

Bachprop: Using Neural Networks to Solve Composer Classification

Anonymous EMNLP submission

Abstract

Music shares many common features with natural language. Both are methods of communication, and have been analyzed in depth and broken down into component parts (grammar for natural languages, rhythms/melodies/harmonies for music). In this paper, we present a state of the art model to classify musical pieces based on their authors. Our model achieves 87.3% accuracy on a dataset of 3 composers, Bach, Beethoven, and Haydn, a 1.3% improvement over existing work. Finally we show that our model generalizes to larger set of composers and gain 72% accuracy on a dataset of 10 composers.

1 Introduction

In 1983, American linguist Ray Jackendoff and music theorist Fred Lerdahl presented “A generative theory of tonal music”, a formal description of music comparable to Chomsky’s generative grammar. This theory influenced and inspired further work by other researchers in the field of music cognition to pursue the creation of a musical grammar. These studies show that there is a strict hierarchical structure in a musical components; that music too follows a vague set of rules just like other natural languages. Supposing that this idea is true, we should be able to use NLP techniques to model musical style, classify musical works, and perhaps even generate music.

There has been a number of studies on rule based approaches for this problem. Previous works mostly includes producing a generative grammar to capture stylistics features of a musical pieces. However, these studies show that musical style happens in ways musicology and cognitive approaches are still unable to perfectly define. In this paper, we explored using data-driven techniques to analyze these passages. We propose to transform the composer identification problem

into a sequence classification problem and train an LSTM to solve it. We believe our approach will be more general and produce better results than rule-based approaches.

One of the challenges of this work is to find a good representation for musical features, a prerequisite for building an accurate classification model. There are number of ways to represent these musical structures and features. In this work, we decided to work with the MIDI files from the KernScores database (?). MIDI provides a symbolic representation of music and low and mid level features are easily extracted from it; specifically, we pulled the pitch, duration, tempo, key signature and time signature information from each piece. We believe this format is better than attempting to extract features from audio recordings, as it eliminates the effects of noise due to the musical performance.

In this paper, we propose a new way to building a composer-classification model that can accurately identify the composer of a musical piece. Specifically, we would like to solve the multi-class version of this problem using a sequence classification approach and recursive neural networks. The rest of the paper is organized as follows. Section 2 reviews related work. Section 3 explains our approach for building the composer-classification model. Section 4 presents and discusses our experiments and their results. Section 5 contains any concluding remarks and future work.

2 Related Work

The earliest work on the field of music information retrieval (MIR) goes back to 1960s (Kassler, 1966) and it became more popular after development of models musical grammars. MIR has been used in various research topics including finding the sim-

ilarities between two musical pieces (Berenzweig et al., 2004), finding a song based on a tune (Ghies et al., 1995) and specifically for composer classification which is the topic of our research.

One of the earliest studies on classifying musical pieces using machine learning was done by Buzzanca (Buzzanca, 2002). He used a supervised learning approach for musical style recognition of Giovanni Pierluigi da Palestrina. He implemented a neural network that could recognize Pierluigi’s style with 97% accuracy. However, his method is not considered fully automated since all the pieces of the music he used are largely preprocessed and classification is only done on selected short main themes.

There have been a few studies on style recognition using an n-gram model. Wolkowicz, Kulka, and Keselj (Wolkowicz et al., 2007) used an n-gram model to classify piano files of five composers. Hillewaere, Manderick, and Conklin (Hillewaere et al., 2010) also used n-gram models to classify musical pieces of two composers (Haydn and Mozart). They achieved an accuracy of 61%.

Mearns, Tidhar, and Dixon (Mearns et al., 2010) used a decision tree and naive Bayes models to classify similar musical pieces. The correctly classified 44 out of 66 pieces with seven composer classes. Dor and Reich (Dor and Reich, 2011) achieved an accuracy of 75% in classifying keyboard scores between Mozart and Haydn by using decision trees, naive Bayes, and support vector machines.

Herremans, Sorensen, and Martens (Herremans et al., 2015) built four composer-classification models to understand the stylistic differences between Beethoven, Bach, and Haydn. The first two models use an if-then rule set and a decision tree and the second two they use a logistic regression model and a semisupervised support vector machine classifier which produce more accurate results. They achieved an accuracy of 86% for the best model.

In this paper, we present a brand new approach to tackle the composer identifier problem. We use a Long Short Term Memory (LSTM) architecture to produce a composer-classification model to recognize the composer from a musical piece. Our model is a multi-class classifier that can be used to identify the composer of a musical piece.

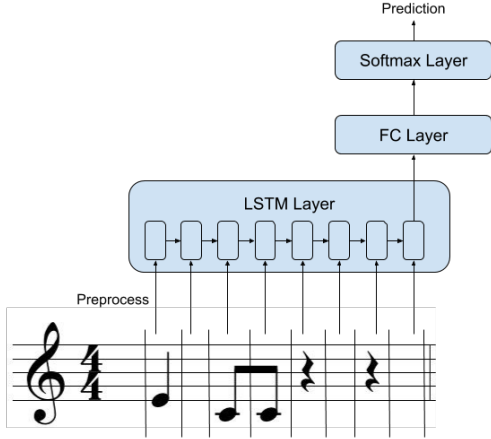
3 Approach

To solve this problem, we are taking a neural network based approach to multiclass classification. We divide the dataset into training, validation, and test datasets, with an 80 : 10 : 10 split. The first step is to preprocess the MIDI file into a sequence of feature vectors. We subdivide the MIDI file into 32nd note divisions in time, and for each division we compute a set of features. The most important feature is what notes are being played at that division of time; this corresponds to the time between the `note_on` and `note_off` messages in the file.

One issue that arises at this point is key normalization. The various pieces from each composer is written in a particular key, which represents the base note of the piece. Many chords, melodies, and harmonic structures are defined in terms of the key; for example, the 1/3/5 chord in C is a C/E/G, while in F it is F/A/C. Because MIDI provides key signature information, we are able to normalize each piece into the key of C by shifting all the notes in the piece either up or down. We believe this will increase performance as the intervals between the notes is of greater importance than their absolute pitch.

MIDI also provides several orthogonal features that we chose to extract; specifically, key signature, time signature, and tempo. Key signature was discussed as above, and represents the “base note” of the piece. A piece’s time signature represents the length and organization of each measure of music, which is similar to a phrase in natural language. For example, the most common time signature is 4/4, which means each measure consists of 4 beats, and each beat is 1 quarter note long. Finally, the tempo of the piece, measured in bpm, is how fast the piece is meant to be played. Key, time, and tempo can change in the middle of the piece, so we extract these features at each time slice as well.

We then convert the notes, time signature, and key signature into categorical one-hot vectors, and treat the tempo as a continuous feature. We then concatenate our features into one feature vector for each time slice, and take a continuous segment of 64 divisions (2 measures of music in 4/4 time) as one training example.



The input sequence is then passed into an LSTM layer. We found that using a 150 length hidden state vector achieved the best results over our validation dataset. The final output of the LSTM layer is then passed into a fully connected linear layer, and finally into a softmax layer to compute the probabilities for each composer. We then pick the composer with the highest probability as the final result for the phrase. To increase accuracy over the whole musical piece, we take a majority vote over all the phrases in the piece. For the rest of this paper, we refer to “phrase accuracy” as accuracy for 1 phrase, while “piece accuracy” is the accuracy for each piece after taking this majority.

To train the network, we use mini-batches of size 500 and the Adam optimizer with a decaying learning rate. Since our dataset is highly imbalanced (we have several times more examples from Beethoven/Bach than lesser known composers), we oversample the composers with fewer unique examples to compensate. The optimizer is set to minimize the cross entropy error over the training dataset. Our deep learning implementation was done in TensorFlow.

4 Experiments and Results

4.1 Data

In this paper we use the KernScores database which contains musical pieces as a MIDI files. The KernScores database is a large collection of virtual musical scores made available by the Center for Computer Assisted Research in the humanities at Stanford University (CCARH). The KernScores database has a total of 7, 866, 496 notes and is available online at kern.ccarh.org

As a baseline, we aim to compare our results with the 86% accuracy result from Herre-

Composers	Examples
Beethoven	10526
Haydn	9758
Bach	8544
Corelli	3555
Buxtehude	3728
Monteverdi	302
Foster	350
Frescobaldi	1904
Josquin	79
Schumann	82
Joplin	1788
Gershwin	890
Giovannelli	139
Mozart	6733

Table 1: List of composers.

mans, Sorensen, and Martens (Herremans et al., 2015). They also used the KernScores dataset and achieved fairly good accuracy. Their work attempts to solve multiclass classification between Bach, Beethoven, and Haydn, so we also chose these composers in our experimental results. An overview of the selected composers was given in table 1.

4.2 Experiments

For this paper, we primarily ran two different experiments; one in which we train on data from Beethoven, Haydn, and Bach, creating a 3 composer classifier. This allows us to get a good comparison with the previous work. The other experiment we explored was creating a 10 composer classifier using more composers; this allows us to see how well the system generalizes. In addition, the 10 composer case contains composers with a wider range of styles, rather than simply classical music. In particular, Joplin is a ragtime composer, and Foster is a modern composer.

We ran each of the experiments on “machine info”.

One other purpose for running these experiments is to find good values for the hyperparameters. The primary parameters we worked with were the number of measures in each sequence, the size of the mini-batches, the learning rate of the optimizer, and the hidden state size. We found that 2 measures of 4/4 time music and 150 hidden units seemed to maximize our validation accuracy while avoiding overfitting. Using size 500 mini-

batches and a decaying learning rate improved our training time. Specifically, the learning rate is initialized to 0.01, and decays by an order of magnitude every 150 iterations. We found that around 450 iterations was enough for validation loss to converge.

4.3 Results

Most previous work identifies a composer from an entire piece, so to properly evaluate our model we should compare our piece accuracy against the other models. Our results show that our model achieves an accuracy of 87.3% on Beethoven, Bach, and Haydn, and 1.3% improvement over the previous work. Figure 1 and 2 shows the phrase and piece accuracy of our model for 3 classical composers.

Figure 1: Phrase Accuracy Training Curve for 3 composers

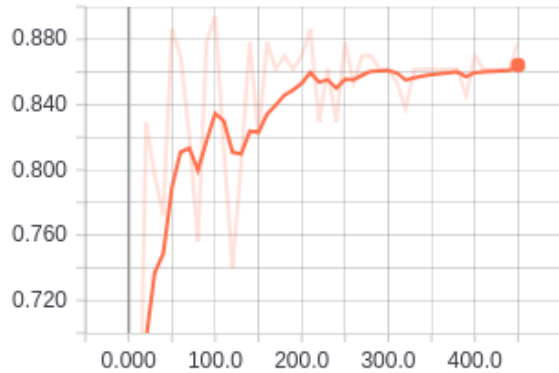
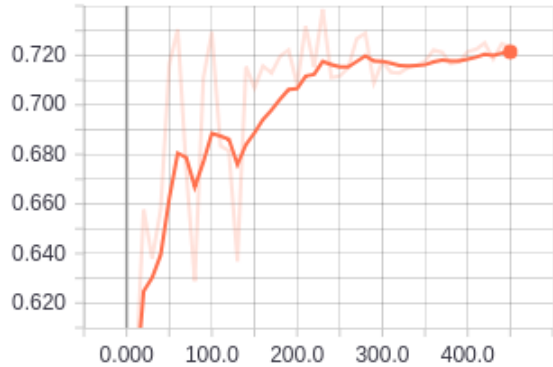


Figure 2: Piece Accuracy Training Curve for 3 composers



In addition to the 3 composer case, we also ran some experiments to see if our classifier generalizes to more composers from a wider range of styles. Using the 10 selected composers above, we were able to achieve an accuracy of 72% using the same model. Table 3 presents the accuracy for

Method	Accuracy (%)
LSTM	87.3
Support vector machines	86
Logistic regression	83
C4.5 decision tree	79
RIPPER rule set	77

Table 2: Model evaluation.

Composer	Piece Accuracy (%)
Beethoven	82
Bach	80
Haydn	66

Table 3: Piece accuracy for 3 composers.

Composers	Accuracy (%)
Beethoven	72
Mozart	47
Foster	59
Frescobaldi	99
Josquin	65
Schumann	62
Joplin	93
Gershwin	95
Giovannelli	96
Vivaldi	95

Table 4: Accuracy.

each composer. Interestingly, the lowest accuracy over all composers is Mozart. We believe this is because his musical style was highly variable and the body of work is vast, which makes it difficult for our model to nail down his style quantitatively. Figure 3 shows the training accuracy of our model. The high training accuracy shows that the training part of our system works well. Figure 4 and 5 summarizes piece accuracy and phrase accuracy respectively for 10 composers list.

5 Conclusion and Future Work

The development of musical grammar shows that there is a hierarchical structures in a musical component. Over the years, researchers around the world tried to classify musical pieces based on these hierarchical structures. With the advances in computers and technologies, researchers started classifying pieces using computers. In this paper, we tackled the question "Can computers accurately identify the composer of a musical piece?".

Figure 4: Training Accuracy Curve

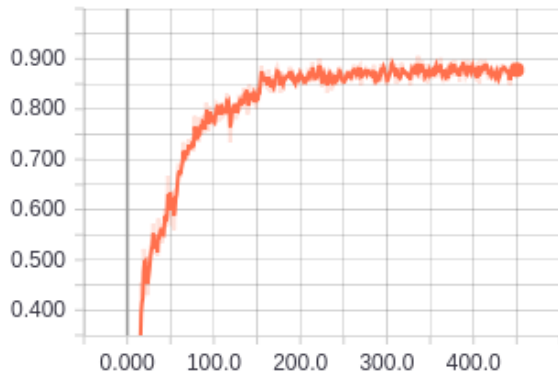
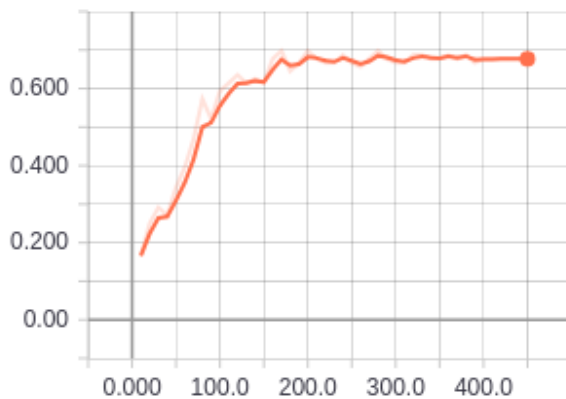


Figure 5: Piece Accuracy Training Curve



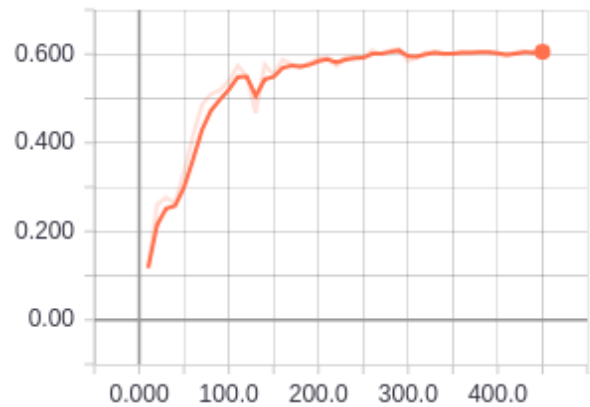
The result is an LSTM model that is capable of identifying the composer of each musical piece. We were able to build a model that surpass the previous state of the art approach by 1.3%. Our approach is generalized to work on a larger list of composers compared to previous approaches.

As for future work, we would like to have an explicit feature extraction. We believe having the right features has a high impact on the accuracy of our model. We would also like to study each composers in depth and have a collaboration with a few music theorist to build a better model. Also, currently we have a multi-class classifier. We would like to expand our work and build a binary classifier as well were we can tell if a musical piece is a work of specific composer.

References

- Adam Berenzweig, Beth Logan, Daniel P. W. Ellis, and Brian P. W. Whitman. 2004. A large-scale evaluation of acoustic and subjective music-similarity measures. *Comput. Music J.*, 28(2):63–76.
- Giuseppe Buzzanca. 2002. A supervised learning approach to musical style recognition.

Figure 6: Phrase Accuracy Training Curve



- O. Dor and Y. Reich. 2011. An evaluation of musical score characteristics for automatic classification of composers. *Computer Music Journal*, 35(3):86–97.

- Asif Ghias, Jonathan Logan, David Chamberlin, and Brian C. Smith. 1995. Query by humming: Musical information retrieval in an audio database. In *Proceedings of the Third ACM International Conference on Multimedia*, MULTIMEDIA '95, pages 231–236, New York, NY, USA. ACM.

- D. Herremans, K. Srensen, and D. Martens. 2015. Classification and generation of composer-specific music using global feature models and variable neighborhood search. *Computer Music Journal*, 39(3):71–91.

- Ruben Hillewaere, Bernard M, and Darrell Conklin. 2010. String quartet classification with monophonic models. In *In ISMIR*, pages 537–542.

- Michael Kassler. 1966. Toward musical information retrieval. *Perspectives of New Music*.

- Lesley Mearns, Dan Tidhar, and Simon Dixon. 2010. Characterisation of composer style using high-level musical features. In *Proceedings of 3rd International Workshop on Machine Learning and Music*, MML '10, pages 37–40, New York, NY, USA. ACM.

- Jacek Wokowicz, Zbigniew Kulka, and Vlado Keelj. 2007. Archives of acoustics 33, 1, 4355 (2008) n-gram-based approach to composer recognition.

Appendices (*i.e.* supplementary material in the form of proofs, tables, or pseudo-code) should be **uploaded as supplementary material** when submitting the paper for review. Upon acceptance, the appendices come after the references, as shown here. Use \appendix before any appendix section to switch the section numbering over to letters.