Appeared in *Proceedings of Forth International Conference on Artificial Neural Networks*, pp.92-97,
Churchill College, University of Cambridge, UK, 26-28 June 1995.

A PARALLEL AND MODULAR MULTI-SIEVING NEURAL NETWORK ARCHITECTURE FOR CONSTRUCTIVE LEARNING

B.-L. Lu [1], K. Ito [1,2], H. Kita [3], and Y. Nishikawa [3]
[1]The Institute of Physical and Chemical Research, Japan [2]Toyohashi Univ. of Tech., Japan [3]Kyoto Univ., Japan

ABSTRACT

In this paper we present a parallel and modular multi-sieving neural network (PMSN) architecture for constructive learning. This PMSN architecture is different from existing constructive learning networks such as the cascade correlation architecture. The constructing element of the PMSNs is a compound modular network rather than a hidden unit. This compound modular network is called a sieving module (SM). In the PMSN a complex learning task is decomposed into a set of relatively simple subtasks automatically. Each of these subtasks is solved by a corresponding individual SM and all of these SMs are processed in parallel.

INTRODUCTION

In the last several years many constructive learning algorithms have been proposed such as the tiling algorithm by M. Mezard and J. P. Nada [9], the cascade correlation architecture by S. Fahlman and C. Lebiere [3], and the extentron algorithm by P. T. Baffers and J. M. Zelle [1]. An important advantage of these algorithms over the back-propagation algorithm is that they can grow as learning proceeds, and therefore, it is not necessary for us to select a suitable network size in advance. In addition, these algorithms improve both learning accuracy and speed. However, these algorithms suffer from two drawbacks that limit their usefulness. Firstly, networks generated by these algorithms are monolithic. Thus even few simple modifications of the learning tasks will require all the trained network parameters to be adjusted. Secondly these monolithic networks tend to be deep with many hidden layers due to the methods of interconnection of each hidden unit. This requires complex learning tasks. Hence, learning speed and response time degrade as the network grows.

In this paper, we present a *parallel and modular multi-sieving neural network* (PMSN) architecture for constructive learning. This PMSN architecture is different from existing constructive learning networks such as the cascade correlation architecture. The constructing element of the PMSNs is a compound modular network rather than a hidden unit. This compound modular network is called a sieving module (SM). The PMSN architecture was developed for implementing the *multi-sieving learning* (MSL) algorithm, a constructive learning algorithm proposed in our earlier work [7, 8]. The basic idea behind MSL is the multi-sieving method. Patterns are classified by a rough sieve at the beginning and they are re-classified further by finer ones in subsequent stages. Fig. 1 illustrates the basic idea behind the multi-sieving learning. In the PMSN architecture, a complex learning task is automatically decomposed into a number of relatively simple subtasks. All the subtasks are then solved by separate individual sieving modules in parallel.

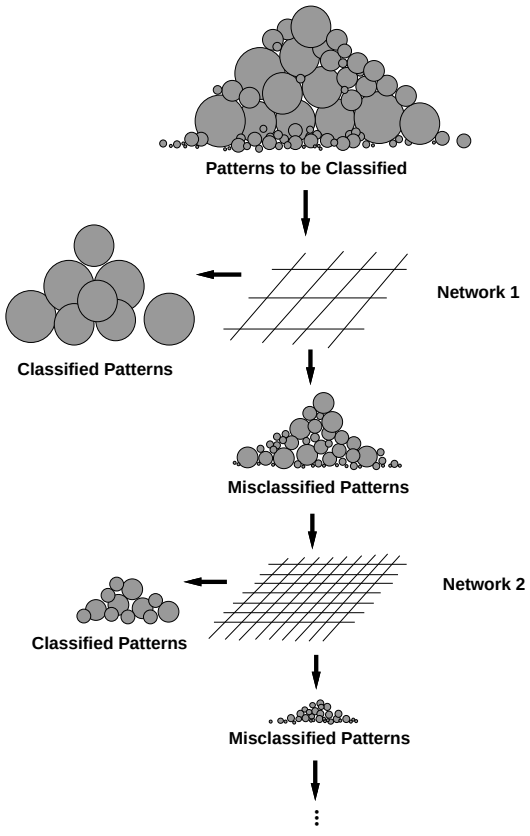


Figure 1: Illustration of the idea behind the multi-sieving learning, where the small balls represent hard patterns to be classified and the big balls represent easy patterns to be classified.

PMSN ARCHITECTURE

In this section, we will first present two different SM structures. We will introduce an output representation scheme used by a recognition network in each SM and then we will define three different kinds of actual outputs. Finally, we will describe the output control circuits and the priority in the SMs.

Sieving Modules

The block diagram of the PMSN architecture is illustrated Fig. 2. The constructing element of this architecture is the sieving module. A sieving module in the PMSN may take one of the two forms according to the learning task, i.e., RC-form or R-form as shown in Figs. 3(a) and (b), respectively. The RC-form sieving module consists of a recognition network RN_i, a control network CN_i, an output judgement unit OJU, an AND gate, two OR gates, and a logical switch. The R-form sieving module is similar to the RC-form, exclusive of the control network and the AND gate.

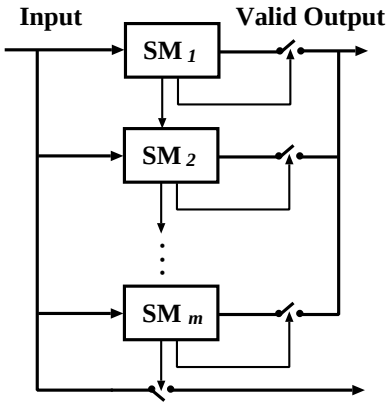


Figure 2: The block diagram of the PMSN architecture.

Output Representation Scheme

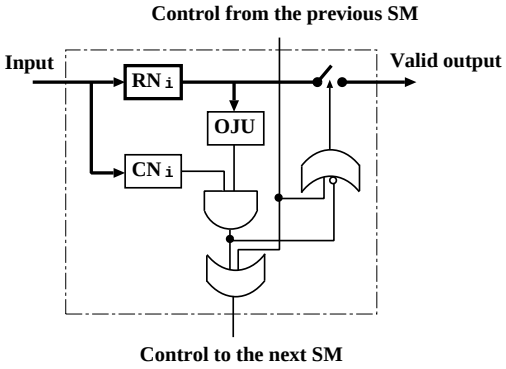
Various output representation schemes can be used for RN_i, such as 1-out-of-N coding [4], binary coding and Gray coding [2]. We use the 1-out-of-N coding method for RN_i throughout this paper. For $p+1$ output classes, we use p or $p+1$ output units. For the kth recognition network RN_k, a desired output pattern $\bar{\boldsymbol{x}}_i^{\mathcal{O}} = \{\bar{x}_{i1}^{\mathcal{O}}, \bar{x}_{i2}^{\mathcal{O}}, \cdots, \bar{x}_{iN_k}^{\mathcal{O}}\}$ must satisfy one of the following rules:

$$\forall j(\bar{x}_{ij}^{\mathcal{O}} < x_{\mathrm{low}}^{\mathcal{O}}) \tag{1}$$

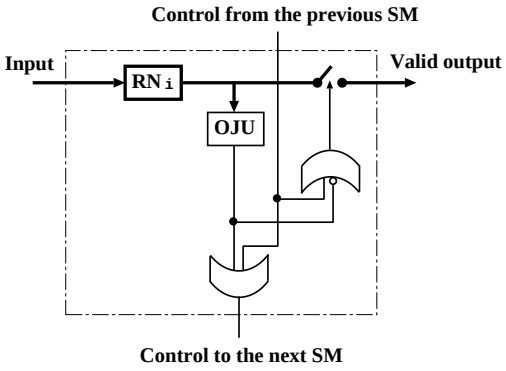
$$\exists j(\bar{x}_{ij}^{\mathcal{O}} > x_{\mathrm{high}}^{\mathcal{O}}) \wedge \forall l\, {}_{l\neq j}(\bar{x}_{il}^{\mathcal{O}} < x_{\mathrm{low}}^{\mathcal{O}}) \tag{2}$$

for j and $l \in B_k$

where $B_k = \{1, 2, \cdots, N_k\}$, N_k is the number of output units in RN_k, and $x_{\mathrm{low}}^{\mathcal{O}}$ and $x_{\mathrm{high}}^{\mathcal{O}}$ represent the low and high bounds for the outputs, respectively. For example, three binary output-units can only represent four valid outputs as follows: $(0,0,0)$, $(0,0,1)$, $(0,1,0)$ and $(1,0,0)$. Other four outputs, $(0,1,1)$, $(1,0,1)$, $(1,1,0)$ and $(1,1,1)$, are considered to be invalid. For a given input pattern $\bar{\boldsymbol{x}}_i^{\mathcal{I}}$, RN_k may generate three kinds of actual outputs:



(a)



(b)

Figure 3: Two forms of sieving modules: the RC-form (a) and the R-form (b)

(a) *Valid output*: The valid output is the correct output which satisfies

$$\forall j \mid \bar{x}_{ij}^{\mathcal{O}} - x_{ij}^{\mathrm{RN}_k} \mid < \delta \quad \text{for } j \in B_k\,, \tag{3}$$

where $\bar{x}_{ij}^{\mathcal{O}}$ is the desired output of the jth unit, $x_{ij}^{\mathrm{RN}_k}$ is the actual output of the jth output unit of the kth recognition network RN_k, and δ denotes a tolerance. If the desired outputs are set to 0 or 1, then, $x_{\mathrm{low}}^{\mathcal{O}} = \delta$ and $x_{\mathrm{high}}^{\mathcal{O}} = 1 - \delta$.

(b) *Pseudo valid output*: The recognition network may generate an output which follows the coding rule (1) or (2), but is not correct. We call such an output the *pseudo valid output*:

$$\exists j\, {}_{j\neq h}(x_{ij}^{\mathrm{RN}_k} > x_{\mathrm{high}}^{\mathcal{O}}) \wedge \forall l\, {}_{l\neq j}(x_{il}^{\mathrm{RN}_k} < x_{\mathrm{low}}^{\mathcal{O}})$$
$$\vee\, \forall l(x_{il}^{\mathrm{RN}_k} < x_{\mathrm{low}}^{\mathcal{O}}) \text{ for } j \text{ and } l \in B_k\,, \tag{4}$$

where the desired output of the hth unit satisfies $\bar{x}_{ih}^{\mathcal{O}} > x_{\text{high}}^{\mathcal{O}}$.

(c) *Invalid output*: Otherwise.

For example, if the desired output pattern is (0, 0, 1), $\delta = 0.2$, $x_{\text{high}}^{\mathcal{O}} = 0.8$, and $x_{\text{low}}^{\mathcal{O}} = 0.2$, then, (0.1, 0.1, 0.9) is a valid output, (0.9, 0.1, 0.1) and (0.1, 0.1, 0.1) are two pseudo valid outputs, and (0.9, 0.1, 0.9) is an invalid output.

Output Control Circuits

In order to differentiate among valid, pseudo valid and invalid outputs, the outputs produced by the recognition network are classified and controlled by the output control circuits as drawn by thin lines in Figs. 3(a) and (b). The output control circuit in the RC-form sieving module consists of an output judgement unit OJU, a control network CN, an AND gate, two OR gates, and a logical switch.

(a) The output judgement unit is used to differentiate the invalid output from other two kinds of outputs. OJU generates 1 or 0 according to

$$\mathcal{O}_{\text{OJU},k} = \begin{cases} 1, & \text{if } x_{ij}^{\text{RN}_k} \text{ is a valid or pseudo} \\ & \text{valid output ;} \\ 0, & \text{otherwise,} \end{cases} \tag{5}$$

where $\mathcal{O}_{\text{OJU},\,k}$ is the output of OJU in the kth sieving module. The process to distinguish the invalid outputs from the two other kinds of outputs can be performed independently of the learning task.

(b) The control network is used to differentiate the valid output from the pseudo valid output. Its output is also 1 or 0, which is determined by

$$\mathcal{O}_{\text{CN},k} = \begin{cases} 1, & \text{if } x_{ij}^{\text{RN}_k} \text{ is a valid output;} \\ 0, & \text{if } x_{ij}^{\text{RN}_k} \text{ is a pseudo} \\ & \text{valid output,} \end{cases} \tag{6}$$

where $\mathcal{O}_{\text{CN},k}$ is the output of the control network in the kth sieving module. To differentiate valid outputs from invalid ones depends on the training data presented to the recognition network and its generalization capability. Hence, it must be achieved by learning.

(c) The logical switch works as follows: if its control input is "1", then the data is blocked by it. Otherwise, the data pass through it.

The key idea behind PMSNs is the introduction of priority to each sieving module. The priority is implemented by means of the output control circuits. In PMSNs, SM_i has higher priority than SM_j for $j > i$. For example, if SM_1 produces a valid output for a given input, then the control signal to SM_2 is set to "1", and all the outputs generated by the succeeding sieving modules SM_2 and SM_3 will be blocked, as shown in Fig. 6.

LEARNING ALGORITHM

The multi-sieving learning algorithm starts with a single SM, then it executes the following three phases repeatedly until all the training samples are successfully learned:

(a) In the learning phase the training samples are learned by the current SM.

(b) In the sieving phase the training samples that have been successfully learned are sifted out from the training set.

(c) In the growing phase the current SM is frozen and a new SM is added in order to learn the remaining training samples.

The block diagram of the multi-sieving learning algorithm is illustrated in Fig. 4.

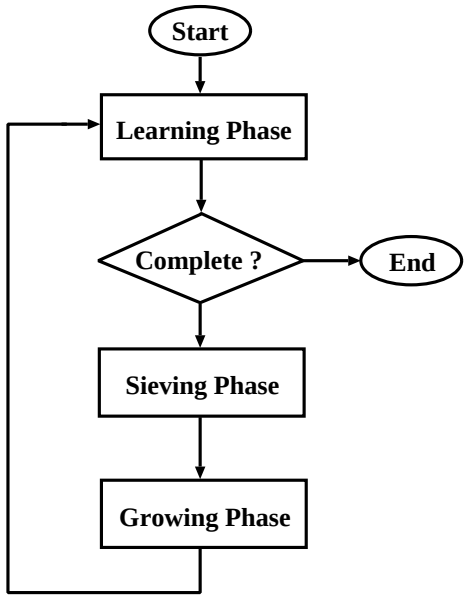


Figure 4: The block diagram of the multi-sieving learning algorithm.

Let T_1 be a set of t_1 training samples:

$$T_1 = \{(\bar{\boldsymbol{x}}_i^{\mathcal{I}},\ \bar{\boldsymbol{x}}_i^{\mathcal{O}}) \mid \text{for } i = 1, 2, \cdots,\ t_1\} \tag{7}$$

where $\bar{\boldsymbol{x}}_i^{\mathcal{I}} \in \mathbf{R}^{N_{\mathcal{I}}}$ and $\bar{\boldsymbol{x}}_i^{\mathcal{O}} \in \mathbf{R}^{N_{\mathcal{O}}}$ are the input and the desired output of the ith sample, respectively. Suppose that the number of iterations for training RN_i is bounded at most by K. The multi-sieving learning algorithm for training PMSNs can be described as follows:

Step 1 : Initially, one recognition network, namely RN_1, is trained on the original set T_1 up to K iterations. Let $m = 1$, and proceed to the following steps.

Step 2 : Compute the number of valid outputs, $N_{\mathrm{vo},m}$, and the number of pseudo valid outputs, $N_{\mathrm{pvo},m}$, according to Eqs. 3 and 4, respectively.

Step 3 : If $\sum_{i=1}^{m} N_{\mathrm{vo},i} = t_1$, i.e., if all t_1 samples are learned by the multi-sieving network, then the training is completed.

Step 4 : If $N_{\mathrm{pvo},m} = 0$, i.e., if there is no pseudo valid output, then the control network is unnecessary. Go to Step 6.

Step 5 : If $N_{\mathrm{pvo},m} > 0$, i.e., if there exist $N_{\mathrm{pvo},m}$ pseudo valid outputs, then a control network, namely CN_m, is selected and trained on the set of $N_{\mathrm{vo},m} + N_{\mathrm{pvo},m}$ samples S_m until all of the samples are classified correctly[1].

$$S_m = \{(\bar{\boldsymbol{x}}_i^{\mathcal{I}},\ \bar{x}_i^{\mathcal{O}}) \mid \text{for } i = 1, 2, \cdots,\ N_{\mathrm{vp},m}\}$$

where $N_{\mathrm{vp},m} = N_{\mathrm{vo},m} + N_{\mathrm{pvo},m}$, $\bar{\boldsymbol{x}}_i^{\mathcal{I}} \in \mathbf{R}^{N_{\mathcal{I}}}$ is the input whose output is a valid or pseudo valid output, and $\bar{x}_i^{\mathcal{O}} \in \mathbf{R}^1$ is the desired output which is determined by

$$\bar{x}_i^{\mathcal{O}} = \begin{cases} 1, & \text{if the actual output of } \bar{\boldsymbol{x}}_i^{\mathcal{I}} \text{ is} \\ & \text{a valid output;} \\ 0, & \text{otherwise} \end{cases}$$

Step 6 : Freeze all of the parameters of RN_m and CN_m (if it exists), remove $N_{\mathrm{vo},m}$ samples which have been successfully classified by RN_m from T_m, and create a new training set consisting of t_{m+1} ($t_{m+1} = t_m - N_{\mathrm{vo},m}$) samples T_{m+1} ($T_{M+1} \subset T_m$), which are not classified by RN_m.

Step 7 : If CN_m exists, construct SM_m in the RC-form. Otherwise make SM_m in R-form.

Step 8 : Join SM_m to SM_{m-1} for $m > 1$ in the parallel structure as shown in Fig. 6.

Step 9 : Select RN_{m+1} and train it on T_{m+1} up to K iterations. Let $m = m + 1$ and go back to Step 2.

From the above description, we can see that the number of SMs needed for learning a given task is determined by the MSL algorithm automatically. Obviously, there exists a trade-off between the learning capability of RN_m and the number of iterations for training RN_m.

Suppose that a PMSN consists of m sieving modules SM_1, SM_2, and SM_m, the processing time required by SM_i is τ_i, and the processing time used by all the output control circuits in the PMSN is much less than the shortest processing time required by a sieving module in the PMSN. From the PMSN architecture which is illustrated in Fig. 2, we can see that if m is a reasonable number, the response time of the PMSN for recognizing an input approximates to $\max\{\tau_1,\ \tau_2,\ \cdots,\ \tau_m\}$, i.e., the longest processing time required by a sieving module in the PMSN. Consequently, the response time of the PMSN for recognizing an input is almost constant since it is independent of the number of sieving modules in the PMSN.

[1] It should be noted that the control network CN_m always learns the classification of the valid and the pseudo valid outputs successfully, by assumption

SIMULATION RESULTS

In order to illustrate the decomposition of learning, the parallel processing and the modification to trained networks performed by PMSNs, two simulations are carried out on the "two-spirals" problem [5] as shown in Figs. 5(a) and 8(a). In the following simulations, the structure of RN_i and CN_i are chosen to be the multilayer quadratic perceptron [6]. The backpropagation algorithm is chosen as the training algorithm [10].

Decomposition of Learning Task

The training inputs of the "two-spirals" problem consists of 194 $\langle x, y \rangle$ points at which RN_i should output 0's or 1's as shown in Fig. 5(a). Each RN_i in the SMs has two input, five hidden units and one output unit. Training of RN_i is stopped after 10000 iterations if the total error between the desired and the actual outputs cannot be less than the given value, 0.1. The "two-spirals" problem is learned by a PMSN according to the multi-sieving learning algorithm. After learning, we obtain a PMSN with three R-form sieving modules as shown in Fig. 6(a). The input patterns that are learned by the first, the second and the third SMs are illustrated in Figs. 5(b) through (d), respectively. The response plots of the PMSN and the first through the third SMs are illustrated in Figs. 7(a) through (d), respectively. From Figs. 7(b) through (d), it is clear that the "two-spirals" problem is decomposed into three relatively manageable subtasks and each subtask is solved by a corresponding sieving module as shown in Fig. 6(a). From Fig. 6(a), we can see that all the sieving modules are processed almost in parallel.

Modification to Trained Network

For most of the existing constructive learning networks such as the cascade correlation architecture, even with few simple modifications to the learning task, all the parameters of the trained network must be re-adjusted. This example illustrates how to implement modification to trained PMSNs without de-

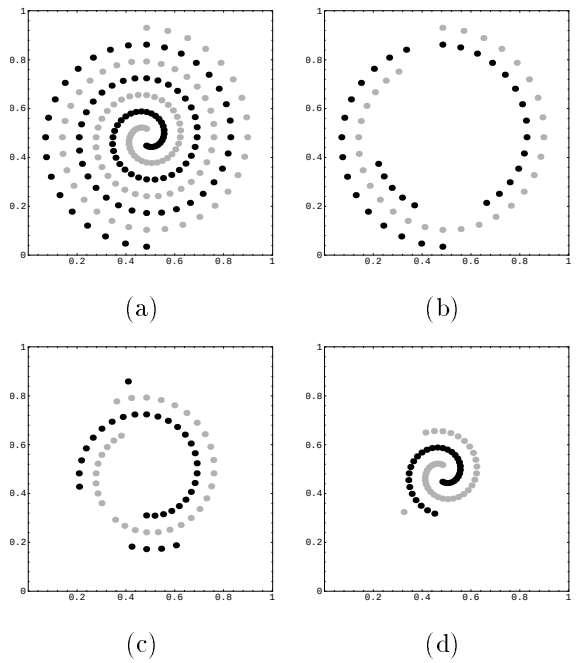


Figure 5: The input patterns of the "two-spirals" problem (a), the input patterns classified by the first SM (b), the second SM (c), and the third SM (d), respectively. For black and grey points, the RN_m is required to generate output 0 and 1, respectively.

stroying any parameter of the trained recognition networks. In our simulated problem we will try to modify the trained PMSN_{old}, shown in Fig. 6(a), to recognise the updated two-spirals problem (see Fig. 8(a)).

We present the 16 updated training inputs to RN_1 of the $\mathrm{PMSN}_{\mathrm{old}}$, we obtain $N_{\mathrm{pvo},1} = 16$ and $N_{\mathrm{vo},1} = 0$, that is, no updated training input is generalized correctly by RN_1. In a similar way, we obtain $N_{\mathrm{pvo},2} = 16$, $N_{\mathrm{vo},2} = 0$, $N_{\mathrm{pvo},3} = 11$ and $N_{\mathrm{vo},3} = 5$. Here $N_{\mathrm{vo},3} = 5$ means that there are 5 updated training inputs which are generalized properly by RN_3. In order to learn the 16 updated training inputs, three control networks CN_1, CN_2, and CN_3 are trained and added to the $\mathrm{PMSN}_{\mathrm{old}}$, and the sieving modules in the $\mathrm{PMSN}_{\mathrm{old}}$ are changed into RC-form. The 11 updated training inputs which can not be generalized correctly by RN_1, RN_2 and RN_3 are learned by a new recognition network RN_4. The modified PMSN with four sieving modules, namely $\mathrm{PMSN}_{\mathrm{new}}$, is illustrated in Fig. 6(b). The response plot of $\mathrm{PMSN}_{\mathrm{new}}$ is illustrated in Fig. 8(b). Comparing Fig. 7(a) with Fig. 8(b), we can see that the 16 training inputs, which have already been learned by $\mathrm{PMSN}_{\mathrm{old}}$, are updated by modifying $\mathrm{PMSN}_{\mathrm{old}}$ without adjusting the parameters of RN_1, RN_2, and RN_3 in $\mathrm{PMSN}_{\mathrm{old}}$.

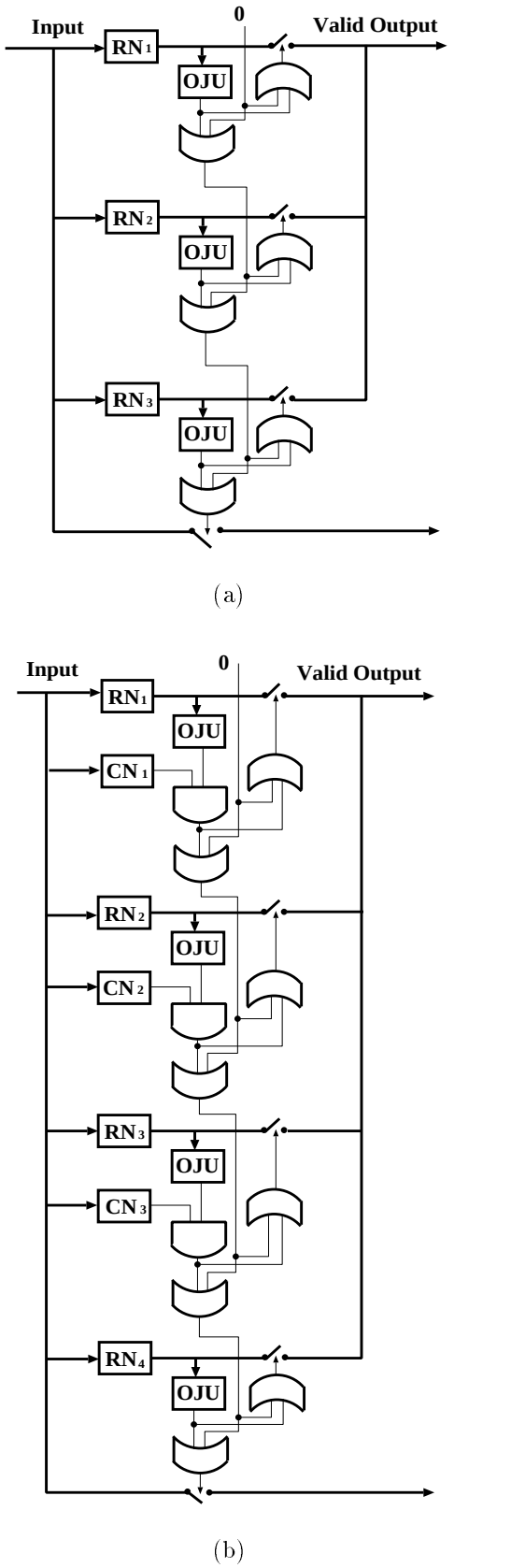
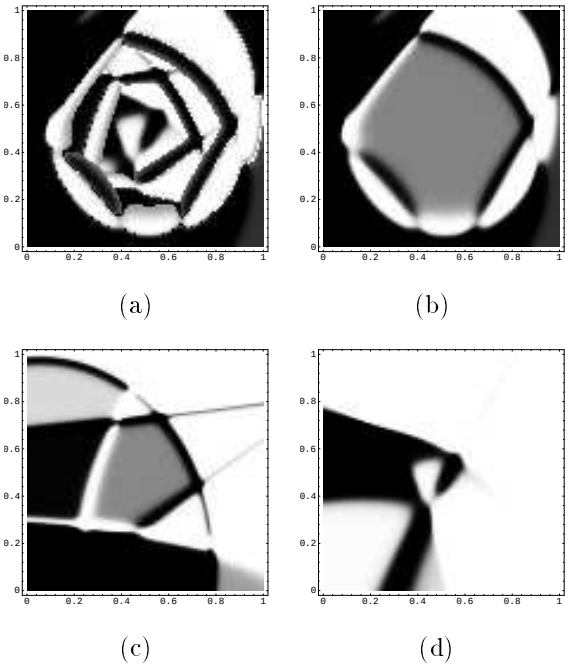


Figure 6: Two PMSNs for learning the "two-spirals" problem (a) and the updated "two-spirals" problem (b), where the control signal to the first sieving modules is set to "0"



(a) (b)

(c) (d)

Figure 7: Response plots of the PMSN (a), the first SM (b), the second SM (c), and the third SM (d). Black and white represent output of "0" and "1", respectively, and grey represents intermediate value.

CONCLUSION

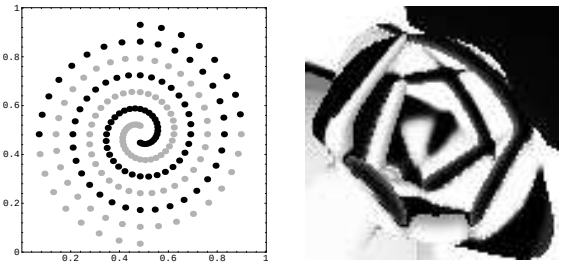
We have presented a parallel and modular multi-sieving network architecture for constructive learning. This architecture offers two major advantages over existing constructive learning networks such as the cascade correlation architecture. Firstly, this architecture is a modular structure and the trained modular networks can be modified without adjusting all the parameters to learn an updated task. Secondly, all of these modules are connected in parallel. Hence, problems which associate with "deep network" does not exist with this architecture. The response time is almost independent of the number of modules.

ACKNOWLEDGEMENT

The authors are grateful to Kenneth H. L. Ho for reading the manuscript.

REFERENCES

[1] Baffers, P. T. and Zelle, J. M., 1992, "Growing layers of perceptrons: Introducing the extentron algorithm", *Proc. of International Joint Conference on Neural Networks*, pp. II-392-II-397, Baltimore, MD.



(a) (b)

Figure 8: The updated two-spirals problem in which 16 training data are different from the original two-spiral problem (a), and the response plot of the modified PMSN (b).

[2] Ersory, O. K. and Hong, D. , 1990, "Parallel, self-organizing, hierarchical neural networks", *IEEE Trans. on Neural Networks*, vol. 1, no.2, pp. 167-178.

[3] Fahlman, S. and Lebiere, C., 1990, "The cascade-correlation learning architecture", *Report CMU-CS-90-100*, Carnegie-Mellon University.

[4] Hechi-Nielsen, R., 1990, "Neurocomputing", Addison-Wesley, Reading, Mass.

[5] Lang, K. and Witbrock, M., 1988, "Learning to tell two spirals apart", *Proc. of 1988 Connectionist Models Summer School*, pp. 52-59, Morgan Kaufmann.

[6] Lu, B.-L., Bai, Y., Kita, H. and Nishikawa, Y., 1993, "An efficient multilayer quadratic perceptron for pattern classification and function approximation", *Proc. of International Joint Conference on Neural Networks*, vol. 2, pp. 1385-1388, Nagoya.

[7] Lu, B.-L., 1994, "Architectures, learning and inversion algorithms for multilayer neural networks", *Ph. D. thesis*, Dept. of Electrical Engineering, Kyoto University.

[8] Lu, B.-L., Kita, H. and Nishikawa, Y., 1994, "A multi-sieving neural network architecture that decomposes learning tasks automatically", *Proceedings of IEEE Conference on Neural Networks*, pp. 1319-1324, Orlando, FL.

[9] Mezard, M. and Nada, J. P., 1989, "Learning in feedforward layered networks: the tiling algorithm", *Journal of Physics A*, vol. 22, pp. 2191-2203.

[10] Rumelhart, D. E., Hinton, G. E. and Williams, R. J., 1986, "Learning representation by backpropagation errors", *Nature*, vol. 323, pp. 533-536.