Music Genre Classification

Shivani Soman
University of California, Los Angeles
UID : 304946159

I. INTRODUCTION

In this day and age of music streaming applications like Spotify, SoundCloud and iTunes, organization and management of music is vital. Gone are the days of CDs and other physical forms of music. With millions of songs available on the internet to download and stream, classification of music for easy retrieval is needed. One way to classify music is according to its genre, which is a conventional category that identifies some pieces of music as belonging to a shared tradition or set of conventions [3]. It is the process of categorizing music into different groups based on style, tempo, rhythm or melody.

However, due to the artistic nature of music genres, these classifications can be very subjective. Also, new genres and subgenres are always being introduced and can also be overlapping in many situations. Over 1200 different types of micro-genres exist in the musical industry today. These reasons make the task of music genre classification quite challenging in the field of Music Information Retrieval (MIR). Humans in general are more likely to browse and search by genre than by recommendation, artist similarity or music similarity, as found by a study conducted a few years ago [9]. Genre can be important factor for the listener in influencing the liking of a particular piece.

Automatic music genre classification can be beneficial especially for online music streaming websites like Spotify and increase user engagement. In this project we aim to classify music into genres using two different machine learning methods. In the first part of the project, we extract time and frequency domain features from the music samples and pass those features to traditional machine learning classifiers. The second method aims to classify the samples by generating MEL spectrograms of the samples and performing image classification using deep learning techniques like CNNs and CRNNs. Finally, we will be also be testing an ensemble model of the best performing classifiers and compare the results.

The rest of this paper is organized as follows. Section II describes the existing methods in the literature for the task of music genre classification. Section III is an overview of the the dataset used in this study and how it was obtained. Section IV provides details on the feature engineering process. The proposed models and the implementation details are discussed in Section V. The results are reported in Section VI, followed by the future work and conclusion from this study in Section VII and VIII.

II. RELATED WORK

The paper "Musical Genre Classification of Audio Signals" [11], published in 2002, was the first paper that made use of the GTZAN Dataset for music genre classification. They explored the extraction of various features from the audio signals such as spectral centroid, rolloff, flux, as well as mel-frequency cepstral coefficients (MFCC). Overall, three feature sets for representing timbral texture, rhythmic content and pitch content of music signals were proposed and evaluated using statistical pattern recognition classifiers. The authors were able to obtain an accuracy score of 61%.

Another paper [8], converted the music samples into fast fourier transforms and also extracted MFCCs. The authors explored Recurrent Neural Networks as a model for classification among other non-deep learning models and obtained a better result than the traditional machine learning models. They utilized 4 genres out of the 10 genres from the GTZAN Dataset and were able to get a accuracy score of 86%.

Paper [7], on the other hand, made use of spectrograms generated from the music samples in the dataset and passed them as images to a Convolutional Neural Network (CNN). The authors implemented a 2-layer CNN and were able to classify the 10 genres with an accuracy of 70% which is comparable to human-level accuracy in genre detection.

Finally, the paper I based the deep learning architecture for this project on, is paper [5], which utilizes a Convolutional Recurrent Neural Network (CRNN).

The authors compared the performances of CNNs with CRNNs on the Million Songs Dataset. They achieved a huge increase in accuracy using a combined CNN+RNN structure and were finally able to get an Area Under ROC Curve (AUC) score of 0.8662.

III. DATASET

The dataset used in this project is the GTZAN Genre Collection. It was originally used for the well known paper in genre classification "Musical genre classification of audio signals".

The files were collected in 2000-2001 from a variety of sources including personal CDs, radio, microphone recordings, in order to represent a variety of recording conditions. The dataset is 1.2 gb in size and consists of 1000 audio tracks each 30 seconds long. It contains 10 genres, each represented by 100 tracks. The tracks are all 22050Hz Mono 16-bit audio files in .wav format [1].

The different genres represented in the dataset are as follows - Blues, Classical, Country, Disco, HipHop, Jazz, Metal, Pop, Reggae and Rock. The classes in the dataset are thus completely balanced.

As the dataset is completely balanced, this eliminates the various problems often faced when dealing with imbalanced data. Due to this, this dataset is extremely useful in machine learning tasks as equal weightage to all genres can be provided which significantly improves classification performance.

IV. FEATURE ENGINEERING

The classification process in this project was carried out using two different methods as follows -

1) Conventional Machine Learning Techniques:
   For this method, we extract multiple temporal and spectral features explained in the following sections and pass them to traditional machine learning classifiers like Logistic Regression, Random Forest Classifier, etc.
2) Deep Learning Techniques:
   The second method utilized mel-spectrograms generated for each music sample and passed them as images to deep learning techniques like Convolutional Neural Networks (CNN) and Convolutional Recurrent Neural Networks (CRNN).

All of the features were extracted using a python library for audio and music analysis called LibROSA [2]. It provides the building blocks necessary to create music retrieval systems. The following sections explain the feature engineering process for the two methods in detail.

A. Temporal Features

Temporal features are the ones extracted from the time domain of the raw audio signal.

1) Basic Features: We first extract basic features like mean, standard deviation, skewness and kurtosis of the amplitude of the raw audio signal of the music sample. Skewness is the measure of the asymmetrical spread of the signal about its mean value. A distribution, or data set, is symmetric if it looks the same to the left and right of the center point. Such data would have a skewness of 0. Kurtosis is a measure of the "tailedness" of the signal. Data sets with high kurtosis tend to have heavy tails, or outliers. Data sets with low kurtosis tend to have light tails, or lack of outliers.

2) Tempo: Tempo is defined as the beats per minute of the music sample. It is a very important feature when it comes to differentiating music by genre. For example, genres like reggae generally have a lower tempo as compared to pop or rock music.

3) Root Mean Square Energy: The energy of a signal corresponding to the total magnitude of the signal. For audio signals, that roughly corresponds to how loud the signal is. The root-mean-square energy (RMSE) in a signal is defined as :

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{n=1}^{N}|x(n)|^2}$$

The RMSE is calculated separately for each frame and we take the mean and standard deviation of the RMSE over all frames as features.

4) Zero Crossing Rate: The zero-crossing rate (ZCR) 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 is a key feature to classify percussive sounds especially used in music retrieval systems. The ZCR is calculated by dividing the music sample into small sections of frame length 2048 and hop length of 512. Finally we take the mean and standard deviation of the ZCR over all these sections as features.

B. Spectral Features

The spectral features are obtained by converting the audio signal to the frequency domain using a Fourier Transform. We extract the following features from the music samples :

1) Spectral Centroid: The spectral centroid indicates where the "center of mass" of the spectrum is located. It is calculated as follows :

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

where $x(n)$ represents the weighted frequency value, or magnitude, of bin number n, and $f(n)$ represents the center frequency of that bin. Also, the spectral centroid is a good predictor of the "brightness" of a sound,it is widely used in digital audio and music processing as an automatic measure of musical timbre. The centroid is again calculated for each frame and we take the mean and standard deviation over all frames as features for a particular sample.

2) Spectral Bandwidth: The p^{th} order spectral bandwidth is the p^{th} order moment of the signal about the centroid. It is calculated as follows :

$$\text{Bandwidth} = (\sum_k S(k)(f(k) - f_c)^p)^{\frac{1}{p}}$$

where $S(k)$ is the spectral magnitude at frequency bin k, $f(k)$ is the frequency at bin k, and f_c is the spectral centroid. When $p = 2$, this is like a weighted standard deviation. We take the mean and standard deviations of the bandwidth vectors at $p = 2$, $p = 3$ and $p = 4$ as features.

3) Spectral Contrast: The spectral contrast is the difference in the maximum and minimum values within each frequency band. Each frame is divided into multiple frequency bands ($bands = 6$ for our project) to calculate the contrast. We take the mean and standard deviation of this value over all frames as features.

4) Spectral Rolloff: Spectral Rolloff the value of frequency under which the majority amount of energy lies. The default threshold for this feature is 85%.

5) Mel-Frequency Cepstral Coefficients (MFCC): Mel-frequency cepstral coefficients (MFCCs) are coefficients that collectively make up an MFC (Mel-frequency cepstrum).In sound processing, the mel-frequency cepstrum (MFC) is a representation of the short-term power spectrum of a sound, based on a linear cosine transform of a log power spectrum on a non-linear mel scale of frequency. MFCCs are commonly derived as follows:

1) Take the Fourier transform of (a windowed excerpt of) a signal.
2) Map the powers of the spectrum obtained above onto the mel scale, using triangular overlapping windows.
3) Take the logs of the powers at each of the mel frequencies.
4) Take the discrete cosine transform of the list of mel log powers, as if it were a signal.
5) The MFCCs are the amplitudes of the resulting spectrum.

Overall, we extracted 97 features in total. Before jumping into the implementation of this first method involving the temporal and spectral features, we'll first look at the creation of mel-spectrograms necessary for the second method involving CNN and CRNNs.

C. Mel-Spectrograms

A spectrogram is a visual representation of the spectrum of frequencies in a sound as they vary with time. A common format is a graph with two geometric dimensions: one axis represents time or RPM, the other axis is frequency; a third dimension indicating the amplitude of a particular frequency at a particular time is represented by the intensity or color of each point in the image. A mel-spectrogram is a spectrogram which has the y-axis of frequencies on the mel scale instead of a linear scale.

The human auditory system doesnt interpret pitch in a linear manner. The human interpretation of the pitch rises with the frequency, which in some applications may be a unwanted feature. To compensate for this the mel-scale was developed. The mel-scale was developed by experimenting with the human ears interpretation of a pitch in 1940s. The sole purpose of the experiment was to describe the human auditory system on a linear scale. The experiment showed that the pitch is linearly perceived in the frequency range 0-1000 hz. Above 1000 hz, the scale becomes logarithmic. An approximated formula widely used for mel-scale is shown below:

$$F_{mel} = \frac{1000}{log(2)} \cdot [1 + \frac{F_{Hz}}{1000}]$$

Fig. 1 shows the relationship between a conventional frequency scale and the logarithmic mel-scale.

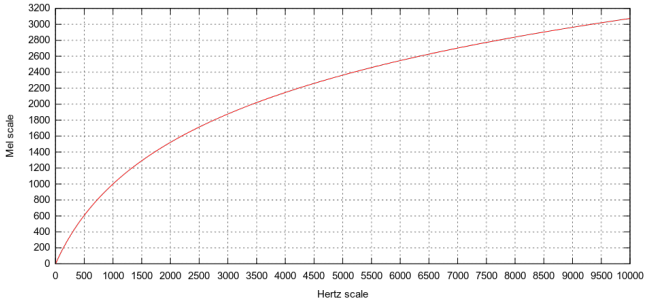

Fig. 1. Relationship between frequency scale and mel scale

For this project, the mel-spectrograms for each of

the 1000 music samples were generated and fed into the CNN and CRNNs. Fig. 2 shows a sample mel-spectrogram generated from one of the music samples in the pop genre.

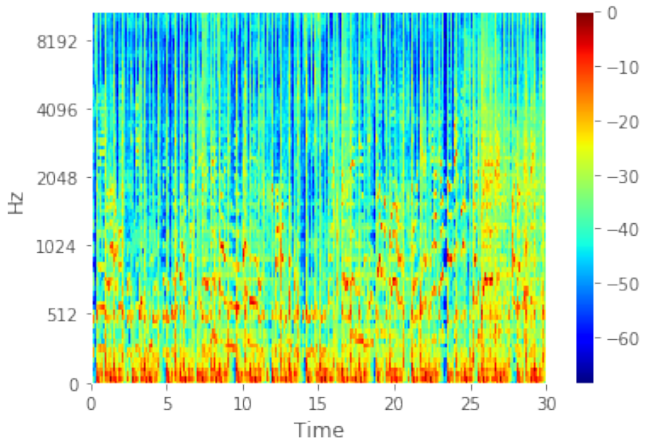

Fig. 2. Mel-Spectrogram for a music sample from the pop genre

V. IMPLEMENTATION

A. Method I - Conventional Machine Learning Techniques

Our problem is essentially a multi-class classification problem with 10 classes corresponding to the 10 genres. We tried out a variety of different classification techniques to get the best model. The methods we tried out are described below.

1) K-nearest neighbor: K-nearest neighbor is a lazy learning classification technique and this makes the training phase very fast. We tried this out for various values of k (the number of nearest neighbors). The model gave the best result for k=10 with an accuracy score of 0.59.

2) Random Forest Classifier: Random Forest Classifiers are ensemble methods which use a number of weak estimators to form a stronger classifier. The Random Forest Classifier gave us a much better performance than the base line KNN, and after tuning the parameters we were able to increase the accuracy from 0.65 to 0.72. The following parameters gave us the best performance - $n\_estimators = 300, max\_depth = 20, max\_features =' log2', min\_samples\_leaf = 3, min\_samples\_split = 8$.

3) Logistic Regression: Logistic Regression is a powerful statistical way of estimating the probabilities of each of the classes being predicted. This linear classifier is generally used for binary classification tasks. For this multi-class classification task, the LR is implemented as a one-vs-rest method. That is, 10 separate binary classifiers are trained. During test time, the class with the highest probability from among the 10 classifiers is chosen as the predicted class. We tuned the parameters C and $penalty$. C is the inverse of regularization strength. Like in support vector machines, smaller values specify stronger regularization. Either 'l1' or 'l2' norm used for penalization is specified by $penalty$. By tuning the parameters we got the best accuracy score of 0.78 using $penalty = "l1"$ and $C = 1.0$.

4) XGBoost: XGBoost is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable. It implements machine learning algorithms under the Gradient Boosting framework. XGBoost proved to be an extremely efficient machine learning framework for our project. Initially, using only default parameter for the implementation, we obtained an accuracy score of 0.78, similar to the Logistic Regression. Then, by fine-tuning the parameters using GridSearchCV, we were able to increase the accuracy to 0.80. The following parameters gave the best performance - $learning\_rate = 0.1, max\_depth = 3, min\_child\_weight = 1, n\_estimators = 1000$.

B. Method II - Deep Learning

In method II, we fed the generated mel-spectrograms to two different classifiers - Convolutional Neural Network (CNN) and Convolutional Recurrent Neural Network (CRNN) in order to compare the effect of adding recurrent neural network layers to a CNN and seeing how much it would increase the classification performance. We used accuracy score and categorical cross-entropy loss as metrics to analyze the results in this method. The separate loss for each class is calculated as follows -

$$\text{Loss} = -\sum_{c=1}^{M} y_{o,c}\, log(p_{o,c})$$

where M is the number of classes, y is the binary indicator (0 or 1) if class label c is the correct classification for observation o and p is the predicted probability that observation o is of class c.

1) Convolutional Neural Network: Convolutional Neural Networks (ConvNets or CNNs) are a category of Neural Networks that have proven very effective in areas such as image recognition and classification. ConvNets have been successful in identifying faces, objects and traffic signs apart from powering vision in robots and self driving cars.

The architecture of CNN used for this project consisted of 5 convolutional blocks stacked together. Each of the convolution blocks was made up of the following components :

- Convolution Layer: The convolutional layer is the core building block of a CNN. The layer's parameters consist of a set of learnable filters. Each filter computes the dot product between the entries of the filter and the input and produces a 2-dimensional activation map of that filter. As a result, the network learns filters that activate when it detects some specific type of feature at some spatial position in the input.
- Pooling Layer: Pooling is used for dimension reduction of the feature map obtained from the convolution step. For example, by max pooling with 2x2 window size, we only retain the element with the maximum value among the 4 elements of the feature map that are covered in this window.
- ReLU Layer: ReLU stands for Rectified Linear Unit which applies an activation function that removes negative values from an activation map by setting them to zero. It increases the non-linearity of the decision function.
- Dropout Layer: Neural networks are prone to over-fitting, which is why we need to utilize certain regularization methods. Dropout is one such regularization method which shuts off some of the neurons randomly during training. This makes the model generalize without any heavy dependence on a subset of the neurons. A dropout rate of 0.25 is used, which means that a given weight is set to zero during an iteration, with a probability of 0.25.

The output from the 5-layer CNN architecture is fed into a fully connected layer which outputs the class probabilities for each of the 10 genre classes using a softmax activation function. Finally the categorical cross-entropy loss is calculated and is used to back-propagate the error, compute the gradients and thereby update the weights of the network until the loss converges to a minimum value.

2) Convolutional Recurrent Neural Network: The convolutional recurrent neural network (CRNN) adds a 2-layer Recurrent Neural Network (RNN) with gated recurrent units (GRU) on top of a 4-layer CNN. In this architecture, the RNN acts as a temporal summarizer for the feature extractor CNN. This takes into account both a local as well as a global structure. Fig. 3 shows the architecture of the CRNN.

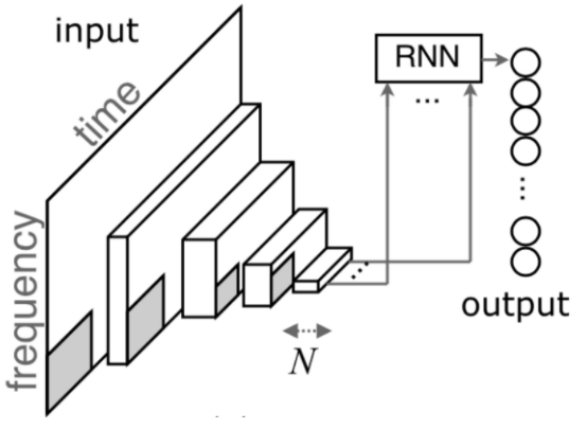

Fig. 3. Convolutional Recurrent Neural Network Architecture

This architecture has been adopted from [5], in which the authors utilized this architecture on Million Songs Dataset with great results. I wanted to compare how this particular architecture would perform on a different dataset such as the GTZAN Dataset used in this project.

All the convolutional and fully-connected layers are equipped with identical optimization techniques and activation functions - batch normalization and ELU activation function. CRNN also has weak dropout (0.1) between convolutional layers to prevent over-fitting of the RNN layers.

We used 64% training, 18% validation and 20% testing data for both the CNN and CRNN models. In order to accurately compare the results, the test and train sets for both the conventional as well deep learning methods were kept the same.

C. Libraries

The project is implemented in Python using Tensorflow [4], Scikit-Learn [10] and Keras [6]. Parameter tuning on the models is carried out with GridSearchCV from the Scikit-Learn library.

VI. RESULTS AND DISCUSSION

The following tables show the results obtained from the two methods carried out in this project.

As is evident from Table I, XGBoost performed the best with an accuracy score of 80%, which provides a great improvement over KNN's 59% accuracy score. This was expected as XGBoost is known to be a powerful classifier. What was surprising however, was that Logistic Regression performed significantly better than Random Forest.

TABLE I

CONVENTIONAL MACHINE LEARNING MODELS

Model	Accuracy	AUC ROC
K-Nearest Neighbors	0.59	0.884
Random Forest Classifier	0.72	0.955
Logistic Regression	0.78	0.964
XGBoost	**0.80**	**0.974**

TABLE II

DEEP LEARNING

Model	Accuracy	Loss
CNN	0.83	0.508
CRNN	**0.86**	**0.468**

Table II shows the deep learning methods utilized using only the generated mel-spectrogram images. CRNN gave the best result of 86%, a huge improvement over the conventional machine learning classifiers and also better than using just the CNN, which gave an accuracy of 83.8%. This supports our hypotheses that utilizing a CNN+RNN architecture would improve performance of the classifier. This is a huge improvement over [7], which gave a score of 70% for 10-genre classification using the GTZAN Dataset.

A. *Feature Importance from XGBoost*

In order to analyze which features contribute most to the classification process, we calculate the feature importance of all the features. Features that were used more times as a decision node in the XGBoost gradient boosting trees have higher importance. The following figure shows the top 15 most important features obtained from the XGBoost classifier.

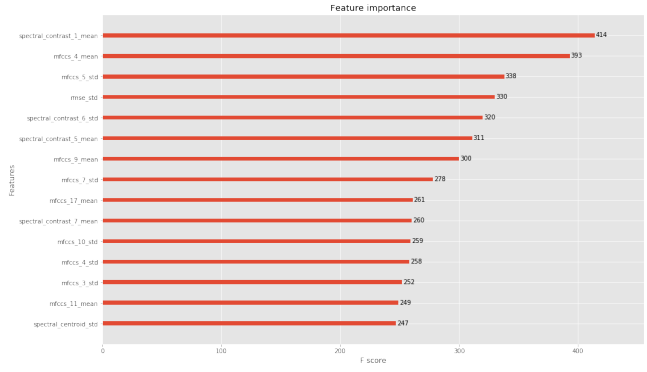

Fig. 4. Feature Importance

As is evident from the feature importance plot, MFCCs appear most number of times, followed by spectral contrast and standard deviation of the RMSE. MFCCs have shown to be extremely useful in speech recognition and music tagging, so it's no surprise that they have proven to be the most important features here as well.

B. *Confusion Matrix and ROC Curve for XGBoost*

The following figures show the confusion matrix and ROC curve obtained from the predictions given by the XGBoost classifier.

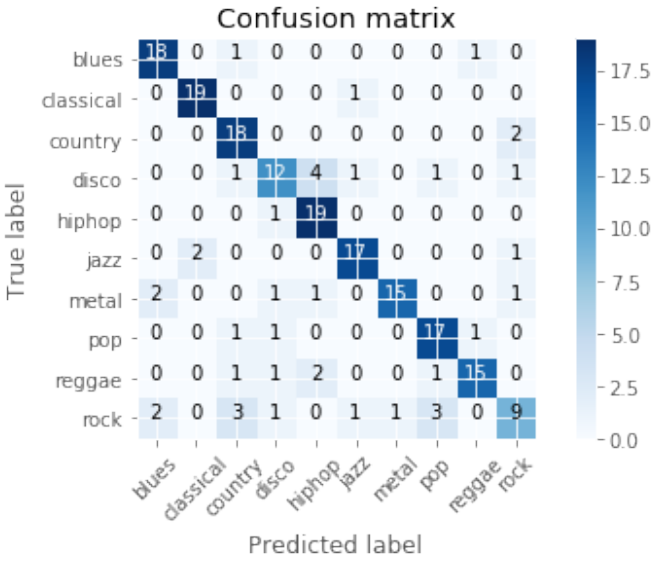

Fig. 5. Confusion Matrix for XGBoost

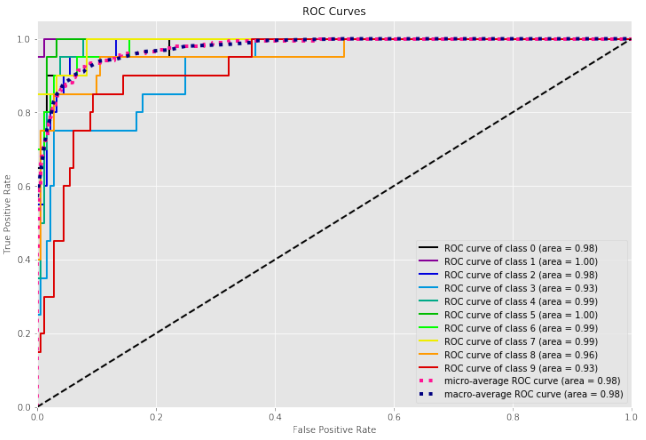

Fig. 6. ROC Curve for XGBoost

The Fig. 5 shows the confusion matrix for our best performing model using XGBoost. As you can see from the matrix, majority of the visits are being correctly classified by the model with a few exceptions. One

such exception is the genre rock, which had the least correctly classified samples out of all of the the genres.

Fig. 6 shows the ROC curves for each genre as well as the micro and macro average ROC curves. Classes 3 and 9 show a smaller area under the ROC curve, which are classes disco and rock respectively. This makes sense as these two were the ones with the lease correctly classified samples.

C. Confusion Matrix for CRNN

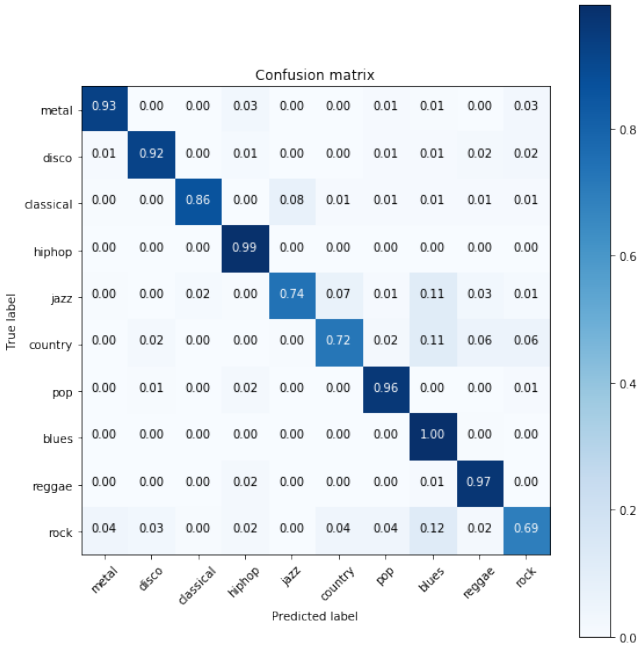

Fig. 7. Confusion Matrix for CRNN

Similar to the confusion matrix shown by XGBoost, rock genre using CRNN is also misclassified. However, CRNN shows a much higher classification accuracy for the disco genre as compared to XGBoost.

VII. Future Work

In the future, more analysis could be done to figure out why certain genres have a better classification accuracy as well as how to distinguish between similar genres. Another future path would be to try out an ensemble classifier of XGBoost and CRNN and see if it proves to be even better in classification than either of them, utilizing features from both the different methods.

VIII. Conclusion

Overall, two different methods were carried out in order to perform music genre classification - extracting temporal and spectral features and generating mel-spectrograms. We classified the music samples using conventional machine learning techniques for the extracted features as well as deep learning using image classification for the mel-spectrograms. XGBoost proved to be the best conventional classifier giving an accuracy score of 80% whereas CRNN gave the best score of 86% using the mel-spectrograms. CRNN also gave a considerable jump in accuracy from a classic CNN architecture. We also analyzed the top most important features for the XGBoost classifier and found that MFCCs and spectral contrast contribute the most in the classification.

References

[1] Gtzan dataset. `http://marsyasweb.appspot.com/download/data_sets/`.

[2] Librosa. `https://librosa.github.io`.

[3] Music genre. `https://en.wikipedia.org/wiki/Music_genre`.

[4] Tensorflow. `https://www.tensorflow.org/`.

[5] Keunwoo Choi, György Fazekas, Mark Sandler, and Kyunghyun Cho. Convolutional recurrent neural networks for music classification. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2392–2396. IEEE, 2017.

[6] François Chollet et al. Keras. `https://keras.io`, 2015.

[7] Mingwen Dong. Convolutional neural network achieves human-level accuracy in music genre classification. *arXiv preprint arXiv:1802.09697*, 2018.

[8] D Pradeep Kumar, BJ Sowmya, KG Srinivasa, et al. A comparative study of classifiers for music genre classification based on feature extractors. In *Distributed Computing, VLSI, Electrical Circuits and Robotics (DISCOVER), IEEE*, pages 190–194. IEEE, 2016.

[9] Jin Ha Lee and J Stephen Downie. Survey of music information needs, uses, and seeking behaviours: Preliminary findings. In *ISMIR*, volume 2004, page 5th. Citeseer, 2004.

[10] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[11] George Tzanetakis and Perry Cook. Musical genre classification of audio signals. *IEEE Transactions on speech and audio processing*, 10(5):293–302, 2002.