrade, you can use a technique called early stopping. It monitors the performance on validation set after each epoch when the performance decreases it automatically stops which is the early stopping method.

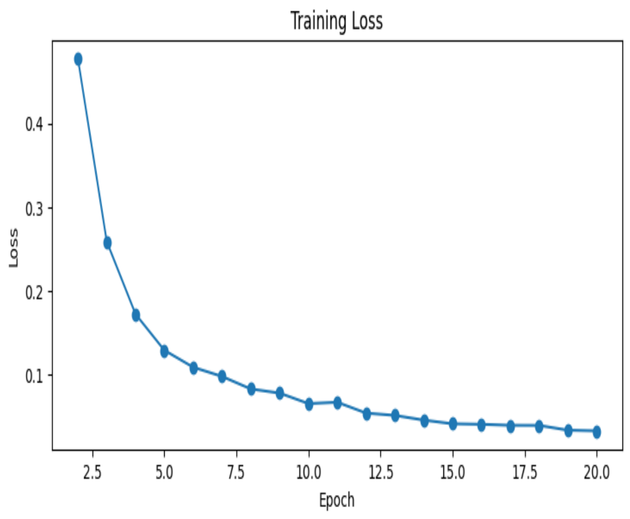
VII. Graphs

A. Learning graphs of the model:

The model's learning is recorded on each and every epoch which details used to plot the graphs so that we can learn all things about the model:

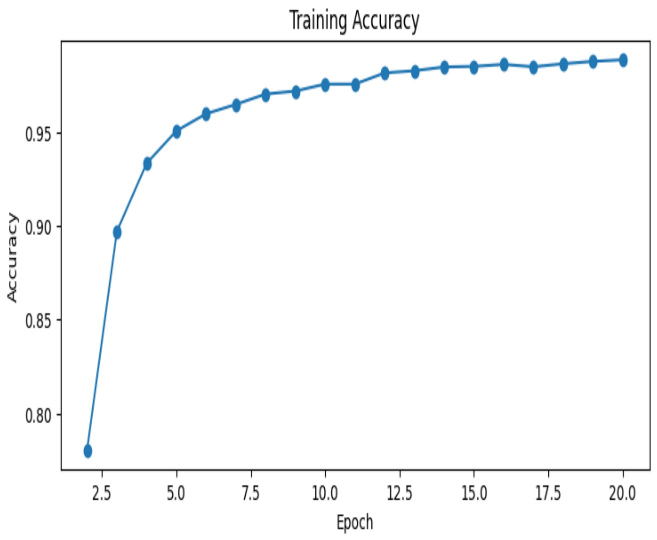
1) Training loss vs Epoch graph:

The graph showed a great descent as the model moves on each of the epoch. It is great sign that the graph reducing on each epoch which indicates the model loss very few datapoints during its training.



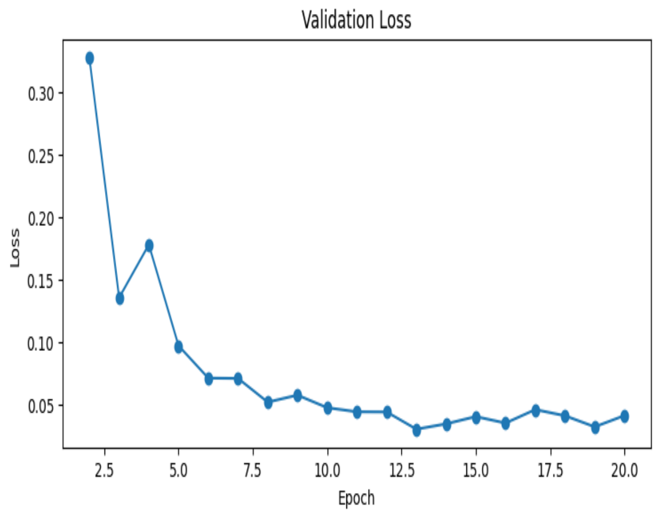
2) Training accuracy vs Epoch:

The graph started at very lesser accuracy of 78% but on next epochs it raises to some how higher extend and then reaches its saturation point on 99.81%. Above the saturation point it cannot reach more accuracy so that the model is stopped at 20 epochs.



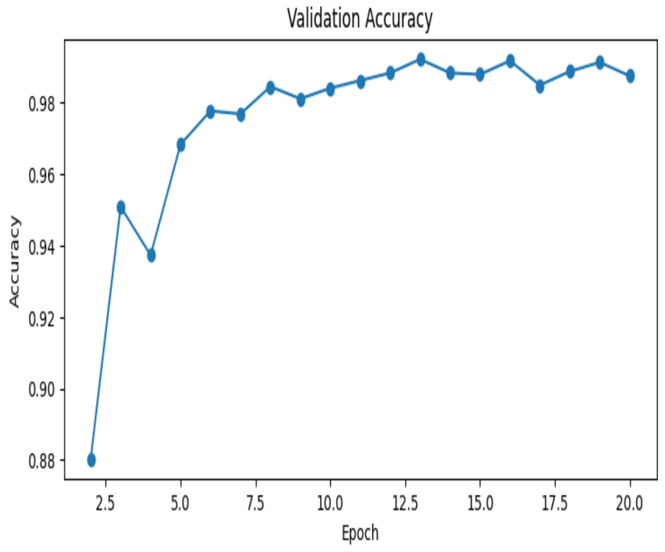
3) Validation loss vs Epoch:

The graph started at huge loss and decreased as the network learns from the dataset. It had a loss of 35% at the start of the model's learning but it decreases as model learn and decreases up to 0.02%.



4) Validation accuracy vs Epoch:

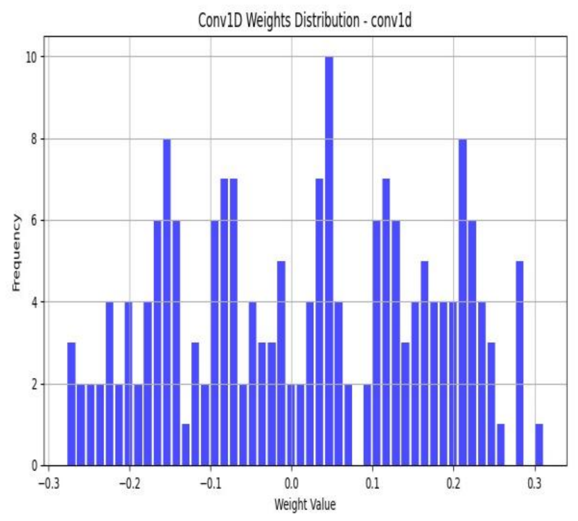
The graph started at the lesser accuracy of 88% on unknown data it increases 95% on epoch 4 and again decreases to 93% on epoch 5. From the epoch 6 it increases and reaches its saturation of 98.56%.



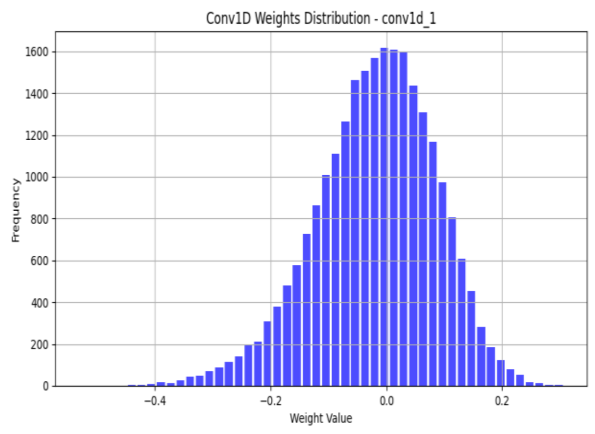
5) Weights of the Layers:

a) Weights of the 1st Convolution layer:

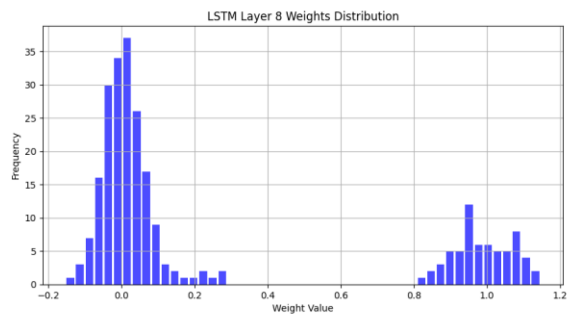
The frequency of weights of the neurons present in the convolutional layer is plotted against its weight value in a histogram plot.

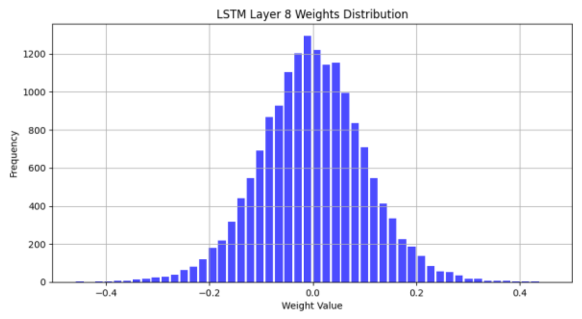


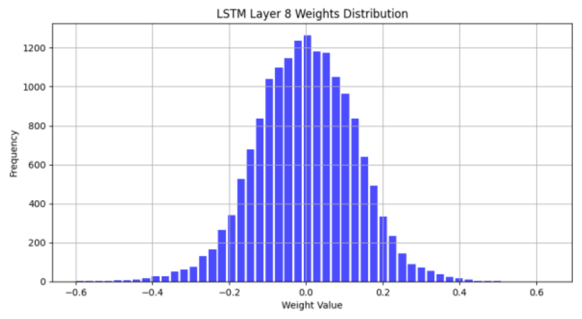
b) Weights of the 2nd Convolution layer:



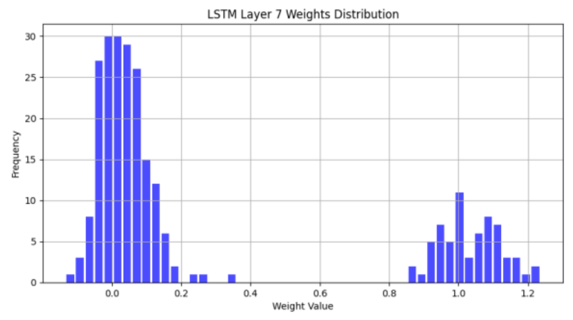
c) Weights of the 2nd LSTM layer:

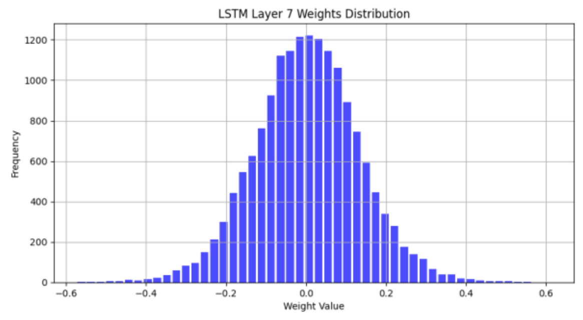


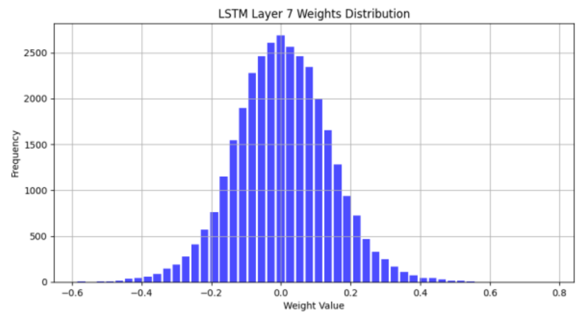




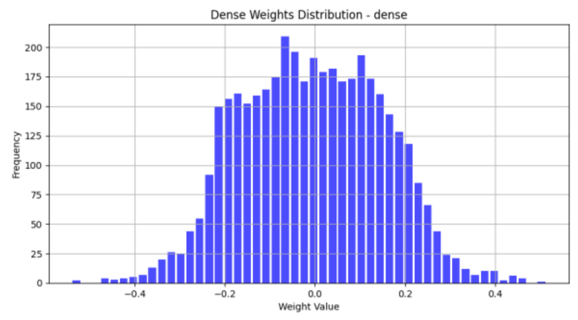
d) Weights of the 1st LSTM layer



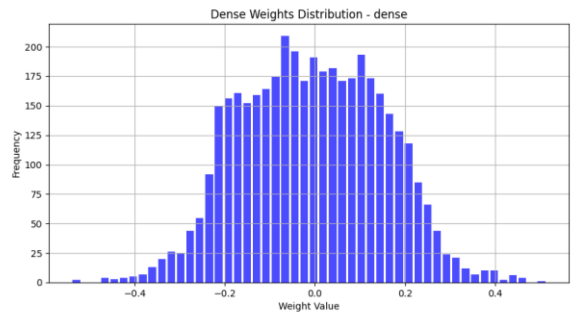




e) Weights of the 1st Dense layer:



f) Weights of the 2nd Dense layer:



B. Gradient Values used to Train the Perceptron:

The gradient descent technique is one of the main technique through which the network learns to minimize the root mean squared error value of the output. Hence the gradient value of each layers value is recorded to track how quick the network learns from the dataset:

i. Layer conv1d: Gradient Norm = 3.1949734687805176

ii. Layer max_pooling1d: Gradient_Norm = 0.13237066566944122

iii. Layer dropout: Gradient Norm = 4.221711158752441

iv. Layer conv1d_1: Gradient Norm = 0.1220540702342987

v. Layer max_pooling1d_1: Gradient Norm = 5.751877784729004

vi. Layer dropout_1: Gradient Norm = 0.5384106040000916

vii. Layer lstm: Gradient Norm = 0.12745781242847443

viii. Layer lstm_1: Gradient Norm = 1.4033795595169067

ix. Layer dense: Gradient Norm = 1.10952889919281

x. Layer dense_1: Gradient Norm = 0.33946794271469116

VIII. Output of the Model

The network has produced an output accuracy of 99.02 and test accuracy of 99.023 loss of 0.0285 and test loss of 0.02851 which great performance as it surpasses all the traditional machine learning and neural network models trained before. The output on each epoch is given below:

Epoch 1/20
295 /295 - 4s 7ms/step - loss: 0.4771 - accuracy: 0.7806 - validation loss: 0.3271 - validation accuracy: 0.8803

Epoch 2/20
295 /295 - 1s 5ms/step - loss: 0.2588 - accuracy: 0.8969 - validation loss: 0.1358 - validation accuracy: 0.9508

```
Epoch 3/20
295 /295 - 1s 5ms/step - loss: 0.1721 - accuracy: 0.9332 -
validation loss: 0.1781 - validation accuracy: 0.9372

Epoch 4/20
295 /295    - 1s 5ms/step - loss: 0.1289 - accuracy: 0.9505 -
validation loss: 0.0974 - validation accuracy: 0.9682

Epoch 5/20
295 /295    - 1s 5ms/step - loss: 0.1087 - accuracy: 0.9597 -
validation loss: 0.0717 - validation accuracy: 0.9775

Epoch 6/20
295 /295    - 1s 5ms/step - loss: 0.0977 - accuracy: 0.9647 -
validation loss: 0.0714 - validation accuracy: 0.9767

Epoch 7/20
295 /295    - 1s 5ms/step - loss: 0.0827 - accuracy: 0.9702 -
validation loss: 0.0525 - validation accuracy: 0.9843

Epoch 8/20
295 /295    - 1s 5ms/step - loss: 0.0778 - accuracy: 0.9718 -
validation loss: 0.0581 - validation accuracy: 0.9809

Epoch 9/20
295 /295    - 1s 5ms/step - loss: 0.0653 - accuracy: 0.9755 -
validation loss: 0.0480 - validation accuracy: 0.9839

Epoch 10/20
295 /295    - 1s 5ms/step - loss: 0.0669 - accuracy: 0.9755 -
validation loss: 0.0448 - validation accuracy: 0.9860

Epoch 11/20
295 /295    - 1s 5ms/step - loss: 0.0540 - accuracy: 0.9814 -
validation loss: 0.0446 - validation accuracy: 0.9881

Epoch 12/20
295 /295    - 1s 5ms/step - loss: 0.0512 - accuracy: 0.9826 -
validation loss: 0.0309 - validation accuracy: 0.9919

Epoch 13/20
295 /295    - 1s 5ms/step - loss: 0.0456 - accuracy: 0.9846 -
validation loss: 0.0351 - validation accuracy: 0.9881

Epoch 14/20
295 /295    - 1s 5ms/step - loss: 0.0412 - accuracy: 0.9849 -
validation loss: 0.0410 - validation accuracy: 0.9877

Epoch 15/20
295 /295    - 1s 5ms/step - loss: 0.0406 - accuracy: 0.9860 -
validation loss: 0.0357 - validation accuracy: 0.9915

Epoch 16/20
295 /295    - 1s 5ms/step - loss: 0.0394 - accuracy: 0.9847 -
validation loss: 0.0465 - validation accuracy: 0.9847

Epoch 17/20
295 /295    - 1s 5ms/step - loss: 0.0393 - accuracy: 0.9863 -
validation loss: 0.0416 - validation accuracy: 0.9885

Epoch 18/20
295 /295    - 1s 5ms/step - loss: 0.0336 - accuracy: 0.9876 -
validation loss: 0.0328 - validation accuracy: 0.9911

Epoch 19/20
295 /295    - 1s 5ms/step - loss: 0.0328 - accuracy: 0.9884 -
validation loss: 0.0418 - validation accuracy: 0.9873

Epoch 20/20
295 /295    - 2s 5ms/step - loss: 0.0321 - accuracy: 0.9882 -
validation loss: 0.0312 - validation accuracy: 0.9911
74/74 - 0s 1ms/step - loss: 0.0312 - accuracy: 0.9911

Test Loss: 0.031169405207037926, Test Accuracy:
0.9910866022109985.
```

REFERENCES

[1] Transfer Learning from Speaker Verification to Multispeaker Text-To-Speech Synthesis by Ye Jia, Yu Zhang, Ron J. Weiss, Quan Wang, Jonathan Shen, Fei Ren, Zhifeng Chen, Patrick Nguyen, Ruoming Pang, Ignacio Lopez Moreno, Yonghui Wu presented on NeurIPS 2018

[2] Neural Voice Cloning with a Few Samples" by Sercan O. Arik, Jitong Chen, Kainan Peng, Wei Ping, Yanqi Zhou, Yu Zhang, Timothy K. Marks presented on NeurIPS 2018

[3] VoiceLoop: Voice Fitting and Synthesis via a Phonological Loop by Sercan O. Arik, Jitong Chen, Kainan Peng, Wei Ping, Yanqi Zhou, Chun-Cheng Chou, Ruoming Pang, Quan Wang, Shiyu Chang, Shujie Liu, Yang Zhang, Patrick Nguyen, Timothy K. Marks presented on ICML2019.

[4] Neural Source-Filter-based Expressive Speech Synthesis by Heewon Jeon, Chiyu Zhang, Yonghui Wu, Ron J. Weiss on ICASSP 2020.

[5] McFee, Brian, et al. "Librosa: Audio and Music Signal Analysis in Python." Proceedings of the 14th Python in Science Conference. 2015.

[6] Abadi, Martín, et al. "TensorFlow: A System for Large-Scale Machine Learning." 12th USENIX Symposium on Operating Systems Design and Implementation(OSDI16),2016.