Music Genre Recognition

Abantika Basak
1217117655
Arizona State University
Tempe, AZ 85281
abasak4@asu.edu

Abhilasha Mandal
1217160477
Arizona State University
Tempe, AZ 85281
abhilasha.mandal@asu.edu

Ajinkya Dande
1217613956
Arizona State University
Tempe, AZ 85281
adande@asu.edu

Omkar Muglikar
1217158150
Arizona State University
Tempe, AZ 85281
omuglika@asu.edu

Rounak Sengupta
1215043206
Arizona State University
Tempe, AZ 85281
rsengup7@asu.edu

1 Introduction

Internet services expose vast amounts of multimedia data for exchange and browsing, which induces the need for effective and unambiguous ways of data description based on small fractions of information. Digital music databases cannot be easily searched through. The diversity of musical trends and genres, uncommon instruments as well as the variety of performers and their compositions compel the existence of numerous music recognition systems. However, the key issue in the automatic query is parametrization. Parametrization has so far experienced extensive development, however, there are still some important areas of Music Information Retrieval, such as for example music genre classification, that is researching this aspect. In this project, we devised a mechanism for recognizing the music genre of given fragments of music tracks.

Companies these days use music classification, either for recommending music to their customers based on their preference (such as Spotify, Soundcloud) or simply as a standalone product (such as Shazam). Determining music genres is the initial and fundamental step in that direction. Machine Learning methods have proved to be successful in extracting trends and patterns from the large pool of data. The same systems can be implemented for doing Music Analysis too.[1] Automatic music categorization is one of the most important tasks for Music Information Retrieval and it helps us to draw meaningful insights from a given audio clip. Machine Learning researchers have implemented different methods to implement automatic genre classification. In this project, we attempt to solve this problem by classifying the genre of a given set of audio files into 10 classes. A musical genre is described by the common characteristics typically related to the instruments used, rhythmic structure, and harmonic content of the music [2]. Every music track is assigned with a label called a Genre which identifies the type of the song based on several similar features of the audio tracks like rhythm, tempo, and lyrics. A few examples of genres are Country, Jazz, Classical, Rock, etc.

Music Genre Recognition has emerged as an important and challenging problem in the area of Sound Engineering. Different music streaming applications can recommend songs to users based on the genre of their preference. Hence, this task has a practical application of building Music Recommendation Systems. The applications can be extended to Automatic Music Transcription systems as well.

2 Problem Description

Given the audio files for each genre, we predict the genre class by use of five different machine learning approaches.i.e Input to the algorithm will be an audio file and output will be a genre it is associated with. The underlying task of music genre recognition is classifying the data into 10 classes (here the music genres). We extract features from the raw audio files. Then, once the feature vectors are obtained, we train different classifiers on the training set of feature vectors. Following are the different classifiers that were used -

- Multinomial Logistic Regression (LR)
- Support Vector Machine (SVM)
- Convolution Neural Networks (CNN)
- Long-Short Term Memory with CNN (CNN-LSTM)
- Bottom-up Broadcast Neural Network (BBNN)

3 Related Work

Evidence of music genre classification has existed ever since the birth of the internet. Initial attempts at solving this problem involved supervised machine learning approaches such as Gaussian Mixture model and k-nearest neighbour classifiers[2]. The features used for this were the timbral structure, rhythmic content and pitch content. Mel-frequency Cepstral Coefficients (MFCCs) and spectral roll-off were also some of the features explored here. Hidden Markov Models (HMMs), have also been explored for music genre classification [3][4]. Support vector machines (SVMs) have also been used for classifying genres [5].

There is evidence that the psycho-acoustic features can be used for recognizing music genre[6], especially the importance of STFT taken on the Bark Scale[7]. Visual and acoustic features were also used to train SVM and AdaBoost classifiers[8].

With the recent success of deep neural networks, a number of studies apply these techniques to speech and other forms of audio data [9][10]. Representing audio in the time domain for input to neural networks is not very straight-forward because of the high sampling rate of audio signals[11].

Spectrograms can be considered as images and used to train convolutional neural networks (CNNs) [12]. A CNN was developed to predict the music genre using the raw MFCC matrix as input[13]. A constant Q-transform (CQT) spectrogram was provided as input to the CNN to achieve the same task[14].

4 Dataset

We will be using the GTZAN music genre dataset[2] for this project which comprises of 1000 audio tracks, each 30 seconds long. We have chosen this dataset because of the fact that it is one of the standard datasets used for the music genre classification task and hence the results obtained on this dataset can be used effectively for comparative analysis. Also, it consists of raw audio data and hence gives us the freedom of choosing meaningful features for predicting the classes. In this dataset, 10 genre class labels are present and 100 audio files per genre, which provides a relatively large enough dataset for the music genre classification problem and all the tracks are 22050 Hz Mono 16-bit files in .wav format. Besides, a lot of research has been conducted on this dataset, which provides some insights for finding the solution of this problem.

5 Feature Extraction

The feature extraction is a very significant element in examining and detecting relationships between different entities. The provided data of audio files cannot be explained by the models directly. Hence, we need to convert them into an intelligible format, for this feature extraction is used.

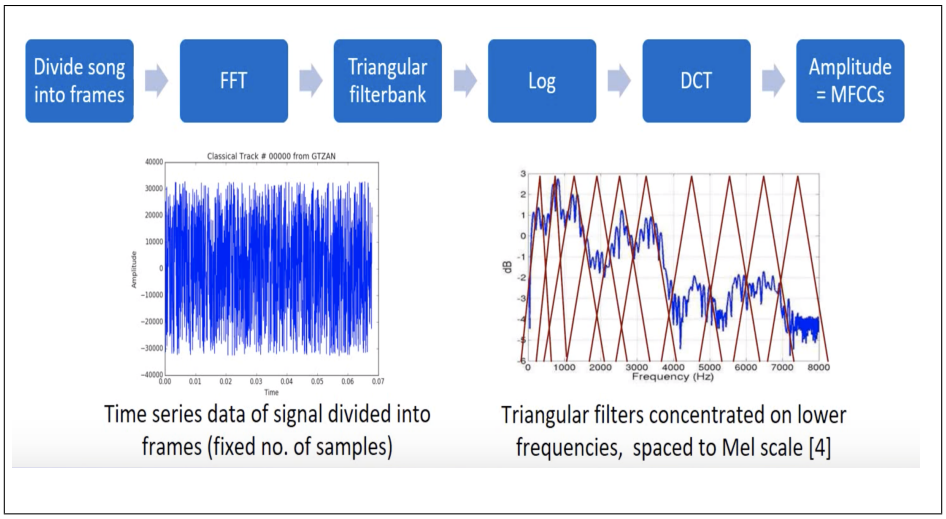


Figure 1: Mel Frequency Cepstrum Coefficients of a song.

5.1 Musical Surface Features

The term "musical surface" is used to denote the characteristics of music related to texture, timbre and instrumentation[2]. To classify our audio clips, we chose the following features: Chroma frequencies, Spectral Centroid, Roll-off, Zero Crossing Rate and the Mel-Frequency Cepstral Coefficients (MFCC). Then, we used different multi-class classifiers and an ensemble of these to obtain our results.

5.1.1 Mel-Frequency Cepstral Coefficients (MFCC)

The Mel-Frequency Cepstral Coefficients (MFCC) feature extraction method is a leading approach for speech feature extraction and current research aims to identify performance enhancements.[15] MFCC's are widely used feature extraction technique for speech and sound recognition as timbrel features. We have decided to use this feature extraction technique as this takes into consideration human perception of sound frequencies. We know that some frequencies are more important to human ear over others, this technique distinguish those variations in the data.

MFCC represents a set of short term power spectrum characteristics of the sound and have been used in the state-of-the-art recognition and sound categorisation techniques. The MFCC's of a signal are a small set of features (usually about 10–20) which concisely describe the overall shape of a spectral envelope as shown in figure 1.

After MFCC Exteraction, each song can be visualized as an array of size $N_{Frames} X N_{MFCC}$ From the visualization, we observe that higher MFCCs do not contain relevant spectral information. For the GTZAN dataset, each frame is configured as 256 samples.

$$N_{Frames} = \frac{SampleRate \times SongLength}{FrameSize} = \frac{22050Hz \times 30sec}{256} = 2854Frames$$

We further take mean vector and covariance of this MFCC matrix to get our final matrix which we will be using for analysis.

5.1.2 Chroma frequencies

Chroma frequency vector discretizes the spectrum into chromatic keys, and represents the presence of each key. We take the histogram of present notes on a 12-note scale as a 12 length feature vector. The chroma frequency have a music theory interpretation. The histogram over the 12-note scale actually is sufficient to describe the chord played in that window. It provides a robust way to describe a similarity measure between music pieces.

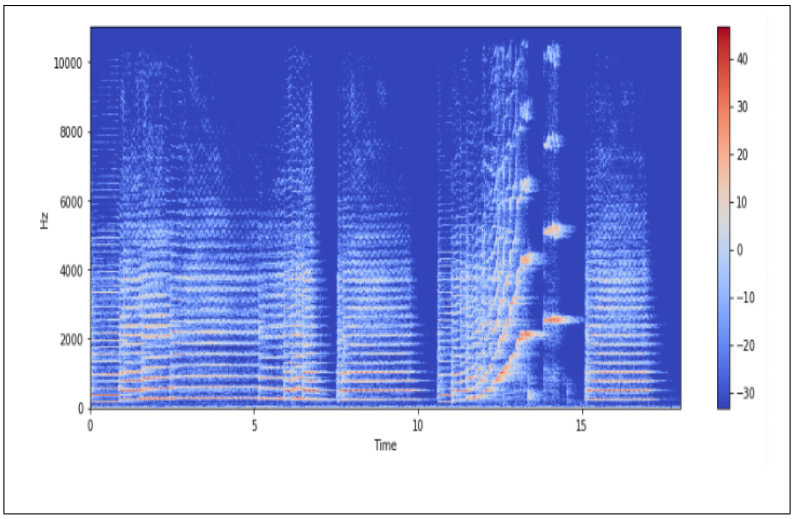


Figure 2: Chroma frequencies of a song.

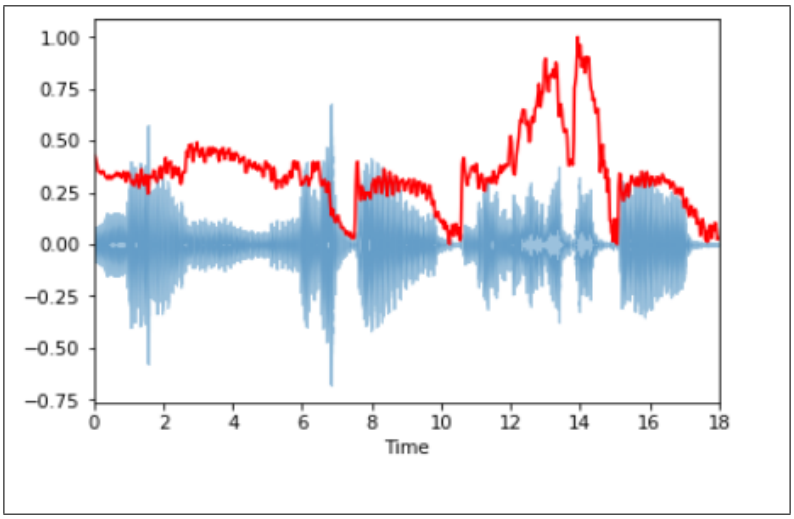


Figure 3: Spectral centroid of a song.

5.1.3 Spectral centroid

Spectral Centroid It describes where the "centre of mass" for sound is. It essentially is the weighted mean of the frequencies present in the sound. Consider two songs, one from blues and one from metal. A blues song is generally consistent throughout it length while a metal song usually has more frequencies accumulated towards the end part. So spectral centroid for blues song will lie somewhere near the middle of its spectrum while that for a metal song would usually be towards its end.

$$C = \frac{\sum_1^N fM[f]}{\sum_1^N M[f]}$$

5.1.4 Roll-off

It is a measure of the shape of the signal. It represents the frequency below which a specified percentage of the total spectral energy.

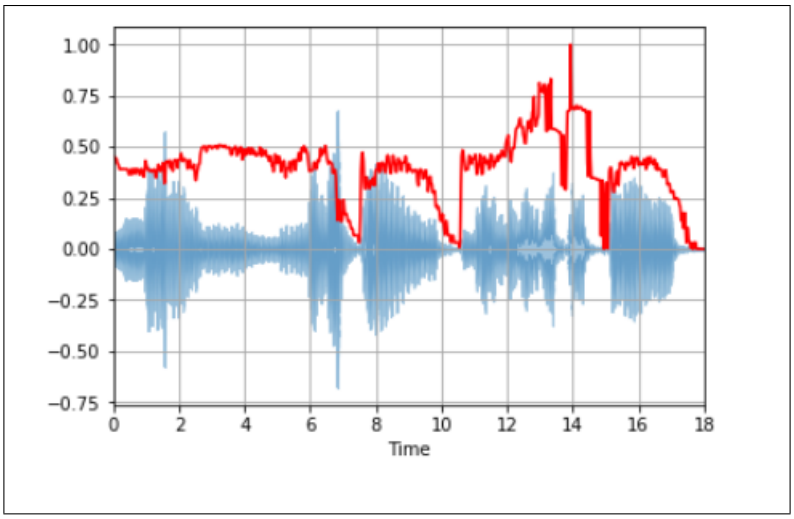


Figure 4: Spectral roll-off of a song.

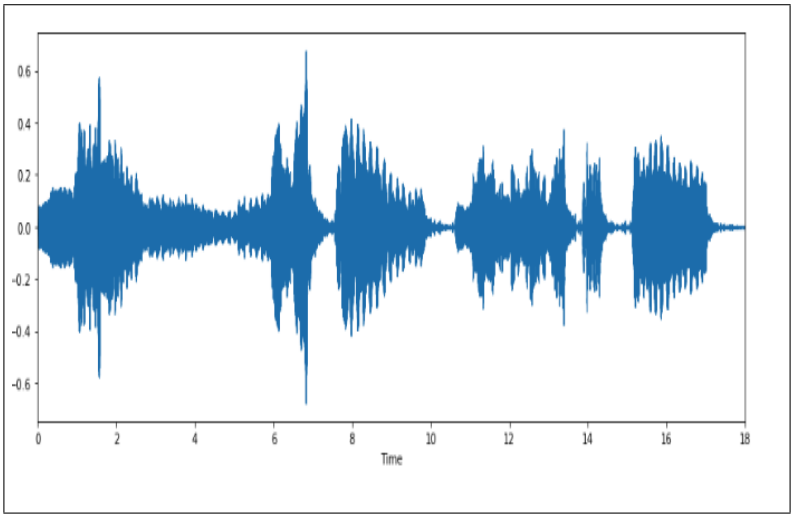


Figure 5: Zero crossing rate of a song.

$$\sum_{1}^{R} M[f] = 0.85 \sum_{1}^{N} M[f]$$

5.1.5 Zero crossing rate

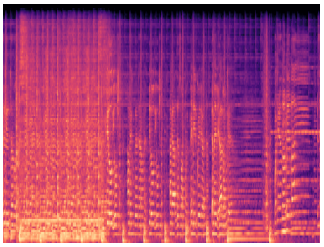
The zero crossing rate is the rate of sign-changes along a signal, i.e., the rate at which the signal changes from positive to negative or back. This feature has been used heavily in both speech recognition and music information retrieval. It usually has higher values for highly percussive sounds like those in metal and rock. Zero Crossings are useful to detect the amount of noise in a signal.

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

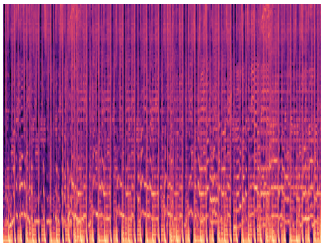
6 Methodology

6.1 Convolution Neural Networks

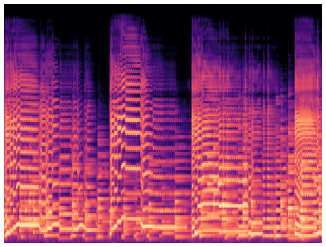
Convolutional Neural Networks is a deep learning algorithm which particularly makes use of convolutional layers to extract information from patterns present in images for classification purposes. CNNs can also be used for image segmentation and signal processing. In the context of Music Genre Recognition, mel spectrograms can be generated for every audio file present in the given dataset. A spectrogram represents the spectrum of frequencies with time. A regular spectrogram, which shows the squared magnitude of the Short term Fourier Transform over time, is converted to a mel-spectrogram by using a mel-scale designed to imitate the way humans hear and perceive sounds [16]. Hence, every mel-spectrogram can be considered as an image which can be fed to a CNN for classification into genres. The mel-spectrogram for audio files belonging to 3 genres namely rock, pop, jazz are shown below :-



(a) Rock Song



(b) Pop Song



(c) Jazz Song

6.1.1 Data preprocessing

The original dataset GTZAN consists of 1000 audio files for every genre. As already mentioned in the Dataset section, there are 10 genres and 100 audio files belonging to each genre. The mel-spectrogram images were generated for every audio file using the Librosa library in Python. We split the images into 900 training and 100 testing samples. The data was shuffled randomly so that the network was exposed to every type of genre with equal probability.

6.1.2 Training Phase

This Neural Network was implemented using Keras, a high level neural networks API that runs on top of Tensorflow. The input to the CNN is an RGB mel-spectrogram image with dimensions 480x640x3. The architecture of the CNN involves the following chain of operations : Conv 2D - RELU - Conv 2D - RELU - Average Pooling 2D which has been performed thrice. The activation function, the Rectified Linear Unit, abbreviated as RELU, is used here as it helps to alleviate the vanishing gradient problem where the gradient exponentially decreases as we go on to the lower layers of the CNN. The output is then passed into 2 dense layers and one final dense layer with Softmax Activation consisting of 10 units to assign a probability to each of the 10 classes. The optimizer used here is RMSprop as it gave the best results among the other optimizers including Adam and Stochastic Gradient Descent. The training took almost 45 minutes for running 100 epochs using GPU support, from Google Colaboratory.

6.1.3 Testing Phase

The test data, containing 100 samples, were evaluated using scikit-learn library in Python, which provided us with the test accuracy of 67% as well as the confusion matrix as shown in Figure 7b.

6.2 Convolutional Recurrent Neural Networks

Another deep learning approach that was undertaken is the Convolutional Recurrent Neural Network. This makes use of both Convolutional Neural Network as well as Recurrent Neural Network.

Since mel-spectrograms represents the spectrum of frequencies over time, hence RNNs might do a good job at capturing the time dependencies by making hidden state at time t dependent on the hidden

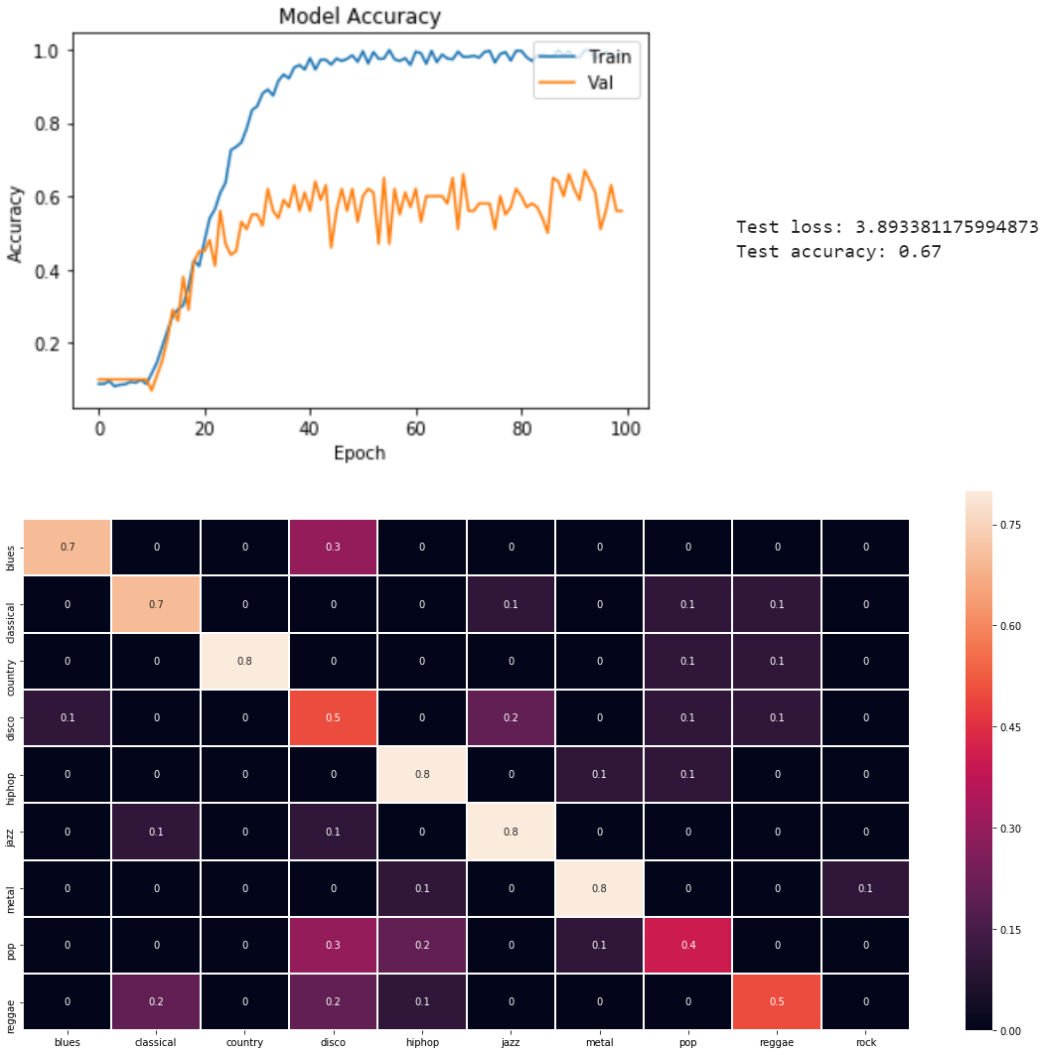


Figure 7: CNN a) Accuracy Vs Epoch Graph b) Confusion Matrix

state at time t-1 [16]. We use the CNN used in Section 6.1, as each spectrogram is still an image, which is later followed by an LSTM, also known as Long Short Term Memory Network. LSTM is a type of Recurrent Neural Network which helps in capturing long term dependencies. The architecture used here in the CNN-LSTM Model is Conv 2D - RELU - Conv 2D - RELU - Conv 2D - RELU - Conv 2D - RELU - Conv 2D - RELU - Conv 2D - RELU - LSTM1 - LSTM2 followed by 3 dense layers where the last dense layer comprises of 10 output units corresponding to the 10 genres.

The same data preprocessing technique as in CNN has been used for CNN LSTM. After following a similar training procedure as in CNN using the same optimizer and activation functions except for the number of epochs i.e, 200 here, we have got a validation accuracy of 48%. The reason behind this could be due to the small size of the dataset. Here, there are only 900 training examples taken into account. However, for complex deep learning models, lots of training examples are generally required. The second reason could also be the inability to tune the hyperparameters properly. We plan to improve this accuracy as a future scope of this project.

6.3 Bottom-up Broadcast Neural Networks

This is a state of the art architecture of CNN specifically designed by [17] for classifying genres of music. This architecture uses ideas from inception blocks as well as skip connections to improve accuracy of classification task. According to [17] the intuition behind this architecture is that - *"sound*

events are accumulated by frequency over time domain which causes the individual genre has different performance sensitivity to different time scales and levels of features. Therefore, it is necessary to design a specific CNN structure which can comprehensively handle multi-scale of audio features." The network addresses this problem by the use of inception blocks. Inception block consists of different convolution layers which output filters of different sizes and then concatenates them into a single output which is fed to the next block.

Also according to [17] *"how to construct an appropriate CNN structure to maximally abstract high-level information and preserve the lower-level features simultaneously, which is just for the task of music classification is of vital importance but challenging. "* This has been tackled by use of skip connections in the network. These connections help to preserve the lower level features. The architecture of network is given in Figure 9

6.3.1 Training phase

We followed the same data preprocessing technique as in CNN. The input to the network was an image of size $(288 \times 432 \times 3)$.The implemented network is shown in 9. It was trained for 100 epochs with a batch size of 3.

We used the categorical cross entropy as a loss function. Also called Softmax Loss. It is a Softmax activation plus a Cross-Entropy loss. If we use this loss, we will train a CNN to output a probability over the C classes for each image. It is used for multi-class classification. Also, among optimizers like root mean square propogation,stochastic gradient descent and Adam, we choose Adam (Adaptive moment estimation) optimizer for training the neural network as it gave better results.

$$CE = -log\left(\frac{e^{s_p}}{\sum_j^C e^{s_j}}\right) \tag{1}$$

The training took 1.3 hours to complete as it had a total of 186,218 trainable parameters. We could not reproduce the results given in the paper exactly but we got an accuracy score close to it. One of the reasons may be the hyperparameter tuning.

6.3.2 Testing and evaluation

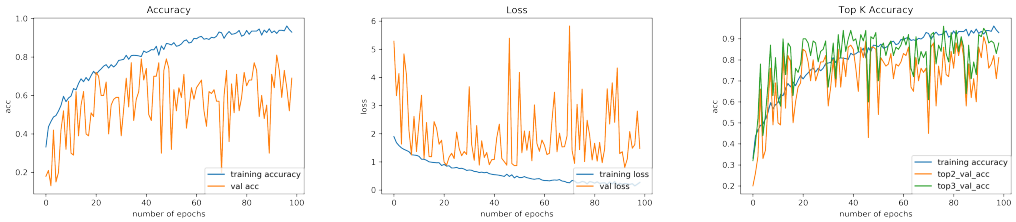
We test the network on the testing data generated before. Since the network outputs class probabilities, it makes sense to loook at top k probabilities of the network. These probabilities give us some confidence with which the input image is correctly classified.

Top k accuracy is the accuracy that the target class is present in the top k predictions of network. Thus we use top2 and top3 accuracies as well for analysis. We calculate its class wise accuracy (confusion matrix), top 2 accuracy, top 3 accuracy and normal accuracy as shown in figure 8

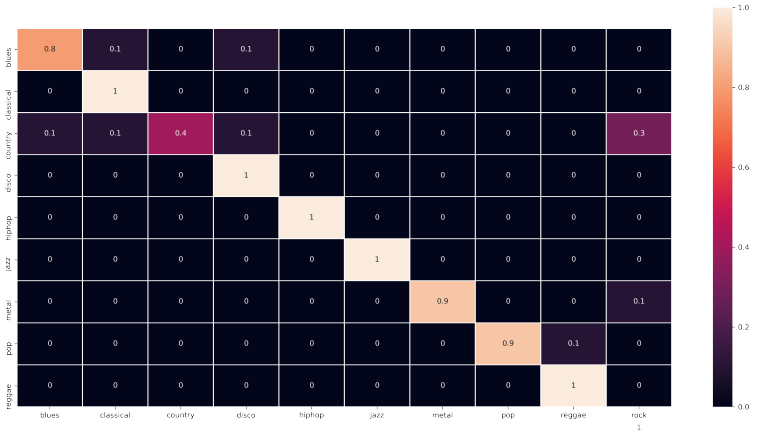
6.3.3 Analysis

We can see from above graphs that the validation loss function does not converge even after training for 100 epochs and it keeps on oscillating leading to high variance in validation error/loss. Hyper parameter tuning could have improved the results greatly.

Also we can see that top 2 and top3 accuracy values are high. This implies that even if the genre was not correctly classified into one class, the correct genre was among the top 2 or top 3 probabilities predicted by the network. This tells us that model was close to the original answer.



(a) Accuracy (b) Loss (c) Top k Accuracy



(d) Confusion Matrix for BBNN

Figure 8: Plots of BBNN

6.4 Support Vector Machines

A Support Vector Machine (SVM) is a discriminative classifier formally defined by a separating hyperplane. In other words, given labeled training data (supervised learning), the algorithm outputs an optimal hyperplane which categorizes new examples. In two dimensional space, this hyperplane is a line dividing a plane into two parts wherein each class lay on either side.

6.4.1 Data pre-processing

The original dataset consisted of 1000 audio files for every genre. Firstly, this audio data was converted into MFCC features using the librosa python library. The split the images into 900 training and 100 testing samples. The data was shuffled randomly so that the network was exposed to every type of genre with equal probability.

6.4.2 Training phase

The training phase is carried out in following steps: Firstly, we will standardize the obtained MFCC feature matrix. Later, we apply Principal Component Analysis with 100 components on the obtained matrix to reduce the dimentionality. We use the reduced matrix to train the SVM classifier from sklearn library in python. We obtain overall 0.99 accuracy on training data. Refer Figure 10

6.4.3 Testing and evaluation

The trained model is on the testing data generated before. Since the model outputs class labels, we checked the obtained labels against the actual class labels to get the testing accuracy of 0.60. Refer Figure 11

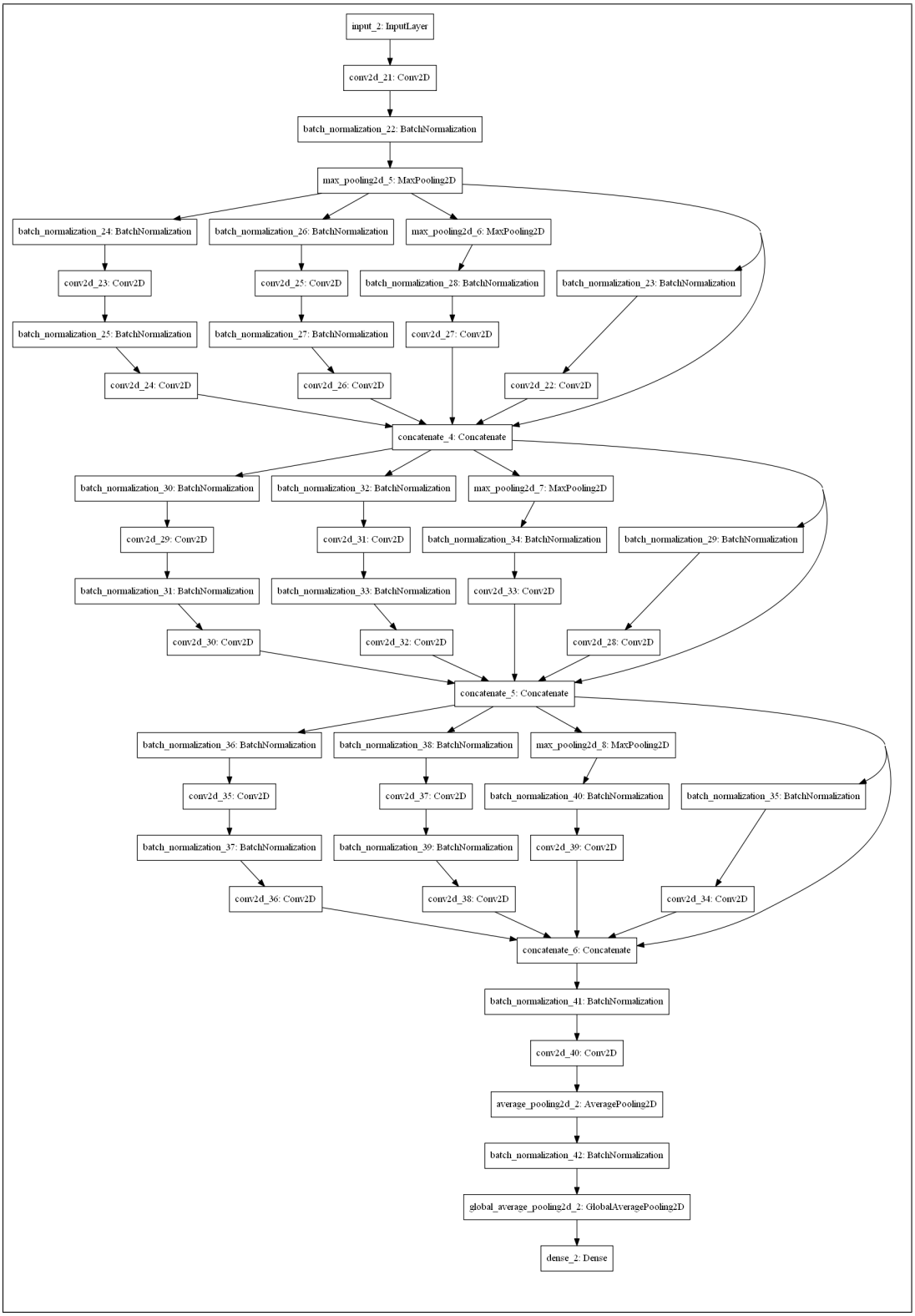


Figure 9: BBNN model

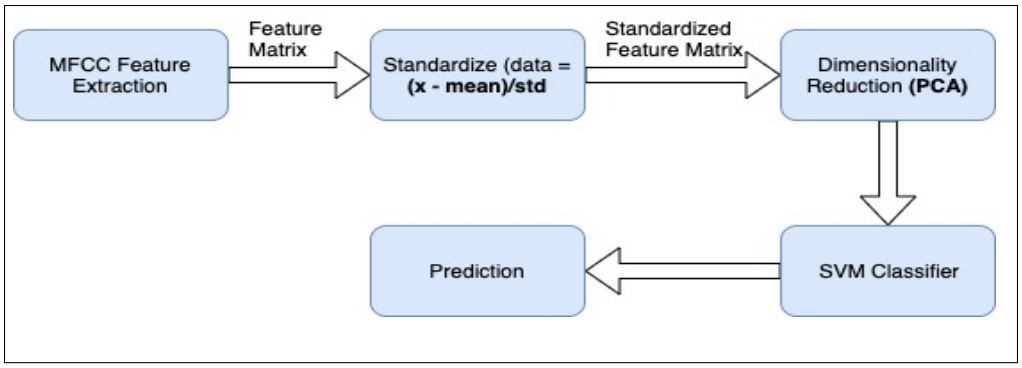


Figure 10: SVM Flow

```
Loading data
Shape of training set after vectorization:
(900, 121800)
Shape of test set after vectorization:
(100, 121800)
Running PCA
Shape of training set after pca:
(900, 100)
Shape of test set after pca:
(100, 100)
Training SVM
Train Accuracy: 99.9%
Test Accuracy: 60.0%
```

Figure 11: SVM accuracy output.

6.5 Multinomial Logistic Regression

Multinomial logistic regression is a classification method that generalizes logistic regression to multiclass problems, i.e. with more than two possible discrete outcomes.[18]

In the logistic regression, the black function which takes the input features and calculates the probabilities of the possible two outcomes is the Sigmoid Function. Later the high probabilities target class is the final predicted class from the logistic regression classifier. When it comes to the multinomial logistic regression the function is the Softmax Function. The equation is given by

$$P(y_i = k|X) = \frac{e^{\beta_k x_i}}{\sum_{j=1}^{K} e^{\beta_j x_i}}$$

where, $P(y_i = K|X)$ is the probability the *i*th observation's target value, y_i, is class *k*, and K is the total number of classes.

6.5.1 Data pre-processing

Again, this audio data was converted into mel-spectrogram images using the librosa python library. And features such as chroma frequencies, spectral bandwidth, rolloff, zero crossing rate and the MFCC coefficients were extracted from it. Finally, all of these values were stored into a CSV as a table with their labels.

6.5.2 Training phase

Training the multinomial logistic regression model requires the features and the corresponding targets. For this, we are going to split the dataset into four datasets. Which are

$$train_x, test_x, train_y, test_y$$

We are going to use the train_x and train_y for modeling the multinomial logistic regression model and use the test_x and test_y for calculating the accuracy of our trained multinomial logistic regression model. We are using the scikit-learn train_test_split method to split the dataset. The music dataset randomly into 80% for training and remaining 20% for testing using the train_test_split method.

In the second approach, we are going pass the multinomial parameter into the scikit-learn logistic regression function before we fit the model with train_x, test_x.

6.5.3 Testing and evaluation

To calculate the accuracy of the trained multinomial logistic regression models we are using the scikit learn metrics method. We are calling the metrics method accuracy_score function with actual targets and the predicted targets.

6.5.4 Analysis

From the result, the binary scikit-learn logistic regression has a lower accuracy than the multinomial logistic regression model.

```
Number of observations :: 1000
Number of columns :: 28
Headers :: ['id' 'chroma_stft' 'rmse' 'spectral_centroid' 'spectral_bandwidth'
 'rolloff' 'zero_crossing_rate' 'mfcc1' 'mfcc2' 'mfcc3' 'mfcc4' 'mfcc5'
 'mfcc6' 'mfcc7' 'mfcc8' 'mfcc9' 'mfcc10' 'mfcc11' 'mfcc12' 'mfcc13'
 'mfcc14' 'mfcc15' 'mfcc16' 'mfcc17' 'mfcc18' 'mfcc19' 'mfcc20' 'label']
Logistic regression Train Accuracy ::  0.8785714285714286
Logistic regression Test Accuracy ::  0.8
Multinomial Logistic regression Train Accuracy ::  1.0
Multinomial Logistic regression Test Accuracy ::  0.9566666666666667

Process finished with exit code 0
```

Figure 12: MLR accuracy output.

7 Results

We finally compare the accuracies obtained by employing above mentioned classifiers. Table 1 has the results of all the different classifiers equipped to tackle the problem. These values are comparable to the current state-of-the-art results available online.

Table 1: Accuracy obtained by different classifiers

Classifier	Accuracy
Multinomial Logistic Regression (MLR)	0.96
Bottom-up Broadcast Neural Network (BBNN)	0.88
Convolution Neural Networks (CNN)	0.76
Support Vector Machine (SVM)	0.60
Long-Short Term Memory with CNN (CNN-LSTM)	0.48

8 Conclusion

In this work, the task of music genre recognition is observed using the GTZAN dataset. We have used 5 different classifiers to complete this task. The Multinomial Logistic Regression gave us the best accuracy of 0.96 with Bottom-up Broadcast Neural Networks giving us an accuracy of 0.88. The understanding is that, complex models don't always give the best accuracy. Also, considering that our data set isn't large (1000 only), it might also be affecting the accuracy of the more complex classifiers.

9 References

[1] P. Pandey, "Using cnns and rnns for music genre recognition,"

[2] G. Tzanetakis and P. Cook, "Musical genre classification of audio signals," *IEEE Transactions on Speech and Audio Processing*, vol. 10, pp. 293–302, July 2002.

[3] N. Scaringella and G. Zoia, "On the modeling of time information for automatic genre recognition systems in audio signals," pp. 666–671, 2005.

[4] H. Soltau, T. Schultz, M. Westphal, and A. Waibel, "Recognition of music types," vol. 2, pp. 1137–1140 vol.2, May 1998.

[5] M. Mandel, M. I. M, and D. Ellis, "Song-level features and support vector machines for music classification," pp. 594–599, 2005.

[6] T. Lidy and A. Rauber, "Evaluation of Feature Extractors and Psycho- Acoustic Transformations for Music Genre Classification.," pp. 34–41, Sept. 2005.

[7] E. Zwicker and H. Fastl, "Psychoacoustics facts and models," 1999.

[8] L. Nanni, Y. M. Costa, A. Lumini, M. Y. Kim, and S. R. Baek, "Combining visual and acoustic features for music genre classification," *Expert Syst. Appl.*, vol. 45, pp. 108–117, Mar. 2016.

[9] O. Abdel-Hamid, A.-R. Mohamed, H. Jiang, L. Deng, G. Penn, and D. Yu, "Convolutional neural networks for speech recognition," *IEEE/ACM Trans. Audio, Speech and Lang. Proc.*, vol. 22, pp. 1533–1545, Oct. 2014.

[10] J. F. Gemmeke, D. P. W. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, M. Plakal, and M. Ritter, "Audio set: An ontology and human-labeled dataset for audio events," 2017.

[11] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. W. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," *CoRR*, vol. abs/1609.03499, 2016.

[12] L. Wyse, "Audio spectrogram representations for processing with convolutional neural networks," *CoRR*, vol. abs/1706.09559, 2017.

[13] T. Li, A. Chan, and A. Chun, "Automatic musical pattern feature extraction using convolutional neural network," pp. 546–550, 2010.

[14] T. Lidy and A. Schindler, "Parallel convolutional neural networks for music genre and mood classification," *MIREX2016*, 08 2016.

[15] M. A. Hossan, S. Memon, and M. A. Gregory, "A novel approach for mfcc feature extraction," *IEEE*, 2010.

[16] P. Dwivedi, "Using cnns and rnns for music genre recognition,"

[17] C. Liu, L. Feng, G. Liu, H. Wang, and S. Liu, "Bottom-up Broadcast Neural Network For Music Genre Classification," *arXiv e-prints*, p. arXiv:1901.08928, Jan 2019.

[18] W. H. Greene, "Econometric analysis (seventh ed.)," pp. 803–806.