International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:2, No:6, 2008

Effective Sonar Target Classification via Parallel Structure of Minimal Resource Allocation Network

W.S. Lim, and M.V.C. Rao

***Abstract*—**In this paper, the processing of sonar signals has been carried out using Minimal Resource Allocation Network (MRAN) and a Probabilistic Neural Network (PNN) in differentiation of commonly encountered features in indoor environments. The stability-plasticity behaviors of both networks have been investigated. The experimental result shows that MRAN possesses lower network complexity but experiences higher plasticity than PNN. An enhanced version called parallel MRAN (pMRAN) is proposed to solve this problem and is proven to be stable in prediction and also outperformed the original MRAN.

***Keywords*—**Ultrasonic sensing, target classification, minimal resource allocation network (MRAN), probabilistic neural network (PNN), stability-plasticity dilemma.

I. Introduction

MANY efficient neural network techniques have been widely used in today's robotic and automation technology over traditional statistical techniques. Pattern classification has become an important topic for robotics research in many applications [1]. These classifiers are capable of predicting the shape of objects or obstacles surrounding the robot by means of processing the input data received from numerous types of sensors or detectors on the robot. They also make less assumption than the traditional statistical methods and hence robustness can certainly be achieved even though the input data is generated through a non-linear system.

There are various types of methods for generating the data of objects surrounding a robot. One reason why sonar or radar systems are favored is because of their cost-effectiveness besides their capability of emulating the remarkable perception and pattern recognition behavior of humans and animal [2]–[4]. A comparison between neural networks and standard classifiers for radar-type emitter detection is given by Wilson [5]. Another acoustic imaging system that combines holography and multi-layer feed-forward neural networks for three-dimensional object recognition is proposed in [6]. In [7], a neural network is used to recognize three-dimensional cubes and tetrahedrons by means of sonar. Neural networks have also been employed to classify the sonar returns from undersea targets [3], [8] for providing the sea floor contour.

Generally, different objects with different curvatures will reflect the sonar signals at different angles and intensity. These signals supply the data of the distance between the target and the detector that serve as the input to the neural networks. Many researchers have employed different kinds of target differentiation algorithms in an earlier work [9]. An RBF network learning algorithm called minimal resource allocation network (MRAN) was developed by Lu YingWei *et al*. [10], [11] as a sequential learning algorithm that employs a scheme for adding and pruning RBFs hidden neurons, so as to achieve a parsimonious network structure. However, RBF type of network is always found to experience the stability-plasticity problem due to its low capability in maintaining the history after adapting to a new environmental change [12].

In this paper, we investigate the use of neural networks in processing the sonar signals reflected by different targets for indoor environments. It describes how MRAN algorithm performs on pattern classification of various targets. A statistical comparison is made between its performance and the one of Probabilistic Neural Network (PNN). Here the robustness and plasticity of the MRAN neural network are tested and compared in two different stages. In the first approach, the comparison result shows that the original MRAN has a high level of plasticity that deteriorates its capability in maintaining the neurons' weights of the previously encountered patterns. Generally, its weights change rapidly adapting towards the most recent received data and hence eventually causing the network to recognize only that particular pattern. In the second approach, an enhanced version that integrates multiple MRANs together is implemented for processing the networks in parallel in order to reduce the plasticity problem suffered in the single MRAN. Generally by assigning one MRAN to handle each pattern to be classified, the performance is proven to have improved drastically and the neural network is no longer plastic and unstable.

II. Fundamentals of Sonar Sensing

The term 'ultrasonic' applied to sound refers to anything above the frequencies of audible sound, and nominally includes anything over 20 kHz. Sonar sensing is commonly used in communication and navigation. Ultrasonic sound can be produced by transducers that operate either by the piezoelectric effect or the magneto-strictive effect.

Researchers have shown that, by proper sonar transducer selection, both the wide and narrow areas can be covered. Besides, sonar sensors are impervious to external disturbances such as vibration, infrared radiation, ambient noise and EMI radiation makes sonar sensing suitable for many applications. Moreover, dust, dirt, or high-moisture environment has very little effect on the performance of the sonar sensors. Since sound can be timed from when it leaves the transducer to when it returns, distance measuring can be achieved. Precise distances of object from the sensor are measured via time intervals between transmitted and reflected echo of the ultrasonic sound. This is commonly known as the time-of-flight (TOF). The distance d between the sensor and the object can be obtained by $d = vt / 2$ when the echo amplitude first exceeds a preset level back at the receiver at time t. Here, v is the speed of sound in air.

The target primitives modeled in this study are wall, corner, and edge (Fig. 1). In our system, a commercially available robot simulator, Amigobot modelled P2AT is employed for data collection. Five identical acoustic sonar transducers on the front side of the robot were utilized as shown in Fig. 2. Each transducer can operate both as a transmitter and a receiver and detects echo signals reflected from targets within its own sensitivity region.

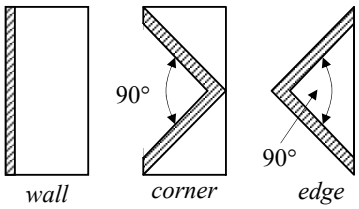

Fig. 1 Cross sections of the target primitives differentiated in this work

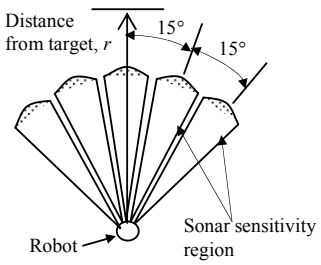

Fig. 2 Sensitivity region of an array of ultrasonic transducers in P2AT robot

III. Target Classification with Neural Network

Till today, many research works have been done to encounter the stability-plasticity problems in neural networks such as in [13]–[16] for pattern classifications and function approximation. This dilemma can be stated as a series of questions. How can a learning system remain adaptive (plastic) in response to a significant input, yet remain stable in response to an irrelevant input? How does the system know to switch between its plasticity and stable modes? How can the system retain previously learned information while continuing to learn new things?

A major restriction on traditional artificial neural networks is that the approximation capability will be frozen after the completion of training process. This results in a gradual degradation of estimation performance when applied to non-stationary environment. In solving this problem, the key challenge is the requirement to maintain a compromise between robustness toward interference and the adaptability to environment changes. Until recent decades, artificial neural networks have been providing us with many successful evidences on the application of multivariate and nonlinear time series prediction [17], [18]. However, traditional neural networks always perform unsatisfactorily in non-stationary cases because of a deficiency of feedback mechanism to accommodate the input distribution changes. Missing this feedback mechanism, the common way to adapt the distribution skewness is to completely clearing the existing network memory and begin with a new training set including information about current changes.

The target differentiation algorithm used in earlier works like in [19] is reviewed. It has given useful ideas of how to differentiate targets by means of their shapes and radius of curvature. In this paper, two types of neural network algorithms are introduced, namely Minimal Resource Allocation Network (MRAN) and Probabilistic Neural Network (PNN) for classifying the three primitive targets. Since PNN is just serving as a comparison tool for experimental purposes, its algorithm and architecture would not be further explored in detail.

A. Minimal Resource Allocation Network

The structure of a basic RBF network with Gaussian functions as its radial basis functions (similar to the structure used in MRAN) can be seen in Fig. 3. It can be observed that to construct a neural network like this, four types of parameters are required. They are the number of hidden neurons h in the network, the center positions μ's for all the hidden neurons h in the network, the corresponding width values σ's for Gaussian function and the connection weights α's between the hidden layer and the output layer.

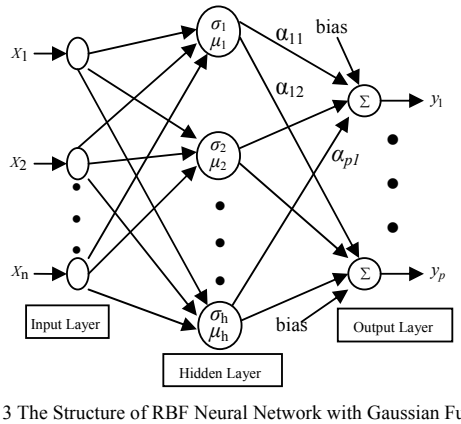

Fig. 3 The Structure of RBF Neural Network with Gaussian Function

The sequential learning MRAN algorithm employs a scheme for adding and pruning RBF hidden neurons, so as to

achieve a parsimonious network structure. In MRAN, the network begins with no hidden neurons. The response (output) of a hidden unit to the network input at the i^{th} instant, x_i, can be expressed as follows,

$$\phi_k(x_i) = \exp(-\frac{1}{(\sigma_k^i)^2}\left\|x_i - \mu_k^i\right\|^2), (k = 1,..., h) \qquad (1)$$

where μ_k^i is the center vector for the k^{th} hidden unit at i^{th} instant and σ_k^i is the width for the Gaussian function at that time. $\|\ \|$ denotes the Euclidean norm and h indicates the total number of hidden neurons in the network. For networks with multiple outputs $\hat{y}_i$ of p dimensions, the overall network response is a mapping f: $\Re^q \rightarrow \Re^p$, which is

$$\hat{y}_i = f(x_i) = \alpha_0^i + \sum_{k=1}^{h} \alpha_k^i \phi_k(x_i), \qquad (2)$$

where $x_i \in \Re^q$. The coefficient vector α_k^i is the connecting weight vector of the k^{th} hidden unit to output layer, which is in the vector form of $\alpha_k^i = \left[\alpha_{1k}^i,..., \alpha_{lk}^i,..., \alpha_{pk}^i\right]^T$. Thus, the coefficient matrix of the network can be expressed as $A_{p\times h}^i = [\alpha_1^i,...\alpha_k^i,..., \alpha_h^i]$. α_0^i is the bias vector which is $\alpha_0^i = [\alpha_{10}^i,..., \alpha_{l0}^i,..., \alpha_{p0}^i]^T$. As each training data pair (input and output) is received the network builds itself up based on two growth criteria, (3) and (4).

$$\|x_i - \mu_{nr}^i\| > \varepsilon_i \qquad (3)$$

$$\|e_i\| = \|y_i - \phi_k(x_i)\| > e_{min} \qquad (4)$$

where μ_{nr}^i is the center (of the hidden unit) which is closest to x_i (the input received). e_i is the calculated error, the difference between output received, y_i and the network output, $\phi_k(x_i)$. ε_i, e_{min} are thresholds to be selected appropriately.

The algorithm adds new hidden neurons or adjusts the existing network parameters according to the training data received. The algorithm also incorporates a pruning strategy that is used to remove hidden neurons that do not contribute significantly to the output. Consequently, this algorithm reduces the network complexity compared to other methods (like PNNs). A complete learning flow diagram of MRAN is given in Fig. 4. Other research work on function approximation by MRAN can also be seen in [20]. Some other ideas of growing and pruning as in [28] and [29] are also referred.

B. Parallel MRAN (pMRAN)

The concept of multiple MRANs has been studied and employed to enhance the performance of the original single MRAN network from stability and plasticity point of view. Initial works have been carried out in this research to determine the pattern classification performance of a single MRAN, and the experimental result shows that it is unstable and plastic, especially in sequential learning [21]. Analysis from the result proves that it stands as a good classifier in terms of quick learning capability and adaptation with low network complexity, but unfortunately when more new patterns are encountered, the old memory deteriorates. Consequently, the single MRAN only classifies well for the pattern most recently encountered. The challenge taken in solving this problem is to stabilize the network in such a way that the network complexity (in terms of number of neurons generated) are still maintained in an acceptable range, and MRAN is of one best choice since it incorporates a pruning strategy. Hence, the concept of multiple MRAN is adopted for this case, where each pattern is assigned to a single MRAN, and the classification accuracy results obtained in this work are promising and favorable.

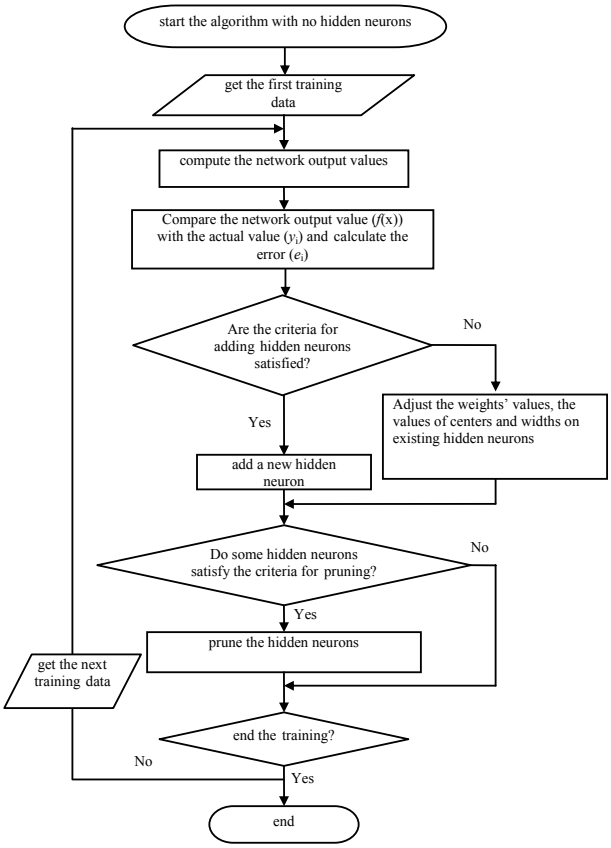

Fig. 4 Flow diagram for MRAN learning algorithm

In this section, the structure and the process flow of the parallel MRAN (as shown in Fig. 5) are discussed. The number of MRAN classifiers used in the network is decided solely by the number of patterns or classes to be classified, meaning that each class is handled by one MRAN. In this way, each MRAN is trained only by the same cluster of information and consequently that network is restricted to recognize only that particular pattern. Firstly, the training data is distributed according to class type by a *pattern separator* before it is transferred to each MRAN classifier respectively. This can be easily realized since each training data comes with its desired output. For three different target types in this experiment, the

codes 001, 010 and 100 are pre-assigned to wall, corner and edge respectively. In the pattern separator, these codes are being identified and the training data pairs (training input and output pair) are then distributed by class. Once all the data is segregated and passed on to the classifiers, the training process is executed in all the classifiers and new neurons are generated according to (3) and (4). The weight (α), center (μ) and width (σ) are three major parameters that represent the Gaussian function of each neuron and these parameters are perpetually tuned to accommodate the incoming training data. Therefore, the values of these parameters are kept within certain specific range for each pattern in the multiple MRANs structure (pMRAN).

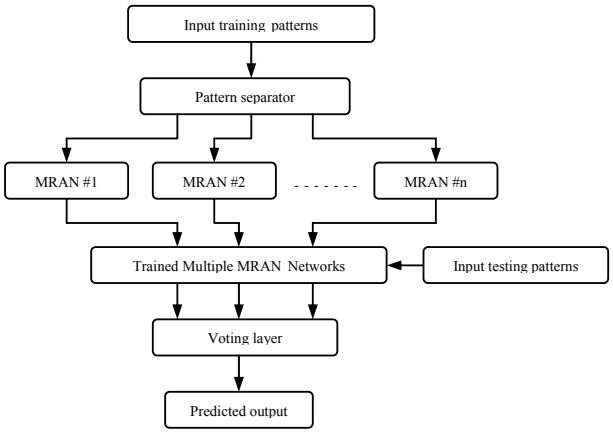

Fig. 5 Parallel MRAN structure

When training is completed, the *trained* pMRAN is tested with new testing data. From this data, each trained classifier calculates and predicts its own output and passes on the information to the *voting layer*. In this stage, we have adopted a classical method of voting where the winner-takes-all in deciding the correct pattern or class. Generally in gaussian nature, the effective gaussian amplitude (G) of a test point (x) is inversely proportional to the distance between the point and the center (μ) of a gaussian function, given by $G(x) = \exp[-(x - \mu)^2 / \sigma^2]$. Typically, when a network detects the pattern of its own, it is very likely to produce a higher output value because the pattern is located nearer to the center(s) of its neuron(s). Whereas the neurons from other classes may comparatively produce lower output value since the pattern is further from their centers. In this sense, we possibly choose the highest output value among the classifiers to indicate that particular classified class.

C. Probabilistic Neural Network

PNN realizes the Parzen-window estimators in feed-forward neural network architecture. For a pattern classification problem of c classes, the hidden activation function (transfer function) w_i (i = 1, 2,…, c) of the network (assuming that each class is represented by one output node) is denoted by the Parzen window function $K(\cdot)$, and those distinct hidden nodes are assigned for different classes in the training process. Then the output y_i of the ith output node for PNN can be expressed as follows:

$$y_i = \sum_{j=1}^{H_i} K\left(\frac{(x - x_j)}{\alpha}\right) \approx p(x / w_i) \qquad i = 1, 2, ..., c \qquad (5)$$

where H_i is the number of hidden nodes of PNN corresponding to the ith class, x_j's are the hidden center vectors of respective jth hidden nodes, which are all the training sample vectors from training sample set for PNN. For a general pattern recognition problem involved in c classes, H_i is just equal to the number N_i of the training samples for the ith class. So, the total number N of training samples for c classes is $N = \sum_i^c H_i$. Generally, the following is always true:

$$H \le N \qquad (6)$$

From (5) and (6), we can see that

- The hidden node number of PNN is just equal to the total number of training samples [22].
- PNN is a self-supervised classifier, in which the nodes do not need any external supervised signals, based on the class labels of the training samples [23].
- PNN is also a directly testing classifier without training (the weights are set to be 0s or 1s), and the decision surfaces among patterns from distinct classes are formed by their conditioned probability density function [24]–[26].

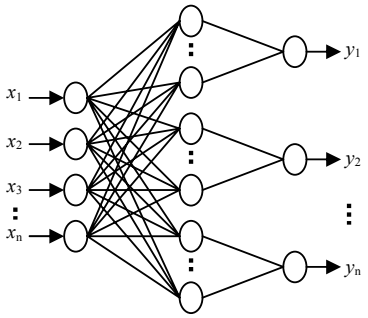

Fig. 6 The structure scheme of the probabilistic neural network

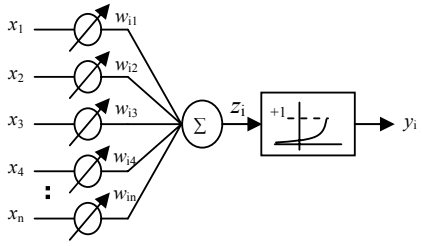

Fig. 7 The structure scheme of one hidden neuron of PNN

IV. Experimental Setup

In our system, a commercially available robot simulator (Fig.8), Amigobot modeled P2AT which incorporates built-in noise and interference to ensemble real physical environment, is employed for data collection. Five identical acoustic sonar transducers on the front side of the robot were utilized, as

shown in Fig. 2. Each transducer can operate both as transmitter and receiver and detect echo signals reflected from targets within its own sensitivity region. On the discrete network location shown in Fig. 9, the robot is positioned at the black dot locations (41 dots in total) for collecting training data while the white dot positions (40 dots in total) are for obtaining testing data. In each position (black or white), 5 echo signals from 5 transducers are collected from over 40 different locations, therefore, providing 600 (5 sensors x 40 locations x 3 target types) training and testing data each. Each collected data is totally independent from one another.

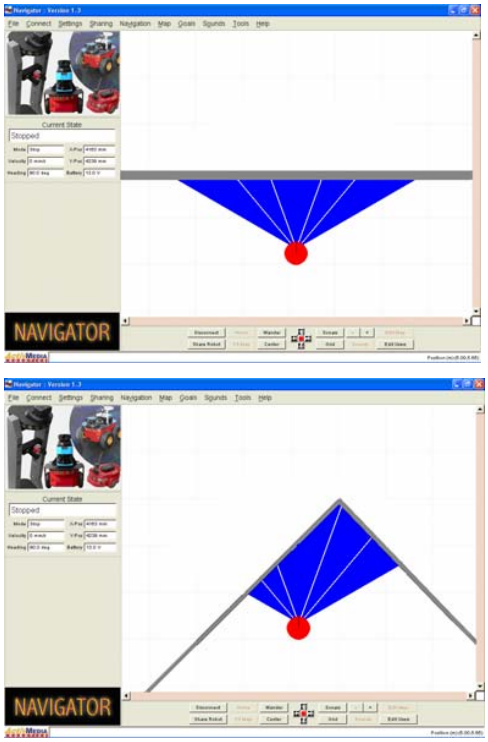

Fig. 8 Amigobot simulation software

Three training methods are employed to judge the robustness and plasticity of the networks, namely the original MRAN, the newly proposed pMRAN and PNN. The main reason for introducing the classical PNN in this work is for comparison purposes due to its high stability behavior. It also confirms that the training and testing data used are appropriate and have certain satisfactory level of correctness. Firstly, the network is trained in a way that it received data collected from one target type, followed by second and then the third target type *in sequence*. In the second method, they are trained by *randomly mixed* data of all three target types. In the third training method, both networks are trained in a similar way to the first method, but it is further extended by *repeating the first* target type training again in sequence. After the training process, the networks are tested to investigate their robustness and stability by data collected at different locations on the discrete network. Each dot is positioned at approximately 10 cm away from its neighboring dots in a square mesh network.

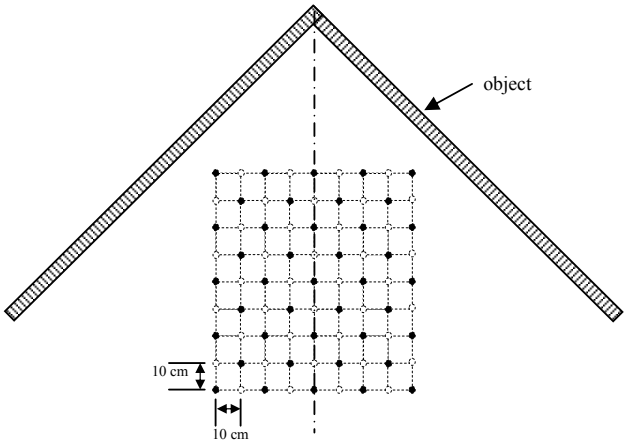

Fig. 9 Discrete network locations. Black-dot: training; White-dot: testing

There are 3 output nodes for MRAN and each of these output nodes corresponds to each type of target. For simplification of calculation, the three outputs are scaled to yield only the value of 0 or 1, where the largest value among the three outputs would be a "1" while the other two remained as "0". The output node with a "1" indicates the type of target detected. In other words, we employed the "winner takes all" method for selecting the network predicted target. This in turn gave every test result in combination of 0 and 1 only (001, 010 or 100). Hence, the correct target type classification percentages are calculated out from the test results and summarized for every training method. Each training data that the network received consists of 5 inputs (from 5 sensors) representing the estimated distances of the robot from the target, and 3 desired outputs that represent the target type. The three desired outputs were classically *predefined* by three different coding, 001, 010 and 100 representing wall, corner and edge respectively. During training process, neurons in MRAN are generated one by one according to (3) and (4) as stated earlier. When a new neuron is generated, it will hold three different important parameters, namely the weight of the neuron (α), the center location (μ) and width (σ) of its gaussian function. Each set of weight, center and width values contained in each neuron is significant and differs from one to another. Since MRAN is also equipped with efficient pruning strategy, it has the capability of pruning any insignificant neuron(s) throughout the training process. This implies that the values of weight, center and width in each neuron are constantly updated whenever a new neuron is generated, or an existing insignificant neuron is pruned.

All the networks are trained and the testing accuracy is computed for all *three training methods*. Whenever a new target is encountered by MRAN, the network has the tendency to adjust its weights to adapt to the new pattern. The adjustment of the weights may affect on the information of past encountered target types stored in the neurons earlier. Consequently, the most recent target type encountered by the network would most probably be getting the highest correct classification accuracy, whilst the earliest encountered target

type might have possibly been forgotten. In view to this plasticity and stability problem, multiple MRANs are combined together to process the data simultaneously. Regardless of how the input training data is mixed up or distributed, the data will always be segregated by individual class first, and then only fed into their respective network for learning. In pMRAN, the number of MRAN networks used depends on the number of classes or patterns to be classified, where each MRAN handles just one class. This approach restricts the network just to deal with only one target type and the weights developed in the neurons are constraint only to one of a kind. After being trained with three different methods on MRAN, and a parallel training of each class on pMRAN, the networks are tested with the same set of testing data. All network parameters settings are common for both MRAN and pMRAN for the entire training process. The corresponding results of correct classification accuracy for MRAN and pMRAN are shown in Table I. The performance test of the classical PNN is also carried out in this work for comparison purposes.

V. Comparative Analysis and Discussion

In this section we present the results on target differentiation performance of MRAN and pMRAN trained in two different approaches, while using PNN as a comparative tool. All the networks are compared on their accuracy of estimating the correct target, namely wall, corner and edge. As discussed in the previous section, all the data used for testing the networks are different from the data used for training. This is to further ensure that the robustness and stability of the networks are tested comprehensively. In the following Table 1, the numerical values show the percentages of correct target-type classification.

TABLE I
Comparative results among MRAN, PNN and pMRAN
The numerical values show the classification accuracy for each target type

Network used	MRAN			PNN		
Target type	Wall	Corner	Edge	Wall	Corner	Edge
Method 1	0%	19%	100%	97%	93%	91%
Method 2	58.2%	48.7%	52.6%	97%	93%	91%
Method 3	100%	26%	0%	97%	93%	91%

pMRAN (Three independent MRAN networks handling three different classes. Each network is trained with only one class type)		
Wall	Corner	Edge
91%	94%	86%

A. MRAN Performance Study

- In the original MRAN, perfect classification (100%) is obtained for object / target-type that is trained ***last*** in the sequence, i.e. for training Method 1 and 3. This implies the network learns very quickly (only 200 training data for each target type) and able to classify accurately on what it has ***just*** learned.
- In training Method 1, *edge* is the ***last*** batch of training data received by the network, and *edge* is the only target accurately classified where as for *wall* and *corner*, the network failed to classify them accurately.
- The same case holds good for training Method 3, where *wall* data is repeated at the end of the training process, and as expected, *wall* has the perfect classification percentage this time.
- In training Method 2, the network classifies poorly in overall when the training data is randomly mixed.

From the accuracy results, it is observed that MRAN has the tendency to adjust its weights to adapt to a new pattern. The adjustment of the weights has affected on the information of past encountered target types stored in the neurons earlier. As a result, the most recent target type encountered by the network possessed the highest correct classification accuracy, whilst the earliest encountered target type have been forgotten. Apart from that, it is observed that the total number of hidden neurons generated in MRAN for each training method is 20, 19 and 11 in methods 1, 2 and 3 respectively. From the statistical analysis, generally more neurons are generated whenever edge target type is encountered. In method 3, wall data is repeated again to train the network and it is observed that some of the neurons generated earlier for target edge have been pruned and as a result lower number of neurons is obtained in this case.

B. pMRAN performance study

- In pMRAN, the multiple MRANs managed to establish a stable classification performance, where the lowest accuracy i.e. 86% is achieved for *edge*. The highest accuracy achieved is from *corner* i.e. 94%, followed by *wall*, 91%.
- The weights in each neuron generated for each target type are maintained throughout the training process since each MRAN is supplied with only one type of class data. Hence, each network only recognizes the target type that it has learned and produces a strong output to indicate that particular target type.

In the observation of pMRAN, the number of neurons generated in each individual network is much lower than that in MRAN, which is 2, 3 and 5 for target type wall, corner and edge respectively. In overall, only 10 neurons are used in pMRAN and are already sufficient to produce stable performance and high accuracy compared to 11 to 20 neurons in MRAN. The number of MRAN classifiers used in the network is decided solely by the number of patterns or classes to be classified. Therefore, each MRAN is trained only by the same cluster of information and consequently that network is

restricted to recognize only that particular pattern. As mentioned before, the weight (α), center (μ) and width (σ) are three major parameters that represent the Gaussian function of each neuron and these parameters are perpetually tuned to accommodate the incoming training data. Therefore, the values of these parameters are kept within certain specific range for each pattern in the pMRAN structure, without being replaced or eliminated due to other classes incoming patterns.

C. PNN performance study

On the other hand, the results from PNN show that

- The highest classification percentage is from *wall*, followed by *edge* and then *corner*.
- Generally, the classification percentage for all targets achieved the satisfactory level (above 90%). The main reason for introducing the classical PNN in this work is for comparison purposes due to its high stability behavior. It also confirms that the training and testing data used are appropriate and have certain satisfactory level of correctness.

The number of hidden neurons generated in a classical PNN depends on the sample or data size used [22]. In this case, the sample size used is 600 data which consequently causing about 600 hidden neurons being generated. Hence, this large amount of hidden neurons adds to the complexity of the network. However, PNN does not suffer from unstable and plasticity problems because generally it creates a separate neuron for each training sample. These generated neurons are neither pruned nor adjusted and hence the weights of every neuron are maintained. For moderately sized databases this is not a problem, but unfortunately it will be a major drawback for large databases and major applications where it increases the complexity of the network and consequently deteriorates the speed.

VI. Conclusion

The percentage of correct target type classification is high at 100% for the last target trained by the classical MRAN for training method 1 and method 3. This shows that the target type that is most recently trained will give high classification accuracy. Unfortunately, it classifies poorly on those target types that are earlier trained. In other words, the weights carried by the hidden neurons during training process experience rapid change, making the network tend to "forget" what it learned previously. No doubt, it learns very quickly, but it only "remembers" what is trained most recently, thus making the classical MRAN unstable and plastic. Due to this reason, MRAN is seldom applied as pattern classifier in earlier research works. However, modifying the original single MRAN to multiple MRANs that learn in parallel solves the plasticity and stability problems as proven in pMRAN. The result shows consistent prediction for all three targets and generally its performance is close to that of PNN that is well known for its good stability. Besides, the advantage of using MRAN network is its low complexity due to a small number of hidden neurons generated during the training, and is furthermore improved by its incorporated pruning strategy. PNN shows to be a stable network but the disadvantage is, it requires large number of hidden neurons that leads to high network complexity which deteriorates its processing speed compared to MRAN.

References

[1] R. P. Lippman, "An introduction to computing with neural nets", *IEEE ASSP Mag.*, pp. 4 – 22, Apr. 1987.
[2] W. W. L. Au, "Comparison of sonar discrimination – dolphin and artificial neural network*", J. Acoust. Soc. Amer*., pt. 1, vol. 95, no. 5, pp. 2728 – 2735, 1994.
[3] H. L. Roitblat, W. W. L. Au, P. E. Nachtogall, R. Shizumura, and G. Moons, "Sonar recognition of targets embedded in sediment", *Neural Netw*., vol. 8, no. 7/8, pp. 1263 – 1273, 1995.
[4] J. A. Simmons, P. A. Saillant, J. M. Wotton, T. Haresign, M. J. Feragamo, and C. F. Moss, "Composition of biosonar images for target recognition by echolocating bats", *Neural Netw*., vol. 18, no. 7/8 , pp. 1239 – 1261, 1995.
[5] G. B. Willson, "Radar classification using a neural network", *Proc. SPIE, Optical Engineering and Photonics in Aerospace Sensing: Application of Neural Networks,* vol. 1294, pp. 200 – 210, 1990.
[6] S. Watanabe and M. Yoneyama, "An ultrasonic visual sensor for three-dimensional object recognition using neural networks", *IEEE Trans. Robot. Automat*., vol. 8, pp. 240 – 249, Apr. 1992.
[7] I. E. Dror, M. Zagaeski, and C. F. Moss, "3-dimensional target recognition via sonar – a neural network model", *Neural Networks*, vol. 8, no. 1, pp. 149 – 160, 1995.
[8] R. P. Gorman and T. J. Sejnowski, "Learned classification of sonar targets using a massively parallel network", *IEEE Trans. Acoust*., *Speech, Signal Processing*, vol. 36, no. 7, pp. 1135 – 1140, 1998.
[9] B. Ayrulu and B. Barshan, "Identification of target primitives with multiple decision-making sonars using evidential reasoning," *Int. J. Robot. Res*., vol. 17, no. 6, pp. 598 – 623, 1998.
[10] L. Yingwei, N. Sundararajan, and P. Saratchandran, "Performance evaluation of a sequential minimal Radial Basis Function (RBF) neural network learning algorithm," IEEE *Trans. Neural Networks*, vol. 9, pp. 308-318, Mar. 1998.
[11] L. Yingwei, N. Sundararajan, and P. Saratchandran, "A sequential learning scheme for function approximation using minimal radial basis function neural networks," *Neural Computing*., vol. 9, pp. 461-478, Feb. 1997.
[12] M. Chan, C. Fung, "Incremental Adaption of Resource-Allocating Network for Non-Stationary Time Series," *Neural Networks*, 1999. IJCNN '99. *IEEE International Joint Conference*, Volume: 3 , 10-16, July 1999, pp. 1554 -1559 vol.3
[13] B. L. Pulito, T. R. Damarla, S. Nariani, "A Two-Dimensional Shift Invariant Image Classification Neural Network which overcomes the Stability / Plasticity Dilemma," *Neural Networks*, 1990., 1990 IJCNN International Joint Conference on 17-21 June 1990, pp. 825 –833, vol.2
[14] J. P. Albright, "An implementation and Evaluation of the ART1 Neural Network For Pattern Recognition," Neural Networks, 1994. *IEEE World Congress on Computational Intelligence., 1994 IEEE International Conference*, Volume: 1 , 27 June-2 July 1994, pp. 498 –502, vol.1
[15] Youngtae Park, "An ART2 trained Two-Stage Learning on Circularly Ordered Data Sequence," *Neural Networks, 1994. IEEE World Congress on Computational Intelligence., 1994 IEEE International Conference*, Volume: 5 , 27 June-2 July 1994, pp. 2928 –2933, vol.5
[16] Rok Rape, Dusan Fefer, Janko Drnovsek. Time Series Prediction with Neural Networks: A Case Study of Two Examples, IMTC'94, *IEEE*, May 1994.
[17] Ben Jacobsen. Time Series Properties of Stock Returns, Kluwer *BedrijfsInformatie,* 1997.
[18] Donald F. Specht, "Enhancements To Probabilistic Neural Networks," Neural Networks, 1992. IJCNN., *IEEE International Joint Conference*, Volume: 1 , 7-11, June 1992, pp. 761 – 768, vol.1
[19] B. Barshan, B. Ayrulu, and S. W. Utete, "Neural Network-Based Target Differentiation Using Sonar for Robotics Applications," *IEEE Trans. Robotics and Automation*, vol. 16, pp. 435 – 442, August 2000.
[20] N. Sundararajan, P. Saratchandran and, Lu Ying Wei, book of "Radial Basis Function Neural Network," by School of Electrical & Electronic Engineering, Nanyang Tech. Univ, S'pore.

[21] W. S. Lim, M.V.C. Rao, C.K. Loo,"Sequential Learning Neural Network For Sonar Target Differentiation," *Neural Network World 2/04*, 187-197.
[22] D. F. Specht, "Probablistic Neural Networks," *Neural Networks 3* (1990) 109-118.
[23] D. F. Specht, "Probabilistic neural networks for classification, mapping, or associative memory," ICNN, vol.1 (San Diego, CA), July 1988, pp. 525-532.
[24] D. J. Marchette and C. E. Priebe, "The adaptive kernel neural network," Reports: AD-A217 230, 1989.
[25] P. S. Naloney, "The use of probabilistic neural networks to improve solution times for hull-to-emitter correlation problems," *IJCNN, Sheraton Washington*, Vol.1, 1989, pp. 289-294.
[26] D. F. Specht, "Application of probabilistic neural networks," SPIE, *Appl. Artif. Neural Networks* 1294 (1990) 344-353.
[27] J. J. Guo, P. B. Luh, "Selecting input factors for clusters of Gaussian radial basis function networks to improve market clearing price prediction," *Power Systems, IEEE Transactions*, Volume 18, Issue 2, May 2003 Page(s):665 – 672.
[28] G. B. Huang, P. Saratchandran, N. Sundararajan, "A generalized growing and pruning RBF neural network for function approximation," Neural Networks, *IEEE Transactions*, Volume 16, Issue 1, Jan. 2005 Page(s):57 – 67.
[29] G. B. Huang, P. Saratchandran, N. Sundararajan, "An efficient sequential learning algorithm for growing and pruning RBF networks," Systems, Man and Cybernetics, Part B, *IEEE Transactions*, Volume 34, Issue 6, Dec. 2004 Page(s):2284 – 2292.