A Neural Network Based Interface to Real Time Control Musical Synthesis Processes

GIOVANNI COSTANTINI[1,2], MASSIMILIANO TODISCO[1], MASSIMO CAROTA[1]
[1]Department of Electronic Engineering
University of Rome "Tor Vergata"
Via del Politecnico, 1 – 00133 ROMA
ITALY
[2]Institute of acoustics "O. M. Corbino"
Via del Fosso del Cavaliere, 100 – 00133 ROMA
ITALY

Abstract: - In this paper, we present an innovative Neural Network system interface that allows an electronic music composer to plan and conduct the musical expressivity of a performer. For musical expressivity we mean all those execution techniques and modalities that a performer has to follow in order to satisfy common musical aesthetics, as well as the desiderata of the composer. The proposed interface or virtual musical instrument is able to transform two input parameters in many sound synthesis parameters. Especially, we focus our attention on mapping strategies based on Neural Network to solve the problem of electronic music expressivity.

Key-Words: - Neural Network, Control, Musical Synthesis Process.

1 Introduction

Traditional musical sound is a direct result of the interaction between a performer and a musical instrument, based on complex phenomena, such as creativeness, feeling, skill, muscular and nervous system actions, movement of the limbs, all of them being the foundation of musical expressivity.

Actually, musical instruments transduce movements of a performer into sound. Moreover, they require two or more control inputs to generate a single sound. For example, the loudness of the sound can be controlled by means of a bow, a mouthpiece, or by plucking a string. The pitch is controlled separately, for example by means of fingering which changes the length of an air column or of a string.

The sound produced is characteristic of the musical instrument itself and depends on a multitude of time-varying physics quantities, such as frequencies, amplitudes, and phases of its sinusoidal partials [1].

The way music is composed and performed changes dramatically [2] when, to control the synthesis parameters of a sound generator, we use human-computer interfaces, such as mouse, keyboard, touch screen or input devices such as kinematic and electromagnetic sensors, or gestural control interfaces [3,4]. As regards musical expressivity, it is important to define how to map few input data onto a lot of synthesis parameters. At present, it is obvious that the simple one-to-one mapping laws regarding traditional acoustical instruments leave room to a wide range of mapping strategies.

The paper is organized as follows: in the second session we describe perceptual considerations; in the third session we describe the Neural Network structure; in the fourth session we describe and illustrate our interface and the mapping strategies adopted; finally, we show a real-time musical application using our interface.

2 Perceptual considerations

To investigate the influence that mapping has on musical expression, let us consider some aspects of Information Theory and Perception Theory [5]:

- the quality of a message, in terms of the information it conveys, increases with its originality, that is with its unpredictability;
- information is not the same as the meaning it conveys: a maximum information message doesn't make sense, if any listener that's able to decode it doesn't exist.

A perceptual paradox [6] illustrating how an analytic model fails in predicting what we perceive from what our senses transduce is the following: both maximum predictability and maximum unpredictability imply minimum information or even no information at all.

A neural network approach is chosen to exceed the perceptual limits above mentioned.

3 Neural Network structure and Learning rule

An artificial neural network [7] is a mathematical model for information processing based on a connectionist approach to computation, inspired by the human brain.

In a neural network model, simple nodes (or "neurons", or "units") are connected together to form a network of nodes. The strength of a connection between a neuron and another is influenced by a weight value.

A typical neural network is arranged with three layers of neurons: input, hidden, and output layer. In this context, we consider feed-forward architectures only, where the information signal propagates from input layer, through intermediate or hidden layer, to output layer, with no loops back to previous neurons.

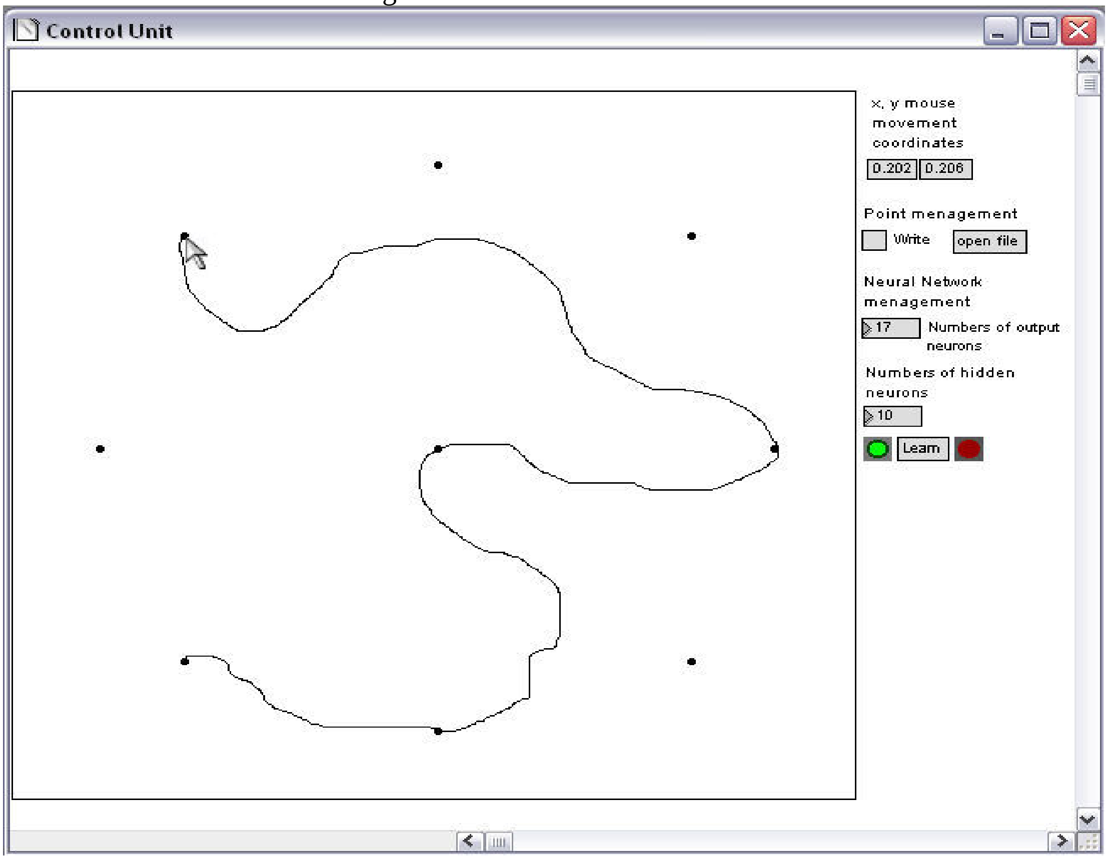

Fig. 1: The control unit

This neural network is known as Multilayer Preceptron (MLP) or FeedForward Backpropagation Neural Network (FFBPNN), due to its learning algorithm. FFBPNN's input layer and output layer represent the points of contact of the net with the external environment, while the hidden layer contributes to form the non linear relations existing between inputs and outputs.

A key property of a neural network is its ability to acquire knowledge by examples. Learning is an iterative process of adjustment applied to the synaptic weights of the network in response to an external stimulus. In particular, we will consider only neural networks trained by means of supervised learning: a training set, which contains both the input patterns and the corresponding desired outputs (or target patterns), is presented iteratively to the network with the aim of implementing a mapping that matches the training examples as closely as possible.

Weights are iteratively modified through two passages, which represent an epoch (backpropagation algorithm):

1. A pattern input is proposed to the network input and then it is propagated to the network output (forward pass); than the error E as the squared difference between desired and actual output is calculated;

2. The error E is back-propagated (backward pass) and weights are updated according to the formula of gradient descent [7].

After all, a FFBPNN works like powerful mathematical trainable interpolation systems to calculate non-linear functions starting from desired inputs/output relationships. The complexity of interpolating function grows with the number of the neurons in the hidden layer, as well as learning capability. These kinds of structures are simple and effective, and have been exploited in a wide assortment of machine learning applications [7].

4 Interface and Mapping strategies

The new virtual musical instrument that we propose has been developed by using the Max/MSP [8] environment. It is constituted by three components: the control unit, the mapping unit and the synthesizer unit. The control unit allow the performer to control two parameters. Fig. 1 show the Max/MSP patch that constitutes the control unit. Particularly, performer draws lines and curves in a bi-dimensional box, by clicking and moving mouse inside the patch itself.

The control unit patch is constituted by the *lcd* Max/MSP object that returns *x*, *y* mouse space coordinates.

The evident points in the Fig. 1 represent the input/output patterns of the training set.

In detail, the two *x, y* control parameters don't influence directly the parameters that rule the behavior of the sound generators, but they are pre-processed by the mapping algorithm.

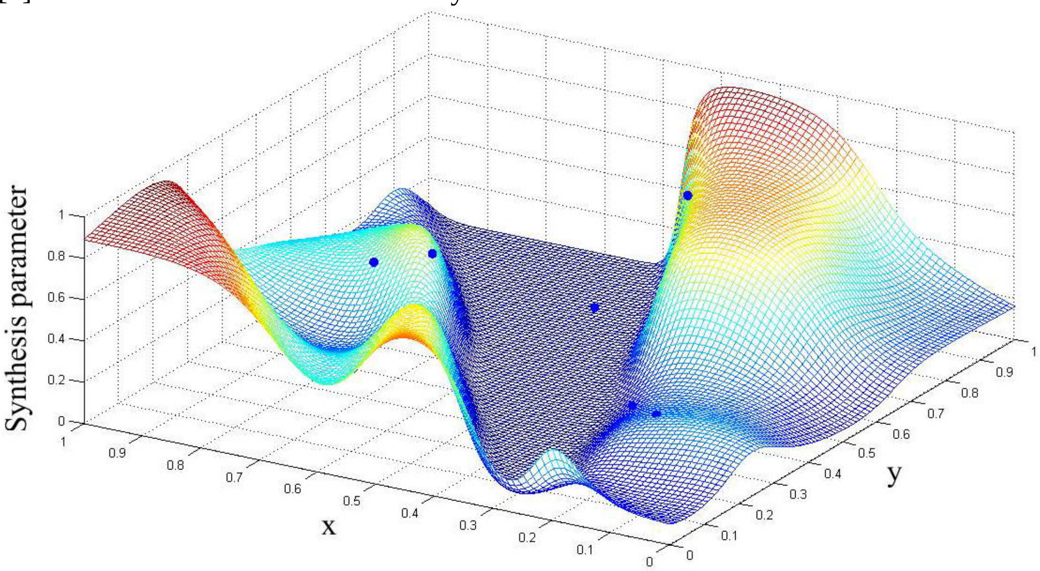

Figure 4: A synthesis parameter as a function of the input *x* and *y*.

The implementation of musical expressivity is accomplished once we define the correspondence between the two *x, y* control parameters and the *m* synthesis parameters, that is to say, once we define the right mapping.

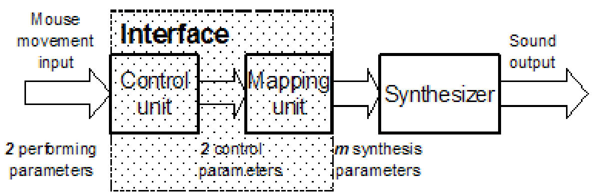

Fig. 2: The virtual musical instrument.

The chosen mapping strategies, by means of which the synthesis parameters are controlled, all influence the way the musician approaches the composition process.

In Fig. 2, the structure of the virtual musical instrument is shown.

Let's assume the following concepts:

1. a predictable musical message be associated to an a priori known functional relation between the surface $\mathbb{R}^2$ and the hyperspaces $\mathbb{R}^m$, that is to say, between the set of all the 2 inputs and the set of all the *m* synthesis parameters;
2. an unpredictable musical message be associated to a non linear and a priori unknown correspondence between $\mathbb{R}^2$ and $\mathbb{R}^m$.

A composer can easily follow the above assumptions by making use of a FFBPNN trained as follows:

1. he fixes a point in the 2-dimensional *x*, *y* space and he links it to a desired configuration of the *m* synthesis parameters;
2. he repeats D times step 1., so as to have D 2-to-*m* examples at his disposal; they constitute the training set for the mapping unit;
3. he chooses the neural network structure, that is to say the number of hidden neurons to use; then, he trains the neural network.
4. he explores the 2-dimensional *x, y* input space by moving through known and unknown points, with the aim of composing his piece of music.

5 Real-time musical application

We tested our interface, developed under the Max/MSP environment, by writing a real-time musical composition. The synthesis process was realized by means of the sound synthesizer "Textures 2.0" [9].

Fig. 3 show "Texture 2.0" standard VST [10] (Virtual Studio Technology from Steinberg ltd) audio plug-in.

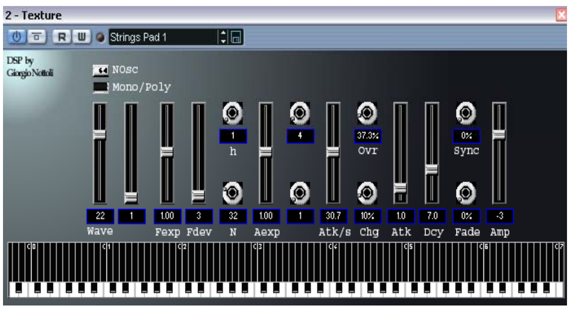

Fig. 3: "Texture 2.0" sound synthesizer.

The sound synthesized with "Texture 2.0" is based on a granular additive synthesis algorithm. There are seventeen sound synthesis parameters [9], showed in Fig. 3, regarding sliders and knobs, through which we can shape the sound waveform.

We have two *x, y* control inputs to operate on the seventeen parameters that influence the synthesis produced by the synthesizer.

We have chosen nine reference points in the 2-dimensional space of the input control. Then, we have trained a FFBPNN with ten neurons in the hidden layer and afterwards explored the 2-dimensional space. Finally, we have chosen, amongst many others, the mouse movements that have to be repeated, in order to reproduce the interesting sounds discovered during exploration.

In Fig. 4 the slope of a synthesis parameter returned by the output of the neural network, as a function of the input *x*, *y* coordinates, is shown. The evident points in the graph represent the input/output patterns of our training set.

4 Conclusion

We have developed a virtual musical instrument for composing and performing expressive musical sound. We direct attention to common musical aesthetics as a determinant factor in musical expressivity.

The interface we have presented is formed by a control unit, that supplies *x* and *y* coordinates of points in a bi-dimensional box, and by a mapping unit, based on a FFBPNN, that processes those points, in order to provide suitable relationships between input mouse movements and sound synthesis parameters.

The experiences made by working with our interface have shown that the mapping strategy is a key element in providing musical sounds with expressivity.

So, we have defined four composition rules by means of which a musician can easily compose his own piece of music with our interface.

At last, a musical composition based on our interface was implemented, in which mouse movements were turned into expressive musical sound.

References:

[1] Neville H. Fletcher, Thomas D. Rossing, *The Physics of Musical Instruments*, Springer, 2nd edition (July 22, 2005).

[2] Curtis Roads, *The computer music tutorial*, The MIT Press, (February 27, 1996).

[3] Bongers, B. 2000, *Physical Interfaces in the Electronic Arts. Interaction Theory and Interfacing Techniques for Real-time Performance*, In M. Wanderley and M. Battier, eds. Trends in Gestural Control of Music. Ircam - Centre Pompidou.

[4] Orio, N. 1999, A Model for Human-Computer Interaction Based on the Recognition of Musical Gestures, *Proceedings of the 1999 IEEE International Conference on Systems, Man and Cybernetics*, pp. 333-338.

[5] Abraham Moles, *Information Theory and Aesthetic Perception*, University Of Illinois Press (1969).

[6] Rudolf Arnheim, *Entropy and Art: An Essay on Disorder and Order*, University of California Press (January 29, 1974).

[7] Hertz J., A. Krogh & R.G. Palmer, *Introduction to the theory of neural*

computation, Addison-Wesley Publishing Company, Reading Massachusetts, 1991.

[8] Cycling74 Max/MSP, documentation avaible on the web at http://www.cycling74.com/products/maxmsp

[9] Giorgio Nottoli, "A sound texture synthesizer based on algorithmic generation of micro-polyphonies", Proc. of 3nd International Conference "Understanding and creating Music", Caserta, December 2003, 11-15.

[10]Steinberg VST Audio Plug-Ins SDK, 3rdparty developer support site at http://www.steinberg.net/324_1.html