Applied Intelligence 9, 191–200 (1998)

Evolutionary Learning of Modular Neural Networks with Genetic Programming

SUNG-BAE CHO
Department of Computer Science, Yonsei University, 134 Shinchon-dong, Sudaemoon-ku, Seoul 120-749, Korea; and ATR Human Information Processing Research Laboratories, 2-2 Hikaridai, Seika-cho, Soraku-gun, Kyoto 619-02, Japan

KATSUNORI SHIMOHARA
ATR Human Information Processing Research Laboratories, 2-2 Hikaridai, Seika-cho, Soraku-gun, Kyoto 619-02, Japan

Abstract. Evolutionary design of neural networks has shown a great potential as a powerful optimization tool. However, most evolutionary neural networks have not taken advantage of the fact that they can evolve from modules. This paper presents a hybrid method of modular neural networks and genetic programming as a promising model for evolutionary learning. This paper describes the concepts and methodologies for the evolvable model of modular neural networks, which might not only develop new functionality spontaneously, but also grow and evolve its own structure autonomously. We show the potential of the method by applying an evolved modular network to a visual categorization task with handwritten digits. Sophisticated network architectures as well as functional subsystems emerge from an initial set of randomly-connected networks. Moreover, the evolved neural network has reproduced some of the characteristics of natural visual system, such as the organization of coarse and fine processing of stimuli in separate pathways.

Keywords: neural networks, evolutionary computation, modules, emergence

1. Introduction

The design of neural networks by evolutionary algorithms has attracted great interest in the quest to develop adaptive systems that can change architectures and learning rules according to differing environments. There are more than one hundred publications that discuss evolutionary design methods applied to neural networks [1–7]. One of the important advantages of evolutionary neural networks is their adaptability to a dynamic environment, and this adaptive process is achieved through the evolution of connection weights, architectures and learning rules [5].

Designing the optimal architecture of neural networks can be formulated as searching in the space in which each point represents an architecture. The performance level of all the architectures forms a surface in the space. There are several characteristics in such a surface which make evolutionary algorithms promising candidates with respect to conventional learning models [8].

Most of the previous evolutionary neural networks, however, show little structural constraints. Some networks assume total connectivity between all nodes. Others assume a hierarchical, multilayered structure where each node in a layer is connected to all nodes in neighboring layers. However, there is a large body of neuropsychological evidence showing that the human information processing system consists of modules, which are subdivisions in identifiable parts, each with its own purpose or function.

The question may then be raised on how to design neural networks with various modules. There have also been extensive works to design efficient architectures

from an engineering point of view which has produced some success in several problems [9–11]. However, we know of no comprehensive analytical solution to the problem of relating architecture to function. This paper focuses on the evolutionary method that makes very few assumptions about the functions to be performed by the architecture.

The architecture of the brain has resulted from a long evolutionary process in which a large set of specialized subsystems emerged interactively carrying out the tasks necessary for survival and reproduction, but it appears that learning a large-scale task from scratch in such networks may take a very long time. Therefore, this paper takes a module as a building block for modular neural networks. Each module has the ability to autonomously categorize input activation patterns into discrete categories, and representations are distributed over modules rather than over individual nodes. Among the general principles are modularity, locality, self-induced noise, and self-induced learning. The proposed model of evolutionary neural networks is able not only to develop new functionality spontaneously but also to grow and evolve its own structure autonomously.

The rest of this paper is organized as follows. Section 2 briefly discusses some related works. In Section 3, we present the evolutionary modular neural networks in detail, and the simulation results with the problem of recognizing handwritten digits are shown in Section 4.

2. Related Works

Related work involves various kinds of growth-model coding for evolving neural networks: Kitano [3] and Gruau and Whitley [12] use grammatical encoding to develop artificial neural networks. Harp et al. [1] try to evolve the gross anatomy and general operating parameters of a network by encoding areas and projections onto them into the genome. Nolfi and Parisi [13] use an abstraction of axon growth to evolve connectivity architectures. Most of these models do not aspire to be biologically defensible, though.

In contrast, a number of other researchers have looked at more biologically inspired models of evolutionary process: Some work is based on the grammar-based approach first developed by Lyndenmayer. For instance, Mjolsness et al. [14] use grammatical rules to account for morphological change, coupled to a dynamical neural network to model the internal regulatory dynamics of the cell. Fleischer and Barr [15] have a hard-coded model for gene-expression that they combine with a cell simulation program.

It is the combination of a biologically defensible model of development with evolutionary methods that we would like to apply to the design of neural networks, something that at this point in time has not yet been addressed in the existing literature. The point of this research is to utilize modules as building blocks for developing neural networks by an evolutionary mechanism that is quite similar to genetic programming [16].

3. Evolutionary Modular Neural Networks

3.1. Overview

In order to evolve neural processing systems consisting of modules, a population of the individuals having various connectivity and size is maintained. Each individual is a modular neural network represented by a tree-structured chromosome. The module derives its basic internal structure from the neocortical minicolumn: Inhibitory connections are mostly short range, while long-range connections are mostly excitatory. As a mechanism for generating change and integrating the changes into the whole system, genetic programming produces the networks that have a variety of intermodule connectivity and module sizes, and Hebb learning rule applies to excitatory intermodular connections, whereas all intramodular connections are assumed to remain fixed. Each module serves as a higher order unit with a distinct structure and function.

Fig. 1 summarizes the overall algorithm for the proposed method. The initial population is generated with tree-structured chromosomes each of which is developed to a modular neural network of different connectivity and module sizes. After the developmental process, Hebb rule is applied to each network to train the intermodular connections with a training set, yielding a fitness value. If an acceptable solution is found, the algorithm stops. Otherwise, the next population is created from the current one with selection and genetic operations: The selection step accepts the better individuals into the mating pool, where genetic manipulation generates the new population with crossover and mutation operators. In our work to date, we have used an evolutionary algorithm similar to genetic programming with rank-based selection scheme [17].

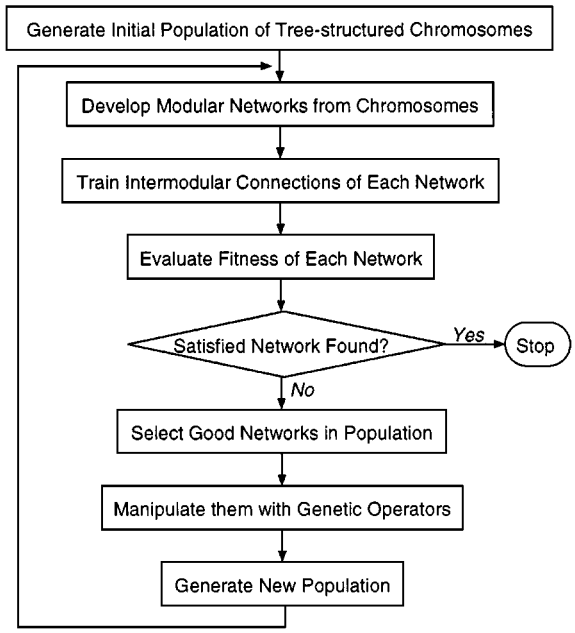


Figure 1. Algorithm for the proposed method.

3.2. *Modular Neural Networks*

The basic elements and structure of a module is designed to model neocortical minicolumns, as first proposed by Murre [18]. The activation of node i at time $t+1$, denoted as $a_i(t+1)$, is a function of its activation at t, $a_i(t)$, and its input excitation e_i which may be either positive or negative as follows:

- positive input:

$$a_i(t+1) = (1-k)a_i(t) + \frac{e_i}{1+e_i} \times [1-(1-k)a_i(t)], \quad e_i \geq 0 \quad (1)$$

- negative input:

$$a_i(t+1) = (1-k)a_i(t) + \frac{e_i}{1-e_i} \times (1-k)a_i(t), \quad e_i < 0 \quad (2)$$

where

$$e_i = \sum_j w_{ij} a_j(t) \quad (3)$$

w_{ij} denotes the weight of a connection from node j to node i (see Fig. 2), and k is a constant between 0 and 1. In our simulation, we set k as 0.05. In the activation

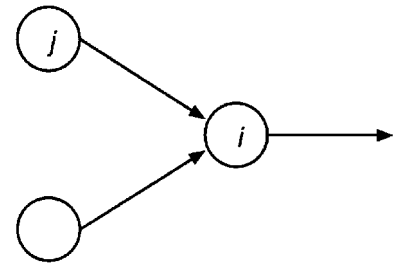


Figure 2. Interconnection between nodes.

rule, $(1-k)a_i(t)$ represents the autonomous decay of the activation, $\frac{e_i}{(1+e_i)}$ squashes the input excitation to a number between 0 and 1, and the third part causes an asymptotic approach to the maximum (or minimum) activation.

A module composed of the above nodes is loosely based on the neural structure of the neocortical minicolumn. A single node is either excitatory or inhibitory, but not both (Dale's law). Processes in nodes only rely on information that is locally available through synapses or through locally dispersed neurotransmitters (principle of locality). The internal connections in a module is fixed and the weights of all intramodular connections are nonmodifiable during learning process (see Fig. 3 and Table 1).

R-nodes have modifiable connections to the R-nodes in other modules and fixed connections to nodes in the same module. They are the only nodes in a module that may send or receive excitation to or from nodes in other modules. V-nodes only have inhibitory outgoing connections. A V-node inhibits all other nodes in a module, and in particular inhibits each other so strongly

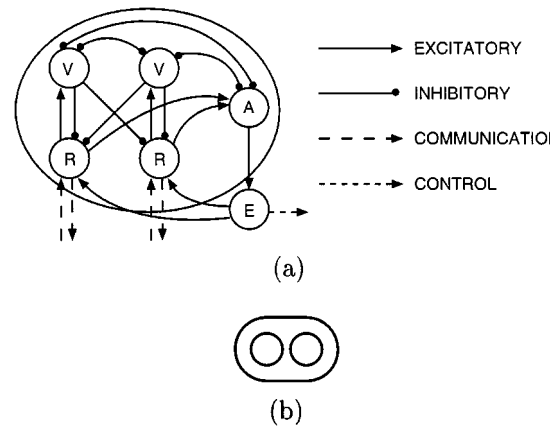


Figure 3. (a) Schematic diagram of the internal structure of a module. (b) Simplified representation of the module (a).

Table 1. Weight values used in each module that consists of several pairs of R-node and V-node.

Weights	Value
Connects R-node to its paired V-node	0.5
Connects V-node to its paired R-node	−1.2
Connects V-node to all R-nodes other than paired R-nodes	−10.0
Interconnects V-nodes	−1.0
Connects V-nodes to A-node	−0.6
Connects R-nodes to A-node	0.4
Connects A-node to E-node	1.0
Connects E-node to R-nodes	0.5

that competition arises among the V-nodes. The R- and V-nodes form matched pairs as every V-node receives excitatory input from only one R-node. A-nodes are excited by all R-nodes in a module and inhibited by all V-nodes. The activation of the A-node is a positive function of the amount of competition in a module, and E-node activation is a measure of the level of competition going on in a module.

The process goes with the resolution of a winner-take-all competition between all R-nodes activated by input. In the first presentation of a pattern to a module, all R-nodes are activated equally, which results in a state of maximal competition. It is resolved by the inhibitory V-nodes and a state-dependent noise mechanism. The noise is proportional to the amount of competition, as measured through the number of active R-nodes by the A-node and E-node.

The functional characteristics of a module does not depend on the size of the module, i.e., the number of R- and V-nodes. This number of the R-V pairs can be determined by the evolutionary process. The most important feature of a module is to autonomously categorize input activation patterns into discrete categories, which is facilitated as the association of an input pattern with a unique R-node.

The interconnection between two modules means that all R-nodes in one module are connected to all R-nodes in the other module. These intermodule connections are modifiable by Hebb rule with the following equation:

$$\Delta w_{ij}(t+1) = \mu_t a_i \left([K - w_{ij}(t)] a_j - L w_{ij}(t) \sum_{f \neq j} w_{if}(t) a_f \right), \tag{4}$$

$$\mu_t = d + w_{\mu_E} a_E, \tag{5}$$

where a_i, a_j and a_f are activations of the corresponding R-nodes, respectively: $w_{ij}(t)$ is the interweight between R-nodes j and i, $w_{if}(t)$ indicates an interweight from a neighboring R-node f (of j) to R-node i, and $\Delta w_{ij}(t+1)$ is the change in weight from j to i at time $t+1$. Note that L and K (K determines the maximum value of an interweight) are positive constants, d is a constant with a small value, and a_E is the activation of the E-node.

The first term within the large parentheses is always positive and represents increases in the weight. The second term is responsible for all decreases in the weight. An inactive connection ($a_j = 0$) to an activated node (a_i) will always decrease, because only the second term will be nonzero. In our simulation we set the parameters as $K = L = 1.0$, $d = 0.005$, and $w_{\mu_E} = 0.05$, but the exact value of the parameters does not seem to be very critical and indeed very large variations have been shown to produce similar qualitative behavior of a module.

3.3. Gene Representation

Two kinds of information should be encoded in the genotype representation: the structure of intermodule connection and the number of nodes in each module. The intermodule weights are determined by the Hebb rule mentioned at the previous section. In order to represent the information appropriately, a tree-like structure has been adopted. An arc in a tree expresses an intermodule connection, and each node represents a specific module and the number of nodes therein.

An example of the genotype is shown in Fig. 4. Each node has a number representing a specific module. In this figure the information on the number of nodes is omitted. The root of the tree is the input module that replaces the start symbol. A child node has a module number to be applied to the symbols which represent the modules connected by its mother module. In the course of decoding, the connections from a module to itself and the ones to the input module are ignored. This representation has some redundant connections as well as some meaningless ones, which might define some module that is not in the path from input to output modules. This is not a critical problem and even may be exploited to increase modeling power, because the literature in biology indicate that there are a lot of redundant and meaningless codes in the real genes.

By performing a number of genetic operators in the gene pool, the interconnection between modules as well as the size of them are changed. The range of

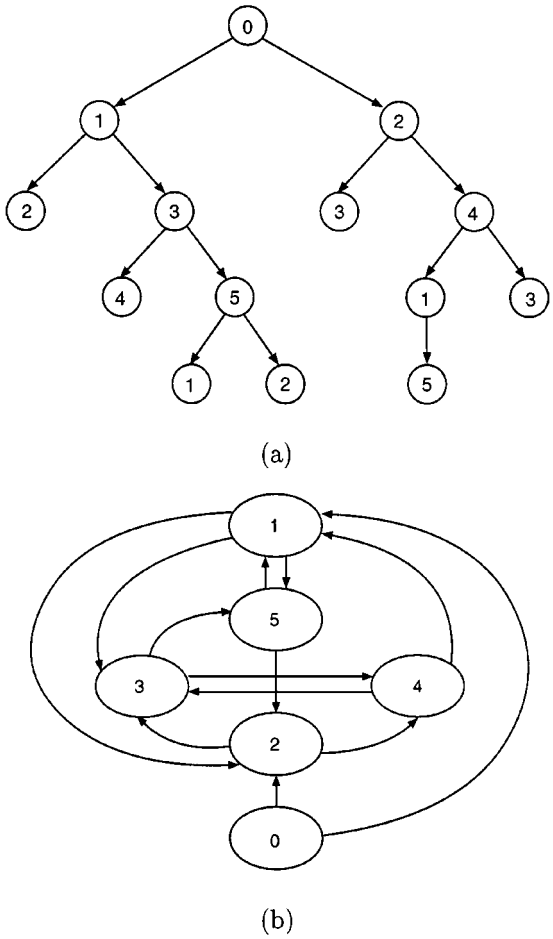


Figure 4. (a) A chromosome encoded by tree structure. (b) A modular neural network architecture developed by the chromosome of diagram (a).

the number of nodes in a module depends on the number of bits to represent it in chromosome, and in our simulation it is set as 10. Designing the chromosome to represent the interconnectivity makes it possible to generate a variety of offsprings and to evolve them.

3.4. Genetic Operators

The following genetic operators are used in our approach.

- **Selection:** Rank-based selection [19] is used. In this selection scheme, each individual survives to the next generation in proportion to the rank of its performance. Elite preserving strategy [16] is also applied to the selection. Some of the best individuals in the population are made to remain to the next generation. This prevents all of the best individuals from being eliminated by stochastic genetic drifts.
- **Crossover:** Crossover exchanges subtrees between two individuals. It is similar to the operator used in genetic programming [16]. By performing crossover, many useful interconnection parts are gathered, and the intermodular connectivity evolves. An example of the crossover is shown in Fig. 5(a).
- **Deletion:** Deletion deletes a subtree from the individual. This operator is expected to cause the deletion of useless parts in the individual. As a result, a more compact individual with the same functions might be generated (see Fig. 5(b)).
- **Mutation:** Mutation changes each tree node to a new node in proportion to the mutation rate. This operator plays the role of changing the number of nodes in the module. An important role of the mutation is to enforce local search and make slight modifications to the connectivity parts obtained by crossover and duplication. An example of mutation is shown in Fig. 5(c).
- **Insertion:** Insertion is to insert subtrees or nodes from another individual below the selected node (see Fig. 5(d)).

4. Simulation Results

In order to confirm the possibility of the proposed model, we have used the handwritten digit database of Concordia University of Canada, which consists of 6000 unconstrained digits originally collected from dead letter envelopes by the U.S. Postal Services at different locations in the U.S. The digits of this database were digitized in bilevel on a 64×224 grid of 0.153 mm square elements, giving a resolution of approximately 166 PPI [20]. Among the data, 300 digits were used for training, and another ten sets of 300 digits for testing. Figure 6 shows some representative samples taken from the database. We can see that many different writing styles are apparent, as well as digits of different sizes and stroke widths.

The size of a pattern was normalized by fitting a coarse, 10×10 grid over each digit. The proportion of blackness in each square of the grid provided 100 continuous activation values for each pattern. Network architectures generated by the evolutionary mechanism were trained with 300 patterns in two rounds of subsequent presentations. A single presentation lasted for 60 cycles (i.e., iterative updates of all activations and learning weights). A fitness value was assigned

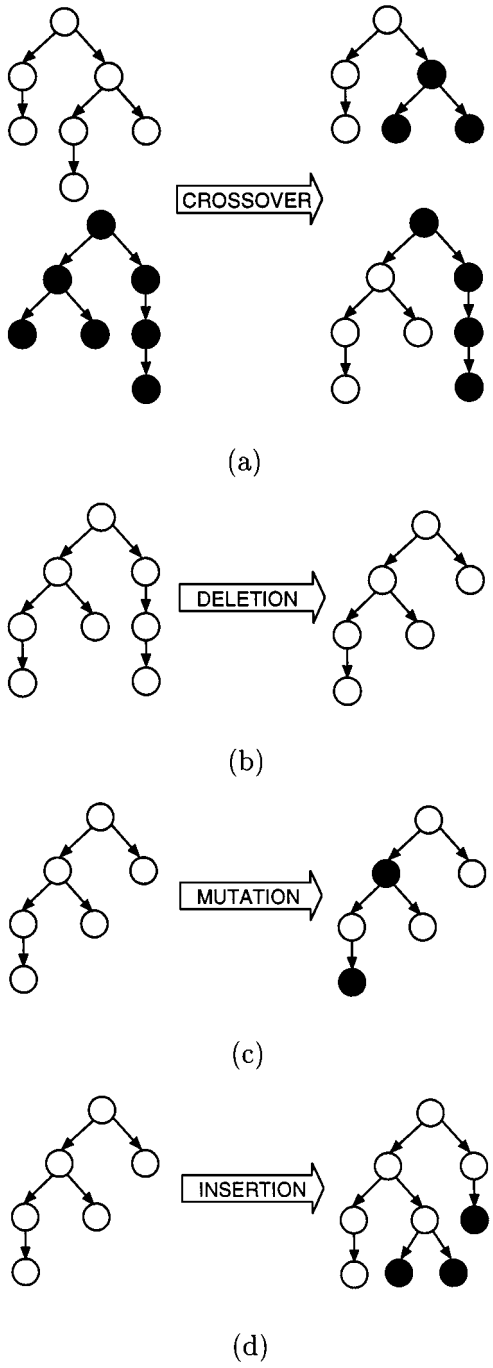


Figure 5. Graphical explanation on the genetic operators used.

to a solution by testing the performance of a trained network with the 300 training digits, and the generalization performance was tested on a set of untrained 300 digits.

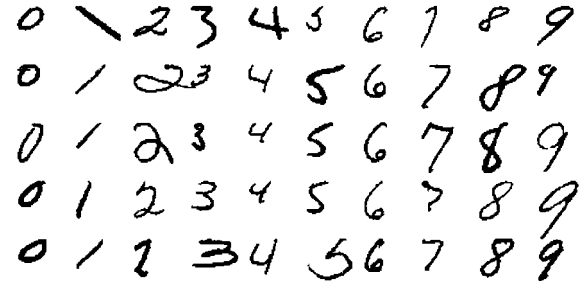


Figure 6. Sample data.

Initial population consisted of 50 neural networks having random connections. Each network contains one input module of size 100, one output module of size 10, and different number of hidden modules. The size in a module means the number of the R-V pairs, and every module can be connected to every other module. The evolution parameters used in this experiment are as follows: crossover is 0.5, mutation is 0.02, and insertion and deletion are 0.001, respectively.

Table 2 shows the results of generalization performance obtained by the networks at different generations, and Fig. 7 shows the corresponding networks evolved. As can be seen, evolution led to increased complexity, and new structures as well as new functionality emerged in the course of this method of training. Generally, the early networks have simple structures. In the early stages of the evolution some complicated architectures emerged like (d) and (e), but they disappeared as the search for an optimal solution matured. The earlier good solutions probably overfit the training set leading to poor generalization.

The last network (g) is the final architecture producing the best result. This contains four hidden modules of size 3, 3, 8 and 3, implementing different subsystems that cooperatively process input at different resolutions. The direct connection from the input module to the output module forms the most fine-grained processing stream. It is supplemented by a sophisticated modular structure in which two modules are globally connected with the input. A sort of hierarchical structure (IN $\rightarrow$ 2 $\rightarrow$ 4 $\rightarrow$ OUT) with feedback connections has emerged, and two coarser processing streams as well as local feedback projections support the main processing stream.

In the test of generalization capability, for the patterns that are similar to the trained, the network produced the direct activation through a specific pathway: one of the five basic pathways in Fig. 7(g) (IN $\rightarrow$ OUT,

Table 2. Generalization performance of the networks evolved with different complexity.

Network	Generation	Correct	No. of module	No. of node	No. of connection
1	5	91.33%	2	17	7
2	21	93.67%	2	8	7
3	35	95.00%	4	16	13
4	65	95.00%	4	16	14
5	178	95.00%	4	16	15
6	179	96.33%	4	14	16
7	203	97.33%	4	17	15

IN $\rightarrow$ 2 $\rightarrow$ 3 $\rightarrow$ OUT, IN $\rightarrow$ 2 $\rightarrow$ 4 $\rightarrow$ OUT, IN $\rightarrow$ 5 $\rightarrow$ 4 $\rightarrow$ OUT, IN $\rightarrow$ 2 $\rightarrow$ 3 $\rightarrow$ 4 $\rightarrow$ OUT). On the contrary, the network oscillated among several pathways to make a consensus for strange patterns. The basic processing pathways in this case complemented each other to result in an improved overall categorization. Furthermore, the recurrent connections utilized bottom-up and top-down information that interactively influences categorization at both directions. The oscillation is stopped when the whole network stabilizes as only one R-node at the output module remains to activate.

It is difficult to fully analyze the neural behaviors because they concern with the oscillatory activation dynamics. To make the analysis simpler, we have presented a sample of the class 1 to the final network and obtained a series of snapshots of the internal activations. This system has turned out to produce the correct result with respect to the input as shown in Table 3. In this network, the coupled oscillatory circuit between Module 4 and OUT module resolves the competition and induces the correct classification.

Table 3. A series of snapshots of the internal activations. The numbers in each column represent the activated nodes and $*$ means that there is no node activated in the module.

Step	OUT	Module 2	Module 3	Module 4	Module 5
1	0123456789	012	$*$	$*$	$*$
2	0123456789	012	012	01234567	012
3	0123456789	012	012	0234567	012
4	0123456789	0	012	023456	012
5	0123456789	0	012	03456	012
6	012345789	0	012	3456	012
7	12345789	0	012	456	012
8	178	0	012	5	012
9	1	0	012	5	012

In order to illustrate the effectiveness of the model proposed, a comparison with modular neural network without evolutionary algorithm and traditional neural network has been conducted. Basically, the structure of the modular neural networks cannot be determined without evolutionary algorithm, but for a comparison we manually design several structures among which the best one is as shown in Fig. 8. On the other hand, multilayer perceptron has been selected as a traditional neural network, because it is well known as a powerful pattern recognizer. The network is constructed as $100 \times 20 \times 10$, where the number of hidden nodes, 20, has been determined after several trial-and-errors. The error backpropagation algorithm is used for training and the iterative estimation process is stopped when an average squared error of 0.9 over the training set is obtained, or when the number of iteration reaches 1000, which is adopted mainly for preventing networks from overtraining. The parameter values used for training are: learning rate is 0.4 and momentum parameter is 0.6.

Table 4 reports the recognition rates of the three different methods over ten different sets of the data. As can be seen, the overall recognition rates for the proposed method are higher than those for the modular neural network without evolutionary algorithm and the multilayer perceptron. The following test, the paired t-test, can further support to determine whether the proposed method is superior to the other methods or not.

For a given test problem, let f_i^a denote the solution at convergence for method a using test data i. To test whether methods a and b have the same mean solution value, we compute the following statistic:

$$t = \frac{\sqrt{n}\bar{x}}{\sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2}} \tag{6}$$

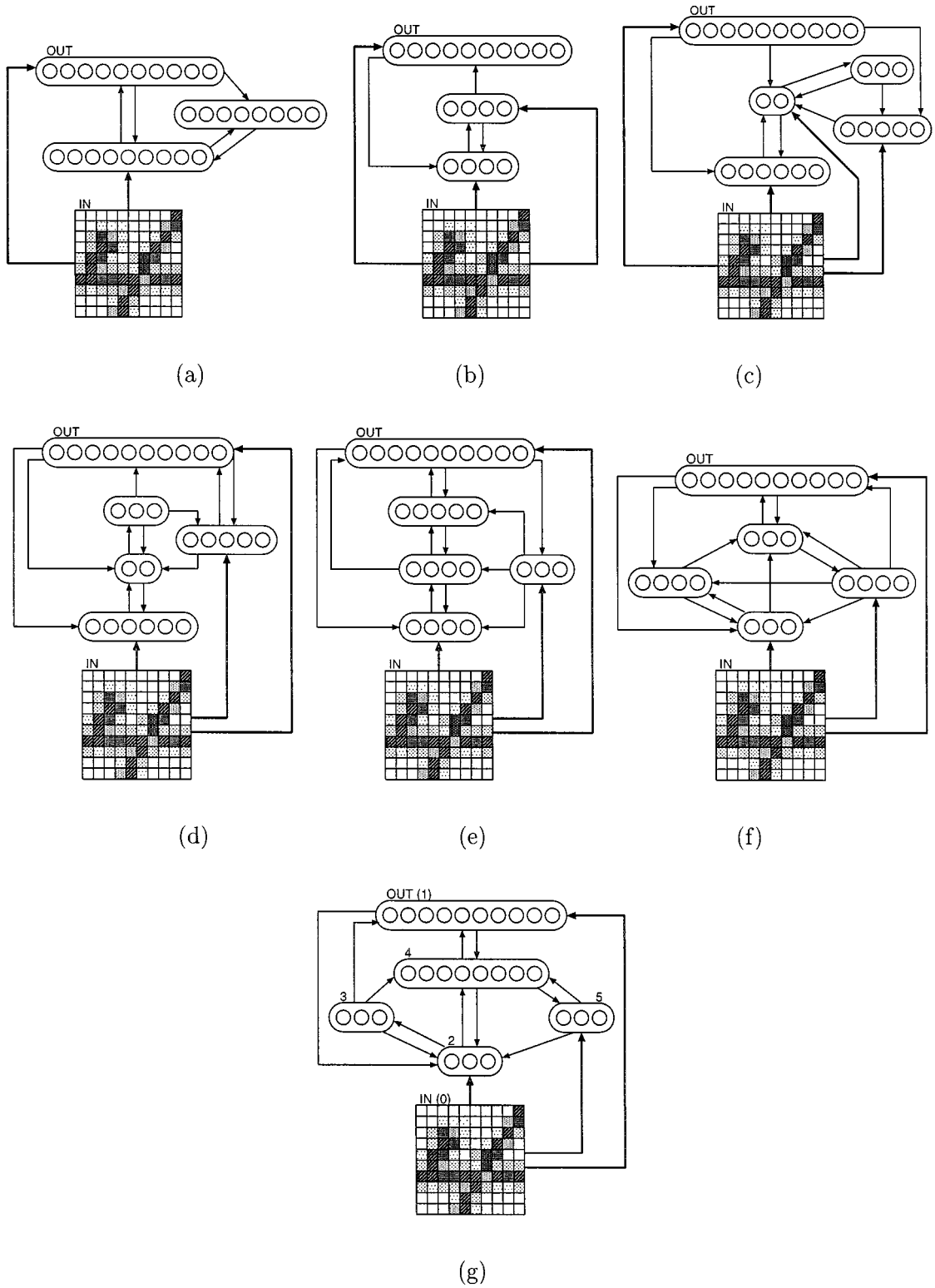


Figure 7. Some of the modular neural networks evolved: (a) 5th generation; (b) 21st generation; (c) 35th generation; (d) 65th generation; (e) 178th generation; (f) 179th generation; and (g) 203rd generation (final network).

Table 4. Comparison of the proposed method with the modular neural network without evolutionary algorithm (MNN without EA) and a traditional neural network (Traditional NN).

Data set	Proposed method	MNN without EA	Traditional NN
1	97.67	90.33	95.67
2	97.67	89.67	96.33
3	97.33	87.67	94.67
4	98.00	89.33	96.67
5	96.33	88.33	93.67
6	97.00	89.00	94.67
7	98.00	89.33	96.00
8	97.00	87.67	95.67
9	96.67	89.00	94.67
10	97.67	89.67	95.33
Mean	97.33	89.00	95.33
S.D.	0.57	0.87	0.92

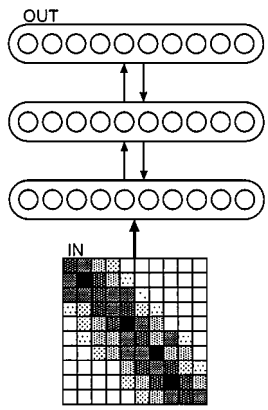


Figure 8. A modular neural network designed by hand for a comparison with the evolved network.

where $n = 10$, $x_i = f_i^a - f_i^b$, and $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$. (In this case the method b is of the proposed method.) From this value we can reject the null hypothesis that $H_0 : \bar{x} \leq 0$ in favor of the alternative that $\bar{x} > 0$ with significance level α, where $\alpha = \Phi(t)$ and $\Phi(t)$ can be obtained from the table of percentage points of the t-distribution.

Since it follows t-distribution, an α point can be computed as the threshold t_α, where α could be 95, 97.5, 99 or 99.5%. Then, if

$$|t| > t_\alpha \tag{7}$$

the null hypothesis is rejected at a $100\% - \alpha$ level of significance, i.e., the proposed method is superior to the alternative method. Otherwise, the null hypothesis is accepted, i.e., we cannot say the proposed method improves the performance significantly.

In the comparison, the degree of freedom is $(n - 1) = 9$, and the threshold t_α's with $\alpha = 95$, 97.25, 99 and 99.5% are 1.833, 2.262, 2.821 and 3.250, respectively. This test indicates that the values of t are greater than t_α with all the α values ($t = -36.059$ for MNN without EA and $t = -12.154$ for Traditional NN). Therefore, for all the cases "no-improvement" hypothesis is rejected at a 0.5% level of significance. This is a strong evidence that the proposed method is superior to the alternative methods.

However, the proposed model requires a large amount of time to evolve, which hinders from increasing more patterns for training, and this is not meant to be a practical pattern recognizer. We can just appreciate the effectiveness of evolution to design complex structures with some sophisticated network architectures automatically. Moreover, the proposed evolutionary neural networks can be learned incrementally, which demonstrates the relative superiority compared with the conventional neural networks such as backpropagation neural network.

5. Concluding Remarks

We have presented the modular neural networks developed by genetic programming, having potential of flexibility and adaptability to environmental changes. It has a modular structure with intramodular competition, and intermodular excitatory connections. This sort of network will also take an important part in several engineering tasks exhibiting adaptive behaviors. We are in the process of applying this method to design the control system for behavior-based robots, especially Khepera. Furthermore, we are exploring the dynamics of the modular neural networks once training is complete, and attempting to make the evolutionary mechanism sophisticated by incorporating the concept of coevolution and some developmental process like L-system into the evolutionary process.

Acknowledgments

The author would like to thank Dr. Y. Tohkura at ATR HIP laboratories for continuous encouragement. This work was supported in part by a grant no. 961-0901-009-2 from the Korea Science and Engineering Foundation (KOSEF) and a grant no. SC-13 from the Ministry of Science and Technology in Korea.

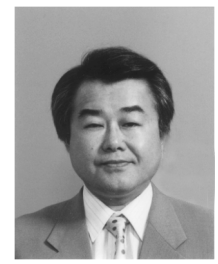
References

1. S.A. Harp, "Towards the genetic synthesis of neural networks," in *Proc. 3rd Int. Conf. Genetic Algorithms and Their Applications*, Morgan Kaufmann: San Mateo, CA, 1989, pp. 360–369.
2. D. Whitley and T. Hanson, "Optimizing neural networks using faster, more accurate genetic search," in *Proc. 3rd Int. Conf. Genetic Algorithms and Their Applications*, Morgan Kaufmann: San Mateo, CA, 1989, pp. 391–396.
3. H. Kitano, "Designing neural networks using genetic algorithms with graph generation system," *Complex Systems*, vol. 4, no. 4, pp. 461–476, 1990.
4. D.T. Cliff, I. Harvey, and P. Husbands, "Incremental evolution of neural network architectures for adaptive behavior," Technical Report CSRP 256, University of Sussex School of Cognitive and Computing Science, 1992.
5. X. Yao, "Evolutionary artificial neural networks," *Int. Journal of Neural Systems*, vol. 4, no. 3, pp. 203–222, 1993.
6. S. Nolfi, O. Miglino, and D. Parisi, "Phenotypic plasticity in evolving neural networks: Evolving the control system for an autonomous agent," Technical Report PCIA-94-04, Institute of Psychology, C.N.R., Rome, 1994.
7. S.-B. Cho and K. Shimohara, "Toward evolvable model of modularized neural networks," in *Proc. the 5th Annual Conf. Japanese Neural Network Society*, Tsukuba, November 1994, pp. 117–118.
8. G.F. Miller, P.M. Todd, and S.U. Hedge, "Designing neural networks using genetic algorithms," in *Proc. 3rd Int. Conf. Genetic Algorithms and Their Applications*, Morgan Kaufmann: San Mateo, CA, 1989, pp. 379–384.
9. S.-B. Cho and J.H. Kim, "Combining multiple neural networks by fuzzy integral for robust classification," *IEEE Trans. Systems, Man, and Cybernetics*, vol. 25, no. 2, pp. 380–384, 1995.
10. R.A. Jacobs, M.I. Jordan, S.J. Nowlan, and G.E. Hinton, "Adaptive mixtures of local experts," *Neural Computation*, vol. 3, pp. 79–87, 1991.
11. J.B. Hampshire II and A. Waibel, "The meta-pi network: Building distributed knowledge representations for robust multisource pattern recognition," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 14, no. 7, pp. 751–769, 1992.
12. F. Gruau and D. Whitley, "The cellular development of neural networks: The interaction of learning and evolution," Research Report 93-04, Laboratoire de l'Informatique du Parallelisme, Ecole Normale Superieure de Lyon, 1993.
13. S. Nolfi and D. Parisi, "Growing neural networks," Report PCIA-91-15, Institute of Psychology, C.N.R., Rome, 1991.
14. E. Mjolsness, D.H. Sharp, and J. Reinitz, "A connectionist model of development," *Journal of Theoretical Biology*, vol. 152, pp. 429–453, 1991.
15. K. Fleischer and A.H. Barr, "A simulation testbed for the study of multicellular development: The multiple mechanisms of morphogenesis," in *Artificial Life III*, edited by C.G. Langton, Addison-Wesley: Reading, MA, pp. 389–416, 1994.
16. J.R. Koza, *Genetic Programming on the Programming of Computers by Means of Natural Selection*, The MIT Press, 1992.
17. D. Whitley, "The GENITOR algorithm and selective pressure: Why rank-based allocation of reproductive trials is best," in *Proc. 3rd Int. Conf. Genetic Algorithms and Their Applications*, Morgan Kaufmann: San Mateo, CA, 1989, pp. 116–121,
18. J.M.J. Murre, R.H. Phaf, and G. Wolters, "CALM: Categorizing and learning module," *Neural Networks*, vol. 5, pp. 55–82, 1992.
19. D.E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, 1989.
20. C.Y. Suen, C. Nadal, T. Mai, R. Legault, and L. Lam, "Recognition of handwritten numerals based on the concept of multiple experts," in *Proc. 1st Int. Workshop Frontiers in Handwriting Recognition*, Montreal, Canada, 1990, pp. 131–144.

Sung-Bae Cho received the B.S. degree in computer science from Yonsei University, Seoul, Korea, in 1988 and the M.S. and Ph.D. degrees in computer science from KAIST (Korea Advanced Institute of Science and Technology), Taejeon, Korea, in 1990 and 1993, respectively.

He worked as a Member of the Research Staff at the Center for Artificial Intelligence Research at KAIST from 1991 to 1993. He was an Invited Researcher of Human Information Processing Research Laboratories at ATR (Advanced Telecommunications Research) Institute, Kyoto, Japan from 1993 to 1995. Since 1995, he has been an Assistant Professor in the Department of Computer Science, Yonsei University. His research interests include neural networks, pattern recognition, intelligent man-machine interfaces, evolutionary computation, and artificial life.

Dr. Cho was awarded outstanding paper prizes from the IEEE Korea Section in 1989 and 1992, and another one from the Korea Information Science Society in 1990. He was also the recipient of the Richard E. Merwin prize from the IEEE Computer Society in 1993. He was listed in Who's Who in Pattern Recognition from the International Association for Pattern Recognition in 1994, and received a best paper award at International Conference on Soft Computing in 1996. He is a Member of the Korea Information Science Society, INNS, the IEEE Computer Society, and the IEEE Systems, Man, and Cybernetics Society.



Katsunori Shimohara received the B.E. degree in 1976 and the M.E. degree in 1978 in information engineering from Kyushu University, Fukuoka, Japan. Currently he is Head of Evolutionary Systems Departments both at ATR Human Information Processing Research Laboratories and at NTT Communication Science Laboratories in Kyoto, and Guest Professor, Graduate School of Informatics, Kyoto University, Kyoto, Japan.