SPEECH EMOTION RECOGNITION

About

The task aims to analyze the work on Speech Emotion Recognition model, study about the features of sound and experimenting with different techniques. The notebook I considered for the above task is by Shivam Burnwal. This contains the work on 4 different datasets that are RAVDEES, CREMA, TESS and SAVEE. The goal is to categorize the input audio to one of following 8 features. Neutral, angry, happy, sad, fear, surprised, disgust and surprised.

Following steps are being followed and analyzed step by step:

Feature Extraction:

After concatenating all labels with their path, features are extracted from each audio file.

Some terminologies are:

Frame: A single unit of data that represents a small chunk of an audio signal.

Frequency: The number of cycles a wave completes in one second, measured in Hertz (Hz), representing the pitch of a sound.

Amplitude: The height of a sound wave, indicating the loudness or intensity of the sound.

Sample Rate: The number of audio samples captured per second, measured in Hertz (Hz), determining the audio quality.

Pitch: The perceived highness or lowness of a sound, which is determined by the frequency of the sound wave.

Features include:

- Zero Crossing Rate (ZCR):

  It is the rate at which the audio signal crosses the zero line. By that it can be interpreted that it is rate at which the signal changes its sign.

  How Useful:

  ZCR can distinguish between a sound with higher frequency and one with lower frequency. Higher the frequency, higher the number of fluctuations, crest and troughs in a frame. Noisy or shouting sounds would have more ZCR while the normal sound would have less ZCR due to less fluctuations.

- Chroma Short-Time Fourier Transform (Chroma_stft):

  It represents the energy distribution across 12 different pitch classes. STFT is calculated for each of frame and that energy is mapped to the classes.

  How Useful:

  It helps to distinguish between the different pitches in a frame, that could help in analyzing different musical notes and variations. Since there are different pitches for every emotion, therefore it can be used to gather emotional information. Like the louder sound has high pitch and hence the energy than soft/sad sounds.

- Mel-Frequency Cepstral Coefficients (MFCC):

  It captures the way human ears would hear a particular sound. It works by calculating the frequency per frames and comparing them with a scale called MEL scale which shows how actually the ears percept a sound. So, the frequencies that are more relevant to what human ears emphasize are noted.

  How Useful:

  It helps to focus on important frequencies rather than all of them.

- Root Mean Square (RMS):

  It is a measure of the average power of the audio signal over a frame, reflecting its loudness. Higher the value of RMS, higher the loudness of sound.

  How Useful:

  Since it denotes the loudness of sound, it can denote the emotional state as well. Louder sound depicts anger or excitement. Softer one can be neutral.

Data Augmentation:

Augmentation means the process of increasing the size, value, or quality of something by adding to it. In ML, it means expanding size of dataset by applying several transformations on dataset.

In case of audio, there can be different techniques that can be applied on data to transform it. Among them, some are:

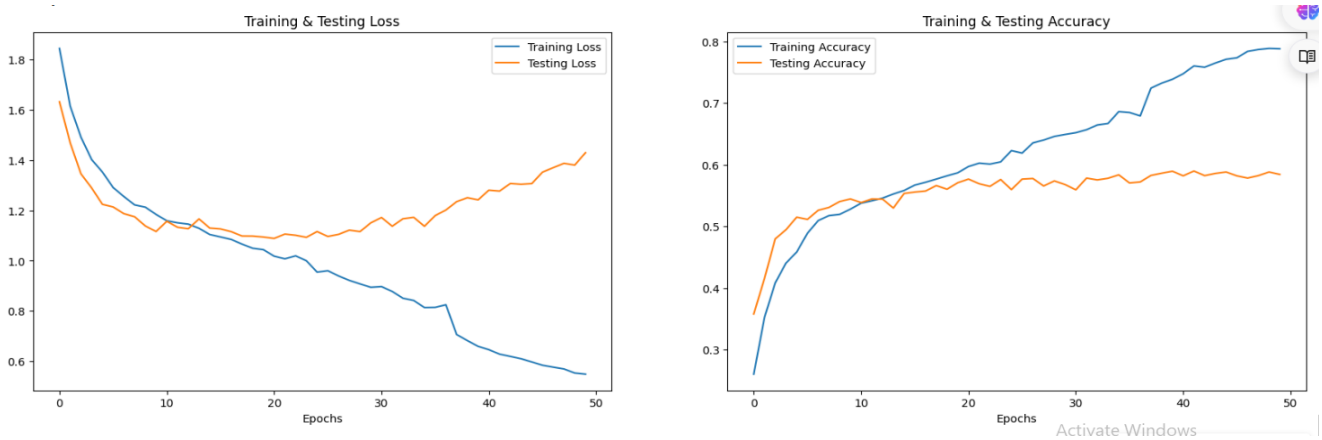
- Adding noise to sound: Since audio is collection of amplitude (a number) recorded within a second. 0 would mean complete silence. So, by adding random numbers to every value, the original audio can be made noisy.

- Pitch Shifting: It includes changing the pitch of audio without changing the duration. Since the frequency is increased, the audio sounds like words are a little stretched and pronounced louder than actual one but the speed as well as amplitude remain same.
- Time Stretching: It includes changing the speed of audio without altering the pitch.

Benefits:

1. Data augmentation can reduce chances of overfitting. This means that model does not learns all the patterns of features as it is. This memorization would cause overfitting. The problem that arises when model is memorizing the patterns but fails on testing data because it has learned not generalized.
2. It increases the size and diversity of data.
3. It would make model learn the patterns so good that would help to give good results even on distorted testing data.

Analysis on Data Augmentation:



BEFORE AUGMENTATION

MODELS:

First step after extracting features is to separate the features and labels. The labels which are categorical are converted to numbers using One Hot Encoding.

One Hot Encoding: The first step involves getting the unique classes in labels. In our case there are 8 different classes. This technique makes 8 different vectors, instead of directly assigning numbers to classes. This is because the numbers signify the ordinality (rank) of features, while vectors are just points that are used for representing nominal data. It is somehow a sparse vector (vector with most entries 0) where the only 1 in vector of zeros is representation of a unique value.

Standard Scaler: A standard scaler is used on features to normalize the data. It is done by subtracting mean from each value divided by standard deviation. Normalized data has mean equals to 0 and standard deviation =1. It is used to ensure that all features are on same scale, hence no one dominated based on the magnitude.

Following model architectures are tried with their comparisons written aside:

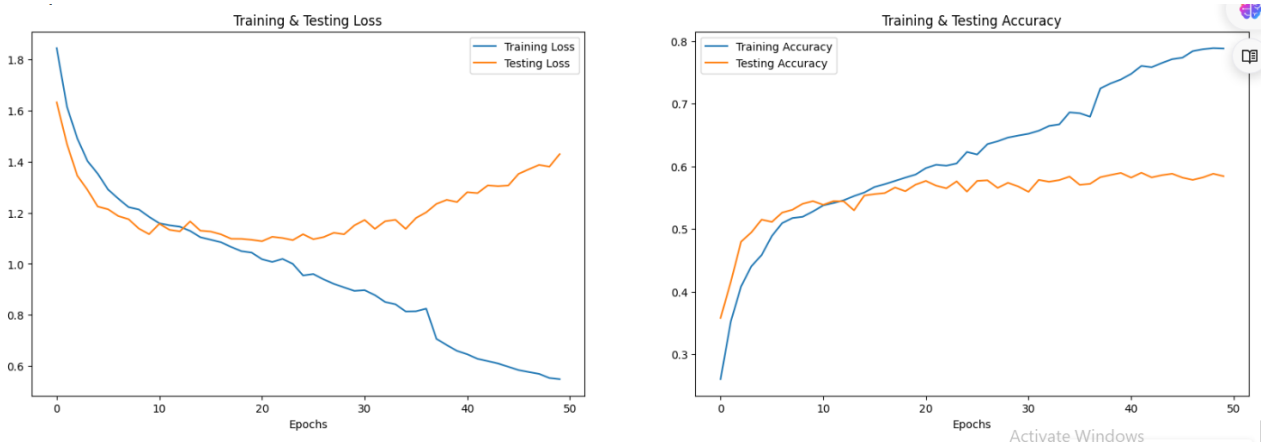
CNN:

1. The initial architecture made in reference notebook was CNN. CNNs are effective in capturing local patterns from input data, especially in the frequency domain. For tasks like Speech Emotion Recognition (SER), CNNs are ideal for identifying spatial features in spectrograms, such as pitch, tone, and intensity changes. Every Convolutional layer is followed by a Max Pooling layer. It is used to lower the dimensions of output of Convolutional layer by taking maximum value from each window. Thus, retaining important information only and reducing dimensions.

The first architecture was the original from reference notebook but without data augmentation:

```
model=Sequential()
model.add(Conv1D(256, kernel_size=5, strides=1, padding='same', activation='relu', input_shape=(x_train.shape[1], 1)))
model.add(MaxPooling1D(pool_size=5, strides = 2, padding = 'same'))

model.add(Conv1D(256, kernel_size=5, strides=1, padding='same', activation='relu'))
model.add(MaxPooling1D(pool_size=5, strides = 2, padding = 'same'))

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()
```

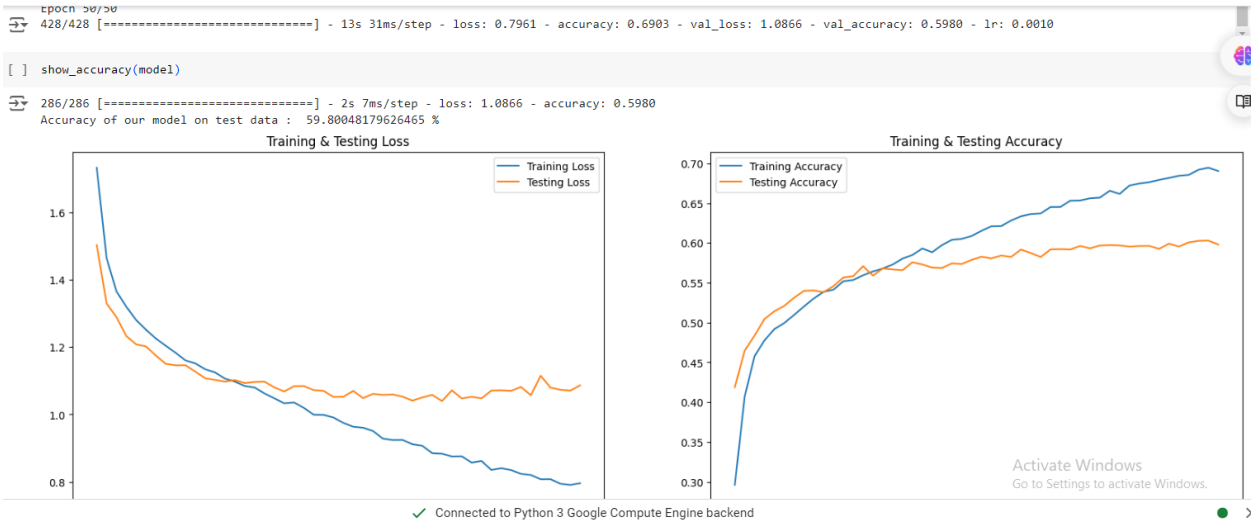
With the results as followed:



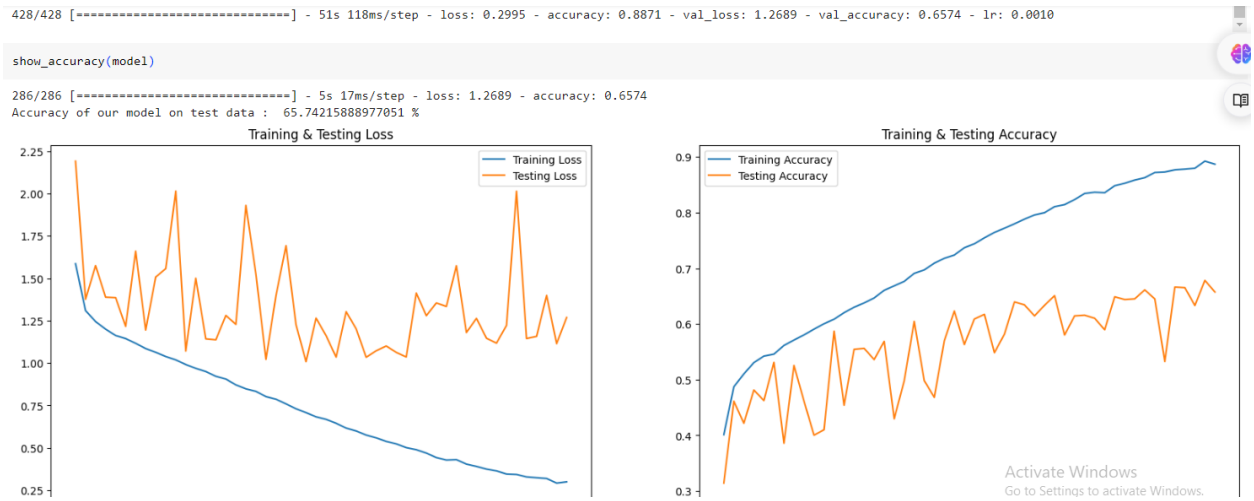
The above graph shows overfitting. The model is moving to memorizing than generalizing the features. This I concluded was due to using actual data without augmentation.

Another notable thing is **Dropout**. Dropout is done by dropping random neurons from the layer. It is also a technique I noticed in this architecture to prevent overfitting. The dropping of some neurons results in information loss that prevents model from memorizing the features.

2. For the second try, same architecture was used but this time original + augmented data was used for training. But unfortunately, accuracy dropped and model overfitted a little bit. I guess it was due to model started learning features again, or since data increased, layers or neuron count should also be increased.



3. For the third try, I added extra layers, added dropouts more and added batch Normalization as well. The later one is used to normalize the data and can help in reducing overfitting. Training, validation and testing accuracies of this model increased however overfitting remained a problem in more of weird way.



WHY NOT ML ALGORITHMS:

Audio signals contain features that are closely related to the previous ones. The emotion expressed in audio is not just a 1 frame thing, but it has important information how it started. In 3seconds audio, each frame is dependent on its previous one. The problem with ML algorithms is that these consider each feature as independent. One feature has no affect on other features. However, in case of audio, each frame has evolved or has effect of its previous one. So, ML algorithms are not suitable for this task.

WHY CNNs have been used for this task:

CNNs are effective in capturing the local patterns. It is done by making frames of input data and extracting features by applying kernel to each frame. This it will get a value from that frame. In next step, the kernel slides over to next values or window and extract features the same way. This helps in capturing the information how the features are evolving with respect to time.

PROBLEMS FACED:

There appeared many little problems while doing this task.

- Terminologies and Technologies used previously were difficult to understand. I understood dependency of features, normalizing features, over and under fitting of model, scaling, one hot encoding and a little about LSTM's, activation functions. But most of them were new.
- Session crashed many times while model training. So, it took a little longer to test.
- Managing time was little hard. But I advanced by deleting distractions from my phone which include some games, and fb.

What could be improved:

Understanding underlying ML and ANN concepts is crucial to select which one would be better rather than doing a hit and trial method. It not only saves time and resources but makes one that capable that they could suggest a code algorithm and architecture by just looking at some stats of features. This is the part I am still learning, and so far, what I learnt, I will ensuring full command on that. So, to me the most important thing is learning the basics.

The actual problem I see is possibly in the architecture. There could be some techniques to provide overfitting and improving accuracy as well.

HELPERS:

1. https://www.upgrad.com/blog/basic-cnn-architecture/
2. https://chatgpt.com
3. Ibrahim Sultan (a friend, a mentor)
4. YouTube.
5. Shivam Burnwal