Music Genre Recognition

Krish Hemant Mhatre

Indiana University

Abstract

As the digital music industry continues to grow rapidly, the demand for better services like recommenders, search results, and automatic labelling is increasing. Demand for an efficient automated music genre recognition system is also increasing. Such a system could be built using various methods like machine learning algorithms (KNNs, Random Forests, SVMs etc.) and neural networks. Neural networks can be trained to predict the genre of a music file more efficiently. This can be done using two general methods - feature extraction and spectrogram analysis. This project involves training and analyzing performances of various neural networks using spectrograms of the audio files. The project mainly focuses on fully connected neural network, 1-dimensional convolutional neural network (1D-CNN), 2-dimensional convolutional neural network (2D-CNN), and convolutional recurrent neural network (CRNN).

1. Introduction

The digital music industry achieved skyrocketing growth over the past decade, especially in the United States. An average digital music user in the US is estimated to spend $50.47 annually on streaming music by 2022 [1]. Also the number of such users is projected to reach 168 million by 2022 [1]. As the industry continues to grow, the market is estimated to get more competitive. Current major players in this industry include Youtube Music, Amazon Prime Music, Apple Music, Spotify, Deezer, iHeartRadio, Pandora, SoundCloud and Tidal. As the competition gets more intense, the need for providing the best user experience increases tremendously. An application that provides better search results and recommendations will certainly have an edge over its competitors. Some of these music applications like Youtube Music and SoundCloud allow anyone to upload their music, which makes it difficult to provide efficient recommendation and search results. In order to automate such a process, there is a need for an efficient genre recognition system and automated tag generation system right at the time of upload. This project was aimed to develop an automated genre recognition system using deep neural networks.

2. Related Work

There have been several projects successfully completed with the aim of classifying the genre of a music file. The most basic way to solve this problem is using K-Nearest Neighbors [3]. There have been projects which combine basic machine learning algorithms (GDA, Random Forests and SVMs) with neural networks [4]. But most of the previously done related projects use deep neural networks. These projects use two general ways for classification - Feature extraction and Spectrogram analysis. Feature extraction algorithm uses features like MFCC, Spectral Centroid, Chroma and Spectral Contrast for classification [5]. Spectrogram Analysis uses decibel-scaled spectrogram images for classification.

3. Dataset

The project uses GTZAN Keras data as the primary dataset [2]. The dataset consists of 1000 music tracks from 10 different genres (100 per genre) - rock, reggae, pop, metal, jazz, hiphop, disco, country, classical and blues. The format of the dataset is represented in Figure 3.1.

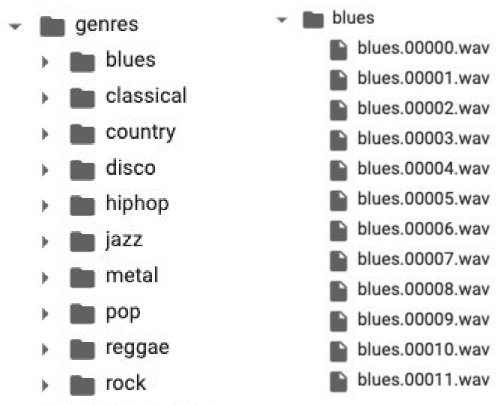

Figure 3.1 - Dataset Structure

4. Methodology

4.1 Importing Data

The filenames of sound tracks in the dataset are imported using “pathlib” library. These filenames are then used to read the audio files (.wav format) using Tensorflow library. The data is divided into two - audio binaries and labels. This data is, then, preprocessed further.

4.2 Data Preprocessing

The audio binaries obtained previously are then decoded using Tensorflow to get audio signals. These audio signals are represented in Figure 4.2.

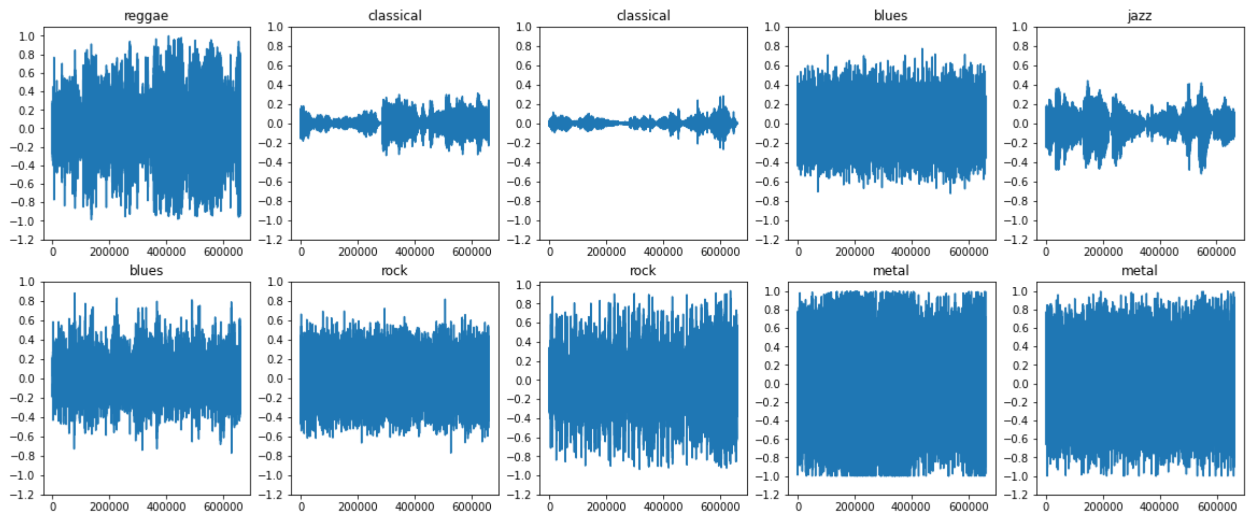

Figure 4.2 - Audio Waves

These audio waves are converted into Numpy arrays and are made uniform in length by padding with zeros. The uniform length arrays are passed through a Short-Time Fourier Transform using Librosa library as shown in Equation ## [7].

$$\mathbf{STFT}\{x(t)\}(\tau,\omega) \equiv X(\tau,\omega) = \int_{-\infty}^{\infty} x(t)w(t-\tau)e^{-i\omega t}\,dt$$

Equation 4.2 [7]

These 2-Dimensional arrays of shape 1025 x 1320 are then converted into absolute values. Now these amplitude spectrograms are converted to dB-scaled spectrograms using Librosa library. These spectrograms are further normalized and scaled between 0 and 1. The resulting spectrograms are represented by Figure 4.3.

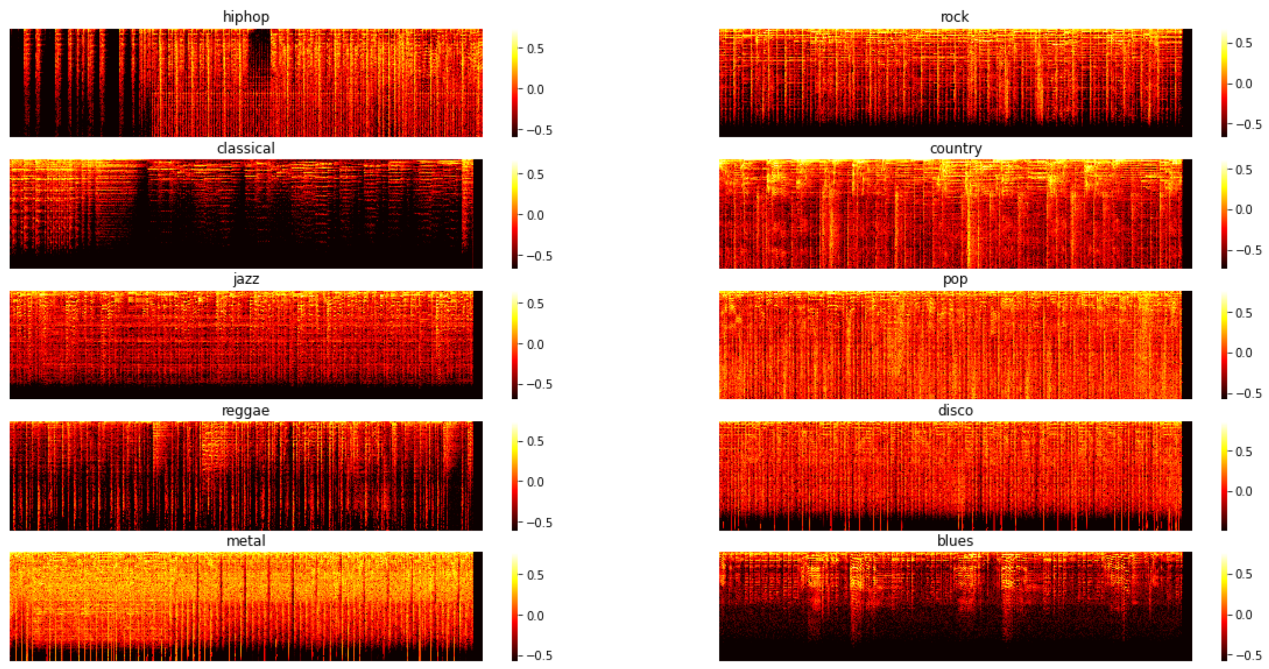

Figure 4.3 - Spectrograms

The labels of the audio signals are encoded into categorical data using the LabelEncoder function from Scikit-learn. This encoder is saved for decoding the predicted labels.

4.3 Neural Network Models

The project initially aimed to build a genre classifier using machine learning algorithms but eventually was made to focus on deep learning algorithms only because of such a large dataset. There are several ways of building a classifier for genre recognition. The two primary ways are Feature Extraction and Spectrogram Analysis. This project focuses on Spectrogram Analysis for genre classification. Four deep learning models have been trained and the performances have been analyzed using the spectrogram data.

The first model to be trained was a Fully Connected Neural Network. The architecture of the network is as follows-

1. Input layer with shape - (Batch Size x 1025 x 1320).
2. Flatten layer.
3. Dense layer with 512 hidden units and ReLU Activation function.
4. Dense layer with 512 hidden units and ReLU Activation function.
5. Dense layer with 512 hidden units and ReLU Activation function.
6. Dense layer with 512 hidden units and ReLU Activation function.

7. Dense layer with 512 hidden units and ReLU Activation function.
8. Output dense layer with 10 units and Softmax Activation function.

The second model trained was a 1-Dimensional Convolutional Neural Network. The architecture of the network is as follows-

1. Input layer with shape - (Batch Size x 1025 x 1320).
2. Convolutional 1D layer with 16 filters, kernel size of 3 and ReLU Activation function.
3. Convolutional 1D layer with 16 filters, kernel size of 3 and ReLU Activation function.
4. 1D Max Pooling with pool size of 2.
5. Dropout layer with a rate of 0.2
6. Convolutional 1D layer with 32 filters, kernel size of 3 and ReLU Activation function.
7. Convolutional 1D layer with 32 filters, kernel size of 3 and ReLU Activation function.
8. 1D Max Pooling with pool size of 2.
9. Dropout layer with a rate of 0.2
10. Convolutional 1D layer with 64 filters, kernel size of 3 and ReLU Activation function.
11. Convolutional 1D layer with 64 filters, kernel size of 3 and ReLU Activation function.
12. 1D Max Pooling with pool size of 2.
13. Dropout layer with a rate of 0.2
14. Flatten layer.
15. Dense layer with 512 hidden units and ReLU Activation function.
16. Dense layer with 512 hidden units and ReLU Activation function.
17. Output dense layer with 10 units and Softmax Activation function.

The third model trained was a 2-Dimensional Convolutional Neural Network. The input for training a 2D-CNN had to be reshaped to 4D. The architecture of the network was as follows -

1. Input layer with shape - (Batch Size x 1025 x 1320 x 1).
2. Convolutional 2D layer with 10 filters, kernel size of (5, 5) and ReLU Activation function.
3. 2D Max Pooling with pool size of 2 and 2 strides.
4. Convolutional 2D layer with 10 filters, kernel size of (5, 5) and ReLU Activation function.
5. 2D Max Pooling with pool size of 2 and 2 strides.
6. Flatten layer.
7. Dense layer with 512 hidden units and ReLU Activation function.
8. Output dense layer with 10 units and Softmax Activation function.

The fourth model trained was a Convolutional-Recurrent Neural Network (CRNN). The architecture of the network was as follows -

1. Input layer with shape - (Batch Size x 1025 x 1320).
2. Convolutional 1D layer with 16 filters, kernel size of 3 and ReLU Activation function.
3. Convolutional 1D layer with 16 filters, kernel size of 3 and ReLU Activation function.
4. 1D Max Pooling with pool size of 2.
5. Batch Normalization Layer.
6. Dropout layer with a rate of 0.2
7. Convolutional 1D layer with 32 filters, kernel size of 3 and ReLU Activation function.
8. Convolutional 1D layer with 32 filters, kernel size of 3 and ReLU Activation function.
9. 1D Max Pooling with pool size of 2.
10. Batch Normalization Layer.
11. Dropout layer with a rate of 0.2
12. Convolutional 1D layer with 64 filters, kernel size of 3 and ReLU Activation function.
13. Convolutional 1D layer with 64 filters, kernel size of 3 and ReLU Activation function.
14. 1D Max Pooling with pool size of 2.
15. Batch Normalization Layer.
16. Dropout layer with a rate of 0.2
17. LSTM layer with 96 units and Tanh Activation function.
18. Flatten layer.
19. Dense layer with 64 hidden units and ReLU Activation function.
20. Output dense layer with 10 units and Softmax Activation function.

4.4 Training the Models

All of the four models were compiled and trained in the same way. Adam optimizer with a learning rate 0.001 was used along with Sparse Categorical Cross-entropy as the loss function. The accuracy metric was used for training and validation. All the models were trained for 100 epochs (except for 2D-CNN which was trained for 50 epochs) with a validation split of 20%. The training results are represented in Figure 4.4.

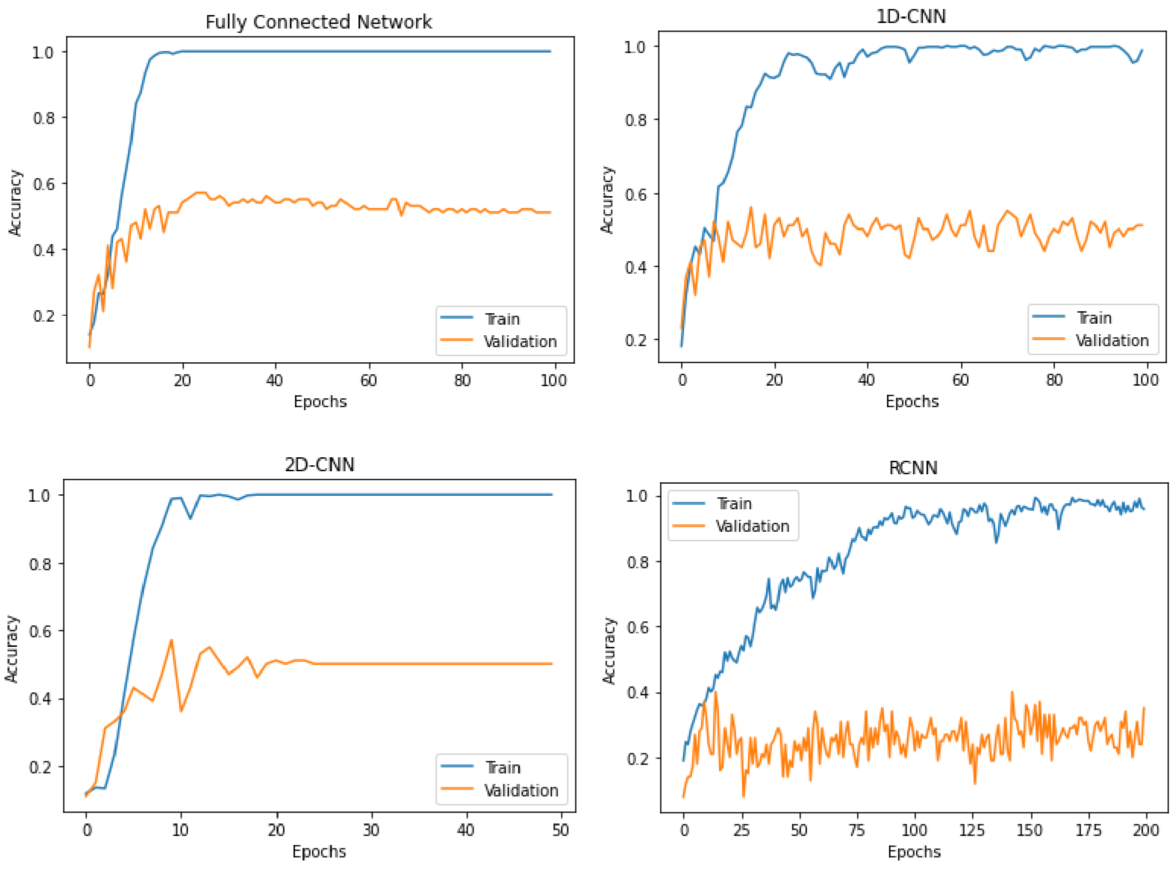

Figure 4.4 - Training Results

5. Results

Each model is tested using 300 music files. These files are preprocessed in the same way as the training files. These audio signals are used to predict numerical labels. These numerical labels are inverse transformed to get the original label name using the previously saved LabelEncoder. Performance of the models are analyzed using a Confusion matrix of predictions and true values for each model. Figure 5.1 shows confusion matrix for each model.

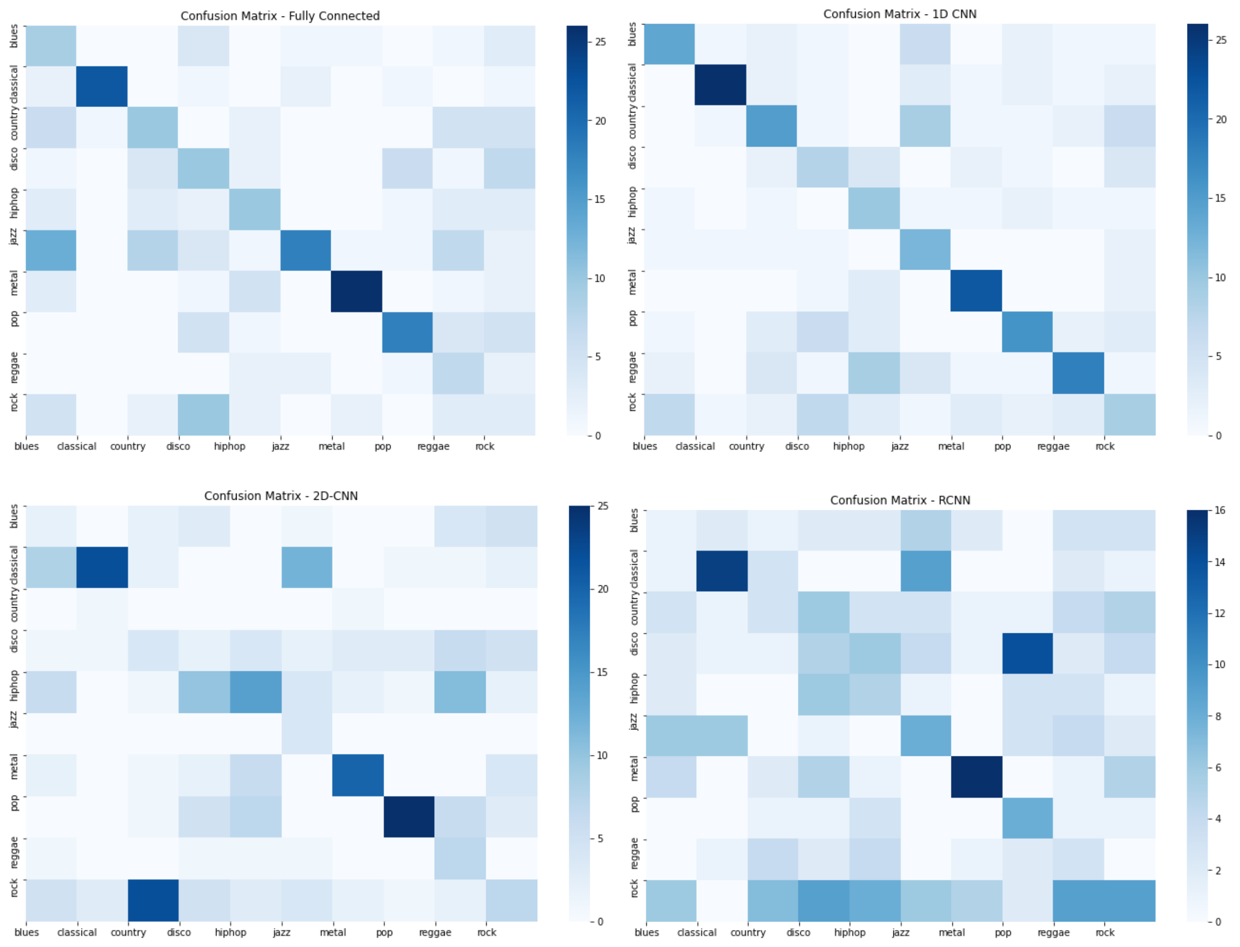

Figure 5.1 - Confusion Matrix

6. Conclusions

Music genre recognition using deep learning is efficient enough using Spectrogram Analysis. The validation accuracy for all models might not align with the expected results but the analysis of predictions using a confusion matrix shows that most of the results are aligned with the expectations. Based on current models, 1-dimensional Convolutional Neural Network is the most efficient model of genre classifier. Convolutional-Recurrent Neural Network (CRNN) might be more efficient in the Feature Extraction method [5]. 2-dimensional Convolutional Network might be fine-tuned further to get better performance. Genre recognition could be made highly efficient by combining this spectrogram analysis method with feature extraction [6]. The project is maintained on GitHub - https://github.com/krishmhatre/MusicGenreRecognition.

References

[1] Watson, Amy. “Digital Music - Statistics & Facts.” *Statista*, Nov. 2018, https://www.statista.com/topics/1386/digital-music/.

[2] Hguimaraes. “Gtzan.keras.” *GitHub*, https://github.com/Hguimaraes/gtzan.keras.

[3] “Python Project - Music Genre Classification.” *DataFlair*, 6 Aug. 2020, https://data-flair.training/blogs/python-project-music-genre-classification/.

[4] Li, Haojun, et al. “Combining CNN and Classical Algorithms for Music Genre Classification.” *ICME, Stanford University*, Department of CS, Stanford University, https://cs229.stanford.edu/proj2018/report/19.pdf.

[5] Ruotsi, Ruoho. “LSTM-Music-Genre-Classification.” *GitHub*, https://github.com/ruohoruotsi/LSTM-Music-Genre-Classification.

[6] Choi, Keunwoo, et al. “CONVOLUTIONAL RECURRENT NEURAL NETWORKS FOR MUSIC CLASSIFICATION.” Arxiv, Cornell University, 21 Dec. 2016, https://arxiv.org/pdf/1609.04243.pdf.

[7] Sejdić, Ervin, et al. “Time–Frequency Feature Representation Using Energy Concentration: An Overview of Recent Advances.” *Digital Signal Processing*, ScienceDirect, 16 Feb. 2008, https://www.sciencedirect.com/science/article/abs/pii/S105120040800002X?via=ihub.