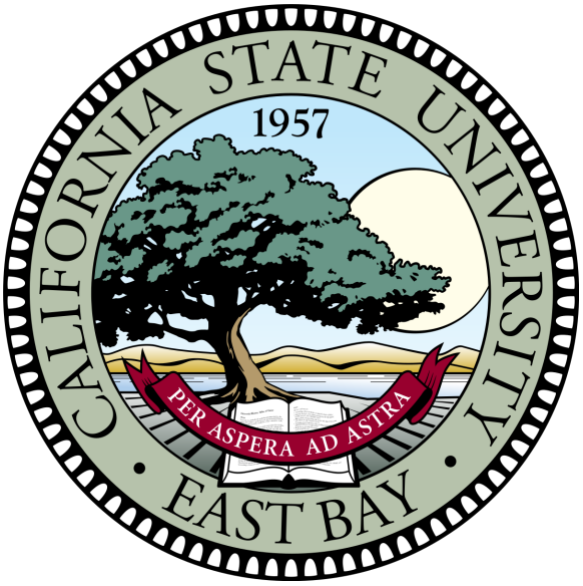
CONTENT BASED MUSIC GENRE CLASSIFICATION USING TEMPORAL AND SPECTRAL FEATURES



Sidhdharth Hareshbhai Pandya

California State University East Bay, Hayward
Department of Computer Science

Advisor: **Dr. Moayed Daneshyari**

Table of Contents

List of Figures

List of Tables

1. Introduction

1.1 Motivation:

Music is heard by everyone. It spreads emotions from one person to another. The music corpus of today is quite diverse. Each person has their own music evaluations. mostly due to the diversity of the composers and musicians. However, many companies that offer music streaming services, like Pandora, Spotify, Google, Apple Music and Prime music utilize state-of-the-art (SOTA) algorithms to identify similarities and patterns between tracks. As a result, recordings are classified into several groups known as "Genres" of tunes. It makes similar audio tracks and music recommendations based on the genres a user listen to.

We have access to an enormous quantity of data, and this number is growing fast every day. As a result, manual curation is becoming impractical, and automated techniques must be used to classify data. The music business is no exception. Automating the music tagging process would result in better data organization, enabling future development with this data easier, such as building themed playlists or recommending songs to users. Machine learning may be used to detect subtle patterns in data that would be difficult to explicitly build methods for. One such use case is deciding what genre a song belongs to, which is covered in this study. Finding patterns in audio is valuable for more than just musical analysis.

Music genre classification, while on the other hand, is, as previously said, an unclear and subjective process. It is worth noting that there is no precise description of what a genre should sound like, as this is a fairly conventional way of looking at music. Saying a song should be categorized a specific way is not right or incorrect; rather, it is a matter of personal opinion based on how the listener is affected by the music and how they identify with it [1]. It is also a challenging field of research, either because of low classification accuracy or because some argue that one cannot categorize genres that do not even have clear guidelines[2]. Though identifying music is not a random process for humans, there must be some consistency when it comes to what a genre sounds like [3].

1.2 Project Goal:

I am motivated to learn more about some of the existing methods for classifying music as listeners, as well as to conduct tests with various machine learning algorithms to see if I can enhance some of the methods now in use.

Individual instances in machine learning have various attributes, and practitioners use intuitions and domain expertise to try to uncover patterns in the data. Dataset will comprise of labelled audio recordings in this case where label represents the genre of the songs/tracks. The goal of this project is to learn how to work with sound files, calculate sound and audio attributes from them, run deep Learning Algorithms on them, and analyze the results.

The main objective is to develop deep learning models that systematically categorizes music samples into various genres. It attempts to guess the genre by using an audio sample as input. The purpose of automating music classification is to make song picking faster and less tedious. Manually classifying songs or music requires listening to a large number of tracks and then selecting the genre. This is both time-consuming and challenging. As a result, I will implement several supervised learning approaches to identify patterns and categorize the songs/tracks.

1.3 Project Roadmap:

For the project definition I'm implementing various Deep learning models to classify the music genres like Multilayer Perception Model, Convolution Neural Networks, Long Short-Term Memory in Recurrent Neural Networks and hierarchical LSTM network. I'll compare the results for all the mentioned models and conclude which one is better for audio data for our dataset.

As data becomes more widely available, classification of that data becomes more important in order to make appropriate use of it. There are several strategies for categorizing data, however it is unclear why one approach should be picked over another. This research intends to assist with this problem by examining and comparing the aforementioned solutions.

Section 2 examines the literature on music genre classification approaches, focusing on the state-of-the-art in the discipline. Section 3 contains a detailed description of the suggested strategy. Section 4 details the experiments we ran with various categorization techniques, input parameters, and music types. Section 5 discussed about future work. Finally, in Section 6, relevant remarks and conclusions demonstrate that significant findings were obtained, strongly motivating further study in this field.

2. Related Work

Besides the content-based music genre classification, other techniques exist as well such as: -

Collaborative filtering:
From [4], this approach makes a prediction about the taste of a user with the help of part of the community that shares the same (or similar) taste (thus called collaborative). An important assumption is made that if a user, say A, listens to the same songs of a genre as the users B, C, D and E, then A would also prefer the songs of other genres listened to by B, C, D and E.

The drawback of this technique would be, for the collaborative filtering approach to work, there must be a large set of users (i.e., the community) and user data.

Knowledge-based:
This approach (from [5]) draws in user interests and feedback at regular periods.
This technique is used mainly when the other two (content-based and collaborative filtering) cannot be applied.

This approach depends on user feedback (an example is the like and dislike option provided by Spotify for each song). Thus, inadequate feedback or unable to obtain feedback regularly hinders the working of the approach. This can be considered as a drawback.

Panagakis and Kotropoulos [6] developed a methodology for music genre classification that takes into account the characteristics of the auditory human perception system, such as 2D auditory temporal modulations representing music and sparse representation. The accuracies they got were higher than any previously reported rate for the GTZAN and ISMIR2004 datasets, at 91% and 93.56%, respectively.

Dannenberg RB, Thom B, Watson D [7] pioneering study identifies one of four forms of musician improvisation using Naive Bayesian and neural network techniques. They were evaluating a performer's capacity to develop purposeful and varied styles on a consistent basis. A database was created to train the classifiers, and when categorizing between four styles, an accuracy of 98% was reached. They achieved an overall accuracy of 77-90% by utilizing eight classifiers trained to respond "yes" or "no" for eight distinct styles.

Li T, Ogihara M, Li Q [8] conducted a comparison analysis of timbral texture, rhythmic content, and pitch content characteristics vs Daubechies Wavelet Coefficient Histogram features (DWCHs). They applied Support Vector Machines (SVMs), Linear Discriminant Analysis (LDA), and other learning approaches to make the classifications. They conducted tests using both the first seconds and the middle sections of music. They also experimented with the One-Against-All (OAA) and Round-Robin (RR) strategies. The greatest overall accuracy (74.2%) was obtained when DWCH characteristics and an SVM classifier based on the OAA technique were used in this test, which was performed with middle sections of songs (seconds 31–60).

Mckay and Fujinaga [9] wrote a study about why researchers should keep working to improve Music Genre Classification. They raise concerns about ambiguity and subjectivity in classifications, as well as the dynamic of music styles. Manually classifying recordings requires a great deal of knowledge and effort, and there is also minimal agreement among human annotators when classifying music by genre. Few genres have unambiguous definitions, and there is frequently substantial overlap between them. Furthermore, categorization tend to be by artist or album rather than individual recordings, and metadata available in audio file has untrustworthy comments. Finally, new genres are introduced on a regular basis, and our knowledge of current genres evolves with time.

Tzanetakis and Cook's [10] work is just another milestone in the field. They offered three distinct feature sets for timbral texture, rhythmic content, and pitch content. To create feature vectors, the Short-time Fourier Transform (STFT), Mel-frequency Cepstral Coefficients (MFCCs), Wavelet Transform (WT), and some extra parameters were utilized. They could train statistical pattern recognition classifiers such as basic Gaussian, Gaussian Mixture Model, and k-Nearest Neighbor [11] using these vectors and real-world audio datasets. They scored 61% accurate classifications for ten musical genres.

Paradzinets A, Harb H, Chen L [12] studied timbre, beat-related, and acoustic information. They applied Piecewise Gaussian Modeling features augmented by modeling of the human auditory filter to acquire acoustic data. In order to do this, they first acquired the PGM properties before applying the equal loudness, particular loudness sensation, and crucial bands filter. They employed wavelet transformations to obtain the 2D-beat histograms and extract the beat-related features. They gathered all detected notes together with the relative loudness of their harmonics in order to determine the timbre characteristics, and then they computed their histograms. Their results demonstrate, among other things: (i) an improvement when using perceptually motivated PGM rather than basic PGM, i.e., accuracy of 43% versus 40.6%; and (ii) training different NNs for each genre is preferable to training just one NN with all the genres taken into account, which equates to an average accuracy of 49.3%.

According to a space-time decomposition dimension, C.N. Silla Jr., A.L. Koerich, C.A.A. Kaestner 0.'s pattern recognition ensemble technique used various feature vectors with varied time segments from the beginning, middle, and end of the music. The following techniques were used: Naive-Bayes, decision trees, k Nearest Neighbors, SVMs, and Multilayer Perceptron Neural Networks. Utilizing a space-time ensemble and round-robin, the best accuracy was 65.06%.

Two strategies were suggested by Ezzaidi and Rouat [14]. They separated the musical compositions into frames, and from averaged spectral energy they obtained the MFCCs. Eventually, they applied Gaussian Mixture Models (GMMs) [15] for purposes of comparison, achieving a maximum recognition rate of 99%.

Deep learning-based methods are fairly popular when it comes to problems such as music genre classification. Most of the architecture has already been discussed but we will compare the results of these models with some modification in the architecture:

I. Convolution Neural network is specialized in image feature recognition. Convolutional Neural Networks (CNN) also have the potential to classify genres of music. We will use spectrograms which represents MFCC along with time and frequency will be the input for CNN model [16].

II. However, music (audio) data has a nice sequential structure. The order of data is important, and data across the timestamps should not be treated as independent. This problem is solved by recurrent neural networks because they are networks with loops that preserve previous knowledge. LSTM networks are a kind of RNN that can learn long-term dependencies. We would like to experiment using a single conventional LSTM network to make the conditions as similar as possible to that of the CNN [17].

III. In the project we have proposed a hierarchical LSTM-based model for the multi-class classification problem. We adopted the hierarchical LSTM architecture from [17] with some modification mentioned below:

- We proposed two different kind of approaches hard prediction and soft predictions.
- We are using sequence length = 256.
- We added Chroma-STFT, Spectral Centroid and Spectral Contrast features.

3. Model Development

3.1. Dataset:

The dataset used in the project is the GTZAN dataset [10] available at the MARYSAS website [18]. Review of over 467 published works in music genre classification done by B. L. Sturm [19] proves that the GTZAN dataset is most used public dataset which is appearing in more than 100 works.

This dataset was originally used in [10]. The files were collected in 2000-2001 from a variety of bases. The audio files were gathered using different sources like CD's, radio, microphone recordings etc. to represent a variety of recording conditions [18]

For a long time, experts have been attempting to comprehend sound and what distinguishes one music from another. Sound visualization. What separate one tone from another. This data, ideally, will provide the possibility to do so.

The dataset is made up of about 1000 audio tracks each of which is 30 seconds long. There are about 10 genres, each covering up 100 tracks for each genre. All tracks have 22050 Hz sampling rate, mono channel with 16-bit sample audio files in .wav format.

When we download the dataset from MARSYAS website [18], the Data folder contains:

- genres_original - Genres Original is a collection of 10 genres (i.e., blues, classical, country, disco, hip-hop, jazz, metal, pop, reggae, and rock), each featuring 100 audio files, all of which are 30 seconds long (the famous GTZAN dataset, the MNIST of sounds)

- images_original - Each audio file is represented visually. Neural networks are one method for classifying data. Because NNs (such as CNN) typically require some type of picture representation, the audio data were converted to Mel Spectrograms to allow for this

- 2 CSV files - CSV files that include information about the audio files' features. 'features_30_sec.csv' file contains a mean and variance calculated across several characteristics collected from an audio recording for each song (30 seconds long). The other file(features_3_sec.csv) has the same format as the first, except the songs were previously separated into 3 second audio files (this way increasing 10 times the amount of data we fuel into our classification models). When it comes to data, more is usually better.

So, utilizing the GTZAN dataset, we can classify music genres using CNN or other computer vision models, or we can train an LSTM or RNN using time-series data like MFCCs. The problem for the latter would be to pre-process the audio data sufficiently to extract all necessary elements, even if it meant employing numerous approaches other than MFCC. Another issue with this dataset is that the small sample size (less than 800 after a good train-val-test split) may not be sufficient to get good results. While you can always add other larger datasets like the Million Song Dataset [20], Spotify Classification [21], or Indian Music Genre Dataset [22] for further practice, it's crucial for novices to get comfortable with a new data format and to set up a machine learning pipeline.

3.2. Preprocessing:

First, for working with audio data in deep learning we need to understand the sound and some features related to music that differentiate into genres.

Sound is produced by the vibration of an object. These vibrations determine oscillation of air molecules. Alternation of air pressure or any elastic medium cause a wave.

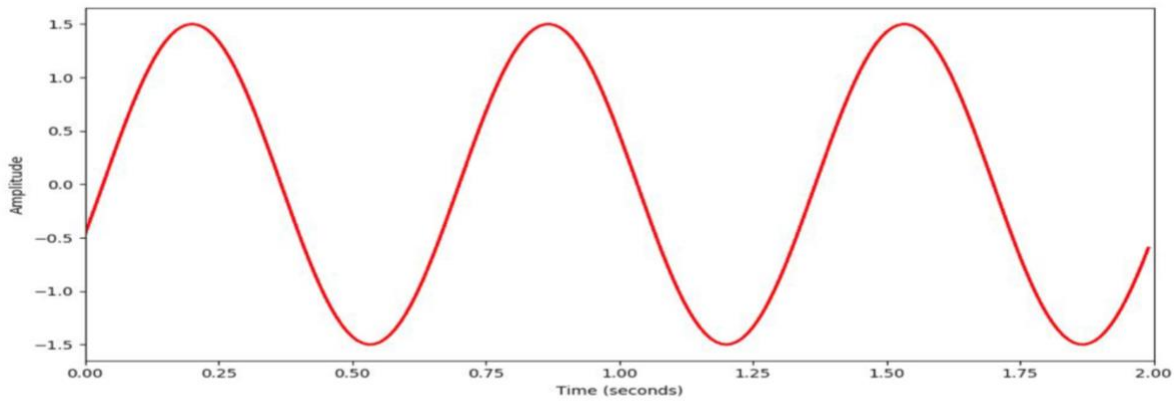


Figure 1: Simple Waveform (Amplitude vs Time)

Waveforms can be useful information of amplitude, frequency, intensity, timbre, pitch, phase etc. In this case we can represent sound wave as sine function:

$$y(t) = A\sin(2\pi f t + \varphi)$$

Pitch[1] is perceived as how "low" or "high" a sound is and represents the cyclic, repetitive nature of the vibrations that make up sound. So, we can say that higher the frequency pitch will be higher. Likewise, we can also derive that larger the amplitude [2] the loudness will be on the higher side.

In computers, audio track is represented as digital signal. Digital signal is discretized representation of the analog signal, and it is sampled at some sample rate. Audio track is either recorded using microphone or synthesized within computer itself.

Analog to digital conversion (ADC) can be done in two steps:

- Sampling- signal sampled at uniform time intervals
- Quantization- amplitude quantised with limited number of bits

The audio sample rate is amount of the samples per second taken by the system from a digital signal.[3] In our GTZAN dataset all tracks have 22050 Hz sampling rate.

[1] Sound/Pitch from Wikipedia
[2] Sound/Loud from Wikipedia
[3] Sample rate from izotope

The bit depth defines the number of possible amplitude values we can record for each sample.[4] In GTZAN dataset, bit depth is 16-bit sample audio files in .wav format.

Digital audio signal feature extraction is done emphasizing these fine-grained features like frequency, amplitude, phase, pitch, timbre, amplitude envelope, power of signal and tonal features.

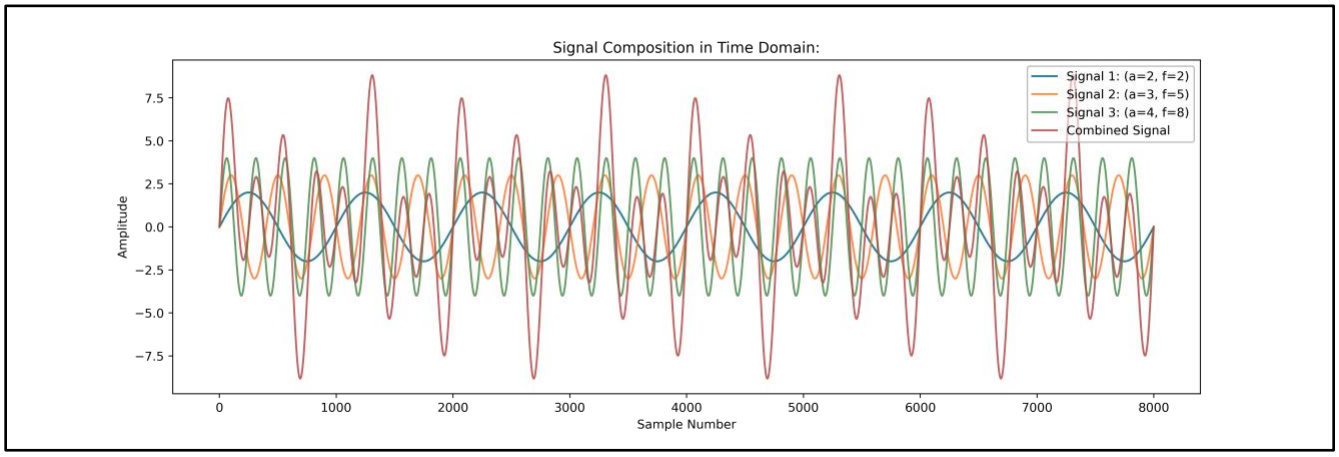


Figure 2: Representation of signals in Time Domain

Audio Signals are represented into two domains namely (1) Time Domain and (2) Frequency Domain. Both empirically and theoretically it is known that complex signals are easy to decompose in the frequency domain. Figure 2 shows a signal in the time domain and Figure 3 shows the same signal in the frequency domain.

The time domain to frequency domain conversion can be done using algorithms like Fast Fourier Transform (FFT). Fourier transform decomposes complex periodic sound into sum of sine waves oscillating at different frequencies. Which is calculated using equation below:

$$y(f) = \int_{-\infty}^{\infty} y(t) e^{-i2\pi ft}\, dt$$

Where $e^{-i2\pi ft}$ is the complex signal with frequency = f and y(t) are our signal. In Discrete Fourier Transform (DFT), we can use the dot product to find the similarity between our signal and the complex signal with frequency f. And we can do this for all frequencies between fmin to fmax to find the transformed signal.

Here in Figure 3, we can see that we have changed the waveform from time domain to frequency domain. Music data is time series data, so it is a sequence of data points indexed in timely manner. Hence when we convert into frequency domain, we miss the very important information of time in music. In music data, we want to know what has changed from the last time, so we are missing significant crux here.

[4] Bit depth from izotope

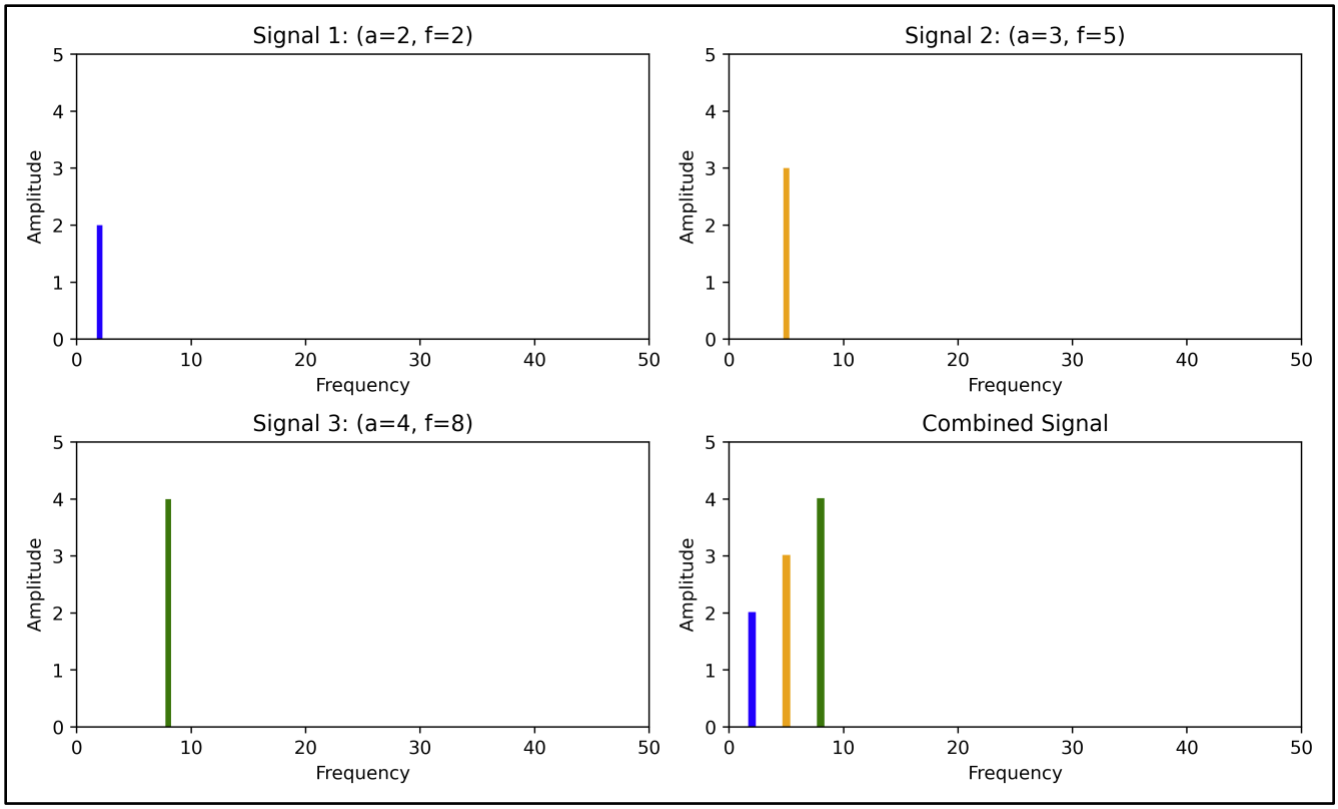


Figure 3: Representations of signals in Freq Domain

3.2.1 Chroma Short Time Fourier Transform (STFT)

Fourier transform of the signal gives the spectrum for whole signal at once. So, it will give global features. However, we are interested to find the non-stationarities in the signals as we are using recurrent models like LSTM. To achieve this, we can use something like windowing method. And perform (Short Time Fourier Transform) STFT on each window which convolves over our signal with stride equal to hop-length. This will give something like spectrum.

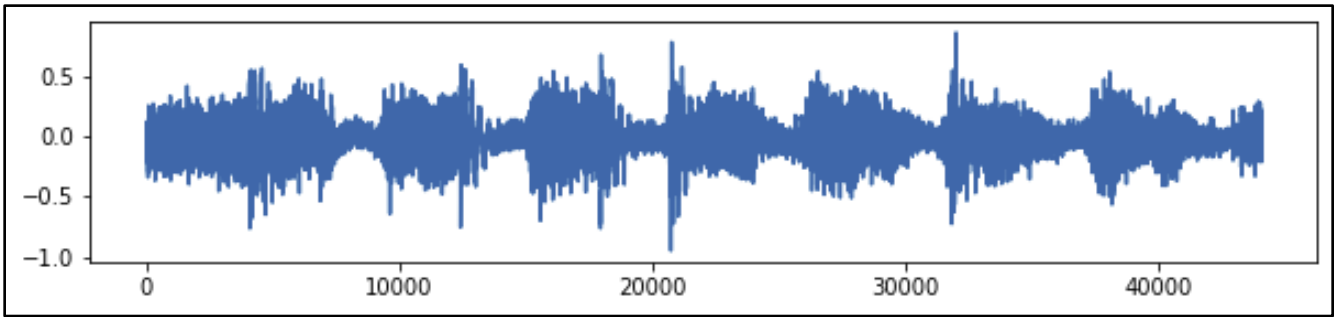


Figure 4: Signal in Time Domain

Frequency perception in human is non-linear in nature and spectrum captures the frequencies in the linear scale. Mel-Spectogram is variant of spectrogram which captures frequency on Mel-Scale which is similar to logarithmic scale.

Figure 4 and Figure 5 shows an example of signal in time domain and its Mel-Spectrogram respectively

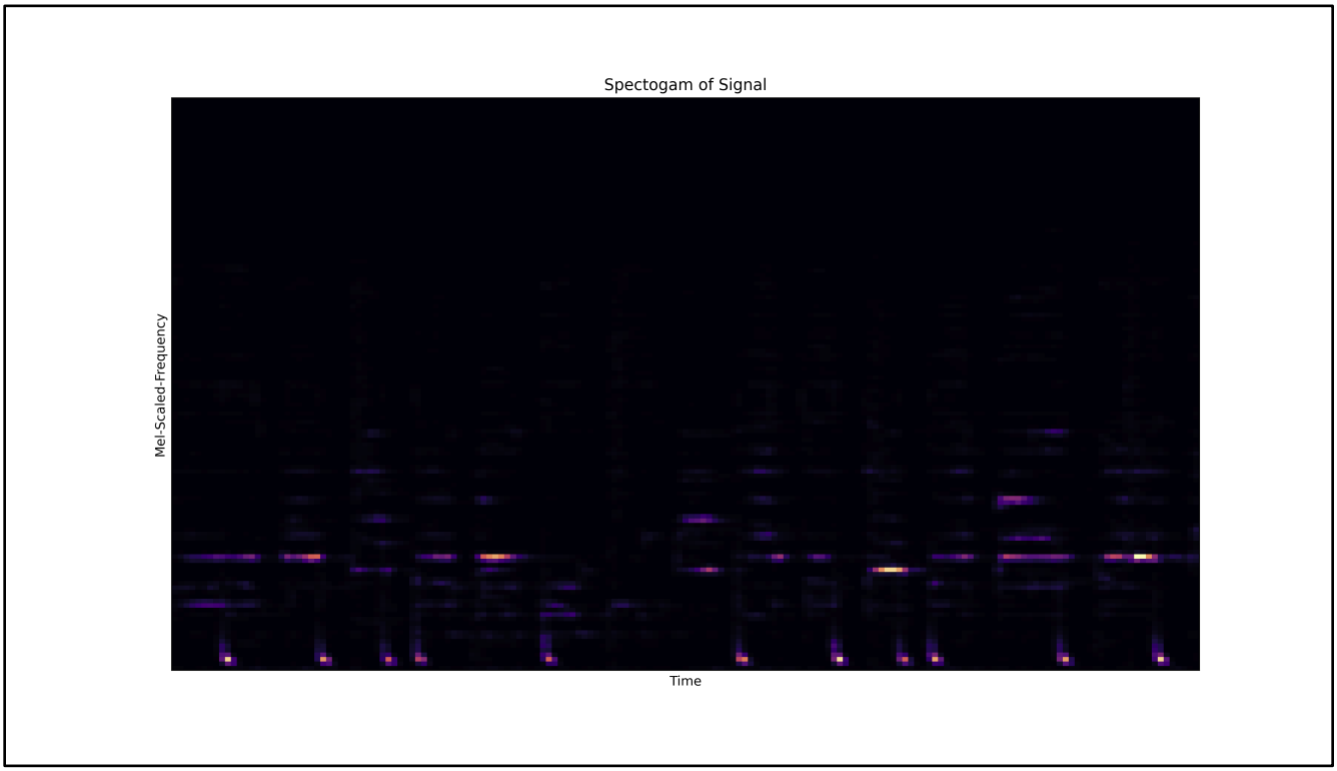


Figure 5: Signal in Mel-Spectogram

We use 22050 Hz sampled, 30-second-long audio file. We are taking window size = 2048 samples and hop-length = 512 samples. So, we will have approx. 1290 timestamps which is calculated using formula below: -

$$\text{No of Windows (Timestamps)} = \frac{(\text{Track Length} * \text{Sample Rate}) - \text{Hop Length}}{\text{Hop Length}}$$

To computes several Fast Fourier, transform at different intervals we use Short Time Fourier Transform (STFT). STFT preserves time information which we are losing in FFT. STFT gives a spectrogram (Time + Frequency + Magnitude) for fixed frame size (e.g., 2048 samples).

3.2.2 Mel Frequency Cepstral Coefficients (MFCCs)

Many studies utilize Mel-frequency cepstral coefficients as short-term spectral characteristics for speech recognition [23], retrieval systems [24], music summarization [25], and speech/music discrimination [26]. The capacity of the MFCC to portray the amplitude spectrum in a compact manner is its main strength. Figure 6 depicts the processes for computing MFCC as described in [27].

Mel-Frequency Cepstral Coefficients (MFCCs) are Cepstral coefficients generated from a signal's power spectrum using a discrete cosine transform. This spectrum's frequency bands are spaced logarithmically according to the Mel scale.

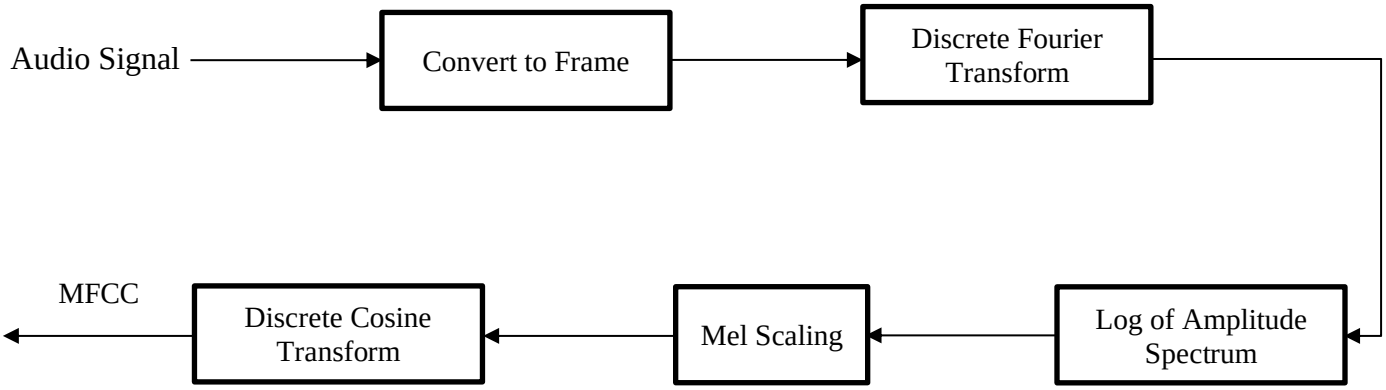


Figure 6: Steps for MFCC

MFCCs, in a sense, compress the data in a Mel-Spectrogram. DCTs are popular because they allow for simple image compression. The shape of the vocal tract is represented by MFCC. Figure 7 depicts the MFCC of an audio file.

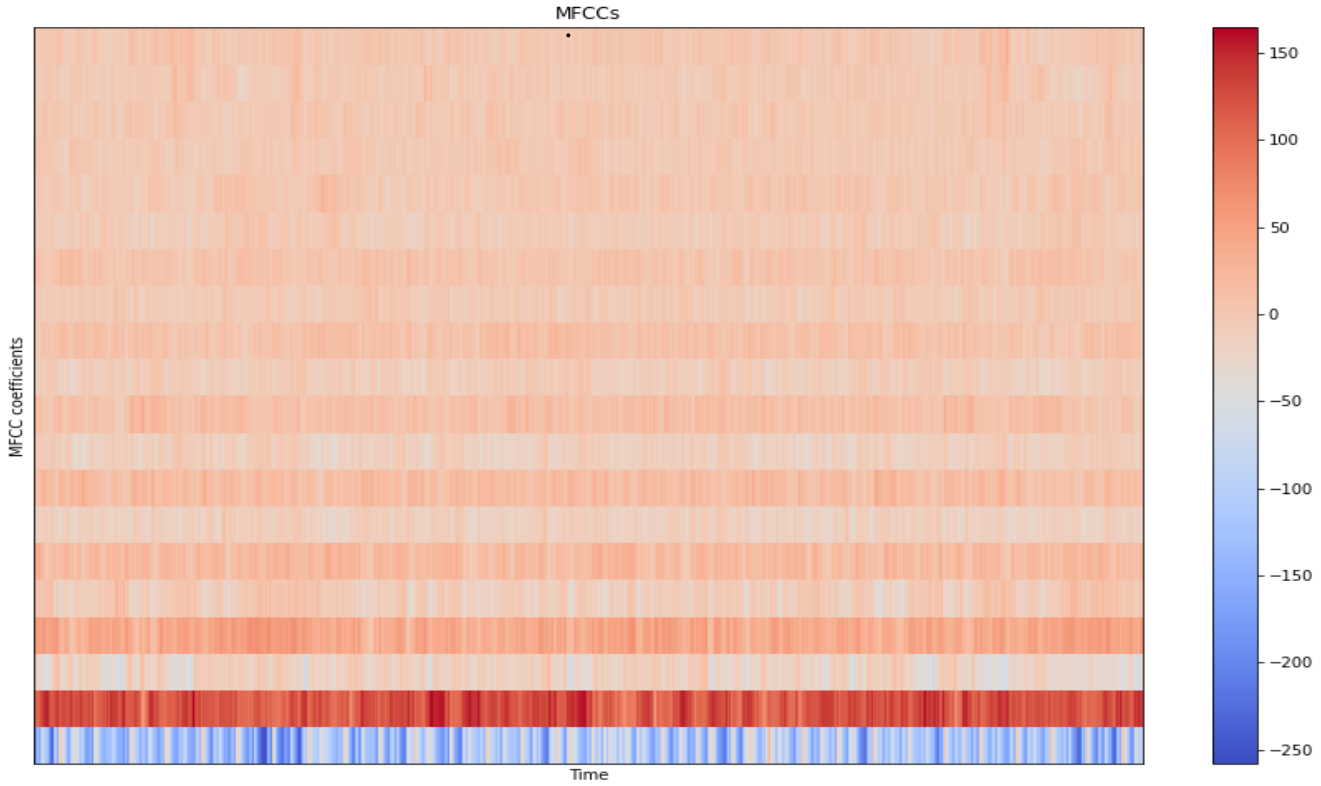


Figure 7: MFCCs

3.2.3 Spectral Centroid

In digital signal processing, the spectral centroid is a metric used to describe a spectrum. It has a strong perceptual link with the perception of brightness of a sound. It denotes the location of the spectrum's center of mass.

[5]

$$Spectral\,Centroid\,(Fc) = \frac{\sum_{k=1}^{K} S(x) f(x)}{\sum_{k=1}^{K} S(x)}$$

Some individuals refer to the spectrum's median as the "spectral centroid." This is a distinct statistic, with the distinction being roughly the same as that between the unweighted median and mean statistics. Because they are both measures of central tendency, they will behave similarly in various instances. Because the spectral centroid is an excellent predictor of a sound's "brightness," it is commonly utilized as an automated measure of musical timbre in digital audio and music processing.

3.2.4 Spectral Contrast

A spectrogram is separated into sub-bands in each frame. Comparing the mean energy at the highest quantile (peak energy) to that of the lowest quantile allows one to determine the energy contrast for each sub-band (valley energy). High contrast levels are often associated with clear, narrow-band signals, whereas low contrast values are typically associated with broad-band noise [28].[6]

Here, each window acts as a timestamp for our network. Since LSTM has restriction on the sequence length, we fix the sequence length = 256. To do this we split the tracks into 5 segments such that each segment has 256 timestamps. Note that here number of timestamps is the same as number of different windows on which we are finding the features.

Our dataset had initially 1000 songs but after enforcing the sequence length = 256, we are kind of augmenting the dataset such that it will have total of 5000 segments each of sequence length = 256.

Now for each timestamp, we calculate following features:

Features	**Dimensionality**
MFCC	20
Chroma STFT	12
Spectral Centroid	1
Spectral Contrast	7

Table 1:Dimensionality

[5] Spectral Centroid from Wikipedia
[6] Spectral Contrast from librosa

The MFCC Features captures the details about the envelope in IDFT of the Log Power Spectrum which is extracts similar features from Mel-Spectogram but represented in dense way. We are using 20, Chroma-STFT features finds the Intensity for different tones of the current window. There are 12 tones namely (C, C#, D, D#, E, F, F#, G, G#, A, A#, B) So, per window its dimensionality is 12. Spectral Centroid captures the average frequency weighted by intensity at that frequency. Per window its dimensionality is 1. And Spectral Contrast is an alternative to MFCC features which works well to do genre classification task as described in paper [29]. Per window its dimensionality is 7. So, combining all above features, total we have 20 (MFCCs) + 12 (Chroma-STFT) + 1 (Spectral Centroid) + 7 (Spectral Contrast) = 40 features.

So, our dataset has total 5000 examples (after augmentation), each example has sequence length of 256 timestamps and each timestamp has dimensionality = 40 (Table 1).

3.3. Multi-Layer Perceptron/Perception (MLP):

In honor of what is perhaps the most practical variety of neural network, the area of artificial neural networks is frequently referred to as neural networks or multi-layer perceptions. The single neuron model known as a perceptron served as a model for bigger neural networks.

It is a subject that studies how basic biological brain models may be utilized to tackle challenging computing problems, such as the predictive modeling problems we encounter in machine learning.

A neural network's hyperparameters, which include the number of layers and neurons, need to be tuned. To determine appropriate values for these, cross-validation techniques must be applied.

Backpropagation is used for weight correction training. Deeper neural networks do better at data processing. However, deeper layers might result in vanishing gradient.

Figure 8 shows fully connected multi-layer neural network.[7]

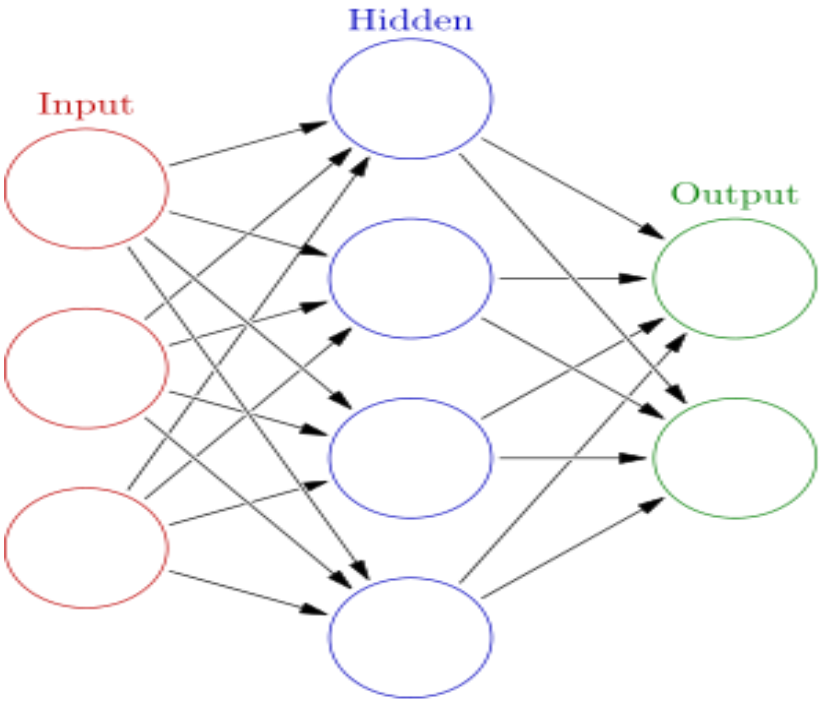


Figure 8: Multi-Layer Perceptron

Input Layer:

The visible layer refers to the portion of the network that is exposed, which is the lowest layer that receives input from your dataset. A neural network is frequently illustrated with a visible layer and one neuron for each input value or column in your dataset. These are not neurons, but rather merely transmit the input value to the next layer.

[7] MLP from MLP by Ritchie Vink

Hidden Layer:

Due to the fact that they are not immediately visible to the input, layers that come after the input layer are known as hidden layers. The most basic network topology is a single neuron in the hidden layer that outputs the value directly.

Very deep neural networks can be built with increased computer power and efficient libraries. Deep learning may be defined as having many hidden layers in neural network.

Output Layer:

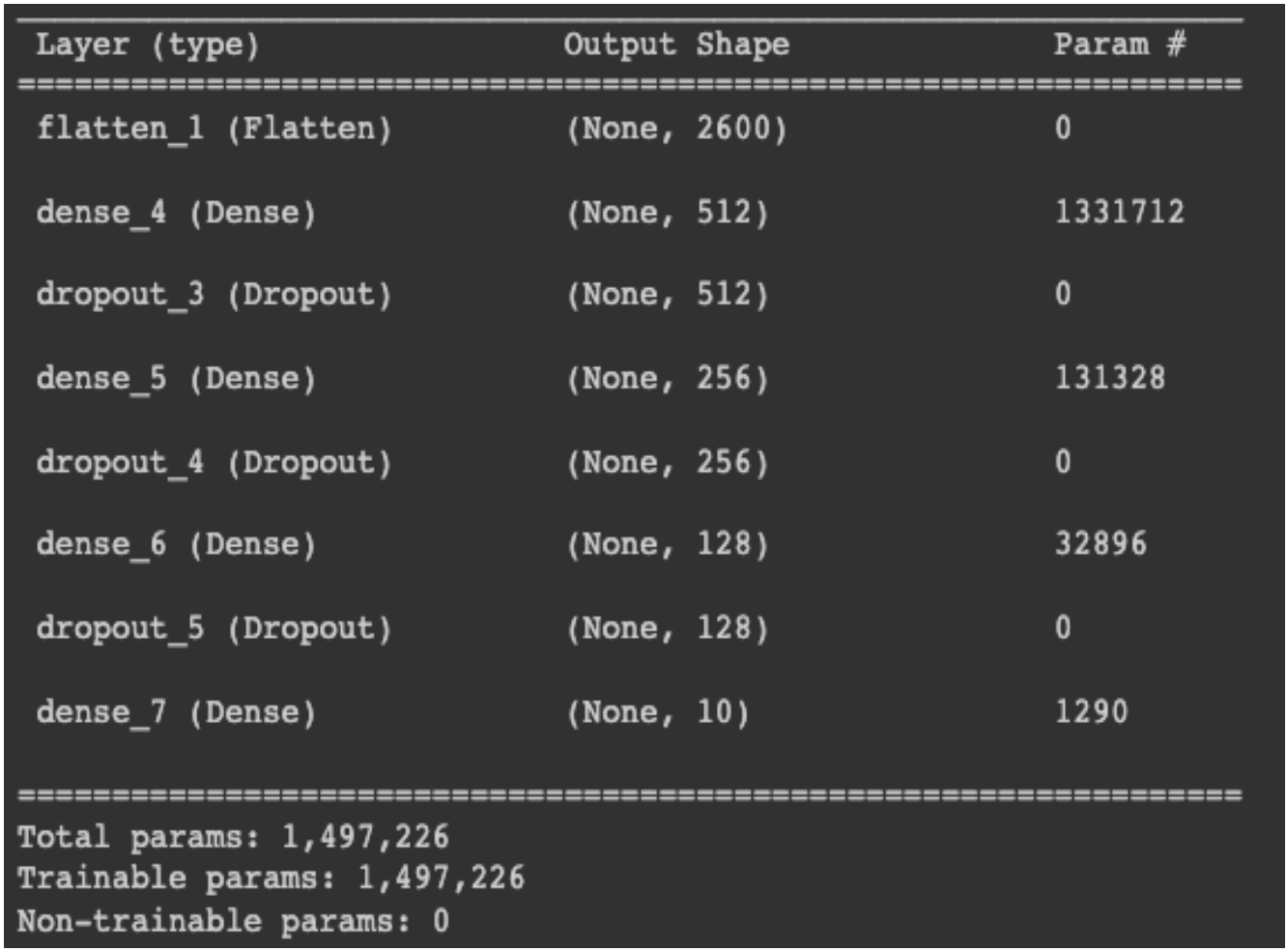
The last layer is known as the output layer, and it is in charge of producing a value or vector of values that conform to the problem's format.

The sort of problem that you are modeling has a significant impact on the activation function that you select for the output layer. As an example:

- A regression problem may have a single output neuron with no activation function.
- A single output neuron in a binary classification issue may utilize a sigmoid activation function to output a number between 0 and 1 to reflect the chance of predicting a value for class 1. By using a threshold of 0.5 and snap values lower than 0, otherwise 1, this may be converted into a crisp class value.
- In the output layer of a multi-class classification issue, there may be many neurons. In this situation, a SoftMax activation function might be used to determine the probability of the network correctly predicting each of the class values. A crisp class classification value can be obtained by selecting the output with the highest probability.[30]

In our experiment, we utilize sequential model with the help of keras library, in which other than input layer and output layer there are three hidden layers with Relu activation function and output layer is multiclass classification hence we are using SoftMax activation function. To prevent the models from overfitting I used both l2 regularization and dropout. 30% of the total data set was set aside as validation data.

Figure 9 shows the architecture of the MLP model. We took 20% test data and 80% Training data. For hidden layers we took 512,256 and 128 neurons respectively along with relu activation, l2 regularization and 0.3 dropout. Output layer has 10 neurons and SoftMax activation function. Also, we use Adam optimizers with learning rate of 0.0001.



Layer (type)	Output Shape	Param #
flatten_1 (Flatten)	(None, 2600)	0
dense_4 (Dense)	(None, 512)	1331712
dropout_3 (Dropout)	(None, 512)	0
dense_5 (Dense)	(None, 256)	131328
dropout_4 (Dropout)	(None, 256)	0
dense_6 (Dense)	(None, 128)	32896
dropout_5 (Dropout)	(None, 128)	0
dense_7 (Dense)	(None, 10)	1290

Total params: 1,497,226
Trainable params: 1,497,226
Non-trainable params: 0

Figure 9: Architecture of MLP Model

3.4. Convolution Neural network (CNN):

CNNs are a type of deep neural network that can identify and classify specific features from images. This is a common task used for analyzing visual images. CNN uses alternating convolution and pooling in the neural network to improve accuracy.

Application of kernels or grids (multidimensional weights) to the input picture in the network is known as convolution. In our case, the kernel and spectrogram will be combined with a dot product, and the resultant grid will be given to the following layer. CNN may employ kernels as feature detectors since they can identify the type of MFFC's over a given period of time.

By pooling, kernel-identified features are collected as the picture is downsized. The size of the feature maps is shrunk by pooling layers. As a result, it lessens the quantity of network computation and the number of parameters that must be learned. An area of the feature map produced by a convolution layer is summarized by the pooling layer's features.

The Fully Connected (FC) layer, which includes the weights and biases as well as the neurons, is used to link the neurons between layers. These layers are often placed prior to the output layer and form the final few levels of a CNN Architecture.

The input picture from the layers is also flattened and supplied to the FC layer. After being flattened, the vector passes through a few more FC levels, where the normal processes for mathematical functions happen. The classification procedure starts to take place at this point. Two layers are linked because two completely connected layers outperform a single connected layer. These CNN layers lessen the need for human oversight.

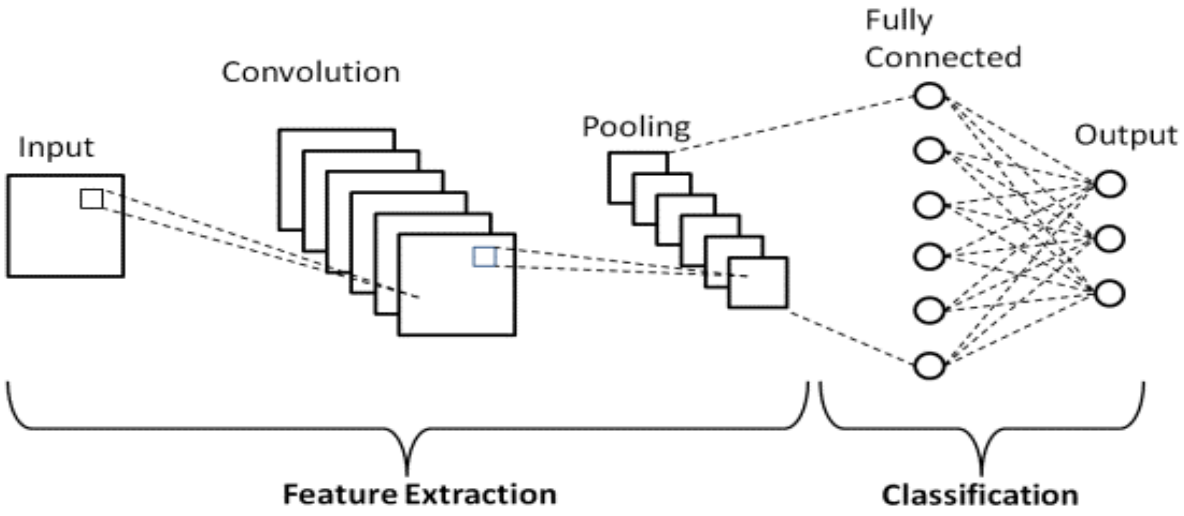


Figure 10: CNN Architecture[8]

Here Figure 10 shows the basic architecture of Convolution Neural Network or ConvNet.

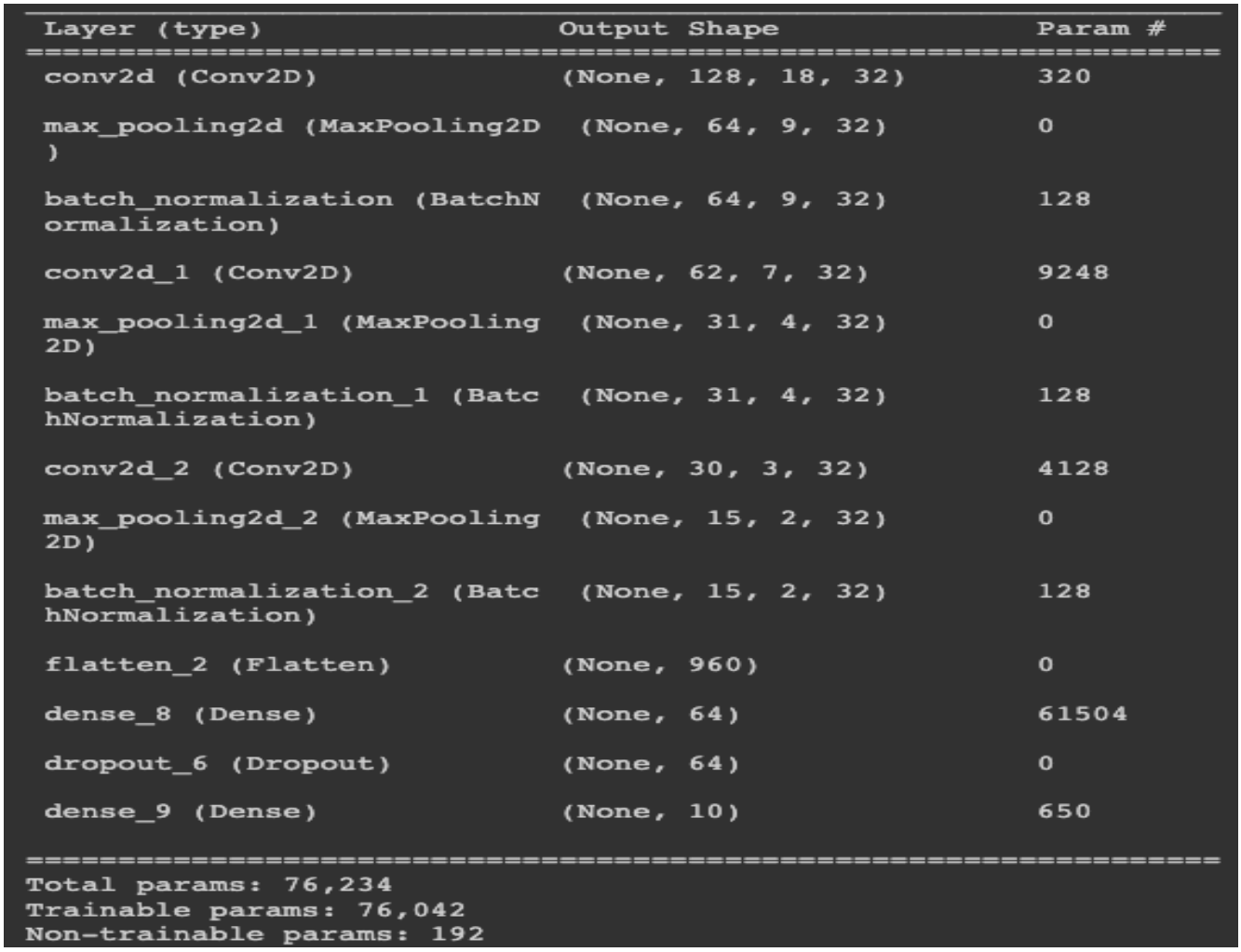
[8] CNN architecture by Sai Balaji

Due to its ability to handle spatial information, convolutional neural networks do well with images. A picture includes two-dimensional information. Because the input nodes of a CNN are not fully linked to every layer, the model may handle enormous input data without drastically increasing the number of nodes necessary. There may be a lot of data in an image because to its high megapixel count. These CNN features, in my opinion, are also useful for our classification issue with music visuals.

In order to provide recommendations, a variety of tasks are employed in music genre classification, including user-item latent feature prediction, music tagging, and genre classification. Convolutional neural networks are used to identify these tasks. To extract the source, CNNs always process the data at various levels of the feature hierarchy.

The information acquired by the hierarchical sources is used to complete the assigned functions that are learnt during supervised training. Instead of being totally linked to the layers, weights and biases in a convolutional neural network are shared. This implies that a cat appearing in an image's left corner will produce the same results as a cat appearing in its right corner. A given rhythm at the start of a song can cause the same neurons to fire as that exact beat at the end of the song.

As previously stated in Figure 5, the x direction displays time data, whereas the y direction displays frequency coefficients. I don't have to restructure the picture matrix in a 1D format when using CNN's.



Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 128, 18, 32)	320
max_pooling2d (MaxPooling2D)	(None, 64, 9, 32)	0
batch_normalization (BatchNormalization)	(None, 64, 9, 32)	128
conv2d_1 (Conv2D)	(None, 62, 7, 32)	9248
max_pooling2d_1 (MaxPooling2D)	(None, 31, 4, 32)	0
batch_normalization_1 (BatchNormalization)	(None, 31, 4, 32)	128
conv2d_2 (Conv2D)	(None, 30, 3, 32)	4128
max_pooling2d_2 (MaxPooling2D)	(None, 15, 2, 32)	0
batch_normalization_2 (BatchNormalization)	(None, 15, 2, 32)	128
flatten_2 (Flatten)	(None, 960)	0
dense_8 (Dense)	(None, 64)	61504
dropout_6 (Dropout)	(None, 64)	0
dense_9 (Dense)	(None, 10)	650

Total params: 76,234
Trainable params: 76,042
Non-trainable params: 192

Figure 11:CNN model

Figure 11 is our CNN model where we have split the data into test, validation and training in 15%,15% and 70% respectively. We build the sequential model using keras library in which we will have three conv layers with 32 kernel of grid size (3,3) and activation function Relu. Alternatively, MaxPool layer is added to the model with pool size (3,3), strides (2,2) and padding is same followed by a flatten layer. Usually, FC used as a connection between conv and dense layers.

Dropout

Normally, overfitting in the training dataset might result from all features being linked to the FC layer. When a given model performs so well on training data that it has a detrimental effect on the model's performance when applied to fresh data, this is known as overfitting.

To solve this issue, a dropout layer is used, in which a small number of neurons are removed from the neural network during training, reducing the size of the model. Fifty percent of the nodes in the neural network are randomly removed upon passing a dropout of 0.5.

A machine learning model performs better thanks to dropout since it reduces overfitting by simplifying the network. During training, neurons are removed from the neural networks.

The output layers are typically made up of the dense layers. The activation that is employed is called "SoftMax," and it provides probabilities for each class that add up to 1. Based on the class with the highest likelihood, the model will make its forecast.

Activation Functions

They are employed to discover and approximation any type of continuous and complicated link between network variables. It determines which model information should shoot forward and which information should not at the network's end.

The network gains nonlinearity as a result. The ReLU, SoftMax, tanH, and Sigmoid functions are a few examples of frequently used activation functions. Each of these operations has a particular use. SoftMax is typically used for multi-class classification. To put it simply, activation functions in a CNN model decide whether or not to activate a neuron. Through the use of mathematical operations, it determines whether the input to the work is significant or not for prediction.

3.5. Long Short-Term Memory Networks:

RNNs, also known as recurrent neural networks, have significantly improved predictive analytics. RNNs are a highly essential type of neural network that is widely utilized in Natural Language Processing. They are a type of neural network that allows prior outputs to be utilized as inputs while having hidden states.

The idea of "memory" in RNN refers to the ability to retain all data related to calculations made up to time step t. RNNs are referred to as recurrent because they do the same task for each element of a sequence, with the result dependent on past calculations.

A RNN's primary characteristic is that it has at least one feed-back link, allowing activations to circulate in a loop. This allows networks to do temporal processing and learn sequences. In sequential jobs, the existing input data is dependent on the previous applied inputs.

RNN attempts to address sequential data by processing each data point in a context that looks back to determine what to anticipate. As a result, RNN is thought to be suitable for processing audio music since audio can be thought of as a time series with ordered points across time.

In contrast to a traditional deep neural network, RNN converts independent activations into dependent activations by applying the same weights and biases to all layers. Furthermore, RNN shares the same parameters throughout all phases. This demonstrates how, despite the fact that the inputs alter at each phase, we are still carrying out the same task. This drastically reduces the number of factors we must memorize. As a result, the complexity grows by adding parameters and memorizing past results by feeding each output into the next hidden layer.

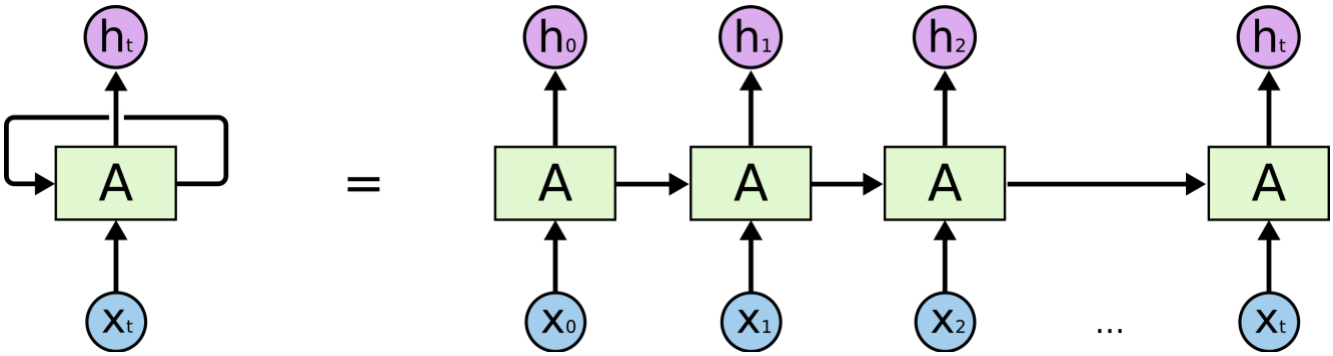


Figure 12: Recurrent Neural Networks [31]

Recursively employing the same cell in a sequential manner is known as calling a recurrent network. Since ReLU layer is not confined and might explode when employing it, RNN addresses the Vanishing Gradient problem. RNN is constrained to values between [-1, 1] by using the "tanh" function. The network is then trained using the backpropagation method, which propagates error across time by unrolling RNN steps and treating each time step as a layer in a feedforward network—calculating the error and back-propagate it through the early stages using the sequence-to-sequence methodology.

This allows us to stabilize the training process while also speeding it up. One limitation of RNN is that it cannot be used to peek into the far past. It does not have a long-term memory. It cannot learn long-term patterns with extensive dependencies, such as those seen in large audio files.

RNN is incapable of understanding and instead produces predictions while keeping track of the context. As a result, the LSTM network is a unique and successful form of RNN network. Despite this, RNN predicts the present state using stored data as information from the past. It fails to link the information when the gap between the present state and the prior state from which the information must be gathered is substantial.

The specifics of the long-term dependency were covered in LSTM architecture in Figure 14 [31].

LSTMs can learn long-term patterns with up to 100 steps but struggle with more than that. As with the prior CNN model, we began by preparing datasets (train, validation, and test sets). To develop an LSTM model, we remove all prior CNN layers and replace them with two LSTM layers with 64 LSTM units and one Dense layer using the ReLU activation function. Then, to prevent the above-discussed overfitting problem, we additionally included a dropout layer.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 130, 64)           21760

 lstm_1 (LSTM)               (None, 64)                33024

 dense (Dense)               (None, 64)                4160

 dropout (Dropout)           (None, 64)                0

 dense_1 (Dense)             (None, 10)                650

=================================================================
Total params: 59,594
Trainable params: 59,594
Non-trainable params: 0
```

Figure 13: LSTM Model

3.6. Hierarchical LSTM network:

To achieve multi-task classification, we have used a LSTM-based model. As we have already discussed, LSTM [32] is good technique to use for music genre classification as it remembers the past result of the cell in the recurrent layer and classify music in a better and efficient way. The equations for the LSTM modules are given as follows: -

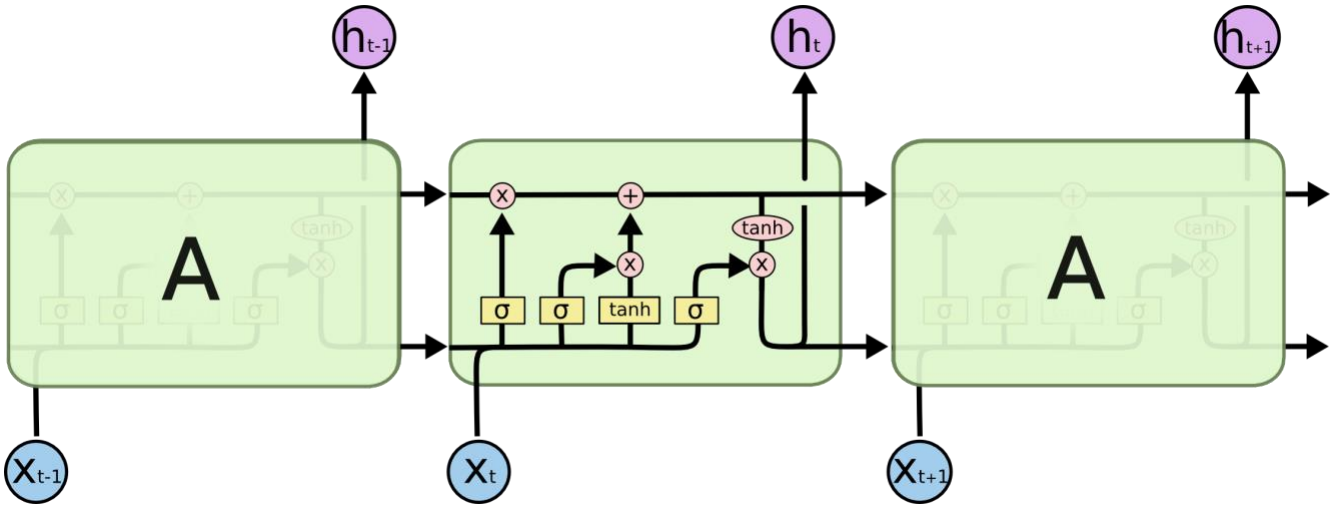


Figure 14: Architecture of LSTM model[9]

$u_t = \tanh (W_{xu} * x_t + W_{hu} * h_{t-1} + b_u)$: update equation

$i_t = \sigma (W_{xi} * x_t + W_{hi} * h_{t-1} + b_i)$: input gate equation

$f_t = \sigma (W_{xf} + x_t + W_{hf} * h_{t-1} + b_f)$: forget gate equation

$o_t = \sigma (W_{xo} * x_t + W_{ho} * h_{t-1} + b_o)$: output gate equation

$c_t = i_t \Theta u_t + f_t \Theta c_{t-1}$: cell state

$h_t = \tanh c_t \Theta o_t$: cell output

output class $= \sigma (h_t * W_{outpara})$

W_{xu} , W_{xi} ,W_{xf} ,W_{xo} and W_{hu} , W_{hi} , W_{hf} , W_{ho} , $W_{outpara}$ are weights; b_u , b_i , b_f , b_o are biases that will be computed during training. At time t, ht is the output of a neuron. Θ represents multiplication by points. σ is a sigmoid function, and tanh is the tanh function. The MFCC parameters at time t are represented by the input x_t. The classification output is the output class [33].

Instead of using a single LSTM network to perform a 10-class classification task, we use a divide and conquer approach, by using a hierarchical tree-based 7 LSTM network architecture. The following Figure 15 shows the proposed hierarchical LSTM architecture.

[9] LSTM architecture from https://colah.github.io/posts/2015-08-Understanding-LSTMs/

The functionality of each of the LSTM networks is given as follows:

• LSTM 1: It classifies between strong (hip-hop, metal, pop, rock, reggae) and mild (jazz, disco, country, classic, blues) genres of music

• LSTM 2a: It classifies between sub-strong 1 (hip-hop, metal, and rock) and sub-strong 2 (pop and reggae)

• LSTM 2b: It classifies between sub-mild 1 (disco and country) and sub-mild 2 (jazz, classic, and blues)

• LSTM 3a: It classifies between hip-hop, metal and rock

• LSTM 3b: It classifies between pop and reggae.

• LSTM 3c: It classifies between disco and country.

• LSTM 3d: It classifies between jazz, classic, and blues.

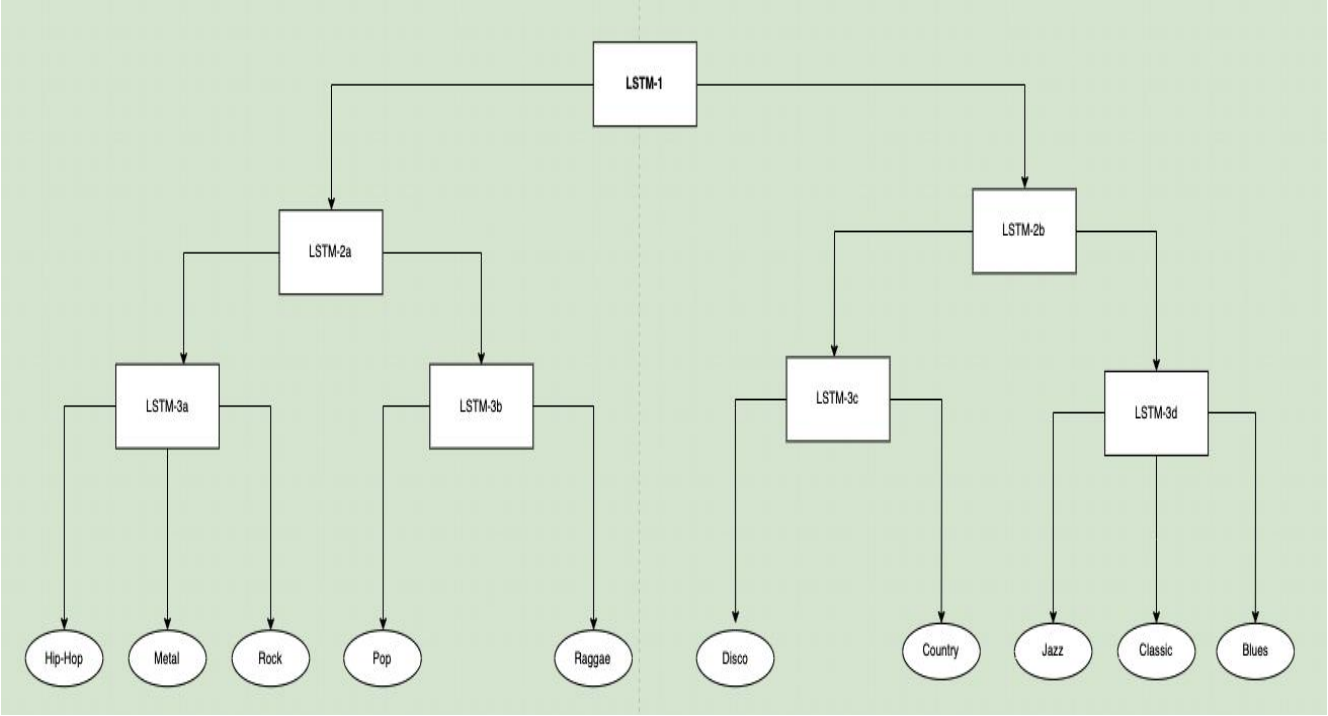


Figure 15: The Hierarchical LSTM architecture

This hierarchical architecture, helps to tackle the multi-class classification problem, by a divide and conquer based approach, where each LSTM in the tree, is trained using samples of the relevant classes. The idea for the hierarchical LSTM model was adopted from [33].

The architecture of the individual LSTMs is given in Figure 15:

LSTM Classifiers	**Total no. of samples**
LSTM1(strong, mild)	5000
LSTM2a (sub-stong1, sub-strong2)	2500
LSTM2b (sub-mild1, sub-mild2)	2500
LSTM3a (hip-hop, metal, rock)	1500
LSTM3b (pop, reggae)	1000
LSTM3c (disco, country)	1000
LSTM3d (jazz, classic, blues)	1500

Table 2: Total no of samples for each LSTM component

The dataset was split into 75 percent for train, 15 percent for validation and 10 percent for test sets, while applying the stratified property. The stratified property was applied as, it is preferable to divide the dataset into train and test sets in such a way that the proportions of instances in each class remain the same as in the original dataset, thus avoiding class imbalance.

Input Layer	Total 40 features (MFCC, Chroma STFT, Spectral Centroid, Spectral Contrast)
Hidden Layer I	64 LSTM units
Hidden Layer II	32 LSTM units
Output Layer	Number of SoftMax units depends on which the number of fan-out results.

Table 3: Hierarchical LSTM Model

4. Results

The code as well as the results are available at my GitHub repository: Audio-genre-classification

4.1. MLP

For MLP model, we had used the Adam optimizer for training the model. The train and test split are 75-25 respectively. Learning rate is 0.0001 and epoch for the training the model is 100.

The RELU activation function is used by all of the hidden layers, whereas the SoftMax function is used by the output layer. The sparse categorical cross entropy function is used to compute the loss.

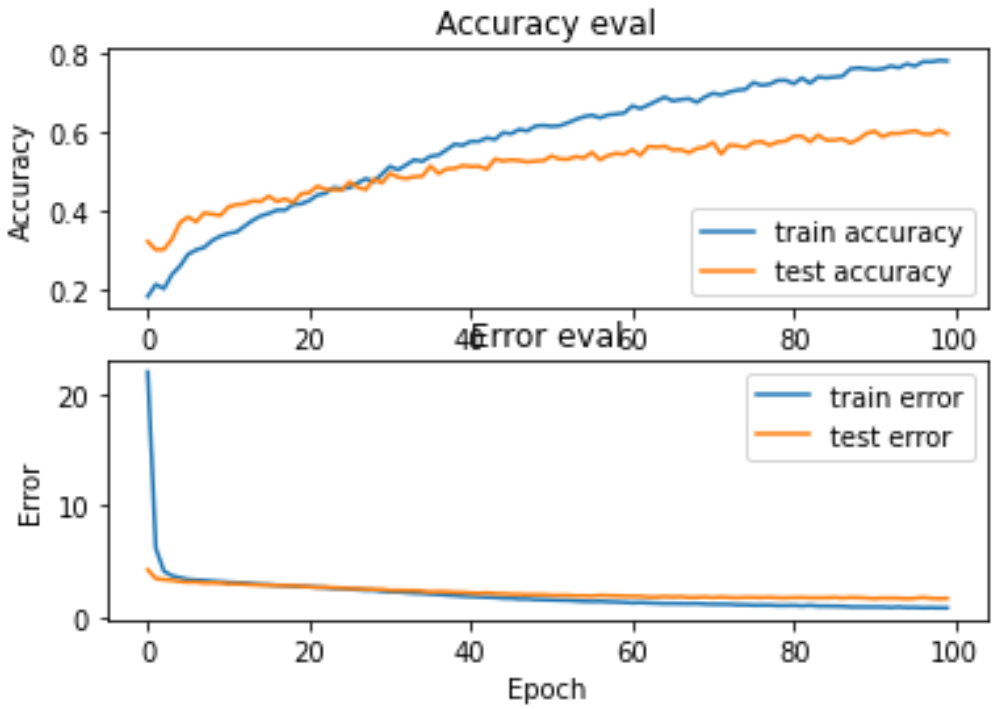


Figure 16: Result of Multi-Layer Perception Model

I have also added a dropout layer and l2 regularization to avoid the overfitting issues. In Figure 16, for the accuracy plot, we can notice that the training and testing sets tend to increase. The same thing is noticed for error plot where testing and training error are stabilized but training error slightly reduced more than the testing error.

No. of epochs=100	Training Accuracy	Testing Accuracy	Training Loss	Testing Loss
Training= 75% Testing= 25%	0.7312	05885	1.0693	1.7467

Table 4:Result of MLP Model

4.2. CNN

I trained the CNN model using the Adam optimizer. The ratio between training, validation and testing is 70-15-10. The learning rate is 0.0001 and the training epoch is 100.

Three layers of Conv2D, MaxPooling2D and batch normalization have been implemented. Then we flatten and feed forward to dense layer of 64 units with dropout of 0.5.

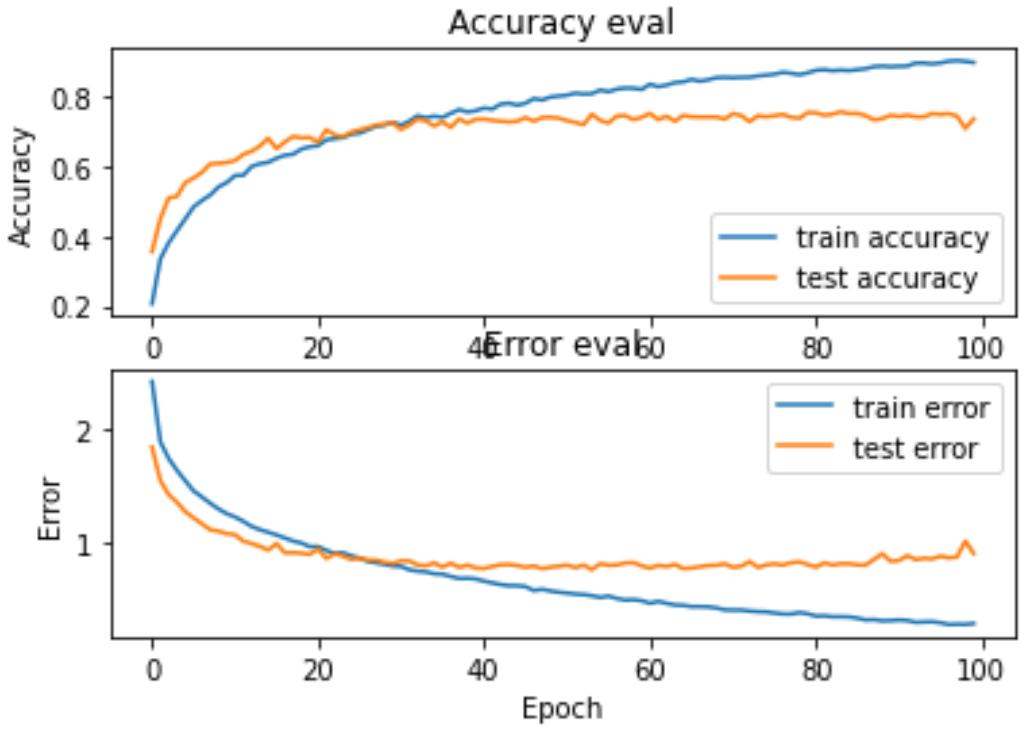


Figure 17: Result of CNN Model

The RELU activation function is used by all of the conv2D layers, whereas the SoftMax function is used by the output layer. We will use 32 kernels of size (3,3) and Max Pooling layer of size (3,3) for two layers of CNN. Third conv layer has 32 kernels of size (2,2) and Max Pooling has also size (2,2). We are using same padding and strides of (2,2) in MaxPool 2D layers. For Standardizing the activation functions of current layer, we are using Batch Normalization.

No. of epochs=100	Training Accuracy	Testing Accuracy	Training Loss	Testing Loss
Training= 75%, Validation= 15% Testing= 10%	0.8658	0.7354	0.2926	0.9042

Table 5:Result of CNN Model

4.3. RNN-LSTM

To develop an LSTM model, we remove all prior CNN layers and replace them with two LSTM layers with 64 LSTM units and one Dense layer of 64 units with the ReLU activation function. Then, to prevent the overfitting issue stated above, we added a dropout layer of 0.5. Output layer is again a dense layer with 10 units as there are 10 genres in our dataset and it is a multiclass classification, so SoftMax activation function is suitable.

We trained the LSTM model using the Adam optimizer. The validation and testing split is 15 and 10 respectively. The learning rate is 0.0001 and the training epoch is 50.

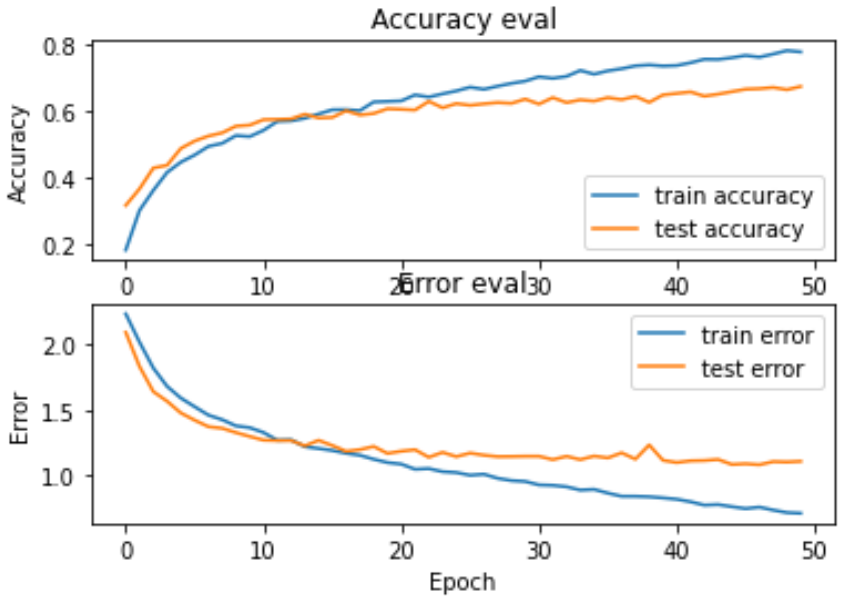


Figure 18: Result of RNN-LSTM Model

In the accuracy plot, we can see that the training and test sets tend to linearly rise, then sort of stable until epoch number 8, with a small improvement for the training set after that. The similar result is seen for the Error plot, where testing error where Error rate tends to keep falling until epoch 10 when testing error sort of stability but training error slightly dropped more than testing error.

No. of epochs=100	Training Accuracy	Testing Accuracy	Training Loss	Testing Loss
Training= 75%, Validation= 15% Testing= 10%	0.7757	0.6923	07161	1.1098

Table 6: Results of RNN-LSTM model

4.4. Hierarchical LSTM

The proposed multi-step classifier involves the 7 LSTMs shown in Figure 15. The input music is initially identified as being either strong or mild by LSTM1 during the testing phase. After that, either LSTM2a or LSTM2b is applied depending on the outcome. In accordance with the outcomes from the previous level, LSTM3a, 3b, 3c, or 3d are then utilized to categorize the music into the desired categories.

Figure 22 displays the results of this experiment. Our method yielded a 75.00% accuracy. It outperformed the convolutional neural network approach, which had an accuracy of 73.54%. The LSTM hierarchy in our multi-step classifier is depicted schematically in Figure 15.

The dataset was split into 75 percent for train, 15 percent for validation and 10 percent for test sets, while applying the stratified property. Using the stratified property, the dataset was divided into 75 percent train, 15 percent validation, and 10 percent test sets. The stratified attribute was used because it is preferable to divide the dataset into train and test sets while preserving the proportions of instances in each class that were seen in the original dataset, hence avoiding class imbalance. The total number of samples for LSTM1, LSTM2a, LSTM2b, LSTM3a, LSTM3b, LSTM3c, LSTM3d are 5000, 2500, 2500, 1500, 1500, 1000, 1000, 1500 respectively.

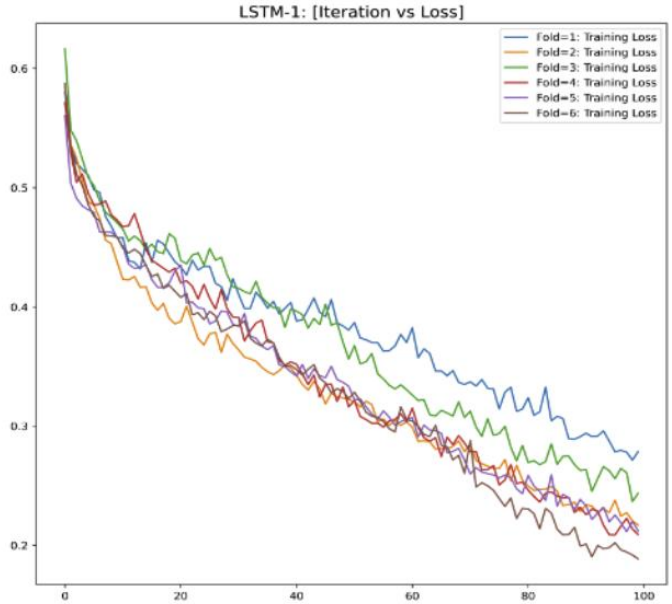


Figure 19: Iteration vs. Cost plot for LSTM-1

Figure 20, Figure 21 and Figure 21 shows the Iteration vs, Cost plot for the 7 LSTMs. Figure 22Figure *22* depicts the confusion matrix hard prediction, soft-prediction (Top-1) and soft-prediction (Top-2) respectively. As we can see that using hard prediction method, accuracy is as good as mentioned in [33]. Using soft prediction method, it improved from more than 10%. In the case of top-2 predictions if any of the two genres (classes) with the highest predicted probabilities matches with the ground truth label, the prediction is deemed as correct. The reason for the accuracy of top-2 predictions being much higher than the top-1 prediction can be attributed to the fact that some genres like pop and reggae, are hard to distinguish even by an avid music listener.

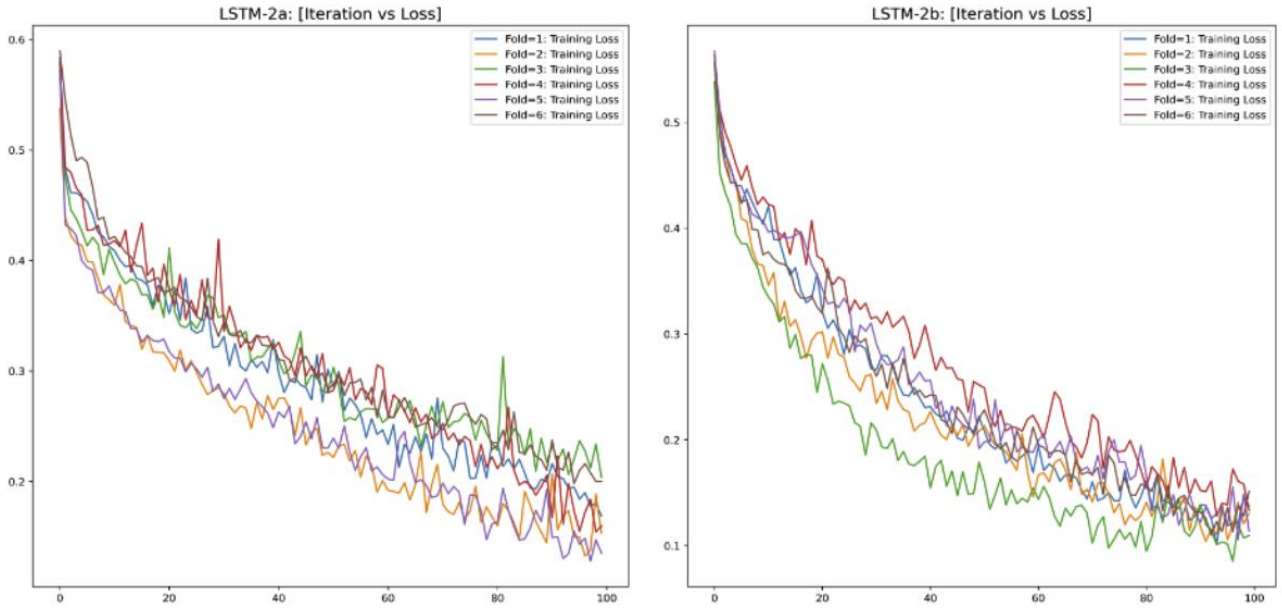


Figure 20: Iteration vs. Cost plot for LSTM-2a (left) and LSTM-2b (right)

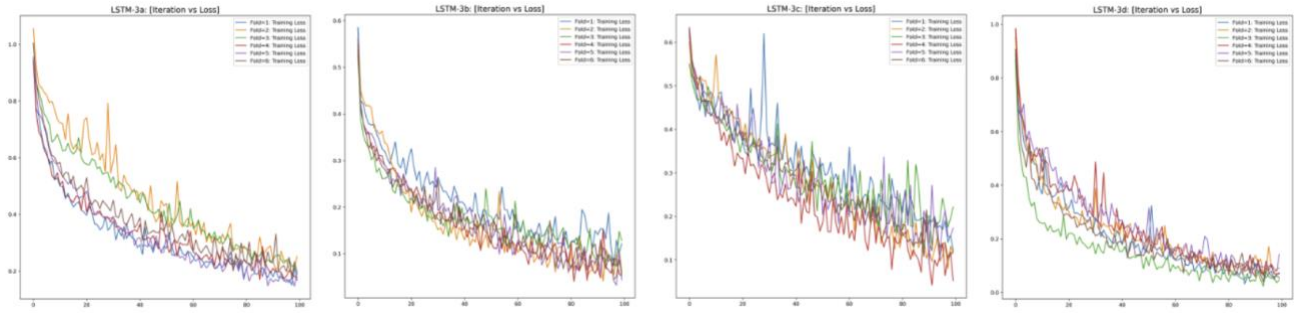


Figure 21: Iteration vs. Cost plot for (from left right) LSTM-3a, LSTM-3b, LSTM-3c and LSTM-3d

We have used two prediction methods:

Hard Prediction

- In this, we perform segmentation on the original track so that each segment will have 256 timestamps after feature extraction.
- Predict the class for each segment.
- Select label with highest frequency.
- For all segments, select the path with majority predicted label.
- Repeat for all internal nodes until leaf nodes are not predicted.
- Return the predicted label.

Soft Prediction

- In this, we perform segmentation on the original track so that each segment will have 256 timestamps after feature extraction.
- Predict the class for each segment.
- For each segment, choose path independently.
- Do this for all internal nodes until leaf nodes are not predicted.
- Return all predicted labels with the frequencies.

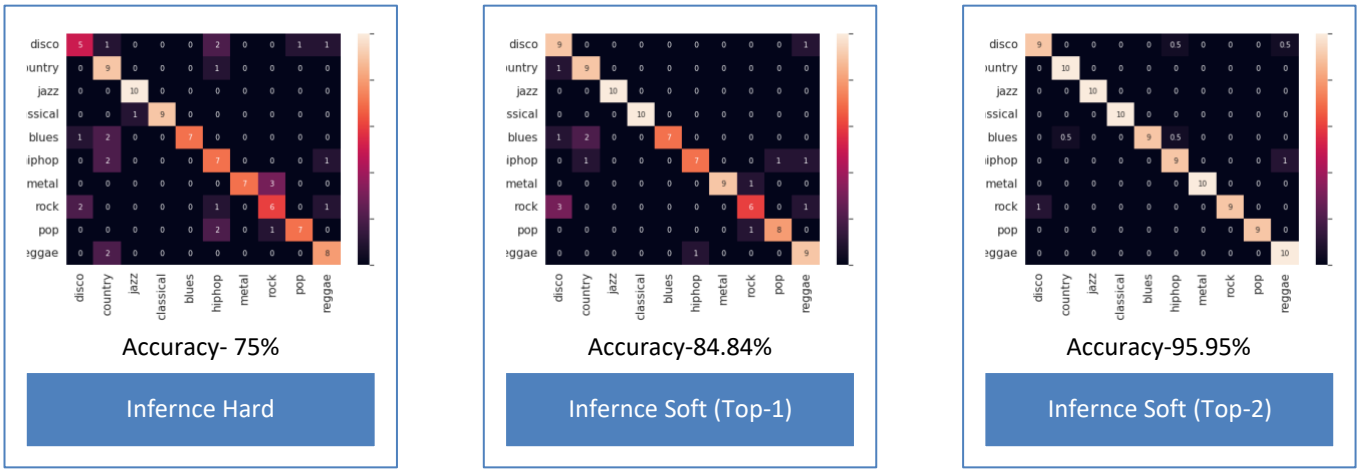


Figure 22: Result of Hierarchical LSTM model

5. Conclusion

The content-based music classification approach is being used in the industry widely. As there are numerous machine learning techniques that work well on extracting pattern, trends, or other useful information from a large dataset, thus these techniques are suitable for performing music analysis. Several companies are using music classification to segment their database according to genre and are either using it to recommend songs to their users like done by Spotify, YouTube Music etc. or even purely as a product like Shazam.

Evaluation Metric	Multi-Layer Perception	CNN with Max-Pooling	RNN with Long Short-Term Memory	Hierarchical LSTM
Training Accuracy	73.12	86.58	77.57	Inference Hard-75% Inference Soft- Top-1 – 84% Top-2 – 95%
Testing Accuracy	58.85	73.54	69.23	

Table 7:Performance comparison of different Models

In conclusion, the experimental results show that our multi-step classifier based on Long Short-Term Memory (LSTM) model is effective in recognizing music genres. I employed a divide-and-conquer strategy to classify ten different genres. We reached an accuracy of 75.00%, which was higher than one of the SOTA approaches, which had an accuracy of 73%.

Even after dealing with overfitting difficulties, CNN with the Max-Pooling function beat both RNN with the LSTM model and the conventional multi- layer model with three hidden layers. Our experiment is carried out on a MacBook Pro with M1 chip running on a mac operating system. Compared to the multilayer design, CNN has the disadvantage of being more costly. As a result, the suggested multilayer model is faster than the CNN model. We run RNN with the LSTM model across 50 epochs, and each epoch takes substantially longer to build than any prior model.

6. Future Work

A shortcoming of content-based approach is that it fails to account for the cultural as well as contextual meaning of the item (in our case song) and only look for similarity between the items. Also, with the content-based approach, there is little room left for surprise as out of the box recommendations are not done with this approach which can help the user to explore his/her taste.

We plan to integrate our content-based approach with the collaborative filtering method so as to overcome the above two shortcomings. This will also provide us with the added benefit of using the information collected from the community.

We also plan to introduce a feedback mechanism which would allow us to draw in feedback from the users and thus improving the current approach even further. This methodology has suggested a few intriguing questions for further research. The distribution of the songs into genres is believed to be the fundamental disadvantage of our work. The training is the most crucial aspect of the entire operation. As a result, a better ordered data collection may be critical for classification outcomes.

Another option is to experiment with additional features, as various features have varied effects on classification performance. Furthermore, we may combine and other characteristics with those already provided to improve classification results. A combination of the elements is likely to produce better outcomes.

Another issue with automated music genre classification is the design of the classifier. As a result, future study might include the use of alternative classification algorithms or fusion with other techniques of music classification. Different classification approaches may produce superior results.

7. Works Cited

[1] Lee, J. H., & Downie, J. S. (2004). Survey of music information needs, uses, and seeking behaviours: preliminary findings. Proceedings of the international conference on music, information retrieval.

[2] North, A. C. (1997). Liking for musical styles. Music Scientae, 1 (1) (1997), pp. 109-128.

[3] TEKMAN, H. G., & HORTACSU, N. (2002). Aspects of stylistic knowledge: what are different styles like and why do we listen to them? Psychol. Music, 30 (1) (2002), pp. 28-47.

[4] J. Ben Schafer, Dan Frankowski, Jon Herlocker, and Shilad Sen. Collaborative Filtering Recommender Systems, pages 291–324. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

[5] Charu C. Aggarwal. Knowledge-Based Recommender Systems, pages 167–197. Springer International Publishing, Cham, 2016.

[6] Panagakis Y, Kotropoulos C, Arce GR. Music genre classification via sparse representations of auditory temporal modulations. In: 17th European signal processing conference (EUSIPCO); 2009.

[7] Dannenberg RB, Thom B, Watson D. A machine learning approach to musical style recognition. In: Proceedings of the international computer music conference; 1997

[8] Li T, Ogihara M, Li Q. A Comparative study on content-based music genre classification. In: Proceedings of the 26th annual international ACM SI-GIR conference on research and development in information retrieval. Toronto: ACM Press; 2003.

[9] McKay C, Fujinaga I. Musical genre classification: is it worth pursuing and how can it be improved? In: 7th Int conf on music, information retrieval (ISMIR-06); 2006.

[10] G. Tzanetakis, P. Cook Musical genre classification of audio signals. IEEE Trans Speech Audio Process, 10 (5) (2002).

[11] R.O. Duda, Peter E. Hart, David G. Stork Pattern classification. (2nd ed.), John Wiley & Sons (2001)

[12] Paradzinets A, Harb H, Chen L. Multiexpert system for automatic music genre classification. <http://liris.cnrs.fr/Documents/Liris-4224.pdf>. Research report; 2009.

[13] C.N. Silla Jr., A.L. Koerich, C.A.A. Kaestner, A machine learning approach to automatic music genre classification, J Braz Comput Soc, 14 (3) (2008)

[14] Ezzaidi H, Rouat J. Automatic musical genre classification using divergence and average information measures. Research report of the world academy of science, engineering and technology; 2006.

[15] L. Deng, D. O'shaugnessy, Deng Deng Speech processing: a dynamic and optimization-oriented approach Marcel Dekker (2003)

[16] Bahuleyan, Hareesh. "Music genre classification using machine learning techniques". In: arXiv preprint arXiv:1804.01149 (2018).

[17] Chun Pui Tang, Ka Long Chui, Ying Kin Yu, Zhiliang Zeng, and Kin HongWong. Music genre classification using a hierarchical long short-term memory (lstm) model. In Third International Workshop on Pattern Recognition, volume 10828, page 108281B. International Society for Optics and Photonics, 2018.

[18] GTZAN dataset for Music genre Classification - http://marsyas.info/download/data_sets or https://www.kaggle.com/datasets/andradaolteanu/gtzan-dataset-music-genre-classification .

[19] B. L. Sturm, "A survey of evaluation in music genre recognition," in Proc. Adaptive Multimedia Retrieval, Copenhagen, Denmark, Oct. 2012.

[20] Million Song Dataset - https://www.kaggle.com/c/mlp2016-7-msd-genre

[21] Spotify Classification - https://www.kaggle.com/c/cs9856-spotify-classification-problem-2021

[22] Indian Music Dataset - https://www.kaggle.com/winchester19/indian-music-genre-dataset

[23] W. Walker, P. Lamere, P. Kwok, B. Raj, E. Gouvea, P. Wolf, and J. Woelfel. Sphinx-4: A flexible open-source framework for speech recognition. cmus-phinx.sourceforge.net /sphinx4/doc/Sphinx4Whitepaper.pdf, (2004)

[24] J. T. Foote. Content-based retrieval of music and audio. In SPIE, (1997)

[25] B. Logan. Mel frequency cepstral coefficients for music modelling. In Intl. Symposium on Music Information Retrieval, (2000)

[26] B. Logan and S. Chu. Music summarization using key phrases. In IEEE Conf. on Acoustics, Speech and Signal Processing, (2000)

[27] L. R. Rabiner and B. H. Juang. Fundamentals of Speech Recognition. Prectice-Hall, (1993)

[28] Jiang, Dan-Ning, Lie Lu, Hong-Jiang Zhang, Jian-Hua Tao, and Lian-Hong Cai. "Music type

classification by spectral contrast feature." In Multimedia and Expo, 2002. ICME'02. Proceedings. 2002 IEEE International Conference on, vol. 1, pp. 113-116. IEEE, 2002.

[29] Chang-Hsing Lee, Jau-Ling Shih, Kun-Ming Yu, and Jung-Mau Su. Automatic music genre classification using modulation spectral contrast feature. In 2007 IEEE International Conference on Multimedia and Expo, pages 204–207. IEEE, 2007.

[30] Jason Brownlee, Multi-Layer Perceptrons Neural Networks

[31] Christopher Olah. Understanding of LSTM networks in https://colah.github.io/posts/2015-08-Understanding-LSTMs/ , 2015

[32] Douglas Eck and Juergen Schmidhuber. Finding temporal structure in music: Blues improvisation with lstm recurrent networks. In Proceedings of the 12th IEEE workshop on neural networks for signal processing, pages 747–756. IEEE, 2002.

[33] Chun Pui Tang, Ka Long Chui, Ying Kin Yu, Zhiliang Zeng, and Kin HongWong. Music genre classification using a hierarchical long short-term memory (lstm) model. In Third International Workshop on Pattern Recognition, volume 10828, page 108281B. International Society for Optics and Photonics, 2018.

Appendix

Libraries Used

- Librosa - Librosa is a package used in python for music/audio analysis. We used Librosa to extract features from the GTZAN dataset.
- TensorFlow - TensorFlow is a very popular open-source platform widely used in Machine Learning. It comprises of numerous tools, libraries etc. which helps in development purposes.
- Matplotlib - Matplotlib is a python library that is used for building static as well as dynamic graphs.
- Numpy - Numpy is used for carrying out mathematical and scientific computations.
- Os – os module allows to interact with underlying operating system.
- Math – math is a built-in module to use mathematical tasks.
- Sklearn – sklearn is one of the most robust libraries for machine learning in python
- Scipy – SciPy is a collection of mathematical algorithms and convince functions built on the NumPy extension of python.
- Seaborn – Seaborn is data visualization library based on matplotlib.
- Pandas – Pandas is one of the most popular libraries in python for data analysis.
- Pydub – pydub is a pyhton library to work with only .wav files.