A Modular Massively Parallel Learning Framework for Brain-Like Computers

Bao-Liang Lu *, Michinori Ichikawa *, & Shegeyuki Hosoe †‡
*Lab. for Brain-Operative Device, Brain Science Institute, RIKEN
2-1 Hirosawa, Wako-shi, 351-0198, Japan; {lu;ichikawa}@brainway.riken.go.jp
†Bio-Mimetic Control Research Center, RIKEN
Anagahora, Shimoshidami, Moriyama-ku, Nagoya 456-0003, Japan
‡Dept. of Electronic-Mechanical Engineering, Nagoya University
Furo-cho, Chikusa-ku, Nagoya, 464-8601, Japan; hosoe@nuem.nagoya-u.ac.jp

Abstract: It is generally believed that a brain-like computer should possess the following essential capabilities: (a) massively parallel and distributed information processing; (b) real-time information processing; (c) flexible information processing; and (d) solving large-scale problems. However, it seems that there are few existing neural network models which can satisfy the above basic requirements currently. In this paper, we present a massively parallel and modular learning framework for brain-like computers. We narrow our sights to consider only pattern recognition problems and discuss the characteristics of the framework from the aspects of modularity and parallelism, responsiveness, plasticity, and scalability. We demonstrate that the framework may provide us with a simple model for implementing specific brain-like computers for pattern recognition.

1 Introduction

To develop brain-like computers is one of the grand goals of both brain science and computer science. Various definitions of brain-like computers have been given by the researchers in different fields [12, 4, 13]. It is generally believed that a brain-like computer should possess the following essential capabilities: (a) massively parallel and distributed information processing; (b) real-time information processing; (c) flexible information processing; and (d) solving large-scale problems. However, it seems that there are few existing neural network models which are suitable for building brain-like computers currently.

In this paper we present a massively parallel and modular learning framework for brain-like computers. Figure 1 gives an overview of the proposed framework. We narrow our sights to consider only pattern recognition problems and discuss the characteristics of the framework. The remainder of the paper is organized as follows. In Section 2, we present the proposed learning framework. In Section 3, we discuss the characteristics of the framework from the aspects of modularity and parallelism, responsiveness, plasticity, and scalability. Finally, the conclusion is given in Section 4.

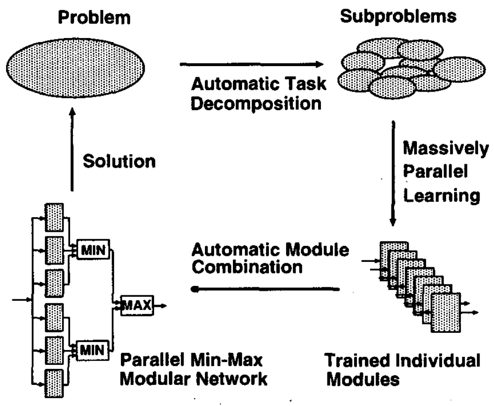

Problem
Subproblems
Automatic Task Decomposition
Solution
Massively Parallel Learning
MIN
MAX
MIN
Automatic Module Combination
Parallel Min-Max Modular Network
Trained Individual Modules

Figure 1: Overview of the proposed modular massively parallel learning framework.

2 Learning Framework

2.1 Automatic task decomposition

Let $\mathcal{T}$ be the training set for a K-class classification problem,

$$\mathcal{T} = \{(\boldsymbol{x}_t, \boldsymbol{y}_t)\}_{t=1}^{N}, \tag{1}$$

where $\boldsymbol{x}_t \in \mathbf{R}^n$ is the input vector, $\boldsymbol{y}_l \in \mathrm{R}^K$ is the desired output, and N is the total number of training data.

We suggest that a K-class problem as defined by (1) can be divided into $\binom{K}{2}$ relatively smaller two-class subproblems [9, 10]. The training set $\mathcal{T}_{ij}$ for each of the subproblems is given by

$$\mathcal{T}_{ij} = \{(\boldsymbol{x}_l^{(i)}, 1-\epsilon)\}_{l=1}^{L_i} \cup \{(\boldsymbol{x}_l^{(j)}, \epsilon)\}_{l=1}^{L_j} \tag{2}$$

$$\text{for } i = 1, \cdots, K \text{ and } j = i+1, \cdots, K,$$

where ϵ is a small real positive number, $\boldsymbol{x}_l^{(i)} \in \mathcal{X}_i$ and $\boldsymbol{x}_l^{(j)} \in \mathcal{X}_j$ are the training inputs belonging to class $\mathcal{C}_i$ and class $\mathcal{C}_j$, respectively, $\mathcal{X}_i$ is the training input set of class $\mathcal{C}_i$, and L_i denotes the number of data in $\mathcal{X}_i$ for $i = 1, \cdots, K$.

Assume that $\mathcal{X}_i$ is partitioned into N_i $(1 \leq N_i \leq L_i)$ subsets in the form

$$\mathcal{X}_{ij} = \{\boldsymbol{x}_l^{(ij)}\}_{l=1}^{L_i^{(j)}} \qquad (3)$$
$$\text{for } j = 1, \cdots, N_i \text{ and } i = 1, \cdots, K,$$

where $\cup_{j=1}^{N_i} \mathcal{X}_{ij} = \mathcal{X}_i$.

According to the above partition of $\mathcal{X}_i$, a K-class classification problem can be divided into

$$\sum_{i=1}^{K} \sum_{j=i+1}^{K} N_i \times N_j \qquad (4)$$

relatively smaller and simpler two-class subproblems.

The training set for each of the subproblems is given by

$$\mathcal{T}_{ij}^{(u,v)} = \{(\boldsymbol{x}_l^{(iu)}, 1-\epsilon)\}_{l=1}^{L_i^{(u)}} \cup \{(\boldsymbol{x}_l^{(jv)}, \epsilon)\}_{l=1}^{L_j^{(v)}} \qquad (5)$$
$$\text{for } u = 1, \cdots, N_i,\ v = 1, \cdots, N_j,$$
$$i = 1, \cdots, K, \text{ and } j = i+1, \cdots, K,$$

where $\boldsymbol{x}_l^{(iu)} \in \mathcal{X}_{iu}$ and $\boldsymbol{x}_l^{(jv)} \in \mathcal{X}_{jv}$ are the training inputs belonging to class $\mathcal{C}_i$ and class $\mathcal{C}_j$, respectively.

If the training set $\mathcal{T}_{ij}^{(u,v)}$ has only two different elements in the form

$$\mathcal{T}_{ij}^{(u,v)} = \{(\boldsymbol{x}_1^{(iu)}, 1-\epsilon) \cup (\boldsymbol{x}_1^{(jv)}, \epsilon)\} \qquad (6)$$
$$\text{for } u = 1, \cdots, L_i,\ v = 1, \cdots, L_j,$$
$$i = 1, \cdots, K, \text{ and } j = i+1, \cdots, K,$$

it is obviously a linearly separable problem because any two different training data can always be separated by a hyper-plane.

From (2) and (5), we can see that dividing a K-class problem into a number of two-class subproblems is simple and straightforward, and no domain specialists or a prior knowledge concerning the decomposition of the problem are required. Consequently, the task decomposition can be performed automatically.

2.2 Massively Parallel Learning

An important feature of the proposed task decomposition method is that each of the two-class subproblems obtained can be treated as a completely separate classification problem in the learning phase. Consequently, all of the subproblems can be learned in parallel.

Let N be the total number of training data for a K-class classification problem, then

$$N = K \times L, \qquad (7)$$

where for simplicity of description, the assumption we made is that each of the classes has the same number of training data L.

If a K-class problem is decomposed into $\binom{K}{2}$ two-class subproblems, the number of training data for each of the subproblems is $2 \times L$. If a K-class problem is decomposed into $\sum_{i=1}^{K} \sum_{j=i+1}^{K} N_i \times N_j$ two-class subproblems, the number of of training data for each of the subproblems is about

$$\lceil L/N_i \rceil + \lceil L/N_j \rceil, \qquad (8)$$

where $\lceil z \rceil$ denotes the smallest integer greater than or equal to z. Since $\lceil L/N_i \rceil + \lceil L/N_j \rceil \ll K \times L$ for a large K, i.e., the number of training data for each of the two-class subproblems is much less than the original K-class problem. Our experiences indicate that to learn each of the subproblems by using a small network module is much faster than to learn the original large problem by using a large single network [10].

2.3 Min-Max Modular Network

After training each of the modules which were assigned to learn associated subproblems, all of the individual trained modules can be easily integrated into a min-max modular (M^3) network by using the MIN, MAX, or/and INV units according to the module combination principles [9]. Figures 2(a) and 2(b) illustrate the M^3 networks for a three-class and a four-class pattern classification problems, respectively, where the three-class problem is decomposed into $\binom{3}{2}$ two-class subproblems and the four-class problem is decomposed into $\binom{4}{2}$ two-class subproblems according to (2).

Let $\boldsymbol{y}$ denote the actual output vector of the M^3 network for a K-class classification problem, and let $g(\boldsymbol{x})$ denote the transfer function of the M^3 network. We may then write

$$\boldsymbol{y} = g(\boldsymbol{x}) = [g_1(\boldsymbol{x}), \cdots, g_K(\boldsymbol{x})]^T \qquad (9)$$

where $\boldsymbol{y} \in \mathbf{R}^K$, and $g_i(\boldsymbol{x}) \in \mathbf{R}$ is called the *discriminant function*, which discriminates the patterns of class $\mathcal{C}_i$ from those of the rest classes.

By replacing the module M_{st} for $s > t$ with the inverse of the output of the module M_{ts}, the discriminant functions $g_i(\boldsymbol{x})$ of the M^3 network which is used to learn the $\binom{K}{2}$ two-class subproblems can be given by

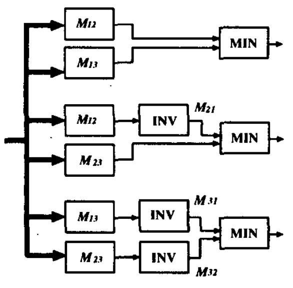
$$g_i(\boldsymbol{x}) = \min \left[\min_{j=i+1}^{K} h_{ij}(\boldsymbol{x}), \min_{r=1}^{i-1} (b - h_{ri}(\boldsymbol{x})) \right] \qquad (10)$$

where the term $b - h_{ri}(\boldsymbol{x})$ denotes the inverse of $h_{ri}(\boldsymbol{x})$. which can be implemented by the INV unit, b denotes the upper limit of the output value of each module, and $h_{ij}(\boldsymbol{x})$ is the activation function of the module M_{ij} trained on $\mathcal{T}_{ij}$ as defined in (2).

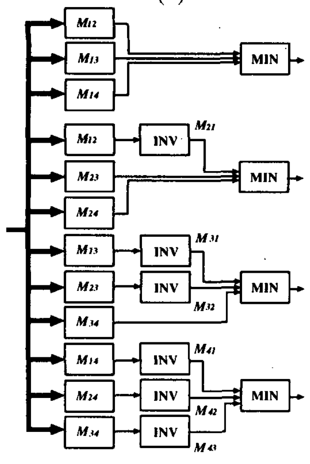
Similarly, the discriminant function $g_i(\boldsymbol{x})$ of the M^3 network which is used to learn $\sum_{i=1}^{K}\sum_{j=i+1}^{K} N_i \times N_j$ two-class subproblems can be expressed as

$$g_i(\boldsymbol{x}) = \min\left[\min_{j=i+1}^{K}\left[\max_{k=1}^{N_i}\left[\min_{l=1}^{N_j} h_{ij}^{(k,l)}(\boldsymbol{x})\right]\right],\right.$$
$$\left.\min_{r=1}^{i-1}\left(b - \max_{k=1}^{N_r}\left[\min_{l=1}^{N_i} h_{ri}^{(k,l)}(\boldsymbol{x})\right]\right)\right], \quad (11)$$

where the term $b - \max_{k=1}^{N_r}[\min_{l=1}^{N_i} h_{ri}^{(k,l)}(\boldsymbol{x})]$ denotes the inverse of $\max_{k=1}^{N_r}[\min_{l=1}^{N_i} h_{ri}^{(k,l)}(\boldsymbol{x})]$ and $h_{ij}^{(k,l)}$ is the activation function of the module $M_{ij}^{(k,l)}$ trained on $\mathcal{T}_{ij}^{(k,l)}$. Figure 2 shows the M^3 network for the DNA problem, which is decomposed into 24 two-class subproblems with $N_1 = 2$, $N_2 = 2$, and $N_3 = 5$ according to (5).



(a)



(b)

Figure 2: The M^3 networks: (a) for a three-class problem and (b) for a four-class problem.

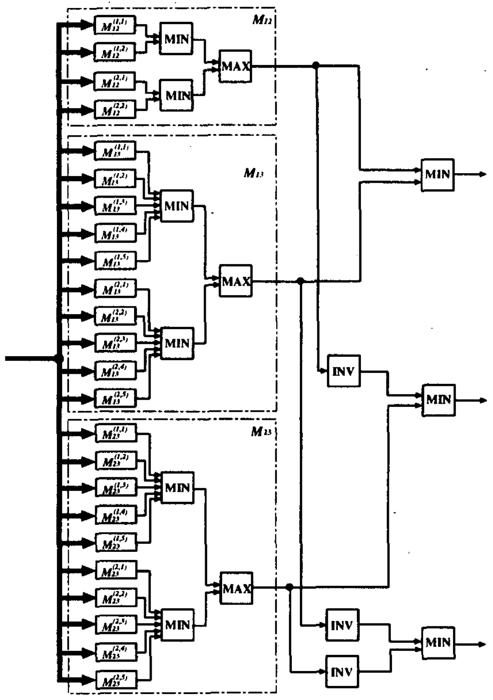


Figure 3: The M^3 network for the DNA problem

3 Characteristics

3.1 Modularity and Parallelism

A neural network is said to be modular if it consists of two or more modules (subnetworks) that operate on distinct inputs without communicating with each other. An output of the modular network is generated by combining the outputs of related modules with integrating units. The function of the integrating unit is to decide which module should output [14].

According to the above definition, the M^3 network is a typical modular network. Figures 4(a) and 4(b) represent the structures of the M^3 networks for implementing $g_i(x)$ of (10) and $g_i(x)$ of (11), respectively.

The degree of parallelism is the number of subtasks available. According to the task decomposition method presented in Section 2, it is easy to see that any pattern classification problem can be divided into a number of completely independent, non-communicating subtasks. This is ideal case called *completely parallelizable* in parallel computing literature [1, 6], since learning algorithms can achieve linear (i.e., proportional to processing elements) speedup as processing elements (PEs) are added.

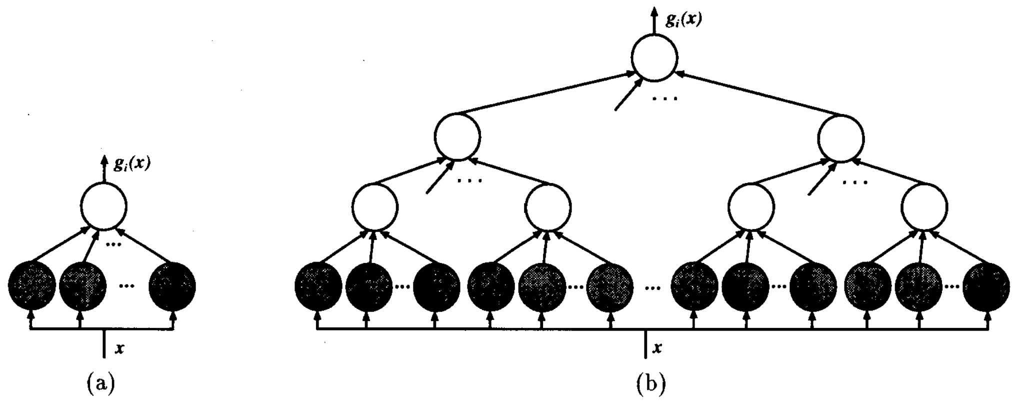


Figure 4: The tree structures of the M^3 networks: (a) the tree for the discriminant function as defined in (10) and (b) the tree for the discriminant function as defined in (10), where open circles denote the MIN or MAX units, and dashed circles denote network modules.

3.2 Responsiveness

Responsiveness is very important because brain-like computers should have rapid response or real-time response as the human brain [15]. Let us analyze the responsiveness of the M^3 networks.

Case 1: If a K-class classification problem is just divided into $\binom{K}{2}$ two-class subproblems, the topology of the M^3 network for the discriminant function $g_i(x)$ of (10) is a tree with two levels as depicted in Fig. 4(a). The running time for recognizing an input x is given by

$$P_i = P_m + \log(K-1) \qquad (12)$$

where P_m is the maximum response time among the $K-1$ individual modules and $\log(K-1)$ is the running time for finding the minimal value from $K-1$ terms with $K(K-1)/2$ PEs [5].

Case 2: If a K-class classification problem is divided into $\sum_{i=1}^{K}\sum_{j=i+1}^{K} N_i \times N_j$ relatively smaller and simpler two-class problems, the topology of the M^3 network for the discriminant function $g_i(x)$ of (11) is a tree with four levels as depicted in Fig. 4(b). Suppose that all the MIN and MAX units are implemented in parallel. The running time for recognizing an input x is given by

$$P_i = P_m + 2\log(N_\alpha) + \log(K-1) \qquad (13)$$

where N_α is the maximal value among $N_1, \cdots, N_K$.

3.3 Plasticity

One of the most important advantages of the human brain over existing artificial neural network models is its plasticity [16]. Although much progress has been made in understanding of the plasticity of synapses of biological neurons [16, 2], few artificial neural network models that have plasticity have been reported in the literature [3, ?]. The ability of a network that can be modified and extended efficiently during or after learning is called ***plasticity*** [11].

Two aspects of the plasticity of a neural network can be considered: *local* plasticity and *global* plasticity. The adjustments of connection weights during learning can be understood as one kind of local plasticity of a network. Almost all of existing research on the plasticity of biological and artificial neural networks in experimental and computational neurosciences remains at the local plasticity level. The global plasticity of a network means the ability of a network whose architecture can be changed efficiently after it had been trained. In this subsection we focus on the global plasticity of the M^3 networks.

Suppose that training set $\mathcal{T}$ had been successfully learned by an M^3 network, namely $net_{\mathcal{T}}$. For some new requirements, $\mathcal{T}$ may be modified. Let $\mathcal{U}$ be a set of new classes of data,

$$\mathcal{U}' = \{(\boldsymbol{x}_i', \boldsymbol{y}_i')\}_{i=1}^{M}, \qquad (14)$$

where $\boldsymbol{x}_i' \in \mathbf{R}^P$ is different from any input vectors in $\mathcal{T}$, and $\boldsymbol{y}_i' \in \mathbf{R}^N (N > K)$. We discuss the problem of how to add $\mathcal{U}$ to previously trained network $net_{\mathcal{T}}$.

According to the task decomposition method and the structures of the M^3 networks mentioned above, the problem of adding $\mathcal{U}$ to $net_{\mathcal{T}}$ can be solved by training modular networks on the following two-class problems and adding the trained modules to $net_{\mathcal{T}}$. These two-class problem can be defined as

$$\mathcal{U}_{ij} = \{(\boldsymbol{x}_l^{(i)}, 1-\epsilon)\}_{l=1}^{L_i} \cup \{(\boldsymbol{x}_l^{(j)}, \epsilon)\}_{l=1}^{L_j} \qquad (15)$$

for $i = 1, \cdots, N$ and $j = K, \cdots, N$ and $j \neq i$

and

$$U'_{ij} = \{(x_l^{(i)},\ 1-\epsilon)\}_{l=1}^{L_i} \cup \{(x_l^{(j)},\ \epsilon)\}_{l=1}^{L_j} \quad (16)$$

for $i = K, \cdots, N$ and $j = 1, \cdots, N$ and $j \neq i$

The number of two-class problems defined in Eqs. (15) and (16), which are different from those in Eq. (2), is given by

$$N \times (N-1) - K \times (K-1) = \alpha(2 \times K + \alpha - 1) \quad (17)$$

and the number of two-class problems that need to be learned is

$$\frac{\alpha(2K+\alpha-1)}{2}, \quad (18)$$

where $\alpha = N - K$.

For example, suppose that net_T is a trained M^3 network for a three-class problem as shown in Fig. 2(a). In order to add one new class of data to net_T, six modules need to be added to net_T. The extended network (net_{TU}) for a new four-class problem is shown in Fig. 2(b). Comparing net_T with net_{TU}, we can see that all of the trained modules in net_T, i.e., M_{12}, M_{13}, and M_{23}, are reused in net_{TU} in their original condition.

Since MIN, MAX, and INV units do not need to be trained and the number of fan-in of MIN and MAX units can be increased easily in both software and hardware, adding new data to the trained network can be achieved efficiently. The main cost is to learn new two-class problems as defined in Eq. (18).

3.4 Scalability

The simplest definition of *scalability* is that the performance of a computing system increases linearly with respect to the number of PEs used for a given application [6]. However, most of existing neural network models face the *scaling problem*, i.e., training neural networks become intractable as problem sizes get large [8].

The M^3 network is able to achieve scalable performance as the learning problem size increases because any problem can be divided into a number of subproblems as small as a user expects and each of the subproblems can be treated as a completely separate problem. Consequently, the proposed learning framework can overcome the scaling problem.

4 Simulation Results

In order to show the effectiveness of the proposed learning framework, we carry out simulations on the handwritten digit recognition problem [7].

The training set and test set for the handwritten digit recognition problem consist of 7291 and 2007 data, respectively. Figure 5 shows ten handwritten numerals that were segmented from the handwritten zip codes. The image for each handwritten ZIP code data contains 16 pixel rows by 16 pixel columns, for a total 256 pixels. Each image corresponds to a vector $x \in \mathbf{R}^{256}$ with each component value varying from 0 (white) to 1 (blank) determined by the gray level in the corresponding pixel.

Table 1: Performance comparison of LeNet and the M^3 network on the handwritten digit problem. Note that the CPU time of LeNet was measured on SUN SPARCstation 1 [2], while the CPU time of the M^3 network was measured on SUN Ultra 30.

Classifiers	Error rates (%)		CPU time (sec.)	
	Training	Test	Max.	Total
LeNet	1.1	4.3	259200	259200
M^3	0.0	5.0	48	9401

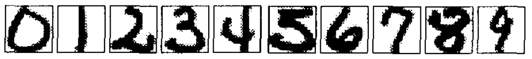


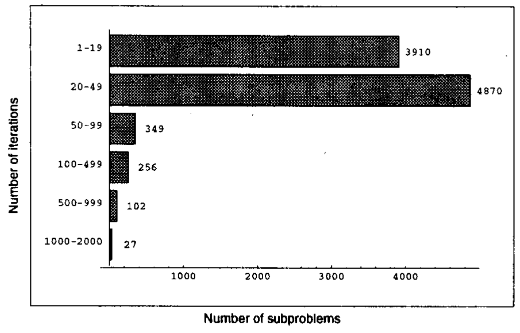
Figure 5: Digits that were segmented from handwritten ZIP codes.

In [7], LeCun, *et al.*, reported that three days were required for training a five-layer feedforward neural network (LeNet) on the handwritten digit recognition problem[1]. In this simulation, the original problem is decomposed into 9514 subproblems randomly. The number of training data in each of the subproblems is about 100. To learn these subproblems, 9514 three-layer MLPs are selected. Each of the MLPs has five hidden units. All of the MLPs were trained by the conventional backpropagation algorithm. The numbers of iterations and CPU times (sec.) required for training the 9514 modules are shown in Figs. 6(a) and 6(b), respectively. From Fig. 6(b), we see that each of 7372 subproblems can be learned within two seconds. The maximum CPU time (see Table 1) for learning a single subproblem is about 48 seconds. This means that to solve the handwritten digit recognition problem requires only 48 seconds, instead of three days, if a complete parallel computer is used. The total CPU time used for learning all 9514 subproblems and the performance of the M^3 are also shown in Table 1.

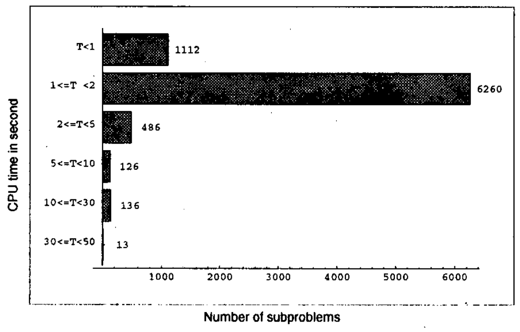
5 Conclusions

In this paper we have presented a massively parallel and modular learning framework for brain-like computers. The advantages of the framework over the

[1]In [7], 7291 handwritten digits and 2549 printed digits were used as training data, while only 7291 handwritten digits were used here.



(a)



(b)

Figure 6: The numbers of iterations (a) and CPU times (b) required for training the 9514 three-layer MLPs.

existing neural network models are its high modularity and parallelism, good responsiveness, plasticity, and scalability. We have demonstrated that the proposed framework may provide us with a simple model for building specific brain-like computers for pattern recognition. The theory, architecture, and learning algorithms for brain-like computers are still in their infancy. In the future work, we would like to investigate the framework theoretically, perform simulations on real-world, large-scale problems, and implement the framework in hardware.

References

[1] G. S. Almasi and A. Gottlieb, *Highly Parallel Computing*, 2nd Edition, The Benjamin/Cummings Publishing Company, Inc. 1994.

[2] M. A. Arbib, : "Plasticity in development and learning", *The Handbook of Brain Theory and Neural Networks*, M. A. Arbib Ed., pp. 53-55, MIT Press, 1995.

[3] G. A. Carpenter and S. Grossberg, "A massively parallel architecture for a self-organizing neural pattern recognition machine", *Computer Vision, Graphics, and Image Processing*, vol. 37, pp. 54-115, 1987.

[4] M. Ichikawa and G. Matsumoto, " Towards to developing brainway computers (in Japanese)", *Bit*, vol. 30, no. 10, pp. 2-8, 1998 - vol. 31, no. 4, pp. 83-91 1999.

[5] M. Garzon, *Models of Massive Parallelism*, Springer, 1995.

[6] K. Hwang, *Advanced Computer Architecture: Parallelism, Scalability, Programmability*, MiGraw-Hill, Inc., 1993.

[7] Y. Le Cun, Y. B. Boser, J. S. Denker, D. Henderson, R. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten ZIP code recognition", *Neural Computation*, vol. 1, no. 4, pp. 541-551, 1989.

[8] J. H. Lin and J. S. Vitter, "Complexity results on learning by neural nets", *Machine Learning*, vol.6 , pp211-230, Kluwer Academic Publishers, Boston, 1991.

[9] B. L. Lu and M. Ito, "Task decomposition based on class relations: a modular neural network architecture for pattern classification", *Biological and Artificial Computation: From Neuroscience to Technology, Lecture Notes in Computer Science*, J. Mira, R. Moreno-Diaz and J. Cabestany, Eds., vol. 1240, pp. 330-339, Springer, 1997.

[10] B. L. Lu and M. Ito, "Task decomposition and module combination based on class relations: a modular neural network for pattern classification", accepted for publication in *IEEE Trans. Neural Networks*.

[11] B. L. Lu and M. Ito, "A plastic modular neural network architecture for pattern recognition", *Proc. of 1997 Annual Conference of Japanses Neural Network Society*, pp. 52-53, 1997.

[12] G. Matsumoto, " The brain and brainway computer", *Proc. of the Fifth International Conference on Neural Information Processing*, pp. 1185-1189, Kitakyushu, Japan, 1998.

[13] T. Omori, "Computational perspective of brain-like computers (in Japanese)", *The Brain & Neural Networks*, vol. 5, no. 4, pp. 194-202, 1998.

[14] D. N. Osherson, S. Weinstein, and M. Stob, "Modular Learning", *Computational Neuroscience*, E. L. Schwartz, Ed., MIT Press, 1990.

[15] M. J. Tovee, *The Speed of Thought: Information Processing in the Cerebral Cortex*, Springer, 1998.

[16] N. Tsukahara, *Plasticity of the Brain and Memory* (in Japanese), Kinokuniya Book Store, 1987.