Composing Music with Neural Networks and Probabilistic Finite-State Machines

Tomasz Oliwa and Markus Wagner

Artificial Intelligence Center, University of Georgia
Institute for Computer Science, University of Koblenz-Landau
oliwa@uga.edu, wagnermar@uni-koblenz.de

Abstract. In this paper, biological (human) music composition systems based on Time Delay Neural Networks and Ward Nets and on a probabilistic Finite-State Machine will be presented. The systems acquire musical knowledge by inductive learning and are able to produce complete musical scores for multiple instruments and actual music in the MIDI format. The quality of our approaches is analyzed in objective and subjective manner with existing techniques.

Key words: Biological Inspired Music, Music Composition, Representation Techniques, Comparative Analysis, Time Delay Neural Networks, Finite State Machines, Inductive Learning

1 Introduction

Artificial music composition systems have been created in the past using various paradigms. Approaches using Recurrent Neural Networks [7] and Long-Short Term Memory (LSTM) Neural Networks [3] architectures to learn from a dataset of music and to create new instances based on the learned information have been taken as well as approaches with genetic algorithms. The latter ones focus on semi-objective [9], i.e. combined computational and human evaluation of the songs, or fully objective fitness functions [8] to generate new songs. Associative Memories have also been tried [5] using a context-sensitive grammar.

Classical Algorithm-based automatic music composition systems, which aim at following predefined rules to construct music, stand or fall by human implementation of the underlying algorithms, which leaves the cumbersome task to derive sets of musical creation rules completely to the human designer. Another approach is to modify existing melodies by applying specific noise function to create new melodies [4], thus possibly reducing the dominance of the human factor. Heuristic Search Algorithms like Genetic Algorithms, on the other side, suffer from the fitness bottleneck [2][6], a gigantic, and in terms of music mostly unusable, search space.

Our machine learning systems extract important features/key elements from a dataset of music (created by humans) and are able to produce new song material which inherits these ideas. They compose music based on the extracted information gained by inductive learning. In both of our following approaches, we

use machine learning techniques for the feature selection of musical information from the music database.

1.1 Music Background

Western Music can be defined as the chronology of notes, a note itself by its pitch and length (which we consider as our atomic unit, and with a "note", we always mean the combined information of note length and note pitch).

In classical music theory, a piece of music is written in a specific musical scale, which defines a subset from the set of all possible notes. Our machine learning systems use an unambiguous mapping of notes to our internal representation, which means that every note can be learned, regardless of its pitch or length.

The music which we considered as our music database for the melody where 19 classical and folk songs [1]. As an addition, we included 77 drum patterns in a drum training dataset from Metallica's rock song "Creeping Death" for the finite state machine (FSM) approach to come up with a multi-instrument song.

2 The Finite-State Machine approach

2.1 Stochastic uniform sampling with accumulated probabilities

Our system parsed all songs from the song database and constructed multidimensional hashmaps (our knowledge base), which contain the probability of all possible note sequences and their successors. Figure 1 shows the underlying algorithm. The use of the accumulated probabilities simplifies the search for the successor(s), based on the idea from the stochastic uniform sampling method. The learned subsequent structures are similar to the "grouping structures" [1].

3 Music Representation

The 19 songs in our database were written in *abc* language [10]. Conversion from and to the *abc* language format from the MIDI format can be done using the abcMIDI package [2]. MIDI (Musical Instrument Digital Interface)[3] defines a communications protocol for instruments and computers to transfer data.

3.1 Feature Extraction

An illustrative example of assigning distinct integer values to notes, pauses, triplets etc. of the beginning of the song "Claret and Oysters" by Andy Anderson is shown in Figure 3.1 in the corresponding *abc* language and our integer representation. The richness of this representation stands in contrast to other approaches with a more restricted search space (like [9], which has no pauses or triplets). We found more diversified music because of our richer representation and thus bigger search space.

[1] http://abc.sourceforge.net/abcMIDI/original/MIDI.zip
[2] http://abc.sourceforge.net/abcMIDI/
[3] http://www.midi.org/

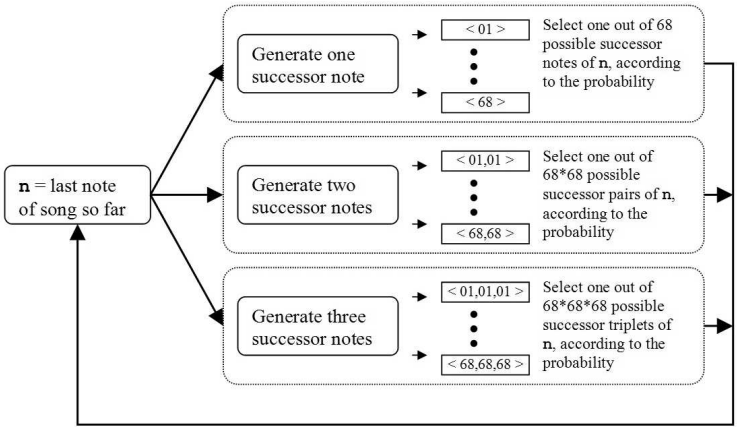

Fig. 1. Example of our algorithm, a Probabilistic FSM for a possible successor (sequence). In practice up to 4 base nodes and 10 successor notes are possible.

```
E2 G2 G2 G F          10, 18, 18, 20, 16,
G2 A2 B2 c2           18, 22, 26, 30,
d2 e2 g e d B         34, 38, 48, 40, 36, 28,
G2 G F G2 A2          18, 20, 16, 18, 22
```

Fig. 2. Beginning of "Claret and Oysters" in the *abc* language (left) and our integer representation (right)

4 The Neural Network approach

An artificial neural network (NN) is a layered network of artificial neurons and is able to learn real-valued functions from examples (in this case, the subsequent nodes). The temporal characteristics of music are exploited with a Time Delay Neural Network (TDNN) architecture, where the input patterns are successively delayed in time. The best result we had was using a Ward Net architecture, as implemented in the NeuroShell 2[4] package and modified it into a TDNN-Ward Net, which is shown in Figure 3. The entire song database was used as the training set.

5 Experimental Results

The reference song "Claret and Oysters" and one song made by the FSM are visualized in Figure 4 with our integer representation, with the integer numbers on the x-axis and the time (sequential numbers) on the y-axis. As can be seen, there exist repeating patterns in the graph, the "landscape" of the song shares similar "hills", for example the notes 50-57 and 80-87.

[4] http://www.wardsystems.com/products.asp?p=neuroshell2

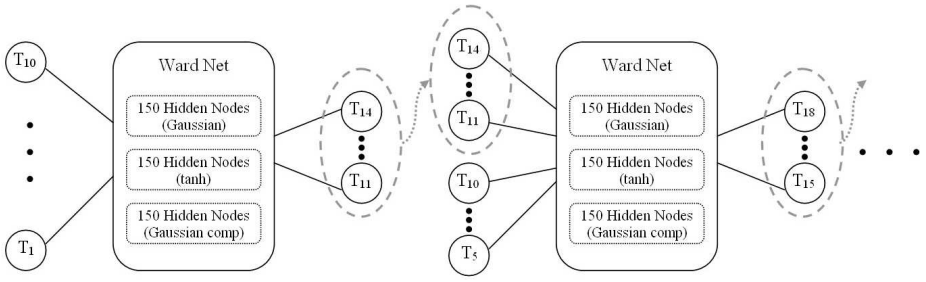

Fig. 3. TDNN illustration using the Ward Net, process of continuously generating four notes, based on the last ten notes, with T_i indicating the i-th note of the song

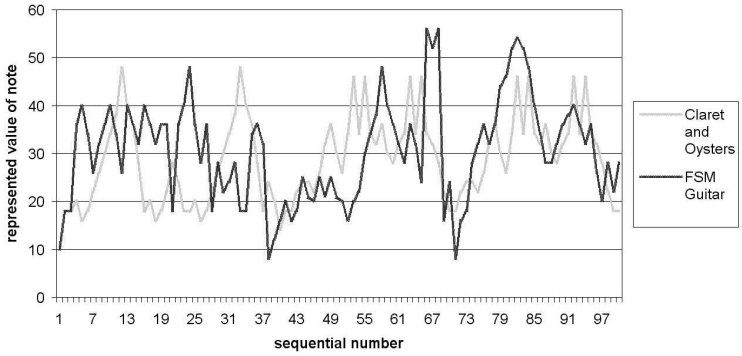

Fig. 4. Visualization of the internal representation

5.1 Results with the FSM

Several smaller and longer learned patterns can be recognized by ear and eye, not only from "Claret and Oysters" in Figure 4, but from different songs of the entire database as well. It is noteworthy that the overall quality of the song does not change over time (in contrast to the NN songs, described in the next section). The beginning of this song is shown in Figure 5 as musical score.

5.2 Results with the NN

In Figure 6, the result from a TDNN-Ward Net, which was trained over 72000 epochs with an average error of 0.0006714 on the training set (the song "Claret and Oysters"), is shown.

In the second half of the song, after a given starting seed, the NN is oscillating more often between extreme notes than in the beginning of the song, it can not predict the song any more. Including multiple songs to the knowledge database did not significantly change the result. That means that even with a large number of epochs the network architecture was not able to memorize the whole song.

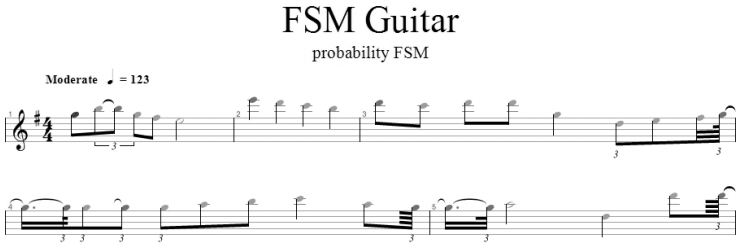

Fig. 5. Sample song generated by the FSM in its musical score

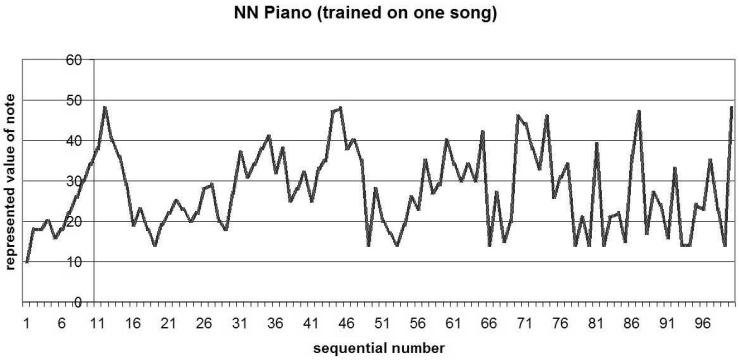

Fig. 6. Sample song generated by the NN, trained on one song (internal representation)

6 Comparison with other approaches

In total, three songs (which feature drums violas, electric guitars played by FSM and TDNN Ward Nets) created by our systems have been made public [5] and we encourage the reader to form his/her own opinion about the music's quality.

6.1 Musical Intervalls - Consonances and Dissonances

Statistics of music intervals, the difference in pitch between two notes, can be provided. In Figure 7, it can be seen that a randomly generated song does not have any preference for any interval. Although higher intervals were used by the FSM occasionally, preferences for medium and low size intervals were observed. "Claret and Oysters" has a preference for lower intervals as well.

7 Conclusion and Future Work

When listening to artificially composed music, critics tend to describe the creations as "..compositions only their mother could love.." [7]. While the local contours normally make sense, the pieces are not musically coherent.

[5] http://www.uni-koblenz.de/~wagnermar/evomusart

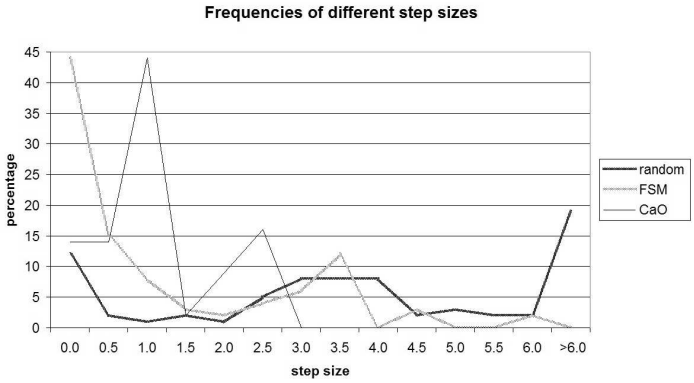

Fig. 7. Frequencies of the intervals of a random generated song, a FSM song and "Claret and Oysters"

Our two advantages are that (a) we learn key elements from analyzed songs and features like distinct scale, arpeggios, musical style expressed through frequent musical patterns and subpatterns are identified and used to create new songs without an explicit human modeling and (b) a bias is induced to produce short coherent sequences at once.

Future research need to recognize and use different coherent structures of a song (like refrain, chorus, solo etc.) A newly composed song would then be structured by the learned musical structures and inherit features which are present only in specific parts of songs, such as the refrain.

References

1. A. Baratè, G. Haus, and L. A. Ludovico. Music analysis and modeling through petri nets. In *CMMR*, volume 3902 of *Lecture Notes in Computer Science*, pages 201–218. Springer, 2005.
2. J. A. Biles. Genjam: A genetic algorithm for generating jazz solos, June 15 1994.
3. D. Eck and J. Schmidhuber. Finding temporal structure in music: Blues improvisation with lstm recurrent networks, 2002.
4. Y.-W. Jeon, I.-K. Lee, and J.-C. Yoon. Generating and modifying melody using editable noise function. In *CMMR*, volume 3902 of *Lecture Notes in Computer Science*, pages 164–168. Springer, 2005.
5. T. Kohonen. A self-learning musical grammar, or "associative memory of the second kind". In *IJCNN*, volume I, pages I–1–I–6, Washington DC, 1989. IEEE.
6. E. R. Miranda and J. A. Biles, editors. *Evolutionary Computer Music*. Springer, March 2007.
7. M. Mozer. Neural network music composition by prediction: Exploring the benefits of psychoacoustic constraints and multi-scale processing, 1994.
8. J. Schoenberger. Genetic algorithms for musical composition with coherency through genotype. Spring 2002.
9. M. Unehara and T. Onisawa. Construction of music composition system with interactive genetic algorithm. Oct, 2003.
10. C. Walshaw. The abc notation system. `http://abcnotation.org.uk`, 1993.