TECHNICAL RESEARCH REPORT

Learning with the Adaptive Time-Delay Neural Network

by D-T. Lin, P.A. Ligomenides and J.E. Dayhoff

T.R. 93-49

The Institute for Systems Research is supported by the National Science Foundation Engineering Research Center Program (NSFD CD 8803012), Industry and the University

Learning with the Adaptive Time-Delay Neural Network [1,2]

Daw-Tung Lin, Panos A. Ligomenides and Judith E. Dayhoff

University of Maryland

College Park, MD 20742.

May 17, 1993

[2]This work was supported in part by the Institute for Systems Research at the University of Maryland, under Grant CDR-88-03012 from the NSF, and also by contract N00014-90K-2010 from the Naval Research Laboratory.

Abstract

The Adaptive Time-delay Neural Network ($ATNN$), a paradigm for training a nonlinear neural network with adaptive time-delays, is described. Both time delays and connection weights are adapted on-line according to a gradient descent approach, with time delays unconstrained with respect to one another, and an arbitrary number of interconnections with different time delays placed between any two processing units. Weight and time-delay adaptations evolve based on inputs and target outputs consisting of spatiotemporal patterns (e.g. multichannel temporal sequences). The $ATNN$ is used to generate circular and figure-eight trajectories, to model harmonic waves, and to do chaotic time series predictions. Its performance outstrips that of the time-delay neural network ($TDNN$), which has adaptable weights but fixed time delays. Applications to identification and control as well as signal processing and speech recognition are domains to which this type of network can be appropriately applied.

1 Introduction

Understanding the dynamic behavior of neural networks is a key issue in the development of network architectures that recognize and produce temporal sequences and spatiotemporal patterns. Whereas simple feed-forward networks can be trained to accomplish pattern recognition tasks with complex nonlinear boundaries, they are limited to processing static patterns - patterns that are fixed rather than temporal in nature. We propose here a network that overcomes this limitation and that is capable of processing temporally modulated signals. Networks with this capability can play an important role in applications domains that have naturally time-varying properties to their signals and dynamic situations. These domains include identification and control as well as signal processing and speech recognition.

An important contribution in this area has been the time-delay neural network (*TDNN*) proposed by Waibel et al [26], which employs time-delays on connections in feedforward network and has been successfully applied to speech recognition [27, 10]. The time-delay neural network also classifies spatiotemporal patterns and provides robustness to noise and graceful degradation [15]. However, a limitation of the *TDNN* as originally posed [26] is its inability to learn or adapt the values of the time delays. Time delays are fixed initially and remain the same throughout training. As a result, the system may have poor performance due to the inflexibility of time delays and due to mis-match between the choice of time delay values and the temporal location of the important information in the input patterns. In addition, the system performance may vary depending on the range of the time delay values.

To overcome this limitation, we have used an Adaptive Time Delay Neural Network model (*ATNN*) [13]. This network adapts its time delay values as well as its weights during training,

to better accommodate to changing temporal patterns, and to provide more flexibility for optimization tasks. The *ATNN* used here allows arbitrary placement of time delays along interconnections and adapts those time delays independently of one another. Furthermore, time-windows are not used as in previous work [2, 26] but instead classification relies on a set of individual time delay values associated with each interconnection. Although other artificial neural network architectures that make use of time delays have been suggested [23, 6, 2, 5], these paradigms employ different training rules or different network topologies, and the network presented here is simple to use and has a general formulation.

In this paper, we describe the *ATNN* network architecture and learning paradigm. The derivation of the *ATNN* training rule is given in the appendix. Applications are shown to trajectory generation, with trajectories of different topologies. *ATNN* modeling of harmonic waves is then described, followed by prediction of a chaotic time series. Performance comparisons show that the *ATNN* has greater capability that its predecessor, the *TDNN*. In conclusion, we discuss the relationship of the *ATNN* to other networks, its biological plausibility, and the applications domains to which it would natually apply.

2 Adaptive Time Delay Neural Model

2.1 Network Architecture

The proposed *ATNN* model employs modifiable time delays along the interconnections between two processing units, and both time delays and weights are adjusted according to system dynamics in an attempt to achieve the desired optimization. The schematic architecture of the connections from one processing unit to another processing unit of the

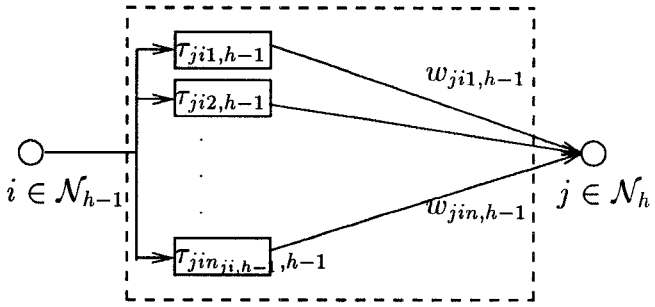

Figure 1: Delay Block: Basic time-delay connections between two processing units (node i of layer $h-1$ and node j of layer h) in *ATNN*, and $n_{ji,h-1}$ is the number of delays applied.

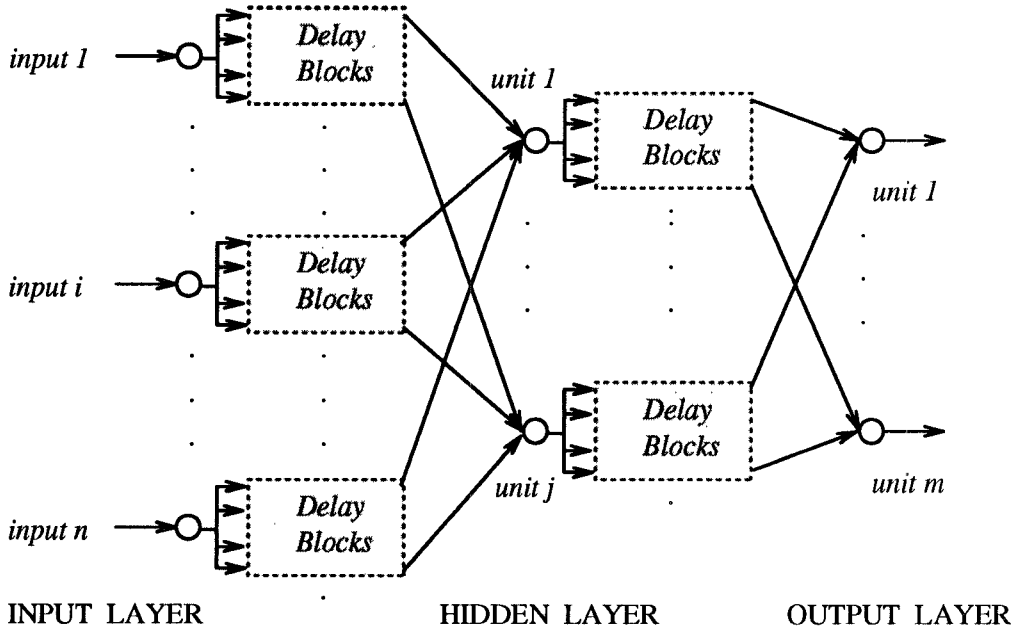

Figure 2: Layer *ATNN*: symbolic diagram

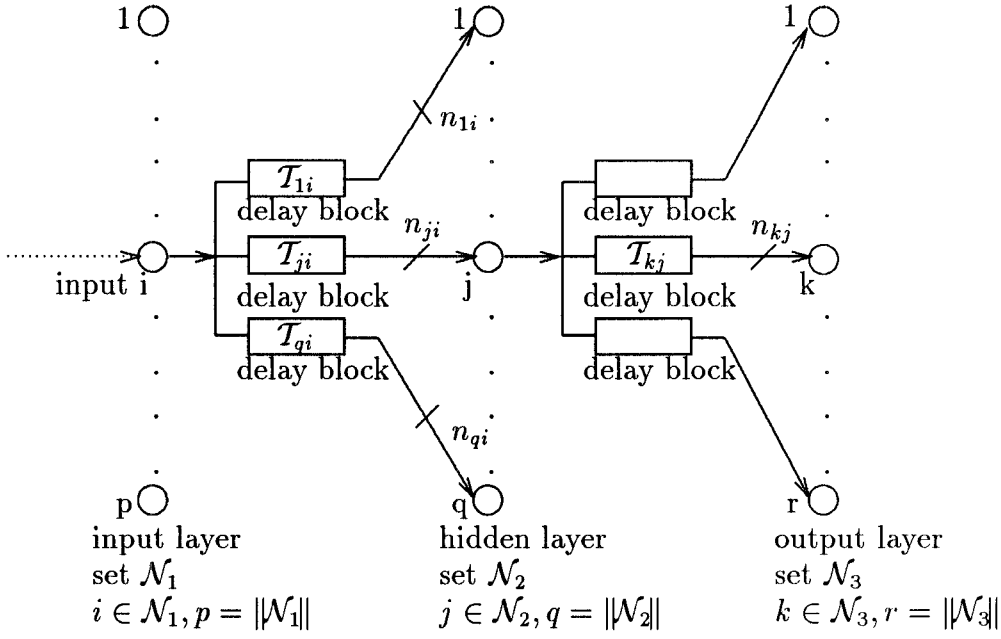

Figure 3: Three layered *ATNN*

ATNN is depicted in Figure 1. The configuration of multiple interconnections between a single pair of units, each with its own delay, is called a delay block. Node i of layer $h-1$ is connected to node j of the next layer h, with the connection line having an independent time-delay $\tau_{jik,h-1}$ and synaptic weight $w_{jik,h-1}$.

The entire network is constructed by the delay blocks which connect neurons layer by layer as illustrated in Figure 2. The multilayered network is not necessarily fully connected layer by layer, but may be sparsely connected, and connections can skip layers. It is not necessary to have the same number of delays for different units in the same layer or the same delay values from different units, since the computation is local for each interconnection. Each connection can have an arbitrary delay value. For the sake of simplicity, in this discussion we assume the network is layered and fully or sparsely connected layer by layer. To illustrate, a three layered *ATNN* is shown in Figure 3.

Next we propose definitions that are used to describe a general *ATNN* architecture with flexible configuration.

Definition 1 *n_{ji} is the number of time-delays employed of the connection to node i from node j (i.e., the number of samples in the time frame $\mathcal{T}_{ji}$).*

Definition 2 Time frame: *Time frame $\mathcal{T}_{ji}$ is a set of time delays $(\tau_{ji1}, ..., \tau_{jin_{ji}})$ employed on the connections to node j from node i.*

Definition 3 *The set of time frames $\mathcal{T}_i$ for connections that originate at node i is defined as:*

$\mathcal{T}_i = (\mathcal{T}_{1i}, \cdots, \mathcal{T}_{ji}, \cdots, \mathcal{T}_{qi})$.

Definition 4 *$p_{ji}(t, \mathcal{T}_{ji})$ is the set of signal values transmitted to node j from node i via time frame $\mathcal{T}_{ji}$ at time t. Thus, $p_{ji}(t, \mathcal{T}_{ji}) = [p_{ji}(t - \tau_{ji1}), ..., p_{ji}(t - \tau_{jin_{j_i}})]$.*

Definition 5 *$P_i(t, \mathcal{T}_i)$ is the set of signal values transmitted from node i to other nodes at time t. Thus,*

$$P_i(t, \mathcal{T}_i) = \begin{pmatrix} p_{1i}(t, \mathcal{T}_{1i}) \\ \vdots \\ p_{ji}(t, \mathcal{T}_{ji}) \\ \vdots \\ p_{qi}(t, \mathcal{T}_{qi}) \end{pmatrix} = \begin{pmatrix} [p_{1i}(t - \tau_{1i1}), ..., p_{1i}(t - \tau_{1in_{1i}})] \\ \vdots \\ \left[p_{ji}(t - \tau_{ji1}), ..., p_{ji}(t - \tau_{jin_{j_i}})\right] \\ \vdots \\ \left[p_{qi}(t - \tau_{qi1}), ..., p_{qi}(t - \tau_{qin_{qi}})\right] \end{pmatrix}$$

Let $J_{i,h}$ be a set of nodes that receive connections from node i on layer h. Node i transmits the set of pattern values $P_i(\mathcal{T}_i)$ to selected subset $J_{i,h}$. In other words, $\forall j \in J_{i,h}$, node j

receives pattern $P_i(\mathcal{T}_{ji})$ from node i through a time frame $\mathcal{T}_{ji}$. The samples in the time frame $\mathcal{T}_{ji}$ correspond to the delays $\tau_{jix,h}$, where $x = 1, 2, \cdots, n_{ji}$, where $n_{ji} = \|\mathcal{T}_{ji}\|, j \in J_{i,h}$, and h is the index of the layer in which i belongs. In the definitions above, we have omitted the layer index h for simplicity.

Definition 6 *$\mathcal{N}_h$ is the set of nodes $\{1, 2, ..., |\mathcal{W}_{h-1}|\}$ of layer h*

Thus, each node j of layer h (eg. $j \in \mathcal{N}_h$), receives n_{ji} inputs from each $i \in \mathcal{I}_{j,h}$, where $\mathcal{I}_{j,h}$ is the subset of nodes on layer $h-1$ ($\mathcal{I}_{j,h} \subseteq \mathcal{N}_{h-1}$) that connects to unit j on layer h. The total number of inputs to j is: $m_j = \sum_{i \in \mathcal{I}_{j,h}} n_{ji}$.

In this paper, we assume that $p = \|\mathcal{W}_1\|, q = \|\mathcal{W}_2\|, r = \|\mathcal{W}_3\|$ are fixed, and that the connectivity between layers is fixed ($\|J_{i,h-1}\|$ for the same layer is fixed), and for the sake of simplity, we assume $\|J_{i,1}\| = \|\mathcal{W}_2\|$ and $\|J_{i,2}\| = \|\mathcal{W}_3\|$ (fully connected). In general, the number of samples in the time frame, n_{ji}, and the values of the delays (therefore the size of $\mathcal{T}_{ji}$) are variables. In this paper, we assume that n_{ji} is fixed (selected) and allow $\tau_{jix,h-1}$ to be variable. We also assume that n_{ji} is the same for all $i \in \mathcal{I}_{j,h}$ and for all $j \in J_{i,h-1}$. If n_{ji} is the same (fixed) for all $i \in \mathcal{I}_{j,h}$, then $m_j = n_{ji} \cdot \|\mathcal{I}_{j,h}\|$.

Definition 7 *n_{h-1} is defined as the number of time-delays on connections originating at node i on layer $h-1$ where $n_{h-1} = n_{ji}$ for all j.*

Then our model possesses the following property:

Property 1 *For each node $i \in \mathcal{N}_{h-1}$ connecting to all $j \in J_{i,h-1}, J_{i,h-1} \subseteq \mathcal{N}_h$, there is a time frame $\mathcal{T}_{ji}$ with n_{h-1} samples, corresponding to the delays $\tau_{jix,h-1}, x = 1, \cdots, n_{h-1}$. Inputs to node j (on layer h) are from node i (on layer $h-1$), and there are $n_{h-1} \cdot \|\mathcal{W}_{h-1}\|$ such inputs.*

In this model, the adaptation variables are time-delay and weight ($\tau_{jik,h}$ and $w_{jik,h}$). Each node sums up the net inputs from the activation values of the previous neurons, through the corresponding time-delays on each connection line, i.e. at time t_n unit j on layer h receives a weighted sum :

$$S_{j,h}(t_n) = \sum_{i \in \mathcal{N}_{h-1}} \sum_{k=1}^{K_{ji,h-1}} w_{jik,h-1} \cdot a_{i,h-1}(t_n - \tau_{jik,h-1}) \tag{1}$$

where

$a_{i,h-1}(t_n - \tau_{jik,h-1})$ $\equiv$ the activation level of unit i on layer $h-1$ at $t_n - \tau_{jik,h-1}$

$\tau_{jik,h-1}$ $\equiv$ the time-delay of the kth connection to node j of layer h from node i of layer $h-1$

$K_{ji,h-1}$ $\equiv$ the total number of connections to node j (layer h) from node i of layer $h-1$

$w_{jik,h-1}$ $\equiv$ the synapse weight of the kth connection to node j from node i of layer $h-1$, and $k = 1, 2, ..., K_{ji,h-1}$

Then the output of node j is governed by a nondecreasing differentiable function f of the net input (sigmoid function is selected in this paper) :

$$a_{j,h}(t_n) = \begin{cases} f_{j,h}(S_{j,h}(t_n)) & \text{if } h \geq 2 \\ a_{j,0}(t_n) & \text{if } h = 1 \end{cases} \tag{2}$$

where

$$f_{j,h}(x) = \frac{\beta_{j,h}}{1 + e^{-\alpha_{j,h}x}} - \gamma_{j,h} \tag{3}$$

and $a_{j,0}(t_n)$ denotes the sampled value of the jth channel of the input signal at time t_n and

$\alpha_{j,h}$, $\beta_{j,h}$ and $\gamma_{j,h}$ are real numbers which define the symmetry point $(0, \frac{\beta_{j,h}}{2} - \gamma_{j,h})$ of $f_{j,h}(x)$ if x is symmetric to 0, the range $[-\gamma_{j,h}, \beta_{j,h} - \gamma_{j,h}]$ of $f_{j,h}(x)$ and the steepness of $f_{j,h}(x)$ (e.g. $f'_{j,h}(0) = \frac{\alpha_{j,h}\beta_{j,h}}{4}$). For simplicity, we use the same sigmoid function for each node in this paper, although in practice $f_{j,h}$ may differ and be modifiable for each node.

2.2 Learning Paradigm

The adaptation of the delays and weights are derived based on the gradient descent method to minimize the energy or cost function E during training. Weight modification is based on error back-propagation [17] and the mathematical derivation of the time-delay modifications is described from a gradient descent approach in [14, 13]. The training set consists of a sequence of spatiotemporal patterns and target outputs over time. An instantaneous error measure is defined as MSE:

$$E(t_n) = \frac{1}{2} \sum_{j \in \mathcal{N}_L} (d_j(t_n) - a_{j,L}(t_n))^2 \tag{4}$$

where L denotes the output layer and $d_j(t_n)$ indicates the desired (target) value of output node j at time t_n.

Furthermore, the continuous real values of desired output are condensed to the center regime of the sigmoid output. Specifically, the target values are scaled to range $[-c\gamma_{j,h}, c(\beta_{j,h} - \gamma_{j,h})]$, where $-\gamma_{j,h}$ and $(\beta_{j,h} - \gamma_{j,h})$ are the upper bound and lower bound of the sigmoid function respectively, and c denotes a scale factor. We selected the scale factor $c = 0.5$ in this paper, which significantly improved the mapping accuracy by a factor of 2 to 10 in terms

of $NMSE$ [1] compared to our previous simulation results that did not use scaling. These improved results occurred because with scaling, the target value is a real number that never reaches the upper or lower bound of the sigmoid function. Thus, the weight adjustment was more limited and stable when the asymptotic region was avoided.

In this paradigm, the network parameters are updated by an on-line algorithm and in a discrete manner, and the input signals are differentiable (f is differentiable too) and sampled at a Nyquist rate, i.e. the sampling time step $r = \frac{1}{2f_{max}}$, where f_{max} is the maximum frequency of all input channels.

The weights and time-delays are updated step by step proportional to the opposite direction of the error gradient respectively:

$$\Delta w_{jik,h} \equiv -\eta_1 \frac{\partial E(t_n)}{\partial w_{jik,h}} \tag{5}$$

$$\Delta \tau_{jik,h} \equiv -\eta_2 \frac{\partial E(t_n)}{\partial \tau_{jik,h}} \tag{6}$$

where η_1 and η_2 are the learning rates.

The derivation of this learning algorithm was addressed explicitly in [14, 13], and is covered in the *Appendix*. We summarize the learning rules as follows.

$$\Delta w_{jik,h-1} = \eta_1 \delta_{j,h}(t_n) a_{i,h-1}(t_n - \tau_{jik,h-1}) \tag{7}$$

$$\Delta \tau_{jik,h-1} = \eta_2 \rho_{j,h}(t_n) w_{jik,h-1} a'_{i,h-1}(t_n - \tau_{jik,h-1}) \tag{8}$$

[1] normalized mean square error: $NMSE = \frac{E[(x(t)-\hat{x}(t))^2]}{E[(x(t)-E[x(t)])^2]}$, where $x(t)$ is the original signal value and $\hat{x}(t)$ is the network output value

where

$$\delta_{j,h}(t_n) = \begin{cases} (d_j(t_n) - a_{j,h}(t_n))f'(S_{j,h}(t_n)) & \text{,if } j \text{ is an output unit} \\ (\sum_{p\in\mathcal{N}_{h+1}} \sum_{q=1}^{K_{pi,h}} \delta_{p,h+1}(t_n) w_{pjq,h}(t_n)) f'(S_{j,h}(t_n)) & \text{,if } j \text{ is a hidden unit} \end{cases} \tag{9}$$

and

$$\rho_{j,h}(t_n) = \begin{cases} -(d_j(t_n) - a_{j,h}(t_n))f'(S_{j,h}(t_n)) & \text{,if } j \text{ is an output unit} \\ -[\sum_{p\in\mathcal{N}_{h+1}} \sum_{q=1}^{K_{pi,h}} \rho_{p,h+1}(t_n) w_{pjq,h}(t_n)] f'(S_{j,h}(t_n)) & \text{,if } j \text{ is a hidden unit} \end{cases} \tag{10}$$

The time delay values $\Delta\tau_{jim,h-1}$ can cross one another during learning, and (10) can specify $\tau_{jim,h-1}$ to go below zero. We reset $\tau_{jim,h-1}$ to zero in this case. Processing units do not receive data through a fixed time window, but gather important information from various time delays which are adapted via the learning procedure. The largest time delay between two units can change during adaptation, and the interconnection that has the largest time-delay can also change. With these mechanisms, the network implements the dynamic delays along the interconnections of the *ATNN*.

3 Simulations

In this section, we demonstrate the ability of the *ATNN* to learn embedded dynamics in nonlinear sequences of states. The *ATNN* learns the transformation of given input sets to output sets in sequence.

3.1 Circular and Figure Eight Trajectory

We trained an *ATNN* to follow the "circular trajectory" and the "figure eight trajectory", as in [19] and [24]. The formation of these two trajectories are:

$$\text{circular trajectory:} \begin{cases} x(t) = A\sin\omega t \\ y(t) = A cos\omega t \end{cases} \qquad \text{figure eight:} \begin{cases} x(t) = A\sin\omega t \\ y(t) = A\sin 2\omega t \end{cases}$$

We selected $A = 0.5$, $\omega = 1$, and time step $\Delta t = 0.1$; therefore, it requires approximately 63 time steps to complete a full circle.

A three layered *ATNN* was configured with two input units (taking data from $x(t)$ and $y(t)$), five hidden units and two output units (2-5-2). There were 4 time delays for each pair of units from input layer to hidden layer, and 6 time delays from each hidden layer unit to each output layer unit. The initial synapse weights were random numbers with uniform distribution between 1 and -1. Time-delays were initiated consecutively and trained to set individual length optimally to catch the necessary or significant information to the whole system.

Instead of taking the whole circle as the training data, the network learned to generate the trajectory by consecutively taking segments of arcs where the length of the needed segment was decided by the time delays that were adjusted during learning.

The progress of learning is plotted in Figure 4, 5, and 6. Figure 4 shows the learning of the first 500 samples, the network starts at a position determined by its initial (random) weights and initial delays. The initial point is calculated by using a segment of arc as input and reading the point's cpprdinates from the output of the network. The network then learns the

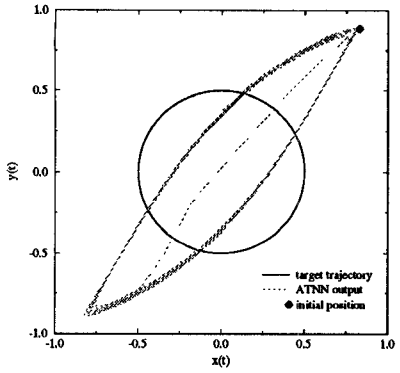

Figure 4: first 500 iteration

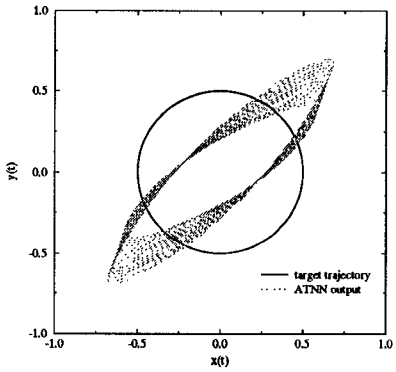

Figure 5: 1000 to 2000 iteration

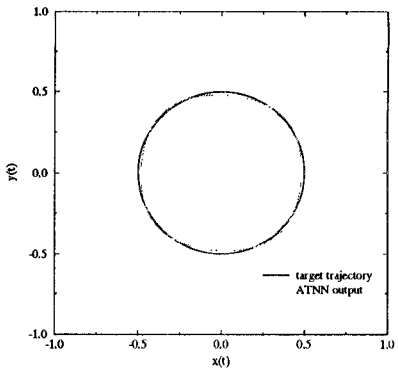

Figure 6: up to 20000 iteration

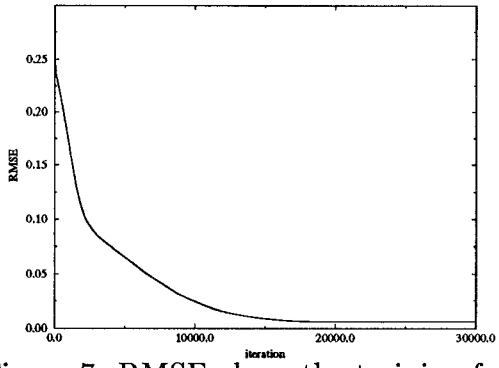

Figure 7: RMSE along the training for zero

trajectory topology (no cross-overs) very quickly. The network learns the circular trajectory with average $RMSE$ of 0.0054 after 20000 samples are presented to the network (shown in Figure 6), and the learning curve in terms of $RMSE$ is drawn in Figure 7. Observing from the resulting values of time delays (the maximum value of time-delays between input and hidden layer is 3, and the maximum value of those between hidden and output layer is 14), the network demands a total length of 18 points to reveal the next position. The length is about a quarter of a complete circle.

Next, we trained the network to learn the figure eight by using the same architecture used in the previous simulation. The learning process is shown in Figures 8, 9, and 10. Apparently, learning the figure eight is a more difficult task, since the trajectory crosses itself; however, it still learns the topology immediately. A trajectory similar to the figure eight appears

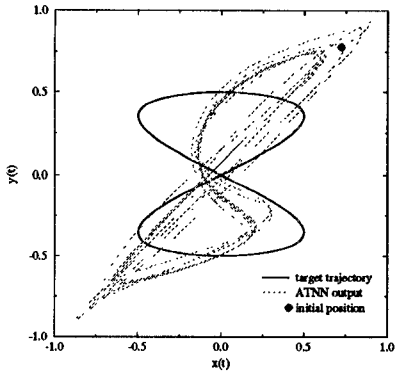

Figure 8: first 500 iteration

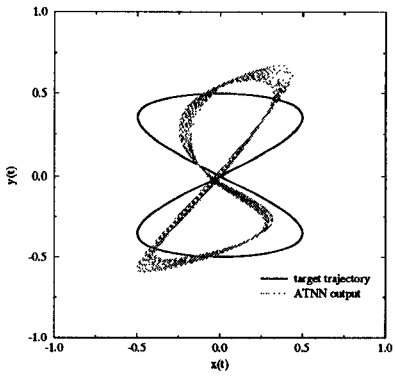

Figure 9: 1000 to 2000 iteration

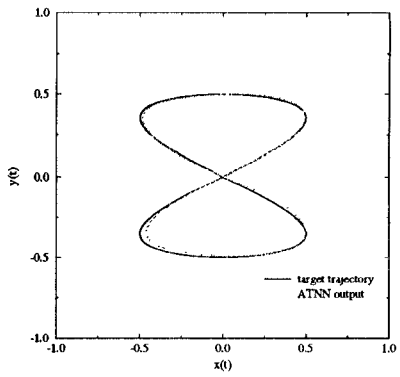

Figure 10: upto 80000 iteration

within the very beginning 500 iterations of learning, and the rest of the learning time is spent to approach the exact position. The $RMSE$ approaches to 0.0441 after 80000 training samples. The network uses information spanning 66 time steps to reveal the next position. (The maximum time-delay value in the first connection layer is 12, and the maximum time-delay value in the second connection layer is 53). The learning curve of the entire process is illustrated in Figure 11. The circle and eight are two different topologies, but the network learned both shapes within the first 10% of the learning iterations, then smoothed and adjusted the shapes during the remaining iterations. This network characteristic can be compared to Kohonen's self-organized feature map, which learns topology in the first 10% of iterations [11], but requires the remaining 90% to smooth its results to match the data effectively.

Next, we tested the network's ability to generate figures given its own outputs as input signals. Given any segment (length must be greater than 17) of the original circle starting at any point along the circular trajectory, the network generated the next new position clockwise, then completed the remaining trajectory recursively by using the new segment generated by the network.

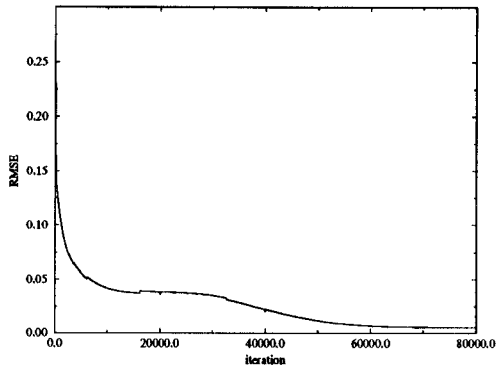

Figure 11: RMSE along the training for eight

From the experiments, the reproduced trajectories of the circle are plotted with dotted lines in Figure 12 (a)-(c). The $RMSE$ values of these three examples are 0.0113, 0.0162 and 0.0114 which are impressively low. In these runs, the network was initially given only a segment of the original circle (length of 18 in this case), but after the initial segment utilized its own past trajectory as input and then completed the whole circle. The difference between these is the starting position. These results demonstrate the capability of the network to perform predictive recognition and figure completion when only partial information of the original data is available. The trained network completes the pattern or figure no matter where the starting position is.

A similar experiment was done for the figure eight. While the performance for the figure eight reproduction is less ($RMSE$ about 0.05 for all starting positions shown in Figure 13 (a)-(c)), the accuracy improves when the network learns for a longer time. It appears that the more difficult the target trajectory is, the more error is encountered during generalization.

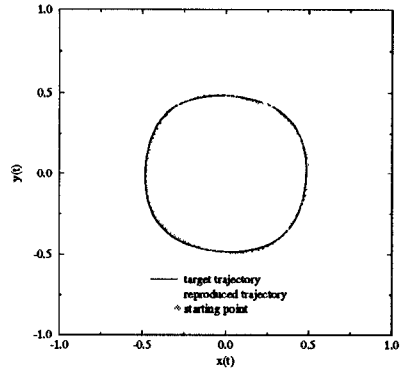

Figure 12: (a)

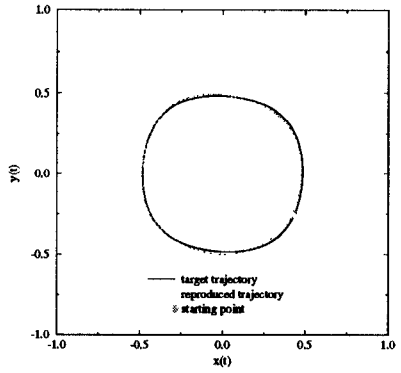

Figure 12: (b)

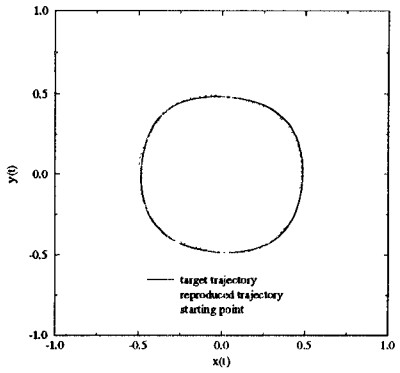

Figure 12: (c)

Figure 12: The results of circle reproduced by *ATNN* start from different initial position (a) $RMSE = 0.0113$ (b) $RMSE = 0.0162$ (c) $RMSE = 0.0114$

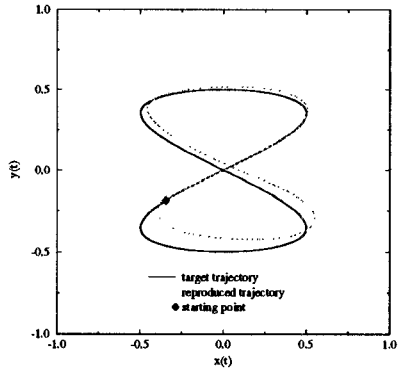

Figure 13: (a)

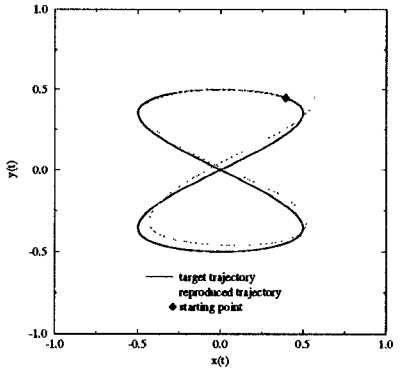

Figure 13: (b)

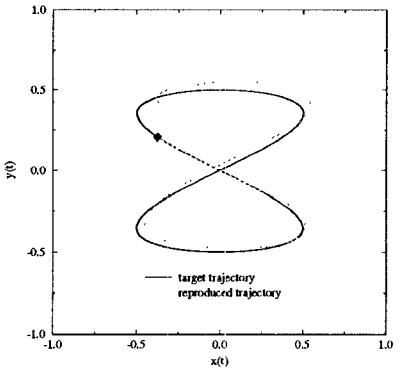

Figure 13: (c)

Figure 13: The results of figure eight reproduced by *ATNN* start from different initial position (a) $RMSE = 0.0507$ (b) $RMSE = 0.0508$ (c) $RMSE = 0.0474$

3.2 Harmonic Wave Approximation

In this experiment, the *ATNN* is trained to model the square wave approximation from the Fourier series expansion of the sum of different odd harmonics. The class of functions which can be represented by Fourier series is surprisingly large and general. The values of various series with constant terms or periodic functions can be obtained by evaluating Fourier series at specific points [12]. If the network can capture the fundamental harmonic characteristics of various waves, then neural networks will lead to a novel approach to describe the representation of signals. Five different wave approximations are used in this study: a fundamental sine wave or first harmonic, a wave with the sum of first and third harmonics ($y = sin(t) + \frac{sin(3*t)}{3}$), a wave with the sum of first to fifth harmonics ($y = sin(t) + \frac{sin(3*t)}{3} + \frac{sin(5*t)}{5}$), a signal with the combination of first to seventh harmonics (i.e. $y = sin(t) + \frac{sin(3*t)}{3} + \frac{sin(5*t)}{5} + \frac{sin(7*t)}{7}$), and one with the sum of first to ninth harmonics (i.e. $y = sin(t) + \frac{sin(3*t)}{3} + \frac{sin(5*t)}{5} + \frac{sin(7*t)}{7} + \frac{sin(9*t)}{9}$). A three layered *ATNN* with one input unit, three hidden units and one output unit was trained to identify these particular wave forms according to the adaptive learning paradigm.

Three of the simulation results are illustrated in Figure 14, 15 and 16. The original functions are plotted in solid lines, and the dotted lines denote the output of network. The system performance is so good that the solid line and dotted line almost overlap and show high accuracy. The error measure of all wave form tests are shown in Table 1. The error measure $NMSE$ is lower than 0.1% in the first three cases, while the $NMSE$ is 0.12% which is still small in the complicated case of the sum of the 1st to 9th odd harmonics. This indicates that the network has learned the various complexities with similar accuracy.

	Wave with the sum of odd harmonics				
	fundamental	1st to 3rd	1st to 5th	1st to 7th	1st to 9th
NMSE	9.2895×10^{-4}	8.8645×10^{-4}	9.2176×10^{-4}	0.0010	0.0012

Table 1: The error measure NMSE of this system output under different approximation to a square wave

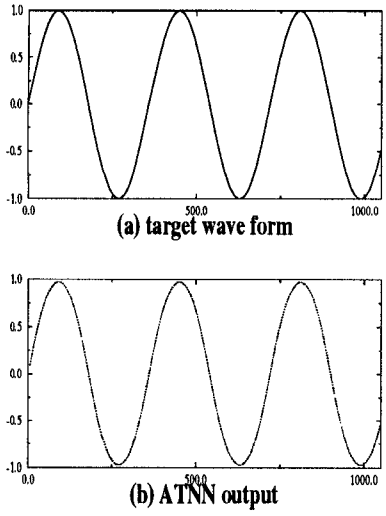

Figure 14: Simulation of identifying the fundamental wave: first harmonic

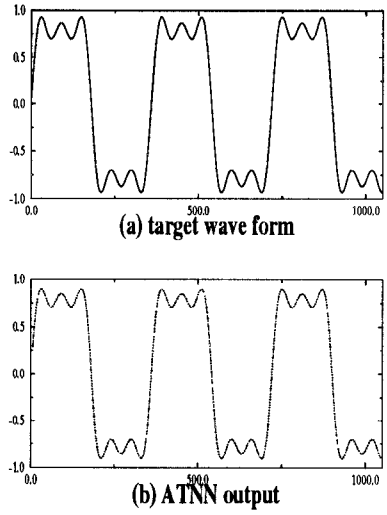

Figure 15: Simulation of identifying the wave with a sum of 1st to 5th add harmonics

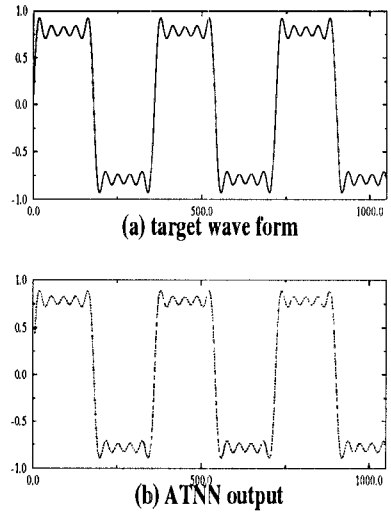

Figure 16: Simulation of identifying the wave with a sum of 1st to 9th add harmonics

3.3 Chaotic Time Series Modeling

Recently, considerable effort has been devoted to finding the embedded dynamics in an input sequence [18], or to capturing the characteristics of chaotic environments, using neural network techniques [20, 9, 22, 25, 28]. The objective of the system modeling problem is to construct a suitable embedded identification function $\hat{\mathcal{P}}$, for a given plant $\mathcal{P}$ and its input-output pair $\{u_i(\cdot), y_m(\cdot)\}$ (where i and m denotes the dimension of input and output vectors correspondingly), such that $\|\hat{y} - y\| = \|\hat{\mathcal{P}}(u) - \mathcal{P}(u)\| \leq \epsilon$ for some desired $\epsilon \geq 0$.

A chaotic time series (based on the Mackey-Glass delay differential equation) was selected for experimentation with the *ATNN*. The delay differential equation of the form

$$\frac{dx(t)}{dt} = F(x(t), x(t-\tau)) \tag{11}$$

describes systems in which a stimulus has a delayed response [8]. In this study, we used the example proposed by Mackey and Glass which describes the production of blood cells [16] as the following

$$\dot{x} = \frac{ax(t-\tau)}{1 + x^c(t-\tau)} - bx(t) \tag{12}$$

where a, b and c are parameters. This differential equation possesses many dynamic properties such as nonlinearity, limit cycle oscillations, aperiodic wave forms and other dynamic behaviors [7], and can be used as a benchmark for temporal learning.

A three layered *ATNN*, with the same complexity as used in the previous examples, was simulated to model chaotic time series of the Mackey-Glass differential equation. The network was trained to resolve values of $\dot{x}$. The training set consisted of the first three

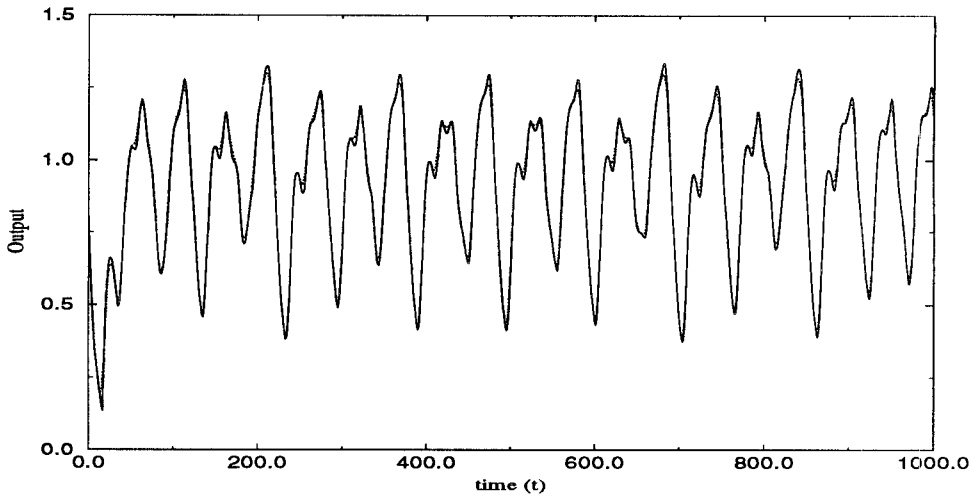

Figure 17: Simulation results of *ATNN* with Mackey-Glass equation modeling t=0 to 1000, $NMSE = 0.002$

adaptive parameters	fixed parameters	equivalence	NMSE
weights	$K_{ji,h-1} = 1$, $\tau_{jik,h-1} = 0$	*BP*	0.0126
weights	time-delays	*TDNN*	0.0097
time-delays	weights	-	0.0264
weights, time-delays	-	*ATNN*	0.003

Table 2: Experiments with different network configurations trained to predict the Mackey-Glass equation. Each table entry is the average of three runs.

hundred samples (from $t = 1$ to $t = 300$), and the testing set included ten thousand samples of the differential equation (from $t = 1$ to $t = 10000$).

The first 1000 time step outputs and last 1000 time step outputs (from 9000 to 10000) of the network is shown in Figure 17 and Figure 18. The solid line and dotted line represent the original equation value and the output of the *ATNN* respectively. As we can observe from these two figures, the network follows the original function, and catches the variation well with a *NMSE* of 0.3% up to $t = 10000$ this is better than our previous results with unscaled outputs ($NMSE = 0.84\%$) [13].

To evaluate performance, the network was trained by various choices of adaptive param-

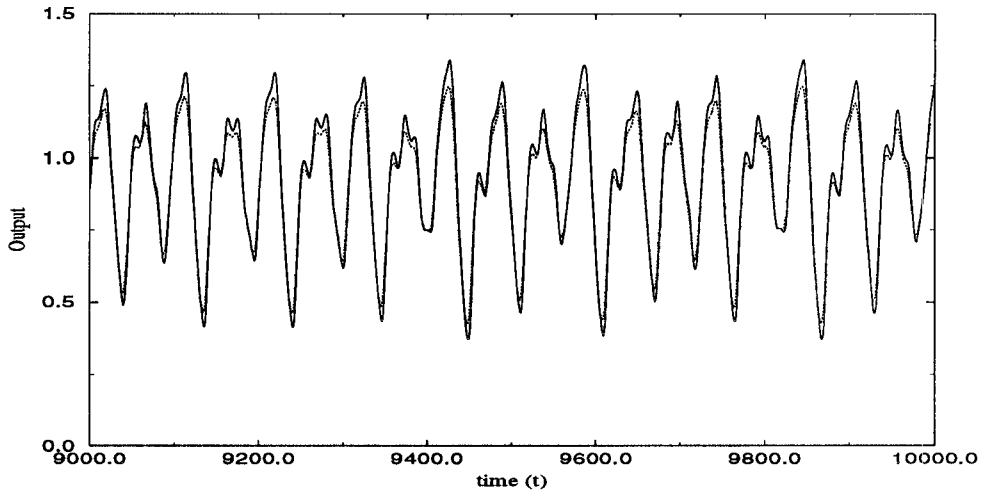

Figure 18: Simulation results of *ATNN* with Mackey-Glass equation modeling, t=9000 to 10000, $NMSE = 0.004$

eters, and the *NMSE* measures of the corresponding system outputs are shown in Table 2. When weights are adapted but no time delays are included (equivalent to *BP*), the *NMSE* was 0.0126. When time delays were included but were fixed and weights were adapted (e.g. a *TDNN*), then the *NMSE* was considerably improved at 0.0097. If time delays were adapted but weights were fixed, the worst performance was attained at $NMSE = 0.0264$. When both weights and time delays were adapted with the *ATNN* paradigm, the best performance was attained at $NMSE = 0.0084$ for the first 1000 samples. The results indicated that the proposed network improves the generalization.

4 Discussion

We have described an Adaptive Time-delay Neural Network (*ATNN*), which has a dynamic learning technique for adapting both weights and time-delays on-line, during training with temporally varying patterns. The network is an efficient and flexible model that addresses learning situations where inputs arrive over a period of time and outputs are produced over

a period of time, thus the network has spatiotemporal patterns as both inputs and outputs. The *ATNN* was trained to learn and identify two-dimensional sequences (trajectories) and one-dimensional sequences (time series prediction). The *ATNN* can be applied to a wide variety of domains, including the recognition of movements, trajectory generation, speech analysis, and signal processing.

The proposed *ATNN* is a generalization of the Error-Back Propagation neural network (*BP*). *TDNN* and *BP* can be viewed as special cases of the *ATNN* if certain parameters of *ATNN*'s paradigm are fixed.

Case 1: If we fixed the time-delay $\tau_{jik,h-1}$ in Equation (1) and applied weight learning (Equation (7)) without updating the time-delay variables, it would become a Time-Delay Neural Network.

Case 2: The Error-Backpropagation Network is also a subset case of the *ATNN*. If we set $K_{ji,h-1} = 1$ and $\tau_{jik,h-1} = 0$, then Equation (1) will become:

$$S^{\mu}_{j,h} = \sum_{i \in \mathcal{N}_{h-1}} w_{jik,h-1} \cdot a^{\mu}_{i,h-1} \tag{13}$$

This equation above is exactly in the same form of Error-Backpropagation's net input rule. If we applied gradient descent learning and update weights, the *ATNN* turns out to be the Error-Backpropagation Network in this restricted case.

Neural networks with time delays are inspired from biological models of nerve signal transmissions and processing. Biological studies have shown that varying time delays do occur along axons due to different conduction times and differing lengths of axonal fibers,

leading to variable delays on inputs to target neurons [4]. In addition, postsynaptic potentials occurring in the dendrites and cell bodies exhibit temporal dependencies that may be varied depending on biochemical substrates and dendritic microstructure [3, 21]

We have applied the *ATNN* to three examples which possess spatiotemporal complexity, with temporal sequences of patterns and in one case with embedded chaos. The simulation results show that the *ATNN* learns the topology of a circular trajectory and figure eight within 500 on-line training iterations, and reproduces the circular trajectory itself dynamically with very high accuracy ($RMSE \leq 0.0162$). Other results show that the $ATNN$ successfully models harmonic waves and does accurate time series predictions of a chaotic sequence. These applications show that the $ATNN$ is a powerful tool and learning technique for spatiotemporal data and for any applications that involve time varying signals or patterns.

Appendix: Time-Delay Learning Derivation

We define the following notation:

$L \equiv$ the number of layers in the network.

$\mathcal{N}_h \equiv$ the set of nodes $\{1, 2, ..., |\mathcal{N}_h|\}$ of layer h.

$\tau_{jik,h-1} \equiv$ the time-delay of the kth connection to node j of layer h from node i of layer $h-1$

$K_{ji,h-1} \equiv$ the total number of connections to node j (layer h) from node i of layer $h-1$

$\mathcal{T}_{ji,h-1} \equiv$ the set of delays on connections to node j (layer h) from node i of layer $h-1$, i.e. $\mathcal{T}_{ji,h-1} = \{\tau_{ji1,h-1}, \tau_{ji2,h-1}, ..., \tau_{jim,h-1}\}$, where $m = K_{ji,h-1}$

$a^{\mu}_{i,0}(t) \equiv$ the ith channel of the input training pattern μ at time t

$t_n \equiv$ the nth sampling time, where r is a single time step (e.g., $t_n = nr$)

$w_{jik,h-1} \equiv$ the synapse weight of the kth connection to node j from node i of layer $h-1$, and $k = 1, 2, ..., K_{ji,h-1}$

The activation value of node j of layer h when input pattern μ is present at time t_n is defined in Equation (2):

$$a^{\mu}_{j,h}(t_n) = \begin{cases} f_j(S^{\mu}_{j,h}(t_n)) & \text{if } h \geq 2 \\ a^{\mu}_{j,0}(t_n) & \text{if } h = 1 \end{cases}$$

where

$$S_{j,h}^{\mu}(t_n) = \sum_{i \in \mathcal{N}_{h-1}} \sum_{k=1}^{K_{ji,h-1}} w_{jik,h-1} \cdot a_{i,h-1}^{\mu}(t_n - \tau_{jik,h-1}) \tag{14}$$

$$f_j(x) = \frac{\beta_j}{1 + e^{-\alpha_j x}} - \gamma_j \tag{15}$$

where α_j, β_j and γ_j are real numbers which have been discussed in Section 2.1.

The adaptation of the delays and weights are derived based on the gradient descent method to minimize the error measure E during training. The training set consists of a set of spatiotemporal patterns and target outputs over time. An instantaneous error measure is defined as MSE (shown in Equation (4)):

$$E(t_n) = \frac{1}{2} \sum_{j \in \mathcal{N}_L} (d_j(t_n) - a_j(t_n))^2 \tag{16}$$

The time-delay is modified step by step proportional to the opposite direction of the error gradient with respect to this delay. The updating rule is therefore:

$$\Delta \tau_{jik,h-1} \equiv -\eta_1 \frac{\partial E(t_n)}{\partial \tau_{jik,h-1}} \tag{17}$$

where η_1 is the learning rate.

By the chain rule

$$\frac{\partial E(t_n)}{\partial \tau_{jik,h-1}} = \frac{\partial E(t_n)}{\partial S_{j,h}} \frac{\partial S_{j,h}(t_n)}{\partial \tau_{jik,h-1}} \tag{18}$$

The second factor of Equation (18) can be expressed as

$$\frac{\partial S_{j,h}(t_n)}{\partial \tau_{jik,h-1}} = \frac{\partial}{\partial \tau_{jik,h-1}} \sum_{p \in \mathcal{N}_{h-1}} \sum_{q=1}^{K_{ji,h-1}} w_{jpq,h-1} a_{p,h-1}(t_n - \tau_{jpq,h-1})$$

$$= -w_{jik,h-1} a'_{i,h-1}(t_n - \tau_{jik,h-1}) \tag{19}$$

We define

$$\rho_{j,h}(t_n) = \frac{\partial E(t_n)}{\partial S_{j,h}} \tag{20}$$

Substitute Equation (19) and (20) into Equation (18), we obtain

$$\frac{\partial E(t_n)}{\partial \tau_{jik,h-1}} = -\rho_{j,h}(t_n) w_{jik,h-1} a'_{i,h-1}(t_n - \tau_{jik,h-1}) \tag{21}$$

$$\stackrel{(17)}{\Longrightarrow} \quad \Delta\tau_{jik,h-1} = \eta_1 \rho_{j,h}(t_n) w_{jik,h-1} a'_{i,h-1}(t_n - \tau_{jik,h-1}) \tag{22}$$

To derive $\rho_{j,h}(t_n)$, we need to apply chain rule and consider two cases:

$$\begin{aligned} \rho_{j,h}(t_n) &= \frac{\partial E(t_n)}{\partial S_{j,h}} \\ &= \frac{\partial E(t_n)}{\partial a_{j,h}} \frac{\partial a_{j,h}(t_n)}{\partial S_{j,h}} \\ &= \frac{\partial E(t_n)}{\partial a_{j,h}} f'(S_{j,h}(t_n)) \end{aligned} \tag{23}$$

To find $\frac{\partial E(t_n)}{\partial a_{j,h}}$, we consider the following two cases:

1. If j is an output unit:

$$\frac{\partial E(t_n)}{\partial a_{j,h}} = -(d_j(t_n) - a_{j,h}(t_n)) \tag{24}$$

$$\stackrel{(23)}{\Longrightarrow} \quad \rho_{j,h}(t_n) = -(d_j(t_n) - a_{j,h}(t_n))f'(S_{j,h}(t_n)) \tag{25}$$

2. If j is a hidden unit:

$$\begin{aligned}
\frac{\partial E(t_n)}{\partial a_{j,h}} &= \sum_{p\in\mathcal{N}_{h+1}} \frac{\partial E(t_n)}{\partial S_{p,h+1}} \frac{\partial S_{p,h+1}(t_n)}{\partial a_{j,h}} \\
&= \sum_{p\in\mathcal{N}_{h+1}} \frac{\partial E(t_n)}{\partial S_{p,h+1}} \frac{\partial}{\partial a_{j,h}} \Big(\sum_{i\in\mathcal{N}_h} \sum_{q=1}^{K_{pi,h}} w_{piq,h} a_{i,h}(t_n - \tau_{piq}) \Big) \\
&= -\sum_{p\in\mathcal{N}_{h+1}} \rho_{p,h+1}(t_n) \Big(\sum_{q=1}^{K_{pi,h}} w_{pjq,h}(t_n) \Big)
\end{aligned} \tag{26}$$

$$\stackrel{(23)}{\Longrightarrow} \quad \rho_{j,h}(t_n) = -[\textstyle\sum_{p\in\mathcal{N}_{h+1}} \sum_{q=1}^{K_{pi,h}} \rho_{p,h+1}(t_n) w_{pjq,h}(t_n)] f'(S_{j,h}(t_n)) \tag{27}$$

We have now found ρ.

From Equation (22) it remains to find $a'_{i,h-1}(t_n - \tau_{jik,h-1})$. The value of $a'_{i,h-1}(t_n - \tau_{jik,h-1})$ can be approximated as following: From the elementary calculus we know that if the function $f(x)$ is differentiable, then

$$f(x_0 + h) - f(x_0) = h \frac{df}{dx} \mid_{x=x_*} \tag{28}$$

for some point x_* such that $x_0 \leq x_* \leq x_0 + h$. The derivative is evaluated at a point x_* (between x_0 and $x_0 + h$), and by the Mean Value theorem of differential calculus, x_* always

exists. Accordingly, we get

$$a'_{i,h-1}(t_n - \tau_{jik,h-1}) \approx \begin{cases} \frac{a(t_n)-a(t_{n-1})}{r} & \text{if } \tau_{jik,h-1} = 0 \\ \frac{a(t_{k+1})-a(t_{k-1})}{2r} & \text{if } t_n - \tau_{jik,h-1} = t_k,\ \tau_{jik,h-1} \neq 0 \end{cases}$$

where $r = t_k - t_{k-1}, k \in \{0, 1, ..., n\}$.

The time-delay learning rule is summarized as follows:

$$\Delta\tau_{jik,h-1} = \eta_1 \rho_{j,h}(t_n) w_{jik,h-1} a'_{i,h-1}(t_n - \tau_{jik,h-1}) \tag{29}$$

where

$$\rho_{j,h}(t_n) = \begin{cases} -(d_j(t_n) - a_{j,h}(t_n)) f'(S_{j,h}(t_n)) & \text{,if } j \text{ is an output unit} \\ -[\sum_{p\in\mathcal{N}_{h+1}} \sum_{q\in\mathcal{T}_{pj,h}} \rho_{p,h+1}(t_n) w_{pjq,h}(t_n)] f'(S_{j,h}(t_n)) & \text{,if } j \text{ is a hidden unit} \end{cases}$$

Similarly, the learning rule for weights can also be obtained in the same manner and is summarized as following:

$$\Delta w_{jik,h-1} = \eta \delta_{j,h}(t_n) a_{i,h-1}(t_n - \tau_{jik,h-1}) \tag{30}$$

where

$$\delta_{j,h}(t_n) = \begin{cases} (d_j(t_n) - a_{j,h}(t_n)) f'(S_{j,h}(t_n)) & \text{,if } j \text{ is an output unit} \\ (\sum_{p\in\mathcal{N}_{h+1}} \sum_{q\in\mathcal{T}_{pj,h}} \delta_{p,h+1}(t_n) w_{pjq,h}(t_n)) f'(S_{j,h}(t_n)) & \text{,if } j \text{ is a hidden unit} \end{cases}$$

References

[1] J.A. Anderson and E. Rosenfeld, editors. *Neurocomputing: Foundations of Research.* MIT Press, Cambridge, 1988.

[2] U. Bodenhausen and A. Waibel. The tempo2 algorithm: Adjusting time-delays by supervised learning. In J. E. Moody R. P. Lippmann and D.S. Touretzky, editors, *Advances in Neural Information Processing Systems*, volume 3, pages 155–161, Denver 1991, 1991. Morgan Kaufmann, San Mateo.

[3] T. H. Bullock, R. Orkand, and A. Grinnell. *Introduction to Nervous Systems.* W. H. reeman and Company, San Francisco, 1977.

[4] C.E. Carr and M. Konishi. Axonal delay lines for time measurement in the owl's brain stem. *Proceedings of the National Academy of Sciences, USA*, 85, 1988.

[5] F. Chapeau-Blondeau and G. Chauvet. Stable, oscillatory, and chaotic regimes in the dynamics of small neural networks with delays. *Neural Networks*, 5:735–743, 1992.

[6] S. P. Day and D. S. Camporese. Continuous-time temporal back-propagation. In *International Joint Conference on Neural Networks*, volume 2, pages 95–100, Seattle, 1991. IEEE, New York.

[7] D. Farmer and J. Sidorowich. Predicting chaotic time series. *Physical Review Letters*, 59:845–848, 1987.

[8] J.D. Farmer. Chaotic attractors of an infinite-dimentional dynamical system. *Physica D*, 4:366–393, 1982.

[9] M.R. Guevara, L. Glass, M.C. Mackey, and A. Shrier. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Trans. on Systems, Man, and Cybernetics*, 13, 1983. Reprinted in [1].

[10] P. Haffner and A. Waibel. Multi-state time delay neural networks for continuous speech recognition. In S. J. Hanson J. E. Moody and R. P. Lippmann, editors, *Advances in Neural Information Processing Systems*, volume 4, pages 135–142, Denver 1992, 1992. Morgan Kaufmann, San Mateo.

[11] T. Kohonen. *Self-Organization and Associative Memory.* Springer-Verlag, Berlin, 3 edition, 1989.

[12] Erwin Kreyszig. *Advanced Engineering Mathematics.* Wiley, New York, 1979.

[13] D.-T. Lin, J. E. Dayhoff, and P. A. Ligomenides. Adaptive time-delay neural network for temporal correlation and prediction. In *SPIE Intelligent Robots and Computer Vision XI: Biological, Neural Net, and 3-D Methods*, volume 1826, pages 170–181, Boston, November, 1992.

[14] D.-T. Lin, J. E. Dayhoff, and P. A. Ligomenides. A learning algorithm for adaptive time-delays in a temporal neural network. Technical Report SRC-TR-92-59, Systems Research Center, University of Maryland, College Park, Md 20742, May 1992.

[15] D.-T. Lin, J. E. Dayhoff, and P. A. Ligomenides. Trajectory recognition with a time-delay neural network. In *International Joint Conference on Neural Networks*, volume 3, pages 197–202, Baltimore, 1992. IEEE, New York.

[16] M.C. Mackey and L. Glass. Oscillation and chaos in physiological control systems. *Science*, 197:287, 1977.

[17] J.L. McClelland, D.E. Rumelhart, and the PDP Research Group. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, volume 2. MIT Press, Cambridge, 1986.

[18] K. S. Narendra and K. Parthasarathy. Identification and control of dynamical systems using neural networks. *IEEE Trans. on Neural Networks*, 1(1):4–26, March 1991.

[19] B.A. Pearlmutter. Learning state space trajectories in recurrent neural networks. In *International Joint Conference on Neural Networks*, volume 2, pages 365–372, Washington 1989, 1989. IEEE, New York.

[20] U. Riedel, R. Kühn, and J.L. van Hemmen. Temporal sequences and chaos in neural nets. *Physical Review A*, 38:1105–1108, 1988.

[21] G.M Shepherd. *Neurobiology.* Oxford, New York, 2 edition, 1988.

[22] H. Sompolinsky, A. Crisanti, and H.J. Sommers. Chaos in random neural networks. *Physical Review Letters*, 61:259–262, 1988.

[23] D.W. Tank and J.J. Hopfield. Neural computation by concentrating information in time. *Proceedings of the National Academy of Sciences, USA*, 84:1896–1900, 1987.

[24] N. B. Toomarian and J. Barhen. Learning a trajectory using adjoint functions and teacher forcing. *Neural Networks*, 5:473–484, 1992.

[25] L. J. van der Van, P. Verschure, and P. Molenaar. A note on chaotic behavior in simple neural networks. *Neural Networks*, 3:119–122, 1990.

[26] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. Lang. Phoneme recognition: Neural networks versus hidden markov models. In *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, pages 107–110, April 1988.

[27] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. Lang. Phoneme recognition using time-delay neural networks. *IEEE Trans. on Acoust., Speech, Signal Processing*, 37:328–339, 1989.

[28] Y. Yao and W. J. Freeman. Model of biological pattern recognition with spatially chaotic dynamics. *Neural Networks*, 3:153–170, 1990.