Preprint:

Continuous-Time Temporal Back-Propagation with Adaptable Time Delays[1]

Shawn P. Day[2]
Michael R. Davenport[3]

August 1991
Revised: April 1992

To appear in: **IEEE Transactions on Neural Networks**

Abstract — **This paper extends back-propagation to continuous-time feed-forward networks with internal, adaptable time delays. The new technique is suitable for parallel hardware implementation, with continuous multidimensional training signals. The resulting networks can be used for signal prediction, signal production, and spatio-temporal pattern recognition tasks. Unlike conventional back-propagation networks, they can easily adapt while performing true signal prediction. We present simulation results for networks trained to predict future values of the Mackey-Glass chaotic signal, using its present value as an input. For this application, networks with adaptable delays had less than half the prediction error of networks with fixed delays, and about one-quarter the error of conventional networks. After training, the network can be operated in a signal production configuration, where it autonomously generates a close approximation to the Mackey-Glass signal.**

[1]This work was supported by the Natural Sciences and Engineering Research Council of Canada, the Science Council of British Columbia, and the Canadian Microelectronics Corporation.

[2]Shawn Day is with the Department of Electrical Engineering, 2356 Main Mall, University of British Columbia, Vancouver, B.C., Canada, V6T 1Z4 (email: shawnd@ee.ubc.ca).

[3]Michael Davenport is with the Department of Physics, 6224 Agricultural Road, University of British Columbia, Vancouver, B.C., Canada, V6T 1Z1 (email: davenpo@physics.ubc.ca).

I. INTRODUCTION

This paper describes an extended form of back-propagation [1]–[4] for multilayer feed-forward networks with internal, adaptable time delays. Adaptation occurs with continuous, time-varying signals, and is suitable for implementation in parallel hardware. One application is signal prediction, where the input to the network is a time-varying signal and the desired output is a prediction of the signal at a fixed time increment in the future. Other examples are signal classification, such as speech recognition, and signal production, where the output autonomously follows a desired trajectory. We have investigated performance on both signal prediction and signal production tasks.

The most common feed-forward networks are static, having no internal time delays and responding to a particular input by immediately generating a specific output. Static networks can respond to temporal patterns if the network inputs are delayed samples of the input signal. These networks, trained with standard back-propagation, have been used as adaptive filters for noise reduction and echo canceling [5], and for chaotic time series prediction [6, 7].

One limitation of static networks is their inability to respond to temporal patterns in the hidden node outputs. To respond to these patterns, the network must have delay elements within each of its layers. The benefits of internal delays were demonstrated in [8], where Time Delay Neural Networks (TDNNs) performed speech recognition tasks using fixed integral multiples of unit delays. The fixed delays allowed the networks to be trained efficiently with back-propagation by unfolding the temporal structure spatially and placing constraints on the duplicated weights. Unfortunately, with arbitrary or adaptable delays, the size of a temporally unfolded network increases rapidly with the number of layers and connections. A recently proposed "temporal back-propagation" algorithm [9, 10] can train a TDNN without spatial expansion or weight constraints, assuming fixed time delays and the discrete-time presentation of input patterns. A continuous-time generalization of this algorithm was presented in [11].

Here, we further generalize temporal back-propagation to include adaptable delays in addition to continuous-time signals. We later show that a network with adaptable delays can achieve smaller errors than a network with fixed delays, if they both have the same number of nodes and connections. As in conventional back-propagation, adaptation minimizes the difference between the network output and a target signal, using only local computation. Local computation means that each node requires information only from other nodes to which it connects, leading to simpler hardware implementations.

Adaptation occurs on-line, making it unnecessary to segment the training data into "epochs". The internal delay lines explicitly store recent information from the input signal, and the weights and time delay values implicitly store older information. The network can potentially track non-stationary signals because adaptation slowly replaces the old information stored in the weights and time delays.

In Section II we derive expressions for the variation in the total error, Ψ, with respect to time-localized changes in weights and delays, and define the back-propagated error signal, $\Delta(t)$. Section III describes a form of delayed gradient descent that minimizes the expected value of Ψ by adapting both the weights and the time delays. In Section IV we compare two implementations with respect to their accuracy and hardware requirements. Section V presents simulation results for an on-line chaotic signal prediction task and a signal production task, both based on the Mackey-Glass differential-delay equation. Finally, Section VI is a short summary.

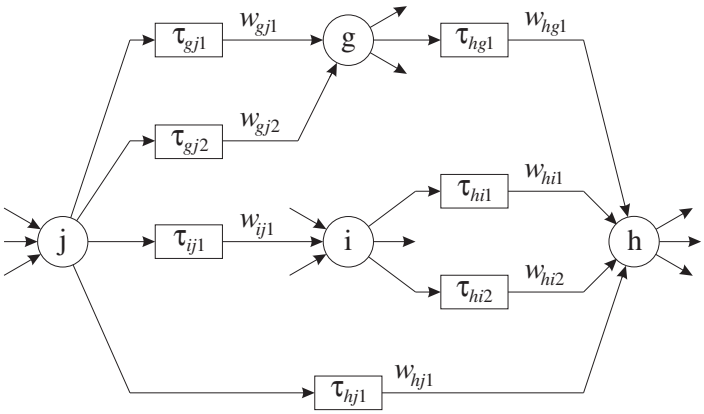

Figure 1: Section of a feed-forward network with internal time delays. Each connection has both a weight, w_{ijk}, and a time delay, τ_{ijk}, that adapt to minimize the error at the output. Features in the input signal spread out in time and space as they travel from left to right. Error signals propagate from right to left, but they are absent from the diagram for clarity.

II. DERIVATION OF THE ERROR GRADIENT

Fig. 1 shows a section of a typical feed-forward network with internal delays. Connections can skip across layers, and there can be more than one connection between two particular nodes. The kth connection from node j to node i has both a weight, $w_{ijk}(t)$, and a time delay, $\tau_{ijk}(t)$. The total input to node i is

$$y_i(t) = \sum_j \sum_k x_{ijk}(t) = \sum_j \sum_k w_{ijk}(t) o_j(t - \tau_{ijk}(t)),$$

where $x_{ijk}(t)$ is the kth input from node j, and $o_j(t)$ is either the output of node j, $o_j(t) = f_j(x_j(t))$, or a network input. The function, $f_j()$, is usually sigmoidal, but must be continuous and monotonically increasing.

Adaptation requires an input signal and a corresponding target signal, $\hat{o}(t)$. The sum,

$$s(t) = \sum_{m \in \mathcal{O}} (\hat{o}_m(t) - o_m(t))^2 = \sum_{m \in \mathcal{O}} s_m(t),$$

measures the instantaneous squared error between $\hat{o}(t)$ and the network output, where $\mathcal{O}$ is the set of all output nodes.

A. The Total Squared Error, Ψ, and its Functional Derivative

In a standard back-propagation network, a short-lived change to a weight results in a single, immediate, short-lived change to the output node activations. With time delayed connections, a short-lived change to a weight results in many future, short-lived changes at the outputs. Adaptation requires knowledge of how the total error depends on the weights and the time delays. We therefore define the total squared error,

$$\Psi = \int_{t=-\infty}^{\infty} s(t) dt = \int_{t=-\infty}^{\infty} \sum_{m \in \mathcal{O}} s_m(t) dt. \tag{1}$$

Although Ψ may be infinite, it has finite functional derivatives with respect to time localized changes in $w_{ijk}(t)$ and $\tau_{ijk}(t)$. Consider the perturbation σ_{ijk}, centered at time t_d, with magnitude h and duration ϵ:

$$\sigma_{ijk}(t, t_d, h, \epsilon) = \begin{cases} h & \text{if } (t_d - \epsilon/2) < t < (t_d + \epsilon/2) \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

Perturbing either $w_{ijk}(t)$ or $\tau_{ijk}(t)$ by $\sigma_{ijk}(t, t_d, h, \epsilon)$ causes measurable changes in Ψ that are approximately proportional to ϵh when ϵ and h are small. The functional derivatives,

$$\frac{\delta \Psi}{\delta w_{ijk}(t_d)} = \lim_{h,\epsilon \to 0} \left\{ \frac{\Psi[w_{ijk}(t) + \sigma_{ijk}(t, t_d, h, \epsilon)] - \Psi[w_{ijk}(t)]}{\epsilon\ h} \right\}, \tag{3}$$

$$\frac{\delta \Psi}{\delta \tau_{ijk}(t_d)} = \lim_{h,\epsilon \to 0} \left\{ \frac{\Psi[\tau_{ijk}(t) + \sigma_{ijk}(t, t_d, h, \epsilon)] - \Psi[\tau_{ijk}(t)]}{\epsilon\ h} \right\}, \tag{4}$$

describe how changes to the weights and delays influence the total squared error. They will be used as error gradients in Section III.

The dependence of Ψ on w_{ijk} involves many nonlinear terms, so $\Psi[w_{ijk}(t) + \sigma_{ijk}(t, t_d, h, \epsilon)]$ can be calculated only in the limit of vanishingly small h and ϵ. Expanding (3) using (1) gives

$$\frac{\delta \Psi}{\delta w_{ijk}(t_d)} = \lim_{h,\epsilon \to 0} \left\{ \frac{1}{\epsilon h} \sum_{m \in \mathcal{O}} \int_{t'=-\infty}^{\infty} \{ s_m(t')[w_{ijk}(t) + \sigma_{ijk}(t, t_d, h, \epsilon)] - s_m(t')[w_{ijk}(t)] \}\, dt' \right\}. \tag{5}$$

A particular weight, w_{ijk}, influences each output error, $s_m(t')$, through $p \in 1 \ldots N_{mi}$ different signal paths, each with a total delay of z_{mip}. Thus, $s_m(t')$ can be viewed as a function not of a single parameter, w_{ijk}, but of several parameters, $w_{ijk}(t' - z_{mip})$, where p acts as a parameter label. According to this interpretation, the notation

$$\frac{\partial s_m(t')}{\partial w_{ijk}(t' - z_{mip})}$$

represents the dependence of the output error at time t' on the weight at time $(t' - z_{mip})$, through path p.

We now expand $s_m(t')[w_{ijk}(t) + \sigma_{ijk}(t, t_d, h, \epsilon)]$ as a Taylor series, to first order in σ:

$$s_m(t')[w_{ijk}(t) + \sigma_{ijk}(t, t_d, h, \epsilon)] = s_m(t')[w_{ijk}(t)] + \sum_{p=1}^{N_{mi}} \sigma_{ijk}(t' - z_{mip}, t_d, h, \epsilon) \frac{\partial s_m(t')}{\partial w_{ijk}(t' - z_{mip})} + O(\sigma_{ijk}^2). \tag{6}$$

Substituting (6) into (5), and neglecting higher order terms yields

$$\frac{\delta \Psi}{\delta w_{ijk}(t_d)} = \lim_{h,\epsilon \to 0} \left\{ \frac{1}{\epsilon h} \sum_{m \in \mathcal{O}} \int_{t'=-\infty}^{\infty} \sum_{p=1}^{N_{mi}} \sigma_{ijk}(t' - z_{mip}, t_d, h, \epsilon) \frac{\partial s_m(t')}{\partial w_{ijk}(t' - z_{mip})} dt' \right\},$$

and the independent variable t' can be replaced by $(t'' + z_{mip})$, giving

$$\frac{\delta \Psi}{\delta w_{ijk}(t_d)} = \lim_{h,\epsilon \to 0} \left\{ \frac{1}{\epsilon h} \sum_{m \in \mathcal{O}} \sum_{p=1}^{N_{mi}} \int_{t''=-\infty}^{\infty} \sigma_{ijk}(t'', t_d, h, \epsilon) \frac{\partial s_m(t'' + z_{mip})}{\partial w_{ijk}(t'')} dt'' \right\}$$

$$= \lim_{h,\epsilon \to 0} \left\{ \frac{1}{\epsilon h} \sum_{m \in \mathcal{O}} \sum_{p=1}^{N_{mi}} \int_{(t_d-\epsilon/2)}^{(t_d+\epsilon/2)} h \frac{\partial s_m(t'' + z_{mip})}{\partial w_{ijk}(t'')} dt'' \right\}$$

$$= \sum_{m \in \mathcal{O}} \sum_{p=1}^{N_{mi}} \frac{\partial s_m(t_d + z_{mip})}{\partial w_{ijk}(t_d)}. \tag{7}$$

A similar development gives the functional derivative of Ψ with respect to each time delay:

$$\frac{\delta \Psi}{\delta \tau_{ijk}(t_d)} = \sum_{m \in \mathcal{O}} \sum_{p=1}^{N_{mi}} \frac{\partial s_m(t_d + z_{mip})}{\partial \tau_{ijk}(t_d)}. \tag{8}$$

B. Back-Propagated Error Signals

If we associate an error signal,

$$\Delta_j(t) = - \sum_{m \in \mathcal{O}} \sum_{p=1}^{N_{mj}} \frac{\partial s_m(t + z_{mjp})}{\partial y_j(t)}, \tag{9}$$

with each node, j, in the network, then (7) and (8) can be expressed in terms of $\Delta_i(t)$ by expanding them using the chain rule:

$$\frac{\delta \Psi}{\delta w_{ijk}(t)} = \sum_{m \in \mathcal{O}} \sum_{p=1}^{N_{mi}} \frac{\partial s_m(t + z_{mip})}{\partial y_i(t)} \frac{\partial y_i(t)}{\partial w_{ijk}(t)} = -o_j(t - \tau_{ijk}) \Delta_i(t), \tag{10}$$

$$\frac{\delta \Psi}{\delta \tau_{ijk}(t)} = \sum_{m \in \mathcal{O}} \sum_{p=1}^{N_{mi}} \frac{\partial s_m(t + z_{mip})}{\partial y_i(t)} \frac{\partial y_i(t)}{\partial x_{ijk}(t)} \frac{\partial x_{ijk}(t)}{\partial \tau_{ijk}(t)} = w_{ijk}(t)\, o_j'(t - \tau_{ijk})\, \Delta_i(t). \tag{11}$$

For output nodes, (9) is

$$\Delta_m(t) = -\frac{\partial s_m(t)}{\partial y_m(t)} = -\frac{\partial s_m(t)}{\partial o_m(t)} \frac{\partial o_m(t)}{\partial y_m(t)} = 2(\hat{o}_m(t) - o_m(t)) f_m'(y_m(t)), \tag{12}$$

and for hidden nodes it can be expanded as

$$\Delta_j(t) = - \sum_{m \in \mathcal{O}} \sum_{i \in \mathcal{C}_j} \sum_{k \in \mathcal{A}_{ij}} \sum_{p=1}^{N_{mi}} \frac{\partial s_m(t + \tau_{ijk} + z_{mip})}{\partial y_i(t + \tau_{ijk})} \frac{\partial y_i(t + \tau_{ijk})}{\partial y_j(t)}$$

$$= - \sum_{i \in \mathcal{C}_j} \sum_{k \in \mathcal{A}_{ij}} \frac{\partial y_i(t + \tau_{ijk})}{\partial y_j(t)} \sum_{m \in \mathcal{O}} \sum_{p=1}^{N_{mi}} \frac{\partial s_m(t + \tau_{ijk} + z_{mip})}{\partial y_i(t + \tau_{ijk})}$$

$$= f_j'(y_j(t)) \sum_{i \in \mathcal{C}_j} \sum_{k \in \mathcal{A}_{ij}} w_{ijk}(t + \tau_{ijk}) \Delta_i(t + \tau_{ijk}). \tag{13}$$

$\mathcal{A}_{ij}$ is the set of all connections from node j to node i, and $\mathcal{C}_j$ is the set of all i such that the output from node j feeds into node i.

This recursive expression for $\Delta_j(t)$ is valid for any feed-forward network, but its evaluation requires future values of Δ_i, making it noncausal. To overcome this problem, we define an "aged"

error signal, $\Delta_j(t,a_j) = \Delta_j(t-a_j)$, for the hidden nodes. Each age, a_j, must be greater than the maximum value of z_{mjp} for all m and p, so that $\Delta_j(t,a_j)$ can be defined recursively and causally as

$$\Delta_j(t,a_j) = f_j'(y_j(t-a_j)) \sum_{i\in\mathcal{C}_j} \sum_{k\in\mathcal{A}_{ij}} \Delta_i(t-[a_j-a_i-\tau_{ijk}],a_i) w_{ijk}(t-[a_j-\tau_{ijk}]). \qquad (14)$$

III. GRADIENT DESCENT ADAPTATION

Assuming the input and target signals are sample functions of random processes, we define the mean total squared error as

$$\mathcal{E} = E[\Psi], \qquad (15)$$

where the expected value represents an ensemble average. To minimize $\mathcal{E}$, the weights and time delays must adapt with changes proportional to the negative gradients,

$$\frac{dw_{ijk}}{dt} = -\mu \frac{\delta \mathcal{E}}{\delta w_{ijk}(t)} \quad \text{and} \quad \frac{d\tau_{ijk}}{dt} = -\nu \frac{\delta \mathcal{E}}{\delta \tau_{ijk}(t)},$$

where μ and ν are learning rates.

Due to causality constraints, it is difficult to estimate $\delta\mathcal{E}/\delta w_{ijk}(t)$ until time $t+a_j$, so we use a delayed version of gradient descent:

$$\frac{dw_{ijk}}{dt} = -\mu \frac{\delta \mathcal{E}}{\delta w_{ijk}(t-a_i)} = -\mu E\left[\frac{\delta \Psi}{\delta w_{ijk}(t-a_i)}\right] = \mu E[o_j(t-a_i-\tau_{ijk})\,\Delta_i(t,a_i)], \qquad (16)$$

$$\frac{d\tau_{ijk}}{dt} = -\nu \frac{\delta \mathcal{E}}{\delta \tau_{ijk}(t-a_i)} = -\nu E\left[\frac{\delta \Psi}{\delta \tau_{ijk}(t-a_i)}\right] = -\nu E[w_{ijk}(t-a_i)\,o_j'(t-a_i-\tau_{ijk})\,\Delta_i(t,a_i)]. \qquad (17)$$

If the learning rates, μ and ν, are small enough that the weights do not change significantly during a period equal to the maximum delay through the network, then these equations are a good approximation to true gradient descent. At high learning rates, the time delays can cause the adaptable parameters to overshoot their optimum values, leading to instability. We observed some instabilities during our simulations, but easily overcame the problem by choosing smaller learning rates. However, we believe the issue of stability deserves further analysis.

For ergodic signals and slow adaptation rates, the expected values in (16) and (17) can be estimated by time averages. One such estimate, $p_{ijk}(t)$, can be obtained with a leaky integrator,

$$p_{ijk}(t) + \eta \frac{dp_{ijk}}{dt} = o_j(t-a_i-\tau_{ijk})\,\Delta_i(t,a_i),$$

and a similar estimate, $q_{ijk}(t)$, can be obtained for the delays. The time constant, η, is similar to the discrete-time momentum parameter discussed in [4]. With these estimates, the adaptation equations become

$$\frac{dw_{ijk}}{dt} = \mu p_{ijk}(t) \quad \text{and} \quad \frac{d\tau_{ijk}}{dt} = -\nu q_{ijk}(t).$$

In the limit where η goes to zero, the leaky integrators follow their inputs precisely, giving

$$\frac{dw_{ijk}}{dt} = \mu o_j(t-a_i-\tau_{ijk})\Delta_i(t,a_i), \qquad (18)$$

$$\frac{d\tau_{ijk}}{dt} = -\nu\, w_{ijk}(t-a_i) o_j'(t-a_i-\tau_{ijk})\Delta_i(t,a_i). \qquad (19)$$

The preceding discussion completely defines the continuous-time temporal back-propagation process. A related method [12] for performing gradient descent in networks with fixed time delays does not take advantage of the computational savings incurred by back-propagating errors through the delays, and is not appropriate for networks with delays in all the connections. A different technique for learning time delay "windows" in feed-forward networks was demonstrated in [13] for a speech recognition task, but without a detailed derivation. For recurrently connected networks, [14] derives related expressions for learning output trajectories.

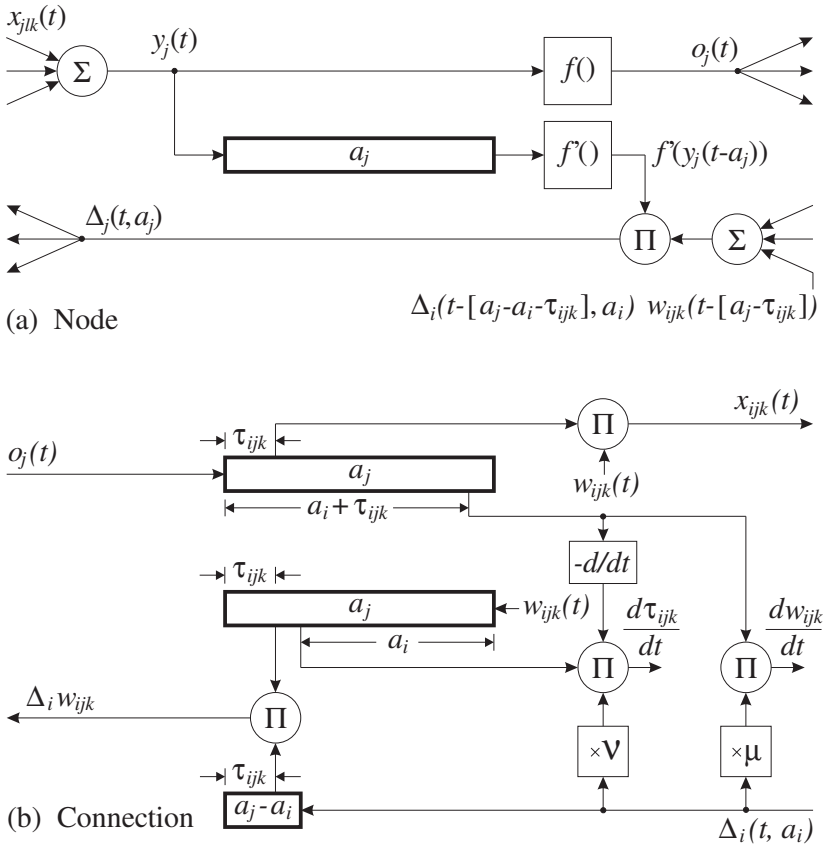

Figure 2: Block diagrams of (a) a node, and (b) a connection, for a direct but inefficient implementation of the network equations. Elongated boxes labeled with an age, a_j, represent tapped delay lines, with a maximum delay proportional to the length of the box. Input signals flow from left to right and error signals flow from right to left.

IV. IMPLEMENTATION CONSIDERATIONS

A. Nodes and Connections

Fig. 2 shows block diagrams for the direct implementation of both nodes and connections. Each node contains a delay line of length a_j, providing the delayed value $f_j'(y_j(t-a_j))$, and each connection contains tapped delay lines for o_j, w_{ijk}, and Δ_i. The time delay, τ_{ijk}, can be adjusted

by moving the taps. In a typical network, the number of connections increases as the square of the number of nodes, N, so the implementation requires $O(N^2)$ storage for the delayed signals.

Fig. 3 shows block diagrams with all the delay buffers moved from the connections to the nodes, reducing the storage requirements to $O(N)$. To accomplish this reduction, all delayed weight values must be replaced with $w_{ijk}(t)$, and the delay buffers must have a separate tap for each connection. For small learning rates, the weights change slowly, and this approximation introduces little error into the gradient calculations. For further reduction, we have used a function, $g_j(o_j)$, to calculate $f_j'(y_j)$ directly from o_j, removing the extra storage from the node in Fig. 2.

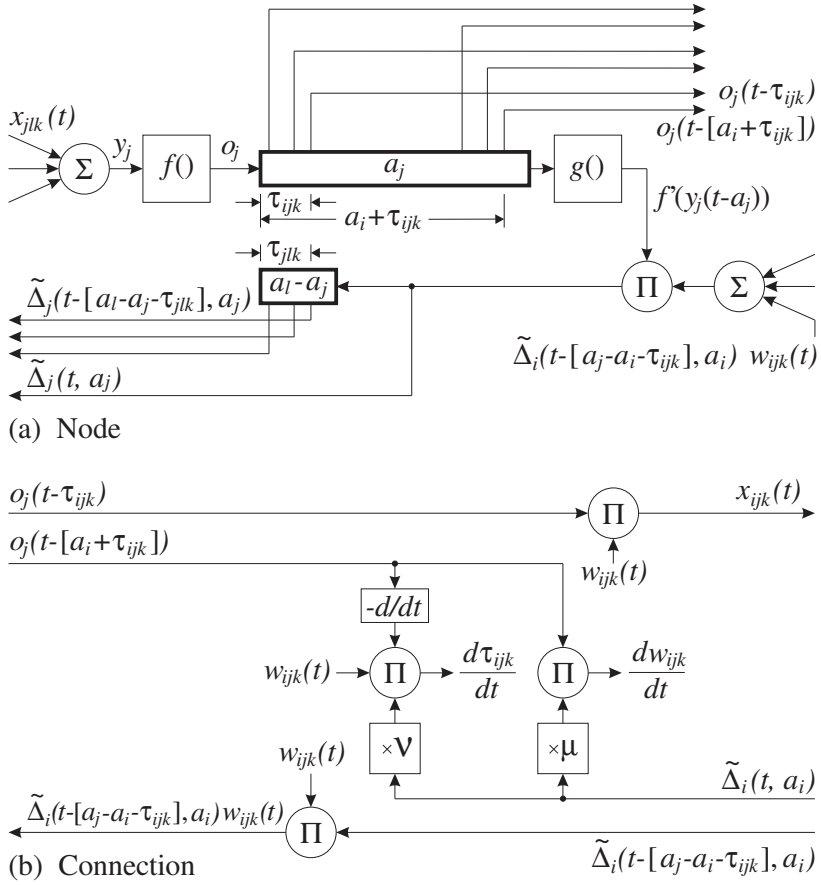

Figure 3: Block diagrams of (a) a node, and (b) a connection, with minimal storage requirements. The current value of $w_{ijk}(t)$ replaces the delayed values shown in Fig. 2. Adaptation still approximates gradient descent for small learning rates, but by moving all delay lines out of the connections, the total storage requirement scales linearly with the number of nodes.

B. Error Signal Ages

Appropriate age parameters, a_j, depend on the requirements of a particular implementation. It is desirable to make each a_j as small as possible, while keeping Δ_j causal, because small age values help to minimize storage requirements, maintain stability, and ensure that adaptation most closely

approximates true gradient descent. A simple solution is to assign an age of zero to all nodes in the output layer, and increment the age by a fixed amount for each preceding layer. We used this approach for our simulations, with a fixed increment of 25 time units. For a given set of ages, each τ_{ijk} must be kept smaller than $a_j - a_i$ by imposing some kind of limiting procedure during adaptation.

V. SIMULATION

A. Test Signal

The chaotic signal produced by integrating the Mackey-Glass delay-differential equation [15, 16],

$$\frac{dx(t)}{dt} = -bx(t) + a\frac{x(t-\tau)}{1+x(t-\tau)^{10}}, \tag{20}$$

provides a useful benchmark for testing predictive techniques [6, 17, 18, 19]. For comparison with these previous results, we chose $\tau = 17$, $a = 0.2$, $b = 0.1$, and trained the network to predict 6 time units into the future. We integrated (20) using a four-point Runge-Kutta method with a step size of 0.1. The initial conditions were $x(t) = 0.8$ for $t < 0$, and the equation was integrated up to $t = 1000$ to allow transients to die out. The resulting signal is quasi-periodic with a characteristic time of $t_{\text{char}} \approx 50$, lying on a strange attractor with a fractal dimension of approximately 2.1 [16]. To speed convergence [20], we removed the DC offset by subtracting 0.92534 before presentation to the network.

B. Network Topology and Parameters

Fig. 4 shows the network topology, which is intentionally similar to that in [6]. There are two hidden layers, consisting of ten sigmoidal nodes each, and a single linear output node. Instead of four input nodes, we used a single input node with four adaptable connections to each node in the first hidden layer. A single adaptable connection joins each other pair of nodes in adjacent layers. Thus, instead of an arbitrarily chosen, fixed embedding space as in [6], the adaptable delays allow the network to discover an efficient embedding space on its own. The choice of embedding space, in terms of its dimensionality and temporal spacing, has a significant effect on the network's learning ability [21].

For the hidden nodes, each $f_j()$ was a hyperbolic tangent with unity gain, and all input and output nodes had linear transfer functions for expanded dynamic range. Adaptable thresholds were implemented by connections from an additional node with a constant output. The initial delays were chosen pseudorandomly from a uniform distribution on the interval $[0, 20]$, and the initial weights were chosen from a uniform distribution on the interval $[-0.1, 0.1]$. A weight learning rate of $\mu = 0.005$ and a delay learning rate of $\nu = 1.0$ were selected, based on a few experimental training runs. For simulations incorporating momentum, we used a time constant of $\eta = 10$ for both the weights and delays. The network equations were integrated using a simple Euler method with a small fixed-length time step, and the input was the continually evolving Mackey-Glass signal, rather than a repeated, finite training set.

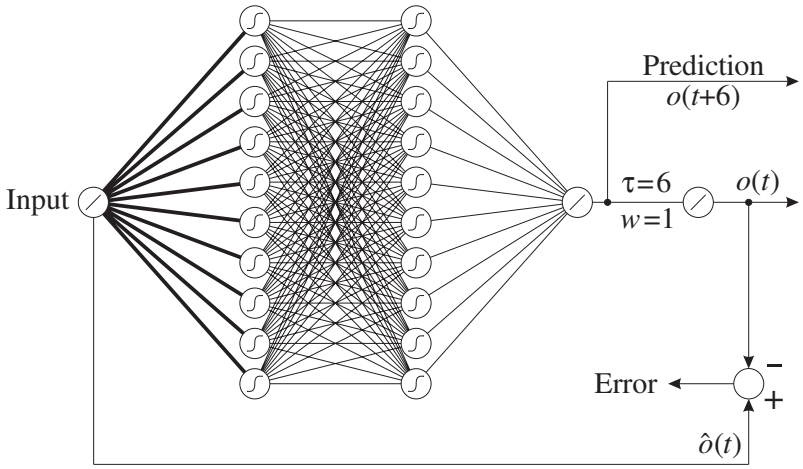

Figure 4: Network architecture for the adaptive prediction filter. Each heavy line between the linear input node and the first hidden layer represents four adaptable weights and time delays. All other connections contain a single adaptable weight and delay, except the connection between the last hidden node and the output node, which has a fixed weight of 1 and a fixed delay of 6. There are a total of 150 adaptable connections and 21 adaptable biases. Each node has either a linear or a sigmoidal transfer function, represented by the node symbols. Minimizing the output error causes the network to predict the input signal 6 time units into the future.

C. Configuration as an Adaptive Prediction Filter

Fig. 4 shows a second output node, connected to the prediction output with a fixed weight ($w = 1$) and a fixed delay ($\tau = 6$). This node allows the network to adapt while performing true signal prediction. Adaptation minimizes the error between $o(t)$ and the network input, $\hat{o}(t)$, forcing the prediction output to produce a signal that approximates $\hat{o}(t + \tau)$.

A conventional back-propagation network, without the additional output node, requires the input signal to be delayed by $T = 6$ so the undelayed signal can be used as the target. However, the resulting configuration "predicts" only the present value of the signal. Adaptive prediction of future values is not possible because removal of the input delay prevents further adaptation. Alternatively, an identical copy of the network, without the delayed input, can perform true prediction if its weights follow the weights in the adapting network [22]. Unfortunately, this type of weight mirroring doubles the required hardware. Additionally, the characteristics of the corresponding network elements must be very closely matched, making it impractical for an analog physical implementation. Dynamic back-propagation [12] offers a better solution, but it requires $O(N^2)$ storage and some nonlocal computations. Previous prediction networks [6, 18] simply assumed the availability of a target signal during training.

D. Signal Prediction Results

Fig. 5 shows learning curves for the adaptable time delay network, an identical network with fixed time delays, and a conventional back-propagation network. The conventional network had the same topology as in [6], where it performed significantly better than the linear predictive

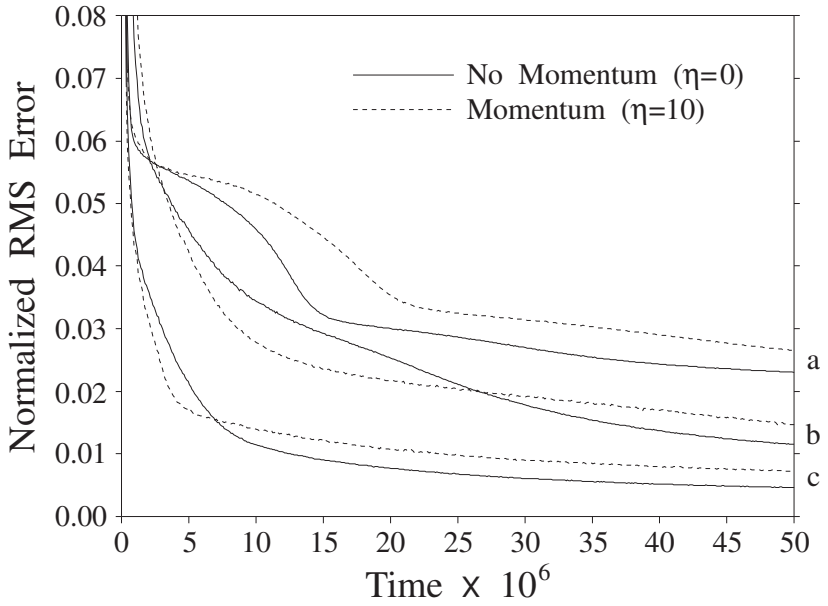

Figure 5: Learning curves showing the normalized rms error (NRMSE) of networks trained to predict the Mackey-Glass signal an interval, $T = 6$, into the future. The curves are for: (a) A static network with identical topology to that used in [6]. Four input nodes represented fixed delays of 0, 6, 12, and 18. (b) A network with randomly chosen fixed delays. (c) A network with adaptable delays, initially identical to (b). The dashed curves are for the same networks trained with momentum ($\eta = 10$). In all cases, the weight learning rate was $\mu = 0.005$, and for part (c) the delay learning rate was $\nu = 1$. The averages used to estimate the NRMSE were over time intervals of length 10^5.

method [23] and the Gabor polynomial predictive method [24].

As in [17], we used the normalized rms error,

$$\text{NRMSE} = \frac{E[\,(\hat{o}(t) - o(t))^2\,]^{1/2}}{E[\,(\hat{o}(t) - E[\hat{o}(t)]\,)^2\,]^{1/2}},$$

to monitor performance. An NRMSE of zero indicates perfect prediction, and an NRMSE of unity means the prediction is no better than a constant predictor at $E[\hat{o}(t)]$. All the networks showed approximately exponential convergence, and after $5 \cdot 10^7$ time units appeared to be approaching asymptotes. At that point, the NRMSE values were 0.023, 0.012, and 0.005 for the conventional network, the fixed-delay network, and the adaptable-delay network, respectively. As discussed in the previous section, it is important to note that the conventional network was not performing true prediction during the learning process.

Fig. 5 also shows learning curves for the same three networks trained with momentum. Convergence was faster during the initial stages, but then leveled off at slightly higher NRMSE values of 0.027, 0.015, and 0.007. These results suggest that the momentum time constant should decrease slowly as the error decreases, but we have not investigated variable momentum. In experiments where we froze the weights and delays after training, we found that networks trained with momentum had lower prediction errors, despite their higher errors while adapting. It appears that

networks without momentum adapt quickly to short-term characteristics of the input signal. Momentum smooths out this adaptation over a longer period, resulting in worse adaptive performance, but better frozen performance.

To predict further into the future, identical copies of the $T = 6$ network can be cascaded to provide outputs for $T = 12, 18, 24,...$ Alternatively, the network can be trained to directly predict further into the future by increasing the fixed delay at the output node. Increasing this delay may degrade the stability of the prediction filter, requiring smaller learning rates. Theoretically, the cascade method should provide more accurate predictions for chaotic signals [19].

E. Signal Production Results

An interesting application of the trained network is to freeze the weights and time delays, remove the input signal, and connect the output labeled $o(t)$ in Fig. 4 directly to the input node, creating a recurrent network. The resulting dynamical system spontaneously generates its own version of the Mackey-Glass signal with the power spectrum shown in Fig. 6. For comparison, the figure includes a spectrum of the actual Mackey-Glass signal, integrated using a four-point Runge-Kutta method with a step size of 0.1. The diagram also shows the fractional error between these spectra for networks trained with and without momentum. The network trained with momentum produced a much more accurate spectrum, despite its higher on-line prediction error in Fig. 5. This result is consistent with our observation that networks trained with momentum have lower prediction errors after the weights and delays have been frozen. The conventional back-propagation network and the network with fixed delays were unstable when connected in this configuration, producing high frequency oscillations.

VI. SUMMARY

We have described a method for adapting both weights and time delays in a feed-forward network. The main implementation difficulty is the coordination of the feed-forward signals with the back-propagated error signals. Our solution is to assign a fixed age to each node, and add delay buffers to ensure that all signals are of the proper age. If the delay learning rate is zero and the network equations are integrated using a simple Euler technique, the new method is equivalent to temporal back-propagation [10]. Furthermore, if the time delays are fixed at zero, temporal back-propagation reduces to standard back-propagation.

The network can adapt while performing true signal prediction, as opposed to conventional back-propagation networks which require an extra delay at the input. We tested a network in this configuration by training it to predict the Mackey-Glass chaotic signal at time $t + 6$ using observations up to time t. After $5 \cdot 10^7$ time units, the prediction error was about one-quarter that of a comparable static network. Our simulations suggest that momentum does not improve adaptive performance, but it appears to be useful when the weights and delays must be frozen after training. With its output connected directly to its input, a frozen predictor network spontaneously generates an imitation of its training signal. The spectrum of the synthesized signal appears to approximate the training signal spectrum more closely for networks trained with momentum. Networks without adaptable delays were unstable in this configuration. These results demonstrate that adaptable time delays can reduce the output error of a feed-forward network with a fixed number of nodes and connections.

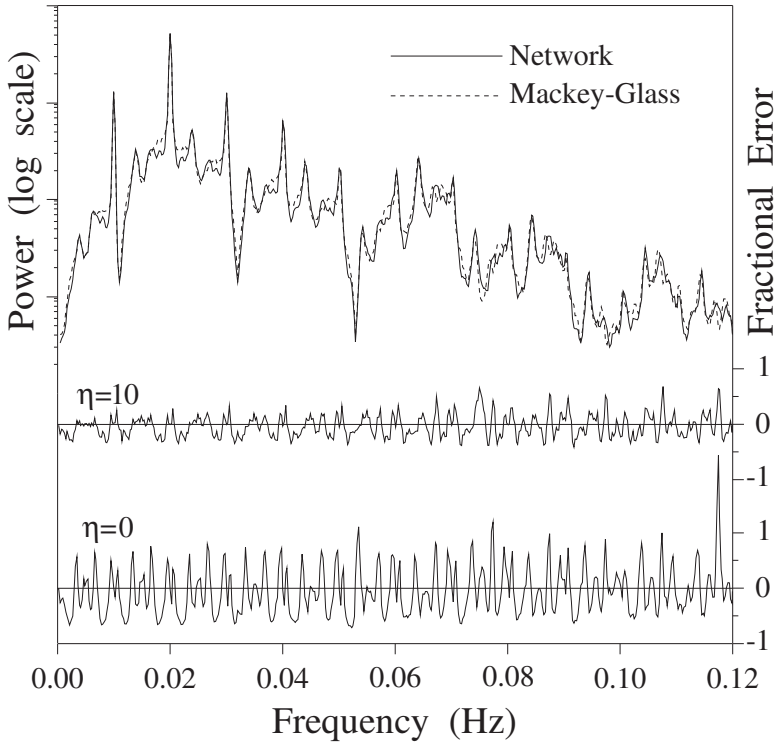

Figure 6: (A) A comparison of power spectra generated by the Mackey-Glass equation and the adaptive-delay network trained with momentum. The weights and time delays in the network of Fig. 4 were frozen after training up to time $t = 3 \cdot 10^7$, and a recurrent network was formed by removing the input signal and connecting the output, $o(t)$, to the input. The resulting dynamical system spontaneously generated its own version of the Mackey-Glass signal, with the power spectrum shown above. The spectra are averages of 100 Fast Fourier Transforms, each using 4096 samples taken at unit intervals, and a Hamming data window. The fractional error is the difference between the two spectra, divided by the magnitude of the Mackey-Glass spectrum. The diagram also shows the fractional error for a network trained without momentum ($\eta = 0$).

REFERENCES

[1] P. J. Werbos, *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. PhD thesis, Harvard University, 1974.

[2] D. B. Parker, "Learning-logic," Tech. Rep. TR-47, Sloan School of Management, M.I.T., Cambridge, MA, 1985.

[3] Y. Le Cun, "A learning scheme for asymmetric threshold networks," in *Proceedings Cognitiva 85*, (Paris), pp. 599–604, 1985.

[4] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, *Learning Internal Representations by Error Propagation*, vol. 1 of *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, ch. 8, pp. 318–362. Cambridge, MA: MIT Press, 1986.

[5] B. Widrow and R. Winter, "Neural nets for adaptive filtering and adaptive pattern recognition," *Computer*, vol. 21, pp. 25–39, Mar. 1988.

[6] A. Lapedes and R. Farber, "Nonlinear signal processing using neural networks: Prediction and system modeling," Tech. Rep. LA-UR-87-2662, Los Alamos National Laboratory, July 1987.

[7] A. S. Weigend, B. A. Huberman, and D. E. Rumelhart, "Predicting the future: A connectionist approach," *International Journal of Neural Systems*, vol. 1, no. 3, pp. 193–209, 1990.

[8] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. J. Lang, "Phoneme recognition using time-delay neural networks," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 37, pp. 328–339, Mar. 1989.

[9] E. A. Wan, "Temporal backpropagation for FIR neural networks," in *International Joint Conference on Neural Networks, 1990*, vol. 1, (San Diego, CA), pp. 575–580, June 1990.

[10] E. A. Wan, "Temporal backpropagation: An efficient algorithm for finite impulse response neural networks," in *Proceedings of the 1990 Connectionist Models Summer School* (D. S. Touretzky, J. L. Elman, T. J. Sejnowski, and G. E. Hinton, eds.), (San Mateo, CA), pp. 131–137, Morgan Kaufmann, 1990.

[11] S. P. Day and D. S. Camporese, "Continuous-time temporal back-propagation," in *International Joint Conference on Neural Networks 1991*, vol. 2, (Seattle, WA), pp. 95–100, July 1991.

[12] K. S. Narendra and K. Parthasarathy, "Gradient methods for the optimization of dynamical systems containing neural networks," *IEEE Transactions on Neural Networks*, vol. 2, pp. 252–262, Mar. 1991.

[13] U. Bodenhausen and A. Waibel, "The Tempo 2 algorithm: Adjusting time-delays by supervised learning," in *Advances in Neural Information Processing Systems 3* (R. P. Lippmann, J. E. Moody, and D. S. Touretzky, eds.), pp. 155–161, San Mateo, CA: Morgan Kaufmann, 1991.

[14] B. A. Pearlmutter, "Dynamic recurrent neural networks," Tech. Rep. CMU-CS-90-196, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, Dec. 1990.

[15] M. C. Mackey and L. Glass, "Oscillation and chaos in physiological control systems," *Science*, vol. 197, pp. 287–289, July 1977.

[16] J. D. Farmer, "Chaotic attractors of an infinite-dimensional dynamical system," *Physica D 4*, pp. 366–393, 1982.

[17] J. D. Farmer and J. J. Sidorowich, "Predicting chaotic time series," *Physical Review Letters*, vol. 59, pp. 845–848, Aug. 1987.

[18] J. Moody and C. J. Darken, "Fast learning in networks of locally-tuned processing units," *Neural Computation*, vol. 1, no. 2, pp. 281–294, 1989.

[19] M. Casdagli, "Nonlinear prediction of chaotic time series," *Physica D 35*, pp. 335–356, 1989.

[20] D. J. Burr, "Experiments on neural net recognition of spoken and written text," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 36, pp. 1162–1168, July 1988.

[21] W. C. Mead, R. D. Jones, Y. C. Lee, C. W. Barnes, G. W. Flake, L. A. Lee, and M. K. O'Rourke, "Using CNLS-Net to predict the Mackey-Glass chaotic time series," in *International Joint Conference on Neural Networks 1991*, vol. 2, (Seattle, WA), pp. 485–490, July 1991.

[22] B. Widrow, J. M. McCool, M. G. Larimore, and C. R. Johnson, Jr., "Stationary and nonstationary learning characteristics of the LMS adaptive filter," *Proceedings of the IEEE*, vol. 64, pp. 1151–1162, Aug. 1976.

[23] J. Makhoul, "Linear prediction: A tutorial review," *Proceedings of the IEEE*, vol. 63, pp. 561–580, Apr. 1975.

[24] D. Gabor, W. P. L. Wilby, and R. Woodcock, "A universal non-linear filter, predictor and simulator which optimizes itself by a learning process," *IEE Proceedings*, vol. 108B, pp. 422–438, 1961.