Tuning a convolutional neural network for classical composer recognition

Sri Datta Budaraju budaraju@kth.se
Peter Mastnak mastnak@kth.se
Nik Vaessen vaessen@kth.se
Laura Cros Vila lcros@kth.se

Abstract

In this paper, we study and analyze the relations between different classical piano compositions and experiment with deep neural network approaches to recognize the composer of the music samples and classify them. Our approach extracts the Mel-scaled spectrogram features of the music samples and leverages the power of convolutions to learn the low-level features and the temporal information using memory units like LSTMs and GRUs. The results obtained show that the architectures used in the experiments were not able to extract enough information from the spectrograms to recognize the composer.

1 Introduction

Automatic music classification has been an interesting subject in machine learning. Being able to classify high-dimensional, stylistically similar compositions are hard, even for humans. In this project, we make use of Convolutional Neural Networks (CNN), which are highly used for image classification as it exploits the correlation between close pixels. When it comes to audio signals, the temporal locality of the samples has to be taken into account. We approach this by making use of spectrograms treated as images. Representing audio as a spectrogram allows convolutional layers to learn the global structure and recurrent layers to learn temporal structure.

What makes the problem of composer recognition so challenging when using audio samples is that, unlike speaker/speech recognition, the person who produces the sound (or instrument) does not give information to identify the composer. In this case, the audio recordings contain piano artists playing the pieces that were composed by the composers we want to recognize. In this sense, the features used to recognize speakers are not useful for this problem, as all the pieces are played with piano.

1.1 Related work

Many researchers have tried to approach the problem of classical composer classification. Most of them, though, base their work on MIDI files, which is a Musical Instrument Digital Interface file that contains no audio data. To classify composers through audio signals, there are two clear approaches: the use of probabilistic models[4][5] and the use of convolutional neural networks (CNN) with some kind of recurrence[6][7][1].

The approaches that use probabilistic models mainly make use of the previously mentioned MIDI files. In our project, though, we wanted to be able to classify the composers through audio samples, so we chose to take the CNN with a recurrence approach.

Some other machine learning approaches for music classification that we explored are K-nearest neighbour[9] and Naive Bayes[2], although previous research using these approaches have been used for genre classification, which is a task not as complex as composer classification.

Project report for DT2219

2 Methods

2.1 Data

The methods proposed below were tested on the maestro dataset [3], a collection of 1282 recordings of 857 unique piano compositions spanning 61 different composers. The audio originates from performances of contestants in the International Piano-e-Competition and was recorded over a time period of 10 years. The audio samples vary in sampling rate (44.1 to 48 kHz) but are consistently stored as 16-bit PCM stereo. To make sure that there are enough samples to split between train, validation and test set a subset of 12 composers were selected. Each of these 12 composers has at least 29 unique compositions in the dataset. Table 1 shows the composer name and some statistics about the length of their compositions.

To test whether the network learned features specific to a composer instead of remembering whether a song belongs to a specific composer, the data was split into a training, validation, and test set such that a particular recording of composition is unique to one split [1]. To have a consistent sampling rate, each audio was downsampled to 22.5 KHz before it was transformed into 128-length Mel-scaled spectrogram features by using a frame length of 1024 samples (~20 milliseconds) and a Fast-Fourier transform of window length 2048.

We took 2 approaches to create a training/validation/test split (Table 2). The first approach was taking only the first window of an audio sample, which with a split of 70/20/10 results in 494 training, 150 validation and 58 test samples, denominated by 'small'. The 'small' approach, therefore, throws a lot of data away. The other approach created as many samples from each audio file as possible, denominated by 'full'. In this approach only a small part, the end of the file which did not fit into a window size anymore was, unused. 4 different window sizes were used. As a single frame is 1024 samples at 22.5 KHz, these window sizes (128, 256, 512, 768) correspond to respectively 6, 12, 24 and 36 seconds of audio.

Table 1: This table shows the names of the composers whose audio was selected from the maestro [3] dataset. For each composer, it shows the number of unique compositions present in the audio, the total duration of the audio, the mean length of their compositions, the standard deviation of the length of their compositions, and the minimum and maximum duration of their compositions in seconds.

composer name	num titles	sum (s)	mean (s)	std (s)	min (s)	max (s)
Alexander Scriabin	26	19268	551	259	102	1195
Claude Debussy	41	21507	478	415	79	2463
Domenico Scarlatti	29	6170	199	129	66	789
Franz Liszt	91	72959	557	365	45	1744
Franz Schubert	96	128959	693	549	106	2625
Frédéric Chopin	107	94389	470	396	71	2623
Johann Sebastian Bach	93	49560	342	223	45	1325
Joseph Haydn	29	17005	425	245	129	1031
Ludwig van Beethoven	96	99358	671	380	236	2458
Robert Schumann	31	56596	1155	627	246	2097
Sergei Rachmaninoff	50	23986	407	389	92	1751
Wolfgang Amadeus Mozart	35	20149	530	321	115	1139

[1] Instructions to reproduce the particular split train/val/test split used can be found at `https://github.com/nikvaessen/pianition`

Table 2: This table shows the 8 different datasets which were used in the experiments.

split name ->	full 128	full 256	full 512	full 768	small 128	small 256	small 512	small 768
input dimension	128x128	128x256	128x512	128x768	128x128	128x256	128x512	128x768
num_train	43280	21504	10616	6986	494	494	494	494
num_val	13824	6869	3398	2239	150	150	150	150
num_test	8763	4360	2159	1426	58	58	58	58

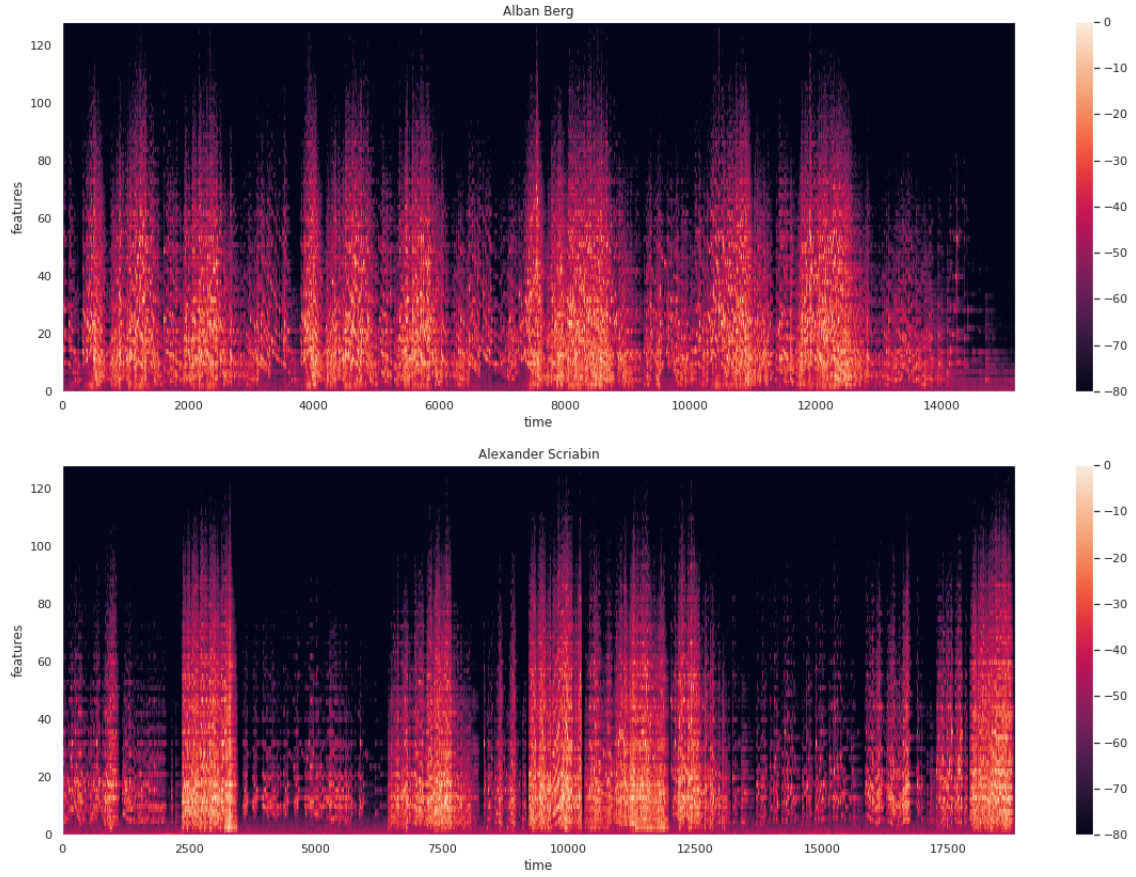

Figure 1: Spectrogram of the music samples

2.2 Neural Network

When it comes to audio processing, recurrent neural networks (RNN) would be reasonable to use as they allow to exhibit temporal dynamic behavior. In our case, that is exactly what we need, as the spectrograms analyzed sample by sample (like in feedforward neural networks) would not provide the information we need for composer classification.

To understand the potential of using RNNs on our dataset, we first implemented a simple network with just two LSTM blocks as illustrated in Figure 2. To further improve the performance we build a convolutional recurrent neural network (CNN+GRU), illustrated in Figure **??**.

The Recurrent Neural Network consisted of 2 LSTM layers with 64 units and tanh as activation function, followed by a fully-connected layer with 12 units and Softmax activation function.

The CNN part of the CNN+GRU model consisted of 3 uni-directional convolution layers with 64, 128 and 128 filters respectively, which had a kernel size of 5. These layers had ReLu as activation

function, and each one was followed by a uni-directional max pooling layer with a pooling size of 2 and strides of 2. After each pooling layer, we added a dropout of rate 0.2, 0.4 and 0.4 respectively. After the CNN part, we added two GRU layers with 64 units and tanh as the activation function. Following those layers, we added another dropout layer with a rate of 0.4. The last layer was a fully connected layer with 12 units and Softmax activation function. The intuition behind this architecture is to first extract low-level features at each time step by performing convolutions on the feature axis and then use the recurrent units to learn the temporal relations between these low-level features.

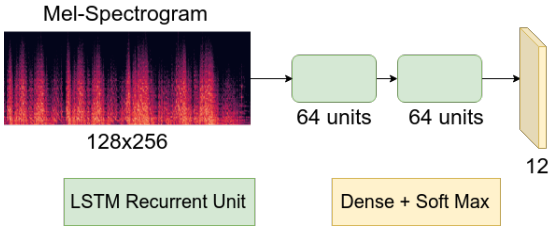

Figure 2: Overview of the RNN model architecture.

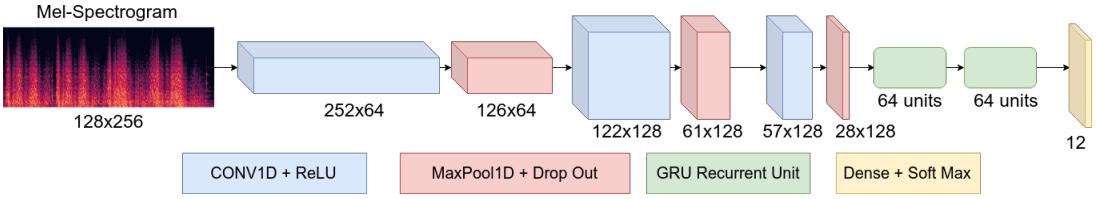

Figure 3: Overview of the CNN+GRU model architecture. Inspired by "Music Artist Classification with Convolutional Recurrent Neural Networks"[7]

3 Experiments

3.1 Data analysis

Before proceeding to the experiments for classification purposes, some data analysis was performed to have more information. The main goal of this part was to know how similar the composers were so that we could get a sense of how difficult the problem was. To do so, we used the FastDTW algorithm [8], which is an approximate Dynamic Time Warping (DTW), to get the distances between different spectrograms. The data used for this experiment was the small dataset. After computing all the distances between the spectrograms, the mean distance between the composers was obtained. See the results in the results section.

3.2 Classification

To carry out the classification task, we tried four different approaches. Before using neural networks, we wanted to try other machine learning methods such as K-Nearest Neighbors (KNN) and Naive Bayes. We performed the experiments over the different datasets (full and small), and for different window sizes (128, 256, 512, and 768).

For the KNN approach, we used the already implemented k-nearest neighbor classifier in the sklearn package. We tried four different numbers of k neighbors: 1, 3, 5, and 7. We selected the configuration that gave the best validation score for each dataset used.

For the Naive Bayes approach, we used the already implemented Naive Bayes classifier for multinomial models in the sklearn package.

For the neural networks, we trained for 75 epochs, using Adam optimizer, a batch size of 32 and a learning rate of 0.001. As for the loss function, we used cross entropy. We used the validation data to evaluate the loss and accuracy of the model at the end of each epoch. Both networks were built using Keras and Tensorflow and trained on Google Cloud Platform.

4 Results

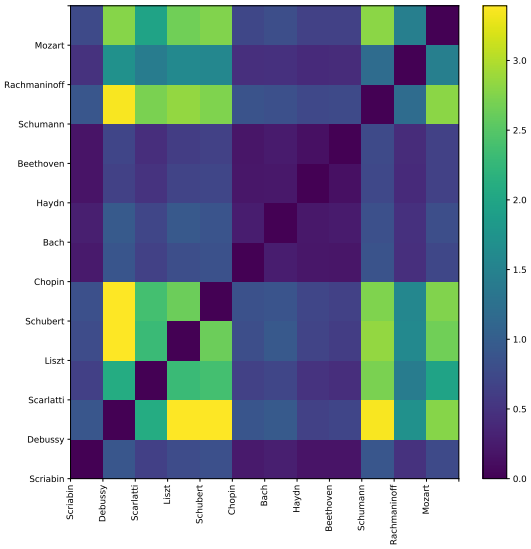

Figure 4: Distances between the composers.

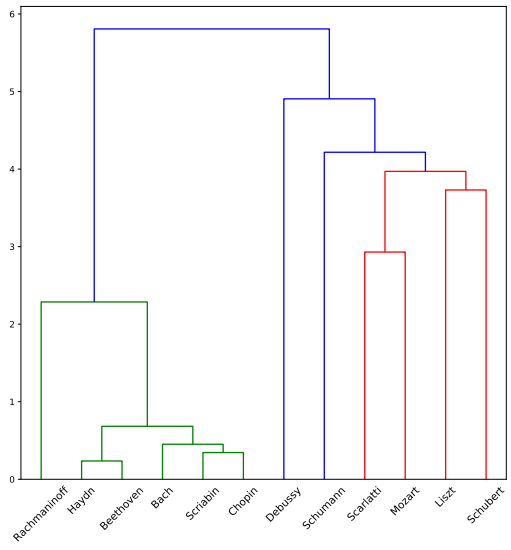

Figure 5: Dendrogram of the distances between the composers.

For the data analysis part, we can see from Figures 4 and 5 that the most similar composers are Haydn and Beethoven, which makes sense as Haydn was Beethoven's mentor. The most distinct composer is Debussy, as he is considered the first Impressionist composer, a trait only shared with Scriabin.

The distances between the composers show how difficult the task of recognizing them is. Composers such as Chopin, Bach, Haydn, and Beethoven can easily be misclassified, as they are very similar to the other composers based on the dataset that we used.

Table 3: This table shows classification test accuracy's on all the different

Technique	full128	full256	full 512	full 768	small128	small256	small512	small768
KNN	0.1472	0.14289	0.1491	0.1521	0.2949	0.2179	0.2436	0.2051
Naive Bayes	0.1294	0.1350	0.1431	0.1536	0.0897	0.1538	0.1410	0.1666
RNN	0.1498	0.1516	0.1501	0.1620	0.1154	0.0769	0.1539	0.0897
CNN+GRU	0.1510	0.1516	0.1507	0.1522	0.1282	0.2308	0.1410	0.1410

The model that gave the best accuracy was KNN with the small128 dataset, although KNN does not perform as good with the full dataset. We can also see from Table 3 that, for the full dataset, the accuracy tends to increase with the window size. This does not happen with the small dataset, which consisted of the 5 first seconds of each audio sample. For the neural networks, even if RNN performs better than CNN+GRU with the full768 dataset, it looks like the convolutional approach performs better for most of the other datasets. The accuracies obtained (except for RNN with small256) are slightly better than random ($\frac{1}{12} = 0.0833$).

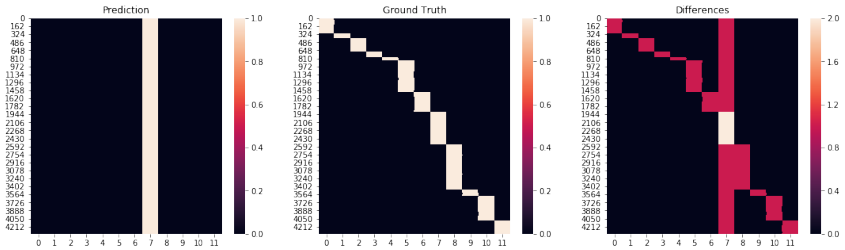

Figure 6: Confusion matrix example for the RNN (LSTM) model. The numbers for the x-axis correspond to the composers, in the same order as in Table 1. The y-axis corresponds to the samples used to obtain the confusion matrix.

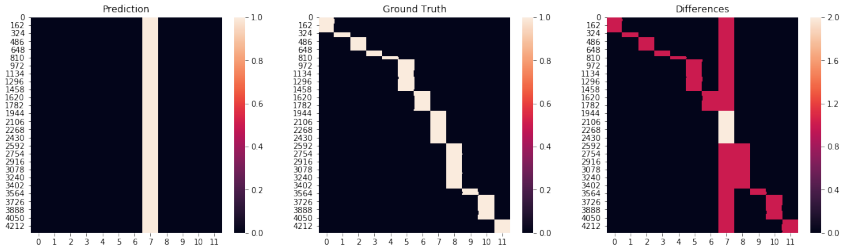

Figure 7: Confusion matrix example for the CNN+GRU model. The numbers for the x-axis correspond to the composers, in the same order as in Table 1. The y-axis corresponds to the samples used to obtain the confusion matrix.

If we look at the predicted values from figure 6 and figure 7 we see that both RNN(LSTM) and CNN+GRU models predict only one label. This means that the model was not able to learn much from/about the composers, as it only classifies one composer. The initial guess was that if the model would predict only one composer, then the predicted composer would be Franz Schubert, as it is the one that has the most samples in the dataset. The predicted label though was Haydn, which is one of the composers with the least samples, but at the same time that has the least distance to the other ones (see Figure 4), so it can be easily misclassified. Hence the model is believed to give the safest prediction by picking the mean class as it is unable to learn any more information from the features.

5 Discussion & Conclusion

In this project, our attempt at recognizing musical composers proved to be more difficult than we initially expected. From the results obtained we can see that there is some information that can be extracted to perform the classification. While our baseline results show that basic methods like Naive Bayes and KNN proved to be slightly better than random ($\frac{1}{12} = 0.0833$), still, it seems that our neural network approaches did not manage to extract features representative enough to identify the composer and did not improve on this baseline by much.

The task of identifying a composer is a very complex one. When testing the models with samples never seen we expect the algorithm to recognize not the sample, but the style of the composer. We saw from the data analysis that some composers used for this project have very similar pieces, and can be easily misclassified. One of the possible directions that we could take with future work is to have as input multiple Mel-spectrogram and therefore have more time information. Another thing that we could do is to set an upper bound on the possible accuracy, we could do that by training a simpler neural network on the MIDI files. Finally, another approach would be to make use of transfer learning, in which first we would train a model to obtain the "transcription" of the piece, having as target the MIDI file. Once this model is trained, stack another model after that one to perform the classification task.

References

[1] Keunwoo Choi, György Fazekas, Mark Sandler, and Kyunghyun Cho. Convolutional recurrent neural networks for music classification. In *2017 IEEE International Conference on Acoustics,*

Speech and Signal Processing (ICASSP), pages 2392–2396. IEEE, 2017.

[2] Zhouyu Fu, Guojun Lu, Kai Ming Ting, and Dengsheng Zhang. Learning naive bayes classifiers for music classification and retrieval. In *2010 20th International Conference on Pattern Recognition*, pages 4589–4592. IEEE, 2010.

[3] Curtis Hawthorne, Andriy Stasyuk, Adam Roberts, Ian Simon, Cheng-Zhi Anna Huang, Sander Dieleman, Erich Elsen, Jesse Engel, and Douglas Eck. Enabling factorized piano music modeling and generation with the MAESTRO dataset. In *International Conference on Learning Representations*, 2019.

[4] Maximos A Kaliakatsos-Papakostas, Michael G Epitropakis, and Michael N Vrahatis. Musical composer identification through probabilistic and feedforward neural networks. In *European Conference on the Applications of Evolutionary Computation*, pages 411–420. Springer, 2010.

[5] Yi-Wen Liu and Eleanor Selfridge-Field. Modeling music as markov chains: Composer identification, 2002.

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

[7] Zain Nasrullah and Yue Zhao. Music artist classification with convolutional recurrent neural networks. *arXiv preprint arXiv:1901.04555*, 2019.

[8] Stan Salvador and Philip Chan. Toward accurate dynamic time warping in linear time and space. *Intelligent Data Analysis*, 11(5):561–580, 2007.

[9] Yoppy Sazaki. Rock genre classification using k-nearest neighbor. *ICON-CSE*, 1(1):81–84, 2015.

6 Peer review suggestions

Here are some of the suggestions we took from the peer reviews received from the draft those were taken into account for the final version:

- *To make it more relevant to the course outcomes: add similarities/differences between composer recognition and speaker/speech recognition.*
  Added in the introduction.
- *The descriptions of points 2, 3 and 4 on the list of related work are however directly copied from the abstract of the respective papers. This counts as plagiarism and is unacceptable in a report like this. If taking a formulation directly from a different paper seems necessary or appropriate, quotation marks should be used.*
  Related work changed.
- *the discussion of data sets should include some review on how widespread the composition style within the oeuvre of a composer can be. The implicit assumption that a composer has one single style which is distinct to him, is not justified.*
  Most of the composers belong to baroque, classical and romantic eras, so they do have similar styles but our hypothesis was that they had some unique traits in their pieces that would distinguish them.
- *It would be nice if you wrote how much of the data was used in the training, validation and test data set.*
  Done in the Data section, in Methods.
- *I do not think that I have ever seen one that has previous studies listed in a bullet form. These are usually in the form of a flowing text.*
  Related work changed.
- *The abstract does not say that you are actually trying to classify the composers.*
  It does now.
- *Gianluca Micchi's paper seems to be from 2018, not 2017.*
  Citation corrected.