5-2018

# Music Genre Classification With Neural Networks: An Examination Of Several Impactful Variables

Jingqing Yang
*Trinity University*, 838544531@qq.com

# Music Genre Classification With Neural Networks: An Examination Of Several Impactful Variables

Jingqing Yang

A DEPARTMENT HONORS THESIS SUBMITTED TO THE
DEPARTMENT OF COMPUTER SCIENCE AT TRINITY UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR GRADUATION WITH
DEPARTMENTAL HONORS

DATE ___04/13/2018_______________

| | |
|---|---|
| Matthew Hibbs_____________ | Yu Zhang__________________________ |
| THESIS ADVISOR | DEPARTMENT CHAIR |

_______________________________________
Michael Soto, AVPAA

# Music Genre Classification with Neural Networks: An Examination of Several Impactful Variables

Jingqing Yang

## Abstract

There have been several attempts to classify music with content-based machine learning approaches. Most of these projects followed a similar procedure with a Deep Belief Network. In this project, we examined the performance of convolutional neural networks (CNN) and recurrent neural networks (RNN) as well as other components of a classification architecture, such as the choice of dataset, pre-processing techniques, and the sample size. Under a controlled environment, we discovered that the most successful architecture was a Mel-spectrogram combined with a CNN. Although our results fell behind the state-of-the-art performance, we outperform other music classification studies that use a CNN by a large margin. By performing binary classification, we also discovered individuality across genres that caused inconsistent performance.

## Acknowledgments

I would like to thank Dr. Matthew Hibbs for his instruction and support throughout the project. I would also like to thank the Trinity University Computer Science department, a close-knit community where professors not only offer knowledge, but also friendship.

# Music Genre Classification with Neural Networks: An Examination of Several Impactful Variables

Jingqing Yang

# Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

Music genre classification is sometimes deemed a subjective matter. Genre tags of musical tracks are often marked by the artist or users. Although there have been commercialized efforts to automatize this procedure using collaborative filtering on user preferences, much fewer attempts have been made using machine learning.

Machine learning research is at an all-time high with exceptional success in image recognition tasks. The ImageNet Large Scale Visual Recognition Challenge (ILSVRC) is an annual contest for visual object recognition, classification and localization. The breakthrough in solving the ILSVRC in 2012 is often considered the beginning of the AI revolution in the 2010s. The winning programs classification error was 28.2% in 2010, but in the latest ILSVRC 2017, 29 out of 38 participating teams achieved an error below 5%, which is the estimated human performance.[10] The success in image recognition is what inspired this thesis project. Music genre classification poses as a similar problem but in the realm of music or audio in general. While recommender systems like collaborative filtering exploits external information such as user preferences, we seek to classify music with a content-based approach using neural networks.

## 1.1 Previous Work

There has been a lot of work done in the field of music information retrieval (MIR) and audio classification.

Lee et al. applied a convolutional Deep Belief Network (CDBN) to unlabeled auditory data and evaluated the learned features on several classification tasks.[6] The tasks included speaker identification, speaker gender classification, phoneme classification, music genre classification, and music artist classification. In order to learn unsupervised features, they first trained on unlabeled datasets and extracted two layers of CDBN features. For the task of music genre classification, the dataset was from the 2004 ISMIR Audio Description Contest. (TODO: ref) The audio data was first represented as spectrograms of 20 ms window size with 10 ms overlaps. After PCA-Whitening to reduce the size of each sample, the data was then fed into the CDBN as input. Lee et al. then evaluated the learned features with a five-genre classification task and achieved 73.1% test accuracy (the baseline was 20%). The five genres were classical, electric, jazz, pop, and rock. Their study showed that the first layer of CDBN features outperformed raw spectrogram, Mel-frequency cepstral coefficients (MFCCs), the second layer of CDBN features, and the combination of both layers of CDBN features.

Later, another group once again approached audio feature learning with Deep Belief Networks.[4] Similar to [6], it first extracted unsupervised features by training a DBN on unlabeled data. The source of its data came from the Tzanetakis dataset, which was consists of ten genres. The audio data was processed through a discrete Fourier transform before being fed into the DBN. It then compared the DBN learned features with the MFCCs by evaluating them on a non-linear support vector machine classifier. The result showed that the DBN features were more effective than the MFCC, with a test accuracy of 84.3% (the baseline was 10%).

Aforementioned studies all shared one similarity: they relied on pre-train feature extraction using DBN. But there are not many studies on other NN models' performance on music genre classification.

One of the few audio classification studies with CNN was research by Li et al.[7] It used three convolutional layers to extract auditory features through supervised training. The preprocessing method they chose was MFCCs. With the GTZAN dataset, training was done in groups of three genres to preserve the accuracy while accelerating convergence. Then trained filters were used as features for a classifier. The study showed comparable performance on the training set, but the validation set had significantly inferior result of below 30%.

The lack of studies on the task of music genre classification with other NN structures inspired this project. In nature, convolutional neural networks and Deep Belief Networks share many similar features. We also believe in the advantages brought by CNN or RNN structure will benefit auditory data classification. With the goal to testing as well as improving performance using other NN models, we started this thesis project.

## 1.2 Background Knowledge

### 1.2.1 Pre-processing

The raw format for audio files is usually the waveform, as commonly seen in audio editing softwares such as Audacity. An example raw waveform of a 0.1 second clip is shown in Figure 1.1. The x-axis represents time and the y-axis represents the amplitude. When reflected as data, the waveforms are stored as one-dimensional arrays. Although neural networks can be used as universal function approximates, a well-represented dataset can

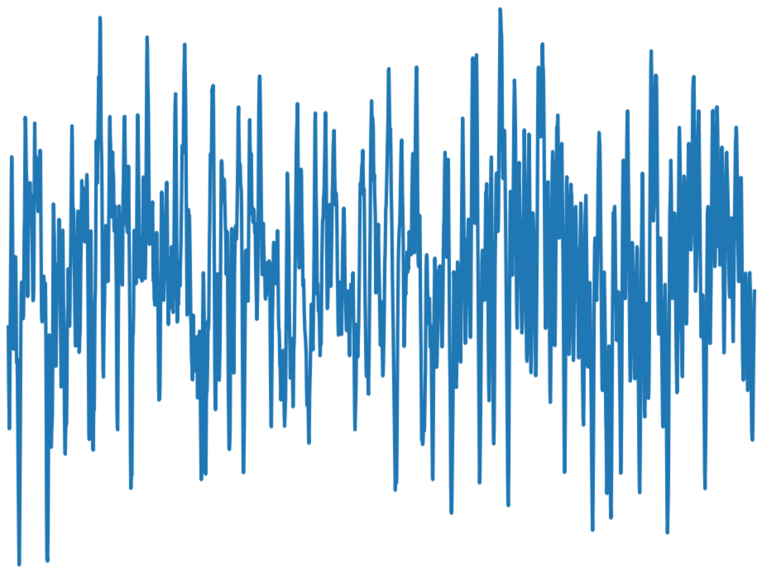


Figure 1.1: Waveform of a 0.1 second clip

greatly benefit the performance. Audio-related researches usually performs pre-processing on raw waveforms so that the auditory features are emphasized. The common approaches for pre-processing includes short-time Fourier transform (STFT), Mel-spectrogram, and Mel-frequency-cepstral-coefficients (MFCC).[2] The fundamental mechanics for these audio preprocessing methods rely on Fourier Transform, whose definition follows.

$$\hat{f}(\xi) = \int_{-\infty}^{\infty} f(x) e^{-2\pi i x \xi} dx \tag{1.1}$$

In the above equation, $f$ is the original function of time where x represents time. $\hat{f}$ is the transformed function of frequency where $\xi$ represents frequency.

The Fourier transform was largely inspired by study of Fourier series, which decompose a complicated function into sums of simple waves. The result of a Fourier transform is multiple frequency bins with their corresponding magnitudes. In form of data, this would be a one dimensional array, where the frequency information is carried by the indices and magnitude information carried in the numbers. A popular way of using Fourier transforms is called the short-time Fourier transform (STFT). It applies a Fourier transform to small windows from a waveform and combines the results into a two-dimensional array. With a proper sampling rate, a long audio file can be broken into several chunks and each can be transformed separately. The combined matrix shows the time-frequency relationship, with the values in each grid representing the magnitude of a certain frequency at a certain time. The result of a STFT is called a spectrogram. With audio data, especially music, a popular upgrade to spectrogram is the to use a mel-scale, instead of a linearly spaced frequency scale. A mel-scale is based on pitch comparisons. As frequency increases, equal mel-intervals require larger and larger frequency leaps. A formula to convert f (Hertz) to m (Mel) is

$$m = 2595\log_{10}(1 + \frac{f}{700}) \tag{1.2}$$

[9] With mel-scales, the mel-spectrogram emphasizes lower frequencies and compresses higher ones. This approximates human auditory perception.

Another common preprocessing technique is to use mel-frequency cepstral coefficients (MFCC). Given a mel-spectrum and a number of selected mel-frequencies, a common practice takes the logs of the powers at each of the mel-frequencies, and then takes the discrete cosine transform at the mel log powers. The amplitudes of the resulting spectrum are MFCCs.[11][14] They are often used to extract features out of audio data.

### 1.2.2 Neural Networks

Neural networks are the core of this project. The history of neural networks in artificial intelligence can be traced back to the 1940s, but their performance only became significant

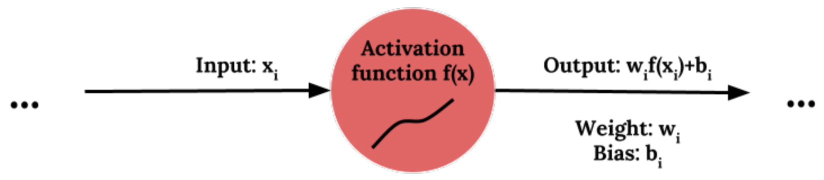


Figure 1.2: A single neuron in a neural network

in the past twenty years.[8] Among many models of neural networks, convolutional neural networks (CNN) and recurrent neural networks (RNN) are used in this project. Any kind of neural network consists of neurons (nodes) and edges. During the learning stage, neurons of the first layer take the inputs, put them through an activation function, and output the results into the next layer.

$$\sigma(x) = \frac{1}{1+e^{-x}} \tag{1.3}$$

An activation function usually maps the input value into a certain range to indicate its potential to ”activate” the next neuron. An example activation function is a sigmoid function, shown in Equation 1.3, which maps the input into $[0, 1]$. Although sigmoid is a popular choice for activation function, it suffers from the vanishing gradient problem. As the input becomes larger, the first order derivative of the sigmoid function becomes smaller. Our models mostly used the rectified linear unit (ReLU) for activation function. It has the advantage of reducing the vanishing gradient problem as well as introducing sparsity into the network. Its equation follows.

$$f(x) = \begin{cases} 0 & \text{if} x > 0 \\ x & \text{if} x \geq 0 \end{cases} \tag{1.4}$$

The mechanism for a single neuron is shown in Figure 1.2. Neurons of the next layer then take the weighted sum of the previous layer's outputs and repeats the same procedure until it reaches the final output layer. For a classification network, the number of neurons in the last layer is equal to the number of categories, meaning that each output neuron represents the possibility of a category.

Then, a loss function compares the predicted values with the ground truth and produces a non-negative value indicating deviance from the truth. For classification tasks, the ground truth usually consists of one-hot-encoded vectors representing the labels. Taking the derivative of the loss function with respect to all weights and biases in the network creates a gradient. This gradient suggests what changes in the weights and biases can cause the loss function to increase the most. The core mechanic of neural networks is backpropagation, which takes the opposite direction of the gradient on the output layer to modify the previous layer, and repeats the process until it reaches the inputs (the first layer). the aforementioned steps describe the training process with one sample. The network repeats this process with all training samples.

### Convolutional Neural Networks

Traditional multilayer perceptron models are fully connected and work fairly well with image recognition tasks. However, they do not scale well with high resolution images due to the restriction of computing power. In addition, multilayer perceptrons do not take into account the spatial structure of visual patterns, and thus distant pixels can have the

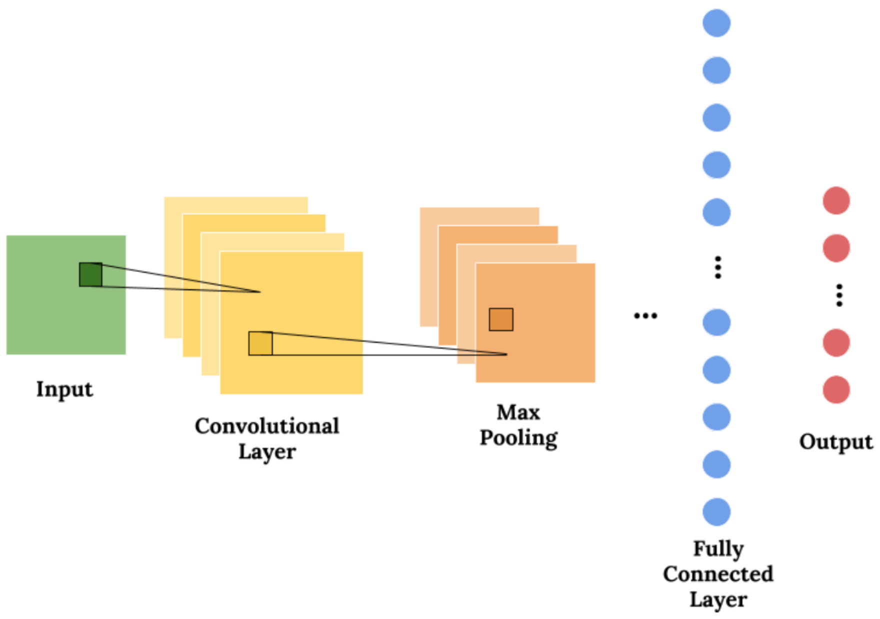


Figure 1.3: General form of a CNN model

same impact in recognition of an area as a closer pixel. CNNs overcome this problem by implementing 3D layers that are only connected to a small region of the previous one and filters in the same layer share the weights and biases. Therefore, the number of parameters in one convolutional layer is given by

$$(n^2 * x) \times 2$$

, where $n$ is the side length for one small region and $x$ is the number of filters in this layer. A general model of a CNN is shown in Figure 1.3.

## Recurrent Neural Networks

Recurrent Neural Networks are often used for sequential data analysis (e.g. text prediction and speech recognition). Units within a RNN are connected along a time sequence, where each unit represents a new time step. There is one input and one output for each time step. A simplified graph for a general RNN model is presented below.

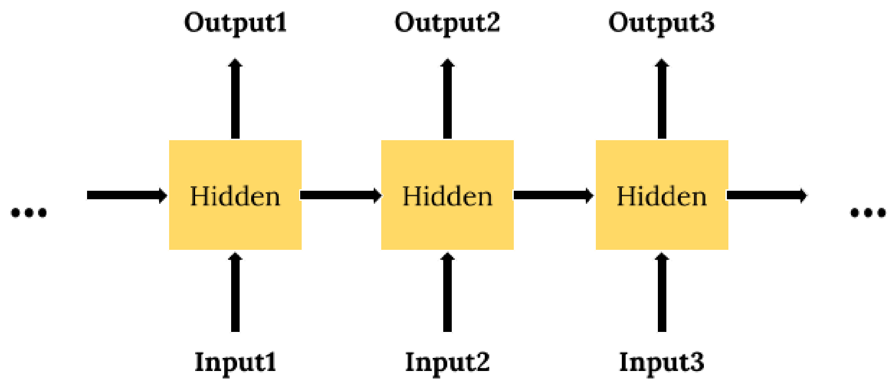


Figure 1.4: Graph representation of a RNN model

The sequential structure is for visualization purpose. In practice, the model is usually circular in nature. There is only one copy of the state, which keeps updating itself with a combination of the previous input and hidden state.

# Chapter 2

# Methods

We experimented with different approaches to classify auditory data, and different combinations of the approaches. All of our approaches involved a three-part procedure. The first step was to find a dataset and split it into a training group, a testing group, and a hold-out group. The second step was to transform the raw audio files into a form with clearer musical features. We then train a neural network on pre-processed data to classify genres.

There were several components in the aforementioned procedure, including choices of dataset, pre-processing methods, and neural network structures. For dataset selection, we utilized the GTZAN dataset and the Million Song Dataset.[13][1] For pre-processing methods, we tested a Fast-Fourier Transform, a mel-spectrogram and mel-frequency cepstral coefficients. As for the neural network model, we experimented with both convolutional neural networks and recurrent neural networks. The results chapter will show that the combination of the GTZAN dataset, mel-spectrograms, and convolutional neural networks yielded the best result.

## 2.1 Datasets

### 2.1.1 GTZAN Dataset

The first dataset we came across was the dataset collected by G. Tzanetakis and P. Cook, often referred to as the GTZAN dataset.[13] This dataset was collected from various sources including personal CDs, radio, microphone recording, and so on. It consists one hundred half-minute audio clips in each of ten genres, totaling 1000 tracks. The ten genres are blues, classical, country, disco, hip-hop, jazz, metal, pop, reggae and rock. The tracks are all monaural and have a 22050 Hz sampling rate.

| Genre | Blues | Classical | Country | Disco | Hiphop | Jazz | Metal | Pop | Reggae | Rock |
|---|---|---|---|---|---|---|---|---|---|---|
| Size | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 |

Table 2.1: GTZAN genres and sizes

The GTZAN has been widely used in music genre classification research since its release in 2002. We chose this dataset as our starting point because it was well organized and frequently cited by previous research we came across. This precedence gives it credibility as well as offering us a frame of reference for our network performance. There were, however, a few drawbacks to using this dataset. The most limiting factor was its size. Although we broke each track into six (five seconds each) to thirty (one second each) sub-clips, there were still at most 30,000 samples in total. To compare the scales of magnitude, the n-MNIST dataset of handwritten digits has 60,000 training samples and the ImageNet has 14,197,122 images in total.[5][3] The insufficiency in our GTZAN dataset caused the network to overfit, as shown in the results section. The diversity of the audio sources was also a problem, because it introduced noise into the music. At the early stages of training, it could have led the feature-learning astray.

| **Genre** | **Size** |
|---|---|
| Rock | 8478 |
| Electronic | 5865 |
| Rap | 1812 |
| Reggae | 1558 |
| Metal | 1452 |
| Jazz | 1335 |
| Blues | 1178 |
| Country | 844 |
| Punk | 559 |
| Pop | 522 |
| Folk | 517 |
| RnB | 510 |
| World | 286 |
| Latin | 268 |
| New Age | 87 |

Table 2.2: MSD genres and sizes

### 2.1.2 Million Song Dataset

The Million Song Dataset (MSD) is a freely available collection of audio features and metadata for a million contemporary tracks.[1] The dataset contains metadata and derived features such as release year, artist, terms of the artist, similar artists, danceability, energy, duration, beats, tempo, loudness, and time signature. The drawback of this dataset is that it contains neither the audio excerpts nor genre tags. However, given the corresponding IDs, MSD provided the code to fetch half-minute preview audio clips from 7digital, a music and radio services platform. Due to availability issues on 7digital, only part of the one million songs were successfully collected. Tagtraum industries provided genre annotation to a subset of the Million Song Dataset.[12] As a result of the aforementioned insufficiencies, the resulting number of usable tracks was 25,271. Table 2.2 shows genres and their corresponding sizes, arranged descending in order of size.

Tagtraum annotation includes the following genres: blues, country, electronic, folk, jazz, latin, metal, new age, pop, punk, rap, reggae, RnB, rock, and world. Same as the GTZAN dataset, the audio clips are monaural and have a sample rate of 22050 Hz.

Overall, the MSD provided a significant increase in the size of our data: about 25 times the size of GTZAN. Moreover, the quality of audio data in the MSD is better, in contrast to the noisy audio found in GTZAN. However, the MSD had its own drawbacks, the most relevant one being inconsistent genre sizes. Although the total amount of tracks in MSD was larger, the smallest genre, New Age, only contained 87 tracks, smaller than GTZAN genres. We considered 1000 tracks per genre a significant upgrade from the GTZAN dataset, where there are 100 tracks per genre. However, only seven genres in MSD have at least 1000 tracks: blues, electronic, jazz, metal, rap, reggae, and rock. Blues is the the smallest genre containing 1178 tracks. Because our CNN model took all training and testing data together, we needed to ensure that the entire input was formed with even genre distribution. Therefore, when using a CNN on the MSD data, we chose only these seven genres and trimmed them down to the same size: 1178 tracks, the size of the smallest genre. The RNN model, on the other hand, took samples one batch at a time. We could then maintain an evenly distributed input while using the entire MSD by drawing random tracks from each genre.

## 2.2 Pre-Processing

### 2.2.1 The Fast-Fourier Transform

For audio processing, we initially read in every 30-second clip with the Sunau library (as the tracks were in .au format), split it into ten-millisecond windows, and applied Fast-Fourier Transform on each small section (220 frames). The data shape of the output was the same as the input, which was an array of 220 complex numbers. They represent intensities of 220 evenly-divided frequency bins. We further edited the array by computing the magnitude of each complex number so that they became real. Combining all FFT outputs in a 30 second clip resulted in a two-dimensional array that could be graphed as a heat map. However, the resulting heat map looked chaotic and random. Figure 2.1 shows the first 500 frames of a classical song, where the horizontal axis represents the time and the vertical axis represents frequency.

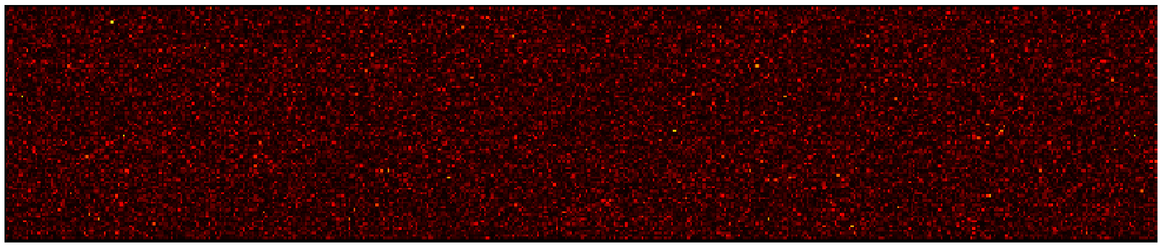


Figure 2.1: Heat map from Sunau and FFT

The randomness in the graphs propelled us to look for other audio processing tools. We eventually settled on the Python library Librosa. After reading in the tracks with Librosa and keeping rest of the procedure intact, the resulting heatmap for a four-second clip from the same song is shown below in Figure 2.2.

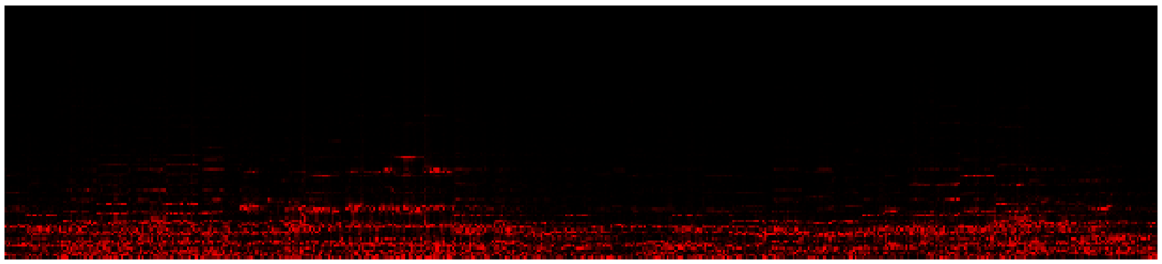
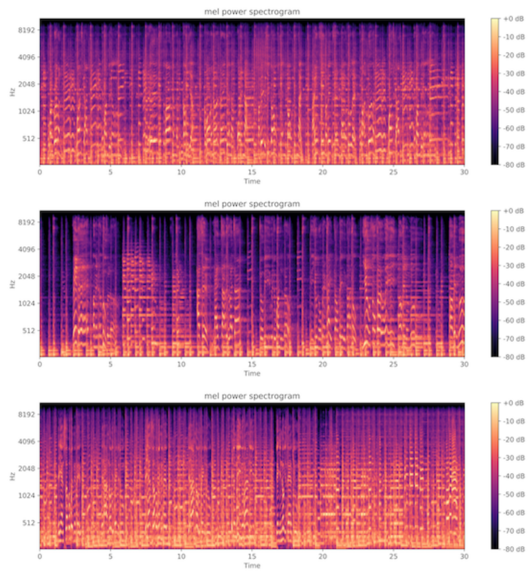


Figure 2.2: Heat map from Librosa and FFT

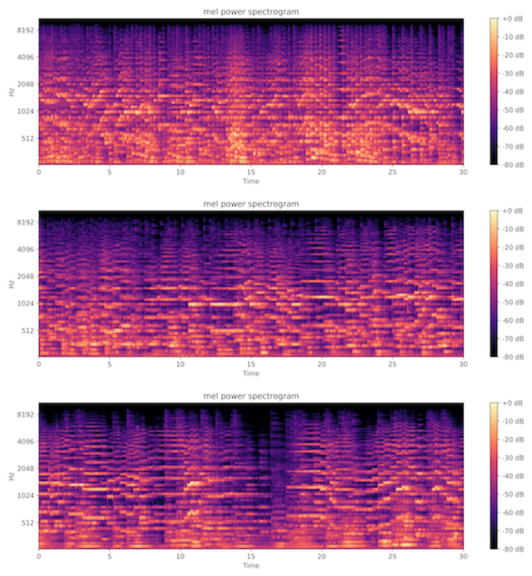
From the heat map we could clearly see musical features such as longer lasting notes that represented classical music. There were also human-recognizable traits from other genres such as the steady, low-frequency, brief, beats found in hip-hop music and chaotic clusters for metal music. These traits gave us confidence that spectrogram was a sensible form of representation. However, we noticed that the top three quarters of the heat map were usually inactive and the bottom quarter was always compact. Such observation lead us to an improved representation: mel-spectrograms.

### 2.2.2 Mel-Spectrograms

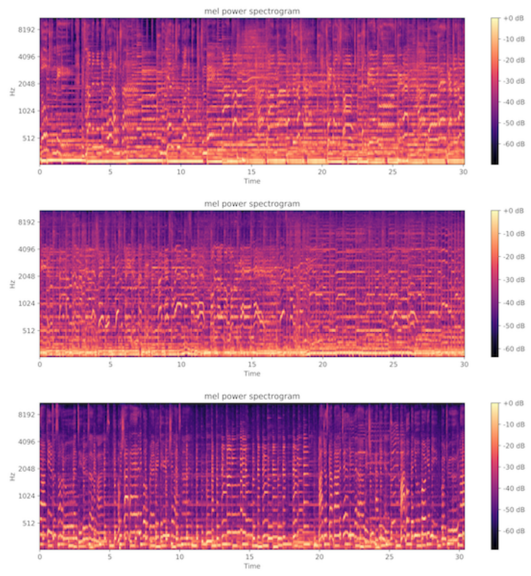
What differentiates Mel-spectrograms from regular spectrograms is their frequency spacing on y-axis. Mel-frequency spacing better approximates the hearing scale for human ears where lower frequencies are emphasized and higher frequencies are compressed. This approach seemed especially appropriate because our heat map results from our fast-Fourier transforms showed that most active sounds occured in lower frequencies. We used Librosa to produce a mel-spectrogram for each track. The processed tracks were split into smaller clips as individual samples. We varied the length of such samples to find the optimal partitioning, which is discussed in the results section. In order to gain a visual perception of the mel-spectrogram results, we picked three random tracks from each genre and plotted them. Below are the graphs for all genres in the GTZAN dataset.



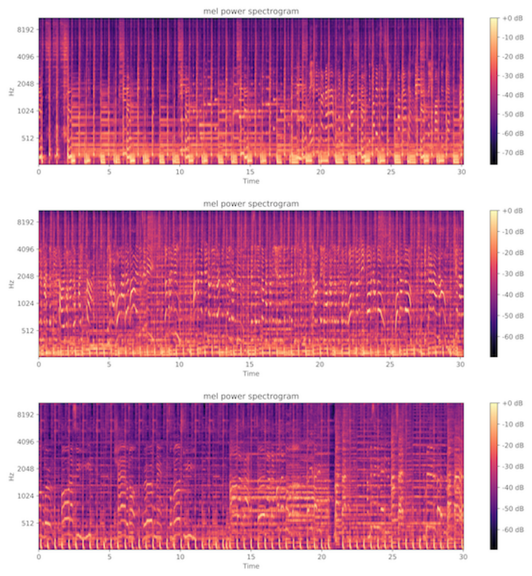
(a) Blues



(b) Classical

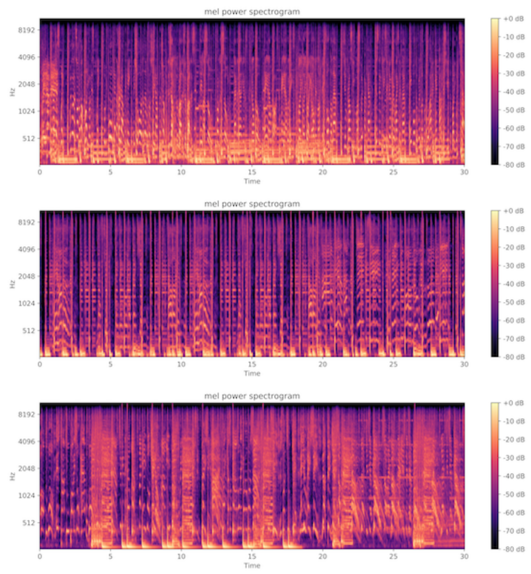


(c) Country

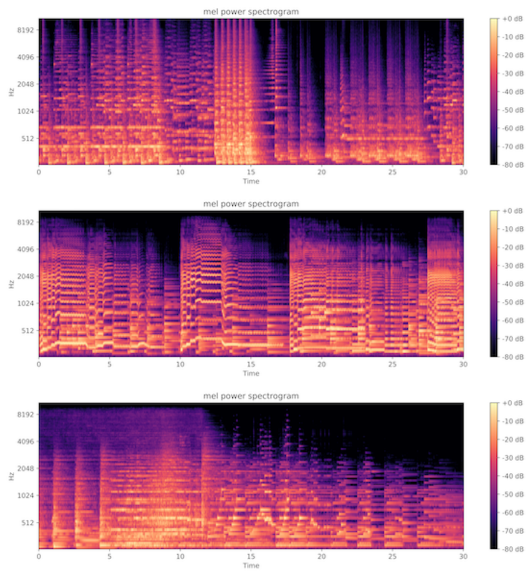


(d) Disco

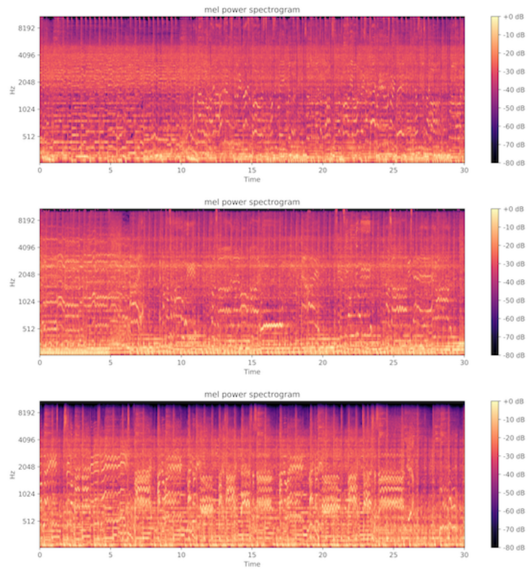
The figures show similarities shared within genres as well as differences across genres. Just by looking at the random samples of mel-spectrograms, we could conclude distinct



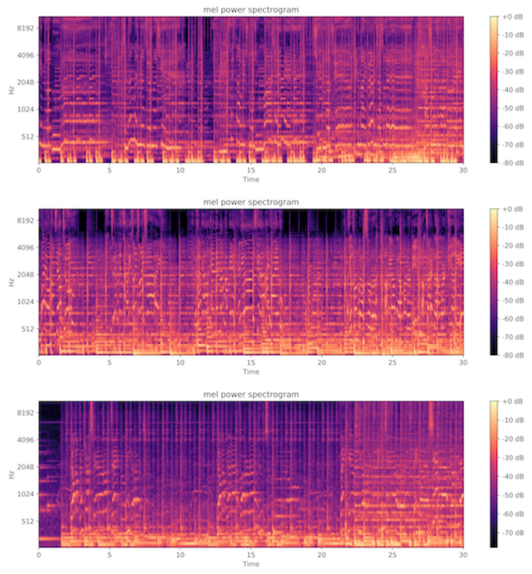
(e) Hiphop



(f) Jazz

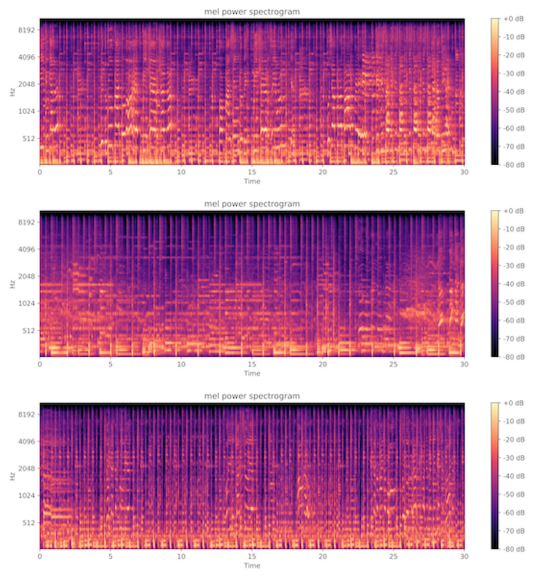


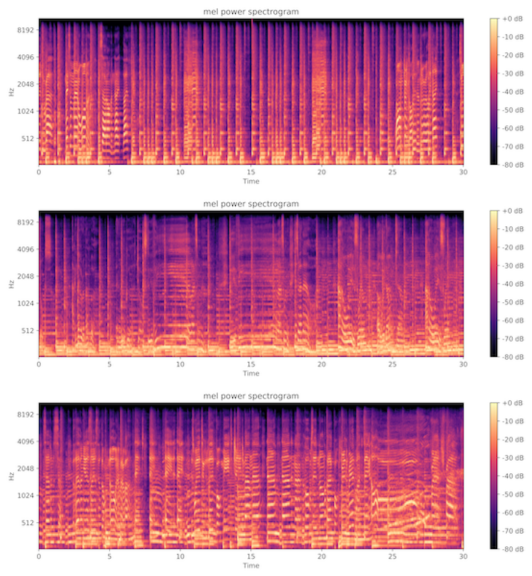
(g) Metal



(h) Pop

features for a few genres, but not all of them. Similar to the results from our fast-Fourier transforms, classical tracks feature long horizontal lines in the spectrograms representing

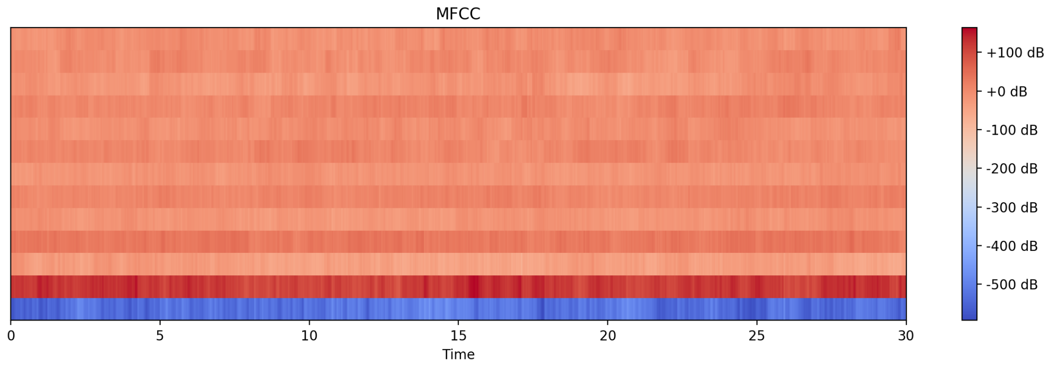

(i) Reggae


(j) Rock

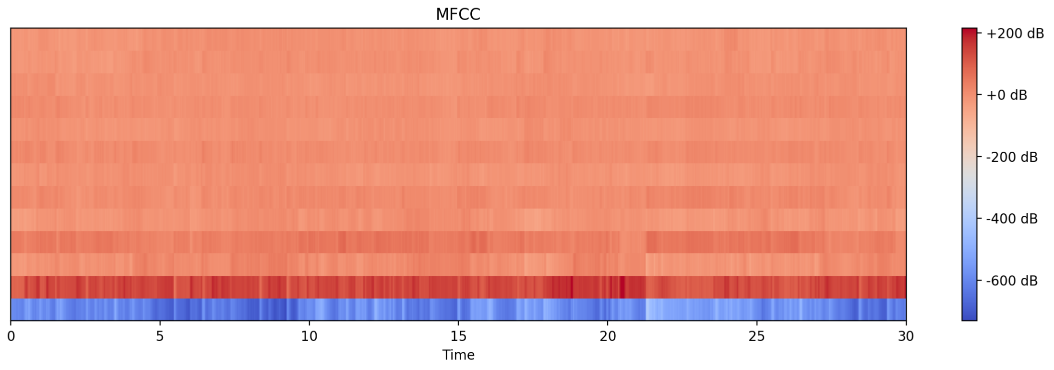
lasting and steady notes. Metal tracks feature heavy activity across the frequency spectrum so that the entire mel-spectrogram appears bright. Disco tracks feature long vertical lines with equal intervals on the spectrograms because of their steady beats. Hiphop tracks also feature vertical lines across the entire frequency spectrum, but the intervals are not uniform across the track, most likely due to the fact that hiphop songs change up their rhythm every once in a while. Although we could interpret some visual features with musical knowledge about these genres, there were some similarities that we could not explain. Also, some genres look very similar, like disco, rock, and pop.

### 2.2.3 MFCCs

With the Librosa library, we were able to apply several other advanced pre-processing methods. Mel-frequency cepstral coefficients (MFCCs) were one of the most frequently used methods by genre classification studies. We chose a window size of one second and



(a) MFCCs of a classical song



(b) MFCCs of a jazz song

Figure 2.4: MFCCs of a classical song and a jazz song

thirteen coefficient channels. Although MFCCs extracted specific features of the audio, we had no way of identifying the meaning of these features. Figure 2.4 shows the MFCCs of a classical song and a jazz song, respectively.

## 2.3 Neural Networks

### 2.3.1 Convolutional Neural Networks

Our first attempt using a neural network was a convolutional neural network. The reason behind this choice was both empirical and intuitional. CNNs are often very good at image recognition tasks and even the simpler models are able to yield high accuracy. Moreover, by looking at our spectrograms it seems like a small portion of the full song should still offer enough information to determine its genre. Since CNNs are usually used for image related learning, each sample is expected to have three dimensions: height, width and three color channels. However, our data had neither the color channels nor the audio equivalence of them (stereo input channels). Therefore we simply added an extra dimension in our data that only had one element, to approximate black and white images. We split the processed the dataset into three parts: training (50%), testing (20%), and holdout (30%). As explained in the Dataset section, when the input set was GTZAN, the CNN model could use the entire dataset. When the input set was MSD, the CNN model could only use seven genres, each containing 1178 tracks. The CNN model was consisted of two groups of convolutional layers followed by a max-pooling layer, and at the end they were flattened, densed, dropped out at a rate of 0.5, and densed again. We adapted this model on Keras from one that was designed for image recognition on the dataset Cifar-10 and had a impressive accuracy with that task. The detail of our model is presented in Figure 2.5.

Ultimately, CNNs were and are usually designed for image recognition and classification. The differences in the fundamental nature of images and audio led us to decide that convolutional neural networks would not work as well for audio feature learning as it does on images. One feature that differentiates music from images is that audio signals carry the sequentiality and relativity that image pixels don't possess. This fundamental difference

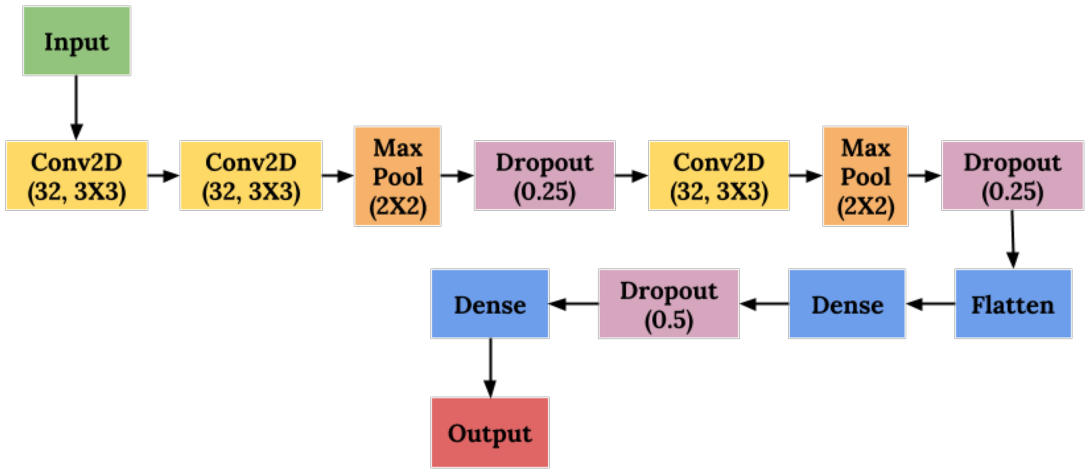


Figure 2.5: CNN Model

caused us to move on and look into the recurrent neural networks.

### 2.3.2 Recurrent Neural Networks

Recurrent neural networks are mostly used on tasks with sequential data, such as speech recognition, grammar learning, or text prediction. Music shares a sequential nature with speech and text, as the flow from one note to the next determines the mood of melody and hence the genre. Given this knowledge, recurrent neural networks seemed like the a logical next step.

We settled on Pytorch as our choice for a machine learning library because of its balance between ease-of-use and full control. The specific structure of our model is shown in Figure 2.6.

Instead of training all thirty seconds of the track, we decided that a subset of it would be enough to distinguish one genre from another. We started with a five second sample size. We

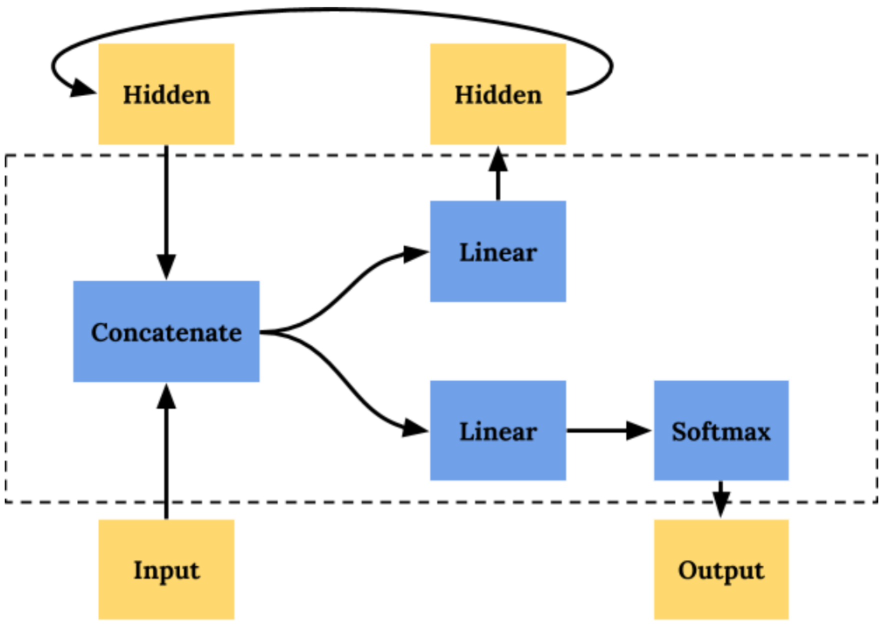


Figure 2.6: RNN Model

increasd it and compared the performance of each sample size. For each training iteration through the RNN, we generated a batch of training samples, consisting of approximately 50 randomly chosen five-second samples from all genres. When the input source was the GTZAN dataset, we drew the samples randomly from a set of mel-spectrogram processed samples, where all genres were aggregated and shuffled. We were able to do this because the genre distribution in GTZAN was already evenly distributed. When the input source was from the MSD, we made sure to draw the same amount of samples from each genre for every batch. We were actually able to utilize the entire MSD this time around as our random selection for each batch could reuse samples between batches. With random sample drawing, we could keep the batch balanced while utilizing the entire dataset.

### 2.3.3 Aggregation

After finishing training and comparing different architectures, we saved the best neural net model with its structure and weights. The holdout group (not used during training or testing) from the same dataset was then fed in to the model to predict labels. We aggregated the predictions for every track, using majority rule. At the end, we compare the track-based predictions to the ground truth to compute the final accuracy.

# Chapter 3

# Results

There are many variables when determining the final classification accuracy. We chose to analyze the effects of the following few: pre-processing method, sample length, number of genres, choice of dataset, and neural network structure. In order to identify each variable's effect individually, we compared results by varying one variable at a time while keeping the rest unchanged.

## 3.1 Pre-processing

| **Pre-processing** | **FFT** | **MFCC** | **Mel-spectrogram** |
|---|---|---|---|
| Sample Length | One second | One second | One second |
| Neural Network | CNN | CNN | CNN |
| Dataset | GTZAN | GTZAN | GTZAN |
| Number of Genres | 10 | 10 | 10 |
| **Accuracy** | **13%** | **36%** | **48%** |

Table 3.1: Comparison between pre-processing methods

Three main methods of pre-processing we used were the fast-Fourier transform, mel-frequency cepstral coefficients, and mel-spectrograms. The following comparison results were gathered with the GTZAN dataset (ten genres), one-second sample length, and a convolutional neural network with Keras. The result is presented in Table 3.1.

It is clear that mel-spectrograms outperform the other two methods by a significant margin. Therefore, we continued to test other variables with mel-spectrogram processed data.

## 3.2 Sample Length

From a human perspective, it usually takes less than a few seconds to determine the genre of an audio excerpt. Therefore, we used five seconds per sample as the upper-bound of our sample size. However, the usual input size for many NN models remains fairly small. The CNN model we adapted from was originally designed for CIFAR-10 data, where each sample was $32 \times 32$ pixels with three color channels. With Librosa's mel-spectrogram function, the recommended number of mel-frequencies is 128, so this became the height of our sample. With the length of the fast-Fourier transform window at 1024 and sample rate of 22050Hz, one second of sample translated to approximately 42 pixels in width. In order to bring the dimensions of our input data close to the original $32 \times 32 \times 3$, our first attempt for sample size was $42 \times 128 \times 1$ (grey-scale for color). We later experimented with three-second and five-second window sizes while keeping the pre-processing method as a mel-spectrogram, neural net model as a CNN, and source of data as the GTZAN dataset with ten genres. The comparison of results is shown below.

| Pre-processing | Mel-spectrogram | Mel-spectrogram | Mel-spectrogram |
|---|---|---|---|
| **Sample Length** | **One second** | **Three seconds** | **Five seconds** |
| Neural Network | CNN | CNN | CNN |
| Dataset | GTZAN | GTZAN | GTZAN |
| Number of Genres | 10 | 10 | 10 |
| **Accuracy** | **48%** | **52%** | **49%** |

Table 3.2: Comparison between various sample lengths

## 3.3 Neural Network Models

Out of all models we tested, CNN in Keras and RNN in Pytorch yielded the most promising results, while other models failed to do so. This may be due to improper implementation, so we are only presenting the results from our CNN and RNN, under the controlled environment of five-second samples from ten genres in GTZAN dataset processed into mel-spectrograms. The table of results can be seen in 3.3

| Pre-processing | Mel-spectrogram | Mel-spectrogram |
|---|---|---|
| Sample Length | Five Seconds | Five Seconds |
| **Neural Network** | **CNN** | **RNN** |
| Dataset | GTZAN | GTZAN |
| Number of Genres | 10 | 10 |
| **Accuracy** | **52%** | **33%** |

Table 3.3: Comparison between different NN models

Table 3.3 clearly shows that our CNN performs better with the task. This could be due to the difference in complexity between the two models, as the CNN model had multiple layers while the RNN model was plain.

## 3.4 Number of Genres

As presented in previous chapters, the Million Song Dataset suffers from an inconsistent genre distribution. Our CNN model's rigidity with batching restricted its utilization of MSD to seven genres and less. Therefore, we chose the Pytorch RNN model to test the number of genres' impact on classification accuracy, with five-second samples processed as Mel-spectrogram. The table of results is presented below as Table 3.4.

| Pre-processing | Mel-spectrogram | Mel-spectrogram |
|---|---|---|
| Sample Length | Five Seconds | Five Seconds |
| Neural Network | RNN | RNN |
| Dataset | MSD | MSD |
| **Number of Genres** | **10** | **14** |
| **Accuracy** | **32%** | **23%** |

Table 3.4: Comparison between numbers of genres

From Table 3.4, it is clear that as the number of genres increases, the classification accuracy decreases. However, this does not provide much insight on the performance of the network because the numbers were not on the same scale. For a ten-category classification, the baseline is 10%, while for a fourteen-category classification, the baseline is around 7%. Without proper conversion metrics, it is meaningless to compare the numbers.

## 3.5 Datasets

Our initial motivation to increase the size of dataset was to solve the overfitting problem at early stages. Due to reasons mentioned in the previous sections, using a CNN on the full MSD was not an option. However, we were able to make comparisons for the rest of the combinations. The first group used the RNN model in Pytorch, with five-second mel-spectrograms drawn from ten genres in each dataset. Table 3.5 shows the results.

| Pre-processing | Mel-spectrogram | Mel-spectrogram |
|---|---|---|
| Sample Length | Five Seconds | Five Seconds |
| Neural Network | RNN | RNN |
| **Dataset** | **GTZAN** | **MSD** |
| Number of Genres | 10 | 10 |
| **Accuracy (Training, Testing)** | **50%, 33%** | **40%, 32%** |

Table 3.5: RNN accuracy comparison between datasets

From Table 3.5, it seems like changing to a larger dataset did not change the network performance. However, a difference exists in the comparison of training accuracy. It shows that using the MSD yielded a much smaller gap between the training and testing accuracy, hence easing the overfitting problem. However, with respect to the overall goal of genre classification, a larger dataset did not improve the performance.

| Pre-processing | Mel-spectrogram | Mel-spectrogram |
|---|---|---|
| Sample Length | Five Seconds | Five Seconds |
| Neural Network | CNN | CNN |
| **Dataset** | **GTZAN** | **MSD** |
| **Number of Genres** | **10** | **7** |
| **Accuracy (Training, Testing)** | **95%, 49%** | **85%, 68%** |

Table 3.6: CNN accuracy comparison between datasets

Table 3.6 shows a comparison of using various datasets with a CNN. From Table 3.6, we see that a CNN shows the same trend. The accuracies shown in Table 3.6 cannot be compared horizontally because GTZAN and MSD used in this comparison contained different numbers of genres (10 for GTZAN and 7 for MSD). However, vertical comparisons show the dataset's impact on the overfitting problem.

## 3.6 Binary Classification

After inspecting the results presented above, we decided to interpret the performance on a genre level and plotted the results from a RNN with GTZAN dataset, with the MSD using 10 genres, and with the MSD using 14 genres, separately.

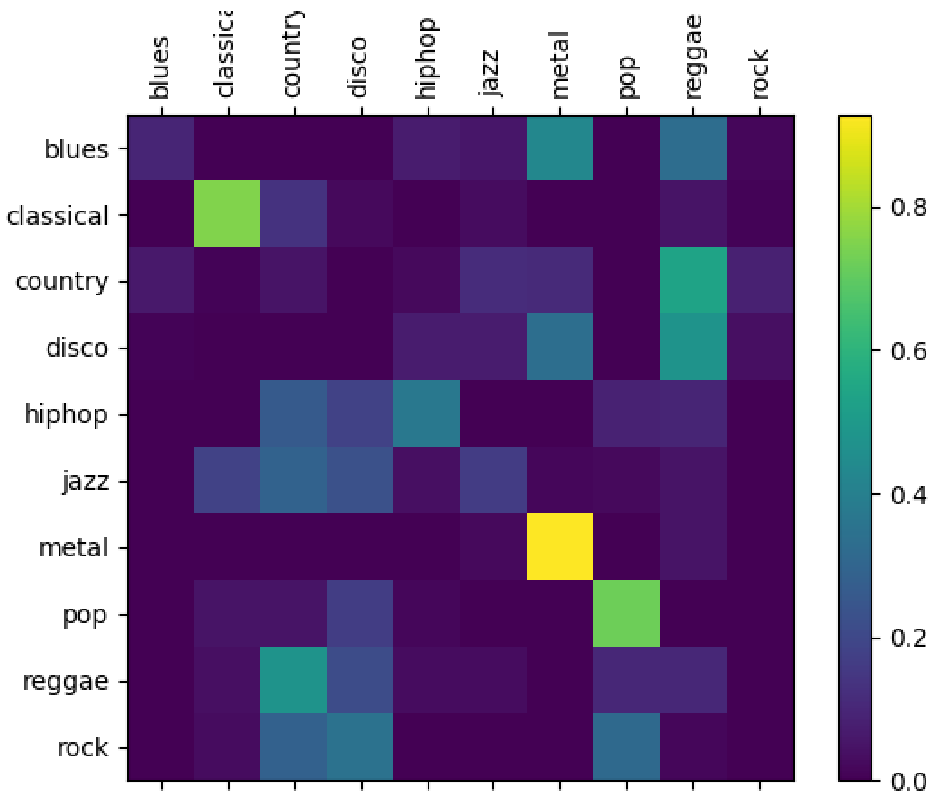


Figure 3.1: Results of a RNN using the GTZAN

The vertical axis represents the prediction and horizontal axis represents the ground truth. The diagonal lines in all three graphs shows that predictions in general reflect the ground truth. It is also clear that the network performs better with some genres than

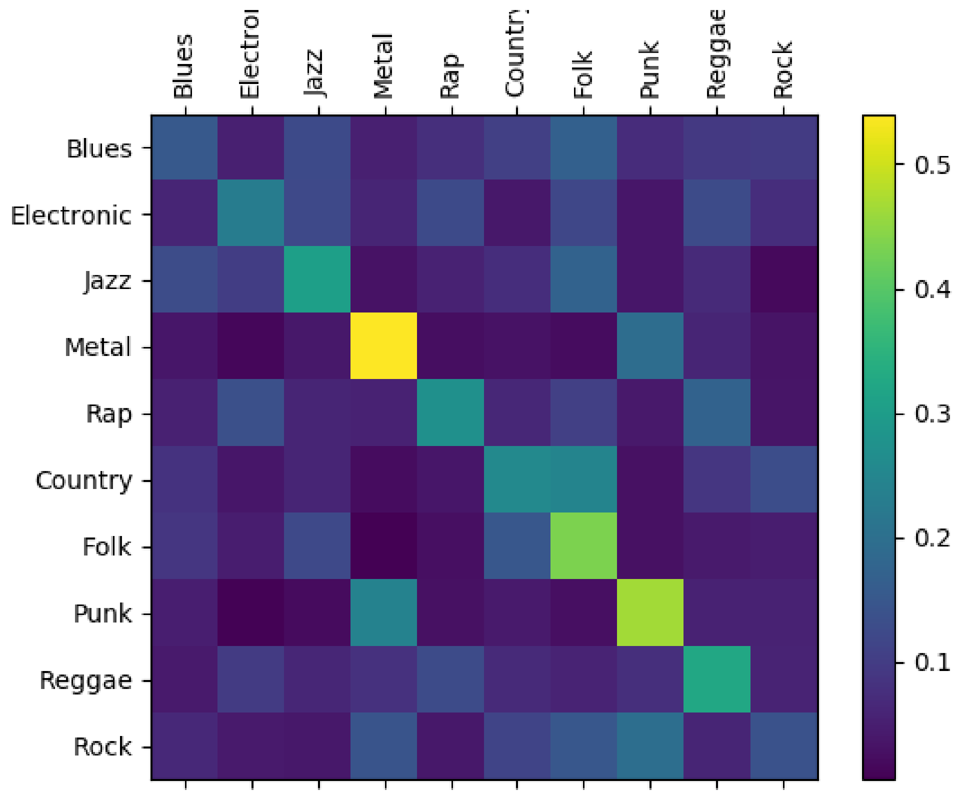


Figure 3.2: Results of a RNN using the MSD with 10 genres

others. Metal, for example, has always been easily identifiable, while blues was hard to identify. The difference between genres inspired us to further investigate the easiness to identify each genre.

We modified our CNN model to perform binary classification and tested the accuracy for all ten genres from GTZAN. The results are shown in Table 3.7.

Table 3.7 show that classification performance was very inconsistent across genres. While metal and classical music are comparatively distinct, rock and blues are often ambiguous. This discovery is consistent with with human performance, as it's more difficult for humans

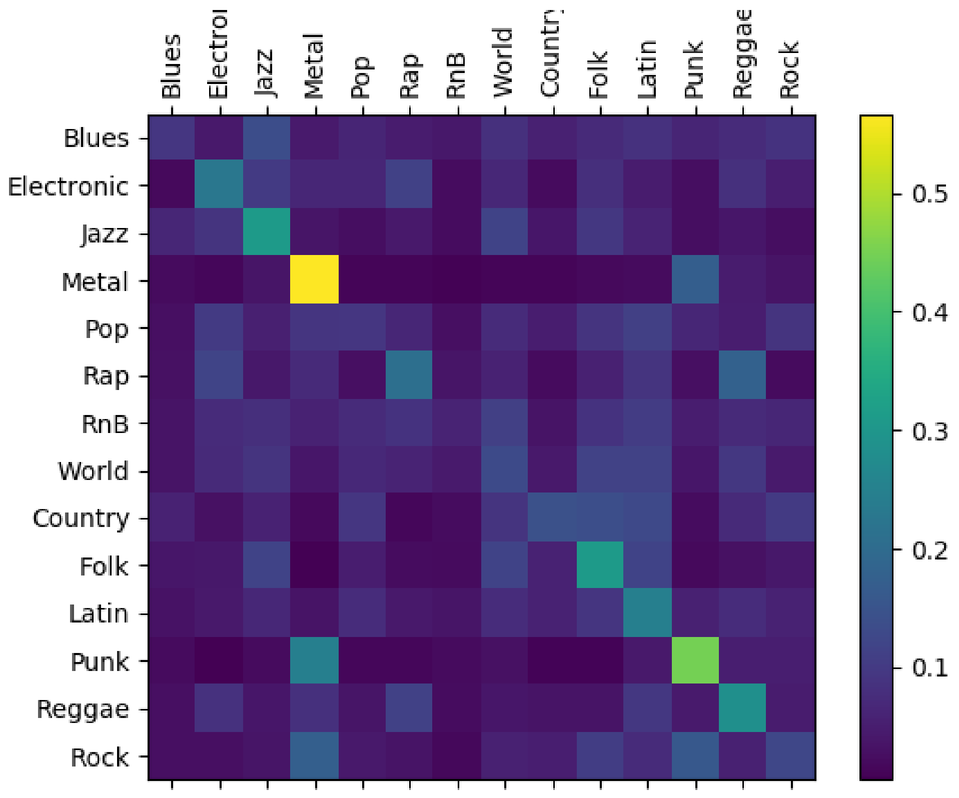


Figure 3.3: Results of a RNN using the MSD with 14 genres

to identify some genres than others. When all genres are combined together, the result only reflects the average performance. It could indicate that different genres require different amount of time to convergence. The inconsistency among convergence time could affect the overall training performance. It also proved that genre selection could greatly affect the difficulty of a classification task. A dataset consisting of metal, classical, hiphop tracks would be significantly easier to classify than one consisting of jazz, rock and blues tracks.

| **Genre** | **Accuracy** |
|---|---|
| Metal | 97.1% |
| Classical | 91.4% |
| Hiphop | 83.1% |
| Reggae | 79.0% |
| Country | 74.3% |
| Pop | 71.2% |
| Disco | 70.0% |
| Jazz | 70.0% |
| Rock | 69.5% |
| Blues | 66.7% |

Table 3.7: Binary genre classification results

## 3.7 Aggregation

In the end, we performed majority voting on our best-performing architecture: a three-second sample size from GTZAN dataset, mel-spectrograms, and a CNN classifier. Each track contained ten such clips that were voted upon. After majority voting, track-based holdout set performance showed an 7% increase, adding up to a 59% accuracy as final result.

# Chapter 4

# Conclusion

After testing several choices of datasets, pre-processing methods, neural network structures, and other factors, we found the optimal combination to be a convolutional neural network using mel-spectrograms of three-second samples of audio. A bigger dataset reduces the overfitting problem but has very little impact on validation accuracy. Our final best validation accuracy turned out to be 59%. Although it was inferior to the state-of-art accuracy for music genre classification, it outperformed other attempts to solve this challenge with convolutional neural networks. We also discovered that the classification accuracy was highly genre-dependent, which could have impeded the overall performance. It also showed genre selection's great impact on the classification difficulty.

# Bibliography

[1] Thierry Bertin-Mahieux, Daniel P.W. Ellis, Brian Whitman, and Paul Lamere. The million song dataset. In *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR 2011)*, 2011.

[2] Keunwoo Choi, George Fazekas, Kyunghyun Cho, and Mark Sandler. A comparison on audio signal preprocessing methods for deep neural networks on music tagging. *arXiv preprint arXiv:1709.01922*, 2017.

[3] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[4] Philippe Hamel and Douglas Eck. Learning features from music audio with deep belief networks.

[5] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[6] Honglak Lee, Peter Pham, Yan Largman, and Andrew Y Ng. Unsupervised feature learning for audio classification using convolutional deep belief networks. In *Advances in neural information processing systems*, pages 1096–1104, 2009.

[7] Tom LH Li, Antoni B Chan, and A Chun. Automatic musical pattern feature extraction using convolutional neural network. In *Proc. Int. Conf. Data Mining and Applications*, 2010.

[8] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[9] D. O'Shaughnessy. *Speech communications: human and machine*. Institute of Electrical and Electronics Engineers, 2000.

[10] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[11] Md. Sahidullah and Goutam Saha. Design, analysis and experimental evaluation of block based transformation in mfcc computation for speaker recognition. *Speech Communication*, 54(4):543 – 565, 2012.

[12] Hendrik Schreiber. Improving genre annotations for the million song dataset. In *ISMIR*, pages 241–247, 2015.

[13] G. Tzanetakis and P. Cook. Musical genre classification of audio signals. *IEEE Transactions on Speech and Audio Processing*, 10(5):293–302, Jul 2002.

[14] Min Xu, Ling-Yu Duan, Jianfei Cai, Liang-Tien Chia, Changsheng Xu, and Qi Tian. Hmm-based audio keyword generation. In *Pacific-Rim Conference on Multimedia*, pages 566–574. Springer, 2004.