INFORMATION TO USERS

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

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

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

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

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

ProQuest Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

UMI®

MODULAR NEURAL NETWORK IN IMAGE PROCESSING

By

Min Su

A dissertation submitted to the Graduate Faculty in

Engineering in partial fulfillment of the requirements for the degree

of Doctor of Philosophy, The City University of New York

2002

This manuscript has been read and accepted for the Graduate Faculty in Engineering in satisfaction of the dissertation requirement for the degree of Doctor of Philosophy.

4/22/02 — Date

Chair of Examining Committee

4/22/02 — Date

Excutive Officer

PROFESSOR JOSEPH BARBA

PROFESSOR EDWARD CIACCIO

PROFESSOR KI H. CHON

PROFESSOR YI SUN

Supervisory Committee

THE CITY UNIVERSITY OF NEW YORK

Abstract

Modular Neural Network in Image Processing

by

Min Su

Advisor: Dr. Mitra Basu

In this dissertation, we study the application of a *learning and adaptive systems* to image restoration problem. This thesis is divided into three parts. In the first part, we focus on a specialized learning system - the projection pursuit learning network (PPLN). We study its capabilities and limitations in the context of noisy and blurred images. A comparison with the commonly used Wiener filter is included to put the learning system's performance in perspective.

In the second part, we digress to get a better understanding of the role of smoothing functions in image restoration problem. Here, we propose a family of exponential functions to smooth noisy images. We observe that optimal results are obtained when the value of the exponent lies within a certain range. We establish that the range found is implicitly present in the structure of all natural images. Furthermore, we discover that this range has been found to be useful in applications other than image smoothing.

In the last part, we consider modular learning network for image restoration problem. We propose a hybrid learning method to combine unsupervised and supervised learning algorithms. We study the complex task of input clustering, a range of gating functions and a novel way to extend the role of input to achieve superior restored images. A new quantitative measurement technique to rank the restored images, that agrees with visual inspection of

images, is proposed here.

Acknowledgments

I consider myself fortunate to have had the opportunity to study under the guidance of Professor Mitra Basu. I greatly appreciate her unrelenting work to stay at the top of the field. Her constant availability, her bright insights, her deep involvement, and her creative imagination are the reason why this work has been possible.

I would like to express my sincere appreciations to Dr. Joseph Barba, Dr. Edward Ciaccio, Dr. Ki H. Chon and Dr. Yi Sun for kindly agreeing to serve in my doctoral dissertation committee.

I am grateful to all members of the faculty who helped me in many different ways during my stay at the department.

I remain indebted to all my friends for their advice, ideas and help during the course of this research work.

I would like to thank my parents for all their love and support.

Contents

List of Figures

List of Tables

Chapter 1

Preface

> "Everyone knows what attention is. It is the taking possession of the mind, in clear and vivid form, of one out of what seem several simultaneously possible objects or trains of thought. Focalization, concentration of consciousness are of its essence. .....", by James [James 1890].

Psychological studies of human visual system have shown that human visual system is one of the most complex system in the world. It has extraordinary abilities to categorize all kinds of images. These abilities seem all the more impressive when we consider that people can accurately identify an image which is noisy and blurred. How we do it, and how we can program a computer to do it, are topics of current research.

One of the major benefits of the digital computer is the speed at which it can complete a task. The simplest digital computer is able to perform mathematical calculations millions or billions of times faster than the human brain. In contrast, these same digital computers are usually unable to perform many pattern processing tasks that are trivial for the human

brain. The computer technology developed in the future is going to combine the speed of the digital computer with the pattern processing capability of the brain to create a more naturally intelligent system. These neural networks, or neurocomputers, will have the ability to perform tasks that would be difficult or impossible to do with convention digital computers. The main hope is that these neural networks will be able to reproduce some of the flexibility and power of the human brain by artificial means. There is potential in this technology to start a new revolution in the computer industry and in the society.

Neural networks will eventually be able to outperform the human brain and digital computers in many pattern recognizing tasks. These tasks will also be completed more rapidly, economically, and reliably. This technology will be able to do jobs that are too dangerous for humans, but need human judgement and learning ability. In addition, the neural network technology will create a whole new set of applications that did not exist at the time it was conceived. A new niche will be created for neural systems just as it was for such technological inventions as the telephone, the automobile, and the microprocessor. Neural networks will eventually become an essential part of society and the everyday existence of the common person.

What is a neural network? A neural network is basically a type of information processing system whose architecture is inspired by the structure of the brain. The brain processes information through a network of millions of essentially simple processing units called neurons. In the brain, each neuron receives an input signal from a large number of other neurons and these inputs are combined and sent as output signals to other neurons. The final signal depends on the strength and construction of the input signals. The neural network systems is designed to simulate the actions and parallel nature of the brain, but are different from

the traditional, parallel architectures. Neural networks perform computations using a dense group of interconnected computing nodes. These nodes operate collectively and simultaneously on all data and inputs which explains the similarity between neural nets and other parallel architectures. An important difference between neural networks and standard IT (information and technology) solutions is their ability to learn. This learning property has yielded a new generation of algorithms that can:

1. learn from the past and predict the future;
2. extract rules for reasoning in complex environments;
3. offer solutions when explicit algorithms and models are unavailable or too cumbersome.

Based on these principles, neural networks have yielded many successful applications in branches as diverse as finance, retail and logistics, medicine and health, marketing, manufacturing and industrial control, and energy and utility.

In contrast to conventional computer systems, neural networks are not programmed with a computer language, but are trained to behave in a certain way. Neural networks are trained using learning rules and algorithms and this replaces the programming that is necessary with conventional computers. Users are responsible for setting up the architecture of the network, specifying the features desired in the nodes, and determining the desired training method for the network. The neural network is then able to begin learning from the entered data and assimilate information that can be used later by the user and the neural network. In this process of learning, neural networks begin to exhibit a capacity to generalize. Capacity to generalize is a trait that ends up being very beneficial and is a major advantage that neural networks possess over conventional computer systems.

In theory, an ideal neural network will exhibit a set of specific characteristics. In the real world, an ideal neural network has not yet been developed. This means there are computer systems that are considered to be neural networks even without exhibiting all of the ideal characteristics of a neural network. An ideal neural network [2]:

1. contains a number of processing elements which communicate through interconnections with variable weights or strengths.

2. stores or represses information as memories represented in a synaptic weights of interconnected neurons. Information is processed by changing pattern of activity in the network.

3. is trained instead of programmed. Some systems are capable of independent learning or learning through trial and error.

4. has three properties that controls it operation: a transfer function of the processing elements, the structure of connections between the processing elements, and the system of learning for the network.

5. has the capacity to assimilate items it is taught and can generalize by physically grouping similar items together. It can also retrieve stored information from incomplete, noisy, or partially incorrect input.

6. is highly fault tolerant and its performance degrades slowly and smoothly as processing elements and connections fail.

7. innately acts as a processor for time-dependent spatial patterns, or spatiotemporal patterns.

8. can self organize with the ability to generalize from training without instructions on exactly what to learn.

These eight specific characteristics of an ideal neural network would be very advantageous in a computer technology. This is why neural networks have attracted the attention of scientists and technologists from a number of disciplines. The advantages of a technology with the strengths of the human brain and the speed of a digital computer are innumerable and are being developed in neural networks.

Traditional digital signal processing (DSP) is based on *algorithms*, changing data from one form to another through step-by-step procedures. Most of these techniques also need *parameters* to operate. For example: recursive filters use recursion *coefficients*, feature detection can be implemented by correlation and *thresholds*, an image display depends on the *brightness and contrast settings*, etc. Algorithms describe what is to be done, while parameters provide a benchmark to judge the data. The proper selection of parameters is often more important than the algorithm itself. Neural networks take this idea to the extreme by using very simple algorithms, but many highly optimized parameters. This is a revolutionary departure from the traditional mainstays of science and engineering: mathematical logic and theorizing followed by experimentation. Instead of studying the most complex mathematical representation, neural networks use approximation to simulate the result. Which means neural networks replace those problem solving strategies with *trial & error, pragmatic solutions* and a *this works better than that* methodology.

1.1 Problem Description

This dissertation is focused on the study of using neural network like learning systems to process degraded image. We address the issues in image recovery/restoration. Problem of image recovery is an important research topic. The goal is to reconstruct the image from its degraded version. Most methods found in the literature assume some mathematical model for the degradation process. Thus, these methods produce acceptable results only when the underlying assumption is true. The following reasons motivated us to formulate a different approach to this problem.

- Degradation sources are not known *a priori*
- There are infinitely many real sources of image degradation. Thus it is not realistic to use a model-based approach.
- Humans are superior at *blocking* the degradation effect. One possible way to explain this ability is to realize that learning plays an important role here.
- Recently, there has been a surge of activities in computational learning theory. Many interesting results are available for real-world applications i.e., these algorithms are tractable (at least in some sense).

Systems that learn are not new to the engineering community. The family of Neural network-like systems have been applied to many engineering problems with promising results.

This dissertation consists of six chapters. Chapter 2 describes the properties of PPLN network and the Hermite polynomial function. A summary of PPLN network training procedure is given. The application of this network in deblurring image, noise remove and the recovery

image from its noisy and blurring model are studied and the comparison result of this network with the Wiener Filter is shown at the end of this chapter. Noise reduction in images, also known as image smoothing, is an essential and first step before further processings are done on the image. The key to image smoothing is to preserve important features while removing noise from the image. Gaussian function is widely used in image smoothing. Recently it has been reported that exponential functions (value of the exponent is not equal to 2) perform substantially better than Gaussian functions in modeling and preserving image features. In chapter 3 we propose family of exponential functions, that include Gaussian, for image smoothing. Experimental results with a variety of images, artificial and real, demonstrate that optimal results are obtained when the value of the exponent is within a certain range. In chapter 4, the frequency properties of three activation functions: RBF, sigmoid function, and Hermite function, which used in chapter 5, are discussed. Chapter 5 is devoted to modular network, clustering rules and gating functions, and their applications to image restoration. A brief summary and the future work are contained in the last chapter.

Chapter 2

Projection Pursuit Learning Networks

The goal of this chapter is to study the capabilities and the limitations of **Projection Pursuit Learning Network** (PPLN) in the context of problems that arise in image processing research area. PPLN is a generalized version of one hidden-layer sigmoidal feed-forward neural network. Although PPLN is used extensively for data analysis in statistics, it is not well-known in the image processing community. Our goal is to study this unexplored method and evaluate its performance on *appropriate* problems. There are a number of issues involving PPLN's implementation (e.g., choice of parameters, convergence etc.).

2.1 Overview

We select **Projection Pursuit Learning Network** and evaluate its performance on two specific image restoration problems - (1) deblurring (2) noise-removal. Preliminary inves-

tigation supports our assumption that systems with learning capability outperforms other model-based methods found in the literature.

The objective of this chapter has been stated. We describe the problem of image restoration and present a survey of related work in next section. In section 2.2.1, we introduce the concept of projection pursuit regression and projection pursuit learning network, present a brief literature survey and discuss in detail the various steps of the PPLN algorithm. Issues on implementing this algorithm is also addressed in this section. Section 2.3 contains the experimental results and comparison with other known methods. A summary is provided in the last section.

2.2 Image Restoration - A Brief Overview

In image restoration, the objective is to reduce or eliminate the image degradation from where it has been degraded in some manner. The most common types of degradation are due to noise and blurring.

Blurring is hard to avoid in any image acquisition system, and can be caused by many sources such as: (1) atmospheric turbulence, (2) an out-of-focus optical system (which results in the point spread function of the imaging system not being an impulse function), and (3) aberrations in the imaging system. Obviously, all these cause cannot be simultaneously captured in a simple model. However, by the central limit theorem of probability theory, the net result of the complex interplay among these independent random sources can, in general, be approximated by a Gaussian blur [84].

Before defining any solution, we must know how the image was formed, i.e. we must have a

mathematical model of the image formation system. For this work, the linear shift invariant model will be used. The linear model gives the following relation:

$$b(m,n) = h(m,n) * f(m,n) \tag{2.1}$$

where b, h and f are the blurred image, the point-spread function (PSF) of the blurring system and the original image, respectively. The aim of image restoration is to recover $f(m,n)$ from the information in $b(m,n)$.

In order to extract the original image $f(m,n)$ from the degraded image $b(m,n)$, a deconvolution is required. In the frequency domain, this can be expressed as an inverse filter:

$$B(u,v) = H(u,v)F(u,v)$$

The inverse or restoring filter, $Q(u,v)$, may be easily found from $H(u,v)$:

$$Q(u,v) = \frac{1}{H(u,v)}$$

However, it is very hard to determine and implement the inverse filter. For this reason an alternative to finding an inverse operator for image restoration is of interest [83].

Several researchers have considered the role of polynomials in deblurring images. One possible way of approaching the problem was considered in [35, 58]. Hummel *et al.* [35] noted that the process of reversing blur is unstable and cannot, in general, be represented as a convolution filter in the spatial domain. They posed the deblurring problem in a variational framework, and constrained the spaces of the original and blurred signals, in an attempt to convert it to a *well-conditioned* problem. They demonstrated that it is possible to invert the Gaussian blur in a stable manner, if the class of input functions is restricted to polynomials

of fixed finite degree. They assume a convolution form for the inverse operator so that the inversion may be realized as a filter operation. The aim was to find an inverse filter which when convolved with the blurred function restores it to the original. The inverse (or deblur) filter $D_N(x)$, which involves a polynomial of degree N, restores the blurred function

$$f(x) = b(x) * D_N(x)$$

Martens [58] presented an algorithm for deblurring and interpolating digital images. He assumed that a signal can be locally described by polynomial coefficients, and that these coefficients can be estimated from the sampled signal $S(kT)$ by means of digital filters H_n, for $n = 0, \ldots, N$. These estimated coefficients can then be used to make a deblurred estimate of the original signal. The resulting signal estimate is given by:

$$L(x) = \sum_j S(jT) \sum_{n=0}^{N} I_n(x - jT)$$

where

$$I_n(x) = \sum_k H_n(k) P_n(x - kT)$$

is the n-th order deblurring filter, and P_n are pattern functions. The overall deblurring function

$$I(x) = \sum_{n=0}^{N} I_n(x)$$

can be controlled by the order N.

The other common source of degradation in image is the presence of noise [85]. Images acquired with an electronic camera are typically corrupted with noise due to the camera's

sensor and its associated electronics. The point-to-point transmission of video images is another common source of noise. Depending on environmental conditions, these images are very often degraded by noise. The common models of noise found in the literature are

- **Uniform Noise:** This produces noise values with equal probability in a specified range.
- **Gaussian Noise:** Gaussian distribution is often used to model the noise (present in an image) due to many independent noise sources.
- **Exponential Noise:** Negative exponential noise appears in images that have been acquired using a laser as an illumination source.
- **Salt & Pepper Noise:** Typically, the cause of this type of noise can be traced to two sources - (1) for images captured with electronic cameras, malfunctioning pixels usually produce pixels with gray-levels of either white or black, (2) dust and lint that appears on the optics during image acquisition.

2.2.1 Projection Pursuit

A Brief Survey on Projection Pursuit Regression

Projection pursuit has become one of the most exciting multivariate data analysis methods in statistics because it has the potential to reduce difficulties due to the "curse of dimensionality" in non-parametric statistics by working in low-dimensional linear projections. The original projection pursuit approach was called exploratory projection pursuit and looked for " interesting" low-dimensional projections of higher dimensional data by numerically

maximizing a projection index.

In 1981, Friedman and Stuetzle [21] introduced projection pursuit regression (PPR) as a new technique for non-parametric multivariate regression. In a regression problem, there is a p-dimensional random vector $\mathbf{X}$, whose components are called predictor variables, and a random variable Y, called the response. A regression surface depicts a general relationship between $\mathbf{X}$ and Y. Traditionally, in addressing the regression problem, it was usually assumed that the functional form of the regression surface was known, thereby minimizing the problem to one of determining a set of parameters. However, it is usually difficult to prove the results, and a wrong assumption can cause incorrect or misleading results. This makes the use of non-parametric methods which make only a few general assumptions about the regression surface highly desirable.

Friedman and Stuetzle's concept of projection pursuit regression avoided many difficulties experienced with other existing non-parametric regression procedures. The poor performance in high dimensions (curse of dimensionality) encountered in kernel and nearest-neighbor methods is avoided in PPR since all estimation (smoothing) is carried out in a univariate setting. Unlike recursive partitioning, PPR does not split the predictor space into two regions thereby allowing, when necessary, more complex models. In addition, interactions of predictor variables are directly considered since linear combinations of the predictors are modeled with general smooth function [21]. Another significant property of PPR is that the results of each iteration can be depicted graphically. The graphical output can be used to modify the major parameters of the procedure – the average smoother bandwidth and the terminal threshold. The PPR method can also be applied to the residuals from any initial model. If the initial model does not fit the data well, PPR will indicate this by augmenting

the model [21].

All stepwise procedures have difficulties modeling regression surfaces that can not be adequately described by models of low complexity in their hierarchy. Because models in PPR are sums of functions, each varying only along a single linear combination of the predictor variables, PPR has difficulty modeling regression surfaces that vary with equal strength along all possible linear combinations [21].

Diaconis and Shahshahani [14] examined PPR theoretically. They addressed the projection pursuit algorithm from the perspective of the approximation theory. They analyzed the necessary and sufficient conditions for the functions to be exactly represented as a linear combination of nonlinear functions, and discussed the nonuniqueness of the represnetation.

Donoho and Johnstone [15] studied the duality between PPR and kernel regression in two dimensions. Their results indicated that PPR produces function extimates which behave well when the underlying function is angularly smooth (oscillating slowly with angle), while kernel regression was suitable for functions with sufficient Laplacian smoothness, such as harmonic functions and solutions to the inhomogeneous heat equation (oscillations averaging out locally). They also showed that if the function to be estimated has nice tail behavior, PPR lowers the dimensionality of the problem. In their case, it behaved as if the dimension of the problem were 1.5 rather than 2. PPR and kernel regressions turn out to complementary: for a given function, if one method offers a dimensionality reduction, the other does not.

Hall [29] devised a tractable mathematical model to describe PPR. The model allows computation of explicit formulae for bias and error about the mean in orientation estimates and curve estimates. The results indicated that the estimate of orientation has most of its error in

the bias, and that the error about the mean is asymptotically negligible in comparison to the bias. Hall also proved that the common form of kernel-based PPR does estimate projections with the same convergence rates as those experienced in one-dimensional problems, although a greater degree of smoothness (an extra derivative) must be assumed to accomplish this.

Chen [11] proposed a PPR scheme with two additional constraints imposed so that the rate of convergence of the estimator is independent of dimensionality. He concluded that to resolve the conflict between the necessity for flexible modeling and the curse of dimensionality due to p-dimensional local averaging, there needs to be a compromise between a parametric regression model and a nonparametric regression model. Let $(\mathbf{X}, Y)$ be a random vector such that $\mathbf{X} = (\mathbf{X_1}, \ldots, \mathbf{X_p})^T$ ranges over R^p. The regression function of Y on $\mathbf{X}$ is $E(Y/\mathbf{X} = \mathbf{x})$, and is assumed to be the sum of no more than p general smooth functions of ${\beta_i}^T\mathbf{x}$, where $\beta_i \in S^{p-1}$, the unit sphere in R^p centered at the origin. Under suitable conditions, the rate of convergence of the proposed estimator is independent of p.

A Brief Survey on Projection Pursuit Learning Networks

Projection pursuit was first introduced in the context of learning networks by Barron and Barron [3]. They analyzed several network schemes and algorithms based on the definition that a *learning network* estimates its function from representative observations of the relevant variables. They considered three types of methodologies – neural networks, adaptive polynomial learning and nonparametric statistical inference – and found substantial similarities in these fields. They found that the most successful learning network strategies adaptively grew the network structure using all the observational data, and used a suitable model selection criterion to ensure a parsimonious network. The best methodologies also

used network structures which were not limited in their approximation capabilities. The primary examples of these successful methodologies were adaptively synthesized polynomial networks and projection pursuit.

Barron and Barron [3] stated that an advantage of projection pursuit networks is that they have been amenable to theoretical examination of some of their approximation properties [15, 32, 44]. In particular, it is known that any square integrable function can be approximated by a theoretical analog of projection pursuit provided sufficiently many levels of the network are utilized.

Intrator [38, 39] applied exploratory projection pursuit to do feature extraction in speech recognition. He defined a measure of multimodality for finding interesting projections based on the observation that high-dimensional clusters translate to multimodal low-dimensional projections. When the data is known in advance to be bimodal, it is relatively straightforward to define a good projection index. However, Intrator considers the case when the structure is not known in advance and defining a general multimodal measure of the projected data is not straight forward.

Intrator formed an objective function whose minimization finds projections with one - dimensional projected distributions that are not Gaussian. This was done using a loss function that has an expected value that leads to the desired projection index. Consider a neuron with input vector $x = (x_1, \ldots, x_N)$ and synaptic weights $w = (w_1, \ldots, w_N)$, the loss function is given by

$$L_w(x) = \frac{-\mu}{3}\left\{\sigma^3(x \cdot \omega) - E[\sigma^2(x \cdot \omega)]\sigma^2(x \cdot \omega)\right\} \tag{2.2}$$

where μ is the learning rate and σ is usually a sigmoidal function.

Based on this formulation, a network can be created that consists of identical nodes which all receive the same input and inhibit each other so as to extract features in parallel.

Intrator compared the classification performance with speech data of the proposed method and a backpropagation network. He found that less features were extracted with the back-propagation network, as this method concentrated mainly on one particular feature of the speech signal and stopped training when the misclassification error fell to zero. The proposed network did not try to reduce the misclassification error and was therefore able to find a significant structure in the speech signals.

Maechler *et al.* [57] and Hwang *et al.* [36, 37] studied several two-dimensional examples to compare projection pursuit learning network (PPLN) and backpropagation neural network (BPNN). They limited their study to the regression problem because there is a better theoretical understanding of the approximation quality of PPLNs. They point out that both types of networks form projections of the data in directions determined from the interconnection weights. However, a BPNN uses a fixed set of nonlinear activations that are parametrically specified by a fixed finite set of parameters, whereas a PPLN estimates nonlinear activations based on an optimization scheme that uses a one-dimensional non-parametric data smoother.

In a simulation study, Maechler *et al.* [57] found that PPLNs and sequential gradient descent BPNNs have similar accuracy for independent test data, but PPLNs are two orders of magnitude faster in training. They concluded that the difference in speed was due to the fact that smoothing in the BPNN involves using many parameters estimated by gradient search. PPLN, on the other hand, uses an efficient optimization strategy. Least squares is

used to determine the linear part of the weights, and the second Gaussian method is used to determine the nonlinear part. Further more, data-smoothing techniques allow fitting non-parametric nonlinear modal functions, so the network adapts more quickly to the observation data.

When Hwang *et al.* [36] compared PPLN and batch Gauss Newton BPNN, they found that both networks have comparable training speed and accuracy for independent test data. However PPLNs are significantly more parsimonious as they require fewer neurons.

In the original PPLN for regression, a variable span smoother, called the super-smoother, was used to obtain smooth estimated activation functions. Hwang *et al.* [36, 37] pointed out that non-parametric smoothers lead to the use of large regression tables, unstable approximation in calculating derivatives, and piecewise interpolation in computing activation values. As a result, they proposed using parametric smoothers that are a linear combination of Hermite functions. The Hermite polynomial approximation of the non-linear activations outperforms the super-smoother version of the PPLN in several aspects of performance evaluations.

Hwang *et al.* [37] also compared a cascade-correlation learning network (CCLN) and PPLN. A CCLN is a unit-growing learning network that increases the size and depth of its hidden layer during training. Like a PPLN, a CCLN forms new candidate hidden units one at a time. The weights of each candidate unit are trained by keeping the weights of the other existing hidden units constant. Unlike a PPLN in which candidate units receive only weighted connections from all the input units, a CCLN's candidate unit receives weight connections from all the input units as well as all the other existing hidden units. This enables a CCLN to detect higher-order features.

Hwang *et al.* [37] found that the cascaded connections used to enable higher-order feature detection in a CCLN are not necessary in a PPLN, since the trainable activations have the same purpose. Training in a CCLN is also more difficult as the cascaded weights increase the input dimensions of the candidate hidden units using a fixed simple nonlinearity activation. A PPLN, on the other hand, works with the same input dimensions, but uses appropriate hidden activation functions. Furthermore, the maximum correlation criterion used in a CCLN to avoid cyclic updating between layers of weights, usually produces saturated hidden units. This results in unsmooth, zigzag classification/regression surfaces, making the CCLN unsuitable for use in most regression applications. The Hermite-based PPLN, on the other hand, uses the minimum L_2 criterion and produces smoother classification/regression surfaces.

Flick *et al.* [18] use a projection pursuit methodology for pattern classification which is based on likelihood ratio estimation. The authors chose to use this approach because the likelihood ratio is an optimal discriminant in a Neyman-Pearson sense, and the Neyman-Pearson classifier requires only a monotone function of the likelihood ratio. Consequently, they need to estimate only a single function rather than work with the ratio of two estimated probability density functions. The monotone function they chose to estimate is bounded above and below, and presents few numerical problems. They show that a measure of scatter can be associated with this function, and that minimization of this scatter is equivalent to finding the optimal discriminant.

Flick *et al.* [18] discuss two algorithms – linear series and nonlinear series – for obtaining discriminant functions from training set data. The projection pursuit scheme in both algorithms is iterative, and at each iteration, a single new ridge function (estimator) and

associated projection direction is introduced. In the linear series algorithm, a linear combination of the ridge functions presented up to the current iteration is used to estimate the monotone function. In the nonlinear series algorithm, a well chosen nonlinear combination of ridge functions is used. In both cases, the direction of the new projection and associated ridge function are chosen to minimize the measure of scatter estimated from the training set.

Zhao *et al.* [88] investigated the implementation issues of projection pursuit regression (PPR). They applied parametric PPLNs to learn the inverse dynamics of robot arms. This is a highly non-monotonic, pure approximation problem. They use the robot arm example to examine the performance of PPR in a high-dimensional space ($d = 6$).

The authors showed that a PPLN can learn simple arm dynamics fairly well, and that parametric PPR with a direct training method achieves better accuracy and training speed than non-parametric PPR. The parametric representation for PPR that they proposed is

$$\hat{y}_i = \sum_{j=1}^{n} \beta_j f_j(\alpha_j{}^T \mathbf{x} + \theta_j) \tag{2.3}$$

where θ_j is the bias term.

They also show experimentally that this modification increases the rate of convergence with respect to the number of hidden units and improves the generalization performance.

[66, 67, 63] proposed a new compression algorithm that fits various artificial neural network models to different segmented image blocks utilizing the theory of PPLNs. The segmentation phase of their algorithm divides the image into variable-size square blocks based on a measure of activity within the block. The size of the finest block is dictated by the combination of

the desired bit rate and the desired peak signal-to-noise ratio.

Once the image has been segmented into various size regions, each block can be coded using the projection pursuit image approximation technique. In every iteration in this phase of the algorithm, a function is selected that best approximates the current image in the given block, from a set of fixed pre-determined basis functions. In the first step of the iteration, the current image is the original image and in the k-th step the current image is the residual image that results from subtraction of the original image and linear combination of all the $(k-1)$-th previous approximations. The optimum parameter for each block must then be quantized before encoding. Based on the distribution of the weights and the biases and their dynamic ranges for each block, separate quantizers were designed for each set of parameters at each iteration of the algorithm. The resulting quantized parameters are then coded using an arithmetic coder. Experimental results show that at rates below 0.5 bits/pixel, this algorithm performs better than JPEG in terms of peak signal-to-noise ratio and subjective image quality.

Hulle [34] proposed a novel approach to non-parametric regression analysis using topographic maps. The maximum entropy learning rule was extended with a neighborhood function, and the extended rule, called eMER was applied in combination with projection pursuit regression learning. The high-dimensional data were interpreted through optimally chosen lower dimensional projections in which topographic maps were trained using eMER. The position of the neurons in the maps were joined using non-smoothing cubic splines.

Hulle applied the eMER/PPR combination to adaptive filtering of gray-scale images to verify the regression performance in high-dimensional spaces. The eMER/PPR combination was trained on a noisy sub-image, then the regression model obtained at converge was applied

on the full noisy image. Experimental results show that the model obtained after training reduces the noise content of the full image by more than 20 dB.

Structure of Projection Pursuit Learning Networks

The projection pursuit learning (PPL) is a statistical procedure proposed for multivariate data analysis using a two-layer network. This procedure derives its name from the fact that interprets high dimensional data through well-chosen lower-dimensional projections. The "pursuit" part of the name refers to optimization with respect to the projection directions.

A projection pursuit learning network (PPLN) possesses a structure very similar to a one hidden-layer neural network (with sigmoidal nonlinearity) (see: Fig:2.1). In the case of the PPLN, the sigmoidal functions are replaced by unknown functions to be learned from the data. Since the sigmoidal functions are replaced by more general functions, PPLN can be viewed as a generalization of a one hidden-layer sigmoidal feed-forward neural network.

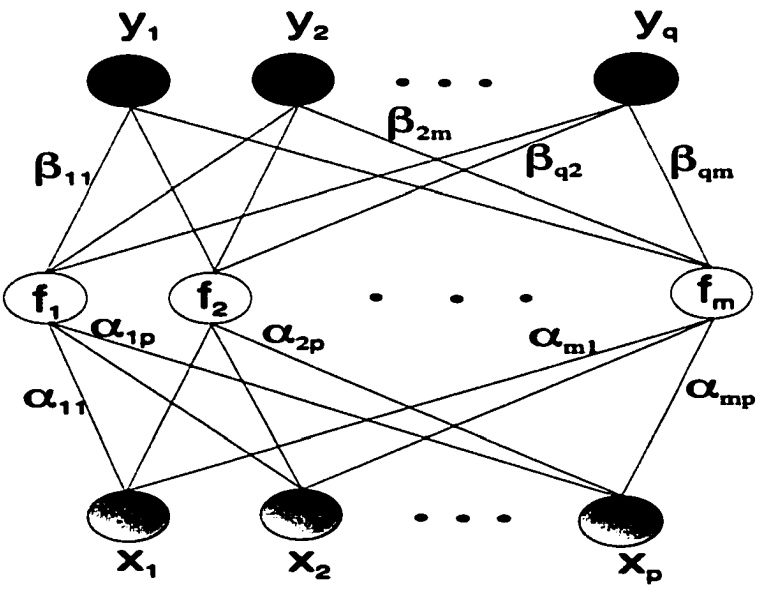


Figure 2.1: One hidden layer neuron network

What we want to do is to build a network model $F(x)$ that can be used as follows. When a vector $\mathbf{x}_i, i = 1, \ldots, n$, is put to the network, a corresponding output $\mathbf{y}_i, i = 1, \ldots, n$ is obtained. Where $\mathbf{x_i}$ is a p-dimensional vector and $\mathbf{y_i}$ is a q-dimensional vector. That is:

$$\hat{\mathbf{y}}_i = F(\mathbf{x_i})$$

and

$$\mathbf{y}_i = \hat{\mathbf{y}}_\mathbf{i} + \varepsilon_i, \qquad i = 1, \ldots, n$$

where $\mathbf{y_i}$ is the desired response of the input $\mathbf{x_i}$, $\hat{\mathbf{y}}_i$ is the actual network output for input $\mathbf{x_i}$ and ε_i is the error value.

After "enough" data has been selected to train the network, F is expected to give a good performance for any new set of $\mathbf{x}$. The quantity ε is a measure of the amount of error. With a large ε, the network performance may not be acceptable. With a small ε, it may be difficult to train the network.

Mathematical Formulation

The basic idea behind projection pursuit regression that we are using is described in [37, 80] for image processing.

Let $\mathbf{y}_i = (y_{i1}, \ldots, y_{iq})$ and $\mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{ip})^T, i = 1, \ldots, n$ denote the response and predictor vector, respectively. The parameters $\alpha_j = (\alpha_{j1}, \ldots, \alpha_{jp})$, where $j = 1, \ldots, M$ and $\beta_k = (\beta_{k1}, \ldots, \beta_{kM})$, where $k = 1, \ldots, q$ are the adjustment vectors.

Suppose you have observation $\mathbf{y}_i$ and corresponding predictors $\mathbf{x}_i$, where $i = 1, 2, \ldots, n$. Let α_j denote p-dimensional unit vectors, as "direction" vectors, and let $\bar{y}_l = \frac{1}{n}\sum_{i=1}^{n} y_{il}$,

$l = 1, 2, \ldots, q$, denote the sample average over all the n training data. The projection pursuit algorithm allows you to find $m = m_0$, direction vectors $\alpha_1, \alpha_2, \ldots, \alpha_{m_0}$ and good nonlinear transformations $f_1, f_2, \ldots, f_{m_0}$ such that

$$\hat{y}_i = \bar{y}_i + \sum_{k=1}^{m} \beta_{ik} f_k(\sum_{j=1}^{p} \alpha_{kj} \mathbf{x}_j) \tag{2.4}$$

where $\beta_{ik}, i = 1, \ldots, q$ are weights connecting the i^{th} output neuron to the k^{th} hidden neuron, f_k is the unknown (trainable) "smooth" activation function of the k^{th} hidden neuron, and $\alpha_{kj}, j = 1, \ldots, p$ denote the hidden-layer weights connecting all the input units to the k^{th} hidden neuron. Since there are normally more than one output, it is advantageous to normalize the hidden outputs f_k for the sake of optimization and network pruning. Therefore, the output weights β_{ik} are required in the network.

The coefficient β_{ik}, α_{kj} and the activation function f_k are parameters of the model and are estimated by least squares error; the estimates are chosen to be those values that minimize the L_2 as shown below:

$$\begin{aligned} L_2 &= \sum_{i=1}^{q} W_i E(y_i - \hat{y}_i)^2 \\ &= \sum_{i=1}^{q} W_i E[y_i - \sum_{k=1}^{m} \beta_{ik} f_k(\alpha_k^T \mathbf{x}_i)]^2 \end{aligned} \tag{2.5}$$

The response weights W_i allows the user to specify the relative contribution of each output prediction square error $E[y_i - \hat{y}_i]^2$ to the total L_2 loss. Typically, one chooses

$$W_i = 1/var(y_i)$$

in case where the y_i have different variances $Var(y_i)$, and these variances are either known

or (more typically) estimated from the data.

A PPLN builds up its network, in a batch learning manner, by growing candidate hidden units one at a time. The PPLN starts from updating the parameters associated with the first hidden unit by updating each β_{i1}, f_1, and α_{1j} cyclically (layer-by-layer) to minimize the loss function L_2 . When the L_2 stops decreasing, it then adds in the second hidden neuron by cyclically updating β_{i2}, f_2, and α_{2j}. A complete updating ends at the updating of the parameters associated with the m^{th} (the last) hidden neuron by consecutively updating β_{im}, f_m, and α_{mj}. Normally, the stopping criterion is chosen to be $(|L_2^{(new)} - L_2^{(old)}|)/L_2^{(old)}$ smaller than a prespecified small constant.

$$|\frac{L_2^{(new)} - L_2^{(old)}}{L_2^{(old)}}| < \varepsilon_1 \tag{2.6}$$

where ε_1 is the value that we select to stop the current hidden neuron update and move to the next step (i.e., either add the next hidden neuron or stop the update process).

The model that takes the form of (2.4) are termed SMART for Smooth Multiple Additive Regression Technique. Details of the algorithm for minimizing (2.5) for SMART is discussed below and also sees [37].

We consider growing the k^{th} hidden unit as an example. The L_2 loss criterion function could be rewritten as the residual function $R_{i(k)}$:

$$R_{i(k)} = y_i - \sum_{j \neq k} \beta_{ij} f_j(\alpha_j^T \mathbf{x})$$

and

$$\begin{aligned} L_2 &= \sum_{i=1}^{q} W_i E[R_{i(k)} - \beta_{ik} f_k(\alpha_k^T \mathbf{x})]^2 \\ &= \frac{1}{n} \sum_{l=1}^{n} \sum_{i=1}^{q} W_i [R_{li(k)} - \beta_{ik} f_k(\alpha_k^T \mathbf{x}_l)]^2 \end{aligned} \tag{2.7}$$

Assume for $j \neq k, \beta_{ij}, i = 1, \ldots, q, a_j = (\alpha_{j1}, \ldots, \alpha_{jp})^T$ and the function f_j are fixed.

For given f_k and α_k we have

$$\frac{\partial L_2}{\partial \beta_{ik}} = 0$$

and for fixed β_{ik} and α_k, we have

$$\frac{\partial L_2}{\partial f_k} = 0$$

Then we can obtain $\hat{\beta}_{ik}$ and f_k^*.

$$\hat{\beta}_{ik} = \frac{E[R_{i(k)} f_k(\alpha_k^T \mathbf{x})]}{E[f_k(\alpha_k^T \mathbf{x})]^2} \qquad i = 1, 2, \ldots, q \tag{2.8}$$

$$f_k^*(\alpha_k^T \mathbf{x}_l) = \frac{\sum_{i=1}^q W_i \beta_{ik} R_{li(k)}}{\sum_{i=1}^q W_i \beta_{ik}^2} \tag{2.9}$$

Here the function f_k^* is not a smooth function. Let $(Z_{kl} = \alpha_k^T \mathbf{x}_l)$, and $(Z_{kl}, f_k^*(Z_{kl}))$. Hwang *et al.* [37] selected using a spline smoother to smooth the scatter-plot and reshape the LS estimated activation f_k^*, with the constraints $E[f_k] = 0$, $E[f_k^2] = 1$, and $\sum_{k=1}^p \alpha_{kj}^2 = 1$ for identifiability of the model components.

Since the L_2 loss function is not a quadratic function of α_k, a Taylor series approximation at $\hat{\alpha_k} = \alpha_k + \Delta$ is used to minimize L_2 with respect to α_k with fixed the β_{ik} and f_k,

$$L_2(\alpha_k + \Delta) \simeq L_2(\alpha_k) + (\frac{\partial L_2(\alpha_k)}{\partial \alpha_k})^T \Delta + \frac{1}{2} \Delta^T (\frac{\partial^2 L_2(\alpha_k)}{\partial \alpha_k^2}) \Delta \tag{2.10}$$

The Gauss Newton iteration chooses Δ so that $L_2(\alpha_k + \Delta)$ is minimized, which results in the following linear equation for Δ:

$$\sum_{i=1}^q W_i E \left[(\frac{\partial u_i}{\partial \alpha_k})(\frac{\partial u_i}{\partial \alpha_k})^T \right] \Delta = -\sum_{i=1}^q W_i E \left[(\frac{\partial u_i}{\partial \alpha_k})^T u_i \right] \tag{2.11}$$

where

$$u_i = R_{i(k)} - \beta_{ik} f_k(\alpha_k^T \mathbf{x})$$

$$\frac{\partial u_i}{\partial \alpha_k} = -\beta_{ik} f_k'(\alpha_k^T \mathbf{x})\mathbf{x}$$

is evaluated at the previous update value of projection direction α_k.

Since the matrix on the left-hand-side of (2.11) is nonnegative definite. Δ is a valid descent direction. Solutions of the form Δ/c_j, where $c_j = 2^j$, $j = 0, 1, 2, \ldots$, are tried in (2:10) and the c_j that minimizes $L_2(\alpha_k + \Delta/c_j)$ is used.

The steps for constructing PPLN consist of a forward growing procedure and a backward pruning procedure. During the forward growing procedure, hidden layer neurons are added and optimized one at a time using the training algorithm described in Section (2.2.1). After the parameters associated with the neuron under consideration have been estimated, a *back-fitting* technique is used to update the parameters of previously installed neurons.

Once the network has grown to m^* neurons, a backward pruning procedure is applied to remove over-fitting neurons one at a time. The PPLN again uses the back-fitting procedure to fit models of decreasing size, $\tilde{m} = m^* - 1, m^* - 2, \ldots, m$, to the data, where m^* and m are specified by the user. The most important $\tilde{m}$ out of $(\tilde{m} + 1)$ hidden neurons are kept at each step. The *importance* is measured by

$$I_k = \sum_{j=1}^{q} W_j |\hat{\beta}_{jk}| \qquad k = 1, \ldots, \tilde{m} + 1 \tag{2.12}$$

where $\hat{\beta}_{jk}$ are the estimates for the $(\tilde{m} + 1)$-hidden neuron model.

Network Training Steps

The traditional training algorithm for a PPLN trains the hidden units one at a time, as opposed to all at once as is the case in a backpropagation neural network. The algorithm can be described as follows for the k-th hidden layer neuron:

1. Make initial guesses for α_k, f_k and β_k.
2. Estimate $\hat{\alpha}_k = \alpha_k + \triangle$ using an iterative optimization method
3. Given $\hat{\alpha}_k$, estimate $\hat{f}_k$ as the smooth curve which best fits the scatter-plot $[z_{kl}, f_k^*(z_{kl})]$, where $z_{kl} = \hat{\alpha}_k^T x_l$.
4. Repeat steps 2-3 for several iterations.
5. Use the most recent values of $\hat{f}_k$ and $\hat{\alpha}_k$ to evaluate β_{ik}. (β_{ik} can be computed by setting the derivatives of the loss function L_2 with respect to β_{ik} equal to zero).
6. Repeat steps 2-5 until the loss function is minimized with respect to all β_{ik}, α_k and f_k associated with the k-th neuron.

This procedure is then repeated for the $(k+1)$th hidden layer neuron.

Hermite Polynomial Function

A Hermite polynomial function is used as the activation function for the hidden neuron in PPLN as described by Hwang *et al.* in [36, 37]. The orthonormal Hermite function [28] is shown as:

$$h_r(z) \equiv (r!)^{-1/2}\pi^{1/4}2^{-(r-1)/2}H_r(z)\phi(z) \qquad (-\infty < z < \infty) \qquad (2.13)$$

where r indicates the order of the Hermite polynomials and $H_r(z)$ are the Hermite polynomials constructed in a recursive manner:

$$\begin{aligned} H_0(z) &= 1 \\ H_1(z) &= 2z \\ &\vdots \\ H_r(z) &= 2(zH_{r-1}(z) - (r-1)H_{r-2}(z)) \end{aligned} \qquad (2.14)$$

and ϕ is the weighting function:

$$\phi(z) = \frac{1}{\sqrt{2\pi}}e^{\frac{-z^2}{2}} \qquad (2.15)$$

used to normalize $h_r(z)$.

The scatter-plot $(z_{kl}, f_k^*(z_{kl}))$ for the kth neuron has points located at $z_{kl} = \alpha_k^T \mathbf{x}_l$. This scatter-plot is smoothed by LS fitting of Hermite functions of order R

$$f_k(z_{kl}) = \sum_{r=1}^{R} c_{kr}h_r(z_{kl}) \qquad (2.16)$$

Let

$$\begin{aligned} y_k &= (f_k(z_{k1}), f_k(z_{k2}), \ldots, f_k(z_{kn}))^T \\ h_{kl} &= (h_1(z_{kl}), h_2(z_{kl}), \ldots, h_R(z_{kl}))^T \end{aligned}$$

$$J_k = \begin{pmatrix} h_{k1}^T \\ h_{k2}^T \\ \vdots \\ h_{kn}^T \end{pmatrix}$$

$$c_k = (c_{k1}, c_{k2}, \ldots, c_{kR})^T$$

Then the LS estimate $\hat{c}_k$ of the coefficient vector c_k is obtained by solving:

$$min_{c_k} \parallel y_k - J_k c_k \parallel^2$$

namely

$$\hat{c}_k = (J_k^T J_k)^{-1} J_k^T y_k$$

The resulting scatter-plot smoothing function is:

$$\hat{f}_k(z) = \sum_{r=1}^{R} \hat{c}_{kr} h_r(z)$$

for arbitrary real valued argument z (with fitted values $f_k(z_{kl})$) at the observed neuron input values $z_{kl}, l = 1, 2, \ldots, n$.

And the derivative of Hermite functions has the following special property as described in [28] that:

$$h_r'(z) = (2r)^{\frac{1}{2}} h_{r-1}(z) - z h_r(z)$$

2.3 Experimental Results

A part of the Lenna image (actual size: 480 × 512) (see the marked square in Fig:2.4 to Fig:2.6) is used to train the PPLN network for noise remove, deblurring and mixing purpose.

We select $L_2 = 0.005$ as an accuracy measure. The trained network is tested on the Lenna image and the Peppers image (see Fig:2.2 and Fig:2.3).

2.3.1 PART I : Blurred Image

In our example, Gaussian function

$$g(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{2.17}$$

is used as the kernel to blur the image. The blurred image is:

$$b(m,n) = h(m,n) * f(m,n) \tag{2.18}$$

where $f(m,n)$ is the original Lenna image (Fig:2.2). A 128 × 128 sub-image of $b(m,n)$ (a square image as shown in Fig:2.4, Fig:2.5 and Fig:2.6) is selected as the training input data. A 3 × 3 square pixel-region is used for creating a 9 × 1 input vector from this sub-image. The desired output is the central pixel value of the corresponding 3 × 3 region from $f(m,n)$.

Hermite function of order 9 is used as the activation function. We considered networks with 3, 5, 7 and 9 hidden neurons.

After training, we obtain the following parameters for the network:

1. The input layer weights
2. Hidden layer weights
3. The coefficients of Hermite function

We use family of exponential functions [4] described below to generate the blurred image for testing the performance of the trained network.

$$g(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^m+y^m}{2\sigma^2}} \tag{2.19}$$

Note that blurring kernels have a parameter m. For Gaussian function m is fixed at 2.

We observe that:

1. If σ is kept fixed then blurring decreases with increasing value of m (Compare Fig:2.5 with Fig:2.21).

2. If m is kept fixed then blurring increases with increasing value of σ.

For testing, we use the Lenna image (used for training) and the Peppers image (not used for training). Both images were blurred using and $m = 0.5$ and $m = 3$. We use two networks, one with 3 hidden neurons and the other one with 7 hidden neurons. Our experiment also included other values of m, σ and number of hidden neurons. However, for brevity, we do not include those results here.

These results (see Fig:2.18 to Fig:2.23) show that PPLN trained with Lenna image (blurred with Gaussian function, $\sigma = 4$) produces successful deblurring in test images [5]:

1. That were not used for training.

2. That were blurred using different values of σ

3. That were blurred using different values of m.

We observe the following:

1. Increasing the number of hidden neurons improves the network output quality (Fig:2.19 and Fig:2.20; Fig:2.22 and Fig:2.23). One can also see this from Fig:2.26 and Fig:2.28. But increase in the number of hidden units leads to longer processing time.

2.3.2 PART II : Image with Noise

Noise found in images are usually characterized as additive noise or multiplicative noise.

Additive Noise:

$$N(m,n) = f(m,n) + \eta(m,n)$$

Multiplicative Noise:

$$N(m,n) = f(m,n) \times \eta(m,n)$$

We experimented with images that contain either additive or multiplicative noise.

Let $\eta(m,n)$ denote a noise (normally is a Gaussian noise). It is added to the image, $N(m,n) = f(m,n) + \lambda \times \eta(m,n)$, where λ is an adjustable coefficient factor used to control the amount of noise.

The following is a list of types of noises used to produce the test images.

1. Additive Gaussian noise with mean 0 and variance 1 (before adjusted).
2. A salt and pepper noise.
3. Multiplicative Gaussian noise with mean 0 and variance 1.

Results are shown in figures Fig:2.7 to Fig:2.14

Observations:

1. With fixed number of hidden neurons, the image quality does not improve when the size of the input vector is increased.

2. With a fixed size input vector, increase in the number of hidden neurons does not improve the image quality for the additive noise (see Fig:2.8, Fig:2.9 and Fig:2.15). But it does imporve the image quality for the multiplicative noise (see Fig: 2.11, Fig:2.12 and Fig:2.17).

3. The network trained with additive noise, does not work well on images that contain multiplicative noise (see Fig:2.11 and Fig:2.12 and Fig:2.17) and vice versa.

4. The network trained with additive noise works well on images containing Gaussian (additive) noise or "salt & pepper" noise (see Fig:2.14 and Fig:2.16).

2.3.3 PART III: Blurred Image with Noise

The training data is obtained from Fig:2.6 basic on the equation as shown below. The training image is blurred by Gaussian function with variance 4. Noise is added to the blurred image to produce the test image $o(m, n)$.

$$o(m, n) = f(m, n) * h(m, n) + \eta(m, n) \tag{2.20}$$

Results are shown in Fig:2.26 and Fig:2.27.

We experimented by varying number of hidden neurons and size of input vector. Results show that:

- Increasing input dimension is detrimental to noise removal.

2.3.4 Comparison with Wiener Filter

Wiener Filter

We follow [85] to describe the wiener filter. In general, a degraded signal could be written as a function (2.20) in spatial domain, or in the frequency domain as

$$R(k,l) = H(k,l)F(k,l) + \rho(k,l)$$

where $R(k,l)$ is the Fourier frequency components of the degraded image, $H(k,l)$ is the Fourier frequency components of the image degradation $h(m,n)$, $F(k,l)$ is the Fourier frequency components of the undegraded image $f(m,n)$, and $\rho(k,l)$ is the Fourier frequency components of the noise $\eta(m,n)$.

For the noise free case:

$$F(k,l) = \frac{R(k,l)}{H(k,l)} \tag{2.21}$$

The division of a degraded image's Fourier components by the Fourier frequency components of the image degradation is known as inverse filtering. If $H(k,l)$ is zero at any particular frequency, $R(k,l)$ will also be zero and (2.21) will yield zero over zero, which is undefined. In other words, for those frequencies in which $H(k,l)$ equals zero, the Fourier frequency components from the original undegraded image will be set to zero in the degraded image. Once this occurs, there is no way to restore these components during the restoration process. The best that can be done is to estimate these Fourier components from surrounding Fourier spectral values.

The use of the Wiener filter assumes that in addition to the degradation model, the second order statistics of the noise and the undegraded image are known. The Wiener-Khintchine

theorem allows the PSD (power spectral density) of the noise and the PSD of the undegraded $M \times N$ image to be written in terms of their autocorrelation functions $r_f(m,n)$ and $r_\eta(m,n)$ as

$$S_f(k,l) = \sum_{y=0}^{N-1} \sum_{x=0}^{M-1} r_f(x,y) exp(-j2\pi(\frac{kx}{M} + \frac{ly}{N})) \tag{2.22}$$

$$S_\eta(k,l) = \sum_{y=0}^{N-1} \sum_{x=0}^{M-1} r_\eta(x,y) exp(-j2\pi(\frac{kx}{M} + \frac{ly}{N})) \tag{2.23}$$

where the autocorrelation function of the undegraded image is given by

$$r_f(x,y) = \sum_{n=-\infty}^{\infty} \sum_{m=-\infty}^{\infty} f(m,n) f(m-x, n-y)$$

and the autocorrelation function of the noise is given by

$$r_\eta(x,y) = \sum_{n=-\infty}^{\infty} \sum_{m=-\infty}^{\infty} \eta(m,n) \eta(m-x, n-y)$$

From (2.22) and (2.23), the Wiener filter is defined as:

$$F(k,l) = \frac{H^c(k,l)}{|H(k,l)|^2 + \gamma \frac{S_\eta(k,l)}{S_f(k,l)}} R(k,l) \tag{2.24}$$

where $H^c(k,l)$ is the complex conjugate of $H(k,l)$ and γ is the parameter that selects the type of Wiener filter that is implemented. For $\gamma = 1$, the Wiener filter is referred to as the parametric Wiener filter. Normally, the $S_f(k,l)$ and $S_\eta(k,l)$ are unknown. One can rewrite (2.24) as

$$F(k,l) = \frac{H^c(k,l)}{|H(k,l)|^2 + K} R(k,l) \tag{2.25}$$

where K is a constant, and could be adjusted until the best restored image is obtained. When $K = 0$, the Wiener filter is turned to inverse filter. When K is increasing, the Wiener filter performs less restoration and begins to degrade the already degraded image.

For the case of restoration of a blurred image, a value of $K = 0$ yields the sharpest image but also enhances the noise that is present in the degraded image. As K increases, the Wiener filter removes less of the blurring from the degraded image. Hence, under this condition, the Wiener filter does not enhance as much of the noise present in the degraded image.

The steps involved in implementing Wiener filter are:

- Take the DFT of the degraded image $o(m, n)$ to obtain R(k,l)
- Take the DFT of degradation model $h(m, n)$ to obtain $H(k, l)$.
- Select a value for the parameter K.
- Compute $F(k, l)$ from (2.25)
- Finally take the inverse DFT to obtain the restored image $f(m, n)$.

Result of Wiener filter

We have used Wiener filter to deblur and remove noise. Wiener filter deblurs successfully when K is small and removes noise when K is large. Thus, unlike PPLN, it can not effectively handle a noisy blurred image. Note that the presence of multiplicative noise can not be reduced with Wiener filtering. The results are shown in Fig:2.28 through Fig:2.30.

2.4 Summary

The implementation issue always plays an important role in the performance of any algorithm. However, in this case, it plays a crucial role. This is because of the fact that PPLN

adaptively estimates the activation functions for the hidden nodes. Usually, other learning systems assume some pre-defined form for the activation function. Let us take a look at the impact of the activation function on the network output. From the basic equation it can be seen that the network output is modeled as a different linear combination of the activations $\{f_k\}$. Each activation is taken as a nonlinear "smooth" function of a different linear combination of the input variables. The form of the function is selected by the user. Thus, if the choice is not appropriate for selected problems then satisfactory results will never be achieved. The form of the activation function is parametric. They are flexible in the sense that their functional form can be modified by a set of parameters, they can adapt themselves to different forms under different situations. The following factors are critical when one selects a specific parametric form.

- To have a fast and accurate derivative calculation
- To have a smooth interpolation in computing the activation values.

In our case, we have used orthonormal Hermite polynomials due to (1) their parametric orthonormal property and (2) the easiness of recursive calculation of the functional values. Although, the Hermite polynomials have been previously used in PPLN as well as in PPR, the image processing related problems have never been their target application [28, 36].

In the course of our work, we found that a major problem with PPLN is that the order R , which has to be chosen *a priori*, is sometimes critical for successful approximation. A wrong selection may lead to poor results in training and testing. This is usually explained as a manifestation of the bias-variance dilemma. By using a large R, one is supposedly able to decrease the approximation error, while taking the risk of increasing the estimation error.

Also a PPLN with a fixed R looses the property of universal approximation. One possible way to address this problem is to include a bias term into each linear combination of the predictors [53].

The forward growing and the backward pruning procedures heavily depend on the parameter β_j (the magnitude of the hidden-to-output weight). Our experiments show that this criterion does not always yield better results. The process also lengthens the training time. We believe that the activation function f_j should be included in the decision process.

The updating rule used in most learning networks is error-correcting type. In PPLN the least square error is minimized for updating the 2 weight vectors and the activation function. Note that the design of PPLN was not motivated by biological learning networks rather it originated in the statistical community and was proposed to solve regression problem for very high dimensional data. Our focus, on the other hand, is on problems in which humans are known to perform better than any existing computational system.

Figure 2.2: Original Lenna image

Figure 2.3: Original peppers image

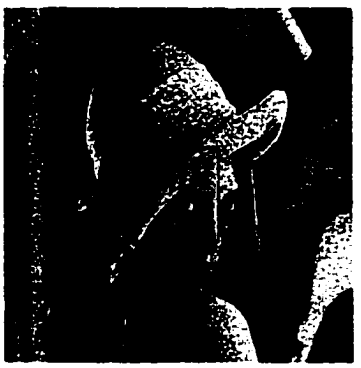


Figure 2.4: Training image: Lenna with 10% Gaussian noise

Figure 2.5: Training image: Lenna blurred by m=2, $\sigma = 4$

Figure 2.6: Training image: Lenna with $m = 2$, $\sigma = 4$, and 10% Gaussian noise

Figure 2.7: Test image: Peppers with 25% Gaussian noise (additive)

Figure 2.8: Result of PPLN-3 on Fig:2.7

Figure 2.9: Result of PPLN-9 on Fig:2.7

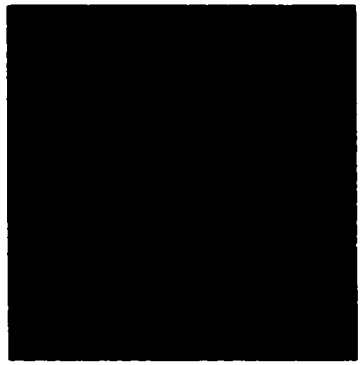


Figure 2.10: **Test Image: Lenna with multiplicative Gaussian noise of *mean* = 0 and *variance* = 1.**

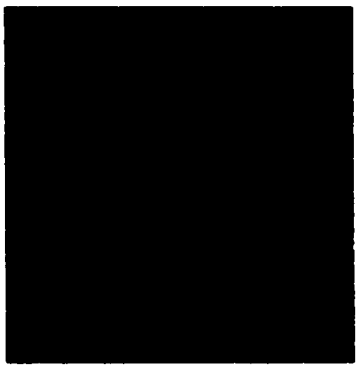


Figure 2.11: Result of PPLN-3 on Fig:2.10

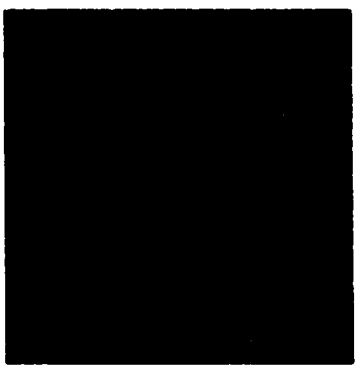


Figure 2.12: Result of PPLN-9 on Fig:2.10

Figure 2.13: Test Image: Lenna with 0.05 "salt & pepper" noise

Figure 2.14: Result of PPLN-3 of Fig:2.13

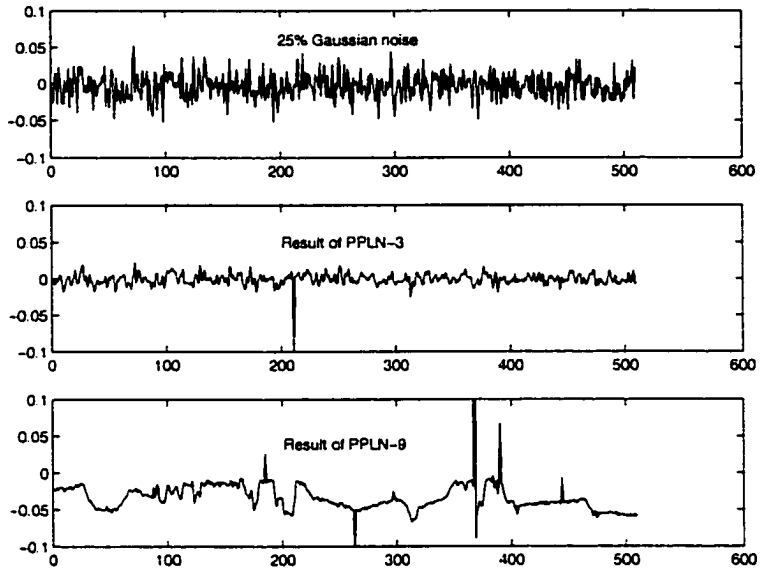


Figure 2.15: Row #300 from Lenna image (Gaussian noise image) and results from PPLN

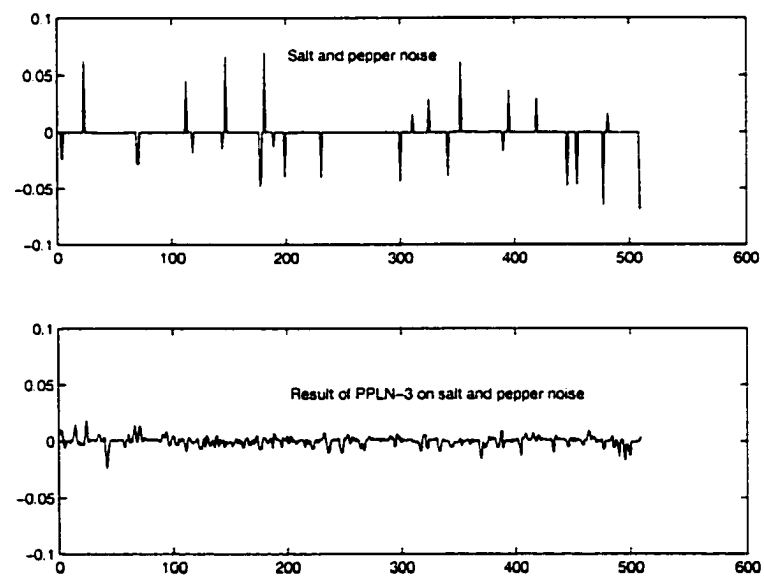


Figure 2.16: Row #300 from Lenna image (with "salt & peppers" noise) and result from PPLN

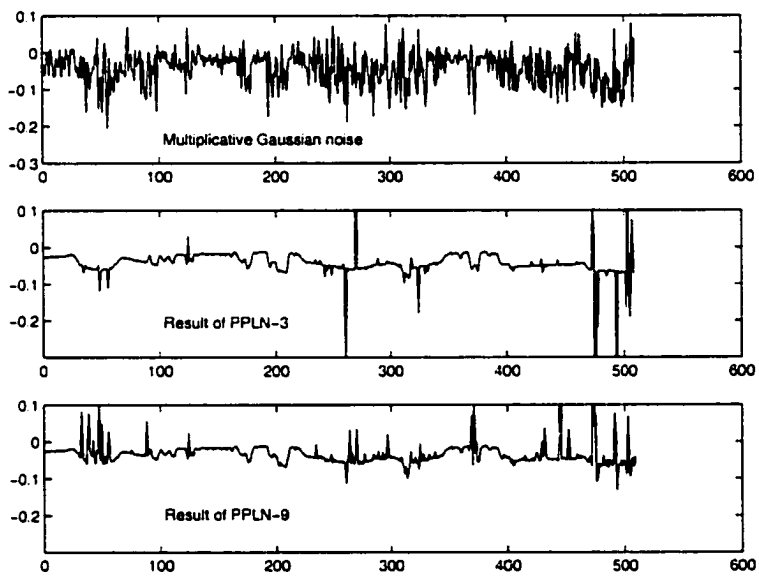


Figure 2.17: Row #300 from Lenna image (with multiplicative noise) and results from PPLN

Figure 2.18: Test Image: Peppers blurred by $m = 3$, $\sigma = 4$

Figure 2.19: Result of PPLN-3 on Fig:2.18

Figure 2.20: Result of PPLN-7 on Fig:2.18

Figure 2.21: Test Image: Lenna blurred by $m = 0.5$ and $\sigma = 4$

Figure 2.22: Result of PPLN-3 on Fig:2.21

Figure 2.23: Result of PPLN-7 on Fig:2.21

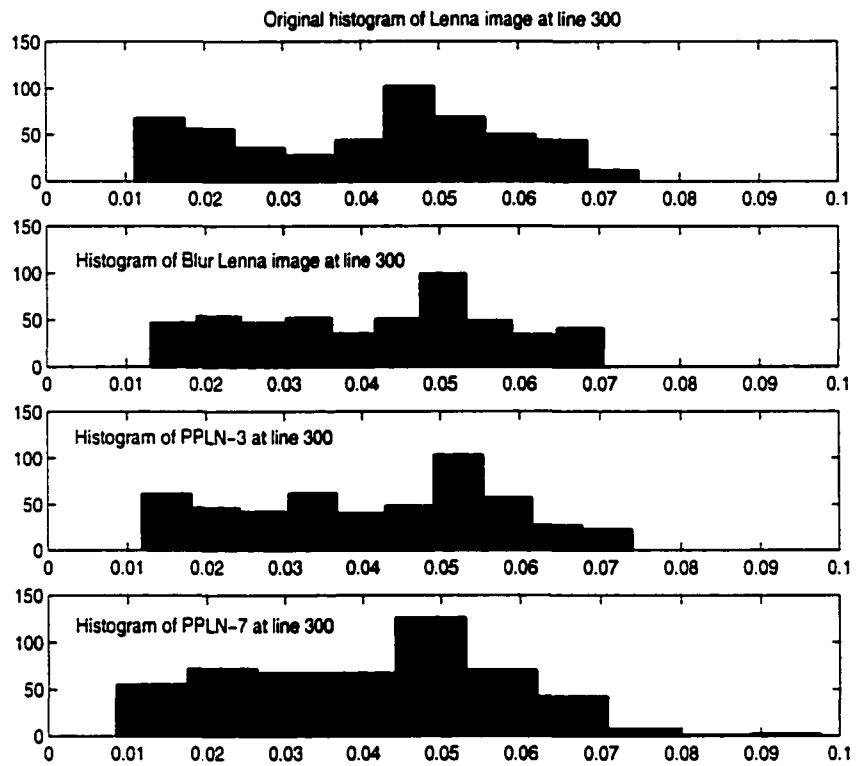


Figure 2.24: Histogram of Lenna image row #300 (blurred with $m = 0.5$ and $\sigma = 4$) and histograms of corresponding results.

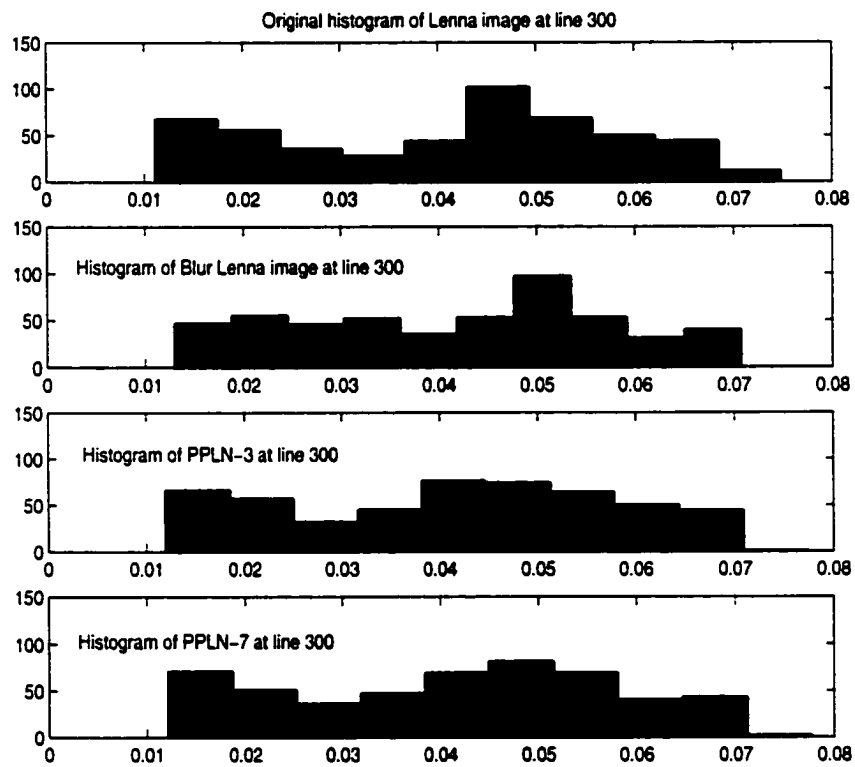


Figure 2.25: Histogram of Lenna image row #300 (blurred with $m = 3$ and $\sigma = 4$) and histograms of corresponding results.

Figure 2.26: Result of PPLN-3 on Fig:2.6

Figure 2.27: Result of PPLN-9 on Fig:2.6

Figure 2.28: Result of Wiener filter on Fig:2.21

Figure 2.29: Result of Wiener filter on Fig:2.2 with 25% Gaussian noise

Figure 2.30: Result of Wiener filter on blurred image ($m = 0.5$, $\sigma = 4$) with 25% Gaussian noise.

Chapter 3

Image Smoothing with Exponential Functions

Any image acquired by optical, eletro-optical or electronic means is likely to be degraded by the sensing environment. The degradation may be in the form of sensor noise, blur due to camera misfocus, relative object-camera motion, random atmospheric turbulence, and so on. Image restoration is concerned with filtering the observed image to minimize the effect of degradation. In this chapter we focus our attention on the process of noise removal. This aspect of image restoration is known as *smoothing*. Smoothing operations are used primarily for diminishing spurious effects that may be present in a digital image. It is a recognized fact that noise must be removed from an image before it is processed further with other operators. For example, consider the edge detection operator. A prominent source of performance degradation in such operators is the presence of noise in the input image.

In the next section we describe the noise model and various types of noise. A brief survey of

noise smoothing techniques is presented in section 3.2. We introduce the family of exponential functions and discuss their suitability with respect to the problem at hand in section 3.3. Experimental results are presented next. Conclusions are discussed in section 3.4.

3.1 Noise Models

A camera uses a lens to form an image that is recorded on a photosensitive film. Images acquired with an electronic camera are typically corrupted with noise due to the camera's sensor and its associated electronics. Photographs of images contain noise due to the finite size of the silver halide grains that are part of the chemical photographic process. Another source of photographic noise is due to the dust that collects on the optics and the negatives during the development process. The point-to-point transmission of video images is another source of noise. For example, video images transmitted via satellites orbiting the earth are susceptible to electro-magnetic interference due to the sunspot activity on the sun. In this section we discuss various types of uncorrelated noise that are commonly encountered in real images [85]. A general model for such system can be expressed as [27]

$$
\begin{aligned}
v(x,y) &= f_2[w(x,y)] + \eta(x,y) \\
w(x,y) &= \int\int_{-\infty}^{\infty} h(x,y;x\prime,y\prime)u(x,y)dx\prime\, dy\prime \\
\eta(x,y) &= f_1[f_2(w(x,y))]\eta_1(x,y) + \eta_2(x,y)
\end{aligned}
$$

The term $u(x,y)$ represents the object (also called the original image), and $v(x,y)$ is the observed image. The image formation system is modeled by the linear system, where $h(.)$ is its impulse response. The functions $f_1(.)$ and $f_2(.)$ are generally nonlinear and represent the characteristics of the image recording mechanism. The term $\eta(x,y)$ represents

the additive noise, which has an image-dependent random component, $f_1[f_2(w)]\eta_1$, and an image-independent random component, η_2. This general noise model is applicable in many situations.

The classification of noise is based upon the shape of the probability density function (or the histogram for the discrete case) of the noise. We describe various types of noise using the histogram.

Uniform noise: Uniform noise produces noise values with equal probability in the range of p and q. The histogram of this noise is given by

$$h(i) = \begin{cases} \frac{1}{q-p}, & \text{for } p \leq i \leq q, \\ 0, & \text{otherwise.} \end{cases}$$

where p, q and i are gray-level values. The mean and the standard deviation are given by

$$m = \frac{p+q}{2} \tag{3.1}$$

and

$$\sigma = \frac{q-p}{\sqrt{12}} \tag{3.2}$$

Gaussian noise: In many situations, the noise that is present in the image can be modeled as the sum of a large number of independent noise sources. Gaussian distribution is used for modeling in these cases. The histogram can be expressed in terms of its mean, m and its variance, σ^2.

$$h(i) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp^{-(i-m)^2/\sigma^2} \quad \text{for} \quad -\infty \leq i \leq \infty$$

Negative exponential noise: This kind of noise usually appears in the image that are acquired using a laser as the illumination source. Often this type of noise is referred to as

laser speckle. The highest probability of occurrence is at zero and drops off exponentially as the gray-levels increase. The histogram is given by

$$h(i) = \frac{1}{a}\exp^{-i/a} \quad \text{for } 0 \leq i \leq \infty$$

where the mean and the standard deviation are both equal to a.

Note that both Gaussian noise and negative exponential noise are defined beyond the dynamic range of a typical digitized image. As a result noise values above the extreme values of the gray-level become saturated. This causes additional pixels within the image to acquire the extreme gray-level values. Care must be taken when comparing the two bins that contain extreme gray-level values.

Salt-and-pepper noise: The following sources give rise to this type of noise.

1. For images captured by an electronic camera, the malfunctioning pixels usually produce pixels with gray-levels of either white or black. This adds black and white pixels to an image, giving the image a salt-and-pepper appearance.

2. The dust and lint that appears on the optics during acquisition of an image causes salt-and-pepper noise.

The histogram is given by

$$h(i) = \begin{cases} p(pepper) & \text{for } i = G_p \\ p(salt) & \text{for } i = G_s \\ 0, & \text{otherwise.} \end{cases}$$

where the probability of the occurrence for the salt-and-pepper noise is $2p$, G_p and G_s are the gray-level values for the pepper noise and the salt noise respectively. In other words, there

will $p\%$ of the pixels at gray-level G_p and $p\%$ of the pixels at gray-level G_s. Salt-and-pepper noise belongs to the family of noise called the *outlier* noise[1].

3.2 Smoothing Techniques

Both spatial and frequency domain smoothing methods are found in the literature [26, 27]. The majority of these methods assume that:

- Noise is random with mean zero and at each pixel it is uncorrelated.
- Noise must be reduced while preserving the important image features (e.g., edges)

These methods use some form of averaging to remove noise. The justification is that in the process of averaging, since the noise satisfies the above assumptions, it will be neutralized. However, methods based on averaging idea also blurs an image. Next, we discuss some of the commonly used techniques for image smoothing [26].

Neighborhood Averaging: This is an example of spatial-domain technique. Given an $N \times N$ image $v(x, y)$, the procedure is to generate a smoothed image $f(x, y)$ whose grey level at every point (x, y) is obtained by averaging the grey level values of the pixels of v contained in a predefined neighborhood of (x, y). The degree of blurring produced by this method is strongly proportional to the size of the neighborhood used. One of the principal

[1]Outlier noise values deviate far beyond the values that are normally expected.

difficulties of this method is that it blurs edges and other sharp details.

Averaging of Multiple Images: Instead of computing the average over a fixed neighborhood in an image, this method forms an image, $\overline{f}(x,y) = 1/M \sum_{i=1}^{M} v_i(x,y)$ by averaging M number of different noisy version of the same image. It follows that the expected value of $\overline{g}$ is given by $E\{\overline{f}(x,y)\} = f_2(w(x,y))$ and the variance of $\overline{f}$ can be expressed in terms of the variance of the noise η, $\sigma_{\overline{f}}^2 = 1/M\sigma_\eta^2$. The expression for the variance indicates that as M increases, the variability of the pixel values decreases.

Median Filtering: In median filtering the grey level value of a pixel (x,y) is replaced by the median of the gray level values in a neighborhood of that pixel, instead of by the average. This method is particularly effective when the noise pattern consists of strong spike-like components, and where the characteristic to be preserved is the edge sharpness. The median m of a set of values is such that half of the values in the set are less than m and half are greater than m. Note that the principal function of median filtering is to force points with distinct intensity value to be more like their neighbors. This eliminates spikes (that appear isolated in the area in the filter mask) and preserves edges.

Lowpass Filtering: Edges in an image contribute heavily to the high-frequency content of its Fourier transform. Thus an alternate approach is to achieve smoothing via the frequency domain by attenuating a specified range of high-frequency components in the transform of the given image. The following equation describes the process

$$S(u,v) = H(u,v)V(u,v) \tag{3.3}$$

The problem is to select a function $H(u, v)$ that yields the smoothed image $S(u, v)$ by attenuating the high frequency components of the observed transformed image $V(u, v)$. Several low-pass filtering approaches are found in the literature [26, 27]. We focus on a specific low-pass filter namely, the Gaussian filter. Gaussian function has the following properties that make it particularly useful in early vision processing.

1. In two dimensions, Gaussian function is rotationally symmetric. This means that the amount of smoothing performed by the filter will be uniform in all direction.

2. The Gaussian filter smoothes by replacing an image pixel with a weighted average of the neighboring pixels such that the weight given to neighbor decreases monotonically with distance from the central pixel. This property helps in preserving neighborhood characteristics.

3. The Fourier transform of a Gaussian function is a Gaussian function. Thus, neighborhood characteristics are preserved in the frequency domain.

4. The variance σ^2 acts as a scale parameter. This aids in multilevel or hierarchical image analysis. Problems arise if Gaussian function is used to smooth images at multiple resolution. Often edges shift from their true location. However, method proposed in [48] can address this problem.

5. Two-dimensional Gaussian filters can be implemented very efficiently because they can be decomposed into two one-dimensional filters.

3.3 The Family of Exponential Density Functions

We consider functions of the following form.

$$g(x) = \frac{1}{K_1}\exp(-K_2|x|^m) \tag{3.4}$$

where constant K_1 is a normalization factor. The exponent m is strictly a positive number. The expression for K_1 involves the exponent m and the constant K_2. The constant K_2 is a function of the variance of $g(x)$.

$$K_1 = (1/K_2^{1/m})(1/m)\Gamma(1/m) \tag{3.5}$$

where the $\Gamma(x)$ is the Gamma function of x and is defined as

$$\Gamma(x) = \int_0^\infty t^{(x-1)}\exp(-t)dt$$

Note that the Gaussian function is a special case of (3.4). The value of the constant K_2 for the Gaussian function is $\frac{1}{2\sigma^2}$. The parameter m is a measure of the non-Gaussian nature of the function. Smaller values of m correspond to more peaked distributions ($m \longrightarrow 0$ yields the δ-function), whereas larger values of m correspond to distribution with flat tops. See Figure 3.1.

We discuss below some of the properties of 2-dimensional exponential function that make it attractive from implementation point-of-view. The Gaussian function is shown as a special case from the derived results for the exponential function.

1. **Symmetric :**

(a) $g(-x,y) = g(x,y)$, y axis symmetry

(b) $g(x,-y) = g(x,y)$, x axis symmetry

(c) $g(-x,-y) = g(x,y)$, symmetry with respect to the origin

In addition to having the above properties, the Gaussian function is also rotationally symmetric.

$$\begin{aligned} g(x,y) &= \frac{1}{2\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}} \\ g(r,\theta) &= \frac{1}{2\sigma^2} e^{-\frac{r^2}{2\sigma^2}} \end{aligned}$$

2. **Separability:**

The separability of exponential function is shown below.

$$\begin{aligned} g(x,y) * f(x,y) &= \sum_{k=1}^{p}\sum_{l=1}^{q} g(k,l) f(x-k, y-l) \\ &= \sum_{k=1}^{p}\sum_{l=1}^{q} e^{-K_2(|k|^m+|l|^m)} f(x-k, y-l) \\ &= \sum_{k=1}^{p} e^{-K_2|k|^m} \sum_{l=1}^{q} e^{-K_2|l|^m} f(x-k, y-l) \end{aligned}$$

Thus convolution of a 2-dimensional exponential function $g(x,y)$ with an image $f(x,y)$ can be performed by two 1-dimensional exponential functions with no restriction on the order.

3. **Cascading Exponential Functions:**

We compute the convolution of a 1-dimensional exponential function with itself. Unlike Gaussian, the result of the convolution is not another exponential function.

$$\begin{aligned} g(x) * g(x) &= \int_{-\infty}^{\infty} e^{-K_2|t|^m} e^{-K_2|x-t|^m} dt \\ &= \int_{-\infty}^{\infty} e^{-K_2(\frac{x}{2}+z)^m} e^{-K_2(\frac{x}{2}-z)^m} dz, \qquad z = t - \frac{x}{2} \end{aligned}$$

Special case $m = 2$:

$$\begin{aligned} g(x) * g(x) &= \int_{-\infty}^{\infty} e^{-K_2(2z^2+\frac{x^2}{2})}dz \\ &= e^{-K_2\frac{x^2}{2}} \int_{-\infty}^{\infty} e^{-2K_2z^2}dz \\ &= \sqrt{\frac{\pi}{2K_2}} e^{-\frac{K_2x^2}{2}} \end{aligned}$$

The product of the convolution of two Gaussian functions with variance σ^2 is a Gaussian function with variance $2\sigma^2$ (scaled by the area under the curve).

4. **Fourier Transform Property:**

The Fourier transform of an exponential function is computed as

$$\begin{aligned} F\{g(x)\} &= \int_{-\infty}^{\infty} g(x)e^{-j\omega x}dx \\ &= \int_{-\infty}^{\infty} e^{-K_2|x|^m} e^{-j\omega x}dx \\ &= \int_{-\infty}^{\infty} e^{-K_2|x|^m}(cos(\omega x) + jsin(\omega x))dx \\ &= \int_{-\infty}^{\infty} e^{-K_2|x|^m} cos(\omega x)dx + j\int_{-\infty}^{\infty} e^{-K_2|x|^m} sin(\omega x)dx \end{aligned}$$

The integrand in the second integral is antisymmetric. Therefore, the second integral must be zero, and the Fourier Transform of the exponential function is

$$F\{g(x)\} = 2\int_0^{\infty} e^{-K_2|x|^m} cos(\omega x)dx \tag{3.6}$$

The Fourier transform of a Gaussian function can be derived from the above expression by substituting 2 for m.

$$\begin{aligned} F\{g(x)\} &= \int_{-\infty}^{\infty} e^{K_2x^2} cos(\omega x)dx \\ &= \frac{\sqrt{\pi}}{\sqrt{K_2}} e^{-\frac{\omega^2}{4K_2}} \end{aligned}$$

A number of factors directly or indirectly influenced our decision to consider the family of parameterized exponential function for image smoothing. We present a brief discussion on a few of them.

- Zhu and Mumford ([89, 90]) suggest that the Gaussian-based filters are not appropriate for edge/feature detection. They focus on (derivative) filters that are used on a data base of diverse natural images. The interesting finding is that natural images possess certain structure irrespective of their origin. The histogram of the first derivative of an individual image shows striking structural resemblance to the histogram of the first derivative averaged over a large number of images. The average histogram, which is quite different from a Gaussian distribution, appears to have high kurtosis and heavier tail. This suggests that a filter that mimics this histogram characteristics is more likely to preserve image structure when applied to an image than a Gaussian filter. The filter that best fits this description is an exponential function with the value of the exponent smaller than one.

- Recently, a mixture of exponential density functions, of the form given in (3.4), is shown to model the probability density of acoustic feature vectors in the space of phonetic units in a more effective manner than a mixture of Gaussian densities [6]. Furthermore, experimental results indicate that the optimal values of the exponent m are smaller than one. This provides support in favor of using exponential functions to represent signal characteristics. Thus, the family of exponential functions ($m < 1$), as compared to Gaussian function, performs better in :

  (1) preserving important signal features when used as a filter

  (2) representing a signal when used as modeling function.

- We conducted a large number of experiments on a set of noisy images to explore the role of the family of exponential functions as a filter. The results are very promising.

3.3.1 Experimental Results

We created a set of filters $filter_{0.1}$, $filter_{0.3}$, $filter_{0.5}$, $filter_{1.0}$, $filter_{2.0}$ and $filter_{3.0}$ for different values of the exponent. For example, $filter_{1.0}$ is a filter designed from the exponential function with m=1.0. Three types of noise are considered for this experiment namely, salt & pepper, Gaussian and random. The data set consists of: (1) real 1-dimensional signals, (2) real images (3) artificial images.

Description of the Experiment: Noise is added to the signal. This noisy signal is then convolved with the filters. We measure the noise variance of the filtered signal. As expected, the value of m and the filter size (when it is implemented) show certain dependence. Thus, to get a clear idea, we experimented with different window and grid sizes on images. The results shown here are for optimal window and grid size that works for all images.

We observe that noise variance characteristics for all real 1-dimensional signals and real and artificial images are almost identical. So, we present one representative plot for 1-dimensional signals and one for images. A rather surprising result is that the artificial and real images produce similar amount of reduction in noise (as measured from the noise-variance graph) for same values of m.

The 1-dimensional signal set contains randomly chosen row from two-dimensional images. An example of such a signal with different noise are shown in Figure 3.2(a). The plots of noise variance for filtered signal are shown in Figure 3.2(b). Note that the noise variance is

at its lowest in [$m = 0.5$, $m = 1.0$] range.

The image data base contains approximately 50 images (real : face, cell, texture images; artificial : star, checker board). Some representative images are shown in Figure 3.3. We show typical noise variance plots for three noise types in Figure 3.4. Note its similarity with Figure 3.2(b). The noise variance is at its lowest in [$m = 0.5$, $m = 1.0$] range. Two images are chosen randomly from the image data base. For visual evaluation we show the noisy and the filtered versions of these two images in Figures 3.5 through 3.7. In all cases we include Gaussian filter output ($m = 2$) for comparison purpose.

Figure 3.8 shows synthetic image with noise and the filtered output. Again, we find optimal noise reduction for values of m in [0.5, 1.0].

3.4 Summary

We study the performance of family of exponential functions in the context of image smoothing. A diverse set of natural and artificial images are used in our experiment. There is strong indication that values of exponent m in [0.5, 1.0] significantly performs better in preserving the images features while reducing noise content.

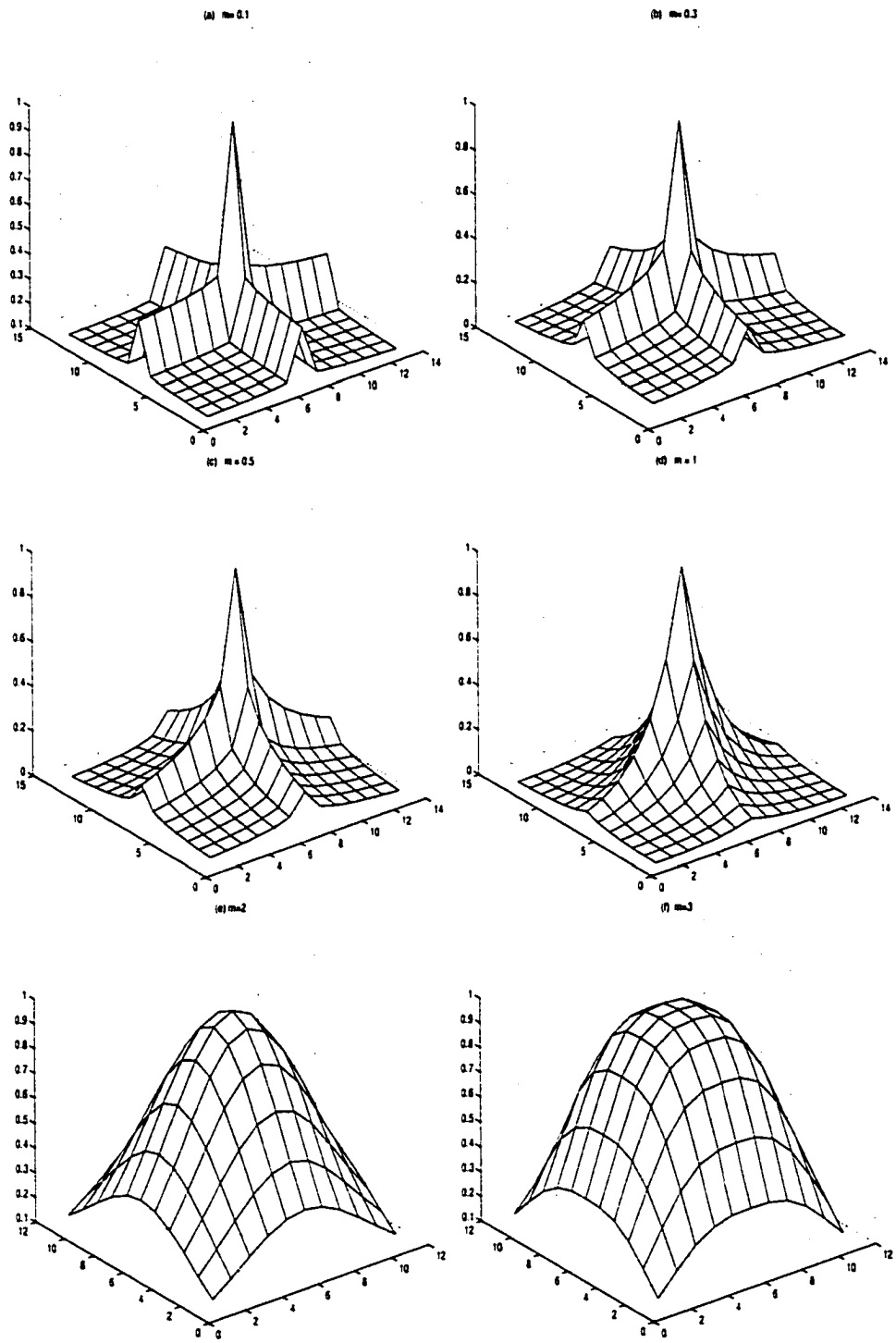
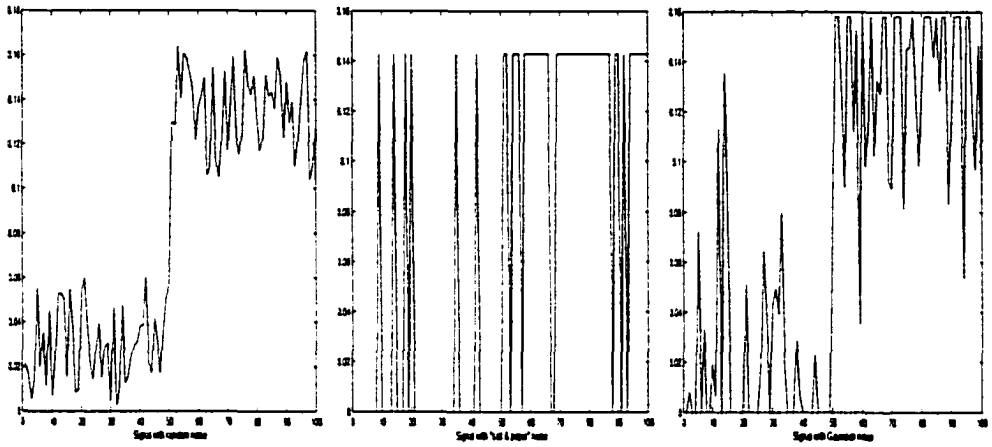
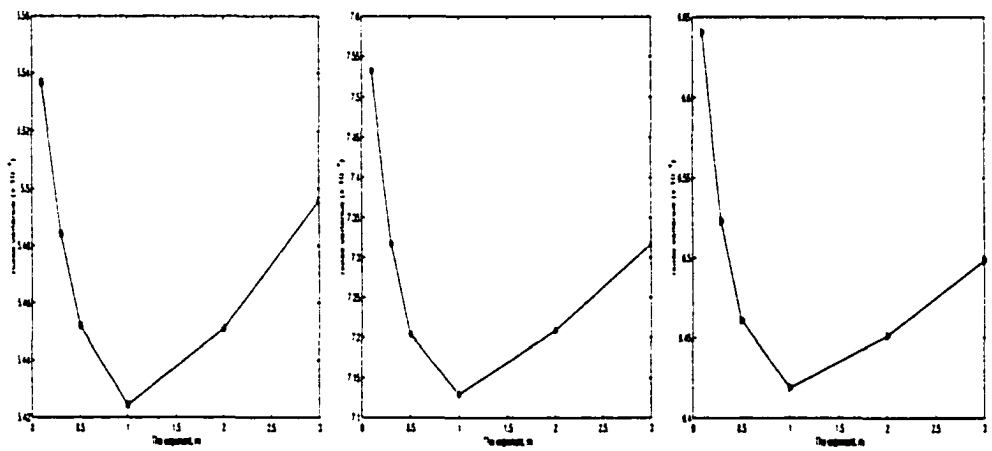


Figure 3.1: Family of exponential functions



(a) 1-dimensional signal with noise



(b) Noise variance for filtered signal

where ∘ for random noise

⋆ for 'salt & pepper' noise

▷ for Gaussian noise

Figure 3.2: Experiment with 1-D signal

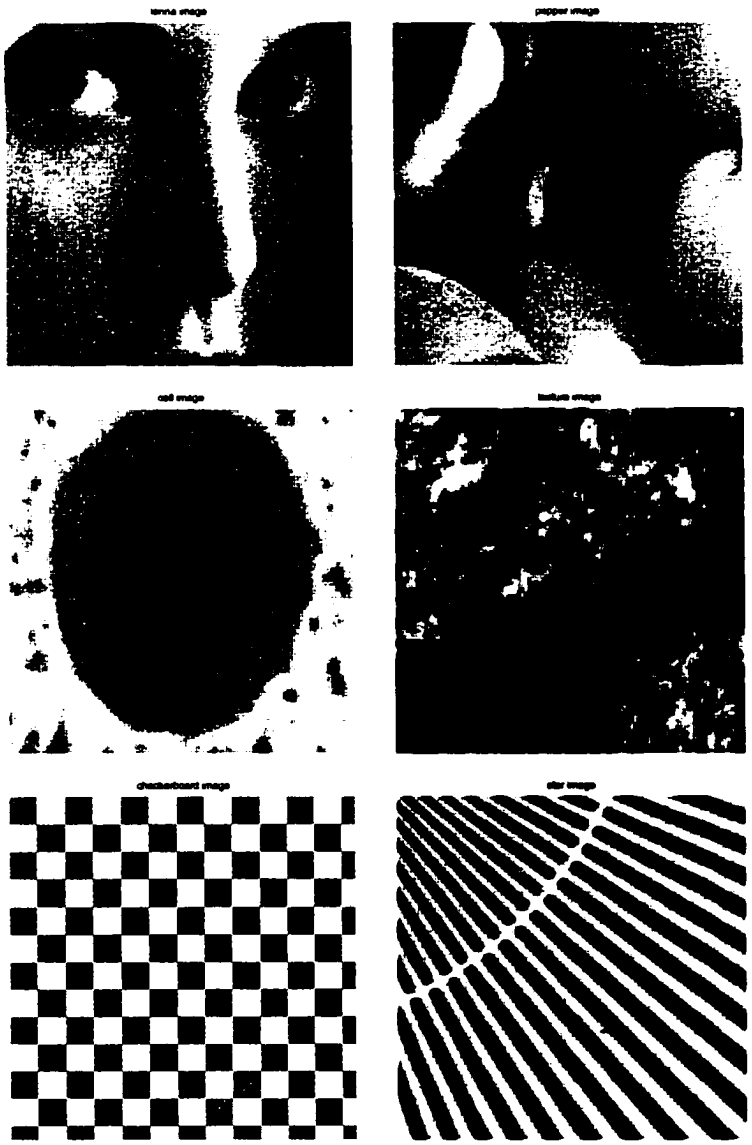
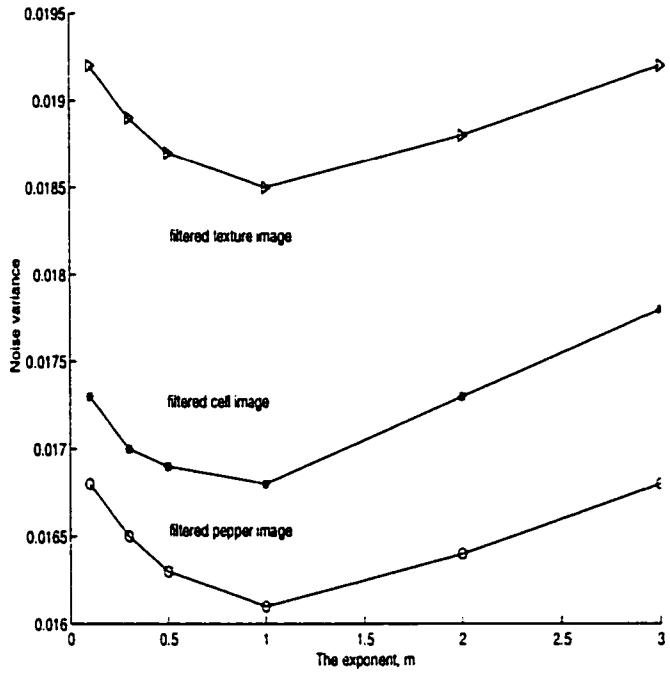
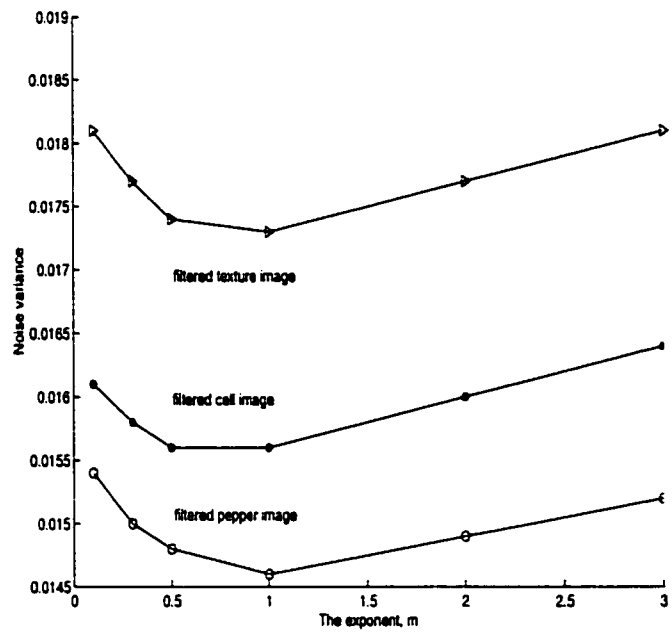


Figure 3.3: Representative images from the data base

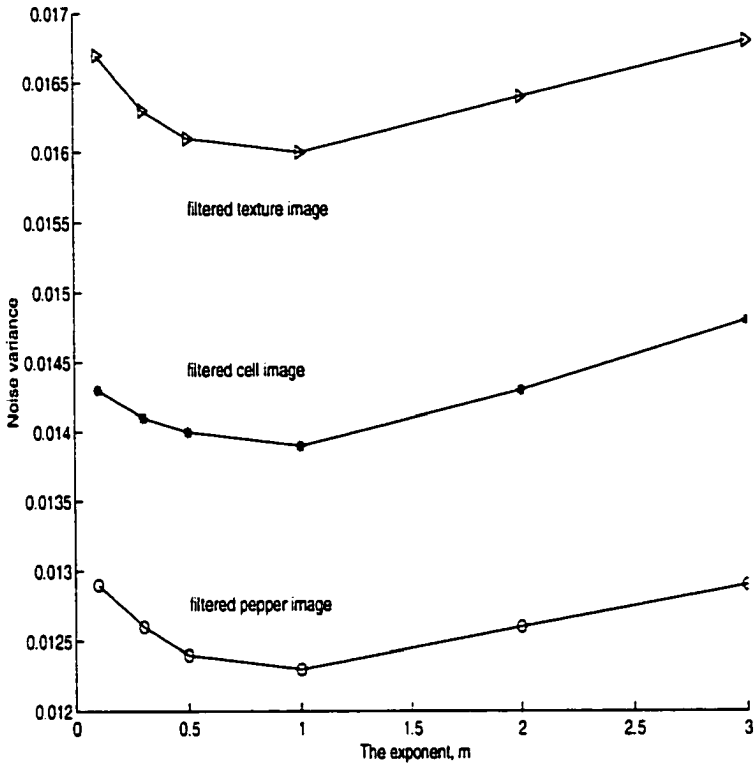


(a)



(b)

Figure 3.4: Role of m in reducing noise



(c)

where (a) random noise

(b) Gaussian noise

(c) 'salt & pepper' noise

Figure 3.4. Contd.

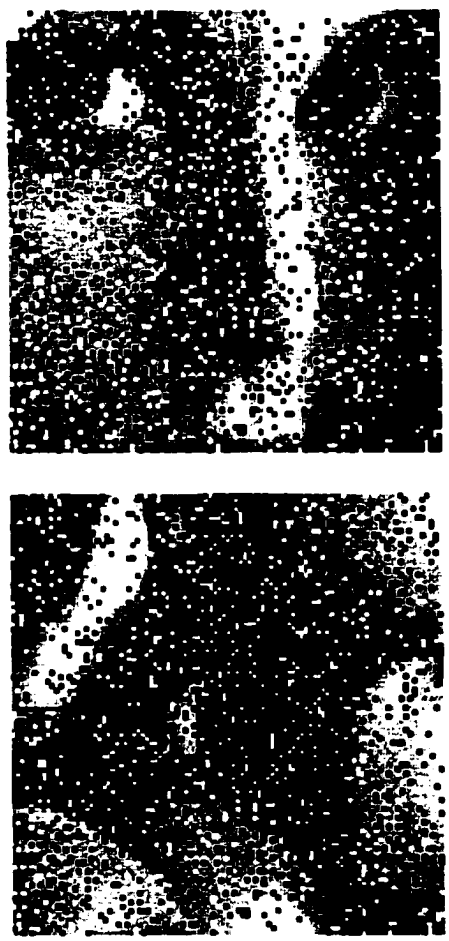


Figure 3.5 (a) Image with "salt & pepper" noise

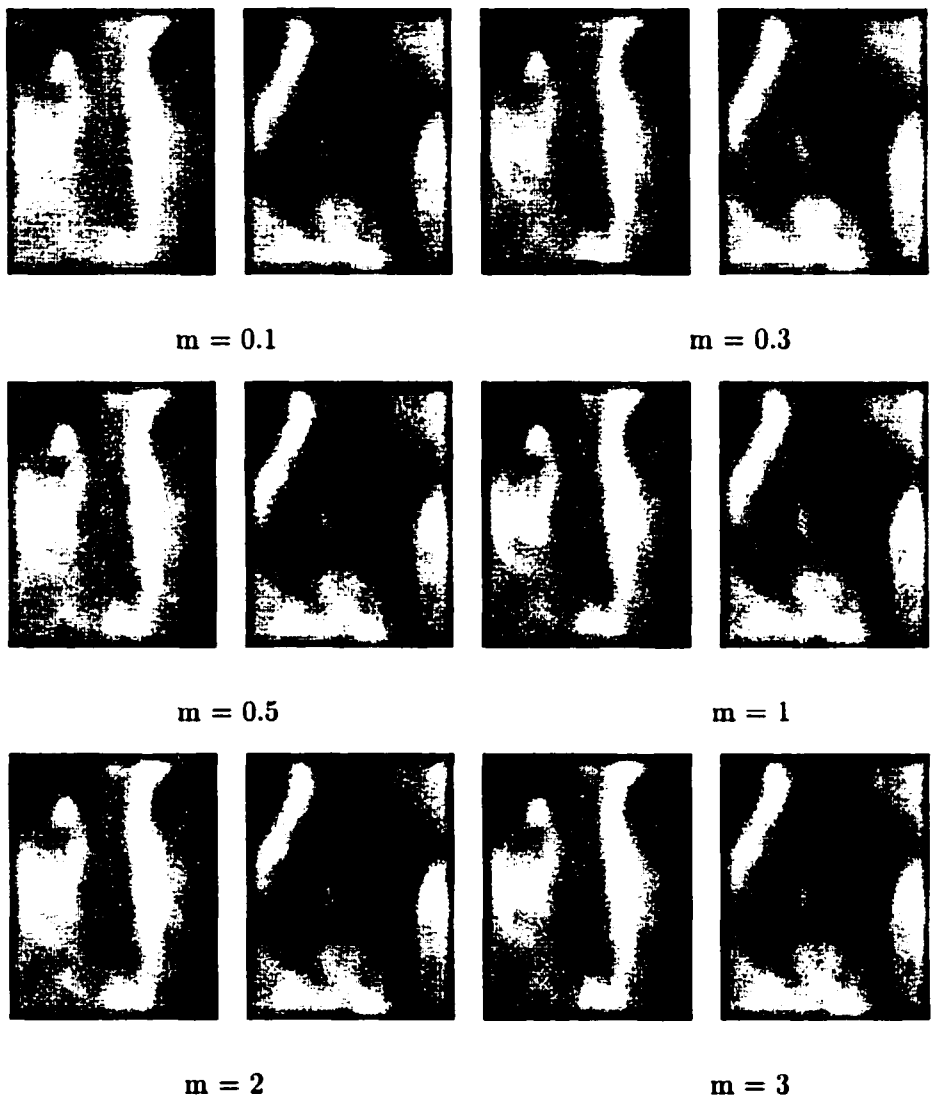


Figure 3.5 (b) Filtered Image

Figure 3.5: Experiment with 'salt & pepper' noise

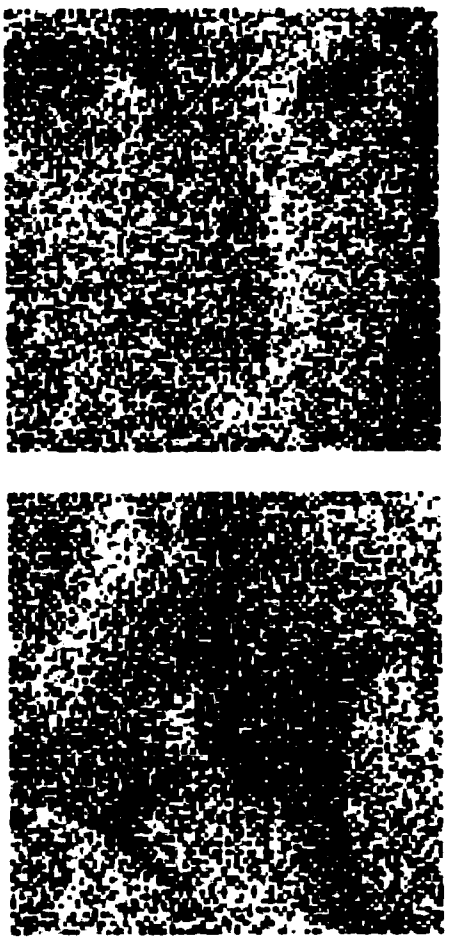


Figure 3.6 (a) Image with Gaussian noise

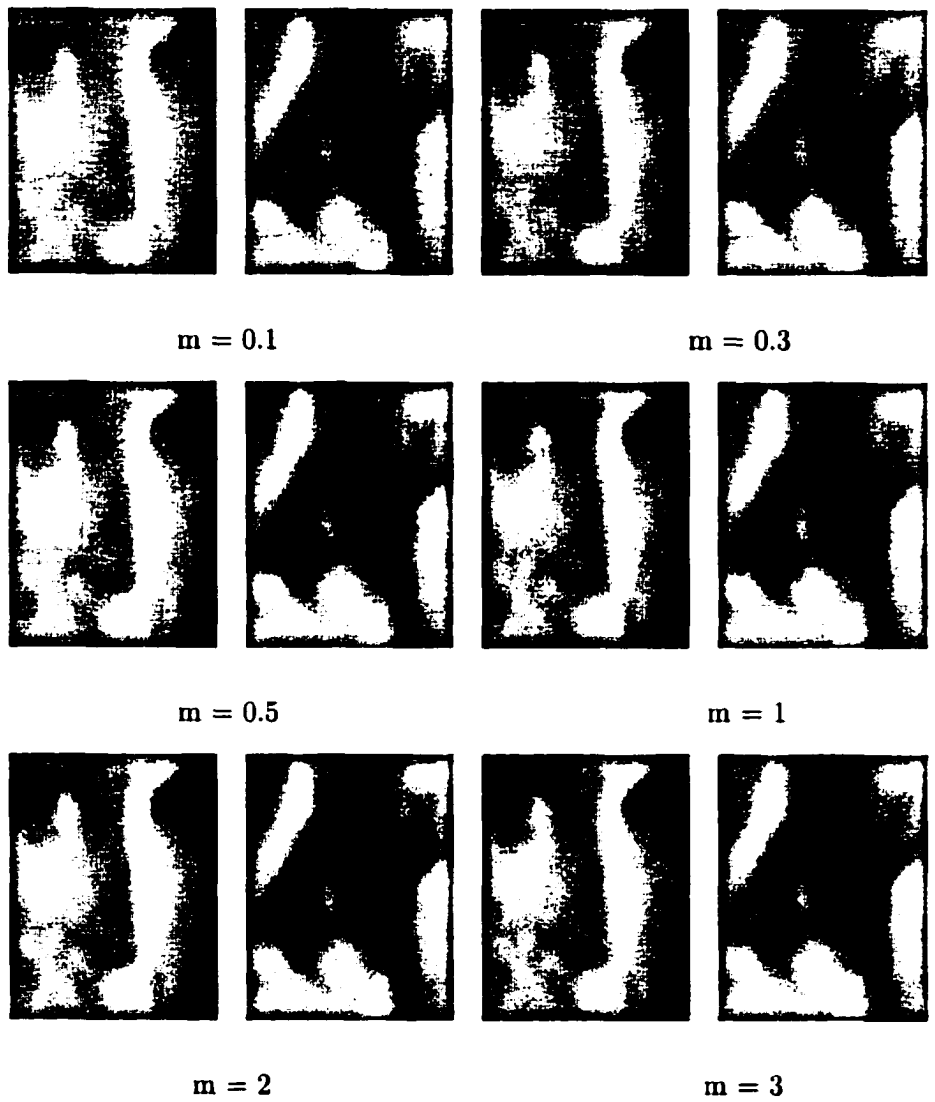


Figure 3.6 (b) Filtered images

Figure 3.6: Experiment with Gaussian noise

Figure 3.7 (a) Image with random noise

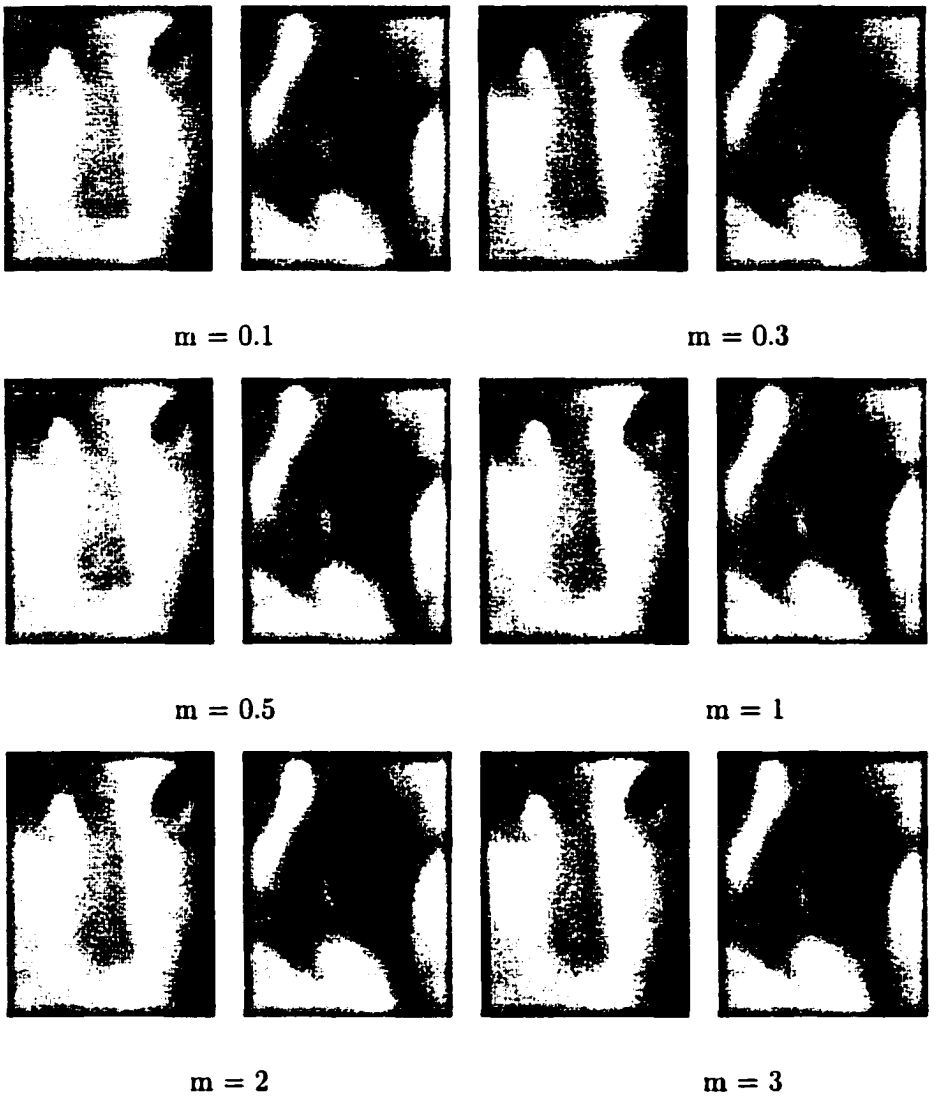


Figure 3.7 (b) Filtered images

Figure 3.7: Experiment with random noise

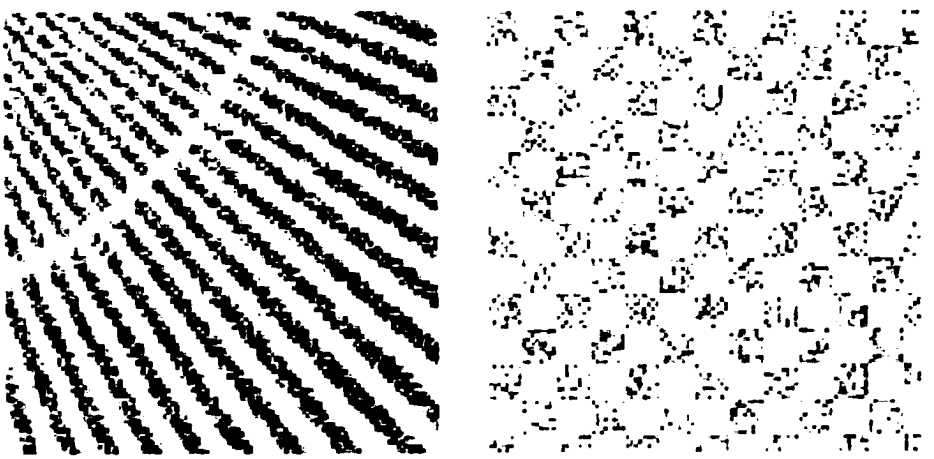


Figure 3.8 (a) Image with random noise

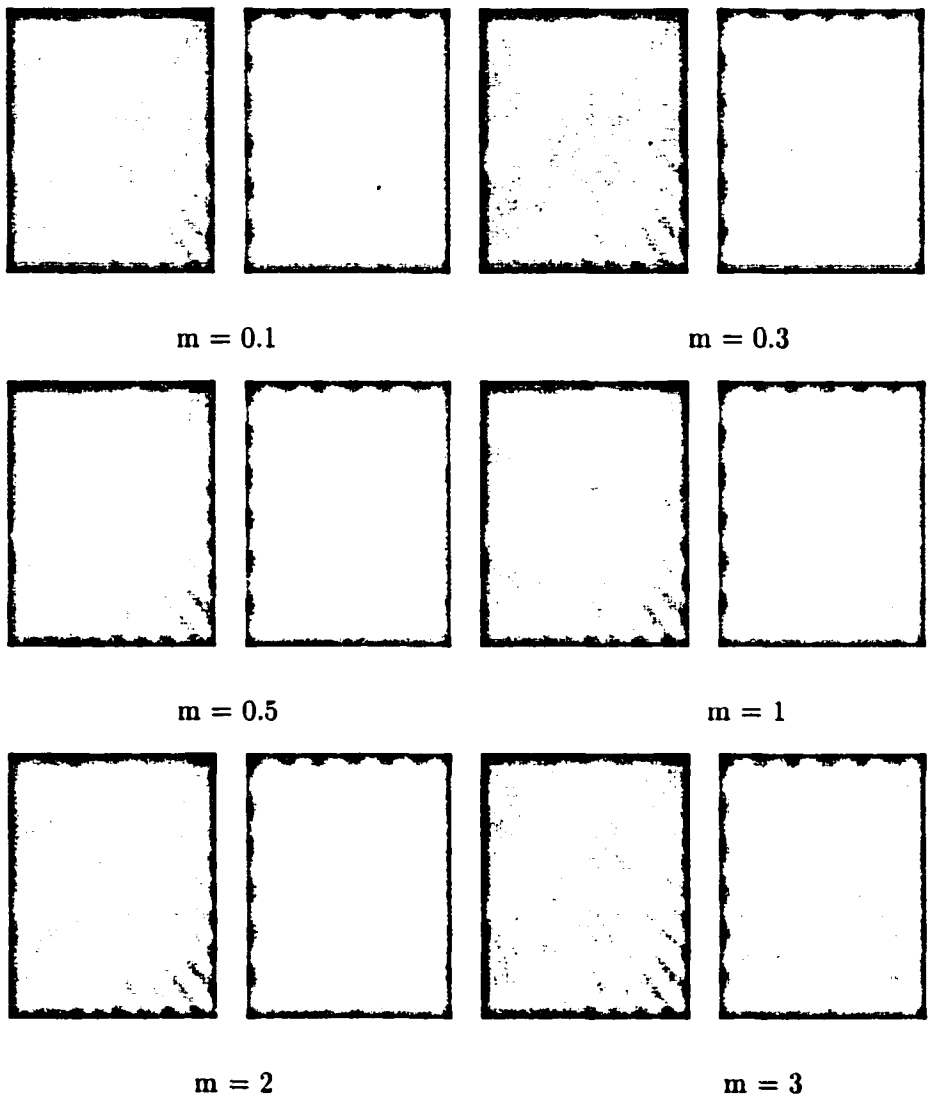


Figure 3.8 (b) Filtered images

Figure 3.8: Experiment with artificial images

Chapter 4

Frequency Property of Three Activation Functions

Mapping of multivariate data onto low-dimensional manifolds for visual inspection is a commonly used technique in data analysis. The discovery of mappings that reveal the salient features of the multidimensional point swarm is often far from trivial. Even when every adequate description of the data requires more variables than can be conveniently perceived (at one time) by humans, it is quite often still useful to map the data into a lower, humanly perceivable dimensionality where the human gift for pattern recognition can be applied.

While the particular dimension-reducing mapping used may sometimes be influenced by the nature of the problem at hand, it usually seems to be dictated by the intuition of the researcher. Potentially useful techniques can be divided into three classes:

- Linear dimension reducers, which are thought of as projections.

- Nonlinear dimension reducers that are defined over the whole high-dimensional space.
- Nonlinear mappings that are only defined for the given points–most of these begin with the mutual interpoint distances as the basic ingredient.

While the nonlinear algorithms have the ability to provide a more faithful representation of the multidimensional point swarm than the linear methods, they can suffer from some serious shortcomings, namely, the resulting mapping is often difficult to interpret; it cannot be summarized by a few parameters; it only exists for the data set used in the analysis, so that additional data cannot be identically mapped.

Classical linear methods include principal components and linear factor analysis. Linear methods have the advantages of straightforward interpretability and computational economy. Linear mappings provide parameters which are independently useful in the understanding of the data, as well as being defined throughout the space, thus allowing the same mapping to be performed on additional data that were not part of the original analysis.

The disadvantage of many classical linear methods is that the only property of the point swarm that is used to determine the mapping is a global one, usually is the swarm's variance along various directions in the multidimensional space. Projection pursuit combine global and local properties of multivariate point swarms to obtain useful linear mappings. Since projection pursuit uses trimmed global measures, it has the additional advantage of robustness against outliers. [20]

When combined with isolation, projection pursuit (PP) has been found to be an effective tool for cluster detection and separation. As projections are found that separate the data into two or more apparent clusters, the data points in each cluster can be isolated. PP algorithm

can then be applied to each cluster separately, finding new projections that may regroup further clustering within each isolated data set. These subclusters can each be isolated and the process can be repeated.

4.1 Hermite Polynomial Function

Hall [28] examines of interestingness based on orthogonal series density estimators. The asymptotic theory for two polynomial-based methods of estimating orientation in projection pursuit density approximation have been proposed. One of the techniques using Legendre polynomials has been described by Friedman [22]. The other employs Hermite functions. The influence of the smoothing parameter (i.e., number of items in the series) is studied. Some interestingness measures that are more robust than others against problem that occur with heavy-tailed densities has been addressed.

A detail description of Hermite Polynomial Function can be found from section 2.2.1.

4.2 Radial Basis Function Networks

The radial basis function (RBF) network is a feedforward structure with a modified activation function in the hidden layer. The activation function is derived from a special class of function known as radial functions. The characteristic feature is that its response decreases monotonically with distance from a central point. The center, the distance scale, and the precise shape of the radial function are the parameters of the model. RBF networks have traditionally been associated with radial function in a single hidden layer network (see Figure

4.1). Here, we focus on single hidden layer networks with Gaussian activation functions (denoted by g_i inside the partially shaded circles in the figure).

Output h_{ij} of the i^{th} hidden unit, when the j^{th} input vector is presented to the network, is given by

$$h_{ij} = \exp^{-||\mathbf{x}_j - \mathbf{t}_i||^2/2\sigma_i^2}$$

Here $\mathbf{t}_i$ denotes the center where the Gaussian activation function corresponding to the i^{th} neuron in the hidden layer is centered. This function's maximum response is concentrated in the neighborhood of the input vectors that are *similar* to $\mathbf{t}_i$, falling off exponentially with the square of the distance. The variance of the Gaussian function, σ_i^2, is another parameter that is used to adjust its width. In our experiment, both parameters i.e., mean and variance are learnable.

The output units are usually linear units; that is, the output of the k^{th} neuron :

$$o_k = \sum_{i=1}^{m} w_{ki} h_{ij}$$

where m denotes the number of neurons in the hidden layer and w_{ki} is the weight associated with the k^{th} output neuron and the i^{th} hidden neuron.

4.3 Sigmoid Function

The network structure of sigmoid function is shown in Fig:4.2.

The only difference between an RBF network and a feedforward neural network with a single hidden layer of sigmoid neurons is the similarity computation performed with the jth hidden neuron. It is easy to see that an RBF network can be obtained from a single-

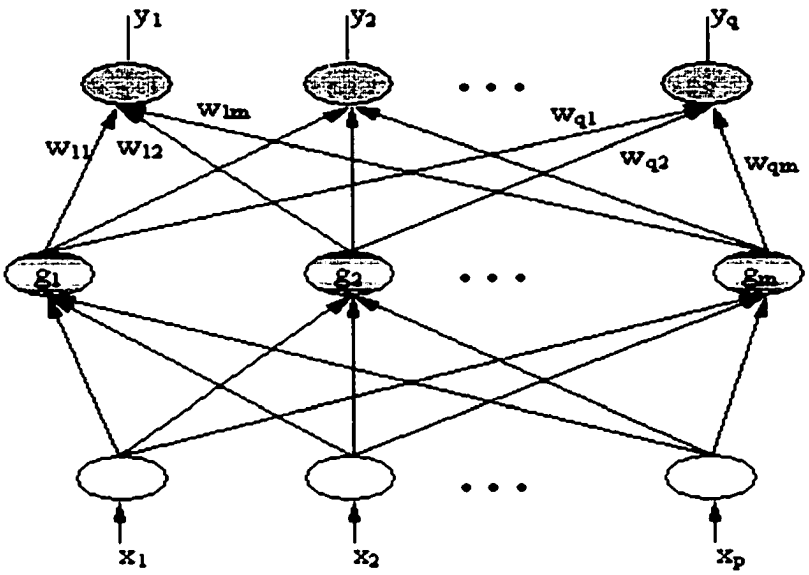


Figure 4.1: Radial Basis Function Network

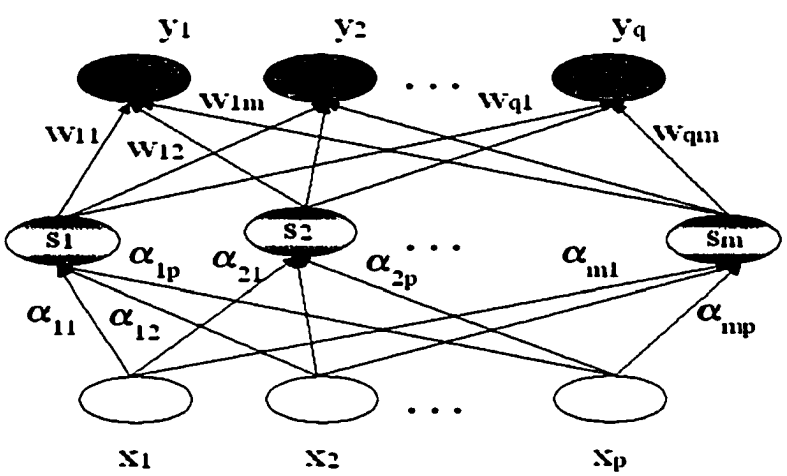


Figure 4.2: Back-propagation Neural Network

hidden-layer neural network with unipolar sigmoid-type neurons and linear output neurons by simply replacing the jth hidden-neuron weighted-sum $net_j = \mathbf{x}^T\mathbf{t}_j$ by the negative of the normalized Euclidean distance $\|\mathbf{x} - \mathbf{t}_j\|^2/\sigma^2$. On the other hand, use of the Gaussian basis function in hidden neurons leads to hidden neurons with Gaussian-type activation functions and with a Euclidean distance similarity computation. In this case, no bias is needed.

In back propagation learning algorithm, the knowledge of the derivative of the activation function associated with that neuron is required. For this derivative to exist, the activation function must be continuous. In general, differentiability is the only requirement that an activation function would have to satisfy. The sigmoid functions have a s-shaped graph. It is defined as a strictly increasing function that exhibits a graceful balance between linear and nonlinear behavior. The logistic function is one of the type of nonlinear sigmoid functions and the commonly used form that satisfy the requirement.

$$s_i = \frac{1}{1 + exp(-v_i)} \tag{4.1}$$

where v_i is the net internal activity level of neuron i, and s_i is the ith hidden neuron output.

The sigmoid functions are not only used in many neural networks, but also used in population models as the logistic function, in spin models in the hyperbolic tangent, in magnetic dipole models in the Langevin function, in special nonlinear approximation theory in piecewise approximators, and in the hysteresis curves in certain nonlinear systems.

4.4 Frequency Domain Analysis

We study the frequency bandwidth response of the three activation functions, Hermite polynomials, RBF and sigmoid functions. It is well known that if the Fourier transform of a signal $f(x)$ is represented as:

$$F\{f(x)\} = F(\omega) \tag{4.2}$$

then for the variable shift in spatial domain, we have

$$F\{f(x+b)\} = e^{j\omega b}F(\omega) \tag{4.3}$$

where

$$|F\{f(x+b)\}| = |F(\omega)| \tag{4.4}$$

The variable shift does not effect the signal magnitude in frequency domain. It also does not change the frequency response bandwidth. It just adjusts the phase response. For the scaled variable, we have

$$F\{f(wx)\} = \frac{1}{|w|}F(\frac{\omega}{w}) \tag{4.5}$$

Here, it effects the frequency bandwidth response.

For Fourier transform of the derivative of $f(x)$, we have:

$$F\{f'(x)\} = jwF(\omega) \tag{4.6}$$

This also changes the frequency response. When $f(x)$ is the Gaussian function, it dose not only shift center of the magnitude, but also changes the phase. Hermite function, which is a

weighted linear combination of different order derivatiives of the Gaussian function, provides ways to control the width & height of each curve. This makes it possible to manipulate the envelop of the Hermite activation function.

The frequency magnitude response of three activation functions are plotted in Fig: 4.3. It is easy to see that sigmoid function has a fixed narrow bandwidth and the Hermite polynomial function has an adjustable bandwidth.

For networks with sigmoidal or Gaussian activation function, each hidden neuron can control the position of the center and the height of the activation function. In case of network with Hermite polynomial as activation function, each hidden neuron in addition to the above paramenters can also control the width & the shape of the activation function. Thus difference is clearly shown in the frequency domain representation of each function. Thus, such a network has more flexibility in selecting the frequency bandwidth.

4.5 Summary

Based on the form of three activation functions in frequency domain, we observe that Hermite polynomial has a structure suitable for adaptation to suit the problem need. As is evident from experimental results, Hermite polynominals when used as activation function produce the best restored images.

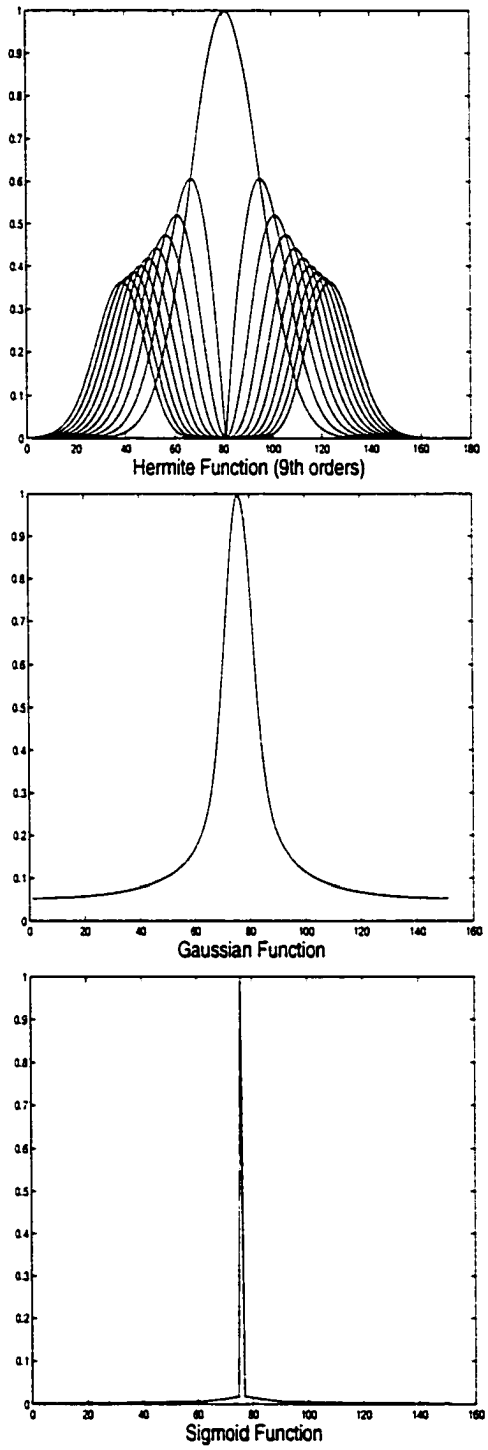


Figure 4.3: Three activation function frequency magnitude

Chapter 5

Committee Machine

> *"The challenge of the next generation of neural networks is not learning by individual weight updating, but the composing of network modules and how to resolve the competition and the cooperation of the different modules".*- Muhlenbein.

It is believed that different parts of the cerebral cortex are designated to perform different tasks. The nature of the task imposes certain structure on the region. Thus, there is a structure-function correspondence in the brain. Furthermore, it has been theorized that different regions compete to perform a task and the task is assigned to the winning region [9, 24, 50]. We may use similar arguments in the context of artificial neural networks to solve real world problems.

The approach used here is based on a commonly used engineering principle: divide and conquer. The basic idea of this principle is nothing but a complex computational task solved by dividing it into a number of computationally simple tasks and then combining the solutions to those tasks. In the learning phase, the training information is divided to a set of

subspaces of input data. And each subspace data is used to train an independent network. The combination of these networks is said to constitute a committee machine.

Two major categories of committee machines based on the gating network performance are described below [62]:

1. *Static Structures:* In this class of committee machines, the responses of several predictors (networks) are combined by means of a mechanism that does not involve the input data, hence the designation "static". The category includes the following method:

   - *Ensemble Averaging:* Where the outputs of the different networks are linearly combined to produce an overall output.
   - *Boosting:* Where a weak learning algorithm is converted into one that achieves arbitrarily high accuracy.

   The ensemble averaging or boosting based committee machines rely on the learning algorithm itself to do the integration. Ensemble averaging improves error performance by:

   - Reducing error due to bias by purposely overfitting the individual experts
   - Reducing error due to variance by using different initial conditions during the training of the individual experts, and then ensemble-averaging respective outputs.

   Boosting improves error performance in following ways:

   - By filtering the distribution of the input data in a manner causing the weak learning models(i.e., experts) to eventually learn the entire distribution [19, 69].

- By resampling the training examples according to a certain probability distribution as in the AdaBoost [16].

2. *Dynamic Structures:* In this second class of committee machines, the input data is directly involved in actuating the mechanism that integrates the outputs of the individual networks into an overall output, hence the designation is "dynamic". Several dynamic structures are:

   - *Mixture of networks:* The individual responses of the networks are nonlinearly combined by means of a single gating network.
   - *Hierarchical Mixture of networks:* The individual responses of the networks are nonlinearly combined by means of several gating networks arranged in a hierarchical fashion.
   - *Gated expert* [1]: Instead of using a probabilistic formulation in the gating network [41], an algorithm called *optimization theory* framework is adopted in the gated network to combine all the outputs of the expert networks to produce the final output. The algorithm also slices up the input space and approximates each of the partitioned regions using an expert network, separately. When compared with the backpropagation algorithm, this method achieves considerable improvements particularly for hard problems.

For computer scientists, modularity is seen as decomposition on a computational level: modules represent problems smaller and more manageable than the original, where the final solution is obtained by combining the results of the individual modules. Connectionists have employed this divide-and-conquer method to model tasks which are suitable for decomposi-

tion into simpler subtasks, in such a way that the individual neural network model specific subtasks, yet these networks are interconnected in some manner.

The mixture of networks and hierarchical mixture of networks may also be viewed as examples of modular networks. A definition of the notion of modularity is given by Osherson et al.[61] as follows: A neural network is said to be modular if the computation performed by the network can be decomposed into two or more modules (subsystems) that operate on distinct inputs without communicating with each other. This factor is handled by the gating network. The gating network is an integrating unit that is not permitted to feed information back to the modules. In particular, the integrating unit decides (1) how an individual output of the modules network should be combined to form the final output of the system, and (2) which modules should learn which training patterns. Improper gating scheme would degrade the network performance.

There are three important issues involved during the learning and retrieval phases of network. For the learning phase, first issue is how to partition the data set in a systematic and reasonable way. Blind partitioning may not be of any help to the learning of the target problem. Second issue is how to initialize the newly added subnets efficiently. For the retrieval phase, the most important function of gating network is to derive the final result from those subnets. So partitioning of the input data not only effects the learning procedure, but also effects the gating network performance.

Figure 5.1 shows a committee network. Note that the size of overall output $\mathbf{Y}_i$ is a vector of the same size as each individual expert network output. Expert networks and gating network share a common input. The overall output $\mathbf{Y}_i$ is generated by the sum of individual expert network output weighted by the gating output. To simplify the presentation, the outputs of

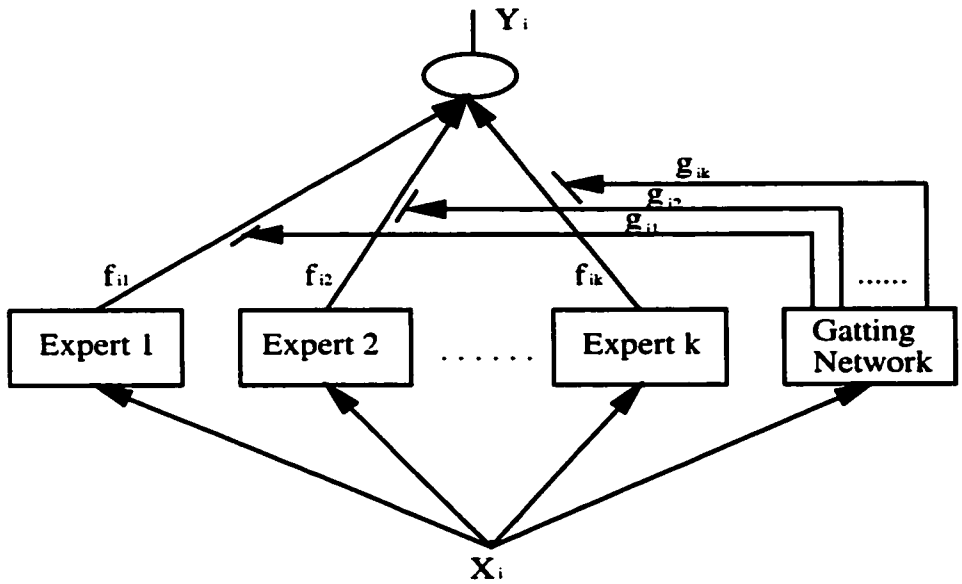


Figure 5.1: Block diagram of a committee machine

the network are assumed to be scalar-valued. Embedding modularity into neural networks leads to many advantages, such as:

- constrained network connectivity
- increased learning capacity
- reduced number of parameters and training data interference
- scalability
- easy combination of different types of neural networks
- reduced risk of overfitting
- localization of computation in the whole system
- suitable for large-scale problems

- simplify the new pattern learning process

The development of a general model of modular neural networks(MNN) will enable a broader use of Neural Networks(NN) and introducing modularity into neural network should be a way to shed more light on its internal behavior.

There are two major phases in a modular network. (1) decomposition of the task, and (2) composition of output from individual committee member. In section 5.1, we will study the decomposition process. The gating function for composition is discussed in section 5.2.

5.1 Clustering of Input Data

5.1.1 A Brief Survey on Data Clustering

We observe that there is currently a paradigm shift from the classical modeling based on first principles to developing models from data. The ability of a system to extract useful knowledge in these data and to act on that knowledge is the key to its success.

Two decomposition methods described by Eric et al. [65] are vertical and horizontal decompositions. When the decomposition concerns the input space, it is called horizontal decomposition. The horizontal decomposition of the input space is usually performed through a spatial clustering. When the decomposition deals directly with the input variable, it is known as vertical decomposition. The vertical decomposition is relevant to the physical and mechanical properties of the system.

The algorithms available for decomposition are summarized below.

1. Local Model Network(LMN) [65]: In LMN, the number of basis functions can be reduced by using a more complex function. Instead of the weighted parameter, a linear function is connected to each basis function. The function is then decomposed into linear segments. The local model is valid so long the relevant part of the function is linear. So, this system enables each basis function to cover larger areas of the input space. Decomposing an input space into regions and then allowing a local model to approximate each of them is a very efficient approach. That is why the LMN can be widely applied to approximate any kind of function. In addition, the use of linear or polynomial local models ensure a learning convergence. Another important feature of a LMN is that its neural representation is clearly interpretable. Since any function can be approximated by polynomials, the use of polynomials instead of linear local models should lead to a significant improvement of LMN.

2. Adaptive Mixture of Experts [41]: In this case, each expert network is an independent single neural network. Each individual expert network performs an independent task. The final result depends on the output of experts and the power of the gating network. It clusters the input space non spatially and completely related to the capacity of approximation of the local models.

3. Reilly et al [64] proposed an algorithm to perform an incremental clustering learning. The input space is decomposed spatially during the learning according to the need of the network. If the neurons in the first layer is not enough to solve the problem then the next one is added and the radius and the prototype of the previous neurons are adapted.

4. Categorizing And Learning Module (CALM) [60]: A CALM module tends to au-

tonomously cluster input data belonging to same region by the way of inhibitory connections between neurons involving competition in the CALM module.

5. Error Correlation Partitioning(ECP) [23]: An error correlation partitioning scheme is proposed here. This scheme will partition the training data to two group during each period of the learning phase based on the number of epoch and the target error accuracy. When the iteration matches the epoch but the error accuracy fails, the learning procedure will separate the training data into two sets, where first set holds the data that matches the error accuracy, and the second set holds the rest of the data. At this time a new subnet is created to train the second set of data. This procedure continues until the entire training data set meet the error accuracy.

6. Vertical decomposition [65]: The vertical decomposition method decomposes the input variables into packs leading each to different subfunctions. This decomposition has to embed at least two interrelated essential properties. First, after integration of the subfunction, the behavior of the function obtained has to fit the original function. Second, each of the subfunctions has to be meaningful according to the physical and mechanical properties of the system. The method based on the training feature separates them to different subsets.

   By using this method, the decomposition can be done before the training each subnet.

7. K-mean algorithm [71]: K-mean algorithm is one of self-organized learning process. It partitions the given set of data vectors into subgroups by placing the centers of the radial-basis functions in only those regions of the input space where significant data are present.

5.1.2 The Cluster Method

Images contain structural informations. Since a blurred image changes its frequency distribution, the objective of any deblurring system is to restore this information. However, we must take into account that information is lost at various scales. For example, although, the edges as well as the flat regions are blurred; they are not blurred at the same rate. This fact must be taken into account. This favors the idea of subdividing the data into appropriate groups. A second reason that directs us to consider grouping of input data is the problem of over-training. In this case, the network may end up memorizing the training data. It may do so by finding a feature(due to noise, for example) that is present in the training data but not true of the underlying function that is to be modeled. When the network is overtrained, it loses the ability to generalize. In the case of deblurring, if a neural network is trained by an entire image that contains different distribution characteristics for data corresponding to different structures in the image; the network may attempt to represent different structures by finding a common ground between the different data distributions. This may result in limiting the recognition ability of the network. Furthermore, attempt to represent input data that are significantly different will cause training to slow down.

Our objective is to divide the input space into a number of sub-spaces, S_n, described by directional unit vectors, $\mathbf{v}_n$, that correspond to some useful information. This creates a certain clustering effect on the input vectors since a vector will lie in the subspace S_n represented by $\mathbf{v}_n$ that is most *similar* to this vector with respect to its information-content. For each such cluster we train a network. In this manner, we create specialized networks - the Committee of Neural Networks. By following this procedure we have eliminated the problem of forcing one network to learn input vectors that are distant from each other. Of course, the choice

of number and directions of vectors remains dependent on the problem at hand. A general procedure is outlined below.

1. Decide the number of sub-spaces N_s
2. Select the sub-space direction vector $\mathbf{v}_n$, $(n = 1, 2, \ldots, N_s)$ that will best represent the sub-space S_n.
3. Normalize the direction vector $\mathbf{v}_n$.
4. Calculate the correlation between each input vector and $\mathbf{v}_n$ to assign it to the correct sub-space.

The Committee of Neural Networks consists of N_s networks. Each network is specially trained to recognize a vector from the corresponding input sub-space. Note that each sub-space direction vector is a function of the distribution of the input values. This is a crucial point, since we are attempting to discover structure in the data.

5.1.3 Training And Testing Data

The data set consists of input vectors of size 9×1 prepared from the 3×3 region of the blurred image. The training set consists of such vectors as input and the center pixel values from the corresponding region of the original image as the output. Let us denote this training set as $\mathbf{T}_{Regular}$. We train a network (henceforth known as Regular Network) [74] with $\mathbf{T}_{Regular}$.

The training data is collected from the blurred Lenna picture. Two different images are used for testing purpose. The summary of training data and testing data sampling frequency are shown in Table 5.1.

Table 5.1: Data sampling

	training data	testing data 1	testing data 2
Sampling Frequency	128x128	254x254	198x135

5.1.4 Experimental Results

Three layered (input layer, hidden layer and output layer) fully connected network is used in our experiment. Networks, trained by non-clustered data, is called Regular network. The expert networks, named as Expert l, where $l = 1, 2, \ldots, N_s$ and N_s is the number of cluster, are trained by clustered data. In both cases, three types of activation functions (Sigmoid, RBF and Hermite polynomial) are used.

We propose a tool to measure the information content of a deblurred image. For a given image, we compute its Fourier Transform and keep the phase part. It is known that the phase part contains more information than the magnitude part. However, the two-dimensional phase plot is difficult to analyze and compare with other similar plots. So, we map the two-dimensional phase data to one-dimensional data in the following manner - sort the data and arrange them from low to high frequency.

Case I: Three Clusters

Consider the following example with $N_s = 3$ and the size of the input vector as 5. We create three directional vectors $\mathbf{v}_h$, $\mathbf{v}_c$, $\mathbf{v}_l$ in the following manner :

- $\mathbf{v}_h$ is obtained by rearranging (any) 5 integers in descending order.

- $\mathbf{v}_c$ is obtained by rearranging (any) 5 integers in a triangular fashion where the highest value occurs in the middle and values on either side are in descending order.

- $\mathbf{v}_l$ is obtained by rearranging (any) 5 integers in ascending order.

Example with a 5×1 vector :

$\mathbf{v}_h = (1/\sqrt{55})[5\ 4\ 3\ 2\ 1]$

$\mathbf{v}_l = (1/\sqrt{55})[1\ 2\ 3\ 4\ 5]$

$\mathbf{v}_c = (1/\sqrt{55})[2\ 4\ 5\ 3\ 1]$

The introduction of N_s directional vectors divides the multi-dimensional input space into N_s distinct sub-regions. Since the input is normalized, the focus is on the direction of the vector. This requires that we train N_s networks. In the training phase, an input vector is compared with all N_s directional vectors for the best match. This selects the network that receives this input vector for training purpose. In the testing phase, the test vector is sent to all N_s networks. This procedure is repeated for each test vector.

We use three directional vectors to characterize three types of edges in the image. Our choice is dictated by the problem of deblurring. Here are the few issues that we consider :

- Edges are important image characteristics.

- Blurring causes loss of edge information from images.

- The process of deblurring may produce a more useful image if enhancement is also achieved along with deblurring.

Here, $N_s = 3$ and the size of the input vector is 9. We create three directional vectors $\mathbf{v}_h$, $\mathbf{v}_c$, $\mathbf{v}_l$ in the following manner :

- $\mathbf{v}_h$ ($\mathbf{v}_l$) is obtained by rearranging (any) 9 integers in descending (ascending) order (a slowly varying edge).
- $\mathbf{v}_c$ is obtained by rearranging (any) 9 integers in a triangular fashion where the highest value occurs in the middle and values on either side are in descending order (a wedge shaped edge).

The introduction of directional vectors divides the 9-dimensional input space into three regions. As opposed to training one network when clustering of input data is not performed, we need to train three networks (net_h, net_l and net_c) in this case. Let us refer to them as Expert 1, Expert 2 and Expert 3 respectively.

Training Phase :

Step 1. Choose an input vector $\mathbf{x}_i$.

Step 2. Compute $t_{ij} = \mathbf{x}_i^T \mathbf{v}_j$ where $j = h, l, c$ to find the best match. Say t_{ic} has the largest value indicating that the input $\mathbf{x}_i$ is most similar to the directional vector $\mathbf{v}_c$.

Step 3. Use $\mathbf{x}_i$ as the input vector and the center pixel value, y_i, of the corresponding window from the original image to train the appropriate network (net_c will be trained when t_{ic} has the largest value).

Step 4. Repeat Steps 1 through 3 until all training vectors have been used.

Testing Phase :

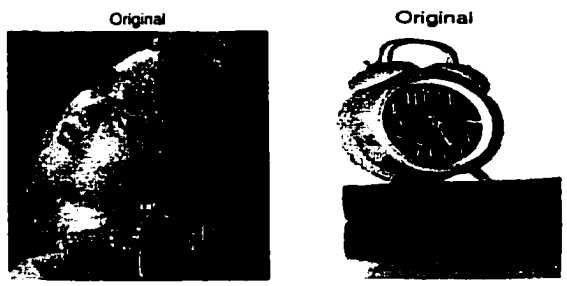


Figure 5.2: Original images

Figure 5.3: Two blurred level images

Test vector $\mathbf{z}_i$ is sent to all three networks. This produces three output values namely, f_h, f_l and f_c which are the outputs for Expert 1, Expert 2 and Expert 3 respectively.

Results :

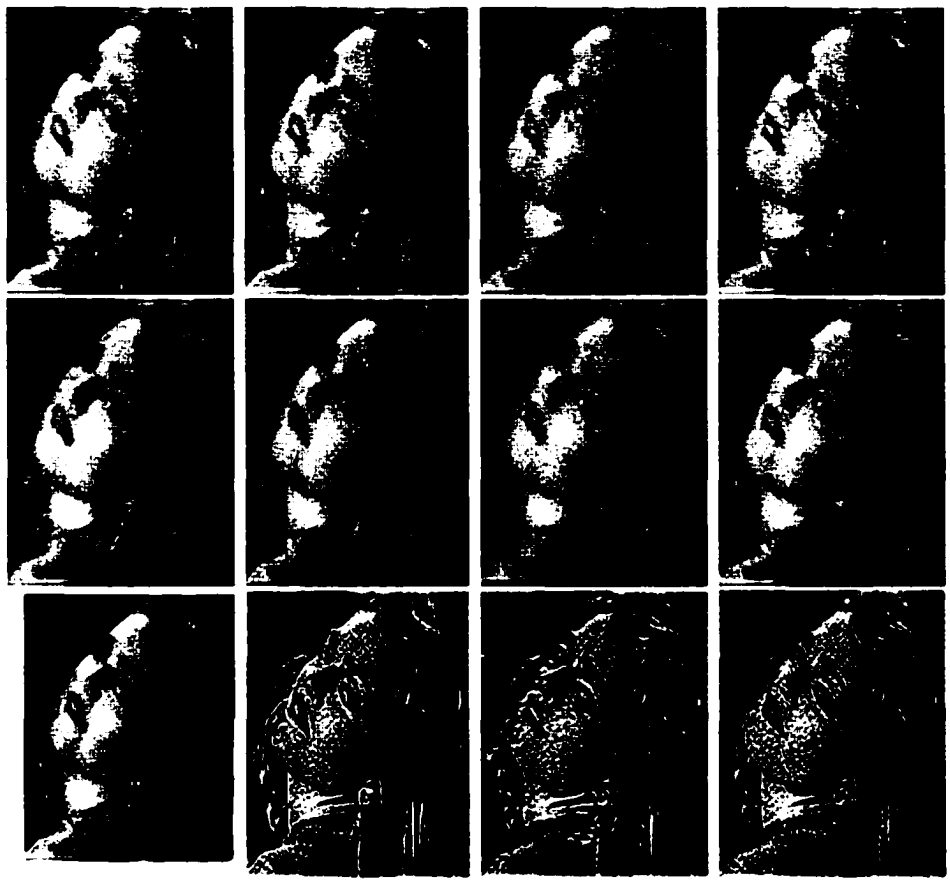
Two sets of blurred images with different levels of blur (see Fig:5.3, we refer to them as image 1 and image 2 respectively) are used to test the network. The original images are shown in Fig:5.2. The outputs of expert network and regular network are shown in Fig:5.4 and Fig:5.5. The corresponing phase plots are shown in Fig:5.6.

Fig:5.6 shows a plot of phase information in one-dimensional space. Three types of activation function outputs are displayed. This plot is in the 0 to π range. The plot in the $-\pi$ to 0

range contains identical information since phase is an odd function. The solid (marked with (1))and the dotted (marked with (2)) curves represent phase data from the original and the blurred images respectively. The phase plot of the output images obtained from the 3 expert networks are shown in the dash line marked with numbers (3) to (5). On the other hand, the phase plot of the output image produced by the network that does not use any input clustering is at the other end of the similarity scale. The results of two images with different degree of blurring (see Fig:5.3) are displayed in Fig:5.8. Among different activation functions, PPLN performs the best. Visual inspection (see Fig:5.4 and Fig:5.5) is in agreement with the phase plot results.

Table 5.2 gives out a statistics of expert network performance from another point of view. This is a process to identify the test vector with one of the input vectors. In other words, a test vector that is similar to one of the directional vectors in terms of its information content is expected to produce the highest output for a certain network. Although, we find evidence contrary to this in some cases (see Table 5.2). It is quite likely that the test vector is equally similar (or dissimilar) to all the test vectors. In that case, none of the network outputs will give us any clear indication.

Table 5.2 has three rows for each image. The last three columns with headings Exp1, Exp2 and Exp3 that refer to the three Experts. Note that each Expert is identified with one of the directional vectors. The first row shows the actual number of input vectors that are most similar to each directional vector and thus should belong to the corresponding group. The next row shows actual number of input vectors that are recognized with minimum error by the Expert correponding to that group. The last row shows the result in a percentage format. One would expect that an Expert designed to capture certain feature performs better



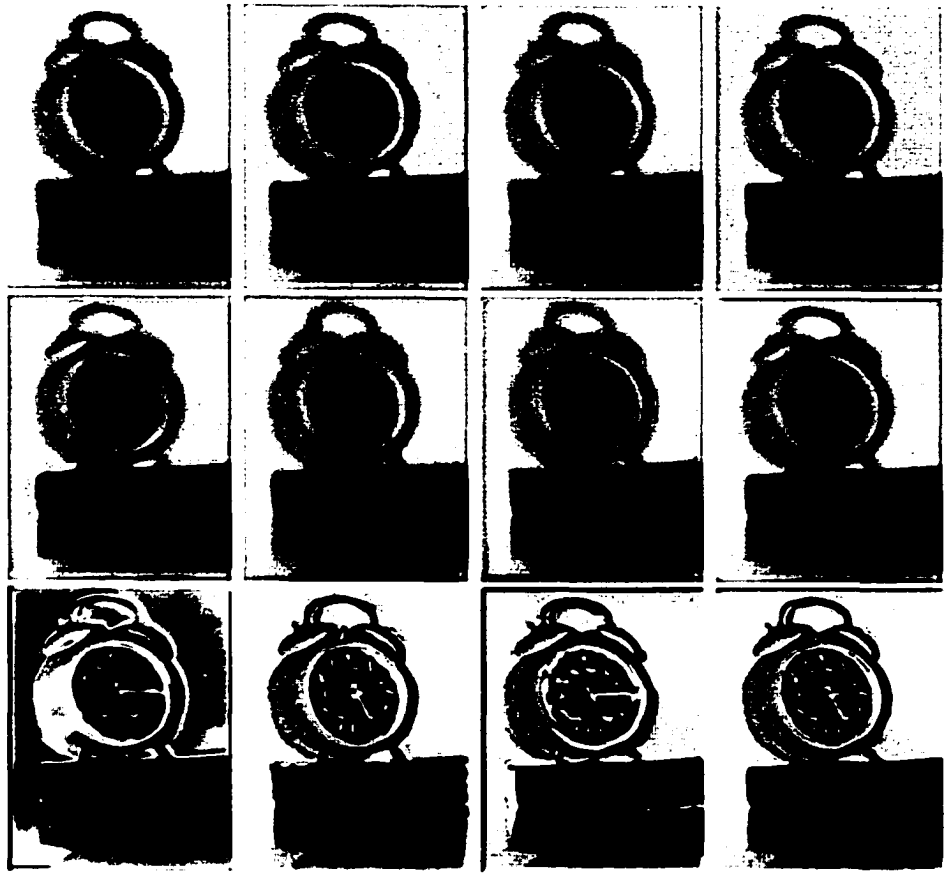
Regular Network Expert 1 Expert 2 Expert 3

row 1: Sigmoid network output

row 2: RBF network output

row 3: PPLN network output

Figure 5.4: Regular and expert network output for image 1



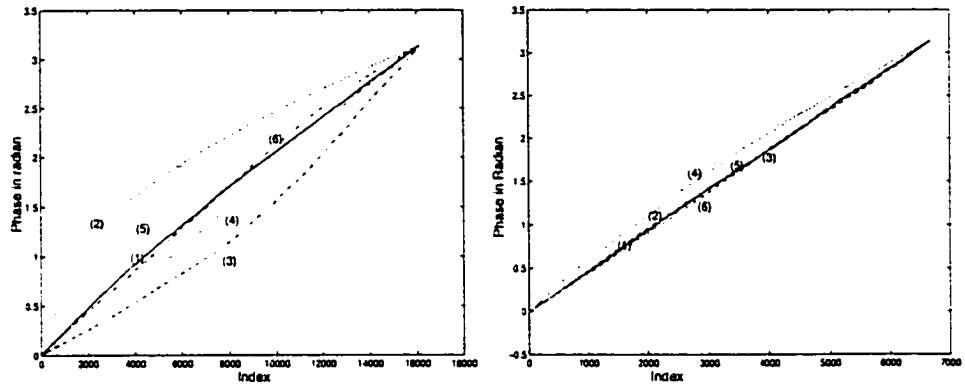
Regular Network Expert 1 Expert 2 Expert 3

row 1: Sigmoid network output

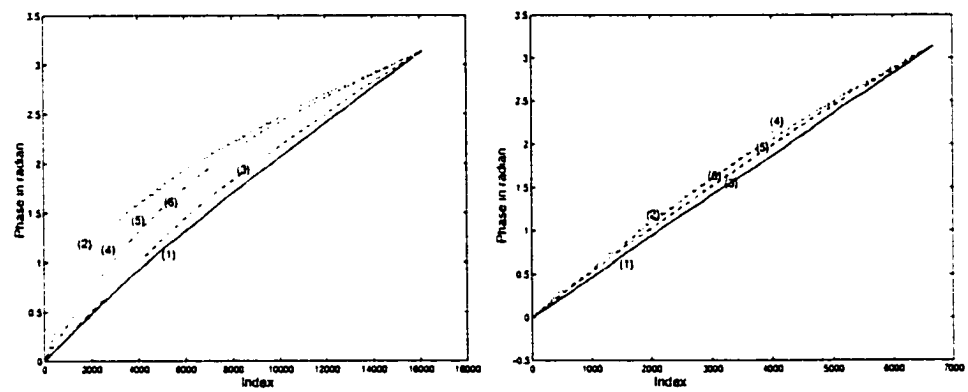
row 2: RBF network output

row 3: PPLN network output

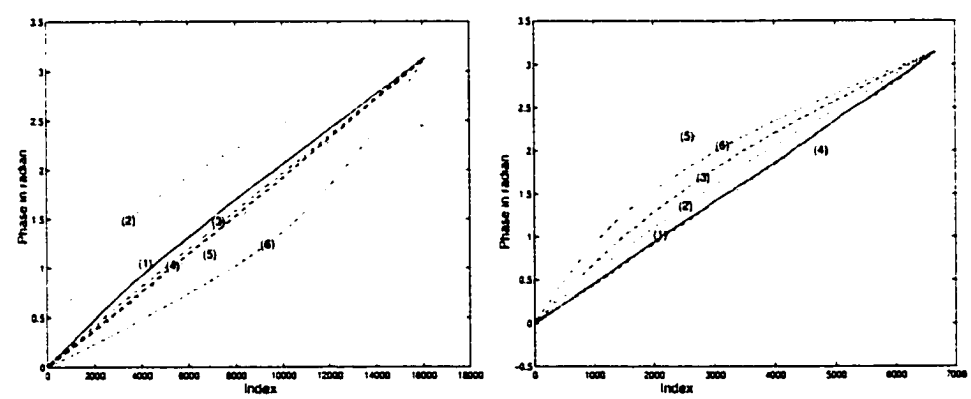
Figure 5.5: Regular and expert network output for image 2



RBF network result



Sigmoid network result



PPLN network result

(1) Original (2) Blurred (3) Expert 1

(4) Expert 2 (5) Expert 3 (6) No cluster

Left column: Madonna image. Right column: clock image

Figure 5.6: Recovered, original and blurred phases for 3-cluster case.

Table 5.2: Pixel statistics

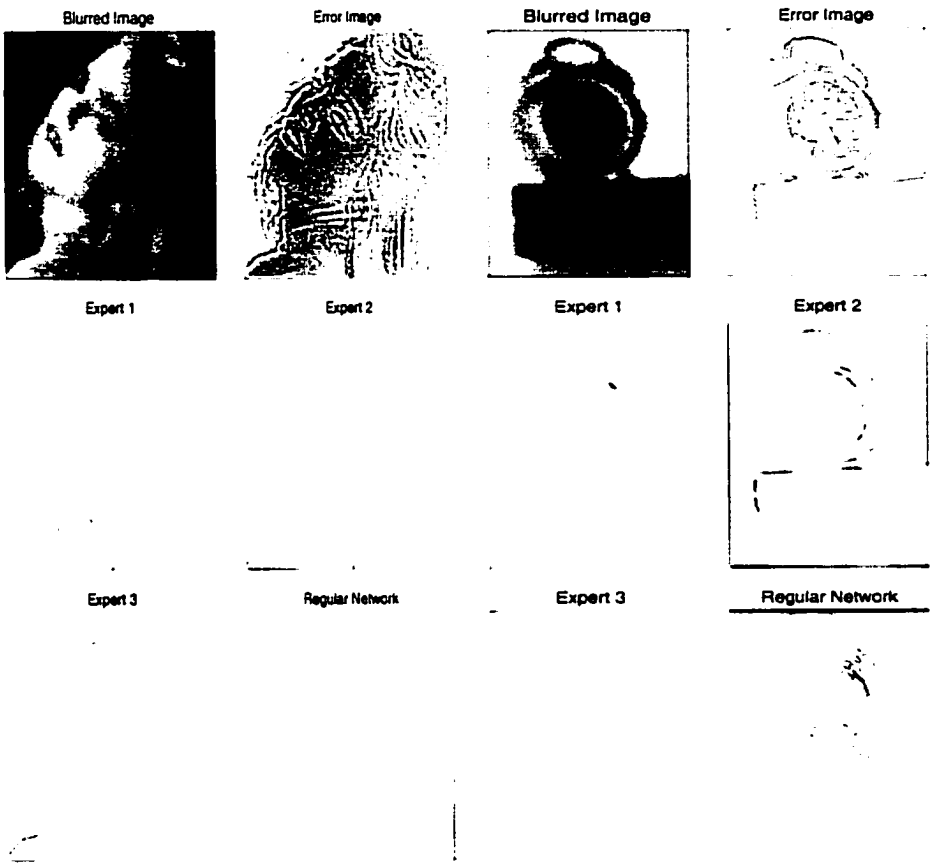
		Exp1	Exp2	Exp3
image 1	# of input vectors in this group	27445	1523	35539
	# of input vectors correctly recognized	11880	543	807
	% of input vectors correctly recognized	43.29	35.65	22.72
image 2	# of input vectors in this group	13701	753	12251
	# of input vectors correctly recognized	7760	168	4455
	% of input vectors correctly recognized	56.64	22.31	36.36

Table 5.3: Average pixel error

	Exp1	Exp2	Exp3	RN
image 1	0.0038	0.0031	0.0045	0.0146
image 2	0.0081	0.0293	0.0080	0.0514

when it encounters input with similar characteristics. Table 5.2 only partially supports this assumption. A possible explanation may be that the size of the input vector is not adequate to capture the neighborhood traits. There are some input vectors, although small, that were not assigned to any of the three groups. This indicates that more directional vectors are needed for this problem.

Table 5.3 reports the average pixel error for the Regular Network (RN) and each of the three Experts. This clearly demonstrates that an image is better restored by a network that is trained for a specific task.



Madonna Image Clock Image

Figure 5.7: Blurred and error images

Here, we show that modular network performs better than the regular network. An error image which is the difference between the original and the blurred for each test image is shown in first row second and fourth column of Fig:5.7. We compute error images (Original Image subtracts Restored Image) for the Regular Network and the three Experts respectively. The next two rows display these error images. Note that the Regular Network which is trained by unclustered data shows most error. On the other hand, the error images from each expert have less error although, it varies from expert to expert.

Table 5.2, Table 5.3 and Fig:5.7 together show that the expert network improves network performance by reducing the training data interference.

Case II: Four Clusters

Instead of clustering the data into 3-cluster, a 4-cluster expert network is used here. The training data are collected from the same source as in Case I. Same blurred level images (see Figure:5.3) are used to test the network performance for the 4-cluster network.

In this case, one more direction vector is added. The classification rule is:

- Same as in Case I.
- v_{cd} is added, which is obtained by rearranging (any) 9 integers in a triangular fashion where the lowest value occurs in the middle and values on either side are in increasing order.

Results :

Fig:5.11 shows the deblurred images produced by the regular and four expert networks from

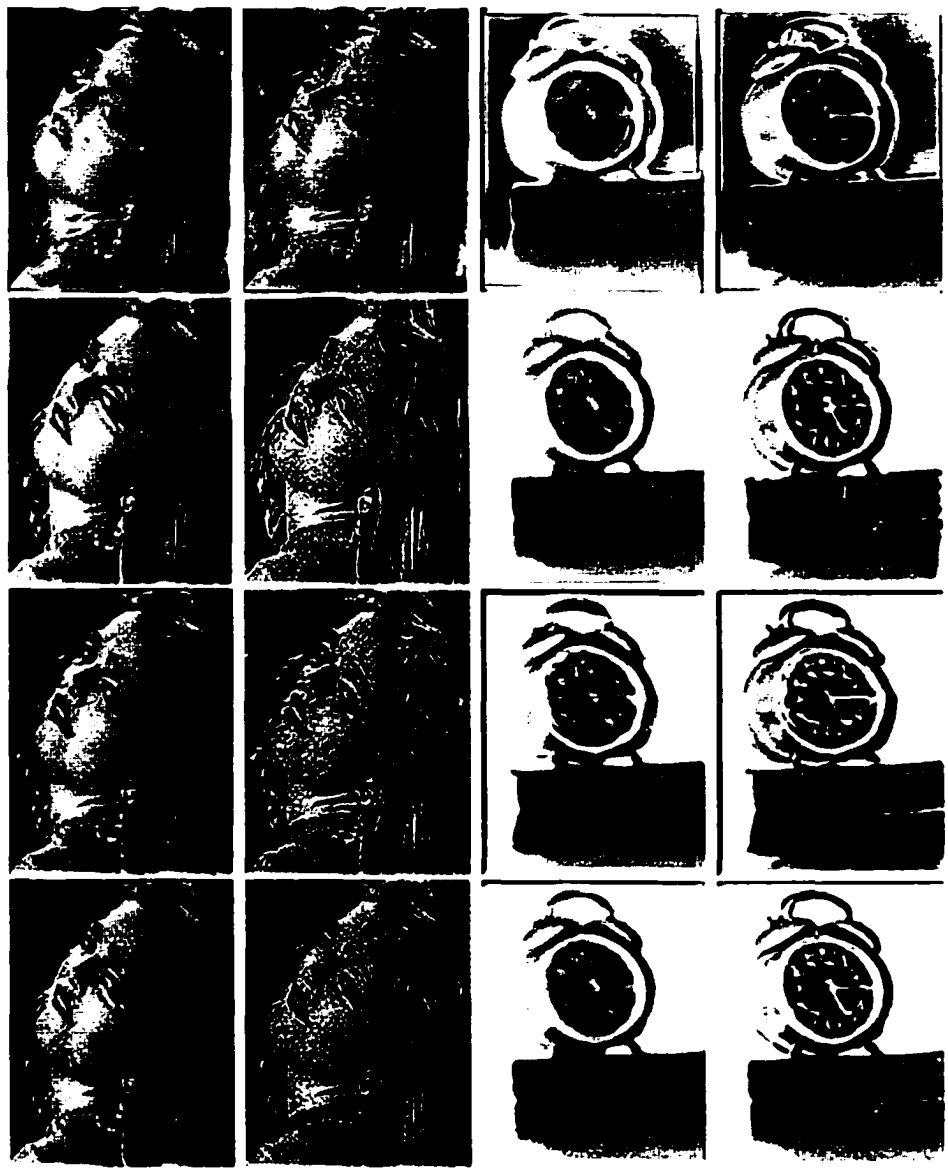


Figure 5.8: Results from Three Cluster Case

the Fig:5.3, respectively. If we see Fig:5.11 as a matrix, then the first row shows the regular network output. The rows 2 to 5 are the results from each individual expert network. For comparison purpose, a row of pixel values selected randomly is plotted in Fig:5.10 (a) and (b) from images in Fig:5.11 columns 1, and 2. It is obvious that Expert 1 has the most similar shape to the original one. Also, all the expert network outputs are more similar to the original one than the regular network output.

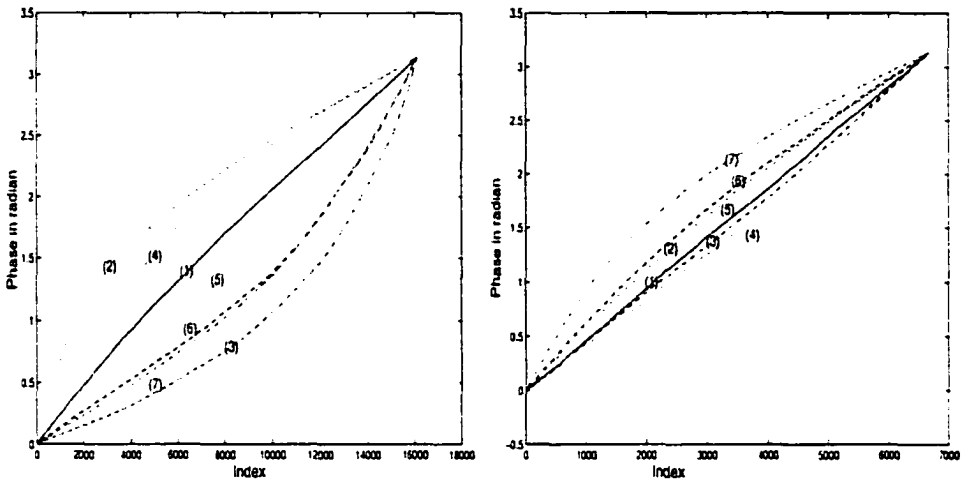
The phase plot for each output images, the original and the blurred image are shown on Fig:5.9. It shows that expert networks perform better than the Regular Network.

The result also shows that the choice of number of clusters effect the network performance. In Case II, the cluster rule 1,2,3 are the same as the rules used in Case I. However, one more cluster organizes the data set in a more effective manner as evident from the result.

Here, the 4-experts performs better than the 3-experts. It again shows that clustering of the training data to reduce interference improves the network performance.

5.1.5 Summary

A *proper* grouping of the data during the training phase has important effect on the network outcome. We observe that a mixture of experts with each expert focused on a specific task delivers an image that clearly shows the restored features. Further work with an integrating unit that decides how the outputs of all experts should be combined to produce the over all output is reported in the following section [74].



Modanno image Clock image

(1) Original

(2) Blurred

(3) Expert 1

(4) Expert 2

(5) Expert 3

(6) Expert 4

(7) No cluster

Figure 5.9: Recovered, original and blurred phases for Four Cluster Case

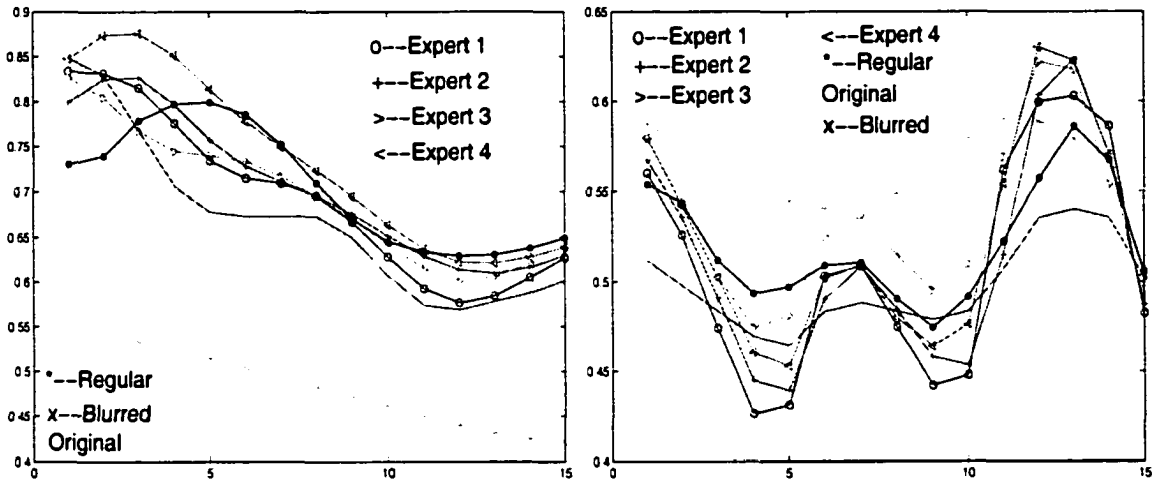


Figure 5.10: A plot of pixel values

5.2 Gating Network

5.2.1 A Brief Survery on Gating Network

A gating network not only decides who is the winning network for every training data in the learning phase, but also organizes the final result. Combination is well-motivated from both an empirical and a theoretical perspective. A combination of structure-function correspondences with the existence of competition between the modules has been speculated by several researchers for the functionally specialized neural modules in the cerebral research area [9, 24, 50]. This theory contains two important phases. The first phase is that there are structure-function correspondences in the brain. The second phase is that brain regions compete for the ability to performs a set of tasks. Different regions learn to perform different functions.

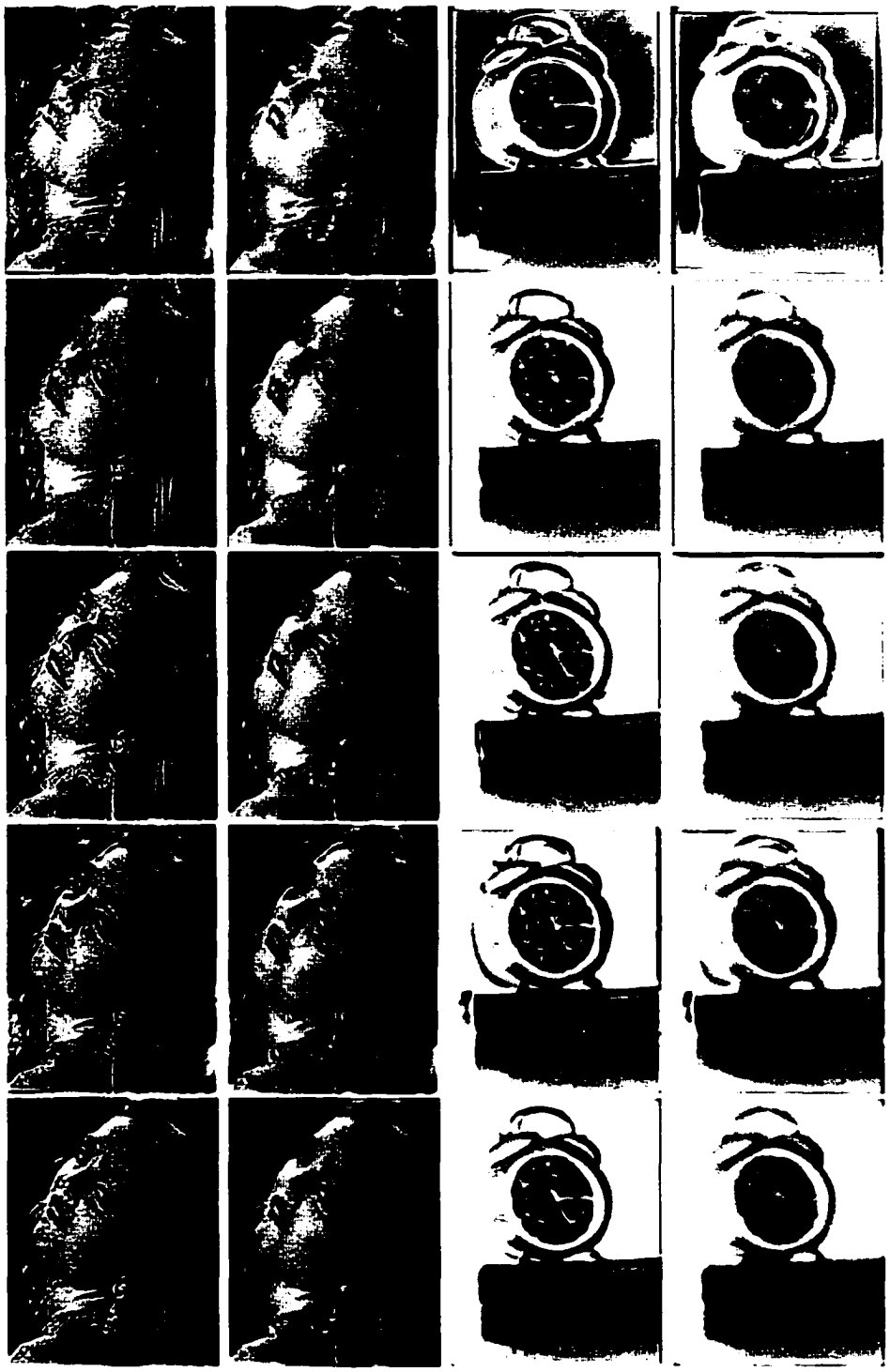


Figure 5.11: Results from Four Cluster Case

Jacobs et. al. [41] proposed that the training data be separated into groups and each group be used to train one network. They use their model in the multispeaker vowel recognition problem and achieve shorter training time and improved recognition accuracy. Later it has been shown that this architecture provides the flexibility of general nonlinear regression while retaining a strong flavor of parametric statistics [45].

Saracevic et al. conducted an extensive user study indicating that query formulations designed by different users for the same information need can retrieve document sets that barely overlap [68].

Combination of evidence has been pursued in other information retrieval(IR) contexts. Examples could be found from ad-hoc search problem [8] and document filtering [33]. In [33], text filtering is approached as a basic text matching problem or an interactive or automatic learning process depending on the degree to which the user is willing to provide relevance judgements about filtered documents. They developed statistical classifiers that produce accurate estimates of the probability of relevance and to seek methods that combine these estimates to improve performance. They found that simple averaging strategies do indeed improve performance, but that direct averaging of probability estimates is not the correct approach. Instead, the probability estimates must be renormalized using logistic regression on the known relevance judgements. A number of complex combination strategies were considered, but showed that they are not as successful as simple averaging strategies. They pointed out that the potential for improvement through method combination is limited because many of classifiers are the strongly correlated, despite the fact that different data representations and optimization strategies are used. An important condition for the success of method combination is training individual classifiers more independently, through the use

of non-overlapping feature sets or partitions of the training data. Their experimental results confirm that if the optimal classifier is unknown, combination can be advantageous even if the combined results are worse than the best individual classifier.

Fu et al. [23] proposed a self-growing modular perceptron network and divide-and-conquer learning schemes for the design of modular neural networks. Instead of modular neural networks(MNN) to specify the number of subnets during the system configuration, they proposed a divide-and-conquer learning (DCL) scheme. The DCL dose not require the designer or the user to specify the number of subnets for solving any given problems. The designer just needs to specify a general architecture for the network. The DCL scheme can automatically create new subnets according to the particular problem's needs during the learning phase. Two learning algorithms: 1) the error correlation based *divide-and conquer learning* (DCL) scheme, 2) the *weight estimation* (WE) method, are used. A two-spirals problem and real-world dataset are used to evaluate and compare the modular perceptron network with several representative neural networks. Their experimental result shows that modular perceptron network achieves better weight learning performance by requiring much less data presentations during the network training phases, better generalization performance, and less processing time during the retrieval phase. On learning the real-world data, the modular perceptron networks shows less overfitting compared to single perceptron network. Besides these, due to its self-growing and fast local learning characteristics, the modular network can easily adapt to on-line and/or incremental learning requirements for a rapidly changing environment.

Divide-and-Conquer algorithm (DCA) has been used in realignment approaches to speed up multiple sequence alignment. Reinert et al. [49] proposed how DCA and the optimal

alignment procedure applied to parts of the sequences can be used to successively improve an initial heuristic alignment, up to optimality. One of major contributions is that the DAC approach is applied to MSA (the implementation of branch-and-bound algorithm) to develop an iterative procedure for computing multiple alignments with a nice time-versus-quality tradeoff. They mentioned that this is the first iterative alignment algorithm that provably converges to the optimal. The protein sequences and four cytochrome p450 sequences are used to test this algorithm.

Yin and Allinson [86] proposed a self-organizing mixture network(SOMN). The SOMN limits the learning in a small neighborhood of the winner. The algorithm requires only scalar and local calculations, and hence is computationally efficient. A winner is chosen according to its kernel output multiplied by its mixing parameter. A learning algorithm is given based on Gaussian mixture or a Gaussian component in a mixture, and Cauchy mixture or a Cauchy component in a mixture. The winner node is selected by the largest response or posterior probability. They summarize three advantages of the SOMN algorithm over the expectation-maximization(EM). They are:

1. The stochastic-gradient-based SOMN algorithm converges much faster than the deterministic gradient-based EM (batch) algorithm.

2. The initial conditions have a greater influence on the convergent results of the EM algorithm. The SOMN, however, is more robust when random initial values are used.

3. The EM algorithm is easily trapped in local optima as it is an exact gradient ascent/descent algorithm. The SOMN, being a stochastic gradient based, can easily except shadow local minima.

Hierarchical Mixtures-of-Experts (HME) Model is described by Peng et al. in [78]. Unlike most of the other divide-and-conquer model, the HME model can summarize the data at multiple scales of resolution due to its use of nested covariate regions and can contain more than one gating network. The other major difference of HME with other models is HME allows regions to overlap so that data items may reside in multiple region. Therefore, the boundaries between the regions are *soft.*

In [82], gating network uses Generalized Regression Neural Networks(GRNN) to forecast the stock index. The modified architecture is suitable for the financial forecasting problem because the importance of each input can change as time changes. Fourteen data attributes of stock market indexes from 10 countries are used to test the model. The result shows that the model trains all input individually using GRNNs and uses a weight-updated gating network to improve the forecast accuracy.

When a neural network operates in a stationary environment, it can be said to have statistical characteristics. The essential statistics of the environment can, in theory, be learned by the network under the supervision of a teacher. In particular, the synaptic weights of the network can be computed by having the network undergo a training session with a set of data that is representative of the environment. Once the training process has been completed, the synaptic weights of the network should capture the underlying statistical structure of the environment, which would justify "freezing" their values thereafter (i.e., an environment whose statistical characteristics do not change with time, or an input data set is large enough to be considered as including all the possible cases).

In algorithms for supervised learning, a set of targets of interest is provided by an external teacher. The targets take the form of a desired input-output mapping, which the network

is required to approximate. In the algorithm for self-organized learning or unsupervised learning, the purpose is to discover significant patterns or features in the input data, and to do so without a teacher. To make the discovery, the algorithm is provided with a set of rules of a local nature. This enables it to learn an input-output mapping with specific desirable properties. The modeling of network structures used for self-organized learning tends to follow neurobiological structures to a much greater extent than for supervised learning.

Each neuron receives information from a limited number of neurons located in an overlying region of the previous layer, which constitutes the receptive field of that neuron. The receptive fields of the network play a crucial role in the synaptic development process because they make it possible for neurons in one layer to respond to spatial correlations of the neuronal activities of the previous layer. Two assumptions of a structural nature are made below.

- The positions of the synaptic connections are fixed for the entire neuronal development process once they have been chosen.
- Each neuron acts with an activation function that has been selected before the process.

The model combines aspects of Hebb-like synaptic modification with cooperative and competitive learning in such a way that the network's outputs optimally discriminate among an ensemble of inputs, with the self-organized learning proceeding on a layer-by-layer basis. That is, the learning process permits the self-organized feature-analyzing properties of each layer to develop fully before proceeding to the next layer.

This architecture has been found to be favorable over single network models with respect to issues such as generalization, speed of learning, interpretable representation, reduction

of temporal and spatial crosstalk [40], scaling and ease of modification of architecture [30], quantification abilities of subitising and counting[7].

5.2.2 Network Structure

In section 5.1, we have shown that network trained by using selected data is more efficient than the network trained by the entire data. Here, we focus on combining outputs from expert networks.

Committee machine consists of several individual expert networks (see Fig:5.1, where each expert is an entire regular network see Fig:2.1). In our experiment, a dynamic structure of gating network is used. The novel method of involving input in the final decision process shows that the role of the input is not over after the training process. Rather, it has a say in matter of preserving its information content in the overall output.

In general, the cluster rule described in section 5.1 is used. The learning process for each expert network follows the same process as a regular neural network [72, 74]. The only difference is that each expert network just learns from the data which it wins. The losing data has not effect on the expert network learning.

5.2.3 Gating Functions

In this section, five methods using *fuzzy* combination of expert network output are described. In traditional committee machine, the gating network function is selected, mostly as softmax function. But the weights which are the synapse of gating network and the input neuron need to be trained. For simplicity, this weight is assigned fixed value and the number of

expert networks is fixed in the following case.

In our experiment, five gating functions are used to compute the contribution of each expert network. They are the following:

1. Normalize gating network output by the sum norm, $g_{ij} = w_{ij} / \sum_{j=1}^{K} w_{ij}$
2. Normalize gating network output by distance (Euclidean norm) as the weight, $g_{ij} = w_{ij} / ||w_{ij}||_j$
3. Use average gating network output as weight, $g_{ij} = w_{ij}/K$.
4. Limit gating network output to a maximum value of 1, that means only winner expert network contribute to the committee machine output, $g_{ij} = w_{ij}/max_j(w_{ij})$.
5. Normalize gating network output by exponential function, $g_{ij} = \frac{exp(w_{ij})}{\sum_{j=1}^{K} exp(w_{ij})}$.

where $i = 1, \ldots, N$ and N is the number of input data and K is the number of expert networks. Computation of w is described below.

For calculating the weight matrix w_{ij}, six major cluster methods that are used in the testing phase are proposed here. These 6 methods could be separated into 3 groups. First group classifies the data based on the phase information. Second group classifies data based on the Euclidean distance. And the last group classifies the data based on the standard deviation. A standardized distance, which is used in method 5 and 6, scales the data so that the result of standard deviation is unity. A standardized distance between vectors $\mathbf{y}$ and $\mathbf{z}$ is defined as:

$$\mathbf{v} = \frac{\mathbf{y} - \mathbf{z}}{\mathbf{s}} \tag{5.1}$$

where $\mathbf{s}$ is standard deviation vector of either data set $\mathbf{y}$ or data set $\mathbf{z}$. This distance associates an important property which is *scale invariant.* If we measure distance in this way, the units we use for the various features will have no effect on the resulting distances, and thus no effect on the final classification.

Six methods to determine weights for gating network are described below.

Method 1 *The inner product of the data with the direction vector* $\mathbf{v}$, *which is* $w_{ij} =< \mathbf{x}_i, \mathbf{v}_j >$

Method 2 *The inner product of the data with the mean vector of the training data, which is* $w_{ik} =< \mathbf{x}_i, \mathbf{m}_k >$

Method 3 *The Euclidean distance between the data and the direction vector, which is* $w_{ij} = |\mathbf{x}_i - \mathbf{r}_j|$

Method 4 *The Euclidean distance between the data and the mean vector of the training data, which is* $w_{ik} = |\mathbf{x}_i - \mathbf{m}_k|$

Method 5 *The standardized distance between the data and the mean vector of the training data, which is* $w_{ij} = |\frac{\mathbf{x}_i - \mathbf{m}_k}{\mathbf{sd}_k}|$

Method 6 *The standardized distance between the data and the mean vector of the training data, which is* $w_{ij} = |\frac{\mathbf{x}_i - \mathbf{m}_k}{\mathbf{st}}|$

where i identifies the i^{th} data vector and j represents the j^{th} training data separation rule vector, m_k is the k^{th} training data set mean vector, $\mathbf{sd}_k$ is the standard deviation vector of training data set k, and the **st** is the standard deviation of testing data.

In testing phase, K set of network outputs are produced by K networks for every input vector. At this point a decision must be made to produce one and only one set of final output vector. The gating network parameter g_{ij} controls the contribution of the j^{th} network output to the final output when the input is the vector $\mathbf{x}_i$. The overall output vector $\mathbf{y}_i$ is computed as follows.

$$\mathbf{y}_i = \sum_{j=1}^{K} g_{ij}\mathbf{f}_j$$

Where the output of the j^{th} network is denoted by $\mathbf{f}_j$. Note that the gating network has K parameters. Computation of g_{ij} consists of two steps. The first step involves the computation of an intermediate parameter, ω_{ij}, based on the rules described above. In our experiment, the input $\mathbf{x}_i$ and the directional vectors $\mathbf{v}_j$ (the shifted directional vectors $\hat{\mathbf{v}}_j$) play a direct role. For example, one may compute $t_{ij} = \mathbf{x}_i^T\mathbf{v}_j$ which measures the similarity between the input and the directional vectors. In the second step, ω_{ij} is applied to the gating function to compute g_{ij}.

5.2.4 Experimental Results

1. Case I Three Clusters

In the 3-cluster experiment, the images in Fig:5.3 are used as the test images and the images in Fig:5.2 are the original image. *Fuzzy* result is shown following the crisp result.

I. Experimental Result 1

First, a crisp selection is used to Image 1 to get the committee machine output. In this case, if the testing data $\mathbf{x}_i$ belong to the expert_j, the expert_j network output will be weighted by $g_{ij} = 1$, otherwise, the expert_j output will be weighted by $g_{ij} = 0$.

This is a process to identify the test vector with one of the directional vectors. In other words, a test vector that is similar to one of the directional vectors in terms of its information content is expected to product the minimum error output by that certain network. Experimental results based on the six cluster rules are shown in Table 5.4. In Table 5.4, (a) shows the number of test vectors which is most similar to one of the directional vector. (b) computes the number of test vectors from (a) which holds minimum absolute error output. (c) computes the ratio of (b) and (a). It is quite likely that the test vector is equally similar (or dissimilar) to all the test vectors. In that case, none of the network outputs will give us any clear indication. The statistics of each method are summarized in Table 5.5.

Based on this experimental results we can see that methods 4-6 produce better outcome.

II. Experimental Result 2

Fuzzy output results are studied in this section. The committee machine consists of a gating network and three expert networks, which are the same as described in section 5.1, During the testing phase, entire testing data is sent to every expert network. Method 1 (refer to section 5.2.3) is used by the gating network to calculate the weight g_{ij} to generate the committee machine outcome. The number of gating network output is same as the number of expert networks.

Table 5.4: Pixel statistics

(a) Number of input vectors in each subset

	method 1	method 2	method 3	method 4	method 5	method 6
expert 1	27445	18294	35962	21928	27557	22010
expert 2	1523	18094	1208	34624	35271	34629
expert 3	35539	28047	27339	7964	1688	7877

(b) Number of input vectors correctly recognized

	method 1	method 2	method 3	method 4	method 5	method 6
expert 1	11800	8207	14715	9255	11906	9301
expert 2	543	6170	394	13066	13227	13068
expert 3	8074	5654	7196	2309	404	2289

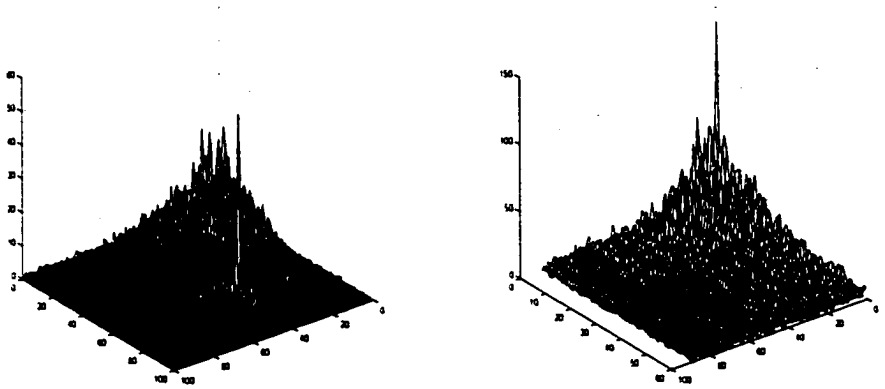
(c) % of input vectors correctly recognized

	method 1	method 2	method 3	method 4	method 5	method 6
expert 1	43.29	44.86	40.92	42.21	42.26	41.04
expert 2	35.65	34.10	32.62	37.74	37.74	31.51
expert 3	22.72	20.16	26.32	28.99	29.06	25.93

Table 5.5: Summary

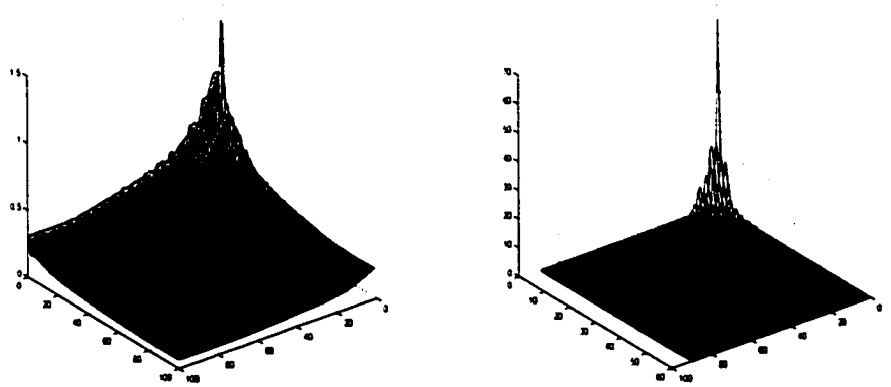
	Total number of data performed minimum error	total number of data have been selected	percent of minimum error pixel (%)	number of unclassified data
method 1	20497	64507	31.77	9
method 2	20031	64435	31.05	81
method 3	22305	64509	34.57	17
method 4	24630	64516	38.18	0
method 5	25537	64516	39.58	0
method 6	24658	64516	38.22	0

First, regular network, static gating network (averaging expert output) and one of dynamic gating network outputs are provided in Figure 5.14 and Figure 5.15 by column first. The restored images from three network output are shown in one page for easy comparison. In a given page, rows 1,2 and 3 show the results achieved through Sigmoid network, RBF network and PPLN network. To get a better insight, we analyze the frequency plots. Their corresponding frequency plots are shown in Figure 5.16 and Figure 5.17 arranged in the same way as the images. The high frequency content is attenuated in a blurred image. The frequency plots of deblurred images will display the high frequency content that has been restored. We compare this with the frequency plots from blurred and original images. See frequency plots in Figure 5.12 for original and test images, Figure 5.13.



(a) Madonna (b) Clock

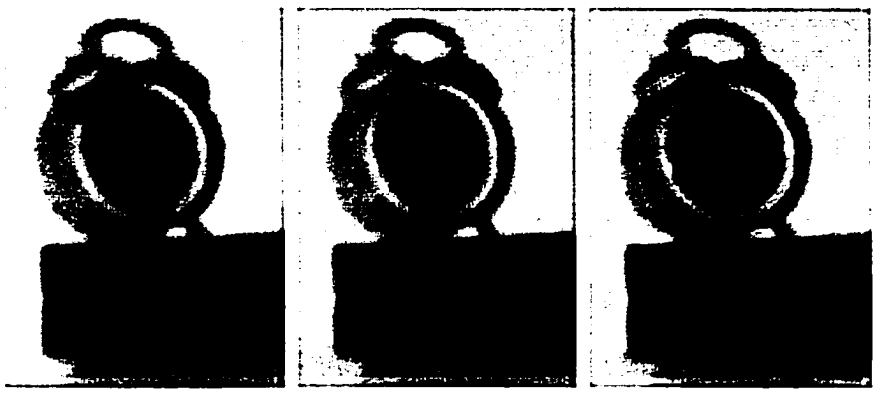
Figure 5.12: Frequency plots for images in Fig:5.2



(a) Madonna (b) Clock

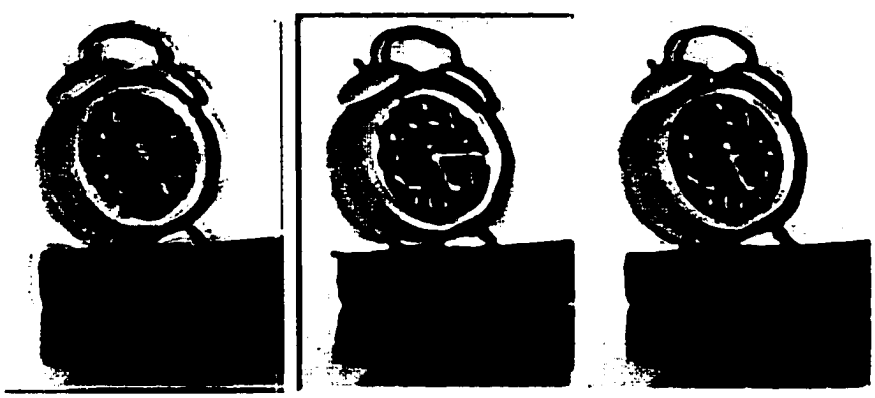
Figure 5.13: Frequency plots for images in Fig:5.3 first and third column

Let us consider the restored images. We notice that better restoration occurs as we move across a row and move down a column. Movement along a row takes us from the use of regular data set to clustering and static gating structure to clustering and dynamic gating structure for the same type of learning system (row 1 is NN etc.). On the other hand, movement down



(i) NN

(ii) RBF



(iii) PPLN

(a) Experiment 1 (b) Experiment 2 (c) Experiment 3

Figure 5.14: Restored clock image

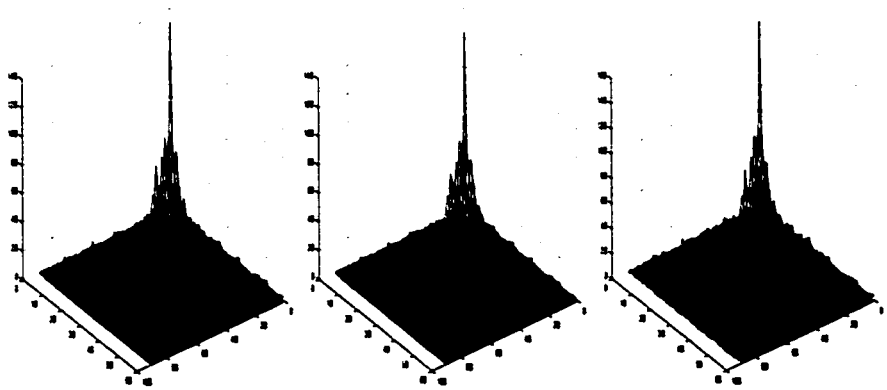
(i) NN

(ii) RBF

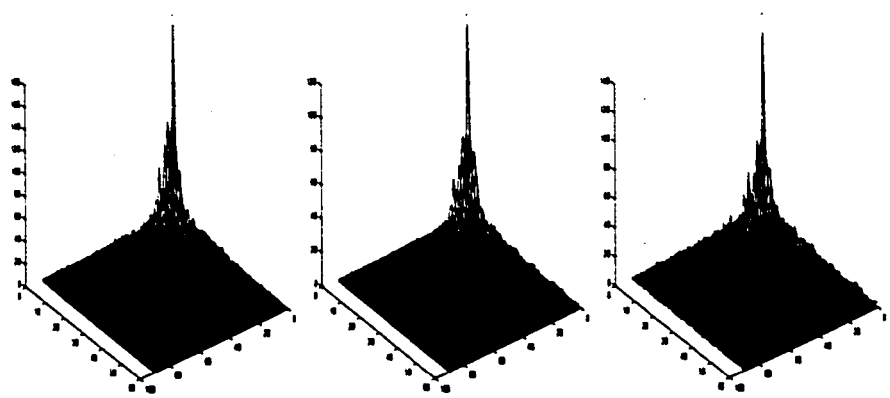
(iii) PPLN

(a) Experiment 1 (b) Experiment 2 (c) Experiment 3

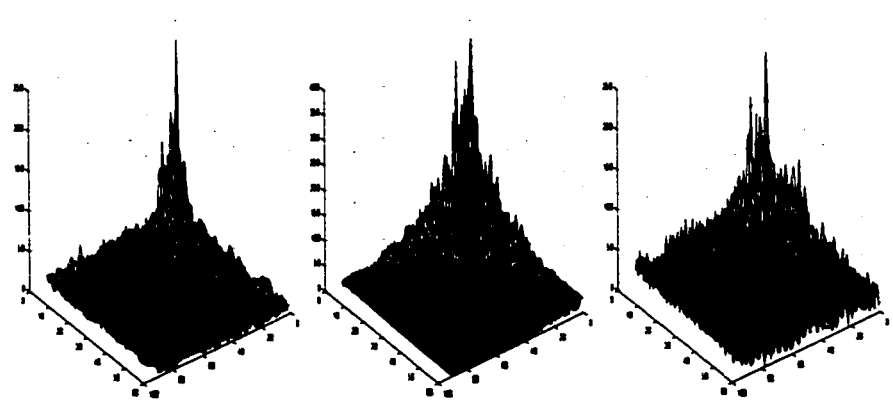
Figure 5.15: Restored Madonna image



(i) NN



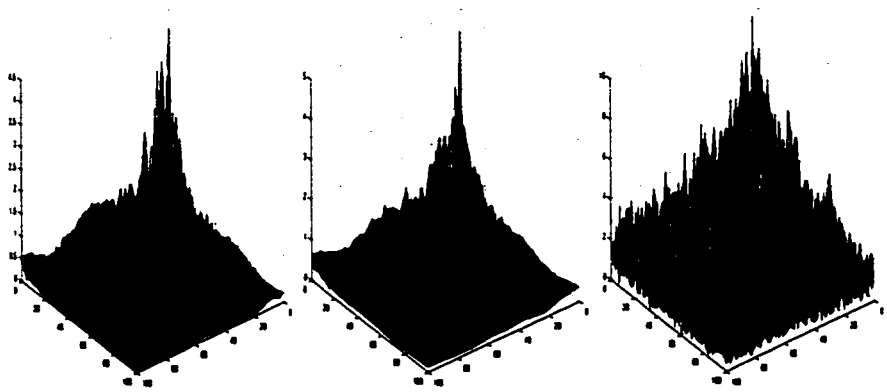
(ii) RBF



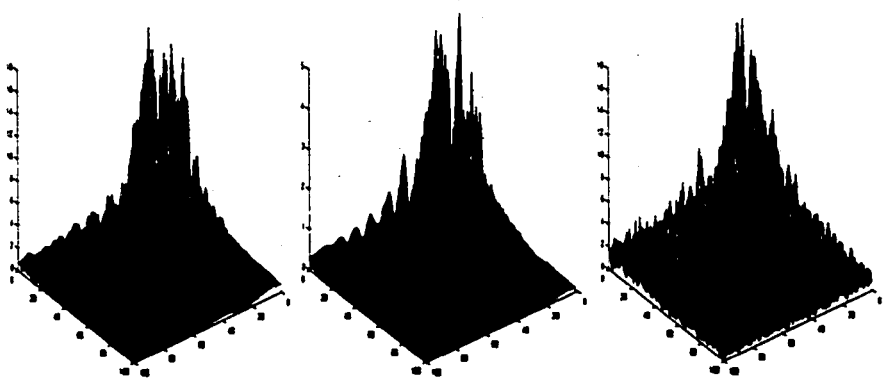
(iii) PPLN

(a) Experiment 1 (b) Experiment 2 (c) Experiment 3

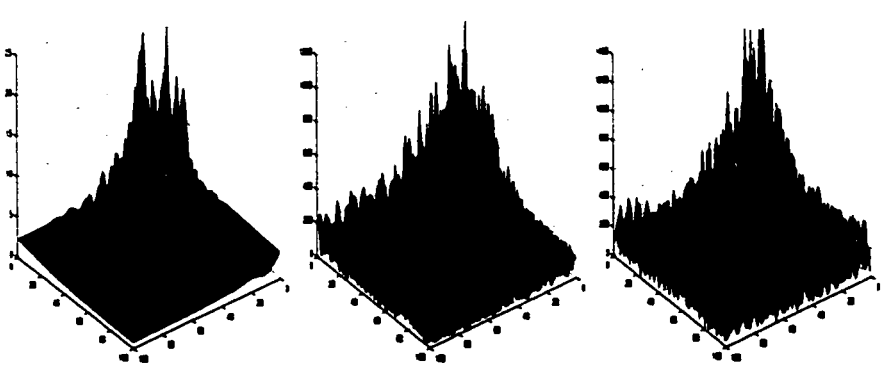
Figure 5.16: Restored clock image



(i) NN



(ii) RBF



(iii) PPLN

(a) Experiment 1 (b) Experiment 2 (c) Experiment 3

Figure 5.17: Restored Madonna image

a column takes us from one type of learning system to another while keeping the network structure fixed. For example, column 2 shows results for clustering and static gating structure for NN, RBF and PPLN. We observe that the PPLN even recovers image information that is not so clear in the original image. Compare the area with a necklace in the original and the PPLN restored Madonna image. It is an interesting finding. The restored images show that a committee of neural networks with PPLN as the learning system and gating with dynamic structure is the best hybrid system.

In addition to the images, we consider the frequency plots (see Figure 5.16 and Figure 5.17) for a better understanding of the restoration process. The plots show the recovery of frequency components in a more clear manner. Notice the front quarter in the frequency plots. If we fix the experiment i.e., focus on a column and consider various learning systems we notice that the output from PPLN shows the most perceptible recovery of frequency components followed by the outputs from RBF and NN. On the other hand, for a given learning system, committee machine with dynamic gating performs the best. We arrive at the same conclusion as the previous paragraph from the analysis of the frequency data.

Next, five committee machines using different gating function. The expertmental results shown in Table 5.6 are the pixel error value from committee machines. Labels (a) to (e) refer to the type of gating function used. It is obvious that function 1,3 and 5 provide better result. Fig:5.20 shows the images from the committee machines. Column 1 and 3 are the output from five committee machines. It also shows that rows (1), (3) and (5) display better images.

The input data although corrupted, contains low frequency information. We show that a linear combination of the gating network output with the input data produces a better image.

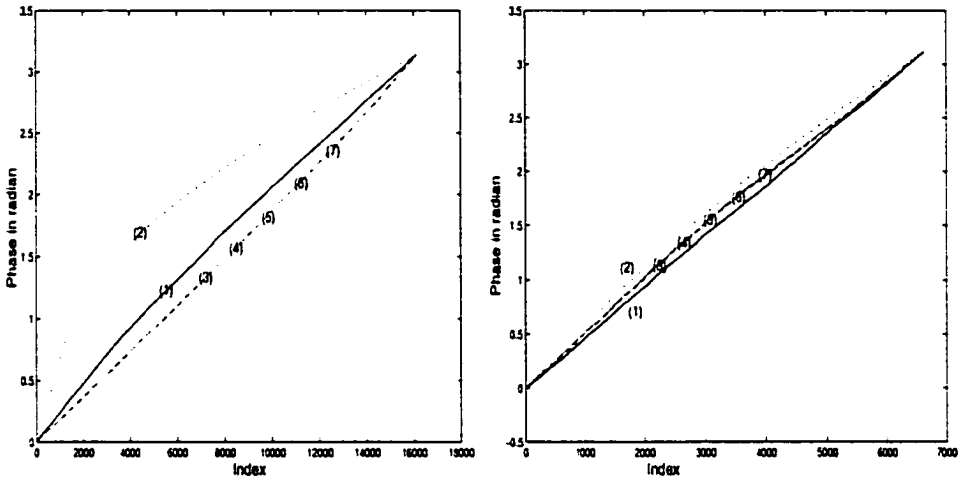
Table 5.6: Output pixel error

	a	b	c	d	e
image 1	0.2816	0.6764	0.2668	1.4338	0.2815
image 2	0.2908	0.7526	0.3080	1.6750	0.2908

See Figure 5.20 (Column 2 and Column 4).

Fig:5.18 shows that the phase of the output produced by the committee machine is closer to the phase of the original images.

In Figure 5.19, we show a plot of randomely chosen pixels from the original, the blurred, the gating network output and the input-gating network combination output images. One can see a clear similarity between the solid curve (original image) and the -o curve (the input combined with the gating network output).



Madonna image Clock image

(1) Original

(2) Blurred

(3) Gating 1

(4) Gating 2

(5) Gating 3

(6) Gating 4

(7) Gating 5

Figure 5.18: 3-cluster gating results on frequency in radian

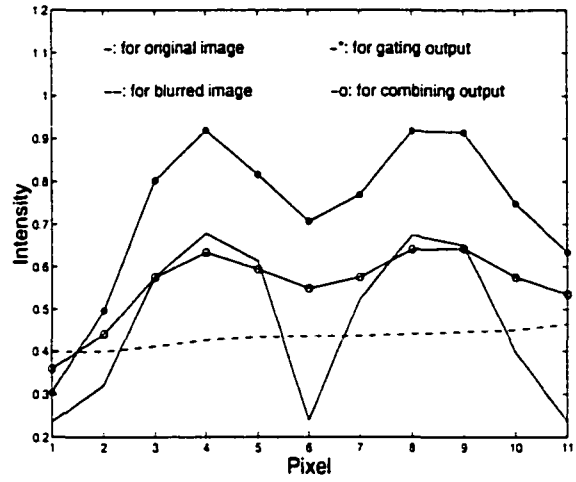


Figure 5.19: A plot of pixel values

III. Case II: Four clusters

A four cluster modular network experimental result is shown in this section. The same cluster rule, which described in section 5.1.4, is used here. The *inner product with directional vector* is used as the gating rule. Five gating functions (see 5.2.3) are applied to the network. The results are shown in Fig:5.21, Fig:5.22, and Fig:5.23 and Table 5.7.

Results that combine the deblurred output with the input signal are shown in Fig:5.21 and Fig:5.22 (second and fourth column). Similar conclusions can be drawn in case of four clusters.

5.2.5 Summary

A *proper* grouping of the data during the training phase has important effect on the network weights. We observe that a mixture of experts with each expert focused on a specific task delivers an image that clearly shows the restored features. Further work with an integrating

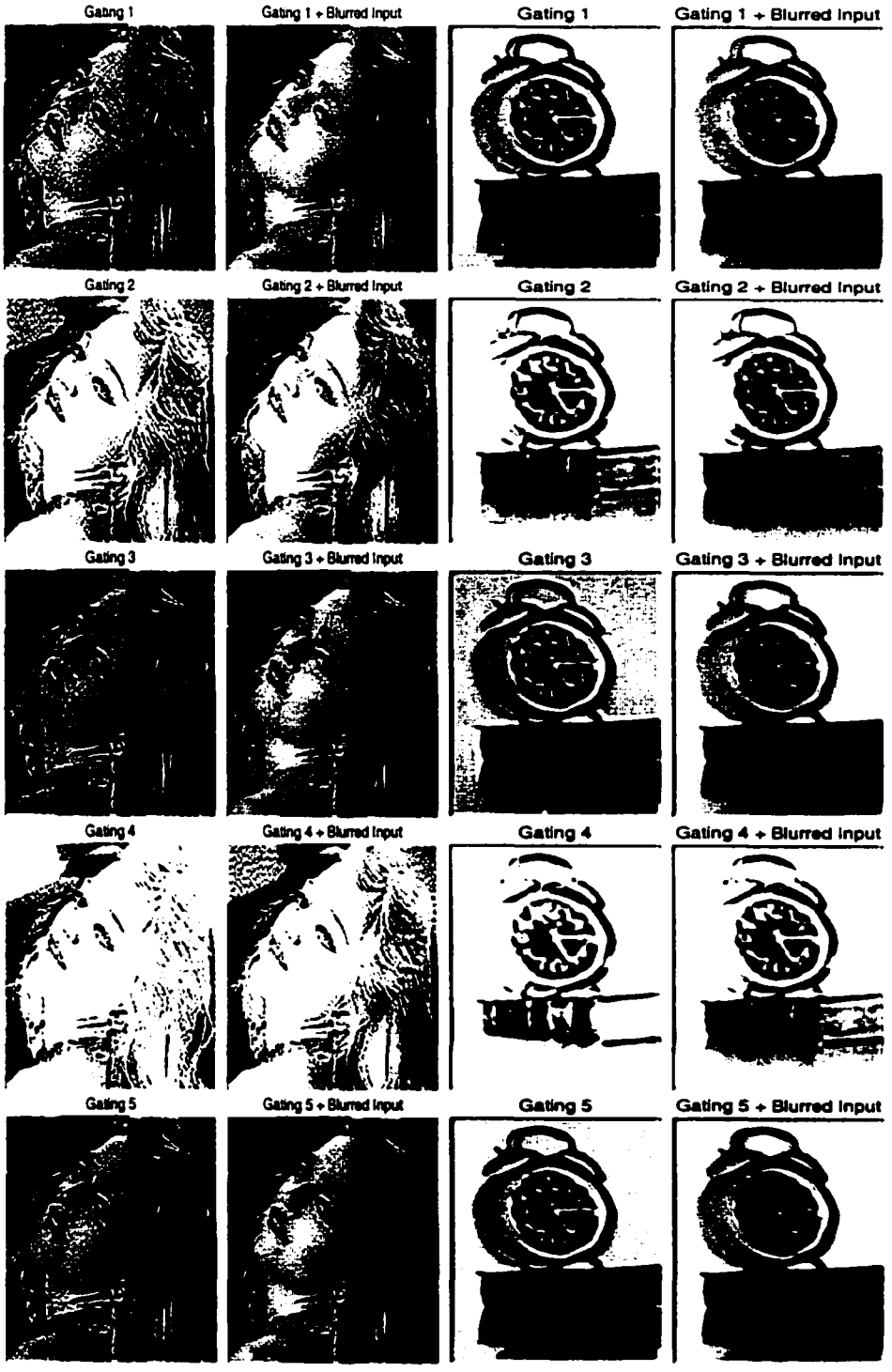


Image 1 Image 2

Figure 5.20: Gating network outputs for Three Cluster Case

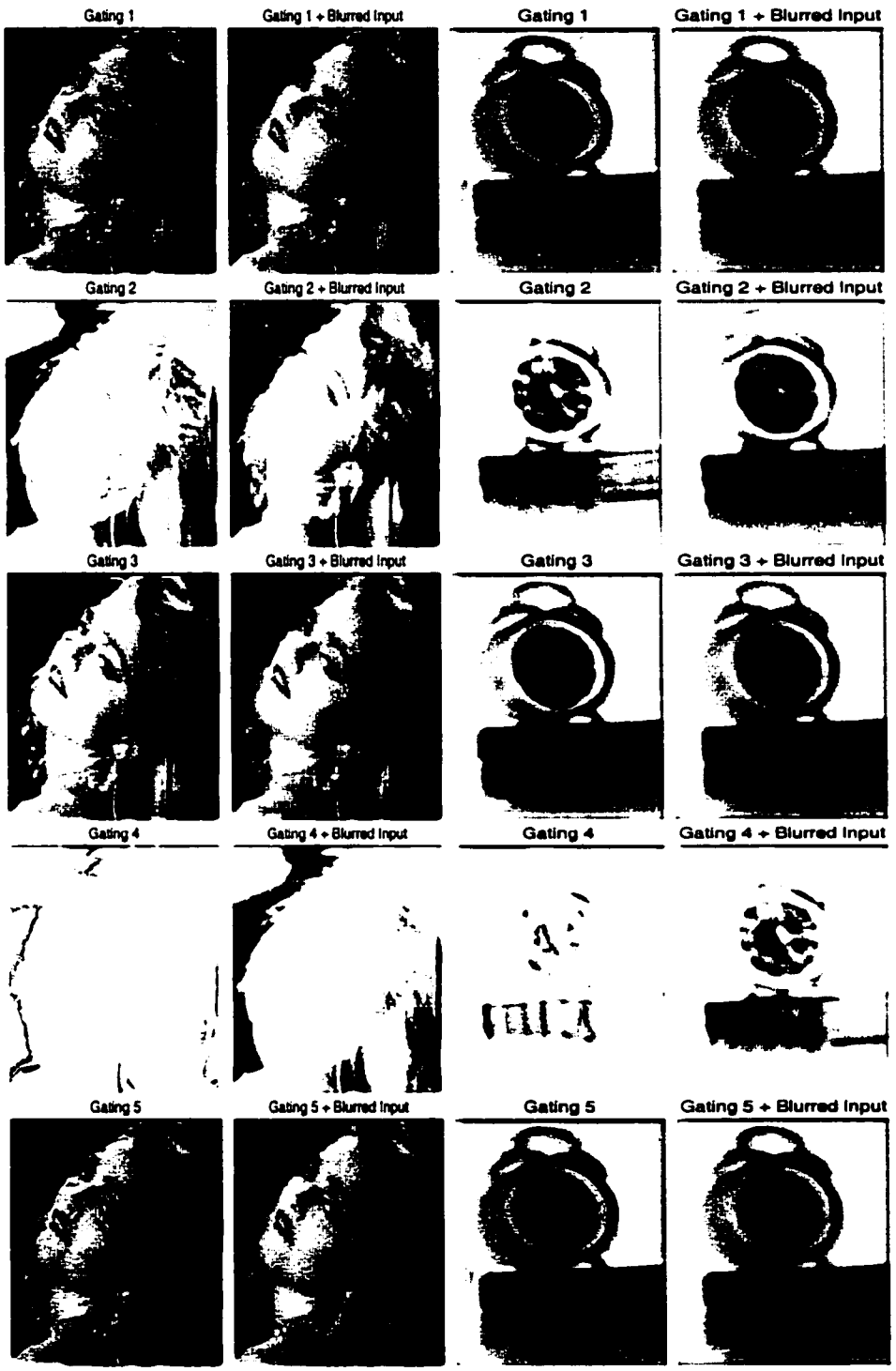
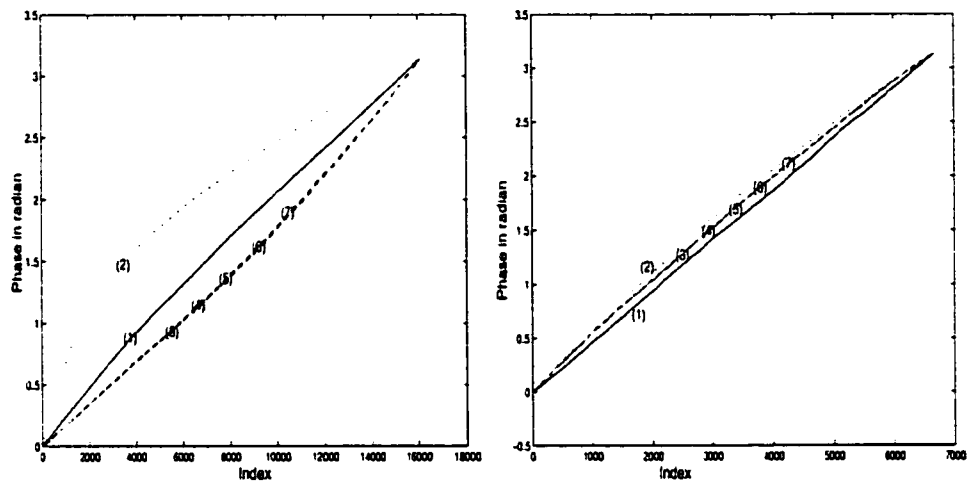


Figure 5.21: Gating network with Four Cluster Case (Heavily blurred test images)

Figure 5.22: Gating network with Four Cluster Case (lighty blurred test images)



Madonna image

Clock image

(1) Original

(2) Blurred

(3) Gating 1

(4) Gating 2

(5) Gating 3

(6) Gating 4

(7) Gating 5

Figure 5.23: Gating network with four cluster case

Table 5.7: Output pixel error

	a	b	c	d	e
image 11	0.0405	0.5396	0.1300	1.5324	0.0405
image 12	0.0730	0.5381	0.1427	1.5093	0.0730
image 21	0.0947	0.5953	0.1277	1.7821	0.0947
image 22	0.1513	0.7173	0.1997	2.0004	0.1514

unit that decides how the outputs of all experts should be combined to produce the over all output is reported in [72, 74].

The resulting images show that gating network function is one of the critical factors that effects the overall performance of the system. Better results are obtained through sum normalization, averaging and softmax functions. We show for the first time that the input data can be used with the gating network output to produce significantly improved images.

The experimental results also show that cluster methods and gating functions play crucial role in improving over all network performance. Of course, selection of number of clusters remains a key issue. One should use knowledge about the problem at hand to make this decision.

From the images and the corresponding frequency plots, (frequency and phase plots), we observe the role of the activation function and the committee of neural networks with static and dynamic gating structures. We keep all other parameters of the three learning networks same while the form of the activation function is increasingly made more complex. We can rank the three learning networks as per their performance on a set of large number of images:

1. Projection Pursuit Learning Network (best performance, most complex activation function)

2. Radial-Basis-Function Network (intermediate performance, activation function with moderate complexity)

3. Back-Propagation Network (worst performance, simplest activation function)

Our second observation is on clustering and the use of the gating structure. A *proper* grouping of the data during the training phase has important effect on the network weights. Such networks show improved performance than that of the corresponding networks which do not receive clustered data. Furthermore, when the role of the input is extended beyond the initial training process and it is allowed to participate in producing the overall output of the network, we observe marked improvement in the system performance. This paper clearly demonstrates the superior performance of the committee of neural networks with dynamic gating structure in deblurring and enhancement of images.

We conclude with a comment on the overall performance of the system. The resulting images are beautifully restored and exceptionally clear. We attribute this to a number of reasons. Note that, the blur function used for training is different from the blur function used in testing. We believe that the following aspects of our design play key role in producing quality restored images.

- A committee of machines cast vote to generate part of the output.

- We use input in reshaping the output of the committee machines. This reduces the effects of some of the overly restored high frequency components.

- The generalization capability of individual network has been achieved through proper training.

Chapter 6

Conclusion and Future works

6.1 Conclusion

We have studied learning systems in the context of problems in image processing. Restoration of images that are out of focus or corrupted with noise is a crucial step in any image processing and computer vision system. Often, it is difficult to model such problems due to unknown or semi-known nature of the degradation source. We have shown that neural network is a viable tool to solve this problem. In this work, we have taken the approach of learning and modeling from data.

The choice of PPLN provided us with the flexibility of selecting the shape of the activation functions. Comparison with other types of learning systems strengthens our belief that activation function plays a crucial role in achieving specific objectives. Moreover, we have shown that selection of activation function should be done with knowledge about the problem domain.

We found that designing learning systems to address specific subtask is extremely useful. The interaction of these sub-task specific learning systems provide fast learning rates, stability, and flexibility for the entire learning system. The proposed committee machine with voting capability and unique contribution from the input signal produced restored images that are of superior quality.

6.2 Future Works

- The training procedure of PPLN consists of a forward growing step and a backward pruning step. Instead of computing the most important hidden neuron based *only* on weight information between the hidden layer and the output layer, which we used in this research, a method to select a hidden neuron will be studied in the future. We select a hidden neuron for removal in the following manner. For each hidden neuron compute the product of its output averaged over one training cycle and the weight that connects it to the output neuron. Remove the hidden neuron which has the minimum product value. We believe that it will be an improvement over the previous method of removing a hidden neuron with least connecting weight. the proposed approach will provide a true measure of contribution for each hidden neuron to the final output.

- One of the major reason for using committee machines is its capability to divide task into sub-tasks. It allows individual committee members to focus on only one sub-case, thus reducing the learning complexity, and increasing the accuracy. Decomposition of the input space not only affects the performance of each individual committee member, but also affects the output of the gating function. A blind cluster of the input data space may not provide much help. We have clustered our data by using the edge

information. In future, we will explore an auto decomposition of the data guided by the characteristics of the problem domain.

- Gating function also plays a critical role. We have proposed five gating functions, each weights individual expert outputs to different degrees in order to provide the final result. The gating function depends on the clustering method used. A more effective combination of cluster method with the gating function will be studied next.

- Signal has its own characteristic in different domains. In committee network, more than one expert network is used to generate the final result. This property makes it possible to use multi-model data. We will explore networks that can handle heterogeneous data.

6.3 Publications Resulting From This Thesis Work

6.3.1 Journal Papers

- M. Basu and M. Su, "Image Smoothing With Exponential Functions," *International Journal of Pattern Recognition and Artificial Intelligence*, Vol. 15, No.4, pp.735-752, 2001.

- M. Su and M. Basu, "A Hybrid Learning System for Image Deblurring," *Pattern Recognition* (in press).

6.3.2 Conference Papers

- M. Su, M. Basu and A. Toure, "Multi-domain Gating Network for Classification of Cancer Cells Using Gene Expression Data," *Proceeding of the International Joint Con-*

ference On Neural Networks, May 2002, Honolulu, Hawaii.

- M. Su and M. Basu, "Mixing Supervised & Unsupervised Learning for Image Deblurring," *Proceeding of Joint Conference On Information Sciences*, March 2002, North Carolina,U.S.A.

- M. Su and M. Basu, "Gating Improves Neural Network Performance," *Proceeding of The International Conference On Neural Networks*, July, Washington D.C. 2001.

- M. Su, M. Basu, "Input Data Clustering to Improve Neural Network Performance," *Proceeding of the International Conference On Neural Networks*, July 2001, Washington D.C.

- M.Basu and M.Su, "Deblurring images using projection pursuit learning network", *Proceeding of The International Conference On Neural Networks* , July 10-16, 1999, Washington,D.C..

Conference Paper (under review)

- M.Su, M. Basu and A. Toure, "Novel Techniques for Analysis of Gene Expression Data for Cancer Classification," *submitted to the 10th International Conference On Intelligent Systems for Molecular Biology, August 3-7, Edmonton, Canada.*

Poster

- M. Su and M. Basu, "Projection Pursuit Learning Network to deblur an Image," *Graduate Center Of City University*, April 2000, New York.

Bibliography

[1] A. Atiya, R.Aiyad and S.Shaheen, "A Practical Gated Expert Network," *Proc. 1998 IEEE Int. Joint Conf. Neural Network(IJCNN)*, Vol. I, AK, pp.419-424, May 1998.

[2] J. Anderson, "An Introduction to Neural Networks", *MIT Press*, 1995

[3] A. R. Barron and R. L. Barron,"Statistical learning networks: A unifying view", in *Proceedings 20th Symposium Interface*, Ed Wegman, Ed. Washington, D.C.: American Statist. Assoc., pp. 192-203, 1988.

[4] M. Basu and M. Su, "Image Smoothing With Exponential Functions," *International Journal of Pattern Recognition and Artificial Intelligence*, Vol. 15, No.4, pp.735-752, 2001.

[5] M.Basu and M.Su, "Deblurring images using projection pursuit learning network", *Proc. Intl. Conf. on Neural Networks* , July 10-16, 1999, Washington,D.C..

[6] S. Basu, C. A. Micchelli & P. Olsen. "Power exponential densities for the training and classification of acoustic feature vectors in speech recognition," *IBM Research Report CS/MATH No. RC21403(96649)*, February, 1999.

[7] T.A. Bale and K. Ahmad, "Simulation of Quantification Abilities Using a Modular Neural Network Approach", http://citeseer.nj.net.com/123644.html

[8] N. Belkin, Kantor P., Fox E., and Shaw J., "Combining the evidence of multiple query representations for information retrieval," *Information Processing & Management*, Vol.31, No.3, pp.431-448, 1995

[9] T.G. Bever, "Broca and Lashley were right: Cerebral dominance is an accident of growth," *In D.Kaplan and N. Chomsky (Eds.), it Biology and Language. Cambridge, MA: MIT Press*, 1980.

[10] D.S. Broomhead, and D. Lowe, "Multivariable functional interpolation and adaptive networks," *Complex Systems*, vol.2, pp.321-355, 1988.

[11] H.Chen, "Estimation of a projection-pursuit type regression model", *Annals Statis.*, vol. 17, no 2, pp.453-555, 1989.

[12] H. Chen, R.W. Liu, "Adaptive distributed orthogonzlization processing for principal components analysis," *International Conference on Acoustics, Speech, and Signal Processing*, Vol.2, pp.293-296, 1992, San Francisco.

[13] P. Diaconis, D. Freedman, "Asymptotics of graphical projection pursuit," *Ann. Statist.* Vol. 12, pp. 793-815, 1984.

[14] P. Diaconis, M. Shashahani, " On nonlinear function of linear combinations." *SLAM J. Sci. statist. Comput.* Vol.5, no.1, pp. 175-191, 1984.

[15] D. L.Donoho and I. Johnstone, "Projection-based approximation and a duality with kernel methods", *Annals Statist.*, vol. 17, no. 1, pp. 58-106, 1989.

[16] H. Drucker and C. Cortes, "Boosting decision trees," in *Advances in Neural Information Processing Systems 8,* D.S. Touretzky, M.C. Mozer, and M.E. Hasselmo, Eds. Cambridge, MA: MIT Press, pp.479-485, 1996.

[17] J. Fill, I.Johnstone, " On Projection Pursuit Measures of multivariate location and dispersion," *Ann. Statist.* Vol. 12, pp.127-141, 1984

[18] T.E. Flick, L.K.Jones, R.G.Priest and C.Herman, "Pattern Classification Using Projection Pursuit", *Pattern Recognition*, vol. 23, no. 12, pp. 1367-1376, 1990.

[19] Y. Freund and R.E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," in *Proc. 2nd European Conf. Comput. Learning Theory,*, pp.23-37, March 1995.

[20] J.H. Friedman, J.W. Tukey, "A Projection Pursuit algorithm for explorithmata analysis," *IEEEE Trans. Comput.* Vol. c-23, No.9, pp. 881-890, Sept. 1974.

[21] J.H. Friedman and W. Stuetzle, "Projection pursuit regression", *J. Amer. Statist. Assoc.*, Vol. 76, no 376, pp. 817-823, 1981.

[22] J. H. Friedman, "Exploratory projection pursuit," *J. Amer. Statist. Assoc.* Vol.82, pp.249-266, 1987

[23] H. C. Fu, Y.P. Lee, C.C. Chiang and H.T. Pao, "Divide-and Conquer Learning and Modular Perceptron Networks," *IEEE Trans. on Neural Networks,* Vol.12, No.2, pp.250-263, March 2001.

[24] N. Geschwind and A.M. Galaburda, "Cerebral Lateralization: Biological Mechanisms, Associations, and Pathology," *Cambridge, MA: MIT Press,* 1987.

[25] G.H. Golub, and C.G. Van Loan, "Matrix Computations," *3rd edition, Baltimore: Johns Hopkins University Press*, 1996.

[26] R. C. Gonzalez & P. Wintz. *Digital Image processing.* Reading, M.A.: Addison-Wesley, 1987.

[27] A. K. Jain. *Fundamentals of Digital Image Processing.* Englewood Cliffs, N.J.: Prentice Hall, 1989.

[28] P. Hall, "On Polynomial-based Projection Indices For Exploratory Projection Pursuit," *The Annals of Statistics,* Vol. 17, No.2, pp.589-605, 1989.

[29] Peter Hall, " On Projection Pursuit Regression," *The Annals of Statistics,* Vol. 17, No. 2, pp. 573-588, 1989.

[30] J.B. Hampshire, A.H. Waibel, "The meta-pi network: Building distributed knowledge representations for robust multisource pattern recognition," *IEEE Trans Patt Analysis & Mach Intell,* Vol.14(7), pp715-769, 1992

[31] P. J. B. Hancock, L. S. Smith and W. A. Phillips, "A biologically supported error-correcting learning rule", *Neural Computation,* 3, pp. 201-212, 1991.

[32] P.J. Huber, "Projection pursuit", *Ann. Statist,* Vol.13, pp.435-525, 1985.

[33] D.A. Hull, J.O. Pedersen and H. Schutze, "Method Combination For Document Filtering," *SIGIR'96,* Zurich, Switzerland ACM, pp.279-287, 1996.

[34] M.M. Van Hulle, "Nonparametric regression analysis achieved with topographic maps developed in combination with projection pursuit learning", *IEEE Trans. Signal Processing,* Vol. 45, no. 11, pp. 2663-2672, 1997.

[35] R. A. Hummel, B. B. Kimia and S. W. Zucker, "Deblurring the Gaussian blur", *Comput. Vision, Graphics & Image Process.*, Vol. 38, pp. 66-80, 1987.

[36] J-N. Hwang, S-R. Lay, M. Maechler, R. D. Martin & J. Schimert. "Regression Modeling in Back-Propagation and Projection Pursuit Learning", *IEEE Transactions on Neural Networks*, Vol. 5. no. 3, pp 342-353, 1994.

[37] J-N. Hwang, S-S. You, S-R. Lay & I-C Jou. "The Cascade-Correlation Learning: A Projection Pursuit Learning Perspective," *IEEE Trans. on Neural Networks*, Vol. 7, No. 2, pp.278-289 , 1996.

[38] N. Intrator, "Exploratory feature extraction in speech signals ", in *Advances in Neural Information Processing Systems 3*, R. P. Lippman, J. E. Moody and D. S. Touretzky, Eds. San Mateo, CA: Morgan Kauffman, 1991.

[39] N. Intrator, "Feature extraction using an unsupervised neural network", *Neural Comput.*, vol. 4, pp.98-107, 1992.

[40] R.A. Jacobs, M.I. Jordan, A.G. Barto. "Task decomposition through competition in a modular connectionist architecture: The What and Where vision tasks," *Cog Sci* , Vol. 15, pp219-250, 1991

[41] R.A. Jacobs, M.I. Jordan, S.J. Nowlan and G.E. Hinton, "Adaptive Mixtures of Local Experts," *Neural Computation*, Vol. 3, pp79-87, 1991.

[42] R. A. Jacobs, "Computational Studies of the Development of Functionally Specialized Neural Modules," *Trends in Cognitive Sciences*, No.3, pp. 31-38, 1999.

[43] W. James, The principles of psychology, New York: Holt, 1890, pp. 403-404.

[44] L. Jones, "On a conjecture of Huber concerning the convergence of projection pursuit regression", *Annals Statist.*, vol. 15, pp. 880-882, 1987.

[45] M.I. Jordan and L. Xu, "Convergence results for the EM approach to Mixtures of Experts Architectures," *Neural Networks*, Vol.8, pp.1409-1431, 1995

[46] L. M. Kennedy. *Experiments in image enhancement using biological and artificial neural networks.* Ph. D. Thesis, City University of New York, New York, 1998.

[47] L. M. Kennedy & M. Basu. "Image enhancement using a human visual system model," *Pattern Recognition*, Vol. 30, No. 12, pp. 2001-2014, 1997

[48] L. M. Kennedy & M. Basu. "A Gaussian derivative operator for authentic edge detection and accurate edge localization," *International Journal of Pattern Recognition and Artificial Intelligence*, Vol. 13, No. 3, 1999.

[49] K. Reinert, J. Stoye and W. Torsten, "Combining Divide-and-Conquer, the A*-algorithm, and Successive Realignment Approaches to Speed up Multiple Sequence Alignment," *www.bioinfo.de/isb/gcb99/talks/reinert/main.html*

[50] S.M. Kosslyn, "Seeing and imagining in the cerebral hemishperes: A computational approach," *Psychological Review*, No. 94, pp.148-175, 1987

[51] J.B. Kruskal, Toward a practical method which helps uncover the structure of a set of multivariate observations by finding a linear transformation which optimizes a new "index of condensation," *Statistical Computation* (R.C. Milton and J.A. Nelder, eds.) pp.427-440, Academic, New York, 1969.

[52] J.B.Kruskal, "Linear transformation of multivariate data to reveal clustering," *Multidimensional Scaling: Theory and Application in the Behavioural Sciences, I. Theory*(R.N.Shepard, A.K.Romney and S.B. Nerlove, ed) pp.181-191. Seminar, New York.

[53] T-Y Kwok and D-Y Yeung, "Use of bias term in projection pursuit learning improves approximation and convergence properties," *IEEE Trans. Neural Networks*, vol. 7, no. 5, pp. 1168-1183, 1996.

[54] R.P. Lippmann, "Pattern classification using neural networks," *IEEE Communications Magazine,* Vol.27, pp47-64, 1989

[55] D. Lowe, "Adaptive radial basis function nonlinearities, and the problem of generalisation," *First IEE International Conference on Artificial Neural Network*, London, pp171-175, 1989.

[56] D. Lowe, "What have neural networks to offer statistical pattern processing?" *Processings of the SPIE Conference on Adaptive Signal Processing*, San Diego, CA. pp460-471, 1991.

[57] M.Maehler, D.Martin, J.Schimert, M.Csoppenszky, and J.N.Hwang, "Projection Pursuit Learning Networks for Regression," *Proc. 2nd Int. IEEE Conf. Tools Artificial Intell.*, pp. 350-358, Nov. 1990.

[58] J.B. Martens, "Deblurring digital images by means of polynomial transforms," *CVGIP*, Vol. 50, pp. 157-176, 1990.

[59] J. Moody, C.J. Darken, "Fast learning in networks of locally-tuned processing units," *Neural Computation*, Vol.1, pp.281-294 1989

[60] J.Murr, R.H. Phaf and G.Wolters, "CALM: Categorizing and learning modele," *Neural Networks*, Vol.5, pp.55-82, 1992.

[61] D.N. Osherson, S. Weinstein and M. Stoli, "Modular learning," *Computational Neuroscience, E.L. schwartz, ed., Cambridge, MA:MIT press*, pp.369-377, 1990.

[62] Simon Haykin, "Neural Networks - A Comprehensive Foundation," second edition, *Master University, Hamilton, Ontario, Canada, Prentice-Hall, Inc.*, 1999.

[63] H. R. Rabiee, R. L. Kashyap and S. R. Safavian, "Multiresolution segmentation-based image coding with hierarchical data structures," *Proc. ICASSP*, vol. 4, pp.1870-1873, 1996.

[64] D.Reilly, L.N.Cooper and C.Elbaum, "A neural model for category learning," *Biological Cybernetics*, Vol.45, pp.35-41, 1982.

[65] E. Ronco, H. Gollee and P. Gawthrop, "Modular Neural Networks and Self-Decomposition," *Technical Report: CSC-96012*, ericr, peterg@mech.gla.ac.uk.

[66] S. R. Safavian, H. R. Rabiee, M. Fardanesh and R. L. Kashyap, "Low bit rate image compression with orthogonal projection pursuit neural networks," *Proc. IEEE Intl. Conf. Neural Networks*, vol. 3, pp.1518-1522, 1997.

[67] S. R. Safavian, H. R. Rabiee and M. Fardanesh, "Projection pursuit image compression with variable block size segmentation," *IEEE Signal Processing Letters*, vol. 4, no. 5, pp.117-120, 1997.

[68] T. Saracevic and P. kantor, "A study of information seeking and retriving III: Searchers, searches, overlap," *Journal of the American Society for Imformation Science*, 39(3), pp.197-216, 1996.

[69] R. Schapire, "The strength of weak learnability," *Machine Learning,* Vol.5, No.2, pp.197-227, 1990.

[70] C.J.Stone, "Additive regression and other nonparametric models," *Ann. Statist,* Vol 13, pp.689-705, 1985.

[71] M. Su, "Vector Qualization Comparing Fuzzy and Crisp Algorithm for Codebook Design," *Master thesis,* City College of New York, New York, 1996

[72] M. Su and M. Basu, "Gating Improves Neural Network Performance," *Proc. of the IJCNN, July, Washington D.C. 2001.*

[73] M. Su, M. Basu, "Input Data Clustering to Improve Neural Network Performance," *Proc. of the IJNCC, July, Washington D.C. 2001.*

[74] M. Su and M. Basu, "A Hybrid Learning System for Image Deblurring," *accepted to the Pattern Recognition Journal.*

[75] P. Switzer, "Numerical classification," *Geostatistics*(D.F. merian, ed.) pp31-43. Plenum, New York, 1970.

[76] P. Switzer and R.M. Wright, "Numerical classification applied to certain Jamaican eocene nummulitids," *math. Geol.* Vol.3, pp297-311, 1971.

[77] M. Vairy and Y.V. Venkatesh, "Deblurring Gaussian blur using a wavelet array transform," *Pattern Recognition,* Vol.28, No.7, pp.965-976, 1995.

[78] F.C. Peng, R.A. Jacobs and M.A. Tanner, "Bayesian Inference in Mixtures-of-Experts and Hierarchical Mixtures-of-Experts Models With an Application to Speech Recognition," *Journal of the American Statistical Association,* Vol.91, pp.953-960, 1996.

[79] T. Poggio and F. Girosi,"Networks for approximation and learning," *Proceeding of the IEEE*, Vol.78, pp.1481-1497, 1990.

[80] *S-Plus - Guide to Statistical and Mathematical Analysis*, Version 3.3, StatSci Division MathSoft, Inc. Seattle, Washington.

[81] R. A. Schalkoff. *Digital Image processing & Computer Vision.* New York, N.Y.:Wiley, 1989.

[82] "Global Stock Index Forecasting Using Multiple Generalized Regression Neural Networks with a Gating Network," *http::://www.umr.edu/ sesl/Global/GlobalMain.htm*

[83] H. Tang and L. W. Cahill, " A new approach for the restoration of noisy blurred images," *Proc. IEEE Intl. Symp. Circuits and Systems*, Vol. 1, pp. 520-523, 1991.

[84] M. Vairy and Y. V. Venkatesh, "Deblurring Gaussian blur using a wavelet array transform," *Pattern Recognition*, Vol. 28, no.7, pp. 965-976, 1995.

[85] A. R. Weeks,Jr. "Fundamentals of Electronic Image Processing," SPIE Optical Engineering Press, 1996.

[86] H.J. Yin and N.M. Allinson, "Self-Organizing Mixture Networks for Probability Density Estimation," *IEEE TRans. On Neural Networks*, pp.405-411, Vol.12, No.2, March 2001.

[87] Yocheved Dotan, Nathan Intrator, " Multimodality Exploration by an Unsupervised Projection Pursuit Neural Network," *IEEE Trans. On Neural Networks*, Vol. 9, No. 3, May 1998.

[88] Y. Zhao and C.G.Atkeson, "Implementing projection Pursuit Learning," *IEEE Trans. Neural Networks*, vol. 7, no. 2, pp. 362-373, 1996.

[89] S.C. Zhu & D. Mumford. "GRADE: Gibbs reaction and diffusion equation," *IEEE Trans. PAMI*, Vol. 19, No. 11, 1997.

[90] S.C. Zhu & D. Mumford. "Learning generic prior models for visual computation," *Int'l. Conf. on Computer Vision and Pattern Recognition*, 1997.