import pandas as pd #for data analysis and associated manipulation of tabular data in DataFrames

import numpy as np #for working with arrays

import os #to deal with files

import seaborn as sns #for visualisation

import matplotlib.pyplot as plt #graph plotting library

import librosa #for audio analysis

import librosa.display

from IPython.display import Audio

import warnings #to deal with warnings

warnings.filterwarnings('ignore')

paths = []

labels = []

for dirname, \_, filenames in os.walk('/kaggle/input'):

for filename in filenames:

paths.append(os.path.join(dirname, filename))

label = filename.split('\_')[-1]

label = label.split('.')[0]

labels.append(label.lower())

if len(paths) == 2800:

break

print('Dataset is Loaded') Dataset is Loaded len(paths) 2800 paths[:5]

['/kaggle/input/toronto-emotional-speech-set-tess/TESS Toronto emotional speech set data/YAF\_fear/YAF\_home\_fear.wav',

'/kaggle/input/toronto-emotional-speech-set-tess/TESS Toronto emotional speech set data/YAF\_fear/YAF\_youth\_fear.wav',

'/kaggle/input/toronto-emotional-speech-set-tess/TESS Toronto emotional speech set data/YAF\_fear/YAF\_near\_fear.wav',

'/kaggle/input/toronto-emotional-speech-set-tess/TESS Toronto emotional speech set data/YAF\_fear/YAF\_search\_fear.wav',

'/kaggle/input/toronto-emotional-speech-set-tess/TESS Toronto emotional speech set data/YAF\_fear/YAF\_pick\_fear.wav']

labels[:5]

['fear', 'fear', 'fear', 'fear', 'fear']

## Create a dataframe

df = pd.DataFrame()

df['speech'] = paths

df['label'] = labels

df.head()

speech label

0 /kaggle/input/toronto-emotional-speech-set-tes... fear

1 /kaggle/input/toronto-emotional-speech-set-tes... fear

2 /kaggle/input/toronto-emotional-speech-set-tes... fear

3 /kaggle/input/toronto-emotional-speech-set-tes... fear

4 /kaggle/input/toronto-emotional-speech-set-tes... fear

df['label'].value\_counts()

fear 400

angry 400

disgust 400

neutral 400

sad 400

ps 400

happy 400

Name: label, dtype: int64

Data Analysis

sns.countplot(df['label'])

<AxesSubplot:xlabel='label', ylabel='count'>

def waveshow(data, sr, emotion):

plt.figure(figsize=(10,4))

plt.title(emotion, size =20)

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

plt.show

def spectogram(data, sr, emotion):

x = librosa.stft(data)

xdb = librosa.amplitude\_to\_db(abs(x))

plt.figure(figsize=(11,4))

plt.title(emotion, size =20)

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

plt.colorbar()

emotion = 'fear'

path = np.array(df['speech'][df['label']==emotion])[0]

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

waveshow(data, sampling\_rate, emotion)

spectogram(data, sampling\_rate, emotion)

Audio(path)

emotion = 'angry'

path = np.array(df['speech'][df['label']==emotion])[50]

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

waveshow(data, sampling\_rate, emotion)

spectogram(data, sampling\_rate, emotion)

Audio(path)

emotion = 'disgust'

path = np.array(df['speech'][df['label']==emotion])[0]

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

waveshow(data, sampling\_rate, emotion)

spectogram(data, sampling\_rate, emotion)

Audio(path)

emotion = 'neutral'

path = np.array(df['speech'][df['label']==emotion])[399]

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

waveshow(data, sampling\_rate, emotion)

spectogram(data, sampling\_rate, emotion)

Audio(path)

emotion = 'sad'

path = np.array(df['speech'][df['label']==emotion])[399]

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

waveshow(data, sampling\_rate, emotion)

spectogram(data, sampling\_rate, emotion)

Audio(path)

emotion = 'ps'

path = np.array(df['speech'][df['label']==emotion])[200]

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

waveshow(data, sampling\_rate, emotion)

spectogram(data, sampling\_rate, emotion)

Audio(path)

emotion = 'happy'

path = np.array(df['speech'][df['label']==emotion])[0]

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

waveshow(data, sampling\_rate, emotion)

spectogram(data, sampling\_rate, emotion)

Adef extract\_mfcc(filename):

y, sr = librosa.load(filename, duration=3, offset=0.5)

mfcc = np.mean(librosa.feature.mfcc(y=y, sr=sr, n\_mfcc=40).T, axis=0)

return mfcc

extract\_mfcc(df['speech'][0])

array([-287.13037 , 87.756935 , -4.139177 , 24.081968 ,

-16.696724 , 12.970632 , 10.522444 , -1.1463214 ,

-0.73337686, 12.855532 , -19.147291 , -6.418063 ,

4.9657683 , -2.6571155 , -10.655444 , 4.9578815 ,

-14.55586 , 15.37587 , 18.444935 , 23.878317 ,

31.495146 , 17.326372 , -4.7648373 , 1.7432437 ,

-12.009847 , 7.34574 , -3.2051265 , -7.171453 ,

-11.410634 , -2.001994 , -5.610964 , 4.5321946 ,

-11.396625 , -8.892363 , -3.7391376 , 4.8819685 ,

-1.5599903 , 2.465447 , 11.59915 , 11.042192 ],

dtype=float32)

x\_mfcc = df['speech'].apply(lambda x:extract\_mfcc(x))

x\_mfcc

0 [-287.13037, 87.756935, -4.139177, 24.081968, ...

1 [-350.0836, 37.654167, -6.2928553, 17.09615, 4...

2 [-341.78152, 56.153652, -16.617884, 23.219698,...

3 [-309.17456, 24.854897, -8.00109, 10.065497, -...

4 [-347.12918, 49.69155, -27.524876, 22.730288, ...

...

2795 [-376.5839, 63.9598, -3.0598662, 11.498796, -2...

2796 [-316.5801, 43.54606, -9.336959, -0.198444, -5...

2797 [-359.7638, 81.01536, -18.355762, 5.3012295, -...

2798 [-354.38315, 103.432144, -15.916284, -10.30884...

2799 [-391.15958, 56.44471, -1.0464002, 0.9587419, ...

Name: speech, Length: 2800, dtype: object

x = [x for x in x\_mfcc]

x = np.array(x)

x.shape

(2800, 40)

##input split

x = np.expand\_dims(x, -1)

x.shape

(2800, 40, 1)

from sklearn.preprocessing import OneHotEncoder

enc = OneHotEncoder()

y = enc.fit\_transform(df[['label']])

y = y.toarray()

y.shape

(2800, 7)

from keras.models import Sequential

from keras.layers import Dense, LSTM, Dropout

model = Sequential([

LSTM(256, return\_sequences=False, input\_shape=(40,1)),

Dropout(0.2),

Dense(128, activation='relu'),

Dropout(0.2),

Dense(64, activation='relu'),

Dropout(0.2),

Dense(7, activation='softmax')

])

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

model.summary()

Model: "sequential\_13"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

lstm\_13 (LSTM) (None, 256) 264192

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dropout\_39 (Dropout) (None, 256) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_39 (Dense) (None, 128) 32896

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dropout\_40 (Dropout) (None, 128) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_40 (Dense) (None, 64) 8256

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dropout\_41 (Dropout) (None, 64) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_41 (Dense) (None, 7) 455

=================================================================

Total params: 305,799

Trainable params: 305,799

Non-trainable params: 0

# Train the model

history = model.fit(x, y, validation\_split=0.2, epochs=50, batch\_size=64)

epochs = list(range(50))

acc = history.history['accuracy']

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

plt.plot(epochs, acc, label='train accuracy')

plt.plot(epochs, val\_acc, label='val accuracy')

plt.xlabel('epochs')

plt.ylabel('accuracy')

plt.legend()

plt.show()

loss = history.history['loss']

val\_loss = history.history['val\_loss']

plt.plot(epochs, loss, label='train loss')

plt.plot(epochs, val\_loss, label='val loss')

plt.xlabel('epochs')

plt.ylabel('loss')

plt.legend()

plt.show()