Music-Genre-Classification

Yuchen Zhou,Peng Li

June 17, 2024

Abstract

Audio data is gaining prominence as a crucial information source in the realm of machine learning. The increasing prevalence of voice interactions with intelligent agents such as Siri and Alexa underscores the significance of this data type. A readily accessible wellspring of audio data lies in music records. Over recent years, deep learning has demonstrated notable success in diverse tasks like music genre classification, recommendation systems, and music generation. In this report, I delve into the exploration of two distinct deep learning architectures designed to classify music audio files into eight different genres.

Both architectures leverage a combination of convolutional and recurrent layers to extract features from both the frequency and time dimensions. The report highlights the optimal outcomes achieved by each model. Additionally, an effort is made to cultivate a deeper comprehension of the deep learning models by employing activation visualization techniques. This involves scrutinizing the features that various layers of the convolution model prioritize.

Keywords—Audio data, Machine learning, Deep learning architectures, Music generation.

1 Introduction

Music Genre Classification poses a significant challenge in machine learning, finding applications in various domains. One key application involves assigning genres and sub-genres to each song within an extensive music corpus, facilitating the identification of similar songs. Another practical use is in the realm of music recommendation systems, where leveraging features from an intermediate layer in the model allows the clustering of similar songs, offering personalized recommendations based on user preferences.

Traditionally, the extraction of features from music audio, such as MFCC, has served as the starting point for classification tasks, as presented in my baseline model in section 3. However, a more advanced approach involves allowing a neural network to autonomously identify relevant features. In this project, I adopted the approach of converting each audio music file into a mel spectrogram – a visual representation depicting amplitude across different frequency bands

over time. Spectrograms exhibit distinct visual patterns for different genres, allowing us to treat them as images. These spectrograms then serve as inputs to a convolutional recurrent model, where either parallel or sequential convolutional and recurrent layers extract features for classification.

The report is structured into several sections: Section 1 defines the problem, Section 2 details the dataset and the generation of mel spectrograms, Section 3 explores the deep learning architectures employed and their results, Section 4 conducts a deep dive into the model through activation visualizations and embedding clustering, and the report concludes in Section 5.

1.1 Related work

Considerable efforts have been devoted to employing convolutional, recurrent, a combination of both, and even feed-forward networks for music genre classification. Particularly influential works include Keunwoo et al.'s [1] presentation of a convolutional recurrent model designed to recognize genre, moods, instruments, and era using the Million Songs dataset. Their approach involved a 2D convolution model followed by recurrent layers and fully connected layers for classification tasks. While my CRNN model, detailed in Section 3.2, shares similarities, I opted for 1D convolution layers instead of 2D.

In the realm of 1D convolution models, Piotr Kozakowski and Bartosz Michalak's work on Deep Sound[9]served as inspiration. They applied a 1D convolution model followed by a time-distributed dense layer on the GTZan dataset. While I borrowed the concept of 1D convolution layers from their work, I discovered that for my dataset, incorporating RNN layers after 1D CNN yielded better performance. Lin Feng and Shenlan Liu [2]

introduced a parallel architecture combining CNN and RNN blocks, enabling the RNN layer to operate on raw spectrograms rather than CNN output. My parallel CNN-RNN model drew substantial influence from this paper, and my final architecture closely aligns with theirs, albeit with some modifications to accommodate the nuances of my smaller dataset.

2 Approach

2.1 Choice of Dataset

I encountered three potential datasets suitable for the music genre classification task: GTZan [3], the Million Songs dataset (MSD) [4], and the Free Music Archive (FMA) [5]. The GTZan dataset comprises 1000 songs, each lasting 30 seconds and evenly distributed across 10 genres. However, I opted not to use this dataset due to its limited 100 songs per genre, which I deemed insufficient for a deep learning project. Additionally, concerns raised in a related paper [6] about faults in the dataset, such as repetitions, mislabeling, and distortions, further discouraged its use.

The MSD dataset contains metadata for a million songs, encompassing details on artists and tracks, including track duration, tempo, pitch, and timbre features. Unfortunately, raw audio files for each song, along with genre information, are not readily available. While genre information for a subset of songs could be obtained from Tagtraum, the cumbersome process of downloading each raw audio file from 7digital.com led me to reconsider. Consequently, I decided to leverage the FMA dataset.

The FMA dataset is categorized into small, medium, and large subsets. FMA small comprises 8000 songs equally distributed among eight genres. The dataset is further subdivided into a training set (6400 songs), a validation set (800 songs), and a testing set (800 songs). Each song in the dataset includes a 30-second audio segment, along with metadata related to genre, MFCC, and Chroma features. The eight genres in FMA are Electronic, Experimental, Folk, Hip-Hop, Instrumental, International, Pop, and Rock.

2.2 Mel Spectogram Generation

Each audio file underwent a transformation into a spectrogram, a visual representation illustrating the spectrum of frequencies over time. A standard spectrogram is the squared magnitude of the short-term Fourier transform (STFT) of the audio signal. To make the audio frequencies more perceptible to humans, this regular spectrogram is compressed using the mel scale. The librosa library's built-in function facilitated the direct conversion of the audio file into a mel-spectrogram. Key parameters for this transformation include the window length, indicating the time window for the Fourier Transform (typically set at 2048, equivalent to about 10ms, the shortest period distinguishable by the human ear), and the hop length, representing the number of samples between successive frames (chosen as 512). Additionally, the mel-spectrograms generated by Librosa underwent scaling using a logarithmic function, mapping the sound data to the logarithmic scale used for determining loudness in decibels (dB), relative to human-perceived pitch. Consequently, each audio signal was transformed into a mel-spectrogram with a shape of 640x128.

Figures 1 and 2 depict the mel-spectrograms for the eight genres. Notably, the spectrograms for the different classes exhibit relatively distinct features, suggesting that a CNN can effectively extract unique characteristics from them. While some classes may have similar-looking spectrograms (e.g., Pop in Figure 1, top left; Rock in Figure 2, top left; and Hip Hop in Figure 2, bottom right), the model's confusion between them is addressed later. All audio files underwent spectrogram conversion and were subsequently pickled, significantly expediting the model training and validation processes.

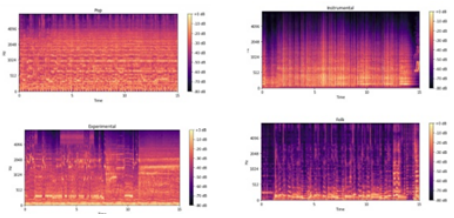

Figure 1: Spectogram - Pop(TL), Instrumental (TR), Experimental (BL) and Folk(BR)

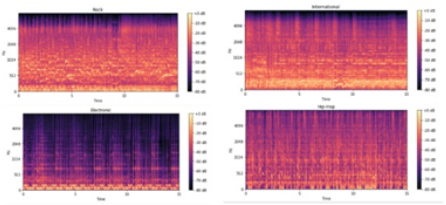

Figure 2: Spectogram - Rock(TL), International (TR), Electronic (BL) and Hip Hop (BR)

3 Modeling Approach and Results

3.1 Baseline Model

I developed a baseline model utilizing the MFCC features available in the FMA dataset. MFCC, short for Mel Frequency Cepstral Coefficients, has demonstrated success in speech recognition, providing 140 features per song in the FMA dataset.

In constructing the baseline model,I applied standardization to the MFCC features by subtracting the mean and scaling to achieve unit variance. Subsequently, I experimented with various classifiers from the sklearn library, including the Decision Tree Classifier, Support Vector Classifier, Random Forest Classifier, and Logistic Regression. The highest performance was achieved using a Support Vector Classifier. The model's test set performance is summarized in Table 1 below, yielding an overall accuracy of 46.3%.

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 theearly layers for understanding the 2D layout of the data. On the other hand RNNs excel in understandsequential 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

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

	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
Interational	0.54	0.48	0.51
Pop	0.34	0.25	0.29
Rock	0.66	0.64	0.65

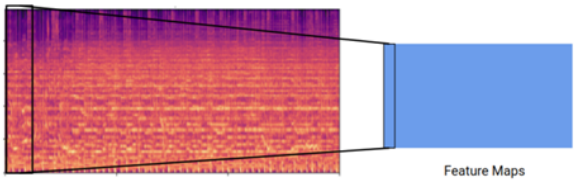

Figure 3: 1D Convolution done by the CRNN model

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 reducesspatial 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.

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.

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

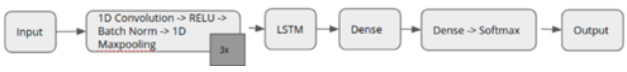

Figure 4: CRNN Model Architecture

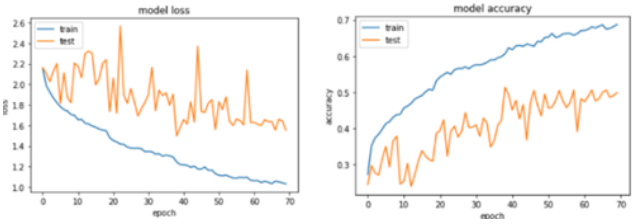

Figure 5: Training and Validation Loss and Accuracy Measures

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

	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
Interational	0.46	0.64	0.55
Pop	0.30	0.27	0.28
Rock	0.64	0.56	0.60

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.

3.3 Parallel CNN RNN Model

Motivated by the concepts presented in [2], I implemented a Parallel CNN-RNN Model. The core idea underlying this network is to address the limitation of CRNN, which relies on RNNs to serve as temporal summarizers but may not preserve the original musical signal's temporal relationships during CNN operations. In this model, the input spectrogram undergoes parallel processing through both CNN and RNN layers. The outputs from these layers are concatenated and passed through a dense layer with softmax activation for classification.

The convolutional block of the model comprises a 2D convolution layer followed by a 2D Max pooling layer. In contrast to the CRNN model, which

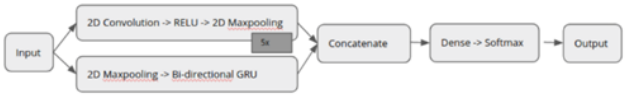

Figure 6: Parallel CNN RNN Model Architecture

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

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

employs 1D convolution and max pooling layers, this model features 5 blocks of Convolution Max pooling layers. All 5 blocks have a kernel size of 3,1. 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, and the final output is flattened, resulting in a tensor of shape None, 256. The recurrent block begins with a 2D max pooling layer with a pool size of 4,2 to reduce the spectrogram's size before LSTM operation, primarily for processing speed. The reduced image is then input to a bidirectional GRU with 64 units, yielding an output tensor of shape None, 128.

The outputs from the convolutional and recurrent blocks are concatenated, resulting in a tensor of shape None, 384. Finally, a dense layer with softmax activation completes the architecture. Figure 6 provides a visual representation of the model.

For training, the model utilized the RMSProp optimizer with a learning rate of 0.0005, and categorical cross-entropy served as the loss function. Training occurred over 50 epochs, with a reduction in learning rate 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 CRNN and Parallel CNN-RNN should perform better than both.

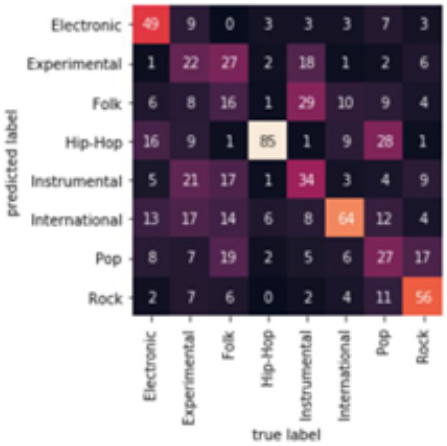

Figure 7: Confusion Matrix - CRNN model

3.4 Error analysis

To gain a more detailed understanding of errors, I examined the confusion matrix generated by the CRNN model, as illustrated in Figure 7. The corresponding accuracy values for each class are outlined in Table 2, highlighting that Folk, Experimental, and Pop are the classes with the lowest accuracy.

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.

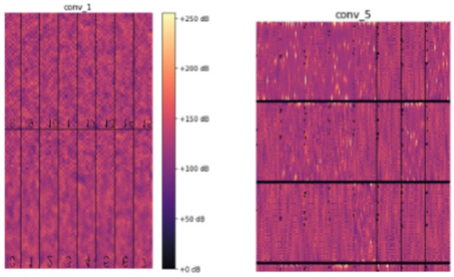

Figure 8: Filter Activations for Conv1 vs Conv5

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 8 below shows the filter activations of all 16 filters in the first convolution blocks vs the first 24 filters of the fifth convolution block. 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

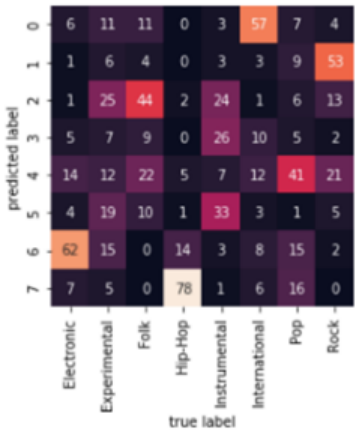

Figure 9: Confusion Matrix - Clustering

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 9 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.

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/datasets/`.

[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:1612.0184.

[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] GenreRecognitionLeaderboard: https://www.crowdai.org/challenges/www-2018-challenge-learning-torecognize-musical-genre/leaderboards.

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

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