An Application of Convolutional Neural Networks to Music Genre Classification

Benjamin Steiner

Olly Mapps

A report submitted to the Department of Statistics of the London School of Economics and Political Science

April 14, 2023

Abstract

In this report, we explore the GTZAN dataset for music genre classification. Our dataset is split into 10 genres of music and we will try to learn the differences between them. For each audio file, we are also provided with an image and metadata. We are therefore able to compare the performance of various neural network architectures such as the CNN, RNN, and also MLP.

A focus of this report is also on the ensemble methods described in Section 2.4 which allow us to use various types of data in a single model. We postulate that by using the various forms of the data we can obtain a better model than the baseline ones.

We found that the XYZ model performed best, attaining a test accuracy of X%. In contrast, the XYZ model performed worst, with an accuracy of Y%. This is likely because...

Contents

Chapter 1

Introduction

1.1 The GTZAN Dataset

Our dataset contains audio recordings of 1000 songs, each 30 seconds long and grouped evenly into 10 genres: Blues, Classical, Country, Disco, Hiphop, Jazz, Metal, Pop, Reggae, and Rock. For each song, there is a Mel-spectrogram and also associated metadata. Each instance of metadata contains auditory characteristics of the recording, such as the tempo. In contrast, the Mel spectrogram is a way of visualising what frequencies are more intense on the melodic scale at a given time.

1.2 Problem Setting

Given an audio file, we aim to predict the genre of the music. The novelty of our problem arises from the fact that there are no well-established neural network architectures to deal with audio signals. Therefore, we must first transform our data into a format that is interpretable by our neural network. We achieve this by a natural application of signal processing using spectograms. Spectograms are a visual representation of the intensity of frequencies as time varies. Now that our data has been converted to an image, we can apply the standard Convolutional Neural Network (CNN) to learn features of our data. While these differences are hard to quantify, we show in Figure 1.1 that there are noticeable differences between genres.

1.3 Model Selection

We first replicate two models used in Pun and Nazirkhanova (2021): the Multi-Layer Perceptron (MLP), and the CNN-MLP ensemble. Due to the time series nature of our data, we also examine the applicability of the Recurrent Neural Network (RNN). We consider a sole RNN, a CNN whose output is fed into an RNN, and finally a CNN-RNN ensemble model. An important part of this report is the exploration of

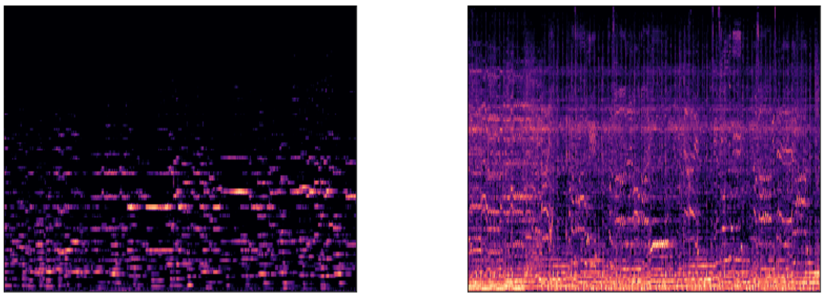

Figure 1.1: Left: A spectogram from a piece of classical music. Right: A spectogram from heavy metal music. We observe that the heavy metal spectogram is far busier than the classical spectogram.

ensemble methods. We hypothesise that by combining two baseline models, we are able to exploit the various features from each model to obtain a better overall model.

Chapter 2

Methodology

2.1 Data Transformation

As mentioned earlier, we are unable to directly use the audio files as input to our CNN. However, we are able to obtain useful Mel-spectograms via a Fourier transformation, but we are graciously already provided with these images. We convert these images into grayscale as we care solely about the intensity of each pixel rather than the actual colour. We visualise an example grayscale image in Figure 2.1:

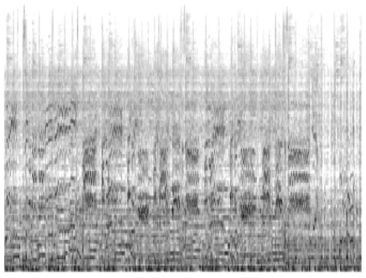

Figure 2.1: An example grayscale Mel spectogram.

For our metadata, we standardise our columns to follow a *Normal(0, 1)* distribution. We do this due to the large differences in the order of magnitude between columns. For example, the variable rms_var has order of magnitude of 10^{-3} whereas spectral_centroid_var has an order of magnitude of 10^5.

2.2 Data Augmentation

An immediate limitation we encountered was the relatively small number of audio files per genre (100 files for each of the 10 genres). In order to improve the performance of our models, we will first augment our data by splitting each spectogram vertically into 5 evenly-sized images. This corresponds to splitting each 30-second audio file into 5 6-second audio files. We claim that each instance of our new dataset should still be representative of the original image it came from.

The reader may wonder whether it is appropriate to split time series data in this way. We argue that, while a valid concern, each smaller image will still be highly correlated with the other splits from the same image. Therefore, we will not be losing too much information, and so the overall increase in data we now have should outweigh the informtaion loss. *Ceteris paribus*, for the CNN model we found that the augmented dataset attained a test accuracy of X%, whereas the original data only achieved an accuracy of Y%, backing up our previous claim.

2.3 Baseline Methods

In this section, we describe the models we consider to be our baseline. This achieves two goals: firstly, for comparison to the more complicated models later on; secondly, a more subtle point is that we can perform exploratory analysis of these models by examining questions such as "do more convolutional layers improve performance?" or "does dropout improve performance?". While the exact answers may not be appropriate for the other models, we can generalise to help us later on.

2.3.1 CNN

As a baseline model, we implement a standard CNN with 5 convolutional layers and 2 Fully Connected (FC) layers. Using a kernel size of $3x3$ with a stride length of 1 for Convolutional layers and a stride of 3 for the Max Pooling layers. In Table 2.1 we label this is as $(1, 3)$. Between each layer we use a combination of ReLU activation functions and connect these to the Max Pooling layers. After flattening to a $1x512$ tensor, we feed this through our FC layers with 64 and then 10 neurons.

While tuning the hyperparameters, we observed the following: that the of use dropout was necessary to prevent some of the overfitting. We also saw that a small number of FC layers improved our results. Furthermore, we found that a small stride length and kernel size performed well. These seem like reasonable observations because the regularisation through dropout and ReLU activation can reduce the training time even though we use a small kernel size and stride. In context, this means that with our relatively small amount of data we can perform more complex calculations for each image, while still retaining a reasonable overall training time.

We also examined the methodologies used in Choi et al. (2017) which compare kernel sizes. In particular, we analyse the differences between using a row vector or column vector as our kernel. The consequence of this is that we are, in essence, collapsing one axis which will help us determine whether there is more information stored in the frequency or time axis of the spectogram. We found that .... performed marginally better.

2.3.2 Metadata MLP

We are given two datasets of metadata: one for the 30-second audio files, and another for 3-second audio files which have been split from the original files. For the 30-second metadata, we used an MLP with 4 FC layers, with 57,32,16, and 10 neurons in each layer. We used the ReLU activation function and a small amount of dropout ($p = 0.25$ in two of the layers).

2.3.3 RNN

Since our data has a time series aspect to it, we now consider the effects of time. As a baseline, we first considered a simple, single-layer RNN. However, this did not achieve sufficient performance to reasonably consider as a baseline. To improve performance, we implemented a bi-directional RNN with Gated Recurrent Activation Units (GRUs). The benefit of using GRUs is two-fold:

2.4 Ensemble Methods

As briefly mentioned earlier, we will consider the effects of combining multiple models together to create a new architecture. With the various data types, we can use appropriate baseline architectures and then merge these outputs together before feeding this through the final FC layers.

2.4.1 CNN-MLP

By combining the image data and metadata, we now train a CNN-MLP. Each instance of data will now consist of the Mel-spectogram and the associated metadata. We can think of our network as two networks which we connect after some training. We input the image to convolutional layers, and the metadata to a separate MLP. These networks are connected after the final convolutional layer and FC layer. We then feed the concatenated data through FC layers.

2.4.2 RNN-CNN

2.5 Summary

In Table 2.1 we briefly summarise the models we used. Recall that a stride of $(1, 3)$ would represent a stride of 1 in the convolutional layer and a stride of 3 in the pooling layer.

	CNN	MLP	RNN	CNN-MLP	RNN-CNN
FC Layers	2	4	x	x	x
Convolutional Layers	5	N/A	N/A	x	x
Dropout	Very Low	Medium	medium	high	None
Stride	(1,3)	N/A	N/A	1	2
Kernel	$3x3$	N/A	N/A	3x3	2x2

Table 2.1: An overview of the models we used.

Chapter 3

Numerical Evaluation

3.1 Baseline Methods

For our baseline CNN, we achieved a test accuracy of 59%, as shown in Figure 3.1. We do observe a fair amount of overfitting after 25 epochs, but the test accuracy is relatively flat after that point whereas the training accuracy continues to increase. We also find that using many convolutional layers performed better, but that many FC layers actually decreased the accuracy.Finally, we opted for Max Pooling layers over Average Pooling in order to retain some of the dimensions and prevent our images c]becoming over-simplified.

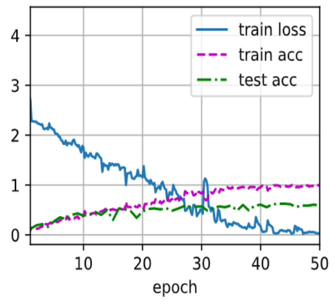

Figure 3.1: The training results of our CNN.

Our MLP exhibited a few curious properties, in particular that our test accuracy hardly changed depending on whether we used 2, 3, or 4 FC layers. In fact, adding too many FC layers decreased accuracy drastically. Another parameter to point out is the learning rate, which is bigger than we would typically see, but it performs much better than a learning rate of 0.01 or smaller. Once again, as long as the order of magnitude was correct, we observe only a marginal difference. For example, using a learning rate $0.1 \leq \eta \leq 0.5$ yields roughly the same result. We show the learning process below in Figure 3.2:

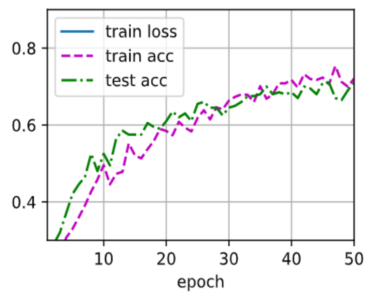

Figure 3.2: The training results of our MLP.

3.2 Ensemble Methods

3.3 Summary

In Table 3.1 we summarise our results:

	CNN	MLP	RNN	CNN-MLP	RNN-CNN
Test Accuracy (%)	59	> 70	x	x	x
Train Accuracy (%)	99	> 70	x	x	x
Batch Size	30	50	N/A	x	x
Learning Rate	0.01	0.1	N/A	x	x
Epochs	50	50	N/A	x	x

Table 3.1: A summary of our results.

Chapter 4

Conclusion

4.1 Findings

4.2 Limitations

There are a few obvious limitations to our analysis, some of which stem directly from the dataset. Firstly, the dataset is relatively small for a complicated task such as this one. Secondly, the subjective nature of the labels - how sure can we be that there is a significant difference between Pop and Hiphop, or Jazz and Blues? There are also some genres which use music from the same composer - reducing the within-genre variation to help us learn the differences. For a more comprehensive overview, we refer the reader to Sturm (2013).

4.3 Further Work

4.4 Team Contributions

We both agree that this project was 50% effort from both of us.

Bibliography

Choi et al. (2017). Convolutional recurrent neural networks for music classification. In: *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2392–2396. IEEE.

Pun, A. and K. Nazirkhanova (2021). Music Genre Classification with Mel Spectograms and CNN .

Sturm, B. L. (2013). The gtzan dataset: Its contents, its faults, their effects on evaluation, and its future use. *arXiv preprint arXiv:1306.1461* .

Appendix A

The first appendix