PROJECT PRESENTATION

Audio Mining

EMOTION DETECTION & SENTIMENT ANALYSIS SYSTEM

GROUP MARYGOLD: DARPAN M, TEJAL R, SHUBHAM P, ANIRUDH S

CS6220: DATA MINING TECHNIQUES

PROF.SARA ARUNAGIRI

Table of CONTENT

PROJECT PRESENTATION

PROJECT PRESENTATION

Project INTRODUCTION

In the current world, free speech exists in various forms including textual as well as audio. Extracting and analyzing sentiments from these audio data is the need of the hour. Understanding the customer's feelings will improve the performance of the product

AUDIO SENTIMENT

TEXT SENTIMENT

PROJECT PRESENTATION

Why THIS PROJECT ?

VISION

There has been a surge in content creation in forms of podcasts, audiobooks, videos and more. Extracting and analyzing sentiments from these audio data is the need of the hour.

ENHANCE CONTENT QUALITY

MISSION

Developing a model to analyse sentiments of audio files, and validate them using text sentiment analysis on converted audio files using Natural Language Processing

SENTIMENT ANALYSIS

PROJECT PRESENTATION

Data SOURCE

480

Audios

7

Emotions

2

Methods

24

Voice Overs

THE EMOTIONS ARE:
NEUTRAL ANGRY HAPPY SAD FEAR DISGUST SURPRISE
SAVEE DATASET

Dataset source link: http://kahlan.eps.surrey.ac.uk/savee/Database.html

METHODOLGY

PART 1

AUDIO SENTIMENT ANALYSIS

Here we analyzed the sentiments of audio files using Keras and Convolutional Neural Network.

PART 2

AUDIO TO TEXT CONVERSION

Speech recognition module and recognize_google() method are used to transcribe our audio files.

PART 3

TEXT SENTIMENT ANALYSIS

This part functions as analyzing the audio files converted to text using Bidirectional LSTM.

PROCEDURE

DATA PREPROCESSING

CLEANING AND DATA PATH

Cleaning dataset, and linking the data paths to implement audio analysis

EDA

SPECTROGRAMS AND PLOTS

Plotting spectrograms to check disparities and plotting them against various emotions

MODEL

CNN AND LSTM

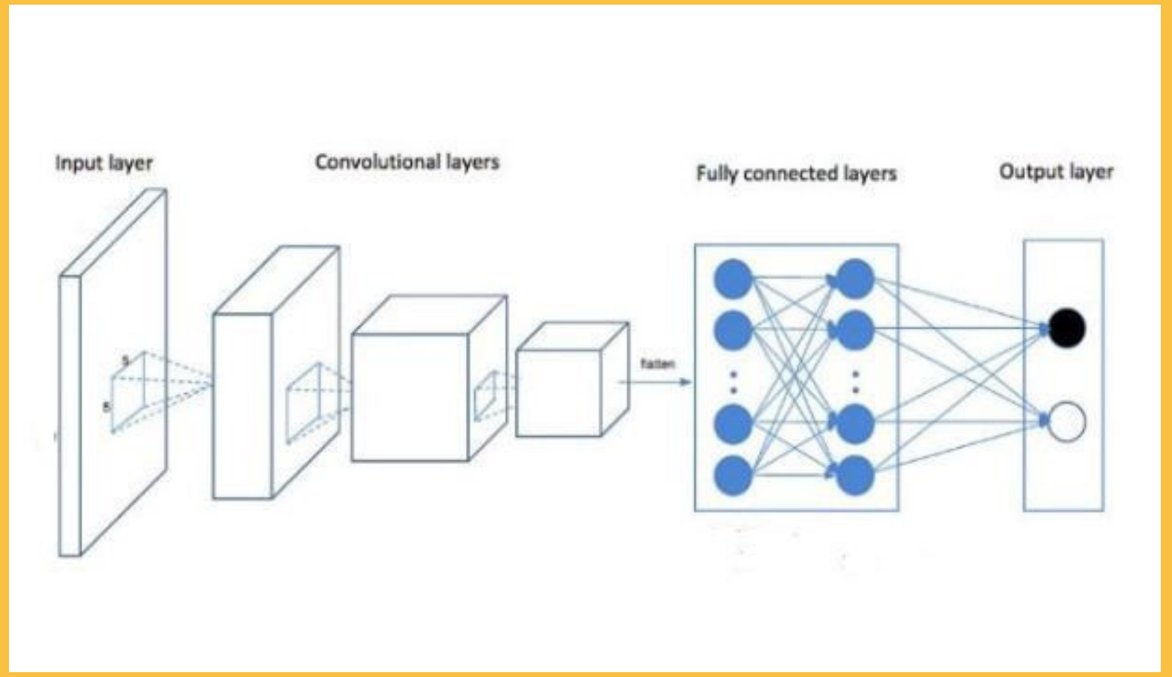
Passing the audio files and text files in the CNN and LSTM Models respectively

PROJECT PRESENTATION

Audio

SENTIMENT ANALYSIS

This part functions as analysing the sentiments of audio using Keras and Convolutional Neural Network



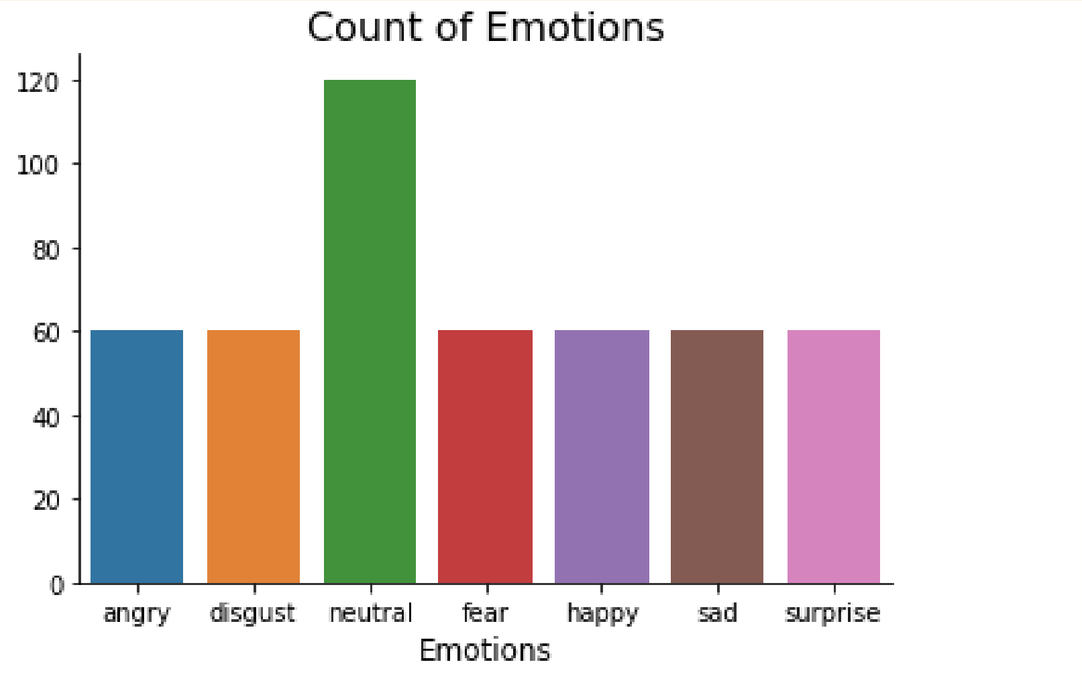
- KERAS
- CONVOLUTIONAL NN

CODE FOR DATA PREPROCESSING

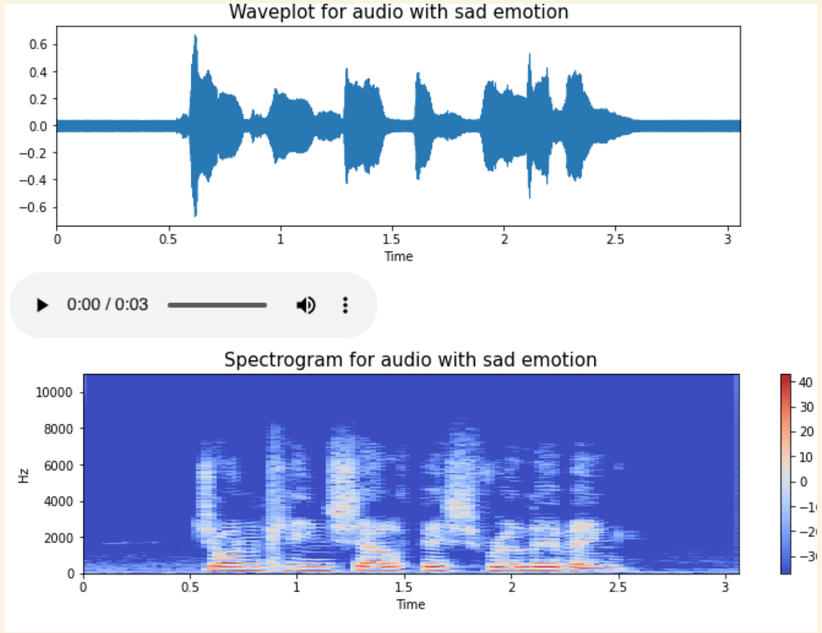
```
#map required to identify type of emotion from the file name
#example : DC_a01.wav => a indicates angry
emotion_map = {
    'h': 'happy',
    'sa': 'sad',
    'n': 'neutral',
    'a': 'angry',
    'd': 'disgust',
    'f': 'fear',
    'su': 'surprise'
}

#read sound files from the directory
file_emotion = []
file_path = []
savee_directory_list = os.listdir(Savee)
for file in savee_directory_list:
    file_path.append(Savee + file)
    file_emotion.append(emotion_map.get(file.split('_')[1][:-6])) #default=None if key not found

#dataframe for emotion from files
df_emotion = pd.DataFrame(file_emotion, columns=['emotion'])

#dataframe for path of files
df_path = pd.DataFrame(file_path, columns=['path'])

#concat path and emotions
df_audio = pd.concat([df_emotion, df_path], axis=1)
df_audio.head()
```

EDA

FREQUENCY OF EMOTIONS IN DATA FILES



WAVE PLOT AND SPECTROGRAMS

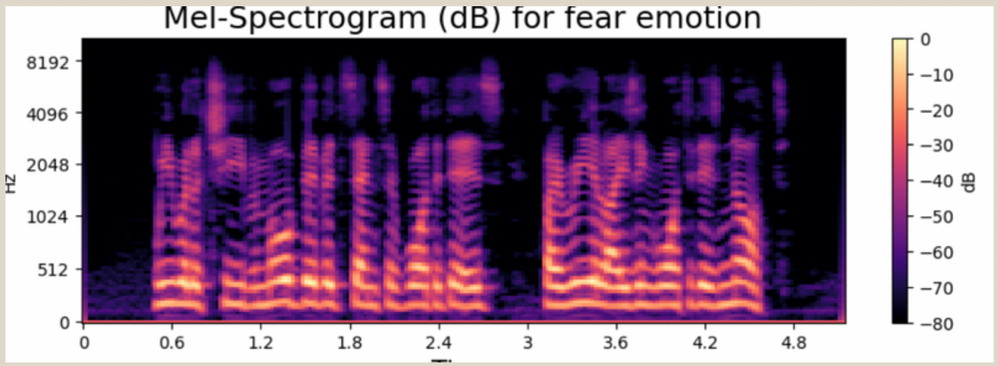


CODE FOR EDA

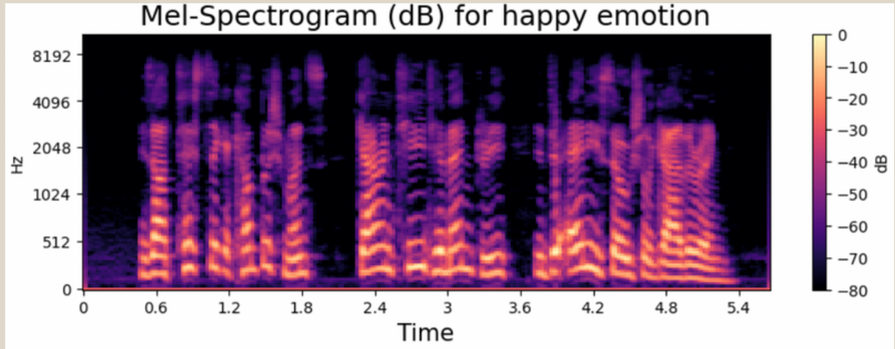
```python
def create_waveplot(data, sr, emo):
    '''
      creates waveplot from the data for the given sampling rate.
    '''
    plt.figure(figsize=(10, 3))
    plt.title('Waveplot for audio with {} emotion'.format(emo), size=15)
    librosa.display.waveshow(data, sr=sr)
    plt.ylabel('amplitude')
    plt.show()

def create_spectrogram(data, sr, emo, freq='hz'):
    '''
      creates spectrogram from the data for the given sampling rate.
      function = ['hz','log']
      output contains a spectrogram with
                    the horizontal axis representing time,
                    the vertical axis represents frequency,
                    and the color intensity represents the amplitude of a frequency at a certain point in time.
    '''
    #stft function converts the data into short term fourier transform
    X = librosa.stft(data)
    Xdb = librosa.amplitude_to_db(abs(X))
    plt.figure(figsize=(10, 3))
    plt.title('Audio Spectrogram with {} emotion [ {} ]'.format(emo, freq), size=15)
    librosa.display.specshow(Xdb, sr=sr, x_axis='time', y_axis=freq)
    plt.colorbar(label='amplitude of frequency')
    plt.show()
```

```python
def create_mel_spectrogram(data, sr, emo):
    '''
      creates mel-spectogram, which is perceptually relevant amplitude and frequency representation.
    '''
    mel_signal = librosa.feature.melspectrogram(y=data, sr=sr)
    spectrogram = np.abs(mel_signal)
    power_to_db = librosa.power_to_db(spectrogram, ref=np.max)
    plt.figure(figsize=(10, 3))
    librosa.display.specshow(power_to_db, sr=sr, x_axis='time', y_axis='mel', cmap='magma')
    plt.colorbar(label='dB')
    plt.title('Mel-Spectrogram (dB) for {} emotion'.format(emo), fontdict=dict(size=18))
    plt.xlabel('Time', fontdict=dict(size=15))
```

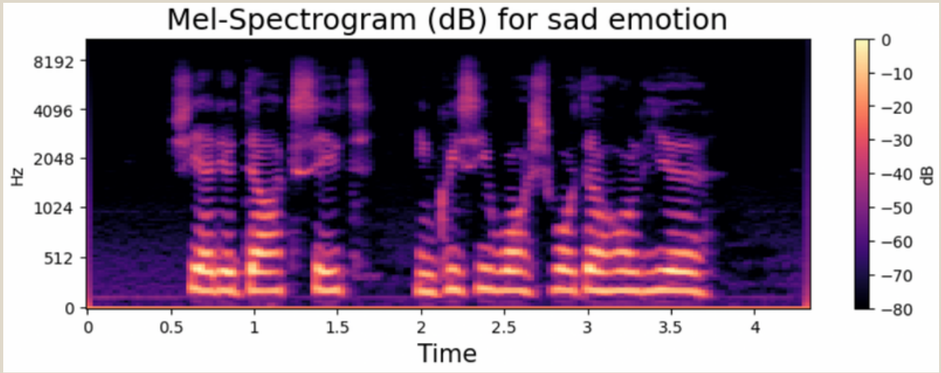
MEL-SPECTOGRAMS



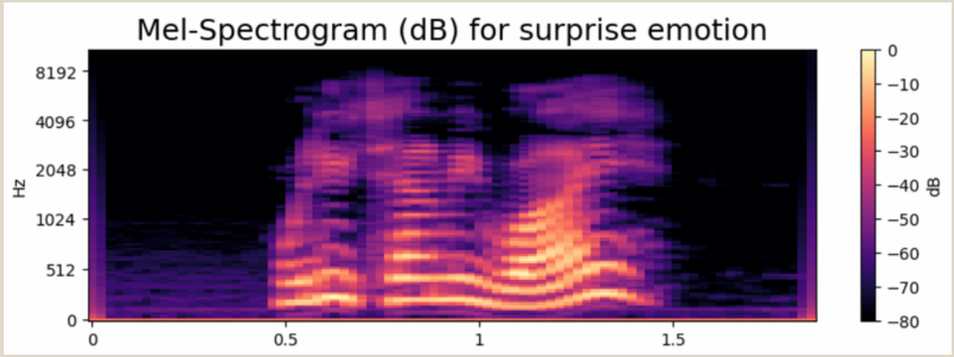
FEAR



HAPPY



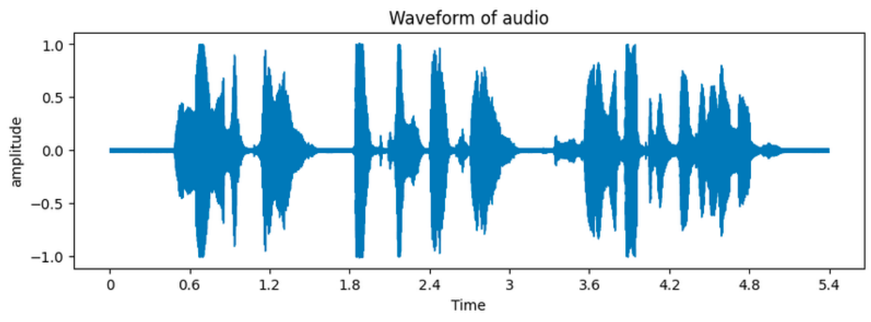
SAD



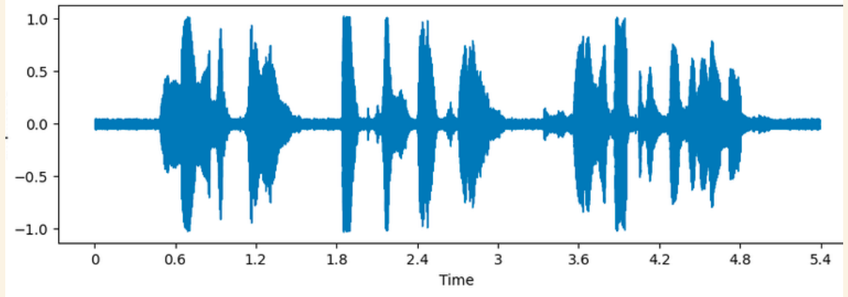
SURPRISE

Data Preparation

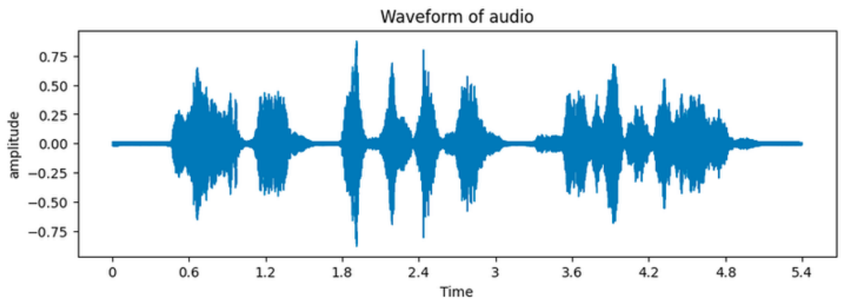
AUGMENTATION



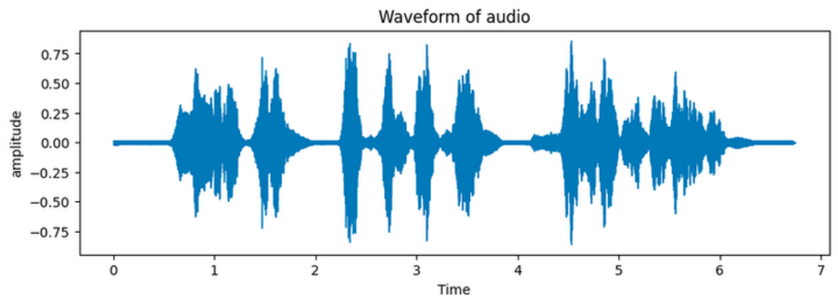
- Original



- Noise



- Pitching



- Stretching

Data Preparation

FEATURE EXTRACTION

```python
def extract_features(data):
    # ZCR
    result = np.array([])
    zcr = np.mean(librosa.feature.zero_crossing_rate(y=data).T, axis=0)
    result=np.hstack((result, zcr)) # stacking horizontally

  # Chroma_stft
    stft = np.abs(librosa.stft(data))
    chroma_stft = np.mean(librosa.feature.chroma_stft(S=stft, sr=sample_rate).T, axis=0)
    result = np.hstack((result, chroma_stft)) # stacking horizontally

    # MFCC
    mfcc = np.mean(librosa.feature.mfcc(y=data, sr=sample_rate).T, axis=0)
    result = np.hstack((result, mfcc)) # stacking horizontally

    # Root Mean Square Value
    rms = np.mean(librosa.feature.rms(y=data).T, axis=0)
    result = np.hstack((result, rms)) # stacking horizontally

    # MelSpectogram
    mel = np.mean(librosa.feature.melspectrogram(y=data, sr=sample_rate).T, axis=0)
    result = np.hstack((result, mel)) # stacking horizontally
    #spectral centroid
    spec_cent=np.mean(librosa.feature.spectral_centroid(y=data, sr=sample_rate).T, axis=0)
    result = np.hstack((result, spec_cent)) # stacking horizontally

    #spectral flux
    onset_env =np.mean( librosa.onset.onset_strength(sr=sample_rate, S=librosa.amplitude_to_db(data, ref=np.max)))
    result=np.hstack((result,onset_env))
    #mler
    Mler=mler(rms)
```

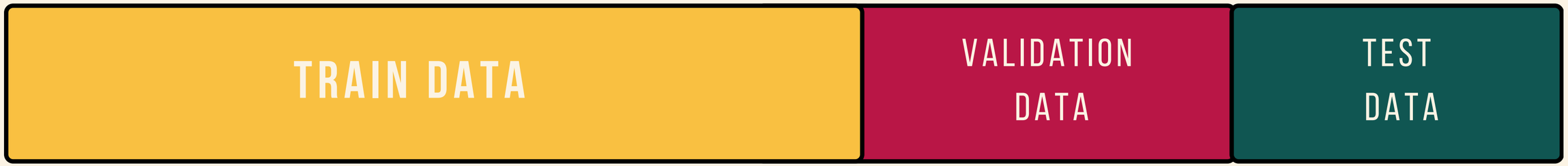
- Zero crossing rate (ZCR) - 1
- Chroma STFT - 12
- Mel-frequency cepstral coefficients (MFCC) -20
- Root mean square energy (RMSE) - 1
- Mel-spectrogram mean - 128
- Spectral centroid - 1
- Spectral flux - 1
- Chroma cens - 1
- Spectral roll off - 1

TOTAL 167

Data Preparation

PARTITIONING

TOTAL SAMPLES : 1440



1166 130 114

Data Preparation

ENCODING - NORMALIZATION - ADDING DIMENSION

- Encoding is the process of converting categorical data into numerical values.
- Normalization is the process of scaling the numerical data in a way that it has a consistent scale and range.
- Addition of 3rd dimension.

```
#Encoding our target variable
encoder = OneHotEncoder()
y_train2 = encoder.fit_transform(np.array(y_train2['labels']).reshape(-1,1)).toarray()
y_val2 = encoder.transform(np.array(y_val2['labels']).reshape(-1,1)).toarray()
y_test2 = encoder.transform(np.array(y_test2['labels']).reshape(-1,1)).toarray()

#Normalizing numeric values
scaler = StandardScaler()
x_train2 = scaler.fit_transform(x_train2)
x_val2 = scaler.transform(x_val2)
x_test2 = scaler.transform(x_test2)

#Adding extra dimension for 1D CNN
#3rd dimension will be set to 1
x_train2 = np.expand_dims(x_train2, axis=2)
x_val2 = np.expand_dims(x_val2, axis=2)
x_test2 = np.expand_dims(x_test2, axis=2)

print("Train set : ", x_train2.shape, y_train2.shape)
print("Valdiation set: " , x_val2.shape, y_val2.shape)
print("Test set: ",  x_test2.shape, y_test2.shape)
```

```
Train set :  (1166, 167, 1) (1166, 7)
Valdiation set:  (130, 167, 1) (130, 7)
Test set:  (144, 167, 1) (144, 7)
```

MODEL ARCHITECTURE

FOR AUDIO EMOTION DETECTION

- Input features
- 4 x 1D-CNN Layers
- Output Emotion

SUMMARY OF AUDIO SENTIMENT

Layer (type)	Output Shape	Param #
conv1d_68 (Conv1D)	(None, 167, 256)	1536
max_pooling1d_68 (MaxPooling1D)	(None, 84, 256)	0
conv1d_69 (Conv1D)	(None, 84, 256)	327936
max_pooling1d_69 (MaxPooling1D)	(None, 42, 256)	0
conv1d_70 (Conv1D)	(None, 42, 128)	163968
max_pooling1d_70 (MaxPooling1D)	(None, 21, 128)	0
dropout_34 (Dropout)	(None, 21, 128)	0
conv1d_71 (Conv1D)	(None, 21, 64)	41024
max_pooling1d_71 (MaxPooling1D)	(None, 11, 64)	0
flatten_17 (Flatten)	(None, 704)	0
dense_34 (Dense)	(None, 32)	22560
dropout_35 (Dropout)	(None, 32)	0
dense_35 (Dense)	(None, 7)	231

Total params: 557,255
Trainable params: 557,255
Non-trainable params: 0

EXPERIMENTS

DataSet-1

RANDOMLY DISTRIBUTED TARGETS

might result in a suboptimal model as some classes may be over/under represented leading to bias in learning.

```
y_train1['labels'].value_counts()

neutral     300
angry       152
happy       145
fear        144
surprise    143
sad         141
disgust     141
Name: labels, dtype: int64
```

DataSet-2

UNIFORMLY DISTRIBUTED TARGETS

to prevent biased sampling and results in more accurate model evaluation.

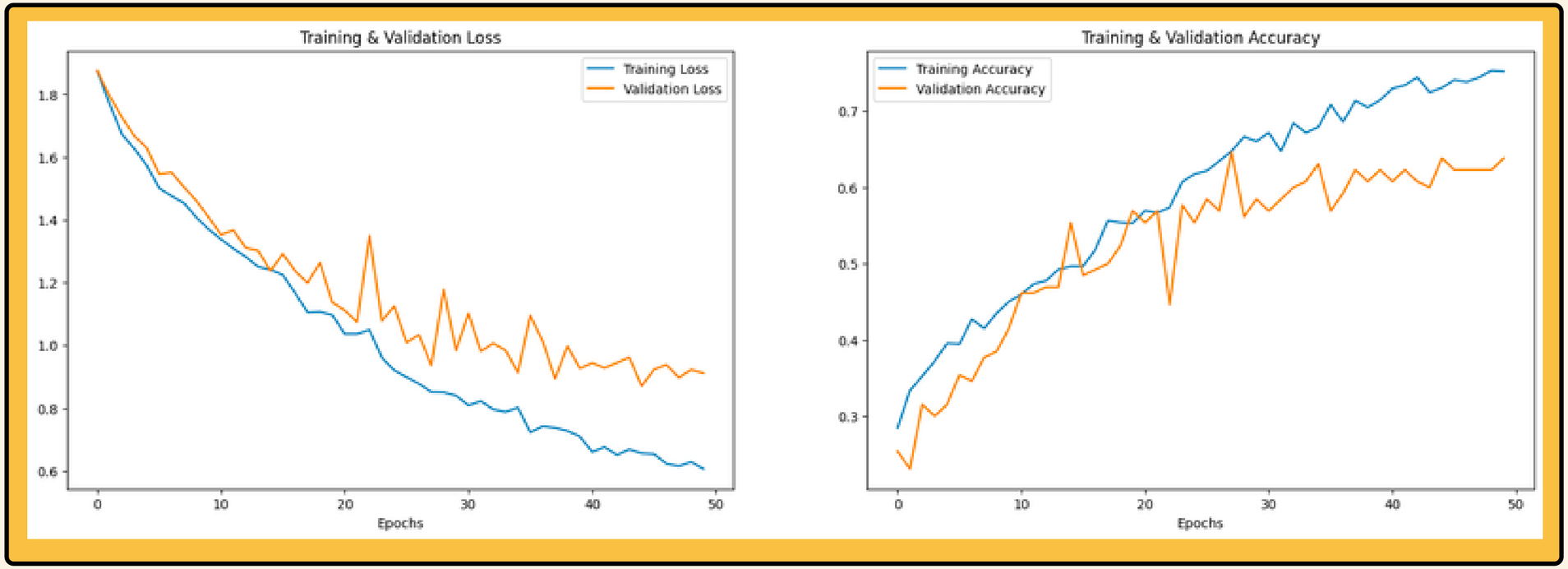
```
[ ] y_train2['labels'].value_counts()

neutral     291
fear        146
surprise    146
angry       146
happy       146
sad         146
disgust     145
Name: labels, dtype: int64
```

EVALUATION

DataSet-1

RANDOMLY DISTRIBUTED TARGETS

- Accuracy increases over time
- Loss decreases over time
- Training Accuracy : 84.82%
- Testing Accuracy : 63.89%

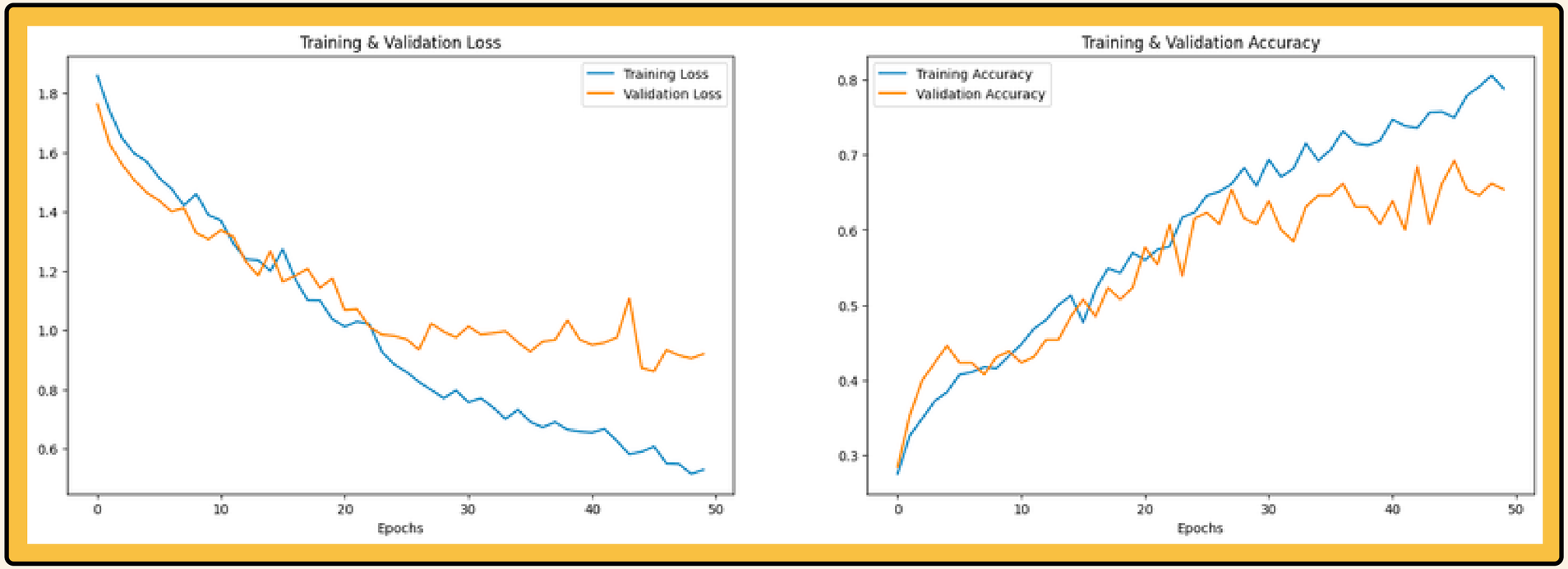


EVALUATION

DataSet-2

UNIFORMLY DISTRIBUTED TARGETS

- Accuracy increases over time
- Loss decreases over time
- Training Accuracy : 87.99%
- Testing Accuracy : 72.91%



Results

CLASSIFICATION REPORT

- Model-1 : poor performance for "fear"
- Model-2 : performance increases for every emotion
- "neutral" & "sad" has very good precision and recall
- 60-70% F1-scores for "disgust", "happy", "fear"
- Comparatively poor performance for “angry” and “surprise”.

	precision	recall	f1-score	support
angry	0.53	0.64	0.58	14
disgust	0.65	0.69	0.67	16
fear	0.38	0.31	0.34	16
happy	0.42	0.50	0.45	20
neutral	0.81	0.92	0.86	38
sad	0.86	0.63	0.73	19
surprise	0.62	0.48	0.54	21
accuracy			0.64	144
macro avg	0.61	0.60	0.60	144
weighted avg	0.64	0.64	0.63	144

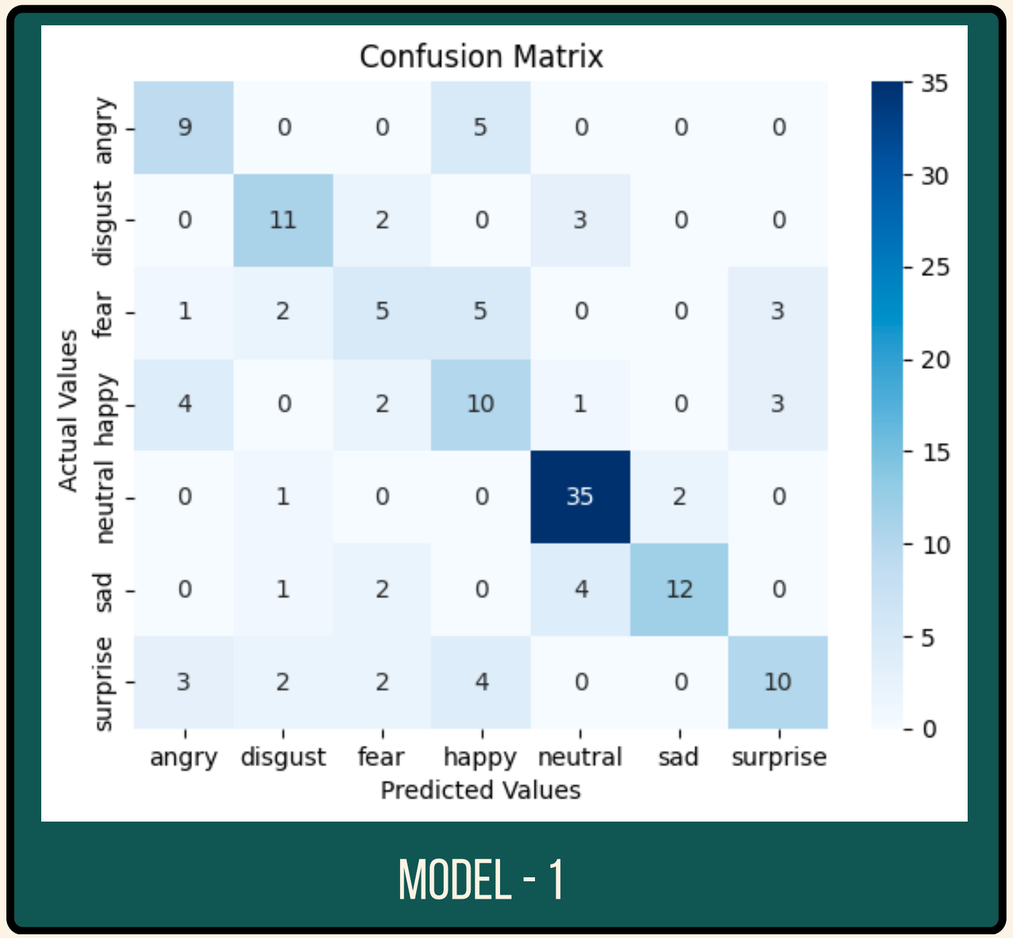
MODEL - 1

	precision	recall	f1-score	support
angry	0.69	0.50	0.58	18
disgust	0.59	0.72	0.65	18
fear	0.67	0.56	0.61	18
happy	0.67	0.78	0.72	18
neutral	0.87	0.92	0.89	36
sad	0.89	0.89	0.89	18
surprise	0.59	0.56	0.57	18
accuracy			0.73	144
macro avg	0.71	0.70	0.70	144
weighted avg	0.73	0.73	0.72	144

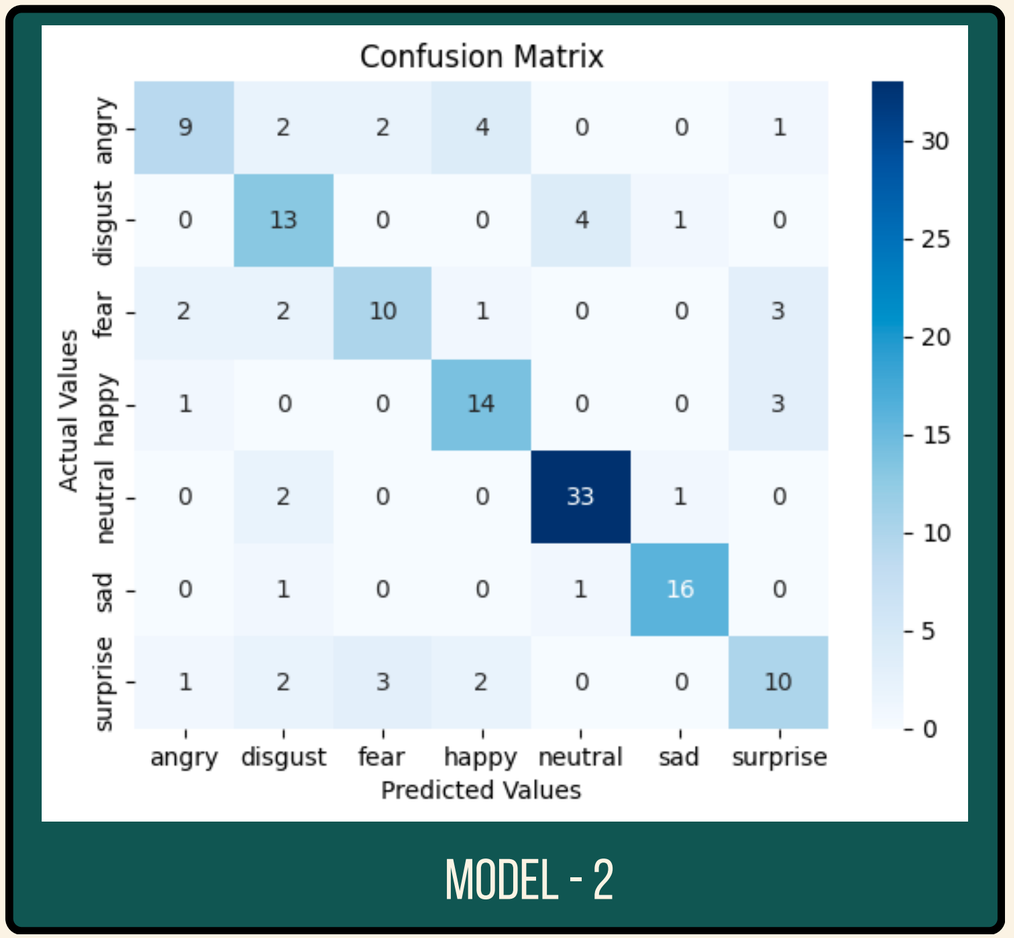
MODEL - 2

Results

CONFUSION MATRIX



MODEL - 1

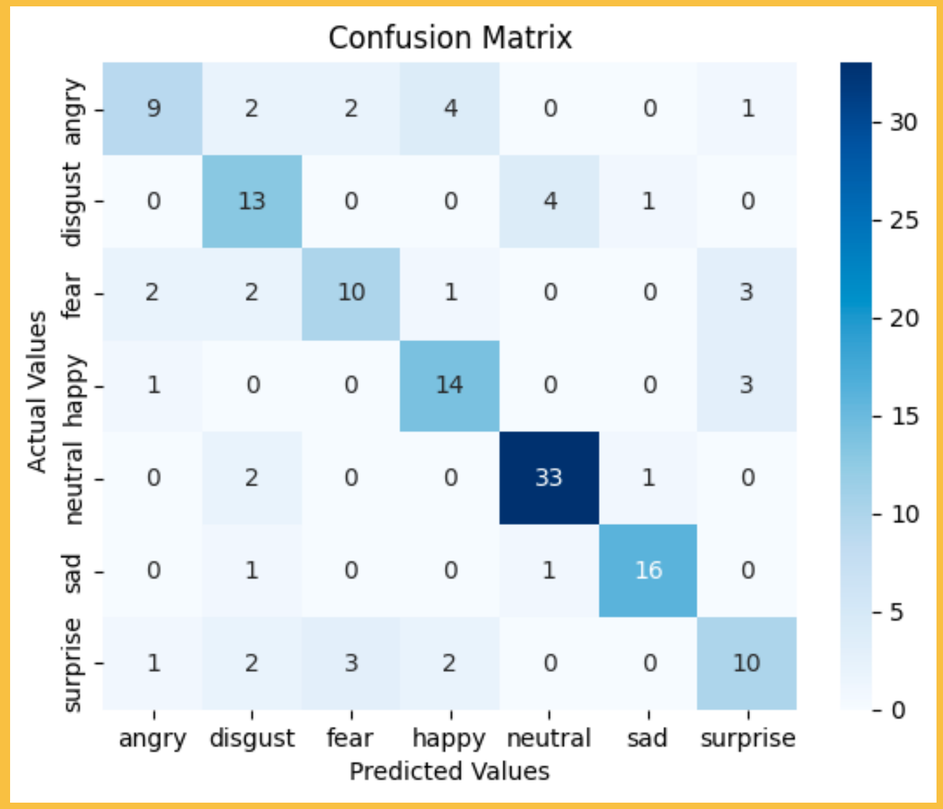


MODEL - 2

Results

CONFUSION MATRIX

- "angry" & "surprise" have high false positives
- "angry" misclassified as "disgust", "fear" & "happy"
- "surprise" misclassified as "angry", "disgust", "fear" & "happy"
- few "happy" samples misclassified as "surprise"
- similarly, few "fear" samples are misclassified as "surprise"



MODEL - 2

TEXT ANALYSIS *Rundown*

CONVERT AUDIO TO TEXT

CONVERT DATA TO USABLE STATE

TRAIN MODEL

CHECK RESULTS

PROJECT PRESENTATION

Audio TO TEXT

Speech recognition module and recognize_google() method are used to transcribe our audio files.

- SPEECH RECOGNITION
- RECOGNIZE_GOOGLE()

CONVERT DATA TO USABLE STATE

CODE

```
#word tokenization of text using nltk
texts = [' '.join(word_tokenize(text)) for text in df.text]
texts_train = [' '.join(word_tokenize(text)) for text in X_train.text]
texts_val = [' '.join(word_tokenize(text)) for text in X_val.text]
texts_test = [' '.join(word_tokenize(text)) for text in X_test.text]
texts train[1:6]
tokenizer = Tokenizer()
tokenizer.fit_on_texts(texts)

sequence_train = tokenizer.texts_to_sequences(texts_train)
sequence_val = tokenizer.texts_to_sequences(texts_val)
sequence_test = tokenizer.texts_to_sequences(texts_test)

X_train_pad = pad_sequences(sequence_train, maxlen = max_seq_len )
X_val_pad = pad_sequences(sequence_val, maxlen = max_seq_len )
X_test_pad = pad_sequences(sequence_test, maxlen = max_seq_len )
```

RESULTS

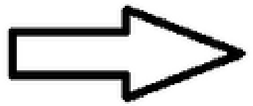
did you talk safety greasy wash with all year

should your dog suit in Greasy wash water all ...

his autistic accomplishments guaranteed him en...

Bob burn paper and leaves with a big bonfire

project development was completing too slowly



```
array([[  0,   0,   0, ...,  57,  16,  28],
       [  0,   0,   0, ...,  38,  16,  28],
       [  0,   0,   0, ...,  66, 216, 217],
       ...,
       [  0,   0,   0, ..., 649,  16,  28],
       [  0,   0,   0, ..., 358,  16, 359],
       [  0,   0,   0, ..., 108, 365, 366]], dtype=int32)
```

CONVERT DATA TO USABLE STATE

CODE

```
encoding = {
    'neutral':0,
    'angry':1,
    'happy':2,
    'sad':3,
    'fear':4,
    'disgust':5,
    'surprise':6
}

# Integer labels
y_train = [encoding[x] for x in data_train.label]
y_val = [encoding[x] for x in data_val.label]
y_test = [encoding[x] for x in data_test.label]

y_train = to_categorical(y_train)
y_val = to_categorical(y_val)
y_test = to_categorical(y_test)
```

RESULTS

label
surprise
angry
happy
disgust
happy

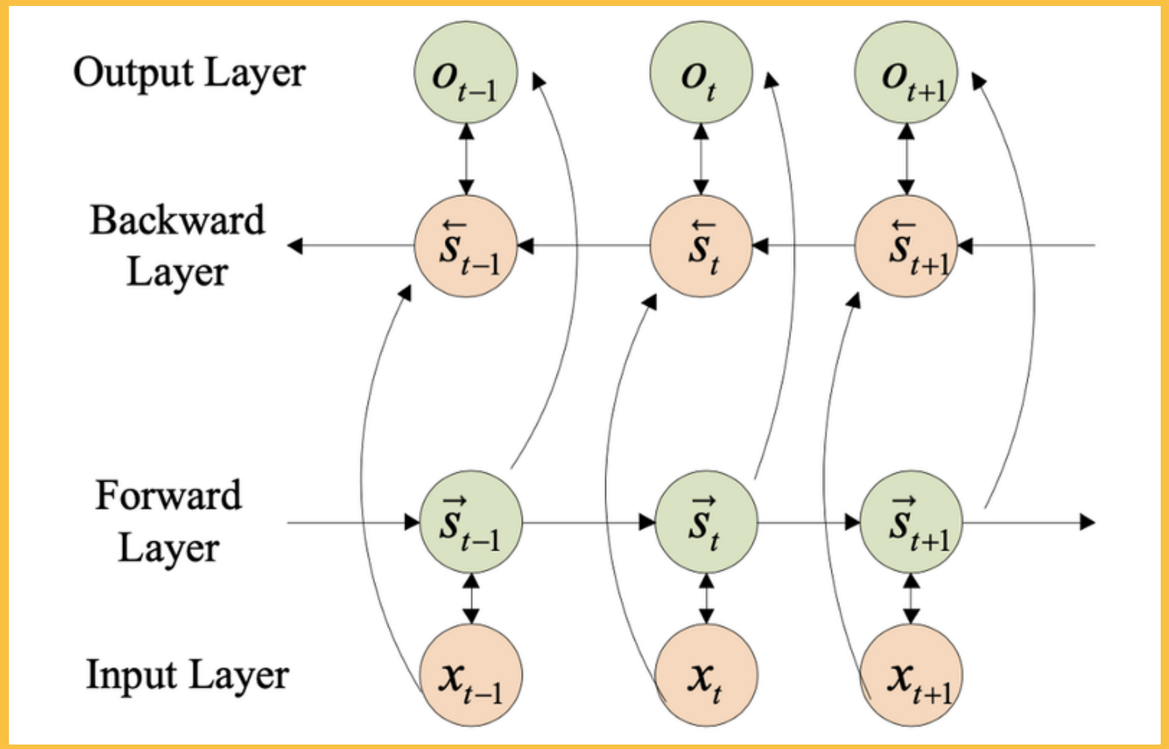
```
array([[0., 0., 0., ..., 0., 0., 1.],
       [0., 1., 0., ..., 0., 0., 0.],
       [0., 0., 1., ..., 0., 0., 0.],
       ...,
       [0., 0., 0., ..., 0., 0., 1.],
       [1., 0., 0., ..., 0., 0., 0.],
       [1., 0., 0., ..., 0., 0., 0.]], dtype=float32)
```

PROJECT PRESENTATION

Text

SENTIMENT ANALYSIS

This part functions as analyzing the audio files converted to text using Keras and Bidirectional GRU.



- KERAS
- BIDIRECTIONAL GRU

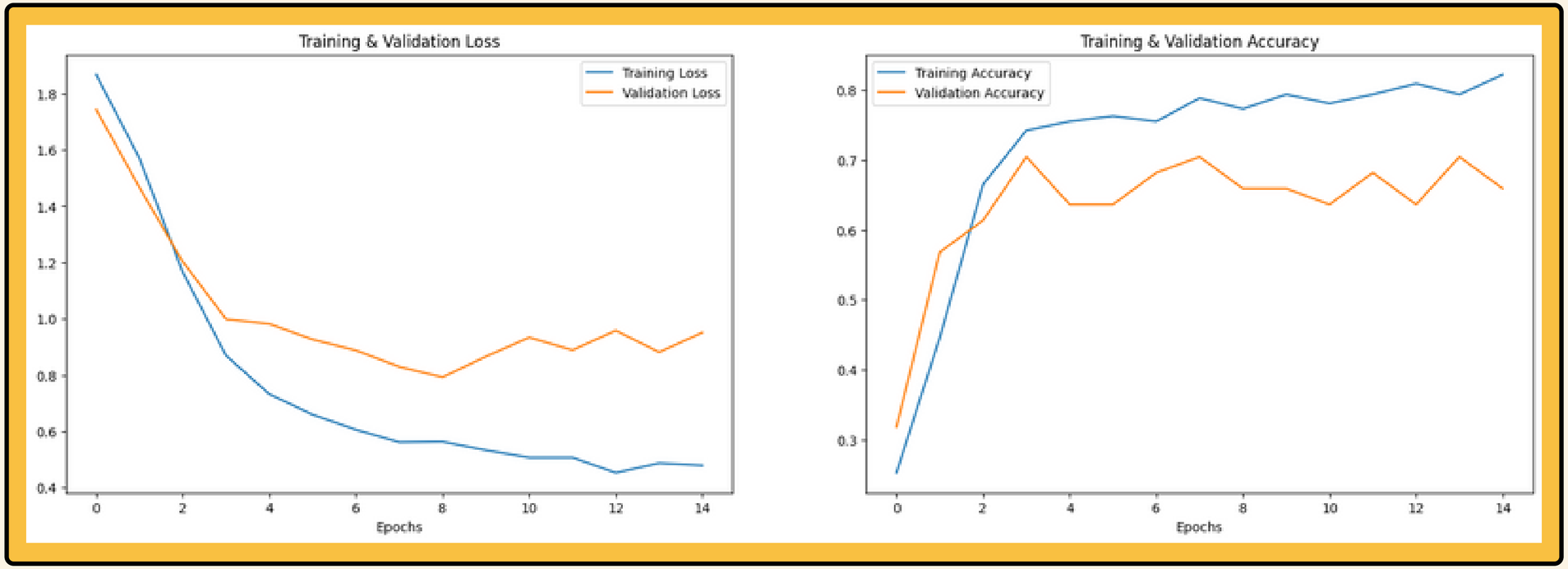
MODEL TRAINING


```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_2 (Embedding)     (None, 500, 300)          199500

 bidirectional_2 (Bidirectio  (None, 256)              330240
 nal)

 dense_38 (Dense)            (None, 7)                 1799

=================================================================
Total params: 531,539
Trainable params: 332,039
Non-trainable params: 199,500
_________________________________________________________________
```

EVALUATION

LOSS AND ACCURACY GRAPHS

- Accuracy increases over time
- Loss decreases over time
- Training Accuracy : 84.54%
- Testing Accuracy : 58.33%



Results

CLASSIFICATION REPORT

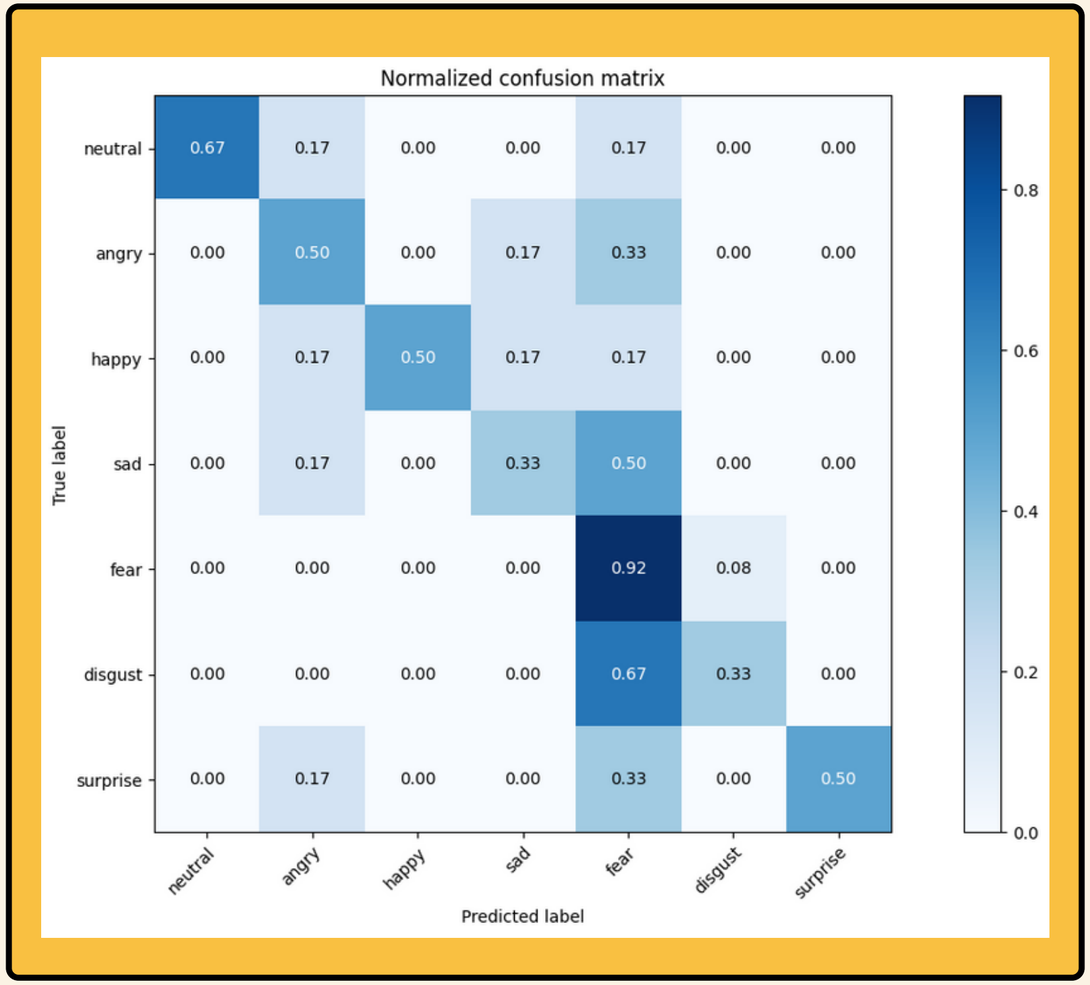
- Poor overall performance
- Good precision and f-score for anger
- "neutral" has very good recall

	precision	recall	f1-score	support
angry	1.00	0.67	0.80	6
disgust	0.43	0.50	0.46	6
fear	1.00	0.50	0.67	6
happy	0.50	0.33	0.40	6
neutral	0.46	0.92	0.61	12
sad	0.67	0.33	0.44	6
surprise	1.00	0.50	0.67	6
accuracy			0.58	48
macro avg	0.72	0.54	0.58	48
weighted avg	0.69	0.58	0.58	48

Results

CONFUSION MATRIX

- "disgust", "sad" misclassified as "fear"
- "surprise" and "angry" are also misclassified as "fear" but to a lower degree
- "fear" has high true positives



Thank You

THE END