# Theoretical Analysis of Cross-Correlation of Time-Series Signals Computed by a Time-Delayed Hebbian Associative Learning Neural Network

David Tam*

*Department of Biological Sciences, University of North Texas, Denton, Texas 76203, USA*

**Abstract:** A theoretical proof of the computational function performed by a time-delayed neural network implementing a Hebbian associative learning-rule is shown to compute the equivalent of cross-correlation of time-series functions, showing the relationship between correlation coefficients and connection-weights. The values of the computed correlation coefficients can be retrieved from the connection-weights.

**Keywords:** Time-delayed neural networks, cross-correlation function, Hebbian learning rule, associative learning, time-series signal processing.

## INTRODUCTION

Neural networks are networks of interconnecting neurons that compute specific functions when given a set of input signals. These networks can be shown to compute complex adaptive functions (including self-learning) using the variable (adaptive) internal connection-weights between neurons to compute their outputs. When specific appropriate learning-rules are used in these networks, they can be shown to compute adaptive complex functions transforming the input into output that may not be solved by traditional analytical techniques, such as self-learning. The ability to perform these unique functions by these neural networks lie in (1) the neural network architecture (nonlinear multi-layered network), (2) the learning-rules, and (3) adaptive connection-weights.

These neural networks have been used in recent years to perform parallelizable computing functions that are capable of learning using the adaptive learning-rules to update their connection-weights. The significance of the computation performed by these networks depends very much on the network architecture and learning-rules. There has been much interest in finding the mathematical relationship between these neural networks and traditional engineering analyses. For instance, the relationships between neural networks and principal component analysis (PCA) have been investigated [1], because PCA is one of the techniques used in data compression and feature extraction. The relationship between a one-layer feedforward network using a Hebbian learning-rule in an unsupervised mode to compute the PCA was initiated [2], and subsequently investigated by many others (e.g., [3-8]).

This paper focuses on extending the analysis to establish the theoretical relationship between a time-delayed Hebbian learning network and the mathematical cross-correlation function. We will show theoretically that a Hebbian network with time-delayed inputs is equivalent to computing the cross-correlation function for time-varying signals.

The neural network architecture introduced in this paper differs from most other traditional feedforward networks. In particular, we use a time-delayed neural network (TDNN) to process time-varying signals. This network takes a time-series signal as its input for processing in order to compute the cross-correlation function automatically. A similar neural network architecture has been shown to process time-series signals generated from biological neurons to extract the correlation between the firing times of neurons [9].

An analytical solution of the computation performed by such a TDNN will be given in closed form showing the relationship between the connection-weights in the network and the cross-correlation coefficients it computes. It provides a formal proof of the mathematical description of the computation performed by such a TDNN network. It can be shown that a time-delayed Hebbian neural network essentially computes the cross-correlation function by storing the correlation coefficients in its connection-weights.

Note that we will limit the discussion of this paper to the theoretical analysis only, while the implementation of this TDNN to solve specific real-world problems will be deferred to subsequent papers in full-length, such as the implication of how biological neurons may use a time-delayed Hebbian network to cross-correlate auditory signals in real-time for sound localization and frequency-tone discrimination.

## CLASSICAL NON-TIME-DELAYED HEBBIAN ASSOCIATIVE LEARNING-RULE

An associative learning-rule was first proposed by Hebb [10] as the mechanism for synaptic-weight change in a biological neural network. Hebb essentially stated that changes in synaptic (connection) weights between neurons occur when the pre-synaptic and post-synaptic neurons fire simultaneously. It is called associative learning because it makes the association between the input and output by modifying the connection-weights between them. The stronger the association, the greater the connection-weights will be.

*Address correspondence to this author at the Department of Biological Sciences, University of North Texas, Denton, Texas 76203, USA;
E-mail: dtam@unt.edu

**1874-110X/07** 

Mathematically, it states that if the input and output of a neuron are activated simultaneously, then the connection-weight for the inputs are changed. To rephrase in the current neural network terminology, it states that the weight connecting two neural elements will change if and only if both neural elements are activated at the same time; otherwise the connection-weight remains the same.

Let $x(t)$ and $y(t)$ denote the real-valued activation functions of input and output of a neuron, respectively, and if $w(t)$ denotes the real-valued connection-weight between $x(t)$ and $y(t)$, and $\Delta w(t)$ represents the weight change between successive time-steps, then the Hebbian associative learning-rule is give by:

$$\text{if } x(t) \neq 0 \text{ and } y(t) \neq 0$$
$$\text{then } \Delta w(t) \neq 0$$
$$\text{else } \Delta w(t) = 0$$

where non-zero value of the activation functions represents activation and zero value represents non-activation. More concisely, a Hebbian learning-rule that satisfies the above condition is expressed mathematically as follows:

$$\Delta w(t) = x(t)y(t) \tag{1}$$

The relationship between the input and output with respect to the associated connection-weight is shown in Fig. (**1**), and the output is given:

$$y(t) = w(t)x(t) \tag{2}$$

## MODIFIED TIME-DELAYED HEBBIAN ASSOCIATIVE LEARNING-RULE

A time-delayed neural network architecture is used to process the time-varying input signal in this neural network. This time-delayed network is similar but different from the hybrid network introduced earlier by Tam [11]. The initial input is delayed successively by a time-delay element in each input stage of the network (see Fig. **1**). Thus, the time-delay produces the modified Hebbian learning-rule such that the connection-weight will change only if the time-delayed input and current output are activated rather than if the current input and current output are activated simultaneously. In other words, the output is associated with the previous input rather than the current input.

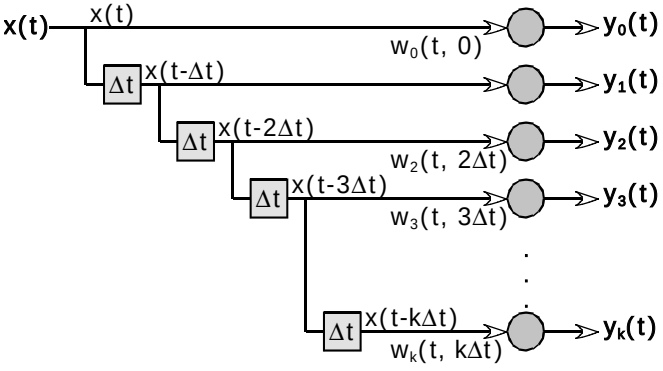


**Fig. (1).** Architecture of the time-delayed neural network showing the relationships between the time-delayed input, $x(t-k\Delta t)$, connection-weights, $w_k(t,k\Delta t)$ and their output, $y_k(t)$.

Let $x(t)$ and $y(t)$ denotes the input and output signals at time $t$, respectively, and $w(t,\tau)$ denotes the connection-weight between them with a lag-time, $\tau$, then the modified time-delayed Hebbian learning-rule is given by:

$$\text{if } x(t-\tau) \neq 0 \text{ and } y(t) \neq 0$$
$$\text{then } \Delta w(t,\tau) \neq 0$$
$$\text{else } \Delta w(t,\tau) = 0$$

where $x(t-\tau)$ denotes the input signal delayed by the lag-time, $\tau$, and $\Delta w(t,\tau)$ denotes the change in connection-weight (or the weight-change). Thus, a continuous-time time-delayed Hebbian learning-rule is given by extending Eq. 1:

$$\Delta w(t,\tau) = x(t-\tau)y(t) \tag{3}$$

For hardware implementation, we use discrete lag-times ($\tau = k\Delta t$) in integral, $k$, multiples of $\Delta t$ to delay the input signal by multiple delay-tap lines. Therefore, the time-delayed Hebbian associative learning-rule at the $k$-th delay-line is given by:

$$\Delta w_k(t,k\Delta t) = x(t-k\Delta t)y_k(t) \tag{4}$$

where $k$ is an integer constant, $\Delta w_k(t,k\Delta t)$ is the change in the $k$-th connection-weight between the time-delayed input, $x(t-k\Delta t)$, and the $k$-th output, $y_k(t)$ (see Fig. **1**).

A single time-series signal is used as the input to the network. This time-delayed input is cascaded into multiple branches as inputs to successive neurons to provide the inputs for the modified Hebbian learning-rule (Eq. 4) to update the corresponding connection-weights. The network would produce as many outputs as there are discrete time-delays. The $k$-th output of the network in Fig. (**1**) is established by:

$$y_k(t) = w_k(t,k\Delta t)x(t-k\Delta t) \tag{5}$$

Alternatively, each of the delay-tap lines in Fig. (**1**) can be considered as feeding into a pseudo-neuron as the first (pseudo) layer of the network in Fig. (**2**). This first layer can be considered as a pseudo-layer for the network because it does not perform extra computation, except for conceptualization of the equivalent neural network architecture.

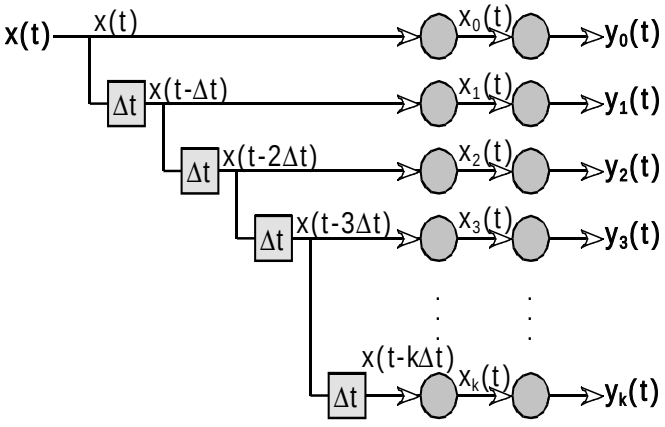


**Fig. (2).** Diagram showing how the time-delayed inputs are cascaded into forming a layer of pseudo-input neurons. This network architecture is equivalent to the diagram shown in Fig. (**1**).

The output of the $k$-th time-delayed pseudo-input neuron (in the first pseudo-layer) can be expressed in terms of the initial input signal by:

$$x_k(t) = x(t - k\Delta t) \tag{6}$$

The main reason why we represent the network in this equivalent architectural form is that now the layer of time-delayed inputs is a parallel layer rather than a cascaded sequential input layer. In other words, it transforms the single sequential time-series input into parallel inputs by the delay-lines, which allows for simultaneous parallel processing rather than sequential processing. This represents the spatio-temporal transformation of the input signal explicitly by the alternate network architecture, although they are equivalent implicitly.

Such a network would have a single sequential input, $x(t)$, branched into $(k+1)$ parallel lines by $k$ discrete delays. It will also have $k$ outputs, $y_k(t)$. The $k$-th output of the network in Fig. (**2**) is given by:

$$y_k(t) = \Delta w_k(t, k\Delta t) x_k(t) \tag{7}$$

These outputs can be further merged into a single output, $y(t)$, to form a network produces a single output signal rather than multiple outputs (see Fig. **3**). This results in the output of the network that computes the weighted-sum of all $k$ time-delayed signals mathematically:

$$\begin{aligned} y(t) &= \sum_{i=0}^{k} y_i(t) \\ &= \sum_{i=0}^{k} \Delta w_i(t, i\Delta t) x_i(t) \\ &= \sum_{i=0}^{k} \Delta w_i(t, i\Delta t) x(t - i\Delta t) \end{aligned} \tag{8}$$

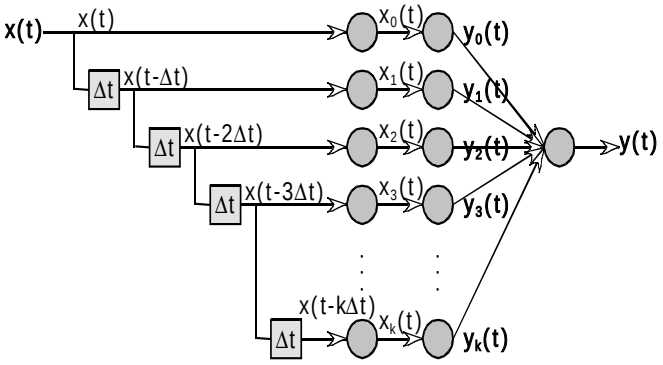


**Fig. (3).** Diagram showing the architecture of the single-input and single-output network that computes the weight-sum of time-delayed input signal.

Thus, this network architecture will provide a *single* input and a *single* output to process the time-series signal using a pseudo-input layer. This design satisfies the main objective of creating a neural network that correlates two time-series signals, $x(t)$ and $y(t)$, using a set of time-delayed Hebbian associative learning-rules.

It will be shown below that the cross-correlation coefficients are computed by the weight-sum of the time-delayed inputs by the output neuron at the $k$-th connection-weight after successive iterative training.

## COMPUTATION OF ADAPTIVE TIME-DELAYED CONNECTION-WEIGHTS

When the network is trained with $n$ iterations of the discrete time step, $\Delta t$, the resulting connection-weight is given by:

$$w_k(n\Delta t, k\Delta t) = w_k(0, k\Delta t) + \sum_{j=0}^{n} \Delta w_k(j\Delta t, k\Delta t) \tag{9}$$

for $t = n\Delta t$ and $\tau = k\Delta t$. The continuous-time time-delayed Hebbian learning-rule of Eq. (3) can be re-expressed in terms of the discrete-time step (for $t = j\Delta t$) as:

$$\Delta w_k(j\Delta t, k\Delta t) = x(j\Delta t - k\Delta t) y(j\Delta t) \tag{10}$$

The resulting connection-weights after iterating $n$ discrete time steps becomes:

$$w_k(n\Delta t, k\Delta t) = w_k(0, k\Delta t) + \sum_{j=0}^{n} x(j\Delta t - k\Delta t) y(j\Delta t) \tag{11}$$

## MATHEMATICAL CROSS-CORRELATION FUNCTION

The standard classical cross-correlation function between two continuous-time stationary time-series signals, $x(t)$ and $y(t)$, is given by:

$$r_{xy}(\tau) = \lim_{T \to \infty} \frac{1}{T} \int_0^T x(t - \tau) y(t) dt \tag{12}$$

The corresponding cross-correlation function for a discrete time step, $\Delta t$, and lag time, $\tau = k\Delta t$, is given by:

$$r_{xy}(k\Delta t) = \lim_{n \to \infty} \lim_{\Delta t \to 0} \frac{1}{n\Delta t} \sum_{j=0}^{k} x(j\Delta t, k\Delta t) y(j\Delta t) \Delta t \tag{13}$$

## RELATIONSHIP BETWEEN CROSS-CORRELATION FUNCTION AND TIME-DELAYED HEBBIAN CONNECTION-WEIGHTS

Substituting Eq. 11 into Eq. 13, the relationship between the cross-correlation function and the time-delayed Hebbian connection-weights is revealed:

$$r_{xy}(k\Delta t) = \lim_{n \to \infty} \lim_{\Delta t \to 0} \frac{1}{n\Delta t} \left[ w_k(n\Delta t, k\Delta t) - w_k(0, k\Delta t) \right] \tag{14}$$

This equation proves that theoretically the cross-correlation function is essentially computed by cumulating the connection-weight after iterations of $n$ learning time steps using the time-delayed Hebbian learning-rule introduced in this paper. This relationship shows that the cross-correlation function is computed simply by the difference

between the *initial* and *final* connection-weights of the TDNN.

The correlation coefficient, $r_{xy}(k\Delta t)$, at lag-time $\tau = k\Delta t$ can be retrieved directly from the $k$-th connection-weight of the network. This provides a theoretical closed-form solution of the relationship between the connection-weights and the correlation coefficients of a cross-correlation function.

This also shows, in contrast with most other neural networks, the computational result of the network is retrieved from the *connection-weights* rather from the *output* of the network. Furthermore, in contrast with most other neural networks, the network performs linear computation rather than nonlinear computation, since cross-correlation is essentially a linear operation. Because of the linearity, there are multiple equivalent networks that can represent the same computation, as already shown in the above analysis.

## TRAINING OF THE NETWORK

The neural network shown above illustrates how the network can self-organize to compute the cross-correlation function by adapting its connection-weights after $n$ iterations of time-steps. The remaining question is: How does the network "know" what the output, $y(t)$, should be? The answer lies in how the network is trained. During the training phase, the network is detached from producing its own output. The time-series signals to be cross-correlated, $x(t)$ and $y(t)$, are fed into the input and output of the network, respectively, so that the connection-weights can be formed internally during training.

After training, in the retrieval phase, the time-series signal $y(t)$ is detached from the network so that the network can produce its own output, $y(t)$, by computing the weighted sum of the time-delayed input based on the adapted connection-weights. The correlation coefficients are retrieved from the connection-weights directly at the $k$-th connection-weight after training.

## DISCUSSIONS

Although it can be shown by other investigators [1-8] that a time-delayed Hebbian network can compute mathematical functions such as PCA, this paper illustrated that a time-delayed Hebbian network can compute cross-correlation function too. A theoretical analysis is given to prove that the connection-weights developed after training is equivalent to the computation of the coefficients of a cross-correlation function. The TDNN processes the time-series signals, $x(t)$ and $y(t)$, in such a way that the connection-weights developed after training would produce the correlation coefficients of the cross-correlation between $x(t)$ and $y(t)$.

The main difference between the computation achieved by this network and other traditional neural networks is that the processing results are retrieved from the connection-weights rather than obtained from the output of the network. Once trained, the network can still predict the output, $y(t)$, like other neural networks by computing the weighed-sum of the time-delayed input signal at the $k$-th connection-weight.

Although this is linear network collapseable into an equivalent single-input, single-output network, it can perform multiple (parallelizable) computations of all the correlation coefficients simultaneously, which are retrievable from each of the connection-weights. Thus, this analysis bridges the conceptual framework between traditional engineering technology in cross-correlation and the novel technology implemented by a time-delayed neural network.

Furthermore, since the cross-correlation function is computed using a neural network, such computation can be implemented in hardware to process signals in real-time. Hardware implementation of this network can provide high-speed processing of time-series signals when cross-correlation computation is required. Demonstration of the applications of this time-delayed Hebbian network for processing time-series signals, such as auditory signals for sound-localization and frequency-tone discrimination will be given in a subsequent paper in further details.

## REFERENCES

[1] E. Oja, "Principal components, minor components, and linear neural networks", *Neural Networks*, vol. 5, pp. 927-936, Nov-Dec 1992.

[2] E. Oja, "A simplified neuron model as a principal components analyzer", *J. Math. Biol.*, vol. 15, pp. 267-273, Nov 1982.

[3] Y. Chauvin, "Principal component analysis by gradient descent on a constrained linear Hebbian cell", *Proc. IJCNN, Washington, DC*, vol. I, pp. 373-380, June 21-24, 1989.

[4] A. Krogh, and J. Hertz, "Hebbian learning of principal components", In: *Parallel Processing in Neural Systems and Computers* (R. Eckmiller, G. Hartmann and G. Hauske, eds.) Elsevier, Amsterdam, pp. 183-186, 1990.

[5] S. Kung, and K. Diamantras. "A neural network learning algorithms for adaptive principal component extraction (APEX)", *Proc. ICASSP-90*, Albuquerque, NM, pp. 861-864, April 1990.

[6] R. Linsker, "Self-organization in a perceptual network", *Computer*, vol. 21, pp. 105-117, Feb 1988.

[7] E. Oja, H. Ogawa, and J. Wangviwattana, "Learning in non-linear constrained Hebbian networks", In: *Artificial Neural Networks*. (T. Kohonen, K. Mäkisara, O. Simula, and J. Kangas, eds.) North-Holland, Amsterdam, pp. 385-390, 1991.

[8] J. A. Sirat, "A fast neural algorithm for principal component analysis and singular value decomposition", *Int. J. Neural Sys.*, vol. 2, pp. 147-155, 1991.

[9] D. C. Tam, "Computation of cross-correlation function by a time-delayed neural network", In: *Intelligent Engineering Systems through Artificial Neural Networks*. (Cihan H. Dagli, Laura I. Burke, Benito R. Fernández, Joydeep Ghosh, eds.), American Society of Mechanical Engineers Press, New York, NY, vol. 3, pp. 51-55, 1993.

[10] D. O. Hebb, *The organization of behavior*. New York: Wiley, 1949.

[11] D. C. Tam, "A hybrid time-shifted neural network for analyzing biological neuronal spike trains", In: *Progress in Neural Networks*, (O. Omidvar, ed.) Ablex Publishing Corporation: Norwood, New Jersey, vol. 2, pp.129-146, 1994.

Received: July 10, 2007 Revised: July 20, 2007 Accepted: July 30, 2007