Modular Neural Networks Evolved by Genetic Programming

Sung-Bae Cho
Dept. of Computer Science
Yonsei University
Seoul 120-749, Korea
E-mail: sbcho@csai.yonsei.ac.kr

Katsunori Shimohara
ATR HIP Research Labs
2-2 Hikaridai, Seika-cho
Soraku-gun, Kyoto 619-02, Japan
E-mail: katsu@hip.atr.co.jp

ABSTRACT

In this paper we present an evolvable model of modular neural networks which are rich in autonomy and creativity. In order to build an artificial neural network which is rich in autonomy and creativity, we have adopted the ideas and methodologies of Artificial Life. This paper describes the concepts and methodologies for the evolvable model of modular neural networks, which will be able not only to develop new functionality spontaneously but also to grow and evolve its own structure autonomously. Although the ultimate goal of this model is to design the control system for such behavior-based robots as Khepera, we have attempted to apply the mechanism to a visual categorization task with handwritten digits. The evolutionary mechanism has shown a strong possibility to generate useful network architectures from an initial set of randomly-connected networks.

I. INTRODUCTION

Autonomous agents are self-governing systems capable of operating (i.e., perceiving and acting) in environments which are complex, uncertain, and dynamic. For the sake of brevity in the rest of the text, autonomous agents will be referred to simply as agents. Recently, some researchers have tried to synthesize the agents by using neural networks.

Most of the currently popular network architectures, however, show little structural constraints. Some networks assume total connectivity between all nodes. Others assume a hierarchical, multi-layered 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.

Question may then be raised at how to design the information processing system with various modules. There have been extensive works to design efficient architectures in engineering point of view [1; 2; 3], which has produced some success worthwhile several problems. On the contrary, the concept of evolving neural networks has appeared recently as a subject of intensive scientific discussions from biological, social, and engineering view points [4; 5; 6; 7; 8].

In order to build artificial neural network which is rich in autonomy and creativity, we have adopted the ideas and methodologies of Artificial Life. In this paper, we describe the concepts and methodologies for the evolvable model of modularized neural networks, which will be able not only to develop new functionality spontaneously but also to grow and evolve its own structure autonomously.

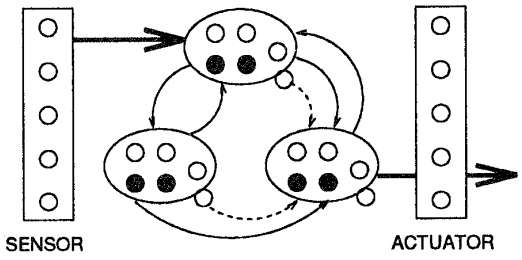

Fig. 1. Schematic diagram of modular neural networks.

II. EVOLUTIONARY MODULAR NEURAL NETWORKS

A. Overview

In order to give autonomy and creativity to a neural processing system, it is vital that the system itself should have some mechanism to spontaneously generate change in its function and structure [9]. The overall scheme of the proposed system looks like Figure 1. The basic idea is to consider a module as a building block resulting in local representations by competition, and develop complex intermodule connections with evolutionary mechanism.

The initial network architecture is encoded as a "gene." In our work to date, we have used a genetic encoding scheme which stores the "wiring diagram" for the network hierarchically. The encoding has been developed to be robust with respect to the genetic operators such as mutation and crossover.

B. Modular Neural Networks

The activation value of each node is calculated as follows:

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

where w_{ij} denotes the weight of a connection from node j to node i. The effective input to node i, e_i, is the weighted sum of the individual activations of all nodes connected to the input side of the node. The input may be either positive (excitatory) or negative (inhibitory).

A module based on such nodes is designed to model neocortical minicolumns [10]. The constraints embodied in the

0-7803-2902-3/96/$4.00 © 1996 IEEE

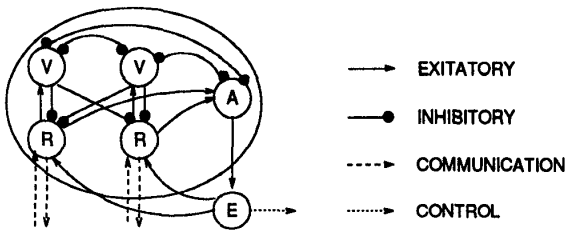

Fig. 2. Internal structure of a module.

model include:

1. Dale's principle (that individual neurons emit only one type of transmitter),
2. learning as a local phenomenon that does not require knowledge of the correct response, and
3. the capacity to differentiate between novel and familiar input and behave differently on that basis.

The internal structure of each module is fixed and the weights of all intramodular connections are non-modifiable (see Figure 2).

In a module, R-node represents a particular pattern of input activations to a module, V-node inhibits all other nodes in a module, and A-node activates a positive function of the amount of competition in a module. Evolutionary mechanism gives a possibility of change to the phenotype of a module through the genetic operators.

Hebb rule is used for learning intermodule connections with the following equation:

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

$$\mu_t = d + w_{\mu E} a_E,$$

where a_i, a_j and a_f are activations of the corresponding 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 $w_{ij}(t+1)$ is the change in weight from j to i at time $t+1$. Note that L and K are positive constants, and a_E is the activation of the E-node. As a mechanism for generating change and integrating the changes into a system, we use module duplication and elimination, and genetic algorithm to determine the parameters in the above learning rule and structure of intermodule connections.

C. Gene Representation

The gene has a tree structure that expresses the intermodule connection in which symbols are replaced to the corresponding modules. A gene example is shown in Figure 3. Each node has a number representing a specific module and several parameters on the number of nodes, as well as local settings of the learning and activation rules, and the fixed internal weights of intramodule connections. 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.

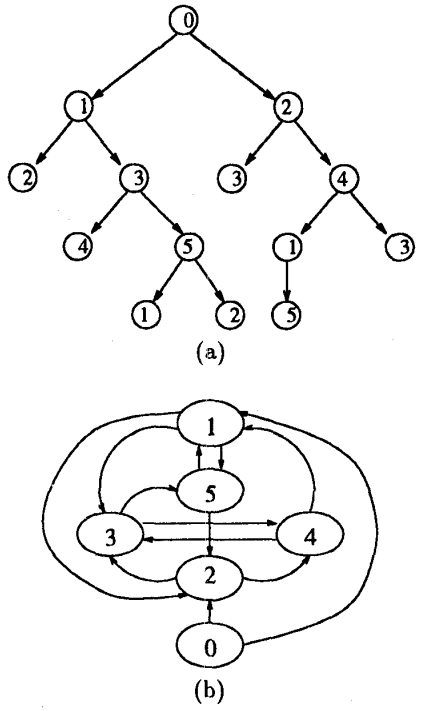

Fig. 3. (a) Genetic code encoded by tree structure; (b) A modular neural network architecture developed by the gene of diagram (a).

By performing the genetic operators in the gene pool, the interconnection between modules as well as the number of them are changed. Designing the gene to represent the interconnectivity makes it possible to generate a variety of offsprings and to evolve them.

D. Genetic Operators

The following genetic operators are used in our approach.

- **Selection**: Roulette wheel selection [11] is used. In roulette wheel selection, each individual survives to the next generation in proportion to its performance. Elitist strategy [12] is also applied to the selection. Some of the best individuals in the population are made to remain to the next generation. Elitist strategy prevents all of the best individuals from being eliminated by stochastic genetic drifts.
- **Crossover**: Crossover exchanges subtrees between two genes. It is similar to the operator used in Genetic Programming [12]. By performing crossover, many useful interconnection parts are gathered, and the intermodule connectivity evolves. An example of the crossover is shown in Figure 4(a).
- **Deletion**: Deletion deletes a function block from a gene. It is expected to delete useless parts in the gene. As a result, a more compact individual with the same functions is generated.

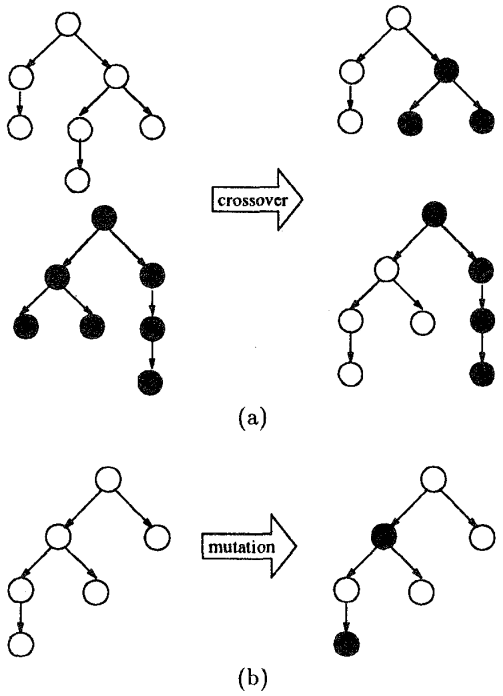

Fig. 4. Graphical explanation on the genetic operators used.

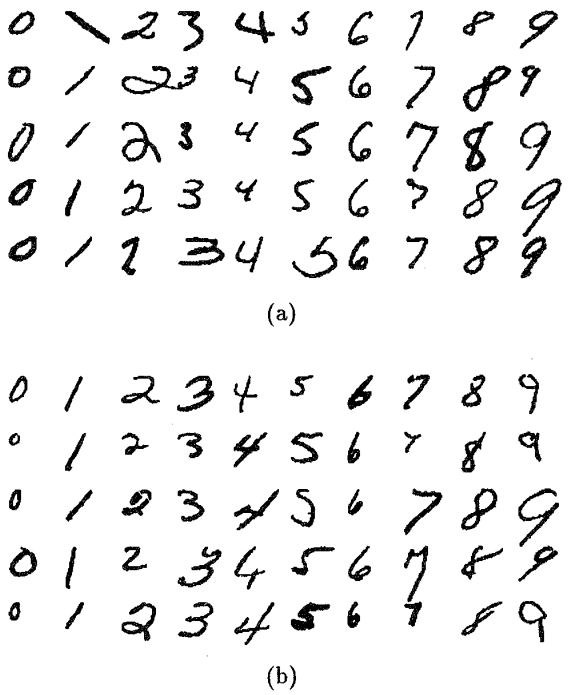

Fig. 5. (a) Sample data for training; (b) Sample data for test.

- **Mutation**: Mutation changes each tree node to a new node in proportion to the mutation rate. The mutated node is replaced with the new node and then the subtree below the original node is deleted. Next a new subtree is created below the new node according to some probability. The main roles of the mutation are on enforcing local search and making slight modifications to the connectivity parts obtained by crossover and duplication. An example of mutation is shown in Figure 4(b).
- **Insertion**: Insertion is similar to duplication except that it inserts a function block from another gene.
- **Duplication**: Duplication imitates the gene duplication [13] in living creatures and makes it possible to evolve the gene from a simple structure to a complex one. It also increases the complexity of functions and expands the network scale.

  Duplication inserts a copy of a function block within the same gene. The function block is the part of the gene where the nodes with the same category appear in a list. Duplication leads to a functionally correct interconnection. Just after duplication is performed, the inserted function block does not affect the individual's behavior and is neutral. Therefore, duplication does not change the functionality for the individual. The inserted block may be modified by mutations and a new function might emerge. The other parts of the gene also change and the inserted block is incorporated into the whole behavior.

III. Simulation Results

In order to confirm the possibility of the proposed model, a data set of handwriting characters was used as a source of both training and test samples. Handwriting characters were inputed to the computer (SUN workstation) by a Photron FIOS-6440 LCD tablet, which samples at the rate of 80 dots per second. The task was to learn and recognize handwritten digits. A sample of the task was initially obtained by having different writers draw 200 digits within prepared square boxes in order to facilitate segmentation. Figure 5 shows some examples of digit images used for training and test.

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 100 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 to a solution by testing the generalization performance of a trained network with the untrained half of the 200 digits.

Initial population consisted of 100 neural networks of having random connections. After making the population to evolve, we got 10 networks selected, and all of the networks produced more than 92% of the recognition rates. Figure 6(b) shows a final network architecture producing the best result, and Figure 6(a) depicts the corresponding

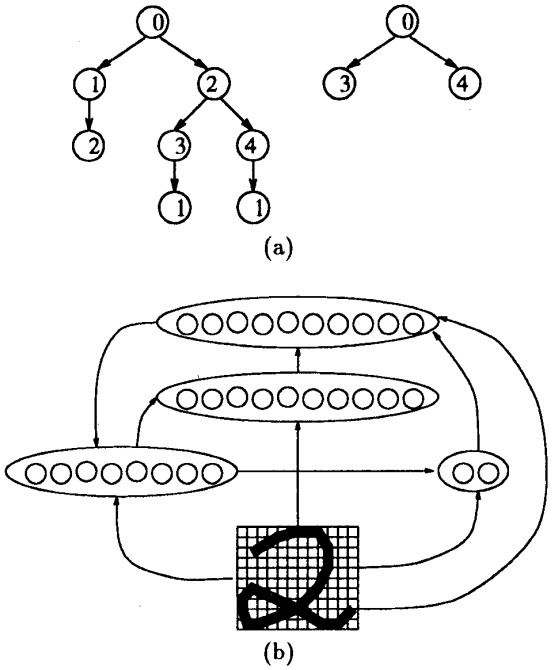

Fig. 6. (a) Genetic code encoded by a network producing the best result; (b) The corresponding modular neural network architecture developed by the gene of diagram (a).

genetic code. This contains three hidden modules of size 2, 8 and 10, 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 that is globally connected with the input and occupies two coarser processing streams as well as local feedback projections.

In the course of simulation, for the trained patterns and the patterns that are similar to the trained, the network produced the direct activation through a specific pathway. On the contrary, the network oscillated among several pathways to make a consensus for the strange patterns.

The performance is not competable to a practical pattern recognizer, but we can assure that the proposed evolutionary neural network works. Especially, we can appreciate the power of evolution to design complex structures with some sophisticated network architectures resulted from the evolution process.

IV. Concluding remarks

We have presented a preliminary design of the modular neural networks built by evolutionary mechanism, rich in flexibility, adaptability to environmental changes, and creativity. 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 under way to apply this method to design the control system for behavior-based robots.

Acknowledgements

This work was supported in part by a grant no. 961-0901-009-2 from the Korea Science and Engineering Foundation (KOSEF), and Center for Artificial Intelligence Research (CAIR), the Engineering Research Center (ERC) of Excellence Program.

References

[1] S.-B. Cho and J.H. Kim, "Combining multiple neural networks by fuzzy integral for robust classification," *IEEE Trans. on Systems, Man, and Cybernetics*, vol. 25, no. 2, pp. 380–384, 1995.

[2] 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.

[3] 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.

[4] S.A. Harp, "Towards the genetic synthesis of neural networks," in *Proc. Genetic Algorithms*, pp. 360–369, 1989.

[5] D. Whitley and T. Hanson, "Optimizing neural networks using faster, more accurate genetic search," in *Proc. Genetic Algorithms*, pp. 391–396, 1989.

[6] H. Kitano, "Designing neural networks using genetic algorithms with graph generation system," *Complex Systems*, vol. 4, no. 4, pp. 461–476, 1990.

[7] 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.

[8] 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.

[9] K. Shimohara, "Evolutionary systems for brain communications–Towards an artificial brain," In R. Brooks and P. Maes (Eds.), *Artificial Life IV*, MIT Press, 1994.

[10] J.M.J. Murre, R.H. Phaf and G. Wolters, "CALM: Categorizing and learning module," *Neural Networks*, 5, 55–82, 1992.

[11] D.E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison Wesley, 1989.

[12] J.R. Koza, *Genetic Programming on the Programming of Computers by means of Natural Selection*, The MIT Press.

[13] K. Wada and S. Tanaka, "Can GAs survive?," *Journal of the Society of Instrument and Control Engineers*, vol. 32, no. 1, pp. 17–23, 1993.