Deep Learning for Music Genre Recognition

Priyanka Dwivedi
SUID: pdwivedi
Stanford University
pdwivedi@stanford.edu

Abstract

Audio data is becoming an important source of information in machine learning. We are using voice more and more to interact with smart agents like Siri and Alexa. One easily available source of audio data is music records. In recent years deep learning has had good success in various tasks like music genre classification, music recommendation, music generation etc. In this report, I explore two different deep learning architectures for classifying music audio files into eight different genres. The two architectures use a combination of convolutional and recurrent layers to extract features from both frequency and time dimension. I present the best results from both the models. I have also tried to build a deeper understanding of the deep learning model by looking at activation visualization to see what features different layers of the convolution model are focusing on. Finally I have extracted embeddings from the trained model to perform clustering which can be used for the task of music recommendation.

1 Introduction

Music Genre Classification is a popular problem in machine learning with several applications.It can be used to tag every song in a huge music corpus with genre and sub genre which can be then be used to identify similar songs. Another application is for a music recommendation system. By using the features from an intermediate layer in the model we can cluster similar songs and those can be shared with users based on their preference.
Traditionally we have extracted features from music audio like MFCC and used those as the starting point for a classification task. I have done this in my baseline model presented in section 3. However a better approach is to let a neural network itself identify features that may be relevant. In this project, I first converted each audio music file into a mel spectogram. A spectogram is a visual representation of amplitude in different frequency bands over time. The spectograms also look visibly different for different genres. One can then think of a spectogram as an image and use these as an input to a convolutional recurrent model where the convolutional and recurrent layers either in parallel or one after the other extract features from the spectogram to perform classification. This report is organized into several different sections - section 1 defines the problem, section 2 describes the data set and generation of mel spectograms, section 3 covers the deep learning architectures we used and their results, in section 4 we deep dive into the model using activation visualizations and embedding clustering and we conclude in section 5.
The code for this project has been written in jupyter notebooks and uploaded to CS221 Project Git hub. The README includes link to the data set with spectograms, any dependencies that need to be installed and instructions on what each notebook does. The goal has been to make results fully reproducible.

1.1 Related work

There has been quite bit of work on using convolutional, recurrent, combination of both and even feed forward networks for the purpose of music genre classification. The most influential papers for me were, Keunwoo et al. [1] which presents a convolutional recurrent model for recognizing genre, moods, instruments and era from the Million Songs data set. They used a 2D convolution model followed by recurrent layers and fully connected layers to perform the classification tasks. My CRNN model described in section 3.2 is similar but I used a 1D convolution layers instead of 2D. Piotr Kozakowski Bartosz Michalak, Deep Sound [9] used a 1D convolution model followed by time distributed dense layer on GTZan dataset. I got the idea for 1D convolution layers from them but found that for my data set the RNN layers after 1D CNN performed better.
Lin Feng, Shenlan Liu [2] paralleled CNN and RNN blocks to allow the RNN layer to work on the raw spectograms instead of the output from the CNN. My parallel CNN-RNN model was heavily influenced by this paper and my final architecture is similar to theirs with some modifications since my data set size was smaller.

2 Approach

In this section, I will talk about our choice of data set and how audio files are converted into a spectogram

2.1 Choice of Dataset

I came across three different data sets that could be used for this task. These were GTZan [3], Million Songs data set (MSD)[4] and Free Music Archive(FMA) [5]. The GTZan data set has 1000 songs of 30 seconds each divided equally into 10 genres. I decided to not use this data set since it only provided 100 songs per genre which I suspected would be a very small sample for a deep learning project and there has been a paper [6] on the faults in the data set which mentions repetitions, mislabeling and distortions. The MSD data set has metadata for million songs. This metadata includes information on artists and tracks including track duration, tempo, pitch and timbre features. However raw audio file for each song as well as its genre is not available. I was able to get genre information for a subset of songs from Tagtraum but downloading each raw audio file from 7digitial.com became quite cumbersome. So finally I decided to use the FMA data set. This data set is divided into small, medium and large subsets. The FMA small has 8000 songs equally distributed among 8 genres. The data set is further split into training set (6400 songs), validation set (800 songs) and a testing set (800 songs). For each song there is a 30 second audio as well as metadata related to genre, MFCC and Chroma features. The eight genres are Electronic, Experimental, Folk, Hip-Hop, Instrumental, International, Pop and Rock.

2.2 Mel Spectogram Generation

Each audio file was converted into a spectogram which is a visual representation of spectrum of frequencies over time. A regular spectogram is squared magnitude of the short term Fourier transform (STFT) of the audio signal. This regular spectogram is squashed using mel scale to convert the audio frequencies into something a human is more able to understand. I used the built in function in the librosa library to convert the audio file directly into a mel-spectogram. The most important parameters used in the transformation are - window length which indicates the window of time to perform Fourier Transform on and hop length which is the number of samples between successive frames. The typical window length for this transformation is 2048 which converts to about 10ms, the shortest reasonable period a human ear can distinguish. The hop length of 512 was chosen.
Further more the Mel-spectrograms produced by Librosa were scaled by a log function. This maps the sound data to the normal logarithmic scale used to determine loudness in decibels (dB) as it relates to the human-perceived pitch. As a result of this transformation each audio signal gets converted to a mel-spectogram of shape - 640, 128.
Figure 1 and 2 below show the mel-spectograms for the eight genres. We can see that spectograms for the eight classes are relatively distinct which implies CNN can extract distinct features from them. For some classes the spectograms do look similar, example Pop (Figure 1, top left), Rock (Figure 2, top left) and Hip Hop (Figure 2, bottom right) and as we see later, the model does con-

fuse between them. All the audio files were converted into spectograms and the spectograms were pickled. This made model training and validation much faster.

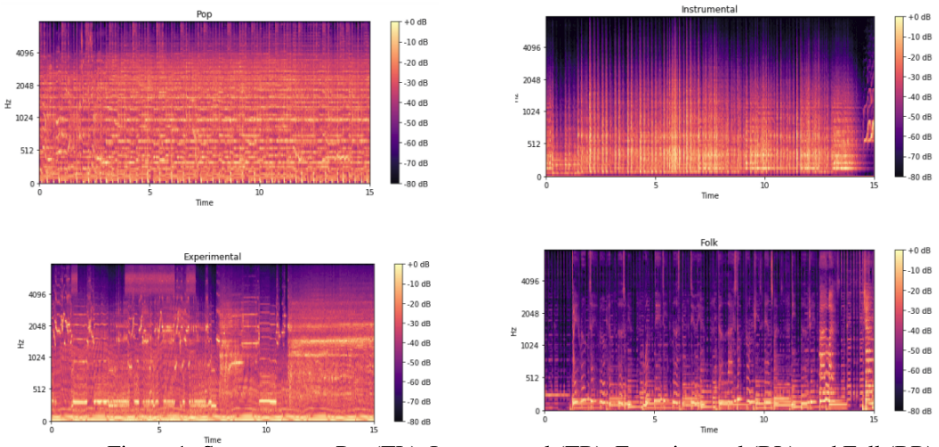

Figure 1: Spectogram - Pop(TL), Instrumental (TR), Experimental (BL) and Folk(BR)

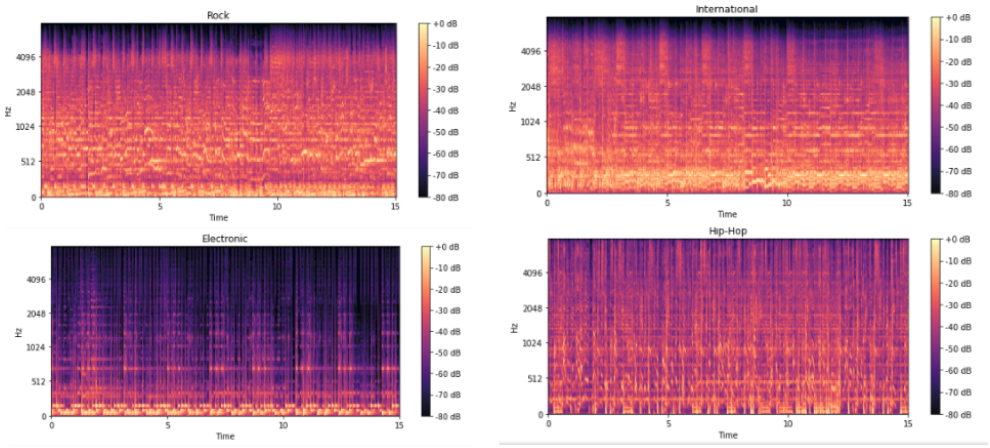

Figure 2: Spectogram - Rock(TL), International (TR), Electronic (BL) and Hip Hop(BR)

3 Modeling Approach and Results

3.1 Baseline Model

I built a baseline model that used the MFCC features provided in the FMA data set. MFCC stands for Mel Frequency Cepstral Coefficients which have been quite successful in speech recognition. The FMA data set provided 140 MFCC features per song.
For building the baseline model, I standardized the MFCC features by subtracting mean and scaling to unit variance. I then tried several different classifier from the sklearn library like Decision Tree Classifier, Support Vector Classifier, Random Forest Classifier and Logistic Regression. I got the best performance using a Support Vector Classifier. The performance of this model on the test set is shown in table 1 below. It had an overall accuracy of 46.3% on the test set.

3.2 Convolutional Recurrent (CRNN) Model

Convolutional Neural Networks (CNNs) are commonly used in image recognition related tasks. They perform convolution operation instead of matrix multiplication and are typically used in the early layers for understanding the 2D layout of the data. On the other hand RNNs excel in understand

	Precision	Recall	F1 score
Electronic	0.44	0.60	0.51
Experimental	0.34	0.44	0.38
Folk	0.24	0.20	0.22
Hip-Hop	0.63	0.68	0.65
Instrumental	0.51	0.42	0.46
International	0.54	0.48	0.51
Pop	0.34	0.25	0.29
Rock	0.66	0.64	0.65

Table 1: Precision, Recall and F1 score of baseline model

sequential data by making the hidden state at time t dependent on hidden state at time t-1. The neural network that we have built here use 1D convolution layers that perform convolution operation just across the time dimension. Each 1D convolution layer extracts features from a small slice of the spectogram as shown in Figure 4 below. RELU activation is applied after the Convolution operation. Batch normalization is done and finally 1D Max Pooling is performed which reduces spatial dimension of the image and prevents over fitting. This chain of operations - 1D Convolution - RELU - Batch Normalization - 1D Max Pooling is performed 3 times. The key parameters used are: Kernel Size of 5 and 56 filters per layer.

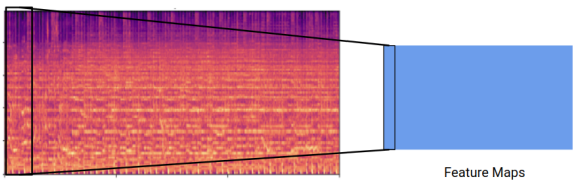

Figure 3: 1D Convolution done by the CRNN model

The output from 1D Convolution Layer is fed into a LSTM which should find short term and long term structure of the song. The LSTM uses 96 hidden units. The output from LSTM is passed into a Dense Layer of 64 units. The final output layer of the model is a dense layer with Softmax activation and 8 hidden units to assign probability to the 8 classes. Both dropout and L2 regularization were used between all the layers to reduce over fitting of the model. Figure 5 below shows the overall architecture of the model.

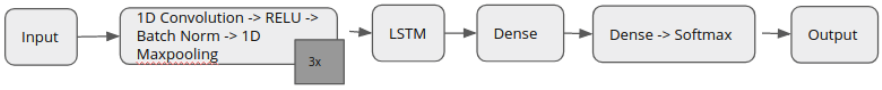

Figure 4: CRNN Model Architecture

The model was trained using Adam optimizer with a learning rate of 0.001 and the loss function was categorical cross entropy. The model was trained for 70 epochs and Learning Rate was reduced if the validation accuracy plateaued for at least 10 epochs.

After hyper parameter tuning, the best trained model had an overall accuracy of 44.1% on the test set. Figure 4 shows the loss and accuracy curves for training and validation samples. As seen, the model has low bias but high variance implying the model is over fitting a bit to training even after using several regularization techniques.

Table 2 shows the precision, recall and F1 score of the CRNN model on the test set. While this model has very similar performance as the baseline model, class wise F1 scores are quite different. This model performs better than baseline for International, Hip Hop and Electronic genres.

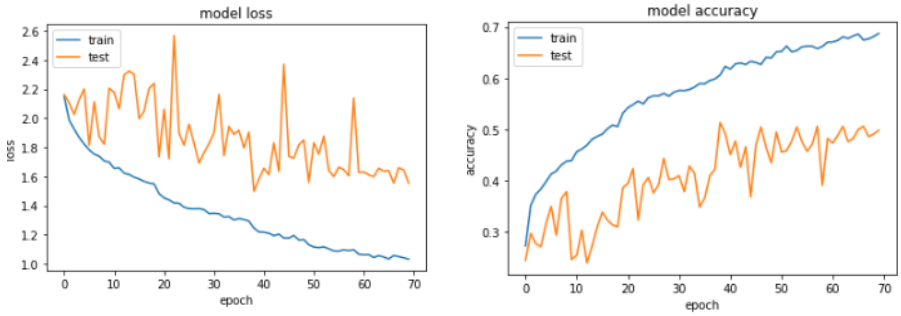

Figure 5: Training and Validation Loss and Accuracy Measures

	Precision	Recall	F1 score
Electronic	0.64	0.49	0.55
Experimental	0.28	0.22	0.25
Folk	0.19	0.16	0.17
Hip-Hop	0.57	0.85	0.68
Instrumental	0.36	0.34	0.35
International	0.46	0.64	0.54
Pop	0.30	0.27	0.28
Rock	0.64	0.56	0.60

Table 2: Precision, Recall and F1 score of CRNN model

3.3 Parallel CNN RNN Model

Inspired by the work in [2], I also tried a Parallel CNN-RNN Model. The key idea behind this network is that even though CRNN has RNNs to be the temporal summarizer, it can only summarize temporal information from the output of CNNs. The temporal relationships of original musical signals are not preserved during operations with CNNs. This model passes the input spectogram through both CNN and RNN layers in parallel, concatenating their output and then sending this through a dense layer with softmax activation to perform classification.

The convolutional block of the model consists of 2D convolution layer followed by a 2D Max pooling layer. This is in contrast to the CRNN model that uses 1D convolution and max pooling layers. There are 5 blocks of Convolution Max pooling layers. The kernel size if 3,1 for all 5 blocks. The filter sizes are 16 for the first block, 32 for the second block and 64 for the remaining 3 blocks. RELU activation is applied after each convolution. The final output is flattened and is a tensor of shape None , 256.

The recurrent block starts with 2D max pooling layer of pool size 4,2 to reduce the size of the spectogram before LSTM operation. This feature reduction was done primarily to speed up processing. The reduced image is sent to a bidirectional GRU with 64 units. The output from this layer is a tensor of shape None, 128.

The outputs from the convolutional and recurrent blocks are then concatenated resulting in a tensor of shape, None, 384. Finally we have a dense layer with softmax activation. Figure 7 shows the model architecture.

The model was trained using RMSProp optimizer with a learning rate of 0.0005 and the loss function was categorical cross entropy. The model was trained for 50 epochs and Learning Rate was reduced if the validation accuracy plateaued for at least 10 epochs.

After hyper parameter tuning, the best trained model had an overall accuracy of 44.3% on the test set. Table 3 shows the precision, recall and F1 score of the Parallel CNN-RNN model on the test set. While this model has very similar performance as the baseline model and the CRNN model, class wise F1 scores are quite different. This model performs better than CRNN model for Experimental, Folk, Hip-Hop and Instrumental genres. Given their unique performance, the ensemble of the

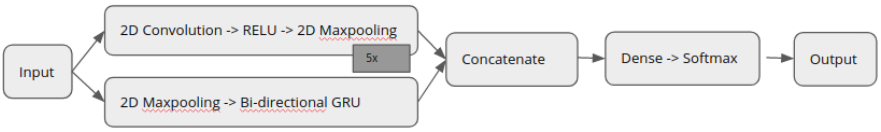

Figure 6: Parallel CNN RNN Model Architecture

	Precision	Recall	F1 score
Electronic	0.59	0.51	0.55
Experimental	0.30	0.30	0.30
Folk	0.24	0.28	0.26
Hip-Hop	0.63	0.80	0.70
Instrumental	0.43	0.41	0.42
International	0.58	0.45	0.51
Pop	0.26	0.19	0.22
Rock	0.49	0.61	0.54

Table 3: Precision, Recall and F1 score of Parallel CNN-RNN model

CRNN and Parallel CNN-RNN should perform better than both.

3.4 Error analysis

I have looked at the confusion matrix to understand errors in more detail. Figure 8 below shows the confusion matrix from the CRNN model. Table 2 shows that the classes with lowest accuracy are Folk, Experimental and Pop.

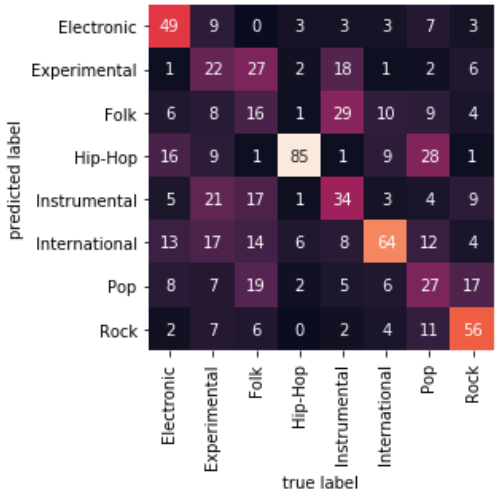

Figure 7: Confusion Matrix - CRNN model

Confusion matrix shows that folk music is often confused with a lot of other classes like Experimental, Instrumental, International, Pop. It might help to have more data for this class as the model is struggling to identify features that uniquely classify folk music.

Experimental is confused with International and Instrumental. Experimental music by definition a bit undefined, something that pushes the boundary. International music may include music from

different genres so this is lack of ability to distinguish between these classes is also somewhat understandable.

Pop is confused with rock and hip-hop. This makes sense as the spectograms for these classes are quite similar and music also tends to be similar.

I tried to find other deep learning models built on this data set to compare my performance to theirs. The closest that I could find was the Genre Recognition Challenge [8] on FMA Medium dataset that measures logloss in classifying up to 16 genres. The top logloss was 1.31 and F1 score was 0.63. In contrast my best model got a logloss of 1.74 and a F1 score of 0.44 but my data set size was only a third of the FMA medium data set.

4 Deeper Understanding of the model

4.1 Visualizing filters learned by the layers

I explored the features learned by initial vs later layers of the convolution model. For this analysis I used the Keras Visualization package [7] and selected Parallel CNN-RNN model as this uses the 2D CNN layers which were easier to visualize.

The first convolution block in this model has 16 filters and the fifth convolution block has 64 filters. To understand what the filter is focusing on, we look at what kind of input maximizes the activations in that filter. Figure 9 below shows the filter activations of all 16 filters in the first convolution blocks vs the first 24 filters of the fifth convolution block.

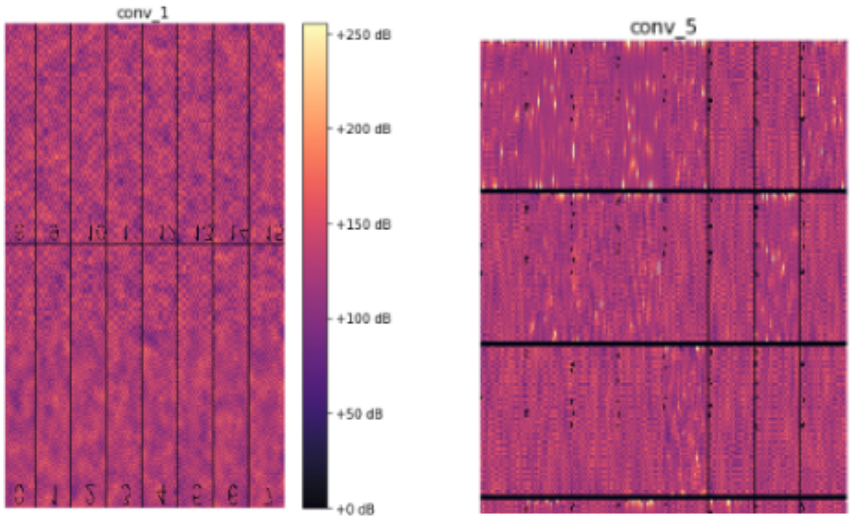

Figure 8: Filter Activations for Conv1 vs Conv5

What I observe here is that the filters for the first layer are relatively straightforward. There are looking at a small kernel size of 3,1. As such they are focusing on different patterns of fluctuations between primarily 50-200 db. In the fifth convolution block, the same filter is looking at a bigger size of the input image due to feature maps being shrunk as a result of convolution and max pooling operations. It is now able to focus on different features like sharp increases in amplitude to 250 db as well as periods of very low amplitude coloured as black region.

4.2 Extending the model to music recommendation

One of the popular applications of music genre classification can be music recommendation. One way of using existing model to do that is to use clustering to identify music groups that are likely to be similar to each other. If a user then likes a few songs from a cluster they have a higher chance of liking other songs from the same cluster.

I approached this task by extracting embeddings from the the first dense layer of the CRNN model that is just before the final layer with softmax activation. This layer has 64 neurons. Clustering analysis was done on the test set with 800 spectograms evenly distributed among the 8 genres.

K-means algorithm was used for clustering. Since I knew there were 8 genres, I set the number of clusters to 8. To evaluate the output of K-means, I looked at two scores - 1. adjusted rand score and 2. Silhouette score.

Adjusted rand score can be calculated in the case when labels are know which is applicable here. The model had an adjusted rand score of 0.22. The rand score is close to 1 for perfect clustering and 0 for random clustering. The rand score for this model makes sense given it has been challenging to get very high accuracy on this data set.

The Silhouette score is indication of whether the clusters are distinct or overlapping. Scores close to 0 indicate overlapping clusters whereas close to 1 indicate distinct clusters. For 8 clusters, the model had a silhouette score of 0.30.

Figure 10 shows the confusion matrix from the clustering. As seen here the model does really well in clustering a few classes like hip hop, electronic and International while there is a lot of confusion among classes like experimental and instrumental.

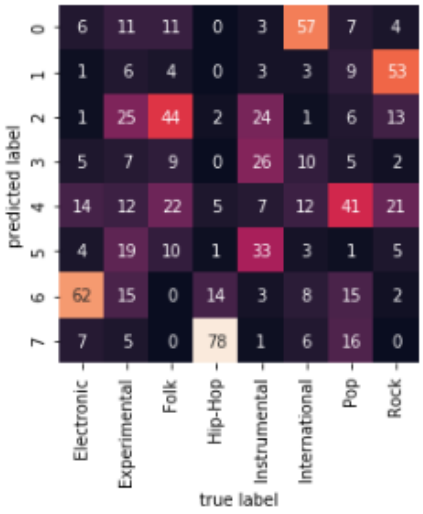

Figure 9: Confusion Matrix - Clustering

5 Conclusion and Future Work

My experiments so far have shown that deep learning models using CNN and RNN can perform as well as the baseline model using MFCC features and SVC. This proves that deep learning can itself extract useful features from raw mel-spectograms. Furthermore while the models have low accuracy, I feel that might be a result of two things - 1. Insufficient sample. Even 1000 spectograms per genre maybe a very small sample here since we are training these models from scratch. A bigger data set should improve results. 2. Challenges with the FMA data set and confusion between classes.Since the top leader board score in genre recognition challenge on FMA Medium data set has only a F1 score of 0.63, this implies that this data set is more challenging than GTZan or Million Songs data set where researchers have attained accuracy exceeding 85 . It is interesting to see that activation visualization does show that the earlier convolution layers focus on features that are more raw which in this case looks like different patterns with smaller variations in decibels. In contrast, the filters in the last convolutional layer focus on features that are more refined and are looking for sharp increases in amplitude which are common in certain genres in lower frequencies. Clustering of embedding gives us a way to extend this model to the task of recommendation if we have a corpus without known labels. For future work, I would like to try the following:

- Further improve the performance of these models by trying different convolutional and recurrent architectures. I am also interested in understanding if transfer learning can be used here by initializing with the weights from another pretrained model like inception or resnet as done in [10].
- Currently I am converting the full 30 seconds of a song to a single spectogram. It is possible that the full 30 seconds is not needed to make a determination of genre but a smaller window like 5 sec or 10 sec would suffice. Certainly humans don't usually take more than 5-10 seconds to determine genre. But this assumes that the song exhibits the characteristics of its label throughout its length. If would be interesting to split each song into smaller windows and create multiple spectograms with the same label and see if this results in a higher accuracy.
- My analysis in section 4.1 shows that filters in different layers are focusing on different features. I would like to develop a better understanding by trying to map these spectograms back to raw audio and see if those filters are audible different.

References

[1] Keunwoo Choi, George Fazekas, Mark Sandler, Kyunghyun Cho. Convolutional Recurrent Neural Networks for Music Classification. ArXiv 1609.04243

[2] Lin Feng, Shenlan Liu, Music Genre Classification with Paralleling Recurrent Convolutional Neural Network, arXiv:1712.08370

[3] GTZan dataset: http://marsyasweb.appspot.com/download/data$_sets/$.

[4] Million Songs dataset: https://labrosa.ee.columbia.edu/millionsong/

[5] Michal Defferrard, Kirell Benzi, Pierre Vandergheynst, Xavier Bresson. FMA: A Dataset For Music Analysis. arXiv:1612.01840

[6] Bob L. Sturm. The GTZAN dataset: Its contents, its faults, their effects on evaluation, and its future use. arXiv:1306.1461v2

[7] Keras Visualization: https://github.com/raghakot/keras-vis

[8] Genre Recognition Leaderboard: https://www.crowdai.org/challenges/www-2018-challenge-learning-to-recognize-musical-genre/leaderboards.

[9] Piotr Kozakowski Bartosz Michalak, Deep Sound Music Genre Recognition: http://deepsound.io/music$_genre_recognition.html$

[10] AD. G. Grzegorz Gwardys. Deep image features in music in-formation retrieval. 2014.