Music Genre Classification

Areeb Khan Shabih A20469525 and Carmen Acero Vivas A20472656

ashabih@hawk.iit.edu, cacerovivas@hawk.iit.edu

Abstract **- Music Genre Classification is a task within the field of Music Information Retrieval. Over time, different techniques have been developed to perform this task, starting from classical algorithms followed by the use of deep learning techniques in the last few years. In the present project, we discuss how prepare the data when dealing with audio files, we evaluate the different features that can be extracted. Then, we analyze different classical algorithm, such as SVM or XGBoosting and some Deep Learning techniques, such as CNN or RCNN. We will use the GTZAN dataset, the most popular database in this area, for all the models in order to be able to compare them with each other**.

***Keywords*: Music Genre Classification, Machine Learning, GTZAN**

1 Problem Description

With the exponential growth of online music streaming services, it has become a challenging task both for users and the service providers to maintain music library manually. Accurate and automated music classification to different genres using machine learning algorithms can not only solve this problem for music streaming companies like Spotify and iTunes and improve user experience as well.

Musical Information Retrieval (MIR) is the science of retrieving information from music. The information can be used for different research topics like genre classification, recommender systems, instrumental recognition or sentiment analysis. In this project we will focus on music genre classification, a field where it can be found rich and varied literature. Different approach has been proposed to solved this task, going from classic algorithms to deep learning models.

Nearly 40,000 tracks get added to Spotify along everyday. A robust classifier is essential to characterize the music and tag unlabelled music. At the same time, lack of standardization and ambiguous boundaries between music genres make this a very challenging task. Moreover, human perception change depending on their experiences and opinion and this idea stands out especially when talking about art, particularly in music. This leads to an absence of a global data-set that can be used to obtain reproducible results. Each data-set has its unique structure, presenting different number of genres and descriptors.

Through our project, we will dive and explore the shallow learning and deep learning approaches to perform classification. We will understand, implement and leverage the work already done for feature extraction and lay majority of the focus towards the actual classification problem. Starting from diverse shallow learning algorithms like SVC, Random Forest or XGBoost. Then, we'll go on to explore the deep learning techniques like CNN and RNN for music classification. The implementation and design of these models will be driven by the type of data available, classification goal and our understanding of the problem. Towards the end of our project we'll do a comparative analysis and see which algorithm performs the best job.

2 Background

The study on this field started in 2002 with Tzanetakis and Cook [1], where the authors extracted 30 features from the audio files to predict 10 different music genres. They also build the GTZAN dataset [2], which has become the most well known dataset. However, this dataset present inconsistencies and is relatively small, with only 1000 songs. Other available and public datasets such as FMA dataset [3] counts with 0.1 million songs, which enables more complex models, that requires a greater amount of data, to be trained. We can also find MillionSong Dataset (MSD) [4] or AudioSet [5]. Historically, this challenge started getting addressed by employing classical methods. The most commonly used classical methods included decision trees, an appealing approach as this is a classification task and trees have proved to show good results [1], probabilistic classifiers, such as Naïve Bayes Classifier [1] and Support Vector Machine (SVM) [1]. These models relied basically on three categories of features; rhythm, pitch and

temporal structure. However, recently using audio spectrogram has become popular for music genre classification. Spectogram encodes time and frequency information of the music track in its entirety [2]. While it is true that the classical algorithms have shown good results, over the course of time, deep learning techniques have demonstrated outstanding results, close to human level of accuracy. Within this field, there exist literature using Fully Connected Neural Networks which potential relies when the audio file is introduced as input with some extra information so the problem becomes richer and more suitable for this type of deep models. Convolutional Neural Networks [2] using spectrogram results as an input have achieved great results. Since the beat, rhythm, frequencies and tone of the music changes with time, Recurrent Neural Networks, have been recently used to model temporal information and sequences. The most popular architecture that has been used in this field is Long-short term memory (LSTM) [6].

3 GTZAN Dataset: Description and feature extraction

After researching different databases we have chosen to use GTZAN. GTZAN is the most popular dataset and most-used public dataset for this type of task. The main reason for using this dataset is to find a common ground to make our algorithms comparable to the research that has already been done.

GTZAN has total 10 different genres, each containing 100 audio clips that are 30 seconds long. Therefore, the dataset consists of 1000 total audio-clips. It comprises of following 10 different genres.

1. Blues
2. Classical
3. Country
4. Disco
5. Hiphop
6. Jazz
7. Metal
8. Pop
9. Reggae
10. Rock

Each instance is a 30 second audio file in .wav format. The instances are audio files, therefore, they should be preprocessed before feeding them into any algorithm we want to train. There are different approaches to audio processing. In the following section, we will show how to carry out each of them and the results obtained. In addition, we will compare each of the genres after processing them.

4 Data/AudioPre-processing

An audio signal should be converted into valid data before inputting it to any algorithm. For that, we will have to look for different characteristics and study if they are distinguishable enough so that they can be used as attributes/features to train our algorithms.

There exist different types of transformations, among which we will study the three following ones:

- Mel-spectrograms (MFC)
- Chroma features
- Mel-frequency cepstrum (MFCC'S)

Audio signals can be transformed to time domain, frequency domain (with Fourier) and time-frequency domain (a combination of the two above, which allows less information to be lost). First, we will overview the time domain features of an audio file. We can represent an audio by plotting the amplitude vs time. We will plot the amplitude feature for different genres and analyze if there are differences.

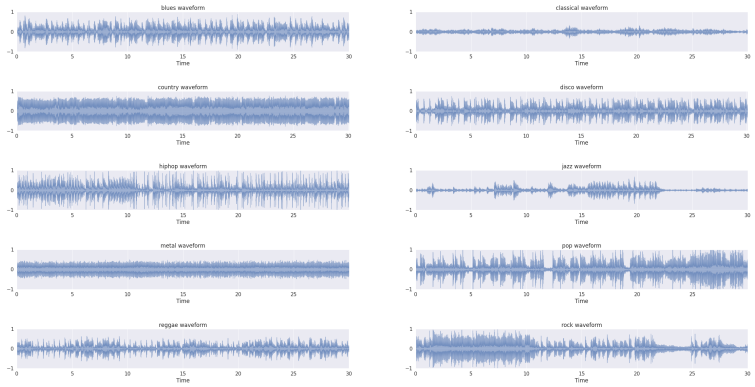


Figure 1. Amplitude vs time for one sample per genre .

As it can be seen in Figure 1, although for some genres the curves are sufficiently differentiable, for some others this task becomes more challenging. For example, disco, pop and hip hop share a common structure, high peaks, that correspond to drums. This feature can help when considering the instruments and/or voices separately, since the waveform of each one of them is different, but when analyzing a song where different instruments and voices are mixed this feature is no longer beneficial.

Also, this type of audio feature extraction is that it is sensitive to outliers. So, we look to other types of features that can be more representative.

4.1 Chroma features

Chroma features represent the 12 semitones (or chroma) versus time. The twelve semitones, also known as pitch classes, are C, C, D, D, E, F, F, G, G, A, A, B. When we plot this feature, in Figure 2, we will observe 12 different boxes stacked on top of each other for each period of time, these boxes represent the pitches. The contribution of each semitone class is represented by the color intensity.

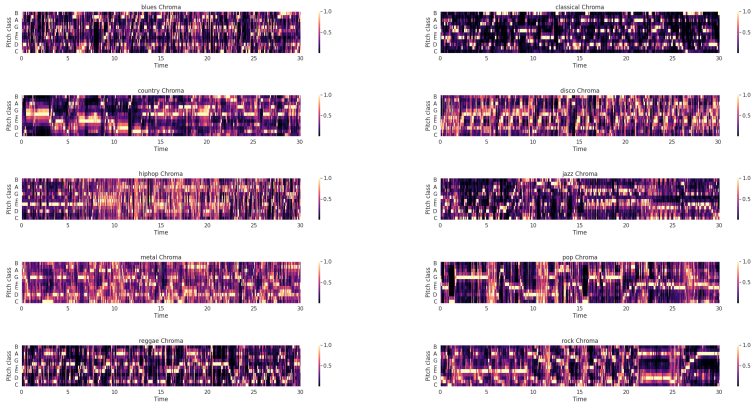


Figure 2. Chroma for one sample per genre .

Chroma features have turned out to be a powerful mid-level feature representation in content-based audio retrieval such as cover song identification. However, the majority of the genres share common pitches, therefore, leading to ambiguity.

4.2 Short-time Fourier transform (STFT)

Fourier transforms converts the audio signals from time domain to frequency domain. It allows us to decompose the signal into individual frequencies it's made of and the corresponding amplitude of those frequency levels. Short-time Fourier transform (STFT) is a sequence of Fourier transforms of a time windowed signal. It allows us to represent the

spectrum of the audio signals as they vary over time. STFT empowers us to discover sensitive information related to frequency in a localized time domain and it gives very useful insights when we are dealing with non-periodic signals. It converts the signal such that we know the amplitude of the given frequency at a given time.

4.3 Mel Spectrograms and Mel frequency cepstral coefficients (MFCC) Features

MFFC features are derived from the Cepstral representation of an audio clip. Cepstral representation is derived by taking inverse fourier transform of the logarithm of the estimated signal. It is useful for investigating periodic trends in frequency spectrum. In Mel frequency cepstrum or MFC, the frequency bands are mapped onto Mel's scale, which approximates the human perception of audio extremely well in comparison to the untransformed frequency bands. Also, MFCC features are very robust to the presence of additive noise and enables the algorithm to get rid of those features which adversely affect the classification of signal. In simple words, MFC is a transformation of STFT on a Mel's scale.

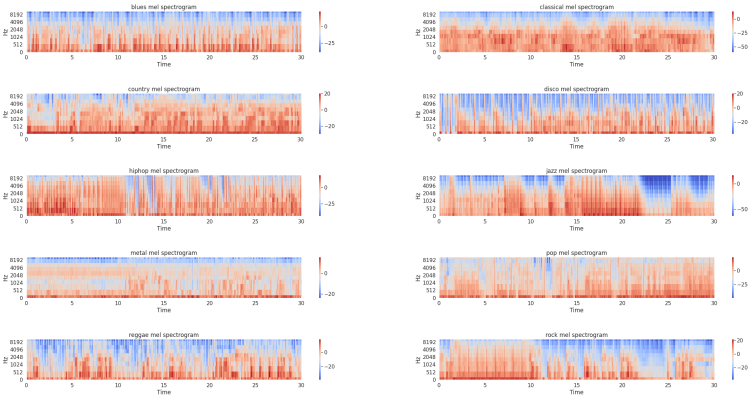


Figure 3. Mel spectrograms for one sample per genre .

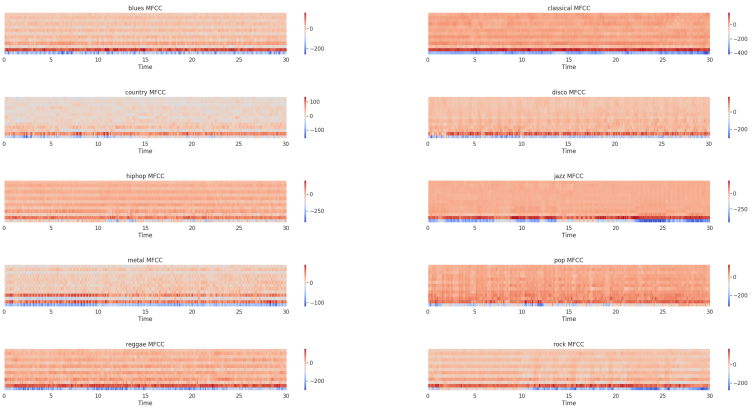


Figure 4. MFCC for one sample per genre .

Direct cosine transform is applied to Mel Frequency bands to obtain MFCC features which perform extremely well in music classification tasks. Below is a flow chart which describes the journey of an audio signal to MFCC features.

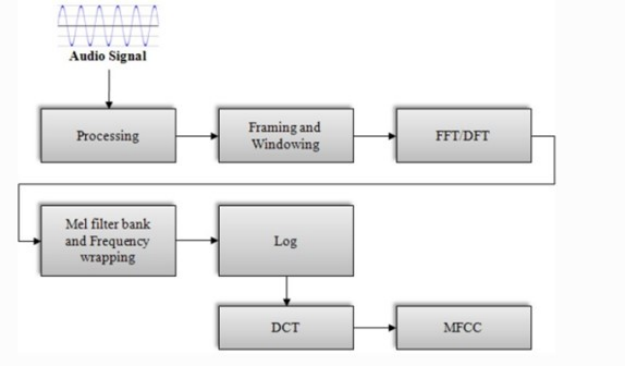


Figure 5. Process from audio signal to MFCC .

4.4 Audio Clip Segmentation

The music signal is extremely high dimensional in time domain. This makes training deep learning models as well as shallow learning models very tedious and time consuming. The workaround of this problem was to divide each and every audio clip into small segments. What we did was dividing each and every audio signal into 10 segments, each segment being of 3 seconds. Intuitively it made sense since every segment of the song should belong to same genre. Moreover, this helped us to achieve two main motives:

1. Reduced the dimension of audio sample in time domain by ten folds
2. Increased the training samples by ten folds since all the segments were used for predictions independently.

4.5 Choice of Sample Rate, Window Size, Overlap, NFFT and hop size

In audio processing, Sample Rate defines the number of discrete data instances used per second to represent an analog sound in digital form. After thorough literature analysis, we set our sample rate to 22050. In order to perform fourier transforms in a localized time domain, we have to define the window size and number of samples over which FFT is computed. The number of samples was set to 2048. The optimal window size is the one which is set in such a way that signal doesn't vary much inside the defined window. The window size after reviewing literature was set to 0.1. Overlap and hop size define how much does the successive time frames on which FFT is performed overlap with one another. The higher the overlap, better the information about consecutive localized time domains but the tradeoff is higher computation cost. The overlap was set to 50 % and hop size to 512.

4.6 Fetching STFT, MFCC and Mel's spectrogram from Audio signals

Feature extraction forms the foundation of machine learning problem. The first challenge was to fetch the right features from the data. After extensive research, we decided to go with STFT, MFCC and Mel's spectrograms to be used in model building. The intuition was that STFT and MFCC features can prove useful for shallow learning and some deep learning models and spectrograms can be experimented with CNN and its variants. We used Librosa python package to extract STFT, MFCC and Mel's spectrograms from the segmented audio clips. The features were saved in different numpy arrays and the reshaped depending on which model we are considering, either classical models or deep learning models.

4.7 Training, validation and test split and label transformation

After feature extraction, the next important step was to transform the labels using one hot encoding and reconstruct them to utilize them in the context of multi-class classification problems. As far as the division between training, validation and testing goes, this was the logical breakdown we came up with

Figure 6. Data split: train, validation and test.

5 Model building

We decided to approach this problem through classical models and deep learning models both. The intent was to see how popular shallow learning models fare in comparison with various deep learning models.

5.1 Shallow learning approach

The input data had following characteristics

- Input features – MFCC feature
- Audio Segments per clip – 5
- Number of samples in one FFT – 2048
- Sampling rate – 22050
- Hop length - 512

MFCC features were chosen because they looked most promising and are the closest approximation to human auditory perception. For shallow learning, we chose to go with these three classifiers,

5.1.1 Support Vector Classifier (SVM)

The support vector classifiers are discriminative classifiers which aim to create the best decision boundary that can segregate the n-dimensional space into different classes depending upon the input set of attributes. The decision boundary is referred to as hyperplane. SVM uses extreme points also called as support vectors to construct the hyperplane.

SVMs are traditionally designed for binary classifications but multi-class classification can be obtained by using one versus rest classification approach which splits the dataset into multiple binary classification problems. Binary classifications are made using SVM and the predictions are made using the model which is most confident. In this approach we train C binary classifiers, where the data from class c is treated as positive and the data from other classes is treated as negative.

When the data is linear then the hyperplane is also a linear one, but for non-linear data we can apply the kernel trick to find the optimal decision boundary. Kernel basically transforms the non-linear data into a high dimensional space to find a hyperplane and then maps the data back to original dimension. Since the audio data is a very high dimensional one, it wouldn't have been wise to go with a linear boundary. Rather, what we did was that we used a popular kernel RBF (Radial Basis Function). If a and b are two feature vectors in some input space then RBF maps these features into a new space by following transformation

$$K(a, b) = exp(\frac{||a - b||^2}{2\sigma^2})$$

Where $||a - b||^2$ is the squared Euclidean distance between two feature vectors and is a free parameter.

Results The results for the SVM model are displayed in Figure 7

CLASSIFICATION REPORT MODEL: svm

	precision	recall	f1-score	support
blues	0.71	0.46	0.56	103
classifcal	0.56	0.66	0.61	80
country	0.64	0.88	0.74	101
disco	0.67	0.87	0.76	110
hiphop	0.53	0.38	0.44	109
jazz	0.41	0.35	0.38	104
metal	0.63	0.53	0.58	107
pop	0.86	0.93	0.90	88
reggae	0.59	0.65	0.62	101
rock	0.49	0.48	0.49	97
accuracy			0.61	1000
macro avg	0.61	0.62	0.61	1000
weighted avg	0.61	0.61	0.60	1000

Figure 7. SVM results report.

5.1.2 Random Forest Classifier

Random forest is a non-parametric ensemble learning method for classification and it operates by constructing a number of decision trees while training. It outputs the class which is the mode of the classes of all decision trees. Simple decision trees are weak learners as they tend to overfit i.e. low bias and high variance. Random forest overcome this challenge by creating a large number of trees on subset of input data and making models more generalizable.

Since they are one of the most popular classifiers for a variety of classification tasks, they were our second choice for music genre classification. Default hyperparameters were used to train these models.

Results The results for the Random Forest Classifier model are displayed in Figure 8

CLASSIFICATION REPORT MODEL: random_forest

	precision	recall	f1-score	support
blues	0.98	0.86	0.92	103
classifcal	0.95	0.96	0.96	80
country	0.87	0.97	0.92	101
disco	0.82	0.98	0.90	110
hiphop	0.95	0.83	0.89	109
jazz	0.92	0.80	0.86	104
metal	0.95	0.91	0.93	107
pop	0.94	0.95	0.95	88
reggae	0.88	0.89	0.89	101
rock	0.86	0.93	0.89	97
accuracy			0.91	1000
macro avg	0.91	0.91	0.91	1000
weighted avg	0.91	0.91	0.91	1000

Accuracy on test is: 90.7 %

Figure 8. Random Forest results report.

5.1.3 Extreme Gradient Boosting (XGBoost)

With huge popularity amongst all the classification algorithms XGBoost was a natural choice. When we studied the literature, we found this technique to be extremely promising. Boosting is an ensemble learning technique which fits multiple models to the data. It starts by fitting an initial model to the data. The second model is built with the focus to accurately predicting the cases where first model performs poorly. The process is repeated many times and the combination of multiple models is expected to perform better than a standalone model. Each successive model attempts to correct the shortcomings of combined boosted ensemble of previous models. Gradient boosting is a type of boosting technique which relies on the intuition that the best possible next model is the one, which when combined with boosted ensemble of previous models, minimizes the loss function. Gradient boosting, thus identifies the short

comings of previous models by using gradients in the loss function. The training is therefore additive as it depends on the additive output of multiple models. The choice of the loss function depends on the machine learning task. The regularization term can be added to account for model or tree complexity.

Results The results for the Random Forest Classifier model are displayed in Figure 9

CLASSIFICATION REPORT MODEL: xgb

	precision	recall	f1-score	support
blues	0.99	0.91	0.95	103
classifcal	0.95	0.97	0.96	80
country	0.97	0.97	0.97	101
disco	0.96	0.98	0.97	110
hiphop	0.97	0.96	0.97	109
jazz	0.98	0.92	0.95	104
metal	0.97	0.98	0.98	107
pop	0.97	1.00	0.98	88
reggae	0.95	0.99	0.97	101
rock	0.95	0.97	0.96	97
accuracy			0.97	1000
macro avg	0.97	0.97	0.97	1000
weighted avg	0.97	0.97	0.97	1000

Figure 9. XGB results report.

5.2 Deep Learning approach

Deep learning networks have proved extremely useful non-linear classification problems recently. Since the deep learning works extremely well with big and high dimensional datasets, with this motivation, we tried different deep learning networks to classify GTZAN dataset.

The plan was to start with a fully connected neural network as a baseline model and modify the network by incorporating CNN, LSTM and different CNN architectures.

The input data had the same characteristics with a little change in the choice of features for few architectures

- Audio Segments per clip – 5
- Number of samples in one FFT – 2048
- Sampling rate – 22050
- Hop length - 512

The network architecture along with the input will be discussed when we discuss each model separately.

5.2.1 Fully Connected Neural Networks (FCNN)

The Fully Connected Neural network was used with MFCC input features. The architecture of the network can be seen in Figure 10

Model: "sequential"

Layer (type)	Output Shape	Param #
flatten (Flatten)	(None, 1690)	0
dense (Dense)	(None, 512)	865792
dense_1 (Dense)	(None, 256)	131328
dense_2 (Dense)	(None, 10)	2570

Figure 10. FCNN Architecture.

Results The results for the FCNN model are displayed in Figure 11

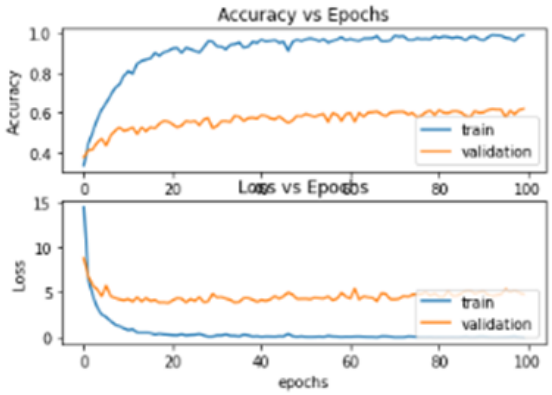


Figure 11. FCNN Results.

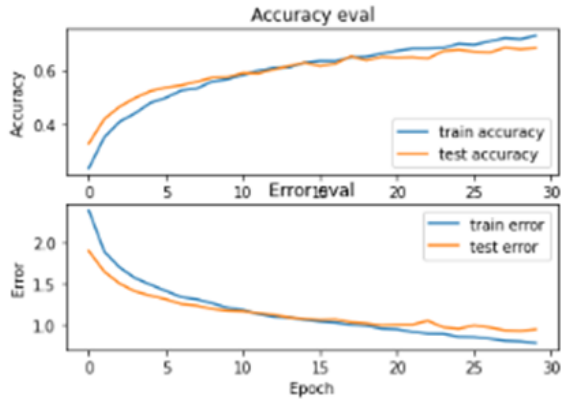
5.2.2 FCNN + CNN

The next step was to try a very simple CNN architecture for feature extraction before inputting the data in FCNN. The type of input used with CNN was still MFCC features. The network had following architecture

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 128, 11, 32)       320
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 64, 6, 32)         0
_________________________________________________________________
batch_normalization (BatchNo (None, 64, 6, 32)         128
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 62, 4, 32)         9248
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 31, 2, 32)         0
_________________________________________________________________
batch_normalization_1 (Batch (None, 31, 2, 32)         128
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 30, 1, 32)         4128
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 15, 1, 32)         0
_________________________________________________________________
batch_normalization_2 (Batch (None, 15, 1, 32)         128
_________________________________________________________________
flatten (Flatten)            (None, 480)               0
_________________________________________________________________
dense (Dense)                (None, 64)                30784
_________________________________________________________________
dropout (Dropout)            (None, 64)                0
_________________________________________________________________
dense_1 (Dense)              (None, 10)                650
=================================================================
Total params: 45,514
```

Figure 12. FCNN + CNN Results.

Results The results for the FCNN combined with CNN model are displayed in Figure 13. The results had a noticeable improvement compared to simple FCNN model and accuracy improved from 63 % to 67 % on test set.




```
79/79 - 1s - loss: 0.9213 - accuracy: 0.6707
```

Figure 13. FCNN + CNN Results.

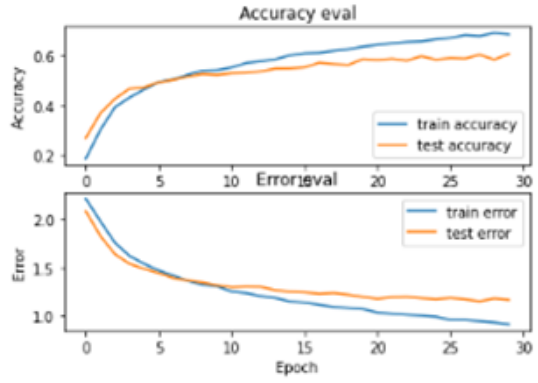
5.2.3 Long Short Term Memory (LSTM)

LSTM is very good with recognizing sequences. As the song is nothing but a sequence of different harmonies and tones, the next logical step was to try with LSTM. LSTM keeps a track of time based sequences in a song. The model was again trained with MFCC features as input. The architecture can be seen in Figure 14.

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm (LSTM)                  (None, 130, 64)           19968
_________________________________________________________________
lstm_1 (LSTM)                (None, 64)                33024
_________________________________________________________________
dense_5 (Dense)              (None, 64)                4160
_________________________________________________________________
dropout_1 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_6 (Dense)              (None, 10)                650
=================================================================
Total params: 57,802
Trainable params: 57,802
Non-trainable params: 0
```

Figure 14. LSTM Architecture.

Results The results for the LSTM are displayed in Figure 15. LSTM alone performed very poor with the classification task




```
79/79 - 2s - loss: 1.1905 - accuracy: 0.5958

Test accuracy: 0.5958383083343506
```

Figure 15. LSTM Results.

5.2.4 C-RNN

Recently it has become increasingly popular to combine CNN with RNN, to process audio signal with the inclusion of sequential information to the model. In convolutional recurrent networks, CNN is used to extract useful features while the RNN part is used to take care of the temporal effect and features of the signal. The input used for C-RNN of our model was a little different compared to other models and we decided to experiment with STFT features. The rest of the input configurations were the same. In Figure 16 we can observe a snapshot of network architecture

Summary..
Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 511, 127, 16)	448
max_pooling2d (MaxPooling2D)	(None, 256, 64, 16)	0
dropout (Dropout)	(None, 256, 64, 16)	0
conv2d_1 (Conv2D)	(None, 254, 62, 32)	4640
max_pooling2d_1 (MaxPooling2	(None, 127, 31, 32)	0
dropout_1 (Dropout)	(None, 127, 31, 32)	0
conv2d_2 (Conv2D)	(None, 127, 31, 64)	18496
max_pooling2d_2 (MaxPooling2	(None, 64, 16, 64)	0
dropout_2 (Dropout)	(None, 64, 16, 64)	0
conv2d_3 (Conv2D)	(None, 64, 16, 128)	73856
max_pooling2d_3 (MaxPooling2	(None, 32, 8, 128)	0
dropout_3 (Dropout)	(None, 32, 8, 128)	0
conv2d_4 (Conv2D)	(None, 32, 8, 64)	73792
max_pooling2d_4 (MaxPooling2	(None, 8, 2, 64)	0
dropout_4 (Dropout)	(None, 8, 2, 64)	0
reshape (Reshape)	(None, 16, 64)	0
lstm (LSTM)	(None, 16, 128)	98816
lstm_1 (LSTM)	(None, 16, 128)	131584
lstm_2 (LSTM)	(None, 64)	49408
dense (Dense)	(None, 32)	2080
dense_1 (Dense)	(None, 10)	330

Total params: 453,450

Figure 16. CRNN Architecture.

Results The results for the CRNN are displayed in Figure 17. This results were the most promising amongst all different neural networks that we have tried up to this point

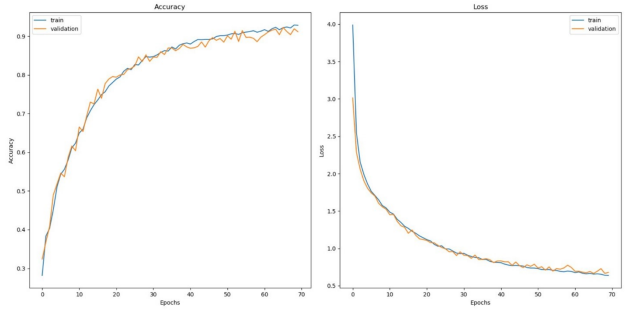


Figure 17. CRNN Results.

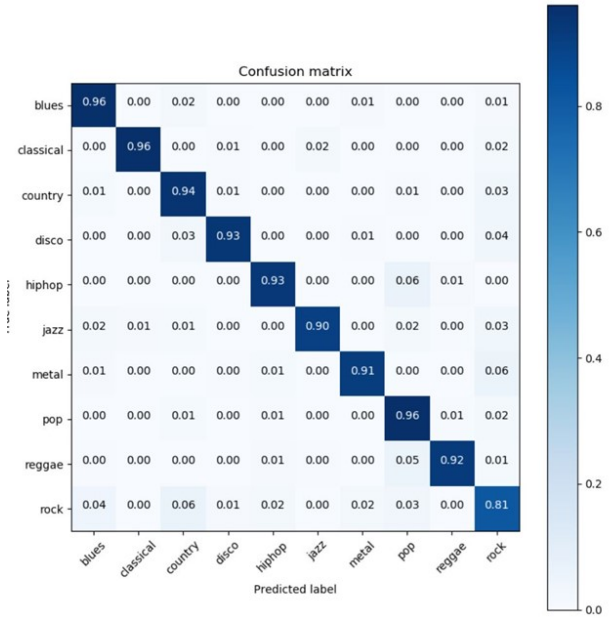


Figure 18. CRNN Confussion Matrix.

5.2.5 CNN VGG 16 with FCNN

The CNN VGG 16 architecture was implemented along with a simple FCNN model for this step. CNN VGG 16 is an extremely popular architecture and works extremely well with images. The input used for this step was mel's spectrogram. The output of VGG16 was fed to a one layer neural network having 32 hidden units with Relu activation. The output layer of FCNN had softmax activation.

Results The results for the CNN VGG 16 wih FCNN are displayed in Figure 19.

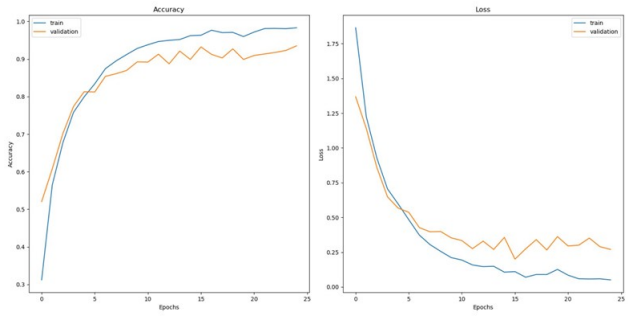


Figure 19. CNN VGG 16 with FCNN Results.

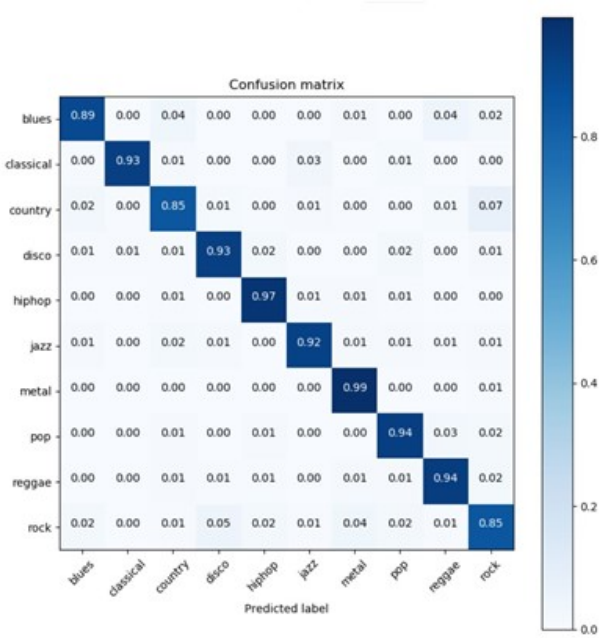


Figure 20. CNN VGG 16 with FCNN Confussion matrix.

6 Final Results

Overall, XGBoost classifier stood out amongst the rest with 96.6% test set accuracy. C-RNN performed the best amongst different deep learning models with 92.2 % accuracy.

Results		
Model	Input Features	Accuracy
SVM	MFCC	61.4 %
Random Forest	MFCC	91%
XGBoost	MFCC	96.6 %
FCNN	MFCC	62 %
FCNN + CNN	MFCC	67 %
RNN LSTM	MFCC	59 %
C-RNN	STFT	92.2 %
CNN (VGG16) + FCNN	Mel Spectrogram	91.9 %

7 What is next?

Following are the next steps we have planned,

1. CNN VGG16 with LSTM
2. CNN VGG16 with bidirectional LSTM
3. CNN VGG16 with XGBM.

We'll be using Mel's spectrogram as inputs for all these configurations. We will also study these algorithms in depth and explore their applications in other domains but will not be practically implementing them for other applications.

References

[1] George Tzanetakis and Perry Cook. Musical genre classification of audio signals. *IEEE Transactions on Speech and Audio Processing*, 10:293 – 302, 08 2002. doi:10.1109/TSA.2002.800560.

[2] Bob L. Sturm. The GTZAN dataset: Its contents, its faults, their effects on evaluation, and its future use. *CoRR*, abs/1306.1461, 2013. URL `http://arxiv.org/abs/1306.1461`.

[3] Michaël Defferrard, Kirell Benzi, Pierre Vandergheynst, and Xavier Bresson. Fma: A dataset for music analysis, 2017.

[4] Thierry Bertin-Mahieux, Daniel Ellis, Brian Whitman, and Paul Lamere. The million song dataset. pages 591–596, 01 2011.

[5] J. R. Castillo and M. J. Flores. Web-based music genre classification for timeline song visualization and analysis. *IEEE Access*, 9:18801–18816, 2021. doi:10.1109/ACCESS.2021.3053864.

[6] Chi Zhang, Yue Zhang, and Chen Chen. Songnet: Real-time music classification. *Standford*, 2018.