Team 7: Anirudh Potturi, Harshit Rajvaidya, Kanika Saraswat

# Assignment 4

Audio Classification System

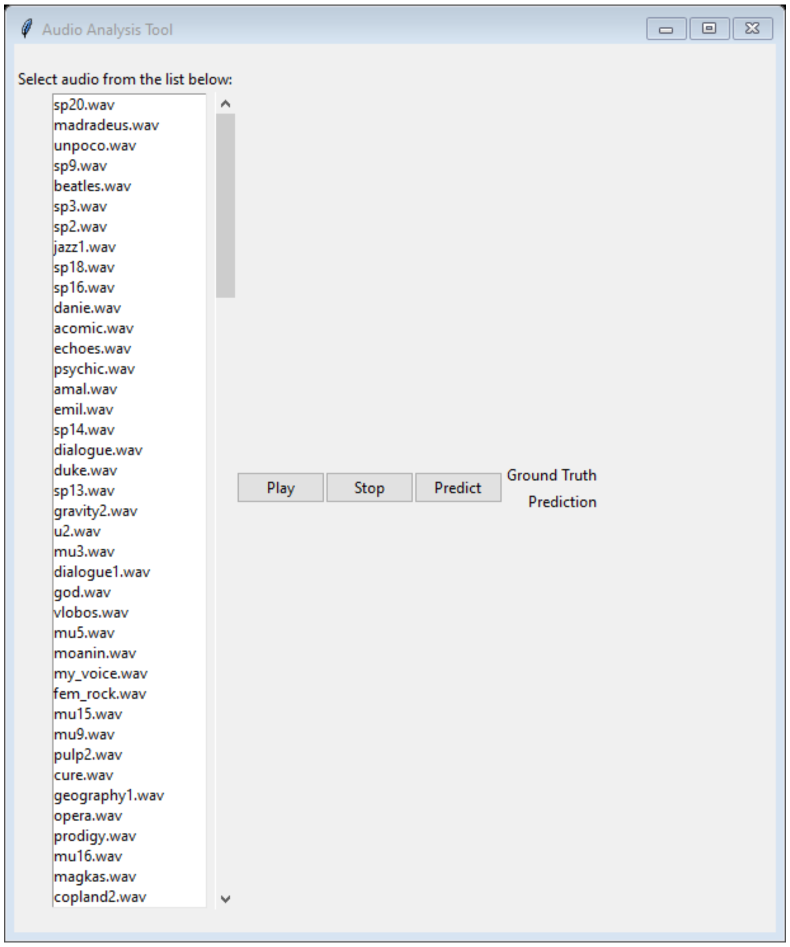
## Requirements

In order to run the application, the user must have a list of frameworks and libraries installed on their systems. The 'requirements.txt' file contains all the prerequisite packages along with the commands. The users may use the command shown below to run the initial installation process:

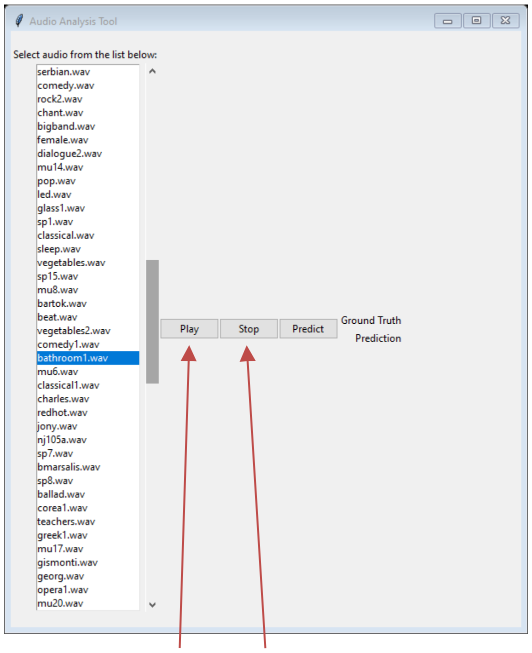
```
pip install -r requirements.txt
```

## User instructions

User launches the application by running the program. The user is presented with a screen as follows:

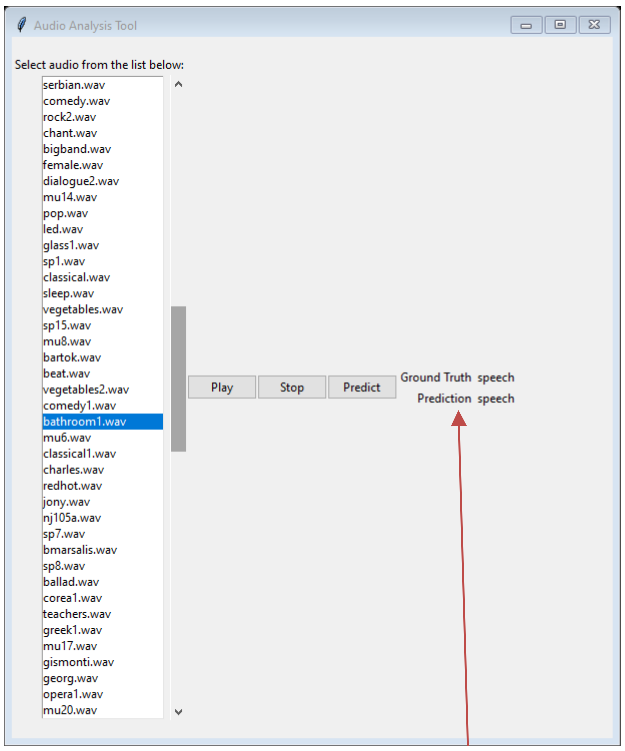


Next, the user selects an audio clip:



Two buttons Play and Stop:
User may click Play to play an audio clip selected.
User may click Stop to stop playing an audio clip.

Next, the user may click Predict to find out what category the audio clip belongs to:



Ground Truth speech
Prediction speech

The system displays the two results. One, is the Ground truth and the other, Predicted output.

## Tools and Libraries

### ***Tkinter***

Tkinter is a built-in framework in Python which is lightweight and easy to use. The main reason for choosing this framework for the GUI is because this framework is cross-platform meaning the application can be run on different operating systems. Tkinter allows an incredible range of features to create an extremely interactive UI. However, there is a compromise made which is the look of the UI does feel outdated. For this application, the modern sheen is not of priority, hence the GUI is kept simple.

### ***Tensorflow and Keras***

Tensorflow is an open-source library developed for Machine Learning and Artificial Intelligence. This library is used to design, build and train deep learning models. Keras acts as an interface for Tensorflow and is used to build artificial neural networks. Our application system applies a artificial neural network trained on audio data for classification.

### ***Librosa***

Librosa is a python package for music and audio analysis. It provides the building blocks necessary to create the music information retrieval systems. Librosa is used in visualizing audio signals and performing feature extractions using different signal processing techniques. In this application, Librosa was used to extract features and prepare features like MFCC, Cero-Crossing Rate, Root Mean Squared Energy and Spectral Centroid which were then used to train the Artificial Neural Network Model.

### ***Numpy and Pandas***

Numpy is a module used to handle numeric data whereas Pandas is a module used to handle tabular data. Pandas provides a powerful tool called Dataframe which was used to handle and analyze data in large sets. Numpy on the other hand was used as a means to handle the feature data.

### ***Pygame***

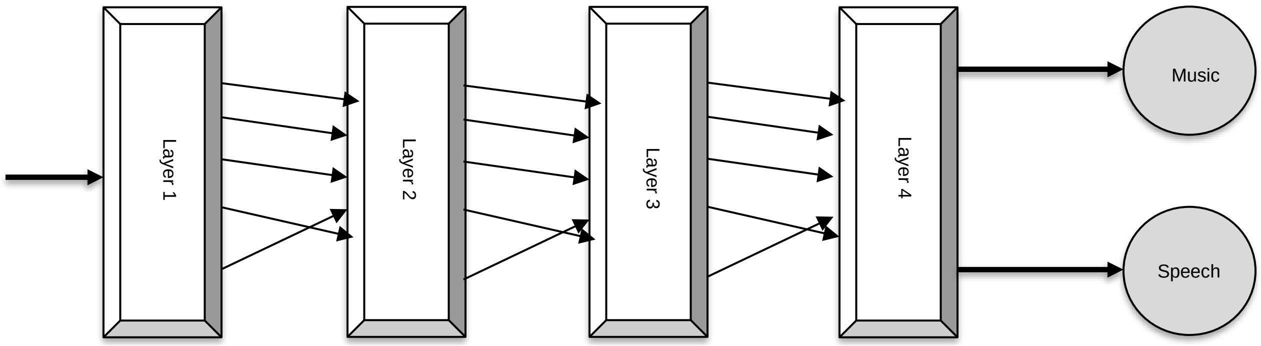
Pygame is a Python wrapper for the SDL library, which stands for *Simple DirectMedia Layer*. SDL provides cross-platform access the underlying multimedia hardware components of a user's system, such as sound, video, mouse, keyboard, and joystick. In this application, Pygame is used to allow the user to play the audio clips on demand.

### ***Sklearn***

Scikit-learn is an open-source machine learning library that supports supervised and unsupervised learning. It also provides various tools for model fitting, data preprocessing, model selection, model evaluation, and many other utilities. This library was used to mainly split the data into Train and Test sets for the Machine Learning implementation section.

## Machine Learning Model

**Neural Network** A neural network is a series of algorithms that endeavors to recognize underlying relationships in a set of data through a process that mimics the way the human brain operates. In this sense, neural networks refer to systems of neurons, either organic or artificial in nature. Our application system implements an Artificial Neural Network . The architecture contains 4 layers. All the layers are dense layers with ReLU as the activation functions. The last layer however has an activation Softwmax because it is useful in determining the probabilities for each class.



## Feature Extraction

Audio features were needed to be extracted from the audio clips to prepare the feature set which would later be passed into the Neural Network. Features extracted from individual audio clips and later combined into a dataframe are as follows:

1. Spectral Centroid: The spectral centroid is a measure used in digital signal processing to characterize a spectrum. It indicates where the center of mass of the spectrum is located. Perceptually, it has a robust connection with the impression of brightness of a sound. It is calculated as the weighted mean of the frequencies present in the signal, determined using a Fourier transform, with their magnitudes as the weights:

$$\text{Centroid} = \frac{\sum_{n=0}^{N-1} f(n)x(n)}{\sum_{n=0}^{N-1} x(n)}$$

where x(n) represents the weighted frequency value, or magnitude, of bin number n, and f(n) represents the center frequency of that bin. The Central Spectroid of a musical audio signal will be higher than for a speech audio signal.

2. Zero-crossing rate: The zero-crossing rate (ZCR) is the rate at which a signal changes from positive to zero to negative or from negative to zero to positive.[1] Its value has been widely used in both speech recognition and music information retrieval, being a key feature to classify percussive sounds. ZCR is defined formally as:

$$zcr = \frac{1}{T-1}\sum_{t=1}^{T-1} 1_{\mathbb{R}_{<0}}(s_t s_{t-1})$$

where ***s*** is a signal of length ***T*** and $\boldsymbol{1_{R<0}}$ is an indicator function. The ZCR of a musical audio signal is higher than that of speech

3. Mel-Frequency Cepstral Coefficients(MFCCs): Mel-frequency cepstral coefficients (MFCCs) are coefficients that collectively make up an MFC. They are derived from a type of cepstral representation of the audio clip (a nonlinear "spectrum-of-a-spectrum"). The difference between the Cepstrum and the Mel-frequency Cepstrum is that in the MFC, the frequency bands are equally spaced on the Mel scale, which approximates the human auditory system's response more closely than the linearly spaced frequency bands used in the normal spectrum. This frequency warping can allow for better representation of sound, for example, in audio compression.
MFCCs are commonly derived as follows:
    a. Take the Fourier transform of (a windowed excerpt of) a signal.
    b. Map the powers of the spectrum obtained above onto the Mel scale, using triangular overlapping windows or alternatively, cosine overlapping windows.
    c. Take the logs of the powers at each of the Mel frequencies.
    d. Take the discrete cosine transform of the list of Mel log powers, as if it were a signal.
    e. The MFCCs are the amplitudes of the resulting spectrum.

There can be variations on this process, for example: differences in the shape or spacing of the windows used to map the scale, or addition of dynamic features such as "delta" and "delta-delta" (first- and second-order frame-to-frame difference) coefficients.

4. Root Mean Square Energy: The RMS value is defined to be the square root of the average of a squared signal.

$$RMS = \sqrt{\frac{1}{M}\sum_{m=0}^{M-1} x^2(m)}$$

Where ***M*** is the total number of samples in a processing window and ***x(m)*** is the value of the ***m<sup>th</sup>*** sample.

## Audio Files in Model Training and Testing

Our application is trained on a combination of the sample Input files given and The 'Music Analysis, Retrieval And Synthesis For Audio Signals' dataset; in short *Marsyas*'s GTZAN music/speech collection dataset. Each audio file is a short clip of maximum length of around 30 seconds. Audio files were split into training and testing sets with 1/3rd of the data being dedicated to the testing data only. For the purposes of maintaining consistency and speeding up the execution time, a model was saved which can be found as a '.h5' file in the folder. This model is precompiled and trained with 2/3rd the data and later tested data. The main reason for loading back a saved model is mainly to make the application run faster.

## Discussion

Our system implement an Artificial Neural Network as previously discussed. The model has 4 layer and has its epochs set to 100. The batch sizes were initially kept at 128. The model was not very accurate. This is because the dataset we have is not large enough. By reducing the batch size to 10, the accuracy of the model increased. With the initial 20 samples however, it was hard to train the model. Which is why, we obtained another dataset and increased the size of our database to a total of 178 samples, 84 of speech and 84 of music. With smaller batch size and a greater number of epochs for such a dataset, the Machine Learning model was able to extract essential features. Going any further than 100 epochs would result in us overtraining the model which did happen during testing. Overall, we were able to effectively capture important audio features used to discriminate between music and speech. However, to test our model's accuracy, the code is included in the program which can be uncommented and executed. Our testing process took place in two phases:

- Testing on the test sample
- Testing on the entire audio data all over again

In both the cases, the model achieved an accuracy of 92%!

There were some interesting observations during the testing the accuracy of our model. In the case of an audio file titled 'beatles.wav', when this clip was played, it could easily be mistaken for a Speech clip by the human ear. Our model however was able to correctly determine that it was a music file. But from the perspective of a user, it could sound more like a Speech than a Music file. In this case, an interesting question to ask is, is the *Model incorrect or is the User?* This is one factor we did not initially consider when obtaining datasets. It could always be possible that the dataset could contain samples that may be labelled incorrectly or may not exactly be relevant to the labelled as in our case. Such datasets when used in larger sizes to train models could potentially decrease a model's performance.

Lastly, we explore our results and obtain the following confusion matrix.

| Confusion Matrix | | Machine Learning Model Prediction | |
|---|---|---|---|
| | | Speech | Music |
| Ground Truths | Speech | 79 | 8 |
| | Music | 5 | 76 |

From this confusion matrix, we can compute Precision, Recall and also an F1 score from the following equations:

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

$TP$ = True positive

$TN$ = True negative

$FP$ = False positive

$FN$ = False negative

Precision = 79 / 84 = 0.9404
Recall = 79 / (79+8) = 79/86 = 0.9186
F1 score = 2 x [ (0.9404 x 0.9186) / (0.9404 x 0.9186) ] = 2 x (0.4646) = 0.9293 ***Precisely equal to the accuracy of our model.