Chaotic Neural Networks with a "Small-World" Topology Can Achieve Pattern Recognition*

Ke Qin[1] and B. J. Oommen[2,3]

[1] University of Electronic Science and Technology of China, Chengdu, China. 611731 (E-mail: qinke@uestc.edu.cn)
[2] Carleton University, Ottawa, ON, Canada. K1S 5B6
[3] University of Agder, Postboks 509, 4898 Grimstad, Norway. (E-mail: oommen@scs.carleton.ca)

Abstract. This paper confirms the fascinating result that we can design chaotic Neural Networks (NNs) that have a "small-world" or "scale-free" topology and that these NNs can achieve *chaotic* Pattern Recognition (PR). What we imply by this is that the NN yields a strong *periodic* or *more frequent* signal when a pattern is recognized, and in between two consecutively recognized patterns, none of the trained patterns are recalled. Finally, and most importantly, if an untrained pattern is presented, the system yields a chaotic signal. The foundational NN that we employ for this is the Adachi Neural Network (AdNN). The latter is a fascinating NN which has been shown to possess chaotic properties, and to also demonstrate Associative Memory (AM) and PR, while variants of the AdNN have also been used to obtain other PR phenomena, and even blurring. The problem with the Adachi NN is that it is a fully-connected network requiring quadratic computations for the training. Our aim in this paper is to reduce the computations needed for the training *significantly.* In [1] we managed to reduce the AdNN's computational cost significantly by merely using a linear number of computations by enforcing a Maximum Spanning Tree topology and a gradient search method. However, from the perspective of a network's *structure*, very few real-life networks have a tree-shaped *linearly*-connected topology. The question we consider in this paper is whether we can reduce the degree of connections of each node to mimic the small-world or scale-free phenomena, more akin to "*real*" NNs. Simultaneously, we shall also attempt to ensure that the newly-obtained network still possesses strong PR characteristics. To achieve this, we first construct a small-world network by means of the so-called N-W model. We then address the problem of computing the weights for the new NN. This is done in such a manner that the modified small-world connection-based NN has approximately the same input-output characteristics, and thus the new weights are themselves calculated using a gradient-based algorithm. By a detailed experimental analysis, we show that the new small-world AdNN-like

*A preliminary version of this paper was presented at CHAOS'14, the 2014 Chaotic Modeling and Simulation International Conference, Lisbon, Portugal, in June 2014. The work of the first author is supported by National Natural Science Foundation of China (Grant NO. 61300093) and Fundamental Research Funds for the Central Universities in China: ZYGX2013J071.

Received: 4 February 2014 / Accepted: 7 September 2014
© 2014 CMSIM ISSN 2241-0503

network possesses PR properties for appropriate settings. As far as we know, such a small-world AdNN has not been reported, and the results given here are novel.
Keywords: Chaotic Neural Networks, Chaotic Pattern Recognition, Adachi-like Neural Networks, Small-world Networks.

1 Introduction

The goal of the field of Chaotic Pattern Recognition (PR) systems can be expressed as follows: We do not intend a chaotic PR system to report the identity of a testing pattern with a "proclamation" of the pattern's class. Rather, what we want to achieve, on one hand, is to have the chaotic PR system give a strong *periodic* or *more frequent* signal when a pattern is to be recognized. Further, between two consecutive recognized patterns, none of the trained patterns must be recalled. Finally, and most importantly, if an untrained pattern is presented, the system must give a chaotic signal. This is analogous to how the brain works. Once a pattern is recalled from a memory location, the brain is not "stuck" to it, it is also capable of recalling other Associated Memory (AM) patterns. This ability to "jump" from one memory state to another *in the absence of a stimulus* is one of the hallmarks of the brain, and *this is one phenomenon that a chaotic PR system has to emulate.*

Adachi *et al* and Calitoiu *et al* have done a lot of ground-breaking work in this area [2–4], and we have built on these results in various avenues [3–6], including that of designing a NN that can yield *ideal* chaotic PR [7]. Generally speaking, the computational burden of the original AdNN and its variants [2–4] is quadratic, rendering them to be impractical machines.

This is also true of most of the current NNs which possess a regular topology, e.g., a completely connected graph or a neighbor-coupled graph. In [1] we managed to reduce the AdNN's computational cost significantly by merely using a linear number of computations by enforcing a Maximum Spanning Tree topology and a gradient search method. In our previous paper [6], we succeeded in creating a Random-AdNN by using the E-R model. Then we computed the weights for the new network by means of gradient search. The newly obtained network was shown to still possess PR and AM properties.

All of these must be contrasted with "real" NNs which usually have irregular topology. In this paper we shall attempt to design and implement a NN (which we shall refer to as "Smallworld-AdNN") that demonstrates such *chaotic* PR properties, even though this newly-designed network, in and of itself, is a "small-world" or "scale-free" network. This is the primary contribution of this paper. Briefly stated, this is achieved by using the N-W model followed by an effective gradient search strategy, whence the computational burden can be significantly reduced. Further, as we shall show presently, the Smallworld-AdNN is almost as effective as the fully-connected AdNN with regard to its chaotic and PR characteristics.

2 Designing the Smallworld-AdNN

2.1 The Topology of the Smallworld-AdNN

To design the Smallworld-AdNN, we shall first arrive at a topology with edges connected in a small-world manner. There are many ways to generate a small-world NN, for example,by invoking the Watts-Strogts (W-S) [8] or the Newman-Watts (N-W) models [9], and we shall use the former.

The general steps of the obtaining a W-S model are as follows:

1. Arrange the neurons in a cycle, and index them from 1 to N.

2. Create a neighbor-coupled network, where each neuron is connected with $k/2$ neurons on both its sides. Thus, the degree for each neuron is k.

2. Re-connect each edge of the network with fixed probability pr. That is, for each edge, delete it with a probability pr and connect it with another randomly chosen neuron.

The reader will observe two special situations that arise from this W-S model: The new network becomes a random network if $pr = 1$ while it remains the same if $pr = 0$.

Obviously, the W-S model has the potential of causing some neurons to become isolated. In [9], Watts and his coauthor improved the W-S model by edge addition instead of deletion. Thus the second step is modified as follows: Randomly connect two unconnected neurons with a fixed probability pr. Again, one can then see that if $pr = 1$, the network becomes fully connected while it remains the same if $pr = 0$. It is worth pointing out that the W-S and N-W are essentially the same when pr is small and the number of neurons, N, is large. In this paper, we shall use the N-W model to create a small-world network. Consequently, we build the topology of the Smallworld-AdNN by invoking the following algorithm.

Algorithm 1 Topology_Smallworld-AdNN

Input: N, the number of neurons in the network, and a set of p patterns which the network has to "memorize".
Output: The topology and initial weights of the Smallworld-AdNN.
Method:

```
1: Create a neighbor-coupled graph, $\mathcal{G}$, with $N$ vertexes which is to represent the AdNN.
2: Connect two randomly chosen unconnected neurons with a fixed probability $pr$.
3: Compute the initial weights of the edges of $\mathcal{G}$, $\{w_{ij}\}$, by the following:
   $w_{ij} = \frac{1}{p}\sum_{s=1}^{p}(2x_i^s - 1)(2x_j^s - 1)$, where $x_i^s$ is the $i^{th}$ component of the $s^{th}$ stored pattern.
4: If there is no edge between vertex $i$ and $j$, then let $w_{ij} = 0$;
```

End Algorithm Topology_Smallworld-AdNN

Since the network topology has been changed and we want the Smallworld-AdNN to maximally approximate the original AdNN, we thus invoke the second step that involves the computation of the weights associated with this structure.

2.2 The Weights of the Smallworld-AdNN: Gradient Search

Since we have successfully created the structure of the Smallworld-AdNN by using the N-W model, our next task is to determine a new set of weights so as to force the Smallworld-AdNN to retain some of its PR properties, namely those corresponding to the trained patterns. We briefly explain below the process for achieving this.

The Smallworld-AdNN is defined by following equations:

$$x_i^S(t+1) = f(\eta_i^S(t+1) + \xi_i^S(t+1)), \tag{1}$$

$$\eta_i^S(t+1) = k_f \eta_i^S(t) + \sum_{e_{ij} \in \mathcal{G}} w_{ij}^{S^*} x_j^S(t), \tag{2}$$

$$\xi_i^S(t+1) = k_r \xi_i^S(t) - \alpha x_i^S(t) + a_i. \tag{3}$$

where $\{w_{ij}^{S^*}\}$, x_i^S, ξ_i^S and η_i^S are the weights, outputs, and state variables of the Smallworld-AdNN respectively, and have similar meanings to $\{w_{ij}\}$, x_i, ξ_i and η_i of the AdNN .To find the optimal values of $\{w_{ij}^{S^*}\}$, we define the square error between the original output of the AdNN and new output at the n^{th} step:

$$E_p = \frac{1}{2} \sum_{i=1}^{N} (x_i^{A,p} - x_i^{S,p}(n))^2, \tag{4}$$

where $x_i^{A,p}$ and $x_i^{S,p}$ imply the outputs of the i^{th} neuron when the p^{th} pattern is presented to the AdNN network and the Smallworld-AdNN network respectively. The overall global error is defined by $E = \sum_{p=1}^{P} E_p$ where P is the number of trained patterns. In order to adjust w_{ij}^S to obtain the least global error E, we consider the gradient, Δw_{ij}^S, and move w_{ij}^S by an amount which equals Δw_{ij}^S in the direction where the error is minimized. Formally:

$$\begin{aligned} \Delta w_{ij}^S &= -\beta \frac{\partial E}{\partial w_{ij}^S} = -\beta \frac{\partial \sum_{p=1}^{P} E_p}{\partial w_{ij}^S} = -\beta \sum_{p=1}^{P} \frac{\partial E_p}{\partial x_i^{S,p}(n)} \cdot \frac{\partial x_i^{S,p}(n)}{\partial w_{ij}^S} \\ &= \beta \sum_{p=1}^{P} (x_i^{A,p} - x_i^{S,p}(n)) \cdot \frac{1}{\varepsilon} \cdot x_i^{S,p}(n) \cdot (1 - x_i^{S,p}(n)) \cdot x_j^{S,p}(n), \end{aligned} \tag{5}$$

where β is the learning rate of the gradient search.

The formal algorithm which achieves the update is given in Algorithm 2. Its Lyapunov analysis is found in [10].

The results of a typical numerical experiment which proceeds along the above gradient search are shown in Fig. 1 and 2. Here, we have chosen the learning rate β to be 0.05 and $pr = 0.5$. Specifically, we report our experiments for three cases, i.e., when $k/2 = 4$, $k/2 = 6$ and $k/2 = 10$ respectively. If $k/2 = 4$, the average value of Δw_{ij}^S does not converge at 0, as shown in Fig. 1.

As $k/2$ increases, e.g., $k/2 = 6$, Δw_{ij}^S converges to 0, as shown in Fig. 2 (a). If $k/2$ is even larger, Δw_{ij}^S also converges to 0 but at a faster rate. This phenomenon can be easily explained: The larger the value of $k/2$, the more are

Algorithm 2 Weights_Smallworld-AdNN

Input: The number of neurons, N, a set of P patterns, and the initial weights $\{w_{ij}^S\}$ of the Smallworld-AdNN. These initial weights are $\{w_{ij}^A\}$ for the edges in the smallworld graph, and are set to *zero* otherwise. The parameters and the setting which we have used are the learning rate $\beta = 0.05$, $\varepsilon = 0.015$, $\alpha = 10, k_f = 0.2$ and $k_r = 1.02$.
Output: The weights $\{w_{ij}^{S^*}\}$ of the Smallworld-AdNN.
Method:

```
1: Compute the outputs of the Smallworld-AdNN corresponding to the $P$ trained inputs.
2: For all edges of the Smallworld-AdNN, compute $\Delta w_{ij}^S$ as per Equation (5). Otherwise, set $\Delta w_{ij}^S = 0$ .
3: $w_{ij}^S \leftarrow w_{ij}^S + \Delta w_{ij}^S$.
4: Go to Step 1 until $E$ is less than a given value or $\Delta w_{ij}^S \approx 0$.
```

End Algorithm Weights_Smallworld-AdNN

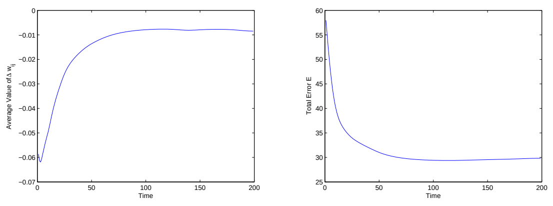

Fig. 1. The figure on the left shows the variation of the average of Δw_{ij}^S (averaged over all values of i and j) over the first 200 iterations of the gradient search scheme. The average converges to a value arbitrarily close to zero after 70 time steps. The figure on the right shows the variation of the global error over the same time frame. Observe that this quantity does not converge to *zero*.

the edges that the Smallworld-AdNN has, leading to a better-fitting effect. In practice, we have opted to choose $k/2 = 6$ to obtain a finer trade-off between the effect of the fit and the associated computational cost.

3 Chaotic and PR Properties of the Smallworld-AdNN

We now briefly report the PR properties of the Smallworld-AdNN. These properties have been gleaned as a result of examining the Hamming distance between the input pattern and the patterns that appear in the output. In this regard, we mention that the experiments were conducted using the Adachi data set, as shown in Fig. 3.

In the ideal setting we would have preferred the Smallworld-AdNN to be chaotic when exposed to untrained patterns, and the output to appear periodically or more frequently when exposed to trained patterns. Besides yielding this phenomenon, the Smallworld-AdNN also goes through a chaotic phase and a PR phase as some of its parameters change.

By studying Fig. 1 and 2 we see that if $k/2 = 6$, the Smallworld-AdNN can fit the original AdNN very well. Thus, we have set the parameters in Algorithm 1 to be $pr = 0.5$ and $K = 6$ so as to obtain a better trade-off effect. In this

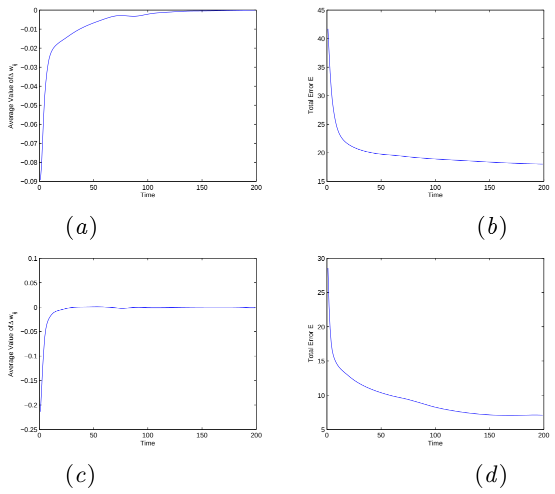

Fig. 2. The figures show the variation of the average of Δw_{ij}^S and the global error over the same time frame. The degree of the connection is $k = 12$ (for (a) and (b)) and $k = 20$ (for (c) and (d)) respectively.

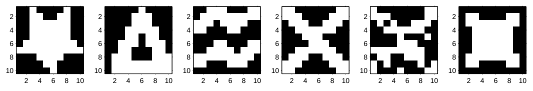

Fig. 3. The 10×10 patterns used by Adachi *et al* . The first four patterns are used to train the network. The fifth patterns are obtained from the corresponding fourth patterns by including 15% noise in (a) and (b) respectively. The sixth pattern is the untrained pattern.

regard, we comment that using the values of $pr = 0.5$ and $k/2 = 6$ are good enough for PR, which also significantly minimizes the computational burden. Indeed, as one can see, the distribution for the degree of each vertex of the Smallworld-AdNN has the form:

$$p(k) = \binom{N}{k-6} \left(\tfrac{3}{N}\right)^{k-6} \left(1 - \tfrac{3}{N}\right)^{N-k+6} \tag{6}$$

which is approximately a Poisson distribution, as shown in Fig. 4.

We summarize the results for the Smallworld-AdNN by using different settings of pr. The others parameters are: $k_f = 0.2$, $k_r = 1.02$, $\alpha = 10$, $\varepsilon = 0.015$, $\beta = 0.05$. The results are tabultaed in Tables 1, 2 and 3.

Consider Table 1. From this table we clearly see that the Smallworld-AdNN is able to "resonate" the input patterns with corresponding output patterns. If P1 is the input, then the network outputs P1 accordingly, while at the same time, no other trained patterns appear in the output sequence. Even when a noisy pattern is presented to the system, e.g., P5, which is a noisy pattern of P4 with 15% noise, the network still "resonates" P4 instead of P5 in the output sequence. Furthermore, if the input is an untrained pattern, e.g., P6, none of

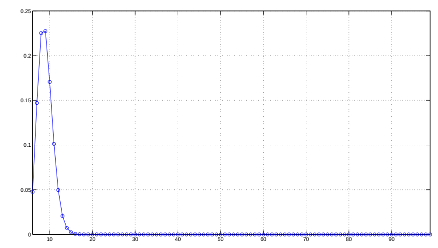

Fig. 4. The degree of each neuron obeys the Poisson distribution. From this figure we can see that most of the neurons have degree 8 or 9, which means that the computational load has been significantly reduced when compared to the original AdNN, which we know has a vertex degree of 99.

Table 1. The frequency of the Hamming distance between the input and the output patterns for the Smallworld-AdNN. The probability $pr = 0.5$ and $k/2 = 6$.

		Input Patterns					
$pr = 0.5$, $k/2 = 6$		P1	P2	P3	P4	P5	P6
	P1	96	0	0	0	0	0
	P2	0	376	0	0	0	0
Retrieved	P3	0	0	108	0	0	0
Patterns	P4	0	0	0	93	136	0
	P5	0	0	0	9	2	0
	P6	0	0	0	0	0	28

Table 2. The frequency of the Hamming distance between the input and the output patterns for the Smallworld-AdNN. The probability $pr = 0.1$ and $k/2 = 6$.

		Input Patterns					
$pr = 0.5$, $k/2 = 6$		P1	P2	P3	P4	P5	P6
	P1	49	0	0	0	0	0
	P2	0	126	0	0	0	0
Retrieved	P3	0	0	53	0	0	0
Patterns	P4	0	0	0	68	78	0
	P5	0	0	0	3	1	0
	P6	0	0	0	0	0	3

Table 3. The frequency of the Hamming distance between the input and the output patterns for the Smallworld-AdNN. The probability $pr = 0.9$ and $k/2 = 6$.

		Input Patterns					
$pr = 0.5$, $k/2 = 6$		P1	P2	P3	P4	P5	P6
	P1	578	0	0	0	0	0
	P2	0	685	0	0	0	0
Retrieved	P3	0	0	309	0	0	0
Patterns	P4	0	0	0	412	389	0
	P5	0	0	0	11	8	0
	P6	0	0	0	0	0	15

the trained patterns will be recalled. In this case, even the input pattern P6 itself, will be retrieved only a few times, as one can see is much less than the other diagonal numbers obtained when input is P1 – P4.

4 Conclusions

In this paper we have concentrated on the field of Chaotic Pattern Recognition (PR), which is a relatively new sub-field of PR. Such systems, which have only recently been investigated, demonstrate chaotic behavior under normal conditions. The system would, however resonate (or produce a single pattern more frequently) when it is presented with a pattern that it is trained with. The network which we have investigated is the Adachi Neural Network (AdNN) [2–4], based on which we have, ourselves, developed results in various avenues [3–6], including that of designing a NN that can yield *ideal* chaotic PR [7]. In this paper we have considered how the topology can be modified so as to render the network to be much closer to "real" neural networks. To achieve this, we have changed the network structure to be that of a Small-world graph, and then computed the best weights for the new graph by using a gradient-based algorithm. Apart from a Lyapunov analysis, by a detailed experimental suite, we have shown that the new Smallworld-AdNN possesses chaotic and PR properties for different settings.

References

1. Qin, K., Oommen, B.J.: Adachi-like chaotic neural networks requiring linear-time computations by enforcing a tree-shaped topology. IEEE Transactions on Neural Networks **20**(11) (2009) 1797–1809
2. Adachi, M., Aihara, K.: Associative dynamics in a chaotic neural network. Neural Networks **10**(1) (1997) 83–98
3. Calitoiu, D., Oommen, B.J., Nussbaum, D.: Desynchronizing a chaotic pattern recognition neural network to model inaccurate perception. IEEE Transactions on Systems Man and Cybernetics Part B-Cybernetics **37**(3) (2007) 692–704
4. Calitoiu, D., Oommen, B.J., Nussbaum, D.: Periodicity and stability issues of a chaotic pattern recognition neural network. Pattern Analysis and Applications **10**(3) (2007) 175–188
5. Qin, K., Oommen, B.J.: Chaotic pattern recognition: The spectrum of properties of the Adachi neural network. In: Lecture Notes in Computer Science. Volume 5342., Florida, USA (2008) 453–460
6. Qin, K., Oommen, B.J.: *Chaotic* pattern recognition using the Adachi neural network modified in a random manner. In: Proceedings of CHAOS'13, the 2013 Chaotic Modeling and Simulation International Conference, Istanbul, Turkey (2013) 540–550
7. Qin, K., Oommen, B.J.: Ideal chaotic pattern recognition is achievable: The Ideal-M-AdNN - its design and properties. In: Transactions on Computational Collective Intelligence (2013) 22–51
8. Watts, D.J., Strogatz, S.H.: Collective dynamics of 'small-world' networks. Nature **393**(4) (1998) 440–442
9. Newman, M., Barabsi, A.L., Watts, D.J.: The Structure and Dynamics of Networks. Princeton University Press, New Jersey (2006)
10. Qin, K.: Generic Analysis of Chaotic Neural Networks and Their Applications in Pattern Recognition and Crypto-systems. PhD thesis, University of Electronic Science and Technology of China, (2010)