Evolution of Hierarchical Neural Networks for Time-dependent Cognitive Processes: Key Recognition for Musical Compositions

Jaime J. Dávila
Hampshire College
School of Cognitive Science
Amherst, MA 01002
jdavila@hampshire.edu

Abstract- This paper presents the results of using the GENDALC GANN system to evolve neural network topologies for music perception. The results obtained are not only better than those for other typically used neural network topologies, but also better than for neural networks that incorporate music theory knowledge. Because the data and task used in these experiments include hierarchical time dependent processing, these results demonstrate GENDALC's ability to evolve good solutions for cognitive tasks, even while using approaches potentially different from those used by humans.

1 Introduction

The GENDALC (Genetic Evolution of Neuron Distribution and Layer Connectivity) system has proved successful in evolving neural networks (NN) capable of processing natural languages (*Dávila* 1999). Of particular interest is that, while GENDALC has no built-in grammatic knowledge, it is still able to find topologies that efficiently take advantage of the grammatic regularities found in the data set used for training.

This paper presents results of using the GENDALC system to evolve NN topologies for music processing. In particular, networks are asked to classify musical compositions as being written in minor or major key. Results are compared with those obtained by several typically used NN topologies, as well as by NN that have been hardwired with knowledge of music theory.

In the next section I will give an overview of work done by others in order to provide NN with music theory knowledge. I will later present the method used by the GENDALC system to evolve NN topologies and the parameters used in experiments with music processing. I then present the results obtained by the GENDALC system, and compare them to the results obtained by other systems. Finally, I will discuss implications of these results, and propose future research avenues.

2 Basic Neural Network Theory

A neural network (NN) is a computational device loosely based on the human brain. Simple processing units called neurons (also referred to as nodes) are connected among themselves. Each node receives input signals via a series of connections, coming from either other nodes or the outside world. Each connection has a weight associated with it. Figure 1 illustrates a typical neuron, with connection weights of 1.2, -3.7, -2.1 and 2.5. If signal values of 4, -2, -7.3, and -3.5 are placed on the inputs to this node, as illustrated, the total weighted sum seen by the neuron is of 4*(1.2) + 2*(-3.7) + (-7.3) *(-2.1) + (-.35)*(2.5) = 11.855.

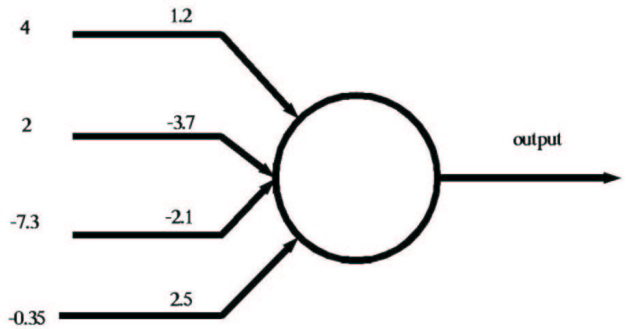

Figure 1: Standard neuron, with sample connection weights

The total weighted input seen by a neuron is fed into a transfer function, and the output of this transfer function is placed on the output of the neuron. Figure 2 illustrates typically used transfer functions.

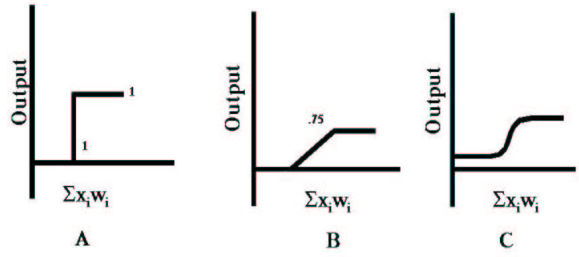

Figure 2: typical transfer functions

The computation performed by any one neuron is very simple, but complex behavior emerges when neurons are connected to each other, forming a network. Instead of having to explicitly program them with a particular data processing algorithm, neural networks have the ability to learn from the data presented to them. Learning in a neural network takes place by the modification of the connection weights between nodes. Backpropagation, the most commonly used method for supervised learning,

modifies weights as to minimize the difference between the intended and actual activation of output nodes (Rummelhart, Hinton, & Williams 1986).

3 NN and Music Processing

Previous research on NN-based music processing can be divided into two broad categories. One presents musical data to self-organizing maps, in order to observe what type of configuration emerges. The other approach is to hardwire neurons to each other in order to encode western musical theory.

As an example of the former category, Taylor and Greenhough (1994) presented an ART neural network (Carpenter & Grossberg 1987a) with sounds from several different instruments, causing the NN to learn to classify the sound sources. Griffith (1994) used ART2 classifiers(Carpenter & Grossberg 1987b) to detect salient characteristics in nursery rhymes.

One of the most complete examples of NN incorporating knowledge of music theory is proposed by Tillmann, Bharucha, and Bigand (Tillmann, Bharucha, & Bigand 2001). They presented a NN with units divided into four layers of twelve units each, representing tones, major chords, minor chords, or keys in western music. Connections among these four layers were set in accordance with elements of one layer being present in elements of the next one. So, for example, the node representing the C major chord was connected to units representing the C, E, and G tones (all of which are part of the C major chord). This same C major chord unit was connected to the C, F, and G key units. Following this pattern, all 48 nodes were connected in both directions, forming a toroid. During activation, the unit for a particular key would become more easily activated if chords and tones that participate in that same key were not only currently active, but had been active in the recent past. Additionally, chord units had their activation decay exponentially through time. This system was intended to provide a model of how musical context could influence processing of future tones, defining a hierarchical relationship through time between notes, chords, and keys. In human brain imaging experiments, the model proposed by Tillmann, Bharucha, and Bigand has been found to have certain similarities with the ways in which human brains seem to be processing music input (Janata et.al. 2002).

Because the model used by Tillmann, Bharucha, and Bigand incorporates a high degree of western music theory, it is the one against which I will most closely compare the topologies evolved by the GENDALC system.

4 The GENDALC system

The GENDALC (Genetic Evolution of Neuron Distribution and Layer Connectivity) system has no built-in knowledge of music theory. Instead, it generates NN with topologies completely determined by values of evolved genomes. The GENDALC system determines NN topologies by distributing 75 hidden nodes among 30 hidden layers, and then determining how these hidden layers are connected to each other (Dávila, 1999). Each topology in the GENDALC system is configured by a genome in a genetic algorithm population. The genome has 30 genes used to code the "relative worth" of each of the hidden layers. The 75 available hidden nodes are distributed among these hidden layers according to each layer's worth relative to the sum of all worth values.

Figure 3 outlines this process, although (solely for the sake of simplicity) with a smaller number of nodes and layers. In this case, the genome is dividing 16 nodes among 6 layers. If the six relative worth genes had the values shown in the top box of figure 3, the total sum of worths would be 2.6. Dividing the 16 available nodes with the above algorithm would assign 1 node to layer 1, 6 nodes to layer two, four nodes to layer 3, and so forth, as illustrated in the lower box of figure 3.

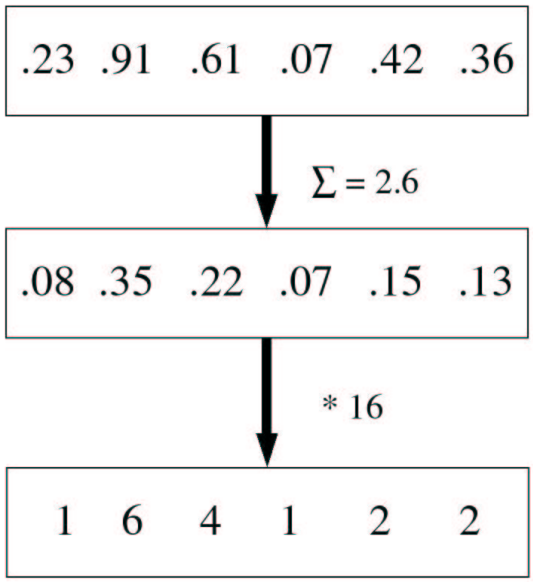

Figure 3: division of nodes into layers

To determine where each hidden layer takes its input from, we use another set of 30 genes as "takes-its-input from" genes. The floating point numbers stored in these genes are multiplied by 32 and rounded to the nearest integer. The resulting number points to which layer this one takes its input from. We multiply by 32 to allow for hidden layers taking their input from any of the 30 hidden layers, as well as either the input or the output layer. A value of 0 means the layer takes its input from the input layer. A value of 32 means the layer takes its input from the output layer. For values between 2 and 31, the layer would take its input from the layer with the (N+1)th highest relative worth.

Where each layer sends its output to is determined in a similar way, using another set of 30 genes. Each of these genes stores a floating point value between 0 and 1. To determine where each layer sends its output to its "sends-output-to" gene value is multiplied by 31 and rounded to the nearest integer. The resulting number points to which

other layer this one will send its output to. We multiply by 31 to allow for hidden layers sending their output to any of the 30 hidden layers, as well as to the output layer. A value of 31 means the layer sends its output to the output layer. For values between 1 and 30, the layer sends its output to the layer with the Nth highest relative worth. No layer sends its output back to the input layer.

5 The Task: Determining Key in Musical Compositions

Neural networks in this experiment received pieces of musical compositions in intervals of 1/8 of a note. Each piece was represented by a vector of 12 values, where one of these 12 values was set to 1 and the rest to 0. Which of the 12 values was set to 1 was determined based on the commonly used 12 tone representation of pitch. NN were then supposed to indicate if the composition being presented was written in major or minor key. NN outputs below 0.5 were interpreted as an indication of minor key, and outputs of 0.5 or above were interpreted as an indication of major key. Networks were trained with six compositions: *Eine Kleine Nachtmusik, Amazing Grace, Auld Lang Syne, Scarborough Fair, House of the Rising Sun,* and *The Battle of Jericho*. They were then tested on a set of 19 songs, which included the six previously listed, and then added *Alouette, Battle Hymn of the Republic, Carmen, Daisy Daisy, Danny Boy, El Condor Pasa, Hava Nagila, Little Brown Jug, Swan Lake, The New World Symphony, The Wedding March,* and *When Johnny Comes Home.* In terms of numbers of patterns, the six songs used for training constituted only 20% of the patterns used for testing. Because the fitness of any one NN was the number of notes incorrectly classified across all patterns in the testing set, networks that simply memorized the patterns present in the training set would tend to have poor fitness values. That is, in order to perform well with the testing set, networks would need to detect general musical patterns in the training set, which it could then properly apply to a broader set of compositions.

6 Experimental Parameters

Genetic populations were initialized with 192 random elements. The population size was kept constant, and elements were paired with equal probability across the population. Once paired, elements were combined by using two random crossover points. Both offspring were evaluated, and their fitness compared with that of all the population. At any point, the best 192 individuals seen during a run were part of the population. A mutation rate of .009 was used on all runs. Runs were repeated 48 times, each time using a different random seed. Values reported here are the averages of these 48 runs.

During their training phase, each NN was presented with the training set for 500 epochs. Weights were adjusted using the backpropagation through time algorithm, which is a generalization of the backpropagation algorithm that allows for feedback loops. For the purposes of evolution, the fitness of a particular NN was determined by counting how many of the input patterns (a total of 2345 12-tone vectors) were incorrectly classified (i.e. a low fitness value was considered better than a high one). All nodes in the input and output layers used identity transfer functions. All hidden nodes used quickly saturating logistic functions similar to the one shown in figure 2-c-. After evolution, and in order to verify consistent performance, NN were cross-validated 48 times (Weiss and Kulikowski, 1991). Values reported in this paper are the average of these 48 runs.

7 Results

7.1 Leaky Units

Experiments were repeated under several input configurations, in order to test the potential advantage of leaky integrator nodes (LIN) (DeVries and Principe, 1992; Elman, 1990; Poddar and Unnikrishnan, 1991). LIN partially preserve the activation they had in previous time steps. When used at the input layer, they provide information of input vectors seen in the recent past. This could potentially benefit NN performance by providing some context for the notes currently being presented. In the experiments reported in this paper, the output of LIN is given by

$$O(t) = \begin{cases} A(t) & \text{If } A(t) > 0 \\ O(t\text{-}1) - 1/L & \text{If } A(t) = 0 \text{ and } O(t\text{-}1) > 0 \\ 0 & \text{Otherwise} \end{cases}$$

where O(t) is the output of the node at time t, A(t) is the activation of the node as per the note played at time t, and L is the number of time steps that it takes for a past activation to decay to zero. Higher values of L cause the node activation to decay more slowly, allowing the input vectors to store more historical information.

Table 1 presents the performance of the best network in the genetic algorithm population after 140 generations. As can be seen, not only are leaky integrator units unnecessary for the task and the system outlined above, they in fact cause performance degradation. Analysis of network behavior while input patterns are presented reveals the source for this surprising result. Hidden nodes fail to develop the precision required to distinguish differing but similar values at the input layer. Because of this, they perform better when values at the input nodes are either 1 or 0, but do less well when input nodes can have values as close to each other as 0.5 and 0.667, as is the case when the time constant of input nodes is 6.

Table 1: NN performance while using leaky integrating neurons with different time constants.

L	Incorrectly classified patterns
1	362 (15.44%)
2	414 (17.65%)
4	485 (20.68%)
6	538 (22.94%)

7.2 Evolved Topologies

Analysis of NN topologies evolved with L = 1 system reveals that networks are generally divided into two collaborating sections. One of these sections provides paths of different lengths from the input layer to the output layer. The other section connects a series of layers in a complex mesh of short paths. Figure 4 shows the general outline of these networks.

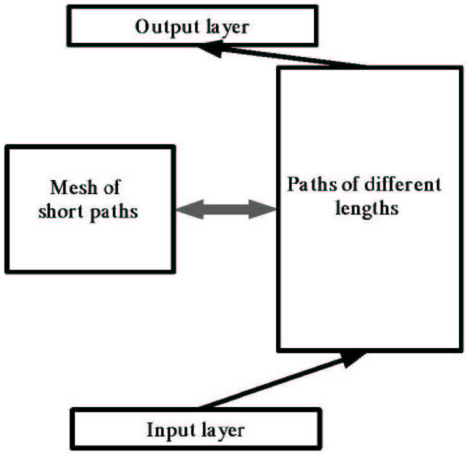

Figure 4: general outline for evolved topologies

Figure 5 shows details of the paths of different lengths going from the input layer to the output layer. These paths tend to be from 5 to 8 hidden layers long.

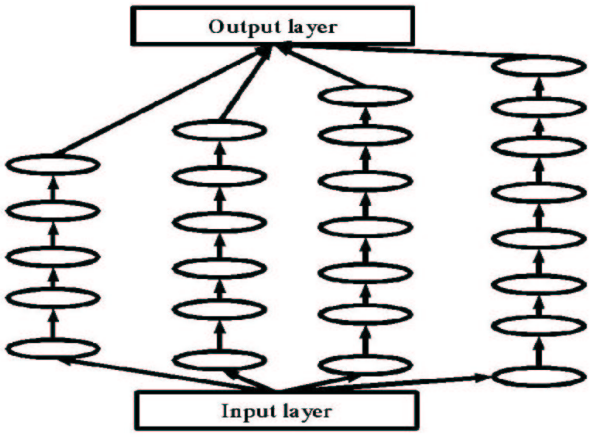

Figure 5: paths of different length from input to output

Figure 6 shows typical details of the section that forms a mesh of short paths. Although these meshes tend to be quite complex, they can be best described as follows: layers fall into one of three levels, where the first level receives inputs from the paths of different lengths, and the third level sends outputs back to the paths of different lengths. Some layers in the first level send their output to layers in the second level; some layers in the first level send their output to layers in the third level; and some layers in the first level send their output to layers in both of the following two levels. Layers in the third level are sequentially connected among themselves.

Connections going from the path of different lengths to the mesh of short paths originated at hidden layers one or two levels after the input layer. Connections going back from the mesh to the path of different lengths ended up at hidden layers one or to levels before the output layer. This is outlined in figure 6.

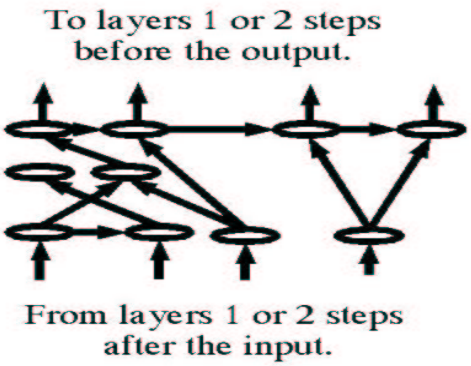

Figure 6: connection between paths of different lengths and mesh of short paths

Surprisingly, the topologies evolved with higher values of L had almost identical structure as those discussed above. In particular, it is interesting to note that, although the networks were similar at the genotype level, they were being defined by rather different genomes. This tends to indicate that these topological characteristics are important enough as to guide/force the evolutionary process into finding different ways of arriving at them.

7.3 Comparisons with other topologies

In order to have a better sense of the success of the GENDALC system in the task discussed in this paper, the same set of experiments were repeated, this time presenting the same musical compositions to NN with other commonly used topologies. Since the task presented here is one that requires temporal processing (that is, the output at any one time depends not only on the current input, but also on previous ones), tests were performed with four different types of recurrent networks: fully connected networks, Elman networks (Elman 1991), Frasconi-Gori-Soda networks (FGS) (Frasconi, Gori, Soda 1992), and Narendra-Parthasarathy networks (N-P) (Narendra, Parthasarathy 1990). Fully connected networks have a single hidden layer (in this case with 75 nodes), where every node of the input layer is connected to every node of the hidden layer, and every node of the hidden layer is connected to every node of both the hidden and output layers. Frasconi-Gori-Soda networks, Elman networks, Narendra-Parthasarathy networks, and Jordan networks are illustrated in figure 7a, 7b, 7c, and 7d, respectively. For purposes of simplicity, these

networks are illustrated here with 4 hidden nodes, as opposed to with the 75 hidden nodes that were actually used.

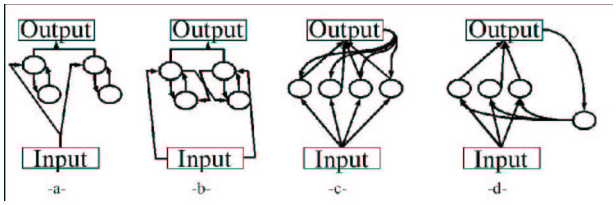

Figure 7: FGS , Elman, N-P, and Jordan NN.

As can be seen in table 2, the topology evolved by the GENDALC system outperform all of these other recurrent topologies.

Table 2: NN performance for several recurrent topologies.

Topology	Incorrectly classified patterns
Evolved by GENDALC	362 (15.44%)
FGS	720 (30.70%)
N-P	589 (25.12%)
Elman	717 (30.58%)
Fully Connected	624 (26.61%)
Jordan	578 (24.65%)

An additional set of comparison runs was performed, this time with topologies that included the 48-node toroid used by Tillmann, Bharucha, and Bigand (which I have briefly discussed in section 2 of this paper). In these topologies, the twelve node input layer was connected to the nodes of the toroid representing notes. Each of the nodes of the toroid was then connected to another set of hidden nodes forming one of the recurrent topologies presented in this section. Therefore, each of the pre-designed topologies was now taking inputs from the music-theory-inspired toroid, as opposed to directly from the input layer. Tests with this type of topology were performed with both 27 and 75 hidden nodes in the 'non-toroid' part of the networks, as well as with one network where the output node was directly connected to each of the nodes in the toroid (in this paper that topology will be referred to as 'pure toroid'). An outline of one such network (in this case for a Jordan topology, with 27 additional nodes) is shown in figure 8. The performance of these networks is shown in table 3. As was the case with previous tests, the topologies evolved by the GENDALC system outperform these pre-designed recurrent topologies.

7.4 Analysis

While the musical task approached in these experiments clearly requires time-dependent processing, exactly which topology would be optimal for it is hard to predict a priory. In fact, the GENDALC system's main advantage is that it searches for good topologies outside any particular pre-designed topology. Some clues can be extracted from the performance of networks with specific topologies. Form the performance of the NN consisting

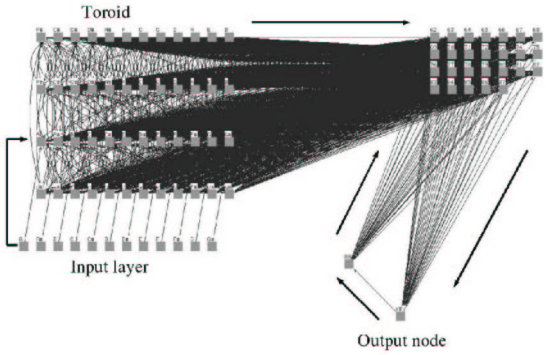

Figure 8: sketch of Jordan network with toroid.

Table 3: NN performance for several recurrent topologies with music-theory inspired toroid.

Topology	Incorrectly classified patterns
Evolved by GENDALC	362 (15.44%)
FGS-27 nodes	589 (25.09%)
FGS-75 nodes	443 (18.89%)
N-P - 25 nodes	749 (31.94%)
N-P - 75 nodes	459 (19.53%)
Elman-27 nodes	770 (32.84%)
Elman 75 nodes	867 (36.97%)
Fully connected - 27 nodes	586 (24.99%)
Fully connected - 75 nodes	586 (24.99%)
Jordan-27 nodes	884 (37.70%)
Jordan-75 nodes	587 (25.03%)
Pure toroid	865 (36.89%)

only of a toroid-like hidden layer, it is clear that, while this topology manages to embed music knowledge, it seems to lack the ability to store and process longer-term input relations. Topologies lacking toroid-based knowledge can use their recurrent connections to provide time-dependent information, but for the most part seem to benefit (or at least not suffer much) from having a manually-designed segment of their hidden layer incorporating music knowledge. Of course, for any one of these topologies, how to combine music knowledge with time-responsive recurrent connections is an open question. It is GENDALC's evolutionary process that allows it to search for combinations of these to seemingly important aspects. Although further study of the behavior of hidden nodes in the evolved topologies is needed, the paths of different lengths from input to output layers seems to provide enough time displaced information to the output layer, while the mesh of short paths could be mimicking enough toroid-based music information to assist with the task. What is certain is that topologies evolved by the GENDALC system outperform a variety of hand-designed topologies that contain characteristics favorable to the task. This shows promise not only for music-oriented tasks, but for a variety of hierarchical time-dependent tasks. This is particularly true in light of results obtained by the GENDALC system in natural language processing tasks (Dávila, 1999).

8 Future Research

While the results obtained in these experiments clearly demonstrate that the GENDALC system can successfully find configurations for the intended task, how the evolved networks are solving their task is still an open question. Potential answers to this question might be obtained once we analyze both the final values for connections between nodes, as well as internal node activations during processing.

Additionally, I am currently working on the evolution of neural networks for music generation, as opposed to music classification. The goal of these experiments is to evolve topologies that allow a NN to be trained with Jazz performances from specific musicians, and then produce performances for different compositions similar to the ones the musicians themselves played. Because of the inherent hierarchical structure of this task, the system should naturally progress towards topologies capable of capturing complex musical knowledge.

9 Conclusions

This paper has presented a brief introduction to the GENDALC system for evolving neural network topologies. Experiments carried out demonstrate the system's ability to evolve topologies that can successfully classify musical compositions based on key. This, combined with results in natural language processing tasks reported elsewhere, is relevant to the broader effort to develop cognitive systems, since the task requires hierarchical time dependent processing.

Acknowledgments

Thanks to Mara Breen, previously an undergraduate student at Hampshire College and now a graduate student at M.I.T., for transcribing the compositions used in this research into numerical format. Thanks also to Professors Joanna Morris and Neil Stillings, at Hampshire College, for general consultations on music and perception theory.

Bibliography

Carpenter, G.A., Grossberg, S. (1987a) A Massively Parallel Architecture for a Self-organizing Neural Pattern Recognition Machine. *Computer Vision, Graphics and Image Processing*, 37. Pp. 54-115.

Carpenter, G.A., Grossberg, S. (1987b) Stable self-organization of pattern recognition codes for analog input patterns. *Applied Optics*, 26. Pp. 4919-4930.

Dávila, J. (1999) Genetic Optimization of NN Topologies for the Task of Natural Language Processing. *Proceedings of the International Joint Conference on Neural Networks, Washington, D.C.*

DeVries, B., & Principe, J. (1992). Short term memory structures for dynamic neural networks. *Proceedings of the 26th ASILOMAR Conference.* Pp. 766-770.

Elman, J. L., (1991) Distributed Representations, Simple Recurrent Networks, and Grammatical Structure. *Machine Learning.* Pp. 71-99

Frasconi, P., Gori, M., Soda, G., (1992) Local feedback multilayered networks. *Neural Computation, 4(1).* Pp. 120-130.

Griffith, N. (1994) Development of Tonal Centers and Abstract Pitch as Categorizations of Pitch Use. *Connection Science, vol. 6.* Pp. 155-175.

Janata, P., Birk, J., Van Horn, J., Leman, M., Tillmann, B., Bharucha, J. (2002) The Cortical Topography of Tonal Structures Underlying Western Music. *Science, Vol. 298, No. 5601.* Pp. 2167-2170.

Jordan, M.I. (1986) Attractor dynamics and parallelism in a connectionist sequential machine. *Proceedings of the Eighth Annual Conference of the Cognitive Science Society. Pp.531-546.*

Naredra, K., Parthasarathy, K. (1990) Identification and control of dynamical systems using neural networks. *IEEE Transactions on Neural Networks, 1(1).* Pp. 4-27

Poddar, P., Unnikrishnan, K. (1991). Non-linear Prediction of Speech Signals Using Memory Neuron Networks. *Proceedings of the IEEE Workshop on Neural Networks for Signal Processing.* Pp.395-404.

Taylor, I., Greenhough, M. (1994) Modeling Pitch Perception with Adaptive Resonance Theory Artificial Neural networks. *Connection Science, Vol. 6.* Pp. 135-154.

Tillmann, B., Bharucha, J. J., Bigand, E. (2001) Implicit Learning of Regularities in Western Tonal Music by Self-Organization . Dans: R. French & J. Sougné (Eds.) *Perspectives in Neural Computing series. Proceedings of the Sixth Neural Computation and Psychology Workshop: Evolution, Learning, and Development.* Springer. Pp. 175-184.

Weiss, S., Kulikowski, C. (1991) Computer Systems that Learn. Classification and Prediction Methods from Statistics. *Neural Nets, Machine Learning, and Expert Systems.* Morgan Kaufmann.