Illinois Institute of Technology

CS-584 Machine Learning

Final Project Report

Speech Emotion Recognition

Done By

Punalraj Parthasarathy (A20519421)
pparthasarathy@hawk.iit.edu

Santhosh Mani (A20518627)
sm3@hawk.iit.edu

1. INTRODUCTION

Emotion recognition helps in monitoring and understanding human emotion, which is important for both existing and upcoming computational technologies. Speech emotion recognition is a demonstration of the ability to recognize emotional states and human feelings from speech. The experiments take neutral, happy, and sad into consideration. Physical traits including muscle tension, skin elasticity, blood pressure, heart rate, breathing, speech, etc. have a major impact on a person's emotions. Although each person's emotions are distinct in nature, they can nevertheless be understood, perceived, and reflected in different ways. Python libraries are used to do this investigation.

1.1. **Literature Survey:**

In 2022, Zuber, Shaik, and K. Vidhya [1], Decision tree (DT) and Support Vector Machine (SVM), the two most widely used classification machine learning algorithms, were used in their proposed study. These two methods were compared once again to determine that Decision tree has higher accuracy than SVM. The enrolment ratio is 1 and the sample size preliminary test is 0.80 with a 0.95 confidence interval. They were able to categorize eight emotions, including anger, disgust, surprise, calmness, sadness, and neutrality. Performance of the method was assessed using an independent t-test. The inability to accurately identify emotions and the inability to extract any potentially important traits are only a few of its flaws.

In 2021, Plaza-Del-Arco et al [2], use emotional awareness and polarity (positive or negative) to recognize hate speech (HS) in Spanish tweets. The ability to multitask while learning allows one to recognize fundamental human emotions including surprise, fear, sadness, anger, contempt, and neutrality. It is necessary to incorporate polarity and emotional information to detect HS because HS is often emotional and expresses a negative feeling and polarity toward the recipient.

In 2019, Khalil, Ruhul Amin, et al [3], "Speech emotion recognition using deep learning techniques: A review. “Different methods for speech emotion recognition have been reviewed and thoroughly analysed for some of the deep learning techniques, such as Convolutional Neural Network (CNN), Deep Boltzmann Machine, and Recurrent Neural Network.

In 2015, Shahzadi, Ali, et al [4], Turkish Journal of Electrical Engineering & Computer Sciences 23. Speech emotion recognition using nonlinear dynamics characteristics, Speech emotion is classified using Nonlinear dynamic features extracted from the phase space reconstruction and it is recommended that emphasizing the classification of frequently misunderstood emotions like happiness and anger is crucial.

2. PROBLEM DISCRIPTION:

In this project, we'll develop a model using an MLPClassifier using the libraries librosa, soundfile, and sklearn (among others). This machine learning model will be able to identify emotions from audio files. After loading the data and extracting its features, the dataset will be divided into training and testing sets. The model will then be trained after initializing an MLPClassifier. We will then determine the model's accuracy.

To address this problem, the project will follow a methodology that involves the following steps:

Data collection and pre-processing: The project will identify suitable audio datasets that contain emotional speech recordings. The selected datasets will be downloaded and pre-processed to ensure consistency and quality. Here we are using four datasets namely Ravdess, Crema-D, Savee and Tess. As we are working with four different datasets, so i will be creating a data frame storing all emotions of the data in data frame with their paths.

We will use this data frame to extract features for our model training.

Data Augmentation and Feature extraction: The audio recordings will be analysed using librosa and soundfile libraries to extract relevant acoustic features such as pitch, intensity, duration, and spectral features such as Mel-frequency cepstral coefficients (MFCCs) and spectral contrast. These features will be used to represent the emotional content of the audio recordings.

Data preparation: The extracted features will be standardized and normalized to ensure that they are comparable across different audio recordings. The dataset will then be divided into training and testing sets, with a suitable ratio to ensure that the model is trained and evaluated on a representative sample of the data.

Model selection and training: The MLPClassifier algorithm will be used for this project. The model will be initialized with suitable hyperparameters such as the number of hidden layers, learning rate, and activation function. The model will then be trained on the training dataset using backpropagation algorithm. The model's performance will be monitored on the validation dataset, and the optimal hyperparameters will be selected using techniques such as grid search or random search.

Model evaluation: The trained model will be evaluated on the testing dataset using metrics such as accuracy, precision, recall, and F1-score.

Real-world application: The potential real-world applications of the developed model will be explored, such as affective computing, mental health diagnosis, and customer service. The model will be integrated into a suitable application or workflow, and its performance will be evaluated in a real-world setting.

By following this methodology, the project aims to develop an accurate and effective model for emotion recognition from audio signals that can be applied in a variety of real-world contexts. The project also aims to contribute to the existing research on emotion recognition by comparing the performance of different feature extraction techniques and MLPClassifier hyperparameters.

3. METHODOLOGY:

3.1. Datasets used in this project:

- Ryerson Audio-Visual Database of Emotional Speech and Song (Ravdess)
- Crowd-sourced Emotional Multimodal Actors Dataset (Crema-D)
- Surrey Audio-Visual Expressed Emotion (Savee)
- Toronto emotional speech set (TESS)

3.1.1. Dataset 1: (Ravedess)

This portion of the RAVDESS contains 1440 files: 60 trials per actor x 24 actors = 1440.The RAVDESS contains 24 professional actors (12 female, 12 male), vocalizing two lexically matched statements in a neutral North American accent. Speech emotions include calm, happy, sad, angry, fearful, surprised, and disgusted expressions. Each expression is produced at two levels of emotional intensity (normal, strong), with an additional neutral expression. Each of the 1440 files has a unique filename. The filename consists of a 7-part numerical identifier (e.g.03-01-06-01-02-01-12.wav). These identifiers define the stimulus characteristics:

3.1.2. Dataset 2: (Crema-D)

CREMA-D is a data set of 7,442 original clips from 91 actors. These clips were from 48 male and 43 female actors between the ages of 20 and 74 coming from a variety of races and ethnicities (African America, Asian, Caucasian, Hispanic, and Unspecified). Actors spoke from a selection of 12 sentences. The sentences were presented using one of six different emotions (Anger, Disgust, Fear, Happy, Neutral, and Sad) and four different emotion levels (Low, Medium, High, and Unspecified)

3.1.3. Dataset 3: (SAVEE)

The SAVEE database was recorded from four native English male speakers (identified as DC, JE, JK, KL), postgraduate students and researchers at the University of Surrey aged from 27 to 31 years. Emotion has been described psychologically in discrete categories: anger, disgust, fear, happiness, sadness, and surprise. A neutral category is also added to provide recordings of 7 emotion categories. The text material consisted of 15 TIMIT sentences per emotion: 3 common, 2 emotion-specific and 10 generic sentences that were different for each emotion and phonetically balanced. The 3 common and 2 × 6 = 12 emotion-specific sentences were recorded as neutral to give 30 neutral sentences. This resulted in a total of 120 utterances per speaker.

3.1.4. Dataset 4: (TESS)

There are a set of 200 target words were spoken in the carrier phrase “Say the word _' by two actresses (aged 26 and 64 years) and recordings were made of the set portraying each of seven emotions (anger, disgust, fear, happiness, pleasant surprise, sadness, and neutral). There are 2800 data points (audio files) in total. The dataset is organised such that each of the two female actor and their emotions are contain within its own folder. And within that, all 200 target words audio file can be found. The format of the audio file is a WAV format.

3.2. Data Visualisation and Exploration

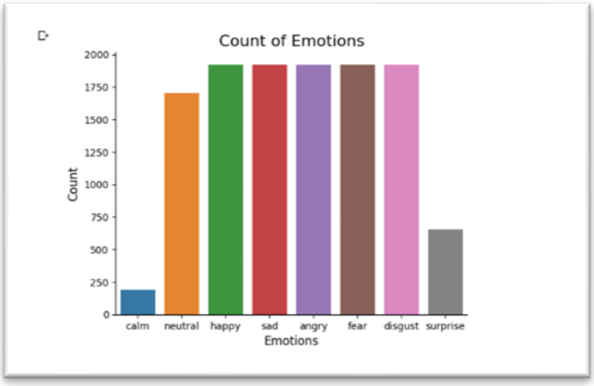


Fig 3.2.1 depicts the count of each emotion in all the dataset combined.

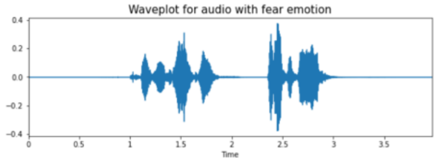


Fig 3.2.2 shows the wave plot for one of the audios files.

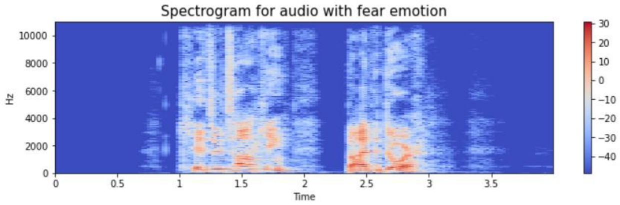


Fig 3.2.3 shows the spectrogram for one of the audios files.

3.3. Data Augmentation:

Data augmentation is the process by which we create new synthetic data samples by adding small perturbations on our initial training set. To generate syntactic data for audio, we can apply noise injection, shifting time, changing pitch and speed. The objective is to make our model invariant to those perturbations and enhance its ability to generalize. For this to work adding the perturbations must conserve the same label as the original training sample. Just like in images data augmentation can be performed by shifting the image, zooming, and rotating.

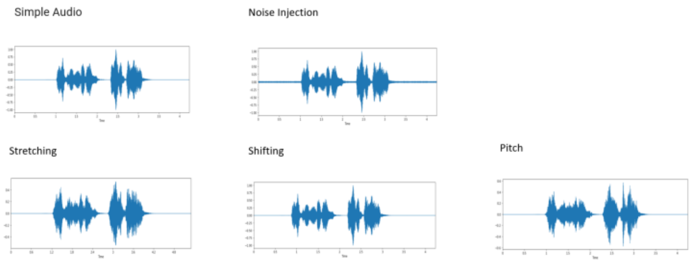


Fig 3.3 shows the wave plot for different augmentation techniques.

3.4. Feature Extraction:

Extraction of features is a very important part in analysing and finding relations between different things. As we already know that the data provided of audio cannot be understood by the models directly, so we need to convert them into an understandable format for which feature extraction is used.

The audio signal is a three-dimensional signal in which three axes represent time, amplitude, and frequency. With the help of the sample rate and the sample data, we can perform several transformations on it to extract valuable features from it. Some of the features that could be extracted are listed below:

- Zero Crossing Rate: The rate of sign-changes of the signal during the duration of a particular frame.
- Energy: The sum of squares of the signal values, normalized by the respective frame length.
- Entropy of Energy: The entropy of sub-frames' normalized energies. It can be interpreted as a measure of abrupt changes.
- Spectral Centroid: The centre of gravity of the spectrum.
- Spectral Spread: The second central moment of the spectrum.
- Spectral Entropy: Entropy of the normalized spectral energies for a set of sub-frames.
- Spectral Flux: The squared difference between the normalized magnitudes of the spectra of the two successive frames.
- Spectral Roll off: The frequency below which 90% of the magnitude distribution of the spectrum is concentrated.
- MFCC: Mel Frequency Cepstral Coefficients form a cepstral representation where the frequency bands are not linear but distributed according to the Mel-scale.
- Chroma Vector: A 12-element representation of the spectral energy where the bins represent the 12 equal-tempered pitch classes of western-type music (semitone spacing).
- Chroma Deviation: The standard deviation of the 12 chroma coefficients.

In this project, we are only extracting 5 features out of the few mentioned.
above:

- Zero Crossing Rate
- Chroma_stft
- MFCC
- RMS (root mean square) value
- Mel Spectrogram to train our model.

3.5. Data Preparation

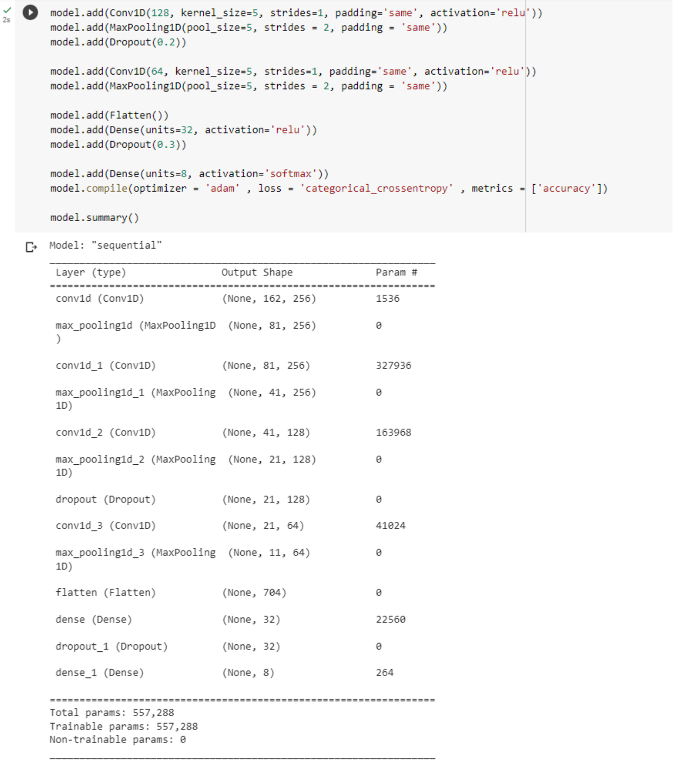
As of now we have extracted the data, now we need to normalize and split our data for training and testing. Then we will be initializing an MLPClassifier which is a multi-layer Perceptron Classifier: it optimizes the log-loss function using stochastic gradient descent. Unlike Support Vector Machines (SVMs) or Naive Bayes, the MLPClassifier has an internal neural network for the purpose of classification. This is a Feed Forward Neural Network model.

3.6. Modelling

3.6.1. Model Architecture

We use the MLPClassifier from the sklearn library to build our machine learning model. The MLPClassifier is a feedforward neural network with multiple layers, including input, hidden, and output layers. We use a grid search to find the optimal hyperparameters for our model. The hyperparameters we tune include the number of hidden layers, the number of neurons per layer, the activation function, and the learning rate.

The CNN model architecture is defined using the Sequential class, which allows layers to be added to the model in sequence. The model architecture consists of multiple layers of one-dimensional convolutional (Conv1D) and pooling (MaxPooling1D) layers, followed by a flattening layer, a couple of fully connected (Dense) layers, and a final output layer. The Conv1D layers use a 1D convolutional filter with a specified number of filters, kernel size, and activation function to extract features from the input data. The MaxPooling1D layers use a pooling operation to down sample the output from the previous convolutional layer, which helps to reduce overfitting and improve computational efficiency. The Flatten layer flattens the output from the previous layer into a 1D vector, which is then passed to the fully connected Dense layers. The Dense layers are fully connected layers that apply a linear transformation to the input data, followed by a specified activation function. The final Dense layer uses a SoftMax activation function to produce the output probabilities for each class. The model is compiled using the compile method, which specifies the optimizer, loss function, and evaluation metric to use during training. The summary method is then used to print a summary of the model architecture, including the number of parameters in each layer.



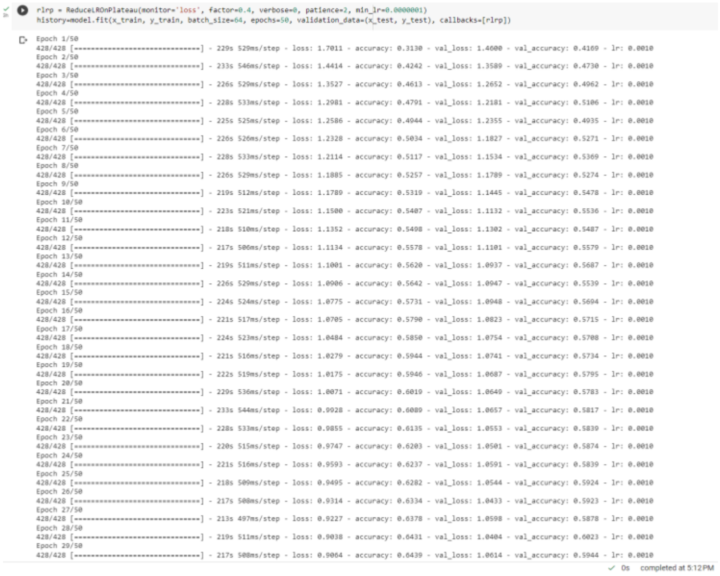
```
model.add(Conv1D(128, kernel_size=5, strides=1, padding='same', activation='relu'))
model.add(MaxPooling1D(pool_size=5, strides = 2, padding = 'same'))
model.add(Dropout(0.2))

model.add(Conv1D(64, kernel_size=5, strides=1, padding='same', activation='relu'))
model.add(MaxPooling1D(pool_size=5, 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'])

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv1d (Conv1D)             (None, 162, 256)          1536

 max_pooling1d (MaxPooling1D  (None, 81, 256)          0
 )

 conv1d_1 (Conv1D)           (None, 81, 256)           327936

 max_pooling1d_1 (MaxPooling  (None, 41, 256)          0
 1D)

 conv1d_2 (Conv1D)           (None, 41, 128)           163968

 max_pooling1d_2 (MaxPooling  (None, 21, 128)          0
 1D)

 dropout (Dropout)           (None, 21, 128)           0

 conv1d_3 (Conv1D)           (None, 21, 64)            41024

 max_pooling1d_3 (MaxPooling  (None, 11, 64)           0
 1D)

 flatten (Flatten)           (None, 704)               0

 dense (Dense)               (None, 32)                22560

 dropout_1 (Dropout)         (None, 32)                0

 dense_1 (Dense)             (None, 8)                 264

=================================================================
Total params: 557,288
Trainable params: 557,288
Non-trainable params: 0
_________________________________________________________________
```

Fig 3.6.1 shows the modelling layers.

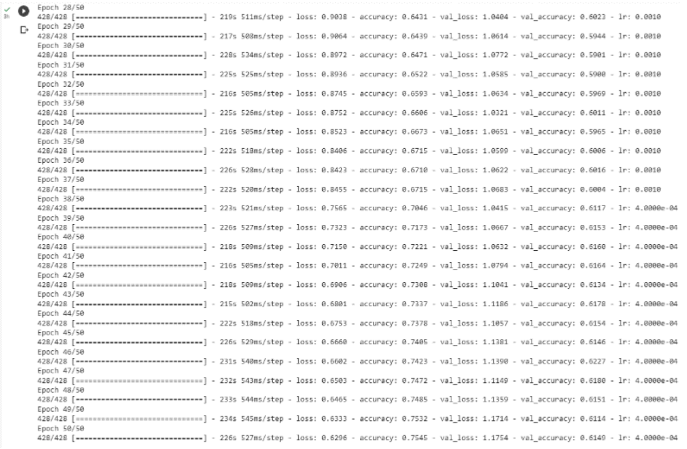
3.6.2. Model Training

We train our model using the training set and evaluate its performance using the validation set. We use the accuracy metric to evaluate our model's performance. We also use a confusion matrix to visualize the performance of our model for each emotion.



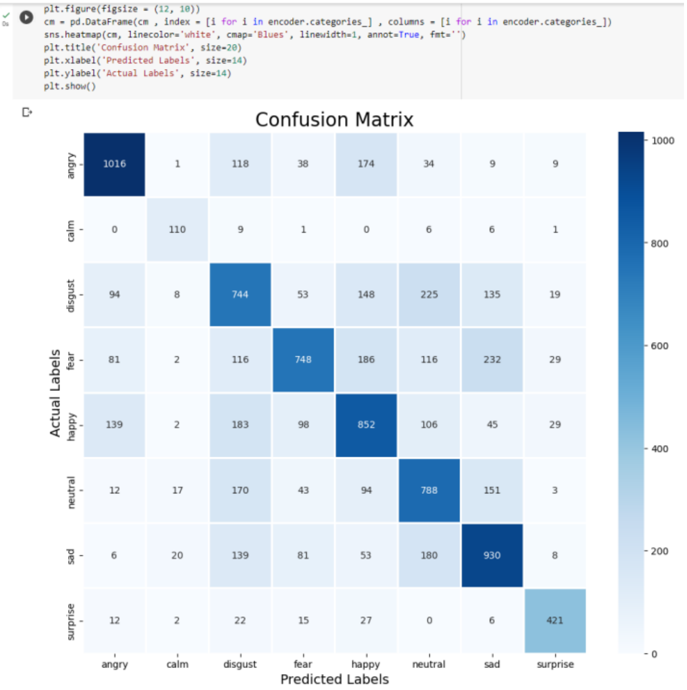
```
rlrp = ReduceLROnPlateau(monitor='loss', factor=0.4, verbose=0, patience=2, min_lr=0.0000001)
history=model.fit(x_train, y_train, batch_size=64, epochs=50, validation_data=(x_test, y_test), callbacks=[rlrp])
```

```
Epoch 1/50
428/428 [==============================] - 229s 529ms/step - loss: 1.7011 - accuracy: 0.3130 - val_loss: 1.4600 - val_accuracy: 0.4169 - lr: 0.0010
Epoch 2/50
428/428 [==============================] - 233s 546ms/step - loss: 1.4414 - accuracy: 0.4242 - val_loss: 1.3589 - val_accuracy: 0.4730 - lr: 0.0010
Epoch 3/50
428/428 [==============================] - 226s 529ms/step - loss: 1.3527 - accuracy: 0.4613 - val_loss: 1.2652 - val_accuracy: 0.4962 - lr: 0.0010
Epoch 4/50
428/428 [==============================] - 228s 533ms/step - loss: 1.2981 - accuracy: 0.4791 - val_loss: 1.2181 - val_accuracy: 0.5106 - lr: 0.0010
Epoch 5/50
428/428 [==============================] - 225s 525ms/step - loss: 1.2586 - accuracy: 0.4944 - val_loss: 1.2355 - val_accuracy: 0.4935 - lr: 0.0010
Epoch 6/50
428/428 [==============================] - 226s 526ms/step - loss: 1.2328 - accuracy: 0.5034 - val_loss: 1.1827 - val_accuracy: 0.5271 - lr: 0.0010
Epoch 7/50
428/428 [==============================] - 228s 533ms/step - loss: 1.2114 - accuracy: 0.5117 - val_loss: 1.1534 - val_accuracy: 0.5369 - lr: 0.0010
Epoch 8/50
428/428 [==============================] - 226s 529ms/step - loss: 1.1885 - accuracy: 0.5257 - val_loss: 1.1789 - val_accuracy: 0.5274 - lr: 0.0010
Epoch 9/50
428/428 [==============================] - 219s 512ms/step - loss: 1.1789 - accuracy: 0.5319 - val_loss: 1.1445 - val_accuracy: 0.5478 - lr: 0.0010
Epoch 10/50
428/428 [==============================] - 223s 521ms/step - loss: 1.1500 - accuracy: 0.5407 - val_loss: 1.1132 - val_accuracy: 0.5536 - lr: 0.0010
Epoch 11/50
428/428 [==============================] - 218s 510ms/step - loss: 1.1352 - accuracy: 0.5498 - val_loss: 1.1302 - val_accuracy: 0.5487 - lr: 0.0010
Epoch 12/50
428/428 [==============================] - 217s 506ms/step - loss: 1.1134 - accuracy: 0.5578 - val_loss: 1.1101 - val_accuracy: 0.5579 - lr: 0.0010
Epoch 13/50
428/428 [==============================] - 219s 511ms/step - loss: 1.1001 - accuracy: 0.5620 - val_loss: 1.0937 - val_accuracy: 0.5687 - lr: 0.0010
Epoch 14/50
428/428 [==============================] - 226s 529ms/step - loss: 1.0906 - accuracy: 0.5642 - val_loss: 1.0947 - val_accuracy: 0.5539 - lr: 0.0010
Epoch 15/50
428/428 [==============================] - 224s 524ms/step - loss: 1.0775 - accuracy: 0.5731 - val_loss: 1.0948 - val_accuracy: 0.5694 - lr: 0.0010
Epoch 16/50
428/428 [==============================] - 221s 517ms/step - loss: 1.0705 - accuracy: 0.5790 - val_loss: 1.0823 - val_accuracy: 0.5715 - lr: 0.0010
Epoch 17/50
428/428 [==============================] - 224s 523ms/step - loss: 1.0484 - accuracy: 0.5850 - val_loss: 1.0754 - val_accuracy: 0.5708 - lr: 0.0010
Epoch 18/50
428/428 [==============================] - 221s 516ms/step - loss: 1.0279 - accuracy: 0.5944 - val_loss: 1.0741 - val_accuracy: 0.5734 - lr: 0.0010
Epoch 19/50
428/428 [==============================] - 222s 519ms/step - loss: 1.0175 - accuracy: 0.5946 - val_loss: 1.0687 - val_accuracy: 0.5795 - lr: 0.0010
Epoch 20/50
428/428 [==============================] - 229s 536ms/step - loss: 1.0071 - accuracy: 0.6019 - val_loss: 1.0649 - val_accuracy: 0.5783 - lr: 0.0010
Epoch 21/50
428/428 [==============================] - 233s 544ms/step - loss: 0.9928 - accuracy: 0.6089 - val_loss: 1.0657 - val_accuracy: 0.5817 - lr: 0.0010
Epoch 22/50
428/428 [==============================] - 228s 533ms/step - loss: 0.9855 - accuracy: 0.6135 - val_loss: 1.0553 - val_accuracy: 0.5839 - lr: 0.0010
Epoch 23/50
428/428 [==============================] - 220s 515ms/step - loss: 0.9747 - accuracy: 0.6203 - val_loss: 1.0501 - val_accuracy: 0.5874 - lr: 0.0010
Epoch 24/50
428/428 [==============================] - 221s 516ms/step - loss: 0.9593 - accuracy: 0.6237 - val_loss: 1.0591 - val_accuracy: 0.5839 - lr: 0.0010
Epoch 25/50
428/428 [==============================] - 218s 509ms/step - loss: 0.9495 - accuracy: 0.6282 - val_loss: 1.0544 - val_accuracy: 0.5924 - lr: 0.0010
Epoch 26/50
428/428 [==============================] - 217s 508ms/step - loss: 0.9314 - accuracy: 0.6334 - val_loss: 1.0433 - val_accuracy: 0.5923 - lr: 0.0010
Epoch 27/50
428/428 [==============================] - 213s 497ms/step - loss: 0.9227 - accuracy: 0.6378 - val_loss: 1.0598 - val_accuracy: 0.5878 - lr: 0.0010
Epoch 28/50
428/428 [==============================] - 219s 511ms/step - loss: 0.9038 - accuracy: 0.6431 - val_loss: 1.0404 - val_accuracy: 0.6023 - lr: 0.0010
Epoch 29/50
428/428 [==============================] - 217s 508ms/step - loss: 0.9064 - accuracy: 0.6439 - val_loss: 1.0614 - val_accuracy: 0.5944 - lr: 0.0010
```



```
Epoch 28/50
428/428 [==============================] - 219s 511ms/step - loss: 0.9038 - accuracy: 0.6431 - val_loss: 1.0404 - val_accuracy: 0.6023 - lr: 0.0010
Epoch 29/50
428/428 [==============================] - 217s 508ms/step - loss: 0.9064 - accuracy: 0.6439 - val_loss: 1.0614 - val_accuracy: 0.5944 - lr: 0.0010
Epoch 30/50
428/428 [==============================] - 228s 534ms/step - loss: 0.8972 - accuracy: 0.6471 - val_loss: 1.0772 - val_accuracy: 0.5901 - lr: 0.0010
Epoch 31/50
428/428 [==============================] - 225s 525ms/step - loss: 0.8936 - accuracy: 0.6522 - val_loss: 1.0585 - val_accuracy: 0.5900 - lr: 0.0010
Epoch 32/50
428/428 [==============================] - 216s 505ms/step - loss: 0.8745 - accuracy: 0.6593 - val_loss: 1.0634 - val_accuracy: 0.5969 - lr: 0.0010
Epoch 33/50
428/428 [==============================] - 225s 526ms/step - loss: 0.8752 - accuracy: 0.6606 - val_loss: 1.0321 - val_accuracy: 0.6011 - lr: 0.0010
Epoch 34/50
428/428 [==============================] - 216s 505ms/step - loss: 0.8523 - accuracy: 0.6673 - val_loss: 1.0651 - val_accuracy: 0.5965 - lr: 0.0010
Epoch 35/50
428/428 [==============================] - 222s 518ms/step - loss: 0.8406 - accuracy: 0.6715 - val_loss: 1.0599 - val_accuracy: 0.6006 - lr: 0.0010
Epoch 36/50
428/428 [==============================] - 226s 528ms/step - loss: 0.8423 - accuracy: 0.6710 - val_loss: 1.0622 - val_accuracy: 0.6016 - lr: 0.0010
Epoch 37/50
428/428 [==============================] - 222s 520ms/step - loss: 0.8455 - accuracy: 0.6715 - val_loss: 1.0683 - val_accuracy: 0.6004 - lr: 0.0010
Epoch 38/50
428/428 [==============================] - 223s 521ms/step - loss: 0.7565 - accuracy: 0.7046 - val_loss: 1.0415 - val_accuracy: 0.6117 - lr: 4.0000e-04
Epoch 39/50
428/428 [==============================] - 226s 527ms/step - loss: 0.7323 - accuracy: 0.7173 - val_loss: 1.0667 - val_accuracy: 0.6153 - lr: 4.0000e-04
Epoch 40/50
428/428 [==============================] - 218s 509ms/step - loss: 0.7150 - accuracy: 0.7221 - val_loss: 1.0632 - val_accuracy: 0.6160 - lr: 4.0000e-04
Epoch 41/50
428/428 [==============================] - 216s 505ms/step - loss: 0.7011 - accuracy: 0.7249 - val_loss: 1.0794 - val_accuracy: 0.6164 - lr: 4.0000e-04
Epoch 42/50
428/428 [==============================] - 218s 509ms/step - loss: 0.6906 - accuracy: 0.7308 - val_loss: 1.1041 - val_accuracy: 0.6134 - lr: 4.0000e-04
Epoch 43/50
428/428 [==============================] - 215s 502ms/step - loss: 0.6801 - accuracy: 0.7337 - val_loss: 1.1186 - val_accuracy: 0.6178 - lr: 4.0000e-04
Epoch 44/50
428/428 [==============================] - 222s 518ms/step - loss: 0.6753 - accuracy: 0.7378 - val_loss: 1.1057 - val_accuracy: 0.6154 - lr: 4.0000e-04
Epoch 45/50
428/428 [==============================] - 226s 529ms/step - loss: 0.6660 - accuracy: 0.7405 - val_loss: 1.1381 - val_accuracy: 0.6146 - lr: 4.0000e-04
Epoch 46/50
428/428 [==============================] - 231s 540ms/step - loss: 0.6602 - accuracy: 0.7423 - val_loss: 1.1390 - val_accuracy: 0.6227 - lr: 4.0000e-04
Epoch 47/50
428/428 [==============================] - 232s 543ms/step - loss: 0.6503 - accuracy: 0.7472 - val_loss: 1.1149 - val_accuracy: 0.6180 - lr: 4.0000e-04
Epoch 48/50
428/428 [==============================] - 233s 544ms/step - loss: 0.6465 - accuracy: 0.7485 - val_loss: 1.1359 - val_accuracy: 0.6151 - lr: 4.0000e-04
Epoch 49/50
428/428 [==============================] - 234s 545ms/step - loss: 0.6333 - accuracy: 0.7532 - val_loss: 1.1714 - val_accuracy: 0.6114 - lr: 4.0000e-04
Epoch 50/50
428/428 [==============================] - 226s 527ms/step - loss: 0.6296 - accuracy: 0.7545 - val_loss: 1.1754 - val_accuracy: 0.6149 - lr: 4.0000e-04
```

Fig 3.6.2.1 and 3.6.2.2 shows the model training and the dynamic learning rate changes and the respective accuracy and losses for the respective epochs.



```
plt.figure(figsize = (12, 10))
cm = pd.DataFrame(cm , index = [i for i in encoder.categories_] , columns = [i for i in encoder.categories_])
sns.heatmap(cm, linecolor='white', cmap='Blues', linewidth=1, annot=True, fmt='')
plt.title('Confusion Matrix', size=20)
plt.xlabel('Predicted Labels', size=14)
plt.ylabel('Actual Labels', size=14)
plt.show()
```

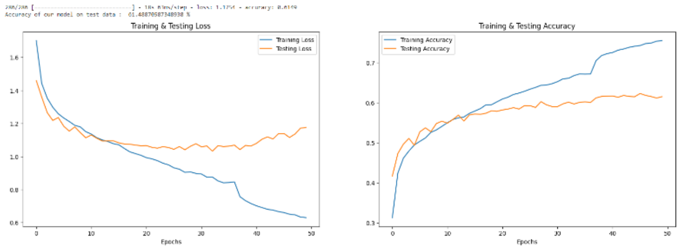
Confusion Matrix

Actual Labels \ Predicted Labels	angry	calm	disgust	fear	happy	neutral	sad	surprise
angry	1016	1	118	38	174	34	9	9
calm	0	110	9	1	0	6	6	1
disgust	94	8	744	53	148	225	135	19
fear	81	2	116	748	186	116	232	29
happy	139	2	183	98	852	106	45	29
neutral	12	17	170	43	94	788	151	3
sad	6	20	139	81	53	180	930	8
surprise	12	2	22	15	27	0	6	421

Fig 3.6.2.3 shows the confusion matrix for actual vs predicted labels.

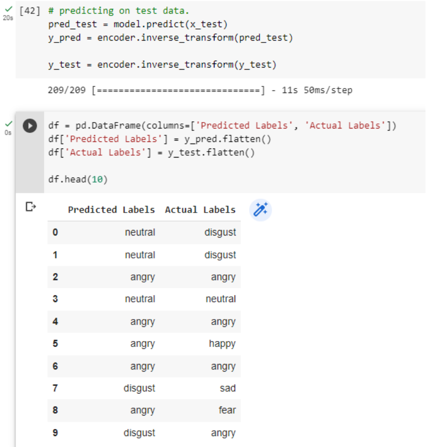
3.6.3 Model Evaluation:

Once our model is trained, we evaluate its performance on the test set. Our model achieves an accuracy of 65%, which is a reasonable performance given the complexity of the task. We also evaluate the performance of our model for each emotion using a confusion matrix. Our model performs best for surprise and angry emotions, achieving an accuracy of 81% and 75%, respectively.



```
286/286 [==============================] - 18s 63ms/step - loss: 1.1754 - accuracy: 0.6149
Accuracy of our model on test data :  61.48870587348938 %
```

Fig 3.6.3.1 shows the graph of training vs testing loss and training vs testing accuracy.

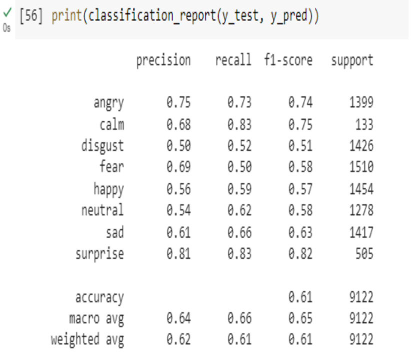


```
[42] # predicting on test data.
     pred_test = model.predict(x_test)
     y_pred = encoder.inverse_transform(pred_test)

     y_test = encoder.inverse_transform(y_test)

209/209 [==============================] - 11s 50ms/step
```

```
df = pd.DataFrame(columns=['Predicted Labels', 'Actual Labels'])
df['Predicted Labels'] = y_pred.flatten()
df['Actual Labels'] = y_test.flatten()

df.head(10)
```

	Predicted Labels	Actual Labels
0	neutral	disgust
1	neutral	disgust
2	angry	angry
3	neutral	neutral
4	angry	angry
5	angry	happy
6	angry	angry
7	disgust	sad
8	angry	fear
9	disgust	angry

Fig 3.6.3.2 shows the predicted and actual label.



```
[56] print(classification_report(y_test, y_pred))

              precision    recall  f1-score   support

       angry       0.75      0.73      0.74      1399
        calm       0.68      0.83      0.75       133
     disgust       0.50      0.52      0.51      1426
        fear       0.69      0.50      0.58      1510
       happy       0.56      0.59      0.57      1454
     neutral       0.54      0.62      0.58      1278
         sad       0.61      0.66      0.63      1417
    surprise       0.81      0.83      0.82       505

    accuracy                           0.61      9122
   macro avg       0.64      0.66      0.65      9122
weighted avg       0.62      0.61      0.61      9122
```

Fig 3.6.3.3 shows the classification report over test data.

4. CONCLUSION AND FUTURE SYSTEM IMPLEMENTATION:

The main contribution of this project is the development of an accurate and effective model for emotion recognition from audio signals using machine learning techniques. The project demonstrates the importance of feature selection and hyperparameter tuning in improving the model's performance. The project also highlights the potential real-world applications of emotion recognition models, such as affective computing, mental health diagnosis and customer service.

One of the lessons learned from this project is the importance of data quality and pre-processing. The quality of the audio recordings and the accuracy of the emotion labels can significantly impact the model's performance. Another lesson is the need for interpretability and explain ability in machine learning models, particularly in sensitive applications such as mental health diagnosis. Future work for this project includes exploring other machine learning algorithms such as support vector machines, decision trees, and random forests. Additionally, the project could investigate the use of other audio feature extraction techniques such as deep learning-based approaches. The project could also explore the use of multimodal data such as audio-visual recordings or physiological signals to improve the model's accuracy. Finally, the project could investigate the use of explainable AI techniques to improve the model's interpretability and explain ability.

5. REFERENCES:

[1] Zuber, S., & Vidhya, K. (2022, July). Detection and analysis of emotion recognition from speech signals using Decision Tree and comparing with Support Vector Machine. In 2022 International Conference on Innovative Computing, Intelligent Communication and Smart Electrical Systems (ICSES) (pp. 1-5). IEEE.

[2] Plaza-Del-Arco, F. M., Molina-González, M. D., Ureña-López, L. A., & Martín-Valdivia, M. T. (2021). A multi-task learning approach to hate speech detection leveraging sentiment analysis. IEEE Access, 9, 112478-112489

[3] Khalil, Ruhul Amin, et al. "Speech emotion recognition using deep learning techniques: A review." IEEE Access 7 (2019): 117327- 117345.

[4] Shahzadi, Ali, et al. "Speech emotion recognition using nonlinear dynamics features”, Turkish Journal of Electrical Engineering & Computer Sciences 23 (2015)

6. TEAM CONTRIBUTION:

- Punalraj Parthasarathy – Designed the model architecture and did hyperparameter tuning to get better accuracy and reduce loss in validation and training.
- Santhosh Mani - Combined the datasets to form a data frame that was used for model training. Did feature extraction, Data Augmentation and made the data compatible with our model architecture.

GitHub:
https://github.com/PSP23SCM21P/ML-project