Aidan O'Beirne
Alvaro Zamora

Abstract

Classification and feature extraction algorithms are a valuable subset of the functions deep neural networks can perform. Music genre classification is an application of these algorithms that can help search engines and similar platforms, as well as users, more efficiently organize music libraries. In this work, we construct a deep CNN that is able to classify short audio clips from songs into one of 10 genres (Classical, Jazz, Metal, Pop, Country, Blues, Disco, Metal, Rock, Reggae and Hip-Hop) with 90% accuracy. Songs from the publicly available GZTAN dataset are cut into short clips and are preprocessed into Mel spectrograms, which serve as the input to the CNN. We analyze the performance of the network and discuss how we will improve the model's performance even further.

Introduction

The objective of this project is to create and train a deep neural network to classify mp3 files into the following categories: Classical, Jazz, Metal, Pop, Country, Blues, Disco, Metal, Rock, Reggae and Hip-Hop (list from Feng 2004). Classifying music into genres is a task well-suited for deep learning; labeled data is readily available and the goal is to label new data into a defined set of classes. Most people define the genre of a song based on the "style" of the song. The "style" of a song is a nebulous concept and there are many different quantitative metrics that can define this word. In this project we define the style of the song as the frequencies that compose the time domain audio signal and their time evolution (frequency contours). An effective way to represent this information is with a spectrogram, which is a two dimensional visual representation of the frequency changes in time. Because the spectrograms are images, we used a convolutional neural net (CNN) architecture. The input to our CNN is a 128 by 128 spectrogram made from a clip of an mp3 file. After 4 convolutional and 1 fully connected layer, the CNN's final softmax layer classifies the clip.

Related Work

The first point of reference for this project was Tao Feng's work (2014) . In this paper, Feng compares deep vs shallow networks on the music genre classification problem. Feng finds that deep neural networks are more accurate on the test set only when 4 or more genres are included in the last softmax layer. Feng used Mel-frequency cepstral coefficients (MFCC), and achieves 60% accuracy with this technique when classifying 4 different genres.

Our second reference was Chang-Hsing Lee et al.'s work (2007). They use a novel feature that they call octave-based modulation spectral contrast (OMSC). The spectral contrast in OMSC refers to the difference between the spectral peaks and valleys, i.e. the difference between the harmonics and noise in a spectral distribution. By analyzing the long term modulation of these spectral differences, Chang-Hsing Lee et al. use a nearest neighbor classifier to classify 7 different genres up to 84.03% accuracy based on the OMSC features. This is an impressive accuracy considering the fact that they did not employ a deep neural net. This suggests that the OMSC features accurately represent genre information. However, the OMSC features rely upon the long-range order in the music audio signal and because a human can typically identify the genre of a song within 3 seconds, we wanted our neural net to do so as well.

Our third reference point was Cyril et al.'s work (2007). They use several standard classification algorithms (e.g. k-nearest neighbors, SVMs) to classify the emotion of a song into 4 different categories. Cyril et al. focus on a wide variety of audio features such as MFCC's for timbral features, tempo for rhythmic features, and harmonic pitch class profiles for tonal features. Cyril et al. achieve up to 98.1 %

accuracy for the "angry" category and as low as 81.5% accuracy for the "happy" category. This paper contains an exceptional amount of information, and we used it as a general resource when brainstorming which features to use. We decided that as a first iteration, it is best to minimize the number of features we include in our input data. In the long term, however, we may add more of the features described in detail by Cyril et al.

Igor Karpov (2002) attempted using a continuous multivariate hidden Markov model to classify music into genres. We used this reference because Igor discusses several methods of extracting features from music including MFCC's, delta and acceleration, and linear predictive cepstra (LPC). Igor shows that using LPC input significantly improves classification performance. Peter Ahrendt et al. (2004) has similar findings, and also discusses using the zero-crossing rate and MPEG-7 features as input.

Dataset and Features

We preprocessed the 30 second long GZTAN audio files into spectrograms of 3 second long files using the librosa python package. Figure 1 shows how each spectrogram is generated. The resulting spectrogram describes the frequency contours as a function of time. Mathematically speaking, the spectrogram represents the time evolution of the 128 amplitudes in the Fourier basis in steps of 20 ms: $a_i(\Delta t * n)$, where n is an integer from 0 to 128, Δt is 20 ms, and i denotes the coefficient in front of the ith sinusoid associated with one of the 128 Mel frequencies. This is precisely the information that we claimed represents the "style" of a song in the introduction.

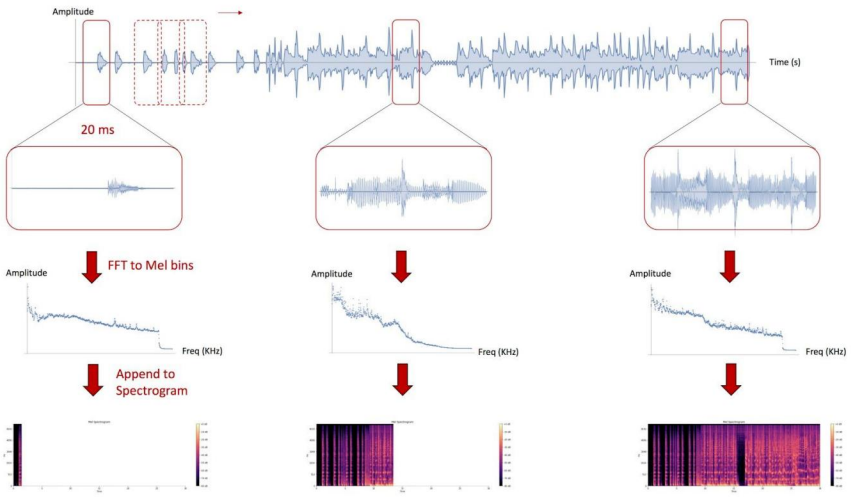

Figure 1: Preprocessing the 3 second long audio time domain signal (top) into the associated mel spectrogram (bottom right). The raw signal is sliced by a 20 ms window (red box). The slice is Fourier transformed and discretized into mel frequencies (middle row). The resulting spectrum is appended to the associated time step in the spectrogram (bottom row). This process is repeated for each 20 ms window with a stride of 15 ms over the entire 3 second time domain signal.

In most applications, the Mel frequencies are further processed into Mel-frequency cepstral coefficients (MFCC). This is usually done in an attempt to decorrelate the input data to avoid overfitting. However, we decided to not use MFCCs for two reasons. First, we wanted to keep as much information about the audio signal as possible, and the discrete cosine transform eliminates information. Second, standard regularization techniques such as dropout, L2 weight decay, and maxnorm should suffice.

Before training the CNN, we decided to visualize our mel spectrogram dataset using the t-SNE algorithm, which reduces the dimensionality of the input data's parameter space while preserving the similarity between examples. We see that Jazz, Classical, and Blues form a similarity group that is very different from the group formed by Disco, Country, and Pop. Metal seems to be distributed into two different clusters; one cluster is grouped with Rock while the other cluster is grouped with Disco and Country. The t-SNE plot is an interesting way to visualize the data, and it can help us better understand the distribution of examples that the neural net incorrectly classifies.

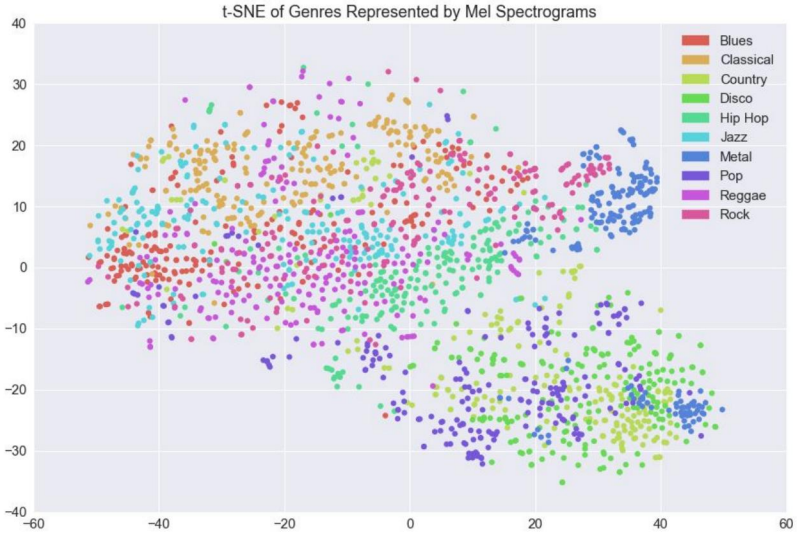

Figure 2: A t-SNE representation of the 10 different genres as represented by mel spectrograms. Jazz, Classical, and Blues form a similarity group and Disco, Country, and Pop form a similarity group.

Methods

We used a categorical cross entropy loss to minimize the distance between the CNN output distribution, y, and the correct label, $\hat{y}$, with an Adam optimizer. Adam's momentum algorithm uses an exponentially weighted average of the derivatives of the weights to modulate the learning rate hyperparameter during optimization. Similarly, Adam's RMSprop uses an exponentially weighted average of the squares of the derivatives of the weights to modulate the learning rate. Both of these algorithms orthogonalize the optimization of the weights, w, and the bias, b. Due to these benefits, the Adam optimizer reduces oscillations in the parameter space during training and reduces training time.

We created the 6 layer CNN architecture depicted in Figure 3. The CNN accepted square inputs of 128 frequencies by 128 time steps. Each convolution was a 2x2 filter with strides of (1,2,1,2) corresponding to layers (1,2,3,4). We selected the layer dimensions (Figure 3) based on previous work (Despois 2016). Reducing the convolutional stride size from 2 to 1 seemed to improve dev/test set accuracy while also significantly increasing training time. As a compromise, we decided to only use a stride of 1 in layers 1 and 3. This resulted in a comparable increase in dev/test accuracy as using all strides of 1, while taking about a third of the time. Each maxpool is a 2x2 filter with a stride of 1. We initialized all weights in the CNN using Xavier initialization. We used a Relu activation output for all layers except for the final softmax layer. The CNN was trained with L2 regularization to prevent training set overfitting.

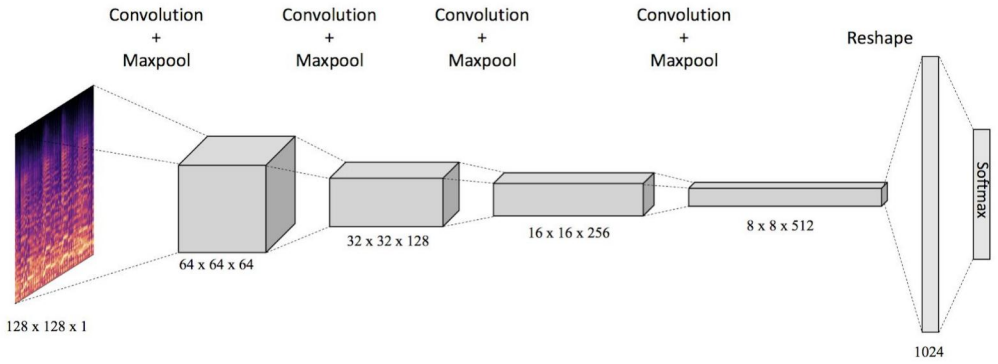

Figure 3: The CNN architecture. The net consists of 4 convolutions followed by a fully connected and softmax layer.

Results

We used a grid search to optimize the learning rate and weight decay parameters. We found that $\alpha = 0.001$ and $\lambda = 0.0019$ maximized the dev set accuracy. We trained the CNN for 10 epochs using a minibatch size of 512, which we chose because it was harder to compare the performance of different model parameters mid-training with smaller batch sizes (larger oscillation). The trained net performs at **90% test set accuracy** and **91% dev set accuracy**, which are our two primary performance metrics. The trained net exhibits some variance, as the train set accuracy is 99%. This discrepancy is likely due to having a small dataset. Overfitting is a possibility, but standard regularization techniques did not help to reduce variance. Adding dropout to the convolutional layers only decreased the dev/test set accuracy. Increasing L2 regularization beyond 0.0019 also decreased performance. Finally, we attempted maxnorm regularization but this also worsened performance. This suggests that either the dataset was too small or that our network was not large enough to capture the complexity of the parameter-space we tried to model.

The model had the most difficulty classifying Country and Rock (77% and 75% test set accuracy, respectively). It easily classified Jazz and Classical (100% test set accuracy). Examining the t-SNE diagram shown previously, we see that the genres we chose to classify are mixed. This helps us understand which genres will be harder to classify and brings about the idea of Bayes error being different for each genre. Some genres like Country are distinct and are easy to classify while others like Rock and Blues can be audibly similar during a 3 second clip and may be harder to classify. A confusion matrix is shown in Figure 4, which makes it clearer what genres were harder to classify and which may be similar in our dataset.

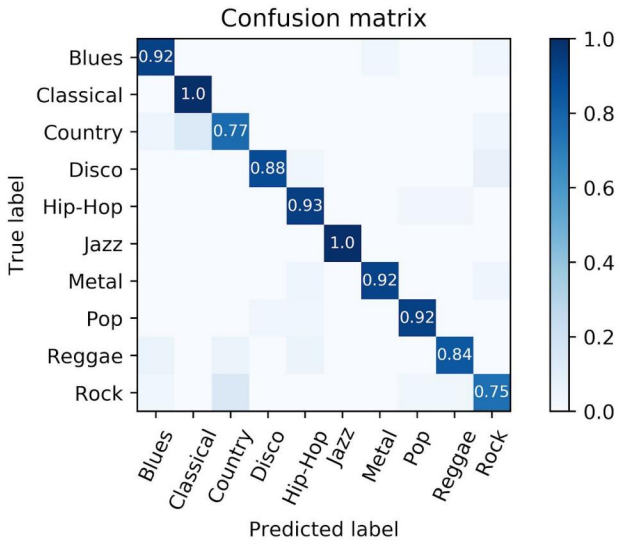

Figure 4: Confusion Matrix. The Classical and Jazz samples in the test set were classified with 100% accuracy, while Rock was only classified with 75% accuracy.

Conclusion & Future Work

We built a CNN that is able to classify music into 10 genres with 90% accuracy. We identified which genres the net has trouble classifying and have a clear plan for how to improve performance. We can expand the dataset to include the publically available FMA dataset with over 100,000 labeled songs. This will help sample the underlying distribution much better to obtain a better model. The CNN performance simply kept improving with more layers and larger filter output depths, so expanding the net to include either 1-2 more convolutional layers or using larger output volumes in each layer is a promising way to improve performance. This will help the net better model the large and complex musical parameter space. Both of these improvements would require more computational resources than those provided by CS230 on AWS. Adding other forms of input such as linear predictive ceptra may improve performance at a low computational cost.

As a long term goal (after this project), we intend to investigate music genre style transfer. One method of doing this would be to use standard neural style transfer, minimizing the difference between the style (genre) of two spectrograms while keeping the content the same, which would generate a spectrogram. After performing style transfer on the spectrogram, it will be necessary to invert the spectrogram back to an audible signal. This is a nontrivial task since the Fourier transform removes phase information. One inversion method is to use a Fast Griffin Lim algorithm as proposed by Perraudin et al. This algorithm works by minimizing the mean squared error between the STFT of the recreated signal and a modified STFT. By iteratively projecting the recreated signal into the time domain and frequency domain and minimizing the mean squared error, a signal can be produced from an FFT.

Contributions

- Preprocessing: Aidan researched and laid out the methodology; Alvaro wrote up the preprocessing code.
- Building the Neural Net: Both members worked together to research what architecture would work best and to build the neural net.
- Training & Tuning: Aidan primarily worked on hyperparameter tuning; both members worked on training and small architecture changes.
- Analysis & Writing: Alvaro primarily did error analysis; both members worked on interpreting results and writing for write-ups/poster.

Link to Repository: https://github.com/aidanobeirne/CS230-1-Project

References

1. Tao Feng. Deep learning for music genre classification
2. Lee, Chang-Hsing, et al. "Automatic music genre classification using modulation spectral contrast feature." Multimedia and Expo, 2007 IEEE International Conference on. IEEE, 2007.
3. Laurier, Cyril, et al. "Audio music mood classification using support vector machine." MIREX task on Audio Mood Classification (2007)
4. Karpov, Igor, and Devika Subramanian. "Hidden Markov classification for musical genres." Course Project (2002).
5. Despois, Julien. 2016. Finding the genre of a song with Deep Learning
6. Perraudin, Nathanaël, et al. (2013). A Fast Griffin-Lim Algorithm. 2013 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics ,
7. Griffin, Daniel and Lim, Jae. (1984). Signal Estimation from Modified Short-Time Fourier Transform
8. Ahrendt, Peter. 2015. Decision time horizon for music genre classification using short time features.