Music Genre Classifier

Austin John - May 27th, 2022

Abstract

Audio and Image processing is one of the most complex tasks in Data Science and Machine Learning. One application of Audio processing is implementing a Music Genre Classifier that aims to classify audio files to a certain genre. This paper will go through implementing the Music Genre Classifier.

1. Introduction

Companies like Spotify, Soundcloud or YouTube Music nowadays use music classification to make more refined and targeted recommendations to their customers. An example of this is Spotify's *Discover Weekly* playlist that updates every week based on your current music choice and then recommends similar music to you; Spotify also does a similar thing with its *Song Radio* feature where it suggests and plays songs like the one you were just listening to.

Spotify, however, defines its genres of music beyond the traditional ones. Its algorithm extracts several features from each song to analyze and fit into or near relevant genre cluster spaces. The Spotify technical team has set up a sample cluster space for all their genres at Every Noise At Once. On this website, you can see nearly 6,000 genres from Spotify clustered based on their similarity to one another. Clicking on a genre will play a sample from it. A demonstration of Spotify's genre classification in practice is hosted on Every Noise Latest Release By Genre. The Spotify Team has also released a document "How We Understand Music Genres" that explains the cluster visualization, how they tackle the term "genre" in terms of data science and their reasoning behind classifying what genre a song is.

Music Classification is also used by standalone products like Shazam that try to identify music playing in your environment by using your phone's microphone. Determining music genres is the first step in this direction and Machine Learning techniques have proven to be quite useful in extracting trends and features from a large data pool. We, however, did not have the same tools, models and/or computing power a decade ago when the GTZAN dataset was first made and companies were scrambling to implement a good recommendation system in their app to bring more users in. With modern day advances in Deep Learning techniques, these services have significantly improved music genre classification. Of Course, generalized audio classification has more far-reaching applications such as speech detection, separating background noise from primary speech, sentiment analysis, speaker identification for voice verification, etc.

2. Motivation

Today, services like Spotify and YouTube instantly create playlists and suggest the next song to be played based on our listening patterns. These patterns are affected by tone, mood and the genre of the songs we listen to. There are a multitude of variables that quantitatively influence the decision, but it ultimately comes down to something as simple as how the song "sounds". For instance, a metal song may have more percussion and a harder bass line as opposed to pop which might just feel "funkier" and more upbeat.

As an avid listener of music and a user of a music streaming service, I was curious to know how we would instruct a model to classify the genres of a song. It is easier for human beings to discern between different genres by just feeling out the song, but you unfortunately cannot replicate that in a model. This piqued my interest and I decided to try implementing the same genre classifier to learn what categories and features a model extracts to classify a song and also to evaluate the amount of data you'd need to build a reliable model.

3. Dataset

The GTZAN genre collection (commonly known as the MNIST of audio) was collected in 2000-2001. It consists of 1000, 30 second long audio files. It consists of 10 genres, each represented by 100 tracks that are all 22050Hz Mono 16-bit audio files in .wav format.
The 10 genres are:

- Blues
- Classical
- Country
- Disco
- Hip Hop

- Jazz
- Metal
- Pop
- Reggae
- Rock

The dataset has the following files/**folders**:

- **genres_original** - A collection of 10 genres with 100, 30 second long audio files each.
- **images_original** - A visual representation of each audio file. One could use this as an input and implement a Neural Network to classify the genres visually.
- features_30_sec.csv - This csv contains the features all the audio files. The file has the mean and variance computed over multiple features that can be extracted from an audio file like zero crossing rate, spectral centroid, tempo, etc.
- features_3_sec.csv - This csv follows the same structure as the previous one but each audio file has been split to 3 second long audio files, hence increasing the size of our data pool.

4. Representing Audio

Selecting the data representation is very important before choosing your Machine Learning Algorithm . Converting audio data into a numeric or vector format will determine how much pivotal information is retained when we change formats.(Guo et al., 2017) For instance: it would be difficult for even the most superior Machine Learning Models to learn and classify genres if the data format cannot represent something as simple as the loudness and pace of a rock song.

While there are several methods to represent Audio Data, one of the more common methods are Mel-Spectograms. Mel-Spectograms visualize audio signals based on their frequency unlike some other datasets like the Spotify Classification Problem 2021 that uses other specifications such as BPM, loudness, acoustics, energy, liveliness, etc to describe each song quantitatively. Moreover we can use music as a time series data using Mel-Frequence Cepstral Coefficients(MFCC's). Each of these data formats has its own benefits and disadvantages and would vary from application to application.

4.1. Audio Processing with Python

Sound is represented in the form of an audio signal with parameters like frequency, bandwidth, decibels, etc. A typical audio signal can be expressed as a function of Amplitude and Time.
These sounds are available in a multitude of formats, some

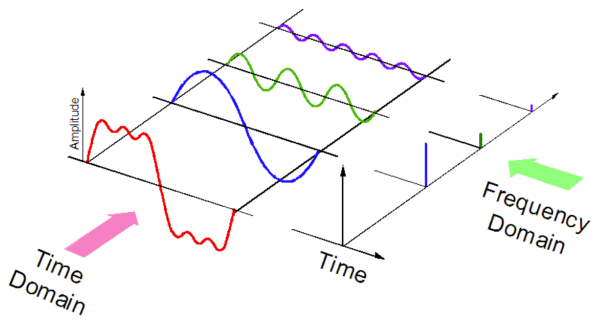

Figure 1. The Central Idea of Audio

of these formats include:

- .mp3 format
- .WMA (Windows Media Audio) Format
- .wav (Waveform Audio File) Format

We will be dealing with the .wav format in this paper since the audio files in the database are saved as .wav files.

4.2. Audio Libraries

Python has some remarkable libraries for audio processing like Librosa and PyAudio. There also are a multitude of of built in libraries that we can leverage when dealing with audio files.

4.2.1. Librosa

Librosa is a python module to analyze audio signals that are geared more towards music as opposed to other forms of audio files like speech. It includes the essential tools one would need to build an MIR (Music Information Retrieval) System. We can also leverage librosa's resampling capabilities by choosing to load files in with a set sample rate. The sample rate of a song is the number of samples of audio carried per second, usually measured in Hz or KHz.

4.2.2. PyAudio

IPython.display.audio lets you play an audio file directly in a jupyter notebook. Leveraging the module lets you add an audio widget in the jupyter notebook where you would be able to play the file.

4.3. Visualizing Audio

We will be using the following methods to visualize audio and extract features of the audio files to train our model.

4.3.1. WAVEFORM

We can leverage librosa and matplotlib to plot the signal waves of each genre. Here we have the plot of the generated **Signal Waves** of Pop and Disco. We can see the differences in the amplitude envelopes among the different genres.

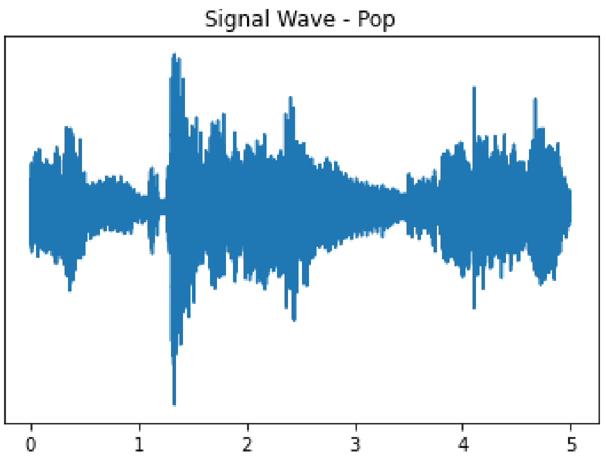

Figure 2. Signal Waveform - Pop

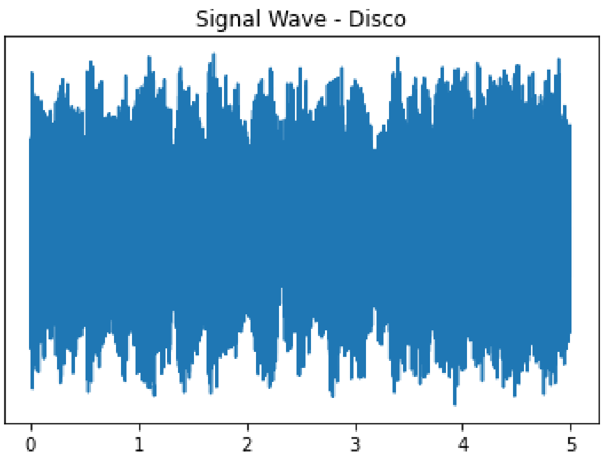

Figure 3. Signal Waveform - Disco

4.3.2. SPECTOGRAM

A **Spectogram** is a visual representation of the spectrum of frequencies of sound to other signals as they vary with time. Spectograms are sometimes also called **sonographs**, **voicegrams** or **voiceprints**. They are also called **Waterfalls** when represented in a 3-Dimensional Plot. We again leverage librosa to display the spectograms of Pop and Disco.

The vertical axis shows frequencies (0 to 10KHz) and the horizontal axis shows the time of the clip.

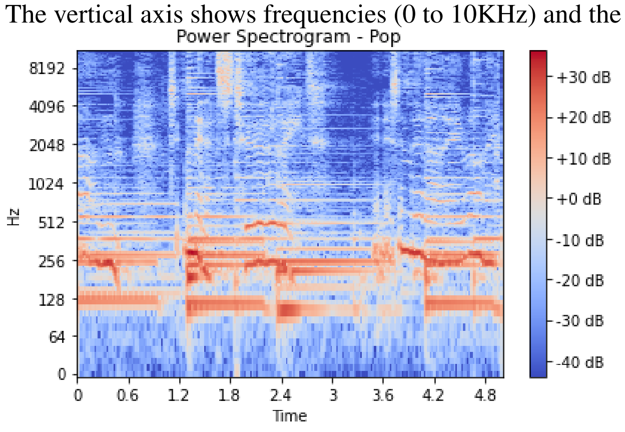

Figure 4. Spectogram Pop

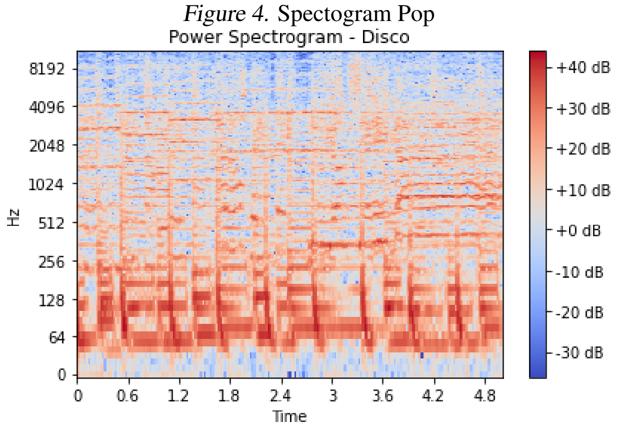

Figure 5. Spectogram Disco

4.3.3. SPECTRAL ROLLOFF

Spectral Rolloff is defined as the frequency R below which 85% of the magnitude distribution is concentrated. The mean and the variance of the rolloff across time frames is used as a feature in our model.

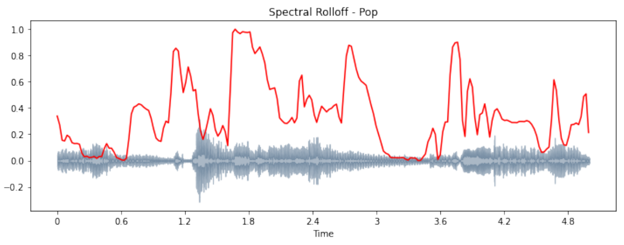

Figure 6. Spectral Rolloff Pop

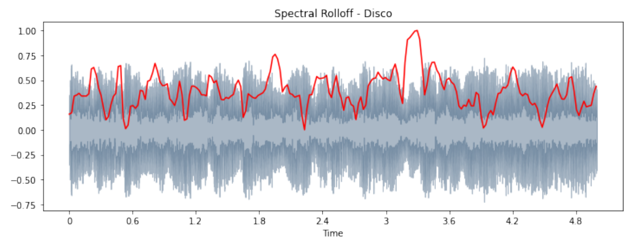

Figure 7. Spectral Rolloff Disco

4.3.4. Chroma Features

Chroma Features are a tool for analyzing music features whose pitches can be categorized and whose tuning approximates to the equal-tempered scale. One main property of chroma features is that they capture harmonic and melodic characteristics of music while being robust to changes in timbre and instrumentation.

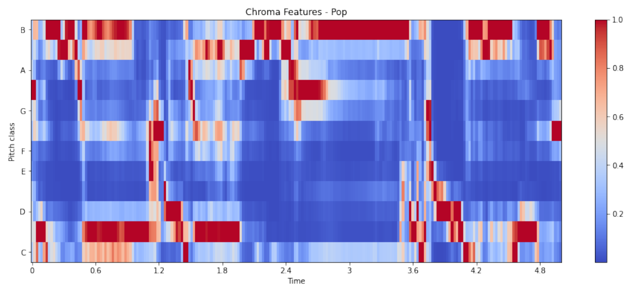

Figure 8. Chroma Features Pop

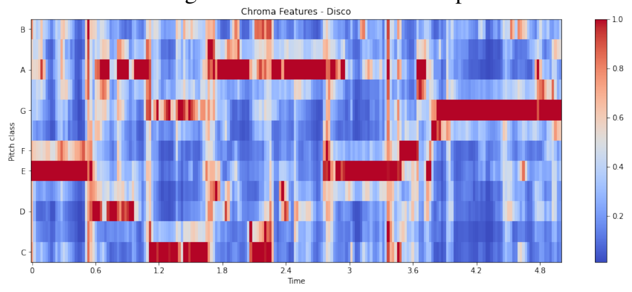

Figure 9. Chroma Features Disco

4.3.5. Zero Crossing Rate

Zero Crossing Rate is the rate at which a signal changes from positive to negative or back. This feature is widely used in both speech recognition and Music Information Retrieval. It usually has higher values for highly percussive sounds like in metal and rock. We can deduce the zero crossing rate from the graph - the number of times the plot changes sign.
Pop: **3** Disco: **22**

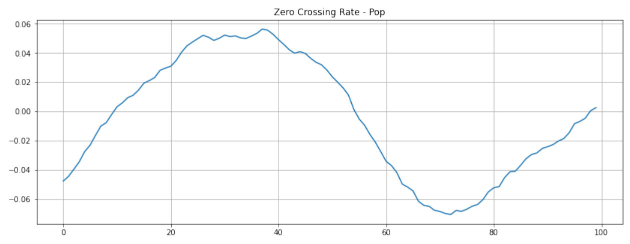

Figure 10. Zero Crossing Pop

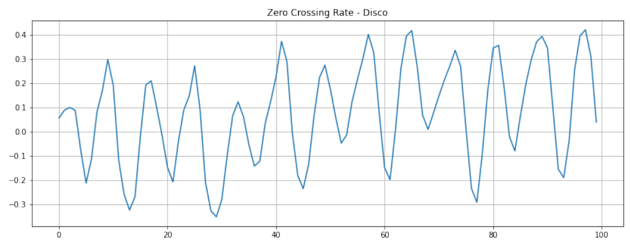

Figure 11. Zero Crossing Disco

5. Methodology

There are various methods to train models to classify this dataset, some of the approaches are:

- Support Vector Machines
- K Means Clustering
- K-Nearest Neighbors
- Convolutional Neural Networks.

I will be implementing a Convolutional Neural Network (CNN) to train my model. I chose this because studies (Tzanetakis & Cook, 2002) have shown it to produce the best result for the problem at hand.

Table 1: Accuracy of predictions by model used.

	With data processing			Without data processing		
	Train	CV	Test	Train	CV	Test
Support Vector Machine	.97	.60	.60	.75	.32	.28
K-Nearest Neighbors	1.00	.52	.54	1.00	.21	.21
Feed-forward Neural Network	.96	.55	.54	.64	.26	.25
Convolution Neural Network	.95	.84	**.82**	.85	.59	.53

We see a clear trend here between using raw data (without data processing) and using Mel-Spectograms (with data processing). We also see a substantial improvement in accuracy in all four models after converting the data to Mel-Spectograms.

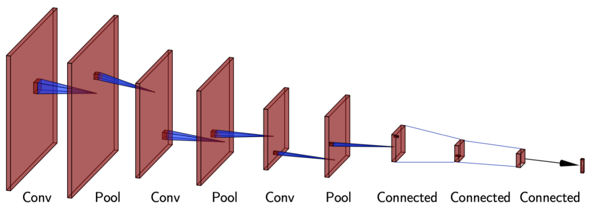

Figure 12. Convolutional Neural Network
(Huang et al., 2018)

6. Convolutional Neural Network

A **Convolutional Neural Network** is a Deep Learning Algorithm which takes an input feature set and then extracts features from it to then classify upon.

The Convolutional Neural Network implemented in my project consisted of 4 convolution layers, where each layer is max pooled and the convolution layers use the ReLU activation function. The final output layer then uses the softmax function.

7. Experimentation

The primary goal of my project was to evaluate what Machine Learning Method would be the best to implement a classifier. The secondary goal was to also evaluate how much of a difference using the .csv file with MIR of 3 second long audio files would make as opposed to using the .csv file with the MIR of the 30 second long audio files with training our model.
With the methodology and process defined, I then created a Jupyter notebook for my test bed since it seemed like the more intuitive choice when working on a Machine Learning Project. I leveraged robust libraries like Pandas, NumPy, SciPy and Matplotlib to import, format the data and to plot data on graphs. The input data is stored in 2 csv files that are imported in as DataFrames using Pandas.

7.0.1. Importing Data

Both csv files contain the same information about the audio files except for the fact that the features_3_sec.csv has 10x more information listed out since the csv contains information about the audio files that have been split to smaller 3 second long audio files essentially increasing the amount of data for our model to train on by tenfold. The csv files are imported as DataFrames using Pandas, and we then drop the filename column from both dataframes since that is not relevant information that our model can train on. This step leaves our dataframes with the following trainable columns.

We have mean and variance of Chroma Features, Root Mean Square, Spectral Centroids, Spectral Bandwidth, Spectral Rolloff and Mel-Frequency Cepstral Coefficients as our columns.

7.0.2. Decoding Audio

We then pass the audio files through librosa's load function. This function returns the **time series** of the audio file. A time series consists of two things:

- The Sample Rate: This variable represents the number of samples recorded per second.
- A 2D Array:
  - First Axis: Represents the recorded samples of amplitudes in the audio
  - Second Axis: Represents the number of channels in the audio file.

The audible range of human hearing goes up to 20 kHz and is used as the default value for sample rate in the load function, but I resampled the audio files to 45600Hz in my project.

7.0.3. Feature Extraction

We need to preprocess the data before we train a model on it. As was seen in Table 1: A model does better during training after data preprocessing.
We will use the last column of the dataframe that contains the labels (genres) to encode our labels with the function LabelEncoder() from the sklearn.preprocessing library. The LabelEncoder() function helps us convert the categorical text data into model-understandable numberical data.
We then use the fit.transform() function to fit the LabelEncoder() data adn return the new Encoded Labels.

7.0.4. Scaling the Features

The Standard Scaler() function is used to standardize features by removing the mean and scaling to unit variance. The standard scope of sample x is calculated as:

$$z = (x - u) / s$$

where u and s is the mean and standard deviation of the training samples respectively.

7.0.5. Training and Testing Sets

I used sklearn's train_test_split() function that splits the data set into training and test sets. I chose to go with the industry standard 80-20% split with my data.

7.0.6. Convolutional Neural Network Model

I implemented a Convolutional Neural Network with 4 convolution layers, each max pooled and regularized with the ReLU activation function. The final output layer uses the softmax function and the loss is calculated as cross entropy loss. I used the Adam optimizer function to train the model. I trained the model for 800 Epochs, but the model seemed to converge to an accuracy score of 92 after 100-150 Epochs.
I used keras's Dropout() function to prevent overfitting by setting input units to 0 and a rate frequency that is passed into the function.
I made 2 CNN models, one that used the features_3_sec.csv file and the other model using the features_30_sec.csv.

8. Observations

Now that we have the models implemented, we can run our test bed and calculate the metrics for the different models and the difference between using both csv files.

- **features_30_sec.csv:** The Dataframe we get from the csv file has 1000 rows and 60 columns for the model to train on.

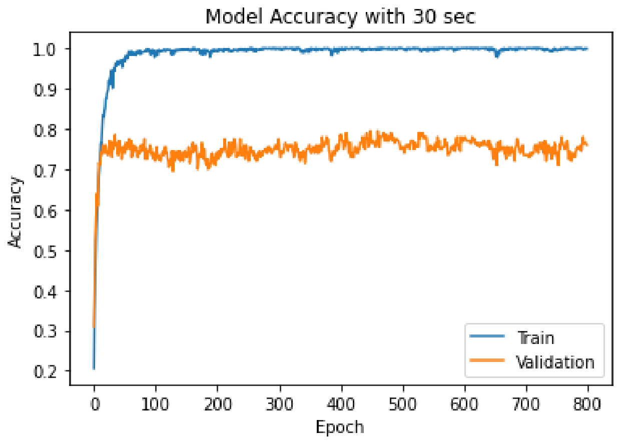

Figure 13. Model Accuracy 3 sec

- **features_3_sec.csv:** The DataFrame from this csv contains 9990 rows and 60 columns.

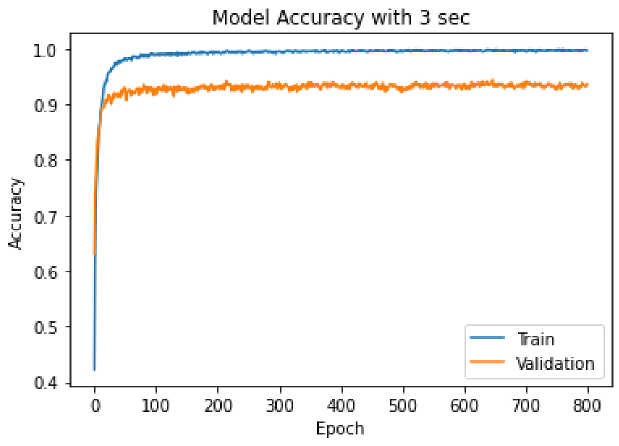

Figure 14. Model Accuracy 30 sec

As we can see, the tenfold increase in the input data size boosts our accuracy score from 73% with the 30 second features to 92% with the 3 second features.

9. Conclusion

This report demonstrates a popular approach to implementing a Genre Classifier using Supervised Machine Learning Methods. The report shows that implementing a Convolutional Neural Network yields a Genre Classifier with the highest accuracy. The report also and that formatting and processing the data would help the model substantially. We also see that our model gets more accurate when we split the audio files in the dataset to increase the size of the training set. The test loss and accuracy for the model with 3 second features is 0.6 and 92% respectively while the test loss and accuracy for the model with 30 second features is 2.6 and 73% respectively.
In conclusion, among all Supervised Machine Learning Techniques, Convolutional Neural Networks prove to be useful to implement the most accurate genre classifier. CNN's extract features more efficiently as opposed to other real world strategies like LTSM for scenarios like Music Genre Classification.

References

Guo, L., Gu, Z., and Liu, T. *Music Genre Classification via Machine Learning*. 2017.

Huang, D., Serafini, A., and Pugh, E. *Music Genre Classification*. 2018.

Tzanetakis, G. and Cook, P. Musical genre classification of audio signals. *IEEE Transactions on Speech and Audio*

Processing, 10(5):293–302, 2002. doi: 10.1109/TSA.2002.800560.