FUZZY NEURAL NETWORK FOR EDGE DETECTION AND HOPFIELD NETWORK FOR EDGE ENHANCEMENT

ZIQING WANG

INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

Bell & Howell Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA

UMI®
800-521-0600

FUZZY NEURAL NETWORK FOR EDGE DETECTION AND HOPFIELD NETWORK FOR EDGE ENHANCEMENT

by

©Ziqing Wang

A thesis submitted to the

School of Graduate Studies

in partial fulfillment of the

requirements for the degree of

Master of Science

Department of Computer Science

Memorial University of Newfoundland

January 1999

St. John's Newfoundland Canada

Abstract

This thesis presents an artificial neural network system for edge detection and edge enhancement. The system can accomplish the following tasks: (a) obtain edges; (b) enhance edges by recovering missing edges and eliminate false edges caused by noise. The research is comprised of three stages, namely, adaptive fuzzification which is employed to fuzzify the input patterns, edge detection by a three-layer feedforward fuzzy neural network, and edge enhancement by a modified Hopfield neural network. The typical sample patterns are first fuzzified. Then they are used to train the proposed fuzzy neural network. After that, the trained network is able to determine the edge elements with eight orientations. Pixels having high edge membership are traced for further processing. Based on constraint satisfaction and the competitive mechanism, interconnections among neurons are determined in the Hopfield neural network. A criterion is provided to find the final stable result which contains the enhanced edge measurement.

The proposed neural networks are simulated on a SUN Sparc station. One hundred and twenty-three training samples are well chosen to cover all the edge and non-edge cases and the performance of the system will not be improved by adding more training samples. Test images are degraded by random noise up to 30% of the original images. Compared with standard edge detection operators, the proposed fuzzy neural network obtains very good results.

Acknowledgment

I wish to express my thanks to my supervisor Dr. Siwei Lu for his insightful guidance, constructive criticism, constant encouragement and financial support. Without his contribution, it would be impossible to give this thesis its current quality.

I would like to thank the systems support staff, and in particular Nolan White, for providing all the help and assistance during my research.

I am also very grateful to the administrative staff who have helped in one way or another in the preparation of this thesis.

In addition, I would like to acknowledge the financial support received from the School of Graduate Studies and the Department of Computer Science.

Special thanks are due to my fellow graduate students, and in particular to Xuede Chen, Gihad Rabi and Kaleem Momin for their useful suggestions and assistance.

This thesis is dedicated to my parents

and my husband, Guang Qi, for their

encouragement and support throughout

the course of my education.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In recent years, an increasing number of researchers have been involved in the subject of fuzzy neural networks in hope of combining the strengths of fuzzy logic and neural networks and achieving a more powerful tool for fuzzy information processing and for exploring the functioning of the human brain. A typical neural network has multiple inputs and outputs which are connected by many neurons via weights to form a parallel structure for information processing. It processes data without uncertainty. However, our universe is full of uncertainty. This uncertainty can be described as the possibility by fuzzy set theory. It is noted that fuzzy logic, a superset of conventional logic, has been used to handle the concept of partial truth - truth values between completely true and completely false. Fuzzy set theory treats objects as members of fuzzy sets, and describes the uncertainty of an object by its membership in fuzzy sets. It is interesting that fuzzy logic is

another powerful tool to model phenomena associated with human thinking and perception[79, 80, 81]. In fact, the neural network approach merges well with fuzzy logic and some research endeavors have given birth to the so-called "fuzzy neural networks" (FNN). Neuro-fuzzy methods have been recognized as state-of-the-art techniques and are believed to have considerable potential in the areas of expert system, medical diagnosis, control system, pattern recognition, computer vision, image processing and system modeling.

Edges are the consequences of changes in some physical and surface properties, such as illumination, geometry or reflectance. An edge image is an image in which the gray level reflects how strongly each corresponding pixel meets the criteria of an edge pixel. It conveys most of the important scene information as there are direct correlations between the edges and the physical properties of a scene. Edge detection is searching for edges between regions. It is an important problem in image processing and an essential part of many computer vision systems. It is hard not to over-emphasize the importance of edge detection in image understanding. Most modules in a vision system depend, somehow, on the performance of the edge detector. Edge detection techniques have various applications such as pattern recognition, robot scene analysis, and image coding.

There are many methods regarding edge detection such as edge detection by using differential operators (Robert, Sobel, Prewitt operators), the template matching edge operators, image filtering techniques or statistical techniques. However,

many edge operators rely totally on grey level differences for their approximation of the image gradient function either directly or by representing these differences in a more analytical form. The statistical or stochastic methods for boundary extraction, such as Markov random field and MAP estimation do not perform very successfully since these methods have a limited structure knowledge. Other approaches include attempts at identifying the type of edge as well as the edge position to sub-pixel accuracy by analyzing the location and type of phase congruence of Hilbert-Fourier representations using local energy[14]. However, the method has not been demonstrated to work in noisy images. Therefore, accurate edge detection is a difficult task and there has been a substantial effort to develop the ideal edge operators or detectors.

As a matter of fact, an edge can not be recognized with 100% confidence. Uncertainty arises because of ambiguity or lack of information or evidence. Current techniques have limited capability to perform good and accurate edge detection for noise corrupted and degraded images. Neural networks are considered a sort of model-free signal processing device. They map data points in input data space to points in output data space. The training data points should be properly distributed and dense enough. On the other hand, fuzzy neural networks try to cut down the requirements on the training data sets by incorporating the expert knowledge with the fuzzy concept. They perform set to set mapping. This is achieved by adding point-to-set (fuzzify) and set-to-point (defuzzify) conversion to the input

and output, respectively. In this thesis, a three-layer feedforward fuzzy neural network is proposed for obtaining edge measurement. It classifies the input pattern in a noise corrupted image into non-edge or edge with one of eight orientations. Two non-edge credit maps and eight edge credit maps are generated. This is a very important stage because its results somehow have an effect on the further processing by a Hopfield network for edge enhancement. Based on the constraint satisfaction, the competitive mechanism and energy function, the structure of the Hopfield neural network is designed. It efficiently captures the topological and structural properties of the edge data obtained from the first phase. Hence, the updating in the neural computation leads towards the right solution by establishing proper interconnections among neurons.

1.1 Overview of the System

Figure 1.1 shows an overview of the proposed system in operation. The design of the system aims to detect and enhance edges in noisy images. In the first stage, before using the proposed fuzzy neural network for edge detection, an adaptive fuzzification procedure is used to adaptively fuzzify input patterns. Three main steps, input pattern generation, adaptive fuzzification and parameter estimation, are included in. They have been considered as an important procedure in the research. It enable the pixels to be measured by the degree of darkness based on the local windows. Because many patterns can be viewed as the same after they

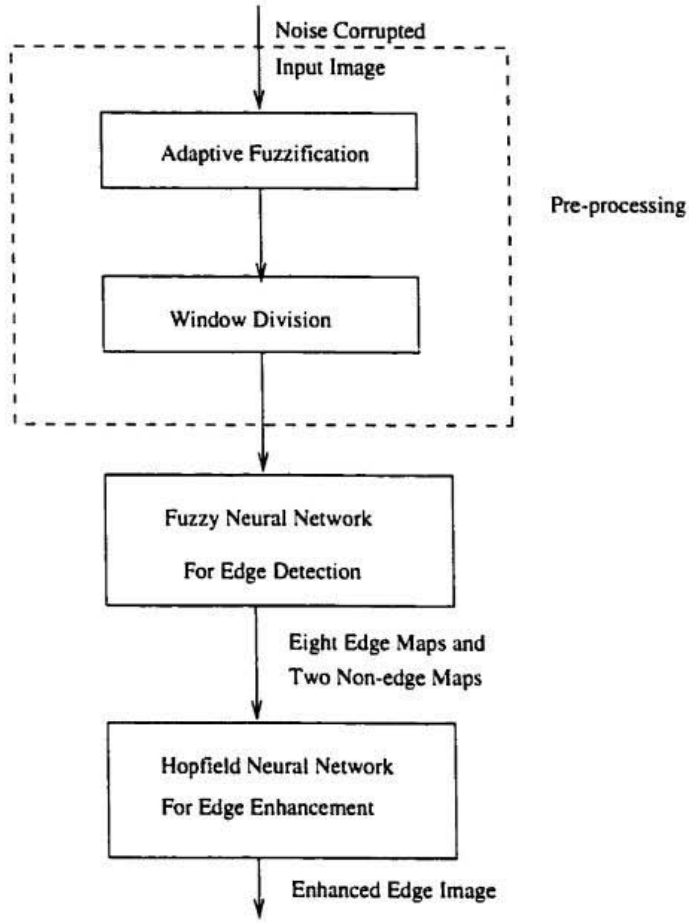


Figure 1.1: Overview of the proposed system

are adaptively fuzzified, therefore, the sample number used to train the proposed FNN is able to be cut down significantly after this pre-processing ia applied to training patterns.

In the research, there are eight edge patterns considered as typical edge patterns, the edge at the east, west, south, north, southeast, southwest, northeast and northwest orientations. Based on the property of typical edge patterns: half relative black and half relative bright, bright (black) part overlapped at certain

area (block)among patterns, a window division technique is also proposed in this stage to apply on each input pattern. The purpose of this pre-processing is to fully utilize the structure information for later classification by the fuzzy neural network.

The fuzzy neural network system is designed for detecting edges in noisy images. It actually outputs the fuzzy measurement of the darkness of pixels in an image. After applying FNN to a given image, eight edge maps corresponding to eight edge orientations and two non-edge maps are obtained. The edge measurement is used as the input of the Hopfield neural network.

The Hopfield network is for removing noise and enhancing the detected edges in the previous edge maps. After applying the Hopfield neural network, eight enhanced edge maps are obtained. They are then assembled into one picture to form the final edge image with high resolution.

1.2 Organization of the Thesis

Chapter 2 surveys methods for edge detection, techniques for edge improvement, the basic structures of neural networks, as well as the fuzzy neural integrated system. Chapter 3 introduces the idea of why an adaptive fuzzification procedure is needed and how it applies to each input pattern. Chapter 4 presents the window division method which divides the inputs of a window into either four square blocks or four triangular blocks, then uses them as the inputs of the fuzzy neural network.

Chapter 5 contains the structure of the proposed fuzzy neural network, the weight updating rule, the training procedure, how the trained fuzzy neural network is used to detect edges on an image, and what information it will provide to the next Hopfield network. Chapter 6 describes the structure of the modified Hopfield neural network and the software implementation for solving the specific edge enhancement problem. In chapter 7, the experimental results and the comparisons with other methods are exhibited. As well, the main conclusions including the contributions and directions for future research are discussed.

Chapter 2

Survey

2.1 Edge Detection & Enhancement Techniques

An edge is the boundary between two regions with relatively distinct grey level properties[22]. Basically, the idea underlying most edge detection techniques is the computation of a local derivative operator. There has been tremendous research in the area of edge detection and edge enhancement. The typical techniques are reviewed below.

2.1.1 Gradient operators

Most of the earlier edge detection techniques used first order derivative operators such as the Roberts edge operator[59], the Prewitt edge operator and the Sobel edge operator[18]. If a pixel falls on the boundary of an object in an image, then

its neighborhood will be a zone of grey-level transition. In edge detection, an important quantity is the magnitude (slope) and direction of the gradient vector. The edge detection operators examine each pixel neighborhood, the slope as well as the direction of the grey-level transition. Most of the methods are based on convolution with a set of directional derivative masks as mentioned above, the Roberts 2×2 pixel mask, the Prewitt 3×3 pixel mask and the Sobel 3×3 mask [5]. First order gradient operators are fast edge detection operators. They respond better on sharp transitions in low-noise images and could sharpen the edge contours, but also inadvertently enhance the noise.

The Laplacian operator[22] is a second order derivative operator for functions of two dimension operators and is used to detect edges at the locations of the zero crossing. However, it will produce an abrupt zero-crossing at an edge and zero-crossings do not always correspond to edges. If a noise-free image has sharp edges, the Laplacian can find them. The binary image that results from thresholding a Laplacian-filtered image at zero grey level will produce closed, connected contours when interior points are eliminated. The smoothing operation with the Gaussian mask tends to blur weak edges, and furthermore, the presence of impulse noise in transmitted images can seriously degrade the performance of the smoothing operator. Another gradient operator is the Canny operator[10] which is used to determine a class of optimal filters for different types of edges. e.g., step edges or ridge edges. A major point in Canny's work is that a trade-off between detection

and localization emerged; as the scale parameter increases the detection increases, localization decreases. Canny proposed to select the smallest scale, provided that a minimum value for the signal to noise ratio (SNR) is obtained. The selection of the minimum SNR satisfying the detection limit comes from the maximization of localization. In order to set the appropriate value for the scale parameter, it is required to know the noise energy. However, it is not an easy task to locally measure the noise energy because both noise and signal affect any local measure.

2.1.2 Template matching edge operators

The Kirsch masks[5], the Robinson masks[60], the Nevatia-Babu masks[52], and the Compass Gradient masks[55] are the popular edge template matching operators. Each point in the image is convolved with eight masks corresponding to eight orientations. Each mask responds maximally to an edge oriented in a particular general direction. The maximum value over all eight orientations is the output value for the edge magnitude image. The index of the maximally responding mask encodes the direction of the edge. Although the edge orientation and magnitude can be rapidly estimated by determining the largest response for a set of masks, template mask methods give rise to large angular errors and do not give correct values for the gradient.

Another template matching technique based on the sum of absolute errors is effective in detecting edges where the form of the edges is known in advance[68].

However, this technique requires smoothing to remove noise before it can detect edges. It affects the detection of weak edges. If impulse noise is presented in the image, the performance is also affected. Besides, this method involves expensive pixel-pixel comparison in the image and in the template.

2.1.3 Expansion matching

One of the optimal step edge detectors is derived from the newly developed Expansion Matching (EXM) [74], a technique for robust recognition of templates in an image. The fundamental approach is to match a given template with a given image by expanding the image signal in terms of nonorthogonal Basis Functions (BFs) which are all translated versions of the template. The expansion coefficients obtained at a particular location signify the presence of a template-similar signal (pattern) at that location. Since the shifted templates form a complete nonorthogonal basis[7], this entails a nonorthogonal expansion which can be quite complex if performed directly. However, since all the bases are shifted versions of the same function, this task can be significantly simplified by using frequency domain techniques[58].

2.1.4 Relaxation labeling

There has been an explosive growth in the study of relaxation labeling techniques for image processing, such as image restoration[18], edge enhancement[64], edge

detection[25], [26], [27], [61], and image segmentation[28]. Relaxation labeling uses contextual information to resolve object labeling ambiguities as locally as possible. The amount of contextual information employed is expanded recursively until a unique labeling results. The problem of relaxation labeling was first described by Rosenfeld et al.[61]. Later, various approaches were developed that fall into two categories: discrete relaxation labeling and probabilistic relaxation labeling. In discrete relaxation labeling, label assignments are either possible or impossible. However, in probabilistic labeling, label assignments are measured by probabilities. Therefore, the construction of update functions over the probability vector space is a critical issue. Nonlinear probabilistic update functions yielded the best results, but their heuristic nonlinear update function induces the problem of bias, convergence, and choice of supporting function, etc. Hummel and Zucker[34] addressed some of these problems. They introduced a projected gradient update scheme and derived the property of local convergence for their update function. However the update function is not easy to implement efficiently.

The effectiveness of relaxation labeling lies in its use of contextual information to eliminate ambiguous labels. This is achieved by iterative label assignment updates. Markov Random Field (MRF) theory provides a theoretical basis. Geman and Geman [21] considered images as instances of MRFs. Energy functions were defined for the MRFs such that the original image has minimal energy. A stochastic approach to minimize the energy utilized a simulated annealing technique

and resulted in a highly parallel relaxation algorithm that uses the posteriori distribution to yield a MAP estimate, restoring image from degraded observations. However, their method has a low convergence rate. Due to the nature of simulated annealing, hundreds of iterations are needed to obtain good restoration.

Pelkowitz[56] developed a probabilistic relaxation algorithm using MRF theory by applying the Maximum Entropy approach and deriving a multilinear relaxation update function. The function is data dependent and the final configuration is a function of observations that locally optimize the posteriori probability.

Kittler and Hancock [37] developed an evidence combining formula in the framework of probability theory. They derived a nonlinear update function that is similar to Rosenfeld's. Because no heuristics were used, the problems that Rosenfeld encountered were solved. However, one of the difficulties was its potential computational complexity. The number of possible label configurations is exponential in the number of objects. In reality, the number of permissible configurations is relatively small because labeling problems are highly structured. An exhaustive enumeration of permissible configurations is possible. This can significantly improve the efficiency of relaxation labeling. In this way, Kittler and Hancock successfully developed their algorithm. Their experimental results show better performance than some well-known edge detection algorithm like Canny's and Spacek's[67].

Recently, a new probabilistic relaxation scheme for edge detection or enhancement was proposed [19] which consists of an update function and a dictionary

construction method. The nonlinear update function is derived from MRF theory and Bayes' formula. The method combines evidence from neighboring label assignments and eliminates label ambiguity efficiently. However, if the initial assignment contains too many labeling errors, label contextual information may not be enough to correct all of them.

2.1.5 Fuzzy edge detection techniques

A fuzzy filter and edge detection technique was introduced by Tyan and Wang [72]. The fuzzy rules are used for contrast enhancement, requiring arbitrary definitions of dark and bright pixels, which the authors admit require adjustment by a human operator. Images are further enhanced by an interesting fuzzy low-pass filter. The edge detection schema also relies on arbitrary dark and bright definitions (again requiring user adjustment), and operates on a 2×2 area. A fuzzy edge detection technique was also introduced by Tao and Thompson[70]. In this technique, sixteen possible edge structures in a 3×3 area are considered, and fuzzy edge membership is determined by fuzzy if-then rules. After redundant edge pixels are discarded, the remaining edge pixels are thresholded based on a noise factor. Neither of the techniques above acknowledge the possibility of various edge shapes such as corners and triple points, leaving grey-level differences among immediate neighbors as the only determining factor in edge extraction. Furthermore, both techniques operate on very small regions, ignoring the possibility of broader edges in the process.

Another edge detection technique has been proposed by Todd Law based on the fuzzy reasoning approach. This technique acknowledges the existence of imperfect edges and of ambiguous edges, and then elaborates a linguistic framework for characterizing these ambiguities and imperfections. These characteristics are then approximated from local statistical parameters, and fuzzy rule based on heuristics are introduced to evaluate these parameters. This algorithm has been tested on a variety of real images with positive results. As the current trends in image analysis show a shift towards top-down image processing techniques, this algorithm still deals with bottom-level detail.

2.1.6 Image filtering techniques

Some of the earliest filtering techniques are linear filtering techniques, which are used for edge detection. These techniques utilized a stochastic model of edge structure and the edge detection problem was formulated as one of least mean-square spatial filtering. Two-dimensional recursive digital filtering was used to detect edges in noisy images. Besides the noise immunity, the recursive nature of the filtering operation leads to significant computational economies. However, linear filtering techniques are generally very complex and have achieved only moderate success. Nonlinear filters have proven to be exceptionally useful in many signal and image restoration applications. In particular, rank-order-based filters are well known for their ability to successfully treat heavy tailed noise and non-stationary

signals. The first and most well known of these rank-order-based filters is the median filter. Building on the success of the median filter, many more sophisticated rank-order filters have been proposed, including multistage median filters[1, 2, 53], center weighted median filters[29, 38], general weighted median and weighted order statistic filters[36, 78], stack filters[16, 17, 46, 75], permutation filters [4], and rank conditioned rank selection filters[30]. These filters have primarily been utilized as smoothing filters in restoration applications where a signal is corrupted by noise. However, they do not perform edge enhancement well. The result being that in an edge region, which is comprised of nondecreasing and non-increasing samples, the ranks of the input samples remain the same for all observation window locations. Thus, they do not help to identify the "side" of an edge to yield gradient enhancement[31].

There are some edge enhancement rank selection filters, such as the comparison and selection (CS) filter[44], the lower-upper-middle (LUM) filter[29] and the weighted majority of samples with minimum range (WMMR) filter[47]. For CS and LUM filters, the observation sample mean is compared to a specified sample within the observation window to determine which rank ordered sample to output. This comparison helps to identify the "side" of an edge on which the filter's window lies. Similarly, WMMR filters use rank ranges to delineate different regions of an edge. Having partitioned the edge into different regions, an appropriate output sample is chosen in each region so as to increase the edge gradient.

Another type of non-linear filter is the neural network filter [57]. The input of the neural network is formed by the nine first differences calculated using adjacent pixels. This feed-forward multi-layer perceptrons were trained with back-propagation, improved with the acceleration technique proposed by Silva and Almeida[54], and it can provide good location and low distortion of edges and also a good response to corners. But the performance of large input support neural networks is less than expected due to the complexity of neural networks.

2.1.7 Edge linking

Edge detection algorithms typically are followed by edge linking which is designed to assemble edge pixels into meaningful boundaries. If the edges are reliably strong, and the noise level is low, one can threshold an edge image and thin the resulting binary image down to single-pixel-wide closed, connected boundaries. However, such an edge image will have gaps that must be filled under less than ideal conditions. There are several techniques suitable for this purpose such as heuristic search techniques, curve fitting techniques and Hough Transform techniques[14]. Heuristic search strategy using A^* algorithm was first used for boundary detection in 1972[49]. Martelli showed that the problem of boundary detection can be brought back to the problem of finding the minimal cost path in a weighted and directed graph, with positive costs. M. Salotti and M. Hatimi[62] have proposed a new heuristic search strategy and shown that a cost function with Gaussian

curvature is more appropriate to develop only the best paths. The advantage of heuristic search strategies holds in the possibility of adding many contextual constraints to the detection, thus making the search adaptive to the application. The major problem is to find a suitable cost function. Also heuristic search techniques become computationally expensive if the edge quality function is complex and the gaps to be evaluated are many and long.

2.2 Neural Networks

Artificial neural networks (ANNs) are systems that are deliberately constructed to make use of some organizational principles resembling those of the human brain. They represent a promising new generation of information processing systems. ANNs are good at tasks such as pattern matching and classification, function approximation, optimization, vector quantization, and data clustering because of their architecture. ANNs have a large number of highly interconnected processing elements (nodes or units) that usually operate in parallel and are configured in regular architectures. The interconnections lead to distributed knowledge representation. The collective behavior of an ANN, like a human brain, demonstrates the ability to learn, recall, and generalize from training patterns or data.

ANNs have a great potential for parallelism since the computations of the components are largely independent of each other. In addition to the high computation rates provided by the massive parallelism, neural networks provide a greater de-

gree of robustness than traditional sequential computers because there are many processing nodes. Damage to a few nodes or links may not affect the overall performance of the net significantly. Neural network memory has the ability of generalization which is one of the most attractive features of neural networks. This enables a model to function competently throughout the pattern space though it has learned from observing only a limited body of examples.

2.2.1 Multi-layer feed-forward neural networks

Multi-layered neural networks are an efficient way to implement nonlinear classifiers. Multi-layered neural networks usually consist of many neuron units. These units are arranged into several layers. The input layer receives the signal and sends its output to the hidden layers. There may be several hidden layers between the input layer and the output layer. The output layer receives the signal from the last hidden layer and produces the final output. The performance of feedforward neural networks has been greatly enhanced because of those hidden layers. The basic operation of each hidden node and output node is to map the weighted sum of output from the previous layer according to an activation function. The multi-layer neural networks are non-linear as the activation function of nodes is a sigmoid function (or Gaussian function). The activation function introduces non-linear into the network, without which the network would not be more powerful than a plain perceptron. The architecture of this kind of neural network is especially useful for

static classification tasks.

The back-propagation (BP) learning algorithm is one of the most important historical developments in neural networks [9, 15, 42]. This learning algorithm is applied to multilayer feedforward networks consisting of processing elements with continuous differentiable activation functions. According to the BP algorithm, the neural network is initialized with random weights. Then, all training data are presented repeatedly to the network. Weights are adjusted after every trial in order to minimize a function of the error between the actual output produced by the network and the desired output.

There are several problems with multi-layered networks. Firstly, feed-forward network learned by BP have slow convergence during training. When complex decision regions are required, learning in multi-layered networks is slow. Secondly, the number of input nodes must be large enough to form a decision region. However, it must not be so large that the weights can not be reliably estimated from the available training data. Therefore, each net is only capable of performing a specific task and any expansion of the original task requires an extensive modification to the structure of the network. Besides, there is no a solid theory to guide designers in determining the number of hidden layers and the number of nodes in each hidden layer for a certain application.

2.2.2 Other feedforward neural networks

There are several different single-layer and multi-layer feedforward neural networks that have also attracted much attention for various applications.

Functional-link networks[54] are single-layered neural networks that are able to handle linearly non-separable tasks using an appropriately enhanced input representation. The key point of this method is to find a suitable enhanced representation of input data. Additional input data used in the scheme usually incorporate high-order effects and thus artificially increase the dimensions of the input space. The expanded input data are used for training. The additional higher-order input terms are chosen such that they are linearly independent of the original pattern components. In this way, the input representation is enhanced and linear separability can be achieved in the extended space. Since the functional-link network has only one layer, it can be trained using the simple delta learning rule. Hence, the learning speed of it is much faster than that of BP networks.

The tree neural network (TNN), like multilayer feedforward neural networks, is a popular approach to the pattern recognition problem[23, 63]. The basic idea is to use a small multilayer network at each decision node to extract non-linear features. TNNs exploit the power of tree classifiers to use appropriate local features at different levels and nodes of the tree. The learning algorithm for constructing a TNN consists of two phases. In the tree-growing phase, a large tree is grown by recursively finding splitting rules until all terminal nodes have pure or nearly pure

class membership or cannot be split further. In the tree-pruning phase, a smaller tree is selected from the pruned subtrees to avoid overfitting the data. When solving difficult pattern recognition problems with complex-decision boundaries, the TNN demonstrated significant decreases in error rate and tree size relative to standard classification tree design methods. The TNN also yielded comparable error rates and shorter training time than a large BP network on the same problem.

2.2.3 Hopfield networks

The publication of Hopfield's seminal papers[32, 33] started the modern era in neural networks. His proposed networks, Hopfield networks, have been used in many applications, especially in associative memory and optimization problems. The typical Hopfield networks can be viewed as single-layer feedforward networks or termed recurrent networks. In this architecture, the nodes are organized as a fully connected layer where every node receives stimulus from all others. There is no self-feedback in a Hopfield network. The weight matrix is symmetric which means that the weights on the connections between two nodes are equal in both directions. The nodes also receive an external input. The values of nodes at any given time define the state of the network. The updating from one state to the other state can be in an asynchronous fashion or in a synchronous fashion. In an asynchronous fashion, only a single node is allowed to update its output for a given time. The next update on a randomly chosen node in a series uses the

already updated output. States change until the state is an equilibrium state of the network. For the case of synchronous update, the update produces a cycle of two states rather than a single equilibrium state. States change until a stable configuration is reached. It indicates that the synchronous update may cause the networks to converge to either fixed points or limit cycles.

The convergence of the neural state of Hopfield models to its stable state is based on the existence of an energy function which directs the flow in state space. The well-known Lyapunov stability theorem[6] is usually used to prove the stability of a dynamic system defined with arbitrarily many interlocked differential equations. The energy function associated with a Hopfield model is a Lyapunov function. Such a function depends on the current state as well as the weight matrix. To guarantee convergence, the weights must be designed such that any update in the network's state will decrease the energy or keep it unchanged. When the network is supposed to act as a content-addressable memory, the weight matrix is calculated by taking the outer product of each vector to be stored in the network with itself. Then all the resulting outer products are superimposed on top of each other.

The original Hopfield model is binary discrete Hopfield model. It can be generalized to a continuous model in which time is assumed to be a continuous variable and the nodes have a continuous, graded output rather than a two-state binary output. Hence, the energy of the network decreases continuously in time.

The continuous Hopfield networks show the same useful properties of a discrete model. Moreover, there is an analogous electronic circuit that uses nonlinear amplifiers and resistors for realizing continuous Hopfield networks. The continuous Hopfield model can be applied to combinatorial optimization problems such as NP-complete[76] problem. In this case, a suitable representation for the problem which corresponds to a Hopfield network should be found. Then, the network's energy function is designed in a way that reflects the constraints of the optimization problem so that the network stabilizes on a class of good solutions depending on its initial configuration.

In practice, the Hopfield network has several limitations. The major problem of Hopfield networks appears to be the local minimum problem. That is, the solution reached does not represent a global optimal solution to the energy function minimization. This is often due to the highly complex shape of the multi-dimensional energy function. However, the solutions achieved, although not optimal, are acceptable in a statistical sense. The Hopfield models give an excellent demonstration of how practical problems that are tremendously difficult can be attacked by neural networks.

2.2.4 Competitive learning and self-organization neural networks

Competitive learning of neural networks has been explored in one form or another over the years. Although the elementary competitive network and its principle is simple, it indeed leads to self-organization behavior. The main difference between feedforward networks and competitive networks lies in the lateral interconnections in competitive networks. These lateral interconnections provide a mechanism to incorporate consistent or contradictory relationships which naturally exist among objects, concepts, events, etc. At the computation level they are implemented as cooperative and competitive processes in neural networks. Typical models are McClelland and Rumelhart's competitive learning model[50], Kohonen's self-organization MAP[39], and Carpenter and Grossberg's ART[12, 23].

2.3 Fuzzy Neural Networks

Over the past decade, we have witnessed a very significant increase in the number of research and applications of fuzzy neural networks to various commercial and industrial products and systems. Neural networks are essentially low-level computational structures and algorithms that offer good performance in dealing with sensory data, while fuzzy logic techniques often deal with issues such as reasoning on a higher level than neural networks. However, since there are no capabilities of

machine learning, memory, or pattern recognition in fuzzy systems, it is difficult for a human operator to tune the fuzzy rules and membership functions from the training data set. Also, because the internal layers of neural networks are always opaque to the users, the mapping rules in the networks are not visible and are difficult to understand; furthermore, the convergence of learning is usually very slow and not guaranteed. Thus, a promising approach for reaping the benefits of both fuzzy systems and neural networks (and solving their respective problems) is to merge them into an integrated system. Fuzzy neural networks retain the basic properties and functions of neural networks with some of their elements being fuzzified. In this approach, a networks domain knowledge becomes formalized in term of fuzzy sets, later being applied to enhance the learning of the network and argument its interpretation capabilities.

2.3.1 Fuzzy logic controller (FLC)

Fuzzy logic control, initiated by the pioneering work of Mamdani and Assilian[48], has emerged as one of the most active and fruitful areas for research in the application ranges from industrial process control to medical diagnosis and securities trading. Many industrial and consumer products using this technology have been built. In contrast to conventional control techniques, FLC is best utilized in complex ill-defined processes that can be controlled by a skilled human operator without much knowledge of their underlying dynamics.

The basic idea behind FLC is to incorporate the expert "experience" of a human operator in the design of a controller for a process whose input-output relationship is described by a collection of fuzzy control rules (e.g., IF-THEN rules) involving linguistic variables rather than a complicated dynamic model. This utilization of linguistic variables, fuzzy control rules, and approximate reasoning provides a means to incorporate human expert experience in designing the controller.

The typical architecture of an FLC is comprised of four principle components: a fuzzifier, a fuzzy rule base, an inference engine, and a defuzzifier. The fuzzifier performs the function of fuzzification which is a subjective valuation to transform measurement data into valuation of a subjective value. Hence, it can be defined as a mapping from an observed input space to labels of fuzzy sets in a specified input universe of discourse. That is, a fuzzifier has the effect of transforming crisp measured data into suitable linguistic values. The fuzzy rule base stores the empirical knowledge of the operation of the process of the domain experts. Fuzzy control rules are characterized by a collection of fuzzy IF-THEN rules in which the preconditions and consequents involve linguistic variables. This collection of fuzzy control rules (or fuzzy control statements) characterize the simple input-output relation of the system. The inference engine is the kernel of an FLC, and it has the capability of simulating human decision making by performing approximate reasoning to achieve a desired control strategy. The generalized modus ponens (forward data-driven inference) plays an especially important role in this context.

The defuzzification is a mapping from a space of fuzzy control actions defined over an output universe of discourse into a space of nonfuzzy (crisp) control actions. It is utilized to yield a nonfuzzy decision or control action from an inferred fuzzy control action by the inference engine. Unfortunately there is no systematic procedure for choosing a defuzzification strategy. Two commonly used methods of defuzzification are the center of area (COA) method and mean of maximum (MOM) method[8, 43]. Another method, the basic defuzzification distribution (BADD), was proposed by Yager and Filev[20, 77] who provide an adaptive learning scheme to obtain the optimal defuzzification parameters.

Although approximately 1,000 commercial and industrial fuzzy systems have been successfully developed for the past few years and fuzzy set theory has grown to become a major scientific domain, fuzzy control systems have some problems and limitations[51]. One of the most important issues for a fuzzy control system is stability. Of the various existing methodologies for stability analysis of fuzzy systems, there is no theoretical guarantee that a general fuzzy system does not go chaotic and stays stable. Another limitation is that fuzzy systems lack capabilities of learning and have no memory. This is why hybrid systems, particularly neuron-fuzzy systems, are becoming popular for certain applications. Besides, determining or tuning good membership functions and fuzzy rules are not always easy. Also, there is a general misconception of the term "fuzzy" as meaning imprecise or imperfect. Many professionals think that fuzzy logic represents some magic

without firm mathematical foundation. Because of these problems and limitations, researchers are motivated to establish various forms of hybrid systems by combining fuzzy logic and other areas such as neural networks and genetic algorithm to achieve a better performance.

2.3.2 Fuzzy neural networks

Fuzzy neural networks are in fact fuzzified neural networks and thus are inherently neural networks. Each part of a neural network (such as the activation function, aggregation functions, weights, input-output data, etc.), each neural network model, and each neural learning algorithm can possibly be fuzzified. There are three types of fuzzy neurons. The first kind of neuron has n nonfuzzy inputs with a single output in the interval [0.1], which may be considered the "level of confidence". The weights are fuzzy set, that is, the weighting operations are membership functions. The aggregation operation may use any aggregation operator such as min (minimum) or max (maximum), and any other t-norms or t-conorms. The second type of fuzzy neuron is similar to the first type of fuzzy neuron except that all the inputs and outputs are fuzzy sets rather than crisp values. Besides, the weighting operation is a modifier to each fuzzy input instead of a membership function. In the third type of neuron, the input-output relation of the fuzzy neuron is represented by one fuzzy IF-THEN rule.

The typical fuzzy neural network model is a fuzzy multilayer feedforward net-

work with a back-propagation learning algorithm[35]. The fuzzy neural network with this structure maps a fuzzy input vector (with linguistic value) to a fuzzy output. It can handle fuzzy input vectors using the fuzzy activation functions on fuzzy numbers to classify n-dimensional fuzzy vectors. The energy function is defined by the fuzzy actual output and the corresponding nonfuzzy target output that indicates the correct class of the fuzzy input vector. As the training procedure is viewed as an extension of BP, it still has problems such as slow convergence and relearning. Another multilayer feedforward FNN was proposed by [40] as a connectionist fuzzy classifier (CFC). CFC has one more layer used for computing the reference similarity from learned samples. Also, it groups the input nodes as a subset of features which may together represent a subconcept in some cases. As CFC employs a hybrid supervised/unsupervised learning scheme to organize referenced pattern vectors, it not only overcomes the local minimum and long training time problems, but also avoids the disadvantage of the huge storage space requirement of the probabilistic neural network.

The self-organizing multilayer fuzzy neural network proposed by Ashish [3] for object extraction uses a four-layer feedforward fuzzy neural network structure. But the self-organizing learning algorithm is applied. Pattern clusters are implemented as fuzzy sets using a membership function with a hyperbox core that is constructed from a min point and max point. The min-max points are determined using the fuzzy min-max learning algorithm. The network stabilizes into pattern clusters in

only a few passes through a data set. It can be reduced to hard cluster boundaries that are easily examined without sacrificing the fuzzy boundaries and provides the ability to incorporate new data and add new clusters without retraining.

The adaptive resonance theory (ART) is a very important theory which is used in neural networks for guiding unsupervised learning. G.A.Carpenter and S.Grossberg pioneered the introduction and the development of ART. They also introduced fuzzy set theory into ART1 and developed the fuzzy ART system, called CGR fuzzy ART[13]. There are two options for this fuzzy ART algorithm: the *fast-commit-slow-record option* and the *input normalization option*. For the first option, fast learning enables a system to adapt quickly to inputs that may occur only rarely and may require immediate accurate performance. The slow-record operation prevents features that have already been incorporated into a category's prototype from being erroneously deleted in response to noise or partial inputs. For the second option, a preprocessing step, called complement coding, uses on-cell and off-cell responses to prevent category proliferation. The complement coding normalizes input vectors while preserving the amplitudes of individual feature activations. The intersection operator used in ART1 learning is replaced by the MIN operator of fuzzy set theory. Learning is stable because all adaptive weights can only decrease in time. However, this fuzzy ART has some problems such as hyperbox cluster overlap which results in pattern cluster ambiguity. There is another fuzzy neural network, Fuzzy Min-Max Clustering Neural Networks(FMMCNN)[66], which also

grew out of the fuzzification of the ART1 network. Compared with CGR fuzzy ART, FMMCNN overcame the hyperbox cluster overlap problem. Additionally, the CGR fuzzy ART bounds the size of hyperboxes through the matching function. The FMMCNN uses an explicit calculation of the size of the hyperbox to bound the size.

To obtain precise training data is difficult or more expensive, so a reinforcement learning was proposed[45]. The proposed reinforcement structure/parameter learning algorithm basically utilizes the techniques of temporal difference, stochastic exploration, and the on-line supervised structure/parameter learning algorithm. It used two NN-based fuzzy logic control systems integrated together. One is for the fuzzy controller (action network), and the other is for fuzzy prediction (evaluation network). Learning will be performed by both networks at the same time and only conducted by a reinforcement signal feedback from the external environment. The reinforcement training data is rough, it is just "evaluative". Besides, reinforcement learning can solve the long time delay problem.

Chapter 3

Adaptive Fuzzification

A fuzzifer used to fuzzify the input space is a basic component of a fuzzy system. A natural and simple fuzzification approach is to convert a crisp value into a fuzzy singleton within the specified universe of discourse. In a more complex case where observed data are disturbed by random noise, a fuzzifer should convert the probabilistic data into fuzzy numbers, that is, fuzzy data. In large-scale systems, some observations relating to the behavior of such systems are precise, others are measurable only in a statistical sense, and some, referred to as "hybrids", require both probabilistic and possibilistic modes of characterization. In this chapter, a method to adaptively fuzzify input patterns of an original image is presented which includes three main steps: input pattern generation, adaptive fuzzification and parameter estimation.

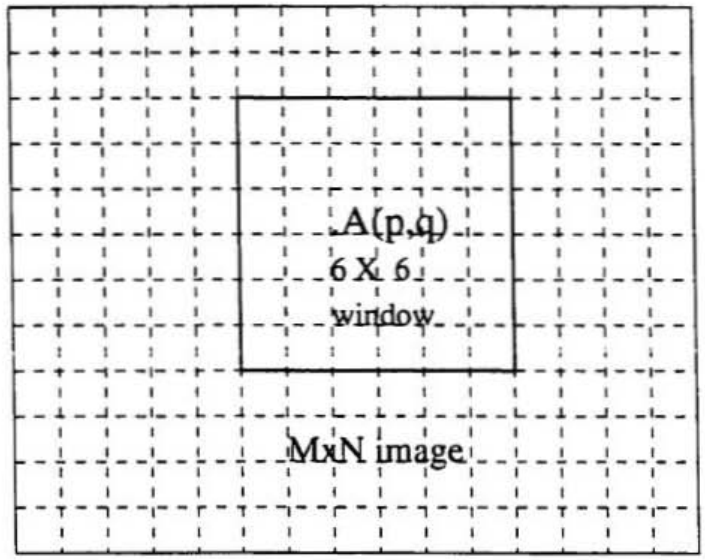


Figure 3.1: Matrix representation of an image

3.1 Input Pattern Generation

The original input image is represented by an $M \times N$ matrix A. M and N are the row and column sizes of the given image respectively. Each element corresponds to a pixel with its grey-value in [0, 255]. That is, A(p,q) with value 150 represents the grey value of the pixel at position of the p-th row and q-th column position of the input image with gray value 150. (Figure 3.1).

Window size 6×6 is chosen in the proposed system. If the window size is too small, the information is not sufficient to make a good judgement for distinguishing patterns. But if the window size is too larger, the window may contain more than one edge. Hence, they can not be detected. Also, the neural networks need more neurons to be constructed. To the system, the local edge pattern is input in an 6×6 window. A 2-D grid system is used in which an edge element is situated at

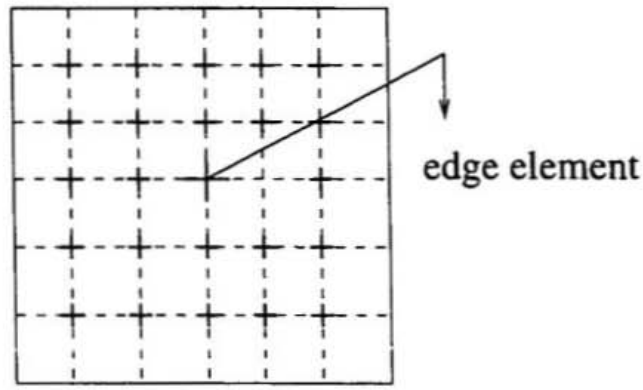


Figure 3.2: Edge element represented by 2-D grid system

the center of four adjacent image pixels shown in Figure 3.2.

If the window is too small, the length of the edge contour in the local edge pattern will be too short to be of any significance, which makes the fuzzy neural networks system less efficient. On the other hand, a large window allows better interpolation of missing edge elements but results in more complicated local edge patterns and a very complex neural net design.

A neural net processes the information from the input pattern in a vector form. For the grid in consideration at position (p,q) of input image, the corresponding window is represented as a vector, then the pattern is:

$$A_{pq} = \{A(0,0), A(0,1), ..., A(5,5)\}$$

Here, A(0, 0) corresponds to A(p-3, q-3),

A(0, 1) corresponds to A(p-3, q-2),

. . . . . ,

A(5, 5) corresponds to A(p+2, q+2).

3.2 Adaptive Fuzzification

Adaptive fuzzification applied on patterns plays a very important role in the proposed system. When considering an edge in a window with northern orientation (Figure 3.3), it can be represented by its grey value as follows:

$$f(x) = \begin{cases} g_2 & \text{if } x > 0 \\ (g_1 + g_2)/2 & \text{if } x = 0 \\ g_1 & \text{if } x < 0 \end{cases}$$

where g_1 is a grey value which indicates the relative dark part, g_2 is a grey value which indicates relative bright part, x as a position of pixels (Figure 3.4(a)), and the origin of the coordinate system is the center grid of the window (Figure 3.3).

The edge image with normal noise is modeled as figure 3.4(b).

For a window with an edge having a relatively bright component in the left part and with a relatively dark component in the right part, the difference of the grey values between two parts which can be distinguished by human eyes in the noise case is more than 20(threshold). That is:

$$|g_1 - g_2| > threshold \quad (g_1, g_2 \in [0, 255])$$

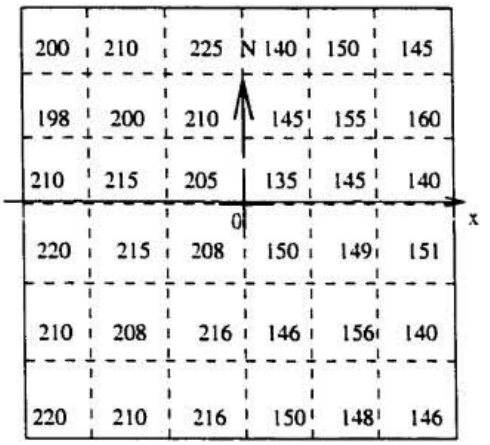
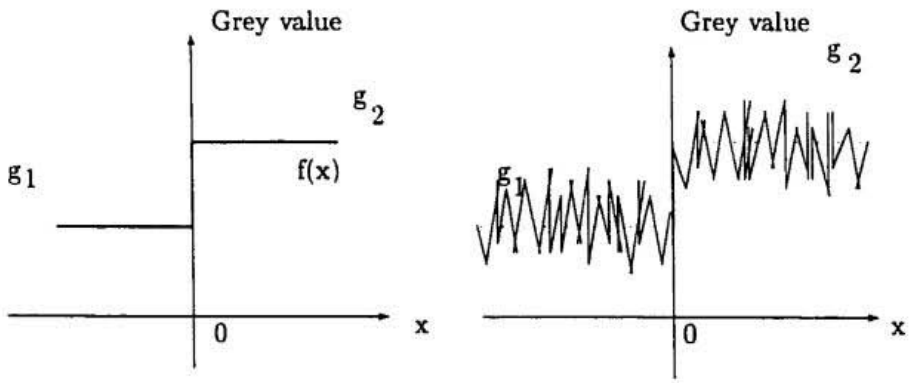


Figure 3.3: Edge pattern with northern orientation



(a) Image without noise (b) Image with noise

Figure 3.4: Grey value representation of an image

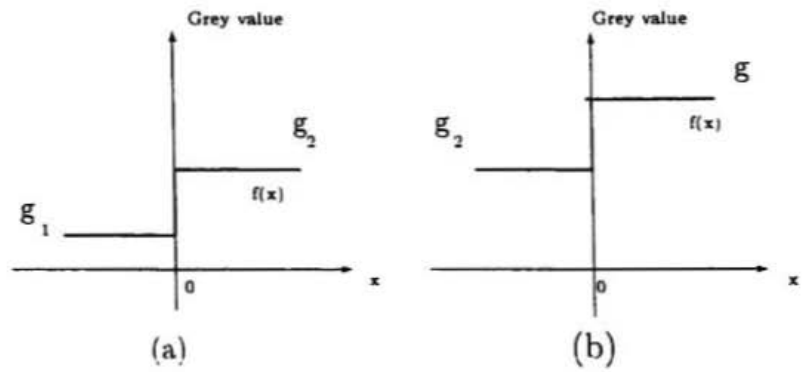


Figure 3.5: Grey value representation of two images in the same class

Hence, there are $1+2+...+234+235$ possible combinations for a window. So, the number of samples for one class is:

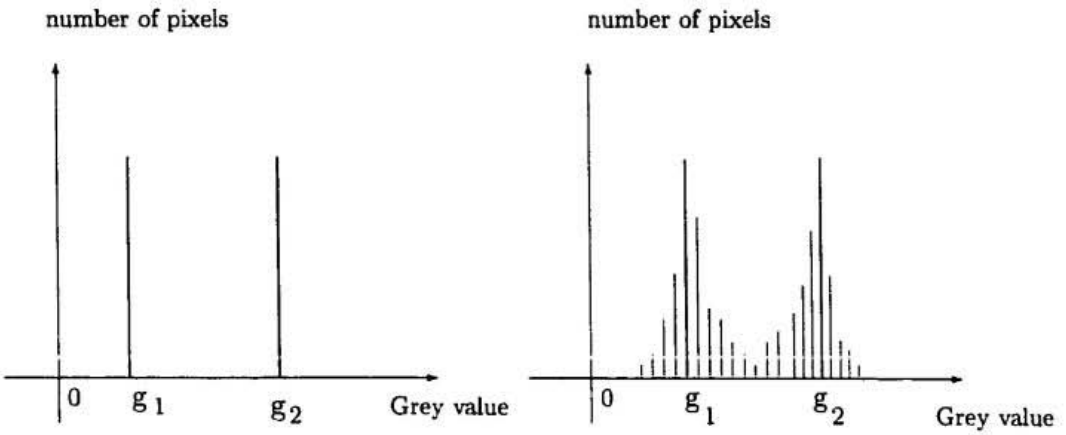
$$N = \frac{235+1}{2} * 235 = 27730$$

Because there are ten classes to be classified, more than two hundred thousand samples are needed to train neural networks to obtain a good result.

However, it is noticed that the following two samples belong to the same case. (Figure 3.5)

For grey value g_2, it represents relative dark in figure 3.5(a), but relative bright in figure 3.5(b). So the "bright" or "dark" is a fuzzy concept depending on the given patterns. For this reason, it is impossible for us to find a relationship between absolute grey value and degree of "bright/dark" and directly apply a global membership function. Hence, an adaptive fuzzification processing is proposed in this research.

When there is an edge in a window, the histogram can be used to represent



(a) Image without noise (b) Image with noise

Figure 3.6: Histogram for the window

the relationship between the number of pixels in a window and their grey values. (Figure 3.6(a)) For the image with normal noise, the grey value of the relatively bright part should be around g_1, and the grey value of the relatively dark part should be around g_2. (Figure 3.6(b))

Based on the above analysis, the following fuzzy membership function is created (Figure 3.7). Here,

$$f_a(A(p,q)) = \begin{cases} 1 & \text{if } A(p,q) < g_1 \\ (A(p,q) - g_2)/(g_1 - g_2) & \text{if } g_1 < A(p,q) < g_2 \\ 0 & \text{if } A(p,q) > g_2 \end{cases}$$

By using the above membership function, each pixel in the input pattern is fuzzified and represented by its fuzzy measurement. The value of the pixel closes to 1 when it is relatively bright and closes to 0 when it is relatively dark. After

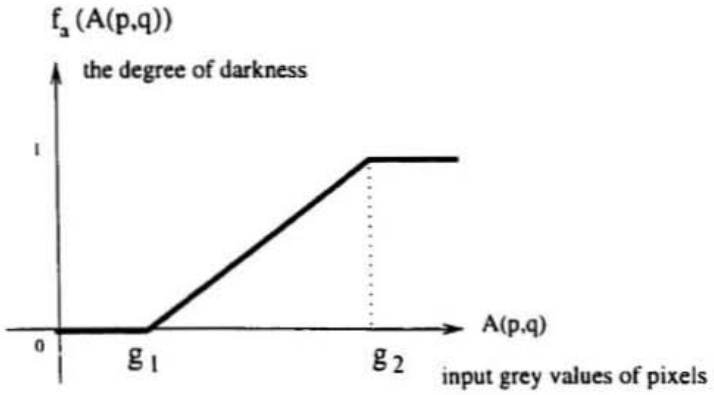


Figure 3.7: Adaptive fuzzy membership function

the adaptive fuzzification, the sample with grey values 200 on the left part and grey values 180 on the right part is the same as the sample with grey values 180 on the left part and grey values 160 on the right part. So the number of samples for training is reduced significantly.

3.3 Parameter Estimation

The fuzzy membership function depends on each local pattern. For an edge in an image with normal noise (see Figure 3.6(b)), the histogram follows a normal distribution (Figure 3.8).

The g_1 and g_2 can be determined by using estimation and learning methods, such as the Parzen Window approach. However, it is too complicated and time-consuming especially when such a technique is applied on large images. Since neural networks will be used to do the further processing, the estimation procedure can be simplified and the values of g_1 and g_2 can be roughly set by using the

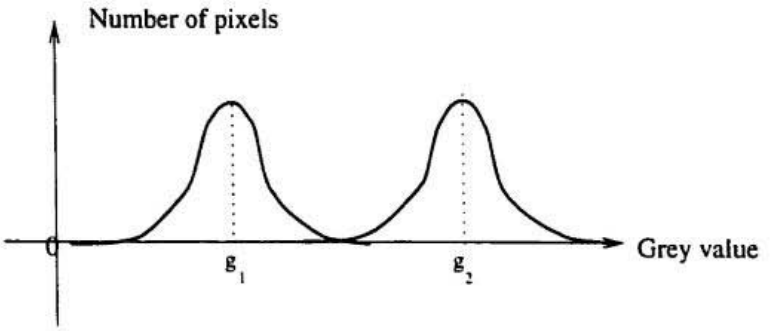


Figure 3.8: Normal distribution for an edge pattern

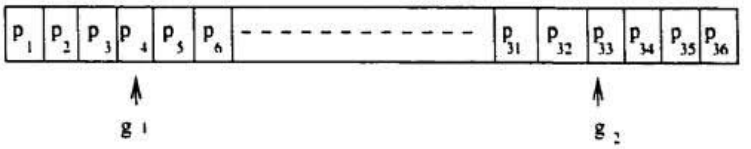


Figure 3.9: Sorted pixels with less than 30% of noise

following approach.

Let the proposed system allow η % input noises. For a $k \times k$ window chosen, the system allows $k^2 * \eta\%$ noise pixels. Assume that the half of the noise pixels' values are either bigger than g_2 or less than g_1, and the other half of the noise pixels' values are between g_1 and g_2. Therefore, the number of pixels with grey values less than g_1 or greater than g_2 is $\frac{k^2*\eta\%}{4}$. The estimation procedure can be done by first sorting the values of pixels of the input pattern in an ascending order, then, setting g_1 at the position of $1 + \frac{k^2*\eta\%}{4}$ and g_2 at the position of $k^2 - \frac{3}{4} * k^2 * \eta\%$ respectively. For example, if 30% of noise is allowed, the pixels in a 6×6 window sorted in an ascending order represented by $P = p_1, p_2, ..., p_{36}$, g_1 should take the fourth element, and g_2 should take the 33rd element of the sorted array (Figure 3.9).

3.4 Algorithm for Adaptive Fuzzification

Suppose the gray values of input pixels in a $k * k$ size window and the percentage of noise be η. The algorithm to obtain g_1 and g_2 described at 3.3.2 is:

Procedure *AdaptiveFuzzification*

{

Create an array $Temp[K * K]$ to store the elements in the window;

Sort the array *Temp* in an ascendant sequence;

{ Calculate the positions (m and n in the array) for g_1 and g_2 }

$m = 1 + \eta * k^2/4$; $\quad\quad$ $n = k^2 - 3 * k^2 * \eta/4$;

{ Set g_1 and g_2 }

$g_1 = temp[m]$;

$g_2 = temp[n]$;

}

Chapter 4

Window Division

In general, ten patterns shown in figure 4.1 are considered as typical edge patterns in this application. The common property of these edge patterns is that the half part of pattern is relatively bright and the other half part is relatively dark. In order to take advantage of the structure information, a window division technique is introduced in this research. The input window is divided into eight groups corresponding to blocks in the input pattern. For the purpose of detecting two different kinds of edges, namely, edges in the horizontal and vertical orientation or edges in the diagonal orientation, the pattern is divided into four rectangular blocks and also into four triangular blocks. The detail discussion is given in the following two sections.

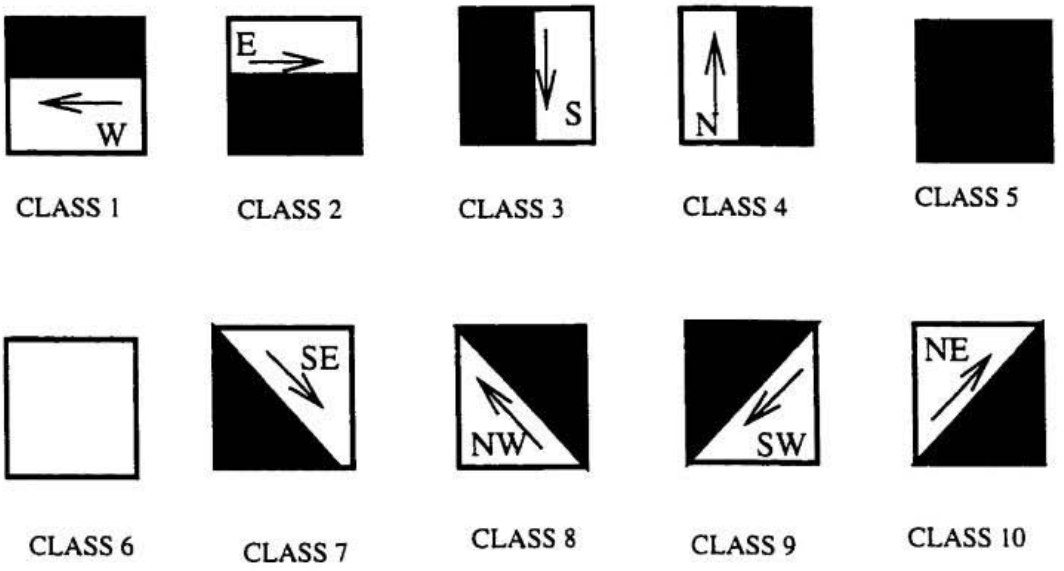


Figure 4.1: Ten edge patterns (eight edge patterns and two non-edge patterns)

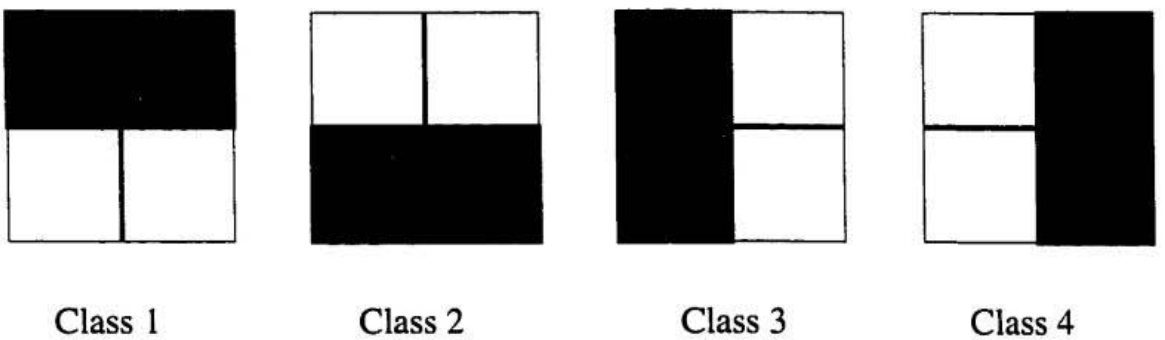


Figure 4.2: Window divided into four square blocks

4.1 Window Divided into Four Square Blocks

Considering a vertical or horizontal edge pattern(class one to class four), the window is first divided into four square blocks (Figure 4.2).

There are two blocks which are relatively dark in a window. For edge patterns which belong to class one or class two, the two dark blocks are either on the top half or bottom half part. For edge patterns which belong to class one or class three,

though one dark block (block 1) and one bright block (block 4) are at the same part of the window, the positionsof another dark part and bright part are different: block 2 is dark and block 3 is bright for class one, and block 2 is bright and block 3 is dark for class three. The same analysis can be applied to the combinations of any other two edge patterns which belong to class one to class four. Besides the dark edge pattern and the bright edge pattern, class five and class six can also be analyzed in the above way. The four blocks in different patterns have totally different darkness, and there are two blocks which have different darkness from above four patterns which belong to class one to class four. Hence, this kind of window division is suitable for patterns belonging to class one to class six.

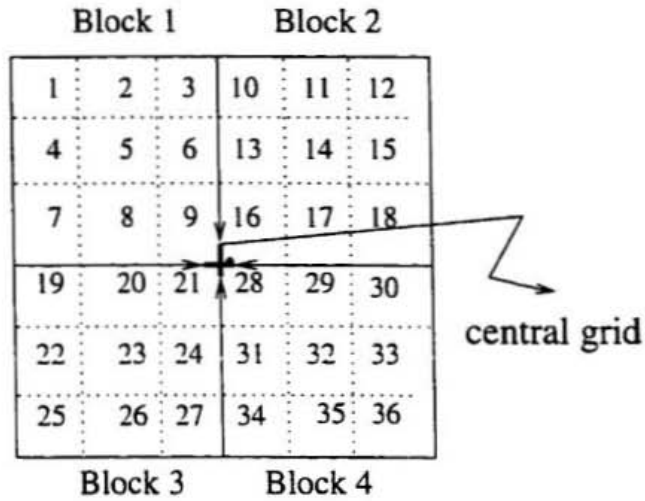
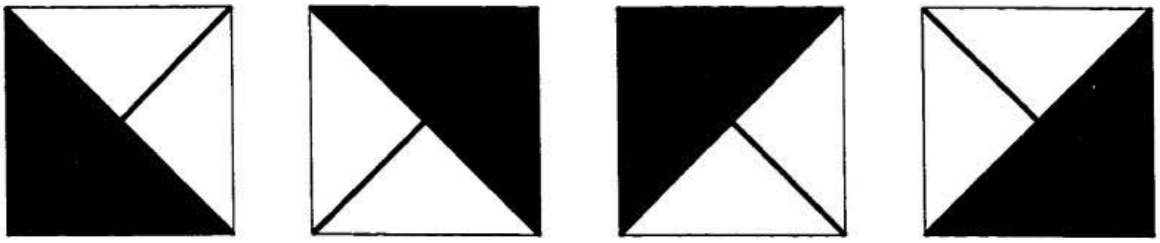


Figure 4.3: Reordered sequence of elements (square blocks)

As the elements of each pattern will be the input of the neural network and they have been grouped based on the block, the sequence of elements in the input vector has to be re-ordered. The actual input vector corresponding to the four square blocks is represented above (Figure 4.3).

4.2 Window Divided into Four Triangle Blocks

A window which is divided into four square blocks is suitable for edge patterns with vertical and horizontal orientations. However, it is not efficient to edge patterns with diagonal orientations (class seven to class ten). For example, class seven and class eight have half relatively bright and half relatively dark in the block one and block four respectively. It is not easy to know whether the top right half part or bottom left half part is relatively bright in the block one. Hence, for these four



Class 7 Class 8 Class 9 Class 10

Figure 4.4: Window divided into four triangular blocks

kinds of edge patterns, the window should be divided into four triangular blocks (Figure 4.4).

But the elements on the two diagonal lines do not belong to any blocks. For example, elements A(0, 0), A(1, 1), and A(2, 2) will be in neither block 5 nor block 6 (Figure 4.5).

If we take the southeast edge pattern (class 7) as an instance, putting the above three elements into block 5 will not cause problems if these three elements represent relatively black. But if these dark elements are put into block 6, the noise will be increased abruptly in block 6 as it was supposed this part would be relatively bright. A similar result can be induced when other patterns are analyzed. Hence, only the elements inside each block are considered in this window division. Therefore, the sequence of original elements are re-ordered shown in figure 4.6.

Based on the above two kinds of window division, there are sixty input elements which are grouped into eight blocks. They will be used as the input of proposed fuzzy neural networks.

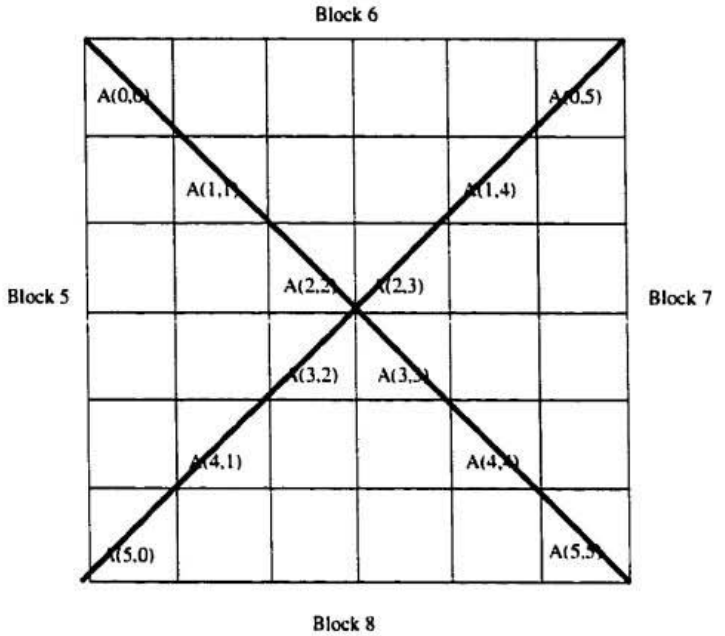


Figure 4.5: Elements on the boundary of the blocks

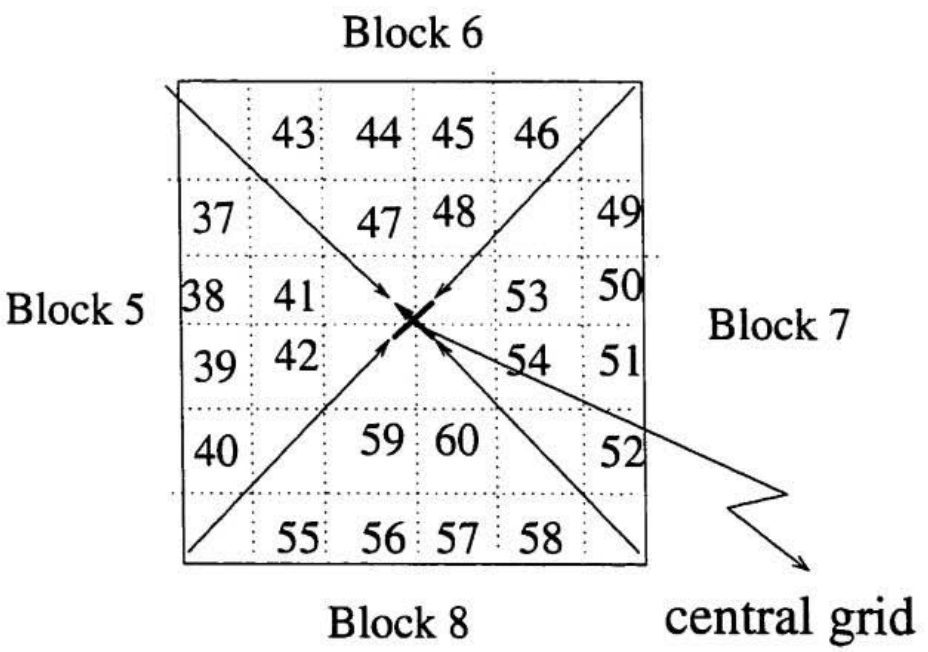


Figure 4.6: Reordered sequence of elements (triangle blocks)

Chapter 5

Fuzzy Neural Network

In chapter three and chapter four, the pre-processing procedure was described, the original inputs of a pattern have been fuzzified and expanded to sixty inputs which are divided into eight blocks. In this chapter, the structure of a fuzzy neural network designed specially for edge detection is proposed. The network can classify the input pattern in a noise corrupted image into non-edge or edge with one of eight orientations after the network is trained by typical edge or non-edge patterns.

5.1 The Overview of Fuzzy Neural Network

The structure of the proposed fuzzy neural network shown in figure 5.1 is a three-layer feedforward network. It consists of two subnets. Subnet 1 is used to identify class one to class six, and subnet 2 is used to classify class seven to class ten. Based on the window division, the inputs of subnet 1 have 36 elements which come from

four square blocks, and the inputs of subnet 2 have 24 elements which come from four triangular blocks. Therefore, there are sixty neurons at the first layer called the input layer. The second layer is a hidden layer. Every two neurons as a group connect to neurons, which accept inputs from the corresponding block, at layer one. One neuron, marked A, is used to measure the darkness of the corresponding block, and the other, marked B, is used to measure the brightness of the corresponding block. Based on the discussion on the chapter four, there are eight edge patterns. Therefore, the inputs are divided into eight groups. The number of neurons at the hidden layer is set as sixteen.

Taking the neurons with inputs coming from block 1 at layer one and the neurons of group 1 at layer two as an example, the links between these two layers are illustrated in figure 5.2(a). The neurons at layer 1 are fully connected to neurons at layer two. The inputs to layer one represent the darkness of the pixels in block 1. The neuron A at layer two provides the fuzzy measurement of how dark the block 1 is, and the neuron B measures how bright the block 1 is.

If all the values of nine inputs approach to 1(relatively dark) , this block is dark. So the output of neuron A approaches to 1. On the other hand, if input values approach to 0, there is little possibility that this block is dark. So the output of neuron A approaches to 0. Therefore, the fuzzy membership function applied on neuron A should have the shape of f_d shown on figure 5.2(b). For the other neuron, neuron B at layer two, a similar analysis can be used such that the neuron

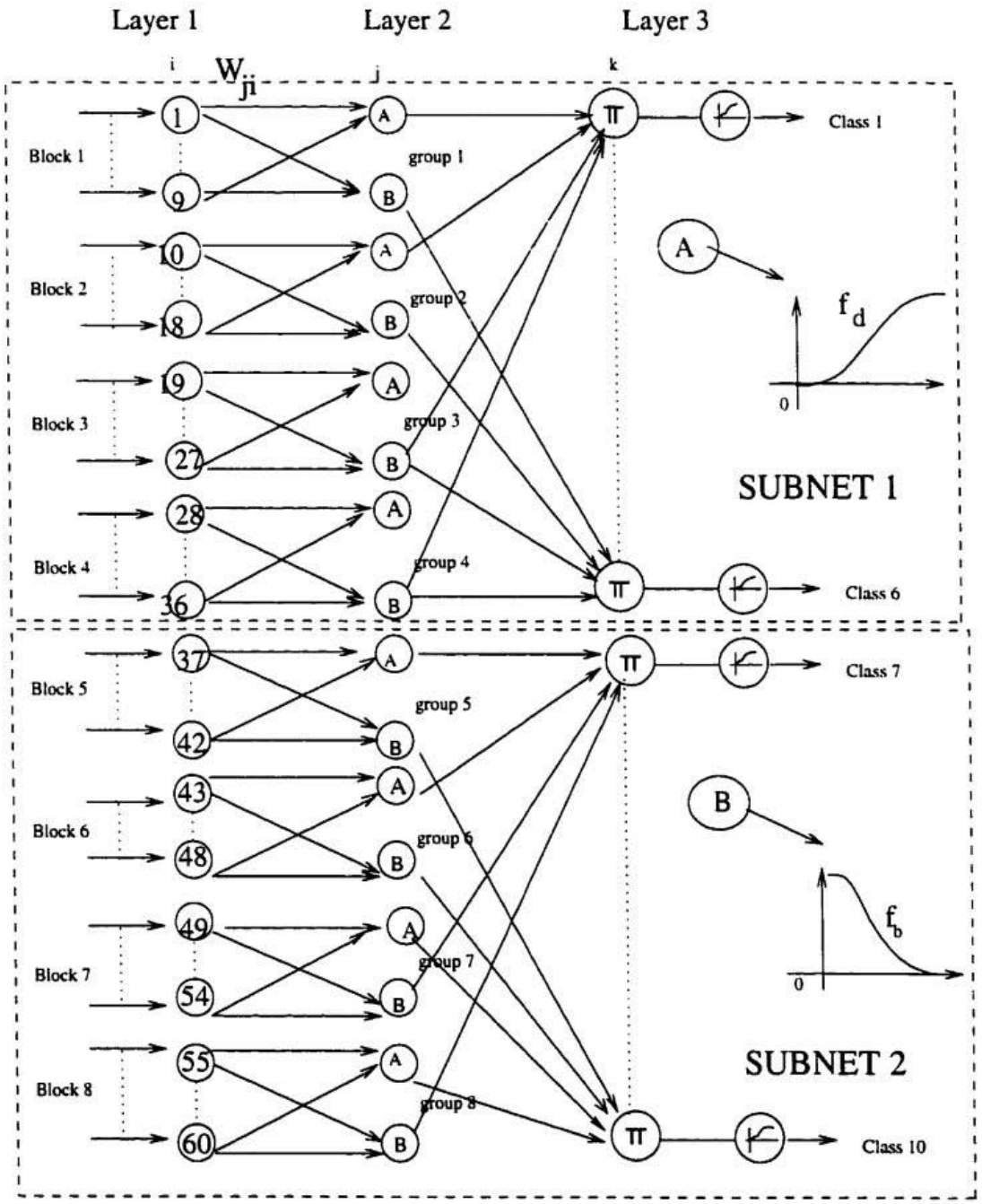


Figure 5.1: The structure of FNN for edge detection

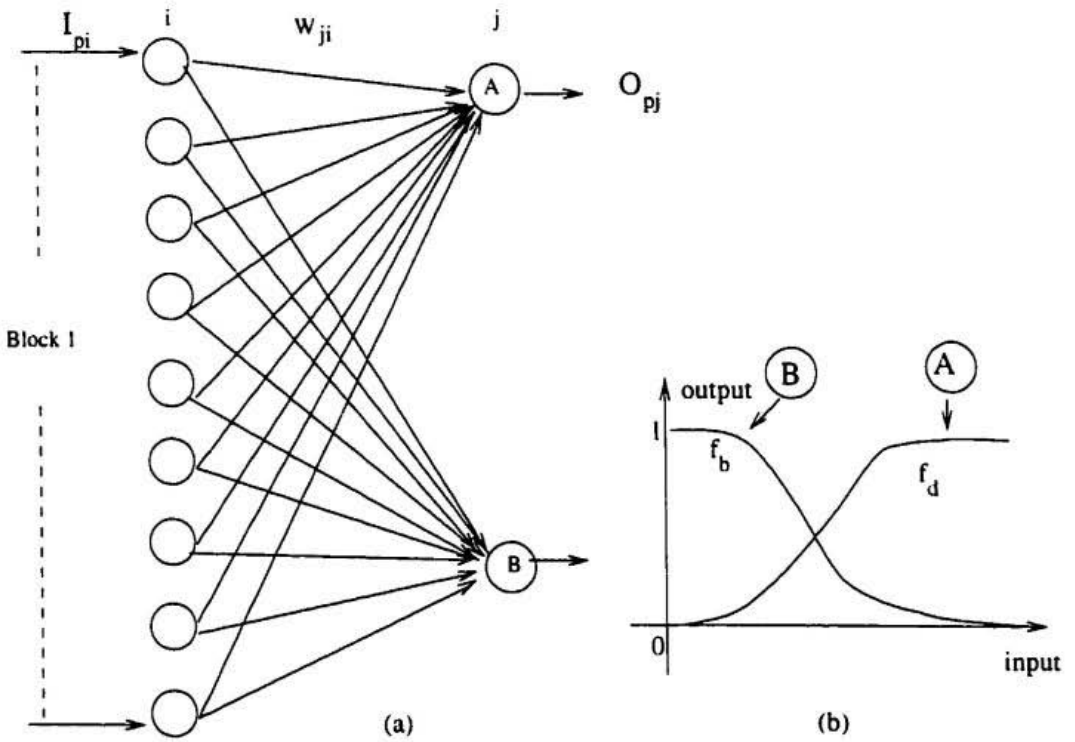


Figure 5.2: The connection between block 1 at layer 1 to layer 2

B should use the fuzzy membership function which has the shape of f_b.

A similar concept is applied on the other seven blocks of layer one and the seven corresponding groups of layer two. The fuzzy measurement structure is used to obtain the characteristic features for the different edge patterns. Hence, the outputs represent the measurement of the certainty of the features for each block.

The third layer is used to model human decision making within the conceptual framework of fuzzy logic and approximate reasoning. Here, neurons perform fuzzy intersection operations. The output of each neuron at layer 3 represents the certainty that this pattern belongs to the corresponding class. For example, the output value of the first node at layer 3 is 0.9 which means the certainty of the input pattern belongs to class one is 0.9. Every neuron connects four neurons in the layer two which come from four groups respectively. Which four neurons are linked to the neuron at the layer three depends on which class the output of this neuron represents.

The net input of the first node at layer three comes from four neuron outputs at layer two which correspond to four square blocks respectively(see Figure 5.3). Two neurons are of type A, and the other two are of type B. If these four outputs all approach to 1, it means that block 1 and 2 are relatively dark, and block 3 and 4 are relatively bright. Hence, the input pattern is definitely an edge pattern which belongs to class 1. The outputs of this layer provide the final fuzzy measurements of the classification.

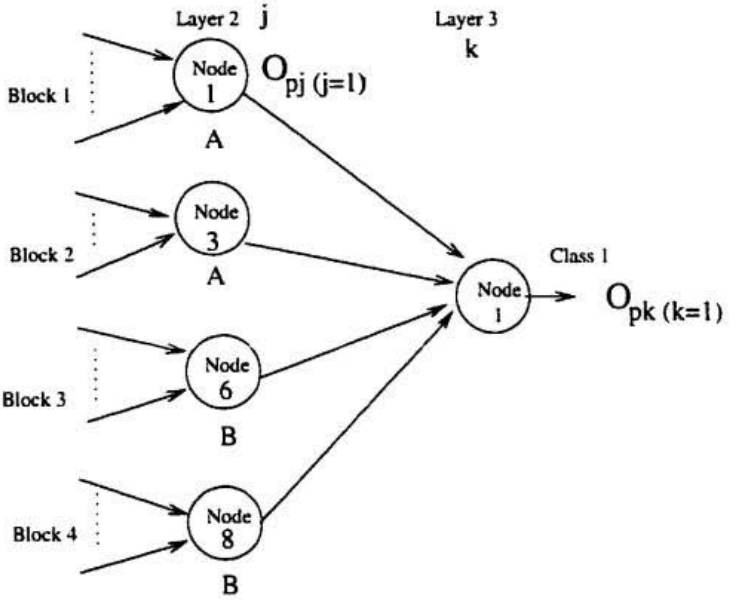


Figure 5.3: The connection between layer 2 and node 1 at layer 3

5.2 Models of Fuzzy Neurons

In fuzzy neural networks, the numerical control parameters in neurons and the connection weights may be replaced by fuzzy parameters. Fuzzy neural networks have greater representation power, higher training speed, and are more robust than conventional neural systems. In fact, fuzzified neural networks are inherently neural networks. Each part of a neural network (such as the activation function, aggregation function, weights, input-output data, etc.) can possibly be fuzzified. In this section, two different models of fuzzy neurons from which the proposed fuzzy neural network is built are discussed. One is the fuzzy neuron with fuzzy signals (fuzzy neuron - model I), and the other is the fuzzy neuron described by fuzzy logic equations (fuzzy neuron - model II) [40,41].

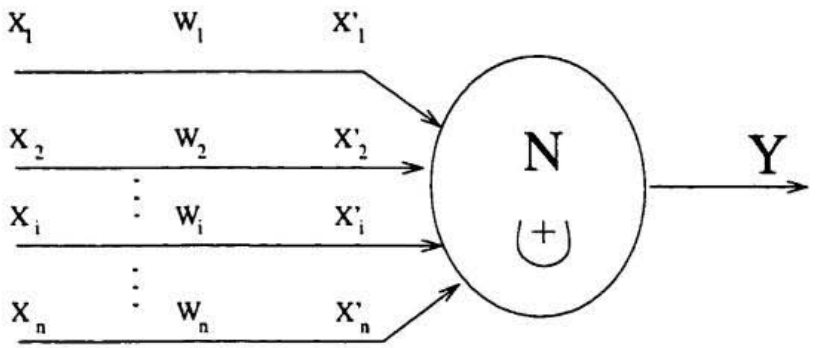


Figure 5.4: Fuzzy neuron - model I

5.2.1 Fuzzy neuron - model I

This type of fuzzy neuron, denoted by N (Figure 5.4), has n fuzzy inputs $x_1, x_2, ..., x_n$. Each fuzzy input x_i undergoes a weighting operation which results in another fuzzy set $x_i' = W_i * x_i$ for some operator $*$. (W_i is the weight, $1 \leq i \leq n$). All the modified inputs are aggregated together to produce a single output in the interval [0. 1] which may be considered the "level of confidence". The mathematical representation of such a fuzzy neuron N is written as:

$$x_i' = W_i * x_i \qquad i = 1, 2, ... n$$

$$Y = u_N(x_1' \uplus x_2' \uplus ... \uplus x_n')$$

where Y is the fuzzy set representing the output of the fuzzy neuron, x_i and x_i' are the ith input before and after the weighting operation, respectively. W_i is the weight on the ith synaptic connection, $\uplus$ is the aggregation operator, and $u_N(.)$ is the membership function of the neuron.

Based on the structure analysis and the model of neuron discussed, the proposed fuzzy neural network built from this type of neurons at layer two forms a fuzzy rule base.

The net input of layer two can be obtained by using the normalized weighted sum of the output from layer one. That is:

$$net_j = \sum_{i=1}^{n} \frac{w_{ji}}{\sum_{i=1}^{n} w_{ji}} I_i = \sum_{i=1}^{n} \hat{w}_{ji} I_i$$

where net_j is the net input of the j-th unit on layer two. w_{ji} is the weight from the j-th unit on layer two to the i-th unit on layer one. The output of node j at the second layer is:

$$O_{pj} = f(net_{pj})$$

Here, the function $f(.)$ is a membership function which will be discussed in 5.3.

5.2.2 Fuzzy neuron - model II

This type of fuzzy neuron with n fuzzy inputs and one fuzzy output is shown in figure 5.5. The input-output relation of the fuzzy neuron is represented by one fuzzy IF-THEN rule:

IF X_1 AND X_2 AND ... X_n, THEN Y,

where $X_1, X_2, ..., X_n$ are the current inputs and Y is the current output.

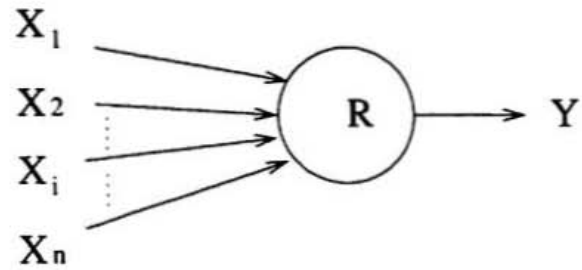


Figure 5.5: Fuzzy neuron - model II

The fuzzy neurons can be described by a fuzzy relation R,

$$R = f(X_1, X_2, ... X_n, Y),$$

where $f(.)$ represents an implication function. A fuzzy neural network constructed from these kinds of fuzzy neurons appears to be ideal for rule extraction.

According to the structure analysis, the proposed fuzzy neural network consists of this type of neurons at layer three. Links between layer two and layer three function as a connectionist inference engine. All the layer-2 nodes form a fuzzy rule base, and layer-3 links define the consequents of the rule nodes. As discussed before, the neuron at layer three performs the fuzzy intersection operation which is implemented by the t - norms "Algebraic Product" [41]. The inputs of node 1 at layer three come from the four nodes (node 1, node 3, node 6, and node 8) of layer two(see Figure 5.3). Hence, the pattern belonging to class 1 is decided by how dark the block 1 and block 2 are, and how bright the block 3 and block 4 are. It can also be interpreted as:

IF (block 1 is dark) AND (block 2 is dark)

AND (block 3 is bright) AND (block 4 is bright)

THEN (class 1).

That is:

If $O_1^{(2)}$ AND $O_3^{(2)}$ AND $O_6^{(2)}$ AND $O_8^{(2)}$ THEN $net_1^{(3)}$

Here the supscripts denote the layer number of the output. For example, $O_1^{(2)}$ means the output of the first node at layer two, and $net_1^{(3)}$ means the net input of the first node at layer three.

The above IF-THEN rule can be implemented by an "Algebraic product" as follows:

$$net_1^{(3)} = O_1^{(2)} * O_3^{(2)} * O_6^{(2)} * O_8^{(2)}$$

5.3 Fuzzy Membership Function

A crisp set is a collection of distinct objects. It is defined in such a way as to dichotomize the elements of a given universe of discourse into two groups: members and nonmembers. Finally, a crisp set can be defined by the so-called "characteristic function". Let U be a universe of discourse. The characteristic function $u_A(x)$ of a crisp set A in U takes its value in { 0, 1 } and is defined such that $u_A(x) = 1$ if x is a member of A and 0 otherwise. That is,

$$u_A(x) = \begin{cases} 1 & \text{if and only if } x \in A \\ 0 & \text{if and only if } x \notin A \end{cases}$$

It is noted that the boundary of set A is rigid and sharp and performs a two-class dichotomization (i.e., $x \in A$ or $x \notin A$). Besides, the universe of U is a crisp set.

A fuzzy set, on the other hand, introduces vagueness by eliminating the sharp boundary that divides members from nonmembers in the group. Thus, the transition between full membership and nonmembership is gradual rather than abrupt. Hence, fuzzy sets may be viewed as an extension and generalization of the basic crisp sets.

A fuzzy set $\tilde{A}$ in the universe of discourse U can be defined as a set of ordered pairs,

$$\tilde{A} = \{(x, u_{\tilde{A}}(x)) \mid x \in U\},$$

where $u_{\tilde{A}}(.)$ is called the "membership function" of $\tilde{A}$ and $u_{\tilde{A}}(x)$ is the degree of membership of x in $\tilde{A}$, which indicates the degree that x belongs to $\tilde{A}$. The membership function $u_{\tilde{A}}(.)$ maps U to the membership space M, that is, $u_{\tilde{A}} : U \rightarrow M$. For fuzzy sets, the range of the membership function(i.e., M) is a set on the unit interval [0, 1].

At layer two of the proposed fuzzy neural network, the net input to each neuron is first obtained by using the normalized weighted sum of the output from layer one. Then the activation function, which is the fuzzy membership function here, applies to the net output of each neuron at layer two. Thus, the output of each neuron at layer two represents the degree of darkness or brightness of the corresponding block. As discussed before (Figure 5.2), if the weighted sum of outputs from layer one approaches to 1, which means the block one is relatively dark, and the degree of darkness is high, but the degree of brightness is low for block one. On the contrary, if the weighted sum of outputs from layer one approaches to 0, the block one is relatively bright, and the degree of darkness is low, then the degree of brightness is high. Furthermore, from the analysis of the above relationship between the input and output of nets at layer two, it is noted that the relationship is non-linear. The neuron with different order, which reflects the position of the corresponding pixel, has a different effect on the final output. This is because the boundary pixels near to the block 2 and block 3 have a higher effect on the result than the pixels inside the block 1. The proposed fuzzy membership function is shown in figure 5.6.

The mathematical representation is:

$$f_d(net) = e^{-a*(net-1)^2}$$

and,

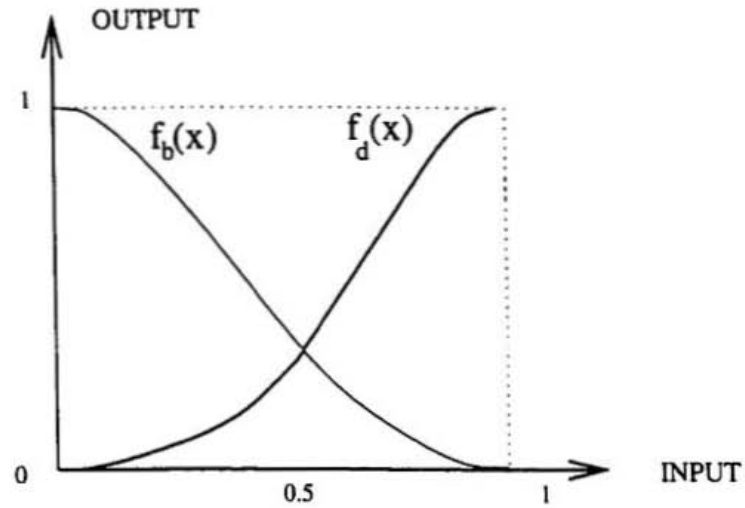


Figure 5.6: Fuzzy membership function

$f_b(net) = e^{-a*net^2}$

The output of node j at the second layer is:

$$O_{pj} = f(net_{pj})$$

Here, the function f is either f_d or f_b according to the pattern of the neuron at layer two, the function f_d is used to measure the relative darkness, and the function f_b is used to measure the relative brightness. The value of parameter a is chosen to be 16 based on experimental results.

The following equations correspond to block one:

$$O_{pj} = f_d(net_{pj}) \qquad for\ neuron\ A$$

$$O_{pj} = f_b(net_{pj}) \qquad for\ neuron\ B$$

A similar concept is applied to the other seven blocks.

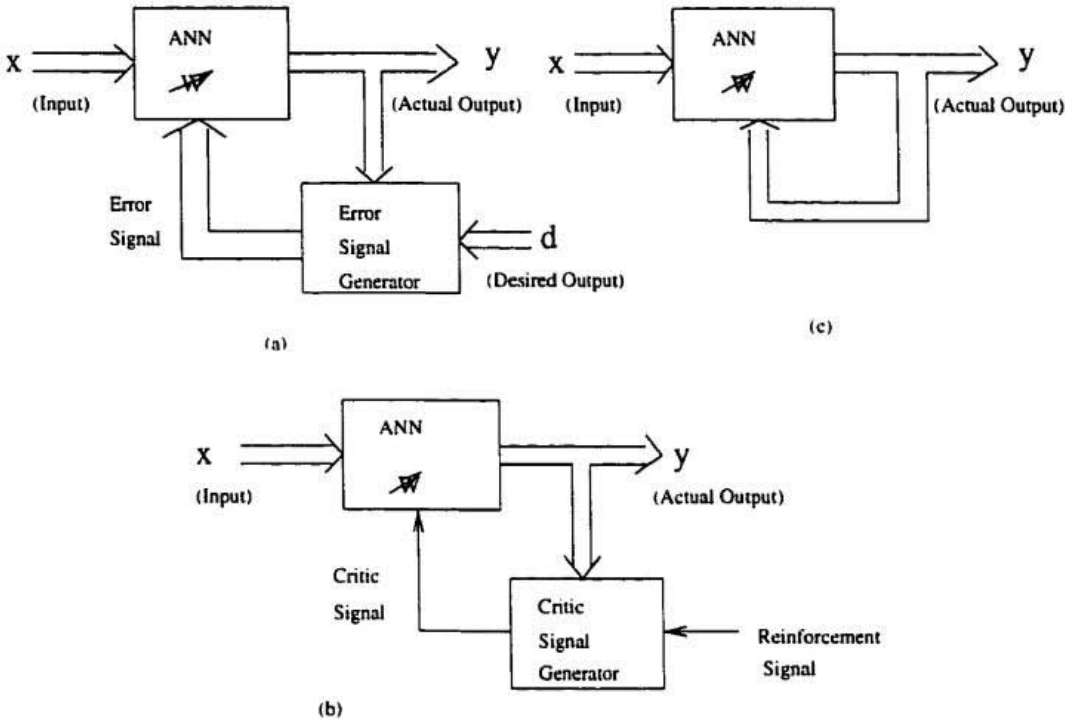


Figure 5.7: Three categories of learning

(a) Supervised learning. (b) Reinforcement learning. (c) Unsupervised learning.

5.4 Training Procedure with Learning Algorithm

A very important issue in specifying an artificial neural network(ANN) is the learning rules. In general, learning rules are classified into three categories: supervised learning, reinforcement learning, and unsupervised learning (see Figure 5.7). In this research, a supervised learning based on the BP (error back propagation) algorithm is applied on the training procedure of the proposed fuzzy neural network. Additionally, the general delta rule is used for weight updating.

5.4.1 Training data

In supervised learning mode (see 5.7(a)), the neural network is supplied with a sequence of training samples, $(\vec{x_1}, \vec{d_1})$, $(\vec{x_2}, \vec{d_2})$, ..., $(\vec{x_k}, \vec{d_k})$, ..., of desired input-output pairs. When each input $\vec{x_k}$ is put into the network, the corresponding desired output $\vec{d_k}$ is also supplied to the network. The difference between actual output $\vec{y_k}$ and the desired output $\vec{d_k}$ is measured in the error signal generator which produces error signals for the network to correct its weights. In such a way, the actual output will move closer to the desired output. The input of the training data of proposed network is a set of typical edge patterns. Each edge pattern has 36 pixels represented by their grey values. Therefore, the input vector has 36 elements and is written as:

$$A_p = \{A(0,0), A(0,1), ... A(5,5)\}$$

Here A_p corresponds to the input of p-th sample.

However, the input to the network has been changed after adaptive fuzzification and window division (called pre-processing) applied on it. Let $I_p = (I_{p1}, I_{p2}, ..., I_{pi})$ (i = 1, 2, ..., 60) represent the actual input of the network after pre-processing. According to figure 4.3 and figure 4.6, here

$$I_{p1} = f_a(A(0,0))$$

$$I_{p2} = I_{p43} = f_a(A(0,1))$$

. . . . . .

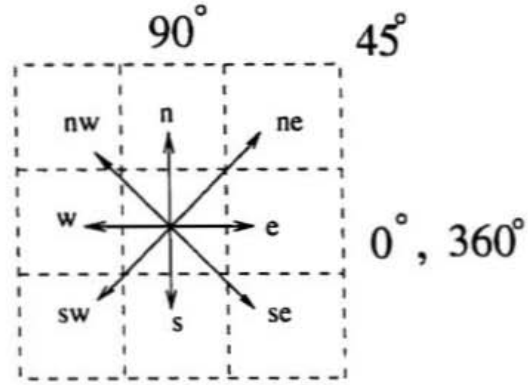


Figure 5.8: Eight principle orientations

$$I_{p60} = I_{p31} = f_a(A(4,3))$$

f_a is the adaptive fuzzy membership function shown in figure 3.7.

The desired output consists of ten elements corresponding to ten classes. Consider the center grid (or edge element) in a window, its direction can be coded into eight principle orientations (see Figure 5.8) and two non-orientations representing the dark window and the bright window.

Each edge element can be represented by a set of values as follows:

$$(t^e, t^w, t^s, t^n, t^d, t^b, t^{se}, t^{nw}, t^{sw}, t^{ne})$$

where the superscripts denote the orientations. For example, "e" denotes "*east*", "ne" denotes "*northeast*", "d" denotes "*dark*", and "b" denotes "bright".

Each element in the set has a value in [0,1] which represents the fuzzy measurement of the classification. If the given sample (assuming pth sample) has an output below, it means that the edge pattern with north orientation has 0.9 certainty, and

the edge pattern with northeast orientation has 0.6 certainty, etc..

$$t_p = (t_p^e, t_p^w, t_p^s, t_p^n, t_p^d, t_p^b, t_p^{se}, t_p^{nw}, t_p^{sw}, t_p^{ne}) = \{0.3, 0, 0, 0.9, 0, 0, 0, 0, 0, 0.6\}$$

In addition to representing training data in desired input-output pairs, it is required that training data be sufficient and proper. However, there is no procedure or rule suitable for all cases in choosing training data. One rule of thumb is that training data should cover the entire expected input space and then during the training process select training-vector pairs randomly from the set. In this research, forty typical edge patterns are first used as training samples. The recognition ability of this trained network is very good to detect edge images with little noise. Because only four samples are used for each class on average, these forty training samples are not enough to cover the most cases of the input-output space. Hence, the performance of the trained network is not satisfied when it is used to detect edges in noisy images, especially when noise in images is more than 30 percent. The input-output space is increased gradually by adding 10 training samples at a time. After 120 typical edge patterns are used, it is found that the performance of the trained network has no improvement.

5.4.2 Weight updating

One of the basic entities to specify for the chosen model of neural network is the learning rules for updating the connecting weights. The weight updating in this

research is based on the generalized delta rule formulated by Rumelhart, Hinton, and Williams [87]. In general, the actual outputs $\{o_{pk}\}$ will not be the same as the desired value $\{t_{pk}\}$. For each pattern, the square of the error is:

$$E_p = \frac{1}{2}\sum_k (t_{pk} - O_{pk})^2$$

The procedure for learning the correct set of weights is to vary the weights in a manner calculated to reduce the error E_p as rapidly as possible. The weight learning is carried out for the sequence of the patterns, one at a time. That is, the weights are changed immediately after a training pattern is presented. The convergence toward improved weights is achieved by taking incremental changes $\triangle w_{kj}$ proportional to $-\partial E_p/\partial w_{kj}$, that is:

$$\triangle w_{kj} = -\eta \frac{\partial E_p}{\partial w_{kj}}$$

Here, η is the learning rate. The learning rate is an important factor that affects the effectiveness and convergence of the BP learning algorithm. A large value of η could speed up the convergence but might result in overshooting, while a small value of η has a complementary effect. The best value of the learning constant at the beginning of the training may not be good at the end of training. Usually, values of η from 10^3 to 10 have been used successfully for many computational BP experiments.

Based on the structure analysis of the proposed fuzzy neural network, the

weights between layer two and layer three are unity. So the weight updating rule derived here is to update weights between layer one and layer two. Let $E_p^{(2)}$ be the error of neurons at layer two (the superscription represents the record layer), and the partial derivative be evaluated using the chain rule. That is:

$$\Delta w_{ji} = -\eta \frac{\partial E_p^{(2)}}{\partial w_{ji}} = -\eta \frac{\partial E_p^{(2)}}{\partial net_{pj}} \frac{\partial net_{pj}}{\partial w_{ji}} \tag{5.1}$$

As the net_{pj} is the input to the p-th neuron at the layer two, and it is the normalized weighted sum of all the outputs from the layer one(see 5.2.1), that is:

$$\frac{\partial net_{pj}}{\partial w_{ji}} = \frac{\partial}{\partial w_{ji}} \frac{\sum_{i=1}^{n} w_{ji} I_{pi}}{\sum_{i=1}^{n} w_{ji}} \tag{5.2}$$

So

$$\frac{\partial net_{pj}}{\partial w_{ji}} = \frac{-(\sum_{l=1, l \neq i}^{n} w_{jl} + 1)}{(\sum_{i=1}^{n} w_{ji})^2} I_{pi} \tag{5.3}$$

Applying chain rule again to the partial derivative $\partial E_p^{(2)} / \partial net_{pj}$ yields:

$$\frac{\partial E_p^{(2)}}{\partial net_{pj}} = \frac{\partial E_p^{(2)}}{\partial O_{pj}} \frac{\partial O_{pj}}{\partial net_{pj}} \tag{5.4}$$

Because the output O_{pj} is the output of the neuron j after fuzzy membership function f (f_d or f_b) is applied on the neuron input, therefore,

$$\frac{\partial O_{pj}}{\partial net_{pj}} = \frac{\partial f(net_{pj})}{\partial net_{pj}} = f'(net_{pj}) \tag{5.5}$$

where $f(net)$ represents the membership function f_d, $f_d(net) = e^{-a*(net-1)^2}$, (see section 5.3)

$$f'_d(net_{pj}) = -2a * (net_{pj} - 1) * e^{-a*(net_{pj}-1)^2} = -2a * (net_{pj} - 1)O_{pj} \qquad (5.6)$$

That is:

$$f'_d(net_{pj}) = -2a * (\frac{\sum_{i=1}^{n} w_{ji} I_{pi}}{\sum_{i=1}^{n} w_{ji}} - 1)O_{pj} = -2a * \frac{\sum_{i=1}^{n} w_{ji}(I_{pi} - 1)}{\sum_{i=1}^{n} w_{ji}} O_{pj} \qquad (5.7)$$

For those neurons which measure the darkness of the corresponding blocks, the $f'(net_{pj})$ uses $f'_d(net_{pj})$.

$$\Delta w_{ji} = -\eta \frac{\partial E_p^{(2)}}{\partial O_{pj}} \frac{-(\sum_{l=1, l\neq i}^{n} w_{jl} + 1)}{(\sum_{i=1}^{n} w_{ji})^2} I_{pi}(-2a * \frac{\sum_{i=1}^{n} w_{ji}(I_{pi} - 1)}{\sum_{i=1}^{n} w_{ji}} O_{pj}) \qquad (5.8)$$

$$\Delta w_{ji} = -2a\eta \frac{\partial E_p^{(2)}}{\partial O_{pj}} \frac{(\sum_{l=1, l\neq i}^{n} w_{jl} + 1)}{(\sum_{i=1}^{n} w_{ji})^3} (\sum_{i=1}^{n} w_{ji}(I_{pi} - 1)) I_{pi} O_{pj} \qquad (5.9)$$

When $f(net)$ represents the membership function f_b, $f_b(net) = e^{-a*net^2}$, (see section 5.3)

$$f_b'(net_{pj}) = -2a * net_{pj} * e^{-a*net_{pj}^2} = -2a * net_{pj} O_{pj} \tag{5.10}$$

That is:

$$f_b'(net_{pj}) = -2a * \frac{\sum_{i=1}^{n} w_{ji} I_{pi}}{\sum_{i=1}^{n} w_{ji}} O_{pj} \tag{5.11}$$

Hence, the equation (1) can be written as:

$$\triangle w_{ji} = -\eta \frac{\partial E_p^{(2)}}{\partial O_{pj}} \frac{\partial O_{pj}}{\partial net_{pj}} \frac{\partial net_{pj}}{\partial w_{ji}} \tag{5.12}$$

$$\triangle w_{ji} = -\eta \frac{\partial E_p^{(2)}}{\partial O_{pj}} \frac{-(\sum_{l=1, l\neq i}^{n} w_{jl} + 1)}{(\sum_{i=1}^{n} w_{ji})^2} I_{pi} * f'(net_{pj}) \tag{5.13}$$

For those neurons which measure the brightness of the corresponding blocks, the $f'(net_{pj})$ uses $f_b'(net_{pj})$.

$$\triangle w_{ji} = -\eta \frac{\partial E_p^{(2)}}{\partial O_{pj}} \frac{-(\sum_{l=1, l\neq i}^{n} w_{jl} + 1)}{(\sum_{i=1}^{n} w_{ji})^2} I_{pi} (-2a * \frac{\sum_{i=1}^{n} w_{ji} I_{pi} - 1}{\sum_{i=1}^{n} w_{ji}} O_{pj}) \tag{5.14}$$

$$\triangle w_{ji} = -2a\eta \frac{\partial E_p^{(2)}}{\partial O_{pj}} \frac{(\sum_{l=1, l\neq i}^{n} w_{jl} + 1)}{(\sum_{i=1}^{n} w_{ji})^3} (\sum_{i=1}^{n} w_{ji} I_{pi}) I_{pi} O_{pj} \tag{5.15}$$

Therefore the rule for weight updating is:

$$w_{ji}(t+1) = w_{ji}(t) + \Delta w_{ji}(t) \tag{5.16}$$

If neuron j measures the darkness, $\Delta w_{ji}(t)$ will be calculated using equation 5.13. If neuron j measures the brightness, $\Delta w_{ji}(t)$ will be calculated using equation 5.15.

5.4.3 Training procedure

The designed training procedure is based on the Back-Propagation algorithm. For a given input-output pair, the BP algorithm performs two phases of data flow. First, the input pattern is propagated from the input layer to the output layer and, as a result of this forward flow of data, it produces an actual output. Then the error signals resulting from the difference between the desired output and actual output are back-propagated from the output layer to the previous layers for them to update their weights. The training of proposed fuzzy neural network can be outlined in the following steps.

Input:

According to the discussion in 5.4.1, a set of training pairs are represented as $\{(A_p, t_p) \mid p = 1, 2, ..., n\}$,

$A_p = \{A(0,0), A(0,1), ..., A(5,5)\}$ and $t_p = (t_{p1}, t_{p2}, ..., t_{p10})$

Here A_p corresponds to the input of the p-th pattern, and t_p is the target output

after classification.

Step 1: *Initialization*

Choose a learning rate $\eta > 0$, a maximum acceptable system error, and an individual acceptable error of each training vector. Initialize the weights to random values in (0,1) and set a maximum acceptable iteration value.

Step 2: *Pre-processing*

This procedure accomplishes the adaptive fuzzification and window division. Based on the section 5.4.1, $I_p = (I_{p1}, I_{p2}, ..., I_{pi})$ (i = 1, 2, ..., 60) fuzzy values are obtained as the actual input of the network.

Step 3: *Calculation from layer one to layer two:*

This is a feedforward calculation procedure. First, each weight is normalized, then the net input is computed as a weighted sum of outputs from the layer one. That is:

$$\tilde{w}_{ji} = \frac{w_{ji}}{\sum_{i=1}^{n} w_{ji}} \quad and, \quad net_{pj} = \sum_{i=1}^{n} \tilde{w}_{ji} I_{pi} \tag{5.17}$$

The activation functions used here are the two different fuzzy membership functions(f_d or f_b) depending on whether to measure darkness or brightness.

Hence, the output of node j at the second layer is:

$$O_{pj} = f(net_{pj}) \tag{5.18}$$

Step 4: *Calculation from layer two to layer three*

This is a feedforward calculation procedure too. Based on the discussion at section 5.2.2, the net input to a neuron at layer three is the product of the outputs of neurons at layer two which have connection with the neuron (reference to the figure 5.1).

That is:

$$Net_{p1}^{(3)} = O_{p1}^{(2)} * O_{p3}^{(2)} * O_{p6}^{(2)} * O_{p8}^{(2)} \tag{5.19}$$

$$Net_{p2}^{(3)} = O_{p2}^{(2)} * O_{p4}^{(2)} * O_{p5}^{(2)} * O_{p7}^{(2)} \tag{5.20}$$

and,

$$Net_{p10}^{(3)} = O_{p10}^{(2)} * O_{p12}^{(2)} * O_{p13}^{(2)} * O_{p15}^{(2)} \tag{5.21}$$

The activation function for the neurons at layer 3 is:

$$O_{pk} = (net_{pk})^{1/4} \qquad (k = 1, 2, ..., 10) \tag{5.22}$$

Step 5: *Output error measurement*

In general, the outputs O_{pk} will not be the same as the target value t_{pk}. For

each pattern, the square of the error is:

$$E_p = \frac{1}{2}\sum_k (t_{pk} - O_{pk})^2 \tag{5.23}$$

and the average system error is:

$$E = \frac{1}{2P}\sum_p \sum_k (t_{pk} - O_{pk})^2 \tag{5.24}$$

From the structure of the proposed network, it is noted that each unit at layer 3 connects with four neurons at layer 2. The error of each neuron at layer 3 comes from these four neurons. Besides, each neuron at layer 2 only has one connection with layer three. To simplify the error back-propagation from layer 3, the error of each node at layer 2 is approximately set as:

$$E_p^{(2)} = \frac{1}{4}E_p \tag{5.25}$$

Applying chain rule to the partial derivative $\partial E_p^{(2)}/\partial O_{pj}$ yields:

$$\frac{\partial E_p^{(2)}}{\partial O_{pj}} = \frac{1}{4}\frac{\partial E_p}{\partial O_{pj}} == \frac{1}{4}\frac{\partial E_p}{\partial O_{pk}}\frac{\partial O_{pk}}{\partial O_{pj}} \tag{5.26}$$

$\partial E_p/\partial O_{pk}$ can be written as:

$$\frac{\partial E_p}{\partial O_{pk}} = \frac{\partial(1/2\sum_k (t_{pk} - O_{pk})^2)}{\partial O_{pk}} = t_{pk} - O_{pk} \tag{5.27}$$

The $\partial O_{pk}/\partial O_{pj}$ can be written as:

$$\frac{\partial O_{pk}}{\partial O_{pj}} = \frac{\partial (net_{pk})^{1/4}}{\partial O_{pj}} = \frac{\partial (C_k O_{pj})^{1/4}}{\partial O_{pj}} = \frac{1}{4} C_k^{1/4} O_{pj}^{-3/4} \tag{5.28}$$

Here, C_k is the product of the outputs of the corresponding three neurons which are along with the jth neuron at layer two to be connected to the kth neuron at layer three. For example, when $k = 1$, for $net_{p1}^{(3)}$ (see fomula 5.19), $C_k = O_{p3}^{(2)} * O_{p6}^{(2)} * O_{p8}^{(2)}$.

Therefore, $\partial E_p^{(2)}/\partial O_{pj}$ can be finally written as:

$$\frac{\partial E_p^{(2)}}{\partial O_{pj}} = \frac{1}{4}(t_{pk} - O_{pk})(\frac{1}{4} C_k^{1/4} O_{pj}^{-3/4}) = \frac{1}{16} C^{1/4} O_{pj}^{-3/4}(t_{pk} - O_{pk}) \tag{5.29}$$

Step 6: *Weight updating*

This step is to propagate the errors backward to update the weights. Based on the discussion at step 5 (see equation(5.29)), the $\triangle w_{ji}(t)$, which is used to update weights between layer one and layer two, can be written as:

$$\triangle w_{ji}(t) = -\frac{1}{8} a\eta C_k^{1/4} O_{pj}^{1/4} I_{pi}(t_{pk} - O_{pk}) * \frac{(\sum_{l=1, l\neq i}^{n} w_{jl} + 1)}{(\sum_{i=1}^{n} w_{ji})^3} \sum_{i=1}^{n} w_{ji}(I_{pi} - 1)$$

if neuron j measures the darkness.

$$\triangle w_{ji}(t) = -\frac{1}{8} a\eta C_k^{1/4} O_{pj}^{1/4} I_{pi} (t_{pk} - O_{pk}) * \frac{(\sum_{l=1, l\neq i}^{n} w_{jl} + 1)}{(\sum_{i=1}^{n} w_{ji})^3} \sum_{i=1}^{n} w_{ji} I_{pi}$$

if neuron j measures the brightness.

Step 7: Training termination check

After the whole set of training data has been presented once, if the error E_p is acceptable small for each of the training vector pair (input pattern - expected result), or the average system error is acceptably small, or the acceptable iteration value is reached, the training will be terminated. Otherwise, initiate new training by going to step 3.

5.5 Edge Detection by Trained FNN

After the FNN is trained, it can be used for edge detection through the following steps.

Step 1: Input an image which is represented by an $M \times N$ matrix.

$$A = \{A(0,0), A(0,1), ..., A(m-1, n-1)\}$$

A size 6×6 window is then applied on the image as discussed in 5.1.1, therefore the number of total patterns for an $m \times n$ image is $l = (m-5) \times (n-5)$.

Step 2: (Detection): Apply the adaptive fuzzification to the p-th pattern to

obtain the net input $I_p = (I_{p1}, I_{p2}, ..., I_{p60})$.

Step 3: Calculate the output of the p-th pattern at layer 2 (the same procedure as the step 3 in 5.4.3).

Step 4: Calculate the output of the p-th pattern at layer 3 (the same procedure as the step 3 in 5.4.4). Ten output values corresponding to ten classes are obtained.

Step 5: Store the above ten output values to corresponding map files respectively. For instance, output O_{p1} will be stored to map file 1 which corresponds to class 1.

Step 6. Check if the whole set of patterns has been processed. If $p < l$, then, $p = p+1$ and go to step 2. Otherwise, the detection is terminated and ten map files corresponding to ten edge maps are obtained. Two maps contain the information for non-edge elements. Eight maps stored the edge credits for each orientation used for edge enhancement.

For an $n \times n$ size image, when applying the pre-processing procedure, Adaptive Fuzzification and Window Division, on the image to get the inputs for FNN, there are $(n-5) \times (n-5)$ patterns generated. For each pattern, we need to re-order the sequence of pixels. The computation complexity is constant since the window size is fixed as 6×6. Therefore, the computational complexity to finish the pre-

processing procedure is $O(n^2)$. When using a trained FNN to detect edges in an $n \times n$ images, to obtain each pixel on one of the output maps, it takes constant time to do the calculation. Hence, the computation complexity to generate ten output maps is $O(n^2)$.

Chapter 6

Edge Enhancement By Hopfield Neural Network

Hopfield networks are widely employed in the area of solving optimization problems. The key problem is to formulate an optimization problem with a proper energy function that can be used to construct a Hopfield network. After obtaining edge maps by using the fuzzy neural network, there are some noise and edges missed on the maps. In this chapter, a modified Hopfield neural network is constructed based on constraint satisfaction and competitive mechanism. The purpose is to enhance edges in edge maps, remove noise and recover missing edges. Several factors should be considered, such as what kind of information contributes to input edge maps, what type of edge structures are stable, and how to update the edge measurement to approach the final result.

6.1 Generate Input Edge Maps

At the edge enhancement stage, first of all, for the whole edge map, there is no way to design an energy function which covers all directions in the map and can also be minimized. Hence, eight Hopfield networks are designed to enhance edges in the eight orientations, east, west, north, south, northeast, northwest, southeast and southwest, respectively. After the first stage, edge detection, there are ten credit maps obtained from the fuzzy neural network. Eight edge credit maps contain the edge measurement corresponding to eight different orientations, and two credit maps contain the non-edge measurement. Therefore, the first stage provides the information which can be used to generate the input edge map to the Hopfield neural network. Suppose a $k \times k$ window is adopted to build the Hopfield network (in this research, window size is 7×7, reason given in 6.2), after considering edges with all different orientations together, there are five possible stable structures of a window:

1. an edge through the window,
2. a corner formed by two edge lines,
3. an intersection of edges,
4. no edge in the window, and
5. a combination of the first three cases.

Because it is difficult or impossible to find a configuration of a Hopfield network

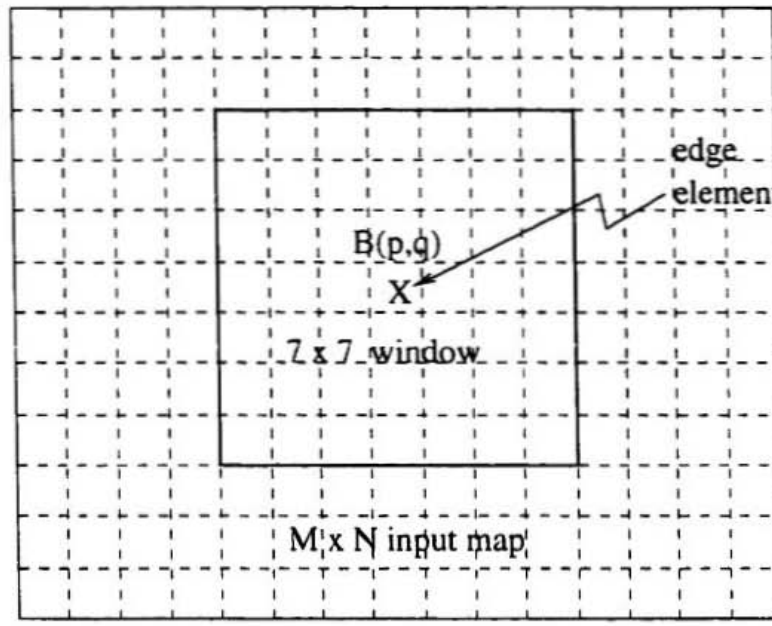


Figure 6.1: Matrix representation of input edge map

for minimizing an energy function to cover all the possible stable states if all different orientations are considered together, eight input edge maps need to be constructed to correspond to the eight orientations. Let an input edge map (eg. the north input edge map) represented by an $M \times N$ matrix B (see Figure 6.1).

To generate the input edge map with one orientation, several credit maps obtained from FNN will be used. For example, to build the input edge map with north orientation, not only the edge credit in north orientation needs to be considered, but also the edge credits of the adjacent orientations need to be considered (see the cases in Figure 6.2(c),(d)). Besides, if there is very small edge credits in the above three orientations, the non-edge credits have to be considered because there exists the case of no edges in some part of the input map. The edge pattern

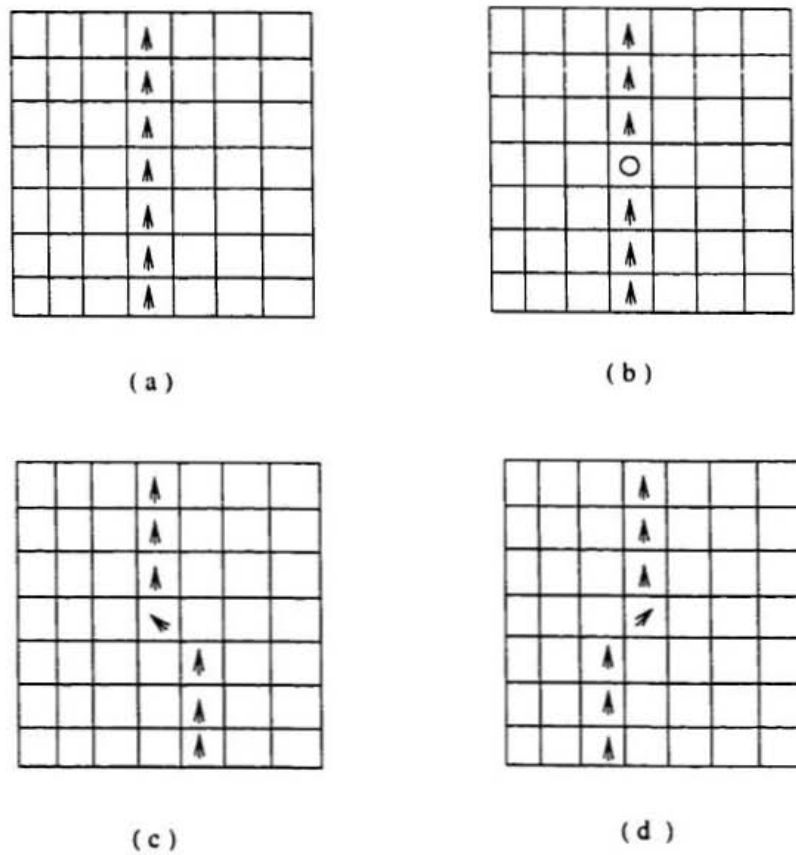
in a window can be stable as the figure 6.2 shows. Therefore, three maps which correspond to north, northeast, and northwest orientations, and two maps which consist of non-edge credits are used to build the input edge map to the Hopfield network in the north orientation.

In general, the procedure to generate an input map follows:

Step 1. If the largest value of output from the FNN is the node corresponding to the orientation under consideration, then the edge credit value is taken into the input edge map (Fig. 6.2(a)).

Step 2. If the largest value of output from FNN is the node corresponding to the adjacent orientation under consideration and there are not more than two elements adjacent to this node in the same orientation, then the edge credit value is taken into the input edge map (Fig. 6.2(c),(d)).

Step 3. If the largest value neither comes from the orientation under consideration, nor the adjacent orientations, but there is an edge credit value for the orientation under consideration, then take it into the input edge map. Otherwise, the largest value of output from non-edge maps is negated and taken into the map (Fig. 6.2(b)).



Note: The arrow points out the orientation of edge element. the circle means the credit of edge element comes from non-edge output maps.

Figure 6.2: Relative edge structures in north orientation

6.2 Stable Structure of a Window

As the entire stable structure is not predictable, depending on the current image, to design such a neural network with an energy function for a whole image is impossible or impractical. However, it is noted that the influence is very small between two distant elements. Hence, when a 7×7 window is applied on the image, the elements under consideration are only those which are inside the window. The correlation between the central element and the element outside the window can be ignored without affecting the final result because the distance between those pixels is more than three pixels so that the influence between them is very little. Analyzing the possible stable edge structure in the north orientation, the window is stable if it is in one of the cases shown in figure 6.3.

A similar analysis can be used when the other orientations are considered. Therefore, it can be concluded that the window is stable if it falls into one of the following three cases.

1. an edge with the orientation under consideration through the window,
2. an edge with the orientation has an endpoint in the window, and
3. no edge in the window.

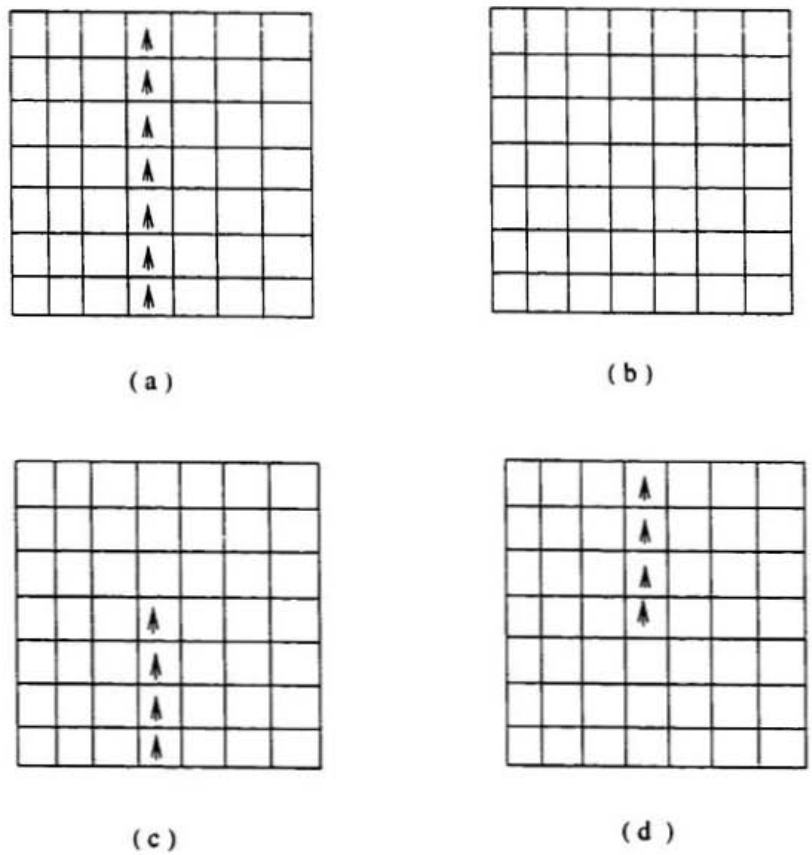


Figure 6.3: Stable structures in north orientation

6.3 The Energy Function

Typically, when using a Hopfield network to solve optimization problems, the energy function of the network is made equivalent to a certain cost function that needs to be minimized. The search for a minimum energy performed by the Hopfield network corresponds to the search for a solution to the optimization problem. The edge enhancement here is to obtain the complete boundary of objects in 2-D image with noise removed and missing edge filled up. Hence, the key work presented in this section is to formulate a proper energy function that can be used to construct a Hopfield network. The creation of the energy function is based on a constraint-competitive mechanism. According to the stable structure of the window discussed above, the energy function for the north orientation should be constructed which favors states that:

1. have each row only one +1 and other all -1, or, have all -1 in all rows;

and,

2. have all -1 or +1 in a column;

Here, value "-1" is an edge credit for the element which is definitely non-edge, and value "+1" is an edge credit for the element which is definitely an edge.

The energy function designed to satisfy the above states for the north orientation is:

$$E = A\sum_{x=1}^{7}\sum_{i=1}^{7}\sum_{j=1,j\neq i}^{7}(V_{x,i}+1)(V_{x,j}+1)$$

$$+B\sum_{x=1}^{7}[\sum_{i=1}^{7}(V_{x,i}+1)-2]^2\sum_{j=1}^{7}(V_{x,j}+1)$$

$$+C\sum_{i=1}^{7}(\sum_{x=1}^{6}V_{x,i}V_{x+1,i}-6)^2$$

$$+D\sum_{i=1}^{7}(\sum_{x=1}^{5}V_{x,i}V_{x+2,i}-5)^2$$

$$+E\sum_{i=1}^{7}(\sum_{x=1}^{4}V_{x,i}V_{x+3,i}-4)^2 \qquad (6.1)$$

Where, V represents the output of the neural cell with values ranging from -1 to +1, and A, B, C, D, E are parameters.

The first two items (with the parameters A and B) are designed to satisfy the first state. When having all -1 in all rows, $(V_{x,i}+1)$ and $(V_{x,j}+1)$ are zero. So the first two items have values zero. When having each row only one +1 and other all -1, since $i \neq j$, either $(V_{x,i}+1)$ or $(V_{x,j}+1)$ will be zero, which yields the first item is zero. For the second item, as only one +1occurs at each row, $\sum_{i=1}^{7}(V_{x,i}+1)$ will be 2, which makes the first term $[\sum_{i=1}^{7}(V_{x,i}+1)-2]^2$ be zero. Therefore, the first two items reach their minimum zero when the edge structure in a window is stable at the state 1. The last three items are designed to satisfy the second state. When having all -1 or +1 in a column, $\sum_{x=1}^{6}V_{x,i}V_{x+1,i}$ will be 6, and the $(\sum_{x=1}^{6}V_{x,i}V_{x+1,i}-6)^2$ will be zero, which makes the whole item with parameter C as zero. The same result can be obtained for the last two items. The above analysis shows that the proposed energy function is able to reach its minimum to meet the requirement of the stable edge structures in a window as discussed above.

Actually, equation 6.1 has defined the system evolution law. The final result of

the edge enhancement is obtained when certain stable edge structures are reached after a certain number of iterations by neuron computation. The neural network structure has to be constructed to perform the updating of edge measurement to approach the final result. The following will show that evolution performs a downhill movement along the surface of the energy function, and eventually reaches a steady point. From the energy function, the time derivative of the central unit in a window can be derived as:

$$\begin{aligned}
&\frac{dy_{4,4}}{dt} = -\frac{y_{4,4}}{\tau} - A\sum_{i=1,i\neq 4}^{7}(V_{4,i}+1) - B[\sum_{i=1}^{7}(V_{4,i}+1)-2]^2 \\
&-2B\sum_{i=1}^{7}(V_{4,i}+1)(V_{4,3}+V_{4,5}+2)(\sum_{j=1}^{7}(V_{4,j}+1)-2) \\
&-2C(V_{3,4}+V_{5,4})\sum_{x=1}^{6}(V_{x,4}V_{x+1,4}-1) \\
&-2D(V_{2,4}+V_{6,4})\sum_{x=1}^{5}(V_{x,4}V_{x+2,4}-1) \\
&-2E(V_{1,4}+V_{7,4})\sum_{x=1}^{4}(V_{x,4}V_{x+3,4}-1)
\end{aligned} \tag{6.2}$$

The terms with coefficient A and B are for interactions between the central unit and the other units on the row. Analyzing the terms with A and B, they are never less than zero. If the parameters A and B are set with positive values, the whole part (including the sign before A and B) will never be positive. That implies the other units at the central row send signals to suppress the central unit. The last three terms determine the interactions between the central unit and the other units on the column. It is known that no summation in these terms will be greater than 0. The first part determines whether the interaction should be

inhibited or excited since the summation part is always less than or equal to zero. Also, it is observed that the closer the unit is to the central unit, the bigger the effect it has on the central unit. Therefore, the parameters C, D and E should be set so that the term with $V_{3,4} + V_{5,4}$ plays a more important role for the lateral interaction than the terms with $V_{2,4} + V_{6,4}$ and $V_{1,4} + V_{7,4}$. Hence, C is greater than D and D is greater than E. All parameters in the above equation are positive. When the system does not reach a stable state, the first parts of the last three terms determine whether to inhibit or excite the central unit, and how strong the interaction is. When it is close to a stable state, the second parts approach to 0 so that the exciting or inhibiting is very small. Furthermore, it can be concluded that the terms with coefficient A and B play a horizontal suppression role and the terms with coefficient C, D, and E play either a vertical suppression or a vertical reinforcement role to the edge element or central unit. If suppression is strong, it is good for eliminating false edge elements caused by noise. On the other hand, reinforcement contributes to edge element enhancement and missing edge element recovery.

According to the energy function, the following stable edge structures correspond to the minimum energy.

$$\begin{bmatrix} -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 \end{bmatrix} \qquad \begin{bmatrix} -1 & -1 & -1 & 1 & -1 & -1 & -1 \\ -1 & -1 & -1 & 1 & -1 & -1 & -1 \\ -1 & -1 & -1 & 1 & -1 & -1 & -1 \\ -1 & -1 & -1 & 1 & -1 & -1 & -1 \\ -1 & -1 & -1 & 1 & -1 & -1 & -1 \\ -1 & -1 & -1 & 1 & -1 & -1 & -1 \\ -1 & -1 & -1 & 1 & -1 & -1 & -1 \end{bmatrix}$$

(1) (2)

The analysis for the south orientation is exactly the same as that to the north orientation above. A similar analysis can also be applied to the horizontal orientation to derive energy function at the east or west orientation as:

$$\begin{aligned} E = & A\sum_{x=1}^{7}\sum_{i=1}^{7}\sum_{j=1,j\neq i}^{7}(V_{i,x}+1)(V_{j,x}+1) \\ & +B\sum_{x=1}^{7}[\sum_{i=1}^{7}(V_{i,x}+1)-2]^2\sum_{j=1}^{7}(V_{j,x}+1) \\ & +C\sum_{i=1}^{7}(\sum_{x=1}^{6}V_{i,x}V_{i,x+1}-6)^2 \\ & +D\sum_{i=1}^{7}(\sum_{x=1}^{5}V_{i,x}V_{i,x+2}-5)^2 \\ & +E\sum_{i=1}^{7}(\sum_{x=1}^{4}V_{i,x}V_{i,x+3}-4)^2 \end{aligned} \tag{6.3}$$

And the differential equation, which yields the update rule at horizontal orientation, is:

$$\frac{dy_{4,4}}{dt} = -\frac{y_{4,4}}{\tau} - A\sum_{i=1,i\neq 4}^{7}(V_{i,4}+1) - B[\sum_{i=1}^{7}(V_{i,4}+1)-2]^2$$

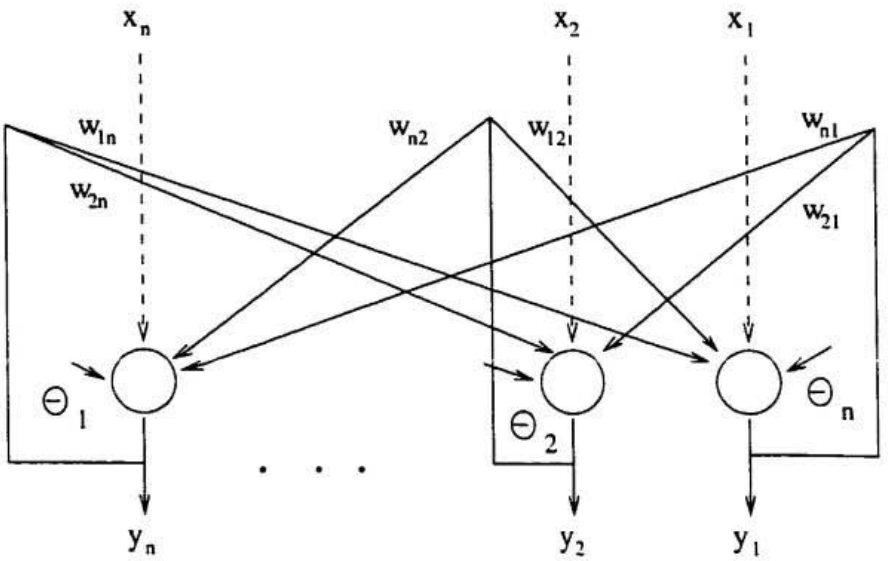


Figure 6.4: Structure of the Hopfield network

$$
\begin{aligned}
&-2B\textstyle\sum_{i=1}^{7}(V_{i,4}+1)(V_{3,4}+V_{5,4}+2)(\sum_{j=1}^{7}(V_{j,4}+1)-2)\\
&-2C(V_{4,3}+V_{4,5})\textstyle\sum_{x=1}^{6}(V_{4,x}V_{4,x+1}-1)\\
&-2D(V_{4,2}+V_{4,6})\textstyle\sum_{x=1}^{5}(V_{4,x}V_{4,x+2}-1)\\
&-2E(V_{4,1}+V_{4,7})\textstyle\sum_{x=1}^{4}(V_{4,x}V_{4,x+3}-1)
\end{aligned}
\tag{6.4}
$$

6.4 Structure of Modified Hopfield Network

In general, a Hopfield network consisting of N units is depicted in figure 6.4. It is a fully connected network. Each node receives input from all other units. There is no distinction between input units, hidden units and output units. Also, there is no self-feedback in a Hopfield network.

The figure shows that the Hopfield network is a single-layer recurrent network which performs a sequential updating process. Each node has an external input

x_j and a threshold θ_j, where $j = 1, 2, ..., n$. An input pattern is first applied to the network, and the network's output is initialized accordingly. Then, the initial input pattern is removed and the output becomes the new, updated input through the feedback connections. The first updated input forces the first updated output; then in turn acts as the second updated input through the feedback links and produces the second updated output. The transition process continues until no new updated responses are produced and the network has reached its equilibrium.

Analyzing the energy function (equation 6.1) for the north orientation, the following edge structures do not correspond to the minimum of the energy function.

1. end-point edge in the window:

$$\begin{bmatrix} -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & 1 & -1 & -1 & -1 \\ -1 & -1 & -1 & 1 & -1 & -1 & -1 \\ -1 & -1 & -1 & 1 & -1 & -1 & -1 \\ -1 & -1 & -1 & 1 & -1 & -1 & -1 \end{bmatrix}$$

2. non-vertical edge through the window:

$$\begin{bmatrix} -1 & -1 & 1 & -1 & -1 & -1 & -1 \\ -1 & -1 & 1 & -1 & -1 & -1 & -1 \\ -1 & -1 & 1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & 1 & -1 & -1 & -1 \\ -1 & -1 & -1 & 1 & -1 & -1 & -1 \\ -1 & -1 & -1 & 1 & -1 & -1 & -1 \\ -1 & -1 & -1 & 1 & -1 & -1 & -1 \end{bmatrix}$$

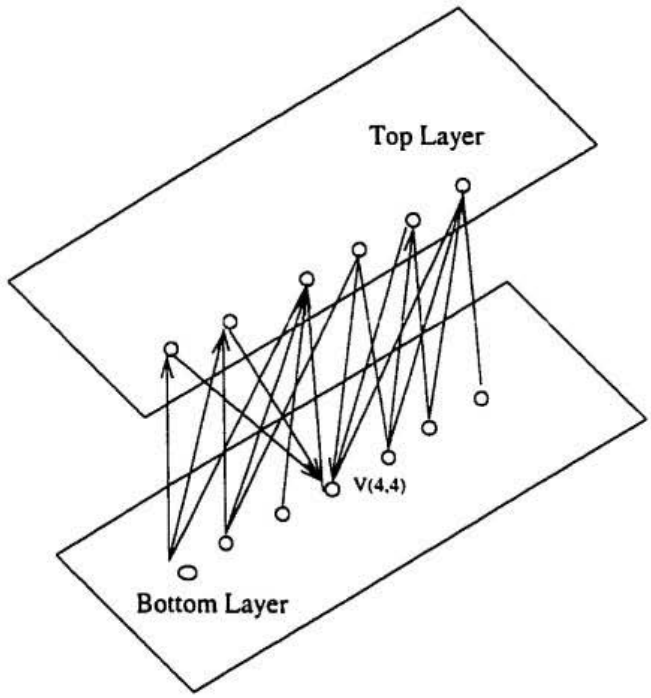
It is very difficult to find an energy function to cover these unstable states. Moreover, the differential equation 6.2 has terms containing the product of several

outputs from the neural units. It is also very difficult to generate such products by neurons. Hence, the neural network structure in the thesis is designed based on both the energy function and the competitive mechanism. Actually, the result is based on local updates to approach the global minimization of the energy function. To clearly represent the structure of the proposed network, there is an additional layer (called the top layer) which is required to get input from the first layer (called the bottom layer) and to give feedback to the first layer through the weights which are derived from the equation 6.2. The modified Hopfield network for edge enhancement at a north orientation is illustrated in Figure 6.5.

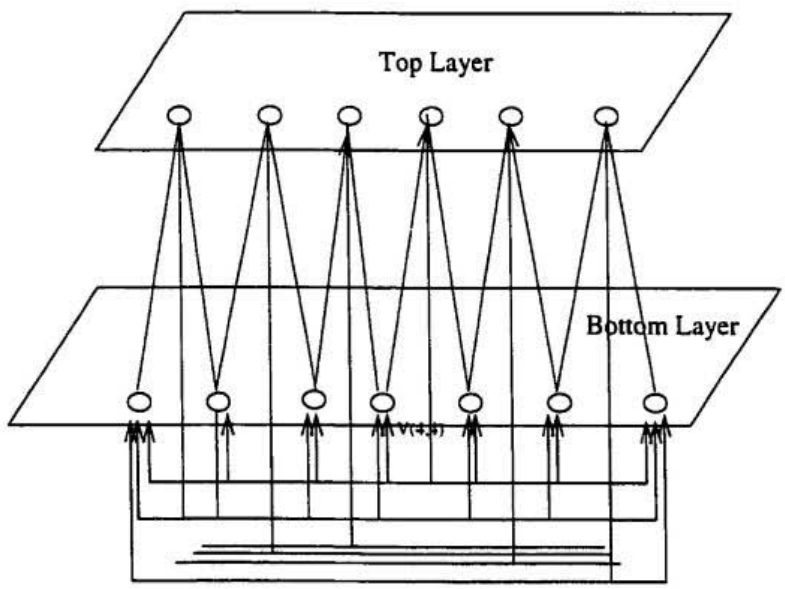
Only the interconnections to the central unit of a 7×7 window ($V_{4,4}$) are shown in figure 6.5. The excitatory or inhibitory signals are not directly from the neighboring units. Instead, they are from the top layer where the structural information is incorporated by the interconnections of the two layers. The weights (coefficients A, B,C ...) are selected such that the neural structure reaches stable states for case 1 (edge through window) and case 3 (no edge in the window), or make a little change of the credit values for case 2 (end-point in the window).

6.5 Edge Enhancement Procedure

Based on the discussion in the above sections, a Hopfield neural network is built to enhance edges in each of the eight orientations. Hence, the algorithm has the advantage of being able to distinguish the different types of noise and therefore



(a) The weight connections for neurons at the same column



(b) The weight connections for neurons at the same row

Figure 6.5: Modified Hopfield network for edge enhancement

can eliminate them separately. Again, taking the north orientation as an example, the edge enhancement procedure includes the following steps.

Step1. Generate input patterns

The input pattern generation is based on the input map in north orientation created in section 6.1. A 7×7 window is applied to the input map to get a sequence of input patterns with forty-nine elements each.

Step 2. Initialization

It is important to choose proper parameters A, B, C, D and E. In this research, these parameters are trade-off determined by the current input image map. The general rules are: 1) $C > D > E$, 2) A and B are less than E, 3) all parameters are positive and less than 1. Besides, the parameter Δt, τ, as well as the gain parameter of the sigmoidal function need to be set. Also the maximum acceptable iteration value ($M_{iteration}$) and the maximum acceptable unstable element number ($M_{unstable}$) should be decided. The initial output of neural cell $V(i,j)$ is set to its input value.

Step 3. Update the edge element in each window (Enhancement Loop)

For each window, calculate the new output of its edge element ($V_{4,4}$) by using the update rule represented in equation 6.2 for vertical orientation edge enhance-

ment. For the horizontal orientation edge enhancement, the update rule follows equation 6.4. Depending on the way that the window applied on input edge maps on the diagonal orientations, the equation 6.2 is also used as the update rule for the northwest and southeast orientations, and the equation 6.4 for the northeast and southwest orientations. Again taking the north orientation as an example, the update procedure is as follows.

Let V^t be the output of edge element at time t, V^{t+1} be the output of edge element at time $t+\Delta t$, $y_{4,4}$ be the center element before updating (at time t), and $\Delta y_{4,4}$ be the change of center element from time t to time $t+\Delta t$. Then,

$$\begin{aligned}
\Delta y_{4,4} = [&-\tfrac{y_{4,4}}{\tau} - A\textstyle\sum_{i=1,i\neq 4}^{7}(V_{4,i}^t + 1) - B[\sum_{i=1}^{7}(V_{4,i}^t + 1) - 2]^2 \\
&-2B\textstyle\sum_{i=1}^{7}(V_{4,i}^t + 1)(V_{4,3}^t + V_{4,5}^t + 2)(\sum_{j=1}^{7}(V_{4,j}^t + 1) - 2) \\
&-2C(V_{3,4}^t + V_{5,4}^t)\textstyle\sum_{x=1}^{6}(V_{x,4}^t V_{x+1,4}^t - 1) \\
&-2D(V_{2,4}^t + V_{6,4}^t)\textstyle\sum_{x=1}^{5}(V_{x,4}^t V_{x+2,4}^t - 1) \\
&-2E(V_{1,4}^t + V_{7,4}^t)\textstyle\sum_{x=1}^{4}(V_{x,4}^t V_{x+3,4}^t - 1)]\Delta t
\end{aligned}$$

Thus, the new output of the center element after updating is:

$$V_{4,4}^{t+1} = f(y_{4,4} + \Delta y_{4,4})$$

The sigmoidal function is used as the activation function $f(.)$.

Step 4. Calculate the number of unstable windows

After the edge element in each window is updated once, compare the values of each edge element at time t and time $t + \Delta t$ to obtain the number of elements which have been changed ($<=$ *threshold*). In this way, the number of windows which still do not reach stable status ($N_{unstable}$) will be obtained.

Step 5. Termination check

If $N_{unstable} < M_{unstable}$ (the maximum acceptable unstable element number), or the acceptable iteration value is reached, the edge enhancement procedure will be terminated and the final result is stored in a file called Map^N corresponding to north orientation. Otherwise, going to step 3.

It is easy to see that the computation complexity to build the input maps is $O(n^2)$. Step 2 only uses constant time to set up the initial parameters. Since the window size (7×7) is fixed, to calculate the time derivation of the central unit in a window by using equation 6.2 or 6.4 only costs constant time. Hence, the time to compute all edge element in an $n \times n$ input map is $O(n^2)$. Step 4 is a simple comparison procedure, which only takes $O(n^2)$ time to do it. To sum up the above analysis, the computational complexity of the proposed Hopfield neural network for edge enhancement is $O(n^2)$.

After applying the above procedure to edge enhancement at each orientation, eight enhanced edge maps are obtained. Let z_{ij} represent the element in the final

assembled map, and y_{ij}^{N}, y_{ij}^{S}, ..., y_{ij}^{NW} be the elements corresponding to the eight map outputs Map^{N}, Map^{S}, ..., Map^{NW}. Then,

$$z_{ij} = Max(y_{ij}^{N}, y_{ij}^{S}, y_{ij}^{E}, y_{ij}^{W}, y_{ij}^{SE}, y_{ij}^{SW}, y_{ij}^{NE}, y_{ij}^{NW})$$

The value of each element in the edge map is the edge credit in [1,-1]. It is necessary to represent each element in edge map by its grey value (G_{ij}) so that the edge map is able to be displayed through a computer. The transferring can be done by the following rule:

$$G_{i,j} = \begin{cases} 0 & \text{if } z_{ij} < 0 \\ 255 * z_{ij} & \text{otherwise} \end{cases}$$

Chapter 7

Conclusions

7.1 Experiment & Result

In this thesis, a system for edge detection and edge enhancement has been presented. The input to the system is a digital image. In the pre-processing phase, the samples used to train the proposed fuzzy neural network are adaptively fuzzified first. Hence, the number of learning samples is reduced significantly. After carefully analyzing the properties of the typical edge patterns, two different window division approaches were used corresponding to two types of edges, edges with horizontal and vertical orientations, and edges with diagonal orientations. Taking advantage of the window division, the structure information was taken into consideration in the fuzzy neural network design so that the proposed fuzzy neural network is powerful enough to accomplish the edge detection after training.

The proposed fuzzy neural network is a three layer feedforward network which is capable of handling uncertainty or impreciseness in the input representation. The inputs of the first layer represent the darkness or brightness of pixels of the given pattern. The neuron output of the second layer measures how dark or bright the corresponding block is. And, the neuron outputs of the third layer decide the certainty that the pattern belongs to each class. The proposed learning algorithm can be viewed as an extension of the BP algorithm in the case of inputs represented by the gray values of input pixels and fuzzy target outputs.

The proposed Hopfield neural network is constructed based on constraint satisfaction, which is represented by energy functions, and a competitive mechanism. The eight orientations are considered separately. Each energy function is built according to the stable structure of the window in the considered orientation. The window reaching its stable structure yields the minimization of the energy function.

The system has been implemented using C language and simulated on a Sun Sparc workstation under the Unix operating system. The fuzzy neural network has been trained by gradually adding typical edge samples until the performance of the system had no further improvement after 120 training samples were used. Then the system is used to detect edges in images. The statistical results of the system performance on a significant test data set is shown on Table 7.1. Three criteria are taken into account:

1. Edge localization error: an edge pixel detected with a location error $0 \leq$

$e_i \leq 2$ pixels from the true position of the edge pixel in the ideal image is considered being detected "normally" with the location error e_i. The variance of e_i gives a measure of edge localization error.

2. False edge error: an edge pixel detected with $e_i > 2$ is considered as a false edge. The ratio of "number of false edge pixels/ total number of edge pixels" measures a false edge error.

3. Missing edge error: an edge pixel which is not normally detected is considered as a missing edge pixel. The ratio of "number of missing edge pixels/ total number of edge pixels" measures missing edge error.

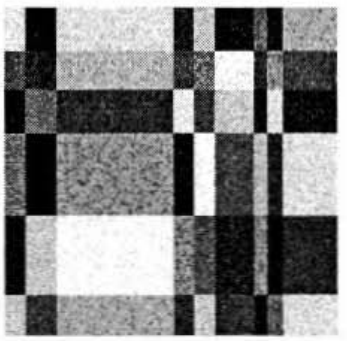
Number of Training Samples	Edge localization error	False edge error (%)	Missing edge error (%)
60	0.0355	3.12	6.58
80	0.0128	1.47	4.61
100	0.0097	0.86	4.35
120	0.0094	0.84	4.31
140	0.0094	0.84	4.31
200	0.0094	0.84	4.30
500	0.0094	0.84	4.30

Table 7.1: Statistical result of the system performance on a test data set

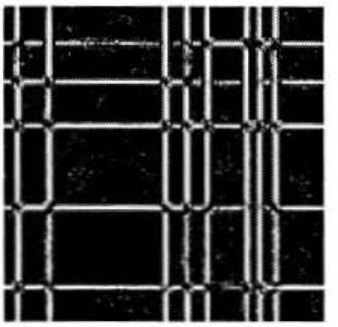
The proposed system was also tested against a set of images with additive random noise and non-uniform illumination. It detected the edge elements effectively in the eight orientations. Compared with the other edge detection operators, such as different masks, the proposed fuzzy neural network works better (see Figure

7.1).

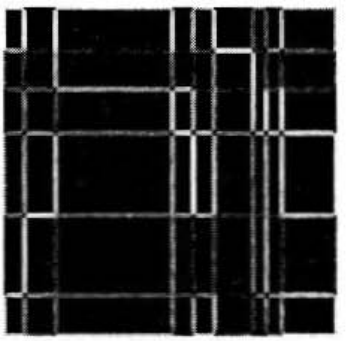
Based on the good results produced by fuzzy neural network, a set of efficient edge maps were obtained so that a better final measurement is able to be achieved by using these maps to form the input of the Hopfield network. Compared with other typical edge extraction methods, such as the optimal difference recursive filter (DRF), the first derivative operator for symmetric exponential filter (GEF), and the second derivative operator for symmetric exponential filter (SDEF), the proposed system also exhibited a better performance (see Figure 7.2 and Figure 7.3). Some other demonstrations are attached in the appendix.



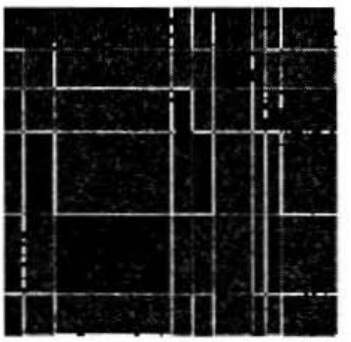
(a) Original noisy image



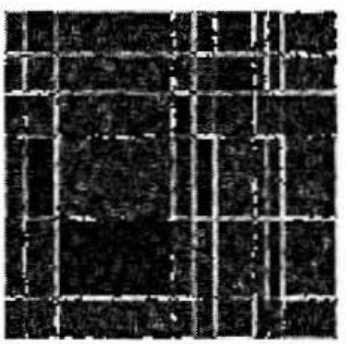
(b) using proposed FNN



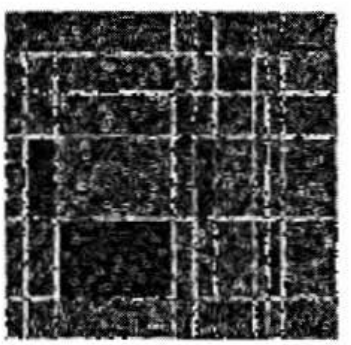
(c) using xv package



(d) using Rebert mask

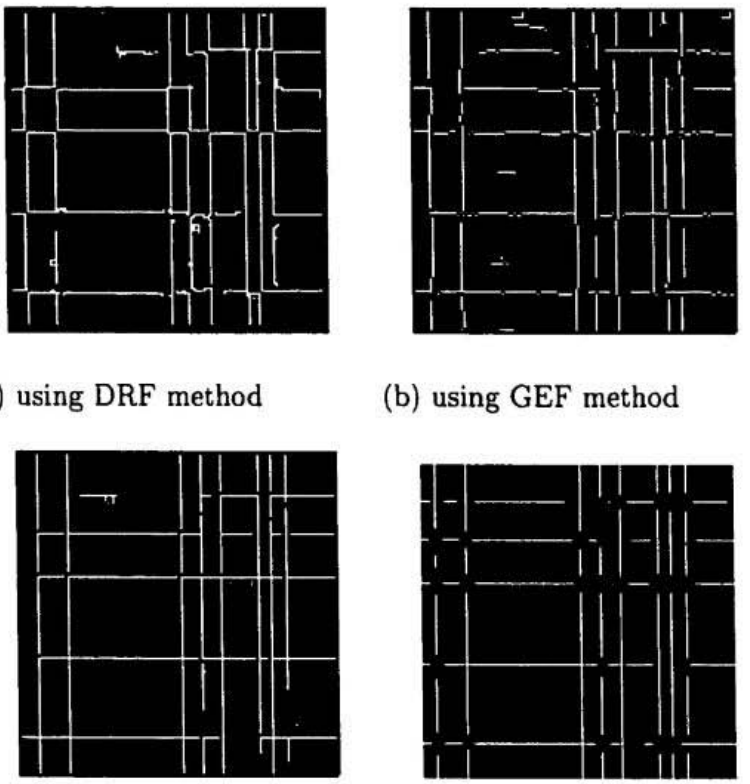


(e) using Prewitt mask



(f) using Sobel mask

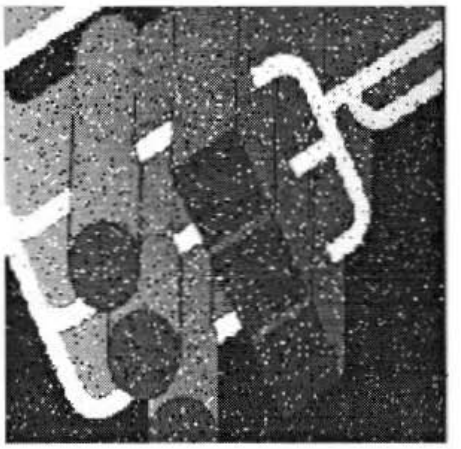
Figure 7.1: The demonstration of the test image(1)



(a) using DRF method (b) using GEF method

(e) using SDEF method (f) using proposed HNN

Figure 7.2: Comparison of the results of edge enhancement by using HNN and other methods

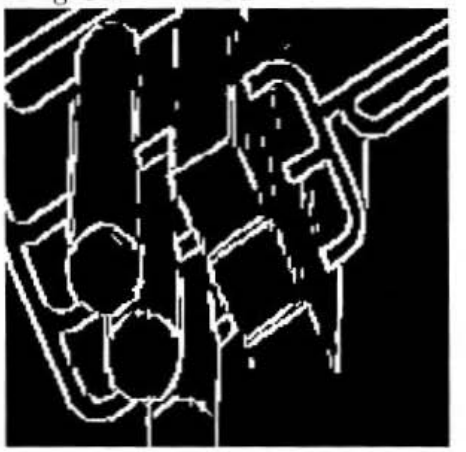


(a) The original image

(b) using DRF method

(c) using GEF method

(e) using SDEF method



(f) using proposed system

Figure 7.3: The demonstration of the test image (2)

7.2 Contributions and Remarks

1. Different pixels with the same grey value may represent relatively dark in one window, and relatively bright in the other window. For those windows, having the same edge structure, but the values of pixels inside each window in different ranges, adaptive fuzzification makes the pixels measured by the degree of darkness based on the local window. Thus, the patterns with the same edge structure, but with different grey values, will look like the same pattern after adaptive fuzzification. This cuts down the training samples significantly.

2. Knowledge of the structure of edge patterns is taken into account by using window division techniques as discussed in chapter 4. The fuzzy neural network design benefited by fully using the structure information after window division.

3. The artificial neural network approach, having adaptive and learning abilities and a high-speed parallel structure, combined with fuzzy logic theory and having the ability to process uncertain or imprecise information, forms a fuzzy neural network which has achieved a more powerful tool for edge detection. The experimental results and comparisons have shown the success of these combinations.

4. Many combinatorial optimization problems can be solved by Hopfield neural networks. The solution lies on the exploration and representation of constraints.

When the neural networks are properly constrained by the energy function, acceptable solutions can be found. The image edge enhancement in this research supports this point.

5. The original gray-level tested images are degraded and corrupted by 15 % to 35 % additive random noise and non-uniform illumination. At the first phase, the fuzzy neural network detected edges in the given images. The classification is very good compared with other edge detection operators. Though noise can be viewed, it is comparatively small. At the second phase, the Hopfield neural network enhanced edge image by filling up missing edges and removing false edges caused by noise. From the experimental results shown, the proposed system is able to handle noise properly at each stage.

7.3 Directions for Future Work

Although the proposed system for edge detection and edge enhancement works very well, more research is needed to further improve the performance of the system.

7.3.1 Thinning of edges

Post-processing could be applied on the final image. One possible method of thinning can be achieved by using the morphological approach. One basic morphological algorithm discussed in [22] is a "thinning" algorithm. The process is to

thin A (an image) by one pass with a structure element B_1 , then thin the result with one pass of B_2, and so on, until A is thinned with one pass of B_n. The entire process is repeated until no further changes occur.

7.3.2 Recovering consecutive missing edge elements

Some missing edge elements are not interpolated well, especially when they occur in more than three consecutive spatial positions. When the neighboring pairs were considered in chapter 6 to derive the updating rule, only those elements which are one row or one column far from the center element were taken into account. Therefore, a possible technique to overcome this is to extend the range of neighboring elements of the center element.

7.3.3 Improving the fuzzy neural network on classifying the intersection of edges

From the experiment result (see Figure 7.1), it is observed that sometimes the intersection of two edges was classified as a dark pattern or bright pattern. The reason is that when this happened, there are three blocks which are relatively dark (or relatively bright) and the other one block is relatively bright (or relatively dark), the fuzzy neural network will then output high credit to one of the non-edge classes. One possible method to solve this problem is to add one new edge pattern (class) corresponding to the intersection case. If the neural network is trained by

using a set of edge intersection samples, then the neural network should be able to classify the intersection of edges.

7.3.4 Finding more effective energy function

The major problem with the Hopfield neural network appears to be the local minimum problem. That is, the solution reached does not represent a global optimal solution to the problem of energy function minimization. In this research, the energy function derived is based on the stable structures for a window. However, it did not cover the cases when there is an end point inside the window. The maximum number of unstable windows was considered in checking the termination conditions for the updating. Therefore, the energy function should be improved to better reflect the constraints of the stable states so that the network stabilized on a class of good solutions.

Bibliography

[1] G. R. Arce and M. P. Mcloughlin, "Theoretical Analysis of Max/Median Filters", IEEE Trans. Acoust., Speech, Signal Processing, Vol. 35, pp. 60-69, Jan. 1987.

[2] G. R. Arce and R. E. Foster, "Detail-preserving Ranked-order Based Filter For Image Processing". IEEE Trans. Acoust., Speech, Signal Processing, Vol. 37, pp. 83-98, Jan. 1989.

[3] G. Ashish. N. R. Pal and S. K. Pal, "Self-Organization for Object Extraction Using a Multilayer Neural Network and Fuzziness Measures". IEEE Trans. on Fuzzy Systems, Vol. 1, No. 1, pp. 54-68, Feb. 1993.

[4] K. E. Barner and G. R. Arce, " Permutation Filters: A Class of Nonlinear Filters Based On Set Permutations", IEEE Trans. Signal Processing, Vol. 42, pp. 782-798, Apr. 1994.

[5] D. H. Ballard and C. M. Brown "Computer Vision", Prentice-Hall Inc, New Jersey, 1982.

[6] S. P. Banks, "Mathematical Theories of Nonlinear System", New York: Prentice-Hall, 1989.

[7] J. Ben-Arie and K. R. Rao "A Novel Approach for Template Matching by Non-Orthogonal Image Expansion", IEEE Trans. on Circuits and Systems for Video Technology, Vol. 3, No. 1, pp. 71-84. 1993.

[8] M. Braae and D. A. Rutherford, "Fuzzy Relation In A Control Setting", Kybernetics Vol. 7, No. 31, pp. 85-188, 1978.

[9] A. E. Bryson and Y. C. Ho. "Applied Optimal Control". New York: Distributed by Halsted Press, 1975.

[10] J. Canny, "A Computational Approach to Edge Detection", IEEE Trans. Pattern Analysis and Machine Intelligence, Vol. 8, No. 6, pp. 679-687, 1986.

[11] J. Canny "A Computational Approach to Edge Detection", IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. 29, No. 1, pp. 47 - 59, 1997.

[12] G. A. Carpenter and S. Grossberg, "ART2: Self-organization of Stable Category Recognition Codes for Analog Input Patterns", Applied Optics 26: pp. 4919-4930, 1987.

[13] G. A. Carpenter, S. Grossberg, and D.B. Rosen, "Fuzzy ART: An Adaptive Resonance Algorithm for Rapid, Stable Classification of Analog Patterns", IJCNN, Vol. 2, pp. 411-416, 1991.

[14] K. R. Castleman, "Digital Image Processing", Prentice-Hall, NJ. 1996.

[15] Y. Chauvin, D. E. Rumelhart, "Back-propagation", Hillsdale, NJ: Lawrence Erlbaum Associates, 1995.

[16] E. J. Coyle and J. H. Lin, "Stack Filters And Median Absolute Error Criterion", IEEE Trans. Acoust., Speech, Signal Processing, Vol. 36, pp. 1244-1254, Aug. 1988.

[17] E. J. Coyle, J. H. Lin, and M. Gabbouj, "Optimal Stack Filtering and The Estimation and Structural Approaches to Image Processing", IEEE Trans. Acoust., Speech, Signal Processing, Vol. 37, pp. 2037-2066, Dec. 1989.

[18] L. S. Davis, "A Survey of Edge Detection Techniques". Computer Graphics, Image Processing, Vol. 4, pp. 248-270, 1975.

[19] W. Deng and S. S. Iyengar, "A New Probabilistic Relaxation Scheme and Its Application to Edge Detection", IEEE Trans. Pattern Analysis and Machine Intelligence, Vol. 18, No. 4, pp. 432-437, 1996.

[20] D. P. Filev, and R. R. Yager. "A Generalized Defuzzification Method via BAD Distributions", Intl. J. Intell. Syst. Vol. 6, No. 7, pp. 687-697, 1991.

[21] S. Geman and D. Geman, "Stochastic Relaxation, Gibbs Distributions and Bayesian Restoration of Images", IEEE Pattern Analysis and Machine Intelligence, Vol. 9, No 6, pp. 721-741, 1984.

[22] R. C. Gonzalez and R. E. Woods, "Digital Image Processing", Addison-Wesley Publishing Company, 1992.

[23] S. Grossberg, "Competitive Learning: From Interactive Activation to Adaptive Reasonance", Cognitive Science: 11, pp. 23-63, 1987.

[24] H. Guo and S. B. Gelfand, "Classification Trees With Neural Network Feature Extraction", IEEE Trans. Neural Networks Vol.3: pp. 923-933, 1993.

[25] E. R. Hancock and J. Kittler, "Edge-Labeling Using Dictionary-Based Relaxation", IEEE Trans. Pattern Analysis and Machine Intelligence, Vol. 12, No. 2, pp. 165-181, 1990.

[26] E. R. Hancock and J. Kittler, "Discrete Relaxation", Pattern Recognition, Vol. 23, pp. 711-733, 1990.

[27] E. R. Hancock and J. Kittler, "A Label Error Process for Discrete Relaxation", IEEE 10th ICPR, Vol. 1, pp. 523-528, 1990.

[28] F. R. Hansen and H. Elliott, "Image Segmentation Using Simple Markov Random Fields", Computer Graphics and Image Processing, Vol. 20, pp. 101-132, 1982.

[29] R. C. Hardie and C. G. Boncelet, "LUM Filters: A Class Rank Order Based Filters For Smoothing And Sharpening", IEEE Trans, Signal Processing, Vol. 41, pp. 1061-1076, May 1993.

[30] R. C. Hardie and K. E. Barner, "Rank Conditioned Rank Selection Filters For Signal Restoration", IEEE Trans. Image Processing, Vol. 3, pp. 192-206, Mar. 1994.

[31] R. C. Hardie and K. E. Barner, "Extended Permutation Filters and Their Application to Edge Enhancement", IEEE Trans. Image Processing, Vol.5, No. 6, pp. 855-867, June 1996.

[32] J. J. Hopfield, "Neural Networks and Physical Systems With Emergent Collective Computational Abilities", Proc. Nat. Acad. Sci. USA 79: pp. 2554-2558, 1982.

[33] J. J. Hopfield, "Neurons With Graded Responses Have Collective Computational Properties Like Those of Two-state Neurons", Proc. Nat. Acad. Sci. USA 81: pp. 3088-3092, 1984.

[34] R.A. Hummel, and S. W. Zucker, "On the Foundations of Relaxation Labeling Processes", IEEE Trans. Pattern Analysis and Machine Intelligence, Vol. 5, pp. 267-287, 1983.

[35] H. Ishibuchi, H. Tanaka, and H. Okada, "An Architecture of Neural Networks With Interval Weights and Its Application to Fuzzy Regression Analysis", Fuzzy Set Systems. Vol. 57, pp. 27-39, 1993.

[36] B. I. Justusson, "Median filter: Statistical Properties", in Topics in Applied Physics, vol. 42: "Two-Dimensional Digital Signal Processing, II- Transforms

and Median Filters", T.S. Huang, Ed. New York: Springer-Verlag, pp. 161-196, 1981.

[37] J. Kittler, and E. R. Hancock, "Combining Evidence in Probabilistic Relaxation", Int'l J. Pattern Recognition Artificial Intelligence, Vol. 3, pp. 29-52, 1989.

[38] S. J. Ko and Y. H. Lee, "Center Weighted Median Filters and Their Applications To Image Enhancement", IEEE Trans. Circuits System, Vol. 38, pp. 984-993, Sept. 1991.

[39] T. Kohonen, "Self-Organization and Associative Memory", Springer-Verlag, Berlin, 1988.

[40] Y. H. Kuo, C. I. Kao, and J. J. Chen, "A Fuzzy Neural Network Model and Its Hardware Implementation", IEEE Trans. on Fuzzy Systems, Vol. 1, No. 3, pp. 171-183, Aug. 1993.

[41] P. M. Larsen, "Industrial Applications of Fuzzy Logic Control", Int.J. Man-Mach. Stud. Vol. 12, No. 1: pp. 3-10, 1980.

[42] Y. LeCun, "A Learning Procedure For Asymmetric Network", Cognitiva 85, pp. 599-604, Paris, 1985.

[43] C. C. Lee, "Fuzzy Logic in Control Systems: Fuzzy Logic Controller - Parts I and II", IEEE Trans. Syst. Man Cybern., Vol. SMS-20, No. 2, pp. 404-435, 1990.

[44] Y. H. Lee and A. T. Fam, "An Edge Gradient Enhancing Adaptive Order Statistic Filter". IEEE Trans. Acoust.. Speech. Signal Processing. Vol. 35, pp. 680-695, May 1987.

[45] C. T. Lin and C. S. George Lee, "Reinforcement Structure/Parameter Learning for Neural-Network-Based Fuzzy Logic Control Systems," IEEE Trans. on Fuzzy Systems, Vol. 2, No. 1, pp. 46-63, Feb. 1994.

[46] J. H. Lin and E. J. Coyle, "Minimum Mean Absolute Error Estimation Over The Class of Generalized Stack Filters", IEEE Trans. Acoust., Speech, Signal Processing, Vol. 38, pp. 663-678, Apr. 1990

[47] H. G. Longbotham and D. Eberly, "The WMMR Filters: A Class of Robust Edge Enhancers", IEEE Trans. Signal Processing, Vol. 41, pp. 1680-1685, Apr. 1993.

[48] E. H. Mamdani, and S. Assilian, "An Experiment In Linguistic Synthesis With a Fuzzy Logic Controller", Int. J.Man-Mach. Stud. Vol. 7, No. 1, pp. 1-13, 1975.

[49] A. Martelli, "Edge Detection Using Heuristic Search Methods", CGIP, Vol. 1, pp. 169-182, 1972.

[50] J. L. McClelland, D. E. Rumelhart, "Explorations In Parallel Distributed Processing", The MIT Press, Cambridge, Massachusetts, 1986.

[51] T. Munakata, and Y. Jani. "Fuzzy Systems: An Overview", Commun. ACM Vol. 37, No. 3, pp. 69-76, 1994.

[52] R. Nevatia and K. R. Babu "Linear Feature Extraction and Description", Computer Graphics and Image Processing, Vol. 13, pp. 257-269, 1980.

[53] A. Nieminen, P. Heinonen, and Y. Neuvo, "A New Class Of Detail Preserving Filters For Image Processing", IEEE Trans. Pattern Analysis and Machine Intelligence, Vol. 9, pp. 74-90, Jan. 1987.

[54] Y. H. Pao, "Adaptive Pattern Recognition and Neural Networks", Reading, MA: Addison-Wesley Publishing Company, 1989.

[55] R. H. Park and W. Y. Choi, "A New Interpretation of the Compass Gradient Edge Operators", Computer Vision, Graphics and Image Processing, Vol. 42, No 2, pp. 259-265, 1989.

[56] L. Pelkowitz, "A Continuous Relaxation Labeling Algorithm for Markov Random Fields", IEEE Trans. Systems, Man, and Cybernetics, Vol. 20, No. 3, pp. 709-715, 1990.

[57] A. J. Pinho and L. B. Almeida, "Edge Detection Filters Based On Artificial Neural Networks", 8th International Conference, ICIAP Image Analysis and Processing, 974, pp 159-164, 1995.

[58] K. R. Rao and J.Ben-Arie, "Multiple Template Matching Using Expansion Matching", IEEE Trans. on Circuits and Systems for Video Technology, Vol. 4, No. 5, pp. 490-503, 1994.

[59] L. G. Roberts, "Machine Perception of Three-dimensional Solids", in "Computer Methods in Image Analysis", J.K. Aggarwal, R. O. Duda, and A. Rosenfeld Eds. Los Angeles: IEEE, pp. 285-323, 1977.

[60] G. S. Robinson, "Edge Detection by Compass Gradient Masks", Computer Graphics and Image Processing, Vol. 6, pp. 492-501, 1977.

[61] A. Rosenfeld, R. A. Hummel, and S. W. Zucker, "Scene Labeling by Relaxation Operations", IEEE Trans. Systems, Man, and Cybernetics, Vol. 6, pp. 420-433, 1976.

[62] M. Salotti and M. Hatimi, "A New Heuristic Search for Boundary Detection", 8th International Conference, ICIAP Image Analysis and Processing, 974, pp. 381-385, 1995.

[63] A. Sankar and R. J. Mammone, "Neural Tree Networks", In R. J. Mammone and Y.Y. Zeevi, eds., "Neural Networks", Vol. 2: 459-473, 1991.

[64] B. J. Schachter, A. Lev, S. W. Zucker, and A. Rosenfeld, "An Application of Relaxation Methods to Edge Reinforcement", IEEE Trans. Systems, Man, and Cybernetics, Vol. 7, pp. 813-816, 1977.

[65] F. M. Silva and L. B. Almeida and C. J. Wellekens, "Neural Networks", Proc. EURASIP Workshop, Sesimbra, Portugal, Springer-Verlag, 1990.

[66] P. K. Simpson, "Fuzzy Min-Max Neural Networks - Part 2: Clustering", IEEE Trans. on Fuzzy Systems, Vol. 1, No. 1, pp. 32-45, Feb. 1993.

[67] L. A. Spacek, "Edge Detection and Motion Detection", Image and Vision Computing, Vol. 4, No. 1, pp. 43-56, 1986.

[68] R. N. Strickland, T. Draelos and Z. Mao, "Edge Detection in Machine Vision Using a Simple L1 Norm Template Matching Algorithm", Pattern Recognition, Vol. 23, No 5, pp. 411-421, 1990.

[69] A. W. Szeto, "Hierarchical Artificial Neural Networks For Edge Enhancement". Master Thesis. Department of Computer Science, Memorial University of Newfoundland, 1991.

[70] C. Tao, W. Thompson, and J. Taur, "A Fuzzy If-Then Approach to Edge Detection", Proc. Second Int'l Conf. Fuzzy Systems, Vol. 2 pp. 1356-1360, San Francisco, 1993.

[71] M. F. Tenorio and W. T. Lee, "Self-organizing Network For Optimum Supervised Learning", IEEE Trans. Neural Networks, Vol. 1 No. 1: pp. 100-110, 1990.

[72] C. Tyan and P. Wang, "Image Processing-Enhancement, Filtering and Edge Detection Using the Fuzzy Logic Approach", Proc. Second Int'l Conf. Fuzzy Systems, Vol. 1, pp. 600-605, San Francisco, 1993.

[73] A. Waibel, "Modular Construction Of Time-delay Neural Networks for Speech Recognition", Neural Comput. Vol. 1, No. 1, pp: 39-46, 1989.

[74] Z. Wang, K. R. Rao and J. Ben-Arie, "Optimal Ramp Edge Detection Using Expansion Matching", IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. 18, No. 11, pp. 1092-1097. 1996.

[75] P. Wendt, E. J. Coyle, and N. C. Gallagher, "Stack Filters", IEEE Trans. Acoust., Speech, Signal Processing, Vol. 34, pp. 898-911, Aug. 1986.

[76] R. J. Williams and D. Zipser, "A Learning Algorithm for Continually Running Fully Recurrent Neural Networks", Neural Computation, Vol. 1, No. 2, pp. 270-280, 1989.

[77] R. R. Yager, " Fuzzy Sets and Approximate Reasoning in Decision and Control", In Proc. IEEE Int. Conf. Fuzzy System, pp. 415-428, San Diego, 1992.

[78] O. Yli-Harja, J. Astola, and Y. Neuvo, "Analysis Of The Properties Of Median And Weighted Median Filters Using Threshold Logic And Stack Filter Representation", IEEE Trans. Acoust., Speech, Signal Processing, Vol. 39, pp. 395-410, Feb. 1991.

[79] L. A. Zadeh. "The Concept of a Linguistic Variable and Its Application to Approximate Reasoning-I", Information Science 8, pp. 199-249, 1975.

[80] L. A. Zadeh, "The Concept of a Linguistic Variable and Its Application to Approximate Reasoning-II", Information Science 8, pp. 301-357, 1975.

[81] L. A. Zadeh, "The Concept of a Linguistic Variable and Its Application to Approximate Reasoning-III", Information Science 9, pp. 43-80, 1975.

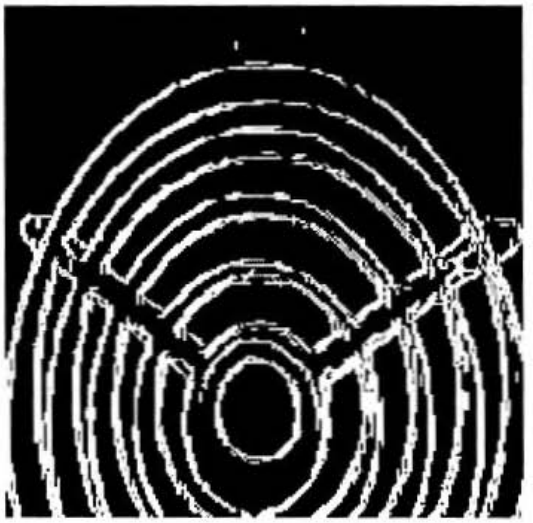
[82] S.W.Lu and A.Szeto, "Hierarchical Artificial Neural Networks for Edge Enhancement", Pattern Recognition, Vol.26, No.8, pp. 1149-1163, 1993.

Appendix A

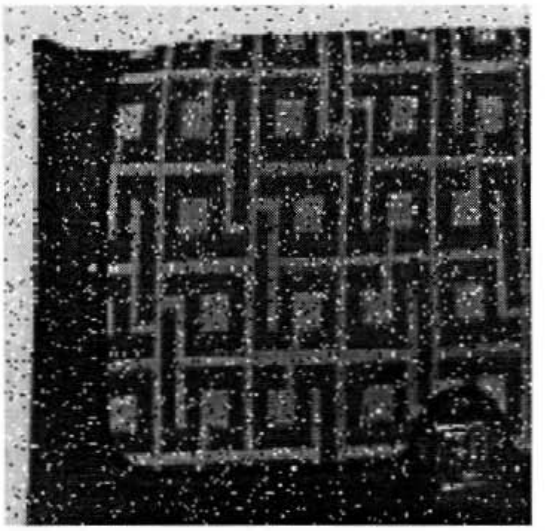
Testing results

(a) The original image

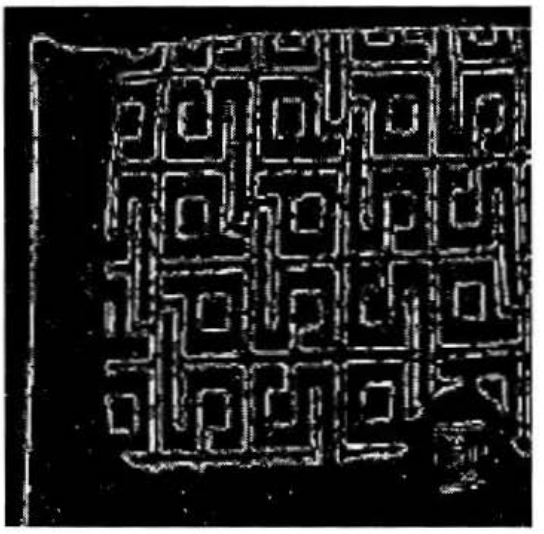
(b) Edges detected by FNN



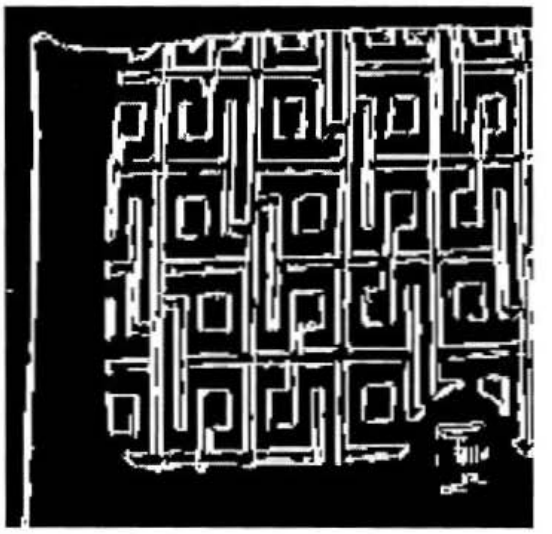
(c) Edges enhanced by HNN



(a) The original image

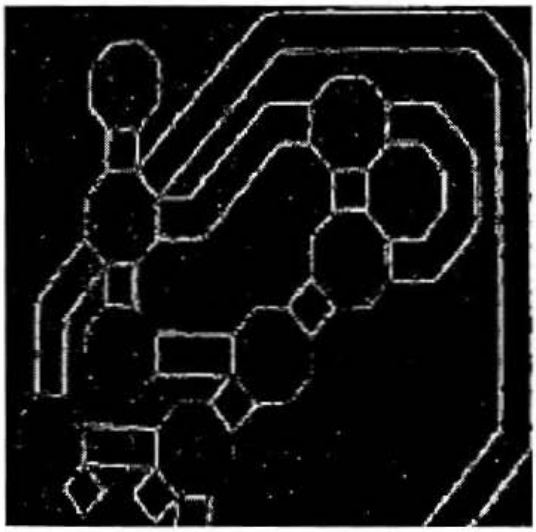


(b) Edges detected by FNN

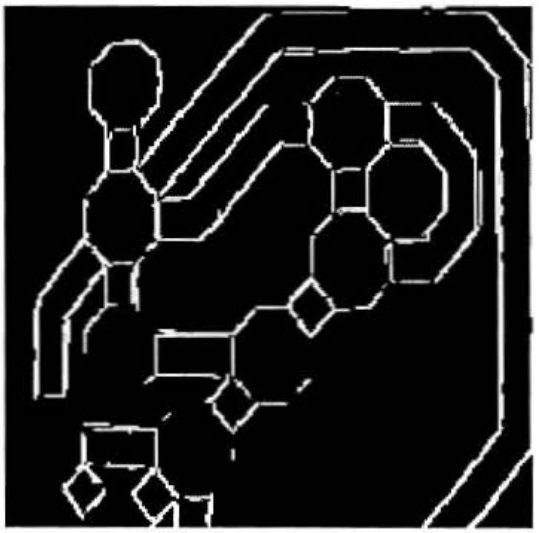


(c) Edges enhanced by HNN

(a) The original image

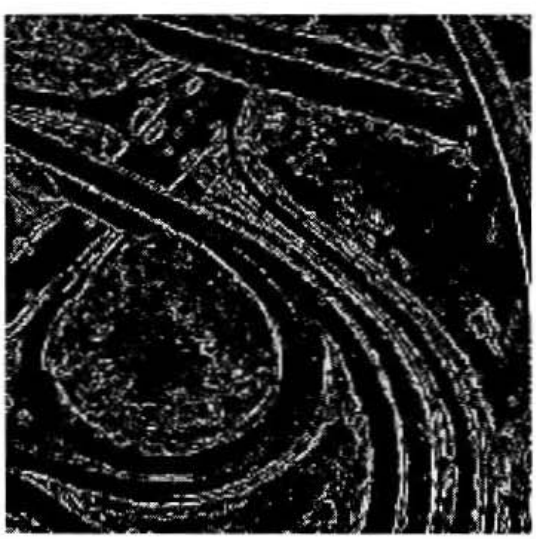


(b) Edges detected by FNN



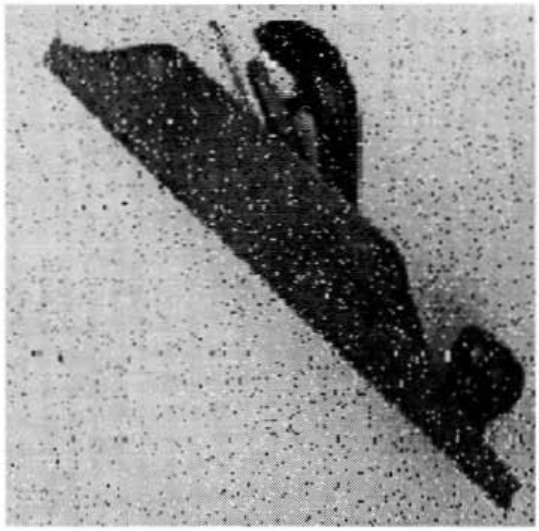
(c) Edges enhanced by HNN

(a) The original image

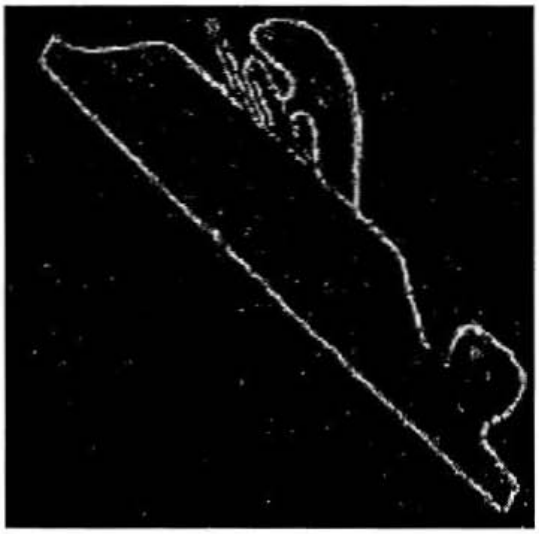


(b) Edges detected by FNN

(c) Edges enhanced by HNN

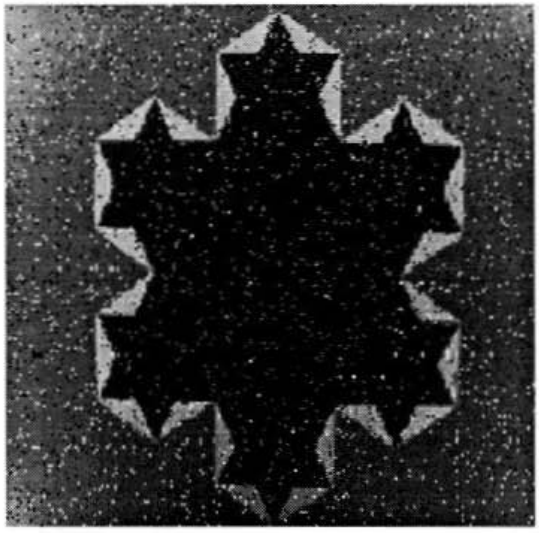


(a) The original image

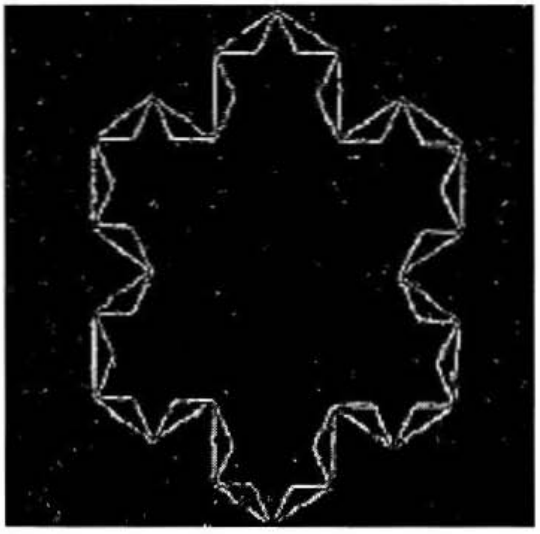


(b) Edges detected by FNN

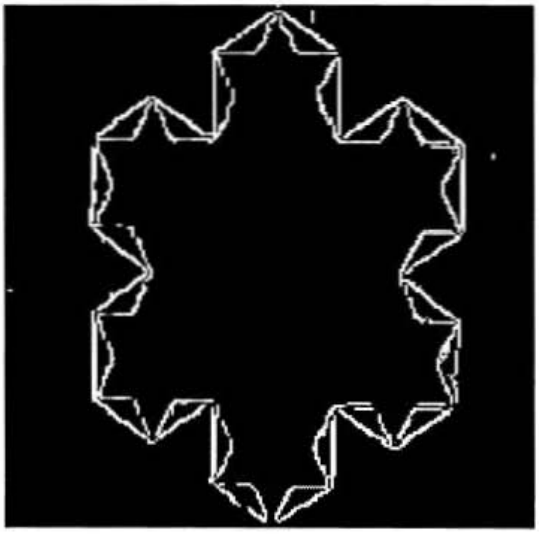
(c) Edges enhanced by HNN



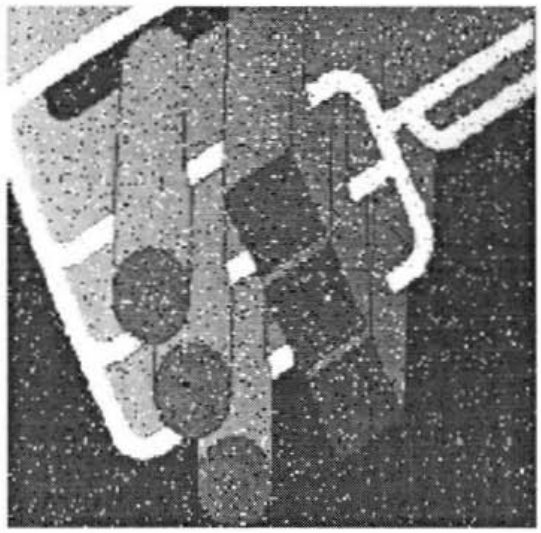
(a) The original image



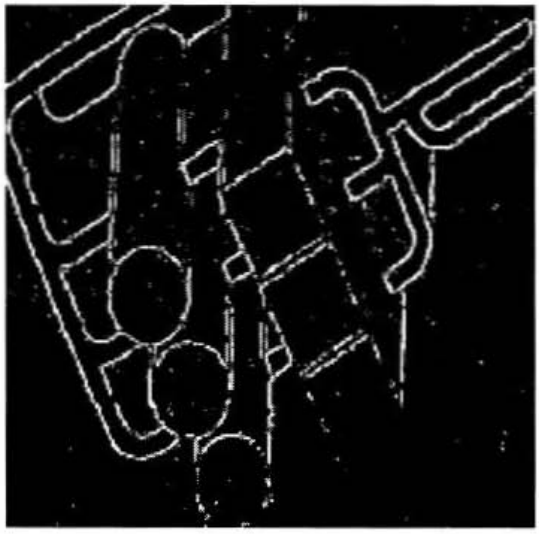
(b) Edges detected by FNN



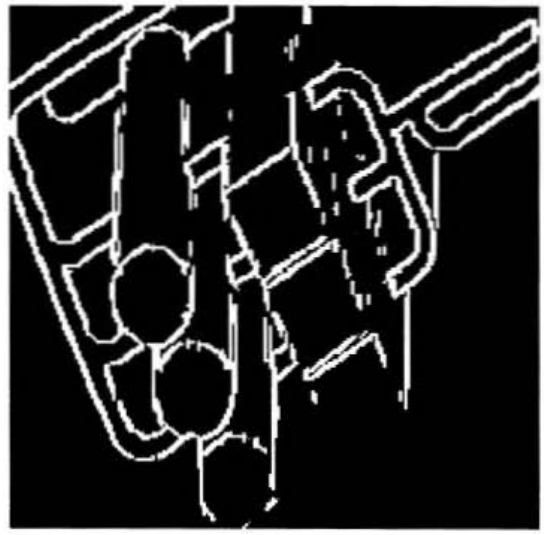
(c) Edges enhanced by HNN



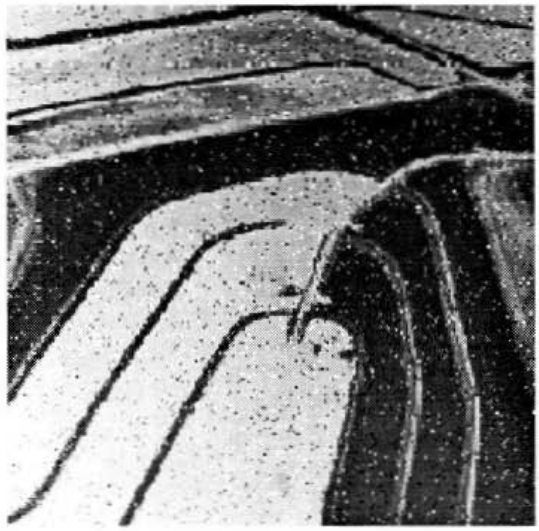
(a) The original image



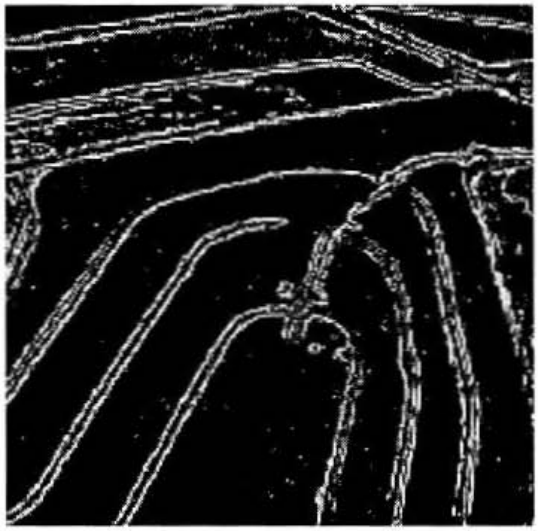
(b) Edges detected by FNN



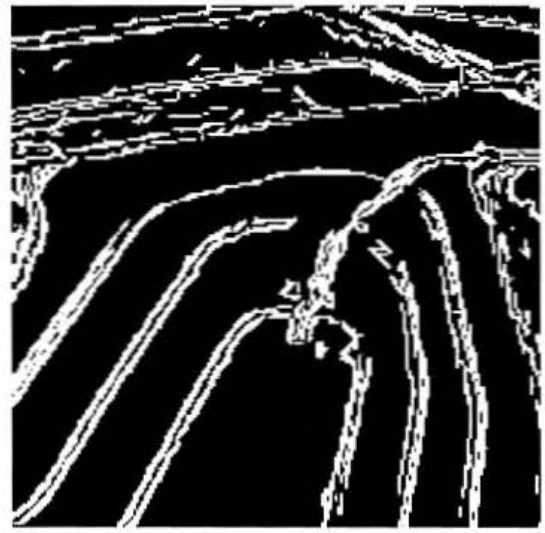
(c) Edges enhanced by HNN



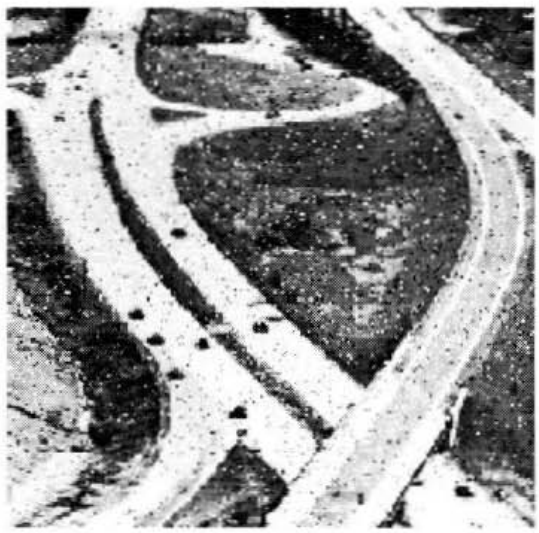
(a) The original image



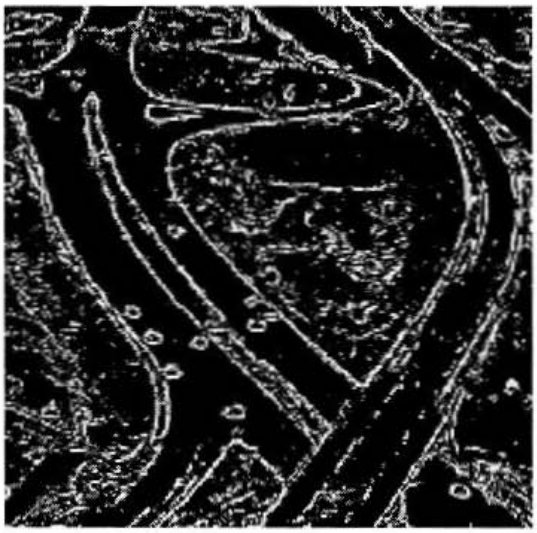
(b) Edges detected by FNN



(c) Edges enhanced by HNN

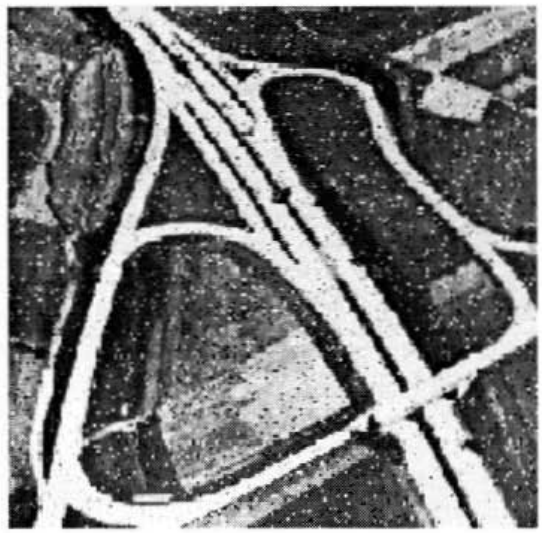


(a) The original image



(b) Edges detected by FNN

(c) Edges enhanced by HNN



(a) The original image

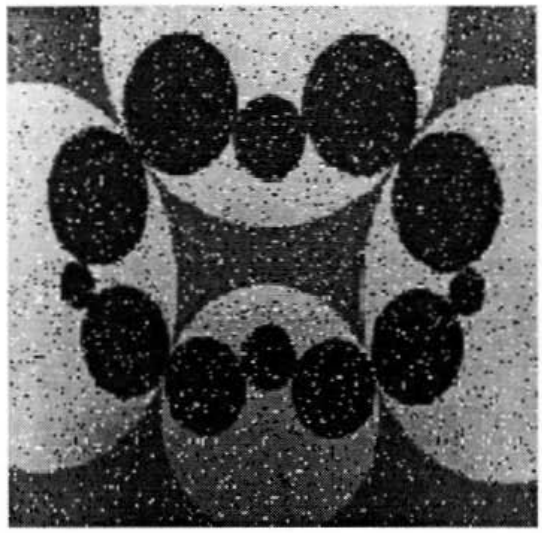
(b) Edges detected by FNN

(c) Edges enhanced by HNN

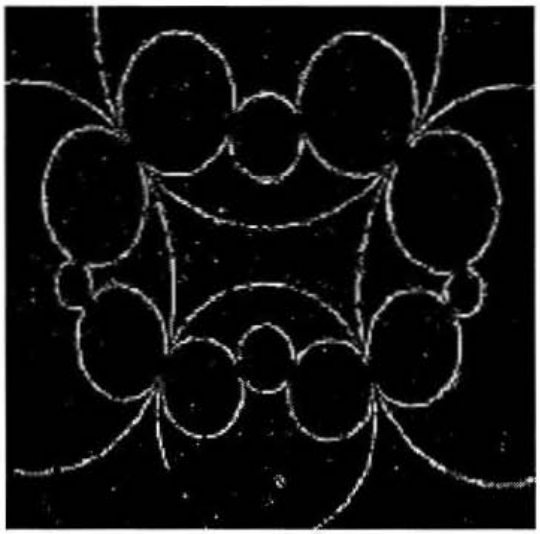
(a) The original image

(b) Edges detected by FNN

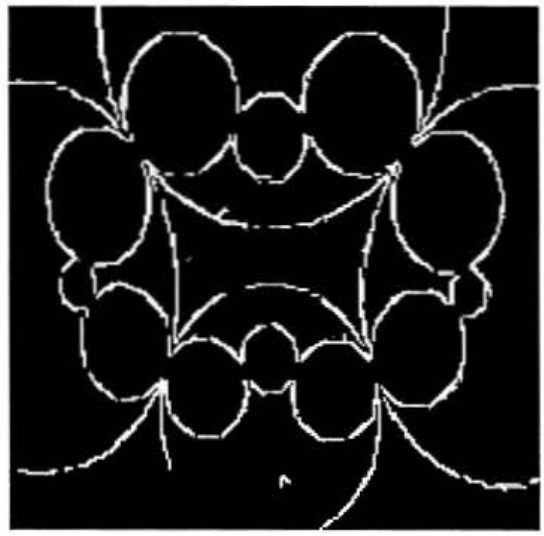
(c) Edges enhanced by HNN



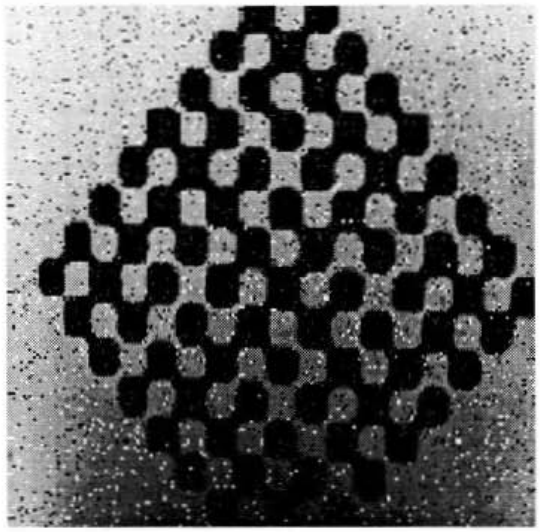
(a) The original image



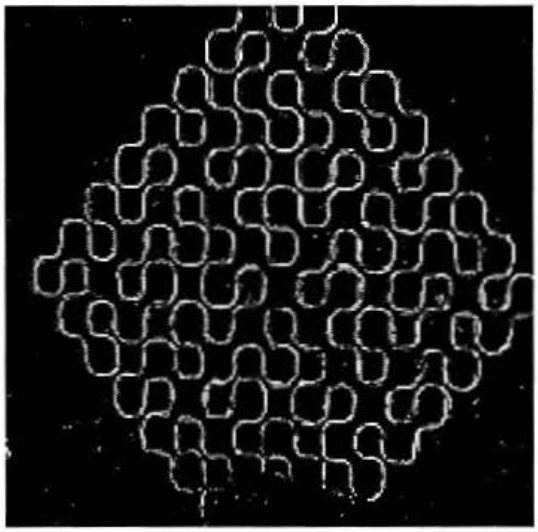
(b) Edges detected by FNN



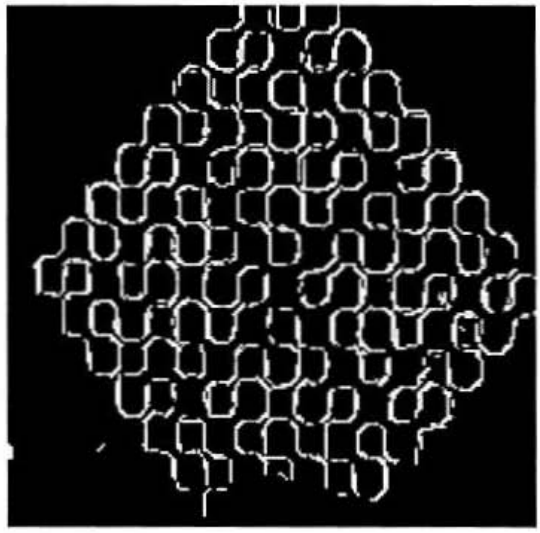
(c) Edges enhanced by HNN



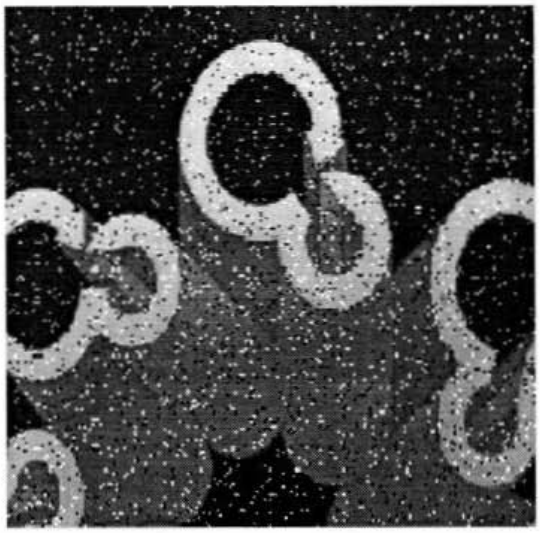
(a) The original image



(b) Edges detected by FNN

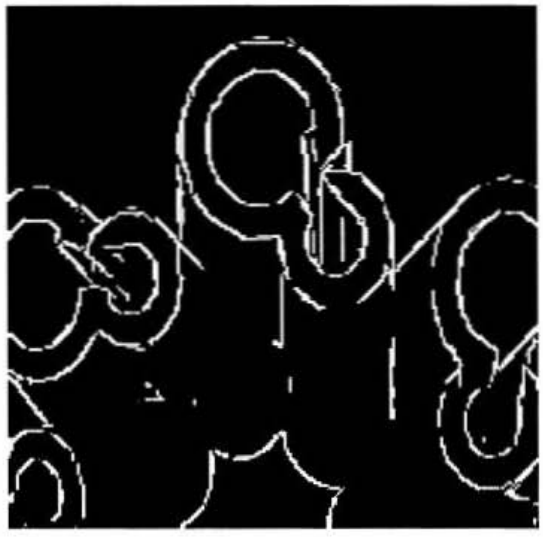


(c) Edges enhanced by HNN



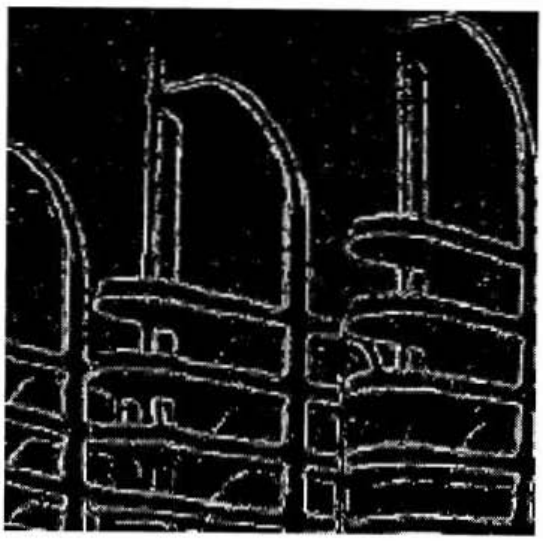
(a) The original image

(b) Edges detected by FNN

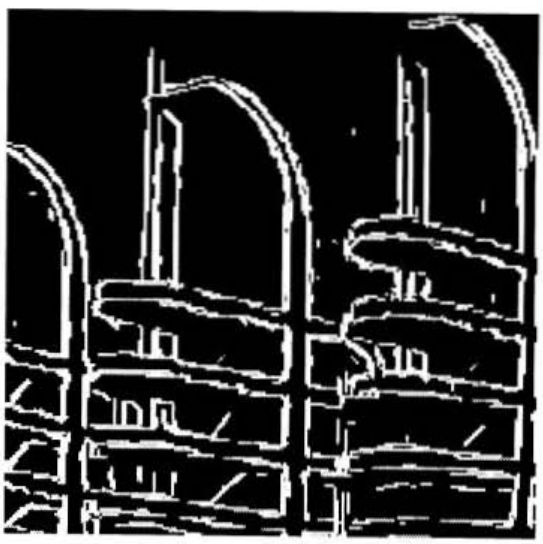


(c) Edges enhanced by HNN

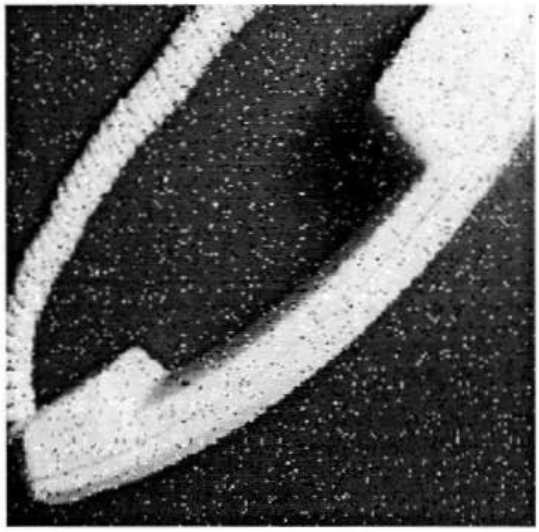
(a) The original image



(b) Edges detected by FNN



(c) Edges enhanced by HNN

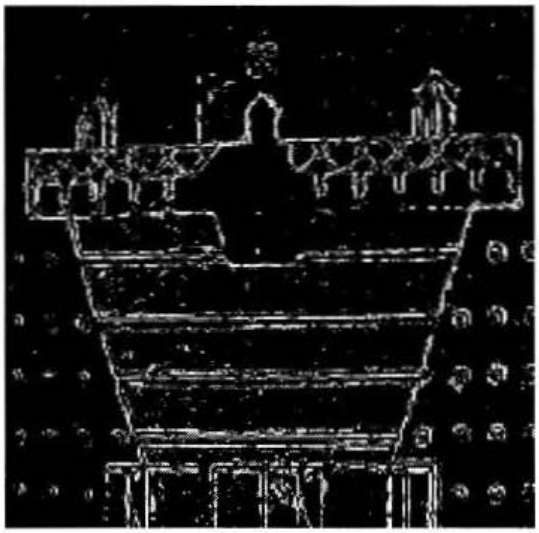


(a) The original image

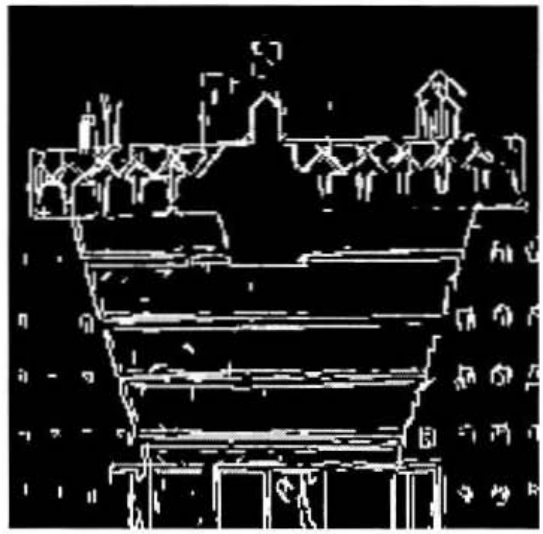
(b) Edges detected by FNN

(c) Edges enhanced by HNN

(a) The original image



(b) Edges detected by FNN



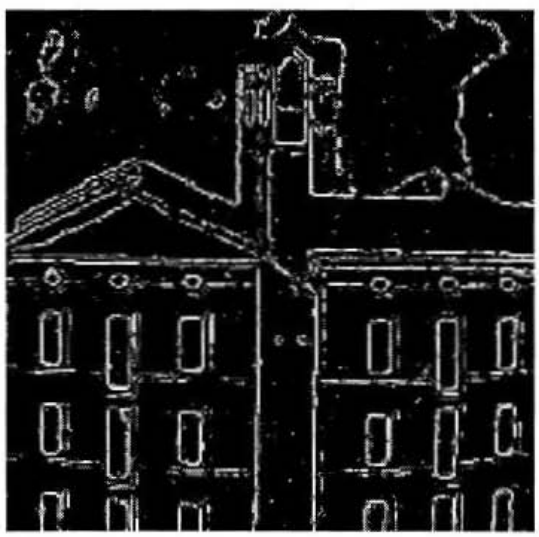
(c) Edges enhanced by HNN

(a) The original image

(b) Edges detected by FNN

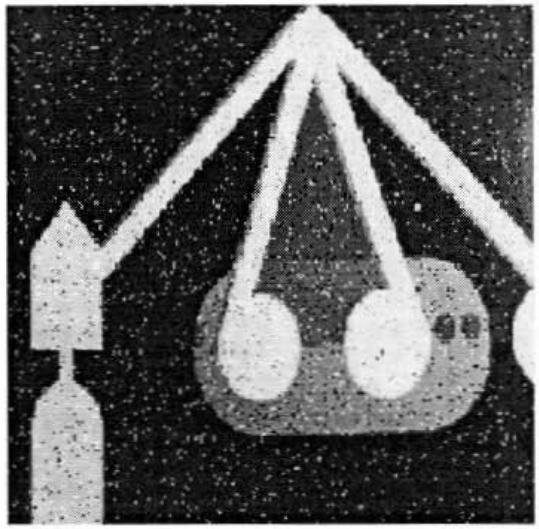
(c) Edges enhanced by HNN

(a) The original image

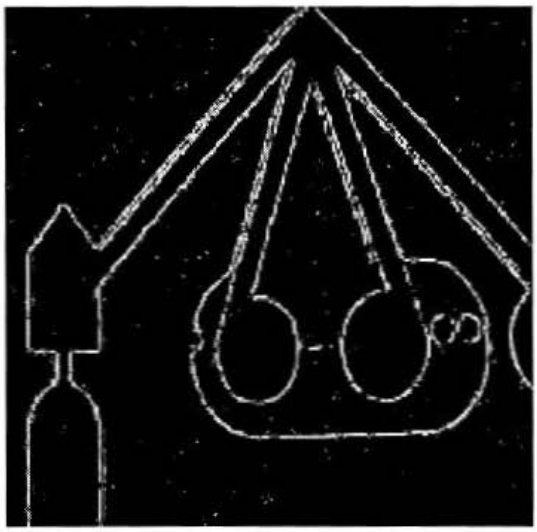


(b) Edges detected by FNN

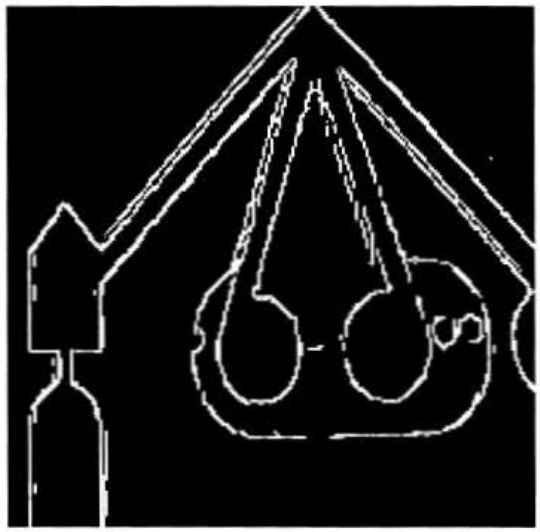
(c) Edges enhanced by HNN



(a) The original image



(b) Edges detected by FNN



(c) Edges enhanced by HNN

(a) The original image

(b) Edges detected by FNN

(c) Edges enhanced by HNN

(a) The original image

(b) Edges detected by FNN

(c) Edges enhanced by HNN