CNN Genre Music Classifier

Julian Heden
pjheden@kth.se
Theo Royer
royer@kth.se

Abstract

In this report, we present an approach for music genre classification using deep neural networks. Tests were carried out using similar structures found in related papers using the GTZAN dataset. LMFCC spectograms were used as an input for the networks. Results showed that the network failed to do the classification on 10 genres but it gave positive results when using less genres. Future improvements could be made by using larger spectograms or trying different network architectures.

1 Introduction

There are more and more papers about music and deep learning but it is still not used for music recommendation. Today, music recommendation systems work like any other recommendation system. They use collaborative filtering and look at the similarities between users to tell which song track they would probably listen to if they were listening to another song track. The benefit of this method is that you don't need to know anything about the song track itself so it is not required to analyze the sound signal. Each song is defined by latent factors from the recommendation model instead of the sound frames. This make the recommendation systems quite independent of what kind of thing we want to recommend. But those who have already used Spotify may have noticed that when listening to a generated playlist based on recommendations, the playlist contains a lot of very well known songs. If you wanted to discover new artists, you would be a bit disappointed. Even sometimes, it does not really fit the music genre you expected. The reason of this problem is that the recommendation systems can only recommend songs other people listen to, which are most of the time famous songs. It cannot recommend new songs.

One article we found was about using a DNN to generate the latent factor for recommendation using spectograms. [1] The inputs were mel-spectograms of size 599x128 (frames x frequency) from 30s excerpts extracted from the middle of the 1 million most popular tracks on Spotify. The architecture of the network was made of 3 convolutional layers with max-pooling operations in between, a global temporal pooling layer (mean,max,L2) and 2 fully-connected layers. The amount of data they had access to was quite significant as they used to it to generate very specific genre playlists from scratch. Their results are quite convincing and if it was to be implemented, then recommendation playlists could be created using songs that were almost never listened by users. Our case is of course much simple because we only predict the genre of a song instead of a lot of latent factors but we used a similar structure for our network.

We also read papers more explicitly about music genre classification. In [2], M. Flores used a similar structure from the Spotify DNN with 3 convolutional layers with max-pooling operations in between, a global temporal pooling layer (mean,max) and 2 fully-connected layers. He used the GTZAN dataset to make the classification. The input used was MFCC generated from mel-spectograms with 128 bins. However, the size of the MFCC input is not specified. What is special about the

Submitted to 30th Conference on Neural Information Processing Systems (NIPS 2016). Do not distribute.

convolution layers used is that they use a 2d kernel moving in only one direction. Each filter covers the entire frequency dimension and moves along the frames direction. Filters are rectangular and their size is typically 4x128 (frames x frequency/filters). From the same paper, it also seems that more conventional kernels can be used and give good results. This paper reports a final accuracy of 58.7% on validation data even though it also reports a lower accuracy on training data (56.1%) and their plots show a consistent lower validation loss than the training loss at the beginning of the training.

The dataset used in this project is the GTZAN dataset. It consists in 1000 30s excerpts from 10 differents music genres. It was used in the very well-known paper "Musical genre classification of audio signals" [3] in 2002. It is a standard and simple dataset so it suits our needs. But there are multiples issues about this dataset that are exposed in [4] because 7% of the excerpts come from the same recording, 10% are mislabelled and the artists distribution inside the categories can be quite bad. For instance, 24 of the 100 pop songs are from Britney Spears and 34 of the 100 reggae songs are from Bob Marley. The risk for genre classification is that it overfits the characteristics of one specific artist instead of the whole genre. Using t-SNE to plot a representation of the data distribution, another paper showed that it might be complicated to use more than 4 classes on the GTZAN for song classification [5]. It seems that the "Million Song Dataset" [6] might be a more appropriate dataset to use in music classification now but it would have been too complicated to use it with the time constraints we had.

2 Method

The GTZAN datasets consists in 1000 30s excerpts from the middle of songs equally distributed in 10 music genres (blues, classical, country, disco, hip hop, jazz, metal, pop, reggae, rock). The tracks are all mono 16-bit .au files sampled at 22050Hz. We computed liftered MFCC for each of these 1000 tracks with a frame size of 2048 samples and an overlap of 1024 samples. Mel-spectograms were generated with 40 frequency bins and the LMFCC were then generated with 13 frequency bins. We kept the first 300 of the 645 frames because of computations issues, it corresponds to 14s for each song which should be more than enough for this classification task. We had issues computing the LMFCC for 10 of the hip-hop tracks, so we removed them from the dataset.

For the data split, all the tracks were shuffled within their genre category to make sure, for instance, that an over-represented artist (ex: Bob Marley) is not in only one of the sets. After that, the first 70% of each genre and sent to the training set, the next 10% to the validation set and the remaining 20% to the test set. Then again, the tracks were shuffled again within each set. The split for the datasets is the following :

- Training set : 693 tracks (70 tracks of each genre except hip hop with 63 tracks)
- Validation set : 99 tracks (10 tracks of each genre except hip hop with 9 tracks)
- Training set : 198 tracks (20 tracks of each genre except hip hop with 18 tracks)

Originally, we didn't normalize the data but we later implemented it. We normalized the data over the whole dataset before splitting. Each of the 13 MFCC is zero mean and unit variance.

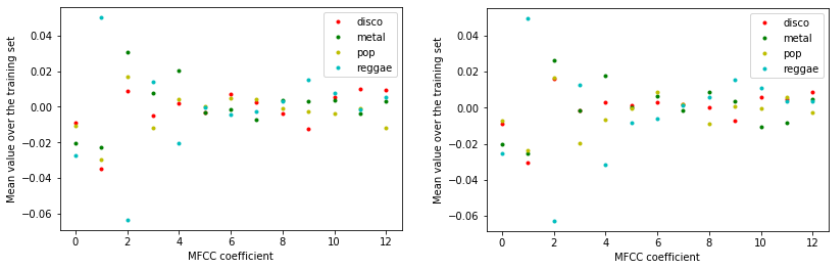

Figure 1: Mean of the MFCC for each genre on 2 different splits of the training set

From figure 1, we see that the normalized LMFCC are able to capture the differences between the different genres. The right side graph look similar which means the dataset is big enough to show the characteristics of each genre. But because we can still see differences between them, it means the split of the data might have a significant effect on the results because the dataset is not that big.

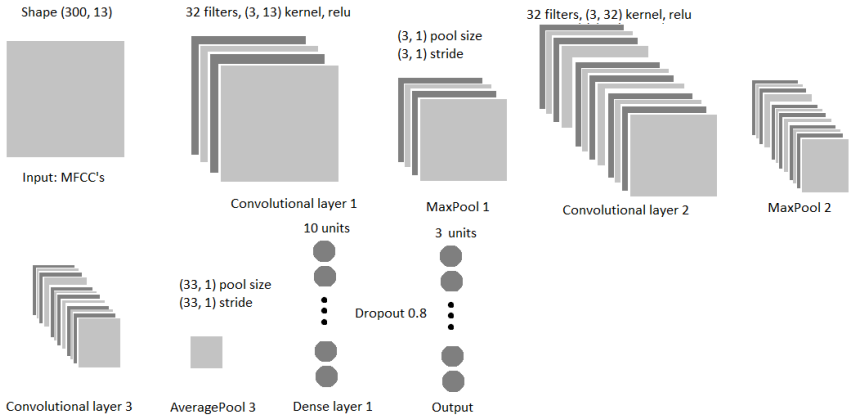

Figure 2: DNN structure.

The network structure we used for our experiments is seen in figure 2. Here we have 3 Convolution layers, followed by a Dense layer with dropout and finially the output layer which correspond to the number of classes. The average pooling layer between the last convolutional layer and the dense layer computes the mean over time for each filter.

Each convolutional layer has 32 filters and uses ReLU as their activation function. The first convolutional layer uses a stride and kernel of size (3, 13). This is so that we make use of all the frequencies over a few frames. The two first layers has a corresponding MaxPooling layer, the last one uses an AveragePooling layer. The idea behind averaging is that the location of the features are not important, which is the opposite when it comes to pictures.

The final dense layer has 10 hidden units, and uses a dropout of 80% as a regularization tool. The choice of our dropout rate came from inspiration from a similar project who found success[1], normally a rate of 50% is used. Finally we output the probability for each of the three classes with softmax. The network is similar to [1] and [2] in the overall structure but we used less layers and filters than in these papers.

3 Experiment

In this part we will discuss intermediate experiments and decision we took that led to the final results presented in the next section.

To begin with we tried running for short period of times with many different structures and parameter. In hopes of finding what worked and what did not. From testing together with inspiration from [1] we ran our experiments on the network mentioned in section 2.

First we built our CNN to take spectograms as an input. After running for many epochs, it was eminent that the validation accuracy would not go much higher than 13%. It was clear that our network could not learn the underlying patters of the spectograms. Instead of altering the network, to try to achieve this, we inspected the input data. For the human eye, they are difficult to distinguish, which is also the case for a network training to learn them.

If the magnitude of the input-data varies a lot, it is helpful to normalize the data. This makes the input comparable to each other. It was also shown that the distribution of classes in the training data was important, i.e., how many samples of each class we trained over should be equally distributed. If one class was dominant, the network quickly converged to only guessing that class.

Pre-processing the data and calculating the Liftered MFCC's, is more suitable as an input for our CNN. It makes it easier for our network to learn. The result from this was not convincing. As seen in figure 4, the validation loss is constantly increasing. As mentioned in the chapter 1, 10 classes are difficult to learn and by looking at the confusion matrix we can confirm that our network have not learned anything at all, see figure 3.

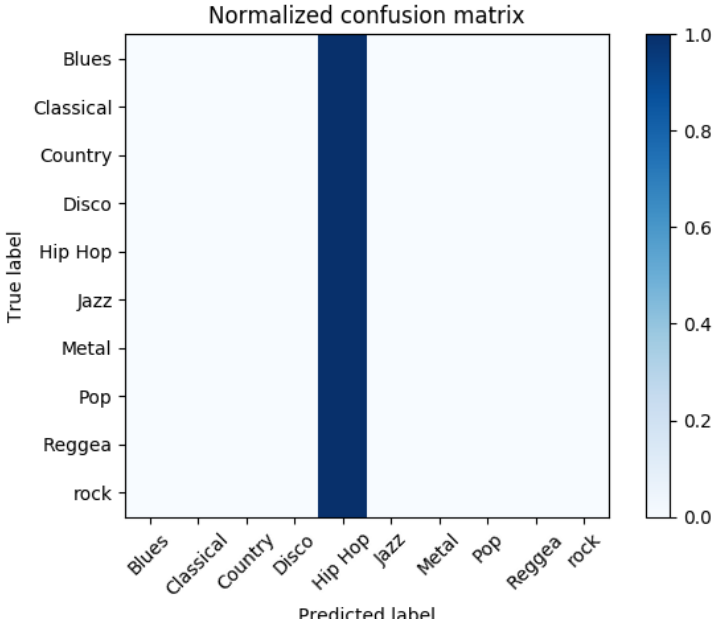

Figure 3: Prediction accuracy over the classes.

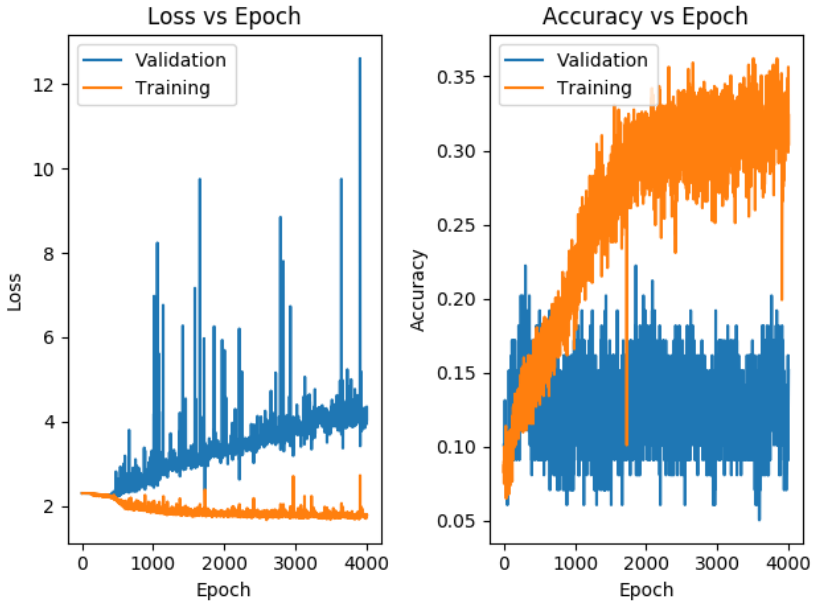

Figure 4: Loss and Accuracy evolution for CNN with 10 classes.

To experiment with the limit of the network structure, the problem was reduced to a subset of only 5 classes. Keeping the categories jazz, disco, metal, pop and reggae. Here the network converge after

only a few epochs (10), and show some strange behaviour where validation accuracy begins in and stays consistently at 20%. Considering these intermediate results, we decided to present the final results for only 3 classes as it seems the network fails to learn more classes.

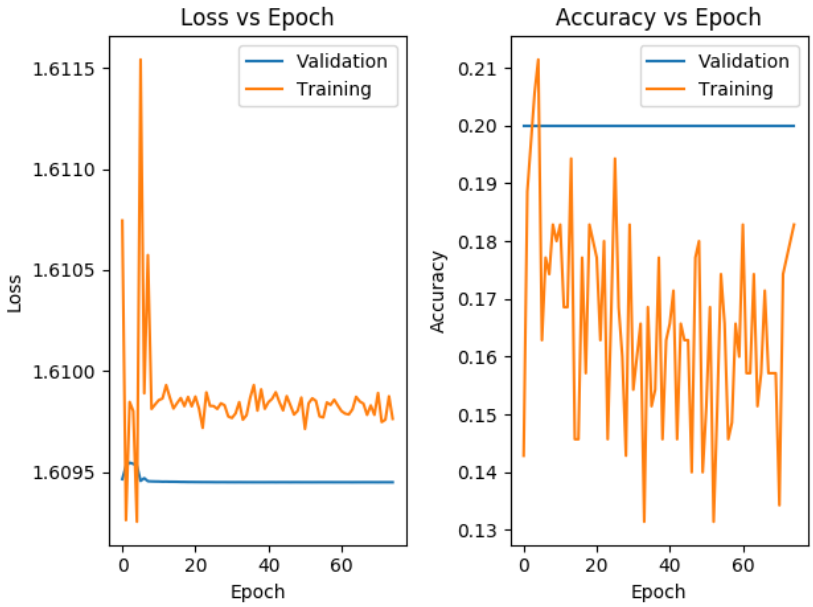

Figure 5: Loss and Accuracy evolution for DNN with 5 classes.

4 Results

By again reducing the problem, and now only trying to predict 3 classes (Metal, Classical and Disco) using LMFCC we get some promising results. The network is learning the underlying structure, and is managing to distinguish between the 3 different classes. Converging after around 1000 epochs, as seen in figure 6. Here we obtain an accuracy for around 60% for the validation set and a corresponding accuracy of 48.33% on the test set.

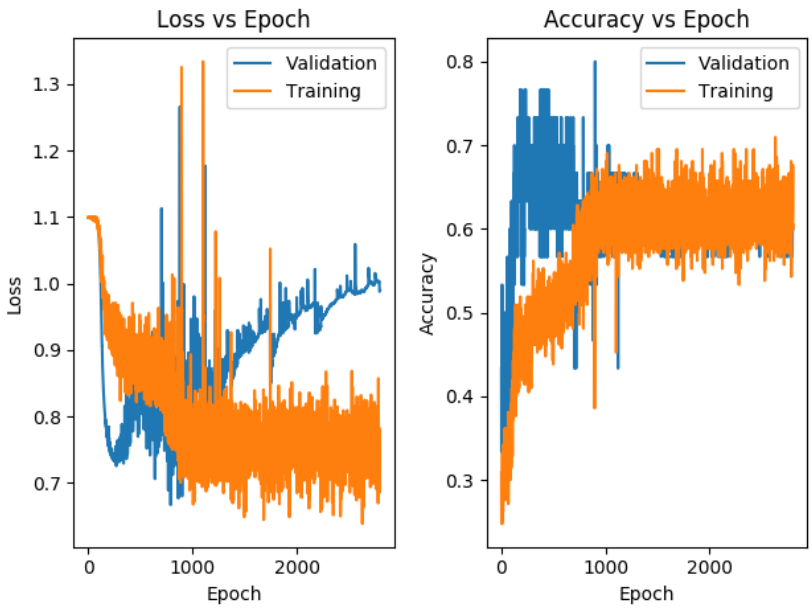

Figure 6: Loss and Accuracy evolution for CNN with 3 classes.

The confusion matrix 7 and table 1 shows variation throughout the classes. Metal is more easily predicted with the highest scores in Precision and F1, while Classical and Disco shows to be less accurate. Giving the idea that some genres could be simpler classify than others.

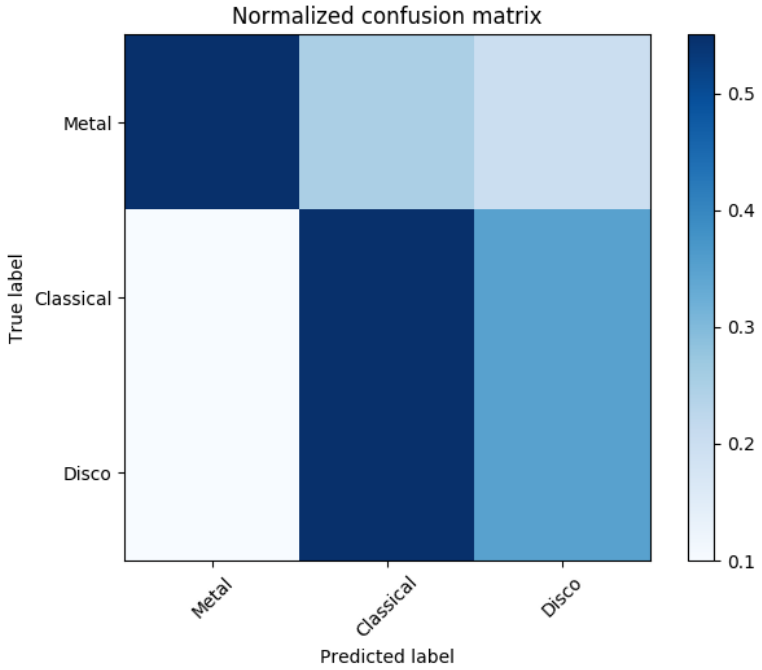

Figure 7: Prediction accuracy over 3 classes.

Table 1: Metrics for 3-class CNN.

Class	Precision	Recall	F1-score	Support
0	0.73	0.55	0.63	20
1	0.41	0.55	0.47	20
2	0.39	0.35	0.37	20
avg / total	0.51	0.48	0.49	60

5 Discussions and Conclusions

The results from our different experiments varied a lot. The CNN's for more than 3 classes had next to no success. As it has been seen done in other papers, then we can think of a few reasons for this.

In comparison to the paper [2], we used less frequency bins for the spectograms. In other words, limiting the information we feed as an input to the network. For instance, using more frequency bins would have helped to capture the harmonic structure or even chords of the songs.

Another change that might have impacted our results or the input format. We used LMFCC's because we believed them to work better compared to the mel-spectograms, but this could have been the result of other factors (such as, normalization, data distribution and network structure). Both have been seen used for classification with good results. A welcomed property of using mel-spectograms is that you can look at the filters to interpret them in the frequency domain and see the frequencies that activate best in each filter.

The network structure itself might be the underlying issues for learning more than 3 classes. There are many parameters to experiment with here, and that is not an easy task to do. We could have changed the number of convolutional and dense layers, add the L2-, Max-, and Mean- layer, and even increased the number of filters or hidden nodes.

The dataset used for this experiment is somewhat small when it comes to deep learning. With the training set being less that 1000 samples, makes the training challenging. A low amount of samples can cause difficulty in learning the underlying structure, makes it prone to noise and increases the chances of over-fitting. It would be interesting to see how increasing the size of the data, could effect the result. For example, one could use a large and freely accessible dataset Million Song Dataset [6]. It has 280 GB worth of samples.

A possibility to test in the future, would be to try and create multiple CNN's that work together. If the 10 classes (or more) are divided into larger groups at first, based on similarities. Then we might be able to narrow it down, so we can eventually make use our 3-class CNN and get the specific answer. A visual representation to help explain is shown in figure 8

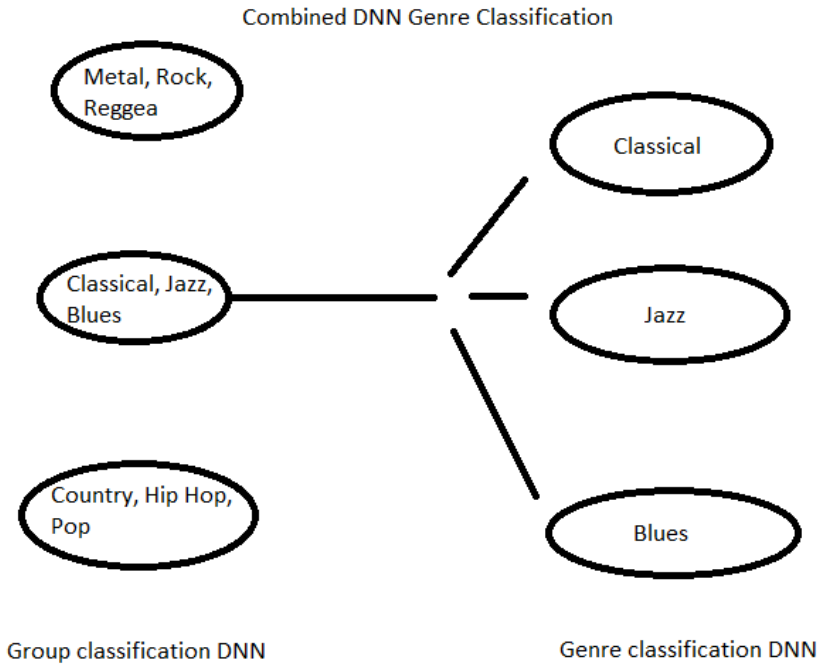

Figure 8: Combine CNN's to classify more than 10 genres.

References

[1] http://benanne.github.io/2014/08/05/spotify-cnns.html : Spotify

[2] Miguel Flores Ruiz de Eguino: *Deep Music Genre*

[3] Tzanetakis G., Essl G., Cook P.: *Automatic Musical Genre Classification Of Audio Signals*

[4] https://arxiv.org/pdf/1306.1461.pdf : critics of GTZAN

[5] Tao Feng: *Deep learning for music genre classification*

[6] https://labrosa.ee.columbia.edu/millionsong/ : Million Song Dataset

6 Incorporation of the suggestions

The suggestions from the peer review were mostly about the language and structure of the report. We believed the language suggestions were all relevant so we fixed the sentences in the report that they thought were not correct.

We also had some remarks about the structure of the report and especially about the confusion between the experiments and results sections. What we wanted to do in the experiments section was to explain and show how we come to the final experiment whose results are shown in the next section. For instance we explain that we got better accuracy results when using LMFCC than mel-spectograms in this section to justify that we will use LMFCC for the final results. We also show that we had to reduce the number of classes to get decent results in this section.

Our idea for the separation between experiments and results was to keep a chronological coherence in the way we explain what we did, but it seems that it is a bit uncommon based on the peer review comments. They thought that it would be better to show the intermediate results in the results section. We think this structure would make our report a bit more easy to follow but it would be hard to explain the reason we decided to try the different experiments in the experiments section without talking about intermediate results. So we made a little compromise by adding some sentences in the experiments to make everything a bit more clear.