Music Genre Classification through Convolutional Neural Networks

Alan Giamatti Steven Smith

December 8, 2020

Abstract

In this project, we aim to use a deep convolutional neural network to tackle music genre classification. We also wanted to compare the performance of a Random Forest Classifier using expert-derived features versus a CNN utilizing raw data and automated feature learning. The experiments will utilize our own newly-created dataset composed of 4,000 thirty-second song samples evenly distributed among 8 genres.

1 Introduction

Music genres are descriptive keywords that aim to identify a musical track's characteristics, such as its instrumentation, rhythmic structure, vocalization, lyrics, and even cultural context. An automated genre classification software has many real-world applications in any music-oriented product as the genre category is crucial to music search and music recommendations.

Previously, automated music genre classification depended on hard-coded feature definitions as defined by domain experts. However, with the rise of machine learning, we can relegate this process to the neural network, feeding it with the raw data and relying on the neural network pipeline with its line transformations, nonlinear activation functions and pooling to automatically learn the features. Here, we perform experiments pitting the older domain-knowledge driven approach to the novel deep learning paradigm. Moreover, we will compare the performance of a convolutional neural network (CNN) fed with raw audio to the performance of hard coded features through a random forest classifier in the task of classifying 8 of the most common genres in Western music.

2 Methodology and Experimental Results

We will detail our process for building the dataset, extracting the features, preprocessing the data and designing the model.

2.1 Genre Selection

We focused solely on Western music. We selected Western classical music as well as 7 additional wide-ranging genres from modern popular music for a total of 8 genres. These include: country, hip-hop (including rap), electronic music (including dance, techno, trance and house), jazz, pop, R&B, rock (including soft, classic and hard rock, metal). While all of these genres are widely popular and highly used categories, there is a great deal of overlap between many of them. For example, songs that can simultaneously be defined as pop and dance, pop and rock, pop and R&B are common. Moreover, pop songs can have rap interludes. As a whole, pop music escapes a clear and unifying musical definition and is particularly prone to overlapping with other genres. We will see how this impacts our CNN's performance in accuracy and recall in the pop category.

2.2 Data Set Building

Though there are some music genre data-sets available, we decided to produce our own for the purposes of this project. Our goal in doing this was to gain experience in building data sets suitable for machine learning which is highly beneficial for the many other projects which don't have established data-sets.
We started with over 30,000 music tracks in various high-quality formats ranging from lossless CD rips (in .alac) to high-bitrate AAC (256 kbps) and MP3 (320 kbps) files. These files were unbalanced in their genre composition. From this general music library, we extracted 6,400 random songs, balanced across all categories. Although the data was previously labelled, we wanted to confirm the genre labelling, so we wrote a small js script that visited Google Music and Spotify APIs to double-check the genre. All songs were then converted to a baseline format. We chose '.ogg' as it is an open source format that is favored by librosa - the python package we will use for feature extraction. The ogg vorbis quality setting was set to '6.0 VBR.'

2.3 Feature Extraction

We used the 'Librosa' Python library to extract the mels spectrogram from the audio file with n_mels set to 128. We used a 22,050 kHz sampling rate, 2048 FFT window, 512 hop length. Additionally, the sound was normalized so our results would be unaffected by loudness difference between the genres. The features were extracted and saved in a csv file. The spectrogram output was pivotal for our CNN as it allowed us to condense raw audio data to something slightly more manageable but without losing much data in the form of compression. Since a spectrogram is a visual representation of sound, we were hoping to leverage CNN's powerful capabilities as image classifiers.

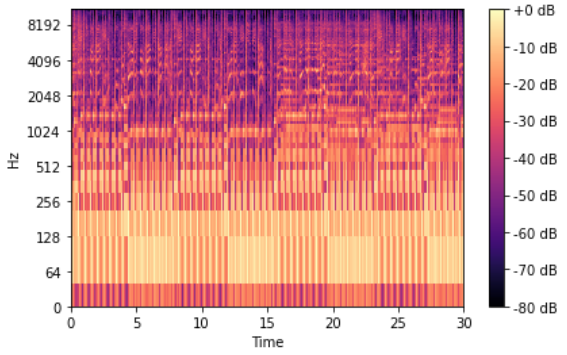

Figure 1: A MELS spectrogram showing a 30s sample of a pop song

As for other sampling settings, we decided on a 30s sampling duration for each song. Our testing showed that 10s to 15s samples resulted in lower accuracy deltas upwards of 10% as well as higher means squared error. Larger samples couldn't be achieved due to hardware limitations. Moreover, since every song is different and some songs can have a very slow or uncharacteristic beginning. In order to avoid sampling these starts, we decided on a 15s minimum offset on each track. However, we also didn't want to simply sample a fixed 15s to 45s interval which would guarantee that we nearly always miss the chorus section. So, we also decided to randomize the offset as integer from 15s up to 120s.

2.3.1 Hardware Limitations

Hardware constraints forced us to not exceed greater than thirty-second samples or 4,000 songs in our data set. Indeed, we needed more than 30GB of RAM to process the samples in csv format, and the highest amount of RAM available to us was 48 GB.

2.4 Data Preprocessing

The genre labels for our data-sets were encoded using sklearn LabelEncoder to transform our string literals label into numerical representation that we could later convert back. This was particularly important for the classification stage as it is much easier for the computer to work with numbers.
The spectrogram data was preprocessed using sklearn Standard Scaler. The reason for this was to standardize the features by removing the mean and scaling it to the unit variance. Generally, preprocessing the data increased the number of epochs required to reduce the loss while training. We also observed higher accuracy scores and lower mean squared error. Random_State = 0 was used when splitting the data-set to ensure that the gains were not entirely due to a different split in the data. And finally, we decided to separate our data in an 80-20 split between the training and testing sets.

2.5 Model Description

The focus of our project was on the development of a convolutional neural network to train on spectrogram images of audio tracks. We chose to implement our model with PyTorch. The goal of the CNN was to accurately classify new images into their proper genre. Therefore, a long time was spent experimenting with the architecture of our network. In the end, the network we settled with had nine layers as shown below:

1. Input Layer: 128 x 1292 (128 mel scales and 1292 time windows representing a 30s sample)
2. Convolutional layer: 32 channels, 3x3 filters
3. Max Pooling layer: 2x4
4. Convolutional layer: 64 channels, 3x3 filters
5. Max pooling layer: 2x4
6. Convolutional layer: 128 channels, 3x3 filters
7. Max pooling layer: 2x4
8. Dense layer: 256 channels, 3x3 filter
9. Output layer: 8 Neurons (8 genres)

Our network follows a clear pattern, every hidden layer alternates between a convolution and a max pooling layer, doubling the neurons on every consecutive convolution. Various configuration, depth

and activation layers were tested but overall this configuration offered better and more consistent results. This change also significantly increased the time required to train the network. Each convolutional layer is followed by a ReLU activation function. Dropout is used to lower over-fitting and flatten is used at the end to make the output layer match our targets shape. Layer 8 is a convolutional network configured such that it would result in a fully connected network, since a Dense layer is not actually available in Pytorch. Our network was influenced by the tests performed by Leland Roberts [2]. Although, by experimenting more, we ended up moving away from his configuration but kept a similar pattern. Softmax was not used as it significantly increased our loss and lowered our accuracy. Our output is manually transformed, fetching the neuron index with the maximum value so normalizing the output with softmax is not required anyways.

2.5.1 Model Hyperparameters

We choose model hyperparameters by looking at the results. In order to avoid both under- and over-fitting, we settled on a 40-55 epoch length. Refer to **Figure 5** in appendix. As for the batch size, our repeated testing showed that using mini-batches (such as batches under 50 would yield more uneven results), and we found the 200 batch size as the optimum medium between larger batch sizes which would limit pure accuracy and smaller mini-batch sizes which would get stuck in local minima quite often.

2.6 Results

To test overall performance of our Convolutional Neural Network we gathered a significant amount of data concerning its performance. Namely, we compute the accuracy score, means squared error and the precision and recall scores on a per class basis. In order to compare our CNN's performance, we built a new data-set using expert-handpicked features extracted from the audio files and ran them in a feed-forward neural network and random forest. We will be comparing the performance of the CNN to only the Random Forest Classifier, as our feed-forward neural network had the worst results.

Figure 2 and **Figure 3** show a confusion matrix and loss vs epoch graph for our CNN using a batch size of 200 and 50 epochs using 3200 training inputs and 800 test inputs. Many tests were made and all had consistent results ranging from 60 - 68% accuracy. From the confusion matrix we see that the classifier has a perfect score classifying classical and does above average for Country, Dance, Hip hop, Jazz and Rock. The classifier struggles a lot with Pop and R&B and this could be attributed to overlapping samples or the closeness of many of our songs in both samples.

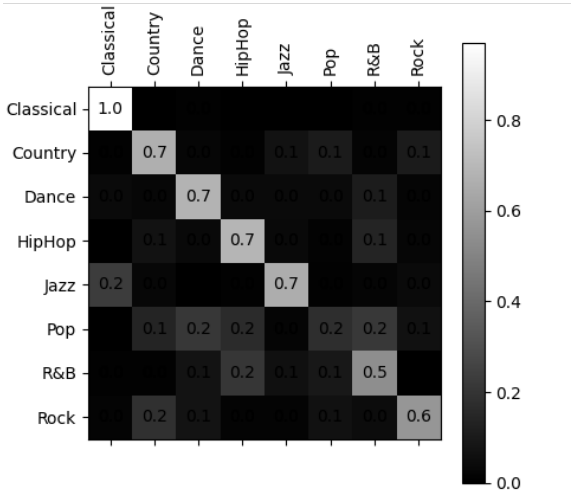

Figure 2: Convolution Neural Network Confusion Matrix batch size 200, 50 epoch

Figure 3 more specifically shows the training results of our network. Loss is subject to variance from run to run, but the figure shows a general depiction of what can happen. We have at times observed better curves that resulted in a higher accuracy. We worked very hard to try and avoid over-fitting the model to the training data. Overall, we are pleased with the result, as the loss continuously decreases with some occasional spikes up that are a characteristic of the stochastic gradient descent method that we employ. Using too many epochs will cause over-fitting and the accuracy of the model and means squared error to worsen.

Table 1 shows the analytic data that was recorded for our full 4,000 data-set (3,200 training, 800 testing). Please refer to the appendix for training data on smaller data-sets. Here we see that the accuracy score of the CNN is 62.50%. Furthermore, if we take a look at the Precision and Recall of Pop, it further reinforces what we saw in **Figure 2**. The neural network struggles to correctly predict the genres of pop songs. R&B also shows significant struggles, but overall still performs decently. The table also reinforces the stellar performance of the classifier for Classical and the good results for Country, Dance, Hip Hop and Jazz. Rock performed slightly better than R&B. **Figure**

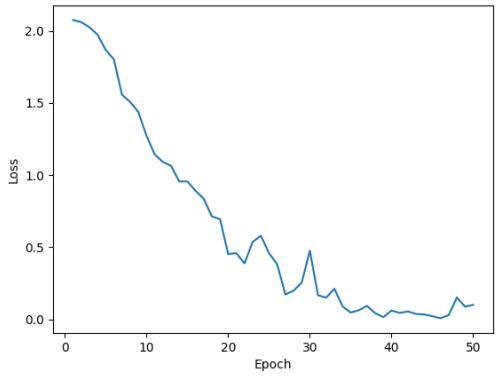

Figure 3: Convolution Neural Network Confusion Matrix batch size 200, 50 epoch

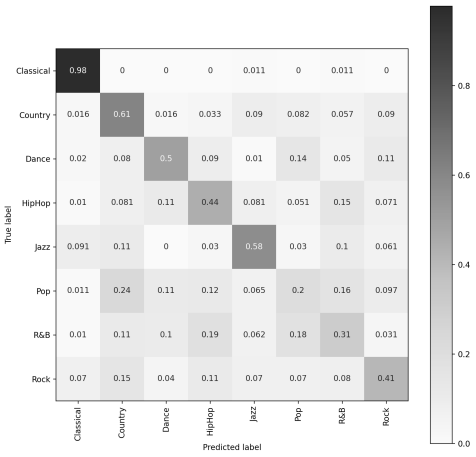

Figure 4: Random Forest Normalized Confusion Matrix (800 Test Input)

2 shows that the classifier sometimes confused rock with country. This is understandable as country can be similar to some style of soft rock.

We have shown and discussed the performance of our Convolutional Neural Network in a fair bit of detail, yet, we have not yet compared it to the Random Forest that we mentioned earlier. **Figure 4** shows the confusion matrix for a random forest classifier using the same tracks as the data-set, but using old fashioned features, like tempo, extracted from the waveform and spectrogram. Refer to the Appendix 4.1: Alternate Feature Extraction for details on the features.

The figure shows an overall weaker performance than our CNN described above. The performance of the random forest can vary a fair bit depending on how the trees are built and where the splits happen. Nonetheless, the RFC never breached the 55% accuracy score threshold. This particular example had an accuracy score of fifty-percent. Since this model was meant for quick comparison purposes we have not collected an in-depth amount of data about its performance. We were only interested to see how the two classifiers performed at a glance.

The confusion matrix shows weaker performance across the board for all classes. However, just like our CNN Pop and R&B performance were very low. This further reinforces the idea that the issue lies either in our data samples or that Pop and R&B categories in our data are inherently harder to classify. Nevertheless, even in these two categories, the CNN managed to perform better. Overall the CNN usually beats the random forest by twenty percent or more. Therefore, we consider the results to be successful at constructing a CNN that can accurately predict many music genres.

Genre	Precision	Recall
Classical	0.7615	0.9701
Country	0.6290	0.6667
Dance	0.6272	0.6832
Hip Hop	0.5825	0.6897
Jazz	0.7223	0.6565
Pop	0.3200	0.17582
R&B	0.4952	0.5417
Rock	0.6932	0.5700

Table 1: Convolutional Neural network precision and recall (800 Test Input).

3 Conclusion

In conclusion, we established that the machine-learning approach to automated feature learning trumps the domain-knowledge approach of features handpicked by domain experts. Furthermore, we believe that we were successful in meeting our goal and we are confident that with more data our neural network would perform even better.

We also ran through the problem that many other data sets exhibit: pop music can heavily overlap with many genres and will continue to be a very difficult music genre to classify. This challenge will not be entirely solved with more data, as music genres, have some degree of intrinsic arbitrariness. Interestingly, removing Pop and R&B from our dataset and running the classifier on the other 6 genres yielded an astounding 79% accuracy. See **Figure 6**.

References

[1] Parul Pandey. Music genre classification with python. https://towardsdatascience.com/music-genre-classification-with-python-c714d032f0d8.

[2] Leland Roberts. Musical genre classification with convolutional neural networks. https://towardsdatascience.com/musical-genre-classification-with-convolutional-neural-networks-ff04f9601a74.

4 Appendix

4.1 Alternate Feature Extraction

During the report we discussed the use of alternate data-set derived from the wave and spectrogram images of our audio tracks. These features were extracted and used for training the random forest. The data was extracted the same way as the spectrogram images using thirty second samples with a random offset. The specific extracted features are [1]:

- Zero Crossing Rate
  - The rate of sign-changes along a signal.
- Spectral Centroid
  - Indicates where the centre of mass for a sound is. It is the weighted mean of the frequencies.
- Spectral Rolloff
  - TMeasure of the shape of the signal.
- Mel-Frequency Cepstral Coefficients
  - A small set of features that describe the shape of a spectral envelope. We use around 20 of them.
- Chroma Frequencies
  - Representation of music spectrum's projected into 12 distinct semitones of musical octave.
- Tempo
  - The speed of the underlying beat

4.2 Alternate Models

Our alternate model is used to compare the performance of our CNN with a more traditional classifier. We used a random forest. The random forest was built using the sklearn library. We did not use any cross validation to determine which hyper-parameters resulted with the best result. As was mentioned previously throughout the report our primary focus was building the best possible Convolutional Neural Network and training using spectrogram images. That process is incredibly time consuming and left little room to do an incredibly in-depth analysis of the random forest performance. The architecture of the Random Forest is shown below:

- Max depth: 26.
- Number of estimator: 100.
- Bootstrap: True.

Note: A basic and generic feed-forward neural network (FFNN) was also built, but little time was invested in maximizing or tweaking it. Time constraints prevented us from trying to build a better FFNN for comparative purposes. Our focus was on the CNN and our extremely long training time on our enormous data set (upwards of 6 hours).

4.3 Optimal Epoch Count

In order to find the epoch count for our CNN, we ran it several times with different values resulting in the figure below. As expected, we saw clear examples of under-fitting and over-fitting in both too low and high numbers. The over-fitting impact on testing accuracy itself was minimal, but there was a clear point where more epochs didn't result in better accuracy.

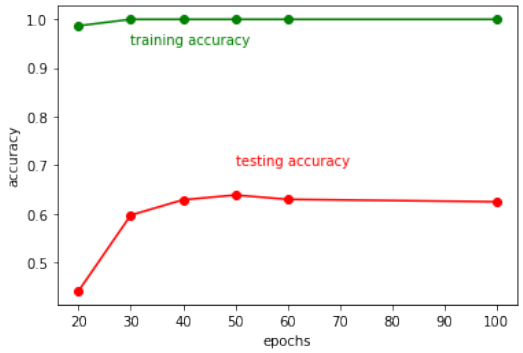

Figure 5: Training and Testing scores across different epoch lengths

4.4 Supplementary Results

In this section we will be providing a list of figures and tables for various other testing scenarios that we have encountered. Some with more or less data samples. Little explanation will be presented. The section is meant to present more evidence and traces of our testing on the CNN and show how the network scales with more data.

Accuracy	MSE
0.6250	4.664

Table 2: Convolutional Neural network Accuracy and Means Squared Error (800 Test Input)

Genre	Precision	Recall
Classical	0.8333	0.8333
Country	0.5357	0.7142
Dance	0.4545	0.0.4761
Hip Hop	0.5185	0.6086
Jazz	0.5384	0.5384
Pop	0.2500	0.1578
R&B	0.2857	0.2500
Rock	0.4583	0.4074

Table 3: Convolutional Neural network precision and recall (158 Test Input 15s samples).

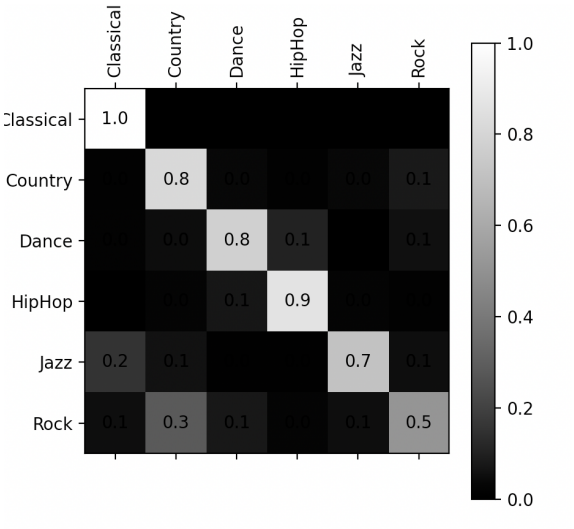

Figure 6: CNN Testing without Pop and R&B 79% Accuracy

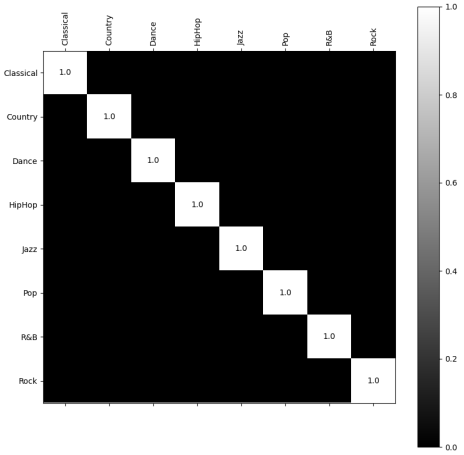

Figure 7: Feed-forward Neural Network over-fitting Confusion Matrix example on training data

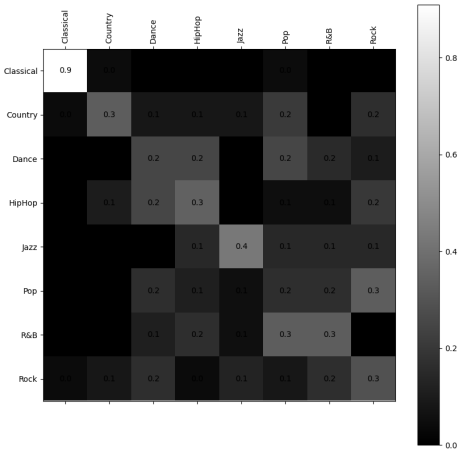

Figure 8: Feed-forward Neural Network Confusion Matrix example on 160 test sample

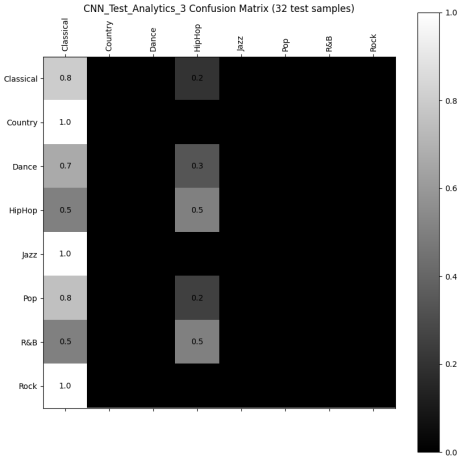

Figure 9: CNN insufficient epoch or unbalanced training set causes local minima

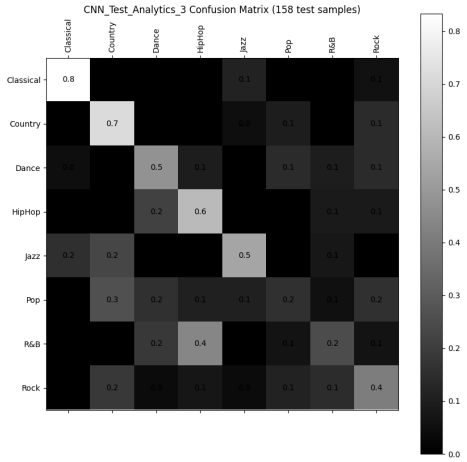

Figure 10: CNN 15s 158 Sample Confusion Matrix

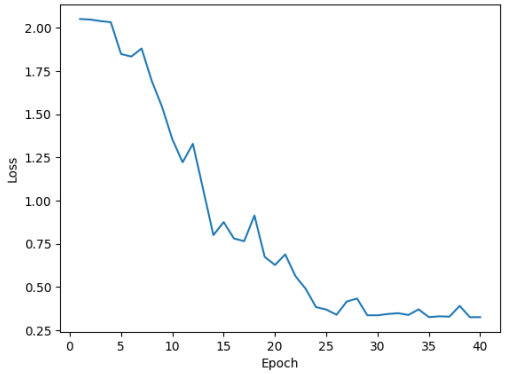

Figure 11: CNN 30sec, 4,000 samples, 32 batch size, loss function demonstrating the difficulty of reaching a low loss even after 40 epochs

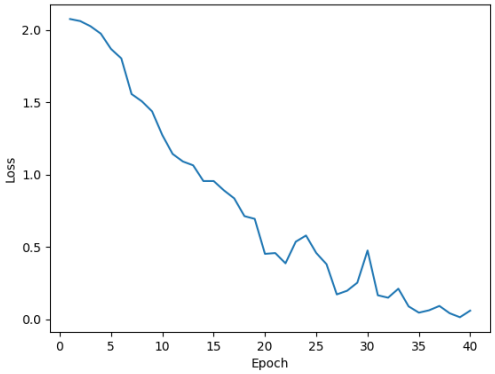

Figure 12: CNN 30sec, 4,000 samples, 200 batch size, loss function demonstrating that after 40 epochs, a normal batch size will get to much lower loss

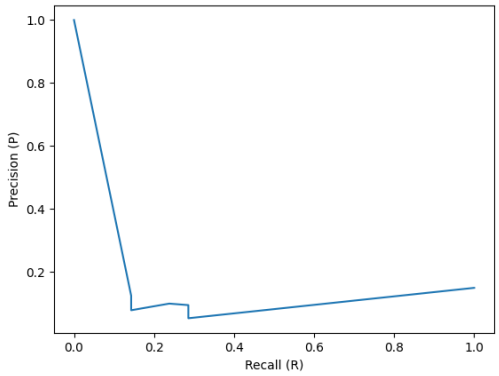

Figure 13: CNN Testing Recall vs Precision example on 160 samples