Copyright Warning & Restrictions

The copyright law of the United States (Title 17, United States Code) governs the making of photocopies or other reproductions of copyrighted material.

Under certain conditions specified in the law, libraries and archives are authorized to furnish a photocopy or other reproduction. One of these specified conditions is that the photocopy or reproduction is not to be "used for any purpose other than private study, scholarship, or research." If a, user makes a request for, or later uses, a photocopy or reproduction for purposes in excess of "fair use" that user may be liable for copyright infringement,

This institution reserves the right to refuse to accept a copying order if, in its judgment, fulfillment of the order would involve violation of copyright law.

Please Note: The author retains the copyright while the New Jersey Institute of Technology reserves the right to distribute this thesis or dissertation

Printing note: If you do not wish to print this page, then select "Pages from: first page # to: last page #" on the print dialog screen

UMI Number: 9539585

UMI

300 North Zeeb Road
Ann Arbor, MI 48103

ABSTRACT

VISUAL PATTERN RECOGNITION USING NEURAL NETWORKS

by
Jenlong Moh

Neural networks have been widely studied in a number of fields, such as neural architectures, neurobiology, statistics of neural network and pattern classification. In the field of pattern classification, neural network models are applied on numerous applications, for instance, character recognition, speech recognition, and object recognition. Among these, character recognition is commonly used to illustrate the feature and classification characteristics of neural networks.

In this dissertation, the theoretical foundations of artificial neural networks are first reviewed and existing neural models are studied. The Adaptive Resonance Theory (ART) model is improved to achieve more reasonable classification results. Experiments in applying the improved model to image enhancement and printed character recognition are discussed and analyzed. We also study the theoretical foundation of Neocognitron in terms of feature extraction, convergence in training, and shift invariance.

We investigate the use of multilayered perceptrons with recurrent connections as the general purpose modules for image operations in parallel architectures. The networks are trained to carry out classification rules in image transformation. The training patterns can be derived from user-defined transformations or from loading the pair of a sample image and its target image when the prior knowledge of transformations is unknown. Applications of our model include image smoothing, enhancement, edge detection, noise removal, morphological operations, image filtering, etc. With a number of stages stacked up together we are able to apply a series of operations on the image. That is, by

providing various sets of training patterns the system can adapt itself to the concatenated transformation. We also discuss and experiment in applying existing neural models, such as multilayered perceptron, to realize morphological operations and other commonly used imaging operations.

Some new neural architectures and training algorithms for the implementation of morphological operations are designed and analyzed. The algorithms are proven correct and efficient. The proposed morphological neural architectures are applied to construct the feature extraction module of a personal handwritten character recognition system. The system was trained and tested with scanned image of handwritten characters. The feasibility and efficiency are discussed along with the experimental results.

VISUAL PATTERN RECOGNITION USING NEURAL NETWORKS

by
Jenlong Moh

A Dissertation
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy

Department of Computer and Information Science

May 1995

APPROVAL PAGE

VISUAL PATTERN RECOGNITION USING NEURAL NETWORKS

by
Jenlong Moh

Dr. Frank Y. Shih, Dissertation Advisor — Date
Associate Professor of Computer and Information Science, NJIT

Dr. Peter A. Ng, Committee Member, — Date
Chairperson and Professor of Computer and Information Science, NJIT

Dr. David Nassimi, Committee Member, — Date
Associate Professor of Computer and Information Science, NJIT

Dr. DaoChaun Hung, Committee Member, — Date
Assistant Professor of Computer and Information Science, NJIT

Dr. Edwin Hou, Committee Member, — Date
Assistant Professor of Electrical and Computer Engineering, NJIT

BIOGRAPHICAL SKETCH

Author: Jenlong Moh

Degree: Doctor of Philosophy in Computer Science

Date: May 1995

Undergraduate and Graduate Education:

Doctor of Philosophy in Computer Science,
New Jersey Institute of Technology, Newark, New Jersey, 1995

Bachelor of Science in Computer Engineering,
National Chiao Tung University, Hsinchu, Taiwan, 1984

Major: Computer and Information Science

Journal Publications:

J. Moh and F. Y. Shih, "One-Pass Training Algorithms for Morphological Operations," submitted to *IEEE Transactions on Image Processing* on March 13, 1995.

J. Moh and F. Y. Shih, "A General Purpose Model for Image Operations Based on Multilayer Perceptrons" to be appeared in *Pattern Recognition*.

F. Y. Shih, J. Moh, and F. C. Chang, "A New ART-Based Neural Architecture for Pattern Classification and Image Enhancement without Prior Knowledge," in *Pattern Recognition*, vol.25, no.5, pp. 533-42, May 1992.

F. Y. Shih and J. Moh, "Implementing Morphological Operations Using Programmable Neural Networks," in *Pattern Recognition*, vol.25, no.1, pp.89-99, Jan. 1992.

F. Y. Shih, J. Moh, and H. Bourne, "A Neural Architecture Applied to the Enhancement of Noisy Binary Images," in *Engineering Application of Artificial Intelligence*, vol.5, no.3, pp.215-22, Nov. 1992.

BIOGRAPHICAL SKETCH
(Continued)

Conference Presentations:

J. Moh and F. Y. Shih, "A New Neural Network Architecture for Personal Handprinting Recognition" submitted to *IEEE International Conference on Image Processing*, Washington D.C., Oct. 22-25, 1995.

J. Moh and F. Y. Shih, "Neural Architectures for Mathematical Morphology and Fuzzy Morphology" *Third Annual Conference on Fuzzy Theory and Technology*, Pinehurst, NC, Nov. 13-16, 1994.

J. Moh and F. Y. Shih, "A General Purpose Model for Image Processing Based on Multilayer Perceptrons" *World Congress on Neural Networks*, San Diego, CA, June 1994.

F. Y. Shih and J. Moh, "Implementing Neural Morphological Operations using Programmable Logic," in *SPIE Conference on Intelligent Robots and Computer Vision IX: Neural, Biological, and 3-D Methods*, Boston, Nov. 1990.

F. Y. Shih and J. Moh, "Improved Adaptive Resonance Theory," *SPIE Conference on Intelligent Robots and Computer Vision IX: Neural, Biological, and 3-D Methods*, Boston, Nov. 1990.

F. Y. Shih and J. Moh, "A Self-Organization Architecture for Clustering Analysis," *23rd Hawaii International Conference on System Science*, Jan. 1990.

F. Y. Shih and J. Moh, "Image Morphological Operations by Neural Circuits," *Proceeding of IEEE 1989 International Symposium on Circuits and Systems*, Portland, Oregon, pp. 774-7, May 1989.

F. Y. Shih, J. Moh, and H. Bourne, "A Neural Architecture Applied to the Enhancement of Noisy Binary Images without Prior Knowledge," *Proceeding of IEEE Inter. Conf. Tools for Artificial Intelligence*, Herndon, VA, pp. 699-705, Nov. 1990.

This dissertation is dedicated to
my beloved wife, Hui-Fen.

ACKNOWLEDGMENT

The doctoral research should be enjoyed and indeed it was joyful. This was partly due to the helpful faculty and colleagues in the Department of Computer and Information Science, from whom, the author received encouragement. The author is grateful to the friends made during this research and would like to extend special thanks to them.

First of all, the author would like to thank his research advisor, Dr. Frank Y. Shih, for his advising and guidance during years of doctoral research. His valuable time was instrumental to the author in processing this research. The author is proud to co-author with him in many conference presentations and journal papers.

Special thanks to Dr. Peter A. Ng, the chairperson of CIS department and the CIS department for the financial support during the period of this research.

The author appreciates the help and suggestions from Dr. Dao-Chaun Hung, Dr. David Nassimi (in CIS dept.), and Edwin Hou (from ECE Dept.) for serving as members of the committee.

Sincere thanks to the author's family members for their never ending support, understanding, and encouragement during the years of advance studies. The author is grateful to his beloved wife Hui-Fen for her love and consistent support in sharing her life with him.

TABLE OF CONTENTS

TABLE OF CONTENTS
(Continued)

TABLE OF CONTENTS
(Continued)

LIST OF TABLES

LIST OF FIGURES

LIST OF FIGURES
(Continued)

LIST OF FIGURES
(Continued)

CHAPTER 1

INTRODUCTION

In the development of information processing technology a new phase has been developed, in which scientists aim to replicate many of biological neural functions artificially. The central goal is to build up new type of computers. Such kind of machines do not execute typical machine instructions of conventional computer rather are made to emulate the behavior of biological neural networks. Nowadays, these machines are well-known as *artificial neural networks* (ANN) or *neural computers*, and are also known by many names such as *connectionist models*, *parallel distributed processing models*, and so on. They are massively parallel interconnected networks of simple and usually adaptive processing elements and are characterized by their hierarchical organizations which are intended to interact with the objects of the real world in the same way as biological nervous systems do [66]. In principle, ANNs are composed of a large number of processing elements (also called *neurons*) with massive connections among neurons. The basic processing operation performed by every processing element is an analog operation or *transformation* of its input signals.

McCulloch and Pitts [76] presented the first sophisticated discussion of neurological networks. Hebb [51] then attempted to base a large-scale theory of psychology on assumption about neural networks. The first defined artificial neural network is called *perceptron* by Rosenblatt [93][94] and was extended and completed by Minsky and Papert [80]. During the same decade, the convergence theorem for machine learning was well discussed by Nilsson [85]. In Minsky and Paperts' book "*Perceptrons*," they demonstrated that perceptrons (and all other neural networks) are too limited to deserve further attention. This made a delay of researches and has caused very few discoveries

in this field until 1980's. ANNs were getting more attention during the decade at the same time when the concept of parallel processing was emphasized and demanded. It can be asserted that the revival of learning machines is made possible due to the recent development of semi-conductor very large scaled integration (VLSI) technology in making massively parallel hardwares.

According to the definition ANNs are trainable and are trained with data instead of being programmed for applications. We believe that a significant portion of today's computer programmers' programming load can be relieved by introducing this technique. Moreover, ANNs can learn and improve themselves from experiences. That is, the more data they are fed, the more accurate or complete their response. In addition, the simple operations that are designed for the processing elements can be handled in high speed and the characteristic of massively parallel structure can also be achieved with today's VLSI technology. These characteristics have attracted researchers trained in a variety of fields like engineering, mathematics, physics, computer science, biology, psychology, and neuroscience. Current research efforts and applications of ANN include analyzing learning and self-organization algorithms, developing design principles and techniques to solve dynamic range and sensitivity problems in large analog system, implementing current algorithm using neuron-like components, building complete systems for pattern (including image and speech) recognition, establishing knowledge data base for stochastic information and knowledge retrieval, applying to robot control, implementing decision making support system, and so on [66][73].

1.1 Motivation

Visual cognition is one of the major functions of human's nervous system. It is estimated to be the most important aid for receiving information from outside world. The importance of human vision is a major inspiration for the impassioned interest and impressive research effort devoted to computer vision systems. As is true humans, vision

capabilities endow a robot with a sophisticated sensing mechanism that allows the machine to respond to its environment in an intelligent and flexible manner.

In the field of computer vision and pattern recognition, a number of research projects had been executed and numerous papers and textbooks have been written and published during the past decades. Even some special-purpose systems have been established, we are still far away from constructing a general-purpose image recognition system which has high performance and fault tolerance like human being.

One of the major problems is caused by the constraint of the machines used to implement systems for image understanding. Earlier vision systems were all implemented on conventional serial computer systems which inherit the limitation of "von Neuman bottleneck." A computer vision system must process an enormous amount of information, even for a simple task. Even the computer switching times are on the order of 10^{-8} seconds (10 nanoseconds), the overall operation times are very slow. Recent advances in parallel computer architectures [7][139] have made significant progresses in improving the processing speed of vision systems, but the complexity of parallel programming and system design becomes another problem in this field.

According to biological observations, a neuron requires on the order of one millisecond to fire, i.e. to send a stimulus to other neurons. Although this is relatively much slower than that of electronic computers, we cannot ignore the fact that biological systems succeed at complex tasks including vision and speech recognition tasks. To account for the computing power and intelligence of human brain, we are forced to rule out the speed of the neuron as the key. We may assert that the speed of human perception must be due to the brain's parallel architecture instead of the speed of individual processing elements.

Another problem is the requirement of a priori knowledge of the task domain. One knows what one is looking for although features observable in the image can be very

weak. Without such expectations image understanding is impossible. On conventional computer system or even on most of the recently developed parallel computers, the size of the storage required for memorizing the knowledge base will be increasing as long as the system is "learning" new image patterns and related knowledge.

In the field of artificial intelligence, a number of representation schemes have been developed for keeping not only information but also knowledge and for purposes of knowledge retrieval and reasoning. These schemes include logic and procedural representations, semantic networks, frame-based representation, model-based representations, case-based reasoning, and so on. All these schemes need a very large size of memory space to store the database or knowledge base. Thus, the limit of memory size becomes another problem of the implementation.

It is believed that the total size of biological memory is fixed in human brain. The learned knowledge is memorized in the form of connection strength. Newly learned pattern and knowledge can be memorized in the brain by changing the strength of all connection relative to this knowledge. The size of the brain does not have to be enlarged to learn more.

After comparing the major differences between current computer vision systems and human's visual cognitive process, we know that the nature of either approach is quite different to the other one. But, since the ultimate goal of these researches is to support making intelligent robots which can do whatever we can do, it seems to be a better approach if we can explore the mystery of information processing activation dominated by human nerve system and can replicate the same activation using electronic components. If we could have the same massive parallelism that we have in the brain, but with electronic computing elements instead of neurons, it seems that we should be able to obtain intelligent systems with 1,000 times the power of the brain (due to the ratio of processing speeds of two types of elements).

1.2 Review of Literatures

The history of the field can be traced back to the explanations of the brain and thinking processes suggested by some ancient Greek philosophers, such as Plato (427 - 347 B.C.) and Aristotle (384 - 322 B.C.). Physical views of mental processes were not held until 16th century by Descartes (1596 - 1650) and later in 18th century by some empiricist philosophers [66]. Practical examples of building neural computers, or called *cybernetic machines* earlier, in the 20th century include the "*Protozoon*" of Lux in 1920's, the "*dogs*" of Philips from 1920 to 1930, the "*Homeostat*" of Ashby from 1948, the "*Machina Speculatrix*" and "*Machina Docilis*" of Walter from 1950, the "ladybird" of Szeged from 1950, and many versions of a "mouse in the labyrinth" [66].

Analytical neural modeling has usually been studied in related to psychological theories and neuro-physiological research. McCulloch and Pitts presented the first sophisticated discussion of neuro-logical networks [76]. It is am attempt to understand what the nervous system might actually be doing, given primitive computing elements that are abstractions of the properties of neurons and their connections as they were known in the year. Five assumptions were given to describe the operations of binary neurons, so called "McCulloch-Pitts" neurons. It is obvious that the McCulloch-Pitts neuron is performing simple threshold logic. The "all-or-none" law of nervous activity is adequate to insure that the activity of any neuron may be represented as a proposition. The connection topology of the network relates to the complexity of propositions. Very complex propositions may be represented by connecting networks corresponding to simple propositions. This paper shows that any finite logical expression can be realized by McCulloch and Pitts neurons. Although the formal proofs and the notation in this early paper are exceptionally difficult, a later book, "*Computation*," written by Minsky [67] gives a much better introduction to McCulloch-Pitts neurons and also extends their works to later works in the theory of automata and computation.

Hebb attempted to base a large-scale theory of psychology on assumption about neural networks [51]. In his book, Hebb proposes his most famous and important idea called "Hebb" synapse or rule. A number of the best known neural network models are defined based on this idea: "When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change take place in one or both cells such that A's efficiency, as one of the cells firing B, is increased." That is, when two neurons are both activated, strength of the connection between these two neurons is increased at a certain level. Besides, Hebb raised the terminologies *distributed*, which is assuming to be the nature of representation used in the nervous system, and *cell assembly*, which defines the interconnected and self-reinforcing sub-modules forming the representation of information in the nervous system.

The first defined artificial neural network is called *perceptron* by Rosenblatt [93][94] and was extended and completed by Minsky and Papert [80]. Perceptron was defined as a learning machine that was potentially capable of complex adaptive behavior. Rosenblatt felt the appropriate way to view brain operation was as a learning associator, which couples classification responses to stimuli. He used as his basic architecture one layer of neurons projecting to another layer of neurons by way of parallel bundles of connections. The idea of reciprocal inhibitory connections was first introduced, which appears in many current neural network models as "winner-take-all" concept. There was no proof of the famous *perceptron convergence theorem* and were only a few statistical calculations indicating plausibility of learning, but overlooking the detailed limitations on learnability that proved so important later.

A later paper written by Block [10] in 1962 demonstrates a great increase in the level of analysis achieved in a few year, including mathematical analysis, computer simulation, and special purpose hardware. A "Mark I" perceptron had been built at Cornell by Rosenblatt and his coworkers. It contains 400 photosensitive receptors, on a 20×20 grid. This paper contains a concise proof of perceptron convergence theorem.

Unfortunately, a number of important classifications are not linearly separable. This led to a loss of interest in perceptrons and related devices because simple perceptrons could not compute a number of important quantities that a brain was clearly able to do [80]. In their book "*Perceptrons*," Minsky and Papert gave discussions on mathematics and the theory of computation. They also placed the study of computational limitations of perceptrons on a solid foundation. Two important limitation classes were used in the proof: order limited and diameter limited. They demonstrated that perceptrons (and all other neural networks) are too limited to deserve further attention. This made a delay of researches and has caused very few discoveries in this field until 1980's.

During the same decade, the convergence theorem for learning machine was well discussed by Nilsson [85]. He documented the thorough study of feedforward networks with one layer of modifiable weights connecting input units to output units, so called threshold logic units. The limits to how much computation can be accomplished by such networks are also delineated in his book. Just as more complex creatures evolved in recent years and now have achieved vastly greater capabilities than percetrons by making use of multilayered architectures with feedback connections. Nevertheless, recent work could not have been accomplished without building on these foundations.

An earlier proposed and implemented technique which is known as *Least Mean Squares* (LMS) algorithm has been applied to *ADAptive LInear NEuron* (ADALINE) since Widrow and Hoff proposed their first paper regarding with their idea [134]. The adaptive system they described in the paper could potentially learn quickly and accurately. It is composed of threshold logic units and variables connection strengths. They assumed that an error signal can be formed to indicate the difference between what the output is supposed to be and what the summation the neuron computes. The connection strengths will be adjusted and then the error signal is generated again and the procedure will repeat until the error signal became zero or an acceptable level. This reduces the square of the error signal to its smallest possible value since minimizing

square error means that there is a simple quadratic error surface with only one global minimum. (In fact, according to the paper by Rumelhart *et al* [95][96][97], back propagation learning is a generalized gradient method and is a generalization of the simple Widrow-Hoff learning rule.) In two recent papers [135][136], Widrow *et al* gave a more formal definition of ADALINE along with complete mathematical analysis and practical applications.

Grossberg has been one of the most visible scientists working in neural networks for nearly twenty years. In his paper [46], a theoretical analysis inspired by the developmental physiology of cortical organization. He mentioned that there is strong evidence for the development and modification of feature detectors in visual cortex in response to a particular environment. The experimental results suggest that cortex tunes itself to pick up the most useful features of the environment and ignores or suppresses other information.

In a later paper [47], Grossberg developed a series of detailed mechanisms in this paper, but most of them are first used in, and arise from, a consideration of error correction. His key point is that the neural network must generally do error correction by itself, without outside help. This places strong constraints on the way the system can be wired up. He applied the mechanisms he described to a number of different systems in the brain. It is typical of Grossberg's work to bring together in one theory several large scale organizing principles, their effects, and their interactions. A series papers regarding with his model, called *Adaptive Resonance Theory* (ART), have been published during the past few years and some of these papers provide theoretical proof to the stability and convergence characteristics [12][13][14][15][49].

Hopfield assumes that the basic elements of the network are threshold logic unit, which sums synaptic inputs, compare the sum with a threshold, and then response 1 if the sum is at or above threshold or 0 otherwise [52]. A new term, *energy*, was first

introduced, which is expected to reach a minimum. A randomly-chosen neuron in the network looks at its inputs, and changes state, depending on whether or not the sum of its input is about or below threshold. It can be seen from the form of the energy term that a state change leads either to a decrease in energy or to the energy remaining the same. The learning rule is based on Hebb rule and, therefore, is an energy minimizing rule which continues modification activities until a stable state is reached. In a later paper [53], he discusses networks of neurons that can show graded intermediate states. The network is recurrent, in that the neurons connect to each other, with the exception that a neuron does not connect to itself. The threshold logic output function was replaced by a sigmoidal nonlinearity. Hopfield demonstrated that his model is identical to a set of operational amplifiers connected together.

Fukushima had proposed earlier [35] a multilayer neural network model with strong self-organizing properties, which he called the "*cognitron.*" A later paper [31], then, described a modification of the cognitron -- called "neocognitron." The model was built up for the application of handwritten character recognition. The system is considerably constrained because of the nature of the inputs, that is, the inputs could only be configurations of lines of varying degrees of complexity. Subsequent papers [37][41][42] had shown that the theoretical background, analyses, and characteristics of the neocognitron along with the experimental results. It is shown that the neocognitron is scaling and translation invariant and can even recognize noisy and/or distorted hand-written characters.

1.3 Subject Outline

This dissertation is divided into eight chapters.

- Chapter 2 reviews the theoretical foundations of artificial neural networks.

- Chapter 3 presents a new ART-based neural architecture for pattern classification and image enhancement.
- Chapter 4 presents an implementation of morphological operations using neural network architectures.
- Chapter 5 presents an investigation of the use of multilayer perceptrons for image processing. An architecture consisting of multilayered perceptron and recurrent connections is designed and is trained to carry out classification rules in image transformation.
- Chapter 6 presents several one-pass training algorithms for variant morphological operations.
- Chapter 7 proposes a morphological neural architecture to construct the feature extraction module of a personal handwritten character recognition system. The system was trained and tested with scanned image of handwritten characters. The feasibility and efficiency are discussed along with the experimental results.
- Chapter 8 summaries the contribution of this dissertation and makes the conclusion.

CHAPTER 2

ARTIFICIAL NEURAL NETWORKS

ANNs are massively parallel interconnected networks of simple and usually adaptive processing elements and are characterized by their hierarchical organizations which are intended to interact with the objects of the real world in the same way as biological nervous systems do. The behavior of the ANNs depends heavily on the connection details. The state of the ANN evolves continually with time. ANNs are considered clever and intuitive because they learn by example rather than by following programmed rules. With experience ANNs become sensitive to subtle relationships in the environment which are not obvious to humans.

2.1 Neurons

In principle, ANNs are composed of a large number of processing elements with massive connections among neurons. The basic processing operation performed by every processing element is an analog operation or *transformation* of its input signals. The basic components of ANNs are mostly called "*neurons*" which are processing elements forms a weighted summation of N inputs and passes the results through a nonlinearity. Fig. 2.1 shows a typical neuron and the mathematical function of the output from the inputs.

The circle with a $\sum$ is the part for weighted summation computation and determines the activation state of the neuron. The inputs are indicated with $x_i, i = 1, \cdots, N$ and the weights with respect to the inputs are represented by $w_i, i = 1, \cdots, N$. The bias weight w_0 is connected to a constant input, x_0, and it controls the threshold level. The most

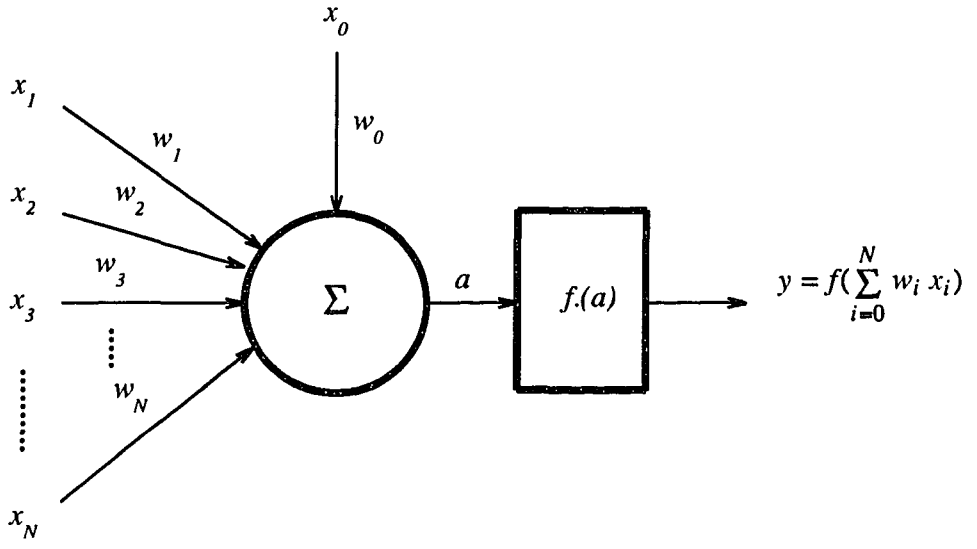
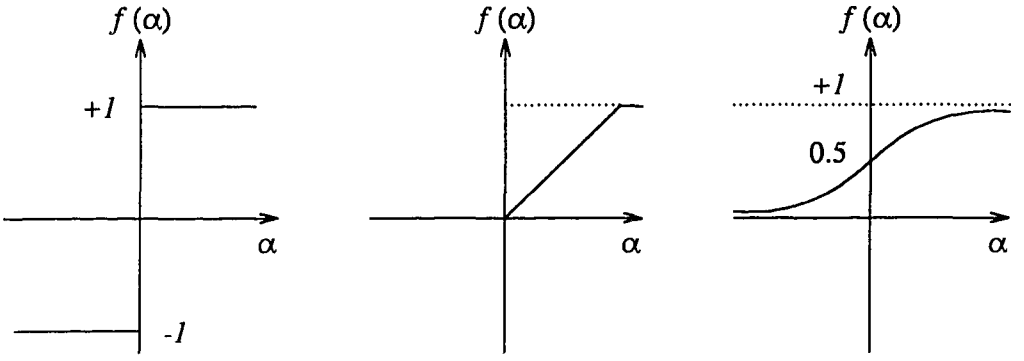


Fig. 2.1 A typical neuron. Each input, $x_i, i = 0, \cdots, N$, is multiplied with a connection weight, $w_i, i = 0, \cdots, N$, and then the neuron sums up all the weighted input to determine the state of activation, a. The output function $f.(a)$ determines the output value of the neuron.

common expression of the activation state is $a = \sum_{i=0}^{N} w_i x_i$. More complex neurons may include temporal integration or other types of time dependencies and more complex mathematical operations.

The output of a neuron is a function of the activation state, a. The square with an $f.(a)$ in Fig. 2.1 is the part determining the output with a nonlinearity defined by $f.(a)$ where "." is the type of nonlinearity. Fig. 2.2 shows three representative nonlinearity: *hard limiter*, *threshold logic*, and *sigmoid* functions [Lip87].

Hard limiter nonlinearity is usually applied on ANN models for binary information processing because of its binary-valued outputs, while sigmoid nonlinearity is suitable for analog signal processing. The threshold logic nonlinearity are widely used in both binary and analog information processing. The neurons in an ANN may be assigned with



$$f_h(\alpha) = \begin{cases} +1 & \text{if } \alpha > 0 \\ -1 & \text{if } \alpha < 0 \end{cases} \qquad f_l(\alpha) = \begin{cases} \alpha & \text{if } \alpha > 0 \\ 0 & \text{if } \alpha < 0 \end{cases} \qquad f_s(\alpha) = \frac{1}{1+e^{-\alpha}}$$

Fig. 2.2 Three representative transfer functions for determining the output value of a neuron.

the same or different output function depended on the network configuration. Sometimes the function is assumed to be a stochastic function in which the output of the neurons depends in a probabilistic fashion on its activation values.

2.2 Connection Topology

There are a number of ANN models proposed and the patterns of connectivity between neurons vary across models. A neuron can be *locally connected* to neighbors, *fully connected* to every other neurons, or *sparsely connected* to a few distant neurons. Neural networks may be layered with exclusively *feed-forward* connections from lower to higher layers, or provided with recurrent *feed-back* connections from higher to lower layers or to the same layers. The effect of feed-forward connections can be viewed as a support to the concept represented by the receiving neuron, while that of the other type is considered as disagreement to the concept.

Associated with each connection is the strength of that connection, mostly called *weight*. There are two types of connections: excitatory and inhibitory. Inhibitory connections, usually represented by a negative weights, tend to prevent firing of the neuron and excitatory connections, positive weights, tends to cause firing of the neuron. A special type of inhibitory connection, called *lateral inhibition*, may connect from one neuron to the rest of the neurons in the same layer (in layered networks). This type of connections are usually located in the output layer and may make only one of the output neurons be activated. This effect of minimizing the number of active neurons is one type of competition. A neuron may receive inputs of different kinds including both excitatory and inhibitory connections, i.e. complex inhibition/excitation combination rules may be required. Fig. 2.3 shows a simple ANN structure with feedback connection and competition.

For example, multi-layer perceptrons and neocognitrons have only feed-forward connections between layers; ART model has both feed-forward and feed-back connections; and the Hopfield model has only lateral inhibitory connection among the neurons in only a single layer. Feed-forward networks have to operate only until the inputs propagate to the output layer which guaranteed stability, while networks with recurrent connections must iterate over cycles to produce an output which may be unstable for certain conditions. An unstable network may have either oscillating outputs or locked-up outputs.

2.3 Learning Paradigms

ANNs can be trained using two types of training procedures:

- *Supervised training* requires labeled training data and an external teacher. The teacher provides the desired/correct response and a feedback error signal after each trial. This teacher only indicates whether a response was correct or incorrect and

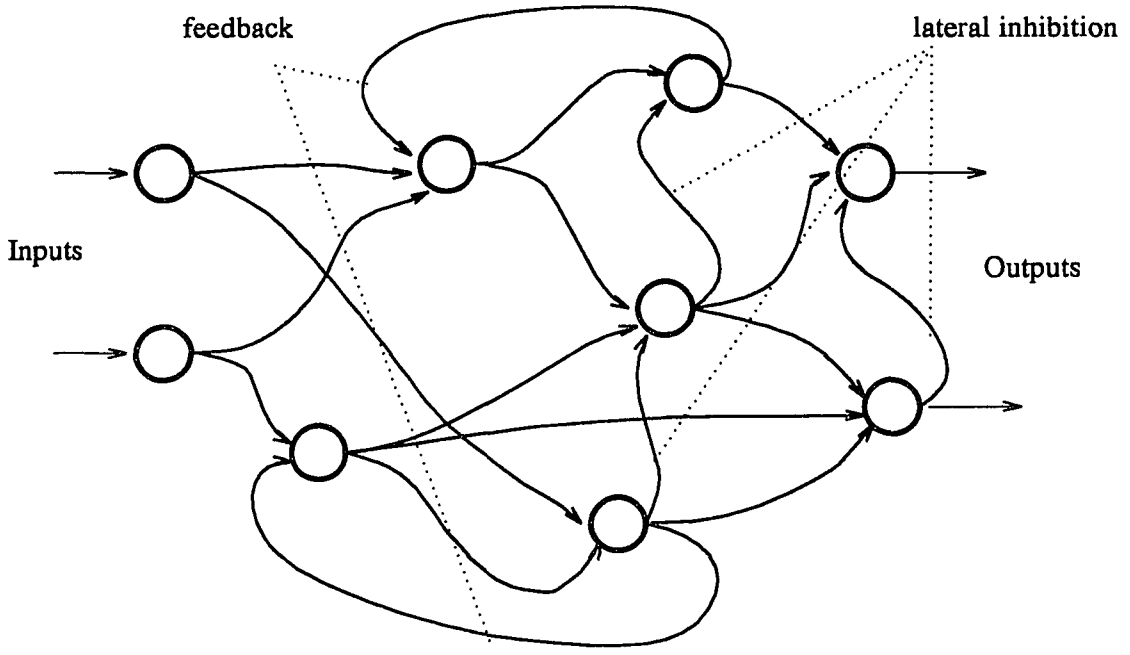


Fig. 2.3 A simple neural network with feed-forward connections (solid lines), feedback connections (dashed lines), and lateral inhibitory connections (dotted lines).

does not provide detailed error information. This is sometimes called *reinforcement learning* or *learning with a teacher*. This group includes rules based on delta rule by Rumelhart *et al.* [RuH86a] and least-mean-square rule by Widrow [WiH60, WiW88a, WiW88b].

- *Unsupervised training*, also called *self-organization*, uses unlabeled training data and required no external teacher. Data is presented and internal categories or clusters are formed which compress the amount of input data that must be processed at higher levels without losing important information. This group includes Hebbian learning rule [Heb49] and competitive learning rule.

The major distinction between supervised and unsupervised learning depends on information: the former uses pattern-class information and the latter does not.

2.4 Taxonomy of ANN Models

A taxonomy of important ANN models that can be used to classify and cluster static patterns is shown in Fig. 2.4. The taxonomy is first divided between models with binary- and continuous-valued inputs. Below this, ANN models are divided between those trained with and without supervision (teachers). ANN models trained with supervision include perceptrons and multi-layer perceptrons described in chapter 1.

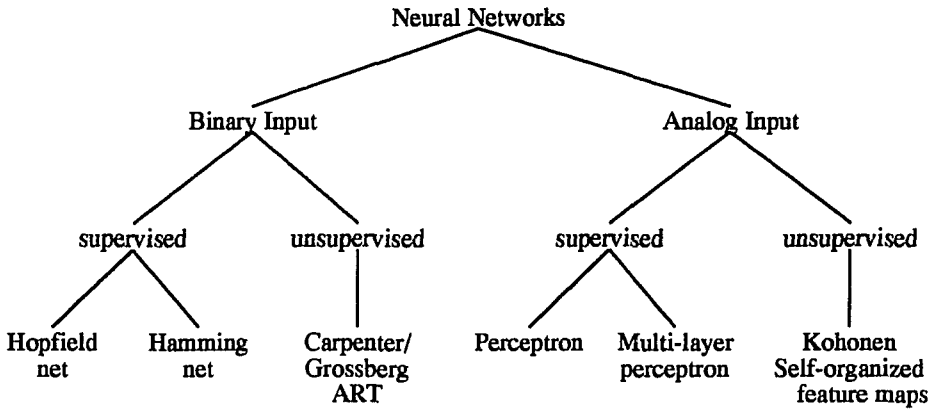


Fig. 2.4 A typical taxonomy of ANN models [DAR88].

These models are provided with labels that specify the correct class for new input patterns during training. Most traditional statistical classifiers, such as Gaussian classifiers, are trained with supervision using labeled training data [DuH73]. Models trained without supervision, such as the Kohonen's feature map forming networks [Koh88], are used as vector quantizers or to form clusters. No information concerning the correct class is provided to these networks during training. A further difference

between ANN models, not indicated in Fig. 2.4, is the connection topology of the models. All models in this taxonomy — except the Hopfield and ART — use predominately feed-forward connections. At the bottom of Fig. 2.4, classical algorithms which are most similar to or perform the same function as the corresponding ANN models are also provided for reference. In some cases, an ANN model implements a classical algorithm exactly. In others, the ANN algorithms represent new classification algorithms that extend the theory of pattern classification.

2.5 Characteristics of ANN Models

ANN models have many terms that describe their behavior and abilities. *Adaptability* is the ability to modify a response to changing conditions. There are four processes produce this ability: *learning*, *self-organization*, *generalization*, and *training*. *Plasticity* is the ability of a group of neurons to adapt their functions to different needs over time. When a portion of the network is damaged, other neurons adapt to take over the functions that the damaged portion once performed. *Self-organization* is when a network modifies all its neurons at once according to a learning rule. This is usually done by modifying the individual synaptic weights, in response to changes in the inputs. *Generalization* is the ability of a neural network to formulate an answer to a problem it never saw before by using related or similar information.

Dynamic stability is the ability of a network to remain within its functional boundaries and reach a stable state. Such a network can be given some extreme value of data and it will still manage to return to a stable state. *Convergence* is the changing state of a network as it moves toward a stable state. *Fault tolerance* is the ability to keep processing, though with reduced accuracy and/or speed, when a small number of neurons are disabled or destroyed. *Normalization* is an adjustment which keeps weights within a prescribed range of acceptable values.

The ANN network approach exhibits two major characteristics in the fields of problem solving. First, neural network architectures are parallel and connectionist. The connections between pairs of neurons play a primary role in the function performed by the whole system. The second characteristic is that the prescriptive computer programming routine is replaced by heuristic learning processes in establishing a system to solve a particular problem. It means that a neural network is provided with representative data, also called *training data*, and some learning process which induces the data regularities. The modification of the transfer function at each neuron is derived empirically according to the training data set in order to maximize the goal.

Two operating phases, learning and execution, are always encountered in neural networks. During the *learning phase*, the networks are programmed to take input from a training set and adjust the connection weights to achieve the desired association or classification. During the *execution phase*, the neural networks are to process input data from the outside world to classify into the corresponding outputs.

The characteristics of biological neural networks that ANN models hope to provide include:

- *Fault tolerance* to loss of a small number of computational elements,
- *Insensitivity* to small variations between computational elements,
- The need for primarily *local connectivity* and *local learning rules*,
- *Real-time response*, and
- *Parallelism*.

2.6 Applications of ANN Models

Researchers developing computational models tend to focus on one or more of the following common problem areas:

- Designing ANN architectures to implement important conventional architectures.
- Developing new highly parallel neural network algorithms.
- Analyzing algorithms using simulation and theory.
- Implementing algorithm in hardware.

Most researchers focus on ANNs that perform those seven major tasks illustrated below.

- *Classification.* Classifier are trained with supervision using labeled training data to partition input patterns into a pre-specified number of groups or classes. These could represent different words for a speech recognizer or different objects for an visual image classifier. The inputs may be either binary or continuous values.
- *Self-organization/Category Formation.* Self-organizing networks partition input examples into groups or clusters using unlabeled training data. This type of clustering is an efficient technique for reducing information that must be processed at higher levels with little loss in performance. It also makes good use of the large amount of unlabeled training data that is typically available in speech and vision problems. The number of clusters formed may be pre-specified or determined from the examples.
- *Associative memory.* An associative, or called *content-addressable memory*, provides a complete memory item from a key consisting of a partial or corrupted version of the memory. For example, it might return a complete object image from some key features or a complete article citation from only the author's name or a complete image of a face from only the bottom half.

- *Sensory Data Processing.* An enormous amount of real-time preprocessing is performed in the peripheral sensory vision and hearing centers. ANNs can perform this function in real time using massive parallelism.
- *Computational Problems.* Custom ANN architectures can be designed to solve specific computation problems, such as the traveling salesman problem and other constrained optimization problems, using nonlinear analog computation.
- *Nonlinear mapping.* Many ANNs can map a vector of analog inputs into an output vector using a nonlinear mapping function which can be learned from training data. These types of mappings are useful in many areas, including robot control and nonlinear signal processing.
- *Multi-sensor automata.* A number of complex, multi-module neural network have been built with visual input and a robot arm to manipulate objects in an environment.

CHAPTER 3

A NEW ART-BASED NEURAL ARCHITECTURE FOR PATTERN CLASSIFICATION AND IMAGE ENHANCEMENT WITHOUT PRIOR KNOWLEDGE

Adaptive Resonance Theory, abbreviated ART, is one of the most famous ANN models and was developed by Carpenter and Grossberg [12][13][15]. This model, which forms clusters and is trained without a supervisor, can achieve the self-organization of input codes for pattern classification in response to random sequences of input patterns. With the concept of such an architecture, the process of adaptive pattern recognition is a special case of the more general cognitive process of hypothesis discovery, testing, searching, classification, and learning. This property makes it feasible to apply in more general problems of adaptively processing a big set of information sources and databases.

The visual field is not perceived as an array of independent picture points. Instead, it is usually seen as consisting of a relatively small number of patterns. The *Gestalt laws of organization* describe these groupings or patterns which are most naturally seen as units. By the *law of good continuation*, when curves cross or branch, parts that smoothly continue one another are seen as belonging together. The grouping that can be specified using the least information is the preferred approach. Here the specification might be formulated in terms of the number of lines, curves, line crossings, etc., in the grouping. When the given picture is two-valued (''black and white''), we can delete noise points and fill in gaps by introducing the features of template subpatterns from which they differ.

In practice, it is very difficult to enhance an image which has been recorded in the presence of one or more sources of degradation, in a single-pass filtering. Noise must be first eliminated and the edges have to be preserved. Several passes may be needed to

clarify the perceived subpatterns; that is similar to the activation of a human's visual system. When one sees a picture, his brain will request the visual system to focus on certain subpatterns which are the most interesting. Humans can identify object edges and regions with exceptional accuracy seemingly instantaneously while conventional computers are not always nearly as accurate or as fast. The human brain is a massively parallel computing machine whereas traditional computers are not. Neural networks are a parallel computing architecture that can therefore more closely emulate the human brain. For the purposes of pattern classification and image enhancement, the neural networks may partially bridge the gap in performance and speed between the human brain and the machine.

In this chapter we first briefly review the ART model and its underlying processes. Second, an improved ART model is presented and applied to optical character classification. Third, a new neural architecture is proposed, which is devised whereby the pre-established classification templates are used for image enhancement, and the experimental results are provided. (This chapter has been published as a journal paper [112] and related topics can be referenced in [111] and [114].)

3.1 Background of the ART Model

The ART model grew out of the analysis of a simple type of adaptive pattern recognition network, often called the *competitive learning* paradigm [14]. Such a combination of adaptive filtering and competition is shared by many other models of pattern recognition and associative learning [100]. There are three classes of ART architectures: ART1 [12] for classifying binary patterns, ART2 [13] for analog patterns, and ART3 [15] for parallel search of learned pattern recognition codes.

The architecture of a typical ART 1 model is shown in Fig. 3.1. The ART1 model contains two parts: the *attentional* subsystem and the *orienting* subsystem. The

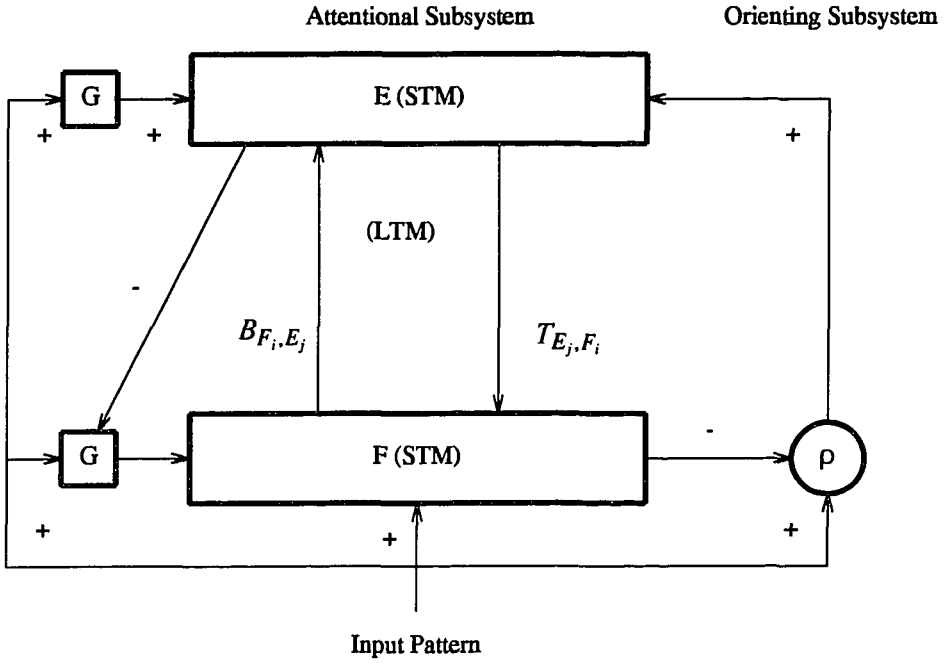


Fig. 3.1 A typical ART 1 architecture. Rectangles represent the attentional subsystem containing two fields where short term memory (STM) patterns are stored. Squares with the label 'G' indicate the gain control devices. The orienting subsystem with vigilance parameter ρ is represented by the circle.

attentional subsystem is composed of two layers: the *feature representation* field (*F*) and the *exemplar* (or *category*) *representation* field (*E*). Neurons in the two fields are fully connected with each other and are accompanied with adaptive *weights* analogous to synapses in animal nervous systems. The network self-adjusts the weights independently without a supervisor. The connections are bi-directional and their weights constitute the *long-term memory* (LTM) traces which are adjusted by input patterns only during training. The activation of neurons in both layers forms the *short-term memory* (STM) traces which are adjusted during both training and classification.

The stimuli of input patterns lead to the activities of neurons in *F*. The signals pass through the connections to neurons in *E* and are multiplied by their corresponding weights. Each neuron in *E* sums up all of those weighted activities. The layer *E* is

designed as a competitive network where each neuron has lateral inhibitory connections to the others. This is capable of choosing the neuron which receives the largest total input, i.e., a *winner-take-all* strategy is used. The winning neuron represents a category and triggers its associative pattern through the connections to F. This is very useful for applications with categorical perception — that is, classifying each input pattern as belonging under one and only one category.

A matching threshold called the *vigilance parameter* (ranging from 0.0 to 1.0) is introduced to determine how closely an input pattern is matched against each category. The vigilance testing is performed by the orienting subsystem which receives two inputs: one is the input pattern with excitatory connection and the other is the overall activity in F with inhibitory connection. If the vigilance testing is satisfied, the input is classified into the category corresponding to the winning neuron. Otherwise, the subsystem sends a *reset* signal to the winning neuron to disable its output temporarily so that the subsystem may consider the second winning neuron. Such a task is repeated until a category is chosen or no more neurons are available; in this case a new neuron is created to represent this input as a new category.

3.2 Improvement of ART Model

In implementing the algorithm of Carpenter and Grossberg by software simulation dealing with optical character recognition, we obtained the same result of category classification as described in [12][17][122]. However, we have observed that the input patterns are classified with variance to their training sequences. For instance, if the sequence is {"E", "F", "L"}, "F" is classified as the same category with the exemplar "E" and replaces "E" as a new exemplar, and then "L" is classified as another new category. If the sequence is {"E", "L", "F"}, "L" belongs to the category with the exemplar "E" and replaces it as a new exemplar, and then "F" is classified as another new category. If the sequence is {"F", "L", "E"}, then all three

characters are in different categories. Even though the vigilance parameter is adjusted, the variation of categories still exists.

In this section, we will first give the problem analysis and then propose an improved model and illustrate the experimental results.

3.2.1 Problem Analysis

In the ART model, each neuron in the higher layer E sums up all the weighted activities from neurons in the lower layer F. The activation equation is

$$\mu_{E_j}^{(t)} = \sum_i B_{F_i,E_j}^{(t)} \cdot x_i, \tag{3.1}$$

where $B_{F_i,E_j}^{(t)}$ denotes weights of bottom-up connections from neuron F_i to neuron E_j at time t and x_i is the i-th component of the input pattern. When a winning neuron E_{j*} is chosen based on the maximum activation, the orienting subsystem activates for the vigilance test, which can be expressed as:

$$\frac{\sum_i T_{E_{j^*},F_i}^{(t)} \cdot x_i}{\|\mathbf{X}\|} \geq \rho, \tag{3.2}$$

where $T_{E_{j^*},F_i}^{(t)}$ denotes weights of top-down connections from the winning neuron E_{j*} to neuron F_i at time t, and $\|\mathbf{X}\|$ is the norm (or magnitude) of the input pattern vector, i.e., $\|\mathbf{X}\| = \sum_i |x_i|$. In Eq. (3.2) for vigilance testing, the numerator picks up all common pixels between the input pattern and the exemplar it's comparing with, and the denominator is the norm of the input vector. This ratio represents the percentage of the pixels of the input pattern existing in the exemplar. Since "**F**" is a subset of "**E**", the ratio is 1.0, i.e., all the pixels of "**F**" are included in "**E**". But, how can we take into account the pixels of "**E**" which are not in "**F**" (the line at the bottom)? The orienting

subsystem is supposed to be able to count not only the pixels in "**F**" and not in "**E**" but also the pixels in "**E**" and not in "**F**" in order to give a fair judgement of the similarity.

Another issue is that for each new training pattern, the network repeats choosing the winning neuron and performing vigilance testing until a category is chosen. Whereas, if none of the existing neurons in E satisfies the vigilance test, a new neuron is created to represent the new category using the pattern as an exemplar. This procedure is time-consuming and the performance is the worst when a new input pattern is quite different from all the existing exemplars. To reduce the searching time of the model, a minimum activation response may be set up so that the model need not go through the neurons with the lower response.

3.2.2 An Improved ART Model for Pattern Classification

The task of vigilance testing is now modified. We use a bi-directional testing to determine the similarity between an input pattern and the exemplar of a chosen category. In addition to Eq. (3.2), another inequality Eq. (3.3) is introduced to determine how similar an exemplar is to the input pattern. That is

$$\frac{\sum_i T^{(t)}_{E_{j^*},F_i} \cdot x_i}{\|\mathbf{T}\|} \geq \rho, \tag{3.3}$$

where $\|\mathbf{T}\|$ is norm (or magnitude) of the exemplar vector, i.e., sum of top-down weights from the winning neuron E_{j^*} to all the neurons in F. The difference between Eq. (3.2) and Eq. (3.3) is the denominator; one is the norm of the input vector and the other is the norm of the exemplar vector. The ratio represents the percentage of the pixels of the exemplar existing in the input pattern. Only if these two equations are satisfied, is the new pattern associated and combined with the exemplar; otherwise, the next exemplar is chosen to compare with. It has to be noted that the vigilance parameters in eqs. (2) and

(3) could be selected differently. Therefore, an improved ART model shown in Fig. 3.2 includes a set of new devices which accumulates the total weights of top-down connections of an active neuron E_j and sends them to the orienting subsystem.

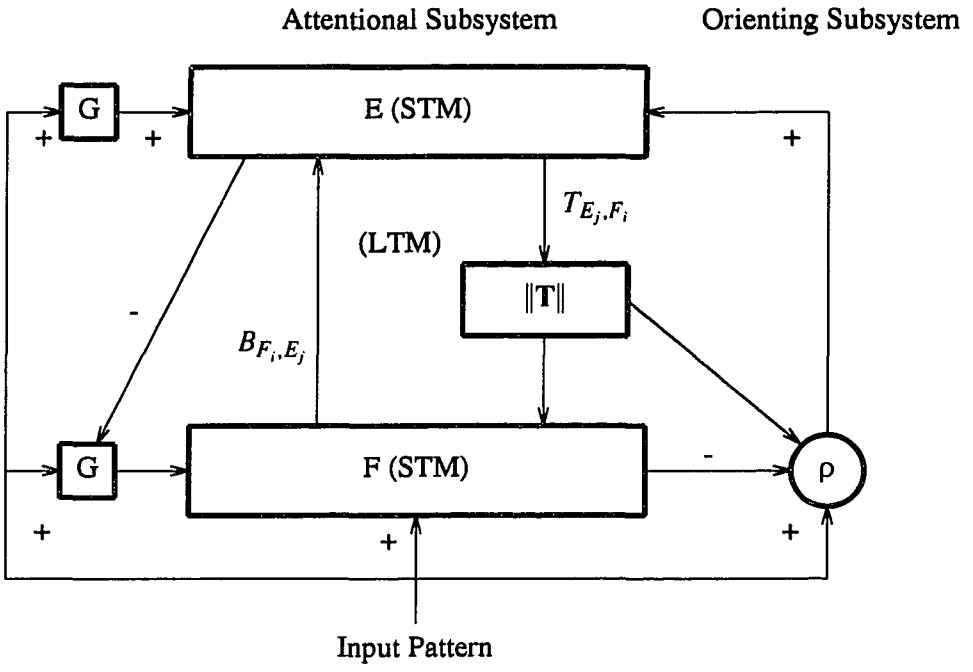


Fig. 3.2 An improved ART model. The activation in layers F and E forms a short term memory (STM) and the connections including bottom-up and top-down construct a long term memory (LTM). Squares with the label "G" indicate the gain control devices and the orienting subsystem with vigilance parameter ρ is represented by a circle.

The algorithm of the improved ART model is described as follows:

Step 1 — Initialize weights

$$T_{E_j,F_i}^{(0)} = 1.0, \tag{3.4}$$

$$B_{F_i,E_j}^{(0)} = \frac{1}{1+N}, \tag{3.5}$$

where $1 \le i \le N$ and $1 \le j \le M$ (N: number of elements in the input vector; M: number of existing exemplars). Also the vigilance parameter ρ, where $0.0 \le \rho \le 1.0$, is selected.

Step 2 — Read input pattern X

Step 3 — Compute neurons' responses

The output of the neuron E_j is computed by

$$\mu_{E_j}^{(t)} = \sum_{i=1}^{N} B_{F_i,E_j}^{(t)} \cdot x_i, \tag{3.6}$$

where $B_{F_i,E_j}^{(t)}$ denotes weights of bottom-up connections from F_i to E_j at time t, and x_i is the i-th component of the input pattern vector **X** which can be either 0 or 1 (binary).

Step 4 — Select a winning neuron with the highest response

Let C be the total number of currently used exemplar neurons.

$$\mu_{E_{j^*}}^{(t)} = \max\{\mu_{E_j}^{(t)} \mid 1 \le j \le C \text{ and } j = C+1 \text{ (the first available unused neuron)}\}, \tag{3.7}$$

where E_{j^*} is the winning neuron.

Step 5 — Bidirectional vigilance

Let $\|\mathbf{X}\| = \sum_{i=1}^{N} x_i$, $\|\mathbf{T}\| = \sum_{i=1}^{N} T_{E_{j^*},F_i}$, and $\|\mathbf{T}\cdot\mathbf{X}\| = \sum_{i=1}^{N} T_{E_{j^*},F_i} \cdot x_i$.

If $\frac{\|\mathbf{T}\cdot\mathbf{X}\|}{\|\mathbf{X}\|} \geq \rho$ and $\frac{\|\mathbf{T}\cdot\mathbf{X}\|}{\|\mathbf{T}\|} \geq \rho$, go to step 7; otherwise, go to step 6.

Step 6 — Disable unmatched category

Disable temporarily the output of the winning neuron chosen and go to step 4 for choosing the next winning neuron.

Step 7 — Adjust weights of the winning neuron

$$T_{E_{j^*},F_i}^{(t+1)} = T_{E_{j^*},F_i}^{(t)} \cdot x_i \tag{3.8}$$

$$B_{E_{j^*},F_i}^{(t+1)} = \frac{T_{E_{j^*},F_i}^{(t)} \cdot x_i}{0.5 + \sum_{i=0}^{N-1} T_{E_{j^*},F_i}^{(t)} \cdot x_i}. \tag{3.9}$$

Step 8 — Enable all neurons and repeat the whole procedure by going to step 2.

The intention of adjusting the weights in the top-down connections is to store the information with respect to the exemplar of each category. Eq. (8) shows that the new weight is derived from the intersection of the existing exemplar and the input pattern. Since the top-down weights are initialized to be all ones, the first classified exemplar is always the same as the input pattern; the updated exemplar will be the structure of the common pixels of all the patterns in each category. This step will ensure that the classified categories are independent of the input sequences.

Eq. (3.5) initializes the weights of all bottom-up connections to be $\frac{1}{1+N}$, where N

is the number of elements in **X**. Thus, when an **X** is presented, the magnitude of any unused neuron's response will be $\frac{\|\mathbf{X}\|}{1+N}$. Referring to eqs. (2) and (3), a perfectly matched pattern will cause the neuron to respond 1.0 and a pattern with totally unmatched pixels will respond with the value 0.0. The higher the response value is, the more matched pixels the pattern has. Therefore, the searching procedure may skip checking the neurons with responses lower than those of unused neurons. In step 4 of the algorithm, the searching always chooses the maximum of all active E neurons plus the first available unused neuron. If all active neurons with higher response than the unused neuron could not pass the vigilance testing, the unused neuron is assigned to represent the new created category. This can save significant time in searching most of the unmatched categories.

3.2.3 Experimental Results of the Improved Model

The input patterns used in this experiment are 16×16 optical characters. Therefore, a 256-neuron layer is designed for F and a 26-neuron layer for E. The vigilance parameter is set to be 0.8. Fig. 3.3 shows the result of character classification and the exemplar in each category.

While the character "**F**" is trained, the existing exemplars already have "**A**", "**B**", "**C**", "**D**", and "**E**", and the exemplar "**E**" responds with the highest value to be the winning neuron. Thus, the vigilance testing is:

$$\frac{\|\mathbf{T}\cdot\mathbf{X}\|}{\|\mathbf{X}\|} = 1.00 > \rho = 0.8 \quad \text{and} \quad \frac{\|\mathbf{T}\cdot\mathbf{X}\|}{\|\mathbf{T}\|} = 0.72 < \rho = 0.8.$$

Since the pattern "**F**" is a subset of the pattern "**E**" (for some certain fonts), the first ratio having value 1.00 signifies that the testing passes. But on the other hand, "**E**" has more pixels than "**F**" has. This means that the second testing fails. The second highest responding neuron, i.e., the category "**B**", is next chosen and the vigilance testing still

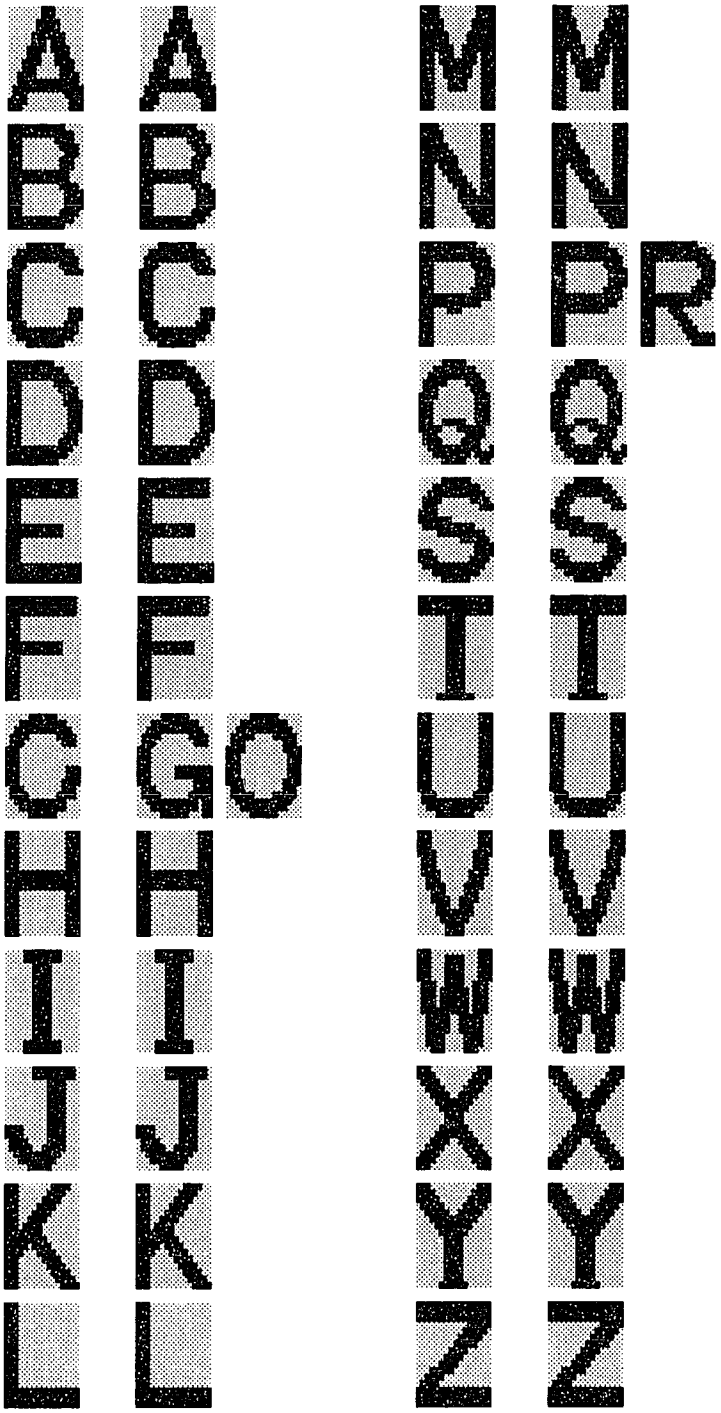


Fig. 3.3 Result of character classification and the exemplar in each category.

fails. Thus, the system creates a new node in the exemplar field to represent the category "**F**".

With $\rho = 0.8$, the two pairs of characters {"**G**", "**O**"} and {"**P**", "**R**"} are classified into the same category, respectively. To achieve 100% classification rate, the higher vigilance parameter, e.g. 0.9, is assigned. In addition, certain noisy patterns are experimented to test the system performance. The weights of bottom-up and top-down connections are fixed. As shown in Fig. 3.4, these noisy patterns can be recognized and classified into their proper categories.

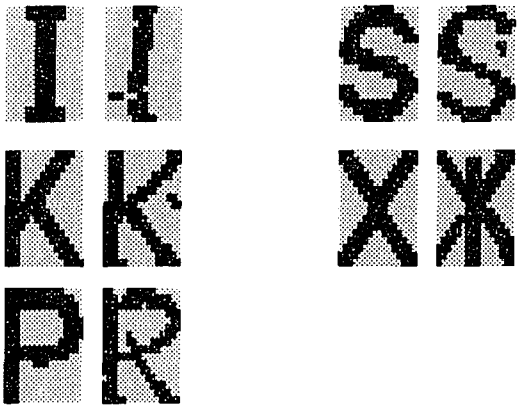


Fig. 3.4 Testing patterns with noise and variances are also properly recognized and classified.

3.3 A New Neural Architecture for Image Enhancement

Whenever a picture is converted from one form to another, e.g., imaged, copied, scanned, transmitted, or displayed, the "quality" of the output picture may be lower than that of the input. In the absence of knowledge about how the given picture was actually degraded, it is difficult to predict in advance how effective a particular enhancement method will be. In this section, an elementary class of image enhancement method adopting the use of neural architecture will be discussed. In general, we can use

enhancement techniques to suppress selected features of a picture or to emphasize such features. From this viewpoint, enhancement can be regarded as selective emphasis and suppression of information in the picture, with the aim of increasing the picture's usefulness.

3.3.1 Overall Architecture

In Fig. 3.5(a) we show an improved ART module to be used in constructing a neural architecture for image enhancement. The overall neural architecture shown in Fig. 3.5(b) includes four layers. The first two layers which are the improved ART-based model described in Section 3 intend to determine whether or not the input pattern is matched with the exemplars of certain features previously learned and stored. The first layer consists of 25 neurons F_i ($1 \leq i \leq 25$) since the local 5×5 neighborhood is considered. The second layer contains M exemplar neurons E_j ($1 \leq j \leq$ M), where M is the total number of features necessary for image enhancement. The third layer, or *region detection* layer, made up of 5 neurons D_k ($1 \leq k \leq 5$) corresponding to the pixel P and its four neighbors, is used to determine whether P should be illuminated or not. Each region detection neuron receives and sums up the responses from all the exemplar neurons in the second layer. If any one of the exemplar neurons has a positive response, indicating that P is classified into an exemplar category and should be illuminated as part of the object region, then the corresponding detection neuron will output "1". The fourth layer contains only one neuron corresponding to the pixel P. It has the input connections from five neurons related to the pixel P and its four neighbors so that a simple binary edge detection can be performed. If any one of five neurons has the value "0", indicating that P is surrounded by the background pixel, then the output will be set ON, otherwise OFF.

The total number of neurons required in the neural architecture is computed as follows. Suppose N neighbors are considered for feature extraction (In Fig. 3.5, N =25)

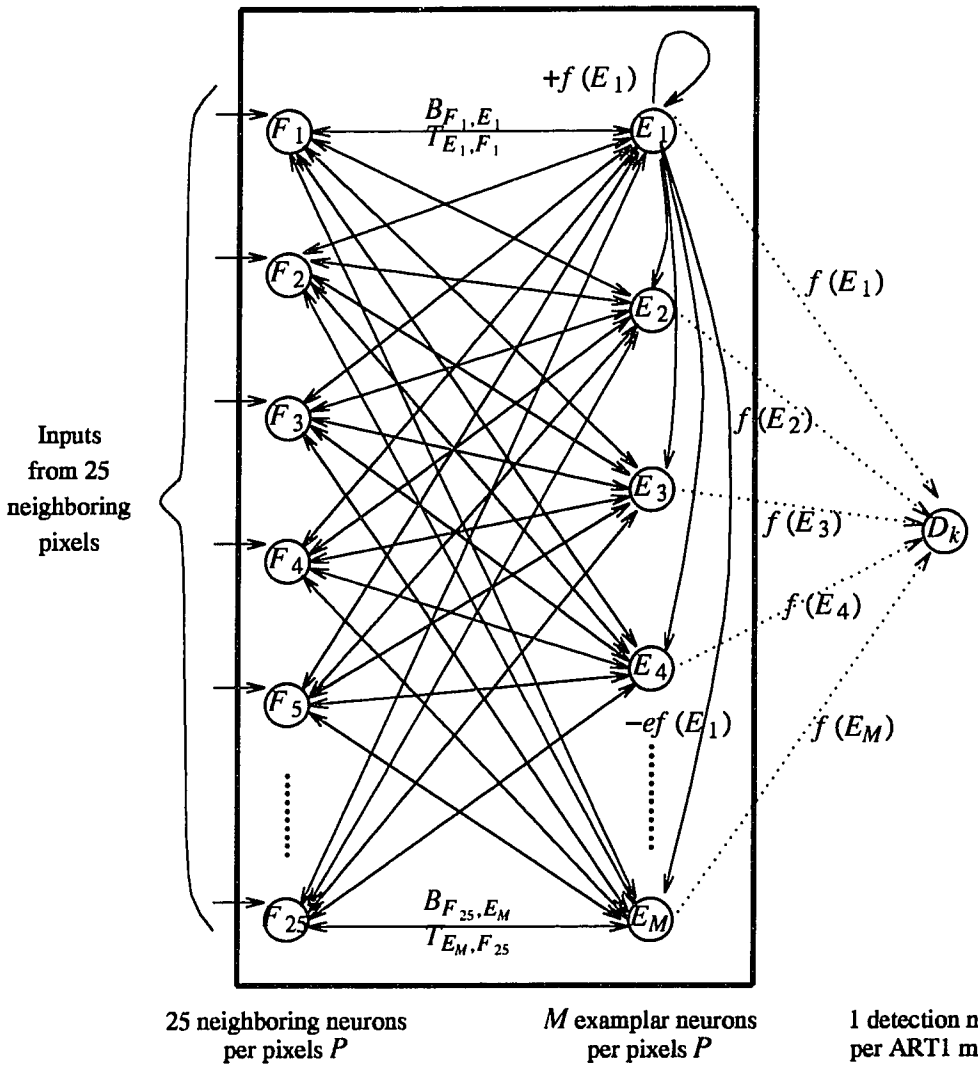


Fig. 3.5(a) An improved ART module.

and M different features are trained as exemplars. For a $q \times q$ input image, Nq^2 neurons in the first layer and Mq^2 exemplar neurons in the second layer are needed. In addition, q^2 region detection neurons are needed in the third layer. Finally the output layer, made up of one neuron O per pixel P, accumulates q^2 neurons. Therefore, the total number of

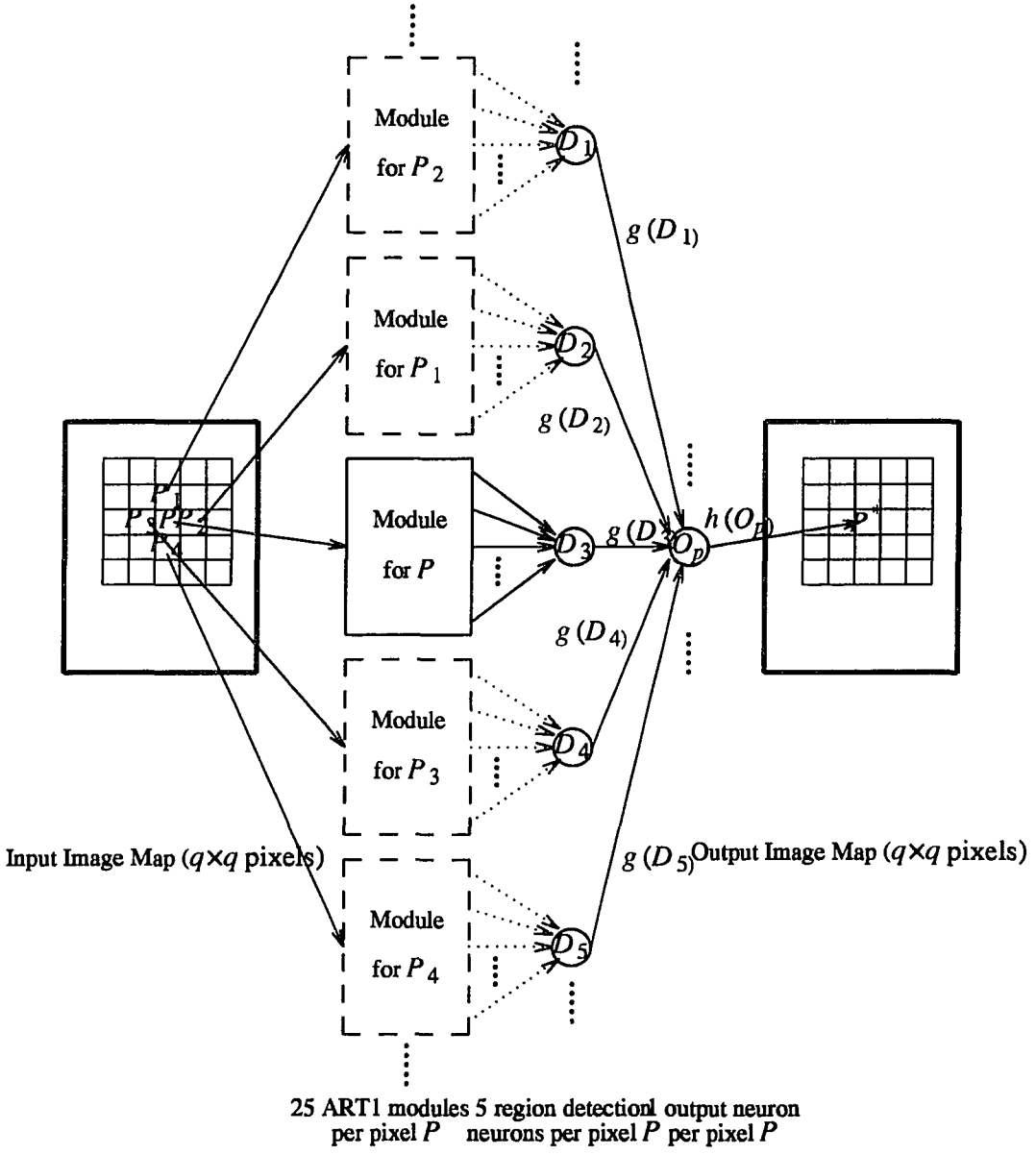


Fig. 3.5(b) An overall neural architecture for image enhancement.

neurons required in the overall architecture is $q^2(N+M+2)$. Additionally, the total number of bottom-up and top-down connections needed is $2NM$.

3.3.2 Algorithm Description

Each exemplar neuron E_j ($j = 1, \cdots, M$) has the inputs from its neighboring neurons F_i ($i = 1, \cdots, 25$) through the bottom-up connections and a lateral inhibitory input from itself and the other exemplar neurons. The lateral inhibitory weight e_{ab} from E_a to E_b is defined as follows:

$$e_{ab} = \begin{cases} 1 & \text{if } a = b \\ -e & \text{if } a \neq b \end{cases} \tag{3.10}$$

where $a, b = 1, \cdots, M$ and $e < \frac{1}{M}$. The purpose of this lateral inhibitory input made by the exemplar neuron is to suppress all the other exemplar neuron outputs toward zero while itself remains unchanged. This process will result in an exemplar output greater than zero and that is the exemplar with the closest category to the input neighborhood of the pixel P. The top-down weights T_{E_j,F_i} are initialized to 1 and updated by Eq. (3.8). As mentioned previously the updating of exemplars is allowed only during the learning phase in order to prevent deterioration of previously learned category exemplars. The updated output f_{E_j} for the exemplar neuron E_j is designed as follows:

$$f_{E_j}^{(t+1)} = \frac{f^{(t)}\left[f_{E_j}^{(t)} - e \sum_{a \neq j} f_{E_a}^{(t)} \right]}{V_{E_j}}, \tag{3.11}$$

where $f_{E_a}^{(t)}$ is the output of exemplar neurons other than E_j and V_{E_j} is the vigilance parameter for exemplar neuron E_j. The function $f(u)$ is a threshold logic type, i.e.,

$$f(u) = \begin{cases} u & \text{if } u \geq 0 \\ 0 & \text{otherwise} \end{cases} \tag{3.12}$$

The incorporation of the vigilance parameter into Eq. (3.11) is to force the network to attempt classification into exemplar categories of lower vigilance first. This prevents

the pixel neighborhood which does not meet with a high vigilance test but meets with a lower vigilance test from misclassifying.

The output function $g(D_k)$ of the detection neuron D_k is the hard-limiter type. That is

$$g(D_k) = \begin{cases} 1 & \text{if } D_k \geq 0 \\ 0 & \text{otherwise} \end{cases}. \tag{3.13}$$

These detection neurons indicate to the output neuron O whether the pixel P should be illuminated or not. The output neuron implements a simple binary edge detection algorithm in which a pixel is illuminated only if it has no neighbor having value 1, indicating that it is noise, and if it has all four neighbors having value 1, indicating that it is an interior point. The four neighbors labeled as P_1, P_2, P_3, and P_4, are located in the east, north, west, and south of the pixel P, respectively.

The output neuron $h(O)$ is given by

$$h(O) = h\left[\sum_{k=1}^{5} g(D_k)\right], \quad \text{where } h(u) = \begin{cases} 1 & \text{if } 1 \leq u \leq 3 \\ 0 & \text{if } u = 0, 4 \end{cases}. \tag{3.14}$$

The algorithm is described as follows:

Step 1 — Network initialization.

The initial top-down weights $T^{(0)}_{E_j,F_i}$ are set to 1, the bottom-up weights $B^{(0)}_{F_i,E_j}$ are set to $\frac{1}{1+5^2}$, and the vigilance parameters V_{E_j} are set to 1.0. Both weights are updated by using eqs. (8) and (9). The detection and output layers are disabled.

Step 2 — Presents Training Patterns and Calculates Outputs

A set of feature patterns are trained into the network and the initial output of the

exemplar neuron E_j is calculated by

$$f_{E_j}^{(0)} = \sum_{i=1}^{25} B_{F_i,E_j}^{(0)} \cdot x_i. \tag{3.15}$$

Step 3 — Choose Maximum Output

The "maxnet" procedure as described in Eq. (3.11) is performed until only one exemplar neuron output is positive.

Step 4 — Vigilance Test

The exemplar neuron E_j pre-stored in the top-down weights T_{E_j,F_i}, is compared with the input pattern **X** by

$$\frac{\sum_i T_{E_j,F_i} \cdot x_i}{\|\mathbf{X}\|} \geq V_{E_j}, \tag{3.16}$$

where $\|\mathbf{X}\|$ is the norm of the input pattern vector, i.e., the total number of one bits in the input vector. If Eq. (3.16) is true, the input pattern belongs to the exemplar category represented by E_j and no new category is created. Otherwise, the input pattern does not belong to the most likely category and a new unused exemplar neuron if available is selected and updated by using eqs. (8) and (9).

Step 5 — Repeat for All Training Patterns

Steps (1) to (4) are repeated until all feature patterns are trained completely. Those steps can be done off-line. After learning, the task of image enhancement is executed.

Step 6 — Prepares for Testing Mode

Weight adjustment is disabled and a suitable vigilance parameter V_{E_j} is selected for each region exemplar. The different vigilance parameters reflect different degrees of importance of features with respect to image enhancement. Specific regions or contours could be easily extracted by the incorporation of a very high vigilance for the particular exemplar in question.

Step 7 — Presents Testing Image

A binary input image is placed into the network and the processing proceeds in parallel for all pixels. One exemplar is selected as in steps (3), (4) and (5) in the learning phase for each pixel and the corresponding output will be positive.

Step 8 — Edge Detection

Each detection neuron D_k outputs a 1 if any output f_{E_j} is positive and 0 otherwise.

Step 9 — Obtains Output Image

Each output neuron O outputs a value 1 or 0 according to Eq. (3.14).

Let us analyze the responses of network activation by using a single template pattern. In Eq. (3.16), the term $\sum_i T_{E_j^*,F_i} \cdot x_i$ is simply the inner product of the two vectors: input pattern **X** and template pattern **T** retrieved, which indicates the number of matched elements between the two patterns. By dividing the inner product with the norm of the input pattern, we obtain a fractional number ranging from 0.0 to 1.0 which indicates the degree of matching. By choosing a vigilance parameter equal to or less than $\frac{1}{\|\mathbf{T}\|}$, any occurrence of a pixel "1" of input pattern matched with the template will result in passing the vigilance test. In this case, the network acts like an expanding operator in image processing. On the other hand, choosing a higher vigilance parameter,

e.g., greater than $\frac{\|\mathbf{T}\| - 1}{\|\mathbf{T}\|}$, makes the vigilance tests difficult to pass. Even though the total pixels $\|\mathbf{T}\| - 1$ of input pattern are matched with the template, it will result in failing the vigilance testing. Thus, this situation is similar to a shrinking operator. The vigilance parameter can be viewed as making a "fuzzy" decision on the pixel concerning the probability of being noisy points. The method depends on first distinguishing noise from nonnoise in the picture, then removing the noise and "mending" the picture by interpolation.

A better estimate could be obtained if we used a larger neighborhood; but then the complexity of detecting edges and curves becomes greater, since there are many ways in which an edge or curve can pass through a large neighborhood. A alternative approach is to use a small neighborhood, but to iterate the enhancement process.

3.3.3 Experimental Results

To understand the performance of the neural architecture applied to image enhancement, we examine their behavior on two simulated images. Assuming that the nonnoisy picture contains a large region of constant gray level, we can get some insight into the noise statistics by analyzing the gray level fluctuations in the corresponding region of the noisy picture. The region contrast is 100. To this image, we add independent Gaussian noise having mean zero and standard deviation 35. Thus the signal to noise ratio (SNR – region contrast/noise deviation) of these images is 2.86. The original and noisy images are shown in Figs. 3.6 and 3.7, respectively.

These vigilance parameters are updated as weighted probabilities at neighboring points. The presence of lines and edges determines how the contribution from each neighboring point is weighted. The method of determining the weights is based solely on the possible existence and distribution of lines and edges. To compare the performance of different vigilance parameters and the number of iterations, we use the conditional

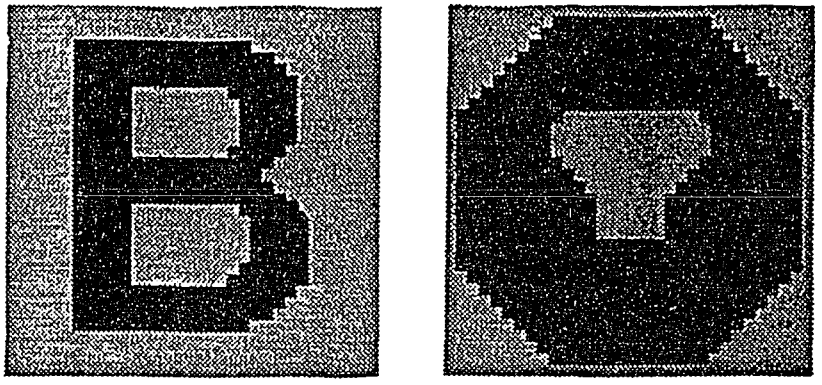


Fig. 3.6 The original binary images.

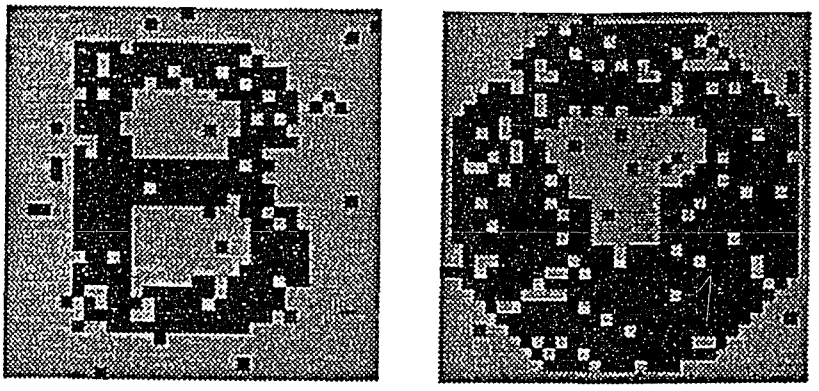


Fig. 3.7 The noisy images with added Gaussian noise having zero mean and standard deviation 35.

probability of the labeled "object-point" given the true object-point, $P(O'|O^*)$ and the conditional probability of a true object-point given the labeled object-point, $P(O^*|O')$. The adjustable parameters of each iteration and the number of iterations are chosen to equalize these two conditional probabilities. The quality of the neural architecture to image enhancement is determined by the value of $P(O'|O^*) = P(O^*|O')$. Although this performance measure is not in general applicable on all kinds of images, it is well suited for these simulated images.

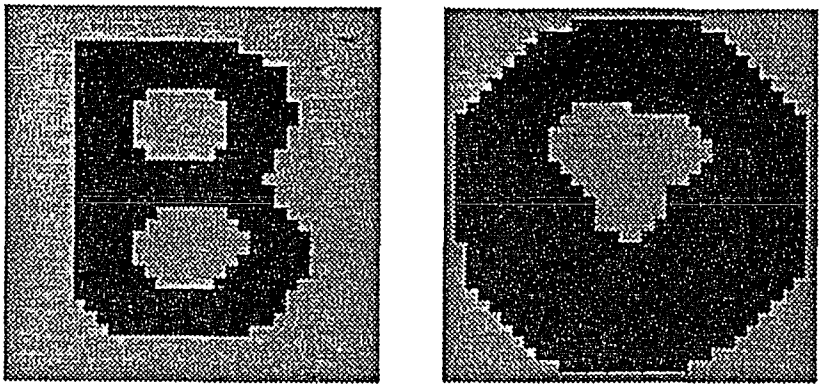


Fig. 3.8 The output of the region detection layer.

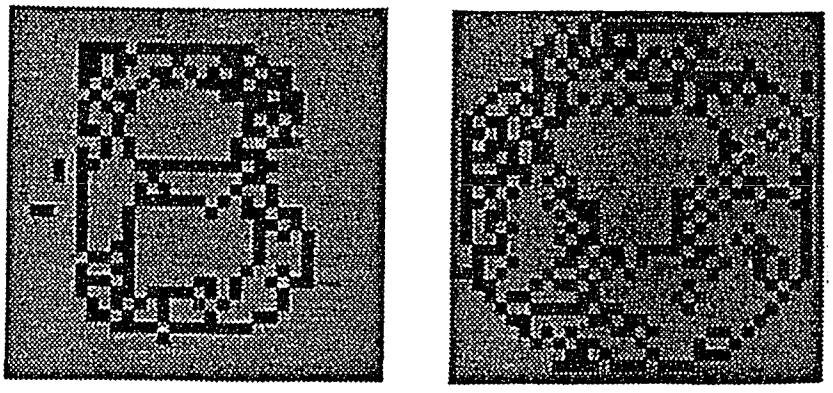


Fig. 3.9 The result of applying the noisy input image in Fig. 3.7 to the network with all layers disabled except the output neuron layer.

Figs. 3.8-3.10 demonstrate the performance of the network for the noisy binary images in Fig. 3.7. Fig. 3.8 shows the output of the region detection layer. Note that a significant portion of the region of the image is filled in and that most extraneous noisy pixels have been eliminated. Fig. 3.9 shows the result of applying the noisy input image in Fig. 3.7 to the network with all layers disabled except the output neuron layer. This would represent the result of simply performing an edge detection operator. Fig. 3.10

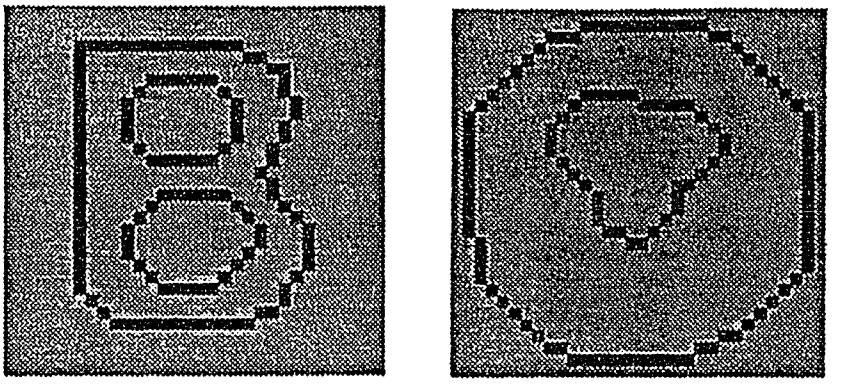


Fig. 3.10 The result of applying Fig. 3.7 to the network with all layers in operation.

shows the result of applying Fig. 3.7 to the network with all layers in operation. There is a significant performance improvement noted when comparing Figs. 3.9 and 3.10. The training patterns used are shown in Fig. 3.11 along with the vigilance selected for each of the contour or region exemplars. The involved two-dimensional features, such as E[4] to E[7], which require higher vigilance values because more restricted requirement must be met for image enhancement. The first iteration uses higher vigilance parameters and the second iteration applies lower ones. The network could reach the best condition, $P(O'|O^*) = P(O^*|O')$, with two iterations. The performance of applying to the character "**B**" is given in Table 3.1.

Tab. 3.1 The Performance of Experiment "B"

Iteration	$P(O'\|O^*)$	$P(O^*\|O')$
1	0.909	0.978
2	0.947	0.944
3	0.947	0.915
4	0.947	0.899

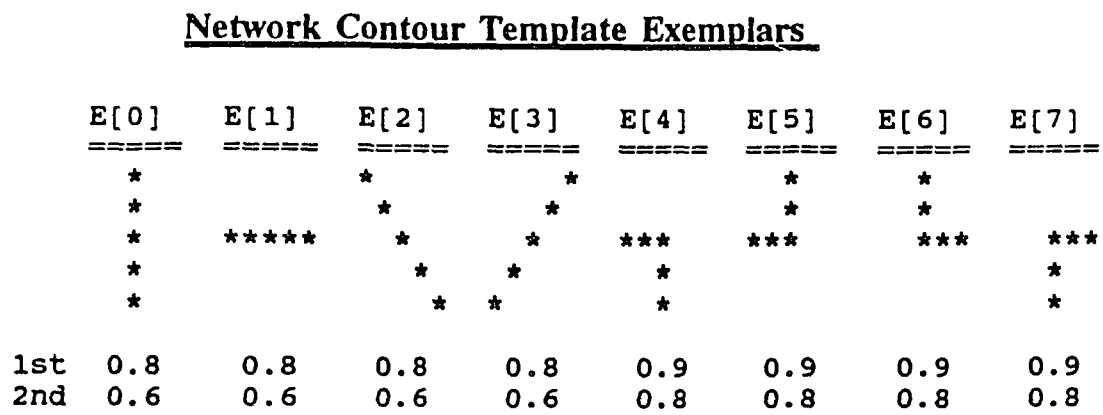


Fig. 3.11 The training patterns used along with the vigilance selected for each of the contour or region exemplars.

3.4 Conclusion

As can be seen from Figs. 3.6-3.10 the neural architecture devised does show feasibility when applied to the pattern classification and the enhancement of noisy binary images independent of the image itself. It has been demonstrated that neural networks can be used to implement simple edge detection schemes and also more importantly they can be used successfully to enhance the performance of these schemes by creating more uniform regions within binary images and reducing unwanted noise. Many gaps in image interiors and edges were filled in and many extra or stray pixels were also successfully eliminated.

There are many avenues of possible performance improvement of the network described that may be investigated. Improved or different edge detection schemes or modified architectures might be employed. The input pixel neighborhood of 5×5 might possibly be changed to include a larger neighborhood, or variable neighborhoods could be employed in an attempt to accurately smooth points of inflection. A most logical extension of this work would be the incorporation of Carpenter and Grossberg's

ART2 network [13] which is designed to handle multivalued or analog inputs. This would enable this work to be expanded to include gray level images.

CHAPTER 4

A GENERAL PURPOSE MODEL FOR IMAGE OPERATIONS BASED ON MULTILAYER PERCEPTRONS

A computer vision system always facilitates the capabilities of low-level image processing, such as smoothing, enhancement, edge detection, noise removal, and morphological operations. These operations can be viewed as automatic reactions, requiring no intelligence involved [45], and are compared to the sensing and adaptation processes which human's eyes go through when he or she tries to find a seat immediately after entering a dark theater. The process of finding an unoccupied seat cannot begin until a suitably clear image is available. Such kind of visual system adaptation to producing an appropriate image is an automatic and unconscious reaction, and is, of course, a highly parallel task in biological visual systems.

Most of image processing algorithms [45][92] use neighborhood processing that takes into account the values of a pixel's neighbors to determine its new value. The computation is defined within a finite mask and is independent of new computed values. Therefore, parallelism can be easily achieved to gain high performance. The introduction of neural network models is intuitively due to their highly parallelism and learning capabilities.

Some work in developing morphological neural networks has been found in literature. Davidson [22][23] redefined the weighted-sum activation function with additive maximum operation, which in fact is the gray-scale dilation in mathematical morphology [120]. Shih and Moh [112] developed a neural architecture to extract the maximum or minimum value among nine inputs by parallelly comparing each pair of inputs in one stage. Morales and Ko [83] proposed an efficient implementation of neural training algorithms and defined an overall equality index related to fuzzy implication,

called *Lukasiewicz &-conjunction*, as a performance index. All the aforementioned architectures can perform a specific kind of operations only. The development of new architectures and algorithms for the general purpose image operations is inevitably needed.

In [69], the parallel implementations of Rosenfeld-Kak's thinning algorithm [45] and Wang-Zhang's thinning algorithm [133] adopted the use of recurrent multistage multilayer neural networks. They applied deletion rules to determine whether an object pixel should be eliminated in skeletonization. The results of using neural networks are as expected the same as those of using traditional programming. However, the architectures are only designed for the thinning task.

Our goal is to design a general purpose architecture which can adapt itself to different sets of operations. We adopt the multilayer perceptrons (MLPs) as processing modules so that no special neural net has to be defined. Such an architecture promises the feasibility of learning different operations as well as applying an operation in parallel on all pixels in the image. The proposed MLP modules may learn transformation rules in two different ways: using training patterns derived from existing algorithms of well-defined transformations and using training patterns extracted from sample images and target images. The former is significant in processing binary images and the latter is meaningful for unknown operations. With a number of stages stacked up together we may apply a series of operations on an image. Besides, recurrent connections from output of the last stage to input of the first stage allow the repetition of a sequence of operations.

In this chapter, section 4.1 discusses how a multilayer perceptron can be used as processing modules for low-level image operations. The structure along with the training and operating phases of the modules are discussed in details. Section 4.2 elaborates the system architecture proposed for adaptive image processing. Section 4.3 shows an example of system design and explains the idea of extracting training patterns from

sample images and their corresponding desired images after processing. Section 4.4 illustrates the experimental results of using the proposed architecture. Section 4.5 concludes the paper and points out some directions for future research.

4.1 Multilayer Perceptron As Processing Modules

Multilayer perceptrons are feed-forward nets with one or more layers, called *hidden layers*, of neural cells, called *hidden neurons*, between the input and output layers. The learning process of MLPs is conducted with the error back-propagation learning algorithm derived from the *generalized delta rule* [95][96]. According to [73], no more than three layers of neurons are required to form arbitrarily complex decision regions in the hyperspace spanned by the input patterns. By using the sigmoidal nonlinearities and the decision rule in selecting the largest output, the decision regions are typically bounded by smooth curves instead of line segments. If training data are sufficiently provided, an MLP can be trained to discriminate input patterns successfully. A discussion on limits of the number of hidden neurons in multilayer perceptrons can be found in [55].

Most of image processing operations, such as smoothing, enhancement, edge detection, noise removal, and morphological operations [45][112], require to check the values of neighboring pixels. Two basic morphological operations, dilation (similar to "expansion") and erosion (similar to "shrink"), are often combined in sequences for image filtering and feature extraction. The size of neighborhood may vary with applications. Eight-neighbor along with the center pixel is often used. The neighborhood processing can be implemented by look up tables to associate the relationship between input and output values. The input-output association can be realized by neural network models with nine input neurons (assuming that the 8-neighbor area is used) and one output neuron with one or more layers of hidden neurons in between. By iteratively

providing input vectors to compute output from the look up table and comparing with the desired output, the error will be used to adjust the weights of connections. Therefore, MLPs can gradually and adaptively learn the input-output association. If the training MLP converges with a set of weights, it can be used to perform the same transformation on any image.

Fig. 4.1 shows the MLP modules designed as the general-purpose image processing modules. The more hidden layers we use, the more complicated discriminant regions it forms in the domain space spanned with input vectors. There is a tradeoff between the complexity of MLP connections and the converge time of the MLP. Since we are interested in the general purpose MLP module, reasonably short training time to adapting the MLP and big capacity of connection weights are expected. An example of determining the number of hidden neurons in the required MLP of Fig. 4.1(a) is discussed in section IV.

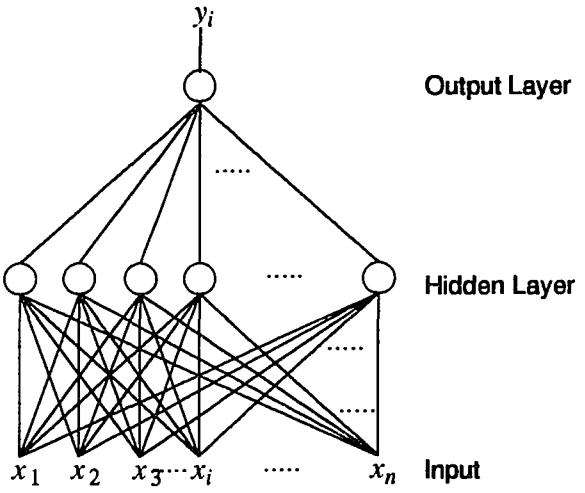
The error back-propagation training algorithm was given in [95]. A sigmoidal logistic nonlinearity

$$f(\alpha) = \frac{1}{1 + e^{-(\alpha-\theta)}} \tag{4.1}$$

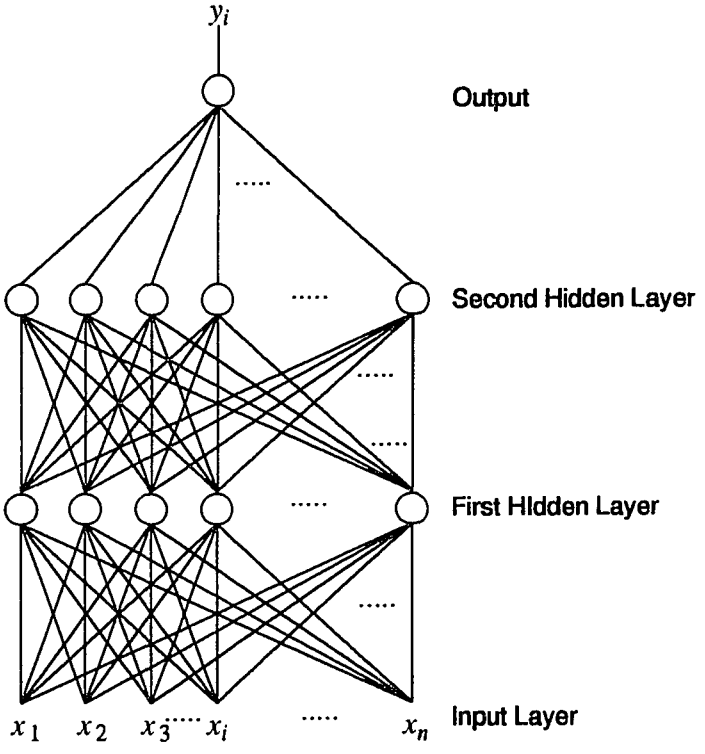
is used as the output function of neurons, where θ is a bias (or a threshold). Let $x_i^{(n)}$ denote the output of neuron i in layer n. Also, let w_{ij} denote the connection weight from neuron i in a lower layer to neuron j in the immediately higher layer. The activation values are calculated by

$$x_j^{(n+1)} = f\left[\sum_i w_{ij}\, x_i^{(n)} - \theta_j \right] \tag{4.2}$$

The major part of the algorithm is to adjust the connection weights according to the difference between the actual output of Eq. (4.2) and the desired output provided in the



(a) A multilayer perceptron includes only one hidden layer.



(b) A three-layer perceptron includes two hidden layers.

Fig. 4.1 Multilayer perceptron modules for the general purpose image processing.

training patterns. The equation for weights adjustment is

$$w_{ij}(t+1) = w_{ij}(t) + \eta \delta_j x_i^{(n)} \tag{4.3}$$

where η is the gain term (or called *learning rate*), δ_j is the error term for neuron j, and x_i is either the output of neuron i or an input when $n = 1$. The error term δ_j adopts the use in [73] as

$$\delta_j = \begin{cases} x_j^{(n+1)} (1 - x_j^{(n+1)}) (d_j - x_j^{(n+1)}) & \text{if neuron } j \text{ is in the output layer} \\ x_j^{(n+1)} (1 - x_j^{(n+1)}) \sum_k \delta_k w_{jk} & \text{if neuron } j \text{ is in a hidden layer} \end{cases} \tag{4.4}$$

where k is with respect to all the neurons in the layer where neuron j is located.

0	1	0
1	1	1
0	1	0

Fig. 4.2 A morphological structuring element.

0 1 0 1 1 1 0 1 0	1 1 0 1 1 1 0 1 0	0 1 0 1 1 1 1 1 0	0 1 0 1 1 1 0 1 1	0 1 1 1 1 1 0 1 0	1 1 0 1 1 1 1 1 0	0 1 0 1 1 1 1 1 1	0 1 1 1 1 1 0 1 1
1 1 1 1 1 1 0 1 0	1 1 0 1 1 1 0 1 1	0 1 1 1 1 1 1 1 0	1 1 0 1 1 1 1 1 1	0 1 1 1 1 1 1 1 1	1 1 1 1 1 1 0 1 1	1 1 1 1 1 1 1 1 0	1 1 1 1 1 1 1 1 1

Fig. 4.3 Patterns having desired output 1 with erosion by the structuring element in Fig. 4.2.

The training patterns can be generated by all the variations in a local window of the specified low-level image operations. For example, in a 3×3 neighboring area, there are up to $2^9 = 512$ different training patterns, with a vector of 9 binary values and one desired output. For instance, a morphological erosion with the structuring element shown

in Fig. 4.2 will respond output 1 at the central pixel for the 16 input patterns in Fig. 4.3 and output 0 for other input patterns.

4.2 The System Architecture

Fig. 4.4 shows the overall architecture consisting of MLP modules for low-level image operations. There is one MLP module corresponding to each pixel in the input image. Every MLP module applies the operation which has been trained to the pixel and its $m \times m$ neighboring pixels where m means the window size of neighborhood. The window size can be changed depending upon the desired operations for training.

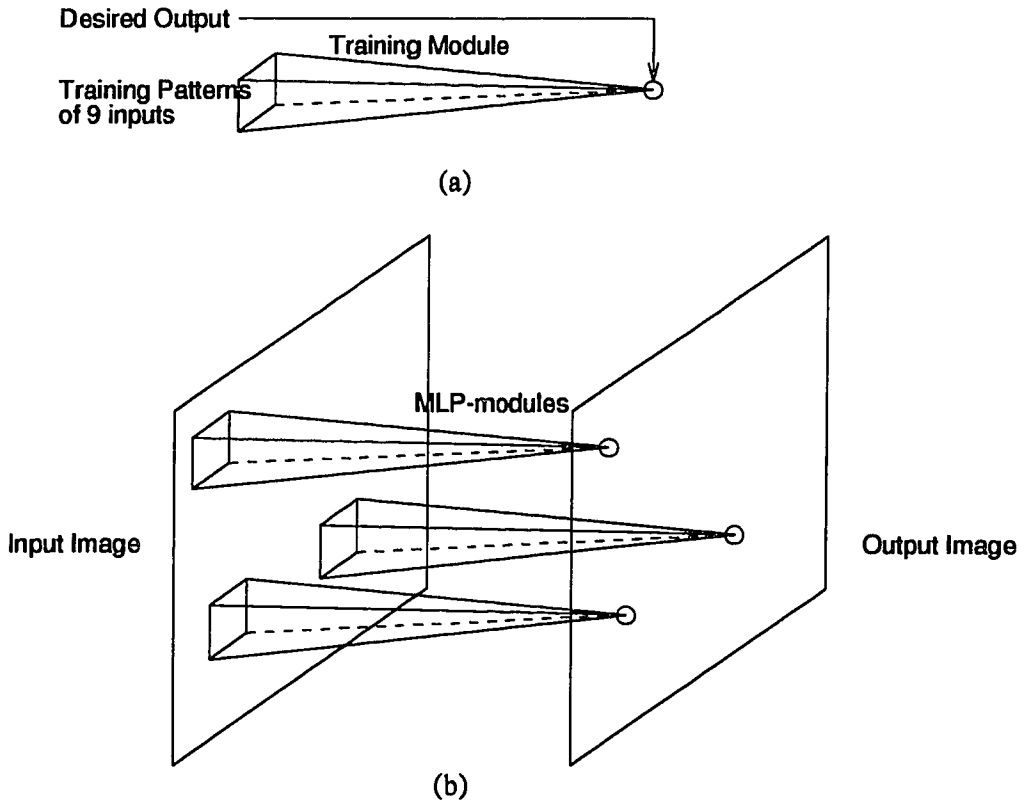


Fig. 4.4 Overall system architecture with MLP modules for low-level image operations. (a) The pyramid is used for training tasks and storage of the trained weights. (b) Each pyramid represents an MLP module and receives m^2 inputs from its $m \times m$ neighboring pixels.

Since the same operation is simultaneously applied to all the modules for all the pixels in the input image, the set of connection weights are considered the same for all MLP modules. That is, all MLP modules in the same stage can share the same set of trained connection weights. This will significantly reduce the required number of local memory for connections weights in MLP module. In Fig. 4.4(a), the pyramid, called the training module, is used for training task and provides shared memory for connection weights. Thus, no local memory is required for individual MLP modules in between the input and output images to hold the connection weights. The training phase is first turned up at the training module. After the weight vector in the training module converges, the connection weights are frozen, shared, and retrieved by all the MLP modules to determine the activation value of each pixel after the operation is applied.

With the introduction of the error back-propagation learning algorithm, the MLP modules are capable of adapting themselves to the expected operations. It is also more flexible than those designed for dilation and erosion operations only [69].

During the experiments with different image operations, we found out that the training sets for some operations, such as dilation, erosion, and noise removal, have a small number of patterns with the desired output 1 or 0. For example, an erosion with the structuring element in Fig. 4.2 has only 16 patterns with output 1, and an erosion with the structuring element shown in Fig. 4.5 expects an output 1 only if all nine inputs are 1's.

1	1	1
1	1	1
1	1	1

Fig. 4.5 The structuring element with all 1's.

By checking the training patterns, we figure out that the values of some m^2 inputs do not even affect the output. Without considering a certain value of the central pixel,

the training pattern set can be reduced. The same effect can be achieved by connecting the output neuron directly with the central input neuron and the MLP module operation is bypassed when the central pixel equals 0 or 1. For instance, applying an erosion to the central pixel with value 0 will never output 1. Also, applying a dilation to the central pixel with value 1 will always output 1. Such bypassing connections can be specified with the desired output as weights shared in all the MLP modules. The bypassing connections are intended to perform an exclusive OR between the central pixel and the unchanged input which is defined with respect to operations. The operation of an MLP is disable if the exclusive OR gets output 1. It means that the central pixel has the same value as specified in the shared module. Fig. 4.6 illustrates the bypassing connection and expresses how it affects the operation of an MLP module. The bypassing control is to enable/disable the activation of neurons in the modules and is determined by

$$E = x_c \ \mathbf{XOR}\ U \tag{4.5}$$

where E means the enable/disable control to neurons, x_c is the value of the central pixel, and U is the expected input. For instance, we set $U = 1$ for dilation and $U = 0$ for erosion. The dashed lines with arrows in Fig. 4.6 show the enable/disable controls to neurons, while the dotted lines indicate the bypassing connection for passing the value of the central pixel directly to the output. The output is defined by

$$\text{Output} = y \,.\, E + x_c \,.\, \bar{E} \tag{4.6}$$

where y is the output of the MLP module.

The proposed architecture can be used as a basis to organize multistage or recurrent networks. We may stack up with more than one stage as shown in Fig. 4.7. By training the stages with a different set of patterns, we can provide and apply a sequence of image operations to the input image. The outputs of the last stage may be connected to the inputs of the first stage to form a recurrent architecture so that the same operations in

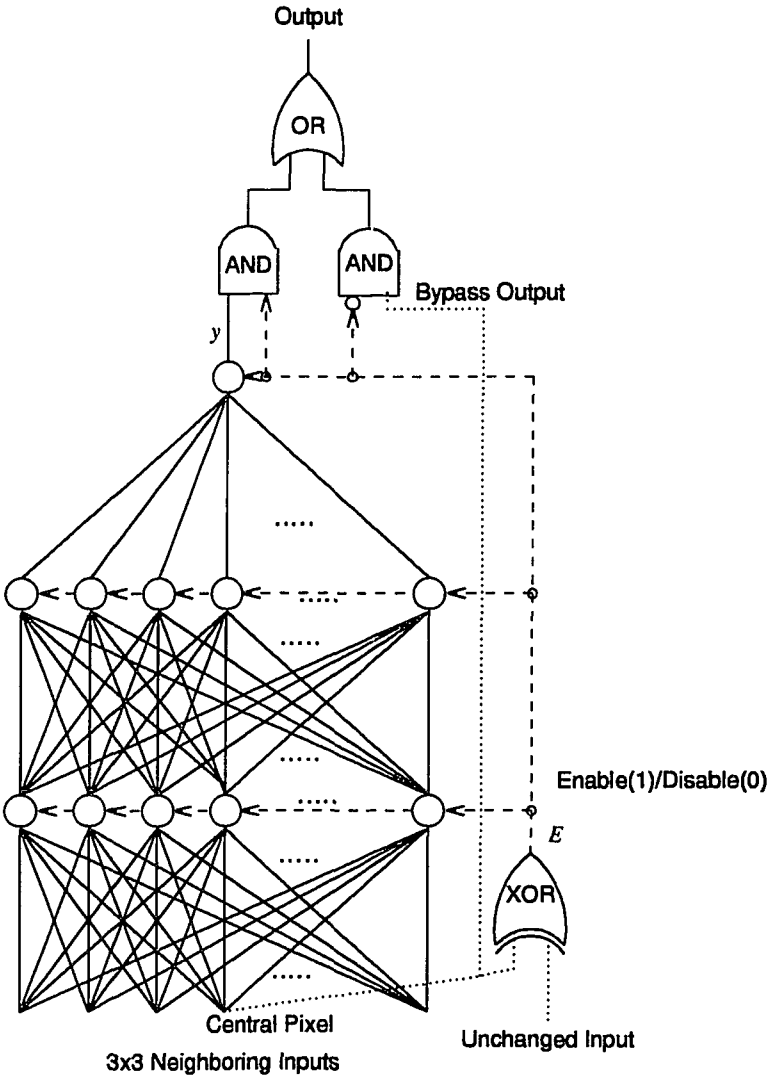


Fig. 4.6 MLP module with a bypass control to enable/disable the activation of neurons in the modules. The dashed lines with arrows show the enable/disable controls to neurons, while the dotted lines indicate the bypassed connection for passing the value of the center pixel to the output.

sequence can be applied to the image for more than one iteration. Therefore, it forms a recurrent neural architecture.

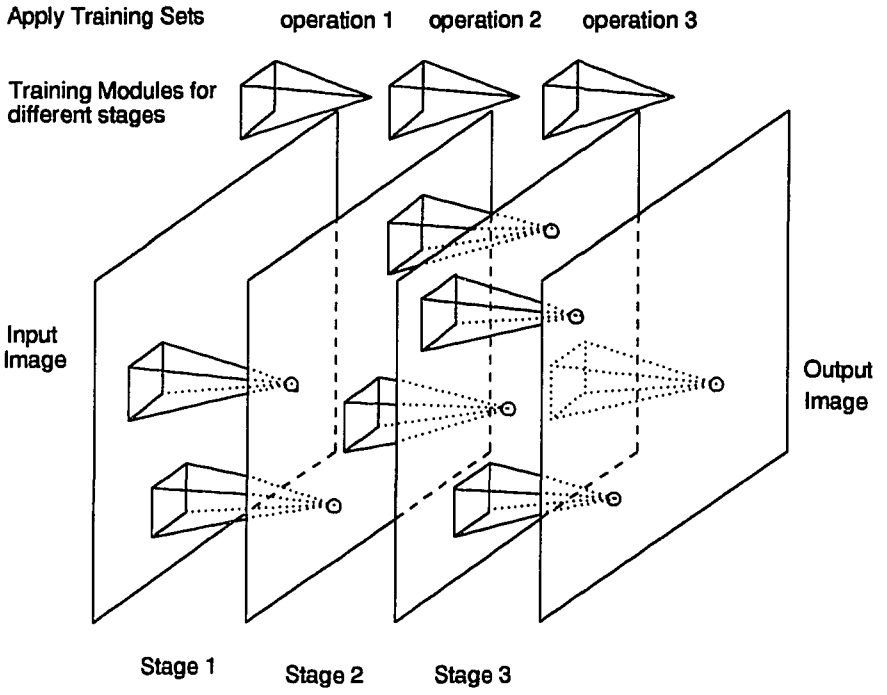


Fig. 4.7 Stacked MLP-based architecture for multiple image operations. Only some MLP-modules are shown to simplify the figure. The separated pyramids at the top of the figure are the training modules for different stages. There may be more stages stacked on the architecture.

4.3 Training Examples

In determination of the number of hidden neurons in MLP modules, we elaborate the following example. We investigate the training iterations and convergence conditions in training the MLP module with patterns obtained from morphological dilation and erosion, edge detection with Sobel operators, and noise removal for binary images. The experimental results for these operations are shown in Tables 4.1, 4.2, 4.3, and 4.4, respectively. The MLP module used in this investigation is the same as the one shown in Fig. 4.1(a) with nine inputs and 1 output neurons. With a fixed momentum, e.g. 0.1

shown at the top of the tables, we change the learning rate, η, by varying from 0.01 to 0.05 and the number of hidden neurons in the hidden layer by varying from 1 to 20 (as shown in the left-most columns) to determine the number of iterations for convergence. A "×" symbol in the table entry indicates that the training can not converge in 30,000 iterations. The same set of random weights is used as initial connection weights for all these operations.

Tab. 4.1 Dilation ($\alpha = 0.1$)

Number of hidden neurons	η				
	0.01	0.02	0.03	0.04	0.05
20	130	97	85	71	64
19	143	138	102	105	93
18	134	140	91	72	70
17	156	138	88	76	59
16	114	61	51	48	41
15	125	99	94	80	64
14	120	96	81	65	59
13	139	92	72	62	53
12	114	59	44	40	40
11	124	63	42	34	31
10	121	61	43	40	43
9	117	74	70	62	52
8	193	139	115	102	71
7	160	100	75	57	44
6	136	72	52	44	40
5	153	94	77	59	47
4	195	111	75	63	46
3	181	92	62	49	41
2	192	99	67	53	50
1	203	108	76	62	53

Tab. 4.2 Erosion ($\alpha = 0.1$)

Number of hidden neurons	η				
	0.01	0.02	0.03	0.04	0.05
20	141	90	71	80	125
19	130	59	73	81	83
18	119	73	65	81	78
17	123	87	57	77	73
16	212	64	77	98	298
15	140	91	64	78	92
14	142	96	73	67	91
13	137	95	64	69	105
12	174	102	72	87	421
11	163	103	107	145	145
10	188	90	89	111	131
9	196	111	85	79	99
8	139	94	83	78	74
7	164	126	102	91	116
6	201	100	112	110	122
5	119	125	86	89	105
4	156	119	121	174	1041
3	204	96	118	272	×
2	250	129	95	118	101
1	224	169	138	133	157

We may determine certain ranges for both the number of hidden neurons and the learning rate so that the MLP-training module is guaranteed to converge in a small number of training iterations. For instance, in the training examples, the number of hidden neurons can be chosen in between 10 and 18 and the learning rate can be set to any value from 0.01 to 0.02. For each operation we create and store a look-up table like this along with training patterns. A signal representing a certain operation will load a good choice of pairs (number of hidden neurons and learning rate) from the table, load the training patterns, and start the training phase.

Tab. 4.3 Sobel Edge Detection ($\alpha = 0.1$)

Number of hidden neurons	η				
	0.01	0.02	0.03	0.04	0.05
20	783	1602	×	×	×
19	905	×	1578	×	×
18	862	1787	×	×	×
17	921	1468	2901	×	×
16	895	534	534	×	×
15	978	2158	×	×	×
14	946	×	×	×	×
13	1190	3762	×	×	×
12	887	1371	×	×	×
11	1080	1390	×	×	×
10	1334	3376	×	×	×
9	1650	×	×	×	×
8	1407	2486	×	×	×
7	6656	×	×	×	×
6	2065	17039	×	×	×
5	4220	2312	×	×	×
4	×	×	×	×	×
3	×	×	×	×	×
2	×	×	×	×	×
1	×	×	×	×	×

4.4 Experimental Results

In the experiment of applying the thinning algorithm in [45], we stacked with four stages of the proposed architecture and trained them with the patterns generated by four deletion rules in the order of north, south, east, and west. The MLP module adopted in this experiment is the same as that shown in Fig. 4.1(a) with only one hidden layer. According to [69], the minimum number of hidden neurons is three in order to accomplish the thinning algorithm. Instead of using this minimum number of hidden neurons for the thinning algorithm only, we aim at designing a flexible and general-

Tab. 4.4 Noise Removal ($\alpha = 0.1$)

Number of hidden neurons	η				
	0.01	0.02	0.03	0.04	0.05
20	100	120	162	247	243
19	106	77	157	219	×
18	90	106	241	219	215
17	107	101	148	302	260
16	97	121	295	962	×
15	103	133	216	136	145
14	108	143	164	138	121
13	102	147	226	186	206
12	94	112	342	×	714
11	109	104	194	253	353
10	96	119	4331	×	×
9	95	142	1798	×	×
8	103	133	194	×	×
7	103	131	213	115	×
6	93	108	232	×	×
5	128	138	307	259	150
4	110	111	848	×	×
3	116	122	1125	298	1162
2	122	160	637	×	×
1	100	93	316	×	×

purpose architecture for image operations. Therefore, we choose a reasonable number, i.e. nine, of hidden neurons obtained from the task discussed in Section IV to reduce the number of iterations. Fig. 4.8(a) shows the original images which are bitmaps of characters "**C**", "**L**", and "**Y**" extracted from a scanner. They are binary images with 64x64 pixels. We also implement the thinning algorithm in C language on a SUN SPARC-1 workstation to verify the correctness of our simulation results. Fig. 4.8(b) shows the thinning results along with their original images to illustrate the relationship between the objects and their corresponding skeletons. The simulating results with the

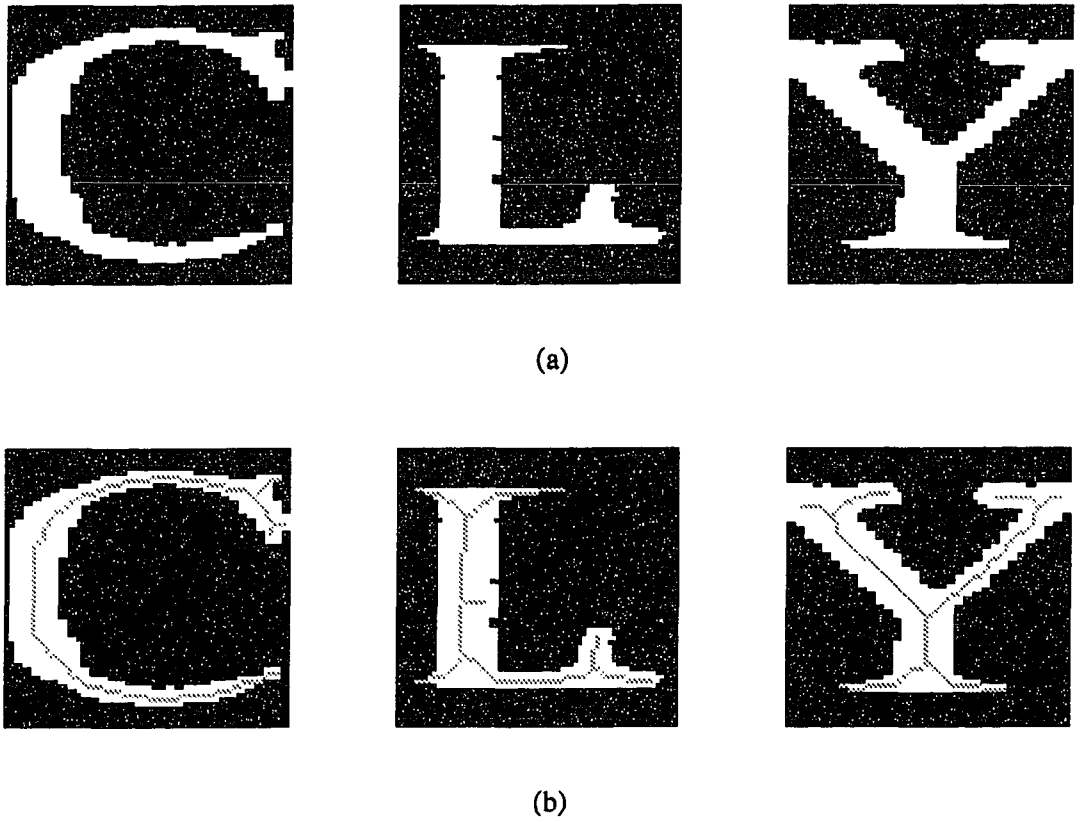


Fig. 4.8 (a) Three original binary images for thinning experiments. (b) The experimental result of RK thinning with the proposed architecture.

proposed architecture for binary images are verified to be identical to Fig. 4.8(b). This shows that the proposed architecture can be trained to perform basic image operations.

4.5 Conclusions

An architecture incorporating MLP's as the fundamental modules is proposed for low-level image processing. This model is flexible — it can adapt itself with training processes to most of the low-level image operations. Experiments have been carried out to show the feasibility of the proposed architecture. Since the training patterns are generated with existing algorithms for the specified operations and the MLP modules are

converged at the end of training, it is guaranteed that the architecture produces the same result as the same algorithm is applied.

The training phase of an MLP module usually takes a long time to converge. This could be a major drawback in the implementation of the proposed architecture. A further research for developing another neural net model as the pixel-based operating modules is expected. Also, self-determining the window size of pyramid modules is another interesting field to study.

CHAPTER 5

IMPLEMENTING MORPHOLOGICAL OPERATIONS USING ARTIFICIAL NEURAL NETWORKS

ANN models can be considered as signal processors which are "value passing" architectures, conceptually similar to traditional analog computers. Each neuron may receive a number of inputs from the input layer or some other layers, and then produces a single output which can be as an input to some other neurons or a global network output. Many neural network models, such as Rosenblatt's Minsky's *Perceptron* [80][94] and Fukushima's *Cognitron* [35], which use binary inputs and produce only binary outputs, can implement Boolean functions.

In this chapter, we are mainly concerned with feed-forward layered neural networks which form a simple hierarchical architecture. In such a network, execution of each layer is performed in a single step. The input neurons in the first layer can be viewed as entry points for sensor information from outside world, or called *sensoring neurons*, and the output neurons in the last layer are considered as decision points control, or called *decision neurons*. The neurons in the intermediate layers, often called *hidden neurons*, play the role of interneurons. The network implicitly memorizes an evaluated mapping from the input vector and the output vector at the execution time, each time the network traversed. The mapping, represented as $\mathbf{X} \rightarrow \mathbf{Y}$, is just a transfer function that takes the n-dimensional input vectors $X = (x_1, x_2, \cdots, x_n)$ and then outputs a m-dimensional output vectors $Y = (y_1, y_2, \cdots, y_m)$.

Mathematical morphology based on set-theoretic concepts can extract object features by choosing a suitable structuring shape as a probe. It was first investigated by Matheron [75] and extended by Meyer [78] and Serra [104]. Image morphology can extract shape features such as edges, fillets, holes, corners, wedges, and cracks by

operating with various structuring shapes. In industrial vision applications, it can be used to implement fast object recognition, image enhancement, and flaw inspection.

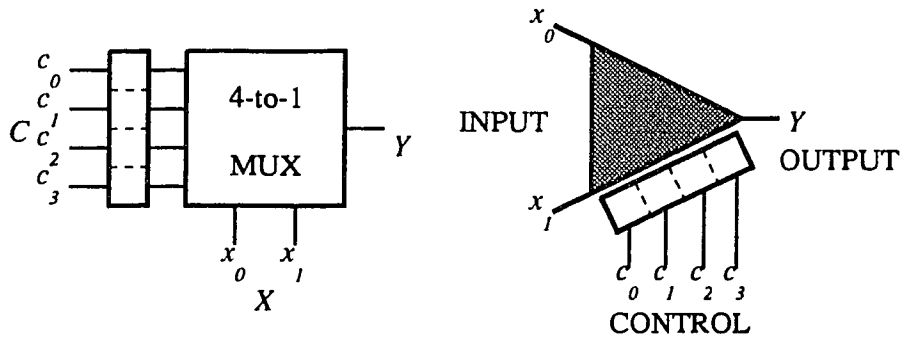
This chapter is divided into four sections. First, the concept of programmable logic neural networks is introduced. Second, the implementation of binary morphological operations using the logic neural networks is developed. Third, the implementation of gray-scale morphological operations by tri-comparators and by tree structures is proposed. Finally, an example of applying the networks to the activation of neocognitron is illustrated. (This chapter has been published as a journal paper [112] and related works can referenced in [107] and [109].)

5.1 Programmable Logic Neural Networks

5.1.1 Structure of Programmable Logic Networks

The concept of building neural network structure with dynamically programmable logic modules (DPLM's) is based on early efforts aimed at providing general design of logic circuits by using *universal logic modules* [5][131]. The neural networks are constructed with node modules which have only a limited number of inputs. Fig. 5.1 shows a module of 2-bit input DPLM which consists of a commonly used 4-to-1 multiplexer and a 4-bit control register. In this circuit, the role of 4-bit input and 2-bit control is exchanged. We use the 2-bit control (X) to select any bit of the 4-bit value in the register (C) as the output (Y).

In Fig. 5.1, $X = (x_0 , x_1)$ denotes the input vector having 4 combinations: { 00, 01, 10, 11 }. The control register C determines the value of the single-bit output Y from the input vector. Therefore, by loading a 4-bit control word, one of the 16 (i.e. 2^4) possible Boolean functions for two input variables can be chosen. These Boolean functions and associated control codes (i.e. the truth table) are given in Table 5.1. A two-input DPLM with all 16 possible Boolean functions is called a *universal* module. If the 16 possible



(a) (b)

Fig. 5.1 (a) A neuron cell consists of a 4-to-1 multiplexer and a 4-bit control register. (b) Symbol for a two-input DPLM with a 4-bit control register.

functions are not fully utilized, the size of the control register could be reduced [131].

Two-input modules are the simplest primitive units. Fig. 5.2 illustrates a general-purpose three-input DPLM composed of six two-input modules, which was proposed in [131]. The desired function is selected by the corresponding control codes loaded to the six two-input modules. For example, that AND codes are loaded to all six modules acts as the AND function of the three inputs; this is called CONVERGE. By loading OR codes, similarly, it performs the OR function of the inputs, called DIVERGE. The accumulative combinations of primitive blocks can be used to achieve more complex architectures for various applications.

In this chapter, we are concerned with neighborhood processing processed especially for morphological operations. Hence the structure of a DPLM in Fig. 5.2 could be reduced into the structure in Fig. 5.3. In the neighborhood operations, only three two-input modules are needed in the pyramid structure. That is, the module in the upper level will receive inputs from the two in the lower level. Instead of loading the same control code to all modules, the new architecture can be loaded with different control codes in each level so that more functions could be performed.

Tab. 5.1 Truth Tables of 16 possible Boolean Functions.

Function	Control code	Input $(x_0\, x_1)$			
		00	01	10	11
FALSE	0000	0	0	0	0
AND	0001	0	0	0	1
LFT	0010	0	0	1	0
LFTX	0011	0	0	1	1
RIT	0100	0	1	0	0
RITX	0101	0	1	0	1
XOR	0110	0	1	1	0
OR	0111	0	1	1	1
NOR	1000	1	0	0	0
XNOR	1001	1	0	0	1
NRITX	1010	1	0	1	0
NRIT	1011	1	0	1	1
NLFTX	1100	1	1	0	0
NLFT	1101	1	1	0	1
NAND	1110	1	1	1	0
TRUE	1111	1	1	1	1

5.1.2 Pyramid Neural Network Structure

To illustrate the process of pyramid model, we use a simple example constructed with three-input DPLM's. Fig. 5.4 shows the structure of the network. In the network, every node (C_{k+1}) in the (k+1)-th layer receives inputs from the 3 neighbors (nodes L_k, C_k, and R_k) in the k-th layer and all nodes in the same layer are assigned the same function. Hence, only one control register is required for each layer. Some functions are illustrated in Table 5.2.

It has to be noted that the CONVERGE and DIVERGE operations are actually the AND and the OR Boolean function, respectively. They are equivalent to the morphological, erosion and dilation, which will be discussed in Section 5.3. The

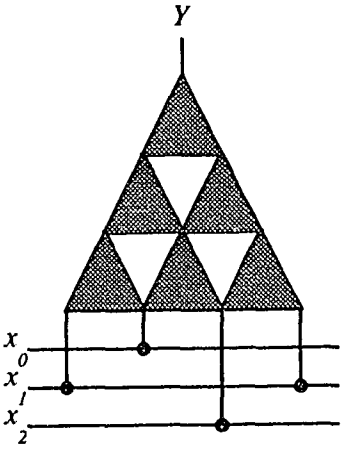


Fig. 5.2 A three-input primitive unit composed of six two-input DPLM's.

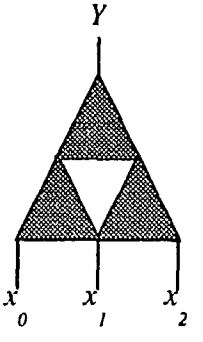


Fig. 5.3 A three-input primitive unit composed of three two-input DPLM's.

RIGHT-ON operation is used to detect the right neighbor and the LEFT-ON operation the left neighbor. Similarly, the CENTER-ON operation detects the center pixel.

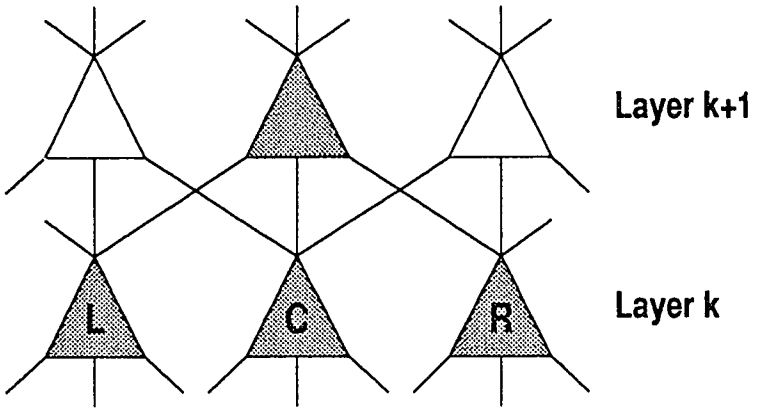


Fig. 5.4 The structure of two-dimensional pyramid neural network composed of three-input DPLM's.

Tab. 5.2 Boolean Functions Performed by Three-Input DPLM's

Function Names	Equations	control	
		upper	lower
CONVERGE	$C_{k+1} \leftarrow L_k\, C_k\, R_k$	AND	AND
DIVERGE	$C_{k+1} \leftarrow L_k + C_k + R_k$	OR	OR
NONE	$C_{k+1} \leftarrow \overline{L_k}\ \overline{C_k}\ \overline{R_k}$	AND	NOR
CENTER-ON	$C_{k+1} \leftarrow C_k$	RITX	LFTX
		LFTX	RITX
RIGHT-ON	$C_{k+1} \leftarrow R_k$	RITX	RITX
LEFT-ON	$C_{k+1} \leftarrow L_k$	LFTX	LFTX

5.2 Binary Morphological Operations by Logic Modules

5.2.1 Binary Morphological Operations

Mathematical morphology operations have been becoming increasingly important in industrial vision applications for object recognition and defect inspection [105]. The essential morphological operations, such as *dilation* and *erosion*, are nonlinear and suited for parallel processing [50]. Dilation is the morphological transformation which combines two sets using vector addition of set elements. If A and B are sets in the n-dimensional Euclidean space (E^n) with elements a and b, respectively, i.e. $a = (a_1, \cdots, a_n)$ and $b = (b_1, \cdots, b_n)$ being n-tuples of element coordinates, the dilation of A by B is the set of all possible vector sums of pairs of elements, one from A and the other from B. Since sets A and B are binary images, the morphological operators applied on the two sets are called *binary morphology*. The *binary dilation* of A by B, denoted by $A \oplus_b B$, is defined as:

$$A \oplus_b B = \{ c \in E^N \mid c = a + b \text{ for some } a \in A \text{ and } b \in B \}. \tag{5.1}$$

The subscript "b" indicates binary. Moreover, the roles of the sets A and B are symmetric. Dilation has a local interpretation. For example, let each point a of A be a seed that grows the flower B_a (by placing the origin of B at a), then the union of all the flowers is the dilation of A by B. The dilation by disk structuring elements corresponds to isotropic expansion algorithm which is popular to binary image processing. Dilation by a small square (3×3) is an 8-neighborhood operation easily implemented by adjacently connected array architectures and is the one known by the name "fill," "expand," or "grow."

Erosion is the morphological dual to dilation. It combines two sets using vector subtraction of set elements. The *binary erosion* of A by B, denoted by $A \ominus_b B$, is defined as:

$$A \ominus_b B = \{ x \in E^N \mid x + b \in A \text{ for every } b \in B \}. \tag{5.2}$$

Erosion does not possess the commutativity property. Erosion $A \ominus_b B$ can be interpreted as the locus of all centers c such that the translation B_c is entirely contained within the set A. One should be aware that erosion is different from Minkowski subtraction [75] in which erosion is the intersection of all translations of A by the elements $b \in B$. On the other hand, dilation is identical to Minkowski addition. Some equivalent terms of erosion are "shrink" and "reduce."

5.2.2 Structure of Morphological Operations using DPLM's

To implement morphological operations, a new neural architecture by the use of the pyramid model is constructed. The connections among the four three-input DPLM's shown in Fig. 5.5 form a nine-input three-dimensional pyramid module. The layers k and $k+1$ are called *regular layer* and the layer $k+\frac{1}{2}$ is called *intermediate layer.*

In the structure of nine-input module, each neuron $U_{k+\frac{1}{2}}(i,j)$ in the intermediate layer $k+\frac{1}{2}$ receives three inputs of its *column-major* neighbors ($U_k(i,j-1)$, $U_k(i,j)$, and $U_k(i,j+1)$) in the preceding layer k. The neuron $U_{k+1}(i,j)$ then collects the three outputs of its three *row-major* neighbors ($U_{k+\frac{1}{2}}(i-1,j)$, $U_{k+\frac{1}{2}}(i,j)$, and $U_{k+\frac{1}{2}}(i+1,j)$) in the preceding intermediate layer $k+\frac{1}{2}$. In other words, the module will extract one-dimensional local features from each column, and then combine the three adjacent 3×1 features into 3×3 two-dimensional local feature. The overall structure of the network is shown in Fig. 5.6. There is no limitation for the number of layers in a neural network, i.e., we could concatenate as many layers as desired and then assign different Boolean functions to the nodes for specific applications.

Since we always apply only one operation on the whole original image at a certain

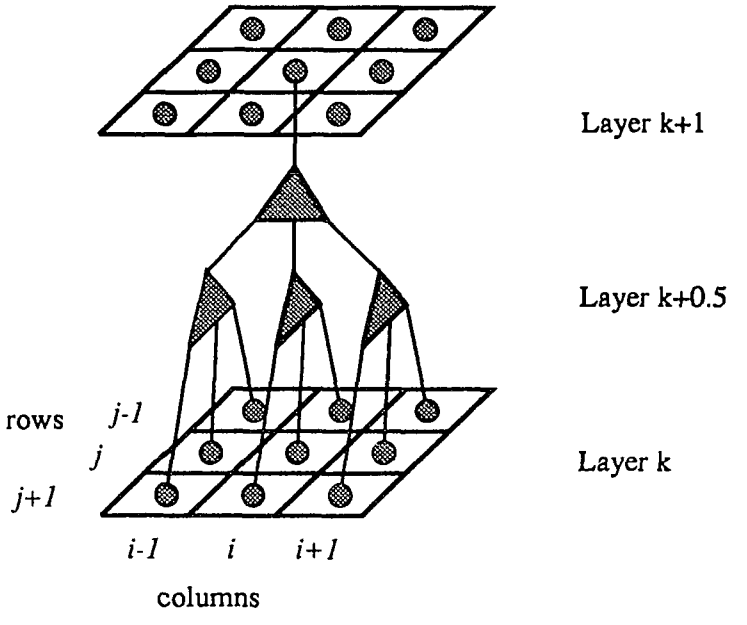


Fig. 5.5 The nine-input pyramid module composed of four three-input DPLM's.

time, loading different control codes to DPLM's on the same major dimension is not required. We connect only two control registers (one for the upper level and the other for the lower level) for each column in any intermediate layer (layer $k+\frac{1}{2}$) as the same as for each row in any regular layer (layer k). That is to apply a certain function with respect to a column or a row. Some examples of morphological operations performed by this neural network for extracting the local features are shown in Table 5.3.

As we can see in the Table, for all dilations, the upper level of the modules should be loaded with the DIVERGE operation, and for all erosions, the CONVERGE operation is employed. The lower level is loaded with different operations for different structuring elements. Suppose we apply an erosion on a binary image with the following 3×3 structuring element:

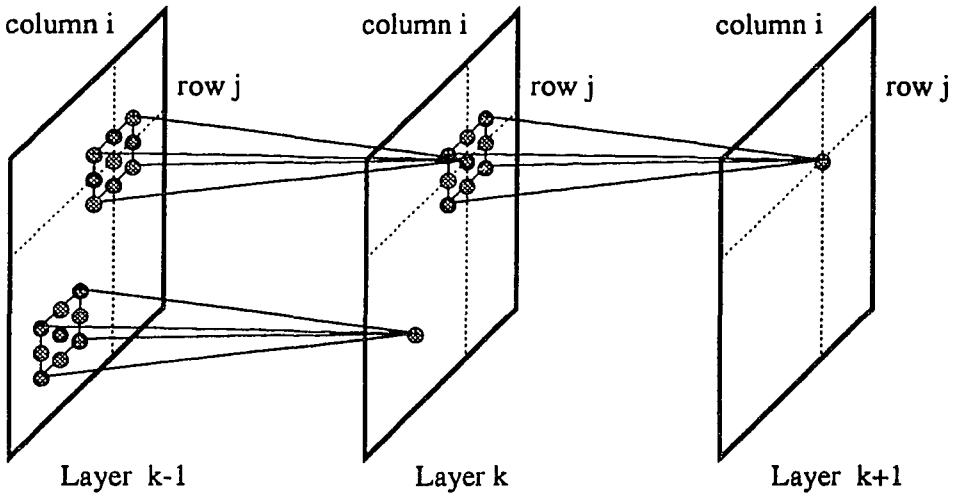


Fig. 5.6 The overall structure of neural network formed with 3×3-input DPLM's for image morphological processing.

$$\begin{matrix} 1\ 1\ 1 \\ 1\ 1\ 1 \\ 1\ 1\ 1 \end{matrix}$$

in the layer k and pass the result to the layer $k+1$. We may simply load CONVERGE to all three-input DPLM's in both intermediate layer $k+\frac{1}{2}$ and regular layer $k+1$. The neurons in layer $k+\frac{1}{2}$ will output a "1" if the three neighbors in column are all ones. The neurons in layer $k+1$ will output a "1" if the three neighboring intermediate neurons in row have all ones, i.e., the nine (3×3) neighboring pixels are all ones. Similarly, for dilation with the same structuring element, we may simply load DIVERGE to all neurons.

If the structuring element is changed to:

$$\begin{matrix} 0\ 0\ 0 \\ 1\ 1\ 1 \\ 0\ 0\ 0 \end{matrix}$$

Tab. 5.3 Morphological Operations Using Pyramid Networks

Structuring Elements	Morphological Operations	Intermediate Layers	Regular Layers
111 111 111	DILATION	DIVERGE	DIVERGE
	EROSION	CONVERGE	CONVERGE
000 111 000	DILATION	CENTER-ON	DIVERGE
	EROSION	CENTER-ON	CONVERGE
010 010 010	DILATION	NONE, CONVERGE, NONE	DIVERGE
	EROSION	NONE, CONVERGE, NONE	CONVERGE
010 111 010	DILATION	CENTER-ON, CONVERGE, CENTER-ON	DIVERGE
	EROSION	CENTER-ON, CONVERGE, CENTER-ON	CONVERGE
100 010 001	DILATION	Left-On, CENTER-ON, Right-On	DIVERGE
	EROSION	Left-On, CENTER-ON, Right-On	CONVERGE
001 010 100	DILATION	Right-On, CENTER-ON, Left-On	DIVERGE
	EROSION	Right-On, CENTER-ON, Left-On	CONVERGE

the erosion requires applying CENTER-ON to neurons in the layer $k+\frac{1}{2}$ and CONVERGE to neurons in layer $k+1$, and the dilation requires applying CENTER-ON to neurons in the layer $k+\frac{1}{2}$ and DIVERGE to neurons in the layer $k+1$. Another example using a different structuring element:

1 0 0
0 1 0
0 0 1

is illustrated as follows. We may load LEFT-ON to the first columns, CENTER-ON to the second columns, and RIGHT-ON to the third columns of each pyramid module in the layer $k+\frac{1}{2}$. At the same time, CONVERGE is loaded in the layer $k+1$. Therefore, any occurrence of line segments with 135° is extracted in the layer $k+1$. Similarly, DIVERGE is also applied to the layer $k+1$ to dilate any occurrence of "1" in the layer k to a diagonal line segment with the length 3.

More complicated neural network could be constructed by extending DPLM's and their connections. By assigning various combinations of Boolean functions, a neural network may act in quite different behaviors. Thus, DPLM networks can be viewed as a class of "digital perceptron" which can support discrete forms of distributed parallel problem solving. They offer a new conceptual framework as well as a promising technology for developing artificial neural networks.

5.3 Gray-Scale Morphological Operations by Tree Models

5.3.1 Gray-Scale Morphological Operations

Mathematical morphology represents image objects as sets in a Euclidean space. In our analysis, the set is the primary notion and a function is viewed as a particular case of a set; e.g., an N-dimensional multi-valued function can be viewed as a set in an $(N+1)$-dimensional space. In this viewpoint then, any function- or set-processing system is viewed as a set mapping from one class of sets into another class of sets. The extensions of the morphological transformations from binary to gray-scale processing by Serra [104], [105], and Sternberg [119] introduce a natural morphological generalization of the dilation and erosion operations. The gray scale dilation and erosion can be computed very fast by using the parallel, logic-gate architecture of threshold decomposition of gray-scale morphology into binary morphology [106].

Let F and K be the domains of the gray-scale image $f(x,y)$ and the gray-scale

structuring element $k(m,n)$, respectively. Since f and k are both gray-scale, the morphological operators applied on these two functions are called *gray-scale morphology*. The *gray-scale dilation* of f by k, denoted by $f \oplus_g k$, is defined as:

$$(f \oplus_g k)(x,y) = \max \{ f(x-m,y-n) + k(m,n) \}, \tag{5.3}$$

for all $(m,n) \in K$ and $(x-m,y-n) \in F$. The *gray-scale erosion* of f by k, denoted by $f \ominus_g k$, is defined as:

$$(f \ominus_g k)(x,y) = \min\{ f(x+m,y+n) - k(m,n) \}, \tag{5.4}$$

for all $(m,n) \in K$ and $(x+m,y+n) \in F$. Note that the subscript "g" indicates gray-scale.

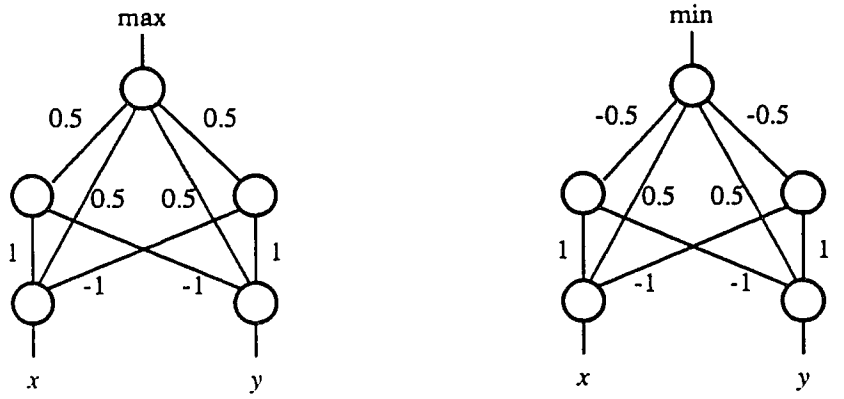


Fig. 5.7 Two comparator subnets which can select the maximum and minimum of two analog inputs x and y at the same time.

5.3.2 Previous Developed Model

Dilation (erosion) is the local maximum (minimum) of an image adding (subtracting) a structuring element. A neural architecture can be used to extract the maximum or minimum. Two comparator subnets which use threshold logic nodes to pick up the maximum or minimum of two analog inputs have been proposed [73]. These nets are suited for the output selected from one of the input samples.

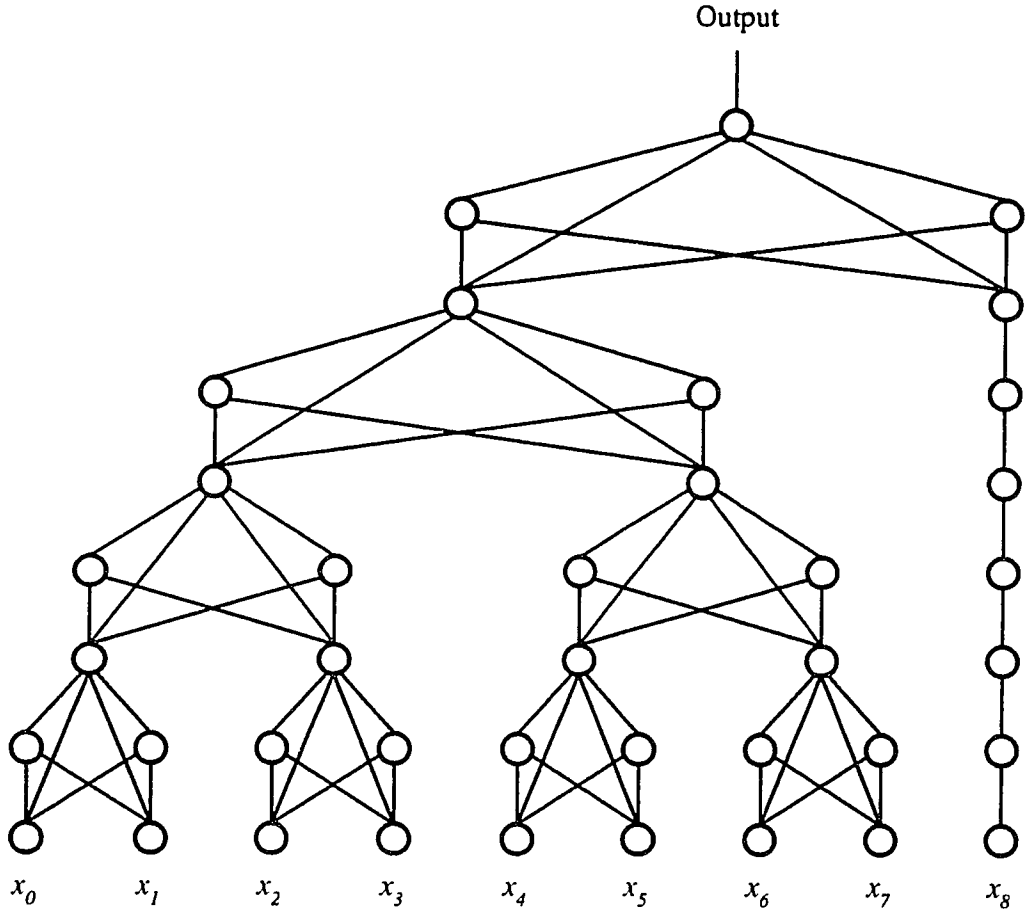


Fig. 5.8 A binary tree structure to pick up the maximum or minimum within a 3×3 mask.

The function of the above bi-comparator subnets can be easily verified. Since all the cells use threshold logic nonlinearity, the output values should be defined as follows:

$$\max = \frac{1}{2}x+\frac{1}{2}y+\frac{1}{2}f_t(x-y)+\frac{1}{2}f_t(y-x) = \frac{1}{2}x+\frac{1}{2}y+\frac{1}{2}|x-y| = \begin{cases} x & \text{if } x \geq y \\ y & \text{if } x < y \end{cases} \quad (5.5)$$

$$\min = \frac{1}{2}x+\frac{1}{2}y-\frac{1}{2}f_t(x-y)-\frac{1}{2}f_t(y-x) = \frac{1}{2}x+\frac{1}{2}y-\frac{1}{2}|x-y| = \begin{cases} x & \text{if } x \leq y \\ y & \text{if } x > y \end{cases} \quad (5.6)$$

where f_t is the threshold logic nonlinearity $f_t(\alpha) = \begin{cases} \alpha & \text{if } x \geq y \\ 0 & \text{if } x < y \end{cases}$.

Comparator subnets can be layered into roughly $\log_2 M$ levels of a binary tree structure to pick up the maximum or minimum within a 3×3 mask and that is illustrated in Fig. 5.8.

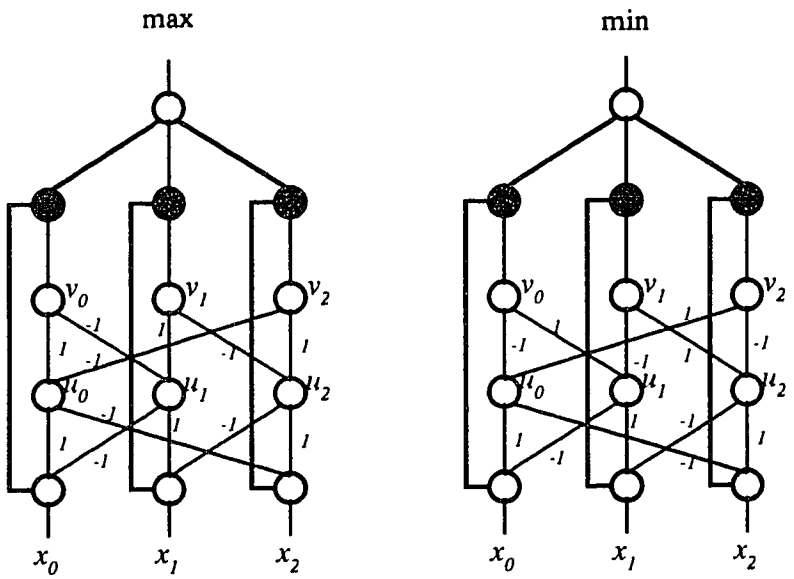


Fig. 5.9 Tri-comparator subnets.

5.3.3 Improvement by Tri-Comparators

The binary tree structure discussed requires eight layers of processing elements. This design will consume much processing time. A better design is to extract the maximum (minimum) of three inputs and then the maximum (minimum) of three selected maxima (minima). Fig. 5.9 shows the tri-comparator subnet for extracting the maximum or minimum of three analog inputs. The black circles indicate the Enable/Disable function and the blank circles denote the simple mathematical function. Let x_i denote the analog inputs, u_i the output neurons in the first hidden layer, and v_i the output in the second hidden layer. Suppose that $f_{h=\theta}$ denotes the hard limiter function with threshold value θ, then we have the following relationship:

$$v_0 = f_{h=0}(u_0 - u_1) = f_{h=0}\left[f_{h=0}(x_0 - x_2) - f_{h=0}(x_1 - x_0)\right], \qquad (5.7)$$

$$v_1 = f_{h=0}(u_1 - u_2) = f_{h=0}\left[f_{h=0}(x_1 - x_0) - f_{h=0}(x_2 - x_1)\right], \tag{5.8}$$

$$v_2 = f_{h=0}(u_2 - u_0) = f_{h=0}\left[f_{h=0}(x_2 - x_1) - f_{h=0}(x_0 - x_2)\right], \tag{5.9}$$

where $f_{h=\theta}(\alpha) = \begin{cases} 1 & \text{if } \alpha > \theta \\ -1 & \text{if } \alpha < \theta \end{cases}$. It can be easily proven that for i, $j = 0$, 1, 2 and $i \neq j$,

$$v_i = \begin{cases} 1 & \text{if } x_i \geq x_j \\ -1 & \text{if } x_i < x_j \end{cases}. \tag{5.10}$$

A feed-forward net which determines the maximum or minimum of nine inputs can be constructed by feeding the outputs of three tri-comparators at the lower layer forward to the higher layers. This forms a triple tree structure shown in Fig. 5.10.

5.3.4 Another Improvement

The triple tree model containing at least six layers of processing elements still takes much time to extract the maximum (minimum) of inputs. Another improved method has been developed to reduce the number of layers and to speed up the neural network.

In order to extract the maximum (or minimum) of n inputs, we need $C(n, 2) = n(n-1)/2$ nodes (say v-nodes) in the first hidden layer to indicate whether any input x_i is greater than or equal to another input x_j. Also, we need n nodes (say u-nodes) in the second hidden layer to indicate if the corresponding input x_i is the maximum (or minimum) of the n inputs. For nine inputs, we need $C(9,2) = 36$ v-nodes in the first hidden layer as shown in Fig. 5.11. The output function of each v-node is defined as follows.

$$v_{ij} = f_{h=0}\left[(1)\, x + (-1)\, x \right] \quad \text{for } i < j. \tag{5.11}$$

These units are used to determine which of two inputs is larger.

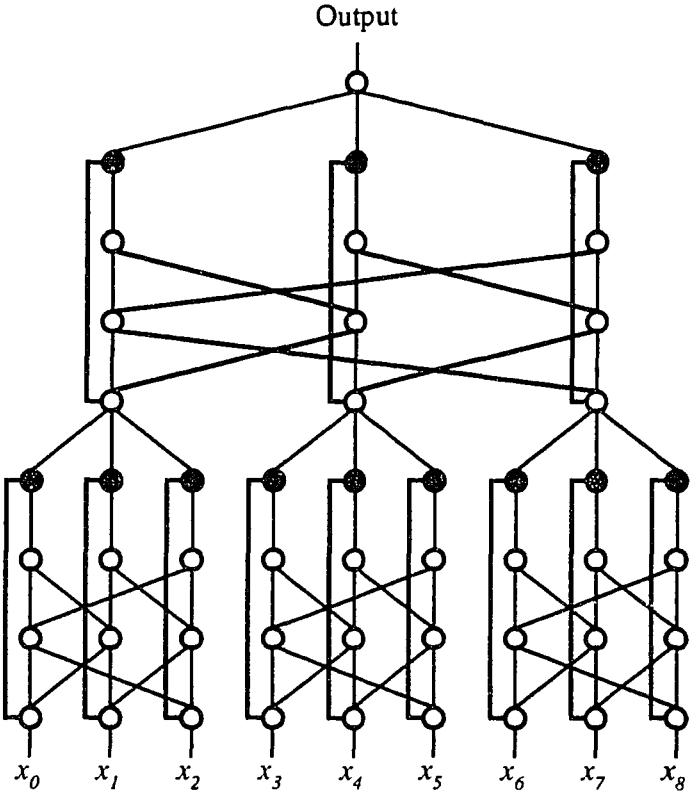


Fig. 5.10 A feed-forward net that determines which of 9 inputs is maximum or minimum using a triple tree structure of comparator subnets shown in Fig. 5.9.

In the second hidden layer, we need nine u-nodes to sum up all the "larger/smaller" information from first hidden layer together to determine whether the particular input is the maximum or minimum. Let $u_i^{\max}$ denote the u-nodes for extracting the maximum and $u_i^{\min}$ for the minimum. Suppose x_i is the maximum of nine inputs, then for all $k \neq i$, the $x_i - x_k$ should be greater than (or equal to) zero. Thus, the following conditions should be satisfied:

$$f_{h=0}(x_i - x_k) = \begin{cases} 1 & \text{for all } k \geq i \\ -1 & \text{for all } k < i \end{cases} \tag{5.12}$$

Also, for the minimum value x_i,

$$f_{h=0}(x_i - x_k) = \begin{cases} -1 & \text{for all } k \geq i \\ 1 & \text{for all } k < i \end{cases} \tag{5.13}$$

The above conditions can be trivially verified since the maximum value must be greater than or equal to any other inputs, and the minimum value must be less than or equal to any other inputs. Therefore, the output function of node $u_i^{\max}$ for extracting the maximum value is defined by:

$$u_i^{\max} = f_{h=7.5}\left[\sum_{k>i} v_{ik} + (-1)\sum_{k<i} v_{ki}\right], \tag{5.14}$$

where $f_{h=7.5}(x) = \begin{cases} 1 & x \geq 7.5 \\ -1 & x < 7.5 \end{cases}$. And the output function of node $u_i^{\min}$ unit for indicating the minimum value is defined by

$$u_i^{\min} = f_{h=7.5}\left[(-1)\sum_{k>i} v_{ik} + \sum_{k<i} v_{ki}\right]. \tag{5.15}$$

Again, we may use transistors at output circuit of nodes in the second layer. The value 1 of $u_i^{\max}$(or $u_i^{\min}$) will "enable" the input x to be conducted to the output of the whole circuit — max or min. In this design, we need no more than three layers of processing elements. Because of the parallelism of processing elements, this circuit can execute much faster than the past models. But the trade-off is the large number of connections and processing elements in the neural network.

5.4 Application of Morphological Operations to Neocognitron

One of the most famous neural network models for visual pattern recognition is "*neocognitron*" which is proposed by Fukushima [36][37][41][42]. It is a multilayered neural network consisting of a cascade of layers of neuron-like cells and can be trained to achieve character recognition.

The hierarchical structure of the network can be found in [42]. There are forward connections between cells in adjoining layers. The input layer, represented by U_0,

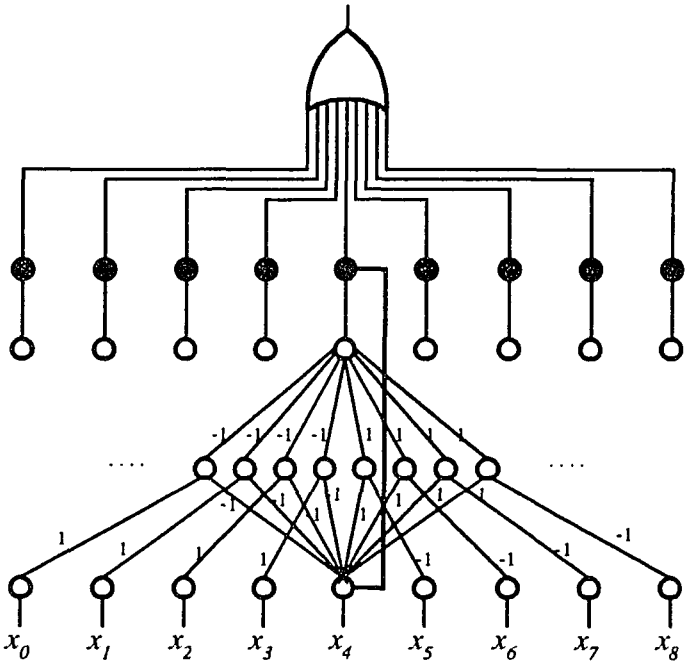


Fig. 5.11 A new design of neural net for extracting the maximum or minimum of nine inputs.

consists of a two-dimensional array of sensoring neurons. Each of the succeeding stages has a layer of "S-cells" followed by a layer of "C-cells." Thus, layers of S-cells and C-cells are arranged alternately in the whole architecture. We use U_{s_l} and U_{c_l} to represent the layers of S-cells and C-cells of the l-th stage, respectively.

According to the activation of the whole architecture, we may say that S-cells are feature-extracting cells. Connection weights converging to feature-extracting S-cells are variable and are reinforced during the learning phase. Generally speaking, in the lower stages, local features, such as a particular orientation, are extracted, and in the higher stages, more global features, such as a longer line segment or a part of input pattern, are extracted. The use of C-cells are allowing positional errors in the features of the stimulus, which defines the acceptable tolerance range. Connection weights from S-cells to C-cells are fixed and unchangeable. Each C-cell receives signals from a group of S-cells

which extract the same feature from slightly different positions. The C-cell is activated if at least one of these S-cells is active (same as Boolean OR operation). Hence, the C-cell's response is less sensitive to translation of the input patterns.

In the first stage, the function of S-cells can be considered as erosion operations using the twelve different local features [41] as structuring elements. And the function of C-cells is the dilation operation with some other structuring elements which are orthogonal to the related local features. That's the reason why U_C layers may response to the same pattern with small translation. It can be verified by the following simple example shown in Fig. 5.12.

To simplify the problem, suppose the input patterns are 5×5 binary images and will be presented in the input layer U_0. In the first stage, there are two 5×5 U_{s_1}-cell planes: one for extracting a vertical line segment and the other for horizontal line segment, and two 7×7 U_{c_1} planes where each of them corresponds to either U_{s_1}-cell planes. Each U_{s_1}-cell receives the output signals from the eight neighboring pixels in the input layer. The U_{c_1} cells can be active if any U_{s_1} cell in the neighborhood is activated. In the example, since there exists a vertical line segment going through the three pixels $U_0(1,2)$, $U_0(2,2)$, and $U_0(3,2)$, the $U_0(2,2)$ is activated and $U_{c_1}(2,1)$, $U_{c_1}(2,2)$, and $U_{c_1}(2,3)$ are all activated.

In the second stage, we use four 3×3 U_{s_2} cell-planes to extract corners of four different orientation with 5×5 structuring elements. According to the overall architecture, the connections from U_{c_1} to U_{s_2} It is obvious that the 3×3 pyramid DPLM shown in Fig. 5.5 is not able to handle such kind of operations. A more complicated structure, which is not going to be discussed in detailed, should be employed in this stage.

The same situation is happened in the third stage. There are four single U_{c_2} cells used to indicate the orientation of the input pattern. Each of these neurons has

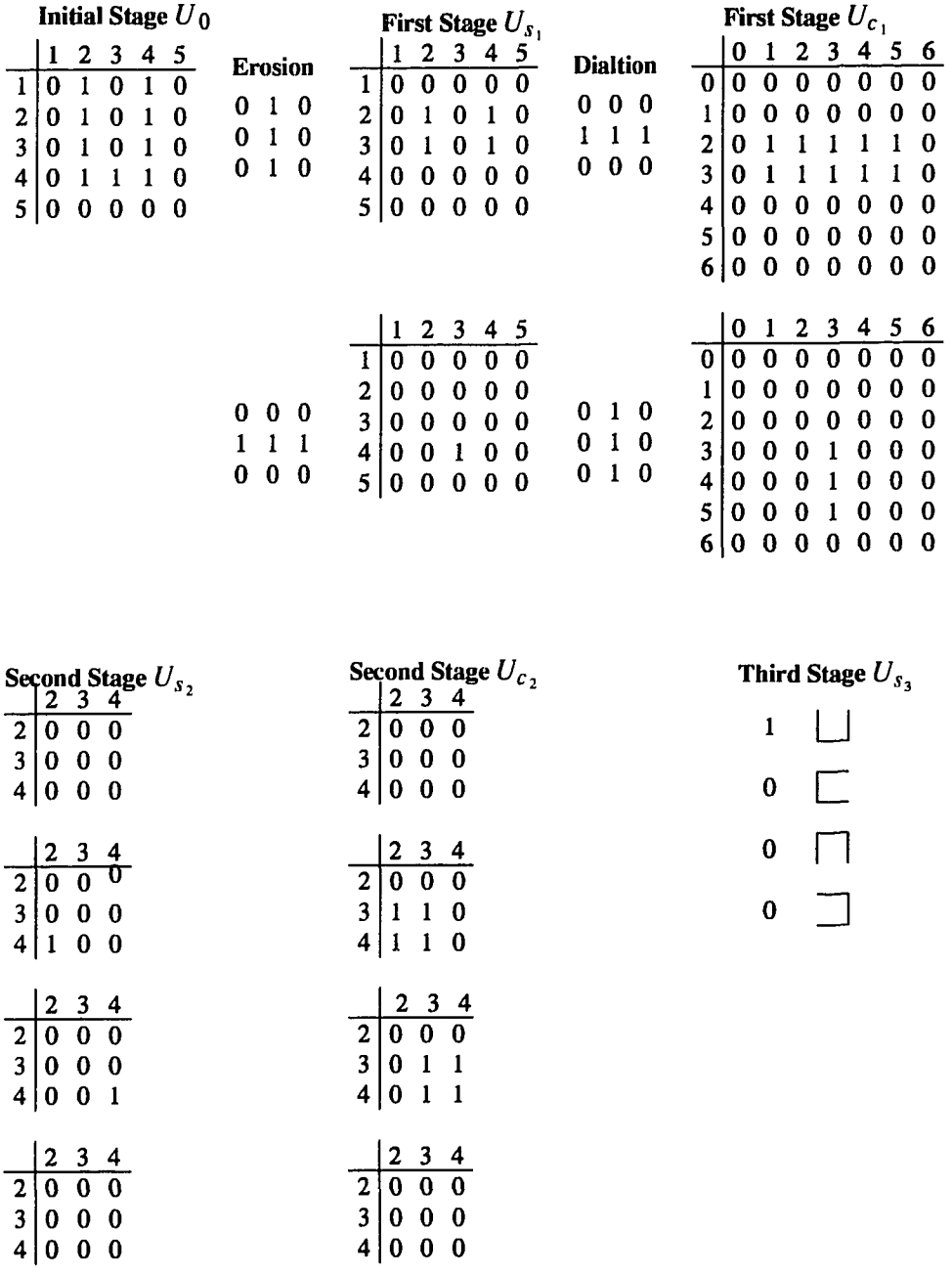


Fig. 5.12 A simple example illustrating the activation of neocognitron.

connections from the four cell planes of U_{C_2} layer, which results in sophisticated connections. After training (i.e. loading appropriate control codes), this simplified neural network can be used to recognize the orientation of any U-like pattern of different sizes.

5.5 Conclusion

Neural networks are a good choice for implementation of parallel hardware because of their parallelism. Due to the incredible improvement of hardware technology, neural network models not only were simulated on conventional computers but also have been implemented in prototype circuitries. This field is a re-discovered area experiencing an explosive growth in research and application interests. Algorithms and architectures hence proliferate. Our research is aimed at analyzing learning algorithms and parallel architectures by the use of neural networks, at building a complete system for image processing and analysis, and at determining the modification of traditional algorithms implemented by neuron-like components. Advances in these areas and in VLSI techniques could lead to practical real-time systems.

CHAPTER 6

ONE-PASS TRAINING ALGORITHMS FOR MORPHOLOGICAL OPERATIONS

Mathematical morphology [50][104] has been becoming increasingly important in industrial vision applications for object recognition and defect inspection. Two fundamental operations, *dilation* and *erosion*, adopt a structuring element as a probe to interact with an active image for features extraction. The operations are nonlinear and suitable for parallel implementation because the new value of each pixel is computed based on the current values of its neighborhood and is independent of the new pixel values. A variety of architectures for morphological operations have been developed [1][22][103], and among them the parallel and neural network implementations are the most popular.

Davidson and Hummer [22] developed a weighted-sum activation function to implement the gray-scale dilation. However, their architecture requires an iterative training algorithm to achieve the convergence of connection weights. Shih and Moh [112] used a programmable logic circuit composed of multiplexers and registers to carry out binary and gray-scale morphological operations. Morales and Ko [83] proposed an iterative training algorithm and used an overall equality index related to fuzzy implication as a performance index for convergence. However, the index computation consumes much time to complete.

Mathematical morphology can be combined with fuzzy sets [141] to deal with the fuzziness of digital images. Wu [138] developed fuzzy mathematical morphology associated with order statistics. Sinha and Dougherty [116] used a fitting paradigm which employs an index for set inclusion to measure the degree to which the structuring element is beneath the active image for fuzzy morphological erosion.

6.1 Theoretical Foundations of Mathematical and Fuzzy Morphology

In binary mathematical morphology, the *erosion* of an image A by a structuring element B is defined as

$$A \ominus B = \bigcap_{x \in -B} \mathbf{T}(A;x), \tag{6.1}$$

where $\mathbf{T}(A;x)$ is the *translation* of A by a vector $x \in U$ (U denotes the Cartesian grid or the Euclidean plane for binary images.) and "$-B$" denotes the reflection of B [104]. Equivalently, we may write

$$A \ominus B = \{ x \mid \mathbf{T}(B;x) \subseteq A \}. \tag{6.2}$$

The *dilation* of an image A by a structuring element B is defined as

$$A \oplus B = \bigcup_{x \in B} \mathbf{T}(A;x) = \bigcup_{x \in A} \mathbf{T}(B;x). \tag{6.3}$$

Equivalently, we may write

$$A \oplus B = \{ x \mid \mathbf{T}(-B;x) \cap A \neq \phi \}. \tag{6.4}$$

In gray-scale morphology we can assume that for every $x \in F$; the surface values $\{y \mid (x,y) \in A\}$ are topologically closed and also the sets F and K are finite. For any function $f(k)$ defined on some subset $F(K)$ of Euclidean $(N-1)$-space the umbra of $f(k)$ is a set consisting of the surface $f(k)$ and everything below the surface. Let $f: F \rightarrow E$ and $k: K \rightarrow E$. Then *gray-scale dilation* $f \oplus_g k: F \oplus_g K \rightarrow E$ can be computed by

$$f \oplus_g k(x) = \mathbf{max} \{ f(x-z) + k(z) \}, \text{ for all } z \in K \text{ and } x - z \in F, \tag{6.5}$$

and *gray-scale erosion* $f \ominus_g k: F \ominus_g K \rightarrow E$ can be computed by

$$f \ominus_g k(x) = \mathbf{min} \{ f(x+z) - k(z) \}, \text{ for all } z \in K \text{ and } x + z \in F. \tag{6.6}$$

The morphological hit-or-miss transform is a basic tool to select out pixels which have certain geometric properties and that performs template matching, shape detection, thinning, and thickening. Let $B = (B_1, B_2)$, where B_1 is the set formed from elements of B associated with an object, and B_2 is the set of elements of B associated with the background. The *hit-or-miss* operation is defined as

$$A \circledast B = (A \setminus B_1) \cap (A^c \setminus B_2) . \tag{6.7}$$

where "\" denotes the set difference and A^c denote the set complement of A.

In fuzzy mathematical morphology [116], an index function is used for set inclusion and is defined as

$$\mathbf{I}(A,B) = \inf_{x \in U} \mu_{A^c \Delta B}(x) , \tag{6.8}$$

where "Δ" denotes the *bold union* of two sets, which is defined as $\mu_{X \Delta Y}(z) = \mathbf{min}$ [1, $\mu_X(z) + \mu_Y(z)$]. The fuzzy complement of the set inclusion index is defined as

$$\mathbf{I}^c(A,B) = 1 - \mathbf{I}(A,B) = \sup_{x \in U} \mu_{A \nabla B^c}(x) , \tag{6.9}$$

where "∇" denotes the *bold intersection* of two sets, which is defined as $\mu_{X \nabla Y}(z) = \mathbf{max}$ [0, $\mu_X(z) + \mu_Y(z) - 1$]. Based on eqs. (8) and (9), the *fuzzy erosion* is defined as

$$\mu_{E(A,B)}(x) = \mathbf{I}(\mathbf{T}(B;x), A) = \inf_{x \in U} \mathbf{min} \left[1, 1 + \mu_A(x) - \mu_{T(B;x)}(x) \right] , \tag{6.10}$$

and the *fuzzy dilation* is defined as

$$\mu_{D(A,B)}(x) = 1 - \mathbf{I}(\mathbf{T}(-B;x), A^c) = \sup_{x \in U} \mathbf{max} \left[0, \mu_{T(-B;x)}(x) + \mu_A(x) - 1 \right] . \tag{6.11}$$

The property of being dual operations between fuzzy erosion and fuzzy dilation is preserved. That is

$$\mu_{D(A,B)}(x) = \mu_{E(A^c,-B)^c}(x)\,. \tag{6.12}$$

We present one-pass algorithms to speed up the operations of hit-or-miss, binary, gray-scale, and fuzzy dilations and erosions. The algorithms can achieve the training phase of neurons in a single iteration and use simple computations in the testing phase. In this correspondence, Section 2 presents the one-pass training algorithms for binary morphological operations. Section 3 describes the construction of fuzzy and gray-scale morphological neurons. Section 4 illustrates the results of software-simulated experiments. Finally, conclusions are made.

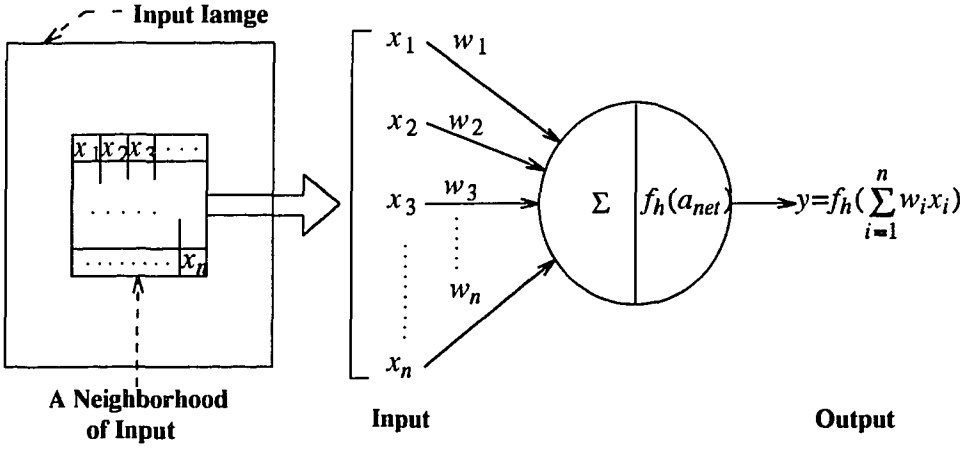


Fig. 6.1 The structure of a binary morphological neuron. The symbol "Σ" denotes that the activation value is defined as $a_{net} = \sum_{i=1}^{n} w_i\, x_i$.

6.2 Binary Morphological Neurons

In morphological operations, the structuring element is translated to each pixel of the active image and perform computations with the corresponding pixels within the windows of the translated structuring element. As shown in Fig. 6.1, the input vector, $\vec{X} = (x_1, x_2, \cdots, x_n)$, denotes the corresponding pixels in a binary image, and the

weighting vector, $\vec{W} = (w_1, w_2, \cdots, w_n)$, denotes the weights in the connections which reflect the structuring element. The net input is obtained by summing up these weighted inputs, i.e. the inner product of $\vec{W}$ and $\vec{X}^T$, as

$$a_{net} = \vec{W} \cdot \vec{X}^T = \sum_{i=1}^{n} w_i x_i , \tag{6.13}$$

where the superscript T denotes the transpose of a vector. The threshold function is defined as

$$f_h(\alpha) = \begin{cases} 1 & \alpha \geq \theta \\ 0 & \alpha < \theta \end{cases}, \tag{6.14}$$

where θ is the threshold value. In the training phase, the structuring element is used as the training vector. Each neuron adjusts the weights according to an operation-oriented training algorithm to learn the training vector. In other words, the structuring element is embedded into the weights oriented toward a desired morphological operation. To avoid the time-consuming iterative training, we design one-pass training algorithms so that the neuron can learn and memorize the training vector in just one step.

One-Pass Hit-or-Miss Training Algorithm

(1) Present the training vector, $\vec{T} = (t_1, t_2, \cdots, t_n)$, to the input connections.

(2) Calculate weights w_i for all the connections by converting binary t_i into bipolar and performing normalization by the total number of elements n as

$$w_i = \frac{2t_i - 1}{n} . \tag{6.15}$$

(3) Select the threshold value θ by the weighted sum of the training vector.

$$\theta = \sum_{i=1}^{n} w_i t_i = \frac{1}{n} \sum_{i=1}^{n} t_i . \tag{6.16}$$

For the morphological hit-or-miss operations, the input vector must match exactly with the learned vector (i.e. the structuring element), so that the net input equals to the threshold value. Any mismatched element will produce $-1/n$ to reduce the net input.

[Proof]: Assume there are n_1 elements with value 1 in the training vector. According to the training algorithm, $\theta = n_1/n$. For testing, if the input vector is the same as the training vector, the net input is

$$a_{net} = \sum_{i=1}^{n} w_i t_i = n_1 \cdot 1 \cdot (1/n) + (n - n_1) \cdot 0 \cdot (-(1/n)) = n_1/n \ . \qquad (6.17)$$

Therefore, the output is 1. If the input vector differs from the training vector in at least one element, the net input will be less than n_1/n. Therefore, the output is 0. The following two cases are illustrated for one-element difference:

(i) If $t_i = 1$ and $x_i = 0$, the corresponding weight is trained to be $1/n$ and the net input becomes $a_{net} = (n_1 - 1)/n$.

(ii) If $t_i = 0$ and $x_i = 1$, the corresponding weight is trained to be $-1/n$ and the net input becomes $a_{net} = (n_1 - 1)/n$. □

One-Pass Dilation Training Algorithm

(1) Present the training pattern, $\vec{T} = (t_n, t_{n-1}, \cdots, t_2, t_1)$, to the input connections. Note that according to eq.(6.2) the sequence of components in the structuring element is reversed.

(2) Calculate weights w_i for all the connections by performing normalization by the total number of elements n. That is, $w_i = t_i/n$.

(3) Select the threshold value θ by the reciprocal of the total number of elements. That is, $\theta = 1/n$.

[Proof]: According to the definition of morphological dilation (eq.(6.4)), if there

exists at least one pixel of value 1 in the window which matches with the component of value 1 in the corresponding structuring element, then the output is 1. Therefore, in this case, the net input is $a_{net} \geq 1/n$, so that the output of the neuron is 1. Otherwise, there does not exist any match of value 1 between the pixel and the component of the structuring element, so that output is 0. □

One-Pass Erosion Training Algorithm

(1) Present the training pattern, $\vec{T} = (t_1, t_2, \cdots, t_n)$, to the input connections.

(2) Calculate weights w_i for all the connections by performing normalization by the total number of elements n. That is, $w_i = t_i / n$.

(3) Set the threshold value θ by the weighted sum of the training vector.

$$\theta = \sum_{i=1}^{n} w_i t_i = \frac{1}{n} \sum_{i=1}^{n} t_i \tag{6.18}$$

[Proof]: Assume there are n_1 elements with value 1 in the training vector. The threshold value is set to n_1/n. According to the definition of morphological erosion (eq.(6.2)), if there exists at least one pixel of value 1 in the window which does not match with the component of value 1 in the corresponding structuring element, then the net input will be less than n_1/n. Therefore, the output is 0. Otherwise, all the components of the structuring element are matched by the input pixels of value 1, so that the output is 1. □

6.3 Fuzzy Morphological Neurons

In this section, we develop one-pass training algorithms for fuzzy morphological operations. Fig. 6.2 shows the architecture for fuzzy erosion. The neurons add an input element and its corresponding weight, and then employ a saturated linear activation function as

$$f_t(\alpha) = \begin{cases} 1 & \alpha \geq 1 \\ \alpha & 0 < \alpha < 1 \\ 0 & \alpha \leq 0 \end{cases} \tag{6.19}$$

One-Pass Fuzzy Erosion Training Algorithm

(1) Present the training pattern, $\vec{T} = (t_1, t_2, \cdots, t_n)$, to the input connections.

(2) Calculate weights w_i for all the connections in the first layer by $w_i = 1 - t_i$.

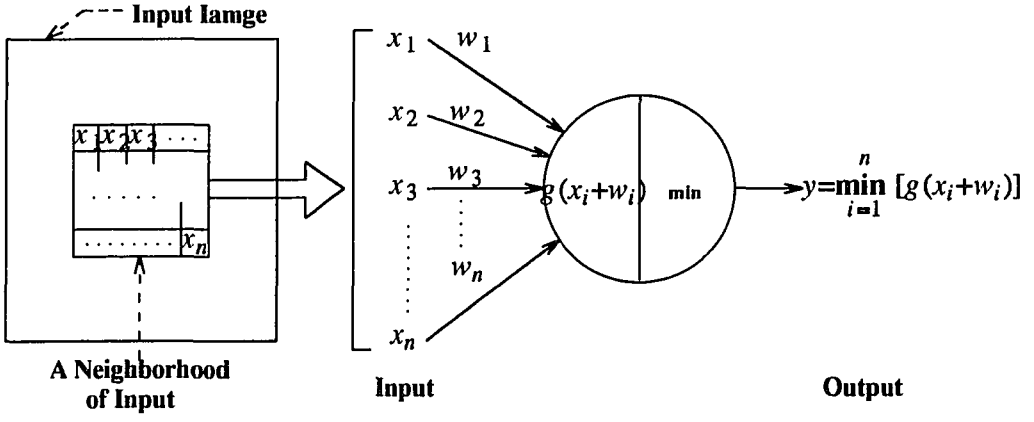


Fig. 6.2 The structure of a fuzzy erosion neuron in which "g" denotes the saturated linear activation function for individual additive inputs.

When an input vector, $\vec{X} = (x_1, x_2, \cdots, x_n)$, is presented to the neuron in Fig. 6.2, each neuron adds its corresponding weight, i.e. $w_i = 1 - t_i$, and then passes through

the saturated linear activation function in eq.(6.19). Therefore, we obtain

$$f_t(x_i + w_i) = \min_{i=1}^{n} \left[1 + x_i - t_i \right] . \tag{6.20}$$

The neural architecture of fuzzy dilation is similar to that shown in Fig. 6.2, except that the order of elements in the training pattern is reversed and the minimum is replaced by the maximum.

One-Pass Fuzzy Dilation Training Algorithm

(1) Present the training pattern, $\vec{T} = (t_n, t_{n-1}, \cdots , t_2, t_1)$, to the input connections.

(2) Calculate weights w_i for all the connections in the first layer by $w_i = t_i - 1$.

When an input vector, $\vec{X} = (x_1, x_2, \cdots , x_n)$, is presented, each neuron add its corresponding weight, $w_i = t_i - 1$, and then passes through the saturated linear activation function in eq.(6.19). Therefore, we obtain

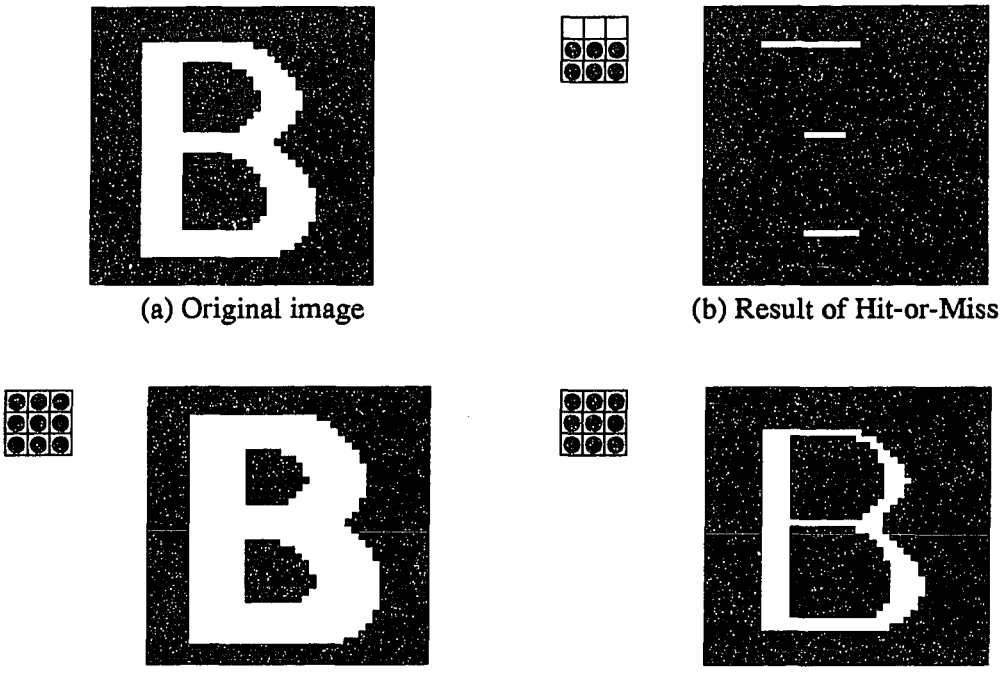
$$f_t(x_i + w_i) = \max_{i=1}^{n} \left[x_i + t_i - 1 \right] . \tag{6.21}$$

Note that the one-pass training algorithms for gray-scale erosion and dilation are similar to the ones for fuzzy erosion and dilation except that the weights w_i is replaced by $-t_i$ and t_i, respectively.

6.4 Experimental Results

We simulated the proposed neurons and training algorithms in software. Fig. 6.3(a) shows a character image "**B**." Fig. 6.3(b) illustrates the structuring element for extracting the upper edges and the image after a hit-or-miss is applied. Figs. 6.3(c) and 6.3(d) show dilated and eroded images respectively associated with 3×3 structuring element. In Fig. 6.4, we illustrate the experimental results of fuzzy dilation and erosion.

The original image is shown in Fig. 6.4(a) with 256 gray levels. Fig. 6.4(b) shows the added Gaussian noisy image with a standard deviation 20 and mean value 0. Figs. 6.4(c) and 6.4(d) show the images after a fuzzy dilation and a fuzzy erosion are respectively applied with a flat structuring element.



(a) Original image (b) Result of Hit-or-Miss

(c) Result of Dilation (d) Result of Erosion

Fig. 6.3 Software-simulated results of morphological neurons trained with the corresponding structuring elements for hit-or-miss, binary dilation, and binary erosion.

6.5 Conclusions

The proposed architectures which consist of simple computations are easy in hardware implementation with today's VLSI technology. Besides, the one-pass training algorithms speed up the time in the learning phase and can be easily adapted for variant morphological operations.

(a) Original image

(b)Added with Gaussian noise

$$\frac{1}{256}\begin{bmatrix} 205 & 205 & 205 \\ 205 & 205 & 205 \\ 205 & 205 & 205 \end{bmatrix}$$

$$\frac{1}{256}\begin{bmatrix} 205 & 205 & 205 \\ 205 & 205 & 205 \\ 205 & 205 & 205 \end{bmatrix}$$

(c) Result of fuzzy dilation

(d) Result of fuzzy erosion

Fig. 6.4 Software-simulated results of fuzzy morphological neurons trained with the corresponding structuring elements for fuzzy dilation, and fuzzy erosion.

CHAPTER 7

A NEW NEURAL NETWORK ARCHITECTURE FOR PERSONAL HANDWRITTEN RECOGNITION

Neural networks have been widely studied in a number of fields, such as neural architectures, neurobiology, statistics of neural network and pattern classification. In the field of pattern classification, neural network models are applied on numerous applications, for instance, character recognition, speech recognition, and object recognition. Among these, character recognition is commonly used to illustrate the feature and classification characteristics of neural networks.

Handwritten character recognition can be categorized into *on-line* and *off-line*. The former is to recognize characters as being written on an input device, while the latter carried out on a previously written paper. On-line systems are capable of capturing temporal and dynamic information of the writing, such as number and order of the strokes, direction and speed of the writing within each stroke, etc. Off-line systems, however, require conversion from scanned data to line drawings which is usually costly and faulty. However, there are numerous real world applications which require recognition of handwritten documents. The demand of off-line recognition systems in terms of accuracy and speed is becoming increasingly.

Commonly used approaches adopt conventional feature extraction algorithms and feed the features to an existing neural network classifier. In [70], Krzyzak *et al.* used four topological features and 30 Fourier descriptors as the feature pattern, and a multilayered perceptron (MLP) with the modified back-propagation learning algorithm for classification. Kageyu *et al.* [61] used the geometrical invariants with complex-log mapping and Fourier transform and an augmented three-layer perceptron. Agui *et al.* [2] used the moment invariants and a three-layer perceptron for the recognition of Katakana

characters. For Chinese character recognition, Liao *et al.* [71] proposed an iterative tracing process to extract the end points in skeleton and to classify each segment as either a curve or a straight line. The extracted strokes are then loaded into a Hopfield neural network for matching against the characters in the database. Idan and Chevallier [57] applied a supervised Kohonen's model to handwritten zip code recognition.

Several special neural network models have been proposed to fit the recognition task. In [40] [32] [44], Fukushima proposed a hierarchical neural network, named *neocognitron*, for handwritten characters recognition. Modified neocognitrons were also proposed in [124] by coupling the neocognitron and the perceptron, in [86] by modifying the U_C neuron activation equation and the training algorithm for weight adaptation, and in [33] by improving with bend-detecting cells. Iwata *et al.* [58] proposed a large scale neural network with comb structure, named "CombNET-II" trained with back propagation algorithm. Deco and Blasig [24] introduced a neural architecture with a learning paradigm using Radial Basis Functions (RBF) and Principal Component Analysis (PCA) for handwritten digit recognition. Sziranyi and Csicsvari [125] proposed a dual cellular neural network architecture (CNND) to extract shadow templates of different orientations and holes for classification. In [127], Thepaut and Autret combined two neural networks for handwritten digit recognition: one using digits as images and the other using digits as contours, and showed that better results can be obtained rather than individual neural networks.

The methodology of dividing an image into several parts, extracting local features in individual parts, and performing recognition based on the contributions from all parts has been introduced. Yoshida [140] applied an unsupervised learning process combined with the Hebb rule to a neocognitron-like architecture for extracting features. He also discussed the optimization of the self-organized features. Zhu *et al.* [142] proposed a layered neural network with a feature-detecting subnet and a recognizing subnet. The function of the feature-detecting subnet is to look for the existence of features, such as

end point, cross point, oblique line, and to represent in a grid of a relatively low resolution, say 3×3, indicating that the feature is located at which part of the image. The error back-propagation learning is adopted in the recognition subnet. Suen *et al.* [123] evaluated the recognition rates of extracting features from distinctive parts of an image. A multi-stage neural network architecture was proposed by Jouny and Sheridan [60]. The input image is divided into two slightly overlapping segments for two multilayer neural networks, and then classified with a third multilayer neural network taking the output patterns of the two networks. Khobragade and Ray [63] presented a new non-adaptive connectionist architecture based feature extractor for alphabetic patterns. The extracted feature patterns are then loaded to a neural network as a pattern classifier for determining the characters. The frame division concept may inspire more useful and meaningful feature extraction idea. For Kanji character recognition, Togawa *et al.* [129] proposed a model of a receptive field neural network as a discriminator. The receptive fields are determined by obtaining high contribution values of features at a certain location for a set of similar Kanji characters.

Syntactic analysis is also incorporated with character recognition to achieve higher recognition rates. Jaravine [59] presented a syntactic neural network constructed by syntactic neurons together with 2-D dictionary represented as quadro-tree. In [130], Tsunoda *et al.* extracted features with Kirsh typed mask and applied the ART1 classifier for individual character recognition. Spelling check is incorporated into the recognition system to ensure a high recognition performance. Background analysis is also used in [126], where Tascini *et al.* proposed a low-cost handwritten character recognizer with six steps: character dilation, character circumscription, region and profile analysis, cut analysis, decision tree descent, and result validation.

How does the human brain recognize different characters? Which parts of the characters is it looking for? What kinds of features do we get when we see a character? Is there some way we can extract character recognition knowledge from humans, who

have become such superb character recognizers? These are the common questions asked when one is dealing with character recognition problems. In this chapter, we introduce a new and efficient neural architecture specially designed to learn and recognize characters written by a single user or a reasonably small group of persons. This is a prototype of personal handwriting recognition system. Multiple neural network models are utilized in the architecture in order to fit the required task in every stage of a recognition system and to achieve higher performance and efficiency.

This chapter is divided into six sections. Section 7.1 depicts the overall architecture of the recognition system. Section 7.2 elaborates the morphological subnet designed for feature preprocessing. Examples of subnet activation are also provided. The structure and training algorithm for the feature extraction and training subnet is described in Section 7.3. In Section 7.4 we discuss the pattern classification subnet which determines the characters according to the extracted features. The experimental results of handwritten character recognition are illustrated in Section 7.5. Section 7.6 concludes the chapter and points out some directions for future research.

7.1 System Architecture

The overall architecture of the personal handwritten character recognition system is shown in Fig. 7.1. The system consists of three modules: morphological feature preprocessing module, stroke extraction module, and pattern classification module.

Input character images are first loaded into the morphological feature preprocessing module containing a multiple layer of morphological neurons. The function of this module is to filter the input image with structuring elements of different orientations and to separate strokes of different directions. Only short line segments with three consecutive foreground pixels are considered in each layer. The layers are arranged so that erosion and dilation are applied to the image alternatively. Lower layers are to sieve

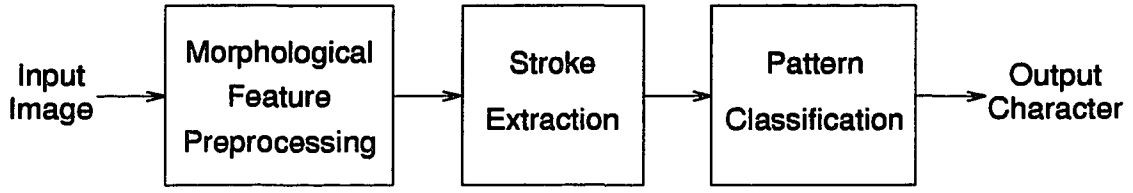


Fig. 7.1 The overall architecture of proposed handwritten character recognizer.

local features (or short line segments) while higher layers are to extract more global features (or longer line segments) in different orientations. Therefore, the output of this module is composed of the locations of long line segments in eight different orientations such as 0°, 22.5°, 45°, 67.5°, 90°, 112.5°, 135°, and 157.5°. Morphological dilation is applied alternatively to allow a certain limit of position and scaling errors. Section 7.2 elaborates the detailed implementation of the module.

The second module, stroke extraction module, is designed to extract more heuristic features for human being. These strokes includes vertical strokes, horizontal strokes, positive slope strokes, negative slope strokes, and vertical and horizontal curves at different relative locations and with different lengths. Each neuron in this module is connected to certain parts of the output vector from the first module depending on the stroke to be extracted. The output values of the neurons indicate the existence of the strokes in the input image. The delta learning rule proposed by Rumelhart *et al.* [96] is adopted to train the connection weights of such neurons. Variations of the same stroke can also be learned with such a training algorithm. We shall discuss the implementation details of this module in Section 7.3.

The stroke vector output from module 2 is now ready for classification. This is done with a higher level of neural network module, named pattern classification module. Any neural network model designed for pattern classification can be used. The possible

candidates considered in our architecture include ART-1 [12] and multilayer perceptron [73]. In Section 7.4, we depict how the models can be incorporated into our architecture.

7.2 Feature Preprocessing Using Morphological Neurons

The first stage in neocognitron [40], including U_{S1} and U_{C1}, is designed for local feature extraction. The U_{S1} neurons are trained with (3-pixel-long) short line segments of different orientations. There are a total of eight different orientations with twelve pattern variations to be detected. The output of U_{S1} indicates the relative location of line segments found in the image. U_{C1} neurons are hard coded with sets of weight patterns to blur the detected location in order to allow positional errors in the extracted features. From our study, we find out that U_{S1} and U_{C1} are actually performing operations similar to morphological erosion and dilation, respectively.

The implementation of neocognitron requires a number of training iterations regardless supervised or unsupervised training. This may cost too much in hardware implementation and takes longer time in the training phase. Instead, pure morphological operations have only simple computations which can be easily coded in hardware. The morphological neuron and the related one-pass training algorithms proposed in [84] are plausible to the task of local feature extraction.

In this section, we depict a hierarchical architecture similar to neocognitron but using morphological neurons as processing elements. The connection topology between neuron planes is illustrated in Fig. 7.2. The architecture consists of several layers of neuron planes. The three dimensional numbers at the bottom indicate the number of neurons in each neuron plane and the number of neuron planes in each layer. For instance, both layers 1 and 2 have 11×11 neurons in each of the 12 neuron planes, while both layers 3 and 4 are respectively composed of 12 and 8 neuron planes with size of 5×5. Fig. 7.3 illustrates concretely how the neurons of each neuron plane are

interconnected to the neurons of other neuron planes. The highlighted lines show the connections from the neurons in the receptive field for the corresponding shaded neurons.

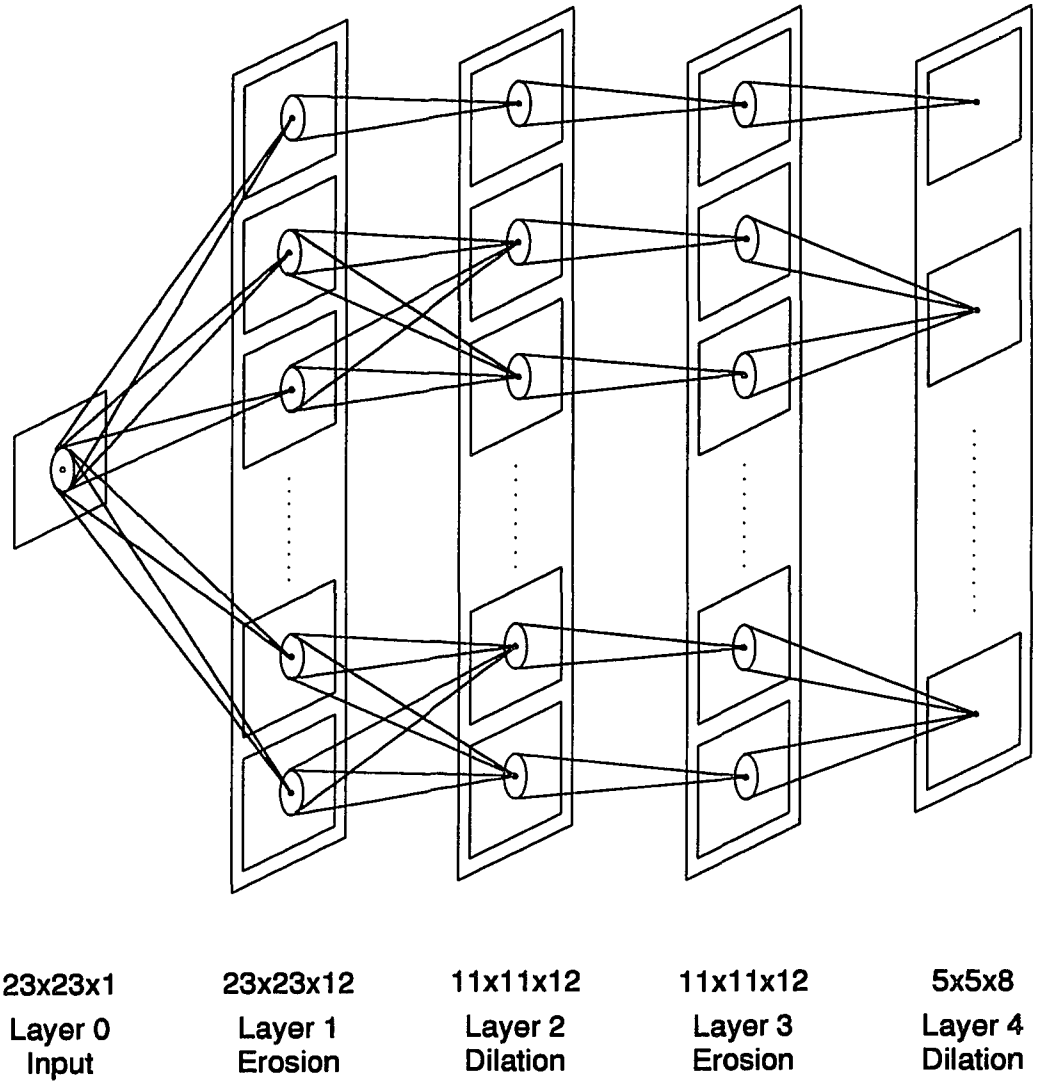


Fig. 7.2 Schematic diagram showing syntactic connections between layers in the proposed feature preprocessor. The circles represent the receptive fields for neurons at the apexes.

The input image is first normalized into 23x23. The parallel thinning algorithm proposed in [115] is used to obtain the one-pixel-wide strokes. The skeleton of the

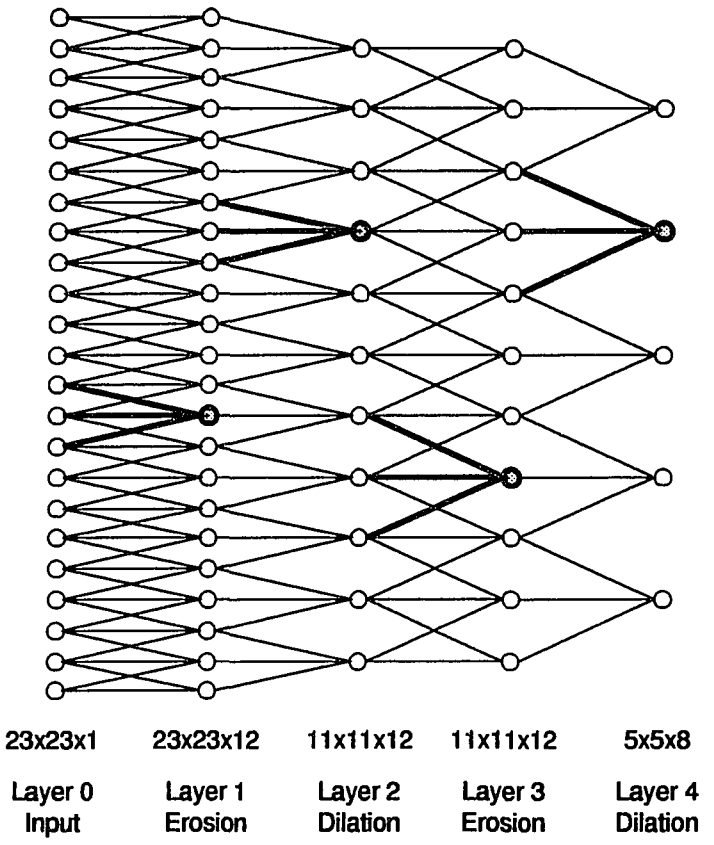


Fig. 7.3 One-dimensional view of connections between neurons of different planes. Only one neuron plane is drawn in each layer. The highlighted connections represent the receptive fields of the shaded neurons.

character is then presented to the input layer in Fig. 7.2. Like neocognitron, there are twelve neuron planes in the first layer. Each plane is trained for erosion with one of the twelve line segments (or features) in a 3×3 window as the structuring element. The twelve training features are illustrated in Fig. 7.4 which is adopted from those used in neocognitron [40]. Each neuron in the plane is trained with the one-pass erosion algorithm [84] to detect existence of the corresponding line segment. Among those training patterns, patterns 1 and 2 represent the same orientation and, therefore, the found patterns will be merged in the dilation layers later. This is also applied to the pattern pairs, 4 and 5, 7 and 8, and 10 and 11.

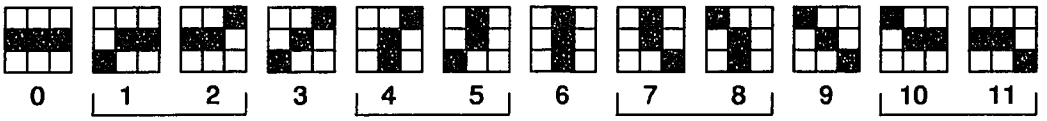


Fig. 7.4 The twelve 3×3 line segments used for erosion training in first layer.

The size of neuron plane in layer 1 has the same plane size as the input image, 23×23. Each neuron is connected to inputs in the receptive field centered at the corresponding position. Neurons at boundaries are connected to inputs within smaller receptive fields at boundaries. A firing neuron in layer 1 indicates the existence of the line segment at the exact location of the input image. There are 12 neuron planes in this layer trained with twelve structuring elements

Layer 2 also has 12 neuron planes and each of them is reduced into a half of the input image size, i.e. 11×11, with each neuron connected to the 3×3 receptive field centered at every two pixels in layer 1 as shown in Fig. 7.3. This is to reduce the dimensionality of the information so that redundant or repeated information can be removed. The reduction rate, $\frac{(n-1)}{2}$, is chosen because of easy implementation of the interconnections between neurons. Although the plane size is reduced, the receptive field of each neurons overlap with its 4 neighboring neurons with 1×3 or 3×1 inputs. This is to prevent loss of information during the reduction task of dimensionality. Besides the reduction of plane size, there are cross interconnections between some neuron planes. For example, planes 1 and 2 (for features 1 and 2 in Fig. 7.4) in layer 1 are both connected (with the corresponding receptive fields) to the neurons at the same position in both planes 1 and 2 of layer 2. That is, an existence of 22.5° line segment should appear in both neurons planes. Same cross interconnections are constructed in other pairs of neuron planes in identically oriented line segments.

Layer 2 is trained with the one-pass dilation algorithm [84] to make the architecture

more flexible to positional and scaling variations of the input image. The structuring element used for these dilation neuron planes is shown in Fig. 7.5. which consists of all value 1's in the 3×3 window. Thus, the location of a line segment found is blurred into a larger area so that a certain degree of positional error can be ignored. The structuring element is designed to allow more variation of the feature in all of the eight directions. An alternative choice is to use the 12 structuring elements shown in Fig. 7.6. Theoretically, they are line segments perpendicular to the corresponding training patterns for erosion. This is because the detected features may only be shifted in the perpendicular direction due to the positional errors.

Fig. 7.5 The dilation training pattern designed for all twelve planes in layers 2 and 4.

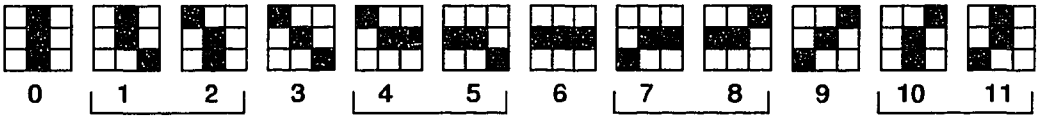


Fig. 7.6 The 12 perpendicular dilation structuring elements allowing shift error.

Each neuron plane in layer 2 is responsible for one orientation and looks for all matching line segments in that orientation. The outputs show groups of 3-pixel line segments appeared in the relative area within the input image. We consider it is the result of decomposing the input image into subimages containing different classes (or orientations) of local features. Due to the high dimensionality, i.e. 11×11×12, of these subimages, we have to apply the same feature extraction and dimensionality reduction task with more layers of morphological neurons.

Layer 3 also consists of 12 neuron planes for different line orientations. Each neuron plane is connected to the output of layer 2 with the same connection topology as layer 1. The same set of erosion training patterns in Fig. 7.4 are used to train the connection weights of the corresponding neuron planes. Therefore, the function of this layer can be interpreted as looking for longer line segments by combining those short line segments found in layer 1 and 2.

For example, both plane 0 in layer 1 and 3 are trained with horizontal line segment with three pixels. The neurons in plane 0 of layer 1 detect the existence of horizontal and short line segments, while those in plane 0 of layer 3 integrate these information and determine the existence of horizontal line segments as long as three times of horizontal short line segments. Due to dilation results of layer 2, the position of detected short line segment can be shifted in any direction. Thus, a firing neuron in plane 0 of layer 3 may indicate the existence of a straight or slightly twisted but nearly straight line segment with length from 3 to 9 pixels.

Layer 4 has the same connection topology and uses the same dilation training pattern as in layer 2. The purpose of this layer is to expand the location of detected line segments into a larger area like upper half, lower half, or center part in the vertical direction and right side, left side, or middle part in the horizontal direction. This is because we only consider such information in the classification module.

Since the size of neuron planes in layer 4 is reduced into 5×5, it is like dividing the input image into 25 regions. Each neuron correspond to one of these regions. Every pair of adjacent neurons covers adjacent regions with 9 rows or 9 columns overlapping in the input image. Fig. 7.7 shows how the covering areas of two adjacent neurons in layer 4 overlap with each other. The dashed lines with shaded circles show the connections and neurons covered by the neuron in layer 4 and the thicker solid lines and circles show the connections and neurons for the center one.

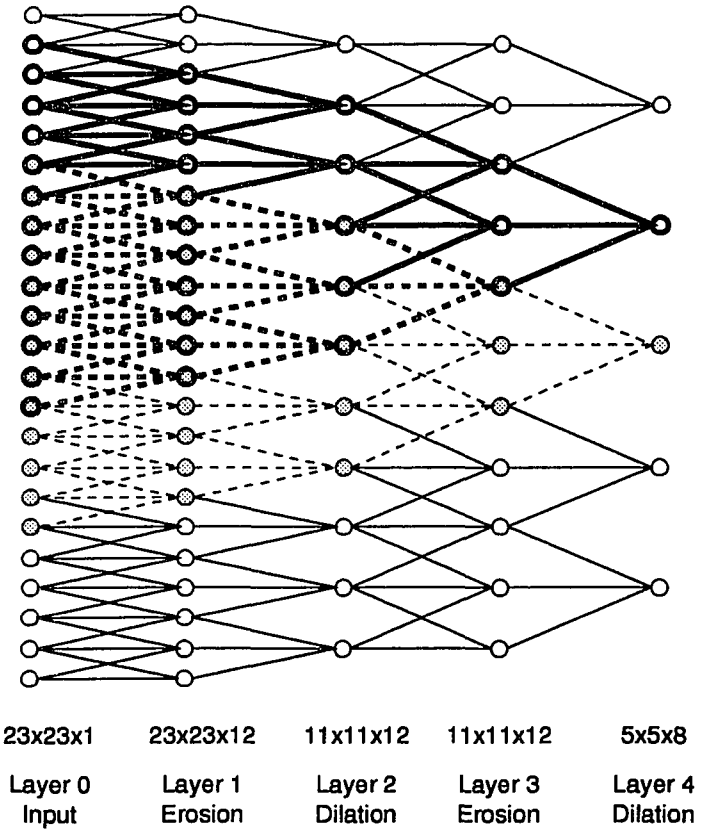


Fig. 7.7 One-dimensional view of overlapping connections between adjacent neurons The thicker dashed lines indicate the overlapping part of the dashed neuron and the thicker solid neuron.

In Fig. 7.8, we show the experimental results of the morphological feature preprocessing module with handwritten characters "A" and "Q." The grey areas indicate pixels with value 0 while the dark areas represent pixels with value 1 which is part of the input character. The left most column shows the original images of three different characters written by the same person. The second column gives the skeletonized characters. The outputs of eight neuron planes for the input characters are shown on the right hand side along with the corresponding structure elements (3×3 local features). The outputs on the right hand side shows the strokes extracted are similar among difference versions. Fig. 7.9 illustrates the preprocessing outputs of one set of 26

handwritten characters by the same person. The characters are written at different time and/or on different days.

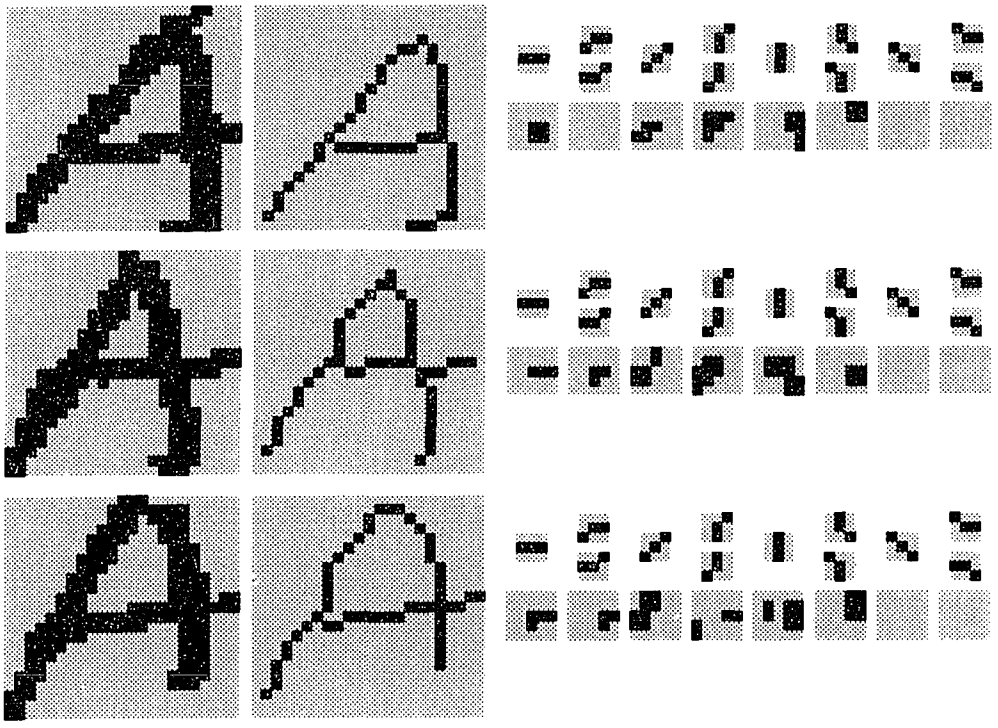


Fig. 7.8 (a) The outputs of three "A" characters.

7.3 Stroke Extraction

The strokes considered in the character recognition task of human being include vertical strokes, horizontal strokes, positive slope strokes, negative slope strokes, and vertical and horizontal curves at different relative locations and with different lengths. We design the strokes extraction module to extract these heuristic features for the recognition task. Table 7.1 shows a list of 35 strokes that we consider significant for character recognition task. These strokes are extracted by the module and passed to the next module for recognition.

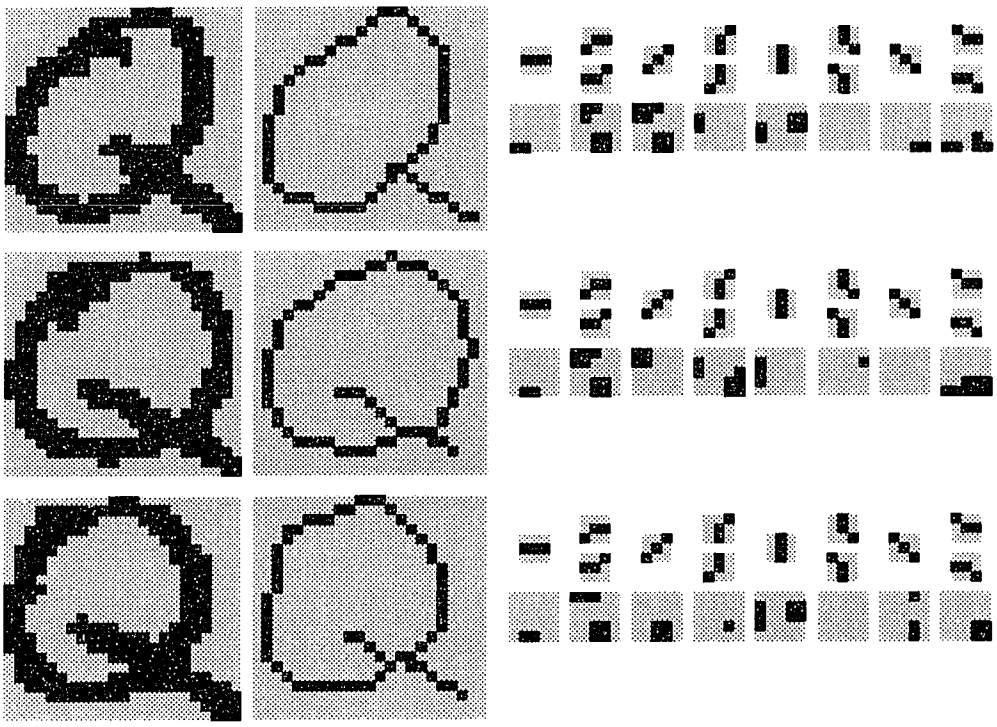


Fig. 7.8 (b) The outputs of three "Q" characters.

Each neuron in this module is connected to certain parts of the output from the preprocessing module based on the stroke to be extracted. This is because a stroke may only appear in a certain part of the input image. The neurons should focus on specified areas for the strokes they are looking for. Figs. 7.10(a) and 10(b) illustrate the specified receptive fields for some of the strokes listed in Table 7.1. Each column represents the output vector of the feature preprocessing module. We show multiple columns of neuron planes to avoid confusion of overlapped receptive fields for different strokes. The receptive fields are indicated with shaded areas along with the names of the corresponding strokes. Note that some strokes are extracted from the same receptive fields of different neuron planes. This is to cover strokes that are tilted or distorted with a certain degree and to allow such variations of strokes to be learned by the module.

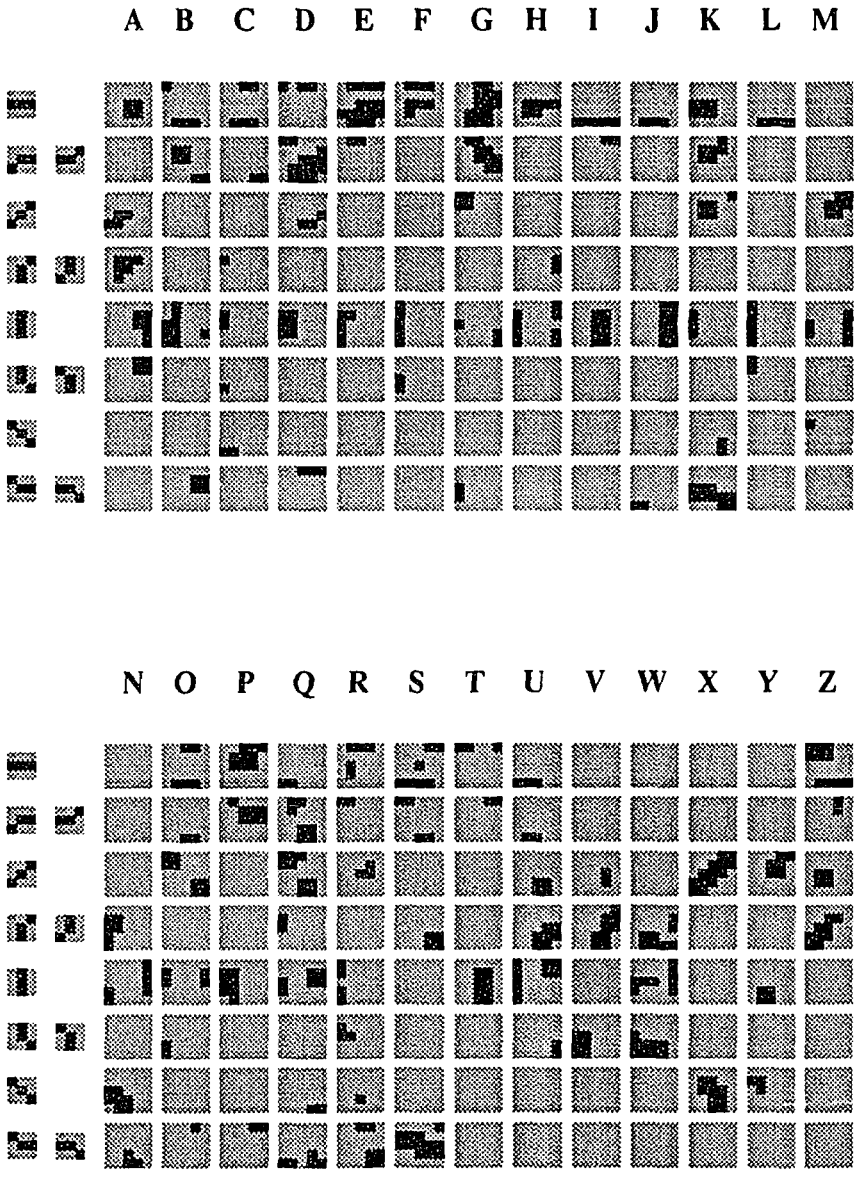


Fig. 7.9 The preprocessing outputs of one set of 26 capital letters by the same person.

Alternatively, we can choose another feature set containing 43 features of the preprocessing outputs shown in Fig. 7.11. These features are determined by dividing the output neuron planes of preprocessing module into regions, such as top, middle, and bottom region in the plane for horizontal strokes, left, center, right, upper left, upper

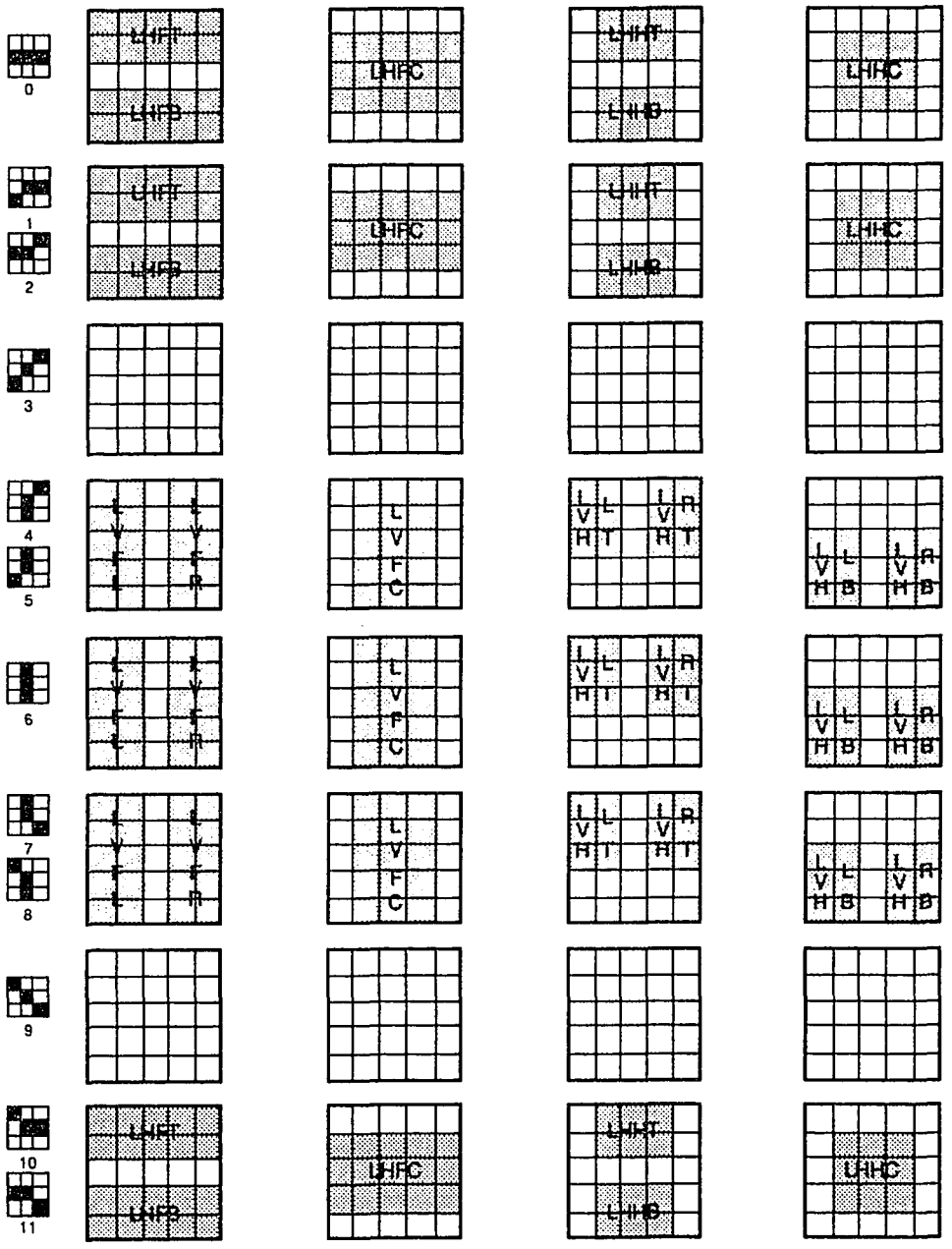


Fig. 7.10 (a) The receptive fields for some of the chosen strokes. Shaded areas represent receptive fields for the named strokes. Multiple columns are used to show overlapped receptive fields for different strokes.

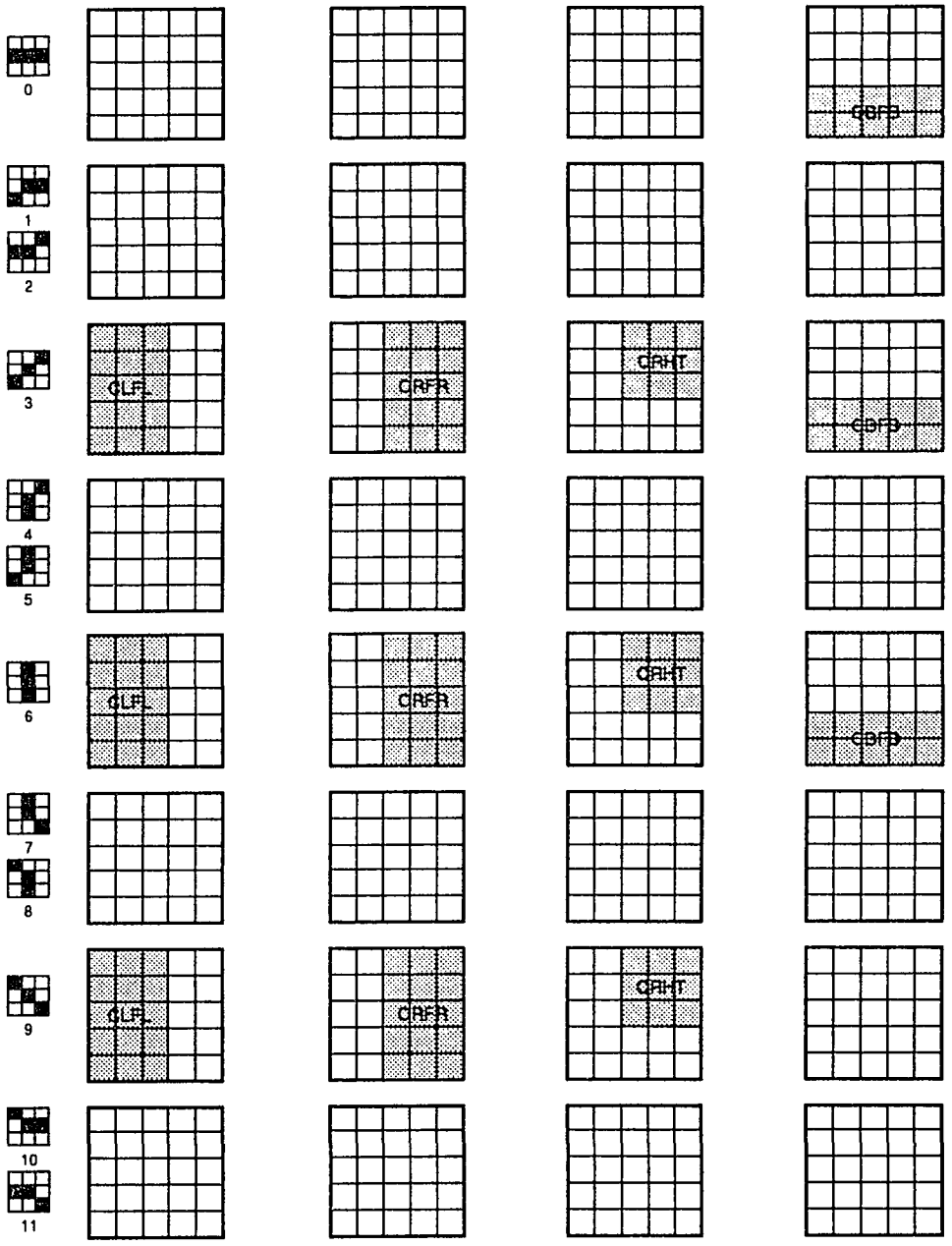


Fig. 7.10 (b) The receptive fields for more strokes.

right, lower left, lower right, and lower center in the plane for vertical strokes. In the experiments, we obtained the same results as those of using the stroke features in Table 7.1.

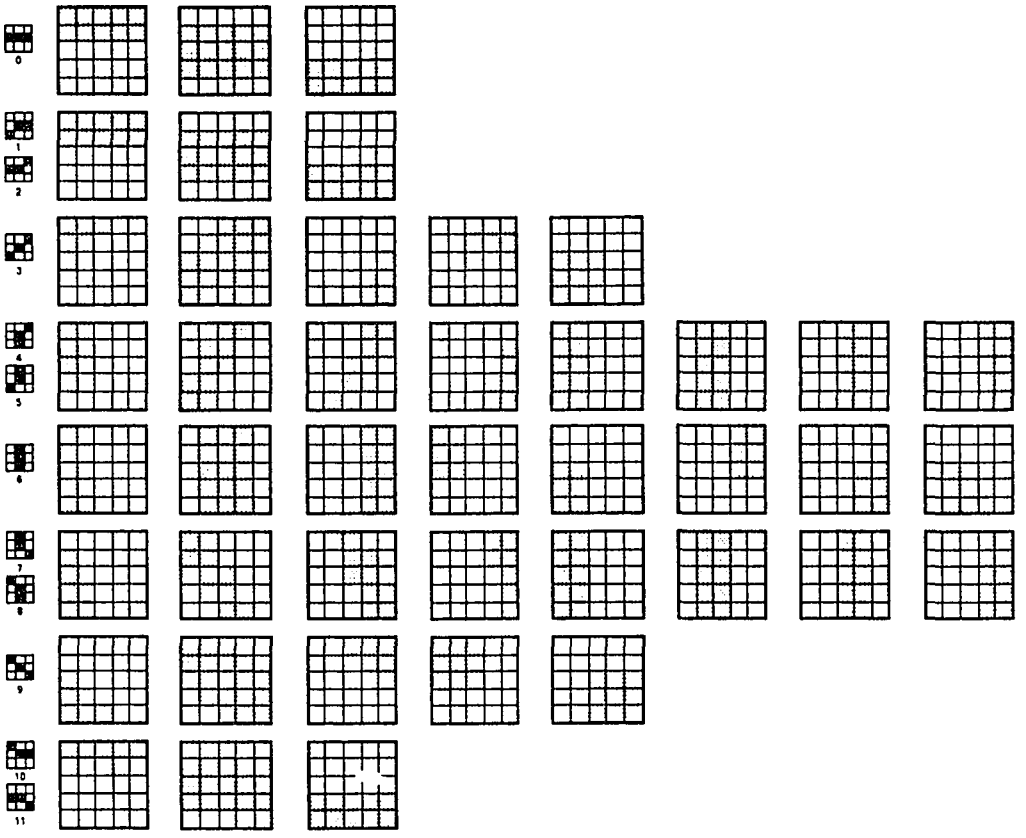


Fig. 7.11 Alternative feature set determined by region division of neuron planes.

Either the delta learning rule proposed by Rumelhart *et al.* [95] or the Kohonen's feature map training algorithm [65] can be adopted to train the connection weights of such neurons. The former is a generalization of the LMS algorithm to minimize an error function equal to the mean square difference between the desired and the actual neuron outputs. The error function is given as

$$E = \frac{1}{2} \sum_{j} (t_j - y_j)^2 \tag{7.1}$$

Tab. 7.1 Useful strokes for handwritten character recognition.

Names	Meaning of Feature Names
LVFR	Line, Vertical, Full length, at Right side
LVFL	Line, Vertical, Full length, at Left side
LVFM	Line, Vertical, Full length, at Middle
LVHRT	Line, Vertical, Half length, at Right Top
LVHLT	Line, Vertical, Half length, at Left Top
LVHMT	Line, Vertical, Half length, at Middle Top
LVHRB	Line, Vertical, Half length, at Right Bottom
LVHLB	Line, Vertical, Half length, at Left Bottom
LVHMB	Line, Vertical, Half length, at Middle Bottom
LHFT	Line, Horizontal, Full length, at Top
LHFB	Line, Horizontal, Full length, at Bottom
LHFC	Line, Horizontal, Full length, at Center
LHHT	Line, Horizontal, Half length, at Top
LHHB	Line, Horizontal, Half length, at Bottom
LHHC	Line, Horizontal, Half length, at Center
SPFM	Slope, Positive, Full length, at Middle
SPFR	Slope, Positive, Full length, at Right side
SPFL	Slope, Positive, Full length, at Left side
SPHT	Slope, Positive, Half length, at Top
SPHB	Slope, Positive, Half length, at Bottom
SNFM	Slope, Negative, Full length, at Middle
SNFR	Slope, Negative, Full length, at Right side
SNFL	Slope, Negative, Full length, at Left side
SNHT	Slope, Negative, Half length, at Top
SNHB	Slope, Negative, Half length, at Bottom
CTFC	Curve, Top part, Full length, at Center
CTFT	Curve, Top part, Full length, at Top
CBFC	Curve, Bottom part, Full length, at Center
CBFB	Curve, Bottom part, Full length, at Bottom
CRHT	Curve, Right part, Half length, at Top
CRHB	Curve, Right part, Half length, at Bottom
CRFR	Curve, Right part, Full length, at Right side
CLHT	Curve, Left part, Full length, at Left side
CLHB	Curve, Left part, Half length, at Bottom
CLFL	Curve, Left part, Half length, at Top

where j stands for the indices of neurons in the receptive fields connected to the current neuron, y is the actual output of the neuron, and t is the target output of the neuron. The derivation can be found in [95] and the weight adjusting equation is

$$w_{ij}(t+1) = w_{(t)} + \eta y_j(1 - y_j)(t_j - y_j)x'_i \tag{7.2}$$

where $w(t+1)$ and $w(t)$ indicate the connection weight at time $t+1$ and t from output neuron j of the preprocessing module to neuron i in the stroke extraction module. The constant η is the learning rate. Since the desired outputs is to be provided along with the training patterns, supervised learning is used.

The Kohonen's self-organized feature map algorithm tries to reduce the distance between the input vector and the weight vector:

$$d_i = \sum_{j=0}^{N-1} \Big[x_j(t) - w_{ij}(t) \Big]^2 \tag{7.3}$$

where j is the index of N input and i is the index of the current neuron. Deriving from this goal function, we may obtain the weight adjusting equation

$$w_{ij}(t+1) = w_{(t)} + \eta(t) \Big[x_j(t) - w_{ij}(t) \Big] \tag{7.4}$$

where $0 < \eta(t) < 1$ is the learning rate adjustable during the training phase. Variations of the same stroke can be learned with such a kind of training algorithms. The output values of the neurons indicate the existence of the extracted strokes in the input image. Unsupervised learning is applied with this training algorithm.

In our experiment, we chose the multilayered perceptron (MLP) model to implement this module and partially connected the outputs to the inputs of the next module for pattern classification. The desired outputs for training this stroke extraction module are self-organized and formed by the pattern classification module when the generalized delta rule is activated and back-propagates errors from the output layer of

pattern classification module. Fig. 7.12 shows the structure of the stroke extraction module. Each circle represents a multilayer perceptron unit (or MLP unit) responsible for one of the listed stroke features in Table 7.1 (or region features in Fig. 7.11). The dashed lines show the input connections from the receptive fields of features given in Fig. 7.10 (or Fig. 7.11). We only show the connections of several MLP units in order to make the figure simple and understandable.

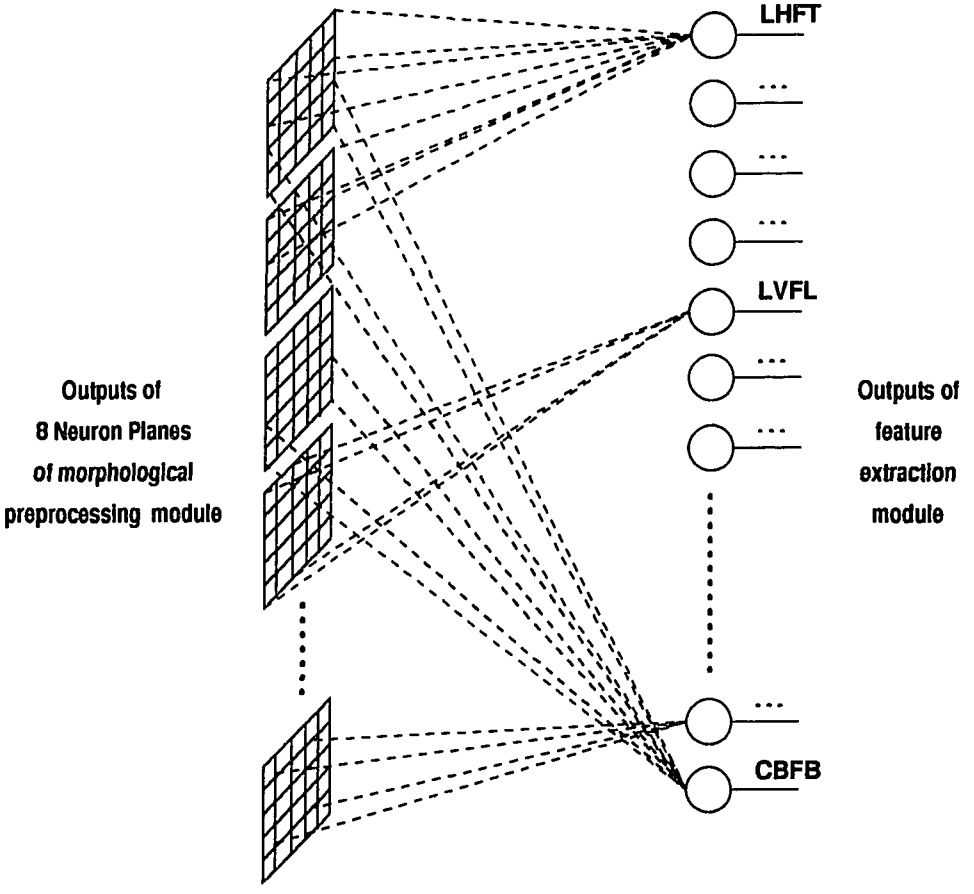


Fig. 7.12 The structure of the stroke extraction module. Each neuron represents and extracts one of the given strokes.

The inputs to the MLP units are the binary output of the preprocessing module, indicating the existence of short line segments. There is only one output neuron in each of the MLP unit. The output of MLP units is a value in the range of 0.0 and 1.0 indicating the degree of existence of the corresponding features. The output values from the MLP units are used as the input of the pattern classification module.

In our software-simulated experiments, we tried MLP units with different numbers of hidden layers and hidden neurons in each hidden layer. From our observation, MLP units with 2 hidden layers which have 20 and 10 neurons in the first and the second hidden layer respectively perform the best results in generalizing stroke information with variations and reasonably short CPU time for training task. Higher number of neurons in hidden layers may be used to enlarge the weight capacity in order to learn more stroke variations but is practical only if neuron structures and connection topology are implemented in hardwares.

7.4 Pattern Classification

As mentioned in the beginning of this chapter, the first category of character recognition adopts a conventional feature extraction algorithm and then loads the obtained feature patterns into an existing neural network classifier as the training/testing patterns. This concept takes the advantage of good performance of neural network models as pattern classifiers. Our pattern classification module is designed with the same concept.

In our experiment, we adopted the multilayered perceptron model as our pattern recognizer and used generalized delta rule for training the connection weights. Because a supervised learning algorithm is used in the module, the module should be provided with desired output indicating what character is loaded for training. In addition, the pattern classification module should back-propagate the error between the actual output and the desired output in order to provide the desired stroke patterns.

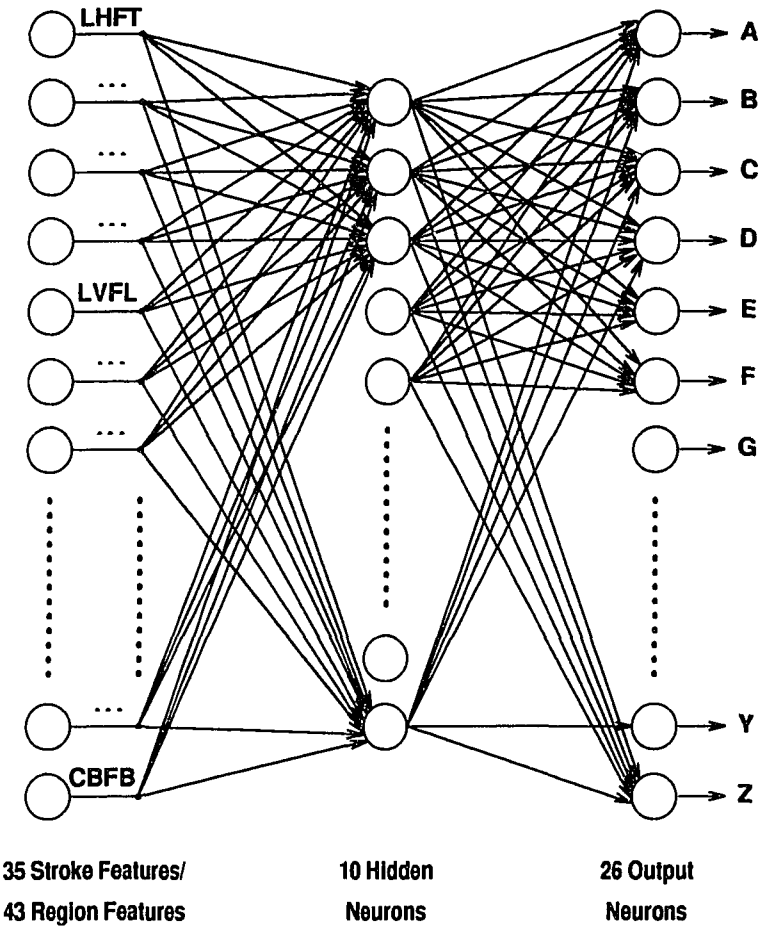


Fig. 7.13 The MLP structure of the pattern classification module.

Fig. 7.13 shows the structure of the pattern classification module. For simplicity, only several neurons in each layer are shown. The input is the feature pattern representing the existence of 35 stroke features (or 43 region features). The output layer has 26 neurons with each neuron representing a character. The output values are in the range of 0 and 1.0 which indicate the possibilities of an input character images belonging to the corresponding character. The number of output neurons should increase if lower case characters, digits, and special characters are to be recognized. We also observed that 20 neurons in the hidden layer made good performances.

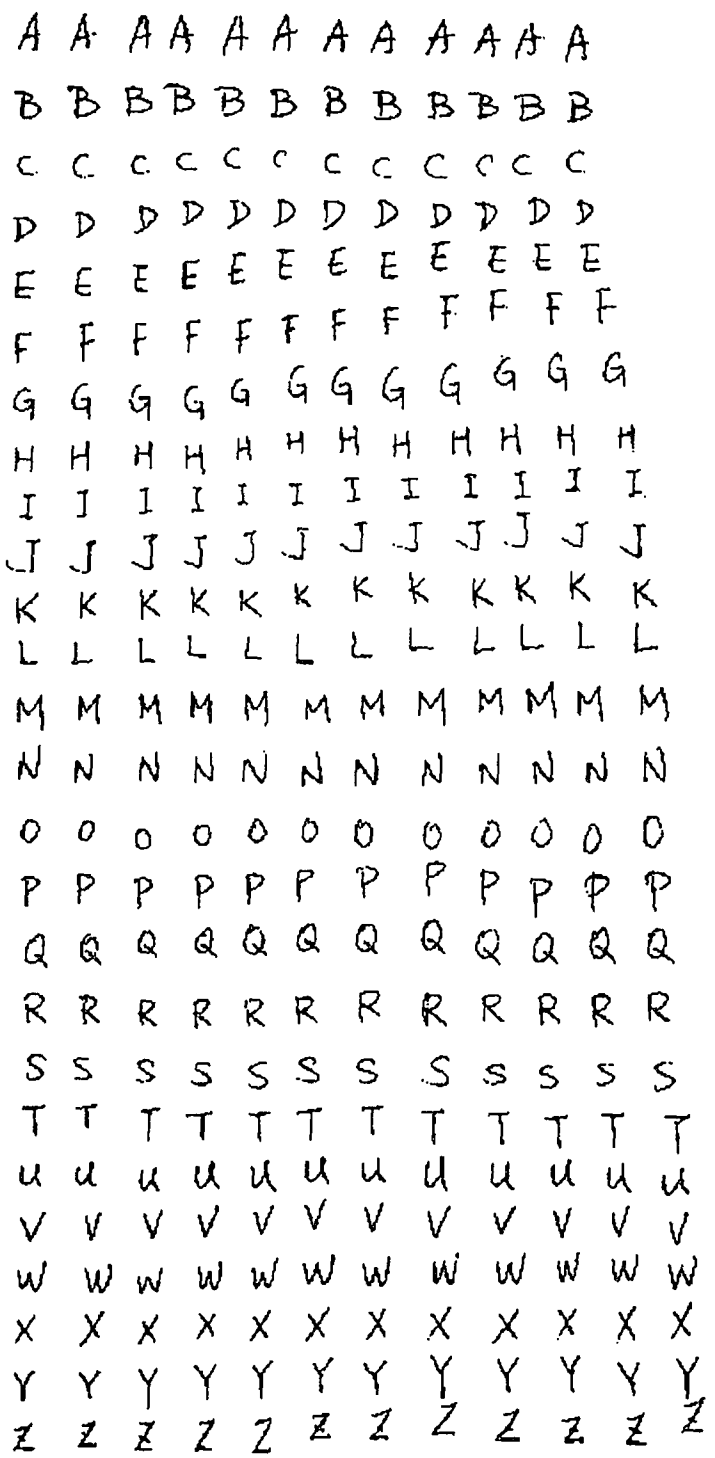


Fig. 7.14 (a) Ten sets of handwritten characters for training/testing.

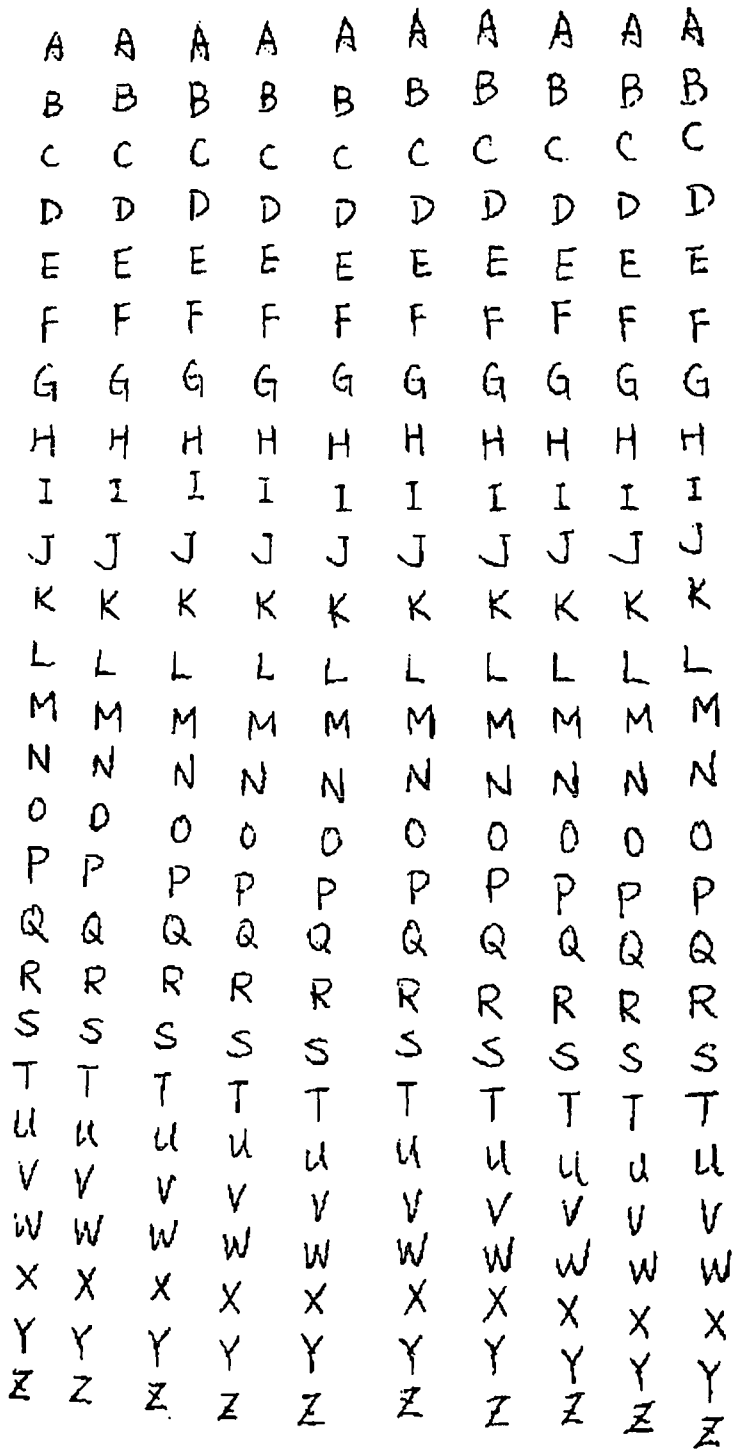


Fig. 7.14 (b) Ten more sets of handwritten characters for training/testing.

7.5 Experimental Results

We simulated the proposed architecture in C++ language on a Sun SPARC-20 workstation. Twenty sets of handwritten capital alphabets written by the same person are pre-scanned in a resolution of 300×300 dot/inch and are segmented into separated binary images of characters with their bounding boxes. That is, total of 520 character images are used for training and/or testing. Fig. 7.14 shows the scanned image of 20 character sets.

Each character is first normalized into a size of 23×23 pixels. This is because we are interested in the relative sizes and positions of features in the whole character image. Next, a parallel thinning algorithm [115] is applied to each character image in order to obtain a skeleton of the character. This is because that the first layer of morphological feature preprocessing module performs erosions on one-pixel-wide strokes using the 12 structuring elements shown in Fig. 7.4.

In our first experiment, we randomly chose n ($1 \leq n \leq 10$) sets from 20 sets of character images and loaded them to the system along with the desired output provided at the output layer of pattern classification module. The generalized delta rule is used to conduct the training phase. The learning rate (i.e. η in eq.(7.2)) is set to 0.2 and the tolerance (μ) is set to 0.001 to determine the convergence condition. The system achieved the convergence condition in between 900 and 1,500 iterations though the maximum number of training iterations was set to 5,000. Fig. 7.15 shows the curves of overall errors versus the training iterations. The convergence curves of different n training sets are drawn and show how similar the system converges during the training phase for different numbers of training sets.

After the system has converged, we loaded all character sets for testing in order to evaluate the system performance. The recognition rates were calculated by dividing the number of correctly recognitioned characters by 20 sets of training and testing sets. Fig.

7.16 shows how the recognition rate increased with more training sets were added for training. The horizontal coordinate is the number of training sets chosen, i.e. n in the previous paragraph, while the vertical coordinate is the recognition rate in unit of percentage. The system keeps increasing recognition rates while generalizing more handwritten variations of stroke features of the same character. After all the 20 handwritten character sets are trained and memorized in the connection weights, the system reached as high as 99.23% recognition rate.

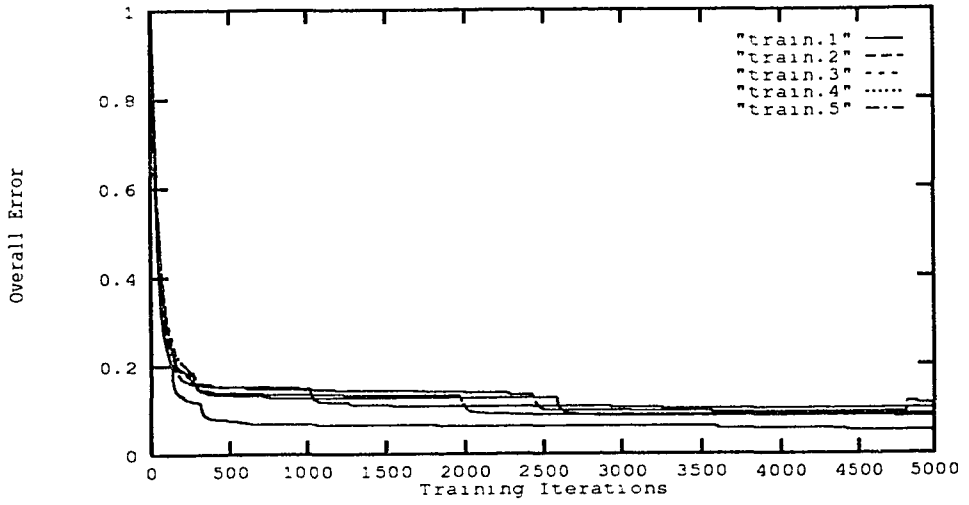


Fig. 7.15 The training curves showing the convergence of the system during training.

In Fig. 7.16, the curve of the recognition rates is not monotonically increasing. We investigated the phenomenon and reviewed the original characters images and the intermediate results of the morphological preprocessing module. An assertion is made that it is because of a set of characters with a significant number of stroke variation comparing with those of pre-learned character sets.

In our second experiment, we would like to verify the system performance in recognition of untrained character sets. We randomly chose 10 sets of character images

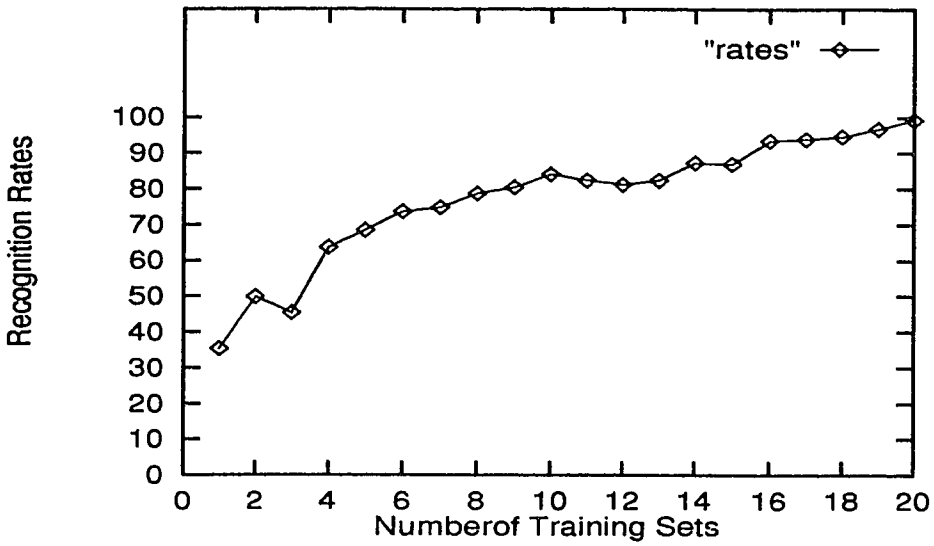


Fig. 7.16 Recognition rate of all 20 character sets including n training sets.

as the training group and used the remaining 10 sets as testing group. We randomly chose n ($1 \leq n \leq 10$) sets from the training group and loaded them to the system along with the desired output provided at the output layer of pattern classification module. Again, the learning rate η is set to 0.2 and the tolerance μ to 0.001. The system takes long time to achieve convergence condition up to 3,000 iterations if more training sets were used. The convergence curves have similar shapes as those shown in Fig. 7.15.

After the system has converged, we loaded the n training sets and the 10 sets in the testing group for testing in order to evaluate the system performance. The recognition rates were calculated by dividing the number of correctly recognitioned characters by training sets or testing sets. Fig. 7.17 shows that the recognition rates for training sets maintains in a higher range while those of testing sets increased after more training sets were added. Again, the horizontal coordinate is the number of training sets chosen, i.e. n, while the vertical coordinate is the recognition rate in unit of percentage. The recognition rates maintains monotonically increasing while generalizing more handwritten variations of stroke features of the same character.

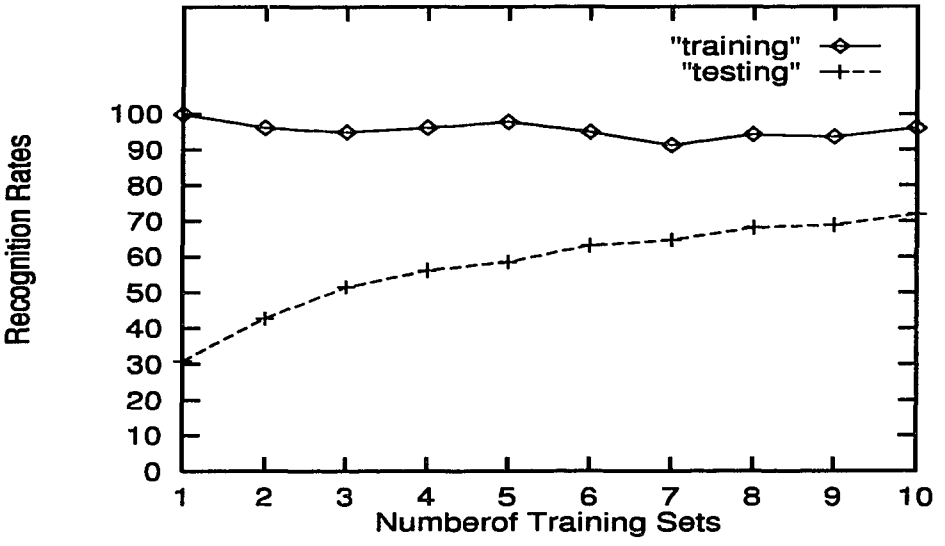


Fig. 7.17 Recognition rate of training character sets and the 10 testing sets.

7.6 Conclusion

A handwritten character recognition system designed for personal use is proposed. The system utilizes neural architecture constructed with morphological neurons along with the fast training algorithm we proposed in the previous chapter as the feature preprocessing module. It takes the advantage of simple architecture and fast training phase and can extract local features such as line segments with less than 10 pixels. This makes it easy with hardware implementation because of simple computations are envolved.

The remaining two parts adopt the multilayered perceptron with delta rule for training. The generalization capability of MLP has been proven [95] and is also shown in our handwritten recognition experiments. The architecture demonstrates a plausible idea in designing more complicated handwriting recognition system for other languages, such as Japanese and Chinese. Further researches can be done in investigating more useful stroke features if lower case, digits, and special characters are envolved.

CHAPTER 8

SUMMARY

This dissertation is aimed to investigate neural network approach of pattern recognition applications, especially in the implementation of mathematical morphology and to develop new useful morphological neural architectures which can be applied to image analysis and object/character recognition. In this final Chapter we summarize the contributions of our research and briefly discuss the further potential research.

8.1 Contribution of this Dissertation

Our research is oriented toward aspects in both architectures and applications.

1. *Improved ART model.* The theoretical background of the ART model was studied and the potential problems were investigated and experimented. Improvements were made to overcome the problem. The improved architecture was applied to image enhancement and printed character recognition.
2. *Implementation with Programmable Logic Modules.* We utilized dynamically programmable logic module to implement morphological operations. Detailed connection topology and signal controls are discussed. This makes hardware implementation of neural morphology practical.
3. *Neural Morphology for Grey-Scale Images.* We proposed different neural architectures to implement grey-scale morphological operations. Improvement of different proposals were discussed.
4. *Multilayered Perceptron (MLP) for Mathematical Morphology.* The theoretical foundation of multilayered perceptron were studied. MLPs was used as processing

modules for learning transformation rules. Practical image processing operations were experimented with the proposed architecture and real images were tested.

5. *One-Pass Training Algorithms for Morphology.* New training algorithms with traditional types of neurons were proposed for morphological operations. The proposed neurons were applied in processing real images.
6. *Personal Handwritten Recognition System.* A new handwritten recognition system was proposed for personal use. The system uses one-pass trained neuron layers for local feature extraction and multilayered perceptrons for global feature generalization and final classification. The experiments show the system achieved high recognition rate after a certain time amount of learning personal handwriting styles and habits.

8.2 Epilogue

My original goal in this dissertation was to study the properties and neural implementation of mathematical morphology. Existing neural network models were investigated and theoretical weakness of these models were found and overcome. These models were also used to implement morphological operations. The final result is that this dissertation has proposed some new neural architectures and training algorithm. A real system for personal handwritten recognition were implemented. I strongly hope that this dissertation will be useful for supporting hardware implementations of neural morphology.

REFERENCES

[1] L. Abbott, R. M. Haralick, and X. Zhuang, "Pipeline Architectures for Morphologic Image Analysis," *Machine Vision, and Applications*, vol.1, pp.23-40, 1988.

[2] T. Agui, H. Takahashi, and H. Nagahashi, "Recognition of Handwritten Katakana in a Frame Using Moment Invariants Based on Neural Network," *Int'l Joint Conf. on Neural Networks,*, vol.1, pp.659-664, Singapore, 1991.

[3] D. L. Alkon, K. T. Blackwell, G. S. Barbour, A. K. Rigler, and T. P. Vogl, "Pattern-Recognition by an Artificial Network Derived from Biologic Neuronal Systems," *Biological Cybernetics*, vol.62, pp.363-376, 1990.

[4] J. A. Anderson and E. Rosenfeld, *Neurocomputing Foundations of Research*, MIT Press, Cambridge, MA, 1988.

[5] W. W. Armstrong and J. Gecsei, "Adaptation Algorithms for Binary Tree Networks," *IEEE Trans. on Systems, Man, and Cybernetics*, vol.SMC-9, pp.276-285, May 1979.

[6] N. P. Archer and S. Wang, "Fuzzy Set Representation of Neural Network Classification Boundaries," *IEEE Trans. on Systems, Man, and Cybernetics*, vol.SMC-21, no.4, pp.735-742, July/August 1991.

[7] A. Barr, P. R. Cohen, and E. A. Feigenbaum, (Eds.) *The Handbook of Artificial Intelligence*, vol.IV, Addison-Wesley, New York, NY, 1989.

[8] E. Barnard and D Casasent, "Shift Invariance and the Neocognitron," *Neural Networks*, vol.3, no.4, pp.403-410, 1990.

[9] K. T. Blackwell, T. P. Vogl, S. D. Hyman, G. S. Barbour, and D. L. Alkon, "A New Approach to Hand-Written Character Recognition," *Pattern Recognition*, vol.25, no.6, pp.655-666, 1992.

[10] H. D. Block, "The Perceptron: a Model for Brain Functioning. I," *Reviews of Modern Physics*, vol.34, 1962.

[11] D. J. Burr, "Experiments on Neural Net Recognition of Spoken and Written Text," *IEEE Trans. on Acoustics, Speech, and Signal Processing*, vol.36, no.7, pp.1162-1168, July 1988.

[12] G. A. Carpenter and S. Grossberg, "A Massively Parallel Architecture for a Self-Organizing Neural Pattern Recognition Machine," *Computer Vision, Graphics, and Image Processing*, vol.37, pp.54-115, 1987.

[13] G. A. Carpenter and S. Grossberg, "ART 2: Self-Organization of Stable Category Recognition codes for Analog Input Patterns," *Applied Optics*, vol.26, pp.4919-4930, December 1987.

[14] G. A. Carpenter and S. Grossberg, "The ART of Adaptive Pattern Recognition by a Self-Organization Neural Network," *IEEE Computer Magazine*, pp.77-88, March 1988.

[15] G. A. Carpenter and S. Grossberg, "ART3: Hierarchical Search Using Chemical Transmitters in Self-Organizing Pattern Recognition Architectures," *Neural Networks*, vol.3, no.2, pp.129-152, 1990.

[16] G. A. Carpenter, S. Grossberg, N. Markuzon, J. H. Reynolds, and D. B. Rosen, "Fuzzy ARTMAP: A Neural Network Architecture for Incremental Supervised Learning of Analog Multidimensional Maps," *IEEE Trans. on Neural Networks*, vol.3, no.5 pp.698-713, September 1992.

[17] G. A. Carpenter, "Neural Network Models for Pattern Recognition and Associative Memory," *Neural Networks*, vol.2, no.4, pp.243-257, 1989.

[18] C. H. Chen and P. G. Mulgaonkar, "CAD-Based Feature-Utility Measures for Automatic Vision Programming," *Proc. of IEEE*, pp.106-114, 1991.

[19] C. H. Chen and P. G. Mulgaonkar, "Automatic Vision Programming," *CVGIP: Image Understanding*, vol.55, no.2, pp.170-183, March 1992.

[20] F. Crick, "Foundation of the Thalamic Reticular Complex: The Searchlight Hypothesis," *Proc. of the National Academy of Sciences*, vol.81, pp.4586-4590, 1984.

[21] Defense Advanced Research Projects Agency (DARPA), *Neural Network Study*, AFCEA International Press, Fairfax, VA, 1988.

[22] J. L. Davidson and F. Hummer, "Morphology Neural Networks: An Introduction with Applications," *Circuits, Systems, Signal Processing*, vol.12, no.2, pp.177-210, 1993.

[23] J. L. Davidson and K. Sun, "Template Learning in Morphological Neural Nets," in *SPIE vol.1568 - Image Algebra and Morphological Image Processing II*, 1991.

[24] G. Deco and R. Blasig, "Handwritten Digit Recognition with Principal Component Analysis and Radial Basis Functions," *Int'l Joint Conf. on Neural Networks*,, vol.3, pp.2253-2256, Nagoya, Japan, November 1993.

[25] A. R. Dill, M. D. Levine, and P. B. Noble, "Multiple Resolution Skeletons," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol.PAMI-9, no.4, pp.495-504, July 1987.

[26] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*, Wiley, New York, NY, 1973.

[27] K. S. Fu, *Syntactic Methods in Pattern Recognition*, Academic Press, New York, NY, 1974.

[28] K. Fukushima, T. Imagawa, and E. Ashida, "Character Recognition with Selective Attention," *IEEE Int'l Joint Conf. on Neural Network*, vol.I, pp.I.593-598, July 1991.

[29] K. Fukushima and T. Imagawa, "Recognition and Segmentation of Connected Characters with Selective Attention," *Neural Network*, vol.6, no.1, pp.33-41, 1993.

[30] K. Fukunaga and W. Koontz, "Application of the Karhunen-Loeve Expansion to Feature Selection and Ordering," *IEEE Trans. on Computer*, vol.C-19, pp.311-318, 1970.

[31] K. Fukushima, S. Miyake, and T. Ito, "Neocognitron: A Neural Network Model for a Mechanism of Visual Pattern Recognition," *IEEE Trans. on System, Man, and Cybernetics*, vol.SMC-13, no.5, pp.826-834, September 1983.

[32] K. Fukushima and N. Wake, "Handwritten Alphanumeric Character Recognition by the Neocognitron," *IEEE Trans. on Neural Networks*, vol.2, no.3, pp.355-365, May 1991.

[33] K. Fukushima and N. Wake, "Improved Neocognitron with Bend-Detecting Cells," *IEEE/INNS Int'l Joint Conf. on Neural Networks*, vol.4, pp.190-195, Baltimore, MD, 1992.

[34] K. Fukunaga, *Introduction to Statistical Pattern Recognition*, Academic Press, New York, NY, 1972.

[35] K. Fukushima, "Cognitron: A Self-organizing Multilayered Neural Network," *Biological Cybernetics*, vol.20, pp.121-136, 1975.

[36] K. Fukushima, "Neocognitron: A Self-organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position," *Biological Cybernetics*, vol.36, pp.193-202, 1980.

[37] K. Fukushima, "A Hierarchy Neural Network Model for Associate Memory," *Biological Cybernetics*, vol.50, pp.105-113, 1984.

[38] K. Fukushima, "A Neural Network Model for Selective Attention in Visual Pattern Recognition," *Biological Cybernetics*, vol.55, pp.5-15, 1986.

[39] K. Fukushima, "Neural Network Model for Selective Attention in Visual Pattern Recognition and Associative Recall," *Applied Optics*, vol.26, no.23, pp.4985-4992, December 1987.

[40] K. Fukushima, "Neocognitron: A Hierarchical Neural Network Capable of Visual Pattern Recognition," *Neural Network*, vol.1, no.2, pp.119-130, April 1988.

[41] K. Fukushima, "A Neural Network for Visual Pattern Recognition," *IEEE Computer Magazine*, pp.65-75, March 1988.

[42] K. Fukushima, "Neocognitron: A Hierarchical Neural Network Capable of Visual Pattern Recognition," *Neural Networks*, vol.1, no.2, pp.119-130, April 1988.

[43] K. Fukushima, "Analysis of the Process of Visual Pattern Recognition by the Neocognitron," *Neural Networks*, vol.2, no.6, pp.413-420, 1989.

[44] K. Fukushima, "Character Recognition with Neural Networks," *Neurocomputing*, vol.4, no.5, pp.221-233, April 1992.

[45] R. C. Gonzales and R. E. Woods, *Digital Image Processing*, Addison-Wesley, Reading, MA, 1992.

[46] S. Grossberg, "Adaptive Pattern Classification and Universal Recoding: I. Parallel Development and Coding of Neural Feature Detectors," *Biological Cybernetics*, vol.23, pp.121-134, 1976.

[47] S. Grossberg, "How Does a Brain Build a Cognitive Code?" *Psychological Review*, vol.87, pp.1-51, 1980.

[48] S. Grossberg (Ed.), *Neural Network and Natural Intelligence*, MIT Press, Cambridge, MA, 1988.

[49] S. Grossberg, "Nonlinear Neural Networks: Principles, Mechanisms, and Architectures," *Neural Networks*, vol.1, no.1, pp.17-62, 1988.

[50] R. M. Haralick, S. R. Sternberg, and X. Zhuang, "Image Analysis Using Mathematical Morphology," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol.PAMI-9, no.4, pp.532-550, July 1987.

[51] D. O. Hebb, *The Organization of Behavior, A Neuropsychological Theory*, Wiley, New York, NY, 1949.

[52] J. J. Hopfield, "Neural Networks and Physical Systems with Emergent Collective Computational Abilities," *Proc. of the National Academy of Sciences*, vol.79, pp.2554-2558, April 1982.

[53] J. J. Hopfield, "Neurons with Graded Response have Collective Computational Properties Like Those of 2-state Neurons," *Proc. of the National Academy of Sciences*, vol.81, pp.3088-3092, May 1984.

[54] L. S. Hsu, K. F. Loe, S. C. Chan, and H. H. Teh, "Two-Values Neural Logic Network," *SPIE vol.1469 - Applications of Artificial Neural Networks*, vol.2, pp.197-207, 1991.

[55] S. C. Huang and Y. F. Huang, "Bounds on the Number of Hidden Neurons in Multilayer Perceptrons," *IEEE Trans. on Neural Networks*, vol.2, no.1, January 1991.

[56] B. Hussain and M. R. Kabuka, "A Novel Feature Recognition Neural Network and its Application to Character Recognition," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol.16, no.1, pp.98-106, January 1994.

[57] Y. Idan and R. C. Chevallier, "Handwritten Digits Recognition by a Supervised Kohonen-Like Learning Algorithm," *Int'l Joint Conf. on Neural Networks.*, vol.3, pp.2576-2581, Singapore, 1991.

[58] A. Iwata, H. Kawajiri, and N. Suzumura, "Classification of Hand-Written Digits by a Large Scale Neural Network "CombNET-II"," *Int'l Joint Conf. on Neural Networks.*, vol.2, pp.1021-1026, Singapore, 1991.

[59] V. A. Jaravine, "Syntactic Neural Network for Character Recognition," *SPIE vol.1661 - Machine Vision Applications in Character Recognition and Industrial Inspections*, pp.215-223, February 10, 1992.

[60] I. Jouny and M. Sheridan, "Character Recognition Using a Multistage Neural Network," *SPIE vol.1700 - Automatic Object Recognition II*, pp.73-82, April 22, 1992.

[61] S. Kageyu, N. Ohnishi, and N. Sugie, "Augmented Multi-Layer Perceptron for Rotation-and-Scale Invariant Hand-Written Numeral Recognition," *Int'l Joint Conf. on Neural Networks.*, vol.1, pp.54-59, Singapore, 1991.

[62] J. M. Keller and D. J. Hunt, "Incorporating Fuzzy Membership Functions into the Preceptron Algorithm," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol.PAMI-7, no.6, pp.693-699, November 1985.

[63] S. W. Khobragade and A. K. Ray, "Connectionist Network for Feature Extraction and Classification of English Alphabetic Characters," *IEEE Int'l Conf. on Neural Networks* , vol.3, pp.1606-1611, San Francisco, CA, March 1993.

[64] J. Kittler, "Feature Selection Methods Based on the Karhunen-Loeve Expansion," in *Pattern Recognition Theory and Application*, (K. S. Fu and A. B. Winston, Eds.) pp.61-74, Hingham, MA, 1977.

[65] T. Kohonen, *Self-Organization and Associative Memory*, Springer-Verlag, Berlin, 1984.

[66] T. Kohonen, "An Introduction to Neural Computing," *Neural Networks*, vol.1, no.1, pp.3-16, 1988.

[67] B. Kosko, "Adaptive Inference in Fuzzy Knowledge Networks", in *Proc. on IEEE First Int. Conf. Neural Networks*, pp 261-268, San Diego, CA, July 1987.

[68] B. Kosko, *Neural Networks and Fuzzy Systems*, Prentice-Hall, Englewood Ciffs, NJ, 1992.

[69] R. Krishnapuram and L. F. Chen, "Implementation of Parallel Thinning Algorithms Using Recurrent Neural Networks," *IEEE Trans. on Neural Networks*, vol.4, no.1, pp.142-147, 1993.

[70] A. Krzyzak, W. Dai, and C. Y. Suen, "Classification of Large Set of Handwritten Characters Using Modified Back-Propagation Model," *IEEE Int'l Joint Conf. on Neural Networks*, vol.3, pp.225-232, San Diego, CA, 1990.

[71] H. Y. Liao, J. S. Huang, and S. T. Huang, "Two-Dimensional Neural Networks for Handwritten Chinese Character Recognition," *Int'l Joint Conf. on Neural Networks*,, vol.3, pp.579-584, Baltimore, MD, 1992.

[72] W. C. Lin, F. Y. Liao, C. K. Tsao, and T. Lingutla, "A Hierarchical Multiple-View Approach to Three-Dimensional Object Recognition," *IEEE Trans. on Neural Networks*, vol.2, no.1, pp.84-92, 1991.

[73] R. P. Lippmann, "An Introduction to Computing with Neural Nets," *IEEE Acoustics, Speech, and Signal Processing Magazine*, pp.4-22, 1987.

[74] J. S. Lipscomb, "A Trainable Gesture Recognizer," *Pattern Recognition*, vol.24, no.9, pp.895-907, 1991.

[75] G. Matheron, *Random Sets and Integral Geometry*, Wiley, New York, NY, 1975.

[76] W. S. McCulloch and W. H. Pitts, "A Logical Calculus of the Ideas Immanent in Nervous Activity," *Bulletin of Mathematical Biophysics*, vol.5, pp.115-133, 1943.

[77] M. Menon and K. G. Heinemann, "Classification of Patterns Using a Self-Organizing Neural Network," *Neural Networks*, vol.1, no.3, pp.201-215, 1988.

[78] F. Meyer, "Contrast Feature Extraction", in *Quantitative Analysis of Microstructures in Material Sciences, Biology, and Medicine*, Special Issue of Practical Metallography (J. L. Charmont, Ed.) Riedner-Verlag, Stuttgart, 1977.

[79] S. Miyake and K. Fukushima, "A Neural Network Model for the Mechanism of Feature-Extraction," *Biological Cybernetics*, vol.50, pp.377-384, 1984.

[80] M. L. Minsky and S. A. Papert, *Perceptrons*, MIT Press, Cambridge, MA, 1969.

[81] M. L. Minsky, *Computation: Finite and Infinite Machines*, Prentice-Hall, New York, NY, 1967.

[82] M. L. Minsky, "A Framework for Representing Knowledge," in *Mind Design*, edited by J. Haugeland, pp.95-128, MIT Press, Cambridge, MA, 1981.

[83] A. Morales and S. J. Ko, "Efficient Neural Network Implementation of Morphological Operations," in *SPIE vol.1658 - Nonlinear Image Processing III* , 1992.

[84] J. Moh and Frank Y. Shih, "Neural Architectures for Mathematical Morphology and Fuzzy Morphology" *Third Annual Conf. on Fuzzy Theory and Technology*, Pinehurst, NC, November 13-16, 1994.

[85] N. Nilsson, *Learning Machines*, McGraw-Hill, New York, NY, 1965.

[86] A. van Ooyen and B. Nienhuis, "Pattern Recognition in the Neocognitron is Improved by Neuronal Adaptation," *Biological Cybernetics*, vol.70, pp.47-53, 1993.

[87] S. K. Pal and S. Mitra, "Multilayer Preceptron, Fuzzy Sets, and Classification," *IEEE Trans. on Neural Networks*, vol.3, no.5 pp.683-697, November 1992.

[88] J. K. Paik, "Image Restoration and Edge Detection Using Neural Networks," Ph. D. Dissertation, Northwestern University, June 1990.

[89] Y. H. Pao, *Adaptive Pattern Recognition and Neural Networks*, Addison-Wesley, Reading, MA, 1989.

[90] E. A. Patrick and F. P. Fisher, "Nonparametric Feature Selection," *IEEE Trans. on Information Theory*, vol.15, pp.557-584, 1969.

[91] S. J. Perantonis and P. J. G. Lisboa, "Translation, Rotation, and Scale Invariant Pattern Recognition by High-Order Neural Networks and Moment Classifiers," *IEEE Trans. on Neural Networks*, vol.3, no.2, pp.241-251, March 1992.

[92] W. K. Pratt, *Digital Image Processing*, 2nd Edition, Wiley, New York, NY, 1991.

[93] F. Rosenblatt, "The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain," *Psychological Review*, vol.65, no.6, pp.386-408, 1958.

[94] F. Rosenblatt, *Principles of Neurodynamics*, Spartan Books, Washington, DC, 1962.

[95] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning Representation by Error Propagation," *Parallel Distributed Processing, chap.8*, vol.1, pp.318-362, MIT Press, Cambridge, MA, 1986.

[96] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning Representations by Back-Propagating Errors," *Nature*, vol.323, pp.533-536, October 9, 1986.

[97] D. E. Rumelhart, G. E. Hinton, and J. L. McClelland, "A General Framework for Parallel Distributed Processing," in *Parallel Distributed Processing: Exploration in the Microstructure of Cognition. vol.1: Foundations*, (D.E. Rumelhart and J.L. McClelland, eds.) chap. 5, pp.45-76, MIT Press, Cambridge, MA, 1986.

[98] D. W. Ruck, S. K. Rogers, and M. Kabrisky, "Feature Selection Using a Multilayer Perceptron," *Journal of Neural Network Computing*, pp.40-48, Fall 1990.

[99] J. Rubner and K. Schulten, "Development of Feature Detectors by Self-Organization: A Network Model," *Biological Cybernetics*, vol.62, pp.193-199, 1990.

[100] D. E. Rumelhart and D. Zipser, "Feature Discovery by Competitive Learning," in *Parallel Distributed Processing: Exploration in the Microstructure of Cognition. vol.1: Foundations*, (Ed. by D. E. Rumelhart, J. L. McClelland and the PDP Research Group), chap. 5, pp.151-193, MIT Press, Cambridge, MA ,1986.

[101] I. A. Rybak, N. A. Shevtsova, and V. M. Sandler, "The Model of a Neural Network Visual Preprocessor," *Neurocomputing*, no.4, pp.93-102, 1992.

[102] E. Saund, "Dimensionality-Reduction Using Connectionist Networks," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol.11, no.3, pp.304-314, March 1989.

[103] R. Schaefer and D. Casasent, "Optical Implementation of Gray Scale Morphology," *SPIE vol.1658 - Nonlinear Image Processing III*, vol.3, pp.287-296, 1992.

[104] J. Serra, *Image Analysis and Mathematical Morphology*, Academic Press, New York, NY, 1982.

[105] J. Serra, *Image Analysis and Mathematical Morphology, vol.2: Theoretical Advances*, Academic Press, New York, NY, 1988.

[106] F.Y. Shih and O.R. Mitchell, "Threshold Decomposition of Grayscale Morphology into Binary Morphology," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, pp.31-42, January 1989.

[107] Frank Y. Shih and Jenlong Moh, "Image Morphological Operations by Neural Circuits," in *IEEE 1989 Int'l Symposium on Circuits and Systems*, Portland, Oregon, pp.774-777, May 1989.

[108] Frank Y. Shih and Jenlong Moh, "A Self-Organization Architecture for Clustering Analysis," in *Proc. of 23rd Hawaii Int'l Conf. on System Science*, January 1990.

[109] Frank Y. Shih and Jenlong Moh, "Implementing Neural Morphological Operations Using Programmable Logic," in *Proc. of SPIE Conf. on Intelligent Robots and Computer Vision IX: Neural, Biological, and 3-D Methods*, Boston, MA, November 1990.

[110] Frank Y. Shih and Jenlong Moh, "Improved Adaptive Resonance Theory," in *Proc. of SPIE Conf. on Intelligent Robots and Computer Vision IX: Neural, Biological, and 3-D Methods*, Boston, MA, November 1990.

[111] Frank Y. Shih, Henry Bourne, and Jenlong Moh, "A Neural Architecture Applied to the Enhancement of Noisy Binary Images without Prior Knowledge," in *Proc. of IEEE Inter. Conf. Tools for Artificial Intelligence*, pp.699-705, Herndon, VA, November 1990.

[112] Frank Y. Shih and Jenlong Moh, "Implementing Morphological Operations Using Programmable Neural Networks," *Pattern Recognition*, vol.25, no.1, pp.89-99, January 1992.

[113] Frank Y. Shih, Jenlong Moh, and Fu-Chun Chang, "A New ART-Based Neural Architecture for Pattern Classification and Image Enhancement without Prior Knowledge," *Pattern Recognition*, vol.25, no.5, pp.533-542, May 1992.

[114] Frank Y. Shih, Jenlong Moh, and Henry Bourne, "A Neural Architecture Applied to the Enhancement of Noisy Binary Images," in *Engineering Application of Artificial Intelligence*, vol.5, no.3, pp.215-222, November 1992.

[115] F. Y. Shih and W. T. Wong, "Fully Parallel Thinning with Tolerance to Boundary Noise," *Pattern Recognition*, vol.27, no.12, pp.1677-1695, December 1994.

[116] D. Sinha and E. R. Dougherty, "Fuzzy Mathematical Morphology," *Journal of Visual Communication and Image Representation*, vol.3, no.3, pp.286-302, September 1992.

[117] L. J. Spreeuwers, "A Neural Network Edge Detector," in *SPIE vol.1451 - Nonlinear Image Proc. II*, pp.204-215, 1991.

[118] S. R. Sternberg, "Cellular Computers and Biomedical Image Processing," in *Biomedical Images and Computers*, J. Sklansky and J. C. Bisconte, Eds., pp.294-319, Springer-Verlag, Berlin, 1982. (Also presented at United States - France Seminar on Biomedical Image Processing, St. Pierre de Chartreuse, France, May 27-31, 1980.)

[119] F. W. Stentiford, "Automatic Feature Design for Optical Character Recognition Using an Evolutionary Search Procedure," in *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol.PAMI-7, no.3, pp.349-355, May 1985.

[120] S. Sternberg, "Gray-Scale Morphology," *Computer Vision, Graphics, and Image Processing*, vol.35, pp.333-355, 1986.

[121] M. C. Stinson, "A Neural Network Technique for Feature Extraction to Improve Object Recognition," in *SPIE vol.1192 - Intelligent Robots and Computer Vision VIII: Algorithms and Techniques*, pp.418-424, 1989.

[122] D. G. Stork, "Self-Organization, Pattern Recognition, and Adaptive Resonance Networks," *Journal of Neural Network Computing*, vol.1, pp.26-42, Summer 1989.

[123] C. Y. Suen, J. Guo, and Z. C. Li, "Analysis and Recognition of Alphanumeric Handprints by Parts," *IEEE Trans. on Systems, Man, and Cybernetics*, vol.24, no.4, pp.614-631, April 1994.

[124] C. H. Sung and D. Wilson, "Percognitron: Neocognitron Coupled with Perceptron," *IEEE Int'l Joint Conf. on Neural Networks*,, vol.3, pp.753-758, San Diego, CA, 1990.

[125] T. Sziranyi and J. Csicsvari, "High-Speed Character Recognition Using a Dual Cellular Neural Network Architecture (CNND)," *IEEE Trans. on Circuits and Systems II: Analog and Digital Processing*, vol.40, no.3, pp.223-231, March 1993.

[126] G. Tascini, P. Puliti, and P. Zinggaretti, "Handwritten Character Recognition Using Background Analysis," *SPIE vol.1906 - The Int'l Society for Optical Engineering*, pp.126-133, San Jose, CA, February 1-2, 1993.

[127] A. Thepaut and Y. Autret, "Handwritten Digit Recognition Using Combined Neural Networks," *Int'l Joint Conf. on Neural Networks*,, vol.2, pp.1365-1368, Nagoya, Japan, 1993.

[128] J. T. Tou and R. C. Gonzalez, *Pattern Recognition Principles*, Addison-Wesley, Reading, MA, 1974.

[129] F. Togawa, T. Ueda, T. Aramaki, and A. Tanaka, "Receptive Field Neural Network with Shift Tolerant Capability for Kanji Character Recognition," *Int'l Joint Conf. on Neural Networks*,, vol.2, pp.1490-1499, Singapore, 1991.

[130] T. Tsunoda, T. Shiraishi, and H. Tanaka, "Character Recognition by Associative Completion on Words," *Int'l Joint Conf. on Neural Networks*,, vol.2, pp.1135-1138, Nagoya, Japan, 1993.

[131] J. J. Vidal, "Implementing Neural Nets with Programmable Logic," *IEEE Trans. on Acoustics, Speech, and Signal Processing*, vol.36, no.7, pp.1180-1190, July 1988.

[132] T. Wakahara, H. Murase, and K. Odaka, "On-Line Handwriting Recognition," *Proc. of the IEEE*, vol.80, no.7, pp.1181-1194, July 1992.

[133] P. S. P. Wang and Y. Y. Zhang, "A Fast and Flexible Thinning Algorithm," *IEEE Trans. on Computers*, vol.38, pp.741-745, May 1989.

[134] B. Widrow and M. Hoff, "Adaptive Switching Circuits," *1960 IRE WESCON Convention Record*, IRE, New York, NY, pp.96-104, 1960.

[135] B. Widrow and R. Winter, "Neural Nets for Adaptive Filtering and Adaptive Pattern Recognition," *IEEE Computer*, vol.21, pp.25-39, March 1988.

[136] B. Widrow, R. G. Winter, and R. A. Baxter, "Layered Neural Nets for Pattern Recognition," *IEEE Trans. on Acoustics, Speech, and Signal Processing*, vol.36, no.7, pp.1109-1118, July 1988.

[137] S. S. Wilson, "Vector Morphology and Iconic Neural Networks," *IEEE Trans. on Systems, Man, and Cybernetics*, vol.SMC-19, no.6, pp.1636-1644, November/December 1989.

[138] M. J. Wu, "Fuzzy Morphology and Image Analysis," *Proc. of 9th Int'l Conf. on Pattern Recognition*, Rome, Italy, pp.453-455, 1989.

[139] T. Y. Young and K. S. Fu, *Handbook of Pattern Recognition and Image Processing*, Academic Press, New York, NY, 1986.

[140] T. Yoshida, "Construction of a Feature Set for Character Recognition," *Int'l Joint Conf. on Neural Networks,*, vol.3, pp.2153-2156, Nagoya, Japan, 1993.

[141] L. A. Zadeh, "Fuzzy Sets," *Information and Control* vol.8, pp.338-353, 1965.

[142] X. Y. Zhu, K. Yamauchi, T. Jimbo, and M. Umeno, "Handwritten Character Recognition by a Layered Neural Network," *Systems and Computers in Japanese*, vol.21, no.13, pp.88-97, 1990.

[143] J. M. Zurada and K. Jagiello, "Recognitron — A Neural Net Model for Character Recognition," *1992 IEEE/INNS Int'l Joint Conf. on Neural Networks*, vol.3, pp.637-642, Baltimore, MD, 1992.