ESANN'1995 proceedings - European Symposium on Artificial Neural Networks
Brussels (Belgium), 19-20-21 April 1995, D-Facto public., ISBN 2-9600049-3-0, pp. 45-50

Active Noise Control with Dynamic Recurrent Neural Networks

Davor PAVISIC Laurent BLONDEL Jean-Philipe DRAYE
Gaëtan LIBERT Pierre CHAPELLE

Parallel Information Processing Laboratory
Faculté Polytechnique de Mons – B-7000 Mons (BELGIUM)

Abstract. We have developed a neural active noise controller which performs better than existing techniques. We used a dynamic recurrent neural network to model the behaviour of an existing controller that uses a Least Mean Squares algorithm to minimize an error signal. The network has two types of adaptive parameters, the weights between the units and the time constants associated with each neuron. Measured results show a significant improvement of the neural controller when compared with the existing system.

1 Introduction.

Active Noise Control (**ANC**) uses the intentional superposition of acoustic waves to create a destructive interference pattern such that a reduction of the unwanted noise occurs (Young's principle). Acoustic waves propagating in a rigid walled waveguide are one of the best candidates for **ANC**, first, because a considerable part of environmental noise is transmitted via ducts (ie. noise from ventilation systems or exhaust pipes), and second, because the sound wave in a duct can be regarded as a one-dimensional propagating wave[1].

Here the problem is reduced to computing the optimal transfer function of the controller (see figure 1), which will reduce the sound power level with the secondary source $y(t)$ at the error signal $e(t)$. The complexity of the optimal transfer function makes the design of the *optimal controller* practically impossible and therefore numerical methods are used to adjust (on a sample by sample basis) the coefficients of a finite impulse response filter. The adaptation of the coefficients of this filter is traditionally performed using a Least Mean Squares algorithm to minimize the output signal of the error sensor $e(t)$ [1]. One of the major problems with this type of algorithm is its sensitivity to feedback from the secondary source and this sensitivity can lead to an unstable filter. As an alternative, we present here a neural network based direct control architecture that achieves the reduction of the sound power level at $e(t)$. The *Dynamic Recurrent Neural Network* we introduce has $x(t)$ and $e(t)$ as inputs and $y(t)$ as output. It presents two types of adaptive parameters: the classical weights between the units and the time constants associated with each artificial neuron.

[1] As long as its frequencies are below the natural frequency of the first cross sectional mode in the duct.

ESANN'1995 proceedings - European Symposium on Artificial Neural Networks
Brussels (Belgium), 19-20-21 April 1995, D-Facto public., ISBN 2-9600049-3-0, pp. 45-50

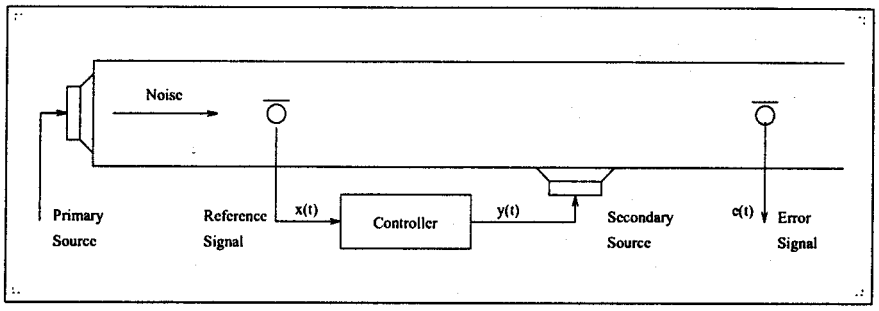

Figure 1: Active Noise Control in a duct. The noise is generated by the source on the left; a microphone picks-up the reference signal $x(t)$; the controller generates the correcting signal $y(t)$; the superposed resulting signal is then picked up by a second microphone $e(t)$ which is used as feedback by the controller.

2 The Network Model, simulation, and training

In this section we will fully describe the neural network model used to replace the traditional LMS controller described above, discuss continuous vs. discrete time models, and derive the backpropagation through time equations which are used for training the network.

2.1 The Dynamic Recurrent Model

We consider neural networks governed by equations (1) and (2) where y_i is the state or activation level of unit i and $F(\alpha)$ is the squashing function $F(\alpha) = (1 + e^{-\alpha})^{-1}$. Equation (1) is the *propagation equation of the network.* The time constant T_i will act like a relaxation process. In order to increase the dynamics of the model, the correction of the time constants will be included in the learning process.

$$T_i \frac{dy_i}{dt} = -y_i + F(x_i) + I_i \quad (1)$$

$$x_i = \sum_j w_{ji} \cdot y_i \quad (2)$$

The network consists of a series of neurons organized in layers. All connections are allowed (feedback, feed-forward, self connection, and even feed-forward and feedback connections between two identical neurons). Some neurons will get the inputs, some will give the outputs and there will be a certain number of hidden units whose inputs and outputs stay within the network. These hidden units allow the network to discover and exploit regularities of the task at hand (such as symmetries or replicated structures).

2.2 Continuous vs. discrete time

We are concerned with continuous-time networks, however, when a continuous-time system is simulated on a digital computer, it is converted into a simple set of first order

ESANN'1995 proceedings - European Symposium on Artificial Neural Networks
Brussels (Belgium), 19-20-21 April 1995, D-Facto public., ISBN 2-9600049-3-0, pp. 45-50

difference equations, which is formally identical to a discrete time network. Therefore, if we use a time step Δt to compute the system, the derivative in (1) can be approximated with equation (3); and substituting (3) in (1) we get equation (4)[2]:

$$\frac{dy_i}{dt} \approx \frac{y_i(t+\Delta t) - y_i(t)}{\Delta t} \tag{3}$$

$$y_i(t + \frac{\Delta t}{T_i}) = (1 - \frac{\Delta t}{T_i}) \cdot y_i(t) + \frac{\Delta t}{T_i} \cdot F(x_i(t)) + \frac{\Delta t}{T_i} \cdot I_i(t) \tag{4}$$

2.3 Backpropagation through time

As classical backpropagation, backpropagation through time will modify all the network weights in order to minimize an error function. Since we want the network to exhibit some particular temporal behaviour, the error function will be a functional defined as in (5), where the moments t_0 and t_1 give the time interval during which the correction process occurs. The function $q(\mathbf{y}(t), t)$ is the cost function at time t which depends on the vector of the neuron activations $\mathbf{y}$ and on time.

$$E = \int_{t_0}^{t_1} q(\mathbf{y}(t), t) \cdot dt \tag{5}$$

$$e_i(t) = \frac{\partial q(\mathbf{y}(t), t)}{\partial y_i(t)} \tag{6}$$

If we define $e_i(t)$ (6) then, intuitively, $e_i(t)$ measures how much a small change to y_i at time t affects E if everything else is left unchanged. We can now derive the learning equations of the backpropagation through time algorithm. We will first introduce the new variables z_i, called the adjoint variables, that will be determined by the system of differential equations in (7) with boundary conditions $z_i(t_1) = 0$ [2]. The learning equations are then (8) and (9).

$$\frac{dz_i}{dt} = \frac{1}{T_i} \cdot z_i - e_i - \sum_j \frac{1}{T_j} \cdot w_{ij} \cdot F'(x_j) \cdot z_j \tag{7}$$

$$\frac{\partial E}{\partial w_{ij}} = \frac{1}{T_i} \int_{t_0}^{t_1} y_i \cdot F'(x_j) \cdot z_j dt \tag{8}$$

$$\frac{\partial E}{\partial T_i} = \frac{1}{T_i} \int_{t_0}^{t_1} z_i \frac{dy_i}{dt} dt \tag{9}$$

The weights and time constants will be adjusted in such a way that the total error decreases; these corrections are then (10) and (11) where η_w and η_T are the learning rates.

[2] It has been proven that the system remains stable after a discretization if the condition $\Delta t \ll T_i, \forall_i$ is respected.

ESANN'1995 proceedings - European Symposium on Artificial Neural Networks
Brussels (Belgium), 19-20-21 April 1995, D-Facto public., ISBN 2-9600049-3-0, pp. 45-50

$$\Delta w_{ij} = -\eta_w \frac{\partial E}{\partial w_{ij}} \tag{10}$$

$$\Delta T_i = -\eta_T \frac{\partial E}{\partial T_i} \tag{11}$$

Note that these equations can be derived either using a finite difference approximation, the calculus of variation, the Lagrange multiplier, or even from the control theory of optimal control in dynamic programming using the Pontryagin Maximum Principle [2] [3].

Because of the extremely long learning phase of this kind of networks, we have tried several acceleration techniques like the addition of a momentum term, adjustment of the learning rate using line search and the method of Silva & Almeida [4]. We found that best results were achieved with the Silva & Almeida Method where each weight and time constant have their own adaptive learning rate [5].

2.4 Sampling and Training

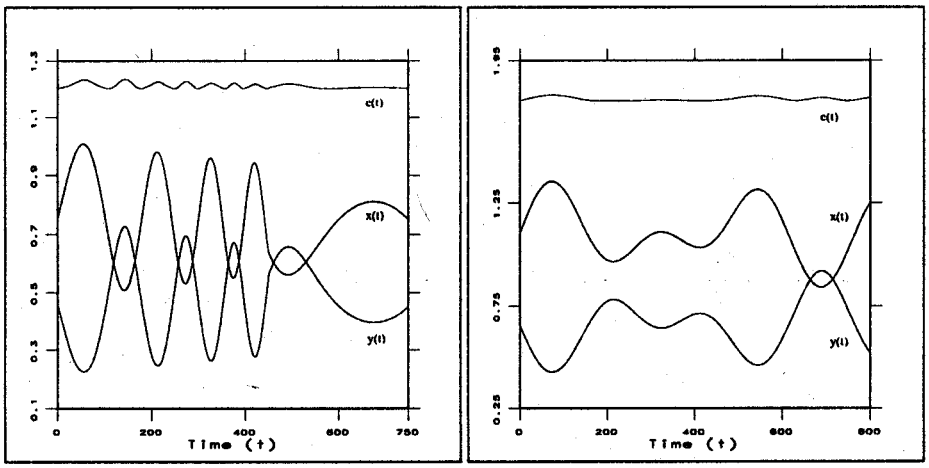

Figure 2: The figure on the left is a signal that varies the amplitude and frequency with time. The figure on the right is a superposition of two pure signals with different frequencies that vary their amplitude with time. In both cases the input signal to the network is labeled $x(t)$, the output of the network is labeled $y(t)$ and the superposition of both signals (taking into account the time delay that exists between the reference microphone and the secondary source) is labeled $e(t)$.

The signals $x(t)$, $y(t)$, and $e(t)$ (refer to figure 1), were sampled from an existing system that performs ANC with the Least Mean Squares algorithm to reduce the error signal $e(t)$. Different signals, in the 100 Hz to 500 Hz range, were sampled at $18\mu s$. with a monolithic A to D converter. These signals were first used to calculate the delay δ between the microphone that takes the reference signal and the secondary source. Note that this delay will affect the phase shift that the signal $y(t)$ must have with respect to

ESANN'1995 proceedings - European Symposium on Artificial Neural Networks
Brussels (Belgium), 19-20-21 April 1995, D-Facto public., ISBN 2-9600049-3-0, pp. 45-50

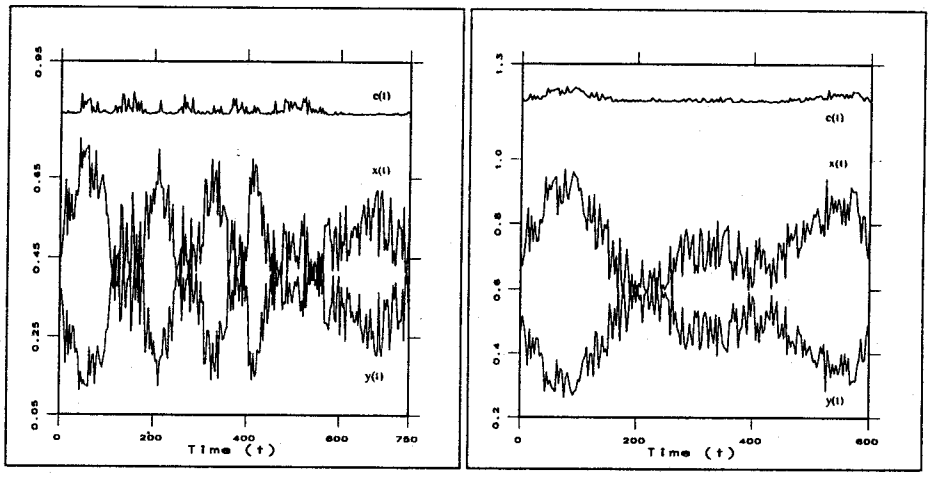

Figure 3: The figures above are the same as the ones in figure 2 but in this case the signals were distorted with 10% white noise.

the signal $x(t)$ (since we are dealing with traveling noise). This result was later used to generate a complete training set of signals varying in phase and amplitude, and finally, they were used to train and validate a fully connected 12 neuron network.

3 Results

Initial results show very good performance while feeding a trained network with different input signals. The network performs a perfect noise suppression when dealing with pure frequency signals, ie. one frequency in the 100-500 Hz range with no harmonics, and drastically reduces the error when dealing with more complex signals like: $x(t) = \sin(2\pi f_1 t) + \sin(2\pi f_2 t)$ where f_1 and f_2 are different frequencies or with signals where the frequency and amplitude vary with time (see figure 2). Very good results are also being obtained when dealing with noisy signals (see figure 3). In this case the input signals were perturbed with white noise and fed to the network. Note that here the error $e(t)$ was also significantly decreased.

Table 1 presents a summary of results we have obtained with pure single frequency signals, more complex signals with two frequency components that vary their frequency and amplitude with time as in figure 2, complex signals disturbed with 10% white noise as in figure 3, and pure white noise signals. In this table the reference and source signals, $x(t)$ and $y(t)$ respectively, were normalized to 0.0 to 1.0 range so that the measured error signal $e(t)$, which is the superposition of $x(t+\delta)$ with $y(t)$, is also normalized in a 0 to 1 range. The *error amplitude* in the table was taken as the maximum amplitude minus the minimum amplitude registered by the error signal $e(t)$. Note that there is a 81% noise reduction in the worst case and a 83% reduction in the best case for the existing system whereas we had a 92% noise reduction in the worst case and a 100% noise reduction in the best case for the neural system.

ESANN'1995 proceedings - European Symposium on Artificial Neural Networks
Brussels (Belgium), 19-20-21 April 1995, D-Facto public., ISBN 2-9600049-3-0, pp. 45-50

error amplitude	Set 0	Set 1	Set 2	Set 3	Set 4	Set 5
LMS system	0.11	0.19	0.19	0.14	0.13	0.63
neural system	0.00	0.032	0.053	0.020	0.033	0.148

Table 1: Here we present the results measured in the existing system and the simulated results of the neural controller. Sets 0 to 5 represent a pure single frequency signal, a pure signal with changes in frequency and amplitude with time, a pure dual frequency signal, a noisy signal with changes in frequency and amplitude, a noisy dual frequency signal, and pure white noise, in that order. Note that both signals,$x(t)$ and $y(t+\delta)$, where normalized to a 0.0 to 1.0 range. Here the error amplitude was taken as the maximum amplitude minus the minimum amplitude registered by the error signal $e(t)$.

4 Conclusion

We have introduced a Dynamic Recurrent Neural Network that replaces a finite impulse responce filter that traditionally updates its parameters via a Least Means Squares algorithm that minimizes an error function $e(t)$. Results show that the *neural controller* adapts much better reducing $e(t)$ better even for cases for which the network was not trained. It also performs well with noisy functions and even with pure white noise. Further work consists in the parallelization of the network and real time implementation.

References

[1] L. Blondel. Active noise control: Fundamentals and applications. Faculte Polytechnique de Mons, March 1994.

[2] J.P. Draye and G. Libert. Learning algorithm for dynamic recurrent neural networks based on the pontryagin maximum principle. Technical report, Faculte Polytechnique de Mons - PIP Laboratory, 1993.

[3] B.A. Pearlmutter. Learning state space trajectories in recurrent neural networks. *Neural Computation*, 1:263–269, 1989.

[4] W. Schiffman, M. Joost, and R. Werner. Optimization of the backpropagation algorithm for training multilayer perceptrons. Technical report, University of Koblenz, 1992.

[5] F. M. Silva and L.B. Almeida. Speeding up backpropagation. *Advanced Neural Computers*, pages 151–158, 1990.