Distinguishing Bach, Beethoven, and Mozart: Deep Learning solutions To The Composer Classification Problem

Siddharth Bachoti*
Department of Physics
Indian Institute of Science Education and Research Bhopal

June 20, 2020

Abstract

In this text, the composer classification problem is described and a new network architecture based on a combination of ConvNets and GRUs is proposed to address the problem. Methods of data augmentation are described and a trend of acccuracy increasing with augmentation factor is reported. The best test set accuracy obtained is 72.1%. The code is available on *GitHub*.

1 Introduction

In the composer classification problem, the aim is to be able to correctly identify the composer of a given musical piece. The solutions proposed in this study are developed for western classical music.

The composer classification problem is very different from other classification problems such as the facial recognition, text recognition, object recognition, etc. The first point of difference is that it is trivial for humans to perform image recognition or text recognition on their own. However, this is not the case in the composer classification problem. Even the most well known musical experts often debate about the true authors of some pieces. The other difference is with regard to availability of freely available data sets. To date, there are extremely few freely available MIDI data sets for some genres that are large enough for deep learning studies. This makes it an extremely interesting problem to study for musicians and computer scientists alike.

2 Related work

Verma and Thickstun[1] used hand-made convolutional machine learning learning models to classify 19 composers. They argued that deep learning models are difficult to design due to unbalanced classes and lack of enough data. They achieved 80% accuracy for their best models. Hu et al.[2] were one of the first ones to propose a relatively successful deep learning model to address this problem. Niko Abeler[3] designed a successful deep neural network and trained it on 400 audio clips of merely 30 seconds each. The accuracy that he obtained was 76%. Gianluca Micchi[4] conducted important deep learning experiments for this problem with the spectogram method. His model used 3 convolutional layers followed by an LSTM and obtained an accuracy between 60 and 70%. A very interesting method of preprocessing was taken up by Velarde et al.[5] wherein they applied Gaussian and Morlet filters on the piano rolls and then subsequently applied their classification models. Their study claims an improvement of results by 5% when such filters were applied.

Among most of the studies mentioned above, the major drawback that I observe is the small size of data sets (number of scores per composer).

Jain et al.[6] classified 6 composers with a CNN

[0]Roll number: 17274

which had an overall structure quite similar to the VGGNet. They used the piano roll representation with 154 seconds of audio clips. They obtained 70% accuracy after data augmentation by a factor of 15. Their data set size far exceeds the ones that have been mentioned above.

3 Data sets

The raw data used in this study consists of 8,000 MIDI (Musical Instrument Digital Interface) files provided by *Classical Archives*. Owing to the wishes of the owner of the website - Mr. Pierre Schwob, the data-sets will not be available to the reader. MIDI files contain a series of messages containing the sequence of notes being played, the various instruments being played, the tempo variations and dynamics, instrument - audio connectors used for feeding the audio into the computer, etc. *Mido* was the primary library used for reading the MIDI messages contained in the data sets. Please refer to Appendix A for more information about MIDI files and how to make sense of the messages.

4 Data representation: Spectogram v/s piano roll

When we think of music, we think of musical audio files (mp3, wav, etc.). However, deep learning models require numerical data as input. This calls for considerable pre-processing of data. There are many ways in which one can represent audio data in numerical data structures. The Fourier Transform spectogram and the piano roll are two of the popular choices.

The Fourier transform spectogram simply represents the audio signal of the song in the frequency domain. This is done using numerical methods such as FFT (Fast Fourier Transform). The subsequent strategy is to treat these spectogram images as NumPy arrays and then provide them as inputs to neural networks. However, this technique has a major drawback - the sequential nature of a song which contains information regarding chord progressions, scale, tempo, dynamics, etc. is not easily perceived in the spectogram.

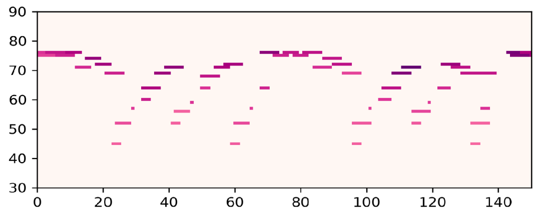


Figure 1: A Piano Roll representation of music

The piano-roll representation preserves the sequential nature of a song and hence it should ideally contain much more information about the song which is used by humans to distinguish composers. The model described in this study use the piano roll representation.

5 The piano roll

An image of a piano roll representation containing the first14 seconds of the famous Für Elise by Ludwig Van Beethoven is shown in 1. The x-axis describes the flow of time and the y-axis describes the variation in pitch. One might be able to mentally map a correspondence between the piano roll and the piece. A Nyquist rate of 10 hertz was taken in order to perform the message sampling. Therefore, once the MIDI file is converted into a piano-roll, the entire MIDI file can be treated as an image which in turn can be viewed as a NumPy array. This enables us to treat the problem as an image recognition problem at least in terms of the syntax and input. However, since the x-axis represents the flow of time, we can also treat this as a sequence classification problem.

The y-axis ranges from 0 to 127 while the x-axis (time) varies with the length of the song (or chosen sample duration).

6 Memory constraints

One of the most important constraints that we come across with the piano-roll representation is memory. Each element of a NumPy integer array takes up 8 bytes. The typical length of a musical piece is 240 seconds or 4 minutes (some pieces can be more than 500 seconds long). With a Nyquist rate of 10, 240 seconds requires 2400 elements for each pitch. Since pitch range goes from 0 to 127, a piano roll representing just one piece has dimensions- 128×2400 which occupies 0.0024 GB. Since the entire data-set contains almost 7000 pieces, the total space occupied just by the data set without any data reduction is almost 17.23 GB. The Kaggle notebooks on which the experiments for this study were conducted provide only 16 GB of RAM (13 GB in case of a GPU). This calls for a massive data reduction as well as dimension reduction in order to carry out the experiments.

Therefore, we have restricted the pitch to between 30 and 100 and we have sampled only the first 70 seconds of each piece.

7 Data augmentation

In order to increase the number of piano rolls for some of the experiments that were conducted, the pieces were transposed. In image recognition language, this refers to a translation of the piano roll in the y-direction. However, transposition is also a musical term. It refers to change in scale or key signature. Therefore, by transposing, we are essentially recreating the same musical piece but with every note raised or lowered by a constant amount.

Translation in the direction of time was also used as a data augmentation method along with scale transposition.

8 Removal of dynamics

A separate set of experiments can be carried out by removal of dynamics. In terms of the effect on data, every non zero element of the NumPy array is replaced with a '1'. In musical terms, this removes the notion of relative loudness or hardness of the notes being played. Every note would have the same 'volume'.

9 Experiments

In general, dense neural networks showed very poor accuracy. CNNs and recurrent networks obtained a much better accuracy than the dense networks.

9.1 Binary classification with a VGGNet style architecture

With a VGGNet style CNN based architecture, binary classification of Bach and Beethoven was attempted. A reasonable accuracy of 69-70% was obtained consistently.

However, This architecture was subsequently discarded due to poor results on the trinary problem (depreciated architecture can be viewed in the code).

9.2 Removal of Dynamics

It was noted that the removal of dynamics reduced the accuracy. Therefore, it is concluded that dynamics do play a role in the identification of a composer.

9.3 Description of the model

The model proposed in this study is given in figure 2. It contains 2 consecutive 1D convolutional layers of 128 filters each (kernel siz = 4), followed by a 1D MaxPooling layer of size 4, followed by a Gated Recurrent Unit of 512 filters, followed by a Dense network of 64 filters, and a final softmax output layer.

The Scaled Exponential Linear Unit (SeLu) was chosen as the activation function. It was concluded through experiments that the SeLu was a much better choice of activation than ReLu. The learning was optimized with the Adam optimizer.

9.4 Data augmentation

It was noted that the augmentation of data increased the overall accuracy of the model as shown in the table 1. While the trend is not monotonic, the overall trend of increasing accuracy is certainly visible.

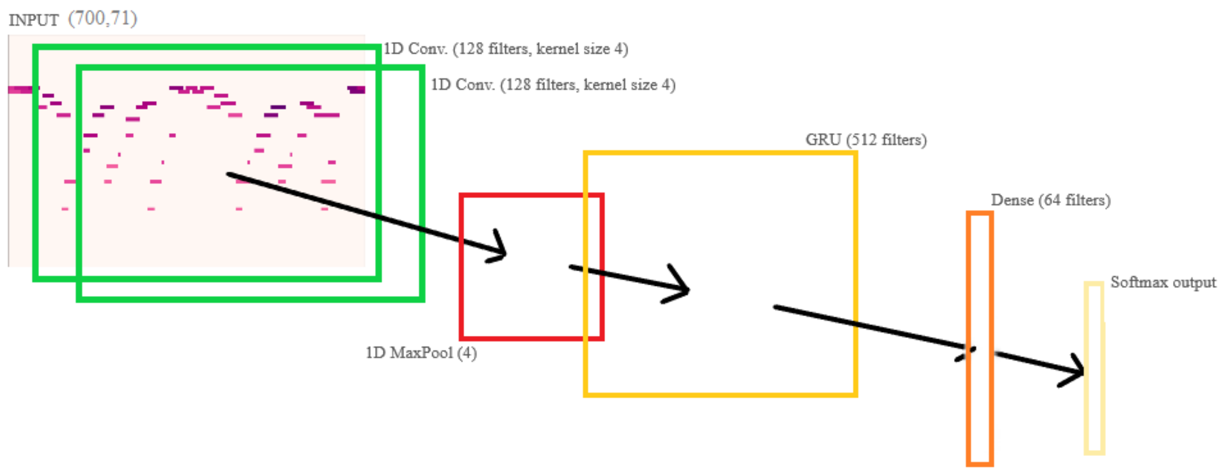


Figure 2: The proposed model

Factor of augmentation	Accuracy
k=1	57.8%
k=2	59.5%
k=3	57.8%
k=4	64.5%
k=5	69.1%
k=6	68.4%
k=7	72.1%

Table 1: Data augmentation results

9.5 Results

The best case accuracy on the test data set that the model obtained was 72.1%. The confusion matrix is given in figure 3.

As can be noted from the matrix, Bach and Beethoven were easily distinguishable. This may be due to the fact that Bach and Beethoven are further apart in terms of the eras in which they composed. Mozart is less easier to distinguish due to being closer to both Bach as well as Beethoven in terms of the era of composition.

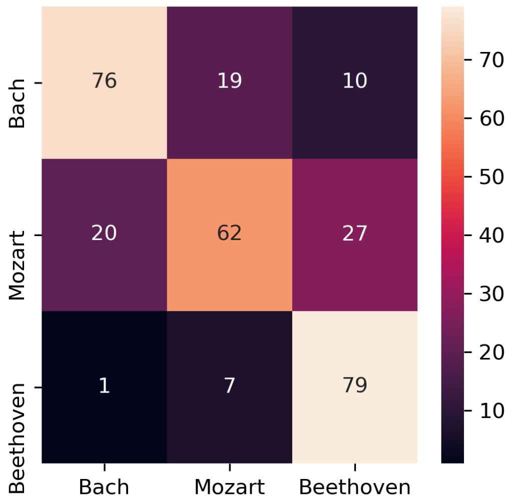


Figure 3: Confusion matrix

10 Scope for further study

10.1 Indian Classical Music

Many music experts have found some similarities between Indian and Western classical music. Therefore, it would be interesting to see how the methods described in this study can be applied on Indian Classical music as well.

10.2 Increasing the number of categories

Since the work here was conducted on limited RAM of 13 GB, the size of data sets had to be heavily reduced compared to some of the other studies carried out on the composer classification problem. With greater RAM and High Performance Computing solutions, the number of composers can certainly be increased.

10.3 Transformers

Given the sequential nature of music, some similarities can be drawn between natural language processing and music information processing. Transformers have been extremely successful in solving NLP problems. It would be interesting to see how such techniques can be applied to the composer classification problem.

10.4 Filters

Following the methods of Velarde et al., we can study how the application of Gaussian and Morlet filters would improve the results and affect the computational load.

11 Acknowledgements

I thank Dr. Parthiban Srinivasan for giving me this opportunity to carry out the studies on a topic of my choice. I thank Mr. Pierre Schwob and Classical Archives for generously providing me with the data sets. I also thank Dr. Kushal Shah (Head, Department of Electrical Engineering and Computer Science, IISER Bhopal) for endorsing my request for the data sets from Mr. Pierre Schwob. Finally, I thank Siddharth Chaini for his inputs on RNNs/LSTMs and his ideas regarding reducing computational load.

References

[1] Harsh Verma and John Thickstun. Convolutional composer classification. In Arthur Flexer, Geoffroy Peeters, Julián Urbano, and Anja Volk, editors, *Proceedings of the 20th International Society for Music Information Retrieval Conference, ISMIR 2019, Delft, The Netherlands, November 4-8, 2019*, pages 549–556, 2019.

[2] Zhang Changshui Hu Zhen, Fu Kun. Audio classical composer identification by deep neural network. *Journal of Computer Research and Development*, 51(9):1945, 2014.

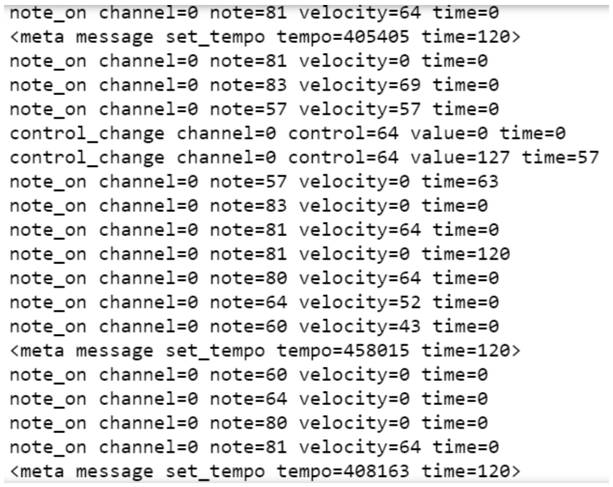
[3] Niko Abeler. Musical composer identification mus-15, 2015.

[4] Gianluca Micchi. A neural network for composer classification. International Society for Music Information Retrieval Conference (ISMIR 2018), 2018. Poster.

[5] Gissel Velarde, Tillman Weyde, Carlos Eduardo Cancino-Chacón, David Meredith, and Maarten Grachten. Composer recognition based on 2d-filtered piano-rolls. In *ISMIR*, 2016.

[6] Jain et al. Analysis and Classification of Symbolic Western Classical Music by Composer.

[7] H. M. de Oliveira and R. C. de Oliveira. Understanding midi: A painless tutorial on midi format, 2017.



```
note_on channel=0 note=81 velocity=64 time=0
<meta message set_tempo tempo=405405 time=120>
note_on channel=0 note=81 velocity=0 time=0
note_on channel=0 note=83 velocity=69 time=0
note_on channel=0 note=57 velocity=57 time=0
control_change channel=0 control=64 value=0 time=0
control_change channel=0 control=64 value=127 time=57
note_on channel=0 note=57 velocity=0 time=63
note_on channel=0 note=83 velocity=0 time=0
note_on channel=0 note=81 velocity=64 time=0
note_on channel=0 note=81 velocity=0 time=120
note_on channel=0 note=80 velocity=64 time=0
note_on channel=0 note=64 velocity=52 time=0
note_on channel=0 note=60 velocity=43 time=0
<meta message set_tempo tempo=458015 time=120>
note_on channel=0 note=60 velocity=0 time=0
note_on channel=0 note=64 velocity=0 time=0
note_on channel=0 note=80 velocity=0 time=0
note_on channel=0 note=81 velocity=64 time=0
<meta message set_tempo tempo=408163 time=120>
```

Figure 4: A sequence of MIDI messages

A Decoding MIDI messages

When unrolled, a MIDI file can be represented as a sequence of messages. An example of such a sequence is given in figure 4. The feature at the beginning of some messages - "note_on" or "note_off" refers to whether a note is "on" or "off". The feature "note" refers to the pitch of the note that is being played. The pitch varies from 0 to 127. The feature "velocity" refers to the hardness or loudness with which the note is played. The meta message "set_tempo" refers to a change in speed with which the musical piece is being played. The feature "time" refers to the number of ticks that have passed since the last message. The expression relating the tempo, elapsed time and ticks is given by:

$$Ticks = \frac{resolution}{tempo} \times time \times 1000 \qquad (1)$$

Where resolution is the number of ticks per quarter note, tempo is the beats per minute and time is the elapsed time in milliseconds.

These are the only essential concepts of MIDI messages that are needed to understand the programs written for this study. For more information about MIDI files and and other features in MIDI messages, please refer to Oleivera's guide[7] to understanding MIDI files.