CNN_emotion_recognition

December 7, 2019

1 I. Importing the required libraries

[1]:
```
# Orignial Notebook: https://github.com/MITESHPUTHRANNEU/
↪Speech-Emotion-Analyzer/blob/master/final_results_gender_test.ipynb

## Python
import os
import random
import sys


## Package
import glob
import keras
import IPython.display as ipd
import librosa
import librosa.display
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import plotly.graph_objs as go
import plotly.offline as py
import plotly.tools as tls
import seaborn as sns
import scipy.io.wavfile
import tensorflow as tf
py.init_notebook_mode(connected=True)


## Keras
from keras import regularizers
from keras.callbacks import ModelCheckpoint, LearningRateScheduler,␣
↪EarlyStopping
from keras.callbacks import  History, ReduceLROnPlateau, CSVLogger
from keras.models import Model, Sequential
from keras.layers import Dense, Embedding, LSTM
from keras.layers import Input, Flatten, Dropout, Activation, BatchNormalization
```

```
from keras.layers import Conv1D, MaxPooling1D, AveragePooling1D
from keras.preprocessing import sequence
from keras.preprocessing.sequence import pad_sequences
from keras.preprocessing.text import Tokenizer
from keras.utils import np_utils
from keras.utils import to_categorical


## Sklearn
from sklearn.metrics import confusion_matrix
from sklearn.preprocessing import LabelEncoder


## Rest
from scipy.fftpack import fft
from scipy import signal
from scipy.io import wavfile
from tqdm import tqdm

input_duration=3
# % pylab inline
```

```
Using TensorFlow backend.
```

2 II. Reading the data

[2]:
```
# Data Directory
# Please edit according to your directory change.
dir_list = os.listdir('data/')
dir_list.sort()
print (dir_list)
```

```
['Actor_01', 'Actor_02', 'Actor_03', 'Actor_04', 'Actor_05', 'Actor_06',
'Actor_07', 'Actor_08', 'Actor_09', 'Actor_10', 'Actor_11', 'Actor_12',
'Actor_13', 'Actor_14', 'Actor_15', 'Actor_16', 'Actor_17', 'Actor_18',
'Actor_19', 'Actor_20', 'Actor_21', 'Actor_22', 'Actor_23', 'Actor_24']
```

[3]:
```
# Create DataFrame for Data intel
data_df = pd.DataFrame(columns=['path', 'source', 'actor', 'gender',
                                'intensity', 'statement', 'repetition',␣
 ↪'emotion'])
count = 0
for i in dir_list:
    file_list = os.listdir('data/' + i)
    for f in file_list:
        nm = f.split('.')[0].split('-')
```

```
        path = 'data/' + i + '/' + f
        src = int(nm[1])
        actor = int(nm[-1])
        emotion = int(nm[2])

        if int(actor)%2 == 0:
            gender = "female"
        else:
            gender = "male"

        if nm[3] == '01':
            intensity = 0
        else:
            intensity = 1

        if nm[4] == '01':
            statement = 0
        else:
            statement = 1

        if nm[5] == '01':
            repeat = 0
        else:
            repeat = 1

        data_df.loc[count] = [path, src, actor, gender, intensity, statement,␣
↪repeat, emotion]
        count += 1
```

[4]:
```
print (len(data_df))
data_df.head()
```

1440

[4]:

```
                                        path source actor gender intensity  \
0  data/Actor_01/03-01-01-01-01-01-01.wav      1     1   male         0
1  data/Actor_01/03-01-01-01-01-02-01.wav      1     1   male         0
2  data/Actor_01/03-01-01-01-02-01-01.wav      1     1   male         0
3  data/Actor_01/03-01-01-01-02-02-01.wav      1     1   male         0
4  data/Actor_01/03-01-02-01-01-01-01.wav      1     1   male         0

  statement repetition emotion
0         0          0       1
1         0          1       1
2         1          0       1
3         1          1       1
4         0          0       2
```

3 III. Plotting the audio file's waveform and its spectrogram

[5]:
```
filename = data_df.path[1021]
print (filename)

samples, sample_rate = librosa.load(filename)
sample_rate, samples
```

```
data/Actor_18/03-01-01-01-01-02-18.wav
```

[5]:
```
(22050, array([0., 0., 0., …, 0., 0., 0.], dtype=float32))
```

[6]:
```
len(samples), sample_rate
```

[6]:
```
(77989, 22050)
```

[7]:
```
def log_specgram(audio, sample_rate, window_size=20,
                 step_size=10, eps=1e-10):
    nperseg = int(round(window_size * sample_rate / 1e3))
    noverlap = int(round(step_size * sample_rate / 1e3))
    freqs, times, spec = signal.spectrogram(audio,
                                    fs=sample_rate,
                                    window='hann',
                                    nperseg=nperseg,
                                    noverlap=noverlap,
                                    detrend=False)
    return freqs, times, np.log(spec.T.astype(np.float32) + eps)
```

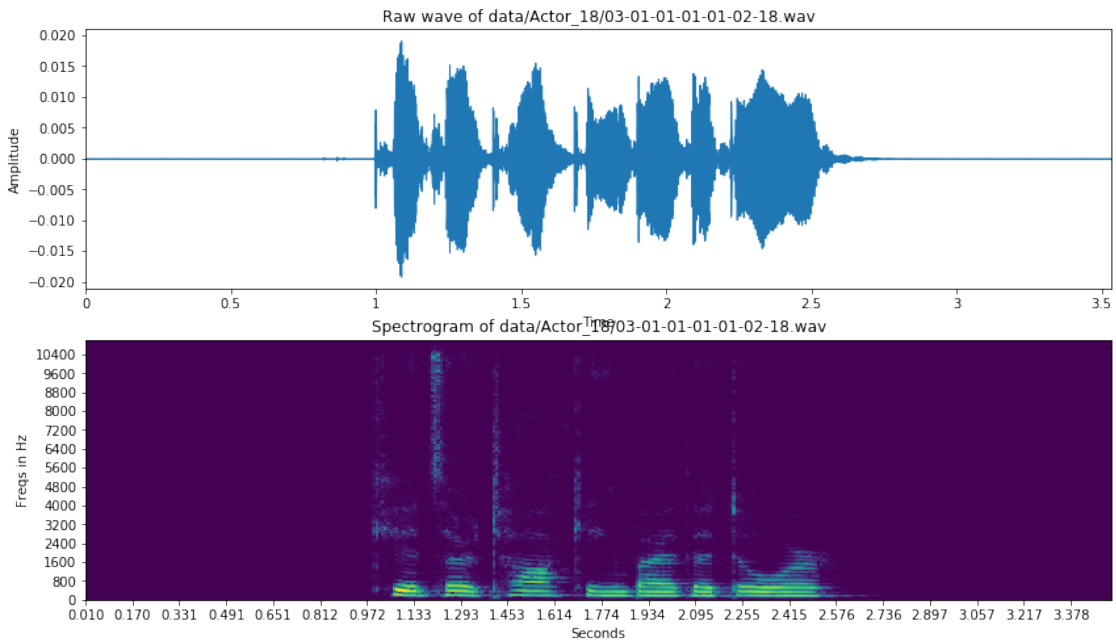
[8]:
```
sample_rate/ len(samples)
```

[8]:
```
0.28273218017925605
```

[9]:
```
# Plotting Wave Form and Spectrogram
freqs, times, spectrogram = log_specgram(samples, sample_rate)

fig = plt.figure(figsize=(14, 8))
ax1 = fig.add_subplot(211)
ax1.set_title('Raw wave of ' + filename)
ax1.set_ylabel('Amplitude')
librosa.display.waveplot(samples, sr=sample_rate)

ax2 = fig.add_subplot(212)
ax2.imshow(spectrogram.T, aspect='auto', origin='lower',
           extent=[times.min(), times.max(), freqs.min(), freqs.max()])
ax2.set_yticks(freqs[::16])
ax2.set_xticks(times[::16])
ax2.set_title('Spectrogram of ' + filename)
```

```
ax2.set_ylabel('Freqs in Hz')
ax2.set_xlabel('Seconds')
```

[9]: Text(0.5, 0, 'Seconds')



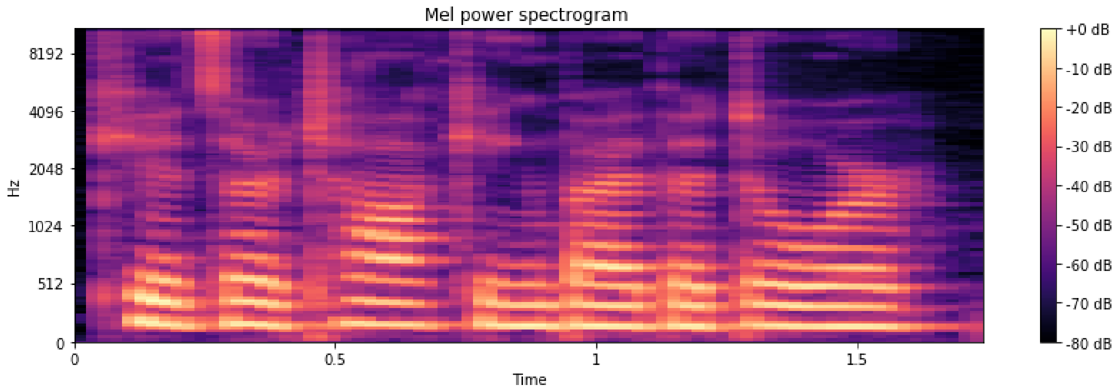
[10]:
```
mean = np.mean(spectrogram, axis=0)
std = np.std(spectrogram, axis=0)
spectrogram = (spectrogram - mean) / std
```

[11]:
```
# Trim the silence voice
aa , bb = librosa.effects.trim(samples, top_db=30)
aa, bb
```

[11]:
```
(array([-1.9141038e-07, -4.1607140e-07,  2.0688096e-06, …,
         5.6699279e-05,  2.1195672e-05,  3.1794041e-06], dtype=float32),
 array([20992, 58880]))
```

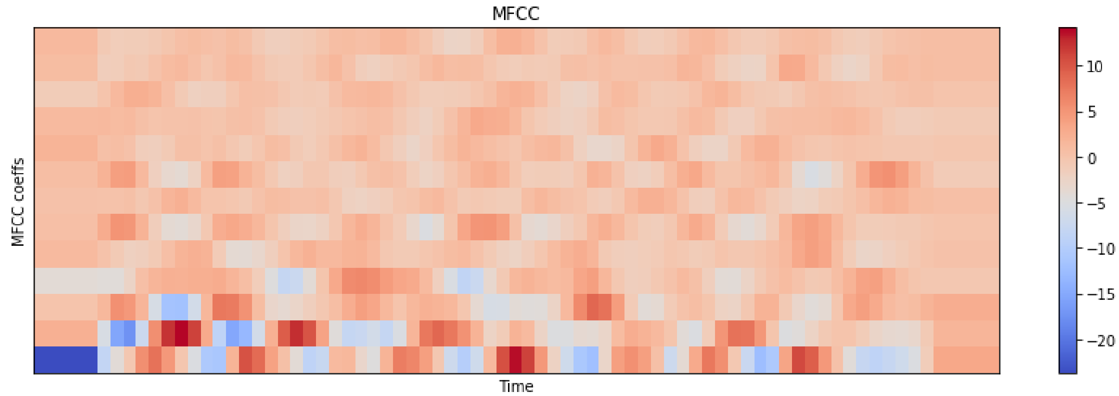
[12]:
```
# Plotting Mel Power Spectrogram
S = librosa.feature.melspectrogram(aa, sr=sample_rate, n_mels=128)

# Convert to log scale (dB). We'll use the peak power (max) as reference.
log_S = librosa.power_to_db(S, ref=np.max)

plt.figure(figsize=(12, 4))
librosa.display.specshow(log_S, sr=sample_rate, x_axis='time', y_axis='mel')
plt.title('Mel power spectrogram ')
```

```
plt.colorbar(format='%+02.0f dB')
plt.tight_layout()
```



[13]:
```
# Plotting MFCC
mfcc = librosa.feature.mfcc(S=log_S, n_mfcc=13)

# Let's pad on the first and second deltas while we're at it
delta2_mfcc = librosa.feature.delta(mfcc, order=2)

plt.figure(figsize=(12, 4))
librosa.display.specshow(delta2_mfcc)
plt.ylabel('MFCC coeffs')
plt.xlabel('Time')
plt.title('MFCC')
plt.colorbar()
plt.tight_layout()
```

```
C:\Users\jimmy\Anaconda3\lib\site-packages\scipy\signal\_arraytools.py:45:
FutureWarning:

Using a non-tuple sequence for multidimensional indexing is deprecated; use
`arr[tuple(seq)]` instead of `arr[seq]`. In the future this will be interpreted
as an array index, `arr[np.array(seq)]`, which will result either in an error or
a different result.
```



[14]:
```
# Original Sound
ipd.Audio(samples, rate=sample_rate)
```

[14]: <IPython.lib.display.Audio object>

[15]:
```
# Silence trimmed Sound by librosa.effects.trim()
ipd.Audio(aa, rate=sample_rate)
```

[15]: <IPython.lib.display.Audio object>

[16]:
```
# Silence trimmed Sound by manuel trimming
samples_cut = samples[10000:-12500]
ipd.Audio(samples_cut, rate=sample_rate)
```

[16]: <IPython.lib.display.Audio object>

4 IV. Defining the truth label

[17]:
```
# 2 class: Positive & Negative

# Positive: Calm, Happy
# Negative: Angry, Fearful, Sad

label2_list = []
for i in range(len(data_df)):
    if data_df.emotion[i] == 2: # Calm
        lb = "_positive"
    elif data_df.emotion[i] == 3: # Happy
        lb = "_positive"
    elif data_df.emotion[i] == 4: # Sad
        lb = "_negative"
    elif data_df.emotion[i] == 5: # Angry
```

```
        lb = "_negative"
    elif data_df.emotion[i] == 6: # Fearful
        lb = "_negative"
    else:
        lb = "_none"

    # Add gender to the label
    label2_list.append(data_df.gender[i] + lb)

len(label2_list)
```

[17]: 1440

[18]:
```
#3 class: Positive, Neutral & Negative

# Positive:  Happy
# Negative: Angry, Fearful, Sad
# Neutral: Calm, Neutral

label3_list = []
for i in range(len(data_df)):
    if data_df.emotion[i] == 1: # Neutral
        lb = "_neutral"
    elif data_df.emotion[i] == 2: # Calm
        lb = "_neutral"
    elif data_df.emotion[i] == 3: # Happy
        lb = "_positive"
    elif data_df.emotion[i] == 4: # Sad
        lb = "_negative"
    elif data_df.emotion[i] == 5: # Angry
        lb = "_negative"
    elif data_df.emotion[i] == 6: # Fearful
        lb = "_negative"
    else:
        lb = "_none"

    # Add gender to the label
    label3_list.append(data_df.gender[i] + lb)

len(label3_list)
```

[18]: 1440

[23]:
```
# 5 class: angry, calm, sad, happy & fearful
label5_list = []
for i in range(len(data_df)):
    if data_df.emotion[i] == 2:
```

```
        lb = "_calm"
    elif data_df.emotion[i] == 3:
        lb = "_happy"
    elif data_df.emotion[i] == 4:
        lb = "_sad"
    elif data_df.emotion[i] == 5:
        lb = "_angry"
    elif data_df.emotion[i] == 6:
        lb = "_fearful"
    else:
        lb = "_none"

    # Add gender to the label
    label5_list.append(data_df.gender[i] + lb)

len(label5_list)
```

[23]: 1440

[24]:
```
# All class

label8_list = []
for i in range(len(data_df)):
    if data_df.emotion[i] == 1:
        lb = "_neutral"
    elif data_df.emotion[i] == 2:
        lb = "_calm"
    elif data_df.emotion[i] == 3:
        lb = "_happy"
    elif data_df.emotion[i] == 4:
        lb = "_sad"
    elif data_df.emotion[i] == 5:
        lb = "_angry"
    elif data_df.emotion[i] == 6:
        lb = "_fearful"
    elif data_df.emotion[i] == 7:
        lb = "_disgust"
    elif data_df.emotion[i] == 8:
        lb = "_surprised"
    else:
        lb = "_none"

    # Add gender to the label
    label8_list.append(data_df.gender[i]  + lb)

len(label8_list)
```

[24]: 1440

[25]:
```
# Select the label set you want by commenting the unwanteds.

data_df['label'] = label2_list
# data_df['label'] = label3_list
# data_df['label'] = label5_list
# data_df['label'] = label8_list
data_df.head()
```

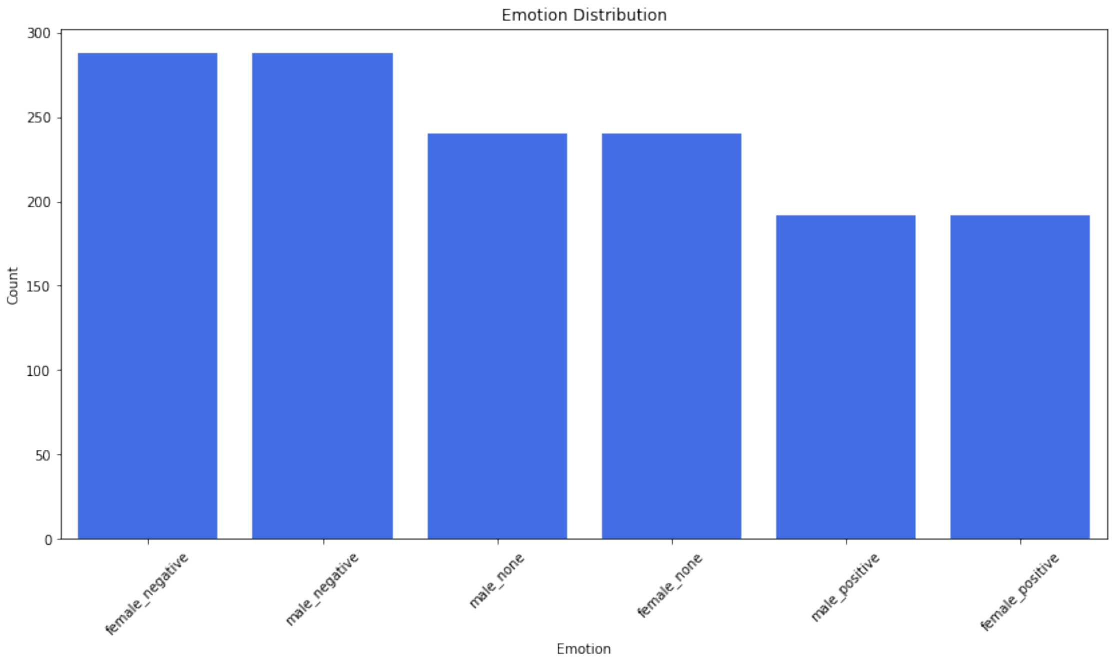
[25]:
```
                                     path source actor gender intensity  \
0  data/Actor_01/03-01-01-01-01-01-01.wav      1     1   male         0
1  data/Actor_01/03-01-01-01-01-02-01.wav      1     1   male         0
2  data/Actor_01/03-01-01-01-02-01-01.wav      1     1   male         0
3  data/Actor_01/03-01-01-01-02-02-01.wav      1     1   male         0
4  data/Actor_01/03-01-02-01-01-01-01.wav      1     1   male         0

  statement repetition emotion          label
0         0          0       1      male_none
1         0          1       1      male_none
2         1          0       1      male_none
3         1          1       1      male_none
4         0          0       2  male_positive
```

[26]:
```
print (data_df.label.value_counts().keys())
```

```
Index(['female_negative', 'male_negative', 'male_none', 'female_none',
       'male_positive', 'female_positive'],
      dtype='object')
```

[27]:
```
# Plotting the emotion distribution

def plot_emotion_dist(dist, color_code='#C2185B', title="Plot"):
    """
    To plot the data distributioin by class.
    Arg:
      dist: pandas series of label count.
    """
    tmp_df = pd.DataFrame()
    tmp_df['Emotion'] = list(dist.keys())
    tmp_df['Count'] = list(dist)
    fig, ax = plt.subplots(figsize=(14, 7))
    ax = sns.barplot(x="Emotion", y='Count', color=color_code, data=tmp_df)
    ax.set_title(title)
    ax.set_xticklabels(ax.get_xticklabels(),rotation=45)
```

[28]:
```
a = data_df.label.value_counts()
plot_emotion_dist(a, "#2962FF", "Emotion Distribution")
```



5 V. Data Splitting

[29]:
```
# Female Data Set

## Uncomment all below to use Female set

data2_df = data_df.copy()
data2_df = data2_df[data2_df.label != "male_none"]
data2_df = data2_df[data2_df.label != "female_none"]
data2_df = data2_df[data2_df.label != "male_happy"]
data2_df = data2_df[data2_df.label != "male_angry"]
data2_df = data2_df[data2_df.label != "male_sad"]
data2_df = data2_df[data2_df.label != "male_fearful"]
data2_df = data2_df[data2_df.label != "male_calm"]
data2_df = data2_df[data2_df.label != "male_positive"]
data2_df = data2_df[data2_df.label != "male_negative"].reset_index(drop=True)

tmp1 = data2_df[data2_df.actor == 22]
tmp2 = data2_df[data2_df.actor == 24]
data3_df = pd.concat([tmp1, tmp2],ignore_index=True).reset_index(drop=True)
data2_df = data2_df[data2_df.actor != 22]
```

```
data2_df = data2_df[data2_df.actor != 24].reset_index(drop=True)
print (len(data2_df))
data2_df.head()
```

400

[29]:

```
                                        path  source  actor  gender  intensity  \
0  data/Actor_02/03-01-02-01-01-01-02.wav       1      2  female          0
1  data/Actor_02/03-01-02-01-01-02-02.wav       1      2  female          0
2  data/Actor_02/03-01-02-01-02-01-02.wav       1      2  female          0
3  data/Actor_02/03-01-02-01-02-02-02.wav       1      2  female          0
4  data/Actor_02/03-01-02-02-01-01-02.wav       1      2  female          1

  statement repetition emotion            label
0         0          0       2  female_positive
1         0          1       2  female_positive
2         1          0       2  female_positive
3         1          1       2  female_positive
4         0          0       2  female_positive
```

[89]:

```
# Male Data Set

## Uncomment all below to use Male set

data2_df = data_df.copy()
data2_df = data2_df[data2_df.label != "male_none"]
data2_df = data2_df[data2_df.label != "female_none"].reset_index(drop=True)
data2_df = data2_df[data2_df.label != "female_neutral"]
data2_df = data2_df[data2_df.label != "female_happy"]
data2_df = data2_df[data2_df.label != "female_angry"]
data2_df = data2_df[data2_df.label != "female_sad"]
data2_df = data2_df[data2_df.label != "female_fearful"]
data2_df = data2_df[data2_df.label != "female_calm"]
data2_df = data2_df[data2_df.label != "female_positive"]
data2_df = data2_df[data2_df.label != "female_negative"].reset_index(drop=True)

tmp1 = data2_df[data2_df.actor == 21]
tmp2 = data2_df[data2_df.actor == 22]
tmp3 = data2_df[data2_df.actor == 23]
tmp4 = data2_df[data2_df.actor == 24]
data3_df = pd.concat([tmp1, tmp3],ignore_index=True).reset_index(drop=True)
data2_df = data2_df[data2_df.actor != 21]
data2_df = data2_df[data2_df.actor != 22]
data2_df = data2_df[data2_df.actor != 23].reset_index(drop=True)
data2_df = data2_df[data2_df.actor != 24].reset_index(drop=True)
print (len(data2_df))
data2_df.head()
```

```
400
```

[89]:

```
                                   path  source  actor gender  intensity  \
0  data/Actor_01/03-01-02-01-01-01-01.wav       1      1   male          0
1  data/Actor_01/03-01-02-01-01-02-01.wav       1      1   male          0
2  data/Actor_01/03-01-02-01-02-01-01.wav       1      1   male          0
3  data/Actor_01/03-01-02-01-02-02-01.wav       1      1   male          0
4  data/Actor_01/03-01-02-02-01-01-01.wav       1      1   male          1

   statement  repetition  emotion          label
0          0           0        2  male_positive
1          0           1        2  male_positive
2          1           0        2  male_positive
3          1           1        2  male_positive
4          0           0        2  male_positive
```

[90]:

```python
print (len(data3_df))
data3_df.head()
```

```
80
```

[90]:

```
                                   path  source  actor gender  intensity  \
0  data/Actor_21/03-01-02-01-01-01-21.wav       1     21   male          0
1  data/Actor_21/03-01-02-01-01-02-21.wav       1     21   male          0
2  data/Actor_21/03-01-02-01-02-01-21.wav       1     21   male          0
3  data/Actor_21/03-01-02-01-02-02-21.wav       1     21   male          0
4  data/Actor_21/03-01-02-02-01-01-21.wav       1     21   male          1

   statement  repetition  emotion          label
0          0           0        2  male_positive
1          0           1        2  male_positive
2          1           0        2  male_positive
3          1           1        2  male_positive
4          0           0        2  male_positive
```

6 VI. Getting the features of audio files using librosa

[91]:

```python
data = pd.DataFrame(columns=['feature'])
for i in tqdm(range(len(data2_df))):
    X, sample_rate = librosa.load(data2_df.path[i],␣
 ↪res_type='kaiser_fast',duration=input_duration,sr=22050*2,offset=0.5)
#     X = X[10000:90000]
    sample_rate = np.array(sample_rate)
    mfccs = np.mean(librosa.feature.mfcc(y=X, sr=sample_rate, n_mfcc=13),␣
 ↪axis=0)
    feature = mfccs
```

```
    data.loc[i] = [feature]
```

```
100%|      | 400/400 [00:24<00:00, 17.37it/s]
```

[92]:
```
data.head()
```

[92]:
```
                                             feature
0  [-70.2677641610773, -70.2677641610773, -70.267…
1  [-67.55739512198222, -67.55739512198222, -67.5…
2  [-69.67328949566406, -69.69331084873151, -69.6…
3  [-69.05139995492158, -69.05139995492158, -69.0…
4  [-73.8413701111492, -73.8413701111492, -73.841…
```

[93]:
```
df3 = pd.DataFrame(data['feature'].values.tolist())
labels = data2_df.label
```

[94]:
```
df3.head()
```

[94]:
```
           0          1          2          3          4          5    \
0 -70.267764 -70.267764 -70.267764 -70.267764 -70.267764 -70.267764
1 -67.557395 -67.557395 -67.557395 -67.557395 -67.557395 -67.557395
2 -69.673289 -69.693311 -69.693311 -69.693311 -69.693311 -69.693311
3 -69.051400 -69.051400 -69.051400 -69.051400 -69.051400 -68.754863
4 -73.841370 -73.841370 -73.841370 -73.719655 -73.841370 -73.841370

           6          7          8          9    …        249  \
0 -70.267764 -70.267764 -70.267764 -70.267764    …   -70.267764
1 -65.239801 -65.536197 -67.557395 -67.557395    …   -67.557395
2 -69.693311 -69.620774 -69.693311 -68.906572    …   -69.693311
3 -69.051400 -69.051400 -69.051400 -68.359101    …   -65.446950
4 -73.841370 -73.303635 -72.806811 -73.841370    …   -73.841370

         250        251        252        253        254        255  \
0 -70.267764 -69.957707 -68.377602 -69.862569 -70.267764 -70.122135
1 -67.557395 -67.557395 -67.557395 -67.557395 -67.557395 -67.557395
2 -69.693311 -69.693311 -69.693311 -69.693311 -69.383522 -69.693311
3 -68.552088 -69.051400 -69.051400 -69.051400 -68.688614 -69.051400
4 -73.841370 -73.841370 -73.841370 -73.841370 -73.841370 -73.841370

         256        257        258
0 -68.554960 -70.206530 -70.267764
1 -67.557395 -67.126574 -67.557395
2 -69.693311 -69.693311 -69.693311
3        NaN        NaN        NaN
4 -73.841370 -73.841370 -73.841370

[5 rows x 259 columns]
```

[95]:
```python
newdf = pd.concat([df3,labels], axis=1)
```

[96]:
```python
rnewdf = newdf.rename(index=str, columns={"0": "label"})
len(rnewdf)
```

[96]: 400

[97]:
```python
rnewdf.head(10)
```

[97]:
```
           0          1          2          3          4          5  \
0 -70.267764 -70.267764 -70.267764 -70.267764 -70.267764 -70.267764
1 -67.557395 -67.557395 -67.557395 -67.557395 -67.557395 -67.557395
2 -69.673289 -69.693311 -69.693311 -69.693311 -69.693311 -69.693311
3 -69.051400 -69.051400 -69.051400 -69.051400 -69.051400 -68.754863
4 -73.841370 -73.841370 -73.841370 -73.719655 -73.841370 -73.841370
5 -69.243253 -69.243253 -69.243253 -69.243253 -68.901972 -67.982999
6 -73.254968 -73.254968 -73.254968 -73.254968 -68.774422 -69.380388
7 -70.746514 -70.746514 -70.025286 -69.131263 -70.746514 -70.746514
8 -63.311078 -63.072484 -63.412433 -63.796762 -63.581991 -58.921211
9 -60.369038 -60.083715 -60.978925 -60.952456 -60.982486 -60.983948

           6          7          8          9      …          250  \
0 -70.267764 -70.267764 -70.267764 -70.267764      …   -70.267764
1 -65.239801 -65.536197 -67.557395 -67.557395      …   -67.557395
2 -69.693311 -69.620774 -69.693311 -68.906572      …   -69.693311
3 -69.051400 -69.051400 -69.051400 -68.359101      …   -68.552088
4 -73.841370 -73.303635 -72.806811 -73.841370      …   -73.841370
5 -68.089201 -67.897329 -65.258010 -67.170980      …   -57.185978
6 -73.254968 -73.254968 -73.254968 -73.254968      …   -50.884085
7 -70.746514 -70.746514 -70.746514 -70.746514      …   -70.746514
8 -57.955046 -61.224968 -63.782931 -63.796762      …   -63.740612
9 -60.981255 -60.981255 -60.981255 -60.249618      …   -60.981255

         251        252        253        254        255        256  \
0 -69.957707 -68.377602 -69.862569 -70.267764 -70.122135 -68.554960
1 -67.557395 -67.557395 -67.557395 -67.557395 -67.557395 -67.557395
2 -69.693311 -69.693311 -69.693311 -69.383522 -69.693311 -69.693311
3 -69.051400 -69.051400 -69.051400 -68.688614 -69.051400        NaN
4 -73.841370 -73.841370 -73.841370 -73.841370 -73.841370 -73.841370
5 -61.188731 -67.108389 -67.508122 -66.245553 -68.733048 -69.243253
6 -55.666730 -54.600013 -53.439110 -56.300120 -57.458272 -58.767075
7 -70.746514 -70.079249 -69.590462 -69.202740 -70.159467 -70.445363
8 -62.410257 -62.489080 -62.494456 -62.632636 -62.824277        NaN
9 -60.981255 -60.981255 -60.981255 -60.981255 -60.981255        NaN

         257        258          label
0 -70.206530 -70.267764  male_positive
```

```
1 -67.126574 -67.557395  male_positive
2 -69.693311 -69.693311  male_positive
3        NaN        NaN  male_positive
4 -73.841370 -73.841370  male_positive
5 -69.243253 -69.243253  male_positive
6 -59.836503 -58.409867  male_positive
7 -68.199043 -67.414208  male_positive
8        NaN        NaN  male_positive
9        NaN        NaN  male_positive

[10 rows x 260 columns]
```

[98]:
```
rnewdf.isnull().sum().sum()
```

[98]:
```
2284
```

[99]:
```
rnewdf = rnewdf.fillna(0)
rnewdf.head()
```

[99]:
```
          0          1          2          3          4          5  \
0 -70.267764 -70.267764 -70.267764 -70.267764 -70.267764 -70.267764
1 -67.557395 -67.557395 -67.557395 -67.557395 -67.557395 -67.557395
2 -69.673289 -69.693311 -69.693311 -69.693311 -69.693311 -69.693311
3 -69.051400 -69.051400 -69.051400 -69.051400 -69.051400 -68.754863
4 -73.841370 -73.841370 -73.841370 -73.719655 -73.841370 -73.841370

          6          7          8          9      …        250  \
0 -70.267764 -70.267764 -70.267764 -70.267764     …  -70.267764
1 -65.239801 -65.536197 -67.557395 -67.557395     …  -67.557395
2 -69.693311 -69.620774 -69.693311 -68.906572     …  -69.693311
3 -69.051400 -69.051400 -69.051400 -68.359101     …  -68.552088
4 -73.841370 -73.303635 -72.806811 -73.841370     …  -73.841370

        251        252        253        254        255        256  \
0 -69.957707 -68.377602 -69.862569 -70.267764 -70.122135 -68.554960
1 -67.557395 -67.557395 -67.557395 -67.557395 -67.557395 -67.557395
2 -69.693311 -69.693311 -69.693311 -69.383522 -69.693311 -69.693311
3 -69.051400 -69.051400 -69.051400 -68.688614 -69.051400   0.000000
4 -73.841370 -73.841370 -73.841370 -73.841370 -73.841370 -73.841370

        257        258          label
0 -70.206530 -70.267764  male_positive
1 -67.126574 -67.557395  male_positive
2 -69.693311 -69.693311  male_positive
3   0.000000   0.000000  male_positive
4 -73.841370 -73.841370  male_positive
```

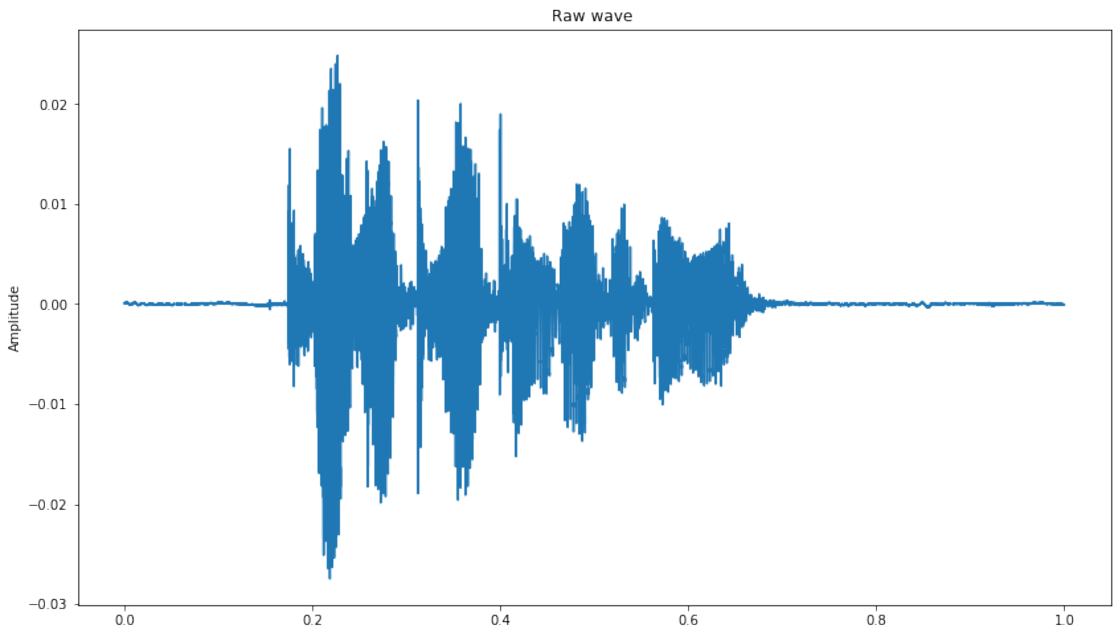
[5 rows x 260 columns]

7 VII. Data Augmentation

[40]:
```python
def plot_time_series(data):
    """
    Plot the Audio Frequency.
    """
    fig = plt.figure(figsize=(14, 8))
    plt.title('Raw wave ')
    plt.ylabel('Amplitude')
    plt.plot(np.linspace(0, 1, len(data)), data)
    plt.show()


def noise(data):
    """
    Adding White Noise.
    """
    # you can take any distribution from https://docs.scipy.org/doc/numpy-1.13.
↪0/reference/routines.random.html
    noise_amp = 0.005*np.random.uniform()*np.amax(data)
    data = data.astype('float64') + noise_amp * np.random.normal(size=data.
↪shape[0])
    return data

def shift(data):
    """
    Random Shifting.
    """
    s_range = int(np.random.uniform(low=-5, high = 5)*500)
    return np.roll(data, s_range)

def stretch(data, rate=0.8):
    """
    Streching the Sound.
    """
    data = librosa.effects.time_stretch(data, rate)
    return data

def pitch(data, sample_rate):
    """
    Pitch Tuning.
    """
    bins_per_octave = 12
    pitch_pm = 2
```

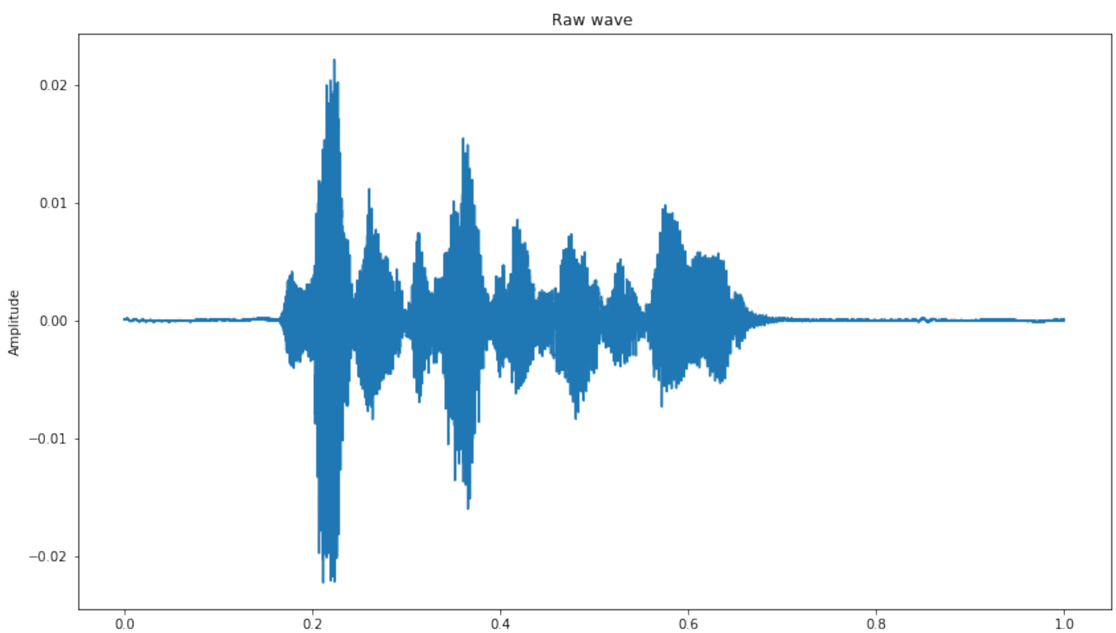
```
    pitch_change =  pitch_pm * 2*(np.random.uniform())
    data = librosa.effects.pitch_shift(data.astype('float64'),
                                      sample_rate, n_steps=pitch_change,
                                      bins_per_octave=bins_per_octave)
    return data

def dyn_change(data):
    """
    Random Value Change.
    """
    dyn_change = np.random.uniform(low=1.5,high=3)
    return (data * dyn_change)

def speedNpitch(data):
    """
    peed and Pitch Tuning.
    """
    # you can change low and high here
    length_change = np.random.uniform(low=0.8, high = 1)
    speed_fac = 1.0  / length_change
    tmp = np.interp(np.arange(0,len(data),speed_fac),np.
 ↪arange(0,len(data)),data)
    minlen = min(data.shape[0], tmp.shape[0])
    data *= 0
    data[0:minlen] = tmp[0:minlen]
    return data
```

[100]:
```
X, sample_rate = librosa.load(data2_df.path[216],␣
 ↪res_type='kaiser_fast',duration=4,sr=22050*2,offset=0.5)
plot_time_series(X)
ipd.Audio(X, rate=sample_rate)
```



[100]: <IPython.lib.display.Audio object>

[101]:
```
x = pitch(X, sample_rate)
plot_time_series(x)
ipd.Audio(x, rate=sample_rate)
```



[101]: <IPython.lib.display.Audio object>

[102]:
```
# Augmentation Method 1

syn_data1 = pd.DataFrame(columns=['feature', 'label'])
for i in tqdm(range(len(data2_df))):
    X, sample_rate = librosa.load(data2_df.path[i],␣
 ↪res_type='kaiser_fast',duration=input_duration,sr=22050*2,offset=0.5)
    if data2_df.label[i]:
#     if data2_df.label[i] == "male_positive":
        X = noise(X)
        sample_rate = np.array(sample_rate)
        mfccs = np.mean(librosa.feature.mfcc(y=X, sr=sample_rate, n_mfcc=13),␣
 ↪axis=0)
        feature = mfccs
        a = random.uniform(0, 1)
        syn_data1.loc[i] = [feature, data2_df.label[i]]
```

```
100%|          | 400/400 [00:25<00:00, 15.34it/s]
```

[103]:
```
# Augmentation Method 2

syn_data2 = pd.DataFrame(columns=['feature', 'label'])
for i in tqdm(range(len(data2_df))):
    X, sample_rate = librosa.load(data2_df.path[i],␣
 ↪res_type='kaiser_fast',duration=input_duration,sr=22050*2,offset=0.5)
    if data2_df.label[i]:
#     if data2_df.label[i] == "male_positive":
        X = pitch(X, sample_rate)
        sample_rate = np.array(sample_rate)
        mfccs = np.mean(librosa.feature.mfcc(y=X, sr=sample_rate, n_mfcc=13),␣
 ↪axis=0)
        feature = mfccs
        a = random.uniform(0, 1)
        syn_data2.loc[i] = [feature, data2_df.label[i]]
```

```
100%|          | 400/400 [01:35<00:00,  4.37it/s]
```

[104]:
```
len(syn_data1), len(syn_data2)
```

[104]: (400, 400)

[105]:
```
syn_data1 = syn_data1.reset_index(drop=True)
syn_data2 = syn_data2.reset_index(drop=True)
```

[106]:
```
df4 = pd.DataFrame(syn_data1['feature'].values.tolist())
labels4 = syn_data1.label
syndf1 = pd.concat([df4,labels4], axis=1)
syndf1 = syndf1.rename(index=str, columns={"0": "label"})
syndf1 = syndf1.fillna(0)
len(syndf1)
```

[106]:
```
400
```

[107]:
```
syndf1.head()
```

[107]:
```
           0          1          2          3          4          5  \
0 -60.182051 -58.117166 -58.369978 -57.745145 -56.251586 -56.962396
1 -67.683375 -67.296194 -66.229817 -66.965311 -66.143959 -65.626530
2 -54.236939 -54.261210 -56.668369 -57.761928 -57.187928 -58.614347
3 -54.730203 -54.156435 -54.802085 -52.028497 -52.576125 -53.800330
4 -56.734252 -56.783589 -57.226110 -57.258337 -55.681235 -52.502595

           6          7          8          9      …        250  \
0 -59.327271 -58.484776 -57.678923 -57.873383      …  -59.587469
1 -64.647545 -63.854617 -66.518488 -66.293804      …  -66.937795
2 -58.185813 -58.328112 -57.191179 -55.921943      …  -57.200956
3 -51.643633 -51.629818 -52.309757 -51.927829      …  -50.962930
4 -54.307751 -55.374062 -56.117597 -59.825737      …  -57.145136

         251        252        253        254        255        256  \
0 -59.988579 -58.260270 -56.493974 -55.495719 -57.909146 -59.712433
1 -67.204075 -67.434871 -67.094666 -66.087321 -65.232261 -66.541181
2 -56.756710 -57.503389 -57.787787 -54.886599 -53.712429 -56.648883
3 -53.486098 -52.132062 -51.323519 -50.473240 -50.679438   0.000000
4 -58.761933 -57.046106 -57.392087 -60.048381 -59.130028 -56.845604

         257        258          label
0 -57.845471 -58.981049  male_positive
1 -66.935943 -65.994166  male_positive
2 -59.739390 -60.516950  male_positive
3   0.000000   0.000000  male_positive
4 -55.362638 -58.250624  male_positive

[5 rows x 260 columns]
```

[108]:
```
df4 = pd.DataFrame(syn_data2['feature'].values.tolist())
labels4 = syn_data2.label
syndf2 = pd.concat([df4,labels4], axis=1)
syndf2 = syndf2.rename(index=str, columns={"0": "label"})
syndf2 = syndf2.fillna(0)
len(syndf2)
```

[108]: 400

[109]:
```
syndf2.head()
```

[109]:
```
           0          1          2          3          4          5  \
0 -70.726569 -70.726569 -70.726569 -70.726569 -70.726569 -70.726569
1 -69.984891 -69.984891 -69.904577 -69.984891 -69.984891 -69.770850
2 -71.181455 -71.483948 -71.762365 -71.762365 -71.762365 -71.762365
3 -70.053410 -70.053410 -70.053410 -70.053410 -70.053410 -70.053410
4 -75.954847 -75.954847 -75.954847 -75.954847 -75.954847 -75.954847

           6          7          8          9  …        250  \
0 -70.726569 -70.726569 -70.726569 -70.726569  …  -70.726569
1 -67.413591 -69.383499 -69.984891 -69.984891  …  -69.984891
2 -71.762365 -71.762365 -71.659095 -70.900970  …  -71.762365
3 -70.053410 -70.053410 -70.053410 -69.935976  …  -69.582117
4 -75.954847 -75.631421 -74.847477 -75.954847  …  -75.954847

         251        252        253        254        255        256  \
0 -70.076103 -70.597671 -70.726569 -70.726569 -70.661296 -70.526060
1 -69.984891 -69.984891 -69.984891 -69.984891 -69.984891 -69.984891
2 -71.762365 -71.762365 -71.762365 -71.720913 -71.762365 -71.762365
3 -70.053410 -70.053410 -70.053410 -70.053410 -70.053410   0.000000
4 -75.954847 -75.954847 -75.954847 -75.954847 -75.954847 -75.954847

         257        258          label
0 -70.726569 -70.726569  male_positive
1 -69.984891 -69.984891  male_positive
2 -71.762365 -71.762365  male_positive
3   0.000000   0.000000  male_positive
4 -75.954847 -75.954847  male_positive

[5 rows x 260 columns]
```

[110]:
```
# Combining the Augmented data with original
combined_df = pd.concat([rnewdf, syndf1, syndf2], ignore_index=True)
combined_df = combined_df.fillna(0)
combined_df.head()
```

[110]:
```
           0          1          2          3          4          5  \
0 -70.267764 -70.267764 -70.267764 -70.267764 -70.267764 -70.267764
1 -67.557395 -67.557395 -67.557395 -67.557395 -67.557395 -67.557395
2 -69.673289 -69.693311 -69.693311 -69.693311 -69.693311 -69.693311
3 -69.051400 -69.051400 -69.051400 -69.051400 -69.051400 -68.754863
4 -73.841370 -73.841370 -73.841370 -73.719655 -73.841370 -73.841370

           6          7          8          9  …        250  \
```

```
0 -70.267764 -70.267764 -70.267764 -70.267764      …       -70.267764
1 -65.239801 -65.536197 -67.557395 -67.557395      …       -67.557395
2 -69.693311 -69.620774 -69.693311 -68.906572      …       -69.693311
3 -69.051400 -69.051400 -69.051400 -68.359101      …       -68.552088
4 -73.841370 -73.303635 -72.806811 -73.841370      …       -73.841370

         251        252        253        254        255        256  \
0 -69.957707 -68.377602 -69.862569 -70.267764 -70.122135 -68.554960
1 -67.557395 -67.557395 -67.557395 -67.557395 -67.557395 -67.557395
2 -69.693311 -69.693311 -69.693311 -69.383522 -69.693311 -69.693311
3 -69.051400 -69.051400 -69.051400 -68.688614 -69.051400   0.000000
4 -73.841370 -73.841370 -73.841370 -73.841370 -73.841370 -73.841370

         257        258          label
0 -70.206530 -70.267764  male_positive
1 -67.126574 -67.557395  male_positive
2 -69.693311 -69.693311  male_positive
3   0.000000   0.000000  male_positive
4 -73.841370 -73.841370  male_positive

[5 rows x 260 columns]
```

[111]:
```
#  Stratified Shuffle Split
from sklearn.model_selection import StratifiedShuffleSplit
X = combined_df.drop(['label'], axis=1)
y = combined_df.label
xxx = StratifiedShuffleSplit(1, test_size=0.2, random_state=12)
for train_index, test_index in xxx.split(X, y):
    X_train, X_test = X.iloc[train_index], X.iloc[test_index]
    y_train, y_test = y.iloc[train_index], y.iloc[test_index]
```

[112]:
```
y_train.value_counts()
```

[112]:
```
male_negative    576
male_positive    384
Name: label, dtype: int64
```

[113]:
```
y_test.value_counts()
```

[113]:
```
male_negative    144
male_positive     96
Name: label, dtype: int64
```

[114]:
```
X_train.isna().sum().sum()
```

[114]:
```
0
```

[115]:
```
X_train = np.array(X_train)
y_train = np.array(y_train)
X_test = np.array(X_test)
y_test = np.array(y_test)
lb = LabelEncoder()
y_train = np_utils.to_categorical(lb.fit_transform(y_train))
y_test = np_utils.to_categorical(lb.fit_transform(y_test))
```

[116]:
```
y_train
```

[116]:
```
array([[1., 0.],
       [1., 0.],
       [0., 1.],
       …,
       [1., 0.],
       [1., 0.],
       [0., 1.]], dtype=float32)
```

[117]:
```
X_train.shape
```

[117]:
```
(960, 259)
```

8 VIII. Changing dimension for CNN model

[118]:
```
x_traincnn = np.expand_dims(X_train, axis=2)
x_testcnn = np.expand_dims(X_test, axis=2)
```

[60]:
```
# Set up Keras util functions

from keras import backend as K

def precision(y_true, y_pred):
    true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
    predicted_positives = K.sum(K.round(K.clip(y_pred, 0, 1)))
    precision = true_positives / (predicted_positives + K.epsilon())
    return precision


def recall(y_true, y_pred):
    true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
    possible_positives = K.sum(K.round(K.clip(y_true, 0, 1)))
    recall = true_positives / (possible_positives + K.epsilon())
    return recall


def fscore(y_true, y_pred):
```

```python
    if K.sum(K.round(K.clip(y_true, 0, 1))) == 0:
        return 0

    p = precision(y_true, y_pred)
    r = recall(y_true, y_pred)
    f_score = 2 * (p * r) / (p + r + K.epsilon())
    return f_score

def get_lr_metric(optimizer):
    def lr(y_true, y_pred):
        return optimizer.lr
    return lr
```

[61]:
```python
# New model
model = Sequential()
model.add(Conv1D(256, 8, padding='same',input_shape=(X_train.shape[1],1)))
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())
# Edit according to target class no.
model.add(Dense(2))
model.add(Activation('softmax'))
# opt = keras.optimizers.SGD(lr=0.0001, momentum=0.0, decay=0.0, nesterov=False)
```

[66]:
```python
# Original Model

# model = Sequential()
# model.add(Conv1D(256, 5,padding='same', input_shape=(X_train.shape[1],1)))
```

```
# model.add(Activation('relu'))
# model.add(Conv1D(128, 5,padding='same'))
# model.add(Activation('relu'))
# model.add(Dropout(0.1))
# model.add(MaxPooling1D(pool_size=(8)))
# model.add(Conv1D(128, 5,padding='same',))
# model.add(Activation('relu'))
# model.add(Conv1D(128, 5,padding='same',))
# model.add(Activation('relu'))
# model.add(Conv1D(128, 5,padding='same',))
# model.add(Activation('relu'))
# model.add(Dropout(0.2))
# model.add(Conv1D(128, 5,padding='same',))
# model.add(Activation('relu'))
# model.add(Flatten())
# model.add(Dense(5))
# model.add(Activation('softmax'))
# opt = keras.optimizers.rmsprop(lr=0.00001, decay=1e-6)
```

[119]:

```
# Plotting Model Summary

model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d_1 (Conv1D)            (None, 259, 256)          2304
_________________________________________________________________
activation_1 (Activation)    (None, 259, 256)          0
_________________________________________________________________
conv1d_2 (Conv1D)            (None, 259, 256)          524544
_________________________________________________________________
batch_normalization_1 (Batch (None, 259, 256)          1024
_________________________________________________________________
activation_2 (Activation)    (None, 259, 256)          0
_________________________________________________________________
dropout_1 (Dropout)          (None, 259, 256)          0
_________________________________________________________________
max_pooling1d_1 (MaxPooling1 (None, 32, 256)           0
_________________________________________________________________
conv1d_3 (Conv1D)            (None, 32, 128)           262272
_________________________________________________________________
activation_3 (Activation)    (None, 32, 128)           0
_________________________________________________________________
conv1d_4 (Conv1D)            (None, 32, 128)           131200
_________________________________________________________________
```

```
activation_4 (Activation)    (None, 32, 128)          0
_________________________________________________________________
conv1d_5 (Conv1D)            (None, 32, 128)          131200
_________________________________________________________________
activation_5 (Activation)    (None, 32, 128)          0
_________________________________________________________________
conv1d_6 (Conv1D)            (None, 32, 128)          131200
_________________________________________________________________
batch_normalization_2 (Batch (None, 32, 128)          512
_________________________________________________________________
activation_6 (Activation)    (None, 32, 128)          0
_________________________________________________________________
dropout_2 (Dropout)          (None, 32, 128)          0
_________________________________________________________________
max_pooling1d_2 (MaxPooling1 (None, 4, 128)           0
_________________________________________________________________
conv1d_7 (Conv1D)            (None, 4, 64)            65600
_________________________________________________________________
activation_7 (Activation)    (None, 4, 64)            0
_________________________________________________________________
conv1d_8 (Conv1D)            (None, 4, 64)            32832
_________________________________________________________________
activation_8 (Activation)    (None, 4, 64)            0
_________________________________________________________________
flatten_1 (Flatten)          (None, 256)              0
_________________________________________________________________
dense_1 (Dense)              (None, 2)                514
_________________________________________________________________
activation_9 (Activation)    (None, 2)                0
=================================================================
Total params: 1,283,202
Trainable params: 1,282,434
Non-trainable params: 768
_________________________________________________________________
```

[120]:

```
# Compile your model

model.compile(loss='categorical_crossentropy', optimizer="adamax",␣
↪metrics=['accuracy'])
```

9 IX. Removed the whole training part for avoiding unnecessary long epochs list

[125]:
```
# Model Training

lr_reduce = ReduceLROnPlateau(monitor='val_loss', factor=0.9, patience=20,␣
↪min_lr=0.000001)
# Please change the model name accordingly.
mcp_save = ModelCheckpoint('model/aug_noiseNshift_2class2_np.h5',␣
↪save_best_only=True, monitor='val_loss', mode='min')
cnnhistory=model.fit(x_traincnn, y_train, batch_size=100, epochs=30,
                     validation_data=(x_testcnn, y_test), callbacks=[mcp_save,␣
↪lr_reduce])
```

```
Train on 960 samples, validate on 240 samples
Epoch 1/30
960/960 [==============================] - ETA: 13s - loss: 0.1441 - accuracy:
0.950 - ETA: 12s - loss: 0.1306 - accuracy: 0.960 - ETA: 10s - loss: 0.1174 -
accuracy: 0.963 - ETA: 8s - loss: 0.1449 - accuracy: 0.945 - ETA: 6s - loss:
0.1490 - accuracy: 0.94 - ETA: 5s - loss: 0.1427 - accuracy: 0.94 - ETA: 3s -
loss: 0.1474 - accuracy: 0.94 - ETA: 2s - loss: 0.1425 - accuracy: 0.94 - ETA:
0s - loss: 0.1502 - accuracy: 0.94 - 15s 16ms/step - loss: 0.1477 - accuracy:
0.9479 - val_loss: 2.6242 - val_accuracy: 0.6083
Epoch 2/30
960/960 [==============================] - ETA: 12s - loss: 0.1349 - accuracy:
0.930 - ETA: 11s - loss: 0.1243 - accuracy: 0.940 - ETA: 9s - loss: 0.1378 -
accuracy: 0.943 - ETA: 8s - loss: 0.1345 - accuracy: 0.94 - ETA: 6s - loss:
0.1261 - accuracy: 0.95 - ETA: 5s - loss: 0.1202 - accuracy: 0.95 - ETA: 3s -
loss: 0.1165 - accuracy: 0.95 - ETA: 2s - loss: 0.1198 - accuracy: 0.95 - ETA:
0s - loss: 0.1191 - accuracy: 0.95 - 15s 16ms/step - loss: 0.1254 - accuracy:
0.9521 - val_loss: 1.6341 - val_accuracy: 0.6292
Epoch 3/30
960/960 [==============================] - ETA: 13s - loss: 0.1208 - accuracy:
0.970 - ETA: 11s - loss: 0.1108 - accuracy: 0.970 - ETA: 10s - loss: 0.1004 -
accuracy: 0.976 - ETA: 8s - loss: 0.1256 - accuracy: 0.967 - ETA: 7s - loss:
0.1295 - accuracy: 0.96 - ETA: 5s - loss: 0.1205 - accuracy: 0.96 - ETA: 4s -
loss: 0.1231 - accuracy: 0.95 - ETA: 2s - loss: 0.1328 - accuracy: 0.95 - ETA:
0s - loss: 0.1296 - accuracy: 0.95 - 17s 17ms/step - loss: 0.1285 - accuracy:
0.9573 - val_loss: 0.8549 - val_accuracy: 0.6167
Epoch 4/30
960/960 [==============================] - ETA: 14s - loss: 0.1463 - accuracy:
0.940 - ETA: 12s - loss: 0.1302 - accuracy: 0.950 - ETA: 10s - loss: 0.1657 -
accuracy: 0.933 - ETA: 9s - loss: 0.1573 - accuracy: 0.935 - ETA: 7s - loss:
0.1432 - accuracy: 0.94 - ETA: 6s - loss: 0.1487 - accuracy: 0.94 - ETA: 4s -
loss: 0.1520 - accuracy: 0.94 - ETA: 2s - loss: 0.1425 - accuracy: 0.94 - ETA:
1s - loss: 0.1364 - accuracy: 0.94 - 18s 18ms/step - loss: 0.1326 - accuracy:
0.9490 - val_loss: 0.8676 - val_accuracy: 0.6792
```

```
Epoch 5/30
960/960 [==============================] - ETA: 14s - loss: 0.1612 - accuracy:
0.910 - ETA: 12s - loss: 0.1151 - accuracy: 0.945 - ETA: 10s - loss: 0.1230 -
accuracy: 0.943 - ETA: 9s - loss: 0.1165 - accuracy: 0.947 - ETA: 7s - loss:
0.1129 - accuracy: 0.95 - ETA: 5s - loss: 0.1258 - accuracy: 0.94 - ETA: 4s -
loss: 0.1157 - accuracy: 0.95 - ETA: 2s - loss: 0.1122 - accuracy: 0.95 - ETA:
0s - loss: 0.1118 - accuracy: 0.95 - 17s 17ms/step - loss: 0.1197 - accuracy:
0.9490 - val_loss: 0.5873 - val_accuracy: 0.7333
Epoch 6/30
960/960 [==============================] - ETA: 13s - loss: 0.0875 - accuracy:
0.980 - ETA: 11s - loss: 0.0904 - accuracy: 0.975 - ETA: 10s - loss: 0.0886 -
accuracy: 0.976 - ETA: 8s - loss: 0.0912 - accuracy: 0.972 - ETA: 7s - loss:
0.0947 - accuracy: 0.96 - ETA: 5s - loss: 0.0977 - accuracy: 0.96 - ETA: 4s -
loss: 0.0911 - accuracy: 0.96 - ETA: 2s - loss: 0.0909 - accuracy: 0.96 - ETA:
0s - loss: 0.0903 - accuracy: 0.96 - 16s 17ms/step - loss: 0.0907 - accuracy:
0.9635 - val_loss: 1.2904 - val_accuracy: 0.6708
Epoch 7/30
960/960 [==============================] - ETA: 14s - loss: 0.1049 - accuracy:
0.940 - ETA: 12s - loss: 0.0975 - accuracy: 0.945 - ETA: 10s - loss: 0.1015 -
accuracy: 0.950 - ETA: 8s - loss: 0.0902 - accuracy: 0.960 - ETA: 7s - loss:
0.0783 - accuracy: 0.96 - ETA: 5s - loss: 0.0788 - accuracy: 0.96 - ETA: 4s -
loss: 0.0774 - accuracy: 0.97 - ETA: 2s - loss: 0.0769 - accuracy: 0.97 - ETA:
0s - loss: 0.0759 - accuracy: 0.97 - 16s 17ms/step - loss: 0.0788 - accuracy:
0.9698 - val_loss: 0.7540 - val_accuracy: 0.6833
Epoch 8/30
960/960 [==============================] - ETA: 13s - loss: 0.0537 - accuracy:
0.980 - ETA: 11s - loss: 0.0708 - accuracy: 0.980 - ETA: 10s - loss: 0.0830 -
accuracy: 0.966 - ETA: 8s - loss: 0.0817 - accuracy: 0.970 - ETA: 7s - loss:
0.0770 - accuracy: 0.97 - ETA: 5s - loss: 0.0753 - accuracy: 0.97 - ETA: 4s -
loss: 0.0801 - accuracy: 0.97 - ETA: 2s - loss: 0.0830 - accuracy: 0.97 - ETA:
0s - loss: 0.0809 - accuracy: 0.97 - 17s 17ms/step - loss: 0.0821 - accuracy:
0.9688 - val_loss: 1.6167 - val_accuracy: 0.6458
Epoch 9/30
960/960 [==============================] - ETA: 16s - loss: 0.0686 - accuracy:
0.980 - ETA: 13s - loss: 0.1596 - accuracy: 0.920 - ETA: 11s - loss: 0.1191 -
accuracy: 0.946 - ETA: 9s - loss: 0.1150 - accuracy: 0.950 - ETA: 7s - loss:
0.1180 - accuracy: 0.95 - ETA: 6s - loss: 0.1274 - accuracy: 0.94 - ETA: 4s -
loss: 0.1383 - accuracy: 0.94 - ETA: 2s - loss: 0.1386 - accuracy: 0.94 - ETA:
1s - loss: 0.1313 - accuracy: 0.94 - 17s 18ms/step - loss: 0.1278 - accuracy:
0.9479 - val_loss: 1.1404 - val_accuracy: 0.6708
Epoch 10/30
960/960 [==============================] - ETA: 13s - loss: 0.1519 - accuracy:
0.910 - ETA: 11s - loss: 0.1064 - accuracy: 0.950 - ETA: 10s - loss: 0.0915 -
accuracy: 0.956 - ETA: 8s - loss: 0.1124 - accuracy: 0.950 - ETA: 7s - loss:
0.1058 - accuracy: 0.95 - ETA: 5s - loss: 0.0947 - accuracy: 0.96 - ETA: 4s -
loss: 0.0894 - accuracy: 0.96 - ETA: 2s - loss: 0.0920 - accuracy: 0.96 - ETA:
0s - loss: 0.0895 - accuracy: 0.96 - 16s 17ms/step - loss: 0.0870 - accuracy:
0.9698 - val_loss: 0.8235 - val_accuracy: 0.6833
```

```
Epoch 11/30
960/960 [==============================] - ETA: 13s - loss: 0.0681 - accuracy:
0.970 - ETA: 11s - loss: 0.0608 - accuracy: 0.975 - ETA: 10s - loss: 0.0705 -
accuracy: 0.966 - ETA: 9s - loss: 0.0692 - accuracy: 0.967 - ETA: 7s - loss:
0.0771 - accuracy: 0.96 - ETA: 5s - loss: 0.0732 - accuracy: 0.96 - ETA: 4s -
loss: 0.0679 - accuracy: 0.97 - ETA: 2s - loss: 0.0623 - accuracy: 0.97 - ETA:
0s - loss: 0.0627 - accuracy: 0.97 - 17s 17ms/step - loss: 0.0624 - accuracy:
0.9740 - val_loss: 0.5985 - val_accuracy: 0.7417
Epoch 12/30
960/960 [==============================] - ETA: 13s - loss: 0.0871 - accuracy:
0.970 - ETA: 12s - loss: 0.0583 - accuracy: 0.985 - ETA: 10s - loss: 0.0531 -
accuracy: 0.990 - ETA: 9s - loss: 0.0519 - accuracy: 0.990 - ETA: 8s - loss:
0.0476 - accuracy: 0.99 - ETA: 6s - loss: 0.0468 - accuracy: 0.99 - ETA: 5s -
loss: 0.0431 - accuracy: 0.99 - ETA: 3s - loss: 0.0431 - accuracy: 0.99 - ETA:
1s - loss: 0.0417 - accuracy: 0.99 - 20s 21ms/step - loss: 0.0427 - accuracy:
0.9906 - val_loss: 0.6889 - val_accuracy: 0.6833
Epoch 13/30
960/960 [==============================] - ETA: 17s - loss: 0.0541 - accuracy:
0.980 - ETA: 16s - loss: 0.0329 - accuracy: 0.990 - ETA: 14s - loss: 0.0354 -
accuracy: 0.986 - ETA: 12s - loss: 0.0325 - accuracy: 0.990 - ETA: 9s - loss:
0.0303 - accuracy: 0.992 - ETA: 7s - loss: 0.0272 - accuracy: 0.99 - ETA: 5s -
loss: 0.0267 - accuracy: 0.99 - ETA: 3s - loss: 0.0268 - accuracy: 0.99 - ETA:
1s - loss: 0.0263 - accuracy: 0.99 - 19s 20ms/step - loss: 0.0265 - accuracy:
0.9937 - val_loss: 0.5456 - val_accuracy: 0.7542
Epoch 14/30
960/960 [==============================] - ETA: 14s - loss: 0.0110 - accuracy:
1.000 - ETA: 12s - loss: 0.0142 - accuracy: 1.000 - ETA: 10s - loss: 0.0201 -
accuracy: 0.993 - ETA: 9s - loss: 0.0243 - accuracy: 0.990 - ETA: 7s - loss:
0.0207 - accuracy: 0.99 - ETA: 5s - loss: 0.0191 - accuracy: 0.99 - ETA: 4s -
loss: 0.0183 - accuracy: 0.99 - ETA: 2s - loss: 0.0175 - accuracy: 0.99 - ETA:
1s - loss: 0.0169 - accuracy: 0.99 - 17s 18ms/step - loss: 0.0168 - accuracy:
0.9958 - val_loss: 0.6294 - val_accuracy: 0.7083
Epoch 15/30
960/960 [==============================] - ETA: 15s - loss: 0.0095 - accuracy:
1.000 - ETA: 13s - loss: 0.0090 - accuracy: 1.000 - ETA: 11s - loss: 0.0094 -
accuracy: 1.000 - ETA: 9s - loss: 0.0090 - accuracy: 1.000 - ETA: 7s - loss:
0.0109 - accuracy: 1.00 - ETA: 5s - loss: 0.0105 - accuracy: 1.00 - ETA: 4s -
loss: 0.0126 - accuracy: 1.00 - ETA: 2s - loss: 0.0139 - accuracy: 0.99 - ETA:
1s - loss: 0.0129 - accuracy: 0.99 - 17s 18ms/step - loss: 0.0127 - accuracy:
0.9990 - val_loss: 0.5994 - val_accuracy: 0.7042
Epoch 16/30
960/960 [==============================] - ETA: 15s - loss: 0.0183 - accuracy:
0.990 - ETA: 12s - loss: 0.0145 - accuracy: 0.995 - ETA: 11s - loss: 0.0179 -
accuracy: 0.993 - ETA: 9s - loss: 0.0160 - accuracy: 0.995 - ETA: 7s - loss:
0.0151 - accuracy: 0.99 - ETA: 5s - loss: 0.0145 - accuracy: 0.99 - ETA: 4s -
loss: 0.0152 - accuracy: 0.99 - ETA: 2s - loss: 0.0136 - accuracy: 0.99 - ETA:
0s - loss: 0.0167 - accuracy: 0.99 - 17s 18ms/step - loss: 0.0178 - accuracy:
0.9958 - val_loss: 0.5366 - val_accuracy: 0.7708
```

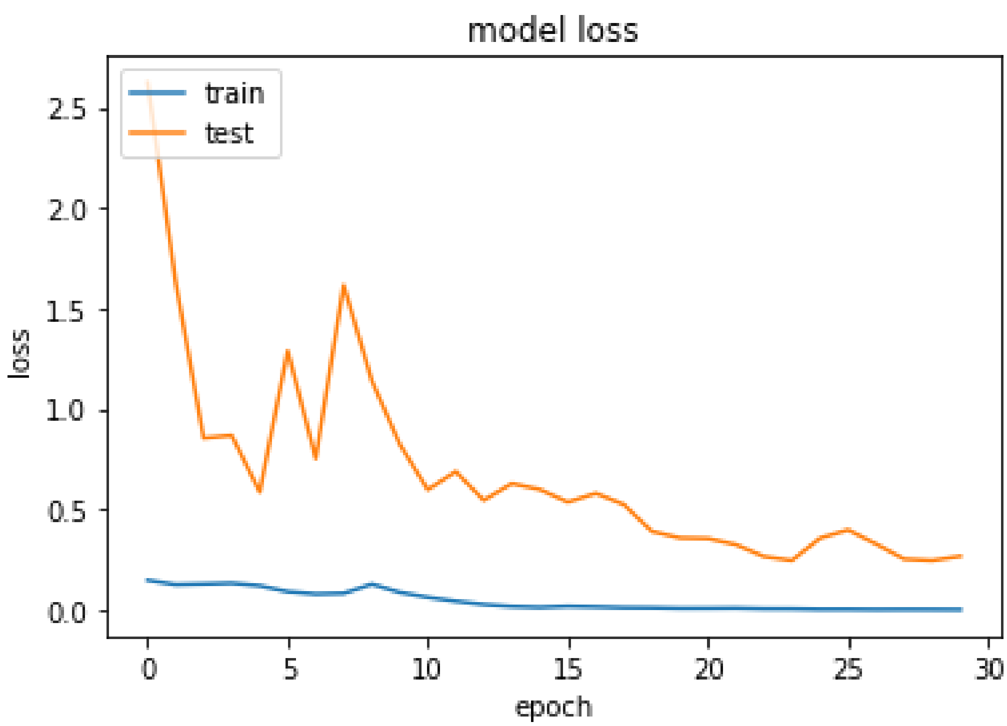
```
Epoch 17/30
960/960 [==============================] - ETA: 14s - loss: 0.0171 - accuracy:
1.000 - ETA: 13s - loss: 0.0148 - accuracy: 1.000 - ETA: 11s - loss: 0.0126 -
accuracy: 1.000 - ETA: 9s - loss: 0.0132 - accuracy: 1.000 - ETA: 7s - loss:
0.0150 - accuracy: 0.99 - ETA: 6s - loss: 0.0144 - accuracy: 0.99 - ETA: 4s -
loss: 0.0138 - accuracy: 0.99 - ETA: 2s - loss: 0.0136 - accuracy: 0.99 - ETA:
1s - loss: 0.0144 - accuracy: 0.99 - 17s 18ms/step - loss: 0.0148 - accuracy:
0.9979 - val_loss: 0.5805 - val_accuracy: 0.7792
Epoch 18/30
960/960 [==============================] - ETA: 14s - loss: 0.0198 - accuracy:
0.990 - ETA: 12s - loss: 0.0117 - accuracy: 0.995 - ETA: 10s - loss: 0.0093 -
accuracy: 0.996 - ETA: 8s - loss: 0.0115 - accuracy: 0.995 - ETA: 7s - loss:
0.0109 - accuracy: 0.99 - ETA: 5s - loss: 0.0153 - accuracy: 0.99 - ETA: 4s -
loss: 0.0137 - accuracy: 0.99 - ETA: 2s - loss: 0.0125 - accuracy: 0.99 - ETA:
0s - loss: 0.0116 - accuracy: 0.99 - 17s 17ms/step - loss: 0.0111 - accuracy:
0.9958 - val_loss: 0.5244 - val_accuracy: 0.7667
Epoch 19/30
960/960 [==============================] - ETA: 13s - loss: 0.0106 - accuracy:
1.000 - ETA: 12s - loss: 0.0091 - accuracy: 1.000 - ETA: 10s - loss: 0.0080 -
accuracy: 1.000 - ETA: 8s - loss: 0.0073 - accuracy: 1.000 - ETA: 7s - loss:
0.0136 - accuracy: 0.99 - ETA: 5s - loss: 0.0131 - accuracy: 0.99 - ETA: 4s -
loss: 0.0117 - accuracy: 0.99 - ETA: 2s - loss: 0.0114 - accuracy: 0.99 - ETA:
0s - loss: 0.0107 - accuracy: 0.99 - 17s 17ms/step - loss: 0.0103 - accuracy:
0.9979 - val_loss: 0.3908 - val_accuracy: 0.8250
Epoch 20/30
960/960 [==============================] - ETA: 13s - loss: 0.0107 - accuracy:
1.000 - ETA: 11s - loss: 0.0084 - accuracy: 1.000 - ETA: 10s - loss: 0.0081 -
accuracy: 1.000 - ETA: 8s - loss: 0.0081 - accuracy: 1.000 - ETA: 7s - loss:
0.0074 - accuracy: 1.00 - ETA: 5s - loss: 0.0068 - accuracy: 1.00 - ETA: 4s -
loss: 0.0074 - accuracy: 1.00 - ETA: 2s - loss: 0.0072 - accuracy: 1.00 - ETA:
0s - loss: 0.0074 - accuracy: 1.00 - 17s 17ms/step - loss: 0.0075 - accuracy:
1.0000 - val_loss: 0.3581 - val_accuracy: 0.8625
Epoch 21/30
960/960 [==============================] - ETA: 13s - loss: 0.0126 - accuracy:
0.990 - ETA: 12s - loss: 0.0070 - accuracy: 0.995 - ETA: 10s - loss: 0.0079 -
accuracy: 0.996 - ETA: 8s - loss: 0.0076 - accuracy: 0.997 - ETA: 7s - loss:
0.0081 - accuracy: 0.99 - ETA: 5s - loss: 0.0076 - accuracy: 0.99 - ETA: 4s -
loss: 0.0079 - accuracy: 0.99 - ETA: 2s - loss: 0.0077 - accuracy: 0.99 - ETA:
0s - loss: 0.0078 - accuracy: 0.99 - 17s 17ms/step - loss: 0.0076 - accuracy:
0.9990 - val_loss: 0.3554 - val_accuracy: 0.8500
Epoch 22/30
960/960 [==============================] - ETA: 13s - loss: 0.0030 - accuracy:
1.000 - ETA: 12s - loss: 0.0080 - accuracy: 0.995 - ETA: 10s - loss: 0.0078 -
accuracy: 0.996 - ETA: 8s - loss: 0.0080 - accuracy: 0.997 - ETA: 7s - loss:
0.0069 - accuracy: 0.99 - ETA: 5s - loss: 0.0090 - accuracy: 0.99 - ETA: 4s -
loss: 0.0086 - accuracy: 0.99 - ETA: 2s - loss: 0.0088 - accuracy: 0.99 - ETA:
0s - loss: 0.0085 - accuracy: 0.99 - 17s 17ms/step - loss: 0.0085 - accuracy:
0.9969 - val_loss: 0.3248 - val_accuracy: 0.8625
```

```
Epoch 23/30
960/960 [==============================] - ETA: 13s - loss: 0.0047 - accuracy:
1.000 - ETA: 12s - loss: 0.0033 - accuracy: 1.000 - ETA: 10s - loss: 0.0048 -
accuracy: 1.000 - ETA: 8s - loss: 0.0044 - accuracy: 1.000 - ETA: 7s - loss:
0.0042 - accuracy: 1.00 - ETA: 5s - loss: 0.0039 - accuracy: 1.00 - ETA: 4s -
loss: 0.0048 - accuracy: 1.00 - ETA: 2s - loss: 0.0048 - accuracy: 1.00 - ETA:
0s - loss: 0.0051 - accuracy: 1.00 - 17s 17ms/step - loss: 0.0054 - accuracy:
1.0000 - val_loss: 0.2656 - val_accuracy: 0.9000
Epoch 24/30
960/960 [==============================] - ETA: 13s - loss: 0.0028 - accuracy:
1.000 - ETA: 12s - loss: 0.0049 - accuracy: 1.000 - ETA: 10s - loss: 0.0047 -
accuracy: 1.000 - ETA: 9s - loss: 0.0072 - accuracy: 0.997 - ETA: 7s - loss:
0.0064 - accuracy: 0.99 - ETA: 5s - loss: 0.0060 - accuracy: 0.99 - ETA: 4s -
loss: 0.0059 - accuracy: 0.99 - ETA: 2s - loss: 0.0056 - accuracy: 0.99 - ETA:
0s - loss: 0.0052 - accuracy: 0.99 - 17s 17ms/step - loss: 0.0051 - accuracy:
0.9990 - val_loss: 0.2461 - val_accuracy: 0.9000
Epoch 25/30
960/960 [==============================] - ETA: 14s - loss: 0.0022 - accuracy:
1.000 - ETA: 12s - loss: 0.0019 - accuracy: 1.000 - ETA: 10s - loss: 0.0022 -
accuracy: 1.000 - ETA: 9s - loss: 0.0021 - accuracy: 1.000 - ETA: 7s - loss:
0.0022 - accuracy: 1.00 - ETA: 5s - loss: 0.0020 - accuracy: 1.00 - ETA: 4s -
loss: 0.0025 - accuracy: 1.00 - ETA: 2s - loss: 0.0026 - accuracy: 1.00 - ETA:
0s - loss: 0.0025 - accuracy: 1.00 - 17s 17ms/step - loss: 0.0026 - accuracy:
1.0000 - val_loss: 0.3576 - val_accuracy: 0.8625
Epoch 26/30
960/960 [==============================] - ETA: 13s - loss: 0.0018 - accuracy:
1.000 - ETA: 12s - loss: 0.0030 - accuracy: 1.000 - ETA: 10s - loss: 0.0036 -
accuracy: 1.000 - ETA: 9s - loss: 0.0029 - accuracy: 1.000 - ETA: 7s - loss:
0.0033 - accuracy: 1.00 - ETA: 6s - loss: 0.0034 - accuracy: 1.00 - ETA: 4s -
loss: 0.0031 - accuracy: 1.00 - ETA: 2s - loss: 0.0029 - accuracy: 1.00 - ETA:
1s - loss: 0.0027 - accuracy: 1.00 - 17s 18ms/step - loss: 0.0026 - accuracy:
1.0000 - val_loss: 0.3984 - val_accuracy: 0.8667
Epoch 27/30
960/960 [==============================] - ETA: 14s - loss: 0.0019 - accuracy:
1.000 - ETA: 13s - loss: 0.0014 - accuracy: 1.000 - ETA: 11s - loss: 0.0013 -
accuracy: 1.000 - ETA: 9s - loss: 0.0016 - accuracy: 1.000 - ETA: 7s - loss:
0.0017 - accuracy: 1.00 - ETA: 6s - loss: 0.0017 - accuracy: 1.00 - ETA: 4s -
loss: 0.0018 - accuracy: 1.00 - ETA: 2s - loss: 0.0016 - accuracy: 1.00 - ETA:
0s - loss: 0.0015 - accuracy: 1.00 - 17s 17ms/step - loss: 0.0015 - accuracy:
1.0000 - val_loss: 0.3280 - val_accuracy: 0.8625
Epoch 28/30
960/960 [==============================] - ETA: 13s - loss: 0.0015 - accuracy:
1.000 - ETA: 12s - loss: 0.0013 - accuracy: 1.000 - ETA: 11s - loss: 0.0016 -
accuracy: 1.000 - ETA: 9s - loss: 0.0024 - accuracy: 1.000 - ETA: 7s - loss:
0.0021 - accuracy: 1.00 - ETA: 5s - loss: 0.0022 - accuracy: 1.00 - ETA: 4s -
loss: 0.0021 - accuracy: 1.00 - ETA: 2s - loss: 0.0020 - accuracy: 1.00 - ETA:
0s - loss: 0.0019 - accuracy: 1.00 - 17s 17ms/step - loss: 0.0018 - accuracy:
1.0000 - val_loss: 0.2531 - val_accuracy: 0.9000
```

```
Epoch 29/30
960/960 [==============================] - ETA: 13s - loss: 6.1667e-04 -
accuracy: 1.000 - ETA: 13s - loss: 9.4631e-04 - accuracy: 1.000 - ETA: 11s -
loss: 0.0020 - accuracy: 1.0000    - ETA: 9s - loss: 0.0019 - accuracy: 1.000 -
ETA: 7s - loss: 0.0019 - accuracy: 1.00 - ETA: 5s - loss: 0.0020 - accuracy:
1.00 - ETA: 4s - loss: 0.0018 - accuracy: 1.00 - ETA: 2s - loss: 0.0016 -
accuracy: 1.00 - ETA: 0s - loss: 0.0016 - accuracy: 1.00 - 17s 17ms/step - loss:
0.0016 - accuracy: 1.0000 - val_loss: 0.2459 - val_accuracy: 0.9042
Epoch 30/30
960/960 [==============================] - ETA: 14s - loss: 0.0011 - accuracy:
1.000 - ETA: 13s - loss: 0.0011 - accuracy: 1.000 - ETA: 11s - loss: 0.0010 -
accuracy: 1.000 - ETA: 9s - loss: 9.9900e-04 - accuracy: 1.00 - ETA: 7s - loss:
9.1569e-04 - accuracy: 1.00 - ETA: 6s - loss: 8.7917e-04 - accuracy: 1.00 - ETA:
4s - loss: 9.8908e-04 - accuracy: 1.00 - ETA: 2s - loss: 9.0562e-04 - accuracy:
1.00 - ETA: 0s - loss: 9.0765e-04 - accuracy: 1.00 - 17s 17ms/step - loss:
8.9991e-04 - accuracy: 1.0000 - val_loss: 0.2669 - val_accuracy: 0.8833
```

[126]:

```python
# Plotting the Train Valid Loss Graph

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



9.1 Saving the model

[127]:
```
# Saving the model.json

import json
model_json = model.to_json()
with open("model.json", "w") as json_file:
    json_file.write(model_json)
```

9.2 Loading the model

[128]:
```
# loading json and creating model
from keras.models import model_from_json
json_file = open('model.json', 'r')
loaded_model_json = json_file.read()
json_file.close()
loaded_model = model_from_json(loaded_model_json)

# load weights into new model
loaded_model.load_weights("model/aug_noiseNshift_2class2_np.h5")
print("Loaded model from disk")

# evaluate loaded model on test data
loaded_model.compile(loss='categorical_crossentropy', optimizer="adamax",␣
↪metrics=['accuracy'])
score = loaded_model.evaluate(x_testcnn, y_test, verbose=0)
print("%s: %.2f%%" % (loaded_model.metrics_names[1], score[1]*100))
```

```
Loaded model from disk
accuracy: 90.42%
```

10 X. Predicting emotions on the test data

[130]:
```
len(data3_df)
```

[130]: 80

[131]:
```
data_test = pd.DataFrame(columns=['feature'])
for i in tqdm(range(len(data3_df))):
    X, sample_rate = librosa.load(data3_df.path[i],␣
↪res_type='kaiser_fast',duration=input_duration,sr=22050*2,offset=0.5)
#     X = X[10000:90000]
    sample_rate = np.array(sample_rate)
```

```
    mfccs = np.mean(librosa.feature.mfcc(y=X, sr=sample_rate, n_mfcc=13),␣
↪axis=0)
    feature = mfccs
    data_test.loc[i] = [feature]

test_valid = pd.DataFrame(data_test['feature'].values.tolist())
test_valid = np.array(test_valid)
test_valid_lb = np.array(data3_df.label)
lb = LabelEncoder()
test_valid_lb = np_utils.to_categorical(lb.fit_transform(test_valid_lb))
test_valid = np.expand_dims(test_valid, axis=2)
```

```
100%|      | 80/80 [00:04<00:00, 17.78it/s]
```

[132]:
```
preds = loaded_model.predict(test_valid,
                         batch_size=16,
                         verbose=1)
```

```
80/80 [==============================] - ETA:  - ETA:  - ETA:  - ETA:  - 0s
5ms/step
```

[133]:
```
preds
```

[133]:
```
array([[9.86237884e-01, 1.37620717e-02],
       [8.64436865e-01, 1.35563090e-01],
       [6.14066422e-01, 3.85933518e-01],
       [6.80679440e-01, 3.19320589e-01],
       [9.99952435e-01, 4.76056848e-05],
       [3.59568442e-03, 9.96404290e-01],
       [4.65872288e-01, 5.34127712e-01],
       [6.06496572e-01, 3.93503398e-01],
       [3.82461622e-02, 9.61753786e-01],
       [4.94476736e-01, 5.05523205e-01],
       [1.72606837e-02, 9.82739329e-01],
       [3.60655218e-01, 6.39344811e-01],
       [2.94867009e-01, 7.05133021e-01],
       [1.45178825e-01, 8.54821146e-01],
       [9.94913220e-01, 5.08677494e-03],
       [7.00582802e-01, 2.99417228e-01],
       [2.77864543e-04, 9.99722064e-01],
       [3.80067853e-04, 9.99619961e-01],
       [2.65352690e-04, 9.99734581e-01],
       [1.48483246e-04, 9.99851465e-01],
       [1.40601798e-04, 9.99859333e-01],
       [1.71381063e-04, 9.99828577e-01],
       [1.81893259e-03, 9.98181105e-01],
       [1.03868544e-03, 9.98961329e-01],
```

```
[1.21962130e-02, 9.87803817e-01],
[4.62811626e-03, 9.95371878e-01],
[9.99538183e-01, 4.61807213e-04],
[9.99999762e-01, 2.52335326e-07],
[5.42745650e-01, 4.57254320e-01],
[7.40037382e-01, 2.59962589e-01],
[9.99976993e-01, 2.30475507e-05],
[9.99990821e-01, 9.16626232e-06],
[9.97676075e-01, 2.32395506e-03],
[2.11535752e-01, 7.88464189e-01],
[2.24326533e-04, 9.99775708e-01],
[6.95544258e-02, 9.30445611e-01],
[8.08761895e-01, 1.91238061e-01],
[9.99070704e-01, 9.29322094e-04],
[9.99999285e-01, 6.74338025e-07],
[9.95826125e-01, 4.17389534e-03],
[6.20119041e-04, 9.99379873e-01],
[1.95969536e-04, 9.99804080e-01],
[5.83443216e-05, 9.99941707e-01],
[4.66044206e-04, 9.99534011e-01],
[2.21244171e-01, 7.78755784e-01],
[1.28578525e-02, 9.87142205e-01],
[3.99795041e-04, 9.99600232e-01],
[7.94181880e-03, 9.92058218e-01],
[           nan,            nan],
[           nan,            nan],
[           nan,            nan],
[           nan,            nan],
[           nan,            nan],
[           nan,            nan],
[           nan,            nan],
[           nan,            nan],
[2.71601379e-01, 7.28398621e-01],
[7.64008641e-01, 2.35991359e-01],
[6.63855433e-01, 3.36144567e-01],
[7.05953985e-02, 9.29404616e-01],
[8.75997663e-01, 1.24002293e-01],
[7.94358365e-03, 9.92056429e-01],
[4.61770535e-01, 5.38229465e-01],
[2.06720099e-04, 9.99793351e-01],
[           nan,            nan],
[           nan,            nan],
[           nan,            nan],
[           nan,            nan],
[           nan,            nan],
[           nan,            nan],
[           nan,            nan],
```

```
       [           nan,            nan],
       [           nan,            nan],
       [           nan,            nan],
       [           nan,            nan],
       [           nan,            nan],
       [9.95565951e-01, 4.43403888e-03],
       [           nan,            nan],
       [8.80912185e-01, 1.19087823e-01],
       [9.85459447e-02, 1.45405317e-02]], dtype=float32)
```

[134]:
```
preds1=preds.argmax(axis=1)
```

[135]:
```
preds1
```

[135]:
```
array([0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1,
       1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], dtype=int64)
```

[136]:
```
abc = preds1.astype(int).flatten()
```

[137]:
```
predictions = (lb.inverse_transform((abc)))
```

[138]:
```
preddf = pd.DataFrame({'predictedvalues': predictions})
preddf[:10]
```

[138]:

	predictedvalues
0	male_negative
1	male_negative
2	male_negative
3	male_negative
4	male_negative
5	male_positive
6	male_positive
7	male_negative
8	male_positive
9	male_positive

[139]:
```
actual=test_valid_lb.argmax(axis=1)
abc123 = actual.astype(int).flatten()
actualvalues = (lb.inverse_transform((abc123)))
```

[140]:
```
actualdf = pd.DataFrame({'actualvalues': actualvalues})
actualdf[:10]
```

[140]:

	actualvalues
0	male_positive

```
1  male_positive
2  male_positive
3  male_positive
4  male_positive
5  male_positive
6  male_positive
7  male_positive
8  male_positive
9  male_positive
```

[141]:
```
finaldf = actualdf.join(preddf)
```

10.1 Actual v/s Predicted emotions

[142]:
```
finaldf[20:40]
```

[142]:
```
     actualvalues predictedvalues
20  male_negative   male_positive
21  male_negative   male_positive
22  male_negative   male_positive
23  male_negative   male_positive
24  male_negative   male_positive
25  male_negative   male_positive
26  male_negative   male_negative
27  male_negative   male_negative
28  male_negative   male_negative
29  male_negative   male_negative
30  male_negative   male_negative
31  male_negative   male_negative
32  male_negative   male_negative
33  male_negative   male_positive
34  male_negative   male_positive
35  male_negative   male_positive
36  male_negative   male_negative
37  male_negative   male_negative
38  male_negative   male_negative
39  male_negative   male_negative
```

[143]:
```
finaldf.groupby('actualvalues').count()
```

[143]:
```
               predictedvalues
actualvalues
male_negative               48
male_positive               32
```

[144]:
```
finaldf.groupby('predictedvalues').count()
```

```
[144]:                    actualvalues
predictedvalues
male_negative                   46
male_positive                   34
```

```
[145]: finaldf.to_csv('Predictions.csv', index=False)
```

```
[84]: def print_confusion_matrix(confusion_matrix, class_names, figsize = (10,7),␣
    ↪fontsize=14):
        """Prints a confusion matrix, as returned by sklearn.metrics.
    ↪confusion_matrix, as a heatmap.

        Arguments
        ---------
        confusion_matrix: numpy.ndarray
            The numpy.ndarray object returned from a call to sklearn.metrics.
    ↪confusion_matrix.
            Similarly constructed ndarrays can also be used.
        class_names: list
            An ordered list of class names, in the order they index the given␣
    ↪confusion matrix.
        figsize: tuple
            A 2-long tuple, the first value determining the horizontal size of the␣
    ↪ouputted figure,
            the second determining the vertical size. Defaults to (10,7).
        fontsize: int
            Font size for axes labels. Defaults to 14.

        Returns
        -------
        matplotlib.figure.Figure
            The resulting confusion matrix figure
        """
        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')
```

[146]:
```
from sklearn.metrics import accuracy_score
y_true = finaldf.actualvalues
y_pred = finaldf.predictedvalues
accuracy_score(y_true, y_pred)*100
```

[146]: 57.49999999999999

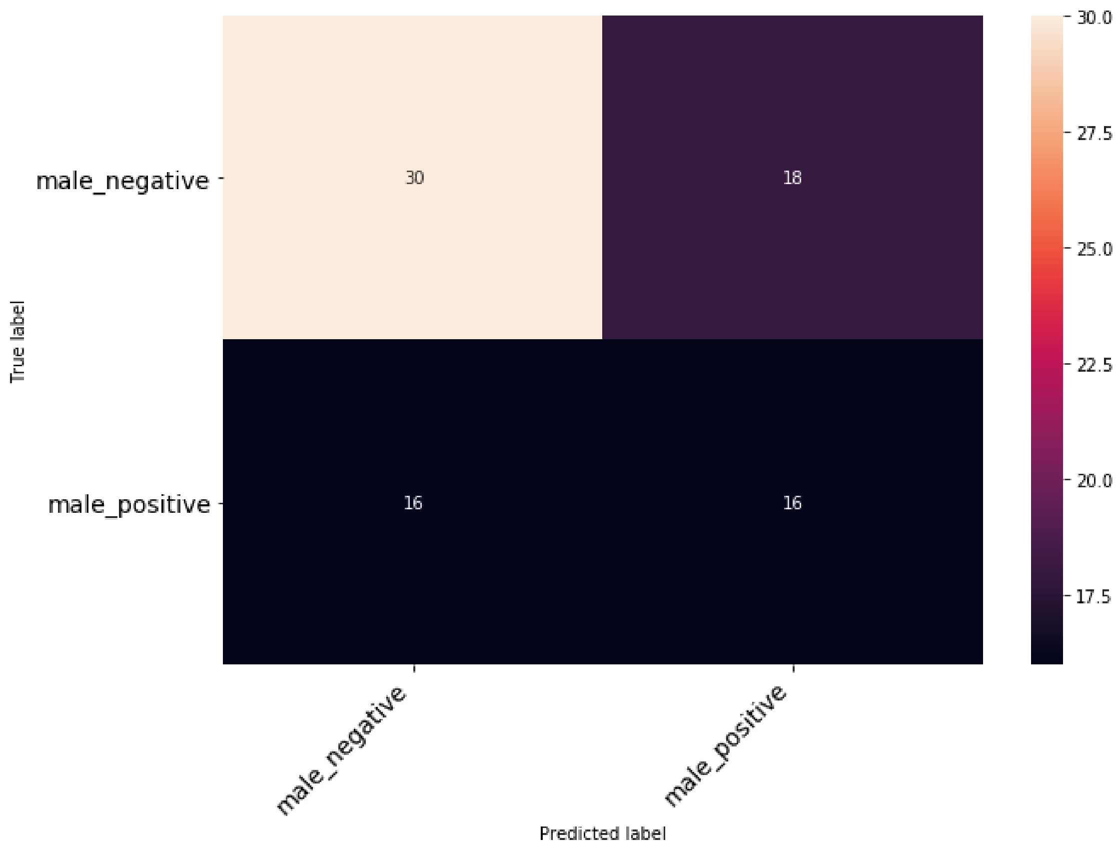
[147]:
```
from sklearn.metrics import f1_score
f1_score(y_true, y_pred, average='macro') *100
```

[147]: 56.15731785944552

[148]:
```
from sklearn.metrics import confusion_matrix
c = confusion_matrix(y_true, y_pred)
c
```

[148]:
```
array([[30, 18],
       [16, 16]], dtype=int64)
```

[149]:
```
# Visualize Confusion Matrix

# class_names = ['male_angry', 'male_calm', 'male_fearful', 'male_happy',␣
 ↪'male_sad']
# class_names = ['female_angry', 'female_calm', 'female_fearful',␣
 ↪'female_happy', 'female_sad']
# class_names = ['male_negative', 'male_neutral', 'male_positive']
class_names = ['male_negative', 'male_positive']
# class_names = ['female_negative', 'female_positive']
# class_names = ['female_angry', 'female_calm', 'female_fearful',␣
 ↪'female_happy', 'female_sad', 'male_angry', 'male_calm', 'male_fearful',␣
 ↪'male_happy', 'male_sad']


print_confusion_matrix(c, class_names)
```


male_negative
male_positive
True label
30
18
16
16
male_negative
male_positive
Predicted label
30.0
27.5
25.0
22.5
20.0
17.5

```
[ ]:
```