Classical Music Analyzer using Deep Learning Architectures
CS 6780 Final Project

Brian Rappaport and DB Lee

May 10, 2019

1. Introduction

Recent developments in artificial intelligence have proved its capabilities in machine perception. In the field of audio signal processing, machine learning has been used for speech recognition (Padmanabhan, Jayashree & Jose Johnson Premkumar, Melvin, 2015), signal separation (Wang, DeLiang & Jitong Chen, 2017), music clustering (Cilibrasi et al., 2003), and many other applications. In particular, services such as Amazon Echo or Shazam are familiar to the public for their ability to understand verbal commands or identify the song from a short clip recorded through a phone microphone. However, these tools do not generalize well to all uses. In classical music, most pieces do not include lyrics, so systems attuned to human speech will not extend readily into instrumental genres. Much of modern music identification software uses hashing to encode songs and stores them to be queried later, a technique closer to memorization than true learning.

By listening to a clip of music, many humans can almost instantly determine the genre: country music from metal, for example. The same is true of the variety of styles of classical music. Moreover, a trained listener can often identify the composer of a piece as well, by their distinctive style within the time period: although Debussy and Ravel are both Impressionist composers, it is certainly possible to identify disparate characteristics of both composers' music and distinguish them. After sufficient exposure to different composers' inherent styles, a listener may identify the composer without knowing the piece itself. In this project, we aim to create a similar system that can learn general patterns of composers only from the audio recordings of various compositions.

2. Problem Setup

This problem has a strongly hierarchical structure: composers can be distinguished by time period (generally speaking, Renaissance, Baroque, Classical, Romantic, and Modern, with many sub-disciplines within these), and different pieces can be distinguished by composer (and in fact different interpretations of the same piece can sound different, too; a trained listener can distinguish between a Bernstein recording and a Toscanini recording, to give two well-known conductors). Here, we consider the simplest case: a binary classifier which distinguishes two composers representative of different styles, here Johann Sebastian Bach and Claude Debussy. Compositions by these two composers show little overlap in compositional techniques: the Baroque music style as shown by Bach is well known for its established tonal theories and the associated rules in composing music, while Debussy spearheaded the Impressionist movement (a subset of the Romantic period), introducing new musical idioms considered inappropriate in the Baroque times. The compositions of these two composers exhibit distinctive patterns in their use of intervals, the pitches that are allowed to coexist at a given instant, and voice leading, the sequence of pitches played adjacently in time. Moreover, the two composers are played with different timbres: Baroque music is (generally speaking) played very lightly and dance-like, while Impressionist music is often characterized by swirling patterns, much like the artwork the movement takes its name from.

3. Description

3.1. Data Processing

Our data consists of modern recordings of different pieces by these two composers, freely available from YouTube. We split each recording into many fragments with a standard (variable) length, each of which forms a single labeled example. We use the spectrogram of each example as our feature: taking short-time Fourier transforms (STFT) of a set of overlapping windows of the overall waveform to produce two-dimensional data with a time and a frequency axis, where the n^{th} time step represents the frequency domain representation of the audio sample at time n, and using the magnitude of the Fourier coefficients.

3.2. Models

The spectrogram representation of audio signals allows two possible interpretations of the data: as a time sequence of features where each feature is an STFT coefficient, and as

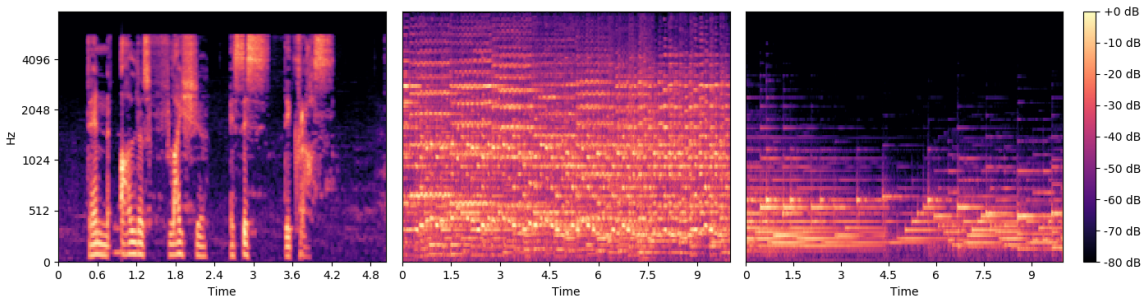

Figure 1. Spectograms of different types of audio: excerpts from speech, orchestral music, and a piano suite (from left to right). Spectrograms of speech contain curves and frequent pauses due to changes in vowel and closing consonants (T or D, say). On the other hand, spectrograms of music shows linear, discontinuous patterns that represent instruments sustaining a pitch and rapidly changing to another.

an monochromatic image. As a sequence, each frequency coefficient could represent a note in a chord, for instance, so in the ideal case where the timestep is equal to the length of each beat, each feature would be the chord at the corresponding beat. Musical signals are temporally correlated and also correlated in nearby frequencies: for instance, a long-held note would be represented as a line of high amplitude on a spectrogram; treating the data as an image (a spectrogram) allows us to capture these relationships.

As such, we consider two architectures for learning the classification, one for each representation. A standard way of classifying sequences is by using a long short-term memory (LSTM) network (Hochreiter & Schmidhuber, 1997). An LSTM is a form of recurrent neural network (RNN) which is good at remembering information, something which regular RNNs have trouble doing effectively. An LSTM takes in many sequences of time-series data and runs each one through a series of cells: each cell modulates the each step of the sequence by a ”forget gate” which determines which information should be remembered in the next step and which should be forgotten and updated, along with an update gate which adds the information for the next state; each of these gates are fully controlled by a series of learned weights and biases. Such recurrent models are used mostly in language processing, but there are reports of success in applying them to music as well. For example, a recurrent network showed success in predicting the next chord in a song given the previous sequence of chords (Korzeniowski et al., 2018). In our context, each small segment sampled from a given audio clip are treated equivalently as the ”words” in a time-series sequence, with the STFT frequency coefficients acting as their embeddings. The network predicts which frequencies will change over time quickly or slowly: for instance, a bass line would often change more slowly than a melody because bass lines are generally more sustained. In the Bach-Debussy comparison, this could be a way to detect the timbre of the different composers, as described earlier.

Using an image representation, it might make more sense to use a convolutional neural network (CNN) to classify the data. Audio signals in classical music exhibit both temporal and frequency locality. In image processing, CNNs use learned filters to capture high-order spatial information that a fully connected network would naïvely discard by flattening the image into a single vector (LeCun et al., 1998). Musical signals carry a similar high-order temporal information. An instrument will tend to produce the same sound it does in nearby time frames. This phenomenon is shown as horizontal line patterns in spectrograms of music. Some studies used this observation in distinguishing music from non-music (Bhattacharjee, Mrinmoy et al., 2018). In this project, we assume this temporal locality and experiment with convolution and max-pooling in the horizontal direction of the images. Previous studies on CNN audio classification utilize architectures such as VGGNet or AlexNet with a massive dataset such as the YouTube 70M (Hershey, S. and S. Chaudhuri and D. P. Ellis and J. F. Gemmeke and A. Jansen et. al, 2017). Unfortunately, these models, although incredibly successful, use large nested networks that require an exorbitant amount of computational resource. Our project instead experiments with the capabilities of a lightweight classifier with at most two convolutional and max-pooling layers. In the extreme case, we perform only max-pooling in the horizontal direction without explicitly learning filter values, in an attempt to fully utilize the locality assumption and to minimize learning time.

4. Results

We ran the LSTM code on a dataset of examples of music from the collected works of Bach and Debussy of all styles: both had pure orchestral pieces, but also included piano, organ, vocal, chamber music, opera, oratorio, and other solo instrumentals. The number of examples used varied on the interval chosen: we calculated a total of 2820 seconds of music, evenly split into intervals of between 1 and 10 sec-

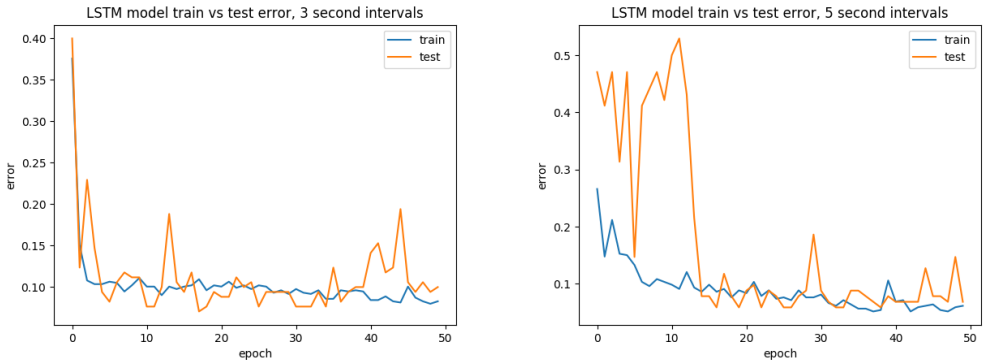

Figure 2. LSTM model train and test accuracy for 3 and 5 second intervals, respectively. Data was split into 10% test, 18% validation, and 82% training sets, with validation being compared to but not tested on. Overall accuracy on the test set was 92.6% for 3 second and 91.1% for 5 seconds.

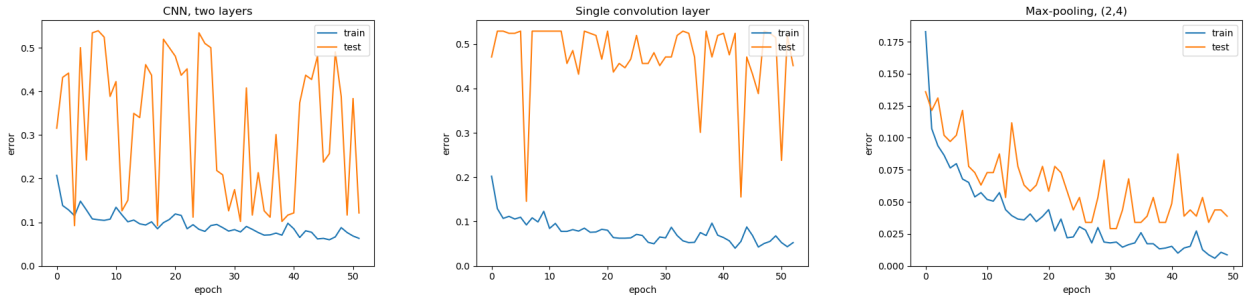

Figure 3. The CNN-based model train and test accuracy for 2, 1, and no convolution layers in the network. Using the same data set, overall test accuracy was 88.9% using two convolutional layers, 61.5% using one convolutional layer, and 95.8% using only max-pooling.

onds: when we used 5 seconds, we had 564 examples and when we used 3 seconds, we had 940 examples total. Each feature used a 256-point fft, downsampled by a factor of 2, and used 50-example batches and 50 epochs of training. Results were promising, getting above 90% for 2, 3, and 5 second intervals; the accuracy was maximal for 3 second intervals and not for higher values because as the intervals got longer the number of training examples decreased; we wanted to compare the same total amount of data so as not to confound the results. Training took approximately 30 minutes per interval.

We ran the CNN-inspired models on the same dataset. The results were surprising: the conventional CNN performed modestly while a network with a single max-pooling layer without convolution showed high levels of success. In the presence of convolution and max-pooling before the fully connected layer, the network's test accuracy fluctuated arbitrarily between 50% and 87%. This behavior persisted through 50 training epochs regardless of hyper-parameters (such as number of filters, training audio length, and max-pooling kernel dimensions). Accordingly, we concluded that the network is either not learning constructively or severely underfitting to the data. Removing convolutional layers altogether and training using just a single dense layer, however, not only sped up the training process but also improved accuracy and consistency, with the network simply taking the max-pooled spectrogram as an input to the fully connected layer (equivalent to a convolutional network with a fixed uniform convolution filter). This step consistently increased the test accuracy above 90%, often achieving 95%. Notably, the validation accuracy across epochs exhibits a smooth convergence pattern, a sign of consistent learning. Varying the training audio length and max-pooling kernel dimensions did not greatly impact performance; the accuracy was maximized for 2 second clips and a 2×4 max-pooling kernel.

5. Conclusion

We proposed a purely learning-based classifier that predicts the composer of a piece given a short audio clip. To achieve this without specializing to individual songs' characteristics, we considered the structures underlying audio signals of music. Because music is by nature sequential, we can approximate it as a sequence of spectral coefficients sam-

pled at regular time intervals. Alternatively, we can use the spectrogram of an audio signal as an image representation which exhibits locality. This allows us to transform audio classification into other tasks such as sequence or image classification, both of which are well-studied problems with state-of-the-art solutions in wide use.

Using the LSTM and CNN models, we exceeded 90% accuracy. This success is largely due to exploiting underlying structure of the data. Notably, a CNN with a fixed uniform filter outperformed a full CNN because the former did not need to explicitly learn the underlying horizontal locality in spectrograms. Using available information from the data distribution helped us build models that perform well.

The next step for this project or related work would be to extend down the hierarchy to a multi-class identifier that can distinguish different composers within a single time period, or between different pieces by a given composer. In the course of this project, we considered all types of music by a given composer as equal, but there is an obvious difference between a Bach piano sonata and a Bach oratorio, in number of musicians, type of music, dynamic contrast, and other features: in future, it may be more illustrative to consider works in a specific type of instrumentation and genre, which would be less useful overall but more helpful in determining true distinctions between the two composers as composers, rather than orchestration and instrumentation, which could easily have been a large confounding variable here.

References

Bhattacharjee, Mrinmoy, S. R. M. Prasanna, and Prithwijit Guha. Time-Frequency Audio Features for Speech-Music Classification. *arXiv e-prints*, art. arXiv:1811.01222, Nov 2018.

Cilibrasi, R., Vitanyi, P., and de Wolf, R. Algorithmic Clustering of Music. *arXiv e-prints*, art. cs/0303025, Mar 2003.

Hershey, S. and S. Chaudhuri and D. P. Ellis and J. F. Gemmeke and A. Jansen et. al. CNN architectures for large-scale audio classification. *IEEE conference on Computer Vision and Pattern Recognition*, pp. 131–135, 2017.

Hochreiter, S. and Schmidhuber, J. Long Short-term Memory. *Neural Computation*, 9:1735–1780, 1997. doi: 10.1162/neco.1997.9.8.1735.

Korzeniowski, F., Sears, D. R. W., and Widmer, G. A Large-Scale Study of Language Models for Chord Prediction. *arXiv e-prints*, art. arXiv:1804.01849, Apr 2018.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-Based Learning Applied to Document Recognition. *Proceedings of the IEEE*, 86(11):2278–2324, Nov 1998.

Padmanabhan, Jayashree and Jose Johnson Premkumar, Melvin. Machine Learning in Automatic Speech Recognition: A Survey. *IETE Technical Review*, 32:1–12, 2015. doi: 10.1080/02564602.2015.1010611.

Wang, DeLiang and Jitong Chen. Supervised Speech Separation Based on Deep Learning: An Overview. *CoRR*, abs/1708.07524, 2017. URL `http://arxiv.org/abs/1708.07524`.