

GETnet: A General Framework for Evolutionary Temporal Neural Networks

Reza Derakhshani

Department of Computer Science and Electrical Engineering
University of Missouri, Kansas City
Kansas City, MO 64110-2499
E-mail: reza@umkc.edu

Abstract— Among the more challenging problems in the design of temporal neural networks are the incorporation of short and long-term memories and the choice of network topology. Delayed copies of network signals can form short-term memory (STM), whereas feedback loops can constitute long-term memories (LTM). This paper introduces a new general evolutionary temporal neural network framework (GETnet) for the automated design of neural networks with distributed STM and LTM. GETnet is a step towards the realization of general intelligent systems that can be applied to a broad range of problems. GETnet utilizes nonlinear moving average and autoregressive nodes and sub-circuits that are trained by enhanced gradient descent and evolutionary search in architecture, synaptic delay, and synaptic weight spaces. The ability to evolve arbitrary time-delay connections enables GETnet to find novel answers to classification and system identification tasks. A new temporal minimum description length policy ensures creation of fast and compact networks with improved generalization capabilities. Simulations using Mackey-Glass time series are presented to demonstrate the above stated capabilities of GETnet.

I. INTRODUCTION

According to Asim Roy [1], the need for human experts to constantly intervene in the design and training process of a neural network, which he dubs as the “baby sitting” problem, has degraded artificial neural networks to “just another way of solving a problem”. He also mentions that the most significant and currently absent resemblance of the artificial neural networks to their biological counterparts should be automatic learning. However, this automation involves addressing issues that are currently considered open-ended. For instance, it is known that the capabilities of artificial neural networks depend on their architecture [2], and thus they need human experts in their trial and error design loop to customize each solution to the problem at hand. This issue becomes more exasperating when even the experts do not readily know what type of neural network system to use.

Addressing this problem is more important for temporal neural networks. Temporal processing prevails in nature. Living organisms model and analyze the external world through the information they receive from their sensory inputs as a stream of multidimensional temporal signals. In biological brains, the temporal association of synaptic inputs activates cellular mechanisms that underlie such diverse processes as learning, memory, and coincidence detection for sound localization. Temporal aspects are realized in biological neural assemblies through repeating units of cellular architecture. Such structures are most easily recognized in cortical territories and tapped delays via branches of axons traversing the entire structure [3], [4], [5], [6]. In design of artificial temporal neural networks, the size of the feature space for time signals usually cannot be determined analytically. For instance, in the case of STMs implemented with input delay lines (e.g. tapped delay line neural networks [7]), what should be the depth of the delay line? The same problem exists for implementation of LTM structures such as Gamma memories [8]. However, nature has found answers to the above-mentioned problems through evolution. Biological evidence supports the role of genetics in both anatomy and behavior of the brain. It has been known that learning and memory are related to synaptic architecture and transmission strength [9], [10], [11], [12]. Genes seem to have a direct role in brain architecture and its learning and memory functions. Studies on artificially mutated *Drosophila* show the genetic source of functional components of learning and memory [13], [14]. Some of the mutants with altered learning and memory capabilities show no sign of anatomical abnormalities in their nervous system, while some other display obvious physical deformations [15], [16]. It has also been shown that synaptic development in *Drosophila* shares features with higher mammals [17], [18]. Thus one can find biological justification for applying evolutionary techniques to the design of artificial neural networks.

Based on the above, a general evolutionary temporal neural network framework, GETnet, is suggested to find the topology, size, distributed memory depth and structure, and synaptic connection strengths of the sought neural network through a hybrid system of deterministic and stochastic searches in weight, delay, and architecture spaces. GETnet evolves a general class of nonlinear recurrent neural networks (RNN) with distributed delay structures. It has been shown that RNNs can represent arbitrary dynamic systems [19], [20], and they are at least as powerful as Turing machines [21]. GETnet also introduces a novel and pragmatic regularization mechanism in order to achieve minimum description length (MDL) solutions [22] to address the bias-variance dilemma and achieve better generalization with smaller training data sets.

II. GETNET'S ALGORITHM

The following summarizes GETnet's algorithm. First, GETnet randomly generates a population of temporal neural networks. Each network carries a chromosome describing its structure (S1-S3) and evolutionary behavior (E1-E5). These individuals go through a cycle of hybrid training to achieve the fittest solution. These genetic descriptors and their adaptation are described below.

A. Individuals' Structure

The structure of each network is described by direct genetic encoding under the following three objects: 1- connection matrix (S1), 2- connection branch weights (S2), and 3- connection branch delays (S3). Each neuron in a network is connected either to itself or to other neurons with single or multiple branches. Each connection branch has a specific weight and delay. These connections can be either feed forward or feeding back into any node in the network. These parallel paths in each connection create scaled and delayed copies of signals traveling in network, constituting the FIR (Finite Impulse Response) filter action of the feed forward connections and the IIR (Infinite Impulse Response) properties of the feedback loops. A more detailed description of the S1-S3 is given below.

1) *Connection Matrix, S1*: This binary matrix represents a digraph describing the node-to-node connections. Column index of each nonzero element corresponds to the source node number and the row index indicates to the destination node number. This means that for a feed forward network the upper triangle and main diagonal elements of the connection matrix are zero while for a recurrent network they will have nonzero elements. Note that network connections can have one or more parallel branches, with the corresponding weight(s) and delay(s) being described by S2 and S3.

2) *Connection Branch Weights, S2*: This object is a matrix with the same dimensions as (S1). Its elements are null-

vectors when the corresponding elements in S1 are zero, and a vector of weight values for each corresponding connection branch otherwise.

3) *Connection Branch Delays, S3*: This object is similar to S2, but its elements indicate the delay associated to each corresponding connection branch.

B. Learning

GETnet utilizes a hybrid learning system. The ontogenetic component of learning is implemented by a time-constrained gradient descent and the phylogenetic component is implemented by adaptive weight and delay mutations.

1) *Ontogenetic Learning*: Scaled conjugate gradient (SCG) was chosen for the gradient descent learning because of its generality, speed, reduced memory requirements, and better performance on sharp error surface valleys [23] that may be produced by GETnet's bias towards compact solutions. The actual CPU training time is adaptively limited to favor faster, more compact networks as they will be able to go through more SCG epochs during the allotted time and thus attain better fitness scores. This race against time results in a temporal MDL that ensures faster performance of the evolved solutions on the hosting hardware by minimizing the networks' time complexity (directly) and their space complexity (indirectly).

2) *Phylogenetic Learning*: In order to avoid local optimums one can add noise to the weights acquired by SCG, which is analogous to inexact knowledge transfer from parent to offspring. It can also be shown that the effect of adding noise to weights is similar to adding noise to the target values which improves generalization and convergence [24]. For this purpose, each weight has a Gaussian standard deviation assigned to it. These weight perturbation standard deviations are stored in a weight deviation matrix (E1) as a part of an individual's chromosome. This object is similar to S2; however its elements indicate the standard deviation of the Gaussian perturbation to be applied to the corresponding weight during mutations. Note that we need to evaluate the fitness of the search neighborhood described by E1 and not just one potential lucky mutation within that span. In order to do so, we evaluate the fitness of a network by averaging the fitness scores resulting from different starting points using variations of the current weight set as determined by E1. The number of corresponding alternate weight sets (or multiple starting points) is estimated by a saturating linear function of normalized deviations in E1.

C. Evolution

1) *Mutation Methodology*: All the standard deviations in E1-E5 are subject to evolution themselves. These parameters are updated using the evolution strategies' technique [25]

$$E_i^{new} = E_i \times \exp(\tau_2 \times N_{0,1} + \tau_1 \times N_{i,0,1}),$$

$$\tau_2 = \frac{1}{\sqrt{2n}}, \tau_1 = \frac{1}{\sqrt{2}\sqrt{2n}} \quad (1)$$

Where n is the number of parameters subject to mutation. The normal random number N_i is generated afresh for each parameter within a network, whereas the normal random number N is generated once per network per generation. Parameters' mutations take place before their utilization so that the resulting fitness score will correspond to the actual values used.

Four other evolution strategy parameters are used to shape the development of the structure and temporal characteristics of the network. These strategy parameters are the pruning threshold (E2), node mutation standard deviation (E3), connection mutation standard deviation (E4), and delay mutation standard deviation (E5).

Pruning threshold deviation, E2: This is the standard deviation for the Gaussian mutation that changes the pruning threshold of a network. This process is reminiscent of synaptic pruning of over-connected young brains in humans and other vertebrates [26], reflecting the activity or energy based synaptic elimination. Pruning is performed by finding the normalized absolute value of each connection branch weight. If this relative synaptic strength falls below the network's pruning threshold, the corresponding branch will be deleted. Pruning along with the temporal MDL help the parsimony and thus generalization capabilities of the evolving network.

Node Mutation Deviation, E3: This is the standard deviation for the Gaussian mutation that changes the number of existing nodes in a given network.

Connection Mutation Deviation, E4: This is the standard deviation for the Gaussian mutation that changes the number of existing node to node connections in a network.

Delay Mutation Deviation, E5: This is the standard deviation for the Gaussian mutation that changes the number of delay lines in between the nodes.

2) *Evolution Loop:* First, GETnet randomly generates a population of temporal neural networks with both feed forward and recurrent delayed connections. The associated weights are initialized by Nguyen-Widrow method [27]. Few simple heuristics are used to ensure functionality of the networks, such as each network and its nodes should have their input(s) and output(s) connected to somewhere, and that zero-delay loops should be avoided. Each neural network is then trained using the SCG method and the given training dataset, with a time that is limited by the temporal MDL policy. The fitness of each individual (temporal neural network) in a generation is calculated as the inverse of its mean squared error on the validation data after this partial training. The fitness score is evaluated multiple times and averaged using multiple starting points within the span given by E1. The networks with higher fitness score are given a

proportionally higher chance to parent the next generation using a roulette-wheel based selection scheme. The parents are then mutated to form offspring. Mutation acts upon general evolution strategy parameters E1-E5 and the structure descriptor objects S1-S3. Crossover is not used in GETnet since it can destroy the knowledge spread throughout the network connections [28]. During the deleting mutations, chained dependencies are taken into account to calculate the overall effect of deletions and avoid the disruptive ones such as removing a network's output path if possible. These smooth mutations reduce the noise in assessment of evolving parameters. This evolution cycle repeats itself until the required precision is reached. After exiting the evolution loop, the best network is fully trained its output for the test data is produced.

III. SIMULATIONS AND RESULTS

Here we will show how GETnet can find a compact solution for the Mackey-Glass time series prediction. Mackey-Glass is used to benchmark time series processing capabilities of many neural networks [29], [30], [31], and is recommended by IEEE Neural Networks Council Standards Committee Working Group on Data Modeling Benchmarks as a reference for comparisons [32]. All the simulations were performed using MATLAB 6.5 and its neural network toolbox v4.

Problem Description: Mackey-Glass series is defined by the following differential equation [33], [34]

$$\frac{dx(t)}{dt} = \frac{0.2x(t-\tau)}{(1+x(t-\tau))^{10}} - 0.1x(t) \quad (2)$$

This series' behavior is dependent on the values of the initial condition $x(0)$ and the parameter τ . When $x(0)=1.2$ and $\tau=17$, Mackey-Glass is a chaotic, pseudo-periodic, and non-convergent time series. The proposed tasks are 6 and 36 step predictions for this series.

1) *Six-step Prediction:* The first 1500 points of a sampled Mackey-Glass series with $\tau=17$ (MG17) were used in this simulation. The sought task is a 6-step prediction. The first 1000 samples were used for training and early stopping. The last 500 points were used for testing. The data itself was

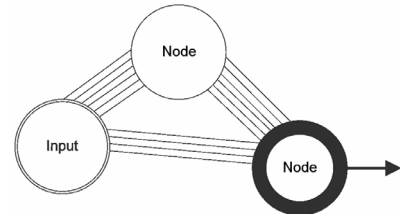


Fig. 1. Best evolved network for MG17 six-step prediction. Each line represents a delayed synaptic connection between the nodes.

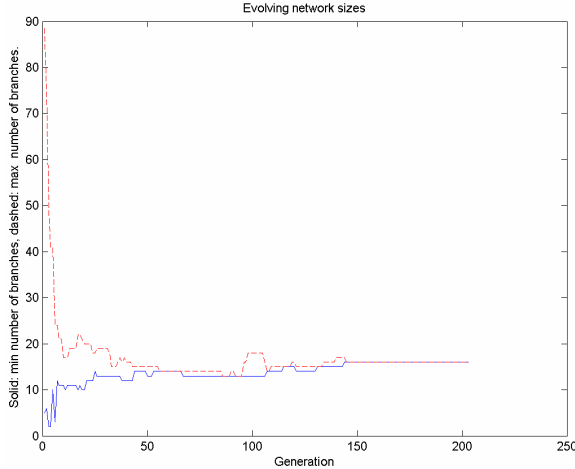


Fig. 3. The size of evolving networks, MG17 six-step prediction

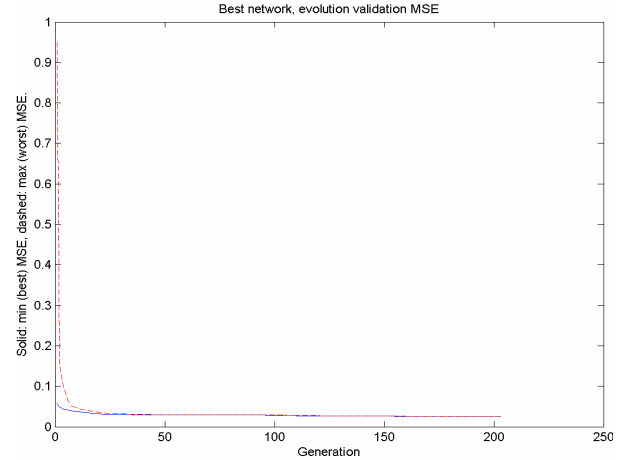


Fig. 2. The MSE of evolving networks, MG17 six-step prediction.

obtained from Eric Wan's benchmark collection of temporal data for FIRnet [35].

Using an initial population of 25 and after 203 generations, the best evolved network produced a training mean squared error (MSE) of 0.0052 and a test MSE of 0.0054. Further details are given in Table 1, where the first row represents the parameters of the ancestor of the best evolved network at first generation and the second row shows the same parameters during the last generation. Figures 1 through 5 show the results of this simulation.

2) *Thirty-six-Step Prediction*: The simulation was repeated for a 36-step Mackey-Glass series prediction task. Using an initial population of 25 and after 175 generations, the best evolved network produced a training MSE of 0.0077 and test MSE of 0.0114. The details of the evolution are given in Table 2, where the first row represents the parameters of the ancestor of the best evolved network at first generation and the second row shows the same parameters during the last generation.

Comparison

Three similar networks in terms of size and structure were used for comparison tests. A closely comparable standard temporal architecture to the 6-step prediction task's three-node evolved network is a three-node, two-layer focused time delay neural network (TDNN). Three such networks were evaluated. Each network was a two layer focused TDNN with three different 11-tap input delay lines with taps placed at steps [0, 1, 2 ... 10], [0, 5, 10 ... 50] and [0, 10, 20 ... 100]. The 11-stage input delay line was selected to match the size of a similar structure in the best-evolved network. The same training and test sets along with the SCG training algorithm were used.

For the first focused TDNN, the best result after several initializations was

Train MSE = 0.0230

Test MSE = 0.0240

For the second focused TDNN, the best result after several initializations was

MSE train = 0.0482

MSE test = 0.0489

And for the third focused TDNN, the best result after several initializations was

MSE train = 0.0687

MSE test = 0.0723

Comparing to GETnet's 0.0052 training MSE and 0.0054 test MSE, we find them to be 4 to 13 times better than the MSEs of these similar focused time delay neural networks. A similar comparison test was performed for the 36-step prediction simulation result, and GETnet's results were found to have a training MSE more than 4 to 9 times and test MSE more than 4 to 7 times better than that of the similar focused time delay neural networks.

IV. CONCLUSIONS

We saw how GETnet can evolve solutions that are compact and fast in terms of actual execution time on the hosting hardware. Some important observations can be made from the MG17 prediction tasks. First, there are almost no differences between the performance of the network on training and test data sets. This generalization capability can be credited to the minimization of the model variance through the novel temporal MDL as well as pruning. Figure 3 shows this strong-

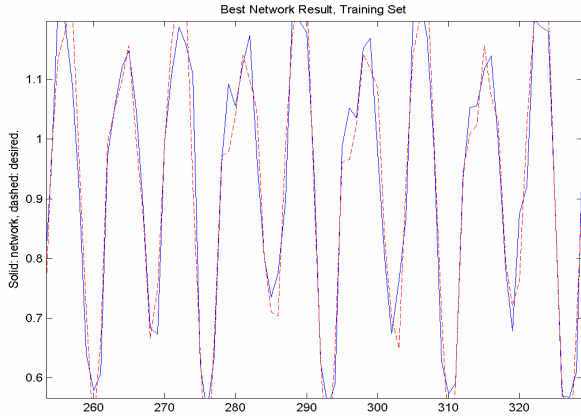


Fig. 4 A sample of the best network's training data performance.

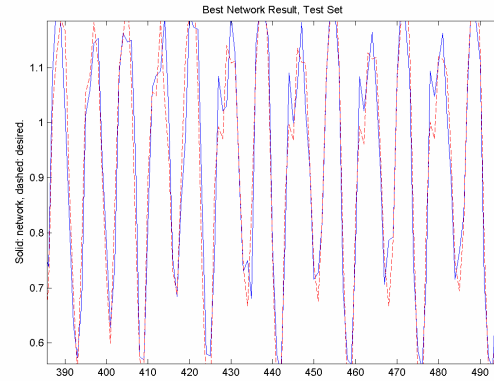


Fig. 5 A sample of the best network's test data performance.

TABLE I
6-STEP MG17 PREDICTION TASK

Branches	Nodes	$\bar{E}1$	E2	E3	E4	E5	Time
22	3	0.0413	0.0026	0.0076	0.1834	0.6959	22.543 s
16	3	0.0014	0.0005307	0.0101	0.0001917	0.0012	6.269 s

TABLE II
36-STEP MG17 PREDICTION TASK

Branches	Nodes	$\bar{E}1$	E2	E3	E4	E5	Time
33	5	0.0465	0.0025	0.1749	0.0243	0.6778	53.778 s
30	2	0.0016	0.0001416	0.1396	0.0702	0.00061462	4.401 s

tendency of GETnet towards parsimony for the first simulation. This property of GETnet is especially important when the training data points are scarce, which is usually the case for the real world problems. Also note that through the course of evolution, the reduction of network size in terms of number of branches is 1.375 for the first simulation while the speedup in training time is about 3.6. For the second simulation, the size reduction in terms of number of branches is 1.1 (or 2.5 times if number of nodes are considered), while the speedup in training time is about 12.2. This is what we desired by choosing a selection pressure that is related to the actual network time complexity.

Comparing the best evolved networks in last vs. first generations, we observe that the ranges of most mutation standard deviations have reduced several folds. Especially, the mean of weight perturbation standard deviations, $\bar{E}1$, has reduced almost 30 times in both tests (we used the mean since $E1$ is a matrix). This effect is somewhat comparable to simulated annealing. It is interesting to note that this behavior was not dictated to the network, but it emerged through the evolutionary process.

GETnet is more flexible and comprehensive than the comparable temporal neural network paradigms such as TDNN [36], FIRnet [37], Elman [38], Jordan [39], PRNN [40], and NARMA [41]. In contrast to GETnet, all the mentioned architectures need human expertise to determine their memory and network arrangements as well as the other

learning parameters (the baby-sitting problem). They also lack an automated mechanism to determine the minimum required network size, an essential issue for generalization. Furthermore, none of the above paradigms offer an arbitrary distributed memory structure comprised of recurrent nodes and sub-circuits as well as delay lines of variable depths.

In terms of implementation, GETnet is well suited for parallel processing. The execution time of GETnet's evolution phase reduces linearly with the number of participating parallel computing nodes. The network chromosomes and small synchronization messages that need to be shared between the parallel processing nodes are typically very small and thus the inter-node communication overheads are negligible. The increasing availability of affordable computer clusters makes this feature of GETnet especially attractive.

REFERENCES

- [1] Roy A. (2003) "Neural Networks: How do we make a widely used technology out of it?" IEEE NNS Connections, Vol. 1, No. 2, pp. 8-12.
- [2] Lawrence S., Giles C. L., and Tsoi A. C. (1996) "What size neural network gives optimal generalization? Convergence properties of backpropagation." Technical Report UMIACS-TR-96-22 and CS-TR-3617, University of Maryland, College Park.
- [3] Medina J. and Mauk M. (2000) "Computer Simulation of Cerebellar Information processing," Nature Neuroscience Supplement, Vol. 3, November, pp. 1205-1211.
- [4] Voogd J. and Glickstein M. (1998), "The Anatomy of the Cerebellum," Trends Neuroscience 21:370-375.
- [5] Eccles J. C., Ito M., and Szentágothai, J. (1967), The Cerebellum as a Neuronal Machine, Springer, Berlin, New York.
- [6] Ito, M. (1984) The Cerebellum and Neural Control, Raven, New York.
- [7] vanVeelen, M., Nijhuis, J.A.G., and Spaanenburg, L. (2000), "Neural network approaches to capture temporal information," pp. 361 - 371, in: (D.M. Dubois) Computing Anticipatory Systems, Proceedings CASYS'99, AIP-517, American Institute of Physics, ISBN 1-56396-933-5.
- [8] Principe, J. (1994) "An Analysis of the Gamma Memory in Dynamic Neural Networks." IEEE Trans. on Neural Networks, 5 (2), 331-337.
- [9] Bailey C. H. and Kandel E. R. (1993) "Structural Changes Accompanying Memory Storage," Ann Rev Physiol 55:397-426.
- [10] Genisman, Y., deToledo Morrell F., Heller R. E., Rossi M., and Parshall, R. F. (1993) "Structural Synaptic Correlate to Long-term

- Potentiation: Formation of Axospinous Synapses with Multiple, Completely Partitioned Transmission Zones," *Hippocampus* 3 (4), 435-445.
- [11] Nicoll, R. A., and Malenka, R. C. (1995) "Contrasting Two Forms of LTP in the Hippocampus," *Nature* 377, 115-118.
- [12] Villa, A., Tsien, R. W., and Scheller, R. H. (1995) "Presynaptic Component of Long-term Potentiation Visualized at Individual Hippocampal Synapses," *Science* 268, 1624-1628.
- [13] Feany, M. B. and Quinn, W. G. (1995) "A Neuropeptide Gene Defined by the *Drosophila* Memory Mutant *Amnesiac*," *Science* 268, 869-873.
- [14] Quinn W.G., Sziber P.P., and Booker R. (1979) "*Drosophila* Memory Mutant *Amnesiac*," *Nature* 277:212-4.
- [15] de Belle J. S., and Heisenberg, M. (1995) "Genetic, Neuroanatomical and Behavioral Analyses of the Mushroom Body Miniature Gene in *Drosophila Melanogaster*," *J Neurogenet* 10:24-30.
- [16] Bouhouche, A., and Vaysse, G. (1991) "Behavioral Habituation of the Proboscis Extension Reflex in *Drosophila Melanogaster*: Effect of the no Bridge," *J. Neurogenet.* 7, 117-128.
- [17] Broadie, K., and Bate, M. (1995) "The *Drosophila* NMJ: A Genetic Model System for Synapse Formation and Function," *Sem. Dev. Biol.* 6, 221-231.
- [18] Broadie, K. (1994) "Synaptogenesis in *Drosophila*: Coupling Genetics and Electrophysiology," *J. Physiology* 88, 123-139.
- [19] Seidl D.R. and Lorenz D., (1991) "A Structure by Which a Recurrent Neural Network can Approximate a Nonlinear Dynamic System," *Proc. Int. Joint Conf. Neural Networks*, Vol. 2, pp. 709-714.
- [20] Siegelmann H. T. and Sontag E. D., (1995) "On the Computational Power of Neural Networks," *J. Comput. Syst. Sci.*, vol. 50, no. 1, pp. 132-150.
- [21] Siegelmann, H.T., Horne, B.G., Giles, C.L. (1997) "Computational Capabilities of Recurrent NARX Neural Networks" *Systems*, *IEEE Transactions on Man and Cybernetics*, Part B, Vol.27, Issue 2, pp 208-215.
- [22] Hansen M. H. and Yu B. (2001) "Model Selection and the Principle of Minimum Description Length," *Journal of the American Statistical Association*, Vol. 96, No. 454, pp. 746-774.
- [23] Moller M. (1993) "A Scaled Conjugate Gradient Algorithm for Fast Supervised Learning," *Neural Networks*, 6:525-533.
- [24] Jim K., Giles C.L., and Horne B.G. (1996) "An Analysis of Noise in Recurrent Neural Networks: Convergence and Generalization", *IEEE Trans. Neural Networks*, Vol. 7, No. 6, pp. 1424-1439.
- [25] Schwefel H. P. (1981). *Numerical Optimization of Computer Models*, John Wiley & Sons, New York.
- [26] Chechik G., Meilijson I., and Ruppin E. (1998) "Synaptic Pruning in Development: A Computational Account." *Neural Computation*, 10(7), 1759-1777.
- [27] Nguyen D. and Widrow, B. (1990) "Improving the Learning Speed of the 2-Layer Neural Networks by Choosing Initial Values of Adapting Weights," in *Proceedings of the International Joint Conference on Neural Networks*, Vol. 3, pp. 21-26, San Diego, CA.
- [28] Yao, X. (1999) "Evolving Artificial Neural Networks," *Proceedings of IEEE*, September, 87 (9) pp. 1423-1447.
- [29] Principe J, Rathie A., and Kuo J.M. (1992) "Prediction of Chaotic Time Series with Neural Networks and the Issue of Dynamic Modeling", *International Journal of Bifurcation and Chaos*, Vol. 2, pp. 989-996.
- [30] Yao X. and Liu Y. (1997) "EPNet for chaotic time-series prediction," in *Selected Papers from the First Asia-Pacific Conference on Simulated Evolution and Learning SEAL'96*, X. Yao, J.-H. Kim, and T. Furuhashi, editors, Vol. 1285 of *Lecture Notes in Artificial Intelligence*, pp. 146-156, Springer-Verlag, Berlin.
- [31] De Falco A. et al (1998) "Optimizing Neural Networks for Time Series Prediction." *Third World Conference on Soft Computing WSC3*.
- [32] <http://neural.cs.nthu.edu.tw/jang/benchmark/>
- [33] Mackey, M. C., and Glass, L. (1977) "Oscillation and Chaos in Physiological Control Systems," *Science*: 197, 287-289.
- [34] de Menezes M. A., and dos Santos R. M. Z. (2000) "The Onset of Mackey-Glass Leukemia at the Edge of Chaos," *International Journal of Modern Physics C*, Vol. 11, No. 8.
- [35] <http://www.cse.ogi.edu/~ericwan/data.html>
- [36] Waibel, A., Hanazawa, T., Hinton, G.; Shikano, K.; Lang, K.J. (1989) "Phoneme Recognition Using Time-Delay Neural Networks," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol.37, Issue 3, pp 328-339.
- [37] Wan E., (1993) "Time Series Prediction Using a Neural Network with Embedded Tapped Delay-Lines", in *Predicting the Future and Understanding the Past*, SFI Studies in the Science of Complexity, Eds. A. Weigend , N. Gershenfeld Addison-Wesley.
- [38] Elman, J (1990) "Finding Structure in Time," *Cognitive Science*, 14, pp 179-211.
- [39] Jordan, M. (1986) "Serial order: A Parallel Distributed Processing Approach," *Institute for Cognitive Science Report 8604*. University of California, San Diego.
- [40] Haykin, S. and Liang Li (1995) "Nonlinear Adaptive Prediction of Nonstationary Signals," *IEEE Transactions on Signal Processing*, [see also *Acoustics, Speech, and Signal Processing*, *IEEE Transactions on*], Vol.43, Issue 2, pp 526-535.
- [41] Narendra, K.S. and Parthasarathy, K. (1990) "Identification and Control of Dynamical Systems Using Neural Networks" *IEEE Transactions on Neural Networks*