# Sound Event Detection

EE603A Course Project

Pratikhya Ranjit
Roll No:190639
EE Department, IIT Kanpur

Rishabh Katiyar
Roll No:190702
EE Department, IIT Kanpur

***Abstract*—Sound event detection is the task of identifying different sound events present in audio along with their respective start and stop times. In this project, we have implemented five methods namely, Linear regression, CNN, RNN, GMM, and a DSP-based method to detect speech and music events in audio files. The highest accuracy obtained is 92.46% with the CNN model on the custom validation dataset.**

***Index Terms*—CNN, RNN, GMM, DSP**

## I. Introduction

Sound Event Detection (SED) is the task of recognizing the sound events and their respective temporal start and end time in a recording. Sound events in real life do not always occur in isolation, but tend to considerably overlap with each other along with the presence of noise in between. Thus we have considered three classes to classify while preparing our models namely speech, music and silence. Machine Learning based methods where the model was able to learn the features and classify the audio gave far better results than the Digital Signal Processing(DSP) Based Method.

## II. Dataset

### A. Dataset Preparation

Open source sound clips of three categories: speech, music, and silence were collected from the web. 400 seconds audio file was created for each category from the sound clips using Praat software which is a free computer software package for speech analysis in phonetics. [1] Training dataset was prepared using 80% dataset and validation dataset using 20% dataset.

### B. Feature Extraction

Mel-frequency Cepstral Coefficient (MFCC) is a popular feature used for training audio classifiers. The MFCC coefficients represent the envelope of the time power spectrum which accurately represents speech signal generated by the human vocal tract [2]. In this project, we use MFCC features to classify audio events as speech or music. The entire audio dataset is divided into 0.1s audio clips with appropriate labeling to prepare the training dataset. The audio files are loaded as floating-point time series[ref] using the Librosa library at a sampling rate of 16000 Hz. Power spectrogram is calculated with n_fft = 1024, hop length = 512 & window length = 1024 for each audio file. The power spectrogram is used to generate the first 13 MFCC coefficients as features of the audio files which are then used to train the models.

## III. Machine Learning Based Models

### A. Convolutional Neural Network

A convolutional neural network (CNN) is a very efficient neural network used for feature extraction in audio classification tasks. [3] In a convolutional layer, the input is convolved with filters so that the output matrix contains important features extracted from the input. In this project, CNN is implemented using a sequential model created using the Keras library of Tensorflow. The model takes in inputs of shape (64,4) where the first dimension is the number of mfcc features used. The first convolutional layer contains 128 filters of shape (3x3) and has ReLU activation. The next layer applies max pooling to remove unwanted features from feature maps. The next layer applies dropout that randomly makes some units in the hidden layers equal to zero in order to prevent the model from overfitting. In the next layer, flattening is done and passed to a fully connected layer with 64 units with ReLU activation. The next layer applies dropout again and the last layer is a softmax layer with 3 units that outputs the probability of input belonging to either speech, music, or silence category.
The model uses Adam optimizer with the loss function of Sparse Categorical Entropy because the training labels are integers. The Keras callback functions (EarlyStopping and ReduceLROnPlateau) are used for training the model in order to prevent overfitting and achieve the best-trained model.

### B. Recurrent Neural Network

A recurrent neural network(RNN) is a type of neural network that learns from sequential data. It uses the concept of "memory" where past data affect predictions in the future. It learns sequential data like speech and music very efficiently. [3] In this project, RNN is implemented using a sequential model. The first layer is the LSTM layer with 32 units. Long-short-term memory (LSTM) is efficient in predicting long-term dependencies in sequences. The next layer applies a dropout of 30% so that model does not overfit. The next layer is a fully-connected layer with 64 units having ReLU activation. The next layer again applies a dropout of 40% and the final layer applies the softmax function with 3 output units to predict the probability of three classes,i.e speech, music, and silence.
The optimizer and loss function for RNN model are also the same. The model uses Adam optimizer with the loss function

of Sparse Categorical Entropy because the training labels are integers. The Keras callback functions (EarlyStopping and ReduceLROnPlateau) are used for training the model in order to prevent overfitting and achieve the best-trained model.

### C. Linear Model

The Model is a Simple Logistic Regression Model implemented from scratch with only one hidden layer and the sigmoid activation in the last layer to classify the samples into two classes Speech & Music.

$$Z = W^T X + b \tag{1}$$

$$\hat{Y} = \frac{1}{1 + \exp^{-Z}} \tag{2}$$

Here X is the input of the shape ($N_{clips}$, $N_{mfcc}$, $N_{frames}$). The output A is of the shape ($N_{clips}$,2) where each clip is given a label either as speech or as music.

The Loss function used is *Binary Cross Entropy* shown in equation (3) with the optimizer as *Gradient Descent* Algorithm.

$$-\frac{1}{N}\sum_{i=1}^{N} y_i log(\hat{y}_i) + (1 - y_i) log(1 - \hat{y}_i) \tag{3}$$

On Differentiating the loss function with the weights and the bias, the update came as

$$dW = \frac{1}{N}(\hat{Y} - Y)X^T \tag{4}$$

$$db = \frac{1}{N}\sum_{i=1}^{N} \hat{Y} - Y \tag{5}$$

Thus we can update our weights and biases according to the Gradient Descent Algorithm to minimize the cost

$$W = W - \alpha * dW \tag{6}$$

$$b = b - \alpha * db \tag{7}$$

where $\alpha$ is the learning rate which can be tuned.

We are predicting the input frame wise which means that each of the clips is getting prediction for each of its frame and the final prediction for that clip will be the *argmax* of the predictions of all its frames.

The model correctly classifies all the pure music and speech samples.

## IV. Probabilistic Models

### A. Gaussian Mixture Model

Gaussian Mixture Model (GMM) is a probabilistic model used to cluster data using the weighted sum of many gaussian probability density functions. It assigns soft weights to each data point(feature vector) that denotes how much it belongs to anyone cluster(1 gaussian density function represented by a mean vector and a covariance matrix) [4]. The cluster to which the data point belongs is calculated using the likelihood function given by

$$P(X|\lambda) = \sum_{k=1}^{K} w_k P_k\left(X|\mu_k, \Sigma_k\right) \tag{8}$$

$$P_k\left(X|\mu_k, \Sigma_k\right) = \frac{1}{\sqrt{2\pi\left|\Sigma_k\right|}} e^{\frac{1}{2}(X-\mu_k)^T \Sigma^{-1}(X-\mu_k)} \tag{9}$$

where $\lambda$ represents training data, $\mu$ represents mean vector, $\Sigma$ respresents covariance matrix, $w_k$ represents weights and $k$ represents the index of GMM cluster.

## V. Digital Signal Processing(DSP) based Models

Human speech is composed almost entirely of tones in the 500-2500 Hz range. Music, however, is composed of a relatively wide range of tones from 20Hz to well over 15kHz. This suggests that the spectrum of a signal could be used to determine if it contained music or speech. Furthermore, because the spectrum of speech is dominated by frequencies below 3kHz, it may be possible to identify the sample by just comparing the relative energy of the signal above and below a certain frequency. Most speech will have a higher ratio of low-frequency energy to high-frequency energy than most kinds of music. The ratio of low-frequency energy to high-frequency energy was used because the amplitude of the audio signal will not affect this ratio. [5]

With the correct cutoff frequencies and ratio threshold, such an algorithm should be capable of identifying most types of music from most types of speech. The correct cutoff frequency is the one that yields the largest difference between the energy ratios of a given set of music and speech samples.

For the purpose of this project, a 1st order Low Pass Filter and a 1st Order High Pass Filter is used. The signal is passed through this Low Pass Filter and the High Pass Filter and the ratio of their energies is calculated. Finally, the ratio of the low spectrum to high spectrum energy is calculated and compared with a threshold. If the ratio is higher than the threshold the input is interpreted at speech, otherwise, it is assumed to be music.

The signal read by the load_audio function of Librosa is of the shape [Number of clips, Number of Samples]. A low pass filter transfer function is of the form

$$H(s) = \frac{\omega_0}{s + \omega_0} \tag{10}$$

We compute the discrete transfer function using the signal's sampling frequency - $\triangle t = \frac{1}{Fs}$. Computing the discrete transfer function using Tustin's method, set

$$s = \frac{2}{\triangle t}\frac{1-z^{-1}}{1+z^{-1}} \tag{11}$$

Thus the Discrete Transfer Function becomes

$$H(z) = \frac{\omega_0}{\frac{2}{\triangle t}\frac{1-z^{-1}}{1+z^{-1}} + \omega_0} = \frac{\triangle t\omega_0(z+1)}{(\triangle t\omega_0+2)z + \triangle t\omega_0 - 2} \tag{12}$$

The coefficients are calculated manually as we know $\triangle t$ and $\omega_0$.

We want to find the filter coefficients for the discrete update:

$$y[n] = a_1y[n-1]+a_2y[n-2]+..+b_0x[n]+b_1x[n-1]+.. \tag{13}$$

The coefficients can be taken directly from the discrete transfer function of the filter in the form:

$$H(z) = \frac{b_0 + b_1z^{-1} + b_2z^{-2} + ...}{1 - a_1z^{-1} - a_2z^{-2} + ...} \tag{14}$$

(This is a result of taking the Z-transform which is not shown here)

Comparing this to a transfer function with coefficients num = [$b_0$, $b_1$, $b_2$] & den = [1, $a_1$, $a_2$] is

$$H(z) = \frac{b_0z^2 + b_1z + b_2}{z^2 + a_1z^{+}a_2} \tag{15}$$

which is equivalent to

$$H(z) = \frac{b_0 + b_1z^{-1} + b_2z^{-2} + ...}{1 + a_1z^{-1} + a_2z^{-2} + ...} \tag{16}$$

So we can take the coefficients in the same order that they are defined in the numerator and denominator of the transfer function object. The only difference is that the coefficients in the denominator need a negative sign.
To filter the signal, apply the filter using the discrete update. The same method can be applied for a 1st order High Pass Filter as well whose Transfer Function is given as

$$H(s) = \frac{s}{s+\omega_0} \tag{17}$$

On applying the *Bilinear Transform* and setting *s* as given in equation(11), the discrete transfer function for the High Pass Filter becomes

$$H(z) = \frac{\frac{2}{\triangle t}\frac{1-z^{-1}}{1+z^{-1}}}{\frac{2}{\triangle t}\frac{1-z^{-1}}{1+z^{-1}} + \omega_0} = \frac{2(z-1)}{(2+\triangle t\omega_0)z + \triangle t\omega_0 - 2} \tag{18}$$

So we can find the coefficients($a_1$, $b_0$, $b_1$) in a similar way as done for the Low Pass Filter and the signal can be filtered by applying the discrete update.
The Power of both the filtered signals can be calculated using

$$P = \frac{1}{n}\sum_{i=1}^{N} y_i^2 \tag{19}$$

The ratio is calculated as

$$R = \frac{PowerofLowPassFilteredSignal}{PowerofHighPassFilteredSignal} \tag{20}$$

The cutoff frequency was chosen to be around 2 KHz & the Ratio threshold was chosen to be around 10. The accuracy achieved with this method was close to 55% on our dataset.

## VI. Audio Event Detection

We have used model-based segmentation method to determine timestamps of different audio events [6]. A trained model is used to classify each audio frame into one of the three categories, namely, music, speech and silence. Then a rule based algorithm is used to calculate timestamp of each frame and combine similar frames together to get the final event detection. The CNN model is used for the audio classification task because it gave the highest validation accuracy equal to 92.46%. After the feature extraction process, each audio file is divided into an array of several 0.1s audio frames. The CNN model is used to predict event labels, i.e., music, speech, or silence for each 0.1s audio frame. The timestamps of audio events are obtained from a function that takes as input the CNN model's predictions and clubs consecutive occurrences of the same event labels into a single timestamp. The function takes a hyperparameter $threshold$ which is the minimum number of consecutive frames for which the same labeled event must occur to be counted as one event.

## VII. Results

The accuracy achieved by neural networks (CNN and RNN) is higher than that of the linear model among the machine learning-based methods. The CNN model gave the highest accuracy of 92.46% amongst all the five models. The accuracy of the linear model and DSP-based models were nearly the same around 55 to 60%. The Gaussian Mixture Model gave the least accuracy.

## References


[1] D. Boersma, Paul & Weenink, "Praat: doing phonetics by computer [computer program]." http://www.praat.org/, 2021. Version 6.1.56.

[2] J. Jogy, "How I Understood: What features to consider while training audio files." https://towardsdatascience.com/how-i-understood-what-features-to-consider-while-training-audio-files-eedfb6e9002b, 2019.

[3] A. Khamees, H. Hejazi, D. M. Alshurideh, and S. Salloum, *Classifying Audio Music Genres Using CNN and RNN*. 03 2021.

[4] "Machine Learning in Action: Voice Gender Detection using GMMs: A Python Primer." https://appliedmachinelearning.blog/2017/06/14/voice-gender-detection-using-gmms-a-python-primer/.

[5] J. P. H. Christopher J. Vondrachek, "A Speech and Music Detector Project for a DSP Class."

[6] T. Theodorou, I. Mporas, and N. Fakotakis, "An overview of automatic audio segmentation," *International Journal of Information Technology and Computer Science*, vol. 6, pp. 1–9, 10 2014.