Machine Learning: Music Genre Classification Using Deep Learning Convolutional Neural Nets

Jackson Dean, Nic Koch-Gallup and Sean Cheng

Abstract

Music genre classification is important for online music libraries to be able to sort and recommend songs to users. This paper is not only relevant for music, but also can provide useful insights into the promising possibilities of machine learning classification in other aspects of life. In this paper, we create a convolutional neural network that classifies music into 10 different genres. We also compare its classification ability with a long short-term memory network that we build. For our training data, we use the GTZAN Genre Collection with 1000 genre-labeled songs. We use the Fourier transform algorithm to convert the audio files into spectrograms so that the networks can analyze the data. We were able to achieve 88.90% accuracy using an LSTM network, and achieved 99.88% accuracy using a CNN. These high-accuracy results show that machine learning is viable method for classifying music into genres. As predicted and seen in other papers, the CNN worked best for this task and dataset.

1. Introduction

Music has always been a way for people to express themselves in and across cultures. Nowadays, we have the necessary technology to record songs so that we can listen to them over and over again whenever we feel like it. All we have to do is access an online music library (e.g. Spotify, Apple Music, Soundcloud, YouTube, etc.) and search for whatever we want to listen to. One of the most common search categories that people like to use is by genre. Music within the same genre is characterized by their similarities in style, content and overall feel. Many people have a favorite genre and rarely venture outside of it, while others like to pick a genre that is representative of their current mood. However, in order for a music library to successfully filter songs according to the genre being searched for, the songs must be categorized and labeled with their respective genres. In this paper, we discuss how we built a music genre classifier that can make accurate predictions of which genre a song belongs to. This tool will help classify uncategorized songs and label them for future reference, allowing search filters to correctly filter these songs by genre. Before we explain our music genre classifier, it is helpful to look at previous research and methods of classification.

2. Background

2.1 Literature Review

Various approaches have been taken to building music genre classifiers with varying amounts of success. Many of these approaches we had touched on in class, such as k-nearest neighbors and naive bayes machine learning algorithms, as well as the Weka and SciKit Learn software.

In Bahuleyan's (2018) article, he explored the use of logistic regression, random forests, gradient boosting and support vector machines. In the end, he found that deep learning neural nets found much more accurate results. He was able to achieve 64% accuracy using a CNN; this research supported our decision to use a CNN for this project. His network differed from ours in that it uses pre-trained weights and uses fine-tuning weights to train it.

With WEKA, Basili et al. (2004) examined MIDI data (instead of using Fourier transform) using Naive Bayes, Voting Feature Intervals, J48, Nearest Neighbor ge, and JRip to classify music into genres. The main takeaway from their study is that musical features from data can provide effective information for genre classification. Their Naive Bayes classifier got the highest accuracy with around 65%, and the rest got between 57-62%. This research used MIDI data and many learning algorithms, while our project used spectrograms and a CNN.

Silla et al. (2008) uses multiple feature vectors and a pattern recognition ensemble approach. They found that using a support vector machine classifier and space-time decomposition allowed them to obtain the best accuracy. The SVM achieved 65.06% accuracy. Similar to our research, they used the GTZAN song database.

Haggblade et al. (2011) looked at a number of different machine learning algorithms, including "K-nearest neighbors (KNN), k-means, multi-class SVM, and neural networks" to classify four music genres. Interestingly, they relied very heavily on using Mel Frequency Cepstral Coefficients, which we had not heard of before reading this paper. It sounds like these coefficients may be a way to take our data from the Fourier's transform and make it even more helpful to our classifier because the MFCCs are based on the human auditory system, which should theoretically be more accurate at predicting music genre. Their results were also very impressive, with 100% accuracy identifying some of the genres. They got the best results using neural networks, which supports some of the other research we have read and validates our decision to implement a CNN.

3. Methods

The methodology for this task was split into two main sections: preprocessing the data and training/testing the data. For clarity, these two functionalities were split into two separate Python files. The code to build the two types of models was split into a third Python file. We used an object-oriented approach, with an object for our application that implemented the classes from our imported libraries.

3.1 Tools and Data

We coded the project in Python using the TensorFlow module. We were running various versions of Python, so we set up a virtual environment to work around TensorFlow compatibility issues.

The training data we used is the GTZAN Genre Collection which contains 1000, 30 second long, song audio tracks in .au format labeled under 10 genres (Blues, Classical, Country, Disco, HipHop, Jazz, Metal, Pop, Reggae, and Rock). From these data, we extracted 20% of the songs to save as test data, and used the other 80% of songs to train the neural net. This ensures that our test results are more accurate as we are not calculating an in-sample error by testing the net on the same data that was used to train it.

3.2 Fourier Transform

We chose to use the Fourier Transform algorithm to pre-process the data because deep learning convolutional neural networks are best suited to analyze images. This algorithm converts our 30-second audio files into a frequency domain spectrogram by taking the original signal and splitting it into its constituent frequencies. The basic idea behind the ingenious algorithm is the waveforms are viewed as a combination of circular paths. These cycles move at different speeds (Hz) and distances (amplitude), and by measuring those over time, we get the frequency. To assist with calculations, we used the librosa.core.stft library.

3.3 Data Pre-processing

In order for the program to accurately analyze songs, the music files have to be converted into a more program/data-friendly format where the program can analyze their sound waves. Using the Fourier Transform algorithm, we were be able to convert these audio files into Mel-Spectrograms, which are visual representations of frequencies, or graphs of the sound waves. After the sound data was converted from *.au files to spectrograms using Fourier's Transform, the data was processed further by applying Mel-frequency cepstral coefficients (MFCCs). MFCCs were used because they closely model the human auditory system by representing audio data as a non-linear spectrum of a spectrum. Both Bahuleyan (2018) and Haggblade et al. (2011)

followed this same data pre-processing procedure in their studies that also involved building a deep learning neural net.

We used spectrograms instead of other forms of data representation because of their natural compatibility with Convolutional Neural Networks. CNNs work on spectrograms as they would on any other image, by ‘sliding’ a filter over regions of the image in order to learn features. In this case, the ‘features’ of the spectrogram image can be thought of as representing features of the songs themselves. Previous research has shown that CNNs trained on spectrograms are more accurate than other types of networks trained on other data, such as hand-crafted feature models (Bahuleyan, 2018).

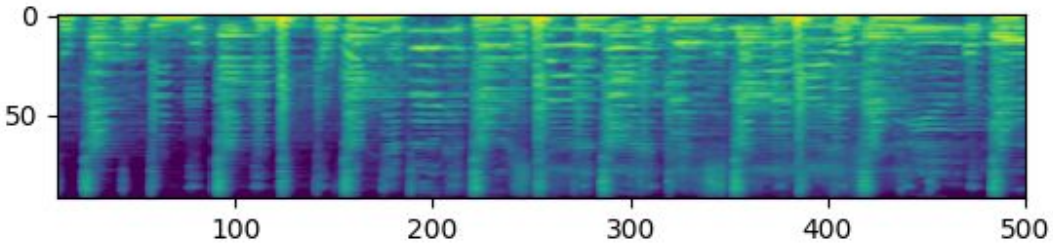

Figure 1: Spectrogram of a 30-second song (Bob Marley’s “Waiting in Vain”). Using Fourier’s transform and MFCC’s, visual data from the song was created. The spectrograms are 1366 X 96 pixels.

No data augmentation was used for our dataset because rather than trying to classify parts of an image, we are considering each song wholistically as a representation of frequencies over time.

Label data was fed to the networks in the form of arrays of size 10. Each index in the array corresponded with a music genre and was either 1 or 0. Values of 1 signified that there was a 100% chance that this song was the genre that corresponded with that index, while values of 0 corresponded with a 0% chance. An example array might look like: [0 0 0 0 1 0 0 0 0 0], indicating that the spectrogram labeled with this array was ‘disco’. The output layer of the networks will also be an array of size 10, where the values in the array correspond with the likelihood that the song is each genre.

4. Implementation

Our classifier uses a deep learning neural network to analyze the patterns in the spectrograms of a variety of song genres, through labeled training data. After processing the song files through the Fourier Transform. We used the TensorFlow library to create a convolutional neural network to analyze every pixel of the

spectrograms. Our program builds a network to analyze similarities in the spectrograms of songs from the same genre, and differentiate between spectrograms of songs from different music genres. As the program encounters these patterns, it is able to build clustering network that maps the patterns it finds to the genres of the labeled songs. Eventually, the neural network will have seen sufficient training data that it can begin to accurately predict what genre a song file belongs to by just reading its audio spectrogram. At this point, the program is ready to analyze the test data, which are songs with unlabeled but known genres. This allows us to test its accuracy in classifying the genre of the songs by comparing the programs predictions of genres of songs to the actual song genres.

4.1 Neural Networks Background

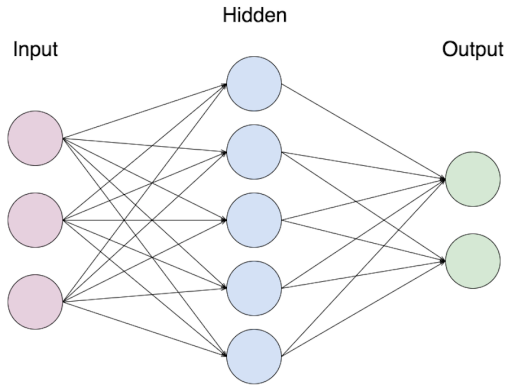

Figure 2: Basic layout of a neural network. Here we see input nodes connected to nodes in the hidden layer, which connects to the output nodes in the output layer (Tine, 2017).

In class, we discussed the history and structure of neural networks. Neural networks contain interconnected nodes that have interesting and powerful effects, and can “learn” data. Neural networks have an input layer, hidden layer(s), and an output layer (see Figure 2). Forward propagation involves giving weights to the input data. Backpropagation calculates the error and adjusts the weights to reduce the error. So, the input layer provides inputs with weights to the hidden nodes. The hidden nodes sum the input connections. Then the activation function is applied to the sums, providing us with an output signal.

In our project, we use a powerful version of a neural network, called the convolutional neural network (CNN). CNNs are a type of deep learning network that does not contain cycles (feed-forward). CNNs use convolution which takes the weighted sum of a small region of an image, and it is repeated many times to cover the image. CNNs also use three dimensions: depth, height, and width (see Figure 3). Two

dimensions are for the x-y size of the image and the third dimension is for the RGB colorspace.

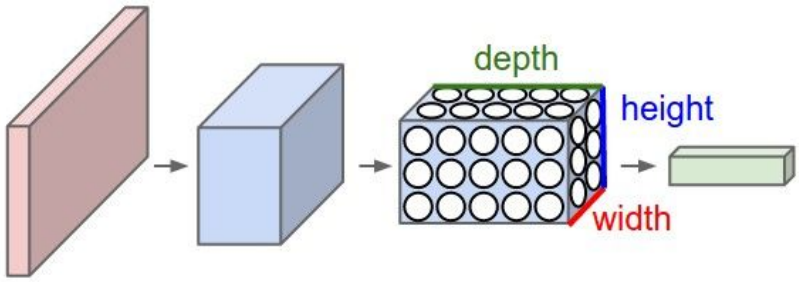

Figure 3: Basic layout of a convolutional neural network. The image depicts the input layer, two hidden layers, and an output layer (Karpathy, n.d.)

In the hidden layers of the CNN, the network performs convolution and pooling (downsampling). The convolution step is where the network extracts features from the input. In the pooling step, it takes the data and reduces it, which helps with speed and memory, and it keeps the important information. Finally, the fully connected layer classifies the features from the hidden layers. To reiterate, convolution and pooling are part of hidden layers and are important for feature detection/extraction; the fully connected layer functions as a classifier. CNNs also use backpropagation to train. For our project, we use a CNN on spectrograms from music data in order to classify music genres; the following section discusses the implementation of it in detail.

4.2 CNN Implementation

Originally, the CNN was not learning, and consistently reached 12.5% accuracy and stopped learning. After hours and hours and various attempts, we finally got the network to improve. We figured out that we just did not have enough layers and did not test enough epochs at first. After experimenting with additional layers, we ended up creating a functional network. We describe the network below, and provide an illustration of it in Figure 4.

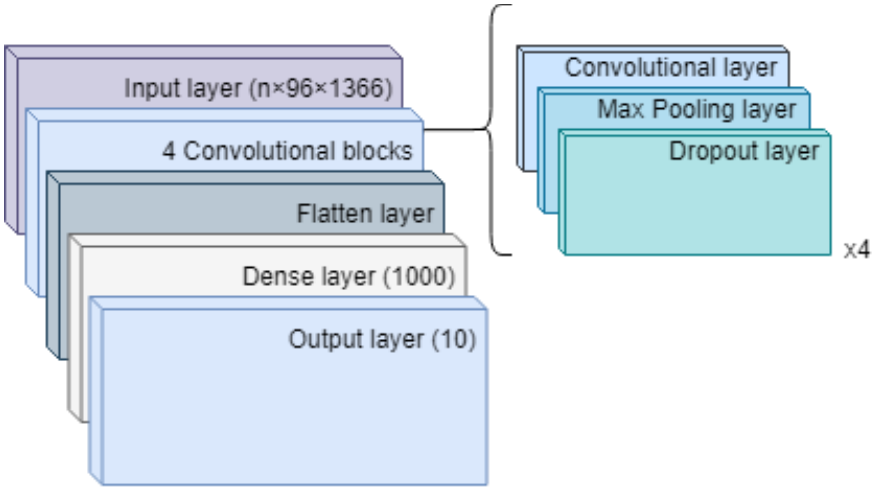

Figure 4: Framework of our convolutional neural network. The network includes an input layer, four convolutional blocks with three layers, a flatten layer, a dense layer, and an output layer.

Our CNN has an input layer, four convolutional blocks, a flatten layer, a dense layer, and an output layer. The input layer takes in the spectrogram data. Our four convolutional blocks each consist of a convolutional layer, a max pooling layer, and a dropout layer.

In the first convolutional block, the convolutional layer had 32 filters, a kernel size of 4x4, made strides of 4x4, and used ReLU activation. The input image was padded using “same” padding so that the output size remains the same as the input size. For convolutional layers in the other three blocks, parameters stayed the same except that there were 64 filters per layer.

The maxpooling layers have a pool shape of 2x2. This is a typical size for maxpooling pools in CNN for image classification tasks. As the size of the input gets larger, it makes sense to increase the size of the pools, especially in lower layers. If we were to continue working on this net, we might try to increase this to 4x4 as our input is fairly large (1366x96).

The dropout layers have a rate of 50% (increased from 20% in our previous model) because we were worried about overfitting given our small dataset. We found that this successfully reduced overfitting but may have contributed to a slightly lower accuracy rate as well.

The flatten layer flattens the feature maps (arrays) so that they can be inserted into the dense layer. The dense layer is the fully connected layer in which all nodes are connected to every node in the previous layer.

We compiled our network and used TensorFlow’s Adam optimizer a learning rate of 0.0003. We used categorical cross entropy for our loss function. We ran the network with a batch size of 400 and 10 epochs. This learning rate proved to be too small as our

loss was not decreasing as much as we would have liked. We ended up increasing the learning rate to 0.003 and found that the loss converged much more quickly.

We decided to increase the number of filters in each layer for this network based on previous research that shows CNN's have more accuracy on this task given more filters (Pons & Serra, 2018). Although the research shows that 512 filters per layer would have been optimal, we were limited by computing power and eventually settled on 32 or 64 filters per convolutional layer.

Overall, our final CNN was extremely successful at classifying the songs into their respective genres. The final accuracy for the CNN on the test data was 99.88%. See Figure 5 for a graph of loss and accuracy over epochs.

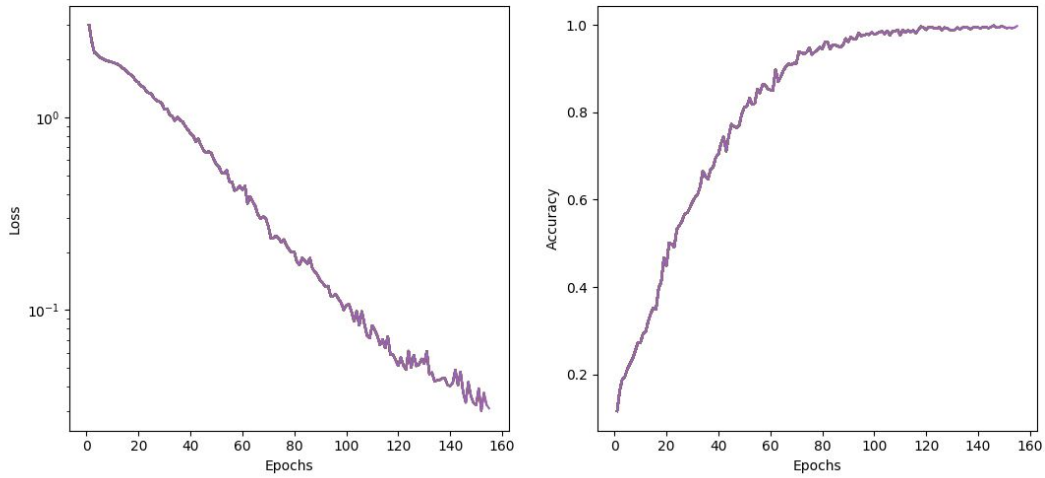

Figure 5: Loss decreasing over 160 epochs (left). Accuracy increasing over 160 epochs (right). In this figure, we see loss being reduced to 0.05, and accuracy increasing to 99.88%.

4.3 Long short-term memory (LSTM) network Implementation

In order to compare the CNN's effectiveness, it is useful to build another type of network to classify the data. LSTM networks are a type of recurrent neural network (RNN). RNNs and LSTMs are unique in that they reuse the outputs. RNNs can contain LSTM units, but if a network is solely made up of LSTM units, it is usually called an LSTM network.

Our LSTM has four layers. The first layer is the input layer, which took in the spectrogram data. The second layer is an LSTM layer with 128 units. This layer has 0.05 dropout, 0.35 recurrent dropout, return sequences enabled, and an input shape of 96x1366. The third layer is an LSTM layer with 32 units, 0.05 dropout, 0.35 recurrent dropout, return sequences disabled. The fourth layer is a dense output layer with 10

units for the 10 categories and softmax activation. We compiled this network with a categorical cross entropy loss function and Keras' default Adam optimizer.

The LSTM network outperformed our original CNN that was only achieving 12.5%. The LSTM achieved an 88.90% accuracy over 1000 epochs (see Figure 6). Although this accuracy is not as high as the CNN accuracy, it did significantly better than the other learning methods used in past research (see Section 2.1 for other method accuracies). The LSTM also served as a good comparison for the CNN. This shows that CNNs are likely the best options for music genre classification.

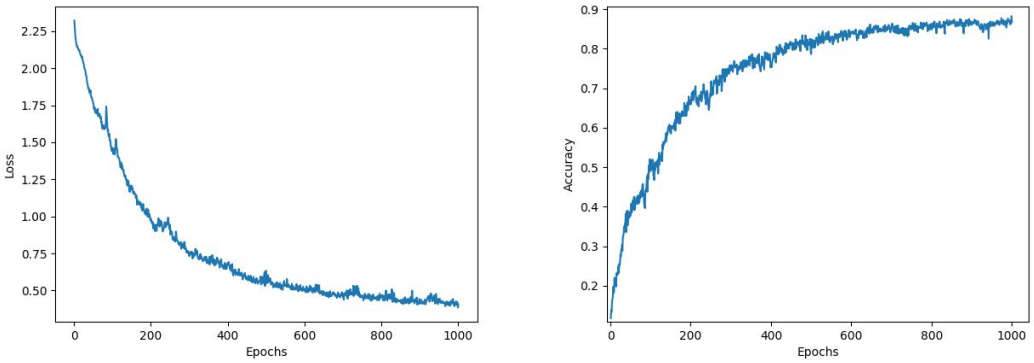

Figure 6: Loss decreasing over 1000 epochs (left). Accuracy increasing over 1000 epochs (right). In this figure, we see loss being reduced to 0.40, and accuracy increasing to 88.90%.

4.4 GUI

We designed a simple user interface program to allow the user to choose the type of model to build (CNN or LSTM). We also wanted users to be able to choose whether to train the model on the dataset or use pre-trained weights to predict the genre of a novel song. We added live graphing functionality to show the progression of loss and accuracy over time.

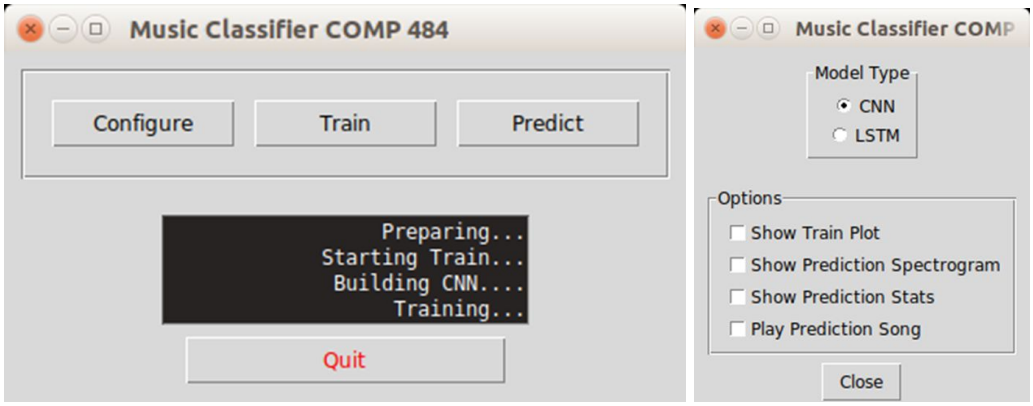

Figure 7: Music classifier GUI.

5. Conclusion

In this paper, we described in detail our work to build a music genre classifier. We covered all the steps we took to research how to create our deep learning neural net, what tools and resources we needed to set up our algorithm, and how to train the neural net to achieve the best results. We found six studies that combined for over 10 different machine learning techniques used to build the same classification software as we were. Once we had our dataset, we used Fourier Transform and MFCCs to preprocess the data for training. At first, we struggled significantly to get our CNN running properly using Tensorflow, in which span, we tried several combinations of layers and filters. We additionally built an LSTM network to compare its findings with our CNN network. After getting the LSTM working, we went back and added many additional layers to the CNN. It finally worked, and we obtained great accuracy!

We were ultimately able to attain a 99.88% accuracy in our final CNN model after 160 epochs, while our LSTM network only reached a 88.9% accuracy. From previous work, we expected that our CNN would outperform the LSTM network as all our sources claimed that their deep learning neural nets produced the most accurate classifications.

Music genres are not clear cut categories, and we were consciously avoiding overfitting our model to the data. We became suspicious of the sources that cited really high classification accuracies, and wondered how well they would do on a different dataset. When we achieved a similar level of accuracy, those overfitting suspicions evaporated as we took several measure to prevent overfitting from using a separate training and test data set to adding a 50% dropout layer.

Future directions of research include getting the network to generalize to new songs. We also could use the network and spectrograms to be able to suggest related songs to users.

Works Cited

Bahuleyan, Hareesh. “Music Genre Classification Using Machine Learning Techniques.” *arXiv:1804.01149 [cs, Eess]*, April 3, 2018. http://arxiv.org/abs/1804.01149.

Basili, Roberto, Alfredo Serafini, and Armando Stellato. “CLASSIFICATION OF MUSICAL GENRE: A MACHINE LEARNING APPROACH.” Accessed October 2, 2018. http://art.uniroma2.it/research/musicIR/BasSeraStel_ISMIR04.pdf.

Creme, Matthew, Charles Burlin, and Raphael Lenain. “Music Genre Classification.” Accessed October 2, 2018. http://cs229.stanford.edu/proj2016/report/BurlinCremeLenain-MusicGenreClassification-report.pdf.

Haggblade, Michael, Yang Hong, and Kenny Kao. “Music Genre Classification,” 2011.

Logan, Beth. “Mel Frequency Cepstral Coefficients for Music Modeling.” *Proc. 1st Int. Symposium Music Information Retrieval*, November 24, 2000.

Karpathy, A. (n.d.). Convolutional Neural Networks for Visual Recognition. Retrieved from http://cs231n.github.io/convolutional-networks/

Pons, Jordi & Serra, Xavier. (2018). Randomly weighted CNNs for (music) audio classification.

Silla, Carlos N., Alessandro L. Koerich, and Celso A. A. Kaestner. “A Machine Learning Approach to Automatic Music Genre Classification.” *Journal of the Brazilian Computer Society* 14, no. 3 (September 1, 2008): 7–18. https://doi.org/10.1007/BF03192561.

“TensorFlow.” TensorFlow. Accessed October 2, 2018. https://www.tensorflow.org/.

Tine. (2017, September 21). Neural Networks in Javascript. Retrieved from https://blog.webkid.io/neural-networks-in-javascript/