Automated composer recognition for multi-voice piano compositions using rhythmic features, n -grams and modified cortical algorithms

Nadine Hajj² · Maurice Filo¹ · Mariette Awad²

Received: 6 March 2016 / Accepted: 19 July 2017 / Published online: 8 August 2017
© The Author(s) 2017. This article is an open access publication

Abstract With the explosive growth of digital music data being stored and easily reachable on the cloud, as well as the increased interest in affective and cognitive computing, identifying composers based on their musical work is an interesting challenge for machine learning and artificial intelligence to explore. Capturing style and recognizing music composers have always been perceived reserved for trained musical ears. While there have been many researchers targeting music genre classification for improved recommendation systems and listener experience, few works have addressed automatic recognition of classical piano composers as proposed in this paper. This paper discusses the applicability of n -grams on MIDI music scores coupled with rhythmic features for feature extraction specifically of multi-voice scores. In addition, cortical algorithms (CA) are adapted to reduce the large feature set obtained as well as to efficiently identify composers in a supervised manner. When used to classify unknown composers and capture different styles, our proposed approach achieved a recognition rate of 94.4% on a home grown database of 1197 pieces with only 0.1% of the 231,542 generated features—which motivates follow-on research. The retained most significant features, indeed, provided interesting conclusions on capturing music style of piano composers.

Keywords Composer recognition · n -grams · Cortical algorithms

Introduction

The exponential growth of the music database easily accessible and stored in the cloud promotes automated classification of musical pieces for improved user experience and recommender systems. For classical music, the uniqueness of each classical music composer is revealed by his/her own distinct style: a style that is characterized by various factors such as the rhythmic structure that defines the speed and variations of the music piece, the pitch that establishes the melody characterizing the music as being joyful, passionate, dramatic, intense, etc. Often, these various styles are the consequences of either the lifestyle of the composers or the time era they lived in. Our work focuses on revealing the structures in piano compositions that define the style of the composer. This allows us to move a step closer to what actually defines music style that distinguishes one piano composer from the other. Ideally, style identification can help music theorists, recommender systems, music annotation and generation, as well as guide future composers in writing music with the style of classical musicians such as Beethoven, Mozart, Bach... Although classical composer identification is often perceived as reserved to trained musical ears, an automated system capable of recognizing composers given raw data is desirable as it suppresses the need for a professional to perform such task. Recent studies involving self-acclaimed experts show a low recognition of 48% [2]. Additionally, given the inherently large aspect of music data, automating this task becomes a necessity rather than a luxury. This task is the focus of this paper which proposes an automated artist recognition system based on a unique framework for feature generation, feature reduction and supervised classification.

✉ Mariette Awad
mariette.awad@aub.edu.lb

Nadine Hajj
njh05@aub.edu.lb

Maurice Filo
maurice.g.filo@gmail.com

¹ Department of Mechanical Engineering, University of California Santa Barbara, Santa Barbara, CA 93117, USA

² Department of Electrical and Computer Engineering, American University of Beirut, Beirut, Lebanon

In general, the musical composition can be either classified as a recording or a symbolic score. A recording can be in the form of an audio file or a real-time sequenced MIDI file. Generating music data as recordings encodes two coupled sets of information: (a) composer style and (b) performer interpretation of that particular composition. Decoupling the two sets of information is a different problem by itself. Since this paper targets the composer recognition problem, the music data used here are step-by-step sequenced MIDI files, which are basically digital symbolic scores obtained from the Humdrum project library [19]. This avoids the interference of the performer interpretation during the composer recognition process since performance is different than style. After all, a Beethoven composition for instance can be interpreted very differently by Vladimir Horowitz and the Beatles. The remainder of this paper is organized such that section “Literature review” reviews previous related works on composer and performer recognition. Section “Feature extraction” illustrates our proposed algorithm for feature generation using music n -grams. Section “Cortical algorithm for feature reduction and classification” describes the adaptation of cortical algorithms (CA) for feature reduction while section “Experimental results” shows our encouraging experimental results with a discussion on captured music style. Finally, section “Conclusion” concludes with potential follow-on work.

Literature review

Literature survey shows research work that attempted to recognize composers in an automated manner. In what follows, we summarize the most related ones.

One of the techniques used to classify composers is language modeling. The style of each composer is encoded in a language model using n -grams based on the assumption that the main features of a music composer style are captured in some repetitions and recurrences, much like writing styles in language modeling. Such an approach was tested on a corpus of five composers from the Baroque and Classical periods with accuracy attaining 95% [17].

N -grams representations were also used in [35] where three types of n -grams: melodic, rhythmic and combined were constructed. The number of occurrences of each n -gram and a similarity measure was calculated on some MIDI files. A grid search using $n = 6$ gave the best accuracy of 84% on the testing set. It should be noted that each piece of the sets was chosen to be well sequenced, i.e., each track has to represent only one staff or hand. Choosing the pieces that satisfy this criterion was the only human preprocessing task carried out on this data. However, it is more challenging to get the midi file in a single track and perform the analysis accordingly.

Traditional machine learning techniques have also been applied for the task of composer recognition, among which

we cite neural network architectures [6, 18], for their similarity to our recognition model. The system used in [18] is a cascade of a two-layer Stacked Denoising Autoencoder (STA), a two-layer Deep Belief Network and a logistic regression layer on the top. The STA extracts features from raw data by corrupting the clean data by adding some noise or simply randomly setting some elements to 0, then trains the autoencoder with the corrupted data as input and the clean data as target in the end. The DBN is a stack of two layers of Restricted Boltzmann Machines aiming at creating a probabilistic representation of the input while the regression layer labels the input. A recognition rate of 76% was obtained on an in-house dataset. On the other hand Buzzanca [6], used artificial shallow neural networks to recognize the style of music compositions trained using back-propagation. Shared weights as proposed by LeCun et al. [24] allowed for a variable number of inputs which is a must in this case because musical phrases have different lengths. A three-hidden layer network topology was employed and the input layer was divided into sets of four in order to represent the 1/8th note (half beat) with two input nodes encoding the pitch. The first two hidden layers serve as encoders of the rhythm and recognize whether a silent note (rest) or sounded note has been played. 400 pieces for different composers were used to train the network which took around 20 h to complete. Results showed an accuracy of 97.09%.

The sequential aspect of music can be captured using Hidden Markov Models. Previous works adopted this approach on both audio and symbolic data such as [3, 21, 25, 29]. A combination of spectral features such as Spectral Centroid, Rolloff, Flux and Mel-Frequency Cepstral Coefficients, and content specific harmonies are fed into a 4-node Markov chain in [3]. A custom database consisting of 400 distinct audio samples of 30 seconds in length belonging to Bach, Beethoven, Chopin, and Liszt is used to validate the model with recognition rates attaining 83%.

A variant of Hidden Markov Model, the weighted Markov Chain Model, has also been applied to classify composers [21]. Structural information from inter onset and pitch intervals between pairs of consecutive notes is used to build a weighted variation of a first-order Markov chain trained using a proposed evolutionary procedure that automatically tunes the introduced weights. Experimental results on a collection of 150 pieces by Haydn, Mozart and Beethoven showed recognition rates up to 88% for Beethoven–Haydn classification, 70% for Haydn–Mozart classification and 87% for Mozart–Beethoven recognition. Liu and Selfridge-Field [25] used Markov chains by defining the state space as the tonality. Hence, after defining the repertoire of a style, encoding and defining the state space, the transition matrix can then be calculated and, thus, the Markov Chain is constructed. A distance measure as proposed by [10] is used to compare an unlabeled instance to previously constructed chains

and hence identify the composer. Experimental results on a collection of 100 movements of Mozart String quartets and 212 movements from Haydn (MIDI files) showed recognition rates between 52.8 and 68%. Representing music as states of relative pitch and rhythm, Pollastri and Simoncelli [29] used Hidden Markov Models (HMMs) to determine the style of a composer which is identified based on a highest probability scheme. Experimental results based on a dataset including 605 pieces for various artists including classical artists and pop artists showed accuracy of 42%. Validation with human subjects returned 48% accuracy for expert musicians and 24.6% for amateurs.

Other methods modeled artist styles and classify composers. Mearns et al. [26] employed counterpoint as a high level feature from musical theory background to classify the style of musical pieces based on a symbolic representation of the score (Kern Format). Counterpoint is the study of music of two or more voices which sound simultaneously and progress according to a set of rules. A classifier based on a Naive Bayes and Decision Tree correctly labeled 44 of the 66 pieces used for testing belonging to Renaissance and Baroque composers.

Geertzen and van Zaanen [9] applied grammatical inference (GI) to automatically recognize composer based on learning recurring patterns in the musical pieces. An unsupervised machine learning technique extensively used in language recognition, GI reveals underlying structure of symbolic sequential data. For composer recognition, melody and rhythm as features were used to build typical phrases or patterns in an unsupervised manner and rhythm was employed only to perform supervised classification. The GI component in their system was realized by Alignment-Based Learning (ABL) [33]. This technique generates constituents based on syntactic similarity by aligning musical phrases, then selects the best constituents that describes the structure of the musical composition. ABL error rates averaged about 21 on a baroque dataset with preludes from Bach, Chopin and about 24 on a classic dataset with quartet pieces from Beethoven, Haydn and Mozart.

Feature extraction

In any machine learning application, raw data are preprocessed in order to create a numerical representation that can be processed by the recognizer. This phase is referred to as feature extraction and it affects tremendously the machine learning performance [22]. For our particular application, the details of this phase are shown below.

Data extraction

Starting with a single track raw midi file, the feature extraction stage aims to construct a feature vector representing the midi file.

Naturally, the first step is to compile the data from which the features are generated. The ultimate goal is to create a representation of audio files for the performed music when recorded using microphones. This would require performing automatic transcription, i.e., identifying the sequence of notes and their characteristics such as the dynamics, rhythms, pitches, etc. However, the ability to perform automatic transcription is still an immature field due to the fact that advanced auditory signal processing techniques are the focus of researchers worldwide especially for polyphonic music [1, 4, 23, 30, 31]. In addition to that, as we mentioned earlier, a recording of the same musical composition might be very different if interpreted by different performers. Hence to classify a recording based solely on the composer style, one has to develop a scheme to “deconvolve” the interpretation of the performer which becomes a completely different problem. For these reasons, the data set is extracted from MIDI files (representations of the sequential progression of musical notes and their characteristics) using an available Matlab toolbox [8].

The most important parameters representing a musical piece are the pitch and duration of each note in a midi sequence. However, since the same musical piece might be played in a different tonality or even tempo; therefore, a relative representation of the pitch and duration is preferable compared to the absolute representation that was proposed in [23]. For the i th and $(i - 1)$ th notes having a pitch p_i and p_{i-1} and a duration t_i and t_{i-1} , respectively, we construct a representation P_i for the pitch difference and T_i for the time duration difference as shown in Eqs. 1 and 2.

$$P_i = p_i - p_{i-1} \quad (1)$$

$$T_i = \text{round} \left(\log_2 \frac{t_i}{t_{i-1}} \right) \quad (2)$$

Feature generation

After data extraction, features are generated. These features should characterize the data in a compact and efficient form for our purpose. As suggested in [32], the highest pitched notes (skyline) reveal the melody of a certain piece, while the progression of the base notes (groundline) gives a good indication of the style of the composer. Therefore, we start first by extracting the melody and the bass notes and then the corresponding pitch and duration n -grams as proposed in [23]. However, in addition, we propose another set of features which characterizes the rhythmic style of the composer. These features quantize the number of notes that have onsets on a complete beat, half of a beat, quarter of a beat, one third of a beat, etc. This captures the style of the composer in placing notes on beats or between beats (the so called syncopé) or even triplets. We included in section “Illustrative examples” an illustrative example of our proposed feature generation.

Fig. 1 Block diagram for feature generation

Figure 1 shows a detailed setup for the feature generation stage from a midi file. First, the note matrix (see Table 1) is constructed, and then the highest and lowest note sequences are extracted. The note sequences are then filtered based on a computed threshold defined as the average of the centroids of the highest and lowest note sequences. The process is repeated resulting in a representation of the piece consisting of a robust melody and bass extraction. The repetition of the process is particularly useful to categorize a solo note sequence as melody or bass. The final feature vector for the input midi file is a vector containing occurrences of N possible n -grams in addition to the onset beat features we propose. It is important to note that N varies with each midi file. Hence to obtain a fixed length feature vector, automatic zero padding is employed where non-existing n -grams are added with 0 occurrences.

Illustrative examples

In this section, we first illustrate the performance of the melody/bass extraction procedure. Then, we show how the n -grams are constructed.

To show the approach of our melody/bass extraction algorithm, we use (as an illustrative example) the first seventy beats of Bach's Prelude No. 17 in A flat major. Figure 2 shows the piano rolls of Bach's Prelude before and after melody/bass separation.

Next, we illustrate with a simple example the construction of n -grams. We use an evaluation copy of Mozart Software [34] to write the main melody of the Lebanese National Anthem and convert it to a MIDI file. The main melody of the first phrase is shown in Fig. 3 and the corresponding data

Table 1 Note matrix of the sample in Fig. 3

Onset (beats)	Midi pitch	Duration (s)
3	62	0.3271
3.75	62	0.1123
4	67	1.2939
7	69	0.3271
7.75	69	0.1123
8	71	0.4346
9	62	0.3271
9.75	62	0.1123
10	64	0.4346
11	66	0.3271
11.75	66	0.1123
12	67	0.4346
13	62	0.1465
13.333	62	0.1465
13.667	62	0.1465
14	67	0.4346

extracted also known as the note matrix are shown in Table 1. Onsets and Durations (whether in beats or in seconds) give rhythm characteristics, pitches represent the name of the note (60 represents the central C note) and velocity represents the dynamics of the note. The data are generated as follows: unigrams are constructed using Eqs. 1 and 2. Then, pitch and duration n -grams extracted are constructed. For example, 3-grams are formed for the sample mentioned. Pitch (duration) unigrams and 3-grams of the sample are shown in Tables 2 and 3, respectively. Finally, the rhythm feature vector for the sample is constructed and shown in Table 4.

Fig. 2 Piano rolls of Bach's Prelude No. 17 in A flat major. The *top* figure shows the piano roll of the full composition. The *bottom* figure shows the piano roll of the separated melody (*red*) and bass (*blue*)

Fig. 3 Sample of the Lebanese national anthem produced by an evaluation copy of Mozart

Table 2 Pitch 3-grams

Absolute pitch	Relative pitch	3-grams	Occurrence
62	–	–	–
62	0	–	–
67	5	–	–
69	2	'0 5 2'	1
69	0	'5 2 0'	1
71	2	'2 0 2'	1
62	–9	'0 2 –9'	1
62	0	'2 –9 0'	1
64	2	'–9 0 2'	1
66	2	'0 2 2'	1
66	0	'2 2 0'	1
67	1	'2 0 1'	1
62	–5	'0 1 –5'	1
62	0	'1 –5 0'	1
62	0	'–5 0 0'	1
67	5	'0 0 5'	1

Table 3 Duration 3-grams

Absolute duration	Relative duration	3-grams	Occurrence
0.327148	–	–	–
0.112305	–1.5	–	–
1.293945	3.5	–	–
0.327148	–2	'–1.5 3.5 –2'	1
0.112305	–1.5	'3.5 –2 –1.5'	1
0.43457	2	'–2 –1.5 2'	1
0.327148	–0.4	'–1.5 2 –0.4'	2
0.112305	–1.5	'2 –0.4 –1.5'	2
0.43457	2	'–0.4 –1.5 2'	2
0.327148	–0.4	–	–
0.112305	–1.5	–	–
0.43457	2	–	–
0.146484	–1.6	'–1.5 2 –1.6'	1
0.146484	0	'2 –1.6 0'	1
0.146484	0	'–1.6 0 0'	1
0.43457	1.6	'0 0 1.6'	1

Cortical algorithm for feature reduction and classification

The large number of n -grams possible combinations leads to a very large feature vector size. As a result, a feature reduction

scheme is highly desirable to build a robust ML model. In this section, we provide a mathematical formulation for CA as introduced in literature, as well as for our proposed feature reduction scheme.

Table 4 Rhythm features

Rhythm features	Occurrence
Complete beats	10
Half beats	0
Quarter beats	4
1/8th beats	0
1/3rd beats	2

Primer on cortical algorithms

CA are a biologically inspired machine learning (ML) approach which structure (columnar organization) has been proposed by Mountcastle and Edelman [7]. The model mimics aspects of the human visual cortex that recalls sequences stored hierarchically in an invariant. A mathematical formulation and a computational implementation of this model were further developed by Hashmi and Lipasti [13–16]. The model consists of an association of six-layered structures of different thickness constituted of a very large number of columns strongly connected via feed-forward and feedback connections. The column, considered as the basic unit in a cortical network, is constituted by a group of neurons that share the same input. An association of columns is referred to as hyper-column or layer. The connections in the network occur in two directions: vertical connections between columns of consecutive layers, and horizontal connections between columns within the same layer. Though CA can be considered as a type of deep neural networks (DNNs) due to its hexa-layered architecture, two main differences with DNNs can be noted at the levels of structure and training: traditional DNNs employ a feedforward architecture while feedback connections are adopted in CA, and while DNNs rely on a backpropagation algorithm for their training, CA aims at forming unique representations in its supervised learning scheme [12].

A mathematical representation for CA, as we developed based on the description provided in [7], is detailed below.

The nomenclature we adopt in our work is as follows: $W_{i,j,k}^{r,t}$ represents the weight of the connection between the j th neuron of the i th column of layer r and the k th column of the previous layer ($r - 1$) during the training epoch t . For the rest of this paper, scalar entities are represented in italic, vector entities in bold, while underlined variables represent matrices [11]. The weight matrix representing the state of a column composed of M nodes at the training epoch t is

$$\underline{W}_i^{r,t} = [W_{i,1}^{r,t}, W_{i,2}^{r,t}, \dots, W_{i,j}^{r,t}, \dots, W_{i,M}^{r,t}] \quad (3)$$

$W_{i,j}^{r,t}$ represents the weight vector of the connections entering neuron j , of column i in layer r which can be written as:

$$W_{i,j}^{r,t} = [W_{i,j,1}^{r,t}, W_{i,j,2}^{r,t}, \dots, W_{i,j,k}^{r,t}, \dots, W_{i,j,L_{r-1}}^{r,t}]^T \quad (4)$$

L_{r-1} is the number of columns in the layer ($r - 1$) and the subscript T stands for the transpose operator. Expanding $W_i^{r,t}$ yields:

$$\underline{W}_i^{r,t} = \begin{bmatrix} W_{i,1,1}^{r,t} & \dots & W_{i,j,1}^{r,t} & \dots & W_{i,M,1}^{r,t} \\ \vdots & \dots & \vdots & \dots & \vdots \\ W_{i,1,k}^{r,t} & \dots & W_{i,j,k}^{r,t} & \dots & W_{i,M,k}^{r,t} \\ \vdots & \dots & \vdots & \dots & \vdots \\ W_{i,1,L_{r-1}}^{r,t} & \dots & W_{i,j,L_{r-1}}^{r,t} & \dots & W_{i,M,L_{r-1}}^{r,t} \end{bmatrix} \quad (5)$$

Defining $Z^{r,t}$ as the output vector of layer r for epoch t and $Z_i^{r,t}$ the output of column i , we can write

$$Z^{r,t} = [Z_1^{r,t}, Z_2^{r,t}, \dots, Z_i^{r,t}, \dots, Z_{L_r}^{r,t}]^T \quad (6)$$

The output of a neuron is the result of the nonlinear activation function $f(\cdot)$ in response to the weighted sum of the connections entering the neuron. The output of a mini-column is defined as the sum of the outputs of the neurons in this column:

$$Z_i^{r,t} = \sum_{j=1}^M Z_{i,j}^{r,t}; Z_{i,j}^{r,t} = f\left(\sum_k W_{i,j,k}^{r,t} Z_k^{r-1,t}\right) \quad (7)$$

$Z_{i,j}^{r,t}$ is the output of the j th neuron constituting the i th column of the r th layer and the activation function is defined as follows:

$$\begin{cases} f(v) = \frac{1}{1 + \exp[-v \times (\phi(v) + \tau)]} \\ \phi(v) = \begin{cases} -2 & \text{if } v = 1 \\ v & \text{otherwise} \end{cases} \end{cases} \quad (8)$$

Note that Z^0 refers to the input vector $X = [X_1, \dots, X_D]$ and we denote by D the size of the input vector which corresponds to the number of features or the data dimensionality. τ is a tolerance parameter empirically selected and it is assumed constant for all epochs and columns. This parameter sets the tolerance of the activation function to large input values, the larger this value is the less sensitive the activation function becomes.

The training of a cortical network has three phases:

1. Random initialization: Initially, all the synaptic weights are initialized to random values that are very close to 0, so that no preference for any particular pattern is shown.

2. Unsupervised feed-forward: The feedforward learning has the objective of training minicolumns to identify features not detected by others. When the random firing of a particular column coincides with a particular input pattern, this activation is enforced, through strengthening of the column weights and inhibiting of neighboring columns. The inhibiting and strengthening weights update rules are shown in Eqs. 9 and 10, respectively.

$$W_{i,j,k}^{r,t+1} = Z_k^{r-1,t} \times \left(W_{i,j,k}^{r,t} - \Omega(W_i^{r,t}) \right) \quad (9)$$

Equation 9 suggests that when a node is inhibited, a value that is proportional to its input $Z_k^{r-1,t}$ (which corresponds to the output of the column connected to it) is subtracted from the corresponding weight value.

$$W_{i,j,k}^{r,t+1} = Z_k^{r-1,t} \times \left(W_{i,j,k}^{r,t} + C_{i,j,k}^{r,t} + \frac{\rho}{1 + \exp \frac{W_{i,j,k}^{r,t} - \tau}{\Omega(W_i^{r,t})}} \right) \quad (10)$$

where $\Omega(W_i^{r,t})$ is given by:

$$\Omega(W_i^{r,t}) = \sum_{j=1}^M \sum_{k=1}^{L_{r-1}} C_{i,j,k}^{r,t} W_{i,j,k}^{r,t}; C_{i,j,k}^{r,t} = \begin{cases} 1 & \text{if } W_{i,j,k}^{r,t} > \epsilon \\ 0 & \text{otherwise} \end{cases}$$

where ρ is a tuning parameter determining the amount of strengthening to add to a particular weight, the larger ρ , the stronger the strengthening. Smaller values of ρ may lead to a lengthier training as minimal change is obtained during strengthening while large values may drive the network towards instability. ϵ is empirically chosen as the firing threshold assumed constant for all epochs and columns. Similarly, Eq. 10 shows that strengthening a connection occurs by adding a value proportional to the input weight and in the form of an exponential function of the weight itself. With repeated exposure, the network extracts features of the input data, by training the columns to fire for specific patterns.

3. Supervised feedback: The supervised feedback has a goal to correct misclassifications of the same pattern. Another variation of the same pattern that is quite different from the previous one may lead to a misclassification when the top layer columns that are supposed to fire for that pattern do not fire. To reach a desired firing scheme over multiple exposures also known as “stable activation”, the error occurring at the top layer generates a feedback signal forcing the column firing for the original pattern to fire

and also inhibiting the column that is firing for the new variation. The feedback signal is propagated back to the previous layer once the top level columns start to give a stable activation for all variations of the same pattern. The feedback signal is sent to preceding layers only once the error in the layer concerned converges to a value below a certain pre-defined tolerance threshold. Each layer is trained until convergence criteria expressed as an error term in function of the actual output and a desired output (scheme of firing) is reached.

Cortical algorithm for feature reduction

One of the advantages of CA is its ability to extract and learn the aspects of the data in a hierarchical manner. This makes CA a possible candidate for feature extraction or reduction, which to the best of our knowledge has not been investigated yet [22]. As described previously, one goal of CA training is to train cortical columns to fire for particular aspects of the patterns during the feedforward unsupervised learning. In addition, during the feedback supervised learning, invariant representations are created on all levels of the network in a hierarchical manner. As a result, a repetitive firing scheme is observed for a particular class of the data. The weights of the input layer directly connected to the feature vector are, therefore, an indication of the significance of the corresponding features. Our method employs hereby denoted by CA-FR, the input layer weights to rank the feature vector in terms of significance, discarding the lowest ranked features, allowing for a higher discriminative performance of the algorithm and a robust model of the raw data. The following algorithm illustrates our approach hereby denoted by CA-FR:

1. Train the cortical network using all features provided.
2. For all input nodes calculate average weight associated with each feature.
3. Rank features in a decreasing order according to their associated weights.
4. Eliminate a pre-defined percentage of the lowest ranked features (heuristically chosen).

Experimental results

Dataset

In this section, we summarize the experimental results obtained. We perform our experiments on a database of note files we compiled from the Humdrum project library [19] described in Table 5. The era of each composer as well as the total number of beats for each composer are also shown in this table.

Table 5 Dataset description

Composer	Area	No. of files	Total size (MB)	Total number of beats
Bach	Baroque	246	4.3	179,259
Beethoven	Classical-romantic	185	10.6	490,485
Chopin	Romantic	87	2.13	97,355
Corelli	Baroque	246	3.6	146,009
Haydn	Classical	212	7.5	338,763
Joplin	Ragtime	45	1.37	66,043
Mozart	Classical	82	3.36	151,912
Scarlatti	Baroque	59	1.3	60,849
Vivaldi	Baroque	35	1.23	55,544

Experiments

Our experiments are designed to test the features we proposed, our feature reduction technique, as well as CA as a classification technique for the recognition of artists. Input weights of the network were used to discern the least significant features (corresponding to the weakest input weights). A fraction of these features is eliminated at each iteration. Thus, we perform four experiments using our dataset:

- Experiment 1: Using our proposed feature vector, and our feature reduction scheme (CA-FR).
- Experiment 2: Using only the n -grams features (as widely proposed in the literature) and our feature reduction algorithm (CA-FR).
- Experiment 3: Using the entire feature vector proposed and a principal component analysis (PCA) for feature reduction.
- Experiment 4: Using only the n -grams features and a PCA for feature reduction.

For each of the four experiments, we subsequently eliminate a fraction of the least significant features. The resulting subsets are fed to three recognizers: CA, SVM one vs. all using Gaussian Radial Basis Function (RBF) kernel and a one nearest neighbor classifier (1NN). SVM and 1NN were chosen for comparison purposes since they are widely adopted in a multitude of machine learning applications [27]. We evaluate the performance of these classifiers in terms of recognition rate. We used a cortical network constituted of 6 layers starting with 2000 columns in the first layer and decreasing this number by half between consecutive layers, each column consisting of 20 nodes. This structure was experimentally proven to perform well on a variety of databases [11]. Simulations were executed using Matlab R2011a, on an Intel core i7 CPU at 2 GHz under Windows 7 operating systems. The CA network is deployed using the CNS library, a framework for simulating cortically organized networks, developed by MIT [28].

Table 7 shows the recognition rates obtained using the three classifiers described above for different percentages of the initial dataset reduced using PCA and CA-FR as described in the experimental setup, while Table 8 shows the required training time. The best recognition rate for each experiment is shown in bold. All results are based on a four-fold cross validation scheme. Table 9 shows the recognition rate obtained using experiment 1 settings on 0.1% of the features for a 2 two-composer classification task. Composers are arranged according to their year of birth to reflect their closeness in eras. In addition, Table 10 provides the mean, standard deviation, maximum and minimum recognition rate obtained for every experiment (for all reduction rates) as well as the overall statistic for every classifier. Table 6 compiles results from recent work on these composers.

The following can be concluded from our experimental results. Using the entire dataset with the proposed added features, the best recognition rate achieved is 88.4% using CA as classifier compared to 70.3 and 60.6% using SVM and NN, respectively, and thus an improvement of CA of 18.1 and 27.8% over SVM and NN, respectively.

Using only 0.1% of the features including the proposed attributes, the best performance achieved is 94.9% using CA as a classifier, 72.8% using SVM and 62.2% using NN. This is a strong indicator of the superior performance of CA's classification compared to classical pattern recognition classifiers.

For the same reduction rate of 99.9%, our proposed added features introduced an improvement from 86.4 to 90.4% using PCA and from 91.1 to 94.9% using CA-FR, showing the significance of our proposed added features in discriminating composers. In other terms, an improvement of 4% is obtained using our proposed features and PCA for feature reduction (Exp. 3 vs. Exp. 4) and 3.8% using our proposed features and CA-FR for feature reduction (Exp.1 vs. Exp. 2)

For the same percentage of features (0.1%), the best recognition rate obtained is 94.9% using CA as classifier and our feature reduction algorithm while PCA led to an accuracy of 90.4% for the same reduction rate. This shows CA-FR's

Table 6 State of the art results

Reference	Task	Recognition rate
[17]	Bach vs. Haydn	96.8
	Bach vs. Mozart	97.5
	Haydn vs. Mozart	74.7
[20]	Bach vs. Beethoven	81.2
	Bach vs. Chopin	87.0
	Bach vs. Haydn	69.3
	Bach vs. Mozart	82.2
	Beethoven vs. Chopin	63.8
	Beethoven vs. Haydn	65.0
	Beethoven vs. Mozart	69.8
	Chopin vs. Haydn	77.8
	Chopin vs. Mozart	77.0
	Haydn vs. Mozart	55.8
	Haydn vs. not Haydn	40
[18]	Bach vs. not Bach	93.1
	Beethoven vs. not Beethoven	63.3
	Chopin vs. not Chopin	98.11
	Mozart vs. not Mozart	74.6
	Vivaldi vs. not Vivaldi	87.0

Table 7 Recognition rates comparison

Exp.	Classifier	% of features					
		100	80	50	20	5	0.1
Exp. 1	CA	76.6	83.5	90.6	92.3	92.6	94.9
	SVM	67.8	69	69.5	70.8	72	72.8
	1NN	57.9	59.8	60.5	61	62.2	62.2
Exp. 2	CA	73.6	78.5	83.6	88.7	90.5	91.1
	SVM	61.6	64.5	65.4	67.8	65.8	67.3
	1NN	53.8	56	57.4	57.1	57.2	56.4
Exp. 3	CA	76	77.7	82.8	88.6	89.9	90.4
	SVM	69.6	68.1	69.1	71.4	69.7	70.5
	1NN	61	59.9	61	61.1	61	60.1
Exp. 4	CA	73.8	76.3	78.5	87.1	87.8	86.4
	SVM	61.9	63.5	66.3	68.2	66.3	64.9
	1NN	54.3	53.4	58.8	59.4	58.3	57.7

Bold values indicate best values

ability in extracting the most significant features while discarding the less important ones.

Using only the n -grams features and our feature reduction technique resulted in a slightly better performance compared with using PCA on the entire set, indicating the strong capacity of CA-FR of extracting meaningful information and creating discriminative representations of the input data.

Comparing feature reduction techniques, the maximal improvement of CA-FR vs. PCA is of 4.9% (94.4 vs. 90.4%) observed on the entire set and 3.3% (91.1 vs. 87.8%) on

Table 8 Training time comparison

Exp.	Classifier	% of features					
		100	80	50	20	5	0.1
Exp. 1	CA	180	135	80	55	37	21
	SVM	28	25	22	18	13	9
Exp. 2	CA	176	127	76	51	33	17
	SVM	24	17	14	10	9	1
Exp. 3	CA	240	210	110	130	82	96
	SVM	103	85	52	93	73	39
Exp. 4	CA	228	200	102	120	70	86
	SVM	93	73	40	85	65	31

the n -grams attributes. In average, using CA-FR for feature reduction resulted in an accuracy of 88.4% compared to 84.2% using PCA, i.e., an improvement of 4.2%.

Evaluating the significance of the proposed features, the maximal accuracy attained using only n -gram features is 91.8% compared to 94.9% using our proposed feature vector.

The effect of the feature reduction stage is most shown in the classification results: an improvement in accuracy is overall noticed in experiments employing our feature reduction algorithm, a smaller effect of feature reduction can be seen for PCA which shows the robustness of our algorithm in discerning significant features.

While a nearest neighbor classifier requires practically zero training time, the high computational demand of computing distances (specially for high dimensional data) renders its use in an online scheme quite challenging. Conversely while CA and SVM require lengthy training, the testing complexity of both these algorithms is relatively low (SVM only requires the computation of the equation of a line while CA requires only the propagation of the input through the network) and hence are more suitable for an online framework while the training can be performed offline.

Comparing our proposed method with the state of the art results shown in Table 6, a notable improvement in recognition rate can be observed in all one-to-one tasks. This is further intensified in tasks involving composers within the same era such as Haydn and Mozart where the recognition rate saw an increase from 74.7 to 82.9% as well as Beethoven and Chopin where our algorithm achieved 86.1% compared to 63.8% reported in the state of the art.

Comparing classifiers, one can see the higher ability of CA to robustly classify composers by analyzing Table 10: while CA's higher standard deviation may be regarded as a drawback, its lowest recognition rates rival the best performances of SVM and 1NN consistently for all experiments, hence justifying its use as a classifier.

Comparing the recognition rates observed for one-to-one composer classification tasks, one can see the robustness

Table 9 Recognition rates for one-to-one composer classification

	Corelli	Vivaldi	Bach	Scarlatti	Haydn	Mozart	Beethoven	Chopin	Joplin
Corelli (1653)	–	94.8	94.3	95.5	97.5	96.8	92.4	99.2	98.8
Vivaldi (1678)		–	96.4	93.9	97	96.5	98.2	95.1	99.5
Bach (1685)			–	95.2	97.8	98.7	97.1	96.5	98.9
Scarlatti (1685)				–	96.7	96.8	96.3	95.5	99.2
Haydn (1732)					–	82.9	90.7	98.6	99.1
Mozart (1756)						–	87.9	94.7	98.7
Beethoven (1770)							–	86.1	97.6
Chopin (1810)								–	98.7
Joplin (1868)									–

Table 10 Statistical comparison of experiments

Metric	Exp. 1			Exp. 2			Exp. 3			Exp. 4		
	CA	SVM	1NN	CA	SVM	1NN	CA	SVM	1NN	CA	SVM	1NN
Mean	88.4	70.3	60.6	84.3	65.4	56.3	84.2	69.7	60.7	81.7	65.2	57
Std.	7	1.9	1.6	7.1	2.2	1.3	6.4	1.1	0.5	6.2	2.2	2.5
Max	94.9	72.8	62.2	91.1	67.8	57.4	90.4	71.4	61.1	87.8	68.2	59.4
Min	76.6	67.8	57.9	73.6	61.6	53.8	76	68.1	59.9	73.8	61.9	53.4

of our proposed method: the lowest reported accuracy is at 86.1% for a Chopin vs. Beethoven task compared with a 63.8% reported in [20], while the best performance marked a 99.5% accuracy classifying Vivaldi vs. Joplin. Overall, our proposed feature set and feature reduction system showed to be powerful for the recognition of music composers with a 94.4% recognition rate obtained using 0.1% of the entire features (n -grams and proposed attributes), CA for feature reduction and classification.

From a musical perspective

A deeper look on our proposed framework can be taken by investigating the retained features for the best performance obtained, i.e., for Exp. 1 at 99.9% reduction. As indicated by the input weights of the network, the five most significant features are the ones corresponding to the occurrences of notes with onset on quarter beats, occurrences of notes with onset on complete beats, occurrence of notes with onset on half beats, and 2 n -gram combinations corresponding to bass notes; all of which are proposed by this work. From a music perspective, these results propose that piano composers mostly distinguish their style via the way they deal with bass notes on one hand, and beats/onsets on the other hand. Obviously, melodies play an important role in revealing the style of a particular composer. However, according to our work, the spreading of notes over beats/onsets, together with the development of bass patterns that accompany the melody, has a lot more to say about the style of a particular composer. It is well known that humans, in general, are

biased to perceive high pitched melodies much more than the accompanied bass notes. However, given any melody, different composers develop different beats and different bass notes which harmonize with the same melody. This is particularly what distinguishes the style of a composer as the retained most significant features propose. Furthermore, looking at the results of one-to-one classification (Table 9), confusion rates ranged from 17.1% (Mozart vs. Haydn) to 0.5% (Vivaldi vs. Joplin) with Joplin being the least confused composer, due to his belonging to a different era than all other composers. The least accurate prediction rates were found for the tasks of distinguishing Haydn vs. Mozart, and Beethoven vs. Chopin, consistent with the similarities in style between the mentioned composers. In fact, it is known in the music society that Haydn and Mozart were friends with the latter being inspired by the former [5]. Additionally, while both Beethoven and Chopin have their own styles, some pieces of these composers possess some similar features (particularly, Beethoven's Adagio of op. 106 or the Arioso in op. 110 has a chromatic style, a feature Chopin has been known for).

Conclusion

In this paper, we have presented an automated identification framework for the recognition of classical music composers using a novel n -gram based feature extraction technique along with a cortical algorithm for the feature reduction and recognition stages. Our proposed system is a power-

ful scheme inheriting the robustness of feature reduction using CA and the discriminative capacity of our proposed feature. Experimental results show that with only 0.1% of the 231542 feature set as identified by CA, our proposed approach achieved a recognition rate of 94.4% on a dataset of 1197 pieces belonging to 9 composers which motivate testing on larger datasets.

Acknowledgements This work is funded by the University Board at the American University of Beirut.

Open Access This article is distributed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits unrestricted use, distribution, and reproduction in any medium, provided you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made.

References

1. Abdallah SA, Plumbley MD (2003) An independent component analysis approach to automatic music transcription. In: Proceedings of the 114th AES convention, Amsterdam, The Netherlands, 22–25 March 2003
2. Abeler N (2015) Musical composer identification MUS-15
3. Bartle A (2012) Composer identification of digital audio modeling content specific features through Markov models
4. Bello JP, Monti G, Sandler MB et al (2000) Techniques for automatic music transcription. In: Proceedings of the first international symposium on music information retrieval (ISMIR-00), Plymouth, Massachusetts, USA
5. Brown P (1992) Haydn and Mozart's 1773 stay in Vienna: weeding a musicological garden. *J Musicol* 10:192–230
6. Buzzanca G (2002) A supervised learning approach to musical style recognition. In: Music and artificial intelligence. Additional proceedings of the second international conference, ICMIAI, vol 2002, p 167
7. Edelman GM, Mountcastle VB (1978) The mindful brain: cortical organization and the group-selective theory of higher brain function. Massachusetts Institute of Technology, Cambridge
8. Eerola T, Toiviainen P (2004) Midi toolbox: Matlab tools for music research. University of Jyväskylä, Jyväskylä, Finland, p 14
9. Geertzen J, van Zaanen M (2008) Composer classification using grammatical inference. In: Proceedings of the MML 2008 international workshop on machine learning and music held in conjunction with ICML/COLT/UAI, pp 17–18
10. Grachten M, Widmer G (2009) Who is who in the end? recognizing pianists by their final ritardandi. In: ISMIR. Citeseer, pp 51–56
11. Hajj N, Awad M (2013) Weighted entropy cortical algorithms for isolated Arabic speech recognition. In: The 2013 international joint conference on neural networks (IJCNN). IEEE, pp 1–7
12. Hajj N, Rizk Y, Awad M (2015) A mapreduce cortical algorithms implementation for unsupervised learning of big data. *Proc Comput Sci* 53:327–334
13. Hashmi A, Lipasti MH (2010) Discovering cortical algorithms. In: International joint conference on computational intelligence (International conference on fuzzy computation and international conference on neural computation), pp 196–204
14. Hashmi A, Berry H, Temam O, Lipasti M (2011) Automatic abstraction and fault tolerance in cortical microarchitectures. In: ACM SIGARCH computer architecture news, vol 39. ACM, pp 1–10
15. Hashmi A, Nere A, Thomas JJ, Lipasti M (2011) A case for neuro-morphic ISAs. *ACM SIGARCH Comput Arch News* 39(1):145–158
16. Hashmi AG, Lipasti MH (2009) Cortical columns: building blocks for intelligent systems. In: IEEE symposium on computational intelligence for multimedia signal and vision processing, CIMSVP'09. IEEE, pp 21–28
17. Hontanilla M, Pérez-Sancho C, Inesta JM (2013) Modeling musical style with language models for composer recognition. In: Pattern recognition and image analysis. Springer, Berlin, pp 740–748
18. Hu Z, Fu K, Zhang C (2013) Audio classical composer identification by deep neural network. [arXiv:1301.3195](https://arxiv.org/abs/1301.3195)
19. Huron D (1997) Humdrum and kern: selective feature encoding. In: Beyond MIDI. MIT Press, Cambridge, pp 375–401
20. Kaliakatsos-Papakostas MA, Epitropakis MG, Vrahatis MN (2010) Musical composer identification through probabilistic and feedforward neural networks. In: European conference on the applications of evolutionary computation. Springer, New York, pp 411–420
21. Kaliakatsos-Papakostas MA, Epitropakis MG, Vrahatis MN (2011) Weighted Markov chain model for musical composer identification. In: Applications of evolutionary computation. Springer, New York, pp 334–343
22. Khanna R, Awad M (2015) Efficient learning machines: theories, concepts, and applications for engineers and system designers. Apress, New York
23. Klapuri AP (2004) Automatic music transcription as we know it today. *J New Music Res* 33(3):269–282
24. LeCun Y et al (1989) Generalization and network design strategies. Connections in perspective. North-Holland, Amsterdam
25. Liu Y-W, Selfridge-Field E (2002) Modeling music as Markov chains: composer identification. <https://ccrma.stanford.edu/~jacobliu/254report/>
26. Mearns L, Tidhar D, Dixon S (2010) Characterisation of composer style using high-level musical features. In: Proceedings of 3rd international workshop on machine learning and music. ACM, pp 37–40
27. Michalski RS, Carbonell JG, Mitchell TM (2013) Machine learning: an artificial intelligence approach. Springer Science & Business Media, New York
28. Mutch J, Knoblich U, Poggio T (2010) CNS: a GPU-based framework for simulating cortically-organized networks. Massachusetts Institute of Technology, Cambridge, Tech. Rep. MIT-CSAIL-TR-2010-013/CBCL-286
29. Pollastri E, Simoncelli G (2001) Classification of melodies by composer with hidden Markov models. In: Proceedings of first international conference on web delivering of music. IEEE, pp 88–95
30. Raphael C (2002) Automatic transcription of piano music. In: Proceedings of the international conference on music information retrieval (ISMIR), Paris, France
31. Ryyänen MP, Klapuri A (2005) Polyphonic music transcription using note event modeling. In: IEEE workshop on applications of signal processing to audio and acoustics. IEEE, pp 319–322
32. Uitdenbogerd A, Zobel J (1999) Melodic matching techniques for large music databases. In: Proceedings of the seventh ACM international conference on multimedia (Part 1). ACM, pp 57–66
33. Van Zaanen MM (2002) Bootstrapping structure into language: alignment-based learning. [arXiv:cs/0205025](https://arxiv.org/abs/cs/0205025)
34. Webber D (2015) Mozart music software. <http://www.mozart.co.uk/>
35. Wołkowicz J, Kulka Z, Keşelj V (2008) N-gram-based approach to composer recognition. *Arch Acoust* 33(1):43–55