NEURAL NETWORKS FOR MUSICAL TONES COMPRESSION, CONTROL, AND SYNTHESIS.

Roberto Bresin, Alessandro Vedovetto
C.S.C. - Centro di Sistemi Computerizzati
Università di Padova - via S. Francesco 11 - 35121 Padova - Italy
phone +39.49.8283757 - fax +39.49.8283733
email: rbr@css.unipd.it, rbl@dei.unipd.it

Abstract

This work consists of a hybrid system: a neural network for data compression and a generic algorithm for sound synthesis. The goal is the reduction of the number of parameters associated with a sound. Because of the growth of the electronic technology, the actual synthesizers are now very complex systems in which the user can't move himself without any difficulty, and expensive loss of time. The basic idea is the reduction of the parameters made by the neural network (NN), and the freedom to choose any algorithm for the sound synthesis.

1 Introduction

In commercial synthesizers is often very difficult, if not impossible, to create new sounds starting from the existing set of the machine. Another problem is the great number of the parameters to control in a synthesis algorithm. What one could desire from a commercial electronic instrument could also be the possibility to change gradually from one timbre to another one. For these and other reasons we decide to build a neural network based system to control a particular synthesis algorithm, to reduce the parameters for controlling it, and to allow a sort of morphing of timbres.

Also computer music composers could benefit from this system since they have to deal with many parameters to synthesise, control, and process musical tones. They would be helped in their work since it would be easier to manipulate sounds in an intuitive way, and with a reduced number of parameters to control. As we will see in the following, the major work is for the developer, but this is made only once.

The user will just move some controllers (i.e. some sliders); the neural network computes its output which is the set of parameters for the synthesizer. For example, with a good choice of the parameters it is possible to generate all the tones moving from a sax to a trumpet, or from a piano to an organ.

2 Musical Tones Processing with Neural Networks

The neural set (NN) used in our system is called "data compression neural network", and it is defined by the two following points:

- input and output have the same number of perceptrons
- the middle layer has less perceptrons than the input one

If a similar set is trained with a back-propagation training algorithm, choosing all the output patterns (targets) to be equal to the input patterns, the first result is that in the hidden level it is available a compressed version of the inputs, the lower half of the net makes the compression of the data, while the upper half performs the inverse operation. The system treated in this work uses only the upper half of the net for synthesis, while the lower half of the net it can be used for the analysis of the compression map, or for the classification of timbres.

2.1 Compression

The NN showed in Figure 1 has a sandglass shape: i.e. the same number (N) of input and output neurons, and a littler number (M < N) of neurons at the middle hidden layer. The NN is a completely connected one: each of the N nodes in input are connected with every node of the hidden layer (M nodes); the output values of these M nodes generate the N output values of the NN. We trained this NN using the back-propagation algorithm [McClelland et al., 1988], and saying that the desired input/output relation should be the identity function, i.e. the set of output values must be equal to the set of input values. In this way the output values at the hidden layer of the NN are equivalent at the corresponding input values since, with the upper part of the NN, it is possible to reconstruct their values. The main result is that the values at the hidden layer are a code of the input data, and represent the input

Sound Synthesis Techniques 368
ICMC Proceedings 1994
data also with a reduced number of inputs. For a NN with \( N \) input (output) neurons, according to McCulloch and Pitts [McCulloch et al., 1943], it is possible to code the \( N \) values of each input pattern into \( \log_2 N \) values, that is also the number of neurons in the middle layer of the NN. We succeeded in mapping \( N \) long input-patterns into \( M \) long coded-patterns, with \( M < \log_2 N \). So NN with a sandglass structure is possible to compress a vector of \( N \) values into a vector of \( M \) values, \( M \geq N \); if we use \( N \) input values some parameters of a sound synthesis algorithm, such as frequencies, amplitudes, etc., we can compress them into a smaller number of parameters.

![Figure 1: Neural Network for Data Compression](image)

2.2 Control

Once each input pattern is codified into a smaller vector, it is possible to rebuild it using the upper part of the NN in Figure 1. If we think to associate the middle-layer neurons to a gestural controller, like the sliders of a midi mixer, it is possible, by simply moving the controller, to pass from a certain configuration of the middle-layer neurons to another one, corresponding to the code of another particular input/output pattern. If we start from a configuration of the middle-layer neurons corresponding to a previously learned input/output pattern, and we move the controller towards a new position corresponding to another previously learned input/output pattern, all the intermediate positions of the controller will correspond to previously learned input/output patterns which belong to the same family of the training patterns, but are new in respect to them. In example, if the NN of Figure 1 was trained with data corresponding to the timbre of the trombone, and of the clarinet, when we move the controller from the position corresponding to the trombone to the clarinet one, we will have the timbres of all the virtual instruments between them: something like the "trumbient", the "claripet", etc.

2.3 Synthesis

If we apply the NN of Figure 1 to the control of a synthesis algorithm like that for the additive synthesis, we could use it to synthesize new sounds starting from fa-sounds training database used to train the NN. If the training database is made of sounds belonging to different families (woodwind, strings, etc.) the action on the controller allows to create hybrid instruments sounds; if the training database is made of sounds belonging to the same family (i.e. various flutes) our system allows to obtain different shades of the same instrument. In the following we propose an example of application of our system.

Example

We will show and discuss some signals obtained with a NN with 3 input nodes, 24 output nodes, and 10 hidden nodes. This NN is the upper part of a sandglass-shaped NN. The output vector is considered as the spectral content of the signal to synthesize: each output neuron gives the amplitude of the partial with frequency \( f_k \) with \( k = 1, \ldots, 24 \) and \( f_0 \) fundamental frequency. In this way the signal generated by the NN has the following representation:

\[
y(nT) = \sum_{k} A_k \sin(2\pi (k+1) - f_0 nT)
\]

where \( A_k \) is the value of the \( k \)-th output neuron. The training of this NN was made using a set of data extracted from the Gray's tables [Gray, 1975]. The instruments of the training set were: trombone, english horn, clarinet, oboe, and cello. We considered the amplitude value of each partial at time \( t = 100 \) mc; the frequency of each partial was considered a multiple of the fundamental. In the following figures are showed the time evolutions of some signals obtain with the above described NN; the corresponding values of the 3 input neurons are written in the note to each figure (the range of this
values is 0.1). It is interesting to compare the signals of Figure 2 to 4, with the signals of the training instruments reconstructed by the NN, Figure 5 to 9. You can observe that the signal in Figure 4 has a wave shape very close to that of the clarinet (Figure 7), and the wave in Figure 3 is very similar to the one of the trombone in Figure 5. This similarities are clear if we look at Figure 10, that represent the distribution in a 3D space of the compressed values corresponding to the training set instruments. In Figure 10 the 24 harmonics of the trombone are associated very closely to the point (1,1,1), so it clear why Figures 3 and 5 are very similar. The signal in Figure 4 has the compressed representation (0.0,0.5,1), and the compressed representation of the clarinet is (0,0,1): so the two signals are near in the 3D space of Figure 10, and this is the reason for their similarity.

3 Results

An important result of the present work is that NNs like that of Figure 1 allow to obtain signals with a shape that changes continuously from one wave to another one. The training set is the database of signals where to find the information to build a new timbre related to a particular configuration of the middle-layer neurones.

The good results obtained encouraged us to build a cascade of two NNs: the first NN as the same kind of structure of that in Figure 1, the second NN implements the evolution in time of the signal in the first 300 ms. An example of signal obtained with this method is shown in Figure 11. Furthermore, the idea of using sandglass shaped NNs to control the additive synthesis algorithms, could be extended to other techniques.

Sound Synthesis Techniques

ICMC Proceedings 1994
4 Conclusions

We succeeded in reducing the parameters to control, and to maintain a good signal quality. If we train the same NN with many different training sets of signals, we will have an instrument with infinite Klangfarbe possibilities, and we could access them by simply move some sliders, or any other controller in an intuitive way.

The system described in this paper was first developed on a NeXT Cube, the program is called NeSSy (Neural network Sound Synthesis): it allows to use both with the mouse and with MIDI controllers. For all the tones generated by NeSSy it is possible to change the envelope aspect, to select a stick point, to set the total duration of the tone, and to switch between smoothed or not smoothed envelope shapes. The number of points of the envelope ranges from a minimum of two points, to an unlimited maximum number.

A version of the program for PC is also available.

References