**7COM1039-0109-2022**

Advanced Computer Science Masters Project

**Interim Progress Report (IPR)**

Human Emotion Detection from the audio using Deep Learning

**Student Id :** 20026202

**Student Name :** Sairam Kandukuri

**Supervisor :** [Chidinma Chiejina](https://herts.instructure.com/courses/100816/users/171944)

**Contents**

1. Background research 2
2. Project plan 5
3. Summary of progress to date 9
4. Consideration of ethical, legal, professional and social issues 10

**Appendices** 11

**References** 25

1. **Background Research**

This project aims to develop a deep learning model which can detect the human emotion from the audio file provided as the input. This project also aimed to solve a research question “Will the high emotion level audio provide more accurate emotion detection (Anger, Disgust, Fear, Happy, Neutral, and Sad) compared to low and medium emotional level audio?”.

For this project CREMA-D dataset [1] is considered from the Kaggle and considered the original dataset which is available in GitHub [2] with some extra important data files (VideoDemographics.csv, finishedResponses.csv) which linked with the original dataset and will are crucial for data analysis.

The Research started with finding human emotion projects and papers across the internet, mainly on IEEE Xplore and other platforms like ResearchGate. The first task in this project is to identify the relevant papers which are similar to this idea to seek some help to start the project, in this process [3,4,5,6,7] papers have been found which are relatively close to my research idea and also the dataset.

The dataset considered from the Kaggle consists of only audio files with some specific notation like " 1001\_DFA\_DIS\_XX.wav”. 1001 represents the actor’s id, DFA represents dialogue the actor has told, DIS represents emotion, and XX represents tone level in the audio. According to [5], the dataset consists of 7442 clips of 91 actors with diverse backgrounds like Caucasian, African American, Asian, few unknown and each actor has chosen to tell from 12 sentences with 6 emotions (anger, disgust, fear, happy, neutral, sad) in 3 tone levels (Low, Medium, High).

This project is considered to have two output variables “Emotion” and “Tone Level”. The emotion column has a nearly equal distribution of data for 6 emotions, whereas the “Tone Level” column has some missed values, which need to fill with the relative data existing in some other dataset by merging based on clip name. In the process of figuring a solution for this, it has been observed that there is a file named "**finishedResponses.csv**" related to the crema-d dataset, in which they have given a hint for assigning the tone levels i.e., "dispVal" - the displayed value "dispLevel" - a numeric representation of the displayed value, 20 for low, 50 for med, 80 for hi. Those unspecified XX values in the Tone\_Level column will be replaced by their actual tone level referring to these two columns and merging them. To get more information about the data, "VideoDemographics.csv” has been merged with the data considered in the beginning.

After merging the required data, the next challenge is to clean the data by removing the null values in the data and then subset the data with the required columns for the analysis. The later step is data augmentation for the audio data because the original data has only 7442 records, for the better training of the model it is necessary to have more records. Hence it is necessary to learn about the augmentation techniques and how they can be done. Augmentation techniques like Pitching, shifting, stretching, and adding noise are used according to the brief description given in [8] about these techniques.

Now, the major part is to extract the features of the data after augmentation because it is very important to get the main features from the audio using feature extraction which will help in the prediction of emotions very well [7]. The feature extraction techniques like Zero-Crossing Rate, MFCC, Chroma STFT, RMS, and Mel Scale Spectrogram have been studied and applied in the project based on the above methods discussed in the paper [3, 4, 7] which helped a lot for the project. Then the next step in the project is to perform one hot encoding for the output variables and then standardize the data using a standard scaler.

Firstly, a baseline model has to be built, just to make sure everything is in place. A CNN model with one hidden layer with one input and two output variables has been developed to test at the first place. The paper [7], “deep learning-based audio processing speech emotion detection” has been used to understand the structure for building CNN model.

The above is the complete background research I have done to understand the needs for doing emotion detection model using deep learning and achieved the results for the base line model which are shown in next section.

**2)** **Project Plan**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Human Emotion Detection from the audio using Deep Learning** | | | | | |
|  |
|  | **University of Hertfordshire, 20026202, SAIRAM KANDUKURI** | | | | |  |
|  |  |  |  |  |  |  |
|  | **Milestone description** | **Progress** | **Start** | **End** | **Days** |  |
| **Project Idea and Data** | Identifying Project Idea | **100%** | 25/01/2023 | 26/01/2023 | 1 |  |
| Identifying Data Source and connections | **100%** | 27/01/2023 | 29/01/2023 | 2 |  |
| Understanding Data | **100%** | 01/02/2023 | 04/02/2023 | 3 |  |
| Identify Hypothesis | **100%** | 06/02/2023 | 07/02/2023 | 2 |  |
| Detailed Project Prososal | **100%** | 06/02/2023 | 07/02/2023 | 2 |  |
| DPP Submission | **100%** | 06/02/2023 | 07/02/2023 | 2 |  |
| **Literature Research (Identify Papers on)** | Crema-D Dataset and Title | **90%** | 08/02/2023 | 31/05/2023 | 112 |  |
| Deep learning papers on emotion detection | **85%** | 09/02/2023 | 31/05/2023 | 111 |  |
| Data Augmentation for audio | **80%** | 10/02/2023 | 31/05/2023 | 110 |  |
| Features Extraction from audio | **100%** | 12/02/2023 | 31/05/2023 | 108 |  |
| **Data Cleaning and Merging** | Data Loading and Seperation of clipname into columns | **100%** | 15/02/2023 | 15/02/2023 | 1 |  |
| Merge VideoDemographics.csv on ActorID and get insights. | **100%** | 16/02/2023 | 17/02/2023 | 2 |  |
| Merge finishedResponses.csv with preciously merged data on ActorID | **100%** | 18/02/2023 | 19/02/2023 | 2 |  |
| Deal with missed values in Tone Column | **100%** | 21/02/2023 | 22/02/2023 | 2 |  |
| Drop any null value rows | **100%** | 23/02/2023 | 23/02/2023 | 1 |  |
| Subset the required columns from cleaned data for further analysis | **100%** | 23/02/2023 | 23/02/2023 | 1 |  |
| **Data Augmentation and Feature Extarction** | Understanding Librosa Library | **100%** | 01/03/2023 | 03/03/2023 | 3 |  |
| Learn Plotting the spectral waveforms | **100%** | 04/03/2023 | 04/03/2023 | 1 |  |
| Learn audio augmentation methods | **100%** | 07/03/2023 | 07/03/2023 | 1 |  |
| Implement audio augmentation methods | **100%** | 08/03/2023 | 08/03/2023 | 1 |  |
| Test noise, stretch, shift, pitch for audio files. | **100%** | 09/03/2023 | 09/03/2023 | 1 |  |
| Learn ZCR, Chroma\_stft, MFCC, RMS, MEL | **100%** | 10/03/2023 | 12/03/2023 | 3 |  |
| Check code references for feature extraction. | **100%** | 13/03/2023 | 14/03/2023 | 2 |  |
| Implement feature extraction for all audio files | **100%** | 15/03/2023 | 17/03/2023 | 3 |  |
| One hot encode ToneLevel and Emotion | 100% | 18/03/2023 | 18/03/2023 | 1 |  |
| Data Frame all extracted features | **100%** | 18/03/2023 | 18/03/2023 | 1 |  |
| **Model Building** | Base Line Model | **100%** | 22/03/2023 | 05/04/2023 | 15 |  |
| Analyse results | **100%** | 06/04/2023 | 08/04/2023 | 3 |  |
| Interim Progress Report | **100%** | 10/04/2023 | 21/04/2023 | 12 |  |
| IPR Submission | **100%** | 01/05/2023 | 01/05/2023 | 1 |  |
| Algorithm Experiments and hyperparameter tuning |  | 13/05/2023 | 20/05/2023 | 8 |  |
| Performance Comparision |  | 21/05/2023 | 21/05/2023 | 1 |  |
| Test unseen data on best model |  | 22/05/2023 | 22/05/2023 | 1 |  |
| **Dissertation Report** | Abstract |  | 23/05/2023 | 24/05/2023 | 2 |  |
| Introduction |  | 25/05/2023 | 27/05/2023 | 3 |  |
| Literature Review |  | 28/05/2023 | 10/06/2023 | 14 |  |
| Meet with Supervisor |  | 07/06/2023 | 07/06/2023 | 1 |  |
| Data Analysis |  | 12/06/2023 | 17/06/2023 | 6 |  |
| Model Building |  | 20/06/2023 | 26/06/2023 | 7 |  |
| Results |  | 27/06/2023 | 30/06/2023 | 4 |  |
| Conclusion |  | 01/07/2023 | 01/07/2023 | 1 |  |
| Future Work |  | 02/07/2023 | 02/07/2023 | 1 |  |
| References |  | 03/07/2023 | 03/07/2023 | 1 |  |
| Appendices |  | 03/07/2023 | 03/07/2023 | 1 |  |
| Plagarism Check |  | 04/07/2023 | 04/07/2023 | 1 |  |
| Final Edits and Review |  | 05/07/2023 | 10/07/2023 | 6 |  |
| Final Plagarism Check |  | 11/07/2023 | 11/07/2023 | 1 |  |
| Meet with Supervisor |  | 12/07/2023 | 12/07/2023 | 1 |  |
| FPR Submission |  | 28/08/2023 | 28/08/2023 | 1 |  |

**Gantt Chart for Project Plan**

**Chart

Description automatically generated**

**3) Summary of Progress to Date**

As mentioned in the above project plan, Data understanding and merging the required columns for analysis has taken a lot of time, but successfully completed the part of data cleaning by finding and dropping the unwanted null values from the data. Later, Understanding the techniques of audio augmentation and feature extraction has been completed along with the practical implementation of code and tested to make sure practical data augmentation techniques and features extraction techniques are working.

Finally, the baseline deep learning model has been implemented with the help of the architecture mentioned in the journal [7] by feeding the extracted features of the audio to the model. The performance has been tracked and achieved 95% accuracy on the training data for output variable 1 and 73% accuracy for output variable 2. On testing data, output variable 1 achieved 90% accuracy and 53.49% for output variable 2. The graph for the results achieved are shown below.

Chart, line chart, scatter chart

Description automatically generated

Chart, line chart, scatter chart

Description automatically generated

This project has been successfully completed with the baseline model to date. Still, a lot of work needs to be done to get better results with advanced deep-learning architectures. And later I have to start writing a dissertation report for this project.

The source code for the project implemented to date is shown in the appendices.

**4) Consideration of ethical, legal, professional, and social issues**

The consideration for this project is the secondary data which is already available on Kaggle and GitHub. Source code is referred through online sources and my own version of the code is implemented with the help of architecture in the journal [7], the implementation and libraries used are completely legal and there are no social issues associated with concept or project implementation.

**Appendices**

Source Code for the Human Emotion Detection using deep learning project is given below:

import matplotlib.pyplot as plt

import numpy as np

import pandas as pd

import os

import sys

import librosa

import librosa.display

import seaborn as sns

from sklearn.preprocessing import StandardScaler, OneHotEncoder

from sklearn.metrics import confusion\_matrix, classification\_report

from sklearn.model\_selection import train\_test\_split

from IPython.display import Audio

import tensorflow as tf

from tensorflow.keras.callbacks import ReduceLROnPlateau

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense, LSTM, Flatten, Dropout, BatchNormalization

from tensorflow.keras.callbacks import ModelCheckpoint

import librosa

import pandas as pd

import numpy as np

import tensorflow as tf

from tensorflow import keras

from tensorflow.keras import layers

import warnings

if not sys.warnoptions:

warnings.simplefilter("ignore")

warnings.filterwarnings("ignore", category=DeprecationWarning)

import tensorflow as tf

from tensorflow import keras

from tensorflow.python.keras.callbacks import ReduceLROnPlateau

from tensorflow.python.keras.models import Sequential

from tensorflow.python.keras.layers import Dense, Conv1D, MaxPooling1D, Flatten, Dropout, Activation

from tensorflow.python.keras.utils import np\_utils

from tensorflow.python.keras.callbacks import ModelCheckpoint

from tensorflow.keras.layers import BatchNormalization

from tensorflow.keras.utils import to\_categorical

Crema = os.listdir("AudioWAV")

Crema.sort()

Crema[0:20]

**1) Seperate the file name to get, dialouge, emotion and tone level, Concatenate the path to respective audiofiles in a folder AudioWAV. Add them as seperate columns.**

c = pd.DataFrame(Crema, columns=["FileName"])

c

*# To store the values in seperate columns*

clipName = []

actor\_id = []

dialouge = []

emotion = []

tone = []

path = []

*#Seperating the FileName based on "\_" seperator to get the dialouge, emotion and tone level in seperate columns.*

*# also adding the path for the audio file in the folder AudioWAV*

for i in c["FileName"]:

c\_split = i.split(".")

clipName.append(c\_split[0])

c\_split\_1 = c\_split[0].split("\_")

actor\_id.append(c\_split\_1[0])

dialouge.append(c\_split\_1[1])

emotion.append(c\_split\_1[2])

tone.append(c\_split\_1[3])

path.append("AudioWav/" + i)

# Adding to the respective empty columns by converting to the list.

c["clipName"] = list(clipName)

c["ActorID"] = list(actor\_id)

c["Dialouge"] = list(dialouge)

c["Emotion"] = list(emotion)

c["Tone\_Level"] = list(tone)

c["File\_Path"] = list(path)

# Converting the ActorID to integer type as it is in object type, for future use of merging.

c["ActorID"] = c["ActorID"].astype(np.int64)

c.head()

c.info()

c['ActorID'].unique()

### 2) Merging the VideoDemographics.csv file based on the ActorID to get the gender of the voice in the audio.

### gend = pd.read\_csv("VideoDemographics.csv")

### gend.head()

*# Merging c (actual data), gend (videodemographics.csv) files to get the gender based on the ActorID as unique column.*

c\_gend=pd.merge(c,gend, left\_on='ActorID', right\_on='ActorID', how='left')

c\_gend

c\_gend.info()

*# checking the duplicates in the filepath.*

(c\_gend.File\_Path.value\_counts()==1).value\_counts()

c\_gend["Race"].value\_counts()

c\_gend["Ethnicity"].value\_counts()

*# SInce this the important column for the analysis of the hypothesis question, analysing this column is important.*

c\_gend["Tone\_Level"].value\_counts()

c\_gend["Emotion"].value\_counts()

tone\_fill = pd.read\_csv("finishedResponses.csv")

tone\_fill.head()

*#We just need, the dispVal and clipName for filling the missing data.*

tone\_fill\_disp\_value = tone\_fill[["clipName","dispVal"]].sort\_values(["clipName","dispVal"])

tone\_fill\_disp\_value

*# there are 219688 rows in the above data, but we just need the unique values. hence removing the duplicates.*

clips\_unique\_tone\_value=tone\_fill\_disp\_value.drop\_duplicates()

clips\_unique\_tone\_value

# we are not considering any null values, after removing the null values, we have got 7442 which is the exact rows.

# which we have in c\_gend dataframe.

clips\_unique\_tone\_value = clips\_unique\_tone\_value[~clips\_unique\_tone\_value.dispVal.isnull()==True]

clips\_unique\_tone\_value

clips\_unique\_tone\_value["dispVal"].value\_counts()

*#Merging clips\_unique\_tone\_value and c\_gend on clipname using left join to get the dispVal values for respective audio clip.*

HED = pd.merge(c\_gend, clips\_unique\_tone\_value, on='clipName', how='left')

HED.info()

#checking if there are any null values in dispVal column.

HED[HED.dispVal.isnull()==True]

# There is one Null Value in dispVal, so we are removing that.

HED = HED.dropna().reset\_index(drop = True)

HED

# Now all the values are non-null for all 7441 records.

# Its time to add the Respective tone level based on the dispVal of the audio.

for i in range(len(HED["Tone\_Level"])):

if i < len(HED["Tone\_Level"]) and HED["Tone\_Level"][i] in ("XX","X"):

if i < len(HED["dispVal"]):

if HED["dispVal"][i] == 20.0:

HED["Tone\_Level"][i] = "LO"

elif HED["dispVal"][i] == 50.0:

HED["Tone\_Level"][i] = "MD"

elif HED["dispVal"][i] == 80.0:

HED["Tone\_Level"][i] = "HI"

HED["Tone\_Level"].value\_counts()

HED.head()

# Now we just need audiofile path,Dialouge in the audiofile, tone\_level of the audio, emotion in the

# audio. so we are having only these four columns for model building.

HED\_DATA = HED[["File\_Path","Tone\_Level","Emotion"]]

for i in range(len(HED\_DATA["Emotion"])):

if i < len(HED\_DATA["Emotion"]):

if HED\_DATA["Emotion"][i] == "ANG":

HED\_DATA["Emotion"][i] = "ANGER"

elif HED\_DATA["Emotion"][i] == "DIS":

HED\_DATA["Emotion"][i] = "DISGUST"

elif HED\_DATA["Emotion"][i] == "FEA":

HED\_DATA["Emotion"][i] = "FEAR"

elif HED\_DATA["Emotion"][i] == "HAP":

HED\_DATA["Emotion"][i] = "HAPPY"

elif HED\_DATA["Emotion"][i] == "NEU":

HED\_DATA["Emotion"][i] = "NEUTRAL"

elif HED\_DATA["Emotion"][i] == "SAD":

HED\_DATA["Emotion"][i] = "SAD"

HED\_DATA.head()

HED\_DATA["Emotion"].value\_counts()

for i in range(len(HED\_DATA["Tone\_Level"])):

if i < len(HED\_DATA["Tone\_Level"]):

if HED\_DATA["Tone\_Level"][i] == "MD":

HED\_DATA["Tone\_Level"][i] = "MEDIUM"

elif HED\_DATA["Tone\_Level"][i] == "LO":

HED\_DATA["Tone\_Level"][i] = "LOW"

elif HED\_DATA["Tone\_Level"][i] == "HI":

HED\_DATA["Tone\_Level"][i] = "HIGH"

HED\_DATA["Tone\_Level"].unique()

HED\_DATA.head()

from imblearn.over\_sampling import RandomOverSampler

from collections import Counter

import pandas as pd

X = HED\_DATA["File\_Path"]

y1 = HED\_DATA['Tone\_Level']

y2 = HED\_DATA["Emotion"]

print('Original dataset shape %s' % Counter(y2))

### 3) converting audiowaves into spectral values.[¶](http://localhost:8888/notebooks/Dropbox/MSc%20Projects/Sairam/MSc_Project_Sairam/Human%20Emotion%20Detection%20from%20the%20audio%20using%20Deep%20Learning.ipynb#3)-converting-audiowaves-into-spectral-values.)

Now, its time to convert the audio file into values using data augmentation techniques at first and extracting features

def create\_waveplot(data, sr, e):

plt.figure(figsize=(10, 3))

plt.title('Waveplot for audio with {} emotion'.format(e), size=15)

librosa.display.waveshow(data, sr=sr)

plt.show()

def create\_spectrogram(data, sr, e):

X = librosa.stft(data)

Xdb = librosa.amplitude\_to\_db(abs(X))

plt.figure(figsize=(12, 3))

plt.title('Spectrogram for audio with {} emotion'.format(e), size=15)

librosa.display.specshow(Xdb, sr=sr, x\_axis='time', y\_axis='hz')

plt.colorbar()

emotion='FEAR'

path = np.array(HED\_DATA.File\_Path[HED\_DATA.Emotion==emotion])[1]

data, sampling\_rate = librosa.load(path)

create\_waveplot(data, sampling\_rate, emotion)

create\_spectrogram(data, sampling\_rate, emotion)

Audio(path)

#Data Augmentation

def add\_noise(aug\_data):

noise\_add = 0.035\*np.random.uniform()\*np.amax(aug\_data)

aug\_data = aug\_data + noise\_add\*np.random.normal(size=aug\_data.shape[0])

return aug\_data

def wav\_stretch(aug\_data):

"""

Streching the sound. Note that this expands the dataset slightly

"""

aug\_data = librosa.effects.time\_stretch(aug\_data, rate = 0.5)

return aug\_data

def wav\_shift(aug\_data):

wav\_shift\_range = int(np.random.uniform(low=-10, high = 10)\*1000)

return np.roll(aug\_data, wav\_shift\_range)

def add\_pitch(aug\_data, sampling\_rate, steps):

return librosa.effects.pitch\_shift(aug\_data, sr=sampling\_rate, n\_steps = steps)

file\_path = np.array(HED\_DATA.File\_Path)[2]

data, sample\_rate = librosa.load(file\_path)

data.shape

plt.figure(figsize=(14,4))

librosa.display.waveshow(y=data, sr=sample\_rate)

Audio(file\_path)

x = add\_noise(data)

plt.figure(figsize = (14, 4))

# orginial plot

librosa.display.waveshow(data, sr = sampling\_rate)

plt.title('Original Audio Waveform', size = 24)

plt.show()

plt.figure(figsize = (14, 4))

librosa.display.waveshow(x, sr = sampling\_rate)

plt.title('Augmented Audio Waveform with Noise', size = 24)

plt.show()

Audio(x, rate = sampling\_rate)

x = wav\_stretch(data)

plt.figure(figsize = (14, 4))

# orginial plot

librosa.display.waveshow(data, sr = sampling\_rate)

plt.title('Original Audio Waveform', size = 24)

plt.show()

plt.figure(figsize = (14, 4))

librosa.display.waveshow(x, sr = sampling\_rate)

plt.title('Augmented Audio Waveform with Stretch', size = 24)

plt.show()

Audio(x, rate = sampling\_rate)

x = wav\_shift(data)

plt.figure(figsize = (14, 4))

# orginial plot

librosa.display.waveshow(data, sr = sampling\_rate)

plt.title('Original Audio Waveform', size = 24)

plt.show()

plt.figure(figsize = (14, 4))

librosa.display.waveshow(x, sr = sampling\_rate)

plt.title('Augmented Audio Waveform with Shift', size = 24)

plt.show()

Audio(x, rate = sampling\_rate)

x = add\_pitch(data,sample\_rate, steps=5)

plt.figure(figsize = (14, 4))

# orginial plot

librosa.display.waveshow(data, sr = sampling\_rate)

plt.title('Original Audio Waveform', size = 24)

plt.show()

plt.figure(figsize = (14, 4))

librosa.display.waveshow(x, sr = sampling\_rate)

plt.title('Augmented Audio Waveform with Pitch Shift', size = 24)

plt.show()

Audio(x, rate = sampling\_rate)

# Features extraction from data

def extract\_features(data):

# ZCR

result = np.array([])

zcr = np.mean(librosa.feature.zero\_crossing\_rate(y=data).T, axis=0)

result=np.hstack((result, zcr)) # stacking horizontally

# Chroma\_stft

stft = np.abs(librosa.stft(data))

chroma\_stft = np.mean(librosa.feature.chroma\_stft(S=stft, sr=sample\_rate).T, axis=0)

result = np.hstack((result, chroma\_stft)) # stacking horizontally

# MFCC

mfcc = np.mean(librosa.feature.mfcc(y=data, sr=sample\_rate).T, axis=0)

result = np.hstack((result, mfcc)) # stacking horizontally

# Root Mean Square Value

rms = np.mean(librosa.feature.rms(y=data).T, axis=0)

result = np.hstack((result, rms)) # stacking horizontally

# MelSpectogram

mel = np.mean(librosa.feature.melspectrogram(y=data, sr=sample\_rate).T, axis=0)

result = np.hstack((result, mel)) # stacking horizontally

return result

def get\_features(path):

# duration and offset are used to take care of the no audio in start and the ending of each audio files as seen above.

data, sample\_rate = librosa.load(path, duration=2, offset=0.6, sr=8025)

# without augmentation

res1 = extract\_features(data)

result = np.array(res1)

# data with noise

noise\_data = add\_noise(data)

res2 = extract\_features(noise\_data)

result = np.vstack((result, res2)) # stacking vertically

# data with stretching and pitching

new\_data = wav\_stretch(data)

data\_stretch\_pitch = add\_pitch(new\_data, sample\_rate, steps=5)

res3 = extract\_features(data\_stretch\_pitch)

result = np.vstack((result, res3)) # stacking vertically

return result

X, Y1,Y2 = [], [], []

for file\_path, tone\_level, emotion in zip(HED\_DATA.File\_Path, HED\_DATA.Tone\_Level, HED\_DATA.Emotion):

feature = get\_features(file\_path)

for element in feature:

X.append(element)

Y1.append(tone\_level)

Y2.append(emotion)

len(X), len(Y1), len(Y2), HED\_DATA.File\_Path.shape

HED\_Features = pd.DataFrame(X)

HED\_Features['Tone\_Level'] = Y1

HED\_Features['Emotion'] = Y2

HED\_Features.to\_csv('HED\_Spectral\_Features.csv', index=False)

HED\_Features.head()

X\_train = HED\_Features.iloc[: ,:-2].values

Y1\_train = HED\_Features['Tone\_Level'].values

Y2\_train = HED\_Features['Emotion'].values

encoder = OneHotEncoder()

Y1 = encoder.fit\_transform(np.array(Y1\_train).reshape(-1,1)).toarray()

Y2 = encoder.fit\_transform(np.array(Y2\_train).reshape(-1,1)).toarray()

x\_train, x\_test, y1\_train, y1\_test, y2\_train, y2\_test = train\_test\_split(X\_train, Y1, Y2, random\_state=0, shuffle=True)

x\_train.shape, y1\_train.shape, y2\_train.shape, x\_test.shape, y1\_test.shape, y2\_test.shape

X\_train

scaler = StandardScaler()

x\_train = scaler.fit\_transform(x\_train)

x\_test = scaler.transform(x\_test)

x\_train.shape, y1\_train.shape, y2\_train.shape, x\_test.shape, y1\_test.shape, y2\_test.shape

# Baseline model

from tensorflow.keras.layers import Input, Conv1D, Activation, Flatten, Dense

from tensorflow.keras.models import Model

# Define input layer

inputs = Input(shape=(x\_train.shape[1], 1))

# Define convolutional layers

conv = Conv1D(64, 8, padding='same')(inputs)

conv = Activation('relu')(conv)

# Define flatten layer

flatten = Flatten()(conv)

# Define output layers

output\_1 = Dense(y1\_train.shape[1], activation='softmax')(flatten)

output\_2 = Dense(y2\_train.shape[1], activation='softmax')(flatten)

# Define the model with input and output layers

model = Model(inputs=inputs, outputs=[output\_1, output\_2])

# Compile the model

model.compile(loss='categorical\_crossentropy', optimizer='adam', metrics=['accuracy'])

# Train the model

history = model.fit(x\_train, [y1\_train, y2\_train], epochs=100, batch\_size=64, validation\_data=(x\_test, [y1\_test, y2\_test]))

import matplotlib.pyplot as plt

# Get training and validation accuracy for each epoch

acc = history.history['dense\_accuracy']

val\_acc = history.history['val\_dense\_accuracy']

acc2 = history.history['dense\_1\_accuracy']

val\_acc2 = history.history['val\_dense\_1\_accuracy']

# Plot the training and validation accuracy for the first output

plt.plot(range(1, len(acc) + 1), acc, label='Training Accuracy')

plt.plot(range(1, len(val\_acc) + 1), val\_acc, label='Validation Accuracy')

plt.title('Training and Validation Accuracy (Output 1)')

plt.xlabel('Epoch')

plt.ylabel('Accuracy')

plt.legend()

plt.show()

# Plot the training and validation accuracy for the second output

plt.plot(range(1, len(acc2) + 1), acc2, label='Training Accuracy')

plt.plot(range(1, len(val\_acc2) + 1), val\_acc2, label='Validation Accuracy')

plt.title('Training and Validation Accuracy (Output 2)')

plt.xlabel('Epoch')

plt.ylabel('Accuracy')

plt.legend()

plt.show()

import matplotlib.pyplot as plt

# Plot the training loss for both output layers

plt.plot(history.history['dense\_loss'])

plt.plot(history.history['dense\_1\_loss'])

plt.title('Training Loss')

plt.ylabel('Loss')

plt.xlabel('Epoch')

plt.legend(['y1\_train', 'y2\_train'], loc='upper right')

plt.show()

# Plot the validation loss for both output layers

plt.plot(history.history['val\_dense\_loss'])

plt.plot(history.history['val\_dense\_1\_loss'])

plt.title('Validation Loss')

plt.ylabel('Loss')

plt.xlabel('Epoch')

plt.legend(['y1\_test', 'y2\_test'], loc='upper right')

plt.show()

The entire implemented source code for this project is pushed to the GitHub link given below. Navigate to the below link to watch the project progress in detail.

**REFERENCES**

[1] www.kaggle.com. (2019). *CREMA-D*. [online] Available at: <https://www.kaggle.com/datasets/ejlok1/cremad>.

[2] GitHub. (2023). *CheyneyComputerScience/CREMA-D*. [online] Available at: <https://github.com/CheyneyComputerScience/CREMA-D>.

[3] M. Gokilavani, Harshith Katakam, Basheer, S. and Srinivas, S. (2022). Ravdness, Crema-D, Tess Based Algorithm for Emotion Recognition Using Speech. *2022 4th International Conference on Smart Systems and Inventive Technology (ICSSIT)*. doi:https://doi.org/10.1109/icssit53264.2022.9716313.

[4] ‌MD Rizwanul Kabir, Muhammad Muhaimin, Md. Abrar Mahir, Mirza Muntasir Nishat, Faisal, F. and Nchouwat N.I. Moubarak (2021). Procuring MFCCs from Crema-D Dataset for Sentiment Analysis using Deep Learning Models with Hyperparameter Tuning. *2021 IEEE International Conference on Robotics, Automation, Artificial-Intelligence and Internet-of-Things (RAAICON)*. doi:https://doi.org/10.1109/raaicon54709.2021.9929975.

[5] Cao, H., Cooper, D.G., Keutmann, M.K., Gur, R.C., Nenkova, A. and Verma, R. (2014). CREMA-D: Crowd-Sourced Emotional Multimodal Actors Dataset. *IEEE Transactions on Affective Computing*, [online] 5(4), pp.377–390. doi:https://doi.org/10.1109/TAFFC.2014.2336244.

‌[6] Tyagi, S. and Sandor Szenasi (2022). Emotion Extraction from Speech using Deep Learning. *2022 IEEE 20th Jubilee World Symposium on Applied Machine Intelligence and Informatics (SAMI)*. doi:https://doi.org/10.1109/sami54271.2022.9780779.

[7] M Kavitha, B Sasivardhan, P Mani Deepak and M Kalyani (2022). Deep Learning based Audio Processing Speech Emotion Detection. *2022 6th International Conference on Electronics, Communication and Aerospace Technology*. doi:https://doi.org/10.1109/iceca55336.2022.10009064.

[8] Aguiar, R.L., Costa, Y.M.G. and Silla, C.N. (2018). Exploring Data Augmentation to Improve Music Genre Classification with ConvNets. *2018 International Joint Conference on Neural Networks (IJCNN)*. doi:https://doi.org/10.1109/ijcnn.2018.8489166.

‌

‌

‌

‌