Bachprop: Using Neural Networks to Solve Composer Classification

Anonymous EMNLP submission

Abstract

Music is the ultimate language and shares many common features with the 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 88% accuracy on a dataset of 3 composers, Bach, Beethoven, and Haydn, a 2% 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

It has been more than four decades since Leonard Bernstein called on researchers at his famous lecture at Harvard University (**?**) to create a musical grammar similar to Noam Chomsky's generative grammar (**?**).

In this paper, we developed a model that could identify the composer of a musical piece using these hierarchical structures in a musical component.

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 them. 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 that 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 multiclass 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 Works

With the advances of new techniques in machine learning and deep learning, recognizing a musical piece composer has become the center of attention in the field of music information retrieval (MIR).

Buzzanca (**?**) 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 97% accuracy.

There have been a few studies on style recogni-

tion using an n-gram model. Wolkowicz, Kulka, and Keselj (**?**) used an n-gram model to classify piano files of five composers. Hillewaere, Manderick, and Conklin (**?**) 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 (**?**) 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 (**?**) 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 (**?**) built four composer-classification models to understand the stylistic differemces 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 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. At this stage, we also transpose the notes into the key of C or C minor based on the currently active key signature. This helps normalize the data and remove any noise caused by key selection. We then convert the set of notes being played into a one-hot vector. We also extract the tempo, time signature, and key signature from each division of time and concatenate those features to the note feature (tempo being treated as continuous, while time and key are treated as categorical one-hot vectors). We then take each 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 pharses 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 are using 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 over the training dataset. The loss corresponds to the cross entropy error of our prediction compared to the actual composer of the musical piece. Our deep learning implementation was done in TensorFlow.

4 Experiments

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 7866496 notes and is available online at `kern.ccarh.org`

As a baseline, we aim to compare our results with the 86% accuracy result from Herremans, Sorensen, and Martens (**?**). 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 experi-

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	9758
Mozart	6733

Table 1: List of composers.

mental results. An overview of the selected composers was given in table 1.

4.2 Experiments

TODO

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 88% on Beethoven, Bach, and Haydn, and 2% improvement over the previous work.

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 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.

4.4 Conclusion and Future Work

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.

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

Table 3: Accuracy.

Figure 1: Piece Accuracy Training Curve

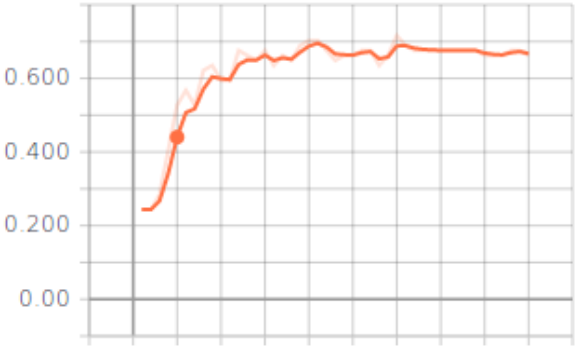

Figure 2: Phrase Accuracy Training Curve