1998 International Computer Symposium
Workshop on Artificial Intelligence
December 17-19, 1998, N.C.K.U., Tainan, Taiwan, R.O.C.

IMPROVING GENERALIZATION PERFORMANCE IN CNN CLASSIFIER USING A MLP-BASED TECHNIQUE

Mostafa Mahmoud Syiam, Member, IEEE

Department of Electrical and Computer Engineering,
Faculty of Engineering, Applied Science University,
11931 Amman, Jordan , Email: syiam@asu.edu.jo

ABSTRACT

This paper presents a multi-layer perceptron (MLP)-based technique for improving generalization performance in condensed nearest-neighbor (CNN) classifier. The CNN classifier is simple and efficient in time due to its condensed set of prototypes. However, its generalization performance is not as good as that of nearest-neighbor (NN) classifier that uses the complete large training data set. The developed MLP-based technique is used to modify the condensed set of prototypes of CNN classifier in order to generate or enhance the useful features of such prototypes so that the CNN classifier could also achieve good generalization performance. The improving in the performance of the developed CNN classifier is shown by experimental results.

1. INTRODUCTION

In the last three decades, the two areas of research on pattern classification and artificial neural networks represent very different approaches to solve pattern recognition problems. However, they are now very closely related. Understanding their relationship helps us to develop techniques to improve the efficiency of each approach. The neural networks algorithms have suggested many new ways to improve significantly the traditional classifiers [1]. On the other hand, traditional classification-based methods have been suggested also to design artificial neural networks efficiently [2]. MLP is a multi-layer feed-forward neural network with a single hidden layer. Among several models of neural networks, MLP has been most frequently used as a powerful tool for pattern classification problems. Its strength is in the discriminative power and the capability to learn. No appreciable difference was found in the discrimination capabilities of MLP and multi-layer feed-forward neural networks with two hidden layers. However, networks with two hidden layers were found to be more difficult to train. Error backpropagation (EBP) algorithm [3] is one of the most important and widely used algorithms for training MLP. One of the major problems of EBP algorithm is the long training time. Many modifications and new training algorithms have been proposed [4,5] to speed up the training of the EBP algorithm.

Since their introduction, nearest-neighbor (NN) classifiers have been shown very effective in practice for many problem domains. The NN classifiers were originally proposed as tools for pattern classification, however, they are widely used in many applications of intelligent systems such as speech recognition, medical diagnosis, and others [6]. The NN classifier is a simple and powerful classification technique. In the basic NN classifier, each training pattern in the design set is used as a prototype, and a test pattern is assigned to the class of the closest prototype. Its classification error is bounded above by twice the error of the optimal Bayes classification rule if its design set has a very large number of training patterns [7]. Recently, the NN classifier has been shown to have equivalent generalization performance as neural network based classifiers [8]. However, it needs a very large set of training patterns to achieve such generalization performance. Thus, the implementation of the NN classifier is computationally expensive in terms of computing time.

The NN classifier can be implemented efficiently if the large set of training patterns is condensed into a reduced set of representing prototypes. Many condensing methods were developed to reduce the size of the design set into a small set of prototypes, so that the NN classifier can be implemented efficiently. The resulting classifier was called CNN classifier [9-11].

2. BACKGROUND

Generalization performance and efficiency are the two most important features for developing classification systems. The first one, addresses the problem of how to develop a classification system to achieve optimal performance on the samples that are not included in the training data set using a finite number of training samples. The generalization performance of a classification system is characterized by its generalization accuracy, which is the probability of correct classification of a random sample by this classifier [12,13].

The NN classifier that trained by a large set of training patterns and the well-trained MLP has a very good generalization performance. The second feature, efficiency, deals with the complexity of the system in time. The time complexity of a classification system means the computational time needed by the system for training and classification. In terms of time complexity, CNN classifier is better than both NN classifier and MLP classifier [14]. Unfortunately, the generalization performance of CNN classifier using the reduced set of prototypes is not as good as that of NN classifier using the complete large training set.

Recently, many methods have been developed to overcome this problem [15,16]. In these methods, the reduced set of prototypes is modified using neural network based methods in order to increase the classification performance of the classifier. In [15], it has been shown that the performance of NN classifier with only reduced number of prototypes can be improved if the classifier is mapped and trained with three layers feed-forward neural network. However, the resulting neural network classifier can not be mapped back to a NN classifier. In [16], a method for optimizing prototypes using a three layers feed-forward neural network has been developed. In this method the initial prototypes are set equal to the averages of clusters of the training patterns. The feature values of prototypes are mapped to the weights of connections of a three layers feed-forward neural networks classifier. After training, the trained neural network could be mapped indirectly to a NN classifier using a mapping model. Three problems could be identified in the previous two methods [15,16]. In both methods, the concept of CNN has not been considered. But, general clustering methods have been used to get the reduced set of prototypes for the NN classifier. Although clustering methods have been used to generate a reduced set of prototypes, they do not take the quality of the final prototypes into consideration for classifying other patterns which are not included in the training data set. On the other hand, in mapping a NN classifier to a three layers feed-forward neural networks classifier, the number of input nodes in input layer was set greater than the number of features which led to increase the number of connections in the network. Moreover, in [16] the final feature values of the modified prototypes can not be get directly from the connection weights of the trained neural network, but it could be obtained using a set of mapping equations.

This paper presents a MLP-based technique to overcome these problems. In this method, a developed condensing technique, that takes the quality of the final prototypes into consideration, is used to condense the complete large training set of patterns into a small number of boundary prototypes such that only training patterns close to the boundaries between classes are retained as prototypes. Then, the developed MLP-based technique is used to modify these prototypes to generate or enhance the useful features of the prototypes, such that the classification rate of this enhanced condensed set of prototypes can be as good as that of the complete large training data set. Experimental results for evaluating and comparing the generalization performance of the developed CNN classifier are presented.

3. THE DEVELOPED MLP- BASED TECHNIQUE

The developed MLP-based technique consists of four steps for improving the generalization performance of CNN classifier. These steps are: 1) reducing the number of prototypes using a developed condensing method, 2) mapping CNN classifier to MLP classifier, 3) training the mapped MLP classifier, using a developed training algorithm, 4) mapping back the trained MLP classifier with its final weights as prototypes to the developed CNN classifier, and implementing the developed classifier.

3.1 Reducing The Number Of Prototypes

The analysis in the present paper gives a great importance for the boundary samples as prototypes for achieving high generalization performance for

CNN classifier. On the other hand, the boundary samples are also important for training neural network classifiers. The experimental results obtained in [17] confirmed that the boundary samples are better than other impeded samples for training neural network classifiers. The importance of the boundary samples for training neural network classifiers is illustrated in Fig. 1 and Fig. 2. In Fig. 1, the samples inside circles are those arbitrary samples which are presented to the classifier as prototypes for training, and the other ones are for testing. However, the classification performance of the resulting decision boundary on the test samples is poor as shown in Fig. 1. On the other hand, by presenting boundary samples to the classifier as prototypes for training, as shown in Fig. 2, the classification performance of the resulting decision boundary on test samples is very good.

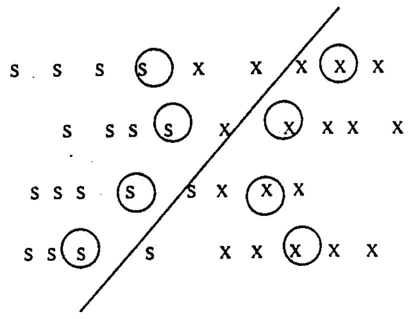
Fig. 1. Using non-boundary samples for training

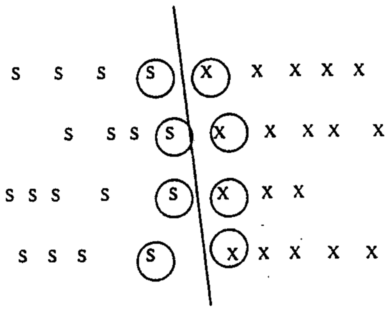
Fig. 2. Using boundary samples for training

In this paper a condensing technique is developed to reduce the large data set of M training patterns into a small number of L boundary prototypes such that only training patterns close to the decision boundary are retained as prototypes. This technique is based on integrating the multi-edit algorithm [10], and the modified condensing technique [11]. The multi-edit algorithm is applied at first to the large set of M training patterns to remove those patterns, which are surrounded by patters from other classes. The decision boundary is formed using only the patterns belonging to outer envelopes of different classes. The patterns, which do not contribute to the forming of the decision boundary, may be deleted with no effect on the classification performance of the classifier. Further reduction is performed using the modified condensing technique. The basic idea of the modified condensing technique is to arrange the edited training patterns in some order such that only patterns close to the decision boundary are retained as prototypes and all other patterns are removed. By obtaining the reduced set of boundary prototypes, the main task now is to modify these prototypes to generate or enhance its useful features in order to improve the generalization performance of the designed CNN classifier.

3.2 Mapping CNN Classifier to MLP Classifier

In this mapping, each input training pattern is represented as a vector X of N features:

$$X = [x_1\ x_2\ x_3\ \dots\ x_N]^T \qquad (1)$$

where x_i, i=1,2, ..., N, are the feature variables.
The developed MLP-based technique uses MLP classifier with only one hidden layer. The number of input nodes in the input layer is equal to the number of features N of the input pattern X. The number of output nodes in the output layer is equal to the number of classes, K. The number of hidden nodes in the hidden layer is selected equals to the number of reduced prototypes L. Thus, the architecture of the MLP classifier after mapping is N-L-K.

3.2.1 Weight Connections Of The Hidden Layer

Each hidden node in the hidden layer, using the developed MLP-based technique, represents a prototype. The weight connections, between each hidden node and an input node in the input layer, are initially set equal to the feature values of the corresponding prototype. The features of the prototype P_j representing a hidden node j is given by:

$$P_j = [p_{j1}\ p_{j2}\ \dots\ p_{jN}]^T \qquad (2)$$

where j=1,2, .., L. Then, the initial weight connections between a hidden node j and an input node *i* are represented as:

$$w_{ji} = p_{ji} \text{ for } j=1,2,\dots,L, \text{ and } i=1,2,\dots,N \qquad (3)$$

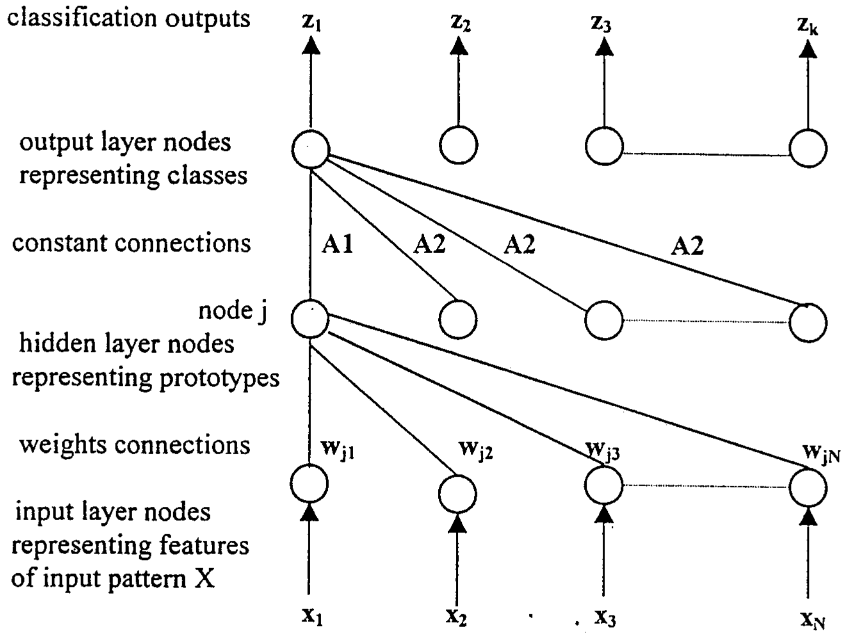

Fig. 3. The developed mapped architecture of the MLP classifier

If we take A1=-A2=1 in (12), then a better matching could be obtained between the links from the inputs and the hidden nodes of group s, and weaker matching could be obtained between the links from the inputs to other hidden nodes. Therefore, the features that have high classification power are enhanced during the training process, while other features with low classification power are not optimized. On the other hand, since A1 and A2 are not updated using the developed training algorithm, so the trained MLP classifier could be mapped back to the CNN classifier and the optimized prototypes feature values could be obtained directly.

3.4 Mapping Back The MLP classifier

Once the MLP classifier is trained successfully using the developed training algorithm trains, it could be mapped back to the developed CNN classifier. The updated weights, between a hidden node j, and N input nodes, represent the features p_{ji} of the enhanced prototype j, and they are given directly as:

$$p_{ji} = w_{ji} \text{ for } (j=1,2, \ldots, L, \text{ and } i=1,2, \ldots, N) \quad (13)$$

Obtaining the enhanced set of prototypes for the developed CNN classifier, the design is complete and the developed classifier could be used for classification. To classify a pattern x from a set of testing patterns F, using the enhanced set of prototypes P, we assign x to the class of the nearest prototype p_{NN}. Thus, the classification rule of the developed CNN classifier is simple and it is given formally by:

$$\forall\, x \in F, \quad \text{Class}(x) = \text{Class}(p_{NN}),$$

$$\text{if } d(x, p_{NN}) = \min d(x,p_j), \quad \forall\, p_j \in P \quad (14)$$

where d is some metric of feature space, and N is the dimension of the feature space. Using Euclidean distance, d is given by:

$$d(x, p_j) = \sum_{j=1}^{N} (x_i - p_{ji})^2 \quad (15)$$

It is clear that the computational complexity of the classification of the developed CNN classifier is very efficient and better than that of multi-layer feed-forward neural networks classifier. The neural classifier has several fully connected layers and activities of nodes in all layers must be computed to determine the classification result. Further reduction in the computational complexity of the developed CNN classifier could be achieved using several

The output y_j of a hidden node j is given by:

$$y_j = f(u_j) \qquad (4)$$

where $f()$ is the sigmoid activation function and it is defined as:

$$f(u_j) = 1 / (1 + \exp(-u_j)) \qquad (5)$$

where u_j is the sum of inputs to the hidden node j from the input layer. In the proposed technique, u_j is equal to the square distance between the input pattern X, and the corresponding prototype P_j of the hidden node j:

$$u_j = d^2(X, P_j) = (x_1 - w_{j1})^2 + \ldots (x_N - w_{jN})^2 \qquad (6)$$

Thus, the output y_j of a hidden node j is:

$$y_j = f((x_1 - w_{j1})^2 + \ldots (x_N - w_{jN})^2) \qquad (7)$$

3.2.2 Weight Connections Of The Output Layer

The connection weight from an output node k to a hidden node j is represented by a constant A1, if the output node k and the hidden node j, represent the same class, and a constant A2 if they represent two different classes. The mapped architecture of the MLP classifier is shown in Fig. 3. In this figure, only the connections from the input nodes to the hidden node j, and the connections from the hidden nodes to the first output node are shown for simplicity.

3.3 Training The Mapped MLP Classifier

As shown in Fig. 3, the input layer of the MLP classifier performs the matching between the input patterns and the prototypes represented by the weights. The hidden layer performs the classification decision making. For an input pattern, in order to have a good classification rate, at least one hidden node that represents the class of the input pattern must have a large response to activate the output node representing the corresponding class.

On the other hand, all other hidden nodes that do not represent the class of the input pattern should have very small responses. Thus, to achieve better generalization performance, the weight connections between the hidden nodes and the input nodes have to be modified during training in a such way that each hidden node responses only to the training patterns of the class which it represents. This goal could be achieved by training the mapped MLP classifier shown in Fig. 3 by a developed training algorithm based on the EBP algorithm [3].

3.3.1 The Developed Training Algorithm

Assuming that the number of output nodes representing classes is K, the desired and the actual output of output node k, k=1,2, .. , K are z_k and z_{kd} respectively for an input training pattern. The developed training algorithm adjusts the weights between the hidden layer and input layer only in order to minimize the error function:

$$E_X = 1/2 \sum_k (z_{kd} - z_k) \qquad (8)$$

The actual output z_k of the node k is given by

$$z_k = 1 / (1 + \exp(-v_k)) \qquad (9)$$

where v_k is the input to the output node k and is given by:

$$v_k = \sum_{j \in s} A_1 y_j + \sum_{j \in \bar{s}} A_2 y_j \qquad (10)$$

where s is the group of all hidden nodes that represent prototypes which belong to the class represented by output node k, and s^- is the group of all other hidden nodes. According to the generalized delta rule [3], a local optimal w_{ji} can be found by changing it with an increment given by:

$$\Delta w_{ji} = -\alpha\, \delta E / \delta w_{ji} \qquad (11)$$

where α is the learning factor. Using chain rule, Δw_{ji} can be expressed as:

$$\Delta w_{ji} = -\alpha \sum_k \delta E/\delta z_k\, \delta z_k/\delta v_k \sum_j \delta v_k/\delta y_j\, \delta y_j/\delta u_j \delta u_j/\delta x_i$$

This chain rule expression of equation (11) can be solved and simplified using equations (1-10) as follows:

$$\Delta w_{ji} = -\alpha \sum_k (z_{kd} - z_k)\, z_k (1 - z_k) * \qquad (12)$$

$$[\sum_{j \in s} A_1 y_j (2 - y_j)(x_i - w_{ji}) + \sum_{j \in \bar{s}} A_2 y_j (2 - y_j)(x_i - w_{ji})]$$

methods. An efficient branch-and-bound algorithm [18] was used to speed up the searching for the closet prototype to the input pattern.

4. EXPERIMENTAL RESULTS

To demonstrate the improving in the generalization performance of CNN classifier using the developed MLP-based technique, we conducted several experiments with a rock type classification problem utilizing well-logging data, released by an oil company.
In this problem, 300 training patterns are obtained from a key well in an oil field. Each training pattern consists of 13 features, where each feature is a well-log electrical measurement. The rock-type of each training pattern is determined using core-to-log correlation study and the knowledge of the geological experts. Thus, the training data set of 300 patterns is subdivided into 5 classes (rock- types) on the basis of core-to-log correlation, and the knowledge of the experienced geologists. Other 300 patterns are obtained from other wells in the same field for testing.

In order to investigate the influence of the training data size on the performance of the developed CNN classifier, the design of the developed CNN classifier was performed using several training data subsets of different sizes. We have used 4 data subsets consisting of 150, 200, 250, and 300 training patterns respectively. We conducted 4 experiments to design and test the developed CNN classifier. In each experiment, we applied the developed condensing technique on each training data subset to obtain the reduced set of boundary prototypes. Table 1 shows the number of reduced prototypes for each experiment. Then, in each experiment, the CNN classifier is mapped to a MLP classifier. The architecture of the MLP classifier in each experiment is 13-L-5, where 13 is the number of input nodes representing features of input pattern, 5 is the number of output nodes representing 5 rock-type classes, and L is the number of hidden nodes and it equals the number of reduced prototypes in such experiment.

The MLP classifier is trained using the developed training algorithm and utilizing all training pattern of the corresponding data subset. After successful training of each MLP, its structure with new weights is mapped back to the CNN classifier. Thus, we obtain a reduced set of enhanced prototypes. Then the developed CNN classifier is tested using the other 300 testing patterns. Table 1 shows the classification testing results. It is important to notice from Table 1 that the developed CNN classifier achieves its best generalization performance (96.2 % classification rate) in experiment 4 when it uses more enhanced prototypes.

To evaluate the improving in the generalization performance of the developed CNN classifier, the testing results of each experiment are compared with the testing results of the conventional CNN classifier. The conventional CNN classifier is designed in each experiment using the same reduced set of prototypes without modification using the developed technique. On the other hand, it is tested in each experiment using the same testing patterns used for testing the developed CNN classifier. The comparison of testing results, shown in Table 1 of both classifiers, indicates that the developed CNN classifier has improved the generalization performance of the conventional CNN classifier by a rate in the range 6.5% to 9%.

Moreover, the developed CNN classifier is compared with other two classifiers, the NN classifier, and the multi-layer neural networks classifier. The NN classifier is trained in each experiment using all training patterns of each training data subset as prototypes. On the other hand, the multi-layer neural networks classifier is trained in each experiment by the standard EBP algorithm [3], using also all training patterns each data subset. The architecture of this classifier is 13 input nodes, 5 output nodes, and a selected number of hidden nodes. The selected number of hidden nodes in the 4 experiments is 5, 7, 10, and 11 respectively. The number of hidden nodes for each experiment is determined as the sub-optimal number using many try-and-error experiments. The multi-layer neural networks classifier is trained using the standard EBP algorithm with learning rate 0.1, momentum zero, and a root mean square less than .005. The NN classifier and the multi-layer neural networks classifier are tested in each experiment using the same testing patterns used for testing the developed CNN classifier. Table 1 summarizes the results of testing these two classifiers. Comparing testing results of these two classifiers with the developed CNN classifier indicate that the developed CNN classifier reaches the performance of the NN classifier, which uses all training patterns as prototypes in all experiments. Moreover, it achieves in many experiments almost the same generalization performance of the multi-layer neural networks classifier.

TABLE 1

COMPARISON OF GENERALIZATION PERFORMANCE (% ACCURACY IN CLASSIFICATION) BETWEEN THE DEVELOPED CNN CLASSIFIER AND OTHER THREE CLASSIFIERS

No of training patterns	No of reduce prototypes	No of testing patterns	NN classifier (%)	Neural Network classifier (%)	Conventional CNN classifier (%)	The Developed CCN classifier (%)
150	11	300	91	91.2	83.4	90.1
200	14	300	91.8	92.4	84.6	91.5
250	19	300	93.4	93.6	86.4	92.8
300	23	300	96.5	96.8	87.1	96.2

5. CONCLUSION

A MLP-based technique is developed in this paper to improve the generalization performance of the CNN classifier. A reduced number of efficient boundary prototypes are obtained from the large training data set using a developed condensing technique. These prototypes are then modified using the developed MLP-based technique to generate or enhance the useful features of such prototypes so that the generalization performance of the developed CNN classifier could be as good as that of the NN classifier. The testing and the comparison results indicate that the best generalization performance of the developed CNN classifier is 96.2%, and it has improved the generalization performance of the Conventional CNN classifier by a rate of the range 6.5% to 9%. The comparison results indicate that the developed CNN classifier achieves in all experiments almost the same generalization performance of both the NN classifier and the multi-layer neural networks classifier. However, the computational complexity of the developed CNN classifier is better than that of the NN classifier and multi-layer neural networks classifier, due to its reduced set of prototypes. There still more works for further evaluation of the developed technique using other classification problems.

6. REFERENCES

[1] C. H. Chen, "On the Relationships between statistical pattern recognition and artificial neural network," International journal of pattern recognition and artificial intelligence, vol. 5, pp. 655-661, Oct., 1991.

[2] Q. Zaho, " Stable on-line evolutionary learning of NN-MLP," IEEE Trans. Neural Network, vol. 8, pp. 1371-1378, Nov, 1997.

[3] D. E. Rumelhart and J. L. McCelland, Parallel Distributed Processing, Vol. 1, MIT Press, 1986.

[4] S. Ergezinger and E. Thomsen, "An accelerated learning algorithm for multilayer perceptrons: optimization layer by layer, "IEEE Trans. Neural Network, vol. 6, pp. 31-34, 1991.

[5] B. Verma, " Fast training of multilayer perceptron," IEEE Trans. Neural Network, vol. 8, pp. 1324-1320, Nov., 1997.

[6] S. Salzberg, A. Delcher, D. Heath, and S. Kasif " Best-Case Results for Nearest-Neighbor Learning," IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 17, pp. 599-608, June, 1995.

[7] T. M. Cover and P. E. Hart, "Nearest neighbor pattern classification, " IEEE Trans. Inf. Theory, IT-13, pp. 21-27, 1967.

[8] P. E. Lee, " Handwritten digit recognition using k nearest neighbor, radial basis function, and backpropagation neural networks, " Neural Computation, vol. 3, pp. 440-449, 1991.

[9] P. E. Hart, "The condensed nearest neighbor rule, " IEEE . Inf. Theory, IT-14, pp. 515-516, 1968.

[10] P. A. Devijver, and J. Kittler, "On the edited nearest neighbor rule, " in proc. of 5th Int. Conf. of pattern recognition, pp. 72-80, 1980.

[11] I. Tomek, "Two modifications of CNN, "IEEE Trans. Man and Cyp. CMC-6, pp. 769-772, 1976.

[12] C. Ji, and S. Ma, " Combination of weak classifiers," IEEE Trans. Neural Network, vol. 8, pp. 32-42, Jan., 1997.

[13] L. holmstron, P. koistinen, j. Laaksonen, and E. Oja," Neural and statistical classifier Taxonomy and Two Case Studies," IEEE Trans. Neural Network, vol. 8, pp. 5-17, Jan., 1997.

[14] S. Gazula, and R. Kubuka, " design of supervised classifiers Using Boolean Neural Network," IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 17, pp. 1239-1246, Dec., 1995.

[15] H. Yan, "Iterative improvement of a nearest neighbor classifier, " Neural Networks, vol. 4, pp. 517-524, 1991.

[16] H. Yan, "Building a robust nearest neighbor classifier containing only a small number of prototypes, " Int. journal of Neural Systems, Vol. 3, pp. 361-369, 1992.

[17] K. G. Mehrota, C. K. Mohan, and S. Raka, "bounds on the number of samples needed for neural learning, " IEEE Trans. Neural Network, vol. 2, pp. 548-558, 1991.

[18] Q. Jang and W. Zhang, "An efficient method for finding nearest neighbors, " Pattern recognition letter 14, pp. 531-535, July 1993.