GTZAN Dataset - Music Genre Classification

Eden Bar-Tov, Yaron Grushka

Submitted as final project report for the Deep Learning course,
IDC, 2021

1 Introduction

Music is fascinating. For many years, scientists have been trying to understand the intricacies of sound and what distinguishes one song from the next. In this project, we attempt to answer this very question.

1.1 Related Works

A similar problem was tackled by Julien Despois, as detailed in this Medium article. It was also suggested as part of a "Kaggle" challenge on the same dataset.

2 Solution

2.1 General approach

2.1.1 Goal

Our goal was to receive an audio file of a song, and classify the song into one of the following genres: **Blues, Classical, Country, Disco, Hip-Hop, Jazz, Metal, Pop, Reggae, and Rock.**

2.1.2 Data Representation

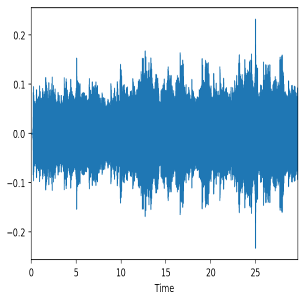
Since dealing with raw audio could be tricky, and since we have a large set of tools to tackle image data, we decided to reduce the problem to that of classifying images. One of the main hurdles in dealing with audio is the sheer amount of information per instance. For example, in a 22-kHz audio sample, every second of audio contains approximately 22,000 values, which might contain too much irrelevant data that could negatively impact the learning process. As such, we looked for a method of visualizing our audio files in the most efficient and effective way possible.

The naive way of visualizing audio is through a Wave Plot, which is a basic 2-dimensional representation of the amplitude, just like we see in equalizers.

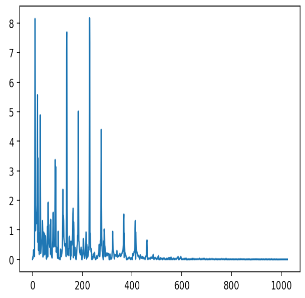
Clearly, by only factoring in amplitude, we can lose a lot of relevant data, such as relation between frequencies and even rhythm (which is obviously very crucial in genre classification). Another approach of visually representing audio is through a Fourier Transform, which allows us to represent the audio as a function of frequencies. This is the first step we take in order to generate our image data. Using the Fourier Transform, one is able to generate a visual representation of audio called a "Mel Spectrogram". As seen in other related works, Mel Spectrograms work well in representing the main features of audio data in a compact form and are a great approach to tackle our problem. A Mel Spectrogram is a combination of 2 concepts:

1. The Mel Scale, which is a non-linear transformation of the frequency scale.

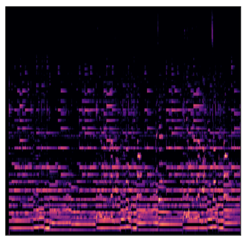
2. The Spectrogram, which is a visual representation of the spectrum of frequencies of a signal through time.



(a) Wave Plot



(b) Fourier Transform



(c) Mel Spectrogram

2.1.3 The Dataset

We searched for data that would satisfy our needs online and chose to use the "GTZAN" Dataset, which consists of 1,000 songs divided uniformly among the aforementioned 10 genres. It is a small dataset indeed, but since music might be subject to copyrights, it is the largest we could find of raw audio (".wav") files, and not a CSV-format data. Therefore, we had to get creative in order to maximize our model's learning under the constraints of our small dataset. Each of the 1,000 songs in the "GTZAN" dataset is represented by a 30-second long, 22-kHz, 16 BPS audio sample, and in addition an image of the Mel Spectrogram corresponding to that audio sample. However, even though we already had the images at hand, we eventually created new spectrograms: we sliced each audio sample to 10 equal 3-second bits, and created a new Mel Spectrogram from each of these slices. We elaborate more on the process behind this in "2.2. Design". Unlike most image classification problems, our images could not be tweaked (rotated, sliced, blurred etc), because the exact Mel Spectrogram is crucial for accurately representing the audio.

2.1.4 Data Challenges

After browsing through our dataset, we reached the conclusion that the following factors will affect our solution:

1. The size of the dataset is small. 100 samples per class is not a lot to work with, so it will be difficult to truly teach our model the intricacies of each genre. It is also important to note that after rationing some of the data for the test set, our dataset becomes even smaller.

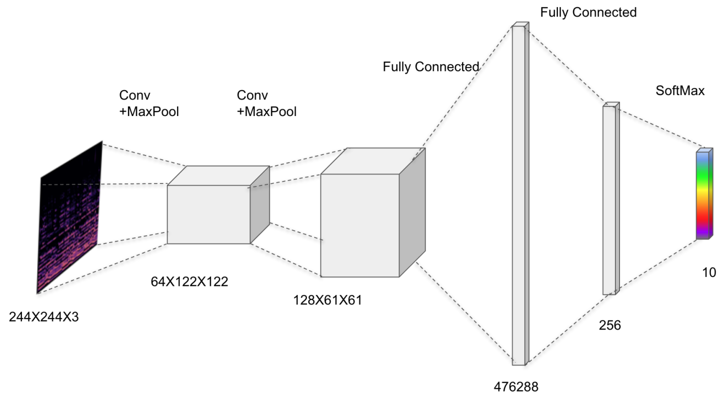
2. The data has high variance. Two different songs can be attributed to the same musical genre, but still sound nothing alike. It is important to consider that each musical genre usually has many sub-genres: for example, Heavy-Metal and Progressive-Metal are two sub-genres of Metal, and while they share some similarities, they are very different from one another.

3. Some genres are very similar to one another. For example, Blues and Jazz are two genres that can easily be mistaken for one another. An average human being will have a hard time classifying a given song to one of these two labels.

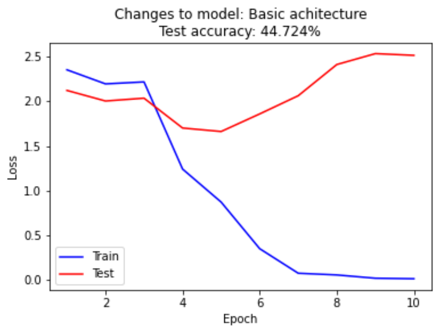
4. Some songs can share multiple musical genres. For example, many songs by "The Beatles" share both Pop and Rock characteristics.

2.2 Design

Since we use images to train our model, we used a Convolutional Neural Network (CNN). We started with a very common CNN architecture model:



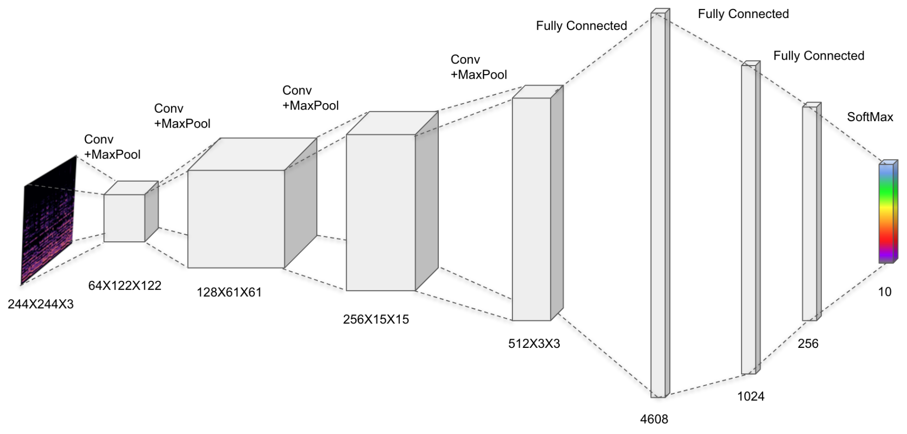
The model consisted of 2 Convolutional layers and 2 Fully-Connected layers. We used Max-Pooling, the ReLU Activation Function, the Cross-Entropy Loss Function and the "Adam" Optimizer. After 10 epochs, the following results were reached:



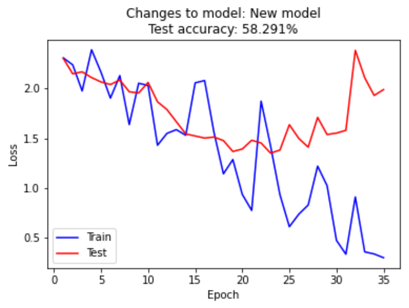
We noticed that our model achieved higher-than-expected results on the training set and underwhelming results on the test set, which suggested that it suffers from over-fitting. As such, we gradually implemented various methods and principles to help us achieve better results, and re-trained our model after every update.

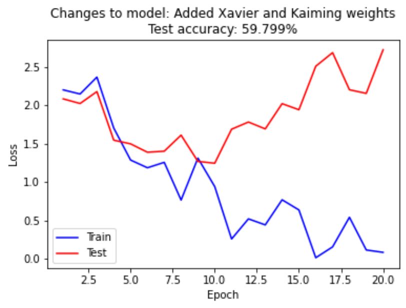
First, we changed our architecture to use 4 Convolutional layers and 3 Fully-Connected layers. The new model achieved better results indeed, reaching 58.291% precision on the test set.

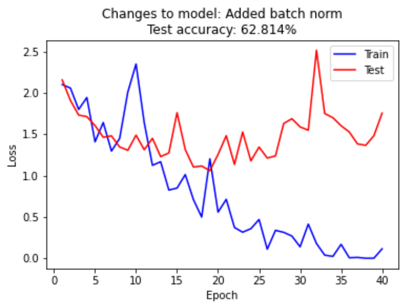
Next, we added weight initialization: we initialized the Convolutional layers with Kaiming Uniform weights, and the Fully-Connected layers with Xavier Normal weights. In the next step, we added Dropout after the first two Fully-Connected layers, and Batch Normalization after each Convolutional layer. And finally, we added the Log-Softmax function at the very end of the model. The final architecture is as follows:

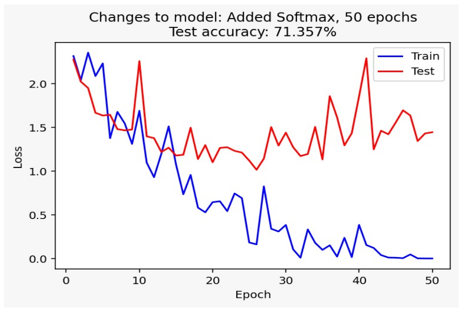


We trained this model several times, each with a different number of epochs and batch sizes and the best results were achieved with 50 epochs, and batch size of 16, where the model scored 71.357% accuracy on the test set:

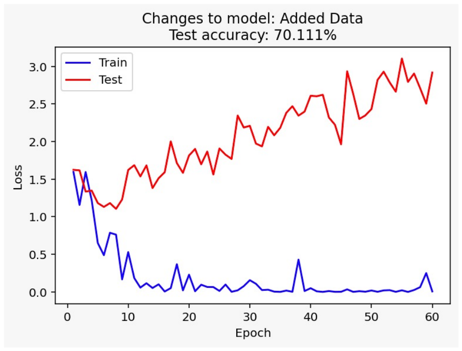








Instead of further changing our model, we decided to alter the data. The biggest constraint we were working under was the small size of our dataset. As such, we decided to take each 30 second audio sample in our dataset, split it into ten 3-second slices, and generate a Mel Spectrogram for each of these slices, thus increasing the size of our dataset tenfold. The justification behind splitting the audio samples in this manner stems from the fact that an average human being usually won't need more than 3 seconds to recognize the genre of a song. We wrote a Python script which automated the process of generating the 10,000 images from our 1,000 audio samples, and it ran for the better part of a day. Training the model on our new dataset took almost an hour, and the trained model achieved the following results:

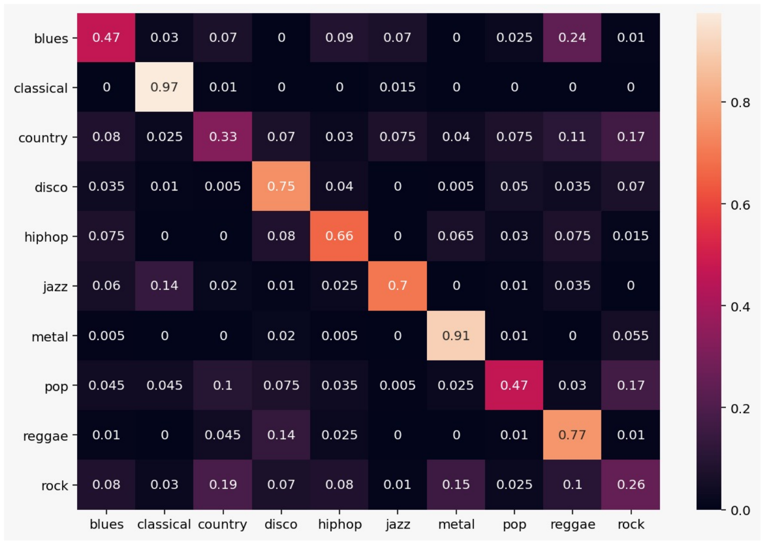


3 Experimental results

The data was split into a training set (80%) and a test set (20%). When working with the original, smaller dataset, the split was pseudo-random. But when working with the large dataset, we wanted to avoid an instance where different parts of the same song appear in both the training and test sets, since that would corrupt the integrity of the test set: the same exact verse of a song can recur more than once in a 30-second time frame. As such, we opted to split the data into predefined training and test sets which contain totally different songs from one another. When working with the small dataset, we measured the accuracy of our model on the test set directly (the percentage of correct classifications out of the entire set). However, working with the large dataset allowed us to take advantage of the fact that each song has 10 individual samples and use a "Majority-Vote" system to better classify each song (rather than each 3-second sample). The same goes for every unseen instance: our model will split it into ten 3-second slices an classify it using the majority vote. When testing our model on the large dataset, we used both methods (direct accuracy calculation for the entire set of 3-second samples, and the majority vote), and found that the results reached with the majority vote method were 5%-7% better.

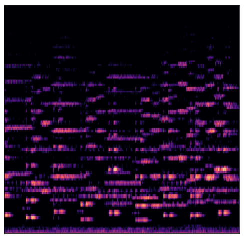
4 Discussion

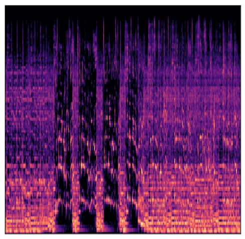
Below is the Confusion Matrix of our model, where the cell positioned in the i^{th} row and j^{th} column represents the proportion of 3-second instances with label i which our model predicted as label j:

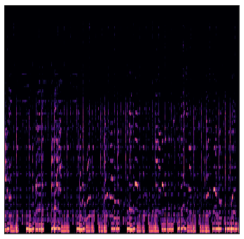


	blues	classical	country	disco	hiphop	jazz	metal	pop	reggae	rock
blues	0.47	0.03	0.07	0	0.09	0.07	0	0.025	0.24	0.01
classical	0	0.97	0.01	0	0	0.015	0	0	0	0
country	0.08	0.025	0.33	0.07	0.03	0.075	0.04	0.075	0.11	0.17
disco	0.035	0.01	0.005	0.75	0.04	0	0.005	0.05	0.035	0.07
hiphop	0.075	0	0	0.08	0.66	0	0.065	0.03	0.075	0.015
jazz	0.06	0.14	0.02	0.01	0.025	0.7	0	0.01	0.035	0
metal	0.005	0	0	0.02	0.005	0	0.91	0.01	0	0.055
pop	0.045	0.045	0.1	0.075	0.035	0.005	0.025	0.47	0.03	0.17
reggae	0.01	0	0.045	0.14	0.025	0	0	0.01	0.77	0.01
rock	0.08	0.03	0.19	0.07	0.08	0.01	0.15	0.025	0.1	0.26

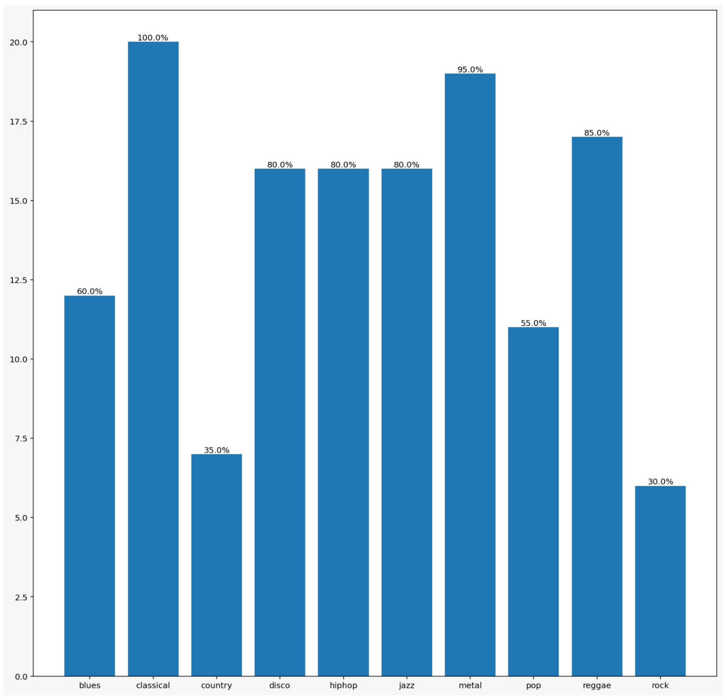
These results are not surprising. The dominance of Classical, Reggae and Metal can be explained by considering how unique each of these genres is compared to the others. While one can feasibly mistake Country for Rock, for example, it is easier to recognize one of the three dominant genres correctly. The uniqueness of these genres can even be seen in the Mel Spectrograms:


(a) ”The Four Seasons, Op. 8 No.1” by Vivaldi


(b) ”The Trooper” by Iron Maiden


(c) ”Three Little Birds” by Bob Marley

Even for the untrained eye, the smoother, cleaner notes of Classical music, the erratic drumming and false-chords of Metal and the ”staccato” offbeat chords of Reggae, each unique to its respective genre, are visible in these spectrograms. When looking at the performance of our model (with the majority-vote method) on each genre individually, we see the following results:



We can see clear improvement. Looking at the main diagonal of our confusion matrix, and comparing it to the bar graph we can see that the rate of True-Positives for each and every genre has experienced an improvement of anywhere from 2% to 13%. So even though the increase in data size did not contribute to the model's accuracy when using the direct accuracy calculation, the ability to use the majority-vote method was a big advantage.

When examining the genres which did not have very high results, we can see that the model mostly mistook one genre to a musically-similar one. For example, many Rock songs were identified as Country, many Blues songs were classified as Jazz, and vice versa. In one of our experiments, we eliminated the Country genre from our dataset (and our model), and achieved a 5-7% increase in performance overall, with the success rate for both Rock and Blues increasing substantially. Both these genres were often mixed up with Country in the original tests. In the end we achieved 75.5% accuracy over the large dataset when using majority vote.

5 Code

GitHub Repository - includes all of our code.
Colab notebook - includes all our models, works on the small dataset
Colab notebook - includes our final model, works on the large dataset.

References

1. "GTZAN" Dataset:
https://www.kaggle.com/andradaolteanu/gtzan-dataset-music-genre-classification

2. Understanding the Mel Spectrogram:
https://medium.com/analytics-vidhya/understanding-the-mel-spectrogram-fca2afa2ce53

3. Getting to Know the Mel Spectrogram:
https://towardsdatascience.com/getting-to-know-the-mel-spectrogram-31bca3e2d9d0

4. Audio Classification with PyTorch Ecosystem Tools:
https://clear.ml/blog/audio-classification-with-pytorchs-ecosystem-tools/

5. Finding the genre of a song with Deep Learning — A.I. Odyssey part. 1:
https://medium.com/@juliendespois/finding-the-genre-of-a-song-with-deep-learning-da8f59a611