Emotion Detection of Audio using CNNs

Introduction

Convolutional Neural Networks are the type of Deep Neural Networks that are mainly famous for image classification that tend to mimic the nature of human visual cortex by using the concepts of pooling layers, feature maps and receptive fields to retrieve valuable information from images. In this project, Convolutional Neural Networks are **implemented differently for emotion detection of the audio data instead of images** using a popular library for audio data manipulation called **'librosa'** which helps to analyse different audio data through amplitude, frequency and time.

Datasets

There are two datasets that have been used in this project and both the datasets contain audio clips of different actors pronouncing different phrases with different emotion in English language. For ex- "Dogs are sitting by the door". **Total number of emotions**- 8 (Neutral, Calm, Happy, Sad, Angry, Fear, Disgust, Surprise)

1. RAVDESS- https://www.kaggle.com/datasets/uwrfkaggler/ravdess-emotional-speechaudio
2. TESS-https://borealisdata.ca/dataset.xhtml?persistentId=doi%3A10.5683%2FSP2%2FE8H2MF

Methodology

Loading the data-

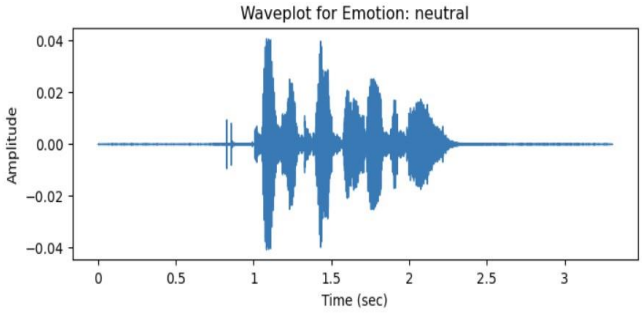
'Librosa' is a famous library for audio data analysis and it simplifies various complex concepts into easy-to-use functions that can be used to implement audio data visualization, augmentation and feature extraction. To load the data use- librosa.load(file_path, sr=None). This function returns-

- Data – floating point time series representation of audio
- Sample-rate- By default, 22050 Hz. It represents the rate at which the audio was sampled while converting the analog signal into digital signal so that the device can interpret it in a similar way as a human does. The higher the value the better details could be captured.

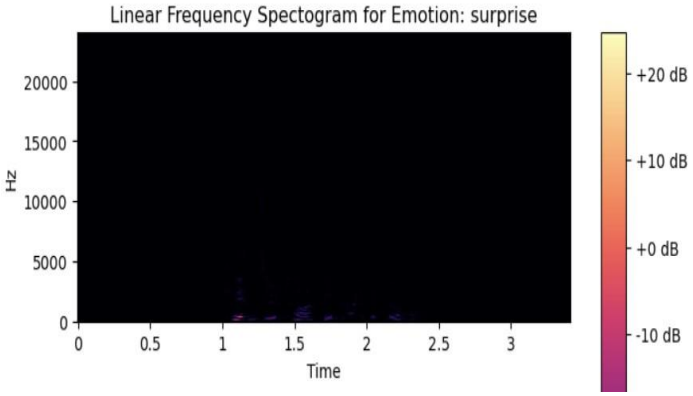
Converting audio data to waveforms and spectrograms.

The audio data can be represented in two domains – **time or frequency**. **Fourier Transform (FT)** is a phenomenon used to convert the time domain into the frequency domain.

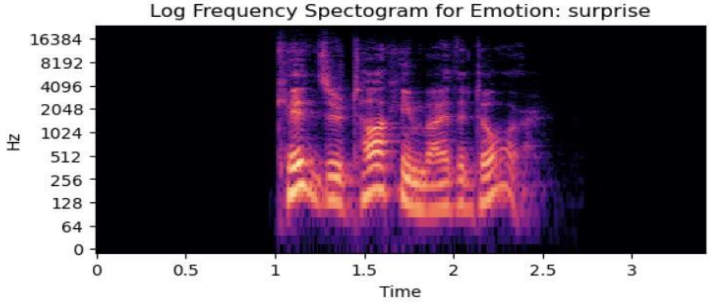
1. **Waveforms**- These are used to represent amplitude of the audio in **time domain** but it is not very helpful to get insights about the presence of different frequencies at different points of time. - librosa.display.waveshow(y= audio, sr= sample_rate)



2. **Spectrograms** – This is used to represent audio in **frequency domain**. To visualize frequencies in an audio file, we need to convert the time domain into frequency domain using FT (Fourier Transform). But Librosa computes it over short periods of time to capture the change for different point of times (STFT), so as result, it helps to achieve frequency-amplitude-time representation of the audio. - librosa.display.specshow()
    - Short Time Fourier Transformation (STFT)- 'stft()' in 'librosa' divides the audio into short time windows and then, FT (Fourier Transform) is applied to convert the domain of these short frames to frequency.



3. **Spectrograms with decibel scale**- The previous representation is still not helpful because we are still not able to see amplitude of different frequencies properly on linear scale of amplitude.
    - 'amplitude_to_scale()' - to convert the amplitude of audio into decibels (a logarithmic scale) because logs are useful to represent the data in a form which is more comparable as opposed to linear scale where data won't be visible because frequencies have large amount of difference and hence, it will be helpful to better capture information in the spectrogram.



The following function was used as a reference from 'Medium.com' and altered to create visualizations of the audio data.

```
def plot_audio(data:np.array, sr:int,title:str):
    fig, (ax1, ax2, ax3)= plt.subplots(3, figsize=(8,10))
    plt.subplots_adjust( bottom=0.1,
                    top=0.9,
                    wspace=0.4,
                    hspace=0.4)

    librosa.display.waveshow(data, sr=sr, ax=ax1)
    ax1.title.set_text(f'Waveplot for Emotion: {title}')
    ax1.set_xlabel('Time (sec)')
    ax1.set_ylabel('Amplitude')

    #to visualize all the frequencies in an audio at given point of time
    data_ft= librosa.stft(data)
    librosa.display.specshow(data_ft, sr=sr, x_axis='time', y_axis='linear', ax=ax2)
    ax2.title.set_text(f'Linear Frequency Spectogram for Emotion: {title}')

    #To convert amplitude into a logarithmic scale in order to compress it so that it ge
    signal_db= librosa.amplitude_to_db(abs(data_ft))
    img=librosa.display.specshow(signal_db, sr=sr, x_axis='time', y_axis='log', ax=ax3)
    ax3.title.set_text(f'Log Frequency Spectogram for Emotion: {title}')

    plt.colorbar(img, ax= [ax2,ax3], format= '%+2.f dB')
    plt.show()
```

Data Augmentation Techniques-

Data augmentation is a technique used to modify the existing data and create new data out of it so that it can be used to make the model more robust. But these techniques should be implemented in a subtle manner so that it doesn't affect the credibility of the audio. For example, it won't be helpful if pitch of an audio containing violin music is altered to an extent that the notes change into something that is out of the range of a violin, it won't be helpful for our model as it is not realistic. There are various data augmentation techniques that can be implemented with 'librosa' or with the help of 'NumPy' and simple mathematical operations-

1. **Time shifting**- In this technique, a certain part of audio is taken and shifted but, it wouldn't be useful when the order of the audio matters.
2. **Time stretching**- The speed of audio is changed without affecting the pitch of the speech.
3. **Pitch scaling**- It is complementary to time stretching, hence used to change the pitch without affecting the speed of the audio.
4. **Noise addition**- A white noise or environmental noise is added in the background of the audio, hence, it helps in training the model for robustness.
5. **Polarity Inversion**-
6. **Random gain**- It simply adds some loudness to the audio.

Feature Extraction Techniques-

These techniques are used to extract features from the audio data so that it can be quantified and used as an input to the model.

1. **Mel Spectrogram**- Mel spectrogram is one step ahead of spectrogram with decibel scale. It uses mel-scale instead of frequency on y-axis. The reason behind this is that

humans do not perceive frequencies linearly and are more likely to detect the differences between lower frequencies rather than higher frequencies, for instance, if there are two given pair of frequencies say 100Hz-200Hz and 1000 Hz- 1100H, then the lower pair would sound further apart to us as compared to pair with higher frequencies, even though the gap between the both is same. Hence, humans perceive frequencies in logarithmic scale, so frequencies are replaced with me-scale. *(Doshi, 2021, How do humans hear frequencies)*

2. **MFCC (Mel Frequency Cepstrum Coefficient)-** Voice depends on the vocal tract, tongue and many other features. MFCC helps to describe overall shape of spectral envelope of the sound which is helpful for identification of sound sources. It represents the timbre quite well. *(Mahajan, 2023)*

3. **Chroma STFT-** This feature tries to identify changes in the pitch and maps the sound in audio to 12 notes of music namely- A, A#, B, C, C#, D, D#, E, F, F#, G, G#.

4. **ZCR (Zero Crossing Rate)-** To extract number of times the audio signal passes the horizontal axis, i.e., the amplitude reaches 0.

5. **RMSE (Root Mean Square of Energy)-** It measures average loudness of the signal by taking into account the energy of the wave signal.

6. **Spectral Centroid**- It measures the central tendency of the signal, i.e., location of centre of mass of the spectrum. (Mahajan, 2023) The code for this particular feature was taken from 'MusincInformationretrieval.com' provided by Iran R. Roman.

The idea of combining both data augmentation and feature extraction was taken from 'Medium.com' provided by Tushar Gupta.

```python
def get_features(audio, sample_rate):
    output_arr= np.array([])

    #Mel spectrogram- Mel frequency cepstral coefficients.
    mel_output= mel_spectrogram(audio, sample_rate, flag=0)
    output_arr= np.hstack((output_arr, mel_output))

    #MFCC
    mfcc_output= mfcc(audio, sample_rate, flag= 0)
    output_arr= np.hstack((output_arr, mfcc_output))

    #Chroma_stft
    chroma_output= chroma(audio, sample_rate, flag=0)
    output_arr= np.hstack((output_arr, chroma_output))

    #ZCR- number of time signal crosses horizontal axis, i.e, amplitude=0.
    zcr_output= zcr(audio, sample_rate, flag=0)
    output_arr= np.hstack((output_arr, zcr_output))

    #RMSE- Root meam square energy of the signal.
    rmse_output= rmse(audio, flag=0)
    output_arr= np.hstack((output_arr, rmse_output))

    #Spectral Centroid
    spectral_output= spectral_centroid(audio, sample_rate, flag=0)
    output_arr= np.hstack((output_arr, spectral_output))

    return output_arr
```

CNN Model

Model: "sequential_5"

Layer (type)	Output Shape	Param #
conv1d_11 (Conv1D)	(None, 163, 128)	768
max_pooling1d_11 (MaxPooling1D)	(None, 82, 128)	0
dropout_12 (Dropout)	(None, 82, 128)	0
conv1d_12 (Conv1D)	(None, 82, 64)	41,024
max_pooling1d_12 (MaxPooling1D)	(None, 41, 64)	0
dropout_13 (Dropout)	(None, 41, 64)	0
flatten_5 (Flatten)	(None, 2624)	0
dense_10 (Dense)	(None, 32)	84,000
dropout_14 (Dropout)	(None, 32)	0
dense_11 (Dense)	(None, 8)	264

Total params: 378,170 (1.44 MB)

Trainable params: 126,056 (492.41 KB)

Non-trainable params: 0 (0.00 B)

The features obtained after applying feature extraction techniques, they were stored in separate dataframes for training, validation and testing data. The dataset was divided into training and testing by 80:20 ratio and then, further the training set was split into training and validation sets by 80:20 ratio. The CNN model has 2 convolutional layers accompanied with max pooling layers. The first 1D convolutional layer contains 128 units with with kernel size as 5, whereas second layer has 64 units. After that the input from previous layer is flattened and fed into dense layer containing 32 units and then to the output layer which uses softmax as its activation function because we need probability of possibility of each of the available emotion classes

and thus, 'categorical crossentropy' is used as a loss function along with it. The model was trained for 20 epochs and validation data was provided as well.

Results

The model was able to achieve accuracy of about 85% for both training and validation datasets. For the test set, the model was able to achieve accuracy of about 79%.

Ethical Impact

The dataset used in this project contains only 8 emotion types so, it is not very inclusive of many different kinds of emotions like, shocked, horrified, envious etc. Although such models are used for emotion detection in applications like spotify but they are also used in more highstakes situations like providing aide in courts to detect state of mind, so accuracy is very important. (Abbaschian, 2023). Also, since the model used is a neural network so, accountability is also another issue if it is used for malicious purposes. So, certain technical and ethical standards should be met while building such models such as de-identification of data to avoid malicious usage of data to replicate someone's voice and following the guidelines of laws like GDPR and Artificial Intelligence Act in Canada.

References

1. Doshi, K. (2021, December 31). Audio Deep Learning Made Simple (Part 2): Why Mel Spectrograms perform better. *Medium*. https://towardsdatascience.com/audiodeep-learning-made-simple-part-2-why-mel-spectrograms-perform-betteraad889a93505
2. Mahajan, K. (2023, October 27). Extracting audio features using Librosa - Kaavya Mahajan - Medium. *Medium*. https://kaavyamaha12.medium.com/extracting-audiofeatures-using-librosa-3be4ff1fe57f
3. Mahajan, K. (2023, October 27). Extracting audio features using Librosa - Kaavya Mahajan - Medium. *Medium*. https://kaavyamaha12.medium.com/extracting-audiofeatures-using-librosa-3be4ff1fe57f
4. Rayden, M., & Rayden, M. (2023, June 12). *What is RMS in audio world?* Major Mixing. https://majormixing.com/what-is-rms-in-audio-world/
5. Abbaschian, B. (2023, May 22). Ethical Concerns in Speech Emotion Recognition,

Part 3, Accuracy, Application, Responsibility, Accountability, and Monitoring. *Medium*. https://medium.com/@baabak/ethical-concerns-in-speech-emotionrecognition-part-3-accuracy-application-responsibility-3f036435b1bb