National College of Ireland

Configuration Manual

MSc Research Project
Data Analytics

Priyanka Prashant Chimthankar
Student ID: x19241721

School of Computing
National College of Ireland

Supervisor: Dr. Christian Horn

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Priyanka Prashant Chimthankar
Student ID:	x19241721
Programme:	Data Analytics
Year:	2020-21
Module:	MSc Research Project
Supervisor:	Dr. Christian Horn
Submission Due Date:	16/08/2021
Project Title:	Configuration Manual
Word Count:	807
Page Count:	18

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Priyanka Prashant Chimthankar
Date:	14th August 2021

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Priyanka Prashant Chimthankar
x19241721

1 Introduction

This configuration manual contains all of the information needed to execute the artefact. It includes a summary of the minimum and recommended requirements for replicating the thesis work. This handbook serves as a link between the artefact and the thesis. Along with the software and hardware prerequisites for this thesis, all of the main components of this thesis are described in detail here via the use of code snippets. It contains instructions for collecting the data, executing the artefact, and presenting the artefact's significant results.

2 Required Specifications

2.1 Hardware

The Figure 1 describes the hardware/device specifications used for this thesis work.

2.2 Software

To continue with running the artefacts, the following software must be installed and working on your system.

1. Anaconda Navigator for Windows (Version 1.9.12)
2. Jupyter Notebook (Version 6.0.3)
3. Python(version 3.8.3)

3 Data Collection

This research made use of five distinct datasets. All of these datasets were obtained through Kaggle's public repository. The links to these data sets are provided below. Each of these datasets contains audio files in the.wav format with recorded utterances labeled with different emotions.

1. Surrey Audio-Visual Expressed Emotion (SAVEE):`https://www.kaggle.com/ejlok1/surrey-audiovisual-expressed-emotion-savee`
2. Toronto emotional speech set (TESS):`https://www.kaggle.com/ejlok1/toronto-emotional-s`

Figure 1: Hardware Specifications

3. Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS): `https://www.kaggle.com/uwrfkaggler/ravdess-emotional-speech-audio`

4. Crowd Sourced Emotional Multimodal Actors Dataset (CREMA-D): `https://www.kaggle.com/ejlok1/cremad`

5. Berlin Database of Emotional Speech (Emo-DB): `https://www.kaggle.com/piyushagni5/berlin-database-of-emotional-speech-emodb`

4 Exploratory Data Analysis

4.1 Importing required libraries

All the the necessary libraries that are to be imported to run the code are listed below.

```
# Importing required libraries
# Keras
import keras
from keras import regularizers
from keras.preprocessing import sequence
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
from keras.models import Sequential, Model, model_from_json
from keras.layers import Dense, Embedding, LSTM
from keras.layers import Input, Flatten, Dropout, Activation,
    BatchNormalization
from keras.layers import Conv1D, MaxPooling1D, AveragePooling1D,Conv2D,
    MaxPooling2D
from tensorflow.keras.utils import to_categorical
from keras.utils import np_utils
from keras.callbacks import ModelCheckpoint
from keras import losses, models, optimizers

```

```
17
18 # sklearn
19 from sklearn.metrics import confusion_matrix, accuracy_score,
       classification_report
20 from sklearn.model_selection import train_test_split
21 from sklearn.preprocessing import LabelEncoder
22
23 # Other
24 import librosa
25 import librosa.display
26 import json
27 import numpy as np
28 import matplotlib.pyplot as plt
29 import tensorflow as tf
30 from matplotlib.pyplot import specgram
31 import pandas as pd
32 import seaborn as sns
33 import glob
34 import os
35 import sys
36 import pickle
37 import IPython.display as ipd  # To play sound in the notebook
38 import warnings
39 # ignore warnings
40 if not sys.warnoptions:
41     warnings.simplefilter("ignore")
42 warnings.filterwarnings("ignore", category=DeprecationWarning)
```

Listing 1: Required Python Libraries

4.2 Loading all four datasets

The audio files in each dataset have different naming conventions, so to extract the labels from each file, parsing of the filename is done.For each dataset seperate code blocks are written as shown below.

```
1 # Get the data location for SAVEE
2 SAVEE_list = os.listdir(SAVEE)
3
4 # parse the filename to get the emotions
5 emotion=[]
6 path = []
7 duration=[]
8 for i in SAVEE_list:
9     if i[-8:-6]=='_a':
10         emotion.append('male_angry')
11     elif i[-8:-6]=='_d':
12         emotion.append('male_disgust')
13     elif i[-8:-6]=='_f':
14         emotion.append('male_fear')
15     elif i[-8:-6]=='_h':
16         emotion.append('male_happy')
17     elif i[-8:-6]=='_n':
18         emotion.append('male_neutral')
19     elif i[-8:-6]=='sa':
20         emotion.append('male_sad')
21     elif i[-8:-6]=='su':
```

```
        emotion.append('male_surprise')
    else:
        emotion.append('male_error')
    path.append(SAVEE + i)
    duration.append(round(librosa.get_duration(filename=SAVEE+i),2))

SAVEE_df = pd.DataFrame(emotion, columns = ['labels'])
SAVEE_df['source'] = 'SAVEE'
SAVEE_df = pd.concat([SAVEE_df, pd.DataFrame(path, columns = ['path'])
   ], axis = 1)
SAVEE_df = pd.concat([SAVEE_df, pd.DataFrame(duration, columns = ['
   duration_seconds'])], axis = 1)

SAVEE_df.labels.value_counts()
```

Listing 2: Loading SAVEE dataset

```
path = []
emotion = []
duration=[]
for i in TESS_list:
    fname = os.listdir(TESS + i)
    for f in fname:
        if i == 'OAF_angry' or i == 'YAF_angry':
            emotion.append('female_angry')
        elif i == 'OAF_disgust' or i == 'YAF_disgust':
            emotion.append('female_disgust')
        elif i == 'OAF_Fear' or i == 'YAF_fear':
            emotion.append('female_fear')
        elif i == 'OAF_happy' or i == 'YAF_happy':
            emotion.append('female_happy')
        elif i == 'OAF_neutral' or i == 'YAF_neutral':
            emotion.append('female_neutral')
        elif i == 'OAF_Pleasant_surprise' or i == '
   YAF_pleasant_surprised':
            emotion.append('female_surprise')
        elif i == 'OAF_Sad' or i == 'YAF_sad':
            emotion.append('female_sad')
        else:
            emotion.append('Unknown')
        path.append(TESS + i + "/" + f)
        duration.append(round(librosa.get_duration(filename=(TESS + i +
    "/" + f)),2))

TESS_df = pd.DataFrame(emotion, columns = ['labels'])
TESS_df['source'] = 'TESS'
TESS_df = pd.concat([TESS_df,pd.DataFrame(path, columns = ['path'])],
   axis=1)
TESS_df = pd.concat([TESS_df, pd.DataFrame(duration, columns = ['
   duration_seconds'])], axis = 1)

TESS_df.labels.value_counts()
```

Listing 3: Loading TESS dataset

```
RAV_list = os.listdir(RAV)
RAV_list.sort()

emotion = []
```

```
gender = []
path = []
duration=[]
for i in RAV_list:
    fname = os.listdir(RAV + i)
    for f in fname:
        part = f.split('.')[0].split('-')
        emotion.append(int(part[2]))
        temp = int(part[6])
        if temp%2 == 0:
            temp = "female"
        else:
            temp = "male"
        gender.append(temp)
        path.append(RAV + i + '/' + f)
        duration.append(round(librosa.get_duration(filename=(RAV + i + '/' + f)),2))

RAV_df = pd.DataFrame(emotion)
RAV_df = RAV_df.replace({1:'neutral', 2:'neutral', 3:'happy', 4:'sad', 5:'angry', 6:'fear', 7:'disgust', 8:'surprise'})
RAV_df = pd.concat([pd.DataFrame(gender),RAV_df],axis=1)
RAV_df.columns = ['gender','emotion']
RAV_df['labels'] =RAV_df.gender + '_' + RAV_df.emotion
RAV_df['source'] = 'RAVDESS'
RAV_df = pd.concat([RAV_df,pd.DataFrame(path, columns = ['path'])],axis=1)
RAV_df = pd.concat([RAV_df,pd.DataFrame(duration, columns = ['duration_seconds'])],axis=1)
RAV_df = RAV_df.drop(['gender', 'emotion'], axis=1)
RAV_df.labels.value_counts()
```

Listing 4: Loading RAVDESS dataset

```
gender = []
emotion = []
path = []
duration=[]

female = [
1002,1003,1004,1006,1007,1008,1009,1010,1012,1013,1018,
1020,1021,1024,1025,1028,1029,1030,1037,1043,1046,1047,
1049,1052,1053,1054,1055,1056,1058,1060,1061,1063,1072,
1073,1074,1075,1076,1078,1079,1082,1084,1089,1091
]
for i in CREMA_list:
    part = i.split('_')
    if int(part[0]) in female:
        temp = 'female'
    else:
        temp = 'male'
    gender.append(temp)
    if part[2] == 'SAD' and temp == 'male':
        emotion.append('male_sad')
    elif part[2] == 'ANG' and temp == 'male':
        emotion.append('male_angry')
    elif part[2] == 'DIS' and temp == 'male':
        emotion.append('male_disgust')
    elif part[2] == 'FEA' and temp == 'male':
```

```
        emotion.append('male_fear')
    elif part[2] == 'HAP' and temp == 'male':
        emotion.append('male_happy')
    elif part[2] == 'NEU' and temp == 'male':
        emotion.append('male_neutral')
    elif part[2] == 'SAD' and temp == 'female':
        emotion.append('female_sad')
    elif part[2] == 'ANG' and temp == 'female':
        emotion.append('female_angry')
    elif part[2] == 'DIS' and temp == 'female':
        emotion.append('female_disgust')
    elif part[2] == 'FEA' and temp == 'female':
        emotion.append('female_fear')
    elif part[2] == 'HAP' and temp == 'female':
        emotion.append('female_happy')
    elif part[2] == 'NEU' and temp == 'female':
        emotion.append('female_neutral')
    else:
        emotion.append('Unknown')
    path.append(CREMA + i)
    duration.append(round(librosa.get_duration(filename=(CREMA + i)),2)
   )
CREMA_df = pd.DataFrame(emotion, columns = ['labels'])
CREMA_df['source'] = 'CREMA'
CREMA_df = pd.concat([CREMA_df,pd.DataFrame(path, columns = ['path'])],
   axis=1)
CREMA_df = pd.concat([CREMA_df,pd.DataFrame(duration, columns = ['
   duration_seconds'])],axis=1)

CREMA_df.labels.value_counts()
```

Listing 5: Loading CREMA-D dataset

4.3 EDA using wave plots and playing sample audio clips

The below code is used to explore the audio files in each dataset by plotting wave plots for visualizations as shown in Figure 2

```
#Librosa library is used for this task
fname = SAVEE + 'DC_f09.wav'
data, sampling_rate = librosa.load(fname)
plt.figure(figsize=(15, 5))
librosa.display.waveplot(data, sr=sampling_rate)

#Playing a sample of fearful speech
ipd.Audio(fname)
```

Listing 6: Plotting Wave plots and playing sample audio files

4.4 Generating Metadata file combined for all four datasets

Further for building a baseline model as well as for building the proposed model this metadata file will be used which contains all the information for all the four datasets combined together.The code for generating this file is given below.

```
df = pd.concat([SAVEE_df, RAV_df, TESS_df, CREMA_df], axis = 0)
print(df.labels.value_counts())
```

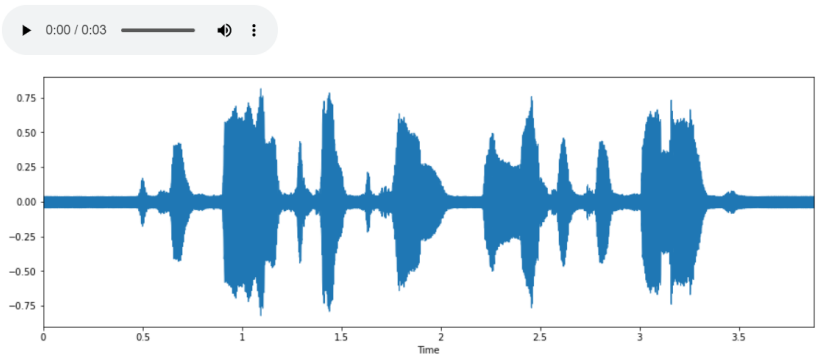

Figure 2: Sample wave plot and play track

```
df.head()
df.to_csv("../data/Metadata_merged_datasets.csv",index=False)
```

Listing 7: Generating Metadata file

4.5 Exploring MFCC Feature

This code block is used to display the sample wave plot of audio clips and the respective MFCC feature visualization as shown in Figure 3.

```
# Source - RAVDESS; Gender - Female; Emotion - Angry
path = "../data/RAVDESS/audio_speech_actors_01-24/Actor_08
    /03-01-05-02-01-01-08.wav"
X, sample_rate = librosa.load(path, res_type='kaiser_fast',duration
    =2.0,sr=22050*2,offset=0.9)
mfcc = librosa.feature.mfcc(y=X, sr=sample_rate, n_mfcc=13)

# audio wave
plt.figure(figsize=(20, 15))
plt.subplot(3,1,1)
librosa.display.waveplot(X, sr=sample_rate)
plt.title('Audio sampled at 44100 hrz')

# MFCC
plt.figure(figsize=(20, 15))
plt.subplot(3,1,1)
librosa.display.specshow(mfcc, x_axis='time')
plt.ylabel('MFCC')
plt.colorbar()

ipd.Audio(path)
```

Listing 8: MFCC Feature Visualization

4.6 Plotting MFCC feature for distinguishing between male and female bands

The difference in MFCC features for male and females in audio samples with same sentence uttered with same emotions can be seen in Figure 4. The code block for the same is given below.

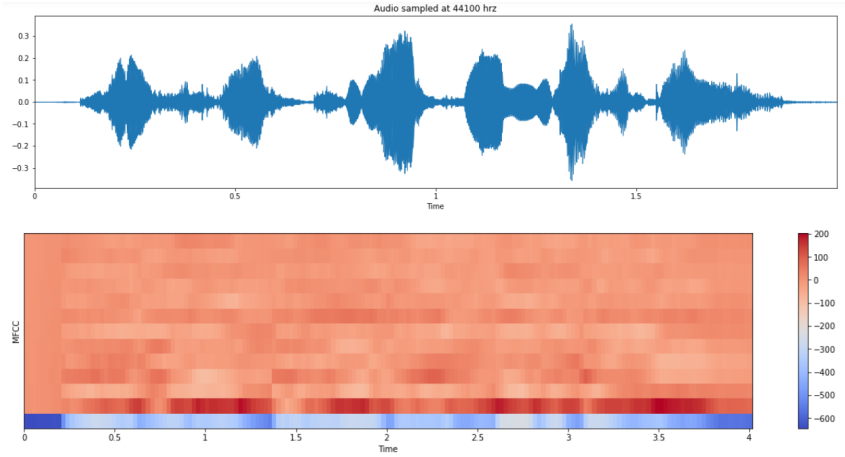

Figure 3: Sample wave plot and MFCC Feature visualization

```
# Source - RAVDESS; Gender - Female; Emotion - Angry
path = "../data/RAVDESS/audio_speech_actors_01-24/Actor_08
    /03-01-05-02-01-01-08.wav"
X, sample_rate = librosa.load(path, res_type='kaiser_fast',duration
    =2.5,sr=22050*2,offset=0.5)
female = librosa.feature.mfcc(y=X, sr=sample_rate, n_mfcc=13)
female = np.mean(librosa.feature.mfcc(y=X, sr=sample_rate, n_mfcc=13),
    axis=0)
print(len(female))

# Source - RAVDESS; Gender - Male; Emotion - Angry
path = "../data/RAVDESS/audio_speech_actors_01-24/Actor_09
    /03-01-05-01-01-01-09.wav"
X, sample_rate = librosa.load(path, res_type='kaiser_fast',duration
    =2.5,sr=22050*2,offset=0.5)
male = librosa.feature.mfcc(y=X, sr=sample_rate, n_mfcc=13)
male = np.mean(librosa.feature.mfcc(y=X, sr=sample_rate, n_mfcc=13),
    axis=0)
print(len(male))

# audio wave
plt.figure(figsize=(20, 15))
plt.subplot(3,1,1)
plt.plot(female, label='female')
plt.plot(male, label='male')
plt.legend()
```

Listing 9: Gender differentiation in MFCC Feature bands

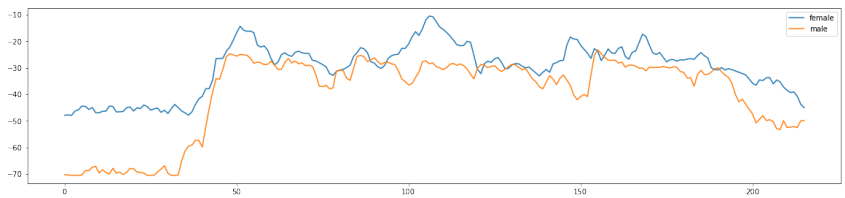

Figure 4: Gender differentiation in MFCC Feature

5 Baseline Model

A baseline model is run on each dataset to verify the CNN architecture's generic structure. This has aided in the preparation of the datasets for modeling and also in the development of a baseline method for constructing a Speech Emotion REcognition (SER) system using the suggested architecture. The code for the basic model architecture is provided below. The basic model is shared across all datasets and is then applied to all four datasets collectively through a merged dataset.Below are the results obtained after applying this baseline model on TESS dataset,on similar lines the results can be obtained for all other datasets as well as the merged dataset.

5.1 Count plot distribution of labels in each dataset

The below code is used to plot the count of emotion labels in each dataset to know the distribution along the dataset. The Figure 5 displays the count plot for distribution of emotions in TESS dataset.

```
plt.title('Count of Emotions', size=14)
sns.countplot(TESS.labels)
plt.ylabel('Count', size=12)
plt.xlabel('Emotions', size=12)
plt.xticks(size=7)

sns.despine(top=True, right=True, left=False, bottom=False)
plt.show()
```

Listing 10: Creating Count plot for distribution of labels a dataset

Figure 5: Count plot of distribution of emotion labels in TESS dataset

5.2 Extracting MFCC features in the form of 2D matrix

Looping over the entire dataset for extracting MFCC features from each audio file in the form of 2D matrix.

```
df = pd.DataFrame(columns=['feature'])

# loop feature extraction over the entire TESS dataset
counter=0
for index,path in enumerate(TESS.path):
    X, sample_rate = librosa.load(path
                                  , res_type='kaiser_fast'
                                  ,duration=1.7
                                  ,sr=44100
                                  ,offset=0.3
                                 )
    sample_rate = np.array(sample_rate)

    mfccs = np.mean(librosa.feature.mfcc(y=X,
                                         sr=sample_rate,
                                         n_mfcc=13),
                    axis=0)
    df.loc[counter] = [mfccs]
    counter=counter+1

# Check a few records to make sure it's processed successfully
print(len(df))
df.head()
```

Listing 11: Extracting MFCC features in 2D matrix form

5.3 Data Processing

```

df = pd.concat([TESS,pd.DataFrame(df['feature'].values.tolist())],axis
    =1)

# replace NA with 0
df=df.fillna(0)
```

Listing 12: Data Processing

5.4 Train-Test Data Split

```
# Split between train and test
X_train, X_test, y_train, y_test = train_test_split(df.drop(['path','
    labels','source','duration_seconds','index'],axis=1)
                                                    , df.labels
                                                    , test_size=0.2
                                                    , shuffle=True
                                                    , random_state=42
                                                   )
```

Listing 13: Train-Test Data Split

5.5 Data Normalization

```
mean = np.mean(X_train, axis=0)
std = np.std(X_train, axis=0)

```

```
X_train = (X_train - mean)/std
X_test = (X_test - mean)/std
```

Listing 14: Data Normalization

5.6 Model architecture

This model architecture is common and used to run against each dataset. The baseline model using CNN is given below.

```
def basemodel(num_classes):
    model = Sequential()
    model.add(Conv1D(256, 8, padding='same',input_shape=(X_train.shape[1],1)))  # X_train.shape[1] = No. of Columns
    model.add(Activation('relu'))
    model.add(Conv1D(256, 8, padding='same'))
    model.add(BatchNormalization())
    model.add(Activation('relu'))
    model.add(Dropout(0.25))
    model.add(MaxPooling1D(pool_size=(8)))
    model.add(Conv1D(128, 8, padding='same'))
    model.add(Activation('relu'))
    model.add(Conv1D(128, 8, padding='same'))
    model.add(Activation('relu'))
    model.add(Conv1D(128, 8, padding='same'))
    model.add(Activation('relu'))
    model.add(Conv1D(128, 8, padding='same'))
    model.add(BatchNormalization())
    model.add(Activation('relu'))
    model.add(Dropout(0.25))
    model.add(MaxPooling1D(pool_size=(8)))
    model.add(Conv1D(64, 8, padding='same'))
    model.add(Activation('relu'))
    model.add(Conv1D(64, 8, padding='same'))
    model.add(Activation('relu'))
    model.add(Flatten())
    model.add(Dense(num_classes)) # Target class number
    model.add(Activation('softmax'))
    # opt = keras.optimizers.SGD(lr=0.0001, momentum=0.0, decay=0.0, nesterov=False)
    # opt = keras.optimizers.Adam(lr=0.0001)
    # opt = optimizers.RMSprop(lr=0.00001, decay=1e-6)
    model.compile(loss='categorical_crossentropy', optimizer='RMSProp', metrics=['accuracy'])
    model.summary()
    return model
```

Listing 15: Model architecture

5.7 Plotting model loss and accuracy

The code below is used for plotting training loss vs validation loss as well as training accuracy vs validation accuracy of the models.Figure 6 shows the plots for TESS dataset baseline model.

```
def create_plot():
    plt.plot(model_history.history['loss'])
```

```
    plt.plot(model_history.history['val_loss'])
    plt.title('model loss')
    plt.ylabel('loss')
    plt.xlabel('epoch')
    plt.legend(['train', 'test'], loc='upper left')
    plt.show()

    plt.plot(model_history.history['accuracy'])
    plt.plot(model_history.history['val_accuracy'])
    plt.title('model accuracy')
    plt.ylabel('accuracy')
    plt.xlabel('epoch')
    plt.legend(['train', 'test'], loc='upper left')
    plt.show()
```

Listing 16: Loss and Accuracy plot

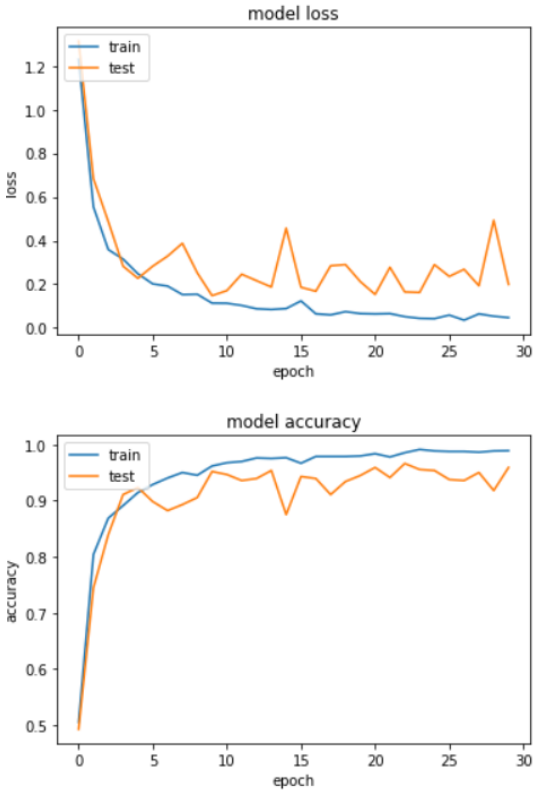

Figure 6: Loss and Accuracy plot for Tess dataset

5.8 Model Serialisation

```
# Save model and weights
model_name = 'Emotion_TESS_Model.h5'
save_dir = os.path.join(os.getcwd(), 'saved_models')

if not os.path.isdir(save_dir):
    os.makedirs(save_dir)
model_path = os.path.join(save_dir, model_name)
```

```
model.save(model_path)
print('Save model and weights at %s ' % model_path)

# Save the model to disk
model_json = model.to_json()
with open("model_json_TESS.json", "w") as json_file:
    json_file.write(model_json)
```

Listing 17: Saving model for re-usability

5.9 Model Validation

5.9.1 Actual vs PRedicted values

Figure 7 displays the actual values of emotions vs the predicted values

```
# predictions
preds = preds.astype(int).flatten()
preds = (lb.inverse_transform((preds)))
preds = pd.DataFrame({'predictedvalues': preds})

# Actual labels
actual=y_test.argmax(axis=1)
actual = actual.astype(int).flatten()
actual = (lb.inverse_transform((actual)))
actual = pd.DataFrame({'actualvalues': actual})

# Lets combined both of them into a single dataframe
finaldf = actual.join(preds)
finaldf[170:180]
```

Listing 18: Actual vs Predicted values

	actualvalues	predictedvalues
170	female_angry	female_angry
171	female_disgust	female_disgust
172	female_disgust	female_disgust
173	female_sad	female_sad
174	female_disgust	female_disgust
175	female_happy	female_surprise
176	female_neutral	female_neutral
177	female_fear	female_fear
178	female_surprise	female_surprise
179	female_fear	female_fear

Figure 7: Actual vs Predicted values

5.9.2 Plotting heat map for Confusion Matrix

Figure 8 displays the heatmap plotted for confusion matrix results for TESS dataset baseline model validation.

```
# the confusion matrix heat map plot
def print_confusion_matrix(confusion_matrix, class_names, figsize =
    (10,7), fontsize=14):

    df_cm = pd.DataFrame(
        confusion_matrix, index=class_names, columns=class_names,
    )
    fig = plt.figure(figsize=figsize)
    try:
        heatmap = sns.heatmap(df_cm, annot=True, fmt="d")
    except ValueError:
        raise ValueError("Confusion matrix values must be integers.")

    heatmap.yaxis.set_ticklabels(heatmap.yaxis.get_ticklabels(),
    rotation=0, ha='right', fontsize=fontsize)
    heatmap.xaxis.set_ticklabels(heatmap.xaxis.get_ticklabels(),
    rotation=45, ha='right', fontsize=fontsize)
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
```

Listing 19: Heatmap for confusion matrix

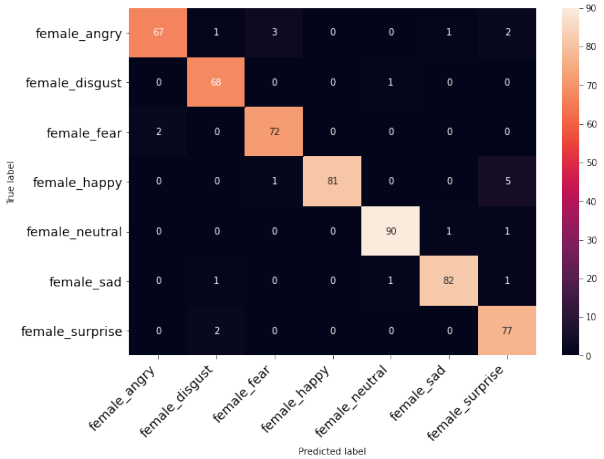

Figure 8: Heatmap plot for confusion matrix

5.9.3 Classification Report

Figure 9 shows the classification report generated for TESS dataset for baseline model implementation

```
# Classification report
classes = finaldf.actualvalues.unique()
classes.sort()
print(classification_report(finaldf.actualvalues, finaldf.
    predictedvalues, target_names=classes))
```

Listing 20: Classification Report Generation

	precision	recall	f1-score	support
female_angry	0.97	0.91	0.94	74
female_disgust	0.94	0.99	0.96	69
female_fear	0.95	0.97	0.96	74
female_happy	1.00	0.93	0.96	87
female_neutral	0.98	0.98	0.98	92
female_sad	0.98	0.96	0.97	85
female_surprise	0.90	0.97	0.93	79
accuracy			0.96	560
macro avg	0.96	0.96	0.96	560
weighted avg	0.96	0.96	0.96	560

Figure 9: Classification Report for Baseline model against TESS dataset

6 Gender-based performance of the model

The merged dataset is divided into two parts based on the gender of the speakers. So there is one dataset that contains only male speakers audio files and one with only females.

6.1 Division of dataset

```
#Data containing only audio files that have male speakers
MALE=ref[ref['labels'].str.startswith('male')].reset_index()

#Data containing only audio files that have female speakers
FEMALE=ref[ref['labels'].str.startswith('female')].reset_index()
```

Listing 21: Division of dataset based on gender of speaker

6.2 Confusion matrix for gender based emotion classification

```
modidf = finaldf
modidf['actualvalues'] = finaldf.actualvalues.replace({'female_angry':'
    female'
                                          , 'female_disgust':'female'
                                          , 'female_fear':'female'
                                          , 'female_happy':'female'
                                          , 'female_sad':'female'
                                          , 'female_surprise':'female'
                                          , 'female_neutral':'female'
                                          , 'male_angry':'male'
                                          , 'male_fear':'male'
                                          , 'male_happy':'male'
                                          , 'male_sad':'male'
                                          , 'male_surprise':'male'
                                          , 'male_neutral':'male'
                                          , 'male_disgust':'male'
                                         })

modidf['predictedvalues'] = finaldf.predictedvalues.replace({'
    female_angry':'female'
                                          , 'female_disgust':'female'
                                          , 'female_fear':'female'
                                          , 'female_happy':'female'
                                          , 'female_sad':'female'
```

```
                                          , 'female_surprise':'female'
                                          , 'female_neutral':'female'
                                          , 'male_angry':'male'
                                          , 'male_fear':'male'
                                          , 'male_happy':'male'
                                          , 'male_sad':'male'
                                          , 'male_surprise':'male'
                                          , 'male_neutral':'male'
                                          , 'male_disgust':'male'
                                         })

classes = modidf.actualvalues.unique()
classes.sort()

# Confusion matrix
c = confusion_matrix(modidf.actualvalues, modidf.predictedvalues)
print(accuracy_score(modidf.actualvalues, modidf.predictedvalues))
print_confusion_matrix(c, class_names = classes)
```

Listing 22: Division of dataset based on gender of speaker

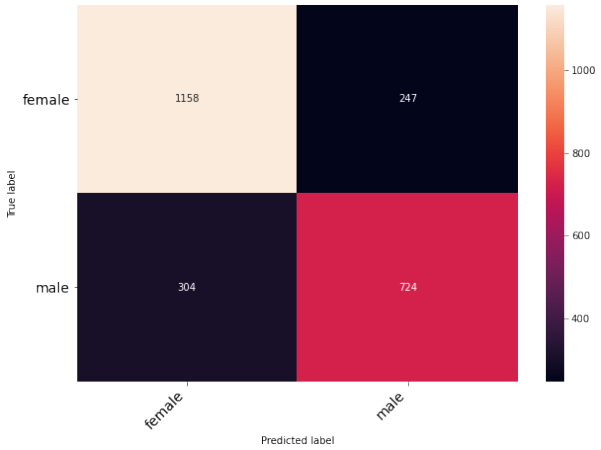

Figure 10: Confusion matrix for gender based emotion classification

7 Proposed Model-2D CNN-LSTM

The proposed architecture is given below. The reference paper by author Zhao et al. (2019) explains a similar architecture for building speech emotion recognition system.

```
def get_2d_cnn_lstm_model(n):

    num_classes = 14
    input_shape = (n,216,1)
    model = keras.Sequential(name='model2d')
    print(input_shape)
    #LFLB1
    model.add(layers.Conv2D(filters=64,
                            kernel_size=3,
                            strides=1,
                            padding='same',
```

```
                            # data_format='channels_first',
                            input_shape=input_shape
                            )
            )
    model.add(layers.BatchNormalization())
    model.add(layers.Activation('relu'))
    model.add(layers.MaxPooling2D(pool_size=2, strides=2))

    #LFLB2
    model.add(layers.Conv2D(filters=64,
                            kernel_size=3,
                            strides=1,
                            padding='same',
                            )
            )
    model.add(layers.BatchNormalization())
    model.add(layers.Activation('relu'))
    model.add(layers.MaxPooling2D(pool_size=2, strides=2))

    #LFLB3
    model.add(layers.Conv2D(filters=128,
                            kernel_size=3,
                            strides=1,
                            padding='same',
                            )
            )
    model.add(layers.BatchNormalization())
    model.add(layers.Activation('relu'))
    model.add(layers.MaxPooling2D(pool_size=2, strides=2))

    #LFLB4
    model.add(layers.Conv2D(filters=128,
                            kernel_size=3,
                            strides=1,
                            padding='same'
                            )
            )
    model.add(layers.BatchNormalization())
    model.add(layers.Activation('relu'))
    model.add(layers.MaxPooling2D(pool_size=2, strides=2))


    model.add(layers.Reshape((-1, 128)))
    #LSTM
    model.add(layers.LSTM(32))

    model.add(layers.Dense(units=num_classes, activation='softmax'))

    #model.summary()
    #opt = optimizers.Adam(0.0006)
    model.compile(loss='categorical_crossentropy', optimizer='Adam',
    metrics=['accuracy'])
    return model
```

Listing 23: Division of dataset based on gender of speaker

References

Zhao, J., Mao, X. and Chen, L. (2019). Speech emotion recognition using deep 1d & 2d cnn lstm networks, *Biomedical Signal Processing and Control* **47**: 312–323.