Title of the Project : **Student Behavioral Analysis via Emotional Speech Recognition**

Team ▸ **SOUBHIK SINHA [19BIT0303]**
SUKANT JHA [19BIT0359]

Abstract
Online Classes have been happening for the past two years because of COVID pandemic. Thus , teachers find it extremely difficult to understand a student's scenario - their progress , their responses, whether they are on track with the study path or not. Their tone of speech reveals their understanding of the topic being learned. Recognizing emotion from speech has become one of the most active research topics in speech processing and human-computer interaction applications. This project is an attempt to see if students can recognise emotions from their spoken replies. Neutral, Anger, Joy, and Sorrow are among the emotions explored in the tests.

I. Introduction

We have three key aims in emotional speech detection. The initial purpose is to keep a current list of all available emotional speech data sets. The amount of emotional states, language, speaker count, and kind of speech are all briefly discussed. The second purpose is to provide the most common acoustic characteristics utilized for emotional speech identification, as well as to evaluate how emotion influences them. Pitch, formants, vocal tract cross-section areas, mel-frequency cepstral coefficients, Teager energy operator-based features, speech signal intensity, and speech rate are all common characteristics. The final purpose is to go through proper procedures for categorizing speech into different emotional states. We compare and contrast categorization approaches that use temporal information vs those that do not. Hidden Markov models, artificial neural networks, linear discriminant analysis, k-nearest neighbors, and support vector machines are all discussed as classification algorithms. Emotion recognition works on the basis of analyzing the acoustic differences that occur while saying the same phrase in various emotional contexts. Aside from the qualities that correspond to

the speaker and/or the speech, sound signals also include certain aspects that indicate the speaker's emotional state.

II. Description of the Project

We are trying to study and predict (as output) the emotions coming out of a student's speech at time of interacting with their teachers - to predict the topic's understanding by the student. From this point onwards we shall take up a technical explanation path. The technique of adding tiny perturbations to an original dataset to generate fresh training examples is known as data augmentation. This is done to make the model more accurate by making it invariant to such disturbances. As stated below, we use four distinct data augmentation strategies. Before generating the MFCC features, which are utilized to train our deep learning model, these perturbations are applied directly to the audio data. The following are the enhancement techniques that were used:

• **Noise** : To build a noised training set for the network, each audio sample is combined with an AWGN.

• **Shifting** : the audio samples are shifted within a certain range.

• **Pitch Shifting** : The audio files' pitch is adjusted without affecting their length. This is a fantastic method of enhancement. With a pitch factor of 0.8, each audio sample was pitch altered.

• **Time Stretching** : The audio samples are stretched in time without modifying their pitch. Three variables are used to extend each audio sample: [0.70, 0.75, 1.25]. If the speed factor is more than one, the length of the audio sample is lengthened. The length of audio samples is reduced when the speed factor is less than one.

Using the numpy and librosa libraries, the aforementioned data augmentation techniques are applied to the supplied datasets. This is done to improve the model's capacity to generalize. These are done before Mel Frequency Cepstral Coefficients (MFCC) are used to transform the audio recordings into a format that our model can understand -

Mel Frequency Cepstral Coefficient - These coefficients are dependent on a person's capacity to hear. Two types of filters are utilized here, one of which is spaced linearly at

low frequencies below 1kHz and the other of which is spaced logarithmically at high frequencies over 1KHz.

The MFCC feature extraction technique is described below –

Pre-emphasis : Pre-emphasis is a method of increasing the energy of a voice signal by passing it through a filter. This increase in energy level provides additional information.

Framing : the audio sample is divided into 20-40ms frames in this case. Because the duration of a human voice can change, this procedure is required to set the size of speech. The voice signal is not a stationary signal (its frequency fluctuates over time), although it might behave like one for a brief period of time.

Windowing : After the framing procedure, this method is used to eliminate signal discontinuities at the beginning and end of each frame. A frame is moved with a 10ms gap, implying that it has some overlap with the previous frame.

Fast Fourier Transform : The frequency of each frame is determined using the Fast Fourier Transform. The FFT is used to transform each sample of each frame from time domain to frequency domain.

Mel Scale Filter Bank : each frame is filtered using 20-30 triangle filters. The Mel Scale Frequency Spectrum determines the amount of energy present in a given frame.

Low-Energy Computation : this is based on how humans perceive sound. Loud loudness is not heard on a linear scale by humans. When the sound level is high, the human ear is unable to detect big fluctuations in energy. As a result, every frame's filter bank energy is filtered using the log function. The properties of log energy calculation are easily understandable by people.

DCT (Discrete Cosine Transform) : The DCT technique is used to return the Mel spectrum to the spatial domain. Because DCT signals have information concentrated in a limited number of coefficients, it is favored over DFT. As a result, representing Mel spectrum in a limited number of coefficients necessitates a little amount of storage. The output of DCT is referred to as MFCC.

Now let us talk about the Neural Network we shall be using in this project. Convolutional Neural Networks are used in the Speech Recognition Application (CNN). The system receives training data, which includes the expression label and That network also has access to weight training. As an input, an audio file is used. After that, the audio is subjected to intensity normalization. For the Convolutional Networks training, normalized audio is used. The data is then remixed, splitting it into train and test groups, and a CNN model and its subsequent layers are built to train the dataset. Following that, the training data is used to predict a human voice emotion.

There are four major layers in the CNN model :

1. Feature map sequence is depicted by a **convolutional layer** that identifies salient areas at intervals, length utterances that are variable, and a convolutional layer that identifies salient regions at intervals.

2. **Activation layer** : As is common for convolutional layer outputs, a non-linear Activation layer function is utilized. During our research, we employed the corrected linear unit (ReLU).

3. **Max Pooling layer** : This layer allows the Dense layers to have the most possibilities. It makes it easier to keep the variable length inputs contained within a fixed-size feature array.

4. **Dense Layer :**

(a) **Audio Features and Visualization** in the Dense Layer

Librosa will be used to analyze and extract properties from any audio stream. The (.load) function takes an audio file and decrypts it into a 1D array of time series x, where SR is the sampling rate of x. Librosa.display is used to show audio files in various formats, including as wave plots, spectrograms, and colormaps.

(b) **To hone the model's computation accuracy -**

With the librosa packages and the mfcc function, we'll get the sampling rate value. Other variables are then stored in this value. Now that audio files and mfcc value have a variable, it will create a list. After that, zip the list and keep two variables, x and y. Then we used the numpy package to express (x, y) shape values.

(c) **Implementation process of CNN model**

Speech is represented as a three-layered picture. Consider the 1st and 2nd derivatives of the speech image with time and frequency while using CNN. CNN can anticipate, evaluate, and learn from talks, as well as recognise words and phrases.

(d) **Speech emotion classification**

To begin, use librosa.load to extract the Characteristics. It turns one data frame into a structured format and displays it. Furthermore, the loaded model is compared using the predict function batch size 32. Finally, it shows the audio file's output to show what kind of emotion it has.

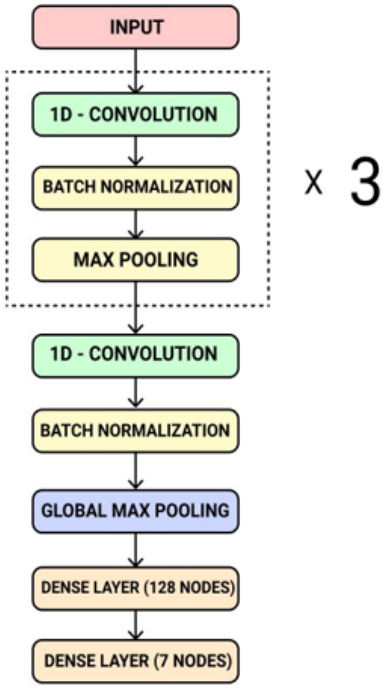


Fig.1 : CNN model

III. Literature Review

1. To achieve this in this [1] study, the authors used a hybrid neural network with four blocks of temporally distributed convolutional layers followed by a layer of Long Short Term Memory. The speech dataset's audio samples are compiled from RAVDESS, TESS, and SAVEE audio datasets, and then enhanced by noise injection. Mel Spectrograms are used to train the neural network using audio samples. We were able to attain a testing accuracy of around 89.26%.

2. The Deep Neural Network (DNN) architecture with convolutional, pooling, and fully connected layers is described in this research as a solution for Speech Emotion Recognition (SER). The authors used 271 labeled recordings with a total length of 783 seconds from the German Corpus (Berlin Database of Emotional

Speech), divided into three classes (angry, neutral, and sad). The raw audio data was normalized to have a zero mean and unit variance for each audio file. Without overlap, each file was divided into 20 millisecond parts. To exclude quiet parts, we employed the Voice Activity Detection (VAD) algorithm and separated the data into three sets: TRAIN (80%), VALIDATION (10%), and TESTING (10%). Stochastic Gradient Descent is used to optimize DNN. We utilized raw data without any feature selection as input. Our model has a 96.97 % accuracy rating [2].

3. Because of the gap between acoustic features and human emotions, Automated Speech Emotion Recognition is a difficult process that relies heavily on the discriminative acoustic characteristics retrieved for a given recognition job. Different people have different emotions and exhibit them in different ways. When contemplating diverse issues, speech emotion has varied intensities, and pitch fluctuations are highlighted. As a result, detecting speech emotion is a difficult issue in computational vision. The voice emotion identification system here is based on the Convolutional Neural Network (CNN) algorithm, which employs several modules for emotion detection and classifiers to discriminate emotions like happy, surprise, anger, neutral mood, sorrow, and so on. Speech samples are used as the dataset for the speech emotion detection system, and the characteristics are collected from these speech samples using the LIBROSA package. The performance of categorization is based on extracted features. Finally, the emotion of a voice signal may be determined [3].

4. The research makes two major contributions: first, we present a deep CNN architecture for categorization of ambient sounds. Second, we suggest using audio data augmentation to solve the problem of data scarcity, and we investigate the impact of various augmentations on the proposed CNN architecture's performance. The suggested approach offers state-of-the-art results for ambient sound categorization when combined with data augmentation. We show that combining a deep, high-capacity model with an augmented training set improves performance: this combination outperforms both the proposed CNN without augmentation and a "shallow" dictionary learning model with augmentation. Finally, we investigate the impact of each augmentation on the model's classification accuracy for each class, finding that each augmentation has a distinct impact on each class, implying that

the model's performance might be enhanced further by using class-conditional data augmentation [4].

5. This study presents a novel MFCC extraction technique for voice recognition. In comparison to the traditional technique, the new algorithm decreases processing power by 53%. The new method has a recognition accuracy of 92.93 percent, according to simulation findings. When compared to the traditional MFCC extraction method, which has a recognition accuracy of 94.43 percent, there is just a 1.5 percent drop in recognition accuracy. The number of logic gates required to implement the new method, on the other hand, is around half that of the MFCC algorithm, making it particularly efficient for hardware implementation [5].

6. Voice coding, speech synthesis, speech recognition, and speaker recognition are all examples of speech processing techniques. Speech processing has a wide range of applications in the field of digital signal processing, hence it is still a hotbed of study. Feature extraction and classification are the two most common processes in speech processing. The selection of feature extraction approach, which plays a vital role in the system accuracy, is the key requirement for a competent speech processing system. Fast Fourier Transforms, Linear Predictive Coding, Mel Frequency Cepstral Coefficients, Discrete Wavelet Transforms, Wavelet Packet Transforms, Hybrid Algorithm DWPD, and its applications in speech processing are all covered in this study [6].

IV. Code and Analysis

Before moving to the implementation , we shall look upon the datasets used in this analysis —

RAVDESS dataset : There are 7356 files in the RAVDESS collection of Emotional Speech and Song. Two lexically-matched phrases are vocalized in a neutral North American accent by 24 professional actors (12 female, 12 male) in the database.

SAVEE : Surrey Audio-Visual Expressed Emotion (SAVEE) database has been recorded as a need for developing an autonomous emotion identification system.

The collection contains 480 British English utterances recorded by four male actors in seven distinct emotions.

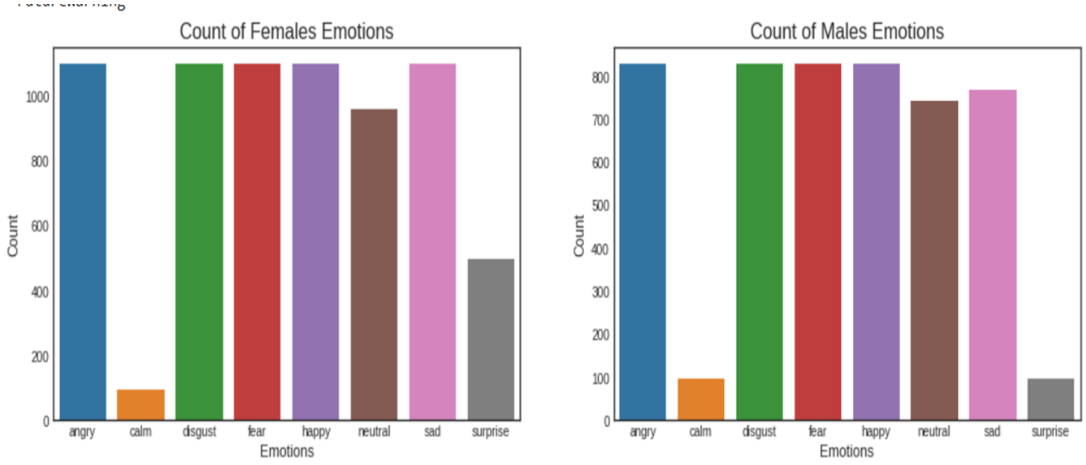
CREMA-D : CREMA-D is a collection of 7,442 unique footage by 91 different performers. These video featured 48 male and 43 female actresses ranging in age from 20 to 74, representing a diverse range of races and ethnicities (African America, Asian, Caucasian, Hispanic, and Unspecified). The actors read from a list of 12 phrases. Six different emotions (Anger, Disgust, Fear, Happy, Neutral, and Sad) and four different emotion degrees were used to convey the phrases.

Toronto emotional speech set (TESS) : Two actresses (ages 26 and 64) recited a set of 200 target phrases in the carrier phrase "Say the word _____," and recordings were taken of the set depicting each of seven moods (anger, disgust, fear, happiness, pleasant surprise, sadness, and neutral). There are a total of 2800 stimuli.

Now lets move on to the significant implementation portions of out code –

Visualization of our data taken -

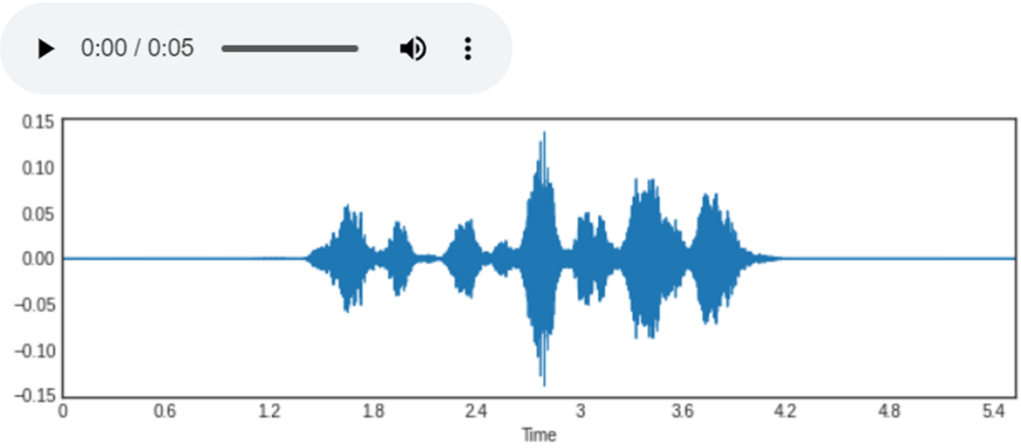
```
# visualization the data

order = ['angry','calm','disgust','fear','happy','neutral','sad','surprise']

fig = plt.figure(figsize=(17, 5))

fig.add_subplot(121)
plt.title('Count of Females Emotions', size=16)
sns.countplot(Females.labels, order = order)
plt.ylabel('Count', size=12)
plt.xlabel('Emotions', size=12)

fig.add_subplot(122)
plt.title('Count of Males Emotions', size=16)
sns.countplot(Males.labels, order = order)
plt.ylabel('Count', size=12)
plt.xlabel('Emotions', size=12)

plt.show()
```

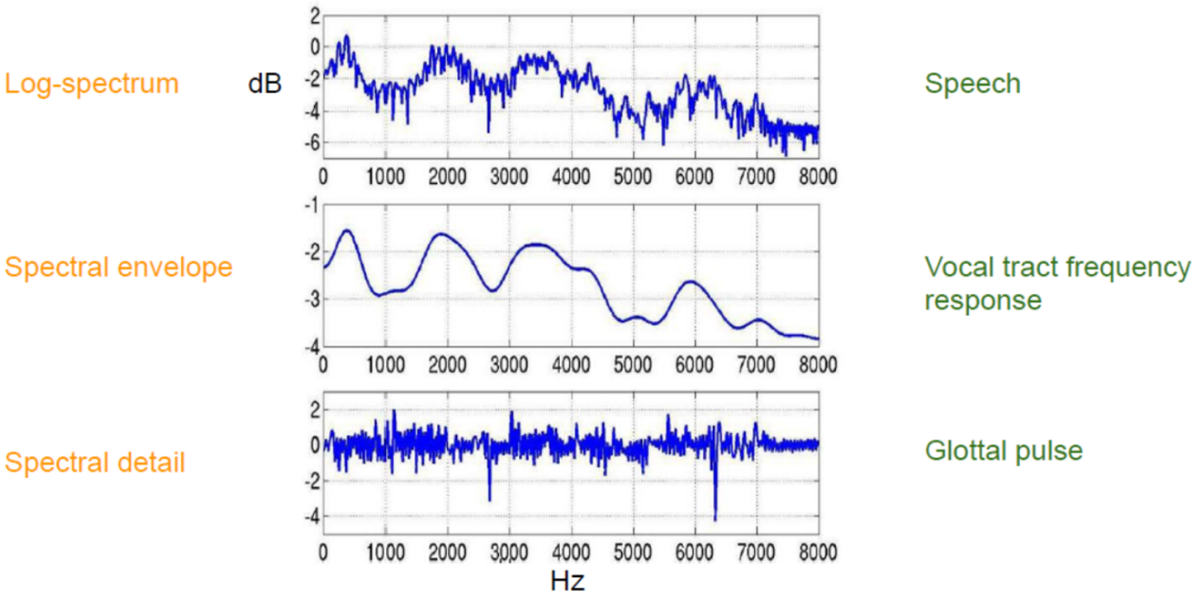


Data Augmentation —

```
def noise(data):
    noise_amp = 0.04*np.random.uniform()*np.amax(data)
    data = data + noise_amp*np.random.normal(size=data.shape[0])
    return data

def stretch(data, rate=0.70):
    return librosa.effects.time_stretch(data, rate)

def shift(data):
    shift_range = int(np.random.uniform(low=-5, high = 5)*1000)
    return np.roll(data, shift_range)

def pitch(data, sampling_rate, pitch_factor=0.8):
    return librosa.effects.pitch_shift(data, sampling_rate, pitch_factor)

def higher_speed(data, speed_factor = 1.25):
    return librosa.effects.time_stretch(data, speed_factor)

def lower_speed(data, speed_factor = 0.75):
    return librosa.effects.time_stretch(data, speed_factor)

path = RAV + '/Actor_01/03-01-05-01-01-01-01.wav'
data, sample_rate = librosa.load(path)
```

```
plt.figure(figsize=(10,3))
x = stretch(data)
librosa.display.waveplot(y=x, sr=sample_rate)
Audio(x, rate=sample_rate)
```





Extracting MFCCs :

```
def extract_features(data):

    result = np.array([])

    mfccs = librosa.feature.mfcc(y=data, sr=22050, n_mfcc=58)
    mfccs_processed = np.mean(mfccs,axis=1)
    result = np.array(mfccs_processed)
    return result
```

```
#without augmentation
res1 = extract_features(data)
result = np.array(res1)

# adding noise
noise_data = noise(data)
res2 = extract_features(noise_data)
result = np.vstack((result, res2)) # stacking vertically

#stretching the audio signal
stretch_data = stretch(data)
res3 = extract_features(stretch_data)
result = np.vstack((result, res3))
```

```
female_X, female_Y = [], []
for path, emotion in zip(Females.path, Females.labels):
    features = get_features(path)
    for elem in features:
        female_X.append(elem)
        female_Y.append(emotion)


male_X, male_Y = [], []
for path, emotion in zip(Males.path, Males.labels):
    features = get_features(path)
    for elem in features:
        male_X.append(elem)
        male_Y.append(emotion)
```

Adding augmentation and obtaining MFCC coefficients for all audio files in the datasets (above)

Splitting the audio files for training and testing purposes

```
x_trainF, x_testF, y_trainF, y_testF = train_test_split(female_X, female_Y, random_state=0, test_size=0.20, shuffle=True)
```

Using the Standardscalar, standardise the characteristics as a normal curve with zero mean and unit variance ().

```
scaler = StandardScaler()

x_train = scaler.fit_transform(x_train)
x_test = scaler.transform(x_test)

x_trainF = scaler.fit_transform(x_trainF)
x_testF = scaler.transform(x_testF)

x_trainM = scaler.fit_transform(x_trainM)
x_testM = scaler.transform(x_testM)
```

Deep learning model

```
with strategy.scope():

    def build_model(in_shape):

        model=Sequential()
        model.add(Conv1D(256, kernel_size=6, strides=1, padding='same', activation='relu', input_shape=(in_shape, 1)))
        model.add(AveragePooling1D(pool_size=4, strides = 2, padding = 'same'))

        model.add(Conv1D(128, kernel_size=6, strides=1, padding='same', activation='relu'))
        model.add(AveragePooling1D(pool_size=4, strides = 2, padding = 'same'))

        model.add(Conv1D(128, kernel_size=6, strides=1, padding='same', activation='relu'))
        model.add(AveragePooling1D(pool_size=4, strides = 2, padding = 'same'))
        model.add(Dropout(0.2))

        model.add(Conv1D(64, kernel_size=6, strides=1, padding='same', activation='relu'))
        model.add(MaxPooling1D(pool_size=4, strides = 2, padding = 'same'))

        model.add(Flatten())
        model.add(Dense(units=32, activation='relu'))
        model.add(Dropout(0.3))

        model.add(Dense(units=8, activation='softmax'))
        model.compile(optimizer = 'adam' , loss = 'categorical_crossentropy' , metrics = ['accuracy'])


        return model
```

Model Training :

```
history = total_model.fit(x_train, y_train, batch_size=batch_size, epochs=n_epochs, validation_data=(x_test, y_test), callbacks=[rlrp])
```

```
Epoch 1/75
2129/2129 [==============================] - 25s 11ms/step - loss: 1.3945 - accuracy: 0.4400 - val_loss: 1.1637 - val_accuracy: 0.5384 - lr: (
Epoch 2/75
2129/2129 [==============================] - 23s 11ms/step - loss: 1.1807 - accuracy: 0.5294 - val_loss: 1.1066 - val_accuracy: 0.5555 - lr: (
Epoch 3/75
2129/2129 [==============================] - 24s 11ms/step - loss: 1.1071 - accuracy: 0.5619 - val_loss: 1.0240 - val_accuracy: 0.5870 - lr: (
Epoch 4/75
2129/2129 [==============================] - 24s 11ms/step - loss: 1.0487 - accuracy: 0.5882 - val_loss: 0.9731 - val_accuracy: 0.6102 - lr: (
Epoch 5/75
2129/2129 [==============================] - 24s 11ms/step - loss: 1.0003 - accuracy: 0.6070 - val_loss: 0.9354 - val_accuracy: 0.6343 - lr: (
Epoch 6/75
2129/2129 [==============================] - 24s 11ms/step - loss: 0.9538 - accuracy: 0.6278 - val_loss: 0.9383 - val_accuracy: 0.6267 - lr: (
Epoch 7/75
2129/2129 [==============================] - 23s 11ms/step - loss: 0.9089 - accuracy: 0.6447 - val_loss: 0.8505 - val_accuracy: 0.6645 - lr: (
Epoch 8/75
2129/2129 [==============================] - 24s 11ms/step - loss: 0.8718 - accuracy: 0.6618 - val_loss: 0.8251 - val_accuracy: 0.6740 - lr: (
Epoch 9/75
2129/2129 [==============================] - 24s 11ms/step - loss: 0.8356 - accuracy: 0.6777 - val_loss: 0.7962 - val_accuracy: 0.6868 - lr: (
```

```
score = total_model.evaluate(x_train,y_train, verbose = 0)
print("Mixed-gender emotions training Accuracy: {0:.2%}".format(score[1]))

score = total_model.evaluate(x_test, y_test, verbose=0)
print("Mixed-gender emotions testing Accuracy: {0:.2%}".format(score[1]))
```

```
Mixed-gender emotions training Accuracy: 97.44%
Mixed-gender emotions testing Accuracy: 88.27%
```

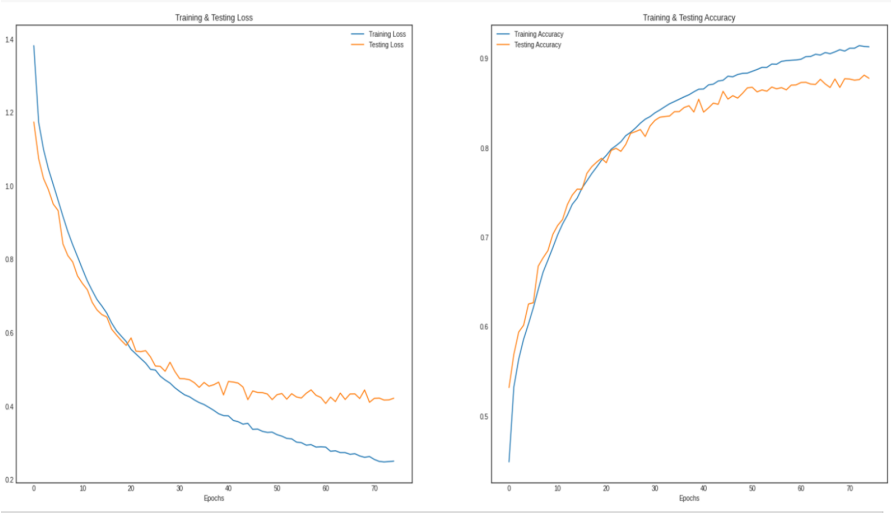
```
score = female_model.evaluate(x_trainF,y_trainF, verbose = 0)
print("Female emotions training Accuracy: {0:.2%}".format(score[1]))

score = female_model.evaluate(x_testF, y_testF, verbose=0)
print("Female emotions testing Accuracy: {0:.2%}".format(score[1]))
```

```
Female emotions training Accuracy: 99.33%
Female emotions testing Accuracy: 93.58%
```

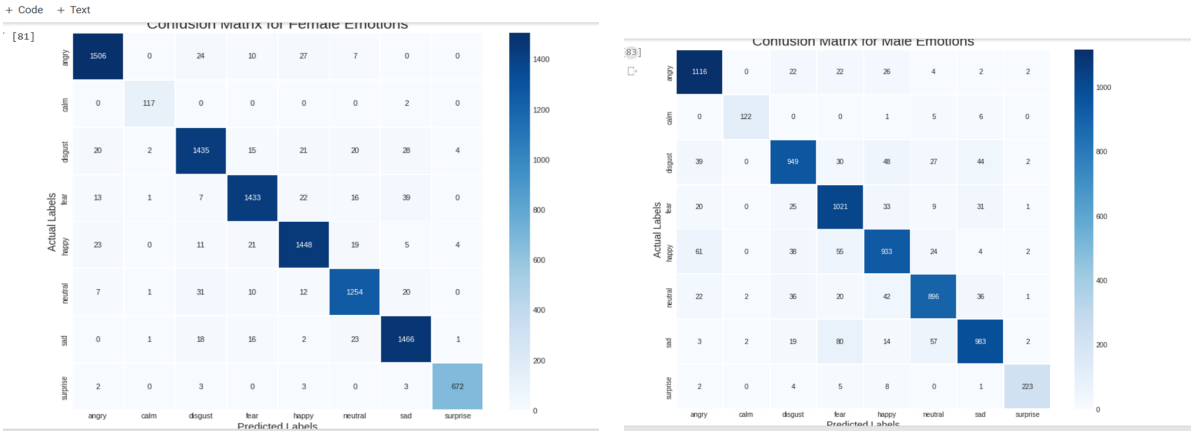
```
score = male_model.evaluate(x_trainM,y_trainM, verbose = 0)
print("Male emotions training Accuracy: {0:.2%}".format(score[1]))

score = male_model.evaluate(x_testM, y_testM, verbose=0)
print("Male emotions testing Accuracy: {0:.2%}".format(score[1]))
```

```
Male emotions training Accuracy: 98.83%
Male emotions testing Accuracy: 88.32%
```

Epochs vs Loss AND Epochs vs Accuracy Plot(s) :



Confusion matrix for Female and Male emotions (respectively) :



V. Outcome of the Project

The results of testing the trained model on the testing dataset for various emotions are summarized in the table below —

Model	Testing accuracy
Overall	87.80 %
Female	94.78%
Male	86.93%

This shows that Testing accuracy of female emotions are much more accurate than that of male emotions , depicted (via datasets).

VI. Conclusion

We tested for various emotions that can be revealed by students , especially in classroom scenarios. We drew out that the overall accuracy of the CNN (Convolutional Neural Network) model , trained and tested on the obtained dataset is 87.80%. It's pretty interesting to know that females depict more clear emotions than males , by showing testing accuracy values of 94.78% and 86.93%. Thus , efficaciousness of the model can clearly be seen , much more significant , on the female side of emotions.

VII. References

[1] A. B. Abdul Qayyum, A. Arefeen and C. Shahnaz, "Convolutional Neural Network (CNN) Based Speech-Emotion Recognition," 2019 IEEE International Conference on Signal Processing, Information, Communication & Systems (SPICSCON), 2019, pp. 122-125

[2] P. Harár, R. Burget and M. K. Dutta, "Speech emotion recognition with deep learning," *2017 4th International Conference on Signal Processing and Integrated Networks (SPIN), 2017,* pp. 137-140

[3] Murugan, Harini. (2020). Speech Emotion Recognition Using CNN. International Journal of Psychosocial Rehabilitation. 24. 10.37200/IJPR/V24I8/PR280260.

[4] J. Salamon and J. P. Bello, "Deep Convolutional Neural Networks and Data Augmentation for Environmental Sound Classification," in *IEEE Signal Processing Letters*, vol. 24, no. 3, pp. 279-283, March 2017.

[5] W. Han, C. F. Chan, C. S Choy, and K. P. Pun. An efficient MFCC extraction method in speech recognition. pages 145–148, 2006

[6] R. Hibare. Feature Extraction Techniques in Speech Processing : A Survey. *International Journal of Computer Applications*, 107(5):1–8, 2014.